AN AUTOMATIC PROGRAM

Automatic programs attempt to control the error based on a tolerance given by user. We illustrate what is involved by constructing a simple program based on the trapezoidal method,

$$y_{n+1} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right], \quad n \geq 0$$

The *first* question to ask is what error is to be controlled. Most people would assume this to mean that we control

$$Y(x_{n+1}) - y_{n+1}$$

This is called the *global error* in the solution y_{n+1}. This error is a consequence of all the preceding errors $Y(x_j) - y_j$, $j = 0, 1, ..., n$, as well as the truncation error T_n in passing from x_n to x_{n+1}. Thus if $Y(x_{n+1}) - y_{n+1}$ is found to be too large, the only solution is to recompute all of the past points y_j with a smaller stepsize h. In general, this is not practical.

In practice, we actually attempt to measure the local error:

$$u_n(x_{n+1}) - y_{n+1}$$

With the trapezoidal rule, we have

$$LE \equiv u_n(x_{n+1}) - y_{n+1} = -\frac{h^3}{12} u_n'''(x_n) + O(h^4)$$

This is what we actually try to estimate; and using it, we attempt to make sure the error LE is neither too large (for obvious reasons) nor too small (as this would result in unnecessary calculations).

We assume a user-supplied error parameter $\epsilon > 0$. Then we attempt to have a stepsize $h = x_{n+1} - x_n$ for which

$$\frac{1}{4}\epsilon\, h \leq |LE| \leq \epsilon\, h$$

at each step. If the upper limit is exceeded, we reduce h; and if the value of $|LE|$ is less that the lower bound, then we increase h. In both cases, we then re-compute the value of y_{n+1} with the new stepsize h.

Using

$$\frac{1}{4}\epsilon\, h \leq |LE| \leq \epsilon\, h \tag{1}$$

is called *controlling the local error per unit stepsize*. This is a commonly used technique in many ODE packages, for reasons we will explain.

Write

$$\begin{aligned} Y(x_{n+1}) - y_{n+1} &= [Y(x_{n+1}) - u_n(x_{n+1})] \\ &\quad + [u_n(x_{n+1}) - y_{n+1}] \end{aligned} \tag{2}$$

The first line on the right side is the error in the local solution $u_n(x_{n+1})$, and the second line is LE. As an example, consider the case of $f_y(x, y) \leq 0$. Then

$$Y'(x) = f(x, Y(x))$$

$$u_n'(x) = f(x, u_n(x)), \quad u_n(x_n) = y_n$$

$$\begin{aligned} Y'(x) - u_n'(x) &= f(x, Y(x)) - f(x, u_n(x)) \\ &= \frac{\partial f(x, \zeta_x)}{\partial y}\, [Y(x) - u_n(x)] \end{aligned}$$

This solves to give

$$Y(x) - u_n(x) = \exp\left[\int_{x_n}^{x} \frac{\partial f(x,\zeta_x)}{\partial y} dt\right] (Y(x_n) - y_n)$$

This shows

$$|Y(x_{n+1}) - u_n(x_{n+1})| \leq |Y(x_n) - y_n|$$

Return to (2),

$$\begin{aligned} Y(x_{n+1}) - y_{n+1} &= Y(x_{n+1}) - u_n(x_{n+1}) \\ &\quad + u_n(x_{n+1}) - y_{n+1} \end{aligned}$$

Then

$$\begin{aligned} |Y(x_{n+1}) - y_{n+1}| &\leq |Y(x_{n+1}) - u_n(x_{n+1})| \\ &\quad + |u_n(x_{n+1}) - y_{n+1}| \\ &\leq |Y(x_n) - y_n| + \epsilon\,(x_{n+1} - x_n) \end{aligned}$$

This leads to

$$|Y(x_n) - y_n| \leq |Y_0 - y_0| + \epsilon\,(x_n - x_0)$$

Thus if we want

$$|Y(x_n) - y_n| \le \delta, \quad x_0 \le x_n \le b$$

and if $y_0 = Y_0$, then we would want to require

$$|Y(x_n) - y_n| \le \epsilon\,(x_n - x_0) \le \delta$$

for all nodes x_n. This is true if

$$\epsilon = \frac{\delta}{b - x_0}$$

Most programs control the local error per unit stepsize, although it is often not obvious that they are doing so.

In the bound

$$\frac{1}{4}\epsilon\,h \le |LE| \le \epsilon\,h$$

the lower bound is such that we do not change stepsizes too often, as that part of the program will be more expensive.

THE PROGRAM

For a general step, with $x_{n+1} - x_n = x_n - x_{n-1}$, we use the midpoint predictor

$$y_{n+1}^{(0)} = y_{n-1} + 2hf(x_n, y_n)$$

and we interate once. Thus we choose $y_{n+1} = y_{n+1}^{(1)}$. From last lecture,

$$u_n(x_{n+1}) - y_{n+1}^{(0)} = \frac{5h^3}{12} u_n'''(x_n) + O(h^4)$$

$$u_n(x_{n+1}) - y_{n+1} = -\frac{h^3}{12} u_n'''(x_n) + O(h^4)$$

Thus,

$$y_{n+1} - y_{n+1}^{(0)} = \frac{h^3}{2} u_n'''(x_n) + O(h^4)$$

This allows us to predict LE:

$$LE = u_n(x_{n+1}) - y_{n+1} \approx -\frac{1}{6}\left[y_{n+1} - y_{n+1}^{(0)}\right]$$

We use this formula to predict the local error. If it does not satisfy our bound (1), then we change the value of h to make it satisfy the bound. This is described on page 375 of the text.

