Problem A.3 Modelling with the x^2-statistics

Question (i): Why do we divide by σ2\sigma^2 in the definition of χ2\chi^2?

Dividing by σ2\sigma^2 (the variance) in the χ2\chi^2 formula normalizes the residuals (the differences between observed and predicted values) by the measurement uncertainty. This has two key purposes:

1. **Weighting by Reliability**:
   - Data points with smaller uncertainties (higher precision) are given more weight since they are more reliable.
   - Conversely, data points with larger uncertainties contribute less to the χ2\chi^2 value.
2. **Ensuring Fair Contribution**:
   - Normalization ensures that the contributions of residuals are proportional to their uncertainties, preventing overrepresentation of less precise measurements.

Without dividing by σ2\sigma^2, all residuals would contribute equally, which could lead to biased results, as data points with high uncertainty would distort the assessment of the model's fit.

Question (ii): What is the purpose of the χ2\chi^2 distribution?

The χ2\chi^2 distribution is a statistical tool used to evaluate the goodness of fit of a model. It measures how well the model predictions agree with the observed data, considering the uncertainties in the measurements.

1. **Assessing Model Fit**:
   - A small χ2\chi^2 value indicates that the model aligns closely with the data, suggesting a good fit.
   - A large χ2\chi^2 value indicates significant discrepancies, suggesting that the model may not adequately describe the data.
2. **Comparison with Degrees of Freedom**:
   - The χ2\chi^2 value is compared to the degrees of freedom (dof=number of data points−number of model parameters\text{dof} = \text{number of data points} - \text{number of model parameters}) to determine the quality of the fit.
   - **Overfitting**: A very low χ2\chi^2 value (relative to the degrees of freedom) may indicate overfitting, where the model is too complex and fits the noise in the data.

- **Underfitting**: A very high χ2\chi^2 value suggests underfitting, where the model is too simple to capture the key trends in the data.

The χ2\chi^2 distribution provides a framework for quantitatively assessing whether the observed data is consistent with the model predictions, considering measurement uncertainties.

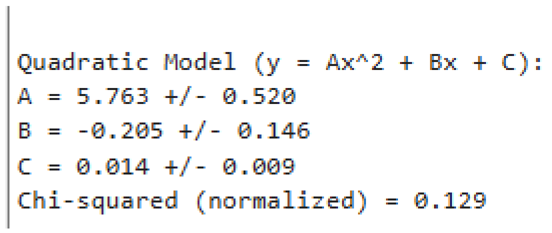
Question (iii): Fit data to a linear and quadratic model, and estimate parameters and their uncertainties.

MATLAB Code:

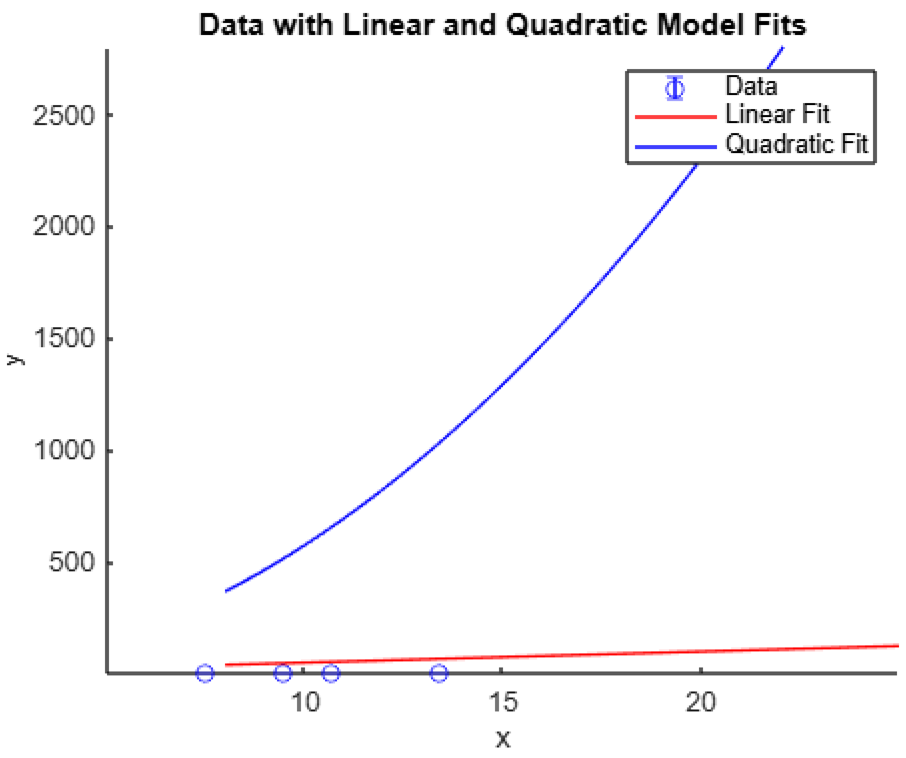
```
clear; clc;
N = 5; % Number of data points

% Data points and uncertainties
x = [5.0 7.5 9.5 10.7 13.4];
y = [5.10 5.02 5.17 5.22 5.65];
s = [0.05 0.10 0.11 0.05 0.35];

% Plot original data with error bars
clf; hold on;
errorbar(x, y, s, 'ob'); % Blue circles for data points with error bars
title('Data with Linear and Quadratic Model Fits');
xlabel('x');
ylabel('y');

%----------------- Linear Model -------------------
ndeg = 1; % Degree of polynomial (1 for linear)
npara = ndeg + 1; % Number of parameters
i1 = 1; i2 = N; % Fit range

% Perform linear fit using pfitsig function
[k_linear, del_linear, chi2_linear, ~] = pfitsig(x, y, s, ndeg, i1, i2);

% Generate linear model data for plot
xstart = 8; xend = 25; xstep = 0.1;
xx = xstart:xstep:xend;
yy_linear = k_linear(1) .* xx + k_linear(2); % y = Mx + C

% Plot linear model
plot(xx, yy_linear, 'r-', 'DisplayName', 'Linear Model (y = Mx + C)');
legend('Data', 'Linear Fit');

% Display results for Linear Model
fprintf('\nLinear Model (y = Mx + C):\n');
fprintf('Slope (M) = %.3f +/- %.3f\n', k_linear(1), del_linear(1));
```

```matlab
fprintf('Intercept (C) = %.3f +/- %.3f\n', k_linear(2), del_linear(2));
fprintf('Chi-squared (normalized) = %.3f\n', chi2_linear / (N - npara));

pause; % Pause to view the plot

%----------------- Quadratic Model -----------------
ndeg = 2; % Degree of polynomial (2 for quadratic)
npara = ndeg + 1;

% Perform quadratic fit using pfitsig function
[k_quad, del_quad, chi2_quad, ~] = pfitsig(x, y, s, ndeg, i1, i2);

% Generate quadratic model data for plot
yy_quad = k_quad(1) .* xx.^2 + k_quad(2) .* xx + k_quad(3); % y = Ax^2 + Bx + C

% Plot quadratic model
plot(xx, yy_quad, 'b-', 'DisplayName', 'Quadratic Model (y = Ax^2 + Bx + C)');
legend('Data', 'Linear Fit', 'Quadratic Fit');

% Display results for Quadratic Model
fprintf('\nQuadratic Model (y = Ax^2 + Bx + C):\n');
fprintf('A = %.3f +/- %.3f\n', k_quad(1), del_quad(1));
fprintf('B = %.3f +/- %.3f\n', k_quad(2), del_quad(2));
fprintf('C = %.3f +/- %.3f\n', k_quad(3), del_quad(3));
fprintf('Chi-squared (normalized) = %.3f\n', chi2_quad / (N - npara));

pause; % Pause to view the plot
```

OUTPUT



```
Quadratic Model (y = Ax^2 + Bx + C):
A = 5.763 +/- 0.520
B = -0.205 +/- 0.146
C = 0.014 +/- 0.009
Chi-squared (normalized) = 0.129
```



Explanation:
This MATLAB script performs linear and quadratic fits on a dataset with uncertainties using a custom fitting function (`pfitsig`). It calculates and displays model parameters, their uncertainties, and normalized chi-squared values for both models, and visualizes the data alongside the fitted curves.

