Lecture 9
Linear ODE and Numerical Methods

9.1 Linear Differential Equations.

If $A : V \to V$ is a continuous linear operator on V then we can also consider A as a vector field on V. The corresponding autonomous ODE, $\frac{dx}{dt} = Ax$ is called a *linear differential equation on* V.

Denote by $C(\mathbf{R}, V)$ the continuous maps of $\mathbf{R}$ into V, and let $F = F^A_{x_0}$ be the map of $C(\mathbf{R}, V)$ to itself defined by $F(x)(t) := x_0 + \int_0^t A(x(s))\,ds$. Since A is linear, this can also be written as $F(x)(t) := x_0 + A\int_0^t x(s)\,ds$. We know that the solution of the IVP with initial value x_0 is just the unique fixed point of F, so let's try to find it by successive approximations starting from the constant path $x^0(t) = x_0$. If we recall that the sequence of successive approximations, x^n, is defined recursively by $x^{n+1} = F(x^n)$, then an elementary induction gives $x^n(t) = \sum_{k=0}^n \frac{1}{k!}(tA)^k x_0$, suggesting that the solution to the initial value problem should be given by the limit of this sequence, namely the infinite series $\sum_{k=0}^\infty \frac{1}{k!}(tA)^k x_0$. Now (for obvious reasons) given a linear operator T acting on V, the limit of the infinite series of operators $\sum_{k=0}^\infty \frac{1}{k!}T^k$ is denoted by e^T or $\exp(T)$ so we can also say that the solution to our IVP should be $e^{tA}x_0$.

The convergence properties of the series for $e^T x$ follow easily from the Weierstrass M-test. If we define $M_k = \frac{1}{k!}|||\,T\,|||^k r$, then $\sum M_k$ converges to $e^{|||\,T\,|||}r$, and since $\left\|\frac{1}{k!}T^k x\right\| < M_k$ when $\|x\| < r$, it follows that $\sum_{k=0}^\infty \frac{1}{k!}T^k x$ converges absolutely and uniformly to a limit, $e^T x$, on any bounded subset of V.

▷ **9.1—Exercise 1.** Provide the details for the last statement.
(Hint: Since the sequence of partial sums $\sum_{k=0}^n M_k$ converges, it is Cauchy, i.e., given $\epsilon > 0$, we can choose N large enough that $\sum_m^{m+k} M_k < \epsilon$ provided $m > N$. Now if $\|x\| < r$, $\left\|\sum_{k=0}^{m+k} \frac{1}{k!}T^k x - \sum_{k=0}^m \frac{1}{k!}T^k x\right\| < \sum_m^{m+k} M_k < \epsilon$, proving that the infinite series defining $e^T x$ is uniformly Cauchy and hence uniformly convergent in $\|x\| < r$.)

Since the partial sums of the series for $e^T x$ are linear in x, so is their limit, so e^T is indeed a linear operator on V.

Next observe that since a power series in t can be differentiated term by term, it follows that $\frac{d}{dt}e^{tA}x_0 = Ae^{tA}x_0$, i.e., $x(t) = e^{tA}x_0$ is a solution of the ODE $\frac{dx}{dt} = Ax$. Finally, substituting zero for t in the power series gives $e^{0A}x_0 = x_0$. This completes the proof of the following proposition.

9.1.1 Proposition. *If A is a linear operator on V then the solution of the linear differential equation $\frac{dx}{dt} = Ax$ with initial condition x_0 is $x(t) = e^{tA}x_0$.*

As a by-product of the above discussion we see that a linear ODE $\frac{dx}{dt} = Ax$ is complete, and the associated flow ϕ_t is just e^{tA}. By a general fact about flows it follows that $e^{(s+t)A} = e^{sA}e^{tA}$, and $e^{-A} = (e^A)^{-1}$, so $\exp : A \mapsto e^A$ is a map of the vector space $L(V)$

of all linear maps of V into the group $\boldsymbol{GL}(V)$ of invertible elements of $L(V)$ and for each $A \in L(V)$, $t \mapsto e^{tA}$ is a homomorphism of the additive group of real numbers into $\boldsymbol{GL}(V)$.

▷ **9.1—Exercise 2.** Show more generally that if A and B are commuting linear operators on V then $e^{A+B} = e^A e^B$. (Hint: Since A and B commute, the Binomial Theorem is valid for $(A+B)^k$, and since the series defining e^{A+B} is absolutely convergent, it is permissible to rearrange terms in the infinite sum. For a different proof, show that $e^{tA}e^{tB}x_0$ satisfies the initial value problem $\frac{dx}{dt} = (A+B)x$, $x(0) = x_0$, and use the uniqueness theorem.)

