Towards higher order numerical stochastic perturbation computation applied to the twisted Eguchi-Kawai model

Ken-Ichi Ishikawa (Hiroshima U.)

in collaboration with

Antonio González-Arroyo (Inst. de Fisica Teorica UAM/CSIC)

Issaku Kanamori (R-CCS,RIKEN)

Kanato Miyahana, Masanori Okawa, Ryoichiro Ueno (Hiroshima U.)

Based on

"Numerical stochastic perturbation theory applied to the twisted Eguchi-Kawai model"

arXiv:1902.09847

- We have implemented a NSPT simulation program for TEK model to extract PT coefficients of Wilson loops up to $O(g^8)$ in [arXiv:1902.09847]. Polynomial Matrix-Matrix multiplication operations are the most compute intensive part of the program. We improve the multiplication operation using FFT based convolution algorithm.
- We implement the CPU (Intel MKL with Xeon) and CUDA (CUDA v9.1 with P100) versions.
- We report the performance in this talk.

Contents

1. Numerical Stochastic Perturbation Theory (NSPT) for Reduced Twisted Eguchi-Kawai Model (TEK)
2. FFT-Convolution algorithm for polynomial matrix multiplication
3. Test results [CPU vs GPU version]
4. Summary

1. NSPT for Reduced Twisted Eguchi-Kawai Model (TEK)

- Numerical Stochastic Perturbation Theory (NSPT)
  - Stochastic Quantization, Based on Langevin algorithm : Parisi-Wu (1981)
  - Perturbation Theory based on Stochastic Quantization, (diagrammatic and analytic).
  - Application to Lattice theory => Numerical computation => NSPT
  - Lattice Gaute Theory NSPT efforts. F.Di Renzo, G.Marchesini, P. Marenzoni,E. Onofri, ...
  - Recent Studies
    - Including Dynamical Quarks : Renzo, Scorzato (2004-)
    - Perturbative evaluation of various improvement coefficients of lattice action and op.
    - Renormalon behavior : Bauer,Bali,Pineda et al.(2012-), Plaquette $O(\alpha^{35})$, Polyakov Line $O(\alpha^{20})\cdots$
    - Algorithm : Brida-Lusher(2017), Brida-Garofalo-Kennedy(2017)
    - Sigma model PCM(N), CP(N) ⋯ : Puhr-Fruckmann (Lat18)
- Large N Matrix Models
  - 't Hooft (1974), Witten(1980) : Large N expansion, Master Field
  - Eguchi-Kawai(1982)
    - Large N SU(N)/U(N) YM ⇔ Volume Reduction, Factoriztion : SD eq. for Wilson loops (loop eq.)
  - González-Arroyo, Okawa(1983)
    - Twisted Gauge Boundary condition. Improve Volume independence. Holds Z_N symmetry

$$\langle A(x)B(y)\rangle = \langle A(x)\rangle\langle B(y)\rangle + O\left(\frac{1}{N^a}\right)$$

- Motivation to Higher Order PT computation for TEK model
  - Relation between Large N limit and Renormalon (IR and Instanton) Gross-Witten,Wadia(1980),Gross-Matysin (1994), Buividovich,Dunne,Valgshev (2016)
  - Large N phase transition and complex saddles/instantons

M. García Pérez, 22 Jun, Plenary talk on Large N gauge theories

1. NPT for TEK Model

- Partition Function of TEK model

U_μ: SU(N) matrix

$$Z = \int \prod_{\mu=1}^{4} dU_\mu \, \mathrm{e}^{-S[U]} \qquad S[U] = \beta \sum_{\mu,\nu=1,\mu\neq\nu}^{4} \mathrm{Tr}\left[I - z_{\mu\nu} U_\mu U_\nu U_\mu^\dagger U_\nu^\dagger\right]$$

- MD eq. for Monte Carlo Simulation

$$\beta = \frac{1}{g^2}$$

$$\dot{U}_\mu = iP_\mu U_\mu, \dot{P}_\mu = F_\mu$$

$$F_\mu = i\beta\left[S_\mu - S_\mu^\dagger - \frac{1}{N}\mathrm{Tr}[S_\mu - S_\mu^\dagger]\right]$$

$$S_\mu = U_\mu\left[\sum_{\nu=1,\nu\neq\mu}^{4}\left(z_{\mu\nu} U_\nu U_\mu^\dagger U_\nu^\dagger + z_{\mu\nu}^* U_\nu^\dagger U_\mu^\dagger U_\nu\right)\right]$$

  - + Gauge dumping part (for NSPT)

- PT expansion for MD eq.

