Resumen de la Guía 0 - Repaso de Cáclulo Numérico

Ruben Weht[1,2] Mariano Forti[1,3]

[1] Instituto de Tecnología Prof. Jorge Sabato
[2]Fisica del Sólido, Edificio TANDAR, weht@cnea.gov.ar, interno 7104
[3]División Aleaciones Especiales, Edificio 47 (microscopía), mforti@cnea.gov.ar, interno 7832

1. Ordenamiento: Método de Burbujeo

2. Expansión en serie, tolerancia y error

En este ejercicio pretenemos conocer qué tan bien podemos aproximarnos al valor de la función exponencial con la expansión en serie alrededor de cero. La Ecuación 1 define tanto la serie hasta su `n-esimo` término como el error cometido por la serie truncada.

$$\begin{aligned} \exp x &= \sum_{i=0}^{n} \frac{x^i}{i!} + ERR[n] \\ ERR[n] &= \left| \frac{exp(x_o) - \sum_{i=0}^{n} \frac{x_o^i}{i!}}{exp(x_o)} \right| \end{aligned} \tag{1}$$

El valor de `n` indica la cantidad de términos que tomamos en nuestra expansión, la cual dependerá de que tan "tolerantes" seamos con nuestro programa. Por ejemplo, una "tolerancia" de $1 \cdot 10^{-4}$. En el código de la Figura 1 se observa cómo se va incrementando el valor de `n` hasta que el

```python
def miexp(x,n):
    serie = 0
    for i in range (n+1):
        serie = serie + (x**i)/factorial(i)
    return serie


def error(xo, intn):
    return abs(miexp(xo, intn) - np.exp(0.5))/abs(np.exp(0.5))


then = 1
ERR = []
ERR.append(np.array(error(0.5, then)))
while ERR[-1] >= 1e-4:
    then = then+1
    ERR.append(error(0.5, then))
```

Figura 1: Implementación en Python de la serie truncada y el cáclulo del error cometido respecto de $x_o = 0{,}5$

error es menor que dicha tolerancia. Si bien esta medida no debe hacerse cada vez que se resuelve el problema, sí es necesario indicar el último término de truncamiento y una estimación del error cometido.

3. Multiplicación de Matrices

Este problema se incluye solamente como ejercicio de memoria. La multiplicación de matrices y en particular el concepto de la multiplicación de una matriz por un vector columna será uno de los cimientos de nuestra materia.

Recordemos simplemente,

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,n}x_n \\ a_{2,1}x_1 + a_{2,2}x_2 + a_{2,n}x_n \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + a_{n,n}x_n \end{pmatrix} \tag{2}$$

Sin embargo, cuando implementemos una multiplicación de matrices, no vamos a usar un programa casero. Por el contrario, usaremos las herramientas disponibles. por ejemplo, en *Matlab* tenemos el operador $*$, en *python*

usaremos la funcuón `np.matmul`. Sin embargo, debemos ser cuidadosos con la construcción del problema, las matrices y vectores deben tener tamaños correctos para participaren la operación.

4. Problema de Algebra Lineal

Aquí tenemos un muy simple problema de álgebra lineal. El objetivo del mismo es simplemente introducir las herramientas que nos permitiran resolverlo. De nunguna manera se prentende que el alumno programe la solución .

Nuestro problema es sencillo,

$$\begin{cases} x - 3y - z = 6 \\ 2x - 4y - 3z = 8 \\ -3x + 6y + 8z = -5 \end{cases} \tag{3}$$

Matricialmente:

$$\underbrace{\begin{pmatrix} 1 & -3 & -1 \\ 2 & -4 & -3 \\ -3 & 6 & 8 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x \\ y \\ x \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} 6 \\ 8 \\ -5 \end{pmatrix}}_{B} \tag{4}$$

La solución es inmediata:

$$X = A^{-1}B \tag{5}$$

La implementación implica usar las funciones disponibles en nuestras herramientas. Como vimos en el apunte de *introducción a la programación*, disponemos de la funcion `numpy.linalg.solve` en *Python* o del operador barra invertida (\ ) en *matlab*.

5. Splines

Este problema ya ha sido tratado en el apunte "Cuentas de Interpolación". Aquí usaremos los resultados que habíamos obtenido allí. simplemente recordaremos que tenemos una serie de puntos experimentales por lo que queremos

hacer pasar una función definida a tramos.

En cada intervalo I_i vale el polinomio f_i de grado 3. A estos polinomios se les pide que sean continuos y que la primer y segunda derivada. Eso desemboca en un sistema de ecuaciones que termina en un sistema lineal para los coeficientes b_i .

