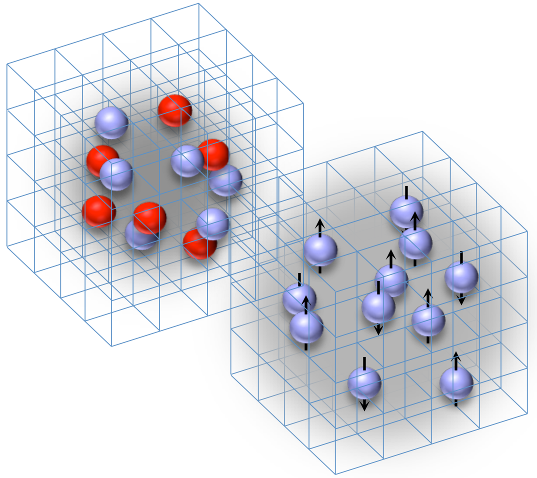
Lecture 21: Spherical Wall Method



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

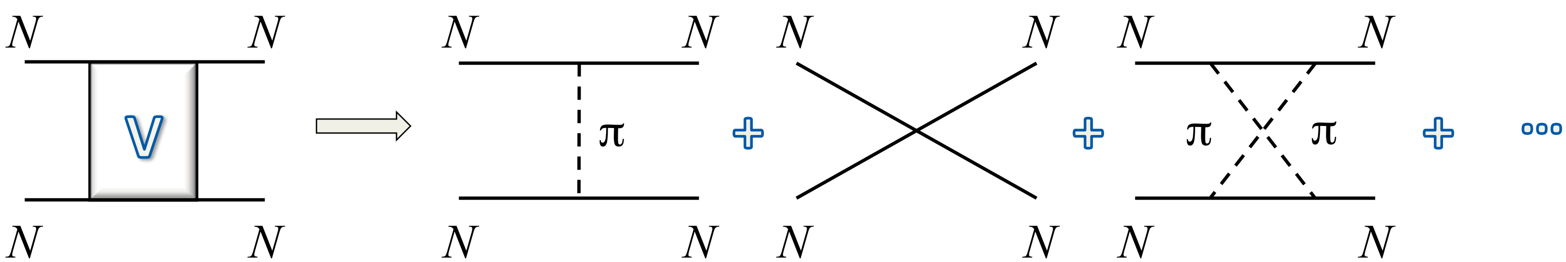
ECT*

From Quarks and Gluons to Nuclear Forces and Structure

Chiral EFT for low-energy nucleons

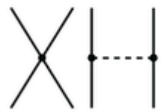
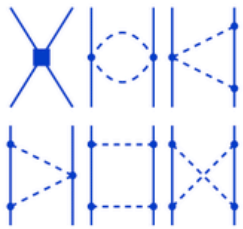
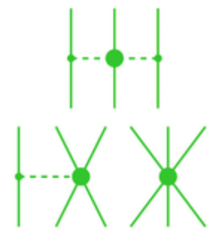
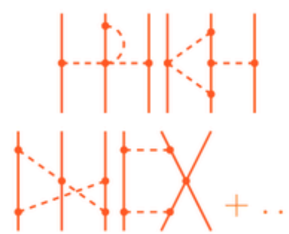
Weinberg, PLB 251 (1990) 288; NPB 363 (1991) 3

Construct the effective potential order by order



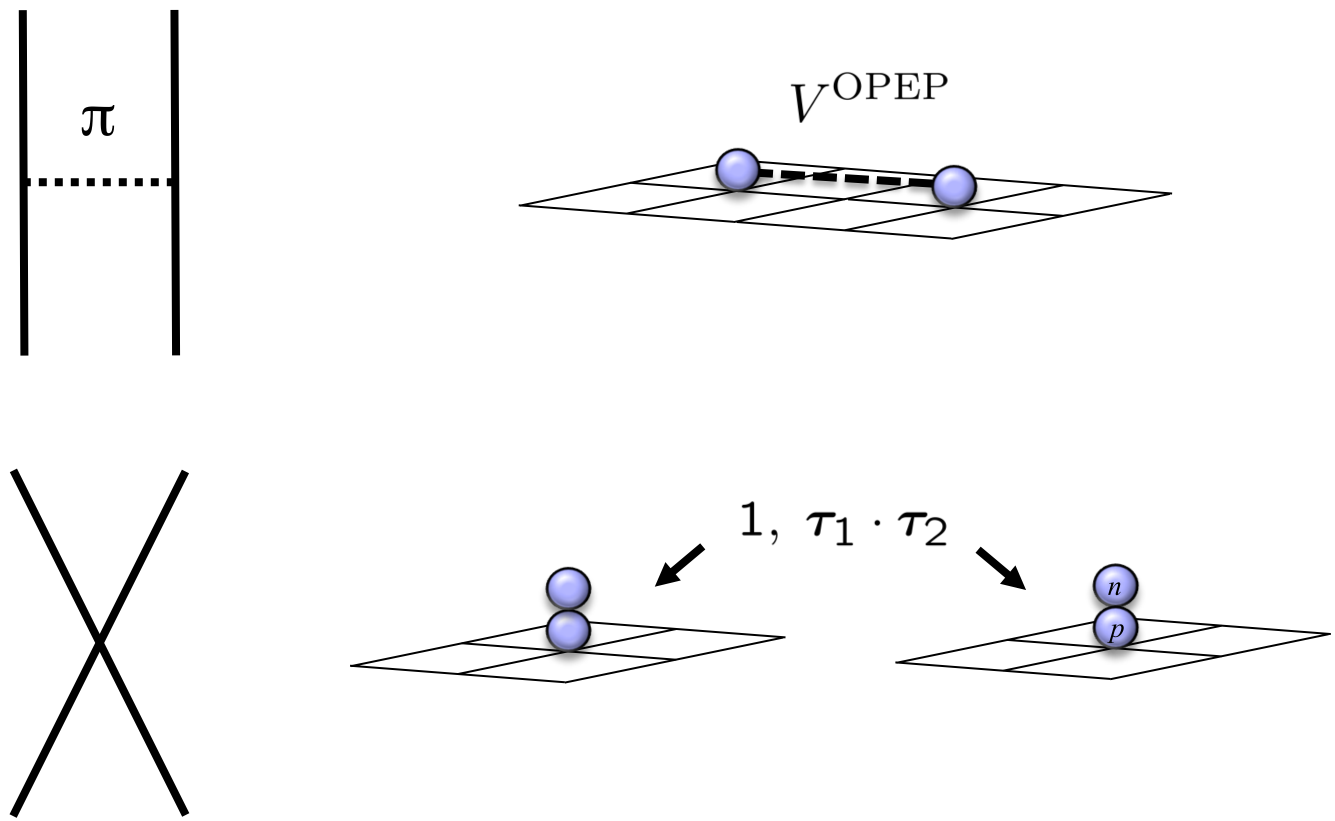
V^{OPEP} Contact interactions $1,\ \boldsymbol{\tau}_1 \cdot \boldsymbol{\tau}_2$ V^{TPEP}

Leading order (LO)   Next-to-leading order (NLO)

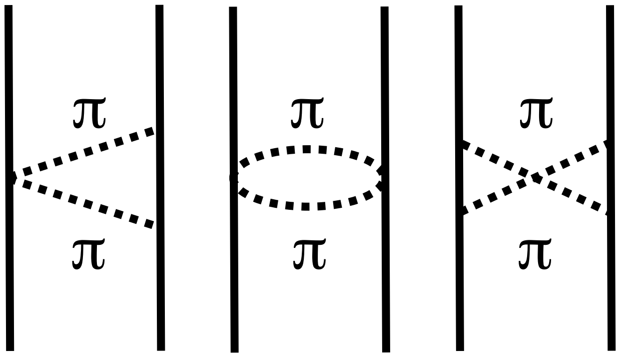
	NN	3N	4N
LO $(Q/\Lambda_\chi)^0$			
NLO $(Q/\Lambda_\chi)^2$			
NNLO $(Q/\Lambda_\chi)^3$			
N^{3}LO $(Q/\Lambda_\chi)^4$			

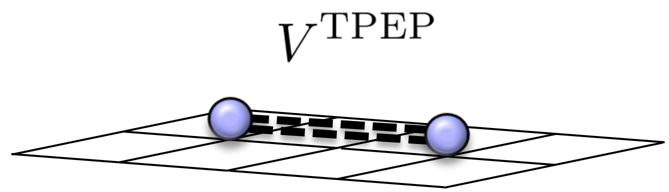
Ordonez et al. '94; Friar & Coon '94; Kaiser et al. '97; Epelbaum et al. '98, '03, …; Kaiser '99-'01; Higa et al. '03; …

Leading order on the lattice

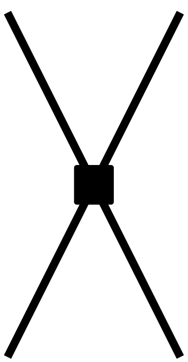


Next-to-leading order on the lattice

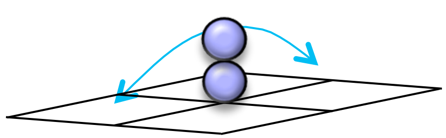


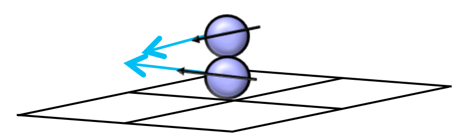


. . .

