Paso adaptativo:

Los métodos de paso adaptativos son especialmente útiles en los problemas donde la solución tiene cambios bruscos. Este tipo de ecuaciones aparecen naturalmente en problemas que involucran circuitos eléctricos o sistemas mecánicos como pueden ser en terremotos o mecánica de estructuras. Vamos a implementar y estudiar métodos de paso adaptativos.

En el siguiente ejemplo usamos el método de Dormand-Prince 5(4) para aproximar la **ecuación de Dahlquist-Bjorck**. El modelo es

$$\left\{\begin{array}{lcl} y'(t) & = & 100\,(sin(t)-y(t)), \quad 0<t\leq 3, \\ y(0) & = & 1 \end{array}\right.$$

y se puede ver como un caso particular del problema

$$\left\{\begin{array}{lcl} y'(t) & = & a\,(sin(t)-y(t)), \quad t>0, \\ y(0) & = & y_0 \end{array}\right.$$

cuya solución es

$$y(t)=e^{-a\,t}y_0+\frac{sin(t)-a^{-1}cos(t)+a^{-1}e^{-a\,t}}{1+a^{-2}}.$$

Para $a >> 1$ el modo transitorio exponencial asociado con $e^{-a\,t}$ decae muy rápido. Aquí la solución se descompone en

$$\text{modo rápido} \sim e^{-a\,t}y_0+\frac{a^{-1}e^{-a\,t}}{1+a^{-2}}.$$

$$\text{modo estacionario o lento} \sim \frac{sin(t)-a^{-1}cos(t)}{1+a^{-2}}.$$

Es conveniente observar las ideas principales del código que sigue a continuación:

1. Se introducen los datos del problema y se inicializan variables

2. Se definen los coeficientes de los métodos Φ y Ψ a utilizar y se establece una tolerancia para empezar a iterar con ellos.

3. Al entrar en el bucle iterativo se calcula el error local de truncatura

4. Si este error es menor que la tolerancia, se usa este valor para avanzar y se amplia h a $hNext = 10h$ (la eleccion $10h$ es arbitraria) teniendo cuidado de no sobrepasar en mucho el valor de tiempo final.