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ h_1 & 2(h_2+h_1) & h_2 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & h_2 & 2(h_3+h_2) & h_3 & \dots & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \cdots & h_{N-3} & 2(h_{N-2}+h_{N-3}) & h_{N-2} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & h_{N-2} & 2(h_{N-1}+h_{N-2}) & h_{N-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{N-2} \\ b_{N-1} \\ b_N \end{pmatrix} = 3 \begin{pmatrix} 0 \\ \frac{Y_3 - Y_2}{h_2} - \frac{Y_2 - Y_1}{h_1} \\ \vdots \\ \frac{Y_N - Y_{N-1}}{h_{N-1}} - \frac{Y_{N-1} - Y_{N-2}}{h_{N-2}} \\ 0 \end{pmatrix} \tag{7}$$

Luego, al resolver el sistema de ecuaciones que sale de las condiciones de borde para cada f_i , tenemos las soluciones para a_i y c_i a partir de los b_i

$$\begin{aligned} d_i =& Y_i \\ a_i =& \frac{1}{3}\frac{b_{i+1} - b_i}{h_i} \\ c_i =& \frac{Y_i - Y_{i-1}}{h_{i-1}} - b_{i-1}h_{i-1} - a_{i-1}h_{i-1}^2 \end{aligned} \tag{8}$$

Con estos datos podemos completar la escritura de los $N-1$ polinomios necesarios. De esta manera, podemos graficar en cada intervalo el tramo correspondiente de la función como se muestra en la Figura 2. Debe notarse que cada polinomio vale solo en su intervalo de definición, como se observa en la Figura 3. Fuera de su propio intervalo, cada polinomio constituye una aproximación pobre a los datos originales.

El problema no ha terminado aún. El Enunciado pide, en pocas palabras, encontrar la profundidad a la que se minimiza la derivada primera de la función, que será cuando la derivada segunda se anule. Hay varias maneras de resolver esto.

En primera instancia, uno podría pensar que la derivada estara bien representada por la derivada de los polinomios. Esta hipótesis se maneja en la Figura 4. El problema es que los polinomios son a lo sumo dos veces derivables, pero lasegunda derivada si bien es continua es “aserrada” de manera

