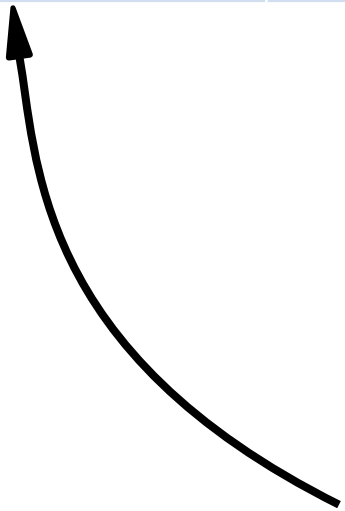
Piotr Luszczek

Half Precision Benchmarks

IEEE 754 (2008)

Precision	Width	Exponent bits	Mantissa bits	Epsilon	Max
Quadruple	128	15	112	10^{-34}	1.2×10^{4932}
Extended	80	15	64	10^{-19}	
Double	64	11	52	10^{-16}	1.8×10^{308}
Single	32	8	23	10^{-7}	3.4×10^{38}
Half	16	5	10	10^{-3}	65504



Only storage format is specified

Data Types

- long double
  - 80 or 128 bits
- double
  - 64 bits
- float
  - 32 bits
- ~~short float~~ __half
  - 16 bits

- real*16
  - 128 bits
- real*8
  - 64 bits
- real*4
  - 32 bits
- real*2
  - 16 bits

FP16 Hardware (current and future)

- AMD
  - MI5, MI8, MI25
- ARM
  - NEON VFP FP16 in V8.2-A
- Intel
  - Knights Mill
- NVIDIA Pascal
  - P100, TX1
- Supers
  - TSUBAME 3.0
  - Tokyo Tech

Applications Using FP16

- Machine Learning
  - Deep Neural Networks
  - OpenVZ
- Linear Algebra
  - Eigen
  - Yours truly
- Molecular dynamics
  - Gromacs

Classic Iterative Refinement

- Linear system Ax=b may be solved through LU factorization:
  - L, U, P ← lu_factor(A)
  - y ← L \ Pb
  - x ← U \ y
  - r ← b - Ax (use higher precision to accumulate)
  - z ← U \ L \ P * r
  - x_{final} ← x+z (use higher precision)
- All operations performed in the same floating-point precision

Mixed-Precision Iterative Refinement

- Linear system in 64-bit precision Ax=b may be solved through LU factorization in 32-bit precision:
  - L,U, P ← lu(A) (n^3) (32 bits)
  - y ← L \ Pb (n^2) (32 bits)
  - x ← U \ y (n^2) (32 bits)
  - r ← b - Ax (n^2) (64 bits)
  - z ← P L \ U \ r (n^2) (32 bits)
  - x_{final} ← x+z (n) (64 bits)
- Requirement:
  - Matrix A must be well conditioned in 32 bits:
    - $\kappa(A) < 10^8$

