LeonardYM

Stefano Piemonte

April 20, 2020

1 Overview

LeonardYM is a software developed in C++ for Monte-Carlo simulations of four-dimensional Yang-Mills theories, ranging from QCD to supersymmetric models. The code is designed to allow many different theories to be simulated within the same framework, rather than being focused and optimized for a single specific target theory. Parameters such as the number of colors of the gauge group SU(N) or the representation of fermion fields can be freely chosen at compile time, allowing a large flexibility on the structure of the theories that can be simulated.

The code has been developed mainly during the work for the PhD thesis of Ref. [1]. The present documentation contains a brief overview of the main features and algorithms implemented in the code. Further details about the algorithms and the settings required for using the code can be found in Ref. [1, 2, 3].

2 Organization of the code

The code is organized around the class "LatticeSweep", which is meant to execute and implement by inheritance a specific measurement or a given update of the field configurations. Each sweep acts on an instance of the Environment class, which contains

1. the gauge-link configuration in the fundamental representation,
2. the gauge-link configuration used in the definition of the fermion covariant derivatives,
3. the gauge-link configuration in the adjoint representation,
4. the scalar field configurations in the adjoint and fundamental representation,
5. an instance of the class StorageParameters, encapsulating parameters such as the fermion mass and the gauge coupling.

Each LatticeSweep can register its own parameters to be read from the configuration file, the standard naming convention is "LatticeSweepName::option_name".

3 Field variables

The code has implemented scalar, fermion and gauge fields in the fundamental and adjoint representation of the gauge group SU(N). All fields are defined from templates of the class

Lattice, which supports access to the field in given site and its lower and upper neighbor by a linear index and by the functions sup(site, mu) and sdn(site, mu). All field variables are based on the matrix library Eigen [15].

The gauge field $U_\mu(x)$ are represented in terms of links, which are $N \times N$ or $(N^2-1)\times(N^2-1)$ matrices in the fundamental and adjoint representation, respectively. The scalar fields are complex vector lattice fields with N or (N^2-1) color indexes, while fermion fields have in addition a spin components.

4 Lattice actions

There are implementations for two possible lattice discretizations of the gauge action, namely the Wilson and the Symanzik action. The Wilson action is defined from the elementary plaquette, the trace of the product of the gauge links around a 1×1 square, while the Symanzik action includes also 2×1 rectangles.

The discretization of the fermion action is more involved, as the Nielsen-Ninomiya theorem forbids a local hermitian action preserving chiral symmetry and free of doublers on the lattice [4]. The Dirac operator is implemented in the code using Wilson fermions, clover-improved Wilson fermions and overlap fermions. Wilson fermions breaks chiral symmetry explicitly and the fermion mass requires additive renormalization and a tuning to the chiral limit, the clover-improved action depends on the clover coefficient c_{sw} tuned to reduce the leading $O(a)$ discretization errors of the Wilson formulation [13]. Overlap fermions allow instead to preserve a modified chiral symmetry on the lattice [5]. For all fermions operators, it is possible to smear the link fields to reduce the ultraviolet fluctuations and to stabilize the integration of the HMC trajectory.

The scalar action is discretized using simple forward derivative, defined in terms of the link fields, and it depends on the couplings of the quartic potentials.

5 Monte-Carlo algorithms implemented in the code

The main algorithms to generate gauge-field configurations implemented in the code differ depending on whether fermions are interacting with gauge bosons or not. In the first case, the most efficient Monte-Carlo algorithms are based on local updates of the link field, while global updates and Hybrid Monte-Carlo (HMC) are the best algorithms for simulations of gauge theories with dynamical fermions [6, 7].

5.1 Pure-gauge updaters

The Cabibbo-Marinari algorithm together with local over-relaxation is the most efficient solution for Monte-Carlo simulations of a pure-gauge theory without fermions [8, 9]. There is also the possibility of performing improved measurements of Polyakov and Wilson loops by averaging their expectation value with respect to (part) of their links (Lüscher-Creutz multilevel algorithm [10]). In this case, the only certain links are updated, while others are kept fixed.

