Multigrid with HYPRE for Lattice QCD

Andrew Pochinsky

Massachusetts Institute of Technology

avp@mit.edu

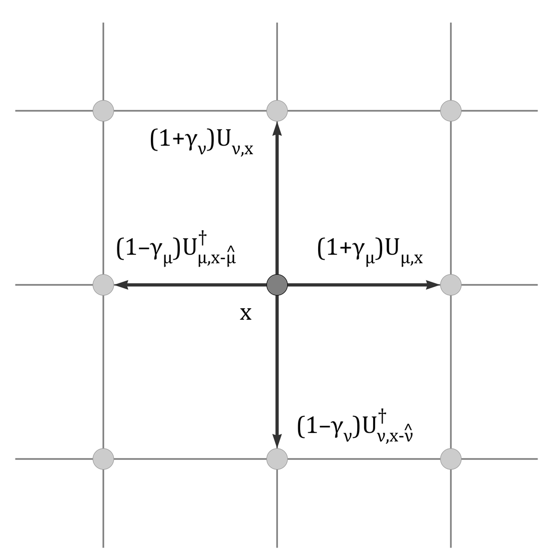
Massachusetts Institute of Technology

Abstract

In lattice QCD calculations, a significant amount of computation time is spent in solving the Dirac equation. Krylov solvers exhibit critical slowing down when the lattice is large and simulation parameters are in a physically interesting region. Multigrid approach promises robust linear solver algorithms with substantial speedup compared to other iterative methods. We present a tool that connects the HYPRE's [1] collection of linear solvers with the body of USQCD code [2] thus allowing one to explore the space of multigrid algorithms while using actual large gauge configurations and corresponding Dirac operators.
This presentation describes a joint project between the SciDAC–3 project Computing Properties of Hadrons, Nuclei and Nuclear Matter from Quantum Chromodynamics *and the FASTMath SciDAC Institute.*
Our FASTMath collaborators are Rob Falgout and Chris Schroeder of LLNL.

Lattice QCD

Lattice QCD is one of the most demanding computational science fields. To contribute significantly to the national NP program, LQCD calculations must move to the larger lattice sizes and simulation parameters approaching the physical point. New algorithms for solving the discretized Dirac equation need to be developed in order to avoid critical slowing down. The Dirac equation is essential both for configuration generation and calculating observables directly related to experiment.



Presently several formulations of fermion actions are used in different areas of the USQCD NP program.

- Staggered fermions
- Wilson–clover fermions
- Domain wall fermions (several kinds)

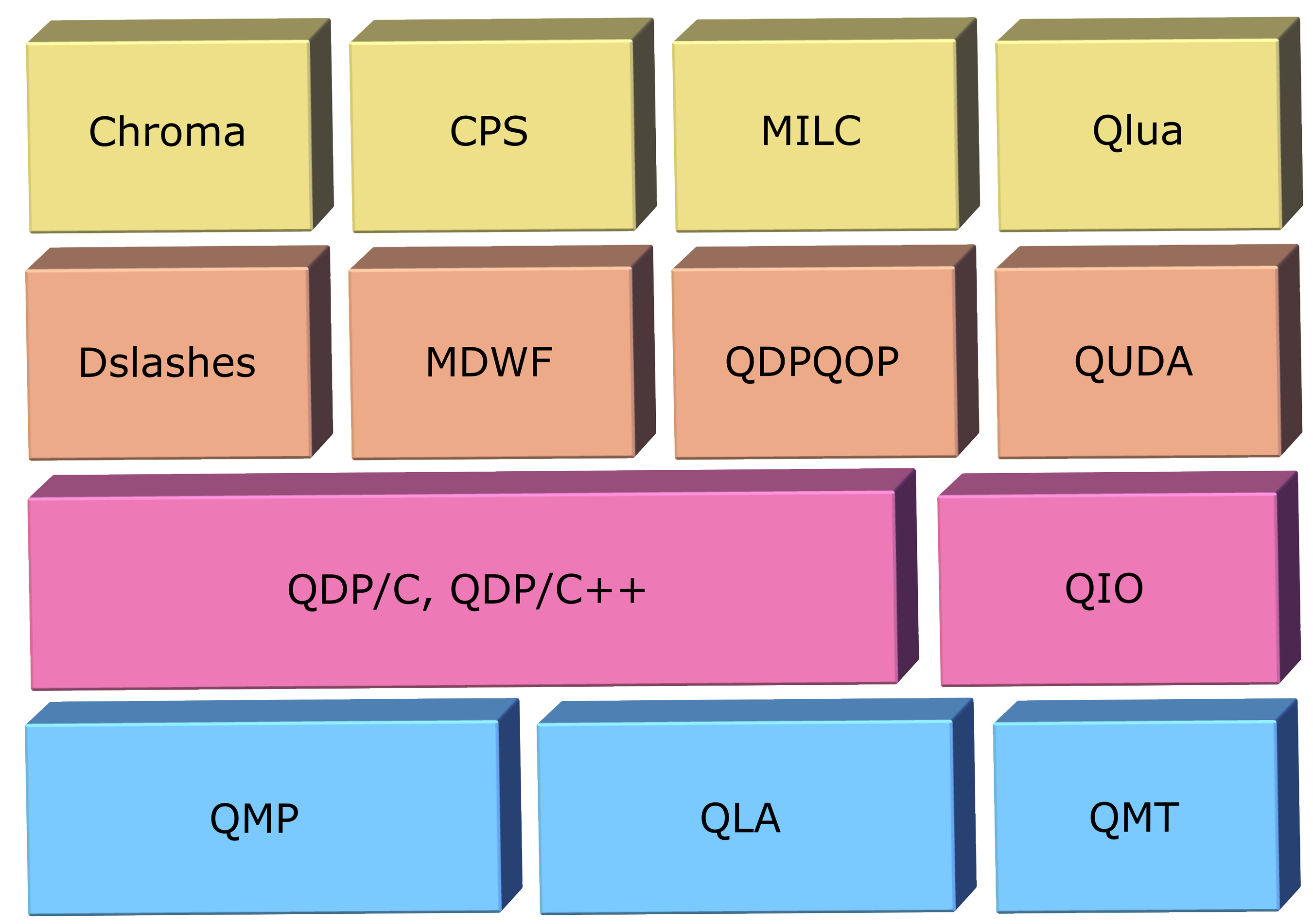
Application of accelerated solver to lattice quantum field theories other than QCD is also of interest.

