P8: CASAS Y JIMENEZ

Table of Contents

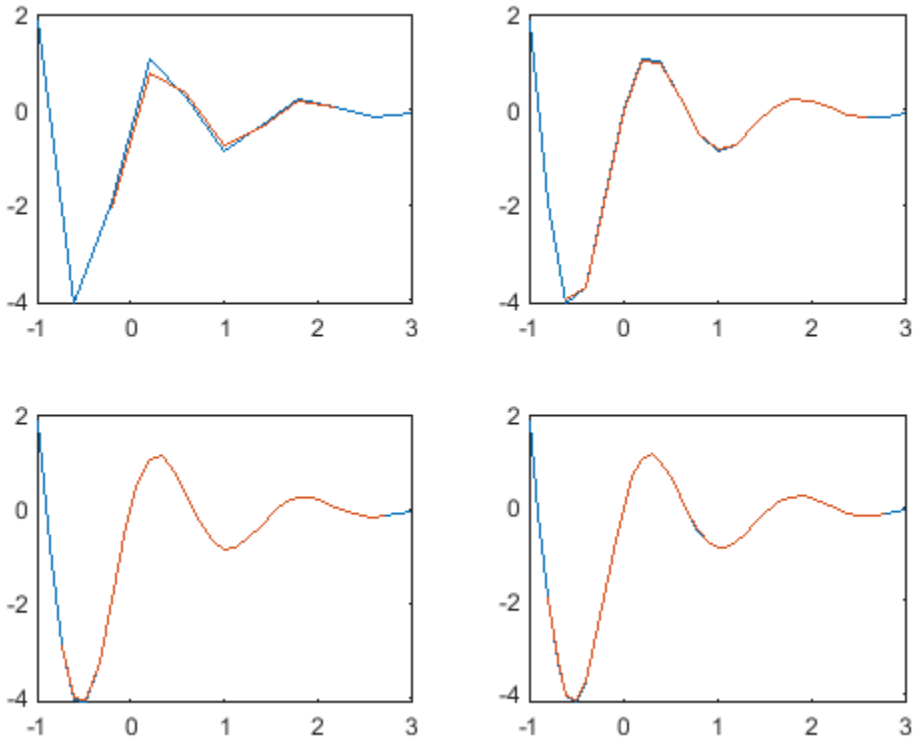
1) Diferencia finita local:

```
%{
Durante las horas de práctica observamos que era posible
realizar este apartado de dos maneras distintas. En primer
lugar, aproximando la derivada con la expresión que aparece
en el enunciado y, en segundo lugar, mediante la construcción
de la matriz de derivación tal y como se hizo en la práctica 7.

Hemos resuelto este apartado de los dos modos.
%}
```

A partir de la fórmula de derivación aproximada:

```
close all;
clear all;
format long g;

comptador = 1;

figure();
for n =10:10:40
    x = linspace(-1,3,n+1);
    h = abs(x(2)-x(1));
    fd_aprox = ((12*h)^(-1))*((-1)*(fun(x+2*h))+8*fun(x+h)-8*fun(x-
h)+fun(x-2*h));
    fd_aprox = fd_aprox(3:end-2);

    fd = (exp(-x)).*(sin(2.*x)).*(4.*cos(2.*x)-sin(2.*x));
    subplot(2, 2, comptador);
    plot(x,fd);
    hold on;
    x = x(3:end-2);
    plot(x,fd_aprox);
    hold off;
```

