Linear Systems Assignment (By Patrice Harapeti)

Table of Contents

Background

Linear Regression and Fast Fourier Transform and Frequency Analysis. Not going to lie... this assignment was tough because it tied together all of the concepts we have learnt over the last 4-5 weeks.

Setup

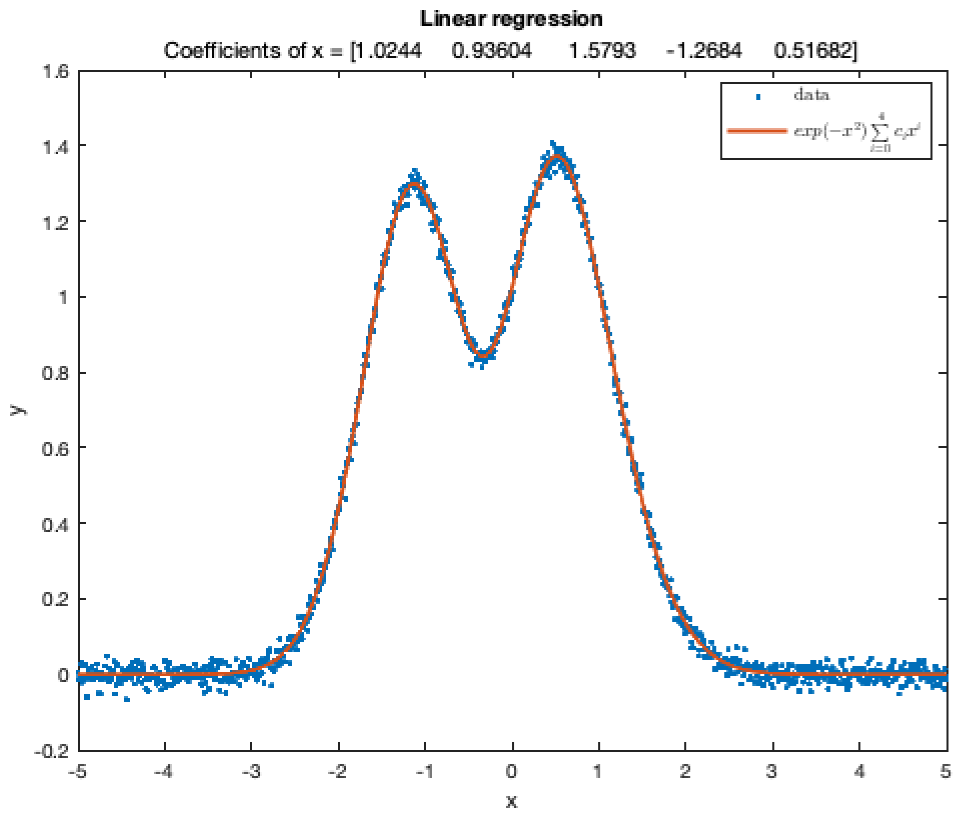
```
clc; clear; close all;
```

Part One : Linear Regression

```
% Given dataset
x = linspace(-1,1,1e3).' * 5;
y = exp(-x.^2) .* polyval(randn([1,5]),x) + randn(size(x)) * 0.02;

% Build model
A = exp(-x.^2) .* [x.^0 x.^1 x.^2 x.^3 x.^4];

% Solve the system for the coefficients for each power of x in the
 model
c = A \ y;

% Calculate your fitted function here using c and the model
yFit = A * c;

% Sum squared error
S = sum((y - yFit) .^ 2);

% Mean squared error
% ... m = number of data points = length(x)
% ... n = number of coefficients = length(c)
% ... therefore m - n
m = length(x);
n = length(c);
meanSquaredError = sqrt(S / (m - n));
fprintf('\nPart One:\n');
fprintf('Mean Squared Error = %f\n', meanSquaredError);
```

```
% Generate Covariance Matrix from diaglon of inverse conjugate matrix
 of A
covarianceMatrix = diag(inv(A.' * A));

% Uncertainty Matrix
uncertainty = meanSquaredError .* sqrt(covarianceMatrix);
disp("Uncertainty for each coefficient of x = [" + ...
    num2str(uncertainty.') + ']');

figure(1);
plot(x, y, '.', 'LineWidth', 1);
hold on;
plot(x, yFit, '-', 'LineWidth', 2);
title('Linear regression');
xlabel('x');
ylabel('y');
subtitle("Coefficients of x = [" + num2str(c.') + ']');
legend({'data', '$exp(-x^2)\sum\limits_{i=0}^4c_{i}x^{i}$'}, ...
    'Interpreter', 'Latex');
```

```
Part One:
Mean Squared Error = 0.020178
Uncertainty for each coefficient of x = [0.0024692    0.0057024
 0.010201    0.0058894    0.0058894]
```

