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

Antonio González-Arroyo[a,b]**, Issaku Kanamori**[c]**, Ken-Ichi Ishikawa**[†d,e]**,**
Kanata Miyahana[d]**, Masanori Okawa**[d]**, Ryoichiro Ueno**[d]

[a] *Instituto de Física Teórica UAM/CSIC, Nicolás Cabrera 13-15, E-28049 Universidad Autónoma de Madrid, Madrid, Spain*

[b] *Departamento de Física Teórica, Módulo 15, Cantoblanco, E-28049 Universidad Autónoma de Madrid, Madrid, Spain*

[c] *RIKEN Center for Computational Science, Kobe 650-0047, Japan*

[d] *Graduate School of Science, Hiroshima University, Higashi-Hiroshima 739-8526, Japan*

[e] *Core of Research for the Energetic Universe, Hiroshima University, Higashi-Hiroshima 739-8526, Japan*

E-mail: ishikawa@theo.phys.sci.hiroshima-u.ac.jp

We have evaluated perturbation coefficients of Wilson loops up to $O(g^8)$ for the four-dimensional twisted Eguchi-Kawai model using the numerical stochastic perturbation theory (NSPT) in [1]. In this talk we present a progress report on the higher order calculation up to $O(g^{63})$, for which we apply a fast Fourier transformation (FFT) based convolution algorithm to the multiplication of polynomial matrices in the NSPT aiming for higher order calculation. We compare two implementations with the CPU-only version and the GPU version of the FFT based convolution algorithm, and find a factor 9 improvement on the computational speed of the NSPT algorithm with SU($N = 225$) at $O(g^{31})$. The perturbation order dependence of the computational time, we investigate it up to $O(g^{63})$, shows a mild scaling behavior on the truncation order.

37th International Symposium on Lattice Field Theory - Lattice2019
16-22 June 2019
Wuhan, China

1. Introduction

The numerical stochastic perturbation theory (NSPT) is one of the numerical method to compute a higher order perturbative series of lattice quantum field theories [2, 3, 4, 5, 6, 7]. Investigating the higher order behavior of perturbation coefficients in terms of expansion order could reveal the non-perturbative properties such as instanton, non-trivial complex saddles, and renormalon effects.

The large N limit of Yang-Mills theory is one of the tools to investigate the non-perturbative aspects keeping the non-perturbative properties with a great simplification (see for recent studies [8]). To tackle the non-perturbative aspects of the large-N Yang-Mills theory through the higher order behavior of the perturbative coefficients, we have evaluated the coefficients of Wilson loops up to $O(g^8)$ for the four-dimensional twisted Eguchi-Kawai (TEK) model [9, 10] using the NSPT [1]. The eighth order of the expansion is not yet sufficient to study the higher order behavior, though, we found that the variance of the coefficients is well controlled by the factorization property of large-N expansion, and there is no *Pepe* effect by which we can access the higher order coefficients with a moderate statistics. We also investigated the computational cost of the NSTP for the TEK model in the large-N limit, and found that the computational cost of the perturbation series multiplication can be an obstacle to access the higher order coefficients in the large-N limit. This is because that the cost scales as N_{trunc}^2 for the multiplication of two perturbative series truncated at $O(g^{N_{\text{trunc}}})$ with the naive multiplication algorithm.

In order to improve the scaling property in a higher order perturbation truncation, we implement a Fast Fourier Transformation based convolution algorithm (FFT-CONV) for the perturbation polynomial multiplication. Because we expand the $SU(N)$ matrix perturbatively in the TEK model, the series expansion becomes polynomial matrices, i.e. the coefficients are $N \times N$ matrices, and the implementation of the FFT-CONV is not trivial.

In this talk we present the FFT based convolution algorithm for the polynomial matrix multiplication, and apply this algorithm to the NSPT algorithm of the TEK model. We first compare the naive-multiplication algorithm and the FFT-CONV algorithm on a CPU architecture machine. Having established a high efficiency of the FFT-CONV algorithm, we further implement the FFT-CONV for a GPU system, and compare the performance between the CPU and GPU versions.

