```
function [t_vals, x_vals, y_vals, X, delta_xy, delta_t] = general_heat_2D(t_iter, ↙
x_iter, y_iter, alpha)

    t_0 = 0;
    t_f = 8;
    t_vals = linspace(t_0, t_f, t_iter);

    L = 4;
    H = 4;
    X = zeros(x_iter, y_iter, t_iter);
    x_0 = @(x, y) double(y >= x);
    x_vals = linspace(0,L,x_iter);
    y_vals = linspace(0,H,x_iter);
    for i = 1:length(x_vals)
        for j = 1:length(y_vals)
            X(i,j,1) = x_0(x_vals(i),y_vals(j));
        end
    end

    X(1,:,1) = X(2,:,1);
    X(end,:,:) = 0;
    X(:,1,1) = X(:,2,1);
    for x_num = 1:x_iter
        X(x_num,end,:) = exp(-t_vals);
    end

    delta_t = t_vals(2) - t_vals(1);
    delta_xy = x_vals(2) - x_vals(1);

    if delta_t > 0.25*delta_xy^2/alpha
        fprintf('Upper Bound: %g\n', 0.25*delta_xy^2/alpha)
        fprintf('Delta t: %g\n', delta_t)
        disp("Unstable, increase t_iter or decrease x_iter")
        return
    end

    for i = 2:t_iter
        X(2:x_iter-1, 2:x_iter-1, i) = alpha*delta_t/delta_xy^2* ...
            (X(3:x_iter, 2:x_iter-1, i-1) + X(1:x_iter-2, 2:x_iter-1, i-1) + ...
            X(2:x_iter-1, 3:x_iter, i-1) + X(2:x_iter-1, 1:x_iter-2, i-1) - 4*X(2: ↙
x_iter-1, 2:x_iter-1, i-1)) +...
            (exp(-t_vals(i))*delta_t + 1)*X(2:x_iter-1, 2:x_iter-1, i-1);
        X(1,:,i) = X(2,:,i);
        X(:,1,i) = X(:,2,i);
    end

    str = "";
    if t_iter >= 1000
        str = "Caution: At current number of iterations, animation may take a long ↙
```

```
time. ";
    end
    decide = input(str + "Press enter to exit. Type 1 for Finite Difference↙
Animation", "s");
    switch decide
        case "1"
            animation_speed = 0.0;

            % Create meshgrid for surface plot
            [X_grid, Y_grid] = meshgrid(x_vals, y_vals);   % Assuming x and y are 1D↙
vectors

            figure;
            hold on
            h = surf(X_grid, Y_grid, X(:,:,1), "FaceAlpha", 0.25, 'FaceColor', 'b');
            xlabel('x');
            ylabel('y');
            zlabel('Temperature');
            title('2D Heat Equation Solution');
            % Fix z-axis limits to avoid recalculating each frame
            zlim([min(X(:)), max(X(:))]);
            view(3);

            % Improve rendering performance by reducing overhead
            set(gcf, 'Renderer', 'painters');

            for timestep = 1:size(X,3)
                set(h, 'ZData', X(:,:,timestep));
                title(sprintf('Temperature Distribution at Time Step: %d',↙
timestep));
                drawnow limitrate;
                pause(animation_speed);
            end
        case isempty(decide)
            return
    end

end
```