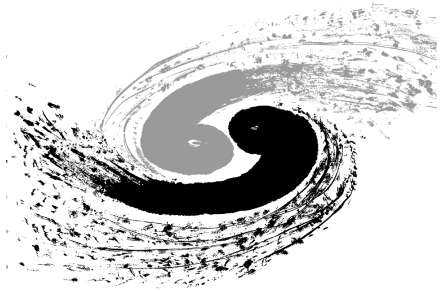
Lattice QCD calculation with Python

第三届中国格点量子色动力学研讨会



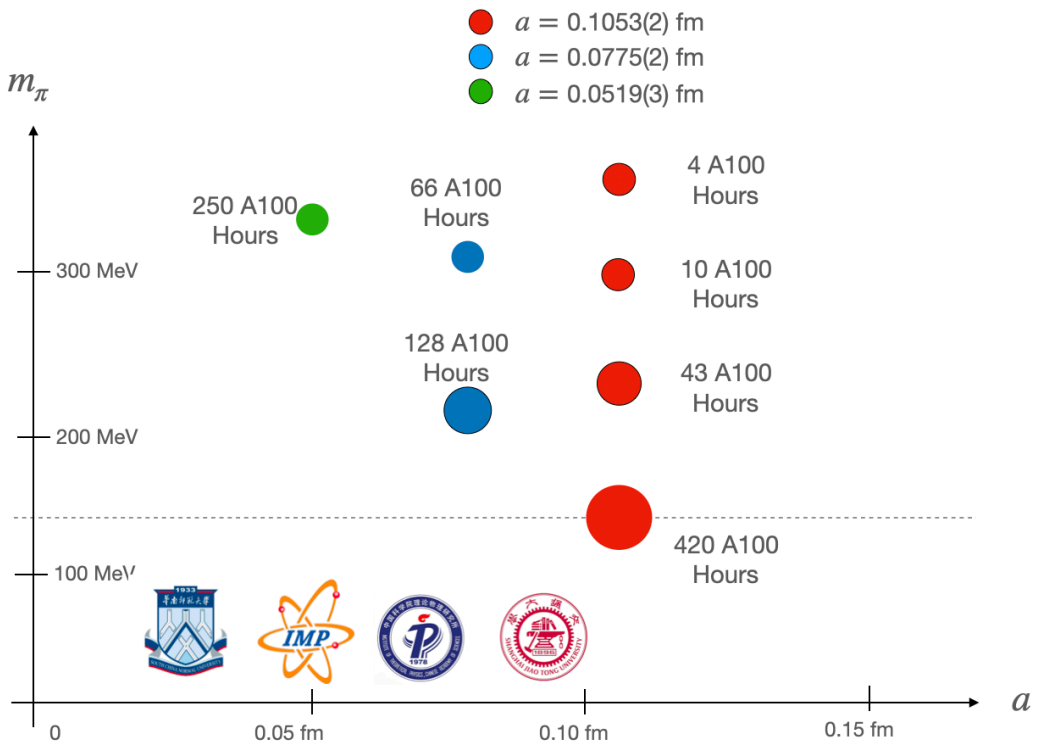
蒋翔宇，施春江
中国科学院理论物理研究所
中国科学院高能物理研究所
2023年10月7日

Table of contents

Introduction

Lattice QCD calculation

- Lattice QCD calculation are mainly about gauge configuration and quark propagator.
  - Numerically solving $\mathscr{M}x = b$ and $\mathscr{M}^\dagger \mathscr{M}x = b$ take more than 95% time in lattice QCD calculation.
- Small lattice spacing and light quark mass make the solver more difficult to converge.



Introduction

Heterogeneous computing

- There are more and more heterogeneous super computers.
- QUDA provides good performance on NVIDIA's and AMD's device.
- We developed QSUNWAY on Sunway architecture.
- We are developing a software on Sugon's device with IMP, CAS.

