```
function [t_vals, x_vals, X, delta_x, delta_t] = general_wave(t_iter, x_iter, c)

    t_0 = 0;
    t_f = 120;
    t_vals = linspace(t_0, t_f, t_iter);

    L = 2;
    X = zeros(t_iter, x_iter);
    x_0 = @(x) sin(pi*x/2);
    x_vals = linspace(0,L,x_iter);
    X(1, :) = x_0(x_vals);
    BC1 = @(t) 0.01*sin(t);
    X(:,1) = BC1(t_vals);
    X(1,end) = X(1,end-1);

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

    if delta_t > delta_x/c
        fprintf('Upper Bound: %g\n', delta_x/c)
        fprintf('Delta t: %g\n', delta_t)
        disp("Unstable, increase t_iter or decrease x_iter")
        return
    end

    ghost_step = X(1,:) - delta_t*cos(pi*x_vals/2);
    i = 2;
    X(2,2:x_iter-1) = (c*delta_t/delta_x)^2*(X(i-1,3:x_iter) - 2*X(i-1,2:x_iter-1) + X(i-1,1:x_iter-2)) + 0.02*delta_t^2*sin(x_vals(2:x_iter-1)+t_vals(i-1)) + 2*X(i-1,2:x_iter-1) - ghost_step(2:x_iter-1);

    for i = 3:t_iter
        X(i,2:x_iter-1) = (c*delta_t/delta_x)^2*(X(i-1,3:x_iter) - 2*X(i-1,2:x_iter-1) + X(i-1,1:x_iter-2)) + 0.02*delta_t^2*sin(x_vals(2:x_iter-1)+t_vals(i-1)) + 2*X(i-1,2:x_iter-1) - X(i-2, 2:x_iter-1);
        X(i,end) = X(i,end-1);
    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;
            figure;
```

```
            hold on
            h = plot(x_vals, X(1,:), 'LineWidth', 2);
            xlabel('Position along the rod, x');
            ylabel('Amplitude');
            title('Wave Equation Animation');
            legend('Estimation', 'Location', 'best')
            grid on;

            % Fix y-axis limits to avoid recalculating each frame
            ylim([min(X(:)), max(X(:))]);
            xlim([0 2]);

            % 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
                title(sprintf('Wave at Time Step: %d', timestep));
                drawnow limitrate;        % significantly improves performance
                pause(animation_speed);   % adjust animation speed
            end
        case isempty(decide)
            return
    end

end
```