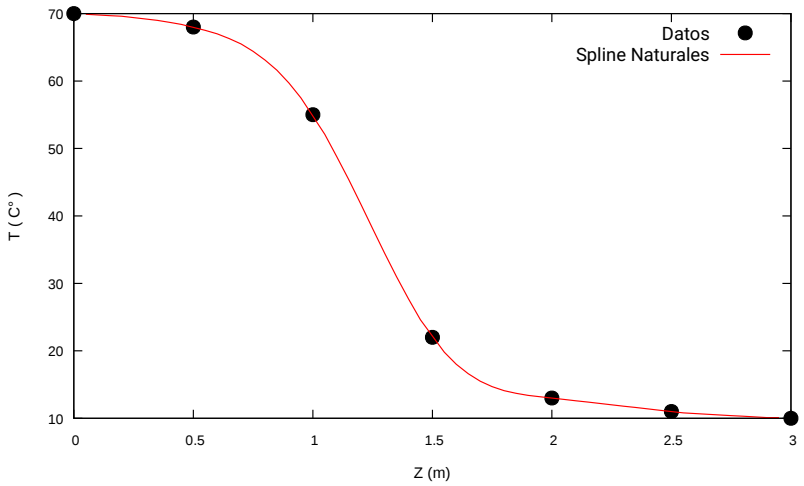


Figura 2: Función de Interpolación Definida a Tramos

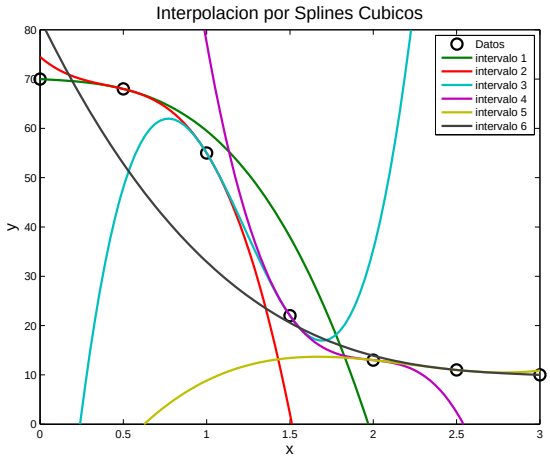


Figura 3: Cada polinomio vale solo en su intervalo

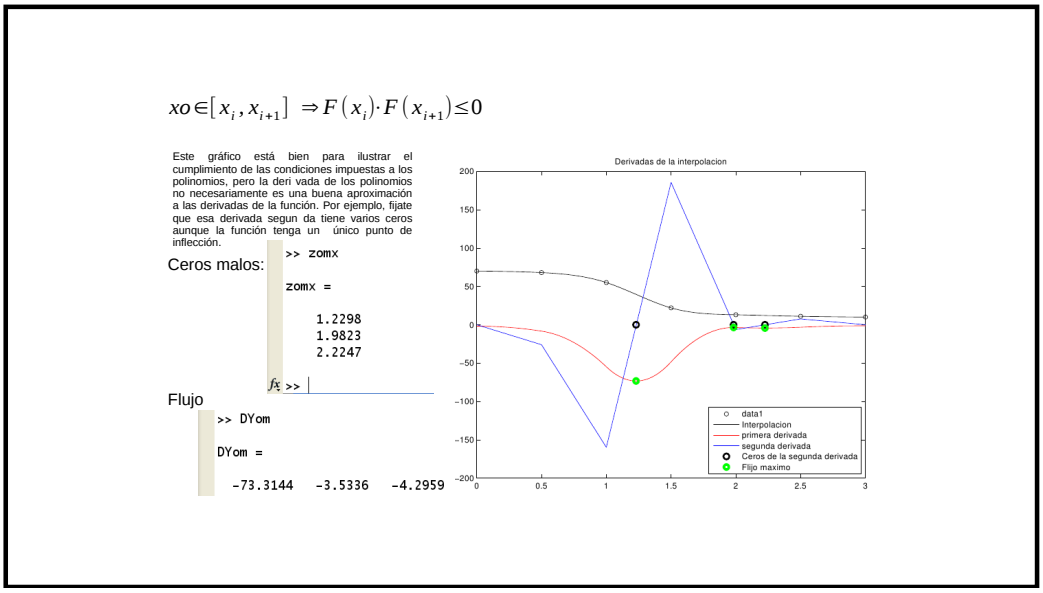


Figura 4:

que la aproximación a su cero puede ser tan buena o mala como nuestra tolerancia a los errores.

Otra forma de resolver el problema seria derivar numéricamente los datos como se muestra en la Figura 5. Notar que con este truco se obtienen derivadas suaves, pero las derivadas numéricas tienen su error intrínseco y nuevamente es nuestra tolerancia a los errores la que determinará la confianza que le tengamos al resultado.

no lo hemos dicho, pero el lugar donde se anula la derivada segunda podemos encontrarlo usando algún algoritmo de optimización. Por ejemplo el método de bisección.

6. Integración Numérica

Este problema esta incluido principalmente por razones históricas, pero sirve para ilustrar algunos conceptos. Como se ha visto en la teórica, la integral de una función $f(x)$ en el intervalo $[a, b]$ puede aproximarse por distintos métodos, resumidos en la Figura 6.

Los métodos de Trapecios y Simpson dividen el intervalo de integración en N intervalos de igual tamaño. *Trapecios* aproxima el área bajo la curva por un trapecio, mientras que *Simpson* aproxima la integral en dos intervalos

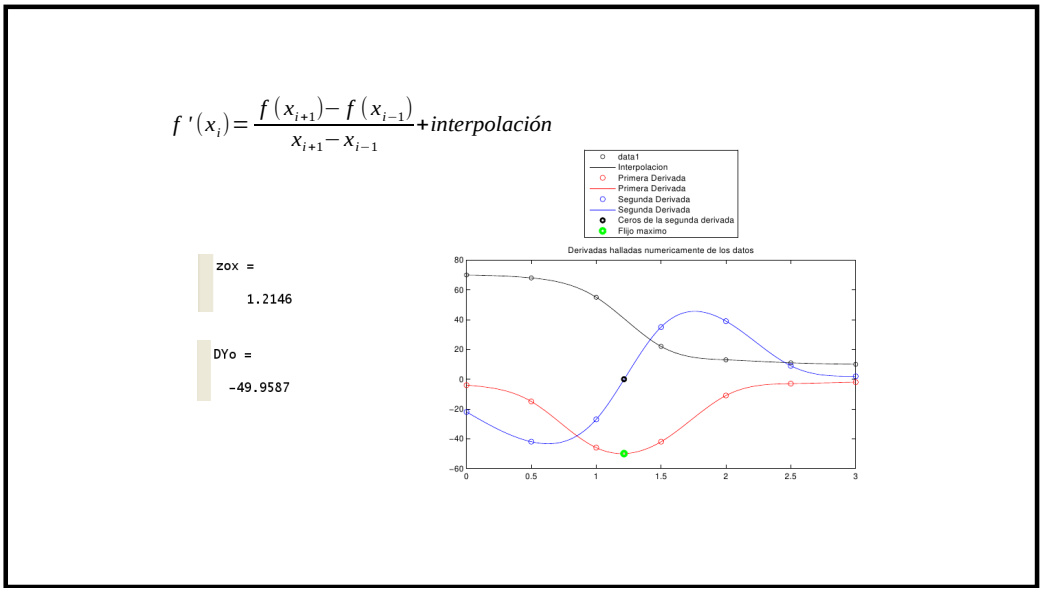


Figura 5: Solucion del problema al derivar numéricamente los datos

consecutivos como la integral bajo un polinomio que pasa por los los puntos definidos por la función. De ahí surge que para aplicar este método es necesario definir un número de intervalos con una paridad definida.

El método más novedoso para nosotros es el de cuadraturas de gauss. En lugar de definir un intervalos regulares para subdividir el intervalo de integración, la idea es definir un conjunto de puntos t_i con $i \in [1, N]$ elegidos de manera que para cualquier polinomio $p_N(t)$ de grado N se tiene la mejor aproximación a la integral

$$\int_a^b p_N(t)dt = \sum_i^N W_i p_N(t_i) \tag{9}$$

La condición de minimizar el error conduce a un sistema de ecuaciones sobre los factores W_i que pueden entenderse de la siguiente manera: se han definido pesos W_i y puntos t_i que aproximan a cualquier función $f(t)$ minimizando el error en la integral. Otra ventaja es que una vez resuelto el sistema de ecuaciones de la Figura 6 los valores obtenidos valen para cualquier función f . Los métodos de Simpson y Trapecios requerían hacer numerosas evaluaciones de la función. Las cuadraturas de Gauss minimizan también la cantidad de evaluaciones.