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,824,768	238.70	304.47	7,404
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096
6	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94.64	125.71	7,438
7	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93.01	125.44	15,371
8	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/LBNL/NERSC United States	761,856	70.87	93.75	2,589
9	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63.46	79.22	2,646
10	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61.44	100.68	18,482

Existing solution

Chroma + QDP-JIT + QUDA

- Pros
  - Legacy code
  - Correctness
  - Various features
  - External solvers for different architecture
    - QPhiX, QUDA, MG_proto…
- Cons
  - QDPXX is not suitable for GPU
  - Overhead on small cases
  - XML input files
    - Lack of documentation
    - Lots of repeat parts
  - Not easy to write extensions

Existing solution

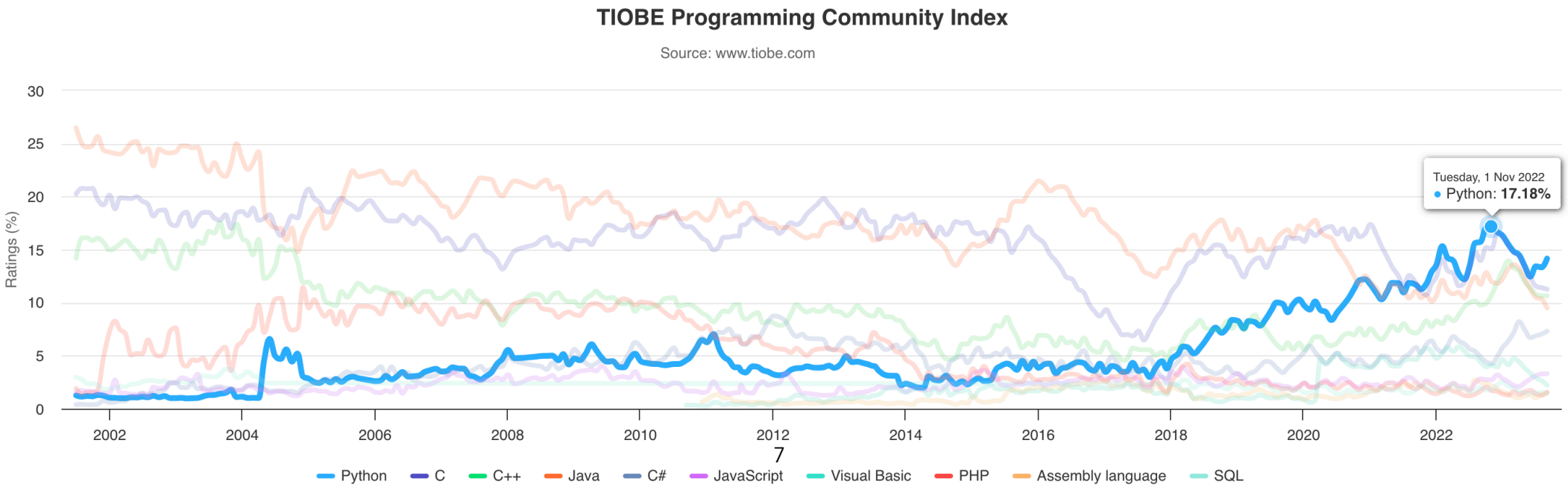
C++ + QUDA

- Pros
  - Active development with support from NVIDIA
  - High performance
  - Multiple backends including CUDA, ROCm and OneAPI
- Cons
  - Lack of feature
  - Buggy
  - Enormous parameters
    - Lack of documentation
  - Even more difficult to write extensions

LQCD with Python

Why Python?

- Python is one of the most popular programming language in the world.
- There are bunch of high quality numerical libraries in Python, which are easy to learn and use.
- The gate of AI.



