Math 128A, Summer 2019
Lecture 23, Wednesday July 31, 2019

1 Stiff Equations

- Linear Stability
- Nonlinear Stability

Today we're going to look at a different problem to test:

$$\begin{aligned} y' &= \lambda(y-\phi(t)) + \phi'(t) \\ (y-\phi)' &= \lambda(y-\phi) \\ y-\phi &= e^{\lambda t}[y(0)-\phi(0)] \\ y(t) &= \phi(t) + \underbrace{e^{\lambda t}[y(0)-\phi(0)]}_{\text{transient}} \end{aligned}$$

where $\lambda \in \mathbb{C}$ is the dampening constant if it's negative (Re$\lambda << 0$), and $\phi(t)$ describes something that oscillates smoothly (like a massage chair), and λ describes how we want to get to that smooth oscillation. We insert a forcing term $+\phi'(t)$ into the first line so this is solvable.
If we happen to start exactly at the correct spot, $\phi(0)$, then $y(0) = \phi(0)$.

$$e^{\lambda t} = e^{\alpha t}(\cos(\beta t) + i\sin(\beta t)), \qquad \text{when } \lambda = \alpha + i\beta$$

(Alternatively, we see this in an old car as we go over a bump; there is a number of oscillations before our shock absorbers stabilize the bumpiness.)
Last time, we observed that Euler's method does not work very well for this situation. To see this explicitly, consider:

$$\begin{aligned} u_{n+1} &= u_n + h\left[\lambda(u_n - \phi_n) + \phi'_n\right] \\ &= \underbrace{(1+h\lambda)u_n} + hg_n \end{aligned}$$

This is the scary part because if $|1+h\lambda| > 1$, then u_{n+1} will oscillate instead of decaying. Alternatively, if $|\text{Re}(\lambda)| > 0$, then this will not blow up exponentially. This method still converges if $h \to 0$, but the issue is that if $\lambda << 0$ (extremely negative), then small h does too much work to follow the smooth solution $y(t) \approx \phi(t)$.

Remark: This restriction $|1+h\lambda| < 1$ will be a very severe restriction, relative to accuracy (it is more than we need for accuracy; it is a very expensive hypothesis).

In the last lecture (22), we observed that **implicit euler** makes this problem go away, but it is only first-order accurate (and may be way too expensive as well). Additionally, in the last lecture, we found that we must solve by Newton's method because fixed point iteration will not converge.

Now, we want to generalize so that we can consider other methods (and not only Euler). One way that delivers 99% of the story is **linear stability theory**.

2 Linear Stability

The idea is we check the method on

$$y' := \lambda y, \qquad \mathrm{Re}(\lambda) \le 0$$

to verify that

$$|u_{n+1}| \le |u_n|.$$

The least we should ask for is that the numerical method should decay, because it is generally not helpful to model something by an explosion.
Recall:

$$\begin{aligned}
\text{Euler: } u_{n+1} &= \overbrace{(1+h\lambda)}^{\mathrm{Re}(z)} u_n \\
\text{Implicit Euler: } u_{n+1} &= u_n + h\lambda \underline{u_{n+1}} \\
(1-h\lambda)u_{n+1} &= u_n \\
u_{n+1} &= \underbrace{\frac{1}{1-h\lambda}}_{\mathrm{Re}(z)} u_n
\end{aligned}$$

We consider Trapezoidal Rule and Midpoint Rule to help with Implicit Euler (for u_{n+1}), where if we apply them to a linear problem, we get the same result:

$$\begin{aligned}
\text{TR: } u_{n+1} &= u_n + \frac{h}{2}\left[f(u_n) + f(u_{n+1})\right] = u_n + \frac{h}{2}\lambda(u_n + u_{n+1}) \\
\text{Midpoint: } u_{n+1} &= u_n + hf\left(\frac{u_n + u_{n+1}}{2}\right) \\
\left(1 - \frac{h}{2}\lambda\right) u_{n+1} &= \left(1 + \frac{h\lambda}{2}\right) u_n \\
\implies u_{n+1} &= \underbrace{\left[\frac{1 + h\lambda/2}{1 - h\lambda/2}\right]}_{\mathrm{Re}(z)} u_n
\end{aligned}$$

Essentially, we can write

$$u_{n+1} = \mathrm{Re}(z) \cdot u_n,$$

where we multiply by the Real part of some $z := h\lambda$. We call $\mathrm{Re}(z)$ the "stability function". Notice Strain writes $R(z)$ instead of $\mathrm{Re}(z)$, but indeed this is simply taking the real part.

