Laboratorio 5 MA-4301:

Diferenciación, Integración
y
Ecuaciones Diferenciales Ordinarias No Lineales

Gonzalo Hernández

UChile - Departamento de Ingeniería Matemática

El objetivo de este laboratorio es aprender a utilizar las funciones y comandos que están disponibles en Matlab para calcular y resolver en forma numérica:

1) Derivadas e integrales

2) Ecuaciones diferenciales no lineales

3) Ecuaciones diferenciales no lineales de orden superior y sistemas de ecuaciones diferenciales no-lineales

1 Diferenciación Numérica

Sean $(n+1)$ puntos $(x_0, y_0), ..., (x_n, y_n)$, tales que: $x_0 < x_1 < \cdots < x_n$. El polinomio de interpolación de Lagrange:

$$p_L(x) = \sum_{j=0}^{n} a_j x^j \tag{1}$$

es el polinomio de menor grado que interpola estos puntos, es decir, que verifica:

$$y_k = p_L(x_k) = \sum_{j=0}^{n} a_j x_k^j \quad \forall k = 0, 1, ..., n \tag{2}$$

Para calcular $p_L(x)$ basta resolver el sistema de Van der Monde:

$$\begin{bmatrix} x_0^n & x_0^{n-1} & \cdots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n^n & x_n^{n-1} & \cdots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{3}$$

O bien, determinar $p_L(x)$ por su fórmula analítica:

$$\begin{aligned} p_L(x) &= \sum_{k=0}^{n} y_k L_{n,k}(x) \\ L_{n,k}(x) &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \quad \forall k = 0, , 1..., n \end{aligned} \tag{4}$$

En el caso que los puntos (x_k, y_k) sean parte de una función $f(x)$ de la cual no se conoce su forma analítica, es decir:

$$y_k = f(x_k) \quad \forall k = 0, 1, ..., n \tag{5}$$

el método de interpolación entrega una forma de aproximar esta función.

En la actividad 3 del laboratorio 1, se solicitaba determinar el polinomio de interpolación de la función $f(x) = e^{\sin(x^3)}$ en $[-2, 2]$ para una malla de puntos equi-espaciados a distancia $h = 0.2$. El resultado de esta cálculo se muestra en la figura 1:

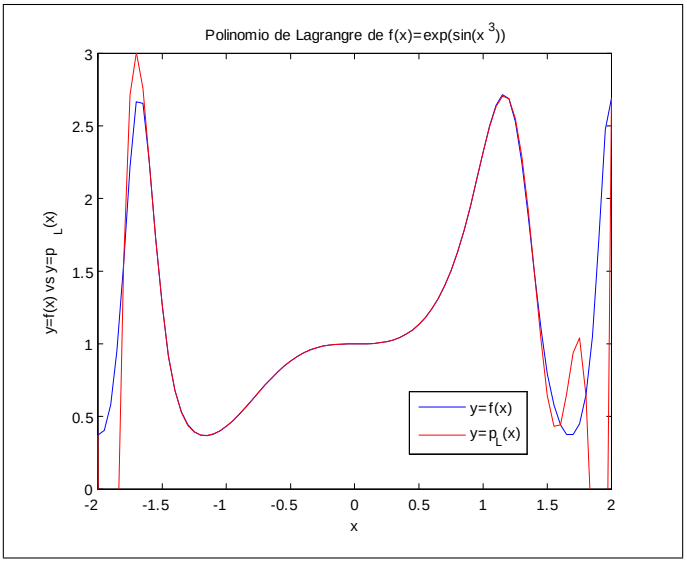

Figura 1. Polinomio de interpolación de la función $f(x) = e^{\sin(x^3)}$ en $[-2, 2]$

Las instrucciones que permiten calcular este polinomio de interpolación son:

```
>> x=[-2:0.2:2]; y=exp(sin(x.^3));
>> n=length(x)-1;
>> p=polyfit(x,y,n);
>> xf=[-2:0.05:2];
>> yf=exp(sin(xf.^3));
>> yp=polyval(p,xf);
>> plot(xf,yf);
>> hold on;
>> plot(xf,yp,'-r');
```

En el laboratorio 3 se enseñara a utilizar en detalle los comandos de Matlab que permiten interpolar y aproximar una función.

Para aproximar la derivada numérica de una función se deriva el polinomio de interpolación de Lagrange de grado n que interpola a la función en $(n+1)$ puntos convenientemente escogidos. La fórmula de los 3 puntos aproxima la primera derivada de una función f en x_0:

$$\frac{df}{dx}(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} - \frac{h^2}{6} f^{(3)}(\eta) \quad \text{para algún } \eta \in [x_0 - h, x_0 + h] \tag{6}$$

Dado que el punto η sólo se puede estimar aproximadamente, no se le considera. Luego:

$$\frac{df}{dx}(x_0) \approx \frac{f(x_0+h)-f(x_0-h)}{2h} \tag{7}$$

En efecto, sean $x_0, ..., x_n \in [a,b]$ $(n+1)$ puntos distintos ordenados de menor a mayor, $f \in \mathcal{C}^{n+1}([a,b])$, luego el polinomio de Lagrange de grado n verifica:

$$\begin{aligned} f(x) &= \sum_{k=0}^{n} f(x_k)L_{n,k}(x) + \frac{f^{n+1}(\xi(x))}{(n+1)!}\prod_{k=0}^{n}(x-x_k) \quad \xi(x)\in[x_0,x_n] \\ L_{n,k}(x) &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \quad \forall k=0,,1...,n \end{aligned} \tag{8}$$

Luego:

$$\frac{df(x)}{dx} = \sum_{k=0}^{n} f(x_k)\frac{dL_{n,k}(x)}{dx} + \frac{f^{n+1}(\xi(x))}{(n+1)!}\frac{d}{dx}\left(\prod_{k=0}^{n}(x-x_k)\right) + \frac{1}{(n+1)!}\prod_{k=0}^{n}(x-x_k)\frac{df^{n+1}(\xi(x))}{dx} \tag{9}$$

Si se evalúa esta fórmula en $x = x_j$:

$$\frac{df(x_j)}{dx} = \sum_{k=0}^{n} f(x_k)\frac{dL_{n,k}(x_j)}{dx} + \frac{f^{n+1}(\xi(x_j))}{(n+1)!}\frac{d}{dx}\left(\prod_{k=0}^{n}(x-x_k)\right)\Bigg|_{x_j} \tag{10}$$

donde:

$$\begin{aligned} \frac{d}{dx}\left(\prod_{k=0}^{n}(x-x_k)\right) &= (x-x_1)(x-x_2)\cdots(x-x_n)+(x-x_0)(x-x_2)\cdots(x-x_n)+\cdots \\ &\quad \cdots+(x-x_0)(x-x_1)\cdots(x-x_{n-1}) \\ \frac{d}{dx}\left(\prod_{k=0}^{n}(x-x_k)\right)\Bigg|_{x_j} &= (x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n) \end{aligned} \tag{11}$$

Luego:

$$\frac{df(x_j)}{dx} = \sum_{k=0}^{n} f(x_k)\frac{dL_{n,k}(x_j)}{dx} + \frac{f^{n+1}(\xi(x_j))}{(n+1)!}\prod_{k=0,k\neq j}^{n}(x_j-x_k) \tag{12}$$

Considerando los puntos $x_0, x_1 = x_0+h, x_2 = x_0+2h$:

$$\frac{df(x_j)}{dx} \approx \sum_{k=0}^{2} f(x_k)\frac{dL_{2,k}(x_j)}{dx} \tag{13}$$

donde:

$$\begin{aligned} \frac{dL_{2,0}(x)}{dx} &= \frac{d}{dx}\left(\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}\right) = \frac{2x-x_1-x_2}{(x_0-x_1)(x_0-x_2)} \\ \frac{dL_{2,1}(x)}{dx} &= \frac{d}{dx}\left(\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}\right) = \frac{2x-x_0-x_2}{(x_1-x_0)(x_1-x_2)} \\ \frac{dL_{2,2}(x)}{dx} &= \frac{d}{dx}\left(\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}\right) = \frac{2x-x_0-x_1}{(x_2-x_0)(x_2-x_1)} \end{aligned} \tag{14}$$

Calculando $\frac{df(x_1)}{dx}$:

$$\begin{aligned} L'_{2,0}(x_1) &= -\frac{1}{2h} \\ L'_{2,1}(x_1) &= 0 \\ L'_{2,2}(x_1) &= \frac{1}{2h} \end{aligned} \tag{15}$$

Luego:

$$\frac{df(x_1)}{dx} \approx \frac{f(x_2) - f(x_0)}{2h} \tag{16}$$

Para obtener la fórnula (7) basta redefinir los puntos:

$$x_0 = x_0 - h, x_1 = x_0, x_2 = x_0 + h \tag{17}$$

Siguiendo con este razonamiento, la fórmula de los 5 puntos:

$$x_1 = x_0 - 2h, x_2 = x_0 - h, x_3 = x_0, x_4 = x_0 + h, x_5 = x_0 + 2h \tag{18}$$

define una aproximación mucho más exacta de la primera derivada de una función f en x_0:

$$\frac{df}{dx}(x_0) = \frac{f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)}{12h} + \frac{h^4}{30} f^{(5)}(\eta) \quad \text{para algún } \eta \in [x_0 - 2h, x_0 + 2h] \tag{19}$$

Procediendo en forma análoga, la segunda derivada de una función se puede aproximar por:

$$\frac{d^2 f}{dx^2}(x_0) = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} - \frac{h^2}{12} f^{(4)}(\eta) \quad \text{para algún } \eta \in [x_0 - h, x_0 + h] \tag{20}$$

$$\tag{21}$$

$$\frac{d^2 f}{dx^2}(x_0) = \frac{-f(x_0 + 2h) + 16f(x_0 + h) - 30f(x_0) + 16f(x_0 - h) - f(x_0 - 2h)}{12h^2} + O(h^4)$$

Las principal aplicación del cálculo numérico de la derivada está en optimización y ecuaciones diferenciales ordinarias o parciales.

