Section 3
ODE, Linear Multistep methods

1 Motivation

sec:Intro

We come back to the topic of Section 1 of the notes, time stepping methods to compute (approximate) solutions of

$$\dot{x} = f(x) \; . \tag{1}$$ ode

Runge Kutta methods estimate $x(t + \Delta t)$ using only the (approximate) value of $x(t)$. *Linear multistep* methods estimate $x(t + \Delta t)$ using $x(t)$, and earlier values $x(t - \Delta t), \ldots$, and $f(x(t))$, $f(x(t - \Delta t)), \ldots$. The methods are called *linear* because the estimate of $x(t + \Delta t)$ is a linear function of this data. By contrast, the Runge Kutta estimate of $x(t + \Delta t)$ is a nonlinear function of its data $x(t)$. The nonlinearity in Runge Kutta comes from applying the nonlinear function f repeatedly. Runge Kutta methods achieve high order of accuracy using many *stages*, which generate the intermediate values k_j. Linear multistep methods achieve high order of accuracy using previously computed solution values $x_k \approx x(t_k)$, and the corresponding function values $f_k = f(x_k)$.

An advantage of linear multistep methods is that the step from x_n to x_{n+1} requires just one evaluation of f, regardless of the order of accuracy. They achieve a high order of accuracy using already computed older values of x and f. With Runge Kutta methods, the number of f evaluations per time step increases with the order of accuracy. But the use of old values creates problems in some applications, as we will see.

Linear multistep methods have new features that were not present in Runge Kutta methods. One is the possibility of non-convergence through *instability*. A Runge Kutta method that should have order p converges with error of order Δt^p. A linear multistep method can have order p and yet "explode" – go to infinity at a fixed time as $\Delta t \to 0$. Stability is one of the most subtle and technical parts of numerical solution of differential equations. Unstable modes are usually *spurious*. Spurious modes are modes that exist in the numerical approximation but not in the physical system itself. A linear multistep method with r lags has r modes for every physical one (we will see this). Of these, at least $r - 1$ are spurious. Spurious modes ruin a computation if they are unstable. Stability analysis of spurious modes is mathematical, not physical, because spurious modes are not physical.

Another new thing in linear multistep methods is the Lax *stability and consistency* convergence argument. This appeared in a simple way in the convergence proof for Runge Kutta methods. But the version used to prove convergence of linear multistep methods is more general and more like the argument used for convergence proofs with partial differential equations.

2 The methods

A linear multistep method with s lags is defined by the recurrence relation

$$\sum_{j=0}^{s} \alpha_j x_{n+j} = \Delta t \sum_{j=0}^{s} \beta_j f_{n+j} = \Delta t \sum_{j=0}^{s} \beta_j f(x_{n+j}) \ . \tag{2}$$ lmm

This is used to determine x_{n+s} given x_{n+j} for $j < s$. Therefore we require that $\alpha_s \neq 0$. We divide through by α_s to get an equivalent relation with $\alpha_0 = 1$. By convention, we always assume $\alpha_s = 1$. The method is *explicit* if $\beta_s = 0$ and *implicit* if $\beta_s \neq 0$. If the method is explicit, we may take a time step by first computing the new x then computing the new f:

$$\begin{aligned} x_{n+s} &= -\sum_{j=0}^{s-1} \alpha_j x_{n+j} + \Delta t \sum_{j=0}^{s-1} \beta_j f_{n+j} \ , \\ f_{n+s} &= f(x_{n+s}) \ . \end{aligned}$$

If the method is implicit, we may take a time step as follows

$$\begin{aligned} y &= -\sum_{j=0}^{s-1} \alpha_j x_{n+j} + \sum_{j=0}^{s-1} \beta_j f_{n+j} \ , \\ &\text{solve for } x_{n+s} \ x_{n+s} - \Delta t \beta s f(x_{n+s}) = y \ . \end{aligned}$$

This formula assumes that the method has a fixed time step Δt which is used for all steps. For convenience, we assume that time starts at $t = 0$, so

$$t_n = n \Delta t \ .$$

We assume that

$$f_n = f(x_n) \ .$$

To take a time step, you first evaluate x_{n+1} using the linear expression on the right of (2). Then you evaluate $f_{n+1} = f(x_{n+1})$. For implicit methods, the sums on the right runs from $k = -1$ rather than from $k = 0$. This means that the unknown x_{n+1} appears on the right and left sides. In this case, you have to solve a system of equations to find x_{n+1}.

The forward Euler method corresponds to $r = 1$ (one step), and $a_0 = 1$ and $b_0 = 1$. These values in (2) give

$$x_{n+1} = x_n + \Delta t f(x_n) \ .$$

The *leapfrog* scheme has $r = 2$, $a_0 = 0$, $a_1 = 1$, $b_0 = 2$, and $b_1 = 0$. These values give

$$x_{n+1} = x_{n-1} + 2 \Delta t f(x_n) \ . \tag{3}$$ lf

This method is derived from the second order centered difference approximation to $\dot{x}(t_n)$:

$$\dot{x}(t_n) = \frac{x(t_n + \Delta t) - x(t_n - \Delta t)}{2 \Delta t} + O(\Delta t^2) \ .$$

You find the leapfrog scheme (3) by substitutions

$$\begin{aligned} \dot{x}(t_n) &= f(x(t_n)) \approx f(x_n) \ , \\ x(t_n + \Delta t) &= \ x(t_{n+1}) \approx x_{n+1} \ , \\ x(t_n - \Delta t) &= \ x(t_{n-1}) \approx x_{n-1} \ . \end{aligned}$$

We will see that leapfrog is second order accurate, while forward Euler is just first order.

There are several commonly used families of methods, with the order of accuracy depending on the number of lags. The methods in these families set either most of the a_k to zero (Adams and Nyström methods) or most of the b_k to zero (BDF methods). You might think that using the most general formula with all a_k and b_k would lead to higher order of accuracy, and it does. But the *Dahlquist barrier theorem* implies that these methods are unstable. A stable method with r lags cannot have order of accuracy more than r. The Adams and BDF methods already achieve these orders of accuracy.

2.1 Adams methods

Adams methods are derived as approximations to the formula

$$x(t_{n+1}) = x(t_n) + \int_{t_n}^{t_{n+1}} f(x(t))\, dt \ . \tag{4}$$ if

They approximate the integral on the right as the integral of some interpolating polynomial. Different Adams methods result from different interpolation formulas. They all have $a_0 = 1$ and $a_k = 0$ for $k \neq 0$. They use only one lag in x. Higher order Adams methods use higher order interpolation using more old f values. They have more lags, a larger r, by having more $b_k \neq 0$.

The *Adams Bashforth* family approximates $f(x(t))$ by an interpolating polynomial $g_k(t)$ that has degree $r-1$ and satisfies the interpolation conditions

$$g_n(t_{n-k}) = f(x_{n-k}) = f_k \ , \quad \text{for } 0 \leq k \leq r-1. \tag{5}$$ bif

The method is

$$x_{n+1} = x_n + \int_{t_n}^{t_{n+1}} g_n(t)\, dt \ . \tag{6}$$ ab

You figure out the formulas for b_k by using polynomial interpolation formulas (Lagrange or Newton or other) and calculating the interpolation. It is straightforward but can involve a lot of not very interesting calculation.

For $r = 1$, the interpolating polynomial has degree $r - 1 = 0$, so it is a constant. The single interpolation condition is $g_n = f_n$, which determines the constant. The integral in (6) is

$$\int_{t_n}^{t_{n+1}} f_n \, dt = \Delta t f_n \ .$$

Therefore, the method (6) is simply

$$x_{n+1} = x_n + \Delta t f(x_n) \ ,$$

which is the forward Euler method.