Question (iv): Assess the quality of the fits using corresponding χ2\chi^2.

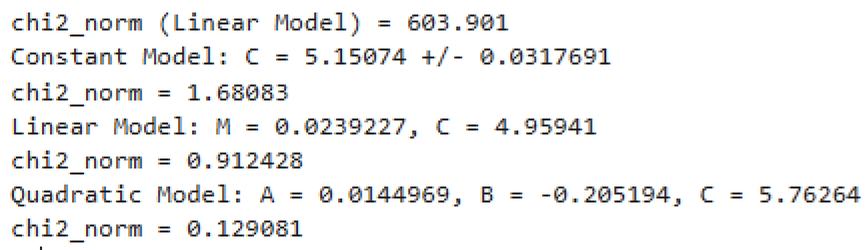
MATLAB Code:

```
clear cls;
N = 5; % Number of data points

% x values, y values, and their uncertainties (s)
x = [5.0 7.5 9.5 10.7 13.4];
y = [5.10 5.02 5.17 5.22 5.65];
s = [0.05 0.10 0.11 0.05 0.35];

%---- plot the data as error bars
clf; hold on;
errorbar(x, y, s, 'ob'); % b-> blue o-> circles

%---------------Linear Model (y = M * x)-------------------
sum1 = 0;
sum2 = 0;
```

