MATH 228 lecture notes for November 22, 24, and 26

Russell Milne

November 2021

1 November 22: More about periodic orbits and the stability of fixed points

Last week, we saw some examples of periodic orbits of dynamical systems. These are any solutions for which the trajectory of all of the state variables follows a closed, deterministic path throughout phase space, which doesn't have to be defined by sine and cosine. You may recall that for the Lotka-Volterra predator-prey system, we showed that there were periodic orbits. Here is the Lotka-Volterra system, for reference:

$$\begin{cases} \frac{dN}{dt} = rN - \alpha NP \\ \frac{dP}{dt} = \beta NP - mP \end{cases} \tag{1}$$

To determine that periodic orbits exist for this system, we first found a fixed point for which the eigenvalues are purely imaginary, which is $(x^*, y^*) = (\frac{m}{\beta}, \frac{r}{\alpha})$, and noted that the nullclines of $N = 0$ and $P = 0$ made it impossible for a solution that started in the first quadrant (i.e. with $N, P \geq 0$) to leave it. Then, we took the derivative $\frac{dP}{dN}$ to represent the motion of this system in the phase plane, then showed that integrating $\frac{dP}{dN}$ resulted in a finite conserved quantity that can be expressed as a function of N and P. This meant that solutions for N and P in the first quadrant for which $0 < N, P < \infty$ had to stay within those bounds, and specifically follow a trajectory in the phase plane defined by what we got when we integrated $\frac{dP}{dN}$. (Note that in Cartesian coordinates, $x^2 + y^2 = 1$ defines the unit circle, and $m \ln N + r \ln P - \beta N - \alpha P = C$ also defines a closed curve in the (N, P)-plane.)

We also talked about how periodic orbits can arise due to a Hopf bifurcation. Specifically, a Hopf bifurcation can occur when all of the eigenvalues of the Jacobian evaluated at a fixed point of a dynamical system have negative real parts, except for two that are purely imaginary conjugates of each other. If some parameter in the system is changed so that the two imaginary conjugate eigenvalues shift to having positive real parts, then the fixed point in question switches from stable to unstable, but a stable limit cycle is created surrounding the fixed point.

There are a few more results that may be useful for finding periodic orbits. To illustrate these, suppose we have the following autonomous dynamical system:

$$\begin{cases} \frac{dx}{dt} = f(x,y) \\ \frac{dy}{dt} = g(x,y) \end{cases} \tag{2}$$

We choose this system to be autonomous because having time dependence can potentially throw a solution off of a limit cycle, just like with a fixed point. Anyway, one important result concerns where orbits and limit cycles can and cannot appear in this system's phase plane (i.e. (x, y)-space). Any closed trajectory (in other words, an orbit) of a two-dimensional dynamical system like the one above must enclose at least one fixed point in the phase plane. Furthermore, if the orbit only encloses one fixed point, then that fixed point cannot be a saddle. (We saw this for Hopf bifurcations, where a limit cycle must enclose a node.) As a matter of fact, any orbit in the phase plane must enclose an odd number of fixed points, $2n + 1$ of them for n a non-negative integer, of which n are saddles and $n+1$ are either sinks (stable nodes) or sources (unstable nodes). These findings come from a field called index theory, which is a bit beyond the scope of this course. A handy way to remember them (which also comes from index theory) is to say that in the phase plane, the index of a source or sink is $+1$, the index of a saddle point is -1, the index of a periodic orbit is also $+1$, and the sum of any curve in the phase plane (regardless of whether it is an orbit) is the sum of the indices of any fixed points that it encloses. This method allows us to demonstrate that a given curve in the phase plane is not an orbit (if it has an index of anything other than $+1$), as well as to rule out orbits entirely in some cases (if it is impossible to draw a closed curve with index $+1$).

Another result is an application of Green's theorem (from Calculus 3) to dynamical systems. Suppose that we have the two-dimensional dynamical system mentioned above, and both $f(x, y)$ and $g(x, y)$ have continuous first partial derivatives. Suppose also that we have some simply connected region R in the phase plane. If the function $\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y}$ does not change sign anywhere in R, then our system $x' = f(x, y),\ y' = g(x, y)$ has no closed orbits in R. As a matter of fact, this is a specific case of a more general result. If we pick any function $u(x, y)$ that is continuously differentiable, such that the function $\frac{\partial}{\partial x}(uf) + \frac{\partial}{\partial y}(ug)$ does not change sign in R, then there are no closed orbits of our system in R. This general case is called Dulac's criterion. It can be used to rule out periodic orbits in regions of phase space, but one downside is that it requires coming up with a suitable function $u(x, y)$.

