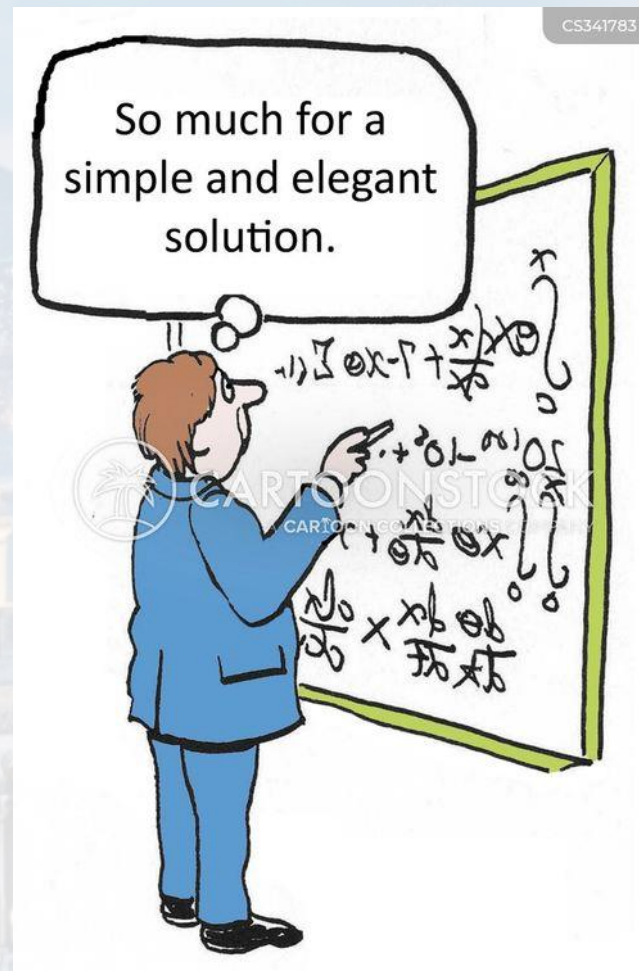


M. Hohle:

Physics 77: Introduction to Computational Techniques in Physics



syllabus:

- Introduction to Unix & Python (week 1 - 2)
- Functions, Loops, Lists and Arrays (week 3 - 4)
- Visualization (week 5)
- Parsing, Data Processing and File I/O (week 6)
- Statistics and Probability, Interpreting Measurements (week 7 - 8)
- Random Numbers, Simulation (week 9)
- Numerical Integration and Differentiation (week 10)
- Root Finding, Interpolation (week 11)
- Systems of Linear Equations (week 12)
- **Ordinary Differential Equations (week 13)**
- Fourier Transformation and Signal Processing (week 14)
- Capstone Project Presentations (week 15)



ordinary differential equation:

- **total** derivative \rightarrow *ordinary*

$$\frac{d^k f(x)}{dx^k} \quad k \in \mathbb{N}$$

- of **n-th** order $\rightarrow n = \max(k)$

- **non-linear** \rightarrow *power of **any** x is not one*

partial differential equation:

- at least one **partial** derivative

$$\frac{\partial y(x)}{\partial t} = [a\Delta + bg(x)] y(x)$$

diffusion

$$\frac{\partial^2 y(x)}{\partial t^2} = [a\Delta + bg(x)] y(x)$$

wave

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python



let's start simple:

constant **relative change** per **time step**

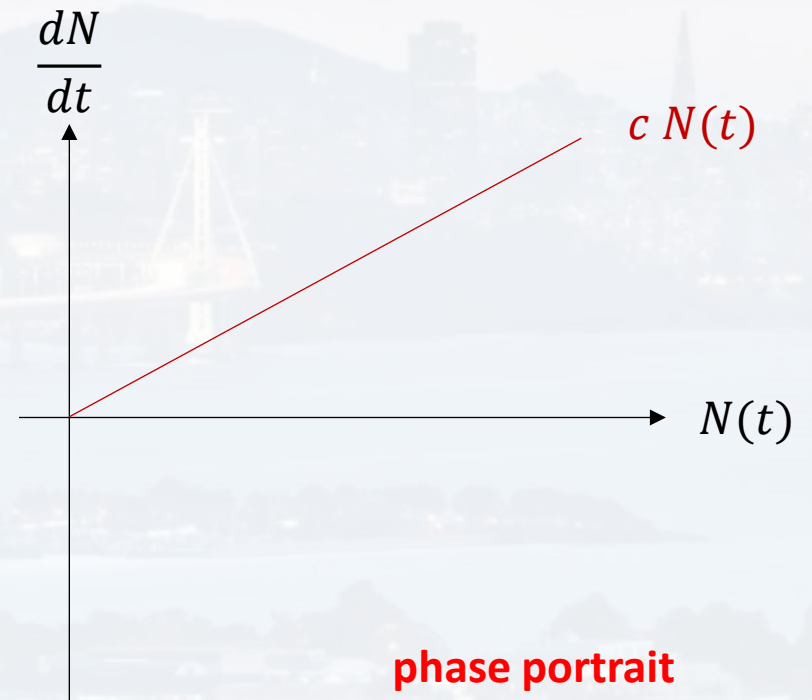
$$\frac{\Delta N}{N} \frac{1}{\Delta t} = c$$

$$\frac{dN}{N} = c dt$$

$$\frac{dN}{dt} = c N$$

$$\int_{N(t=0)}^N \frac{1}{N} dN = c \int_0^{\tau} dt$$

$$N(t) = N(t=0) e^{ct}$$



What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python



let's start simple:

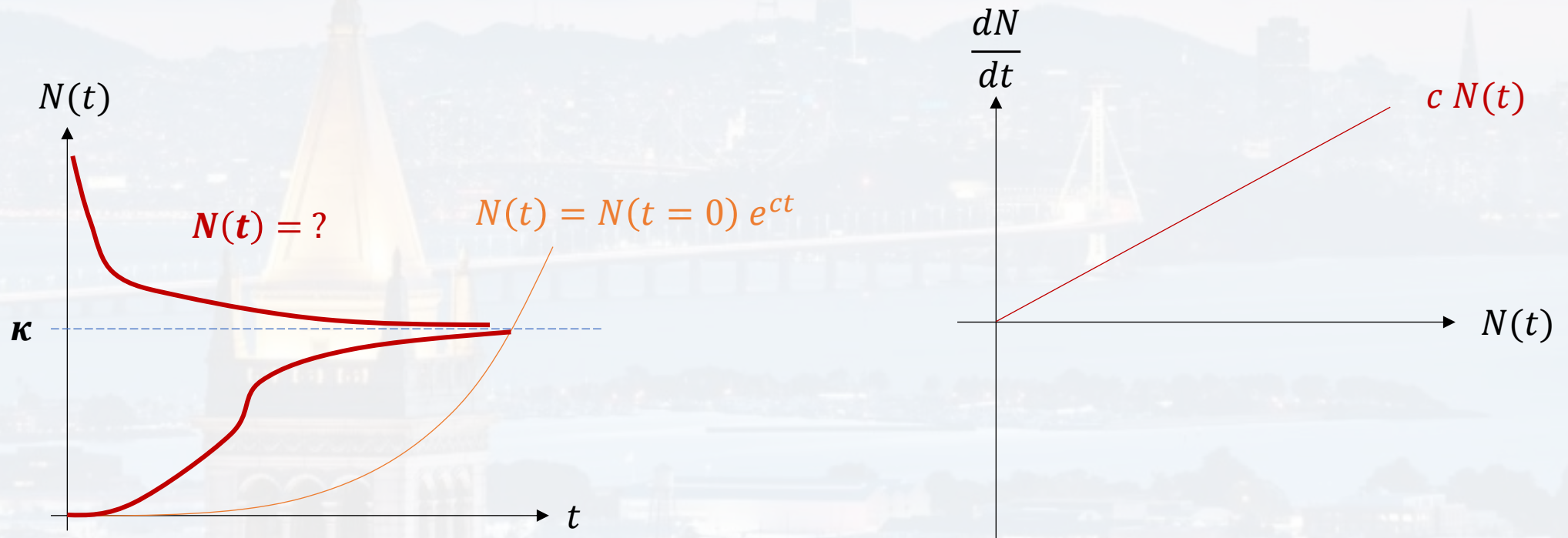
What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

We want the model to have a limited carrying capacity κ

$$\frac{dN}{N} = c dt$$





let's start simple:

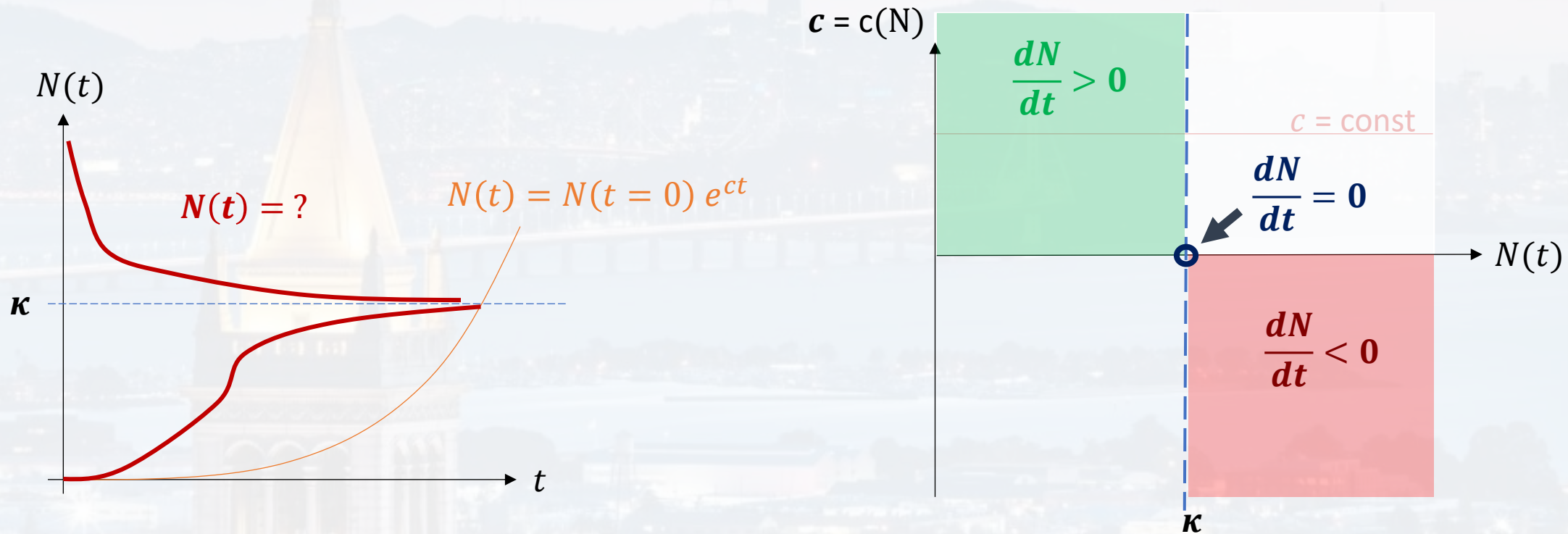
What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

We want the model to have a limited carrying capacity κ

$$\frac{dN}{N} = c dt$$





let's start simple:

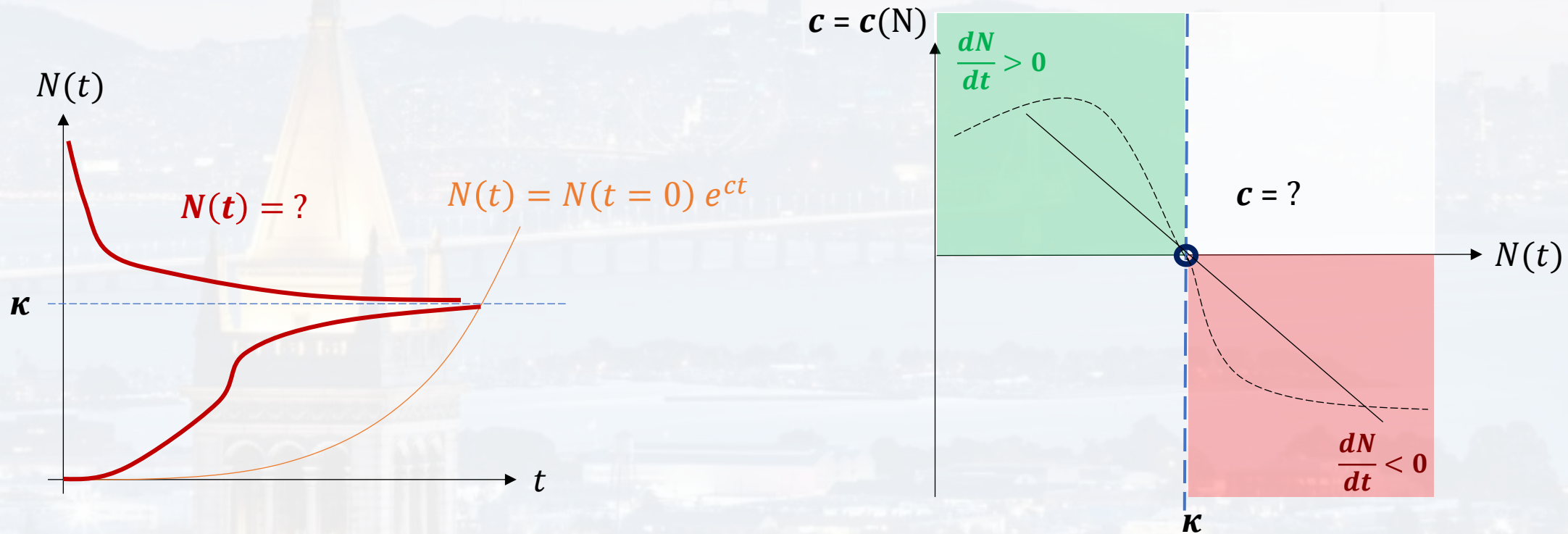
What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

