P7: CASAS Y JIMENEZ

Table of Contents

1) Introducción al comando toeplitz:

```
ejemplo = toeplitz([1 3 5 7],[1 2 3 4])
```

```
ejemplo =

     1     2     3     4
     3     1     2     3
     5     3     1     2
     7     5     3     1
```

2) Construcción de las matrices de diferenciación:

```
%{
Para construirnos las matrices de diferenciación mediante el
comando toeplitz, nos generaremos dos vectores de ceros de
longitud n, y cambiaremos su primera y segunda o tercera
componente por la que nos sea conveniente, para que una vez
ejecutado el comando podamos obtener las matrices que se
nos piden. Dichas matrices nos permitiran hacer la derivada
forward y centered.
%}

clear all;
close all;
format rat;
for n = [5,20,40,80]
    u = zeros(1,n+1);
    v = zeros(1,n+1);
    u(1) = -1; v(1) = -1;
    v(2) = 1;
    FD = toeplitz(u, v);
    FD = FD(1:end-1,:);
    w = zeros(1,n+1);
    x = zeros(1,n+1);
    w(1) = -1/2; x(1) = -1/2 ;
```

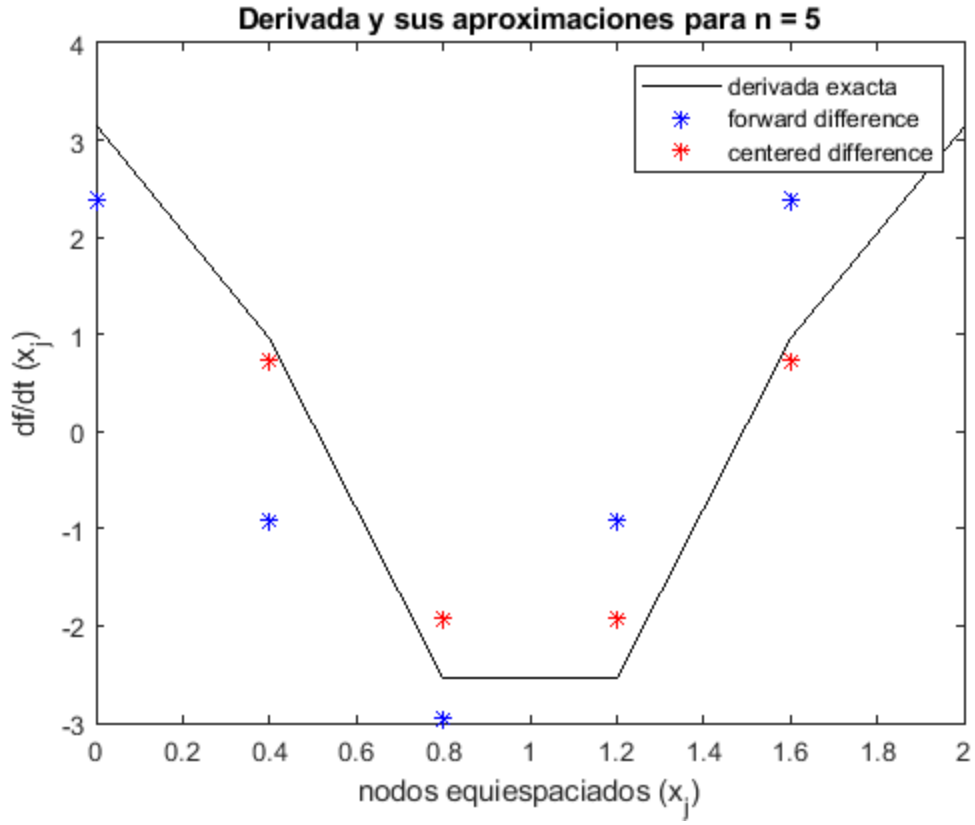
```
    x(3) = 1/2;
    CD = toeplitz(w, x);
    CD = CD(1:end-2,:);
    if n==5
        FD
        CD
    end
end
```

