9.07 Matlab Tutorial

Camilo Lamus
lamus@mit.edu

October 3, 2010

Aim

- Transform students into Matlab ninjas!

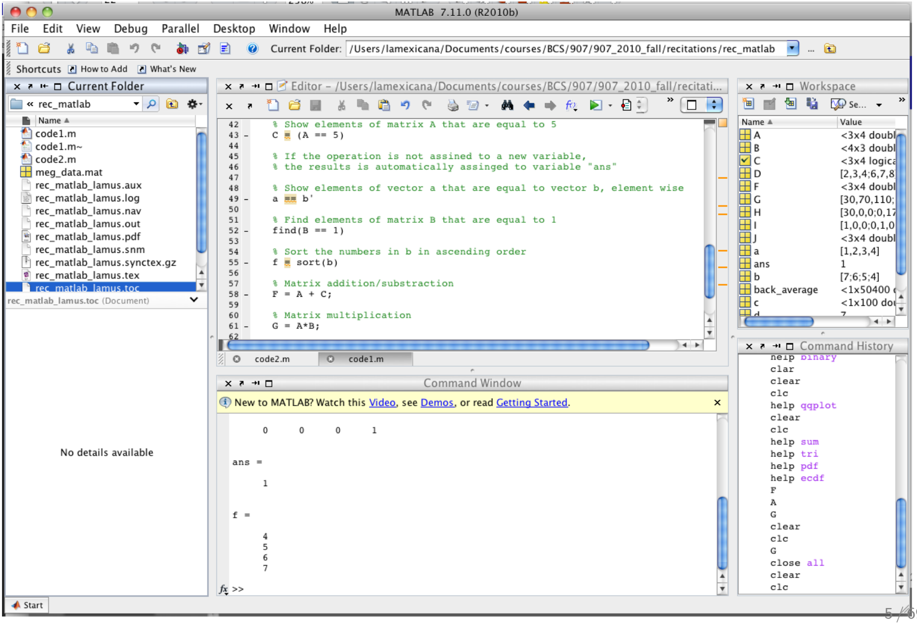
Contents

- Matrix operations: `a + b, a*b, A.*B, A.^B, sort, etc`
- Loops: `for i=1:n, while "statement is true"`
- Some useful functions: `rand, randn, min, max, etc`
- Function for displaying results:
  `figure, plot, subplot, bar, hist, title, xlabel, etc`
- Translating algorithms into Matlab code.

Getting started

- Getting Matlab: http://web.mit.edu/student-matlab/
- Matlab support: http://www.mathworks.com/support/, in search support box select "Function list for all products"
- Matlab support: Goto Help>Product Help
- In command prompt type: `help <name of funct>`

The Matlab interface



Constructing matrices

- Generate the (1×4) row vector $a = [1\ 2\ 3\ 4]$

```
a = [1, 2, 3, 4];
```

Constructing matrices

- Generate the (1×4) row vector $a = [1\ 2\ 3\ 4]$

```
a = [1, 2, 3, 4];
```

- Generate the (4×1) column vector $b = [7\ 6\ 5\ 4]'$

```
b = [7; 6; 5; 4];
```

Constructing matrices

- Generate the (1×4) row vector $a = [1\ 2\ 3\ 4]$

```
a = [1, 2, 3, 4];
```

- Generate the (4×1) column vector $b = [7\ 6\ 5\ 4]'$

```
b = [7; 6; 5; 4];
```

- Generate the row vector $c = [1\ 2\ \ldots\ 100]$

```
c = [1:1:100];
```

Constructing matrices

- Generate the (1×4) row vector $a = [1\ 2\ 3\ 4]$

```
a = [1, 2, 3, 4];
```

- Generate the (4×1) column vector $b = [7\ 6\ 5\ 4]'$

```
b = [7; 6; 5; 4];
```

