Generating random numbers: The rand() function

The `rand()` function generates random numbers between 0 and 1 that are distributed uniformly (all numbers are equally probable). If you attempt the extra credit, you likely will need to use the `rand()` function.

- `rand(1)` – generates a single random number
- `rand(N)` – generates a NxN array of random numbers
- `rand(1,N)` – generates an array of N random numbers

Example:
```matlab
number1 = rand(1)
number2 = rand(1)
N = 3;
Nnumbers = rand(1,N)
Nnumbers(2)
```
After execution:
```plaintext
number1 = 0.42932
number2 = 0.29074
Nnumbers =
  0.27551  0.33193  0.71718
ans = 0.33193
```

If you want to generate random numbers from 0 to 10, you multiply the random number by 10.

Example:
```matlab
multfactor = 10;
randomArray = rand(1,5)
multfactor*randomArray
```
After execution:
```plaintext
randomArray =
  0.74785  0.20773  0.23973  0.60396  0.47957
ans =
  7.4785  2.0773  2.3973  6.0396  4.7957
```

If you want to generate N random numbers from A to B, use the following formula:

```matlab
A + (B-A)*rand(1,N);
```

“(B-A)” makes the difference between the lowest and highest random number the same as the difference between A and B.

“A +” adjusts the lower part of the random number range to A
Example:
\[ A = 5; \ B = 10; \]
\[ \text{randomArray} = A + (B-A) \cdot \text{rand}(1,5); \]

After execution:
\[ \text{intarray} = \]
\[ 9.7675 \ 9.3214 \ 6.5982 \ 7.0010 \ 9.8172 \]

If you want to generate random integers from A to B in Matlab, you can use the \texttt{randi()} function. However, this function does not exist in Octave, so let’s create our own random integer generator. Let’s first look try using the formula for creating random numbers from A to B.

\[ \text{randomArray} = A + (B-A) \cdot \text{rand}(1,5); \]

If we tried \( A=1, \ B=10, \)

\[ 1 + (10-1) \cdot \text{rand}(1,5) \]
creates random numbers from 1 to 10. We can use the \texttt{floor()} command to round the random numbers down to integers. For example, \( \text{floor}(9.6234) \) is 9.

\[ \text{intArray} = \text{floor(randomArray)} \]

This creates a list of integers 1 to 9, which is too small of a range. One way around this problem is to add 1 to \( B-A. \)

\[ \text{randomArray} = A + (B-A+1) \cdot \text{rand}(1,5); \]
\[ \text{intArray} = \text{floor(randomArray)} \]

\[ 1 + (10) \cdot \text{rand}(1,5) \]
creates random numbers from 1 to 11. The \texttt{floor()} function creates an array of integers ranging from 1 to 10.

Example: Generate random integers from 5 to 10.
\[ A = 5; \ B = 10; \]
\[ \text{randomArray} = (A-1) + (B-(A-1)) \cdot \text{rand}(1,5); \]
\[ \text{intArray} = \text{floor(randomArray)} + 1 \]

After execution:
\[ \text{intarray} = \]
\[ 10 \ 10 \ 6 \ 7 \ 9 \]
PLOTTING:

With many other computer languages, such as Fortran, you can write the output to a file but must plot the data with a separate program (such as Excel or Gnuplot). However, Matlab/Octave has a built-in plotting program. I won’t be showing you all the features of this program, but it can do quite a lot.

You plot data with the `plot(x,y)` function. This function requires at minimum two arguments, the x-coordinates and y-coordinates.

Example: Plot $y(x) = x^2$ for from $x = 0$ to 10.

In `main.m`:
```
x = (0:1:10);
for i=1:numel(x)
  y(i) = x(i)^2;
end
plot(x,y)
```
% The first set of values will be treated as the x-coordinates
% The second set of values will be treated as the y-coordinates

Type ‘main’ at the command line and the following plot should appear.
You can add axis labels, a title, and gridlines too. IMPORTANT: You must create a graph before you add the title and labels.

In main.m:
```matlab
x = (0:1:10);
for i=1:numel(x)
    y(i) = x(i)^2;
end
plot(x,y)
xlabel('x, meters')
ylabel('y, meters squared')
title('A simple plot');
grid on;
```

Type ‘main’ at the command line and the following plot should appear.

You can add labels to the data and change the line style by adding some information in the `plot` command within single quotation marks. Change the `plot(x,y)` command in the previous example to

```matlab
plot(x,y, '--xr')
```

-- changes the line type to dashed
x puts x-marks at the data points
r changes the line color to red
Table 5.2 (page 147) in your book shows you the various line type, point type, and color options. Here are a few more:

<table>
<thead>
<tr>
<th>Line Type</th>
<th>Point Type</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>- solid</td>
<td>. point</td>
<td>b blue</td>
</tr>
<tr>
<td>: dotted</td>
<td>x x-mark</td>
<td>g green</td>
</tr>
<tr>
<td>-. dash-dot</td>
<td>+ plus</td>
<td>r red</td>
</tr>
<tr>
<td>-- dashed</td>
<td>* star</td>
<td>k black</td>
</tr>
</tbody>
</table>

**Changing Axes:**

Matlab/Octave usually will choose appropriate ranges for the axes. However, you can change the axes’ range using the `axis()` function. Notice that an array of values is sent to the `axis()` function.

```matlab
axis([ xmin, xmax, ymin, ymax ])  
```

In m-file:
```
x = (0:1:10);
y = x.^2;
plot(x,y)
xlabel('x, meters')
ylabel('y, meters squared')
title('A simple plot');
axis([2,8, -20,80])
grid on;
```
Although you have data from $x = 0$ to 10 and $y = 0$ to 100, you only display a subset of that data.

**Plotting Points (no line):**

Sometimes you just want to plot data points without a line connecting them. Simply omit the line type in the format string.

In m-file:
```matlab
x = (0:1:10);
y = x.^2;
plot(x,y,'or')  % NOTICE: The dashes have been omitted.
xlabel('x, meters')
ylabel('y, meters squared')
title('A simple plot');
grid on;
```
Adding text to a figure:

If you want to add text to a figure, use the `text()` function.

text( x-coordinate, y-coordinate, ‘string’)

In m-file:
```matlab
x = (0:1:10);
y = x.^2;
plot(x,y,'or')
xlabel('x, meters')
ylabel('y, meters squared')
title('A simple plot');
text(3,20,'A very simple string')
grid on;
```