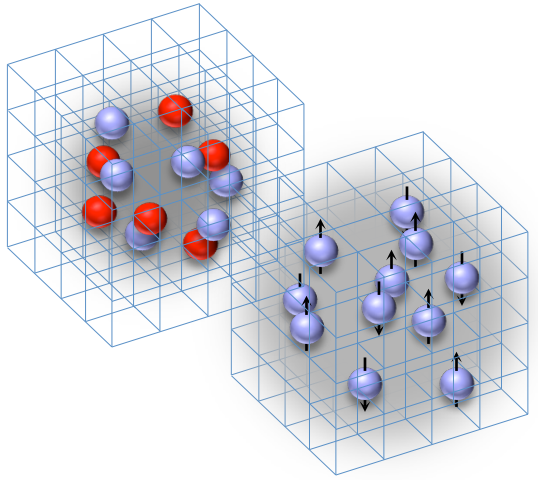
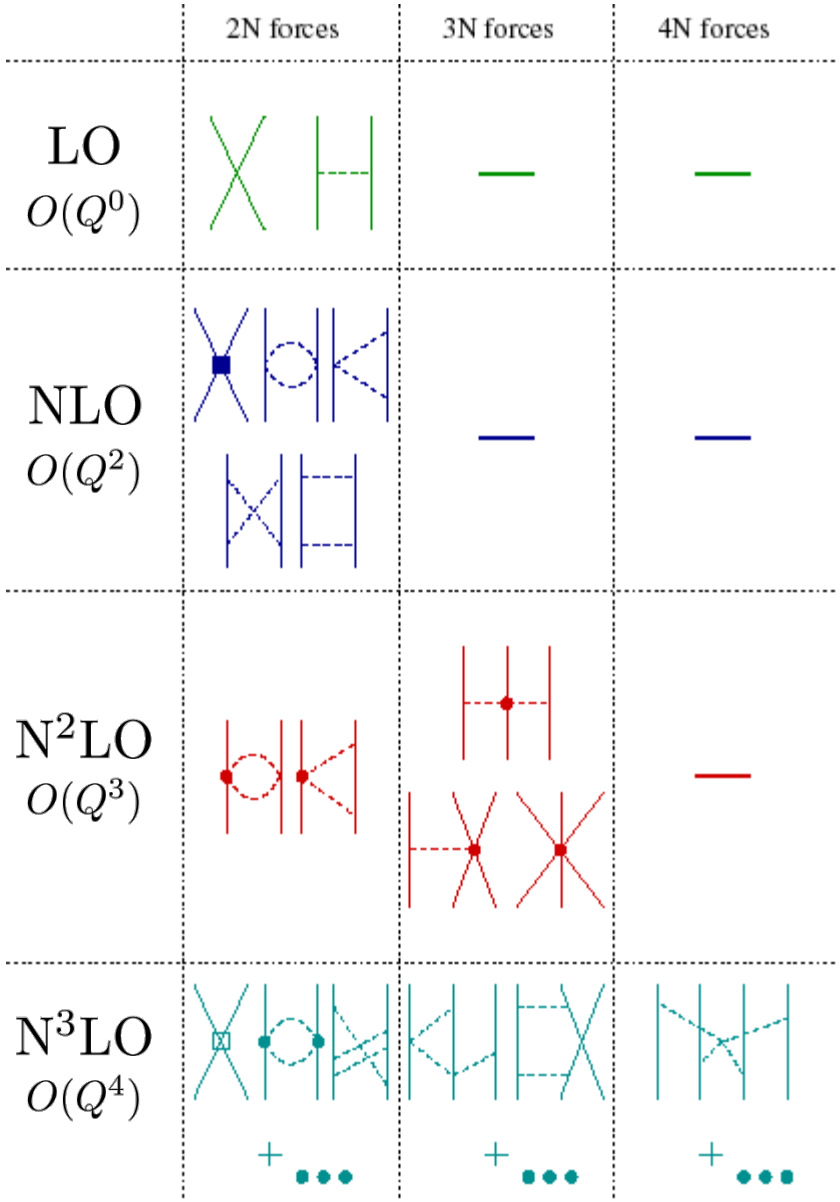
Lecture 25: Chiral EFT on the Lattice

Dean Lee
Facility for Rare Isotope Beams
Michigan State University
Nuclear Lattice EFT Collaboration
July 29, 2019



From Quarks and Gluons to Nuclear Forces and Structure


2N forces
3N forces
4N forces
LO
$O(Q^0)$
NLO
$O(Q^2)$
N^{2}LO
$O(Q^3)$
N^{3}LO
$O(Q^4)$

Chiral effective field theory on the lattice

We first give an overview of chiral effective field theory formalism on the lattice. We then simplify to a simple leading-order action and go into greater depth. We define the smeared annhilation and creation operators.

$$a_{i,j}^{s_{\rm NL}}(\mathbf{n}) = a_{i,j}(\mathbf{n}) + s_{\rm NL} \sum_{|\mathbf{n}'|=1} a_{i,j}(\mathbf{n}+\mathbf{n}')$$

$$a_{i,j}^{s_{\rm NL}\dagger}(\mathbf{n}) = a_{i,j}^{\dagger}(\mathbf{n}) + s_{\rm NL} \sum_{|\mathbf{n}'|=1} a_{i,j}^{\dagger}(\mathbf{n}+\mathbf{n}')$$

Next we form bilinear functions of the annihilation operators with spin and isospin quantum numbers S, S_z, I, I_z.

$$[a(\mathbf{n})a(\mathbf{n}')]^{s_{\rm NL}}_{S,S_z,I,I_z} = \sum_{i,j,i',j'} a_{i,j}^{s_{\rm NL}}(\mathbf{n}) M_{ii'}(S,S_z) M_{jj'}(I,I_z) a_{i',j'}^{s_{\rm NL}}(\mathbf{n}')$$

Li, Elhatisari, Epelbaum, D.L., Lu, Meißner, PRC 98, 044002 (2018)

We introduce orbital angular momentum using solid spherical harmonics

$$R_{L,L_z}(\mathbf{r}) = \sqrt{\frac{4\pi}{2L+1}} r^L Y_{L,L_z}(\theta,\phi),$$

that are written as functions of the lattice derivatives on one of the annihilation operators

$$P^{2M,s_{\rm NL}}_{S,S_z,L,L_z,I,I_z}(\mathbf{n}) = [a(\mathbf{n})\nabla^{2M}_{1/2} R^*_{L,L_z}(\nabla) a(\mathbf{n})]^{s_{\rm NL}}_{S,S_z,I,I_z}$$

We then project onto the selected spin and orbital angular momentum using Clebsch-Gordan coefficients

$$O^{2M,s_{NL}}_{S,L,J,J_z,I,I_z}(\mathbf{n}) = \sum_{S_z,L_z} \langle SS_z LL_z | JJ_z \rangle P^{2M,s_{NL}}_{S,S_z,L,L_z,I,I_z}(\mathbf{n})$$

We use these structures to construct the short-range interactions. We also specifically construct a short-range interaction that is Wigner SU(4) symmetric at leading-order with a tunable local regulator.

