Table of Contents


```
%{
EP501 HW3

RUN THIS SCRIPT

DEPENDENCIES:
Gauss_elim.m
backsub.m
%}

clc
clear
close all
```

Load variables into workspace if they are not already there

```
if ~exist('sigmay','var') || ~exist('x','var') ||
 ~exist('ynoisy','var')
    load test_lsq.mat
end % if - exist

if ~exist('xg','var') || ~exist('xgi','var') || ~exist('yg','var') ...
        || ~exist('ygi','var') || ~exist('f2D','var')
    load test_interp.mat
end % if - exist
```

1a. Fit an arbitrary polynomial to data using linear least squares

see linlsqr.m

```
fprintf('===== Problem 1
 ============================================\n')
```

```
===== Problem 1 ==============================================
```

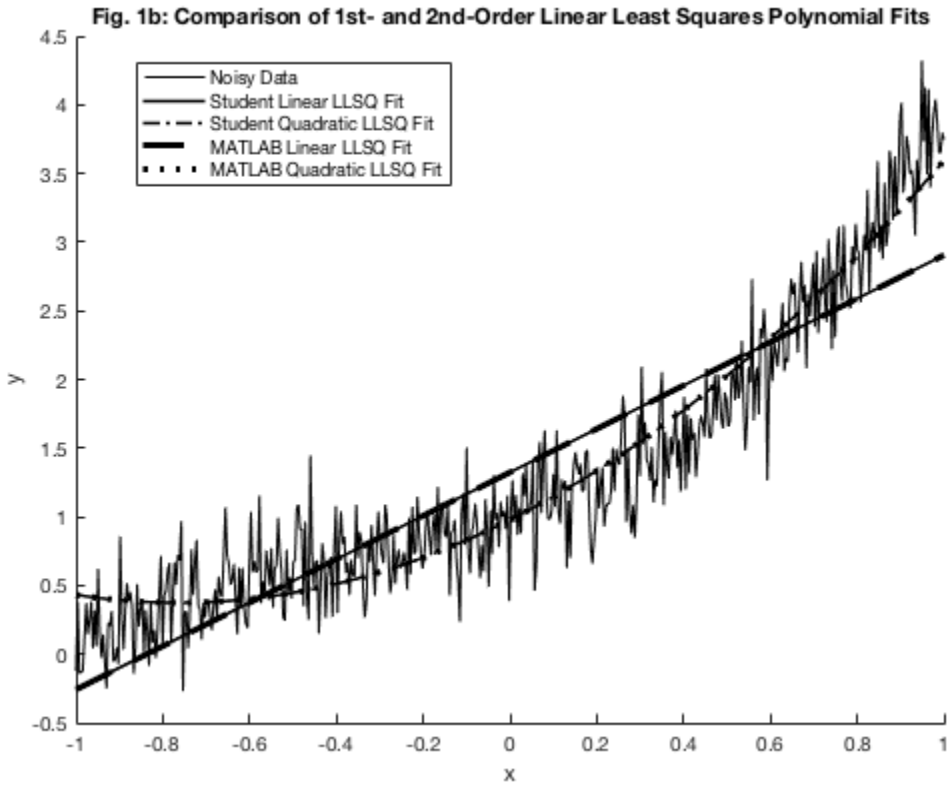
1b. Compare linear and quadratic fits, compare results with built-ins

generate coefficients of fitted polynomial

```
polycoeff1 = linlsqr(1,x,ynoisy); % line [a bx]
polycoeff2 = linlsqr(2,x,ynoisy); % quadratic [a bx cx^2]

% initialize polynomial value arrays
poly1 = zeros(length(x),1);
poly2 = zeros(length(x),1);

% evaluate polynomial functions from given x values and coefficient
 vectors
for i = 1:length(x)
    % hardcoded for-loop for a linear and quadratic fit; automating
 the
    % loop flow for arbitrary order polynomials seems like too much
 work.
    % set j = length of highest-order polynomial coefficient vector
    for j = 1:length(polycoeff2)
        poly2(i) = poly2(i) + polycoeff2(j)*x(i)^(j-1);
        if j > length(polycoeff1)
            break
        else
            poly1(i) = poly1(i) + polycoeff1(j)*x(i)^(j-1);
        end % if - j
    end % for - j
end % for - i

% Matlab built-in function results
Mlinfit = polyfit(x,ynoisy,1);
Mquadfit = polyfit(x,ynoisy,2);

Mlinpoly = polyval(Mlinfit,x);
Mquadpoly = polyval(Mquadfit,x);

figure
hold on
plot(x,ynoisy,'k')
plot(x,poly1,'-k','LineWidth',1.5)
plot(x,poly2,'-.k','LineWidth',1.5)
plot(x,Mlinpoly,'--k','LineWidth',3)
plot(x,Mquadpoly,':k','LineWidth',3)
hold off

legend('Noisy Data','Student Linear LLSQ Fit',...
    'Student Quadratic LLSQ Fit','MATLAB Linear LLSQ Fit',...
    'MATLAB Quadratic LLSQ Fit','Location','Best')
```

```
fprintf('As seen in Fig. 1b, the built-in MATLAB results are identical
\nto my program''s output. ')
fprintf('The quadratic polynomial is clearly a\nbetter fit for the
 data.\n')
xlabel('x')
ylabel('y')
title('Fig. 1b: Comparison of 1st- and 2nd-Order Linear Least Squares
 Polynomial Fits')
```