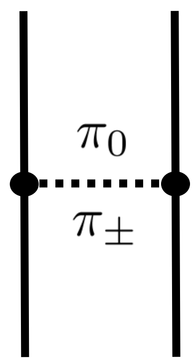


$$\vec{\nabla}_1 \cdot \vec{\nabla}_2 \qquad (\vec{\sigma}_1 \cdot \vec{\nabla}_1)\,(\vec{\sigma}_2 \cdot \vec{\nabla}_2)$$



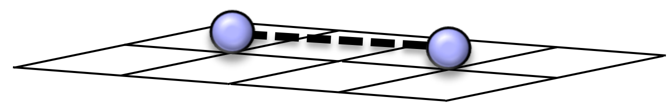


Pion mass difference

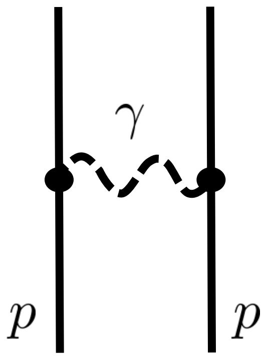


$$m_{\pi_0} \neq m_{\pi_\pm}$$

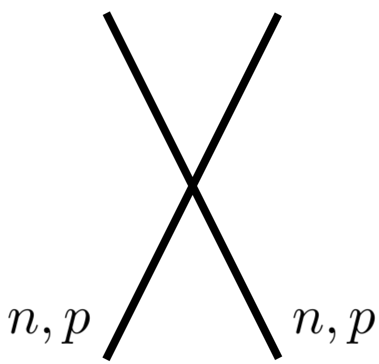
$$V_{\pi_0}^{\mathrm{OPEP}}, V_{\pi_\pm}^{\mathrm{OPEP}}$$

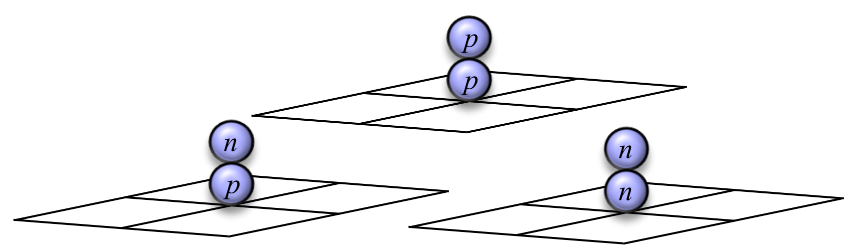


Coulomb potential



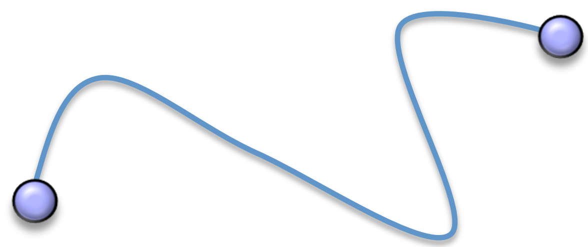
Charge symmetry breaking
Charge independence breaking





Spherical Wall Method

Imagine a massless string connecting two particles. There is no effect on the center-of-mass motion. However, the two particles cannot separate beyond the length of the string. We have imposed a hard spherical wall boundary condition on the relative motion.



This can now be used to extract scattering phase shifts for the two interacting particles.

Borasoy, Epelbaum, Krebs, D.L., Meißner, EPJA 34 (2007) 185

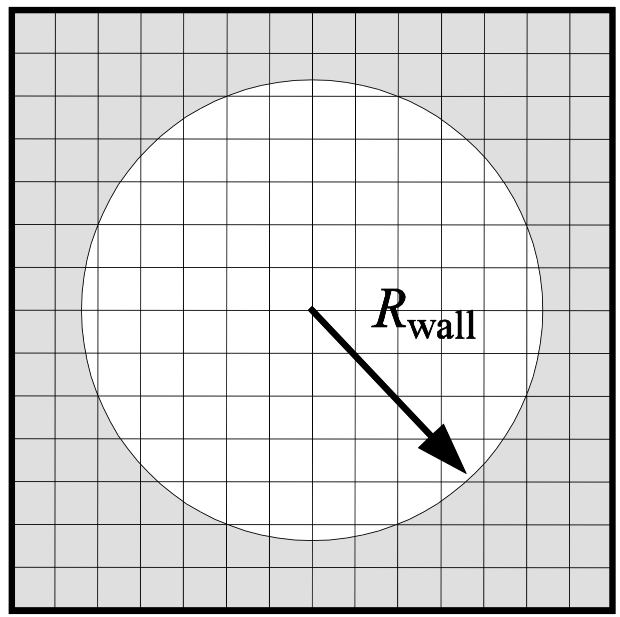
Lu, Lähde, D.L., Meißner, PLB 760 (2016) 309

Bovermann, Epelbaum, Krebs, D.L. arXiv:1905.02492

The spherical wall method has been used in continuous space calculations.

Carlson, Pandharipande, Wiringa, NPA 424 (1984) 47

But the method is particularly useful for finite-volume lattice calculations.



In particular, it removes the breaking of rotation symmetry caused by the periodic boundaries.

The radial Schrödinger equation gives

$$\left\{-\frac{1}{2\mu}\frac{d}{dr}\left(r^2\frac{d}{dr}\right)+\frac{\ell(\ell+1)}{2\mu r^2}+V(r)\right\}R(r)=ER(r)$$

$$u(r)=rR(r)$$

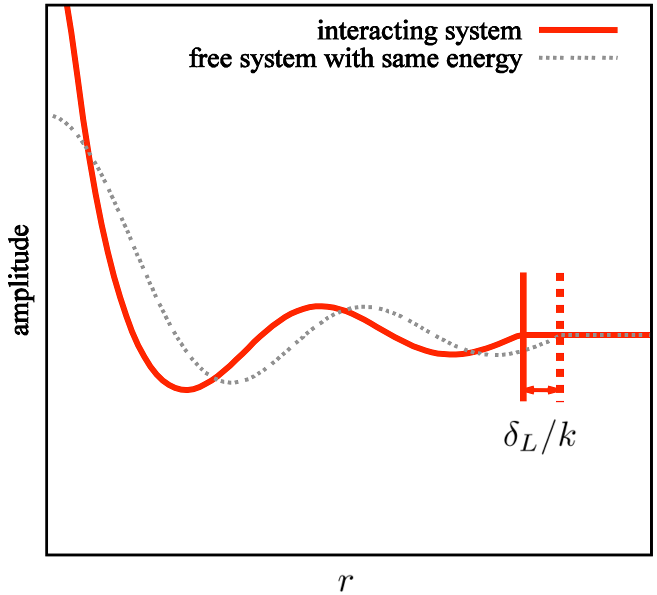
$$-\frac{1}{2\mu}\frac{d^2u}{dr^2}+\left[\frac{\ell(\ell+1)}{2\mu r^2}+V(r)\right]u(r)=Eu(r)$$

Beyond the range of the interaction, the wave function has the form

$$R(r)\propto\cos\delta_\ell j_\ell(kr)-\sin\delta_\ell y_\ell(kr)$$

The wave function vanishes at the wall boundary. Therefore the phase shift is

$$\delta_\ell = \tan^{-1}\left[\frac{j_\ell(kR_{\rm wall})}{y_\ell(kR_{\rm wall})}\right]$$



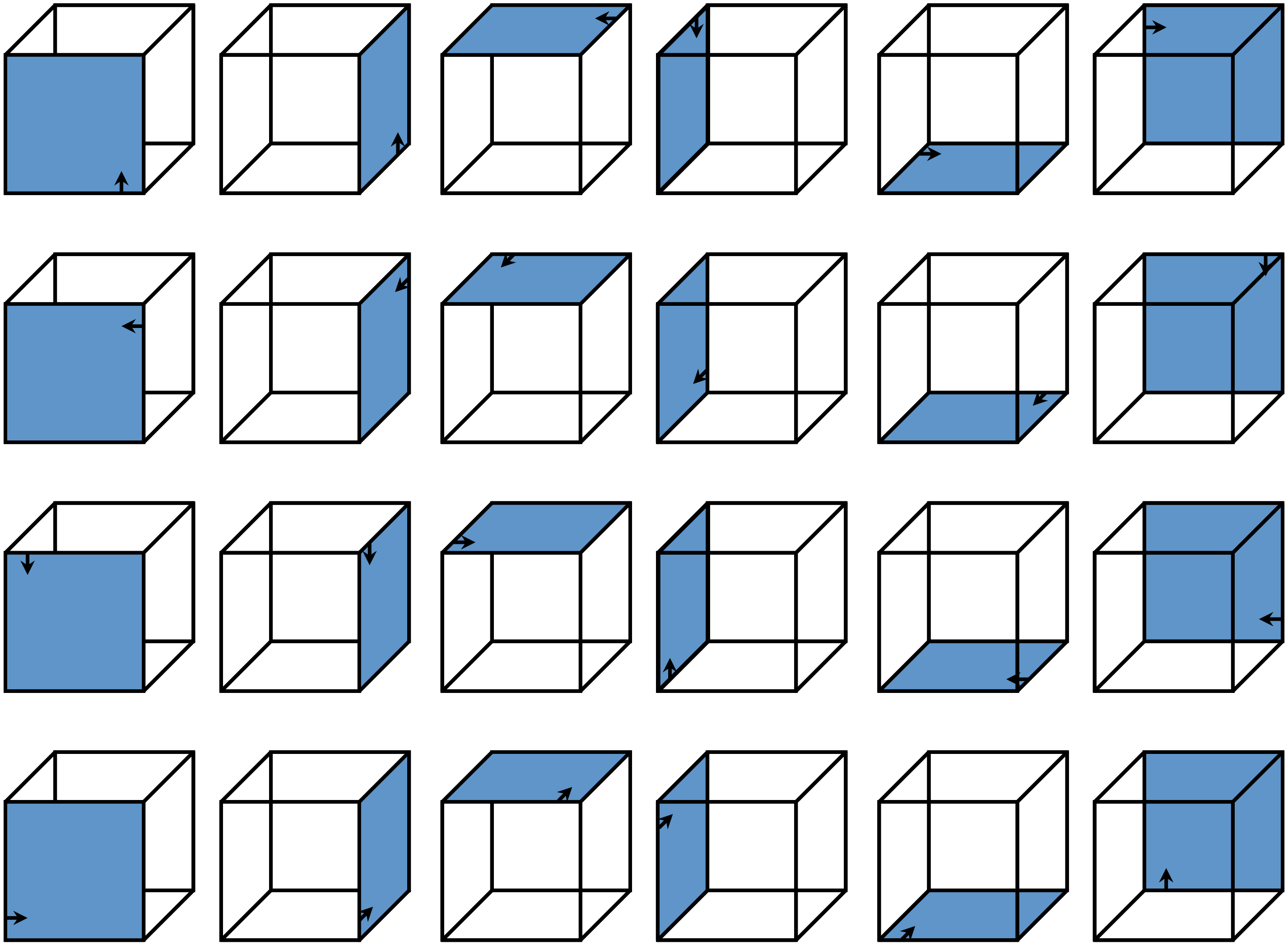
In order to reduce systematic errors, it is useful to also calculate the same quantities for standing waves in the free theory with the same number of radial nodes.

