

## Appendix A

### Monte Carlo simulation Matlab code

The following pages contain an example of Monte Carlo simulation code used for simulations shown in Chapters 3 and 5. This particular simulation is intended to provide insight into 1-D QNLC for the realistic experimental conditions described in section 3.1.3. In this case, the green quenching pulse is no longer being used to directly assist in the cooling process. This adds an additional random recoil associated with the excitation of the atom from the excited  $^3P_1$  state to the quenching  $^1S_0$  state, before it decays to the ground  $^1P_1$  state. Also, we do not assume in this simulation that we have enough quenching power to quench 100 % of the atoms.

```

%MATLAB 6.5 Monte Carlo simulation cool3.m

%This simulation cycles "num" number of atoms "steps" times, with a cooling sequence of
%a 657 nm cooling pulse from the right, then a 657 nm cooling pulse from the left,
%followed by a 552 nm quenching pulse. Then the sequence is repeated with cooling
%pluses first incident from the left. Switching the pulse direction
%order helps maintain symmetry in the resultant velocity distribution.
%The probability of quenching the excited state is no longer 100%, it is "percent"
%Random blue, IR, and green photon recoil kicks have been included in the simulation due to the
quenching.

clear All % resets all variables

num=100000; %sets number of atoms in simulation
steps=10; %sets number of cooling cycles
percent=0.3; %sets green efficiency 0.3 = 30 %

%To initialize a gaussian distribution with a FWHM 82.13 cm/s
%(vrms = 70 cm/s) for blue-cooled atoms

Halfwidth=82.13*1.2;
for i=1:num
    r1=rand(1);
    if r1 > 0
        V(i)=Halfwidth*sqrt(-log(r1)).*cos(2.*pi.*rand(1));
    else
        V(i)=Halfwidth*sqrt(-log(rand(1))).*cos(2.*pi.*rand(1));
    end
end

[N,Vinitbins]=Hist(V,400); % creates a Histogram of initial velocity distribution
initdata=[Vinitbins',N'];
fn='c:\Data\greeninitdata_30.txt';
dlmwrite(fn,initdata,','); % writes data to file

Rrecoil=1.51; % Recoil velocity due to red 657 nm photon (cm/s)
Grecoil=1.79; % Recoil velocity due to green 552 nm photon (cm/s)
Brecoil=2.34; % Recoil velocity due to blue 423 nm photon (cm/s)
IRrecoil=0.96; % Recoil velocity due to IR 1.03 um photon (cm/s)
Rlambda=657.459e-7; % Wavelength of red light (cm)
k=2*pi/Rlambda; % magnitude of k-vector for red light (cm^-1)
detune=13.14; % detuning of red light from zero velocity (cm/s)
Rabifreq=2*pi*200000; % Red transition Rabi frequency (s^-1)
vzero=detune;
T=2.5e-6; % Length of red cooling pulses (s)

GorE=-1.*ones(size(V)); % This next statement works in place of an if statement to initialize
%all atoms in the ground state, GorE=-1 (excited state is GorE=1)

```

```

for step=1:steps          %How many cooling cycles? steps
  for i=1:num             %How many atoms? num

% light coming from the right FIRST TIME
  if V(i)+vzero == 0 % What if right on resonance?
    GorE(i)=GorE(i).*-1;
    V(i)=V(i)+GorE(i).*Rrecoil;
  else

    GenRabi=sqrt(Rabifreq.^2+(k.*(V(i)+vzero)).^2); %Generalized Rabi frequency
    Probexc=Rabifreq.^2.*((sin(GenRabi.*(T/2)))).^2/GenRabi.^2; %Probability of being excited

    if rand(1) <= Probexc
      %Gets excited
      GorE(i)=GorE(i).*-1;
      %in excited state now, add one recoil to the velocity
      V(i)=V(i)+GorE(i).*Rrecoil;
    %else
      %stays in the ground state
      % V(i)=V(i);
    end
  end
end
%adding in the recoils for decay to the ground state
if GorE(i) == 1 %if the atom is in the excited state
  %this tells us if the green quenching worked for that atom.
  %If it does, then we add all the random recoils.
  %if not, it stays in the excited state.
  if rand(1) < percent
    %setting up some random numbers.
    if rand(1) > 0.5
      blah1 = 1;
    else
      blah1 = -1;
    end
    if rand(1) > 0.5
      blah2 = 1;
    else
      blah2 = -1;
    end
    if rand(1) > 0.5
      blah3 = 1;
    else
      blah3 = -1;
    end
    % Adding the recoil velocities. Assuming green NOT in same
    % direction as red photon and only quenching at "percent"
    V(i)=V(i)+blah3.*rand(1).*Grecoil+blah1.*rand(1).*IRrecoil+blah2.*rand(1).*Brecoil;
    GorE(i)=-1;
  %else

