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

Laboratorio: Optimización sin restricciones

Índice

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

1. Introducción

El objetivo para esta entrega consiste en encontrar la curva que une los puntos $(0,1)$ y $(1,1)$ de forma que se minimice el área de la superficie de revolución alrededor del eje x, es decir, hemos de minimizar

$$I(u) = \int_0^1 u(x)\sqrt{1+u'(x)^2}dx$$

Siendo $u(0) = u(1) = 1$.

1.1. Metodología

En primer lugar, discretearemos el intervalo $[0,1]$ en en intervalos de la forma $[\frac{j}{n}, \frac{j+1}{n}]$. En segundo lugar, recordamos que si consideramos una función $f(x)$, la integración por trapecios nos da la fórmula:

$$\int_0^1 f(x)dx = \sum_{j=0}^{n-1} \frac{f(x_j)+f(x_{j+1})}{2}\frac{1}{n} = \sum_{j=0}^{n-1} T_j(f(x))$$

En nuestro caso, $f(x) = u(x)\sqrt{1+u'(x)^2}$. Seguidamente, definimos $u(x_j) = u_j$, y como estamos aproximando $u(x)$ por una poligonal, podemos tomar las derivadas por la derecha y por la izquierda para tener la casi igualdad

$$u^{'}(x_j) \approx u^{'}(x_{j+1} \approx \frac{u_{j+1}-u_j}{\frac{1}{n}} = (u_{j+1}-u_j)\cdot n$$

Por tanto, la aproximación de la integral inicial vendrá dada por

$$I(u) \approx T(u) = \sum_{j=0}^{n-1} T_j(u)$$

Con

$$T_j(u) = \frac{(u_{j+1}+u_j)\sqrt{1+n^2(u_{j+1}-u_j)^2}}{2n}$$

Gracias a esto, hemos conseguido una función F que va de $\mathbf{R}^{n+1}$ en $\mathbf{R}$, que podremos optimizar utilizando el método del gradiente.

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

2. Resolución

2.1. Calcular el gradiente.

Para calcular el gradiente, desarrollaremos $T(u)$ para que nos sea más cómodo.

$$T(u) = \frac{1}{2n}[(u_1+u_0)\sqrt{1+n^2(u_1-u_0)^2} + (u_2+u_1)\sqrt{1+n^2(u_2-u_1)^2} + \cdots + (u_{n-1}+u_{n-2})\sqrt{1+n^2(u_{n-1}-u_0)^{n-2}} + (u_n+u_{n-1})\sqrt{1+n^2(u_n-u_{n-1})^2}]$$

Si derivamos con respecto a u_0 obtendremos

$$g_0 = \frac{\partial T(u)}{\partial u_0} = \frac{1}{2n}(\sqrt{1+n^2(u_1-u_0)^2} - (u_1+u_0)\frac{n^2(u_1-u_0)}{\sqrt{1+n^2(u_1-u_0)^2}})$$

Análogamente para $j = n$,

$$g_n = \frac{\partial T(u)}{\partial u_n} = \frac{1}{2n}(\sqrt{1+n^2(u_n-u_{n-1})^2} - \frac{n^2(u_n-u_{n-1})(u_n+u_{n-1})}{\sqrt{1+n^2(u_n-u_{n-1})^2}})$$

A diferencia de $j = 0$ y $j = n$, el resto de u_i con $j = 1, \ldots, n-1$ aparecen en dos términos, por tanto su derivada parcial será diferente:

$$g_i = \frac{\partial T(u)}{\partial u_i} = \frac{1}{2n}(\sqrt{1+n^2(u_i-u_{i-1})^2} + \frac{n^2(u_i+u_{i-1})(u_i+u_{i-1})}{\sqrt{1+n^2(u_i-u_{i-1})^2}} + \cdots + \sqrt{1+n^2(u_{i+1}-u_i)^2}) - \frac{n^2(u_{i+1}+u_i)(u_{i+1}+u_i)}{\sqrt{1+n^2(u_{i+1}-u_i)^2}})$$

Y podemos definir el gradiente como

$$\nabla F = (g_0, g_1, \ldots, g_n)$$

.

