filter_xyz.m

```
fileIn    = 'preproperties_predictions.csv';
fileOut = 'filtered_preproperties_predictions.csv';

T = readtable(fileIn, 'FileType', 'text', 'TextType', 'string');

origNames = T.Properties.VariableNames;
cleanNames = matlab.lang.makeValidName(strtrim(origNames));
T.Properties.VariableNames = cleanNames;

need = {'delta_energy','delta_modulus_re','formation'};
missing = setdiff(need, T.Properties.VariableNames);
if ~isempty(missing)
      error('Missing required columns: %s\nAvailable columns: %s, ...
            strjoin(missing, ', '), strjoin(T.Properties.VariableNames, ', '));
end

xrange = [2.68, 2.685];          % delta_energy
yrange = [28.9, 29.95];          % delta_modulus1
zrange = [-609, -608.5];          % formation

mask =    T.delta_energy    >= xrange(1) & T.delta_energy    <= xrange(2) & ...
            T.delta_modulus_re>= yrange(1) & T.delta_modulus_re<= yrange(2) & ...
            T.formation          >= zrange(1) & T.formation          <= zrange(2);

T_out = T(mask, :);

writetable(T_out, fileOut);

fprintf('Filtered %d rows of data and saved to: %s\n', height(T_out), fileOut);
```

plot_heatmap_2d.m

```
function plot_crg_heatmap(fileIn)

    if nargin < 1
        fileIn = 'filtered_preproperties_predictions.csv';
    end

    T = tryReadTableFlexible(fileIn);

    rawNames = T.Properties.VariableNames;

    normNames = lower(regexprep(rawNames, '[^a-zA-Z0-9]+', '_'));
    T.Properties.VariableNames = matlab.lang.makeUniqueStrings(normNames);
    names = T.Properties.VariableNames;

    xName    = pickFirst(names, {'delta_energy'});
    yName    = pickFirst(names, {'delta_modulus_re','delta_modulus1','delta_modulus','delta_modulus_reo'});
    crgName = pickFirst(names, {'xgb_predicted_crg','cvmodel_predicted_crg','crg'});

    x = []; y = []; crg = [];

    if ~isempty(xName) && ~isempty(yName) && ~isempty(crgName)
        x   = toNum(T.(xName));
        y   = toNum(T.(yName));
        crg = toNum(T.(crgName));
        fprintf('Using columns: x=%s, y=%s, crg=%s\n', xName, yName, crgName);
    else
        A = buildNumericMatrix(T);
        validCols = find(sum(isfinite(A),1) >= 3);
        if numel(validCols) >= 3
            x   = A(:, validCols(1));
            y   = A(:, validCols(2));
            crg = A(:, validCols(3));
            fprintf('No matching named columns found, using the first three numeric columns instead.\n');
        else
            need = 'x ∈ {delta_energy}, y ∈ {delta_modulus_re|delta_modulus1|delta_modulus|delta_modulus_reo}, crg ∈ {xgb_predicted_crg|cvmodel_predicted_crg|crg} or 3 numeric columns (x y crg)';
            error('Unrecognized columns: expected %s. Available columns: %s',
```

```
need, strjoin(rawNames, ', '));
            end
        end

        valid = isfinite(x) & isfinite(y) & isfinite(crg);
        x = x(valid); y = y(valid); crg = crg(valid);
        if numel(x) < 3
            error('Insufficient valid points (more than three required)');
        end

        crg = min(max(crg, 0), 4.5);

        DT    = delaunayTriangulation(x, y);
        tri = DT.ConnectivityList; P = DT.Points;
        figure('Color','w');
        h = trisurf(tri, P(:,1), P(:,2), crg);
        view(2); shading interp; set(h,'EdgeColor','none');
        axis equal tight; box on;
        xlabel('delta\_energy');
        ylabel(strrep(yName,'_','\_'));
        colormap(parula);
        caxis([0 4.5]);
        cb = colorbar; cb.Label.String = 'crg';
        cb.Ticks = 0:0.5:4.5;
        title('CRG Heatmap');

        xlim([2.6805 2.6827]);
        ylim([28.91 28.93]);

        hold on; plot(x, y, '.', 'MarkerSize', 6, 'Color', [0 0 0]); hold off;

        outPng = 'crg_heatmap.png';
        try
            exportgraphics(gcf, outPng, 'Resolution', 300);
        catch
            print(gcf, '-dpng', '-r300', outPng);
        end
        fprintf('Heatmap saved to: %s\n', outPng);
end

function name = pickFirst(names, candidates)
        name = '';
        for i = 1:numel(candidates)
            if ismember(candidates{i}, names)
```

```
            name = candidates{i};
            return;
        end
    end
end

function T = tryReadTableFlexible(fileIn)
    try
        opts = detectImportOptions(fileIn, 'FileType','text', ...