La convección o transporte de una cantidad o propiedad física $u(x,t)$, a lo largo del eje $\mathbb{R}$, a velocidad constante c está representada por la siguiente ecuación diferencial parcial (edp):

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} + c\frac{\partial u(x,t)}{\partial x} &= f(x,t) \quad \forall x \in \mathbb{R}, \forall t > 0 \\ u(x,0) &= u_0(x) \quad \forall x \in \mathbb{R} \end{aligned} \tag{22}$$

donde la producción de $u(x,t)$ está modelada por $f(x,t)$. Un ejemplo de esta edp es el transporte de un contaminante en la atmósfera. En este caso c es la velocidad del viento. Para calcular la solución numérica de (22) se discretizan la variables (x,t) en un rectángulo finito: $[a,b] \times [0,T]$:

$$\begin{aligned} x_j &= a + j\delta_x \quad \delta_x = \frac{(b-a)}{n} \quad j = 0, 1, ..., n \\ t_k &= k\delta_t \quad \delta_t = \frac{T}{m} \quad k = 0, 1, ..., m \end{aligned} \tag{23}$$

$$u_{j,k} = u(x_j, t_k) \quad j = 0, 1, ..., n, k = 0, 1, ..., m$$

$$f_{j,k} = f(x_j, t_k) \quad j = 0, 1, ..., n, k = 0, 1, ..., m \tag{24}$$

$$u_{j,0} = u_0(x_j) \quad j = 0, 1, ..., n \tag{25}$$

Aplicando las fórmulas de los tres puntos (7), o bien una aproximación simple de $\frac{\partial u(x,t)}{\partial t}$, $\frac{\partial u(x,t)}{\partial x}$:

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} &= \frac{u_{j,k+1} - u_{j,k-1}}{2\delta_t} \\ \frac{\partial u(x,t)}{\partial x} &= \frac{u_{j+1,k} - u_{j,k-1}}{2\delta_x} \end{aligned} \tag{26}$$

Luego:

$$u_{j,k+1} = u_{j,k-1} - \left(c\frac{\delta_t}{\delta_x}\right)(u_{j+1,k} - u_{j-1,k}) + \delta_t f_{j,k} \tag{27}$$

Este método se denomina diferencias finitas. El error de la discretización de $\frac{\partial u(x,t)}{\partial t}$, $\frac{\partial u(x,t)}{\partial x}$ es orden $O(\delta_t^2)$ y $O(\delta_x^2)$, respectivamente.

Para calcular $u_{j,k}$ se aplica el siguiente método:

Paso 1: Calcular:

$$u_{j,1} = u_{j,0} - \left(c\frac{\delta_t}{\delta_x}\right)(u_{j+1,0} - u_{j,0}) + \delta_t f_{j,0} \quad \forall j = 0, 1, ..., n \tag{28}$$

Paso 2: Para $k = 1, ..., m$:

$$u_{0,k+1} = u_{0,k} - \left(c\frac{\delta_t}{\delta_x}\right)(u_{1,k} - u_{0,k}) + \delta_t f_{0,k} \tag{29}$$

Para $j = 1, ..., n$:

$$u_{j,k+1} = u_{j,k-1} - \left(c\frac{\delta_t}{\delta_x}\right)(u_{j+1,k} - u_{j-1,k}) + \delta_t f_{j,k} \tag{30}$$

2 Integración Numérica

En esta sección se enseñara a calcular integrales de funciones de una y varias variables. Para ejemplificar los cálculos, trabajaremos con la siguiente función:

$$g(x) = \frac{xe^{-x^2}}{1+x^2} - 5x\cos(x^3) \tag{31}$$

Su gráfico esta dado por:

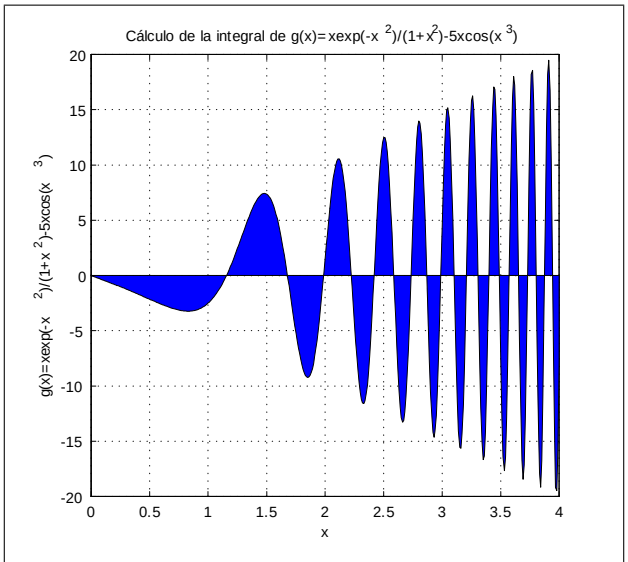

Figura 2. Gráfico función $g(x) = \frac{xe^{-x^2}}{1+x^2} - 5x\cos(x^3)$

El comando de Matlab que calcula integrales de funciones a variable real es:

$$[q, fcnt] = quad('funcion', a, b, tol) \tag{32}$$

Este comando implementa el algoritmo de Simpson Recursivo Adaptivo. Las salidas de *quad* son:

1°) q: Valor de la integral:

$$q = \int_a^b f(x)dx \tag{33}$$

2°) $fcnt$: Número de evaluaciones de funciones realizadas por el método.

Los argumentos de *quad* son:

1) $'funcion'$ es la función en una variable que se desea integrar. Hay 3 formas de ingresar una función a *quad*. Tomemos como ejemplo la función:

$$f(x) = \cos(x^2) \tag{34}$$

   (a) Como string:

$$[q, fcnt] = quad('\cos(x.\hat{}2)', 0, 1) \tag{35}$$

   (b) Como función anónima:

$$\begin{aligned} f &= @(x)(\cos(x.\hat{}2)); \\ [q, fcnt] &= quad(f, 0, 1) \end{aligned} \tag{36}$$

   (c) Como función .m:

```
function y=f(x)
y=cos(x.^2);
end
```

   En este caso, el comando *quad* se utiliza de la siguiente forma:

$$[q, fcnt] = quad(@f, 0, 1) \tag{37}$$

   La función .m se pasa por referencia: $@f$, y no por valor.

2) Los parámetros a, b definen el intervalo de integración $[a, b]$. Por ejemplo, si $[a, b] = [0, \pi]$ y la función se pasa por referencia:

$$[q, fcnt] = quad(@f, 0, pi) \tag{38}$$

3) El parámetro tol define el máximo error tolerado en valor absoluto. A valores más grandes en tol se hacen menos evaluaciones de la función f (y por lo tanto es cálculo es más rápido), pero la precisión de la integral es menor. El valor por default es $tol = 1.0 \times 10^{-6}$.

Veamos algunos ejemplos.

```
>> format('long')
>> f=@(x)(cos(x.^2));
>> [q,fcnt]=quad(f,0,1)
q =
    0.904524260466284
fcnt =
    17

>> [q,fcnt]=quad(f,0,1,1e-12)
q =
   0.904524237900278
fcnt =
   265

>> [q,fcnt]=quad(f,0,4,1e-8)
q =
   0.594460328699523
fcnt =
   305

>> [q,fcnt]=quad(f,0,4,1e-10)
q =
   0.594460327623617
fcnt =
   745

>> [q,fcnt]=quad(f,0,4,1e-12)
q =
   0.594460327497798
fcnt =
        1981
```

Los mensajes de warning del comando *quad* son:

Mensaje	Descripción
'Minimum step size reached'	Indica que la subdivisión recursiva de la integral produjo un subintervalo de largo menor que el error de redondeo. Si esto ocurre, posiblemente se está integrando una función que tiene alguna singularidad en $[a, b]$.
'Maximum function count exceeded'	Indica que se ha excedido el numéro máximo de evaluaciones de la función: 10000. Si esto ocurre, posiblemente se está integrando una función que tiene alguna singularidad en $[a, b]$.
'Infinite or Not-a-Number function value encountered'	Indica un overflow de operaciones de punto flotante o división por cero durante el calculo de la integral en $[a, b]$

En Matlab también se pueden integrar funciones de 2 o 3 variables. En 2 dimensiones, si se quiere calcular:

$$q = \iint_R f(x,y)dxdy \tag{39}$$

Podemos ejecutar el comando *dblquad* :

$$q = dblquad('funcion', x_{\min}, x_{\max}, y_{\min}, y_{\max}, tol) \tag{40}$$

Como podemos observar, sus argumentos de entrada y salida son similares a los del comando *quad*. Los argumentos de *dblquad* son:

1) $'funcion'$ es la función que se desea integrar. Hay 2 formas de ingresar una función a *dblquad*. Tomemos como ejemplo la función:

$$f(x,y) = xy\sin(x^2+y^2) \tag{41}$$

cuyo gráfico está dado por:

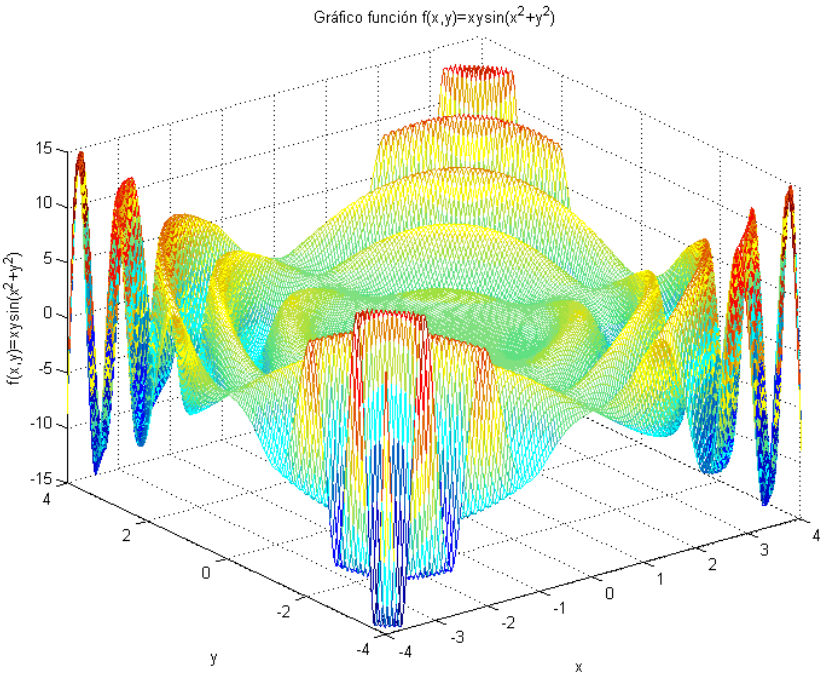

Figura 3. Grafico de la función $f(x,y) = xy\sin(x^2+y^2)$.

(a) Como función anónima:

$$f = @(x,y)(x.*y.*\sin(x.\hat{}2 + y.\hat{}2)); \tag{42}$$

Si la región de integración es $R = [0,1]\times[0,1]$:

$$q = dblquad(f, 0-4, 4, -4, 4) \tag{43}$$

(b) Como función .m:

```
function z = f(x,y)
z = x.*y.*sin(x.^2+y.^2);
```

end

Si $R = [-1, 1] \times [0, 3]$:

$$q = dblquad(@f, -1, 1, 0, 3) \tag{44}$$

La función .m de 2 variables se pasa por referencia: $@f$, y no por valor.

2) Los parámetros $R = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ definen la región rectangular de integración. Por ejemplo si: $R = [-\pi/2, \pi/2] \times [1, \sqrt{2}]$:

$$q = dblquad(@f, -pi/2, pi/2, 1, sqrt(2)) \tag{45}$$

3) El parámetro *tol* define el máximo error tolerado en valor absoluto. El valor por default es $tol = 1.0 \times 10^{-6}$.

ACT 1: Diferenciación e Integración Numérica:

1) Aplicando el método de las diferencias finitas resuelva la edp lineal de convección (22) definiendo a elección $c, f(x,t), u_0(x)$ y los intervalos $[a, b] \times [0, T]$. Grafique la solución obtenida y la superficie de error, comparando con la solución exacta.

2) Calcule $q = \iint_R h(x,y)dxdy$ para 3 funciones $h(x, y)$ y tres regiones de integración R de la siguiente forma:

   (a) Defina $h : \mathbb{R}^2 \to \mathbb{R}$ y R a su gusto. Programe h como función .m y ejecute el comando *dblquad* con 3 diferentes valores para *tol*. Anote en el informe como utilizó el comando y el detalle del resultado obtenido: Valor de la integral, tiempo de ejecución, error.

   (b) En Matlab es posible también integrar funciones en regiones R no rectangulares. Por ejemplo, para calcular:

$$q = \iint_R e^{-x^2-y^2} dxdy \tag{46}$$

   en la región $R = \left\{(x, y) \in \mathbb{R}^2 / 0 \leq x^2 + y^2 \leq 1\right\}$, es decir en el círculo unitario, se integra la función $e^{-x^2-y^2}$ en el rectángulo $[-1, 1] \times [-1, 1]$ multiplicada por la indicatriz de la región R:

$$I_R(x, y) = \begin{cases} 1 & \text{si } x^2 + y^2 \leq 1 \\ 0 & \text{en otro caso} \end{cases} \tag{47}$$

   Esta función se implementa mediante la condición lógica: $I_R(x, y) = \left(x^2 + y^2 <= 1\right)$. Finalmente:

$$q = \iint_R e^{-x^2-y^2} dxdy = \int_{-1}^{1}\int_{-1}^{1} e^{-x^2-y^2} I_R(x, y)dxdy \tag{48}$$

   La integral anterior se calcula mediante el comando:

$$q \approx dblquad(@(x, y)(\exp(-x.\hat{}2 - y.\hat{}2). * (x.\hat{}2 + y.\hat{}2 <= 1), -1, 1, -1, 1) \tag{49}$$

   Construya entonces 3 ejemplos para una de las funciones h del ítem anterior. Anote en el informe como utilizó el comando y el detalle del resultado obtenido: Valor de la integral, tiempo de ejecución, error.

La integración de funciones sobre $\mathbb{R}^3$ se realiza de manera similar mediante el comando *triplequad*.

3 Ecuaciones diferenciales de primer, segundo orden y orden superior

Desde el siglo 18, las leyes físicas del universo han sido descritas, principalmente, por ecuaciones diferenciales ordinarias o parciales. Estas ecuaciones tienen derivadas de funciones que se desconocen y que representan la fuerza, presión, temperatura, potencial electromagnético, etc., ver ref. [6]. Quizás la ecuación diferencial más conocida es la segunda ley de Newton:

$$f(t) = m\frac{d^2x}{dt^2} \tag{50}$$

donde $x(t)$ es la posición de un cuerpo de masa m, constante, que se mueve horizontalmente debido a la acción de una fuerza $f(t)$. Esta ley que fue publicada en 1687 en la obra "Philosophiae Naturalis Principia Mathematica". En Ciencia e Ingeniería, las ecuaciones diferenciales representan, es decir modelan, el comportamiento de un sistema real. Se obtienen como resultado de un proceso iterativo de abstracción, análisis, hipótesis, síntesis, verificación y validación. Se pueden clasificar de acuerdo a: Tipo (ordinaria o parcial); Orden (definido por la mayor derivada presente en la ecuación); Linealidad.
En este laboratorio trabajaremos con ecuaciones diferenciales ordinarias, de primer y segundo orden, lineales y no lineales. Las ecuaciones diferenciales lineales de orden 1 y 2 tienen la forma:

$$a_1(t)\frac{dy}{dt} + a_0(t)y(t) = g(t) \qquad \forall t \in [t_0, T] \tag{51}$$

$$a_2(t)\frac{d^2y}{dt^2} + a_1(t)\frac{dy}{dt} + a_0(t)y(t) = g(t) \qquad \forall t \in [t_0, T] \tag{52}$$

donde:
$y : I \supseteq [t_0, T] \to \mathbb{R}$ es una función de clase $\mathcal{C}^1(I)$ o $\mathcal{C}^2(I)$.
$a_2, a_1, a_0, g : I \supseteq [t_0, T] \to \mathbb{R}$ son funciones de clase $\zeta^0(I)$.
Por ejemplo, el péndulo simple de la figura 1, define una ecuación diferencial de segundo orden:

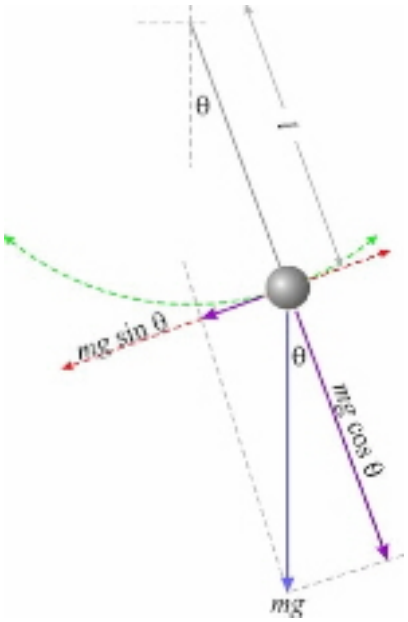

Figura 4. Péndulo simple oscilando en un plano, ver Wikipedia.

