Tarea 3 MAT2605

Diego Pérez

31 de Agosto

Problema 1

(a) Hay $1+2+\cdots+n = \mathcal{O}(n^2)$ restas y $1+2+\cdots+n = \mathcal{O}(n^2)$ productos. Además, hay $n = \mathcal{O}(n)$ sumas. Juntando todo, hay $\mathcal{O}(n^2)$ flops.

(b) Se calculan varias veces cada resta y producto, por lo que podemos escribirlo de la siguiente manera (para simplificar, sea $f[x_0, \cdots, x_j] = a_j$).

$$p(x) = a_0 + (x - x_0)[a_1 + (x - x_1)[a_2 + (x - x_2)[a_3 + \cdots]]]$$

Notar que ahora se calcula una vez cada suma, producto y resta, por lo que cada uno de estas operaciones aportan $\mathcal{O}(n)$ operaciones, dando un total de $\mathcal{O}(n)$ flops.

(c) Sea p_0 el polinomio original (p en enunciado). El polinomio actualizado es

$$p(x) = p_0(x) + f[x_0, \cdot, x_n, x_n](x - x_0)(x - x_1)\cdots(x - x_n)$$

Si se trabaja el polinomio como en el inciso (b), se agregan 2 operaciones, junto con $\mathcal{O}(n)$ operaciones para hallar $f[x_0, \cdot, x_n, x_n]$. En total se usan $\mathcal{O}(n)$ flops extra.

(d) La tabla de Newton queda:

x	$f(x)$	$-$	$-$	$-$
-2	2	16	-6	3
-1	18	-2	9	-
1	14	34	-	-
3	82	-	-	-

Por lo que el polinomio queda:

$$\begin{aligned} P(x) &= 2 + 16(x+2) - 6(x+2)(x+1) + 3(x+2)(x+1)(x-1) \\ &= 3x^3 - 5x + 16 \end{aligned}$$

(e) Función:

```
function tab = DividedDifferences(x,fx)
n = length(x);
tab = zeros(n,n+1);
tab(:,1) = x ;
tab(:,2) = fx;
for j = 3:n+1
    for i = 1:(n+2-j)
        tab(i,j) = (tab(1+i,j-1)-tab(i,j-1))/(tab(i+j-2,1)-tab(i,1));
    end
end
end
```

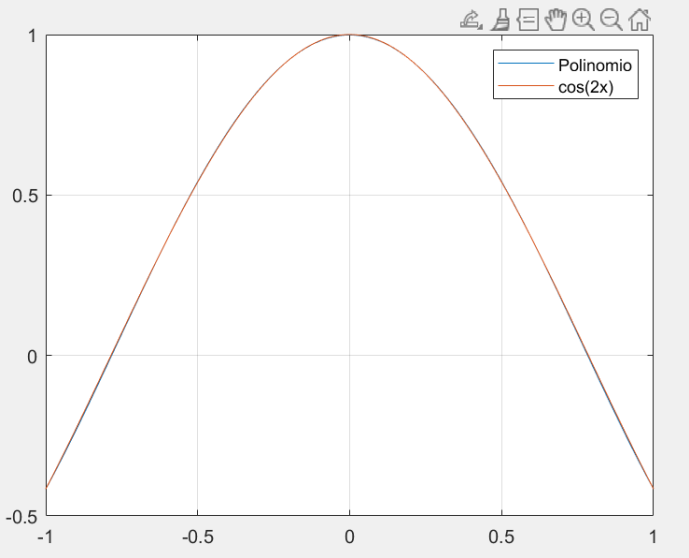

Figura 1: 1e: Aproximación por polinomio de $\cos 2x$

Gráfico:

```
xnod = -1:(2/5):1
ynod = [0,0,0,0,0,0];
for i = 1:6
    ynod(i) = cos(2*xnod(i)) ;
end
x = -1:.01:1;
y_ = zeros(1,length(x));
for i = 1:(length(x))
    y_(i) = pol(x(i));
end

plot(x,y_,x,cos(2*x))
grid on
legend('Polinomio','cos(2x)')

function res = pol(x)
res = -0.4161+1.9463*(x+1)-0.6869*(x+0.6)*(x+1)-0.8826*(x+0.6)*(x+1)*(
   x+0.2)+0.5516*(x-0.2)*(x+0.6)*(x+1)*(x+0.2)
end
```

