LatticeQCD.jl : Lattice QCD code with Julia


Akio Tomiya (International professional university of technology in Osaka, Assistant Professor)
akio@yukawa.kyoto-u.ac.jp

Yuki Nagai (Japan Atomic Energy Agency, Senior Scientist)

Introduction

(1)

- Lattice QCD = Multi-dimensional integral over SU(3)

$$S[U,\psi,\bar{\psi}] = \sum_n \left[-\frac{1}{g^2}\mathrm{Re\ tr}\ U_{\mu\nu} + \bar{\psi}(\not{D}+m)\psi \right]$$

$$\langle \mathcal{O} \rangle = \frac{1}{Z}\int \mathcal{D}U\mathcal{D}\bar{\psi}\mathcal{D}\psi e^{-S[U,\psi,\bar{\psi}]}\mathcal{O}[U,\psi,\bar{\psi}]$$

$$\mathcal{D}U = \prod_{n\in\{\mathbb{Z}/L\}^4}^{10^4}\prod_{\mu=1}^{4} dU_\mu(n) \qquad >1000 \text{ dimension.}$$

- This integral gives non-perturbative information of QCD
- Monte-Carlo is used to calculate (Numerical error independent to the dimensionality!)
- C++/Fortran have been used for simulations since it costs a lot! Supercomputers are needed for large scale calculations
- We make an open source code for lattice QCD with **Julia language**!

Julia?

(2)

- Programing language for science (Ref. 1) since 2012. Free, open
- Fast as C/Fortran (Fig1), productive as Python (Fig2)

Fig1.

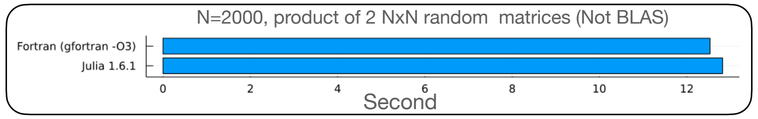


Fig2.

```
for i in [1,2,3]
  println("Hello world ", i)
end
```

- 33.9k star on Github. NASA uses Julia [2]. Runnable on Supercomputers
- Easy start: Binary available for Win, macOS and Linux, run everywhere
- Good package control system unlike python environment.
- Just-in-compiling, dynamic type. We can use Python/Fortran/C libraries. **Machine learning friendly!**

Why we make?

(3)

- To examine capability of Julia
- Ease of install/compiling
- Machine learning friendly lattice QCD code is needed
- Educational purpose/ Ease of modification

USAGE

Only 4 steps! See our Github webpage in details

1. Download Julia binary from the official webpage
2. Add lattice QCD using built-in package control system
3. (optional) Make parameter file with the wizard (type `run_wizard()`)
4. Execute! (type `run_LQCD("my_parameters.jl")`)

We also provide Google Colab notebook [4]!

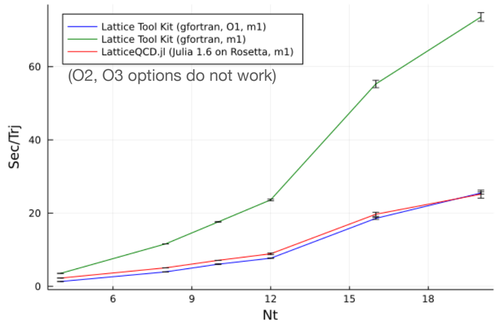
Features

(4)

- General gauge action (plaq+rect+chair +…) for SU(N) is supported
- Dynamical clover-Wilson (Nf=2), staggered fermions (Nf = 2-8). Both can be run with/without stout.
- (R)HMC, Heatbath (for quenched), self-learning Monte-Carlo, etc are supported
- Measurements: Plaquette, Polyakov loop, Chiral condensate, Pion correlator, topological charge
- Gradient flow with general gauge action
- ILDG I/O support
- Work on Google colab/ batch job / REPL (Julia prompt)
- (parallelization is in progress)
- Parameter wizard

Benchmark

(5)



Machine:
m1 mac mini
- Julia 1.6.1 + Rosetta2
- gfortran11 (with/wo O1)

L=4^3 x Lt
Lt = 4, 8, 10, 12, 16, 20
kappa = 0.141139
beta = 5.5
Nmd = 10, ε=0.1
CG eps = 10-8
(Default of Lattice tool kit [3])

- We compare with Lattice Tool kit (Fortran) , (same algorithm)
- Ls=4, Lt=4-20, beta = 5.5, kappa = 0.141139, full HMC
- Performance is good so far on single thread/core

Summary

(6)

- Julia is fast as Fortran/C, productive as Python (easy to write)
- LatticeQCD.jl works well, fast as a fortran code
- Future work: Overlap, domain-wall, parallelization
- (Modified version of) this code used for arXiv 2010.11900 and arXiv 2103.11965. Talk in session 27th, 13:00-, Algorithms

Reference

1. Julia: https://julialang.org
2. LTK: https://nio-mon.riise.hiroshima-u.ac.jp/LTK/
3. NASA: https://modelingguru.nasa.gov/docs/DOC-2783
4. Google Colab https://bit.ly/3yytQjG