A Stabilized FP16 Mixed-Precision Solver

Issaku Kanamori, Hideo Matsufuru, Tasumi Aoyama,
Kazuyuki Kanaya, Yusuke Namekawa, Hidekatsu Nemura, Keigo Nitadori
(Bridge++ Project)

Lattice 2026
July 27, 2026, University of Maryland

RIKEN
Center for
Computational Science

Introduction

Trends of supercomputers

- Flop to Bytes: data movement is expensive
  Lattice QCD is memory bound application
- Demand from AI application: strong supports to smaller data types (FP16/BF16/FP8 etc.)

$\Rightarrow$utilizing low precision arithmetics (i.e., small data type) is important

FP16: available on A64FX (Supercomputer Fugaku)

- pros: 2 times faster than FP32 (4 $\times$ than FP64), reduces data transfer
- cons: solver may not converge$\Rightarrow$ A stabilized solver algorithm

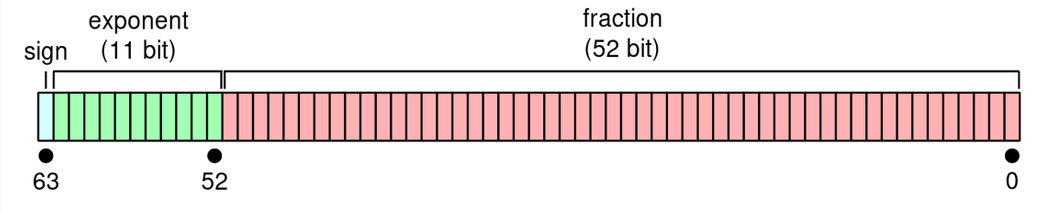
Ref: I.K. et al., SCA/HPCAsiaWS '26 [DOI:10.1145/3784828.3785398]

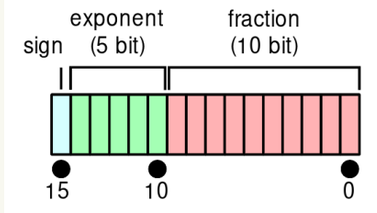
Contents

- Algorithm
- Benchmark with almost physical point configuration

IEEE FP16 and implementation

fraction 10(+1)bits, exponent 5 bits (3-decimal digits、$6.1 \times 10^{-5} - 6.6 \times 10^{4}$)
the exponent is only 5 bit (cf. BF16: 8 bit; A64FX has only FP16)





from wikipedia

- code set: Bridge++ 

  S. Ueda et al., J. Phys. Conf. Ser. 523, 012046 (2014); Y. Akahoshi, J.Phys.Conf.Ser. 2207, 1, 012053 (2022)
  Implementation for Fugaku: T. Aoyama et al., PoS LATTICE2022 284 (2023) [Lattice2022]
- SIMD tuning with Arm C-Language Extension (ACLE)
  lattice point dof in SIMD : `site = (site % 32) + 32 * (site/32)`
- types `_Float16` (C11 standard) and `svfloat16_t` (ACLE)
- inner products and norms: calculate after converting FP32

Iterative refinement (or preconditioned Richardson iterations)

we want: x s.t. $Ax = b$ $\quad A = 1 - D_{ee}^{-1} D_{eo} D_{oo}^{-1} D_{oe}$
we know an approximate solution: x_n (approx. x)
$x_n + A^{-1} r$ is the solution with $r = b - Ax_n$ $\quad A(x_n + A^{-1} r) = Ax_n + r = b$

- even if the cost of $A^{-1} r$ high, we assume that an approx estimate A_{app}^{-1} is easy to obtain
- input: x_0

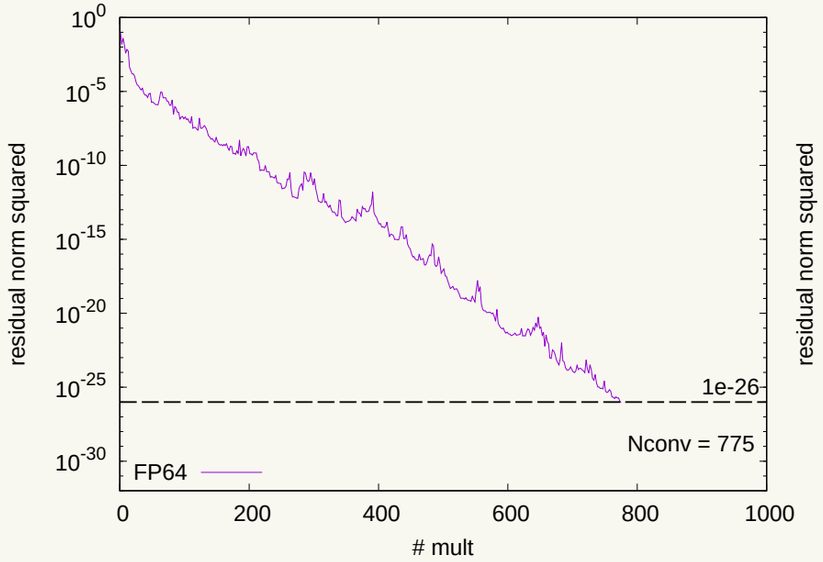
---

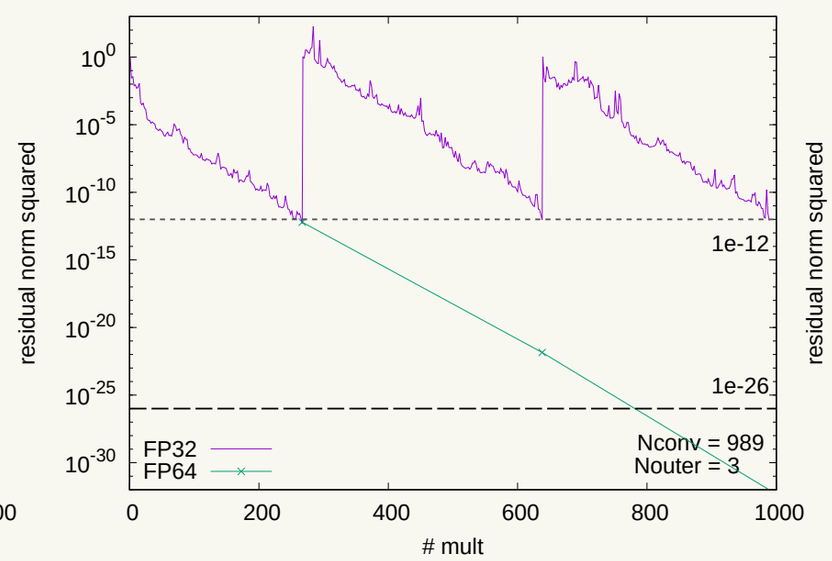
1 $r_0 := b - Ax_0$
2 $i = 0$
3 **while** $|r_i|$ *is not small enough* **do**
4 $\quad x_{i+1} := x_i + A_{\text{app}}^{-1} r_i$
5 $\quad r_{i+1} := b - Ax_{i+1}$
6 $\quad i := i + 1$

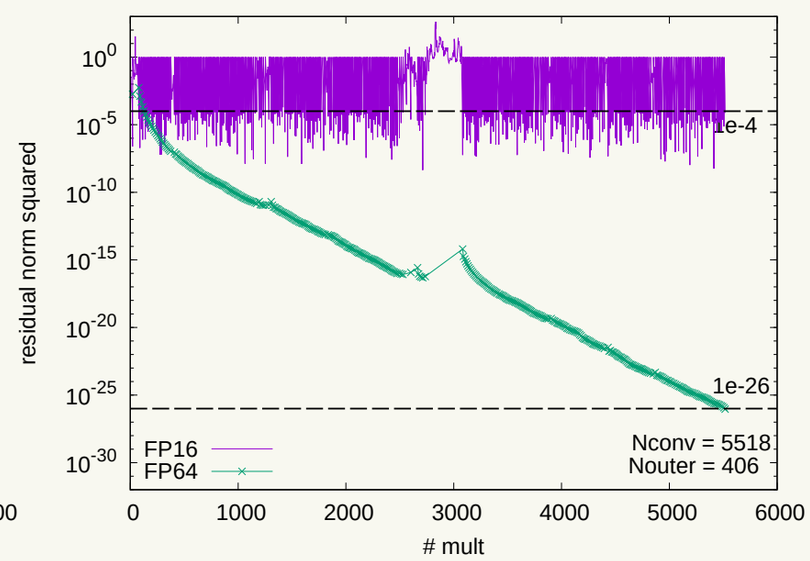
We apply low(half)-precision BiCGStab solver for A_{app}^{-1}

convergence behavior （Wilson Fermion）

Fugaku, 16 nodes ($32 \times 32 \times 32 \times 64$ lattice; a test configuration included in Bridge++)







	double	w/ FP32	w/ FP16
iteration counts	1	1.2	7.2
elapsed time	1	0.70	3.0
outer iter	—	3	406

- mixed with FP32 is the fastest
- need some care for FP16.

eigenvalues are complex; $|\lambda_{\max}|/|\lambda_{\min}| = 1.22/0.069 = 17.7$

Improved Algorithm

Why is FP16 case so slow?

$r_0 := b - Ax_0$
$i = 0$
while $|r_i|$ *is not small enough* **do**
    $x_{i+1} := x_i + A_{\text{app}}^{-1} r_i$
    $r_{i+1} := b - Ax_{i+1}$
    $i := i + 1$

Why is FP16 case so slow?

$r_0 := b - Ax_0$
$i = 0$
while $|r_i|$ *is not small enough* **do**
  $x_{i+1} := x_i + A_{\text{app}}^{-1} r_i$
  $r_{i+1} := b - Ax_{i+1}$
  $i := i + 1$

$x = 0,\ r = b$
while $|r|$ *is not small enough* **do**
  $s = |r|,\ y = r/s$ // $|y| = 1$
  solve low precision system: $\tilde{A}\tilde{t} = \text{low\_prec}(y)$ for $\tilde{t}$
  $x := x + \frac{1}{s}\text{double}(\tilde{t})$
  $r := b - Ax$

r_i become smaller as the iterations develop, so we in practice we use this with $|y| = 1$

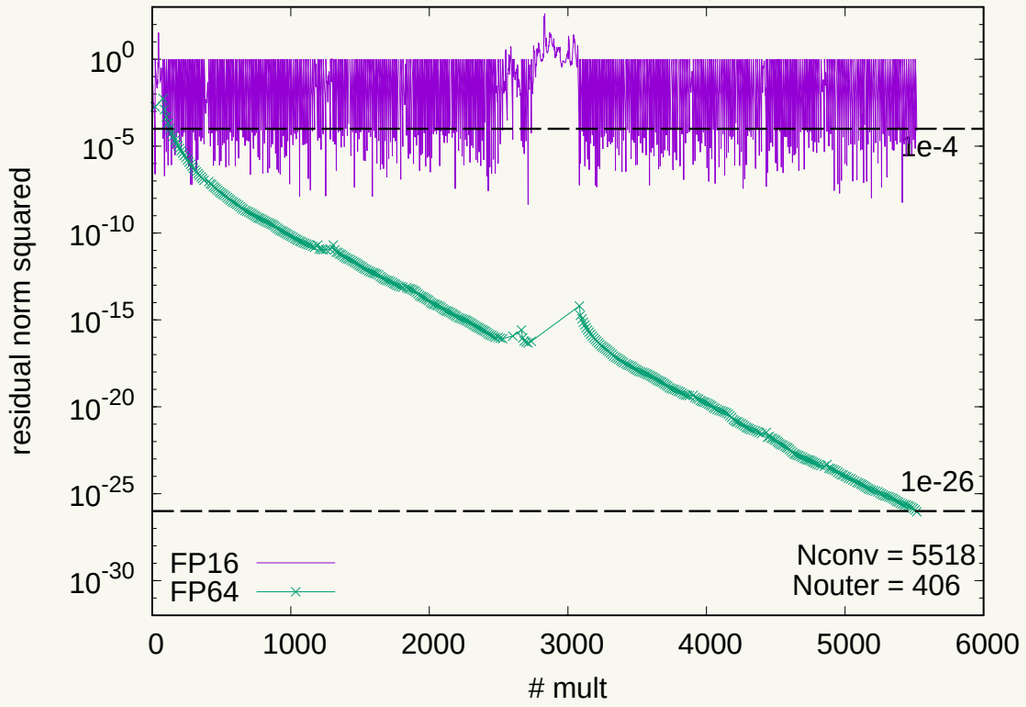
- If y is a random vector with N elements, each element is $\sim 1/\sqrt{N}$.
- For $32^3 \times 64$ lattice, $N = 25165824$ and $1/\sqrt{N} \sim 2 \times 10^{-4}$
- smallest value in FP16: 6×10^{-5}, lots of underflow, maybe?

