MATLAB 与工程应用

Statistics

Interpolation

Statistics and Histograms

MEAN	Average or mean value of array
MODE	Most frequently occurring value
MEDIAN	The middle value

```
M = mean(A)
M = mean(A,'all')
M = mean(A,dim)
```

```
M = mode(A)
M = mode(A,'all')
M = mode(A,dim)
```

```
M = median(A)
M = median(A,'all')
M = median(A,dim)
```

The way the data are spread around the mean can be described by a histogram plot. A histogram is a plot of the frequency of occurrence of data values versus the values themselves. It is a bar plot of the number of data values that occur within each range, with the bar centered in the middle of the range.

To ensure proper quality control, a thread manufacturer selects samples and tests them for breaking strength. Suppose that 20 thread samples are pulled until they break, and the breaking force is measured in newtons rounded off to integer values. The breaking force values recorded were 92, 94, 93, 96, 93, 94, 95, 96, 91, 93, 95, 95, 95, 92, 93, 94, 91, 94, 92 and 93. Plot the histogram of the data.

```
% Thread breaking strength data for 20 tests.
y = [92,94,93,96,93,94,95,96,91,93,...
   95,95,95,92,93,94,91,94,92,93];
% The six possible outcomes are 91,92,93,94,95,96.
x = 91:96;
hist(y,x),axis([90 97 0 6]),ylabel('Absolute Frequency'),...
   xlabel('Thread Strength (N)'),...
   title('Absolute Frequency Histogram for 20 Tests')
```

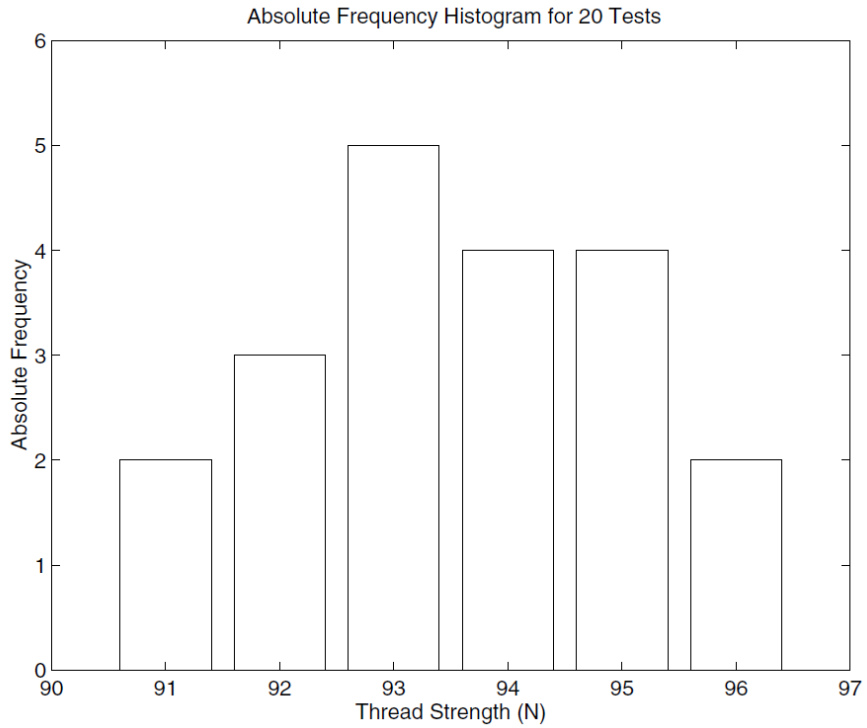


Figure 7.1–1 Histograms for 20 tests of thread strength.

