Contents

This function evaluates the plasma rise time for given Ry densities

I set n_min as lower decay boundry. States reaching levels <= n_min are considered to decay immediately

```
% den0   = arrays of Ry atom densities for all shells
% vol    = volume of each shell
% n0     = initial PQN
% t_final= final time in ns
% dt     = time steps

% t is time array from 0ns to t_final ns
% nden is matrix with densities of PQN n states.
 %      first 100 values are for first shell, next 100 values for next
 %      shell and so on
% eden - aray with electron densities
% deac - array with densities of deactivated states (n<10)
% Te   - array with electron temperatures

% i.e.: [t,nden,eden,aden,Te,~]=shell_rate_eqn_sim(0.3,50,50);

%save workspace save('den0=1 n0=50, t0=50.mat','t','nden','eden','aden','Te')

function [t,nden,eden,deac_n_min,deac_dr,deac_pd,Te,r_x,r_y,r_z,v_x,v_y,v_z,v,y0]=shell_rate_eqn_sim(den0, rx ,ry, rz ,n0, t_final, single, vectorize)


%%%%%%%%%%%%%%%%%%%%%%%%%
% Initial conditions %
%%%%%%%%%%%%%%%%%%%%%%%%%

N=length(den0); %number of shells

% Set constants and lower cut-off for PQN distribution:
kB = 1.3806504e-23;                 % #m2 kg s-2 K-1;J K-1; 8.62e-5 #eV K-1,
Ry = 2.179872e-18;                  % #J: Rydberg energy in J

kBm=0.0002770924685030197; %k_b / m_NO #um2 ns-2 K-1

firstn=1;
n_min=10;
numlev=100;                         % this is the number of n levels initially considered
deac=0;                             % start with all Rydberg states
nl=(firstn:firstn+numlev-1)';       % array of accessible n levels, from 1 to 100

ns=length(nl);                      %number of n-states

NL=zeros(ns,N);                     %matrix of shells per n level

DEN0=zeros(ns,N);
NDEN=NL*0;
EDEN=zeros(N,1);                    %electron densities at each shell
DEAC=zeros(N,1);
DEAC_DR=zeros(N,1);                 % N + O from NO^+ + e^- at each shell
DEAC_PD=zeros(ns,N);                % N + O from NO^* of each accessible level at each shell
DEAC_N_MIN=zeros(n_min,N);
T_PENNING=zeros(1,N);               %temperature at each shell after penning ionization

ux=zeros(size(rx));
uz=zeros(size(rx));
uy=zeros(size(rx));

volume= 4/3*pi* (rx.*ry.*rz - [0; rx(1:end-1)].*[0; ry(1:end-1)].*[0; rz(1:end-1)]);
%tspan=0:dt:t_final;%linspace(0,t_final,500);


% Sets initial conditions for electron and n-level distributions:
% no initial electrons -> calc. Penning seed electrons (look up Robicheaux)


if vectorize==false
for ii=1:N %loop though all shells
    h=1+(ii-1)*ns;
    k=ii*ns;
    hh=1+(ii-1)*n_min;
    kk=ii*n_min;


    [PF,eden,rden]=penningfraction(n0,den0(ii));
    % Redistributes the Penning partners over lower n's:
    f=@(x)5.95*x.^5;                % This is the penning fraction distribution
    np=firstn:fix(n0/sqrt(2));      % Array of n states allowed after Penn ion
    ind=1:length(np);               % This is the distribution of penning fraction
    nden=nl*0;
```

```
    nden(ind)=eden*f(np/n0)/sum(f(np/n0));  % dividing by the sum normalizes the function

    nden(nl==n0)=rden;              % set n0 to rden

    % Set initial temperature:    (Robicheaux 2005 JPhysB)
    T_PENNING(ii)=(-Ry*den0(ii)/n0^2 + Ry*rden/n0^2 + Ry*sum(nden(ind)./nl(ind).^2) )*1/(3/2*kB*eden); % by energy conservation

    deac=sum(nden(1:n_min));    % allow n<=n_min to decay
    D_DEAC_N_MIN(:,ii)=nden(1:n_min); %
    nden(1:n_min)=zeros(n_min,1);

    NL(:,ii)=nl;       % save values for this shell in arrays
    DEN0(:,ii)=repmat(den0(ii),ns,1);
    NDEN(:,ii)=nden;
    EDEN(ii)=eden;
    DEAC(ii)=deac;
end

else
%same procesure but without for loop, not really helping...