Una última cosa a tener en cuenta, es que el sistema de ecuaciones para los W_i, t_i es única e independiente de la función a integrar. La solución de ese

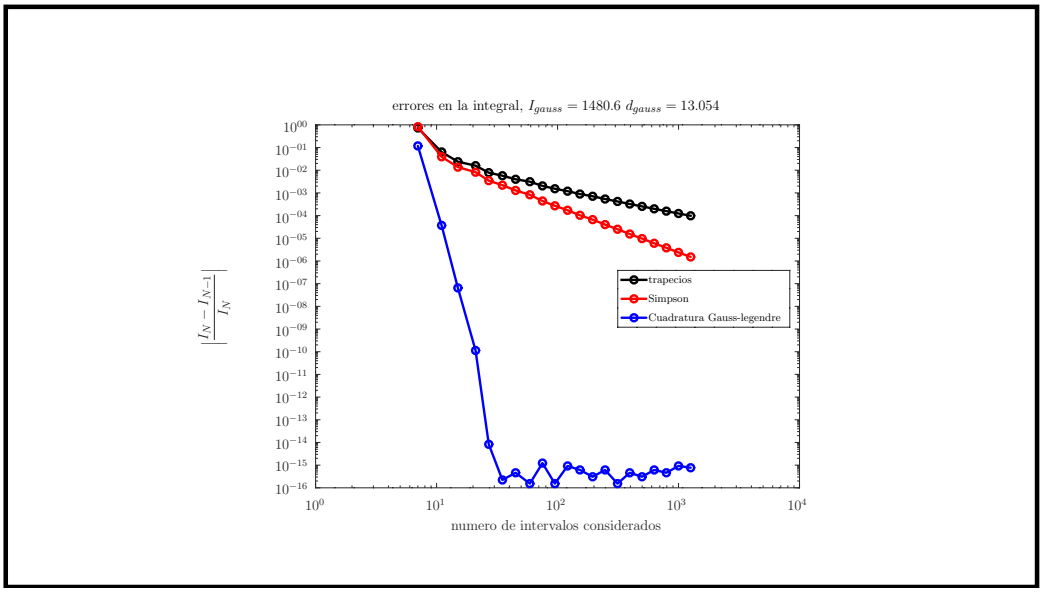


Figura 6:

sistema de ecuacionies se alcanza con los polinomios de Chebyshev-Gauss-Lobatto y existen scripts donde se resuelven paravarios lenguajes.

En la **??** se observa el comportamiento del error para los distintos métodos de integración en función del número de intervalos , que corresponde con el número de evaluaciones de la función.Se ve claramente que las cuadraturas de Gauss alcanzan rápidamente errores muy pequeños.

De hecho, para las cuadraturas de gauss el error satura en el épsilon de la máquina cuando para los otros métodos apenas se alcanzó un error razonablemente bajo.

7. Ecuaciones Diferenciales Ordinarias

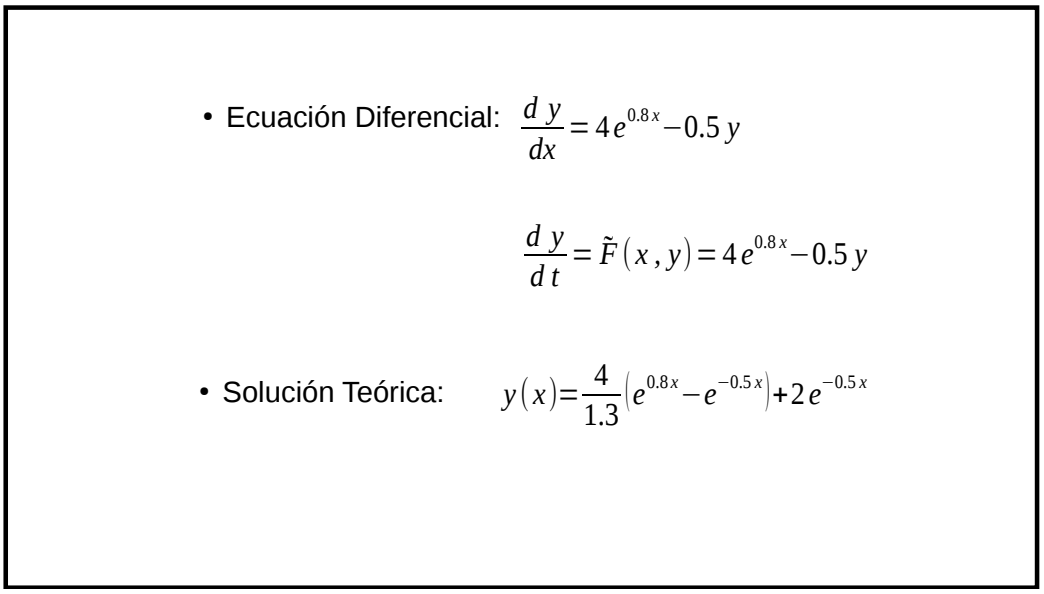


Figura 7: Para resolver el problema necesitamos escribir la ecuación diferencial a resolver de manera de reconocer la forma $\partial y/\partial x = F(x, y)$

