Photoluminescence: data analysis

Load data

```
clear;
RhoData = ParsePLdata('rhodamine', 'photonE');  % Rhodamine

Temps = 10:10:290;
RubyData = cell(numel(Temps),1);    % Ruby
for it = 1:numel(Temps)
    RubyData{it} = ParsePLdata('ruby', Temps(it), 'photonE');
end

RubyRoomData = ParsePLdata('rubyRtemp', 'photonE');
```

Define loss function for fitting

```
function loss = peak_sensitive_loss(yhat, y)
    delta = max(y) * 1e-3;
    loss = mean(((yhat - y).^2) ./ (abs(y) + delta) .* (y>300 | (y<300 &
yhat>300)));
end
loss_func = @peak_sensitive_loss;
```

Define R-square function for fitting

```
function Rsq = Rsqcal(fitted, experimental)
%RSQCAL   Compute coefficient of determination (R^2)
%   Rsq = Rsqcal(fitted, experimental) returns the R^2 value between
%   the predicted data in fitted and the observed data in experimental.
%
%   R^2 = 1 - SS_res / SS_tot
%     where SS_res = sum((y_obs - y_fit).^2)
%           SS_tot = sum((y_obs - mean(y_obs)).^2)
%
%   Inputs must be vectors of the same length.

    % Ensure inputs are column vectors
    f = fitted(:);
    y = experimental(:);

    % Check sizes
    if numel(f) ~= numel(y)
        error('Rsqcal:InputSizeMismatch', ...
              '''fitted'' and ''experimental'' must have the same number of
elements.');
    end
```

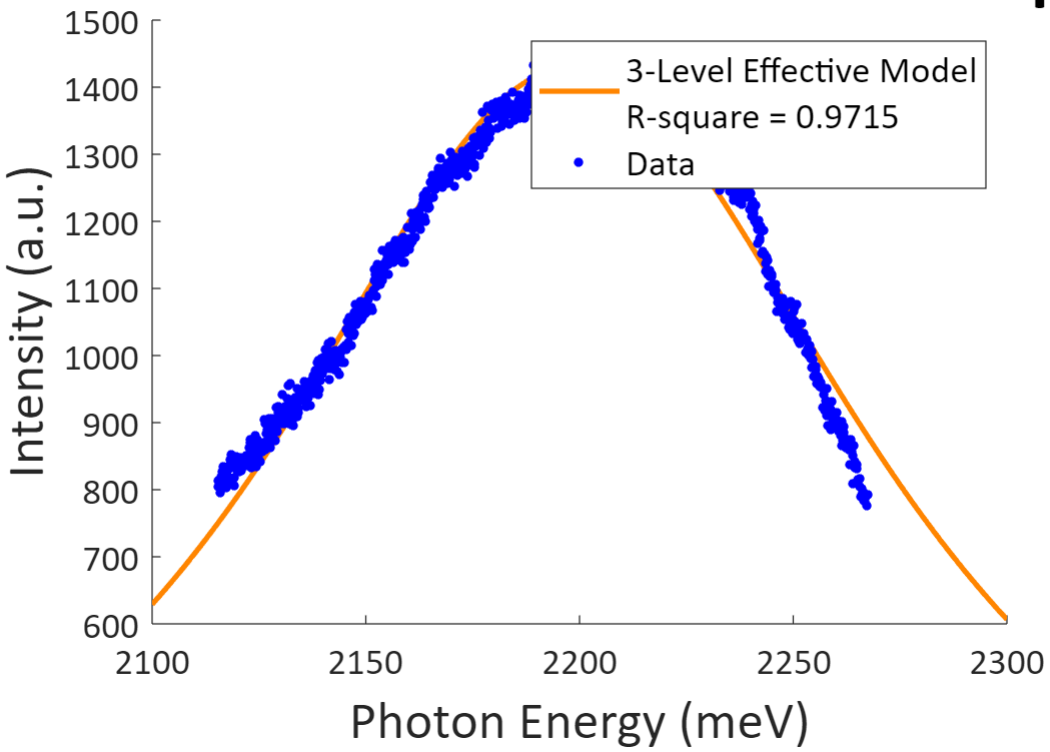
```
    % Total sum of squares
    ymean = mean(y);
    SS_tot = sum((y - ymean).^2);

    if SS_tot == 0
        % all y are identical → variance zero → R^2 undefined
        warning('Rsqcal:ZeroVariance', ...
                'All experimental values are identical; R^2 is undefined. Returning
NaN.');
        Rsq = NaN;
        return;
    end

    % Residual sum of squares
    SS_res = sum((y - f).^2);

    % Coefficient of determination
    Rsq = 1 - SS_res/SS_tot;
end
```

Fit & Plot Rhodamine Data to 3-level effective model (single Lorentzian)

```
% photon energy range
Erange = [2100, 2300];
Energy = linspace(Erange(1), Erange(2), 1000);

% lower and upper bounds of fitting parameters
LowerBound = [1e3, -5, 2150];
UpperBound = [1e6, 3, 2250];

% choose data near the peaks
Idx = RhoData(:,1) > Erange(1) & RhoData(:,1) < Erange(2);
E_data = RhoData(Idx,1);
I_data = RhoData(Idx,2);

% make parameter bounds and loss function into a single struct variable
options.lb = LowerBound;
options.ub = UpperBound;
options.loss_type = loss_func;

% define fitting curve
FitModel = @(Params, Energies) Spec_3lev(Params(1), power(10, Params(2)),
Params(3), Energies);

% iterate to find best fit
best_loss = Inf;
for it = 1:3
    options.rng_seed = it;          % Try different seeds
    Params0 = rand(1,3) .* (UpperBound - LowerBound) + LowerBound;      % try
different initial parameters
```

```
    [params_i, loss_hist_i] = de_curve_fit(FitModel, E_data, I_data, Params0,
options);     % differential evolution fitting
    if loss_hist_i(end) < best_loss
        best_loss = loss_hist_i(end);
        FitParams = params_i;
        loss_history = loss_hist_i;
    end
end

I0 = FitParams(1);
Linewidth = power(10, FitParams(2));
Ecen = FitParams(3);

% plot
figure;
hold on;
I_fit = Spec_3lev(I0, Linewidth, Ecen, Energy);
I_fit_sameidx = Spec_3lev(I0, Linewidth, Ecen, E_data);
Rsq = Rsqcal(I_fit_sameidx, I_data);
plot(Energy, I_fit, 'color', '#FF8500', 'LineWidth', 2);
plot(E_data, I_data, 'o', 'MarkerEdgeColor', 'blue', 'MarkerFaceColor', 'blue',
'MarkerSize', 3);
ax = gca;
ax.FontSize = 14;
ax.FontName = 'Calibri';
xlabel('Photon Energy (meV)', 'FontName', 'Calibri', 'FontSize', 20);
ylabel('Intensity (a.u.)', 'FontName', 'Calibri', 'FontSize', 20);
title('Rhodamine 590 Photoluminescence Spectrum', 'FontName', 'Calibri',
'FontSize', 28);
lg = legend({sprintf('3-Level Effective Model\nR-square = %.4f', Rsq),'Data'},
'Location','northeast', 'FontName', 'Calibri', 'FontSize', 14);
```

damine 590 Photoluminescence Spe




```
pos = lg.Position;
% choose a point just below the legend box
x_txt = 0.900;
y_txt = 0.800;       % tweak the 0.05 to move it closer/further

% show fit parameters
disp(['Loss = ', sprintf('%.4g', best_loss)]);
```

```
Loss = 1.151
```

```
disp(['I0 = ', sprintf('%.4g', I0)]);
```

```
I0 = 1.247e+05
```

```
disp(['Linewidth = ', sprintf('%.4g', Linewidth(1))]);
```

```
Linewidth = 174.4
```

```
disp(['Ecen = ', sprintf('%.4g', Ecen)]);
```

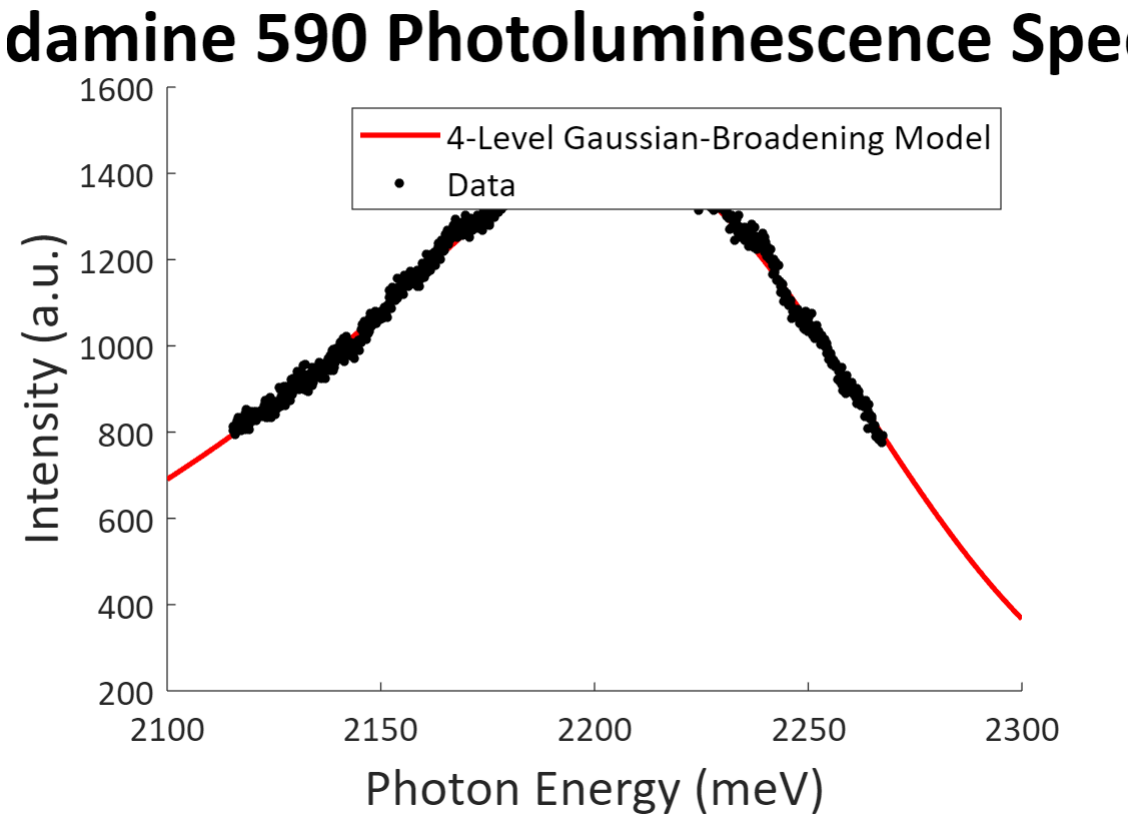
```
Ecen = 2198
```

Fit & Plot Rhodamine Data to Double Gaussian Model

```
% broaden option
Broaden = 'Gauss';

% photon energy range
```

```
Erange = [2100, 2300];
Energy = linspace(Erange(1), Erange(2), 1000);

% lower and upper bounds of fitting parameters
LowerBound = [10, 0, -5, -5, 1e-2, 2150];
UpperBound = [1e6, 10, 3, 3, 1e2, 2250];

% fit data
T = 290;    % temperature

% choose data near the peaks
Idx = RhoData(:,1) > Erange(1) & RhoData(:,1) < Erange(2);
E_data = RhoData(Idx,1);
I_data = RhoData(Idx,2);

% make parameter bounds and loss function into a single struct variable
options.lb = LowerBound;
options.ub = UpperBound;
options.loss_type = loss_func;

% define fitting curve
FitModel = @(Params, Energies) Spec_4lev(Params(1), Params(2), [power(10,
Params(3)), power(10, Params(4))], Params(5), Params(6), T, Energies, 'Broaden',
Broaden);

% iterate to find best fit
best_loss = Inf;
for it = 1:3
    options.rng_seed = it;          % Try different seeds
    Params0 = rand(1,6) .* (UpperBound - LowerBound) + LowerBound;      % try
different initial parameters
    [params_i, loss_hist_i] = de_curve_fit(FitModel, E_data, I_data, Params0,
options);     % differential evolution fitting
    if loss_hist_i(end) < best_loss
        best_loss = loss_hist_i(end);
        FitParams = params_i;
        loss_history = loss_hist_i;
    end
end

% show fit parameters
I0 = FitParams(1);
TransAmpR = FitParams(2);
Linewidth = [power(10, FitParams(3)), power(10, FitParams(4))];
Delta = FitParams(5);
E1 = FitParams(6);

% plot
figure;
hold on;
```

```
I_fit = Spec_4lev(I0, TransAmpR, Linewidth, Delta, E1, T, Energy, 'Broaden',
Broaden);
plot(Energy, I_fit, 'color', 'red', 'LineWidth', 2);
plot(E_data, I_data, 'o', 'MarkerEdgeColor', 'black', 'MarkerFaceColor', 'black',
'MarkerSize', 3);
ax = gca;
ax.FontSize = 14;
ax.FontName = 'Calibri';
xlabel('Photon Energy (meV)', 'FontName', 'Calibri', 'FontSize', 20);
ylabel('Intensity (a.u.)', 'FontName', 'Calibri', 'FontSize', 20);
title('Rhodamine 590 Photoluminescence Spectrum', 'FontName', 'Calibri',
'FontSize', 28);
legend({'4-Level Gaussian-Broadening Model','Data'}, 'Location','northeast',
'FontName', 'Calibri', 'FontSize', 14);

hold off;
```




```
% show fit parameters
disp(['Temperature = ', sprintf('%d', T), ' K']);
```

```
Temperature = 290 K
```

```
disp(['Loss = ', sprintf('%.4g', best_loss)]);
```

```
Loss = 0.2881
```

```
disp(['I0 = ', sprintf('%.4g', I0)]);
```

```
I0 = 1.678e+05
```

```
disp(['d_{20}/d_{10} = ', sprintf('%.4g', TransAmpR)]);
```

```
d_{20}/d_{10} = 2.938
```

```
disp(['Linewidths = [', sprintf('%.4g', Linewidth(1)), ', ', sprintf('%.4g',
Linewidth(2)) ']']);
```

```
Linewidths = [88.43, 48.27]
```

```
disp(['Delta = ', sprintf('%.4g', Delta)]);
```

```
Delta = 66.51
```

```
disp(['E1 = ', sprintf('%.4g', E1)]);
```

```
E1 = 2150
```

Fit Ruby Data to 4-Level Effective Model (T = 10K - 290K, double Lorentzian)

```
% photon energy range
Erange = [1785, 1795];
Energy = linspace(Erange(1), Erange(2), 1000);

% lower and upper bounds of fitting parameters
LowerBound = [2000, 0.5, -0.9, -0.8, 3.5, 1785];
UpperBound = [3400, 3, -0.2, -0.35, 4, 1795];

% variables to store fit parameters
I0 = zeros(29,1);
TransAmpR = zeros(29,1);
Linewidth = zeros(29,2);
Delta = zeros(29,1);
E1 = zeros(29,1);

% fit all 29 data
for Dnum = 1:29

    T = 10*Dnum;    % temperature

    % choose data near the peaks
    Idx = RubyData{Dnum}(:,1) > Erange(1) & RubyData{Dnum}(:,1) < Erange(2);
    E_data = RubyData{Dnum}(Idx,1);
    I_data = RubyData{Dnum}(Idx,2);

    % make parameter bounds and loss function into a single struct variable
    options.lb = LowerBound;
    options.ub = UpperBound;
    options.loss_type = loss_func;

    % define fitting curve
```

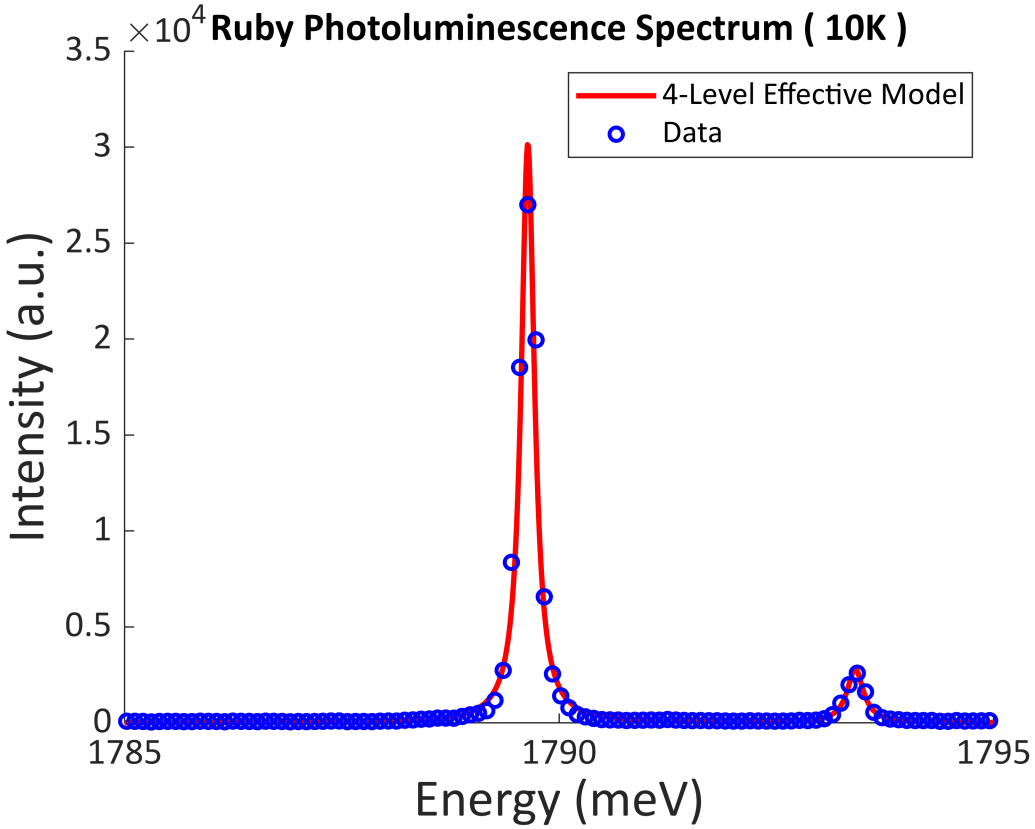
```
    FitModel = @(Params, Energies) Spec_4lev(Params(1), Params(2), [power(10,
Params(3)), power(10, Params(4))], Params(5), Params(6), T, Energies);

    % iterate to find best fit
    best_loss = Inf;
    for it = 1:3
        options.rng_seed = it;          % Try different seeds
        Params0 = rand(1,6) .* (UpperBound - LowerBound) + LowerBound;      % try
different initial parameters
        [params_i, loss_hist_i] = de_curve_fit(FitModel, E_data, I_data, Params0,
options);     % differential evolution fitting
        if loss_hist_i(end) < best_loss
            best_loss = loss_hist_i(end);
            FitParams = params_i;
            loss_history = loss_hist_i;
        end
    end

    I0(Dnum) = FitParams(1);
    TransAmpR(Dnum) = FitParams(2);
    Linewidth(Dnum,:) = [power(10, FitParams(3)), power(10, FitParams(4))];
    Delta(Dnum) = FitParams(5);
    E1(Dnum) = FitParams(6);

    % show message
    disptime(['Data #', sprintf('%d', Dnum), ' | loss = ', sprintf('%.4g',
best_loss)]);

end
```

```
25-06-10 11:02:24 | Data #1 | loss = 49.98
25-06-10 11:02:28 | Data #2 | loss = 43.87
25-06-10 11:02:33 | Data #3 | loss = 39.89
25-06-10 11:02:37 | Data #4 | loss = 39.45
25-06-10 11:02:41 | Data #5 | loss = 30.66
25-06-10 11:02:46 | Data #6 | loss = 27.68
25-06-10 11:02:58 | Data #7 | loss = 25.03
25-06-10 11:03:10 | Data #8 | loss = 20.75
25-06-10 11:03:23 | Data #9 | loss = 20.75
25-06-10 11:03:35 | Data #10 | loss = 13.79
25-06-10 11:03:48 | Data #11 | loss = 16.85
25-06-10 11:04:01 | Data #12 | loss = 18.88
25-06-10 11:04:12 | Data #13 | loss = 17.28
25-06-10 11:04:21 | Data #14 | loss = 17.82
25-06-10 11:04:25 | Data #15 | loss = 17.82
25-06-10 11:04:29 | Data #16 | loss = 20.43
25-06-10 11:04:35 | Data #17 | loss = 23.6
25-06-10 11:04:49 | Data #18 | loss = 21.36
25-06-10 11:05:02 | Data #19 | loss = 7.19
25-06-10 11:05:14 | Data #20 | loss = 21.98
25-06-10 11:05:27 | Data #21 | loss = 20.05
25-06-10 11:05:39 | Data #22 | loss = 25.05
25-06-10 11:05:49 | Data #23 | loss = 8.6
25-06-10 11:05:53 | Data #24 | loss = 7.995
25-06-10 11:05:56 | Data #25 | loss = 26.73
25-06-10 11:06:01 | Data #26 | loss = 22.84
```

