

## Cálculo Numérico (521230)

### Laboratorio 9

### Ecuaciones Diferenciales Ordinarias

El objetivo de este laboratorio es aprender técnicas para la resolución numérica de problemas de valores iniciales (P.V.I.) para ecuaciones diferenciales ordinarias (E.D.O.) y sistemas de E.D.O. MATLAB tiene varios comandos para la resolución numérica de P.V.I. para E.D.O.:

```
Ordinary differential equation solvers.  
(If unsure about stiffness, try ODE45 first, then ODE15S.)  
ode45    - Solve non-stiff differential equations, medium order method.  
ode23    - Solve non-stiff differential equations, low order method.  
ode113   - Solve non-stiff differential equations, variable order method.  
ode23t   - Solve moderately stiff differential equations, trapezoidal rule.  
ode15s   - Solve stiff differential equations, variable order method.  
ode23s   - Solve stiff differential equations, low order method.  
ode23tb  - Solve stiff differential equations, low order method.
```

Como se ve en esta lista, hay métodos para resolver E.D.O. *stiff* y no *stiff*. Además hay métodos de orden bajo, medio, alto y variable.

Todos ellos tienen una sintaxis semejante. Por ejemplo, para resolver el P.V.I.

$$\begin{cases} y' = f(t, y), \\ y(t_0) = y_0, \end{cases}$$

en el intervalo  $[t_0, t_f]$  mediante el comando `ode45` en su opción más sencilla, debe ejecutarse:

```
[t,y]=ode45('f',[to tf],yo);
```

donde:

- `f` es el nombre de la función  $f(t, y)$  (típicamente definida mediante un programa *function* en un archivo `f.m`);
- `to` y `tf` son los extremos del intervalo donde se desea conocer la solución;
- `yo` es el valor de la solución en `to` (es decir el valor de la condición inicial  $y(t_0) = y_0$ );
- `t` devuelve los valores de la variable independiente  $t$  donde el método calcula el valor de la solución;
- `y` devuelve los valores de la solución en cada uno de los puntos  $t$ .

Estos comandos no requieren como dato un paso de integración  $h$  pues todos ellos determinan de manera automática en cada paso  $k$ , el tamaño del paso de integración  $h_k$  necesario para mantener los errores por debajo de una tolerancia determinada. Los valores de  $t$  que entrega corresponden a los puntos  $t_k = t_{k-1} + h_k$ ,  $k = 1, 2, \dots$ , en los que el comando necesitó calcular el valor de  $y(t_k)$ .

Si se desea conocer la solución para ciertos valores de  $t$ , puede alternativamente ejecutarse:

```
[t,y]=ode45('f',tspan,yo);
```

donde `tspan` es el vector de valores donde se desea conocer la solución. Por ejemplo, `tspan=0:0.1:1`. En ese caso, la salida `t` coincide con `tspan` e `y` contiene los valores de la solución en esos puntos.

La tolerancia predeterminada de estos métodos es  $10^{-3}$ , para el error relativo, y  $10^{-6}$ , para el error absoluto. Si se desea calcular la solución con otras tolerancias, deben prefijarse las opciones elegidas mediante el comando `odeset`. Además, en la ejecución del comando para resolver la E.D.O., debe agregarse el parámetro adicional de opciones. La sintaxis para realizar esto es, por ejemplo:

```
options=odeset('RelTol',1e-6,'AbsTol',1.e-8);  
[t,y]=ode45('f',[to tf],yo,options);
```

Si se ejecuta `options=odeset('RelTol',1e-6,'AbsTol',1.e-8)` sin el “;” puede verse que hay otras opciones que pueden prefijarse, además de las tolerancias de los errores.