- Generate the row vector $c = [1\ 2\ \ldots\ 100]$

```
c = [1:1:100];
```

- Generate the 3×4 matrix $A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$

```
A = [1 2 3 4;5 6 7 8;9 10 11 12];
```

Constructing matrices

- Accessing a portion of the matrix A
  - Take the element in the 2nd row and 3rd column of matrix A

    ```
    d = A(2,3);
    ```

  - Take the elements in the 1st through 2nd rows and 2nd through 4th columns

    ```
    D = A(1:2,2:4);
    ```

Constructing matrices

- Accessing a portion of the matrix A
  - Take the element in the 2nd row and 3rd column of matrix A

```
d = A(2,3);
```

  - Take the elements in the 1st through 2nd rows and 2nd through 4th columns

```
D = A(1:2,2:4);
```

- Find the transpose of matrix, $B = A'$

```
B = A';
```

Constructing matrices

- Accessing a portion of the matrix A
  - Take the element in the 2nd row and 3rd column of matrix A

```
d = A(2,3);
```

  - Take the elements in the 1st through 2nd rows and 2nd through 4th columns

```
D = A(1:2,2:4);
```

- Find the transpose of matrix, $B = A'$

```
B = A';
```

- Not using a semicolon ";" after an expression prints the output

```
B = A'
```

Constructing matrices

- Accessing a portion of the matrix A
  - Take the element in the 2nd row and 3rd column of matrix A

    ```
    d = A(2,3);
    ```

  - Take the elements in the 1st through 2nd rows and 2nd through 4th columns

    ```
    D = A(1:2,2:4);
    ```

- Find the transpose of matrix, $B = A'$

  ```
  B = A';
  ```

- Not using a semicolon ";" after an expression prints the output

  ```
  B = A'
  ```

- Generate a (3×3) identity matrix

  ```
  I = eye(3);
  ```

Constructing matrices

- Accessing a portion of the matrix A
  - Take the element in the 2nd row and 3rd column of matrix A

    ```
    d = A(2,3);
    ```

  - Take the elements in the 1st through 2nd rows and 2nd through 4th columns

    ```
    D = A(1:2,2:4);
    ```

- Find the transpose of matrix, $B = A'$

  ```
  B = A';
  ```

- Not using a semicolon ";" after an expression prints the output

  ```
  B = A'
  ```

- Generate a (3×3) identity matrix

  ```
  I = eye(3);
  ```

- Load data from a ".mat" file meg_data.mat

  ```
  load meg_data
  ```

Matrix operations

- Show the elements of matrix A that are equal to 5

```
C = (A == 5)
```

Matrix operations

- Show the elements of matrix A that are equal to 5

```
C = (A == 5)
```

- Show the elements of vector a that are equal to vector b, element-wise

```
a == b'
```

Matrix operations

- Show the elements of matrix A that are equal to 5

```
C = (A == 5)
```

- Show the elements of vector a that are equal to vector b, element-wise

```
a == b'
```

- Find elements of matrix B that are equal to 1

```
find(B == 1)
```

Matrix operations

- Show the elements of matrix A that are equal to 5

```
C = (A == 5)
```

- Show the elements of vector a that are equal to vector b, element-wise

```
a == b'
```

- Find elements of matrix B that are equal to 1

```
find(B == 1)
```

- Sort the numbers in b in ascending order

```
f = sort(b)
```

Matrix operations

- Show the elements of matrix A that are equal to 5

```
C = (A == 5)
```

- Show the elements of vector a that are equal to vector b, element-wise

```
a == b'
```

- Find elements of matrix B that are equal to 1

```
find(B == 1)
```

- Sort the numbers in b in ascending order

```
f = sort(b)
```

- Add matrices A and C, i.e., $F = A + C$

```
F = A + C;
```

Matrix operations

- Show the elements of matrix A that are equal to 5

```
C = (A == 5)
```

- Show the elements of vector a that are equal to vector b, element-wise

```
a == b'
```

