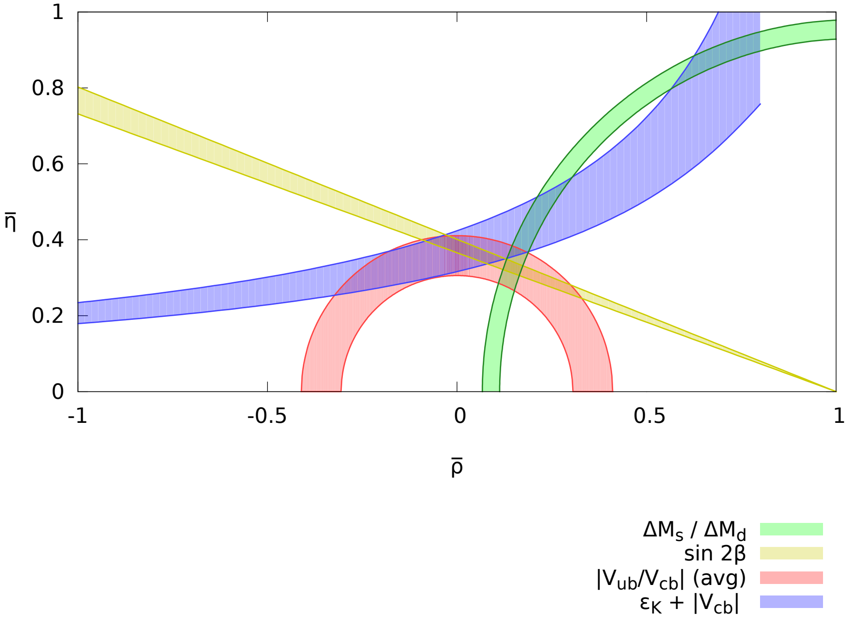
A new computer algebra system for (lattice) perturbation theory and the RBC/UKQCD heavy quark physics program

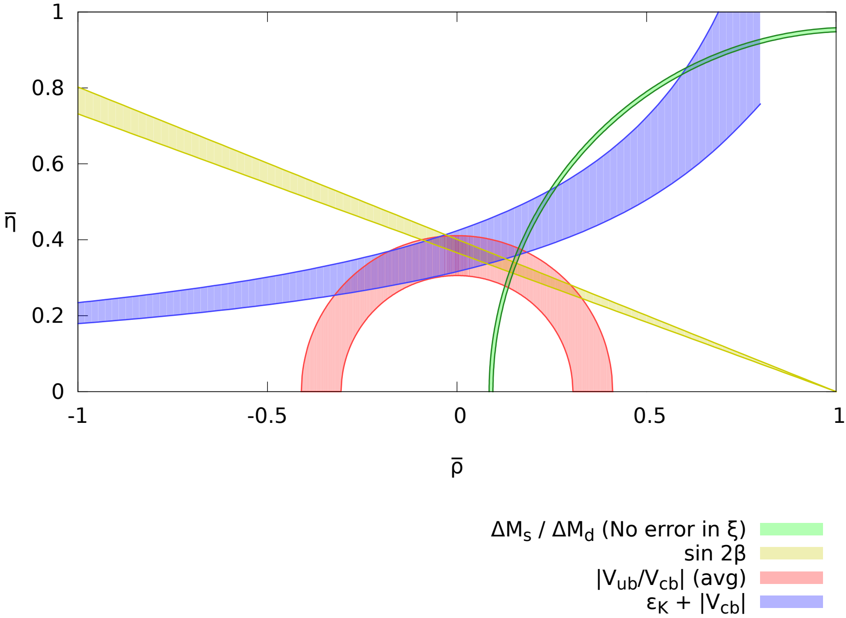
Christoph Lehner
BNL

Lattice 2013 – July 30 2013

(PDG, CKMfitter, latticeaverages.org, HFAG)

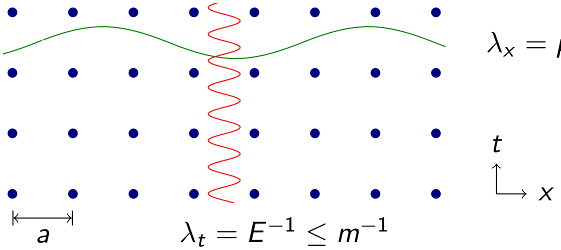


(PDG, CKMfitter, latticeaverages.org, HFAG)