We now have some tools for determining that there are no closed orbits in some region of the (x, y)-plane for our dynamical system $x' = f(x, y),\ y' = g(x, y)$. What if we want to prove that an orbit exists in some region? That one's a little tougher. However, we do have one result that we can use. Suppose we have some region R in the phase plane that is bounded, contains its boundary, does not contain any fixed points of the system $x' = f(x, y),\ y' = g(x, y)$, and on which $f(x, y)$ and $g(x, y)$ are both continuously differentiable. If there is some solution trajectory that remains in R for all values of t greater than some fixed value t_0, then R contains a periodic orbit of the system. (Note that R cannot be simply connected, since the periodic orbit in question needs to enclose a fixed point of the system, which cannot itself be in R.) This result is called the Poincaré-Bendixson Theorem.

So, we now have a few results on the topic of orbits that we can use. What about fixed points? Is it possible to determine the stability of a fixed point without having to go through all the trouble of finding the eigenvalues of the Jacobian? (This is a very legitimate question for large systems, since computing anything involving large matrices can get quite intense.) It turns out that there is. One method is by finding what is called a Lyapunov function. Suppose that we have an autonomous dynamical system of the following form:

$$\begin{cases} \frac{dx_1}{dt} = f_1(x_1, \ldots, x_n) \\ \cdots \\ \frac{dx_n}{dt} = f_n(x_1, \ldots, x_n) \end{cases} \tag{3}$$

Suppose also that this dynamical system has an isolated fixed point at the origin. (Mathematically, this means that there is some neighbourhood of the fixed point at the origin which no other fixed points are in.) If it has one that is not at the origin, then we can move it there by using a coordinate transform. Now, consider a function $V(x_1, \ldots, x_n)$ that is "positive definite". This means that V is equal to zero at the origin and is positive everywhere else. Consider also the time derivative of V:

$$\frac{dV}{dt} = \frac{\partial V}{\partial x_1}\frac{dx_1}{dt} + \ldots + \frac{\partial V}{\partial x_n}\frac{dx_n}{dt} \tag{4}$$

If the time derivative of V is "negative definite", i.e. it is zero at the origin and negative everywhere else, then the origin is a stable fixed point of the system under consideration, and we call V a "Lyapunov function". More specifically, the origin will be what we call "asymptotically stable", which means that there exists some $\delta > 0$ such that all solution trajectories that start a distance less than δ away from the origin will have their distance towards the origin go to zero in the limit case as $t \to \infty$. (This is more or less equivalent to the origin being a sink.) We get a stronger result if V is also radially unbounded. "Radially unbounded" means that $||\mathbf{x}|| \to \infty \implies V(\mathbf{x}) \to \infty$, for some norm $|| \cdot ||$ and $\mathbf{x}$ being the vector with entries $x_1, \ldots, x_n$. We can take $|| \cdot ||$ to be any norm, so the 1-norm $x_1 + \ldots + x_n$ or the 2-norm (which is the Euclidean distance) will suffice. In this case, then the origin is "globally asymptotically stable", which means that δ can be chosen to be any finite positive number (and hence the origin attracts solutions that start anywhere in phase space). If $\frac{dV}{dt}$ is only negative definite in some bounded region surrounding the origin, then we can only prove that solutions starting in that region tend to the origin. On the other hand, if $\frac{dV}{dt}$ is positive definite, then the origin is unstable. The theory behind Lyapunov functions can produce very strong results, particularly for large systems when other methods might be infeasible. However, as with Dulac's criterion, it requires coming up with a suitable function.

Let's see a simple example of this. Suppose we have the following system:

$$\begin{cases} \frac{dx}{dt} = -x \\ \frac{dy}{dt} = -y \end{cases} \tag{5}$$

We know based on past experience that there is one fixed point for this system, namely the origin, and that it is stable. Can we prove this using a Lyapunov function? Let's try $V(x, y) = x^2 + y^2$. This is clearly positive definite, and it is also radially unbounded, since any combination of x and y that goes to infinity also makes V go to infinity. We will now calculate the time derivative of V:

$$\frac{dV}{dt} = \frac{\partial V}{\partial x}\frac{dx}{dt} + \frac{\partial V}{\partial y}\frac{dy}{dt} = -2x^2 - 2y^2 \tag{6}$$

This is negative definite, so we can conclude that the origin is a globally asymptotically stable fixed point for this dynamical system.

