User-defined functions

Importance sampling

黃世強 (Sai-Keung Wong)

National Chiao Tung University, Taiwan

while-loop

while loop: repeat a process when condition is true

Syntax

```
while expression
   statements
end
```

Example

Write a program to ask to input a positive number.

Example

Enter a series of numbers. The input process is finished if -1 is input. Report their average and standard deviation. Note -1 is not counted.

Example

Enter a series of numbers. The input process is finished if -1 is input. Report their average and standard deviation. Note -1 is not counted.

```
function [a, d] = myFunc( )
  x = [ ]
  while true
    x1 = input('Input a number:');
    if x1 == -1
      break;
    end
   x = [x x1];
  end
  a = average(x);   d = std( x );
end
```

User-defined Functions

```
function output = functionName( argument list)
   statements
end
```

Example:

```
function [y1,...,yN] = myfun(x1,...,xM)
```

User-defined Functions

```
function output = functionName( argument list)
   statements
end
```

- Each function should be created in its own file.

- The file name should be the same as the function name (the extension .m is not counted).

Example
User-defined Functions

```
function output = functionName( argument list)
   statements
end
```

Define a function which computes the sum of a series of numbers, starting from 1, 2, up to n.

Example
User-defined Functions

function output = functionName(argument list)

statements

end

Define a function which computes the sum of a series of numbers, starting from 1, 2, up to n.

```
function output = m_matlab_simpleSum(n)
    sum = 0;
    for i = [1:n] sum = sum + i; end
    output = sum;
end
```

Example
User-defined Functions

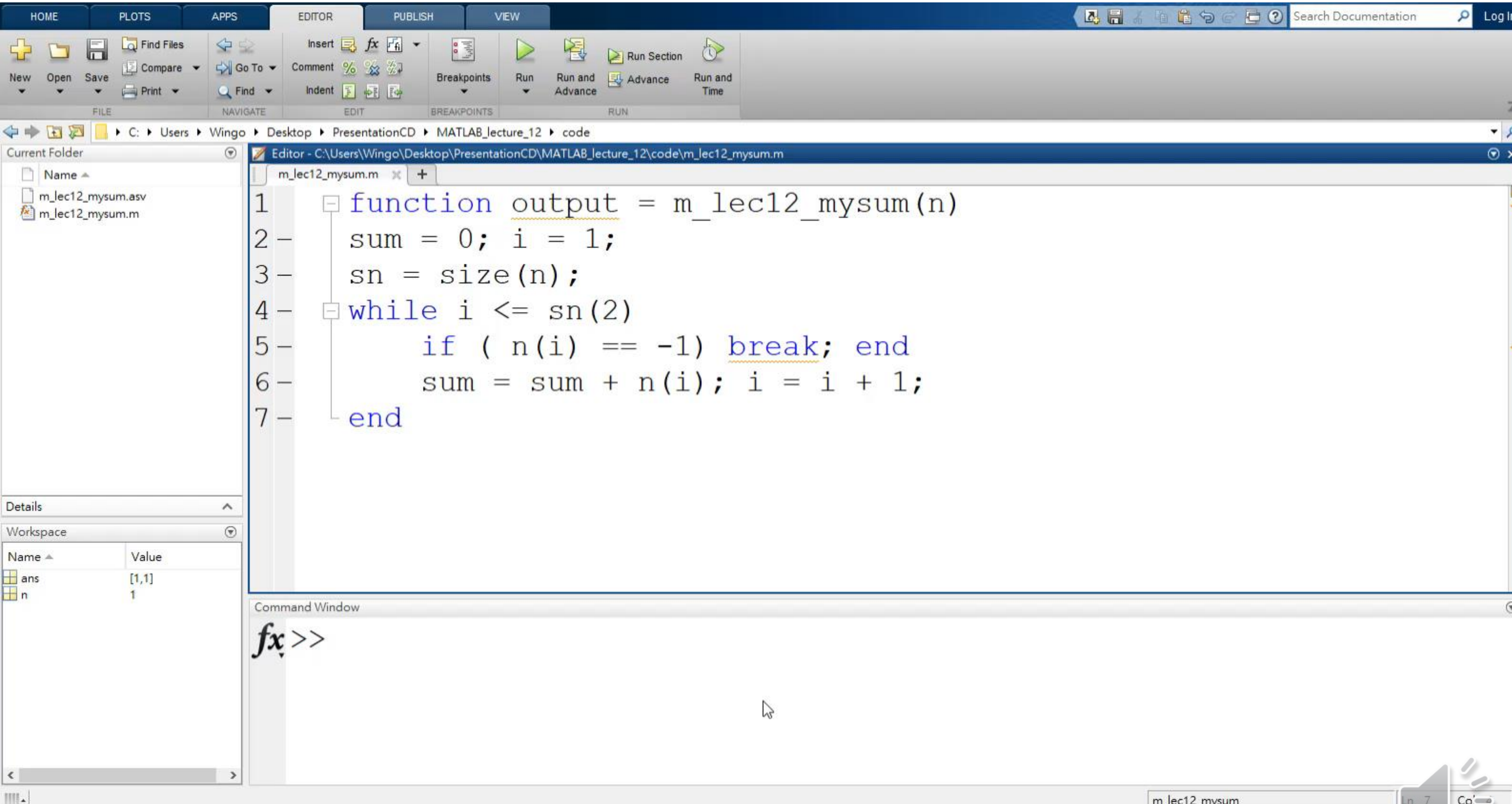
invoke the function:

```
v = m_matlab_simpleSum(10)
```

Define a function which computes the sum of a series of numbers, starting from 1, 2, up to n.

```
function output = m_matlab_simpleSum(n)
    sum = 0;
    for i = [1:n] sum = sum + i; end
    output = sum;
end
```

Demo
`m_lect12_mysum`



Example

Compute the average of random numbers generated uniformly inside the interval [r1,r2]. Assume the number of random numbers is n.

Probability density function p(x) = 1/(r2-r1),
for all x in [r1,r2].

Example

Compute the average of random numbers generated uniformly inside the interval [r1,r2]. Assume the number of random numbers is n.

Probability density function $p(x) = 1/(r2-r1)$,
for all x in [r1,r2].

Example

Compute the average of random numbers generated uniformly inside the interval [r1,r2]. Assume the number of random numbers is n.

Probability density function $p(x) = 1/(r2-r1)$, for all x in [r1,r2].

```
n = 1000;
i = 1;
while i <= n
   r = rand(1);
   a(i) = r;
   i = i + 1;
end
amean = mean(a);
```

Example

Compute the average of random numbers generated uniformly inside the interval [r1,r2]. Assume the number of random numbers is n.

Probability density function p(x) = 1/(r2-r1), for all x in [r1,r2].

```
n = 1000;
i = 1;
while i <= n
   r = rand(1);
   a(i) = r;
   i = i + 1;
end
amean = mean(a); %
```
$\frac{1}{n}\sum_{i=1}^{n} a_i$

$$p(x) = \frac{1}{r2-r1}$$

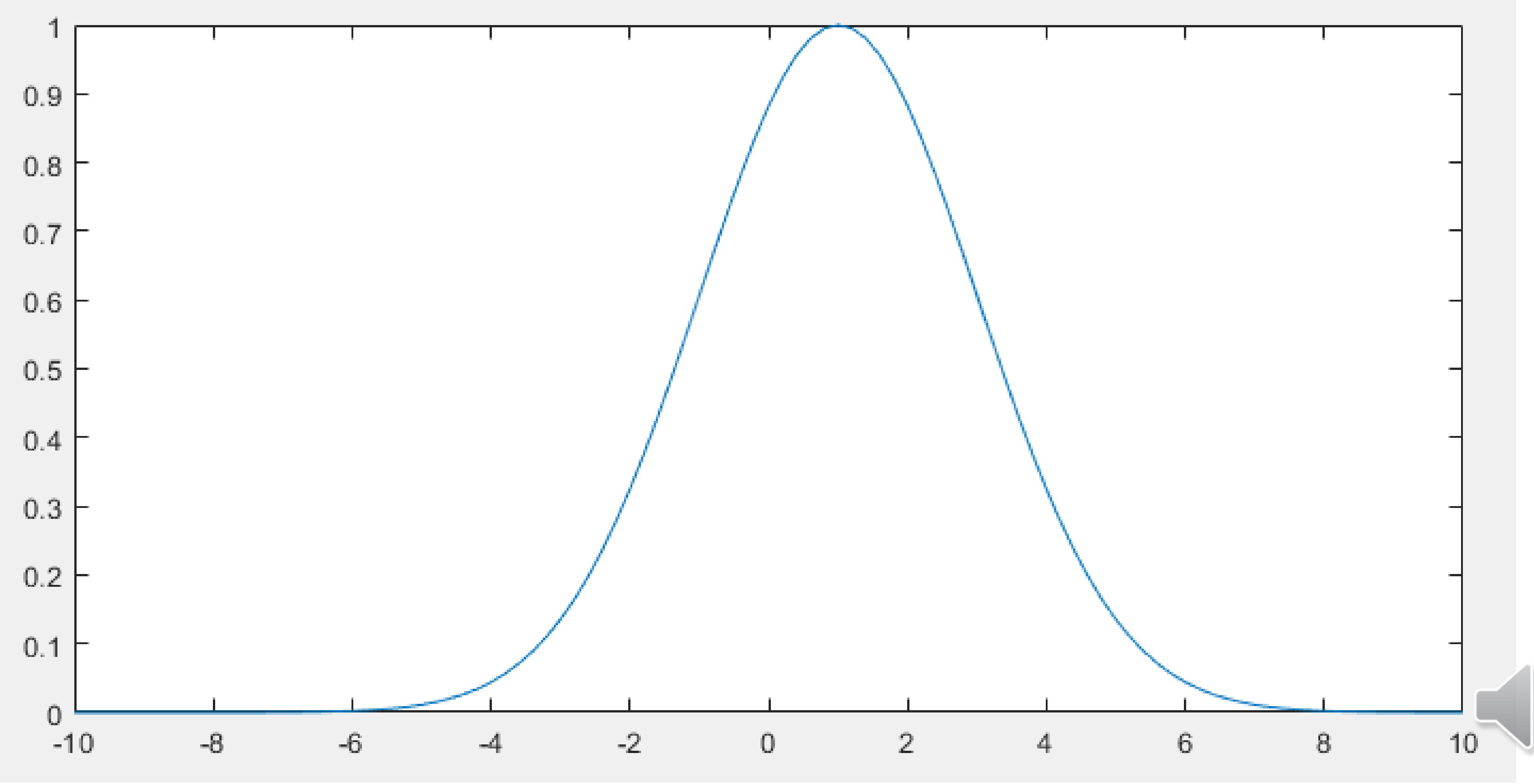
$$\int_{r1}^{r2} \frac{1}{r2-r1}\,dx \quad = 1$$

