Asignatura	Datos del alumno	Fecha
Modelado y Simulación Numérica	Alvarez Calderon, Alvaro Antonio Balsells Orellana, Jorge Augusto Diaz Saborio, Erick Wilfredo Luna Rodríguez, Daniel Antonio Perla Álvarez, Josué Aristides	01/Feb/2021

Actividad grupal: Simulación de sistemas continuos

Generación de números aleatorios

1. Completa el siguiente esquema con las condiciones que deben verificarse para tener el periodo máximo o completo:

   Métodos de generación de números aleatorios congruenciales

   Fórmula general: $x_n = (a\, x_{n-1} + b)\ mod\ m$

   ➢ Métodos congruenciales mixtos ($b = número\ entero\ positivo$)

   - Caso general m cualquier valor
     - m y b son coprimos
     - m/q donde q es primo y divide a a-1 (es decir factores de m dividen a a-1)
     - m múltiplo de cuatro y a-1 debe ser múltiplo de cuatro
   - Caso $m = 2^k$ (m potencia de 2)
     - b es impar
     - a mod 4=1

   ➢ Métodos congruenciales multiplicativos ($b = 0$)

   - Caso general m cualquier valor
     - m es primo
     - a es raíz primitiva de módulo m ($a^k\ mod\ m = b$)
   - Caso $m = 2^k$ (m potencia de 2)
     - x_o es impar
     - a mod 8 es 3 o 5

2. Para cada uno de los cuatro casos anteriores. Propón valores para $a, b,\ m\ y\ x_0$ para que el periodo sea máximo/completo y superior a 50.

Asignatura	Datos del alumno	Fecha
Modelado y Simulación Numérica	Alvarez Calderon, Alvaro Antonio Balsells Orellana, Jorge Augusto Diaz Saborio, Erick Wilfredo Luna Rodríguez, Daniel Antonio Perla Álvarez, Josué Aristides	01/Feb/2021

$x_n = (ax_{n-1} + b)\, m$

a) En el primer caso T=m por lo que $m \geq 50$ y múltiplo de 4, se elige 100 como valor de m que cumple estas condiciones.

Los factores de 100 son 2,2,5,5 y 1, b debe ser coprimo con m por lo que no debe compartir factores iguales entonces se elige b=3.

a-1 debe ser múltiplo de cuatro y debe dividir a un número q que es primo factor de m, siendo q=5. Entonces

$$a - 1 = 80 \; a = 81$$

Solución: $x_n = (81x_{n-1} + 3)\, 100$ cuyo periodo es 100

b) Para el caso dos $m = 2^k$, se elige $m = 2^7 = 128$, b es impar y se elige b=5. Por último $a4 \; = 1$, para que esto se cumpla a se elige de 17. Solución:

$$x_n = (17x_{n-1} + 5)\, 128$$

Cuyo periodo es de 128.

c) Caso tres, b=0 y m es primo, el primo superior a 50 es 53, por lo que m=53. Y debe ser raíz primitiva de módulo m. Al tomar a igual a dos cumple $a^k 53 \; = b$, como se verifica en las tablas siguientes:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	4	8	16	32	11	22	44	35	17	34	15	30	7	14	28	3	6	12

20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
24	48	43	33	13	26	52	51	49	45	37	21	42	31	9	18	36	19

38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
38	23	46	39	25	50	47	41	29	5	10	20	40	27	1

Entonces la solución propuesta es $x_n = (2x_{n-1})\,53$ con periodo de 52.

d) El valor de m debe ser potencia de 2 por lo que se elige 128 y a debe cumplir que $a8 = 5 \text{ ó } 3$ se elige a=29 para que al sacarle módulo 8 de 5. La raíz es 1 para que sea impar. Siendo solución: $x_n = (29x_{n-1})\,128$. Su periodo es $T = 2^{k-2} \rightarrow T = 2^{7-2} = 32$

3. Realizar una función en Matlab que genera valores aleatorios. La cabecera de la función deberá ser:

$$Function\ [u,x] = gen\_num\_alea(a,b,m,x)$$

La variable x será una variable opcional tanto de entrada como de salida. Si se da se utilizará como semilla inicial. Si esta variable no se introduce en la llamada de la función, el método utilizará el último valor empleado (deberán usarse variables globales o persistentes). Si la variable x no se ha definido la función devolverá un error.

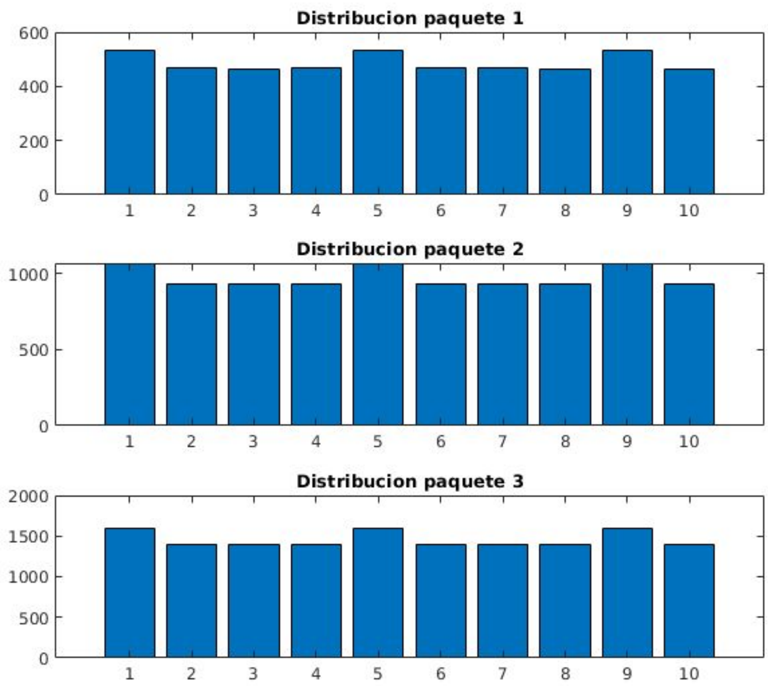
```
function [u,x] = gen(a,b,m,varargin)
global global_seed;
%Generacion de numeros aleatorios.
%gen(a,b,m,x)
%"a" debe ser menor a "m"
%"b" debe ser menor a "m"
%si existe "b", es metodo mixto, si no existe "b" es metodo
multiplicativo.
%si no se ingresa "x", se toma "x" inicial a partir de la semilla
global xn;

valid_scalar_b = @(x) isnumeric(x) && isscalar(x) && (x >= 0);
valid_scalar = @(x) isnumeric(x) && isscalar(x) && (x > 0);
@ -8,7 +14,7 @@ p = inputParser;
addRequired(p, 'a', valid_scalar);
```

```
addRequired(p, 'b', valid_scalar_b);
addRequired(p, 'm', valid_scalar);
addOptional(p, 'x', global_seed, valid_scalar)
addOptional(p, 'x', xn, valid_scalar)
parse(p, a, b, m, varargin{:});

a = p.Results.a;
@ -23,8 +29,8 @@ elseif (a<m) && (b>m)
elseif (a>m) && (b>m)
    error('Valor no válido de a y b')
elseif (a<m) && (b<m)
      x = mod((a*x+b), m)
      u = x/m
      x = mod((a*x+b), m);
      u = x/m;
end
end
```

