Table of Contents

This script visualize the linear and non-linear preddictions of tau

1 - Define paths

```
workpath = '/Users/akv020/Projects/Dataverse/source/figure4';
datapath = '/Users/akv020/Projects/Dataverse/data/250m_resolution';
```

2 - Define data

Define the parameter values

```
v = [0.8, 1.3, 1.8]*1e3;
l = [2, 6, 10]*1e3;
n = [1e11, 5e11, 10e11];

% Generate all combinations of parameters
[V, N, L] = ndgrid(v, l, n);
params = [V(:), L(:), N(:)];
```

3 - Calculate saturation time

find all 250 m resolution files

```
cd(datapath)
files = dir('*.nc');

% exclude aurora files
files(endsWith({files.name},'_aurora_Q0.5.nc')) = [];
files(endsWith({files.name},'_aurora_Q0.2.nc')) = [];

% loop over all files
for i = 1:numel(files)
```

```
    cd(datapath)
    nev = ncread(files(i).name,'ne');
    nev = permute(nev,[2 1 3]);
    filename = files(i).name;
    if str2num(filename(1)) == 5
        if str2num(filename(13)) == 8 && str2num(filename(12)) == 0
            KHIstring = ['$n_p = 5 \times 10^{11}$', ' $\Delta V$ = 0.',
 num2str(str2num(filename(13))), ' km/s'];
        else
            KHIstring = ['$n_p = 5 \times 10^{11}$', ' $\Delta V$ = ',
 num2str(str2num(filename(12))),'.',num2str(str2num(filename(13))), ' km/s'];
        end
    elseif str2num(filename(4)) == 1
        if str2num(filename(11)) == 8 && str2num(filename(10)) == 0
            KHIstring = ['$n_p = 1 \times 10^{11}$', ' $\Delta V$ = 0.',
 num2str(str2num(filename(12))),'.',num2str(str2num(filename(11))), ' km/s'];
        else
            KHIstring = ['$n_p = 1 \times 10^{11}$', ' $\Delta V$ = ',
 num2str(str2num(filename(11))),'.',num2str(str2num(filename(12))), ' km/s'];
        end
    else
        if str2num(filename(11)) == 8 && str2num(filename(10)) == 0
            KHIstring = ['$n_p = 1 \times 10^{12}$', ' $\Delta V$ = 0.',
 num2str(str2num(filename(12))),'.',num2str(str2num(filename(11))), ' km/s'];
        else
            KHIstring = ['$n_p = 1 \times 10^{12}$', ' $\Delta V$ = ',
 num2str(str2num(filename(11))),'.',num2str(str2num(filename(12))), ' km/s'];
        end
    end

    KHI{i} = KHIstring;
    cd(workpath)
    % Calculate the saturation treshold and time of crossing
    [saturation_threshold, threshold_crossing_time, param] =
 saturation_finder(nev,0);
    if saturation_threshold > 1
        name{i} = files(i).name;
        ts(i) = threshold_crossing_time;
    else
        name{i} = files(i).name;
        ts(i) = NaN;
    end

end

% needed since names are not in increasing order
ts_order = [ts(10:18), ts(1:9), ts(19:end)];

% Remove NaN values
valid_indices = ~isnan(ts_order);
params = params(valid_indices, :);
y = ts_order(valid_indices)';

% Extract parameter values
```

```
v = params(:, 1);
n = params(:, 2);
l = params(:, 3);
```

3 - Calculate predicted values

Calculate estimated outputs using the first equation

```
y1_est = 115.4 * (l ./ v);

% Calculate estimated outputs using the second equation
y2_est = 0.0002130 * (n ./ v.^2) + 1.213e+05 * (l ./ v.^2) + 596.6 * ((l .*
 v.^2) ./ n);
```

4 - Calculate residuals and performance metrics

Residuals

```
residuals_y1 = y - y1_est;
residuals_y2 = y - y2_est;

% RMSE (Root Mean Squared Error)
rmse_y1 = sqrt(mean(residuals_y1.^2));
rmse_y2 = sqrt(mean(residuals_y2.^2));

% MAD (Median Absolute Deviation)
mad_y1 = median(abs(residuals_y1));
mad_y2 = median(abs(residuals_y2));

% R^2 (Coefficient of Determination)
ss_tot = sum((y - mean(y)).^2);
ss_res_y1 = sum(residuals_y1.^2);
ss_res_y2 = sum(residuals_y2.^2);
r2_y1 = 1 - (ss_res_y1 / ss_tot);
r2_y2 = 1 - (ss_res_y2 / ss_tot);
```