$$mean = E(x) = \int_{r1}^{r2} x\frac{1}{r2-r1}\,dx$$

Example

Compute the average of random numbers generated by normal distribution inside the interval [r1,r2].

Normal distribution $$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

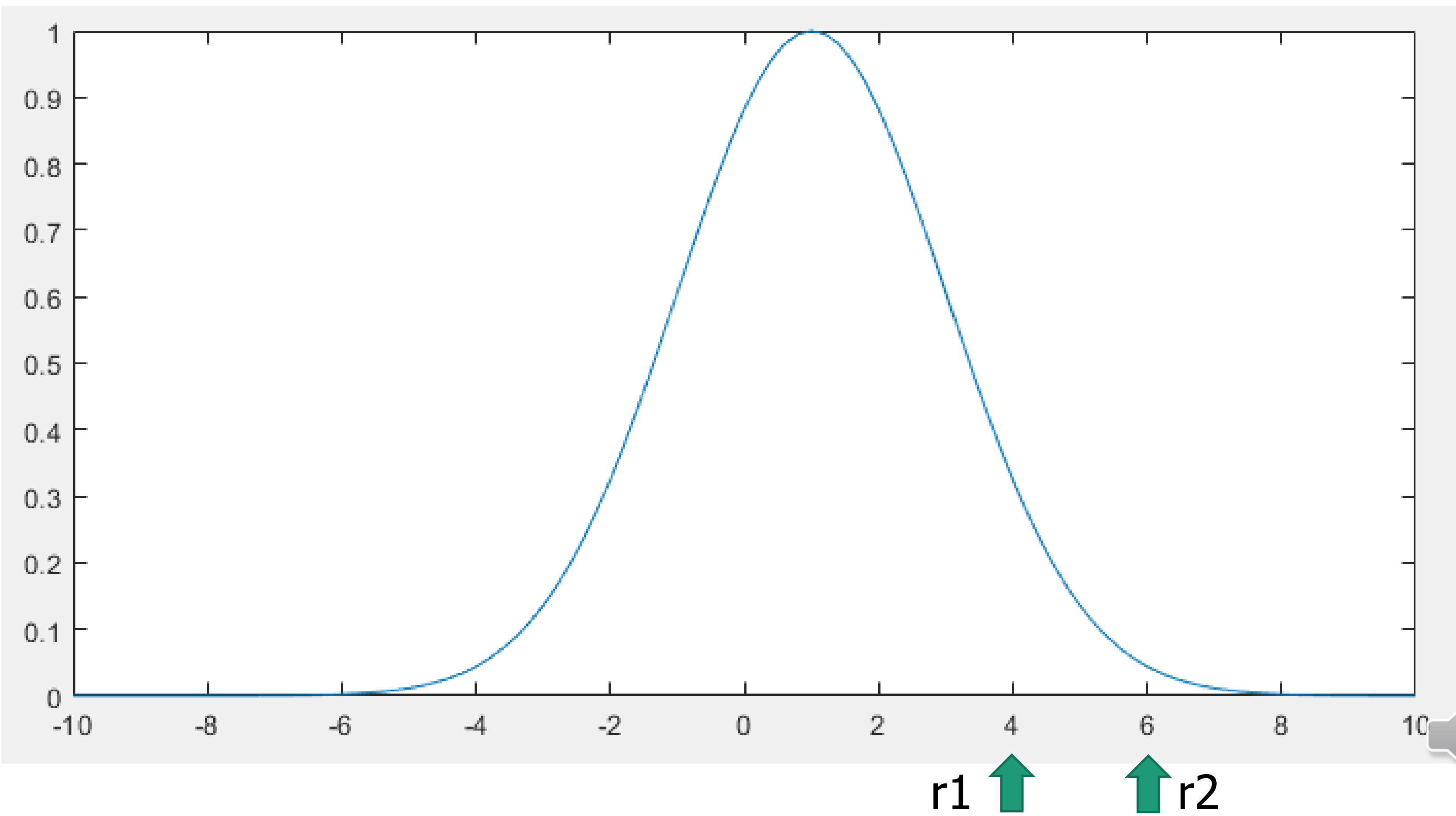


Example

Compute the average of random numbers generated by the normal distribution inside the interval [r1,r2].

Normal distribution: may not be a pdf!

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$



Example

Compute the average of random numbers generated by the normal distribution inside the interval [r1,r2].

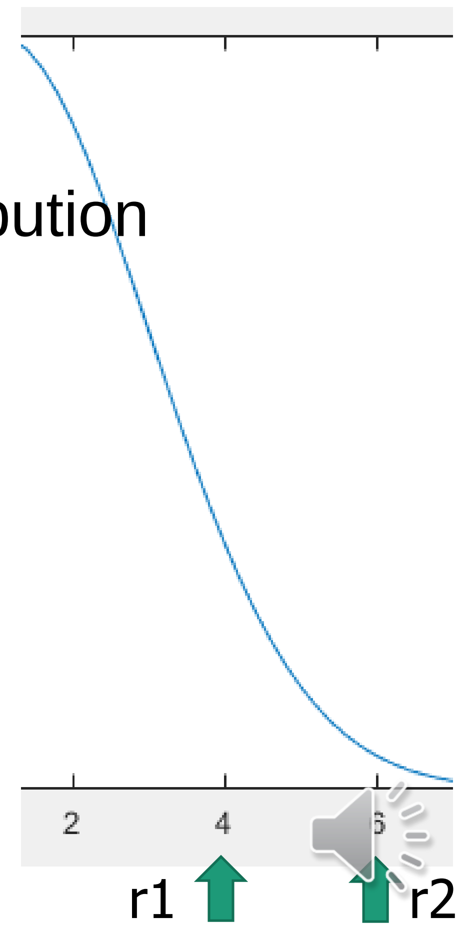
$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

repeat

r = generate a random number by normal distribution

if (r inside [r1,r2]) a $\leftarrow$ [a,r] %add r to a

until termination condition is satisfied



Example

Compute the average of random numbers generated by the normal distribution inside the interval [r1,r2].

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

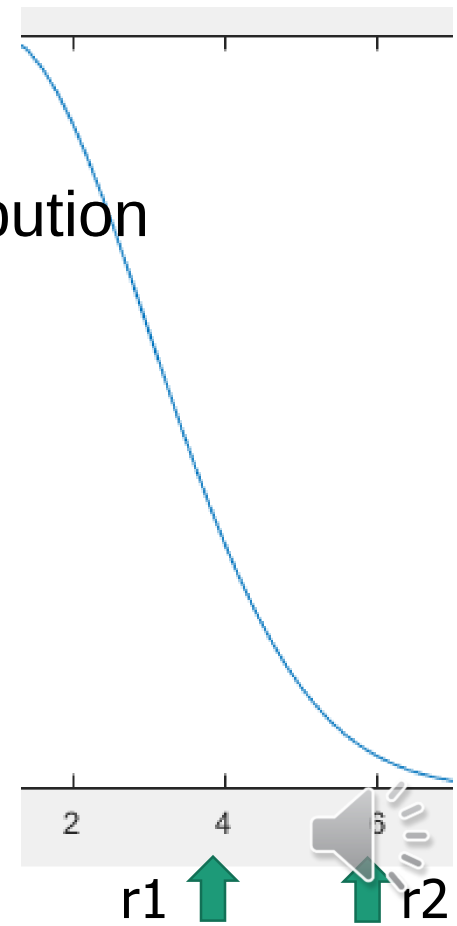
repeat

  r = generate a random number by normal distribution

  if (r inside [r1,r2]) a $\leftarrow$ [a,r]   %add r to a

until termination condition is satisfied

Question: generate n numbers.