LQCD with Python

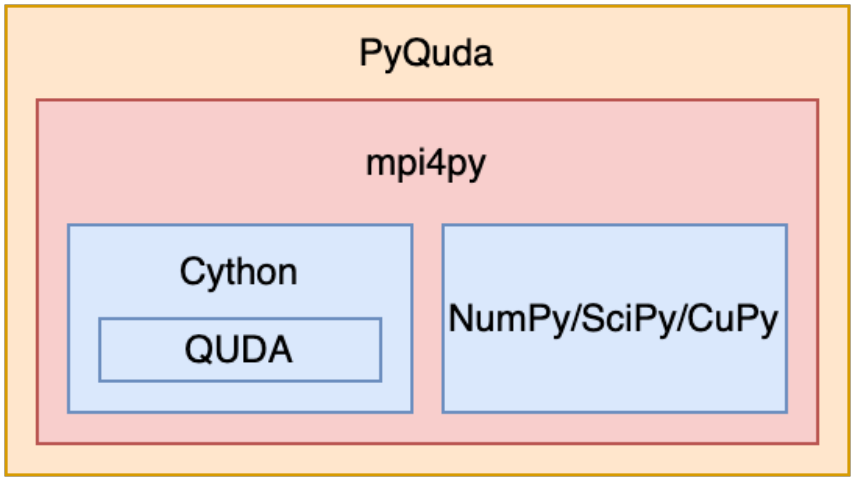
What we use now

- High performance numerical library: NumPy and SciPy.
  - Linear algebra, eigen solver, FFT, interpolation, statistics, …
- NumPy/SciPy on GPU: CuPy and PyTorch.
  - Example: einsum on GPU for distillation method.
    - Now we are blocked by I/O on HDD…
- Data analysis and visualization library: gvar, lsqfit and Matplotlib.
  - Used by IHEP and ITP.
- More: Pandas, Plotly, PyTorch, …

QUDA with Python

Framework

- Introducing PyQuda: https://github.com/CLQCD/PyQuda
  - Use Cython to wrap QUDA C interface.
    - C++ wrapper will be much more complicate
  - Use NumPy/SciPy/CuPy/PyTorch to implement aux functions.
    - Anti periodic t, anisotropy, …
  - Use mpi4py to use multiple GPUs



QUDA with Python

Quark propagator

	Isotropic	Anisotropic
Wilson	○	○
Clover	○	○
Twisted mass	?	?
Twisted clover	?	?
Staggered	?	✕
ASQTAD/HISQ	○	✕
Domain wall	✕	✕
Mobius	✕	✕

QUDA with Python

Quark propagator

```
import os

from pyquda import core, mpi, field
from pyquda.field import Nc, Ns
from pyquda.utils import gauge_utils

field.CUDA_BACKEND = "torch"

os.environ["QUDA_RESOURCE_PATH"] = ".cache"
mpi.init([1, 1, 1, 2])

latt_size = [4, 4, 4, 8]
Lx, Ly, Lz, Lt = latt_size
Vol = Lx * Ly * Lz * Lt

xi_0, nu = 2.464, 0.95
kappa = 0.115
coeff = 1.17
coeff_r, coeff_t = 0.91, 1.07

mass = 1 / (2 * kappa) - 4
```

```
dslash = core.getDslash(
    latt_size,
    mass,
    1e-9,
    1000,
    xi_0,
    nu,
    coeff_t,
    coeff_r,
    multigrid=[[4, 4, 4, 4], [4, 4, 4, 4]],
)
gauge =
gauge_utils.readQIO("weak_field.lime")
dslash.loadGauge(gauge)

propagator = core.invert(
    dslash,
    “gaussian",
    [0, 0, 0, 0],
    None,
    2.0,
    5,
)
pion = torch.einsum("etzyxijab,etzyxijab->t",
propagator.data.conj(), propagator.data)

dslash.destroy()
```

QUDA with Python

Gauge configuration

	Isotropic	Anisotropic
Pure gauge	○	✕
Clover	2-flavor without stout smearing	✕
Twisted clover	?	✕
Staggered	?	✕
HISQ	?	✕

QUDA with Python

Gauge configuration

```
import os
import numpy as np
import cupy as cp

import pyquda
from pyquda import core, field
from pyquda.hmc import HMC
from pyquda.field import Nc, Ns

os.environ["QUDA_RESOURCE_PATH"] =
".cache"
pyquda.init()

latt_size = [16, 16, 16, 16]
Lx, Ly, Lz, Lt = latt_size
Vol = Lx * Ly * Lz * Lt

mass = 4
kappa = 1 / (2 * (mass + 4))
csw = 1.0
hmc = HMC(latt_size, mass, 1e-9,
1000, csw, True)

gauge =
field.LatticeGauge(latt_size, None)
hmc.loadGauge(gauge)
hmc.loadMom(gauge)
t = 1.0
dt = 0.1
steps = round(t / dt)
dt = t / steps
warm = 20
for i in range(100):
    hmc.gaussMom(i)

    cp.random.seed(i)
    phi = 2 * cp.pi *
cp.random.random((2, Lt, Lz, Ly,
Lx // 2, Ns, Nc))
    r = cp.random.random((2, Lt, Lz,
Ly, Lx // 2, Ns, Nc))
    noise =
core.LatticeFermion(latt_size,
cp.sqrt(-cp.log(r)) * (cp.cos(phi) +
1j * cp.sin(phi)))

    hmc.initNoise(noise, i)

    kinetic = hmc.actionMom()
    potential = hmc.actionGauge(path,
lengths, coeffs, num_paths,
max_length)
    potential +=
hmc.actionFermion(noise)
    energy = kinetic + potential
    for step in range(steps):
        hmc.updateGaugeField(0.5 * dt)
        hmc.computeGaugeForce(1.0 *
dt, force, flengths, fcoeffs,
num_fpaths, max_length - 1)
        hmc.computeCloverForce(1.0 *
dt, noise, -(kappa**2), -kappa * csw /
8)
        hmc.updateGaugeField(0.5 * dt)

    hmc.reunitGaugeField(1e-15)

    kinetic1 = hmc.actionMom()
    potential1 = hmc.actionGauge(path,
lengths, coeffs, num_paths,
max_length)
    potential1 +=
hmc.actionFermion(noise)
    energy1 = kinetic1 + potential1

    accept = np.random.rand() <
np.exp(energy - energy1)
    if warm > 0:
        warm -= 1
    if accept or warm:
        hmc.saveGauge(gauge)
    else:
        hmc.loadGauge(gauge)
```