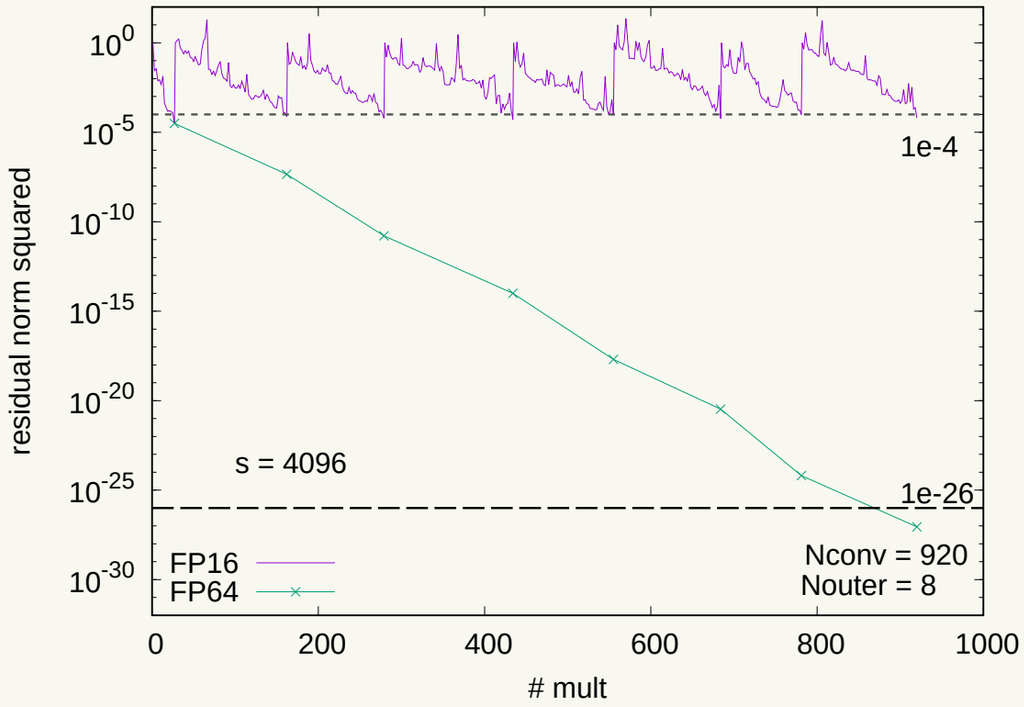
new iterative refinement

1 $x = 0$, $r = b$, $s =$ given
2 **while** $|r|$ *is not small enough* **do**
3 rescale the residual vector: $y = \alpha r$ s.t. $\boxed{|y| = s}$
4 $\boxed{\tilde{y} = \text{low\_prec}(y)}$
5 $\hat{s} = |\text{single}(\tilde{y})|$
6 solve low precision system: $\boxed{\tilde{A}\tilde{t} = \tilde{y}}$ for $\tilde{t}$
7 $x := x + \frac{1}{\hat{s}}\text{double}(\tilde{t})$
8 $r := b - Ax$

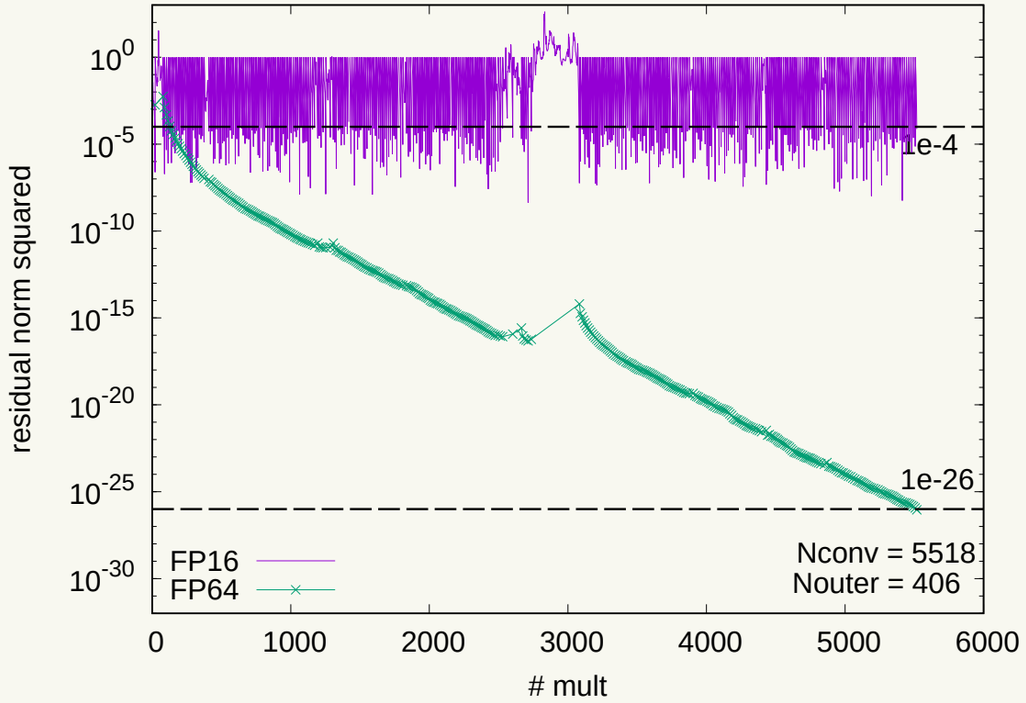
- s is a tunable parameter, assumed large
- because of rounding error, $|y| \neq |\text{low\_prec}(y)|$: we recalculated value $1/\hat{s}$

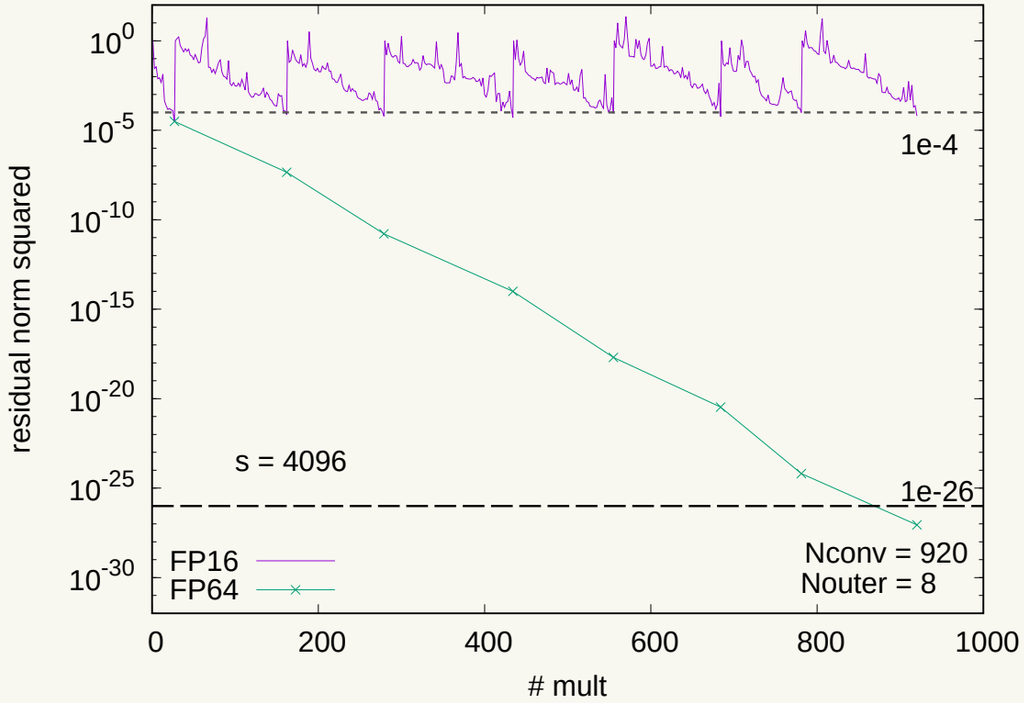
Wow!　Mat-vec mult: 5518 → 920

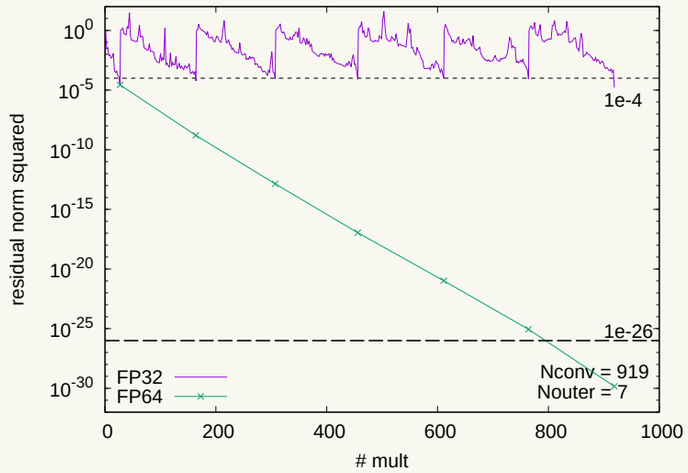




Wow! Mat-vec mult: 5518 → 920







mixed prec. with single prec., tolerance for low prec. solver is the same as FP16 case.

Further improvements

Inside the BiCGStab solver:

- the residual vector (and working vectors) may have the same underflow problem
- the solution vector may have overflow: $A^{-1}|\lambda_1\rangle = \frac{1}{\lambda_1}|\lambda_1\rangle$, $|\lambda_1| \ll 1$

BiCGStab: improved

1 $x = x_0$, $r = b - Ax$, σ = given, σ_λ = given
2 $r^* = r$, $p = 0$, $v = 0$, $\rho' = |r|^2$
3 $\omega' = \alpha' = \gamma' = 1$
4 **while** $|r|/\gamma'$ *is not small enough* **do**
5 $\rho = (r^*, r)$, $\beta = \frac{\rho\alpha'}{\rho'\omega'}$
6 $p := \beta(p - \omega' v) + r$
7 $v = Ap$, $\alpha = \frac{\rho}{(r^*, v)}$
8 $r := r - \alpha v$, $t = Ar$
9 $\omega = \frac{(t,r)}{(t,t)}$, $\hat{\omega} = \frac{|(t,r)|}{\sqrt{(t,t)(r,r)}}$, if$(\hat{\omega} < \omega_0)$ $\omega := \omega\omega_0/\hat{\omega}$
10 $x := x + \frac{\lambda'}{\gamma'}\omega r + \frac{\lambda'}{\gamma'}\alpha p$
11 $r := r - \omega t$
12 $g = |r|$, $\gamma = \sigma/g$
13 $r := \gamma r$
14 $\gamma := g/|r|$
15 $p := \gamma p$, $v := \gamma v$
16 $\omega' = \omega$, $\alpha' = \alpha$, $\rho' = \gamma\rho$, $\gamma' := \gamma'\gamma$
17 $\lambda = \sigma_\lambda/|x|$
18 $x := \lambda x$
19 $\lambda' := \lambda'\lambda$

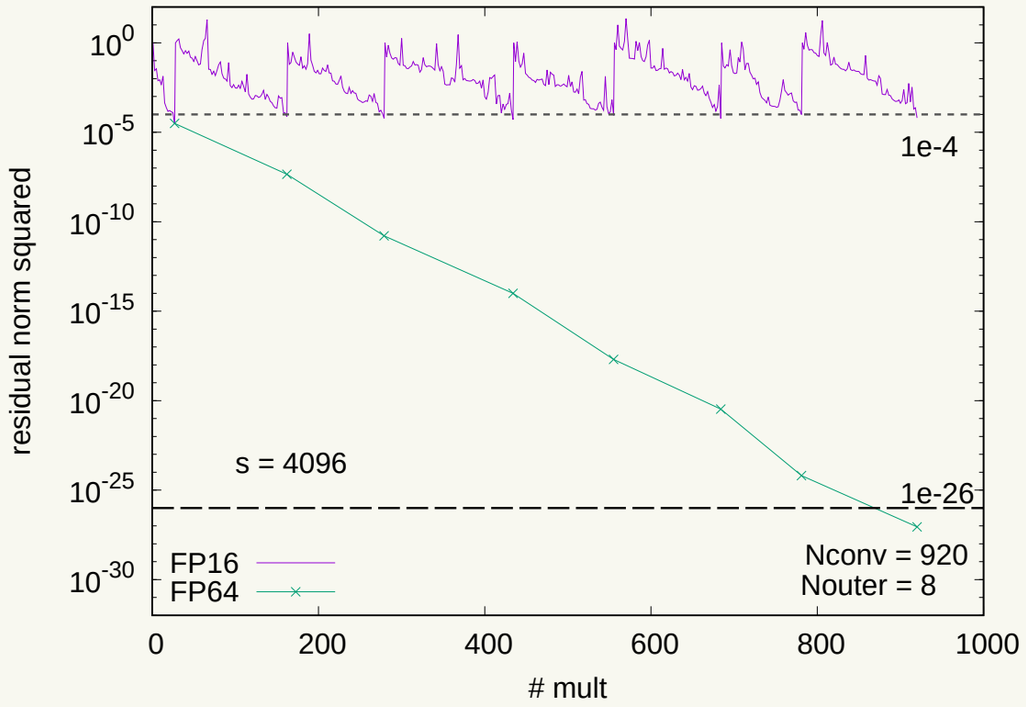
res. vector r: rescaled σ, true res. is r/γ'
sol. vector x: rescaled σ_λ, true sol. is x/λ'