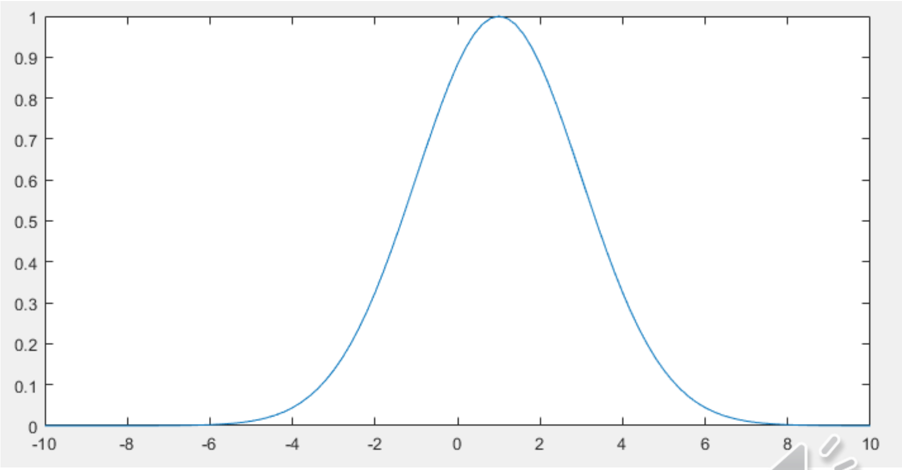


Example

Compute the average of random numbers generated by the normal distribution inside the interval [r1,r2].

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

```
n = 10000;
i = 0; j = 1; r1 = 4; r2 = 6;
while (j<=n)
    c = 1; sigma = 2;
    r  = normrnd( c, sigma );
    if (r >= r1 && r <= r2)
        a(j) = r;
        j = j +1;
    end
end
mean(a)
```



mean = 4.6997. What is the problem?

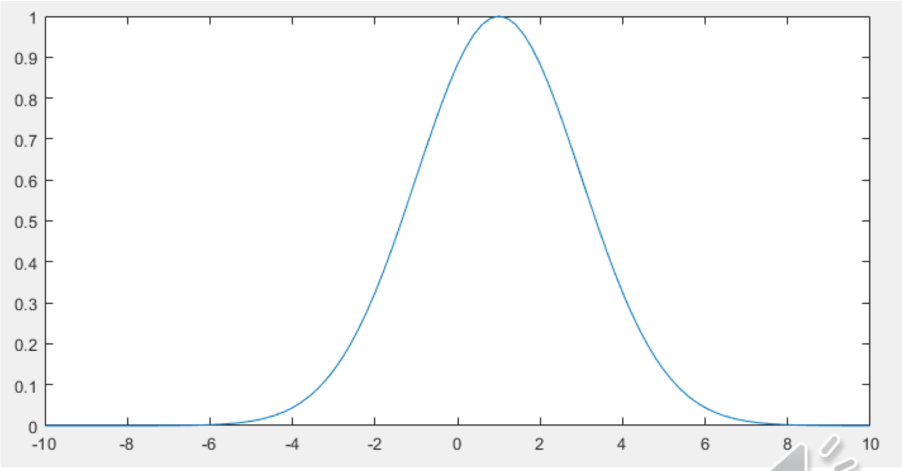
Example

Compute the average of random numbers generated by the normal distribution inside the interval [r1,r2].

```
n = 10000;
i = 0; j = 1; r1 = 4; r2 = 6;
while (j<=n)
    c = 1; sigma = 2;
    r  = normrnd( c, sigma );
    if (r >= r1 && r <= r2)
        a(j) = r;
        j = j +1;
    end
end
mean(a)
```

Note: the total area underneath the curve is: 5.0132

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$



mean = 4.6997. What is the problem? Generate a lot of redundant numbers !

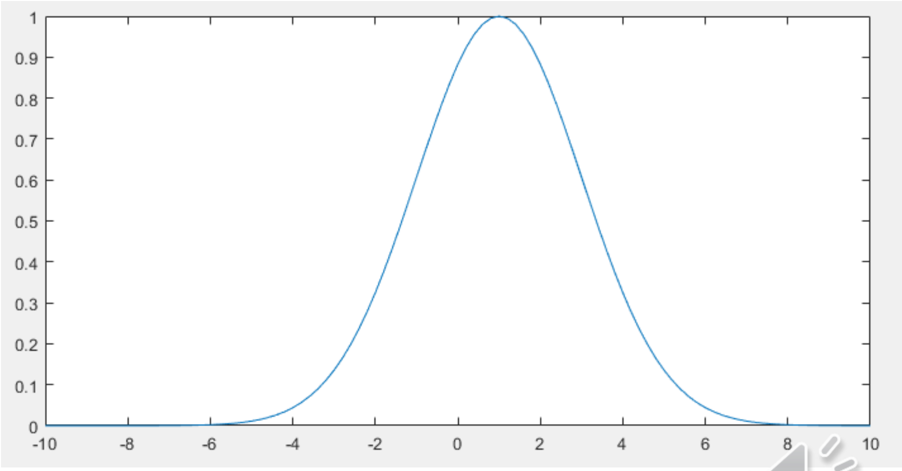
Example

Compute the average of random numbers generated by the normal distribution inside the interval [r1,r2].

```
n = 10000;
i = 0; j = 1; r1 = 4; r2 = 6;
while (j<=n)
    c = 1; sigma = 2;
    r  = normrnd( c, sigma );
    if (r >= r1 && r <= r2)
        a(j) = r;
        j = j +1;
    end
end
mean(a)
```

Note: the total area underneath the curve is: 5.0132

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$



mean = 4.6997. What is the problem? Generate a lot of redundant numbers !

Example

Compute the average of random numbers generated by the normal distribution inside the interval [r1,r2].

```
n = 10000; count = 0;
i = 0; j = 1; r1 = 4; r2 = 6;
while (j<=n)
    count = count + 1;
    c = 1; sigma = 2;
    r  = normrnd( c, sigma );
    if (r >= r1 && r <= r2)
        a(j) = r;
        j = j +1;
    end
end
mean(a)
```

count: the total number of generated numbers.

count = 167590 >> 10000
Too many!

mean = 4.6997

count = 167590

Example

Compute the average of random numbers generated by the normal distribution inside the interval [r1,r2].

```
n = 10000; count = 0;
i = 0; j = 1; r1 = 4; r2 = 6;
while (j<=n)
    count = count + 1;
    c = 1; sigma = 2;
    r  = normrnd( c, sigma );
    if (r >= r1 && r <= r2)
        a(j) = r;
        j = j +1;
    end
end
mean(a)
```

count: the total number of generated numbers.

count = 167590 >> 10000
Too many!

mean = 4.6997

count = 167590

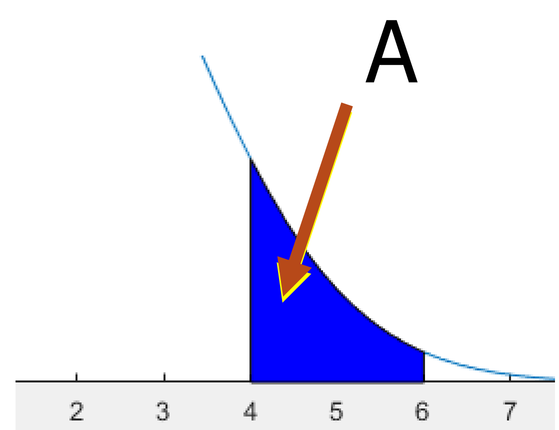
Example
Importance sampling

Compute the average of random numbers generated by the normal distribution (1,2) inside the interval [r1,r2].

Main idea: Generate samples that are effective in computing the main value.

Use another sampling method

to compute the target



$$\mu = \int_{\mathcal{D}} x \; p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{D}} \frac{x \;\; p(\boldsymbol{x})}{q(\boldsymbol{x})}\, q(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{E}_q\left(\frac{X \;\; p(\boldsymbol{X})}{q(\boldsymbol{X})} \right)$$

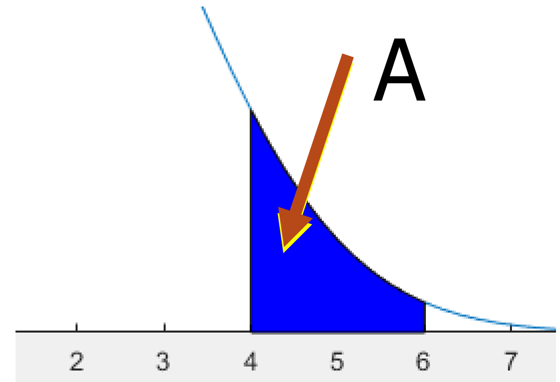
Example
Importance sampling

Compute the average of random numbers generated by the normal distribution (1,2) inside the interval [r1,r2].