QUDA with Python

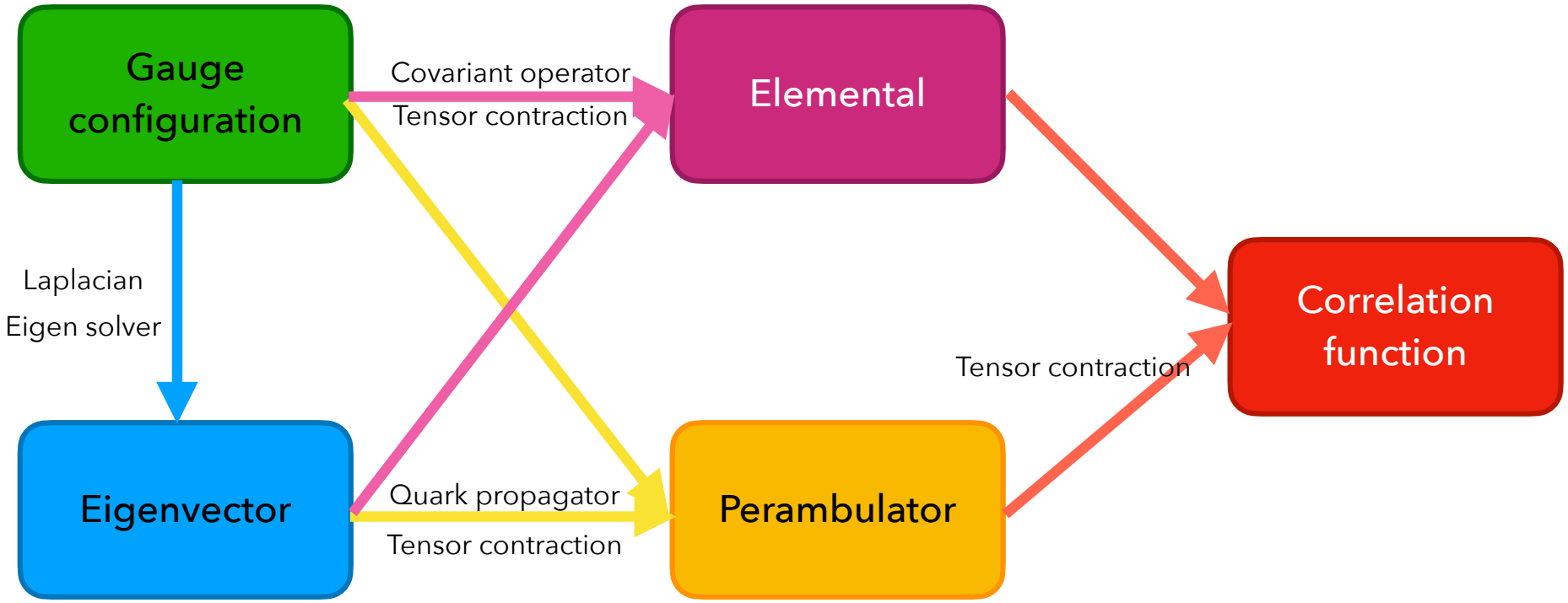
Others

	Isotropic	Anisotropic
Gauge fixing	⭕	❌
Gauge smearing	⭕	❌
Distance precondition (Heavy quark)	⭕	⭕
QIO/MILC gauge input	⭕	❌

