===

Práctica 11. Integración numérica

Contenidos

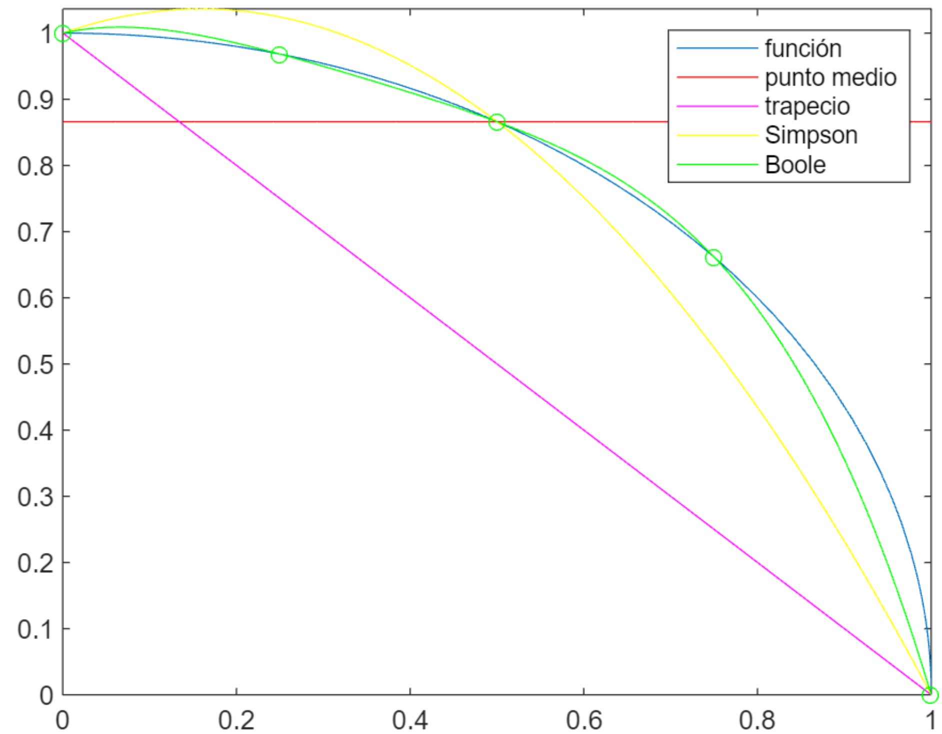
Ejercicio 1. Fórmulas de Newton-Cotes

Consideramos la integral

$$\int_0^1 \sqrt{1-x^2}\,dx.$$

- Calcular el valor que da MATLAB.
- Representar la función en ese intervalo y representar gráficamente los puntos y polinomios que se utilizan para los métodos (simples) (a) del punto medio, (b) de los trapecios, (c) de Simpson y (d) de Boole.

```
intf_matlab =
   0.785398163397448
```



Ejercicio 2. Integración de Romberg

Implementar de manera eficiente la integración de Romberg para conseguir un error de orden $\mathcal{O}(h^{32})$ con $h = 1/2$ y calcular el error absoluto en la integración de la siguiente función:

$$\int_0^1 \sqrt{1 - x^2}\, dx.$$

```
intf_matlab =
   0.785398163397448

int_aprox =
   0.683012701892219

R = 16×16
   0.683012701892219                   0                   0                   0 ···
   0.748927267025610   0.770898788736741                   0                   0
   0.772454786089293   0.780297292443854   0.780923859357662                   0
   0.780813259456935   0.783599417246149   0.783819558899636   0.783865522384429
   0.783775605719283   0.784763054473399   0.784840630288549   0.784856837770912
   0.784824228194921   0.785173769020134   0.785201149989917   0.785206872524859
   0.785195198099154   0.785318854733898   0.785328527114815   0.785330548973941
   0.785326395739308   0.785370128286025   0.785373546522834   0.785374261116612
   0.785372788179914   0.785388252326782   0.785389460596166   0.785389713200505
   0.785389191634755   0.785394659453035   0.785395086594786   0.785395175896351
       ⋮

aprox_romberg =
   0.785398157618301
```

```
error =
    -5.779147826956432e-09
```