2.2. Calcular al curva óptima para distintos valores de n.

Para resolver este ejercicio, hemos de optimizar nuestros u_j de tal forma que minimicen la función anterior, para ello, planteamos

$$u_{k+1} = u_k - \alpha \cdot \nabla F(u_k)$$

Como punto de inicio, tomaremos $u_0 = [1, 1, \ldots, 1]$, que es la línea recta y curva más simple que une los puntos dados. En este caso, obtenemos la gráfica:

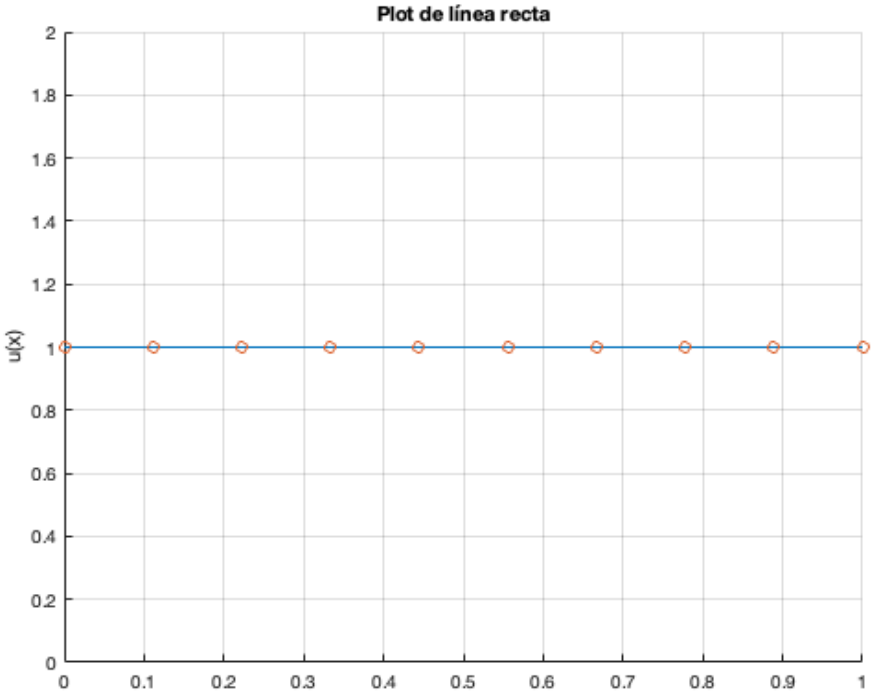

Figura 1: Línea recta, caso básico.

Usaremos ahora el gradiente para optimizar, utilizando el siguiente código:

```
tol = 1e-5;
maxiter = 1000;
alpha = 0.15;
iter= 0;
numb_points=10;
u0 = ones(1,numb_points);
[fx,dfx] = fun(u0);
while norm(dfx)>tol && iter<maxiter
    %[alpha, ~, ~] = budi(@(alpha)fun_alpha(alpha,x0,dfx), 0, 0.5, tol, maxiter);
    u0 = u0-alpha*dfx;
    u0(1)=1; u0(end)=1;
    [fx,dfx] = fun(u0);
    iter= iter+1;
end
```

Podemos ver que tenemos establecido $alpha = 0{,}15$, pero también podemos buscar el óptimo descomentando la linea de *budi*. Tenemos el parámetro *numb_points* que nos sirve para variar el n del problema. Además, se puede ver que forzamos que $u_0 = u_n = 1$ para cumplir con las restricciones. La función *fun* contiene f y df definidas, y se encuentra definida en el anexo, junto a *fun_alfa* Probemos ahora a optimizar para $alfa = 0{,}15$, $n = 5$:

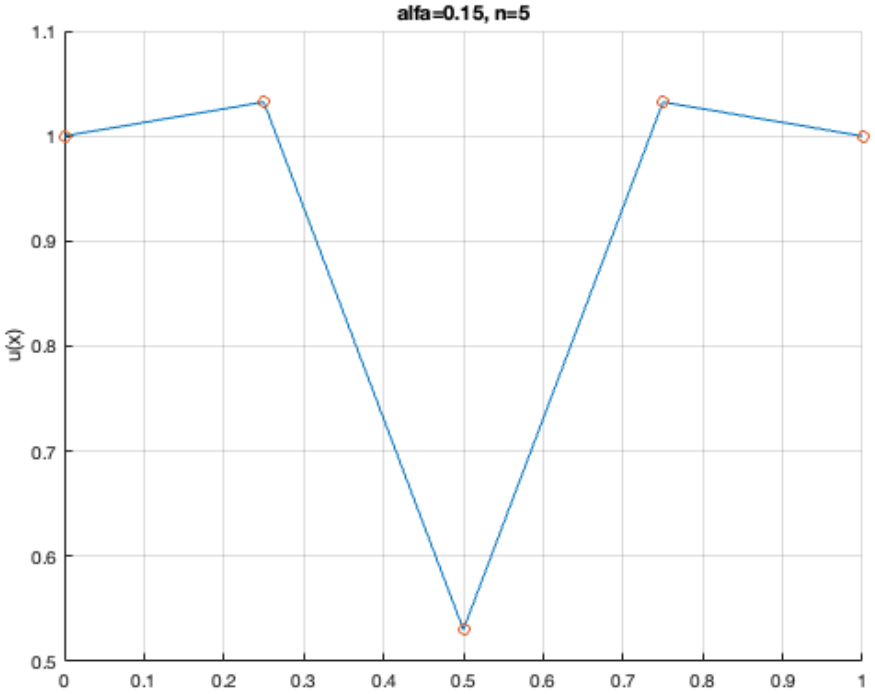

Figura 2: alfa=0.15, n=5

Si tomamos la norma de $f(u)$, aproximará a la integral, en este caso tenemos

$$\|f(u_n)\| = 1{,}2597$$

Tomando $alfa = 0{,}15$, $n = 10$:

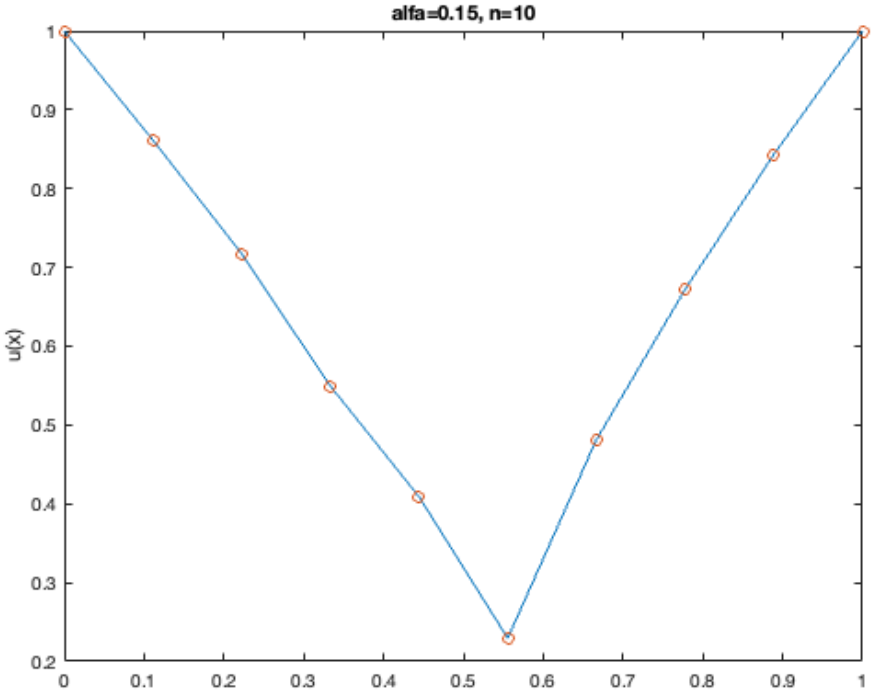

Figura 3: alfa=0.15, n=10

$$\|f(u_n)\| = 0{,}7281$$

Tomando $alfa = 0{,}15$, $n = 15$::

