Scientific computing | Week 9
Linear algebra and dynamical systems

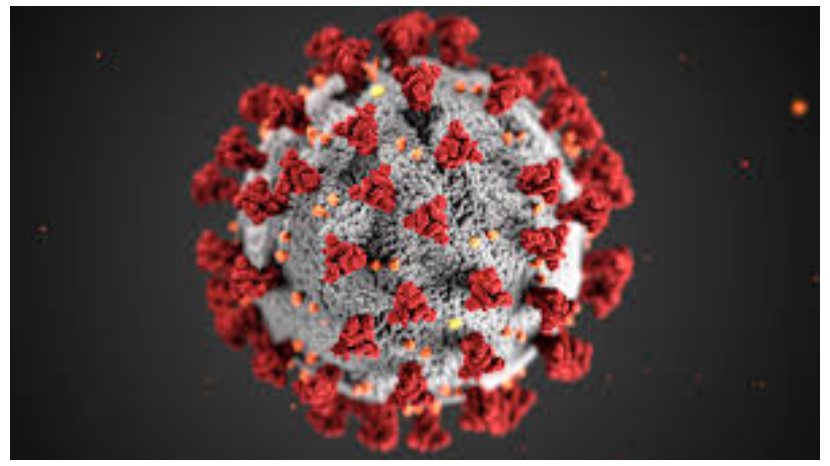
Volker Roth, Ivan Dokmanić

Why study differential equations?

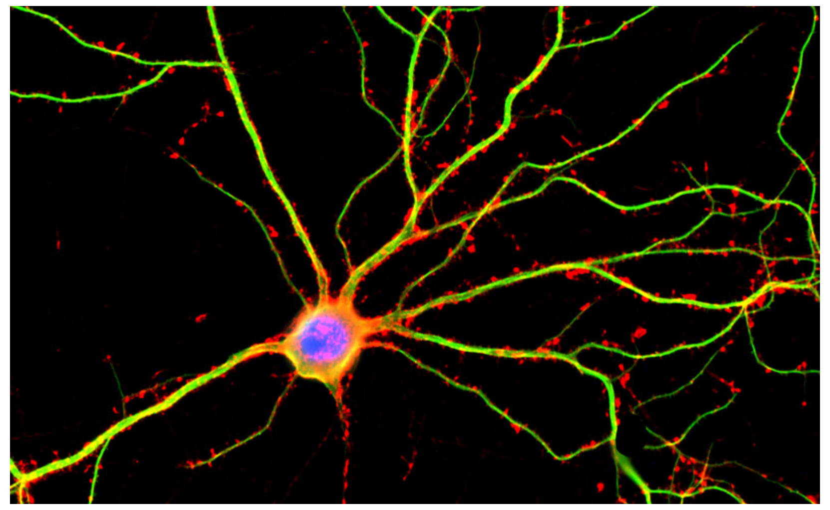
- But differential equations are so 20th century :-(

Life sciences

{Epi, Pan}demics

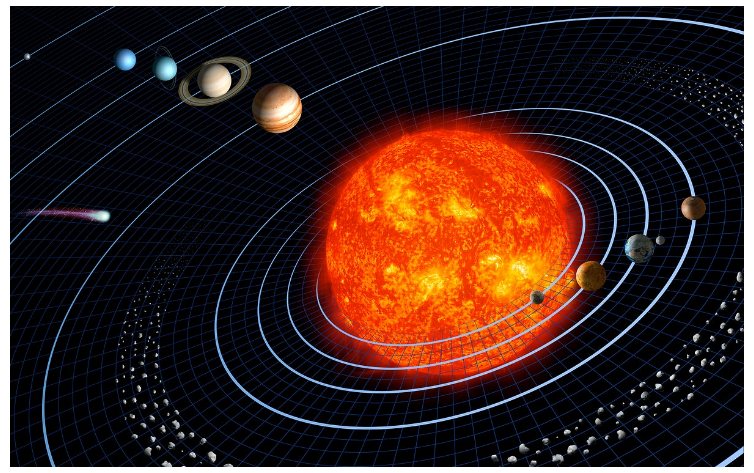


Neuroscience



Physics

Planetary dynamics



Turbulence

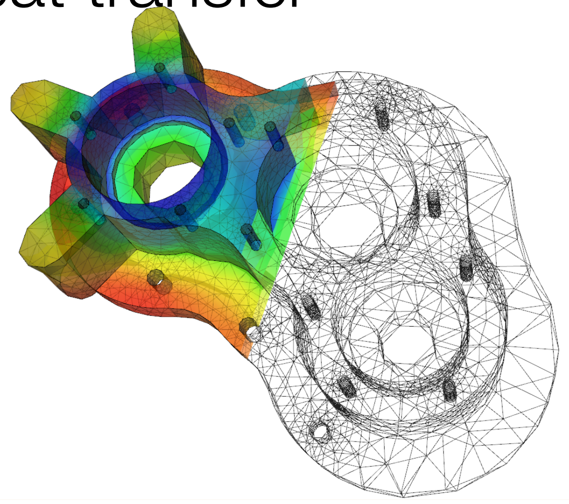
Environment

CO_2 concentration

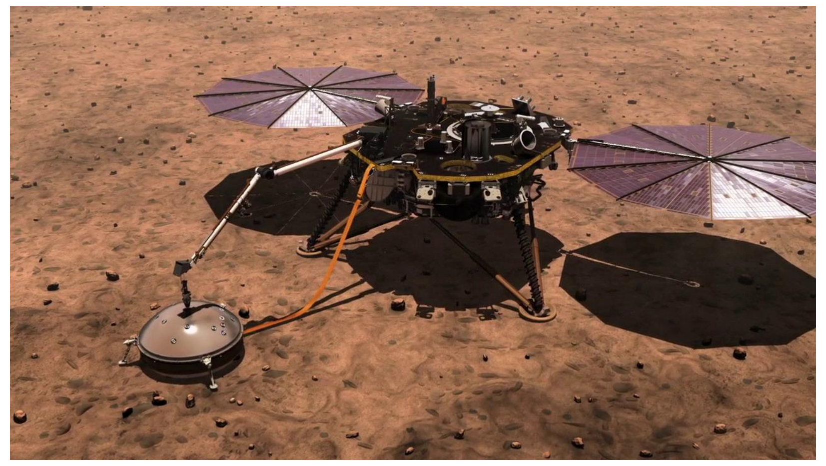
Ice melting

Engineering

Heat transfer



Trajectory design



Finance

Black–Scholes

Market crashes

Recap: linear ODEs with linear algebra

A system of first-order linear ODEs (homogeneous, constant-coefficient)

$$\frac{d\boldsymbol{u}}{dt} = A\boldsymbol{u} \qquad \boldsymbol{u}(t) \in \mathbb{R}^n \qquad A \in \mathbb{R}^{n\times n}$$

Entries of $\boldsymbol{u}$ model positions, velocities, CO_2 concentrations, …

Scalar case ($n = 1$)

$$u'(t) = \alpha u(t) \qquad u(t) = e^{\alpha t}u(0)$$

Vector case

$$\boldsymbol{u}(t) = e^{At}\boldsymbol{u}_0$$

The meaning of e^{At}

Defined via the Taylor (power) series

$$e^{At} := \sum_{k=0}^{\infty} \frac{(At)^k}{k!}$$

Use to check the solution

$$(e^{At})' := \sum_{k=1}^{\infty} \frac{kt^{k-1}A^k}{k!} = A\sum_{k=1}^{\infty} \frac{t^{k-1}A^{k-1}}{(k-1)!} = A\sum_{k=0}^{\infty} \frac{t^k A^k}{k!} = Ae^{At}$$

For **diagonalizable matrices**, we get a very simple rule

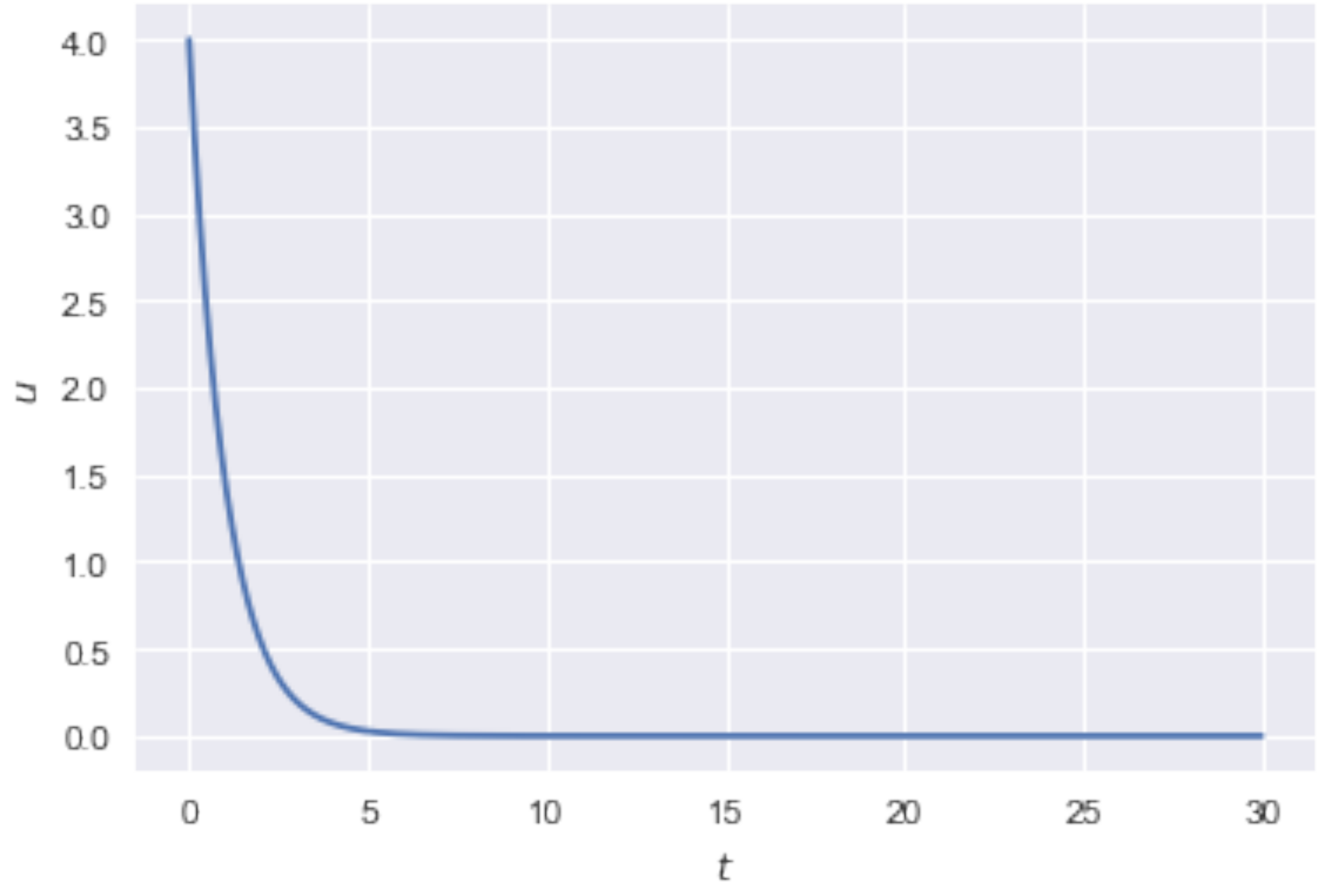
$$A = V \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix} V^{-1} \quad \Longrightarrow \quad e^{At} = V \begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & \ddots & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & e^{\lambda_n t} \end{bmatrix} V^{-1}$$

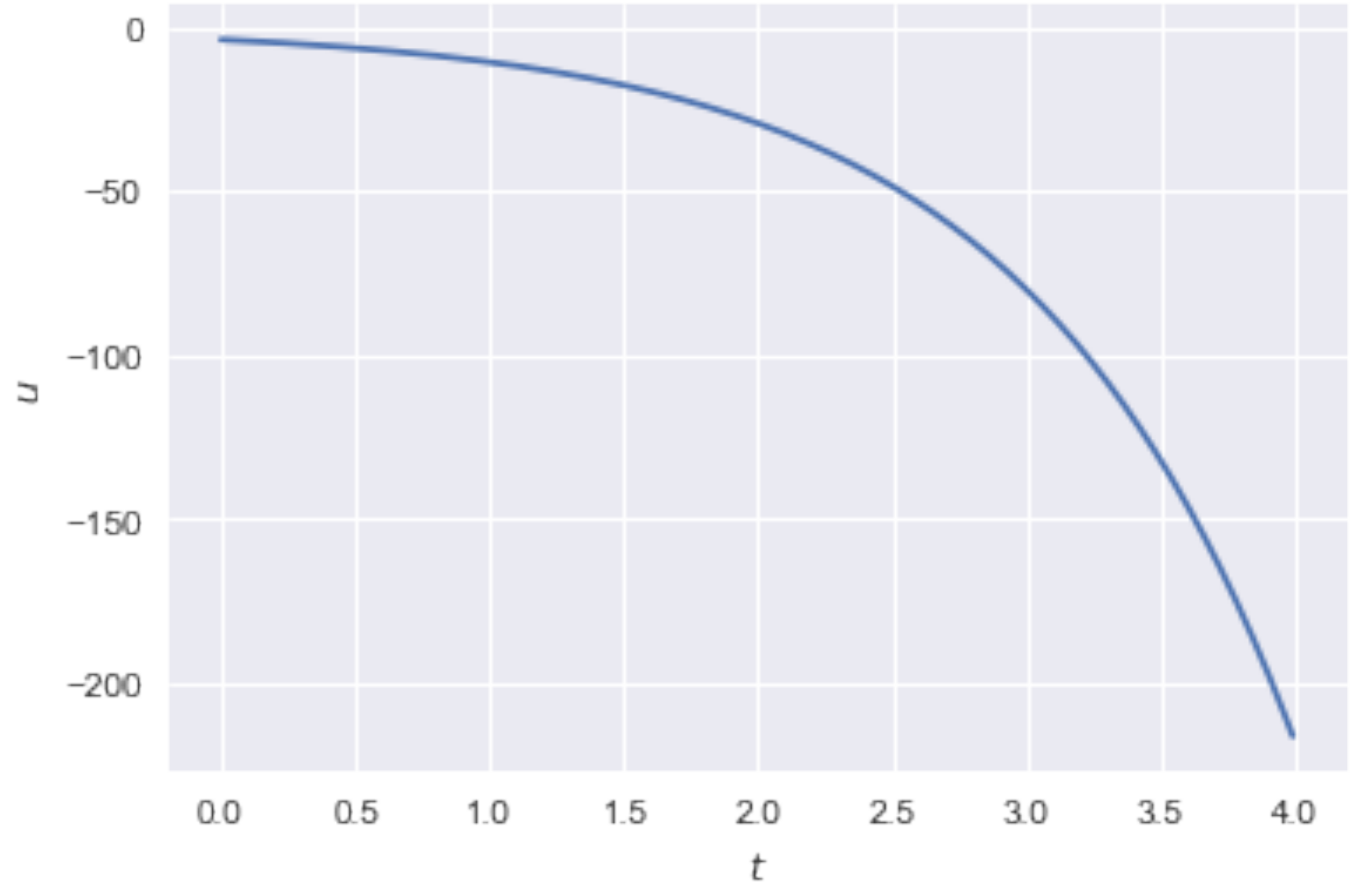
Behavior of first-order equations for $n = 1$

When α is a real number,

$$u'(t) = \alpha u(t) \qquad\qquad u(t) = e^{\alpha t}u(0)$$

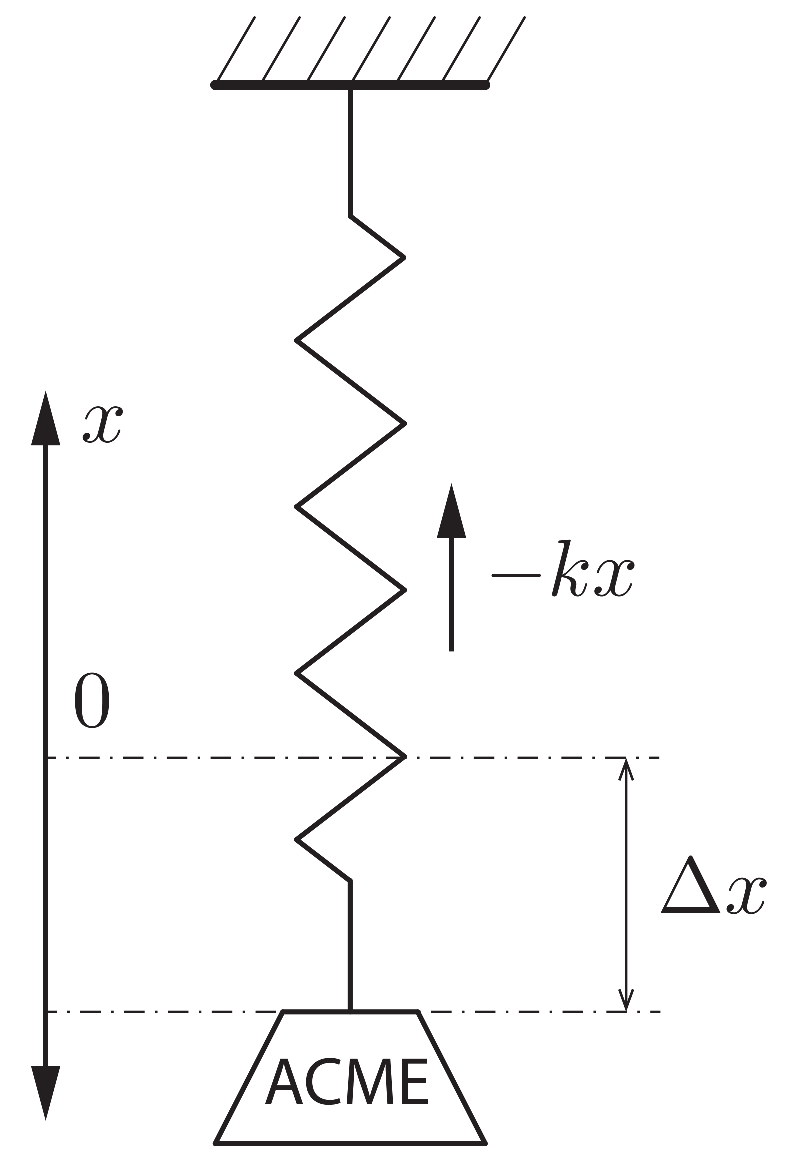
In this case the possible dynamics are quite boring… (but they can also be dangerous!)