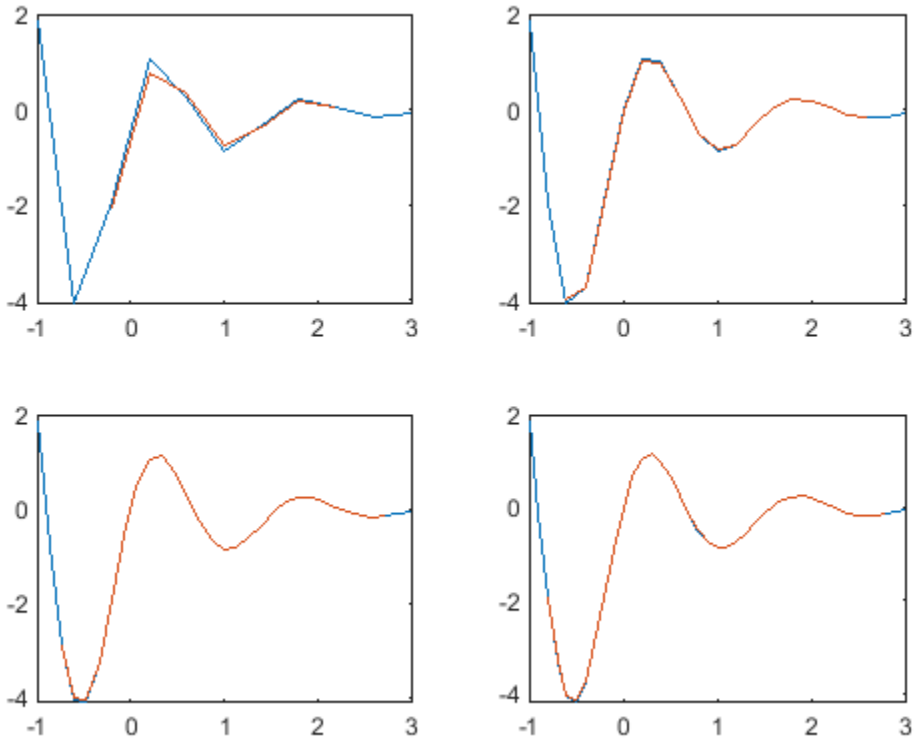
```
    comptador = comptador + 1;
end
```



A partir de la matriz de derivación:

```
clear all;
format rat;

comptador=1;
figure;
for n =10:10:40
    x = linspace(-1,3,n+1);
    h = abs(x(2)-x(1));
    fd = (exp(-x)).*(sin(2.*x)).*(4.*cos(2.*x)-sin(2.*x));

    fx = fun(x);
    fx = fx';
    u = zeros(1,n+1);
    v = zeros(1,n+1);
    % No cal que toquem el primer element perque ja es 0 en els dos.
    u(2) = 8;
    u(3) = -1;
    v = -u;

    fd = (exp(-x)).*(sin(2.*x)).*(4.*cos(2.*x)-sin(2.*x));
    subplot(2,2,comptador)
```