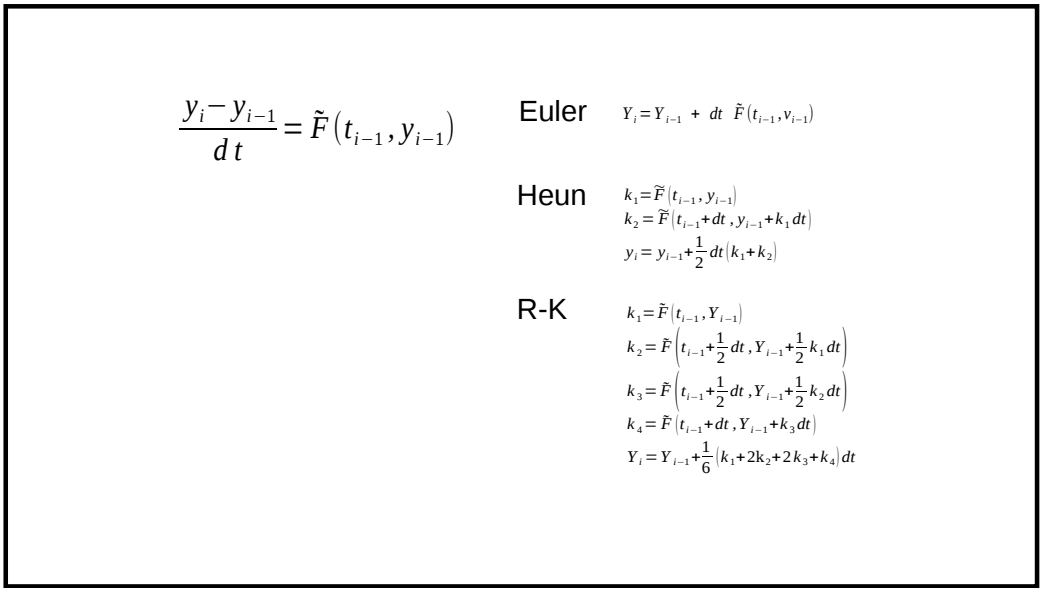


Figura 8: Los distintos métodos realizan aproximaciones sucesivas al paso siguiente en función del paso acutal. Al aumentar la precision de la aproximación se necesitan varias evaluaciones de la función.

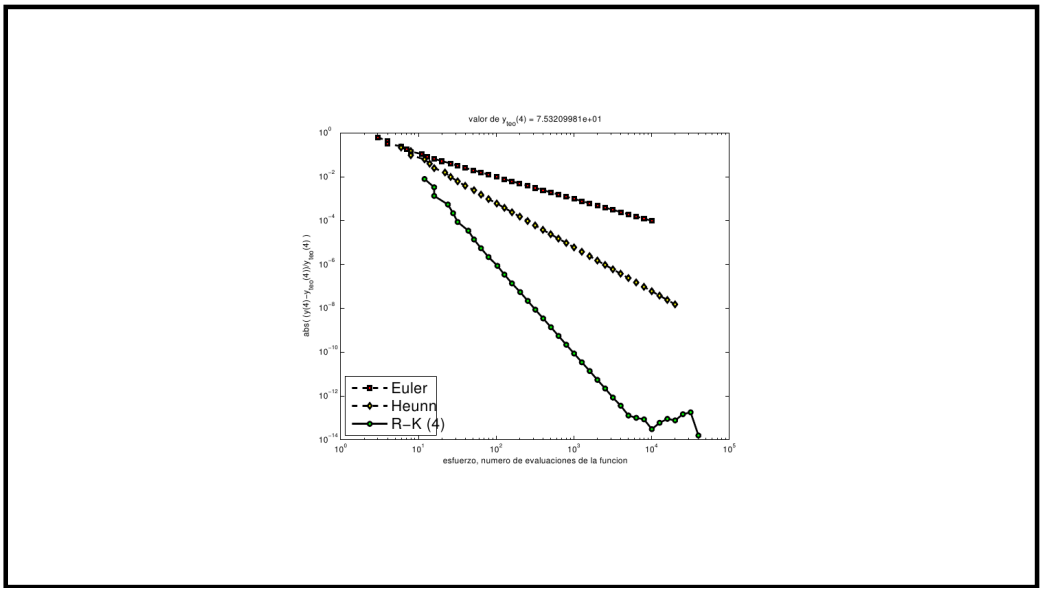


Figura 9: Como se ve aquí, a pesar de necesitar mas evaluaciones de la función, el método de Runge Kutta de orden 4 alcanza precisiones mucho mas altas que los otros métodos.

