Obligatorio 2

Araújo, Ignacio CI:5.103.875-1
Assereto, Camila CI:5.183.140-8
Bernardez, Santiago CI:5.165.323-2
Brun, Facundo CI:5.265.175-2
Gilmet , Lucía CI: 5.147.832-5
Grupo 66 - Tutora: Juliana Faux
Obligatorio 2 - Métodos Numéricos 2022.

16 de noviembre de 2022

Índice

1. Introducción

El problema planteado en el obligatorio es el de aproximar, mediante diferentes métodos, la solución de un Problema de Valores Iniciales en particular:

$$y'(t) = y^2(t)(1 - 3t) - 3y(t) \tag{1}$$
$$y(-1) = y_0 \tag{2}$$

Para esto se nos presentan cuatro diferentes condiciones iniciales.

Más allá de cada caso particular, es posible encontrar una solución general exacta para la ecuación diferencial planteada.

Para resolverlo de manera exacta la ecuación, primero observamos que es una ecuación diferencial de Bernoulli, es decir, se pueden reconocer elementos de manera que la ecuación cumple la forma de:

$$\frac{dy(t)}{dt} + P(x)y(t) = Q(x)y(t)^{\alpha} \tag{3}$$

De esta manera, se puede plantear:

$$\frac{dy(t)}{dt} = (-3t + 1)y(t)^2 - 3y(t) \tag{4}$$

Y aplicar el cambio de variable sugerido por Bernoulli que en este caso sería $u(t) = \frac{1}{y(t)}$, obteniendo:

$$\frac{du(t)}{dt} = 3u(t) + 3t - 1 \tag{5}$$

Esta última ecuación es significativamente más sencilla que resolver que la inicial. Al resolverla encontramos que:

$$u(t) = Ce^{3t} - t \tag{6}$$

Posteriormente, se puede deshacer el cambio de variable, obteniendo la solución general de la ecuación diferencial original. Esta es:

$$y(t) = \frac{1}{Ce^{3t} - t} \tag{7}$$

2. Marco teórico

Ecuaciones diferenciales

Una ecuacion diferencial es una ecuación que relaciona las derivadas de una o más variables dependientes respecto a una o más variables independientes.

En particular, una ecuación diferencial ordinaria(EDO) es una ecuación diferencial que relaciona una función desconocida de una única variable independiente con sus derivadas.

Dada una función $f : R^2 \to R$, el Problema de Valores Iniciales consiste en hallar la función $y = y(x)$ tal que:

$$y'(x) = f(x, y(x)) \tag{8}$$
$$y(x_0) = y_0 \in \Re \tag{9}$$

Para el Problema de valores iniciales planteado anteriormente, se puede usar el método de Euler para aproximar la derivada en el punto x_k mediante el cociente de la diferencia de dos pasos consecutivos y_{k+1} e y_k al considerar el paso h. De esta manera se obtiene una escuación en diferencias:

$\frac{y_{k+1}-y_k}{h} \approx y'(x_k) = f(x_k, y(x_k)) \approx f(x_k, y_k)$

A partir de esto, el método de Euler hacia adelante consiste de la iteración:
(Euler hacia adelante): $\begin{cases} y_{k+1}^E = y_k^E + hf\left(x_k, y_k^E\right) \\ y_0^E = y_0 \end{cases}$

Para el mismo Problema de Valores iniciales se tiene el método del trapecio, en el cual se obtiene la siguiente identidad de integrar en el intervalo $[x_k, x_{k+1}]$. En este caso, la integral se aproxima con un trapecio de bases $f(x_k, y_k)$ y $f(x_{k+1}, y_{k+1})$, llegando a la siguiente relación:

$y_{k+1} - y_k = \frac{h}{2}\left[f(x_k, y_k) + f(x_{k+1}, y_{k+1})\right.$

De manera que la iteración del método del Trapecio es:

(Trapecio): $\begin{cases} y_{k+1}^T = y_k^T + \frac{h}{2}\left[f\left(x_k, y_k^T\right) + f\left(x_{k+1}, y_{k+1}^T\right)\right] \\ y_0^T = y_0 \end{cases}$

Para resolver métodos explicitos como lo es el método del Trapecio, lo que hacemos es usar el método PREDICTOR-CORRECTOR, el cual para determinar Y_{k+1} utiliza la ecuación de punto fijo : $x = g(x)$ en la que Y_{k+1} toma el rol de x. Llamamos Predictor de este metodo,al valor inicial que comienza la iteración, generalmente se utiliza un metodo explicito simple para hallarlo (por ejemplo un paso de Euler), tal que $Y_{k+1}^0 = Y_k + h/2[f(X_k, Y_k) + f(X_{k+1}, Y_{k+1}^0]$ Mientras que definimos al Corrector como la iteracion que cumple $Y_{k+1}^{n+1} = y_k + h/2[f(X_k, Y_k) + f(X_{k+1}, Y_{k+1}^n)]$ siendo los valores con indice k conocidos del paso anterior.

Por otro lado, cuando en el método del Trapecio no se realizan iteraciones de punto fijo y la predicción se hace con Euler, se le llama método de Heun y su iteración es la siguiente:

(Heun): $\begin{cases} y_{k+1}^H = y_k^H + \frac{h}{2}\left[f\left(x_k, y_k^H\right) + f\left(x_{k+1}, y_k^H + hf\left(x_k, y_k^H\right)\right)\right] \\ y_0^H = y_0 \end{cases}$

Interpolación

La interpolación con Splines resuelve el problema de encontrar una función que interpole un conjunto de datos, y que además tenga derivada segunda continua. Para este fin, realizaremos una interpolación a trozos, utilizando un polinomio cúbico $s_i(x)$ en cada intervalo $I_i = [xi, xi + 1]$, con $i = 0, 1, ..., n - 1$.

Esta interpolación a trozos debe ser tal que, .ªl pegar"los polinomios cúbicos de los distintos intervalos, se obtenga una curva dos veces derivable en todos los puntos, y con derivada segunda continua. Para lograr esto, al polinomio de cada intervalo se le impondrá no sólo por dónde debe pasar en sus extremos, sino además con una pendiente y una concavidad tal que se solapen con la pendiente y concavidad de los polinomios vecinos.