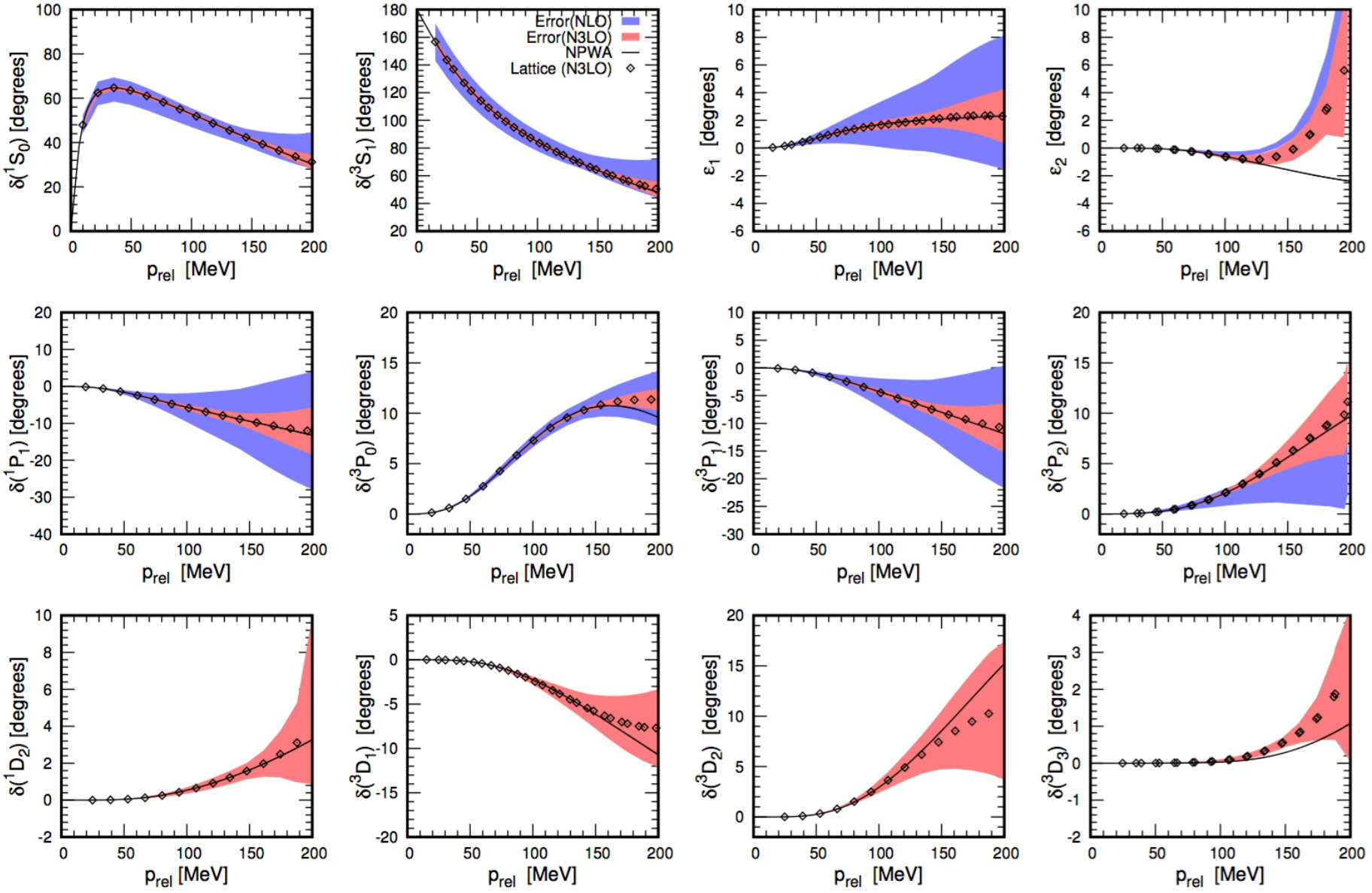
$$\frac{C_0}{2} : \sum_{\mathbf{n}',\mathbf{n},\mathbf{n}''} \sum_{i',j'} a^{s_{\rm NL}\dagger}_{i',j'}(\mathbf{n}') a^{s_{\rm NL}}_{i',j'}(\mathbf{n}') f_{s_{\rm L}}(\mathbf{n}'-\mathbf{n}) f_{s_{\rm L}}(\mathbf{n}-\mathbf{n}'') \sum_{i'',j''} a^{s_{\rm NL}\dagger}_{i'',j''}(\mathbf{n}'') a^{s_{\rm NL}}_{i'',j''}(\mathbf{n}'') :$$

Elhatisari, *et al.*, PRL 119, 222505 (2017); Elhatisari, *et al.*, PRL 117, 132501 (2016)

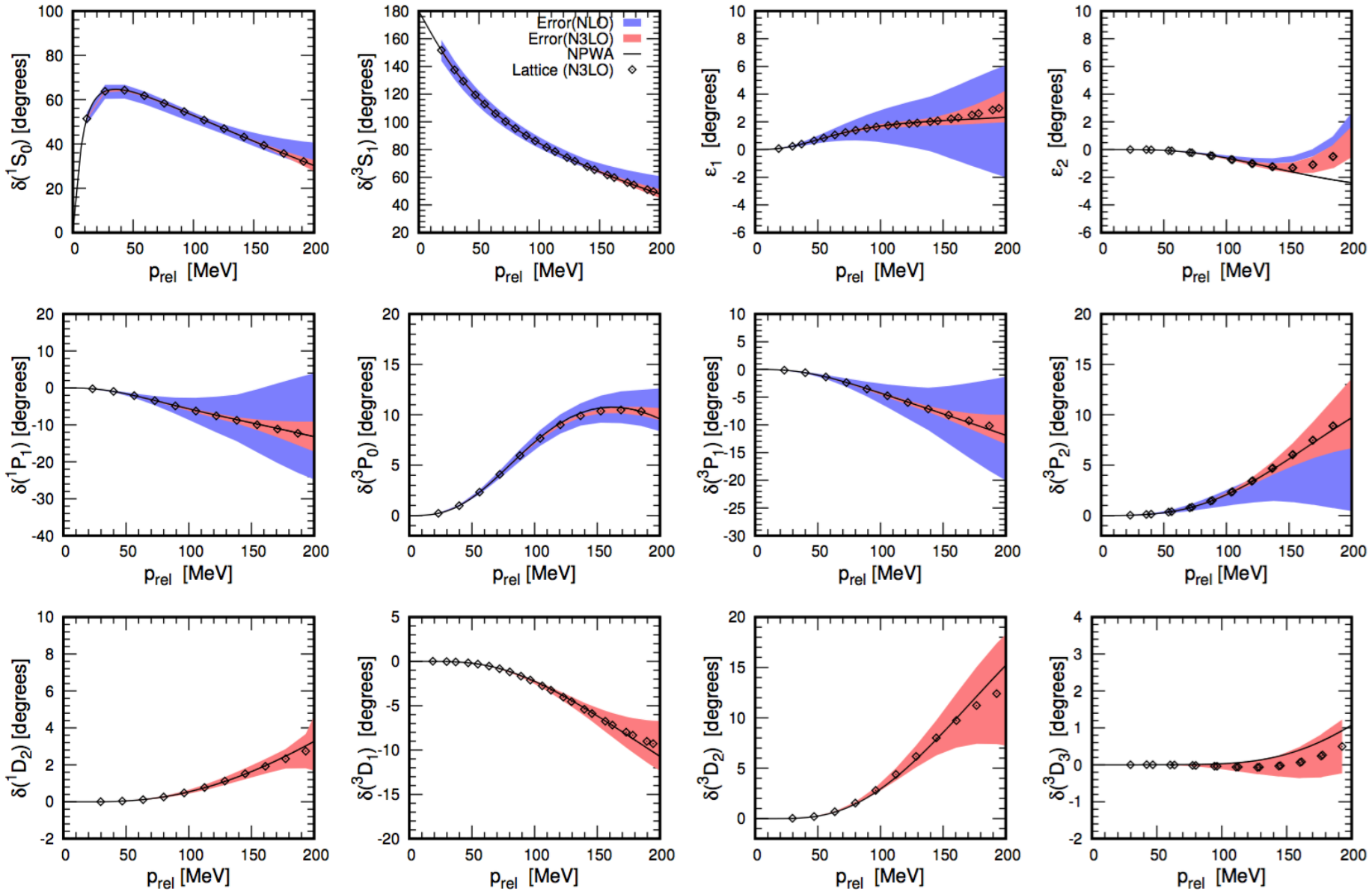
For the long-range interactions, we include the one-pion exchange potential and the two-pion exchange potential for smaller lattice spacings. For coarser lattice spacings, the difference between the two-pion exchange potential and short-range interactions are not resolved. We also include the Coulomb interaction between protons and isospin-breaking interactions.

Li, Elhatisari, Epelbaum, D.L., Lu, Meißner, PRC 98, 044002 (2018)