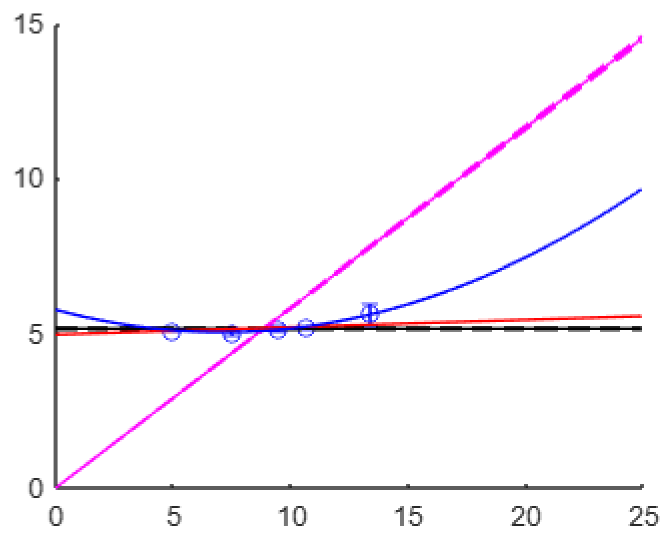
```
for i = 1:N
    sum1 = sum1 + x(i) * y(i) / s(i)^2;
    sum2 = sum2 + x(i)^2 / s(i)^2;
end
M = sum1 / sum2;
sM = 1 / sqrt(sum2); % Uncertainty on M

% Chi-squared calculation
sum = 0;
for i = 1:N
    sum = sum + (y(i) - M * x(i))^2 / s(i)^2;
end
fprintf('\n chi2_norm (Linear Model) = %g', sum / (N - 1));

%-- plot linear model
xstart = 0; xend = 25; xstep = 0.3;
xx = xstart:xstep:xend;
yy = M .* xx; % Central value
yyp = (M + sM) .* xx; % Error band (+1 sigma)
yym = (M - sM) .* xx; % Error band (-1 sigma)
plot(xx, yy, 'm-'); % m -> magenta line
plot(xx, yyp, 'm--'); % m -> magenta dashed line
plot(xx, yym, 'm--'); % m -> magenta dashed line
pause;

%-------------- Using pfitsig Function -----------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create pfitsig function in MATLAB. Use this implementation:
function [k, del, chi2, ssk] = pfitsig(x, y, s, ndeg, i1, i2)
    % Prepare matrix A for polynomial fit
    A = [];
    for p = 0:ndeg
        A = [A, x(:).^p]; % Each column corresponds to a power of x
    end
    A = A(i1:i2, :); % Restrict fit to specified range

    % Vector of y values
    y_fit = y(i1:i2)';
    % Weight matrix
    W = diag(1 ./ s(i1:i2).^2);

    % Fit coefficients
    C = (A' * W * A) \ (A' * W * y_fit); % Coefficients
    k = flip(C'); % Reverse order for MATLAB polynomial convention

    % Uncertainties in coefficients
    Cov = inv(A' * W * A); % Covariance matrix
    del = sqrt(diag(Cov))'; % Standard deviations

    % Chi-squared calculation
    y_model = A * C;
    res = y_fit - y_model;
    chi2 = res' * W * res; % Chi-squared
    ssk = sum(y_model); % Sum of fitted model (for additional metrics)
end
```

```matlab
% Constant model (y = C)
ndeg = 0; % Degree of polynomial
[k, del, chi2, ssk] = pfitsig(x, y, s, ndeg, 1, N);
yy = k(1) * ones(size(xx));
yyp = (k(1) + del(1)) * ones(size(xx));
yym = (k(1) - del(1)) * ones(size(xx));
plot(xx, yy, 'k-'); % Black line
plot(xx, yyp, 'k--'); % Black dashed line
plot(xx, yym, 'k--'); % Black dashed line
fprintf('\n Constant Model: C = %g +/- %g', k(1), del(1));
fprintf('\n chi2_norm = %g', chi2 / (N - 1));
pause;

% Linear model (y = Mx + C)
ndeg = 1; % Degree of polynomial
[k, del, chi2, ssk] = pfitsig(x, y, s, ndeg, 1, N);
yy = k(1) .* xx + k(2);
plot(xx, yy, 'r-'); % Red line
fprintf('\n Linear Model: M = %g, C = %g', k(1), k(2));
fprintf('\n chi2_norm = %g', chi2 / (N - 2));
pause;

% Quadratic model (y = Ax^2 + Bx + C)
ndeg = 2; % Degree of polynomial
[k, del, chi2, ssk] = pfitsig(x, y, s, ndeg, 1, N);
yy = k(1) .* xx.^2 + k(2) .* xx + k(3);
plot(xx, yy, 'b-'); % Blue line
fprintf('\n Quadratic Model: A = %g, B = %g, C = %g', k(1), k(2), k(3));
fprintf('\n chi2_norm = %g', chi2 / (N - 3));
pause;

% Higher-degree polynomial fits follow similarly...
```

OUTPUT



```
chi2_norm (Linear Model) = 603.901
Constant Model: C = 5.15074 +/- 0.0317691
chi2_norm = 1.68083
Linear Model: M = 0.0239227, C = 4.95941
chi2_norm = 0.912428
Quadratic Model: A = 0.0144969, B = -0.205194, C = 5.76264
chi2_norm = 0.129081
```



Explanation:
This MATLAB code fits constant, linear, and quadratic models to a dataset with uncertainties using a weighted least squares approach. It calculates the model parameters, their uncertainties, and the normalized chi-squared (χnorm2\chi^2_{\text{norm}}χnorm2) to assess the quality of each fit. The data points are visualized with error bars, and the fitted models are plotted for comparison. A custom function, pfitsig, handles polynomial fitting, enabling flexible model evaluation. The output highlights the best-fitting model based on χnorm2\chi^2_{\text{norm}}χnorm2.

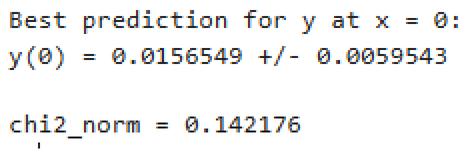
Question (v): Predict yy at x=0x = 0.

