Local Truncation Error

The local truncation error (LTE) of a numerical method is an estimate of the error introduced in a single iteration of the method, assuming that everything fed into the method was perfectly accurate. Recall that \(y_1, y_2, \ldots, y_N\) refer to the numerically computed values and \(y(t_1), y(t_2), \ldots, y(t_N)\) refer to the corresponding exact values (so that \(y_n \approx y(t_n)\)). To determine the local truncation error, analyse a general iteration of a method where the value \(y_{n+1}\) is computed. We want to determine the difference,

\[
LTE = y(t_{n+1}) - y_{n+1}
\]

based on the assumption that \(y_{n+1}\) is determined from exact information. That is, if we have a method of the form

\[
y_{n+1} = \phi(t_n, y_n, f, h)
\]

where \(\phi\) represents the formula for the numerical method, then we are going to assume that \(y_n = y(t_n)\), i.e.

\[
y_{n+1} = \phi(t_n, y(t_n), f, h)
\]

and, having made this assumption, we are going to examine the difference \(y(t_{n+1}) - y_{n+1}\) which we call the local truncation error.

Approach:

1. Replace all approximations (e.g. \(y_n\)) on the right-hand side of the method with their exact counterparts (e.g. \(y(t_n)\)). This gives an expression for \(y_{n+1}\) as determined by the method, assuming all inputs are exactly correct.

2. Taylor-expand everything that isn’t \(y(t_n)\) in the right-hand side (from step 1).

3. Taylor-expand the exact value \(y(t_{n+1})\).

4. Subtract the expanded form of the approximation \(y_{n+1}\) (from step 2) from the expanded form of the exact value \(y(t_n)\) (from step 3) to get the local truncation error:

\[
LTE = y(t_{n+1}) - y_{n+1}.
\]

If all goes well, things will simplify and cancel out so you’ll just be left with an upper bound on the error in big-O notation.
**Forward Euler:**

\[ y_{n+1} = y_n + hf(t_n, y_n) \]

This is a simple case where the method is based on a first order Taylor expansion. Assuming \( y_n = y(t_n) \) we have

\[
\begin{align*}
  y_{n+1} &= y(t_n) + hf(t_n, y(t_n)) \\
  &= y(t_n) + hy'(t_n)
\end{align*}
\]

since we are solving the ODE \( y'(t) = f(t, y(t)) \). Taylor’s expansion of the exact value is

\[ y(t_{n+1}) = y(t_n) + hy'(t_n) + O(h^2). \]

So the local truncation error is

\[ y(t_{n+1}) - y_{n+1} = O(h^2). \]

**Trapezoidal (Crank-Nicolson):**

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

Assume all the inputs to the method are exactly correct:

\[
\begin{align*}
  y_{n+1} &= y(t_n) + \frac{h}{2} \left( f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1})) \right)
\end{align*}
\]

Following the approach described previously, we need to Taylor-expand the term \( f(t_{n+1}, y_{n+1}) \). Well, since \( f(t_{n+1}, y(t_{n+1})) = y'(t_{n+1}) \), we will apply Taylor’s expansion to the derivative \( y'(t_{n+1}) \):

\[ y'(t_{n+1}) = y'(t_n) + y''(t_n)h + O(h^2) \]

Then (again, since \( y'(t) = f(t, y(t)) \)) we have

\[
\begin{align*}
  y_{n+1} &= y(t_n) + \frac{h}{2} \left( y'(t_n) + y'(t_n) + y''(t_n)h + O(h^2) \right) \\
  &= y(t_n) + hy'(t_n) + y''(t_n) \frac{h^2}{2} + O(h^3)
\end{align*}
\]

since \( \frac{h}{2} \times O(h^2) = O(h^3) \).

The Taylor series expansion of the exact value \( y(t_{n+1}) \) is

\[ y(t_{n+1}) = y(t_n) + y'(t_n)h + y''(t_n) \frac{h^2}{2} + O(h^3). \]

So the local truncation error is

\[ y(t_{n+1}) - y_{n+1} = O(h^3). \]
Another way to show the local truncation error for this method is to derive it, keeping track of the remainder terms as we go along. Once the numerical formula is derived, whatever error terms are left hanging around in the end will be the local truncation error. The idea was to start with a second order Taylor expansion:

\[ y(t_{n+1}) = y(t_n) + y'(t_n)h + y''(t_n)\frac{h^2}{2} + O(h^3) \]

and then use a slope approximation for the second derivative

\[ y''(t_n) = \frac{y'(t_{n+1}) - y'(t_n)}{h} + O(h). \]