Por ejemplo, si se desea resolver el P.V.I.

$$\begin{cases} y' = y, \\ y(0) = 1, \end{cases}$$

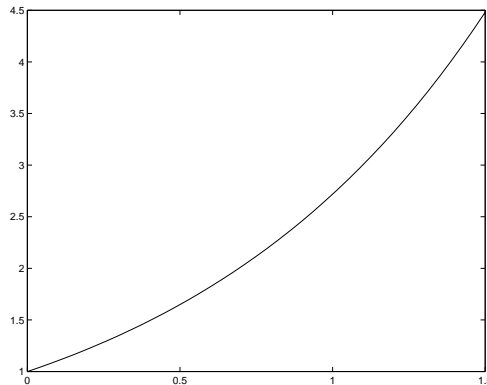
en el intervalo  $[0, 1.5]$ , mediante el comando `ode45` y visualizar la solución obtenida, debe crearse un fichero `f.m` como sigue:

```
function z=f(t,y)  
z=y;
```

y ejecutarse:

```
[t,y]=ode45('f',[0 1.5],1);  
plot(t,y)
```

Así se obtiene la siguiente gráfica:



El siguiente ejemplo resuelve la misma ecuación en los puntos `t=0:0.1:1.5`, con error absoluto menor a  $10^{-6}$  y calcula los errores cometidos restando los valores calculados a los de la solución verdadera, que en este caso es  $y(t) = e^t$ :

```

options=odeset('AbsTol',1.e-6);
tspan=0:.1:1.5;
[t,y]=ode45('f',tspan,1,options);
error=exp(t)-y
error =
    1.0e-06 *
         0
    -0.0003
    -0.0248
    -0.0448
    -0.0076
    -0.0415
    -0.0694
    -0.0200
    -0.0669
    -0.1056
    -0.0402
    -0.1048
    -0.1586
    -0.0721
    -0.1612
    -0.0989

```

La salida que se presenta indica que los errores son efectivamente menores en valor absoluto a  $10^{-6}$ .

**Evaluación de la solución de una E.D.O.** Si se desea tener la solución de la E.D.O. como una función a evaluar, se puede utilizar el comando `deval`. La siguiente instrucción :

```

sxint=deval(sol,xint);

```

entrega los valores de la solución de una E.D.O. donde `sol` es la estructura que entrega el comando

```

sol=ode45(@f,[to tf],yo);

```

o cualquier otro comando de resolución de P.V.I., y `xint` es el punto o el vector de puntos donde se desea evaluar la solución.

1. Construya una función `Y(t)` que evalúe la solución de la E.D.O. del ejemplo anterior, y utilice dicha función para graficar la solución de la siguiente manera :

```

>> xx=0:0.01:1.5;
>> plot(xx,Y(xx));

```

Que beneficios cree usted que se obtienen al contar con una función que permite evaluar la solución de una E.D.O. en lugar de evaluarla directamente con el comando `ode45` ?

La resolución de P.V.I. para sistemas de E.D.O. se realiza mediante los mismos comandos. En tal caso,  $\mathbf{f}(t, \mathbf{y})$  debe ser una función a valores vectoriales (es decir un vector columna de funciones) e  $\mathbf{y}$  un vector columna de variables de la misma dimensión. Además, la condición inicial  $\mathbf{y}_0$  también debe ser un vector columna de la misma dimensión.

Por ejemplo, consideremos el P.V.I.

$$\begin{cases} x' = y, & x(0) = 1, \\ y' = -x, & y(0) = 0, \end{cases}$$

cuya solución exacta es

$$\begin{cases} x = \cos t, \\ y = \sin t. \end{cases}$$

Por lo tanto los puntos  $(x(t), y(t))$  solución de este sistema de E.D.O, describen la circunferencia unitaria.

Este sistema escrito vectorialmente resulta:

$$\begin{cases} Y' = F(t, Y), \\ Y(0) = Y_0, \end{cases} \quad \text{con } Y = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}, \quad F(t, Y) = \begin{pmatrix} y \\ -x \end{pmatrix} = \begin{pmatrix} Y_2 \\ -Y_1 \end{pmatrix} \quad \text{e } Y_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

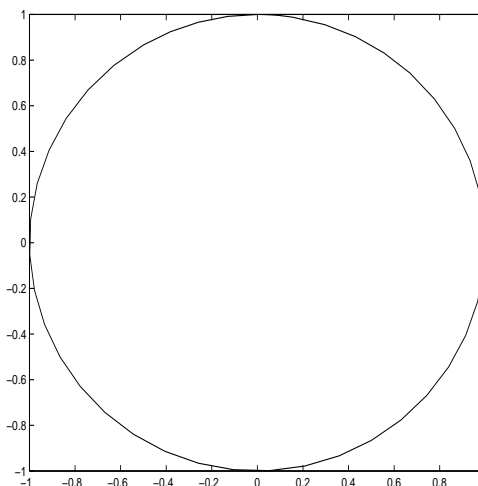
Para resolverlo debe crearse un fichero `F.m` como sigue:

```
function Z=F(t,Y)
Z=[Y(2);-Y(1)];
```

