HMC, Chroma and ...

C. Urbach

Institut für Physik
HU-Berlin

Lattice Practices 2008

Outline

1. The Chroma Library: a short introduction
   Tutorial

Outline

1. The Chroma Library: a short introduction
   Tutorial
2. Writing Chroma XML input

Outline

1. The Chroma Library: a short introduction
   Tutorial
2. Writing Chroma XML input
3. The HMC algorithm
   using the harmonic oscillator as example

Outline

1. The Chroma Library: a short introduction
   Tutorial
2. Writing Chroma XML input
3. The HMC algorithm
   using the harmonic oscillator as example
4. The HMC for QCD
   Tutorial

Outline

1. The Chroma Library: a short introduction
   Tutorial
2. Writing Chroma XML input
3. The HMC algorithm
   using the harmonic oscillator as example
4. The HMC for QCD
   Tutorial
5. The HMC for QCD using Chroma
   Tutorial

The Chroma Library for Lattice Field Theory

- CHROMA is a C++ software for lattice QCD
- it provides
  - a library with a large variety of functions to be used in lattice field theory
  - a number of executables, most importantly: `chroma, hmc`
  - it is free software with many authors, but mainly Robert Edwards and Balint Joo
  - it is architecture independent

The Chroma Library for Lattice Field Theory

- it can be build with architecture specific improvements in particular
  - SSE for PC processors
  - DubbleHummer for Blue Gene systems
- it provides some standard applications
  - gauge configuration generation quenched (`purgaug`)
  - dynamical gauge configuration generation (`hmc`)
  - inverter (`chroma`)
- with many features, to be chosen at run time
  - system size, n_f, n_c
  - fermion actions (Wilson, Wilson clover, Wilson twisted mass, Domain Wall, ...)
  - gauge actions (Plaquette, tlSym, DBW2, ...)
  - HMC types (HMC, RHMC, ...)
  - measurements for many observables (long list)

Where to get it...?

- it can be obtained from
  `http://usqcd.jlab.org/usqcd-docs/chroma/`
- anonymous CVS access is possible
- everyone is welcome to contribute
- it ships with a GNU build environment,
  allowing (in principle) easy build on many architectures
- it depends on a number of other libraries
  that need to be compiled in addition,
  which are free software as well
- it also ships with a detailed test suite

Great! Any reason for not using it ...?

- well ...!?

Great! Any reason for not using it ...?

- well ...!?
- documentation is (in my opinion) problematic
  chroma provides a lot, but it is not so easy to find out what
- compiling is not as easy as one might hope
- the code is well written, but heavy C++
  adding own features is not easy if one is not a C++ expert
- compiling can take forever and the compiler must be a recent version of gcc
- the needed libraries sometimes do not compile
- the produced code can be rather heavy
  one might run into memory problems
- heavy usage of XML

However, ...

- we use it here, because it is freely available
- and because it is very general
- and because it is very good software

- I shall not cover in my lectures
  how to use chroma as a library
- I'll discuss the usage of the `chroma` executable in this first lecture
- and the `hmc` executable tomorrow

- I'll be far from explaining everything
- but try to get you going

Outline

1. The Chroma Library: a short introduction
   Tutorial
2. Writing Chroma XML input
3. The HMC algorithm
   using the harmonic oscillator as example
4. The HMC for QCD
   Tutorial
5. The HMC for QCD using Chroma
   Tutorial

Goal of this lecture

- at the end of this lecture and tutorial you should be able to use basic features of chroma
- and write XML input files for it
- in particular, you should be able to use chroma to solve

$$D[U]\psi = \eta$$

  for ψ given η and some gauge field U
- and use the result for measurements

- I'll not cover error anlysis and correlation functions
- I'll not explain algorithms or something like this in this lecture

Running the chroma executable

- everything what chroma is supposed to do
  is decided at run time
- the communication with the executable goes via XML
- once an input file, say `in.xml` is written, one runs with
  `path-to-chroma/chroma -i in.xml -o out.xml`
- output of that run is stored in `out.xml`
- rest of this lecture is about writing `in.xml`
  heavily based on a talk by Balint Joo

Basic XML file layout

```
<?xml version="1.0"?>
<chroma>
  <annotation>some comment</annotaion>
  <Param>
    <InlineMeasurements>...
    </InlineMeasurements>
    <nrow>8 8 8 16</nrow>
  </Param>
  <RND>...</RNG>
  <Cfg>
    <cfg_type>SCIDAC</cfg_type>
    <cfg_file>./conf.test.ildg</cfg_file>
  </Cfg>
</chroma>
```

Basic XML file layout

- everything enclosed by the `<chroma>` tag
- `<nrow>` specifies the lattice size
- `<InlineMeasurements>` lists the tasks to be performed by the `chroma` executable

Inline Measurements layout

```
<InlineMeasurements>
  <elem>
    <Name>MAKE_SOURCE</Name>
    <Frequency>1</Frequency>
    <Param>
    ... some task specific parameters ...
    </Param>
    <NamedObject>
      <gauge_id>default_gauge_field</gauge_id>
      ... other ids ...
    </NamedObject>
  </elem>
  <elem>...</elem>
</InlineMeasurements>
```

