Stability Analysis

Siyang Wang

September 26, 2013

We analyze the stability condition of forward Euler method and backward Euler method by using the so called test equation.

Forward Euler method

The test equation reads

\[ y' = \lambda y \] (1)
\[ y(0) = \hat{y} \] (2)

where \( \lambda \) is a complex number. We also assume \( \hat{y} \neq 0 \), otherwise we get the trivial zero solution. The forward Euler method reads

\[ y_{k+1} = y_k + hf(t_k, y_k). \]

Substituting ODE to the above formula, we get

\[ y_{k+1} = y_k + h\lambda y_k = (1 + h\lambda)y_k. \]

Therefore, by induction we have

\[ y_k = (1 + h\lambda)^k \hat{y}. \] (3)

The exact solution of the test equation is

\[ y(t) = \hat{y}e^{\lambda t}. \]

(To see this, we take derivative with respect to \( t \) on both sides of the above equation, we have \( y'(t) = \lambda \hat{y}e^{\lambda t} = \lambda y \), which is the same as the ODE.) If we restrict \( \lambda \) such that its real part is negative, i.e. \( \text{Re}(\lambda) < 0 \), then the exact solution eventually decays to 0 as \( t \) goes to infinite (Do you know why?).
Numerically, this corresponds to the case that $y_k$ goes to 0 when $k$ goes to infinite in Equation (3). To satisfy this, we have to require

$$|1 + h\lambda| < 1.$$  \hspace{1cm} (4)

Inequality (4) is the stability condition for forward Euler method. We can also plot the stability condition to visualize the stability region as shown in Figure 1.

![Figure 1: Stability region (shadowed circle) for forward Euler method](image)

If we restrict that $\lambda$ is a real number (which is often the case in practice), then we can simplify the stability condition. From inequality (4), we obtain

$$-1 < 1 + h\lambda < 1$$

$$-2 < h\lambda < 0$$

$$0 < h < -2/\lambda$$

$h$ is the step size, so it is always positive. Then the stability condition is simplified to

$$h < -2/\lambda$$  \hspace{1cm} (5)

This is also what you have seen in the lab session.
**Backward Euler method**

We apply backward Euler method to the test equation (1) with the initial condition (2),

\[ y_{k+1} = y_k + hf(t_{k+1}, y_{k+1}) = y_k + h\lambda y_{k+1}. \]

Put \( y_{k+1} \) terms on one side,

\[ (1 - h\lambda)y_{k+1} = y_k \]

which gives

\[ y_{k+1} = \frac{1}{1 - h\lambda} y_k. \]

Therefore, by induction, we get

\[ y_k = \left( \frac{1}{1 - h\lambda} \right)^k \hat{y} \quad \text{(6)} \]

Again, we restrict \( \lambda \) to have negative real part so that the exact solution decays to 0 when \( t \) goes to infinite. Correspondingly, the numerical solution should have the same behaviour. Thus, we require

\[ \left| \frac{1}{1 - h\lambda} \right| < 1 \quad \text{(7)} \]

Before we claim the stability condition of backward Euler method, we can visualize inequality (7) in Figure 2.

Remember that we restrict that the real part of \( \lambda \) is negative. This corresponds to the left half plane, which is covered by the stability region. This means that backward Euler method is unconditionally stable.

**If we restrict that \( \lambda \) is a real number** (which is often the case in practice), then we can see the unconditional stability from inequality (7):

\[ |1 - h\lambda| > 1 \]

\[ 1 - h\lambda > 1 \text{ or } 1 - h\lambda < -1 \]

\[ h\lambda < 0 \text{ or } h\lambda > 2 \]

Since \( h > 0 \) (\( h \) is the step size, it must be a positive number) and \( \lambda < 0 \), we always have \( h\lambda < 0 \). Thus, backward Euler method is unconditionally stable.
\textbf{Remark}

Implicit methods are in general unconditionally stable. For a stiff problem, explicit methods need to take very small time steps. And implicit methods could be preferred in this case.

In MATLAB, we have ODE solvers suitable for stiff problems, for example \texttt{ode15s}.