Guia de Ejercicios - Matlab

1. Dado el vector $x = [3, 15, 9, 12, -1, 0, -12, 9, 6, 1]$, escriba los comandos necesarios para lograr que:

   (a) se deje en 0 los elementos negativos de x. Resultado esperado: $[3, 15, 9, 12, 0, 0, 0, 9, 6, 1]$

   (b) se deje en 3 los elementos que son múltiplos de 3. Resultado esperado: $[3, 3, 3, 3, -1, 3, 3, 3, 3, 1]$

   (c) se multiplica los elementos pares de x por 5. Resultado esperado: $[3, 15, 9, 60, -1, 0, -60, 9, 30, 1]$

   (d) se deje los valores de x mayores que 10 en un nuevo vector. Resultado esperado: $[15, 12]$

   (e) se deje en 0 los elementos de x que son menores que el promedio. Resultado esperado: $[0, 15, 9, 12, 0, 0, 0, 9, 6, 0]$

   Solución. Pueden escribir scripts o funciones que retornan un vector nuevo, o pueden hacer todo en una linea, usando condiciones para especificar cuales son los elementos del vector que van a modificar.

```
function y = preg1a(x)
for i = 1:length(x)
   if x(i) < 0
      y(i) = 0;
   else
      y(i) = x(i);
   end
end
```

```
function y = preg1b(x)
for i = 1:length(x)
   if mod(x(i), 3) == 0
      y(i) = 3;
   else
      y(i) = x(i);
   end
end
```

```
function y = preg1c(x)
for i = 1:length(x)
   if mod(x(i), 2) == 0
      y(i) = 5 * x(i);
   else
      y(i) = x(i);
   end
end
```

```
function y = preg1d(x)
j = 1;
for i = 1:length(x)
   if x(i) > 10
      y(j) = x(i);
      j = j + 1;
   end
end
```

```
function y = preg1e(x)
prom = mean(x);
for i = 1:length(x)
   if x(i) < prom
      y(i) = 0;
   else
      y(i) = x(i);
   end
end
```

```
display(["1.a"])
x = [3, 15, 9, 12, -1, 0, -12, 9, 6, 1]
x(x < 0) = 0
display(["1.b"])
x = [3, 15, 9, 12, -1, 0, -12, 9, 6, 1]
x(mod(x, 3) == 0) = 3
display(["1.c"])
x = [3, 15, 9, 12, -1, 0, -12, 9, 6, 1]
x(mod(x, 2) == 0) = 5 * x(mod(x, 2) == 0)
```

```
display(["1.d"])
x = [3, 15, 9, 12, -1, 0, -12, 9, 6, 1]
y = x(x > 10)
display(["1.e"])
x = [3, 15, 9, 12, -1, 0, -12, 9, 6, 1]
x(x < mean(x)) = 0
```

2. Grafiquen las siguientes funciones usando Matlab y Maple, eligiendo rangos apropiados para x e y. Revisen que los gráficos generados en ambos programas queden similares:

   (a) $f(x) = e^{-x^2}\cos(3x)$

   (b) $f(x) = \frac{1-\frac{3}{5}x+\frac{3}{20}x^2-\frac{1}{60}x^3}{1+\frac{2}{5}x+\frac{1}{20}x^2}$

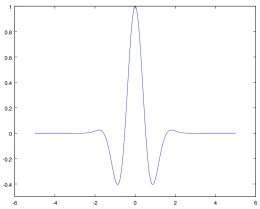
   (c)
   $$f(x) = \sum_{k=0}^{20} \frac{-1^k}{2^{2k}(k!)^2}x^{2k}$$

   Ojo que en este caso, deben calcular el valor de $\sum_{k=0}^{20}\ldots$ para cada valor de x.

