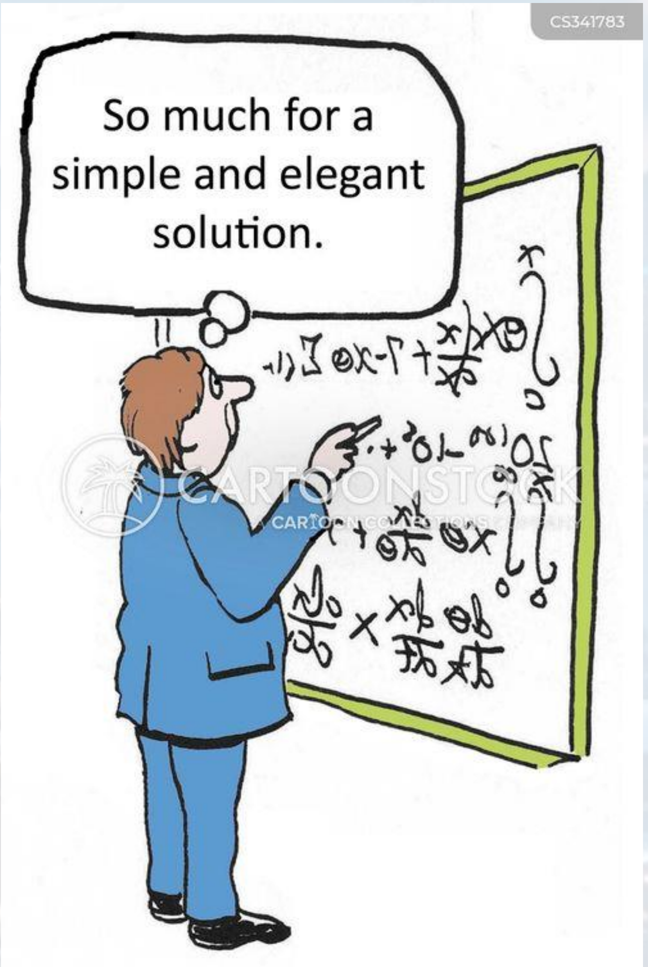
M. Hohle:

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

- of **n-th** order → *n = max(k)*

- **non-linear** → *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

let's start simple:

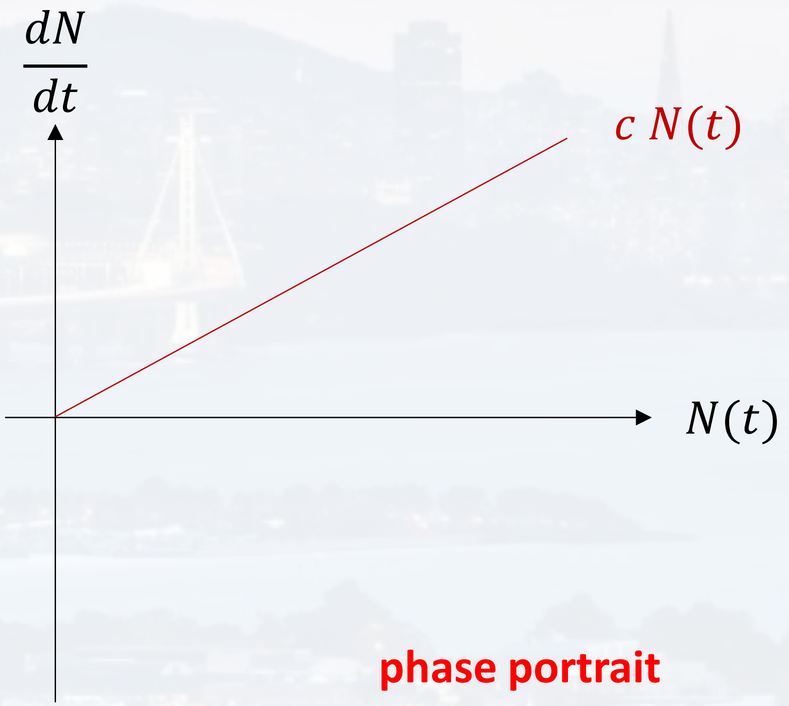
constant relative change per time step

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

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

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

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

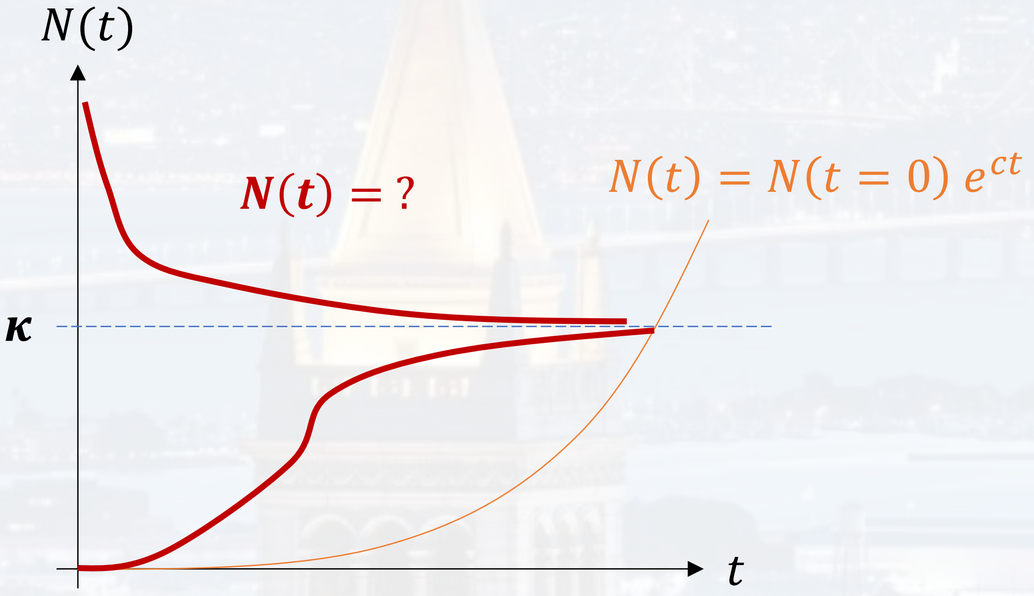


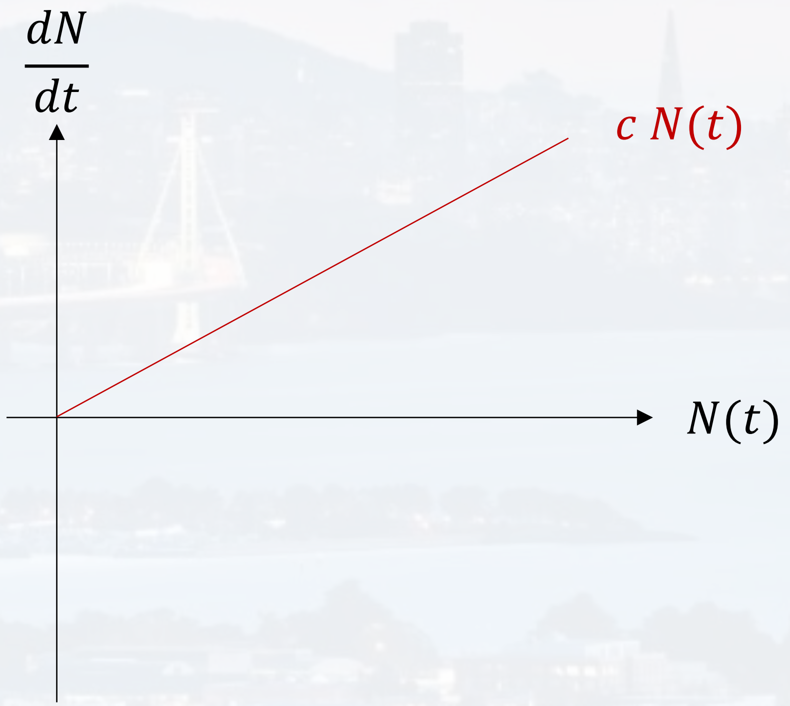
phase portrait

let's start simple:

We want the model to have a limited carrying capacity $\boldsymbol{\kappa}$

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

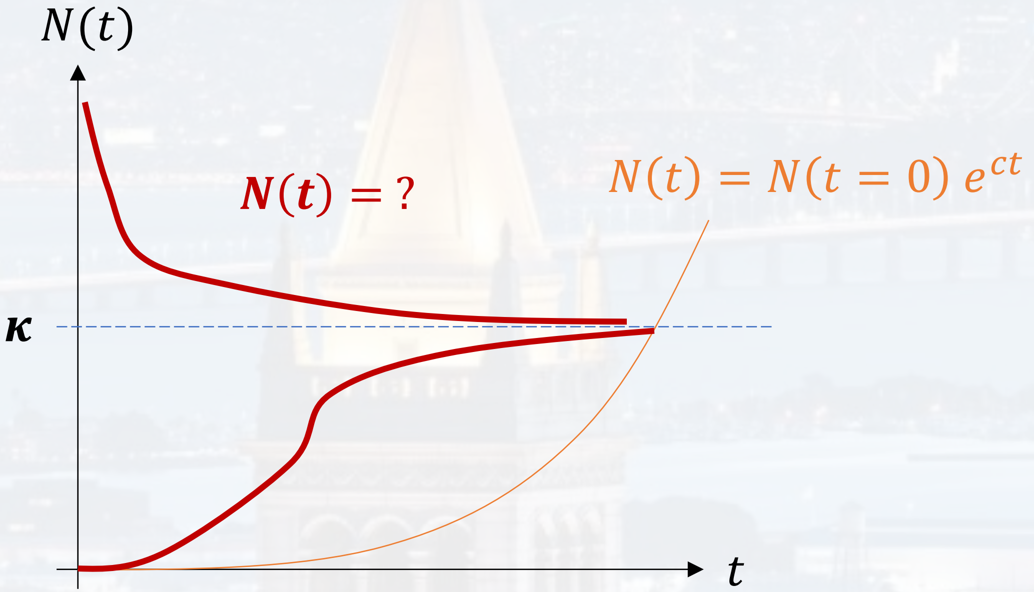


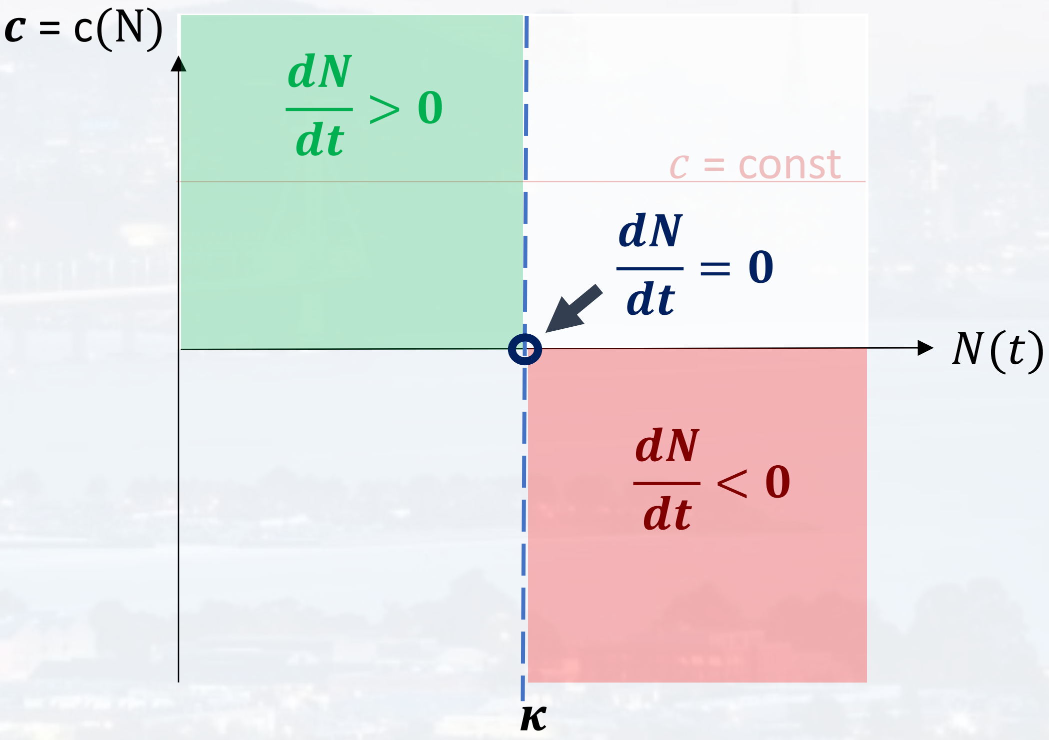


let's start simple:

We want the model to have a limited carrying capacity $\boldsymbol{\kappa}$

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

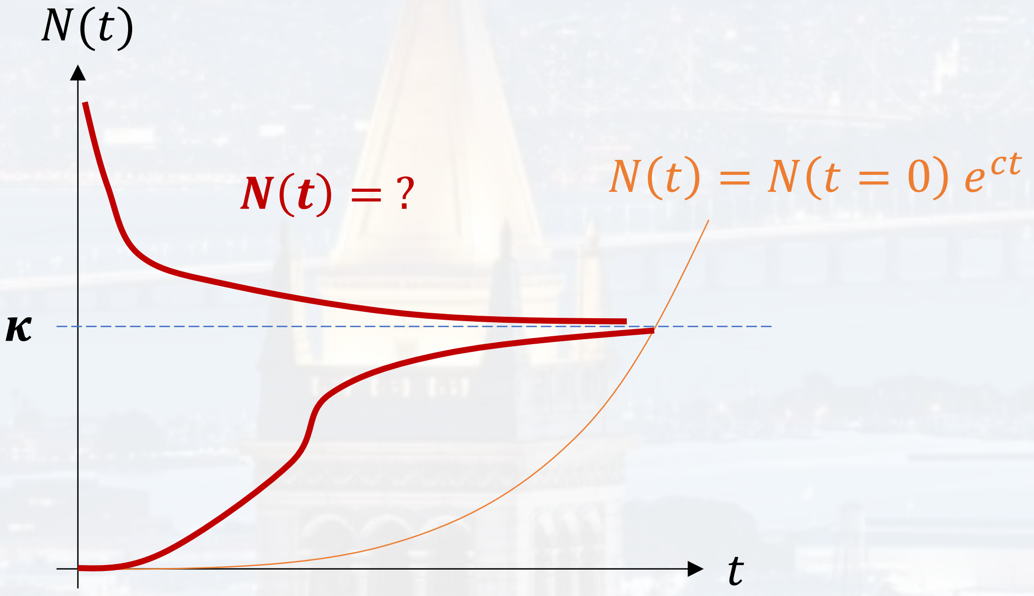


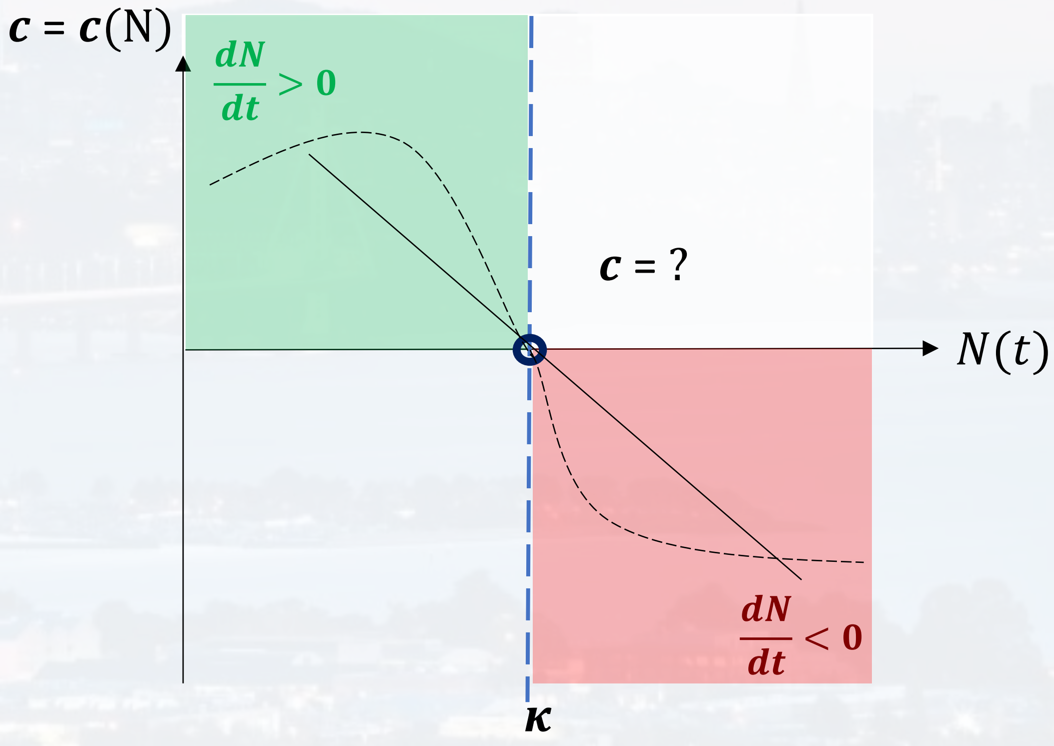


<u>let's start simple:</u>

We want the model to have a limited carrying capacity $\boldsymbol{\kappa}$

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

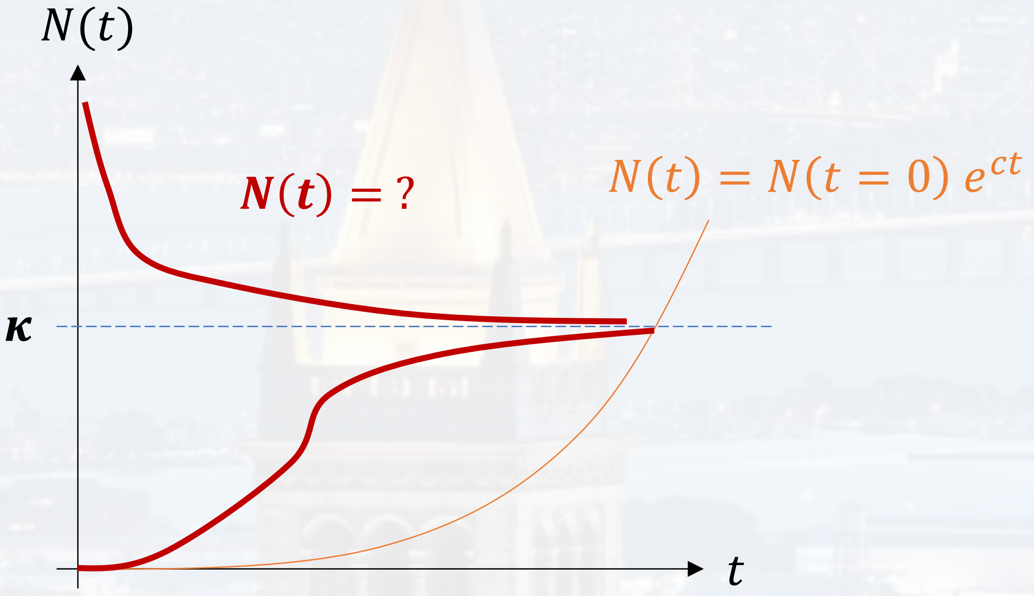




let's start simple:

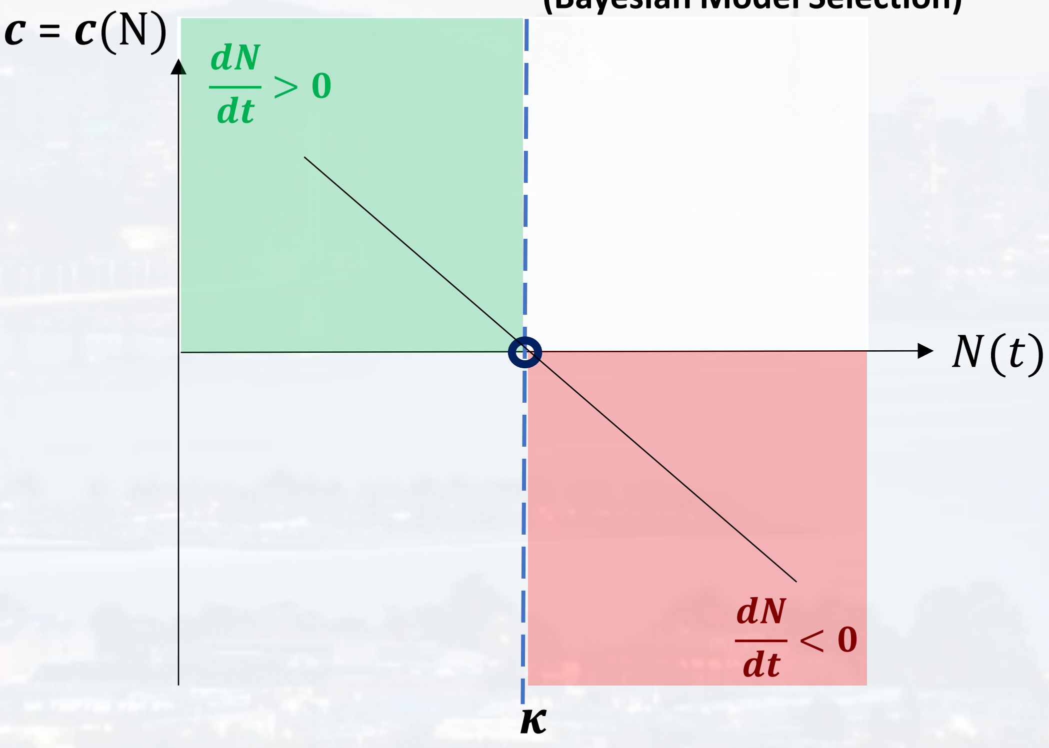
We want the model to have a limited carrying capacity $\boldsymbol{\kappa}$