La ecuación de la dinámica del péndulo se obtiene aplicando la segunda ley de Newton para el arco $s = l\theta$:

$$m\frac{d^2s}{dt} = -mg\sin(\theta) \Rightarrow \frac{d^2\theta}{dt} = -\frac{g}{l}\sin(\theta) \tag{53}$$

Esta ecuación diferencial es no lineal, en general de dificil resolución analítica. Para encontrar una solución se aproxima linealmente la parte no-lineal o se resuelve numéricamente. Para aproximar linealmente la ecuación (53) calculamos el desarrollo de Taylor en torno a $\theta_0 = 0$ de la parte no lineal:

$$\sin(\theta) = \theta - \tfrac{1}{6}\theta^3 + \tfrac{1}{120}\theta^5 + O(\theta^7) \tag{54}$$

El último término del desarrollo es de la forma $O(\theta^7) \leq c\theta^7$. Si $\theta \approx 0$, basta quedarse con el primero o los dos primeros términos:

$$\sin(\theta) \approx \theta \qquad \sin(\theta) \approx \theta - \tfrac{1}{6}\theta^3 \tag{55}$$

El error de esta aproximación es: $E(\theta) \leq c\theta^5$, lo que es despreciable si $\theta \approx 0$. Luego:

$$\frac{d^2\theta}{dt} \approx -\frac{g}{l}\theta \Rightarrow \theta(t) \approx c_1 \sin\left(\sqrt{\tfrac{g}{l}}t\right) + c_2\cos\left(\sqrt{\tfrac{g}{l}}t\right) \tag{56}$$

Volvamos a la ecuación lineal general de primer y segundo orden. Para resolver analíticamente y numéricamente la ecuación (52) se la reduce a un sistema de ecuaciones de primer orden, mediante el cambio de variables:

$$y_1 = y \qquad y_2 = \frac{dy}{dt} \tag{57}$$

Si $a_2(t) \neq 0\ \forall t \in [t_0, T]$, la ecuación (52) queda:

$$\begin{aligned} \frac{dy_1}{dt} &= y_2 \\ \frac{dy_2}{dt} &= -\frac{a_0(t)}{a_2(t)}y_1 - \frac{a_1(t)}{a_2(t)}y_2 + \frac{g(t)}{a_2(t)} \end{aligned} \tag{58}$$

Si se define:

$$\alpha_1(t) = -\frac{a_0(t)}{a_2(t)} \qquad \alpha_2(t) = -\frac{a_1(t)}{a_2(t)} \qquad \beta(t) = \frac{g(t)}{a_2(t)} \tag{59}$$

El sistema anterior queda:

$$\frac{dy_1}{dt} = y_2 \tag{60}$$

$$\frac{dy_2}{dt} = \alpha_1(t)y_1 + \alpha_2(t)y_2 + \beta(t) \tag{61}$$

Y en forma matricial:

$$\begin{bmatrix} \frac{dy_1}{dt} \\ \frac{dy_2}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \alpha_1(t) & \alpha_2(t) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \beta(t) \end{bmatrix} \tag{62}$$

La existencia y unicidad de la ecuación (51) se demuestra a su vez llevándola a la forma del problema de Cauchy. Si $a_1(t) \neq 0\ \forall t \in [t_0, T]$, la ecuación (51) es equivalente a:

$$\frac{dy}{dt} + p(t)y(t) = q(t) \qquad y(t_0) = y_0 \quad \forall t \in [t_0, T] \tag{63}$$

donde:

$$p(t) = \frac{a_0(t)}{a_1(t)} \qquad q(t) = \frac{g(t)}{a_1(t)} \tag{64}$$

De la ecuación (63) se obtiene directamente el problema de Cauchy:

$$\frac{dy}{dt} = f(t,y) \quad y(t_0) = y_0 \quad \forall t \in [t_0, T]\,, \tag{65}$$

donde $y_0 \in \mathbb{R}, f : \mathbb{R}^2 \to \mathbb{R}$ es una función diferenciable definida por:

$$f(t,y) = q(t) - p(t)y(t) \tag{66}$$

La existencia y unicidad de la solución de la ecuación (65) está asegurada por la condición de Lipschitz de f en la variable y, ver refs. [3, 4]:

$$|f(t,y_1) - f(t,y_2)| < L\,|y_1 - y_2| \qquad \forall t \in [t_0, T] \;\; \forall y_1, y_2 \in \mathbb{R} \tag{67}$$

donde L es una contante positiva a determinar. En el caso de la ecuación (65) se tiene que:

$$|f(t,y_1) - f(t,y_2)| \leq |p(t)y_1 - p(t)y_2| \leq \max_{t\in[t_0,T]} |p(t)|\,|y_1 - y_2| \qquad \forall t \in [t_0, T] \;\; \forall y_1, y_2 \in \mathbb{R} \tag{68}$$

Basta tomar $L = 1 + \max\limits_{t\in[t_0,T]} |p(t)|$. Esta constante es finita pues $p(t) \in \zeta^0(I)$. Concluimos entonces que el problema (65) tiene solución única $\forall\,(t_0, y_0) \in [t_0, T] \times \mathbb{R}$. Para encontrar esta solución, trabajamos con diferenciales. La ecuación (65) es equivalente a:

$$\begin{aligned} (p(t)y - q(t))\,dt + dy &= 0 \quad \forall t \in [t_0, T] \\ y(t_0) &= y_0 \end{aligned} \tag{69}$$

este diferencial no es exacto. El factor integrante $\mu(t)$ debe ser talque el diferencial:

$$\mu(t)\,(p(t)y - q(t))\,dt + \mu(t)dy = 0 \tag{70}$$

es exacto, luego:

$$\begin{aligned} \frac{\partial}{\partial y}\left(\mu(t)\,(p(t)y - q(t))\right) &= \frac{\partial}{\partial t}\left(\mu(t)\right) \\ \mu(t)p(t) &= \frac{d\mu(t)}{dt} \Rightarrow \mu(t) = e^{\int p(t)dt} \end{aligned} \tag{71}$$

Se definimos:

$$\begin{aligned} M(t,y) &= e^{\int p(t)dt}\,(p(t)y - q(t)) \\ N(t,y) &= e^{\int p(t)dt} \end{aligned} \tag{72}$$

Entonces, existe $F(t,y)$ talque:

$$\frac{\partial F(t,y)}{\partial t} = M(t,y) \qquad \frac{\partial F(t,y)}{\partial y} = N(t,y) \tag{73}$$

y la solución de la ecuación (69) y por lo tanto de la ecuación (63) está dada por:

$$F(t,y) = c = y(t)e^{\int p(t)dt} - \int e^{\int p(t)dt} q(t)dt \tag{74}$$

$$y(t) = ce^{-\int p(t)dt} + \int e^{\int p(t)dt} q(t)dt$$

4 Solución numérica de ecuaciones diferenciales no lineales

En esta sección aprenderemos a resolver ecuaciones diferenciales no-lineales con los comandos que están disponibles en Matlab. Primero, resolveremos una ecuación diferencial no lineal de primer orden. Utilizaremos como ejemplo la ecuación de crecimiento logístico:

$$\begin{aligned} \frac{dp}{dt} &= \alpha p(t) - \beta p(t)^2 \quad \forall t \in [t_0, T] \\ p(0) &= p_0 \end{aligned} \tag{75}$$

que modela el crecimiento de la población de Estados Unidos durante el siglo 20. Los datos reales del problema son:

Año	Tiempo t_k	$p(t)$ real [millones de personas]
1900	0.0	76.2122
1910	10.0	92.2285
1920	20.0	106.0215
1930	30.0	123.2026
1940	40.0	132.1646
1950	50.0	151.3258
1960	60.0	179.3232
1970	70.0	203.3020
1980	80.0	226.5422
1990	90.0	248.7099
2010	100.0	281.4219

(76)

Aplicando un proceso de ajuste de parámetros, los modeladores determinaron que: $\alpha = 0.02, \beta = 0.00004$, y de acuerdo a esto, la solución exacta de (75) está dada por:

$$p(t) = \frac{500}{\left(1 + \frac{4239}{761} e^{-\frac{1}{50}t}\right)} \tag{77}$$

La ecuación (75) es una ecuación diferencial de Bernoulli de orden $n = 2$, que se define para cualquier real $n \neq 0, 1$ según:

$$\frac{dy}{dt} + p(t)y = q(t)y^n \qquad t_0 \leq t \leq T \quad y(t = t_0) = y_0 \tag{78}$$

Esta ecuación diferencial es no lineal pero integrable. Multiplicando (78) por $(1-n)y^{-n}$ se obtiene:

$$\begin{aligned} (1-n)y^{-n}\frac{dy}{dt} + (1-n)p(t)y^{1-n} &= q(t)(1-n) \\ \frac{d}{dt}\left(y^{1-n}\right) + (1-n)p(t)y^{1-n} &= q(t)(1-n) \end{aligned} \tag{79}$$

Si se define $u(t) = y(t)^{1-n}$ se obtiene la ecuación diferencial de primer orden lineal en la variable $u(t)$:

$$\frac{du}{dt} + (1-n)p(t)u = q(t)(1-n) \tag{80}$$

ecuación que se puede resolver aplicando la fórmula (74). La demostracion de la igualdad (77) se deja como ejercicio propuesto.

Los métodos que resuelven numéricamente una ecuación diferencial de valor inicial se definen sobre el problema de Cauchy de primer orden:

$$\frac{dy}{dt} = f(t, y) \quad \forall t \in [t_0, T] \qquad y(t_0) = y_0 \tag{81}$$

donde $f : \mathbb{R}^2 \to \mathbb{R}$ es una función diferenciable, $y_0 \in \mathbb{R}$. Este problema define a su vez una ecuación diferencial no-lineal de primer orden no homogénea. La existencia y unicidad de la solución está asegurada por la condición de Lipschitz de f en la variable y, ver refs. [3, 4].

La familia de métodos de Runge-Kutta de orden $2, 3, 4$ son los más utilizados para resolver la ecuación (81). Estos métodos determinan el valor de la solución $y(t)$ evaluada en puntos equi-espaciados definidos según:

$$\begin{aligned} h &= \frac{(T - t_0)}{n} \\ t_k &= t_0 + hk \quad \forall k = 0, ..., n \\ y_k &= y(t_k) \end{aligned} \tag{82}$$

utilizando el valor inicial $y(t_0) = y_0$. Una vez calculados estos puntos y_k se puede interpolar o aproximar polinomialmente la solución $y(t)$. Por lo tanto, mientras más es la cantidad de puntos determinada, mejor será la aproximación de la solución. La demostración de las iteraciones se basa en mejorar el clásico método de Euler, ver ref. [1].