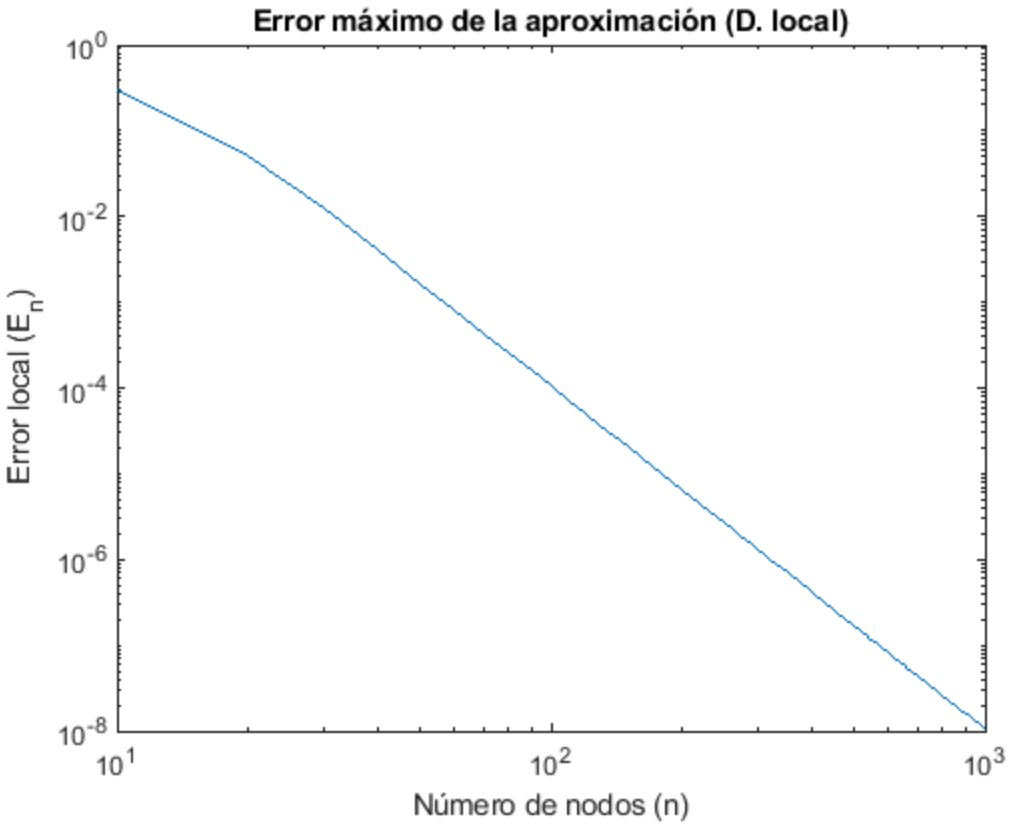
```
    plot(x,fd);
    hold on;
    M = toeplitz(v, u);
    fd_aprox = 1/(12*h).*M*fx;
    fd_aprox = fd_aprox(3:end-2);
    x = x(3:end-2);
    plot(x,fd_aprox);
    hold off;
    comptador = comptador+1;
end
```



Error máximo y representación:

```
%{
Para resolver este apartado, repetiremos el procedimiento anterior
(solo con la matriz de derivación) para una n creciente n=10:10:1000
y consideraremos, para cada n, el error màximo, lo acumularemos
en un vector y lo representaremos en escala doble logarítmica en
función de la n que le corresponda.
%}

clear all;
format long g;
errorsMax = [];
Ns = [10:10:1000];
figure;
```