De tal forma que dos splines adyacentes quedan relacionados por las siguientes ecuaciones:
* $s_{i-1}(x_{i-1}) = y_{i-1}\ 1 \leq i \leq n$
* $s_{i-1}(x_i) = y_i\ 1 \leq i \leq n$
* $s'_{i-1}(x_i) = s'_{i-1}(x_i)\ 1 \leq i \leq n-1$
* $s''_{i-1}(x_i) = s''_{i-1}(x_i)\ 1 \leq i \leq n-1$

De esta forma,entre cada par de puntos $[x_i, x_{i+1}]$, se tiene un polinomio cúbico con cuatro parámetros a determinar(sus coeficientes): $s_i(x) = \alpha_{i0} + \alpha_{i1}x + \alpha_{i2}x^2 + \alpha_{i3}x^3$. Por lo tanto, se tienen en total 4 incógnitas.

Por otro lado, contando la cantidad de restricciones impuestas por cada item, se tiene: n por pasar por el extremo izquierdo del intervalo, n por pasar por el extremo derecho del intervalo, n-1 para tener derivada continua, y n-1 para tener derivada egunda continua. Es decir:un total de 4n-2 ecuaciones. Obsérvese que se tienen más incógnitas que ecuaciones, por lo que faltarían dos restricciones más,que corresponden a las derivadas en los extremos x_0 y x_n. Para resolver esto en lugar de expresar el polinomio cúbico de cada intervalo en la base canónica, vamos a considerar una base alternativa. Para esto introducimos la siguiente notación, y una nueva variable t en cada intervalo:

- $h_i = x_{i+1} - x_i$ (ancho del intervalo i)
- $t = \frac{x - x_i}{h_i}$ (nueva variable t del intervalo i)
- $\Delta_i = y_{i+1} - y_i$ (variación en y en el,intervalo i)
- $d_i = s'_i(x_i)$ (pendiente del polinomio s_i en el punto x_i)

Utilizando esto es posible calcular los cuatro coeficientes de cada spline (en la base canónica), ya que tenemos cuatro ecuaciones que los determinan:

$$\begin{cases} s_i(x_i) = y_i \\ s_i(x_{i+1}) = y_{i+1} \\ s_i'(x_i) = d_i \\ s_i'(x_{i+1}) = d_{i+1} \end{cases} \Leftrightarrow \begin{cases} \alpha_{i0} + \alpha_{i1}x_i + \alpha_{i2}x_i^2 + \alpha_{i3}x_i^3 & = y_i \\ \alpha_{i0} + \alpha_{i1}x_{i+1} + \alpha_{i2}x_{i+1}^2 + \alpha_{i3}x_{i+1}^3 = & y_{i+1} \\ 0 + \alpha_{i1} + 2\alpha_{i2}x_i + 3\alpha_{i3}x_i^2 & = d_i \\ 0 + \alpha_{i1} + 2\alpha_{i2}x_{i+1} + 3\alpha_{i3}x_{i+1}^2 & = \quad d_{i+1} \end{cases}$$

Para poder resolver el sistema con incógnitas di, es necesario especificar el valor de las pendientes d_0 y d_n. Para esto existen distintos criterios. Si imponemos $d_0 = d_n = 0$, se llama Spline natural. Si especficamos d_0 y d_n con cualquier otro par de valores, se llama Spline completos.

Por último tenemos la siguiente expresion para las variables d_i:

$$\frac{2}{h_i}d_i + \left(\frac{4}{h_i} + \frac{4}{h_{i+1}}\right)d_{i+1} + \frac{2}{h_{i+1}}d_{i+2} = \frac{6}{h_{i+1}^2}\Delta_{i+1} + \frac{6}{h_i^2}\Delta_i, \quad \forall i - 0, 1, \ldots, n-2$$

Mínimos cuadrados:

Mínimos cuadrados es una técnica de análisis numérico enmarcada dentro de la optimización matemática, en la que, dados un conjunto de pares ordenados —variable independiente, variable dependiente— y una familia de funciones, se intenta encontrar la función continua, dentro de dicha familia, que mejor se aproxime a los datos (un "mejor ajuste"), de acuerdo con el criterio de mínimo error cuadrático.

En su forma más simple, intenta minimizar la suma de cuadrados de las diferencias en las ordenadas (llamadas residuos) entre los puntos generados por la función elegida y los correspondientes valores en los datos.

Mínimos cuadrados no lineales:

Un problema de mínimos cuadrados no lineal esta dado por ciertos pares de datos como se observa en la siguiente tabla:

t	y
t_1	y_1
$\vdots$	$\vdots$
t_m	y_m

y una función de ajuste $\Phi(x_1, x_2, \ldots, x_n, t) = f(X, t)$ no lineal con respecto a los parámetros de ajuste $x_1, x_2, \ldots, x_n$.

Definimos el residuo, como la siguiente función:

$$R(x_1, x_2, \cdots, x_n) - R(X) = \begin{bmatrix} f(X, t_1) - y_1 \\ f(X, t_2) - y_2 \\ \vdots \\ f(X, t_m) - y_m \end{bmatrix} = F(X) - Y$$

donde denotamos

$$F(X) = \begin{bmatrix} f(X, t_1) \\ f(X, t_2) \\ \vdots \\ f(X, t_m) \end{bmatrix}$$

que depende de los parámetros $x_1, x_2, \cdots, x_n$.