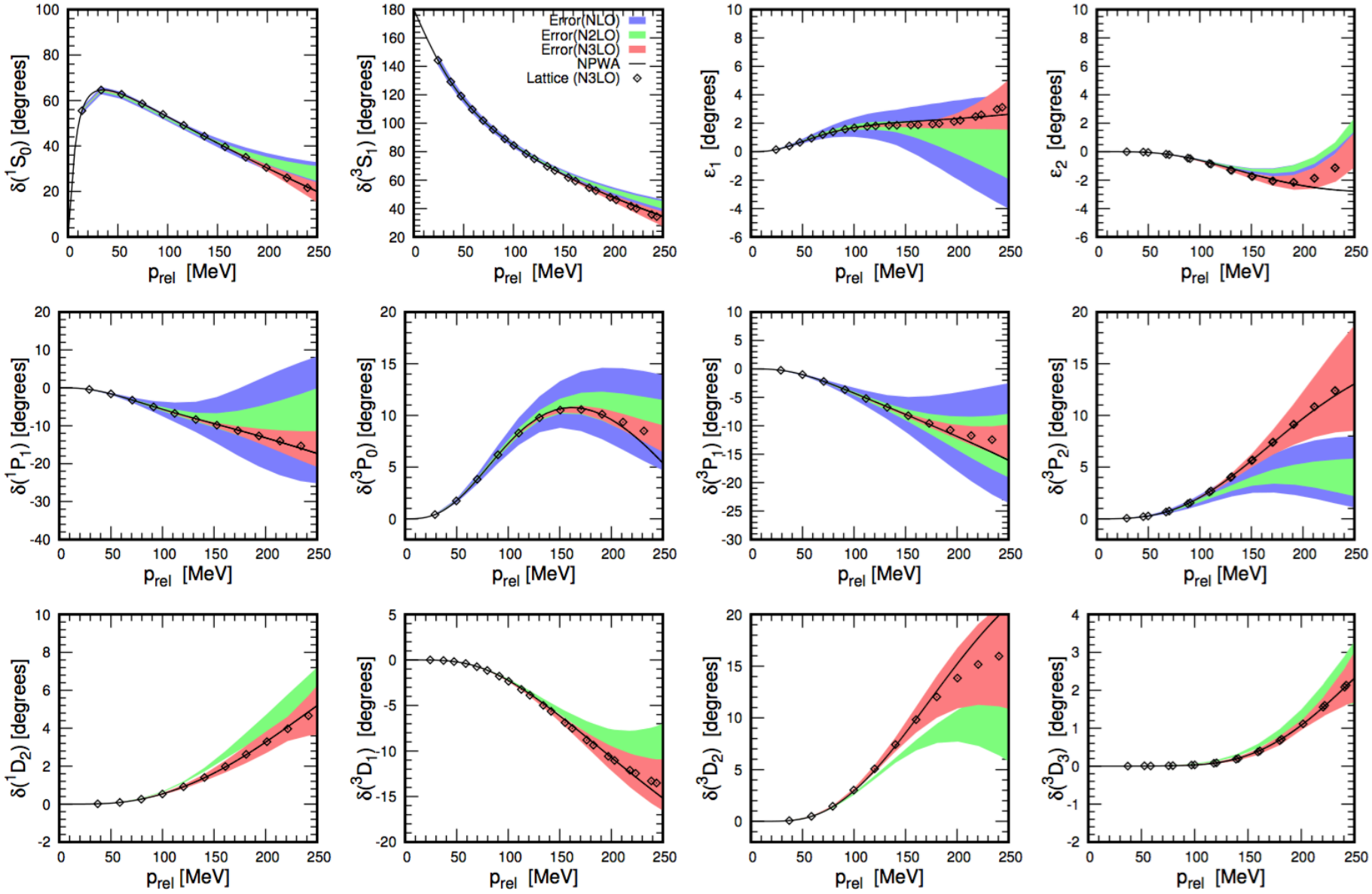
$a = 1.973\,\mathrm{fm}$



$a = 1.644\,\mathrm{fm}$



$a = 1.315\,\mathrm{fm}$



$a = 0.987\,\mathrm{fm}$

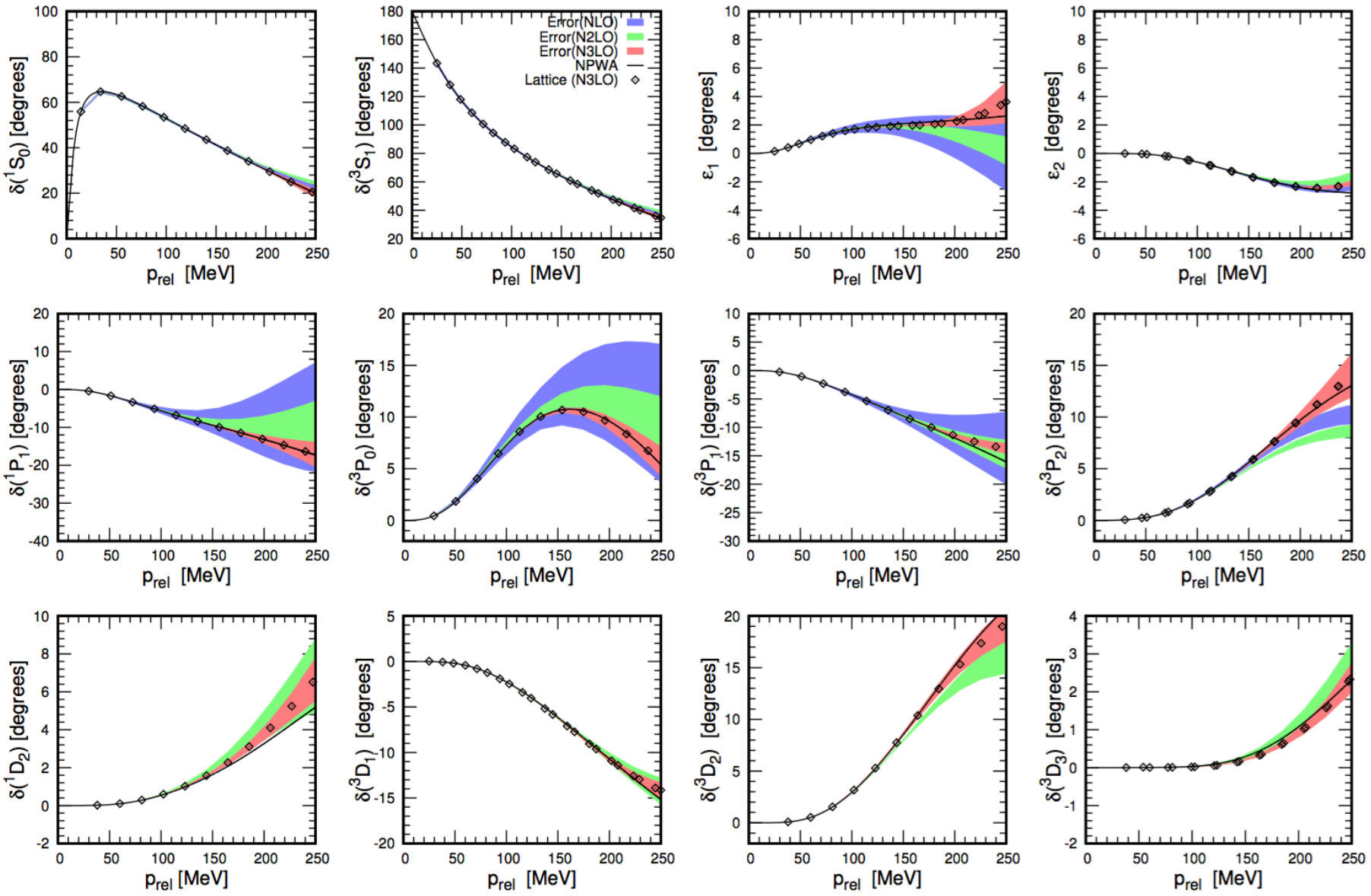
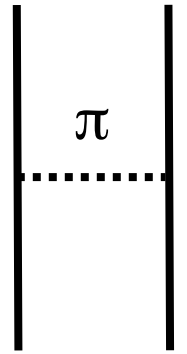


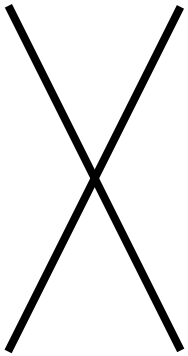
TABLE II. The deuteron properties and S-wave parameters calculated with the full NN interaction up to chiral order $O(Q^4)$ using $a = 0.99$ fm. The error bars we quote in this table indicate uncertainties from the fitting procedure only.

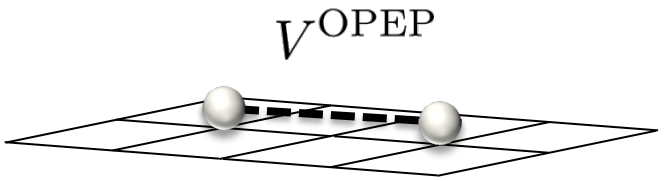
	LO	NLO	N^{2}LO	N^{3}LO	Empirical
E_d (MeV)	2.2246 ± 0.0002	2.224575 ± 0.000016	2.224575 ± 0.000025	2.224575 ± 0.000011	2.224575(9)[24]
$A_s(\mathrm{fm}^{-1/2})$	0.8662 ± 0.0007	0.8772 ± 0.0003	0.8777 ± 0.0004	0.8785 ± 0.0004	0.8846(9)[25]
η	0.0212 ± 0.0000	0.0258 ± 0.0001	0.0257 ± 0.0002	0.0254 ± 0.0001	0.0256(4) [26]
$Q_d(\mathrm{fm}^2)$	0.2134 ± 0.00000	0.2641 ± 0.0016	0.2623 ± 0.0023	0.2597 ± 0.0013	0.2859(3) [27]
r_d (fm)	1.9660 ± 0.0001	1.9548 ± 0.0005	1.9555 ± 0.0008	1.9545 ± 0.0005	1.97535(85) [28]
$a_{^3S_1}$	5.461 ± 0.000	5.415 ± 0.001	5.421 ± 0.002	5.417 ± 0.001	5.424(4) [29]
$r_{^3S_1}$	1.831 ± 0.0003	1.759 ± 0.002	1.760 ± 0.003	1.758 ± 0.002	1.759(5)[29]
$a_{^1S_0}$	-23.8 ± 0.1	-23.69 ± 0.05	-23.8 ± 0.2	-23.678 ± 0.038	−23.748(10)[29]
$r_{^1S_0}$	2.666 ± 0.001	2.647 ± 0.003	2.69 ± 0.02	2.647 ± 0.004	2.75(5) [29]

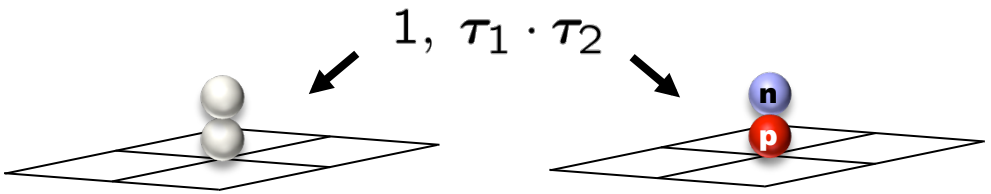
Li, Elhatisari, Epelbaum, D.L., Lu, Meißner, PRC 98, 044002 (2018)