The next order Adams Bashforth method has $r = 2$. Here, g_n is a degree $r - 1 = 1$ polynomial (linear) that may be written in the form

$$g_n(t) = \alpha_n + \beta_n (t - t_n) \ .$$

It satisfies the interpolation conditions

$$g_n(t_n) = f_n \ , \ \ g_n(t_{n-1}) = f_{n-1} \ .$$

The first interpolation condition gives

$$\alpha_n = f_n \ .$$

The second interpolation condition then becomes

$$f_n + \beta_n (t_{n-1} - t_n) = f_{n-1} \ .$$

Therefore

$$\beta_n = \frac{f_n - f_{n-1}}{t_n - t_{n-1}} = \frac{f_n - f_{n-1}}{\Delta t} \ .$$

You get

$$g_n(t) = f_n + \frac{f_n - f_{n-1}}{\Delta t} (t - t_n) \ .$$

You should[1] recognize this as the Newton form of the interpolating polynomial. Finally,

$$\begin{aligned} \int_{t_n}^{t_n + \Delta t} \left(f_n + \frac{f_n - f_{n-1}}{\Delta t} (t - t_n) \right) dt &= \Delta t f_n + \frac{\Delta t^2}{2} \left(f_n + \frac{f_n - f_{n-1}}{\Delta t} \right) \\ &= \Delta t \left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right) \ . \end{aligned}$$

The second order Adams Bashforth method is

$$x_{n+1} = x_n + \Delta t \left(\frac{3}{2} f(x_n) - \frac{1}{2} f(x_{n-1}) \right) \ .$$

This fits the general linear multistep framework (2) with

$$\begin{aligned} r &= 2 \\ a_0 &= 1 \ , \ \ a_1 = 0 \\ b_0 &= \frac{3}{2} \ , \ \ b_1 = -\frac{1}{2} \ . \end{aligned}$$

[1] If you are not familiar with the Newton form of the interpolating polynomial, the elegant discussion in *Numerical Methods* by G. Dahlquist and Å. Björk is recommended enthusiastically.

We will see that this method is stable and second order accurate. The Adams Bashforth method of order r is stable (this will be obvious) and has order of accuracy r (this is an exercise, but is also "obvious" once you've seen it).

Adams Moulton methods are implicit methods where $g_n(t)$ is the polynomial that interpolates also $f_{n+1} = f(x_{n+1})$ at time t_{n+1}. The lowest order Adams Moulton method takes g_n to be the constant "polynomial" that interpolates at time t_{n+1}, so $g_n(t) = f(x_{n+1})$ for all t. The Adams integral is

$$\int_{t_n}^{t_{n+1}} g_n(t)\,dt = \Delta t f(x_{n+1}) \; .$$

The method is

$$x_{n+1} = x_n + \Delta t f(x_{n+1}) \; . \tag{7}$$ be

This is the backward Euler method, which is first order accurate. The backward Euler time step algorithm to calculate x_{n+1} is:

$$\text{find } x_{n+1} \text{ so that} \quad x_{n+1} - \Delta t f(x_{n+1}) = x_n \; .$$

For the second order Adams Moulton method, $g_n(t)$ is the linear polynomial that interpolates at times t_n and t_{n+1}. As before, the interpolating linear polynomial may be written in the form $g_n(t) = \alpha_n + \beta_n(t - t_n)$. The interpolation condition $g_n(t_n) = f_n$ gives $\alpha_n = f_n$, as before. The interpolation condition $g_n(t_{n+1}) = f(x_{n+1})$ gives

$$\beta_n = \frac{f(x_{n+1}) - f_n}{\Delta t} \; .$$

The Adams integral is

$$\begin{aligned} \int_{t_n}^{t_{n+1}} g_n(t)\,dt &= \Delta t f_n + \frac{\Delta t^2}{2} \frac{f(x_{n+1}) - f_n}{\Delta t} \\ &= \Delta t \left(\frac{1}{2} f(x_{n+1}) + \frac{1}{2} f_n \right) \; . \end{aligned}$$

The second order Adams Moulton method is

$$x_{n+1} = x_n + \frac{\Delta t}{2} f(x_{n+1}) + \frac{\Delta t}{2} f_n \; . \tag{8}$$ am2

To take a time step, the computer must

$$\text{find } x_{n+1} \text{ so that} \quad x_{n+1} - \frac{\Delta t}{2} f(x_{n+1}) = x_n + \frac{\Delta t}{2} f_n \; .$$

This method is sometimes called the *trapezoid rule*, because it comes from the trapezoid rule integral formula

$$\int_{t_n}^{t_{n+1}} f(x(t))\,dt \approx \frac{\Delta t}{2} \left(f(x(t_{n+1})) + f(x(t_n)) \right) \; .$$

The method may be written in a symmetrical way as

$$\frac{x_{n+1} - x_n}{\Delta t} = \frac{1}{2}\left(f(x_{n+1}) + f(x_n)\right) \ . \tag{9}$$ tr

We will soon see that the symmetry makes the second order accuracy obvious. The left side is a second order accurate approximation to $\dot{x}(t_n + \Delta t/2)$, the derivative at the midpoint in time. The right side is a second order approximation to $f(x(t_n + \Delta t/2))$. Therefore the differential equation $\dot{x} = f(x)$ is satisfied to second order at time $t_n + \Delta t/2$.

2.2 Differentiation formulas

These methods achieve high order of accuracy using high order accurate finite difference approximations to $\dot{x}$. For example, consider the first order one sided approximation

$$\dot{x}(t) = \frac{x(t + \Delta t) - x(t)}{\Delta t} + O(\Delta t) \ .$$

If we take $t = t_n$, then $t + \Delta t = t_{n+1}$ and this becomes

$$\dot{x}(t_n) \approx \frac{x_{n+1} - x_n}{\Delta t} \ .$$

The differential equation turns into the approximation

$$\frac{x_{n+1} - x_n}{\Delta t} = f(x_n) \ .$$

This is yet another derivation of the forward Euler method.

The second order centered approximation to $\dot{x}(t)$ is

$$\dot{x}(t) = \frac{x(t + \Delta t) - x(t - \Delta t)}{2\Delta t} + O(\Delta t^2) \ .$$

This leads to the approximation

$$\frac{x_{n+1} - x_{n-1}}{2\Delta t} = f(x_n) \ .$$

This is the leap frog method (3). It is second order accurate. lf

The third order accurate method of this type uses a third order finite difference approximation to $\dot{x}(t_n)$. We get an explicit method if the approximation to $\dot{x}(t)$ uses the values $x(t + \Delta t)$, $x(t)$, $x(t - \Delta t)$ and $x(t - 2\Delta t)$. Some algebra will turn this into a formula for x_{n+1} in terms of x_n, x_{n-1}, x_{n-2}, and f_n. Calculating the coefficients is straightforward but possibly time consuming. The method has $r = 3$, non-zero a_0, a_1, and a_2, but $b_1 = 0$ and $b_1 = b_2 = 0$.

The *backward differentiation formula*, or *BDF*, methods are more commonly used methods of this type. They use $f(x_{n+1})$ instead of $f(x_n)$. In other words, they apply the differential equation at time t_{n+1} rather than at time t_n. This

makes the methods implicit, which is the reason for using them. The first order two point backward differentiation formula is

$$\dot{x}(t) = \frac{x(t) - x(t - \Delta t)}{\Delta t} + O(\Delta t) \ .$$

This implies that if $\dot{x}(t) = f(x(t))$, then

$$\frac{x(t_{n+1}) - x(t_n)}{\Delta t} = f(x(t_{n+1})) + O(\Delta t) \ . \tag{10}$$ ber

The time stepping method is derived from this by replacing $x(t_n)$ with the approximation x_n and ignoring the $O(\Delta t)$ error term:

$$\frac{x_{n+1} - x_n}{\Delta t} = f(x_{n+1}) \ .$$

This is yet another way to derive the backward Euler method (7).

The higher order BDF methods are based on higher order one sided difference approximations. The second order method uses the three point second order approximation

$$\dot{x}(t) = \frac{\alpha x(t) + \beta x(t - \Delta t) + \gamma x(t - 2\Delta t)}{\Delta t} + O(\Delta t^2) \ . \tag{11}$$ 3pt

We find the coefficients using elementary numerical analysis – Taylor series calculations. The notation is simplified by leaving out the argument t, so x means $x(t)$, $\dot{x}$ means $\dot{x}(t)$, etc:

$$\begin{aligned}
&\frac{\alpha x(t) + \beta x(t - \Delta t) + \gamma x(t - 2\Delta t)}{\Delta t} \\
=&\frac{\alpha x + \beta \left(x - \Delta t \dot{x} + \frac{1}{2}\Delta t^2 \ddot{x} + O(\Delta t^3) \right) + \gamma \left(x - 2\Delta t \dot{x} + \frac{4}{2}\Delta t^2 \ddot{x} + O(\Delta t^3) \right)}{\Delta t} \\
=&\frac{1}{\Delta t} \left(\alpha + \beta + \gamma \right) + \dot{x} \left(-\beta - 2\gamma \right) + \Delta t \ddot{x} \left(\frac{1}{2}\beta + 2\gamma \right) + O(\Delta t^2) \ .
\end{aligned}$$

This achieves the $O(\Delta t^2)$ approximation rate in (11) if the following equations are satisfied:

$$\begin{aligned}
0 &= \alpha + \beta + \gamma && (a) \\
1 &= -\beta - 2\gamma && (b) \\
0 &= \frac{1}{2}\beta + 2\gamma && (c)
\end{aligned}$$