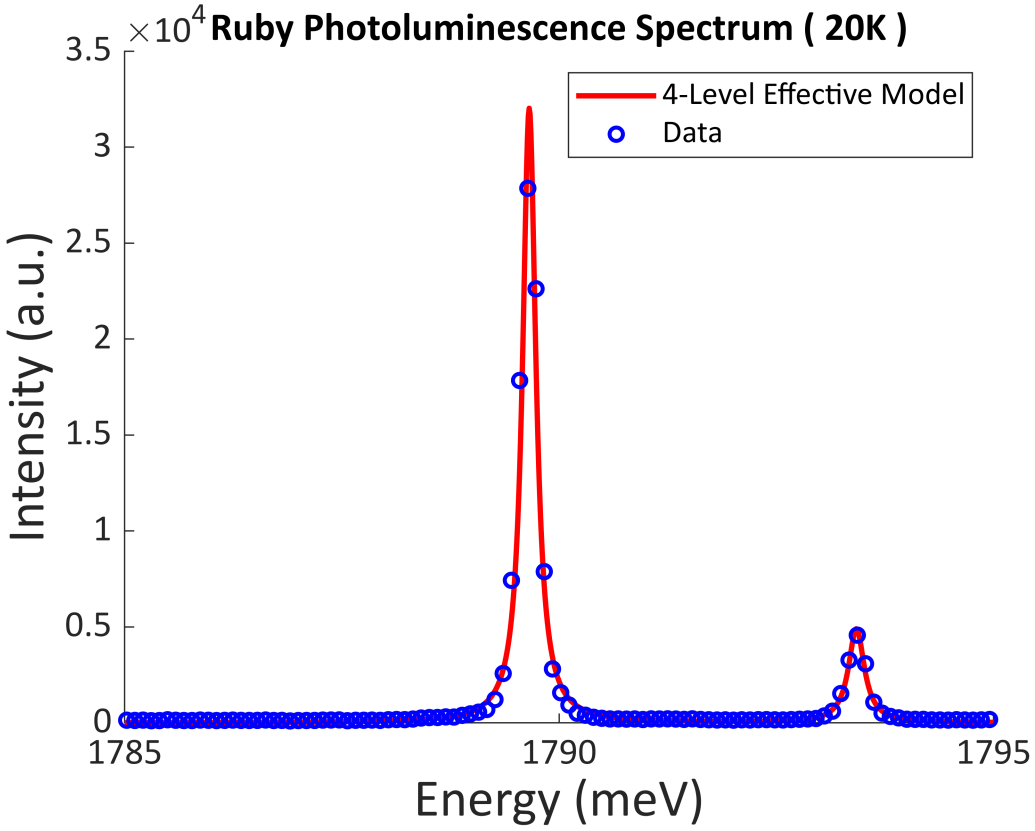
```
25-06-10 11:06:07 | Data #27 | loss = 27.09
25-06-10 11:06:11 | Data #28 | loss = 2.582
25-06-10 11:06:14 | Data #29 | loss = 26.43
```

Plot Ruby Data with Fit Curve

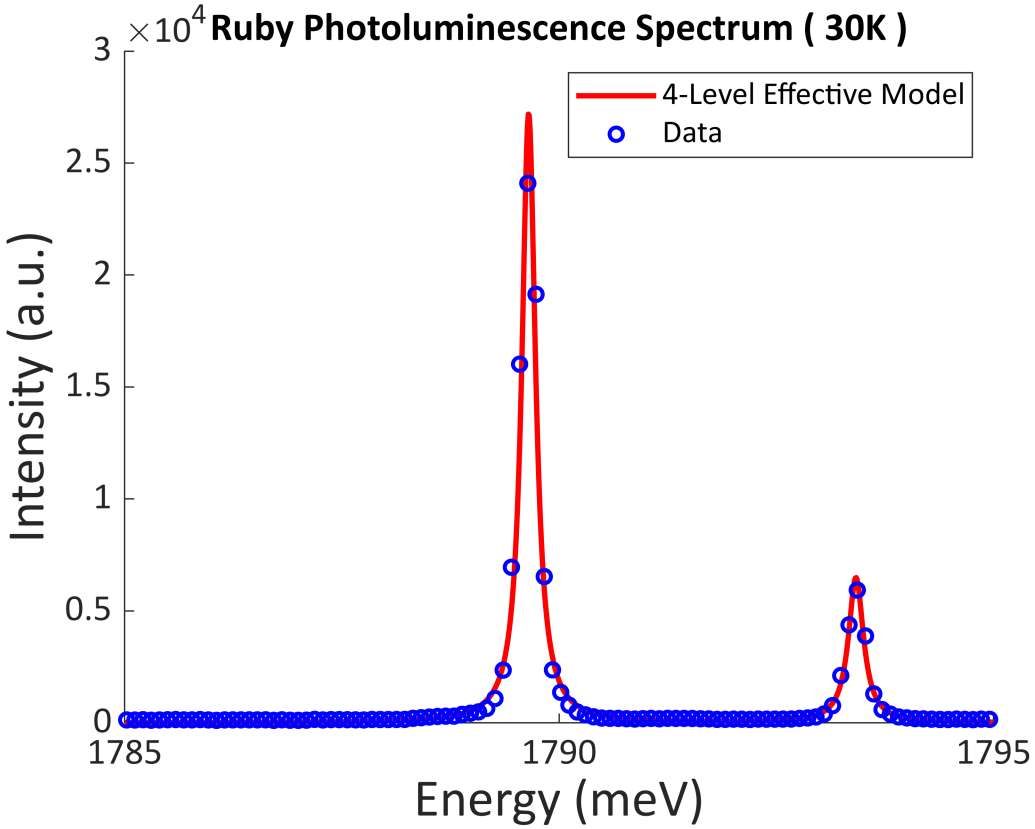
```
for Dnum = 1:29

    T = 10*Dnum;    % temperature

    % choose data near the peaks
    Idx = RubyData{Dnum}(:,1) > Erange(1) & RubyData{Dnum}(:,1) < Erange(2);
    E_data = RubyData{Dnum}(Idx,1);
    I_data = RubyData{Dnum}(Idx,2);

    % plot
    figure;
    hold on;
    I_fit = Spec_4lev(I0(Dnum), TransAmpR(Dnum), Linewidth(Dnum,:), Delta(Dnum),
E1(Dnum), T, Energy);
    plot(Energy, I_fit, 'color', 'red', 'LineWidth', 2);
    plot(E_data, I_data, 'o', 'color', 'blue', 'Markersize', 5, 'LineWidth', 1.5);
    ax = gca;
    ax.FontSize = 14;
    ax.FontName = 'Calibri';
    xlabel('Energy (meV)', 'FontName', 'Calibri', 'FontSize', 20);
    ylabel('Intensity (a.u.)', 'FontName', 'Calibri', 'FontSize', 20);
    title(['Ruby Photoluminescence Spectrum ( ', sprintf('%d', T), 'K )'],
'FontName', 'Calibri', 'FontSize', 15);
    legend({'4-Level Effective Model', 'Data'}, 'Location','northeast', 'FontName',
'Calibri', 'FontSize', 12);
    hold off;

    % show fit parameters
    disp('Fit Parameters :')
    disp(['T = ', sprintf('%d', T), ' K']);
    disp(['I0 = ', sprintf('%.6g', I0(Dnum))]);
    disp(['d_{20}/d_{10} = ', sprintf('%.6g', TransAmpR(Dnum))]);
    disp(['Linewidths = [', sprintf('%.6g', Linewidth(Dnum,1)), ', ',
sprintf('%.6g', Linewidth(Dnum,2)) ']']);
    disp(['Delta = ', sprintf('%.6g', Delta(Dnum))]);
    disp(['E1 = ', sprintf('%.6g', E1(Dnum))]);
end
```



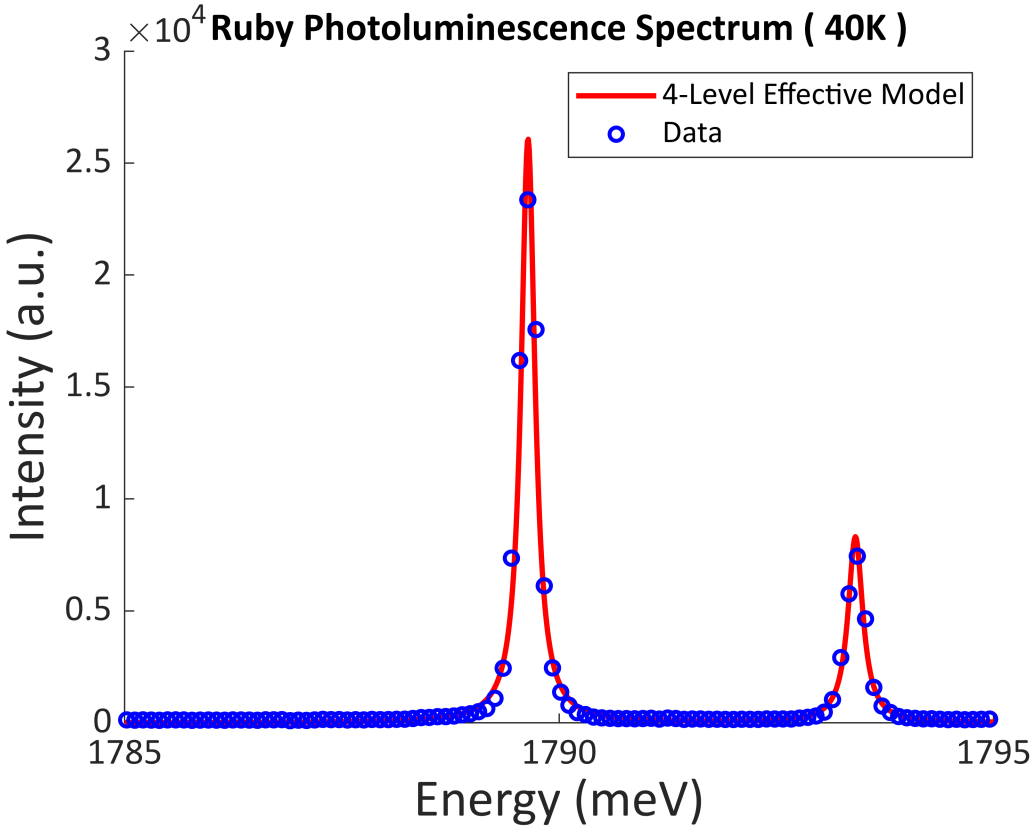

```
Fit Parameters :
T = 10 K
I0 = 2799.65
d_{20}/d_{10} = 2.99441
Linewidths = [0.185489, 0.227227]
Delta = 3.76835
E1 = 1789.64
```



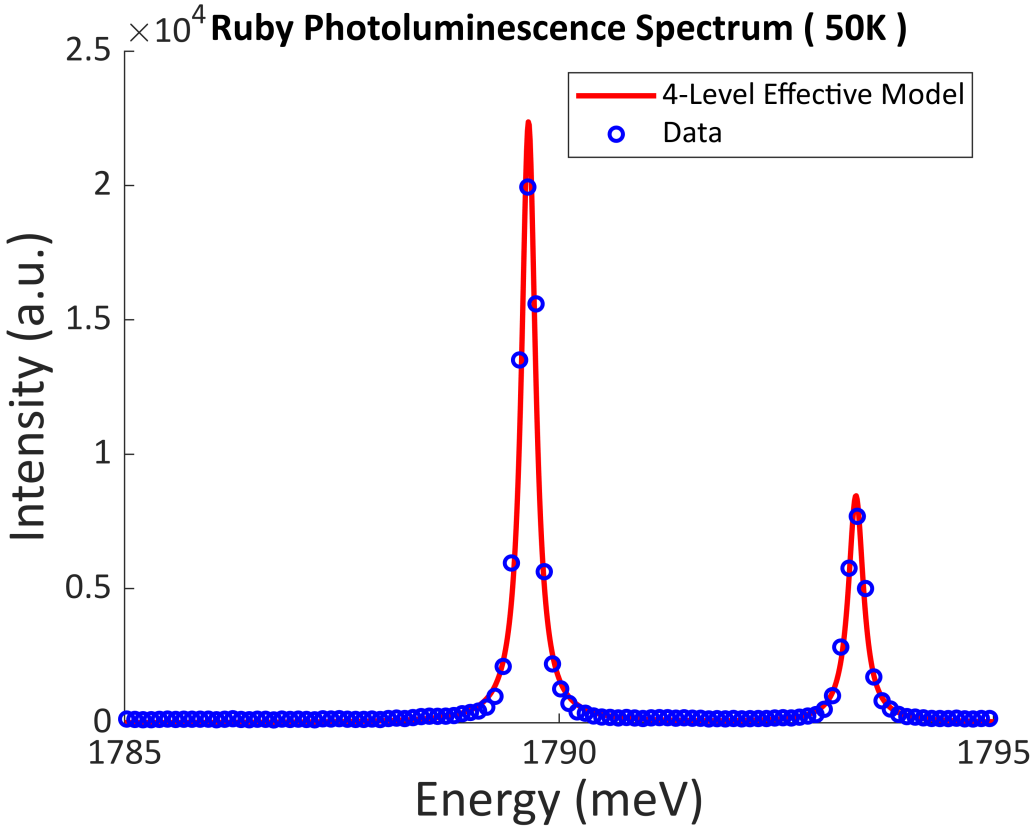

```
Fit Parameters :
T = 20 K
I0 = 2959.91
d_{20}/d_{10} = 1.27405
Linewidths = [0.184879, 0.220937]
Delta = 3.7652
E1 = 1789.66
```



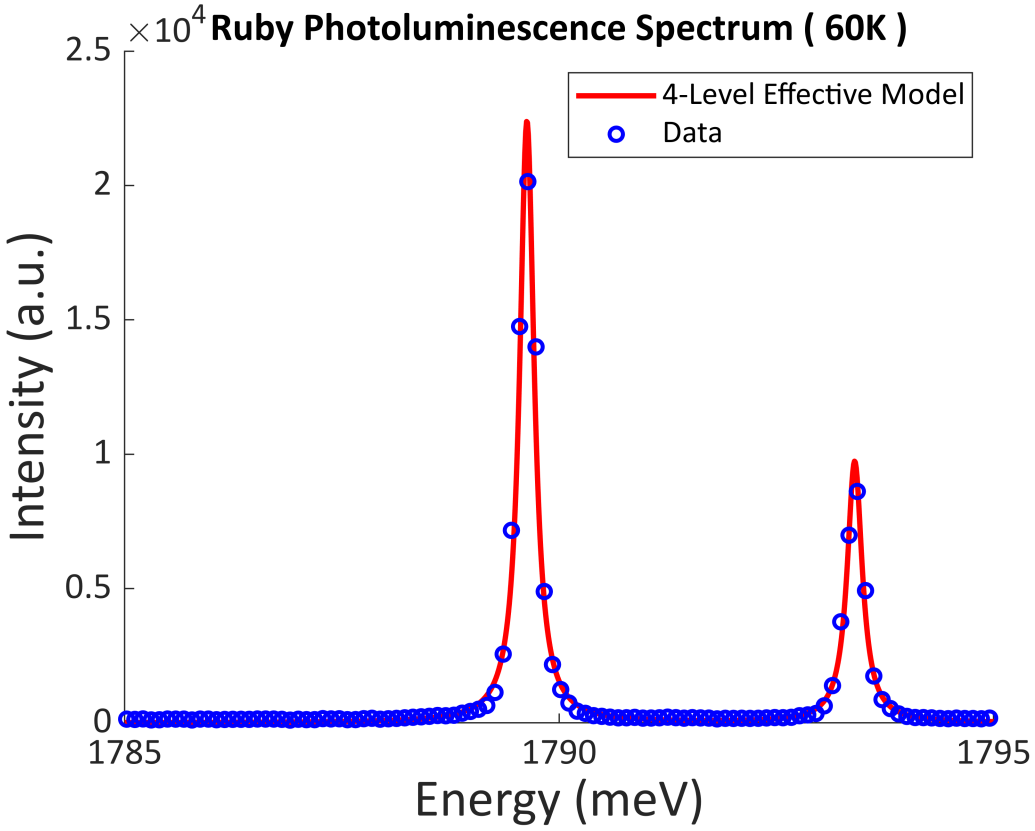

```
Fit Parameters :
T = 30 K
I0 = 2589.38
d_{20}/d_{10} = 1.06497
Linewidths = [0.190166, 0.211084]
Delta = 3.76654
E1 = 1789.65
```



