Applied Numerical Methods with MATLAB for Engineers and Scientists (4th Edition)

Chapter 14, Problem 16P

Problem

Modify the linregr function in Fig. 14.15 so that it **(a)** computes and returns the standard error of the estimate and **(b)** uses the subplot function to also display a plot of the residuals (the predicted minus the measured *y*) versus *x*. Test it for the data from Examples 14.2 and 14.3.

FIGURE 14.15 An M-file to implement linear regression.

```
function [a, r2] = linregr(x,y)
% linregr: linear regression curve fitting
%   [a, r2] = linregr(x,y): Least squares fit of straight
%           line to data by solving the normal equations

% input:
%   x = independent variable
%   y = dependent variable
% output:
%   a = vector of slope, a(1), and intercept, a(2)
%   r2 = coefficient of determination

n = length(x);
if length(y)~=n, error('x and y must be same length'); end
x = x(:); y = y(:);     % convert to column vectors
sx = sum(x); sy = sum(y);
sx2 = sum(x.*x); sxy = sum(x.*y); sy2 = sum(y.*y);
a(1) = (n*sxy-sx*sy)/(n*sx2-sx^2);
a(2) = sy/n-a(1)*sx/n;
r2 = ((n*sxy-sx*sy)/sqrt(n*sx2-sx^2)/sqrt(n*sy2-sy^2))^2;
% create plot of data and best fit line
xp = linspace(min(x),max(x),2);
yp = a(1)*xp+a(2);
plot(x,y,'o',xp,yp)
grid on
```

Examples 14.2 Generating Uniform Random Values of Drag

Problem Statement. If the initial velocity is zero, the downward velocity of the freefalling bungee jumper can be predicted with the following analytical solution [Eq. (1.9)]:

$$v = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}}\, t\right)$$

Suppose that g = 9.81m/s^2, and m = 68.1 kg, but c_d is not known precisely. For example, you might know that it varies uniformly between 0.225 and 0.275 (i.e., ± 10% around a mean value of 0.25 kg/m). Use the rand function to generate 1000 random uniformly distributed values of c_d and then employ these values along with the analytical solution to compute the resulting distribution of velocities at t = 4 s.

Solution. Before generating the random numbers, we can first compute the mean velocity:

$$v_{\text{mean}} = \sqrt{\frac{9.81(68.1)}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{68.1}}\,4\right) = 33.1118\ \frac{\text{m}}{\text{s}}$$

We can also generate the range:

$$v_{\text{low}} = \sqrt{\frac{9.81(68.1)}{0.275}} \tanh\left(\sqrt{\frac{9.81(0.275)}{68.1}}\,4\right) = 32.6223\ \frac{\text{m}}{\text{s}}$$

$$v_{\text{high}} = \sqrt{\frac{9.81(68.1)}{0.225}} \tanh\left(\sqrt{\frac{9.81(0.225)}{68.1}}\,4\right) = 33.6198\ \frac{\text{m}}{\text{s}}$$

Thus, we can see that the velocity varies by

$$\Delta v = \frac{33.6198 - 32.6223}{2(33.1118)} \times 100\% = 1.5063\%$$

The following script generates the random values for c_d, along with their mean, standard deviation, percent variation, and a histogram:

clc, format short gn = 1000;t = 4;m = 68.1;g = 9.81;cd = 0.25;cdmin = cd−0.025, cdmax = cd + 0.0

The results are

meancd = 0.25018stdcd = 0.014528Deltacd = 9.9762

These results, as well as the histogram (Fig. 14.5*a*) indicate that rand has yielded 1000 uniformly distributed values with the desired mean value and range. The values can then be employed along with the analytical solution to compute the resulting distribution of velocities at t = 4 s.

vrand = sqrt(g*m./cdrand).*tanh(sqrt(g*cdrand/m)*t);meanv = mean(vrand)Deltav = (max(vrand)−mi

Eq. 1.9:

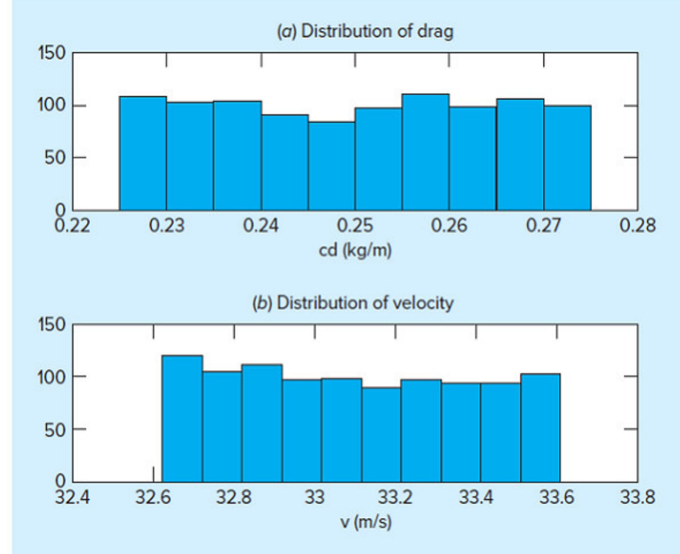
$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}}\,t\right)$$

The results are

meanv = 33.1151Deltav = 1.5048

These results, as well as the histogram (Fig. 14.5*b*), closely conform to our hand calculations.

FIGURE 14.5 Histograms of (*a*) uniformly distributed drag coefficients and (*b*) the resulting distribution of velocity.



Examples 14.3 Generating Normally Distributed Random Values of Drag

Problem Statement. Analyze the same case as in Example 14.2, but rather than employing a uniform distribution, generate normally distributed drag coefficients with a mean of 0.25 and a standard deviation of 0.01443.

Solution. The following script generates the random values for c_d, along with their mean, standard deviation, coefficient of variation (expressed as a %), and a histogram:

clc,format short gn = 1000;t = 4;m = 68.1;g = 9.81;cd = 0.25;stdev = 0.01443;r = randn(n,1);cdran

The results are

meancd = 0.24988stdevcd = 0.014465cvcd = 5.7887

These results, as well as the histogram (Fig. 14.6*a*) indicate that randn has yielded 1000 uniformly distributed values with the desired mean, standard deviation, and coefficient of variation. The values can then be employed along with the analytical solution to compute the resulting distribution of velocities at t = 4 s.

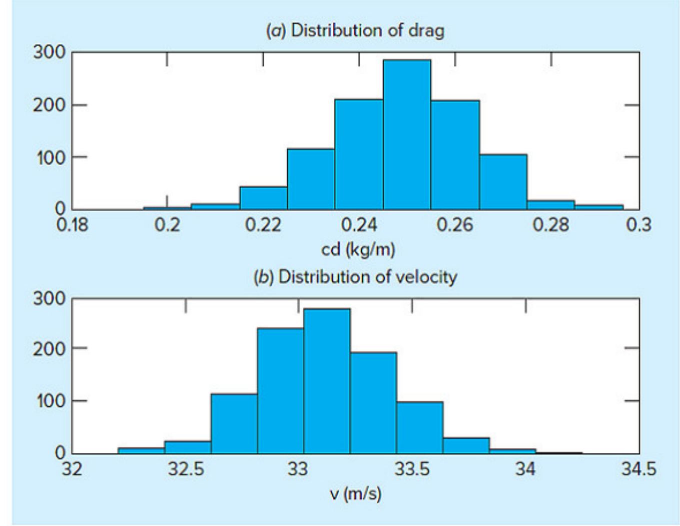
vrand = sqrt(g*m./cdrand).*tanh(sqrt(g*cdrand/m)*t);meanv = mean(vrand),stdevv = std(vrand)cvv

The results are

meanv = 33.117stdevv = 0.28839cvv = 0.8708

These results, as well as the histogram (Fig. 14.6*b*), indicate that the velocities are also normally distributed with a mean that is close to the value that would be computed using the mean and the analytical solution. In addition, we compute the associated standard deviation which corresponds to a coefficient of variation of ±0.8708%.