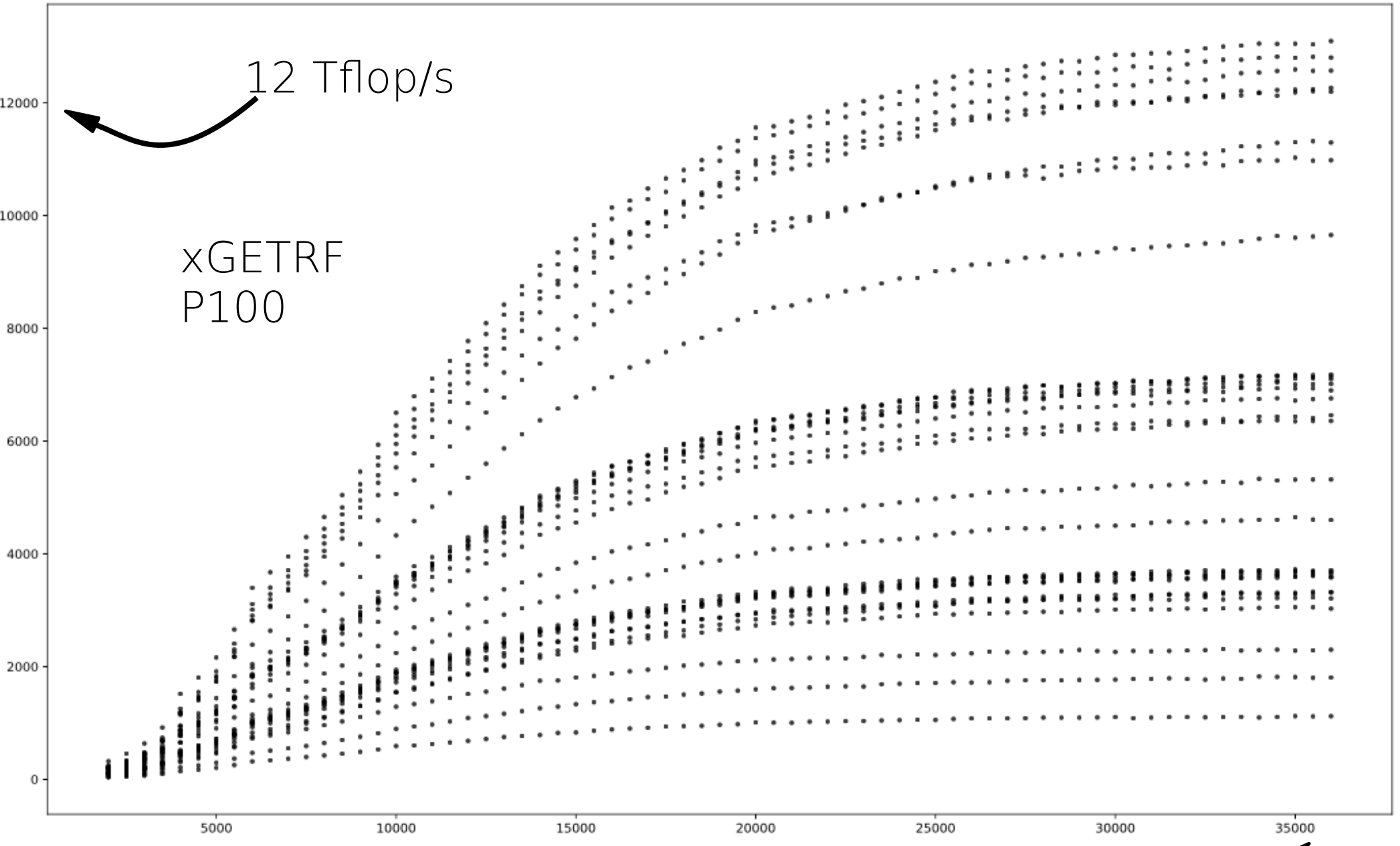
Support: Assembly and Intrinsics

- x86
  - CVTSH_SS, CVTSS_SH
  - emmintrin.h
    - _cvtss_sh(), _cvtsh_ss()
  - f16cintrin.h → x86intrin.h
    - _mm_cvtph_ps(), _mm_cvtps_ph(), _mm256_cvtph_ps(), _mm256_cvtps_ph()
- PTX
  - cvt.f16.*
  - fma.f16x2
- ARM
  - vld1_f16, vst1_f16, vcvt_f16_f32