Leading order interactions









We first consider the leading order chiral EFT interaction on the lattice in the Grassmann path integral formalism

$$\mathcal{Z} = \int DcDc^* \exp\left[-S\left(c^*, c\right)\right]$$

$$S(c^*, c) = S_{\text{free}}(c^*, c) + S_{\text{int}}(c^*, c)$$

$$S_{\text{free}}(c^*, c) = \sum_{\vec{n}, n_t, i} \boxed{c_i^*(\vec{n}, n_t)\left[c_i(\vec{n}, n_t + 1) - c_i(\vec{n}, n_t)\right]} \rightarrow c_i^* \frac{\partial c_i}{\partial t}$$

$$-\frac{\alpha_t}{2m} \sum_{\vec{n}, n_t, i} \sum_{l=1,2,3} \boxed{c_i^*(\vec{n}, n_t)\left[c_i(\vec{n} + \hat{l}, n_t) - 2c_i(\vec{n}, n_t) + c_i(\vec{n} - \hat{l}, n_t)\right]}$$

$$\rightarrow c_i^* \frac{\partial^2 c_i}{\partial x_l^2}$$

It is convenient to view c without indices as a column vector and c^* without indices as a row vector

$$c^* = \left[c_{\uparrow,p}^* c_{\downarrow,p}^* c_{\uparrow,n}^* c_{\downarrow,n}^*\right] \qquad c = \begin{bmatrix} c_{\uparrow,p} \\ c_{\downarrow,p} \\ c_{\uparrow,n} \\ c_{\downarrow,n} \end{bmatrix}$$

The first interaction we consider is the short-range interaction between nucleons which is independent of spin and isospin

$$S^C_{\rm int}(c^*,c) = \alpha_t \frac{C}{2} \sum_{\vec{n},n_t} \left[c^*(\vec{n},n_t)c(\vec{n},n_t)\right]^2$$

Using the auxiliary field s, we can write this interaction as

$$\exp\left[-S^C_{\rm int}(c^*,c)\right] = \int Ds \ \exp\left[-S_{ss}(s) - S_s(c^*,c,s)\right]$$

where

$$S_{ss}(s) = \frac{1}{2} \sum_{\vec{n},n_t} s^2(\vec{n},n_t)$$

$$S_s(c^*,c,s) = \sqrt{-C\alpha_t} \sum_{\vec{n},n_t} s(\vec{n},n_t)c^*(\vec{n},n_t)c(\vec{n},n_t)$$

Next we have the short-range interaction dependent on isospin

$$S_{\rm int}^{C'}(c^*,c) = \alpha_t \frac{C'}{2} \sum_{\vec{n},n_t,I} \left[c^*(\vec{n},n_t)\tau_I c(\vec{n},n_t)\right]^2$$

where we are using the Pauli matrices in isospin space

$$\tau_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}_{\rm isospin} \qquad \tau_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}_{\rm isospin} \qquad \tau_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}_{\rm isospin}$$

In terms of three auxiliary fields s_I, we can write the interaction as

$$\exp\left[-S_{\rm int}^{C'}(c^*,c)\right] = \int \prod_I Ds_I \exp\left[-S_{s_I s_I}(s_I) - S_{s_I}(c^*,c,s_I)\right]$$

$$S_{s_I s_I}(s_I) = \tfrac{1}{2} \sum_{\vec{n},n_t,I} s_I^2(\vec{n},n_t)$$

$$S_{s_I}(c^*,c,s_I) = \sqrt{-C'\alpha_t} \sum_{\vec{n},n_t,I} s_I(\vec{n},n_t) c^*(\vec{n},n_t)\tau_I c(\vec{n},n_t)$$

The remaining interaction is the one pion exchange potential (OPEP). We will not include time derivatives in the free pion action, and hence the the pion is not treated as a dynamical field. Instead it resembles an auxiliary field that produces an exchange potential for the nucleons.

$$\exp\left[-S_{\rm int}^{\rm OPEP}(c^*,c)\right] = \int \prod_I D\pi_I \exp\left[-S_{\pi_I\pi_I}(\pi_I) - S_{\pi_I}(c^*,c,\pi_I)\right]$$

$$S_{\pi_I\pi_I}(\pi_I) = \tfrac{1}{2}\alpha_t m_\pi^2 \sum_{\vec{n},n_t,I} \pi_I^2(\vec{n},n_t)$$
$$-\tfrac{1}{2}\alpha_t \sum_{\vec{n},n_t,I,\hat{l}} \pi_I(\vec{n},n_t)\left[\pi_I(\vec{n}+\hat{l},n_t) - 2\pi_I(\vec{n},n_t) + \pi_I(\vec{n}-\hat{l},n_t)\right]$$

The pion coupling to the nucleon is

$$S_{\pi_I}(c^*, c, \pi_I) = \frac{g_A \alpha_t}{2 f_\pi} \sum_{\vec{n}, n_t, l, I} \Delta_k \pi_I(\vec{n}, n_t) c^*(\vec{n}, n_t) \sigma_k \tau_I c(\vec{n}, n_t)$$

where g_A is the axial charge, f_π is the pion decay constant, and we have used the Pauli spin matrices

$$\sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}_{\text{spin}} \qquad \sigma_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}_{\text{spin}} \qquad \sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}_{\text{spin}}$$

And the gradient of the pion field is

$$\Delta_l \pi_I(\vec{n}, n_t) = \frac{1}{2} \left[\pi_I(\vec{n} + \hat{l}, n_t) - \pi_I(\vec{n} - \hat{l}, n_t) \right]$$

We can reexpress everything in terms of normal-ordered transfer matrix operators

$$\mathcal{Z} = \int Ds \prod_I (Ds_I D\pi_I)$$
$$\exp\left[-S_{ss}(s) - S_{s_I s_I}(s_I) - S_{\pi_I \pi_I}(\pi_I)\right] \mathrm{Tr}\left\{M^{(L_t-1)} \dots M^{(0)}\right\}$$

where

$$M^{(n_t)} =: \exp\left[-H^{(n_t)}(a^\dagger, a, s, s_I, \pi_I)\alpha_t\right]$$

$$H^{(n_t)}(a^\dagger, a, s, s_I, \pi_I)\alpha_t = H_{\mathrm{free}}\alpha_t + S_s^{(n_t)}(a^\dagger, a, s) + S_{s_I}^{(n_t)}(a^\dagger, a, s_I) + S_{\pi_I}^{(n_t)}(a^\dagger, a, \pi_I)$$

with

$$S_s^{(n_t)}(a^\dagger, a, s) = \sqrt{-C\alpha_t} \sum_{\vec{n}} s(\vec{n}, n_t) a^\dagger(\vec{n}) a(\vec{n})$$

$$S_{s_I}^{(n_t)}(a^\dagger, a, s_I) = \sqrt{-C'\alpha_t} \sum_{\vec{n}, I} s_I(\vec{n}, n_t) a^\dagger(\vec{n}) \tau_I a(\vec{n})$$

$$S_{\pi_I}^{(n_t)}(a^\dagger, a, \pi_I) = \frac{g_A \alpha_t}{2 f_\pi} \sum_{\vec{n}, k, I} \Delta_k \pi_I(\vec{n}, n_t) a^\dagger(\vec{n}) \sigma_k \tau_I a(\vec{n})$$

For the auxiliary-field projection Monte Carlo calculation we compute

$$Z(L_t) = \int Ds \prod_I (Ds_I D\pi_I)$$
$$= \exp\left[-S_{ss}(s) - S_{s_I s_I}(s_I) - S_{\pi_I \pi_I}(\pi_I)\right] Z(s, s_I, \pi_I, L_t)$$

where

$$Z(s, s_I, \pi_I, L_t) = \det \mathbf{Z}(s, s_I, \pi_I, L_t)$$

and the matrix of single nucleon amplitudes is

$$\mathbf{Z}_{i,j}(s, s_I, \pi_I, L_t) = \langle f_i | M^{(L_t - 1)} \dots M^{(0)} | f_j \rangle$$

We store the set of vectors for each single-particle initial state at each time step

$$|v_j^{(n_t)}\rangle = M^{(n_t-1)} \dots M^{(0)} |f_j\rangle$$

as well as the dual vectors at each time step propagating in the reverse temporal direction

$$\langle v_i^{(n_t)}| = \langle f_i| M^{(L_t-1)} \dots M^{(n_t)}$$

These are useful in computing the update to an auxiliary field value at time step n_t, using the following relations:

$$Z(s, s_I, \pi_I, L_t) = \det \mathbf{Z}(s, s_I, \pi_I, L_t)$$

$$\mathbf{Z}_{i,j}(s, s_I, \pi_I, L_t) = \langle v_i^{(n_t+1)}| M^{(n_t)}(s, s_I, \pi_I) |v_j^{(n_t)}\rangle$$