$$\delta_\ell^{\mathrm{free}} = \tan^{-1}\left[\frac{j_\ell(k^{\mathrm{free}} R_{\mathrm{wall}})}{y_\ell(k^{\mathrm{free}} R_{\mathrm{wall}})}\right]$$

By subtracting the two, we can get a better estimate of the phase shifts

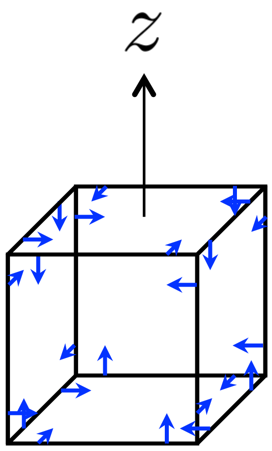
$$\delta_\ell^{\mathrm{improved}} = \delta_\ell - \delta_\ell^{\mathrm{free}}$$

Cubic symmetry group



A_1

$J_z = 0 \bmod 4$

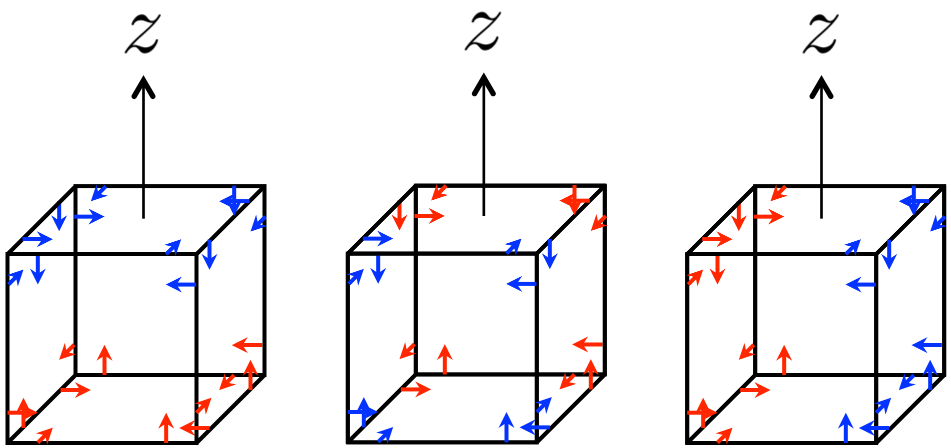


blue = +1

red = −1

$$T_1$$

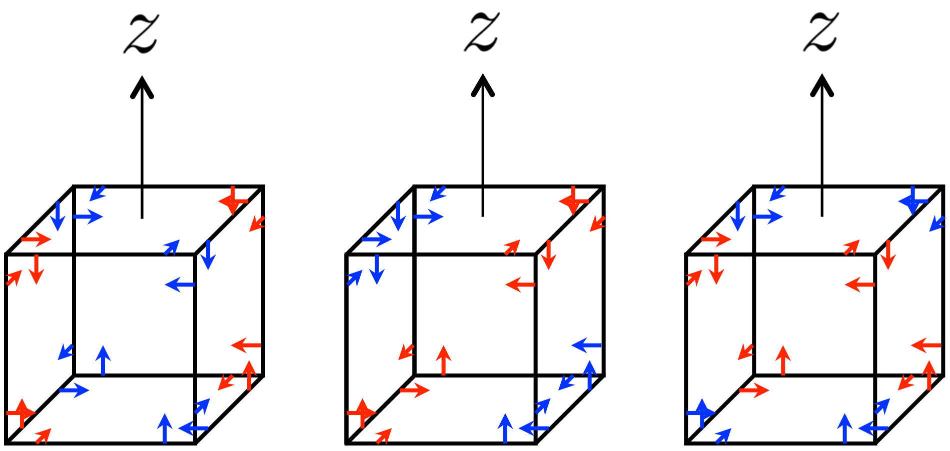
$$J_z = 0, 1, 3 \bmod 4$$



blue = +1
red = −1

$$T_2$$

$$J_z = 1, 2, 3 \bmod 4$$

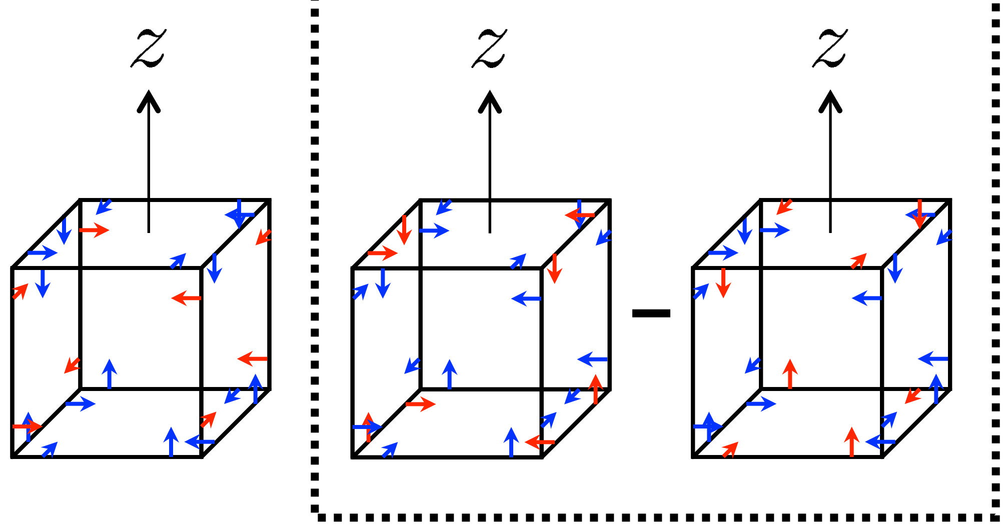


blue = +1

red = −1

E

$J_z = 0, 2 \bmod 4$

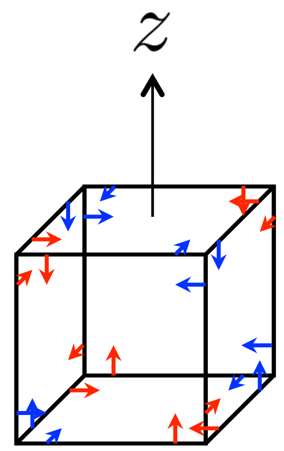


blue = +1

red = −2

$$A_2$$

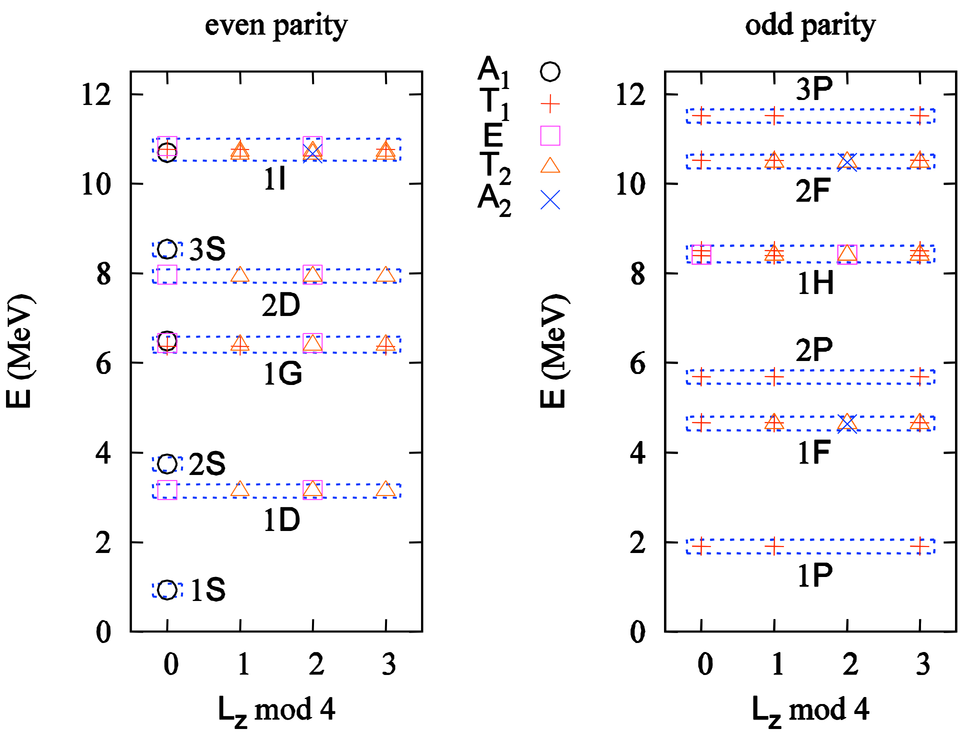
$$J_z = 2 \bmod 4$$



blue = +1
red = −1

Free energy levels

$$R_{\text{wall}} = 10a$$
$$a = 1.97 \text{ fm}$$



Example: Gaussian potential in continuum