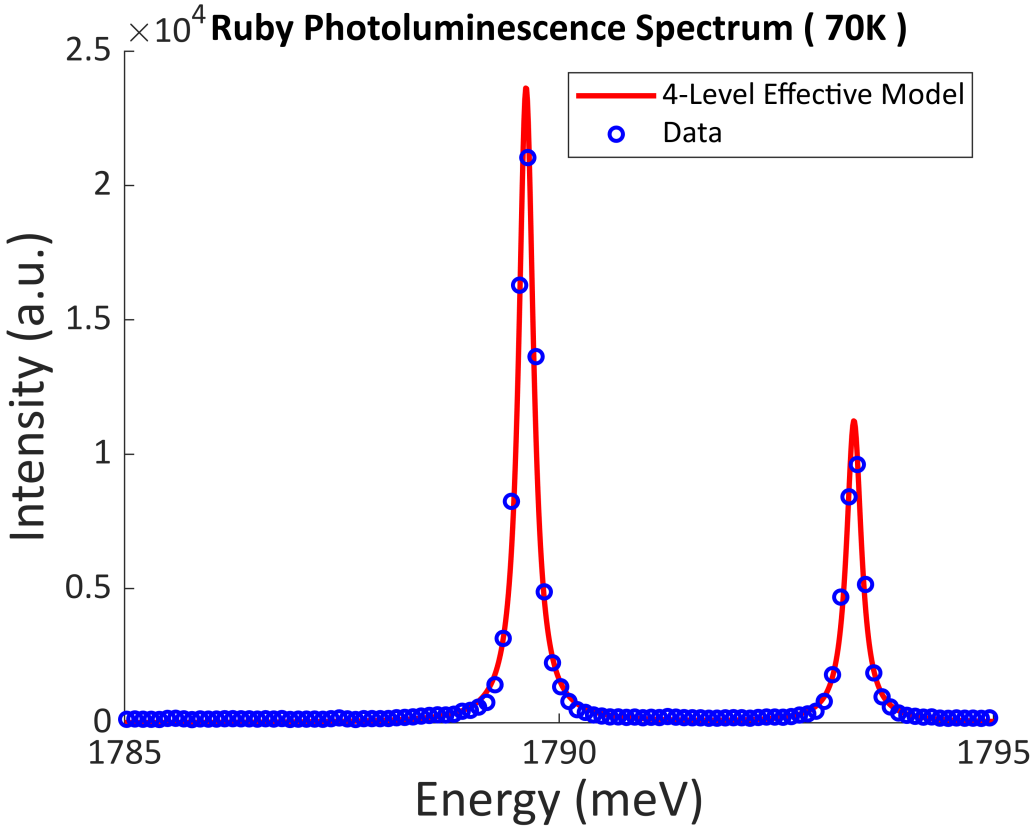

```
Fit Parameters :
T = 40 K
I0 = 2532.89
d_{20}/d_{10} = 1.00761
Linewidths = [0.194199, 0.207606]
Delta = 3.76785
E1 = 1789.64
```



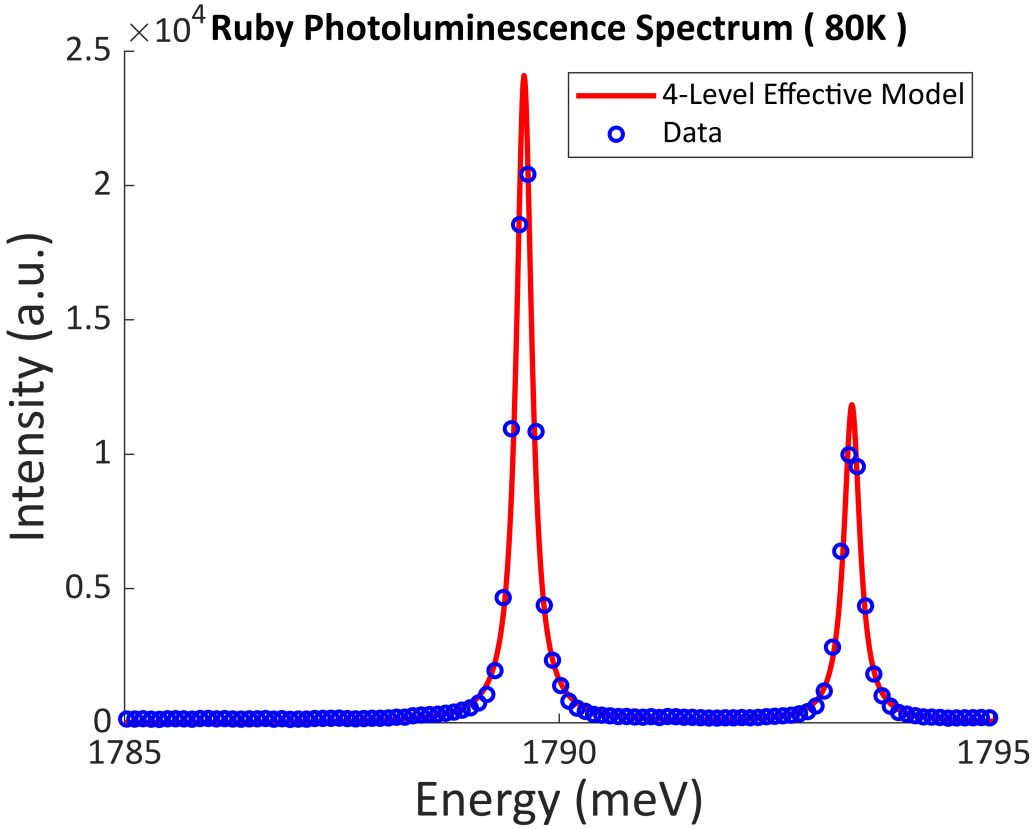

```
Fit Parameters :
T = 50 K
I0 = 2207.86
d_{20}/d_{10} = 0.983594
Linewidths = [0.197364, 0.210891]
Delta = 3.76787
E1 = 1789.65
```



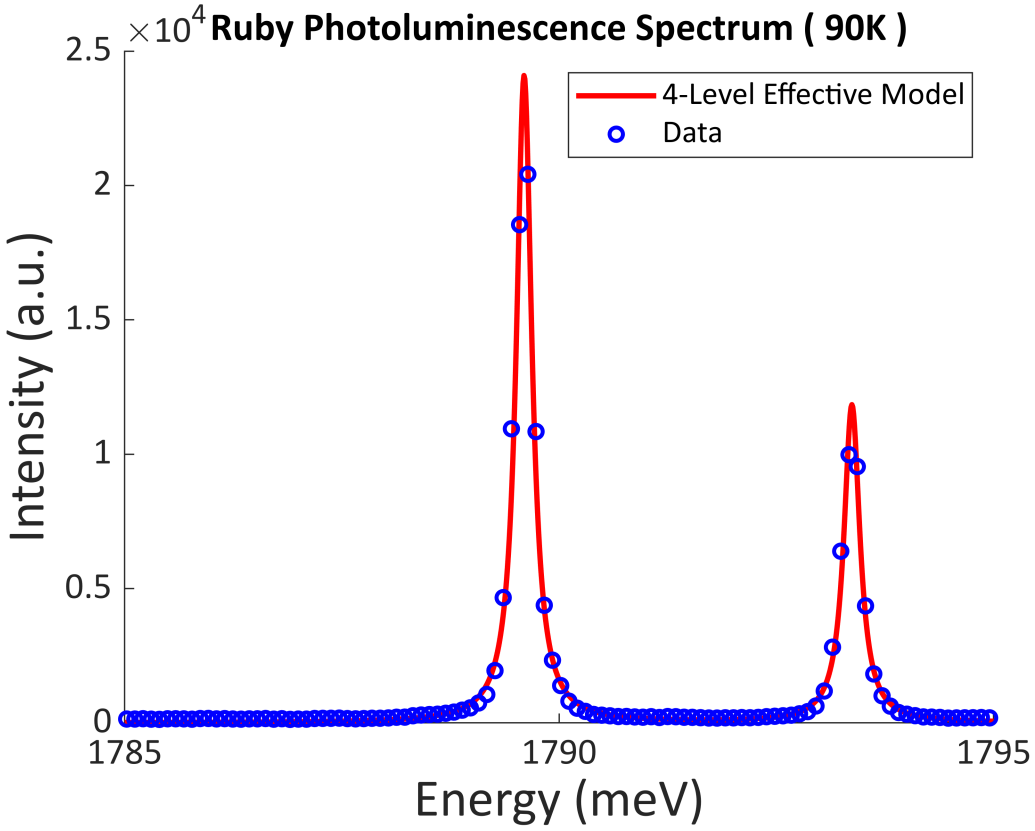

```
Fit Parameters :
T = 60 K
I0 = 2259.41
d_{20}/d_{10} = 0.967991
Linewidths = [0.201637, 0.209764]
Delta = 3.77205
E1 = 1789.63
```



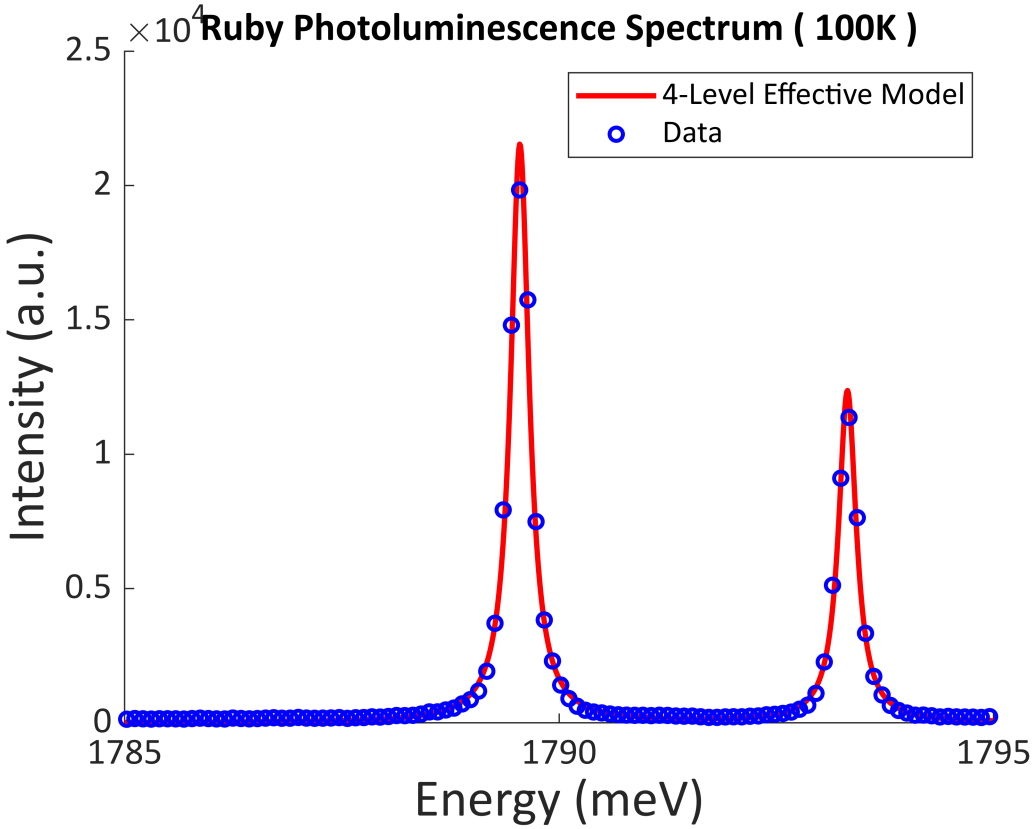

```
Fit Parameters :
T = 70 K
I0 = 2430.6
d_{20}/d_{10} = 0.948496
Linewidths = [0.205448, 0.208163]
Delta = 3.7731
E1 = 1789.62
```



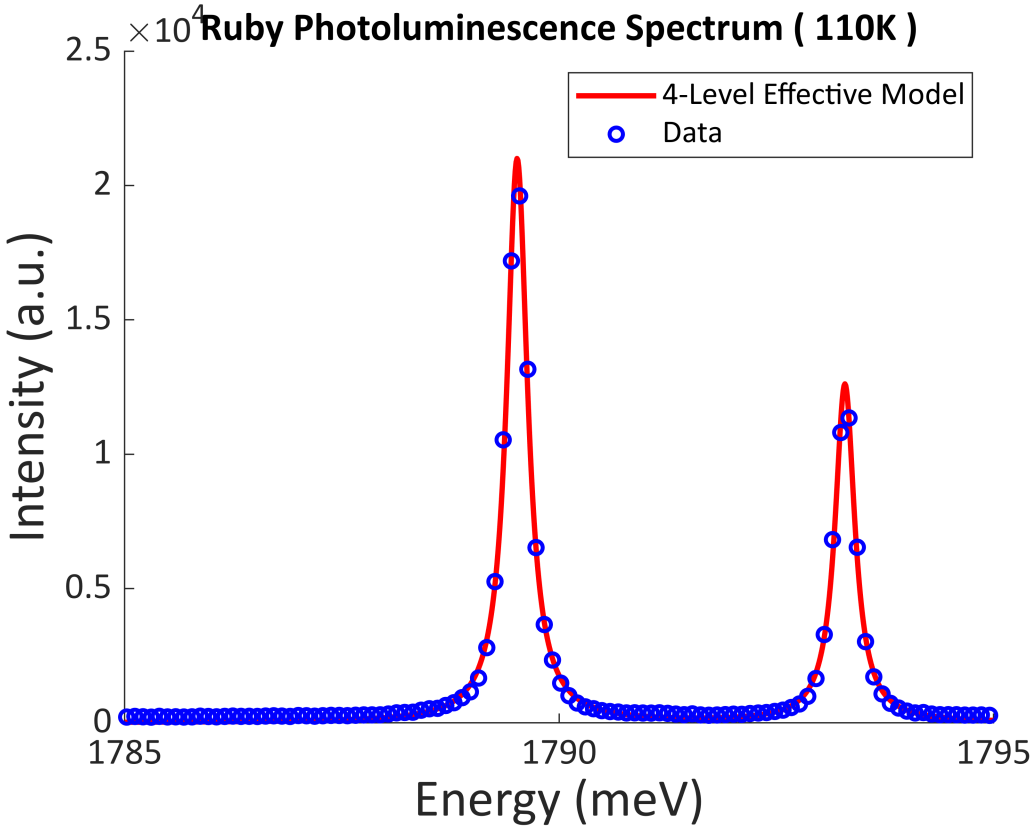

```
Fit Parameters :
T = 80 K
I0 = 2588.04
d_{20}/d_{10} = 0.937551
Linewidths = [0.214889, 0.22258]
Delta = 3.77235
E1 = 1789.6
```



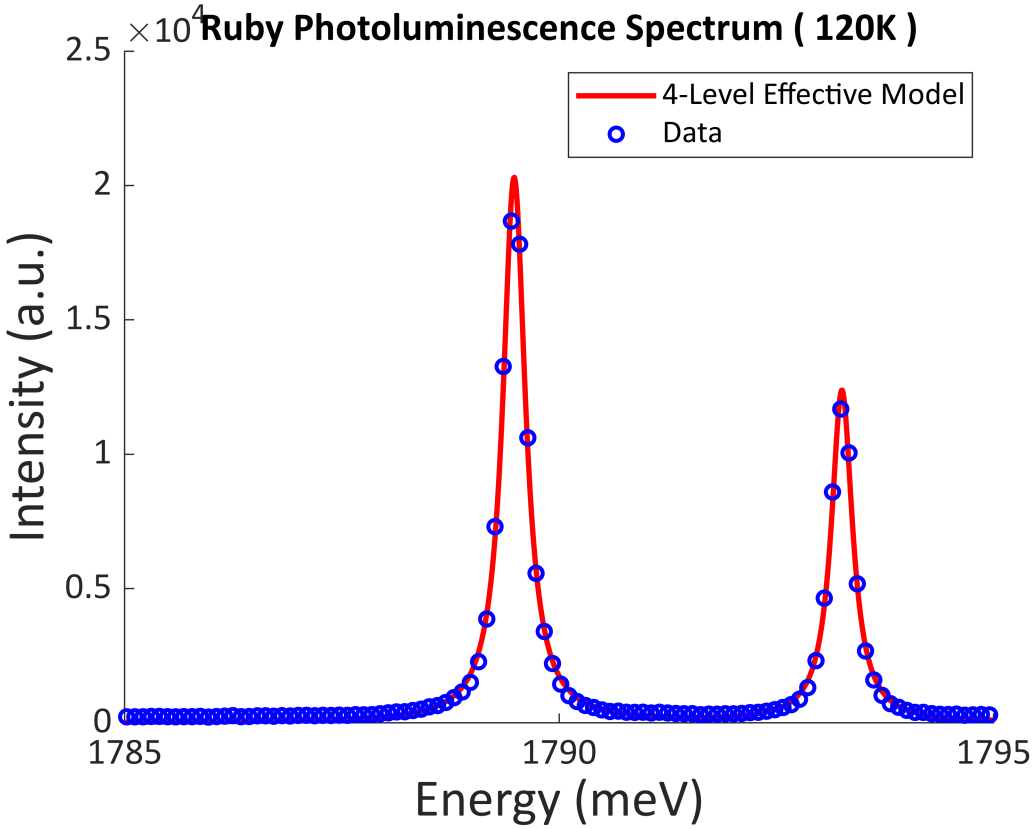

```
Fit Parameters :
T = 90 K
I0 = 2588.04
d_{20}/d_{10} = 0.909468
Linewidths = [0.214889, 0.22258]
Delta = 3.77235
E1 = 1789.6
```



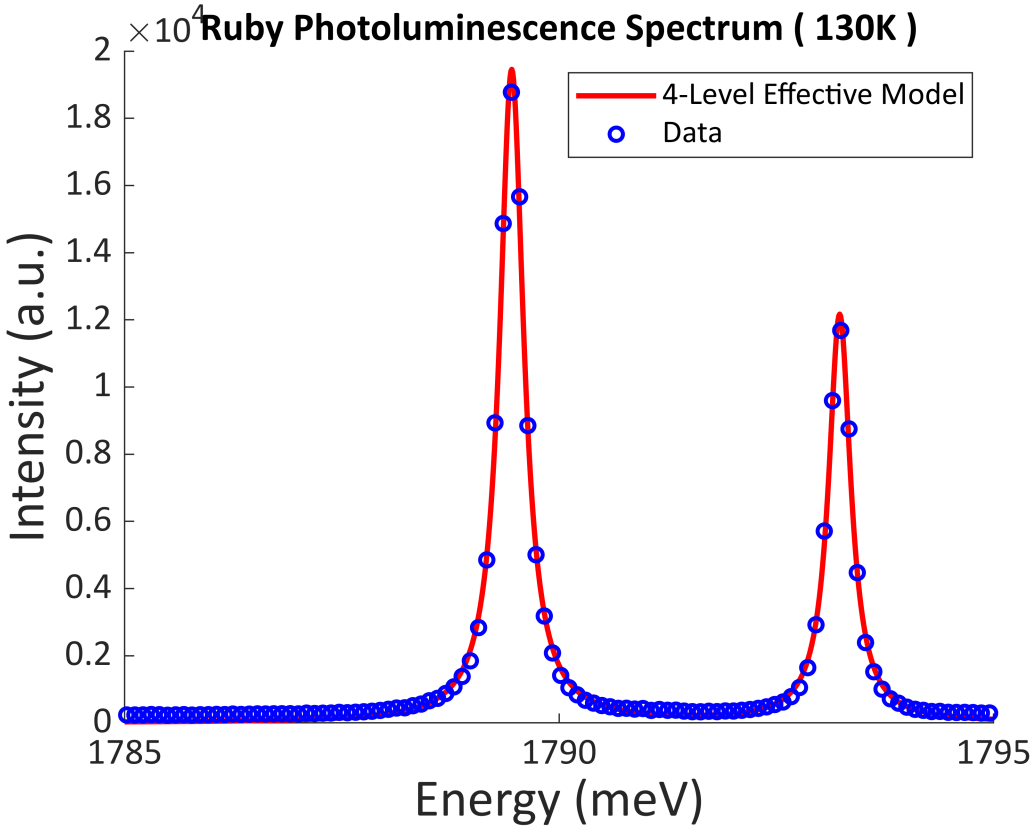

```
Fit Parameters :
T = 100 K
I0 = 2781.09
d_{20}/d_{10} = 0.921691
Linewidths = [0.258281, 0.247062]
Delta = 3.76962
E1 = 1789.55
```