Resumen de la Guía 0 - Repaso de Cáclulo Numérico

Modelización de Propiedades y Procesos 2019

Ruben Weht[1,2] Mariano Forti[1,3]

[1]Instituto de Tecnología Prof. Jorge Sabato

[2]Fisica del Sólido, Edificio TANDAR, weht@cnea.gov.ar, interno 7104

[3]División Aleaciones Especiales, Edificio 47 (microscopía), mforti@cnea.gov.ar, interno 7832

Resumen de la Guía 0 - Repaso de Cáclulo Numérico

Ordenamiento: Método de Burbujeo

Resumen de la Guía 0 - Repaso de Cáclulo Numérico

Expansión en serie, tolerancia y error

Aproximación a la exponencial

$$\exp x = \sum_{i=0}^{n} \frac{x^i}{i!} + ERR[n]$$

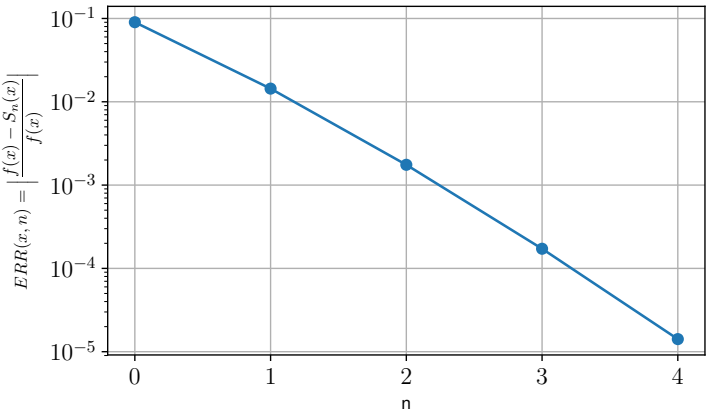
$$ERR[n] = \left| \frac{exp(x_o) - \sum_{i=0}^{n} \frac{x_o^i}{i!}}{exp(x_o)} \right|$$

Implementación de una expansion en serie truncada

```
def miexp(x,n):
    serie = 0
    for i in range (n+1):
        serie = serie + (x**i)/factorial(i)
    return serie


def error(xo, intn):
    return abs(miexp(xo, intn) - np.exp(0.5))/abs(np.exp(0.5))


then = 1
ERR = []
ERR.append(np.array(error(0.5, then)))
while ERR[-1] >= 1e-4:
    then = then+1
    ERR.append(error(0.5, then))
```

Comportamiento del error con N

$$S_n(x) = \sum_{i=1}^{n} \frac{x^i}{i!}$$



Resumen de la Guía 0 - Repaso de Cáclulo Numérico

Multiplicación de Matrices

Multiplicación Matriz $\times$ Vector Columna

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,n}x_n \\ a_{2,1}x_1 + a_{2,2}x_2 + a_{2,n}x_n \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + a_{n,n}x_n \end{pmatrix}$$

Resumen de la Guía 0 - Repaso de Cáclulo Numérico

Problema de Algebra Lineal

Problema de Algebra Lineal

$$\begin{cases} x - 3y - z = 6 \\ 2x - 4y - 3z = 8 \\ -3x + 6y + 8z = -5 \end{cases}$$

Matricialmente:

$$\underbrace{\begin{pmatrix} 1 & -3 & -1 \\ 2 & -4 & -3 \\ -3 & 6 & 8 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x \\ y \\ x \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} 6 \\ 8 \\ -5 \end{pmatrix}}_{B}$$

La solución es inmediata:

$$X = A^{-1}B$$

Resumen de la Guía 0 - Repaso de Cáclulo Numérico

Splines

Splines

$$(X_i, Y_i)_{i=1...N}$$

$$I_i = [X_i, X_{i+1}]$$

$$f_i = a_i(x - X_i)^3 + b_i(x - X_i)^2 + c_i(x - X_i) + d_i$$

Sistema De Ecuaciones para Splines

$$
\begin{pmatrix}
1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\
h_1 & 2(h_2+h_1) & h_2 & 0 & \dots & 0 & 0 & 0 & 0 \\
0 & h_2 & 2(h_3+h_2) & h_3 & \dots & 0 & 0 & 0 & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & 0 & 0 & \cdots & h_{N-3} & 2(h_{N-2}+h_{N-3}) & h_{N-2} & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & h_{N-2} & 2(h_{N-1}+h_{N-2}) & h_{N-1} \\
0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1
\end{pmatrix}
\times
\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{N-2} \\ b_{N-1} \\ b_N \end{pmatrix}
= 3
\begin{pmatrix}
0 \\
\frac{Y_3 - Y_2}{h_2} - \frac{Y_2 - Y_1}{h_1} \\
\vdots \\
\frac{Y_N - Y_{N-1}}{h_{N-1}} - \frac{Y_{N-1} - Y_{N-2}}{h_{N-2}} \\
0
\end{pmatrix}
$$