It is convenient to redefine the normalization of $\boldsymbol{\pi}_I$

$$\pi'_I(\vec{n}, n_t) = \pi_I(\vec{n}, n_t)\sqrt{\alpha_t(m_\pi^2 + 6)}$$

So that

$$S_{\pi_I\pi_I}(\pi_I) = \tfrac{1}{2}\alpha_t m_\pi^2 \textstyle\sum_{\vec{n},n_t,I} \pi_I^2(\vec{n}, n_t)$$
$$-\tfrac{1}{2}\alpha_t \textstyle\sum_{\vec{n},n_t,I,\hat{l}} \pi_I(\vec{n}, n_t)\left[\pi_I(\vec{n}+\hat{l}, n_t) - 2\pi_I(\vec{n}, n_t) + \pi_I(\vec{n}-\hat{l}, n_t)\right]$$

$$= \tfrac{1}{2}\textstyle\sum_{\vec{n},n_t,I} \pi'^2_I(\vec{n}, n_t) - \frac{\alpha_t}{2q_\pi}\sum_{\vec{n},n_t,I,\hat{l}} \pi'_I(\vec{n}, n_t)\left[\pi'_I(\vec{n}+\hat{l}, n_t) + \pi'_I(\vec{n}-\hat{l}, n_t)\right]$$

and

$$S_{\pi_I}^{(n_t)}(a^\dagger, a, \pi_I) = \frac{g_A\alpha_t}{2f_\pi\sqrt{q_\pi}}\textstyle\sum_{\vec{n},k,I} \Delta_k\pi'_I(\vec{n}, n_t)a^\dagger(\vec{n})\sigma_k\tau_I a(\vec{n})$$

with

$$q_\pi = \alpha_t(m_\pi^2 + 6)$$

```
SUBROUTINE getzvecs(s,sI,zvecs,zvecsinit,nt2,nt1, &
     pion,ztau2x2,num)

  IMPLICIT INTEGER(i-n)
  IMPLICIT DOUBLE PRECISION(a-h,o-y)
  IMPLICIT COMPLEX*16(z)
  INCLUDE "input.f90"

  DIMENSION s(0:L-1,0:L-1,0:L-1,0:Lt-1)
  DIMENSION sI(0:L-1,0:L-1,0:L-1,0:Lt-1,1:3)
  DIMENSION zvecs(0:L-1,0:L-1,0:L-1,0:Lt,0:1,0:1,0:num-1)
  DIMENSION zvecsinit(0:L-1,0:L-1,0:L-1,0:1,0:1,0:num-1)
  DIMENSION pion(0:L-1,0:L-1,0:L-1,0:Lt-1,1:3)
  DIMENSION zpi2x2(0:1,0:1)
  DIMENSION zsI2x2(0:1,0:1)
  DIMENSION zsSI2x2(0:1,0:1,0:1,0:1)
  DIMENSION ztau2x2(0:1,0:1,0:3)

  !   nt2 > nt1

  INCLUDE "improve.f90"

  DO ni = 0,1; DO ns = 0,1
     DO nz = 0,L-1; DO ny = 0,L-1; DO nx = 0,L-1
        DO npart = 0,num-1
           zvecs(nx,ny,nz,nt1,ns,ni,npart) = zvecsinit(nx,ny,nz,ns,ni,npart)
        END DO
     END DO; END DO; END DO
  END DO; END DO
```

```
DO nt = nt1+1, nt2
   DO np = 0,num-1
      DO nz = 0,L-1; DO ny = 0,L-1; DO nx = 0,L-1
         DO ni = 0,1; DO ns = 0,1

            zvecs(nx,ny,nz,nt,ns,ni,np) = zvecs(nx,ny,nz,nt-1,ns,ni,np) &
                 * (1.D0-6.D0*w0_N*h+CDSQRT(-c0*atovera*(1.D0,0.D0))*s(nx,ny,nz,nt-1))

            zvecs(nx,ny,nz,nt,ns,ni,np) = zvecs(nx,ny,nz,nt,ns,ni,np) &
                 + w1_N*h*zvecs(MOD(nx+1,L),ny,nz,nt-1,ns,ni,np) &
                 + w1_N*h*zvecs(MOD(nx-1+L,L),ny,nz,nt-1,ns,ni,np) &
                 + w1_N*h*zvecs(nx,MOD(ny+1,L),nz,nt-1,ns,ni,np) &
                 + w1_N*h*zvecs(nx,MOD(ny-1+L,L),nz,nt-1,ns,ni,np) &
                 + w1_N*h*zvecs(nx,ny,MOD(nz+1,L),nt-1,ns,ni,np) &
                 + w1_N*h*zvecs(nx,ny,MOD(nz-1+L,L),nt-1,ns,ni,np)

            IF (improveN >= 1) THEN
               zvecs(nx,ny,nz,nt,ns,ni,np) = zvecs(nx,ny,nz,nt,ns,ni,np) &
                    - w2_N*h*zvecs(MOD(nx+2,L),ny,nz,nt-1,ns,ni,np) &
                    - w2_N*h*zvecs(MOD(nx-2+L,L),ny,nz,nt-1,ns,ni,np) &
                    - w2_N*h*zvecs(nx,MOD(ny+2,L),nz,nt-1,ns,ni,np) &
                    - w2_N*h*zvecs(nx,MOD(ny-2+L,L),nz,nt-1,ns,ni,np) &
                    - w2_N*h*zvecs(nx,ny,MOD(nz+2,L),nt-1,ns,ni,np) &
                    - w2_N*h*zvecs(nx,ny,MOD(nz-2+L,L),nt-1,ns,ni,np)
            END IF

            IF (improveN == 2) THEN
               zvecs(nx,ny,nz,nt,ns,ni,np) = zvecs(nx,ny,nz,nt,ns,ni,np) &
                    + w3_N*h*zvecs(MOD(nx+3,L),ny,nz,nt-1,ns,ni,np) &
                    + w3_N*h*zvecs(MOD(nx-3+L,L),ny,nz,nt-1,ns,ni,np) &
                    + w3_N*h*zvecs(nx,MOD(ny+3,L),nz,nt-1,ns,ni,np) &
                    + w3_N*h*zvecs(nx,MOD(ny-3+L,L),nz,nt-1,ns,ni,np) &
                    + w3_N*h*zvecs(nx,ny,MOD(nz+3,L),nt-1,ns,ni,np) &
                    + w3_N*h*zvecs(nx,ny,MOD(nz-3+L,L),nt-1,ns,ni,np)
            END IF

         END DO; END DO
      END DO; END DO; END DO
   END DO
```

```
DO nz = 0,L-1; DO ny = 0,L-1; DO nx = 0,L-1

   DO nii = 0,1; DO ni = 0,1
      DO nss = 0,1; DO ns = 0,1
         zsSI2x2(ns,nss,ni,nii) = (0.D0,0.D0)
      END DO; END DO
   END DO; END DO

   zsI2x2(0,0) = sI(nx,ny,nz,nt-1,3)
   zsI2x2(1,1) = -sI(nx,ny,nz,nt-1,3)
   zsI2x2(0,1) = sI(nx,ny,nz,nt-1,1) - (0.D0,1.D0)*sI(nx,ny,nz,nt-1,2)
   zsI2x2(1,0) = sI(nx,ny,nz,nt-1,1) + (0.D0,1.D0)*sI(nx,ny,nz,nt-1,2)

   DO nii = 0,1; DO ni = 0,1
      DO ns = 0,1
         zsSI2x2(ns,ns,ni,nii) = zsSI2x2(ns,ns,ni,nii) &
              + (0.D0,1.D0)*CDSQRT(cI*atovera*(1.D0,0.D0))*zsI2x2(ni,nii)
      END DO
   END DO; END DO
```

```
DO iso = 1,3

   pi1 = 0.D0 &
        + o1/2.D0*pion(MOD(nx+1,L),ny,nz,nt-1,iso) &
        - o1/2.D0*pion(MOD(nx-1+L,L),ny,nz,nt-1,iso) &
        - o2/2.D0*pion(MOD(nx+2,L),ny,nz,nt-1,iso) &
        + o2/2.D0*pion(MOD(nx-2+L,L),ny,nz,nt-1,iso) &
        + o3/2.D0*pion(MOD(nx+3,L),ny,nz,nt-1,iso) &
        - o3/2.D0*pion(MOD(nx-3+L,L),ny,nz,nt-1,iso)

   pi2 = 0.D0 &
        + o1/2.D0*pion(nx,MOD(ny+1,L),nz,nt-1,iso) &
        - o1/2.D0*pion(nx,MOD(ny-1+L,L),nz,nt-1,iso) &
        - o2/2.D0*pion(nx,MOD(ny+2,L),nz,nt-1,iso) &
        + o2/2.D0*pion(nx,MOD(ny-2+L,L),nz,nt-1,iso) &
        + o3/2.D0*pion(nx,MOD(ny+3,L),nz,nt-1,iso) &
        - o3/2.D0*pion(nx,MOD(ny-3+L,L),nz,nt-1,iso)

   pi3 = 0.D0 &
        + o1/2.D0*pion(nx,ny,MOD(nz+1,L),nt-1,iso) &
        - o1/2.D0*pion(nx,ny,MOD(nz-1+L,L),nt-1,iso) &
        - o2/2.D0*pion(nx,ny,MOD(nz+2,L),nt-1,iso) &
        + o2/2.D0*pion(nx,ny,MOD(nz-2+L,L),nt-1,iso) &
        + o3/2.D0*pion(nx,ny,MOD(nz+3,L),nt-1,iso) &
        - o3/2.D0*pion(nx,ny,MOD(nz-3+L,L),nt-1,iso)

   zpi2x2(0,0) = pi3
   zpi2x2(1,1) = -pi3
   zpi2x2(0,1) = pi1 - (0.D0,1.D0)*pi2
   zpi2x2(1,0) = pi1 + (0.D0,1.D0)*pi2

   DO nii = 0,1; DO ni = 0,1
      DO nss = 0,1; DO ns = 0,1
         zsSI2x2(ns,nss,ni,nii) = zsSI2x2(ns,nss,ni,nii) &
              - gA*atovera/(2.D0*fpi*SQRT(qpi3))*zpi2x2(ns,nss) &
              * ztau2x2(ni,nii,iso)
      END DO; END DO
   END DO; END DO

END DO
```

```
        DO np = 0,num-1
           DO nii = 0,1; DO ni = 0,1
              DO nss = 0,1; DO ns = 0,1
                 zvecs(nx,ny,nz,nt,ns,ni,np) = zvecs(nx,ny,nz,nt,ns,ni,np) &
                      + zsSI2x2(ns,nss,ni,nii)*zvecs(nx,ny,nz,nt-1,nss,nii,np)
              END DO; END DO
           END DO; END DO
        END DO

     END DO; END DO; END DO

  END DO

END SUBROUTINE getzvecs
```

