▷Project 4. Implement Euler's Method and Runge-Kutta in Matlab

1.1 Review of Stepping Methods for Solving IVPs.

In what follows 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*.

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.

1.1.1 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, and this can be proved rigorously, so Euler's Method is convergent.

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:
$k_1 = \Delta t\, X(x_0, t_0)$
$k_2 = \Delta t\, X(x_0 + \frac{1}{2}k_1, t_0 + \frac{\Delta t}{2})$
$k_3 = \Delta t\, X(x_0 + \frac{1}{2}k_2, t_0 + \frac{\Delta t}{2})$
$k_4 = \Delta t\, X(x_0 + k_3, t_0 + \Delta t)$

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.

Fourth Matlab Project.

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.