El problema de mínimos cuadrados no lineal consiste en hallar un $\hat{X}$ que alcance el siguiente mínimo:
$\text{mín}_{X \in \mathbb{R}^n} \|F(X) - Y\|_2^2$

Algoritmo de Gauss-Newton:

Para resolver los problemas de minímos cuadrados no lineales contamos con el algoritmo de Gauss-Newton. La idea del algoritmo es suponer que tenermos una buena aproximación de la solucion X_k y linealizar el residuo en un entorno de X_k. Luego resolver un problema de mínimos cuadrados lineal para obtener una mejor aproximación X_{k+1}.

$$R(Z) \approx R(X_k) + J_R(X_k)(Z - X_k)$$

Buscamos minimizar $R(Z)$ y definimos el proximo punto X_{k+1} de la sucesión en el Z en que se da este mínimo. Por tanto, haciendo el cambio de variable $P_k = X_{k-1} - X_k$, en cada paso de la iteración se deberá hallar:

$$\min_{P_k \in R^n} \|R(X_k) + J_R(X_k) P_k\|_2^2$$

Observación: $J_R(X_k) = -J_F(X_k)$

Por tanto, esquematizamos el algoritmo de resolución de PMCNL:

$0 : X_0 \in \mathbb{R}^n$
$k + 1 : \Lambda_k \leftarrow J_k(X_k)$
$Y_k \leftarrow Y - F(X_k)$
Resuelvo PMCL:
$P_k \leftarrow$ solución de mín $P_{k \in R^n} \|Y_k - \Lambda_k P_k\|_2^2$
$X_{k+1} \leftarrow X_k + P_k$

3. Problema

3.1. Parte 1

En el problema planteado en la parte 1 se nos pide resolver una ecuacion diferencial con diferentes condiciones iniciales. Para este problema tenemos el archivo tiempos.mat, el cual nos da puntos que necesitamos para obtener los distintos intervalos de tiempos que son necesarios para aplicar el metodo .

Al plantear el Método de Heun para f tal cual está indicada en la letra, se observa que para las dos últimas condiciones iniciales el método diverge. Como solución para este problema, utilizamos el cambio de variable Bernoulli a f tal que $u = \frac{1}{y}$, de manera que se obtiene una nueva ecuación diferencial $u' = 3u + 3t - 1$, al igual que se plantea en la introducción. A continuación se plantean los algoritmos para cada uno de los casos:

Algoritmo 1: Método de Heun con $y_0 < 1$

$f(y,t) = y^2(1-3t) - 3y$
$T = [-1, \text{tiempos}]$ siendo tiempos el archivo dado llamado *tiempos.mat*
$Y =$ vector de 1x152 vacio
$Y_1 = y_0$
Para cada intervalo de tiempo
  $h = T_{i+1} - T_i$
  $p = Y_i + h * g(T_i, Y_i)$
  $Y_{i+1} = Y_i + (h/2)(f(T_i, Y_i)) + f(T_{i+1}, p))$

Devolver sucesion Y

Algoritmo 2: Método de Heun con $y_0 > 1$

$f*(y,t) = 3u + 3t - 1$
T = [−1, tiempos.mat]
Y = vector de 1x152 vacio
$Y_1 = \frac{1}{y_0}$
Para cada intervalo de tiempo
  $h = T_{i+1} - T_i$
  $p = Y_i + hf*(T_i, Y_i)$
  $Y_{i+1} = Y_i + (h/2)(f*(T_i, Y_i)) + f*(T_{i+1}, p))$
Para cada entrada de Y

$Y_i = \frac{1}{Y_i}$

Devolver sucesion Y

En el segundo caso, para $y_0 > 1$ después de aplicar el método de Heun a la función con el cambio de variable y obtener el resultado, se debe deshacer el cambio de variable. Para esto se aplica un loop en el que se hace $Y_i = \frac{1}{Y(i)}$ para cada entrada del vector Y. Para este método, vemos que las condiciones iniciales son muy importantes, ya que dependiendo de cuál sea la condición inicial el resultado va a cambiar. En nuestro caso, para las primeras dos condiciones iniciales($y_0 = 0{,}9142$ y $y_0 = 0{,}8655$), el método es efectivo (similar a la solucion brindada por el comando lsode) mientras que para las otras dos condiciones iniciales($y_0 = 1{,}0861$ y $y_0 = 1{,}0832$) el método no es tan efectivo. Esto ocurre ya que luego de calcular la solucion de la ecuacion diferencial, observamos que para las condiciones iniciales mayores a 1, el denominador se anula,esto hace que el método es menos efectivo que con condiciones iniciales menores a 1.

Las gráficas obtenidas para las cuatro condiciones iniciales a través del método de Heun son las siguientes:

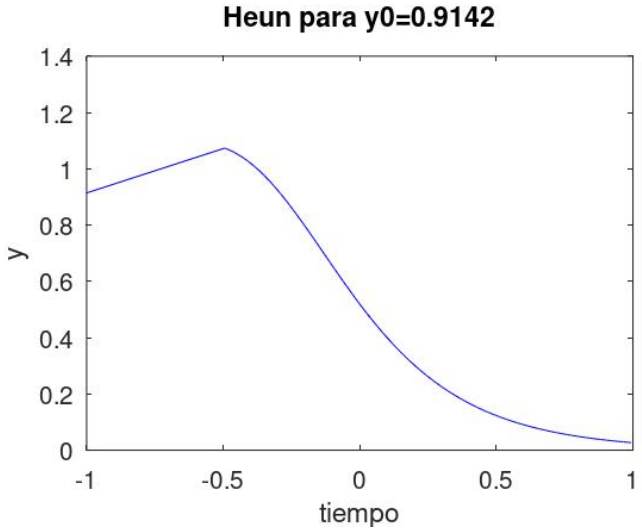

Figura 1: Gráfica para la primera condición inicial.

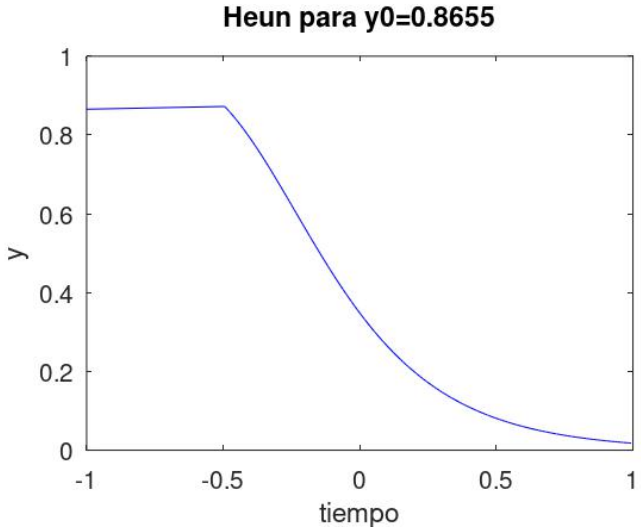

Figura 2: Gráfica para la segunda condición inicial.