Los siguientes comandos resuelven este P.V.I. en el intervalo  $[0, 2\pi]$  y grafican la curva  $(x(t), y(t))$ , para  $0 \leq t \leq 2\pi$ , que se obtiene:

```
[t,Y]=ode45('F',[0 2*pi],[1;0]);
plot(Y(:,1),Y(:,2));
```

Así se obtiene la siguiente gráfica:



2. El desplazamiento  $u$  de la posición de equilibrio de una masa  $m$ , sujeta a un resorte de constante  $k$ , inmersa en un medio viscoso que ejerce una resistencia al movimiento  $bu'$  (es decir, proporcional a la velocidad de la masa  $u' = du/dt$ ) y sobre la que se ejerce una fuerza  $f$ , se modela mediante la E.D.O.

$$\begin{cases} mu'' + bu' + ku = f, \\ u(0) = u_0, \quad u'(0) = v_0, \end{cases}$$

donde  $u_0$  y  $v_0$  son el desplazamiento y la velocidad de la masa en el instante inicial  $t = 0$ .

Calcular y graficar el desplazamiento de la masa a lo largo de 1 minuto para los siguientes datos:

$$m = 1.2 \text{ kg}, \quad k = 15 \frac{\text{kg}}{\text{s}^2}, \quad b = 0.3 \frac{\text{kg}}{\text{s}},$$

en los siguientes casos:

- cuando la masa se desplaza 1 m de su posición de equilibrio y, desde el reposo, se la deja oscilar libremente;
  - cuando desde su posición de equilibrio y en reposo, se le aplica una fuerza externa periódica  $f(t) = \cos t$  (con el tiempo  $t$  medido en segundos y la fuerza  $f$  en  $\text{kg m/s}^2$ );
  - igual que el anterior pero con  $f(t) = \cos \omega t$ , donde  $\omega = \sqrt{\frac{k}{m}}$  es la frecuencia de oscilación libre del sistema sin amortiguamiento.
3. Considere un ecosistema simple consistente de conejos con una cantidad más que suficiente de alimento y zorros que depredan los conejos para su alimentación. Un modelo clásico debido a Volterra describe este ecosistema mediante el siguiente par de ecuaciones no lineales de primer orden:

$$\begin{cases} \frac{dc}{dt} = 2c - \alpha cz, & c(0) = c_0, \\ \frac{dz}{dt} = -z + \alpha cz, & z(0) = z_0, \end{cases}$$

donde  $t$  es el tiempo medido en años,  $c = c(t)$  es el número de conejos y  $z = z(t)$  el número de zorros, ambos en el instante  $t$ , y  $\alpha$  es una constante positiva que mide la probabilidad de interacción entre miembros de las dos especies.

- Cuando  $\alpha = 0$ , conejos y zorros no interactúan. Resuelva la ecuación diferencial a lo largo de un año en el caso en que inicialmente hay 100 animales de cada especie. Compruebe que en tal caso los conejos hacen lo que mejor saben hacer, mientras los zorros se van muriendo de hambre.
  - Calcule la evolución de ambas poblaciones a lo largo de 12 años en el caso en que la constante de interacción es  $\alpha = 0.01$  y que la población inicial es de 300 conejos y 150 zorros. ¿Qué conclusión puede extraer en este caso?
  - Repita la simulación anterior pero con poblaciones iniciales de 15 conejos y 22 zorros. ¿Cuál es ahora la conclusión?
4. El P.V.I. siguiente,

$$\begin{cases} y' = -\alpha(y - \sin t) + \cos t, \\ y(0) = 1, \end{cases}$$

con  $\alpha$  un número positivo grande, es un ejemplo de problema *stiff*.

