Métodos Numéricos y Simulaciones en Astrofísica

Parte 7: Métodos de Paso Múltiple, Implícitos y Predictor - Corrector

Métodos Implícitos

Cómo encontrar el valor de y_{n+1}?

Método de Newton, de la secante o iteración

EXAMPLE 6.10

Consider the IVP

$$y' = -y \ln y, \quad y(0) = y_0. \tag{6.28}$$

Applying the trapezoid method (6.25) to this yields the computation

$$y_{n+1} = y_n - \frac{1}{2}h[y_{n+1} \ln y_{n+1} + y_n \ln y_n], \tag{6.29}$$

where we now view y_{n+1} as the (unknown) root of the function

$$F(y) = y - y_n + \frac{1}{2}h[y \ln y + y_n \ln y_n].$$

Alternatively, we can view (6.29) as defining y_{n+1} as the fixed point of the function

$$g(y) = y_n - \frac{1}{2}h[y \ln y + y_n \ln y_n].$$

Métodos Implícitos

Regardless of which view we take, we can apply the methods of Chapter 3 to solve for y_{n+1}. Denote the individual iterates as $y_{n+1,k}$, where k is the iteration counter.[4] Then Newton's method applied to $F(y)$ yields the iteration

$$y_{n+1,k+1} = y_{n+1,k} - \left(\frac{y_{n+1,k} - y_n - \frac{1}{2}h[f(t_{n+1},y_{n+1,k}) + f(t_n,y_n)]}{1 - \frac{1}{2}hf_y(t_{n+1},y_{n+1,k})} \right),$$

where

$$f_y(t,y) = \frac{\partial f}{\partial y}(t,y);$$

the secant method yields the iteration

$$y_{n+1,k+1} = y_{n+1,k} - \left(y_{n+1,k} - y_n - \frac{1}{2}h[f(t_{n+1},y_{n+1,k}) + f(t_n,y_n)] \right) \times \left(\frac{y_{n+1,k} - y_{n+1,k-1}}{y_{n+1,k} - \frac{1}{2}hf(t_{n+1},y_{n+1,k}) - y_{n+1,k-1} + \frac{1}{2}hf(t_{n+1},y_{n+1,k-1})} \right);$$

and a simple fixed-point iteration yields the iteration

$$y_{n+1,k+1} = y_n - \frac{1}{2}h[y_{n+1,k} \ln y_{n+1,k} + y_n \ln y_n]. \tag{6.30}$$

Métodos Predictor - Corrector

Existe otra forma de encontrar el valor de y_{n+1}?

Consider the trapezoid method, defined by (6.25); instead of actually solving for the exact value of y_{n+1} that satisfies this equation, we instead apply a simpler method to estimate y_{n+1} and then use this estimated value on the right side of (6.25). For the trapezoid method, it is common to use Euler's method as the estimator, or *predictor*:

$$\bar{y}_{n+1} = y_n + hf(t_n, y_n). \tag{6.31}$$

Then the *corrector* step is accomplished by

$$y_{n+1} = y_n + \frac{1}{2}h\left[f(t_{n+1}, \bar{y}_{n+1}) + f(t_n, y_n)\right]. \tag{6.32}$$

Métodos Predictor - Corrector

Ejemplo:

Consider the IVP

$$y' = -y \ln y, \quad y(0) = \frac{1}{2}.$$

We will use the trapezoid rule predictor-corrector method to approximate the solution at $t = 1$, using the rather crude value $h = 0.25$. We have the following sequence of computations:

$$\begin{aligned}
\bar{y} &= y_0 + hf(t_0, y_0) = 0.5 - 0.25 \times 0.5 \ln 0.5 = 0.5866433976 \\
y_1 &= y_0 + \frac{1}{2}h[f(t_0, y_0) + f(t_1, \bar{y})] = 0.58243161136465 \\
\bar{y} &= y_1 + hf(t_1, y_1) = 0.66113901764113 \\
y_2 &= y_1 + \frac{1}{2}h[f(t_1, y_1) + f(t_2, \bar{y})] = 0.65598199856663
\end{aligned}$$

$$\begin{aligned}
\bar{y} &= y_2 + hf(t_2, y_2) = 0.72512609790319 \\
y_3 &= y_2 + \frac{1}{2}h[f(t_2, y_2) + f(t_3, \bar{y})] = 0.71968686944048
\end{aligned}$$

$$\begin{aligned}
\bar{y} &= y_3 + hf(t_3, y_3) = 0.77887015094459 \\
y_4 &= y_3 + \frac{1}{2}h[f(t_3, y_3) + f(t_4, \bar{y})] = 0.77360953103925.
\end{aligned}$$

■

Métodos Predictor - Corrector

Ejemplo:

To illustrate the difference between actually solving the nonlinear equation and using a predictor-corrector method, we continue with the same IVP (6.28) as used in the previous example. The exact solution here is $y = e^{(-\ln 2)e^{-t}}$. Table 6.2 shows the results of approximating the solution to this equation using the predictor-corrector method (6.31)–(6.32) (second column) and actually solving equation (6.29) (using a fixed-point iteration) for the exact value of y_{n+1} (fourth column). Clearly, the two methods are not producing wildly different approximations. Note also that the calculation based on exactly solving the recursion is more accurate; in fact, the error in the iterated approximation (fifth column) starts out at about 61% of the error for the predictor-corrector approximation, and ends up at around 9%. ∎