Distillation with Python

Framework

- Introducing EasyDistillation: https://github.com/IHEP-LQCD/EasyDistillation. (Pre alpha)



Distillation with Python

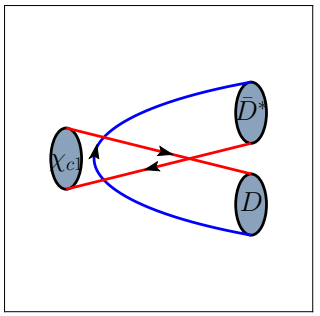
Framework

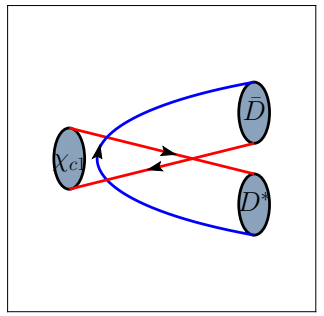
- Use einsum to perform most tensor contraction.
- Use NumPy/SciPy/CuPy functions to perform Laplacian, eigen solver and covariant operator.
- Use PyQuda to calculate quark propagator.
- Use SymPy to construct meson correlation functions.
  - Mesons can be constructed by linear combination of derivatives and gamma matrices.
- Use feynman to draw quark schematic diagram.

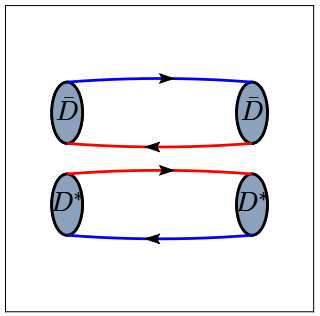
Distillation with Python

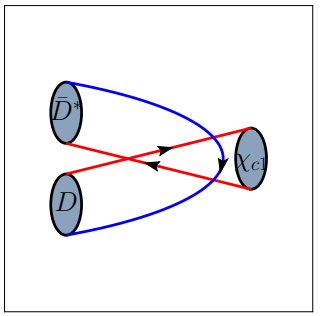
Example

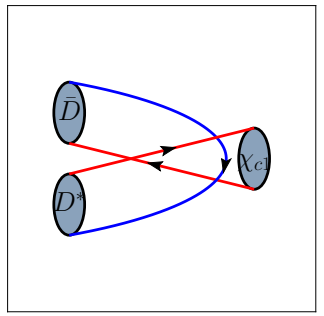
- Using adjacency matrix to represent a quark contraction path.
  C. Shi, et al., in preparation.