Hybrid Monte Carlo

We want to do efficient nonlocal updates of the auxiliary fields. Suppose we want to sample configurations according to the target probability

$$P_{\text{target}}(s) \propto \exp[-V(s)]$$

Hybrid Monte Carlo does this by introducing a conjugate momentum variable p_s for each variable s and sampling according classical molecular dynamics to the target probability

$$P_{\text{target}}[s, p_s] \propto \exp\{-H(s, p_s)\}$$
$$H(s, p_s) \equiv \tfrac{1}{2} \sum_{\vec{n}, n_t} \left[p_s(\vec{n}, n_t)\right]^2 + V(s)$$

Gottlieb, Liu, Toussaint, Renken, Sugar, Phys. Rev. D35, 2531 (1987)

Duane, Kennedy, Pendleton, Roweth, Phys. Lett. B195, 216 (1987)

We start by selecting the initial p_s configuration according to the random Gaussian distribution

$$P[p_s^0(\vec{n}, n_t)] \propto \exp\left\{-\tfrac{1}{2}\left[p_s^0(\vec{n}, n_t)\right]^2\right\}$$

Then we do classical molecular dynamics update of p_s and s which keep $H(s, p_s)$ approximately fixed. We use the leapfrog method which gives p_s a half step at the beginning and half step at the end, with full steps in between. In contrast s gets full steps at every stage.

Initial half step for p_s:

$$\tilde{p}_s^0(\vec{n}, n_t) = p_s^0(\vec{n}, n_t) - \frac{\varepsilon_{\text{step}}}{2}\left[\frac{\partial V(s)}{\partial s(\vec{n}, n_t)}\right]_{s=s^0}$$

Full steps for s and p_s:

$$s^{i+1}(\vec{n}, n_t) = s^i(\vec{n}, n_t) + \varepsilon_{\text{step}} \tilde{p}_s^i(\vec{n}, n_t)$$

$$\tilde{p}_s^{i+1}(\vec{n}, n_t) = \tilde{p}_s^i(\vec{n}, n_t) - \varepsilon_{\text{step}} \left[\frac{\partial V(s)}{\partial s(\vec{n}, n_t)} \right]_{s=s^{i+1}}$$

Cut the last step for p_s so it is a half step:

$$p_s^{N_{\text{step}}}(\vec{n}, n_t) = \tilde{p}_s^{N_{\text{step}}}(\vec{n}, n_t) + \frac{\varepsilon_{\text{step}}}{2} \left[\frac{\partial V(s)}{\partial s(\vec{n}, n_t)} \right]_{s=s^0}$$

We accept the new configurations for s and p_s if the uniform random number r between 0 and 1 satisfies

$$r < \exp\left[-H(s^{N_{\text{step}}}, p_s^{N_{\text{step}}}) + H(s^0, p_s^0) \right]$$

Then return back and repeat the steps listed above.

For our leading order auxiliary-field projection Monte Carlo calculations we have the target probability equal to

$$\exp[-V] = |Z(s, s_I, \pi_I, L_t)| \exp\left[-S_{ss}(s) - S_{s_I s_I}(s_I) - S_{\pi_I \pi_I}(\pi_I)\right]$$

where $Z(s, s_I, \pi_I, L_t)$ is the determinant of the A x A matrix of single nucleon amplitudes $\mathbf{Z}(s, s_I, \pi_I, L_t)$. The derivative of V with respect to an element of s is computed using