El método de Runge-Kutta de orden 2 está definido por:

$$\begin{aligned} q_k^1 &= hf\left(t_k, y_k\right) \\ q_k^2 &= hf\left(t_k + \frac{h}{2}, y_k + \frac{q_k^1}{2}\right) \\ y_{k+1} &= y_k + q_k^2 \qquad \forall k = 0, 1, \ldots, n \end{aligned} \tag{83}$$

Su error es de orden $O(h^3)$. El método de Runge-Kutta de orden 3 está definido por:

$$\begin{aligned} q_k^1 &= hf\left(t_k, y_k\right) \\ q_k^2 &= hf\left(t_k + \frac{h}{2}, y_k + \frac{q_k^1}{2}\right) \\ q_k^3 &= hf\left(t_k + h, y_k - q_k^1 + 2q_k^2\right) \\ y_{k+1} &= y_k + \frac{1}{6}\left(q_k^1 + 4q_k^2 + q_k^3\right) \quad \forall k = 0, 1, \ldots, n \end{aligned} \tag{84}$$

Su error es de orden $O(h^4)$. El método de Runge-Kutta de orden 4 está definido por:

$$\begin{aligned} q_k^1 &= hf(t_k, y_k) \\ q_k^2 &= hf\left(t_k + \frac{h}{2}, y_k + \frac{q_k^1}{2}\right) \\ q_k^3 &= hf\left(t_k + \frac{h}{2}, y_k + \frac{q_k^2}{2}\right) \\ q_k^4 &= hf\left(t_k + h, y_k + q_k^3\right) \\ y_{k+1} &= y_k + \frac{1}{6}\left(q_k^1 + 2q_k^2 + 2q_k^3 + q_k^4\right) \qquad \forall k = 0, 1, \ldots, n \end{aligned} \tag{85}$$

Su error es de orden $O(h^5)$. Claramente, a medida que aumenta el orden del método aumenta su precisión, pero tambien aumenta la cantidad de cálculo a realizar.

Por ejemplo, para resolver la edo no lineal:

$$\begin{aligned} \frac{dy}{dt} &= \frac{y}{t} - \left(\frac{y}{t}\right)^2 \qquad \text{para } t \in [1, 2] \\ y(1) &= 1 \end{aligned} \tag{86}$$

mediante el método de Runge-Kutta de orden 2 para el valor del incremento $h = 0.25$ y comparar los resultados numéricos con la solución analítica:

$$y(t) = \frac{t}{(1+\ln(t))} \tag{87}$$

Se construye primero la iteración:

$$\begin{aligned} f(t,y) &= \frac{y}{t} - \left(\frac{y}{t}\right)^2 \\ h &= \frac{(T-t_0)}{n} = 0.25 \Rightarrow n = 4 \\ t_k &= t_0 + hk = 1 + 0.25k \qquad \forall k = 0,1,\ldots,4 \\ q_k^1 &= hf\left(t_k, y_k\right) = 0.25\left(\frac{y_k}{t_k} - \left(\frac{y_k}{t_k}\right)^2\right) \\ q_k^2 &= hf\left(t_k + \frac{h}{2}, y_k + \frac{q_k^1}{2}\right) = 0.25\left(\frac{y_k + \frac{q_k^1}{2}}{t_k + \frac{h}{2}} - \left(\frac{y_k + \frac{q_k^1}{2}}{t_k + \frac{h}{2}}\right)^2\right) \\ y_{k+1} &= y_k + q_k^2 \qquad \forall k = 0,1,\ldots,4 \end{aligned} \tag{88}$$

Y luego se desarrollan los cálculos, mostrándolos en una tabla como la siguiente:

k	t_k	$y(t_k)$ exacto	q_k^1	q_k^2	y_k	E_{abs}	E_{rel}
0	1.0	1.0	0	0.0247	1.0	0	0
1	1.25	1.0219	0.0370	0.0458	1.0247	0.0028	0.0027
2	1.5	1.0672	0.0511	0.0549	1.0705	0.0033	0.0031
3	1.75	1.1221	0.0574	0.0592	1.1254	0.0033	0.0029
4	2.0	1.1812	0	0	1.1846	0.0034	0.0030

(89)

Para mejorar la precisión bastar disminuir el valor de h, aumentando por lo tanto los puntos de la malla, o subir el orden del método de Runge - Kutta. Se recomienda aplicar el método de Runge-Kutta de orden 4, por su precisión y estabilidad numérica.

Finalmente, en general se acostumbra trabajar con mallas equi-espaciadas, pero también es posible definir una malla no equi-espaciada de la siguiente forma:

$$t_k = t_{k-1} + h_k \quad \forall k = 0,...,n \qquad y_k = y(t_k)$$

Esto es de gran utilidad en funciones que son irregulares en una parte del intervalo donde se está resolviendo la edo.

Existe en Matlab una familia de comandos que permite resolver ecuaciones y sistemas de ecuaciones diferenciales. Los de mejor comportamiento numérico son:

$ode23$	$ode15s$
$ode45$	$ode23s$

(90)

Los comandos $ode45$ y $ode23$ implementan los metodos de Runge-Kutta adaptivos de orden 4 y 2, siendo el método de orden 4 el de mayor precisión. Los comandos $ode15s$ y $ode23s$ implementan métodos para edo rígidas (stiff) o no suaves de difícil solución numérica en la práctica debido a singularidades o fuertes

variaciones en un intervalo pequeño, ver refs. [2, 4]. Un ejemplo de una edo no lineal rigida de segundo orden es la ecuación del oscilador de Van der Pol que modela un circuito electrónico:

$$\frac{d^2y}{dt^2} + y(t) + \mu(y^2(t) - 1)\frac{dy}{dt} = 0 \tag{91}$$

donde $y(t)$ es la corriente eléctrica, que modela el circuito eléctrico con un diodo de tunel de la figura 2 :

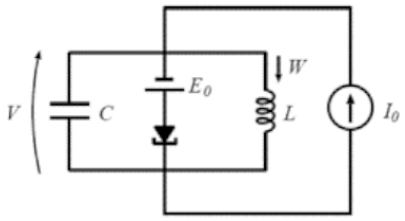

Figura 5. Circuito de Van der Pol

La forma de utilizar estos comandos es:

$$[t, y] = dsolver('funcion', [t0\ T], y0, options) \tag{92}$$

donde *dsolver* es alguno de los comandos $ode45, ode23, ode15s, ode23s$. Las salidas de *dsolver* son:

1) t: vector de instantes de tiempo, donde $t_0 \leq t_k \leq T\ \forall k = 1, ..., n$. Matlab determina este vector adaptivamente, dependiendo de la complejidad de la función.

2) y: Solución de la ecuación diferencial evaluada en los instantes de tiempo t

Los argumentos de *dsolver* son:

1) $'funcion'$ es la función que define la edo, i.e. es igual a $f(t, y)$. Debe ser ingresada como función .m. y ser pasada por referencia.

2) $[t0\ T]$ define el intervalo donde se revolverá la edo.

3) $y0$ es el valor inicial

4) *options* son las opciones del comando *dsolver*. Utilizaremos las opciones por default.

Por ejemplo, para resolver la edo no lineal:

$$\begin{aligned} \frac{dy}{dt} &= \frac{\sin(y^2 + t)}{1 + t^2} \quad \forall t \in [0, 10] \\ y(t_0) &= 0 \end{aligned} \tag{93}$$

Primero se debe programar $f(t, y) = \dfrac{\sin(y^2 + t)}{1 + t^2}$ como función .m:

```
function dy= f(t,y)
dy = sin(y.^2+t)./(1+t.^2);
end
```

la que debe aceptar vectores como argumentos t, y. Luego se ejecuta algún comando de *dsolver* pasando la función f por referencia. Se recomienda utilizar *ode*45 por su precisión y rapidez de cálculo. Si *ode*45 demora en entregar la solución, se debe ocupar *ode*15*s*:

$$[t, y] = ode45(@f, [0\ 10], 0);$$

Podemos apreciar la solución graficando:

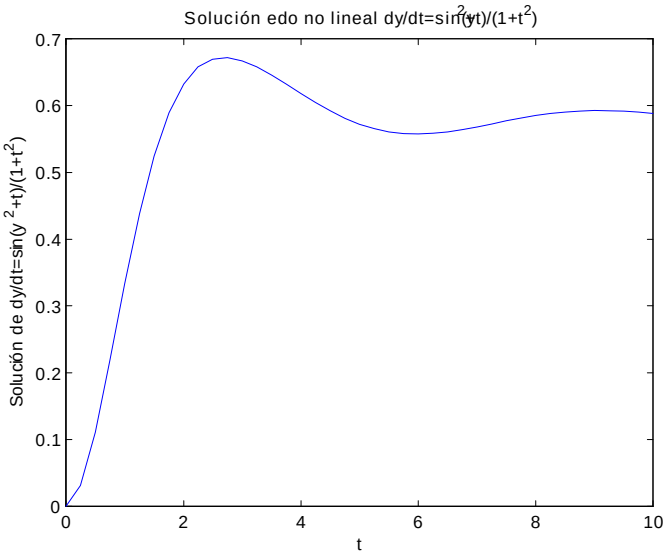

Figura 6. Solución numérica edo no lineal: $\dfrac{dy}{dt} = \dfrac{\sin\left(y^2 + t\right)}{(t^2 + 1)}$

Volvamos ahora a nuestro ejemplo inicial.