Definition: Region of Absolute Stability (RAS) -

A numerical method for $y' = f(y)$ has a <u>Region of absolute stability</u> wherever:

$$\{z : |\mathrm{Re}(z)| \le 1\}$$

For example, in **explicit Euler**, $\mathrm{Re}(z) := 1 + z$, and the region of absolute stability is **within** a unit circle in the complex plane centered around

$(-1,0)$. What this tells us is that if we solve a problem for some given λ and associated angle in the complex plane polar coordinates, the scaling factor h must be small enough to land in the circle.
We already have other methods which are much better, say for instance implicit euler, where

$$\mathrm{Re}(z) = \frac{1}{1-z}, \qquad |\mathrm{Re}(z)| \le 1, \qquad |1-z| \ge 1,$$

and the region for absolute stability is **outside** a unit circle centered about $(1,0)$.
And in the case for Midpoint or Trapezoidal rule, we have:

$$|\mathrm{Re}(z)| = \left|\frac{1+z/2}{1-z/2}\right| \le 1, \qquad |2+z| \le |2-z|,$$

To see that the region of absolute stability is all of the left complex plane, let $z := x + yi$ per usual:

$$\begin{aligned} |2+z|^2 &\le |2-z|^2 \\ (2+x)^2 + y^2 &\le (2-x)^2 + y^2 \\ 4 + 4x + x^2 + y^2 &\le 4 - 4x + x^2 + y^2 \implies x \le 0 \end{aligned}$$

It seems this is the best possible situation (but it's not, because setting $z := iy$ makes the solution oscillate rapidly, and $z := -\infty$). Although this seems to give the perfect region for absolute stability, it has its problems.

Definition: A-stable -

A numerical method is said to be **A-stable** if the Region of Absolute Stability (RAS) includes the **entire left half of the complex plane**.

The reason for this definition is that A-stable methods ought to be OK (good) for stiff problems. It turns out there's a sub-class that performs even better (L-stability).

Definition: L-stable -

A numerical method is **L-stable** if it is A-stable with the additional condition:

$$\lim_{|z|\to\infty} \mathrm{Re}(z) = 0.$$

Theorems: that there is no A-stable multi-step method with $k \ge 2$. Additionally, no explicit Runge-Kutta method is A-stable.
We prove the second. We look at Runge-Kutta methods and write:

$$\begin{aligned} k_i &= f\left(u_n + h\sum a_{i,j}k_j\right) \\ &= \lambda\left(u_n + h\sum a_{i,j}k_j\right) \\ \implies k_i - h\lambda \sum a_{i,j}k_j &= \lambda u_n \\ u_{n+1} &= u_n + \underbrace{h}\sum b_i k_i \end{aligned}$$

What we are doing here is multiplying for each i, k is a vector:

$$\begin{aligned}
[I - h\lambda A]\, k &= \lambda u_n e, \qquad e := [1; 1; \cdots; 1] \\
k &= [I - zA]^{-1} \lambda u_n e \\
&= \lambda u_n [I - zA]^{-1} e \\
u_{n+1} &= u_n + h b^T k \\
&= u_n + \underbrace{h\lambda}_{z} u_n b^T [I - zA]^{-1} e \\
&= \left[1 = z b^T [I - zA]^{-1} e\right] u_n,
\end{aligned}$$

so we proved that the region of absolute stability for a Runge Kutta method can be read off from the Butcher array:

$$R(z) = 1 + z b^T [I - zA]^{-1} e,$$

where for example in Trapezoidal Rule,

$$\begin{bmatrix} 0 & 0 \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

and hence:

$$u_{n+1} = u_n + \frac{h}{2}\left[\underbrace{f(u_n)}_{k_1} + \underbrace{f(u_{n+1})}_{k_2}\right]$$

and hence inverting the matrix $[I - zA]$, we have:

$$\begin{aligned}
[I - zA]^{-1} e &= \begin{bmatrix} 1 & 0 \\ \frac{-z}{2} & 1 - \frac{z}{2} \end{bmatrix}^{-1} e \\
&= \frac{1}{1 - z/2} \begin{bmatrix} 1 - \frac{z}{2} & 0 \\ \frac{z}{2} & 1 \end{bmatrix} [1 \quad 1] \\
&= \frac{1}{1 - z/2} \begin{bmatrix} 1 - z/2 \\ 1 + z/2 \end{bmatrix},
\end{aligned}$$

and hence:

$$\begin{aligned}
R(z) &= 1 + z\frac{1}{1 - z/2}\left[\frac{1}{2}\left(1 - \frac{z}{2}\right) + \frac{1}{2}\left(1 = \frac{z}{2}\right)\right] \\
&= \frac{1 - z/2 + z}{1 - z/2} = \frac{1 + z/2}{1 - z/2} \qquad ,
\end{aligned}$$

which is the correct expression as we derived earlier today.

Break time. After the break we'll apply this to the geometric series (taylor expansion).

Now they taylor series of geometric series on $(I - zA)$ gives:

$$(I - zA)^{-1} = I + zA + z^2A^2 + z^3A^3 + \cdots + z^pA^p,$$

if A is nilpotent and $A^{p+1} = 0$. For example, in an **explicit** Runge-Kutta method, our diagonal values of the matrix A are all zero, so successive powers cause the zeros to 'march down'.

Remark: For any explicit Runge Kutta method, the matrix A is strictly lower triangular, so $A^s = 0$ and $R(z)$ is a polynomial, which can be seen as $\lim_{|z|\to\infty} R(z) = +\infty$. Hence we proved that any **explicit** Runge Kutta method cannot be A-stable.

2.1 Trying for L-stability

Recall that the Trapezoidal rule is second-order accurate, so we want something slightly better. Now let's consider making a a **second-order implicit Runge-Kutta method** which is L-stable. We try **deferred correction** on implicit euler, and maybe it will be L-stable or A-stable.

Try $p = 2$. We first take a step of implicit euler:

$$\begin{aligned} u_{n+1} &= u_n + hf(u_{n+1}) \\ u_{n+2} &= u_{n+1} + hf(u_{n+2}) \\ u(t) &= \cdots u_n, u_{n+1}, u_{n+2} \end{aligned}$$

Next we look at the errors. $e_n := 0$.

$$\begin{aligned} e_{n+1} &= e_n + h\left[f(u_{n+1} + e_{n+1}) - u'(t_{n+1})\right] \\ e_{n+2} &= e_{n+1} + h\left[f(u_{n+2} + e_{n+2}) - u'(t_{n+2})\right] \\ u^1_{n+1} &= u_{n+1} + e_{n+1} \\ u^1_{n+2} &= u_{n+2} + e_{n+2} \\ &= u_n + 2h\left[b_1k_1 + b_2k_2 + b_3k_3 + b_4k_4\right] \end{aligned}$$

Our plan here is to find $|R(z)| \le 1$. We don't feel like doing the tedious algebra, so we fix $h := \lambda$ and vary $\lambda = z$ over some box in the left side of the complex plane. Because we don't know the values of coefficents b_i, we just use our above equation

$$u^1_{n+2} = u_{n+2} + e_{n+2}$$

and assert

$$|u_{n+1}| \le 1 \implies z \in RAS$$

To do this, we write:

$$\begin{aligned} u_{n+1} &= \frac{1}{1-h\lambda}u_n \\ u_{n+2} &= \frac{1}{(1-h\lambda)^2}u_n \\ &\vdots \end{aligned}$$

Notice that our invented method is not a multistep method, as we go from u_n to u_{n+2}, moving two steps at a time.
Alternatively, we can find $\{z : |R(z)| = 1\}$ which is called the 'boundary locus technique'.