How about a more challenging example? Suppose we have the following system:

$$\begin{cases} \frac{dx}{dt} = -x + y^2 \\ \frac{dy}{dt} = 3x^2 - 2y \end{cases} \tag{7}$$

This system has a fixed point at the origin, but it also has another one at $(x^*, y^*) = \left(\left(\frac{2}{3}\right)^{2/3}, \left(\frac{2}{3}\right)^{1/3}\right)$. Therefore, the origin cannot be globally asymptotically stable, because it cannot attract solutions that start at the other fixed point. However, we can still try to find a Lyapunov function. Consider the following positive definite, radially unbounded function:

$$V(x, y) = \frac{x^2}{2} + \frac{y^2}{4} \tag{8}$$

The time derivative of this is as follows:

$$\frac{dV}{dt} = -x^2 + xy^2 - y^2 + \frac{3}{2}yx^2 = -x^2(1 - \frac{3}{2}y) - y^2(1 - x) \tag{9}$$

This is negative definite when $x < 1$ and $y < \frac{2}{3}$, so we at least know that solutions that start within those bounds will tend towards the origin. (The actual region of phase space in which solutions tend towards the origin is actually much bigger, but this is as far as Lyapunov functions will take us.)

Another tool in our arsenal for evaluating the stability of fixed points is the Routh-Hurwitz criterion. If we already have evaluated the Jacobian at a fixed point, then we will be able to find the characteristic polynomial. However, characteristic polynomials of order 3 or 4 might be hard to find roots for, and of course any polynomial of order at least 5 cannot be solved algebraically. This means that we need some other way to assess the stability of fixed points in systems that large. Anyway, suppose that we have a characteristic equation as follows:

$$a_n r^n + a_{n-1} r^{n-1} + \ldots + a_1 r^1 + a_0 = 0 \tag{10}$$

We won't be able to algebraically determine the roots of this polynomial if it's large enough, but we will be able to determine if they have negative real parts based solely on the coefficients of the polynomial. This will be done by using the Routh-Hurwitz stability criterion, and to use this criterion we first need to construct a table using the coefficients called the "Routh array". If our characteristic polynomial is of degree n, then our table will have n rows, which we will denote r^n, r^{n-1}, and so on down to r^1. The first two rows of the table will be constructed out of alternating coefficients in the characteristic equation, with a_n being the first entry:

$$\begin{array}{r|cccc} r^n & a_n & a_{n-2} & a_{n-4} & \cdots \\ r^{n-1} & a_{n-1} & a_{n-3} & a_{n-5} & \cdots \end{array} \tag{11}$$

If we run out of coefficients for a particular row, we just let everything to the right of the last coefficient in that row be zero. For instance, if n is even, then the first row will end with a_0, so there won't be any coefficent that we can put directly below a_0 in the second row (so we let that entry in the second row be zero).

To construct the rest of the rows in the table, we will use a recursive formula. If we call the entries of the third row b_1, b_2, b_3, and so on, then the formulae for them are as follows:

$$b_1 = \frac{-1}{a_{n-1}} \det \begin{bmatrix} a_n & a_{n-2} \\ a_{n-1} & a_{n-3} \end{bmatrix}, \; b_2 = \frac{-1}{a_{n-1}} \det \begin{bmatrix} a_n & a_{n-4} \\ a_{n-1} & a_{n-5} \end{bmatrix}, \; \ldots \tag{12}$$

If we extend down to the fourth row, the entries (which we can call c_1, c_2, c_3, etc.) will follow a similar pattern:

$$c_1 = \frac{-1}{b_1} \det \begin{bmatrix} a_{n-1} & a_{n-3} \\ b_1 & b_2 \end{bmatrix}, \; c_2 = \frac{-1}{b_1} \det \begin{bmatrix} a_{n-1} & a_{n-5} \\ b_1 & b_3 \end{bmatrix}, \; \ldots \tag{13}$$

Eventually, the Routh array will look like this, with q denoting the last entry obtained from this process:

$$\begin{array}{r|cccc} r^n & a_n & a_{n-2} & a_{n-4} & \cdots \\ r^{n-1} & a_{n-1} & a_{n-3} & a_{n-5} & \cdots \\ r^{n-2} & b_1 & b_2 & b_3 & \cdots \\ r^{n-3} & c_1 & c_2 & c_3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ r^0 & q & 0 & 0 & 0 \end{array} \tag{14}$$

The entries in any given row (besides the first two) will depend on the entries in the two rows directly above it. Additionally, when calculating any given entry, the left-hand column of the determinant in the formula for that entry will include the first entries in each of the two rows directly above the row that is being created, and the right-hand column of the determinant will gradually move right in terms of which entries it contains. The determinant will always be divided by the negative of the first entry in the row directly above the one currently being created. Mathematically, if we denote the Routh array as a matrix $\mathbf{M}$, then we get the following formula for $m_{i,j}$, for $i > 2$:

$$m_{i,j} = \frac{-1}{m_{(i-1),1}} \det \begin{bmatrix} m_{(i-2),1} & m_{(i-2),(j+1)} \\ m_{(i-1),1} & m_{(i-1),(j+1)} \end{bmatrix} \tag{15}$$

Now that we have our Routh array, we can use the actual Routh-Hurwitz criterion. The number of times that the sign of an entry in the first column is different than the sign of the entry directly above it is the number of roots that the characteristic polynomial has with positive real parts. Therefore, if the entries in the first column of the Routh array are either all positive or all negative, then the fixed point in question is stable. A simplified version of this criterion for second-order polynomials $r^2 + a_1 r + a_0$ is that the polynomial's roots will all have negative real parts if and only if a_1 and a_0 are both positive. For third-degree polynomials $r^3 + a_2 r^2 + a_1 r + a_0$, we need a_2 and a_0 to both be positive and $a_2 a_1 > a_0$.

