Math 128a: Absolute Stability

GSI: Andrew Shi

Contents

1 Stiffness
   1.1 Motivation .................................................. 2
   1.2 Stability Functions .......................................... 5
   1.3 Examples ...................................................... 8
1 Stiffness

1.1 Motivation

Today we are going to study a particular class of ODEs known as “stiff ODEs”. These are a very common class of ODEs where it’s practically a necessity to use Implicit methods to get a reasonable solution. This will answer the question as to why we would ever use an implicit method despite the additional cost of needing to solve nonlinear equations.

Let’s consider the IVP:

\[ y' = -1000y, \quad y(0) = y_0 \]

The true solution for this IVP is easy to find and given by \( y(t) = y_0 e^{-1000t} \), which exhibits exponential decay. Let’s apply Euler’s method \( w_{n+1} = w_n + hf(t_n, w_n) \) to this IVP for some \( h \):

\[ w_{n+1} = w_n - 1000hw_n = (1 - 1000h)w_n \]

So we can see that the numerical solution is obtained by simply multiplying the one at the previous time by \( (1 - 1000h) \). Since the true solution exhibits exponential decay, at the very least we ask for our numerical solution to decay in absolute value (not even exponential decay like the true solution, we’re not being that greedy), which gives us the restriction:

\[ |(1 - 1000h)| < 1 \implies -2 < -1000h < 0 \implies h < \frac{2}{1000} \]

Typically, we like to pick our stepsize \( h \) based on accuracy concerns alone (based off LTE), but for this IVP we have an additional restriction on \( h \) here to make sure our numerical solution doesn’t blow up.

Now apply Implicit Euler’s method \( w_{n+1} = w_n - hf(t_{n+1}, w_{n+1}) \) to this IVP:

\[ w_{n+1} = w_n + h\lambda w_{n+1} \implies (1 - 1000h)w_{n+1} = w_n \]

\[ \implies w_{n+1} = \frac{1}{1 - 1000h}w_j \]

Here \(-1000h < 0\), so for any choice of of \( h > 0 \) the numerical solution will decay. So unlike the numerical solution from Explicit Euler’s method, there is no additional restriction on \( h \). This is the payoff of the implicit method - they are practically necessary to solve what we call “stiff” equations like this one. There is no precise definition for what constitutes a stiff equation. We might say that a stiff equation is one for which certain numerical methods are numerically unstable unless the step size is taken to be extremely small. The main idea is that the equation includes some terms which may lead to rapid variation in the solution. This is more obvious in the context of a system of equations.

Example 1 Consider the linear IVP:

\[ y' = -L(y(t) - \varphi(t)) + \varphi'(t) \quad y(0) = 2 \]
where \( \varphi(t) = \cos(30t) \). a) Solve the IVP exactly.

b) Write a script that will use Euler’s method to solve the IVP for \( 0 \leq t \leq 1 \) with \( L = 10^k \) for \( k = 1 \) to \( 5 \). For each \( L \) use \( h = 10^{-j} \) with \( j = 1 \) to \( 6 \). Tabulate the errors at the final time.

c) Repeat part b) with Implicit Euler. Comment on what you see.

**Solution:**
a) Multiply by an integrating factor:

\[
y' = -L(y(t) - \varphi(t)) + \varphi'(t) \implies y' + Ly(t) = \varphi'(t) + L\varphi(t)
\]

\[
\implies e^{Lt}y' + e^{Lt}Ly(t) = e^{Lt}\varphi' + Le^{Lt}\varphi
\]

\[
\implies (e^{Lt}y)' = (e^{Lt}\varphi)'
\]

\[
\implies e^{Lt}y = e^{Lt}\varphi(t) + C
\]

\[
y = \varphi(t) + Ce^{-Lt}
\]

Enforcing the initial condition \( y(0) = 2 \), we have

\[
2 = \varphi(0) + C
\]

\[
= 1 + C
\]

\[
\implies C = 1
\]

Therefore, the exact solution is:

\[
y(t) = \varphi(t) + (y_0 - 1)e^{-Lt} = \cos(30t) + \frac{e^{-Lt}}{\text{steady state, rapidly decaying}}
\]

b) For Euler’s Method, the below shows the table of the error at \( T = 1 \) for various values of \( L \) and \( h \).