As seen in Fig. 1b, the built-in MATLAB results are identical
to my program's output. The quadratic polynomial is clearly a
better fit for the data.



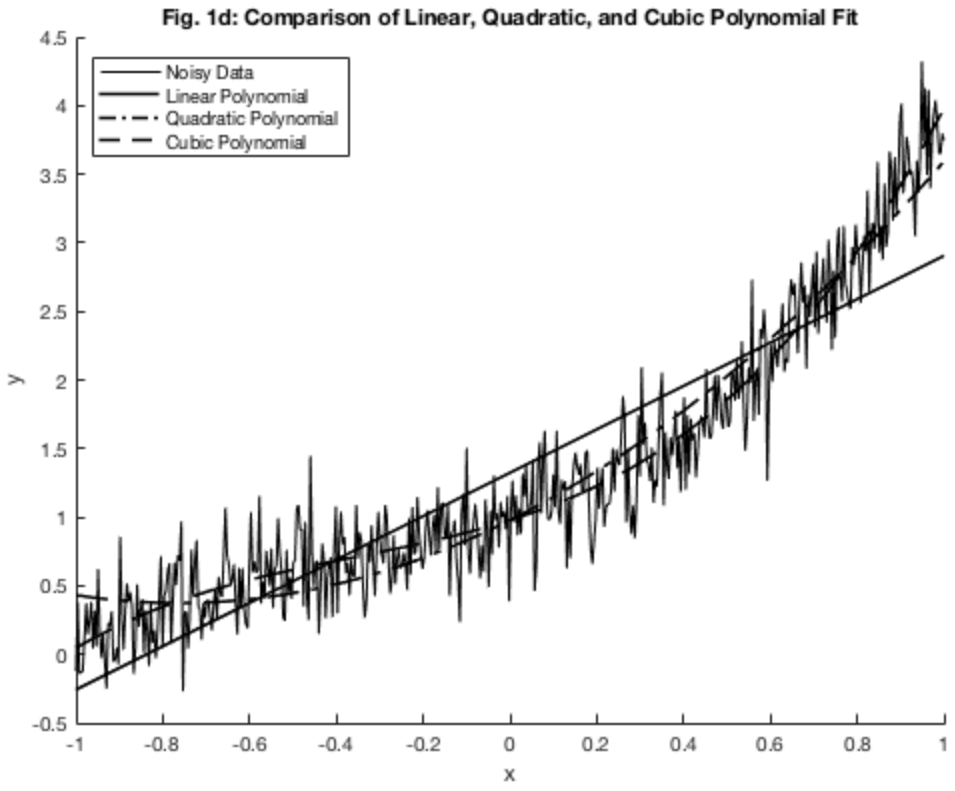
1c. Determine "goodness of fit" using Reduced Chi-Squared Statistic

generate cubic fit

```
polycoeff3 = linlsqr(3,x,ynoisy); % cubic [a bx cx^2 dx^3]
poly3 = zeros(length(x),1);

% evaluate polynomial
for i = 1:length(x)
    for j = 1:length(polycoeff3)
        poly3(i) = poly3(i) + polycoeff3(j)*x(i)^(j-1);
```

```
    end % for - j
end % for - i

figure
hold on
plot(x,ynoisy,'k')
plot(x,poly1,'-k','LineWidth',1.5)
plot(x,poly2,'-.k','LineWidth',1.5)
plot(x,poly3,'--k','LineWidth',1.5)
hold off

title('Fig. 1d: Comparison of Linear, Quadratic, and Cubic Polynomial
 Fit')
xlabel('x')
ylabel('y')
legend('Noisy Data','Linear Polynomial','Quadratic Polynomial','Cubic
 Polynomial',...
    'Location','Best')

% calculate chi^2 values for each order of polynomial
chisqrd1 = rcss(1,ynoisy,sigmay,poly1);
chisqrd2 = rcss(2,ynoisy,sigmay,poly2);
chisqrd3 = rcss(3,ynoisy,sigmay,poly3);

fprintf('\nGoodness-of-fit metric Chi^2\n')
fprintf('--------------------------------\n')
fprintf('Linear polynomial: %.4f\n',chisqrd1)
fprintf('Quadratic polynomial: %.4f\n',chisqrd2)
fprintf('Cubic polynomial: %.4f\n',chisqrd3)
```

```
Goodness-of-fit metric Chi^2
--------------------------------
Linear polynomial: 2.9125
Quadratic polynomial: 1.3754
Cubic polynomial: 1.0285
```