2 November 24: Floating point arithmetic, Euler's method, and error bounds

Previously, we have learned about dynamical systems that may be hard or impossible to solve by hand. This is generally because they are nonlinear. Despite this, these kinds of dynamical systems make up the vast majority that you will encounter in real life, and so finding solutions to them is important. Therefore, we can use various numerical methods to approximate the solutions to the dynamical systems in question. Numerical integration is just one part of the field of numerical analysis, which also includes things like methods for approximating the eigenvalues of a matrix (which I will touch upon later on in this course).

In general, numerical integration of systems of differential equations relies on starting from a known quantity (i.e. the initial condition of the system), then using the rates of change of each state variable to approximate what a solution will look like afterwards. This process can be iterated over and over again to build a curve that approximates the analytical solution. One very simple way to illustrate this is with Euler's method. Suppose we have the following very simple one-dimensional ODE with an initial condition:

$$\frac{dx}{dt} = f(t, x), \;\; x(t = 0) = x_0 \tag{16}$$

We can express $\frac{dx}{dt}$ as a limit, using the form that you learned in Calculus 1. Using this, the ODE becomes the following:

$$\lim_{h\to 0}\frac{x(t+h)-x(t)}{h} = f(t,x) \tag{17}$$

However, we will not take this limit, and instead assume that h is a very small number. We can then do some algebra to get the following:

$$x(t+h) = x(t) + hf(t,x) \tag{18}$$

Now, we have a formula in which we can use $x(t)$, the value of our solution at the current time, and $f(t,x)$, the derivative evaluated at the current time, to find $x(t+h)$, the value of the solution at some time in the future. This is a recurrence relation, so we'll need something to start with. An obvious choice would be x_0, since that was provided for us in the problem. We can use that to get x_h, then x_{2h}, x_{3h}, and so forth. In higher dimensions, the formula still holds, although our inputs and outputs will be vector-valued:

$$\begin{cases} \frac{dx_1}{dt} = f_1(t,x_1,\dots,x_n) \\ \cdots \\ \frac{dx_n}{dt} = f_n(t,x_1,\dots,x_n) \end{cases} \implies \begin{cases} x_1(t+h) = x_1(t) + hf_1(t,x_1,\dots,x_n) \\ \cdots \\ x_n(t+h) = x_n(t) + hf_n(t,x_1,\dots,x_n) \end{cases} \tag{19}$$

Now that we have this tool for approximating solutions, let's see if it works. We will test it out on an ODE that we know the solution to:

$$\frac{dx}{dt} = x^2,\ x(t=0) = -1 \tag{20}$$

We can solve this analytically to get $x(t) = \frac{-1}{t+1}$, which has a simple functional form for us to compare our answers with. Now, let's do a few steps of Euler's method. We'll start with $h = 0.1$ as our step size, to keep things tractable for now. We get the following:

$$\begin{aligned} &x(0) = -1 \\ &x(0.1) = -1 + 0.1\cdot(-1)^2 = -0.9 \\ &x(0.2) = -0.9 + 0.1\cdot(-0.9)^2 = -0.819 \\ &x(0.3) = -0.819 + 0.1\cdot(-0.819)^2 \approx -0.752 \end{aligned} \tag{21}$$

Now, let's compare this to the actual values of our solution, to see how well we did. Here are the actual values of $x(t)$ at the points that we specified:

$$\begin{aligned} &x(0) = -1 \\ &x(0.1) = \tfrac{-1}{0.1+1} \approx -0.909 \\ &x(0.2) = \tfrac{-1}{0.2+1} \approx -0.833 \\ &x(0.3) = \tfrac{-1}{0.3+1} \approx -0.769 \end{aligned} \tag{22}$$

It's not terrible; we do capture the overall trend (increasing towards zero at a decreasing rate) as well as one digit of accuracy following the decimal point. The reason why we don't get closer estimates is mainly because we used a relatively large step size. We can see this by considering the Taylor expansion of $x(t)$ around some point t_0:

$$x(t) = x(t_0) + x'(t_0)(t - t_0) + \frac{x''(t_0)}{2}(t - t_0)^2 + \mathcal{O}(t^3) \tag{23}$$

