```
clear all;
close all;
clc;

% Root finding parameters
convergence_thresh = 1e-6;
maxiter = 100;
vector = -10:10;

% Initialize solution vectors
sol = zeros(21, 21, 2);
step_sols = zeros(3, 6, 2);

% Create x_g grid
[C, R] = meshgrid(vector, vector);
grid = cat(3, R, C);

saved_sol = 0;
for row=vector
    for col=vector
        x_g = squeeze(grid(row+11, col+11, :));
        save = false;
        if isequal(x_g, [4; -4]) || isequal(x_g, [6; 0]) || isequal(x_g,
[-5; 5])
            step_sol = zeros(2, 6);
            save = true;
            saved_sol = saved_sol + 1;
        end

        step = inf;
        iter = 1;
        while step > convergence_thresh && iter < maxiter

            if save == true && iter <= 6
                step_sol(:, iter) = x_g;
            end

            x = x_g;
            f = [x(1) + x(2) + x(1)*x(2) + 5;
            x(1)^2 + 2*x(2) - x(2)^2 - 2];
            dfdx = [1+x(2), 1 + x(1);
            2*x(1), 2-2*x(2)];

            x_g = x_g - dfdx\f;
            step = norm(x_g - x);

            iter = iter + 1;
        end
        if iter == maxiter
            x_g = [inf, inf];
        end
```

```
        % Save the solution
        sol(row+11, col+11, :) = x_g;
        if save == true
            step_sols(saved_sol, :, :) = step_sol.'; % TODO
        end

    end
end

% Reshape to columns
col_sol = reshape(permute(sol, [3 1 2]), size(sol, 3), []).';
col_guess = reshape(permute(grid, [3 1 2]), size(grid, 3), []).';

% Remove divergneces
valid_indices = all(~isnan(col_sol), 2);
col_sol = col_sol(valid_indices, :);
col_guess = col_guess(valid_indices, :);

% Round for finding uniqueness
col_sol = round(col_sol, 4);
[unique_sols, ~, labels] = unique(col_sol, 'rows');
num_unique_sols = size(unique_sols, 1);


% USED GEMINI FOR SOME BASIC PLOTTING
figure;
hold on;

% Define the symbols and colors we'll cycle through
symbols = ['o', 's', '^', 'v', 'P', '*', 'X', 'D'];


for k = 1:num_unique_sols
    % Get the symbol
    sym = symbols(mod(k-1, length(symbols)) + 1);

    % Find the indices of relevant guesses
    indices_for_this_group = (labels == k);

    % Get  coordinates
    guesses_to_plot = col_guess(indices_for_this_group, :);

    % Plot all initial guesses
    plot(guesses_to_plot(:, 1), guesses_to_plot(:, 2), ...
        'Marker', sym, ...
        'LineStyle', 'none');

    % Get the solution
    sol = unique_sols(k, :);

    % Plot plot it
    plot_handles(k) = plot(sol(1), sol(2), ...
        'Marker', sym, ...
        'MarkerEdgeColor', 'r', ...
```

```
        'MarkerSize', 12, ...
        'LineWidth', 2);
end

hold off;
margin = 0.10;
x_min = min(vector);
x_max = max(vector);
y_min = min(vector);
y_max = max(vector);
x_range = x_max - x_min;
y_range = y_max - y_min;
x_pad = x_range * margin;
y_pad = y_range * margin;
xlim([x_min - x_pad, x_max + x_pad]);
ylim([y_min - y_pad, y_max + y_pad]);
axis equal;
xlabel('X-coordinate');
ylabel('Y-coordinate');
title('Initial Guesses Grouped by Convergent Solution');

% Print the saved steps
steps_1 = step_sols(1, :, :)
steps_2 = step_sols(2, :, :)
steps_3 = step_sols(3, :, :)


%%%%%%%%%%%%%%%%%% Explanation %%%%%%%%%%%%%%%%%%%%%%%%
% This plot shows a saddle, where guesses directly on the saddle do not
% converge since the slope there is zero

% 6 guesses diverged, meaning 21^2-6 = 435 guesses converged
% Or maybe you're asking how many local minima there are in this region and
% the answer to that is 2
```

```
Warning: Matrix is singular to working precision.
