Asignatura	Datos del alumno	Fecha
Métodos Numéricos I	**Apellidos:** Avilés Cahill	27/02/2024
	Nombre: Adán	

Laboratorio: Una capa para Gastón

1. Actividad 1

Una de las medidas que Gastón necesita conocer es la longitud de la curva en azul de la figura 2, anexada en los apuntes. Sabemos que la longitud de una curva $f(x)$ viene determinada por

$$L = \int_a^b \sqrt{1 + (f'(x))^2} dx \tag{1}$$

Se nos dice que $f(x) = \sqrt{R^2 - x^2}$, con límites de integración $\frac{-R}{2}$ y $\frac{9R}{10}$. Derivando la función inicial, obtenemos $f'(x) = \frac{x}{\sqrt{R^2-x^2}}$. Así, es fácil ver que:

$$L = \int_{\frac{-R}{2}}^{\frac{9R}{10}} \sqrt{1 + (\frac{x}{\sqrt{R^2 - x^2}})^2} dx = \int_{\frac{-R}{2}}^{\frac{9R}{10}} \frac{dx}{\sqrt{1 - (\frac{x}{R})^2}} \tag{2}$$

Tomamos $R = 56$ y calcularemos la longitud de la curva con los métodos de trapecios con $n = 8$, Simpson con $n = 8$ y Gauss-Legendre con $n = 3$. Como sabemos que $R = 56$, podemos sustituir R por su valor. Definimos en Matlab las variables y la función:

```
format long g
a = (-56)/2
b = (9*56)/10
L = @(x) 1./(sqrt(1-(x/56).^2))
```

Calculamos ahora, usando los códigos que hemos dejado en 4.1, 4.2 y 4.3 los valores de la Integral:

```
% Para las  fórmulas de NewtonCotes
n = 8
I_trapecios = trapecios(L,a,b,n)
I_simpson = simpson(L,a,b,n)
```

Obteniendo como resultado:

```
I_trapecios = trapecios(L,a,b,n)
I_simpson = simpson(L,a,b,n)
n = 8
I_trapecios = 93.51360290246893
I_simpson = 92.31957331439618
```

Figura 1: Resultado de las fórmulas de trapecio y Simpson

```
% Para  de GaussLegendre
n = 3
I_gausslegendre = gausslegendre(L, n, a, b)
```

cuyo resultado es:

```
I_gausslegendre = 91.23664791989698
```

Figura 2: Resultado de Gauss-Legendre

2. Actividad 2

Ahora, hemos de calcular el área de la figura, que obtenemos como

$$A = \int_{\frac{-R}{2}}^{\frac{9R}{10}} f(x) = \sqrt{R^2 - x^2} dx \tag{3}$$

Tomamos $R = 56$ y calcularemos el área con los métodos de trapecios con $n = 8$, Simpson con $n = 8$ y Gauss-Legendre con $n = 3$.

```
% Para las  fórmulas de NewtonCotes
f = @(x) sqrt(56^2-x.^2)
a = (-56)/2
b = (9*56)/10
n = 8
Area_trapecios = trapecios(f,a,b,n)
Area_simpson = simpson(f,a,b,n)
```

con resultado:

```
a = (-56)/2
b = (9*56)/10
n = 8
Area_trapecios = trapecios(f,a,b,n)
Area_simpson = simpson(f,a,b,n)
f =

@(x) sqrt (56 ^ 2 - x .^ 2)

a = -28
b = 50.40000000000000
n = 8
Area_trapecios = 3850.259702330668
Area_simpson = 3869.755080799235
```

Figura 3: Área usando Trapecios y Simpson

y para el método de Gauss-Legendre:

```
% Para las  fórmulas de NewtonCotes
n = 3
Area_gausslegendre = gausslegendre(f, n, a, b)
```

que nos devuelve

```
Area_gausslegendre =

          3875.08048973802
```

Figura 4: Área usando Gauss-Legendre.