Inline Measurements layout

- Frequency Tag for `hmc`:
  do measurement if `update_no % frequency == 0`
- for `chroma`: `update_no = 0` always
- `<xml_file>` tag: write log output to that file
- `<NamedObject>` tag: a must for every task!
  - `<gauge_id>` tag must be specified!
    everything depends on the gauge field
  - the gauge field read from file
    is called `default_gauge_field`

Inline Measurement layout

```
<InlineMeasurements>
  <elem>...
  </elem>
  <elem>...</elem>
  <elem>...</elem>
</InlineMeasurements>
```

- `chroma` will work through the `<elem>` tags sequentially (each called a task)
- you can create objects, use and erase them again
- this is possible via the `<NamedObject>` tag

Other Basic XML tags

- `<Cfg>` get the default configuration
- `<RNG>` seed for the random number generator
- `<FermionAction>` set the fermion action to be used
  - `<FermState>`: to chose the boundary conditions `<FermionBC>` and e.g. stout smearing
- `<InvertParams>` parameters of the inverter
- all of them will be explained in the following

<Cfg>

```
<Cfg>
  <cfg_type>ABC</cfg_type>
  <cfg_file>abc</cfg_file>
</Cfg>
```

- <cfg_type>
  - read from file SZIN, SCIDAC, MILC, CPPACS,...
  - without file UNIT, DISORDERED, WEAK_FIELD
- SCIDAC reads ILDG format as well
- <cfg_file> the filename, ignored if not needed
- this taks becomes the <NamedObject> default_gauge_field

<RNG>

- used to set the random number seed
- linear congruential generator
  - modulus $m = 2^{47}$
  - increment $c = 0$
  - multiplier $a = 2^{36}M_3 + 2^{24}M_2 + 2^{12}M_1 + M_0$

```
<RNG>
  <Seed>
    <elem>M0</elem>
    <elem>M1</elem>
    <elem>M2</elem>
    <elem>M3</elem>
  </Seed>
</RNG>
```

<FermionAction>

```
<FermionAction>
  <FermAct>WILSON</FermAct>
  <Kappa>0.11</Kappa>
  <FermState>
     <Name>SIMPLE_FERM_STATE</Name>
     <FermionBC>
       <FermBC>SIMPLE_FERMBC</FermBC>
       <boundary>1 1 1 -1</boundary>
     </FermionBC>
  </FermState>
</FermionAction>
```

<FermionAction>

- <boundary> 1 = periodic, 0 = Dirichlet, -1 = antiperiodic
- <FermState> can be also <Name>STOUT_FERM_STATE</Name>
- then stout smearing is used.

```
<FermState>
 <Name>STOUT_FERM_STATE</Name>
 <rho>0.14</rho>
 <n_smear>6</n_smear>
 <orthog_dir>3</orthog_dir>
 ...
```

Wilson fermions

```
<FermionAction>
  <FermAct>WILSON</FermAct>
  <Mass>-0.1</Mass>
  <FermState/>
</FermionAction>
```

or equivalently

```
<FermionAction>
  <FermAct>WILSON</FermAct>
  <Kappa>0.1</Kappa>
  <FermState/>
</FermionAction>
```

Wilson clover fermions

```
<FermionAction>
  <FermAct>CLOVER</FermAct>
  <Mass>-0.001</Mass>
  <clovCoeff>1.0</clovCoeff>
  <FermState/>
</FermionAction>
```

- alternative to `<Mass>` also `<Kappa>` can be used
- `WILSON, CLOVER` refers here to even/odd preconditioned operators
  use e.g. `UNPRECONDITIONED_CLOVER` instead for no preconditioning

Domain Wall fermions

```
<FermionAction>
  <FermAct>DWF</FermAct>
  <OverMass>1.2</OverMass>
  <Mass>0.4</Mass>
  <N5>12</N5>
  <FermState/>
</FermionAction>
```

- one can specify the extension of 5^{th} direction using the <N5> tag
- other representations of domain wall fermions available

The Inverter

```
<InvertParam>
  <invType>CG_INVERTER</invType>
  <RsdCG>1.e-10</RsdCG>
  <MaxCG>1000</MaxCG>
</InvertParam>
```

- also available: multiple mass solver and `BICGSTAB`
- `<RsdCG>` is the target relative precision
- `<MaxCG>` the maximal iteration number

Generating a source

```
<elem>
  <Name>MAKE_SOURCE</Name>
  <Frequency>1</Frequency>
  <Param>
    <version>6</version>
    <Source>
      <version>2</version>
      <SourceType>POINT_SOURCE</SourceType>
      <j_decay>3</j_decay>
      <t_srce>0 0 0 0</t_srce>
    </Source>
  </Param>
  <NamedObject>
    <gauge_id>default_gauge_field</gauge_id>
    <source_id>pt_source</source_id>
  </NamedObject>
</elem>
```

Generating a source

- `POINT_SOURCE` is a point source
- `SHELL_SOURCE` is a smeared source
  - `<SmearingParam>` tag is used for smearing
- `<j_decay>` specifies the direction in which to measure e.g. a mass
- `<t_srce>` coordinates of the (center of) the source
- much more possible
- the source is created under the id specified with `<source_id>`
  important for later usage
- for more about smearing see tutorial
- most of this carries over to task `SINK_SMEAR`