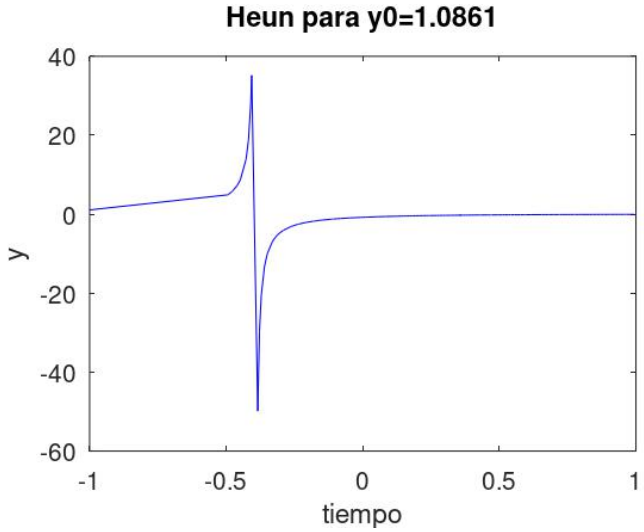

Figura 3: Gráfica para la tercera condición inicial.

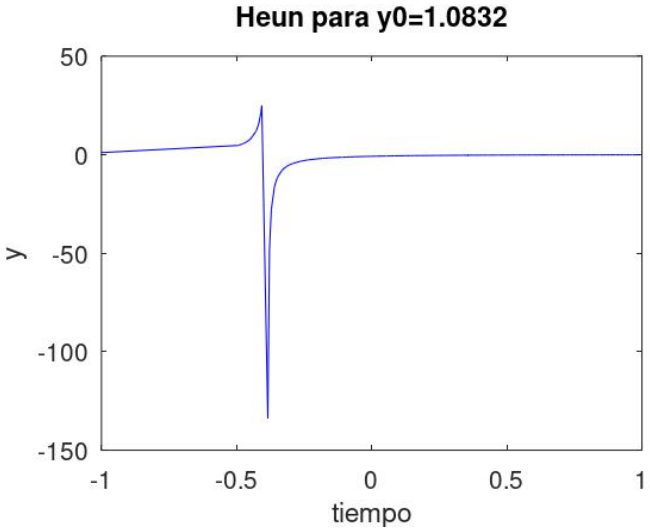

Figura 4: Gráfica para la cuarta condición inicial.

3.2. Parte 2

En la parte 2 se plantea resolver el mismo problema mediante la función *lsode* brindada por Octave.

Ésta función es de la forma: $[x, istate, msg] = lsode(fcn, x_0, t)$ La misma soluciona ecuaciones diferenciales ordinarias.

Los argumentos básicos que requiere lsode son:

fcn := Una cadena de caracteres o una función anónima que llame a la función f que determina la ecuación diferencial.

x_0 := El vector columna con los valores iniciales.

t := Un vector columna con los nodos en los que se quiere calcular la aproximación.

El comando devuelve la matriz x donde cada columna contiene los valores aproximados del vector solucion sobre cada uno de los nodos que se han indicado en el tercer argumento.

El comando lsode determina, de forma automática, tanto el método a emplear como el tamaño de paso empleado en función de ciertas tolerancias preestablecidas, de manera que el usuario obtiene un resultado sin necesidad de saber nada sobre el proceso de generado. No obstante, debemos indicar que existen situaciones en las que la simulación numérica puede no finalizar bien y, en ese caso, el comando nos puede avisar de dos maneras: bien emitiendo

mensajes de alerta, bien a través de un segundo argumento en la respuesta (istate) que, en caso de ser distinto de 2, indica una finalización incorrecta del proceso.

Algoritmo 3: lsode con $y_0 < 1$

$f(y,t) = y^2(1-3t) - 3y$
Y=lsode("f", y0, tiempo);
Devolver sucesion Y

Algoritmo 4: lsode con $y_0 > 1$

$f*(y,t) = 3u + 3t - 1$
aplico el cambio de variable a la condicion inicial
Y=lsode("f*", y0, tiempo);
Para cada entrada de Y
$Y_i = \frac{1}{Y_i}$

Devolver sucesion Y

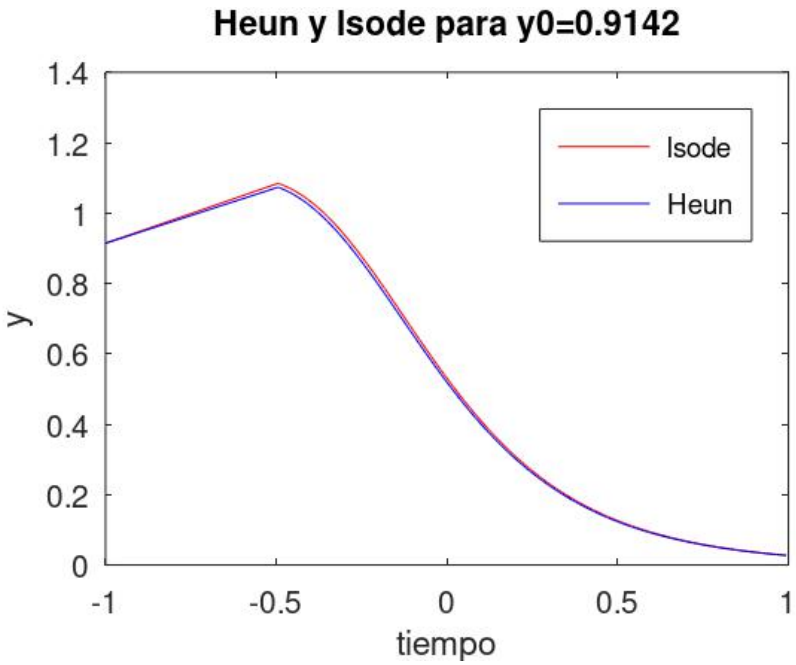

Figura 5: Gráficas de Lsode y Heun para la primera condición inicial.