ACT 2: Modelo de Crecimiento Logístico:

1) Determine la solución numérica de la edo de crecimiento logístico utilizando los comandos *ode*23 y *ode*45 :

$$\begin{aligned} \frac{dp}{dt} &= 0.02p(t) - 0.00004p(t)^2 \\ p(0) &= 76.1 \end{aligned} \tag{94}$$

Compare el tiempo de ejecución y grafique las soluciones entregada por *ode*23 y *ode*45. Explore algunas opciones del comando.

Calcule el error relativo del método utilizando los datos reales (76). Llamaremos este error E_{RK23} y E_{RK45}. Incluya en el informe como utilizó los comandos y los gráficos de las soluciones y errores. Explique cual método es más exacto.

Evalúe el comportamiento del error relativo fuera del intervalo $[0, 90]$. Para esto considere los datos

reales de población del siglo 19:

Año	Tiempo t_k	$p(t)$ real [millones de personas]
1890	-10.0	62.9798
1880	-20.0	50.1892
1870	-30.0	38.5584
1860	-40.0	31.4433
1850	-50.0	23.1919
1840	-60.0	17.0634
1830	-70.0	12.8607
1820	-80.0	9.6384
1810	-90.0	7.2399
1800	-100.0	5.3085

(95)

2) Determine el error relativo del modelo comparando los datos reales del siglo 19 y 20 con la solución exacta de la ecuación:

$$p(t) = \frac{500}{\left(1 + \frac{4239}{761} e^{-\frac{1}{50}t}\right)} \tag{96}$$

Llamaremos este error E_M. Anote en el informe como calculó E_M y el detalle del resultado obtenido.

3) Analice el error del método y del modelo. De acuerdo a sus resultados:

(a) Puede concluir que la solución entregada por el método de Runge - Kutta es precisa ?

(b) Puede concluir que el modelo de crecimiento logístico representa bien los datos reales de crecimiento de población ?

(c) Proponga dos formas de disminuir el error del modelo.

Justifique claramente sus respuestas en base a los cálculos realizados.

5 Solución de sistemas de ecuaciones diferenciales no lineales

Un sistema de ecuaciones diferenciales no lineales con condiciones iniciales, está definido por el sistema:

$$\begin{aligned}
\frac{dy_1}{dt} &= f_1(t, y_1, y_2, ..., y_n) \qquad y_1(t_0) = y_{1,0} \\
\frac{dy_2}{dt} &= f_2(t, y_1, y_2, ..., y_n) \qquad y_2(t_0) = y_{2,0} \\
&\vdots \\
\frac{dy_n}{dt} &= f_n(t, y_1, y_2, ..., y_n) \qquad y_n(t_0) = y_{n,0}
\end{aligned} \tag{97}$$

donde $t \in [t_0, T]$ y $f_1, ..., f_n$ son funciones no lineales con argumentos $t, y_1, ..., y_n$. Para asegurar la existencia y unicidad de la solución de (97) existe una condición similar a la del problema de Cauchy (81):

Teorema 1 *Supongamos que las funciones f_i son continuas y satisfacen la condición de Lipschitz en el dominio $D = \{(t, z_1, ..., z_n) / t_0 \leq t \leq T, -\infty < z_j < \infty \ \forall j = 1, ..., n\} : \forall (t, u_1, ..., u_n), (t, v_1, ..., v_n) \in D$*

$$|f_i(t, u_1, ..., u_n) - f_i(t, v_1, ..., v_n)| \leq L_i \sum_{j=1}^{n} |u_j - v_j| \tag{98}$$

para $L_i > 0$, entonces el sistema de ecuaciones (97) tiene solución única.

La condición de Lipschitz para f_i se satisface si $f_i \in \zeta^1((t_0, T))$ y

$$\left| \frac{\partial f_i\left(t, u_1, ..., u_n\right)}{\partial u_j} \right| \leq L_i \ \forall \left(t, u_1, ..., u_n\right) \in D \tag{99}$$

La demostración de este teorema se puede encontrar en [4].

Para calcular la solución numérica de (97) se utilizan las versiones para sistemas de ecuaciones de la familia de métodos de Runge-Kutta de orden $2, 3, 4$. Estos métodos determinan el valor de la solución:

$$y(t) = (y_1(t), y_2(t), ..., y_n(t)) \tag{100}$$

evaluada en $n \times m$ puntos equi-espaciados definidos según:

$$\begin{aligned} h &= \frac{(T - t_0)}{m} \Rightarrow t_k = t_0 + hk \quad \forall k = 1, ..., m \\ y_{1,1} &= y_1(t_1), y_{1,2} = y_1(t_2), ..., y_{1,m} = y_1(t_m) = y_1(T) \\ y_{2,1} &= y_2(t_1), y_{2,2} = y_2(t_2), ..., y_{2,m} = y_2(t_m) = y_2(T) \\ &\vdots \\ y_{n,1} &= y_n(t_1), y_{n,2} = y_n(t_2), ..., y_{n,m} = y_n(t_m) = y_n(T) \end{aligned} \tag{101}$$

utilizando los valores iniciales $y_1(t_0) = y_{1,0}, y_2(t_0) = y_{2,0}, ..., y_n(t_0) = y_{n,0}$. Una vez calculados los puntos $y_{i,k} \ \forall i = 1, ..., n, \forall k = 1, ..., m$ se puede interpolar o aproximar polinomialmente la solución $y(t) = (y_1(t), y_2(t), ..., y_n(t))$. Por lo tanto, mientras mayor es la cantidad de puntos determinada, mejor será la aproximación de la solución.

El método de Runge-Kutta de orden 2 para el sistema (97) está definido por, ver refs. [1, 4]: Para $k = 1, ..., m$, dados los puntos $t_k, y_{1,k}, y_{2,k}, ..., y_{n,k}$ se calculan los puntos $y_{1,(k+1)}, y_{2,(k+1)}, ..., y_{n,(k+1)}$ según:

$$\begin{aligned} q^1_{1,k} &= h f_1\left(t_k, y_{1,k}, y_{2,k}, ..., y_{n,k}\right) \\ &\vdots \\ q^1_{n,k} &= h f_n\left(t_k, y_{1,k}, y_{2,k}, ..., y_{n,k}\right) \\ q^2_{1,k} &= h f_1\left(t_k + \frac{h}{2}, y_{1,k} + \frac{q^1_{1,k}}{2}, y_{2,k} + \frac{q^1_{2,k}}{2}, ..., y_{n,k} + \frac{q^1_{n,k}}{2}\right) \\ &\vdots \\ q^2_{n,k} &= h f_n\left(t_k + \frac{h}{2}, y_{1,k} + \frac{q^1_{1,k}}{2}, y_{2,k} + \frac{q^1_{2,k}}{2}, ..., y_{n,k} + \frac{q^1_{n,k}}{2}\right) \\ y_{1,(k+1)} &= y_{1,k} + q^2_{1,k} \qquad y_{2,(k+1)} = y_{2,k} + q^2_{2,k} \qquad y_{n,(k+1)} = y_{n,k} + q^2_{n,k} \end{aligned} \tag{102}$$

Es decir, se calculan primero los $q^1_{1,k}, q^1_{2,k}, ..., q^1_{n,k}$. Luego se calculan los $q^2_{1,k}, q^2_{2,k}, ..., q^2_{n,k}$. Finalmente se calculan los $y_{1,(k+1)}, y_{2,(k+1)}, ..., y_{n,(k+1)}$. Veamos un ejemplo de la aplicación de este método al sistema Lotka - Volterra de 2 ecuaciones diferenciales.

Ejemplo 2 *El modelo de Lotka - Volterra predice la evolución de una población con 2 especies, una presa $y_1(t)$ y la otra predadora $y_2(t)$. Se supone que la población presa tiene suficiente comida, su natalidad es proporcional a la cantidad de presas vivas: $k_1 y_1(t)$ y su mortalidad depende del número de presas y predadores: $k_2 y_1(t) y_2(t)$. La natalidad de la población predador es: $k_3 y_1(t) y_2(t)$ y su mortalidad es: $k_4 y_2(t)$. Se expresa el cambio en la poblaciones presa y predador mediante el sistema de ecuaciones diferenciales: Para $t = 1, ..., 4$*

$$\begin{aligned}\frac{dy_1}{dt} &= k_1 y_1 - k_2 y_1 y_2 \quad k_1 = 3 \quad k_2 = 0.002 \quad y_1(t_0 = 1) = 1000\\ \frac{dy_2}{dt} &= k_3 y_1 y_2 - k_4 y_2 \quad k_3 = 0.0006 \quad k_4 = 0.5 \quad y_2(t_0 = 1) = 500\end{aligned}$$

En este caso:

$$f_1(t, y_1, y_2) = k_1 y_1 - k_2 y_1 y_2 \qquad f_2(t, y_1, y_2) = k_3 y_1 y_2 - k_4 y_2 \tag{103}$$

