LECTURE 33: Local Analysis of One-Step Integrators

What can be said of the error between the computed solution $x_k$ at time $t_k = t_0 + kh$ and the exact solution $x(t_k)$? In this lecture and the next, we analyze this error, as a function of $k$, $h$, and properties of the ODE, for an important class of algorithms that generalize the forward Euler method.

5.2.1 Runge–Kutta Methods

To obtain increased accuracy in Euler’s method,

$$x_{k+1} = x_k + hf(t_k, x_k),$$

one might naturally reduce the step-size, $h$. Since Euler’s method derives from a first-order approximation to the derivative, we might expect the error to decay linearly in $h$. Before making this rigorous, first consider some better approaches: we are rarely satisfied with order-1 accuracy! By improving upon Euler’s method, we hope to obtain an improved solution while still maintaining a large time-step.

First consider a modification that might not look like such a big improvement: simply replace $f(t_k, x_k)$ by $f(t_{k+1}, x_{k+1})$ to obtain

$$x_{k+1} = x_k + hf(t_{k+1}, x_{k+1}),$$

called the backward Euler method. Because $x_{k+1}$ depends on the value $f(t_{k+1}, x_{k+1})$, this scheme is called an implicit method; to compute $x_{k+1}$, one needs to solve a (generally nonlinear) system of equations, rather more involved than the simple update required for the forward Euler method.

One can improve on both Euler methods by averaging the updates they make to $x_k$:

(5.1) \[ x_{k+1} = x_k + \frac{1}{2}h \left( f(t_k, x_k) + f(t_{k+1}, x_{k+1}) \right). \]

This method is the trapezoid method, for it can be derived by integrating the equation $x'(t) = f(t, x(t))$,

$$\int_{t_k}^{t_{k+1}} x'(t) \, dt = \int_{t_k}^{t_{k+1}} f(t, x) \, dt,$$

and approximating the integral on the right using the trapezoid rule. The fundamental theorem of calculus gives the exact formula for the integral on the left, $x(t_{k+1}) - x(t_k)$. Together, this gives

(5.2) \[ x(t_{k+1}) - x(t_k) \approx \frac{t_{k+1} - t_k}{2} \left( f(t_k, x(t_k)) + f(t_{k+1}, x(t_{k+1})) \right). \]

At each step, one must find a zero of the function

$$G(x_{k+1}) = x_{k+1} - x_k - hf(t_{k+1}, x_{k+1})$$

using, for example Newton’s method or the secant method. If $h$ is small and $f$ is not too wild, we might hope that we could get an initial guess $x_{k+1} \approx x_k$, or $x_{k+1} \approx x_k + hf(t_k, x_k)$. Note that this nonlinear iteration could require multiple evaluations of $f$ to advance the backward Euler method by one time step.
Replacing the inaccessible exact values $x(t_k)$ and $x(t_{k+1})$ with their approximations $x_k$ and $x_{k+1}$, and using the time-step $h = t_{k+1} - t_k$, equation (??) suggests
\[ x_{k+1} - x_k = \frac{h}{2} \left( f(t_k, x_k) + f(t_{k+1}, x_{k+1}) \right). \]
Rearranging this equation gives the trapezoid method (??) for $x_{k+1}$.

Like the backward Euler method, the trapezoid rule is implicit, due to the $f(t_{k+1}, x_{k+1})$ term. To obtain a similar explicit method, replace $x_{k+1}$ by its approximation from the explicit Euler method:
\[ f(t_k + h, x_{k+1}) \approx f(t_k + h, x_k + hf(t_k, x_k)). \]
The result is called Heun’s method or the improved Euler method:
\[ x_{k+1} = x_k + \frac{h}{2} \left( f(t_k, x_k) + f(t_k + h, x_k + hf(t_k, x_k)) \right). \]

Note that this method can be implemented using only two evaluations of the function $f(t, x)$.

The modified Euler method takes a similar approach to Heun’s method:
\[ x_{k+1} = x_k + hf(t_k + \frac{1}{2}h, x_k + \frac{1}{2}hf(t_k, x_k)), \]
which also requires two $f$ evaluations per step.