We want the model to have a limited carrying capacity κ

$$\frac{dN}{N} = c dt$$

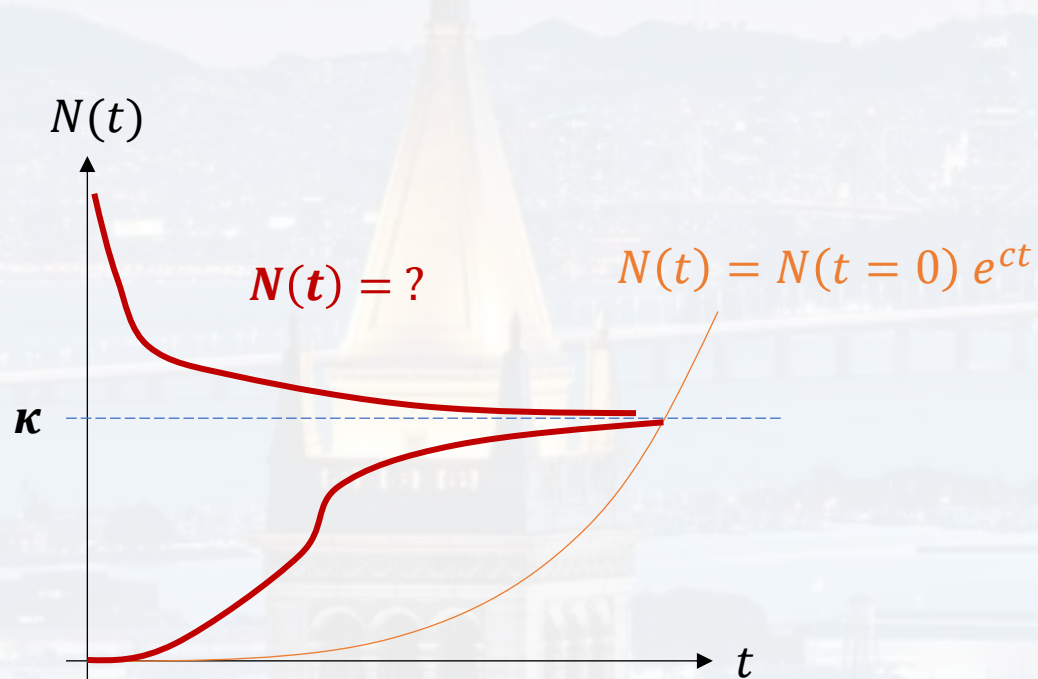




let's start simple:

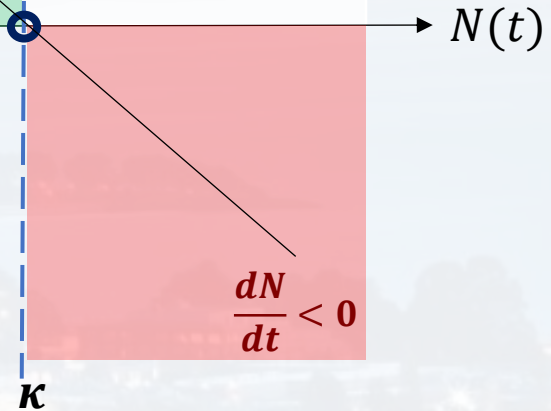
We want the model to have a limited carrying capacity κ

$$\frac{dN}{N} = c dt$$



$c = c(N)$

$$\frac{dN}{dt} > 0$$



What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

Occam's razor:

prefer simple model
(Bayesian Model Selection)



let's start simple:

We want the model to have a limited carrying capacity κ

$$\frac{dN}{N} = c dt$$

$$c(N) = c_0 + m N$$

$$c(\kappa) = 0 \quad c(\kappa) = 0 = c_0 + m\kappa \quad m = -\frac{c_0}{\kappa}$$

$$c(0) = c_0$$

$$c(N) = c_0 \left(1 - \frac{1}{\kappa} N \right)$$

$$\frac{dN}{N} = c_0 \left(1 - \frac{1}{\kappa} N \right) dt$$

Verhulst Equation

$c = c(N)$

$\frac{dN}{dt} > 0$

κ

$\frac{dN}{dt} < 0$

$N(t)$

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

Occam's razor:

prefer simple model
(Bayesian Model Selection)



let's start simple:

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

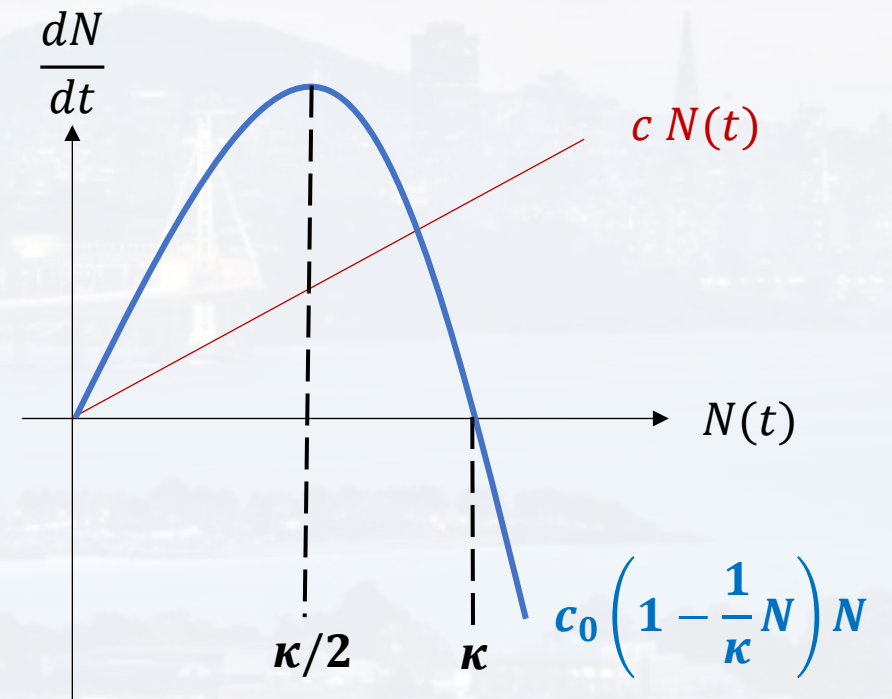
We want the model to have a limited carrying capacity κ

$$\frac{dN}{N} = c dt$$

$$\frac{dN}{N} = c_0 \left(1 - \frac{1}{\kappa} N \right) dt$$

Verhulst Equation

$$\frac{dN}{dt} = c_0 \left(1 - \frac{1}{\kappa} N \right) N$$





let's start simple:

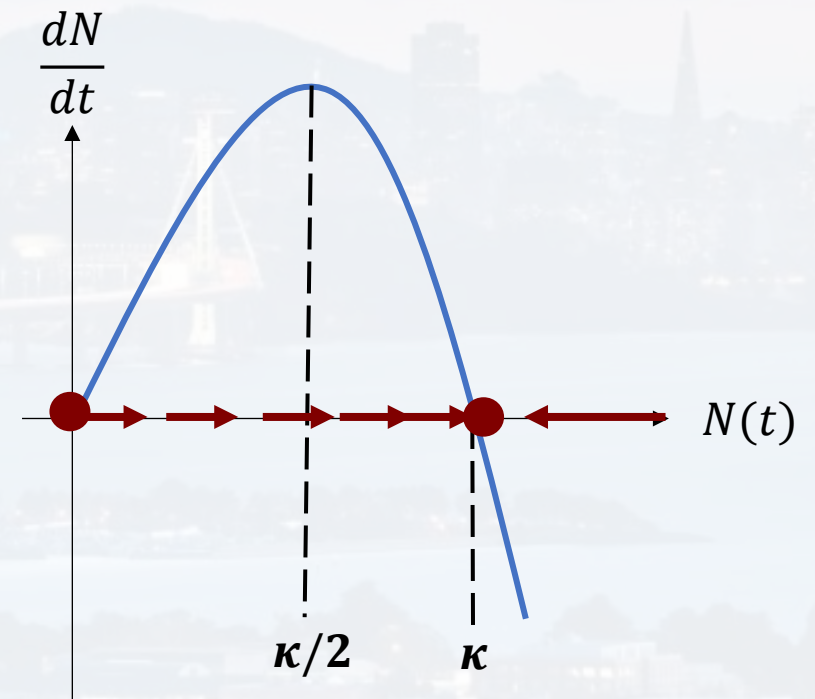
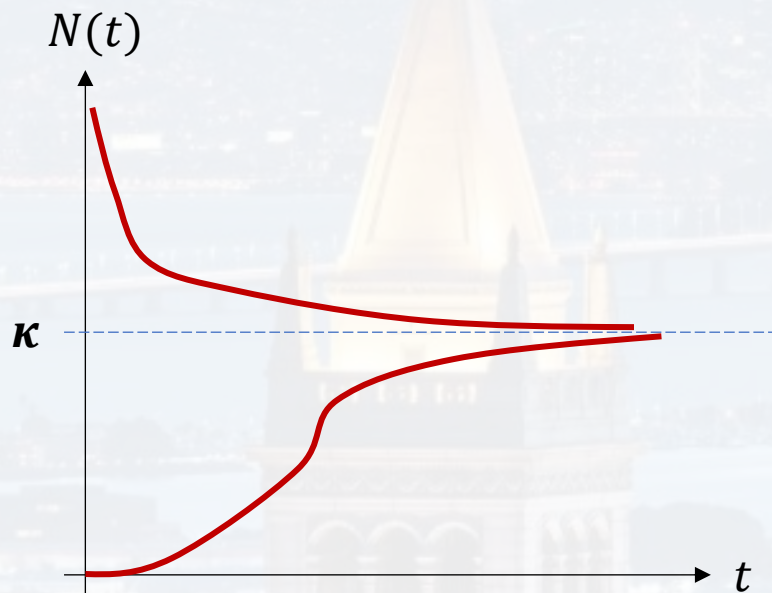
What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

We want the model to have a limited carrying capacity κ

$$\frac{dN}{dt} = c_0 \left(1 - \frac{1}{\kappa} N \right) N \quad \text{Verhulst Equation}$$





let's start simple:

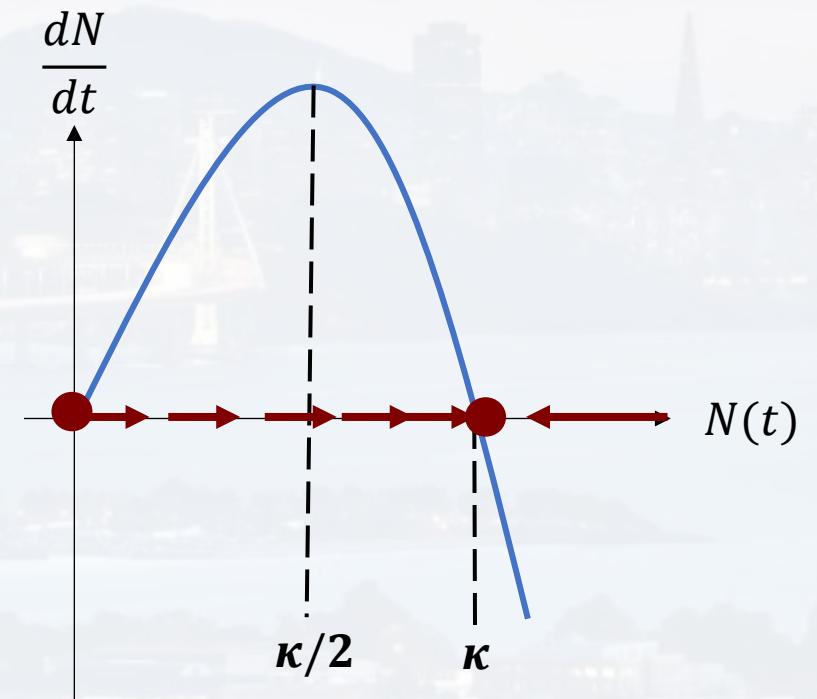
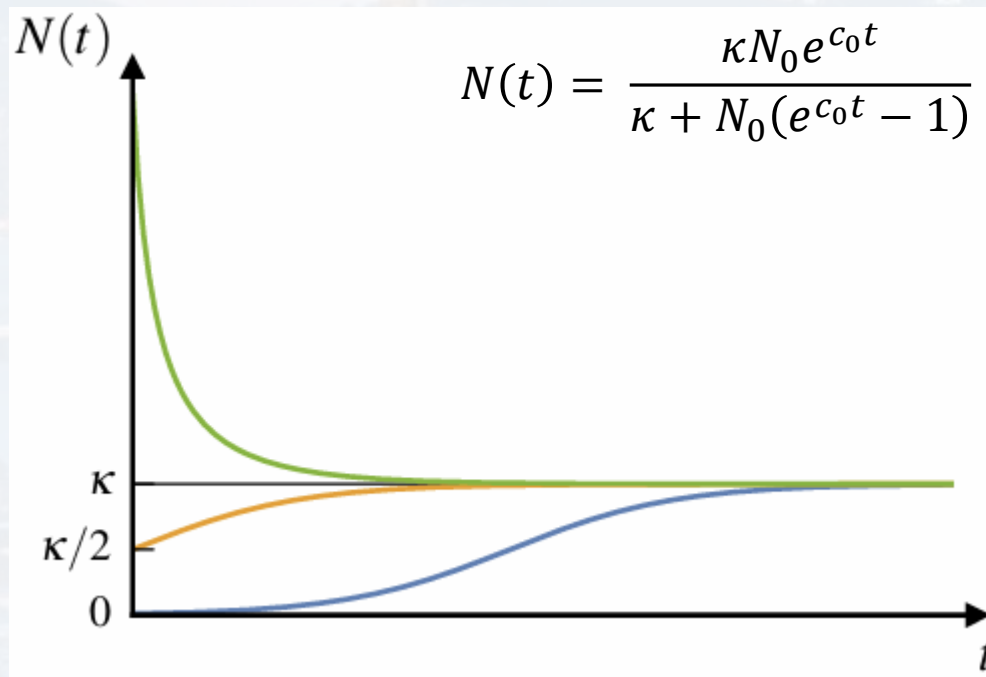
What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

We want the model to have a limited carrying capacity κ

$$\frac{dN}{dt} = c_0 \left(1 - \frac{1}{\kappa} N \right) N \quad \text{Verhulst Equation}$$





let's start simple:

What is an ODE?

Solving ODEs by thinking

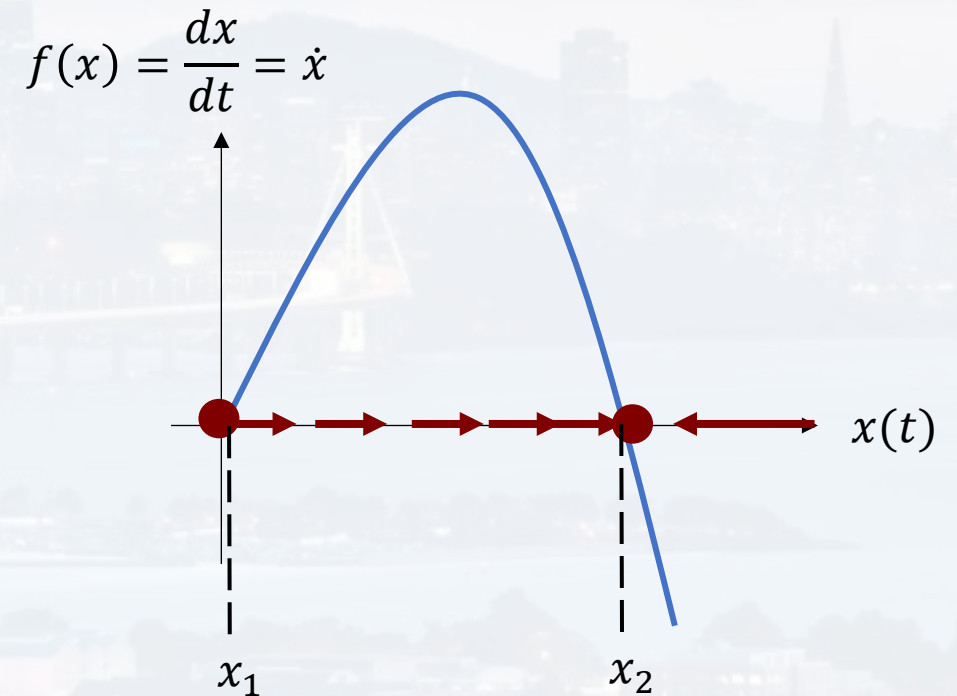
Solving ODEs with Python

We want the model to have a limited carrying capacity κ

$$\frac{dN}{dt} = c_0 \left(1 - \frac{1}{\kappa} N \right) N \quad \text{Verhulst Equation}$$

x_1, x_2 fixed points

$$f(x) = \frac{dx}{dt} = \dot{x}$$





fixed points x^*

x_1 : repeller

→ unstable

$$\frac{df(x)}{dx} = \frac{d}{dx}\dot{x} > 0$$

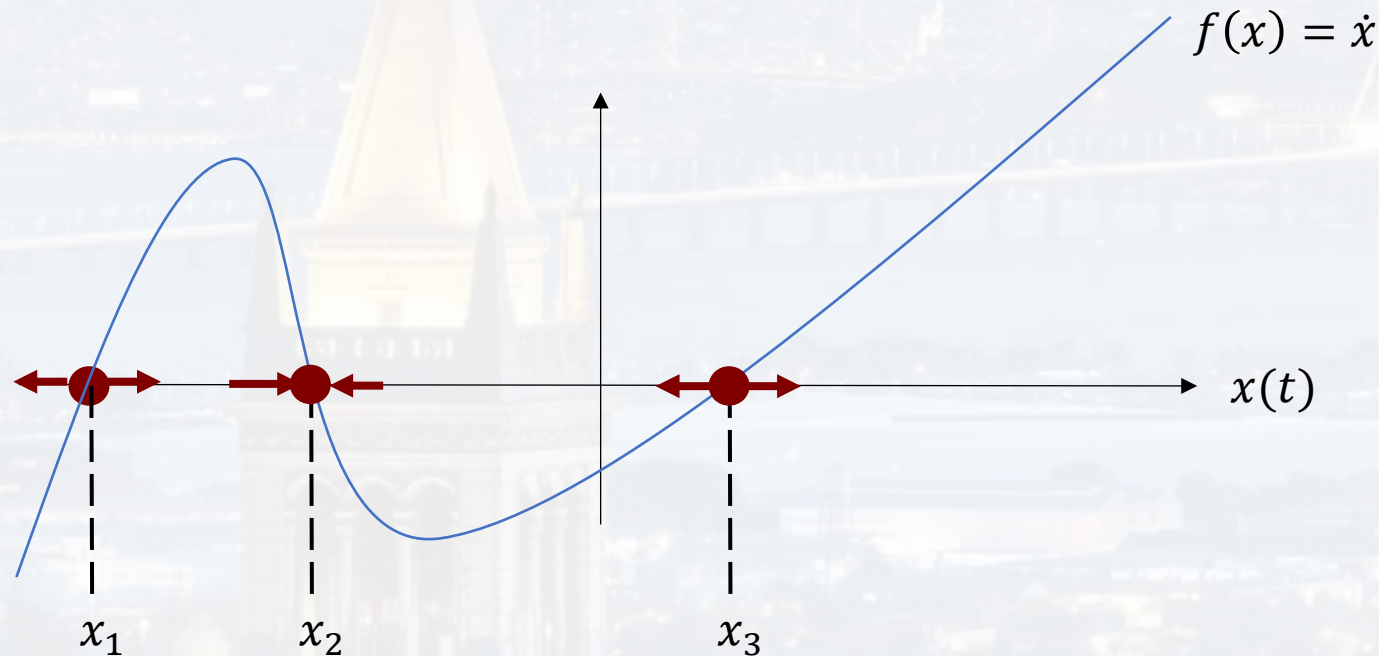
x_3 : repeller

→ unstable

x_2 : attractor

→ stable

$$\frac{df(x)}{dx} = \frac{d}{dx}\dot{x} < 0$$



What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python



fixed points x^*

small perturbation $\varepsilon(t)$

What is an ODE?

Solving ODEs by thinking

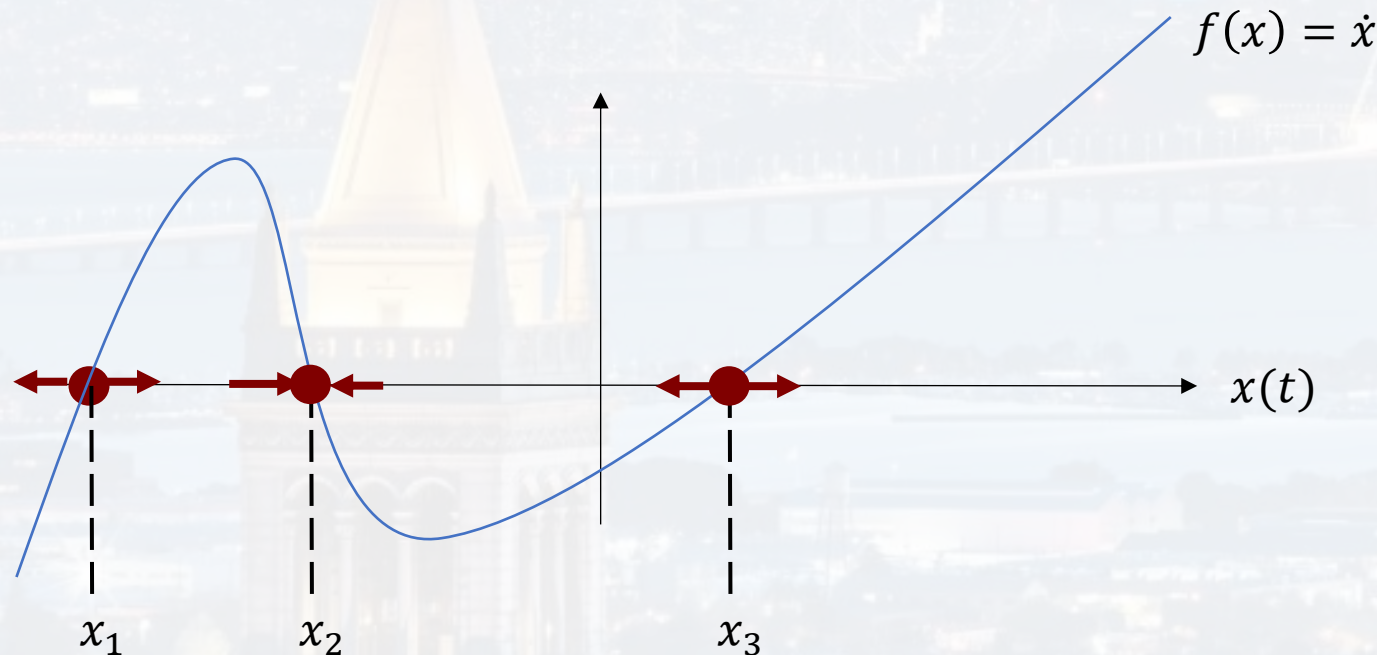
Solving ODEs with Python

$$x(t) = x^* + \varepsilon(t)$$

$$\frac{d\varepsilon(t)}{dt} = \frac{d}{dt}[x(t) - x^*] = f(x) + 0 = f(x^* + \varepsilon(t))$$

$$\boxed{\frac{d\varepsilon(t)}{dt}} = f(x^* + \varepsilon(t)) \approx f(x^*) + \left. \frac{df(x)}{dx} \right|_{x=x^*} \varepsilon(t) = 0 + \boxed{\left. \frac{df(x)}{dx} \right|_{x=x^*} \varepsilon(t)}$$

$$\boxed{\varepsilon(t) = \varepsilon_0 e^{\left. \frac{df(x)}{dx} \right|_{x=x^*} t}}$$