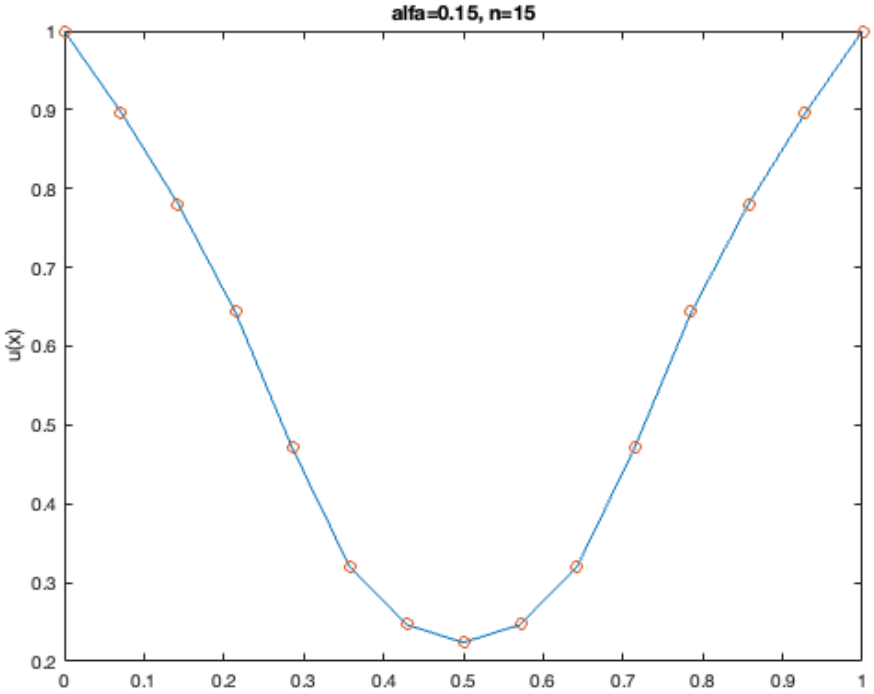

Figura 4: alfa=0.15, n=15

$$\|f(u_n)\| = 0{,}6309$$

Podría parecer que incrementar n hace que tengamos una mejor aproximación, pero puede ocurrir que no converja, fijémonos en el caso de tomar $n = 30$