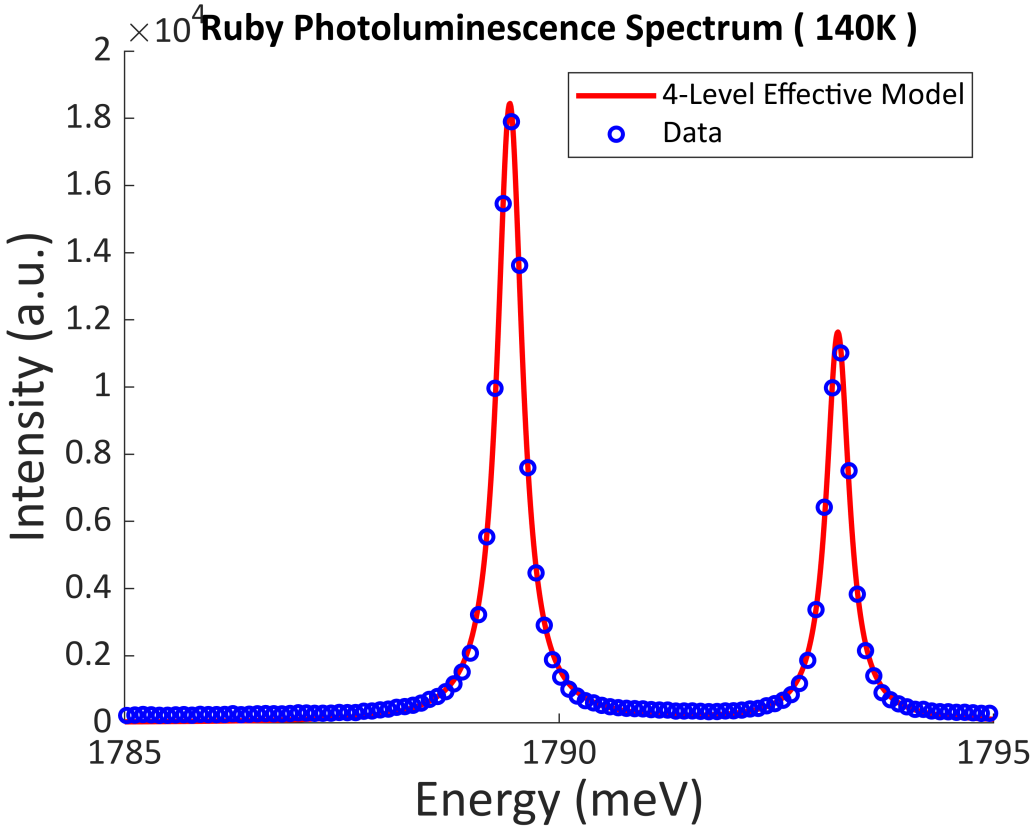

```
Fit Parameters :
T = 110 K
I0 = 3047.61
d_{20}/d_{10} = 0.919896
Linewidths = [0.290382, 0.275231]
Delta = 3.77204
E1 = 1789.52
```



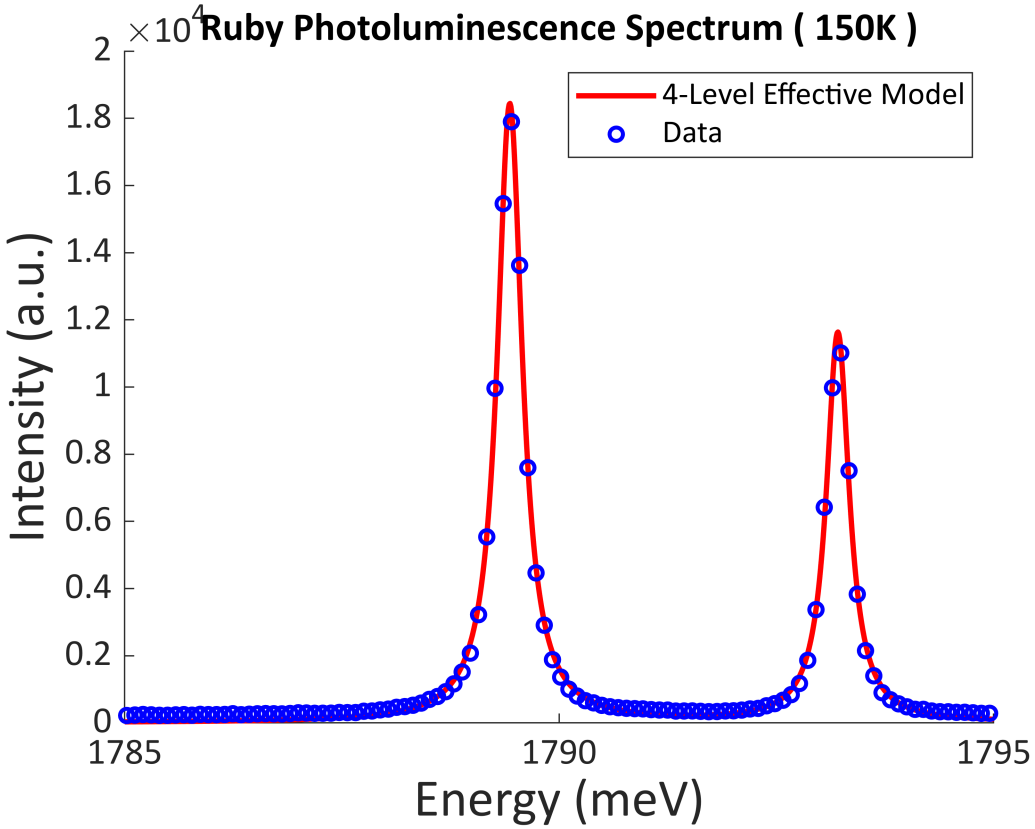

```
Fit Parameters :
T = 120 K
I0 = 3195.55
d_{20}/d_{10} = 0.908303
Linewidths = [0.314963, 0.296232]
Delta = 3.77214
E1 = 1789.48
```



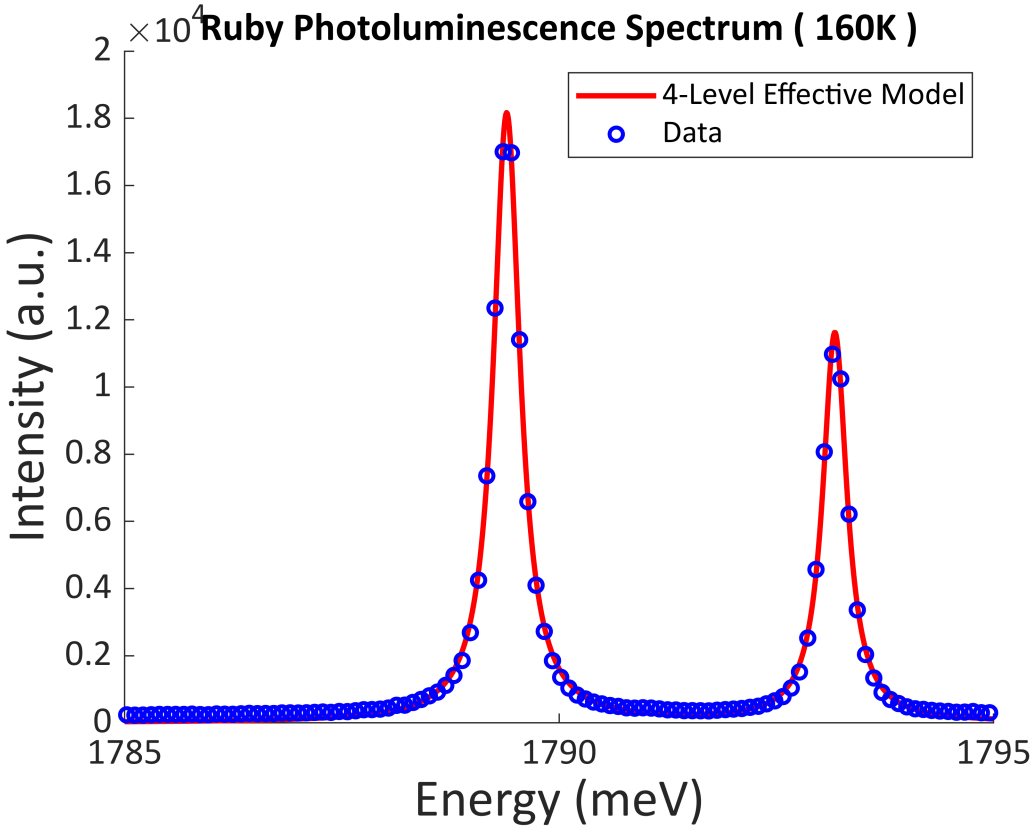

```
Fit Parameters :
T = 130 K
I0 = 3264.75
d_{20}/d_{10} = 0.897232
Linewidths = [0.335945, 0.309329]
Delta = 3.77433
E1 = 1789.45
```



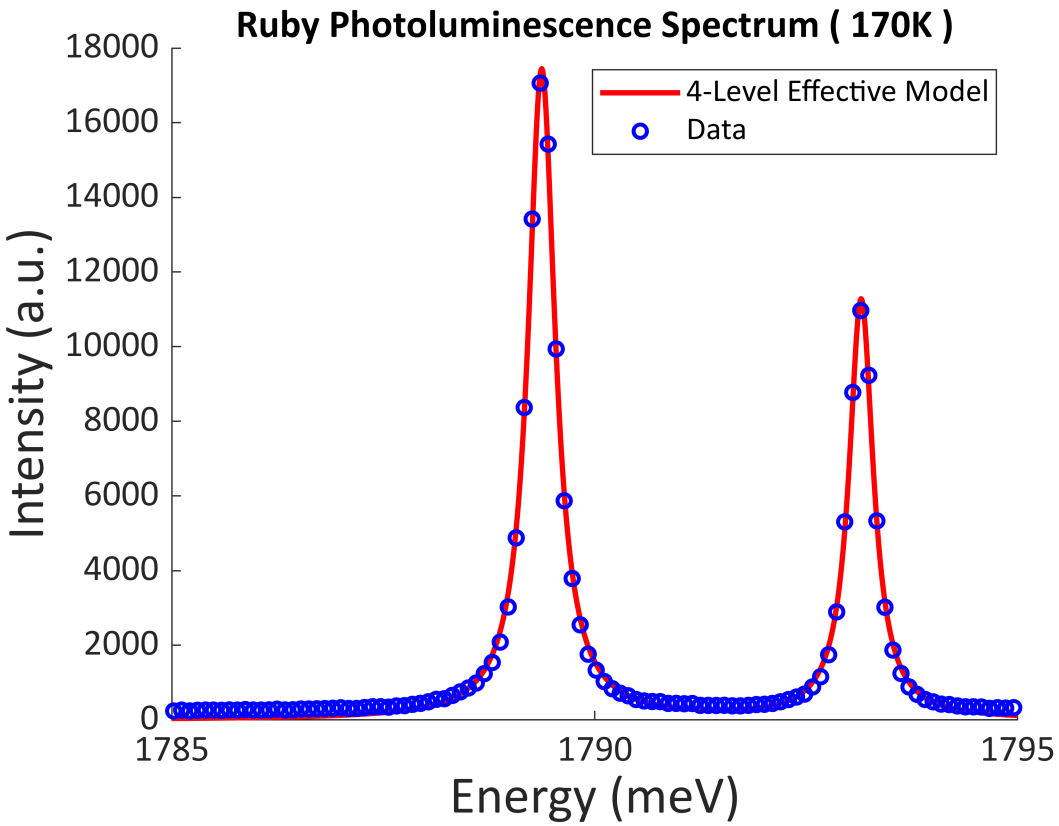

```
Fit Parameters :
T = 140 K
I0 = 3215.04
d_{20}/d_{10} = 0.887194
Linewidths = [0.348969, 0.319069]
Delta = 3.77518
E1 = 1789.43
```



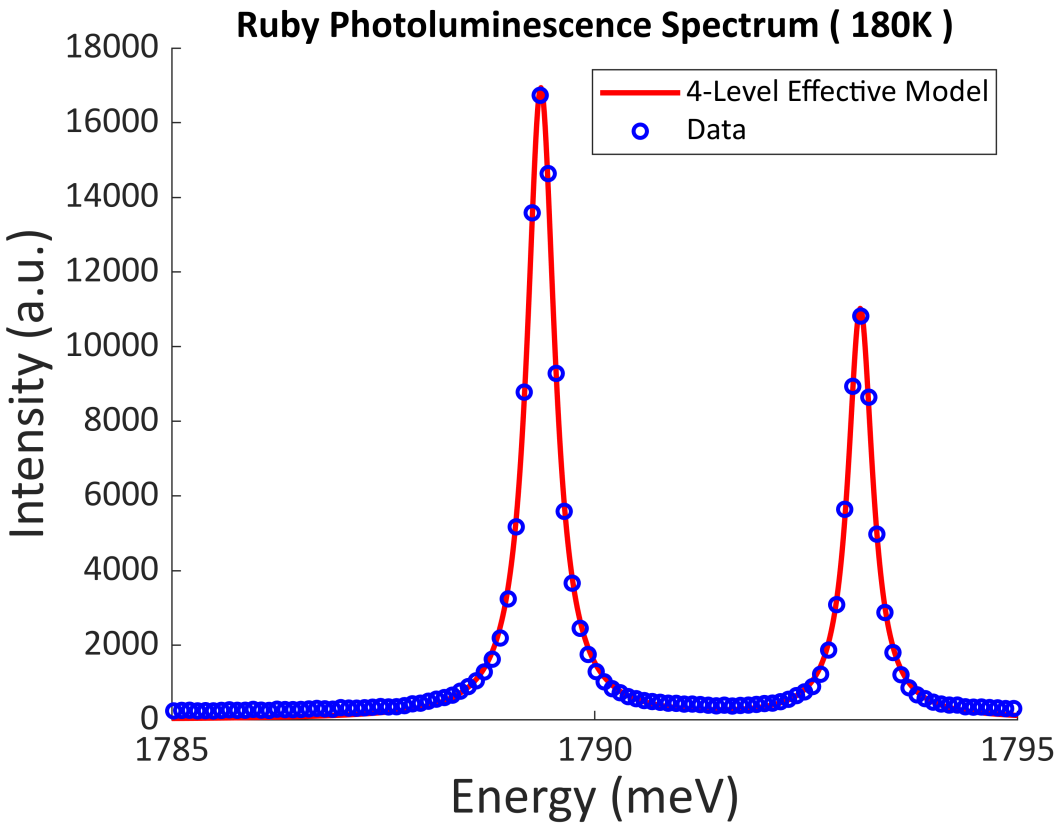

```
Fit Parameters :
T = 150 K
I0 = 3215.04
d_{20}/d_{10} = 0.877985
Linewidths = [0.348969, 0.319069]
Delta = 3.77518
E1 = 1789.43
```



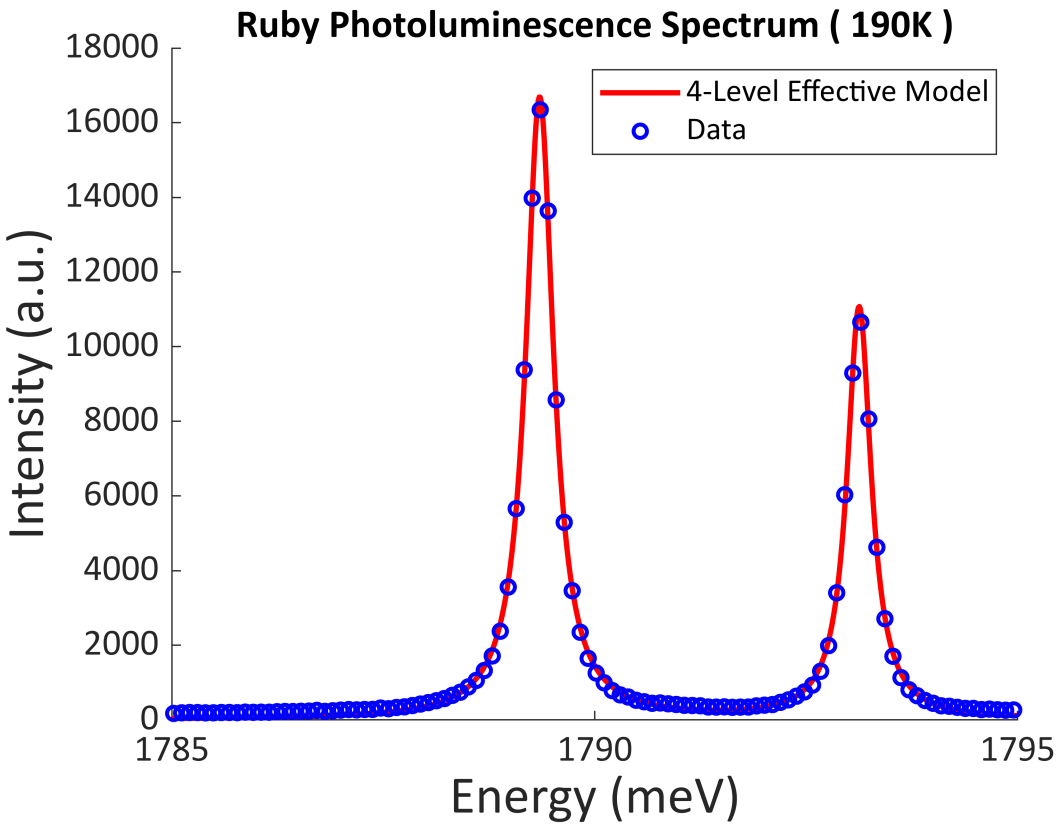

```
Fit Parameters :
T = 160 K
I0 = 3380
d_{20}/d_{10} = 0.872823
Linewidths = [0.372489, 0.337944]
Delta = 3.77652
E1 = 1789.4
```



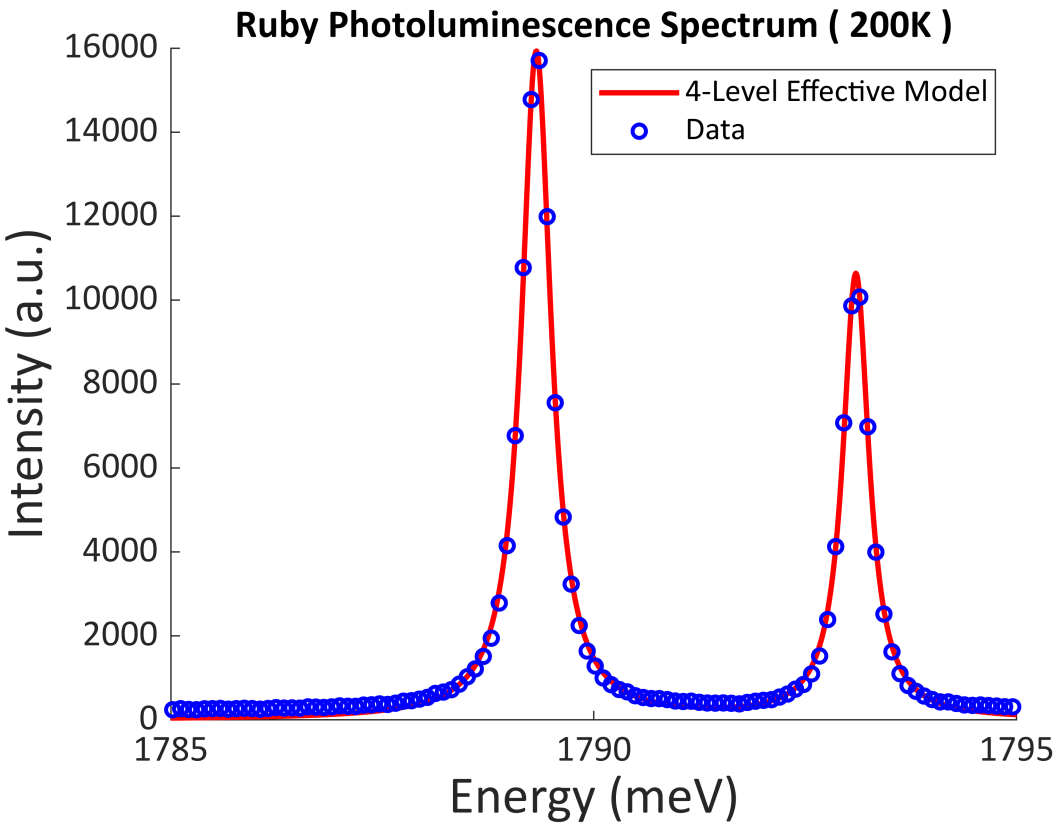

```
Fit Parameters :
T = 170 K
I0 = 3385.15
d_{20}/d_{10} = 0.865745
Linewidths = [0.388508, 0.348816]
Delta = 3.77685
E1 = 1789.37
```