4. Utilizando la función del ejercicio anterior generar 3 paquetes de 5000 iterados de valores aleatorios entre 0 y 1. Para cada paquete, agrupados en 10 grupos por fragmentos de 0.1 unidades. Realiza un histograma para cada paquete y comprueba que la función es uniforme.

```
% Variable global
global xn;
xn = 42;
x2 = 3;
x3 = 42;

for i=1:5000
    [u1,xn] = gen(16,4,75);
    [u2,x2] = gen(16,4,75,x2);
    [u3,x3] = gen(16,4,75,x3);

    paquete_1(i) = u1;
    paquete_2(i) = u2;
```

```
    paquete_3(i) = u3;
end

% ========== Problema 4 ============
packages = [paquete_1, paquete_2, paquete_3];
index = 0;
x=[1:10];
for k=1:3
    pkg = packages(index*5000+1:5000*k);
    bin=1;
    for i=0.1:0.1:1
        j=i-0.1;
        y(bin) = length(pkg(pkg>=j & pkg<i));
        bin = bin +1;
    end

    subplot(3,1,k),bar(x,y)
    title(['Distribucion paquete ' num2str(k)])
end
```



Las distribuciones son aproximadamente homogéneas según puede observarse en la imagen de arriba.

5. Dados los siguientes generadores congruenciales:

Calcular (y justificar debidamente) el periodo de los métodos.

Para validar el valor del período junto a la justificación, se realizó el siguiente programa en Matlab, el cual valida 3 paquetes. Cada uno de los paquetes contiene dentro un vector con 5000 valores de cada uno de los incisos, siendo el paquete 1 correspondiente a Xn, paquete 2 correspondiente a Yn y paquete 3 correspondiente a Zn.

```
vector = [paquete_1;paquete_2;paquete_3];
Period = [0 0 0];
```

```
for a=1:3
    Temp = vector(a,:);

    Index1 = find(Temp==vector(a,1));
    Index2 = find(Temp==vector(a,2));

    if Index1(1)==(Index2(1)-1)
        Period(a) = Index1(2) - Index1(1) -1;
    end
end
disp(Period);


El resultado mostrado es el siguiente:
>> periodo
    75   128    64
```

- $x_n = 16x_{n-1} + 4 \; mod \; 75, \; x_0 = 3$

  Prueba caso en que T=m, los factores de b=4 son 2*2 y los factores de 75 son 5*5*3, por lo que b y m son coprimos. El valor anterior al multiplicador es 15 y siendo q=5 este divide a 15, pero 15 no es múltiplo de 4 ni el módulo lo es. Se procede a determinar mediante el conteo del ciclo de repetición con el uso de la función adjunta. Con ello se determina que el periodo es siempre igual a m por lo que T=75.

- $y_n = 9y_n + 7 \; mod \; 128, \; y_0 = 3$

  El valor de b 7 y es primo, m es potencia de dos por lo que b y m son coprimos.

El valor anterior del multiplicador es 8 y siendo q=2, este divide a 8, y este con el módulo son múltiplos de 4 por lo que cumple que T=m, entonces el periodo es 128.

- $z_n = 21 z_n \; mod \; 256, z_0 = 7$

  Siendo que b es cero y m es potencia de dos (2^8) y la semilla es impar y 21 mod 8 es 5, entonces se cumple que T= 2^{k-2} = $2^{8-2} = 2^6 = 64$.

- Combinar los tres generadores para obtener un método de generación de un periodo superior. Calcular el periodo de este nuevo método.

M=max(75,128,256)=256

$$w_n = (z_n - y_n + x_n) w_{n-1} \; mod \; 256, w_0 = 7$$

$$T = \frac{(256-1)(128-1)(75-1)}{2^{3-1}} = 599,122$$

6. Generación de variables aleatorias

a) Describe el algoritmo de la transformada inversa

Paso 1: Identificar la función que describe la distribución.

Paso 2: Generar la distribución acumulada, integrar $f(x) \to F(x)$.

Paso 3: Resolver x en términos de u, $x = F^{-1}(u)$

Paso 4: Transformar los números aleatorios $x_i = F^{-1}(u_i)$

b) Calcula mediante el método de la transformada inversa tres valores aleatorios siguiendo la distribución:

$$f(x) = \{ \tfrac{3x^2}{8}, \; si \; 0 \leq x \leq 2 \qquad 0, \; en \; caso \; contrario$$

Paso 1: $f(x) = \frac{3x^2}{8}$

Paso 2: $F(x) = \int_0^x \frac{3x^2}{8} dx \to \frac{3}{8} \int_0^x x^2 dx \to \frac{3}{8} \bullet \left(\frac{x^3}{3} \right) |x \; 0 \; \to \frac{1}{8} x^3 - 0$

$F(x) = \frac{1}{8}x^3$

Paso 3: Si $F(x) = u \rightarrow u = \frac{1}{8}x^3$

$F^{-1}(u) \rightarrow x = 2\sqrt[3]{u}$

Paso 3: Transformar los números aleatorios

u	x
0.7365	1.8061
0.2095	1.1878
0.6284	1.7130

c) Elabora una función en Matlab que utilice la función del ejercicio 3 para generar valores aleatorios siguiendo esta distribución. La cabecera de la función deberá ser:

```
function x=gen_var_trainv()
u=gen_num_alea(3,1,1048576)
if u>=0 & u<=1
    x=2*(u).^(1/3);
else
    x=0;
end
end
```

Observación: la función no deberá tener ningún valor de entrada.