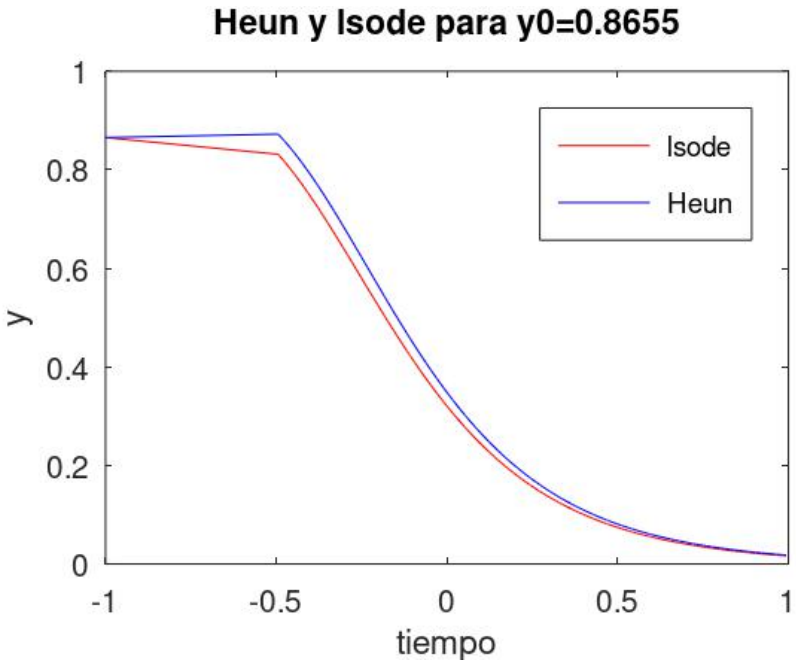

Figura 6: Gráficas de Lsode y Heun para la segunda condición inicial.

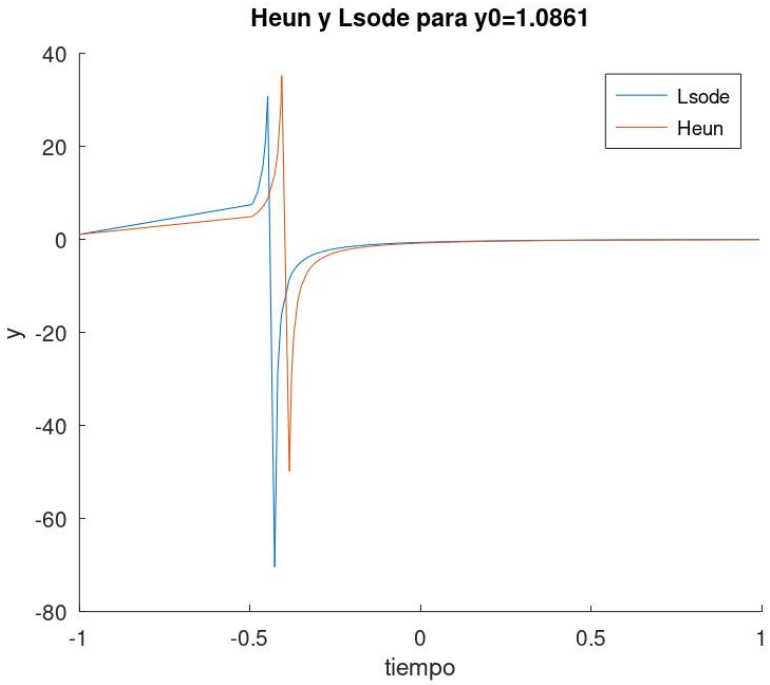

Figura 7: Gráficas de Lsode y Heun para la tercera condición inicial.

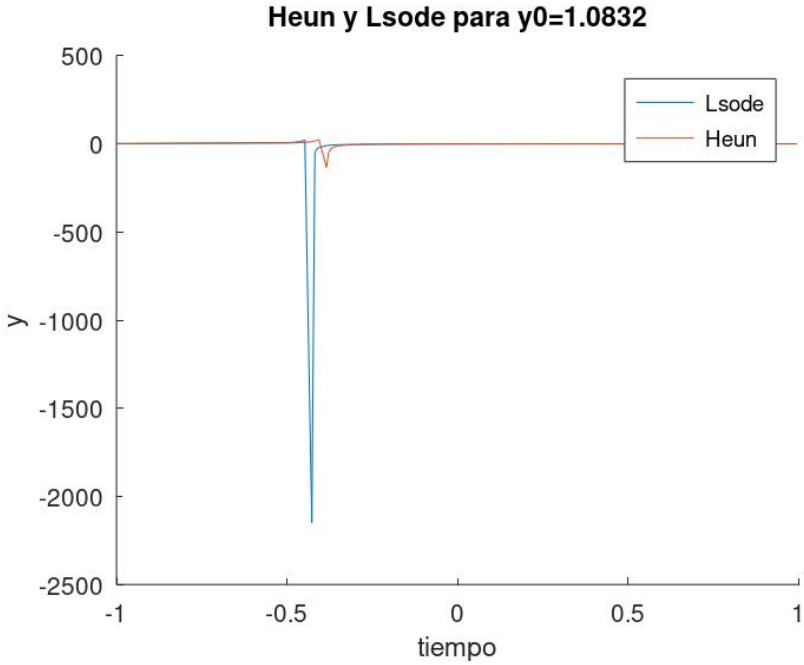

Figura 8: Gráficas de Lsode y Heun para la cuarta condición inicial.