Now, take $t = t_1$ for some time $t_1 > t_0$:

$$x(t_1) = x(t_0) + x'(t_0)(t_1 - t_0) + \frac{x''(t_0)}{2}(t_1 - t_0)^2 + \mathcal{O}((t_1 - t_0)^3) \tag{24}$$

We can specify $h = t_1 - t_0$, and we also know that $x'(t_0) = f(t_0)$. Therefore, the first two terms of the Taylor series turn into one step of Euler's method, and we get the following:

$$x(t_0 + h) = x(t_0) + hf(t_0) + \frac{x''(t_0)}{2}h^2 + \mathcal{O}(h^3) \tag{25}$$

Note that the t^2, t^3, etc. terms in the Taylor series will turn into h^2, h^3, etc., since the dependence is now on $(t_1 - t_0) = h$. We ignore everything from the third term onward when we perform a step of Euler's method, but those terms are still necessary parts of the solution (since the Taylor expansion is an infinite sum). Therefore, the difference between the real value of x at $t = t_0 + h$ and our simulated value (which we will call x_1, by analogue with $x(t_0) = x_0$) is the following:

$$\text{LTE} = x(t_0 + h) - x_1 = \frac{x''(t_0)}{2}h^2 + \mathcal{O}(h^3) \tag{26}$$

This is specifically the error that accumulates by taking one step with Euler's method, which we call the "local truncation error" (hence "LTE" above) because it results from truncation of the Taylor series. If we assume that $h < 1$, which is more or less always the case, then we get $h^2 > h^n \ \forall n > 2$. This allows us to combine the later terms in the Taylor series, and say that the local truncation error is on the order of h^2. (This holds so long as the third derivative x''' is bounded, since if it isn't, then we can't assume that the later terms in the Taylor series are less prominent than the h^2 term.) What about the overall error present in the approximation? In other words, how different will the curve that we simulate be from the actual solution? Well, if the beginning of our simulations is the point t_0 that we already specified, and we run the simulations until some other time t_{final}, then the total number of steps is $\frac{1}{h}(t_{\text{final}} - t_0)$. Therefore, if there is $\mathcal{O}(h^2)$ error in every step, then the total amount of error (the "global truncation error") will be the following:

$$\text{GTE} = \frac{t_{\text{final}} - t_0}{h}\frac{x''(t_0)}{2}h^2 = \mathcal{O}(h) \tag{27}$$

This is, as we can see, proportional to h. Therefore, the smaller we take h, the less truncation error (local and global) we get. Since the global truncation error is proportional to the first power of h, i.e. h^1, we say that Euler's method is a "first-order method". (If the global truncation error depended on h^2, it would be a second-order method, and so on.) What if we instead increased the step size, for instance taking $h = 1$? If we simulate the same ODE as before, we will get the following:

$$\begin{aligned} x(0) &= -1 \\ x(1) &= -1 + 1 \cdot (-1)^2 = 0 \\ x(2) &= 0 + 0 = 0 \end{aligned} \tag{28}$$

Having too big of a step size can make the method break down completely. In practice, your step size usually shouldn't be anywhere near $h = 1$; I mostly use $h = 10^{-3}$ myself if I don't need extra precision.

However, even though we showed that taking lower values for h leads to lower truncation error, that doesn't mean that we always need to choose as low a value for h as possible. There are a couple important reasons for this. The first one is that smaller h means more steps that need to be computed, which in turn means that your numerical integration will take more time. (One way to balance this is to integrate your dynamical system with two different values of h, one being the value that you are using and the other being a value one order of magnitude greater. If the difference between the two solutions produced is acceptably small, then it's probably fine to use the larger value of h to save computation time.)

The other reason that lower h is not always better also has to do with the number of computations performed. Computers only have a finite amount of memory, and base-10 numbers are stored in a computer's memory in the form of base-2 approximations with a finite number of digits. This means that there are only a finite number of computer-representable numbers, and hence in most cases, the computer representation of a decimal number will not be the same as the form that we would write out by hand. Because of this, mathematical operations done on a computer will typically lose a small amount of accuracy, due to having to round off the operation's output to something which is representable by the computer. This is called "round-off error". An example of this is as follows. Consider the decimal number 0.1, or $(0.1)_{10}$. When converted into binary, it has an infinite number of decimal places:

$$(0.1)_{10} = (0.00011001100110011\ldots)_2 \tag{29}$$

This means that representing it on a computer (with finite precision as opposed to infinite precision) must necessarily eliminate all of its digits after a certain point, resulting in error. A corollary of this is that two numbers that differ by less than the computer's precision will be represented as the same number. For example, suppose we have a very primitive computer that can only handle five binary digits after the decimal point (i.e. up to 2^{-5}). This computer's representation of $(0.1)_{10}$ would be $(0.00011)_2$, but that would be the same as the representation of (for instance) $(0.1 + 2^{-6})_{10}$. In this way, two operations on our primitive computer that have expected outputs of $(0.1)_{10}$ and $(0.1 + 2^{-6})_{10}$ would in practice be indistinguishable. If h is large, or at least fairly large, the total amount of round-off error will be at least an order of magnitude smaller than the truncation error. However, if h is sufficiently small, round-off errors will build up and cause substantial total error if many computational steps are performed. Therefore, an ideal value for h would be not too large and not too small.