Métodos Predictor - Corrector

TABLE 6.2 Implicit Method Applied to $y' = -y \ln y$

	Predictor-corrector		Fixed-point solution	
n	y_n	*Error*	y_n	*Error*
1	0.521438685E+00	0.629350068E–05	0.521441082E+00	0.389659700E–05
2	0.542415401E+00	0.124265310E–04	0.542420677E+00	0.715031332E–05
3	0.562889543E+00	0.183888337E–04	0.562898148E+00	0.978351030E–05
4	0.582826911E+00	0.241665678E–04	0.582839249E+00	0.118283433E–04
5	0.602199375E+00	0.297435050E–04	0.602215794E+00	0.133243757E–04
6	0.620984504E+00	0.351021428E–04	0.621005290E+00	0.143164168E–04
7	0.639165163E+00	0.402246989E–04	0.639190535E+00	0.148526117E–04
8	0.656729097E+00	0.450939634E–04	0.656759208E+00	0.149827925E–04
9	0.673668505E+00	0.496939995E–04	0.673703441E+00	0.147570906E–04
10	0.689979615E+00	0.540106915E–04	0.690019401E+00	0.142247986E–04
11	0.705662271E+00	0.580321478E–04	0.705706870E+00	0.134334649E–04
12	0.720719531E+00	0.617489703E–04	0.720768874E+00	0.124059468E–04
13	0.735157278E+00	0.651544045E–04	0.735211228E+00	0.112040544E–04
14	0.748983859E+00	0.682443879E–04	0.749042237E+00	0.986672690E–05
15	0.762209745E+00	0.710175127E–04	0.762272333E+00	0.842946303E–05
16	0.774847210E+00	0.734749197E–04	0.774913760E+00	0.692422919E–05

Métodos Predictor - Corrector

En general, a menos que la ecuación diferencial sea muy sensible a cambios en los datos un método predictor – corrector es tan bueno como el proceso de encontrar los valores exactos de y_{n+1} mediante un método de recursión.

Métodos Predictor - Corrector

Si la ecuación diferencial es lineal es posible evitar el método implícito:

$$y' = a(t)y + b(t), \quad y(t_0) = y_0,$$

where a and b are known functions of t only. If we apply the trapezoid method to this equation, we initially have the implicit recursion

$$y_{n+1} = y_n + \frac{1}{2}h\left[a(t_n)y_n + b(t_n) + a(t_{n+1})y_{n+1} + b(t_{n+1})\right].$$

However, we can make this explicit by using the linearity of the equation to solve for y_{n+1}:

$$y_{n+1} = \left(\frac{1 + \frac{1}{2}ha(t_n)}{1 - \frac{1}{2}ha(t_{n+1})}\right) y_n + \frac{1}{2}h\frac{b(t_n) + b(t_{n+1})}{1 - \frac{1}{2}ha(t_{n+1})}. \tag{6.33}$$

Métodos de Paso Múltiple

Cómo encontrar los valores iniciales?

EXAMPLE 6.15

We take as our initial value problem the same one we looked at in the beginning of the previous section,

$$y' = -y \ln y, \quad y(0) = \frac{1}{2}. \tag{6.36}$$

We want to apply, say, the midpoint method,

$$y_{n+1} = y_{n-1} + 2hf(t_n, y_n),$$

with $h = \frac{1}{4}$ to this problem. We will first use Euler's method to provide the estimate of $y(t_1)$ that is necessary to proceed with the computation. We get

$$y_1 = y_0 + hf(t_0, y_0) = \frac{1}{2} + \frac{1}{4}\left(-\frac{1}{2}\ln\frac{1}{2}\right) = 0.5866433976.$$

Métodos de Paso Múltiple

We can now compute using the midpoint method:

$$y_2 = y_0 + 2hf(t_1, y_1) = \frac{1}{2} + 2 \times \frac{1}{4}(-0.5866433976 \ln 0.5866433976) = 0.6564396503$$

$$y_3 = y_1 + 2hf(t_2, y_2) = 0.7247991686,$$

and so forth.

Suppose now that we want to use the trapezoid rule predictor-corrector method (6.31)–(6.32) to provide the estimate of $y(t_1)$. Again using $h = \frac{1}{4}$, we compute

$$\bar{y} = y_0 + hf(t_0, y_0) = 0.5 - 0.25 \times 0.5 \ln 0.5 = 0.5866433976$$

$$y_1 = y_0 + \frac{1}{2}h[f(t_0, y_0) + f(t_1, \bar{y})] = 0.5824316114,$$

so that

$$y_2 = y_0 + 2hf(t_1, y_1) = \frac{1}{2} + 2 \times \frac{1}{4}(-0.5824316114 \ln 0.5824316114) = 0.6574148126$$

$$y_3 = y_1 + 2hf(t_2, y_2) = 0.7203046741,$$

etc. Regardless of which method is used to produce y_1, we can now compute y_k values using only the midpoint method.

Métodos de Paso Múltiple

Now, by taking $h = 1/16$ and computing out with $t = 1$, we get the results shown in Table 6.4. Compare this with Table 6.2. Note that, as expected, we are getting better accuracy when we use the more accurate starting value, although it should be noted that the discrepancy is not large.

This example prompts an interesting question. Note that we used Euler's method, which is only $O(h)$, to generate the starting values for the midpoint method, which is $O(h^2)$. Is the use of a lower-order method for the starting values going to affect the accuracy of the overall computation?

A complete explanation is beyond the scope of this text, but it can be shown that a $(p-1)$-order method can be used to generate the starting values for a p-order method without affecting the overall order of convergence. It would of course be more accurate to use a p-order method than to use a $(p-1)$-order method for the starting values, but the net order of accuracy of the method is not affected. ■