La solución de este P.V.I. puede calcularse analíticamente:  $y(t) = e^{-\alpha t} + \sin t$ ,

- Resuelva este problema para  $\alpha = 1000$  en el intervalo  $0 \leq t \leq 1$  mediante los comandos `ode45` y `ode15s`, en ambos casos con tolerancia del error absoluto de  $10^{-3}$ , y grafique la solución obtenida en cada caso, mostrando los puntos donde el método calculó la solución.

- (b) El comando `ode45` se basa en un método *Runge-Kutta-Fehlberg* de orden 4 y es la opción aconsejada por MATLAB para problemas no *stiff*. En cambio, `ode15s` se basa en un método implícito de orden variable y es la opción recomendada por MATLAB para problemas *stiff*.

Verificar que ambos métodos calculan la solución del P.V.I. con error cercano a la tolerancia prefijada, pero que el primero requiere muchísimas más iteraciones que el segundo.

5. En aplicaciones de la aerodinámica aparece la ecuación de *Blasius*,

$$2f''' + ff'' = 0,$$

que da el perfil de velocidad de un fluido incompresible que se desliza sobre una placa delgada (la variable independiente se suele denotar por  $\eta$ ).

Dos condiciones iniciales naturales para este problema son  $f(0) = 0$  y  $f'(0) = 0$ . Además se sabe que  $f'(\eta) \rightarrow 1$  cuando  $\eta \rightarrow \infty$ .

- (a) Experimente con distintos valores de  $f''(0)$  entre 0.1 y 0.5 para ver que valor de  $\eta$  puede tomarse como “ $\infty$ ”.
- (b) Determine el valor correcto de  $f''(0)$  para que  $f'(\eta) \rightarrow 1$  cuando  $\eta \rightarrow \infty$ .
- (c) Se sabe que el momento del fluido es proporcional a

$$\vartheta(u) = \int_0^u f'(\eta)[1 - f'(\eta)] d\eta.$$

Calcule  $\vartheta(5.0)$ .

RAD/RRS/RRA/MSC

<http://www.ing-mat.udec.cl/pregrado/ asignaturas/521230/>  
31/05/06

## Soluciones propuestas

1. Ej. 1.

```
function y=Y(x)
sol=ode45(@f,[0 1.5],1);
y=deval(sol,x);
```

2. Ej. 2(a).

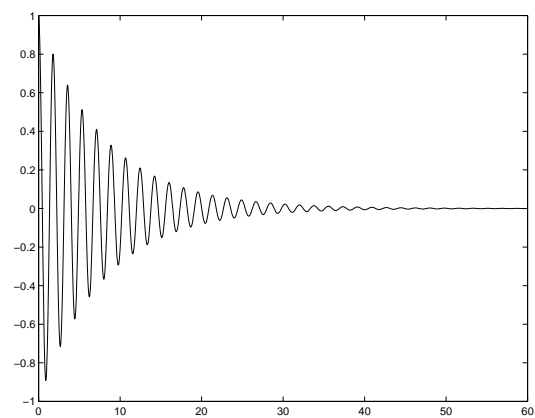
File `F1a.m`:

```
function Z=F1a(t,Y)
m=1.2;
b=0.3;
k=15;
Z=[Y(2); (-k*Y(1)-b*Y(2))/m];
```

Ejecución:

```
>> [t,Y]=ode45('F1a',[0 60],[1;0]);
>> plot(t,Y(:,1))
```

Salida:



3. Ej. 2(b).

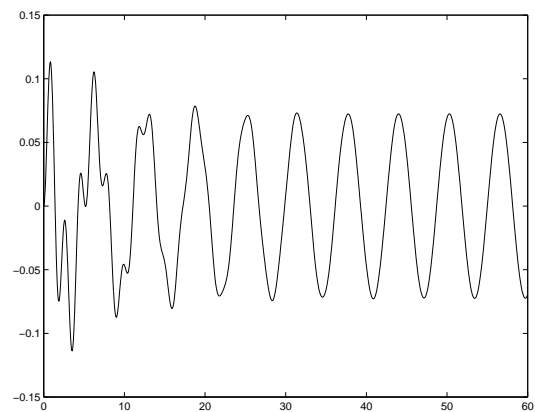
File `F1b.m`:

```
function Z=F1b(t,Y)
m=1.2;
b=0.3;
k=15;
Z=[Y(2); (cos(t)-k*Y(1)-b*Y(2))/m];
```