Equation (c) is satisfied if $\beta = -4\gamma$. Equation (b) then becomes $1 = 2\gamma$, or $\gamma = \frac{1}{2}$ and then $\beta = -2$. Equation (a) then reduces to $0 = \alpha - 2 + \frac{1}{2}$, or $\alpha = \frac{3}{2}$. The finite difference approximation (11) is now

$$\dot{x}(t) = \frac{\frac{3}{2}x(t) - 2x(t - \Delta t) + \frac{1}{2}x(t - 2\Delta t)}{\Delta t} + O(\Delta t^2) \ . \tag{12}$$ 3ptn

With the differential equation applied at time t_{n+1}, this gives

$$\frac{\frac{3}{2}x(t_{n+1}) - 2x(t_n) + \frac{1}{2}x(t_{n-1})}{\Delta t} = f(x(t_{n+1})) + O(\Delta t^2) \; .$$

This is equivalent to

$$x(t_{n+1}) - \frac{2}{3}\Delta t f(x(t_{n+1})) = \frac{4}{3}x(t_n) - \frac{1}{3}x(t_{n-1}) + O(\Delta t^3) \; .$$

The second order three step BDF method based on this formula is

$$x_{n+1} - \frac{2}{3}\Delta t f(x_{n+1}) = \frac{4}{3}x_n - \frac{1}{3}x_{n-1} \; . \tag{13}$$ bdf2

The preceding paragraph illustrates the derivation of new computational methods. It starts with a general principle, then descends to a mess of arithmetical calculation. The final method (13) has numerical coefficients ($-\frac{2}{3}$, etc.) that have no physical explanation. But the method with this extra complexity and these specific coefficients is much more accurate and useful than the simple "physical" method of the same type (7).

3 Order of accuracy

The order of accuracy conditions for linear multistep methods are simpler than the conditions for Runge Kutta methods. We will see that it is easier to create families of high order linear multistep methods of specific kinds, such as Adams methods (α_k mostly zero), and BDF formulas ($\beta_k = 0$ for $k < r$).

If a method is stable, its *order of accuracy* is determined by its *local truncation error*, or *residual*. The order of accuracy is the largest exponent p so that (with technical conditions added below)

$$|x_n - x(t_n)| \leq C_t \Delta t^p \quad \text{if } t_n \leq t \; . \tag{14}$$ oa

The left side is the difference between the computed approximation and the actual solution. The right side is order Δt^p. The constant C_t can grow with t, but it provides an error bound that holds for all times $t_n \leq t$. For a fixed t, it is possible that $n \to \infty$ as $\Delta t \to 0$ with $t_n \leq t$. That is, the bound should hold at a specified physical time, not a specified number of time steps. The number of time steps needed to reach a given physical time goes to infinity as $\Delta t \to 0$.

The definition and reasoning around the residual will be familiar from Runge Kutta methods. The residual is the amount by which the actual solution to the differential equation fails to satisfy the difference equations. The actual solution at time t_n is $x(t_n)$. We define the residual, r_n, with a factor of Δt taken out. For linear multistep methods (2), r_n is defined by plugging the exact values $x(t_j)$ and $f(x(t_{n+j}))$ into the time step formula (2):

$$\sum_{j=0}^{s} \alpha_j x(t_{n+j}) = \Delta t \sum_{j=0}^{s} \beta_j f(x(t_{n+j})) + \Delta t \, r_n \; . \tag{15}$$ lmmr

We expand x and f about t_n with Taylor series:

$$\begin{aligned} x(t_{n+j}) &= x(t_n + j\Delta t) \\ &= x(t_n) + j\Delta t \dot{x}(t_n) + \tfrac{1}{2} j^2 \Delta t^2 \ddot{x}(t_n) + \cdots \end{aligned}$$

At this point, when we were doing Runge Kutta methods, we used $\dot{x} = f(x)$ to express $\ddot{x}$ in terms of x and f. Here, it is better to express f and its time derivatives as time derivatives of $x(t)$, so we use $f(x(t)) = \dot{x}(t)$, $\frac{d}{dt} f(x(t)) = \frac{d}{dt}\dot{x}(t) = \ddot{x}(t)$, etc: Instead, we write the derivatives of f in terms of derivatives of x. That is the last line in this calculation:

$$\begin{aligned} f(x(t_{n+j})) &= f(x(t_n + j\Delta t)) \\ &= f(x(t_n)) + \Delta t \frac{d}{dt} f(x(t_n)) + \tfrac{1}{2} j^2 \Delta t^2 \frac{d^2}{dt^2} f(x(t_n)) + \cdots \\ &= \dot{x}(t_n) + j\Delta t \ddot{x}(t_n) + \tfrac{1}{2} j^2 \Delta t^2 \frac{d^3}{dt^3} x(t_n) + \cdots \ . \end{aligned}$$

We substitute these expansions into the time step formula (]reflmmr) and collect terms order by order to find the accuracy conditions. The largest terms are the ones without Δt (they have Δt^0). Collecting these gives

$$\sum_{j=0}^{s} \alpha_j x(t_n) = 0 \ .$$

The condition on the linear multistep coefficients is

$$\sum_{j=0}^{s} \alpha_j = 0 \ . \tag{16}$$ lmm0

Next we calculate and equate the terms of order Δt. The residual term $\Delta t r_n$ is left out because r_n itself is supposed to be at least order Δt:

$$\sum_{j=0}^{s} j\Delta t \alpha_j \dot{x}(t_n) = \Delta t \sum_{j=0}^{s} \beta_j \dot{x}(t_n) \ .$$

It is remarkable here that we have the same $\dot{x}(t_n)$ throughout, which cancels. The condition on the coefficients is

$$\sum_{j=0}^{s} j\alpha_j = \sum_{j=0}^{s} \beta_j \ . \tag{17}$$ lmm1

If these two conditions (16) and (17) are satisfied, then the linear multistep method is at least first order accurate in the sense that the residual is order Δt or smaller.

If we assume that r_n is order Δt^2 or smaller and equate terms of order Δt, we get the condition for second order accuracy. Keeping all Δt^2 terms gives

$$\sum_{j=0}^{s} \tfrac{1}{2} j^2 \alpha_j \Delta t^2 \ddot{x}(t_n) = \Delta t \sum_{j=0}^{s} \beta_j j \Delta t \ddot{x}(t_n) \; .$$

The condition for second order accuracy is

$$\tfrac{1}{2} \sum_{j=0}^{s} j^2 \alpha_j = \sum_{j=0}^{s} j \beta_j \; . \tag{18}$$ lmm2

The terms of order Δt^k are (check this please)

$$\frac{1}{k!} \Delta t^k \sum_{j=0}^{s} j^k \alpha_j \frac{d^k}{dt^k} x(t_n) = \Delta t \frac{1}{(k-1)!} \sum_{j=0}^{s} j^{k-1} \Delta t^{k-1} \beta_j \frac{d^k}{dt^k} x(t_n) \; .$$

The resulting accuracy condition is

$$\frac{1}{k} \sum_{j=0}^{s} j^k \alpha_j = \sum_{j=0}^{s} j^{k-1} \beta_j \; . \tag{19}$$ lmmk

The conclusion is that the linear multistep method has order of accuracy p if (16) and the conditions (19) are satisfied up to $k = p$. You will see (exercises) that these conditions are easy to apply.

From here on we assume that $\alpha_s = 0$. If $\alpha_s \neq 0$, we can get an equivalent formula by dividing through by α_s. It's reasonable to assume that $\alpha_s \neq 0$ because the point of a linear multistep method is to compute x_{n+s} as a function of previously computed stuff.

There is an elegant formulation of the accuracy conditions in terms of two associated polynomials (note that $\alpha_s = 1$ as we said it would be)

$$\rho(z) = z^s + \alpha_{s-1} z^{s-1} + \cdots + \alpha_1 z + \alpha_0 \tag{20}$$ rho

$$\sigma(z) = \beta^s z^s + \cdots + \beta_0 \; . \tag{21}$$ sig

The accuracy conditions are equivalent to

$$\rho(e^{\Delta t}) = \Delta t \sigma(e^{\Delta t}) + O(\Delta t^{p+1}) \; . \tag{22}$$ rsp

For example, if you set $\Delta t = 0$, you get the condition $\rho(1) = 0$, which is the same as the first accuracy condition (16). You get the rest of the conditions (19) by expanding the exponential. If $z = e^{\Delta t}$, then

$$z^j = 1 + j\Delta t + \tfrac{1}{2} j^2 \Delta t^2 + \cdots \; .$$

The equation (22) becomes

$$\sum_{j=0}^{s} \alpha_j \left(1 + j\Delta t + \tfrac{1}{2} j^2 \Delta t^2 + \cdots \right) = \Delta t \sum_{j=0}^{s} \beta_j \left(1 + j\Delta t + \tfrac{1}{2} j^2 \Delta t^2 + \cdots \right)$$

$$\sum_{j=0}^{s} \alpha_j + \Delta t \sum_{j=0}^{s} j\alpha_j + \tfrac{1}{2} \Delta t^2 \sum_{j=0}^{s} j^2 \alpha_j + \cdots = \Delta t \sum_{j=0}^{s} \beta_j + \Delta t^2 \sum_{j=0}^{s} j \beta_j + \cdots$$

Equating coefficients of Δt^k on both sides gives the accuracy conditions (19).