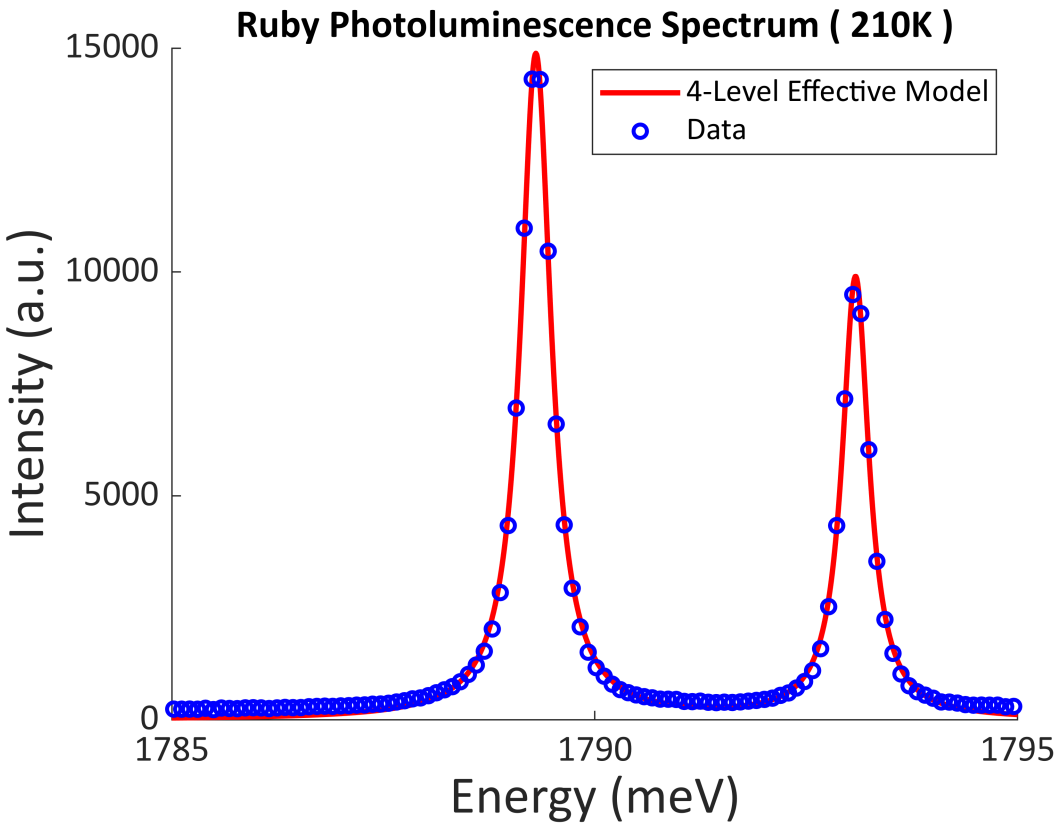

```
Fit Parameters :
T = 180 K
I0 = 3363.74
d_{20}/d_{10} = 0.860755
Linewidths = [0.397715, 0.355828]
Delta = 3.77645
E1 = 1789.36
```



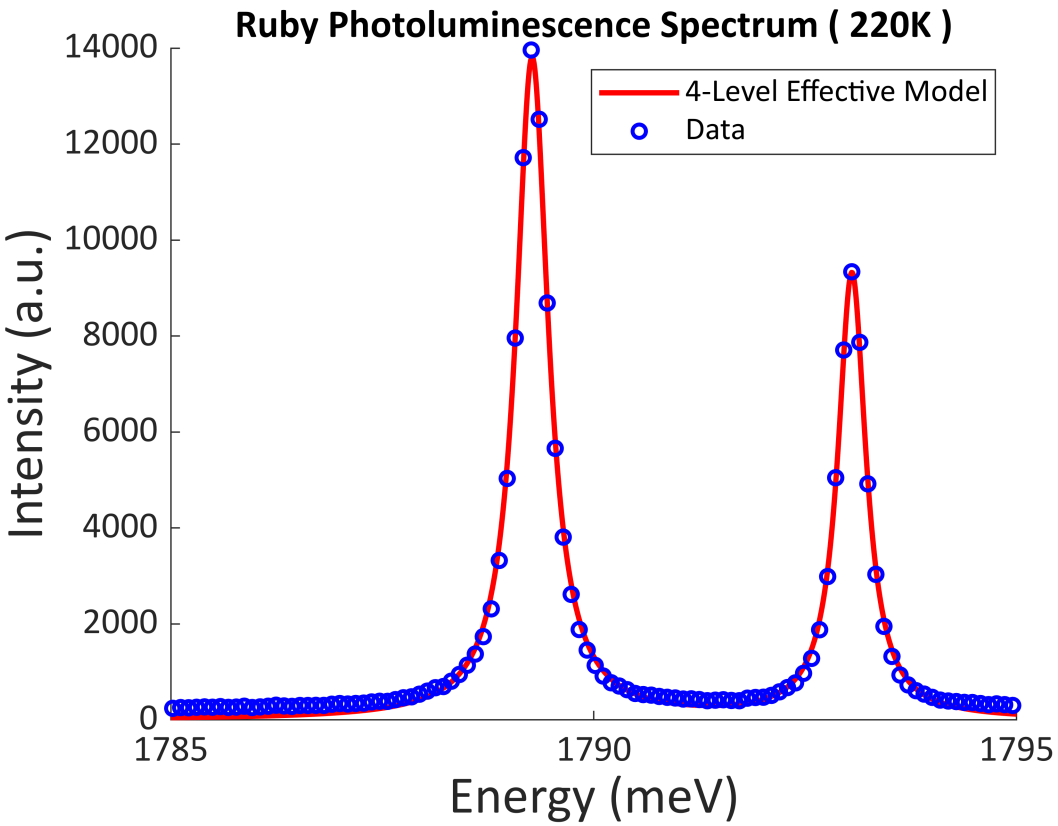

```
Fit Parameters :
T = 190 K
I0 = 3308.4
d_{20}/d_{10} = 0.85465
Linewidths = [0.396832, 0.347818]
Delta = 3.77709
E1 = 1789.35
```



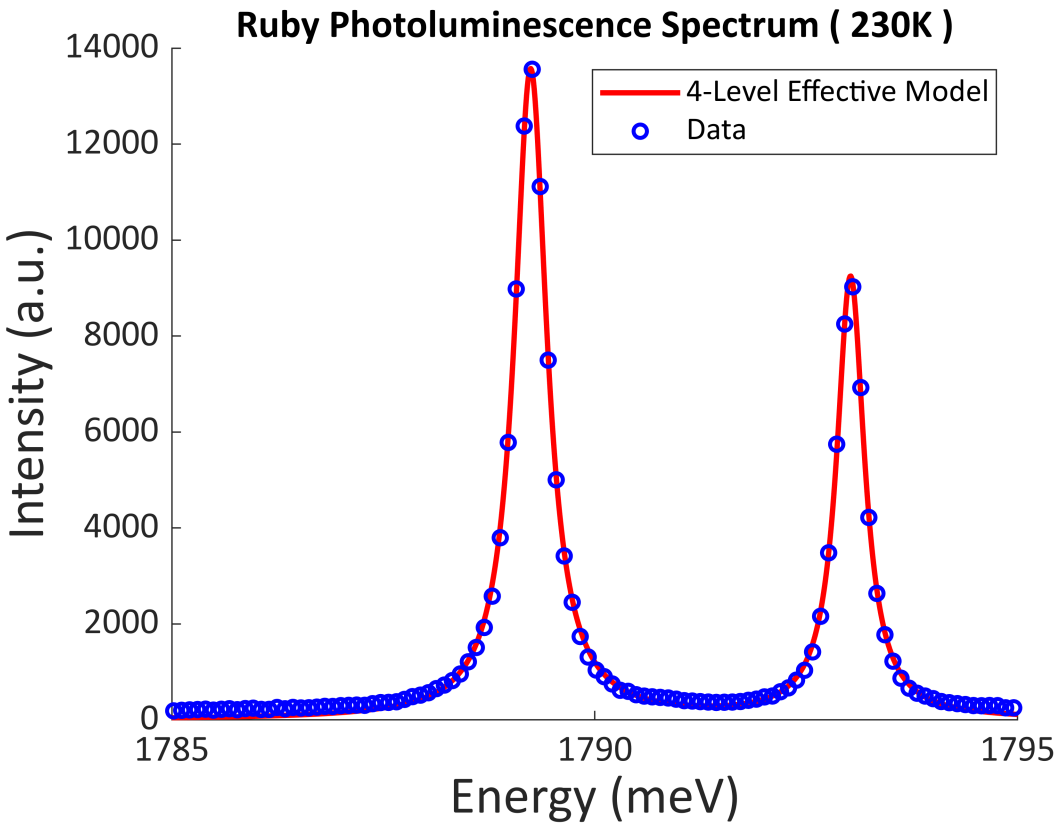

```
Fit Parameters :
T = 200 K
I0 = 3400
d_{20}/d_{10} = 0.853366
Linewidths = [0.427434, 0.37561]
Delta = 3.77827
E1 = 1789.32
```



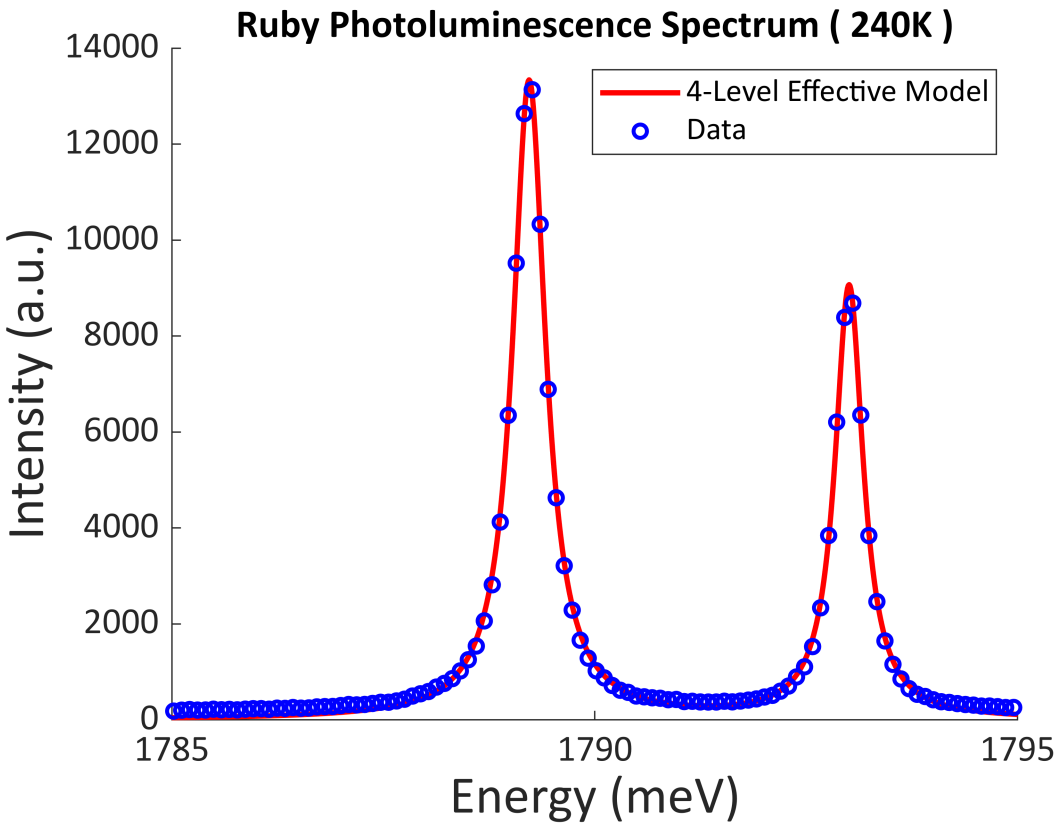

```
Fit Parameters :
T = 210 K
I0 = 3253.2
d_{20}/d_{10} = 0.846827
Linewidths = [0.437869, 0.384091]
Delta = 3.77924
E1 = 1789.3
```



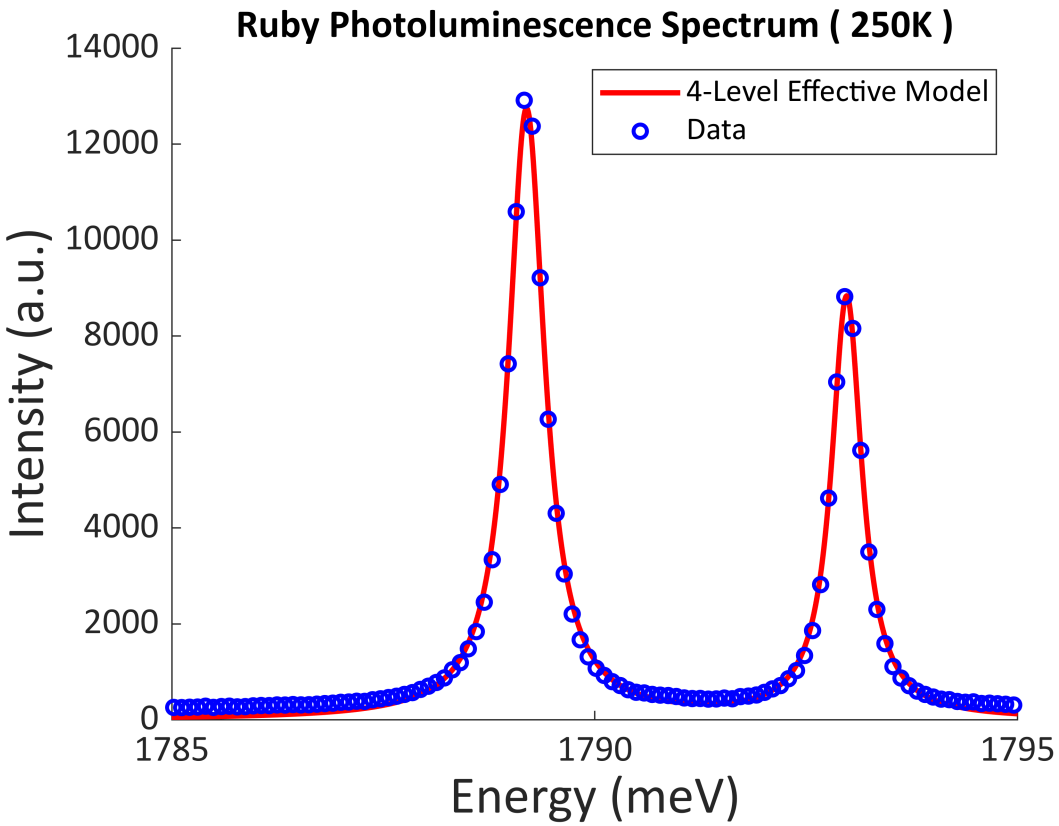

```
Fit Parameters :
T = 220 K
I0 = 3213.66
d_{20}/d_{10} = 0.842941
Linewidths = [0.466054, 0.403303]
Delta = 3.78052
E1 = 1789.27
```



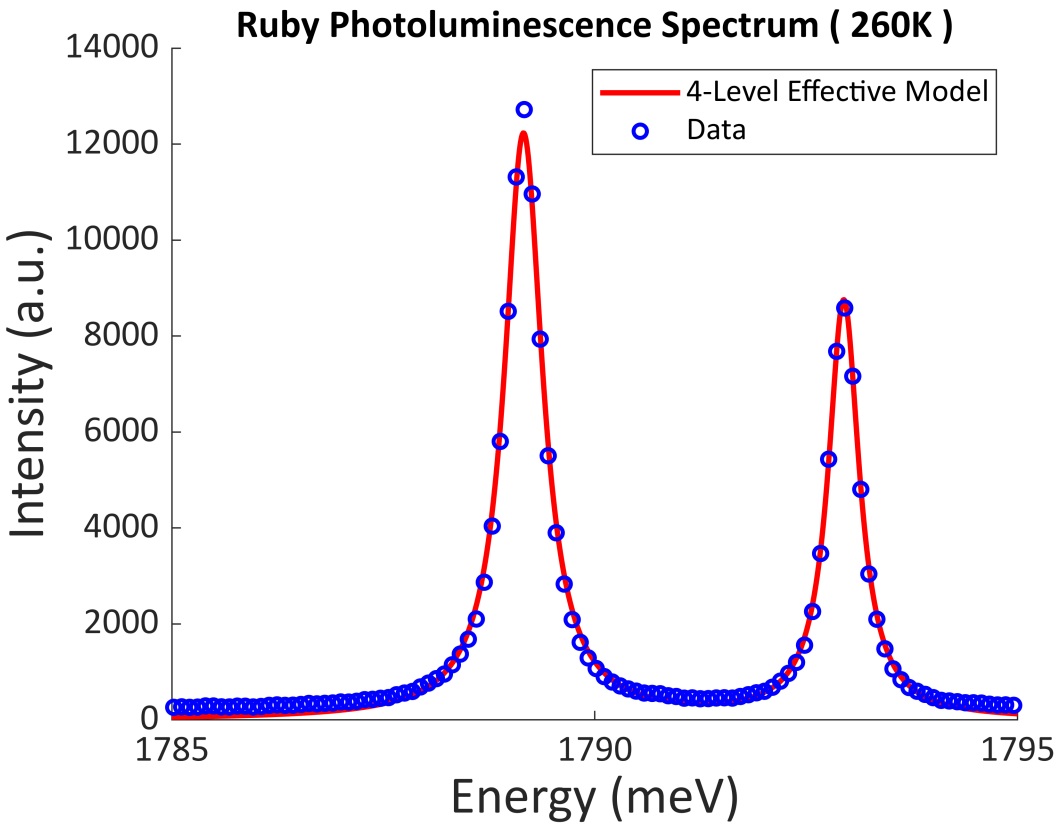

```
Fit Parameters :
T = 230 K
I0 = 3151.23
d_{20}/d_{10} = 0.838754
Linewidths = [0.465159, 0.398113]
Delta = 3.78162
E1 = 1789.24
```