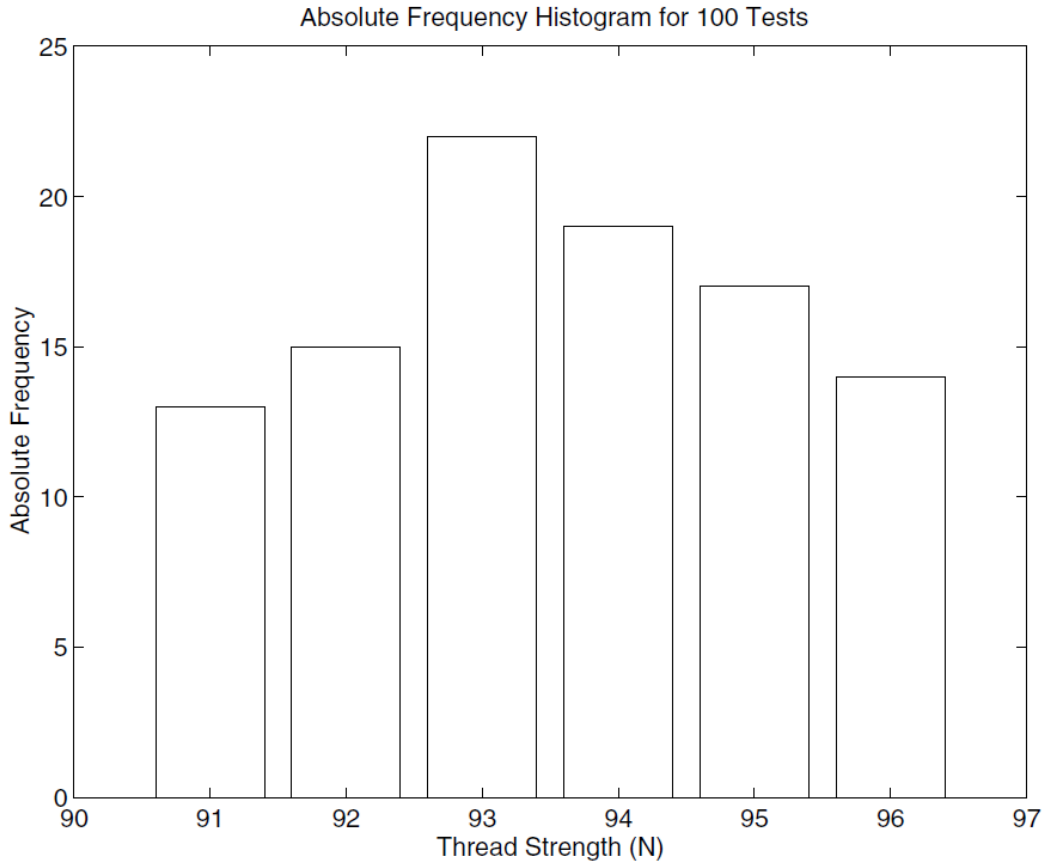
Absolute Frequency

The absolute frequency is the number of times a particular outcome occurs. For example, in 20 tests these data show that a 95 occurred 4 times. The absolute frequency is 4, and its relative frequency is 4/20, or 20 percent of the time.

Relative Frequency

The absolute frequency is the number of times a particular outcome occurs. For example, in 20 tests these data show that a 95 occurred 4 times. The absolute frequency is 4, and its relative frequency is 4/20, or 20 percent of the time. When there is a large amount of data, you can avoid typing in every data value by first aggregating the data. The following example shows how this done using the ones function. The following data were generated by testing 100 thread samples. The number of times 91, 92, 93, 94, 95, or 96 N was measured is 13, 15, 22, 19, 17, and 14, respectively.

```
% Thread strength data for 100 tests.
y = [91*ones(1,13),92*ones(1,15),93*ones(1,22),...
   94*ones(1,19),95*ones(1,17),96*ones(1,14)];
x = 91:96;
hist(y,x),ylabel('Absolute Frequency'),...
   xlabel('Thread Strength (N)'),...
   title('Absolute Frequency Histogram for 100 Tests')
```

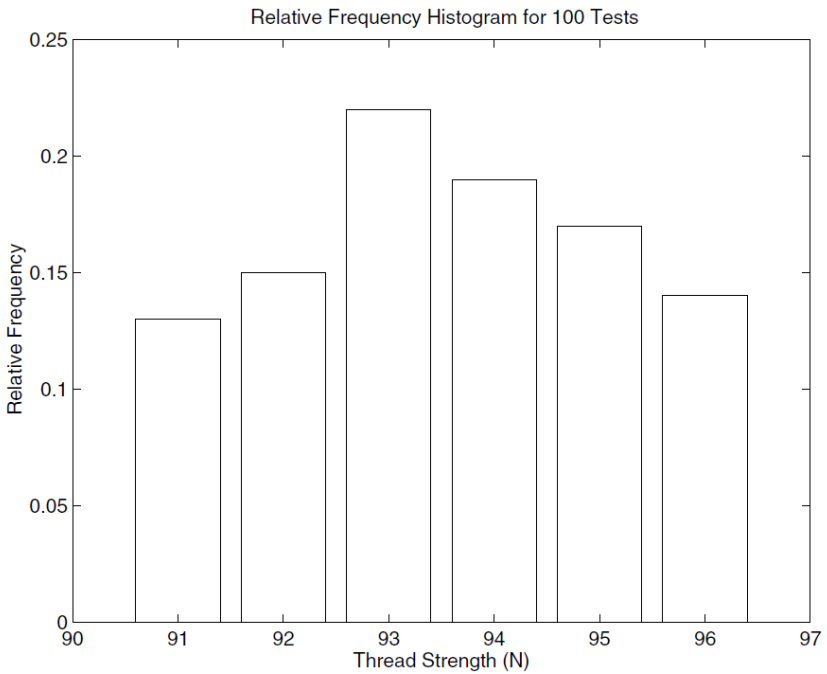


Absolute frequency histogram for 100 thread tests

The hist function is somewhat limited in its ability to produce useful histograms. Unless all the outcome values are the same as the bin centers (as is the case with the thread examples), This case occurs when you want to obtain a relative frequency histogram. In such cases you can use the bar function to generate the histogram. In such cases you can use the bar function to generate the histogram. The following script generates the relative frequency histogram for the 100 thread tests. Note that if you use the bar function, you must aggregate the data first.

```
% Relative frequency histogram using the bar function.
tests = 100;
y = [13,15,22,19,17,14]/tests;
x = 91:96;
bar(x,y),ylabel('Relative Frequency'),...
   xlabel('Thread Strength (N)'),...
   title('Relative Frequency Histogram for 100 Tests')
```

```
tests = 100;
y = [91*ones(1,13),92*ones(1,15),93*ones(1,22),...
   94*ones(1,19),95*ones(1,17),96*ones(1,14);
```



Relative frequency histogram for 100 thread tests

Histogram functions

Command	Description
`bar(x,y)`	Creates a bar chart of `y` versus `x`.
`hist(y)`	Aggregates the data in the vector `y` into 10 bins evenly spaced between the minimum and maximum values in `y`.
`hist(y,n)`	Aggregates the data in the vector `y` into `n` bins evenly spaced between the minimum and maximum values in `y`.
`hist(y,x)`	Aggregates the data in the vector `y` into bins whose center locations are speci ed by the vector `x`. The bin widths are the distances between the centers.
`[z,x] = hist(y)`	Same as `hist(y)` but returns two vectors `z` and `x` that contain the frequency count and the bin locations.
`[z,x] = hist(y,n)`	Same as `hist(y,n)` but returns two vectors `z` and `x` that contain the frequency count and the bin locations.
`[z,x] = hist(y,x)`	Same as `hist(y,x)` but returns two vectors `z` and `x` that contain the frequency count and the bin locations. The returned vector `x` is the same as the user-supplied vector `x`.

hist

Histogram plot (not recommended; use `histogram`)

`hist` is not recommended. Use `histogram` instead.