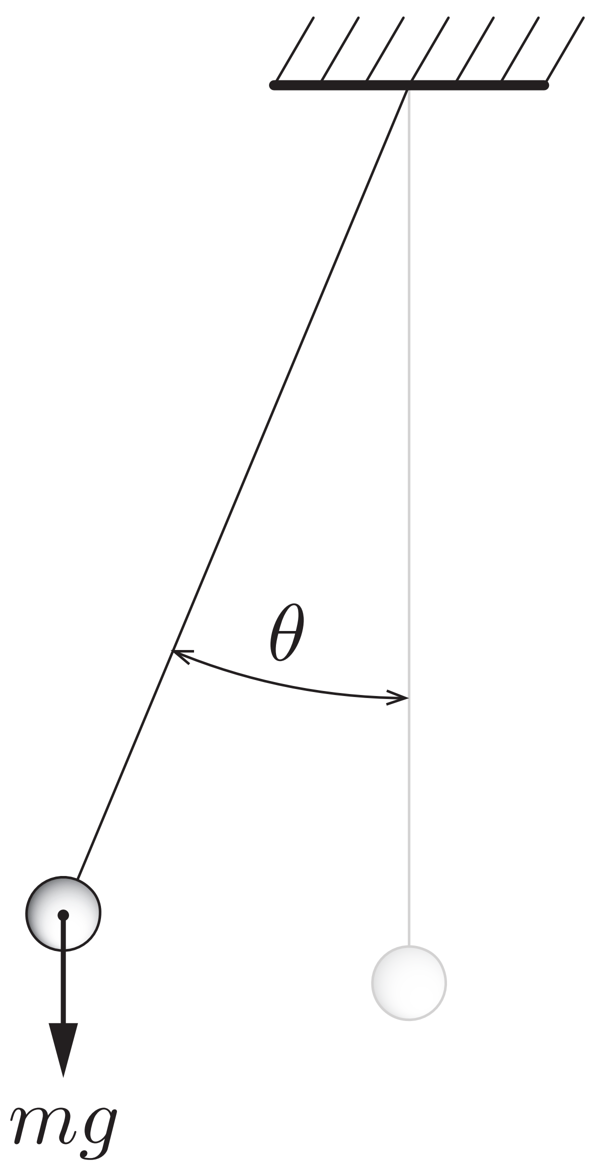


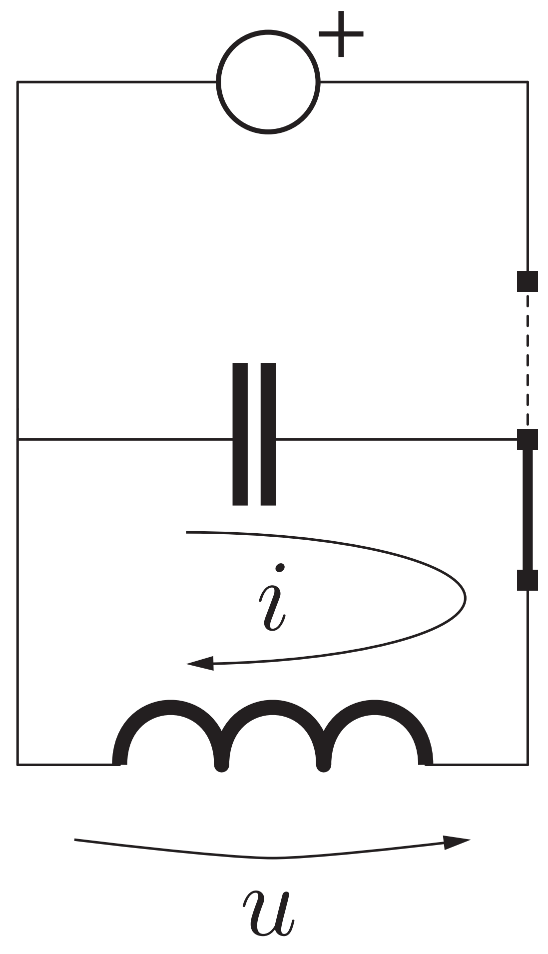


Not so boring: second-order differential equations

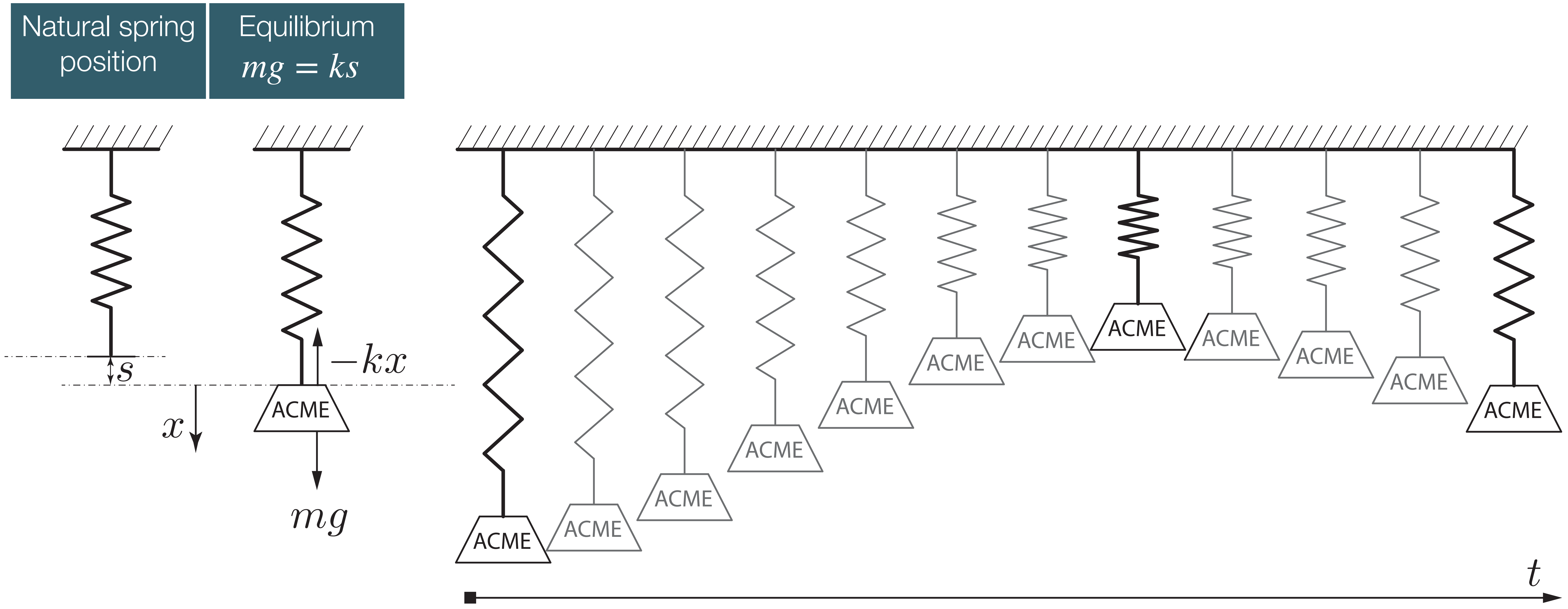
$$mx'' + bx' + kx = 0$$







Mass on a spring



$$F_G = mg \qquad F_S = -k(s + x)$$

Mass on a spring

Gravitational force

$$F_G = mg$$

Restoring force in the spring

$$F_S = -k(s + x)$$

Newton says $F_{tot} = ma = mx''$

$$F_{tot} = F_S + F_G \implies mx'' + kx = 0$$

$$\boxed{x'' = -\omega^2 x} \qquad k = \omega^2$$

A second-order linear ODE! Converting to first order lets us use linear algebra:

$$\boldsymbol{u} := \begin{bmatrix} x \\ x' \end{bmatrix} \qquad \frac{d\boldsymbol{u}}{dt} := \begin{bmatrix} x' \\ x'' \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x \\ x' \end{bmatrix}}_{\boldsymbol{u}}$$

Writing down the solution

By solving $\det(\lambda I - A) = 0$ we get the **eigenvalues** of A as

$$\lambda_1 = j\omega \qquad \lambda_2 = -j\omega$$

Solving $Av_{1,2} = \lambda_{1,2}v_{1,2}$ we further get the **eigenvectors**

$$\boldsymbol{v}_1 = \begin{bmatrix} 1 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} 1 \\ j\omega \end{bmatrix} \qquad \boldsymbol{v}_2 = \begin{bmatrix} 1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -j\omega \end{bmatrix}$$

Any solution can thus be written as (for some constants c_1 and c_2)

$$\boldsymbol{u}(t) = c_1 e^{j\omega t}\boldsymbol{v}_1 + c_2 e^{-j\omega t}\boldsymbol{v}_2$$

The constants c_1 and c_2 can be determined from two initial conditions (on x and x')

$$\boldsymbol{u}(0) = c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2$$

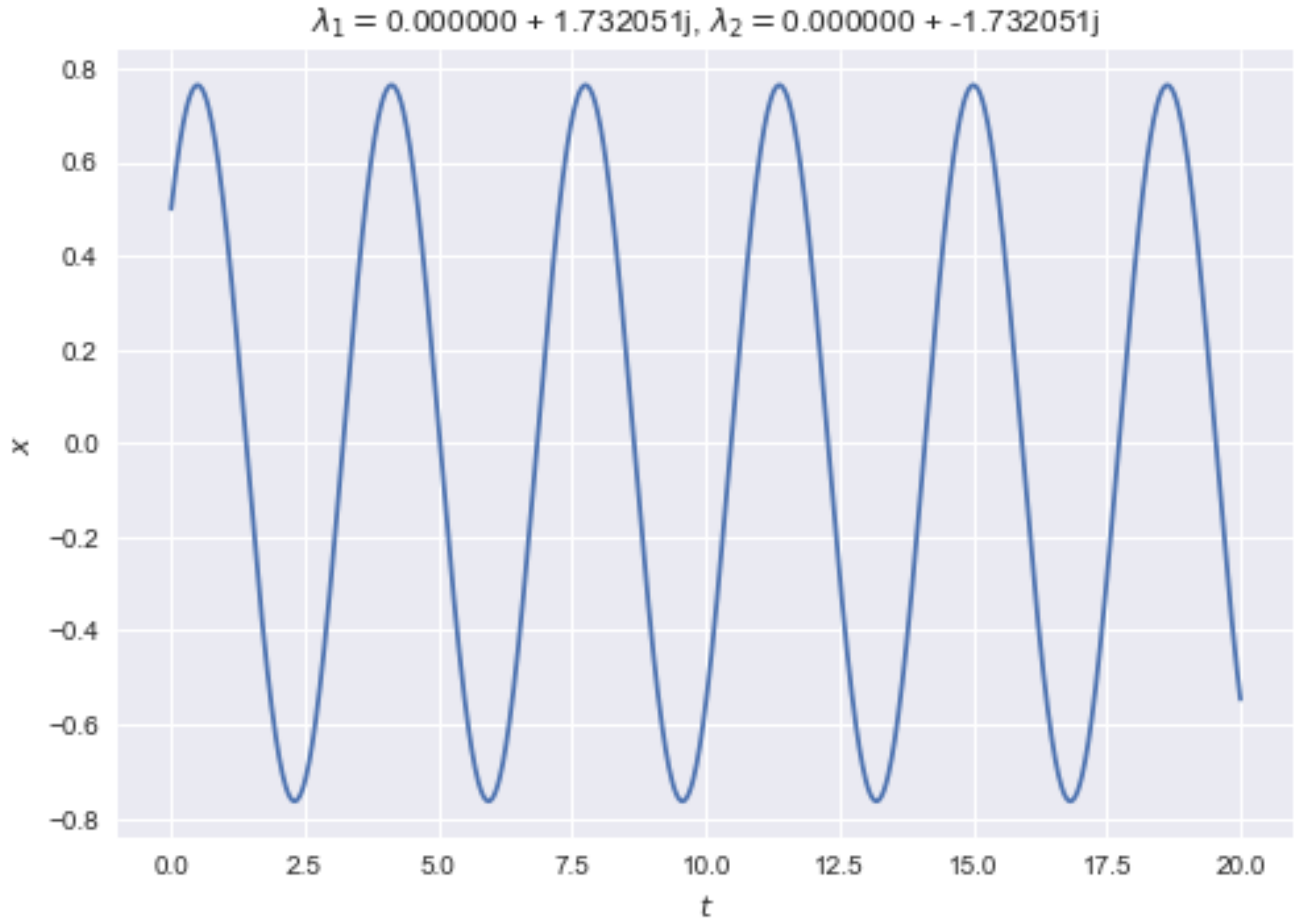
We get the familiar harmonic oscillations

$$x(t) = c_1 e^{j\omega t} + c_2 e^{-j\omega t}$$

$$\mathfrak{I}(x(t)) = 0\,\big|_{t=0} \implies \mathfrak{I}(c_1) = -\,\mathfrak{I}(c_2)$$

$$\mathfrak{I}(x(t)) = 0\,\big|_{t=\frac{\pi}{2}} \implies \mathfrak{R}(c_1) = \mathfrak{R}(c_2)$$

$$\implies c_1 = \bar{c}_2$$



So finally, for some real A and B (or α and ϕ)

$$x(t) = A\cos(\omega t) + B\sin(\omega t) = \alpha\sin(\omega t + \phi)$$

Damped oscillations with $F_D = -bx'$

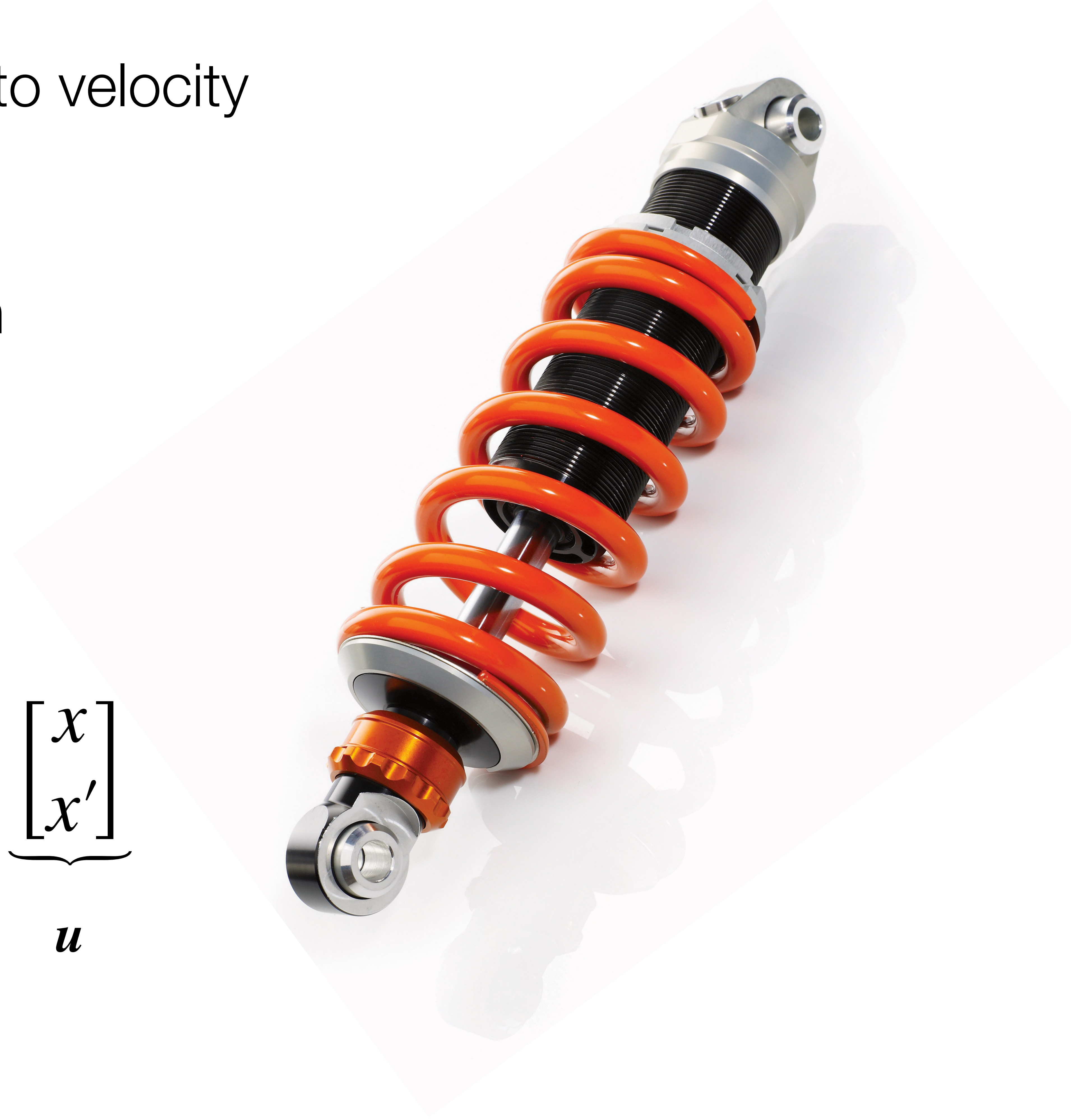
The force $F_D = -bx'$ describes damping, friction, proportional to velocity