$$\begin{aligned}
\frac{\partial V}{\partial s(\vec{n}, n_t)} &= \frac{\partial S_{ss}(s)}{\partial s(\vec{n}, n_t)} - \frac{\partial \mathrm{Re}[\ln(\det \mathbf{Z})]}{\partial s(\vec{n}, n_t)} \\
&= \frac{\partial S_{ss}(s)}{\partial s(\vec{n}, n_t)} - \mathrm{Re}\left[\sum_{k,l} \mathbf{Z}^{-1}_{lk} \frac{\partial \mathbf{Z}_{kl}}{\partial s(\vec{n}, n_t)}\right]
\end{aligned}$$

```
! ************************************************************

SUBROUTINE dV(zvecs,zdualvecs,ztau2x2,zcorrinv,zdVall)

IMPLICIT integer(i-n)
IMPLICIT double precision(a-h,o-y)
IMPLICIT complex*16(z)
INCLUDE "input.f90"

DIMENSION zvecs(0:L-1,0:L-1,0:L-1,0:Lt,0:1,0:1,0:n_f-1)
DIMENSION zdualvecs(0:L-1,0:L-1,0:L-1,0:Lt,0:1,0:1,0:n_f-1)
DIMENSION ztau2x2(0:1,0:1,0:3)
DIMENSION zcorrinv(0:n_f-1,0:n_f-1)
DIMENSION zdVall(0:L-1,0:L-1,0:L-1,0:Lt-1,0:3,0:3)
DIMENSION zmat(0:1,0:1,0:1,0:1)

DO nt = 0,Lt-1

   DO nz = 0,L-1; DO ny = 0,L-1; DO nx = 0,L-1

      DO nii = 0,1; DO ni = 0,1; DO nss = 0,1; DO ns = 0,1
         zmat(ns,nss,ni,nii) = 0.D0
         DO np2 = 0,n_f-1; DO np1 = 0,n_f-1
            zmat(ns,nss,ni,nii) = &
                 zmat(ns,nss,ni,nii) &
                 + zdualvecs(nx,ny,nz,nt+1, &
                 ns,ni,np1) &
                 *zvecs(nx,ny,nz,nt,nss,nii,np2) &
                 *zcorrinv(np2,np1)
         END DO; END DO
      END DO; END DO; END DO; END DO

      DO iso = 0,3; DO nsvec = 0,3
         zdVall(nx,ny,nz,nt,nsvec,iso) = 0.D0
         DO nii = 0,1; DO ni = 0,1; DO nss = 0,1; DO ns = 0,1
            zdVall(nx,ny,nz,nt,nsvec,iso) = &
                 zdVall(nx,ny,nz,nt,nsvec,iso) &
                 + zmat(ns,nss,ni,nii) &
                 *ztau2x2(ns,nss,nsvec) &
                 *ztau2x2(ni,nii,iso)
         END DO; END DO; END DO; END DO
      END DO; END DO

   END DO; END DO; END DO

END DO

END SUBROUTINE dV

! ************************************************************
```

```
bose = 0.D0
DO nt = 0,Lt-1
   DO nz = 0,L-1; DO ny = 0,L-1; DO nx = 0,L-1
      bose = bose &
           + s(nx,ny,nz,nt)**2.D0/2.D0 &
           + p_s(nx,ny,nz,nt)**2.D0/2.D0
      DO iso = 1,3
         bose = bose &
              + sI(nx,ny,nz,nt,iso)**2.D0/2.D0 &
              + p_sI(nx,ny,nz,nt,iso)**2.D0/2.D0 &
              + pion(nx,ny,nz,nt,iso)**2.D0/2.D0 &
              + atovera/qpi3*pion(nx,ny,nz,nt,iso)*( &
              - w1_P*pion(MOD(nx+1,L),ny,nz,nt,iso) &
              - w1_P*pion(nx,MOD(ny+1,L),nz,nt,iso) &
              - w1_P*pion(nx,ny,MOD(nz+1,L),nt,iso) &
              + w2_P*pion(MOD(nx+2,L),ny,nz,nt,iso) &
              + w2_P*pion(nx,MOD(ny+2,L),nz,nt,iso) &
              + w2_P*pion(nx,ny,MOD(nz+2,L),nt,iso) &
              - w3_P*pion(MOD(nx+3,L),ny,nz,nt,iso) &
              - w3_P*pion(nx,MOD(ny+3,L),nz,nt,iso) &
              - w3_P*pion(nx,ny,MOD(nz+3,L),nt,iso)) &
              + p_pion(nx,ny,nz,nt,iso)**2.D0/2.D0
      END DO
   END DO; END DO; END DO
END DO
```

```
DO nt = 0,Lt-1
   DO nz = 0,L-1; DO ny = 0,L-1; DO nx = 0,L-1
      sHMC(nx,ny,nz,nt,0) = s(nx,ny,nz,nt)
      DO iso = 1,3
         sIHMC(nx,ny,nz,nt,iso,0) = &
              sI(nx,ny,nz,nt,iso)
         pionHMC(nx,ny,nz,nt,iso,0) = &
              pion(nx,ny,nz,nt,iso)
      END DO
   END DO; END DO; END DO
END DO

!     initial half step for p_sHMC, p_sIHMC, p_pionHMC

CALL getzvecs(s,sI,zvecs,zwave,Lt,0, &
     pion,ztau2x2,n_f)
CALL getzdualvecs(s,sI,zdualvecs,zdualwave, &
     Lt,0,pion,ztau2x2,n_f)

CALL getinvcorr(zvecs,zdualvecs,zldeter, &
     zcorrmatrix,zcorrinv,Lt)

aldeterabs = DBLE(zldeter)
zdeterphase = CDEXP((0.D0,1.D0)*DIMAG(zldeter))
act = bose - aldeterabs
```

```
CALL dV(zvecs,zdualvecs,ztau2x2,zcorrinv,zdVall)

DO nt = 0,Lt-1
   DO nz = 0,L-1; DO ny = 0,L-1; DO nx = 0,L-1

      DO npart1 = 0,n_f-1; DO npart2 = 0,n_f-1
         zdcorrmatrix(npart2,npart1) = 0.D0
         DO ni = 0,1; DO ns = 0,1
            zdcorrmatrix(npart2,npart1) = &
                 zdcorrmatrix(npart2,npart1) + &
                 zdualvecs(nx,ny,nz,nt+1,ns,ni,npart2) &
                 *zvecs(nx,ny,nz,nt,ns,ni,npart1) &
                 *CDSQRT(-c0*atovera*(1.D0,0.D0))/L**3
         END DO; END DO
      END DO; END DO

      dVds(nx,ny,nz,nt) = s(nx,ny,nz,nt)
      DO npart1 = 0,n_f-1; DO npart2 = 0,n_f-1
         dVds(nx,ny,nz,nt) = dVds(nx,ny,nz,nt) &
              - DBLE(zdcorrmatrix(npart2,npart1) &
              *zcorrinv(npart1,npart2))
      END DO; END DO

      p_sHMC(nx,ny,nz,nt,0) = &
           p_s(nx,ny,nz,nt) - 0.5D0*eHMC*dVds(nx,ny,nz,nt)

      DO iso = 1,3
         dVdsI(nx,ny,nz,nt,iso) = &
              sI(nx,ny,nz,nt,iso) &
              - DBLE((0.D0,1.D0)*CDSQRT(cI*atovera*(1.D0,0.D0)) &
              *zdVall(nx,ny,nz,nt,0,iso)/L**3)
      END DO
```

```
DO iso = 1,3

   dVdpion(nx,ny,nz,nt,iso) = &
        pion(nx,ny,nz,nt,iso) &
        + atovera/qpi3*( &
        - w1_P*pion(MOD(nx+1,L),ny,nz,nt,iso) &
        - w1_P*pion(MOD(nx-1+L,L),ny,nz,nt,iso) &
        - w1_P*pion(nx,MOD(ny+1,L),nz,nt,iso) &
        - w1_P*pion(nx,MOD(ny-1+L,L),nz,nt,iso) &
        - w1_P*pion(nx,ny,MOD(nz+1,L),nt,iso) &
        - w1_P*pion(nx,ny,MOD(nz-1+L,L),nt,iso) &
        + w2_P*pion(MOD(nx+2,L),ny,nz,nt,iso) &
        + w2_P*pion(MOD(nx-2+L,L),ny,nz,nt,iso) &
        + w2_P*pion(nx,MOD(ny+2,L),nz,nt,iso) &
        + w2_P*pion(nx,MOD(ny-2+L,L),nz,nt,iso) &
        + w2_P*pion(nx,ny,MOD(nz+2,L),nt,iso) &
        + w2_P*pion(nx,ny,MOD(nz-2+L,L),nt,iso) &
        - w3_P*pion(MOD(nx+3,L),ny,nz,nt,iso) &
        - w3_P*pion(MOD(nx-3+L,L),ny,nz,nt,iso) &
        - w3_P*pion(nx,MOD(ny+3,L),nz,nt,iso) &
        - w3_P*pion(nx,MOD(ny-3+L,L),nz,nt,iso) &
        - w3_P*pion(nx,ny,MOD(nz+3,L),nt,iso) &
        - w3_P*pion(nx,ny,MOD(nz-3+L,L),nt,iso))
```

```
            dVdpion(nx,ny,nz,nt,iso) = &
                 dVdpion(nx,ny,nz,nt,iso) &
                 + gA*atovera &
                 /(2.D0*fpi*dsqrt(qpi3)*L**3) &
                 *DBLE(0.D0 &
                 + o1/2.D0*zdVall(MOD(nx-1+L,L),ny,nz,nt,1,iso) &
                 + o1/2.D0*zdVall(nx,MOD(ny-1+L,L),nz,nt,2,iso) &
                 + o1/2.D0*zdVall(nx,ny,MOD(nz-1+L,L),nt,3,iso) &
                 - o1/2.D0*zdVall(MOD(nx+1,L),ny,nz,nt,1,iso) &
                 - o1/2.D0*zdVall(nx,MOD(ny+1,L),nz,nt,2,iso) &
                 - o1/2.D0*zdVall(nx,ny,MOD(nz+1,L),nt,3,iso) &
                 - o2/2.D0*zdVall(MOD(nx-2+L,L),ny,nz,nt,1,iso) &
                 - o2/2.D0*zdVall(nx,MOD(ny-2+L,L),nz,nt,2,iso) &
                 - o2/2.D0*zdVall(nx,ny,MOD(nz-2+L,L),nt,3,iso) &
                 + o2/2.D0*zdVall(MOD(nx+2,L),ny,nz,nt,1,iso) &
                 + o2/2.D0*zdVall(nx,MOD(ny+2,L),nz,nt,2,iso) &
                 + o2/2.D0*zdVall(nx,ny,MOD(nz+2,L),nt,3,iso) &
                 + o3/2.D0*zdVall(MOD(nx-3+L,L),ny,nz,nt,1,iso) &
                 + o3/2.D0*zdVall(nx,MOD(ny-3+L,L),nz,nt,2,iso) &
                 + o3/2.D0*zdVall(nx,ny,MOD(nz-3+L,L),nt,3,iso) &
                 - o3/2.D0*zdVall(MOD(nx+3,L),ny,nz,nt,1,iso) &
                 - o3/2.D0*zdVall(nx,MOD(ny+3,L),nz,nt,2,iso) &
                 - o3/2.D0*zdVall(nx,ny,MOD(nz+3,L),nt,3,iso))

         END DO

         DO iso = 1,3
            p_sIHMC(nx,ny,nz,nt,iso,0) = &
                 p_sI(nx,ny,nz,nt,iso) &
                 - 0.5D0*eHMC*dVdsI(nx,ny,nz,nt,iso)
            p_pionHMC(nx,ny,nz,nt,iso,0) = &
                 p_pion(nx,ny,nz,nt,iso) &
                 - 0.5D0*eHMC*dVdpion(nx,ny,nz,nt,iso)
         END DO

      END DO; END DO; END DO
   END DO
```

A tale of two interactions

Two LO chiral EFT interactions, A and B, have nearly identical nucleon-nucleon phase shifts and well as three- and four-nucleon bound states

Nucleus	A (LO)	B (LO)	A (LO + Coulomb)	B (LO + Coulomb)	Experiment
^{8}Be	−58.61(14)	−59.73(6)	−56.51(14)	−57.29(7)	−56.591
^{12}C	−88.2(3)	−95.0(5)	−84.0(3)	−89.9(5)	−92.162
^{16}O	−117.5(6)	−135.4(7)	−110.5(6)	−126.0(7)	−127.619
^{20}Ne	−148(1)	−178(1)	−137(1)	−164(1)	−160.645

Elhatisari, Li, Rokash, Alarcon, Du, Klein, Lu, Meißner, Epelbaum, Krebs, Lähde, D.L., Rupak, PRL 117, 132501 (2016)

Nucleus	A (LO)	B (LO)	A (LO + Coulomb)	B (LO + Coulomb)	Experiment
^{8}Be	$-58.61(14)$	$-59.73(6)$	$-56.51(14)$	$-57.29(7)$	-56.591
^{12}C	$-88.2(3)$	$-95.0(5)$	$-84.0(3)$	$-89.9(5)$	-92.162
^{16}O	$-117.5(6)$	$-135.4(7)$	$-110.5(6)$	$-126.0(7)$	-127.619
^{20}Ne	$-148(1)$	$-178(1)$	$-137(1)$	$-164(1)$	-160.645

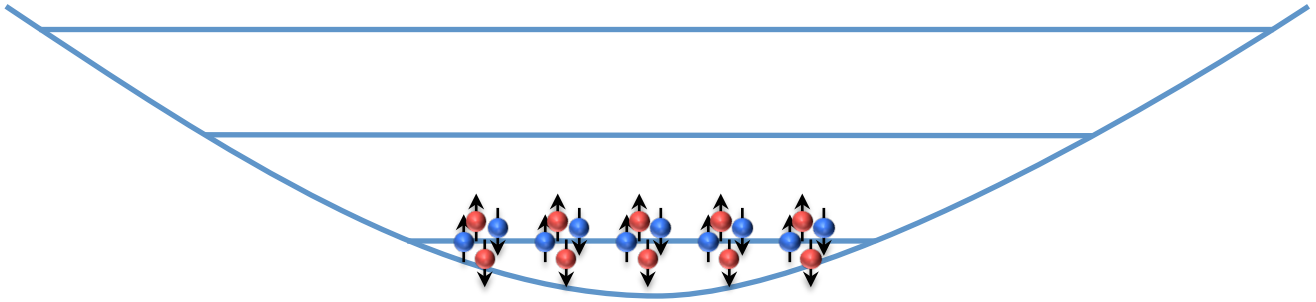
$$\frac{E_{^{8}\mathrm{Be}}}{E_{^{4}\mathrm{He}}} = 1.997(6)$$

$$\frac{E_{^{12}\mathrm{C}}}{E_{^{4}\mathrm{He}}} = 3.00(1)$$

$$\frac{E_{^{16}\mathrm{O}}}{E_{^{4}\mathrm{He}}} = 4.00(2)$$

$$\frac{E_{^{20}\mathrm{Ne}}}{E_{^{4}\mathrm{He}}} = 5.03(3)$$

Bose condensate of alpha particles!