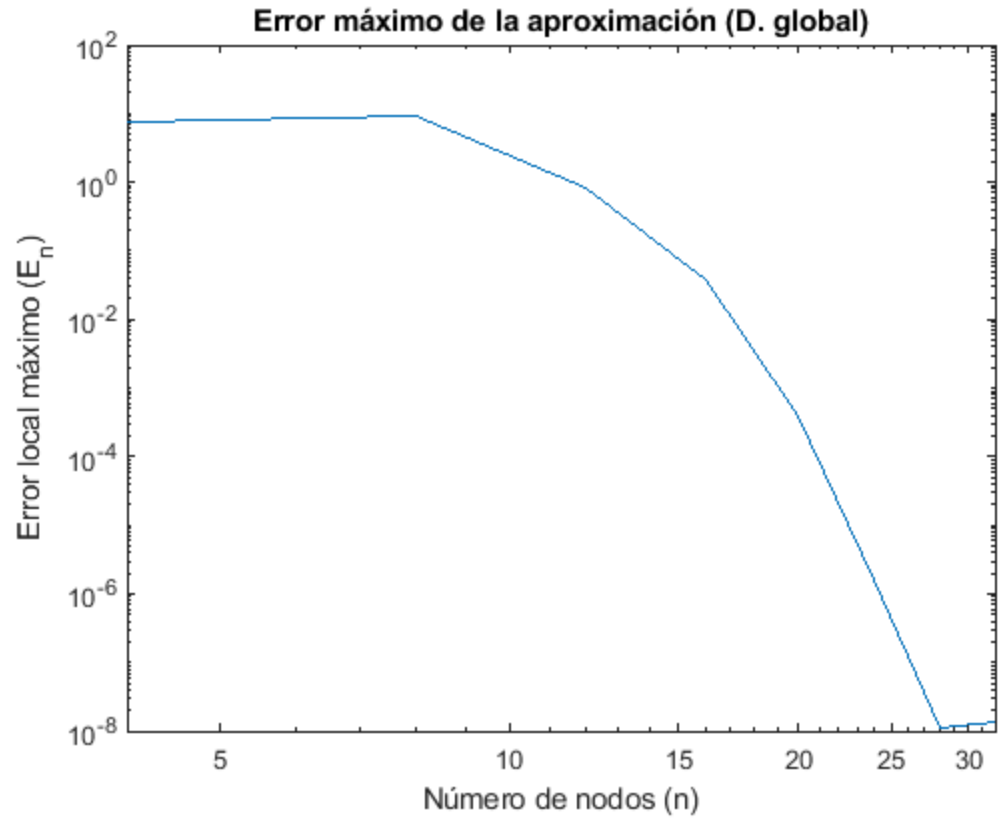
```
for n = Ns
    x = linspace(-1,3,n+1);
    h = abs(x(2)-x(1));
    fd = (exp(-x)).*(sin(2.*x)).*(4.*cos(2.*x)-sin(2.*x));

    %Obtenim la matriu i el vector de les derivades aproximades:
    fx = fun(x);
    fx = fx';
    u = zeros(1,n+1);
    v = zeros(1,n+1);
    u(2) = 8;
    u(3) = -1;
    v = -u;
    M = toeplitz(v, u);
    fd_aprox = 1/(12*h).*M*fx;
    % Vector de derivades exactes:
    fd = (exp(-x)).*(sin(2.*x)).*(4.*cos(2.*x)-sin(2.*x));

    fd_aprox = fd_aprox(3:end-2);
    fd = fd(3:end-2);

    error = max(abs(fd_aprox'-fd));
    errorsMax = [errorsMax error];

end
%{
Una vez representado, y para comprovar que decae con potencia p=-4,
utilizaremos el comando regression para construirnos una recta
lo más semejante posible a la representación del error y
obtendremos de esa recta su pendiente.
%}
loglog(Ns, errorsMax);
title('Error máximo de la aproximación (D. local)');
xlabel('Número de nodos (n)');
ylabel('Error local (E_n)');
[r,m,~] = regression(log10(Ns), log10(errorsMax))

%{
Como podemos ver, la recta generada es de buena calidad (|r|~1) y su
pendiente es m~-4, por lo que confirmamos la hipótesis inicial: el
error decae con potencia p=-4.
%}


r =

        -0.999520475735653


m =

          -3.9334264830878
```