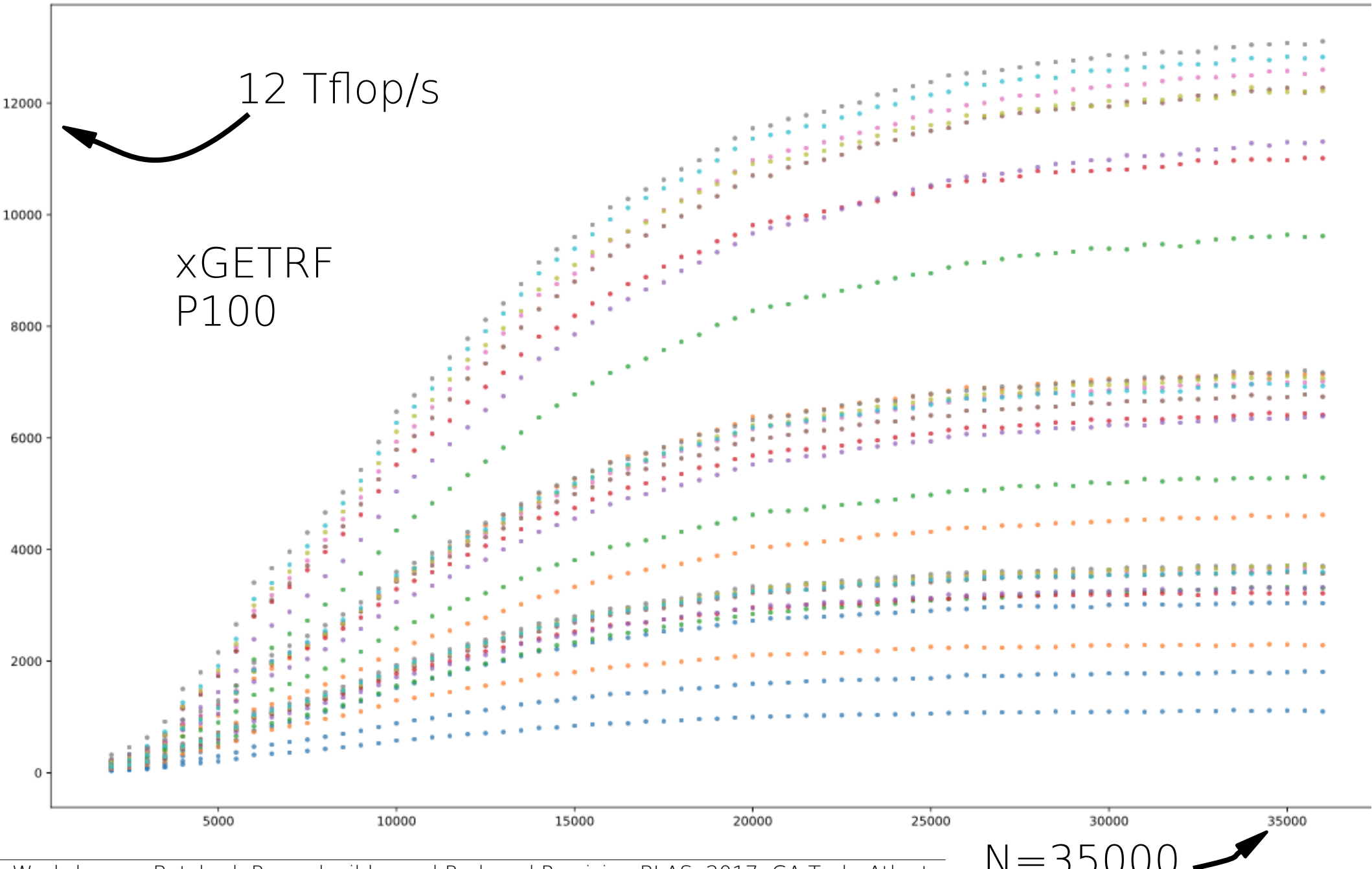
Programming Environments

- Julia
  - A = zeros(Float16, N, N); b = zeros(Float16, N,N);
  - A[:,:] = randn(N, N); b[:,:] = randn(N,1);
  - x = A \ b; # OK
- Python
  - numpy.float16
  - linalg.solve(randn(N,N,float16), randn(N,1,float16))
  - TypeError: array type float16 is unsupported in linalg
- MATLAB
  - MEX

