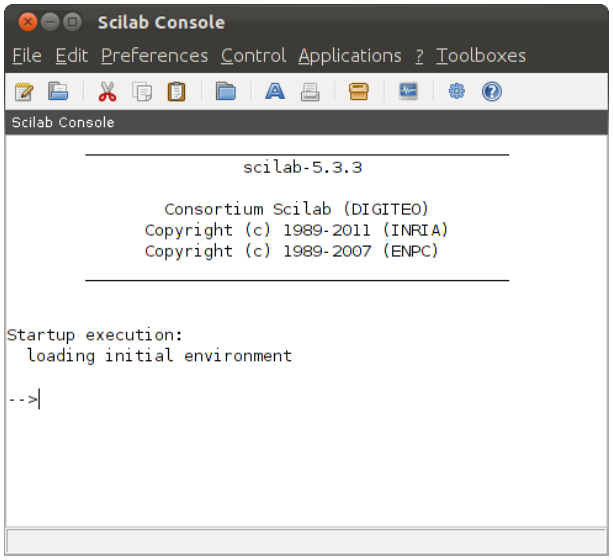
Universidad del Valle
Escuela de Ingeniería de Sistemas y Computación
Curso: Métodos Numéricos
Docente: Daniel Barragán Calderón

Clase Introducción a la Programación con Scilab (Parte 5)

Introducción

Scilab fue creado en 1990 por investigadores en INRIA (*Institut national de recherche en informatique et en automatique*) y ENPC (*École nationale des ponts et chaussées*). Scilab es un lenguaje de programación de código abierto, multiplataforma, orientado a cálculos numéricos. Puede ser usado para procesamiento de señales, análisis estadístico, tratamiento de imágenes, simulación de fluidos, optimización y modelamiento y simulación de sistemas dinámicos.

Enlace de descarga:
http://www.scilab.org/



Interfaz Gráfica Scilab

Guía Básica
En esta guía se abordaran otros aspectos de la solución de ecuaciones diferenciales con Scilab.

Introducción
Scilab permite solucionar ecuaciones diferenciales ordinarias por medio de la función **ode**. En esta guía se emplearan la siguiente estructura de llamado a la función **ode**:

[y] = ode["solver", y0, t0, t, f]

Donde:
type: cadena de caracteres con el nombre del método a emplear. En esta guía se usará "rk" que corresponden al método de runge kutta de cuarto orden.
y0: vector o matriz con las condiciones iniciales
t0: un valor con el tiempo inicial
t: un vector con los valores en los que la solución es estimada (ti:stepsize:tf)
f: una función con la ecuación diferencial

Para mas información acerca de otras posibilidades de ejecución consulte la ayuda de Scilab de la función **ode**.

Ecuaciones Diferenciales Ordinarias
Ejemplo 1: En el siguiente ejemplo se soluciona la siguiente ecuación diferencial por medio del método de runge kutta de cuarto orden:

$$\frac{dy}{dt} = y^2 - y\sin(t) + \cos(t)$$

Condición inicial: $y(0) = 0$

Para el desarrollo de la solución se sugiere crear un archivo en scilab de nombre **ejemplo_ode1.sci** en este archivo debe incluir las líneas de código que se mencionan a continuación.

De acuerdo con los argumentos que recibe la función **ode**, se debe construir una función de scilab que represente la ecuación diferencial.

```
function ydot=f(t,y)
   ydot = y^2-y*sin(t)+cos(t)
endfunction
```

Posteriormente se inicializan los argumentos y se hace el llamado a la función **ode.**

```
y0=0; // condiciones iniciales
t0=0; // tiempo inicial
t=0:0.1:%pi; // valores de evaluacion
y = ode("rk", y0, t0, t, f); // solucion
```

Puede graficar la solución incluyendo las siguientes lineas.

```
plot(t,y) // grafica de la solucion
xlabel("t","fontsize",4)
ylabel("y","fontsize",4)
set(gca(),"grid",[1 1])
```

El archivo **ejemplo_ode1.sci** debe quedar de la siguiente forma.

```
function ydot=f(t, y)
   ydot=y^2-y*sin(t)+cos(t)
endfunction
y0=0;
t0=0;
t=0:0.1:%pi;
y = ode("rk", y0, t0, t, f);
plot(t,y) // grafica de la solucion
xlabel("t","fontsize",4)
ylabel("y","fontsize",4)
set(gca(),"grid",[1 1])
```