Getting a propagator

```
<elem>
  <Name>PROPAGATOR</Name>
  <Frequency>1</Frequency>
  <Param>
    <version>10</version>
    <quarkSpinType>FULL</quarkSpinType>
    <obsvP>false</obsvP>
    <FermionAction/>
    <InvertParam/>
  </Param>
  <NamedObject>
    <gauge_id>default_gauge_field</gauge_id>
    <source_id>pt_source</source_id>
    <prop_id>pt_prop</prop_id>
  </NamedObject>
</elem>
```

Getting a propagator

- `<quarkSpinType>`
  - `FULL` relativistic 12 components
  - `UPPER` non-relativistic with $1 + \gamma_4$
  - `LOWER` non-relativistic with $1 - \gamma_4$
- `<obsvP>` only used for 5d domain wall fermions
- important: propagator needs input:
  - gauge field
  - source
- output is written to object with `<prop_id>`

Computing Correlators: HADRON_SPECTRUM

```
<elem>
  <Name>HADRON_SPECTRUM</Name>
  <Frequency>1</Frequency>
  <Param>
    <version>1</version>
    <MesonP>true</MesonP>
    <CurrentP>true</CurrentP>
    <BaryonP>true</BaryonP>
    <time_rev>false</time_rev>
    <mom2_max>3</mom2_max>
    <avg_equiv_mom>true</avg_equiv_mom>
  </Param>
  ...
```

Computing Correlators: HADRON_SPECTRUM

```
<elem>
  ...
  <NamedObject>
    <gauge_id>default_gauge_field</gauge_id>
    <sink_pairs>
      <elem>
        <first_id>sh_pt_sink_1</first_id>
        <second_id>sh_pt_sink_1</second_id>
      </elem>
      <elem>
        <first_id>sh_sh_sink_1</first_id>
        <second_id>sh_sh_sink_1</second_id>
      </elem>
    </sink_pairs>
  </NamedObject>
  <xml_file>hadspec.dat.xml</xml_file>
</elem>
```

Computing Correlators: HADRON_SPECTRUM

- Specify correlators to compute `MesonP, CurrentP, BaryonP`
- `<time_rev>true</time_rev>` can time revers baryons

- the id's used in `<sink_pairs>` must be output objects of task `SINK_SMEAR` (see tutorial)

- we combine quark and anti-quark prop

- and for different smearing combinations

- `<xml_file>` provides the filename where to dump the correlators...

Writing objects to disk

Example: writing a propagator to disk

```
<elem>
  <Name>QIO_WRITE_NAMED_OBJECT</Name>
  <Frequency>1</Frequency>
  <NamedObject>
    <object_id>sh_prop_1</object_id>
    <object_type>LatticePropagator</object_type>
  </NamedObject>
  <File>
    <file_name>./sh_prop_1</file_name>
    <file_volfmt>SINGLEFILE</file_volfmt>
  </File>
</elem>
```

Reading objects from disk

Reading it back into memory from disk

```
<elem>
  <Name>QIO_READ_NAMED_OBJECT</Name>
  <Frequency>1</Frequency>
  <NamedObject>
    <object_id>sh_prop_1</object_id>
    <object_type>LatticePropagator</object_type>
  </NamedObject>
  <File>
    <file_name>./sh_prop_1</file_name>
  </File>
</elem>
```

`<file_volfmt>` automatic for readin

Saving Memory: erasing objects again

Suppose we don't need the source anymore

```
<elem>
  <Name>ERASE_NAMED_OBJECT</Name>
  <Frequency>1</Frequency>
  <NamedObject>
    <object_id>sh_source_1</object_id>
  </NamedObject>
</elem>
```

maybe useful: list all named objects

```
<elem>
  <Name>LIST_NAMED_OBJECT</Name>
  <Frequency>1</Frequency>
</elem>
```

The Tutorial

Browse to

`people.physik.hu-berlin.de/~urbach/lap08/chroma.html`

and have fun!

Outline

1. The Chroma Library: a short introduction
   Tutorial
2. Writing Chroma XML input
3. The HMC algorithm
   using the harmonic oscillator as example
4. The HMC for QCD
   Tutorial
5. The HMC for QCD using Chroma
   Tutorial

The Working Example

- as an example to illustrate the HMC algorithm we shall use the harmonic oscillator [Creutz, Freedman, Annals Phys.132:427,1981]
- the action

$$S = \int_0^T dt \; \{ \; \frac{1}{2}\dot{x}^2 + \omega^2 x^2 \; \}$$

- discretise time direction $T = N \cdot \Delta t$

$$S(x) = \sum_{i=1}^{N} a \left[\frac{(x_{i+1} - x_i)^2}{2a^2} + \omega^2 x_i^2 \right]$$

- the set $x = \{x_i\}$, $i = 0, ..., N-1$ is called a configuration
- we impose periodic boundary conditions in time, i.e. $x_i \equiv x_{i+N}$

