Frameworks for evolution and algorithm exploration

James C. Osborn

Argonne National Laboratory

USQCD Algorithms and Computing Workshop

November 10-11, 2011, LLNL

Lattice QCD algorithms

- Sparse linear solvers
  - Hermitian / non-Hermitian, positive definite / indefinite
  - Multishift
  - Mixed precision
  - Preconditioners, deflation, multigrid
- Hybrid molecular dynamics / Monte Carlo
  - Symmetric, symplectic integrators
  - Polynomials, rational functions, preconditioning
  - Multi-timescale
  - Tuning with shadow Hamiltonians / Poisson brackets
  - Force-gradient terms
  - Domain Decomposition HMC, multigrid

Lattice QCD(++) actions

- Main quark discretizations
  - Domain wall (5d)
  - Staggered (4d)
  - Wilson (4d)
- QCD
  - gauge field (gluons) are SU(3) matrices
  - quarks in fundamental representation
  - 2 light flavors of quarks
- Growing interest in non-QCD theories (beyond standard model)
  - Large number of quark flavors
  - Arbitrary Nc [SU(Nc)]
  - Other quark representations (adjoint, 2-index symmetric, …)
  - Other gauge groups
  - Arbitrary dimension

Lattice QCD codes

- Main application suites used in US
  - Chroma
  - CPS (Columbia Physics System)
  - MILC (MIMD Lattice Computation) collaboration code
- LQCD SciDAC Libraries
  - Communications (QMP), linear algebra (QLA), I/O (QIO), data parallel (QDP/QDP++)
  - Plus many "level 3" libraries (solvers, force terms)
- Application suites make use of LQCD libraries to varying degrees
  - Chroma built completely on top of LQCD SciDAC libraries
  - CPS & MILC use QMP and QIO libraries
  - All can use some "level 3" libraries

Lattice QCD development

- Performance optimization
  - A lot of work on “level 3” codes
  - Some optimization in other LQCD libraries
  - Also some optimization directly in application suites

- Algorithmic research
  - Solvers: in stand alone “level 3”, or built into application suite
  - HMC: in application suites, no stand alone framework previously available
  - Application suites provide a lot of useful tools for testing new algorithms, but may not be flexible enough (multigrid), or may be difficult to learn/modify
  - Simple but flexible frameworks can make testing algorithms easier, don't need to modify application suite until final algorithm has been found
  - Can incorporate improvements directly into application, or through “level 3”

Sparse solvers for LQCD

- Standard algorithms
  - CG, BiCGStab
  - Even/odd preconditioning
  - (re)implemented in every application suite
- "Advanced" algorithms
  - EigCG, multigrid, ...
  - More complicated to implement
  - Can be reimplemented, but much easier to use as external library, if possible

Sparse solvers for LQCD

- Current multigrid code
  - implemented on top of SciDAC QDP/C library
  - plan to package as “level 3” library, but not done yet
  - supports even/odd preconditioning, mixed precision
  - some optimizations for x86, Blue Gene/L,P
  - works great for Wilson-clover quarks (up to 25x speedup)
  - have implementations for staggered and domain wall quarks, but algorithm not effective yet

Opportunities for Collaboration

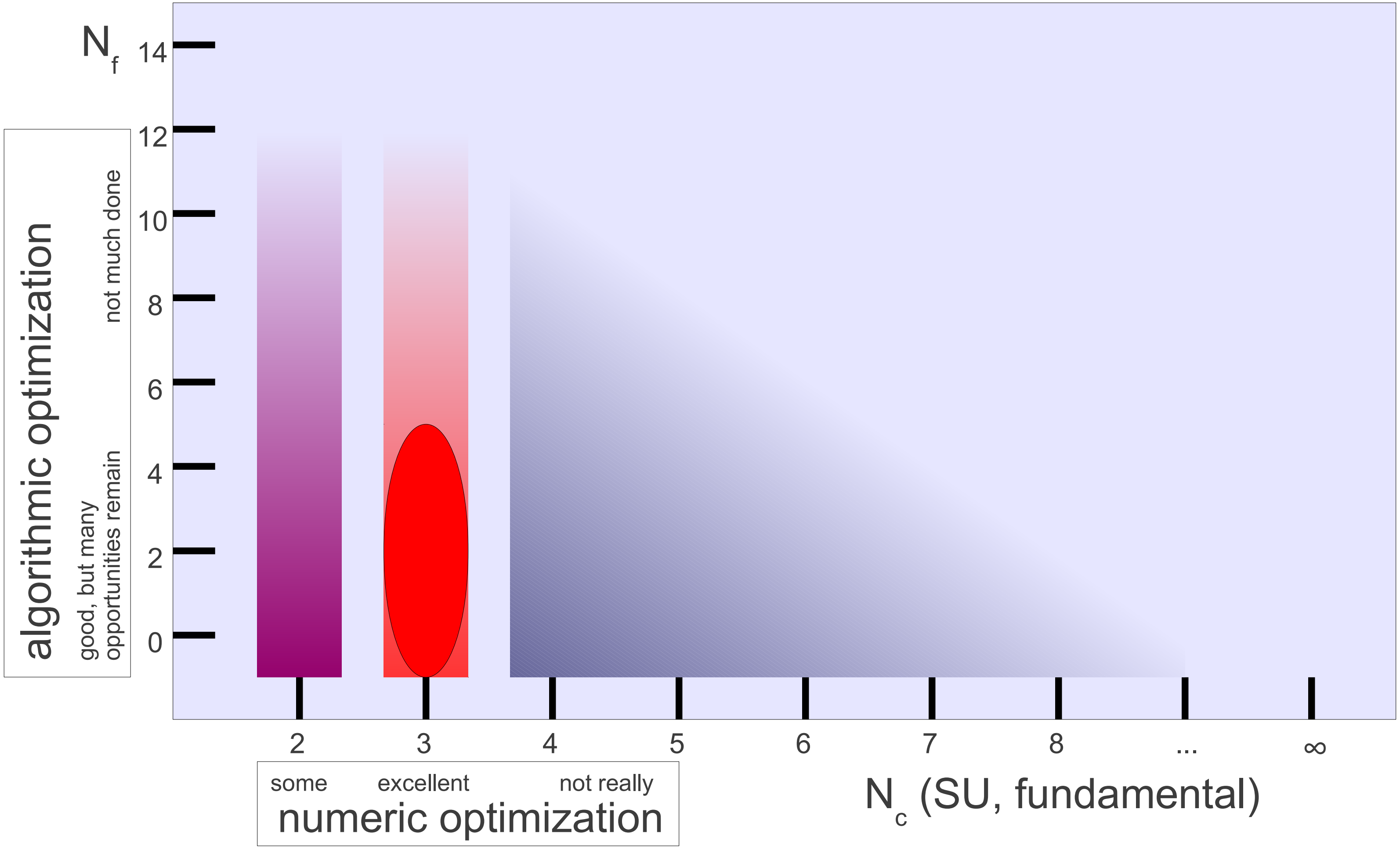
- Tools
  - Write solvers for LQCD in Hypre and/or PETSc (other useful packages?) (domain wall, staggered and/or Wilson)
  - Extend solvers to incorporate multigrid or other preconditioners important to LQCD
  - Integrate solvers into application suites as "level 3"

