```
(* Radiative Quenching *)

(* All values in atomic units *)
(*electron mass *)
m_e = 1;
(* Proton Mass *)
m_p = 1836;
(* Reduced Mass *)
m_μ = m_p / 2;
ℏ = 1;
(*SpeedOfLight *)
c = 137 ;
(* Boltzman constant *)
k_B = 3.167*^-6 ;

(* Compute the eigenvalues A, p from the system of equations *)
(* RawData format R, p, A, E, E + 1/R *)
SetDirectory[NotebookDirectory[]];
vS1RawData = Import["./s1_state.mat"]⟦1⟧;
vS2RawData = Import["./s2_state.mat"]⟦1⟧;
vP2PlusRawData = Import["./p2uPlus_state.mat"]⟦1⟧;
vP2MinusRawData = Import["./p2uMinus_state.mat"]⟦1⟧;

vS1Data = Table[{vS1RawData⟦i⟧⟦1⟧, vS1RawData⟦i⟧⟦5⟧}, {i, 1, Length[vS1RawData]}];
vS2Data = Table[{vS2RawData⟦i⟧⟦1⟧, vS2RawData⟦i⟧⟦5⟧}, {i, 1, Length[vS2RawData]}];
vP2PlusData = Table[
   {vP2PlusRawData⟦i⟧⟦1⟧, vP2PlusRawData⟦i⟧⟦5⟧}, {i, 1, Length[vP2PlusRawData]}];
vP2MinusData = Table[{vP2MinusRawData⟦i⟧⟦1⟧, vP2MinusRawData⟦i⟧⟦5⟧},
   {i, 1, Length[vP2MinusRawData]}];

lastR = vS1Data⟦Length[vS1Data]⟧⟦1⟧ + 1;

(* Extrapolate to R = 10 *)
vS1Data =
  Join[vS1Data, Table[{i, vS1Data⟦Length[vS1Data]⟧⟦2⟧}, {i, lastR, 10, 1}]];
vS2Data = Join[vS2Data, Table[{i, vS2Data⟦Length[vS2Data]⟧⟦2⟧}, {i, lastR, 10, 1}]];
vP2PlusData = Join[vP2PlusData,
   Table[{i, vP2PlusData⟦Length[vP2PlusData]⟧⟦2⟧}, {i, lastR, 10, 1}]];
vP2MinusData = Join[vP2MinusData,
   Table[{i, vP2MinusData⟦Length[vP2MinusData]⟧⟦2⟧}, {i, lastR, 10, 1}]];
```

```
(* 1sg state, potential curve *)
vs1 = Interpolation[
   Transpose[{vS1Data[[All, 1]], vS1Data[[All, 2]]}], InterpolationOrder → 3];
(* 2sg state, potential curve *)
vs2 = Interpolation[
   Transpose[{vS2Data[[All, 1]], vS2Data[[All, 2]]}], InterpolationOrder → 3];
(* 2P+ state *)
vp2p = Interpolation[
   Transpose[{vP2PlusData[[All, 1]], vP2PlusData[[All, 2]]}], InterpolationOrder → 3];
(* 2sg state, potential curve *)
vp2m = Interpolation[Transpose[{vP2MinusData[[All, 1]], vP2MinusData[[All, 2]]}],
   InterpolationOrder → 3];


limit = 5;
Plot[{vs1[r], vs2[r], vp2p[r], vp2m[r]},
 {r, 0.2, limit}, PlotLabels → Automatic, PlotRange → {5, -4}]


(* Compute the Dipole momement D(r) for the single value of R *)
singleDR[r_, p1_, a1_, p2_, a2_, limit_] :=
 Module[{lSol1, mSol1, lSol2, mSol2, norm1, norm2, dInt, ll1, ll2, mm1, mm2},
  (* 1st wavefunction *)
  (* x = λ, y = μ **)
  lSol1 = NDSolve[{(λ^2 - 1) L''[λ] + λ * L'[λ] + (a1 + 2 r λ - p1^2 λ^2) L[λ] == 0 ,
      L[1.00001] == 0, L'[1.00001] == 1 }, L[λ], {λ, 1.00001, limit}][[1]];
  mSol1 = NDSolve[{(1 - μ^2) M''[μ] - μ * M'[μ] - (a1 + p1^2 μ^2) M[μ] == 0,
      M[0] == 1, M'[0] == 0}, M[μ], {μ, -.99999, .99999}][[1]];
  (* 2nd wavefunction *)