Metropolis Monte Carlo

- goal: generate configurations x according to distribution

$$e^{-S(x)}$$

- generate a Marcov Chain using the following algorithm

1. start with some configuration x
2. chose a test configuration x' with probability $P(x')$ $P(x) \neq 0 \; \forall x$
3. accept x' with probability

$$P(x \rightarrow x') = \min\{1, \exp[\Delta S = S(x') - S(x)]\}$$

4. continue with step 2

Main ingredience: detailed balance condition
$\exp(-S(x))P(x \rightarrow x') = \exp(-S(x'))P(x' \rightarrow s)$

Metropolis Monte Carlo

- how to obtain the test configuration x'?
  1. chose x' randomly
     completely uncorrelated
     in general one expects large ΔS
  2. use $x' = x + \delta x$ with random but small δx
     δx can be tuned for ΔS to be small
     correlation between x and x'
- keeping in mind ergodicity!
- in case the computation of ΔS is very expensive both choices turn out to be not efficient
- desired: a global update with large acceptance

The Hamiltonian Monte Carlo (Hybrid Monte Carlo)

[Duane, Kennedy, Pendleton, Roweth, 1987]

- Introduce p_i conjugate to fundamental fields x_i and a Hamiltonian

$$\mathcal{H} = \frac{1}{2}\sum_i p_i^2 + S(x)$$

- Molecular dynamics (MD) evolution of p and x by numerical integration of the corresponding equations of motion

$$(p, x) \quad \rightarrow \quad (p', x')$$

- Metropolis accept/reject step to correct for discretisation errors of the numerical integration

$$P_{\text{acc}} = \min\{1, \exp(-\Delta\mathcal{H} = \mathcal{H}(p', x') - \mathcal{H}(p, x))$$

HMC algorithm

In more detail

1. generate momenta p_i randomly from distribution

$$P \sim e^{-p^2/2}$$

   and compute initial Hamiltonian $\mathcal{H}$.

2. Integrate the equations of motion

$$\dot{x}_i = \frac{\partial \mathcal{H}}{\partial p_i} = p_i \qquad \dot{p}_i = -\frac{\partial \mathcal{H}}{\partial x_i} = -\frac{\partial S}{\partial x_i} \quad \forall i$$

   by means of some numerical integration scheme
   needs to be reversible and area preserving

3. the Hamiltonian is conserved up to discretisation errors

4. compute final Hamiltonian $\mathcal{H}'$ and accept/reject

$$P_{\text{acc}} = \min\{1, \exp(-\Delta\mathcal{H} = \mathcal{H}' - \mathcal{H})$$

   to correct for discretisation errors

HMC algorithm

- one can show this algorithm is exact
- it generates configurations according to distribution

$$\sim e^{-S(x)}$$

- as long as
  - integration is reversible (it usually is, up to round off)
  - integration is area preserving
  - the Molecular Dynamics update is ergodic
- useful property $\langle \exp(-\Delta\mathcal{H}) \rangle = 1$ (see tutorial)

Detailed Balance for HMC

- Need to show

$$e^{-S(x)}P(x \to x') = e^{-S(x')}P(x' \to x)$$

- $P(x \to x')$ is a product of

$$P_{\mathrm{G}} = e^{-\sum p^2/2}$$
$$P_{\mathrm{MD}}[(x,p) \to (x',p')] = P_{\mathrm{MD}}[(x',-p') \to (x,-p)]$$
$$P_{\mathrm{A}} = \min(1, \exp(-\Delta\mathcal{H}))$$

- so

$$e^{-S(x)}\, P(x \to x') = e^{-S(x)} \int \mathcal{D}p\, \mathcal{D}p'\, P_{\mathrm{G}}\, P_{\mathrm{MD}}\, P_{\mathrm{A}}$$
$$= \int \mathcal{D}p\, \mathcal{D}p'\, e^{-\mathcal{H}(x,p)} P_{\mathrm{MD}}\, P_{\mathrm{A}}$$

Detailed Balance for HMC

- From the Metropolis algorithm we know that

$$e^{-\mathcal{H}(x,p)}\, P_{\mathrm{A}}[(x,p) \to (x',p')] = e^{-\mathcal{H}(x',p')} P_{\mathrm{A}}[(x',p') \to (x,p)]$$

- and, because $\mathcal{H}$ is quadratic in p

$$\mathcal{H}(x,p) = \mathcal{H}(x,-p)$$

- hence, using all of the above and a change of variables we obtain

$$\begin{aligned} e^{-S(x)}\, P(x \to x') =& e^{-S(x')} \int \mathcal{D}p\, \mathcal{D}p' P_{\mathrm{G}}(p') \times \\ & \times P_{\mathrm{MD}}[(x',p') \to (x,p)]\, P_{\mathrm{A}}[(x',p') \to (x,p)] \\ =& e^{-S(x')}\, P(x' \to x) \end{aligned}$$

Detailed Balance for HMC

- the proof makes clear that we need

$$e^{-S(x)}\, P_{\mathrm{G}}(p) \propto e^{-\mathcal{H}(x,p)}$$

- the reversibility of the MD part is explicitly used
- area preserving is property of MD is explicitly used
- note: we do have reversibility violations due to round-off
- in fact: violation grow exponentially
  final investigation of this is missing

  [Jansen, Liu, 1997]