5. En caso contratrio se obtiene un nuevo $h = hNext$ y se procura que cumpla $0.1\,h < hNext < 10h$ al mismo tiempo que no sobrepase en mucho el valor de tiempo final.

```
%
% RK Dorman-Prince 5(4)
% Ecuacion de Dalquist-Bjork
%
%
clear all;
close all;
a=100;
t0=0; %Tiempo inicial
T=3;
Tfin=t0+T;  %Tiempo final
N=1000; % Numero de puntos maximo
t=zeros(1,N+1);% dimensionaliza t
t(1)=t0;

% Vector para Runge-Kutta
y=zeros(1,N+1);% dimensionaliza y
ytrue=zeros(1,N+1);% dimensionaliza ytrue
ytadapt=zeros(1,N+1);% dimensionaliza ytadapt
y0=1.0; %Dato inicial
y(1)=y0;


% Solucion exacta en una particion fina

ttrue=t0:0.001:Tfin; %Particion fina
for i=1:size(ttrue,2)
ytrue(i)=DBsol(ttrue(i),a,y0);
end


% Coeficientes

c2 = 0.2;
c3 = 0.3;
c4 = 0.8;
c5 = 8/9.0;
c6 = 1.0;
c7 = 1.0;

beta1 = 35/384;
beta3 = 500/1113;
beta4 = 125/192;
```

```
beta5 = -2187/6784;
beta6 = 11/84;

b1 = 5179/57600;
b3 = 7571/16695;
b4 = 393/640;
b5 = -92097/339200;
b6 = 187/2100;
b7 = 1/40;

a21 = 0.2;
a31 = 0.075;
a32 = 0.225;
a41 = 44/45;
a42 = -56/15;
a43 = 32/9;
a51 = 19372/6561;
a52 = -25360/2187;
a53 = 64448/6561;
a54 = -212/729;
a61 = 9017/3168;
a62 =-355/33;
a63 = 46732/5247;
a64 = 49/176;
a65 = -5103/18656;
a71 = 35/384;
a73 = 500/1113;
a74 = 125/192;
a75 = -2187/6784;
a76 = 11/84;
tol =1e-3;
n=1;% Para el primer tiempo y primer valor
h=0.5; % Paso inicial
Tau=1; % Truncatura inicial para entrar en el bucle

%while ((n<=N) && (t(n)<Tfin)&&(Tau>0))
  while ((n<=N) && (t(n)<Tfin))
k1 = DBfun(t(n),y(n),a);
k2 = DBfun(t(n)+c2*h,y(n)+h*a21*k1,a);
k3 = DBfun(t(n)+c3*h,y(n)+h*(a31*k1+ a32*k2),a);
k4 = DBfun(t(n)+c4*h,y(n)+h*(a41*k1+ a42*k2 + a43*k3),a);
k5 = DBfun(t(n)+c5*h,y(n)+h*(a51*k1+ a52*k2 + a53*k3 + a54*k4),a);
k6 = DBfun(t(n)+c6*h,y(n)+h*(a61*k1+ a62*k2 + a63*k3 + a64*k4 + a65*k5),a);
```

```
k7 = DBfun(t(n)+c7*h,y(n)+h*(a71*k1+ a73*k3 + a74*k4 + a75*k5 + a76*k6),a);
%
% Incremento para la solucion de orden 5
% dada por y(n) + dy
%
%dy = h*(b1*k1 + b3*k3 + b4*k4 + b5*k5 + b6*k6 +b7*k7);
dy = h*(beta1*k1 + beta3*k3 + beta4*k4 + beta5*k5 + beta6*k6);
%
% Vemos los valores calculados
%
tt=t(n)+h;
ytt=DBsol(tt,a,y0);
 disp(['En tt = ',num2str(tt)]);
 disp(['ytrue = ',num2str(ytt),' yaprox (orden 5)= ',num2str(y(n) + dy)]);
  %
  % Estimamos el error local de truncatura
  % haciendo la diferencia
  % entre la solucion de orden 5 y la de orden 4
  %
  % Error local de truncatura es
  %
Tau = abs((beta1-b1)*k1+(beta3-b3)*k3+(beta4-b4)*k4+(beta5-b5)*k5+(beta6-b6)*k6-b7*k7)
disp(['Tau = ',num2str(Tau)]);
%
% Aceptamos el paso dado y avanzamos
%
if  (Tau <= tol)
disp([' ACEPTAMOS Y AVANZAMOS CON  h= ',num2str(h) ', a tt = ',num2str(t(n)+h)]);
   y(n+1)=  y(n) + dy;
   t(n+1)=  t(n) + h;
   Nfin=n;
   disp([' Tau<tol: t(n+1)= ',num2str(t(n+1))]);
   if (t(n+1)>=Tfin)% Se sobrepasa Tfin
       disp(' t(n+1) sobrepasa Tfin==> ultima iteracion ');
   else % Si no se sobrepasa
       h=min(10*h,Tfin-t(n+1));
   disp([' Tau<tol: min(10*h,Tfin-t(n+1))= ',num2str(h)]);
   end
   n=n+1;
   %pause;
end % Fin de trabajo cuando se acepta el paso
%
% tau>= tol....No aceptamos el paso dado
```

```
%               --> Recalculamos h y volvemos
%               --->a obtener aproximacion para y(t(n)+h)
%
if  (Tau >= tol)
    Nfin=n;
%%
%% hNext =h*s
%% para ser usado si el error local de truncatura no es
%% lo suficientemente pequeno
%%
hNext = 0.9*(tol/Tau)^(1/5.0)*h;
disp([' tol = ',num2str(tol),' Tau = ',num2str(Tau)]);
disp(['h = ',num2str(h),' hNext = ',num2str(hNext)]);
% Controlamos hNext
  if (hNext< 0.1*h) % Que no sea menor que 0.1*h
    hNext = 0.1*h;
  end
  if (hNext > 10*h) % Que no sea mayor que 10*h
      hNext=10*h;
  end
  if (t(n) + hNext > Tfin)% Que no se sobrepase  t0+T
    hNext = Tfin - t(n);

  end
   if(t(n) + hNext < Tfin)
     disp([' RE-CALCULAMOS CON  hNext Ajustado = ',num2str(hNext)]);
   end
  h = hNext; % repetimos el trabajo con h=hNext
 end
%pause;
end
%%
%% Figuras
%%

figure(1);
% subplot(2,1,1)
plot(t(1:Nfin+1),y(1:Nfin+1),'o-',ttrue,ytrue,'r-');
legend('Adapt','exacta','Location','Best');
title(['RKDP5(4) adaptacion con N= ',num2str(Nfin)]);
%
%
% Calculo solucion en la particion obtenida
```

```
%
for i=1:Nfin+1
ytadapt(i)=DBsol(t(i),a,y0);
end
err=abs(y(1:Nfin+1)-ytadapt(1:Nfin+1));
errmax=max(abs(y(1:Nfin+1)-ytadapt(1:Nfin+1)));
disp([' Error global  = ',num2str(errmax),' tol  = ',num2str(tol)]);
disp([' Nfin  = ',num2str(Nfin)]);

%
% Calculo con RK4 clasico de orden 4 con el mismo numero de puntos
% particion uniforme
%
    h=T/Nfin;% talla de la particion
    tRK4=zeros(1,Nfin+1);% dimensionaliza t
    %tRK4(1)=t0;
    tRK4=t0:h:t0+T;
% Vector para Runge-Kutta orden 4
    yRK4=zeros(1,Nfin+1);% dimensionaliza y
    yexactRK4=zeros(1,Nfin+1);% dimensionaliza ytadaptRK4
    yRK4(1)=y0;
    yexactRK4(1)=y0;
for n=1:Nfin
    %Calculo RK4
    k1=DBfun(tRK4(n),yRK4(n),a);
    k2=DBfun(tRK4(n)+h/2,yRK4(n)+h/2*k1,a);
    k3=DBfun(tRK4(n)+h/2,yRK4(n)+h/2*k2,a);
    k4=DBfun(tRK4(n)+h,yRK4(n)+h*k3,a);
    yRK4(n+1)=yRK4(n)+h*(k1+2*k2+2*k3+k4)/6;
    %tRK4(n+1)=tRK4(n)+h;
    yexactRK4(n+1)=DBsol(tRK4(n+1),a,y0);
end
figure(2);
plot(tRK4,yRK4,'x-',ttrue,ytrue,'r-');
legend('RK4','exacta','Location','Best');
title([' RK clasico orden 4 con N= ',num2str(Nfin)]);
% plot(t(1:Nfin+1),y(1:Nfin+1),'o-',tRK4,yRK4,'x-',ttrue,ytrue,'r-');
% legend('Adapt','RK4','exacta','Location','Best');
% title([' Adaptacion vs RK4 clasico con N= ',num2str(Nfin)]);
%
% Valores exactos en la particion para RK4
%
errRK4=abs(yRK4(1:Nfin+1)-yexactRK4(1:Nfin+1));
```

```
figure(3);
subplot(2,1,1)
semilogy(t(1:Nfin+1),err(1:Nfin+1),'o-');
legend('Adapt','Location','Best');
title([' Errores Adaptativo con N= ',num2str(Nfin)]);
subplot(2,1,2)
semilogy(tRK4,errRK4,'d-');
legend('RK4','Location','Best');
title([' Errores RK4 clasico con N= ',num2str(Nfin)]);
```