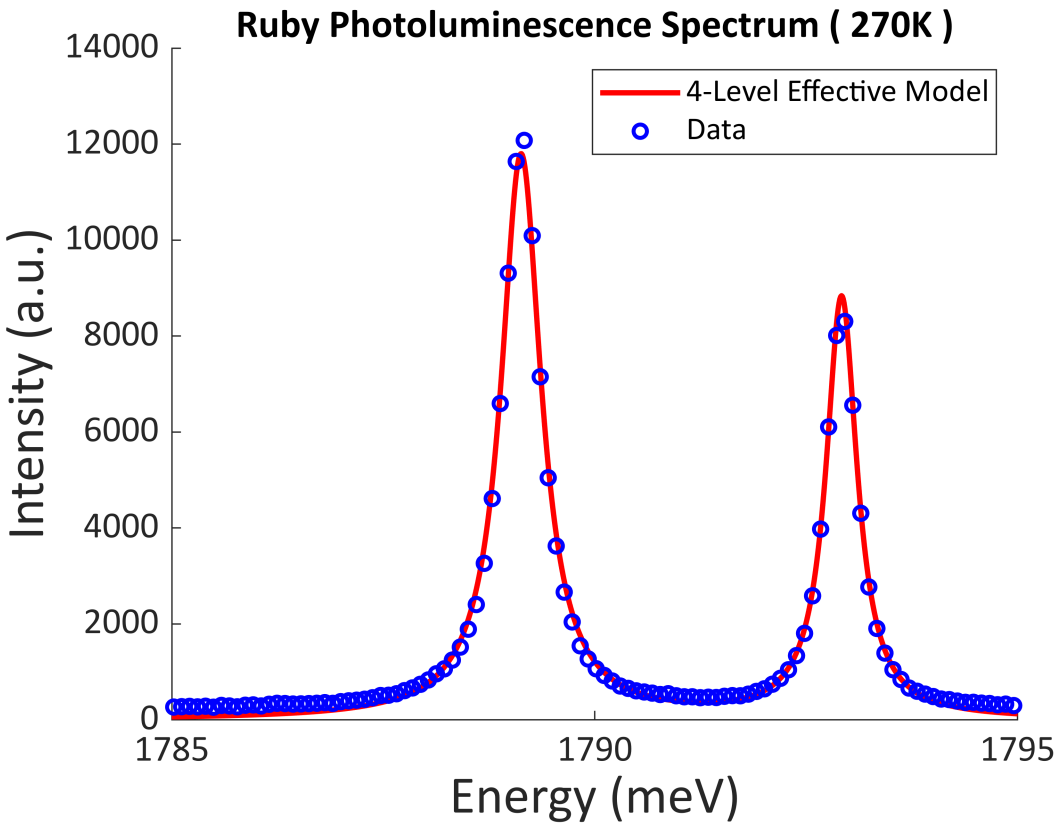

```
Fit Parameters :
T = 240 K
I0 = 3140.24
d_{20}/d_{10} = 0.83559
Linewidths = [0.471783, 0.4049]
Delta = 3.78275
E1 = 1789.22
```



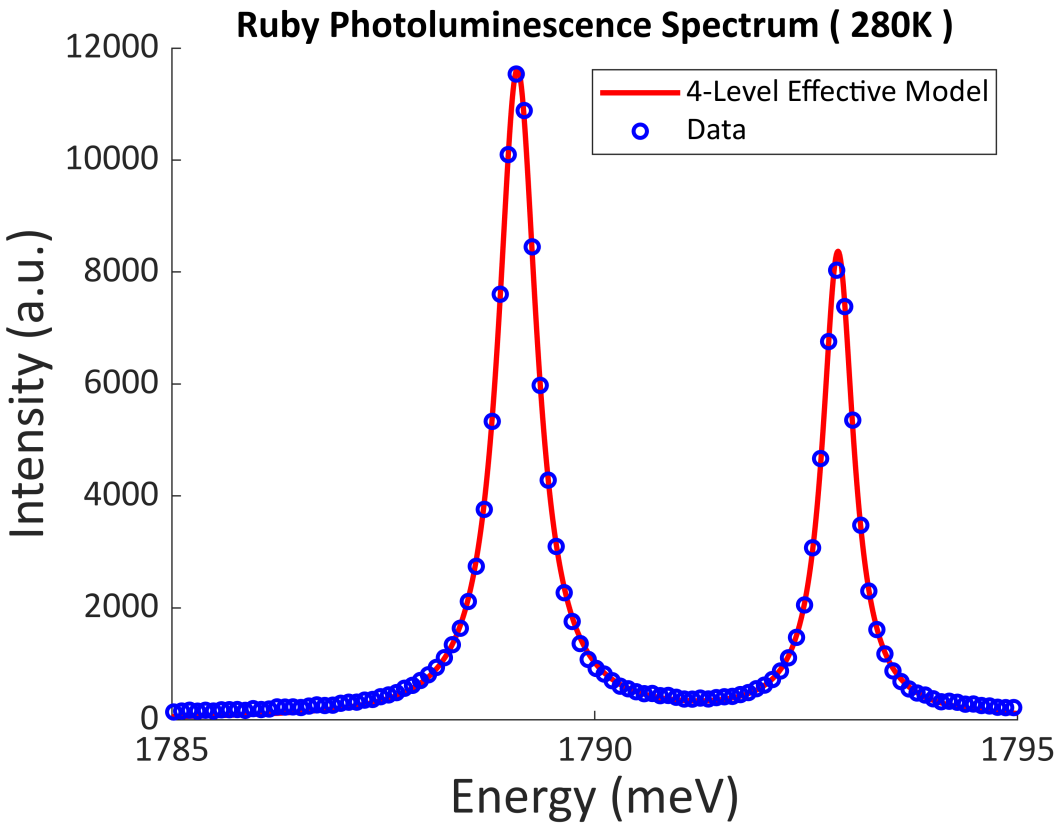

```
Fit Parameters :
T = 250 K
I0 = 3301.53
d_{20}/d_{10} = 0.835394
Linewidths = [0.519036, 0.440315]
Delta = 3.78479
E1 = 1789.19
```



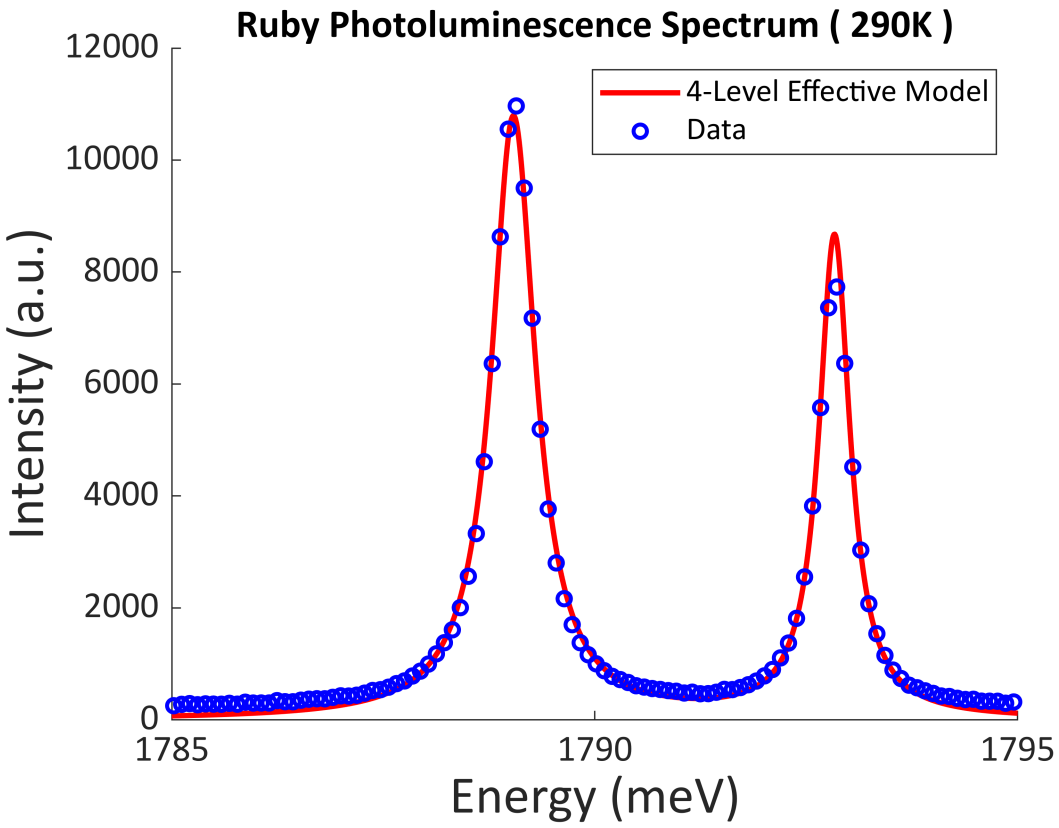

```
Fit Parameters :
T = 260 K
I0 = 3338.63
d_{20}/d_{10} = 0.83
Linewidths = [0.547167, 0.446684]
Delta = 3.78609
E1 = 1789.16
```




```
Fit Parameters :
T = 270 K
I0 = 3397.71
d_{20}/d_{10} = 0.823804
Linewidths = [0.577269, 0.446684]
Delta = 3.7887
E1 = 1789.13
```




```
Fit Parameters :
T = 280 K
I0 = 3213.28
d_{20}/d_{10} = 0.822121
Linewidths = [0.553958, 0.446684]
Delta = 3.78976
E1 = 1789.09
```



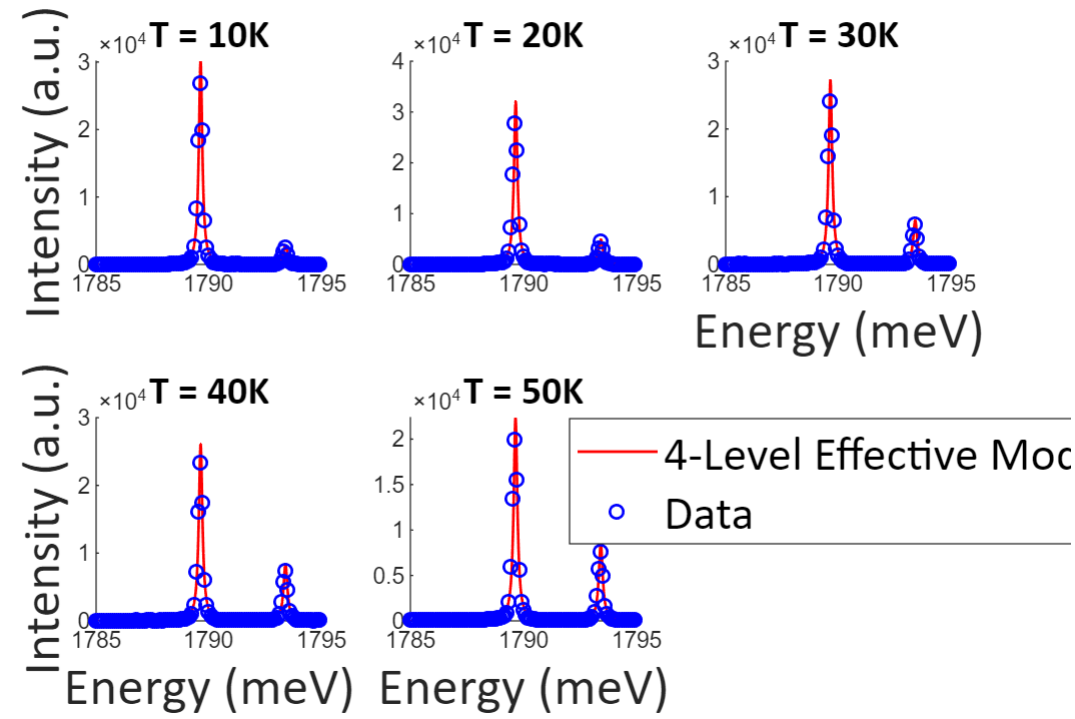

```
Fit Parameters :
T = 290 K
I0 = 3396.2
d_{20}/d_{10} = 0.811605
Linewidths = [0.630957, 0.446684]
Delta = 3.79507
E1 = 1789.04
```

Plot Ruby data in subplots

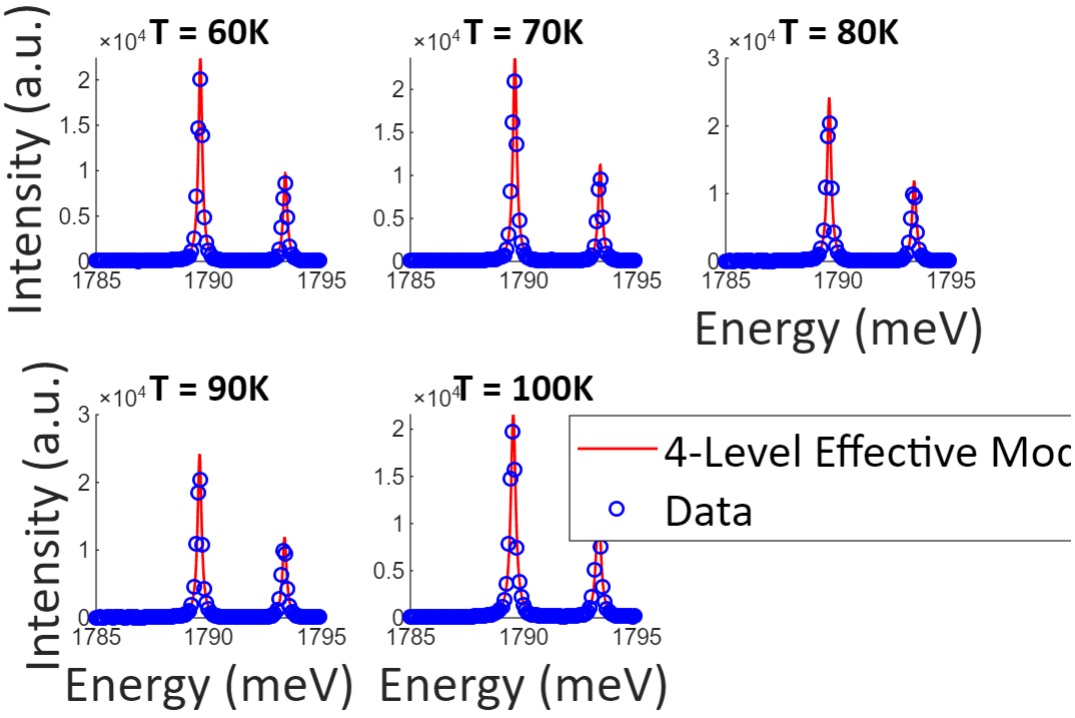
```
for itN = 0:5

    if itN == 5
        Snum = 4;
    else
        Snum = 5;
    end

    figure;
    hold on;
    sgtitle('Ruby Photoluminescence Spectrum', 'FontName', 'Calibri', 'FontSize',
25);

    for itS = 1:Snum
        Dnum = 5*itN + itS;
        T = 10*Dnum;

        % choose data near the peaks
```

```
        Idx = RubyData{Dnum}(:,1) > Erange(1) & RubyData{Dnum}(:,1) < Erange(2);
        E_data = RubyData{Dnum}(Idx,1);
        I_data = RubyData{Dnum}(Idx,2);

        % draw subplot
        ax = subplot(2,3,itS);
        hold on;
        title(['T = ', sprintf('%d', T), 'K'], 'FontName', 'Calibri','FontSize',15);
        I_fit = Spec_4lev(I0(Dnum), TransAmpR(Dnum), Linewidth(Dnum,:),
Delta(Dnum), E1(Dnum), T, Energy);
        plot(Energy, I_fit, 'color', 'red', 'LineWidth', 1);
        plot(E_data, I_data, 'o', 'color', 'blue', 'Markersize', 5, 'LineWidth', 1);
        if itS > 2
            xlabel('Energy (meV)', 'FontName', 'Calibri', 'FontSize', 20);
        end
        if itS == 1 || itS ==4
            ylabel('Intensity (a.u.)', 'FontName', 'Calibri', 'FontSize', 20);
        end
        hold off;
    end

    hLeg = legend({'4-Level Effective Model', 'Data'}, 'Location','northeast',
'FontName', 'Calibri', 'FontSize', 18, 'location', 'Southeast');
    set(hLeg, 'Units', 'normalized');
    set(hLeg, 'Position', [0.75, 0.25 0.1 0.1]);  % Adjust as needed
    hold off;
end
```