Simulation of heavy quarks on the lattice

- Problem: Heavy mesons "fall through the lattice"



- Mesons with mass m, momentum p, and energy $E=\sqrt{m^2+p^2}$
- Typical scales:
  $a^{-1} \approx 2$ GeV, $m_D \approx 2$ GeV, $m_B \approx 5$ GeV $\Rightarrow am \geq 1$;
  $m_\pi \approx 0.2$ GeV, $L = 32a \Rightarrow m_\pi L \approx 3$

Relativistic heavy quarks

(El-Khadra et al. 1997)
(S. Aoki et al. 2003) (Christ et al. 2006)

- Columbia formulation:

$$S = \sum_x \overline{Q}(x)\left((\gamma_0 D_0 - \tfrac{1}{2} D_0^2) + \zeta \sum_{i=1}^{3} (\gamma_i D_i - \tfrac{1}{2} D_i^2) + m_0 + c_P \sum_{\mu,\nu=0}^{3} \tfrac{i}{4} \sigma_{\mu\nu} F_{\mu\nu}(x) \right) Q(x)$$

- Tune coefficients of dimension 4 and 5 operators to remove $|a\vec{p}|$, $(am)^n$, $|a\vec{p}|(am)^n$ errors in on-shell quantities:

$$m_0,\ \zeta,\ c_P$$

- Matching of on-shell matrix elements of, e.g., HL operators requires $Q'(x) = Q(x) + d_1 \sum_{i=1,2,3} \gamma_i D_i Q(x)$ with parameter d_1.

Relativistic heavy quarks

(El-Khadra et al. 1997)

(S. Aoki et al. 2003) (Christ et al. 2006)

RBC/UKQCD NP and perturbative tuning: PRD 86, 116003 (2012)

- Columbia formulation:

$$S = \sum_x \overline{Q}(x)\left((\gamma_0 D_0 - \tfrac{1}{2}D_0^2) + \zeta \sum_{i=1}^{3}(\gamma_i D_i - \tfrac{1}{2}D_i^2) + m_0 + c_P \sum_{\mu,\nu=0}^{3} \tfrac{i}{4}\sigma_{\mu\nu}F_{\mu\nu}(x)\right)Q(x)$$

- Tune coefficients of dimension 4 and 5 operators to remove $|a\vec{p}|$, $(am)^n$, $|a\vec{p}|(am)^n$ errors in on-shell quantities:

$$m_0,\ \zeta,\ c_P$$

- Matching of on-shell matrix elements of, e.g., HL operators requires $Q'(x) = Q(x) + d_1 \sum_{i=1,2,3}\gamma_i D_i Q(x)$ with parameter d_1.

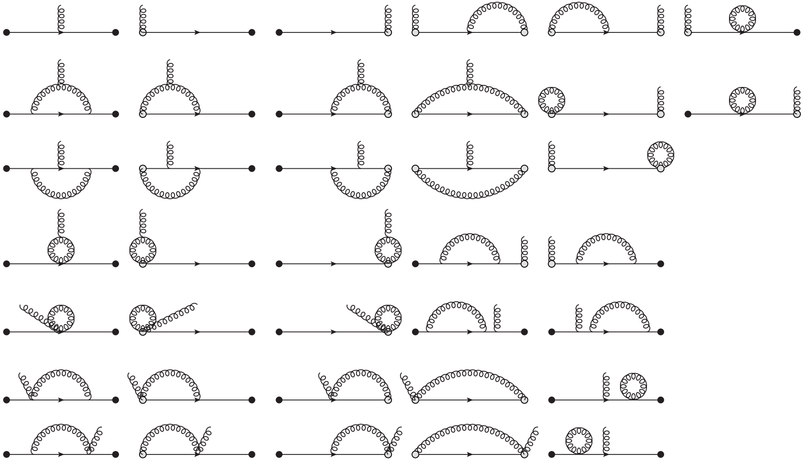
Calculations for a wide class of regulators

- The perturbative tuning of the lattice action,
- suppression of lattice artifacts,
- and matching of lattice matrix elements to, e.g., $\overline{\mathrm{MS}}$-renormalized matrix elements

require the perturbative calculation of physical observables in different regulators.