$F_S + F_G + F_D = mx''$ now gives the full second-order equaton

$$mx'' + bx' + kx = 0$$

Rewrite again as a first-order system

$$\frac{d\mathbf{u}}{dt} := \begin{bmatrix} x' \\ x'' \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x \\ x' \end{bmatrix}}_{\mathbf{u}}$$



General solution

Solving $\det(\lambda I - A) = 0$ gives

$$\lambda_1 = \frac{-b + \sqrt{b^2 - 4mk}}{2m} \qquad \lambda_2 = \frac{-b - \sqrt{b^2 - 4mk}}{2m}$$

General solution

$$x(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t}$$

Behavior depends on the sign of the discriminant

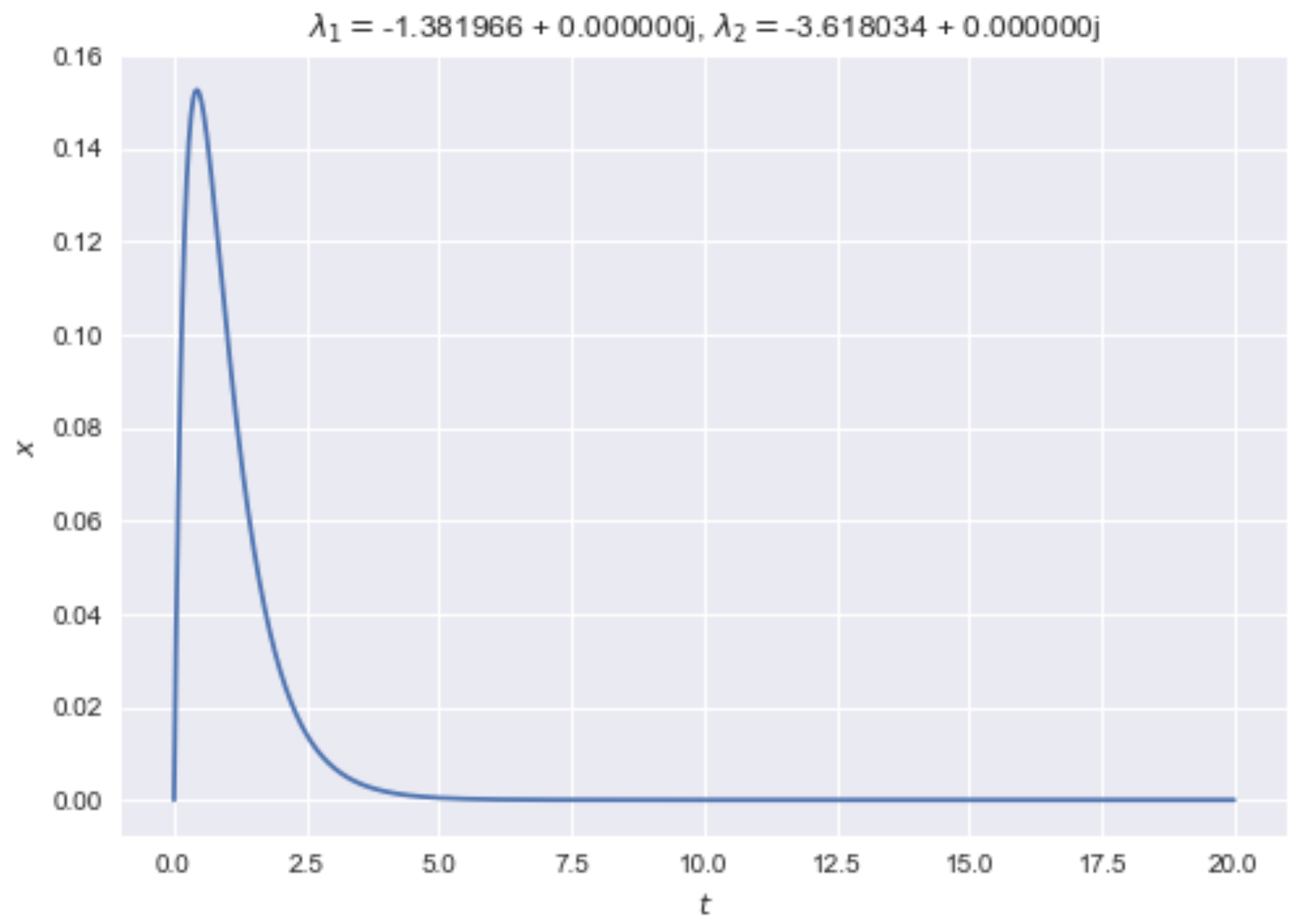
$$b^2 - 4mk \lesseqgtr 0$$

Different kinds of solutions

Overdamped

$$b^2 > 4mk \quad \lambda_{1,2} < 0$$

$$x(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t}$$

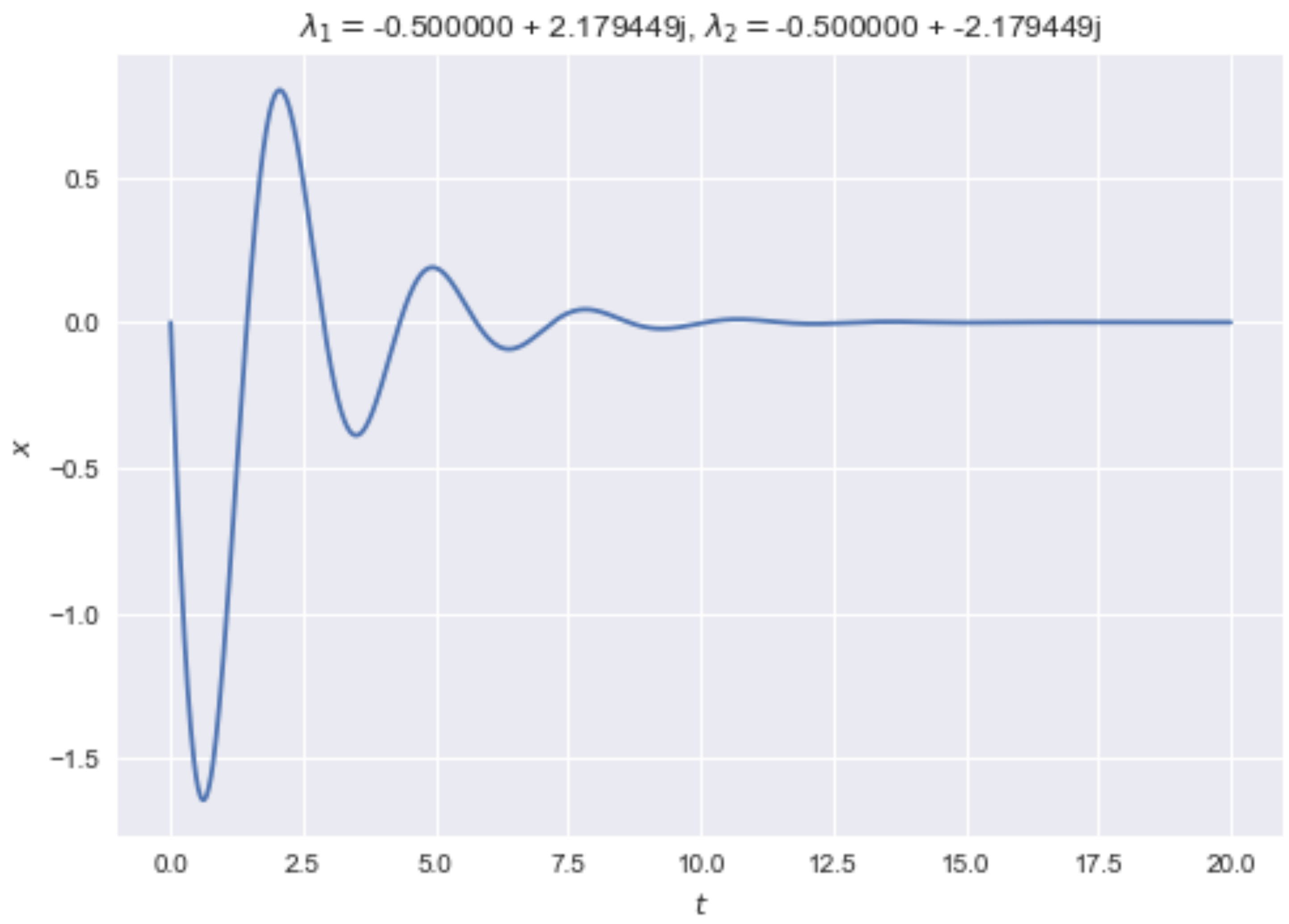


Underdamped

$$b^2 < 4mk$$

$$\Re(\lambda_1) = \Re(\lambda_2) < 0$$

$$x(t) = e^{-\alpha t}(A\cos(\omega t) + B\sin(\omega t))$$



Forced oscillations

So far: the right-hand side of the ODE is zero: **natural modes** of the spring-mass system

In most practical applications there is an **external forcing**

- A voltage source in a circuit
- Uneven road hits the wheels
- Greenhouse gas emissions

In our second-order linear case this is modeled as a right-hand side $f(t)$

$$mx''(t) + bx'(t) + kx(t) = f(t)$$

General solution to the non-homogeneous equation

A general solution to $mx''(t) + bx'(t) + kx(t) = f(t)$ can be written as

$$x(t) = c_1x_1(t) + c_2x_2(t) + x_p(t)$$

Solution to the **homogeneous** equation

$$mx'' + bx' + kx = 0$$

A **particular** solution

In a damped system, x_p dictates the long-term behavior—steady-state solution

Forced oscillations: example

$$x'' + 8x' + 16x = 8\sin(4t) \qquad x(0) = x'(0) = 0$$

Homogeneous

$$A = \begin{bmatrix} 0 & 1 \\ -16 & -8 \end{bmatrix} \quad \Longrightarrow \quad e^{At} = \begin{bmatrix} e^{-4t}(4t+1) & te^{-4t} \\ -16te^{-4t} & -e^{-4t}(4t-1) \end{bmatrix}$$

$$x_h(t) = c_1 e^{-4t} + c_2 t e^{-4t}$$

Forced oscillations: total solution

Typical forms of the particular integral [edit]

In order to find the particular integral, we need to 'guess' its form, with some coefficients left as variables to be solved for. This takes the form of the first derivative of the complementary function. Below is a table of some typical functions and the solution to guess for them.

Function of *x*	Form for *y*
ke^{ax}	Ce^{ax}
$kx^n,\ n = 0, 1, 2, \ldots$	$\sum_{i=0}^{n} K_i x^i$
$k\cos(ax)$ or $k\sin(ax)$	$K\cos(ax) + M\sin(ax)$
$ke^{ax}\cos(bx)$ or $ke^{ax}\sin(bx)$	$e^{ax}(K\cos(bx) + M\sin(bx))$
$\left(\sum_{i=0}^{n} k_i x^i\right)\cos(bx)$ or $\left(\sum_{i=0}^{n} k_i x^i\right)\sin(bx)$	$\left(\sum_{i=0}^{n} Q_i x^i\right)\cos(bx) + \left(\sum_{i=0}^{n} R_i x^i\right)\sin(bx)$
$\left(\sum_{i=0}^{n} k_i x^i\right)e^{ax}\cos(bx)$ or $\left(\sum_{i=0}^{n} k_i x^i\right)e^{ax}\sin(bx)$	$e^{ax}\left(\left(\sum_{i=0}^{n} Q_i x^i\right)\cos(bx) + \left(\sum_{i=0}^{n} R_i x^i\right)\sin(bx)\right)$

If a term in the above particular integral for *y* appears in the homogeneous solution, it is necessary to multiply by a sufficiently large power of *x* in order to make the solution independent. If the function of *x* is a sum of terms in the above table, the particular integral can be guessed using a sum of the corresponding terms for *y*.[1]

(From Wikipedia)

Particular

Method of undetermined coefficients suggests to try

$$x_p(t) = A\cos(4t) + B\sin(4t)$$

$$\implies \quad x_p(t) = -\frac{1}{4}\cos(4t)$$

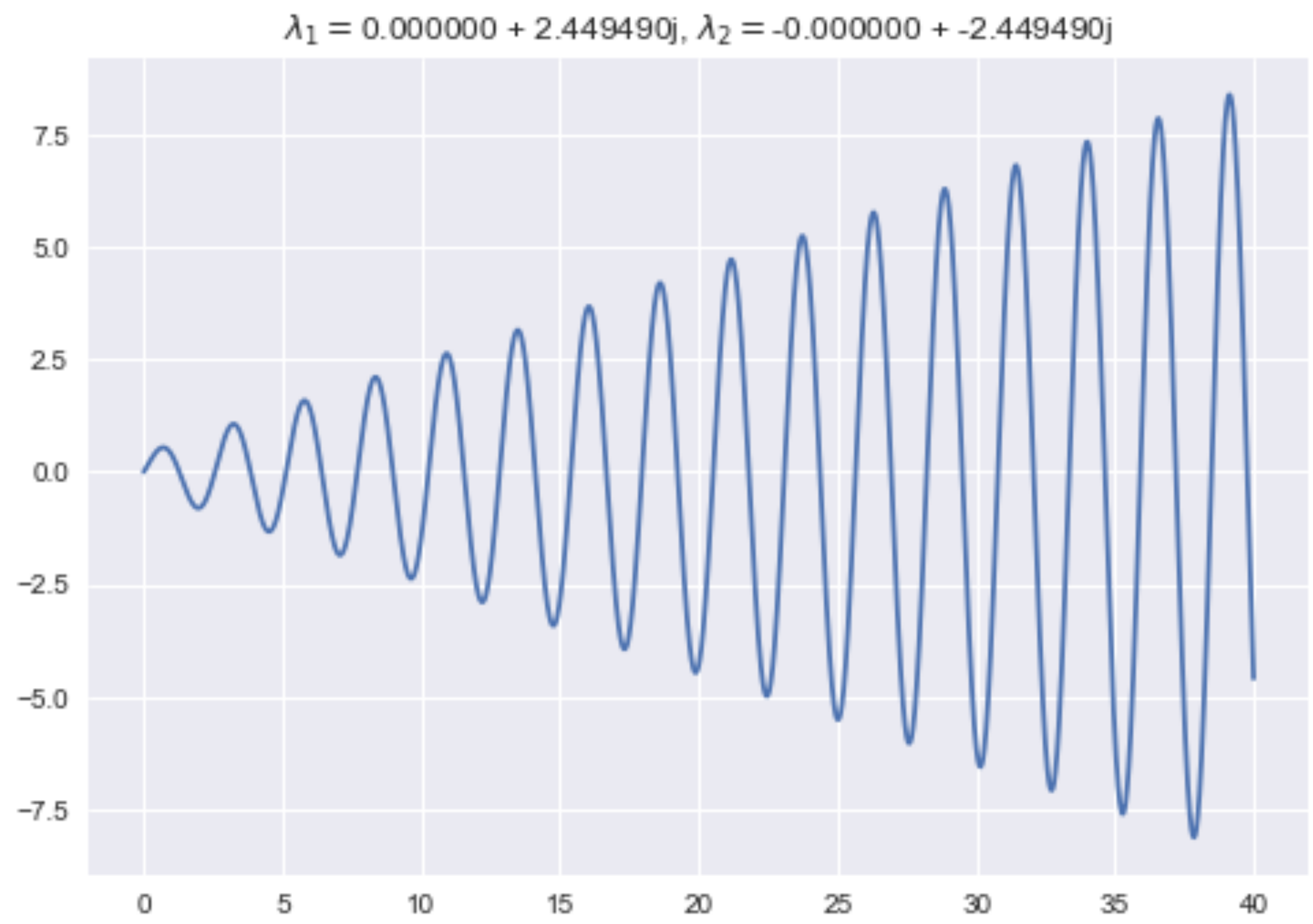
Total

$$\frac{1}{4}e^{-4t} + te^{-4t} - \frac{1}{4}\cos(4t)$$

Resonance

Systems that are not overdamped have their own **natural modes** or **resonant frequencies**

https://sites.lsa.umich.edu/ksmoore/research/tacoma-narrows-bridge/



Example

$x''(t) + x(t) = 5\cos(t)$

- Homogeneous solution is $x_h(t) = A\sin(t + \varphi)$
- Method of undetermined coefficients gives