1d. Determine which order of polynomial best fits the data

From the Wikipedia page on reduced chi-squared statistic, it seems that a value of RCSS greater than 1 implies that the estimated function (the fitted polynomial) does not agree entirely with the data. A value of 1 implies that the discrepancy between the measured values and the estimated function is about the same as the estimated error in the measured values. Thus, whichever order polynomial's RCSS value is closest to 1 is the best fit for the given data.

```
chi = [chisqrd1,chisqrd2,chisqrd3];
bestfit = min(chi);
fprintf('\nA polynomial of order %d is the best fit for these data,
 with\na Chi^2 value of %.4f.\n',find(chi==bestfit),bestfit)
```

```
A polynomial of order 3 is the best fit for these data, with
a Chi^2 value of 1.0285.
```

2a. Write program to find indices to interpolate from, 1D

```
fprintf('\n===== Problem 2
 ==============================================\n')
```

```
% see intrpindx1.m
```

```
===== Problem 2 ============================================
```

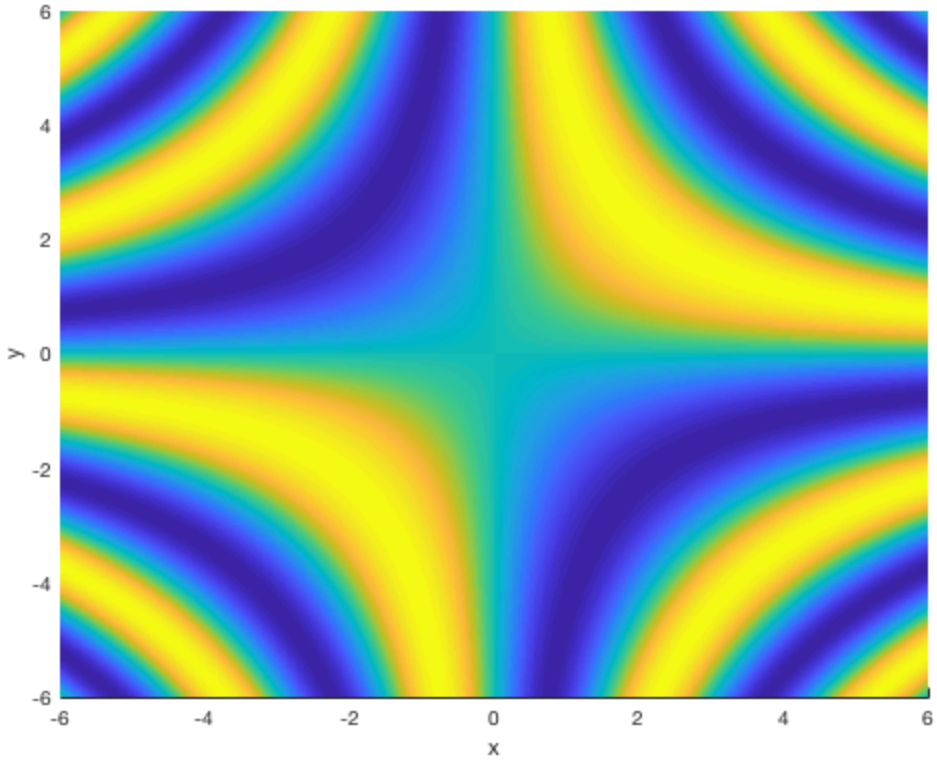
2b. Write program to find indices to interpolate from, 2D

```
% see intrpindx2.m
```

2c. Write program to accept list of points to interpolate at, 2D

```
f2Dintrp = intrp2d(xg,yg,xgi,ygi,f2D);
MATLABintrp = interp2(xg,yg,f2D,xgi,ygi');

[XGI,YGI] = meshgrid(xgi,ygi);
figure
hold on
surf(XGI,YGI,f2Dintrp,'LineStyle','none')
xlabel('x')
ylabel('y')
zlabel('Interpolated function')
hold off

% difference between student results and MATLAB 'interp2' results
diffsum = sum(sum(f2Dintrp-MATLABintrp));

format short

fprintf('Subtracting the MATLAB interpolation matrix from the student
\n')
fprintf('interpolation matrix and summing the differences gives %.4e,
\n',diffsum)
fprintf('which implies that the two are identical to machine accuracy.
\n')
```

```
Subtracting the MATLAB interpolation matrix from the student
interpolation matrix and summing the differences gives 6.6527e-12,
which implies that the two are identical to machine accuracy.
```



Published with MATLAB® R2018b