3. Actividad 3

Finalmente, hemos de calcular el volumen del paraguas que usará Gastón en la visita de los Reyes Caléndula. Sabemos que el volumen se puede calcular como

$$V = \int\int_{\sigma} g(x,y)dxdy \tag{4}$$

Donde $g(x,y) = +\sqrt{R^2 - x^2 - y^2}$, donde σ es el círculo de radio R. Si hacemos el cambio de variables $x = \rho\cos(\varphi)$ e $y = \rho\sin(\varphi)$, por el teorema de cambio de variables y calculando el jacobiano

$$\mathbf{J}_{x,y} = \begin{bmatrix} \frac{\partial x}{\partial \rho} & \frac{\partial x}{\partial \varphi} \\ \\ \frac{\partial y}{\partial \rho} & \frac{\partial y}{\partial \varphi} \end{bmatrix} = \rho\cos^2(\varphi) + \rho\sin^2(\varphi)) = \rho$$

Por tanto,

$$V = \int_0^R \int_0^{2\pi} \rho\sqrt{R^2 - \rho^2} d\varphi d\rho \tag{5}$$

Tomando $R = 56$, calculamos el volumen del paraguas utilizando el método de Trapecios con $n = m = 8$, el método de Simpson con $n = m = 8$ y el método de Gauss-Legendre con $n = 3$. Empezamos con los métodos del trapecio y de Simpson:

```
% Definimos la funcion y las variables
g = @(x,y) x.*sqrt(56^2-x.^2);
a =  0; b = 56;
c = 0; d = 2*pi;
n = 8; m= 8;
vol_trapecios = trapecios_varias_variables_optimizadas(g, a, b, c, d, n, m)
vol_simpson = simpson_varias_variables_optimizada(g, a, b, c, d, n, m)
```

obteniendo así:

y para el caso de Gauss-Legendre:

```
% Definimos la funcion y las variables
```

```
vol_trapecios = 352311.3402183482
vol_simpson = 362058.2582302729
```

Figura 5: Volumen usando trapecios y Simpson.

```
n = 3;
vol_gausslegendre = gausslegendre_varias(g, n, a, b, c, d)
```

que nos devuelve

```
vol_gausslegendre =

          372107.237663042
```

Figura 6: Volumen usando Gauss-Legendre.

4. Anexos: Código

4.1. Código Trapecios

```
function I = trapecios(f,a,b,n)
    % I = trapecios(f,a,b,n) devuelve la integral de f(x)
    % con la fórmula de trapecios compuesta.
    h = (b-a)/n; %calculo paso h
    x = a:h:b; % calculo los pasos de a b
    pesos = [1 2*ones(1, n-1) 1]; %Tal y como aparece en la formula. 1 2 2 ...1
    I = (h/2)*sum(pesos.*f(x)); %añado el . para multiplicar vectores
end
```

4.2. Código Simpson

```
function I = simpson(f,a,b,n)
    % I=simpson(f,a,b,n) obtiene la integral de f(x)
    % con la fórmula de Simpson compuesta.
    h = (b-a)/n; %calculo paso h
    x = a:h:b; % calculo los pasos de a b
    pesos = ones(1, n+1) ; % Devuelve un array de n+1 unos
    pesos(2:2:n) = 4; % Hago que los pares sean 4
    pesos (3:2:n-1) = 2; %Hago que los impares sean 2
    I = (h/3)*sum(pesos.*f(x)); %añado el . para multiplicar vectores
end
```

4.3. Código Gauss-Legendre

Lo ideal sería poner los parámetros a y b como opcionales (en *Octave* solo tendriamos que poner $a = -1$, $b = 1$) para que en caso de que no se pasen por valores, se tomen automáticamente.

```
function I  = gausslegendre(f, n, a, b)
    % xi ci I  = gausslegendre(n) obtiene los
    % coeficientes ci y nodos xi de la cuadratura de
    % Gauss-Legendre para un n dado
    syms x%Creo una variable simbolica para los polinomios
    p{1} = 1; %Creo p_0, p_1.
    p{2} = x;
    %Calculo de forma recursiva los pols legendre
    for k = 1:n-1
        p{k+2} = (1/(k+1))*((2*k+1)*x*p{k+1}-k*p{k});
    end
    %Resuelvo la ecuación
    xi = double(solve(p{n+1}==0,x));
    derpn = diff(p{n+1}); %derivada del polinomio
    ci = double(2./((1-xi.^2).*(subs(derpn,x,xi)).^2));
    valores_funcion = f(xi*(b-a)/2 + (b+a)/2);
    I = ((b-a)/2)*sum(ci.*valores_funcion);
end
```