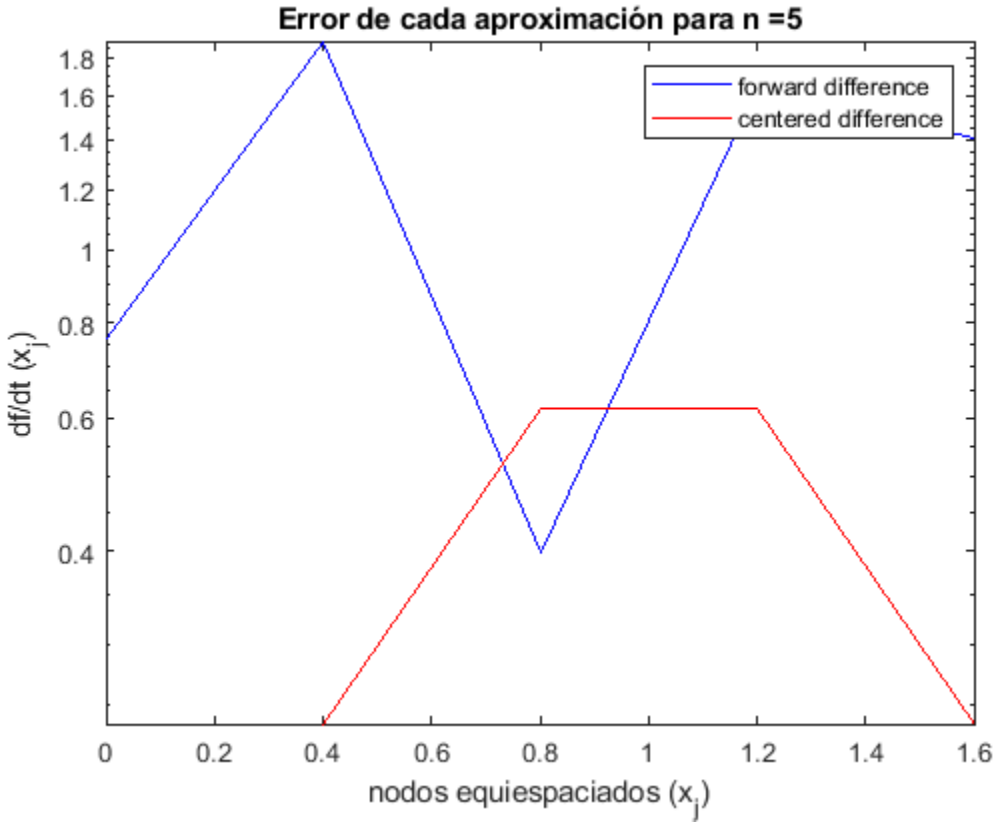
```
FD =

  Columns 1 through 5

      -1             1             0             0             0

       0            -1             1             0             0

       0             0            -1             1             0

       0             0             0            -1             1

       0             0             0             0            -1

  Column 6

       0
       0
       0
       0
       1

CD =

  Columns 1 through 5

    -1/2             0           1/2             0             0

       0          -1/2             0           1/2             0

       0             0          -1/2             0           1/2

       0             0             0          -1/2             0

  Column 6

       0
       0
       0
     1/2
```