▷ **9.1—Exercise 3.** Now assume that V is a finite dimensional inner-product space. Show that $\left(e^A\right)^* = e^{A^*}$. (Hint: Recall that $(AB)^* = B^*A^*$.) Show that if A is skew-adjoint (meaning $A^* = -A$) than e^A is in the orthogonal group. Conversely show that if e^{tA} is in the orthogonal group for all $t \in \mathbf{R}$ then A must be skew-adjoint.

▷ **9.1—Exercise 4.** Let's say that $A \in L(V)$ is tangent to the orthogonal group $\boldsymbol{O}(V)$ at the identity if there is a differentiable path $\sigma : (a,b) \to L(V)$ defined near 0, such that $\sigma(t) \in \boldsymbol{O}(V)$ for all t, $\sigma(0) = I$, and $\sigma'(0) = A$. From what we have just seen, every skew-adjoint A is tangent to $\boldsymbol{O}(V)$ at I. Show that conversely any A that is tangent to $\boldsymbol{O}(V)$ at I must be skew-adjoint. (Hint: Differentiate the identity $\sigma(t)\sigma(t)^* = I$ at $t = 0$.)

9.2 Numerical Solutions of ODE.

Once we go beyond the linear case, only a few rather special initial value problems can be solved in closed form using standard elementary functions. For the general case it is necessary to fall back on constructing a solution numerically. But what algorithm should we use? A natural first guess is successive approximations. But while that is a powerful theoretical tool for studying general properties of initial value problems—and in particular for proving existence and uniqueness—it turns out to be so inefficient as an approach to calculating numerical solutions, that it is rarely used.

In fact there is no unique answer to the question of what numerical algorithm to use for solving ODEs, for there is no one method that is "best" in all situations. There are integration routines that are fast and accurate when used with the majority of equations one meets. Perhaps the most popular of these is the fourth order Runge-Kutta algorithm (which we will consider below). But there are many special situations that require a more sophisticated approach. Indeed, this is still a very active area of research, and there are literally dozens of books on the subject.

The initial goal will be to give you a quick first impression of how numerical methods work by describing one of the oldest numerical approaches to solving an initial value problem, the so-called "Euler Method". While rarely a good choice for practical computation, it is intuitive, simple, and effective, and it is also the basis for some of the more sophisticated algorithms. This makes it an excellent place to become familiar with the basic concepts that enter into the numerical integration of ODE.

In what follows we will suppose that X is a C^1 time-dependent vector field on V, and given t_0 in $\mathbf{R}$ and x^0 in V we will denote by $\sigma(X, x^0, t_0, t)$ the maximal solution, $x(t)$, of the differential equation $\frac{dx}{dt} = X(x, t)$ satisfying the initial condition $x(t_0) = x^0$. The goal in the numerical integration of ODE is to devise effective methods for approximating $x(t)$ on an interval $I = [t_0, T]$. The strategy that many methods use is to interpolate N equally spaced gridpoints $t_1, \ldots t_N$ in the interval I, defined by $t_k := t_0 + k\Delta t$ with $\Delta t = \frac{T-t_0}{N}$, and then use some rule to define values $x^1, \ldots, x^N$ in V, in such a way that when N is large each x^k is close to the corresponding $x(t_k)$. The quantity $\max_{1\leq k\leq N} \left\|x^k - x(t_k)\right\|$ is called the *global error* of the algorithm, and if it converges to zero as N tends to infinity (for every choice of X, t_0, x^0, and T), then we say that we have a *convergent algorithm*. Euler's Method is a convergent algorithm of this sort.

One common way to construct the algorithm that produces the values $x^1, \ldots, x^N$ uses a recursion based on a so-called "stepping procedure", namely a function, $\Sigma(X, x^0, t_0, \Delta t)$, having as inputs:

1) a time-dependent vector field X on V,

2) an initial condition x^0 in V,

3) an initial time t_0 in $\mathbf{R}$, and

4) a "time-step" Δt in $\mathbf{R}$,

and with output a point of V that for small Δt approximates $\sigma(X, x^0, t_0, t_0 + \Delta t)$ well. More precisely, the so-called "local truncation error", defined by

$$\left\|\sigma(X, x^0, t_0, t_0 + \Delta t) - \Sigma(X, x^0, t_0, \Delta t)\right\|,$$

should approach zero at least quadratically in the time-step Δt. Given such a stepping procedure, the approximations x^k of the $x(t_k)$ are defined recursively by $x^{k+1} = \Sigma(X, x^k, t_k, \Delta t)$. Numerical integration methods that follow this general pattern are referred to as finite difference methods.

