Algoritmos Numéricos por Computadora

Segundo parcial - Otoño 2020

Sube tu archivo resultado a Canvas → Examen Parcial 2 antes de las 10am.

Cada pregunta vale 5 puntos.

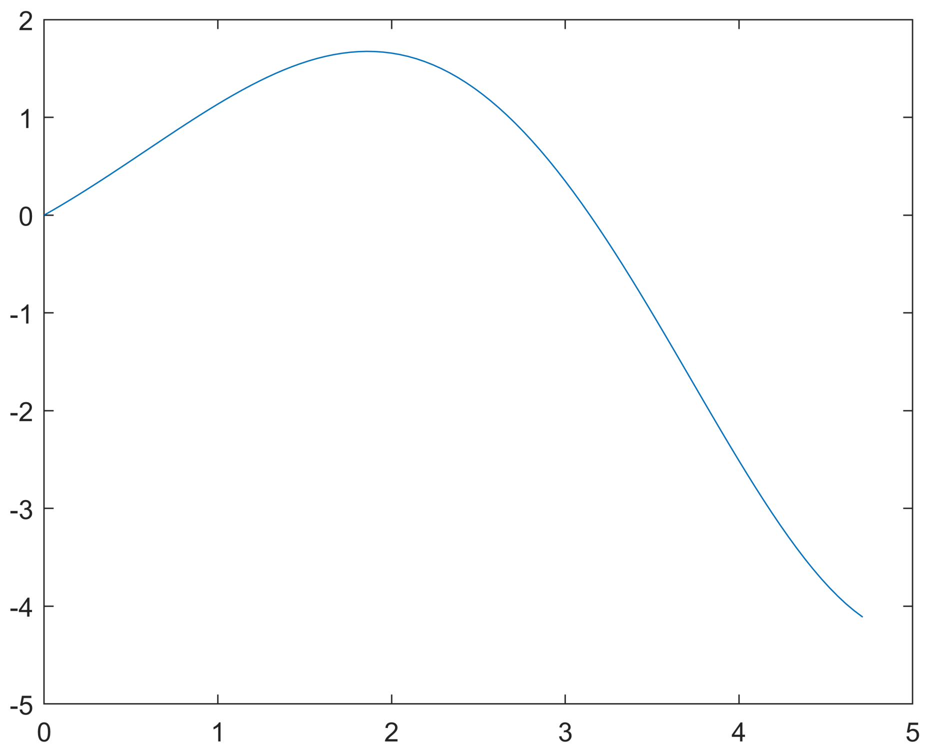
Los exámenes son trabajos individuales. Está estrictamente prohibido dar o recibir ayuda de cualquier persona.

El artículo 29 del Reglamento de Alumnos establece que "se calificará como no acreditada (N.A.) cualquier materia cuando el alumno incurra en alguna práctica fraudulenta".

1. Responde las siguientes preguntas de la manera más eficiente posible.

Dados los vectores *x*, *y*

```
x = linspace(0,3*pi/2);
y = sin(x).*exp(0.3*x);
plot(x,y)  % primero crece y luego decrece
```



1a. Encuentra e imprime los valores máximo y mínimo del vector *y*

```
[val1,indexMax]=max(y)
```

```
val1 = 1.6746
indexMax = 40
```

```
[val2, indexMin]=min(y)
```

```
val2 = -4.1112
indexMin = 100
```

1b. Suma el valor mínimo de *y* a los elementos del vector *y* donde *y* es creciente (los primeros elementos hasta el índice del elemento máximo)