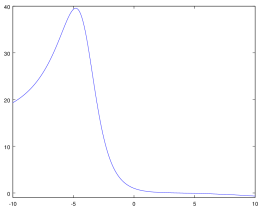
   Solución. En todos los casos deben generar vectores `x`, `y` de coordenadas. Ojo con el uso de `.^`, `.*` y `./`

```
display(["2.a"])
x = -5:0.01:5;
y = exp(-x.^2) .* cos(3 * x);
plot(x, y)
ylim([-0.5, 1])

display(["2.b"])
x = -10:0.05:10;
y = (1 - 3/5*x + 3/20*(x.^2) - 1/60*(x.^3)) ./
    (1 + 2/5*x + 1/20*(x.^2))
plot(x, y)
ylim([-1, 40])
```

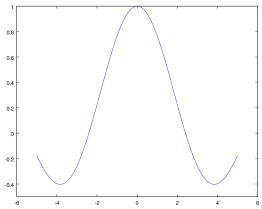
```
function y = preg2c(x)
for i = 1:length(x)
   suma = 0;
      for k = 0:20
         factor = (-1)^k / (2^(2*k) * factorial(k)^2);
         suma = suma + factor * x(i)^(2*k);
      end
   y(i) = suma;
end

% en el workspace
x = -5:0.01:5;
y = preg2c(x);
plot(x, y)
ylim([-0.5, 1])
```



Pregunta 2.a



Pregunta 2.b



Pregunta 2.c

3. Implemente una función que calcule una aproximación de π, usando la siguiente serie:

$$\sum_{i=0}^{\infty} \frac{1}{(2i-1)^2(2i+1)^2}$$

- parámetros de la función: la cantidad de términos a considerar en la aproximación
- valor retornado: el valor aproximado de π

Solución.

```
function pi_aprox = preg3(n)
   pi_aprox = 0;
   for i = 0:n
      pi_aprox = pi_aprox + 1 / ((2*i - 1)^2 * (2*i + 1)^2);
   end
```

```
% en el workspace
>> format long
>> preg3(1000)
ans =  1.11685027504729
>> preg3(10000)
ans =  1.11685027506795
>> preg3(100000)
ans =  1.11685027506795
```

Esta serie claramente no aproxima π. La serie de Leibniz[1] si entrega un valor aproximado de π:

$$4\sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1}$$

```
function pi_aprox = preg3_alt(n)
   pi_aprox = 0;
   for i = 0:n
      pi_aprox = pi_aprox + (-1)^i/(2*i + 1) ;
   end
   pi_aprox = 4 * pi_aprox;
```

```
% en el workspace
>> format long
>> preg3_alt(1000)
ans =  3.14259165433954
>> preg3_alt(10000)
ans =  3.14169264359053
>> preg3_alt(100000)
ans =  3.14160265348972
```

[1] https://en.wikipedia.org/wiki/Leibniz_formula_for_n

4. Los números de Fibonacci se generan usando la siguiente formula:

$$F_n = F_{n-1} + F_{n-2}$$

donde $F_0 = F_1 = 1$. Implemente una función que retorne el n-esimo numero de Fibonacci:

- parámetros de la función: n
- valor retornado: F_n

Solución. Cuidado con los índices de los vectores: en Matlab, el primer elemento de un vector esta en posición `1`, por lo que el n-esimo numero de Fibonacci quedará en la posición `n+1` porque empezamos con F_0.

```
function fib_n = preg4(n)
   fib(1) = 1;
   fib(2) = 1;
   for i = 3:(n+1)
      fib(i) = fib(i-1) + fib(i-2);
   end
   fib_n = fib(n+1);
```

```
>> preg4(3)
ans =  3
>> preg4(6)
ans =  13
>> preg4(10)
ans =  89
```

5. Implemente una función que retorne ciertos valores de un vector. Ejemplo:

```
>> x = 10:-1:1
>> indices = [1, 3, 7]
>> valores = seleccion(x, indices)
valores =
   10   8   4
```

Solución. Si usan un `while`, recuerden actualizar las variables que aparecen en la condición, sino quedaran pegados en un ciclo infinito:

```
function valores = seleccion(x, indices)
   i = 1;
   while i <= length(indices)
      valores(i) = x(indices(i));
      i = i + 1;
   end
```

