# Computer project:

***(a) (25 pts) Fit the data in a least squares sense with polynomials of degrees 1 through 5. Compare the fitted polynomial with erf(t) for values of t between the data points. How does the maximum error depend on the polynomial degree? Make a chart of maximum errors versus n. (Question to ponder: How would you compute the maximum error?).***

Using least square fitting on polynomial order function the following coefficients are estimated.

***Order 1:*** *C1: 1.79e-17, C2: 0.14*

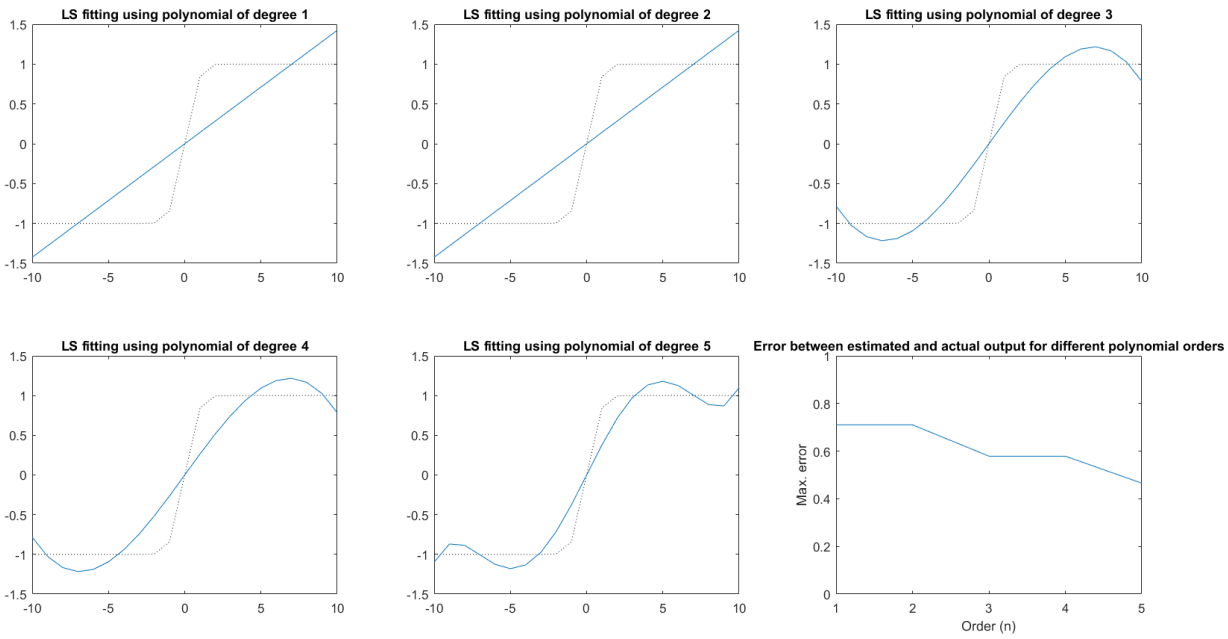
***Order 2:*** *C1: 5.26e-17, C2: 0.14, C3: -1.74e-18*

***Order 3:*** *C1: 8.51e-17, C2: 0.26, C3: -3.66e-18, C4: -1.8e-3*

***Order 4:*** *C1: 3.00e-18, C2: 0.26, C3: 3.00e-18, C4: -1.8e-3, C5: -4.37e-20*

***Order 5:*** *C1: -1.80e-17, C2: 0.38, C3: 6.54e-18, C4:-6.98e-3, C5: -9.41e-20, C6: 4.24e-05*

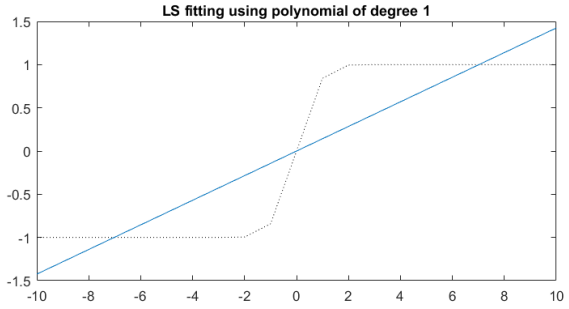
Their response of the polynomial equation when we plug in these coefficients are shown below:

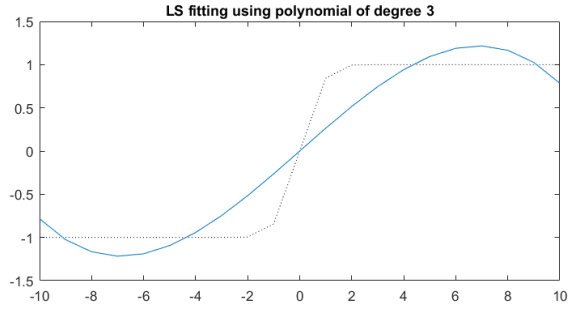


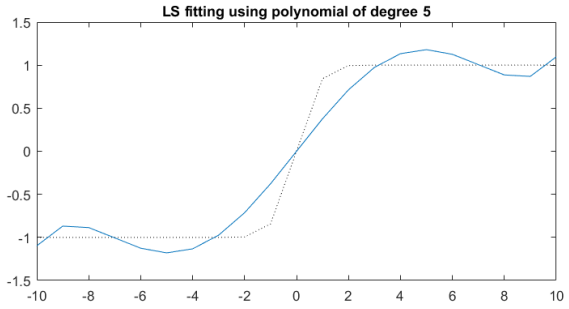

In the bottom right graph, it can be seen that the maximum error is gradually decreasing as n is increased.