For more information, including suggestions on updating code, see Replace Discouraged Instances of hist and histc.

Syntax

```
hist(x)
hist(x,nbins)
hist(x,xbins)
```

histogram

R2019a

Histogram plot

expand all in page

Description

Histograms are a type of bar plot for numeric data that group the data into bins. After you create a `Histogram` object, you can modify aspects of the histogram by changing its property values. This is particularly useful for quickly modifying the properties of the bins or changing the display.

Creation

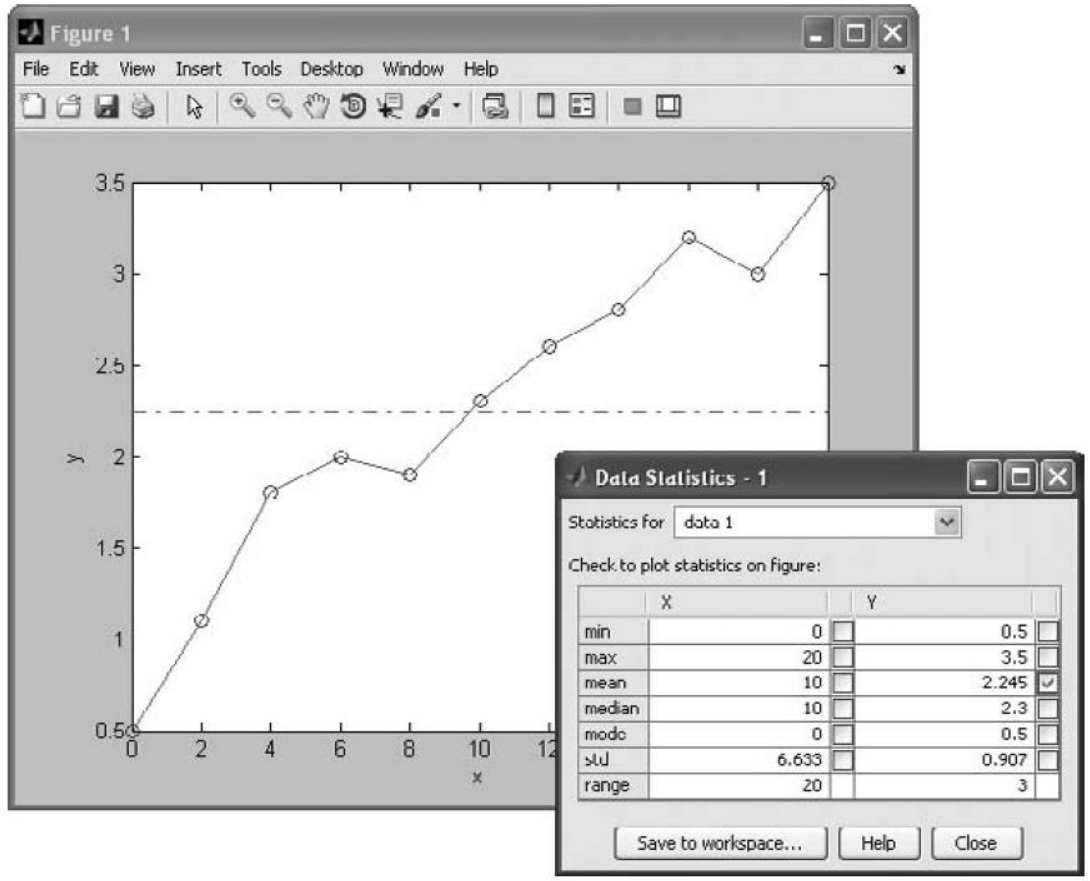
Syntax

```
histogram(X)
histogram(X,nbins)
histogram(X,edges)
histogram('BinEdges',edges,'BinCounts',counts)
```

In 50 tests of thread, the number of times 91, 92, 93, 94, 95, or 96 N was measured to be 7, 8, 10, 6, 12, and 7, respectively. Obtain the absolute and relative frequency histograms.

The Data Statistics Tool

With the Data Statistics tool you can calculate statistics for data and add plots of the statistics to a graph of the data. The tool is accessed from the Figure window after you plot the data. Click on the Tools menu, then select Data Statistics. To show the mean of the dependent variable (y) on the plot, click the box in the row labeled mean under the column labeled Y, as shown in the figure. A horizontal line is then placed on the plot at the mean. You can plot other statistics as well; there are shown in the figure. You can save the statistics to the workspace as a structure by clicking on the Save to Workspace button. This opens a dialog box that prompts you for a name for the structure containing the x data, and a name for the y data structure.



The Data Statistics tool

The Normal Distribution

Rolling a die is an example of a process whose possible outcomes are a limited set of number, namely, the integers from 1 to 6. For such processes the probability is a function of a discrete-valued variable, wat is , a variable having a limited number of values. For example, the following table gives the measured heights of 100 men 20 years of age. The heights were recorded to the nearest ½ in., so the height variable is discrete-valued.