- Find elements of matrix B that are equal to 1

```
find(B == 1)
```

- Sort the numbers in b in ascending order

```
f = sort(b)
```

- Add matrices A and C, i.e., $F = A + C$

```
F = A + C;
```

- Multiply the matrices A and B, i.e., $G = A * B$

```
G = A*B;
```

Matrix operations

- Multiply element-wise matrix G and the identity matrix I, i.e.,

$$G = \begin{bmatrix} 130 & 70 & 110 \\ 70 & 174 & 278 \\ 110 & 278 & 446 \end{bmatrix}, I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} \\ h_{3,1} & h_{3,2} & h_{3,3} & h_{3,4} \end{bmatrix} = \begin{bmatrix} 130 * 1 & 70 * 0 & 110 * 0 \\ 70 * 0 & 174 * 1 & 278 * 0 \\ 110 * 0 & 278 * 0 & 446 * 1 \end{bmatrix}$$

```
H = G.*I;
```

Matrix operations

- Multiply element-wise matrix G and the identity matrix I, i.e.,

$$G = \begin{bmatrix} 130 & 70 & 110 \\ 70 & 174 & 278 \\ 110 & 278 & 446 \end{bmatrix}, \; I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} \\ h_{3,1} & h_{3,2} & h_{3,3} & h_{3,4} \end{bmatrix} = \begin{bmatrix} 130*1 & 70*0 & 110*0 \\ 70*0 & 174*1 & 278*0 \\ 110*0 & 278*0 & 446*1 \end{bmatrix}$$

```
H = G.*I;
```

- Find the cube of the elements of matrix A

```
J = A.^3;
```

Loops

- Load the MEG data from meg_data.mat and compute the sample mean of the first 500 samples with a `for` loop: $\bar{y} = \frac{1}{500}\sum_{i=1}^{500} y_i$

Loops

- Load the MEG data from meg_data.mat and compute the sample mean of the first 500 samples with a `for` loop: $\bar{y} = \frac{1}{500} \sum_{i=1}^{500} y_i$

```
% load the meg data
load('meg_data.mat');
% Take the required values
y = back_average(1:500);
% Initialize a variable accumulate the sum
acum = 0;
for i=1:500
    % Accumulates the sum of the data
    acum = acum + y(i);
end
% Divide by number of samples
y_bar1 = acum / 500;
```

Loops

- Compute the sample mean again using a `while` loop

Loops

- Compute the sample mean again using a `while` loop

```
% Initialize a temporary variable
acum = 0;
counter = 0;
while counter < 500
    % Updates the counter
    counter = counter + 1;
    % Accumulates the sum of the data
    acum = acum + y(counter);
end
% Divide by number of samples
y_bar2 = acum / 500;
```

Some useful functions

- Simulate 500 independent samples from a uniform distribution: $u_i \sim U([0,1])$, $i = 1, 2, \ldots, 500$

Some useful functions

- Simulate 500 independent samples from a uniform distribution: $u_i \sim U([0,1])$, $i = 1, 2, \ldots, 500$

```
% Draw 500 samples from the uniform distibution
u = rand(1,500);
```

Some useful functions

- Simulate 500 independent samples from a uniform distribution: $u_i \sim U([0,1])$, $i = 1, 2, \ldots, 500$

```
% Draw 500 samples from the uniform distibution
u = rand(1,500);
```

- Simulate 500 independent samples from a standard Gaussian distribution: $x_i \sim N(0,1)$, $i = 1, 2, \ldots, 500$

Some useful functions

- Simulate 500 independent samples from a uniform distribution: $u_i \sim U([0,1])$, $i = 1, 2, \ldots, 500$

```
% Draw 500 samples from the uniform distibution
u = rand(1,500);
```

- Simulate 500 independent samples from a standard Gaussian distribution: $x_i \sim N(0,1)$, $i = 1, 2, \ldots, 500$

```
% Draw 500 samples from the Standard Gaussian
x = randn(1,500);
```