7.

a) Describe el algoritmo del método de la composición.

Paso 1: Identificar las funciones que describen la distribución.

Paso 2: Generar la distribución acumulada, integrar $f(x) \rightarrow F(x)$ de forma que la integral es igual a la unidad.

Paso 3: Generar un índice $P(I = i) = p_i$

Paso 4: Resolver x en términos de u, $x = F^{-1}(u)$

Paso 5: Transformar los números aleatorios $x_i = F^{-1}(u_i)$ de acuerdo con p_i

b) Calcula mediante el método de la composición tres valores aleatorios siguiendo la distribución:

$$f(x) = \{\frac{1}{4\pi} + \frac{\cos cos\,(\frac{x}{4})}{8\sqrt{2}} + \frac{\sin sin\,(\frac{x}{2})}{4},\ \ si\ 0 \le x \le \pi \quad 0\ ,\ en\ caso\ contrario$$

Haciendo una pequeña modificación a la función de distribución en el intervalo

$$0 \le x \le \pi\,.$$

$$f(x) = \frac{1}{4\pi} + \frac{2}{2\sqrt{2}}\frac{\cos cos\,(\frac{x}{4})}{4} + \frac{1}{2}\frac{\sin sin\,(\frac{x}{2})}{2}$$

Encontrando las funciones de densidad

$$a\int_0^{\pi} dx = ax|_0^{\pi} = a\pi$$

$$a\pi = 1 \rightarrow a = \frac{1}{\pi}$$

$$f_1(x) = \frac{1}{\pi}1$$

$$b\int_0^{\pi} \frac{\cos cos\,(\frac{x}{4})}{4} dx = b \sin sin\ \left(\frac{x}{4}\right)\ |_0^{\pi} = \frac{\sqrt{2}}{2}b$$

$$\frac{\sqrt{2}}{2}b = 1 \rightarrow b = \frac{2}{\sqrt{2}}$$

$$f_2(x) = \frac{2}{\sqrt{2}}\frac{\cos cos\,(\frac{x}{4})}{4}$$

$$c\int_0^{\pi} \frac{\sin sin\,(\frac{x}{2})}{2} dx = -\,c \cos cos\ \left(\frac{x}{2}\right)\ |_0^{\pi} = c$$

$$c = 1 \rightarrow c = 1$$

$$f_3(x) = 1\frac{\sin sin\left(\frac{x}{2}\right)}{2}$$

Entonces la función de densidad con sus pesos es

$$f(x) = P_1\frac{1}{\pi} + P_2\frac{2}{\sqrt{2}}\frac{\cos cos\left(\frac{x}{4}\right)}{4} + P_3\frac{\sin sin\left(\frac{x}{2}\right)}{2}$$

Encontrando el valor de $P_1, P_2 \; y \; P_3$

$$P_1\frac{1}{\pi} = \frac{1}{4\pi} \rightarrow P_1 = \frac{1}{4}$$

$$P_2\frac{2}{\sqrt{2}}\frac{\cos cos\left(\frac{x}{4}\right)}{4} = \frac{2}{2\sqrt{2}}\frac{\cos cos\left(\frac{x}{4}\right)}{4} \rightarrow P_2 = \frac{1}{4}$$

$$P_3\frac{\sin sin\left(\frac{x}{2}\right)}{2} = \frac{1}{2}\frac{\sin sin\left(\frac{x}{2}\right)}{2} \rightarrow P_3 = \frac{1}{2}$$

Nótese que $P_1 + P_2 + P_3 = \frac{1}{4} + \frac{1}{4} + \frac{1}{2} = 1$ y la función de distribución queda de la siguiente manera identificando los pesos.

$$f(x) = \frac{1}{4}\frac{1}{\pi} + \frac{1}{4}\frac{2}{\sqrt{2}}\frac{\cos cos\left(\frac{x}{4}\right)}{4} + \frac{1}{2}\frac{\sin sin\left(\frac{x}{2}\right)}{2}$$

Encontrando la función de distribución resultando

$$\int f(x) = \frac{1}{4}\int\frac{1}{\pi}dx + \frac{1}{4}\int\frac{2}{\sqrt{2}}\frac{\cos cos\left(\frac{x}{4}\right)}{4}dx + \frac{1}{2}\int\frac{\sin sin\left(\frac{x}{2}\right)}{2}dx$$

$$F(x) = \frac{1}{4}\frac{1}{\pi}x + \frac{1}{4}\frac{2}{\sqrt{2}}\sin sin\left(\frac{x}{4}\right) - \frac{1}{2}\cos cos\left(\frac{x}{2}\right)$$

Invirtiendo las funciones de distribución

$$F_1(x) = \frac{1}{\pi}x = R_2 \rightarrow x = \pi R_2$$

$$F_2(x) = \frac{2}{\sqrt{2}}\sin sin\left(\frac{x}{4}\right) = R_2 \rightarrow x = 4\left(\frac{\sqrt{2}}{2}R_2\right)$$

$$F_3(x) = \left(\frac{x}{2}\right) = R_2 \rightarrow x = 2(-R_2)$$

Construyendo los siguientes pesos, tenemos que:

$P_1 = p_1 = \frac{1}{4}$

$P_2 = p_1 + p_2 = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$

$P_3 = p_1 + p_2 + p_3 = \frac{1}{4} + \frac{1}{4} + \frac{1}{2} = 1$

Generamos los valores aleatorios R_1 y R_2 con $U(0,1)$

Si $R_1 \leq \frac{1}{4} \rightarrow x = \pi R_2$

Si $\frac{1}{4} < R_1 \leq \frac{1}{2} \rightarrow x = 4\left(\frac{\sqrt{2}}{2} R_2\right)$