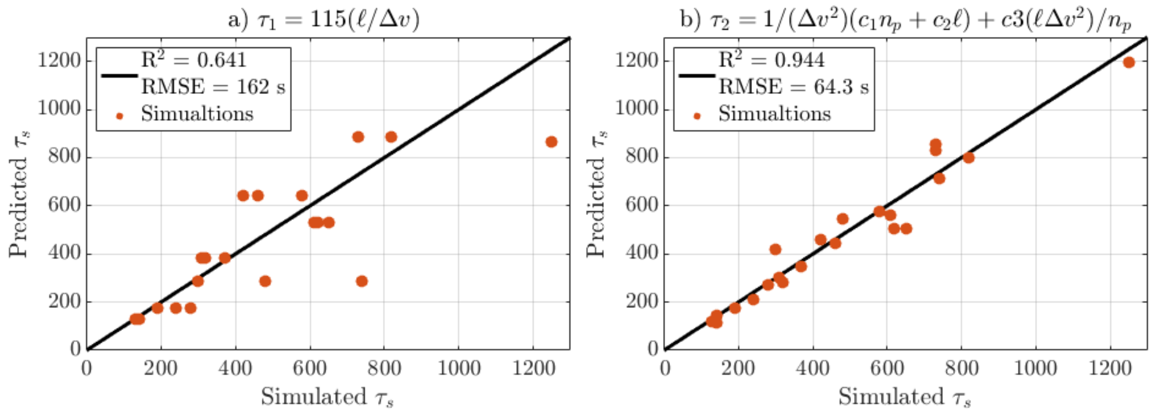
5 Plot comparison between predicted and observed KH instability growth times

```
% Define figure properties
FIG = figure('units','centimeters','position',[0,0,36.0,39.0]);
sx = 0.075;
sy = 0.075;
fz = 18;
lw = 3;
al = 0.8;
mz = 100;
colormap(inferno)

% Plot and compare the t1 growth times
```

```
subplot_tight(3,2,1,[sx, sy])
plot([0 1300],[0 1300],'k','LineWidth',lw)
hold on
scatter(y,y1_est,mz,'filled','color',[0.6350, 0.0780, 0.1840])
xlabel('Simulated $\tau_s$','Interpreter','latex','FontSize',fz)
ylabel('Predicted $\tau_s$','Interpreter','latex','FontSize',fz)
xlim([0 1300])
ylim([0 1300])
title('a) $\tau_1 = 115 ( \ell/ \Delta v)$
 ','Interpreter','latex','FontWeight','normal')
legend(['$\mathrm{R}^2$ = ',num2str(round(r2_y1,3)),
 newline, 'RMSE = ',num2str(round(rmse_y1)), '
 s'],'Simualtions','Location','northwest','FontSize',fz,'interpreter','latex')
grid on
xaxisproperties= get(gca, 'XAxis');
xaxisproperties.TickLabelInterpreter = 'latex'; % latex for x-axis
yaxisproperties= get(gca, 'YAxis');
yaxisproperties(1).TickLabelInterpreter = 'latex';   % tex for y-axis
set(gca,'fontsize',fz)

% Plot and compare the t2 growth times
subplot_tight(3,2,2,[sx, sy])
plot([0 1300],[0 1300],'k','LineWidth',lw)
hold on
scatter(y,y2_est,mz,'filled','color',[0.6350, 0.0780, 0.1840])
xlabel('Simulated $\tau_s$','FontSize',fz,'Interpreter','latex')
ylabel('Predicted $\tau_s$','FontSize',fz,'Interpreter','latex')
xlim([0 1300])
ylim([0 1300])
title('b) $\tau_2 = 1/(\Delta v^2)(c_1 n_p + c_2 \ell) + c3 (\ell \Delta v^2)/
n_p$','interpreter','latex','FontWeight','normal')
legend(['$\mathrm{R}^2$ = ',num2str(round(r2_y2,3)),
 newline, 'RMSE = ',num2str(round(rmse_y2,1)), '
 s'],'Simualtions','Location','northwest','FontSize',fz,'interpreter','latex')
grid on
xaxisproperties= get(gca, 'XAxis');
xaxisproperties.TickLabelInterpreter = 'latex'; % latex for x-axis
yaxisproperties= get(gca, 'YAxis');
yaxisproperties(1).TickLabelInterpreter = 'latex';   % tex for y-axis
set(gca,'fontsize',fz)
```