Scaled Frequency Histogram

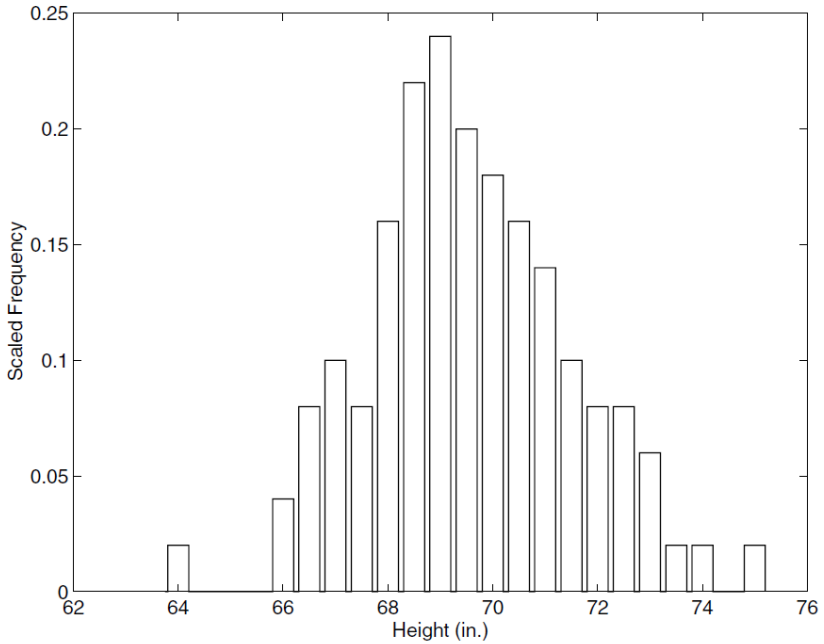
You can plot the data as a histogram using either the absolute or relative frequencies. However, another useful histogram uses data scaled so that the total area under the histogram's rectangles is 1. This scaled frequency histogram is the absolute frequency histogram divided by the total area of that histogram. The area of each rectangle on the absolute frequency histogram equals the bin width times the absolute frequency for that bin. Because all the rectangles have the same width, the total area is the bin width times the sum of the absolute frequencies.

Height data for men 20 years of age

Height (in.)	Frequency	Height (in.)	Frequency
64	1	70	9
64.5	0	70.5	8
65	0	71	7
65.5	0	71.5	5
66	2	72	4
66.5	4	72.5	4
67	5	73	3
67.5	4	73.5	1
68	8	74	1
68.5	11	74.5	0
69	12	75	1
69.5	10		

```
% Absolute frequency data.
y_abs=[1,0,0,0,2,4,5,4,8,11,12,10,9,8,7,5,4,4,3,1,1,0,1];
binwidth = 0.5;
% Compute scaled frequency data.
area = binwidth*sum(y_abs);
y_scaled = y_abs/area;

% De ne the bins.
bins = 64:binwidth:75;
% Plot the scaled histogram.
bar(bins,y_scaled),...
   ylabel('Scaled Frequency'),xlabel('Height (in.)')
```



Scaled histogram of height data

Random Number Generation

We often do not have a simple probability distribution to describe the distribution of outcomes in many engineering applications. For example, the probability that a circuit consisting of many components will fail is a function of the number and the age of the components, but we often cannot obtain a function to describe the failure probability. In such cases we often resort to simulation to make predictions. The simulation program is executed many times, using a random set of numbers to represent the failure of one or more components, and the results are used to estimate the desired probability.

Random number functions

rand

Uniformly distributed random numbers

Syntax

```
X = rand
X = rand(n)
X = rand(sz1,...,szN)
X = rand(sz)
```

```
X = rand(__,typename)
X = rand(__,'like',p)
```

> **Note**
>
> The `'seed'`, `'state'`, and `'twister'` inputs to the `rand` function are not recommended. Use the `rng` function instead. For more information, see Replace Discouraged Syntaxes of rand and randn.

Example 1 — Retrieve and Restore Generator Settings

Save the current generator settings in `s`:

```
s = rng;
```

Call `rand` to generate a vector of random values:

```
x = rand(1,5)

x =

    0.8147    0.9058    0.1270    0.9134    0.6324
```

Restore the original generator settings by calling `rng`. Generate a new set of random values and verify that `x` and `y` are equal:

```
rng(s);
y = rand(1,5)

y =

    0.8147    0.9058    0.1270    0.9134    0.6324
```

You can use the rand function to generate random numbers in an interval other than [0, 1]. For example, to generate values in the interval [2, 10], generate a random number between 0 and 1, multiply it by 8 (the difference between the upper and lower bounds), and add the lower bound (2). The result is a value that is uniformly distributed in the interval [2, 10]. The general formula for generating a uniformly distributed random number y in the interval [a, b] is

$$y = (b - a)x + a$$

where x is a random number uniformly distributed in the interval [0, 1]. For example, to generate a vector y containing 1000 uniformly distributed random numbers in the interval [2, 10], you type $y = 8 * rand(1{,}1000) + 2$. You can check the results with the mean, min, and max functions. You should obtain values close to 6, 2, and 10, respectively.

You can use rand to generate random results for games involving dice, for example, but you must use it to create integers. An easier way is to uses the randperm(n) function, which generates a random permutation of the integers from 1 to n. For example, randperm(6) might generate the vector [3 2 6 4 1 5], or some other permutation of the numbers from 1 to 6.

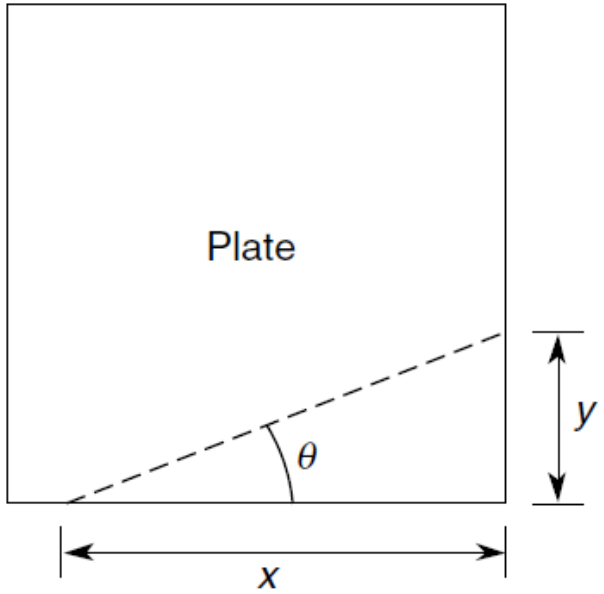
Generating a random number having a standard deviation of σ and a mean of μ

$$y = \sigma x + \mu$$

For example, to generate a vector y containing 2000 random numbers normally distributed with a mean of 5 and a standard deviation of 3, you type `y = 3*randn(1,2000) + 5`. You can check the results with the `mean` and `std` functions. You should obtain values close to 5 and 3, respectively.