  lSol2 = NDSolve[{(λ^2 - 1) L''[λ] + λ * L'[λ] + (a2 + 2 r λ - p2^2 λ^2) L[λ] == 0 ,
      L[1.00001] == 0, L'[1.00001] == 1}, L[λ], {λ, 1.00001, limit}][[1]];
  mSol2 = NDSolve[{(1 - μ^2) M''[μ] - μ * M'[μ] - (a2 + p2^2 μ^2) M[μ] == 0,
      M[0] == 0, M'[0] == 1}, M[μ], {μ, -.99999, .99999}][[1]];

  (* To evaluate NDSolve output at a single point *)
  ll1[x_] := L[λ] /. lSol1 /. λ → x;
  ll2[x_] := L[λ] /. lSol2 /. λ → x;
  mm1[y_] := M[μ] /. mSol1 /. μ → y;
  mm2[y_] := M[μ] /. mSol2 /. μ → y;

  (* Compute <ψ|ψ> = <ψ1s|ψ1s> + <ψ2s|ψ2s> *)
  (* <ψ1s|ψ1s> *)
```

```
  norm1 = NIntegrate[Conjugate[(L[λ] /. lSol1) (M[μ] /. mSol1)] (L[λ] /. lSol1)
     (M[μ] /. mSol1), {μ, -.99999, .99999}, {λ, 1.00001, limit}];
  (* <ψ2s|ψ2s> *)
  norm2 = NIntegrate[Conjugate[(L[λ] /. lSol2) (M[μ] /. mSol2)] (L[λ] /. lSol2)
     (M[μ] /. mSol2), {μ, -.99999, .99999}, {λ, 1.00001, limit}];
  dInt = (r/2) 1/Sqrt[norm1 + norm2] (
     (mm1[-0.99999] × mm2[-0.99999] + mm1[0.99999] × mm2[0.99999]) NIntegrate[
        Conjugate[(L[λ] /. lSol1)] λ (L[λ] /. lSol2), {λ, 1.00001, limit}] +
      NIntegrate[
       Conjugate[(M[μ] /. mSol1)] μ (M[μ] /. mSol2), {μ, -0.99999, 0.99999}]
    );
  {r, (-1) dInt - r }
 ]


GetInputData[state1_, state2_] := Module[{inputData, deltaV},
   Switch[state1,
    "s1",
        Switch [state2,
         "p+",
             deltaV[r_] := Abs[vs1[r] - vp2p[r]];
             inputData = Transpose[
                {vS1RawData[[All, 1]], vS1RawData[[All, 2]],
        vS1RawData[[All, 3]], vP2PlusRawData[[All, 2]], vP2PlusRawData[[All, 3]]}],
         "p-",
             deltaV = Abs[vs1[r] - vp2m[r]];
             inputData = Transpose[
                {vS1RawData[[All, 1]], vS1RawData[[All, 2]], vS1RawData[[All, 3]],
        vP2MinusRawData[[All, 2]], vP2MinusRawData[[All, 3]]}],
         _, Throw["Invalid State"]],
    "s2",
        Switch [state2,
         "p+",
             deltaV[r_] := Abs[vs2[r] - vp2p[r]];
             inputData = Transpose[
                {vS2RawData[[All, 1]], vS2RawData[[All, 2]],
        vS2RawData[[All, 3]], vP2PlusRawData[[All, 2]], vP2PlusRawData[[All, 3]]}],
         "p-",
             deltaV = Abs[vs2[r] - vp2m[r]];
             inputData = Transpose[
                {vS2RawData[[All, 1]], vS2RawData[[All, 2]], vS2RawData[[All, 3]],
        vP2MinusRawData[[All, 2]], vP2MinusRawData[[All, 3]]}],
```

```
          _, Throw["Invalid State"]],
     _, Throw["Invalid State"]];
    {inputData, deltaV}
   ];

{inputDataPlus, deltaV} = GetInputData["s1", "p+"];
{inputDataMinus, deltaV} = GetInputData["s1", "p-"];
```

In[ ]:=

```