Problema 2

(a) Si lo es. Sea S_0 el primer polinomio, S_1 el sugundo y S_2 el tercero. Para chequear que es el spline deseado, basta verificar las siguientes igualdades:

$$
\begin{aligned}
S_0(1) &= 2 \\
S_0(2) &= S_1(2) = 2 \\
S_1(3) &= S_2(3) = 1 \\
S_2(4) &= 11 \\
S_0'(2) &= S_1'(2) \\
S_1'(3) &= S_2'(3) \\
S_0''(1) &= 0 \\
S_0''(2) &= S_1''(2) \\
S_1''(3) &= S_2''(3) \\
S_2''(4) &= 0
\end{aligned}
$$

Sin embargo, esto es directo de:

$$
\begin{aligned}
S_0'(x) &= 1 - 3(x-1)^2 \\
S_1'(x) &= -2 - 6(x-2) + 12(x-2)^2 \\
S_2'(x) &= 4 + 18(x-3) - 9(x-3)^2 \\
S_0''(x) &= -6(x-1) \\
S_1''(x) &= -6 + 24(x-2) \\
S_2''(x) &= 18 - 18(x-3)
\end{aligned}
$$

(b) Funciones:

```
function res=SplineConstruction(x,fx)
n = length(x)-1;
h = zeros(1,n);
for i = 1:(n)
    h(i) = x(i+1)-x(i);
end
mat = zeros(n+1,n+1);
b_ = zeros(1,n+1);
for i=1:(n+1)
    if i==1
        mat(i,i) = 1;
    elseif i==(n+1)
        mat(i,i) = 1;
    else
        mat(i,i) = 2*(h(i)+h(i-1));
        mat(i,i-1) = h(i-1);
        mat(i,1+i) = h(i);
        b_(i) = 3*((fx(i+1)-fx(i))/h(i)-(fx(i)-fx(i-1))/h(i-1));
    end
end
c = linsolve(mat,b_.');
b = zeros(n,1);
d = zeros(n,1);
```

```
for i=1:n
    b(i) = (fx(i+1)-fx(i))/h(i)-h(i)*(2*c(i)+c(i+1))/3;
    d(i) = (c(i+1)-c(i))/(3*h(i))

res = zeros(n,4)
res(:,1) = fx(1:n)
res(:,2) = b
res(:,3) = c(1:n)
res(:,4) = d
end
end

function S=EvaluateSpline(x,Scf,xplot)
n = length(x)-1;
t_len = length(xplot);
S = ones(1,t_len);
for i = 1:t_len
    t = xplot(i);
    for j = 1:n
        if x(j) <= t & x(j+1) >= t
            d = t-x(j);
            S(i) = Scf(j,1)+Scf(j,2)*d+Scf(j,3)*d^2+Scf(j,4)*d^3;
        end
    end
end
end
```

Gráfico:

```
x_ = [-1,-0.6,-0.2,0.2,0.6,1];
fx_ = [0,0,0,0,0,0];
for i=1:6
    fx_(i) = 1/(1+25*x_(i)^2);
end
Scf_ = SplineConstruction(x_,fx_);
xplot_ = -1:0.01:1;
y = zeros(1,201)
for i=1:201
    y(i)= fun(xplot_(i));
end

S = EvaluateSpline(x_,Scf_,xplot_);

plot(xplot_,S,'blue',xplot_,y,'red')
legend('Aproximacion Spline','Funcion Original')
```

(c) Supongamos por contradicción que A es singular, por lo que existiría un vector c no trivial con $Ac = 0$. Tomamos un i tal que $|c_i| \geq |c_k|$ para todo k. Sea a el vector que representa la fila i de A. Sigue que:

$$0 = (Ac)_i = \sum_j a_j c_j \implies \left| \sum_{j \neq i} a_j c_j \right| = |a_i c_i|$$

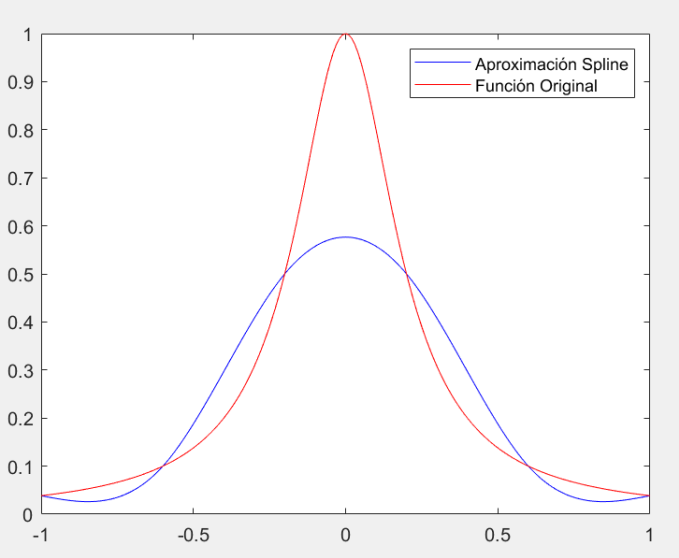

Figura 2: 2b: Aproximación por Spline

Aplicando la desigualdad triangular:

$$|a_i c_i| = \left|\sum_{j\neq i} a_j c_j\right| \leq \sum_{j\neq i} |a_j c_j| \leq \sum_{j\neq i} |a_j c_i| = |c_i| \sum_{j\neq i} |a_j| < |c_i a_i|$$

Contradicción. Se conlcuye por principio de contradicción.

(d) Notar que si la matriz que describe el sistema de ecuaciones es invertible, entonces la solución es única, entregando un spline único. Por el ítem anterior, solo basta demostrar que $|2(h_i + h_{i+1})| > |h_i| + |h_{i+1}|$ (y 1>0, pero esto es trivial). Como los x_i estan ordenados, todos los h_i son positivos y como son todos distintos, $h_i > 0$ para todo i. Sigue que

$$|2(h_i + h_{i+1})| = (h_i + h_{i+1}) + (h_i + h_{i+1}) > (h_i + h_{i+1}) = |h_i| + |h_{i+1}|$$

Como queríamos.

Problema 3

(a) Función:

```
function v=BezierCurve(v0,c0,c1,v1,t)
n = length(t)-1
v = zeros(2,n+1)
for i=1:(n+1)
    tt = t(i);
    v(:,i) = (1-tt)^3*v0+3*(1-tt)^2*tt*c0+3*(1-tt)*tt^2*c1+tt^3*v1;
end
end
```

Gráficos:

```
t_=0:0.01:1
v0_=[0,0];
```

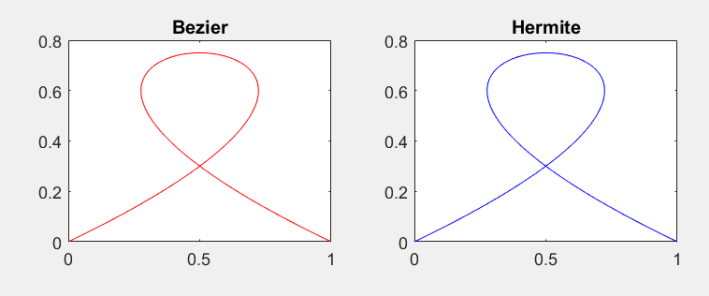

Figura 3: 3a: Bezier vs Hermite

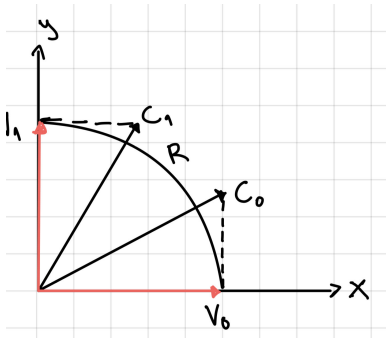

Figura 4: 3a: Diagrama cuarto círculo

```
v1_=[1,0];
c0_=[2,1];
c1_=[1,0]+[-2,1]

cor = BezierCurve(v0_,c0_,c1_,v1_,t_)

xh = [0,1];
fh = [0,1];
dfh = [6,6];

Her_x = Hermite(xh,fh,dfh,t_);

xh = [0,1];
fh = [0,0];
dfh = [3,-3];

Her_y = Hermite(xh,fh,dfh,t_);

plot(cor(1,:),cor(2,:))
subplot(2,2,1);
plot(cor(1,:),cor(2,:),'red')
title('Bezier')

subplot(2,2,2);
plot(Her_x,Her_y,'blue')
title('Hermite')
```