5.2 Hybrid Monte Carlo

In relativistic quantum field theories, time is a dimension and not simply a parameter. In order to translate the action of a four-dimensional field theory to a standard Hamiltonian system, an additional time dependence τ is added to the usual four space-time dimensions, for example a link variable depends on five variables, $U_\mu(\vec{x}, t, \tau) \equiv U_\mu(x, \tau)$. The fictitious time τ parametrizes the trajectory of the system in the phase space. The momenta are independent variables from the fields in the Hamiltonian formalism of classical mechanics, therefore the partition function can be multiplied by the quadratic Gaussian integral

$$\int \prod_{x,\mu} d\pi_\mu(x,\tau) \exp(-\mathrm{Tr}(\pi_\mu(x,\tau)^\dagger \pi_\mu(x,\tau))$$

and the free field $\pi_\mu(x,\tau)$ is interpreted as the canonical variable conjugate to the link $U_\mu(x,t,\tau)$. The equation of motion for $U(x,\tau)$ simply reads

$$\frac{d}{d\tau} U_\mu(x,\tau) = \pi_\mu(x,\tau) U_\mu(x,\tau), \tag{1}$$

preserving carefully the order of the matrix multiplications. The equation of motion of the momenta $\pi_\mu(x,\tau)$ are

$$\frac{d}{d\tau} \pi_\mu(x,\tau) = -\frac{\partial}{\partial U_\mu(x,\tau)} \left(S(U_\mu)(U_\mu)\right) = -F\,. \tag{2}$$

being S the action of the four-dimensional theory.

The equation of motion (1) and (2) can be integrated numerically with a further discretization of the fictitious time τ in steps large ϵ. An update $\mathcal{U}_\pi(\epsilon)$ of the momenta is simply

$$\pi_\mu(x,\tau) \to \mathcal{U}_\pi(\epsilon)\pi_\mu(x,\tau) = \pi_\mu(x,\tau+\epsilon) = \pi_\mu(x,\tau) - \epsilon F,$$

keeping unchanged the links $U_\mu(x,\tau)$, while an update of the links $\mathcal{U}_U(\epsilon)$ is

$$U_\mu(x,\tau) \to \mathcal{U}_U(\epsilon) U_\mu(x,\tau) = U_\mu(x,\tau+\epsilon) = \exp\left(\epsilon \pi_\mu(x,\tau)\right) U_\mu(x,\tau),$$

keeping the momenta fixed. The symbol "exp" denotes in this context the exponential map.

The simplest reversible scheme to numerically integrate the equations of motion can be constructed by the following chain of link and momenta updates

$$\mathcal{U}_\pi(\epsilon/2)\mathcal{U}_U(\epsilon)\mathcal{U}_\pi(\epsilon/2),$$

the specified final time $\tau_f = N\epsilon$ will be reached after N repetitions of this scheme. This integrator is called Leap-Frog and it converges toward the exact solution as $O(\epsilon^4)$. Many other numerical integrators are possible with a smaller discretization error [11, 12]. The code has implementation for integrators up to the sixth order and for Omelyan integrators up to the fourth order.

Once that the equations of motion can be integrated numerically, the Hybrid Monte Carlo algorithm is implemented in three simple steps. A link configuration is generated for example

randomly and the following steps are iterated starting from $k = 0$

1. Generate randomly with a Gaussian distribution the momenta $\pi_\mu(x, 0)$. Set the link to their initial configuration $U_\mu(x, 0) \equiv U_\mu(x)$. Measure the energy $E_i = H(\pi_\mu(x, 0), U_\mu(x, 0))$.

2. Integrate numerically the equations of motion for a time τ in N steps of ϵ length ($\tau = N\epsilon$), to produce a new configuration of links $U_\mu(x, \tau)$ and momenta $\pi_\mu(x, \tau)$. Compute the energy $E_f = H(\pi_\mu(x, \tau), U_\mu(x, \tau))$.