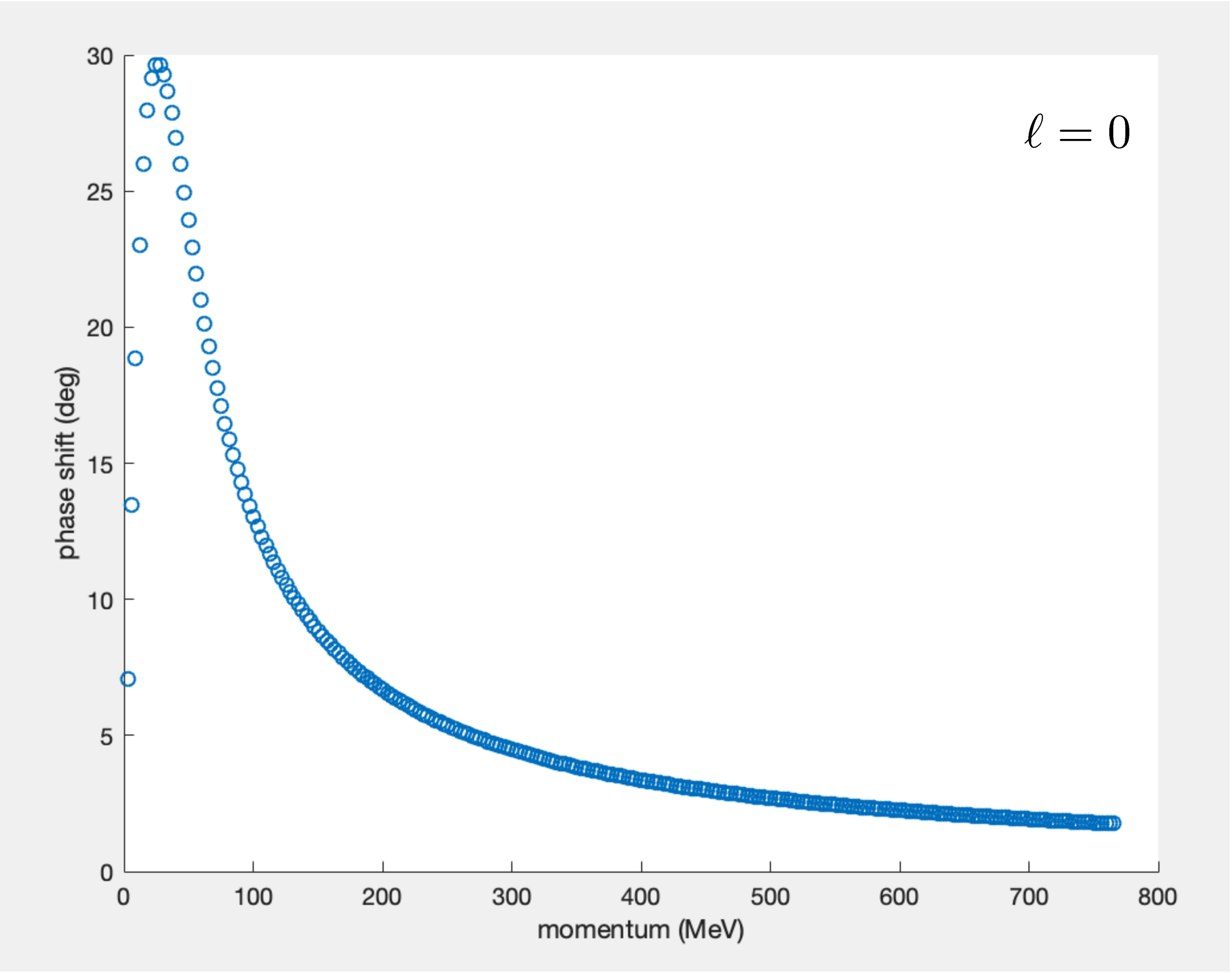
$$V(r) = Ce^{-\frac{r^2}{2R_0^2}}$$

$$C = -2 \text{ MeV},\ R_0 = 2 \times 10^{-2} \text{ MeV}^{-1}$$

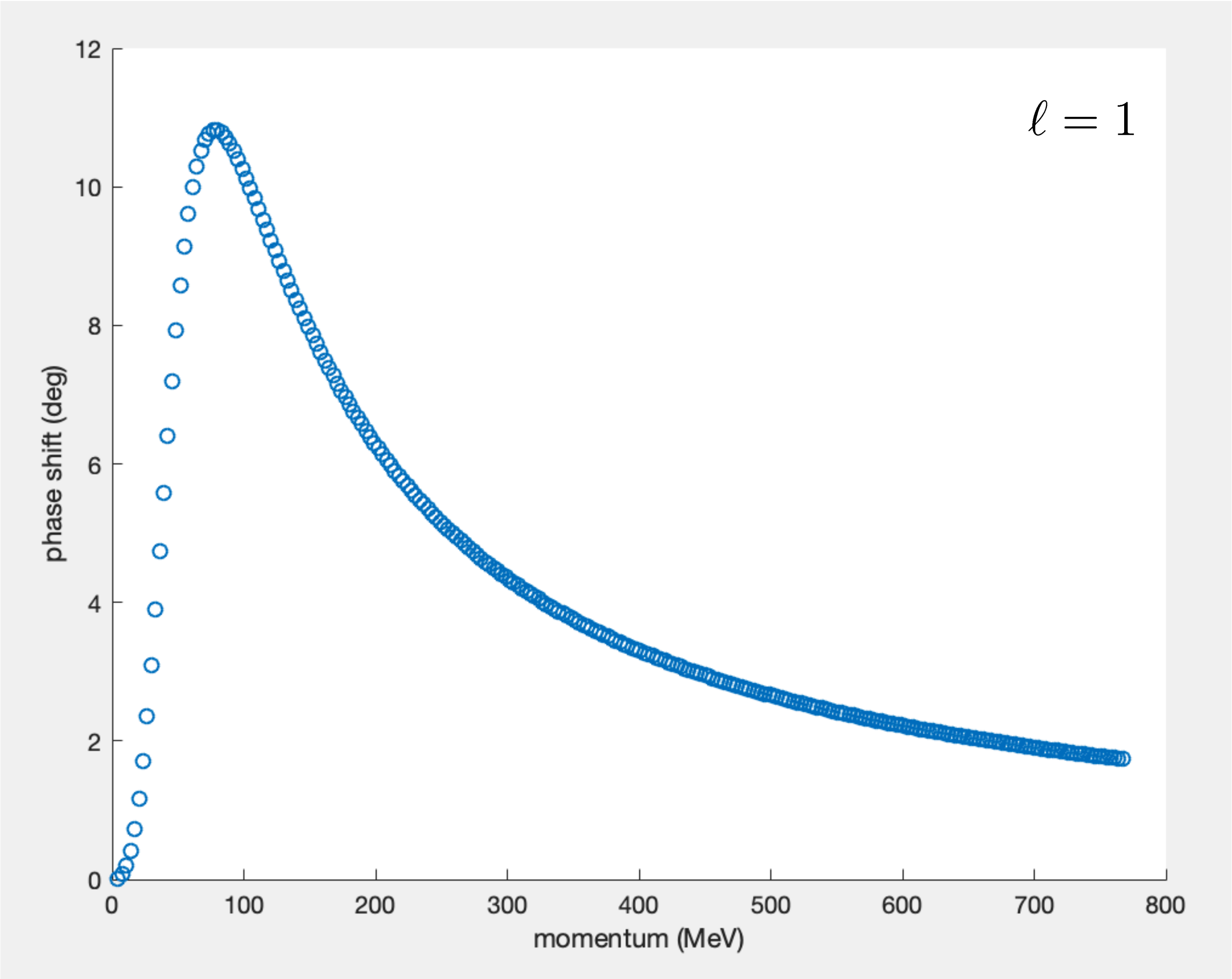
$$\mu = m/2,\ m = 938.92 \text{ MeV}$$

```
clear all

% The physical parameters in units of MeV raised to the appropriate power

lattsp = 0.001;
Rwall = 1.000;
ell = 0;

Gausswidth = 0.02;
Gaussdepth = 2;

mass = 0.5*938.92*lattsp;
nwall = floor(Rwall/lattsp);
numeigs = floor(nwall/4);

r = [0:nwall];
centrifugal = (ell*(ell+1))./(2*mass*max(r.^2,0.5));
V = -Gaussdepth*lattsp*exp(-0.5*r.^2/(Gausswidth/lattsp)^2);

Hfree = sparse(r+1,r+1,1.0/mass);
Hfree = Hfree + sparse(mod(r+1,nwall+1)+1,r+1,-0.5/mass);
Hfree = Hfree + sparse(mod(r-1,nwall+1)+1,r+1,-0.5/mass);
Hfree = Hfree + sparse(r+1,r+1,centrifugal);

% Must force psi(0) to be zero
Hfree(1,1) = Hfree(1,1) + 10^9;
Hfree(nwall+1,nwall+1) = Hfree(nwall+1,nwall+1) + 10^9;
```