Coeficientes a_i y c_i

$$
\begin{aligned}
d_i &= Y_i \\
a_i &= \frac{1}{3}\frac{b_{i+1} - b_i}{h_i} \\
c_i &= \frac{Y_i - Y_{i-1}}{h_{i-1}} - b_{i-1}h_{i-1} - a_{i-1}h_{i-1}^2
\end{aligned}
$$

Resultado Splines

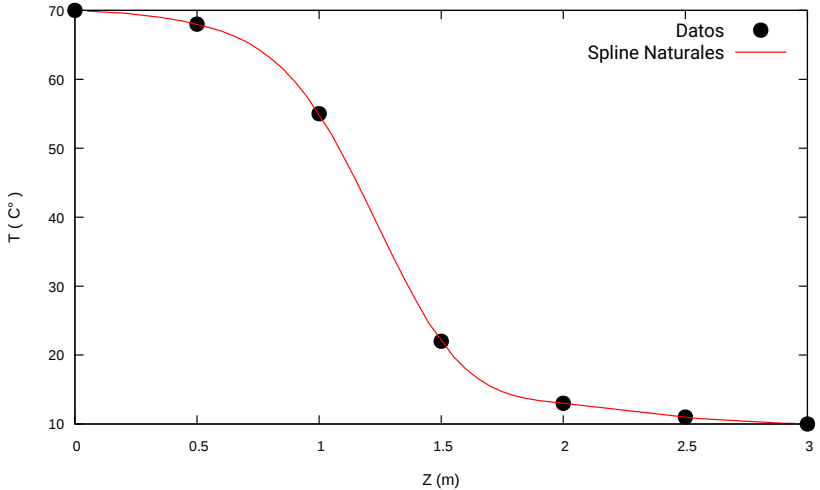


Figura: Función de Interpolación Definida a Tramos

Splines en todo el intervalo

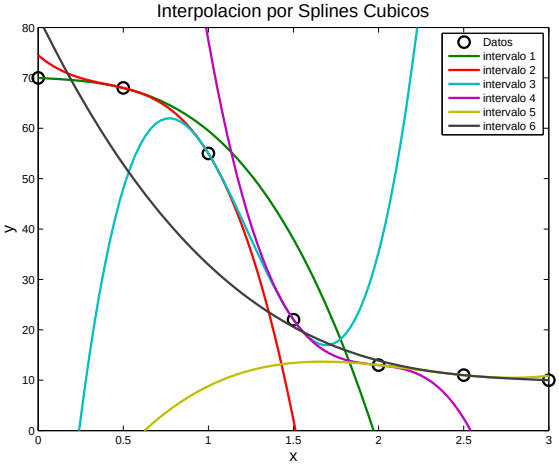


Figura: Cada polinomio vale solo en su intervalo

Derivar polinomios

$$xo \in [x_i, x_{i+1}] \Rightarrow F(x_i) \cdot F(x_{i+1}) \leq 0$$

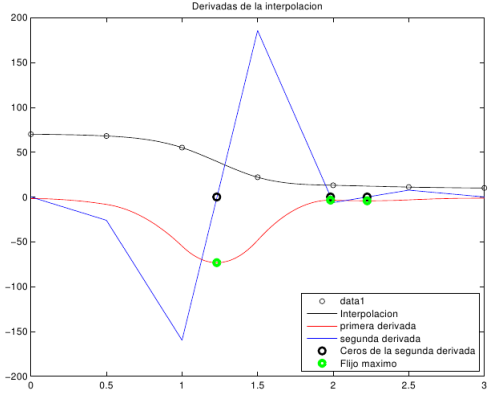
Este gráfico está bien para ilustrar el cumplimiento de las condiciones impuestas a los polinomios, pero la deri vada de los polinomios no necesariamente es una buena aproximación a las derivadas de la función. Por ejemplo, fijate que esa derivada segun da tiene varios ceros aunque la función tenga un único punto de inflección.

Ceros malos:

```
>> zomx

zomx =

    1.2298
    1.9823
    2.2247

>>
```

Flujo

```
>> DYom

DYom =

  -73.3144   -3.5336   -4.2959
```

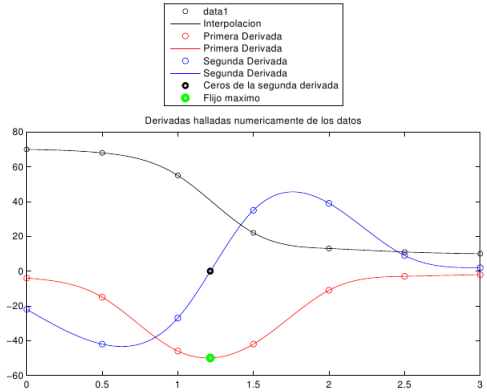


Interpolar la derivada numérica de los datos

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}} + interpolación$$

```
zox =

    1.2146
```

```
DYo =

  -49.9587
```