3. Accept or reject the new configuration $U_\mu(x, \tau)$ with probability

$$\min(1, \exp(E_i \ - \ E_f)) \, .$$

   Set the configuration $C_k = U_\mu(x, 0)$ or $C_k = U_\mu(x, \tau)$ accordingly and proceed to the next iteration $k = k + 1$.

6 Available measurement and updater sweeps

6.1 Pure gauge updaters

The following updaters are designed to simulate an SU(N) gauge theory without matter interactions.

- The **PureGaugeCM** sweep performs an update of the gauge links using heat-bath Monte-Carlo local updates following the Cabibbo-Marinari algorithm [8].

- The **PureGaugeWilsonLoops** sweep performs a measurements of the Wilson loops by averaging the part of the link variables, following the Lüscher-Weisz multilevel algorithm [10].

- The **PureGaugeHMCUpdater** sweep updates all gauge links using the Hybrid Monte Carlo algorithm.

- The **PureGaugeOverrelaxation** sweep performs an over-relaxation update of all gauge links [9].

6.2 Fermion updaters

The following updaters are designed to simulate an SU(N) gauge theory interacting with fermion matter fields.

- The **TwoFlavorHMCUpdater** is the Hybrid Monte Carlo algorithm implementation of a theory interacting with two Dirac fermion fields.

- The **MultiStepNFlavor** is the main sweep to update the gauge links of a QCD-like theory interacting with an arbitrary number of fermions in any representation, using the Hybrid Monte Carlo algorithm.

6.3 Scalar field updaters

The following updaters are designed to simulate an SU(N) gauge theory interacting with scalar and/or fermion matter fields.

- The **HiggsGaugeHMC** sweep performs an update of the gauge links for a theory interaction with scalars.
- The **ScalarFermionHMC** sweep performs an update of the gauge links for a theory interaction with scalar and fermion fields.
- The **AdjointMCScalar** sweep performs a Metropolis update of all scalar fields in the adjoint representation of the gauge group.
- The **FundamentalMCScalar** sweep performs a Metropolis update of all scalar fields in the fundamental representation of the gauge group.

6.4 Gauge action

- The **Plaquette** sweep performs a measurement of the plaquette, i.e. the trace of a Wilson loop of size 1×1. It can be used to monitor thermalization or unexpected deviations from equilibrium, as well as the latent heat of a first order phase transition.
- The **GaugeEnergy** measures the total gauge energy of a given configuration.

6.5 Wilson and Polyakov loops

- The **WilsonLoops** sweep is used to measure the rectangular $R \times T$ Wilson loops.
- The **PolyakovLoop** sweep measures the Polyakov loop in all directions, i.e. a Wilson loop wrapping around the four compact dimensions. It is related to the free energy of a single isolated static charge and its non-zero expectation value is a signal for deconfinement.
- The **AdjointPolyakovLoop** sweep measures the Polyakov loop constructed from links in the adjoint representation.
- The **PolyakovLoopEigenvalues** sweep measures the distribution of the eigenvalues of Polyakov line.

6.6 WilsonFlow

- The **WilsonFlow** sweep integrates numerically the Wilson flow and measures the energy and the topological charge during the integration.

6.7 Correlators

- The **Glueball** sweep performs a measurement of the zero momentum projection of the 0^{++} and 2^{++} glueball operator on each timeslice, required for the computation of the connected correlators.

- The **PolyakovLoopCorrelator** is a measurement of the correlator of two Polyakov loops, that can be used to extract the static quark string tension.

- The **GluinoGlue** sweep performs a measurement of the gluino-glue correlator, corresponding to an exotic spin-1/2 particle which can exist if the fermions are interacting with the gluons in the adjoint representation of the gauge group. The gluino-glue is measured from a volume source placed at a single timeslice, multiple timeslices in a single configuration can be measured in sequence.

- The **MesonCorrelator** sweep measures the connected correlator of the scalar and pseudoscalar connected correlators, required to extract the pion mass and the bare PCAC fermion mass.

The stout-smearing ρ and the number of stout-smearing iterations of the gauge links, together with the Jacobi smearing counterparts for the gluino-glue, are tunable parameters to improve the overlap with the ground state.