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



Occam's razor:

prefer simple model
(Bayesian Model Selection)



let's start simple:

We want the model to have a limited carrying capacity $\boldsymbol{\kappa}$

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

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

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

$$c(0) = c_0$$

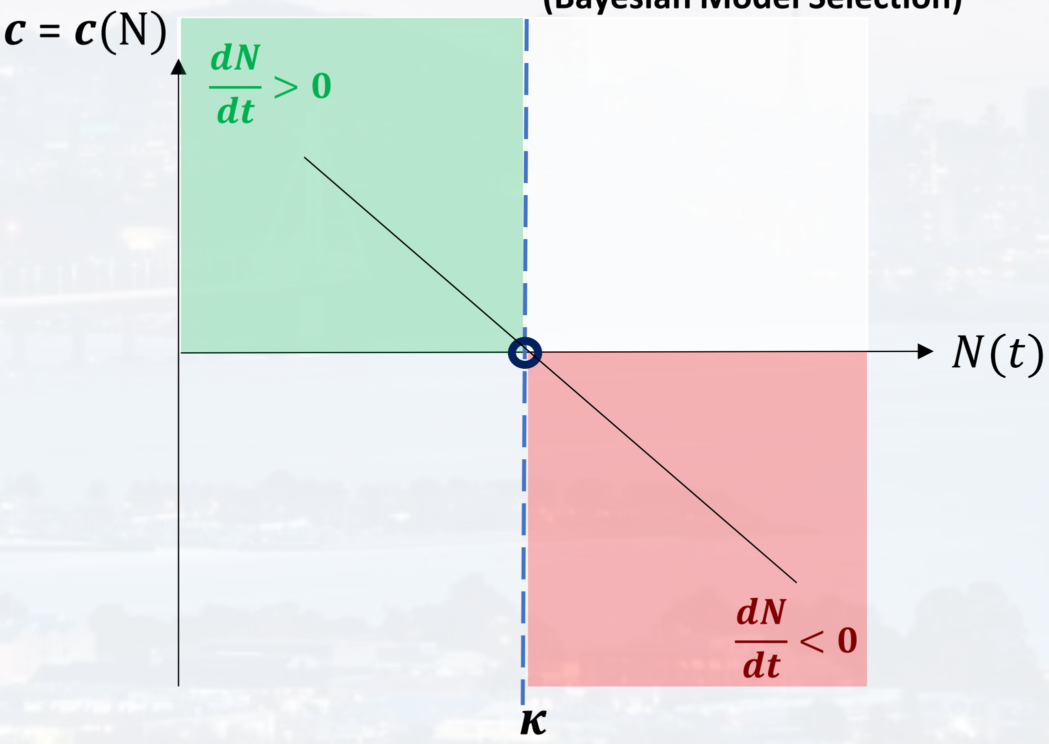
$$c(N) = c_0\left(1 - \frac{1}{\kappa}N\right)$$

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

Verhulst Equation

Occam's razor:

prefer simple model
(Bayesian Model Selection)



let's start simple:

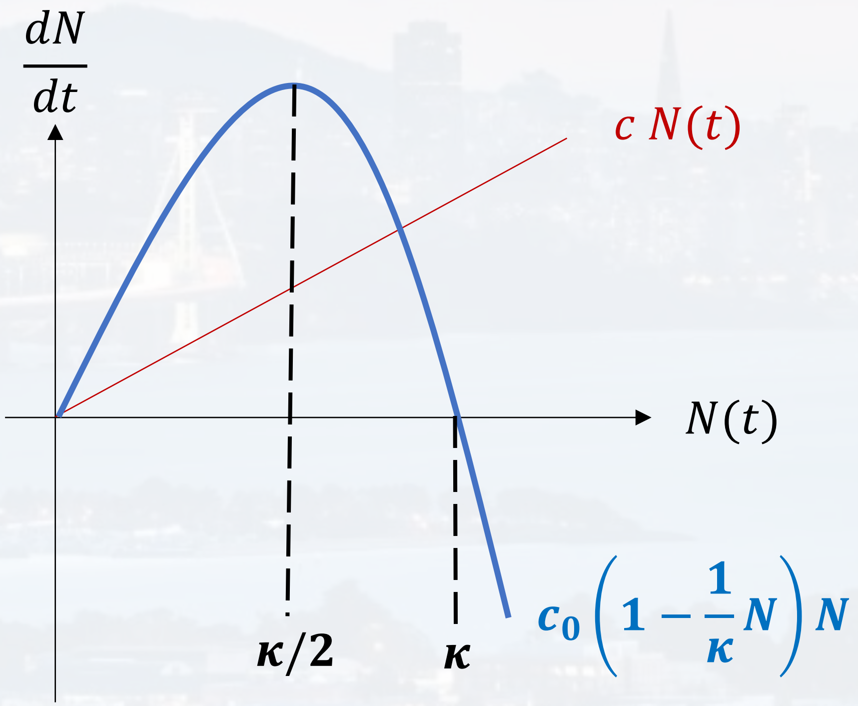
We want the model to have a limited carrying capacity κ