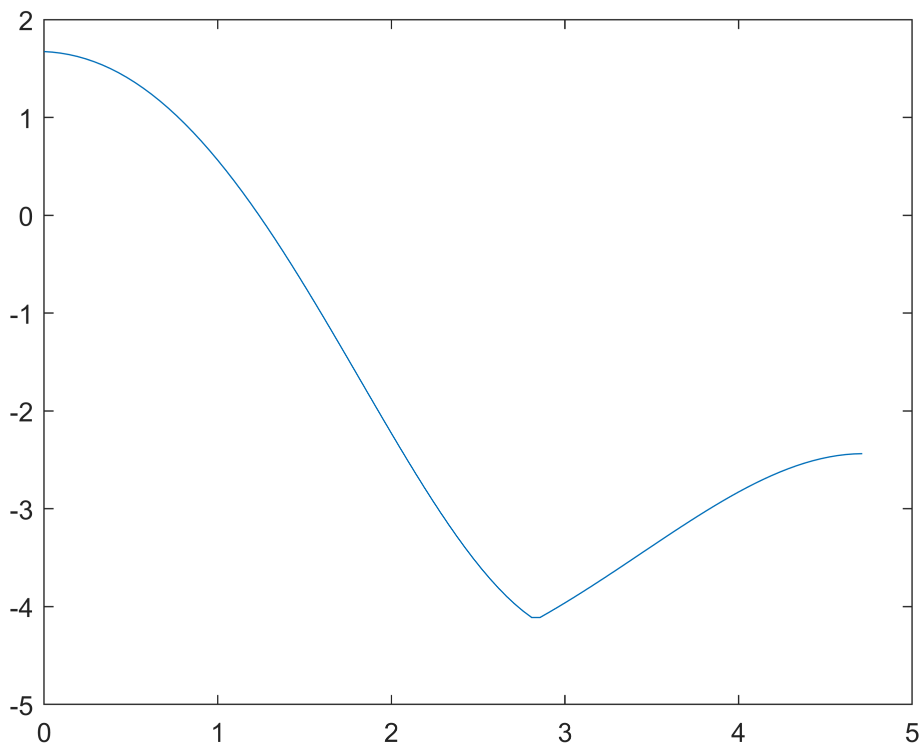
```
y(1:indexMax)=y(1:indexMax)+val2;
```

1c. Intercambia los segmentos del vector *y* donde *y* es creciente y donde es decreciente (de tal manera que el resultado primero decrece y luego crece)

```
%
y=[y(indexMax+1:end) y(1:indexMax)];
```

Graficamos (la gráfica debe ser continua, aunque no diferenciable en un punto)

```
plot(x,y)
```



1d. Multiplica todos los elementos del tercer renglon de A que se encuentran a la derecha de la diagonal principal por los últimos elementos de x. Imprime el escalar resultante.

```
A = magic(5)
```

```
A = 5×5
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
%A la derecha de la diagonal sería A(3,4:5) dado que A es 5 x 5
%Y los ultimos valores de x, serían los ultimos dos para que sea valida la
%operacion
x = [1:5]'
```

```
x = 5×1
     1
     2
     3
     4
     5
```

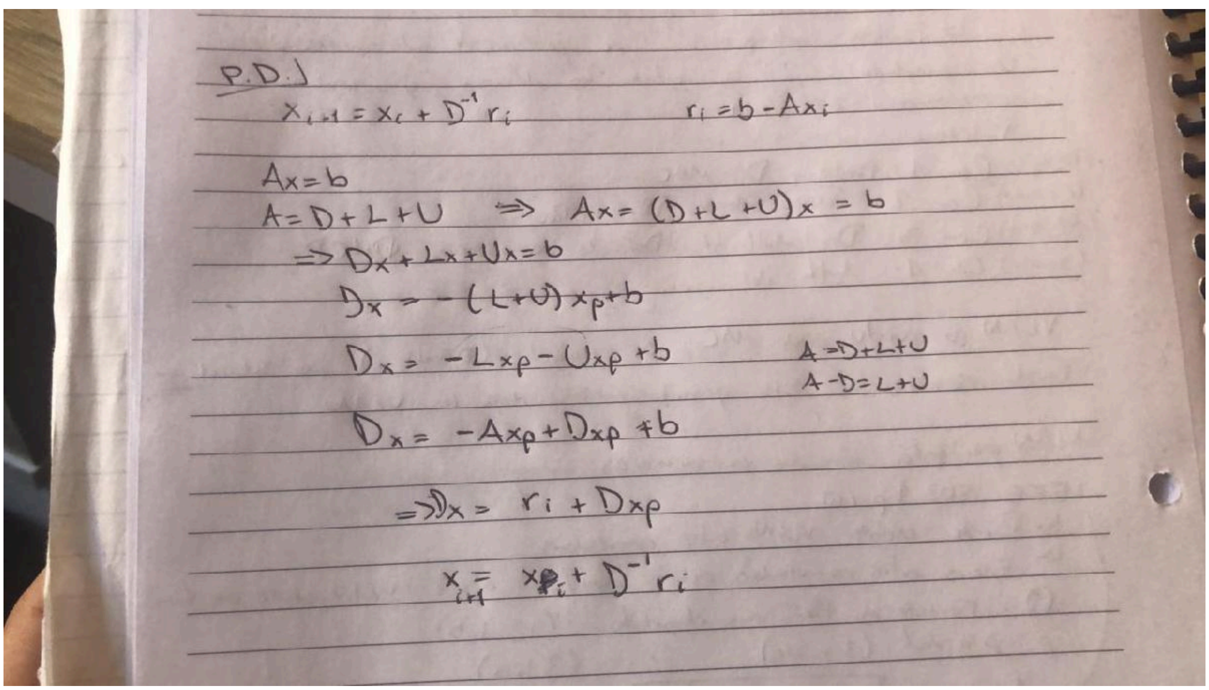
```
r = 3;       % renglón 3
%20*4 + 22*5 = 190
A(3,4:end)*x(4:end)
```

```
ans = 190
```