Multigrid

Multigrid methods form a group of algorithms for solving systems of linear equations using decomposition of the vector space into a hierarchy of subspaces. They are especially useful for systems related to discretized linear differential equations and are an example of techniques very useful in problems exhibiting multiple scales of behavior. Multigrid methods are essentially linear solvers and as such, they can be used both as solvers and as preconditioners.
The main idea of multigrid is to improve the convergence of a basic iterative method by using a solution of a smaller (coarse) related problem as a guess to the solution of the original problem. This principle is similar to interpolation between coarser and finer grids. The typical application for multigrid is in the numerical solution of elliptic partial differential equations in two or more dimensions.
Multigrid methods can be applied in combination with any of the common linear solver techniques. In many cases, multigrid methods are among the fastest solution techniques presently known. Unlike other methods, multigrid methods are general and surprisingly robust. They do not depend on special properties of the equation.

LQCD Software

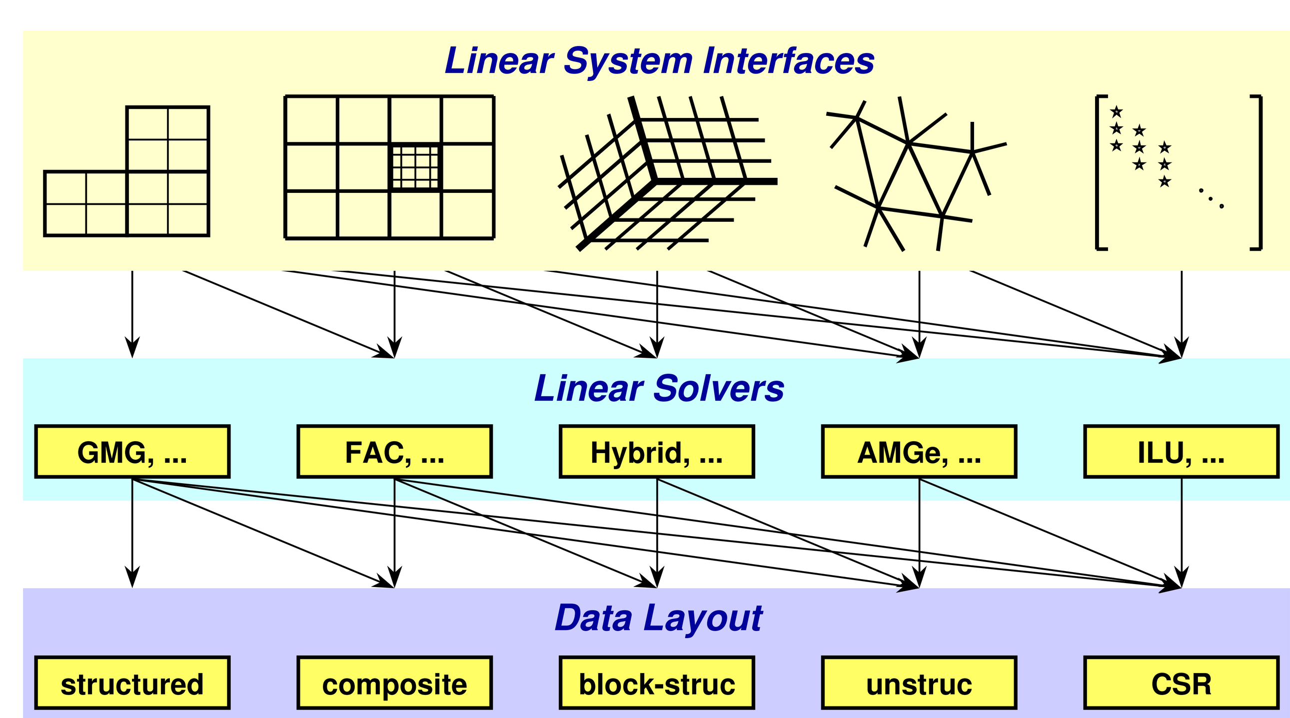
Whe USQCD software suite enables lattice QCD computations to be performed with high performance across a variety of architectures, including leadership facilities, custom and emerging architectures, and commodity clusters. This software is made up of software libraries that can be used by higher–level applications.