6.8 Gauge fixing and gauge fixed measurements

- The **LandauGaugeFixing** sweep is used to impose the Landau gauge on a given link configuration. The minimization starts from parallel tempering and a series of random gauge transformations to try to search for the global minimum of the gauge-fixing functional, followed by a series of local over-relaxation sweeps to bring the configuration toward the local minimum.

- The **LandauGhostPropagator** sweep measures the ghost propagator in the Landau gauge, using conjugate gradient for computing the inverse of the ghost operator. The gauge-field configuration is assumed to be already gauge-fixed.

- The **LandauGluonPropagator** sweep performs a measurement of the gluon propagator in the Landau gauge. The gauge-field configuration is assumed to be already gauge-fixed.

- The **NPRVertex** sweep measures the fermion propagator and the connected vertex function required for the non-perturbative renormalization of fermion bilinear operators.

6.9 Fermion measurements

- The **ChiralCondensate** sweep measures the chiral condensate, and the chiral susceptibility, including connected and disconnected parts.

- The **Eigenvalues** sweep performs a measurement of the eigenvalues of the Dirac operator using the implicitly restarted Arnoldi algorithm.

6.10 Miscellaneous utilities

- The **ReUnit** sweep ensures the correct unitary of the gauge links, which could be spoiled by accumulated numerical truncation errors during a Monte Carlo simulation.

- The **RandomGaugeTransformation** sweep performs a random gauge transformation of the gauge link fields. Given that many lattice observables are gauge invariant, the sweep can be used to test the correct implementation of the code for a measurement, by just running the sweep in between two consecutive identical measurements.
- The **Output** sweeps is used to write and save the gauge-link field configuration.
- The **TestLinearAlgebra** can be employed to check the correct implementation of the Dirac operator and to spot possible compiler issues.
- The **TestSpeedDiracOperators** sweep measures the performance of the multiplication of the Dirac operator to a vector.
- The **TestCommunication** sweep is used to ensure that the MPI communications are working properly.
- The **ReadGaugeConfiguration** sweep is used to load a previously saved link-gauge configuration.

7 Compiling the code

The code requires two libraries, namely boost [14] and Eigen [15]. In addition, MPI libraries and compilers are required if MPI parallelism is enabled. The number of colors of the gauge group is specified by the compiler flag NUMCOLORS. The representation of the gauge group SU(N) under which the fermion fields are transforming is by default the fundamental, the fermions can be switched to the adjoint representation by the compiler flag ADJOINT. MPI and OpenMP parallelism are enabled by the compiler flags ENABLE_MPI and MULTITHREADING, respectively.

A few Makefile examples are included in the main distribution. For instance, execute

```
make -f Makefile.mth nc=3 adjoint=true
```

to compile for Monte Carlo simulations with the gauge group SU(3) interacting with adjoint fermions using only multithreading, or

```
make -f Makefile.mpi nc=3 adjoint=true
```

to include also MPI parallelism.

8 Running the code and example scripts

Once the code is compiled, it can be executed by the command line

```
leonardYM.exe --configfile=my_config_file.cfg
```

or by the corresponding MPI execution command. The configuration file contains the options and settings of the Monte Carlo simulation.

A minimal configuration file must contain first the definition of the lattice geometry

```
glob_x = 8
glob_y = 8
glob_z = 8
glob_t = 16
```

and, if MPI parallelism is enabled, the MPI grid

```
pgrid_x = 1
pgrid_y = 1
pgrid_z = 1
pgrid_t = 2
```

to specify to the four dimensional lattice is divided among the MPI tasks. In the example above, the lattice size is $8^3 \times 16$, divided in two MPI task along the time direction.

