Table of Contents

P9 : Cortés y García

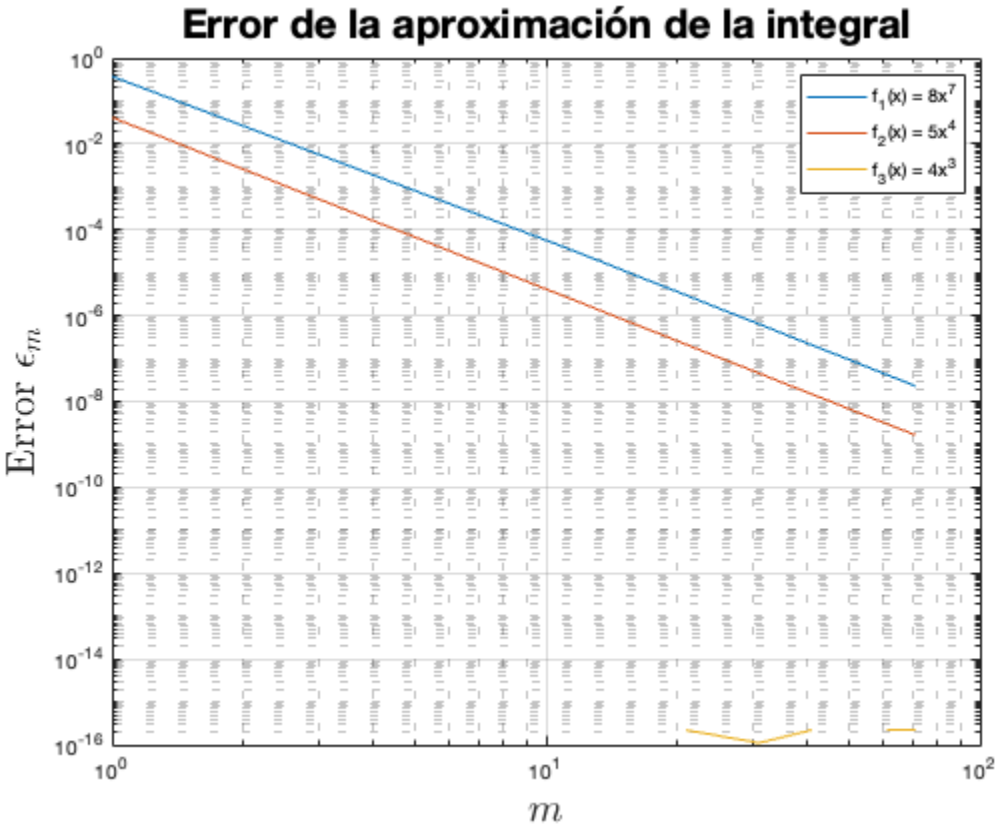
```
clear all
format long
```

Ejercicio 1: Exactitud

```
a = 0; b = 1; % valores de los límites de la integral

% definimos las funciones que queremos integrar
f1 = @(x) 8*x.^7; f2 = @(x) 5*x.^4; f3 = @(x) 4*x.^3;

%vectores que guardaran els errors comesos per cada valor de m
error1 = []; error2 = []; error3 = [];

for mm = 1:10:80
    H = (b-a)/mm;
    k = [0:1:2*mm];
    xk = a + k*H/2;

    % calculamos los diferentes valores que necesitamos para cada
 integral
    % para la primera integral
    f01 = f1(xk(1));
    sum11 = sum(f1(xk(3:2:2*mm-1)));
    sum21 = sum(f1(xk(2:2:2*mm)));
    f2m1 = f1(xk(2*mm+1));

    I1 = H/6*(f01+2*sum11+4*sum21+f2m1);
    error1 = [error1 abs(1-I1)];

    % para la segunda integral
    f02 = f2(xk(1));
    sum12 = sum(f2(xk(3:2:2*mm-1)));
    sum22 = sum(f2(xk(2:2:2*mm)));
    f2m2 = f2(xk(2*mm+1));

    I2 = H/6*(f02+2*sum12+4*sum22+f2m2);
    error2 = [error2 abs(1-I2)];

    % para la tercera integral
    f03 = f3(xk(1));
    sum13 = sum(f3(xk(3:2:2*mm-1)));
```

```
    sum23 = sum(f3(xk(2:2:2*mm)));
    f2m3 = f3(xk(2*mm+1));

    I3 = H/6*(f03+2*sum13+4*sum23+f2m3);
    error3 = [error3 abs(1-I3)];
end

% hacemos un plot a escala logaritmica de los errores cometidos en
 función
% de m en cada integral
figure(1)
loglog([1: 10 :80], error1, 'lineWidth', 1)
hold on
loglog([1: 10 :80], error2, 'lineWidth', 1)
hold on
loglog([1: 10 :80], error3, 'lineWidth', 1)
hold off
grid on
title('Error de la aproximación de la integral', 'Fontsize', 20);
xlabel('$m$', 'Interpreter', 'Latex', 'FontSize', 20);
ylabel('Error $\epsilon_m$', 'Interpreter', 'Latex', 'FontSize', 20);
legend('f_1(x) = 8x^7','f_2(x) = 5x^4','f_3(x) = 4x^3')

% Vemos que el error decae a medida que augmentamos el valor de m.
% Observamos también que el error es menor cuando el grado de f es