What is the target probability density function, p?

$$\mu = \int_{\mathcal{D}} f(\boldsymbol{x}) p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

f(x) = x



$$\mu = \int_{\mathcal{D}} x \; p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

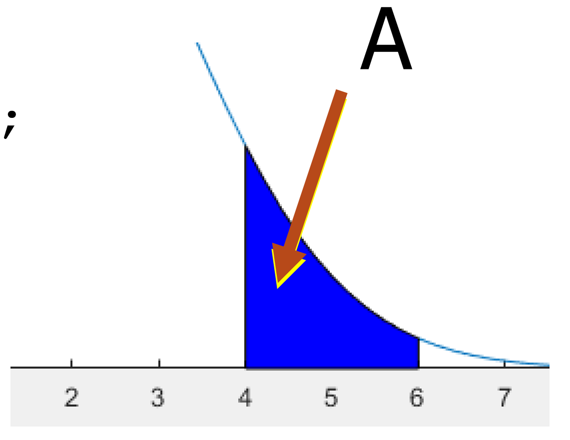
$$p\left(x; \sigma, c\right) = e^{\frac{-(x-c)^2}{2\sigma^2}} \Big/ A$$

Example
Importance sampling

Compute the average of random numbers generated by the normal distribution (1,2) inside the interval [r1,r2].

Use *int* to compute the area A.

```
c = 1, sigma = 2; r1 = 4; r2 = 6;
y = sym('2.718281828^(-(x-1)^2/8)');
A = int(y,’x’, 4, 6);
```



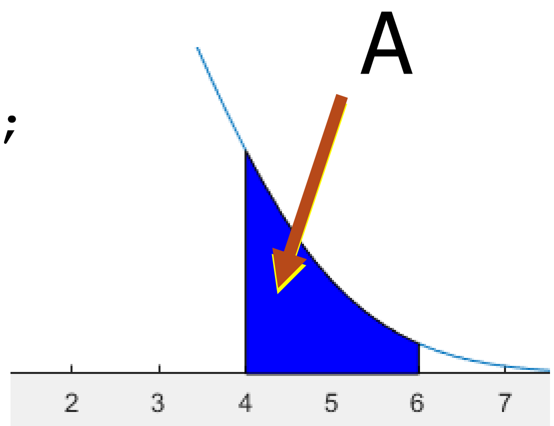
$$\mu = \int_{\mathcal{D}} x \; p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{D}} \frac{x \; p(\boldsymbol{x})}{q(\boldsymbol{x})}\, q(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{E}_q\left(\frac{X \; p(\boldsymbol{X})}{q(\boldsymbol{X})}\right)$$

Example
Importance sampling

Compute the average of random numbers generated by the normal distribution (1,2) inside the interval [r1,r2].

Use *int* to compute the area A.

```
c = 1, sigma = 2; r1 = 4; r2 = 6;
y = sym('2.718281828^(-(x-1)^2/8)');
A = int(y,'x', 4, 6);
```



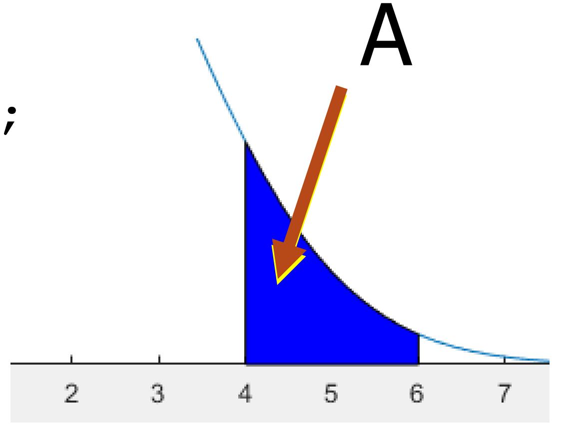
$$\mu = \int_{\mathcal{D}} \mathrm{x}\ p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{D}} \frac{\mathrm{x}\ p(\boldsymbol{x})}{q(\boldsymbol{x})}\, q(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{E}_q\left(\frac{\mathrm{X}\ \ p(\boldsymbol{X})}{q(\boldsymbol{X})}\right)$$

Importance sampling (Brief)

Compute the average of random numbers generated by the normal distribution (1,2) inside the interval [r1,r2].

Use *int* to compute the area A.

```
c = 1, sigma = 2; r1 = 4; r2 = 6;
y = sym('2.718281828^(-(x-1)^2/8)');
A = int(y,’x’, 4, 6);
```

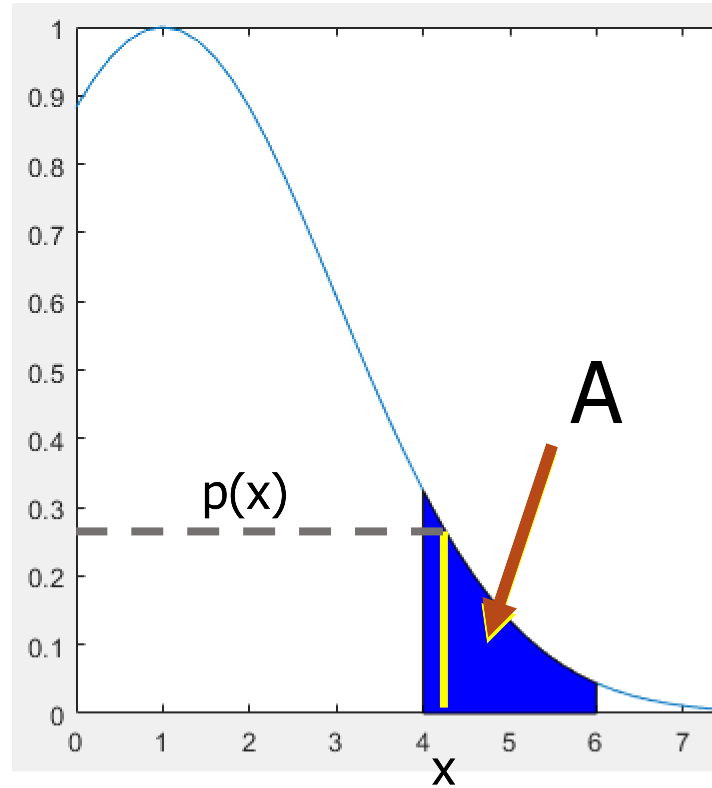


$$\mu = \int_{\mathcal{D}} x \; p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{D}} \frac{x \; p(\boldsymbol{x})}{q(\boldsymbol{x})}\, q(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{E}_q\left(\frac{X \; p(\boldsymbol{X})}{q(\boldsymbol{X})}\right)$$

Importance sampling

```
Use the uniform sampling method to generate a number
x inside the interval [r1,r2]. Then compute p(x).
```

```
%Importance sampling
n = 10000; j = 1; r1 = 4; r2 = 6;
while (j<=n)
    x = r1 + (r2-r1)*rand(1);
    c = 1; sigma = 2;
    p  = gaussmf(x, [sigma, c])/A;
    r = x*p*(r2-r1);
    a(j) = r;
    j = j +1;
end
mean(a)
```



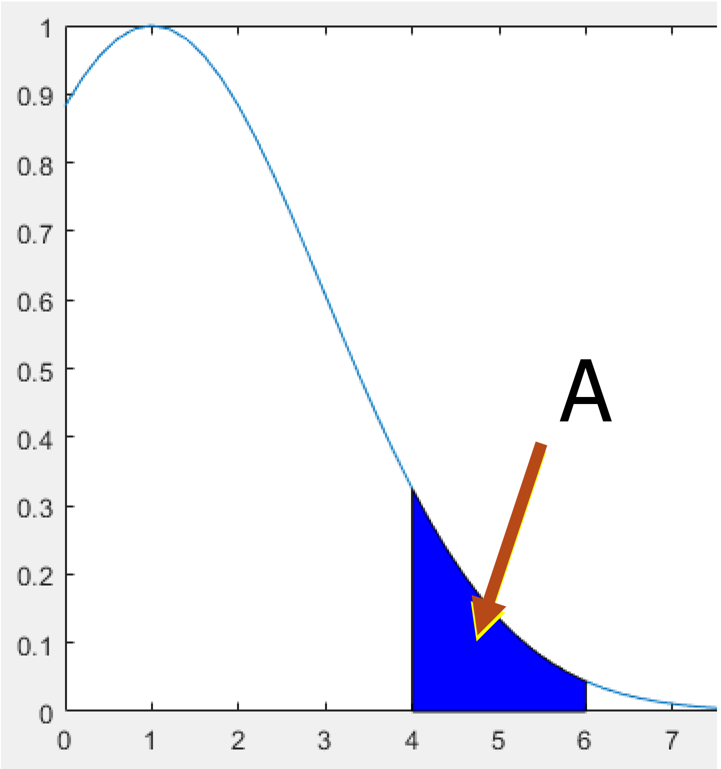
q(x) = 1/(r2-r1)

$$\mu = \int_{\mathcal{D}} \text{x} \;\; p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{D}} \frac{\text{x} \;\; p(\boldsymbol{x})}{q(\boldsymbol{x})}\, q(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{E}_q\left(\frac{\text{X} \;\; p(\boldsymbol{X})}{q(\boldsymbol{X})}\right)$$