2. En el método de Jacobi, demuestra que

$x_{i+1} = x_i + D^{-1} r_i \quad$ donde $r_i = b - Ax_i$

Pega tu respuesta aquí.



P.D.)

$x_{i+1} = x_i + D^{-1} r_i \qquad r_i = b - Ax_i$

$Ax = b$

$A = D + L + U \Rightarrow Ax = (D + L + U)x = b$

$\Rightarrow Dx + Lx + Ux = b$

$Dx = -(L+U)x_p + b$

$Dx = -Lx_p - Ux_p + b \qquad A = D + L + U$

$A - D = L + U$

$Dx = -Ax_p + Dx_p + b$

$\Rightarrow Dx = r_i + Dx_p$

$x_{i+1} = x_i + D^{-1} r_i$

3. Considera los siguientes tres conjuntos de ecuaciones lineales:

Set One	Set Two	Set Three
$8x + 3y + z = 13$	$x + y + 6z = 8$	$-3x + 4y + 5z = 6$
$-6x + 8z = 2$	$x + 5y - z = 5$	$-2x + 2y - 3z = -3$
$2x + 5y - z = 6$	$4x + 2y - 2z = 4$	$2y - z = 1$

3a. Identifica un conjunto que, estamos seguros, puede resolverse usando el método de Gauss-Seidel.

```
%Set 1
A=[8,3,1; -6,0,8; 2,5,-1]
```

```
A = 3×3
     8     3     1
    -6     0     8
     2     5    -1
```

```
b=[13;2;6]
```

```
b = 3×1
    13
     2
     6
```

```
P=[1,0,0;0,0,1;0,1,0]
```

```
P = 3×3
     1     0     0
     0     0     1
     0     1     0
```

```
%Reacomodamos
A=P*A
```

```
A = 3×3
     8     3     1
     2     5    -1
    -6     0     8
```

```
b=P*b
```

```
b = 3×1
    13
     6
     2
```

3b. ¿Por qué estamos seguros que el método converge con este conjunto?

```
%Porque es Diagonalmente dominante y por lo tanto converge, usaremos la
%funcion que se declaró abajo
dominanteDiagonal(A)
```

```
ans = logical
   1
```

3c. Resuelve el conjunto identificado usando el método de Gauss-Seidel.

```
%
[x,i]=gaussSeidelMethod(A,b,1)
```

```
x = 3×1
    1.1250
    0.9688
    1.0938
i = 28
```

3d. Verifica que tu solución es correcta.

```
%A*x=b, por lo tanto, A*x-b debería ser 0 en caso de que los resultados
%sean correctos
A*x-b
```

```
ans = 3×1
     0
     0
     0
```