$$\text{time scale } \tau = \frac{1}{\left. \frac{df(x)}{dx} \right|_{x=x^*}}$$

$$\frac{df(x)}{dx} > 0 \quad \text{unstable}$$

$$\frac{df(x)}{dx} < 0 \quad \text{stable}$$



fixed points x^*

small perturbation $\varepsilon(t)$

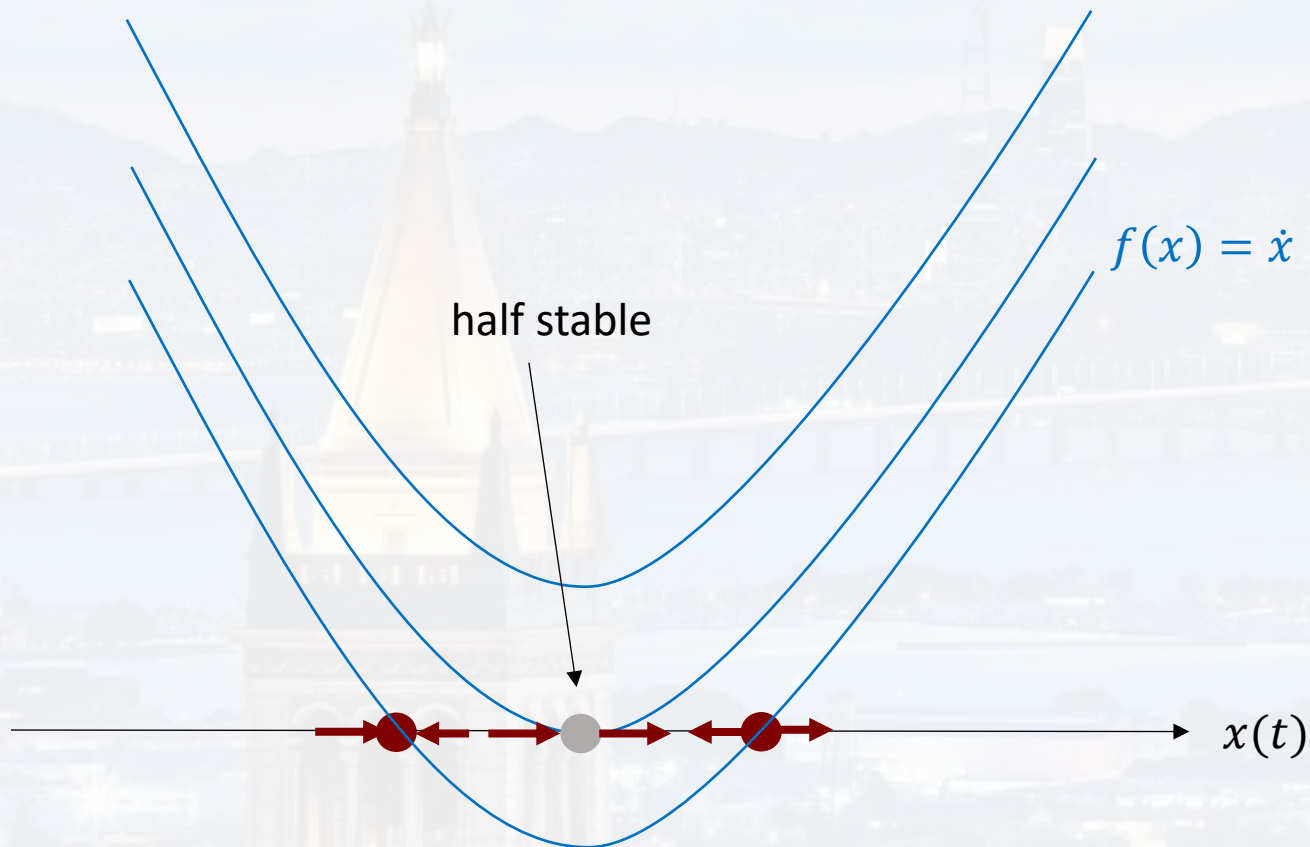
What is an ODE?
Solving ODEs by thinking
Solving ODEs with Python

$$\varepsilon(t) = \varepsilon_0 e^{\frac{df(x)}{dx}|_{x=x^*} t}$$

$$\text{time scale } \tau = \frac{1}{\frac{df(x)}{dx}|_{x=x^*}}$$

$$\frac{df(x)}{dx} > 0 \quad \text{unstable}$$

$$\frac{df(x)}{dx} < 0 \quad \text{stable}$$





fixed points x^*

small perturbation $\varepsilon(t)$

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

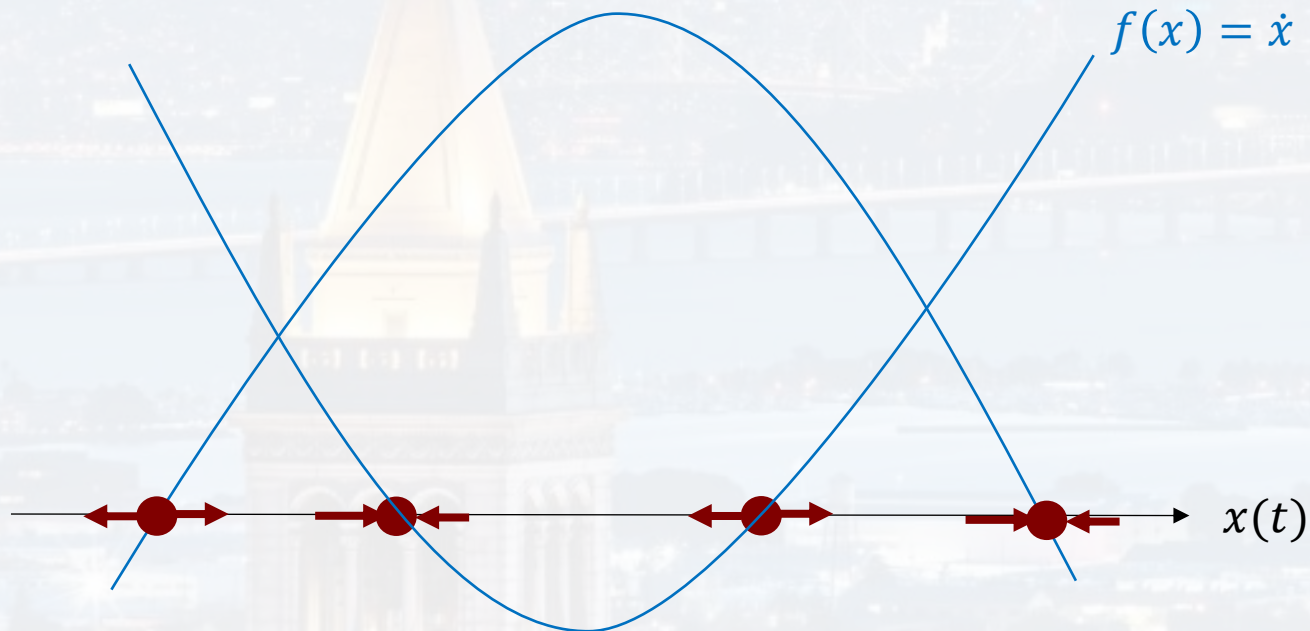
$$\varepsilon(t) = \varepsilon_0 e^{\frac{df(x)}{dx}|_{x=x^*} t}$$

$$\text{time scale } \tau = \frac{1}{\frac{df(x)}{dx}|_{x=x^*}}$$

$$f(x) = ax^2 + bx + c$$

$$\frac{df(x)}{dx} > 0 \quad \text{unstable}$$

$$\frac{df(x)}{dx} < 0 \quad \text{stable}$$



if coupled to another system: \rightarrow can change dynamics drastically (chem reactions)



2D system

$$f(x, y) = \dot{x}$$

$$\dot{x} = -x + a y + x^2 y$$

$$g(x, y) = \dot{y}$$

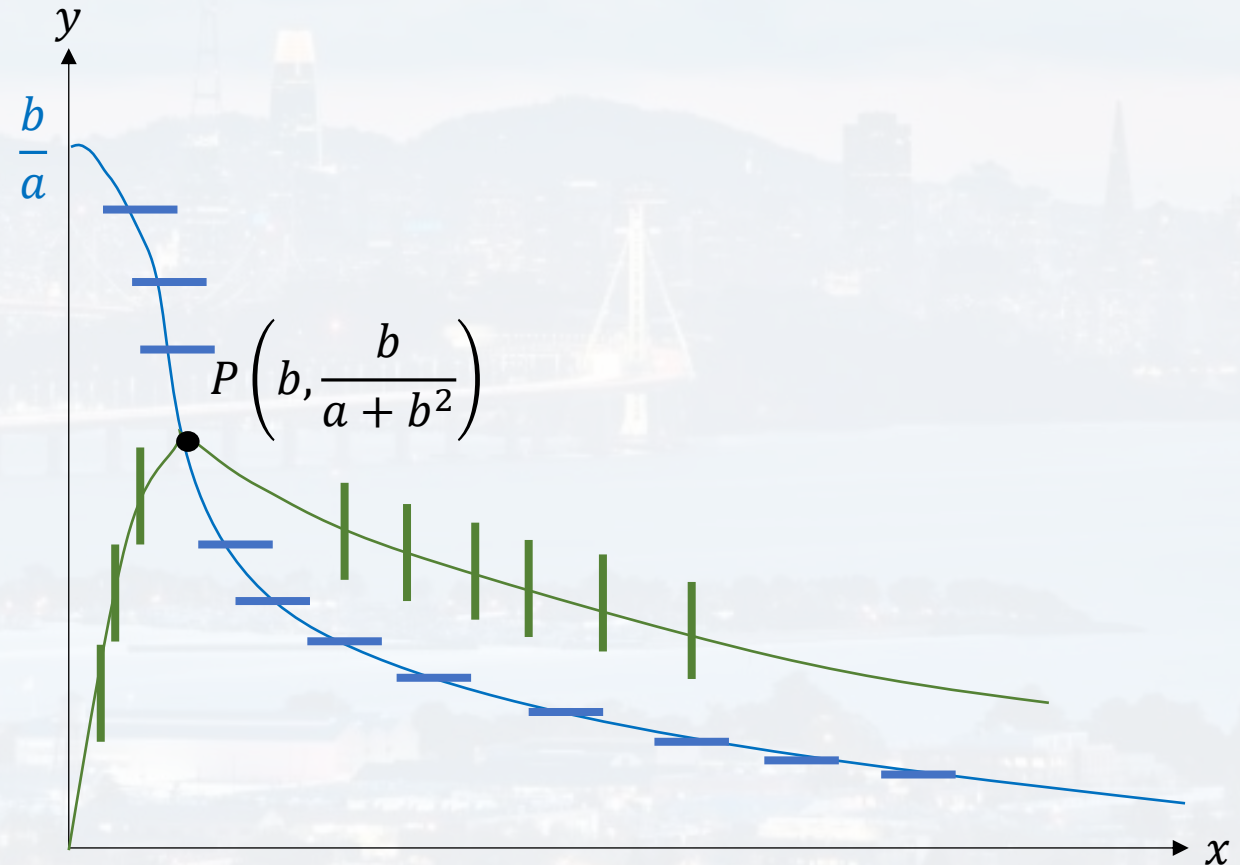
$$\dot{y} = b - a y - x^2 y$$

non-linear, coupled ODEs

null clines

$$\dot{x} = 0 \rightarrow y_1 = \frac{x}{a + x^2}$$

$$\dot{y} = 0 \rightarrow y_2 = \frac{b}{a + x^2}$$



Find out which way the system moves!

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python



2D system

$$f(x, y) = \dot{x}$$

$$\dot{x} = -x + a y + x^2 y$$

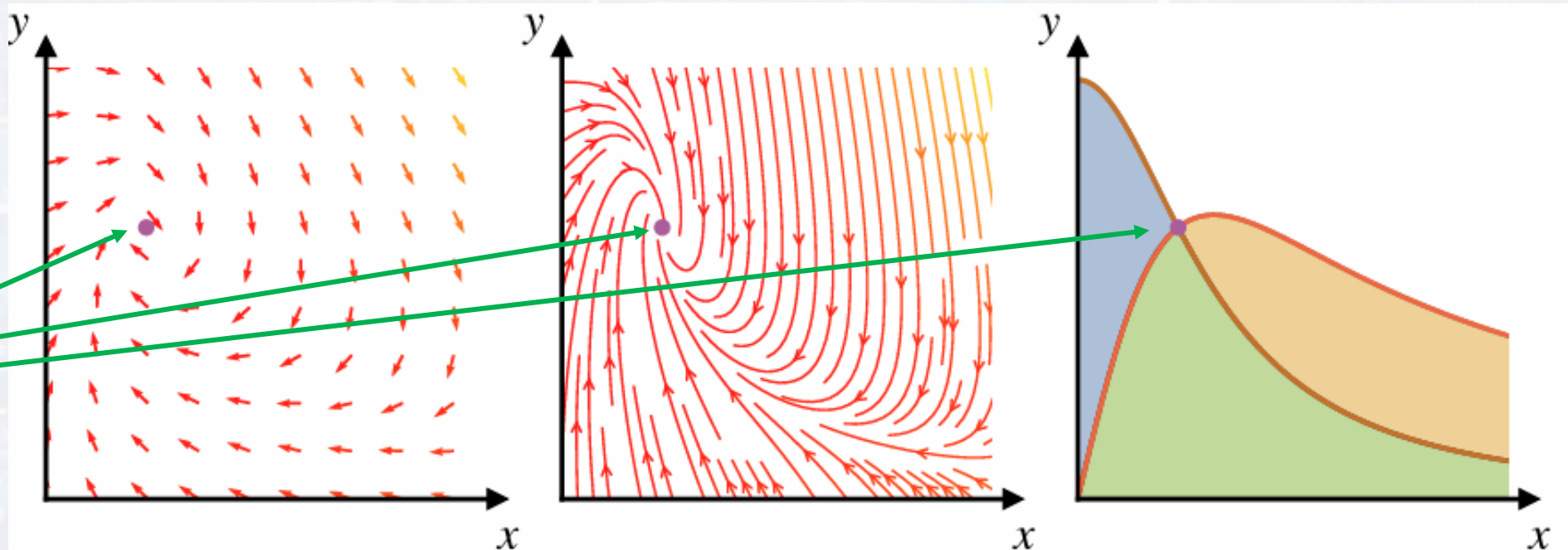
$$g(x, y) = \dot{y}$$

$$\dot{y} = b - a y - x^2 y$$

non-linear, coupled ODEs

$$\dot{x} = 0 \Rightarrow y_1 = \frac{x}{a + x^2}$$

$$\dot{y} = 0 \Rightarrow y_2 = \frac{b}{a + x^2}$$



attractor or
repeller?