4 Stability

If a fancy high order method does not work, it is likely because the method is unstable. Exercise 2 has an example, which is an explicit third order lag $s = 2$ linear multistep method. The $s = 2$ Adams Bashforth method is only second order. The "too good to be true" third order method does not work because it is unstable.

The two stability concepts in this section are *zero stability* and *stability region*. A method is *zero stable* if it is stable (precise definition below) for the differential equation $\dot{x} = 0$. This is $\dot{x} = f(x)$ in one dimension with $f(x) = 0$. The convergence theorem of Dahlquist is that if a method is zero stable then it converges with accuracy p as $\Delta t \to 0$ up to a fixed time T. We proved this for Runge Kutta methods, as any one step method is zero stable. For $f = 0$, no matter how many stages there are you get $x_{n+1} = x_n$. Linear multi-step methods are more complicated if many of the α_j are not zero. Zero stability depends on the α_j but not the β_j.

The *stability region* of a linear multi-step method is the set of $\mu \in \mathbb{C}$ for which the method is stable with $\Delta t = 1$ for

$$\dot{x} = \mu x \; . \tag{23}$$

As a joke, people sometimes call the test equation (23) the *Dahlquist equation*. If you apply the linear multistep method with $\Delta t = 0$ to the Dahlquist equation, you get the *linear recurrence relation*

$$\sum_{j=0}^{s} (\alpha_j - \mu\beta_j) \, x_{n+j} \; . \tag{24}$$

The stability region is a set $E \subset \mathbb{C}$. A complex number μ is in the stability region if the recurrence relation (24) is stable. It is necessary to consider complex numbers μ even when the differential equation you actually want to solve (not the Dahlquist equation) is entirely real. A method is zero stable if $0 \in E$.

Stability regions are important for designing time stepping methods for partial differential equations. A linear partial differential equation for a time dependent field $u(x,t)$ may be written abstractly as $\partial_t u = Lu$, where L is a differential operator that involves x derivatives but not t derivatives. A semi-discrete approximation is a large ODE system $\dot{U} = AU$. You get a fully discrete approximation by applying a time stepping method to the semi-discrete ODE system. The dimension of U and the matrix A depend on the space discretization size Δx. If the time stepping method is zero stable, it will converge to the solution of $\dot{U} = AU$, any fixed Δx, as $\Delta t \to 0$. The trouble is (more detail below) that if Δx is small than Δt may have to be very very small in order to have a stable computation. The zero stability analysis is a sort of "double limit" where $\Delta x \to 0$ on the outside and $\Delta t \to 0$ on the inside. The stability

region analysis can tell us a relation between Δt and Δx as they go to zero at the same time so that the numerical solution converges to the time dependent field $u(x,t)$.

A general linear recurrence relation takes the form

$$\sum_{j=0}^{s} \gamma_j x_{n+j} = 0 \; . \tag{25}$$ glrr

We assume that $\gamma_s \neq 0$, so that the recurrence relation is "genuinely" of length s. Given $\gamma_s \neq 0$ we may assume $\gamma_s = 1$ when convenient. The coefficients γ_j and the sequence x_n may be complex. The x_j may be complex even when the γ_j are real. We suppose that the sequence x_n is defined for $n \geq 0$. The first numbers $x_0, x_1, \ldots, x_{s-1}$ are the initial conditions. Once these are given, then x_s, x_{s+1}, and the rest are determined by the recurrence relation. For example, if we put $n = 0$ in (25), we can solve for x_s in terms of the earlier values. The space of infinite sequences that satisfy (25) is a linear vector space of dimension s.

A linear recurrence is *stable* if every solution (the infinite sequence $x_0, x_1, \ldots$) is bounded. More precisely, for any sequence x_n defined for $n \geq 0$ that satisfies (25) there is a constant C so that $|x_n| \leq C$ for all n. In this definition we do not say how C depends on the sequence, only that each sequence has a C. The *characteristic polynomial* for the recurrence relation is

$$f(z) = \sum_{j=0}^{s} \gamma_j z^j \; . \tag{26}$$ cp

The basic stability theorem is: The recurrence relation is stable if and only if its characteristic polynomial satisfies the *root condition*. A *root* is a complex number z so that $f(z) = 0$. The root condition has two parts. First, all roots must be in the unit disk of the complex plane: if $f(z) = 0$ then $|z| \leq 1$. Second, a root on the unit circle must be simple: if $|z| = 1$ and $f(z) = 0$, then $f'(z) \neq 0$. If $f(z_*) = 0$ and $f'(z_*) = 0$ than z_* is (at least) a *double root*. That is because we can write $f(z) = (z - z_*)^2 g(z)$ where $g(z)$ is a polynomial of degree $s - 2$. If z_* is a simple root, then $f(z) = (z - z_*)g(z)$ where $g(z_*) \neq 0$.

For example, consider the recurrence

$$x_{n+2} - \frac{3}{2} x_{n+1} + \frac{1}{2} x_n = 0 \; .$$

The characteristic polynomial is $f(z) = z^2 - \frac{3}{2}z + \frac{1}{2} = (z-1)(z - \frac{1}{2}$. The roots are $z = 1$ and $z = \frac{1}{2}$. Both roots are in the unit disk and the root on the unit circle is simple. The recurrence

$$x_{n+2} + \frac{1}{4} x_n = 0$$

has characteristic polynomial $f(z) = z^2 + \frac{1}{4}$. The roots are $z = \frac{i}{2}$ and $z = \frac{-i}{2}$. These have $|z| = \frac{1}{2} \leq 1$, so this recurrence satisfies the root condition. The

recurrence

$$x_{n+2} + x_{n+1} - 2x_n = 0$$

has characteristic polynomial $f(z) = z^2 + z = 2$. The roots are $z = 1$, which is fine, and $z = -2$, which is outside the unit disk. This does not satisfy the root condition. The recurrence

$$x_{n+3} - x_{n+2} - x_{n+1} + x_n = 0$$

has characteristic polynomial $f(z) = z^3 - z^2 - z + 1 = (z-1)^2(z+1)$. This is a simple root at $z = -1$ and a double root at $z = 1$. Because of the double root, this recurrence does not satisfy the root condition. The sequence $x_n = n$ satisfies the recurrence relation, and is not bounded.

Here is a proof of the stability theorem (stability $\Longleftrightarrow$ root condition). But this stability theorem is not enough to prove the Dahlquist convergence theorem that is our actual goal. For that, we will use the longer proof of the stronger theorem in the next subsection. We already said that the space of solution sequences is a complex vector space $\mathcal{S}$ of dimension s. We will show two things. First, if the root condition is violated then there is a sequence $x \in \mathcal{S}$ with $|x_n| \to \infty$ as $n \to \infty$ (an unbounded sequence). Second, if the root condition is satisfied then there is a basis $v_\alpha \in \mathcal{S}$ consisting of bounded sequences.

Both parts are proven by showing that the solution of the recurrence relation is expressed in terms of roots of the characteristic polynomial. If $f(z) = 0$ then $x_n = z^n$ satisfies the recurrence (25). If you substitute in $x_n = z^n$, you can calculate

$$\sum_{j=0}^{s} \gamma_j z^{n+j} = z^n \left(\sum_{j=0}^{s} \gamma_j z^j \right) = z^n f(z) = 0 \; .$$

If $|z| = r > 1$, then this solution sequence has $|x_n| = |z^n| = r^n$. Thus, if z is a root outside the unit disk, the corresponding sequence grows exponentially with n. Now suppose z is a root of multiplicity two or higher, so $f(z) = 0$ and $f'(z) = 0$. Then the derivative sequence $x_n = nz^{n-1}$ satisfies the recurrence relation. You can see this with a calculation:

$$\begin{aligned}
\sum_{j=0}^{s} \gamma_j x_{n+j} &= \sum_{j=0}^{s} \gamma_j (n+j) z^{n+j-1} \\
&= n \sum_{j=0}^{s} \gamma_j z^{n+j-1} + \sum_{j=0}^{s} \gamma_j j z^{n+j-1} \\
&= n z^{n-1} \sum_{j=0}^{s} \gamma_j z^j + z^n \sum_{j=0}^{s} \gamma_j j z^{j-1} \\
&= n z^{n-1} f(z) + z^n f'(z) \\
&= 0 \; .
\end{aligned}$$

If z is a root on the unit circle with $f'(z) = 0$, then this sequence has $|x_n| = n \left| z^{n-1} \right| = n$, which is unbounded.