3 Nonlinear Stability

Nonlinear stability is a lot simpler than linear stability. In linear stability, we said $y' = \lambda y$, and $|\lambda| < 0$. For assertions or conditions on nonlinear stability, we cannot say that the solution goes to 0 because the exact solution may not be zero. The real question if the exact solutions get closer together? We look at two different trajectories with $1 \leq j \leq p$:

$$\begin{aligned} u_j'(t) &= f_j(u(t)) \\ v_j'(t) &= f_j(v(t)), \end{aligned}$$

where we don't worry about t because we can make these autonomous. Looking at the distance between these, we have:

$$||u(t) - v(t)|| = \sqrt{\sum_j (u_j(t) - v_j(t))^2},$$

where we want the distance to decrease as t increases. We use the euclidean norm because it is simple when we differentiate. We want to assert:

$$\begin{aligned} \frac{d}{dt}||u(t) - v(t)||^2 &= \frac{d}{dt}\sum_j [u_j(t) - v_j(t)]^2 \\ &= \sum_j 2\left(u_j(t) - v_j(t)\right)\left(u_j'(t) - v_j'(t)\right) \\ &= \sum_j 2\left(u_j - v_j\right)\left(f_j(u) - f_j(v)\right) \leq 0, \end{aligned}$$

where we want to assert this is true for all pairwise elements (and all t). This is equivalent to saying:

$$\frac{d}{dt}||u(t) - v(t)||^2 = 2(u - v)^T\left[f(u) - f(v)\right]$$

This all suggests a definition:

Definition: Dissipative -

We say f is dissipative if

$$\left(f(u) - f(v)\right)^T(u - v) \leq 0$$

Many functions are dissipative, such as if we have a positive-definite or negative-definite function.
Our above calculations above prove a theorem:

Theorem 3.1. If function f is dissipative and $u' = f(u), v' = f(v)$, then

$$||u(t) - v(t)||$$

is decreasing.

Our goal is to design a method for which the distance between two different numerical solutions get closer to gether whenever f is dissipative. We call this "contractive" (except we don't have the factor of $\frac{1}{2}$). That is,

$$||u_{n+1} - v_{n+1}|| \leq ||u_n - v_n|| \qquad \text{(contractive)}$$

For example, let us take implicit euler. We know this is a nice L-stable method.

$$\begin{aligned} u_{n+1} &= u_n + hf(u_{n+1}) \\ v_{n+1} &= v_n + hf(v_{n+1}), \end{aligned}$$

and substracting yields

$$v_{n+1} - v_{n+1} = u_n - v_n + h(f(u_{n+1}) - f(v_{n+1}))$$

If we dot both sides of this equation, we have:

$$\begin{aligned} ||u_{n+1} - v_{n+1}||^2 &\leq (u_{n+1})^T(u_n - v_n) + \underbrace{h(u_{n+1} - v_{n+1})^T\,(f(u_{n+1}) - f(v_{n_1}))}_{\leq 0 \text{ if } f \text{ dissipative}} \\ &\leq \underbrace{||u_{n+1} - v_{n+1}||}\,||u_n - v_n|| \end{aligned}$$

$$\implies ||u_{n+1} - v_{n+1}|| \leq ||u_n - v_n||$$

By the Cauchy Schwarz inequality, the dot product of two vectors is less than the product of their lengths, to get the last inequality. We call this B-stable, which brings us to the corresponding definition:

Definition: B-stable -

We say a numerical method is B-stable if

$$||u_{n+1} - v_{n+1}|| \leq ||u_n - v_n|| \qquad \text{("contractive")}$$

whenever f is dissipative.

Dissipative functions f guarantee a specific type of stability in differential equations.

Lecture ends here.

4 Key Results

- Region of Absolute Stability
- Definitions: RAS, A-stable,
- Implicit Euler is first-order accurate
- Trapezoidal/Midpoint Rules are second-order accurate
- Additional conditions can give A-stability or L-stability

5 Office Hours

$$\begin{aligned} y_{n+1} &= y_n + \sum p_j \underline{f_{n-j}} + h\tau_n \\ &= y_n + \int_{t_n}^{t_{n+1}} [y'(s) - p(s)] + \underline{p(s)}\ ds \\ h\tau_n &= \int_{t_n}^{t_{n+1}} \underbrace{y'(s) - p(s)}\ ds \end{aligned}$$

where this error is:

$$\frac{y^{(k+2)}}{(k+1)!}\omega(s).$$

The reason for the mismatch between the derivative order and the denominator factorial is that we are interpolating the derivative!