Ejecución:

```
>> [t,Y]=ode45('F1b',[0 60],[0;0]);
>> plot(t,Y(:,1))
```

Salida:



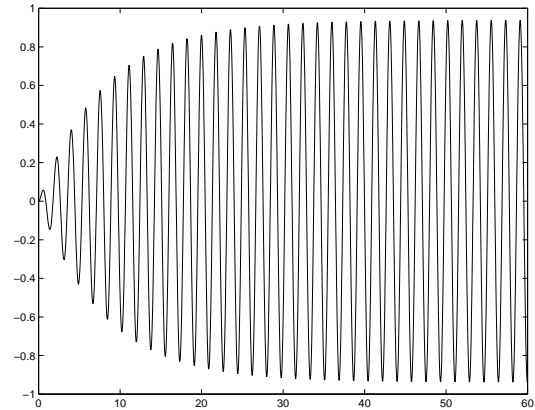
4. Ej. 2(c).

File `F1c.m`:

```
function Z=F1c(t,Y)
m=1.2;
b=0.3;
k=15;
om=sqrt(k/m);
Z=[Y(2);(cos(om*t)-k*Y(1)-b*Y(2))/m];
```

Ejecución: como antes.

Salida:



5. Ej. 3(a).

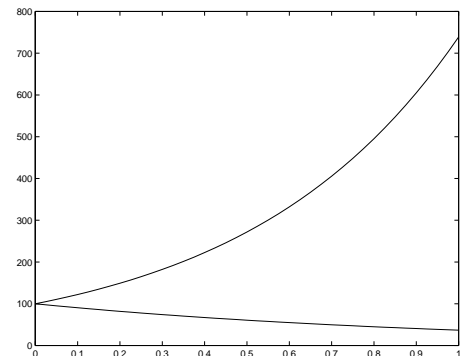
File `F2a.m`:

```
function Z=F2a(t,Y)
alfa=0;
Z=[2*Y(1)-alfa*Y(1)*Y(2);-Y(2)+alfa*Y(1)*Y(2)];
```

Ejecución:

```
>> [t,Y]=ode45('F2a',[0 1],[100;100]);
>> plot(t,Y)
```

Salida:



6. Ej. 3(b).

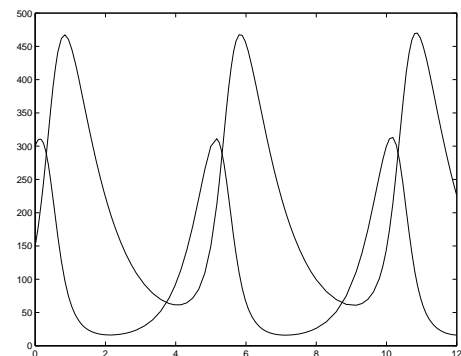
File `F2b.m`:

```
function Z=F2b(t,Y)
alfa=0.01;
Z=[2*Y(1)-alfa*Y(1)*Y(2);-Y(2)+alfa*Y(1)*Y(2)];
```

Ejecución:

```
>> [t,Y]=ode45('F2b',[0 12],[300;150]);
>> plot(t,Y)
```

Salida:



Conclusión: el crecimiento y decrecimiento de ambas poblaciones es periódico con un período de

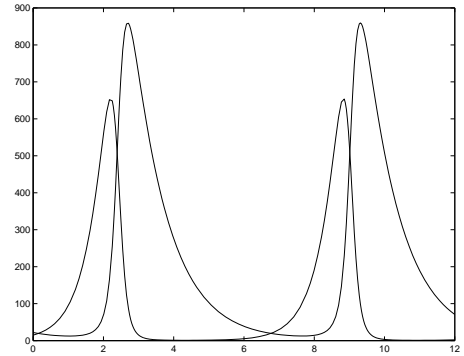
aproximadamente 5 años.

7. Ej. 3(c).

Ejecución:

```
>> [t,Y]=ode45('F2b',[0 12],[15;22]);  
>> plot(t,Y)  
>> I=find(Y(:,1)<1);  
>> J=min(I)  
J =  
    64  
  
>> [t(J),Y(J,1)]  
ans =  
    4.0887    0.9930
```

Salida:



Conclusión: al cabo de 4 años la población de conejos se hace inferior a 1, lo que indica que estos se extinguieron y que, por lo tanto el modelo deja de ser válido.