For the second part (root condition $\Longrightarrow$ stability), start by noting that if $|z| < 1$ then $x_n = n^k z^n$ is bounded, and if $|z| = 1$ then $x_n = z^n$ is bounded. Let z_α be the roots, and suppose z_k has multiplicity k_α. There are k_α solution sequences for z_α, which correspond to derivatives

$$x_n = \left(\frac{d}{dz}\right)^k z^n \ , \quad k = 0, \ldots, m-1; . \tag{27}$$ pp

For $m = 1$, this is the sequence we already used $x_n = nz^{n-1}$. The sequence for $m = 2$ is $x_n = n(n-1)z^{n-2}$. If $f(z) = 0$ and $f'(z) = 0$ and $f''(z) = 0$, then this satisfies the recurrence. You can see this through the calculation

$$\begin{aligned}
\sum_{j=0}^{s} \gamma_j \left(\frac{d}{dz}\right)^2 z^{n+j} &= \left(\frac{d}{dz}\right)^2 \sum_{j=0}^{s} \gamma_j z^{n+j} \\
&= \left(\frac{d}{dz}\right)^2 \left(z^n \sum_{j=0}^{s} \gamma_j z^j \right) \\
&= \left(\frac{d}{dz}\right)^2 (z^n f(z)) \\
&= n(n-1)z^{n-2} f(z) + 2nz^{n-1} f'(z) + z^n f''(z) \ .
\end{aligned}$$

With our hypotheses, the terms on the right are all zero. This argument works for higher derivatives. We conclude that if z is a root of multiplicity m, then there are m solutions of the form (27). In this way, we construct s solutions for a linear recurrence of order s. These particular solutions are all bounded if the root condition is satisfied. In order to prove that all solutions are bounded, we need to show that the particular solutions are linearly independent and that a linear combination of bounded solutions is a bounded solution.

The linear combination part is easy. Suppose $v_\alpha \in \mathcal{S}$ for $\alpha = 1, \ldots, s$ are bounded solutions and

$$x_n = \sum_{\alpha=1}^{s} a_\alpha v_{\alpha,n} \ .$$

Suppose also that $|v_{\alpha,n}| \le C_\alpha$ for all $n \ge 0$. Then (everything is complex, including the coefficients a_α)

$$|x_n| \le \sum_{\alpha=1}^{s} |a_\alpha| \, C_\alpha = C \ , \quad \text{for all } n \ge 0 \ .$$

The sequence x_n is also bounded.

The linear independence part is easy too, but involves more machinery. One way to see it starts from the Vandermonde determinant formula for any s com-

plex numbers

$$\det \begin{pmatrix} 1 & z_1 & \cdots & z_1^{s-1} \\ 1 & z_2 & & z_2^{s-1} \\ \vdots & \vdots & & \vdots \\ 1 & z_s & & z_s^{s-1} \end{pmatrix} = V(z_1, \ldots, z_s) = \prod_{j<k} (z_k - z_j) \ .$$

The matrix on the left is the Vandermonde matrix. There are many proofs of this formula (see e.g., wikipedia or any good textbook on abstract algebra). One way starts by noting that the determinant is a polynomial in the variables z_k (by general formula for determinant of a matrix). If we fix $z_2, \ldots, z_s$, then we can look at V as a polynomial in the single variable z_1. This polynomial has degree $s-1$ and is equal to zero when $z_1 = z_k$ for any $k > 1$, because the Vandermonde matrix is singular of two rows are equal. Therefore, we may write

$$V(z_1, \ldots, z_s) = \left(\prod_{1<k} (z_k - z_1) \right) W(z_2, \ldots, z_s) \ .$$

We can apply this reasoning also to see that

$$W(z_2, \ldots z_s) = \left(\prod_{2<k} (z_k - z_2) \right) X(z_3, \ldots, z_s)$$
$$V(z_1, \ldots, z_s) = \left(\prod_{1<k} (z_k - z_1) \right) \left(\prod_{2<k} (z_k - z_2) \right) X(z_3, \ldots, z_s) \ .$$

You can continue in this way to prove the Vandermonde formula up to a constant, which is not zero. There are various ways of showing the constant is 1, but that doesn't matter for our purpose here. Our purpose is to show that if the z_k are distinct, then the sequences $v_{k,n} = z_k^n$ are linearly independent.

What about the case of a double root? For example, suppose $z_1 = z_2$ is a double root and the other z_k are distinct. We want to show that the sequences $v_{1,n} = z_1^n$, and $v_{2,n} = n z_1^{n-1}$, and $v_{k,n} = z_k^n$ for $k > 2$ are s linearly independent sequences. The corresponding determinant is

$$\det \begin{pmatrix} 1 & z_1 & \cdots & z_1^{s-1} \\ 0 & 1 & 2z_1 & (s-1) z_1^{s-2} \\ 1 & z_3 & \cdots & z_3^{s-1} \\ \vdots & \vdots & & \vdots \\ 1 & z_s & & z_s^{s-1} \end{pmatrix} = D(z_1, z_3, \ldots, z_s) \ .$$

Why is this determinant not equal to zero? The Vandermonde determinant $V(z_1, z_2, \ldots)$ is a polynomial of degree $s-1$ in the variable z_2. This polynomial has roots at $z_2 = z_1$ and $z_2 = z_k$ for $k > 2$. By hypothesis, these $s-1$ roots are distinct. Therefore they are simple roots. Because the roots are simple, the

derivative does not vanish at the root:

$$\partial_{z_2} V(z_1, z_2, \ldots, z_s)\Big|_{z_2=z_1} \neq 0 \; .$$

But the determinant D is exactly the left side here, so $D \neq 0$ if $z_1 \neq z_k$ for $k > 2$ and the z_k are distinct. I think this is as much about Vandermonde determinants and algebra as most numerical analysis people have the patience for.

4.1 Root/eigenvalue stability and Lyapunov stability

This section is pretty technical. Unlike the Vandermonde technicalities above, these are central to big parts of theoretical numerical analysis. So the material in this subsection is worth a lot of time and effort.

There are two things here. The first is to reformulate and generalize the discussion of one variable recurrence relations and characteristic polynomials to matrix/vector recurrences with just one lag. From the scalar sequence x_n we define a sequence of vectors $\vec{x}_n \in \mathbb{C}^s$ and an $s \times s$ complex matrix A so that

$$\vec{x}_{n+1} = A\vec{x}_n \; . \tag{28}$$ mr

The sequence is bounded if there is a C so that

$$\|\vec{x}_n\| \leq C \; , \quad \text{for all } n > 0 \; . \tag{29}$$ vsb

The root condition for the characteristic polynomial is generalized to a non-degeneracy condition for eigenvalues. Suppose z is an eigenvalue of A and $\vec{v} \neq 0$ is an eigenvector:

$$z\vec{v} = A\vec{v} \; .$$

The *root condition* is that either $|z| < 1$ or $|z| = 1$ and there is no *Jordan structure* (defined below) for z.

One way to understand the "no Jordan structure" condition is to look at a Jordan block with eigenvalue z

$$A = \begin{pmatrix} z & t \\ 0 & z \end{pmatrix}$$

Consider the vectors

$$v_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad , \quad v_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \; .$$

Then $Av_1 = zv_1$ so v_1 is a genuine eigenvector with eigenvalue z. But $Av_2 = zv_2 + tv_1$. If $t \neq 0$, then v_2 is a "generalized" eigenvector (not an eigenvector, but related to one) with eigenvalue z. Only if $t = 0$ is v_2 a true eigenvector. If $t \neq 0$ we say that A has Jordan structure. There is a two dimensional space. The eigenvalue z has multiplicity two. But there is only one linearly independent eigenvector.

Here is a precise statement of the "root condition" for a matrix/vector recurrence (28). The characteristic polynomial is

$$f(z) = \det(A - zI) \ .$$

Let $z \in \mathbb{C}$ be an eigenvector of A with multiplicity m. Then either $|z| < 1$ or there are m linearly independent eigenvectors of A with eigenvalue z. If A and $\vec{x}_n$ comes from a scalar recurrence relation (25), then any root of multiplicity $m > 1$ has Jordan structure, so the two root conditions are the same. But other matrix recurrences can satisfy the root condition even with multiplicity $m > 1$. When we talk about vector recurrences we often drop the arrow and write $x_n \in \mathbb{C}^s$ instead of $\vec{x}_n \in \mathbb{C}^s$.