Importance sampling

Use the uniform sampling method to generate a number x inside the interval [r1,r2]. Then compute p(x).

```
%Importance sampling
n = 10000; j = 1; r1 = 4; r2 = 6;
while (j<=n)
    x = r1 + (r2-r1)*rand(1);
    c = 1; sigma = 2;
    p  = gaussmf(x, [sigma, c])/A;
    r = x*p*(r2-r1);
    a(j) = r;
    j = j +1;
end
mean(a)
```



q(x) = 1/(r2-r1)

$$\mu = \int_{\mathcal{D}} \text{x} \; p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{D}} \frac{\text{x} \; p(\boldsymbol{x})}{q(\boldsymbol{x})} q(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{E}_q\left(\frac{\text{X} \; p(\boldsymbol{X})}{q(\boldsymbol{X})}\right)$$

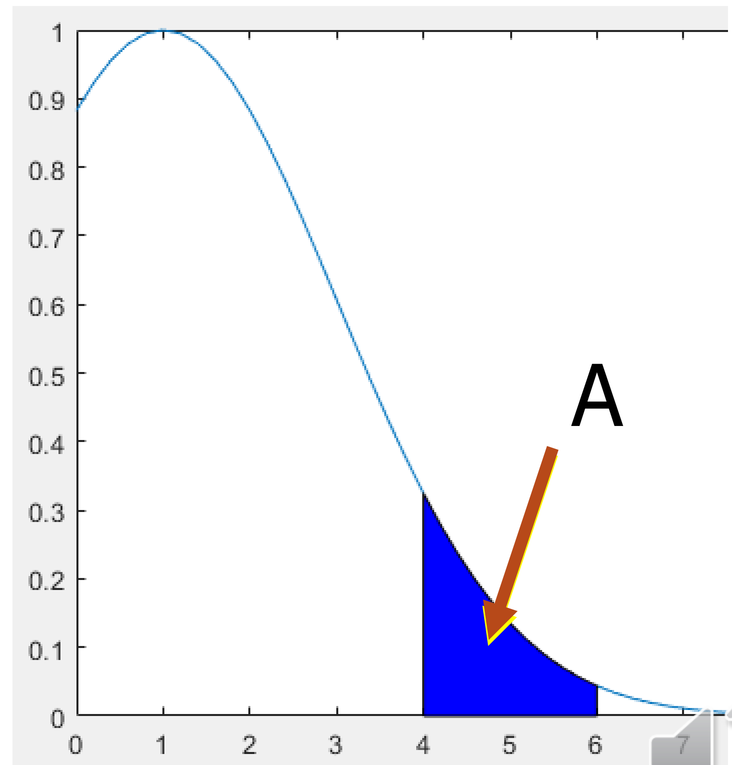
Importance sampling

Use the uniform sampling method to generate a number x inside the interval [r1,r2]. Then compute p(x).

```
%Importance sampling
n = 10000; j = 1; r1 = 4; r2 = 6;
while (j<=n)
    x = r1 + (r2-r1)*rand(1);
    c = 1; sigma = 2;
    p  = gaussmf(x, [sigma, c]);
    r = x*p*(r2-r1);
    a(j) = r;
    j = j +1;
end
```

mean(a)/A

mean(a) = 1.4299
A = 0.30379…
mean(a)/A = 4.7067

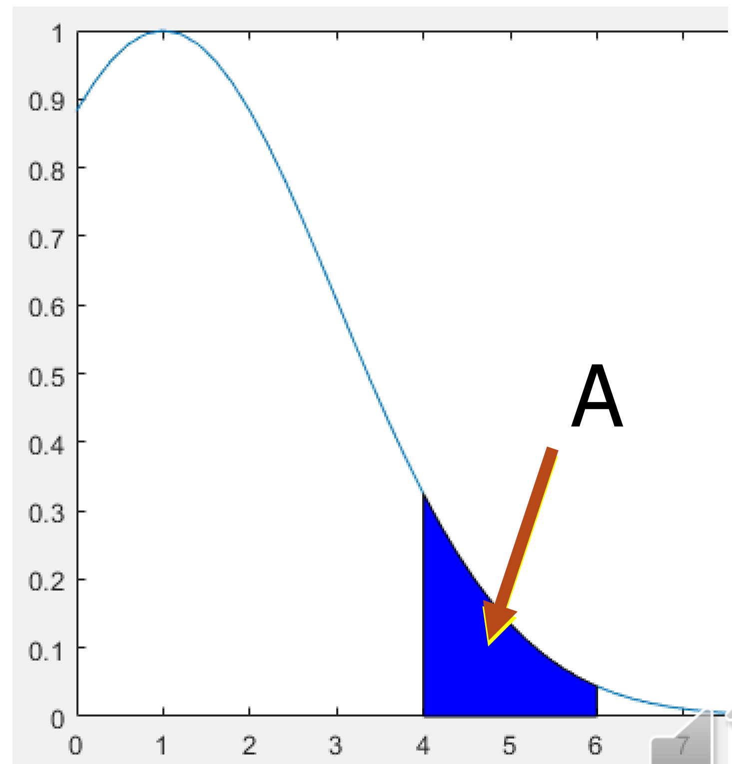


Importance sampling

```
Use the uniform sampling method to generate a number x
inside the interval [r1,r2]. Then compute p(x).
```

```
%Importance sampling
n = 10000; j = 1; r1 = 4; r2 = 6;
while (j<=n)
    x = r1 + (r2-r1)*rand(1);
    c = 1; sigma = 2;
    p  = gaussmf(x, [sigma, c]);
    r = x*p*(r2-r1);
    a(j) = r;
    j = j +1;
end
mean(a)/A
```

mean(a) = 1.4299
A = 0.30379…
mean(a)/A = 4.7067



Find the execution time of a matlab program

```
tic
n = 10000;
j = 1; r1 = 4; r2 = 6;
while (j<=n)
  c = 1; sigma = 2;
  r  = normrnd(c,sigma );
  if (r >=r1 && r<= r2)
     a(j) = r;
     j = j +1;
  end
end
mean(a);
toc
```

Elapsed time is 1.065695 seconds.

```
tic
n = 10000;
j = 1; r1 = 4; r2 = 6;
while (j<=n)
    x = r1 + (r2-r1)*rand(1);
    c = 1; sigma = 2;
    p  = gaussmf( x, [sigma, c] );
    r = t*p*(r2-r1);
    a(j) = r;
    j = j +1;
end
y = sym('2.718281828^(-(x-
1)^2/8)');
A = int(y,'x', 4, 6);
mean(a)/A;
toc
```

Elapsed time is 0.093241 seconds. With ;
Elapsed time is 0.073122 seconds. Without ;

Find the execution time of a matlab program

```
tic
n = 1000000;
j = 1; r1 = 4; r2 = 6;
while (j<=n)
 c = 1; sigma = 2;
 r  = normrnd(c,sigma );
 if (r >= r1 && r <= r2)
        a(j) = r;
        j = j +1;
 end
end
mnaive = mean(a);
toc
```

Elapsed time is 110.171087 seconds.

`mnaive =` mean = 4.6964

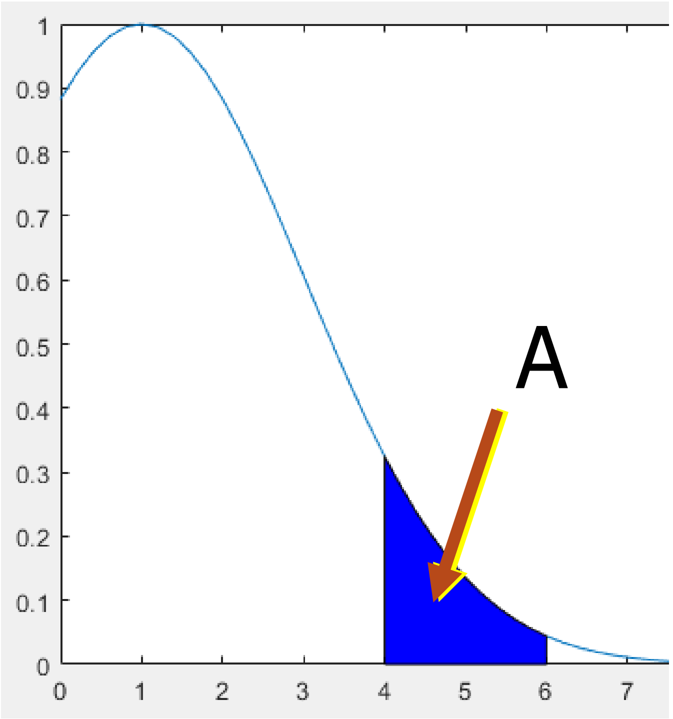
```
tic
n = 1000000;
j = 1; r1 = 4; r2 = 6;
while (j<=n)
    x = r1 + (r2-r1)*rand(1);
    c = 1; sigma = 2;
    p  = gaussmf( x, [sigma, c] );
    r = t*p*(r2-r1);
    a(j) = r;
    j = j +1;
end
y = sym('2.718281828^(-(x-
1)^2/8)');
A = int(y,'x', 4, 6);
mean(a)/A;
toc
```