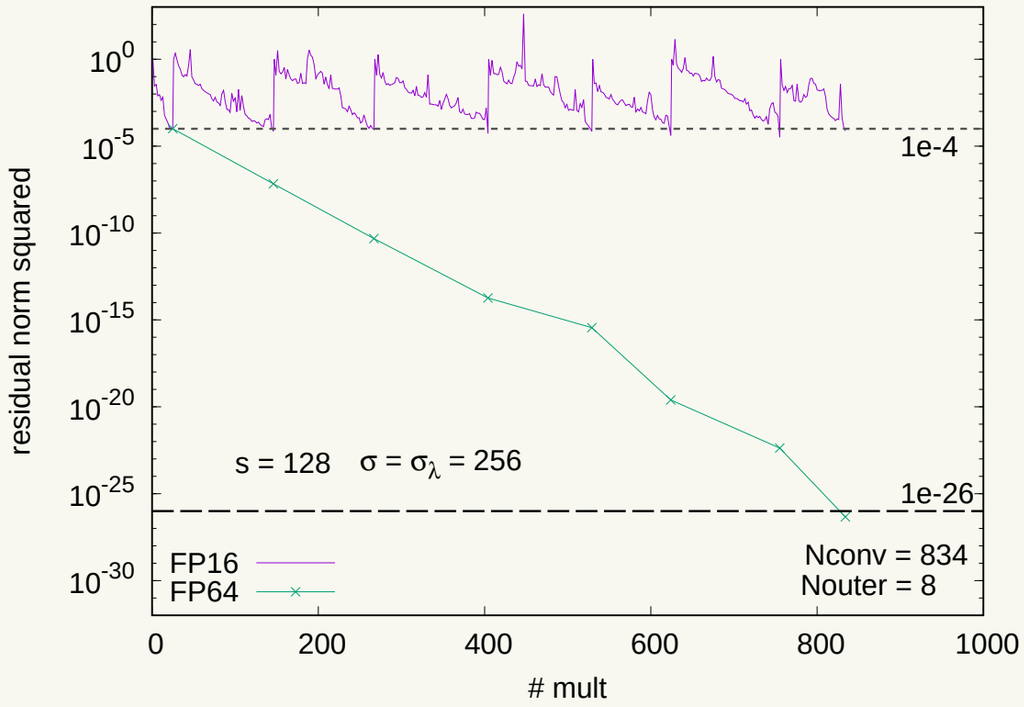
update with rescaling

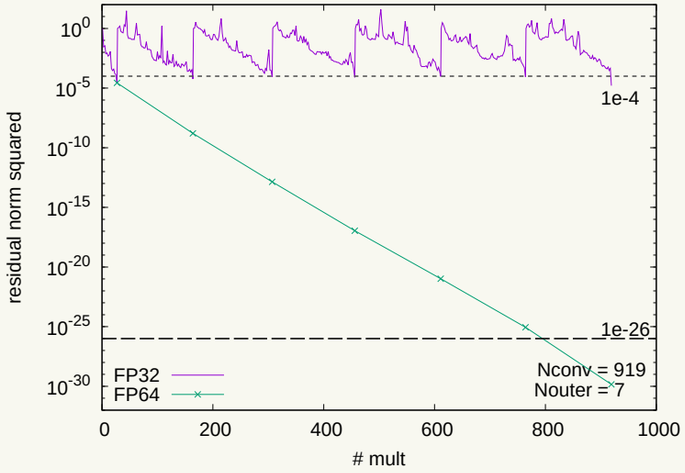
normalize the residual and working vectors

normalize the solution vector

Improved result Mat-vec mult: 920 → 834







mixed with single prec., tolerance for low prec. solver is the same as FP16 case.

New parameters

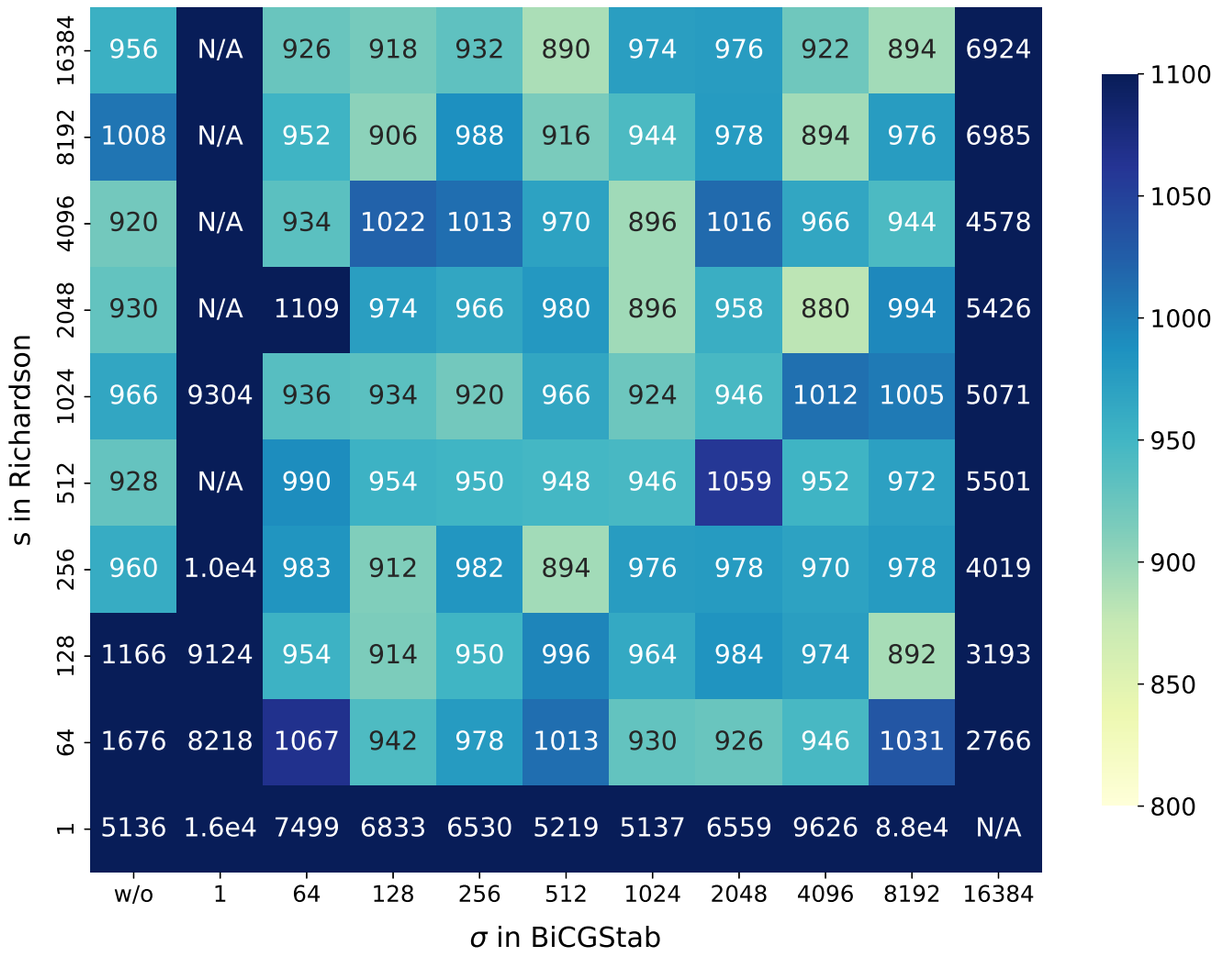
We have 3 additional parameters:

- s: norm of the input for the low prec. solver
- σ: normalization of the residual vector in the low prec. solver
- σ_λ: normalization of the solution vector in the low prec. solver

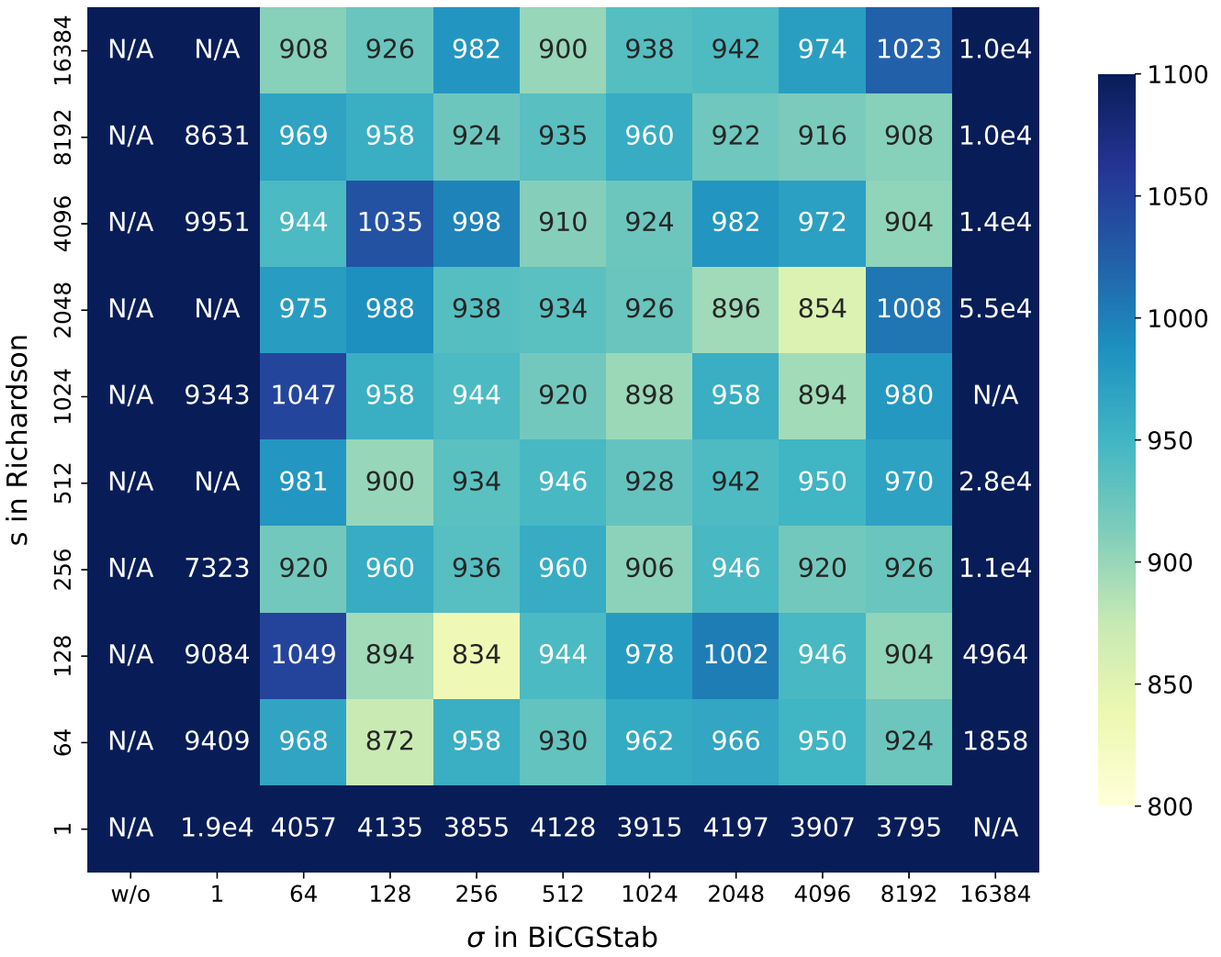
$\Rightarrow$Two-parameter (s, σ) or (s, $\sigma_\lambda = \sigma$) search of the optimal point without rescaling the solution vector

FP16 matrix-vec counts

(s, σ)