6 - Define ranges to visualize Equaiton 2

Define the ranges for l, v, and n

```
l_range = linspace(1e3, 12e3, 300); % Range for l in meters
v_range = linspace(0.5e3, 2e3, 300); % Range for v in meters/second
n_range = linspace(0.8e11, 1.2e12, 300); % Range for n in particles per cubic
 meter

% Fixed values for the subplots
```

```
l_fixed = 5e3; % l fixed at 5000 meters
v_fixed = 1.3e3; % v fixed at 1500 meters/second
n_fixed = 5e11; % n fixed at 5 * 1e11 particles per cubic meter
```

7 - Define contour levels

```
contour_levels = (1.7:0.2:3.1);
```

8 - Visualize model over the considered ranges for t1

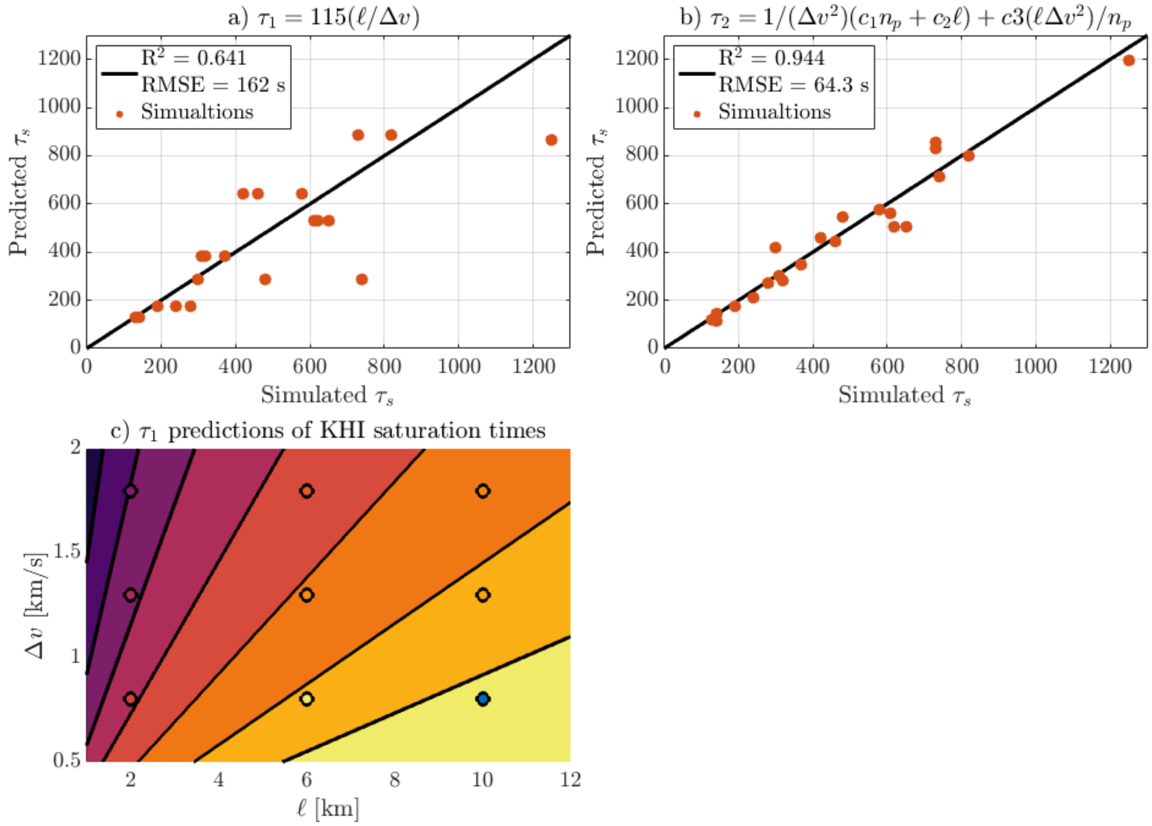
```
% Define meshgrid
[L, V] = meshgrid(l_range, v_range);

% Predict y1_est as a function of l and v for n = 1e11 (most points) for y1
 predictions
[n_fixed_idx] = find(n == 1e11);
y1_est_lv = 115.4 * (L ./ V);
y1_lv = y(n_fixed_idx);
l_lv = l(n_fixed_idx);
v_lv = v(n_fixed_idx);

% Plot first for t1 predictions
subplot_tight(3,2,3,[sx, sy])
hold on
contourf(l_range/1e3,v_range/1e3,log10(y1_est_lv),contour_levels,'k','LineWidth',lw)
clim([1.5 3.2])

% Get current colormap and color limits
cmap = colormap('inferno');
cmin = 1.5; % from clim