Previously, we found that the global truncation error of Euler's method was proportional to h. Do methods exist that have greater accuracy for a given step size? Yes, in fact there are several. One of the most commonly used ones is the Runge-Kutta fourth-order method, or RK4. Since this is a fourth-order method, its global truncation error will be $\mathcal{O}(h^4)$, and by extension its local truncation error will be $\mathcal{O}(h^5)$. This is much more accurate than Euler's method, which is first order and therefore has a global truncation error of $\mathcal{O}(h)$.

So, how does the fourth-order Runge-Kutta method work? The main concept behind it is that we will predict the next step in our numerical solution, $x(t+h)$, by using a weighted average of the predicted slopes of x in the interval $[t, t+h]$. It's more accurate for the same reasons that the trapezoidal rule is more accurate than the various rectangular approximation methods for calculating integrals, or why Simpson's method is more accurate than the trapezoidal rule. RK4 takes a weighted average of four different slopes in this interval. This means that the recurrence relation for getting the next step of a solution in RK4 has four terms with h in them. The specific formulation of RK4 is as follows, for an ODE $x' = f(t,x)$, an initial condition $x(t=0) = x_0$, and the assumption that each step advances time t by h units:

$$\begin{aligned} &x_{n+1} = x_n + h\left(\tfrac{1}{6}k_1 + \tfrac{1}{3}k_2 + \tfrac{1}{3}k_3 + \tfrac{1}{6}k_4\right) \\ &k_1 = f\left(t_n, x_n\right) \\ &k_2 = f\left(t_n + \tfrac{h}{2}, x_n + k_1\tfrac{h}{2}\right) \\ &k_3 = f\left(t_n + \tfrac{h}{2}, x_n + k_2\tfrac{h}{2}\right) \\ &k_4 = f\left(t_n + h, x_n + k_3 h\right) \end{aligned} \tag{30}$$

Note that k_1 is linear in terms of h, k_2 involves taking h times k_1 and is thus quadratic in h, and likewise k_3 and k_4 are cubic and quartic in h, respectively. This is what makes the local truncation error on the order of h^5 (and hence the global truncation error on the order of h^4 once we multiply by $\frac{1}{h}(t-t_0)$). Let's iterate this a few times with the differential equation we looked at earlier ($x' = x^2$, $x(0) = -1$) and see if it does better than Euler's method. I'll omit the algebra that takes place in terms of calculating the steps, and just show you the results, to ten decimal places this time:

$$\begin{aligned} &x(0) = -1 \\ &x(0.1) \approx -0.9090911863 \\ &x(0.2) \approx -0.8333337288 \\ &x(0.3) \approx -0.7692312058 \end{aligned} \tag{31}$$

In this case (and in general), RK4 does indeed do better than Euler's method, and by a considerable margin. The actual solution values are (to ten decimal places) -0.9090909091, -0.8333333333, and -0.7692307692, which means that we get five digits of accuracy even with a relatively large step size of $h = 0.1$.

3 November 26: Multistep methods, stiff equations and stability

In the previous lecture, we looked at the fourth-order Runge-Kutta method, which is probably the most widely-used numerical integration scheme in the majority of cases. However, there are some cases where other methods may be desired. For instance, certain ODEs may have solutions that change very rapidly in time, such that accurately approximating a solution may be impossible without taking an extremely small step size. These are called "stiff" ODEs, and methods such as Euler's method may have trouble solving them. For stiff ODEs, we need to use methods that can handle these sharp changes in the solution; the ability of a numerical integration method to do this is referred to as its "stability".

For an illustrative example, consider the ODE $x' = -100x$ with initial condition $x(0) = 1$. This has an analytical solution of $x(t) = e^{-100t}$, and it should therefore decay to 0 quite quickly. What happens when we attempt to use Euler's method on it, with a step size of 0.1?

$$\begin{aligned}
&x(0) = 1 \\
&x(0.1) = 1 + 0.1(-100 \cdot 1) = -9 \\
&x(0.2) = -9 + 0.1(-100 \cdot -9) = 81 \\
&x(0.3) = 81 + 0.1(-100 \cdot 81) = -729
\end{aligned} \tag{32}$$