            'VariableNamingRule','preserve');
        try opts.Delimiter = {'\t', ',', ';', ' ', '|'}; catch, end
        T = readtable(fileIn, opts, 'TextType','string', ...
            'ReadVariableNames', true, 'MultipleDelimsAsOne', true);
    catch
        T = readtable(fileIn, 'FileType','text', 'TextType','string', ...
            'Delimiter', {'\t', ',', ';', ' ', '|'}, ...
            'ReadVariableNames', true, 'MultipleDelimsAsOne', true, ...
            'VariableNamingRule','preserve');
    end

    if width(T) == 1
        fid = fopen(fileIn, 'r');
        assert(fid>0, 'Unable to open file: %s', fileIn);
        C = textscan(fid, '%s', 1, 'Delimiter', '\n', 'Whitespace','');
        hdr = C{1}{1}; fclose(fid);

        if contains(hdr, sprintf('\t')), delim = '\t';
        elseif contains(hdr, ','),       delim = ',';
        elseif contains(hdr, ';'),       delim = ';';
        elseif contains(hdr, '|'),       delim = '|';
        else,                            delim = ' ';
        end

        T = readtable(fileIn, 'FileType','text', 'TextType','string', ...
            'Delimiter', delim, 'ReadVariableNames', true, ...
            'MultipleDelimsAsOne', true, 'VariableNamingRule','preserve');
    end
end

function v = toNum(col)
    if isnumeric(col), v = double(col);
    else,              v = str2double(string(col));
    end
end
```

```
function A = buildNumericMatrix(T)
    names = T.Properties.VariableNames;
    n = height(T); m = width(T);
    A = nan(n, m);
    for j = 1:m
        A(:, j) = toNum(T.(names{j}));
    end
end
```

plot_heatmap_3d.m

```
function plot_3d_thermo_umhcolor_roi

csvFile          = 'preproperties_predictions.csv';
outFormation = 'surface3D_formation_umhcolor_roi.png';
outEnthalpy    = 'surface3D_enthalpy_umhcolor_roi.png';

xRange = [2.67 2.69];
yRange = [28.8 29.0];
zRangeForm = [-610 -608];
zRangeEnth = [];

useReverse = false;
showPoints = true;
viewAZEL      = [45 28];
lightOn        = true;
faceAlpha    = 1.0;

climRange    = [0 7];
cbTicks        = 0:1:7;

set(groot,'DefaultAxesFontName','Times New Roman');
set(groot,'DefaultTextFontName','Times New Roman');

T = readWithPreserve(csvFile);
vn = T.Properties.VariableNames;

x    = colD(T,'delta_energy');
y    = colD(T,'delta_modulus_re');
zF = colD(T,'formation');
zH = colD(T,'enthalpy');

try
      C_orig = colD(T,'CVModel_Predicted_crg');
catch
      C_orig = colD(T,'XGB_Predicted_crg');
end

C = min(max(C_orig, climRange(1)), climRange(2));
C(isnan(C)) = climRange(1);

in = x >= xRange(1) & x <= xRange(2) & y >= yRange(1) & y <= yRange(2);
x = x(in); y = y(in); zF = zF(in); zH = zH(in); C = C(in);
```

```
if numel(x) < 3
    error('Too few data points within ROI (<3). Please expand xRange/yRange or check the data.');
end

tri = delaunay(x, y);

cmap = makeRYGB(256);
if useReverse, cmap = flipud(cmap); end