cmax = 3.2; % from clim
% Normalize the scatter data to fit within the color limits
scatter_data = log10(y1_lv);
norm_data = (scatter_data - cmin) / (cmax - cmin);
% Ensure the normalized data is within [0, 1]
norm_data(norm_data < 0) = 0;
norm_data(norm_data > 1) = 1;
% Map the normalized data to the colormap
color_idx = round(norm_data * (size(cmap, 1) - 1)) + 1;
scatter_colors = cmap(color_idx, :);
% Create the scatter plot of observed values
scatter(l_lv/1e3, v_lv/1e3, mz,
 scatter_colors, 'filled','MarkerEdgeColor','k','LineWidth',lw);
scatter(10,0.8, mz, 'filled','MarkerFaceColor',[0, 0.4470,
 0.7410],'MarkerEdgeColor','k','LineWidth',lw);
% Define axis and title properties
xlabel('$\ell$
 [km]','interpreter','latex','FontSize',fz,'interpreter','latex')
ylabel('$\Delta v$ [km/
s]','interpreter','latex','FontSize',fz,'interpreter','latex')
```

```
title('c) $\tau_1$ predictions of KHI saturation
 times','interpreter','latex','fontsize',fz,'FontWeight','normal')
xaxisproperties= get(gca, 'XAxis');
xaxisproperties.TickLabelInterpreter = 'latex'; % latex for x-axis
yaxisproperties= get(gca, 'YAxis');
yaxisproperties(1).TickLabelInterpreter = 'latex';   % tex for y-axis
set(gca,'fontsize',fz)
```