Resumen de la Guía 0 - Repaso de Cáclulo Numérico

Integración Numérica

Métodos de Integración

$$I^{Trapecios}_{N\ intervalos} = \frac{1}{2}\left(\frac{b-a}{N}\right)\left(f(a)+2\sum_{j=2}^{N} f(x_j)+f(b)\right)$$

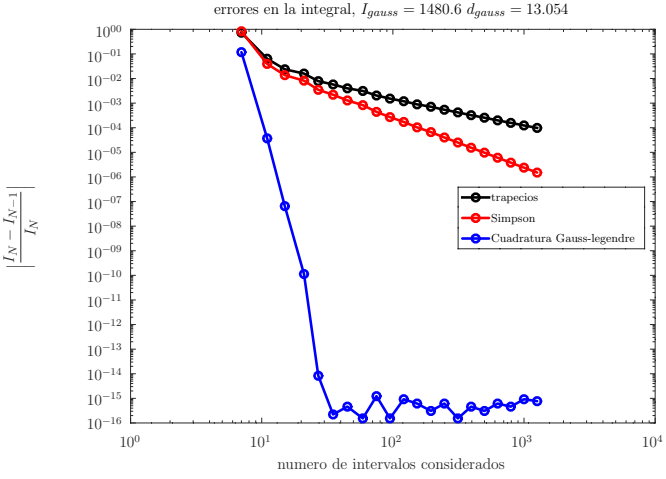
$$I^{Simpson}_{N\ intervalos} = \frac{1}{3}\left(\frac{b-a}{N}\right)\left(f(a)+4\sum_{j=3,\,impar}^{N} f(x_j) \;+\; 2\sum_{j=2,\,pares}^{N-1} f(x_j)+f(b)\right)$$

$$I^{Gauss}_{N\ intervalos} = \sum_{j=1}^{N+1} W_j\, f(t_j)$$

$$\begin{cases} \sum_{j=1}^{N+1} W_j = \int_{-1}^{1} dt \\ \sum_{j=1}^{N+1} W_j t_j = \int_{-1}^{1} t\,dt \\ \sum_{j=1}^{N+1} W_j t_j^2 = \int_{-1}^{1} t^2\,dt \\ \vdots \\ \sum_{j=1}^{N+1} W_j t_j^{2N+1} = \int_{-1}^{1} t^{2N+1}\,dt \end{cases}$$

Subrutina lgwt de matlabcentral

Métodos de Integración



Resumen de la Guía 0 - Repaso de Cáclulo Numérico

Ecuaciones Diferenciales Ordinarias

Ecuaciones Diferenciales Ordinarias

- Ecuación Diferencial: $\frac{dy}{dx} = 4e^{0.8x} - 0.5y$

$$\frac{dy}{dt} = \tilde{F}(x, y) = 4e^{0.8x} - 0.5y$$

- Solución Teórica: $y(x) = \frac{4}{1.3}\left(e^{0.8x} - e^{-0.5x}\right) + 2e^{-0.5x}$

EDO - Métodos de resolución

$$\frac{y_i - y_{i-1}}{dt} = \tilde{F}\left(t_{i-1}, y_{i-1}\right)$$

Euler

$$Y_i = Y_{i-1} + dt \;\; \tilde{F}\left(t_{i-1}, v_{i-1}\right)$$

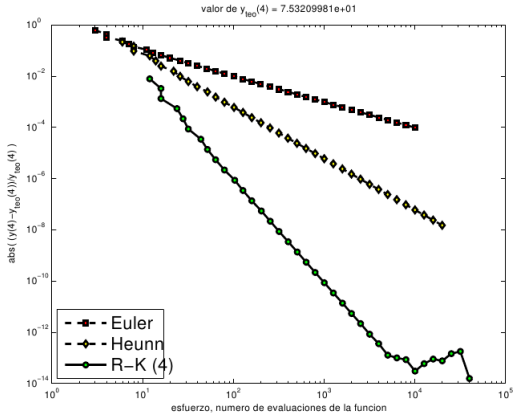
Heun

$$k_1 = \widetilde{F}\left(t_{i-1}, y_{i-1}\right)$$
$$k_2 = \widetilde{F}\left(t_{i-1} + dt, y_{i-1} + k_1 dt\right)$$
$$y_i = y_{i-1} + \frac{1}{2} dt \left(k_1 + k_2\right)$$

R-K

$$k_1 = \tilde{F}\left(t_{i-1}, Y_{i-1}\right)$$
$$k_2 = \tilde{F}\left(t_{i-1} + \frac{1}{2} dt, Y_{i-1} + \frac{1}{2} k_1 dt\right)$$
$$k_3 = \tilde{F}\left(t_{i-1} + \frac{1}{2} dt, Y_{i-1} + \frac{1}{2} k_2 dt\right)$$
$$k_4 = \tilde{F}\left(t_{i-1} + dt, Y_{i-1} + k_3 dt\right)$$
$$Y_i = Y_{i-1} + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) dt$$

EDO - Errores


valor de y_teo(4) = 7.53209981e+01
abs((y(4)-y_teo(4))/y_teo(4))
esfuerzo, numero de evaluaciones de la funcion
Euler
Heunn
R–K (4)