9.2.1 Remark. There are two sources that contribute to the global error, $\left\|x^k - x(t_k)\right\|$. First, each stage of the recursion will give an additional local truncation error added to what has already accumulated up to that point. But, in addition, after the first step, there will be an error because the recursion uses $\Sigma(X, x^k, t_k, \Delta t)$ rather than the unknown $\Sigma(X, x(t_k), t_k, \Delta t)$. (In practice there is a third source of error, namely machine round-off error from using floating-point arithmetic. We will usually ignore this and pretend that our computers do precise real number arithmetic, but there are situations where it is important to take it into consideration.)

For Euler's Method the stepping procedure is simple and natural. It is defined by:

Euler Step

$\Sigma^E(X, x^0, t_0, \Delta t) := x^0 + \Delta t\, X(x^0, t_0)$.

It is easy to see why this is a good choice. If as above we denote $\sigma(X, x^0, t_0, t)$, by $x(t)$, then by Taylor's Theorem,

$$\begin{aligned} x(t_0 + \Delta t) &= x(t_0) + \Delta t\, x'(t_0) + O(\Delta t^2) \\ &= x^0 + \Delta t\, X(x^0, t_0) + O(\Delta t^2) \\ &= \Sigma^E(X, x^0, t_0, \Delta t) + O(\Delta t^2), \end{aligned}$$

so $\big\|\sigma(X, x^0, t_0, t_0 + \Delta t) - \Sigma(X, x^0, t_0, \Delta t)\big\|$, the local truncation error for Euler's Method, does go to zero quadratically in Δt. When we partition $[0, T]$ into N equal parts, $\Delta t = \frac{T - t_0}{N}$, each step in the recursion for computing x^k will contribute a local truncation error that is $O(\Delta t^2) = O(\frac{1}{N^2})$. Since there are N steps in the recursion and at each step we add $O(\frac{1}{N^2})$ to the error, this suggests that the global error will be $O(\frac{1}{N})$, and hence will go to zero as N tends to infinity. However, because of the remark above, this is not a complete proof that Euler's Method is convergent, and we will not give the details of the rigorous argument.

▷ **9.2—Exercise 1.** Show that Euler's Method applied to the initial value problem $\frac{dx}{dt} = x$ with $x(0) = 1$ gives $\lim_{N\to\infty}(1 + \frac{t}{N})^N = e^t$.

Two positive features of Euler's method are it's intuitiveness and the ease with which it can be programmed. Beyond that there is little to recommend it as a practical method for solving real-world problems. It requires very small time steps to get reasonable accuracy, making it too slow for serious applications, and in fact it is rarely used except for pedagogical purposes.

On the other hand, as we have already said, there is one excellent general purpose finite difference method for solving IVPs, and it goes by the name Runge-Kutta—or more properly the fourth order Runge-Kutta Method—since there is a whole family of Runge-Kutta methods. The stepping procedure for fourth order Runge-Kutta is:

Runge-Kutta Step

$\Sigma^{RK^4}(X, x_0, t_0, \Delta t) := x_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$, where:
$k_1 = \Delta t\, X(x_0, t_0)$
$k_2 = \Delta t\, X(x_0 + \frac{1}{2}k_1, t_0 + \frac{\Delta t}{2})$
$k_3 = \Delta t\, X(x_0 + \frac{1}{2}k_2, t_0 + \frac{\Delta t}{2})$
$k_4 = \Delta t\, X(x_0 + k_3, t_0 + \Delta t)$

It is of course natural to ask where this formula comes from. But if you recall Simpson's Rule then the above should not look all that unreasonable, and indeed if $f(x, t) = \phi(t)$ then recall that the solution of the IVP reduces to the integral of ϕ and in this case the Runge-Kutta formula reduces precisely to Simpson's Rule. But even better, like Simpson's Rule, Runge-Kutta is fourth order, meaning that the local truncation error goes to zero as the fifth power of the step-size, and the global error as the fourth power. So if for a fixed step-size we have attained an accuracy of 0.1, then with one-tenth the step-size (and so ten times the number of steps and ten times the time) we can expect an accuracy of 0.00001, whereas with the Euler method, ten times the time would only increase accuracy from 0.1 to 0.01.

▷ **9.2—Exercise 2.** Write a Matlab M-File function `Euler(X,x0,T,n)`that estimates $x(T)$, the solution at time T of the initial value problem $\frac{dx}{dt} = X(x)$, $x(0) = x0$ by applying the Euler stepping method to the interval $[0, T]$ with n time steps. Similarly write such a function `RungeKutta(X,x0,T,n)` that uses the Runge-Kutta stepping method. Make some experiments using the case $V = \mathbf{R}$, $\frac{dx}{dt} = x$ with $x0 = 1$ and $T = 1$, so that $x(T) = e$. Check how large n has to be to get various degrees of accuracy using the two methods.