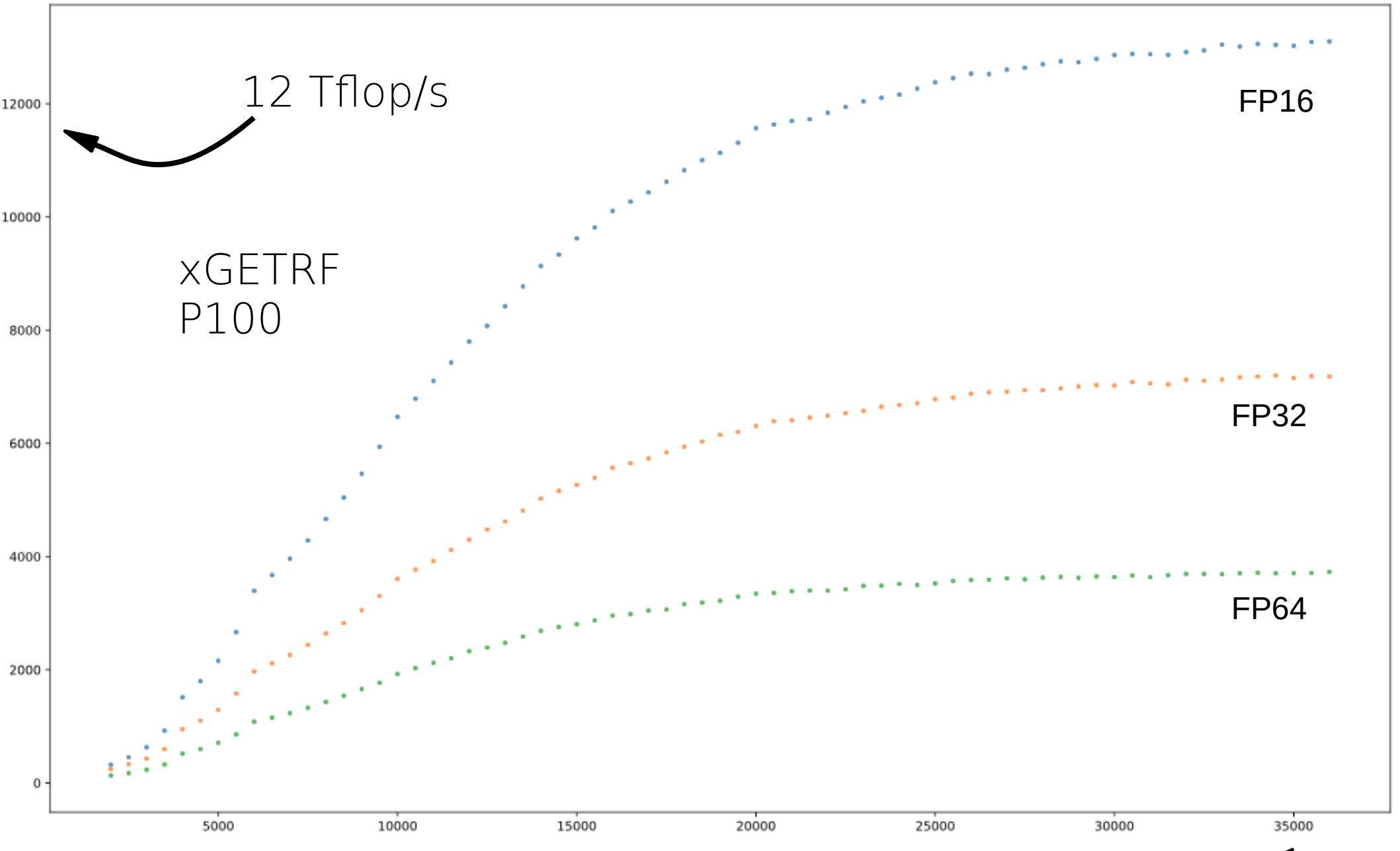
Autotuning with FP16, FP32, and FP64