$$U_\mu = \sum_{k=0}^{\infty} \beta^{-k/2} U_\mu^{(k)} = U_\mu^{(0)} + gU_\mu^{(1)} + g^2 U_\mu^{(2)} + \cdots \qquad P_\mu = \beta^{1/2} \sum_{k=1}^{\infty} \beta^{-k/2} P_\mu^{(k)} = P_\mu^{(1)} + gP_\mu^{(2)} + g^2 P_\mu^{(3)} + \cdots$$

$U_\mu^{(0)} = \Gamma_\mu$: PT vacuum

- EoM for NSPT

$$F_\mu^{(k)} = i\left[S_\mu^{(k)} - S_\mu^{(k)\dagger} - \frac{1}{N}\mathrm{Tr}[S_\mu^{(k)} - S_\mu^{(k)\dagger}]\right]$$

$$\acute{U}_\mu^{(k)} = i\left(P_\mu \star U_\mu\right)^{(k)}, \acute{P}_\mu^{(k)} = F_\mu^{(k)} \qquad S_\mu^{(k)} = \left(U_\mu \star \left[\sum_{\nu=1,\nu\neq\mu}^{4}\left(z_{\mu\nu} U_\nu \star U_\mu^\dagger \star U_\nu^\dagger + z_{\mu\nu}^* U_\nu^\dagger \star U_\mu^\dagger \star U_\nu\right)\right]\right)^{(k)}$$

$\star$-Product : Convolution = Polynomial multiplication

Numerically solve EoM for NSPT order by order with Gaussian noise momentum to generate ensemble $\{U_\mu^{(k)}\}$.

1. NSPT for TEK Model

- Observable: Wilson Loops
- NPT definition in TEK model

$$W_{\mu\nu}(R,T) = \frac{1}{N}\left(z_{\mu\nu}\right)^{RT}\mathrm{Tr}\left[\left(U_\mu\right)^R(U_\nu)^T\left(U_\mu^\dagger\right)^R(U_\nu^\dagger)^T\right] = \sum_{k=0}^{\infty} g^k W_{\mu\nu}^{(k)}(R,T)$$

- NSPT version

$$W_{\mu\nu}^{(k)}(R,T) = \frac{1}{N}\left(z_{\mu\nu}\right)^{RT}\mathrm{Tr}\left[\left(\left(U_\mu\right)^{\star R}\star(U_\nu)^{\star T}\star\left(U_\mu^\dagger\right)^{\star R}\star\left(U_\nu^\dagger\right)^{\star T}\right)^{(k)}\right]$$

- Monte Carlo for NSPT

1. Numerically solve EoM for NSPT order by order with Gaussian noise momentum to generate ensemble $\{U_\mu^{(k)}\}$.
2. Evaluate Observable on the each configuration $U_\mu^{(k)} \to W_{\mu\nu}^{(k)}(R,T)$
3. Take statistical average for $\left\langle W_{\mu\nu}^{(k)}(R,T)\right\rangle$

1. NSPT for TEK Model

- Computationally Intensive part of NSPT for TEK model in large N region is Perturbative Matrix-Matrix multiplication (convolution product)
  - A, B, C : Polynomial Matrix (N x N) N= O(100)-O(500) for Large N

$$A = \sum_{k=0}^{\infty} g^k A^{(k)} = A^{(0)} + gA^{(1)} + g^2 A^{(2)} + \cdots \qquad B = \sum_{k=0}^{\infty} g^k B^{(k)} = B^{(0)} + gB^{(1)} + g^2 B^{(2)} + \cdots$$

$$AB = C = \sum_{k=0}^{\infty} g^k C^{(k)} = C^{(0)} + gC^{(1)} + g^2 C^{(2)} + \cdots$$

$$C^{(k)} = \sum_{\ell=0}^{k} A^{(\ell)} B^{(k-\ell)} \equiv (A \star B)^{(k)}$$

$\star$-Product : Convolution = Polynomial multiplication

  - Naive Computational Cost truncated at $g^{N_{trunc}}$

$$\frac{1}{2}(N_{trunc}+1) \times N_{trunc} \times [\text{Complex Matrix-Matrix Multiplication=ZGEMM(BLAS)}]$$
$$\propto O(N_{trunc}^2)$$

  - ZGEMM cost scales as $O(N^3)$ (N: matrix size)

- We have implemented the NSPT algorithm for TEK model with the naive matrix polynomial convolution algorithm and evaluated Wilson loops at O(g^8) in arXiv:1902.09847. For a higher order NSPT, N_{trunc} = O(100), FFT-convolution is preferable.

