```
function [t_vals, x_vals, X_analytical, X, U_sol, t_sol, delta_x, delta_t] =↙
basic_heat(t_iter, x_iter)
    t_0 = 0;
    t_f = 2;
    t_vals = linspace(t_0, t_f, t_iter);

    L = 3;
    X = zeros(t_iter, x_iter);
    x_0 = @(x) sin(pi*x) + sin(2*pi*x);
    x_vals = linspace(0,L,x_iter);
    X(1, :) = x_0(x_vals);
    X(:,1) = 0;
    X(:,end) = 0;

    K = zeros(4, x_iter);
    delta_t = t_vals(2) - t_vals(1);
    delta_x = x_vals(2) - x_vals(1);
    alpha = 2;

    if delta_t > 0.5*delta_x^2/alpha
        fprintf('Upper Bound: %g\n', 0.5*delta_x^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
        K(1,2:x_iter-1) = alpha*(X(i-1,3:x_iter) - 2*X(i-1,2:x_iter-1) + X(i-1,1:↙
x_iter-2))/delta_x^2;
        K(2,2:x_iter-1) = alpha*((X(i-1,3:x_iter) + (delta_t/2)*K(1,3:x_iter)) - 2*(X↙
(i-1,2:x_iter-1) + (delta_t/2)*K(1,2:x_iter-1)) + (X(i-1,1:x_iter-2) + (delta_t/2)*K↙
(1,1:x_iter-2)))/delta_x^2;
        K(3,2:x_iter-1) = alpha*((X(i-1,3:x_iter) + (delta_t/2)*K(2,3:x_iter)) - 2*(X↙
(i-1,2:x_iter-1) + (delta_t/2)*K(2,2:x_iter-1)) + (X(i-1,1:x_iter-2) + (delta_t/2)*K↙
(2,1:x_iter-2)))/delta_x^2;
        K(4,2:x_iter-1) = alpha*((X(i-1,3:x_iter) + (delta_t)*K(3,3:x_iter)) - 2*(X↙
(i-1,2:x_iter-1) + (delta_t)*K(3,2:x_iter-1)) + (X(i-1,1:x_iter-2) + (delta_t)*K(3,1:↙
x_iter-2)))/delta_x^2;

        X(i,:) = X(i-1,:) + (delta_t/6)*(K(1,:) + K(2,:) + K(3,:) + K(4,:));

    end

    X_analytical = zeros(t_iter, x_iter);

    f = @(x,t) exp(-2*pi^2*t)*sin(pi*x) + exp(-8*pi^2*t)*sin(2*pi*x);

    for i = 1:length(t_vals)
        for j = 2:length(x_vals) - 1
```

```
            X_analytical(i,j) = f(x_vals(j),t_vals(i));
        end
    end

    U0 = x_0(x_vals);
    tspan = t_vals;

    % Solve using ode45
    [t_sol, U_sol] = ode45(@(t,U) heatODE(t, U, alpha, delta_x, x_iter), tspan, U0);

    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 RK4 vs. Analytical, Type 2 ↙
for RK4 vs ode45, Type 3 for all three.", "s");
    switch decide
        case "1"
            animation_speed = 0.05;
            figure;
            hold on
            h = plot(x_vals, X(1,:), 'LineWidth', 2);
            g = plot(x_vals, X_analytical(1,:), 'LineWidth', 2);
            xlabel('Position along the rod, x');
            ylabel('Temperature');
            title('Heat Equation Animation');
            legend('Estimation', 'Analytical', 'Location', 'best')
            grid on;

            % Fix y-axis limits to avoid recalculating each frame
            ylim([min(X(:)), max(X(:))]);

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

            % Efficiently animate without redrawing the full figure
            for timestep = 1:size(X,1)
                h.YData = X(timestep,:);  % only updating data, very efficient
                g.YData = X_analytical(timestep,:);
                title(sprintf('Temperature Distribution at Time Step: %d',↙
timestep));
                drawnow limitrate;        % significantly improves performance
                pause(animation_speed);   % adjust animation speed
            end
        case "2"
            animation_speed = 0;

            figure;
```

```
            hold on
            g = plot(x_vals, X(1,:), 'LineWidth', 2);
            h = plot(x_vals, U_sol(1,:), 'LineWidth', 2);
            ylim([min(U_sol(:)) max(U_sol(:))]);
            grid on;
            xlabel('Position along the rod, x');
            ylabel('Temperature');
            legend('RK4', 'ode45', 'Location', 'best')
            for k = 1:length(t_sol)
                set(h, 'YData', U_sol(k,:));
                set(g, 'YData', X(k,:));
                title(sprintf('Time = %.2f s', t_sol(k)));
                drawnow limitrate;
                pause(animation_speed);
            end
        case "3"
            animation_speed = 0;

            figure;
            hold on
            g = plot(x_vals, X(1,:), 'LineWidth', 2);
            h = plot(x_vals, U_sol(1,:), 'LineWidth', 2);
            i = plot(x_vals, X_analytical(1,:), 'LineWidth', 2);
            ylim([min(U_sol(:)) max(U_sol(:))]);
            grid on;
            xlabel('Position along the rod, x');
            ylabel('Temperature');
            legend('RK4', 'ode45', 'Analytical', 'Location', 'best')
            for k = 1:length(t_sol)
                set(h, 'YData', U_sol(k,:));
                set(g, 'YData', X(k,:));
                set(i, 'YData', X_analytical(k,:));
                title(sprintf('Time = %.2f s', t_sol(k)));
                drawnow limitrate;
                pause(animation_speed);
            end
        case isempty(decide)
            return
    end
end

function dUdt = heatODE(t, U, alpha, delta_x, x_iter)
    dUdt = zeros(x_iter,1);

    % Boundary conditions (example: Dirichlet - fixed endpoints)
    U(1) = 0;         % u(0,t) boundary (fixed)
    U(x_iter) = 0;         % u(L,t) boundary (fixed)

    % Interior points (second-order finite difference)
```

```
    for i = 2:x_iter-1
        dUdt(i) = alpha*(U(i+1)-2*U(i)+U(i-1))/delta_x^2;
    end
end
```