figure('Color','w','Position',[100 100 780 680]);
hs = trisurf(tri, x, y, zF, C);
set(hs,'EdgeColor','none','FaceColor','interp','FaceAlpha',faceAlpha);
xlabel('delta\_energy'); ylabel('delta\_modulus_re'); zlabel('formation');
title('3D surface: formation (colored by Predicted umh)');
colormap(cmap);
cb = colorbar; cb.Label.String = 'Predicted umh'; cb.Ticks = cbTicks;
caxis(climRange);
xlim(xRange); ylim(yRange);
if ~isempty(zRangeForm), zlim(zRangeForm); end
shading interp; view(viewAZEL); axis tight; box on; grid on;
hold on;
if showPoints
    scatter3(x, y, zF, 18, C, 'filled', 'MarkerEdgeColor','k', 'LineWidth',0.2);
end
hold off;
if lightOn, camlight headlight; material dull; end
saveFigure(outFormation);

figure('Color','w','Position',[920 100 780 680]);
hs2 = trisurf(tri, x, y, zH, C);
set(hs2,'EdgeColor','none','FaceColor','interp','FaceAlpha',faceAlpha);
xlabel('delta\_energy'); ylabel('delta\_modulus_re'); zlabel('enthalpy');
title('3D surface: enthalpy (colored by Predicted umh)');
colormap(cmap);
cb = colorbar; cb.Label.String = 'Predicted umh'; cb.Ticks = cbTicks;
caxis(climRange);
xlim(xRange); ylim(yRange);
if ~isempty(zRangeEnth), zlim(zRangeEnth); end
shading interp; view(viewAZEL); axis tight; box on; grid on;
hold on;
if showPoints
    scatter3(x, y, zH, 18, C, 'filled', 'MarkerEdgeColor','k', 'LineWidth',0.2);
end
```

```
hold off;
if lightOn, camlight headlight; material dull; end
saveFigure(outEnthalpy);

fprintf('Saved:\n    %s\n    %s\n', outFormation, outEnthalpy);
end

function T = readWithPreserve(csvFile)
    try
        opts = detectImportOptions(csvFile);
        try, opts.VariableNamingRule = 'preserve'; catch, end
        T = readtable(csvFile, opts);
    catch
        T                                                                                    =
readtable(csvFile,'FileType','text','Delimiter',{'\t',','},'ReadVariableNames',true);
    end
end

function v = colD(T, name)
    vn = T.Properties.VariableNames;
    idx = find(strcmpi(vn, name), 1);
    if isempty(idx)
        vnNorm = lower(regexprep(vn,'[^a-z0-9]',''));
        target = lower(regexprep(name,'[^a-z0-9]',''));
        idx = find(strcmp(vnNorm, target), 1);
    end
    if isempty(idx)
        error('Required column not found: %s; available columns: %s', name,
strjoin(vn, ', '));
    end
    v = double(T{:, idx});
end

function cmap = makeRYGB(N)
    if nargin < 1, N = 256; end
    stops = [0; 1/3; 2/3; 1];
    cols  = [1 0 0; 1 1 0; 0 1 0; 0 0 1];
    xi = linspace(0,1,N)';
    r = interp1(stops, cols(:,1), xi, 'linear');
    g = interp1(stops, cols(:,2), xi, 'linear');
    b = interp1(stops, cols(:,3), xi, 'linear');
    cmap = [r, g, b];
end
```

```
function saveFigure(path)
	if exist('exportgraphics','file')
		exportgraphics(gcf, path, 'Resolution',300);
	else
		set(gcf,'PaperPositionMode','auto');
		print(gcf, path, '-dpng', '-r300');
	end
end
```