2. FFT-Convolution algorithm for polynomial matrix multiplication

- Polynomial multiplication can be accelerated by using FFT-convolution (FFT-CONV) algorithm.
- We can extend FFT-CONV to matrix polynomial multiplication.
- However, the ordering of array rank, the matrix indices and PT order index, could affects the computer performance.
  - Matrix-Matrix multiplication : BLAS ZGEMM is typically optimized for matrix-index major data layout.
  - FFT / invFFT : is typically optimized for order index major data format.

    A(ic,jc,ipt) VS A(ipt,ic,jc) (Fortran form)

    (ic,jc): SU(N) matrix indices, ipt : PT-order index
  - We simply insert a transpose operation between FFT/invFFT and ZGEMM multiplication to get the best performance from BLAS and FFT libralies.

2. FFT-Convolution algorithm

$$\boxed{O(N_{trunc}^2) \Rightarrow O(N_{trunc}\log(N_{trunc}))}$$

- ZGEMM_NSPT : ZGEMM operation for polynomial matrices.

$$C^{(k)} = \alpha\,(A \star B)^{(k)} + \beta C^{(k)} \quad \text{for } k = 0, \dots, N_{trunc}\ (k = 0, \dots, N_p - 1)$$

- c.f BLAS ZGEMM : $C = \alpha\, AB + \beta C$

- We employ the *linear convolution algorithm* (trunc'd at $N_p - 1$).

0. We employ FFT major format. A(ipt,ic,jc) = $A_{ic,jc}^{(ipt)}$
1. Extend the PT series length of A and B by two and fill zero on the extended coefficient matrices.

$$\left(A^{(0)}, A^{(1)}, \dots, A^{(Np-1)}\right) \rightarrow \left(A^{(0)}, A^{(2)}, \dots, A^{(Np-1)}, A^{(Np)}, \dots, A^{(2Np-1)}\right)$$

with $A^{(k)} = 0$ for $k = N_p \ \dots\ 2N_p - 1$

Similarly extends B.

2. FFT(length 2Np) A and B for each color index independently

$$\tilde{A}_{ij}^{(p)} = \sum_{k=0}^{2Np-1} A_{ij}^{(k)}\, w^{kp},\ \tilde{B}_{ij}^{(p)} = \sum_{k=0}^{2Np-1} B_{ij}^{(k)}\, w^{kp}, w = \exp\left(\frac{2\pi i}{2Np}\right)$$

3. Transpose the color indices and PT-order index to have matrix major format. (tildeA(ipt,ic,jc) => tildeA(ic,jc,ipt))
4. Pointwise GEMM for each coefficient. $\tilde{C}^{(p)} = \tilde{A}^{(p)} B^{(p)}$ for $p = 0, \dots, 2N_p - 1$.
5. Transpose to FFT major format (tildeC(ic,jc,ipt) => tildeC(ipt,ic,jc)).
6. Inverse FFT for each color index independently.

$$C_{ij}^{(k)} = \frac{1}{(2N_p)} \sum_{p=0}^{2Np-1} \tilde{C}_{ij}^{(p)}\, w^{-kp}$$

7. Truncate extended higher order components.
8. alpha and beta operations.

2. FFT-Convolution algorithm

- Machine and Libraries used.
  - Machine
    - ITO subsystem A and B at Kyushu University.

ITO	Subsystem A	Subsystem B
CPU/node	Intel Xeon Gold 6154 (Skylake-SP) (**3.0** GHz (Turbo 3.7 GHz), 18 core) × 2	Intel Xeon Gold 6140 (Skylake-SP) (**2.3** GHz (Turbo 3.7 GHz), 18 core) × 2
GPU/node	NA	NVIDIA Tesla P100 (Pascal) (1,328 - 1,480 MHz, 56 SM (3584 CUDA core)) × 4
DP Peak Perform.	3,456 GFLOPS	CPU : 2,649.6 GFLOPS GPU : 5.3 x TFLOPSx4 = 21.2 TFLOPS
Memory Bandwidth	255.9 GB/sec	CPU:255.9 GB/sec GPU: 732GB/sec x 4 = 2928 GB/sec

- Peak performance ratio:

Subsystem B GPU performance / Subsystem A CPU performance

= 6.13

- Peak memory bandwidth ratio:

Subsystem B GPU Mem B.W. / Subsystem A CPU Mem B.W.