<table>
<thead>
<tr>
<th>( L )</th>
<th>( h ) ( 10^{-1} )</th>
<th>( 10^{-2} )</th>
<th>( 10^{-3} )</th>
<th>( 10^{-4} )</th>
<th>( 10^{-5} )</th>
<th>( 10^{-6} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 10^1 )</td>
<td>3.316</td>
<td>1.351×10^{-1}</td>
<td>1.272×10^{-2}</td>
<td>1.265×10^{-3}</td>
<td>1.264×10^{-4}</td>
<td>1.264×10^{-5}</td>
</tr>
<tr>
<td>( 10^2 )</td>
<td>2.598×10^{-9}</td>
<td>2.026×10^{-3}</td>
<td>5.515×10^{-4}</td>
<td>5.833×10^{-5}</td>
<td>5.865×10^{-6}</td>
<td>5.866×10^{-7}</td>
</tr>
<tr>
<td>( 10^3 )</td>
<td>8.859×10^{-19}</td>
<td>2.644×10^{-6}</td>
<td>6.050×10^{-5}</td>
<td>5.647×10^{-6}</td>
<td>5.606×10^{-7}</td>
<td>5.626×10^{-8}</td>
</tr>
<tr>
<td>( 10^4 )</td>
<td>9.880×10^{-29}</td>
<td>3.658×10^{-199}</td>
<td>NaN</td>
<td>6.852×10^{-7}</td>
<td>6.806×10^{-8}</td>
<td>7.0429×10^{-9}</td>
</tr>
<tr>
<td>( 10^5 )</td>
<td>9.988×10^{-39}</td>
<td>9.047×10^{-299}</td>
<td>NaN</td>
<td>NaN</td>
<td>6.875×10^{-9}</td>
<td>9.275×10^{-10}</td>
</tr>
</tbody>
</table>

c) For Implicit Euler, the below shows the table of the error at \( T = 1 \) for various values of \( L \) and \( h \). (Results may vary depending on nonlinear solver in Implicit Euler).

<table>
<thead>
<tr>
<th>( L )</th>
<th>( h ) ( 10^{-1} )</th>
<th>( 10^{-2} )</th>
<th>( 10^{-3} )</th>
<th>( 10^{-4} )</th>
<th>( 10^{-5} )</th>
<th>( 10^{-6} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 10^1 )</td>
<td>0.879</td>
<td>0.119</td>
<td>1.256×10^{-2}</td>
<td>1.263×10^{-3}</td>
<td>1.264×10^{-4}</td>
<td>1.264×10^{-5}</td>
</tr>
<tr>
<td>( 10^2 )</td>
<td>0.212</td>
<td>9.054×10^{-3}</td>
<td>6.215×10^{-4}</td>
<td>5.903×10^{-5}</td>
<td>5.872×10^{-6}</td>
<td>5.871×10^{-7}</td>
</tr>
<tr>
<td>( 10^3 )</td>
<td>2.471×10^{-2}</td>
<td>1.138×10^{-4}</td>
<td>5.154×10^{-5}</td>
<td>5.557×10^{-6}</td>
<td>5.598×10^{-7}</td>
<td>5.578×10^{-8}</td>
</tr>
<tr>
<td>( 10^4 )</td>
<td>2.513×10^{-3}</td>
<td>2.329×10^{-6}</td>
<td>6.362×10^{-7}</td>
<td>6.763×10^{-8}</td>
<td>6.809×10^{-9}</td>
<td>6.572×10^{-10}</td>
</tr>
</tbody>
</table>
Some observations:

- Forward Euler requires smaller $h$ as $L$ gets large and the problem becomes more stiff. But backwards Euler has no such issues, demonstrating its attractive stability properties.

- For a given $L$, we can see that error scales with the mesh as expected, $O(h)$.

- These tables shows the point – for a given problem (a given $L$), you would like to just have to pick $h$ based off the desired level of accuracy. However, for an explicit method applied to a stiff equation, you also have to make sure to pick $h$ based on stability concerns.

