Computational Methods (practice) - Lecture 5

Peter Boyle (BNL, Edinburgh)

- Walk through an example code: u/d/s/c Meson spectrum
- Under 300 lines of code
- How to make Point, Z2, Gaussian, Sequential propagators
- Final words

Observables

Importance sampling has reduced:

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int_U e^{-S_G[U]} \mathcal{O}(U) dU \to \frac{1}{N} \sum_i \mathcal{O}(U_i)$$

- Zero momentum pion, kaon or B meson two point function

$$\sum_x \langle \bar{u} \gamma_0 \gamma_5 d(x,t) \bar{d} \gamma_0 \gamma_5 u(0,0) \rangle = \frac{1}{N} \sum_i \mathrm{trace}\{\gamma_0 \gamma_5 M_d^{-1}(x,t;0,0) \gamma_0 \gamma_5 M_u^{-1}(0,0;x,t)\}$$

- Euclidean space $\propto Ae^{-mt}$
- Tune bare mass until interacting meson mass is correct, prefactor gives pion, kaon, B meson decay constant
- etc..

Hadronic observables

- Many ways to use Grid to assemble Hadronic observables
- Hadrons https://github.com/aportelli/Hadrons
- GPT https://github.com/clehner/gpt
- As a library (e.g. CPS, MILC, Chris Kelly)
- Write your own?
  - to add **new** functionaly to any of the above, it is necessary to know how to write in Grid
  - developing simple code is a useful base

Example code

- Developed in Grid in about 6h, Mon 30th Aug
- Only loosely tested: intended to be illustrative

https://github.com/paboyle/Grid/blob/develop/examples/Example_Mobius_spectrum.cc

```
template<class Gimpl,class Field> class CovariantLaplacianCshift : public SparseMatrixBase<Field>
template<class Field> void GaussianSmear(LatticeGaugeField &U,Field &unsmeared,Field &smeared);
void PointSource(Coordinate &coor,LatticePropagator &source);
void Z2WallSource(GridParallelRNG &RNG,int tslice,LatticePropagator &source);
void GaussianSource(Coordinate &site,LatticeGaugeField &U,LatticePropagator &source);
void GaussianWallSource(GridParallelRNG &RNG,int tslice,LatticeGaugeField &U,LatticePropagator &source);
void SequentialSource(int tslice,Coordinate &mom,LatticePropagator &spectator,LatticePropagator &source);
void MakePhase(Coordinate mom,LatticeComplex &phase);
template<class Action> void Solve(Action &D,LatticePropagator &source,LatticePropagator &propagator);
void MesonTrace(std::string file,LatticePropagator &q1,LatticePropagator &q2,LatticeComplex &phase)
```

Gist of the programme

```
LatticePropagator point_source(UGrid);
LatticePropagator wall_source(UGrid);
LatticePropagator gaussian_source(UGrid);

Coordinate Origin({0,0,0,0});
PointSource   (Origin,point_source);
Z2WallSource  (RNG4,0,wall_source);
GaussianSource(Origin,Umu,gaussian_source);

std::vector<LatticePropagator> PointProps(nmass,UGrid);
std::vector<LatticePropagator> GaussProps(nmass,UGrid);
std::vector<LatticePropagator> Z2Props   (nmass,UGrid);

for(int m=0;m<nmass;m++) {

  Solve(*FermActs[m],point_source   ,PointProps[m]);
  Solve(*FermActs[m],gaussian_source,GaussProps[m]);
  Solve(*FermActs[m],wall_source    ,Z2Props[m]);

}

LatticeComplex phase(UGrid);
Coordinate mom({0,0,0,0});
MakePhase(mom,phase);

for(int m1=0 ;m1<nmass;m1++) {
for(int m2=m1;m2<nmass;m2++) {
  std::stringstream ssp,ssg,ssz;

  ssp<<config<< "_m" << m1 << "_m"<< m2 << "_point_meson.xml";
  ssg<<config<< "_m" << m1 << "_m"<< m2 << "_smeared_meson.xml";
  ssz<<config<< "_m" << m1 << "_m"<< m2 << "_wall_meson.xml";

  MesonTrace(ssp.str(),PointProps[m1],PointProps[m2],phase);
  MesonTrace(ssg.str(),GaussProps[m1],GaussProps[m2],phase);
  MesonTrace(ssz.str(),Z2Props[m1],Z2Props[m2],phase);
}}
```