Additional function evaluations can deliver increasingly accurate explicit one-step methods, an important family of which are known as Runge–Kutta methods. In fact, the forward Euler and Heun methods are examples of one- and two-stage Runge–Kutta methods. The four-stage Runge–Kutta method is among the most famous one-step methods:
\[ x_{k+1} = x_k + \frac{1}{6}h \left( k_1 + 2k_2 + 2k_3 + k_4 \right), \]
where
\[ k_1 = f(t_k, x_k) \]
\[ k_2 = f(t_k + \frac{1}{2}h, x_k + \frac{1}{2}hk_1) \]
\[ k_3 = f(t_k + \frac{1}{2}h, x_k + \frac{1}{2}hk_2) \]
\[ k_4 = f(t_k + h, x_k + hk_3). \]

We must address an important consideration: these more sophisticated methods might potentially give better approximations of the solution $x(t)$, but they require more evaluations of the function $f$ per step than the forward Euler method. Many interesting applications give functions $f$ that are expensive to evaluate. One must make a trade-off: methods with greater accuracy allow for larger time-step $h$, but require more function evaluations per time step. To understand the interplay between accuracy and computational expense, we require a more nuanced understanding of the convergence behavior of these various methods.
5.2.2 Truncation Error

All explicit one-step methods can be written in the general form

\[ x_{k+1} = x_k + h \Phi(t_k, x_k; h). \]

Such methods incur two types of error:

1. The error due to the fact that even if the method was exact at \( t_k \),
   the updated value \( x_{k+1} \) at \( t_{k+1} \) will not be exact. This is called
   truncation error, or local error.

2. In practice, the value \( x_k \) is not exact. How is this discrepancy,
   the fault of previous steps, magnified by the current step? This
   accumulated error is called global error.

Let us make these notions of error more precise. At every given

time \( t_k, k = 1, 2, \ldots \), we have some approximation \( x_k \) to the value
\( x(t_k) \). Denote the global error by

\[ e_k := x(t_k) - x_k. \]

We seek to understand this error as a function of the step size \( h \).

To analyze the global error \( e_k \), first consider the approximations
made at each iteration. In the last lecture, we saw that Euler’s
method made an error by approximating the derivative \( x'(t_k) \) by a
finite difference,

\[ \frac{x(t_{k+1}) - x(t_k)}{h} \approx x'(t_k) = f(t_k, x(t_k)). \]

This type of error is made at every step. Generalize this error for all
explicit one-step methods.

**Definition 5.4.** The truncation error of an explicit one-step ODE integrator is defined as

\[ T_k = \frac{x(t_{k+1}) - x(t_k)}{h} - \Phi(t_k, x(t_k); h). \]

If \( T_k \to 0 \) as \( h \to 0 \), the method is consistent. If \( T_k = O(h^p) \), the
method has order-\( p \) truncation error.

The key to understanding truncation error is to note that \( T_k \) is essen-
tially just a rearranged version of the general one-step method, except
that the exact solutions \( x(t_k) \) and \( x(t_{k+1}) \) have replaced the approximations
\( x_k \) and \( x_{k+1} \). Thus, the truncation error can be regarded as a measure
of the error the method would make in a single step if supplied with
perfect data, \( x(t_k) \).
Example 5.9. It is simple to compute $T_k$ for the explicit Euler method:

$$T_k = \frac{x(t_{k+1}) - x(t_k)}{h} - \Phi(t_k, x(t_k); h) = \frac{x(t_{k+1}) - x(t_k)}{h} - f(t_k, x(t_k)) = \frac{x(t_{k+1}) - x(t_k)}{h} - x'(t_k).$$

This last substitution, $f(t_k, x(t_k)) = x'(t_k)$, is valid because $f$ is evaluated at the exact solution $x(t_k)$. (Recall that in general, $f(t_k, x_k) \neq x'(t_k)$.) Assuming that $x(t) \in C^2[t_k, t_{k+1}]$, we can expand $x(t)$ in a Taylor series about $t = t_k$ to obtain

$$x(t_{k+1}) = x(t_k) + hx'(t_k) + \frac{1}{2}h^2x''(\xi)$$

for some $\xi \in [t_k, t_{k+1}]$. Rearrange this to obtain a formula for $x'(t_k)$, and substitute it into the formula for $T_k$, yielding

$$T_k = \frac{x(t_{k+1}) - x(t_k)}{h} - x'(t_k) = \frac{x(t_{k+1}) - x(t_k)}{h} - \frac{x(t_{k+1}) - x(t_k)}{h} + \frac{1}{2}hx''(\xi) = \frac{1}{2}hx''(\xi).$$

Thus, the forward Euler method has truncation error $T_k = O(h)$, so $T_k \to 0$ as $h \to 0$.

Similarly, one can find that Heun’s method and the modified Euler’s method both have $O(h^2)$ truncation error, while the error for the four-stage Runge–Kutta method is $O(h^4)$. Extrapolating from this data, one might expect that a method requiring $m$ evaluations of $f$ can deliver $O(h^m)$ truncation error. Unfortunately, this is not true beyond $m = 4$, hence the fame of the four-stage Runge–Kutta method. All Runge–Kutta methods with $O(h^5)$ truncation error require at least six evaluations of $f$.

Next we must address a fundamental question: Does $T_k \to 0$ as $h \to 0$ ensure global convergence, $e_k \to 0$, for each $k = 1, 2, \ldots$?