Here are some accurate solutions for each $L$ (using $h = 10^{-6}$). Note as $L$ gets larger, the transient part of the solution (exponential decay) disappears faster.
1.2 Stability Functions

In fact, we can take any one-step method and apply it to the so-called “test problem” $y' = \lambda y \quad \lambda < 0$ to arrive at an expression of the form $w_{n+1} = Q(h\lambda)w_n$. This expression $Q(h\lambda)$ is known as the stability function of the numerical method. This is also commonly written as $R(z)$, where we let $z := h\lambda$. So we have for Explicit Euler’s method that $Q(h\lambda) = R(z) = 1 + z$ and for Implicit Euler’s method that $R(z) = \frac{1}{1 - z}$.

The numerical solution:

- Grows exponentially if $|Q(h\lambda)| > 1$
- Remains Bounded if $|Q(h\lambda)| = 1$
- Converges to zero if $|Q(h\lambda)| < 1$

We say that the region of absolute stability (RAS) is:

$$\text{RAS} = \{z \in \mathbb{C} : |Q(z)| < 1\}$$

So the region of absolute stability for Euler’s method is given by:

$$\text{RAS} = \{z \in \mathbb{C} : |1 + z| < 1\}$$

It is given by the shaded portion in the complex plane.
This means that for a given $\lambda$ in the problem for all $h$ such that $h\lambda$ in the RAS, it will result in a numerical solution that converges to 0. Here’s a concrete example: solving $y' = 100y$ requires a stepsize of $h > \frac{2}{100}$ to give a numerical solution that decays to zero.

For Backward Euler:

$$\left|\frac{1}{1-z}\right| < 1 \implies |1 - z| > 1 \implies \text{RAS} = \{ z \in \mathbb{C} : |1 - z| > 1 \}$$

![Backward Euler](image)

**Example 2** Consider the Implicit Trapezoidal Method:

$$w_{n+1} = w_n + \frac{h}{2} [f(t_n, w_n) + f(t_{n+1}, w_{n+1})]$$

Find the stability function and the corresponding region of absolute stability.

**Solution:** Apply the method to the test equation $y' = \lambda y$:

$$w_{n+1} = w_n + \frac{h}{2} [\lambda w_n + \lambda w_{n+1}]$$

$$= w_n + \frac{h\lambda}{2} w_n + \frac{h\lambda}{2} w_{n+1}$$

$$\implies \left(1 - \frac{h\lambda}{2}\right) w_{n+1} = \left(1 + \frac{h\lambda}{2}\right) w_n$$

$$\implies w_{n+1} = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} w_n$$

Let’s rewrite the stability function using $z = h\lambda$:

$$Q(\lambda) = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} = \frac{1 + \frac{z}{2}}{1 - \frac{z}{2}} = \frac{2 + z}{2 - z} = R(z)$$

Let’s find the Region of Absolute Stability. Rewrite $z = x + iy$:

$$|Q(z)| = \left|\frac{2 + x + iy}{2 - x - iy}\right| = \frac{(2 + x)^2 + y^2}{(2 - x)^2 + y^2} = \frac{\text{num}}{\text{denom}}$$
So to have $|Q(z)| < 1$ we need $\text{num} < \text{denom}$.

\[
\begin{align*}
\text{num} &< \text{denom} \\
(2 + x)^2 + y^2 &< (2 - x)^2 + y^2 \\
(2 + x)^2 &< (2 - x)^2 \\
x^2 + 4x + 4 &< x^2 - 4x + 4 \\
8x &< 0 \\
x &< 0
\end{align*}
\]

So the Region of Absolute Stability is given by the the left half-plane.

Here is another way we could have done this without the calculation. We want to find

\[
\{ z \in \mathbb{C} : |2 + z| < |2 - z| \}
\]

In english, this is the set of all points on the complex plane that are close to the point -2 than the point 2. But that set of points is precisely the left half-plane (not including the $y$-axis).

Finally, we have one more definition. We say a method is **A-stable** if its Region of Absolute Stability includes the left half-plane. So based off the three pictures above:

- Explicit Euler: Not A-stable
- Implicit Euler: A-stable
- Implicit Trapezoid Rule: A-stable (but just barely!)