8. Ej. 4.

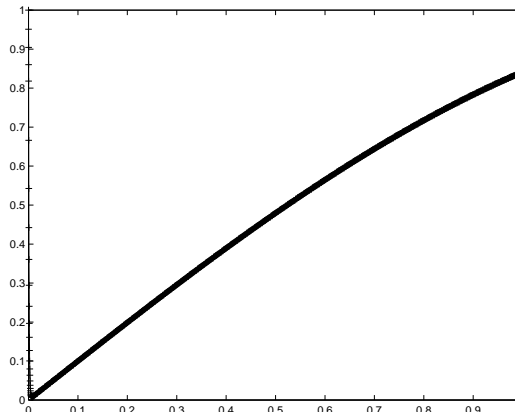
File `f3.m`:

```
function z=f3(t,Y)  
alfa=1000;  
z=-alfa*(y-sin(t))+cos(t);
```

Ejecución (ode45):

```
>> exac=inline('exp(-1000*t)+sin(t)');
>> options=odeset('AbsTol',1.e-3);
>> [t,y]=ode45('f',[0 1],1,options);
>> plot(t,y,'+-')
>> error=norm(exac(t)-y,inf)
error =
    0.0012
>> length(t)
ans =
    1229
```

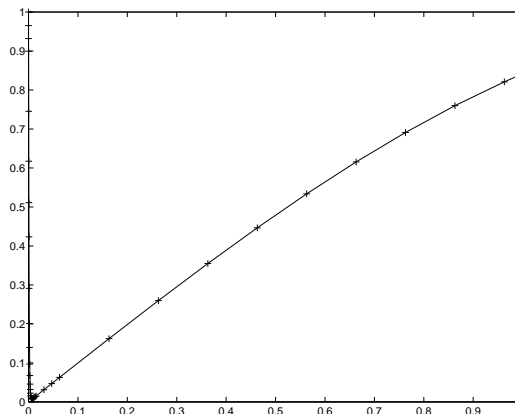
Salida (ode45):



Ejecución (ode15s):

```
>> [t,y]=ode15s('f',[0 1],1,options);
>> plot(t,y,'+-')
>> error=norm(exac(t)-y,inf)
error =
    0.0011
>> length(t)
ans =
    40
```

Salida (ode15s):