El archivo **DBfun.m** es:

```
function dydt=DBfun(tt,yy,a)
dydt=a*(sin(tt)-yy);
end
```

y el archivo **DBsol.m** es:

```
function ytt=DBsol(tt,a,y0)
aa2=a*a;
aux=1/(1+1/aa2);
inva=1.0/a;
ytt=y0*exp(-a*tt)+(sin(tt)-cos(tt)*inva+exp(-a*tt)*inva)*aux;
end
```

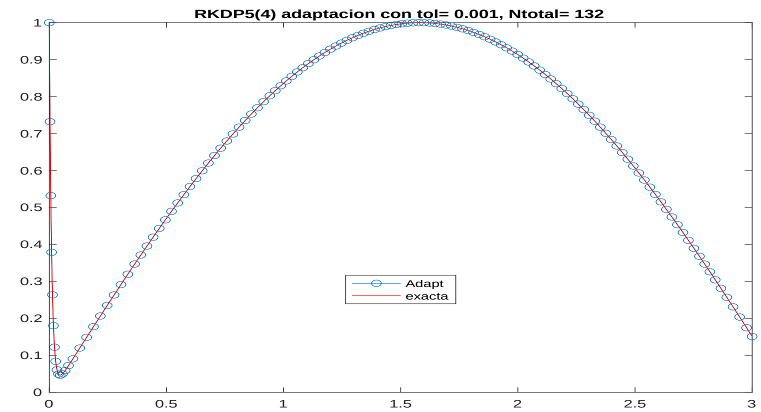

Figura 1: Dormand-Prince 5(4) para ecuación de Dahlquist-Bjorck, $a = 100, tol = 1e-3$