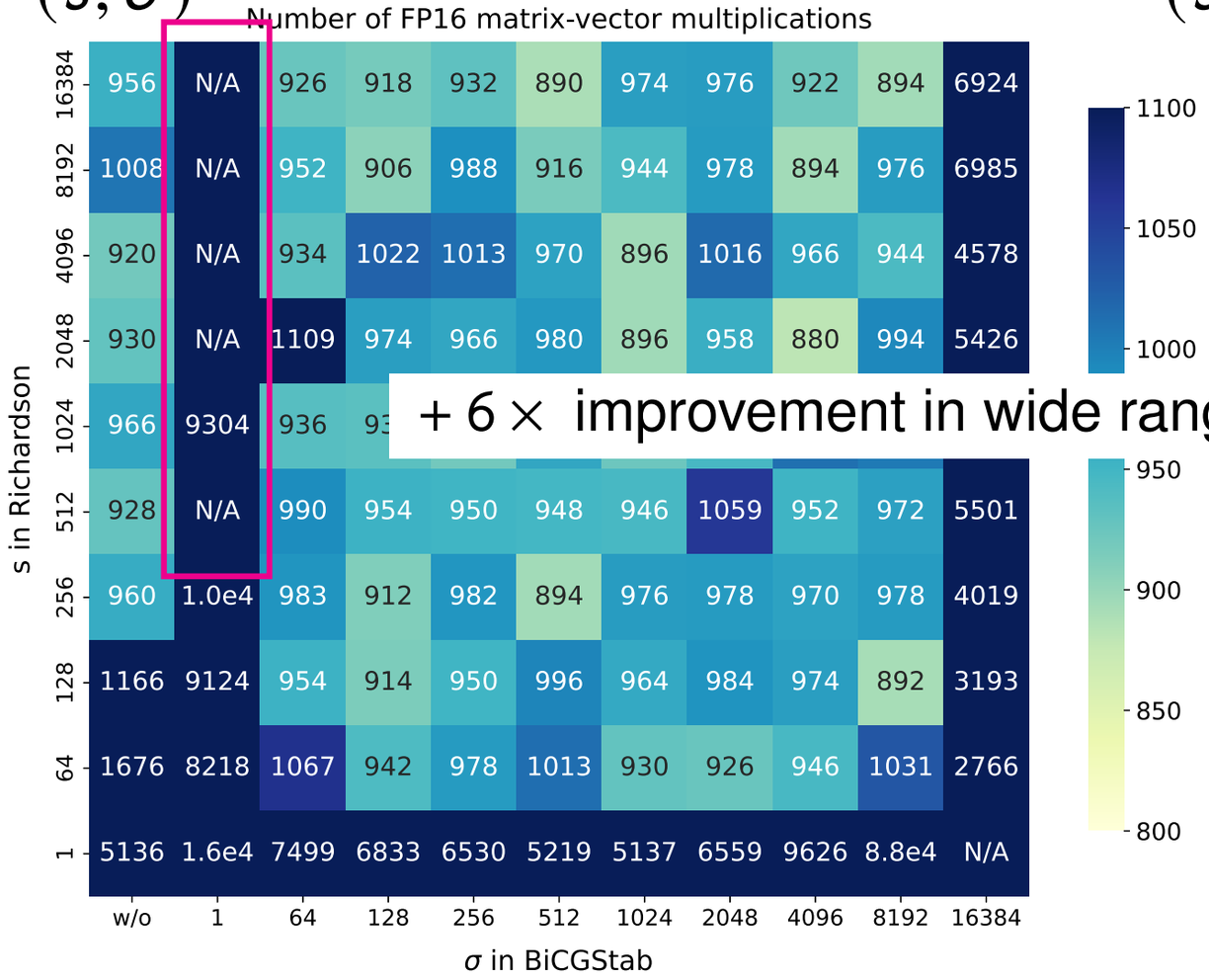
$(s, \sigma_\lambda = \sigma)$



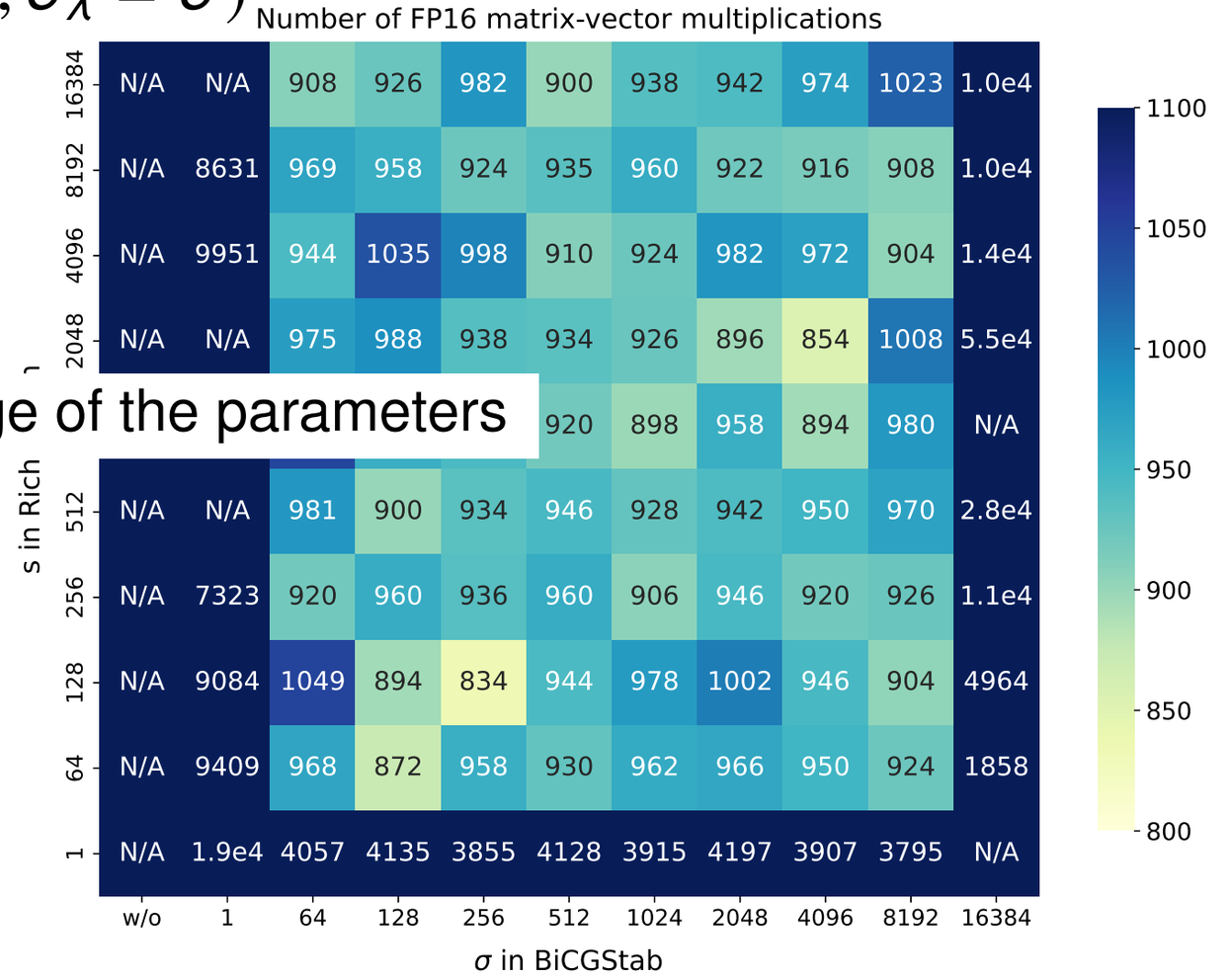
FP16 matrix-vec counts

N/A: due to overflow of the solution vector

(s, σ)



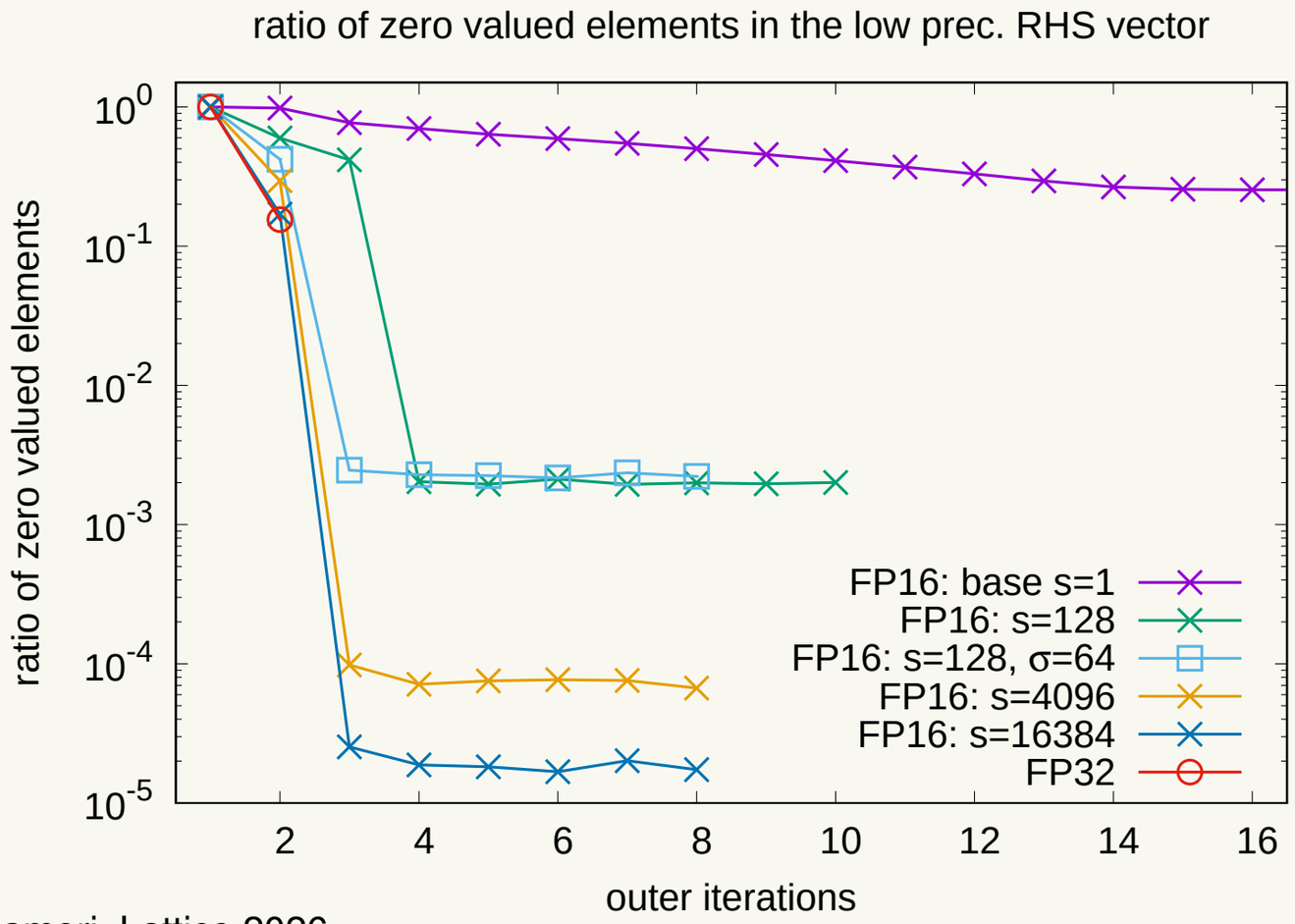
$(s, \sigma_\lambda = \sigma)$



$+6\times$ improvement in wide range of the parameters

Underflow?