N0=n0*ones(1,N);
[PF,eDen,rDen]=penningfraction(N0,den0);

 f=@(x)5.95*x.^5;                % This is the penning fraction distribution
 np=firstn:fix(n0/sqrt(2));      % Array of n states allowed after Penn ion
 nDen=nl*0;                      % This is the distribution of penning fraction of penning partner (Rydberg molecules)
 ind=1:length(np);
 nDen(ind)=f(np/n0)/sum(f(np/n0));% dividing by the sum normalizes the function
 NDEN=nDen*eDen;                  % eDen is equal to the number of the remaining penning partner
 NDEN(nl==n0,:)=rDen;             % set n0 to rden which is the remaining Rydberg molecules in state n0
 EDEN=eDen';

 T_PENNING=(-Ry*den0./n0^2 + Ry*rDen./n0^2 + Ry*sum(NDEN(ind,:)./nl(ind).^2)).*1./(3/2*kB.*eDen); % by energy conservation, initial before penning is zero

 %DEAC=sum(NDEN(1:n_min,:))';
 D_DEAC_N_MIN=NDEN(1:n_min,:);     % allow n<=n_min to decay, contributes to initial N + 0 from each n<=n_min on each shell
 DEAC=sum(D_DEAC_N_MIN)';          % allow n<=n_min to decay, contributes to initial total N + 0 of each shell
 NDEN(1:n_min,:)=0;
 NL=nl*ones(1,N);                % save values for this shell in arrays
 %DEN0=ones(ns,1)*den0;
end




%the penning temperature is equal for all densities! (density scaling factor cancels)
T_penning = sum(T_PENNING.*(volume').*den0)/sum((volume').*den0); %calculate equlibrated penning temperature by weigted average

T_penning=5;

%%%%%%%%%%%%%%%%%
% Calculations %
%%%%%%%%%%%%%%%%%

y0=[reshape(NDEN,[ns*N,1]);EDEN;DEAC;DEAC_DR;reshape(DEAC_PD,[ns*N,1]);T_penning;rx;ry;rz;ux;uy;uz;volume;reshape(DEAC_N_MIN,[n_min*N,1])]; % set initial value for ODE

ncrit=@(T)round(sqrt(Ry/(kB*T)));  % to calculate n-max (got by fitting)

[ni,nf,II,minn,maxn,diffsn]=buildns(nl);   % is needed to calculate the rate coeffs

function dy=eqrateode(t,y)              %has to be a colume vector
```

```
    % Select valiables
    nden=y(1:ns*N);                     % pick out Rydberg molecule density over distribution of n
    eden=y(ns*N+1:(ns+1)*N);            % pick out electron density at each shell
    deac=y((ns+1)*N+1:(ns+2)*N);        % pick out density of radiatively decayed atoms
    deac_dr=y((ns+2)*N+1:(ns+3)*N);     %       ...           dissociative recombination
    deac_pd=y((ns+3)*N+1:(2*ns+3)*N);   %       ...           predissociation
    T=y((2*ns+3)*N+1);                  % temperature

    RX=y((2*ns+3)*N+2:(2*ns+4)*N+1);    % x radius of each shell
    RY=y((2*ns+4)*N+2:(2*ns+5)*N+1);
    RZ=y((2*ns+5)*N+2:(2*ns+6)*N+1);

    UX=y((2*ns+6)*N+2:(2*ns+7)*N+1);    % x velocity of each shell
    UY=y((2*ns+7)*N+2:(2*ns+8)*N+1);
    UZ=y((2*ns+8)*N+2:(2*ns+9)*N+1);

    V= y((2*ns+9)*N+2:(2*ns+10)*N+1);   % slice of volume between each shell

    deac_n_min= y((2*ns+10)*N+2:end);


    nc=ncrit(T);            % calculates n max with this temperature
%     Adjusts max allowed n:
    if nc>=nl(1) && nc<nl(end)
        index=find(nl==ncrit(T));  %index of max allowed n
    elseif nc<nl(1)
```

```
        index=1;
    else
        index=numlev;
    end

    index=numlev; %deactive thermal criterion

    %preallocate arrays for all shells
    D_NDEN=zeros(ns,N);
    D_NDEN_OLD=zeros(ns,N);
    D_EDEN=zeros(N,1);
    D_DEAC=zeros(N,1);
    D_EDEN_OLD=zeros(N,1);
    D_DEAC_DR=zeros(N,1);
    D_DEAC_PD=zeros(ns,N);
    D_DEAC_N_MIN=zeros(n_min,N);

    %VOL=zeros(ns,N);
    %D_VOL=zeros(ns,N);

    k_n_np=knnp(ni,nf,II,minn,maxn,diffsn,T);   % nl * nl matrix
    kion_one=kION(nl,T); % 1*nl row
    k_tbr_one=kTBR(nl(1:index),T); % index*1  column

    k_CT=100*kTBR(nl(1:index),T);  % This is just an approximation of k_CT of Hydrodynamic recombination
    k_amb=1*kDR(T);        % Loss of NO+ ions to ambipolar expansion,  approximation


    D_RX=UX;
    D_RY=UY;
    D_RZ=UZ;

     diff_rho=diff(eden',1,2)'; %#ok<*UDIM>
     diff_rx=diff(RX',1,2)';
     diff_ry=diff(RY',1,2)';
     diff_rz=diff(RZ',1,2)';

     if single
        dux=zeros(N,1);%*mean((-kBm*T*(diff_rho./eden(1:end-1))./diff_rx)./RX(2:end));
        duy=zeros(N,1);%*mean((-kBm*T*(diff_rho./eden(1:end-1))./diff_ry)./RY(2:end));
        duz=zeros(N,1);%*mean((-kBm*T*(diff_rho./eden(1:end-1))./diff_rz)./RZ(2:end));


        D_UX= zeros(N,1);%*dux*RX; %% added 4 zeros
        D_UY= zeros(N,1);%*duy*RY;
        D_UZ= zeros(N,1);%*duz*RZ;


        D_V=zeros(N,1);%*4/3*pi*( D_RX.*RY.*RZ+RX.*D_RY.*RZ+RX.*RY.*D_RZ - ( [0; D_RX(1:end-1)].*[0; RY(1:end-1)].*[0; RZ(1:end-1)]+[0; RX(1:end-1)].*[0; D_RY(1:end-1)].*[0; RZ(1:e
        D_v=zeros(N,1);
     else
        dux=mean((-kBm*T*(diff_rho./eden(1:end-1))./diff_rx)./RX(2:end)); % eqiuation (15) Rafeal's paper
        duy=mean((-kBm*T*(diff_rho./eden(1:end-1))./diff_ry)./RY(2:end));
        duz=mean((-kBm*T*(diff_rho./eden(1:end-1))./diff_rz)./RZ(2:end));


        D_UX= dux*RX;
        D_UY= duy*RY;
        D_UZ= duz*RZ;

        D_V=4/3*pi*( D_RX.*RY.*RZ+RX.*D_RY.*RZ+RX.*RY.*D_RZ - ( [0; D_RX(1:end-1)].*[0; RY(1:end-1)].*[0; RZ(1:end-1)]+[0; RX(1:end-1)].*[0; D_RY(1:end-1)].*[0; RZ(1:end-1)]+[0; RX
        D_v=D_V./V;
     end




     VOL=repmat(V',ns,1);
     D_VOL=repmat(D_V', ns,1);

%    for r=1:N %loop through all shells
%
%         VOL(:,r)=repmat(V(r),[ns,1])
%         D_VOL(:,r)=repmat(D_V(r), [ns,1]);
%     end

    % Evaluate the updated rate terms:



if vectorize
```

start vectorizing

```
    %this approach is in parellel with for loop approach. The physical meaning of each term is indicated in the for loop below
    nden=reshape(nden,[ns,N]); % density of NO^* Rydberg on ns states and N shells
    deac_pd=reshape(deac_pd, [ns,N]);
    deac_n_min=reshape(deac_n_min,[n_min,N]);

    d_ct=zeros(numlev,N); % charge transfer term with rate k_CT
```

```
    d_ct(1:index,:)= k_CT.*nden(1:index,:).*eden'.^2;  % ns*N matrix with 1:index row nonzero