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

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

Verhulst Equation

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

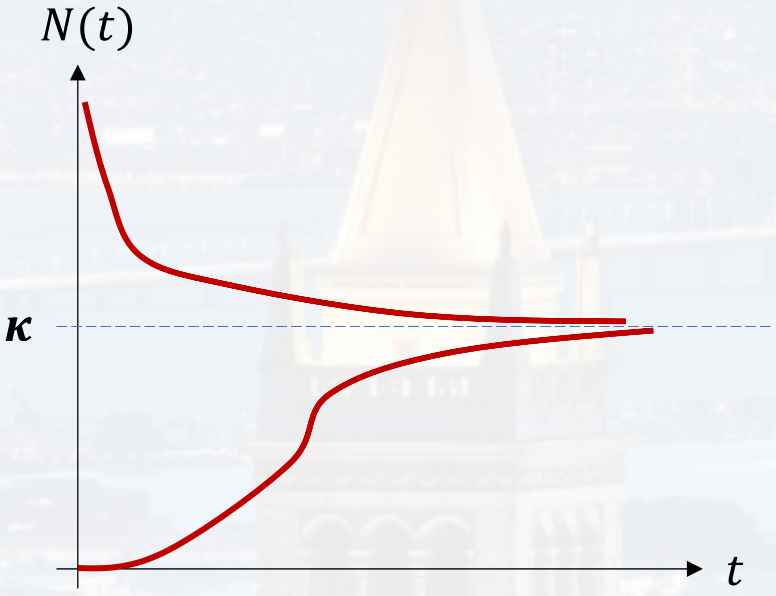


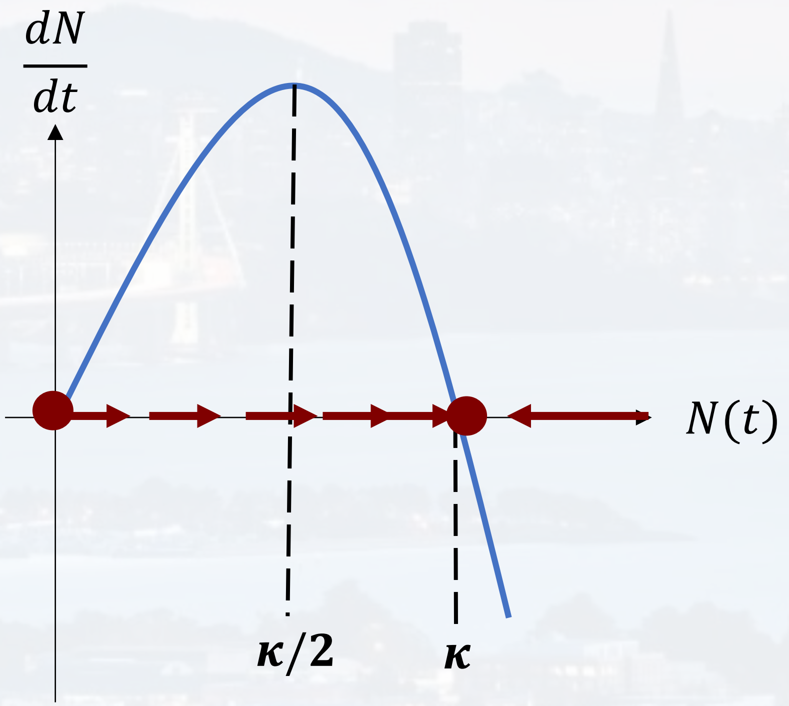
let's start simple:

We want the model to have a limited carrying capacity $\boldsymbol{\kappa}$

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

Verhulst Equation



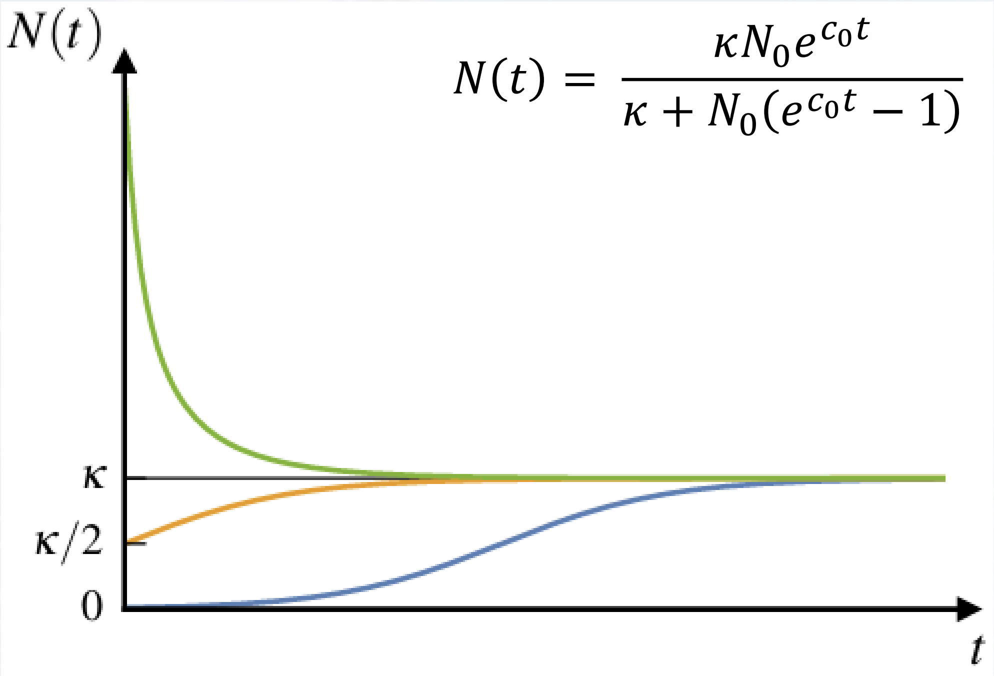


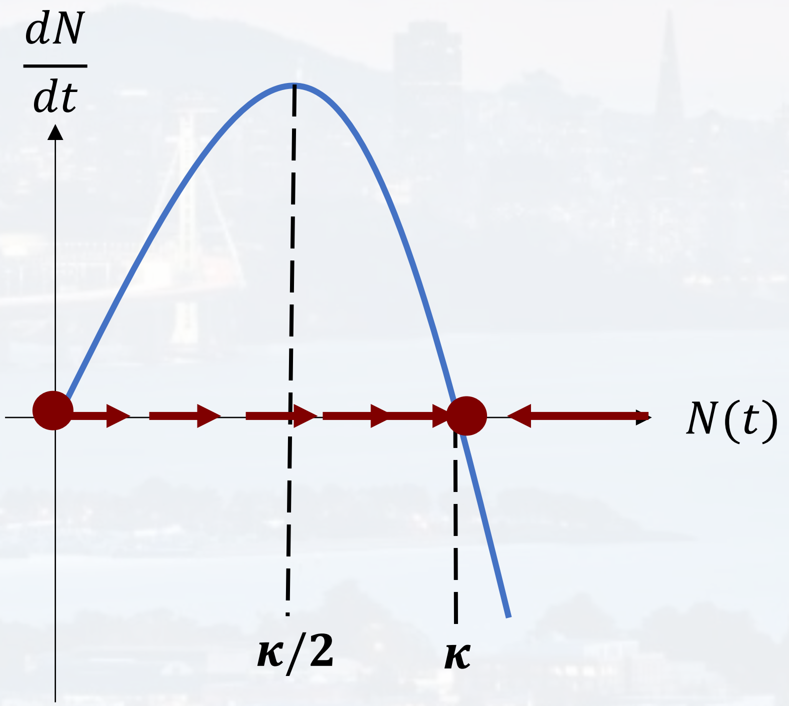
let's start simple:

We want the model to have a limited carrying capacity $\boldsymbol{\kappa}$

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

Verhulst Equation





let's start simple:

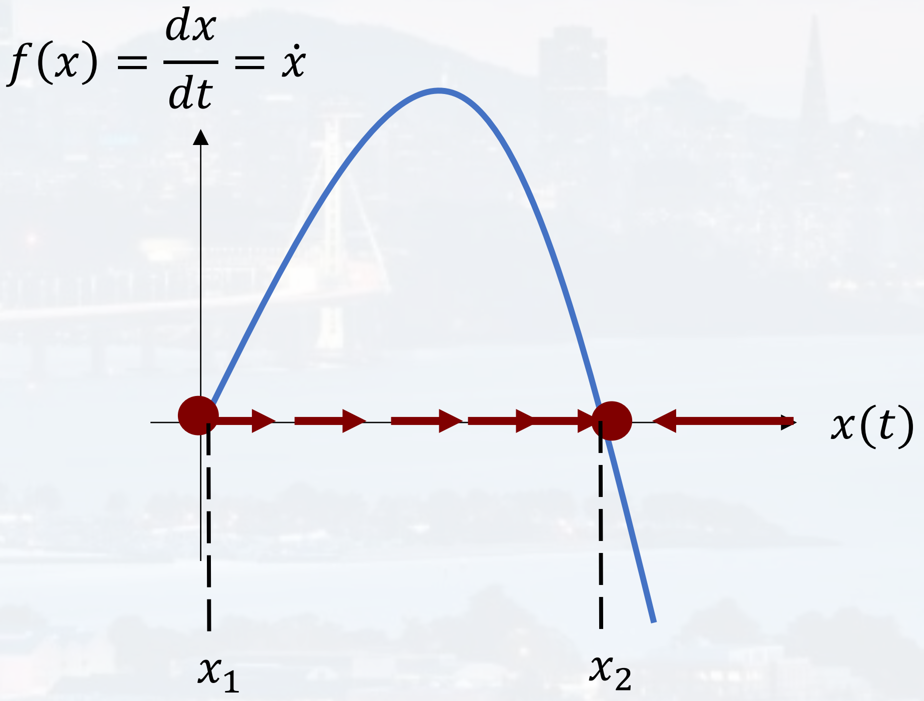
We want the model to have a limited carrying capacity $\boldsymbol{\kappa}$