We have chosen Qlua [3] as our development platform for its simplicity and modularity.

HYPRE

The FASTMath SciDAC Institute develops HYPRE [1], a software library of high performance preconditioners and solvers for the solution of large, sparse linear systems of equations. The primary goal of the HYPRE library is to provide users with advanced parallel preconditioners. The present project uses the library's parallel multigrid solvers for structured grids. The HYPRE's conceptual linear system interfaces are further abstracted via the HQL intermediate layer to better map into lattice QCD data types and operations.



To support LQCD astractions, two extensions are required to the HYPRE's core:

- support for more than three spacial dimensions
- support for complex numbers

HQL

HQL is an abstraction software layer that glues HYPRE and LQCD codes together. Its goal is to isolate design peculiarities of its respective clients from each other and provide necessary translation services. Allowing LQCD code to talk to HQL instead of HYPRE simplifies software design and provides a lightweight mechanism for a future transition to high performance MG inverters if necessary. It also preserves QCD domain-specific symmetries.

High Level Interfaces

Linear Operators

For a linear operator, one needs to define its stensil structure and non-trivial gamma–matrix and gauge factors. Domain wall fermion operator may be described either as 5-d or in 4-d using an extra `flavor` attribute (not shown).

```
function wilson_hql(U, kappa)
   local op = {}
   local L = U[1].lattice
   local hg = qcd.hql{Lattice = L, Colors = 3}
   local i, j
   local stencil = {}
   local function make_offset(n, d)
      local offset = {}
      for j = 1, #L do offset[j] = 0 end
      offset[n+1] = d
      return offset
   end
   local offset = make_offset()
   stencil[#stencil+1] = {offset = offset}
   for i = 0, #L - 1 do
      offset = make_offset(i, 1)
      stencil[#stencil+1] = {offset = offset,
                             gamma = -kappa * (1 - gamma{mu=i}),
                             U = U[i+1]}
      offset = make_offset(i, -1)
      stencil[#stencil+1] = {offset = offset,
                             gamma = -kappa * (1 + gamma{mu=i}),
                             U = U[i+1]:adjoin():shift(i, "from_backward")}
   end
   local WM = hg:matrix(stencil)
   function op:vector(v) return WM:vector(v) end
   function op:export(v) return v:export()   end
   function op:apply(v)  return WM:apply(v)  end
   function op:dot(a,b)  return qcd.dot(a,b) end
   return op
end
```

Primitive Solvers

A Qlua mechanism for solvers and preconditioners high–level description is straightforward; its simplicity and expressive power need to be balanced for optimal useability. The design and the implementation are work in progress.

Solver calculus

It appears worthwhile to provide a way to combine preconditioners and solvers as well as to chain preconditioners, thus providing a kind of solver calculus to the user. Details of both the design and the implementation are being worked out.

References

[1] http://computation.llnl.gov/casc/linear_solvers/sls_hypre.html

[2] http://www.usqcd.org/

[3] https://usqcd.lns.mit.edu/redmine/projects/qlua_code