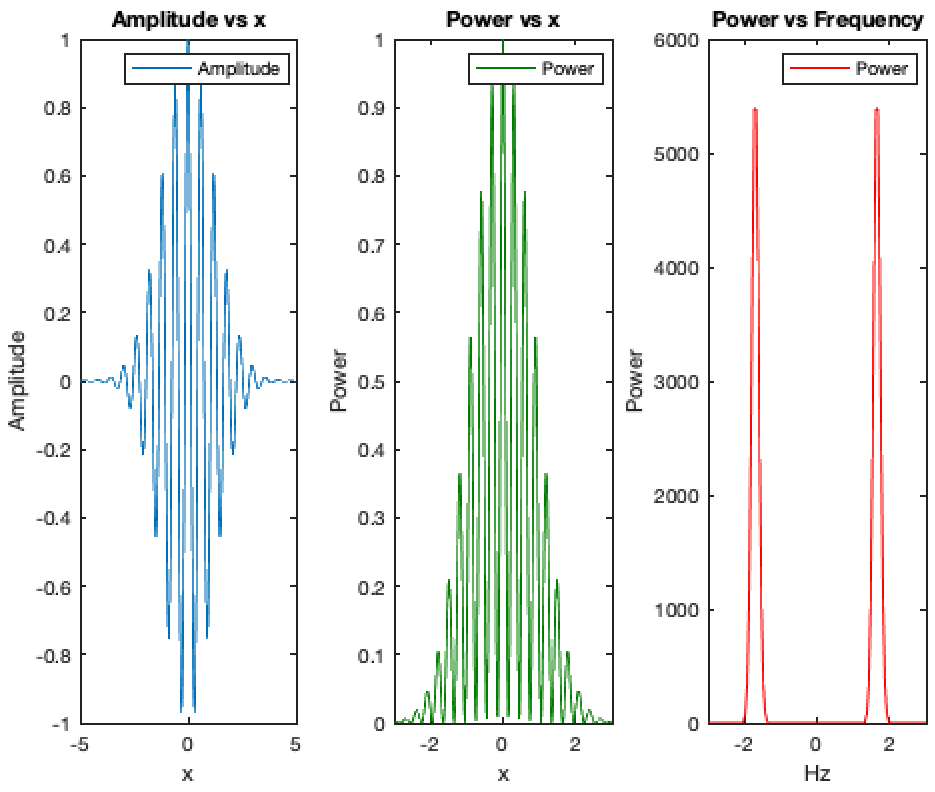


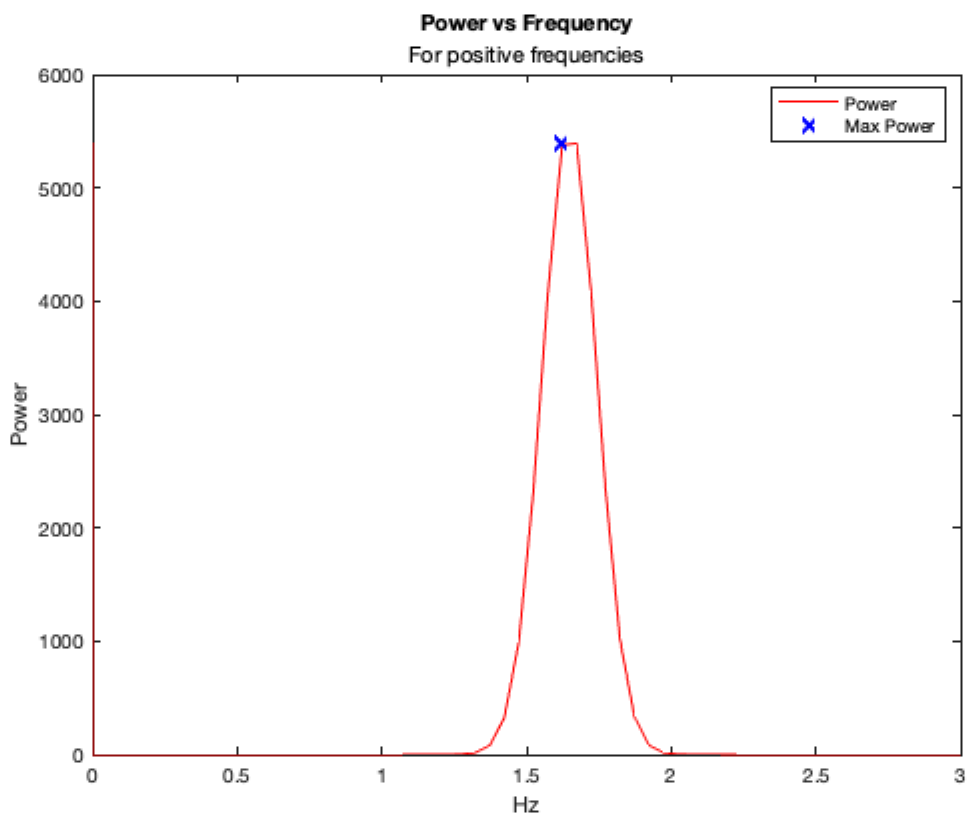
Part Two : Fast Fourier Transform

```
% Delete temporarily variables of Part 2 (memory optimisation)
clear;

% Specify given dataset
x = linspace(-1,1,1+1e3).' * 10;
y = exp(-(x/(1+rand(1))).^2) .* cos(2*pi*(1+rand(1))*x);
power_real = abs(y).^2;

% Plot amplitude vs x
figure(2);
subplot(1, 3, 1);
plot(x, y);
title('Amplitude vs x');
xlim([-5, 5]);
xlabel('x');
ylabel('Amplitude');
legend('Amplitude');

% Plot real power vs x
figure(2);
subplot(1, 3, 2);
plot(x, power_real, 'Color','#008000');
title('Power vs x');
xlim([-3, 3]);
xlabel('x');
ylabel('Power');
legend('Power');

% Calculate parameters for transformation matrix
N = length(y); % number of samples
sampleSpacing = mean(diff(x)); % sample spacing
frequencySpacing = 1/(N * sampleSpacing); % frequency spacing
samplingFrequency = N * frequencySpacing; % sampling frequency

% Calculate frequency vector
frequency = (-N/2:(N/2)-1) * frequencySpacing;

% Compute forward transformation matrix
M = exp(-2 * 1i * pi * frequency .* x);

% Apply transformation matrix onto y to convert from [amplitude, x]
 domain
% to[amplitude, frequency] domain
Y = M * y;

% Calculate power spectrum of signal
power_freq = abs(Y) .^ 2;

% Plot Power vs Frequency
subplot(1, 3, 3);
plot(frequency, power_freq, '-r');
```

```
title('Power vs Frequency');
xlim([-3, 3]);
xlabel('Hz');
ylabel('Power');
legend('Power');

% Limit power (in frequency domain) to be positive for frequency
 analysis
frequencyPositive = frequency .* (frequency > 0);
powerPositive = power_freq .* (power_freq > 0);

% Plot power (frequency domain) vs frequency for positive frequencies
figure(3);