Gráfica de la solución

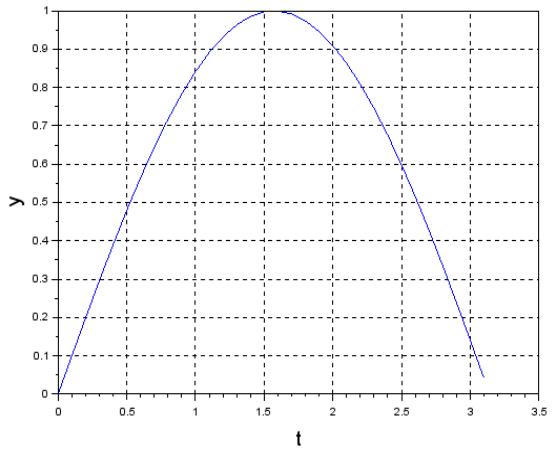


Figura 1: Solución aproximada a la ecuación diferencial

Ejemplo 2: En el siguiente ejemplo se soluciona la siguiente ecuación diferencial por medio del método de runge kutta de cuarto orden:

$$\frac{dx}{dt} = A * x(t) + B * u(t)$$

$$u(t) = \sin(omega * t)$$

Como la función posee múltiples argumentos: A, u, B, estos se pasan a la función **ode** como una lista

```
function xdot=linear(t, x, A, u, B, omega)
   xdot=A*x+B*u(t,omega)
endfunction

function ut=u(t,omega)
   ut=sin(omega*t)
endfunction

A=[1 1;0 2];
B=[1;1];
omega=5;
y0=[1;0];
t0=0;
t=[0.1,0.2,0.5,1];
ode(y0,t0,t,list(linear,A,u,B,omega))
```

Ecuaciones Diferenciales de Alto Orden

Ejemplo 3: En el siguiente ejemplo se soluciona la siguiente ecuación diferencial por medio del método de runge kutta de cuarto orden:

$$\frac{d^2 y}{dt} + \sin(y) = 0$$

Condiciones iniciales: $y(0) = \frac{\pi}{4}$, $\frac{dy}{dt} = 0$

Para solucionar una ecuación diferencial de orden dos o superior en Scilab se debe re-expresar el problema de la siguiente manera.

$$\frac{dy}{dt} = z$$

$$\frac{dz}{dt} = -\sin(y)$$

La función a declarar en Scilab debe tener estas dos componentes o ecuaciones diferenciales y recibir como parámetro las condiciones iniciales para **y** y para **dy/dt**. Nótese que para cada una de las ecuaciones diferenciales el problema proporciona la condición inicial, es decir, se tiene el valor inicial para el valor de **y** y se tiene la condición inicial para **dy/dt**.

El vector de tiempos en que se evaluará la solución y las condiciones iniciales para **y** y **dy/dt** se almacenarán en un vector. Los vectores tendrán la siguiente estructura.

y0 = [condición inicial para y; condición inicial para dy/dt]
t0 = [tinicio:stepsize:tfin]

Para representar el problema se crea una función de Scilab que recibe como parámetro el vector de tiempos en que se evaluará la solución **t** y el vector de condiciones iniciales **y**. Nótese que dentro de la función se almacena en un vector el resultado de evaluar cada condición inicial en la ecuación diferencial respectiva.

```
function dy = dydtsys(t,y)
dy = [y(2); -sin(y(1))];
endfunction
```

Nota: Observando detenidamente la línea de código: **dy = [y(2); -sin(y(1))]** , cabe comprender lo siguiente:

- La primera columna de **dy** representa a **dy/dt = z**, la segunda columna de **dy** representa **dz/dt = -sin(y)**
- La primera columna de **y** es la condición inicial para el valor de **y**, la segunda columna de **y** es la condición inicial para **dy/dt**

Posteriormente se inicializan los argumentos y se hace el llamado a la función **ode.**

```
y0 = [%pi/4;0] // puesto que el problema plantea que y = %pi/4, dy/dt = 0
t0 = 0.0 // puesto que el problema plantea que el inicio es t=0
t=0:0.25:20;
y=ode("rk",y0, t0, t, dydtsys);
```

La salida de la función anterior es una matriz de nombre **y** con dos columnas que corresponden a los valores de **y** (primera columna) y **dy/dt** (segundo columna). Puede graficar la solución mostrando los valores de la primera columna con respecto al tiempo por medio de las siguientes líneas.

```
plot(t, y(1,:),'color','red')
xlabel("t","fontsize",4)
ylabel("y","fontsize",4)
set(gca(),"grid",[1 1])
```

El archivo **ejemplo_ode2.sci** debe quedar de la siguiente forma.