    d_amb=k_amb*eden;  %term due to ambipolar loss,  N*1 column



    d_tbr=zeros(numlev,N);
    d_tbr(1:index,:)=k_tbr_one*eden'.^3;  % three-body recombination term with rate k_tbr; ns*N matrix with 1:index row nonzero


    d_ion=kion_one.*nden.*eden';   % electron impact ionization term;  ns*N matrix

    d_n_np=sum(k_n_np,2).*nden.*eden'; %collisional l-mixing term with transfering rate from n to n'; ns*N matrix

    k_np_n=k_n_np';                      % ns*ns matrix

    k_np_n(index+1:end, :)=0;

    d_np_n=k_np_n*nden.*eden';      % l-mixing transfer from n' to n; ns*N matrix

    d_n_npion=sum(k_n_np(1:index,index+1:numlev),2)'*nden(1:index,:).*eden';  %  transfer from n's above ncrit(T) to eden;  1*N  times  1*N =  1*N row

    D_DEAC_DR=kDR(T)*eden.^2;       %DISSOCIATIVE RECOMBINATION;  N*1 column

%    D_DEAC_PD=kpd_const.*nden;      % ns*N matrix
    D_DEAC_PD=kpd_slow.*nden;      %  PREDISSOCIATION;  ns*N matrix

    dv=D_v;                      % change of Volume;  N*1 column

 %   D_EDEN_OLD=sum(d_ion-d_tbr)'+d_n_npion';  % N*1 colum
    D_EDEN_OLD=sum(d_ion-d_tbr-d_ct)'+d_n_npion';  % N*1 colum

    D_EDEN=D_EDEN_OLD-D_DEAC_DR-d_amb-eden.*dv; % calculate the total change of electron density in N shell; N*1 column

%    d_nden=d_tbr-d_ion-d_n_np+d_np_n;  % ns*N matrix
    d_nden=d_ct+d_tbr-d_ion-d_n_np+d_np_n;  % calculate the total change of Rydberg molecules on ns states in N shell; ns*N matrix

    D_NDEN_OLD=d_nden;   % % ns*N matrix

    D_DEAC=sum(d_nden(1:n_min,:))';   % N*1 column

    D_DEAC_N_MIN=d_nden(1:n_min,:);   % density change of radiatively decayed atoms;  n_min*N matrix

    d_nden=d_nden-D_DEAC_PD-nden.*dv'; % ns*N matrix

    d_nden(1:n_min,:)=0;       % ns*N matrix

    D_NDEN=d_nden;             % ns*N matrix

    D_DEAC_DR=D_DEAC_DR-deac_dr.*dv;  % density change of dissociative recombination atoms; N*1 column

    D_DEAC_PD=D_DEAC_PD-deac_pd.*dv';  % density change of predissociation atoms;  ns*N matrix

    D_DEAC_N_MIN=D_DEAC_N_MIN-deac_n_min.*dv'; % n_min*N matrix
```

```
else
```

start looping

```
   for z=1:N


        h=1+(z-1)*ns;
        k=z*ns;

        hh=1+(z-1)*n_min;
        kk=z*n_min;

        d_tbr=zeros(numlev,1);
        d_tbr(1:index)=k_tbr_one*eden(z)^3; % units [d_tbr] = um^-3 ns^-1

        d_ion=kion_one.*nden(h:k)*eden(z); % units [d_ion] = um^-3 ns^-1

        % rate for transfer from n to n', unit [kn_np] = um^3 ns^-1

        d_n_np=sum(k_n_np,2).*nden(h:k)*eden(z);             %[um^-3 ns^-1]

        % rate for transfer from n' to n, [knp_n] = ns^-1
        k_np_n=zeros(numlev,numlev);
       % only to levels <= nc
         k_np_n(1:index,1:numlev)=(k_n_np(1:numlev,1:index).*(nden(h:h+numlev-1)))';

        d_np_n=sum(k_np_n,2)*eden(z);                    %[um^-3 ns^-1]

        % transfer from n's above ncrit(T) to eden
        k_n_npion=zeros(numlev,1);