- ratio of zeros in the input vector for low-prec. solver
- mixed prec. with FP32 (red circle): after 2nd outer iteration, all elements become non-zero
- Ratio of zeros ~ ratio of underflowed elements

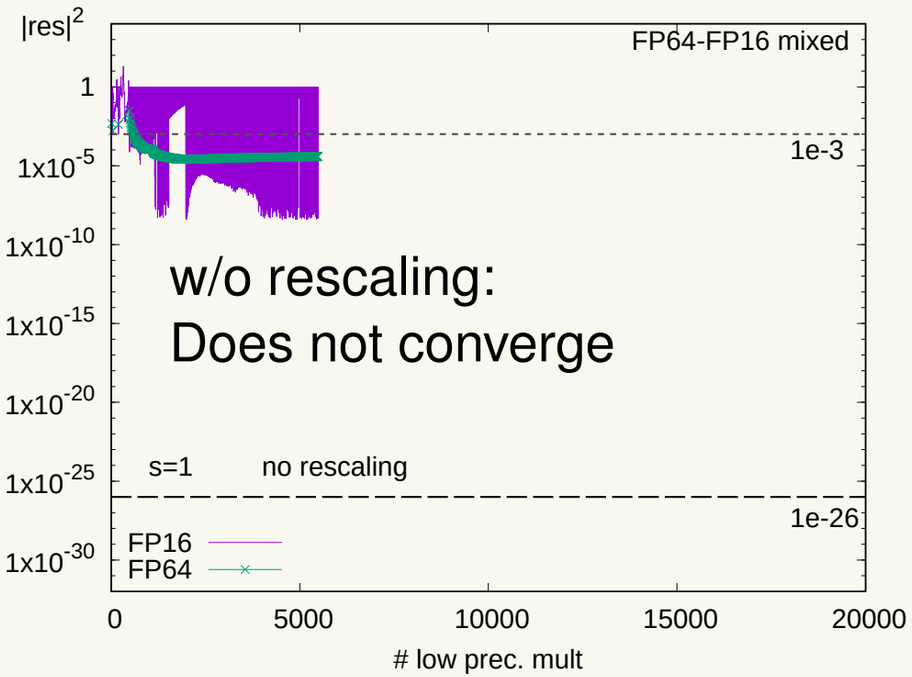


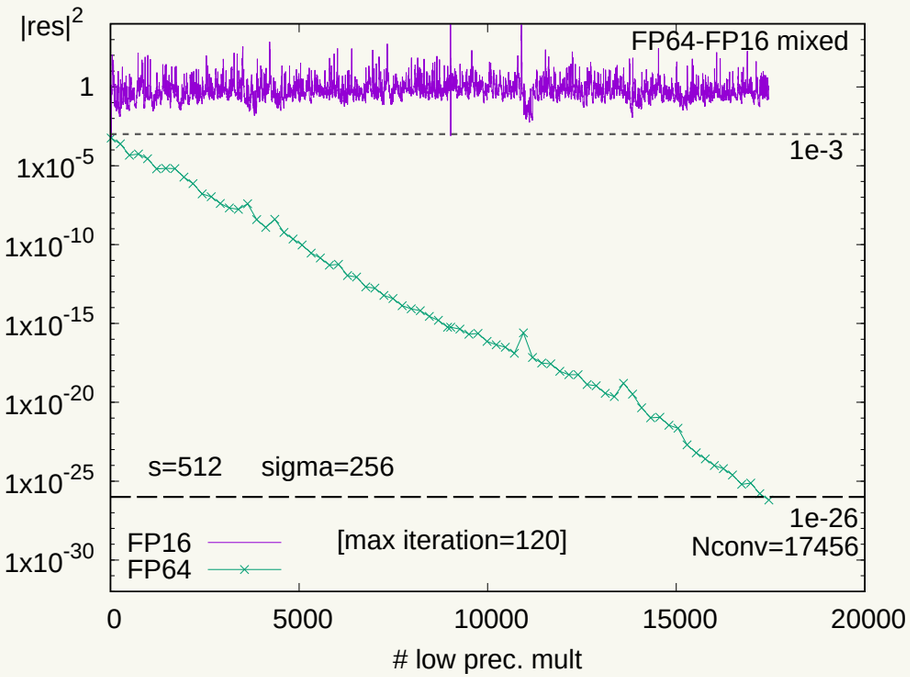
$$A_{\text{low}} x_{\text{low}} = \underbrace{b_{\text{low}}}_{\text{How many zeros?}}$$

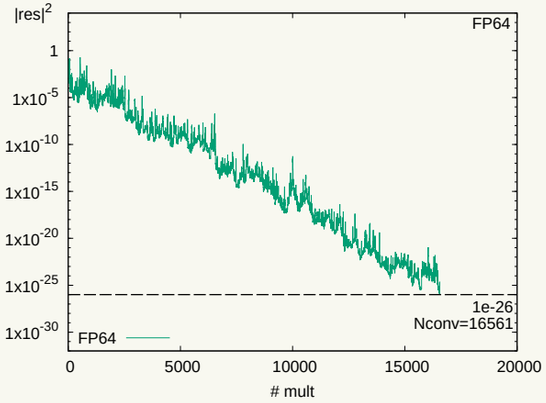
Benchmark with Clover fermion

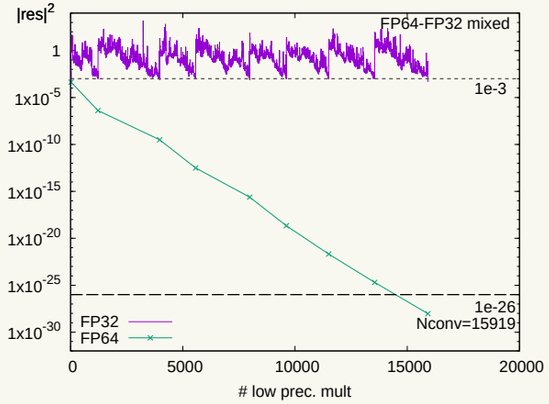
It works: 96^4 lattice, $M_\pi \sim 146$ MeV

configuration: PACS collab., PoS LATTICE2015 075 (2016)









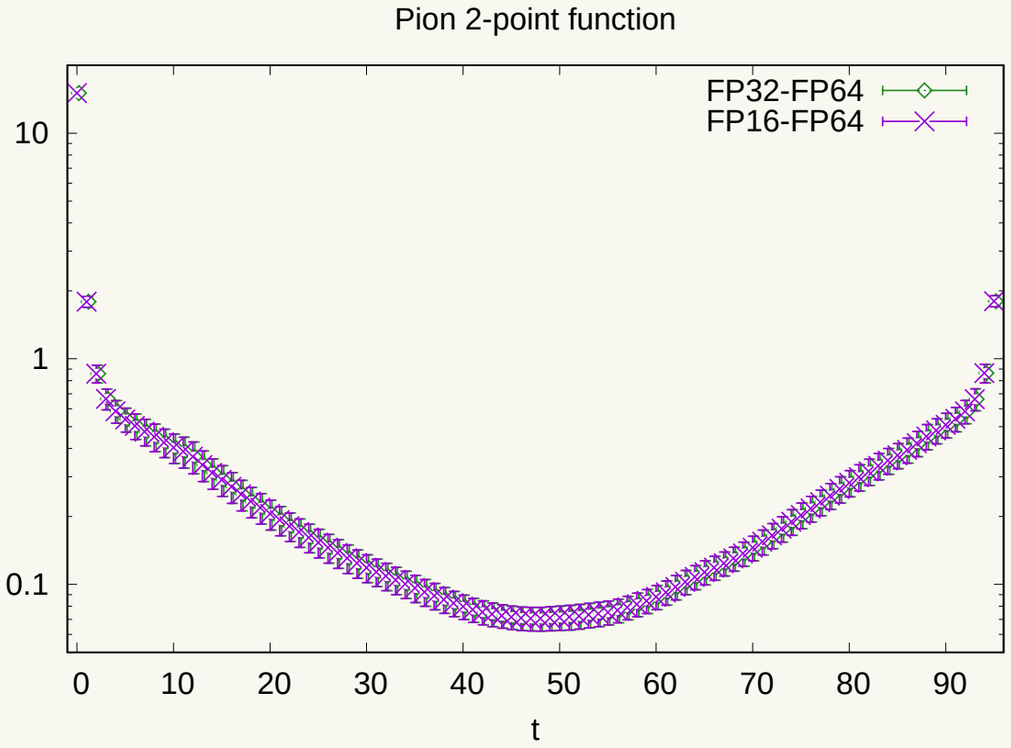
	time [sec]	speed up
FP64 BiCGStab	142.2	—
FP64-FP32 mixed	67.3	2.1
FP64-FP16 mixed	45.3	3.1

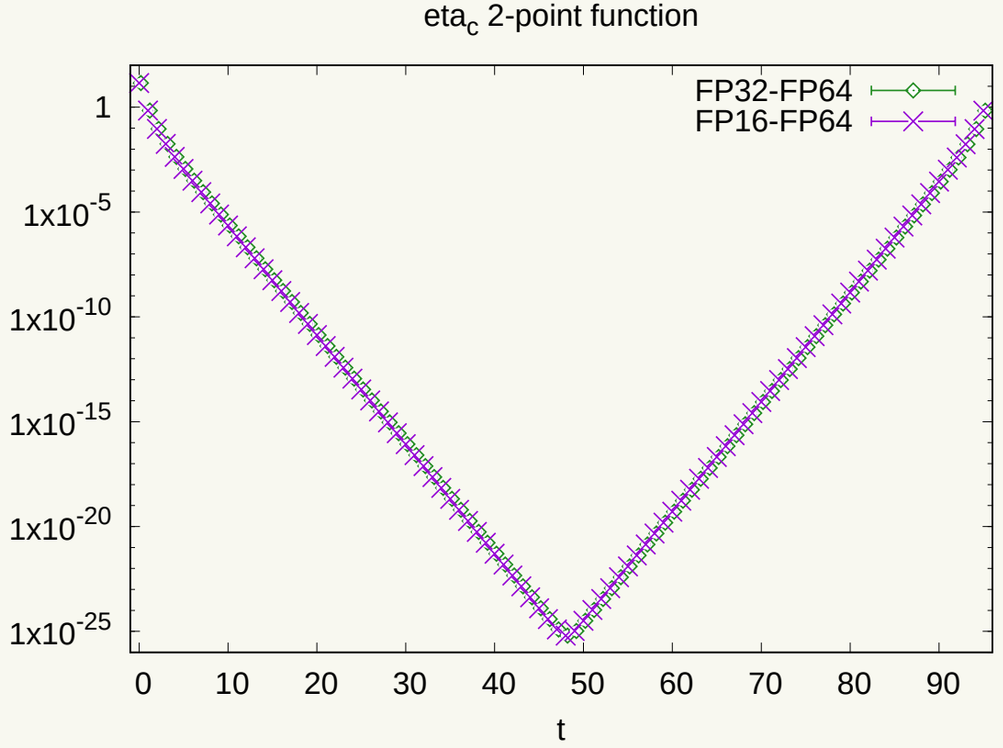
Fugaku 144 nodes

2-point functions: $m_q \sim$ light/charm quarks

small residual norm does not always mean small error
$\Rightarrow$fine, the same result as FP32-FP64 mixed prec. solver

$|\mathrm{res}|^2$ is 10^{-26}





11 conf. 9347 sec $\rightarrow$ 6970 sec (1.34$\times$ vs FP32-FP64) 88 sec $\rightarrow$ 51 sec (1.72$\times$) 144 nodes

Performance of Mat-Vec mult ($A = 1 - D_{ee}^{-1} D_{eo} D_{oo}^{-1} D_{oe}$)

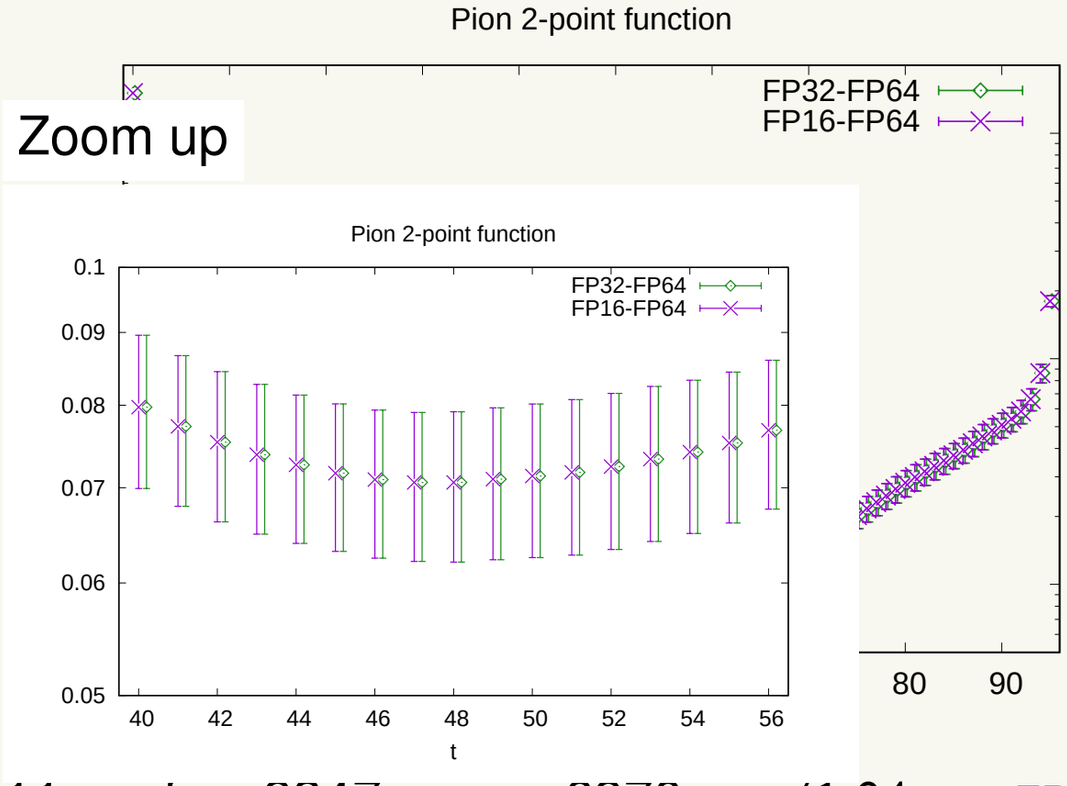
$32 \times 24 \times 12 \times 16$ lattice /node

