Initial-Value Problems

Definition

- An *n*th-order initial-value problem is an ordinary differential equation of the form

$$\frac{d^{(n)}y}{dt^{(n)}} = f\left(t, y, \frac{dy}{dt}, \ldots \frac{d^{(n-1)y}}{dt^{(n-1)}}\right)$$

where we are given initial conditions

$y(t=t_0)=y0$

$dy/dt|_{t=t0} = dy_0$

…

$d^{(n-1)}y/dt^{(n-1)}|_{t=t0}=dny0$

First-Order Initial Value Problems

- We will consider only first order IVPs, since we will discover that any higher-order equation can be represented as a system of first-order equations. Thus we have

$$\frac{dy}{dt} = f(t, y), y(t_0) = y_0$$

- This equation need not be a function of time but we will conventionally represent it as such.

Taylor Series

- We can expand the unknown function y as a Taylor series to obtain

$$y(t) = y(t_0) + \left.\frac{dy}{dt}\right|_{t=t_0} (t - t_0) + \left.\frac{d^2y}{2!dt^2}\right|_{t=t_0} (t - t_0)^2 + \ldots$$

- Let $t-t_0=h$ and derivatives be denoted by primes. Then we can write

  $y(t)=y(t_0)+hy'(t_0)+(1/2)h^2y(t_0)''+(1/6)h^3y'''(t_0)+\ldots$

  Substituting from initial conditions and the ODE gives

  $y(t)=y_0+hf(t_0,y_0)+O(h^2)$

  The error is called the *local truncation error* or just the *local error*.

Euler's Method

- We observe that this Taylor series may be written for any small increment h away from any value of t, and our approximation will not be too bad. Hence
- $Y(t_0+h)=y(t_0)+hf(t_0,y_0)+O(h^2)$
- $Y(t_0+h+h)=y(t_0+h)+hf(t_0+h,y(t_0+h))+\ldots$
- Generalizing, we get

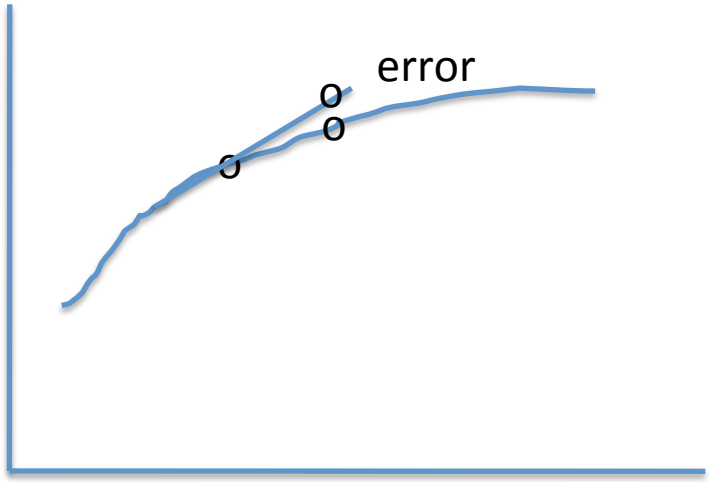
  $y(t+h)=y(t)+hf(t,y)$

  which we can write as

  $y^{(n+1)}=y^{(n)}+hf(t^n,y^n)$
- This is Euler's method. It has an error at *each step* of order h^2.
  - It can be shown that the global error that accumulates over many steps is O(h)

Euler's Method (Continued)

- Euler's method basically fits a line at each point and extrapolates to get the value of the next point.



Modified Euler Method

- We might try to improve Euler's method by averaging the slopes at the beginning and end of the interval:
- $y^{n+1}=y^n+0.5h(y'^n+y'^{n+1})$
- Of course, we don't know y'^{n+1}
- So let us estimate it by using the Euler approximation $y^{n+1}=y^n+hf(t^n,y^n)$ or
- $y^{n+1}=y^n+0.5h(f(t^n,y^n)+f(t^{n+1},y^n+hf(t^n,y^n)))$
- It can be shown that the local error for this method is $O(h^3)$ and the global error is $O(h^2)$.

Runge-Kutta Methods

- The Runge-Kutta methods are the workhorse of initial-value integration. They are highly accurate and work well for all but stiff systems.
- The derivation is based on a careful construction of Taylor series so as to cancel errors as much as possible.
- The modified Euler method (with no additional corrections applied) is a low-order Runge-Kutta scheme.