Métodos de Paso Múltiple

TABLE 6.4 Midpoint Method Applied to $y' = -y \ln y$, $h = 1/16$

	y_1 computed by predictor-corrector		y_1 computed by Euler's method	
n	y_n	Error	y_n	Error
1	0.521438685E+00	0.629350068E–05	0.521660849E+00	-0.215870648E–03
2	0.542442735E+00	-0.149082970E–04	0.542433042E+00	-0.521501377E–05
3	0.562913366E+00	-0.543392852E–05	0.563136000E+00	-0.228068587E–03
4	0.582876067E+00	-0.249892345E–04	0.582854530E+00	-0.345267290E–05
5	0.602241522E+00	-0.124030017E–04	0.602465395E+00	-0.236276553E–03
6	0.621050404E+00	-0.307978504E–04	0.621015069E+00	0.453741300E–05
7	0.639220651E+00	-0.152631109E–04	0.639446837E+00	-0.241449481E–03
8	0.656807255E+00	-0.330637366E–04	0.656756294E+00	0.178973669E–04
9	0.673732972E+00	-0.147732929E–04	0.673962850E+00	-0.244651773E–03
10	0.690066203E+00	-0.325779206E–04	0.689997851E+00	0.357749024E–04
11	0.705732009E+00	-0.117057153E–04	0.705967262E+00	-0.246958289E–03
12	0.720811392E+00	-0.301122659E–04	0.720723877E+00	0.574032598E–04
13	0.735229212E+00	-0.678027590E–05	0.735471822E+00	-0.249390296E–03
14	0.749078472E+00	-0.263680981E–04	0.748969952E+00	0.821511474E–04
15	0.762281388E+00	-0.625682149E–06	0.762533643E+00	-0.252880568E–03
16	0.774942633E+00	-0.219485394E–04	0.774811136E+00	0.109548790E–03

Métodos de Paso Múltiple

$$y_{n+1} = y_{n-1} + 2hf(t_n, y_n).$$

Let's consider the midpoint method (6.20) as applied to the very simple differential equation

$$y' = -y, \quad y(0) = 1, \tag{6.37}$$

which has exact solution $y(t) = e^{-t}$. To minimize the effects of any error in the starting values, let's use

$$y_1 = y(t_1) = e^{-h}, \tag{6.38}$$

where h is the mesh spacing. Note that this is the *exact* value of $y(t_1)$; i.e., there is no error in using this starting value. Figure 6.4 shows the results of applying (6.20) to (6.37) using (6.38) as the starting value, for a sequence of mesh values $h^{-1} = 4, 8, 16, \ldots, 128$.

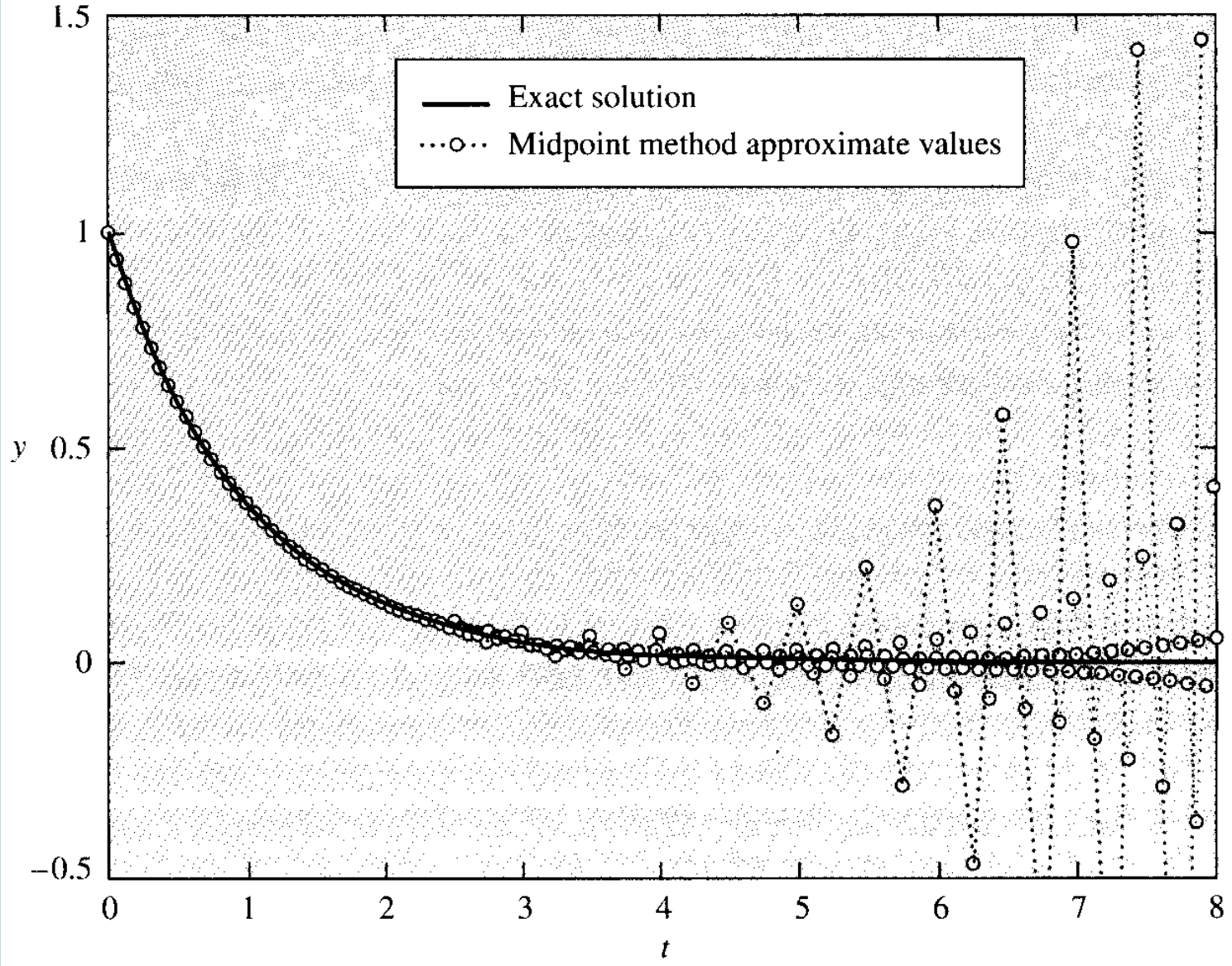

FIGURE 6.4 Illustration of weak stability, $h = 4^{-1}, 8^{-1}, 16^{-1}$.

