§11.2. Numerical methods

In this section we briefly study how to solve systems of ODEs numerically (i.e. on a computer)

§11.2.1. Euler's method

This is the simplest method of solving a system of ODEs numerically. (It's not the most efficient one: in practice people often use the Runge-Kutta method.)

We are solving $\boxed{\vec{y}' = \vec{F}(t, \vec{y})}$

with initial condition $\boxed{\vec{y}(0) = \vec{y}^0}$

We fix a step size $h > 0$.

The smaller the h, the better is our approximation to the solution. But small h require more time for the computation.

The method works as follows:
define $\vec{y}^0$ to be the initial condition
and compute $\vec{y}^1, \vec{y}^2, \ldots$ iteratively by the formula

$$\boxed{\vec{y}^{k+1} = \vec{y}^k + h\vec{F}(t_k, \vec{y}^k)}$$

Where $t_k = h \cdot k$
(i.e. $t_0 = 0, t_1 = h, t_2 = 2h, \ldots$)
Then $\vec{y}^k$ is an approximation for $\vec{y}(t_k)$.

Why does this work?

Example: $\vec{y}' = A\vec{y}$, $A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$

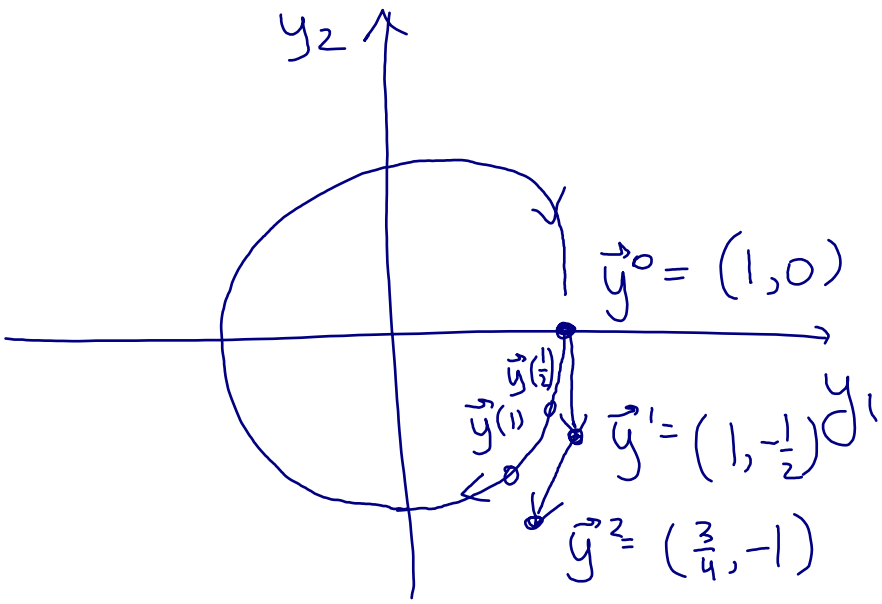
So $\vec{F}(y_1, y_2) = (y_2, -y_1)$.

Solving with $\vec{y}(0) = (1, 0)$.

Take $h = 1/2$. Then we compute

$$\vec{y}^1 = \vec{y}^0 + h\vec{F}(\vec{y}^0) = (1,0) + \tfrac{1}{2}(0,-1) = (1, -\tfrac{1}{2})$$

$$\vec{y}^2 = \vec{y}^1 + h\vec{F}(\vec{y}^1) = (1, -\tfrac{1}{2}) + \tfrac{1}{2}(-\tfrac{1}{2}, -1) = (\tfrac{3}{4}, -1)$$



The method works because for small h,

$$\vec{y}(t_{k+1}) = \vec{y}(t_k + h) \approx \vec{y}(t_k) + h \cdot \vec{y}'(t_k)$$
$$= \vec{y}(t_k) + h \cdot F(t_k, \vec{y}(t_k)).$$

The method would be exact if $\approx$ above was an equality

Error bound: For bounded $t_k = h \cdot k$ we have $\boxed{|\vec{y}(t_k) - \vec{y}^k| \leq Ch}$

where C is some constant independent of h.

Informal explanation: each time $\approx$ has error of order h^2, and we incur this error $\sim 1/h$ times which is the number of steps

§11.2.2. An example: the SEIR model

Recall from §8.1 the SEIR model

$$\begin{cases} S' = -\beta S \cdot I \\ E' = \beta S \cdot I - \alpha E \\ I' = \alpha E - \gamma I \\ R' = \gamma I \end{cases}$$

For COVID-19 by some estimates $\alpha = 0.2$, $\gamma = 0.5$, $\beta = 1.75$ but social distancing reduces β.

S = Susceptible
E = Exposed
I = Infected
R = Recovered

} proportion of the population

Now I will show you a Matlab simulation (using Runge-Kutta) of the solution with the initial condition $S(0) = 1 - 10^{-4}$, $E(0) = 10^{-4}$, $I(0) = R(0) = 0$