Statistical Analysis and Manufacturing Tolerances

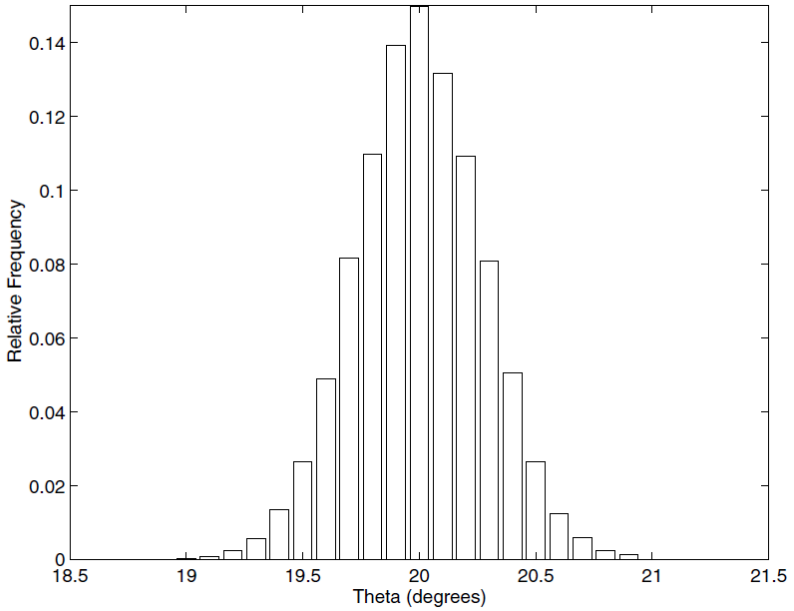
Suppose you must cut a triangular piece off the corner of a square plate by measuring the distances x and y from the corner. The desired value of x is 10 in., and the desired value of θ is 20°. This requires that y=3.64 in. We are told that measurements of x and y are normally distributed with means of 10 and 3.64, respectively, with a standard deviation equal to 0.05 in. Determine the standard deviation of θ and plot the relative frequency histogram for θ.



Dimensions of a triangular cut

We see that the angle θ is determined by $\theta = tan^{-1}(\frac{y}{x})$. We can find the statistical distribution of θ by creating random variables x and y that have means of 10 and 3.64, respectively, with a standard deviation of 0.05. The random variable θ is then found by calculating$\theta = tan^{-1}\left(\frac{y}{x}\right)$ for each random pair (x,y)

```
s = 0.05; % standard deviation of x and y
n = 8000; % number of random simulations
x = 10 + s*randn(1,n);
y = 3.64 + s*randn(1,n);
theta = (180/pi)*atan(y./x);
mean_theta = mean(theta)
sigma_theta = std(theta)
xp = 19:0.1:21;
z = hist(theta,xp);
yp = z/n;
bar(xp,yp),xlabel('Theta (degrees)'),...
    ylabel('Relative Frequency')
```



Scaled histogram of the angle θ

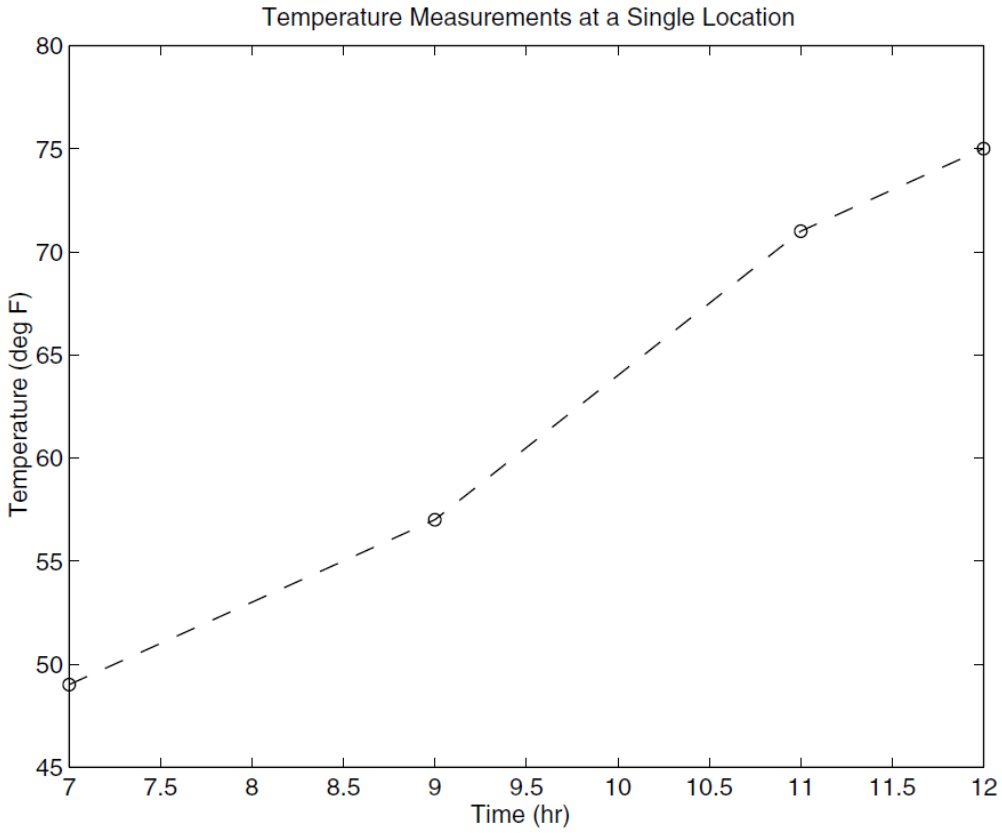
The choice of 8000 simulations was a compromise between accuracy and the amount of time required to do the calculations. You should try different values of n and compare the results. The results gave a mean of $19.9993°$ for θ with a standard deviation of $0.2730°$. Although the plot resembles the normal distribution, the values of θ are not distribute normally. From the histogram we can calculate that approximately 65 percent of the values of θ lie between 19.8 and 20.2. This range corresponds to a standard deviation of $0.2°$, not $0.273°$ as calculated from the simulation data. Thus the curve is not a normal distribution.