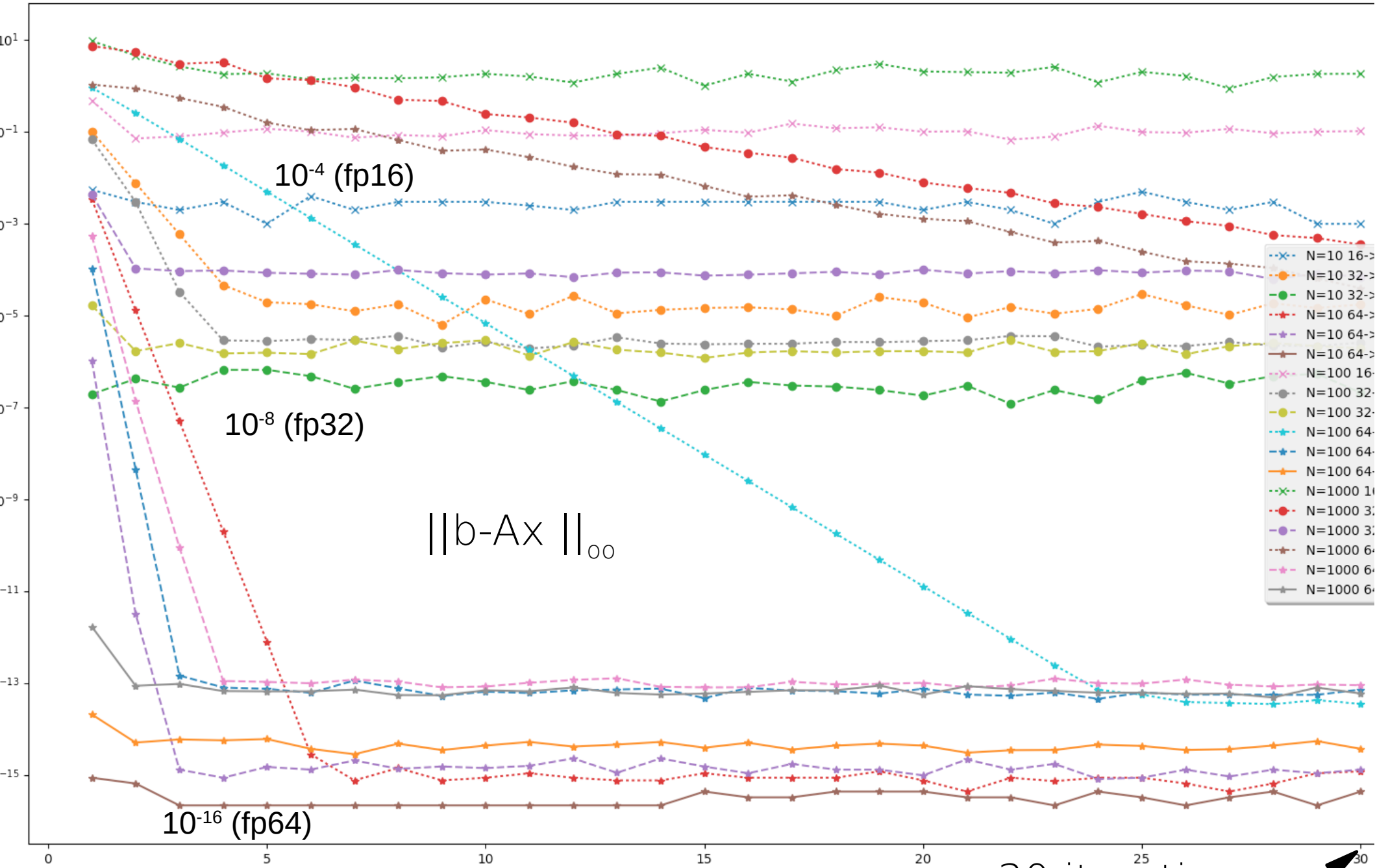
Autotuning with FP16,FP32,FP64 (color)