3.3. Parte 4

Para interpolar mediante splines, interpolamos a trozos, usando un polinomio cúbico s_i en cada intervalo. En esta interpolación lo que buscamos es que al pegar cada polinomio de los intervalos , obtengamos una curva que tenga derivada segunda continua y sea dos veces derivable, por lo tanto :
* $s_{i-1}(x_{i-1}) = y_{i-1}\ 1 \leq i \leq n$
* $s_{i-1}(x_i) = y_i\ 1 \leq i \leq n$
* $s'_{i-1}(x_i) = s'_{i-1}(x_i)\ 1 \leq i \leq n-1$
* $s''_{i-1}(x_i) = s'_{i-1}(x_i)\ 1 \leq i \leq n-1$

Tenemos así, entre cada par de puntos $[x_i, x_{i+1}]$, un polinomio de la forma:
$s_i(x) = \alpha_{i0} + \alpha_{i1}x + \alpha_{i2}x^2 + \alpha_{i3}x^3$,teniendo así $4n$ incognitas.
Por otra parte, si contamos las restricciones dadas por cada item(*) tenemos un total de $4n-2$ ecuaciones.
Como tenemos dos incognitas mas que ecuaciones, nos faltan dos restricciones.
Para determinar los valores α_{ik} para todo $k \in \{1,2,3,4\}$ lo que hacemos es un sistema de ecuaciones lineales que tiene la siguiente forma:

$$(*)\begin{cases} s_i(x_i) = y_i \\ s_i(x_{i+1}) = y_{i+1} \\ s'_i(x_i) = d_i \\ s'_i(x_{i+1}) = d_{i+1} \end{cases}$$

siendo $d_i = s'_i(x_i)$,si imponemos $d_0 = 0$ y $d_n = 0$,obtenemos las dos ecuaciones que nos

faltan, llamamos a esta interpolación como Spline natural.
Entonces, para poder resolver este problema, primero debemos hallar cada d_i. Para hallarlos, lo que hacemos es una matriz que luego resolveremos con el algoritmo de Thomas, por ser tridiagonal. Esta matriz se obtiene de imponer condiciones sobre la ecuacion: $s_i(x_i + th_i) = y_i + t\Delta_i + t(t-1)(\Delta_i - h_i d_i) + t^2(t-1)(h_i(d_i + d_{i+1}) - 2\Delta_i)$.
Siendo $t = x - x_i/h_i$, $h_i = x_{i+1} - xi$, $\Delta_i = y_{i+1} - y_i$ y d_i el definido anteriormente. De esta forma, luego de imponer las condiciones la matriz queda así:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \frac{2}{h_0} & \left[\frac{4}{h_0} + \frac{4}{h_1}\right] & \frac{2}{h_1} & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{2}{h_1} & \left[\frac{4}{h_1} + \frac{4}{h_2}\right] & \frac{2}{h_2} & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{2}{h_{n-3}} & \left[\frac{4}{h_{n-3}} + \frac{4}{h_{n-2}}\right] & \frac{2}{h_{n-2}} & 0 \\ 0 & 0 & 0 & \dots & 0 & \frac{2}{h_{n-2}} & \left[\frac{4}{h_{n-2}} + \frac{4}{h_{n-1}}\right] & \frac{2}{h_{n-1}} \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \\ d_n \end{bmatrix} =$$

$$\begin{bmatrix} d_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{n-2} \\ z_{n-1} \\ d_n \end{bmatrix}$$

Algoritmo 5: parte 4

Para hacer la matriz mencionada anteriormente, y luego aplicar el metodo de Thomas, para poder hallar los d_i:

para cada $i \in \{1, ..., 150\}$
$h_i = t_{i+1} - t_i$ (siendo ti los intervalos de tiempo dados)

para cada $i \in \{1, ..., 150\}$
$\Delta_i = y_{i+1} - y_i$

definimos el vector z como
$z_1 = 0$
$z_{152} = 0$

para cada intervalo de tiempo i
$z_{i+1} = \frac{6\Delta_{i+1}}{h_{i+1}^2} + \frac{6\Delta_i}{h_i^2}$

definimos el vector a como
$a_{151} = 0$
$a_{152} = 0$

para cada $i \in \{1, ..., 150\}$
$a_i = \frac{2}{h_i}$

definimos el vector b como
$b_1 = 1$
$b_{152} = 1$
para cada $i \in \{2, ..., 150\}$
$b_i = \frac{4}{h(i-1)} + \frac{4}{h(i)}$

definimos el vector c como
$c_1 = 0$
$c_{152} = 0$
para cada $i \in \{2, ..., 150\}$
$c_i = \frac{2}{h(i)}$

Aplicamos metodo de Thomas para los anteriores vectores:

El primer paso del metodo es modificar los coeficientes como sigue:

$$c'_i = \begin{cases} \frac{c_i}{b_i} & ; i = 1 \\ \frac{c_i}{b_i - c'_{i-1} a_i} & ; i = 2, 3, \ldots, n-1 \end{cases}$$

donde se marcan con superindice ' los nuevos coeficientes.
De igual manera se opera:

$$d'_i = \begin{cases} \frac{z_i}{b_i} & ; \quad i = 1 \\ \frac{z_i - z'_{j-1} a_i}{b_i - c'_{i-1} a_i} & ; \quad i = 2, 3, \ldots, n. \end{cases}$$

a lo que se llama barrido hacia adelante. A continuacion se obtiene la solucion por sustitucion hacia atras:

$x_n = z'_n$
$x_i = z'_i - c'_i x_{i+1} \quad ; i = n-1, n-2, \ldots, 1.$

Creamos la matriz A para resolver el sistema de ecuaciones $(*)$ visto anteriormente:

$A[1,j] = [1, t_2(i), t_2^2(i), t_2^3(i)]$
$A[2,j] = [1, t_2(i+1), t_2^2(i+1), t_2^3(i+1)]$
$A[3,j] = [0, 1, 2t_2(i), 3t_2^2(i)]$
$A[4,j] = [0, 1, 2t_2(i+1), 3t_2^2(i+1)]$

siendo $t_2(i)$ cada intervalo de tiempo i del archivo *tiempos2.mat* y j cada columna de la matriz.

Luego creamos el vector b de valores independientes:

$b[1] = y_1$
$b[2] = y_{i+1}$
$b[3] = z_i$
$b[4] = z_{i+1}$

Para cada intervalo de tiempo i
graficamos el polinomio $p_i = A[i,1] + A[i,2]x + A[i,3]x^2 + A[i,4]x^3$

A continuación se presentan las gráficas obtenidas mediante la implementación anterior, comparadas con la solución dada por *lsode*.
Para las primeras tres graficas, vemos que la interpolación por Splines es exactamente igual a la que nos da el comando Lsode de Octave salvo para el intervalo inicial, ya que en ese intervalo no tenemos la cantidad de puntos necesaria para que la interpolación se comporte de mejor manera.
Para la ultima, si bien el intervalo inicial se comporta mejor que las otras 3 interpolaciones, luego, el polinomio interpolante tiene puntos que difieren de la solucion de Lsode entre el intervalo de tiempo [-0.5,-0.3] aproximadamente(en los picos de la funcion de Lsode).