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python



2D system

$$f(x, y) = \dot{x}$$

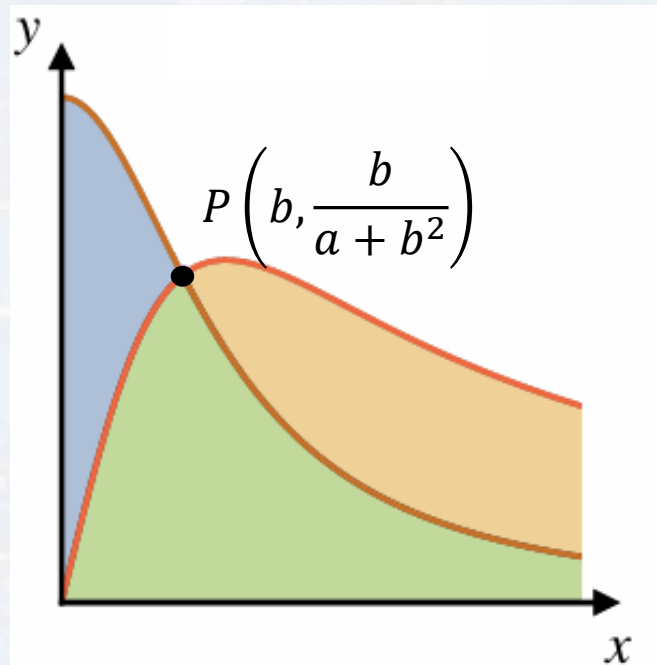
$$\dot{x} = -x + a y + x^2 y$$

$$g(x, y) = \dot{y}$$

$$\dot{y} = b - a y - x^2 y$$

non-linear, coupled ODEs

stability of P:



$$\frac{d \varepsilon_x(t)}{dt} \approx f(x^* + \varepsilon_x, y^* + \varepsilon_y) - f(x^*, y^*) \approx \frac{\partial f(x, y)}{\partial x} \Big|_{x^*, y^*} \varepsilon_x + \frac{\partial f(x, y)}{\partial y} \Big|_{x^*, y^*} \varepsilon_y$$

α β

$$\frac{d \varepsilon_y(t)}{dt} \approx g(x^* + \varepsilon_x, y^* + \varepsilon_y) - g(x^*, y^*) \approx \frac{\partial g(x, y)}{\partial x} \Big|_{x^*, y^*} \varepsilon_x + \frac{\partial g(x, y)}{\partial y} \Big|_{x^*, y^*} \varepsilon_y$$

$= 0$ γ δ

$$\dot{\vec{\varepsilon}} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \vec{\varepsilon}$$

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python



2D system

$$f(x, y) = \dot{x}$$

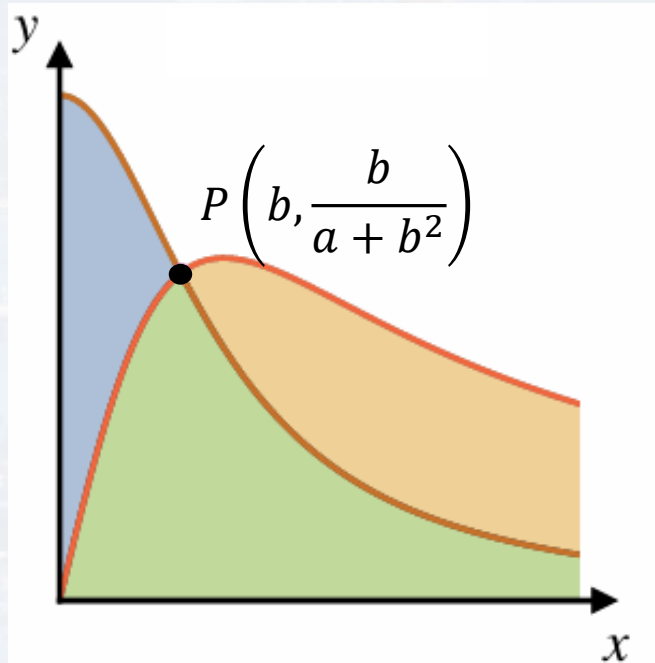
$$\dot{x} = -x + a y + x^2 y$$

$$g(x, y) = \dot{y}$$

$$\dot{y} = b - a y - x^2 y$$

non-linear, coupled ODEs

stability of P:



$$\dot{\vec{\epsilon}} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \vec{\epsilon} \quad A = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

$$\vec{\epsilon}(t) = \vec{\epsilon}(t=0) e^{\lambda t}$$

$$\epsilon(t) = \epsilon_0 e^{\frac{df(x)}{dx}|_{x=x^*} t}$$

λ : eigenvalue of A

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python



2D system

$$f(x, y) = \dot{x}$$

$$\dot{x} = -x + a y + x^2 y$$

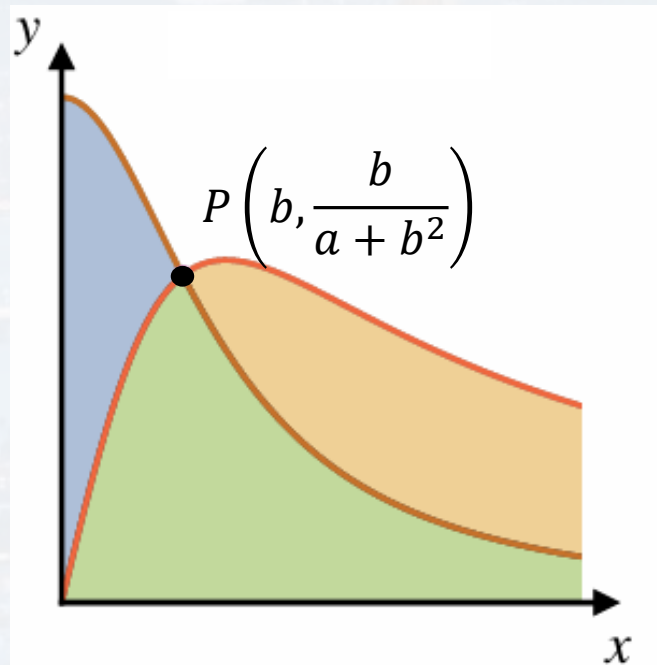
$$g(x, y) = \dot{y}$$

$$\dot{y} = b - a y - x^2 y$$

non-linear, coupled ODEs

stability of P:

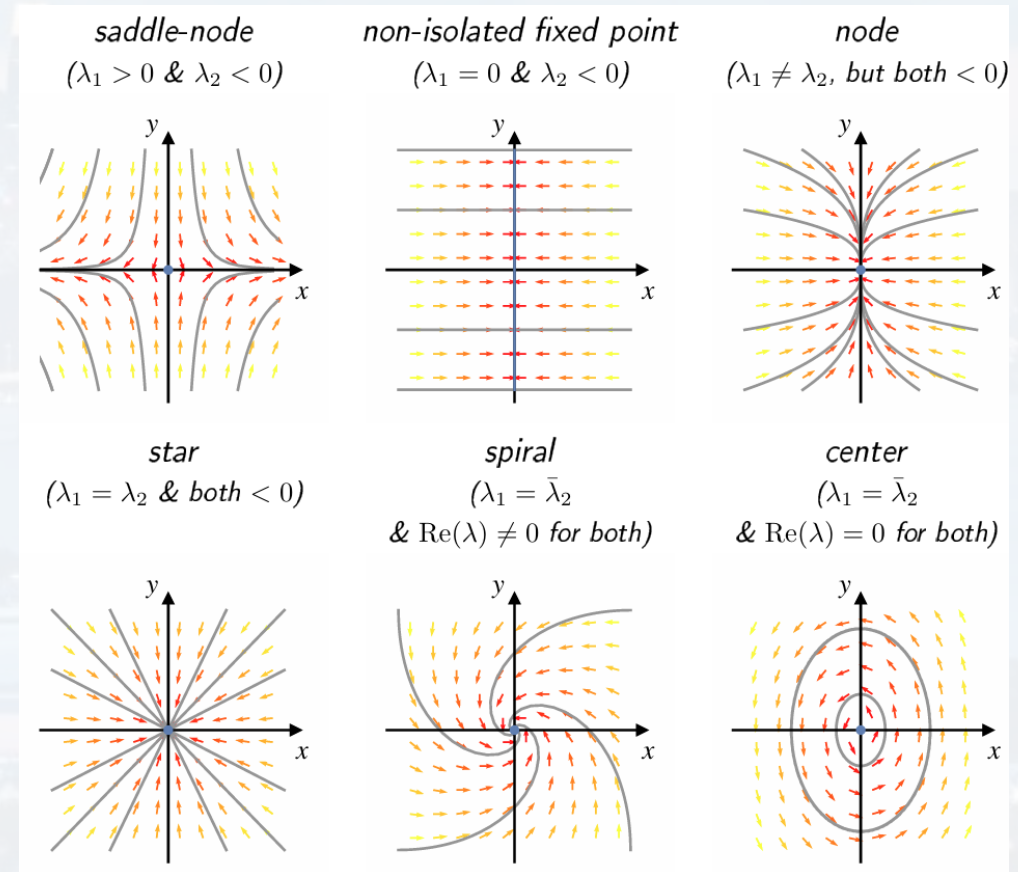
$$\vec{\epsilon}(t) = \vec{\epsilon}(t=0) e^{\lambda t}$$