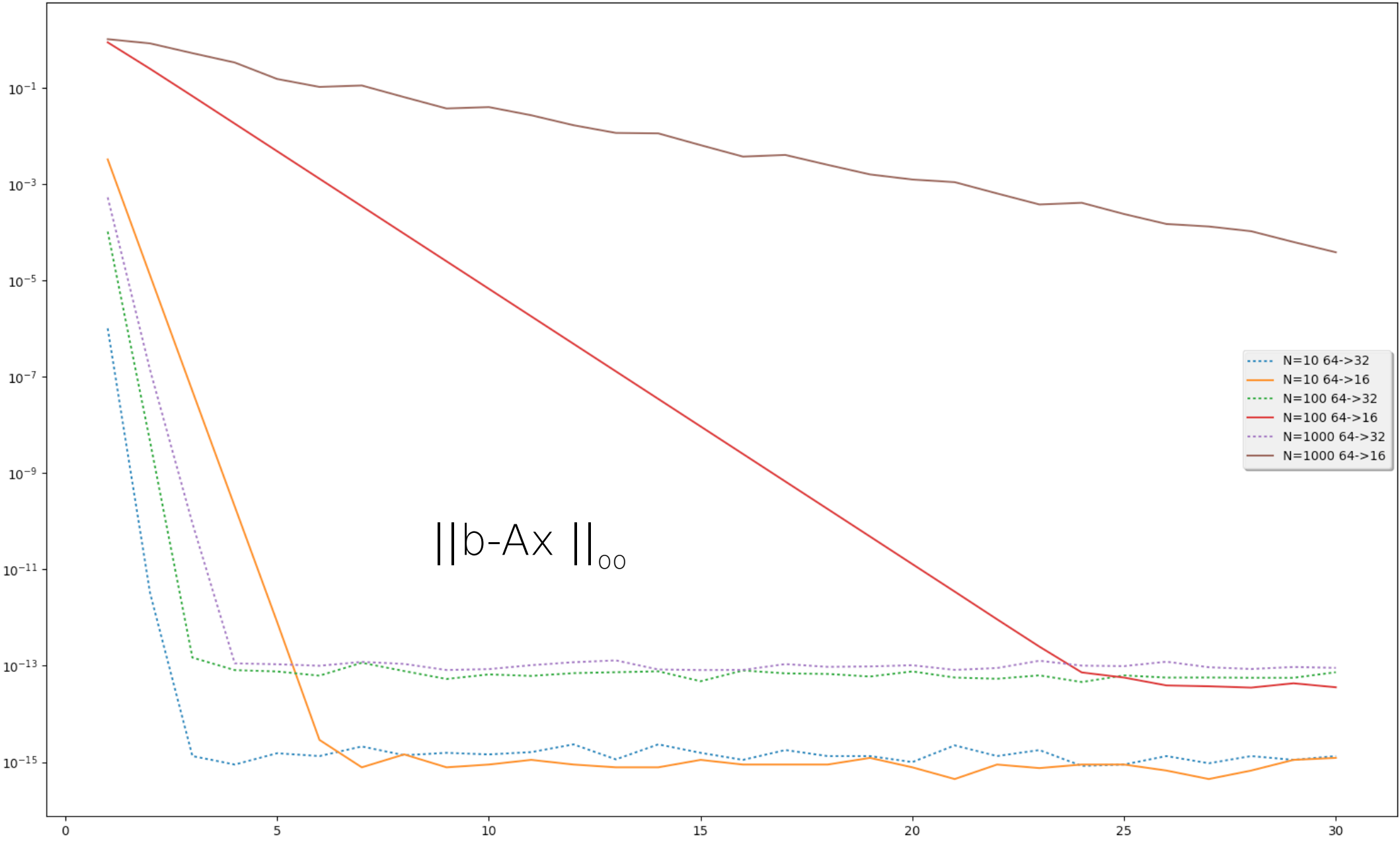
Best Performers for FP16, FP32, FP64



Convergence Results: All Precisions

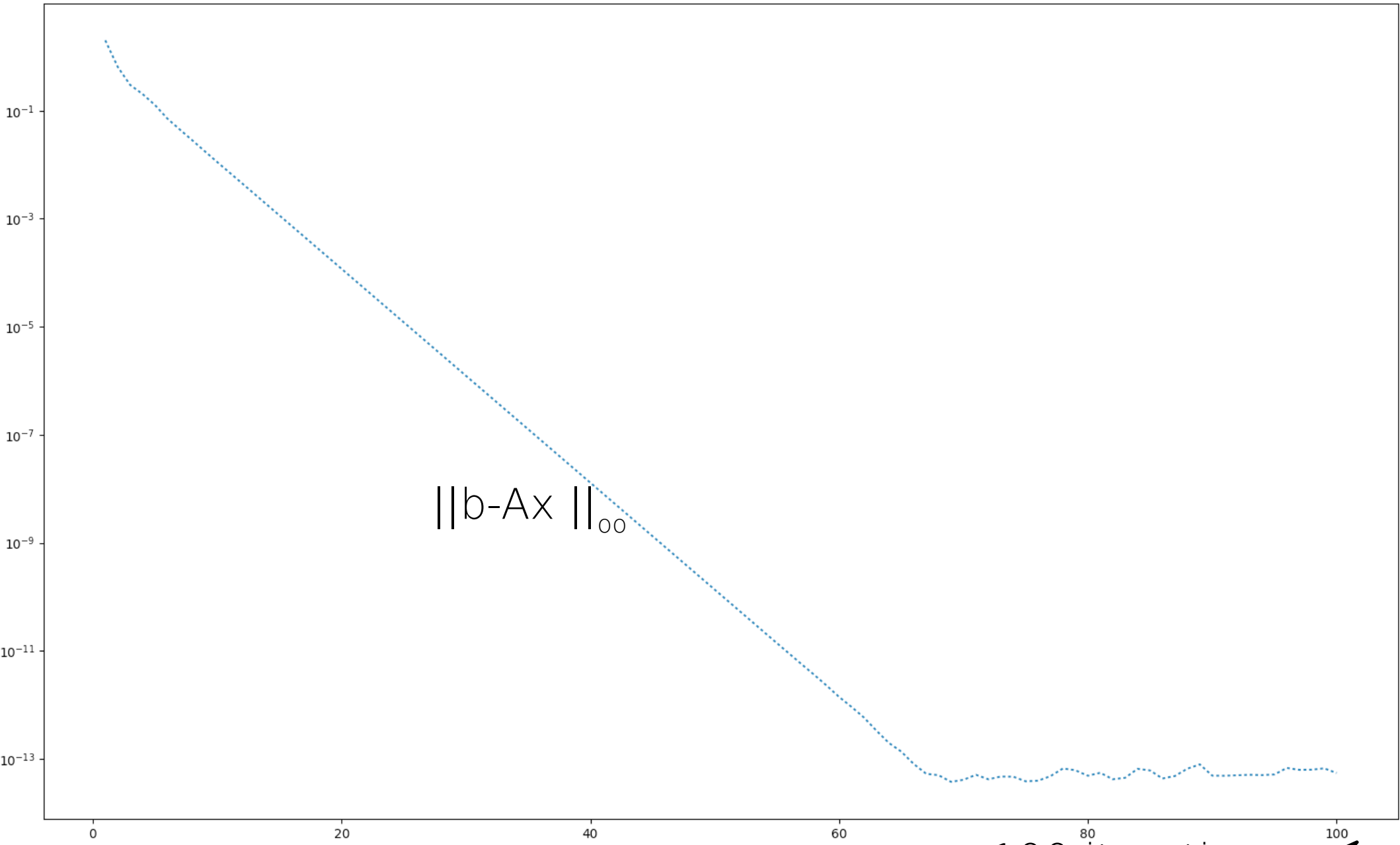


Convergence: FP64 to FP32 / FP16



30 iterations

Convergence: FP64 → FP16



Future Work

- Hardware
  - IBM/NVIDIA Minsky
  - ARM/Cavium
  - Tegra/Jetson
- Algorithm
  - New iterative schemes
  - New precision tweaks to increase accuracy
- Verification
  - Up-casting
  - Down-casting
  - Convergence
- Performance
  - Improve 16-bit kernels
  - Use 32-bit kernels on non-supporting hardware