Nucleus	A (LO)	B (LO)	A (LO + Coulomb)	B (LO + Coulomb)	Experiment
^{8}Be	−58.61(14)	−59.73(6)	−56.51(14)	−57.29(7)	−56.591
^{12}C	−88.2(3)	−95.0(5)	−84.0(3)	−89.9(5)	−92.162
^{16}O	−117.5(6)	−135.4(7)	−110.5(6)	−126.0(7)	−127.619
^{20}Ne	−148(1)	−178(1)	−137(1)	−164(1)	−160.645

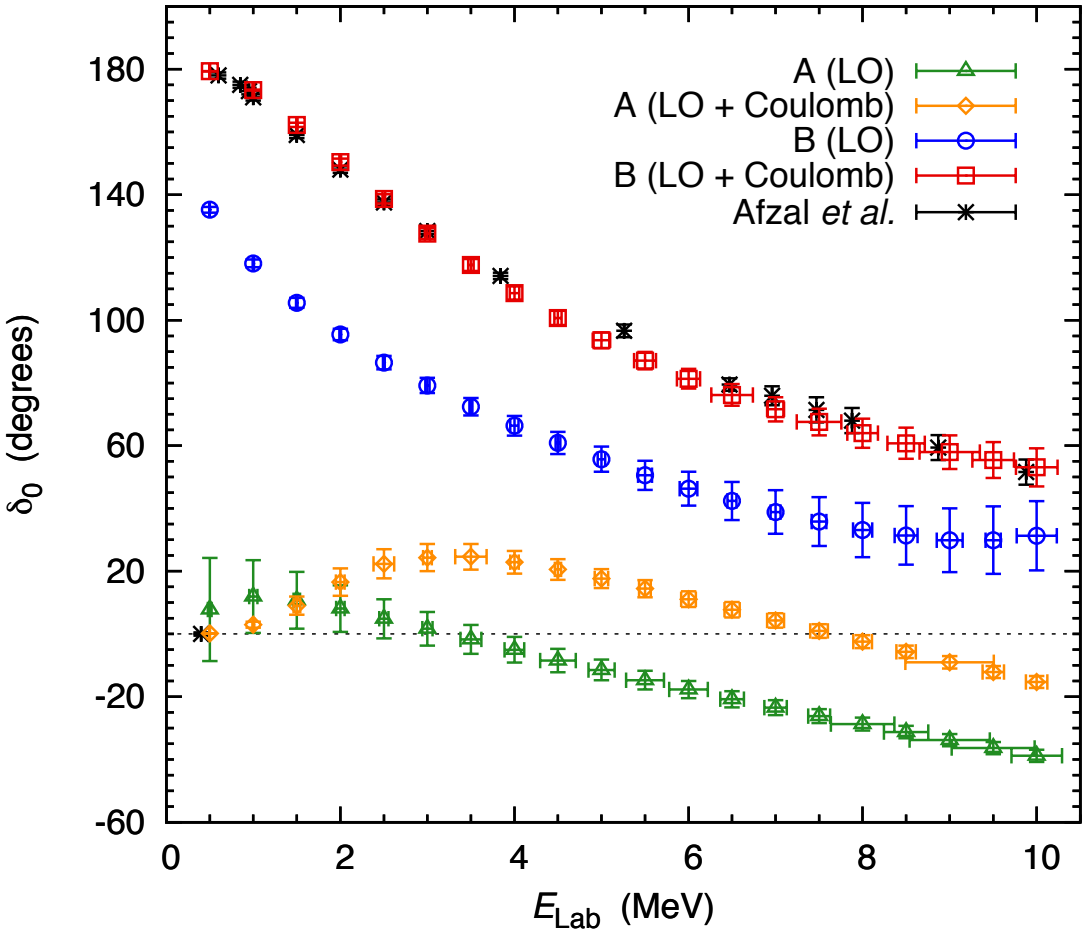
$$\frac{E_{^{8}\mathrm{Be}}}{E_{^{4}\mathrm{He}}} = 1.997(6)$$

$$\frac{E_{^{12}\mathrm{C}}}{E_{^{4}\mathrm{He}}} = 3.00(1)$$

$$\frac{E_{^{16}\mathrm{O}}}{E_{^{4}\mathrm{He}}} = 4.00(2)$$

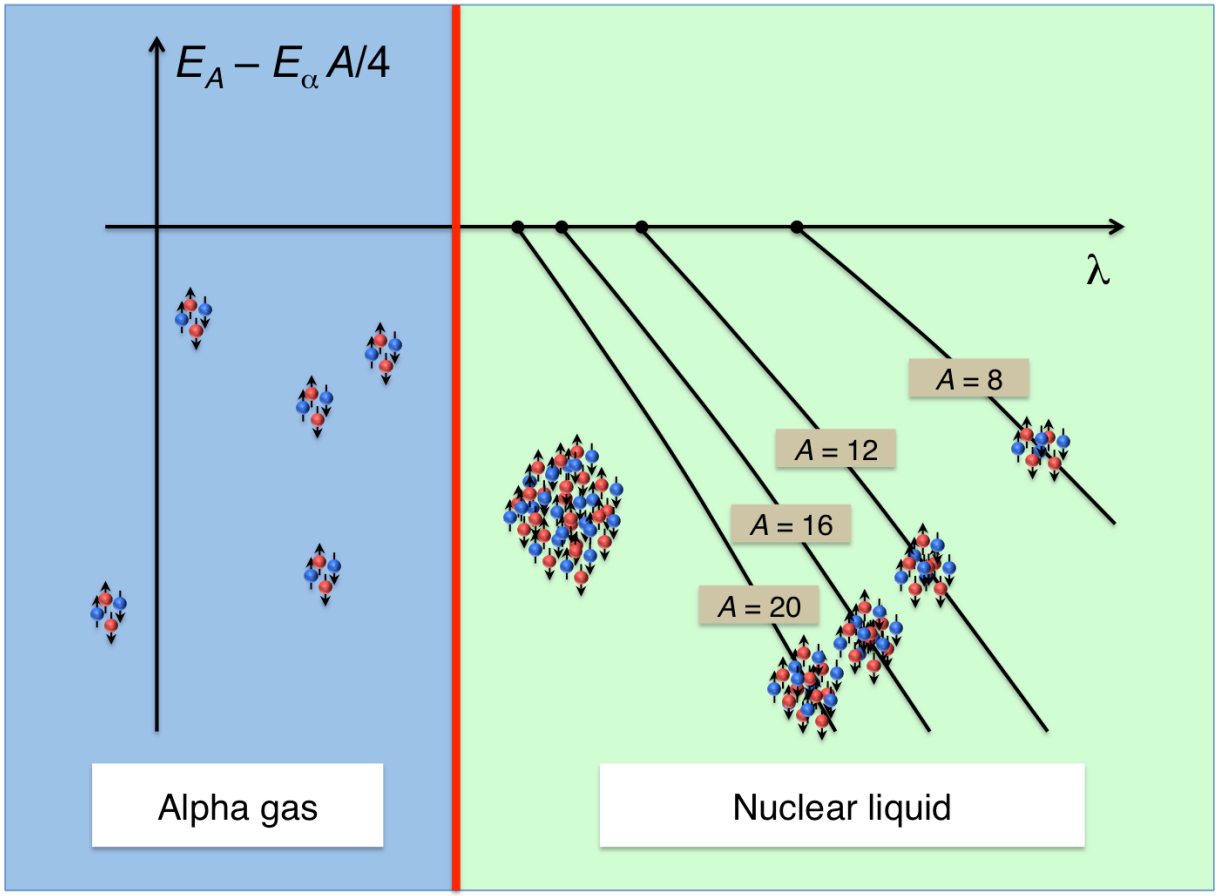
$$\frac{E_{^{20}\mathrm{Ne}}}{E_{^{4}\mathrm{He}}} = 5.03(3)$$

Alpha-alpha scattering



Alpha-alpha interaction not uniquely determined by low-energy few-body data

Control parameters: Sensitivity to interaction range and locality



Elhatisari, Li, Rokash, Alarcon, Du, Klein, Lu, Meißner, Epelbaum, Krebs, Lähde, D.L., Rupak, PRL 117, 132501 (2016)