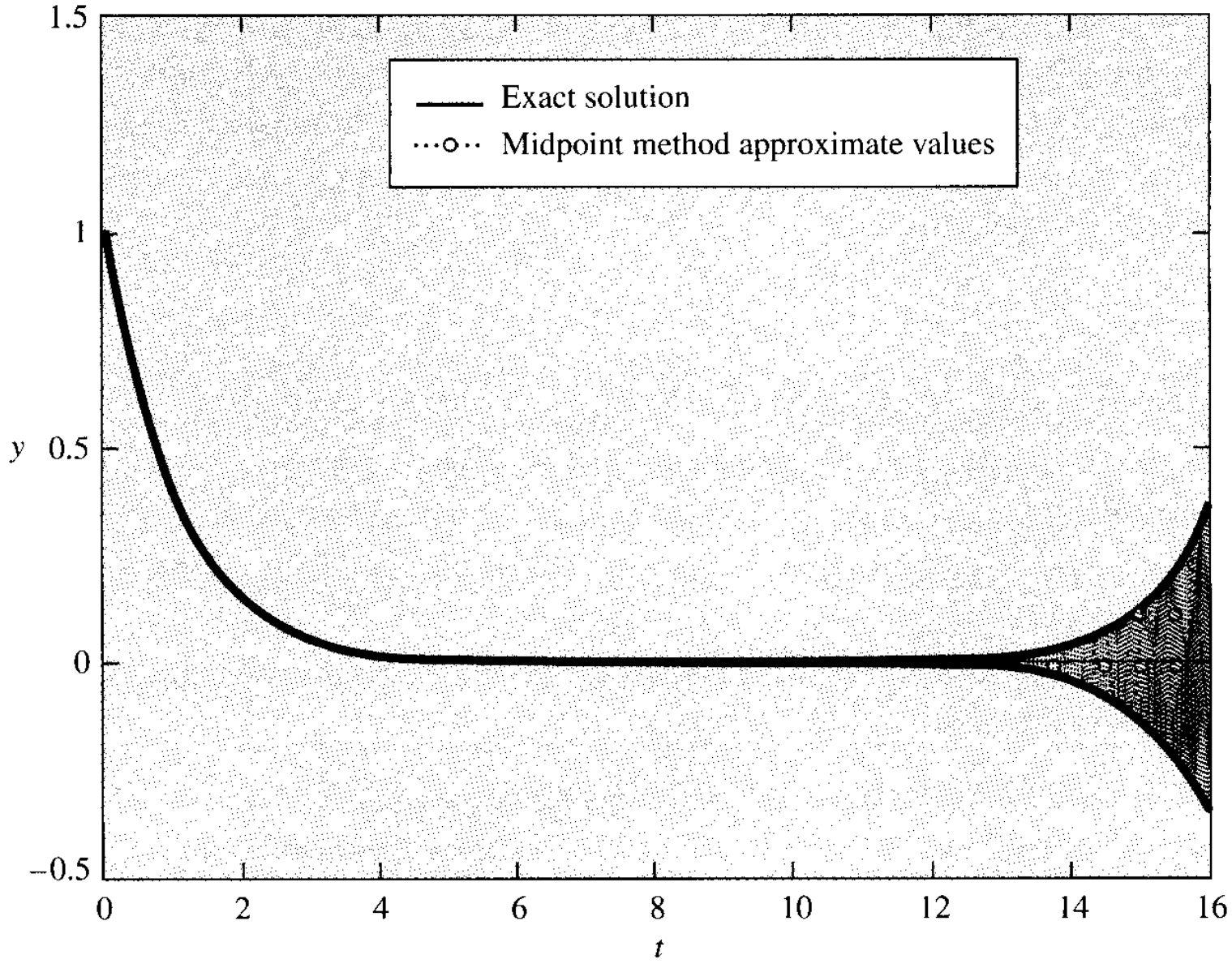

FIGURE 6.5 Illustration of weak stability, $h = 128^{-1}$.

Métodos de Runge-Kutta

Los métodos de Runge-Kutta son muy populares para resolver problemas de valor inicial y poseen gran exactitud.

Recall the usual predictor-corrector formulation of the trapezoid method:

$$\begin{aligned} \bar{y}_{n+1} &= y_n + hf(t_n, y_n) \\ y_{n+1} &= y_n + \frac{1}{2}h\left[f(t_{n+1}, \bar{y}_{n+1}) + f(t_n, y_n)\right]. \end{aligned}$$

We can directly substitute the predictor into the corrector to write this as a single recursion:

$$y_{n+1} = y_n + \frac{1}{2}h\left[f(t_{n+1}, y_n + hf(t_n, y_n)) + f(t_n, y_n)\right]. \qquad (6.44)$$

Métodos de Runge-Kutta

$$y_{n+1} = y_n + c_1 h f(t_n, y_n) + c_2 h f(t_n + \alpha h, y_n + \beta h f(t_n, y_n)), \tag{6.45}$$

where c_1, c_2, α, and β are parameters as yet undetermined. We want to choose these so that the approximate solution defined by (6.45) is as accurate as possible; i.e., we want to make the truncation error as small as possible, in terms of powers of h. Thus we look at the (admittedly rather imposing) expression

$$R = y(t+h) - y(t) - c_1 h f(t, y(t)) - c_2 h f(t + \alpha h, y(t) + \beta h f(t, y(t))).$$

To reduce this so we can infer the correct values of c_1, c_2, α, and β that will yield the smallest residual R, we will need to use Taylor's Theorem in two variables, which we state here, without proof:

$$F(x+h, y+\eta) = F(x,y) + hF_x(x,y) + \eta F_y(x,y) + \frac{1}{2}(h^2 F_{xx}(x,y) + h\eta F_{xy}(x,y) + \eta^2 F_{yy}(x,y)) + O(h^3 + \eta^3).$$

We can thus expand the last term in R as

$$\begin{aligned} f(t + \alpha h, y(t) + \beta h f(t, y(t))) = {} & f(t, y(t)) + \alpha h f_t(t, y(t)) + \beta h f(t, y(t)) f_y(t, y(t)) \\ & + \frac{1}{2}(\alpha^2 h^2 f_{tt}(t, y(t)) + \alpha\beta h^2 f(t, y(t)) f_{ty}(t, y(t)) \\ & + \beta^2 h^2 [f(t, y(t))]^2 f_{yy}(t, y(t))) \quad + O(h^3). \end{aligned}$$