9 - Visualize model over the considered ranges for t2 as a function of l and v with n fixed

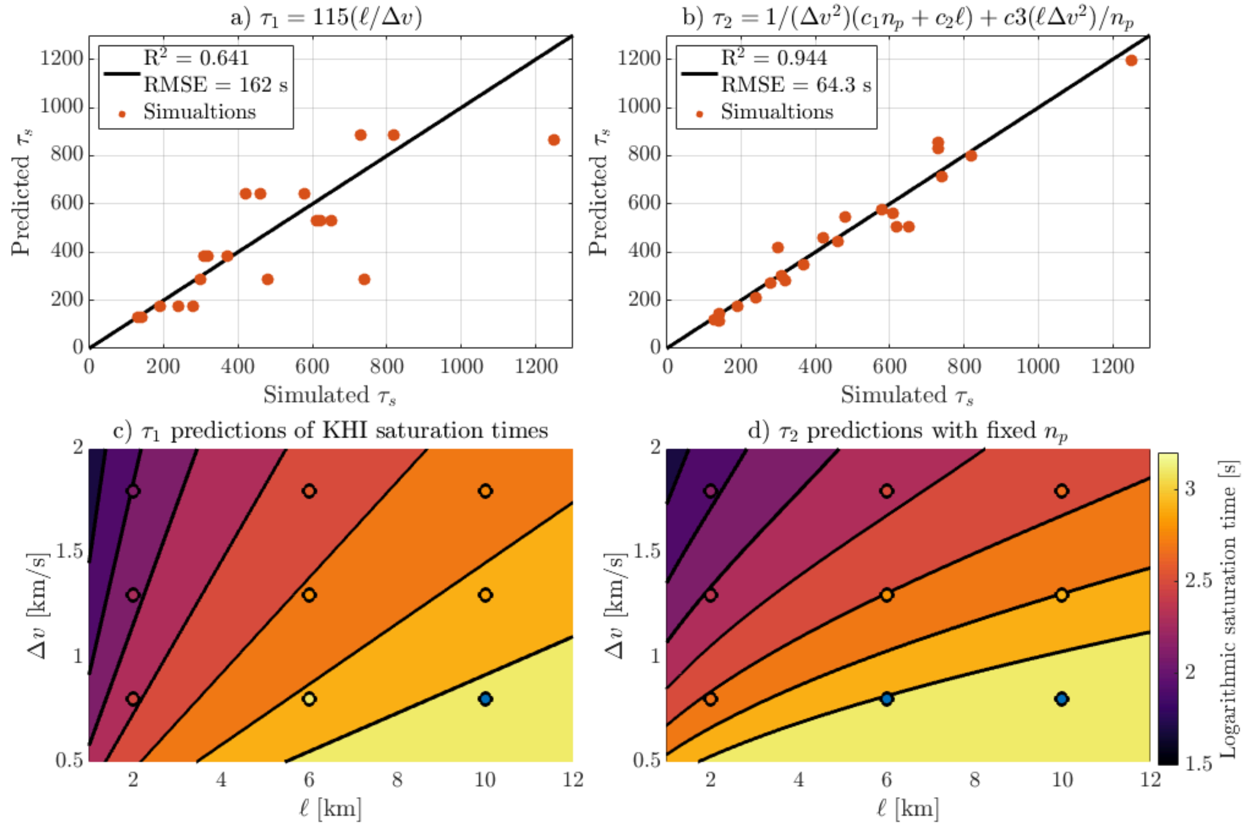
```
% Get obsevred values
[n_fixed_idx] = find(n == 5e11); % Define indexes with n = 5e11 for y2
 predictions
y2_lv = y(n_fixed_idx); %

% Get predicted values t2
[L, V] = meshgrid(l_range, v_range);
y2_est_lv = 0.0002130 * (n_fixed ./ V.^2) + 1.213e+05 * (L ./ V.^2) + 596.6 *
 ((L .* V.^2) ./ n_fixed);


% Plot the t2 predictions
subplot_tight(3,2,4,[sx, sy])
pcolor(l_range/1e3,v_range/1e3,log10(y2_est_lv))
hold on
contourf(l_range/1e3,v_range/1e3,log10(y2_est_lv),contour_levels,'k','LineWidth',lw)
clim([1.5 3.2])

% Normalize the scatter data to fit within the color limits
scatter_data = log10(y2_lv);
norm_data = (scatter_data - cmin) / (cmax - cmin);
% Ensure the normalized data is within [0, 1]
norm_data(norm_data < 0) = 0;
norm_data(norm_data > 1) = 1;
% Map the normalized data to the colormap
color_idx = round(norm_data * (size(cmap, 1) - 1)) + 1;
scatter_colors = cmap(color_idx, :);
% Create the scatter plot
scatter(l(n_fixed_idx)/1e3, v(n_fixed_idx)/1e3, mz,
 scatter_colors, 'filled','MarkerEdgeColor','k','LineWidth',lw);
scatter([6 10],[0.8 0.8], mz, 'filled','MarkerFaceColor',[0, 0.4470,
 0.7410],'MarkerEdgeColor','k','LineWidth',lw);

% Define axis and title properties
xlabel('$\ell$ [km]','interpreter','latex','FontSize',fz)
ylabel('$\Delta v$ [km/s]','interpreter','latex','FontSize',fz)
c = colorbar;
c.Label.String = 'Logarithmic saturation time [s]';
c.FontSize = fz;
c.Label.Interpreter = 'latex';
set(c,'TickLabelInterpreter','latex')
set(c,'Position',[0.9319,0.3807,0.0157,0.233]);
title('d) $\tau_2$ predictions with fixed $n_p
$','interpreter','latex','fontsize',fz,'FontWeight','normal')
xaxisproperties= get(gca, 'XAxis');
xaxisproperties.TickLabelInterpreter = 'latex'; % latex for x-axis
yaxisproperties= get(gca, 'YAxis');
yaxisproperties(1).TickLabelInterpreter = 'latex';   % tex for y-axis
set(gca,'fontsize',fz)
```