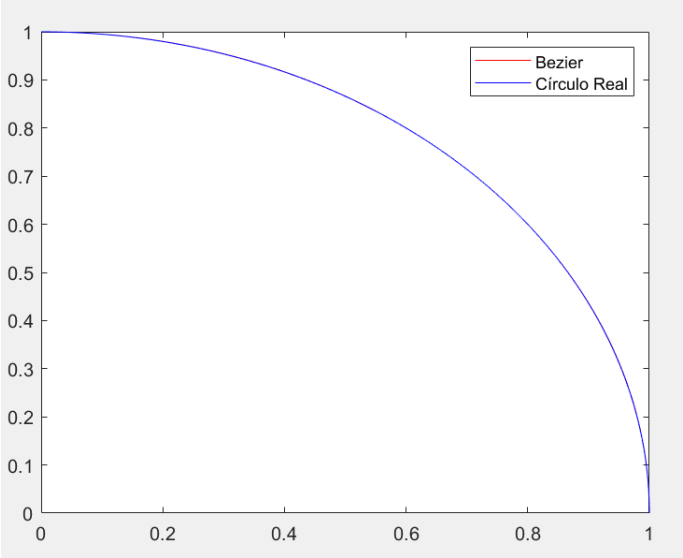

Figura 5: 3b: Aproximación por Spline

(b) (I)

Se escogen estos puntos porque queremos que la derivada de la curva coincida con la derivada del círculo, por lo que escoger dichos valores, estamos seleccionando en el fondo 2 vectores ($v_0 - c_0$ y $v_1 - c_1$) de igual magnitud que son tangentes al círculo.

Reemplazando $t = 1/2$, obtenemos:

$$\vec{v}(1/2) = \frac{1}{8}(\vec{v_0} + \vec{v_1}) + \frac{3}{8}(\vec{c_0} + \vec{c_1}) = \left(\frac{3}{8}H + \frac{1}{2}R\right)\begin{bmatrix}1\\1\end{bmatrix} = R\sqrt{2}\begin{bmatrix}1\\1\end{bmatrix}$$

Donde la útlima expresión se debe a que en $t = 1/2$, se ha avanzado un octavo del círculo total. Resolviendo se obtiene $H = \frac{4R}{3}(\sqrt{2} - 1)$.

(c) Corriendo el siguiente código, se obtiene que el error es aprox 3.2801e-05.

```
R = 1
H = 4/3*(sqrt(2)-1)
t_=0:0.01:1;
v0_=[R;0];
v1_=[0;R];
c0_=[R;H];
c1_=[H;R]

cor = BezierCurve(v0_,c0_,c1_,v1_,t_);

t=linspace(0,1);
circ_x = cos(t*pi/2);
circ_y = sin(t*pi/2);

plot(cor(1,:),cor(2,:),'red',circ_x,circ_y,'blue')

mas_lejano = (cor(:,1)+cor(:,2))/2
error = abs(1-norm(mas_lejano))
```

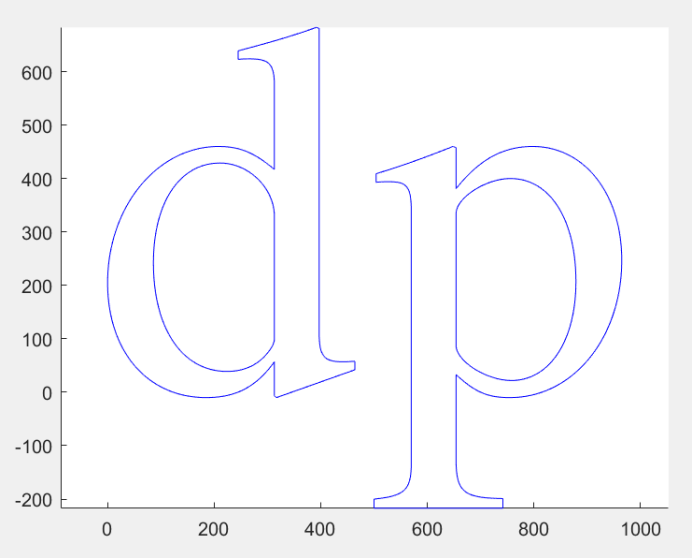

Figura 6: 3c: Iniciales dp

.

```
load('C:\Users\Panormica\Desktop\2023 S2\Clculo Cientfico\Tarea 3\
    AttachedFiles\Glyph-d.mat')
t=0:0.01:1;
figure
hold on
for j=1:size(V0,2);
    v=BezierCurve(V0(:,j),C0(:,j),C1(:,j),V1(:,j),t);
    plot(v(1,:),v(2,:),'b')
    axis equal
end

load('C:\Users\Panormica\Desktop\2023 S2\Clculo Cientfico\Tarea 3\
    AttachedFiles\Glyph-p.mat')
for j=1:size(V0,2);
    v=BezierCurve(V0(:,j),C0(:,j),C1(:,j),V1(:,j),t);
    v(1,:) = (v(1,:)+ones(1,101)*500);
    plot(v(1,:),v(2,:),'b')
    axis equal
end
```