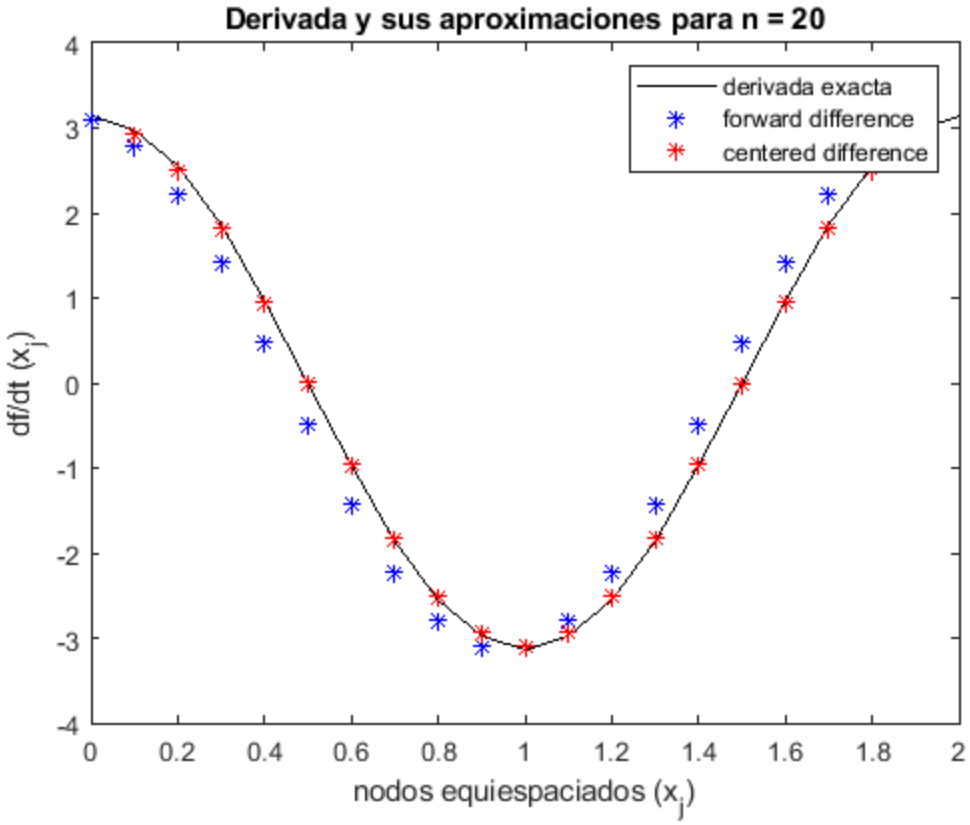
3) Derivada de sin(pi*x) y error:

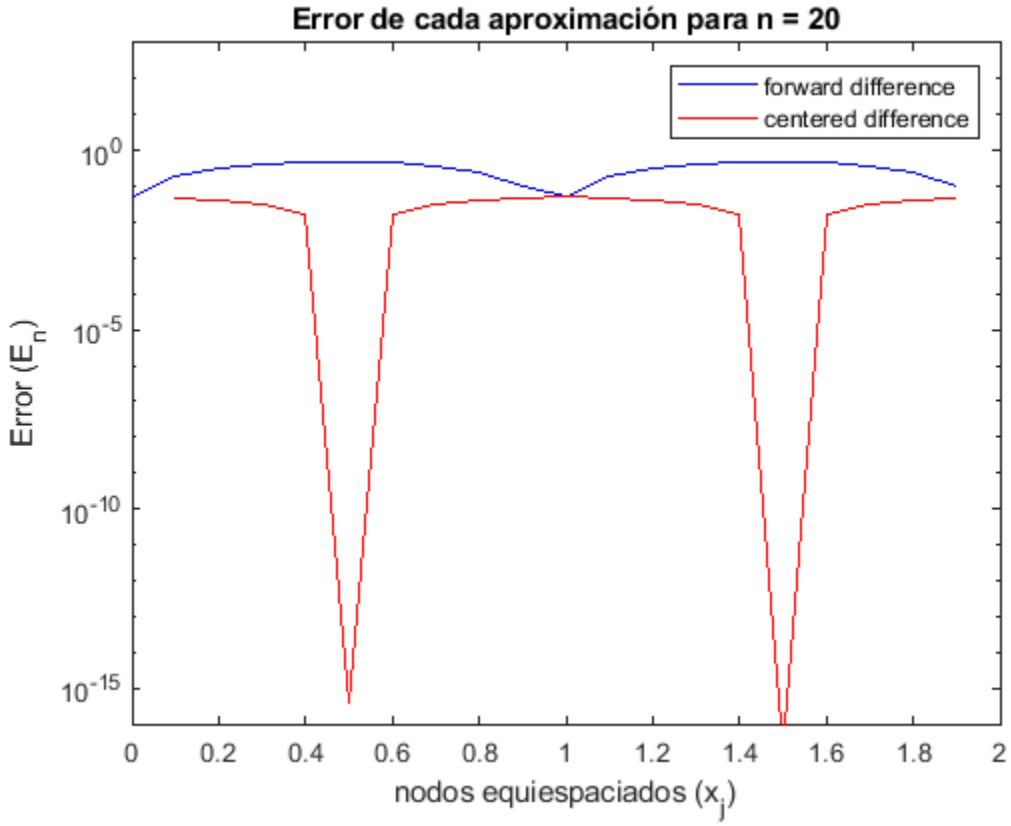
```
clear all;
format long g;

for n = [5,20,40,80]
    xj = linspace(0,2,n+1); %obtenemos los n+1 nodos equiespaciados.

    %{
    Volvemos a construir las matrices en este apartado para
    poder aprovechar el bucle y poder hacer la diferenciación
    para todas las n con un mismo código.
    %}
    u = zeros(1,n+1); v = zeros(1,n+1);
    u(1) = -1; v(1) = -1; v(2) = 1;
    FD = toeplitz(u, v);
    FD = FD(1:end-1,:);
    w = zeros(1,n+1); x = zeros(1,n+1);
    w(1) = -1/2; x(1) = -1/2 ; x(3) = 1/2;
    CD = toeplitz(w, x);
    CD = CD(1:end-2,:);

    %{
    Una vez construidas las matrices, generamos el vector columna
    con el valor de la función en los nodos y lo multiplicamos
    por las matrices que acabamos de obtener.
    %}
    f = sin(pi.*xj); %es la función que se nos pide derivar.
    f = f';
    h=abs(xj(1)-xj(2)); %la h es la distancia entre dos nodos.
    der_fd = ((1/h).*(FD*f))';
    der_cd = ((1/h).*(CD*f))';

    %Y a continuación represento los resultados:
    fDer = pi*cos(pi.*xj); %es la derivada de la función en los nodos.
    figure();
    plot(xj, fDer, 'k');
    titol_plot_der = sprintf('Derivada y sus aproximaciones para n =
 %d', n);
    title(titol_plot_der);
    xlabel('nodos equiespaciados (x_j)');
    ylabel('df/dt (x_j)');
    hold on;
    plot(xj(1:end-1), der_fd, '*b'); %es la derivada forward.
    hold on;
    plot(xj(2:end-1), der_cd, '*r'); %es la derivada centered.
    legend('derivada exacta','forward difference','centered
 difference');
    hold off;

    %{
    Una vez aproximada la derivada de las dos maneras, calculamos
    el error de dichas aproximaciones para cada n. Se puede
```