Putting them together, we have

\[ y(t_{n+1}) = y(t_n) + y'(t_n)h + \left( \frac{y'(t_{n+1}) - y'(t_n)}{h} + O(h) \right) \frac{h^2}{2} + O(h^3) \]

\[ = y(t_n) + y'(t_n)h + (y'(t_{n+1}) - y'(t_n))h \frac{h}{2} + O(h^3) + O(h^3) \]

\[ = y(t_n) + \frac{h}{2} (y'(t_n) + y'(t_{n+1})) + O(h^3) \]

Finally, since \( y'(t) = f(t, y(t)) \), we have

\[ y(t_{n+1}) = y(t_n) + \frac{h}{2} (f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))) + O(h^3) \]

from which we get the Trapezoidal method. The terms on the right-hand side, other than the \( O(h^3) \) term, are what you would get from the method \( y_{n+1} \) if the inputs were exactly correct. i.e.

\[ y_{n+1} = y(t_n) + \frac{h}{2} (f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))). \]

So this shows that

\[ y(t_{n+1}) - y_{n+1} = O(h^3). \]

**Modified Euler:**

\[ y^*_n = y_n + hf(t_n, y_n) \]

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

We have seen that Forward Euler has \( O(h^2) \) local truncation error. So if

\[ y^*_n = y(t_n) + hf(t_n, y(t_n)) \]

then

\[ y(t_{n+1}) - y^*_n = O(h^2). \]

Now, for the trapezoidal step, we will assume that the inputs on the right-hand side are all exact – except for \( y^*_n \), because that was computed as part of the method and we need to address the error in the forward Euler step in our analysis. But we have assumed that the inputs to the forward Euler step, which produced \( y^*_n \), were all exact. So we have

\[ y_{n+1} = y(t_n) + \frac{h}{2} \left( f(t_n, y(t_n)) + f(t_{n+1}, y^*_n) \right) \]
To reiterate, we are assuming exact inputs in the forward Euler step to get \( y_{n+1}^* \), and we are using that in the trapezoidal step and assuming exact inputs for the rest of the inputs.

Now, recall that we had an expression for the exact value (equation (1)), from which the trapezoidal method was derived:

\[
y(t_{n+1}) = y(t_n) + \left( f(t_n, y(t_n)) + f(t_{n+1}, y_{n+1}^*) \right) \frac{h}{2} + O(h^3)
\]

Now now we want to expand the \( f(t_{n+1}, y(t_{n+1})) \) term in order to relate it to \( f(t_{n+1}, y_{n+1}^*) \). Writing it as the first derivative \( y'(t_{n+1}) \) isn’t helpful here because we are dealing with a change with respect to the \( y \) variable (not \( t \)). Here is where the Taylor series expansion for functions of two variables is useful. In this case, the “delta” in the first variable \( t \) is 0, and we are expanding about the point \((t, y) = (t_{n+1}, y_{n+1}^*)\) with the “delta” in the second variable being \( y(t_{n+1}) - y_{n+1}^* \):

\[
f(t_{n+1}, y(t_{n+1})) = f(t_{n+1}, y_{n+1}^*) + f_y(t_{n+1}, y_{n+1}^*)(y(t_{n+1}) - y_{n+1}^*) + O((y(t_{n+1}) - y_{n+1}^*)^2)
\]

Since \( y(t_{n+1}) - y_{n+1}^* = O(h^2) \) (the local truncation error of the forward Euler step) we can say that\(^1\)

\[
f_y(t_{n+1}, y_{n+1}^*)(y(t_{n+1}) - y_{n+1}^*) = O(h^2).
\]

And since \( O((y(t_{n+1}) - y_{n+1}^*)^2) = O(h^4) \) we have

\[
f(t_{n+1}, y(t_{n+1})) = f(t_{n+1}, y_{n+1}^*) + O(h^2)
\]

(the lower-order term \( O(h^2) \) dominates when \( h \to 0 \)).

Finally, bringing this back to (3) we have

\[
y(t_{n+1}) = y(t_n) + \frac{h}{2} \left( f(t_n, y(t_n)) + f(t_{n+1}, y_{n+1}^*) + O(h^2) \right) + O(h^3)
\]

\[
y(t_{n+1}) = y(t_n) + \frac{h}{2} \left( f(t_n, y(t_n)) + f(t_{n+1}, y_{n+1}^*) \right) + O(h^3)
\]

and subtracting (2) from this gives

\[
y(t_{n+1}) - y_{n+1} = O(h^3).
\]

\(^1\)Of course, by eliminating \( f_y(t_{n+1}, y_{n+1}^*) \) we are losing some information here, but when multiplied by something of order \( h^2 \) (which is very small) it does not contribute much to the result. Remember that this is the same thing we did when replacing the remainder term of a \( p \)-th-order Taylor expansion with \( O(h^{p+1}) \).