Elapsed time is 5.561721 seconds.

mean(a)/A = 4.69555025

Expected Value

```
y = sym('2.718281828^(-(x-1)^2/8)');
A = int(y,'x', 4, 6);
y1 = sym('x*2.718281828^(-(x-1)^2/8)');
meanX = int(y1,'x', 4, 6)/A;
```



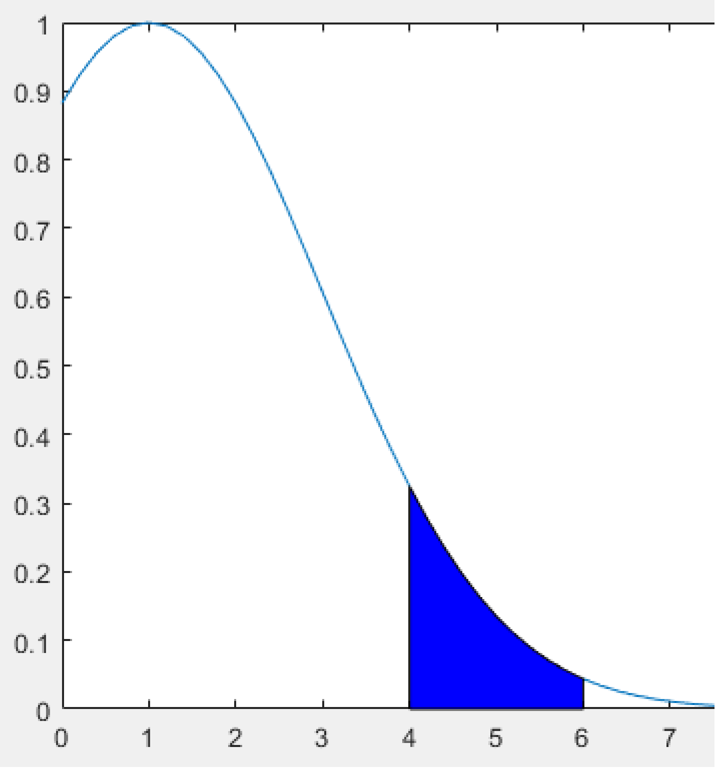

```
meanX = 4.6961666          % integral
mean(a)/A = 4.69555025     % importance sampling
mnaive = 4.6964            % naïve approach
```

$$\int_{r1}^{r2} x\, f(x) dx = \int_{r1}^{r2} x \frac{p(x)}{A} dx$$

$f(x)$ is a pdf in the interval [r1,r2].
p(x) is the pdf in the interval [-∞,∞].

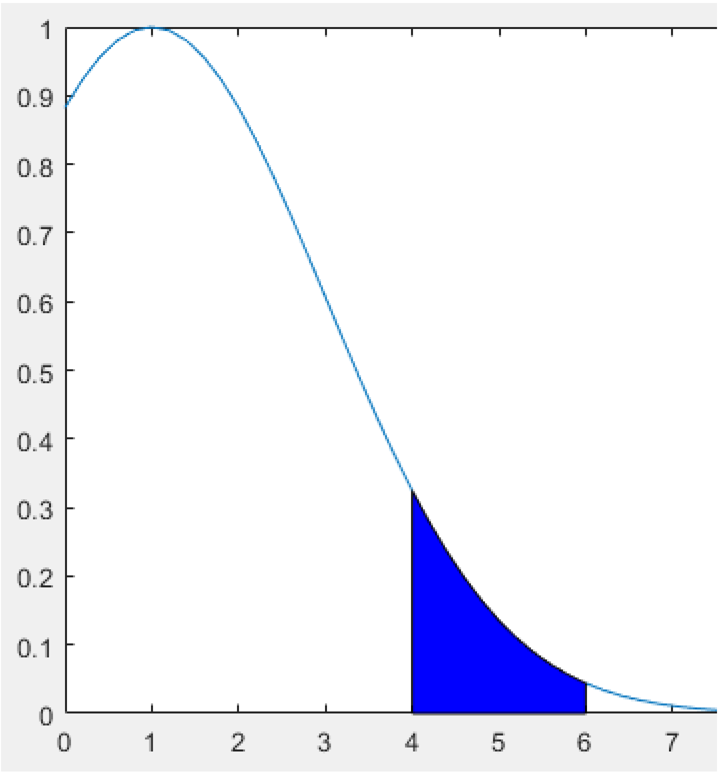
f(x) = p(x)/A

How to draw the figure?



How to draw the figure?

```
figure
x = 0:0.2:10
sig = 2; c = 1;
y = gaussmf(x, [sig c]);
plot(x,y);
hold on
x = 4:0.2:6
sig = 2; c = 1;
y = gaussmf(x, [sig c]);
x = [x,6];    %extra point
y = [y,0];
x = [x,4];    %extra point
y = [y,0];
fill(x,y,'b');
```



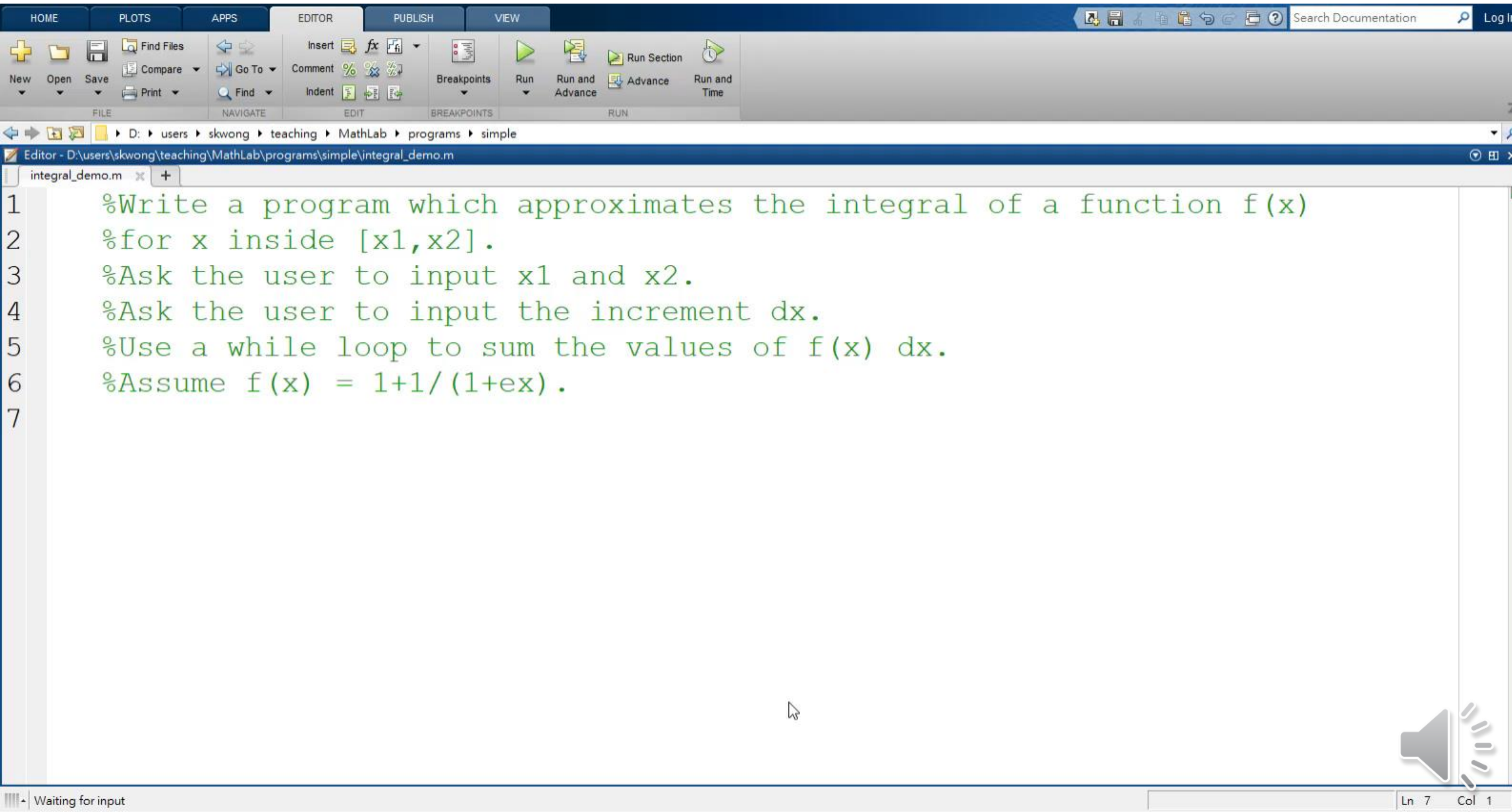
Demo

- Write a program which approximates the integral of a function f(x) for x inside [x1,x2].
- Ask to input x1 and x2.
- Ask to input the increment dx.
- Use a loop structure to sum the values of f(x) dx.
- $f(x) = 1+1/(1+e^x)$.
- Use the following equation:

$$\int_{x1}^{x2} f(x)dx \approx \sum_{x=x1}^{x2} f(x)\Delta x$$

- show the value of the integral to within four significant digits.