Automate such calculations in a common framework

One-loop vertex graphs (RHQ)



There are more vertices in lattice regulators and they are more complex.

Introducing PhySyHCAl

A **Phy**sics **Sy**stem based on **H**ierarchical **C**omputer **Al**gebra

- Idea: Automate lattice PT analog to continuum PT in FORM-based approaches
- Problem: Simple expansion of terms prohibitive (complex vertices) $\Rightarrow$ keep factorization (call graphs)
- FORM not suited to work on such call-graphs
- Wrote new computer algebra system (CAS) as a C++ library

On top of new CAS: unified LPT, continuum PT framework

PhySyHCAl, http://physyhcal.lhnr.de/

libcas (C++)

- Pattern matching and replacement
- Function map
- General summation indices
- Hooks

libqft (C++)

- Wick contractions
- Lattice / continuum actions
- Derivatives
- Reduction to master integrals
- Field rotations / operators
- Code generator

libint (C++)

- Numerical evaluation of algebraic expressions
- Various integrators
- Linear combinations of integrands

libint-python (Python)

- Evaluate perturbative amplitudes with error propagation
- Automated n-loop matching

Different classes of regulators

- 4-dimensional momentum space: Wilson, RHQ, Gauge, Continuum (*d*-dimensional)

- 4-dimensional momentum space plus one extra dimension: Domain Wall Fermions (algebraic or numerical treatment of 5d)

- 3-dimensional momentum space plus temporal dimension in position space: Schroedinger Functional implementations

Different classes of regulators

- 4-dimensional momentum space: Wilson, RHQ, Gauge, Continuum (d-dimensional)
- 4-dimensional momentum space plus one extra dimension: Domain Wall Fermions (algebraic or numerical treatment of 5d)
- 3-dimensional momentum space plus temporal dimension in position space: Schroedinger Functional implementations

Thanks to A. Shindler

Different classes of regulators

- 4-dimensional momentum space: Wilson, RHQ, Gauge, Continuum (*d*-dimensional)

```
// Relativistic Heavy Quark Action
from( " sum(x)*Qb(x)*( "
      " + sum(i1,4)*zeta(i1)*Ngamma(i1)*aD(i1,x) + am0 "
      " - sum(i1,4)*r(i1)*aD(i1,i1,x)/2 "
      " + sum(i1,4)*sum(i2,4)*i_*cp(i1,i2)/4"
         "*Nsigma(i1,i2)*aF(i1,i2,x) "
      " )*Q(x) " );
```

- 4-[illegible] ension: Do[illegible] ment of 5d)

- 3-dimensional momentum space plus temporal dimension in position space: Schroedinger Functional implementations

Different classes of regulators

- 4-din … e,
  Cont

```
// Domain Wall Action
from( " sum(x)*sum(s,1,N5)*sum(t,1,N5)*Qb(s,x)*( "
     "+ ( sum(i1,4)*Ngamma(i1)*aD(i1,x) "
     "    - 1/2*sum(i1,4)*aD(i1,i1,x) - aM5Q )*delta(s,t)"
     "+ ( delta(s,t) + d5Q*("
     "          - PL*delta(s+1,t) - PR*delta(s-1,t)) )"
     "+ (  PL*delta(s,N5)*delta(t,1) "
     "   + PR*delta(s,1)*delta(N5,t))*am0Q"
     " )*Q(t,x) " );
```

- 4-dimensional momentum space plus one extra dimension: Domain Wall Fermions (algebraic or numerical treatment of 5d)

- 3-dimensional momentum space plus temporal dimension in position space: Schroedinger Functional implementations

Different classes of regulators

- 4-dimensional momentum space: Wilson, RHQ, Gauge, Continuum (*d*-dimensional)

```
// Schroedinger Functional (Wilson)
from( " sum(x,space3)*sum(x0,time)*Qb(pos(x0,x))*( "
     // Wilson
     " + sum(i1,4)*Ngamma(i1)*aD(i1,pos(x0,x)) + am0 "
     " - sum(i1,4)*aD(i1,i1,pos(x0,x))/2 "
     // \delta D_v
     " + sum(i1,4)*sum(i2,4)*i_*csw/4*Nsigma(i1,i2)"
     "                         *aF(i1,i2,pos(x0,x))"
     // \delta D_b
     " + (ct - 1)*("
     "   delta(x0,1) + delta(x0,T-1)"
     "   )"
     " )*Q(pos(x0,x)) " );
```