= 11.4

2. FFT-Convolution algorithm

- Machine and Libraries used.
  - Libraries / Languages
    - CPU version (For SubSystem A)
      - Intel Fortran v18.0.0.128
      - Libs : Intel MKL contains BLAS, FFT(Dfti)
        - BLAS extension for transposition op.: MKL_ZOMATCOPY
        - BLAS extension for batch mode ZGEMM : ZGEMM_BATCH
        - batch mode is available in MKL FFT(Dfti)
    - GPU version (For SubSystem B)
      - Intel Fortran v18.0.0.128 + CUDA v9.1.85
      - Libs: cuBLAS, cuFFT
        - cuBLAS extension for transposition op.: cublasZgeam
        - CUDA streams are used for cublasZgemm for batch mode
        - batch mode is available in cuFFT
        - 4GPU cards are used for each direction U_μ, $\mu = 1,2,3,4$

3. Test results [CPU vs GPU version]

- We compare the timing of the CPU version and GPU version.
- The parameters of the TEK NSPT program are

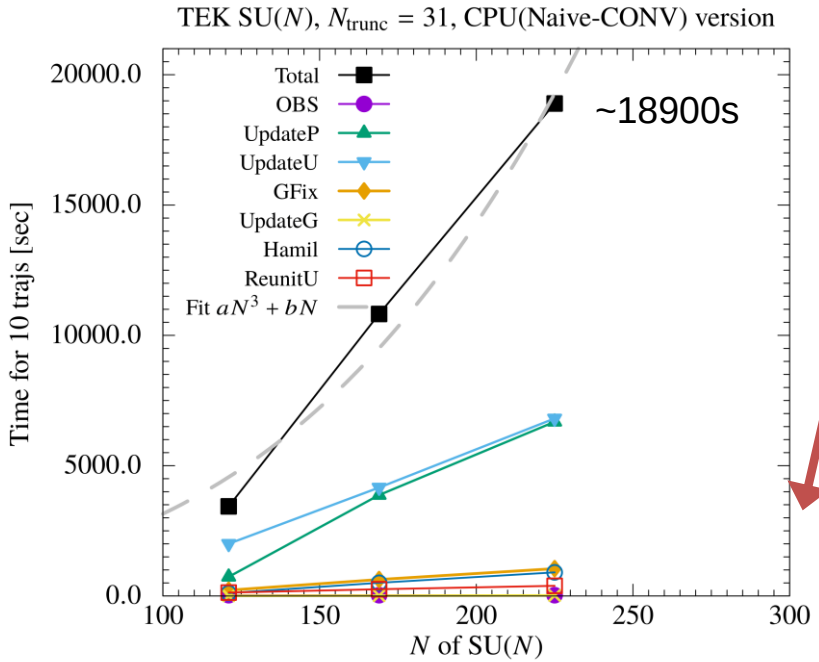
TEK-NPT parameters	MD steps N_MD	N of Color SU(N)	PT truncation order N_trunc	
CPU(Naive-CONV)	40	121, 169, 225	7,15,31	CPU: Subsystem A 36 OpenMP threads
CPU(FFT-CONV)	40	121, 169, 225	7,15,31,63	CPU: Subsystem A 36 OpenMP threads
CUDA(FFT-CONV)	40	121, 169, 225, 289	15,31,63	GPU: Subsystem B 4 GPU + 36 OpenMP threads

- We measure the timing for 10 trajectories including config. generation and Wilson loop computation.
- We show
  - Timing scaling on N and N_trunc
  - Timing breakdown in the prgram: ZGEMM, Transpose, FFT in the program
  - Timing comparison between CPU and GPU versions.

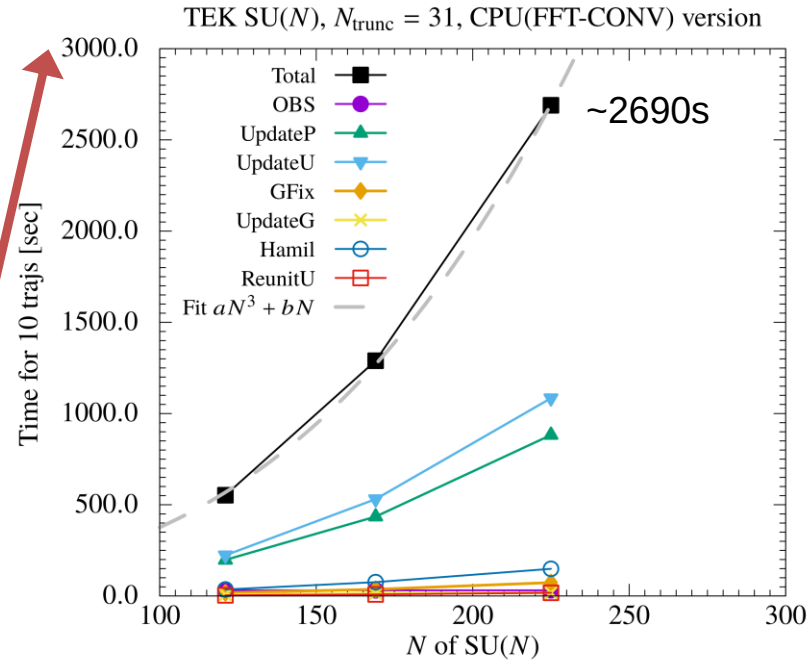
3. Test results [CPU vs GPU version]