 menor.
% Cabe remarcar que en el caso de f de grado menor 4 el error tiene
% directamente a 0 (0 numérico de matlab), ya que como vemos
 analiticamente
% el error es proporcional a la derivada cuarta de la funcion, que en
% estos casos es nula. En los casos de polinomios de grado >= 4, el
 error
% decrece cuando crece m (teoricamente como m^(-4)).

% Comprovamos que efectivamente el error decae como m^(-4) con la
 funcion
% polyfit
polyfit(log10([1: 10 :80]), log10(error1),1)
polyfit(log10([1: 10 :80]), log10(error2),1)

% En cambio, si la usamos para f3, nos da que tanto el pendiente de la
% recta como su ordenada son NaN. Esto nos deja entrever que no tiene
% sentido buscar una regresión lineal para grado 3 (o menor, puesto
 que
% sucedería lo mismo como ya se comprobó en clase analíticamente).
polyfit(log10([1: 10 :80]), log10(error3),1)


ans =

  -3.899214982143437  -0.394531464668239


ans =
```

```
   -3.999999993338158   -1.380211246472189

ans =

   NaN   NaN
```



Ejercicio 2: Cálculo de pi

```
% Definimos la función que queremos integrar
f = @(x) 4./(1 + x.^2);
a = 0; b = 1; % El intervalo de integración es [0, 1]

% Calculamos para m = 1
m = 1;
H = (b-a)/m;
k = [0:2*m];
xk = a + H/2.*k;
I_f = H/6*(f(xk(1)) + 2*sum(f(xk(3:2:2*m-1))) + 4*sum(f(xk(2:2:2*m)))
 + f(xk(2*m+1)));

% A partir del resultado para m = 1, vamos iterando para m's mayores
while abs(pi - I_f) > 1e-13
    m = m+1;
```

```
    % Reaproximamos para el nuevo valor de m
    H = (b-a)/m;
    k = [0:2*m];
    xk = a + (H/2)*k;
    I_f =
 (H/6)*(f(xk(1))+2*sum(f(xk(3:2:2*m-1)))+4*sum(f(xk(2:2:2*m)))+f(xk(2*m
+1)));
end

disp(m)
disp(I_f)

% Podemos comprobar que para m = 43, el valor aproximado de pi es
% 3.141592653589695 y que el error respecto al verdadero valor de pi
 es
% menor que 1e-13 (esto significa que nuestro resultado tiene 14
 cifras
% significativas).
```

```
    43
```