Si $\frac{1}{2} < R_1 \leq 1 \rightarrow x = 2(-R_2)$

Generando tres valores aleatorios

1) $R_1 = 0.24$ y $R_2 = 0.71$

   Como $R_1 = 0.24 \leq \frac{1}{4} \rightarrow x = \pi R_2 = \pi(0.71) = 2.23$

2) $R_1 = 0.93$ y $R_2 = 0.77$

   Como $R_1 = 0.93 \leq 1 \rightarrow x = 2(-R_2) \;\; = 2(-0.77) \;\; = 4.89$

3) $R_1 = 0.38$ y $R_2 = 0.74$

   Como $R_1 = 0.38 \leq \frac{1}{2} \rightarrow x = 4\left(\frac{\sqrt{2}}{2} R_2\right) \;\; = 4\left(\frac{\sqrt{2}}{2}(0.74)\right) \;\; = 2.20$

c) Elabora una función en Matlab que utilice la función del ejercicio 3 para generar valores aleatorios siguiendo esta distribución. La cabecera de la función deberá ser:

```
function u = gen_var_compo()
global xn;
[R1,xn] = gen(16,4,75,xn);
[R2, xn] = gen(16,4,75,xn);

if R1 <= 1/4
    fx=pi*R2
elseif (1/4< R1) && (R1<=1/2)
```

```
    fx= 4* asin((sqrt(2)/2)*R2)
elseif (1/2<R1) && (R1<=1)
    fx=2*acos(-1*R2)
end

u=fx
end
```

8.

a) Describe el algoritmo del método de aceptación rechazo.

La función de densidad de X es conocida y está acotada [a,b] y es no nula.

El procedimiento para seguir es el siguiente:

1) Generar dos números aleatorios u y v independientes y uniformemente distribuidos.

2) Hacer $x = a + (b - a)u$ en una variable aleatoria distribuida uniformemente en el soporte [a,b]

3) Hacer y=cv, para transformar v en una variable aleatoria distribuida uniformemente en el intervalo [0,c]

4) Como último paso se calcula $f(x)$ y en caso de que $y \leq f(x)$, se acepta x como valor generado de la variable aleatoria X. En caso contrario se vuelve al paso 1.

b) Calcula mediante el método de aceptación-rechazo tres valores aleatorios siguiendo la distribución:

$$f(x) = \{ \tfrac{x}{6}, \ \ si\ 0 \leq x \leq 3 \ \ 2 - \tfrac{x}{2}, \ \ si\ 3 \leq x \leq 4 \ \ 0\ , \ \ en\ caso\ contrario$$

Como $f(x)$ esta compuesta por funciones lineales se evalúan en el extremo superior si es creciente y en su extremo inferior si es decreciente para determinar su máximo:

En el intervalo [0,3] el máximo es $f(3)=\frac{3}{6}=\frac{1}{2}$, al ser continua el máximo está en 0.5.

$$x_0 = 0 + u(4-0) \;\; x_0 = 4u \;\; y = 0.5v$$

1) u=0.6933 y v=0.4, para el cual $x_0 = 4(0.6933) = 2.7732$ y y=0.5(0.4)=0.2

$$f(2.7732)=\frac{2.7732}{6}=0.4622$$

Cumple que $0.2 \leq 0.4622$ se acepta valor de 2.7732 como valor de la variable aleatoria.

2) u=0.72 y v=0.52, para el cual $x_0 = 4(0.72) = 2.88$ y y=0.5(0.52)=0.26

$$f(2.88)=\frac{2.88}{6}=0.48$$

Cumple que $0.26 \leq 0.48$ se acepta valor de 2.88 como valor de la variable aleatoria.

3) u=0.0267 y v=0.36, para el cual $x_0 = 4(0.0267) = 0.1068$ y y=0.5(0.36)=0.18

$$f(0.1068)=\frac{0.1068}{6}=0.0178$$

Cumple que $0.26 \leq 0.48$ se acepta valor de 2.88 como valor de la variable aleatoria.

c) Elabora una función en Matlab que utilice la función del ejercicio 3 para generar valores aleatorios siguiendo esta distribución. La cabecera de la función deberá ser:

```
function u = gen_var_acerec()
    global xn;
```

```
    flag = 0;

    while flag==0
        [u1, xn] = gen(16,4,75,xn);
        [v1, xn] = gen(16,4,75,xn);

        a = 0;
        b = 4;
        c = 3;

        x = a+(b-a)*u1;
        y = c*v1;

        if x>=0 && x<3
            fx = x/6;
        elseif x>=3 && x<4
            fx = 2 - (x/2);
        end

        if y<=fx
            flag = 1;
        end
    end
    u = fx;
end
```

Observación: la función no deberá tener ningún valor de entrada.

9. Para cada función generada en los tres ejercicios anteriores:

- Calcular 10000 (diez mil) valores aleatorios.
- Agruparlos en 20 grupos equidistribuidos en el soporte de las funciones de densidad.