Interpolation

Paired data might represent a *cause and effect,* or *input-output relationship,* such as the current produced in a resistor as a result of an applied voltage, or a *time history,* such as the temperature of an object as a function of time. Another type of paired data represents a *pro le,* such as a road pro le (which shows the height of the road along its length). In some applications we want to estimate a variable's value between the data points. This process is called *interpolation*. In other cases we might need to estimate the variable's value outside the given data range. This process is called *extrapolation*. Interpolation and extrapolation are greatly aided by plotting the data. Such plots, some perhaps using logarithmic axes, often help to discover a functional description of the data.

Suppose we have the following temperature measurements, taken once an hour starting at 7:00 A.M. The measurements at 8 and 10 A.M. are missing for some reason, perhaps because of equipment malfunction.

Time	7 A.M.	9 A.M.	11 A.M.	12 noon
Temperature (°F)	49	57	71	75



A plot of temperature data versus time

Linear interpolation in MATLAB is obtained with the `interp1` and `interp2` functions. Suppose that `x` is a vector containing the independent variable data and that `y` is a vector containing the dependent variable data. If `x_int` is a vector containing the value or values of the independent variable at which we wish to estimate the dependent variable, then typing `interp1(x,y,x_int)` produces a vector the same size as `x_int` containing the interpolated values of `y` that correspond to `x_int`. For example, the following session produces an estimate of the temperatures at 8 and 10 A.M. from the preceding data. The vectors `x` and `y` contain the times and temperatures, respectively.

```
>>x = [7, 9, 11, 12];
>>y = [49, 57, 71, 75];
>>x_int = [8, 10];
>>interp1(x,y,x_int)
ans =
     53
     64
```

Linear interpolation functions

Command	Description
`y_int=interp1(x,y,x_int)`	Used to linearly interpolate a function of one variable: $y = f(x)$. Returns a linearly interpolated vector `y_int` at the speci ed value `x_int`, using data stored in `x` and `y`.
`z_int=interp2(x,y,z,x_,y_int)`	Used to linearly interpolate a function of two variables: $y = f(x, y)$. Returns a linearly interpolated vector `z_int` at the speci ed values `x_int` and `y_int`, using data stored in `x`, `y`, and `z`.

You must keep in mind two restrictions when using the `interp1` function. The values of the independent variable in the vector `x` must be in ascending order, and the values in the interpolation vector `x_int` must lie within the range of the values in `x`. Thus we cannot use the `interp1` function to estimate the temperature at 6 A.M., for example.

The `interp1` function can be used to interpolate in a table of values by defining `y` to be a matrix instead of a vector. For example, suppose that we now have temperature measurements at three locations and the measurements at 8 and 10 A.M. are missing for all three locations. The data are as follows:

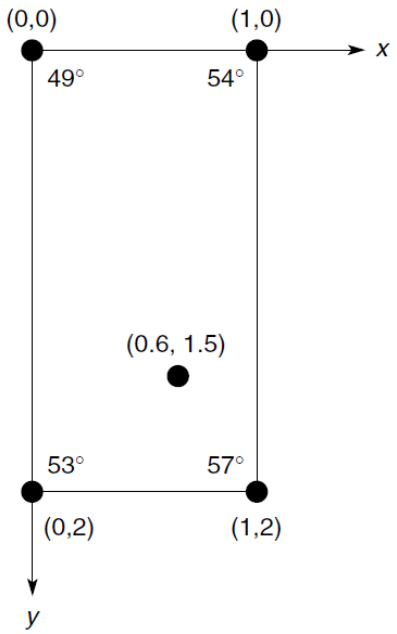
	Temperature (°F)		
Time	**Location 1**	**Location 2**	**Location 3**
7 A.M.	49	52	54
9 A.M.	57	60	61
11 A.M.	71	73	75
12 noon	75	79	81

We define `x` as before, but now we define `y` to be a matrix whose three columns contain the second, third, and fourth columns of the preceding table. The following session produces an estimate of the temperatures at 8 and 10 A.M. at each location.

```
>>x = [7, 9, 11, 12]';
>>y(:,1) = [49, 57, 71, 75]';
>>y(:,2) = [52, 60, 73, 79]';
>>y(:,3) = [54, 61, 75, 81]';
>>x_int = [8, 10]';
>>interp1(x,y,x_int)
ans =
     53.0000     56.0000     57.5000
     64.0000     65.5000     68.0000
```

Two-Dimensional Interpolation

Now suppose that we have temperature measurements at four locations at 7 A.M. These locations are at the corners of a rectangle 1 mile wide and 2 mile long. Assigning a coordinate system origin (0,0) to the first location, and the coordinates of the other locations are (1,0), (1,2), and (0,2); The temperature is a function of two variables, the coordinates x and y. MATLAB provides the interp2 function to interpolate functions of two variables. If the function is written as $z = f(x, y)$ and we wish to estimate the value of z for $x = x_i$ and $y = y_i$, the syntax is interp2(x,y,z,x_i,y_i).