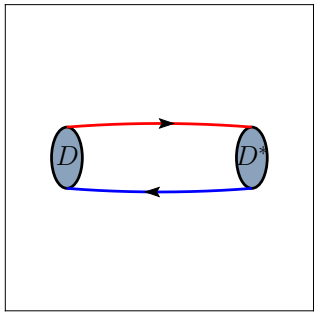


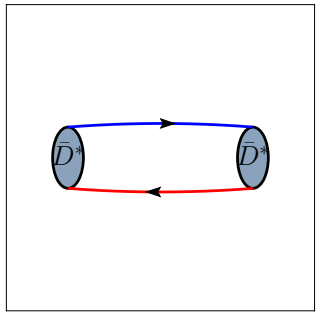


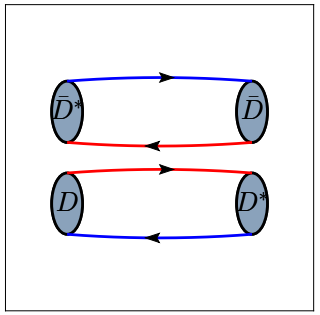


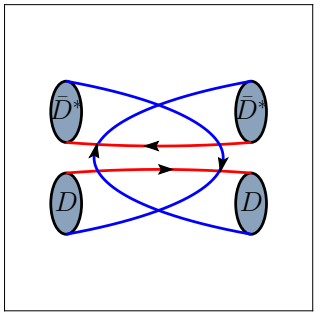


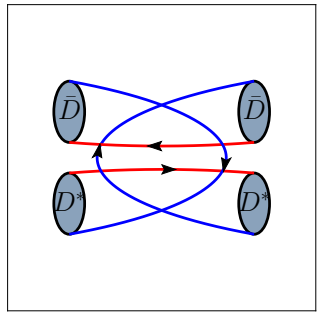


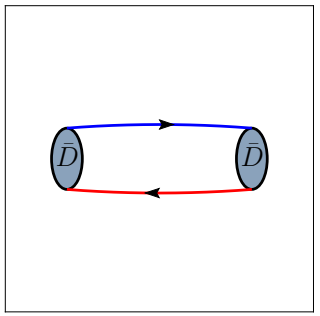


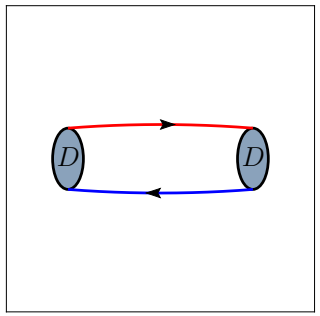


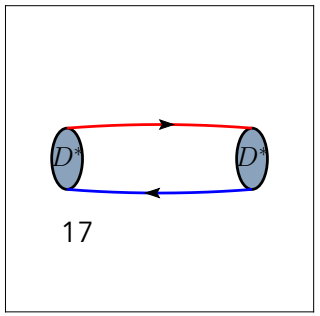


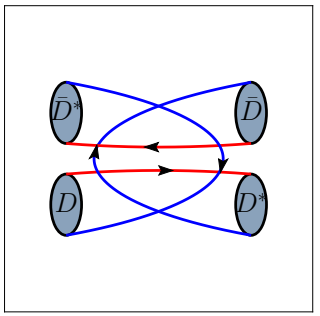


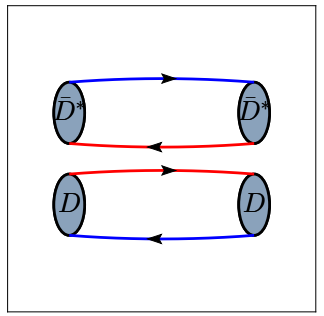












Distillation with Python

Example

```python
from sympy import S, sqrt

from lattice.quark_contract import Meson, Tag, quark_contract
from lattice.quark_draw import draw_multi_diagrams

eta_source = (
    S(1) / sqrt(2) * (Meson("u", R"$\gamma_5$", "u", Tag(0, 0), True) + Meson("d", R"$\gamma_5$", "d", Tag(0, 0), True))
)
eta_sink = (
    S(1) / sqrt(2) * (Meson("u", R"$\gamma_5$", "u", Tag(1, 1), False) + Meson("d", R"$\gamma_5$", "d", Tag(1, 1), False))
)

mat, coeff, rep, line = quark_contract(eta_sink * eta_source, 2)
print(mat)
print(coeff)
print(rep)
print(line)
```