What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python





2D system

$$f(x, y) = \dot{x}$$

$$\dot{x} = -x + a y + x^2 y$$

$$g(x, y) = \dot{y}$$

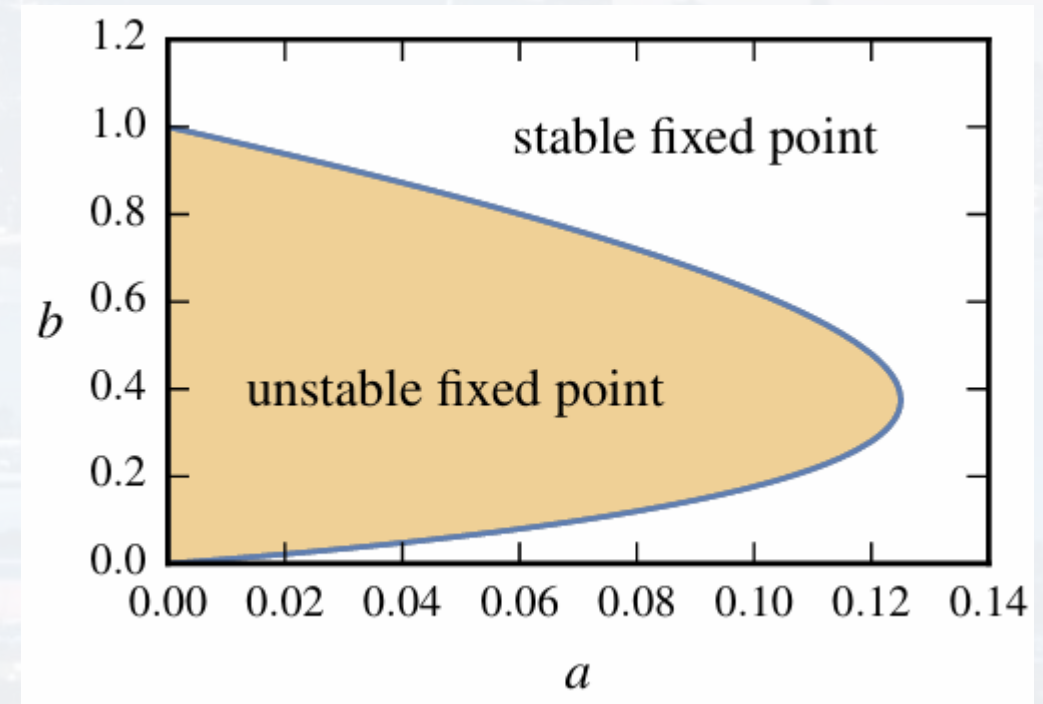
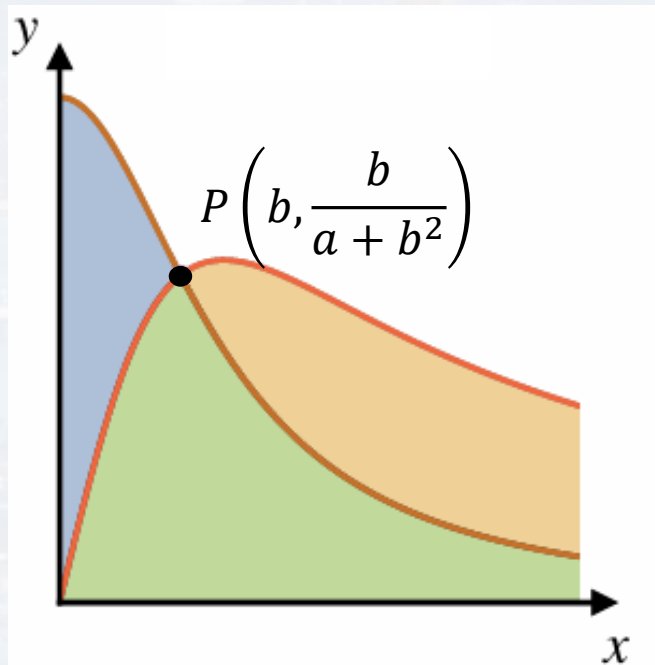
$$\dot{y} = b - a y - x^2 y$$

non-linear, coupled ODEs

stability of P:

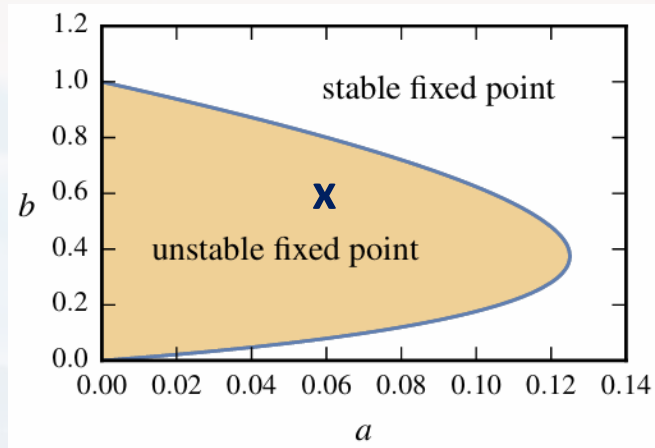
$$\vec{\epsilon}(t) = \vec{\epsilon}(t=0) e^{\lambda t}$$

for this particular system:





2D system

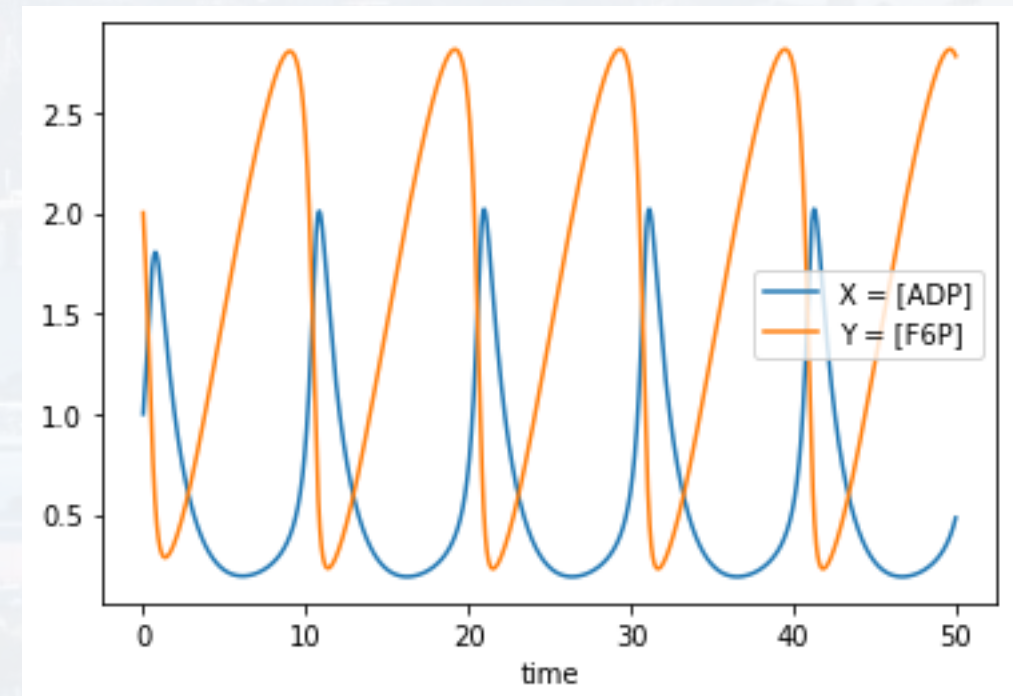
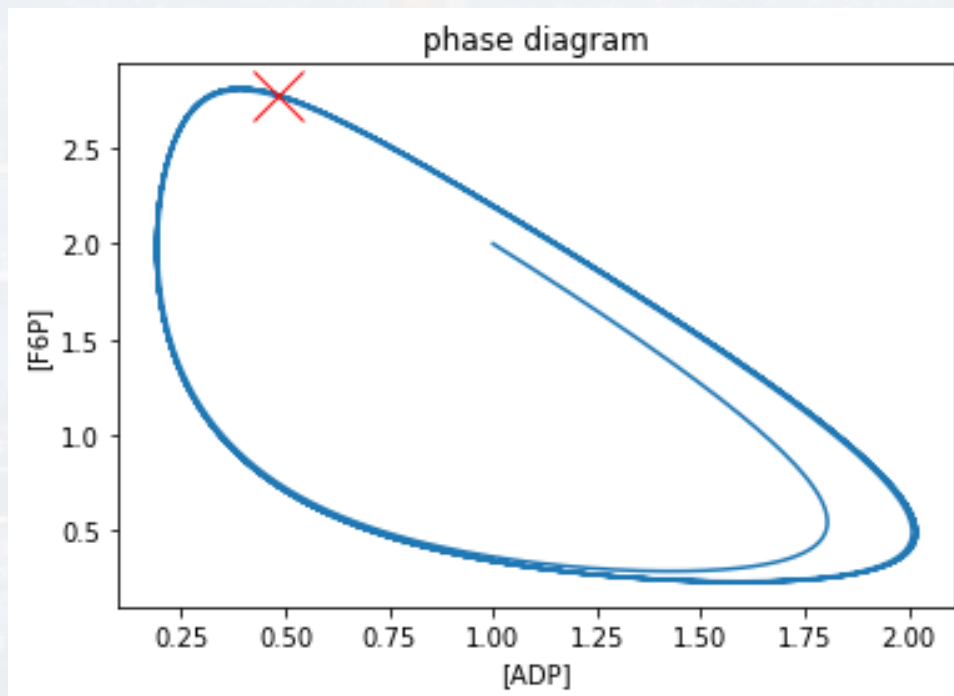


$$P\left(b, \frac{b}{a + b^2}\right) = (0.6, 1.42)$$

What is an ODE?

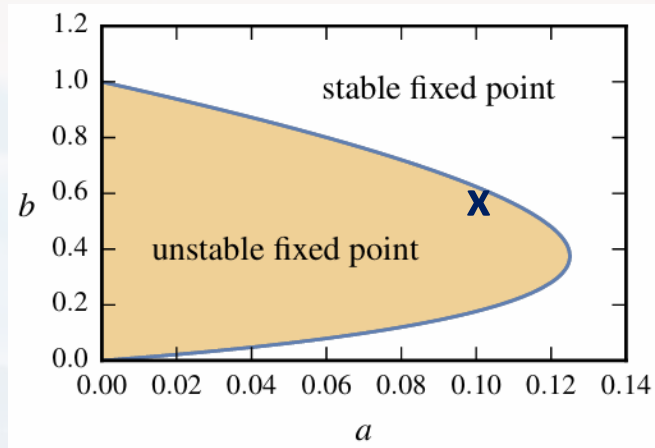
Solving ODEs by thinking

Solving ODEs with Python





2D system

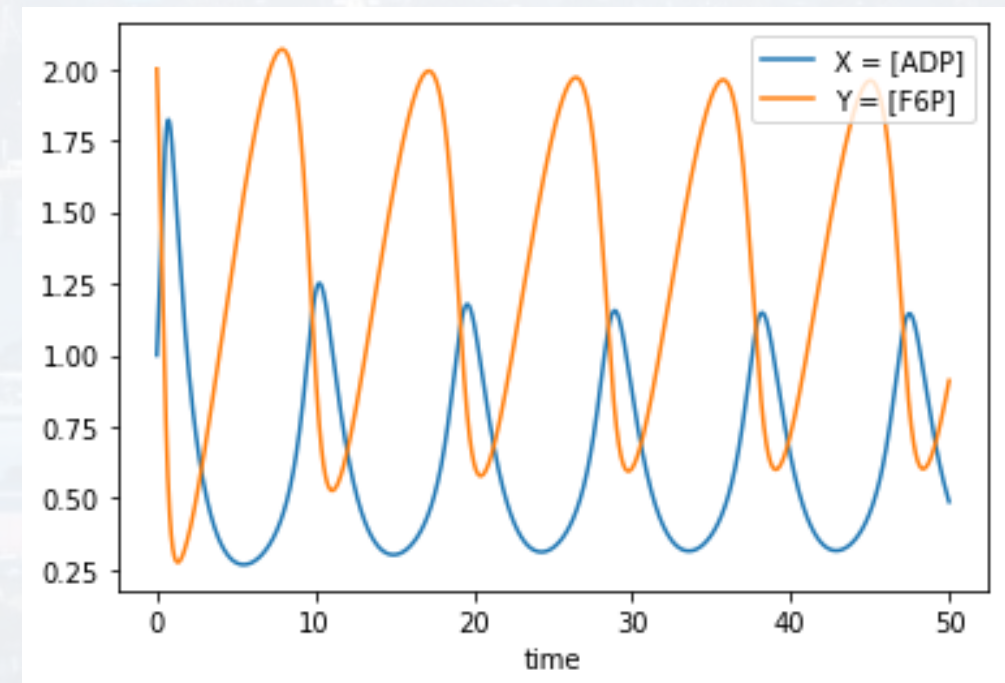
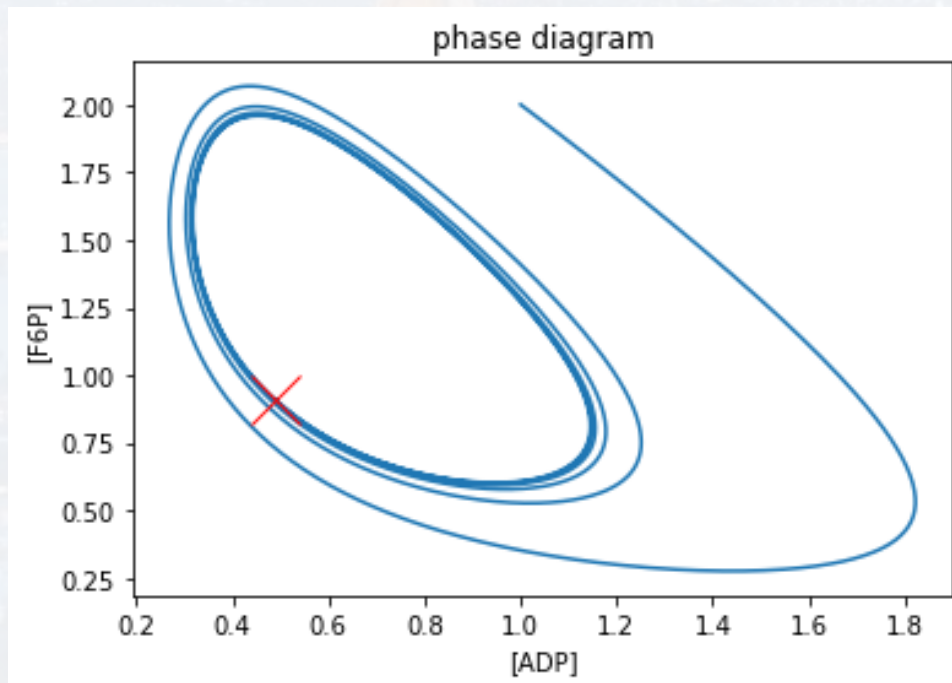


$$P\left(b, \frac{b}{a + b^2}\right) = (0.6, 1.30)$$

What is an ODE?