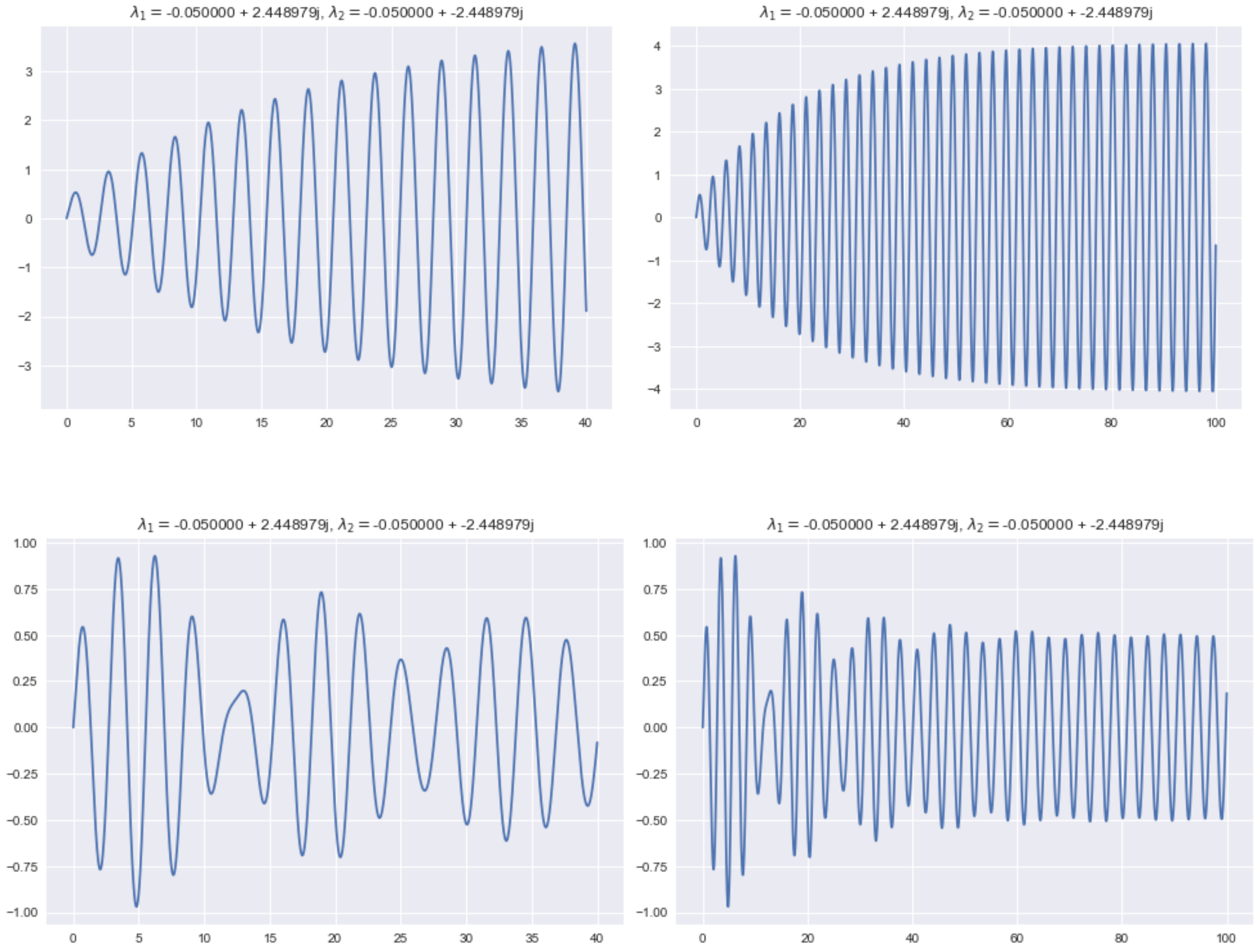
$$x_p(t) = \frac{1}{2} t \sin(1t)$$

- The total solution is then

$$\begin{aligned} x(t) &= x_h(t) + x_p(t) \\ &= A\sin(t+\varphi) + \frac{1}{2} t \sin(1t) \end{aligned}$$

It happens even in real systems with damping!

```
def f_osc(x, t, m=1, b=1, k=6, c=1, alpha=0, omega=1):
    A = np.array([[ 0,   1],
                  [-k/m, -b/m]])

    dydt = A.dot(x) + [0,
                       c * t**alpha * np.cos(omega*t)
                       ]
    return dydt

m = 1
b = 0.1
k = 6
u_0 = [0, 1]

A = np.array([[ 0,   1],
              [-k/m, -b/m]])
lam, V = np.linalg.eig(A)

T = 100
dt = 0.01
tspan = np.arange(0.0, T, dt)

f_osc_kb = lambda x, t : f_osc(x, t, b=b, k=k,
                                omega=np.abs(lam[0].imag))

u = odeint(f_osc_kb, u_0, tspan)
```



Application 1: Predicting the CO_2 concentration in the atmosphere

The DICE model

(Therina Groenewald/Shutterstock)

(https://www.unenvironment.org/)

The **Dynamic Integrated Climate-Economy model**

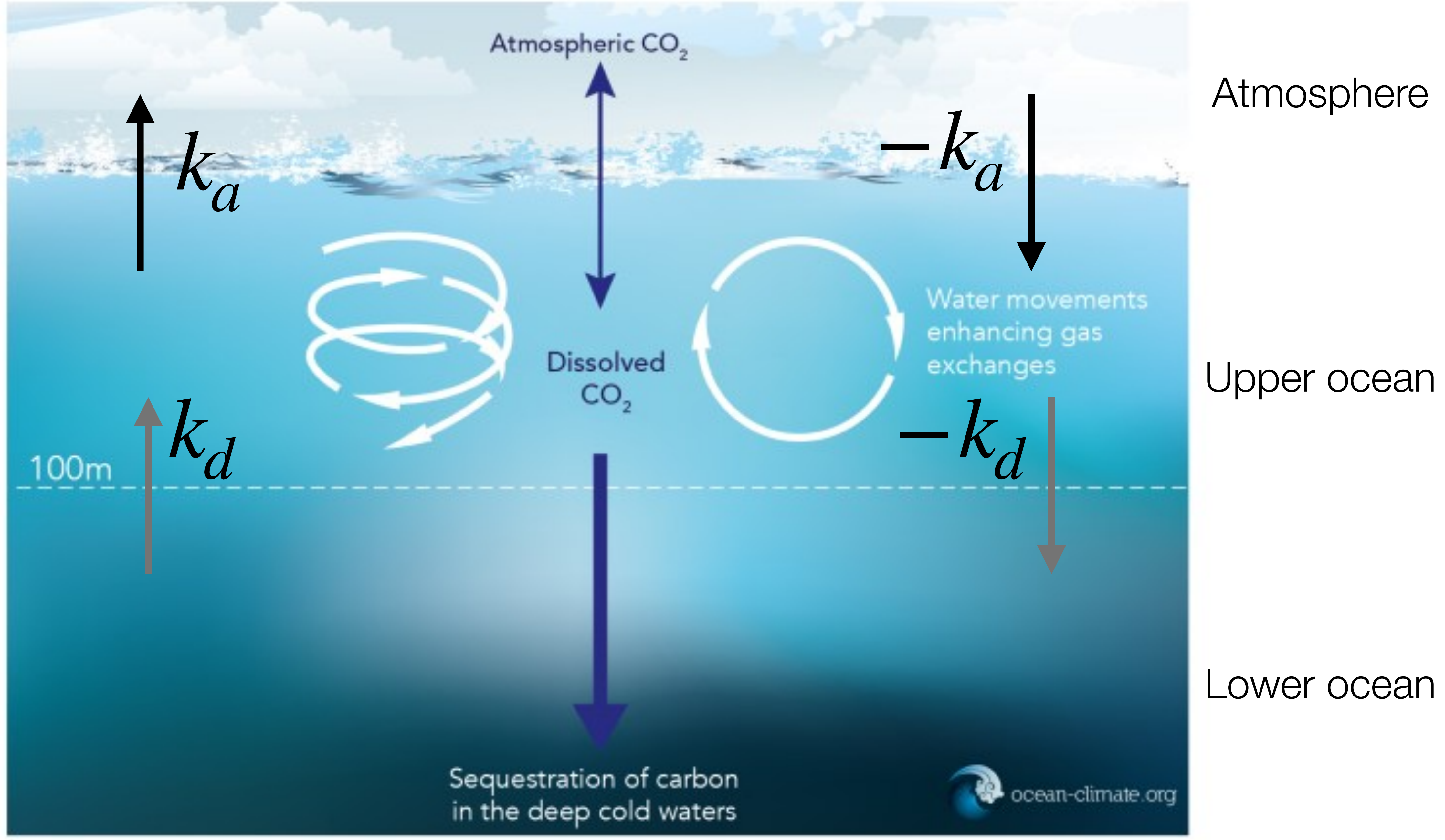
Economics | Carbon cycle | Climate science | Policy

William Nordhaus, 2018 Nobel Prize in Economics

Subject of **quite a bit of controversy** (likely a gross underestimate of the adverse effects)

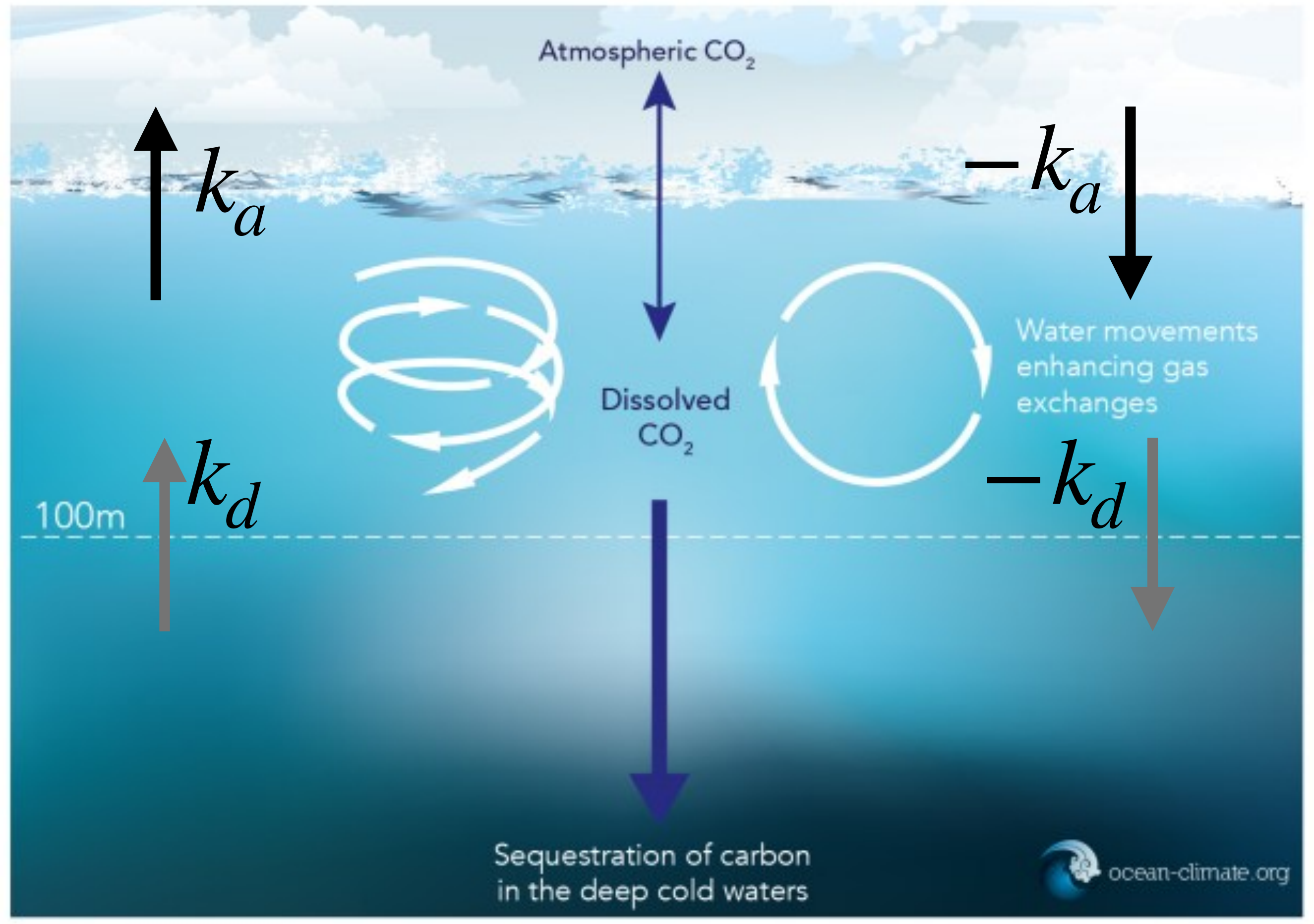
Coupling between the CO_2 containers

- An example of a **box model**: split the total CO_2 into *boxes* (**atmosphere**, **upper and lower ocean**)
- The boxes exchange CO_2 with certain rates (often determined via experimental fitting)



Physical carbon pump

Coupling between the CO_2 containers



Physical carbon pump

$$\frac{dM_{AT}}{dt} = E(t) - k_a \cdot (M_{AT} - A \cdot B \cdot M_{UP})$$

$$\frac{dM_{UP}}{dt} = k_a \cdot (M_{AT} - A \cdot B \cdot M_{UP}) - k_d \cdot (M_{UP} - \frac{M_{LO}}{\delta})$$

$$\frac{dM_{LO}}{dt} = k_d \cdot (M_{UP} - \frac{M_{LO}}{\delta})$$

- M_{AT}, M_{UP}, M_{LO} model CO_2 mass in atmosphere, upper, and lower ocean (in gigaton)
- $E(t)$ is the emission rate (gigaton / year)
- AB is the equilibrium ratio of CO_2 between the atmosphere and the upper ocean
- δ is the volume ratio between upper and lower ocean
- k_a, k_d are CO_2 exchange rates between atmosphere/upper ocean and upper/lower ocean

A linear algebra problem?

$$\frac{d\boldsymbol{m}}{dt} = K\boldsymbol{m} + \boldsymbol{e}(t)$$

$$\boldsymbol{m} = \begin{bmatrix} M_{AT} \\ M_{UP} \\ M_{LO} \end{bmatrix} \qquad K = \begin{bmatrix} -k_a & k_a AB & 0 \\ k_a & -k_a AB - k_d & k_d/\delta \\ 0 & k_d & -k_d/\delta \end{bmatrix} \qquad \boldsymbol{e}(t) = \begin{bmatrix} E(t) \\ 0 \\ 0 \end{bmatrix}$$

- Now we have an **inhomogeneous *system*** of ODEs
- Is there a “principled” way to integrate (solve) such systems?