The second thing in this subsection, the hard thing, is the *symmetrizer* matrix M. This is an $s \times s$ positive definite hermitian matrix so that if $x_{n+1} = Ax_n$, then

$$x_{n+1}^* M x_{n+1} \leq x_n^* M x_n \ . \tag{30}$$

A complex matrix is *hermitian* if it is equal to its complex conjugate:

$$M^* = M \ , \quad M_{jk} = \overline{M}_{kj} \ .$$

If M is hermitian, then x^*Mx is real for any complex vector $x \in \mathbb{C}^s$. A matrix (hermitian or not) is *positive definite* if

$$x^*Mx > 0 \ , \ \text{ if } x \neq 0 \ .$$

This subsection pretty technical. An *efficient* discussion is one that uses a small number of words. The discussion here is not efficient in that sense. Instead it wanders toward the conclusion so that you can see how you can think about this material. When you're done, you will see that you can construct the symmetrizer M directly. But how would anyone think of that construction without playing with examples?

Roughly speaking, *stability* means that the sequence x_n remains bounded[2] as $n \to \infty$. A recurrence relation can be stable in this sense but still allow sequences to grow. That growth must *saturate*, which means there is a limit to how much growth there can be. *Lyapunov stability* is the seemingly stronger fact that there is a *Lyapunov function* that does not grow at all and controls the size of x_n. The *root condition* is what you need to know that solutions of recurrences remain bounded. If a recurrence satisfies the root condition, then (we will see) the sequence has a *Lyapunov function*. The convergence proof for linear multistep methods uses the Lyapunov function to show that even with the scheme is stable (in some sense) even with $\Delta t \beta_j f(x_{n+j})$ terms that are not part of the definition of zero stability. If you start with a zero stable recurrence and add terms that are order Δt, the new recurrence grows by at most order Δt per time step. The convergence proof for Runge Kutta methods allowed order Δt growth per time step.

[2] A sequence $x_0, x_1, \ldots$ is *bounded* if there is some *upper bound* $B < \infty$ so that $|x_n| \leq B$ for all n. A sequence of vectors is bounded if $\|\vec{x}_n\| \leq B$ (for some $B < \infty$).

The stability condition is that the characteristic polynomial ρ has no roots outside the unit circle and that all roots on the unit circle are simple. However, roots with $|z| < 1$ may have higher multiplicity. This means that there are solutions of the recurrence relation (16) that have the form $x_n = n^p z^n$, depending on the multiplicity of z. The magnitude is $n^p r^n$, where $r < 1$. Solutions like this are bounded in the sense that there is an M with $|x_n| \leq M$ for all $n \geq 0$. If r is close to 1, then $n^p r^n$ can grow a lot before it starts to decay (an easy calculus problem).

A more general view of recurrence relations makes it easier to find Lyapunov functions. Instead of a scalar s term recurrence relation, we consider a one-lag vector recurrence. Define a vector $\vec{x}_n \in \mathbb{C}^s$ as

$$\vec{x}_n = \begin{pmatrix} x_n \\ x_{n+1} \\ \vdots \\ x_{n+s-1} \end{pmatrix}$$

The vector recurrence is $\vec{x}_{n+1} = A\vec{x}_n$. The matrix A is the *companion matrix* for the recurrence relation. We find the entries of A by writing the vector recurrence out in components:

$$\begin{pmatrix} x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+s} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ & \vdots & & \ddots & & \vdots \\ & & & & & \\ & & & & & 1 \\ -\alpha_0 & -\alpha_1 & \cdots & & & -\alpha_{s-1} \end{pmatrix} \begin{pmatrix} x_n \\ x_{n+1} \\ \vdots \\ x_{n+s-1} \end{pmatrix} .$$

The first row of A equates x_{n+1} on the left with x_{n+1} on the right. The second to last row equates x_{n+s-1} on the left with x_{n+s-1} on the right. The bottom row codes the recurrence relation (16) in the form (with the convention that $\alpha_s = 1$)

$$x_{n+s} = -\alpha_0 x_n - \cdots - \alpha_{s-1} x_{n+s-1} .$$

You may see the companion matrix written in different but equivalent ways, such as the $-\alpha_j$ on the top row, or the transpose of this A with the $-\alpha_j$ as the last column.

Properties of s lag scalar recurrence relation have analogues for one lag vector recurrences. The scalar sequence x_n is bounded if and only if norms of the vector sequence $\|\vec{x}_n\|$ are bounded. It doesn't matter which vector norm you use, but the bound may depend on the norm. There is a one to one correspondence between roots of ρ and eigenvalues of the companion matrix A. A complex z satisfies $\rho(z) = 0$ if and only if z is an eigenvalue of A. A complex z is a double root (or multiple root with multiplicity at least 2) if and only if there is a non-trivial (size bigger than one) Jordan block of A corresponding to eigenvalue z. Therefore, the recurrence relation is stable (all solutions bounded) if and only

if A satisfies its own stability condition: the eigenvalues are all inside or on the unit circle, there are no non-trivial Jordan blocks for eigenvalues on the unit circle. A general matrix recurrence can have eigenvalues with multiplicity > 1 on the unit circle without being unstable. But there must be no non-trivial Jordan structure. For example, the matrix $A = I$ is associated to the stable recurrence $\vec{x}_{n+1} = \vec{x}_n$ and yet $z = 1$ is an eigenvalue with multiplicity s. If A is the companion matrix of a scalar recurrence relation, then A cannot have multiplicity without Jordan structure.

For the rest of this subsection, $\vec{x}_n \in \mathbb{C}^s$ will be any sequence of vectors that satisfy a one lag linear recurrence

$$\vec{x}_{n+1} = A\vec{x}_n \ . \tag{31}$$ vrr

We say that the vector recurrence (31) is *stable* if any sequence that satisfies (31) has a bound

$$\|\vec{x}_n\| \leq B \ , \qquad \text{for all } n \geq 1.$$

The bound B depends on the sequence. We saw, and will see again in more quantitative detail, that a recurrence is stable if every eigenvalue λ_j of A satisfy

- $|\lambda_j| \leq 1$
- $|\lambda_j| = 1 \implies$ no Jordan structure at λ_j.

A quadratic Lyapunov function is defined by a positive definite matrix M. The Lyapunov function is

$$V_M(\vec{x}) = \vec{x}^{\,t} M \vec{x} \ . \tag{32}$$ qLf

If M is positive definite (as we saw in a homework exercise) then the Lyapunov function $V(\vec{x})^{1/2}$ is a vector norm *equivalent* to any other vector norm in the sense that there is a number $\kappa > 0$ so that

$$\kappa^{-1}\sqrt{\vec{x}^{\,t} M \vec{x}} \leq \|\vec{x}\| \leq \kappa\sqrt{\vec{x}^{\,t} M x} \ . \tag{33}$$ neq

This is supposed to hold for every $\vec{x} \in \mathbb{C}^s$. The condition number κ depends on the norm $\|\cdot\|$. The function V_m is a *Lyapunov function* for the matrix recurrence if

$$\vec{x}_{n+1} = A\vec{x}_n \implies \vec{x}_{n+1}^{\,t} M \vec{x}_{n+1} \leq \vec{x}_n^{\,t} M \vec{x}_n \tag{34}$$ Lfd

The two properties (34) and (33) imply that sequences $\vec{x}_n$ are bounded:

$$\|\vec{x}_n\| \leq \kappa^2 \vec{x}_n^{\,t} M \vec{x}_n \leq \kappa^2 \vec{x}_0^{\,t} M \vec{x}_0 \leq \kappa^4 \|\vec{x}_0\| \ .$$

This is the easy half of the stability theorem:

Theorem: *A vector recurrence relation is stable if and only if it has a quadratic Lyapunov function.*

The Lyapunov function is non-increasing even when $\vec{x}_n$ seems to be increasing measured in a more direct way. For example, consider the vector recurrence

$$\vec{x}_{n+1} = \begin{pmatrix} 1 & 1 \\ 0 & 1-\delta \end{pmatrix} \vec{x}_n \ .$$

The eigenvalues are $\lambda_1 = 1$ and $\lambda_2 = 1 - \delta$. Assume that δ is a small positive number. Then the matrix satisfies our stability condition: all eigenvalues (both of them) are inside the unit disk and the eigenvalue on the unit circle is simple.