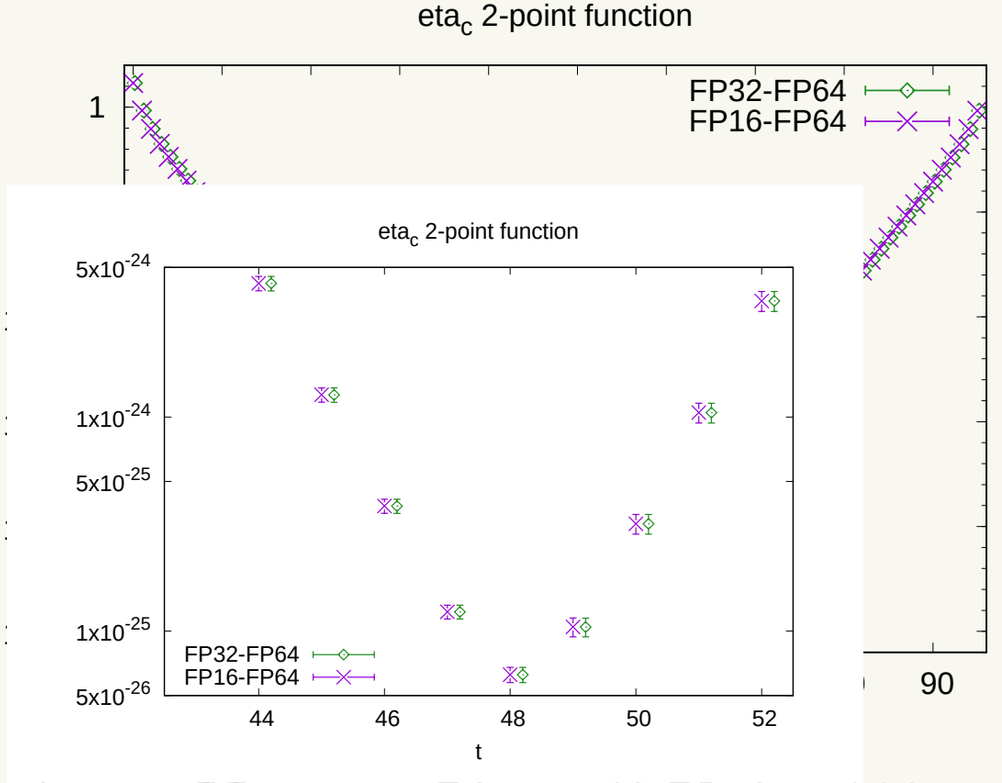
FP64	161 GFlops/node
FP32	329
FP16	521

2-point functions: $m_q \sim$ light/charm quarks

small residual norm does not always mean small error
⇒fine, the same result as FP32-FP64 mixed prec. solver

$|\mathrm{res}|^2$ is 10^{-26}





11 conf. 9347 sec → 6970 sec (1.34× vs FP32-FP64) 88 sec → 51 sec (1.72×) 144 nodes

Performance of Mat-Vec mult ($A = 1 - D_{ee}^{-1} D_{eo} D_{oo}^{-1} D_{oe}$)

$32 \times 24 \times 12 \times 16$ lattice /node

FP64	161 GFlops/node
FP32	329
FP16	521

Additional tricks

- Implementation in FP16 against underflow: $(U_\mu\psi)^R_1 = \underbrace{(U_\mu)^R_{11}\psi^R_1}_{\text{may underflow}} - (U_\mu)^I_{11}\psi^I_1 + \ldots$

  cf. $|(U_\mu)^R_{ij}|,\ |(U_\mu)^I_{ij}| \leq 1$

  - hopping: $\kappa U_\mu\psi \Rightarrow \frac{\kappa}{c} V_\mu\psi$ with $V_\mu = \text{toFP16}(\underbrace{cU_\mu}_{\text{FP64}})$, $c > 1$
  - clover: $D^{-1}_{ee} = 1 + \ldots$
    $D^{-1}_{ee}\psi \Rightarrow \frac{1}{c'} G_{ee}\psi + \psi$ with $G_{ee} = \text{toFP16}(\underbrace{c'(D^{-1}_{ee} - 1)}_{\text{FP64}})$, $c' > 1$
  - in this talk: $c = 128$, $c' = 64$
- maximum iteration for FP16 solver: fixed (but exits immediately if converged)

Conclusion and Outlooks

Conclusion and Outlooks

- Mixed precision solver with half prec. (FP16): New algorithm with recaling
- 3 times faster than double prec., 1.3–2 times faster than mixed prec. with FP32

Outlooks

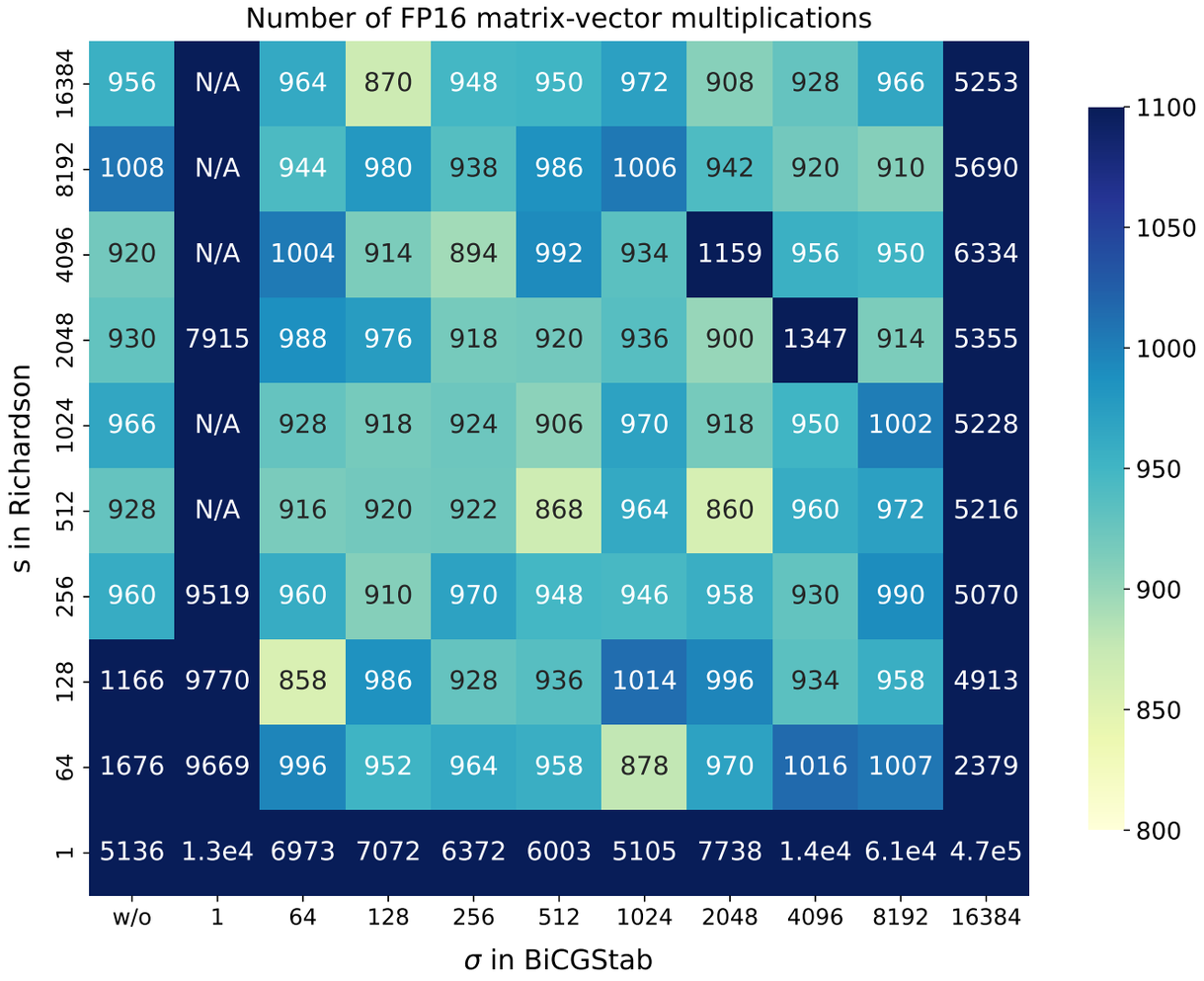
- Domain-wall fermion
  - more intermediate arithmetics and may cause more rounding error
  - applying the rescaling to CG is straightforward
- feedback to GPU implementation

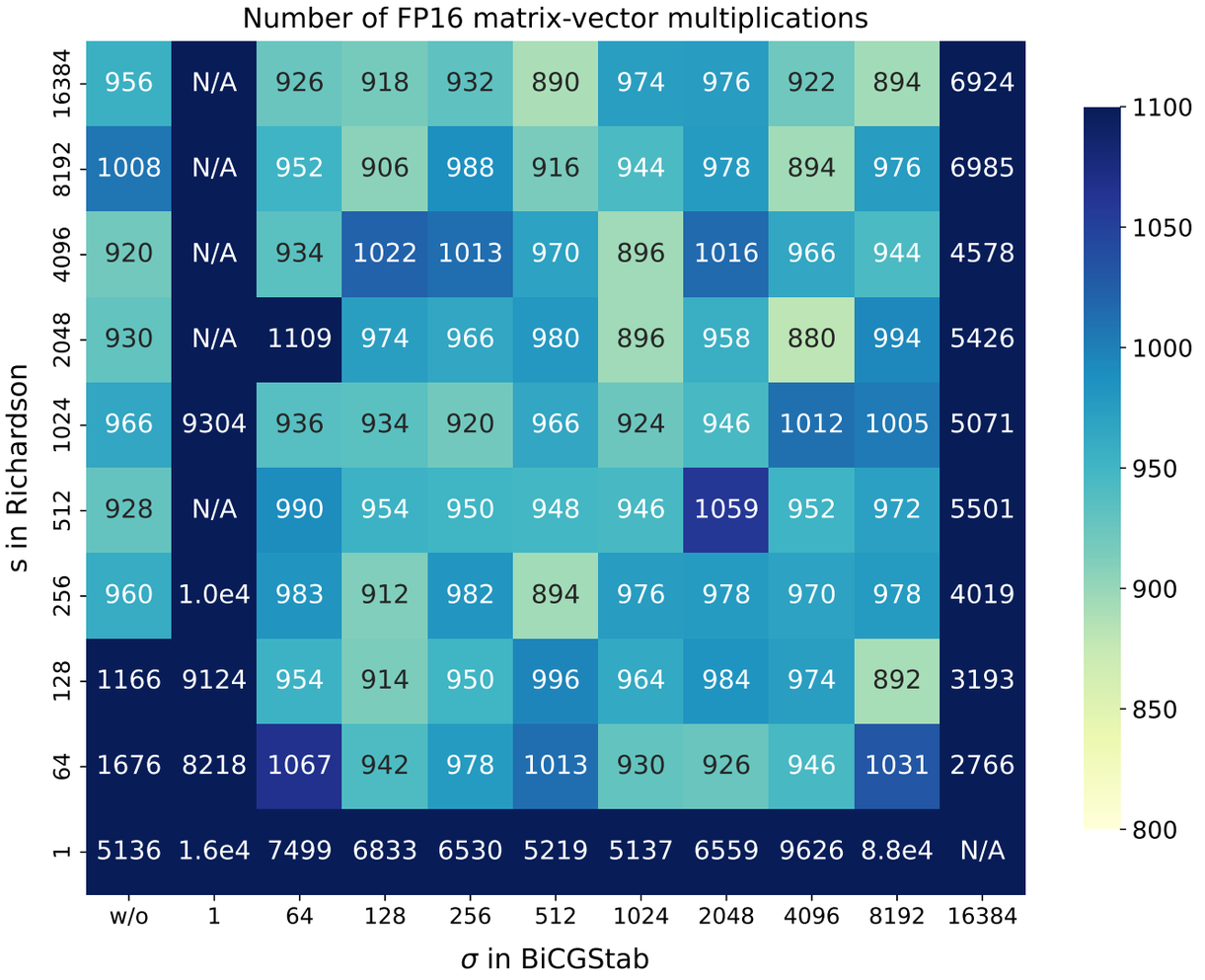
Acknowledgments

- Computational resources
  - Supercomputer Fugaku (RIKEN R-CCS, ra250026, ra260025)
  - Wisteria-O (Univ. of Tokyo & Univ. Tsukuba, through MCRP in CCS, Univ. of Tsukuba)
  - Miyabi-G (Univ. of Tsukuba, through MCRP in CCS, Univ. of Tsukuba)
- Grant: JSPS KAKENHI (JP22H01224, JP25H01109)