- Comparison between CPU versions (Naive-CONV vs FFT-CONV)
- N dependence (Ntrunc=31)

CPU(Naive-CONV)



CPU(FFT-CONV)



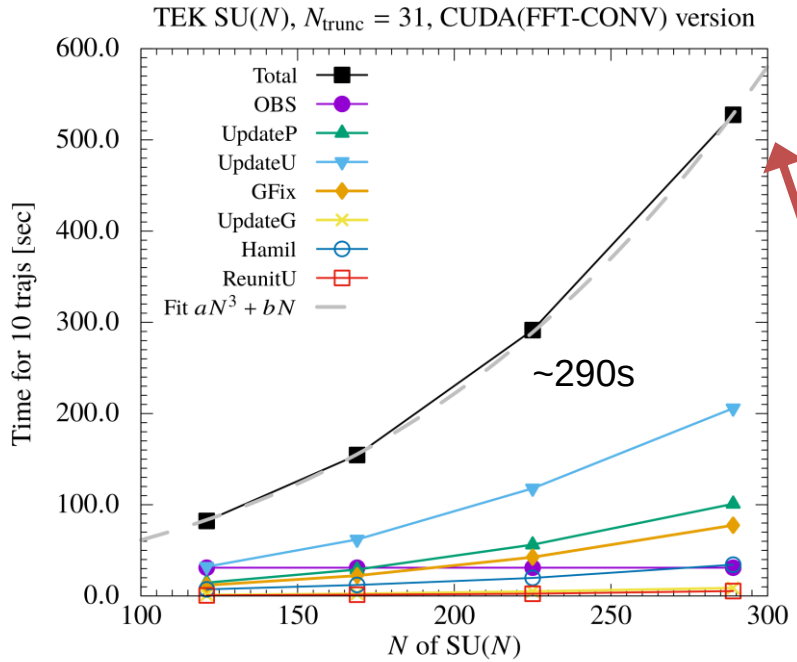
- Naive-CONV version is too slow for higher order with large N cases. A factor 7 improvement with FFT-CONV for N=225 at Ntrunc=31 on CPU.

Hereafter we only compare FFT-CONV vers.

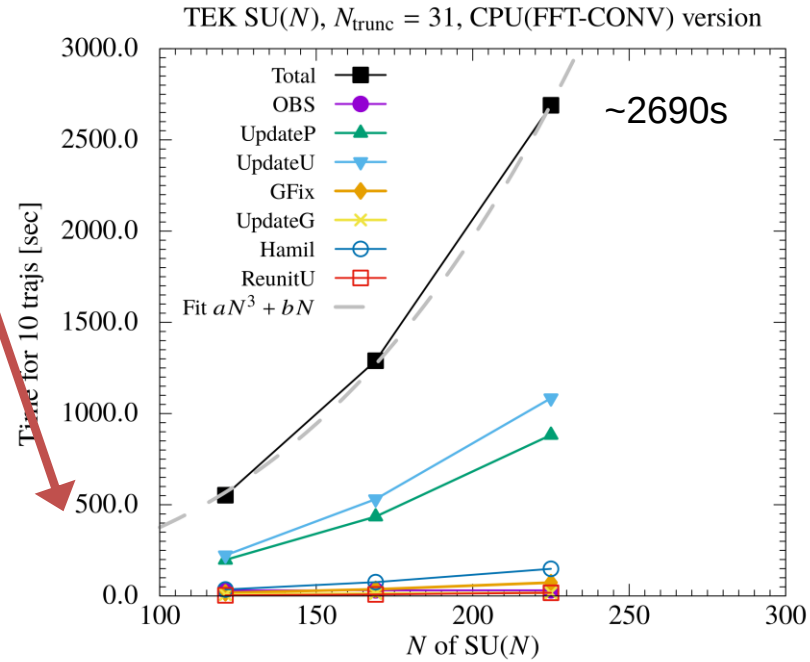
3. Test results [CPU vs GPU version]

- Comparison between CPU and GPU versions (FFT-CONV)
- N dependence (Ntrunc=31)

GPU(FFT-CONV)



CPU(FFT-CONV)