Para aplicar el método de Runge-Kutta de orden 2 se construye primero la iteración para cada y_1, y_2 :

$$\begin{aligned}t_0 &= 1, T = 4, h = 0.25\\ h &= \frac{(T - t_0)}{m} = 0.25 \Rightarrow m = 12\\ t_k &= t_0 + hk = 1 + 0.25k \qquad \forall k = 1, ..., 12\\ f_1(t, y_1, y_2) &= k_1 y_1 - k_2 y_1 y_2 = 3y_1 - 0.002 y_1 y_2\\ f_2(t, y_1, y_2) &= k_3 y_1 y_2 - k_4 y_2 = 0.0006 y_1 y_2 - 0.5 y_2\end{aligned} \tag{104}$$

$$\begin{aligned}q^1_{1,k} &= h f_1(t, y_{1,k}, y_{2,k}) = 0.25\left(3y_{1,k} - 0.002 y_{1,k} y_{2,k}\right)\\ q^1_{2,k} &= h f_2(t, y_{1,k}, y_{2,k}) = 0.25\left(0.0006 y_{1,k} y_{2,k} - 0.5 y_{2,k}\right)\\ q^2_{1,k} &= h f_1\left(t_k + \frac{h}{2}, y_{1,k} + \frac{q^1_{1,k}}{2}, y_{2,k} + \frac{q^1_{2,k}}{2}\right)\\ &= 0.25\left(3\left(y_{1,k} + \frac{q^1_{1,k}}{2}\right) - 0.002\left(y_{1,k} + \frac{q^1_{1,k}}{2}\right)\left(y_{2,k} + \frac{q^1_{2,k}}{2}\right)\right)\\ q^2_{2,k} &= h f_2\left(t_k + \frac{h}{2}, y_{1,k} + \frac{q^1_{1,k}}{2}, y_{2,k} + \frac{q^1_{2,k}}{2}\right)\\ &= 0.25\left(0.0006\left(y_{1,k} + \frac{q^1_{1,k}}{2}\right)\left(y_{2,k} + \frac{q^1_{2,k}}{2}\right) - 0.5\left(y_{2,k} + \frac{q^1_{2,k}}{2}\right)\right)\end{aligned} \tag{105}$$

$$\begin{aligned}y_{1,k+1} &= y_{1,k} + q^2_{1,k}\\ y_{2,k+1} &= y_{2,k} + q^2_{2,k}\end{aligned}$$

Y luego se desarrollan los cálculos, mostrándolos en una tabla como la siguiente:

k	0	1	2	3	4	5	6	7	8	9	10	11	12
t_k	1	1.25	1.5	1.75	2.0	2.25	2.5	2.75	3.0	3.25	3.5	3.75	4.0
$q^1_{1,k}$	500	784.9	1113.8	1182.9	127.3	-2034.8	-1.5304	-682.3	-303.9	-136.8	-63.6	-30.9	-
$q^1_{2,k}$	12.5	62.8	163.9	388.8	846.1	1.1565	366.9	-71.3	-230.9	-280.8	-283.7	-267.5	-
$q^2_{1,k}$	621.1	943.3	1227.8	941.3	-885.7	-2315.1	-836.7	-341.5	-161.9	-79.5	-40.0	-20.5	-
$q^2_{2,k}$	31.6	99.7	244.8	570.1	1111.4	939.4	-36.3	-245.8	-293.3	-295.9	-280.2	-257.3	-
$y_{1,k}$	1000	1621.1	2564.4	3792.2	4733.5	3847.8	1532.7	696.0	354.4	192.5	113.0	73.1	52.6
$y_{2,k}$	500	531.6	631.3	876.1	1446.2	2557.6	3497.0	3460.7	3214.8	2921.6	2625.7	2345.5	2088.1

Se puede apreciar la evolución de la población presa y predadora en la siguiente figura.

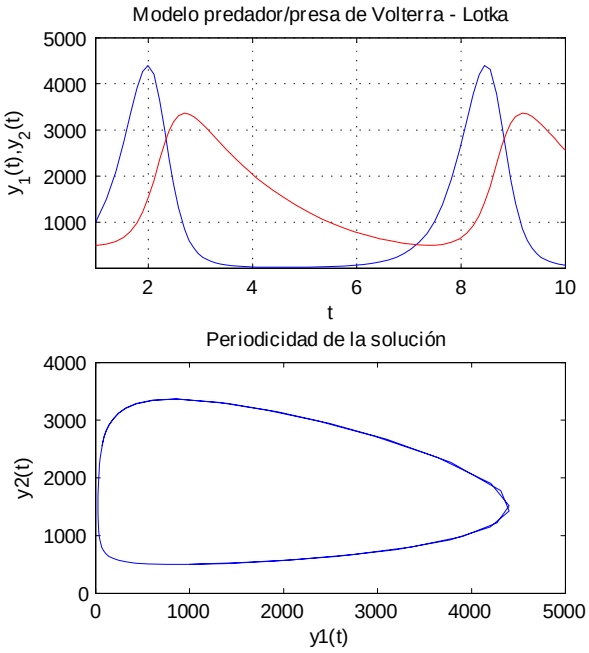

Figura 7. Solución sistema de Volterra - Lotka

Si comparamos los cálculos resumidos en la tabla con los valores que se muestran en la Figura 4, podemos apreciar diferencias, que se deben a los errores generados por el paso $h = 0.25$. *Para disminuir estos errores, el paso debe ser escogido tal que* $h \leq 0.1$.

El método de Runge-Kutta de orden 4 para el sistema (97) está definido por, ver refs. [1, 4]: Para $k = 1, ..., m$, dados los puntos $t_k, y_{1,k}, y_{2,k}, ..., y_{n,k}$ se calculan los puntos $y_{1,(k+1)}, y_{2,(k+1)}, ..., y_{n,(k+1)}$ según:

$$
\begin{aligned}
q^1_{j,k} &= hf_j\left(t_k, y_{1,k}, y_{2,k}, ..., y_{n,k}\right) \quad \forall j = 1, ..., n \\
q^2_{j,k} &= hf_j\left(t_k + \frac{h}{2}, y_{1,k} + \frac{q^1_{1,k}}{2}, y_{2,k} + \frac{q^1_{2,k}}{2}, ..., y_{n,k} + \frac{q^1_{n,k}}{2}\right) \quad \forall j = 1, ..., n \\
q^3_{j,k} &= hf_j\left(t_k + \frac{h}{2}, y_{1,k} + \frac{q^1_{1,k}}{2}, y_{2,k} + \frac{q^1_{2,k}}{2}, ..., y_{n,k} + \frac{q^1_{n,k}}{2}\right) \quad \forall j = 1, ..., n \\
q^4_{j,k} &= hf_j\left(t_k + h, y_{1,k} + q^3_{1,k}, y_{2,k} + q^3_{2,k}, ..., y_{n,k} + q^3_{n,k}\right) \quad \forall j = 1, ..., n
\end{aligned}
\tag{106}
$$

$$
\begin{aligned}
y_{1,(k+1)} &= y_{1,k} + \tfrac{1}{6}\left(q^1_{1,k} + 2q^2_{1,k} + 2q^3_{1,k} + q^4_{1,k}\right) \\
y_{2,(k+1)} &= y_{2,k} + \tfrac{1}{6}\left(q^1_{2,k} + 2q^2_{2,k} + 2q^3_{2,k} + q^4_{2,k}\right) \\
&\vdots \\
y_{n,(k+1)} &= y_{n,k} + \tfrac{1}{6}\left(q^1_{n,k} + 2q^2_{n,k} + 2q^3_{n,k} + q^4_{n,k}\right)
\end{aligned}
$$

Su error es de orden $O(h^5)$. Claramente, a medida que aumenta el orden del método aumenta su precisión, pero tambien aumenta la cantidad de cálculo a realizar. Para mejorar la precisión bastar disminuir el valor de h, aumentando por lo tanto los puntos de la malla, o subir el orden del método de Runge - Kutta. Se recomienda aplicar el método de Runge-Kutta de orden 4, por su precisión y estabilidad numérica.

Los comandos de Matlab que permiten resolver edo también se pueden aplicar a sistemas de ecuaciones diferenciales. La forma de utilizar estos comandos es:

$$[t, y] = dsolver('funcion', [t0\ T], y0, options) \tag{107}$$

donde *dsolver* es alguno de los comandos $ode45, ode23, ode15s, ode23s$. Las salidas de *dsolver* son:

1) t: vector de instantes de tiempo, donde $t_0 \leq t_k \leq T\ \forall k = 1, ..., n$. Matlab determina este vector adaptivamente, dependiendo de la complejidad de la función.

2) Solución de la ecuación diferencial evaluada en los instantes de tiempo t :

$$y = \begin{bmatrix} y_{1,0} & y_{1,1} & \cdots & y_{1,m} \\ y_{2,0} & y_{2,1} & \cdots & y_{2,m} \\ \vdots & \vdots & \cdots & \vdots \\ y_{n,0} & y_{n,1} & \cdots & y_{n,m} \end{bmatrix} \tag{108}$$

Los argumentos de *dsolver* son:

1) $'funcion'$ es la función $f(t, y)$ que define la ecuación diferencial. Debe ser ingresada como función .m. y ser pasada por referencia.

2) $[t0\ T]$ define el intervalo donde se revolverá la ecuación diferencial.

3) $y0 = [y_{1,0}\ ...\ y_{n,0}]$ es el vector de valores iniciales

4) *options* son las opciones del comando *dsolver*. Utilizaremos las opciones por default.