Método transformada inversa

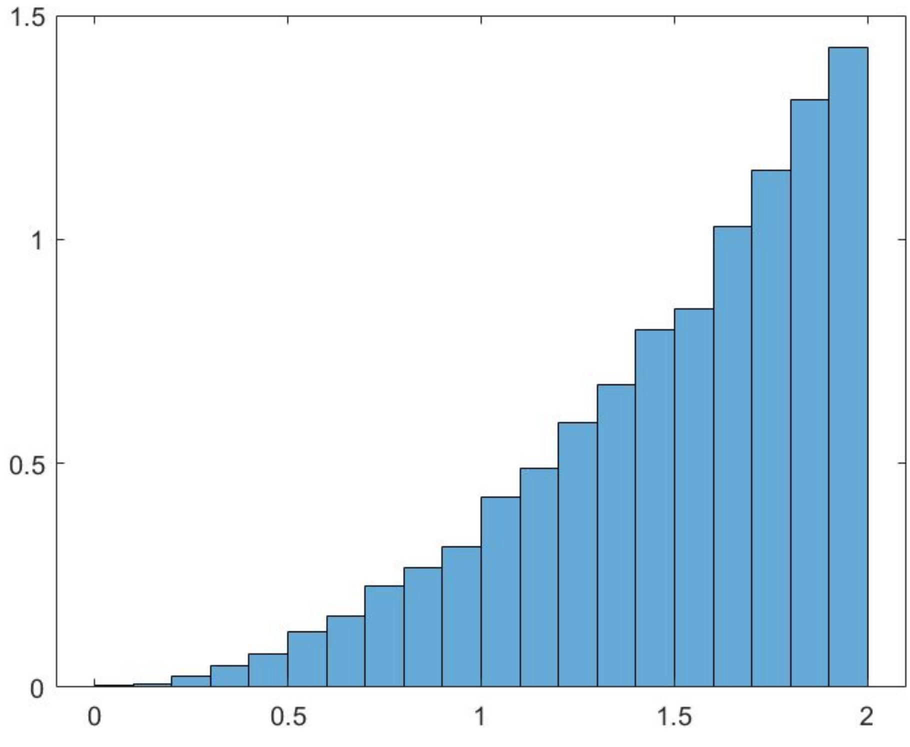
```
% Número aleatorio prueba con aleatorios de Matlab
r=rand(1,1000);
%Función transformada inversa
x=2*(r).^(1/3);
t=0:0.01:2;
y=3*t.^2/8;
histogram(x,20,'Normalization','pdf')
hold on
plot(t,y)
title('Números aleatorios transformados a la
distribución de la función')
```

Método composición

```
global xn;
 global xn2;
 xn = 42;
 xn2 = 1;
for i=1:10000
    rand_numbers(i) = gen_var_compo();
end
x = 0:pi/100:pi;
y = (1/4*pi) + (cos(x/4)/(8*sqrt(2))) + (sin(x/2)/4);
%y = -(sin(x/4)/(32*sqrt(2))) + (cos(x/2)/8)
%y = (1/pi) + (cos(x/4)/2*sqrt(2)) + sin(x/2)/2
histogram(rand_numbers,20, 'Normalization', 'pdf')
hold on
plot(x,y)
```

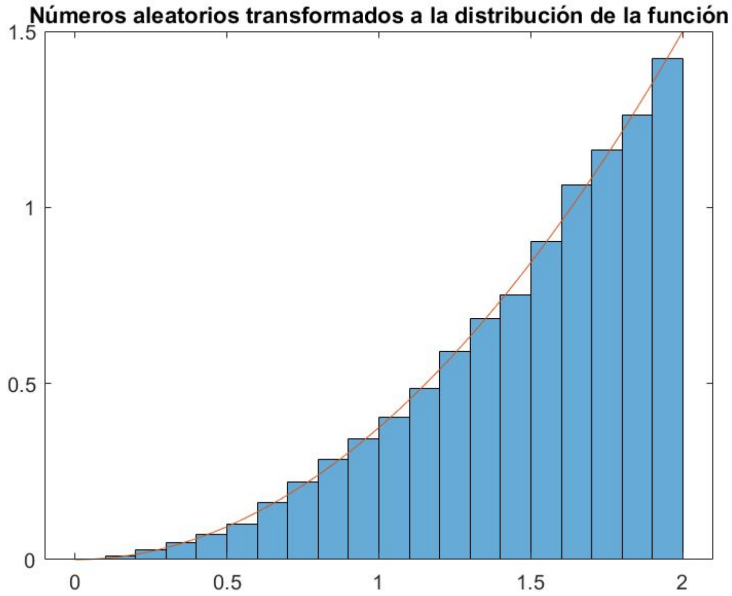
- Dibujar el histograma normalizado.

  Método transformada inversa

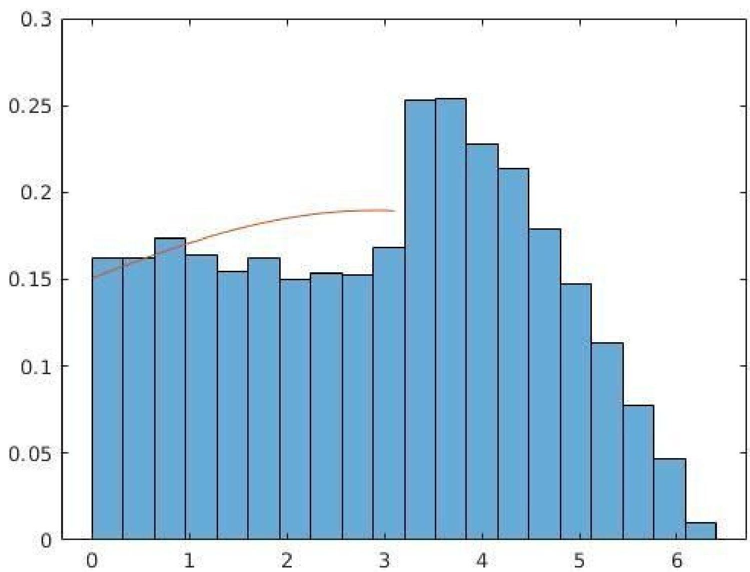


- Comparar el histograma anterior con la función de densidad.

Método transformada inversa



Método composición



Asignatura	Datos del alumno	Fecha
Modelado y Simulación Numérica	Alvarez Calderon, Alvaro Antonio Balsells Orellana, Jorge Augusto Diaz Saborio, Erick Wilfredo Luna Rodríguez, Daniel Antonio Perla Álvarez, Josué Aristides	01/Feb/2021