Temperature measurements at four locations

Suppose we want to estimate the temperature at the point whose coordinates are $(0.6, 1.5)$. Put the x coordinates in the vector `x` and the y coordinates in the vector `y`. Then put the temperature measurements in a matrix `z` such that going across a row represents an increase in x and going down a column represents an increase in y. The session to do this is as follows:

```
>>x = [0,1];
>>y = [0,2];
>>z = [49,54;53,57]
z =
    49    54
    53    57
>>interp2(x,y,z,0.6,1.5)
ans =
     54.5500
```

Thus the estimated temperature is 54.55°F.

Cubic Spline Interpolation

High-order polynomials can exhibit undesired behavior between the data points, and this can make them unsuitable for interpolation. A widely used alternative procedure is to t the data points using a lower -order polynomial between *each pair* of adjacent data points. This method is called *spline* interpolation and is so named for the splines used by illustrators to draw a smooth curve through a set of points.

Spline interpolation obtains an exact t that is also smooth. The most common procedure uses cubic polynomials, called *cubic splines,* and thus is called *cubic spline interpolation*. If the data are given as n pairs of (x, y) values, then $n - 1$ cubic polynomials are used. Each has the form

$$y_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

for $x_i \leq x \leq x_{i+1}$ and $i = 1, 2, \ldots, n - 1$. The coef cients a_i, b_i, c_i, and d_i for each polynomial are determined so that the following three conditions are satis- ed for each polynomial:

1. The polynomial must pass through the data points at its endpoints at x_i and x_{i+1}.
2. The slopes of adjacent polynomials must be equal at their common data point.
3. The curvatures of adjacent polynomials must be equal at their common data point.

MATLAB provides the `spline` command to obtain a cubic spline interpolation. Its syntax is `y_int = spline(x,y,x_int)`, where `x` and `y` are vectors containing the data and `x_int` is a vector containing the values of the independent variable x at which we wish to estimate the dependent variable y. The result `y_int` is a vector the same size as `x_int` containing the interpolated values of y that correspond to `x_int`. The spline t can be plotted by plotting the vectors `x_int` and `y_int`. For example, the following session produces and plots a cubic spline t to the preceding data, using an increment of 0.01 in the x values.

For example, a set of cubic splines for the temperature data given earlier follows (y represents the temperature values, and x represents the hourly values). The data are repeated here.

x	7	9	11	12
y	49	57	71	75

We will shortly see how to use MATLAB to obtain these polynomials. For $7 \le x \le 9$,

$$y_1(x) = -0.35(x - 7)^3 + 2.85(x - 7)^2 - 0.3(x - 7) + 49$$

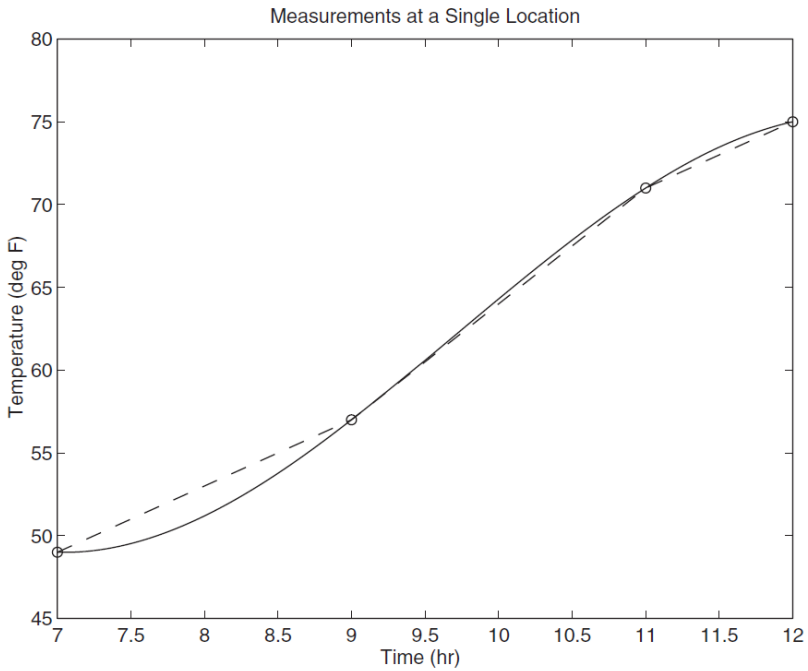
For $9 \le x \le 11$,

$$y_2(x) = -0.35(x - 9)^3 + 0.75(x - 9)^2 + 6.9(x - 9) + 57$$

For $11 \le x \le 12$,

$$y_3(x) = -0.35(x - 11)^3 - 1.35(x - 11)^2 + 5.7(x - 11) + 71$$

```
>>x = [7,9,11,12];
>>y = [49,57,71,75];
>>x_int = 7:0.01:12;
>>y_int = spline(x,y,x_int);
>>plot(x,y,'o',x,y,'--',x_int,y_int),...
   xlabel('Time (hr)'),ylabel('Temperature (deg F)'), ...
   title('Measurements at a Single Location'), ...
   axis([7 12 45 80])
```