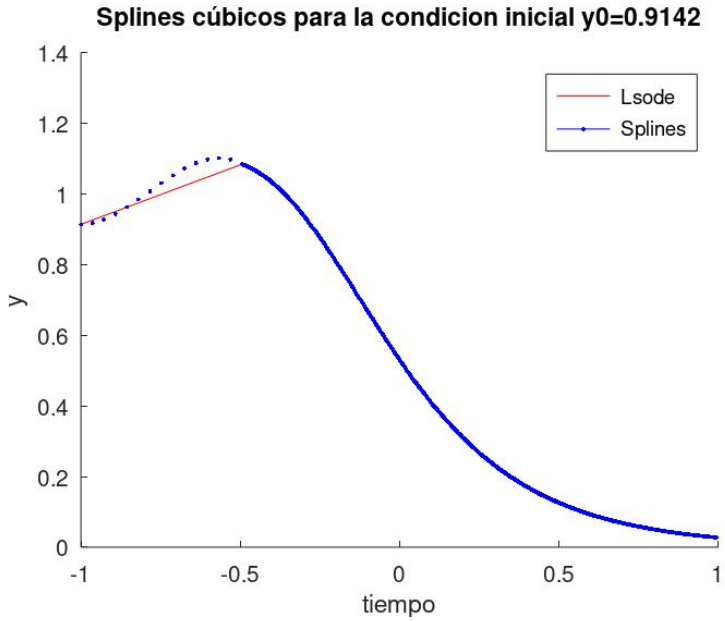

Figura 9: Gráficas Lsode y Splines cubicos para la primera condición inicial.

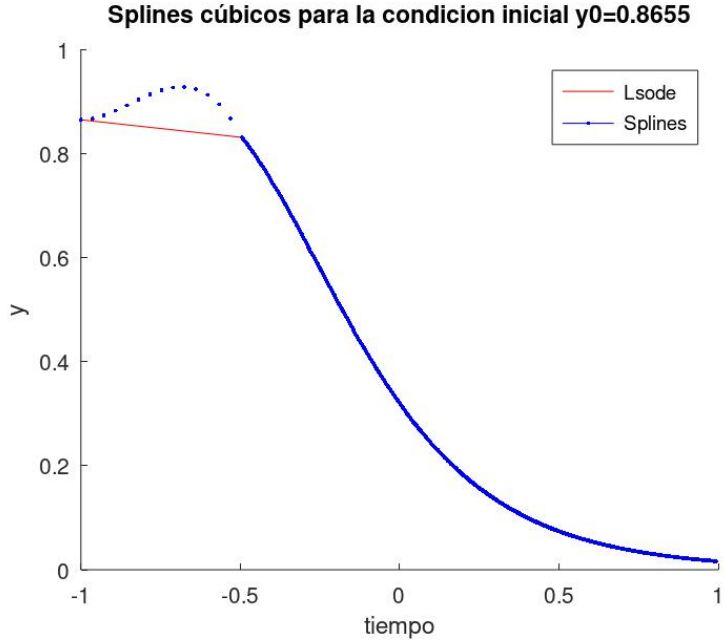

Figura 10: Gráficas Lsode y Splines cubicos para la segunda condición inicial.

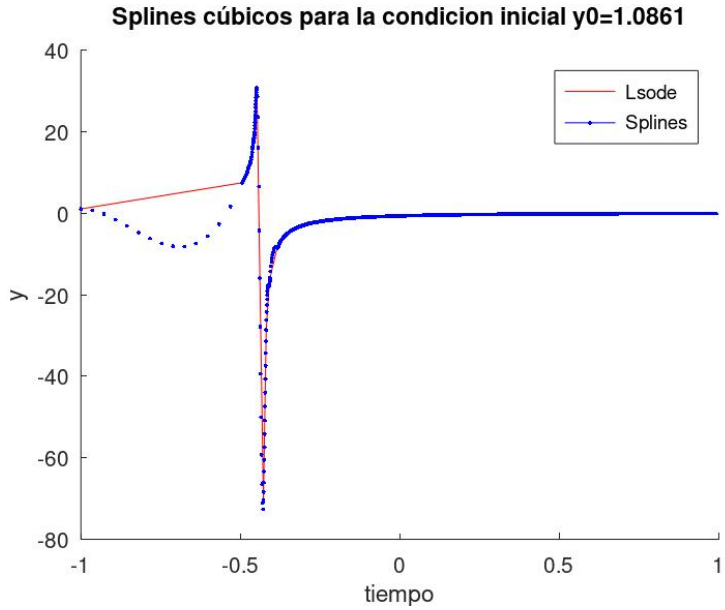

Figura 11: Gráficas Lsode y Splines cubicos para la tercera condición inicial.

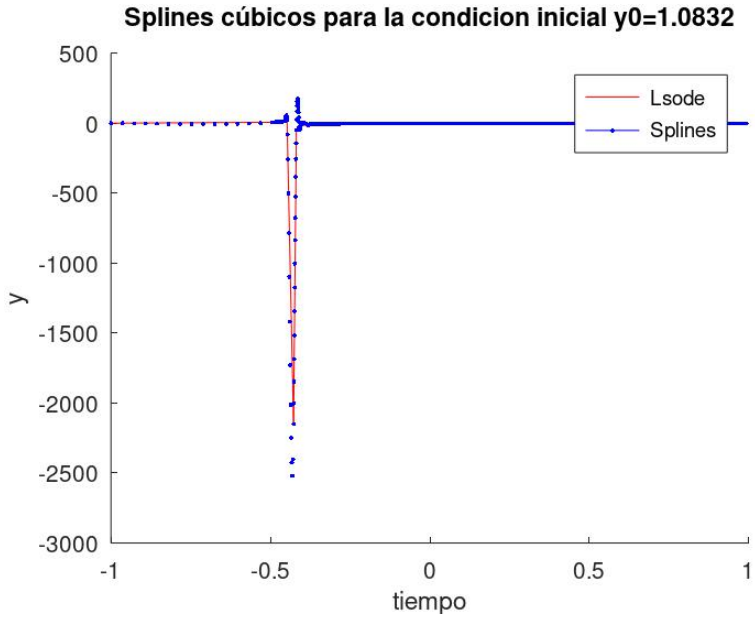

Figura 12: Gráficas Lsode y Splines cubicos para la cuarta condición inicial.