The main global parameter β is required to set the gauge coupling, in the example below, of the gauge action StandardWilson.

```
beta = 5.6


name_action = StandardWilson
```

The Monte Carlo simulation is divided in two parts, in the first the "warm-up sweeps" are executed, followed by the "measurement sweeps". The warm-up sweeps consist typically of some updates required for thermalization, while during the measurement phase new configurations are generated and some observable is measured.

```
start = hotstart


number_warm_up_sweeps = 300
warm_up_sweeps = "{{PureGaugeCM,1,1},{PureGaugeOverrelaxation,1,5}}"
number_measurement_sweeps = 2500
measurement_sweeps = "{{PureGaugeCM,1,1},{PureGaugeOverrelaxation,1,5},
        {Plaquette,1,1},{PolyakovLoop,5,1}}"
```

The start parameter specify how the gauge configuration is initialized at the beginning of the Monte Carlo simulations (hotstart corresponds to a random initialization of the link variables). In the example above, 2500 measurement iterations are executed after 300 warm-up iterations. In each iteration, the sweep PureGaugeCM is called once, while the PureGaugeOverrelaxation is called five times (third parameter in each curly bracket). The same happens during the measurement phase, but in addition the Plaquette sweep is called every iteration and the PolyakovLoop is measured every fifth iteration (second parameter in curly brackets).

Next, we need to specify a directory, a name and a format for the output of the measurements.

```
output_directory_measurements = /path/to/output/directory/
output_name = pure_gauge_8c16
measurement_output_format = xml
```

Finally, we need to specify the number of OpenMP threads.

```
number_threads = 4
```

Further configuration script examples can be found in the directory configuration_scripts.

9 Testing

The most important tests to check the correctness of the Hybrid Monte-Carlo algorithm can be enabled by the compiler flags REVERSIBILITY_CHECK and DEBUGFORCE. The first compiler flag enables the integration of the equations of motion backward in time, and by construction at the end the difference of the action must be zero within numerical errors. The second compiler flag enables the check of the correct implementation of the force as a derivative of the action with respect to the a variation of the links.

References

[1] S. Piemonte, $\mathcal{N} = 1$ *supersymmetric Yang-Mills theory on the lattice*, PhD thesis (2014).

[2] I. Montvay: *Majorana fermions on the lattice*, hep-lat/0108011.

[3] S. Ali, G. Bergner, H. Gerber, I. Montvay, G. Münster, S. Piemonte, P. Scior: *Numerical results for the lightest bound states in* $\mathcal{N} = 1$ *supersymmetric SU(3) Yang-Mills theory*, Phys. Rev. Lett. 122, 221601 (2019).

[4] H.B. Nielsen, M. Ninomiya, *A no-go theorem for regularizing chiral fermions*, Phys. Lett., B105: 219 (1981).

[5] P. H. Ginsparg and K. G. Wilson, *A remnant of chiral symmetry on the lattice*,Phys. Rev. D25 (1982) 2649.

[6] S. Gottlieb, W. Liu, D. Toussaint, R. L. Renken, R. L. Sugar: *Hybrid-molecular-dynamics algorithms for the numerical simulation of quantum chromodynamics*, Phys. Rev. D **35** (1987) 2531.

[7] S. Duane; A.D. Kennedy, B. J. Pendleton, R. Duncan: *Hybrid Monte Carlo* Phys. Lett. B bf 195 (1987) 216.

[8] N. Cabibbo, E. Marinari: *A new method for updating SU(N) matrices in computer simulations of gauge theories*, Phys. Lett. B119(1982) 387.

[9] M. Creutz: *Overrelaxation and Monte Carlo simulation*, Phys. Rev. D36(1987) 515.

[10] M. Lüscher, P. Weisz: *Locality and exponential error reduction in numerical lattice gauge theory*, JHEP 0109:010,2001.

[11] I.P. Omelyan, I.M. Mryglod and R. Folk: *Optimized verlet-like algorithms for molecular dynamics simulations*, Phys. Rev. E **65** (2002) 056706.

[12] T. Takaishi, P. de Forcrand: *Testing and tuning symplectic integrators for Hybrid Monte Carlo algorithm in lattice QCD*, Phys. Rev. E **73** (2006) 036706.

[13] K. Jansen, C. Liu: *Implementation of Symanzik's Improvement Program for Simulations of Dynamical Wilson Fermions in Lattice QCD*, Comput. Phys. Commun. **99** (1997) 221.

[14] www.boost.org.

[15] eigen.tuxfamily.org.