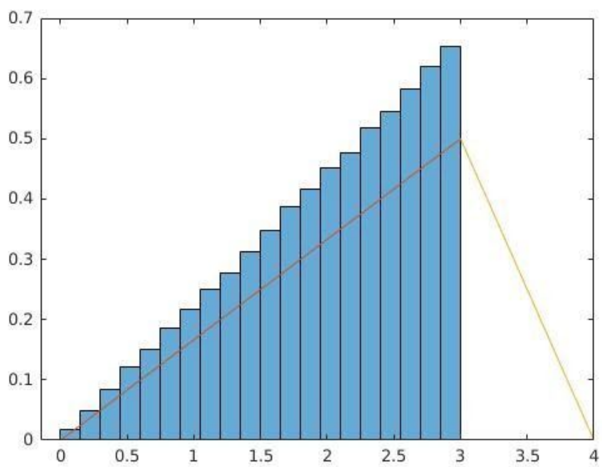
Método de aceptación rechazo

```
function [u] = gen_var_acerec()
    %global xn
    %global xm
    flag = 0;

    while flag==0
        %[u1, xn] = gen(3,1,1048576,xn);
        %[v1, xm] = gen(3,1,1048576,xm);

        u1 = rand();
        v1 = rand();

        a = 0;
        b = 4;
        c = 1/2;
        x = a+(b-a)*u1;
        y = c*v1;

        if x>=0 & x<3
            fx = x/6;
        elseif x>=3 & x<4
            fx = 2 - (x/2);
        end

        if y<=fx
            flag = 1;
        end
    end
    u = fx;
end
```

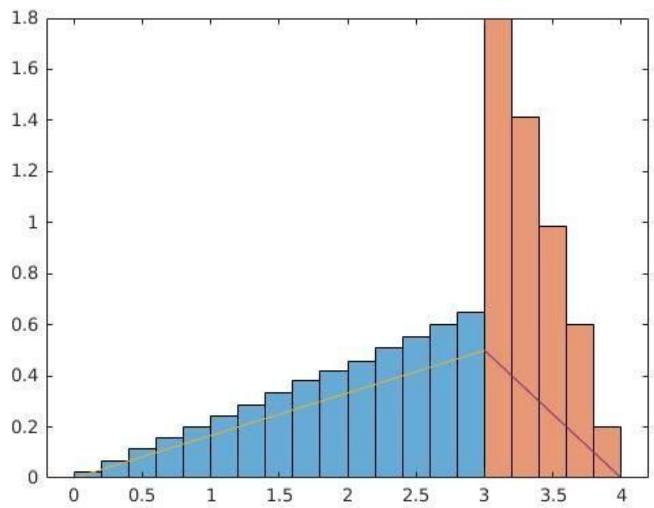
Graficando la función en 2 partes diferentes obtenemos el siguiente código:

```
        for j=1:100000
            u1 = gen_var_acerec1();
            paquete(j) = u1;
```

```
end

%Index1 = find(paquete==paquete(1));
%Index2 = find(paquete==paquete(2));
%if Index1(1)==(Index2(1)-1)
%    Period = Index1(2) - Index1(1);
%end
%disp(Period);

paq1 = paq1.*6;
histogram(paq1,15,'Normalization','pdf')
hold on

paq2 = (4-paq2.*2)
histogram((paq2),5,'Normalization','pdf')

x = [0:0.01:3];
y = x/6;
plot(x,y)

x = [3:0.01:4];
y = 2 - x/2;
plot(x,y)
```



En este caso, estamos obteniendo un histograma de todos los valores de 0 a 0.5 obtenidos, el cual se normaliza y se grafica, tal y como se observa en la gráfica anterior. Al ser una función por partes, tenemos 2 funciones totalmente diferentes para determinar los puntos bajo el gráfico, por lo cual optamos por separar los valores de 0 a 3 en un vector, y los valores de 3 a 4 en otro vector. Esto nos genera un gráfico como el que se encuentra a continuación, en el cual se visualizan por separado ambos vectores.



Simulación de eventos discretos y análisis estadístico

10. Escribir las siguientes funciones en Matlab teniendo en cuenta que lista es una lista de valores que se quiere analizar.
    - function [linf, lsup]=ic_media(lista, alpha, desviación): esta función calcula el intervalo de confianza para la media con desviación conocida (si desviación está presente) o desconocida (si desviación no está presente). El nivel de significación será alpha.

```
function
```

```
[linf,lsup]=ic_media(lista,alpha,desviacion)
p=1-alpha/2;
n=length(lista);
z=norminv(p);
t=tinv(p,n-1);
x=mean(lista);
if(nargin==2)
    if n<30
    cvar=var(lista);
    linf=x-cvar*t/sqrt(n);
    lsup=x+cvar*t/sqrt(n);
    else
        linf=x-z*std(lista)/sqrt(n);
        lsup=x+z*std(lista)/sqrt(n);
    end
else
    linf=x-z*desviacion/sqrt(n);
    lsup=x+z*desviacion/sqrt(n);
end
end
```

- `function [linf, lsup]=ic_desv(lista, alpha)`: esta función calcula el intervalo de confianza para la desviación típica con nivel de significación alpha.

```
function [linf,lsup]=ic_desv(lista,alpha)
p=alpha/2;
p1=1-p;
n=length(lista);
chi2inv(p,n-1)
chi2inv(p1,n-1)
z=norminv(p);
x=mean(lista);
v=var(lista);% varianza
    linf=sqrt((n-1)*v/chi2inv(p1,n-1));
```

```
        lsup=sqrt((n-1)*v/chi2inv(p,n-1));
end
```

11. En este problema veremos la diferencia entre usar un modelo simplificado o uno elaborado. Vamos a considerar dos modelos gravitatorios para la Tierra:

- Masa puntual: en este modelo la Tierra se considera como una masa esférica con toda la gravedad concentrada en un único punto.
- Geoide: en este modelo la tierra se considera como un geoide. Se trata de un modelo más realista pero, a su vez, más complejo y costoso en términos de capacidad de cálculo.

La función ***propaga_orbita*** toma una posición inicial x y una velocidad inicial v de una nave espacial alrededor de la Tierra y las propaga durante t unidades de tiempo. El cuarto parámetro es una variable booleana (toma valores 0 o 1) y determina si consideramos la tierra como masa puntual (0) o como geoide (1). La última entrada determina el tipo de salida. Si entramos un 0 nos devolverá el valor en el tiempo final. Si entramos un 1 nos devolverá una lista de posiciones y tiempos (esto nos permitirá dibujar la órbita resultante). Encontrarás las funciones necesarias y sus descriptores en el apartado de documentación en la plataforma.

Para comparar los dos casos vamos a seguir la siguiente secuencia:

- Utilizando la función randn de Matlab, generar tres valores aleatorios r_i siguiendo una gaussiana con media 0 y desviación típica 0.05, define el vector $\vec{r} = (r_1, r_2, r_3)$, y definir la posición inicial como:

$$x = \vec{x}_0 + \vec{r}$$

Con $\vec{x}_0 = (1.5, 0, 0)$. Tomar la velocidad inicial como $v_0 = (0, 1, 0)$.