Solving the inhomogeneous equation

We now know that when K is constant in time, the solution to $\dfrac{d\boldsymbol{m}}{dt} = K\boldsymbol{m}$ is

$$\boldsymbol{m}(t) = e^{Kt}\boldsymbol{m}(0)$$

Duhamel's principle

Massage the inhomogeneous equation into a homogeneous form

$$\frac{d\boldsymbol{m}}{dt} = K\boldsymbol{m} + \boldsymbol{e}(t) \quad \Longleftrightarrow \quad e^{tK}\frac{d}{dt}\left(e^{-tK}\boldsymbol{m}(t)\right) = \boldsymbol{e}(t)$$

It follows that

$$\boldsymbol{m}(t) = e^{tK}\boldsymbol{m}(0) + \int_0^t e^{(t-s)K}\boldsymbol{e}(s)ds$$

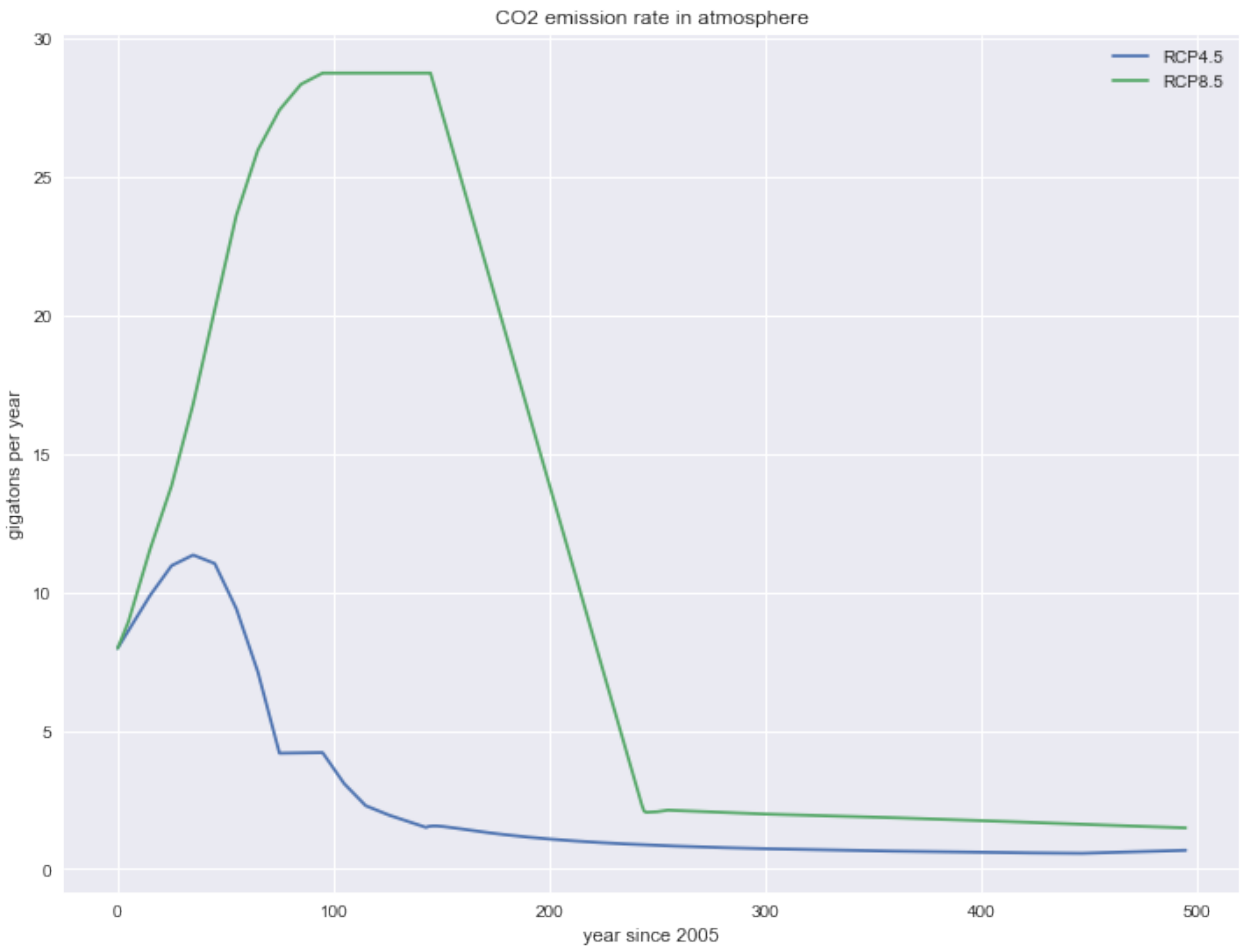
CO_2 emission scenarios

- **Representative concentration pathways (RCPs)**: Emission scenarios from pre-industrial period to year 2050

- Consolidated by the **Intergovernmental Panel on Climate Change (IPCC)**

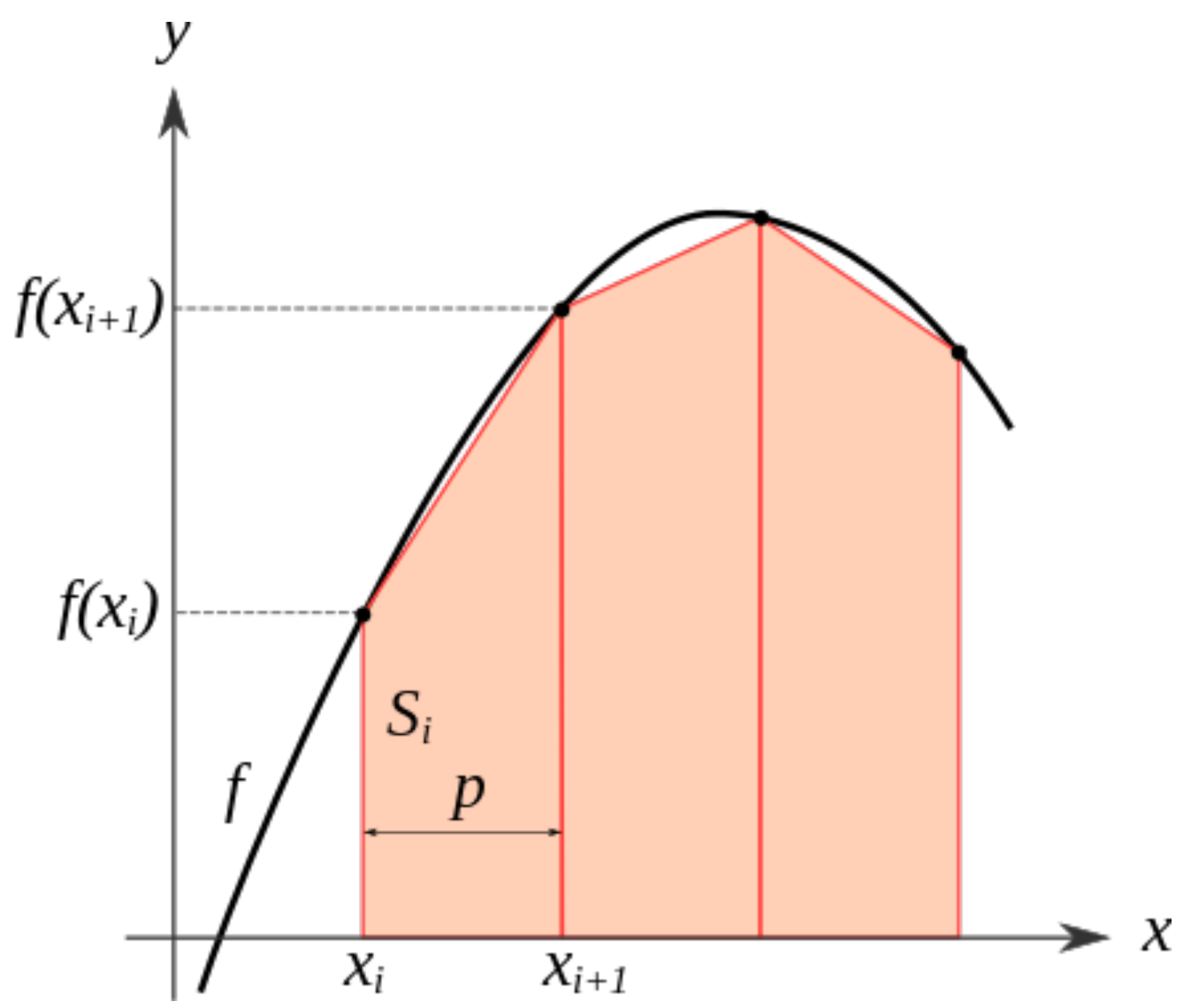
- **RCP**4.5 = intermediate emission; emissions peak in 2040 and then decline

- **RCP**8.5 = *business-as-usual* emissions; worst case



Approximating the integral by the trapezoidal rule

$$\int_0^t e^{(t-s)K} ds \simeq \sum_{j=0}^{N} \frac{\Delta t}{2} \left(e^{(t-j\Delta t)K} + e^{(t-(j+1)\Delta t)K} \right)$$



(Wikipedia)

```
# Solution for the emission rate using RCP8.5

M85 = np.zeros([nt, M0.shape[0]])
M85[0, :] = M0

# precompute matrix exponentials
expKt = np.zeros((nt, 3, 3))
for i in range(nt):
    expKt[i] = la.expm(K * dt * i)

for i in range(nt - 1):
    addsrc = np.zeros([1, 3])

    # integrate using trapezoidal rule
    for j in range(i - 1):
        addsrc += 0.5 * dt * (np.matmul(expKt[i + 1 - j],
                                        emis85[j,:])
                              +
                              np.matmul(expKt[i + 1 - (j + 1)],
                                        emis85[j + 1, :]))

    M85[i + 1, :] = np.matmul(expKt[i + 1], M0) + addsrc
```

CO2 concentration and the surface temperature

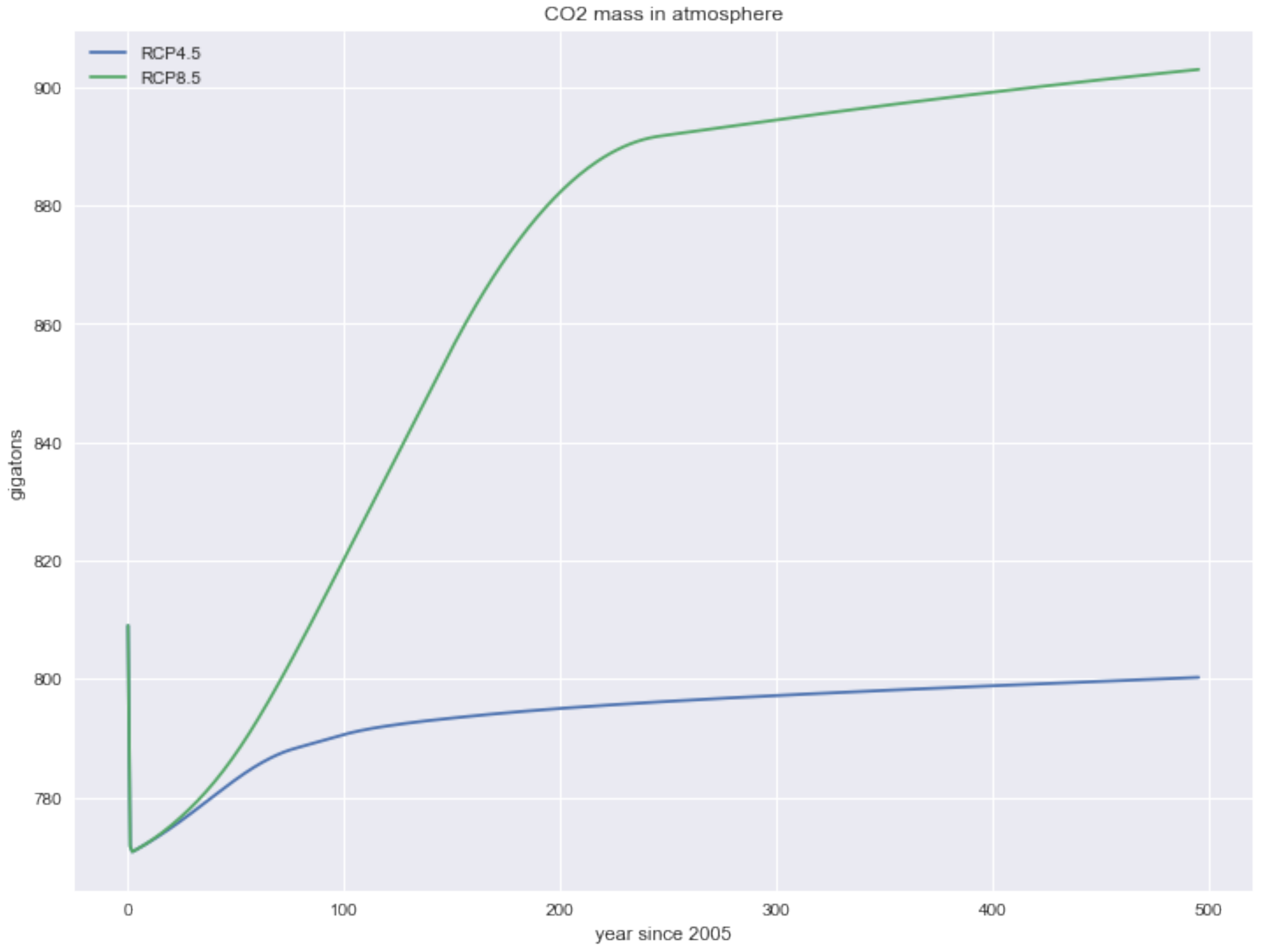
- The temperature change with respect to a pre-industrial reference is estimated as

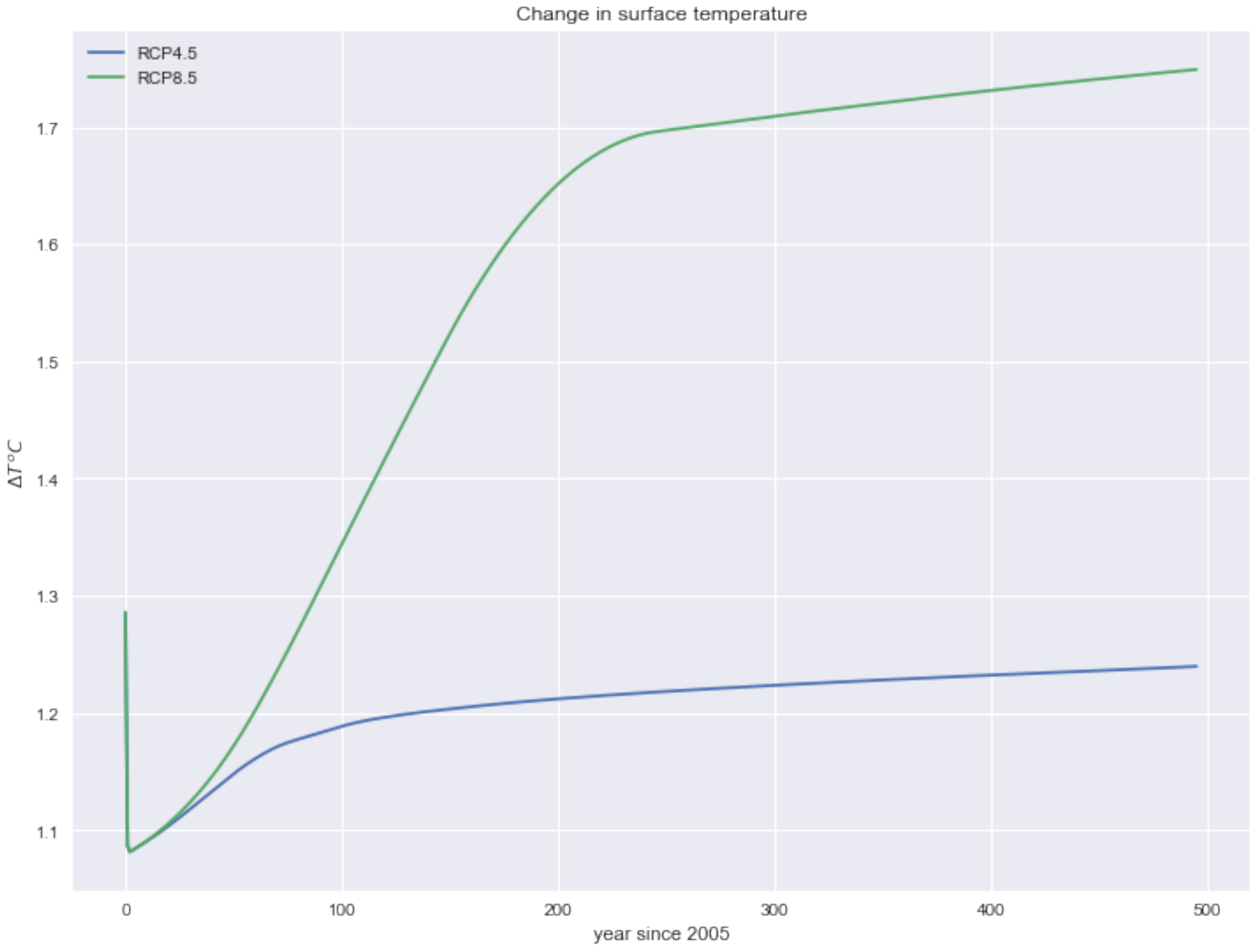
$$\Delta T = \frac{\alpha}{\lambda}\log_2\left(\frac{M_{AT}}{M_{AT,ref}}\right)$$

$\alpha = 3.8$ W/m^2

$\lambda = 1.3$ W/m^2/°C

$M_{AT,ref} = 596.4$ GtC





Limitations of the model

- A major limitation of the model is that K is a constant matrix independent of time and the current concentrations
- In reality the carbonate chemistry dictates that the absorption capacity of the ocean drops after initial absorption, resulting in huge errors over longer timescales
- One remedy is to allow the coefficients $k_a, k_d, AB, \ldots$ to depend on time and the current concentrations M_{AT}, M_{UP}, M_{LO}

- NB: Nordhaus's work and models have been even more heavily criticized for how they measure economic utility, in that they "overemphasize growth as the ultimate measure of economic success" (https://www.sciencemag.org/news/2018/10/roles-ideas-and-climate-growth-earn-duo-economics-nobel-prize)

Limitations of the model

- A major limitat … ent of time and
  the current co …
- In reality the c … of the ocean
  drops after init … cales
- One remedy is … e and the
  current conce …

Nobel Prize for the economics of innovation and climate change stirs controversy

By Adrian Cho | Oct. 8, 2018 , 9:40 PM

Often, the awarding of a Nobel Prize triggers a round of carping about who else should have shared in the prize. This year's prize for economics—officially, the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel—has sparked a rarer controversy. Some economists argue one winner's work is wrongheaded and has compromised humanity's ability to deal with the existential threat of climate change.