We explore the recurrence in more detail, writing

$$\vec{x}_n = \begin{pmatrix} \xi_n \\ \eta_n \end{pmatrix} \ .$$

The recurrence is

$$\begin{aligned} \xi_{n+1} &= \xi_n + \eta_n \\ \eta_{n+1} &= (1-\delta)\eta_n \ . \end{aligned}$$

First, η_n has simple monotone decay, as $\eta_n = (1-\delta)^n \eta_0$. But if δ is small, then η_n goes to zero slowly, and

$$\xi_n = \xi_0 + \sum_{k=0}^{n-1} \eta_k = \xi_0 + \frac{1-(1-\delta)^n}{\delta} \eta_0$$

can become much larger than ξ_0. As $n \to \infty$, $\xi_n \to \xi_0 + (1/\delta)\eta_0$. This is finite, as the recurrence is stable, but it can be much larger than ξ_0.

If $V(\xi, \eta)$ is a continuous Lyapunov function for this recurrence, then

$$\lim_{n\to\infty} V(\xi_n, \eta_n) = V(\xi_0 + (1/\delta)\eta_0, 0) \le V(\xi_0, \eta_0) \ .$$

This means that V must weight η more than it weights ξ, so that $\eta_0 \to 0$ reduces V enough to compensate for $\xi_0 \to \xi_0 + (1\delta)\eta_0$. We seek a quadratic Lyapunov function, so it is natural to try something like

$$V(\xi, \eta) = \xi^2 + w\eta^2 \ .$$

Here, w is a *weight*, probably a large number, that makes small decreases in η compensate for larger increases in ξ. The condition $V(\xi_{n+1}, \eta_{n+1}) \le V(\xi_n, \eta_n)$ is

$$\begin{aligned} (\xi_n + \eta_n)^2 + w\left((1-\delta)\eta_n\right)^2 &\le \xi_n^2 + w\eta_n^2 \\ \xi_n^2 + 2\xi_n\eta_n + \eta_n^2 + w(1-\delta)^2\eta_n^2 &\le \xi_n^2 + w\eta_n^2 \\ 2\xi_n\eta_n &\le \left[\delta(2-\delta)w - 1\right]\eta_n^2 \ . \end{aligned}$$

So we see this doesn't quite work.

The last inequality isn't true (for all ξ_n and η_n) not matter how large w is. Looking back, we were trying to make $\xi_{n+1}^2 + w\eta_{n+1}$ not larger than $\xi_n^2 + w\eta_n^2$.

But if ξ_n is large and η_n is small (a situation we come to for large n because $\eta_n \to 0$ but not ξ_n), then $V_{n+1} \approx (\xi_n + \eta_n)^2 \approx \xi_n^2 + 2\xi_n\eta_n$ (because $\eta_n^2 \ll \eta_n$ when η_n is small). This is not smaller than ξ_n^2. We need a fancier argument.

The key is to realize that bad term $\xi_n\eta_n$ is also turning itself off since $\eta_n \to 0$ and ξ_n is bounded. This suggests that we can add a term proportional to $\xi_n\eta_n$ in the Lyapunov function. Let us change notation (which often happens when we discover that an argument is more complicated than we hoped) and write

$$V(\xi, \eta) = \xi^2 + 2m_{12}\xi\eta + m_{22}\eta^2 \ .$$

This is the same as (in terms of the components ξ and η, or the vector $\vec{x}$)

$$V(\xi, \eta) = \begin{pmatrix} \xi & \eta \end{pmatrix} \begin{pmatrix} 1 & m_{12} \\ m_{12} & m_{22} \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} \ ,$$

$$V(\vec{x}) = \vec{x}^{\,t} M \vec{x} \ , \quad M = \begin{pmatrix} 1 & m_{12} \\ m_{12} & m_{22} \end{pmatrix} \ .$$

Repeating the calculation above, here is what we want to happen:

$$V(\xi_{n+1}, \eta_{n+1}) \leq V(\xi_n, \eta_n)$$

$$(\xi_n + \eta_n)^2 + 2m_{12}(1-\delta)\xi_n\eta_n + m_{22}(1-\delta)^2\eta_n^2 \leq \xi_n^2 + 2m_{12}\xi_n\eta_n + m_{22}\eta_n^2$$

$$2(1 - m_{12}\delta)\xi_n\eta_n \leq$$

5 Exercises

1. Check explicitly that the second order Adams Bashforth method AB2 and the second order BDF method BDF2 satisfy the accuracy conditions (16) and (19) up to $k = 2$ but not $k = 3$.

HighOrderFail

2. Use the accuracy conditions to find an explicit method with $s = 2$ that is third order accurate. Show that this method is not zero stable. If our algebra agrees, the roots of the stability polynomial $\rho(z)$ are $z = 1$ and $z = -5$. Why is $z = 1$ a root for any consistent method?

3. (*Fourier sine series Dirichlet boundary conditions*) Most partial differential equations come with boundaries and *boundary conditions*. A field (a function of x) satisfies *Dirichlet* (pronounced in English French as "dearie-shlay") boundary conditions at 0 and R if

$$u(0) = 0 \ , \ \ u(R) = 0 \ .$$

Later, we will call this *homogeneous* Dirichlet boundary conditions, "homogeneous" meaning zero. The *Neumann* (first syllable rhymes with "toy") boundary condition is

$$\partial_x(0) = 0 \ , \ \ \partial_x u(R) = 0 \ .$$

A field u satisfies a Laplace equation with homogeneous boundary conditions in the interval $[0, R]$ if

$$L_D u = h \iff \partial_x^2 u(x) = h(x) \text{ for } 0 < x < R \,, u(0) = 0 \,, \; u(R) = 0 \,.$$

The symbol L_D tells us that the boundary conditions are part of the definition of the operator.

(a) Show that the modes $v_\alpha(x) = \sin(\pi\alpha x/R)$ are Dirichlet Laplace eigenfunctions: $L_D v_\alpha = \lambda_\alpha$. There are two conditions to check, the differential equation in the "interior" $x > 0$ and $x < R$, and the boundary conditions.

(b) Show that v_α are not Neumann Laplace eigenfunctions for $[0, R]$.

(c) Find the Neumann Laplace eigenfunctions for $[0, R]$. They have the form $w_\alpha(x) = \cos(k_\alpha x)$. The wave numbers k_α include $k_0 = 0$. Note that $-\alpha$ and α give the "same" eigenfunction.

(d) Let $u(x)$ be a periodic function with period $2R$. Show that u may be written as a sum of *even* and *odd* parts

$$u(x) = u_o(x) + u_e(x) = \tfrac{1}{2}\left[u(x) - u(-1)\right] + \tfrac{1}{2}\left[u(x) + u(-x)\right] \,.$$

Show that u_o satisfies Dirichlet boundary conditions for $[0, R]$ and u_e satisfies Neumann boundary conditions.

(e) Suppose that u is real and consider the Fourier series representation of u for period $2R$. Show that u_o is a *Fourier sine series*

$$u_o(x) = \sum_{\alpha=1}^{\infty} \widehat{u}_{o,\alpha} \sin(\pi\alpha x/R)$$

and u_e is a *Fourier cosine series*. Use the Fourier series formula to find a formula of the form

$$\widehat{u}_{o,\alpha} = C \int_0^R v_\alpha(x) u_o(x)\, dx \,.$$

(f) Consider the diffusion equation with Dirichlet boundary conditions:

$$\partial_t u(x,t) = D\partial_x^2 u(x,t) \,, \;\; u(0,t) = 0 \,, \; u(R,t) = 0 \,.$$

The initial condition is $u(x, 0) = f(x)$. Show that the solution is

$$u(x,t) = \sum_{1}^{\infty} \widehat{f}_\alpha e^{\lambda_\alpha t} \sin(k_\alpha x) \,.$$

Find λ_α and k_α.

(g) Use this to show that for large t, $u(x,t) \approx \widehat{f}_1 e^{\lambda_1 t} \sin(\pi x/R)$ in the sense that the error in the approximation is much smaller than u (unless $\widehat{f}_1 = 0$).

4. (*Discrete sine transform*) Suppose that $U \in \mathbb{R}^{n-1}$ is an approximation to $u(\cdot, t)$ in the sense that

$$U_j(t) \approx u(x_j, t) \; .$$

The grid points are $x_j = j\Delta x$, where Δx is chosen so that $x_0 = 0$ and $x_n = R$. If u satisfies Dirichlet boundary conditions, then $U_0 = 0$ and $U_n = 0$, so the unknowns are $U_1, \ldots, U_{n-1}$.