9. Ej. 5(a).

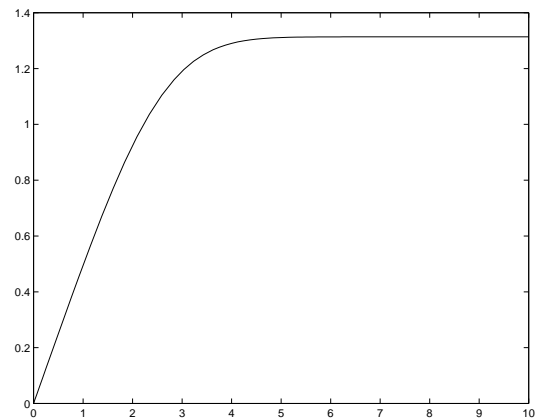
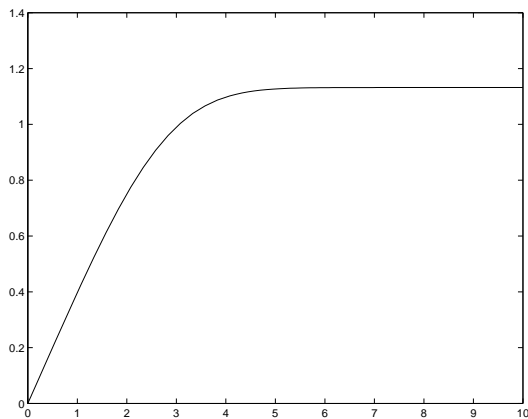
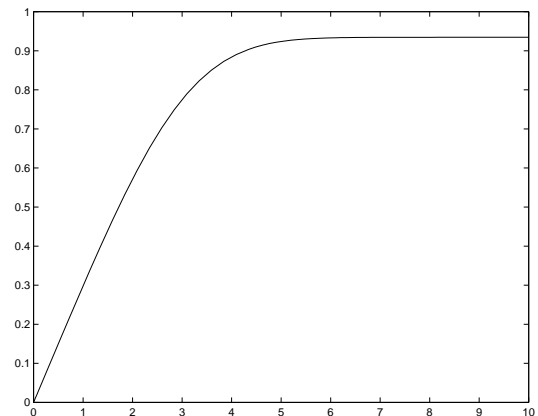
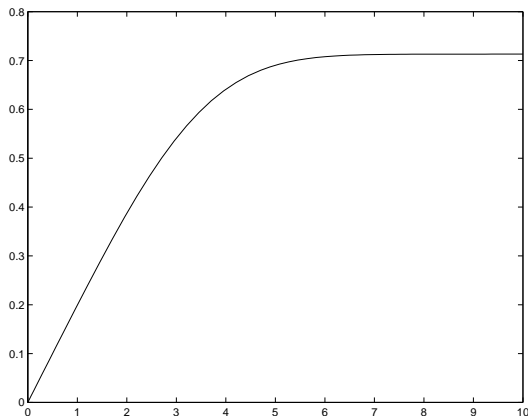
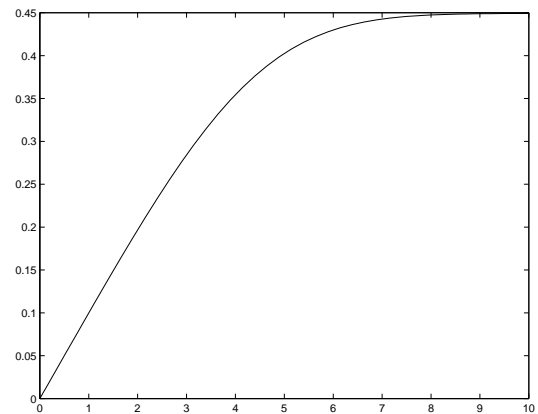
File `F4.m`:

```
function Z=F4(t,Y)
Z=[Y(2);Y(3);-Y(1)*Y(3)/2];
```

Ejecución:

```
>>[t,Y]=ode45('F4',[0 10],[0;0;0.1]);
>>plot(t,Y(:,2))
>>[t,Y]=ode45('F4',[0 10],[0;0;0.2]);
>>plot(t,Y(:,2))
>>[t,Y]=ode45('F4',[0 10],[0;0;0.3]);
>>plot(t,Y(:,2))
>>[t,Y]=ode45('F4',[0 10],[0;0;0.4]);
>>plot(t,Y(:,2))
>>[t,Y]=ode45('F4',[0 10],[0;0;0.5]);
>>plot(t,Y(:,2))
```

Salidas:



En los gráficos se ve que puede tomarse  $\eta = 10$  como “ $\infty$ ”.

10. Ej. 5(b).

File `Blasius.m`:

```
function z=Blasius(x)
[t,Y]=ode45('F',[0 10],[0;0;x]);
z=Y(length(t),2)-1;
```

Ejecución:

```
>> raiz=fzero('Blasius',0.3)
Zero found in the interval: [0.26606, 0.33394].
raiz =
    0.3320
```

11. Ej. 4(c). Si se deriva la expresión del momento se obtiene:

$$\vartheta'(\eta) = f'(\eta)[1 - f'(\eta)].$$

Además  $\vartheta(0) = 0$ . Por lo tanto  $\vartheta$  es solución del P.V.I.

$$\begin{cases} \vartheta' = f'(1 - f'), \\ \vartheta(0) = 0. \end{cases}$$

Se añade esta ecuación a la de Blasius (escrita ya como sistema de E.D.O.) y se resuelve el P.V.I. para el nuevo sistema de 4 E.D.O.

File `F4c.m`:

```
function Z=F4c(t,Y)
Z=[Y(2);Y(3);-Y(1)*Y(3)/2;Y(2)*(1-Y(2))];
```

Ejecución:

```
>> [t,Y]=ode45('F4c',[0 0.5],[0;0;raiz;0]);
>> mom=Y(length(t),4)
mom =
    0.0369
```