The Molecular Dynamics update

- a reversible and area preserving integration scheme is the so called leap-frog integration scheme
- Discrete updates for time step $\Delta\tau$

$$T_{\mathrm{x}}(\Delta\tau): \quad x \quad \rightarrow \quad x' = x + \Delta\tau p$$
$$T_{\mathrm{S}}(\Delta\tau): \quad p \quad \rightarrow \quad p' = p - \Delta\tau \frac{\partial S}{\partial x}$$

- basic Leap Frog time evolution step

$$T = T_{\mathrm{S}}(\Delta\tau/2)\; T_{\mathrm{x}}(\Delta\tau)\; T_{\mathrm{S}}(\Delta\tau/2)$$

- trajectory of length τ: $N_{\mathrm{MD}} = \tau/\Delta\tau$ successive applications of T
- $\mathcal{H}' = \mathcal{H} + \mathcal{O}(\Delta\tau^3)$

Reducing Discretisation errors: 2MN integrator

- higher order integration schemes can eliminate higher order errors in $\Delta\tau$
- but they often do not perform too well
- try to reduce errors by tuning an additional parameter
- popular choice: second order minimal norm (2MN)

2MN integration scheme

- basic 2MN time evolution step

$$T_{\mathrm{2MN}} = T_{\mathrm{S}}(\lambda\Delta\tau)\, T_{\mathrm{x}}(\Delta\tau/2)\, T_{\mathrm{S}}((1-2\lambda)\Delta\tau)\, T_{\mathrm{x}}(\Delta\tau/2)\, T_{\mathrm{S}}(\lambda\Delta\tau)$$

- trajectory of length τ: $N_{\mathrm{MD}} = \tau/\Delta\tau$ successive applications of T_{2MN}
- λ additional real tunable parameter
- choice $\lambda = 1/6$ is called Sexton-Weingarten integration scheme (see tutorial)
- $\lambda \approx 0.21$ is known to be close to optimal

[Takaishi, De Forcrand, hep-lat/0505020]

Multiple Time Scale Integration

[Sexton, Weingarten, 1992]

- Assume: $\mathcal{H} = \frac{1}{2}\sum p^2 + S_0(x) + S_1(x)$
- Define ($j = 0, 1$):

$$T_x(\Delta\tau): \quad x \quad \rightarrow \quad x' = x + \Delta\tau p$$

$$T_{S_j}(\Delta\tau): \quad p \quad \rightarrow \quad p' = p - \Delta\tau \frac{\partial S_j}{\partial x}$$

- and recursively:

$$T_0 = T_{S_0}(\Delta\tau_0/2)\; T_x(\Delta\tau_0)\; T_{S_0}(\Delta\tau_0/2)\,,$$

$$T_1 = T_{S_1}(\Delta\tau_1/2)\; [T_0]^{N_0}\; T_{S_1}(\Delta\tau_1/2)$$

- trajectory of length τ: $\left[T_1\right]^{N_1}$

Multiple Time Scale Integration

- time steps must fulfil: $N_1 = \tau / \Delta\tau_1 \,, \qquad N_0 = \Delta\tau_1 / \Delta\tau_0$
- S_0 must be computed $N_0 \cdot N_1$ times
  S_1 only N_1 times
- note the recursive structure!

$$\left[T_1\right]^{N_1} = \left[T_{S_1}(\Delta\tau_1/2)\,\left[T_0\right]^{N_0}\,T_{S_1}(\Delta\tau_1/2)\right]^{N_1}$$

- I'll discuss tomorrow why this might be useful

The Tutorial

Browse to

`people.physik.hu-berlin.de/~urbach/lap08/harmonic.html`

and have fun!

Outline

1. The Chroma Library: a short introduction
   Tutorial
2. Writing Chroma XML input
3. The HMC algorithm
   using the harmonic oscillator as example
   Tutorial
4. The HMC for QCD
5. The HMC for QCD using Chroma
   Tutorial

Quantum Chromodynamics is formally described by the Lagrange density:

$$\mathcal{L}_{\mathrm{QCD}} = \bar{\psi}(i\not{D} - m_q)\psi - \frac{1}{4} G_{\mu\nu} G^{\mu\nu}$$

Lattice regularisation: discretise Euclidean space-time

- hyper-cubic $L^3 \times T$-lattice with lattice spacing a
- derivatives $\Rightarrow$ finite differences
- integrals $\Rightarrow$ sums
- gauge potentials A_μ in $G_{\mu\nu} \Rightarrow$ link matrices U_μ

Why is it so expensive?

- we need to compute:

$$\mathcal{Z}_{\mathrm{QCD}} = \int \mathcal{D}\bar{\psi}\, \mathcal{D}\psi\, e^{-\bar{\psi}(\gamma_\mu D_\mu + m_0)\psi} \propto \det(\gamma_\mu D_\mu + m_0)$$

- determinant can be represented by bosonic fields:

$$\det(\gamma_\mu D_\mu + m_0) \propto \int \mathcal{D}\phi^\dagger\, \mathcal{D}\phi\, e^{-\phi^\dagger(\gamma_\mu D_\mu + m_0)^{-1}\phi}$$

- solving

$$\varphi = (\gamma_\mu D_\mu + m_0)^{-1}\phi$$

for φ becomes very expensive for small quark mass and large lattice extent L/a. ($\propto k = \lambda_{\max}/lambda_{\min}$)

The Hamiltonian Monte Carlo

- Introduce traceless Hermitian momenta $P_{x,\mu}$ conjugate to fundamental gauge fields $U_{x,\mu}$ and Hamiltonian

$$\mathcal{H} = \frac{1}{2}\sum_{x,\mu} P_{x,\mu}^2 + S[U] \; .$$

- Molecular dynamics evolution of P and U by numerical integration of the corresponding equations of motion.
- Metropolis accept/reject step to correct for discretization errors of the numerical integration.