Demo
This demo also demonstrates how to debug.



```
%Integral computation. x in [1,2]
dx = -1;
while  dx < 0.0
   dx = input(‘Input the interval dx:');
   if (dx > 0.0 && dx <= 1.0)  break;  end
end

x1 = 1; x2 = 2; x = x1; F = 0;
while (x<=x2)
   F = F + eval_func(x)*dx;     x = x + dx;
end

function output = eval_func(x)
   output = x^2;       % f(x) = x^2
end
```

$$\int_{x1}^{x2} f(x)dx \approx \sum_{x=x1}^{x2} f(x)\Delta x$$

```
%Integral computation . x in [1,2]
dx = -1;
while  dx < 0.0
   dx = input(‘Input the interval dx:');
   if (dx > 0.0 && dx <= 1.0)  break;  end
end

x1 = 1; x2 = 2; x = x1; F = 0;
while (x<=x2)
   F = F + eval_func(x)*dx;     x = x + dx;
end

function output = eval_func(x)
   output = cos(x);   % f(x) = cos(x)
end
```

```
%Integral computation. x in [1,2]
dx = -1;
while  dx < 0.0
   dx = input('Input the interval dx:');
   if (dx > 0.0 && dx <= 1.0)  break;  end
end

x1 = 1; x2 = 2; x = x1; F = 0;
while (x<=x2)
   F = F + eval_func(x)*dx;     x = x + dx;
end

function output = eval_func(x)
   output = x^2;      % f(x) = x^2
end
```

$$\int_{x1}^{x2} f(x)dx \approx \sum_{x=x1}^{x2} f(x)\Delta x$$

```
%Integral computation . x in [1,2]
dx = -1;
while  dx < 0.0
   dx = input('Input the interval dx:');
   if (dx > 0.0 && dx <= 1.0)  break;  end
end

x1 = 1; x2 = 2; x = x1; F = 0;
while (x<=x2)
   F = F + eval_func(x)*dx;     x = x + dx;
end

function output = eval_func(x)
   output = cos(x);   % f(x) = cos(x)
end
```

integral function

```
% anonymous functions
% lambda
fun = @(x) (1+1./(1+exp(x)));

% integrate fun between x1 and x2
integral(fun,x1,x2)

integral(@(x) (1+1./(1+exp(x))),x1,x2)
```

lambda functions

Syntax:

@(argument list) expression;

Examples:

1) sqr = @(x) x.^2;

How to use?

sqr(5)

2) q = integral(@(x) x.^2,0,1);

lambda functions

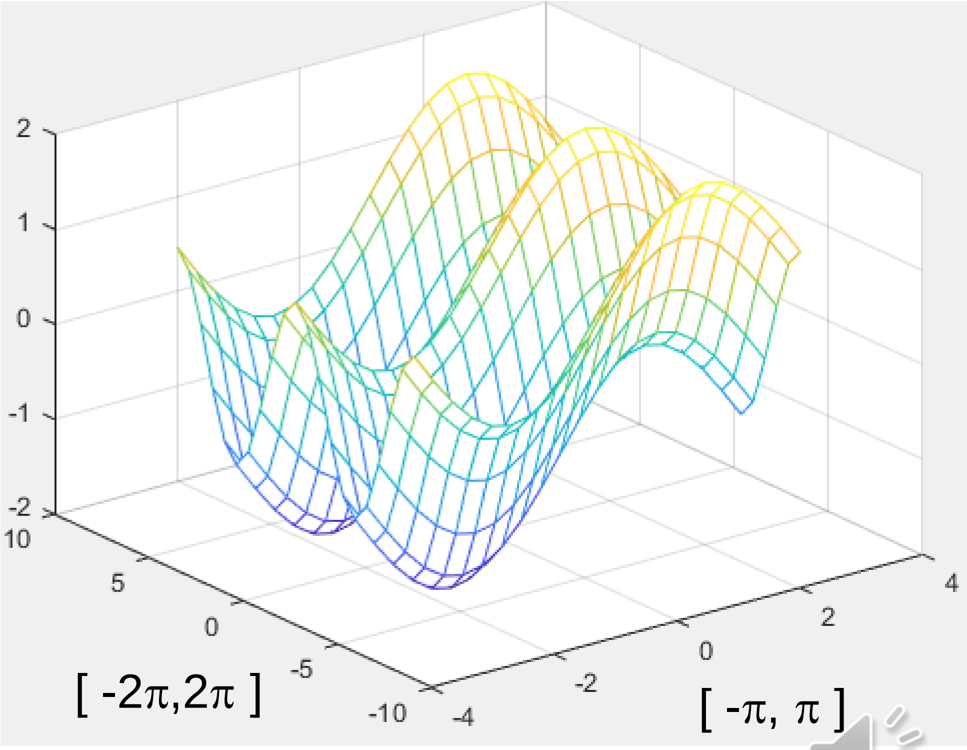
```
% do not need to be stored in a file
func = @(x,y) sin(x) +x*y;

%The function defined by function must be stored in a file.
function v = func(x,y)
        v = sin(x) + x*y;
end
```

Functions with Multiple Inputs or Outputs

```
myfunction = @(x,y) (x^2 + y^2 + x*y);
x = 1; y = 10;
z = myfunction(x,y)
```

Functions with Multiple Inputs or Outputs

```
c = 10;
mygrid = @(x,y) ndgrid((-x:x/c:x),(-y:y/c:y));
[x,y] = mygrid(pi,2*pi);
z = sin(x) + cos(y);
mesh(x,y,z)
```

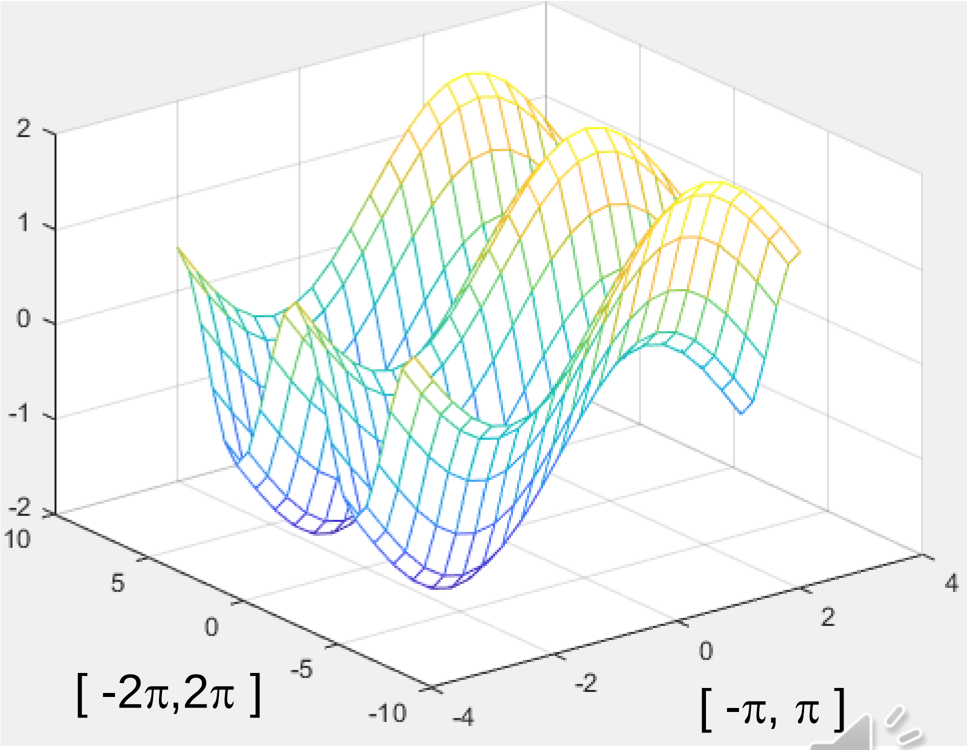


Note: the *ndgrid* function can return as many outputs as the number of input vectors.