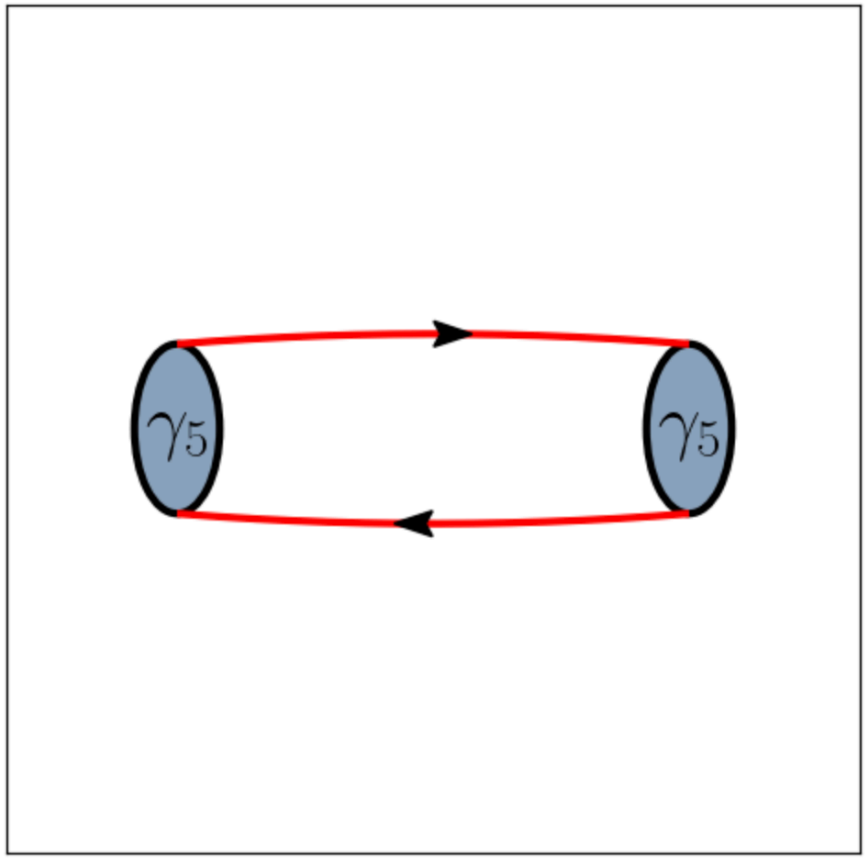
✓ 0.0s Python

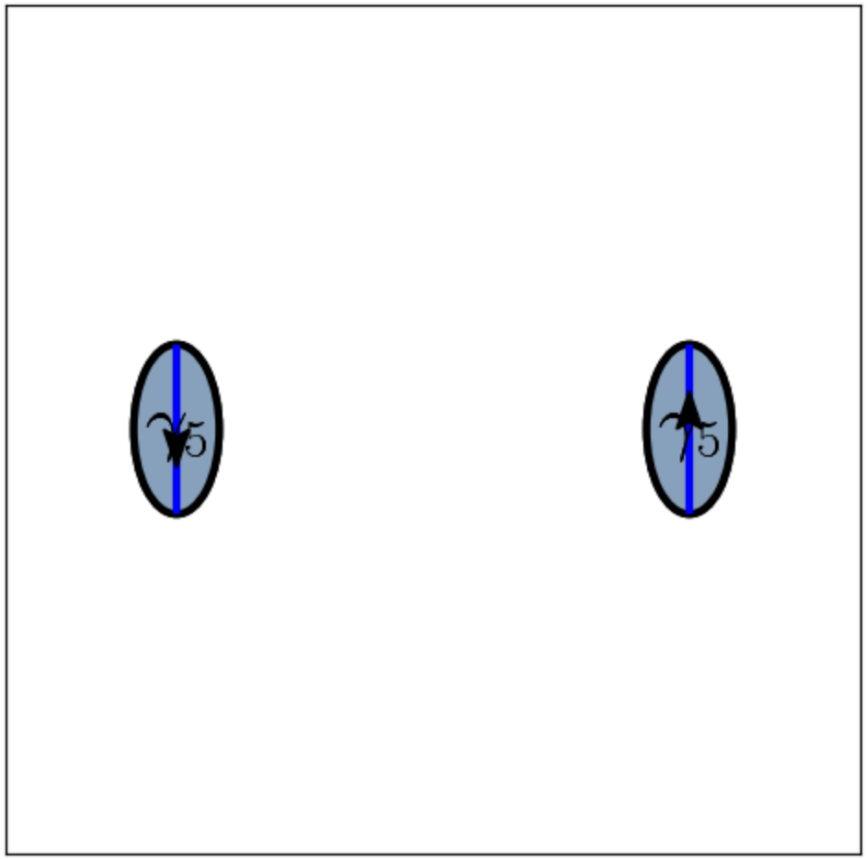
```
[[[0, 1], [1, 0]], [[2, 0], [0, 2]]]
[-1, 2]
['$\\gamma_5$^\\dagger', '$\\gamma_5$']
[None, 'S^q', 'S^q_\\mathrm{local}']
```

Distillation with Python

```python
rep = [
    dict(
        pos="src" if gamma_ins.endswith(R"^\dagger") else "snk",
        type="meson",
        name=gamma_ins[:-8] if gamma_ins.endswith(R"^\dagger") else gamma_ins,
    )
    for gamma_ins in rep
]
draw_multi_diagrams(mat, rep, [None, "r", "b"])
```

✓ 0.0s Python





Summary and outlook

- We can solve Clover and HISQ propagators in Python.
- We can auto construct and compute meson correlation functions using distillation method in Python.
- Preliminary Clover HMC support.
- We need more features in QUDA.
  - RHMC, STOUT smeared force, Hasenbusch term, …
- We need more features in EasyDistillation.
  - Baryon correlation function.
- We are planning to support Sugon’s DCU with PyQuda.

Thank you for your attention!