Linear and cubic-spline interpolation of temperature data

Polynomial interpolation functions

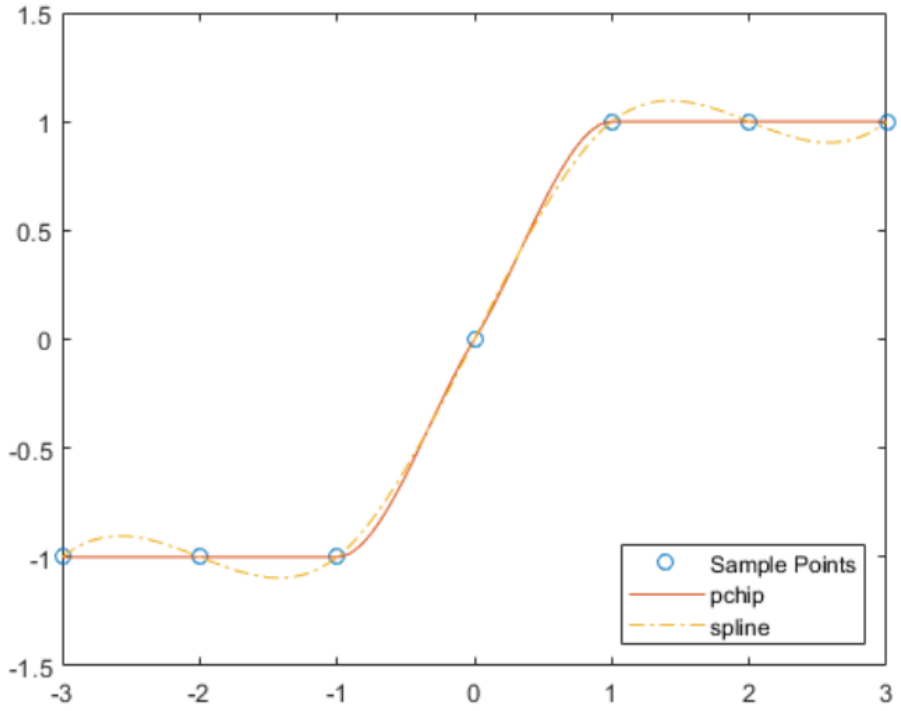
Command	Description
`y_est = interp1(x,y,x_est, method)`	Returns a column vector `y_est` that contains the estimated values of *y* that correspond to the *x* values speci ed in the vector `x_est`, using interpolation speci ed by `method`. The choices for `method` are 'nearest', 'linear', 'spline', 'pchip', and 'cubic'.
`y_int = spline(x,y,x_int)`	Computes a cubic spline interpolation where `x` and `y` are vectors containing the data and `x_int` is a vector containing the values of the independent variable *x* at which we wish to estimate the dependent variable *y*. The result `y_int` is a vector the same size as `x_int` containing the interpolated values of *y* that correspond to `x_int`.
`y_int = pchip (x,y,x_int)`	Similar to `spline` but uses piecewise cubic Hermite polynomials for interpolation to preserve shape and respect monotonicity.
`[breaks, coeffs, m, n] = unmkpp(spline(x,y))`	Computes the coef cients of the cubic spline polynomials for the data in `x` and `y`. The vector `breaks` contains the *x* values, and the matrix `coeffs` is an $m \times n$ matrix containing the polynomial coef cients. The scalars `m` and `n` give the dimensions of the matrix `coeffs; m` is the number of polynomials, and `n` is the number of coef cients for each polynomial.

Data Interpolation Using `spline` and `pchip`

Compare the interpolation results produced by `spline` and `pchip` for two different functions.

Create vectors of `x` values, function values at those points `y`, and query points `xq`. Compute interpolations at the query points using both `spline` and `pchip`. Plot the interpolated function values at the query points for comparison.

```
x = -3:3;
y = [-1 -1 -1 0 1 1 1];
xq1 = -3:.01:3;
p = pchip(x,y,xq1);
s = spline(x,y,xq1);
plot(x,y,'o',xq1,p,'-',xq1,s,'-.')
legend('Sample Points','pchip','spline','Location','SouthEast')
```



In this case, `pchip` is favorable since it does not oscillate as freely between the sample points.

Homework for Week 6 （Part I）

Problem 1 Write a script le to play a simple number guessing game as follows. The script should generate a random integer in the range 1, 2, 3, . . . , 14, 15. It should provide for the player to make repeated guesses of the number, and it should indicate if the player has won or give the player a hint after each wrong guess. The responses and hints are as follows:

- "You won" and then stop the game.
- "Very close" if the guess is within 1 of the correct number.
- "Getting close" if the guess is within 2 or 3 of the correct number.
- "Not close" if the guess is not within 3 of the correct number.

Problem 2

Interpolation is useful when one or more data points are missing. This situation often occurs with environmental measurements, such as temperature, because of the dif culty of making measurements around the clock. The following table of temperature versus time data is missing readings at 5 and 9 hours. Use linear interpolation with MATLAB to estimate the temperature at those times.

Time (hours, P.M.)	1	2	3	4	5	6	7	8	9	10	11	12
Temperature (°C)	10	9	18	24	?	21	20	18	?	15	13	11

Problem 3

The following table gives temperature data in °C as a function of time of day and day of the week at a speci c location. Data are missing for the entries marked with a question mark (?). Use linear interpolation with MATLAB to estimate the temperature at the missing points.

	Day				
Hour	**Mon**	**Tues**	**Wed**	**Thurs**	**Fri**
1	17	15	12	16	16
2	13	?	8	11	12
3	14	14	9	?	15
4	17	15	14	15	19
5	23	18	17	20	24

Problem 4

Computer-controlled machines are used to cut and to form metal and other materials when manufacturing products. These machines often use cubic splines to specify the path to be cut or the contour of the part to be shaped. The following coodinates specify the shape of a certain car's front fender. Fit a series of cubic splines to the coordinates, and plot the splines along with the coordinate points.

x (ft)	0	0.25	0.75	1.25	1.5	1.75	1.875	2	2.125	2.25
y (ft)	1.2	1.18	1.1	1	0.92	0.8	0.7	0.55	0.35	0