```

```

        %V(i)=V(i) and GorE(i) still is excited
    end
end

% light coming from the left FIRST TIME
if V(i)+vzero == 0
    GorE(i)=GorE(i).*-1;
    V(i)=V(i)-GorE(i).*Rrecoil;
else
    GenRabi=sqrt(Rabifreq.^2+(k.*(V(i)-vzero)).^2);
    % Doppler shift is negative, due to light incident on atoms from opposite direction
    Probexc=Rabifreq.^2.*((sin(GenRabi.*(T/2)))).^2/GenRabi.^2;

    if rand(1) <= Probexc
        GorE(i)=GorE(i).*-1;
        V(i)=V(i)-GorE(i).*Rrecoil;
    %else
    %stays in the ground state
    %V(i)=V(i);
    end
end

if GorE(i) == 1 %in the excited state

    if rand(1) < percent
        if rand(1) > 0.5
            blah1 = 1;
        else
            blah1 = -1;
        end
        if rand(1) > 0.5
            blah2 = 1;
        else
            blah2 = -1;
        end
        if rand(1) > 0.5
            blah3 = 1;
        else
            blah3 = -1;
        end
        V(i)=V(i)+blah3.*rand(1).*Grecoil+blah1.*rand(1).*IRrecoil+blah2.*rand(1).*Brecoil;
        GorE(i)=-1;
    %else
    %V(i)=V(i) and GorE(i) still is excited
    end
end

% light coming from the left SECOND TIME – repeat of last section above
if V(i)+vzero == 0

```

```

GorE(i)=GorE(i).*-1;
V(i)=V(i)-GorE(i).*Rrecoil;
else

GenRabi=sqrt(Rabifreq.^2+(k.*(V(i)-vzero)).^2);
Probexc=Rabifreq.^2.*((sin(GenRabi.*(T/2))).^2/GenRabi.^2);

if rand(1) <= Probexc
    GorE(i)=GorE(i).*-1;
    V(i)=V(i)-GorE(i).*Rrecoil;
%else
%stays in the ground state
%V(i)=V(i);
end
end

if GorE(i) == 1
    if rand(1) < percent
        if rand(1) > 0.5
            blah1 = 1;
        else
            blah1 = -1;
        end
        if rand(1) > 0.5
            blah2 = 1;
        else
            blah2 = -1;
        end
        if rand(1) > 0.5
            blah3 = 1;
        else
            blah3 = -1;
        end
        V(i)=V(i)+blah3.*rand(1).*Grecoil+blah1.*rand(1).*IRrecoil+blah2.*rand(1).*Brecoil;
        GorE(i)=-1;
        %else
        %V(i)=V(i) and GorE(i) still is excited
    end
end

% light coming from the right SECOND TIME - same as first sequence
if V(i)+vzero == 0
    GorE(i)=GorE(i).*-1;
    V(i)=V(i)+GorE(i).*Rrecoil;
else

GenRabi=sqrt(Rabifreq.^2+(k.*(V(i)+vzero)).^2);
Probexc=Rabifreq.^2.*((sin(GenRabi.*(T/2))).^2/GenRabi.^2);

if rand(1) <= Probexc

```

```

    GorE(i)=GorE(i).*-1;
    V(i)=V(i)+GorE(i).*Rrecoil;
    %else
    %stays in the ground state
    %V(i)=V(i);
    end
end

if GorE(i) == 1
    if rand(1) < percent
        if rand(1) > 0.5
            blah1 = 1;
        else
            blah1 = -1;
        end
        if rand(1) > 0.5
            blah2 = 1;
        else
            blah2 = -1;
        end
        if rand(1) > 0.5
            blah3 = 1;
        else
            blah3 = -1;
        end
        V(i)=V(i)+blah3.*rand(1).*Grecoil+blah1.*rand(1).*IRrecoil+blah2.*rand(1).*Brecoil;
        GorE(i)=-1;
        %else
        %V(i)=V(i) and GorE(i) still is excited
    end
end
end
end
% All cooling cycles are now completed

% Long green pulse that sends all the atoms back to the ground
% state with additional recoils.
for i=1:num
    if GorE(i) == 1
        if rand(1) > 0.5
            blah1 = 1;
        else
            blah1 = -1;
        end
        if rand(1) > 0.5
            blah2 = 1;
        else
            blah2 = -1;
        end
        if rand(1) > 0.5

```

```

        blah3 = 1;
    else
        blah3 = -1;
    end
    V(i)=V(i)+blah3.*rand(1).*Grecoil+blah1.*rand(1).*IRrecoil+blah2.*rand(1).*Brecoil;
    GorE(i)=-1;
end
end

Grinc=find(GorE == -1);
length(Grinc)           %Counts the number of atoms in the ground state (should be num)
for i=1:length(Grinc)
    Vgr(i)=V(Grinc(i)); % Makes an array of velocities
end

[N,Vgrbins]=Hist(Vgr,400); %creates of Histogram of velocity distribution with 400 bins.
grdata=[Vgrbins,N'];
fn='c:\Data\greengrdata_30.txt';
dlmwrite(fn,grdata,','); % writes velocity histogram to file

```