Backup

FP16 matrix-vec counts: without rescaling of solution vector: (s, σ)



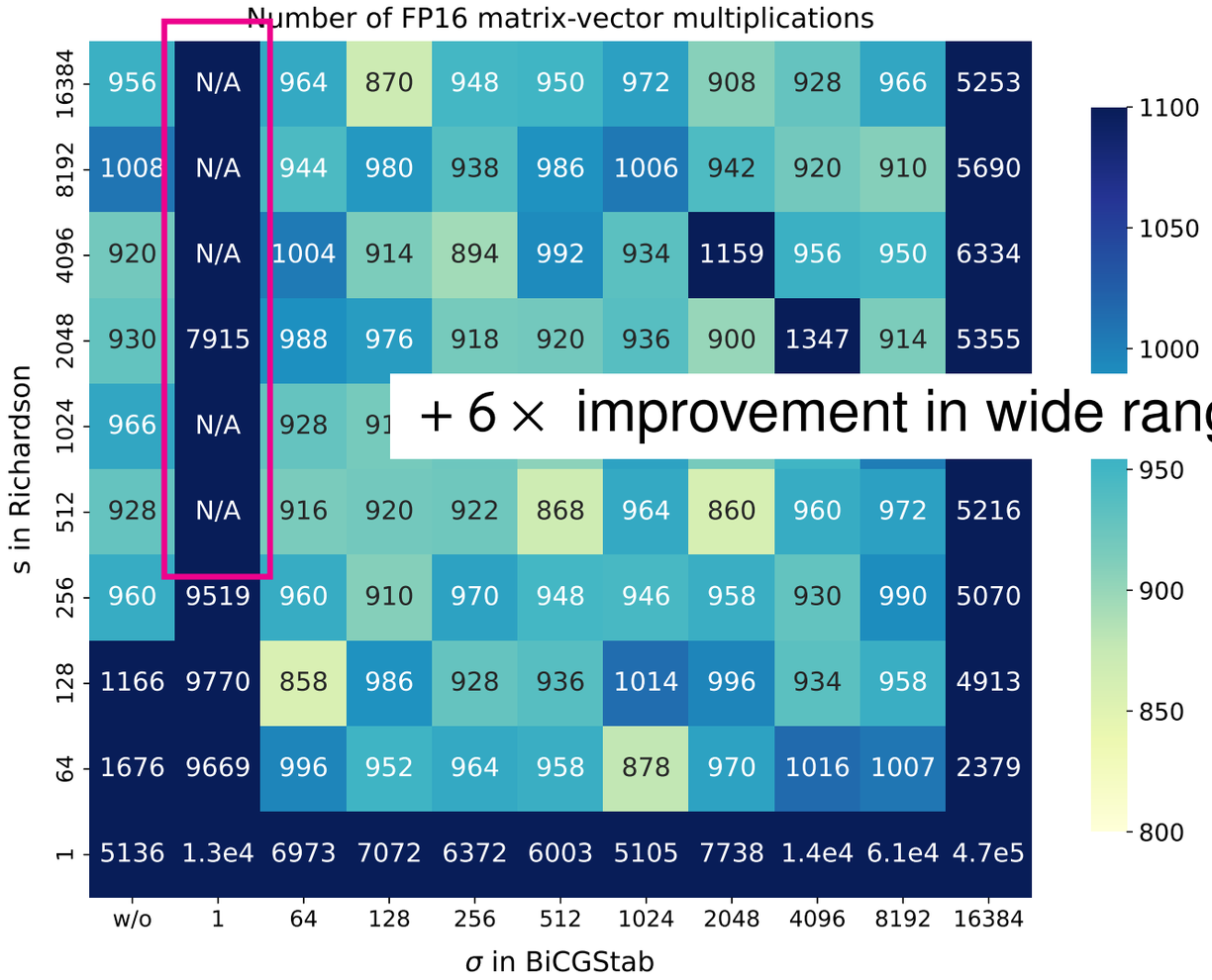


w/o recalculating γ

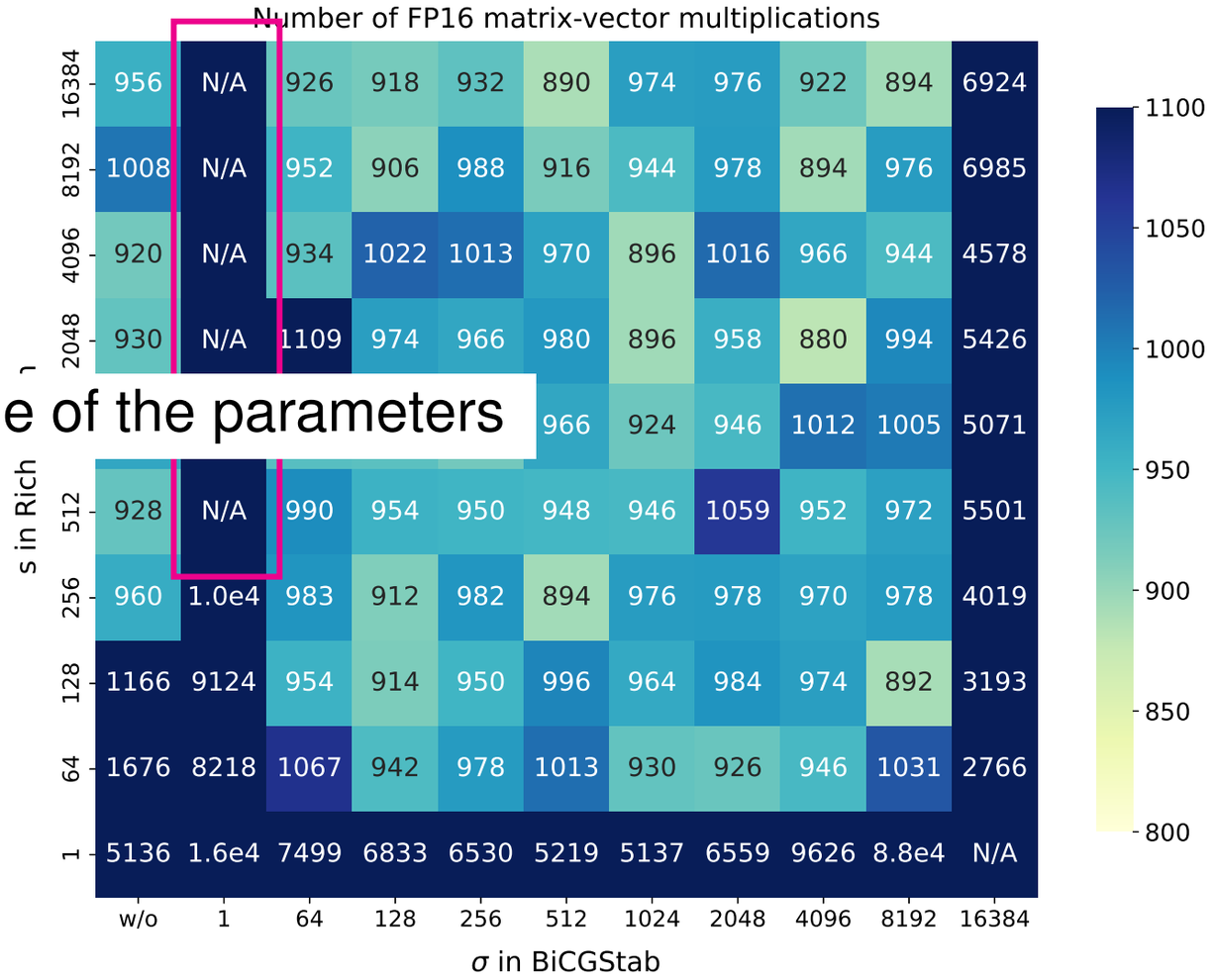
w/ recalculating γ

FP16 matrix-vec counts: without rescaling of solution vector: (s, σ)

N/A: due to overflow of the solution vector



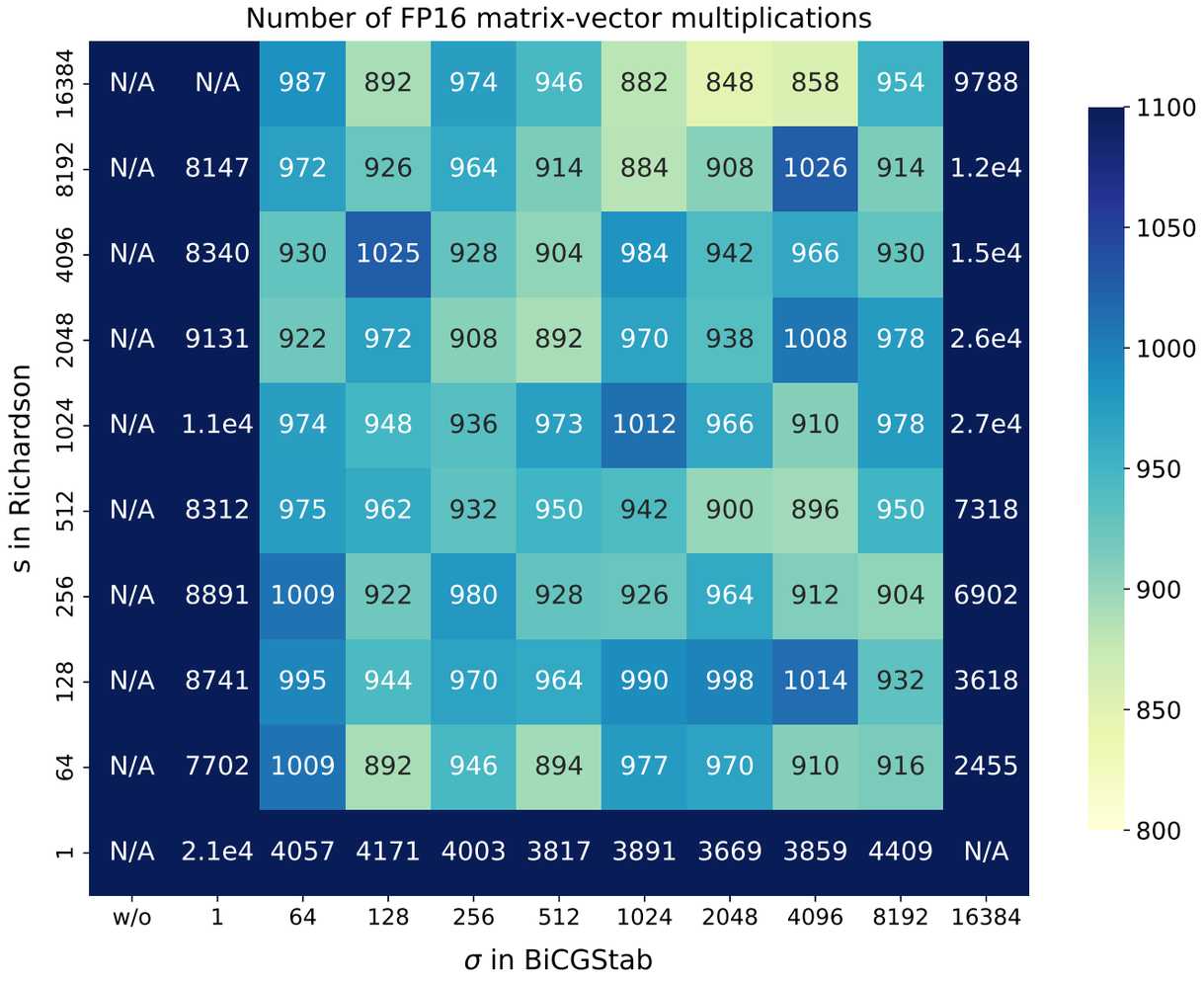
+ 6× improvement in wide range of the parameters