10 - Visualize model over the considered ranges for t2 as a function of l and n with v fixed

```
% Get obsevred values
[v_idx] = find(v == 1300); % Define indexes with v = 1.3 km/s for y2
 predictions
y2_ln = y(v_idx);
```

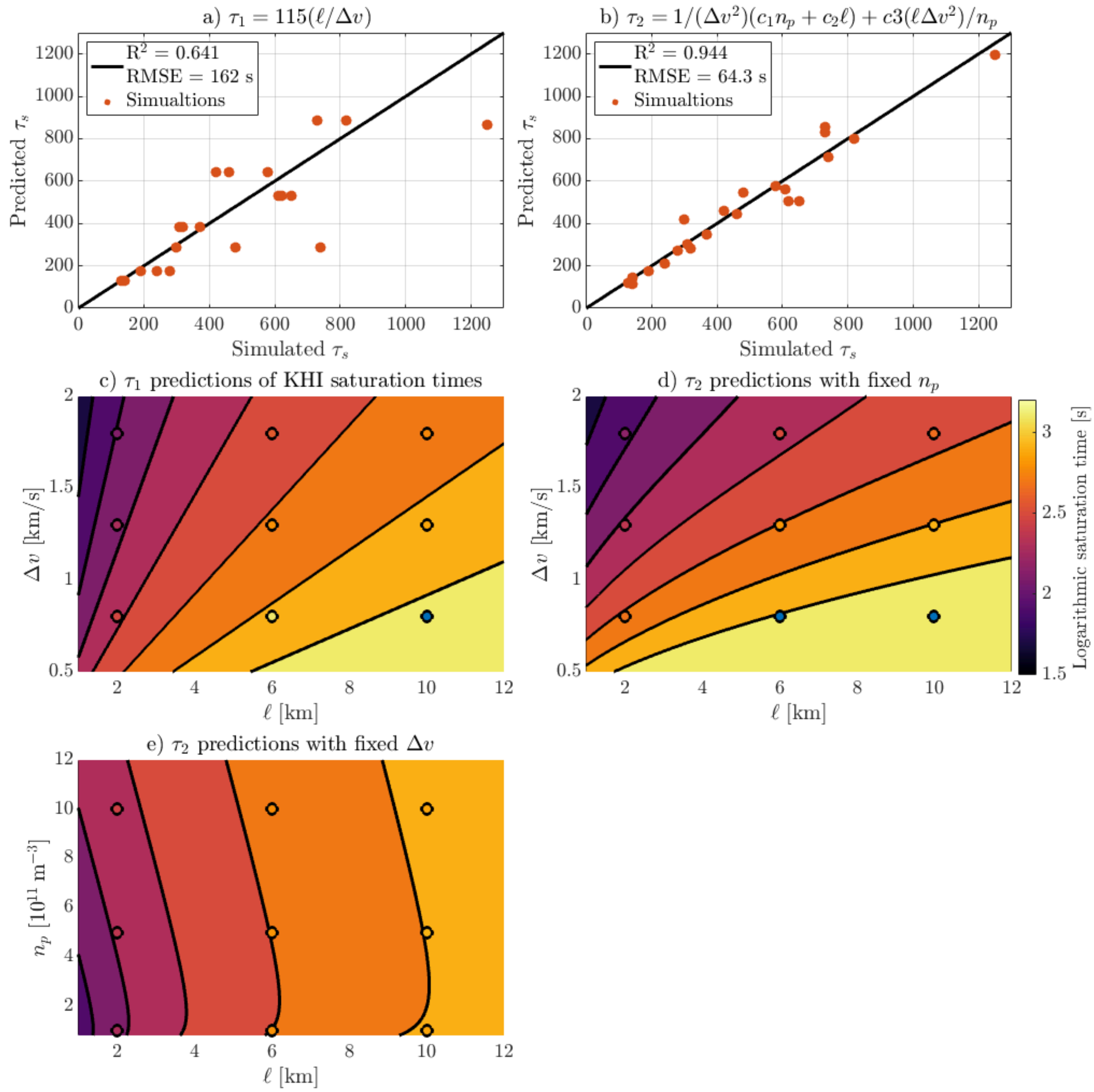
```
l_ln = l(v_idx);
n_ln = n(v_idx);

% Get predicted values t2
[L, N] = meshgrid(l_range, n_range);
y2_est_ln = 0.0002130 * (N ./ v_fixed.^2) + 1.213e+05 * (L ./ v_fixed.^2) +
 596.6 * ((L .* v_fixed.^2) ./ N);

% Plot the t2 predictions
subplot_tight(3,2,5,[sx, sy])
pcolor(l_range/1e3,n_range/1e11,log10(y2_est_ln))
hold on
contourf(l_range/1e3,n_range/1e11,log10(y2_est_ln),contour_levels,'k','LineWidth',lw)
clim([1.5 3.2])

% Normalize the scatter data to fit within the color limits
scatter_data = log10(y2_ln);
norm_data = (scatter_data - cmin) / (cmax - cmin);
% Ensure the normalized data is within [0, 1]
norm_data(norm_data < 0) = 0;
norm_data(norm_data > 1) = 1;
% Map the normalized data to the colormap
color_idx = round(norm_data * (size(cmap, 1) - 1)) + 1;
scatter_colors = cmap(color_idx, :);
% Create the scatter plot
scatter(l_ln/1e3, n_ln/1e11, mz,
 scatter_colors, 'filled','MarkerEdgeColor','k','LineWidth',lw);

% Define axis and title properties
xlabel('$\ell$ [km]','interpreter','latex','FontSize',fz)
ylabel('$n_p$ $[10^{11} \,
\mathrm{m}^{-3}]$','interpreter','latex','FontSize',fz)
title('e) $\tau_2$ predictions with fixed $\Delta v
$','interpreter','latex','fontsize',fz,'FontWeight','normal')
xaxisproperties= get(gca, 'XAxis');
xaxisproperties.TickLabelInterpreter = 'latex'; % latex for x-axis
yaxisproperties= get(gca, 'YAxis');
yaxisproperties(1).TickLabelInterpreter = 'latex';   % tex for y-axis
set(gca,'fontsize',fz)
```