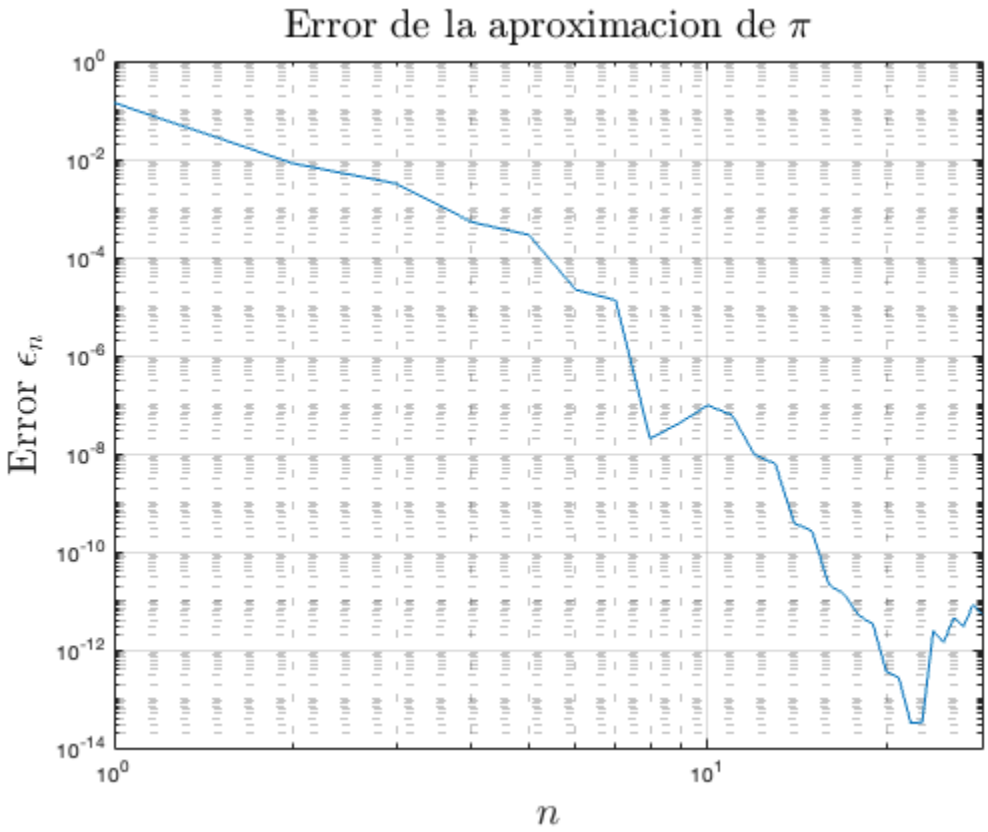
```
   3.141592653589695
```

Ejercicio OPCIONAL: Cálculo de pi

```
f = @(x) 4./(1 + x.^2);
a = 0; b = 1;

error = [];

for n = 1:29
    % Creamos los nodos equiespaciados
    h = (b-a)/n;
    k = [0:n];
    xk = a + h*k;

    % Plantemos la matriz de Vandermonde y el vector de valores I_k
    V = xk.^(k');
    I = 1./(k'+1);

    % Calculamos los pesos
    w = V \ I;

    % Evaluamos la función en cada nodo
    F = f(xk);

    pi_aprox = F*w; % Calculamos la aproximación de pi

    error = [error abs(pi-pi_aprox)];
end
```

```
figure(2)
loglog([1:29], error);
grid on;
title('Error de la aproximacion de $\pi
$', 'Interpreter', 'Latex', 'Fontsize', 20);
xlabel('$n$', 'Interpreter', 'Latex', 'FontSize', 20);
ylabel('Error $\epsilon_n$', 'Interpreter', 'Latex', 'FontSize', 20);

% Vemos que, efectivamente, nos acercamos al valor de pi con un error
% cercano a 1e-14 alrededor de n=20. Sin embargo, también vemos que
 justo
% después el error vuelve a aumentar considerablemente y que para
 valores
% de n anteriores, este no sigue una tendencia "limpia".

% Dado que la función de la que estamos aproximando el cálculo de la
% integral es 1/(1+x^2) y ya hemos visto en sesiones y prácticas
% anteriores que es bastante dada a errores o fallos en la
 convergencia
% debido al fenómeno de Runge. Esto podría arreglarse con otros
 métodos de
% integración numérica.

% Vemos también que la advertencia al calcular el vector de pesos sale
 (de
% hecho aparece para cada n) y nos avisa de que el determinante de V,
% aunque no se anula, se acerca bastante al 0.

% Concluimos que, aunque se trata de una aproximación válida, también
% presenciamos varios fenómenos que dejan entrever que quizás otras
 formas
% de integración numérica serían más útiles.

Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  1.561119e-16.
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  1.968896e-17.
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  2.419987e-18.
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  3.020811e-19.
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  4.041973e-20.
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  5.953735e-21.
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  3.390793e-22.
```

```
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  1.729319e-22.
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  7.125035e-23.
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  3.085872e-22.
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  1.281250e-23.
Warning: Matrix is close to singular or badly scaled. Results may be
 inaccurate.
RCOND =  4.109238e-24.
```



Published with MATLAB® R2020b