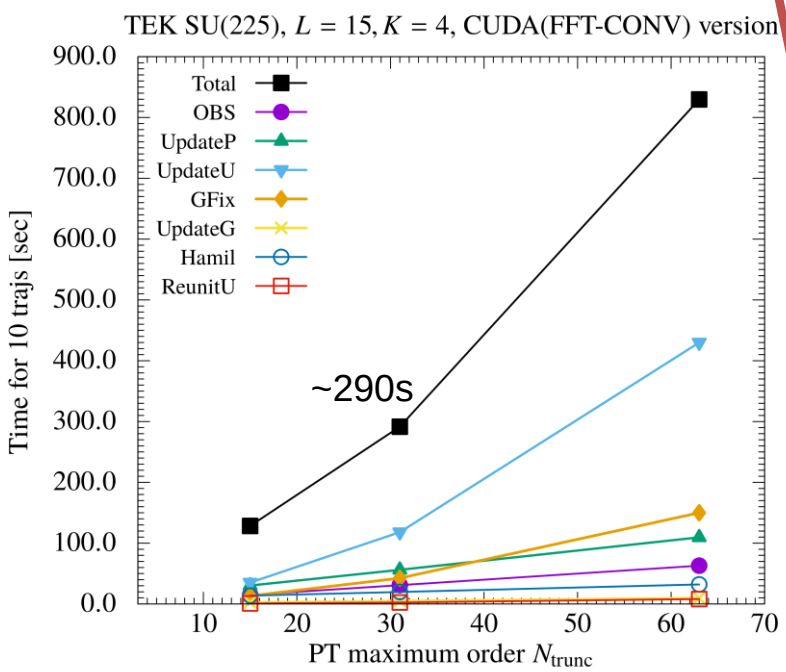


- GPU(FFT-CONV) version is faster by a factor 9 than CPU(FFT-CONV) version for N=225 at Ntrunc=31.
- Consistent with O(N^3) scaling for SU(N) size

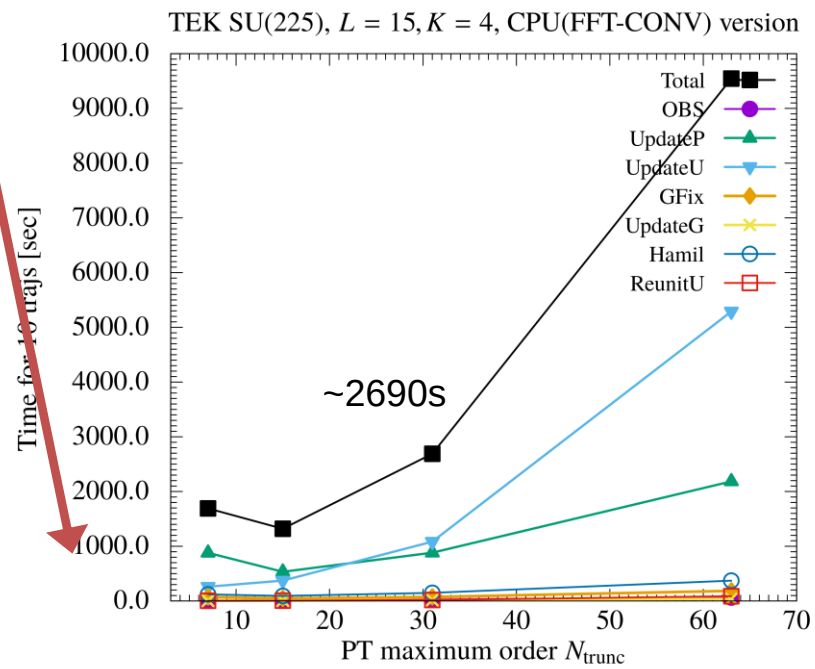
3. Test results [CPU vs GPU version]

- Comparison between CPU and GPU versions (FFT-CONV)
- Ntrunc dependence (N=225)

GPU(FFT-CONV)



CPU(FFT-CONV)