4.4. Código Varias Variables Trapecios

El código tiene la muletilla de *optimizadas* pues en un primer lugar, usaba dos bucles *for* para iterar por los distintos $f(x_i, y_k)$ haciéndolo mucho menos eficiente, esto se puede evitar utilizando la traspuesta de y y tomándola de menor a mayor.

```
function I = trapecios_varias_variables_optimizadas(f, a, b, c, d, n, m)
    % Devuelve la integral de la funcion F
    % [a,b] = limites para x
    % [c,d] = limites para y
    % n = intervalos para x
    % n = intervalos para y
    h = (b-a)/n; %calculo paso h
    k = (d-c)/m; %calculo paso h
    x = a:h:b; % calculo los pasos de a b
    y = d:-k:c; % calculo los pasos de c d
    % Crearé la matriz de pesos
    pesos_x = [1 2*ones(1, n-1) 1]; %Matriz fila
    pesos_y = [1 2*ones(1, m-1) 1]'; %Matriz columna
    pesos = pesos_y*pesos_x; %Matriz de pesos
    matriz_valores = f(x,y');
    % Tendré 1f(x0,ym) 2f(x2,ym)....1f(xn,ym)
    % ....   2f(x0,ym-1) 4(x1,ym-2)....
    % Para finalmente sumar todos los valores
    I = ((h*k)/4)*sum(sum(pesos.*matriz_valores));
    %I = ((h*k)/4)*sum(pesos.*matriz_valores, 'all'); %añado el . para multiplicar vec
end
```

4.5. Código Varias Variables Simpson

```
function I = simpson_varias_variables_optimizadas(f, a, b, c, d, n, m)
    h = (b-a)/n;
    k = (d-c)/m;
    x = a:h:b; % calculo los pasos de a b
    y = d:-k:c;
    pesos_x = ones(1, n+1);   pesos_y = ones(1, n+1)';
    pesos_x(2:2:n) = 4; pesos_y(2:2:n) = 4;
    pesos_x(3:2:n-1) = 2; pesos_y(3:2:n-1) = 2;
    pesos = pesos_y*pesos_x; % Matriz de pesos
    % Hace todas las permutaciones f(x_i,y_j)
    matriz_valores = f(x,y');
    I = ((h*k)/9)*sum(sum(pesos.*matriz_valores));
end
```

4.6. Código Varias Variables Gauss-Legendre

```
function I = gausslegendre_varias(f, n, a, b, c, d )
    % xi, ci = gausslegendre(n) obtiene los
    % coeficientes ci y nodos xi de la cuadratura de
    % Gauss-Legendre para un n dado
    syms x %Creo una variable simbolica para los polinomios
    p{1} = 1; %Creo p_0, p_1.
    p{2} = x;
    for k = 1:n-1
        p{k+2} = (1/(k+1))*((2*k+1)*x*p{k+1}-k*p{k}); %Calculo de forma recursiva los p
    end
    xi = double(solve(p{n+1}==0,x)); %resuelvo la eq
    derpn = diff(p{n+1});
    ci = double(2./((1-xi.^2).*(subs(derpn,x,xi)).^2));
    %%% Comienzo la parte de varias variables
    C = ci*ci'; u=xi; v=xi'; %trasponemos v
    U = repmat(u,1,n);
    V = repmat(v,n,1);
    valor_nuevo_U = U*(b-a)/2 + (b+a)/2;
    valor_nuevo_V = V*(d-c)/2 + (d+c)/2;
    parametro_U = (b-a)/2;
    parametro_V = (d-c)/2;
    valores_funcion = f(valor_nuevo_U,valor_nuevo_V);
    I = parametro_U*parametro_V*sum(sum(C.*valores_funcion));
end
```