6. Implemente una función que calcule el coeficiente binomial $\binom{n}{m}$:

- parámetros de la función: n, m
- valor retornado: $\binom{n}{m}$

Solución. No olviden revisar casos borde usando `if`:

```
function coef = preg6(n, m)
   coef = 0;
   if n >= m
      coef = factorial(n)/(factorial(m)*factorial(n-m));
   end
```

```
>> preg6(10, 5)
ans =  252
>> preg6(5, 5)
ans =  1
>> preg6(3, 5)
ans = 0
```

7. Dado un vector $[x_1, x_2, \ldots, x_n]$, podemos calcular el producto "acumulado" con respecto a la posición j de la siguiente forma: $p_j = x_1 \times x_2 \times \cdots \times x_j$. Implemente una función que calcule el producto acumulado de un vector:

- parámetros de la función: vector `x`, posición `j`
- valor retornado: p_j

Solución. Una versión con `for`, otra usando la función `prod`:

```
function pj = preg7(x, j)
   if j < 1 || j > length(x)
      warning('ingrese una posicion valida');
      return;
   end
   pj = 1;
   for i = 1:j
      pj = pj * x(i)
   end
```

```
function pj = preg7_alt(x, j)
   if j < 1 || j > length(x)
      warning('ingrese una posicion valida');
      return;
   end
   pj = prod(x(1:j));
```

8. Implemente una función que cree matrices que tienen la siguiente forma:

```
% Ejemplo: n = 5
     0     1     2     3     4
     1     0     1     2     3
     2     1     0     1     2
     3     2     1     0     1
     4     3     2     1     0

% Ejemplo: n = 8
     0     1     2     3     4     5     6     7
     1     0     1     2     3     4     5     6
     2     1     0     1     2     3     4     5
     3     2     1     0     1     2     3     4
     4     3     2     1     0     1     2     3
     5     4     3     2     1     0     1     2
     6     5     4     3     2     1     0     1
     7     6     5     4     3     2     1     0
```

- parámetros de la función: n
- valor retornado: la matriz generada

Solución. Hay muchas formas de generar estas matrices. Si se fijan, hay una relacion entre las coordenadas de una celda y el valor que queremos guardar ahí. Por ejemplo, queremos dejar el valor 1 en las posiciones (1, 2), (2, 3), (3, 4), ... (diferencia de 1 entre las coordenadas), dejar el valor 2 en las posiciones (1, 3), (2, 4), (3, 5), ... (diferencia de 2 entre las coordenadas), etc. Generalizado:

```
function mat = preg8(n)
        mat = zeros(n, n);
        for i = 1:(n-1)
                j = 1;
                while j + i <= n
                        mat(j, j+i) = i;
                        mat(j+i, j) = i;     % la matriz es simetrica
                        j = j + 1;
                end
        end
```

9. Escriba una función que implemente la criba de Eratóstenes, un algoritmo que permite encontrar todos los números primos menores que un número natural dado n. Pueden encontrar una descripción del algoritmo y ejemplos en Wikipedia: `https://es.wikipedia.org/wiki/Criba_de_Erat%C3%B3stenes`

- parámetros de la función: n
- valor retornado: un vector con los números primos encontrados

Solución.

```
function primos = preg9(n)
        criba = 1:n;
        p = 2;
        while p^2 <= n
                % tachando los multiplos de p
                criba(p^2:p:n) = 0;
                i = 1;
                % encontrar el siguiente primo
                while criba(i) <= p & i < length(criba)
                        i = i + 1;
                end
                p = criba(i);
        end
        % crear el vector de numeros primos en la criba
        j = 1;
        primos = [];
        for i = 1:length(criba)
                if criba(i) > 1
                        primos(j) = criba(i);
                        j = j + 1;
                end
        end
```

10. Escriba una función que convierta números decimales a binario. Pueden encontrar una descripción del algoritmo y ejemplos en Wikipedia: `http://es.wikihow.com/convertir-de-decimal-a-binario`.