Fourth-Order Runge-Kutta

- Most widely used of all methods for IVPs. Accuracy is $O(h^4)$.
- Given by

$y_0=a$ (initial value)

$k_1=hf(t^n,y^n)$

$k_2=hf(t^n+0.5h,y^n+0.5k_1)$

$k_3=hf(t^n+0.5h,y^n+0.5k_2)$

$k_4=hf(t^{n+1},y^n+k_3)$

$y^{n+1}=y^n+(k_1+2k_2+2k_3+k_4)/6$

Runge-Kutta-Fehlberg

- Estimates the local error, often by using a Runge-Kutta method of one order higher, then adapts the step size as needed in order to contain the error within specified bounds. This is often called Runge-Kutta-Fehlbert integration.
- Runge-Kutta fourth order, especially with adaptive step sizes, is the main method used now. It is particularly good at finding its way along an irregular function, as long as the step size can be sufficiently small.

Multistep Methods

- These methods fit a polynomial of moderate order (such as a cubic) to $(t^n,y^n),(t^{n-1},y^{n-1}),\ldots..$ (t^{n-m},y^{n-m}).
  - This utilizes information accumulated during the run
  - But it requires that we use some other method to generate the M points that we need initially.
- Some multistep methods (Milne's in particular) are prone to instabilities. A *numerical instability* occurs when the error blows up.

Fourth-Order Adams-Bashforth

$y_0=a, y_1=a_1, y_2=a_2, y_3=a_3$

$$y^{n+1}=(h/24)[55f(t^n,y^n)-59f(t^{n-1},y^{n-1})+ 37f(t^{n-2},y^{n-2})-9f(t^{n-3},y^{n-3})]$$

Implicit Methods

- So far everything we have considered uses only known values to compute the next unknown. These are called *explicit* methods.
- Suppose that instead of using the m values from n to n-m we fit a polynomial from n+1 to n-m+1. We do not know y^{n+1} so it will appear on both sides of the equation. We do not know before developing the method whether we can solve for y^{n+1} and it could involve solving a linear system at each step. This can be quite slow.
- So why use implicit methods?
  - Very stable, in general, and a larger timestep can be used.
  - Not actually used much for ODEs anymore, except for stiff systems, but important for PDEs.

Adams-Moulton Three-Step Method

$y_0=a_0, y_1=a_1, y_2=a_2$

$$y^{n+1}=y^n+(h/24)[9f(t^{n+1},y^{n+1})+19f(t^n,y^n)$$
$$-5f(t^{n-1},y^{n-1})+f(t^{n-2},y^{n-2})]$$

This is a fourth-order method. Notice that we get a higher order for less work than the similar explicit Adams-Bashford scheme. In practice it also typically has a smaller absolute error than an explicit method. However, depending on the form of f(t,y) it is not always possible to solve for the n+1st value.

Predictor-Corrector Methods

- We cannot write a general program for an implicit method. Thus in practice their most common use is as part of a *predictor-corrector* method. We can use an explicit method to predict a value y^{n+1}, then use the generally more accurate implicit method to correct this value.
- One popular such method was to start off with a Runge-Kutta fourth order, then apply the four-step (and fourth order) Adams-Bashford method as the predictor, finally using the three-step Adams-Moulton method as the corrector.
- The two approximations to y^{n+1} can give us an estimate of the absolute error, so the algorithm could monitor this and reduce the step size if the error began to grow. This is another example of an adaptive method.

Extrapolation Methods

- The most important extrapolation method is the Bulirsch-Stoer (sometimes Gregg-Bulirsch-Stoer-) method. This method fits a rational function (a ratio of two polynomials) at various step sizes (Romberg integration), then making a clever choice so that the error is in terms of h^2 rather than h.
- This method is good for high-precision applications, or where function evaluations are expensive (though not many functions are "expensive" anymore).

Requirements for a Method

- Any method for solving a differential equation numerically must have the following properties.

1. Consistency. A method is *consistent* with the equation it approximates if

$$\lim_{h \to 0} \max_{1 \le i \le N} \left| \tau_i \right| = 0$$

where τ_i is the local truncation error at each step.

Requirements (Continued)

2. Convergence. A method is said to be *convergent* if

$$\lim_{h \to 0} \max_{1 \le i \le N} |y_i - w_i| = 0$$

where y_i is the exact solution and w_i is the approximation at each step.

3. Stability. A method is *stable* if small perturbations in the solution at each step (the initial conditions for the next step) produce small changes in the computed solution.