Warning: Matrix is singular to working precision.
Warning: Matrix is singular to working precision.
Warning: Matrix is singular, close to singular or badly scaled. Results may
be
inaccurate. RCOND = NaN.
Warning: Matrix is singular to working precision.
Warning: Matrix is singular to working precision.
Warning: Matrix is singular to working precision.
Warning: Matrix is singular to working precision.
Warning: Matrix is singular to working precision.
Warning: Matrix is singular, close to singular or badly scaled. Results may
be
inaccurate. RCOND = NaN.
Warning: Matrix is singular to working precision.
Warning: Matrix is singular to working precision.

steps_1(:,:,1) =
```

```
   -5.0000   -2.8182   -2.1319   -2.0548   -2.0542         0
```

```
steps_1(:,:,2) =
```

```
    5.0000    3.2727    2.8127    2.7935    2.7944         0
```

```
steps_2(:,:,1) =
```

```
    4.0000    3.1429    3.1184    3.1320    3.1320         0
```

```
steps_2(:,:,2) =
```

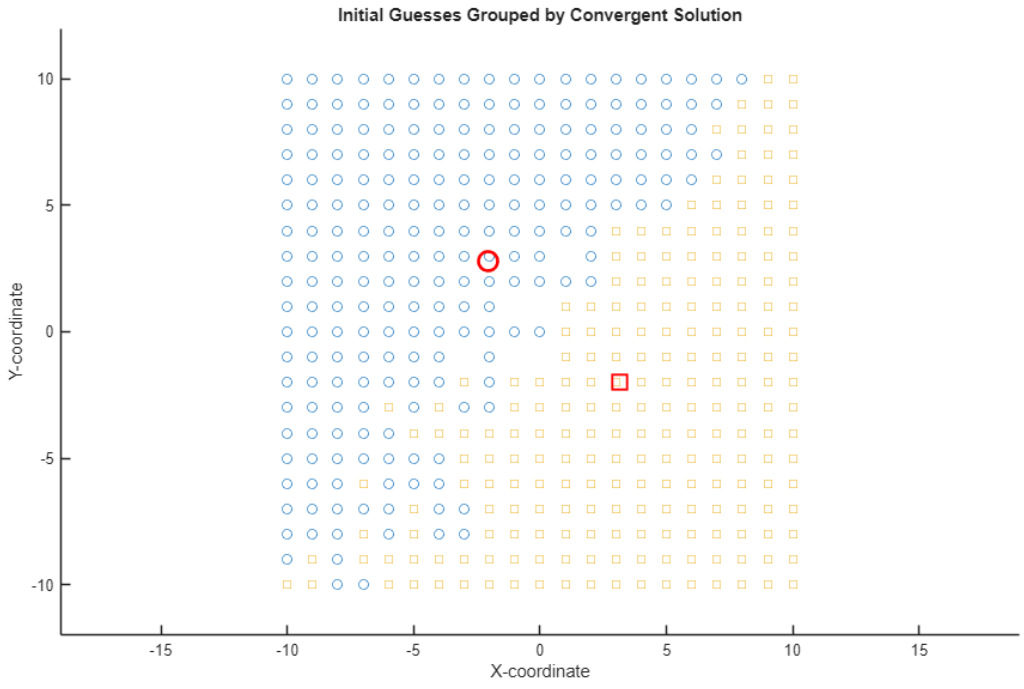
```
   -4.0000   -2.3143   -1.9733   -1.9680   -1.9681         0
```

```
steps_3(:,:,1) =
```

```
    6.0000    3.3659    3.0274    3.1340    3.1320         0
```

```
steps_3(:,:,2) =
```

```
         0   -1.1951   -1.9313   -1.9685   -1.9681         0
```



Published with MATLAB® R2025a