Métodos de Runge-Kutta

We can also expand $y(t+h)$ in terms of $y(t)$ as follows:

$$y(t+h) = y(t) + hy'(t) + \frac{1}{2}h^2 y''(t) + O(h^3).$$

But the differential equation implies that $y' = f(t,y(t))$ and therefore, in addition,

$$\begin{aligned} y''(t) = \frac{d}{dt} f(t,y(t)) &= f_t(t,y(t)) + f_y(t,y(t))y'(t) \\ &= f_t(t,y(t)) + f_y(t,y(t))f(t,y(t)). \end{aligned}$$

Now let us substitute both of these expansions into our expression for R, where we get

$$\begin{aligned} R = &\left(y + hf + \frac{1}{2}h^2(f_t + f_y f) + O(h^3) \right) - y - c_1 hf \\ &- c_2 h \left(f + \alpha h f_t + \beta h f f_y + \frac{1}{2}(\alpha^2 h^2 f_{tt} + \alpha\beta h^2 f f_{ty} + \beta^2 h^2 f^2 f_{yy}) + O(h^3) \right). \end{aligned}$$

Métodos de Runge-Kutta

Si reescribimos la fórmula para R:

$$
\begin{aligned}
R &= h(f - c_1 f - c_2 f) \\
&\quad + h^2 \left(\frac{1}{2}(f_t + f_y f) - c_2 \alpha f_t - c_2 \beta f f_y \right) + O(h^3) \\
&= h(f - c_1 f - c_2 f) \\
&\quad + h^2 \left[\left(\frac{1}{2} - c_2 \alpha \right) f_t + \left(\frac{1}{2} - c_1 \alpha \right) f f_y \right] + O(h^3),
\end{aligned}
$$

from which we can extract the equations

$$
1 - c_1 - c_2 = 0, \quad \frac{1}{2} - c_2 \alpha = 0, \quad \frac{1}{2} - c_2 \beta = 0.
$$

Métodos de Runge-Kutta

If these three equations are satisfied, then the residual satisfies $R = O(h^3)$; thus the truncation error is $O(h^2)$. Since this is a set of three equations in four unknowns, there is more than one solution. Typically, c_1 is regarded as arbitrary, and we then have

$$c_2 = 1 - c_1 \tag{6.46}$$

$$\alpha = \beta = \frac{1}{2c_2}. \tag{6.47}$$

Thus some possible solutions include

Solución 1 $$c_1 = c_2 = \frac{1}{2}, \quad \alpha = \beta = 1;$$

this yields the trapezoid rule predictor-corrector method:

$$y_{n+1} = y_n + \frac{1}{2}h\left[f(t_{n+1}, y_n + hf(t_n, y_n)) + f(t_n, y_n)\right].$$

Or

Solución 2 $$c_1 = 0, \quad c_2 = 1, \quad \alpha = \beta = \frac{1}{2};$$

this yields the midpoint rule predictor-corrector (6.34)–(6.35):

$$y_{n+1} = y_n + hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hf(t_n, y_n)\right). \tag{6.48}$$

Métodos de Runge-Kutta

We could also try

Solución 3

$$c_1 = \frac{1}{4}, \quad c_2 = \frac{3}{4}, \quad \alpha = \beta = \frac{2}{3},$$

which yields a method sometimes called the method of Heun:[6]

$$y_{n+1} = y_n + \frac{1}{4}h\left[f(t_n, y_n) + 3f\left(t_n + \frac{2}{3}h, y_n + \frac{2}{3}hf(t_n, y_n)\right)\right]. \tag{6.49}$$

All three of these have second-order truncation errors, and it can be shown that all of them are second-order accurate in the sense that, for any $T > 0$,

$$\max_{t_k \le T} |y(t_k) - y_k| \le C_T h^2,$$

where C_T will depend on T but not on h.

Método de Runge-Kutta (cuarto orden)

The most commonly used Runge-Kutta method is the fourth-order method, derived in a manner very similar to what we did here, but using four slope values instead of only two. The required manipulations are necessarily much more involved, and so we skip them. The method is usually written as follows:

$$k_1 = hf(t_n, y_n) \tag{6.50}$$

$$k_2 = hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right) \tag{6.51}$$

$$k_3 = hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right) \tag{6.52}$$

$$k_4 = hf\left(t_n + h, y_n + k_3\right) \tag{6.53}$$

$$y_{n+1} = y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right). \tag{6.54}$$

Método de Runge-Kutta

Ejemplo:

For illustration, let's consider our usual example IVP

$$y' = -y\ln y, \quad y(0) = \frac{1}{2},$$

which we will first solve with a very coarse grid, using both the second-order method (6.49) and the fourth-order method (6.50)–(6.54). We then will show the results of solving the same equation using the same methods, but with finer grids. Let $h = 1/2$; to compute out to $t = 1$, then, requires two steps with each method. For the method of Heun, we compute as follows:

1:

$$\bar{f} = f(t_0,y_0) = f(0,1/2) = -(1/2)\ln(1/2) = 0.3465735903$$

$$\bar{y} = y_0 + (2/3)h\bar{f} = 0.5 + (1/3)\bar{f} = 0.6155245301$$

$$y_1 = y_0 + \frac{1/2}{4}\left[\bar{f} + 3f(t_0 + (2/3)h, \bar{y})\right]$$

$$= 0.5 + 0.125\left[0.3465735903 + 3(0.2987020395)\right] = 0.6553349636$$

2:

$$\bar{f} = f(t_1,y_1) = f(0.5,0.6553349636) = 0.276950309$$

$$\bar{y} = y_1 + (2/3)h\bar{f} = 0.6553349636 + (1/3)(0.276950309) = 0.7476517333$$

$$y_2 = y_1 + \frac{1/2}{4}\left[\bar{f} + f(t_1 + (2/3)h, \bar{y})\right]$$

$$= 0.6553349636 + 0.125\left[0.276950309 + 3(0.2174305868)\right] = 0.7714902223$$