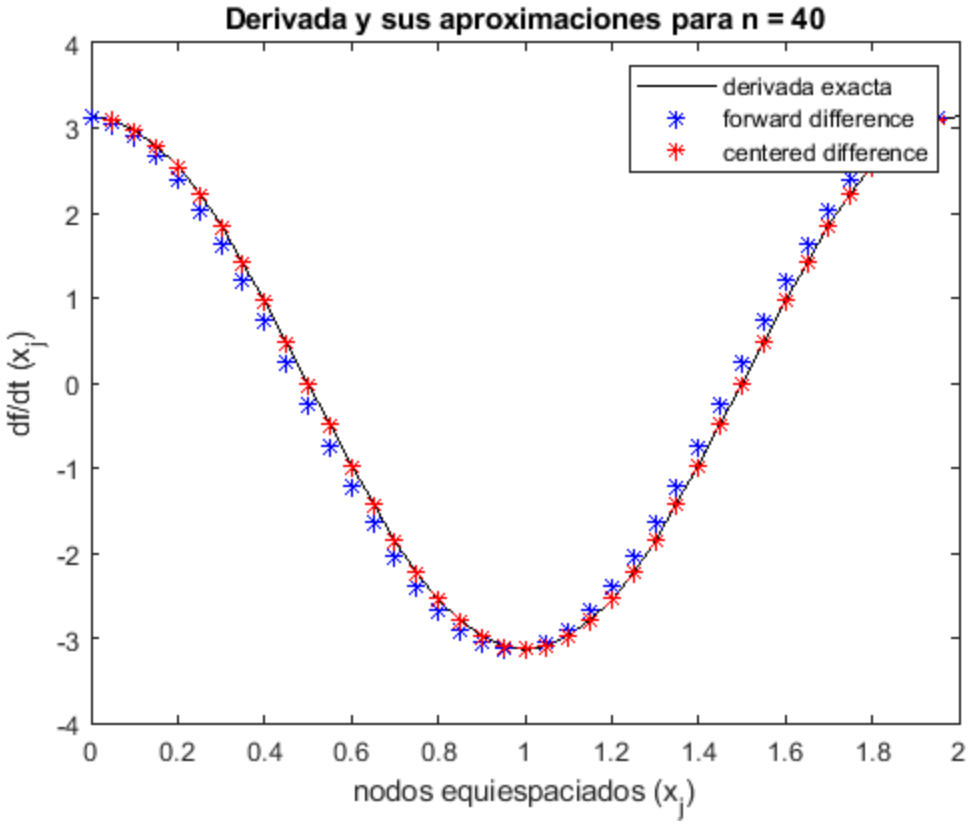
```
    observar en las gráficas que el error disminuye notoriamente
    en augmentar la n, y también que su comportamiento es muy
    diferente cuando derivamos forward que cuando derivamos
    centered, en concreto, el error es menor con la centered.
    %}
    error_fd = abs(fDer(1:end-1) - der_fd);
    error_cd = abs(fDer(2:end-1) - der_cd);

    if n==5
    figure;
    semilogy(xj(1:end-1), error_fd, 'b');
    title('Error de cada aproximación para n =5');
    xlabel('nodos equiespaciados (x_j)');
    ylabel('Error (E_n)');
    xlabel('nodos equiespaciados (x_j)');
    ylabel('df/dt (x_j)');
    hold on;
    semilogy(xj(2:end-1), error_cd, 'r');
    legend('forward difference','centered difference');
    hold off;
    else
    figure;
    semilogy(xj(1:end-1), error_fd, 'b');
    titol_plot_err = sprintf('Error de cada aproximación para n = %d',
 n);
    title(titol_plot_err);
    xlabel('nodos equiespaciados (x_j)');
    ylabel('Error (E_n)');
    axis([0 2 10*10^-17 1000]);
    hold on;
    semilogy(xj(2:end-1), error_cd, 'r');
    legend('forward difference','centered difference');
    hold off;
    end
end
```