- Algorithms
  - Develop efficient multigrid for domain wall and staggered
  - Improve Wilson multigrid algorithm (setup)
  - Test other methods for improving solvers

Configuration generation

- Uses HMD/HMC
- Requires few main routines (solver, force terms, gauge action, Dslash), plus a handful of extra routines (random source, matrix exponential (3x3), …)
- Application suites have code for major quark types
- Current focus is on QCD
  - Gauge fields are SU(3) matrices
  - 2 light quarks + a few heavier ones
- Increasing interest in theories other than QCD
- Currently have some support for theories other than QCD, but strong desire for more

Current community lattice software



New Framework for Unified Evolution of Lattices (FUEL)

- High level layer focused on gauge configuration generation
  - algorithmic abstraction: generation algorithm independent of gauge group, action, etc.
  - easy to write new high-level algorithms, tune parameters
  - easy to plug in new routines
  - new routines can be written in any other language/framework
  - perfect for scripting language
- Scripting language requirements
  - Small
  - Easy to use
  - Easy to port
  - Easy to embed and interface with existing or new libraries

Lua

- Small, simple, fast and powerful scripting language
- Developed at Computer Graphics Technology Group (Tecgraf) at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), in Brazil
- Name means “moon” in Portuguese
- About 17k lines of ANSI C (easily ported)
- Designed to be embedded and easily interface with C libraries
- Liberal MIT license

Initial prototype

- Initially using SciDAC QDP/C and QOPQDP libraries to provide needed routines
- Supports SU(3) lattice generation with HISQ (highly improved staggered quarks)
  - using RHMC (Rational Hybrid Monte Carlo) algorithm
  - also supports HMC with mass precondition (suitable for $N_f = 4,8,12,...$)
- Matches conventions of current MILC code (and can parse same input files)
- Size:
  - QOPQDP/Lua interface: 3k lines
  - (R)HMC/bookkeeping in Lua: 1k lines

Hybrid Monte Carlo example

```
function hmcstep(fields, params)
  fields:save()
  local Sold = fields:action()

  local intparams =
         setupint(fields, params)
  integrate(fields, intparams)

  local Snew = fields:action()
```

```
  local ds = Snew - Sold
  local p = math.exp(-ds)
  local r = globalRand()
  if( r > p ) then -- reject
    fields:reject()
  else
    fields:accept()
  end
end
```

Integration example

- Integration also abstracted, e.g. leapfrog (1 field and 1 force term):

```
fields:updateField(1, eps/2)
fields:updateMomentum(1, 1, eps)
fields:updateField(1, eps/2)
```

- Actual code completely general
  - arbitrary integration patterns (leapfrog; Omelyan, et al.; custom)
  - any number of fields and force terms per field
  - different number of steps for each force term

Current plans

- Add domain wall quark support
- Add QUDA (GPU) routines as alternatives
- Add SU(2) support

- Use as testbed for algorithmic research
  - improve HMC for large N_f (e.g. mass preconditioning)
  - improved integrators (e.g. force gradient)
  - plug in improved solvers (e.g. multigrid)

Opportunities for collaboration

- Tools
  - Extend solver framework (Hypre, PETSc, etc.) to do complete evolution
  - Wrap solver framework in another framework (FUEL, …)
  - Add advanced integrators to framework
- Algorithm research
  - Test and tune improved integrators for QCD
  - Test and tune improved integrators for large Nf, other actions
  - Add closer integration with solvers and integration (DD-HMC, multigrid)

Summary

- Main LQCD algorithms: sparse solvers, HMD/HMC (integrators)
- Both have seen large algorithmic advances;
  much more possible
- Simple, flexible frameworks provide an efficient means to test new algorithms
- USQCD has been designing some frameworks,
  but can't handle everything, need more algorithms/flexibility
- Find ways to integrate other packages with LQCD codes to leverage new features and help with algorithm development and deployment