- NB: Nordhaus's work and models have been even more heavily criticized for how they measure economic utility, in that they "overemphasize growth as the ultimate measure of economic success" (https://www.sciencemag.org/news/2018/10/roles-ideas-and-climate-growth-earn-duo-economics-nobel-prize)

Application 2: Modeling the COVID-19 pandemic

The simplest useful model: SIR

The SIR model [edit]

In 1927, W. O. Kermack and A. G. McKendrick created a model in which they considered a fixed population with only three compartments: susceptible, $S(t)$; infected, $I(t)$; and recovered, $R(t)$. The compartments used for this model consist of three classes:[13]

- $S(t)$ is used to represent the individuals not yet infected with the disease at time t, or those susceptible to the disease of the population.
- $I(t)$ denotes the individuals of the population who have been infected with the disease and are capable of spreading the disease to those in the susceptible category.
- $R(t)$ is the compartment used for the individuals of the population who have been infected and then removed from the disease, either due to immunization or due to death. Those in this category are not able to be infected again or to transmit the infection to others.

(From Wikipedia)

The MSEIR (…) family of models

Idea: divide population into groups according to their status relative to the disease

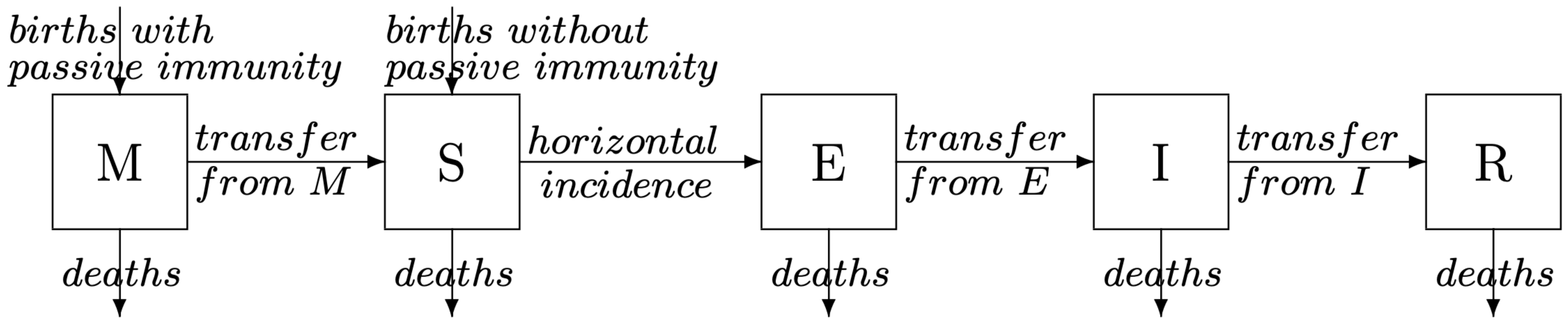


Fig. 1 *The general transfer diagram for the MSEIR model with the passively immune class M, the susceptible class S, the exposed class E, the infective class I, and the recovered class R.*

The simplest useful model: SIR

$$\frac{dS}{dt} = -\beta\frac{I}{N}S \qquad \frac{dI}{dt} = \beta\frac{I}{N}S - \gamma I \qquad \frac{dR}{dt} = \gamma I$$

Infection rate $\lambda = \beta\frac{I}{N}$ **Recovery rate** γ

Is this a linear algebra problem?

- Letting $\boldsymbol{u}(t) = \begin{bmatrix} S(t) \\ I(t) \\ R(t) \end{bmatrix}$, can we write $\dfrac{d\boldsymbol{u}(t)}{dt} = A\boldsymbol{u}(t)$ for some matrix A that does not depend on S, I, R?

- Sadly, no… the expressions contain multiplications between S and I

- A superposition of two solutions is in general **not** a solution

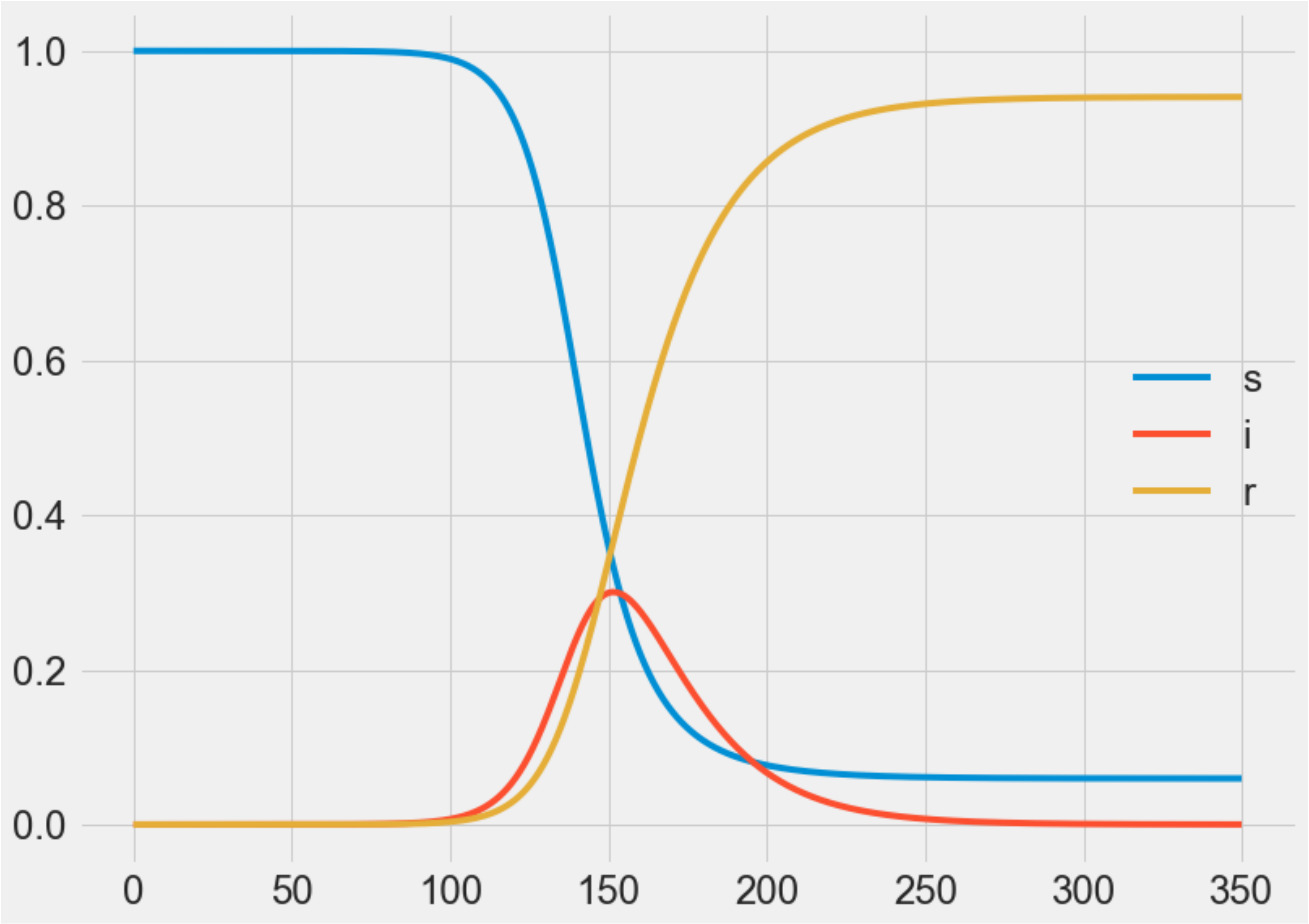
- Perhaps not everything is lost…

$$\frac{dS}{dt} = -\beta \frac{I(t)}{N} S(t)$$

$$\frac{dI}{dt} = \beta \frac{I(t)}{N} S(t) - \gamma I(t)$$

SIR curves

```
def f_sir(x, t, gamma=1/18, R0=3):
    s, i, r = x

    dydt = [-gamma*R0*s*i,
             gamma*R0*s*i - gamma*i,
             gamma*i
            ]
    return dydt

# parameters
T = 350
dt = 0.1

i_0 = 1e-7   # 33 = 1E-7 * 330 million
s_0 = 1.0 - i_0
r_0 = 0.0
y_0 = [s_0, i_0, r_0]  # initial condition

tspan = np.arange(0.0, T, dt)

y = odeint(f_sir, y_0, tspan)


ax = plt.plot(tspan, y)
plt.legend(['s', 'i', 'r'], fontsize=24)
```



(

Stability of dynamical systems / ODEs

Key principle When things are non-linear, linearize them!

Taylor series (first two terms)

$$f(t) = f(t_0) + f'(t_0)(t - t_0) + O(|t - t_0|^2)$$

Key question Linearize about which point? How to choose t_0?

Equilibria of dynamical systems

Good choice: **equilibria** of

$$\frac{d\boldsymbol{u}(t)}{dt} = F(t, \boldsymbol{u}(t))$$

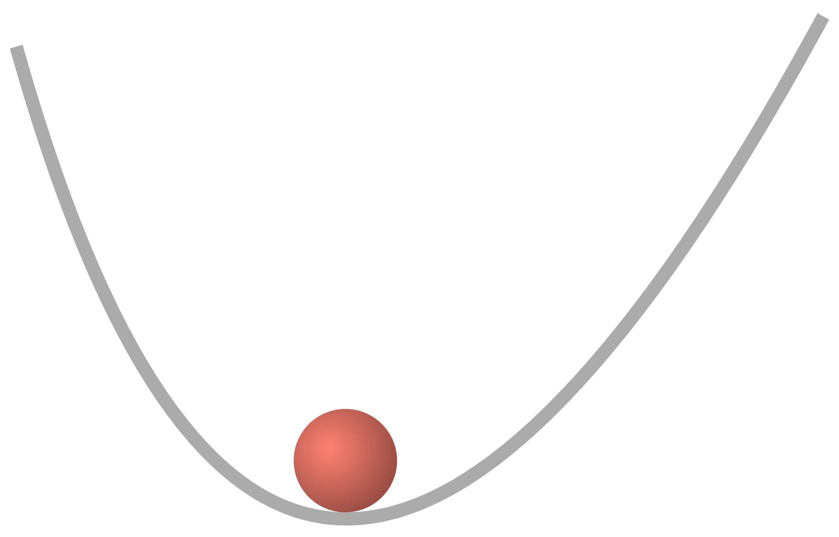
In an equilibrium, $\boldsymbol{u}$ does not change:

$$\frac{d\boldsymbol{u}}{dt} = \mathbf{0} \iff F(t, \boldsymbol{u}(t)) = 0$$

What happens when we tap a system in equilibrium?

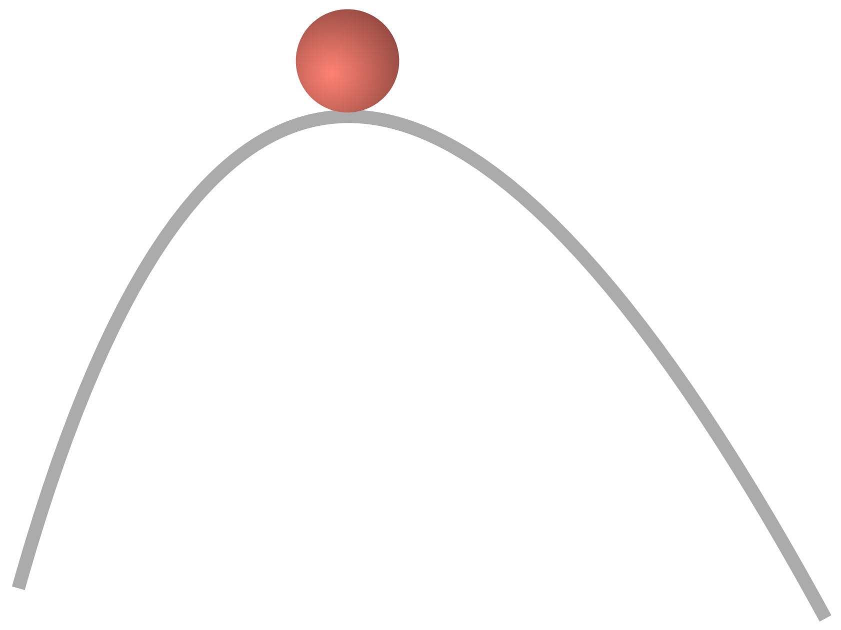
Stable and unstable equilibria

Stable



$$\frac{d\boldsymbol{u}}{dt} = \mathbf{0}$$

Unstable



$$\frac{d\boldsymbol{u}}{dt} = \mathbf{0}$$

A stability criterion

- Let us look at a simple first-order ODE $\frac{du}{dt} = \alpha u$
- Equilibria are at $\frac{du}{dt} = 0$ which is solved by $u = u_0 = 0$

When we **perturb** the system around an **equilibrium**, do we **come back** to the equilibrium or we **go away** from it?

- We know that a general solution is given as $u(t) = ce^{\alpha t}$
- Thus starting from a point $u(0) = u_0 + \epsilon = \epsilon$, do we go back to 0 or not? We already know this!