Some useful functions

- Simulate 500 independent samples from a uniform distribution: $u_i \sim U([0,1])$, $i = 1, 2, \ldots, 500$

```
% Draw 500 samples from the uniform distibution
u = rand(1,500);
```

- Simulate 500 independent samples from a standard Gaussian distribution: $x_i \sim N(0,1)$, $i = 1, 2, \ldots, 500$

```
% Draw 500 samples from the Standard Gaussian
x = randn(1,500);
```

- Simulate 500 independent samples from Bernoulli distribution with $p = 0.5$ (500 fair coin flips): $b_i \sim B(0.5)$, $i = 1, 2, \ldots, 500$

Some useful functions

- Simulate 500 independent samples from a uniform distribution: $u_i \sim U([0,1])$, $i = 1, 2, \ldots, 500$

```
% Draw 500 samples from the uniform distibution
u = rand(1,500);
```

- Simulate 500 independent samples from a standard Gaussian distribution: $x_i \sim N(0,1)$, $i = 1, 2, \ldots, 500$

```
% Draw 500 samples from the Standard Gaussian
x = randn(1,500);
```

- Simulate 500 independent samples from Bernoulli distribution with $p = 0.5$ (500 fair coin flips): $b_i \sim B(0.5)$, $i = 1, 2, \ldots, 500$

```
% Draw 500 samples from Bernoulli distribution
p = 0.5;
b = rand(1,500) > 0.5;
```

Some useful functions

- Computed the minimum, maximum, mean, standard deviation, and variance from simulated sample from the standard Gaussian distribution: x_i, $i = 1, 2, \ldots, 500$, where the sample variance is $\hat{\sigma}^2 = \frac{1}{500}\sum_{i=1}^{500}(x_i - \bar{x})^2$, and the sample standard deviation is $\hat{\sigma}$

Some useful functions

- Computed the minimum, maximum, mean, standard deviation, and variance from simulated sample from the standard Gaussian distribution: $x_i,\ i = 1, 2, \ldots, 500$, where the sample variance is $\hat{\sigma}^2 = \frac{1}{500}\sum_{i=1}^{500}(x_i - \bar{x})^2$, and the sample standard deviation is $\hat{\sigma}$

```
% The minimum
x_min = min(x);

% The maximum
x_max = max(x);

% The sample mean
x_bar = mean(x);

% The sample variance
sigma2_hat = var(x);

% The sample standard deviation
sigma_hat = sigma2_hat^(1/2);
```

Functions for displaying results

- Load and plot the MEG data

Functions for displaying results

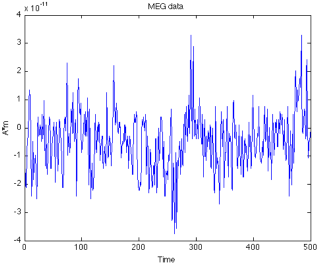
- Load and plot the MEG data

```
load('meg_data.mat');
% Take the required values
y = back_average(1:500);
figure, plot(y)
title('MEG data')
xlabel('Time'), ylabel('A*m')
```

Functions for displaying results

- Load and plot the MEG data

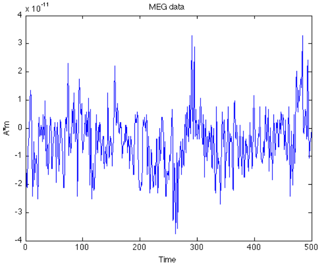
```
load('meg_data.mat');
% Take the required values
y = back_average(1:500);
figure, plot(y)
title('MEG data')
xlabel('Time'), ylabel('A*m')
```



Functions for displaying results

- Load and plot the MEG data

```
load('meg_data.mat');
% Take the required values
y = back_average(1:500);
figure, plot(y)
title('MEG data')
xlabel('Time'), ylabel('A*m')
```