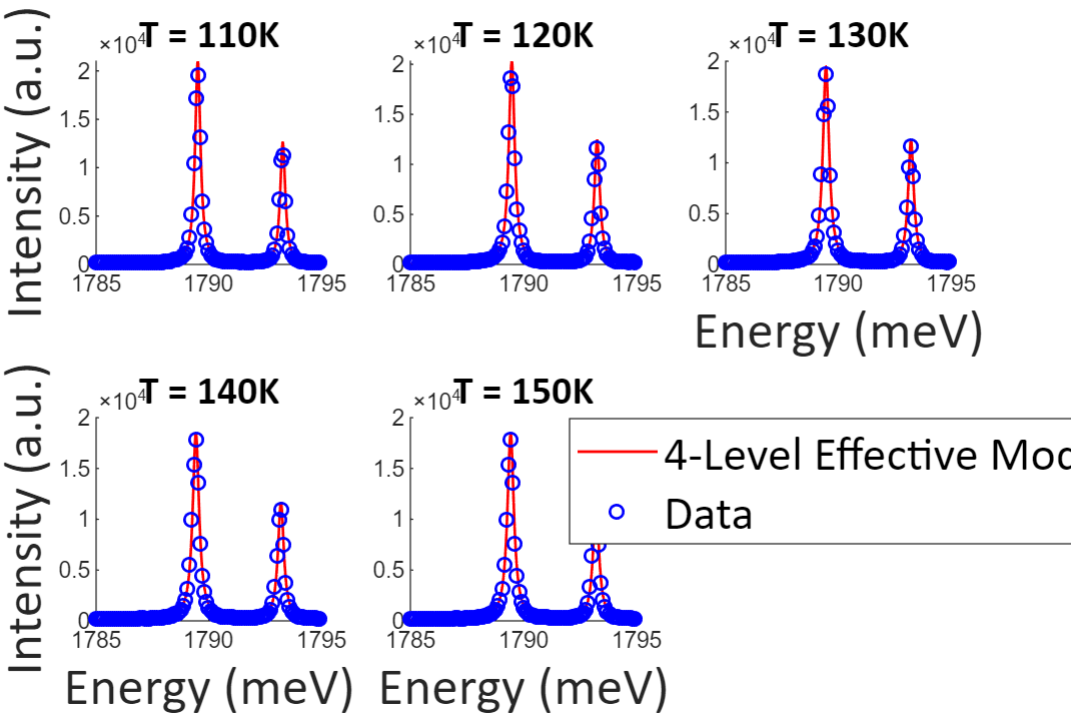
Ruby Photoluminescence Spectrum



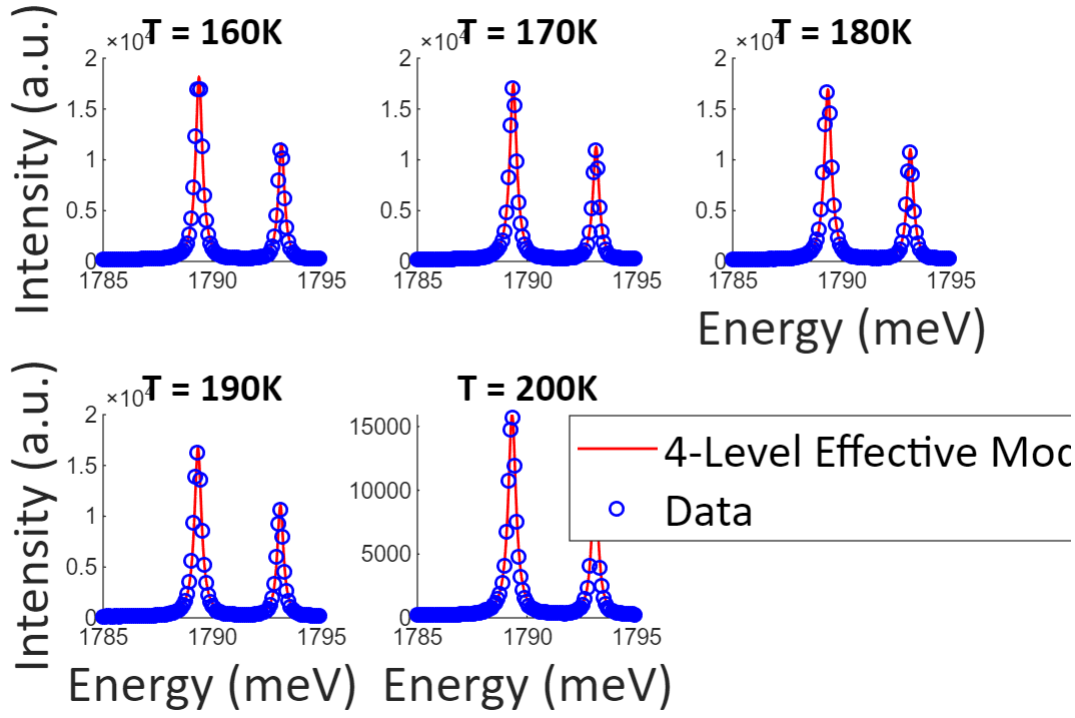
Ruby Photoluminescence Spectrum



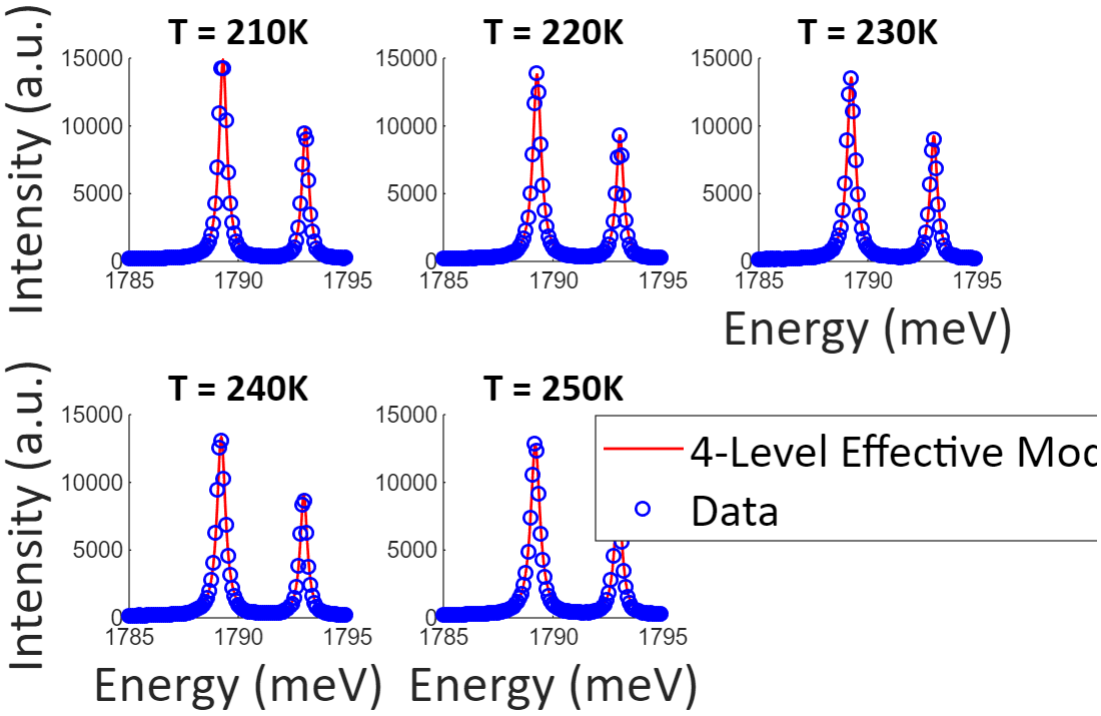
Ruby Photoluminescence Spectrum



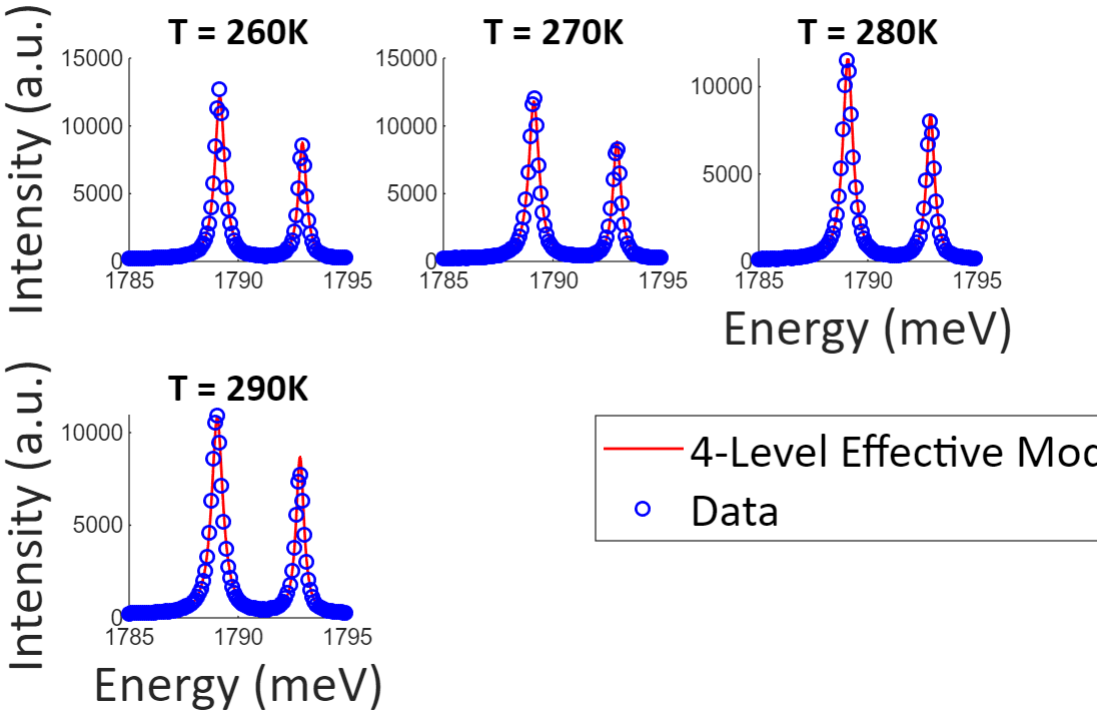
Ruby Photoluminescence Spectrum



Ruby Photoluminescence Spectrum



Ruby Photoluminescence Spectrum



Fit Room Temperature Ruby Data at 7 Different Temperatures, to 4-level Effective Model

```
% photon energy range
Erange = [1785, 1795];
Energy = linspace(Erange(1), Erange(2), 1000);

% lower and upper bounds of fitting parameters
LowerBound = [500, 0.5, 0, 0, 3.5, 1785];
UpperBound = [3000, 3, 1, 1, 4, 1795];

% variables to store fit parameters
I0_R = zeros(7,1);
TransAmpR_R = zeros(7,1);
Linewidth_R = zeros(7,2);
Delta_R = zeros(7,1);
E1_R = zeros(7,1);

% try fitting for 7 different temperatures
for itN = 1:7

    T = 5*(itN+54);    % temperature

    % choose data near the peaks
    Idx = RubyRoomData(:,1) > Erange(1) & RubyRoomData(:,1) < Erange(2);
    E_data = RubyRoomData(Idx,1);
    I_data = RubyRoomData(Idx,2);

    % make parameter bounds and loss function into a single struct variable
    options.lb = LowerBound;
    options.ub = UpperBound;
    options.loss_type = loss_func;

    % define fitting curve
    FitModel = @(Params, Energies) Spec_4lev(Params(1), Params(2), [power(10,
Params(3)), power(10, Params(4))], Params(5), Params(6), T, Energies);

    % iterate to find best fit
    best_loss = Inf;
    for it = 1:3
        options.rng_seed = it;          % Try different seeds
        Params0 = rand(1,6) .* (UpperBound - LowerBound) + LowerBound;      % try
different initial parameters
        [params_i, loss_hist_i] = de_curve_fit(FitModel, E_data, I_data, Params0,
options);     % differential evolution fitting
        if loss_hist_i(end) < best_loss
            best_loss = loss_hist_i(end);
            FitParams = params_i;
            loss_history = loss_hist_i;
```

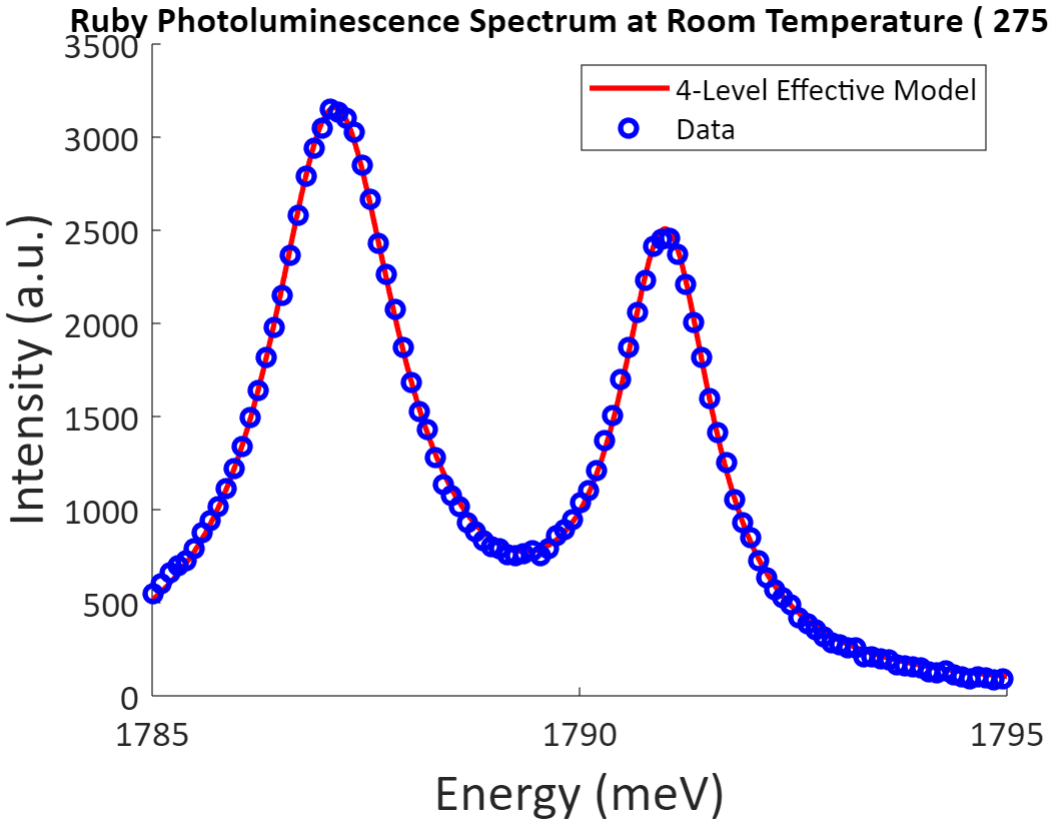
```
        end
    end

    I0_R(itN) = FitParams(1);
    TransAmpR_R(itN) = FitParams(2);
    Linewidth_R(itN,:) = [power(10, FitParams(3)), power(10, FitParams(4))];
    Delta_R(itN) = FitParams(5);
    E1_R(itN) = FitParams(6);

    % show message
    disptime(['T = ', sprintf('%d', T), 'K | loss = ', sprintf('%.4g', best_loss)]);
end
```

```
25-06-10 11:17:09 | T = 275K | loss = 0.5606
25-06-10 11:17:14 | T = 280K | loss = 0.5606
25-06-10 11:17:17 | T = 285K | loss = 0.5606
25-06-10 11:17:21 | T = 290K | loss = 0.5606
25-06-10 11:17:24 | T = 295K | loss = 0.5606
25-06-10 11:17:28 | T = 300K | loss = 0.5606
25-06-10 11:17:31 | T = 305K | loss = 0.5606
```

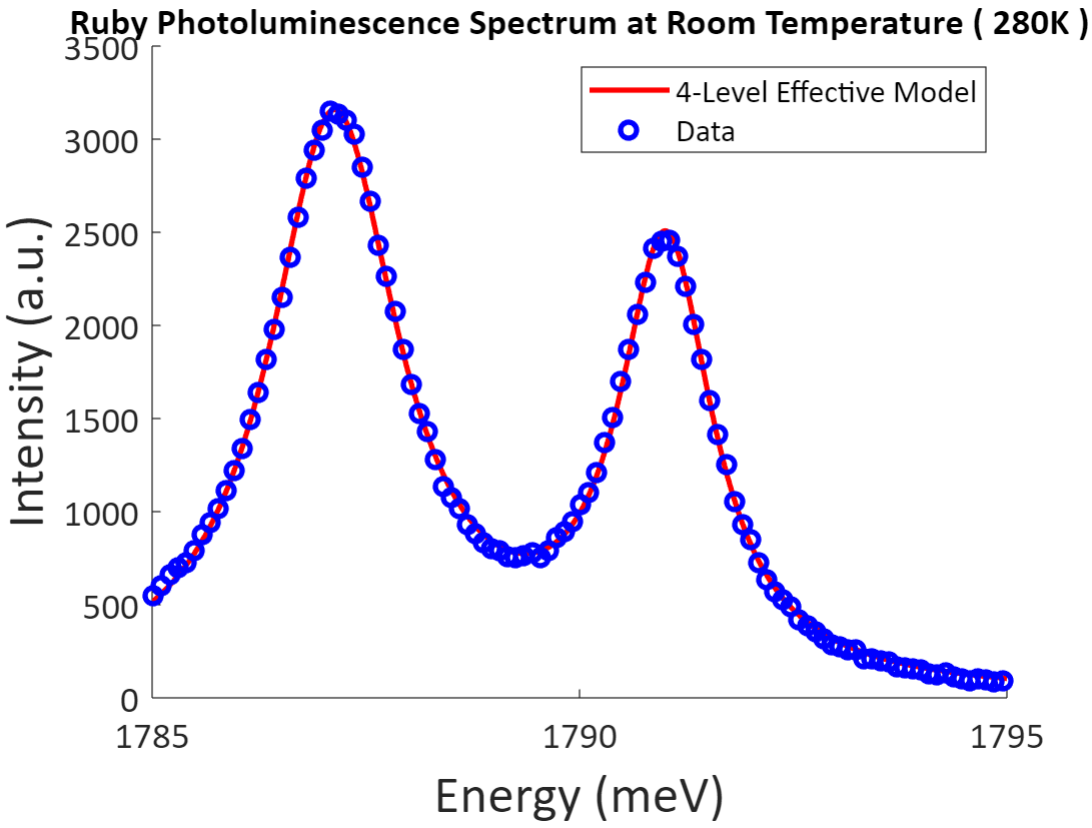
Plot Room Temperature Ruby Data with Fit Curve

```
for itN = 1:7

    T = 5*(itN+54);    % temperature

    % choose data near the peaks
    Idx = RubyRoomData(:,1) > Erange(1) & RubyRoomData(:,1) < Erange(2);
    E_data = RubyRoomData(Idx,1);
    I_data = RubyRoomData(Idx,2);

    % plot
    figure;
    hold on;
    I_fit = Spec_4lev(I0_R(itN), TransAmpR_R(itN), Linewidth_R(itN,:),
Delta_R(itN), E1_R(itN), T, Energy);
    plot(Energy, I_fit, 'color', 'red', 'LineWidth', 2);
    plot(E_data, I_data, 'o', 'color', 'blue', 'LineWidth', 2);
    ax = gca;
    ax.FontSize = 14;
    ax.FontName = 'Calibri';
    xlabel('Energy (meV)', 'FontName', 'Calibri', 'FontSize', 20);
    ylabel('Intensity (a.u.)', 'FontName', 'Calibri', 'FontSize', 20);
    title(['Ruby Photoluminescence Spectrum at Room Temperature ( ', sprintf('%d',
T), 'K )'], 'FontName', 'Calibri', 'FontSize', 14);
    legend({'4-Level Effective Model', 'Data'}, 'Location','northeast', 'FontName',
'Calibri', 'FontSize', 12);
    hold off;

    % show fit parameters
```

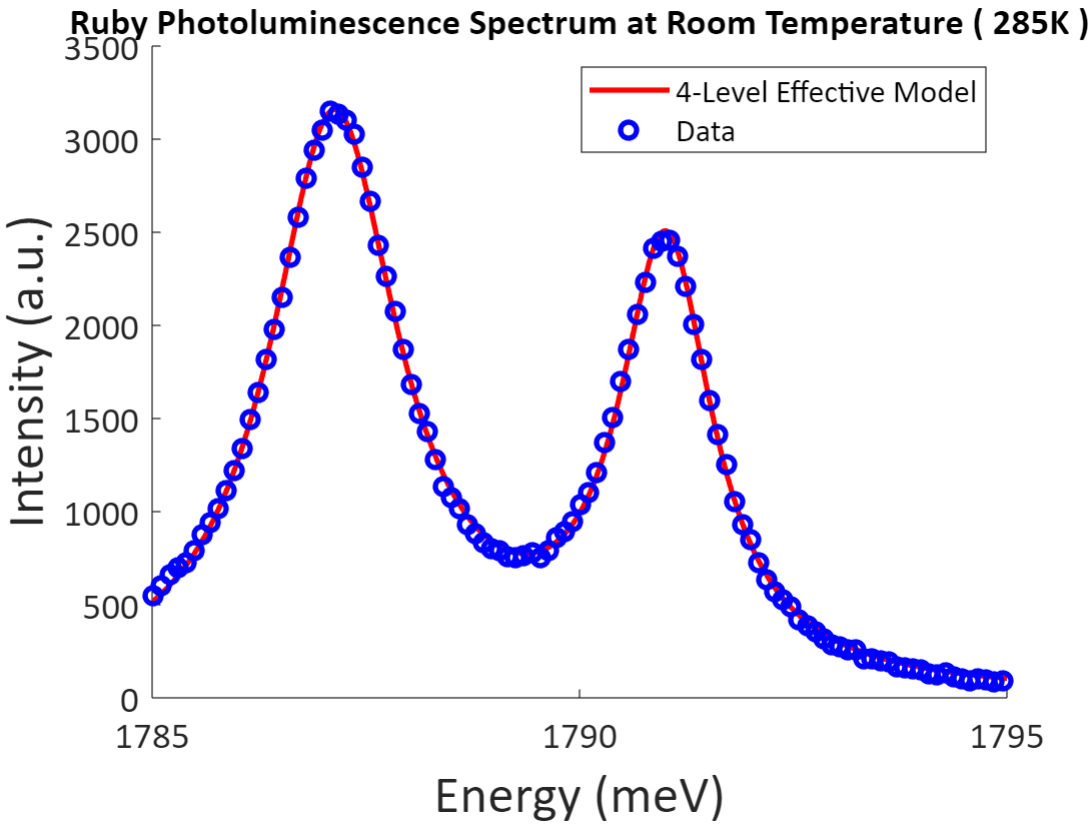
```
    disp('Fit Parameters :')
    disp(['T = ', sprintf('%d', T), ' K']);
    disp(['I0 = ', sprintf('%.4g', I0_R(itN))]);
    disp(['d_{20}/d_{10} = ', sprintf('%.4g', TransAmpR_R(itN))]);
    disp(['Linewidths = [', sprintf('%.4g', Linewidth_R(itN,1)), ', ',
sprintf('%.4g', Linewidth_R(itN,2)) ']']);
    disp(['Delta = ', sprintf('%.4g', Delta_R(itN))]);
    disp(['E1 = ', sprintf('%.4g', E1_R(itN))]);
end
```