2. FFT based convolution algorithm

The TEK model is defined by

$$Z = \int \prod_{\mu=1}^{4} dU_\mu \exp\left[-\beta \sum_{\mu \neq \nu} \text{Tr}\left[1 - z_{\mu\nu} U_\mu U_\nu U_\mu^\dagger U_\nu^\dagger \right] \right], \tag{2.1}$$

where U_μ are SU(N) matrices and $z_{\mu\nu}$ is an anti-symmetric tensor called the twist-tensor [9, 10]. The non-perturbative simulation algorithm based on the molecular dynamics algorithm, such as HMC, relies on the following equation of motion.

$$\dot{U}_\mu = iP_\mu U_\mu, \qquad F_\mu = i\beta\left[S_\mu - S_\mu^\dagger - \text{Tr}[S_\mu - S_\mu^\dagger]/N \right],$$
$$\dot{P}_\mu = F_\mu, \qquad S_\mu = U_\mu\left[\sum_{\nu \neq \mu}\left(z_{\mu\nu} U_\nu U_\mu^\dagger U_\nu^\dagger - z_{\mu\nu}^* U_\nu^\dagger U_\mu^\dagger U_\nu \right) \right], \tag{2.2}$$

where dot $\dot{}$ means the fictitious time derivative. The NSPT algorithm is derived from the equation of motion by expanding them perturbatively in $g = 1/\sqrt{\beta}$. Substituting $U_\mu = \sum_{k=0} g^k U_\mu^{(k)}$ and $P_\mu = (1/g)\sum_{k=1} g^k P_\mu^{(k)}$ into (2.2), expanding them and rescaling the fictitious time, we have

$$\dot{U}_\mu^{(k)} = i(P_\mu \star U_\mu)^{(k)}, \quad \dot{P}_\mu^{(k)} = F_\mu^{(k)}, \quad [\text{+ stochastic gauge fixing part}]$$

$$F_\mu^{(k)} = i\beta \left[S_\mu^{(k)} - S_\mu^{(k)\dagger} - \text{Tr}[S_\mu^{(k)} - S_\mu^{(k)\dagger}]/N \right],$$

$$S_\mu^{(k)} = \left(U_\mu \star \left[\sum_{\nu \neq \mu} \left(z_{\mu\nu} U_\nu \star U_\mu^\dagger \star U_\nu^\dagger - z_{\mu\nu}^* U_\nu^\dagger \star U_\mu^\dagger \star U_\nu \right) \right] \right)^{(k)}, \tag{2.3}$$

where the details of the stochastic gauge fixing part are omitted. The tower of the equations is truncated at $O(g^{N_{\text{trunc}}})$ and is solved with a symplectic integrator with periodic momentum refreshment to yield the statistical ensemble for the perturbative coefficients. The star $\star$ product is the convolution product explained in the following.

Let A, B, C are $N \times N$ polynomial matrices truncated at $O(g^{N_{\text{trunc}}})$ defined by

$$A = \sum_{k=0}^{N_{\text{trunc}}} g^k A^{(k)}, \quad B = \sum_{k=0}^{N_{\text{trunc}}} g^k B^{(k)}, \quad C = \sum_{k=0}^{N_{\text{trunc}}} g^k C^{(k)}. \tag{2.4}$$

The matrix product $C = AB$ is expanded and $C^{(k)}$'s are expressed with the convolution of $A^{(k)}$ and $B^{(k)}$ as

$$C^{(k)} = \sum_{\ell=0}^{k} A^{(\ell)} B^{(k-l)} \equiv (A \star B)^{(k)}, \tag{2.5}$$

where we define $\star$ as the convolution product. As the coefficients $A^{(k)}$ and $B^{(k)}$ are $N \times N$ matrices the ordering of multiplication have to be kept. The computational cost of the naive convolution scales with $N^3 \times N_{\text{trunc}}^2$, where N^3 comes from the $N \times N$ matrix-matrix multiplication and N_{trunc}^2 from the naive convolution.