- Utilizar la función propaga_orbita con masa puntual (cuarto parámetro 0) para generar una predicción de la posición de la nave espacial al cabo de 12 horas. Ten en cuenta que las unidades en las que está programado el sistema son diferentes. Se tienen las equivalencias: 1UD=6378km, 1UT=806.8s.
- Calcula la distancia al centro de la Tierra (el origen) de la nave.
- Repite el experimento 1000 (mil) veces para generar 1000 valores de distancias posibles. Añade las distancias a un vector lista.

```
v=[0,1,0];% vector velocidad
t=12*60*60/806.8; %conversión de 12h a 1UT
for i=1:1000
    r=0.05.*randn(1,3);%vector r aleatorio normalizado
    x=[1.5,0,0]+r;
    xf(i,:)=propaga_orbita(t,x,v,0,0); %1000 vectores
posición modelando la tierra como masa puntual
       xff(i)=6378*sqrt(xf(i,1)^2+xf(i,2)^2+xf(i,3)^2);
%obteniendo magnitud de vector posición
end
```

- Utiliza la función ic_media que has escrito en el ejercicio 1 para calcular un intervalo de confianza para la media para un nivel de significación $\alpha = 0.05$.

```
[mpinf,mpsup]=ic_media(xff,0.05);
%obtener intervalos de confianza de la media de los valores
de posición
```

Intervalo de confianza de la media [2.2617e+04,2.3563e+04]

- Utiliza la función ic_desv que has escrito en el ejercicio 10 para calcular un intervalo de confianza para la desviación típica para un nivel de significación $\alpha = 0.05$.

Intervalo de confianza de la varianza [7.3163e+03,7.9872e+03]

```
[dping,dpsup]=ic_desv(xff,0.05);
%obtener intervalos de confianza de la varianza de los valores de
posición

Repite los pasos a) a d) activando el modelo geoide.
for i=1:1000
    r=0.05.*randn(1,3); %vector r aleatorio normalizado

    x=[1.5,0,0]+r;
    xf2(i,:)=propaga_orbita(t,x,v,1,0);
    %1000 vectores posición modelando la tierra como Geoide
xff2(i)=6378*sqrt(xf(i,1)^2+xf(i,2)^2+xf(i,3)^2);
end
```

- Queremos saber si el cambio de modelo es significativo para el estudio del movimiento de una nave en dicha órbita. Realiza un test de hipótesis para verificar dicha afirmación.
    - La hipótesis nula es que hay igualdad de medias de modelar la tierra como un geoide y como una masa puntual.
    - La hipótesis alternativa es que hay diferencia en la media.

Dado que el intervalo de confianza es 0.05, se ha de rechazar la hipótesis nula si z<-1.96 ó z>1.96.

```
Av0=mean(xff); %valor medio de posiciones mediante la tierra de
```

```
una masa puntual
std0=std(xff);
%valor de la desviación estándar de posiciones mediante la tierra
de una masa puntual
n=1000;
Av1=mean(xff2);
%valor medio de posiciones mediante la tierra como geoide.
z=(Av0-Av1)*sqrt(n)/std0

Av0=2.309e+04
Av1=2.309e+04
Z=0
```

Dado que 0>-1.96 se acepta la hipótesis nula por lo que hay igualdad de medias de modelar la tierra como un geoide y como una masa puntual, es decir no afecta que se modele de una forma o de la otra.

12. Queremos comprobar si una moneda está trucada o no. Para ello vamos a realizar un test de hipótesis de χ^2. Para ello deberemos:

- Simular el lanzamiento de una moneda:
  - Generar un número aleatorio r entre 0 y 1.
  - Transformarlo a 0 (cara) o 1 (cruz). (Podemos asignar 0 si $r < 0.5$ y 1 si $r > 0.5$).

- Realizar 10 000 lanzamientos de la moneda.

```
for i=1:10000
    r(i)=rand;
    if r(i)<0.5
        r(i)=0;
    else if r(i)>=0.5
            r(i)=1;
        end
    end
end
```

- Realizar los siguientes test de hipótesis:
  - Test de hipótesis de un lanzamiento (únicamente valoraremos si es cara o cruz).

Como solo es un lanzamiento y hay solo dos posibilidades la probabilidad es de 50% para ambos casos.

La hipótesis nula $\mathcal{H}0$ representa que la secuencia de monedas no está truncada.

La hipótesis alternativa $\mathcal{H}1$ es que la hipótesis nula es falsa.

```
function [xa,xnk]=chimoneda1(lista)
% xa valor chi cuadrado según el valor de nivel de significancia y
los grados de libertad
% xnk nivel de ajuste
k=2-1;
alpha=0.05;
xa=chi2inv(1-alpha,k);
n1=0;
n0=0;
for i=1:10000
    if lista(i)==1
        n1=n1+1;
    else if lista(i)==0
            n0=n0+1;
        end
    end
end
xnk=(n1-10000*0.5)^2/(10000*0.5)+(n1-10000*0.5)^2/(10000*0.5);
end
```

$x_\propto = 3.8415$, $x[n,k]^2 = 0.7056$, entonces $x_\propto \geq x[n,k]^2$

Se acepta la hipótesis nula porque $x[n,k]^2 \leq x_\propto$ por lo que no es una moneda truncada, sigue la distribución esperada.

- Test de hipótesis de dos lanzamientos (agrupamos los lanzamientos de dos en dos, podemos tener: XX, XO, OX, OO.