Loading a configuration

```
LatticeGaugeField Umu(UGrid);
std::string config;
if( argc > 1 && argv[1][0] != '-' )
{
  std::cout<<GridLogMessage <<"Loading configuration from "<<argv[1]<<std::endl;
  FieldMetaData header;
  NerscIO::readConfiguration(Umu, header, argv[1]);
  config=argv[1];
}
else
{
  std::cout<<GridLogMessage <<"Using hot configuration"<<std::endl;
  SU<Nc>::ColdConfiguration(Umu);
  //    SU<Nc>::HotConfiguration(RNG4,Umu);
  config="ColdConfig";
}
```

- SciDAC/ILDG and Binary formats too
- MPI2IO is used

Fermion action

```
std::vector<RealD> masses({ 0.03,0.04,0.45} ); // u/d, s, c ??

int nmass = masses.size();

std::vector<MobiusFermionR *> FermActs;

std::cout<<GridLogMessage <<"======================"<<std::endl;
std::cout<<GridLogMessage <<"MobiusFermion action as Scaled Shamir kernel"<<std::endl;
std::cout<<GridLogMessage <<"======================"<<std::endl;

for(auto mass: masses) {

  RealD M5=1.0;
  RealD b=1.5;// Scale factor b+c=2, b-c=1
  RealD c=0.5;

  FermActs.push_back(new MobiusFermionR(Umu,*FGrid,*FrbGrid,*UGrid,*UrbGrid,mass,M5,b,c));

}
```

- Wilson, Clover, Twisted mass, Staggered, other chiral fermions similar

Red Black Solver

```
template<class Action>
void Solve(Action &D,LatticePropagator &source,LatticePropagator &propagator)
{
  GridBase *UGrid = D.GaugeGrid();
  GridBase *FGrid = D.FermionGrid();

  LatticeFermion src4  (UGrid);
  LatticeFermion src5  (FGrid);
  LatticeFermion result5(FGrid);
  LatticeFermion result4(UGrid);

  ConjugateGradient<LatticeFermion> CG(1.0e-8,100000);
  SchurRedBlackDiagMooeeSolve<LatticeFermion> schur(CG);
  ZeroGuesser<LatticeFermion> ZG; // Could be a DeflatedGuesser if have eigenvectors
  for(int s=0;s<Nd;s++){
    for(int c=0;c<Nc;c++){
      PropToFerm<Action>(src4,source,s,c);

      D.ImportPhysicalFermionSource(src4,src5);

      result5=Zero();
      schur(D,src5,result5,ZG);
      std::cout<<GridLogMessage
       <<"spin "<<s<<" color "<<c
       <<" norm2(src5d) "   <<norm2(src5)
               <<" norm2(result5d) "<<norm2(result5)<<std::endl;

      D.ExportPhysicalFermionSolution(result5,result4);

      FermToProp<Action>(propagator,result4,s,c);
    }
  }
}
```

- Interface works for both 4d and 5d fermion types

Sources

```
void PointSource(Coordinate &coor,LatticePropagator &source)
{
  source=Zero();
  SpinColourMatrix kronecker; kronecker=1.0;
  pokeSite(kronecker,source,coor);
}
void Z2WallSource(GridParallelRNG &RNG,int tslice,LatticePropagator &source)
{
  GridBase *grid = source.Grid();
  LatticeComplex noise(grid);
  LatticeComplex zz(grid); zz=Zero();
  LatticeInteger t(grid);

  RealD nrm=1.0/sqrt(2);
  bernoulli(RNG, noise); // 0,1 50:50

  noise = (2.*noise - Complex(1,1))*nrm;

  LatticeCoordinate(t,Tdir);
  noise = where(t==Integer(tslice), noise, zz);

  source = 1.0;
  source = source*noise;
}
```

Smearing

- Reuse the smearing we developed in earlier lectures!

```
template<class Field>
void GaussianSmear(LatticeGaugeField &U,Field &unsmeared,Field &smeared)
{
  typedef CovariantLaplacianCshift <PeriodicGimplR,Field> Laplacian_t;
  Laplacian_t Laplacian(U);

  Integer Iterations = 40;
  Real width = 2.0;
  Real coeff = (width*width) / Real(4*Iterations);

  Field tmp(U.Grid());
  smeared=unsmeared;
  //  chi = (1-p^2/2N)^N kronecker
  for(int n = 0; n < Iterations; ++n) {
    Laplacian.M(smeared,tmp);
    smeared = smeared - coeff*tmp;
  }
}
void GaussianSource(Coordinate &site,LatticeGaugeField &U,LatticePropagator &source)
{
  LatticePropagator tmp(source.Grid());
  PointSource(site,source);
  std::cout << " GaussianSource Kronecker "<< norm2(source)<<std::endl;
  tmp = source;
  GaussianSmear(U,tmp,source);
  std::cout << " GaussianSource Smeared "<< norm2(source)<<std::endl;
}
void GaussianWallSource(GridParallelRNG &RNG,int tslice,LatticeGaugeField &U,LatticePropagator &source)
{
  Z2WallSource(RNG,tslice,source);
  auto tmp = source;
  GaussianSmear(U,tmp,source);
}
```