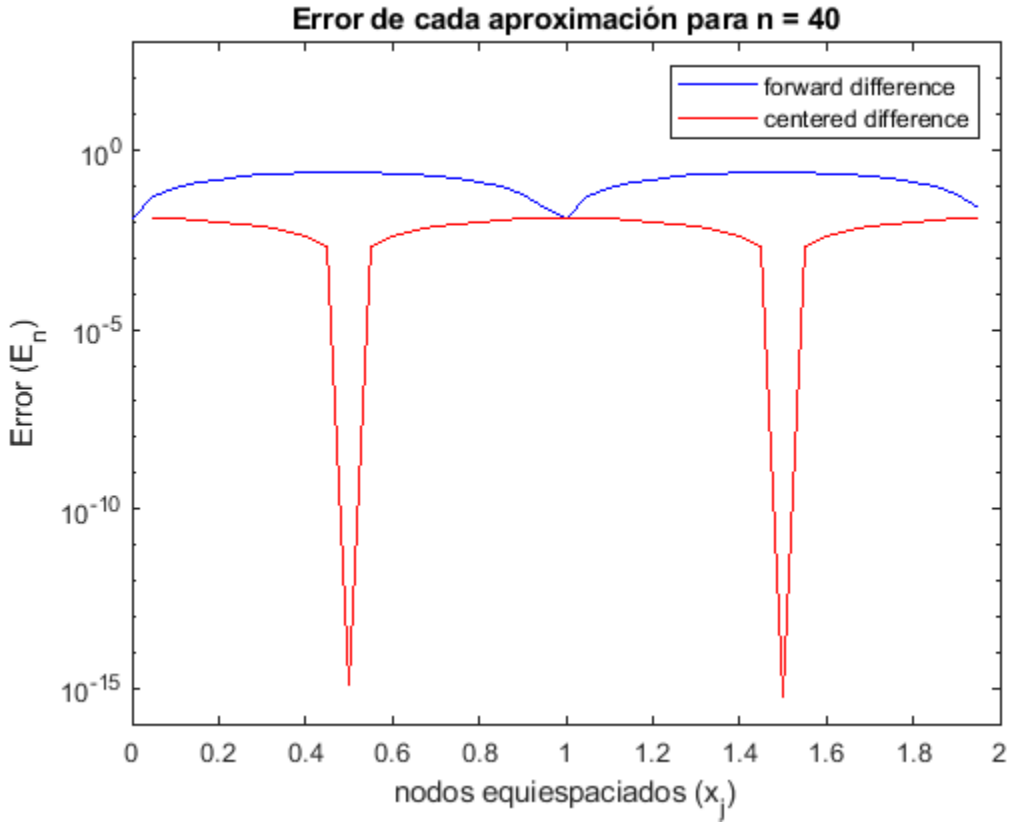

Derivada y sus aproximaciones para n = 5
derivada exacta
forward difference
centered difference
df/dt (x_j)
nodos equiespaciados (x_j)