- 4-dim ... ion: Dom ... ent of 5d)

- 3-dimensional momentum space plus temporal dimension in position space: Schroedinger Functional implementations

Example: *n*-loop RHQ self-energy

```
#include "libqft.hh"
using namespace cas; using namespace std; using namespace qft;
#define n 1
int main(int argc, char* argv[]) {
  Context c;

  ActionRHQ rhq(&c,"Q",2*n);
  ActionGAUGE gauge(&c,2*n);

  Expression* iprop =
  Series::inverted_even(&c,
    (Wick(&c) << rhq << gauge).contract(
      "sum(q,mom)*QmomT(p)*QbmomT(q)",2*n),
   "gs",2*n);

  c.hints << "p" | "C-type" = "V4";

  IntegratorFactory() <<
  PerturbativeIntegrands("RHQiprop", &c, iprop, n) >>
  "integrators/rhq";

  delete iprop;
  return 0;
}
```

Example: *n*-loop cont. quark self-energy

```
#include "libqft.hh"
using namespace cas; using namespace std; using namespace qft;
#define n 1
int main(int argc, char* argv[]) {
  Context c;

  ActionCONTQ contq(&c,"QC");
  ActionCONTG contg(&c);

  Expression* ipropCON =
  Series::inverted_even(&c,
    (Wick(&c) << contq << contg).contract(
      "Zqc*sum(q,mom)*QCmomT(p)*QCbmomT(q)",2*n),
   "gs",2*n);

  c.commuting << "Zqc" << "Zmc" << "ampQR";
  ipropCON = ipropCON->replaceExpression("ampQC", "ampQR*Zmc");

  DimReg(&c, ipropCON).expandZfactor("Zmc",n).expandZfactor("Zqc",n).
   expandInCoupling(n).combineFermionLines().prepareTensorIntegrals().
   tensorToScalarIntegrals().scalarToMasterIntegrals().expandInEpsilon(0).
   apply(insertZFactors).masterIntegralsToFeynmanParameterIntegrals();

  c.hints << "p" | "C-type" = "V4";
  c.hints << "ampQR" | "C-type" = "double";

  IntegratorFactory() <<
  PerturbativeIntegrands("CONiprop", &c, ipropCON, n) >>
  "integrators/con_iprop";

  delete ipropCON;
  return 0;
}
```

First applications

Reproducing the literature:

- 1-loop DWF renormalization (S. Aoki et al. 2003)
- 1-loop tuning of Tsukuba-RHQ (S. Aoki et al. 2004)
- 1-loop matching and O(a)-improvement of Tsukuba-RHQ–DWF (axial-)vectors (Yamada et al. 2005)

RBC/UKQCD heavy quark physics:

- 1-loop tuning of Columbia-RHQ (PRD 86, 116003 (2012))
- 1-loop matching and O(a)-improvement of Columbia-RHQ–DWF (axial-)vectors (to be published soon)

All new PhySyHCAl results are calculated in at least two covariant gauges and also given for two methods of tadpole resummation.

Performance

Timing on single thread on 2.6 GHz Intel Core i7

Example shown before: RHQ 1-loop propagator

- Algebraic manipulations to produce integrands: 4.3s on first run, 0.2s on further runs with same action
- Numerical integration of 7 diagrams to 10^{-5} relative accuracy: 2.4s

Example shown before: 1-loop cont. quark propagator

- Algebraic manipulations: 1.6s on first run, 0.9s on further runs with same action
- Numerical integration: negligible (all master integrals known)

Work in progress

- Parallelization of algebraic manipulations
- Parallelization of integration routines

Conclusion and Outlook

A **Phy**sics **Sy**stem based on **H**ierarchical **C**omputer **Al**gebra

- Tutorials, interactive online playground, and documentation at http://physyhcal.lhnr.de/
- First introduced in CL PoS LATTICE 2012, more details published soon
- First applications within RBC/UKQCD heavy quark physics program
- Wide range of regulators: dimensional regularization as well as Wilson-type, DWF-type, and SF-type lattice regulators

Outlook:

- Two-loop matching
- Tuning of highly-improved actions
- Implementation of further regulators (and, e.g., Gradient Flow)