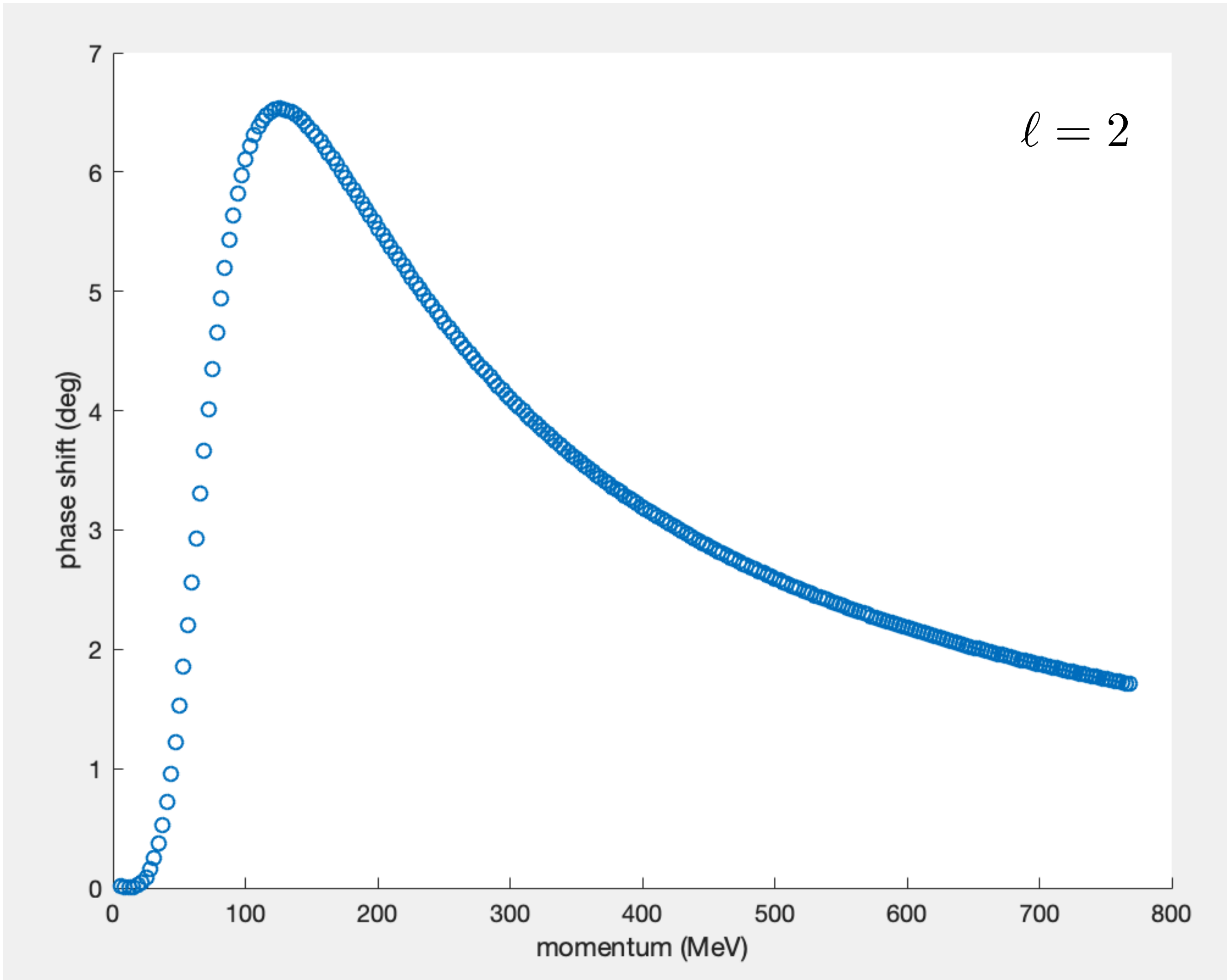
```
H = Hfree + sparse(r+1,r+1,V);

Efree = eigs(Hfree,numeigs,'SA');
E = eigs(H,numeigs,'SA');

bound = 0;

for nn = 1:numeigs
    if (E(nn) > 0)
        pfree(nn) = sqrt(2*mass*Efree(nn));
        p(nn) = sqrt(2*mass*E(nn));
        phase(nn) = ...
            mod(atan(besselj(ell+0.5,p(nn)*nwall) ...
            /bessely(ell+0.5,p(nn)*nwall)) ...
            - atan(besselj(ell+0.5,pfree(nn)*nwall) ...
            /bessely(ell+0.5,pfree(nn)*nwall)),pi);
    else
        bound = bound + 1;
    end
end

scatt_states = [bound+1:numeigs];
scatter(p(scatt_states)'/lattsp,phase(scatt_states)*180/pi')
xlabel('momentum (MeV)')
ylabel('phase shift (deg)')
```



$\ell = 0$
phase shift (deg)
momentum (MeV)
0
5
10
15
20
25
30
0
100
200
300
400
500
600
700
800



$\ell = 1$
phase shift (deg)
momentum (MeV)
0
2
4
6
8
10
12
0
100
200
300
400
500
600
700
800



$\ell = 2$
phase shift (deg)
momentum (MeV)
0
1
2
3
4
5
6
7
0
100
200
300
400
500
600
700
800

In the original spherical wall paper,

Borasoy, Epelbaum, Krebs, D.L., Meißner, EPJA 34 (2007) 185

we solved for eigenstates of the full three-dimensional lattice Hamiltonian. This was numerically expensive.

Later we realized that we could construct an approximate but very high-quality radial equation by grouping together lattice coordinates with nearly the same magnitude and prescribing the angular dependence according to spherical harmonic projections

$$\psi(\vec{r}) = R(r_{\rm bin}) Y_{\ell,\ell_z}(\hat{r}), \quad ||\vec{r}| - r_{\rm bin}| < \Delta_{\rm bin}$$

Example: Gaussian potential on the lattice

$$V(r) = Ce^{-\frac{r^2}{2R_0^2}}$$

$$C = -2\ \mathrm{MeV},\ R_0 = 2 \times 10^{-2}\ \mathrm{MeV}^{-1}$$

$$\mu = m/2,\ m = 938.92\ \mathrm{MeV}$$

```
clear all

L = 80;
lattsp = 0.005;
mass = 0.5*938.92*lattsp;
ell = 2;
ellz = 0;

Gausswidth = 0.02;
Gaussdepth = 2;

Rwall = 0.18;
nwall = floor(Rwall/lattsp);
if (nwall >= L/2)
    "L is too small for Rwall"
    stop
end
numeigs = nwall/2;

r = [0:L^3-1];
nx = mod(r,L);
ny = mod((r-nx)/L,L);
nz = mod((r-ny*L-nx)/L^2,L);
x = (nx < L/2).*nx + (nx > L/2).*(nx-L);
y = (ny < L/2).*ny + (ny > L/2).*(ny-L);
z = (nz < L/2).*nz + (nz > L/2).*(nz-L);
```

```
r2 = min(nx.^2,(L-nx).^2) + min(ny.^2,(L-ny).^2) + min(nz.^2,(L-nz).^2);
rabs = sqrt(r2);
rabs(1) = 1.E-10;

nstep = 0;
dstep = 0.1;
for step = 0:dstep:nwall
    points = find(rabs >= step - 0.5*dstep & rabs < step + 0.5*dstep);
    numpoints = size(points,2);
    if (size(points,2) > 0)
        vpoints = zeros(L^3,1);
        % normalization will be fixed later
        if (ell == 0)
            vpoints(points) = 1;
        elseif (ell == 1 & ellz == 1)
            vpoints(points) = -(x(points)+i*y(points))./rabs(points);
        elseif (ell == 1 & ellz == 0)
            vpoints(points) = z(points)./rabs(points);
        elseif (ell == 1 & ellz == -1)
            vpoints(points) = (x(points)-i*y(points))./rabs(points);
        elseif (ell == 2 & ellz == 2)
            vpoints(points) = (x(points)+i*y(points)).^2./rabs(points).^2;
        elseif (ell == 2 & ellz == 1)
            vpoints(points) = ...
                -(x(points)+i*y(points)).*z(points)./rabs(points).^2;
        elseif (ell == 2 & ellz == 0)
            vpoints(points) = ...
                (2*z(points).^2-x(points).^2-y(points).^2)./rabs(points).^2;
```