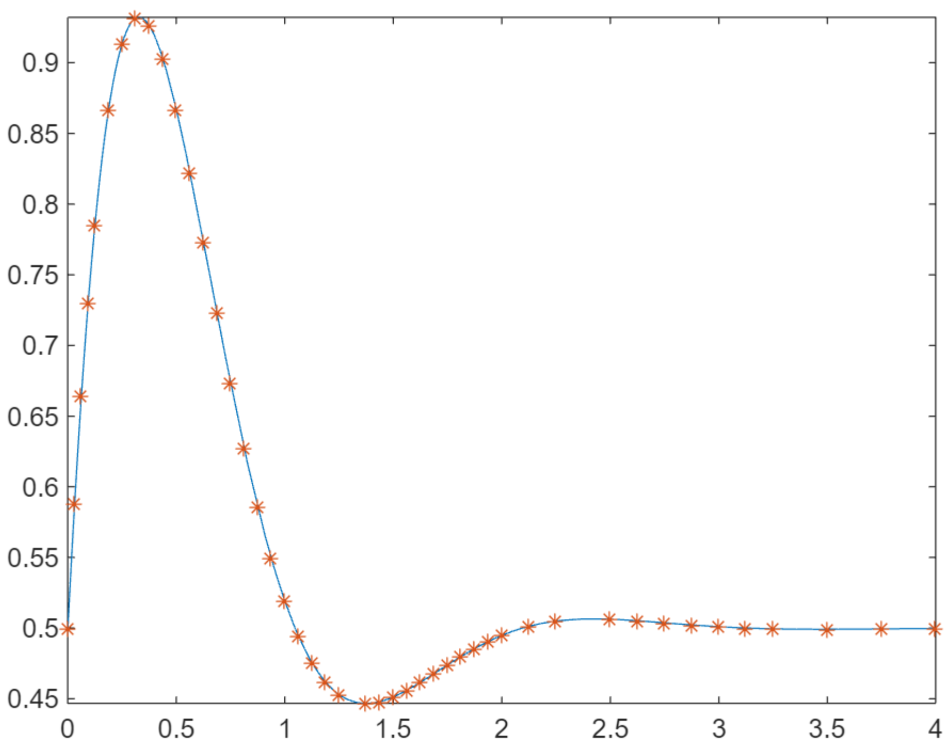
Ejercicio 3. Integración adaptativa

Calcular de manera adaptativa la siguiente integral con tolerancia 10^{-7}. Representar la función y todos los puntos que se han usado para obtenerla.

$$\int_0^4 e^{-2x}\sin(3x) + 0.5dx$$

```
intf_matlab =
   2.230731596609429

tol =
     1.000000000000000e-07
```




```
error =
    -7.585782801911023e-08
```

Ejercicio 4. Integración de Montecarlo

- Estimar π usando el método hit-or-miss con 1000 puntos aleatorios y un círculo de radio unidad. Representar el método y calcular el error absoluto.