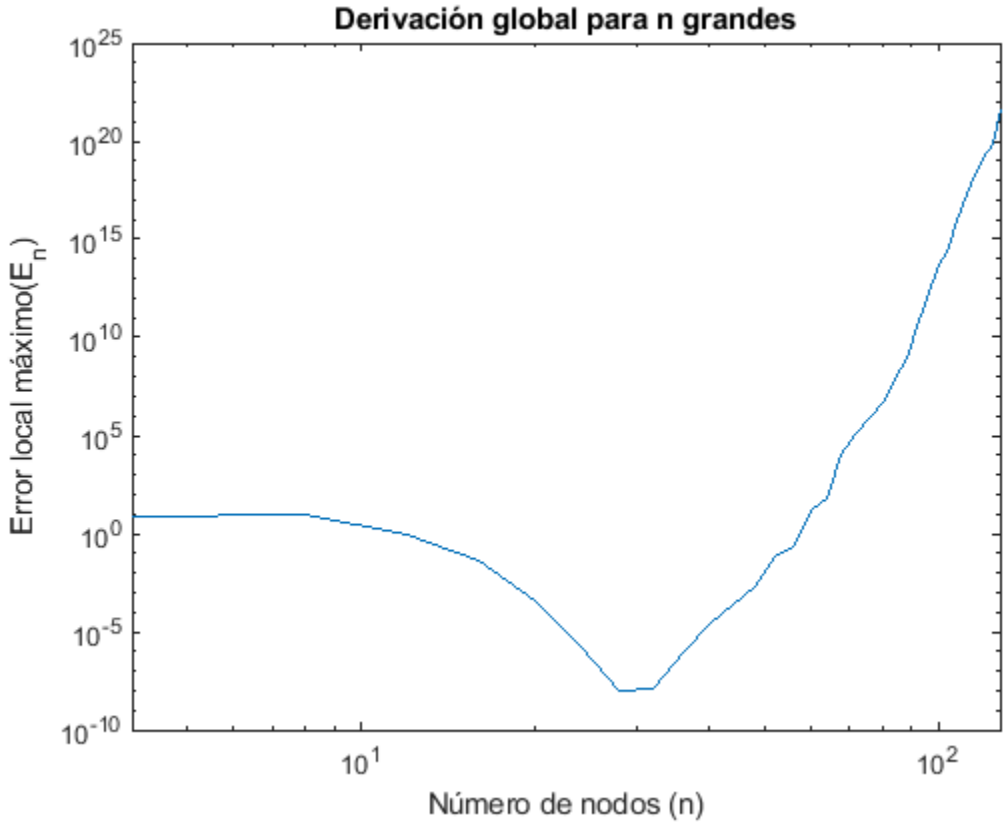


2) Derivación global:

```
%{
En este apartado debemos aproximar la derivada de la función de
manera global, es decir, con un único polinomio interpolador
que contenga la información de todos los nodos (el valor de la
función en ellos) y a los que habrá que aplicar la matrix
de diferenciación para un n creciente n=4:4:32.

Como hemos hecho antes, acumularemos en un vector el valor del
error máximo y lo representaremos en función de la n que le
corresponda.
%}
clear all;
errorsMax = [];

Ns = [4:4:32];

for n = Ns
    x = linspace(-1,3,n+1); % n+1 punts equiespaciats entre -1 i 3.
    fx = fun(x);
    fx = fx';

    D = diferenciacion(x);
    fd_aprox = D*fx;
```

```
    % Vector de derivades exactes:
    fd = (exp(-x)).*(sin(2.*x)).*(4.*cos(2.*x)-sin(2.*x));
    fd_aprox = fd_aprox';

    error = max(abs(fd_aprox-fd));
    errorsMax = [errorsMax error];
end
figure;
loglog(Ns, errorsMax);
title('Error máximo de la aproximación (D. global)');
xlabel('Número de nodos (n)');
ylabel('Error local máximo (E_n)');

%{
En aquesta figura és dificil veure Runge, tanmateix el podem
observar en el rebot de l'error que es produeix a partir de n = 28.
Si fem el mateix procés però arribant a n més altes (per exemple 128)
podem veure el fenomen de Runge més clarament:
%}

clear all;
errorsMax = [];

Ns = [4:4:128];

for n = Ns
    x = linspace(-1,3,n+1);
    % n+1 punts equiespaciats entre -1 i 3.
    fx = fun(x);
    fx = fx';

    D = diferenciacion(x);
    fd_aprox = D*fx;

    % Vector de derivades exactes:
    fd = (exp(-x)).*(sin(2.*x)).*(4.*cos(2.*x)-sin(2.*x));
    fd_aprox = fd_aprox';

    error = max(abs(fd_aprox-fd));
    errorsMax = [errorsMax error];
end

figure;
loglog(Ns, errorsMax);
title('Derivación global para n grandes')
xlabel('Número de nodos (n)');
ylabel('Error local máximo(E_n)');

%{
En aquesta gràfica és fàcil veure el fenomen de Runge ja que
apareixen errors de fins a 10^25.
%}
```


Error máximo de la aproximación (D. global)
Error local máximo (E_n)
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
5
10
15
20
25
30
Número de nodos (n)


Derivación global para n grandes
Error local máximo(E_n)
10^{25}
10^{20}
10^{15}
10^{10}
10^{5}
10^{0}
10^{-5}
10^{-10}
10^{1}
10^{2}
Número de nodos (n)

Published with MATLAB® R2018b