Método de Runge-Kutta

For fourth-order Runge-Kutta (RK4), the computation is longer, of course:

1:

$$k_1 = hf(t_0, y_0) = 0.17328679513999$$
$$z_1 = y_0 + (1/2)k_1 = 0.58664339756999$$
$$k_2 = hf(t_0 + (1/2)h, z_1) = 0.15643965031862$$
$$z_2 = y_0 + (1/2)k_2 = 0.57821982515931$$
$$k_3 = hf(t_0 + (1/2)h, z_2) = 0.15837474613445$$
$$z_3 = y_0 + k_3 = 0.65837474613445$$
$$k_4 = hf(t_0 + h, z_3) = 0.13759406302779$$
$$y_1 = y_0 + (1/6)(k_1 + 2k_2 + 2k_3 + k_4) = 0.65675160851232$$

2:

$$k_1 = hf(t_1, y_1) = 0.13806541027436$$
$$z_1 = y_1 + (1/2)k_1 = 0.72578431364950$$
$$k_2 = hf(t_1 + (1/2)h, z_1) = 0.11630780609087$$
$$z_2 = y_1 + (1/2)k_2 = 0.71490551155775$$
$$k_3 = hf(t_1 + (1/2)h, z_2) = 0.11996289517416$$
$$z_3 = y_1 + k_3 = 0.77671450368648$$
$$k_4 = hf(t_1 + h, z_3) = 9.8131054204024\text{E}-02$$
$$y_2 = y_1 + (1/6)(k_1 + 2k_2 + 2k_3 + k_4) = 0.77487458634706$$

TABLE 6.5 Runge-Kutta Examples, $y' = -y \ln y$

$n = h^{-1}$	RK2 error	RK4 error
4	0.767315E–03	0.269490E–05
8	0.180964E–03	0.162549E–06
16	0.439226E–04	0.997759E–08
32	0.108187E–04	0.617969E–09
64	0.268460E–05	0.384481E–10
128	0.668655E–06	0.239708E–11
256	0.166852E–06	0.149658E–12
512	0.416742E–07	0.976996E–14

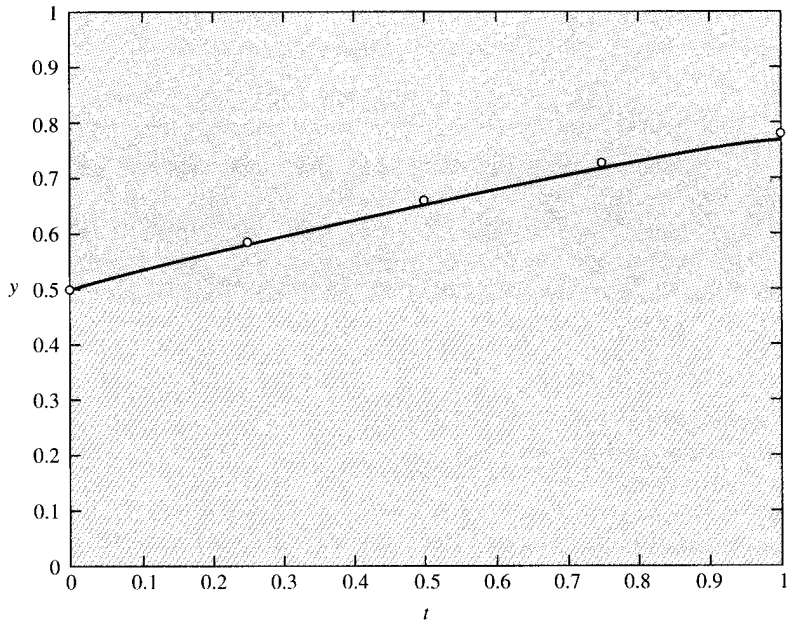

FIGURE 6.6 Solution of $y' = -y \ln y$, $y(0) = \frac{1}{2}$, along with second-order Runge-Kutta approximate values for $h = \frac{1}{4}$.

Uno de los problemas más serios del método de Runge-Kutta es que requiere más evaluaciones de la función que otros métodos.

Métodos de Pasos Múltiples

El grupo de métodos de Adams:

- Métodos Explícitos: Adams-Bashforth
- Métodos Implícitos: Adams-Moulton

Método de Adams-Bashforth

Recall the differential equation $y'(t) = f(t,y(t)), \quad y(t_0) = y_0.$
Starting at any value $t_n \geq t_0$, we can solve this, formally, by simple integration to get the value at the next step:

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(s,y(s))\, ds.$$

Suppose now that we have values $y(t_{n-k})$, $k = 0,1,2,\ldots,p$, of the exact solution. We can construct a polynomial of degree p that interpolates to $F(s) = f(s,y(s))$ at these points:

$$q_p(s) = \sum_{k=0}^{p} L_k(s) f(t_{n-k},y(t_{n-k})),$$

where the L_k are essentially the Lagrange functions from Section 4.1 (although the notation and indexing are slightly different):

$$L_k(s) = \prod_{\substack{i=0 \\ i\neq k}}^{p} \frac{s - t_{n-i}}{t_{n-k} - t_{n-i}}. \tag{6.55}$$

We then have that

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} \sum_{k=0}^{p} L_k(s) f(t_{n-k},y(t_{n-k}))ds + R_p(t_{n+1}), \tag{6.56}$$

where (recall that $f(t,y(t)) = y'(t)$)

$$R_p(t_{n+1}) = \int_{t_n}^{t_{n+1}} \frac{1}{(p+1)!}(t - t_n)(t - t_{n-1})\ldots(t - t_{n-p}) y^{(p+2)}(\theta_{h,t})\, dt.$$