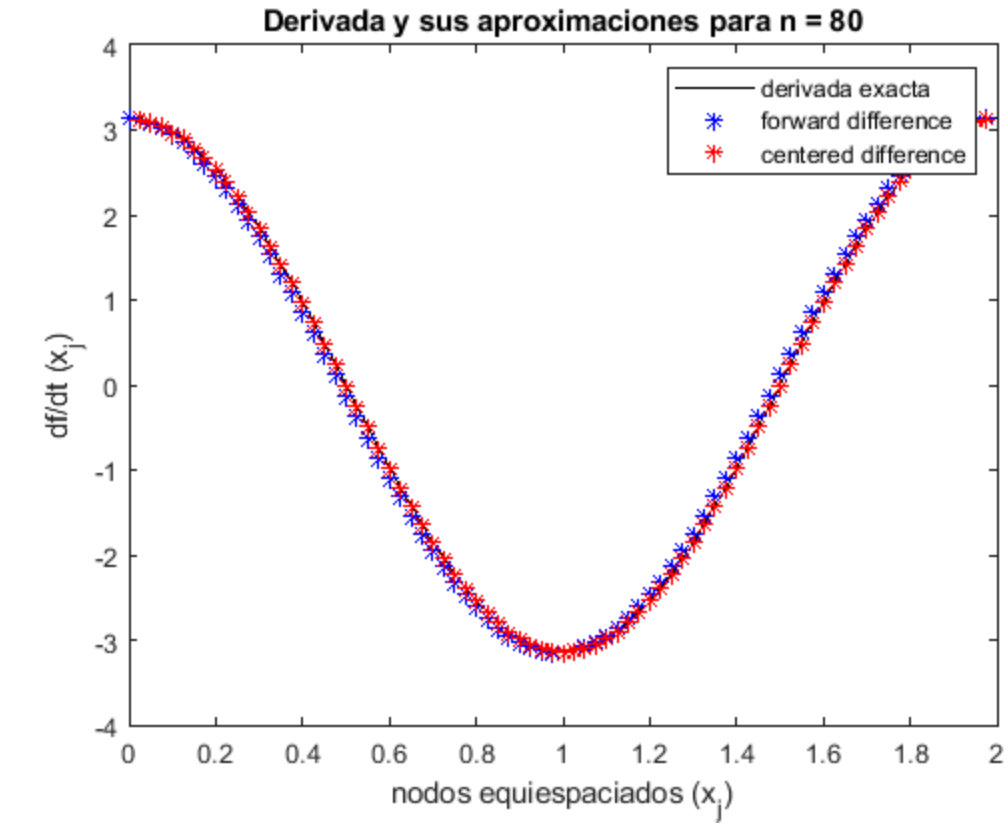

Error de cada aproximación para n =5
forward difference
centered difference
df/dt (x_j)
nodos equiespaciados (x_j)