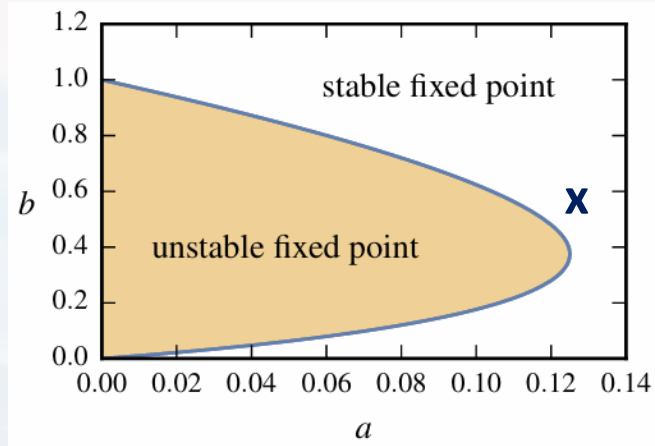
Solving ODEs by thinking

Solving ODEs with Python





2D system

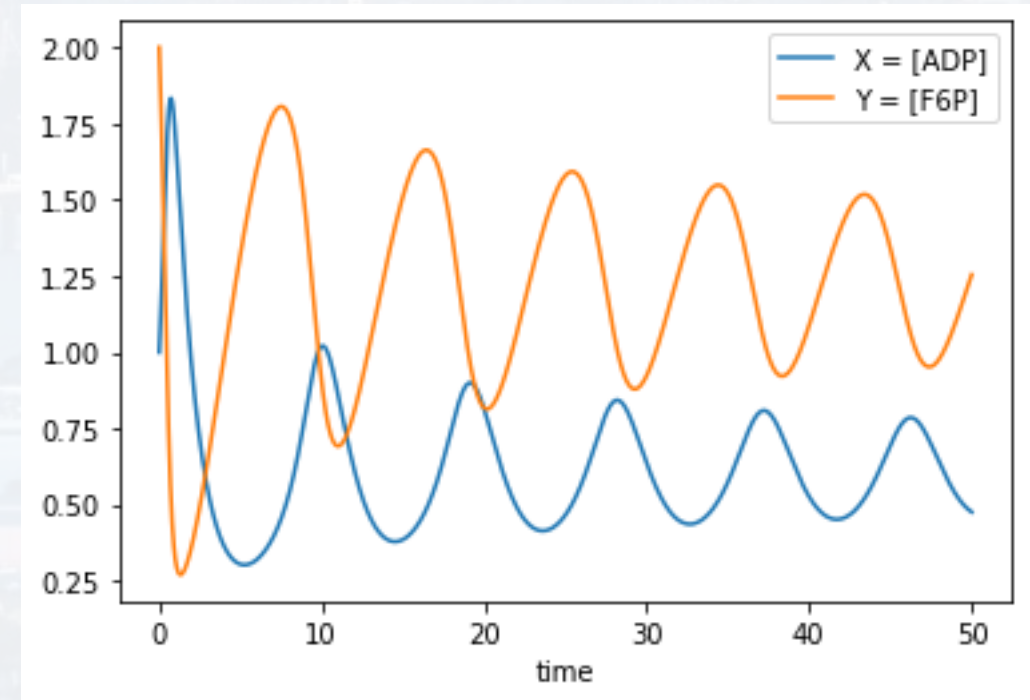
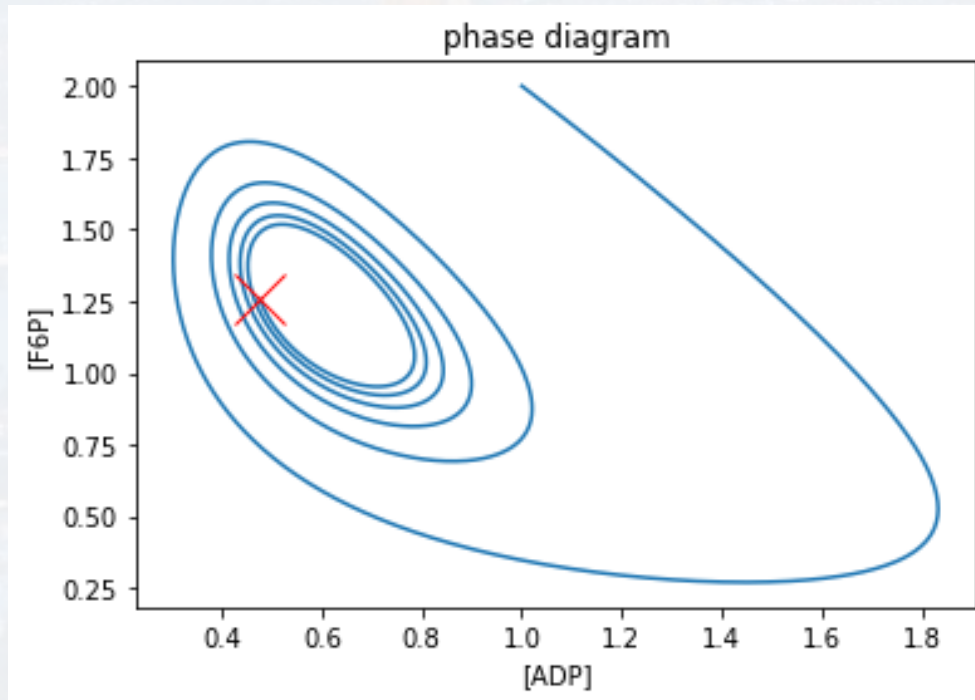


$$P\left(b, \frac{b}{a + b^2}\right) = (0.6, 1.24)$$

What is an ODE?

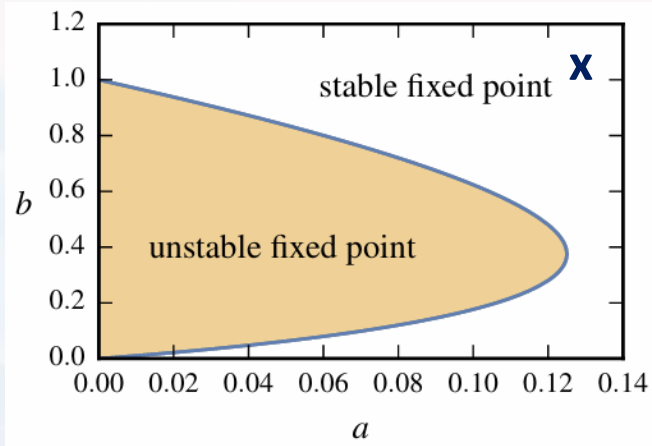
Solving ODEs by thinking

Solving ODEs with Python





2D system

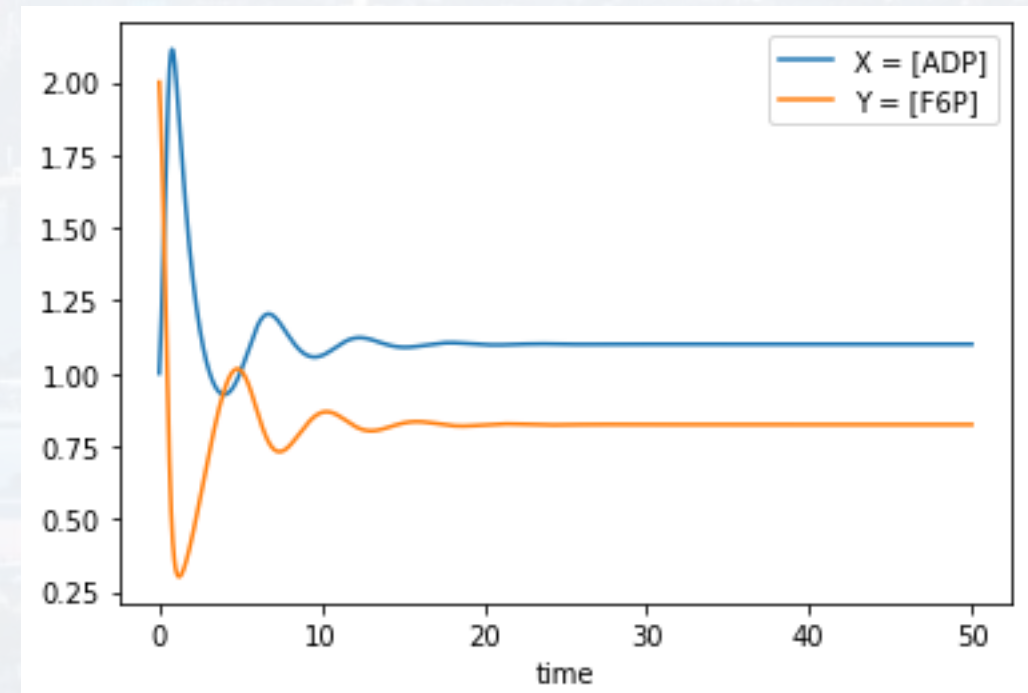
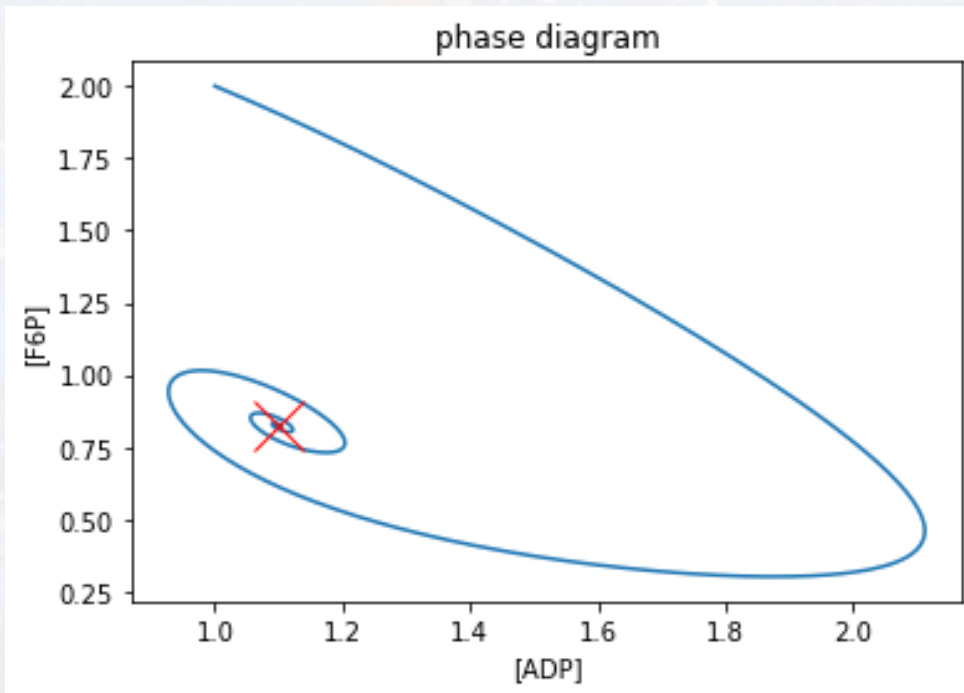


$$P\left(b, \frac{b}{a + b^2}\right) = (1.1, 0.82)$$

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python





Runge-Kutta-Heun

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

$$\frac{dy}{dt} = f(x(t), t)$$

initial condition:

$$y_0 = y(t_0)$$

goal:

$$y(t)$$

$$y(t + dt) = y(t) + \frac{dy}{dt} dt + \frac{1}{2} \frac{d^2 y}{dt^2} dt^2 + \dots$$

$$y(t + dt) = y(t) + f(x, t) dt + \frac{1}{2} \frac{d}{dt} f(x, t) dt^2 + \dots$$

$$y(t + dt) = y(t) + f(x, t) dt + \frac{1}{2} \left[\frac{\partial f}{\partial x} \frac{dx}{dt} + \frac{\partial f}{\partial t} \right] dt^2 + \dots$$

note: more general

$$\frac{dy}{dx} = f(x(t), y(x(t)))$$



Runge-Kutta-Heun

```
from scipy.integrate import solve_ivp
```

```
method = 'RK45'
```

4: referring to the number of subintervals for integration (4 is equivalent to the Simpson rule)