  Como es de dos lanzamientos y hay cuatro posibles eventos, la probabilidad es de 25% para todos los casos.

```
function [xa,xnk]=chimoneda2(lista)
%Test de hipótesis para dos lanzamientos de moneda
% xa valor chi cuadrado según el valor de nivel de
significancia y los grados de libertad
% xnk nivel de ajuste
% son 4 agrupaciones por lo que es el 25% de probabilidad para
cada evento
k=4-1;
alpha=0.05;
xa=chi2inv(1-alpha,k);
n11=0;
n10=0;
n01=0;
n00=0;
for i=1:3333
    if eq(lista(2*i-1:2*i),[1,1])
        n11=n11+1;
    end
    if eq(lista(2*i-1:2*i),[1,0])
            n10=n10+1;
    end
    if eq(lista(2*i-1:2*i),[0,1])
            n01=n01+1;
    end
    if eq(lista(2*i-1:2*i),[0,0])
            n00=n00+1;
    end
end

xnk=(n11-5000*0.25)^2/(5000*0.25)+(n10-5000*0.25)^2/(5000*0.25
)+(n01-5000*0.25)^2/(5000*0.25)+(n00-5000*0.25)^2/(5000*0.25);
end
```

$$x_{\infty} = 7.8147 \text{ , } x[n,k]^2 = 1.3168$$

Se acepta la hipótesis nula porque $x[n,k]^2 \leq x_\infty$ por lo que no es una moneda trucada, sigue la distribución esperada.

- Test de hipótesis de tres lanzamientos: XXX, XXO, XOX, XOO, OXX, OXO, OOX, OOO.

Como es de tres lanzamientos y hay ocho posibles suceso y la probabilidad es de 12.5% para todos los casos.

```
function [xa,xnk]=chimoneda3(lista)
%Test de hipótesis para tres lanzamientos de moneda
% xa valor chi cuadrado según el valor de nivel de
significancia y los grados de libertad
% xnk nivel de ajuste
% son 8 agrupaciones por lo que es el 12.5% de probabilidad
para cada evento
k=8-1;
alpha=0.05;
xa=chi2inv(1-alpha,k);
n111=0;
n110=0;
n101=0;
n100=0;
n011=0;
n010=0;
n001=0;
n000=0;
for i=1:3333
    if eq(lista(3*i-2:3*i),[1,1,1])
        n111=n111+1;
    end
    if eq(lista(3*i-2:3*i),[1,1,0])
            n110=n110+1;
    end
    if eq(lista(3*i-2:3*i),[1,0,1])
```

```
                n101=n101+1;
        end
        if eq(lista(3*i-2:3*i),[1,0,0])
                n100=n100+1;
        end
         if eq(lista(3*i-2:3*i),[0,1,1])
            n011=n011+1;
        end
        if eq(lista(3*i-2:3*i),[0,1,0])
                n010=n010+1;
        end
        if eq(lista(3*i-2:3*i),[0,0,1])
                n001=n001+1;
        end
        if eq(lista(3*i-2:3*i),[0,0,0])
                n000=n000+1;
        end
    end
    xnk=(n111-3333*0.125)^2/(3333*0.125)+(n110-3333*0.125)^2/(33
    33*0.125)+(n101-3333*0.125)^2/(3333*0.125)+(n100-3333*0.125)
    ^2/(3333*0.125)+(n011-3333*0.125)^2/(3333*0.125)+(n010-3333*
    0.125)^2/(3333*0.125)+(n001-3333*0.125)^2/(3333*0.125)+(n000
    -3333*0.125)^2/(3333*0.125);
    end
```

$$x_{\infty} = 14.067,\ x[n,k]^2 = 2.8608\,,\ x_{\infty} \geq x[n,k]^2$$

Se acepta la hipótesis nula porque $x[n,k]^2 \leq x_{\infty}$ por lo que no es una moneda trucada, sigue la distribución esperada.

Se confirma que la moneda no es truncada pues en los tres test el resultado es el mismo y señalan este mismo resultado

- En el apartado de documentación encontrarás 3 archivos con 10 000 lanzamientos de moneda cada uno (con nombres moneda1.dad, moneda2.dad y moneda3.dad). Determina si alguna de las tres monedas que se han empleado para generar los lanzamientos ha sido trucada.

Se mantienen tanto la hipótesis nula como la alternativa del primer apartado.

Lanzamientos de Moneda1	Uno	Dos	Tres
$x_{\propto}$	3.8415	7.8147	14.0671
$x[n,k]^2$	376.36	385.2064	422.946
$¿x[n,k]^2 \geq x_{\propto}?$	Si, se rechaza H_0	Si, se rechaza H_0	Si, se rechaza H_0

En los tres escenarios, de uno, dos y tres lanzamientos se rechaza la hipótesis nula por lo que los datos no siguen la distribución esperada entonces la moneda1 es truncada.

Lanzamientos de Moneda2	Uno	Dos	Tres
$x_{\propto}$	3.8415	7.8147	14.0671
$x[n,k]^2$	0	5400	2531
$¿x[n,k]^2 \geq x_{\propto}?$	No, se acepta H_0	Si, se rechaza H_0	Si, se rechaza H_0

En los últimos dos escenarios, para dos y tres lanzamientos se rechaza la hipótesis nula por lo que los datos no siguen la distribución esperada entonces la moneda2 es truncada, aunque en el primer caso haya aparentemente seguido la distribución, pero fueron datos manipulados.

Lanzamientos de Moneda3	Uno	Dos	Tres
x_∞	3.8415	7.8147	14.0671
$x[n,k]^2$	0.4624	2.256	3.0528
$¿x[n,k]^2 \geq x_\infty?$	No, se acepta H_0	No, se acepta H_0	No, se acepta H_0

En todos los escenarios, se acepta la hipótesis nula por lo que los datos siguen la distribución esperada, entonces la moneda3 no es truncada.

Anexos:

- Para determinar el periodo, se desarrolló el siguiente programa en matlab:

```
Index1 = find(vector==vector(1));
Index2 = find(vector==vector(2));
Period = Index1(2) - Index1(1);
if Index1(1)==(Index2(1)-1)
    disp(Period)
end
```

- Para facilitar el trabajo colaborativo código se versiona en GitHub: https://github.com/unir-math-master/actividad_modela

Referencias

1. *Input parser for functions - MATLAB - MathWorks España*. (2020). es.Mathworks.com. https://es.mathworks.com/help/matlab/ref/inputparser.html
2. Alvarez, B. (2013). *AprendBlog » Optional arguments for Matlab functions*. Http://Aprendtech.Com/. http://aprendtech.com/wordpress/software/optional-arguments-for-matlab-functions.html