FIGURE 14.6 Histograms of (*a*) normally distributed drag coefficients and (*b*) the resulting distribution of velocity.



Step-by-step solution

Step 1 of 4

The MATLAB function described in the figure 14.15 of the book is to be modified to calculate the standard error of the estimate and to display a plot of the residuals versus *x*.

The Standard Error of Estimate is:

$$S_{y/x} = \sqrt{\frac{S_r}{n-2}}$$

Where

S_r is the sum of the squares

$$S_r = \sum_{i=1}^{n} \left(y_i - y_e\right)^2$$

Where

y_i is the actual data

y_e is the expected data

In the example, the slope and the intercept of the linear line are already calculated. The expected values of the *y* can be found from the slope and the intercept. Expected values can be subtracted from the real values to get the sum of the squares. Then, we can find the standard error of the estimate.

Comment

Step 2 of 4

The MATLAB code is given below:

Input:

```
function [a, Se] = linreg(x, y)
% linreg: Straight line fit
% Input:
% x = independent data
% y = dependent data
% Output:
% a = vector having slope and intercept
% Se = standard error of estimate
n = length(x);
% we have to calcualte the slope and the coefficient
% using the formula
% the formula is
% SLOPE = (n*sum(xy) - sum(x)*sum(y))/(n*sum(x^2) - (sum(x))^2)
% COEFF = mean(y) - SLOPE*mean(x)
% caclulating all summations
Sx = sum(x);
Sy = sum(y);
Sxy = sum(x.*y);
Sx2 = sum(x.*x);
a(1) = (n*Sxy - Sx*Sy)/(n*Sx2 - Sx^2);
a(2) = mean(y) - a(1)*mean(x);
% calculating the expected values
ye = a(1)*x + a(2);
% calculating the residuals
res = y - ye;
% calculating the sum of squares
Sr = sum(res.^2);
% calculating the standard error of the estimate
Se = sqrt(Sr/(n-2));
% plotting
subplot(2,1,1)
plot(x, y, 'o', x, ye)
xlabel('x')
title('Data and Best Fit Line')
grid on
subplot(2,1, 2)
plot(x, res)
xlabel('x')
ylabel('Residuals')
title('Plot of residuals')
grid on
end
```

Comment

Step 3 of 4

Checking the above code with the data in example.

x	10	20	30	40	50	60	70	80
y	25	70	380	550	610	1220	830	1450

Input:

```
x = [10:10:80];
y = [25 70 380 550 610 1220 830 1450];
[a Se] = linreg(x, y)
```

Comment

Step 4 of 4

Output:

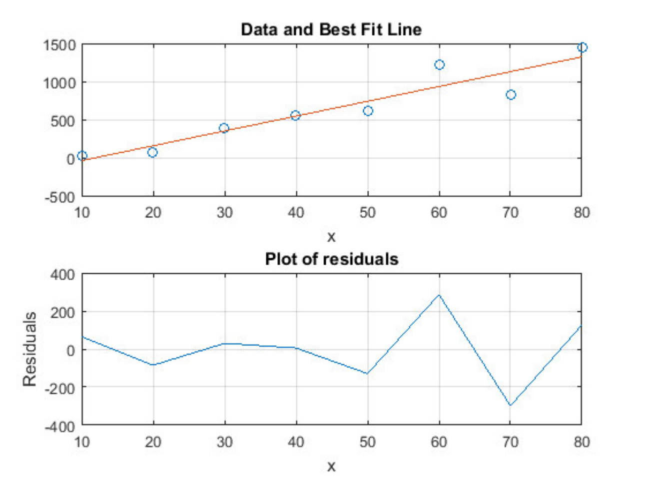
```
Command Window
>> x = [10:10:80];
>> y = [25 70 380 550 610 1220 830 1450];
>> [a Se] = linreg(x, y)

a =

   19.4702 -234.2857


Se =

  189.7885
```



The standard error of the estimate from MATLAB code is $\boxed{189.7885}$ which is same as that obtained in the example.

Comment