Instead of monotonically converging to 0, our simulated solution diverges while oscillating about the horizontal axis, which is the opposite behaviour of what we wanted. It turns out that Euler's method is not particularly stable when stiff equations are concerned. One way to see this is the concept of "A-stability". A numerical integration scheme is A-stable if it successfully captures the fact that all ODEs of the form $x' = kx$, for k a constant with negative real part, tend to zero as $t \to \infty$. A related concept is that of the "stability region" of a numerical integration scheme. For a numerical scheme that takes x_{n+1} to be a function of x_n, integrating a test function of the form used in the definition of A-stability (namely $x' = kx$) means that the method of producing x_{n+1} from x_n will depend on the step size h and the constant k. As a matter of fact, for functions of this type, it will invariably depend on their product hk. Therefore, we can say that $x_{n+1} = u(hk)x_n$ for some function u, with the specific function depending on the numerical scheme used. It can clearly be seen that if $|u(hk)| < 1$, then $|x_{n+1}| < |x_n|$, and the simulated solution x will go to zero. The stability region for a numerical scheme is the set of complex numbers z such that $|u(z)| < 1$, with the implication that for a specified test function $x' = kx$ and step size h, we can check if the numerical scheme defined by $u(z) = u(hk)$ produces a solution that converges to zero. It follows that an A-stable numerical integration scheme is one for which the stability region includes the entire left half of the complex plane (i.e. all complex numbers with negative real parts), as that means that a simulation of $x' = kx$ will converge to zero for any k with negative real part, regardless of the step size h.

Let's see an example of this. We have previously attempted to solve a stiff equation using Euler's method, so we'll find the stability region of Euler's method first. The recurrence relation defining Euler's method is $x_{n+1} = x_n + hf(x_n)$, so for $f(x) = kx$, we have $x_{n+1} = x_n + hkx_n = x_n(hk + 1)$. It follows that the function used for finding the stability region is $u(hk) = hk + 1$, or $u(z) = z + 1$. Therefore, the actual stability region for Euler's method is $\{z \in \mathbb{C} : |z + 1| < 1\}$. This does not encompass all complex numbers with negative real parts, so Euler's method is not A-stable. We can also see that for $h = 0.1$ and $k = -100$, we get $hk = -10$ and $|hk + 1| = 9 > 1$, confirming what we saw earlier that Euler's method with a step size of $h = 0.1$ cannot successfully integrate the ODE $x' = -100x$. (In fact, the values taken by x during our simulations went up by a factor of 9 with every step, as the stability region calculations show.)

How about the fourth-order Runge-Kutta method? Expanding out the various coefficients that make up the method give us the following recurrence relation for x_{n+1} as a function of x_n:

$$u_{\text{RK4}}(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} \tag{33}$$

Note that this agrees with the Taylor expansion up to the fourth-order term, which is necessary for RK4 to be a fourth-order numerical integration scheme. The region of stability for RK4 is the area in which $|u_{\text{RK4}}(z)| < 1$. Because $u_{\text{RK4}}(z)$ is a fourth-degree polynomial function of z, it would be hard to graph this by hand (although you could do so using mathematical software). However, the most important thing is that the stability region does not cover all complex numbers with negative real parts. This means that RK4 is also not A-stable, and therefore that you will need very small step sizes to accurately capture the dynamics of very stiff systems using RK4. For the example that we used before with $z = hk = 0.1 \cdot -100 = -10$, we get the following:

$$|u_{\text{RK4}}(-10)| = 1 - 10 + \frac{100}{2} - \frac{1000}{6} + \frac{10000}{24} = 291 > 1 \tag{34}$$

This one surprisingly does even worse than Euler's method, although it would be more manageable if we decreased the step size. (Remember that if $z = hk < 1$, then $z^n < 1$ for $n > 1$.) What if we wanted a method that was A-stable, or in other words that it would integrate problems of this type well regardless of the choices of h and k? This can actually be done, and it can be done relatively easily. The numerical integration schemes that we would use to satisfy such a constraint are all what we call "implicit methods", so named because when evaluating x_{n+1}, they do so using an implicit formula that includes x_{n+1} and x_n instead of providing x_{n+1} as an explicit function of x_n. One extremely straightforward implicit method is the implicit Euler method, also called the backward Euler method. This is nearly identical to the (forward) Euler method, but with one very important distinction. Remember that the forward Euler method defines x_{n+1} as being $x_{n+1} = x_n + hf(x_n)$, or $x_{n+1} = x_n + hf(t_n, x_n)$ for a time-dependent ODE $x' = f(t, x)$. The implicit formula for the backward Euler method is as follows:

$$x_{n+1} = x_n + hf(t_{n+1}, x(t_{n+1})) = x_n + hf(t_{n+1}, x_{n+1}) \tag{35}$$