Método de Adams-Bashforth

Since the polynomial part of the integrand does not change sign on $[t_n, t_{n+1}]$, there exists a value ξ_n such that

$$R_p(t_{n+1}) = y^{(p+2)}(\xi_n) \int_{t_n}^{t_{n+1}} \frac{1}{(p+1)!}(t-t_n)(t-t_{n-1})\dots(t-t_{n-p})\,dt$$

for

$$= \rho_p y^{(p+2)}(\xi_n),$$

$$\rho_p = \int_{t_n}^{t_{n+1}} \frac{1}{(p+1)!}(t-t_n)(t-t_{n-1})\dots(t-t_{n-p})\,dt.$$

The expression (6.56) then simplifies to

$$y(t_{n+1}) = y(t_n) + \sum_{k=0}^{p} \lambda_k f(t_{n-k}, y(t_{n-k})) + \rho_p y^{(p+2)}(\xi_n)$$

where

$$\lambda_k = \int_{t_n}^{t_{n+1}} L_k(s)\,ds.$$

By dropping the residual and setting $y(t_k)$ to the approximate value y_k, we get the numerical method, known as the *Adams-Bashforth method* of order $p+1$:

$$y_{n+1} = y_n + \sum_{k=0}^{p} \lambda_k f(t_{n-k}, y_{n-k}).$$

Método de Adams-Bashforth

Si asumimos una malla uniforme con un espaciado h, entonces las fórmulas para λ_k y ρ_p se simplifican considerablemente (y están tabuladas):

TABLE 6.6 Adams-Bashforth Coefficients and Residual Terms

Steps	p	λ_0	λ_1	λ_2	λ_3	R_p
1	0	h				$\frac{1}{2}h^2y''(\xi_n)$
2	1	$\frac{3}{2}h$	$-\frac{1}{2}h$			$\frac{5}{12}h^3y'''(\xi_n)$
3	2	$\frac{23}{12}h$	$-\frac{16}{12}h$	$\frac{5}{12}h$		$\frac{3}{8}h^4y''''(\xi_n)$
4	3	$\frac{55}{24}h$	$-\frac{59}{24}h$	$\frac{37}{24}h$	$-\frac{9}{24}h$	$\frac{251}{720}h^5y'''''(\xi_n)$

ρ_p

Método de Adams-Bashforth

Ejemplo:

We return to our example problem,

$$y' = -y \ln y, \quad y(0) = \frac{1}{2},$$

the solution to which we will approximate using the second-order Adams-Bashforth scheme, commonly denoted AB2, with $h = \frac{1}{8}$. We choose to use Euler's method to generate the starting value; [8] thus our first computation is

$$y_1 = y_0 + hf(t_0, y_0) = \frac{1}{2} + \frac{1}{8}\left(-\frac{1}{2} \times \ln \frac{1}{2}\right) = 0.5433216988.$$

Now we can do the first AB2 step, for y_2:

2	1	$\frac{3}{2}h$	$-\frac{1}{2}h$	$\frac{5}{12}h^3 y'''(\xi_n)$

$$\begin{aligned} y_2 &= y_1 + \frac{1}{2}h(3f(t_1, y_1) - f(t_0, y_0)) \\ &= 0.5433216988 + \frac{1}{2} \times \frac{1}{8}\left((-3(0.5433216988)\ln(0.5433216988) + \frac{1}{2}\ln\frac{1}{2}\right) \\ &= 0.583808738. \end{aligned}$$

Método de Adams-Bashforth

Ejemplo (cont.):

2	1	$\frac{3}{2}h$	$-\frac{1}{2}h$	$\frac{5}{12}h^3y'''(\xi_n)$

This is followed by the second AB2 step, for y_3:

$$
\begin{aligned}
y_3 &= y_2 + \frac{1}{2}\frac{1}{8}(3f(t_2,y_2) - f(t_1,y_1)) \\
&= 0.583808738 + \frac{1}{2} \times \frac{1}{8}(-3(0.583808738)\ln(0.583808738) \\
&\quad + 0.5433216988\ \ln(0.5433216988)\ ) \\
&= 0.622004388.
\end{aligned}
$$

Thus, $y(t_3) = y(0.375) \approx 0.622004388$. The exact solution is $y = e^{-(\ln 2)e^{-t}}$; thus $y(0.375) = 0.6210196063$, so our accuracy is not bad. We would have done better using a more accurate method for the starting value, of course. ■

Método de Adams-Moulton

The Adams-Bashforth methods are based on an interpolating polynomial that is defined using the nodes $t_n, t_{n-1}, \ldots, t_{n-p}$. If we use the same number of nodes but include $t = t_{n+1}$, then (6.56) becomes

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} \sum_{k=-1}^{p-1} L_k(s) f(t_{n-k}, y(t_{n-k}))\, ds + R_p(t_{n+1}), \tag{6.57}$$

where the Lagrange functions are now

If $k = -1$ then $t_{n-k} = t_{n+1}$

$$L_k(s) = \prod_{\substack{i=-1 \\ i \neq k}}^{p-1} \frac{s - t_{n-i}}{t_{n-k} - t_{n-i}}, \tag{6.58}$$

and so the numerical method—known as the *Adams-Moulton method* of order $p+1$—is

$$y_{n+1} = y_n + \sum_{k=-1}^{p-1} \gamma_k f(t_{n-k}, y_{n-k}),$$

where the constants are given by

If $k = -1$ then $t_{n-k} = t_{n+1}$

$$\gamma_k = \int_{t_n}^{t_{n+1}} L_k(s)\, ds.$$

Método de Adams-Moulton

Ejemplo (cont.):

Consider the example

$$y' = -y \ln y, \quad y(0) = y_0, \tag{6.59}$$

which has exact solution $y = e^{(-\ln 2)e^{-t}}$ when $y_0 = \frac{1}{2}$. We solve this in two ways:

1. Via the second-order Adams-Bashforth method, with the trapezoid rule predictor-corrector used to generate the starting value
2. Via the fourth-order Adams-Bashforth-Moulton predictor-corrector, using the fourth-order Runge-Kutta method to generate the starting values

The error results are summarized in Table 6.8.