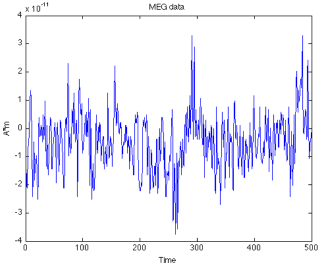


- Make a histogram of the simulated sample from the standard Gaussian distribution $x_i,\ i = 1, 2, \ldots, 500$

Functions for displaying results

- Load and plot the MEG data

```
load('meg_data.mat');
% Take the required values
y = back_average(1:500);
figure, plot(y)
title('MEG data')
xlabel('Time'), ylabel('A*m')
```



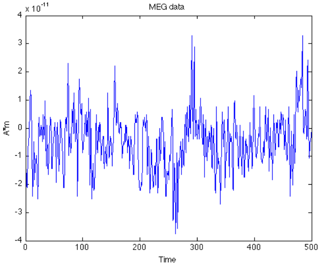
- Make a histogram of the simulated sample from the standard Gaussian distribution $x_i,\ i = 1, 2, \ldots, 500$

```
%select the number of bins
m = 25;
figure, hist(x,m)
title('Histogram')
xlabel('x'), ylabel('Count')
```

Functions for displaying results

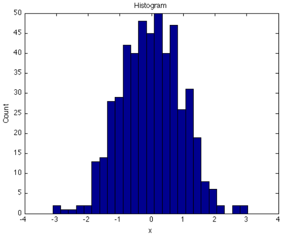
- Load and plot the MEG data

```
load('meg_data.mat');
% Take the required values
y = back_average(1:500);
figure, plot(y)
title('MEG data')
xlabel('Time'), ylabel('A*m')
```



- Make a histogram of the simulated sample from the standard Gaussian distribution $x_i, i = 1, 2, \ldots, 500$

```
%select the number of bins
m = 25;
figure, hist(x,m)
title('Histogram')
xlabel('x'), ylabel('Count')
```



Translating a problem to an algorithm to a code

- Simulate 500 independent samples from the exponential distribution with a “fair” coin

Translating a problem to an algorithm to a code

- Simulate 500 independent samples from the exponential distribution with a “fair” coin
  - With the “fair” coin we can obtain a samples from the Bernoulli distribution with parameter $p = 0.5$. Recall that the pdf of the Bernoulli is given by:
    $f_{ber}(b) = p^b(1-p)^{(1-b)}$, where $b \in \{0, 1\}$

Translating a problem to an algorithm to a code

- Simulate 500 independent samples from the exponential distribution with a "fair" coin
  - With the "fair" coin we can obtain a samples from the Bernoulli distribution with parameter $p = 0.5$. Recall that the pdf of the Bernoulli is given by: $f_{ber}(b) = p^b(1-p)^{(1-b)}$, where $b \in \{0, 1\}$
  - Recall that if $t_i \sim Exp(\lambda)$, $i = 1, 2, \ldots, 500$, then its pdf is given by: $f_{exp}(t) = \lambda e^{-\lambda t}$, where $t > 0$

Translating a problem to an algorithm to a code

- Simulate 500 independent samples from the exponential distribution with a "fair" coin
  - With the "fair" coin we can obtain a samples from the Bernoulli distribution with parameter $p = 0.5$. Recall that the pdf of the Bernoulli is given by:
    $f_{ber}(b) = p^b(1-p)^{(1-b)}$, where $b \in \{0, 1\}$
  - Recall that if $t_i \sim Exp(\lambda)$, $i = 1, 2, \ldots, 500$, then its pdf is given by:
    $f_{exp}(t) = \lambda e^{-\lambda t}$, where $t > 0$
- Wow! This sound impossible!