The Hamiltonian Monte Carlo

- Molecular dynamics evolution of P and U by numerical integration of the corresponding equations of motion.

Multiple Time Scale Integration

- Assume: $\mathcal{H} = \frac{1}{2}\sum_{x,\mu} P^2_{x,\mu} + S_0 + S_1$
- Define ($j = 0, 1$):

$$\begin{aligned} T_{\mathrm{U}}(\Delta\tau): \quad U \quad &\rightarrow \quad U' = \exp\left(i\Delta\tau P\right) U\,, \\ T_{\mathrm{S}_j}(\Delta\tau): \quad P \quad &\rightarrow \quad P' = P - i\Delta\tau\delta S_j \end{aligned}$$

- and recursively:

$$T_0 = T_{\mathrm{S}_0}(\Delta\tau_0/2)\; T_{\mathrm{U}}(\Delta\tau_0)\; T_{\mathrm{S}_0}(\Delta\tau_0/2)\,,$$

$$T_1 = T_{\mathrm{S}_1}(\Delta\tau_1/2)\; [T_0]^{N_0}\; T_{\mathrm{S}_1}(\Delta\tau_1/2)$$

- trajectory of length τ: $\left[T_1\right]^{N_1}$

Preconditioning

- Most expensive part: fermion determinant
- Precondition by factorisation (with suitable C and E):

$$\det Q^2 = \det(C) \cdot \det(E)$$

  with C and E better "behaved" than Q^2.
- for instance
  - mass preconditioning [Hasenbusch]
  - polynomial filtering [Peardon, Sexton]
  - domain decomposition [Lüscher]
  - n^{th}-root [Clark, Kennedy]
- whereas often:
  - C is cheap
  - E is expensive

n^{th}-root Trick

- use the following factorisation

$$\det Q^2 = \sqrt{\det Q^2} \cdot \sqrt{\det Q^2}$$

- in terms of condition numbers

$$k \quad \rightarrow \quad 2 * \sqrt{k}$$

- in general

$$\det Q^2 = [(\det Q^2)^{1/n}]^n$$

- allows to save large factors

Hasenbusch Trick

- Precondition the fermion determinant ($Q = \gamma_5 D_{\mathrm{W}}$, $n_f = 2$):

$$\det Q^2 = \det\left[Q^2 + \mu^2\right] \cdot \det\left[\frac{Q^2}{Q^2 + \mu^2}\right].$$

[Hasenbusch, 2001]

- Corresponding effective action:

$$S_{\mathrm{eff}} = S_{\mathrm{G}} + \phi_1^\dagger \frac{1}{Q^2 + \mu^2} \phi_1 + \phi_2^\dagger \frac{Q^2 + \mu^2}{Q^2} \phi_2 = S_{\mathrm{G}} + S_{\mathrm{PF}_1} + S_{\mathrm{PF}_2}.$$

- Can be extended to $N_{\mathrm{PF}} > 2$ pseudo-fermion fields.
- Can be combined with even/odd preconditioning.

Why does it help?

If possible, tune preconditioner such that:

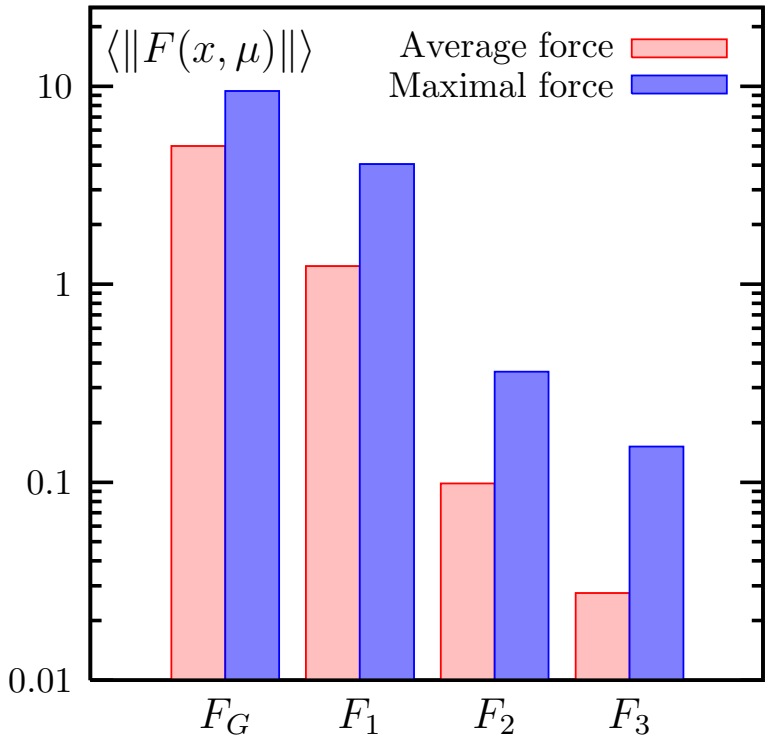
- the most expensive part (S_1)
  contributes the least to the total force
  $\Rightarrow$ can be integrated with large $\Delta\tau$
- the cheaper the action part, the smaller $\Delta\tau$
- different parts can be integrated on *different* time scales chosen according to their force magnitude:

$$\Delta\tau_j \|F_j\| = \text{const}$$

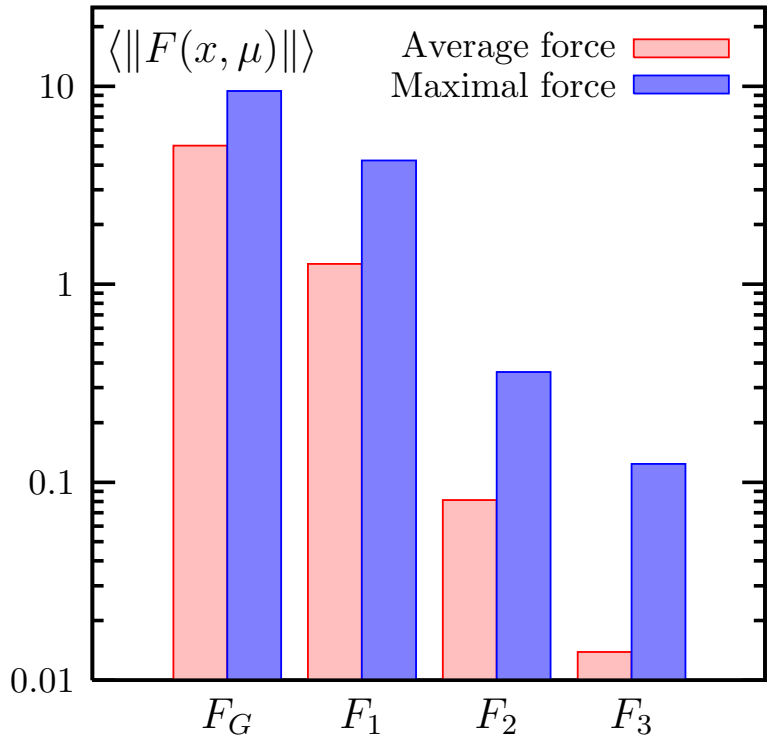
as a tuning guidline

Molecular Dynamics Forces

run with $m_{\mathrm{PS}} \approx 485$ MeV:



run with $m_{\mathrm{PS}} \approx 294$ MeV:



Further Improvements

- realise that in the MD you can do whatever you want
  - as long as you don't spoil the exactness of the HMC
  - and $\Delta\mathcal{H}$ is not too large
- reduced precision in the solver
- reduced precision Dirac operator
- space for ideas...

Further Improvements

- chronological predictor :
- predict from history of solutions a better initial guess for the next inversion
- depending on $\Delta\tau$ one sees a large improvement
- But take always care: reversibility violations!

Outline

1. The Chroma Library: a short introduction
   Tutorial
2. Writing Chroma XML input
3. The HMC algorithm
   using the harmonic oscillator as example
   Tutorial
4. The HMC for QCD
5. The HMC for QCD using Chroma
   Tutorial

Basic Building Blocks in XML

The HMC XML input file has the following stucture

```
<Params>
  <MCControl>
  ...
  </MCControl>
  <HMCTrj>
  ...
  </HMCTrj>
</Params>
```

MCControl tag

Controls the basic Markov chain parameters

- random number generator `<RNG>...</RNG>`
- starting configurations `<CFG>...</CFG>`
- number of trajectories
- inline measurements
- and more

Mostly self explaining, see the tutorial...

HMCTrj tag

- controls the physics and the algorithm to be simulated
- can/must contain several sub tags:

```
<HMCTrj>
  <nrow>... </nrow>
  <Monomials>... </Monomials>
  <Hamiltonian>... </Hamiltonian>
  <MDIntegrator>... </MDIntegrator>
</HMCTrj>
```

- discussed in more detail in the following

Monomials

- the Monomial tag may contain several elements for instance one for each part in the action.
- each monomial is named uniquely: `<monomial_id>` tag
- for example consider:

$$S = S_{\mathrm{g}} + \phi^{\dagger} \frac{1}{\hat{Q}^2} \phi$$

hence we need to create one gauge monomial and one fermion monomial:

Gauge monomial

- gauge monomial:

```
<elem>
  <Name>GAUGE_MONOMIAL</Name>
  <GaugeAction>
    <Name>WILSON_GAUGEACT</Name>
    <beta>5.7</beta>
    <GaugeBC>
      <Name>PERIODIC_GAUGEBC</Name>
    </GaugeBC>
  </GaugeAction>
  <NamedObject>
    <monomial_id>gauge</monomial_id>
  </NamedObject>
</elem>
```