Since they are such high-order methods, we need accurate techniques for getting the starting values for the Adams methods. Generally, the Runge-Kutta methods (Section 6.5) are used for this purpose. ■

Método de Adams-Moulton

Ejemplo (cont.):

TABLE 6.8 Adams Family Example for $y' = -y \ln y$

$n = h^{-1}$	Second-order AB $\max_{k \le n} \lvert y(t_k) - y_k \rvert$	Fourth-order ABM PC $\max_{k \le n} \lvert y(t_k) - y_k \rvert$
8	0.284743357305 12E–02	0.19742002576040E–05
16	0.67170745938105E–03	0.12099205870530E–06
32	0.16212367000479E–03	0.72652883709168E–08
64	0.39755173993794E–04	0.44197434601045E–09
128	0.98386454340238E–05	0.27204238861600E–10
256	0.24469431793017E–05	0.16866508190105E–11
512	0.61013351659867E–06	0.10536016503693E–12
1024	0.15233231198675E–06	0.76605388699136E–14

Método de Diferencias Retrógradas

BDF (Backward Difference Formula):

Again, we let the differential equation be $y'(t) = f(t, y(t))$,

and this time we construct an interpolating polynomial to $y(t)$ (instead of $f(t, y(t)) = y'(t)$) at the nodes $t_{n-p+1}, t_{n-p+2}, \ldots, t_{n+1}$. Thus we have

$$q(t) = \sum_{k=-1}^{p-1} L_k(t) y(t_{n-k}) \approx y(t),$$

where the L_k are defined as in the case of Adams-Moulton. Now we use this polynomial to approximate $y'(t_{n+1})$, using the scheme outlined in Section 4.5. (This choice of argument to y' means that the BDF methods are all implicit.) Thus we have

$$y'(t_{n+1}) = q'(t_{n+1}) + \frac{1}{(p+1)!} w_p'(t_{n+1}) y^{(p+1)}(\xi_p),$$

where

$$w_p(t) = \prod_{k=-1}^{p-1} (t - t_k),$$

$$f'(x_i) - p_n'(x_i) = \frac{1}{(n+1)!} w_n'(x_i) f^{(n+1)}(\xi_i), \qquad (4.20)$$

Método de Diferencias Retrógradas

BDF (Backward Difference Formula):

We substitute this into the differential equation to get

$$\sum_{k=-1}^{p-1} L_k'(t_{n+1})y(t_{n-k}) = f(t_{n+1}, y(t_{n+1})) - \frac{1}{(p+1)!} w_p'(t_{n+1}) y^{(p+1)}(\xi_p),$$

which we can solve for $y(t_{n+1})$ to get

$K = -1$

$$y(t_{n+1}) = \sum_{k=0}^{p-1} \mu_k y(t_{n-k}) + \nu f(t_{n+1}, y(t_{n+1})) - \frac{\nu}{(p+1)!} w_p'(t_{n+1}) y^{(p+1)}(\xi_p)$$

where the constants are defined as

$$\mu_k = -\frac{L_k'(t_{n+1})}{L_{n+1}'(t_{n+1})}, \qquad \nu = \frac{1}{L_{n+1}'(t_{n+1})}.$$

Método de Diferencias Retrógradas

BDF (Backward Difference Formula):

TABLE 6.9 Backward Difference Family Coefficients and Residual Terms

Steps	p	ν	μ_0	μ_1	μ_2	μ_3	R_p
1	1	h	1				$-\frac{1}{2}h^2 y''(\xi_n)$
2	2	$\frac{2}{3}h$	$\frac{4}{3}$	$-\frac{1}{3}$			$-\frac{2}{9}h^3 y'''(\xi_n)$
3	3	$\frac{6}{11}h$	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$		$-\frac{3}{22}h^4 y''''(\xi_n)$
4	4	$\frac{12}{25}h$	$\frac{48}{25}$	$-\frac{36}{25}$	$\frac{16}{25}$	$-\frac{3}{25}$	$-\frac{12}{125}h^5 y'''''(\xi_n)$

Estabilidad

Definition 6.5 (Stability Polynomial) *Consider a multistep method in the form*

$$y_{n+1} = \sum_{k=0}^{p} a_k y_{n-k} + \sum_{k=-1}^{p} b_k f(t_{n-k}, y_{n-k}).$$

Then the stability polynomial for this method is given by

$$\sigma(r) = r^{p+1} - \sum_{k=0}^{p} a_k r^{p-k}. \tag{6.62}$$

Estabilidad

Definition 6.6 (Root conditions) *Consider a multistep method with stability polynomial σ. Denote the roots of σ by $r_0, r_1, \ldots, r_p$. Define the following conditions.*

Root Condition *If*

$$|r_k| \leq 1, \quad 0 \leq k \leq p$$

and all roots that satisfy $|r_j| = 1$ are simple roots, then we say that the stability polynomial satisfies the root condition.

Strong Root Condition *If*

$$r_0 = 1, |r_k| < 1, \quad 1 \leq k \leq p,$$

then we say that the stability polynomial satisfies the strong root condition.

Estabilidad

Then we have the following:

1. If the stability polynomial satisfies the root condition, then the method is stable, in the sense that for h sufficiently small it will deliver accurate results over a closed interval $[0,T]$.

2. If the stability polynomial satisfies the strong root condition, then the method is *relatively stable*, meaning that, for h sufficiently small, the parasitic solution components will go to zero as $n \to \infty$.

3. A method that is stable but not relatively stable is called weakly stable and will exhibit the kind of behavior seen in Section 6.4.4.

Estabilidad

Ejemplo:

Consider the $(p+1)$-step Adams-Bashforth method:

$$y_{n+1} = y_n + \sum_{k=0}^{p} \lambda_k f(t_{n-k}, y_{n-k}).$$

The stability polynomial, based on the formal definition (6.62), will be

$$\sigma(r) = r^{p+1} - r^p;$$

thus the roots are $r = 1$ and $r = 0$ (multiplicity p). This satisfies the strong root condition, so we know the Adams-Bashforth methods are relatively stable. The same analysis holds for the Adams-Moulton methods, so we know they are relatively stable as well. ■