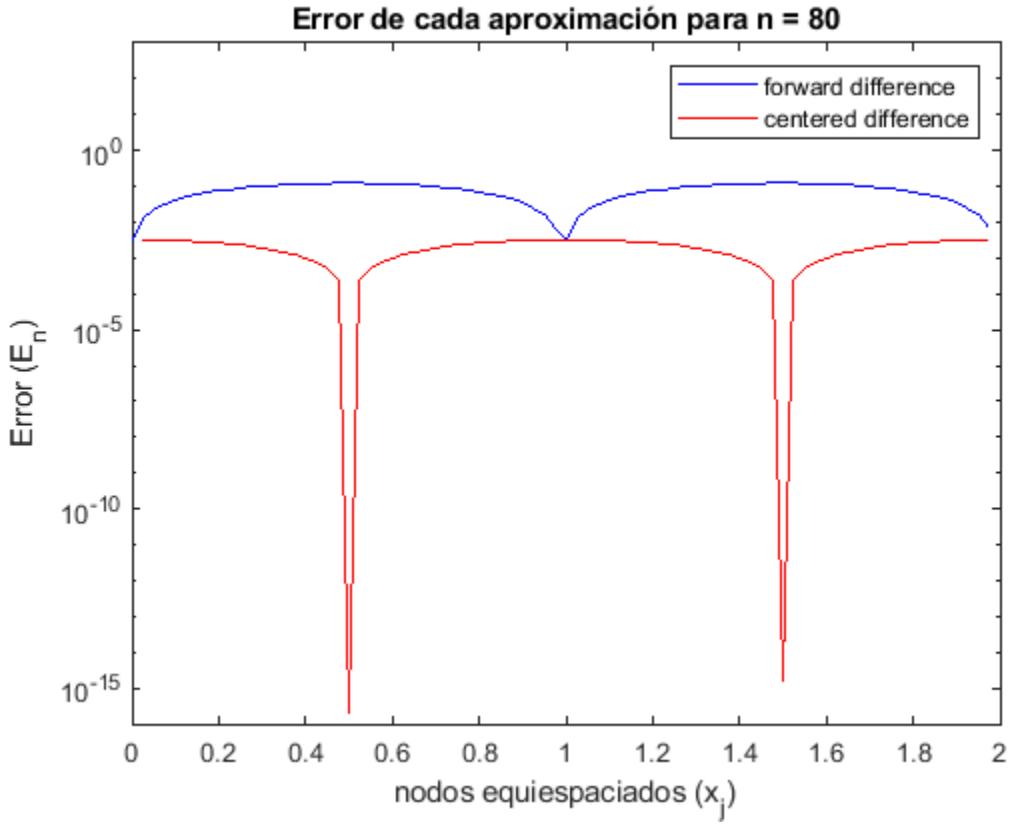

Derivada y sus aproximaciones para n = 20
derivada exacta
forward difference
centered difference
df/dt (x_j)
nodos equiespaciados (x_j)


Error de cada aproximación para n = 20
forward difference
centered difference
Error (E_n)
nodos equiespaciados (x_j)


Derivada y sus aproximaciones para n = 40
derivada exacta
forward difference
centered difference
df/dt (x_j)
nodos equiespaciados (x_j)


Error de cada aproximación para n = 40
forward difference
centered difference
Error (E_n)
nodos equiespaciados (x_j)