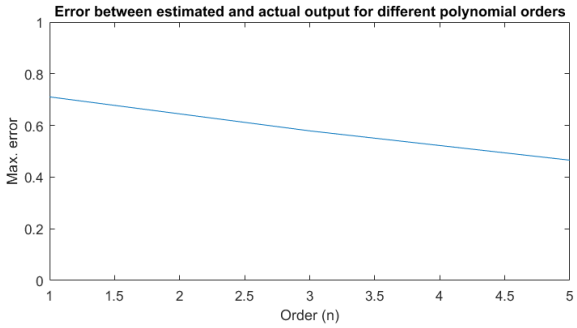
***(b) (15 pts) Because erf(t) is an odd function of t, that is, erf(x) = −erf(−x), it is reasonable to fit the data by a linear combination of odd powers of t. Try degrees 1, 3, and 5. Again, study how the error between data points depends on n. (Question to ponder: Would this extra care of exploiting oddness help to improve the fitting?)***

As can be seen in the above graph the max. error was not dropping linearly but in steps. Therefore, it will be a nice idea to try it for odd values of n. With this condition the plot is shown below:



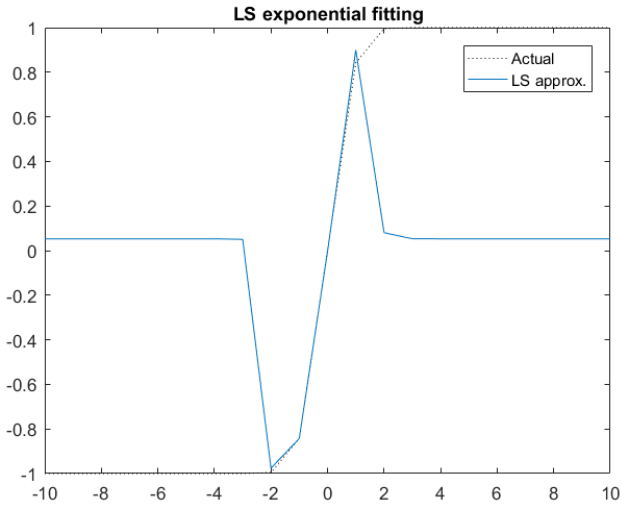




Now it can be seen that max. error linearly decreasing as odd values of n are increased.

***(c) (20 pts) Polynomials are not particularly good approximants for erf(t) because they are unbounded for large t, whereas erf(t) approaches 1 for large t. Try a model of the form c(t) = c1 + e −t 2 (c2 + c3z + c4z 2 + c5z 3 ), where z = 1/(1 + t). (How does the error between the data points compare with the polynomial models?)***

Now by considering this type of model, the least square approximation resulted in the plot shown below:




It can be seen that the following model fits well for values of time between -2 to 2 seconds more than the maximum order polynomial. Therefore, max. error is expected to be much less within this region for values within the range of -2 to 2.

***Appendix: Code***

Code for part 1:

```
% Generating 21 random points
% tk = 20*rand(21,1) - 10;
tk = [-10:10]';
yk = erf(tk);

% Fitting data using polynomial least square
for deg = 1:5

    T{deg} = [];

    for i = 0:deg
        T{deg} = [T{deg} tk.^i];
    end;

    coeff{deg} = T{deg}\yk;

end;

t = [-10:10]';
y = erf(t);

% Getting output given the estimated coefficients
% and comparing it with actual output
figure;
for deg = 1:5

    T = [];

    for i = 0:deg
        T = [T t.^i];
    end;

    yest(:,deg) = T*coeff{deg};

    subplot(2,3,deg);
    plot(t,y,'k:');
    hold on; plot(t,yest(:,deg));
    title(['LS fitting using polynomial of degree ',num2str(deg)]);

end;

% Getting max error in the estimated and actual outputs
maxerr = max(repmat(y,[1,5]) - yest);

subplot(2,3,6);
plot(1:5,maxerr);
title('Error between estimated and actual output for different polynomial orders');
xlabel('Order (n)');
ylabel('Max. error');
ylim([0 1]);
```

Code for part 2:

```
clear('T','yest')

% Fitting data using polynomial least square
D = [1,3,5];
for deg = 1:length(D)

    T{deg} = [];

    for i = 0:D(deg)
        T{deg} = [T{deg} tk.^i];
    end;

    coeff{deg} = T{deg}\yk;

end;

t = [-10:10]';
y = erf(t);

% Getting output given the estimated coefficients
% and comparing it with actual output
figure;
for deg = 1:length(D)

    T = [];

    for i = 0:D(deg)
        T = [T t.^i];
    end;

    yest(:,deg) = T*coeff{deg};

    subplot(2,2,deg);
    plot(t,y,'k:');
    hold on; plot(t,yest(:,deg));
    title(['LS fitting using polynomial of degree ',num2str(D(deg))]);

end;

% Getting max error in the estimated and actual outputs
maxerr = max(repmat(y,[1,3]) - yest);

subplot(2,2,4);
plot(D,maxerr);
title('Error between estimated and actual output for different polynomial orders');
xlabel('Order (n)');
ylabel('Max. error');
ylim([0 1]);
```

Code for part 3:

```
clear('T','yest');

tk = [-10:10]';
yk = erf(tk);

% Fitting data using polynomial least square
T = ones(length(tk),1);
for i = 0:3
    T = [T exp(-tk.^2).*(1./(1.1+tk)).^i];
end;

coeffexp = T\yk;

t = [-10:10]';
y = erf(t);

% Getting output given the estimated coefficients
% and comparing it with actual output
T = ones(length(t),1);
for i = 0:3
    T = [T exp(-t.^2).*(1./(1.1+t)).^i];
end;
yestexp = T*coeffexp;

figure;plot(t,y,'k-');
hold on; plot(t,yestexp);
title('LS exponential fitting');
legend('Actual','LS approx.');
```