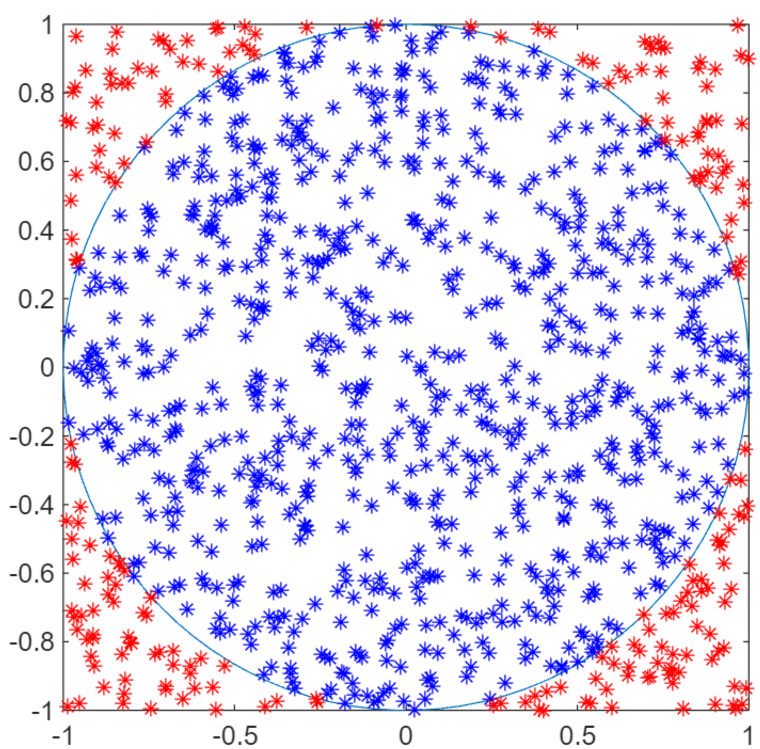
```
rands = 1000×2
   0.744153200657828   0.433840587520583
   0.292952123977180   0.589834907817769
```

```
   0.688413589264656   0.021879628187711
   0.288746312491886   0.483329327655758
   0.100509810711995   0.098461999698944
   0.485313464342827   0.573996042774767
   0.606981092314164   0.785408966703836
   0.493960258939277   0.927377758138399
   0.451577751245957   0.983022620676445
   0.609064660143211   0.946297955636096
      ⋮

rands = 1000×2
   0.488306401315656  -0.132318824958833
  -0.414095752045640   0.179669815635538
   0.376827178529312  -0.956240743624578
  -0.422507375016228  -0.033341344688484
  -0.798980378576011  -0.803076000602112
  -0.029373071314346   0.147992085549534
   0.213962184628329   0.570817933407671
  -0.012079482121447   0.854755516276797
  -0.096844497508087   0.966045241352890
   0.218129320286423   0.892595911272192
      ⋮

int_aproxMontecarlo2 =
   0.772000000000000

pi_aprox =
   3.088000000000000
```



- Usar la integración de Montecarlo por media muestral para aproximar la siguiente integral usando 1000 números aleatorios.

$$\int_1^2 (4 - x^2)^{3/2} dx.$$

Dar el error absoluto.

```
intf_matlab =
   2.386070990149613

int_aprox_Montecarlo_media =
   2.435511830317429

error_aprox_Montecarlo_media =
   0.049440840167817
```

Ejercicio 5. Integración gaussiana

Integrar la siguiente función usando ocho puntos y el método de Gauss-Legendre:

$$\int_0^4 e^{-2x} \sin(3x) + 0.5 dx.$$

Calcular el error absoluto.

Nota: puede ayudar para los pesos calcular los ceros del polonomio de Legendre $P_n(x)$ *que da MATLAB en simbólico (las funciones vpasolve() y legendreP() pueden ayudar para esto). Los pesos vienen dados por la fórmula*

$w_i = \dfrac{2}{\left(1 - x_i^2\right)\left[P_n'(x_i)\right]^2}$ *donde los* x_i *son los nodos. (Cuidado, el polinomio está derivado en esta fórmula.)*

```
nodos = 8×1
  -0.960289856497536
  -0.796666477413627
  -0.525532409916329
  -0.183434642495650
   0.183434642495650
   0.525532409916329
   0.796666477413627
   0.960289856497536

pesos = 8×1
   0.101228536290376
   0.222381034453375
   0.313706645877887
   0.362683783378362
   0.362683783378362
   0.313706645877887
   0.222381034453375
   0.101228536290376

intf_matlab =
   2.230731596609429

int_aprox_gausslegendre =
   2.230728836961660

error_aprox_gausslegendre =
    -2.759647768968421e-06
```

Funciones internas

`Documento preparado por I. Parada, 15 de mayo de 2024`