(a) Find $n-1$ discrete Fourier sine modes of the form $V_{\alpha,j} = \sin(\pi\alpha\Delta x j)$.

(b) Use the DFT formulas for a periodic function with period $2n$ to find the representation formula

$$U = \sum_1^{n-1} \widehat{U}_\alpha V_\alpha \; , \quad U_j = \sum_1^{n-1} \widehat{U}_\alpha \sin(\pi\alpha j \Delta x) \; .$$

It is possible to do this directly by calculating the orthogonality properties of discrete Fourier sine modes, but then you would not be able to ...

(c) Show that the $\widehat{U}_\alpha$ may be calculated using the usual complex FFT for periodic functions with period $2n$.

(d) Show how to use this to create a discrete Dirichlet Laplace operator on U that is spectrally accurate. Use this to define a spectral semi-discrete approximation to the diffusion equation with Dirichlet boundary conditions. Give an algorithm using an FFT and inverse FFT to calculate the exact solution $U(T)$ in terms of $U(0)$.

(e) Suppose instead that we want a second order finite difference approximation to the Dirichlet Laplace operator. Show that the discrete sine modes are the $n-1$ eigenvectors of this and find the eigenvalues. Use these to show that the condition number of this matrix is $O(n^2)$.

5. (*Splitting*) It is common that a differential equation has several terms that model different physical processes. For example, the Allen Cahn equation has one term for "reaction" and another term for diffusion. A generic mathematical model is

$$\dot{x} = f(x) + g(x) \; . \tag{35}$$

fg

It can happen that there is a good method for solving $\dot{x} = f(x)$ and a different good method for solving $\dot{x} = g(x)$. *Splitting* is putting these two solvers together to make a solver for the combined problem (35). Suppose $\Phi_f(x,t)$ is the flow map for $\dot{x} = f(x)$, and $\Phi_g(x,t)$ is the flow map for $\dot{x} = g(x)$. Let Φ_c be the flow map for the combined problem (35). Let $\Psi_f(x,t)$ and $\Psi_g(x,t)$ be time step maps with order of accuracy p:

$$|\Psi_f(x,t) - \Phi_f(x,t)| = O(t^{p+1}) \; .$$

(a) The *basic splitting* is the approximation that you "run" the combined problem (35) for time Δt by running first one then the other. In the first equivalent formula, the ring $\circ$ denotes composition of functions.

$$\begin{aligned} \Psi_{cb}(\cdot,\Delta t) &= \Phi_f(\cdot,\Delta t)\circ\Phi_g(\cdot,\Delta t) \\ z = \Psi_{cb}(x,\Delta t) \;\text{ if }\; y = \Phi_g(x,\Delta t) \;\text{ and }\; z &= \Phi_f(y,\Delta t) \\ \Psi_{cb}(x,\Delta t) &= \Phi_f(\Phi_g(x,\Delta t),\Delta t)\ . \end{aligned} \tag{36}$$

bsp

Show that this basic splitting is first order accurate in the sense that

$$\Psi_{cb}(x,\Delta t) - \Phi_c(x,\Delta t) = O(\Delta t^2)\ .$$

(b) Suppose the differential equations are linear:

$$f(x) = Ax\ ,\ \ g(x) = Bx\ .$$

Then the flow maps are given by the fundamental solutions, also called matrix exponentials:

$$\Phi_f(x,t) = e^{tA}x\ ,\ \ \Phi_g(x,t) = e^{tB}x\ ,\ \ \Phi_c(x,t) = e^{t(A+B)}x\ .$$

The matrix exponentials may be defined by power series; for example

$$e^{tA} = I + tA + \tfrac{1}{2}t^2A^2 + \tfrac{1}{6}t^3A^3 + \cdots\ .$$

Show that the basic splitting scheme (36) for matrix exponentials is

$$e^{\Delta t(A+B)} \approx e^{\Delta tA}e^{\Delta tB}\ .$$

Show that the basic splitting scheme is more than first order accurate if and only if the matrices A and B *commute*. Matrices A and B commute if $AB = BA$. This exercise is related to the *Baker Campbell Hausdorff* formula (e.g., wikipedia).

(c) Suppose that use approximations for Φ_f and Φ_g that are at least first order accurate. Show that the combined approximation is first order accurate for any $p \geq 1$. In other words, if you're using basic splitting, there's little benefit to solving the subproblems to more than first order accuracy. Technically, you're being asked to show

$$\Psi_f(\Psi_g(x,\Delta t),\Delta t) - \Phi_c(\cdot,\Delta t) = O(\Delta t^2)\ .$$

(d) The *Strang splitting*[3] is the more symmetric approximation of putting a whole step of f between two half steps of g:

$$\Phi_{cs} = \Phi_g(\cdot,\tfrac{1}{2}\Delta t)\circ\Phi_f(\cdot,\Delta t)\circ\Phi_g(\cdot,\tfrac{1}{2}\Delta t)\ .$$

Show that the Strang splitting is second order accurate.

[3] Gilbert Strang is a smart numerical computing person and the writer of clear books on the finite element method, linear algebra, and applied mathematics.

(e) Show that you can take n time steps of Strang splitting by doing half steps only at the beginning and the end:

$$x \to (\tfrac{1}{2}\Delta t \text{ of } g) \to (\Delta t \text{ of } f) \to (\Delta t \text{ of } g) \to \cdots \to (\Delta t \text{ of } f) \to (\tfrac{1}{2}\Delta t \text{ of } g)$$

This uses the "semigroup property" of the flow map, as in

$$\Phi_f(\cdot, \tfrac{1}{2}\Delta t) \circ \Phi_f(\cdot, \tfrac{1}{2}\Delta t) = \Phi_f(\cdot, \Delta t) \ .$$

6. (*2D DFT, etc.*) Consider a discrete grid function of two variables U_{jk}. Suppose this is periodic in both variables with period n:

$$U_{jk} = U_{j+n,k} = U_{j,k+n}$$

Consider 1D DFT modes $V_{\alpha,j} = e^{2\pi i \alpha j/n}$ and 2D modes $V_{\alpha\beta,jk} = V_{\alpha,j} V_{\beta,k}$. Find 2D versions of the basic facts of the 1D DFT, starting with the representation

$$U_{jk} = \sum_{\alpha,\beta} \widehat{U}_{\alpha\beta} V_{\alpha,j} V_{\beta,k} \ .$$

Find the DFT formula for $\widehat{U}_{\alpha\beta}$ and the Parseval formula. You can do this directly, or by using the 1D formulas, first in "x" (the j variable), then in y. You first treat k as a constant and use the 1D formula

$$U_{jk} = \sum_{\alpha} \widetilde{U}_{\alpha,k} V_{\alpha,j} \ .$$

The 1D Parseval formula is, for each k,

$$\sum_j |U_{jk}|^2 = C \sum_{\alpha} \left| \widetilde{U}_{\alpha,k} \right|^2 \ .$$

Then you treat α as constant and use the 1D DFT to get

$$\widetilde{U}_{\alpha,k} = \sum_{\beta} \widehat{U}_{\alpha,\beta} V_{\beta,k} \ .$$

7. The *Allen Cahn equation* is

$$\partial_t u = \triangle u + u - u^3 \ . \tag{37}$$ AC

It models the motion of smooth approximate phase boundaries in a solid material. The *minus* phase corresponds to $u = -1$ and the *plus* phase corresponds to $u = 1$. If u is independent of x, y, then the PDE (37) becomes the ODE $\dot{u} = u - u^3$. If $u(0) \neq 0$ then $u(t) \to \pm 1$ as $t \to \infty$. In the PDE, there will be regions $u \approx +1$ and $u \approx -1$ separated by transition layers called *phase boundaries*. If you start from random initial data (or any data that isn't too simple), the solution quickly goes to a state like this, then the phase boundaries move slowly.

(a) Create semi-discrete approximation to (AC 37) in 2D with Dirichlet boundary conditions at $x = 0$, $x = R$, $y = 0$, and $y = R$. You can use a spectral or a finite difference discretization of the Laplace operator with Dirichlet boundary conditions.

(b) Solve this by splitting. Do the linear piece

$$\partial_t u = f(u) = \triangle u + u$$

exactly using the FFT. Do the remaining nonlinear piece

$$\partial_t u = -u^3$$

exactly in physical space. Give computational evidence that the resulting method using Strang splitting is second order accurate.

(c) Use the Python compiler `cython` as in the posted code `heat`.

(d) If you can, make a movie of the Allen Cahn solution. I was unable to do this. Otherwise, print some snapshots (contour plots at specific times) that show the phase boundary structure. You will have to take a reasonably large box and wait a while for this structure to appear.