Translating a problem to an algorithm to a code

- Simulate 500 independent samples from the exponential distribution with a "fair" coin
  - With the "fair" coin we can obtain a samples from the Bernoulli distribution with parameter $p = 0.5$. Recall that the pdf of the Bernoulli is given by:
    $f_{ber}(b) = p^b(1-p)^{(1-b)}$, where $b \in \{0, 1\}$
  - Recall that if $t_i \sim Exp(\lambda)$, $i = 1, 2, \ldots, 500$, then its pdf is given by:
    $f_{exp}(t) = \lambda e^{-\lambda t}$, where $t > 0$
- Wow! This sound impossible!
- It is possible if we could generate a sample from the uniform distribution using a coin
- And from the uniform sample simulate a new sample of the exponential distribution using the Inverse Transform method!

Translating a problem to an algorithm to a code

- Simulate 500 independent samples from the exponential distribution with a “fair” coin
  - With the “fair” coin we can obtain a samples from the Bernoulli distribution with parameter $p = 0.5$. Recall that the pdf of the Bernoulli is given by:
    $f_{ber}(b) = p^b(1-p)^{(1-b)}$, where $b \in \{0, 1\}$
  - Recall that if $t_i \sim Exp(\lambda)$, $i = 1, 2, \ldots, 500$, then its pdf is given by:
    $f_{exp}(t) = \lambda e^{-\lambda t}$, where $t > 0$
- Wow! This sound impossible!
- It is possible if we could generate a sample from the uniform distribution using a coin
- And from the uniform sample simulate a new sample of the exponential distribution using the Inverse Transform method!
  - Find an algorithm
  - Write the code

From the problem to an algorithm

- To find the algorithm we should note that a number u between 0 and 1 can be represented with the binary expansion $0.b_1b_2b_3\ldots$, i.e:

$$u = \sum_{i=1}^{\infty} b_i/2^i, \text{where } b_i \in \{0, 1\}$$

From the problem to an algorithm

- To find the algorithm we should note that a number u between 0 and 1 can be represented with the binary expansion $0.b_1b_2b_3\ldots$, i.e:

$$u = \sum_{i=1}^{\infty} b_i/2^i, \text{where } b_i \in \{0,1\}$$

- For example:

From the problem to an algorithm

- To find the algorithm we should note that a number u between 0 and 1 can be represented with the binary expansion $0.b_1 b_2 b_3 \ldots$, i.e:

$$u = \sum_{i=1}^{\infty} b_i/2^i, \text{where } b_i \in \{0, 1\}$$

- For example:
  - If $u = 0.75$ then $b_1 = 1$, $b_2 = 1$, $b_j = 0$, $j = 3, 4, \ldots$, since $u = 1/2 + 1/2^2 + 0/2^3 + \cdots$

From the problem to an algorithm

- To find the algorithm we should note that a number u between 0 and 1 can be represented with the binary expansion $0.b_1 b_2 b_3 \ldots$, i.e:

$$u = \sum_{i=1}^{\infty} b_i / 2^i, \text{where } b_i \in \{0, 1\}$$

- For example:
  - If $u = 0.75$ then $b_1 = 1$, $b_2 = 1$, $b_j = 0$, $j = 3, 4, \ldots$, since $u = 1/2 + 1/2^2 + 0/2^3 + \cdots$
  - If $u = 0.958$ then $0.b_1 b_2 b_3 \ldots = 0.1111010101\bar{0}$

From the problem to an algorithm

- To find the algorithm we should note that a number u between 0 and 1 can be represented with the binary expansion $0.b_1 b_2 b_3 \ldots$, i.e:

$$u = \sum_{i=1}^{\infty} b_i/2^i, \text{where } b_i \in \{0, 1\}$$

- For example:
  - If $u = 0.75$ then $b_1 = 1$, $b_2 = 1$, $b_j = 0$, $j = 3, 4, \ldots$, since $u = 1/2 + 1/2^2 + 0/2^3 + \cdots$
  - If $u = 0.958$ then $0.b_1 b_2 b_3 \ldots = 0.1111010101\bar{0}$
  - If $u = 0.3288$ then $0.b_1 b_2 b_3 \ldots = 0.0101010000101001001\bar{0}$

From the problem to an algorithm

- The million dollar question: If the binary coefficients b_i in expansion $u = \sum_{i=1}^{\infty} b_i/2^i$ come from a Bernoulli distribution with $p = 0.5$ (the "fair" coin), what is the distribution of u?