The derivation of this comes from the concept of integrating $\frac{dx}{dt}$ to get x. If we take a definite integral of $\frac{dx}{dt} = f(t, x)$ from time t_n to time $t_{n+1} = t_n + h$, then we can evaluate it to get the following:

$$\int_{t_n}^{t_{n+1}} f(t, x(t))\ dt = x(t_{n+1}) - x(t_n) = x_{n+1} - x_n \tag{36}$$

However, we can also use a very simple Riemann sum (just one rectangle) to approximate the integral above. Taking the value of the function f at the left-hand bound of the interval would just result in the forward Euler method, so we'll use the value on the right-hand bound. Since the length of the interval that we are integrating over is h, we get the following approximation:

$$\int_{t_n}^{t_{n+1}} f(t, x(t))\ dt \approx hf(t_{n+1}, x(t_{n+1})) = hf(t_{n+1}, x_{n+1}) \tag{37}$$

Putting these together yields the formula for the backward Euler method, as described above. As this is an implicit formula, we will need to use the specific form of f to isolate x_{n+1}. Obviously, this may be challenging if f is complicated. Is the backward Euler method A-stable? Let's check the function defining its stability region. If we assume that f takes the form $f(x) = kx$, we get the following:

$$x_{n+1} = x_n + hf(x_{n+1}) \implies x_{n+1}(1 - hk) = x_n \implies u(z) = (1 - z)^{-1} \tag{38}$$

We can see that $|u(z)| < 1$ when $1 < |1 - z|$. This is true for all z with negative real parts, and even most z with positive real parts. Therefore, the backward Euler method is A-stable, although it may produce other inaccuracies depending on the step size chosen. However, we do have better options available to us. Since Euler's method is based on the rectangle rule for approximating an integral, maybe a better method than the rectangle rule would produce a better numerical integration scheme. How about the trapezoid rule? If we perform the same steps as we did to derive the backward Euler method, but approximate the integral of $f(t, x(t))$ using the trapezoid rule instead of the rectangle rule, we get the following formula relating x_{n+1} and x_n:

$$x_{n+1} = x_n + \frac{h}{2}\left(f(t_n, x_n) + f(t_{n+1}, x_{n+1})\right) \tag{39}$$

This is the trapezoidal method for integrating ODEs, and it is also an implicit method. Note that we're averaging the values of f at t_n and t_{n+1}, hence the factor of $\frac{1}{2}$ (in addition to the step size h). This makes the trapezoidal method a "multistep method", since the equation $x_{n+1} - x_n$ that defines how much our solution will change by depends on the value of x at multiple different locations. So, how does this method do when evaluating stiff problems? We will calculate the region of stability for it to find out. If we assume that $f(x) = kx$, our formula for the trapezoidal method can be rewritten as the following:

$$x_{n+1} = x_n + \frac{h}{2}(kx_n + kx_{n+1}) \tag{40}$$

Solving for x_{n+1} yields the following:

$$x_{n+1} = \frac{1 + \frac{1}{2}hk}{1 - \frac{1}{2}hk} x_n \implies u(z) = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z} \tag{41}$$

In order for $|u(z)|$ to be less than 1, we need that $|1 + \frac{1}{2}z| < |1 - \frac{1}{2}z|$. However, this is true whenever z is closer to -1 than to 1, or in other words the entire left half of the complex plane. From this, we get the result that the trapezoidal method is A-stable. We also get the stronger condition that for any ODE $x' = kx$, the simulated solution of that ODE using the trapezoidal method will converge to zero if and only if the actual solution does.

So, we now have a method that has been shown to work very well for stiff dynamical systems. The trapezoidal method is second-order; its local truncation error can be found to be $\mathcal{O}(h^3)$, and therefore its global truncation error is $\mathcal{O}(h^2)$. Can we improve upon this to get better accuracy (like we improved on the forward Euler method to get RK4), while keeping A-stability? The answer to this is actually that we can't. Explicit multistep methods cannot be A-stable, and implicit multistep methods that are third-order or higher can also never be A-stable. This result was proved by Germund Dahlquist in 1963 and is known as the second Dahlquist barrier. (This means that you have actually seen a result proved in the last 50 years in this class!)

Now that we know a good way to approach stiff dynamical systems, where can we expect to encounter them in real life? One important class of dynamical systems that may be stiff is those that contain a separation of timescales, when some processes that make up the system happen orders of magnitude more quickly than others. Here is a simple example of this, for ε a very small constant:

$$\begin{cases} \frac{dx}{dt} = f(x, y) \\ \varepsilon \frac{dy}{dt} = g(x, y) \end{cases} \tag{42}$$

Some chemical reaction networks contain ODEs that are on a faster timescale than the rest of the system like this, and population dynamics in which some species have much faster life cycles than others can also be modelled in this way. Additionally, most dynamical systems used in neuroscience are stiff. This is because they model (among other things) the electrical potential found in a neuron, which undergoes a large spike when the neuron fires. This, however, is a topic for next week.