5: referring to the order of the Taylor approximation for the derivatives

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python



$$\dot{x} = -x + a y + x^2 y$$

$$\dot{y} = b - a y - x^2 y$$

What is an ODE?
Solving ODEs by thinking
Solving ODEs with Python

```
def SolveGlycolysis(Init, t_span, a, b):
    XY = ode_solver(Glycolysis, Init, t_span, method = 'RK45',\
                     a = a, b = b)

    t = XY.t
    X = XY.y[0,:]
    Y = XY.y[1,:]

    #####plotting result#####

    #...
```

initial values: $x(t=0)$ and $y(t=0)$

$[t_{\text{start}}, t_{\text{end}}]$

parameter of the function

contains the actual ODEs



$$\dot{x} = -x + a y + x^2 y$$

$$\dot{y} = b - a y - x^2 y$$

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

```
def SolveGlycolysis(Init, t_span, a, b):
```

```
    XY = ode_solver(Glycolysis, Init, t_span, method = 'RK45',\n                    a = a, b = b)
```

```
def Glycolysis(Init, t, a, b):
```

```
    x = Init[0]
```

```
    y = Init[1]
```

```
    dx = -x + a*y + (x**2)*y
```

```
    dy = b - a*y - (x**2)*y
```

```
    D = [dx, dy]
```

```
    return D
```

note: t is an input
variable, even though it is
not being used explicitly
→ integration over t



$$\dot{x} = -x + a y + x^2 y$$

$$\dot{y} = b - a y - x^2 y$$

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

```
def SolveGlycolysis(Init, t_span, a, b):
```

```
    XY = ode_solver(Glycolysis, Init, t_span, method = 'RK45',\n                    a = a, b = b)
```

```
    t = XY.t  
    X = XY.y[0,:]  
    Y = XY.y[1,:]
```

```
#####plotting result#####
```

```
#...
```

the actual solver



$$\dot{x} = -x + a y + x^2 y$$

$$\dot{y} = b - a y - x^2 y$$

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

```
def SolveGlycolysis(Init, t_span, a, b):
```

```
    XY = ode_solver(Glycolysis, Init, t_span, method = 'RK45',\n                    a = a, b = b)
```

```
from scipy.integrate import solve_ivp
```

```
def ode_solver(ode_func, Init, t_span, method = 'RK45', **params):
```

```
    result = solve_ivp(fun = lambda t, y: ode_func(y, t, **params),\n                       t_span = t_span, y0 = Init, method = method,\n                       rtol = 1e-9, atol = 1e-9, max_step = 0.01)
```

```
    return result
```



$$\dot{x} = -x + a y + x^2 y$$

$$\dot{y} = b - a y - x^2 y$$

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

```
def SolveGlycolysis(Init, t_span, a, b):
```

```
    XY = ode_solver(Glycolysis, Init, t_span, method = 'RK45',\n                    a = a, b = b)
```

```
from scipy.integrate import solve_ivp
```

```
def ode_solver(ode_func, Init, t_span, method = 'RK45', **params):
```

```
    result = solve_ivp(fun = lambda t, y: ode_func(y, t, **params),\n                      t_span = t_span, y0 = Init, method = method,\n                      rtol = 1e-9, atol = 1e-9, max_step = 0.01)
```

```
    return result
```



$$\dot{x} = -x + a y + x^2 y$$

$$\dot{y} = b - a y - x^2 y$$

run:

$a = 0.125$

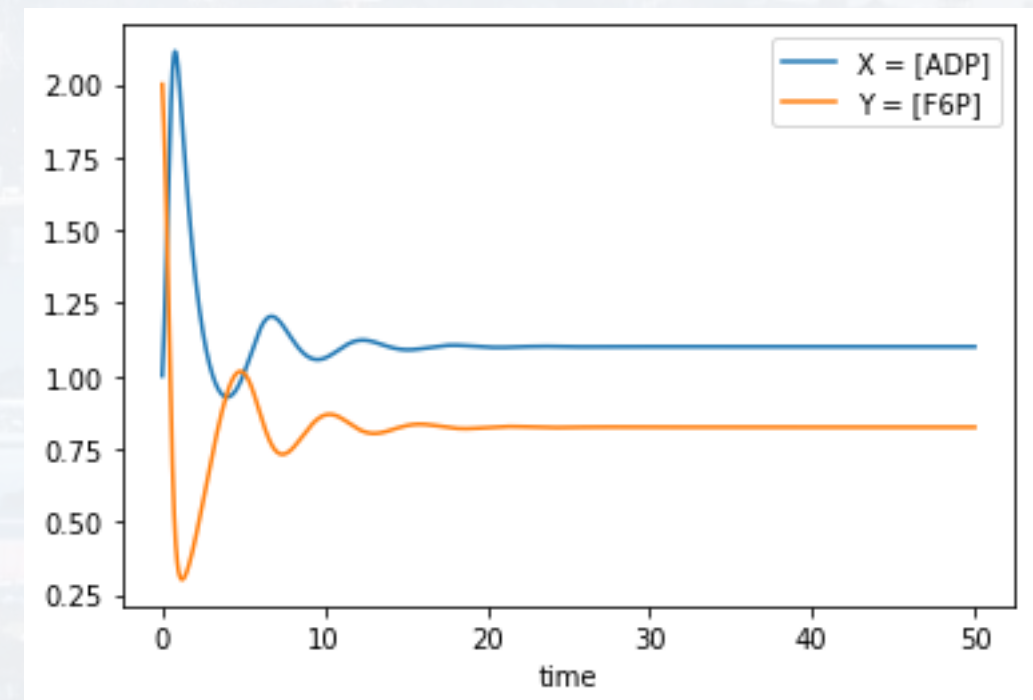
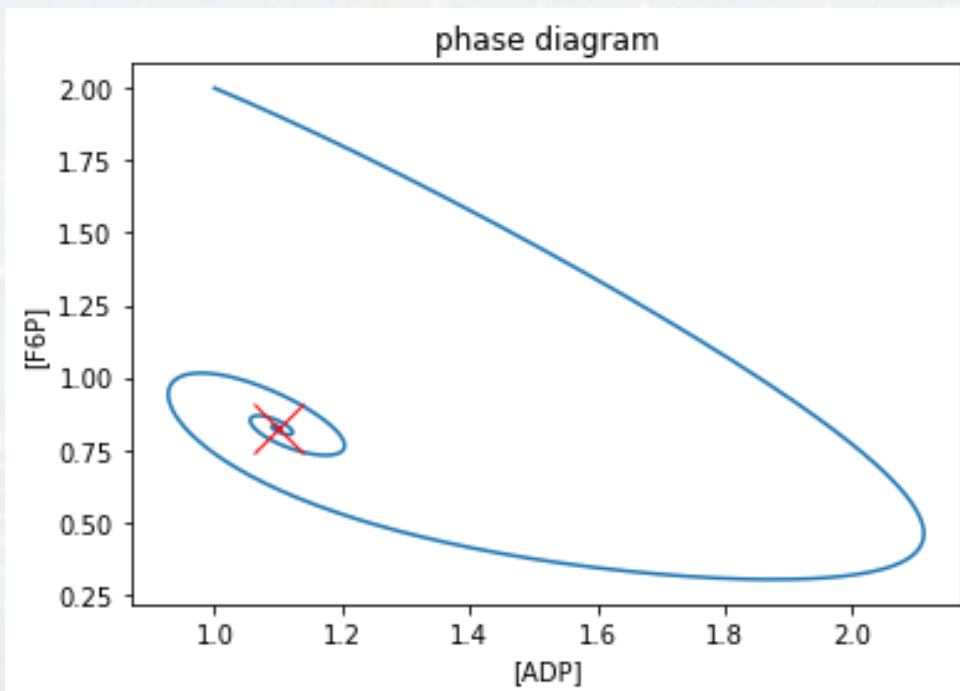
$b = 1.1$

`SolveGlycolysis([1, 2], [0, 50], a, b)`

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python





run:

SolvePhageTherapy(Init, t_span, rates)

Synergistic elimination of bacteria by phage and the immune system

Chung Yin (Joey) Leung* and Joshua S. Weitz†
School of Biology, Georgia Institute of Technology, Atlanta, Georgia 30332, USA and
School of Physics, Georgia Institute of Technology, Atlanta, Georgia 30332, USA

$$\begin{aligned}\dot{B} &= \overbrace{rB(1 - \frac{B}{K_C})}^{\text{Growth}} - \overbrace{\phi BP}^{\text{Lysis}} - \overbrace{\frac{\epsilon IB}{1 + B/K_D}}^{\text{Immune killing}}, \\ \dot{P} &= \overbrace{\beta \phi BP}^{\text{Replication}} - \overbrace{\omega P}^{\text{Decay}}, \\ \dot{I} &= \overbrace{\alpha I(1 - \frac{I}{K_I}) \frac{B}{B + K_N}}^{\text{Immune stimulation}}.\end{aligned}$$

B: bacteria
P: phages
I: immune cells

What is an ODE?

Solving ODEs by thinking

Solving ODEs with Python

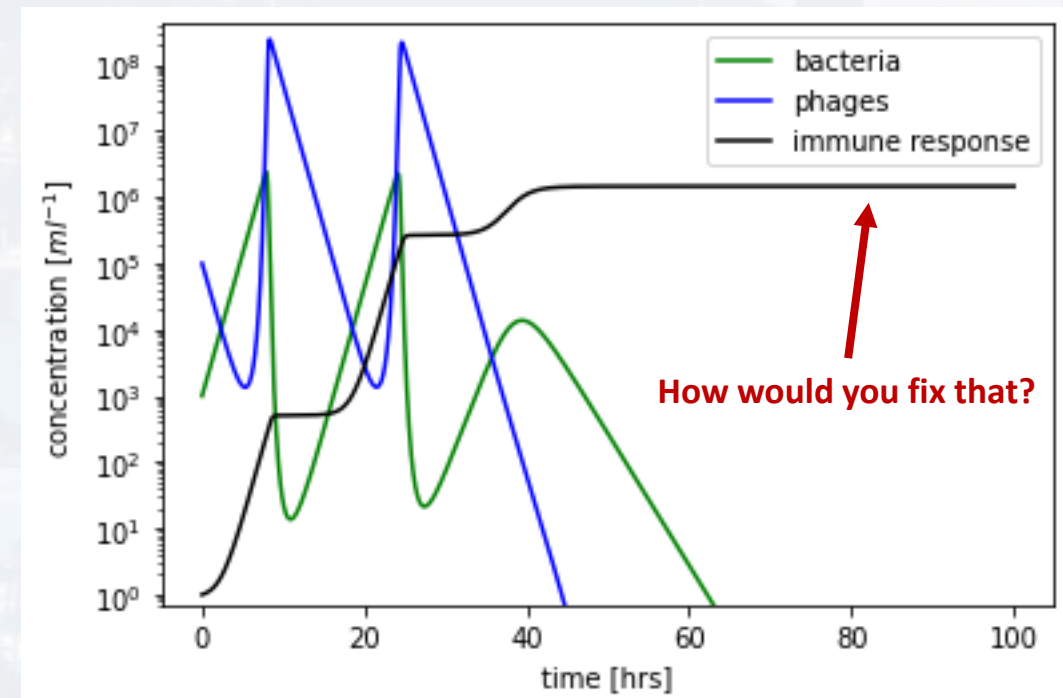
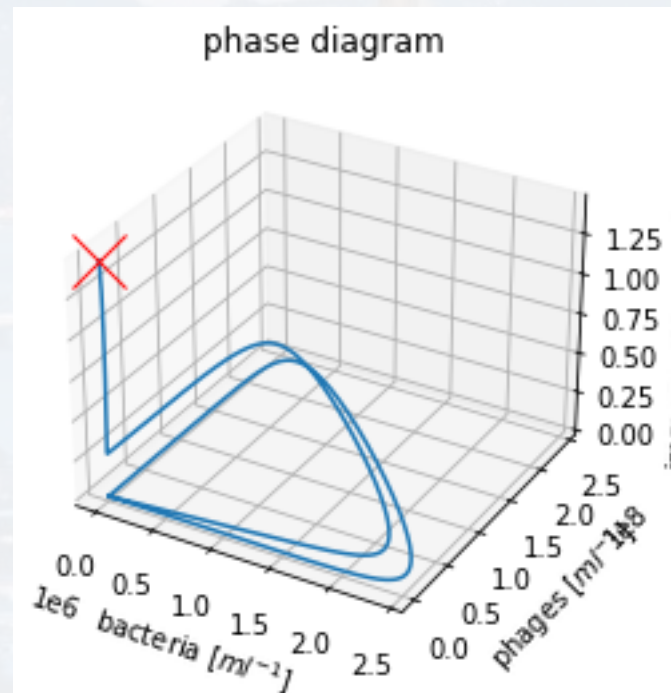


run:

```
SolvePhageTherapy(Init, t_span, rates)
```

Synergistic elimination of bacteria by phage and the immune system

Chung Yin (Joey) Leung* and Joshua S. Weitz†
School of Biology, Georgia Institute of Technology, Atlanta, Georgia 30332, USA and
School of Physics, Georgia Institute of Technology, Atlanta, Georgia 30332, USA



Thank you for your attention!