Functions with Multiple Inputs or Outputs

```
c = 10;
mygrid = @(x,y) ndgrid((-x:x/c:x),(-y:y/c:y));
[x,y] = mygrid(pi,2*pi);
z = sin(x) + cos(y);
mesh(x,y,z)
```

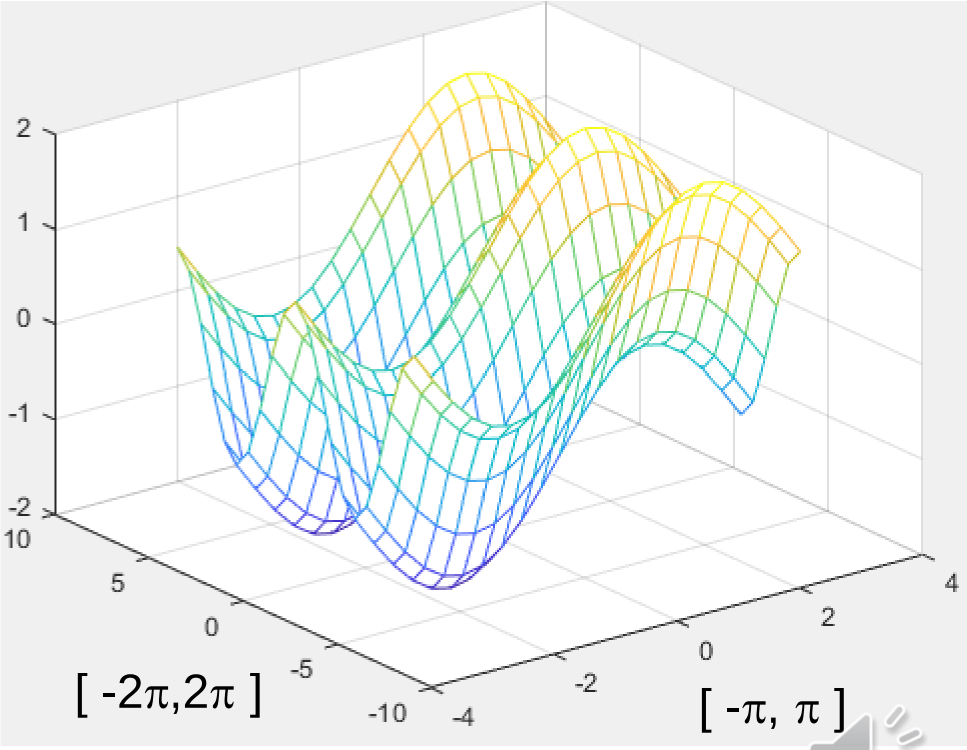
Note: the *ndgrid* function can return as many outputs as the number of input vectors.



Functions with Multiple Inputs or Outputs

```
c = 10;
mygrid = @(x,y) ndgrid((-x:x/c:x),(-y:y/c:y));
[x,y] = mygrid(pi,2*pi);
z = sin(x) + cos(y);
mesh(x,y,z)
```

Note: the *ndgrid* function can return as many outputs as the number of input vectors.



Arrays of Anonymous Functions

```
f = {
@(x)x.^2;
@(y)y+10;
@(x,y)x.^2+y+10
};
```

```
f =
  3×1 cell array
    {      @(x)x.^2}
    {      @(y)y+10}
    {@(x,y)x.^2+y+10}
```

```
f = {
    @(x) (x.^2);
    @(y) (y + 10);
    @(x,y) (x.^2 + y + 10)
};
```

How to call the functions?

```
f{1}(1)
f{2}(5)
f{3}(4, 5)
```

Arrays of Anonymous Functions

```
f = {
@(x)x.^2;
@(y)y+10;
@(x,y)x.^2+y+10
};
```

```
f =
  3×1 cell array
    {      @(x)x.^2}
    {      @(y)y+10}
    {@(x,y)x.^2+y+10}
```

```
f = {
    @(x) (x.^2);
    @(y) (y + 10);
    @(x,y) (x.^2 + y + 10)
};
```

How to call the functions?

```
f{1}(1)
f{2}(5)
f{3}(4, 5)
```

Arrays of Anonymous Functions

```
f = {
@(x)x.^2;
@(y)y+10;
@(x,y)x.^2+y+10
};
```

```
f =
  3×1 cell array
    {      @(x)x.^2}
    {      @(y)y+10}
    {@(x,y)x.^2+y+10}
```

```
f = {
    @(x) (x.^2);
    @(y) (y + 10);
    @(x,y) (x.^2 + y + 10)
};
```

How to call the functions?

```
f{1}(1)
f{2}(5)
f{3}(4, 5)
```

Arrays of Anonymous Functions

```
f = {
@(x)x.^2;
@(y)y+10;
@(x,y)x.^2+y+10
};
```

```
f =
  3×1 cell array
    {      @(x)x.^2}
    {      @(y)y+10}
    {@(x,y)x.^2+y+10}
```

Spaces are interpreted as column separators. Thus, either 1) omit spaces from expressions, or 2) enclose expressions in parentheses, such as

```
f = {
    @(x) x.^2 + 4
    @(y) y +    10
    @(x,y)  x.^2  + y  + 10
};
```

```
3×1 cell array

    {    @(x)x.^2+4}
    {      @(y)y+10}
    {@(x,y)x.^2+y+10}
```

Arrays of Anonymous Functions

```
f = {
@(x)x.^2;
@(y)y+10;
@(x,y)x.^2+y+10
};
```

```
f =
  3×1 cell array
    {      @(x)x.^2}
    {      @(y)y+10}
    {@(x,y)x.^2+y+10}
```

Spaces are interpreted as column separators. Thus, either 1) omit spaces from expressions, or 2) enclose expressions in **parentheses**, such as

```
f = {
    @(x) (x.^2);
    @(y) (y + 10);
    @(x,y) (x.^2 + y + 10)
};
```

How to call the functions?

```
f{1}(1)
f{2}(5)
f{3}(4, 5)
```

Recursive functions

A function references itself.

```
function output=m_lec_recursive_func(n)
  if (n==0)
     output = 1;
  else
     output = m_lec_recursive_func(n-1)*n;
  end
end
```

Recursive functions

S(0) = 1.

A function references itself.

```
function output=m_lec_recursive_func(n)
  if (n==0)
      output = 1;
  else
      output = m_lec_recursive_func(n-1)*n;
  end
end
```

Recursive functions

A function references itself.

S(0) = 1.

S(n) = S(n-1)*n.

```
function output=m_lec_recursive_func(n)
  if (n==0)
     output = 1;
  else
     output = m_lec_recursive_func(n-1)*n;
  end
end
```

Recursive functions

A function references itself.

```
function output=m_lec_recursive_func(n)
 if (n==0)
    output = 1;
 else
    output = m_lec_recursive_func(n-1)*n;
 end
end
```

S(0) = 1.

S(n) = S(n-1)*n.

n >= 0

Recursive functions

A function references itself.

```
function output=m_lec_recursive_func(n)
  if (n==0)
     output = 1; // boundary case
  else
     output = m_lec_recursive_func(n-1)*n;
  end
end
```

$S(0) = 1.$

$S(n) = S(n-1)*n.$

$n >= 0$

$n! = 1*2*3...*n$
$\quad = (n-1)! * n$

Fibonacci Sequence

S(1) = S(2) = 1;

S(n) = S(n-1) + S(n-2), for n >=3.

Define a function, fib, to compute the number for a given n.

Fibonacci Sequence (30 sec)

S(1) = S(2) = 1;

S(n) = S(n-1) + S(n-2), for n >=3.

Define a function, fib, to compute the number for a given n.

Fibonacci Sequence

$S(1) = S(2) = 1$;

$S(n) = S(n-1) + S(n-2)$, for $n >= 3$.

Structure plan:

- Determine the number of required arguments and their data types.
- Determine the outputs and their data types.
- Write down the boundary cases.
- Write down other cases.

Fibonacci Sequence

S(1) = S(2) = 1;

S(n) = S(n-1) + S(n-2), for n >=3

```
function r = fib(n)
   if (n<=2)
     r = 1;
   else
     r = fib(n-1) + fib(n-2)
   end
end
```

Exercise: A Sequence

$S(1) = S(2) = 1; S(3) = 5;$

$S(n) = S(n-1) + S(n-2) + 3S(n-3)$, for $n >= 4$.

Define a function, seq, to compute the number for a given n.

Exercise: A Sequence

S(i) = i^2, for 0<= i <= m;

S(n) = $\sum_{i=0}^{m} iS(n-i-1)$, for n >=m+1.

Define a function, seq, to compute the number for a given n.

Exercise
Importance sampling

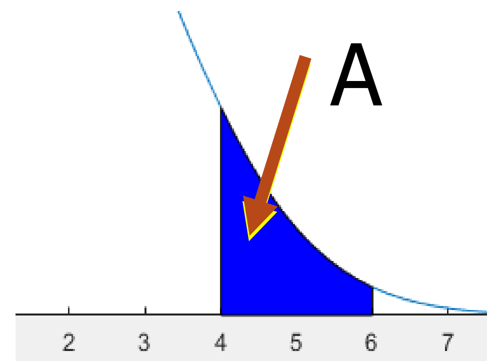
Compute the average of random numbers generated by the normal distribution (1,2)

inside the intervals [r1,r2] and [r3,r4].

What is the target probability density function, p?

$$\mu = \int_{\mathcal{D}} f(\boldsymbol{x}) p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$$

f(x) = x



$$\mu = \int_{\mathcal{D}} x\; p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$$

$$p(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}} \Big/ A$$

Exercises

- Write a code that approximates the integral of $\sin(x^2)/x$ from 1 to 2. Use a while loop to sum the values of $\sin(x^2)/x$ from 1 to 2. Let the user input the increment to be used.
- Use a lambda function.
- Use the following approximation:
    1. Midpoint Rule (or rectangle rule)
    2. Trapezoidal rule

- The numerical value of this integral, to within four significant digits.

Exercises

Let y1(x) = x.

A. Draw the curve of y1(x) interactively from x = 0 to 2. The curve is black. The linewidth is 3.

B. At the same time, interacitvely fill a region with red.

Assume that x = x0 at the current step. The region is bounded by y1, the x-axis, and the interval of the region is [0, x0] along the x-axis.