3.4. Parte 5

Para el problema de mínimos cuadrados no lineal de la parte 5, en la cual nos sugieren la función $f(t) = \alpha g_1^{\beta}(t) + g_2^{\gamma}(t)$ y un conjunto de puntos $A = \{(t'_1, y(t'_1)), ..., (t'_n, y(t'_n))\}$ donde

$t'(i)$ son los intervalos de *tiempos2.mat* e $y(t)$ es la solución del PVI con $y(-1) = 0{,}9526$, tomamos como g_1 y g_2 a las funciones interpoladas mediante Splines que minimizan el error respecto a la solución dada por el comando *Lsode* de Octave. Por lo tanto, g_1 y g_2 son las funciones interpoladas de las primeras dos condiciones iniciales ($y_0 = 0{,}9142$ y $y_0 = 0{,}8655$) respectivamente.
De esta manera, para resolver el Problema de Mínimos Cuadrados No Lineal utilizamos el algoritmo de Gauss-Newton, así podemos pasar de un PMCNL a un PMCL y resolver con las ecuaciones normales.
Algoritmo 6: parte 5

Para obtener el polinomio de cada algoritmo:
    para los valores obtenidos de α, β y γ
    genero los puntos de la funcion sugerida para cada tiempo:
    for i $\in$ [1, max(T)] siendo T los intervalos de tiempo
      $aux(i) = \alpha g_1^{\beta}(i) + g_2^{\gamma}(i)$
    devolver aux

Para resolver Minimos Cuadrados Lineal:
    $Y_2 \leftarrow A^t Y$
    $A_2 \leftarrow A^t A$
    genero una matriz M que es la matriz ampliada de A_2 con Y_2
    aplico pivoteo parcial
    devuelvo el resultado

Para Minimos Cuadrados No Lineal:
    resuelvo el problema de minimos cuadrados no lineales a traves del metodo Gauss–Newton:
    $A_k \leftarrow J_F(X_k)$
    $Y_k \leftarrow Y - F(X_k)$
    Resuelvo PMCL resolviendo las ecuaciones normales y obtengo P_k como solucion
    $X_{k+1} \leftarrow X_k + P_k$

Para aplicar el método utilizamos los valores iniciales (0.5, 0, 0.5) y obtuvimos como resultado $\alpha = 0{,}57456$, $\beta = 0,25139$ y $\gamma = 0{,}47587$, que se corresponden con la siguiente gráfica:

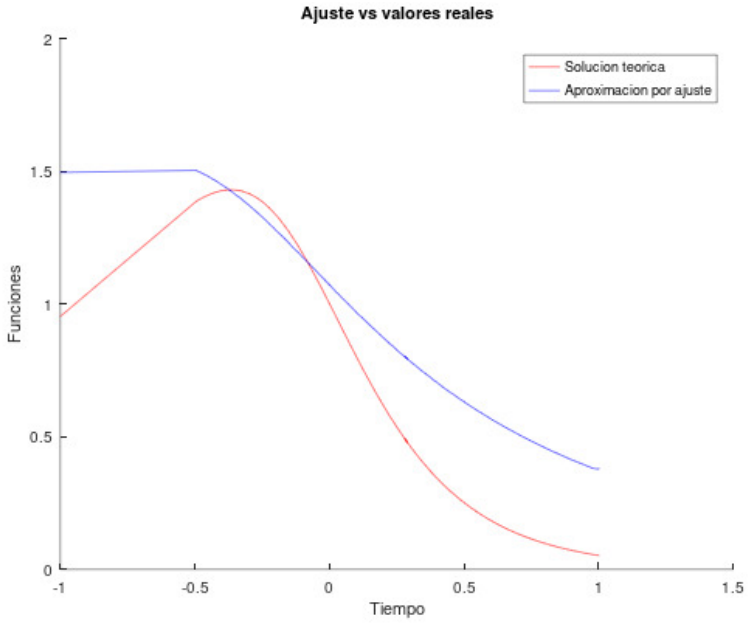

Figura 13: Gráficas de ajuste por mínimos cuadrados y solución analítica.

3.5. Parte 6

Para el modelo ajustado, los valores obtenidos para $f(0{,}20)$, $f(0{,}30)$, $f(0{,}35)$ y $f(0{,}75)$ son:

- $f(0{,}20) = 0{,}87231$
- $f(0{,}30) = 0{,}78384$
- $f(0{,}35) = 0{,}74196$
- $f(0{,}75) = 0{,}48314$

3.6. Parte 7

Para la parte 7 debemos comparar los resultados obtenidos mediante $f(t)$ con $y(t)$. Si comparamos con los valores de la parte 6, podemos ver que la funcion $f(t)$ no tiene valores muy similares, aunque en varios puntos se aproxima a $y(t)$. Observamos que en el comienzo de la grafica, más especificamente entre los tiempos -1 y -0.5, las curvas estan alejadas y tienen un comportamiento distinto, ya que la roja es creciente y la azul parece constante, esto puede deberse a que en el vector de tiempos otorgado no tenemos muchos puntos en este intervalo, dando como resultado una recta en lugar de la curva esperada.

Referencias

[1] Instituto Matemática y Estadística "*RafaelLaguardia*", *Apuntes de teórico-Métodos numéricos.*

[2] CDF Online- https://www.cfd-online.com/Wiki/Tridiagonal_matrix_algorithm_-_TDMA_(Thomas_algorithm)

[3] Antonia M. Delgado, Juanjó Nieto, Aureliano M. Robles, Óscar Sánchez, *Métodos numéricos básicos con Octave.*