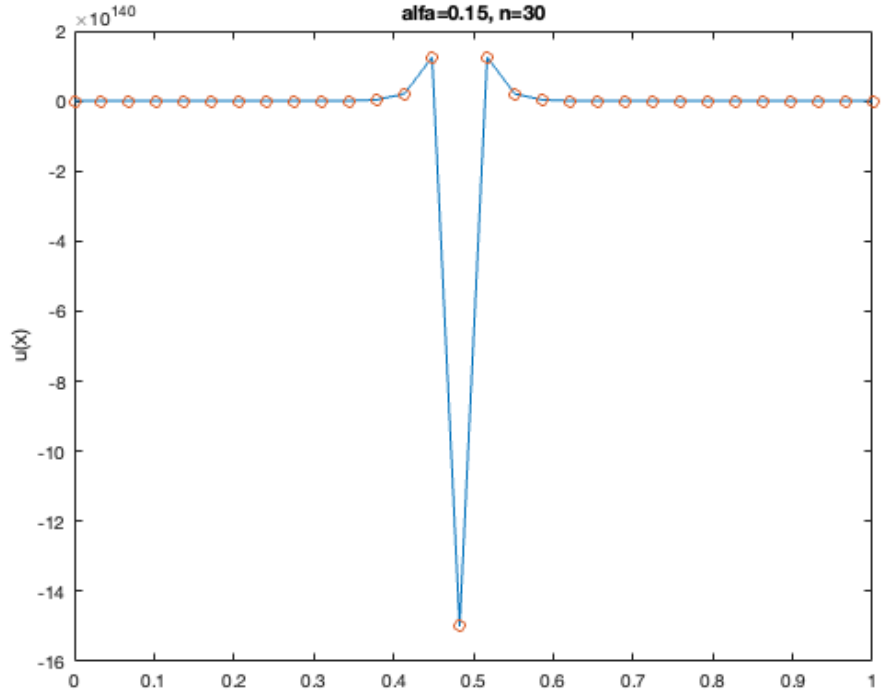

Figura 5: alfa=0.15, n=30

Podemos ver que diverge, además la norma se dispara. En este caso, hemos de ser cuidadosos de tomar un alfa adecuado. De hecho, para un mismo n, cambiando el alfa podemos obtener mejores resultados también (fijado un número máximo de iteraciones, claro está)

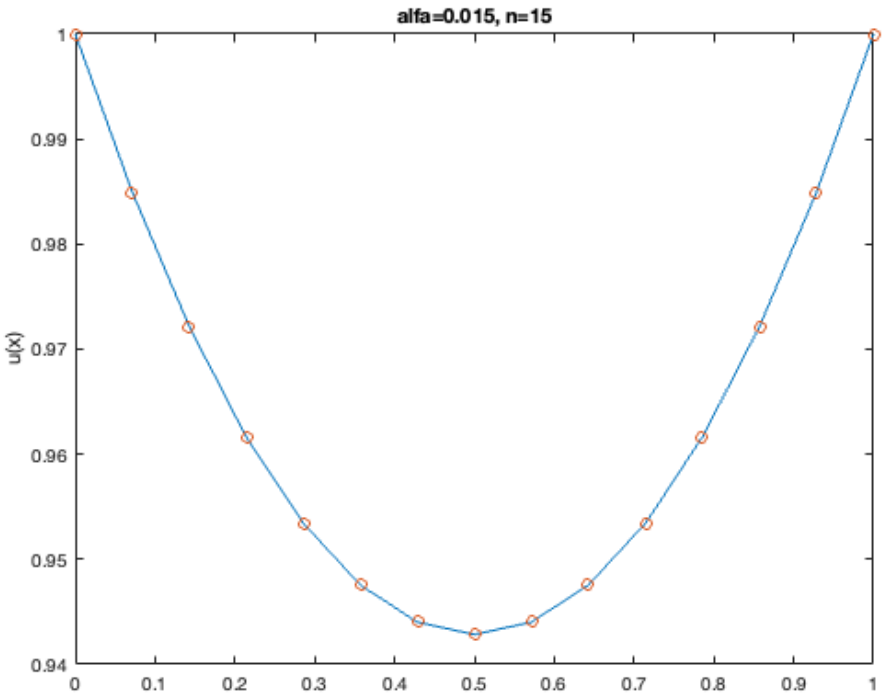

Figura 6: alfa=0.015, n=15

Con

$$\|f(u_n)\| = 0{,}4754$$

En este caso, sí podríamos aumentar n a 50 sin llegar a diverger.