We introduce the FFT based convolution algorithm (FFT-CONV) in the following. In order to treat the truncated polynomial multiplication, the FFT-CONV requires twice a longer series padded with zero coefficients to avoid the overlapping effect due to the periodicity of FFT. This type of method is called the linear convolution algorithm. The FFT is known to be very efficient when the length of data is a power of 2. We restrict the order of truncation N_{trunc} to be $N_{\text{trunc}} = N_p - 1$ with $N_p = 2^s$ and $s \in \mathbb{N}$. The FFT-CONV for the polynomial matrix multiplication is summarized as follows.

Step 1 Double the perturbative series length with zero coefficient matrices.

$$\left\{ A^{(0)}, A^{(1)}, \ldots, A^{(N_p-1)} \right\} \Rightarrow \left\{ A^{(0)}, A^{(1)}, \ldots, A^{(N_p-1)}, A^{(N_p)}, \ldots, A^{(2N_p-1)} \right\}, \tag{2.6}$$

with $A^{(j)} = 0, (j = N_p, \ldots, 2N_p - 1)$. Double the length for $B^{(k)}$ similarly.

Step 2 Do FFT with the length $2N_p$ on $\{A^{(k)}\}$ and $\{B^{(k)}\}$ to obtain the Fourier coefficients $\{\tilde{A}^{(p)}\}$ and $\{\tilde{B}^{(p)}\}$.

$$\tilde{A}^{(p)} = \sum_{k=0}^{2N_p-1} A^{(k)} w^{kp}, \quad \tilde{B}^{(p)} = \sum_{k=0}^{2N_p-1} B^{(k)} w^{kp}, \quad w \equiv \exp\left(\frac{2\pi i}{2N_p}\right). \tag{2.7}$$

	ITO Subsystem A	ITO 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, 3584 CUDA cores)×4
DP peak perf./node	3,456 GFLOPS	CPU: 2,649.6 GFLOPS GPU: 5.3 TFLOPS × 4 = 21.2 TFLOPS
Memory B.W./node	255.9 GB/sec	CPU: 255.9 GB/sec GPU: 732GB/sec × 4 = 2928 GB/sec

Table 1: System description of the ITO supercomputer.

Step 3 Evaluate $\left\{\tilde{C}^{(p)}\right\}$ with the Fourier-mode-wise multiplication.

$$\tilde{C}^{(p)} = \tilde{A}^{(p)}\tilde{B}^{(p)}, \tag{2.8}$$

Step 4 Do inverse FFT on $\left\{\tilde{C}^{(p)}\right\}$ to obtain $\left\{C^{(k)} = (A \star B)^{(k)}\right\}$.

$$C^{(k)} = \frac{1}{2N_p}\sum_{p=0}^{2N_p-1}\tilde{C}^{(p)}w^{-kp}. \tag{2.9}$$

Step 5 Discard higher order coefficients $C^{(k)}, (k = N_p, \ldots, 2N_p - 1)$.

The computational cost of the FFT-CONV algorithm scales with $N^3 \times N_{\text{trunc}}\log(N_{\text{trunc}})$.

Several tricks should be inserted in the above steps to get a more better numerical performance in larger N. Most of the FFT libraries, such as FFTW, would be optimized for transforming scalar data. It is the case we have to move the perturbation order index to the major index of the array to extract the best performance of the FFT library. Similarly the matrix-matrix multiplication scans the gauge group indices, the group indices should be located in the major indices of the array to have the best performance. Therefore we need to permute the order of the gauge group and perturbation order indices before and after the FFT, mode-wise matrix-matrix multiplication, and inverse FFT steps to maximize the performance of fundamental operations of FFT and matrix-matrix multiplication.

A further improvement can be achieved by noticing the independence or parallelism of the operations. The FFT and inverse FFT operations can be carried out for each gauge group matrix element independently. The batch mode of FFT can be utilized if the library supports. Similarly the matrix-matrix multiplication for each frequency mode is also independent each other, the batch mode of the matrix-matrix multiplication is also applicable if it supports.