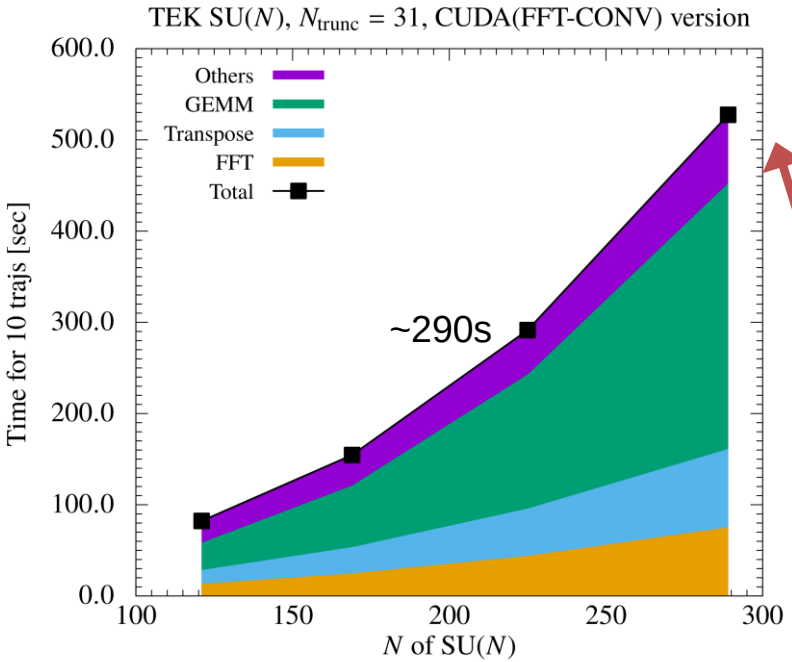


- GPU(FFT-CONV) has a milder Ntrunc dependence.
- CPU(FFT-CONV) also show milde dependence for Ntrunc >30, but too slow (by a factor 9).

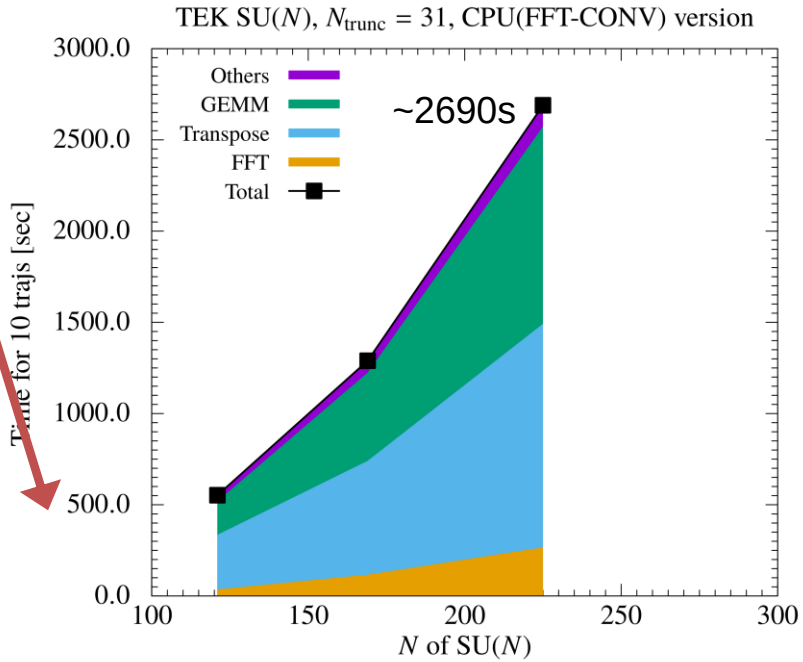
3. Test results [CPU vs GPU version]

- Comparison between CPU and GPU versions (FFT-CONV)
- N dependence (Ntrunc=31) GEMM/TRANSP/FFT breakdown

GPU(FFT-CONV)



CPU(FFT-CONV)

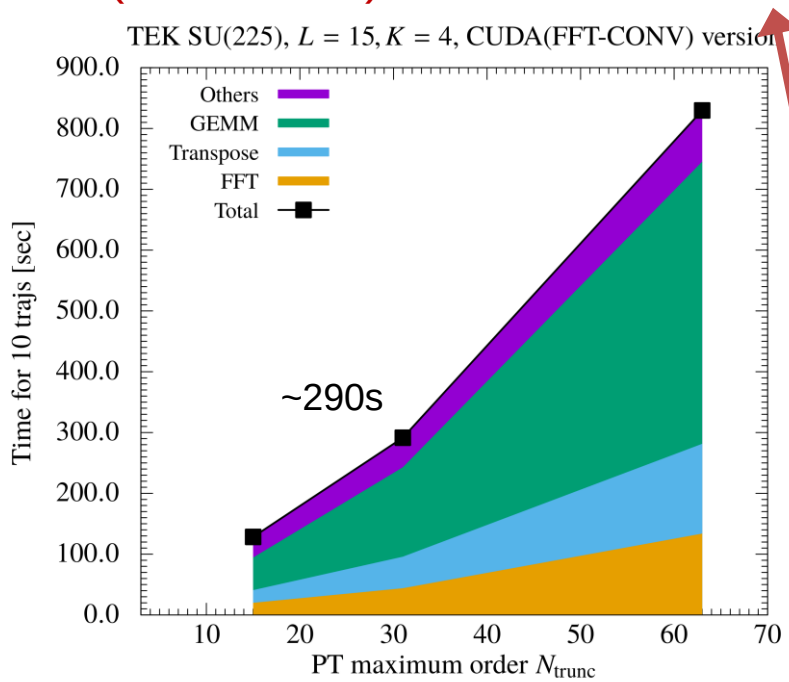


- CPU(FFT-CONV) version: Transpose operation dominates the 54%-46% of the timing.
- GPU(FFT-CONV) version: ZGEMM dominates 36%-55%.