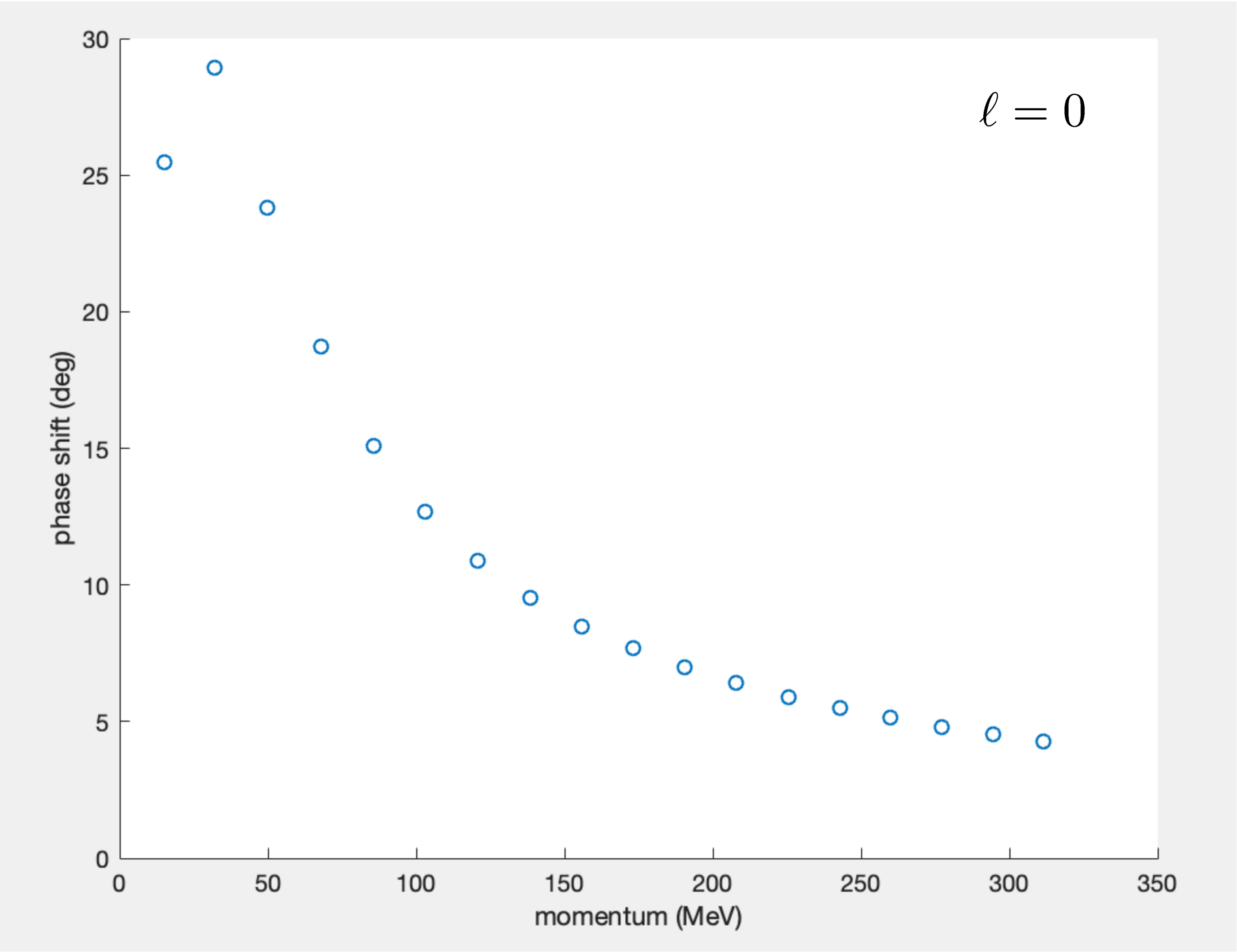
```
        elseif (ell == 2 & ellz == -1)
            vpoints(points) = ...
                (x(points)-i*y(points)).*z(points)./rabs(points).^2;
        elseif (ell == 2 & ellz == -2)
            vpoints(points) = (x(points)-i*y(points)).^2./rabs(points).^2;
        else
            "ell, ellz not available"
            stop
        end
        norm = sqrt(vpoints'*vpoints);
        if (norm > 0)
            nstep = nstep + 1;
            projectors(:,nstep) = vpoints/norm;
            dist(nstep) = step;
        end
    end
end

V = -Gaussdepth*lattsp*exp(-0.5*r2/(Gausswidth/lattsp)^2);
Vwall = 10^6*(rabs > nwall)';

%%%%%%%%%%%%%%%

w0 = 49.D0/36.D0;
w1 = 3.D0/2.D0;
w2 = 3.D0/20.D0;
w3 = 1.D0/90.D0;
```

```
% getting the coordinates for nearest neighbors in the x,y,z
% directions (xp is +1 in the x-direction, xm is -1 in the
% x-direction, and so on)

r_xp = nz*L^2          + ny*L          + mod(nx+1,L);
r_xpp = nz*L^2          + ny*L          + mod(nx+2,L);
r_xppp = nz*L^2          + ny*L          + mod(nx+3,L);
r_xm = nz*L^2          + ny*L          + mod(nx-1,L);
r_xmm = nz*L^2          + ny*L          + mod(nx-2,L);
r_xmmm = nz*L^2          + ny*L          + mod(nx-3,L);

r_yp = nz*L^2          + mod(ny+1,L)*L + nx;
r_ypp = nz*L^2          + mod(ny+2,L)*L + nx;
r_yppp = nz*L^2          + mod(ny+3,L)*L + nx;
r_ym = nz*L^2          + mod(ny-1,L)*L + nx;
r_ymm = nz*L^2          + mod(ny-2,L)*L + nx;
r_ymmm = nz*L^2          + mod(ny-3,L)*L + nx;

r_zp = mod(nz+1,L)*L^2 + ny*L          + nx;
r_zpp = mod(nz+2,L)*L^2 + ny*L          + nx;
r_zppp = mod(nz+3,L)*L^2 + ny*L          + nx;
r_zm = mod(nz-1,L)*L^2 + ny*L          + nx;
r_zmm = mod(nz-2,L)*L^2 + ny*L          + nx;
r_zmmm = mod(nz-3,L)*L^2 + ny*L          + nx;
```

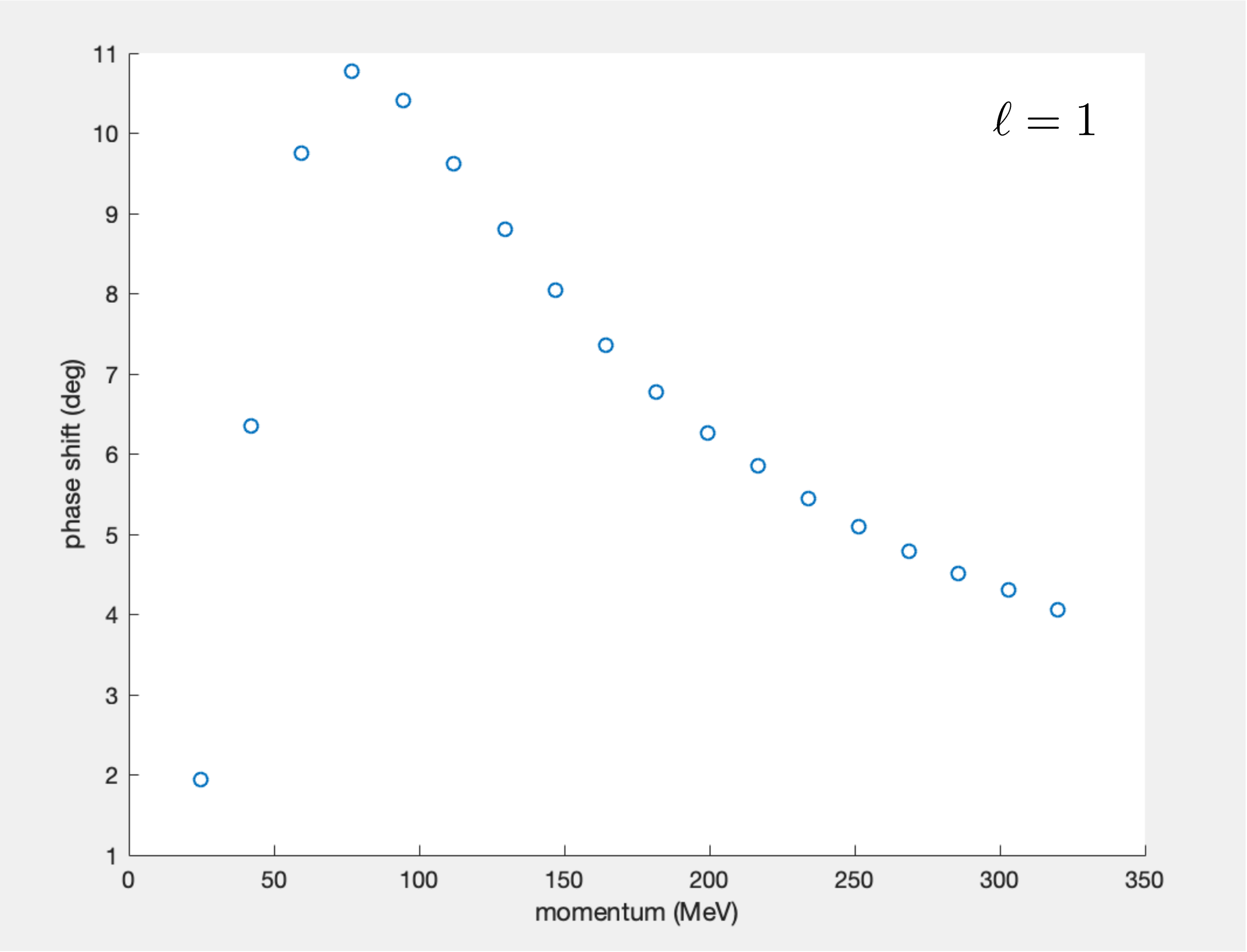
```
Hfree = sparse([1:L^3],[1:L^3],3.0/mass*w0);

Hfree = Hfree + sparse([1:L^3],r_xp+1,-0.5/mass*w1);
Hfree = Hfree - sparse([1:L^3],r_xpp+1,-0.5/mass*w2);
Hfree = Hfree + sparse([1:L^3],r_xppp+1,-0.5/mass*w3);
Hfree = Hfree + sparse([1:L^3],r_xm+1,-0.5/mass*w1);
Hfree = Hfree - sparse([1:L^3],r_xmm+1,-0.5/mass*w2);
Hfree = Hfree + sparse([1:L^3],r_xmmm+1,-0.5/mass*w3);

Hfree = Hfree + sparse([1:L^3],r_yp+1,-0.5/mass*w1);
Hfree = Hfree - sparse([1:L^3],r_ypp+1,-0.5/mass*w2);
Hfree = Hfree + sparse([1:L^3],r_yppp+1,-0.5/mass*w3);
Hfree = Hfree + sparse([1:L^3],r_ym+1,-0.5/mass*w1);
Hfree = Hfree - sparse([1:L^3],r_ymm+1,-0.5/mass*w2);
Hfree = Hfree + sparse([1:L^3],r_ymmm+1,-0.5/mass*w3);

Hfree = Hfree + sparse([1:L^3],r_zp+1,-0.5/mass*w1);
Hfree = Hfree - sparse([1:L^3],r_zpp+1,-0.5/mass*w2);
Hfree = Hfree + sparse([1:L^3],r_zppp+1,-0.5/mass*w3);
Hfree = Hfree + sparse([1:L^3],r_zm+1,-0.5/mass*w1);
Hfree = Hfree - sparse([1:L^3],r_zmm+1,-0.5/mass*w2);
Hfree = Hfree + sparse([1:L^3],r_zmmm+1,-0.5/mass*w3);

Hfree = Hfree + sparse([1:L^3],[1:L^3],Vwall);
H = Hfree + sparse([1:L^3],[1:L^3],V);
```