We implemented the FFT-CONV algorithm for both CPU and GPU architectures. As the benchmark we measure and compare the timing of NSPT runs for 10 trajectories at several N for SU(N) and truncation order N_{trunc}. The machines and libraries, and the benchmark results are shown in the next section.

3. Numerical Tests

3.1 Machine and Libraries

We employ two subsystems A and B of the ITO supercomputer system at Kyushu University for the benchmark. The details of the systems are tabulated in Table 1. The subsystem A is used

	CPU version (Subsystem A)	GPU version (Subsystem B)
Compiler	Intel Fortran v18.0.0.128	Intel Fortran v18.0.0.128 + CUDA v9.1.85
FFT	Intel MKL FFT (Dfti) batch mode available	cuFFT batch mode available
ZGEMM	Intel MKL BLAS ZGEMM_BATCH batch mode	cuBLAS cublasZgemm batch mode using CUDA streams
Transpose	Intel MKL BLAS MKL_ZOMATCOPY BLAS extension	cuBLAS cublasZgeam BLAS extension

Table 2: Compilers and libraries

	MD steps	N of SU(N)	N_{trunc}
CPU(Naive-CONV)	40	121, 169, 225	7, 15, 31
CPU(FFT-CONV)	40	121, 169, 225	7, 15, 31, 63
GPU(FFT-CONV)	40	121, 169, 225, 289	15, 31, 63

Table 3: Parameters

for benchmarking the CPU version and the subsystem B for the GPU version. We employ single node. In order to utilize the four GPU cards on the node, we assign computational tasks, which are independent in the four-dimensional direction, on each GPU card so that the FFT-CONV algorithm for a matrix-matrix multiplication is carried out within a GPU card.

Table 2 shows the compilers and libraries for FFT and matrix-matrix multiplication. The ZGEMM subroutine from the BLAS library is employed for the matrix-matrix multiplication. In order to permute the group indices and the perturbation index in the FFT-CONV algorithm, we employ BLAS extension subroutines to extract the best performance on each architecture.

3.2 Benchmark Results

The parameters used in benchmarking are shown Table 3. All CPU cores are utilized using OpenMP multithreading.

We first measure the performance of the naive convolution (Naive-CONV) algorithm and the FFT-CONV algorithm on the CPU system. Figure 1 shows the timing for 10 trajectories of the NSPT algorithm at $N_{\text{trunc}} = 31$ as a function of the color size N. Total time (black squares) accumulates the breakup of various functions involved in the NSPT algorithm. The most timing consuming part is the gauge momentum update (green up-triangles) and the gauge link update (cyan down-triangles). The FFT-CONV version (b) is faster by a factor 7 than the Naive-CONV version (a) at $N = 225$ on the CPU architecture. The gray dashed line shows the scaling with N^3, which originates from the computational cost of matrix-matrix multiplication. The total timing of the FFT-CONV fits with the scaling.

Having observed a high performance for the FFT-CONV on the CPU, we compare the FFT-CONV between the CPU version (Fig. 1 (b)) and the GPU version (Fig. 2 (a)). A factor 9 improvement on the timing is achieved at $N = 225, N_{\text{trunc}} = 31$ by employing the GPU cards. The scaling is consistent with N^3 (gray dashed line). Figure 2 (b) shows the breakup on the fundamental operations of FFT, GEMM, and transpose, involved in the NSPT. The most timing consuming part
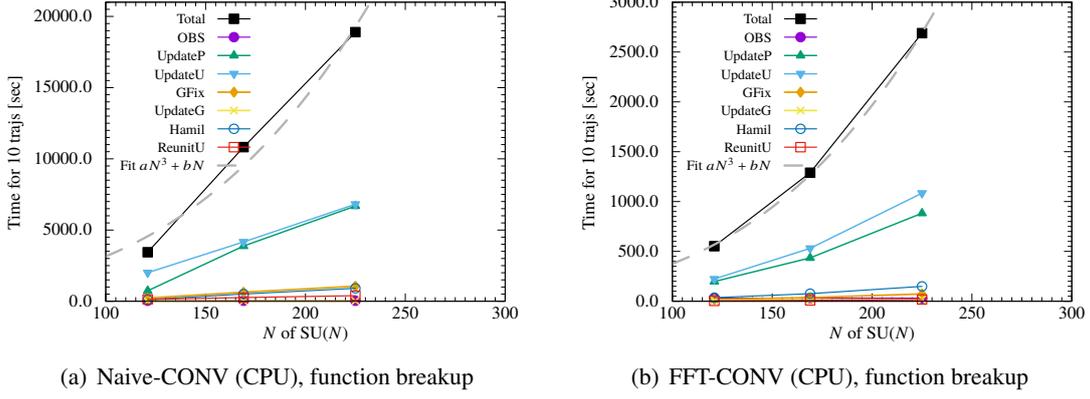
(a) Naive-CONV (CPU), function breakup