11 - Visualize model over the considered ranges for t2 as a function of v and n with l fixed

```
% Get obsevred values
[l_idx] = find(l == 6000); % Define indexes with l = 6 km for y2 predictions
y2_vn = y(l_idx);
v_vn = v(l_idx);
```

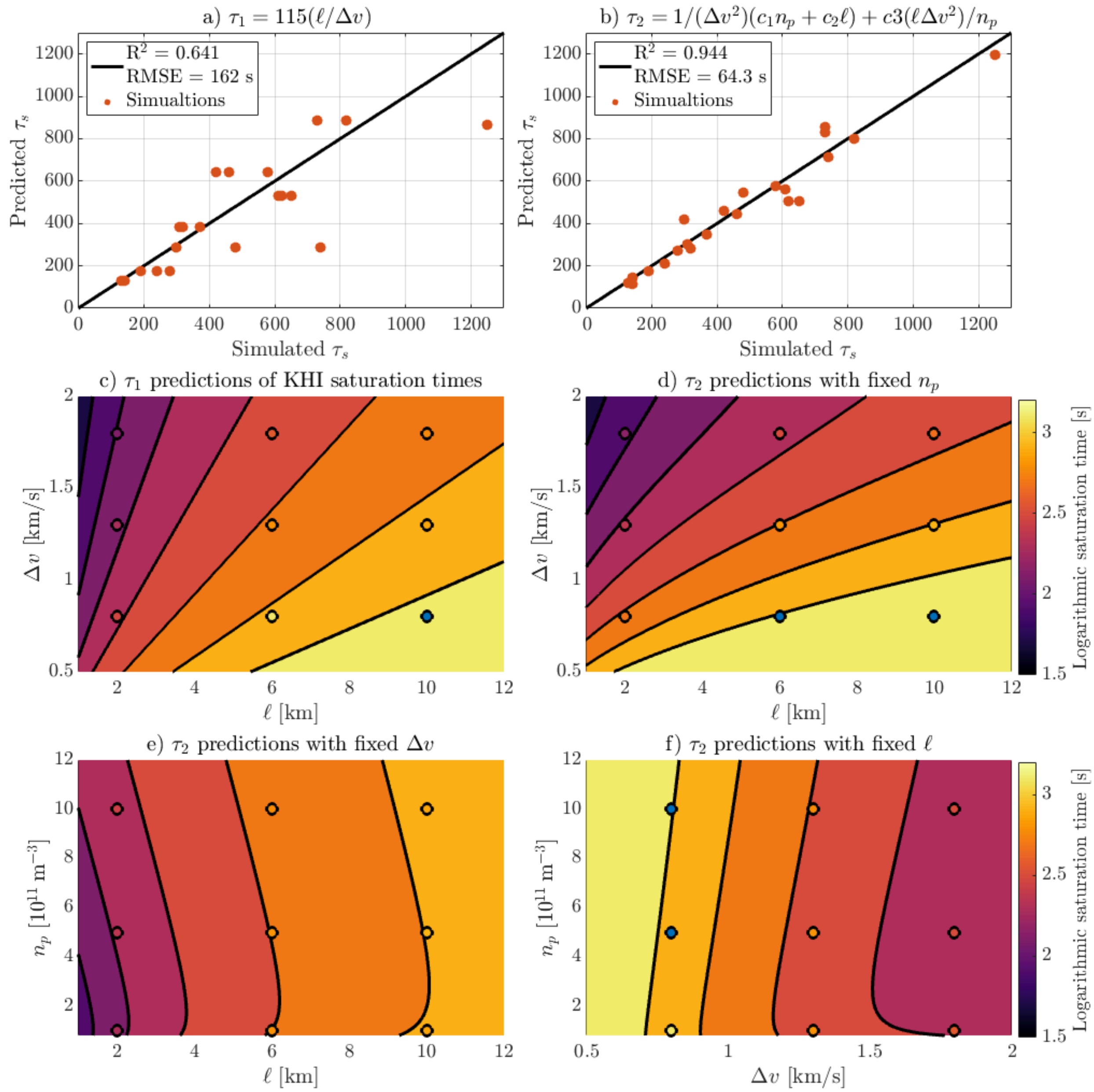
```
n_vn = n(l_idx);

% Get predicted values t2
[V, N] = meshgrid(v_range, n_range);
y2_est_vn = 0.0002130 * (N ./ V.^2) + 1.213e+05 * (l_fixed ./ V.^2) + 596.6 *
 ((l_fixed .* V.^2) ./ N);

% Plot the t2 predictions
subplot_tight(3,2,6,[sx, sy])
pcolor(v_range/1e3,n_range/1e11,log10(y2_est_vn))
hold on
contourf(v_range/1e3,n_range/1e11,log10(y2_est_vn),contour_levels,'k','LineWidth',lw)
clim([1.5 3.2])

% Normalize the scatter data to fit within the color limits
scatter_data = log10(y2_vn);
norm_data = (scatter_data - cmin) / (cmax - cmin);
% Ensure the normalized data is within [0, 1]
norm_data(norm_data < 0) = 0;
norm_data(norm_data > 1) = 1;
% Map the normalized data to the colormap
color_idx = round(norm_data * (size(cmap, 1) - 1)) + 1;