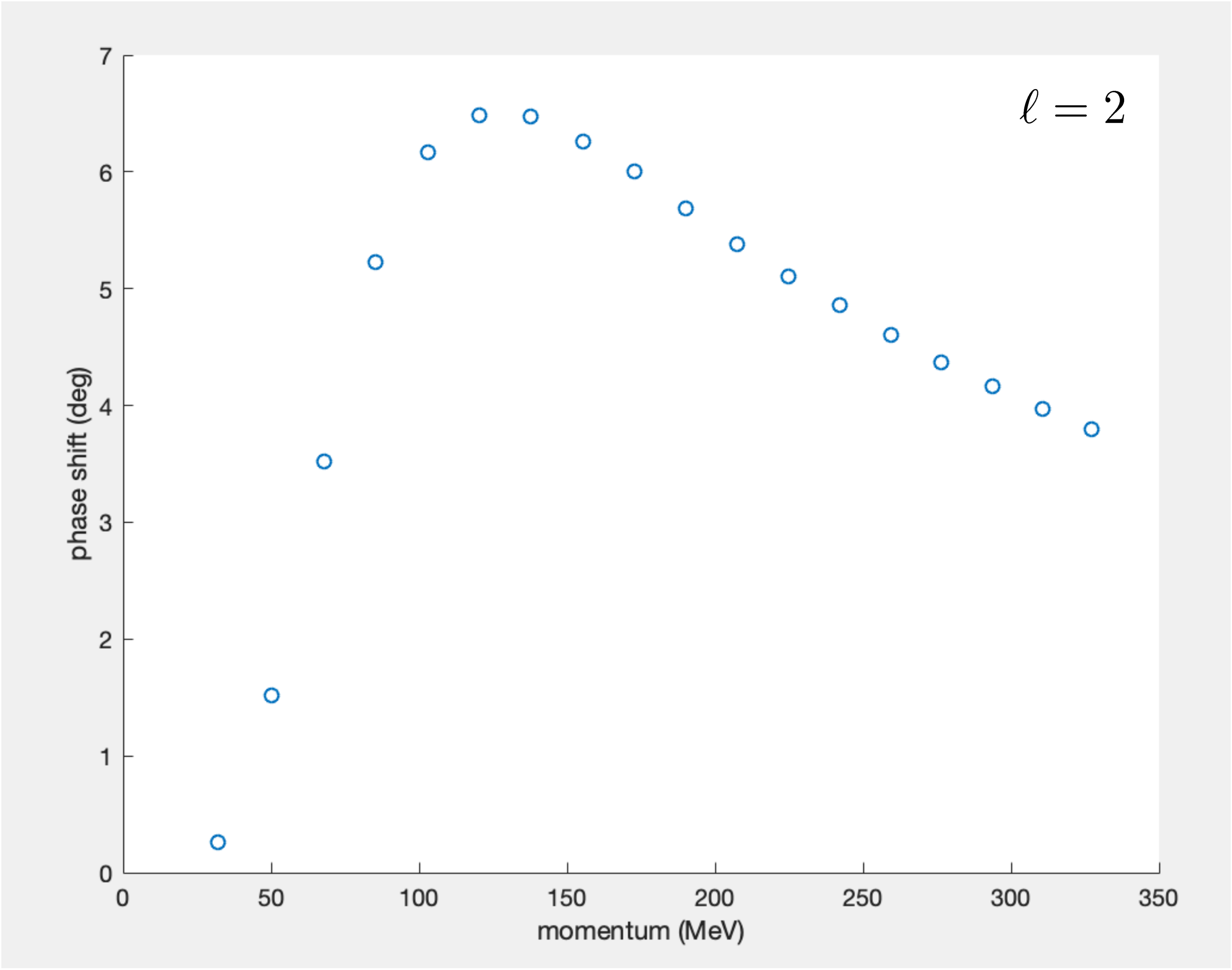
```
Hfreerad = projectors'*(Hfree*projectors);
Hrad = projectors'*(H*projectors);
Hfreerad = (Hfreerad + Hfreerad')/2;
Hrad = (Hrad + Hrad')/2;

[vfreerad,dfreerad] = eigs(Hfreerad,numeigs,'sr');
Efreerad = diag(dfreerad);
Efreerad = sort(Efreerad);

[vrad,drad] = eigs(Hrad,numeigs,'sr');
Erad = diag(drad);
Erad = sort(Erad);

format long
[Erad Efreerad]
```

```
bound = 0;

for nn = 1:numeigs
    if (Erad(nn) > 0)
        pfreerad(nn) = sqrt(2*mass*Efreerad(nn));
        prad(nn) = sqrt(2*mass*Erad(nn));
        phase(nn) = ...
            mod(atan(besselj(ell+0.5,prad(nn)*nwall) ...
            /bessely(ell+0.5,prad(nn)*nwall)) ...
            - atan(besselj(ell+0.5,pfreerad(nn)*nwall) ...
            /bessely(ell+0.5,pfreerad(nn)*nwall)),pi);
    else
        bound = bound + 1;
    end
end

scatt_states = [bound+1:numeigs];
scatter(prad(scatt_states)'/lattsp,phase(scatt_states)*180/pi')
xlabel('momentum (MeV)')
ylabel('phase shift (deg)')

% To check with full Hamiltonian results:

% numeigsfull = 60;
% energiesfree = eigs(Hfree,numeigsfull,'sa');
% energies = eigs(H,numeigsfull,'sa');
```



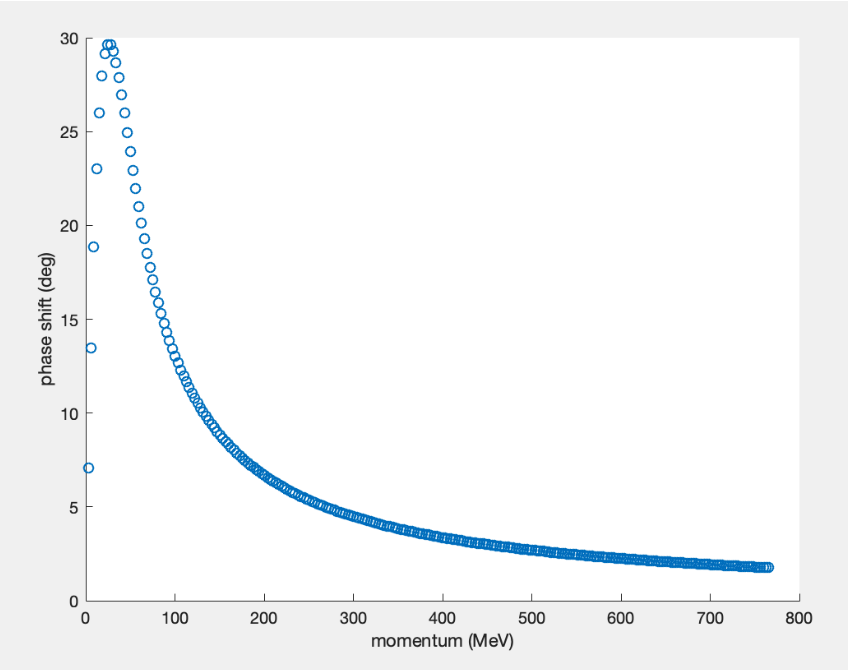
$\ell = 0$
phase shift (deg)
momentum (MeV)
0
5
10
15
20
25
30
0
50
100
150
200
250
300
350


$\ell = 1$
phase shift (deg)
momentum (MeV)
11
10
9
8
7
6
5
4
3
2
1
0
50
100
150
200
250
300
350