$\alpha < 0$ **stable**

$\alpha > 0$ **unstable**

A stability criterion

- The key parameter in a linear first-order ODE is α;
  we would like to generalize to $\frac{du}{dt} = f(u)$ with a nonlinear $f(u)$.
- For the linear $f(u) = \alpha u$, the key parameter α equals $f'(u)$.
  Coincidence?
- What happens when we move very slightly out of an equilibrium?

$$\left.\frac{du}{dt}\right|_{u=u_0+\epsilon} = f(u_0 + \epsilon) \approx f(u_0) + f'(u_0)\epsilon = f'(u_0)\epsilon$$

A stability criterion

$$\left.\frac{du}{dt}\right|_{u=u_0+\epsilon} = f(u_0+\epsilon) \approx f(u_0) + f'(u_0)\epsilon = f'(u_0)\epsilon$$

A constant scalar

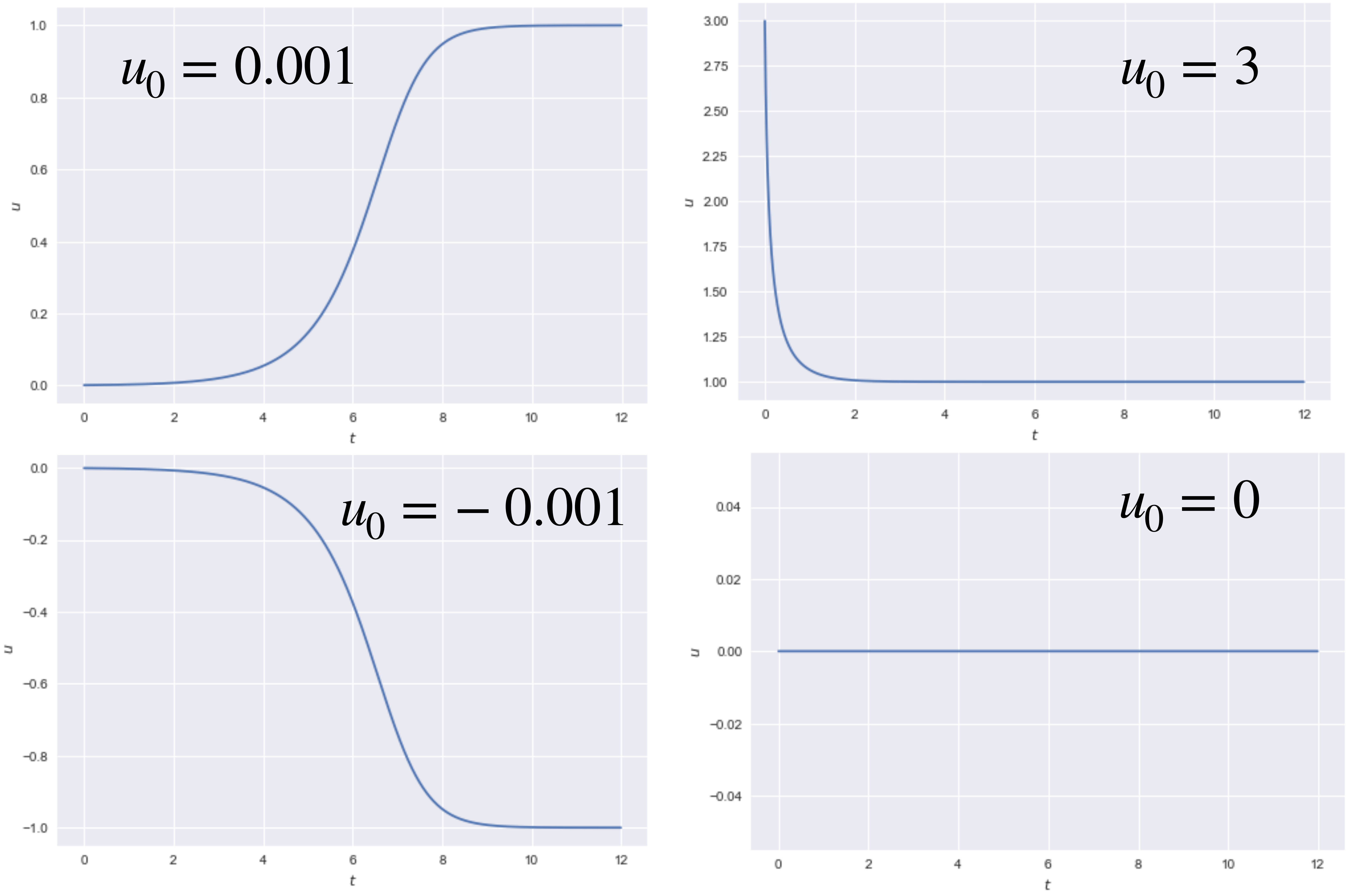
- For small ϵ (close to U) the above approximation is accurate: $f'(U)$ plays the role of α!
- Since $\left.\frac{du}{dt}\right|_{u=U+\epsilon} = \frac{d\epsilon}{dt}$, we effectively **linearized** our nonlinear equation around U

Example 1: Simple 1D systems

	$\frac{du}{dt} = \alpha u$	$\frac{du}{dt} = u - u^2$	$\frac{du}{dt} = u - u^3$
$\frac{du}{dt} = 0$	$u_0 = 0$	$u_0 = 0,\ 1$	$u_0 = 0, \pm 1$
$\left.\frac{df}{du}\right\vert_{u=U}$	α	$f'(0) = 1$ $f'(1) = -1$	$f'(0) = 1$ $f'(\pm 1) = -2$
Stable?	$\alpha < 0$ ✓ $\alpha > 0$ ✗	$U = 0$ ✗ $U = 1$ ✓	$U = 0$ ✗ $U = 1$ ✓ $U = -1$ ✓

A quick numerical check…

```python
# Simple 1D examples

def f_sys1(x, t, alpha=-1):
    return alpha*x

def f_sys2(x, t):
    return x - x**2

def f_sys3(x, t):
    return x - x**3


T = 12
dt = 0.01
tspan = np.arange(0.0, T, dt)

u_0 = 0.001
u = odeint(f_sys3, u_0, tspan)

plot(tspan, u)
plt.xlabel('$t$')
plt.ylabel('$u$')
```



Example 2: FitzHugh—Nagumo model of a spiking neuron

v = membrane potential $\qquad$ w = recovery variable

$$\frac{dv}{dt} = I_{app} + v - \frac{v^3}{3} - w \qquad \frac{dw}{dt} = \epsilon(v - \alpha w + \beta)$$

$$\boldsymbol{u} = \begin{bmatrix} v \\ w \end{bmatrix} \qquad \frac{d\boldsymbol{u}}{dt} = \boldsymbol{F}(\boldsymbol{u})$$

Example 2: FitzHugh—Nagumo model of a spiking neuron

$$I_{app} = 0.01 \ \text{A}$$

$$\epsilon = 0.01$$

$$\alpha = 5.00$$

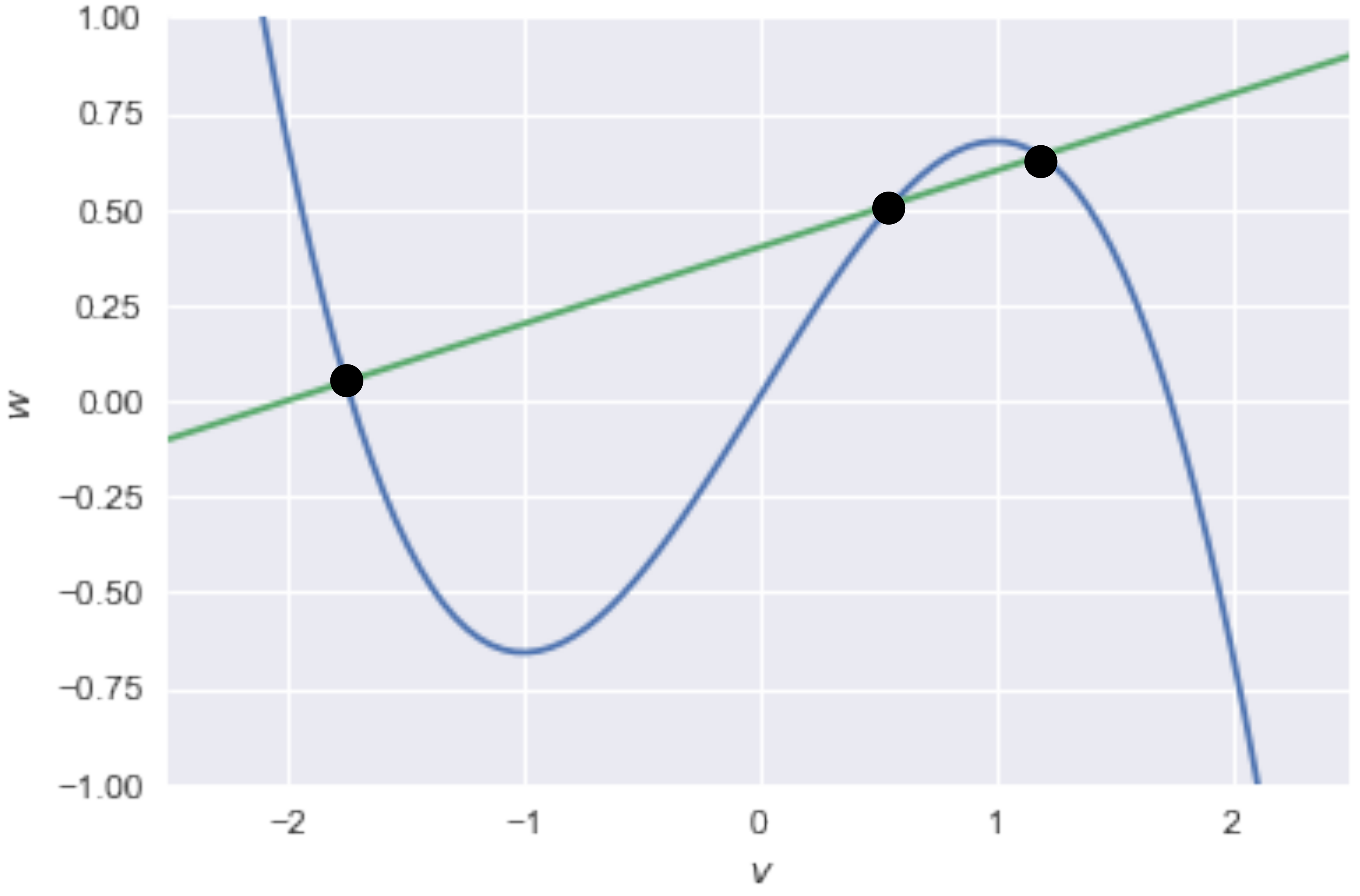
$$\beta = 2.00$$

(Not necessarily a realistic choice of parameters)

Equilibria

$$I_{app} + v - \frac{v^3}{3} - w = 0$$

$$\epsilon(v - \alpha w + \beta) = 0$$




```
v0 = np.roots([-1.0/3.0, 0, 1.0 - 1.0 / alpha, I_app - beta / alpha])
v0 = np.sort(v0.real)
w0 = 1.0 / alpha * (v0 + beta)

print(v0)
```

v-roots [-1.75156349 0.56110887 1.19045461]

Stable or unstable equilibria?

Multivariate Taylor (linear term)

$$\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{F}(\boldsymbol{u_0}) + \nabla_{\boldsymbol{u}}\boldsymbol{F}(\boldsymbol{u_0})(\boldsymbol{u} - \boldsymbol{u}_0) + O(\|\boldsymbol{u} - \boldsymbol{u}_u\|^2)$$

$\nabla_{\boldsymbol{u}}\boldsymbol{F}$ is the **Jacobian**

$$\nabla_{\boldsymbol{u}}\boldsymbol{F} = \begin{bmatrix} \frac{dF_1}{dv} & \frac{dF_1}{dw} \\ \frac{dF_2}{dv} & \frac{dF_2}{dw} \end{bmatrix} = \begin{bmatrix} 1 - v^2 & -1 \\ \epsilon & -\epsilon\alpha \end{bmatrix}$$

```
for idx in [0, 1, 2]:
    D = [[1 - v0[idx]**2, -1],
         [epsilon, -alpha*epsilon]
        ]
    evals, _ = np.linalg.eig(D)
    print(evals)
```

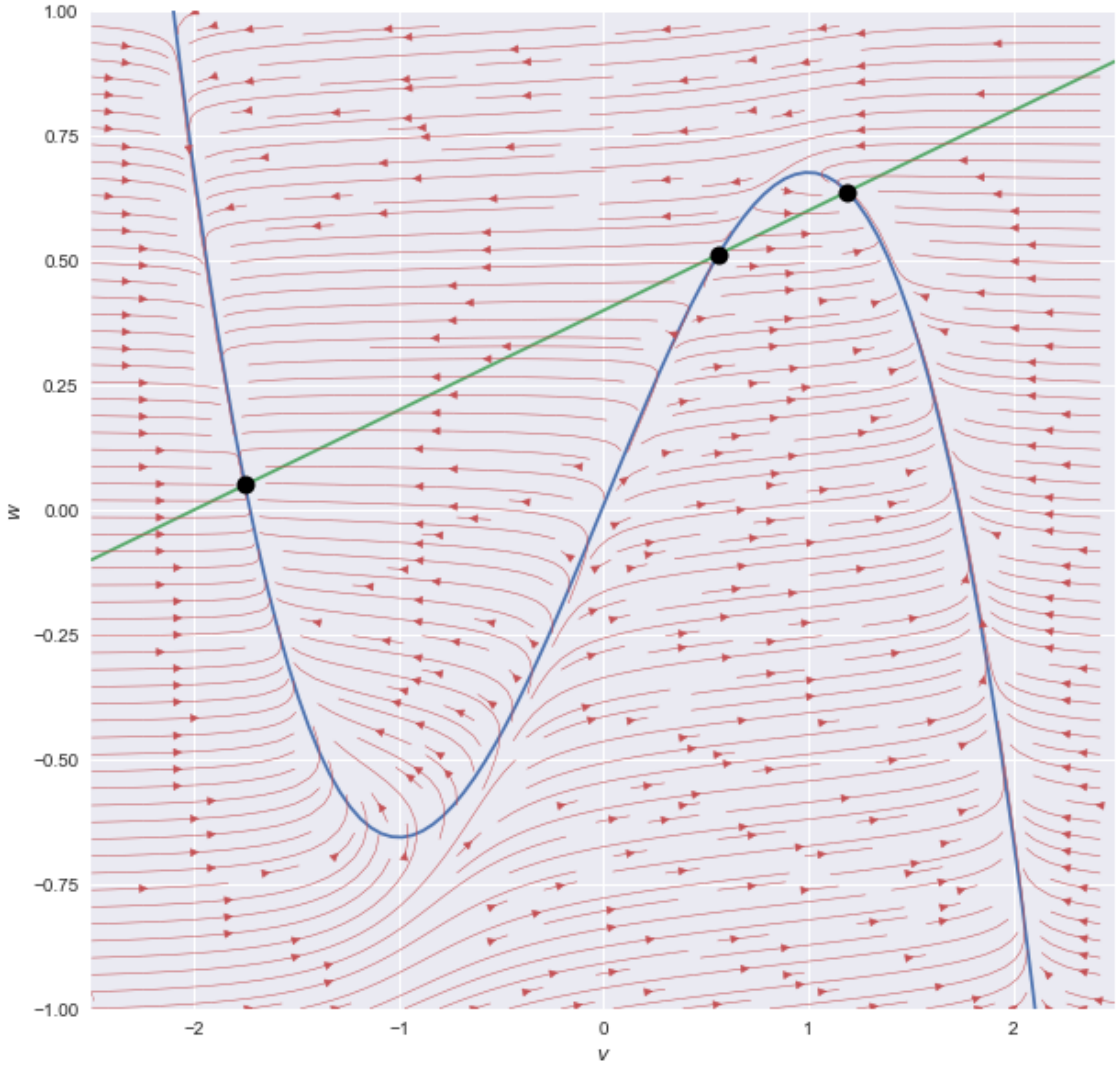
```
[-2.06300695 -0.05496769]
[ 0.67129284 -0.03613601]
[-0.38755761 -0.07962457]
```

Phase portrait: FitzHugh—Nagumo

- Start the evolution of the system at many points and track where it goes

```
for idx in [0, 1, 2]:
    D = [[1 - v0[idx]**2, -1],
         [epsilon, -alpha*epsilon]
        ]
    evals, _ = np.linalg.eig(D)
    print(evals)
```

```
[-2.06300695 -0.05496769]
[ 0.67129284 -0.03613601]
[-0.38755761 -0.07962457]
```



)

OK, back to COVID…

Application to the SIR model

$$\frac{dS}{dt} = -\beta \frac{I(t)}{N} S(t)$$

$$\frac{dI}{dt} = \beta \frac{I(t)}{N} S(t) - \gamma I(t)$$

$$\frac{d}{dt} \begin{bmatrix} S \\ I \end{bmatrix} = \boldsymbol{F}(S, I) = \begin{bmatrix} F_1(S, I) \\ F_2(S, I) \end{bmatrix}$$

Since $R = N - S - I$, if $S(t)$ and $I(t)$ don't change, neither does R

$$\frac{dS}{dt} = 0$$

$$\frac{dI}{dt} = 0$$

$$\Longrightarrow$$

Epidemic equilibria

$$(S, I) = (N, 0)$$

$$(S, I) = (0, 0)$$

Linearize around the $(N, 0)$ equilibrium

Taylor series (first two terms)

$$\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{F}(\boldsymbol{u_0}) + \nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{u_0})(\boldsymbol{u} - \boldsymbol{u}_0) + O(\|\boldsymbol{u} - \boldsymbol{u}_u\|^2)$$

$$\frac{d}{dt}\begin{bmatrix} S \\ I \end{bmatrix} = \begin{bmatrix} -\beta \frac{I}{N} S \\ \beta \frac{I}{N} S - \gamma I \end{bmatrix}$$

$$\approx \begin{bmatrix} -\beta \frac{I}{N} S \\ \beta \frac{I}{N} S - \gamma I \end{bmatrix} \Bigg|_{S=N, I=0} + \begin{pmatrix} \frac{d}{dS}\left(-\beta \frac{I}{N} S\right) & \frac{d}{dI}\left(-\beta \frac{I}{N} S\right) \\ \frac{d}{dS}\left(\beta \frac{I}{N} S - \gamma I\right) & \frac{d}{dI}\left(\beta \frac{I}{N} S - \gamma I\right) \end{pmatrix} \Bigg|_{S=N, I=0} \left(\begin{bmatrix} S \\ I \end{bmatrix} - \begin{bmatrix} N \\ 0 \end{bmatrix} \right)$$

Finally…

$$\frac{d}{dt}\begin{bmatrix} S \\ I \end{bmatrix} \approx \begin{bmatrix} 0 & -\beta \\ 0 & \beta - \gamma \end{bmatrix} \left(\begin{bmatrix} S \\ I \end{bmatrix} - \begin{bmatrix} N \\ 0 \end{bmatrix} \right)$$