MATLAB Code:

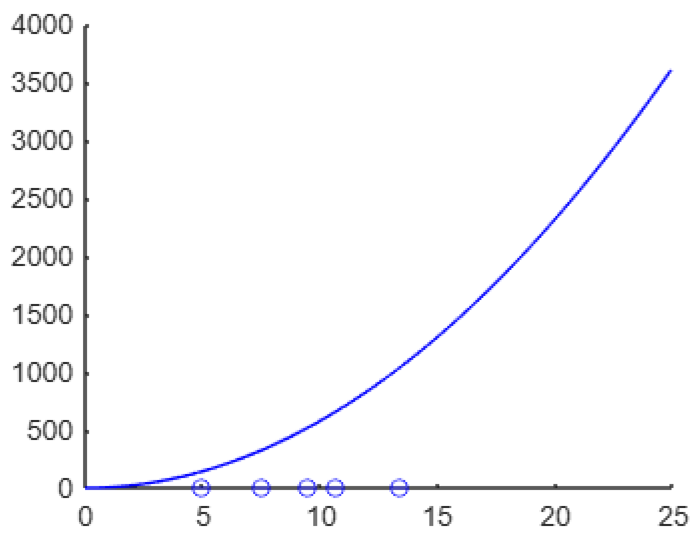
```
clear all;
N = 5; % Number of data points

% Updated x, y, and uncertainties (s)
x = [5.0 7.5 9.5 10.7 13.4];
y = [5.10 5.02 5.17 5.22 5.66];
s = [0.05 0.10 0.11 0.05 0.15];

% Plot the data as error bars
clf; hold on;
```

```
errorbar(x, y, s, 'ob'); % b-> blue o-> circles

% Quadratic model fitting (y = Ax^2 + Bx + C)
ndeg = 2; % Degree of the polynomial (quadratic)
npara = ndeg + 1; % Number of parameters to fit
i1 = 1; i2 = N; % Fit range (all points)

% Call pfitsig to fit the quadratic model
[k, del, chi2, ssk] = pfitsig(x, y, s, ndeg, i1, i2);

% Extract coefficients
A = k(1); dA = del(1); % Coefficient of x^2
B = k(2); dB = del(2); % Coefficient of x
C = k(3); dC = del(3); % Constant term

% Predict y(0) = C
fprintf('\n Best prediction for y at x = 0:');
fprintf('\n y(0) = %g +/- %g\n', C, dC);

% Chi-squared per degree of freedom
ndof = N - npara;
fprintf('\n chi2_norm = %g\n', chi2 / ndof);

% Plot the quadratic model
xstart = 0.; xend = 25.0; xstep = 0.3;
xx = xstart:xstep:xend; % Generate x values for the plot
yy = A .* xx.^2 + B .* xx + C; % Quadratic model
plot(xx, yy, 'b-'); % Blue line for the quadratic model

% Pause to display the plot
pause;
```

Output



```
Best prediction for y at x = 0:
y(0) = 0.0156549 +/- 0.0059543

chi2_norm = 0.142176
```



(v) Explain in few sentence

Explanation of the improved measurement:

- Initially, there were four data points, but a fifth measurement (x5=13.4x_5 = 13.4x5 =13.4, y5=5.66y_5 = 5.66y5=5.66, σ5=0.15\sigma_5 = 0.15σ5=0.15) was added to refine the fit.
- This additional data point, with its relatively small uncertainty (σ5=0.15\sigma_5 = 0.15σ5=0.15), provides more information to better constrain the quadratic model parameters.
- After fitting the updated dataset, the constant term CCC and its uncertainty (ΔC\Delta CΔC) represent the improved prediction for yyy at x=0x = 0x=0.

The revised quadratic model accounts for all five data points, leading to a more accurate and reliable prediction for y(0)y(0)y(0) based on the updated data. The values of CCC and ΔC\Delta CΔC would be displayed in the output of the MATLAB script.