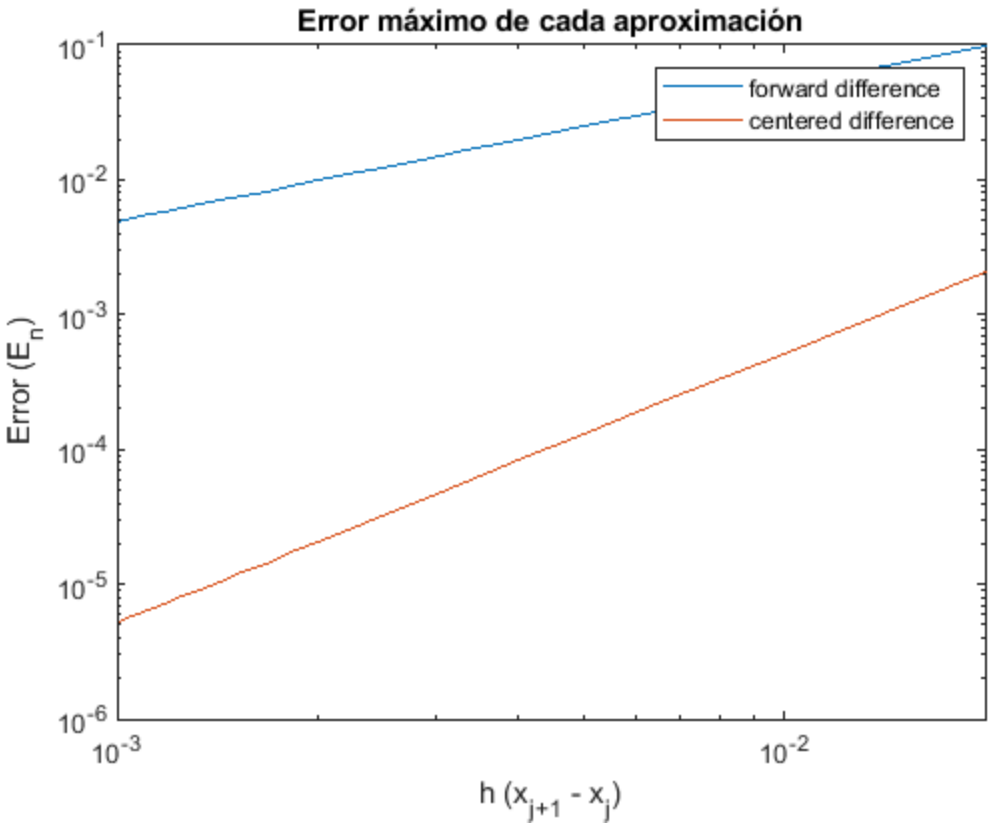

Derivada y sus aproximaciones para n = 80
derivada exacta
forward difference
centered difference
df/dt (x_j)
nodos equiespaciados (x_j)


Error de cada aproximación para n = 80
forward difference
centered difference
Error (E_n)
nodos equiespaciados (x_j)

4) Valor máximo del error frente al h utilizado:

```
%{
En este apartado repetiremos el procedimiento del apartado anterior
pero, a diferencia de este, aproximaremos de la derivada para una
n creciente de 100 en 100. Adicionalmente, calcularemos el error
máximo de cada aproximación y lo representaremos en función de la
h utilizada.
%}

clear all;
format long g;

errorMaxFD = [];
errorMaxCD = [];
haches = [];

for n = 100:100:2000
    xj = linspace(0,2,n+1);

    u = zeros(1,n+1); v = zeros(1,n+1);
    u(1) = -1; v(1) = -1; v(2) = 1;
    FD = toeplitz(u, v);
    FD = FD(1:end-1,:);
    w = zeros(1,n+1); x = zeros(1,n+1);
    w(1) = -1/2; x(1) = -1/2 ; x(3) = 1/2;
    CD = toeplitz(w, x);
    CD = CD(1:end-2,:);

    f = sin(pi.*xj);
    f = f';
    h=abs(xj(1)-xj(2));
    der_fd = ((1/h).*(FD*f))';
    der_cd = ((1/h).*(CD*f))';

    fDer = pi*cos(pi.*xj);
    error_fd = abs(fDer(1:end-1) - der_fd);
    error_cd = abs(fDer(2:end-1) - der_cd);
    %{
    Debemos acumular el error máximo de la aproximación para cada
    n, para luego poder representarlo en función de la h que
    tenga asociada. Para ello también deberemos acumular en un
    vector haches las h utilizadas en cada iteración.
    %}
    errorMaxFD = [errorMaxFD max(error_fd)];
    errorMaxCD = [errorMaxCD max(error_cd)];
    haches = [haches h];
end

%Representamos la información obtenida en el bucle anterior:
loglog(haches, errorMaxFD);
title('Error máximo de cada aproximación');
xlabel('h (x_j_+_1 - x_j)');
```

```
ylabel('Error (E_n)');
hold on;
loglog(haches, errorMaxCD);
legend('forward difference','centered difference');
hold off;
%{
Para interpretar mejor la gràfica anterior, utilizaremos el comando
regression y obtendremos las ecuaciones de las rectas que nos aparecen
al representar el error máximo en función de la h.
%}
[r1,m_fd,~] = regression(log10(haches),log10(errorMaxFD))
[r2,m_cd,~] = regression(log10(haches),log10(errorMaxCD))

%{
Como podemos observar, las dos rectas son de calidad (r~1) y el
pendiente de la que corresponde a la aproximación forward es
m_fd~1, lo que ratifica su convergencia lineal, mientras que
la pendiente de la recta que corresponde a la aproximación
centered es m_cd~2, que nos indica su convergencia cuadrática.

Con esto hemos deducido por qué la centered nos aproxima mejor
que la forward para la misma n, y esto es porque la convergencia
de la primera es cuadrática mientras que la de la segunda es lineal.
%}


r1 =

         0.999999998271828


m_fd =

         0.999930861819295


r2 =

         0.999999999844506


m_cd =

          1.99995851511355
```



Published with MATLAB® R2018b