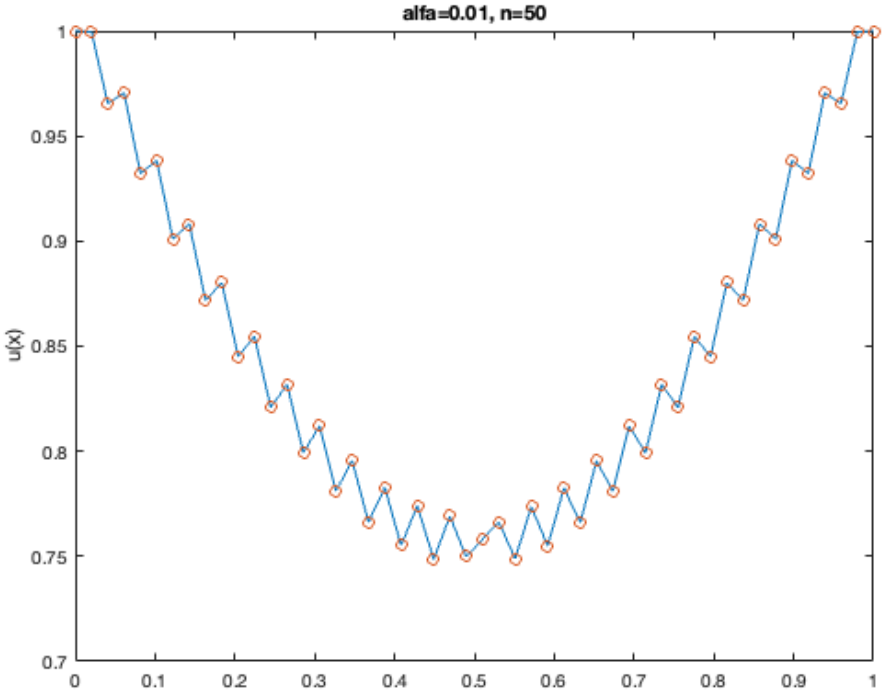

Figura 7: alfa=0.01, n=50

Con

$$\|f(u_n)\| = 0{,}3597$$

Esto demuestra que no solo el número de subintervalos es importante, si no el tomar un alfa óptimo.

3. Anexo

3.1. fun

```
function [fx, Dfx] = fun(x)
% En nuestro caso, u0=x
    n = length(x);
    % Como tenemos F definido, podemos calcularlo directamente
    fx_punt = zeros(1,n);
    fx_punt(1) = (1/(2*n))*(x(2)+x(1))*sqrt(1+n^2*(x(2)-x(1))^2);
    fx_punt(end) = (1/(2*n))*(x(end)+x(end-1))*sqrt(1+n^2*(x(end)-x(end-1))^2);
    for i = 1:n-2
        fx_punt(i+1) = (1/(2*n))*(x(i+1)+x(i))*sqrt(1+n^2*(x(i+1)-x(i))^2) + ...
                       (1/(2*n))*(x(i+2)+x(i+1))*sqrt(1+n^2*(x(i+2)-x(i+1))^2);
    end
    fx = fx_punt;

    % Calculamos ahora el gradiente
    Dfx_punt = zeros(1,n);
    Dfx_punt(1) = (1/(2*n))*(sqrt(1+(n^2)*(x(2)-x(1))^2)- ...
        (2*n^2*(x(2)-x(1))*(x(2)+x(1)))/sqrt(1+(n^2)*(x(2)-x(1))^2)...
        );
    Dfx_punt(end) = (1/(2*n))*(sqrt(1+n^2*(x(end)-x(end-1))^2)+ ...
        (2*n^2*(x(end)-x(end-1))*(x(end)+x(end-1)))/sqrt(1+n^2*(x(end)-x(end-1))^2)...
        );
    for i = 1:n-2
    Dfx_punt(i+1) = (1/(2*n))*(sqrt(1+n^2*(x(i+1)-x(i))^2)+ ...
        (2*n^2*(x(i+1)-x(i))*(x(i+1)+x(i)))/sqrt(1+n^2*(x(i+1)-x(i))^2)+...
        sqrt(1+n^2*(x(i+2)-x(i+1))^2)-...
```

```
        (2*n^2*(x(i+2)-x(i+1))*(x(i+2)+x(i+1)))/sqrt(1+n^2*(x(i+2)-x(i+1))^2));
    end
    Dfx = Dfx_punt;

end
```

3.2. fun_alpha

```
    function [Fa] = fun_alpha(alpha,x0, Dfx)
     y = x0-alpha*Dfx;
     [Fa, ~] = fun(y);
end
```

4. Bibilografia

Referencias

[1] Apuntes de la asignatura de Optimización