From the problem to an algorithm

- The million dollar question: If the binary coefficients b_i in expansion $u = \sum_{i=1}^{\infty} b_i/2^i$ come from a Bernoulli distribution with $p = 0.5$ (the "fair" coin), what is the distribution of u?
- The answer is: u is a uniform random variable. But this is difficult to show!

From the problem to an algorithm

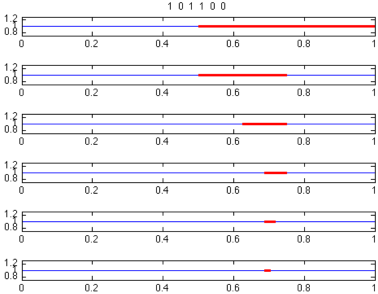
- The million dollar question: If the binary coefficients b_i in expansion $u = \sum_{i=1}^{\infty} b_i/2^i$ come from a Bernoulli distribution with $p = 0.5$ (the "fair" coin), what is the distribution of u?
- The answer is: u is a uniform random variable. But this is difficult to show!
- We will use a heuristic argument to convince our selves

From the problem to an algorithm

- The million dollar question: If the binary coefficients b_i in expansion $u = \sum_{i=1}^{\infty} b_i/2^i$ come from a Bernoulli distribution with $p = 0.5$ (the "fair" coin), what is the distribution of u?
- The answer is: u is a uniform random variable. But this is difficult to show!
- We will use a heuristic argument to convince our selves
- If $b_1 = 1 \rightarrow u >= 0.5$ and if $b_1 = 0 \rightarrow u < 0.5$

From the problem to an algorithm

- The million dollar question: If the binary coefficients b_i in expansion $u = \sum_{i=1}^{\infty} b_i/2^i$ come from a Bernoulli distribution with $p = 0.5$ (the "fair" coin), what is the distribution of u?
- The answer is: u is a uniform random variable. But this is difficult to show!
- We will use a heuristic argument to convince our selves
- If $b_1 = 1 \rightarrow u >= 0.5$ and if $b_1 = 0 \rightarrow u < 0.5$
- More generally, with a figure we see that:



From the algorithm to the code

- Simulate a sample of size 1000 from the uniform distribution using samples from the Bernoulli distribution with parameter $p = 0.5$:

$$u_j = \sum_{i=1}^{500} b_{i,j}/2^i, \text{ where } j = 1, 2, \ldots, 1000, \ i = 1, 2, \ldots, 500$$

and $b_{i,j}$ are iid Bernoulli with parameter p

From the algorithm to the code

- Simulate a sample of size 1000 from the uniform distribution using samples from the Bernoulli distribution with parameter $p = 0.5$:

$$u_j = \sum_{i=1}^{500} b_{i,j}/2^i, \text{ where } j = 1, 2, \ldots, 1000, \ i = 1, 2, \ldots, 500$$

and $b_{i,j}$ are iid Bernoulli with parameter p

```
% Sample size
n = 1000;
% Compute the denominators expansion
m = 500;
d = 1./2.^[1:m];
% For loop to obtain the 500 uniform rvs
for i =1:n
    % Simulate 100 Bernoulli rvs (b_{i,j})
    b = rand(1,m)>0.5;
    % Compute the summation
    u(i) = sum(b.*d);
end
```