The reason we like $A$-stable methods is that if the RAS contains the left half-plane, this means we can pick any stepsize and the resulting numerical solution of the test equation will decay to zero. With $A$-stable methods, the stepsize is controlled by the truncation error rather than by stability criteria. In other words, for stiff problems, implicit methods allow you to take much larger steps for a given error tolerance.
1.3 Examples

Example 3 Compute the stability function \( Q(h\lambda) \) - also known as \( R(z) \) - of the following Runge-Kutta method, and determine if its A-stable.

\[
  w_{n+1} = w_n + \frac{h}{4} f(t_n, w_n) + \frac{3h}{4} f(t_n, w_n + \frac{2}{3} hf(t_n, w_n))
\]

Solution: Apply the method to the test equation \( y' = \lambda y \):

\[
  w_{n+1} = w_n + \frac{h\lambda}{4} w_n + \frac{3h\lambda}{4} \left( w_n + \frac{2h\lambda}{3} w_n \right)
  = \left( 1 + \frac{h\lambda}{4} + \frac{3h\lambda}{4} + \frac{1}{2}(h\lambda)^2 \right) w_n
  = \left( 1 + z + \frac{z^2}{2} \right) w_n
\]

So the stability function is \( R(z) = 1 + z + \frac{z^2}{2} \). This is clearly not A-stable since \( R(-4) = 1 - 4 + 8 = 5 \), so \( |R(z)| \) is not bounded by 1 for \( z \in \mathbb{C}^- \).

Example 4 Consider the implicit midpoint method for solving \( y' = f(t, y) \):

\[
  k_1 = f \left( t_n + \frac{1}{2}h, w_n + \frac{1}{2}hk_1 \right)
  w_{n+1} = w_n + hk_1
\]

Compute the stability function \( Q(z) \) for this scheme. Is the method A-stable?

Solution: Apply the method to the test equation \( y' = \lambda y \):

\[
  k_1 = \lambda w_n + \frac{1}{2}h\lambda k_1 \implies \left( 1 - \frac{1}{2}h\lambda \right) k_1 = \lambda w_n \implies k_1 = \frac{\lambda}{1 - \frac{1}{2}h\lambda} w_n
  \implies w_{n+1} = \left( 1 + \frac{h\lambda}{1 - \frac{1}{2}h\lambda} \right) w_n = \frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda} w_n = \frac{2 + h\lambda}{2 - h\lambda} w_n
\]

So \( Q(z) = \frac{2 + z}{2 - z} \). This is A-stable as we showed earlier, since it’s the same stability function as the Implicit Trapezoid Method.
**Example 5** Find the stability function of the fourth-order Runge-Kutta method. Is it A-stable?

\[
\begin{align*}
    k_1 &= hf(t_j, w_j) \\
    k_2 &= hf(t_j + \frac{h}{2}, w_j + \frac{1}{2}k_1) \\
    k_2 &= hf(t_j + \frac{h}{2}, w_j + \frac{1}{2}k_2) \\
    k_2 &= hf(t_{j+1}, w_j + k_3) \\
    w_{j+1} &= w_j + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{align*}
\]

**Solution:** Details left to the reader. \( R(z) = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \frac{z^4}{4!} \) and not A-stable.

A **final word on stiff equations and absolute stability:** We have devoted a lot of energy to understanding how numerical methods will work on the simple problem

\[
y' = \lambda y \quad y(0) = y_0 \quad \lambda < 0
\]

But why? Why did we spend so much effort and develop all these definitions based off a simple problem we can solve exactly?

The idea is that if our numerical method is to have any hope of performing well on a more complex stiff equation/system, at the very least it should perform well on this simple problem. In fact, all of these bounds on \( h \) we can get as a result of this are only a guarantee for this test problem. As such, these only give us a necessary (but far from sufficient) restriction when faced with a more general, difficult problem. For the more general class of nonlinear ODEs we want to solve numerically, we actually have no hope of developing any general theory, so this is in some sense the best we can do.