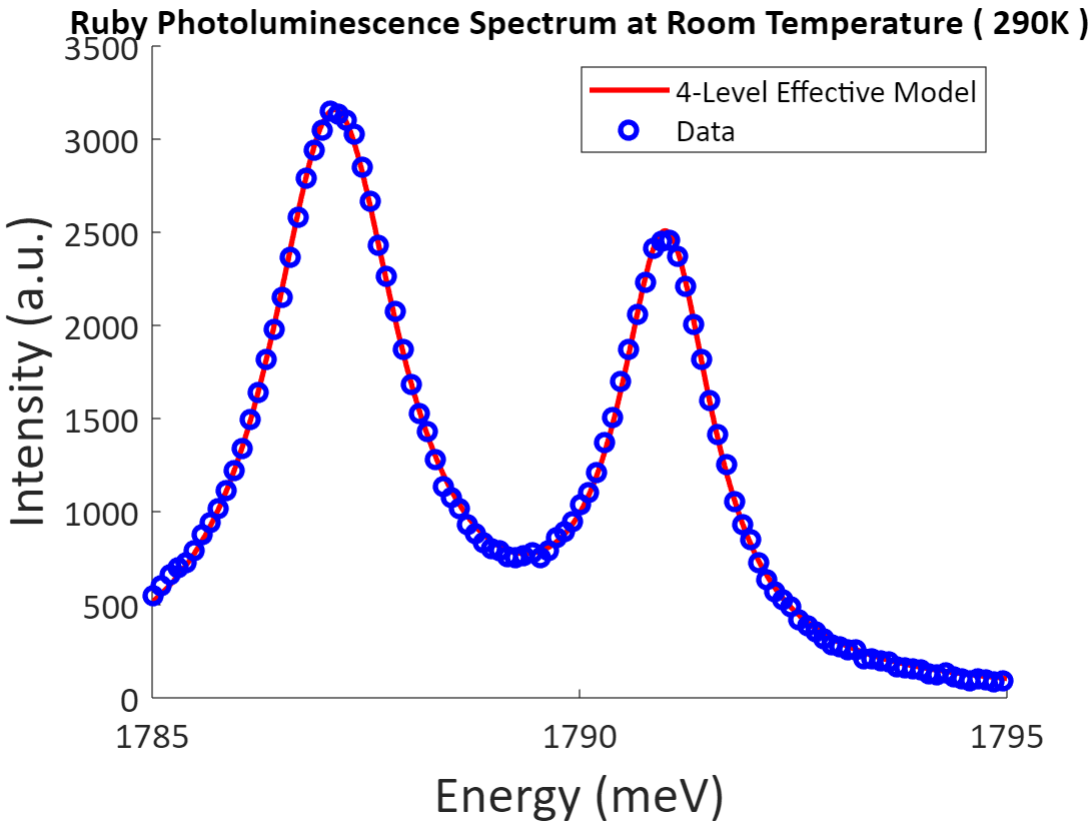

```
Fit Parameters :
T = 275 K
I0 = 2845
d_{20}/d_{10} = 0.8007
Linewidths = [1.84, 1.322]
Delta = 3.879
E1 = 1787
```



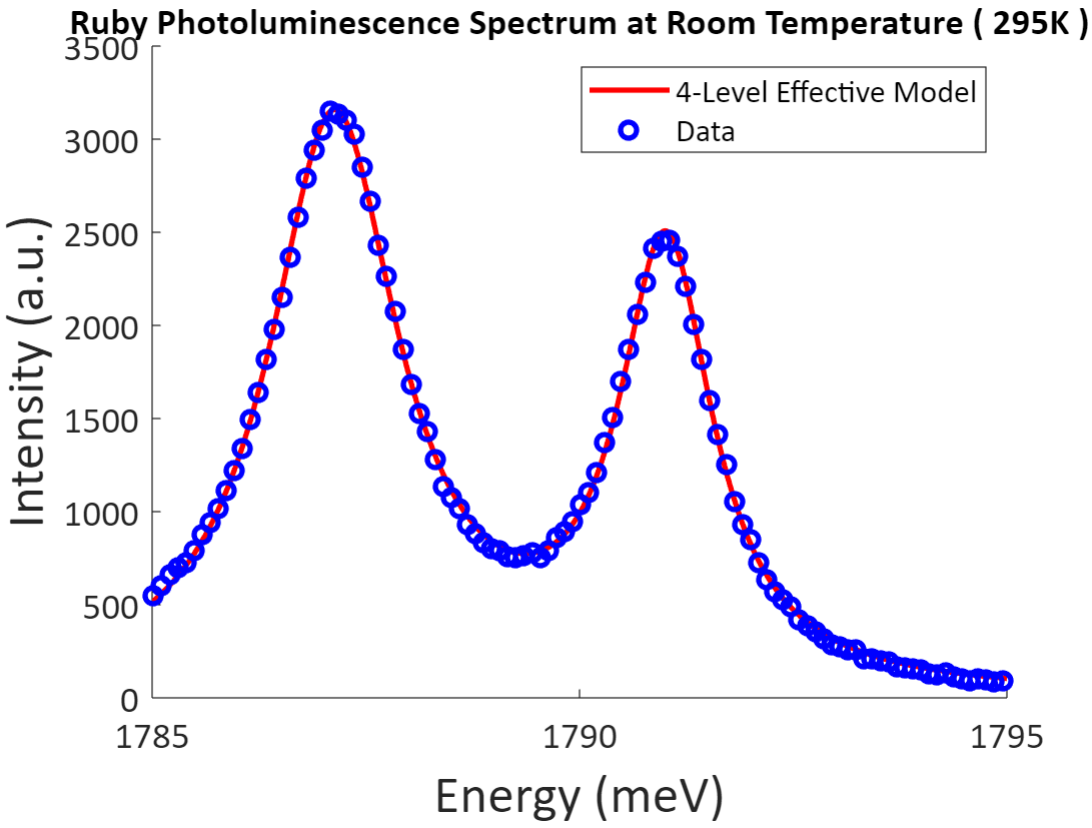

```
Fit Parameters :
T = 280 K
I0 = 2845
d_{20}/d_{10} = 0.7995
Linewidths = [1.84, 1.322]
Delta = 3.879
E1 = 1787
```



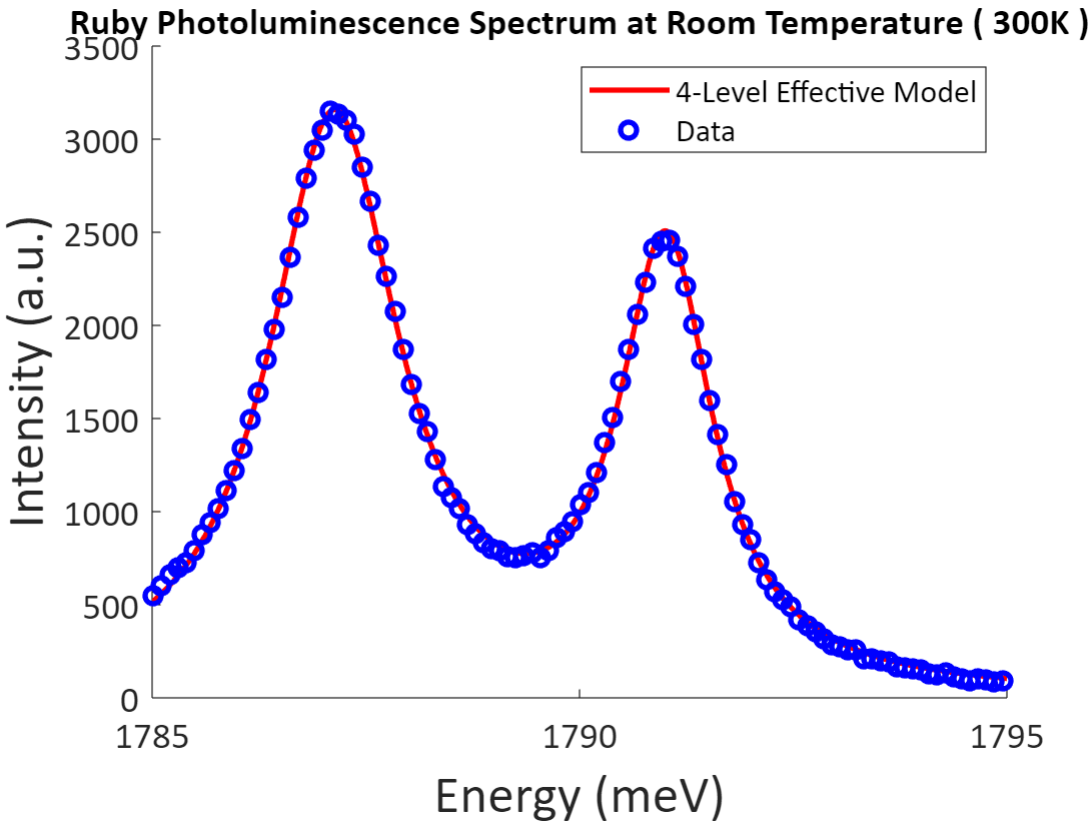

```
Fit Parameters :
T = 285 K
I0 = 2845
d_{20}/d_{10} = 0.7984
Linewidths = [1.84, 1.322]
Delta = 3.879
E1 = 1787
```



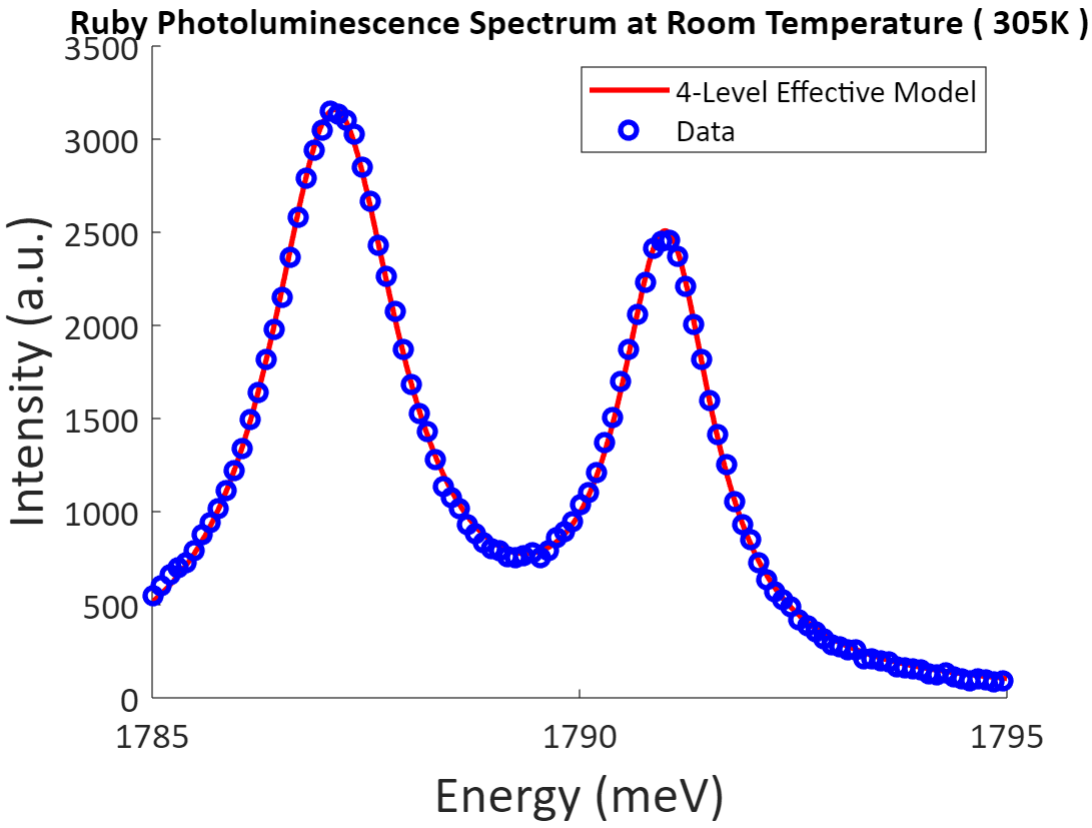

```
Fit Parameters :
T = 290 K
I0 = 2845
d_{20}/d_{10} = 0.7973
Linewidths = [1.84, 1.322]
Delta = 3.879
E1 = 1787
```




```
Fit Parameters :
T = 295 K
I0 = 2845
d_{20}/d_{10} = 0.7963
Linewidths = [1.84, 1.322]
Delta = 3.879
E1 = 1787
```




```
Fit Parameters :
T = 300 K
I0 = 2845
d_{20}/d_{10} = 0.7953
Linewidths = [1.84, 1.322]
Delta = 3.879
E1 = 1787
```

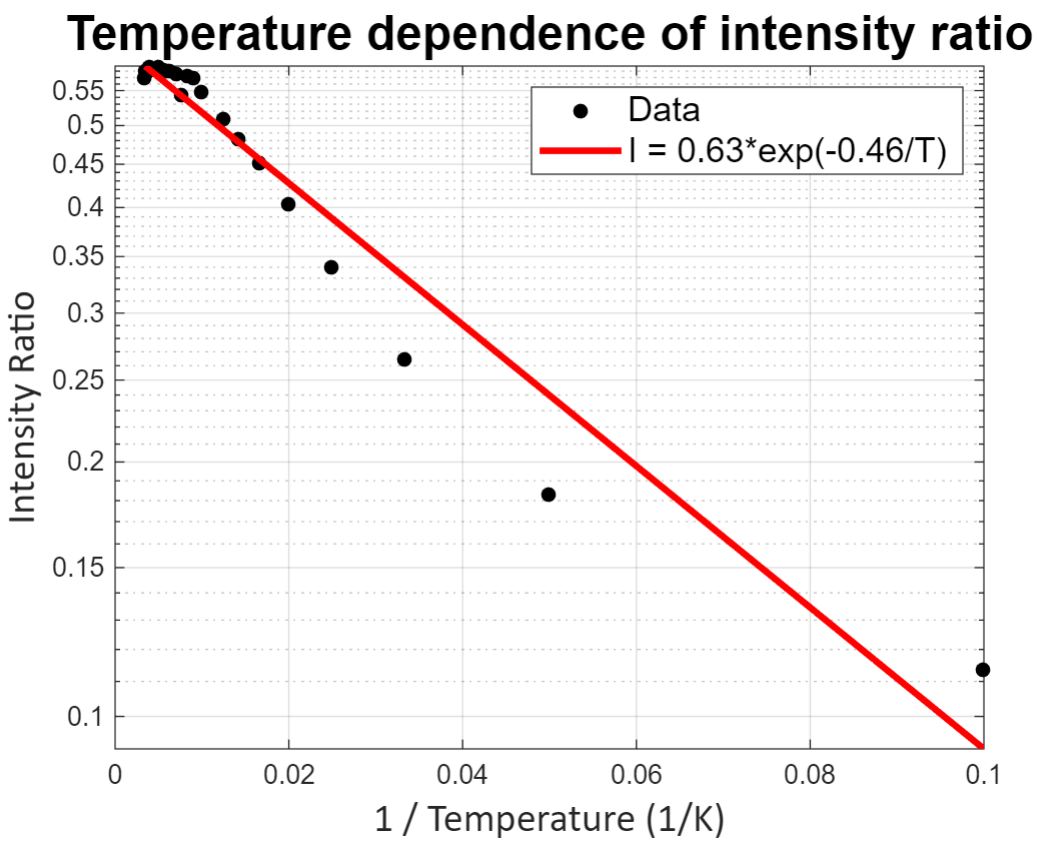



```
Fit Parameters :
T = 305 K
I0 = 2845
d_{20}/d_{10} = 0.7943
Linewidths = [1.84, 1.322]
Delta = 3.879
E1 = 1787
```

Temperature dependence of relative peak heights

```
% Load data
data =
load('C:\Users\hsjun\OneDrive\SNU\IntermedPhysLab\Photoluminescence\PLdata\Area.txt'
);

% Extract columns
T = data(:,1);    % Temperature (K)
y = data(:,2);    % Area

% Select valid data (y > 0, to avoid log of 0 or negative)
idx = (T > 0) & (y > 0);
T_fit = T(idx);
y_fit = y(idx);

% Prepare data for linear fit
X = 1 ./ T_fit;          % 1/T
Y = log10(y_fit);          % log(y)
```

```
% Perform linear fitting: Y = m*X + c
p = polyfit(X, Y, 1);    % p(1) = slope, p(2) = intercept

% physical constants
e = 1.602 * 1e-19;
kb = 1.38 * 1e-23;
kb_e = kb / e;

% fit parameters
C_fit = power(10, p(2));
B_fit = log(10)*p(2);

% Compute fitted curve for plotting
X_plot = linspace(min(X), max(X), 200);
Y_fit_curve = power(10, polyval(p, X_plot));

% Plotting in linearized space
figure;
set(gcf, 'Color', 'w'); % White background
set(gca, 'YScale', 'log');
hold on; box on;

% Plot data points
plot(X, y, 'ko', 'MarkerFaceColor', 'k', 'MarkerEdgeColor', 'k', 'MarkerSize', 5,
'DisplayName', 'Data');

% Plot fitted line
fit_label = sprintf('I = %.2f*exp(%.2f/T)', C_fit, B_fit);
plot(X_plot, Y_fit_curve, 'r-', 'LineWidth', 2.5, 'DisplayName', fit_label);

% Labels and title
xlabel('1 / Temperature (1/K)', 'FontName', 'Calibri', 'FontSize', 15);
ylabel('Intensity Ratio', 'FontName', 'Calibri', 'FontSize', 15);
title('Temperature dependence of intensity ratio', 'FontSize', 18);

% Legend
legend('Location', 'northeast', 'FontSize', 13);
grid on;
```




```
% Display fitted parameters
fprintf('Linear fit results:\n');
```

```
Linear fit results:
```

```
fprintf('C = %.3f\n', C_fit);
```

```
C = 0.628
```

```
fprintf('B = %.3f\n', B_fit);
```

```
B = -0.465
```

```
% Optional: Compute R² in linearized space
Y_model = polyval(p, X);
SS_res = sum((Y - Y_model).^2);
SS_tot = sum((Y - mean(Y)).^2);
R_squared = 1 - (SS_res / SS_tot);

fprintf('R^2 (linearized) = %.4f\n', R_squared);
```

```
R^2 (linearized) = 0.9485
```