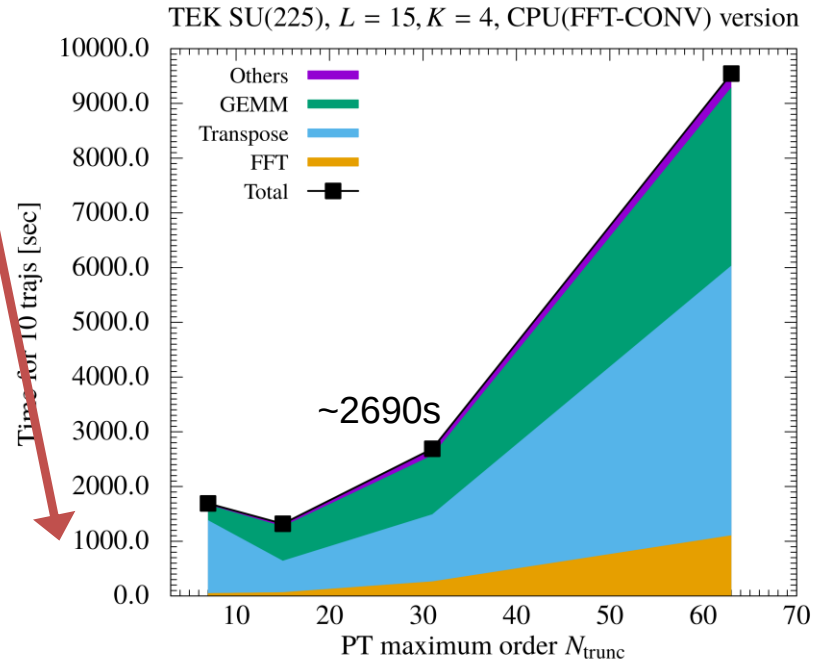
3. Test results [CPU vs GPU version]

- Comparison between CPU and GPU versions (FFT-CONV)
- Ntrunc dependence (N=225) GEMM/TRANSP/FFT breakdown

GPU(FFT-CONV)



CPU(FFT-CONV)



- CPU(FFT-CONV) has a large overhead for the transpose operation (Skyblue) at low Ntrunc=7. We employ outplace transpose routine. This may not work optimally for narrow matrix transpose A(0:7,225,225)⇔A(225,225,0:7) on CPU.

3. Test results [CPU vs GPU version]

- Comparison between CPU and GPU versions (FFT-CONV)
- Timing Comparison (N=225, Ntrunc=31)

	Total [sec]	ZGEMM[sec]	Transpose[sec]	FFT[sec]
CPU(FFT-CONV)	2689	1080	1226	265
GPU(FFT-CONV)	291	147	52	44
CPU/GPU ratio	9.2	7.3	23.6	6.0

- The peak performance ratio of SubSystem A for CPU (3.456TFLOPS) and Subsystem B for GPU (21.2 TFLOPS) is 6.13. The mem B.W. ratio (2928 [GPU]/255.9[CPU]) is 11.4.
- ZGEMM is compute intensive operation and the performance would reflect the pure peak computational performance of the systems. Thus the measured ZGEMM ratio is comparable to the peak performance ratio.
- Transpose operation is memory or cache memory bandwidth bounded. The measured transp. timing ratio, 23.6, differs by a factor two the expected mem B.W. ratio 11.4.
- Total improvement of GPU(FFT-CONV) from CPU(FFT-CONV) is a factor 9.2, which is roughly an average (arithmetic/geometric) of the peak comput. perf ratio and the mem BW ratio of the two subsystems.

4. Summary

- In order to investigate the behavior of higher order perturbative coefficients of Large N limit of YM theories, we employ NSPT algorithm to TEK model.
- In order to get coefficients at O(g^30) or higher, the performance of the polynomial matrix multiplication is the bottleneck of the NSPT TEK program.
- We have implemented the FFT based convolution algorithm (FFT-CONV) to improve the polynomial matrix multiplication.
- The performance gain was drastic on CPU version.
- Using GPUs, we could gain further speed up within a single node, which is quite preferable as the TEK model does not have d.o.f of lattice sites to be easily parallelized.