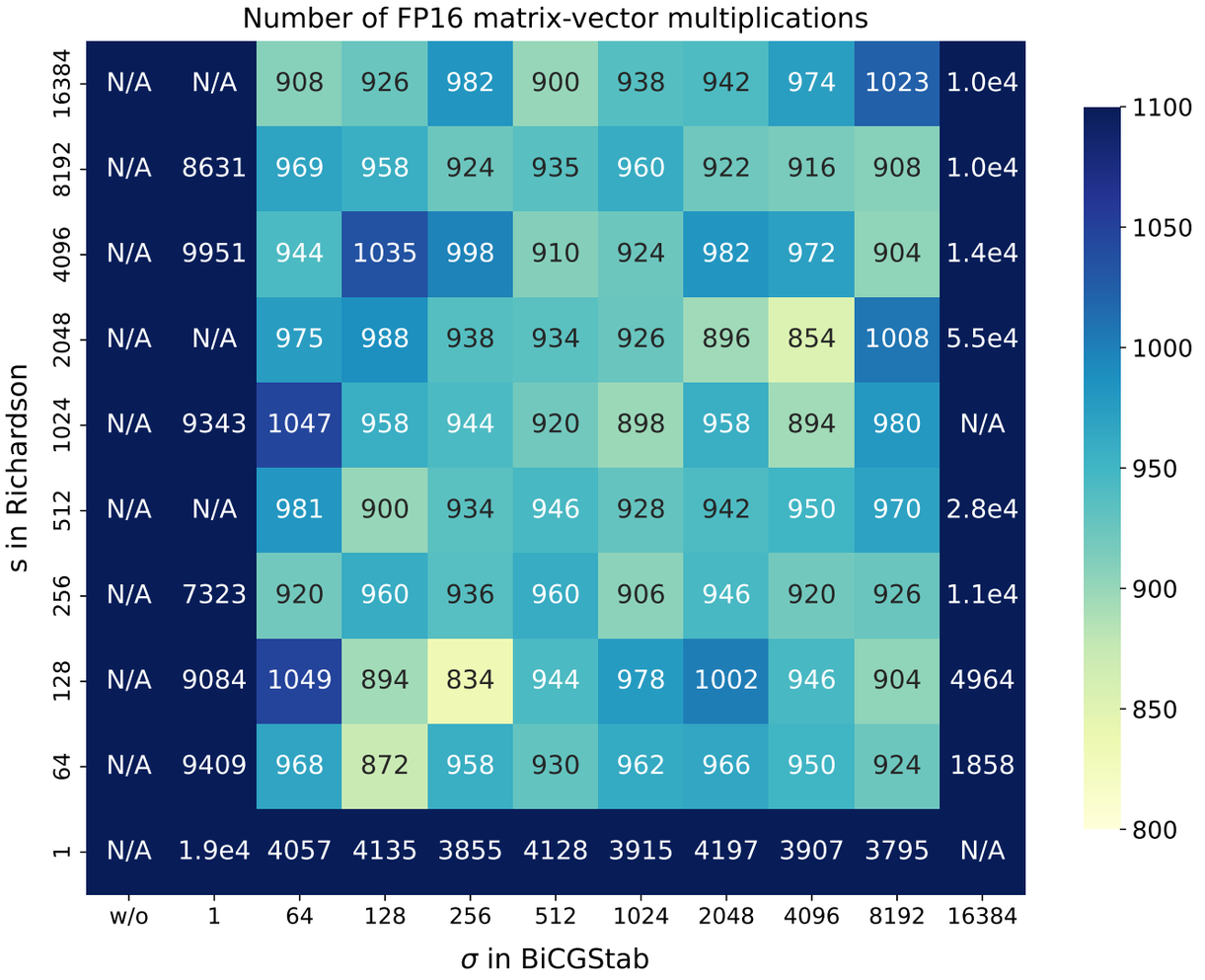
w/o recalculating γ

w/ recalculating γ

FP16 matrix-vec counts: with rescaling of solution vector ($s, \sigma_\lambda = \sigma$)

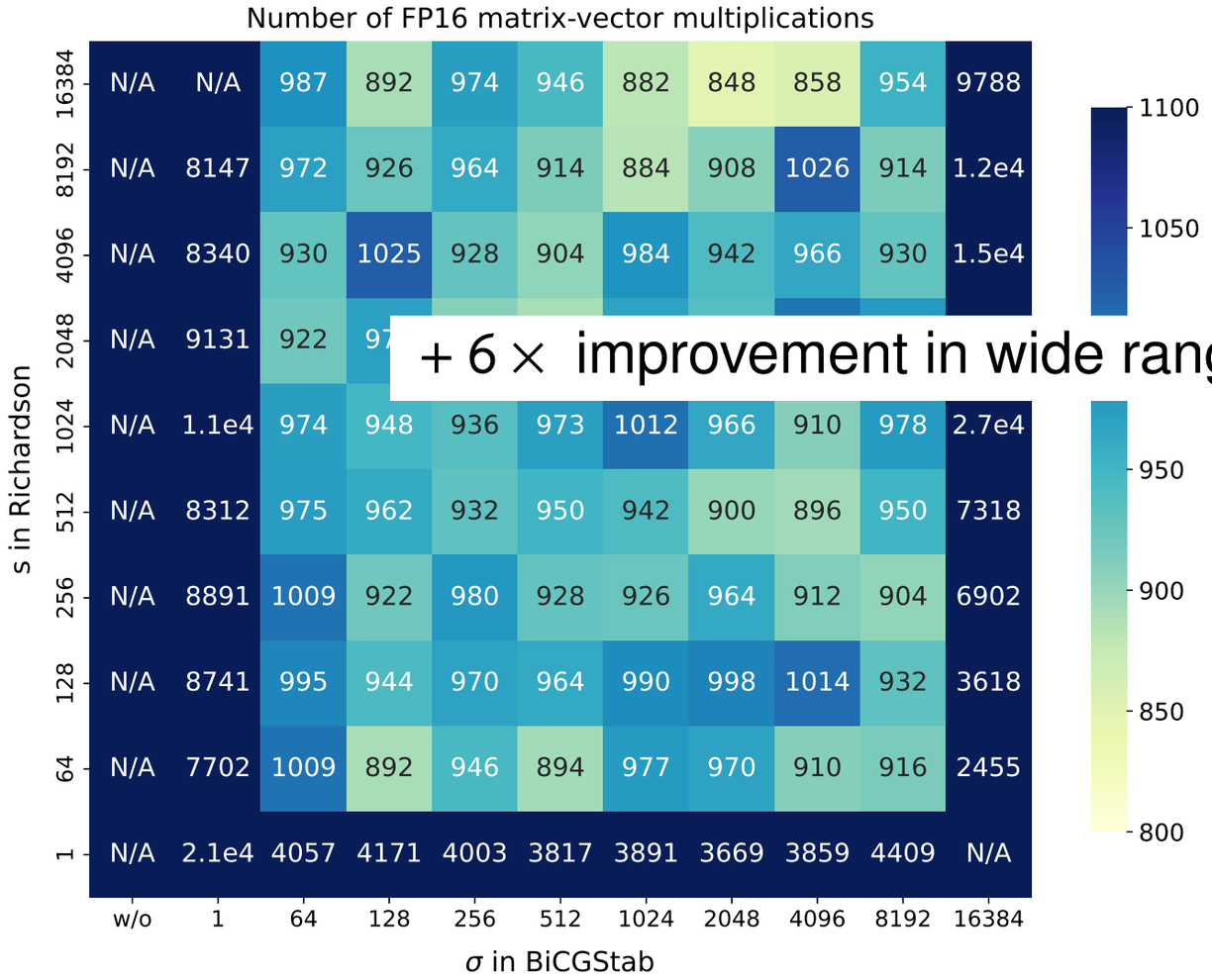


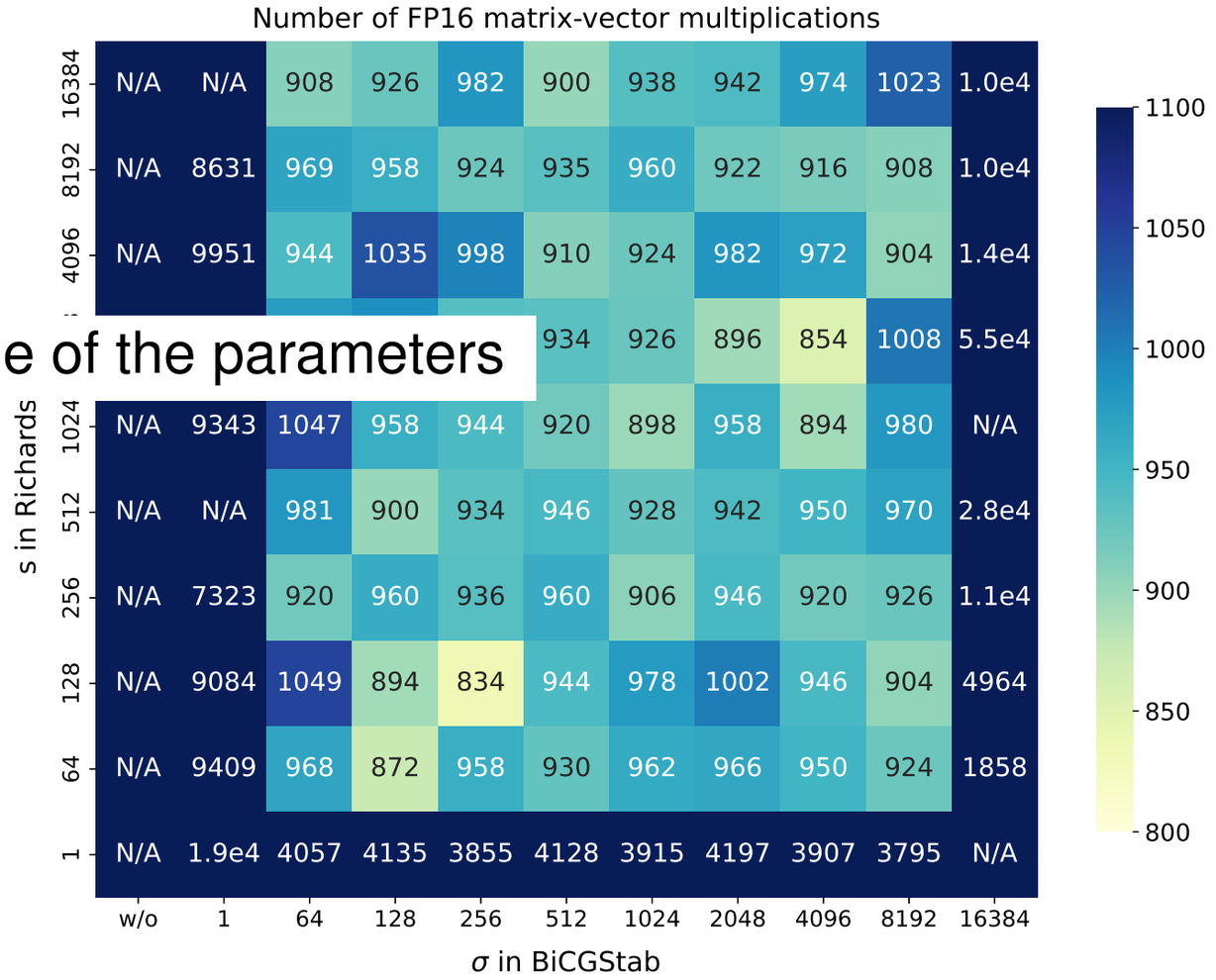
w/o recalculating γ



w/ recalculating γ

FP16 matrix-vec counts: with rescaling of solution vector ($s, \sigma_\lambda = \sigma$)





$+6\times$ improvement in wide range of the parameters

w/o recalculating γ

w/ recalculating γ

Performance (Wilson)

time to solution

scheme	s	σ	σ_λ	low prec. $\epsilon_{\text{tol}}{}^2$	#mult (low)	time [sec.]
double	—	—	—	—	775	1.39
mixed w/ FP32	1	—	—	10^{-12}	989	0.96
mixed w/ FP32	1	—	—	10^{-4}	919	0.92
mixed w/ FP16	128	64	—	10^{-4}	858	0.46
	4096	—	—	10^{-4}	920	0.47
	128	256	256	10^{-4}	834	0.46

Mat-Vec multiplication kernel

lattice size	# node	double	single	half
$32 \times 16 \times 16 \times 16$	1	145	316	767 GFlops
$32 \times 32 \times 32 \times 64$	16	2035	3895	8246 GFlops

$\Rightarrow$half-prec performance is more than 4-times of double prec. performance
(maybe effect of the cache)