Eigenvalues of the Jacobian matrix

$$\lambda_1 = 0 \qquad \lambda_2 = \beta - \gamma$$

$\beta > \gamma \Rightarrow$ epidemic

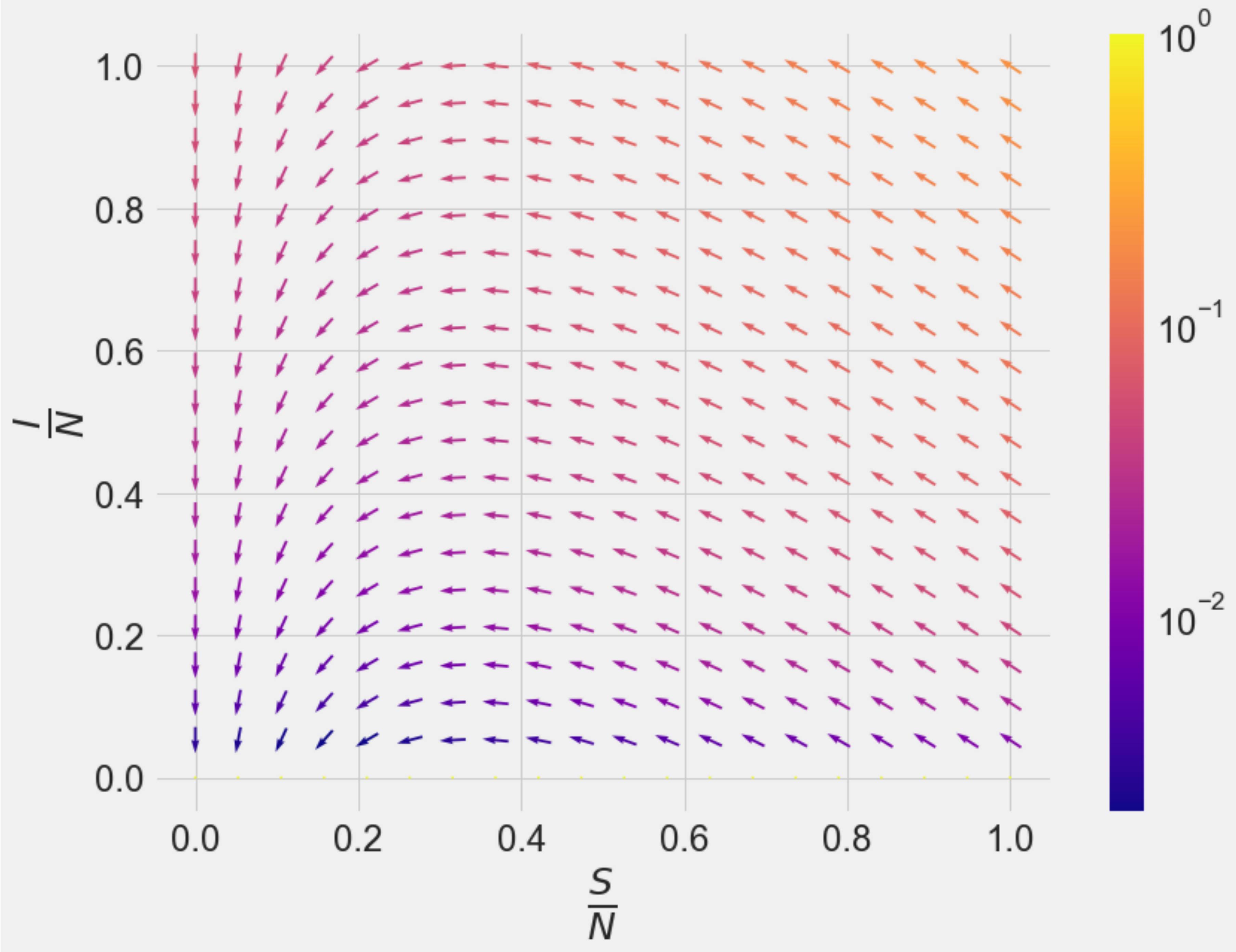
$\beta < \gamma \Rightarrow$ no epidemic

key parameter

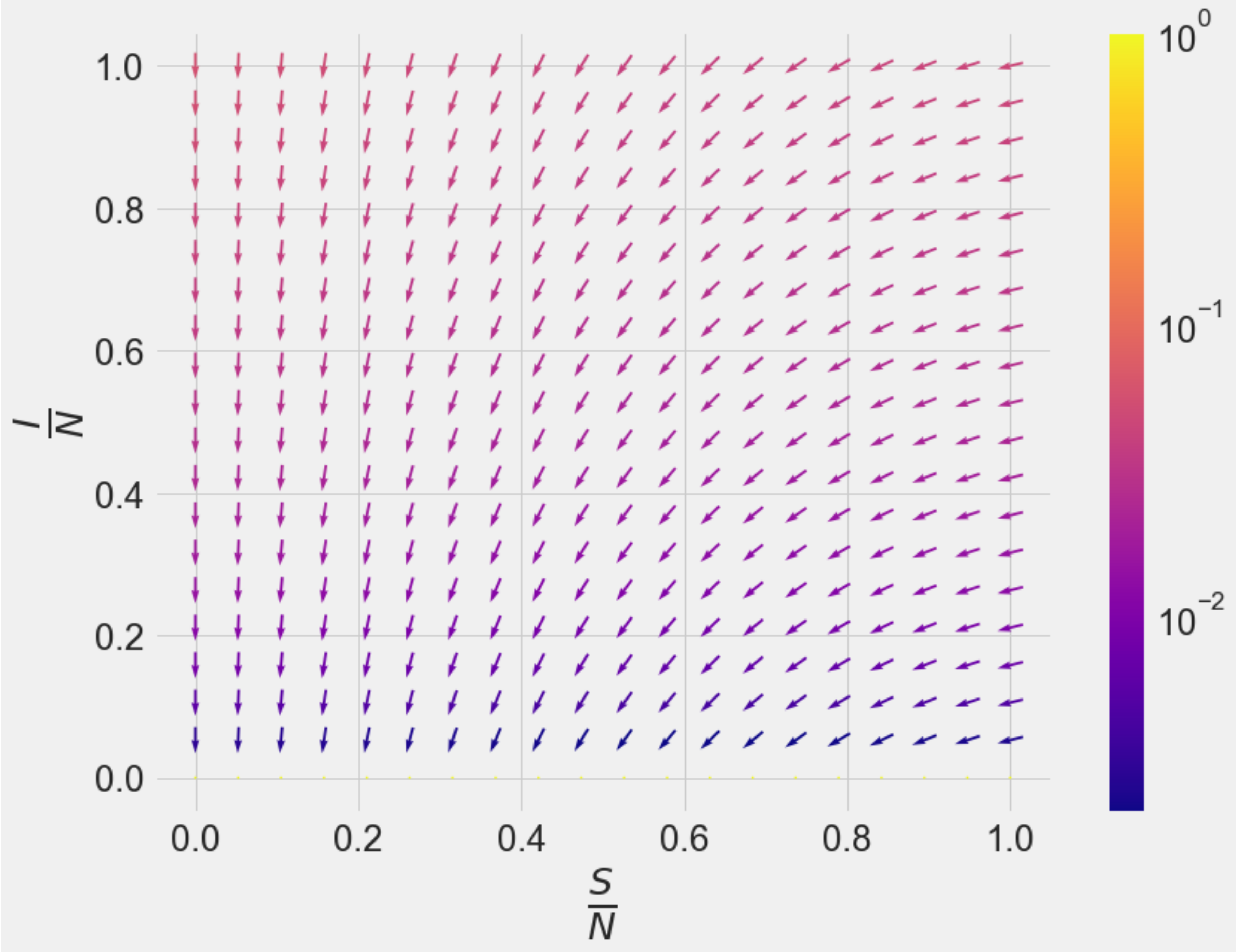
$$R_0 := \frac{\beta}{\gamma}$$

Phase portraits

$R_0 = 3$

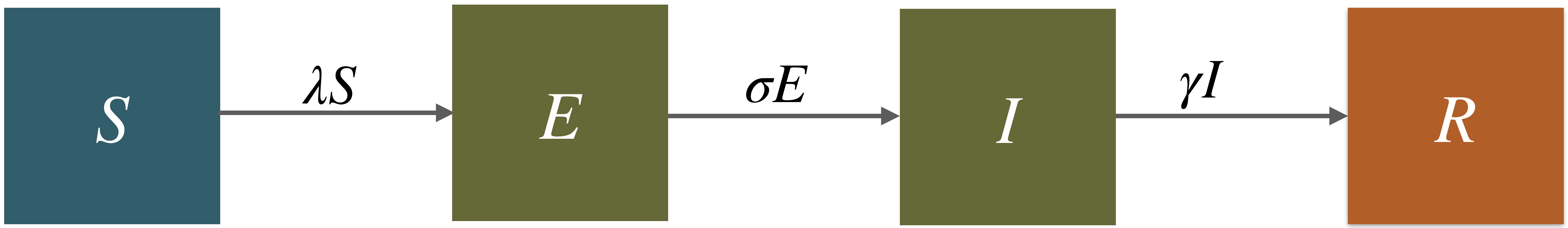


$R_0 = 0.8$



Extending the model

- We can first improve the model by adding a 4th compartment, **E**
- This models *exposed* individuals who will become infected after an **incubation period**



$$\frac{dS}{dt} = -\beta \frac{I}{N} S \qquad \frac{dE}{dt} = \beta S \frac{I}{N} S - \sigma E \qquad \frac{dI}{dt} = \beta \frac{I}{N} S - \gamma I \qquad \frac{dR}{dt} = \gamma I$$

Extending the model

- Next, we normalize everything by the total population, $s = S/N, i = I/N, e = E/N, r = R/N$
- Reparameterize the equations in terms of R_0; here it is defined as $R_0 = \dfrac{\beta}{\gamma}$

$$
\begin{aligned}
\dot{s} &= -\gamma R_0 \, s \, i \\
\dot{e} &= \gamma R_0 \, s \, i - \sigma e \\
\dot{i} &= \sigma e - \gamma i \\
\dot{r} &= \gamma i
\end{aligned}
$$

$$s + e + i + r = 1$$

Mitigation

- The idea is that R_0 can be influenced by policy—a lockdown hopefully makes it smaller
- R_0 does not change instantaneously

$$\frac{dR_0}{dt} = \eta(R_{\text{target}} - R_0)$$

- It will be interesting to track the **cumulative caseload** $c = i + r$ and the **number of deaths**

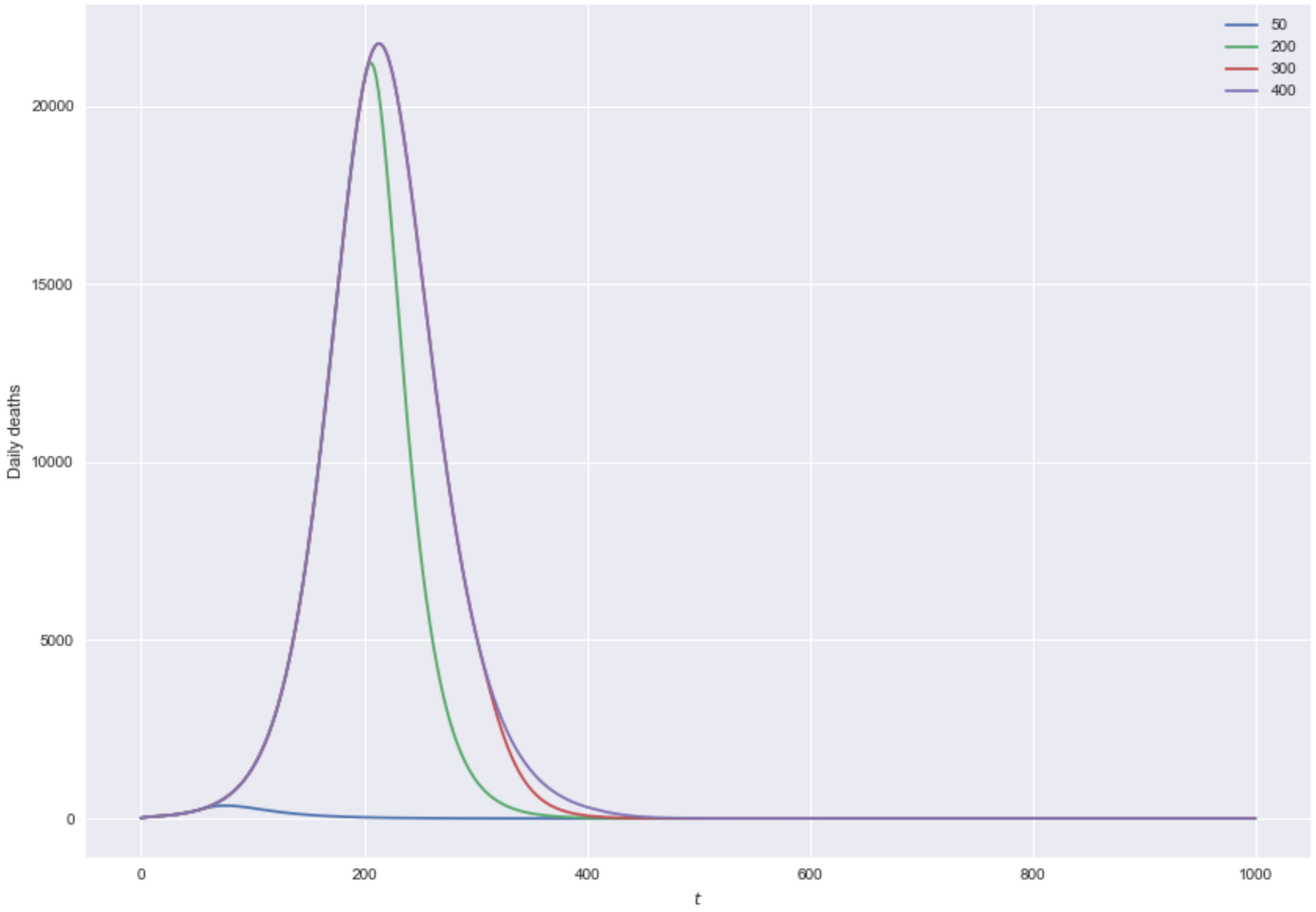
$$\frac{dc}{dr} = \sigma e \qquad \frac{dd}{dt} = \delta\gamma i$$

Modeling in python

```
def f_seir_ld(x, t, gamma=1.0/18, sigma=1/5.2, R0_1=2.0,
              R0_2=0.5, t_change=100, eta=1.0/20, delta=0.01):
    s, e, i, r, R0, c, d = x

    R0_inf = R0_1 if t < t_change else R0_2

    dydt = [-gamma*R0*s*i,                 # ds/dt = -γR₀si
            gamma*R0*s*i - sigma*e,         # de/dt =  γR₀si -σe
            sigma*e   - gamma*i,            # di/dt =         σe -γi
                        gamma*i,            # dr/dt =             γi
            eta*(R0_inf - R0),
            sigma*e,
            delta*gamma*i
           ]
    return dydt
```

Introducing lockdown

```
# lifting or introducing lockdown

lift = False

R0_L = 0.5
R0_NL = 2.0
t_change_list = [50, 200, 300, 400]

R0_1, R0_2 = (R0_L, R0_NL) if lift else (R0_NL, R0_L)

T = 1000
dt = 1
tspan = np.arange(0.0, T, dt)

plt.figure(figsize=(14, 10))
for t_change in t_change_list:
    f_seir_ld_t = lambda x, t : f_seir_ld(x, t, t_change=t_change,
                                          R0_1=R0_1, R0_2=R0_2)

    y_0 = [s_0, e_0, i_0, r_0, R0_1, 0, 0]

    y = odeint(f_seir_ld_t, y_0, tspan)
    deaths = N * delta * gamma * y[:, 2]
    _ = plt.plot(tspan, deaths)
```



Lifting lockdown

```python
# lifting or introducing lockdown

lift = True

R0_L = 0.5
R0_NL = 2.0
t_change_list = [50, 200, 300, 400]

R0_1, R0_2 = (R0_L, R0_NL) if lift else (R0_NL, R0_L)

T = 1000
dt = 1
tspan = np.arange(0.0, T, dt)

plt.figure(figsize=(14, 10))
for t_change in t_change_list:
    f_seir_ld_t = lambda x, t : f_seir_ld(x, t, t_change=t_change,
                                          R0_1=R0_1, R0_2=R0_2)

    y_0 = [s_0, e_0, i_0, r_0, R0_1, 0, 0]

    y = odeint(f_seir_ld_t, y_0, tspan)
    deaths = N * delta * gamma * y[:, 2]
    _ = plt.plot(tspan, deaths)
```