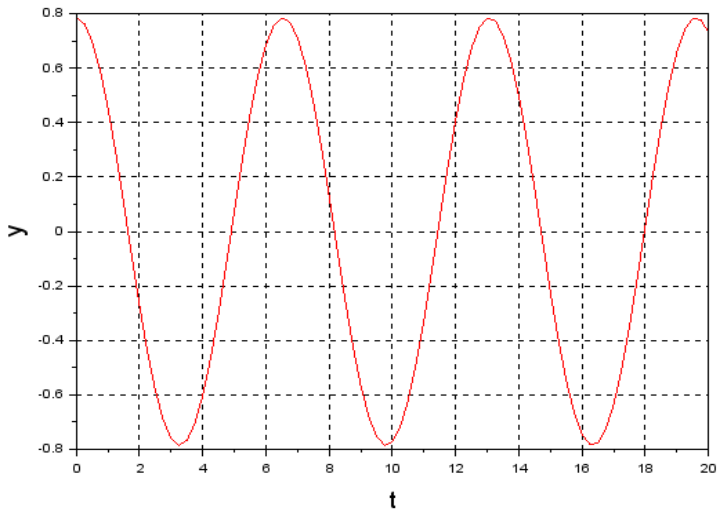
```
function dy = dydtsys(t,y)
dy = [y(2); -sin(y(1))];
endfunction

y0 = [%pi/4;0]
```

```
t0 = 0.0
t=0:0.25:20;
y=ode("rk",y0, t0, t, dydtsys);

plot(t, y(1,:),'color','red')
xlabel("t","fontsize",4)
ylabel("y","fontsize",4)
set(gca(),"grid",[1 1])
```

Gráfica de la solución



Ejemplo 4: En el siguiente ejemplo se soluciona la siguiente ecuación diferencial por medio del método de runge kutta de cuarto orden deste t=0 hasta t=4 empleando un tamaño de paso (stepsize) de 0.1

$$\frac{d^2 y}{dt} + 4\, y = 0$$

Condiciones iniciales: $y(0) = 1$, $\frac{dy}{dt} = 0$

Nota: La solución exacta es $y = \cos(2t)$

Para solucionar una ecuación diferencial de orden dos o superior en Scilab se debe re-expresar el problema de la siguiente manera.

$$\frac{dy}{dt} = z$$

$$\frac{dz}{dt} = -4y$$

El script para la solución del problema es el siguiente:

```
function dy = dydtsys(t,y)
dy = [y(2); -sin(y(1))];
endfunction

y0 = [1;0]
t0 = 0.0
t=0:0.1:4;
y=ode("rk",y0, t0, t, dydtsys);

plot(t, y(1,:),'color','red')
xlabel("t","fontsize",4)
ylabel("y","fontsize",4)
set(gca(),"grid",[1 1])
```

La gráfica de comparación de la solución aproximada y la solución analítica es la siguiente:

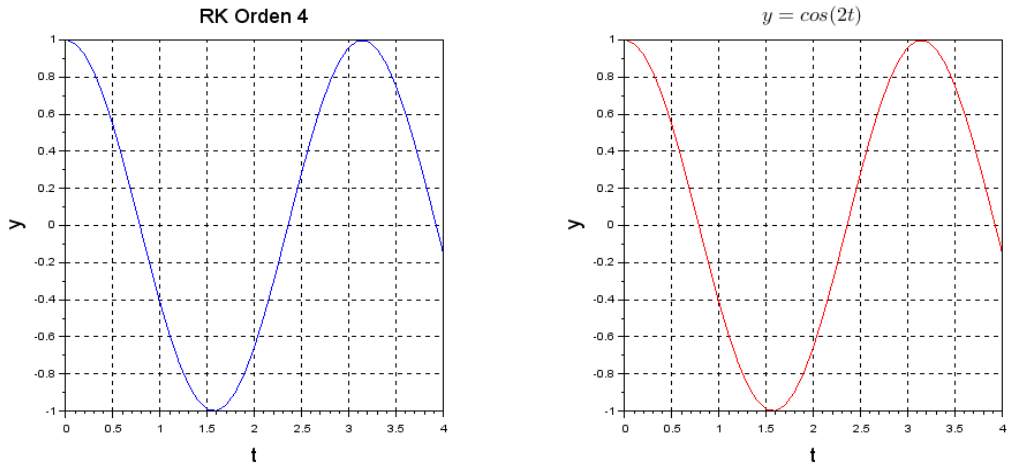


Figura 2: Solución aproximada (izquierda), Solución analítica (derecha)

Problemas

1. Resuelva manualmente empleando los métodos vistos en clase un ejercicio por cada uno de los anexos problemasCap22.pdf a problemasCap24.pdf.
2. Resuelva un problema de su carrera que implique la solución de ecuaciones diferenciales ordinarias (ODE)
3. Investigue una forma de como solucionar ecuaciones diferenciales parciales (PDE) en Scilab. Resuelva un problema de ingeniería empleando la forma encontrada.