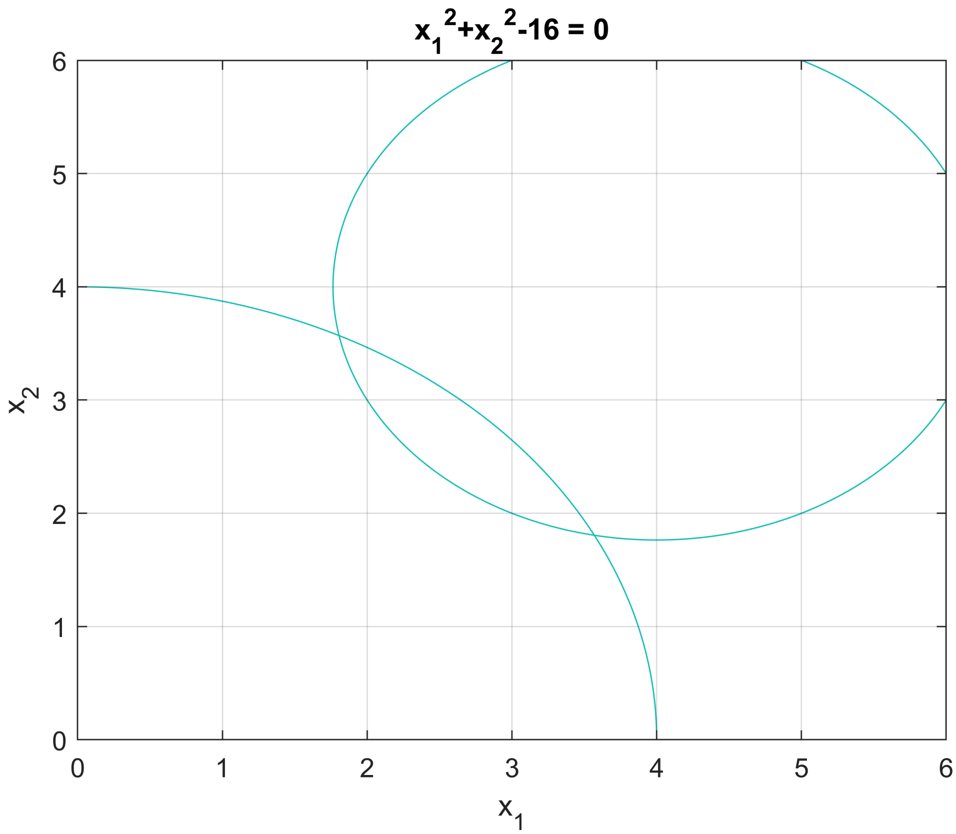
4. Determina las dos raíces de las siguientes ecuaciones simultaneas (intersección de dos circunferencias):

(x − 4)^2 + (y − 4)^2 = 5

x^2 + y^2 = 16

4a. Usa un enfoque gráfico (en el intervalo [0,6]) para obtener tus aproximaciones iniciales.

```
%
f1 = @(x1,x2) (x1-4).^2 + (x2-4).^2 -5;
f2 = @(x1,x2) x1.^2+x2.^2 -16;
ezplot(f1,[0,6])
grid on
hold on
ezplot(f2,[0,6])
hold off
```



4b. Escribe la función f y la matriz J del método de Newton-Raphson

```
%
f = @(x)[(x(1)-4)^2+(x(2)-4)^2-5;
         x(1)^2+x(2)^2-16]
```

```
f = function_handle with value:
    @(x)[(x(1)-4)^2+(x(2)-4)^2-5;x(1)^2+x(2)^2-16]
```

```
J=@(x)[2*(x(1)-4), 2*(x(2)-4);
    2*x(1), 2*x(2)]
```

```
J = function_handle with value:
    @(x)[2*(x(1)-4),2*(x(2)-4);2*x(1),2*x(2)]
```

4c. Usa el método de Newton-Rapshon para encontrar una raíz.

```
%
NewtonRaphsonVV(f,[4;1])
```

```
ans = 2×1
    3.5692
    1.8058
```

4d. Usa el método de Newton-Rapshon para encontrar la otra raíz.

```
%
NewtonRaphsonVV(f,[1;4])
```

```
ans = 2×1
```

```
    1.8058
    3.5692
```

5. Usa la factorización de Cholesky para resolver un sistema de ecuaciones lineales.

5a. Comprueba que la matriz A es definida positiva (simétrica y con todos sus valores propios positivos).

```
A = [2,-2,-3; -2,5,4;-3,4,5]
```

```
A = 3×3
     2    -2    -3
    -2     5     4
    -3     4     5
```

```
[m,n]=size(A);
if m==n & A == A' & eig(A)>0 %cuadrada, simetrica y eigenvalores positivos
        bol=true
else
        bol=false
end
```

```
bol = logical
   1
```

5b. Factoriza la matriz A usando la función cholesky del problema 6. (Plan B, utiliza la función chol(A,'lower') de Matlab.)

```
%
L=cholesky(A)
```

```
L = 3×3
    1.4142         0         0
   -1.4142    1.7321         0
   -2.1213    0.5774    1.0000
```