Por ejemplo, para resolver el sistema de Volterra - Lotka:

$$\begin{aligned} \frac{dy_1}{dt} &= k_1 y_1 - k_2 y_1 y_2 \\ y_1(t_0 &= 1) = 1000 \qquad k_1 = 3, k_2 = 0.002 \\ \frac{dy_2}{dt} &= k_3 y_1 y_2 - k_4 y_2 \\ y_2(t_0 &= 1) = 500 \qquad k_3 = 0.0006, k_4 = 0.5 \end{aligned} \tag{109}$$

Primero se debe programar el sistema como función .m:

```
function dy=Volterra_Lotka(t,y)
%Modelo Volterra Lotka para un sistema depredador - presa
%Condiciones iniciales:
%y1(1)=1000; y2(1)=500; t in [1,4]
%Parametros
k1=3.0; k2=0.002; k3=0.0006; k4=0.5;
%Sistema ecuaciones diferenciales
dy=zeros(2,1);
dy(1) = k1*y(1)-k2*y(1)*y(2);
dy(2) = k3*y(1)*y(2)-k4*y(2);
end
```

Luego se ejecuta algún comando de *dsolver* pasando la función f por referencia. Se recomienda utilizar *ode*45 por su precisión y rapidez de cálculo. Si *ode*45 demora en entregar la solución, se debe ocupar *ode*15*s*:

$$[t, y] = ode45(@Volterra\_Lotka, [1\ 4], [1000\ 500]); \tag{110}$$

Resolveremos a continuación un sistema de ecuaciones diferenciales no lineales de primer orden que modela las reacciones químicas del ozono en la alta atmósfera, ver ref. [4].

ACT 3: El proceso químico de decaimiento del ozono (O_3) en la alta atmósfera debido al efecto de la radiación del sol se describe mediante la fórmulas:

$$O_3 + O_2 \underset{k_2}{\overset{k_1}{\rightleftharpoons}} O + 2O_2 \qquad O_3 + O \xrightarrow{k_3} 2O_2 \tag{111}$$

donde k_1, k_2, k_3 son los parámetros cinéticos. Si se denota por y_1, y_2, y_3 las concentraciones de O_3, O, O_2, la reacción (111) se puede modelar mediante el sistema de edo no lineales de primer orden:

$$\begin{aligned} \frac{dy_1}{dt} &= -k_1 y_1 y_3 + k_2 y_2 y_3^2 - k_3 y_1 y_2 \\ \frac{dy_2}{dt} &= k_1 y_1 y_3 - k_2 y_2 y_3^2 - k_3 y_1 y_2 \\ \frac{dy_3}{dt} &= -k_1 y_1 y_3 + k_2 y_2 y_3^2 + k_3 y_1 y_2 \end{aligned} \tag{112}$$

Para simplificar este sistema usualmente se supone que la concentración de O_2 es constante. Dándole valor a las constantes y re-escalando se obtiene el sistema:

$$\begin{aligned} \frac{dy_1}{dt} &= -y_1 - y_1 y_2^2 + 294 y_2 \quad y_1(0) = 1 \\ \frac{dy_2}{dt} &= \frac{(y_1 - y_1 y_2)}{98} - 3y_2 \quad y_2(0) = 0 \end{aligned} \tag{113}$$

1) Resuelva el sistema (113) mediante los comandos *ode*45 y *ode*15*s* en el intervalo $[0, 10000]$. Incluya en el informe como utilizó los comandos en cada caso y los gráficos de:

   (a) $y_1(t)$ v/s $t, y_2(t)$ v/s $t, \dfrac{dy_1}{dt}$ v/s $t, \dfrac{dy_2}{dt}$ v/s t

   (b) $y_1(t)$ v/s $\dfrac{dy_1}{dt}, y_2(t)$ v/s $\dfrac{dy_2}{dt}$

   (c) $y_1(t)$ v/s $y_2(t), \dfrac{dy_1}{dt}$ v/s $\dfrac{dy_2}{dt}$

   en el intervalo [0 10000]. Qué puede decir acerca de la rapidez de la reacción de O_3 y O ?

2) Aproxime el sistema (113) mediante un desarrollo de Taylor de primer orden en torno a $(y_1(0), y_2(0)) = (1, 0)$, para obtener un sistema de ecuaciones diferenciales lineales y resuelva este sistema en el intervalo $[0, 10000]$. Incluya en el informe como utilizó los comandos en cada caso y los gráficos de:

   (a) $y_1(t)$ v/s $t, y_2(t)$ v/s $t, \dfrac{dy_1}{dt}$ v/s $t, \dfrac{dy_2}{dt}$ v/s t

   (b) $y_1(t)$ v/s $\dfrac{dy_1}{dt}, y_2(t)$ v/s $\dfrac{dy_2}{dt}$

   (c) $y_1(t)$ v/s $y_2(t), \dfrac{dy_1}{dt}$ v/s $\dfrac{dy_2}{dt}$

   en el intervalo [0 10000]. Qué puede decir acerca de la rapidez de la reacción de O_3 y O ?

   La aproximación lineal mantiene la rapidez de la reacción de O_3 y O observada en el caso no lineal ?

6 Solución de ecuaciones diferenciales no lineales rígidas

Como mencionamos anteriormente, las edo rígidas (stiff) son de difícil solución numérica en la práctica debido a singularidades o fuertes variaciones en un intervalo pequeño, ver refs. [2, 4]. Crear una edo rígida es más simple de lo que parece. Por ejemplo, consideremos la edo no lineal:

$$\frac{dy}{dt} = \frac{\sin(y^2 + t)}{(1 + \sin(t^2))} \quad y(t_0) = 0 \quad \forall t \in [0, 25] \tag{114}$$

muy similar a la ecuación (93), pero con mayor variabilidad. El comando *ode*45 no converge en este caso. Al resolver esta edo con el comando *ode*15*s* se obtiene:

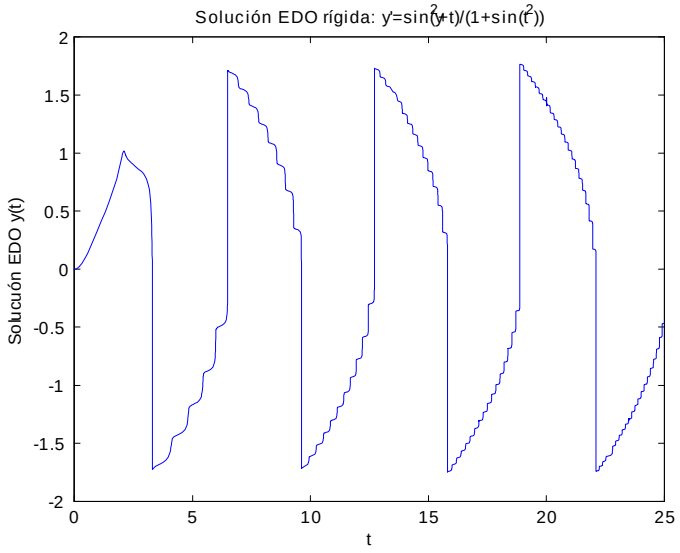

Figura 9. Solución numérica edo no lineal rígida: $\frac{dy}{dt} = \frac{\sin\left(y^2 + t\right)}{(1 + \sin(t^2))}$

Se observa del gráfico las fuertes variaciones de la solución. En la proxima actividad se resolverá una edo stiff de segundo orden.

ACT 4: Oscilador de Van der Pol

Considere la ecuación del oscilador de Van der Pol que modela un circuito electrónico utilizado en las radios alrededor de 1920:

$$\frac{d^2y}{dt^2} + y(t) + \mu(y^2(t) - 1)\frac{dy}{dt} = 0 \tag{115}$$

La función $y(t)$ es la corriente eléctrica y la constante μ mide la no linealidad del sistema.

1) Transforme la edo de Van der Pol en un sistema de ecuaciones diferenciales de primer orden aplicando la transformación standard:

$$y_1(t) = y(t) \qquad y_2(t) = \frac{dy}{dt} \tag{116}$$

Programe el sistema resultante como función .m. Llame a esta función vdp.m. Escriba el código en el informe.

2) Aproxime la edo (115) mediante una edo lineal. Resuelva esta edo lineal en el intervalo $[0, 6000]$. Incluya en el informe la aproximación, como utilizó los comandos en cada caso y los gráficos de: $y(t)$ v/s t, $\frac{dy}{dt}$ v/s t, $\frac{dy}{dt}$ v/s $y(t)$.

3) Para $\mu = 200, 400, 600, 800, 1000$ resuelva la edo no lineal de Van der Pol en el intervalo [0 6000] utilizando el comando *ode*15*s* y las condiciones iniciales $[1; 0]$ y $[2; 0]$. Incluya en el informe como utilizó el comando en cada caso y los gráficos de $y_1(t)$ y $y_2(t)$ en el intervalo [0 6000]. Podrá apreciar el comportamiento oscilatorio y periódico de la solución.

   Incluya en el informe como utilizó los comandos en cada caso y los gráficos de: $y(t)$ v/s t, $\frac{dy}{dt}$ v/s t, $\frac{dy}{dt}$ v/s $y(t)$.

   Discuta la influencia del parámetro μ en la rígidez de la solución.

   Compare la solución lineal y no-lineal. La aproximación lineal reproduce el comportamiento de la solución no lineal ?

4) En este caso, se puede hacer un análisis de error de los métodos ? Proponga una forma de estudiar el error.

References

[1] Burden, R., J.D. Faires, Numerical Analysis, Seventh Edition, Brooks Cole, 2000.

[2] Gerald, C., P. O. Wheatley, Applied Numerical Analysis, Seventh Edition, Pearson – Addison Wesley, 2004.

[3] Henrici, P., Elements of Numerical Analysis, Wiley, 1964.

[4] Stoer, J., R. Bulirsch, Introduction to Numerical Analysis, Third Edition, Springer, 1993.

[5] Kiusalaas, J., Numerical Methods in Engineering with Matlab, Cambridge University press, 2005.

[6] Harman, T. L., J. Dabney, N. Richert, Advanced Engineering Mathematics with Matlab, Second Edition, Brooks/Cole, 2000.