When we are just beginning with a new stepsize h, we cannot use the midpoint predictor. Then we use the Euler predictor,

$$y_{n+1}^{(0)} = y_n + hf(x_n, y_n)$$

and we iterate twice: $y_{n+1} = y_{n+1}^{(2)}$. The description of predicting the local error LE in this case is described on page 374 of the text. Changing h is expensive when compared to the cost of an ordinary step using the midpoint predictor $y_{n+1}^{(0)}$; and we do not want to change steps too often.

ASSORTED NOTES

A practical code must take into account special situations which may arise.

- The local error

  $$-\frac{h^3}{12}u_n'''(x_n)$$

  may be quite small due to $u_n'''(x_n) \approx 0$. Then the lower bound

  $$\frac{1}{4}\epsilon\, h \leq |LE|$$

  may be violated, and we may attempt to increase h by too large an amount. For this reason, we never allow h to increase by more than a factor of 2.

- What if h is made so large that important information about the solution is being missed. The user is asked to supply a parameter h_{max}, and this is not to be exceeded.

- What if h is made so small that it computation becomes too expensive. We ask the user for a parameter h_{min} to avoid this happening. When this reached, the user is warned, but the computation continues.

EXAMPLE

Using *DETRAP*, solve

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

The true solution is

$$Y(x) = \frac{x}{1+x^2}$$

For the local error, we note that

$$Y'''(x) = -\frac{6(x^4 - 6x^2 + 1)}{\left(1+x^2\right)^4}$$

and this is zero at $x \doteq \pm 0.414, \pm 2.414$. Thus the local error should be around zero at these points, irregardless of the value of h. Look at the tables from *DETRAP* for this problem. In the second table, the true local error is predicted by another ODE solver of high accuracy.

A BACKWARD ERROR ANALYSIS

Using the local solution $u_n(x)$ and the local error $LE_n = u_n(x_{n+1}) - y_{n+1}$, define

$$v(x) = u_n(x) - \frac{x - x_n}{x_{n+1} - x_n} LE_n, \quad x_n \le x \le x_{n+1}$$

and do this for each subinterval $[x_n, x_{n+1}]$ within the total interval $[x_0, b]$. On $[x_n, x_{n+1}]$, the function $v(x)$ satisfies

$$v(x_n) = y_n, \qquad v(x_{n+1}) = y_{n+1}$$

and thus it is continuous over $[x_0, b]$. The derivative, however, is discontinuous at the node points

$$\begin{aligned} v'(x) &= u_n'(x) - \frac{LE_n}{x_{n+1} - x_n} \\ &= f(x, u_n(x)) - \frac{LE_n}{x_{n+1} - x_n}, \qquad x_n \le x \le x_{n+1} \end{aligned}$$

Define the *residual function*

$$r(x) = v'(x) - f(x, v(x)), \qquad x_0 \le x \le b$$

This also can be written as

$$v'(x) = f(x, v(x)) + r(x)$$

showing $v(x)$ to be the solution of a perturbation of the original differential equation. The residual $r(x)$ is discontinuous at the node points. On the interval $x_n \le x \le x_{n+1}$,

$$\begin{aligned}
r(x) &= f(x, u_n(x)) - f(x, v(x)) - \frac{LE_n}{x_{n+1} - x_n} \\
&= f(x, u_n(x)) - f\left(x, u_n(x) - \frac{x - x_n}{x_{n+1} - x_n} LE_n\right) \\
&\qquad - \frac{LE_n}{x_{n+1} - x_n} \\
|r(x)| &\le K \frac{x - x_n}{x_{n+1} - x_n} |LE_n| + \frac{|LE_n|}{x_{n+1} - x_n} \\
&\le \left(K + \frac{1}{x_{n+1} - x_n}\right) |LE_n|, \qquad x_n \le x \le x_{n+1}
\end{aligned}$$

As earlier, suppose the local error LE_n is so chosen that it satisfies

$$|LE_n| \leq \epsilon\,(x_{n+1} - x_n)$$

for each step of the trapezoidal rule. Then we have that on $[x_n, x_{n+1}]$,

$$|r(x)| \leq (K\,(x_{n+1} - x_n) + 1)\,\epsilon, \qquad x_n \leq x \leq x_{n+1}$$

This says that the perturbed differential equation is close to the original differential equation in most cases. Whether or not the numerical solution is close to the true solution will then depend on whether or not the differential equation is well-conditioned or stable.

This entire discussion generalizes to all other numerical methods for the initial value problem for ODEs. For additional discussion of backward error analysis, see L. Shampine, *Numerical Solution of Differential Equations*, 1994, Section 2.2.