    - parámetros de la función: n
    - valor retornado: un vector de 1's y 0's (la representación binaria de n)

    Solución.

```
function binario = preg10(n)
        i = 1;
        while n > 0
                binario(i) = mod(n, 2);
                if binario(i) == 0
                        n = n/2;
                else
                        n = (n-1)/2;
                end
                i = i + 1;
        end
        % ojo que los numeros binarios se escriben desde el ultimo residuo
        binario = binario(length(binario):-1:1);
```

11. El método de bisección es un algoritmo que permite encontrar los ceros de una función f en forma numérica. Se elige un intervalo $[a, b]$, y se determina los valores de $f(a)$ y $f(b)$. Si $f(a)$ y $f(b)$ tienen signos opuestos y f es continua, entonces hay un cero en el intervalo $[a, b]$. En este caso, se calcula m, el punto medio del intervalo, y $f(m)$, el valor de la función en ese punto. Si $f(m) = 0$, encontramos un cero de la función. Sino, seguimos buscando el cero en el intervalo $[a, m]$ o $[m, b]$, dependiendo de los signos de $f(a), f(b)$ y $f(m)$. Este proceso continua, hasta lograr el nivel de precisión deseada (p.ej., el tamaño del intervalo $\leq \mathcal{E}$, donde $\mathcal{E}$ es un parámetro). Pueden encontrar una descripción del algoritmo en Wikipedia: `https://es.wikipedia.org/wiki/M%C3%A9todo_de_bisecci%C3%B3n`

    Implemente una función que use el método de bisección para encontrar ceros de la función $f(x) = x^3 + 2x^2 - 7$

    - parámetros de la función: a, b, $\mathcal{E}$
    - valor retornado: arrojar un "warning" si no hay raíz en el intervalo $[a, b]$, sino retornar el valor de la raíz encontrada

Solución.

```
function m = preg11(a, b, epsilon)
   % primero revisar si hay una raiz en el intervalo dado
   f_a = a^3 + 2*a^2 -7;
   f_b = b^3 + 2*b^2 -7;
   if f_a == 0
      m = a;
      return;
   elseif f_b == 0
      m = b;
      return;
   elseif f_a * f_b > 0
      warning('no hay raiz en ese intervalo')
      return;
   end
   % continua en siguiente columna ..
```

```
   % calcular un valor aproximado de raiz
   while b - a >= epsilon
      m = (a + b)/2;
      f_m = m^3 + 2*m^2 -7;
      f_a = a^3 + 2*a^2 -7;
      if f_m == 0
         break;
      elseif f_m * f_a < 0
         b = m;
      else
         a = m;
      end
   end
```

12. La suma de Riemann se usa para calcular un valor aproximado de la área bajo una curva $f(x)$ en un intervalo $[a, b]$. Para calcular la suma de Riemann, se divide el intervalo en partes iguales, generando rectángulos bajo la curva usando la definición de $f(x)$. La suma de las áreas de estos rectángulos aproxima el valor de la área bajo la curva. Pueden encontrar una descripción del algoritmo en Wikipedia: `https://es.wikipedia.org/wiki/Suma_de_Riemann`

    Implemente una función que calcule la suma de Riemann para la función $f(x) = e^{-x^2}$.

    - parámetros de la función: a, b, numero de rectángulos a considerar
    - valor retornado: la suma de Riemann

```
function suma = preg12(a, b, n)
        dx = (b - a)/n;
        suma = 0;
        for i = 1:n
                % sumando los valores de f(x)
                c = a + i*dx;
                suma = suma + exp(-c^2);
        end
        % multipicar por dx
        suma = dx * suma;
```

```
function suma = preg12_alt(a, b, n)
        dx = (b - a)/n;
        % generar todos los valores x
        x = (a + dx):dx:b;
        % .. y los f(x)
        y = exp(-x .^ 2);
        suma = sum(y) * dx;
```