5c. Verifica (visualmente o numéricamente) que L*L' = A

```
%
L*L'
```

```
ans = 3×3
    2.0000   -2.0000   -3.0000
   -2.0000    5.0000    4.0000
   -3.0000    4.0000    5.8333
```

```
A
```

```
A = 3×3
     2    -2    -3
    -2     5     4
    -3     4     5
```

5d. Resuelve el sistema de ecuaciones $A\,x = b$ usando la matriz L (y dos sustituciones, puedes usar \ aquí).

```
b = [7;-12;-12];
%A*x=b => L*L'*x=b => x=L'\L\b
x=L'\(L\b)
```

```
x = 3×1
    2.0278
   -1.7222
    0.1667
```

```

```

```
ans = 3×1
   -0.0000
         0
   -0.1389
```

6. La factorización o descomposición de Cholesky toma su nombre del matemático André-Louis Cholesky, quien encontró que una matriz simétrica definida positiva puede ser descompuesta como el producto de una matriz triangular inferior ***L*** (con elementos positivos en la diagonal) y su traspuesta (Wikipedia).

Codifica aquí, de manera **estructurada** y **eficiente**, la función L=cholesky(A) que utiliza el algoritmo de Cholesky-Crout para calcular ***L*** columna por columna, iniciando en la esquina superior izquierda.

$$L_{j,j} = +\sqrt{A_{j,j} - \sum_{k=1}^{j-1} L_{j,k}^2},$$
$$L_{i,j} = \frac{1}{L_{j,j}}\left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k}L_{j,k}\right) \quad \text{for } i > j.$$

Tips para optimizar la codificación (después de que funcione):

a) Las sumatorias de productos pueden calcularse sin usar un for

b) El producto punto de dos vectores permite simplificar el cálculo de las sumatorias de productos.

c) En cada columna j, todos los elementos de ***L*** bajo la diagonal (i > j) pueden calcularse sin usar un for.

```
function L = cholesky(A)
    % Entrada: A - matriz definida positiva
    % Salida:  L - matriz triangular inferior tal que L*L'= A

    [m,n]=size(A);
    if ~(m==n & A == A' & eig(A)>0), error('Matriz debe ser definida positiva'), end

    L=zeros(size(A));
    for j=1:m
        L(j,j)=+sqrt(A(j,j)-dot(L(1:j-1),L(1:j-1)));
        for i=j+1:m
            L(i,j)=(1/L(j,j))*(A(i,j)-dot(L(i,1:j-1),L(j,1:j-1)));
        end
    end
end
```

```
function bool=dominanteDiagonal(A)
    [n,m] = size(A);
    if n~=m
        error('A no es una matriz cuadrada.')
    end
    i = 1;
    bool = true;
    while i<=n && bool
        bool = 2*abs(A(i,i)) > sum(abs(A(i,:)));
        i = i+1;
    end
end
function [x,i]=gaussSeidelMethod(A,b,lambda)
    if ~dominanteDiagonal(A),warning('La matriz no es dominante diagonal'),end
    [n,~] = size(A);
    if(n~=size(b)), error('Las dimensiones no son iguales'), end
    D = diag(diag(A));
    U = triu(A,1);
    L = tril(A,-1);
    x = zeros(size(b));
    MAX = 1000;
    i = 1;
    cond = true;
    while cond
        xp = x;
        x(1) = (b(1)- U(1,2:n)*xp(2:n))/D(1,1);
        for j=2:n-1
            x(j) = (b(j) - (L(j,1:j-1)*x(1:j-1)+U(j,j+1:n)*xp(j+1:n)))/D(j,j);
        end
        x(n) = (b(n) - L(n,1:n-1)*x(1:n-1))/D(n,n);
        i = i+1;
        x=lambda*x+(1-lambda)*xp;
        cond = norm((x-xp)./x, Inf) > eps && i<MAX;
    end
end

function [x,i] = NewtonRaphsonVV(f,x0)
    [m,~] = size(x0);
    xs = sym('x',[1 m]);

    fx = f(xs);
    Js= jacobian(fx,xs);

    x = x0;
    MAX = 100;
    cond = true;
    i = 0;
    while cond
```

```
        xp = x;
        J  = double(subs(Js,xs,xp'));
        x = xp - J\f(xp);
        i = i+1;
        cond = norm((x-xp)./x,Inf) >eps && i<MAX;
    end
end
```