(b) FFT-CONV (CPU), function breakup

Figure 1: Timing comparison between the Naive-CONV and FFT-CONV on the CPU at $N_{\text{trunc}} = 31$.

(a) FFT-CONV (GPU), function breakup

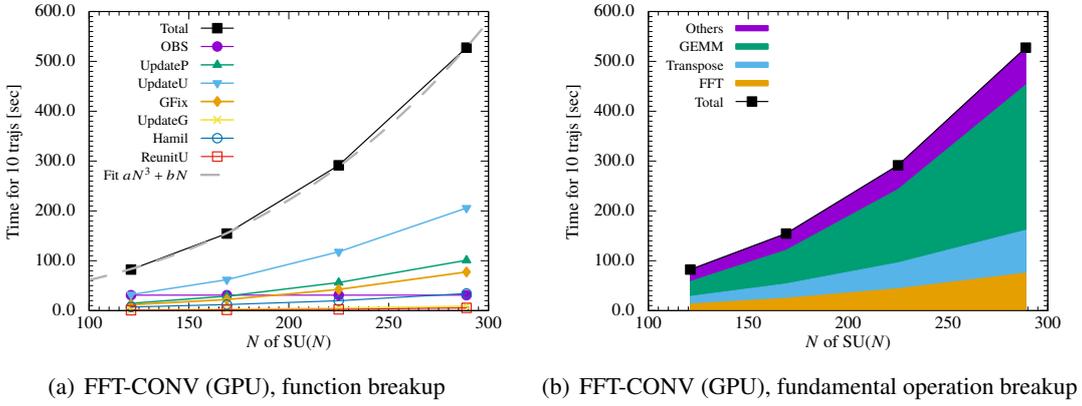(b) FFT-CONV (GPU), fundamental operation breakup

Figure 2: Timing of the FFT-CONV on the GPU architecture at $N_{\text{trunc}} = 31$.

is in the matrix-matrix multiplication of ZGEMM. On the GPU architecture, ZGEMM is quite highly optimized in the cuBLAS library, the timing is already at the best performance.

Figure 3 shows the timing as a function of truncation order N_{trunc}. The GPU version shows a milder scaling behavior, while the CPU version has less trivial behavior in the lower truncation region. Looking at the timing breakup with fundamental operations, the CPU version has a large overhead in the transpose operation, especially in the lower truncation order. The shape of the array with the color and perturbation order indices is quite narrow at lower N_{trunc}, the CPU version might have a penalty in the transpose operation for such a narrow array because its performance is sensitive to the cache and the memory subsystem of the architecture. On the other hand the GPU version does not show such a irregular behavior, we expect that the transpose operation cublasZgeam of cuBLAS extension is well optimized even for such narrow arrays.

The peak performance ratio of the subsystem B (GPU) to A (CPU) is about 6, and the peak memory bandwidth ratio (B/A) is about 11. Observed improvement factor of 9 at $N = 225, N_{\text{trunc}} = 31$ is roughly at the arithmetic or geometric average of the two peak ratio. We observed a consistent performance improvement using the GPU architecture for the FFT-CONV algorithm.