Fermion monomial

```
<elem>
  <Name>TWO_FLAVOR_EOPREC_CONSTDET_FERM_MONOMIAL</Name>
  <InvertParam>
    <invType>CG_INVERTER</invType>
    <RsdCG>1.0e-12</RsdCG>
    <MaxCG>1000</MaxCG>
  </InvertParam>
  <FermionAction>
    <FermAct>WILSON</FermAct>
    <Kappa>0.11</Kappa>
    <FermionBC>
       <FermBC>SIMPLE_FERMBC</FermBC>
       <boundary>1 1 1 -1</boundary>
    </FermionBC>
  </FermionAction>
  ...
```

Fermion monomial cont.

```
    ...
    <ChronologicalPredictor>
      <Name>LAST_SOLUTION_4D_PREDICTOR</Name>
    </ChronologicalPredictor>
    <NamedObject>
      <monomial_id>wilsontwoflavour</monomial_id>
    </NamedObject>
  </elem>
```

Hamiltonian tag

Collection of monomials, without the momenta!

```
<Hamiltonian>
  <monomial_ids>
    <elem>wilsontwoflavour</elem>
    <elem>gauge</elem>
  </monomial_ids>
</Hamiltonian>
```

This will provide a Hamiltonian like

$$\mathcal{H} = \frac{1}{2}P^2 + S_{\mathrm{g}} + \phi^\dagger \frac{1}{Q^2}\phi$$

MDIntegrator tag

Simple leapfrog integrator with one timescale:

```
<MDIntegrator>
  <tau0>0.5</tau0>
  <Integrator>
    <Name>LCM_STS_LEAPFROG</Name>
    <n_steps>25</n_steps>
    <monomial_ids>
      <elem>wilsontwoflavour</elem>
      <elem>gauge</elem>
    </monomial_ids>
  </Integrator>
</MDIntegrator>
```

Multiple Timescales: the <SubIntegrator> tag

Assume the gauge part should be on a smaller timescale:

```
<tau0>1.0</tau0>
<Integrator>
  <Name>LCM_STS_LEAPFROG</Name>
  <n_steps>5</n_steps>
  <monomial_ids><elem>wilsontwoflavour</elem>
  </monomial_ids>
  <SubIntegrator>
    <Name>LCM_STS_LEAPFROG</Name>
    <n_steps>2</n_steps>
    <monomial_ids><elem>gauge</elem></monomial_ids>
  </SubIntegrator>
</Integrator>
```

Multiple Timescales: the <SubIntegrator> tag

- total trajectory length $\tau = 1$
- fermionic part on the outer timescale
  5 steps: $\Delta\tau_{\mathrm{f}} = 0.2$
- gauge part on the inner timescale
  $2 \cdot 5 = 10$ steps: $\Delta\tau_{\mathrm{g}} = 0.1$
- leap-frog on all time scales

Hasenbusch Trick

Introduce an additional monomial

```
<elem>
  <Name>
  TWO_FLAVOR_EOPREC_CONSTDET_HASENBUSCH_FERM_MONOMIAL
  </Name>
  <FermionAction>
    <!-- Parameters of physical operator -->
  </FermionAction>
  <FermionActionPrec>
    <!-- Parameters of prec. operator -->
  </FermionActionPrec>
  <NamedObject>
  <monomial_id>hasenbusch</monomial_id></NamedObject>
</elem>
```

Hasenbusch Trick

- This monomial represents a ratio of operators
- for instance:

$$S_{\mathrm{PF}_2} = \phi_2^\dagger \frac{Q^2 + \mu^2}{Q^2} \phi_2$$

- hence, the first monomial must be adopted to represent

$$S_{\mathrm{PF}_1} = \phi_1^\dagger \frac{1}{Q^2 + \mu^2} \phi_1$$

- the two monomials can then be integrated on different timescales

The <copyList> tag

- Assume you want reduced precision in the MD integration
- create an additional monomial:

```
<elem>
  <Name>TWO_FLAVOR_EOPREC_CONSTDET_FERM_MONOMIAL
  </Name>
  <InvertParam>
    <invType>CG_INVERTER</invType>
    <RsdCG>1.0e-7</RsdCG>
  ...
  <NamedObject>
    <monomial_id>wilsontwoflavour_md</monomial_id>
  </NamedObject>
</elem>
```

The <copyList> tag

Change MDIntegrator as follows:

```
<MDIntegrator>
  <copyList><elem>
    <copyFrom>wilsontwoflavour</copyFrom>
    <copyTo>wilsontwoflavour_md</copyTo>
  </elem></copyList>
  ...
    <monomial_ids>
      <elem>wilsontwoflavour_md</elem>
      <elem>gauge</elem>
    </monomial_ids>
  </Integrator>
</MDIntegrator>
```

Some general Remarks

- chroma writes configurations in single precision per default
  - not a general problem, all computations are in double precision
  - however, take care in reproducibility tests!
    - you will be not able to reproduce sub-parts of a longer run
- chroma does not write configuration after every traj. don't make the interval too large, if your machine is not 100% reliable.

The Tutorial

Browse to

`people.physik.hu-berlin.de/~urbach/lap08/hmc/`

and have fun!