plot(frequencyPositive, powerPositive, '-r');
title('Power vs Frequency');
subtitle('For positive frequencies');
xlim([0, 3]);
xlabel('Hz');
ylabel('Power');
hold on;

% Estimate center x in real domain
powerSum_real = sum(power_real);
weightedPowerSum_real = sum(power_real .* x);
expectedCenterX_real = weightedPowerSum_real ./ powerSum_real;

% Calculate full-width half maximum in real domain
FWHM_real = expectedCenterX_real .^2 / sqrt(log(2));

% Estimate center frequency in frequency domain
powerSum_freq = sum(powerPositive);
weightedPowerSum_freq = sum(powerPositive .* frequencyPositive.');
expectedCenterFrequency_freq = weightedPowerSum_freq ./ powerSum_freq;

% Calculate full-width half maximum in frequency domain
FWHM_freq = expectedCenterFrequency_freq .^2 / sqrt(log(2));

% Print out expectation values and FWHM values in each domain
fprintf('\nPart Two:\n');
disp(['Expected center x in real domain: ', ...
    num2str(expectedCenterX_real)]);
disp(['Expected center frequency in frequency domain: ', ...
    num2str(expectedCenterFrequency_freq)]);
disp(['FWHM in real domain: ', num2str(FWHM_real)]);
disp(['FWHM in frequency domain: ', num2str(FWHM_freq)]);

% Find maxima of positive power vs frequency
[powerMaxima, maximaIndex] = max(powerPositive);
frequencyMaxima = frequencyPositive(N - maximaIndex);

% Mark maxima on plot of positive power vs frequency
plot(frequencyMaxima, powerMaxima, 'xb', 'LineWidth', 2);
legend('Power', 'Max Power');
```

```
Part Two:
Expected center x in real domain: -1.4154e-17
Expected center frequency in frequency domain: 0.8244
FWHM in real domain: 2.4062e-34
FWHM in frequency domain: 0.81633
```





Published with MATLAB® R2020b