4. Summary

We have implemented the FFT based convolution algorithm for the polynomial matrix mul-

5

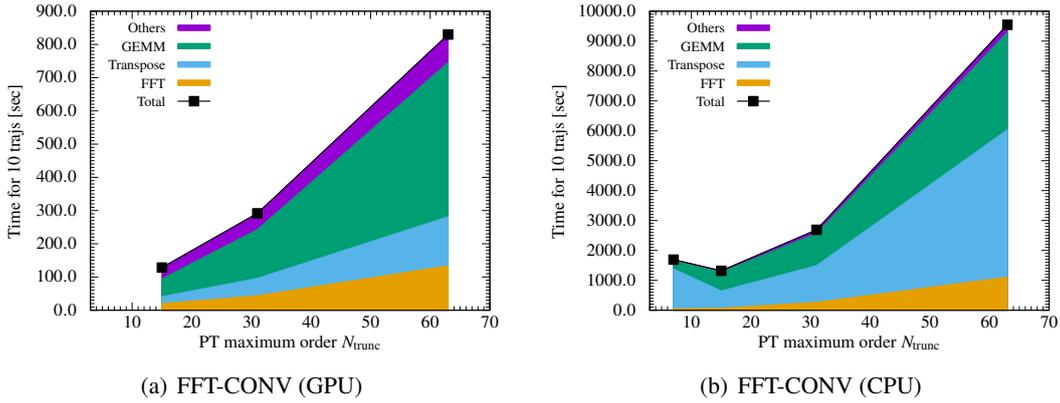(a) FFT-CONV (GPU)   (b) FFT-CONV (CPU)

Figure 3: Timing comparison of the FFT-CONV between GPU version and CPU version at $N = 225$.

tiplication in order to improve the performance of the NSPT algorithm for the TEK model in the large N region. The FFT-CONV performs quite well even on the CPU system, we can further accelerate it by employing GPU cards. The TEK model is a reduced model and has no lattice sites to be parallelized, GPU systems is quite preferable to the TEK model as we observed in this benchmark report.

References

[1] A. González-Arroyo, I. Kanamori, K. I. Ishikawa, K. Miyahana, M. Okawa and R. Ueno, *Numerical stochastic perturbation theory applied to the twisted Eguchi-Kawai model*, JHEP **1906** (2019) 127 doi:10.1007/JHEP06(2019)127 [arXiv:1902.09847 [hep-lat]].

[2] F. Di Renzo, G. Marchesini, P. Marenzoni and E. Onofri, *Lattice perturbation theory by Langevin dynamics*, hep-lat/9308006.

[3] F. Di Renzo, G. Marchesini, P. Marenzoni and E. Onofri, *Lattice perturbation theory on the computer*, *Nucl. Phys. Proc. Suppl.* **34** (1994) 795.

[4] F. Di Renzo, E. Onofri, G. Marchesini and P. Marenzoni, *Four loop result in SU(3) lattice gauge theory by a stochastic method: Lattice correction to the condensate*, *Nucl. Phys.* **B426** (1994) 675 [hep-lat/9405019].

[5] F. Di Renzo, E. Onofri and G. Marchesini, *Renormalons from eight loop expansion of the gluon condensate in lattice gauge theory*, *Nucl. Phys.* **B457** (1995) 202 [hep-th/9502095].

[6] G. Burgio, F. Di Renzo, G. Marchesini and E. Onofri, Λ^2 *contribution to the condensate in lattice gauge theory*, *Phys. Lett.* **B422** (1998) 219 [hep-ph/9706209].

[7] F. Di Renzo and L. Scorzato, *Numerical stochastic perturbation theory for full QCD*, *JHEP* **10** (2004) 073 [hep-lat/0410010].

[8] M. García Pérez, Plenary talk at Lattice2019, in these proceedings.

[9] A. González-Arroyo and M. Okawa, *A Twisted Model for Large N Lattice Gauge Theory*, *Phys. Lett.* **120B** (1983) 174.

[10] A. González-Arroyo and M. Okawa, *The Twisted Eguchi-Kawai Model: A Reduced Model for Large N Lattice Gauge Theory*, *Phys. Rev.* **D27** (1983) 2397.