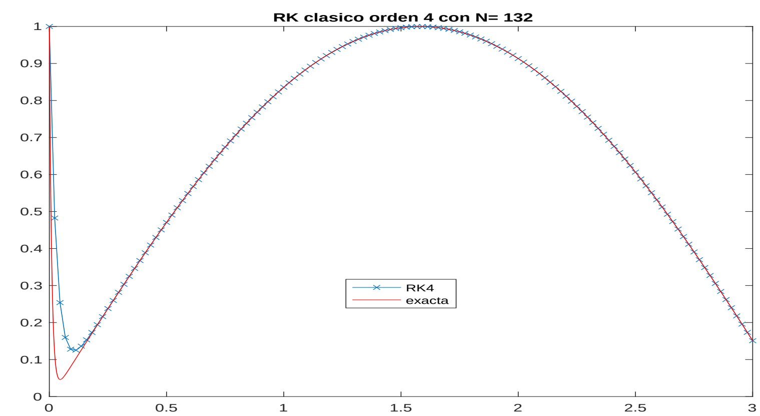

Figura 2: Runge-Kutta clásico para ecuación de Dahlquist-Bjorck, $a = 100$.

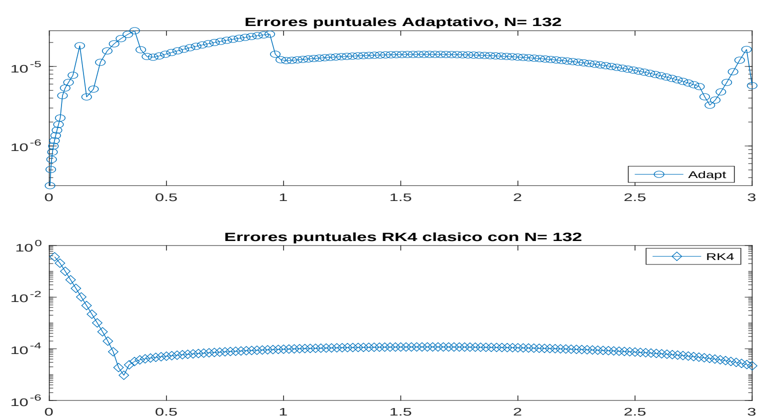

Figura 3: Errores con ambos métodos

Ejercicio: Adaptar el código anterior para usar Runge-Kutta Fehlberg 4(5) en vez de Dormand-Prince 5(4) en cualquiera de estos problemas y así comparar con Runge-Kutta clásico de orden 4 en cada ejemplo.

1. **Comparación en un problema no lineal:** Al encender una cerilla la bola de fuego crece hasta alcanzar un estado estacionario donde se equilibra al absorción de oxígeno del exterior con el consumo interior. Un modelo simple es el propuesto por Larry Shampine (autor de las librerias de edos para MATLAB, OCTAVE y SCILAB entre otros)

$$\left\{\begin{array}{rcl} y'(t) &=& y(t)^2 - y(t)^3, \quad 0 < t < 2/\delta, \\ y(0) &=& \delta \end{array}\right.$$

   en donde $y(t)$ es el radio de la bola de fuego, $\delta > 0$ es un radio inicial pequeño y los términos $y(t)^2$ e $y(t)^3$ están relacionados con la superficie y el volumen de la bola de fuego.

   El parámetro crítico es el radio inicial $\delta > 0$ que es pequeño y el fenómeno físico de interés ocurre en el tiempo $t_\star \approx 1/\delta$. Para $0 < t < 1/\delta$ se observa un crecimiento moderado del radio y un crecimiento repentino en torno a $t_\star \approx 1/\delta$ para llegar al valor $y(t) \approx 1$ en donde se estabiliza. Usando $\delta = 10^{-3}$

   a) usar un método adaptativo RK-4(5) para resolver este problema con una tolerancia de 0.0001. Contar el número de pasos usados.

   b) Resolver el mismo problema con el método de RK clásico con un paso escogido tal que el número de puntos sea similar al del apartado anterior.

2. El problema

$$\left\{\begin{array}{rcl} y'(t) &=& 51150e^{-50t}y(t)^2, \quad t \in [0,3], \\ y(0) &=& 1/1024 \end{array}\right.$$

   cuya solución $y(t) = (1 + 1023e^{-50t})^{-1}$ (aquí $51150 = 1023 * 50$) presenta un incremento brusco de $y(0)$ a 1.

3. **Comparación en un modelo muy rígido:** Consideremos el problema

$$\left\{\begin{array}{rcl} y'(t) &=& 101 + 100(t - y), \quad 0 < t \leq 1, \\ y(0) &=& 1 \end{array}\right.$$

   La solución exacta $y(t) = 1 + t$ viene de la solución general $y(t) = 1 + t + ce^{-100t}$ que tiene un término que decae muy rápidamente.