From the algorithm to the code

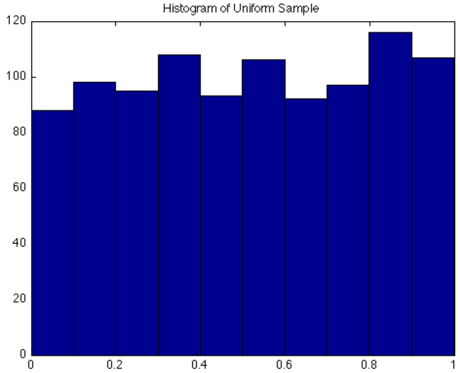
- Make a histogram of the obtained uniform sample

```
figure, hist(u)
title('Histogram of Uniform Sample')
```

From the algorithm to the code

- Make a histogram of the obtained uniform sample

```
figure, hist(u)
title('Histogram of Uniform Sample')
```



From the problem to an algorithm

- Generate a sample of size 1000 of the exponential distribution from a sample from the uniform distribution using the Inverse Transform method (find F_{exp}^{-1})

From the problem to an algorithm

- Generate a sample of size 1000 of the exponential distribution from a sample from the uniform distribution using the Inverse Transform method (find F_{exp}^{-1})
- Recall that $f_{exp}(t) = \lambda e^{-\lambda t}$, where $t > 0$

From the problem to an algorithm

- Generate a sample of size 1000 of the exponential distribution from a sample from the uniform distribution using the Inverse Transform method (find F_{exp}^{-1})
- Recall that $f_{exp}(t) = \lambda e^{-\lambda t}$, where $t > 0$
- The cdf is given by:

$$F_{exp}(t) = \int_0^t \lambda e^{-\lambda \tau} d\tau$$

Make change in variable $w = -\lambda\tau \rightarrow d\tau = -dw/\lambda$

$$F_{exp}(t) = -\int_0^{-\lambda t} e^w dw = 1 - e^{-\lambda t}$$

From the problem to an algorithm

- Generate a sample of size 1000 of the exponential distribution from a sample from the uniform distribution using the Inverse Transform method (find F_{exp}^{-1})
- Recall that $f_{exp}(t) = \lambda e^{-\lambda t}$, where $t > 0$
- The cdf is given by:

$$F_{exp}(t) = \int_0^t \lambda e^{-\lambda \tau} d\tau$$

Make change in variable $w = -\lambda\tau \to d\tau = -dw/\lambda$

$$F_{exp}(t) = -\int_0^{-\lambda t} e^w dw = 1 - e^{-\lambda t}$$

- Now we can obtain F_{exp}^{-1} as:

$$F_{exp}^{-1}(u) = -\frac{\log(1-u)}{\lambda}$$

From the algorithm to the code

- Generate a sample of size 1000 of the exponential distribution with parameter $\lambda = 3$ using a sample from the uniform distribution (u_j) using the Inverse Transform method:

$$t_j = -\frac{\log(1 - u_j)}{3}, \text{ where } j = 1, 2, \ldots, 1000$$

From the algorithm to the code

- Generate a sample of size 1000 of the exponential distribution with parameter $\lambda = 3$ using a sample from the uniform distribution (u_j) using the Inverse Transform method:

$$t_j = -\frac{\log(1-u_j)}{3}, \text{ where } j = 1, 2, \ldots, 1000$$

```
t = -log(1-u)/3;
```

From the algorithm to the code

- Generate a sample of size 1000 of the exponential distribution with parameter $\lambda = 3$ using a sample from the uniform distribution (u_j) using the Inverse Transform method:

$$t_j = -\frac{\log(1-u_j)}{3}, \text{ where } j = 1, 2, \ldots, 1000$$

```
t = -log(1-u)/3;
```

- Make a histogram of the sample simulated from the exponential distribution

```
figure, hist(t)
title('Histogram of Exponential Sample')
```

Did it WORK???

- Raise you hand if you think it did work

Did it WORK???

- Raise you hand if you think it did work