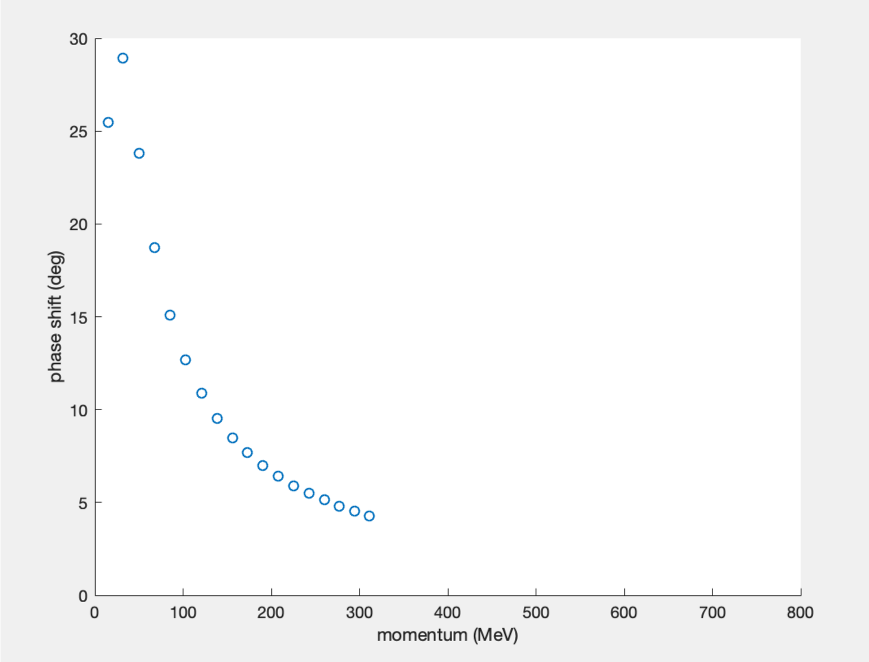


$\ell = 2$
phase shift (deg)
momentum (MeV)
0
1
2
3
4
5
6
7
0
50
100
150
200
250
300
350

$$\ell = 0$$

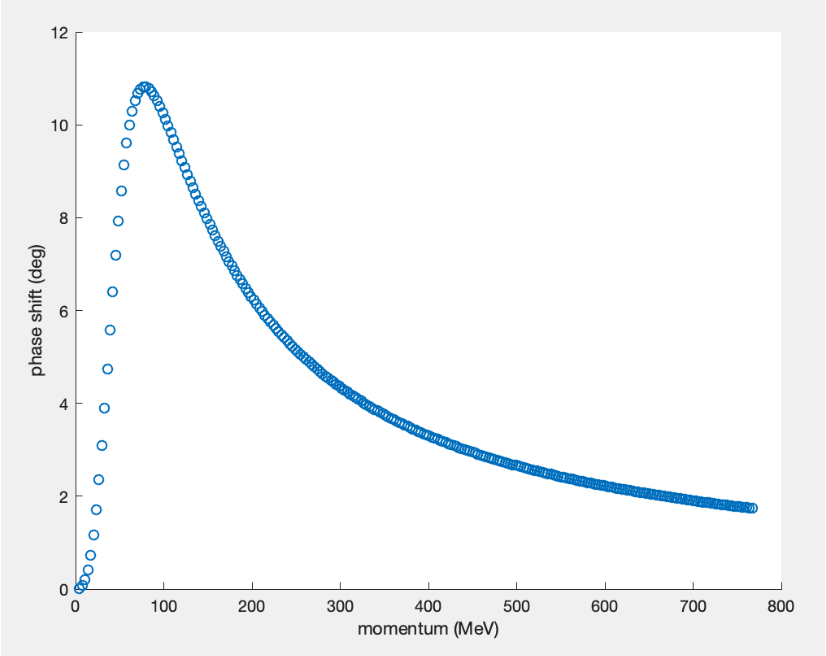


phase shift (deg)
momentum (MeV)
0
100
200
300
400
500
600
700
800
0
5
10
15
20
25
30

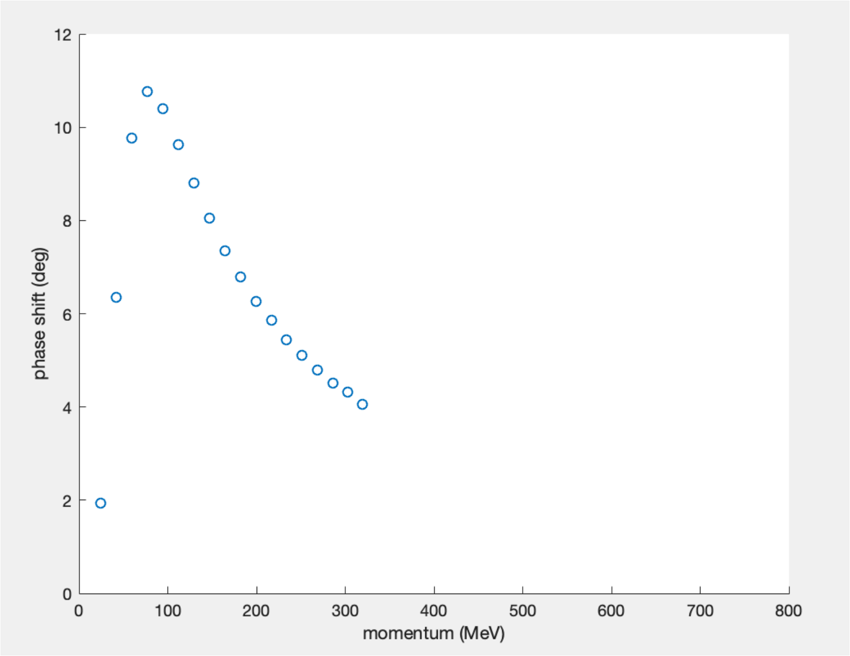


phase shift (deg)
momentum (MeV)
0
100
200
300
400
500
600
700
800
0
5
10
15
20
25
30

$$\ell = 1$$

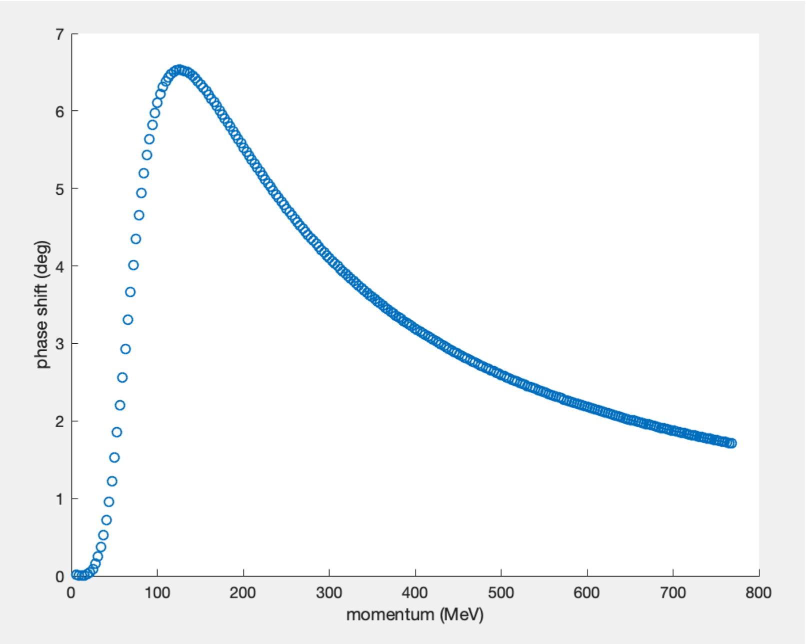


phase shift (deg)
momentum (MeV)
0
2
4
6
8
10
12
0
100
200
300
400
500
600
700
800

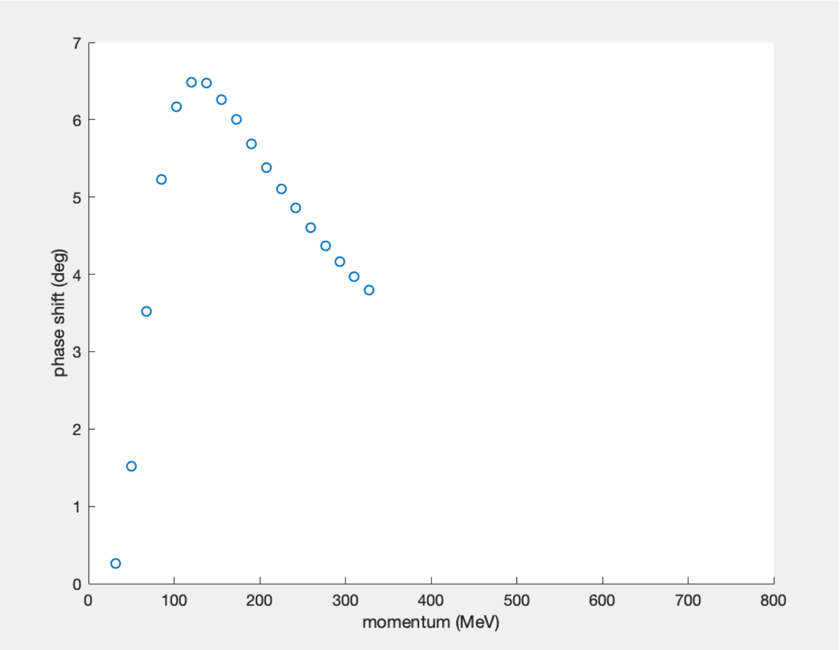


phase shift (deg)
momentum (MeV)
0
2
4
6
8
10
12
0
100
200
300
400
500
600
700
800

$$\ell = 2$$



phase shift (deg)
momentum (MeV)
0
1
2
3
4
5
6
7
0
100
200
300
400
500
600
700
800



phase shift (deg)
momentum (MeV)
0
1
2
3
4
5
6
7
0
100
200
300
400
500
600
700
800

Homework for July 29

Write your own spherical wall code to calculate the phase shifts for the Gaussian potential in continuous space:

$$V(r) = Ce^{-\frac{r^2}{2R_0^2}}$$

$$C = -2 \text{ MeV}, \; R_0 = 2 \times 10^{-2} \text{ MeV}^{-1}$$

$$\mu = m/2, \; m = 938.92 \text{ MeV}$$

$$-\frac{1}{2\mu}\frac{d^2u}{dr^2} + \left[\frac{\ell(\ell+1)}{2\mu r^2} + V(r)\right]u(r) = Eu(r)$$

Homework for July 30

Write your own spherical wall code to calculate the phase shifts for the Gaussian potential on the lattice:

$$V(r) = Ce^{-\frac{r^2}{2R_0^2}}$$

$$C = -2 \text{ MeV}, \; R_0 = 2 \times 10^{-2} \text{ MeV}^{-1}$$

$$\mu = m/2, \; m = 938.92 \text{ MeV}$$

$$-\frac{1}{2\mu}\frac{d^2u}{dr^2} + \left[\frac{\ell(\ell+1)}{2\mu r^2} + V(r)\right]u(r) = Eu(r)$$