CalcDR[inputData_, limit_] := Module[{allDR, dR},
    allDR = Map[singleDR[#[[1]], #[[2]], #[[3]], #[[4]], #[[5]], limit] &, inputData];
    dR = Interpolation[
      Transpose[{allDR[[All, 1]], allDR[[All, 2]]}], InterpolationOrder → 3];
     dR
   ];

dRPlus = CalcDR[inputDataPlus, limit];
dRMinus = CalcDR[inputDataMinus, limit];
```

In[ ]:=

```
(*Plot[{dRPlus[r],dRMinus[r]},{r,.2,5}, PlotLabels→{"2p+ -> 1s+","2p+ -> 1s+"},
 Frame→True,FrameLabel→{"R (units of a_u)", "D(R) (units of a_u}"} ]*)
Plot[{dRPlus[r]}, {r, .2, 5}, Frame → True,
 FrameLabel → {"R (units of a_u)", "D(R) (units of a_u}"} ]
```

```
aR[x_, state1_, state2_, limit_] := Module[{inputData, deltaV, dR, dV},
    {inputData, dV} = GetInputData[state1, state2];
    dR = CalcDR[inputData, limit];
    deltaV[r_] := dV[r];
    4/3 (dR[x])^2 (deltaV[x])^3/c^3
   ];
```

```
(* Energy  Range, for temperatures from 0K - 2K
   k_B T = 1/2 m v^2 = E, t in Kelvin  *)

energies = Interpolation[ Table[k_B t, {t, 0, 20, .5}], InterpolationOrder → 3];
k_a[r_] := Sqrt[2 m_μ (energies[r] - v_s1[10])];

solF[j_] := Sort[Transpose[NDEigensystem[
      1/r D[r D[ff[r], r], r] - (v_s1[r] + j^2/r^2) ff[r], ff, {r, 0.2, limit}, 5,
      Method → {"SpatialDiscretization" →
         {"FiniteElement", {"MeshOptions" → {MaxCellMeasure → 0.001}}}}]]][[1]];


(* 2D Schrodinger equation, m is a paramete #  (called J in 3D),
grab the lowest eigenvalue, i.e. value of k *)
solS1J = Sort[Transpose[NDEigensystem[
       1/r D[r D[ff[r], r], r] - (v_s1[r] + #^2/r^2) ff[r], ff, {r, 0.2, limit}, 5,
       Method → {"SpatialDiscretization" →
          {"FiniteElement", {"MeshOptions" → {MaxCellMeasure → 0.001}}}}]]][[1]] &;
(*solS2J[j_]:=Sort[Transpose[
    NDEigensystem[1/r D[r D[ff[r],r],r] -(v_s2[r]+j^2/r^2)ff[r],ff[r],{r,0.2, limit}, 5,
     Method→{"SpatialDiscretization"→
        {"FiniteElement",{"MeshOptions"→{MaxCellMeasure→0.001}}}}]]];*)
(* Here m = 0 for 1S state *)
allS1Js = Map[solS1J, Table[i, {i, 0, 0}]];
allS1Ks =  allS1Js[[All, 1]];
allS1JFunc = allS1Js[[All, 2]] ;
k_a = allS1Ks[[1]]

fS1[j_, x_] := allS1JFunc[[j]][x];
(* Calculate the integral *)

 η_s1pp[j_] := NIntegrate[(Abs[ fS1[j,x]])^2 aR[x,"s1", "p+", limit],{x,.2,limit}];
ScientificForm[η_s1pp[1]]
```

```
ηs1pm[j_] :=
  NIntegrate[(Abs[ allS1JFunc[[j]][x] ])^2 aR[x, "s1", "p-", limit], {x, .2, limit}];

ηs2pp[j_] :=
  NIntegrate[(Abs[allS2JFunc[[j]][x]])^2 aR[x, "s2", "p+", limit], {x, .2, limit}];

ηs2pm[j_] :=
  NIntegrate[(Abs[allS2JFunc[[j]][x]])^2 aR[x, "s2", "p-", limit], {x, .2, limit}];


(* Cross Section *)

σ = π/kA^2 Sum[1 - Exp[-4 ηspp[j]], {j, 1, 10}];
σ
```