scatter_colors = cmap(color_idx, :);
% Create the scatter plot
scatter(v_vn/1e3, n_vn/1e11, mz,
 scatter_colors, 'filled','MarkerEdgeColor','k','LineWidth',lw);
scatter([0.8 0.8],[5 10], mz, 'filled','MarkerFaceColor',[0, 0.4470,
 0.7410],'MarkerEdgeColor','k','LineWidth',lw);

% Define axis and title properties
xlabel('$\Delta v$ [km/s]','interpreter','latex','FontSize',fz)
ylabel('$n_p$ $[10^{11} \,
\mathrm{m}^{-3}]$','interpreter','latex','FontSize',fz)
c = colorbar;
c.Label.String = 'Logarithmic saturation time [s]';
c.FontSize = fz;
c.Label.Interpreter = 'latex';
set(c,'Position',[0.9319,0.07325,0.0157,0.233]);
set(c,'TickLabelInterpreter','latex')
title('f) $\tau_2$ predictions with fixed $\ell
$','interpreter','latex','fontsize',fz,'FontWeight','normal')
xaxisproperties= get(gca, 'XAxis');
xaxisproperties.TickLabelInterpreter = 'latex'; % latex for x-axis
yaxisproperties= get(gca, 'YAxis');
yaxisproperties(1).TickLabelInterpreter = 'latex';   % tex for y-axis
set(gca,'fontsize',fz)
```



Published with MATLAB® R2022b