Meson three point functions

- Use sequential source approach; contract an extended propagator via standard meson contraction

```
void SequentialSource(int tslice,Coordinate &mom,LatticePropagator &spectator,LatticePropagator &source)
{
  assert(mom.size()==Nd);
  assert(mom[Tdir] == 0);

  GridBase * grid = spectator.Grid();
  Gamma G5(Gamma::Algebra::Gamma5);

  LatticeInteger ts(grid);
  LatticeCoordinate(ts,Tdir);
  source = Zero();
  source = where(ts==Integer(tslice),spectator,source); // Stick in a slice of the spectator, zero everywhere else

  LatticeComplex phase(grid);
  MakePhase(mom,phase);

  source = G5*source *phase;
}
```

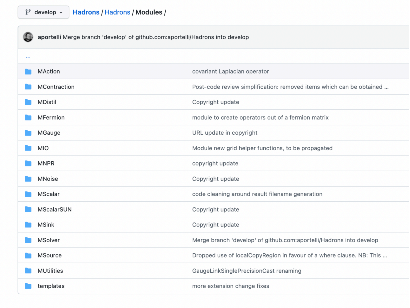
Contractions

```
class MesonFile: Serializable {
public:
  GRID_SERIALIZABLE_CLASS_MEMBERS(MesonFile, std::vector<std::vector<Complex> >, data);
};

void MesonTrace(std::string file,LatticePropagator &q1,LatticePropagator &q2,LatticeComplex &phase)
{
  const int nchannel=4;
  Gamma::Algebra Gammas[nchannel][2] = {
    {Gamma::Algebra::Gamma5      ,Gamma::Algebra::Gamma5},
    {Gamma::Algebra::GammaTGamma5,Gamma::Algebra::GammaTGamma5},
    {Gamma::Algebra::GammaTGamma5,Gamma::Algebra::Gamma5},
    {Gamma::Algebra::Gamma5      ,Gamma::Algebra::GammaTGamma5}
  };

  Gamma G5(Gamma::Algebra::Gamma5);

  LatticeComplex meson_CF(q1.Grid());
  MesonFile MF;

  for(int ch=0;ch<nchannel;ch++){

    Gamma Gsrc(Gammas[ch][0]);
    Gamma Gsnk(Gammas[ch][1]);

    meson_CF = trace(G5*adj(q1)*G5*Gsnk*q2*adj(Gsrc));

    std::vector<TComplex> meson_T;
    sliceSum(meson_CF,meson_T, Tdir);

    int nt=meson_T.size();

    std::vector<Complex> corr(nt);
    for(int t=0;t<nt;t++){
      corr[t] = TensorRemove(meson_T[t]); // Yes this is ugly, not figured a work around
      std::cout << " channel "<<ch<<" t "<<t<<" " <<corr[t]<<std::endl;
    }
    MF.data.push_back(corr);
  }

  {
    XmlWriter WR(file);
    write(WR,"MesonFile",MF);
  }
}
```

Hadrons

- Connects many of these ideas and more in reusable modules
- Connect outputs to inputs in dataflow style programming/graphs

https://github.com/aportelli/Hadrons/tree/develop/Hadrons/Modules



develop Hadrons / Hadrons / Modules /

aportelli Merge branch 'develop' of github.com:aportelli/Hadrons into develop

..	
MAction	covariant Laplacian operator
MContraction	Post-code review simplification: removed items which can be obtained ...
MDistil	Copyright update
MFermion	module to create operators out of a fermion matrix
MGauge	URL update in copyright
MIO	Module new grid helper functions, to be propagated
MNPR	copyright update
MNoise	Copyright update
MScalar	code cleaning around result filename generation
MScalarSUN	Copyright update
MSink	Copyright update
MSolver	Merge branch 'develop' of github.com:aportelli/Hadrons into develop
MSource	Dropped use of localCopyRegion in favour of a where clause. NB: This ...
MUtilities	GaugeLinkSinglePrecisionCast renaming
templates	more extension change fixes

Final words

- Aims
  - convince you that LQCD software can be elegant, portable and fast
  - convince you that algorithms can be easy to implement
  - convince you that code can be elegant, portable and fast
  - convince you to get your hands dirty!
  - draw connections between a sample of the core algorithms & methods of LQCD
  - keep the exposition simple while still covering the depth
- Please provide feedback: what worked and what didn't
- I hope you enjoyed the course