        if index<=numlev
            k_n_npion(1:index)=sum(k_n_np(1:index,index+1:numlev),2).*nden(h:h+index-1); % [kn_npion] = ns^-1
```

```
        end
        d_n_npion=sum(k_n_npion)*eden(z);               %[um^-3 ns^-1]

        %comment to deactivate DISSOCIATIVE RECOMBINATION
        D_DEAC_DR(z)=kDR(T)*eden(z)^2;

        %comment to deactivate PREDISSOCIATION
        D_DEAC_PD(:,z)=kpd_const.*nden(h:k);

        %dv=(D_V(z)/V(z));
        dv=D_v(z);

        D_EDEN_OLD(z)=sum(d_ion-d_tbr)+d_n_npion;
        D_EDEN(z)=D_EDEN_OLD(z)-D_DEAC_DR(z)-eden(z)*dv;

        d_nden=d_tbr-d_ion-d_n_np+d_np_n;
        D_NDEN_OLD(:,z)=d_nden; %array whithout radiative decay for temperature calculation
            % Implements radiative decay/PD for levels n <= n_min:
        D_DEAC(z)=sum(d_nden(1:n_min));     % aden is the number of Ry's decayed radiatively to the groundstate
        D_DEAC_N_MIN(:,z)=d_nden(1:n_min);

        d_nden=d_nden-D_DEAC_PD(:,z)-nden(h:k)*dv; %reduce by predissociation
        d_nden(1:n_min)=zeros(n_min,1);

        D_NDEN(:,z)=d_nden;

        D_DEAC_DR(z)=D_DEAC_DR(z)-deac_dr(z)*dv;
        D_DEAC_PD(:,z)=D_DEAC_PD(:,z)-deac_pd(h:k)*dv;
        D_DEAC_N_MIN(:,z)=D_DEAC_N_MIN(:,z)-deac_n_min(hh:kk)*dv;

    end
```

```
end
```

```
    %now calculate change in equilibrated temperature, use old D_NDEN_OLD
    %value to ensure deactivated states do not affect temperature
    %I guess here a quasi-equalibrium is assumend where dV=0
    dT=(Ry*sum(sum(D_NDEN_OLD./NL.^2.*VOL))-1.5*kB*T*sum(sum(D_EDEN_OLD.*V)) ...
        -sum(sum(0.98263e-20*V.*eden.*(UX.*D_UX + UY.*D_UY + UZ.*D_UZ)))/3  )...
        /(1.5*kB*sum(sum(eden.*V)));

%     Energy=-Ry*sum(nden./(NL.^2).*VOL)+1.5*kB*T*sum(eden.*V)
dy=[reshape(D_NDEN,[ns*N,1]);D_EDEN;D_DEAC;D_DEAC_DR;reshape(D_DEAC_PD,[ns*N,1]);dT;D_RX;D_RY;...
       D_RZ;D_UX;D_UY;D_UZ;D_V;reshape(D_DEAC_N_MIN,[n_min*N,1])];
```

```
end

%%%%%%%
% ODE %
%%%%%%%
kpd_const=kPD(n1);
kpd_slow=0.0*kPD_slow(n1);  % predissociation rate represented by high l Rydberg only

% options=odeset('reltol',1e-8);
% used to be:
% [t,y]=ode23(@(t,y)eqrateode(t,y),tspan,y0);
[t,y]=ode23(@(t,y)eqrateode(t,y),[0 t_final],y0);


nden=y(:,1:ns*N);                   % pick out density over distribution of n
eden=y(:,ns*N+1:(ns+1)*N);          % electron density
% deac=y(:,(ns+1)*N+1:(ns+2)*N);    % density of radiatively decayed atoms
deac_dr=y(:,(ns+2)*N+1:(ns+3)*N);   % dissociative recombination
deac_pd=y(:,(ns+3)*N+1:(2*ns+3)*N); % predissociation
Te=y(:,(2*ns+3)*N+1);               % pick out temperature

r_x=y(:,(2*ns+3)*N+2:(2*ns+4)*N+1);
r_y=y(:,(2*ns+4)*N+2:(2*ns+5)*N+1);
r_z=y(:,(2*ns+5)*N+2:(2*ns+6)*N+1);

v_x=y(:,(2*ns+6)*N+2:(2*ns+7)*N+1);
v_y=y(:,(2*ns+7)*N+2:(2*ns+8)*N+1);
v_z=y(:,(2*ns+8)*N+2:(2*ns+9)*N+1);

v= y(:,(2*ns+9)*N+2:(2*ns+10)*N+1);

deac_n_min=y(:,(2*ns+10)*N+2:end);


end
```

Published with MATLAB® R2020b