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

Verhulst Equation

x_1, x_2 fixed points

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



fixed points x^*

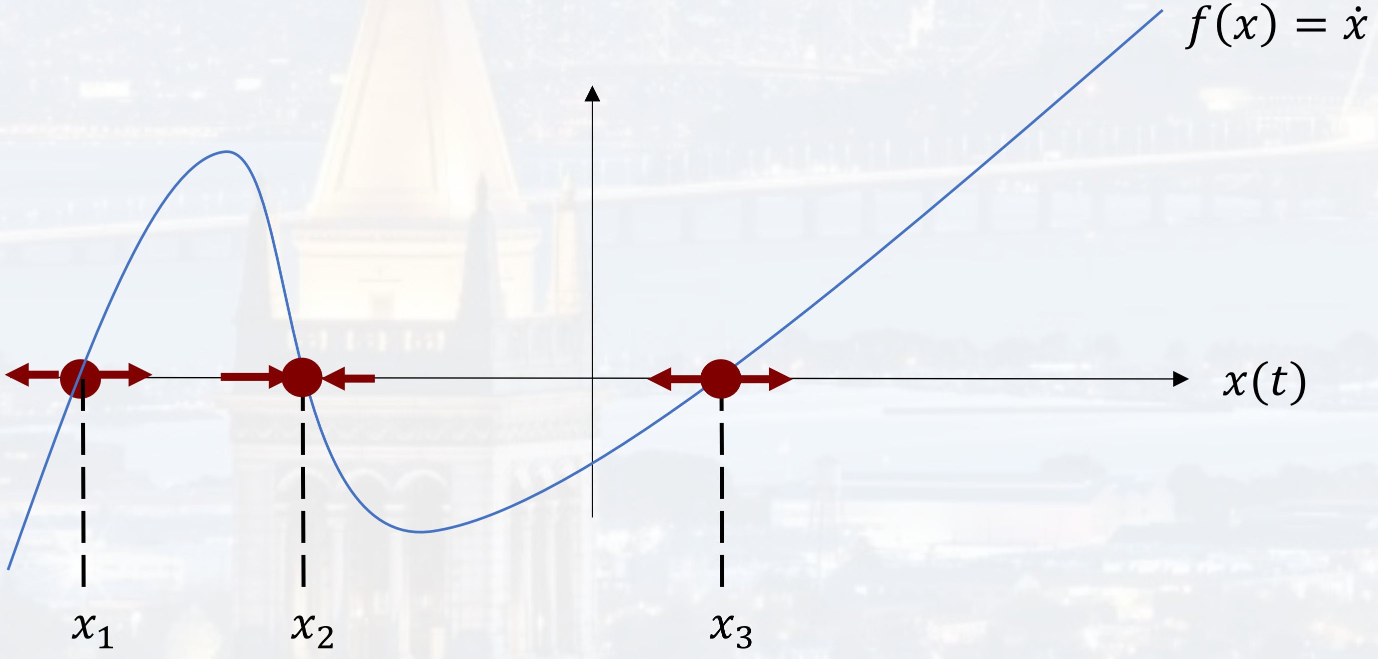
x_1: repeller → unstable

x_3: repeller → unstable

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

x_2: attractor → stable

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



fixed points x^*

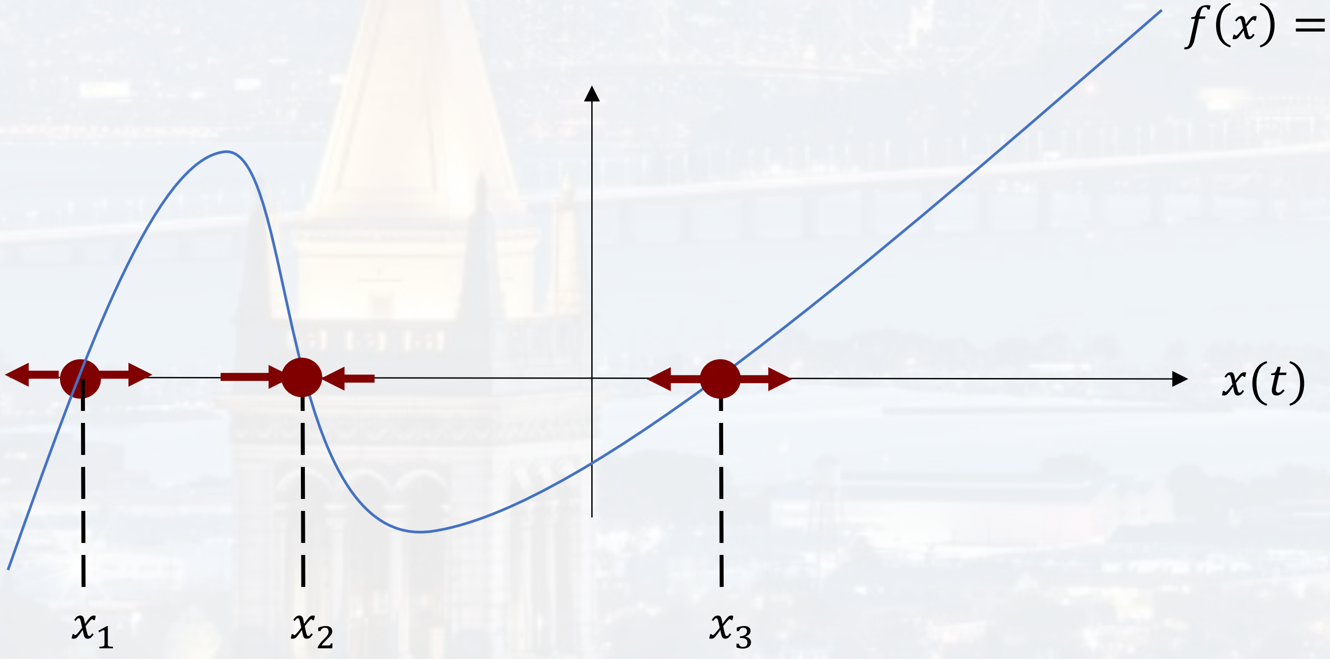
small perturbation $\varepsilon(t)$

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

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

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

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



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

$\dfrac{df(x)}{dx} > 0$ unstable

$\dfrac{df(x)}{dx} < 0$ stable

fixed points x^*

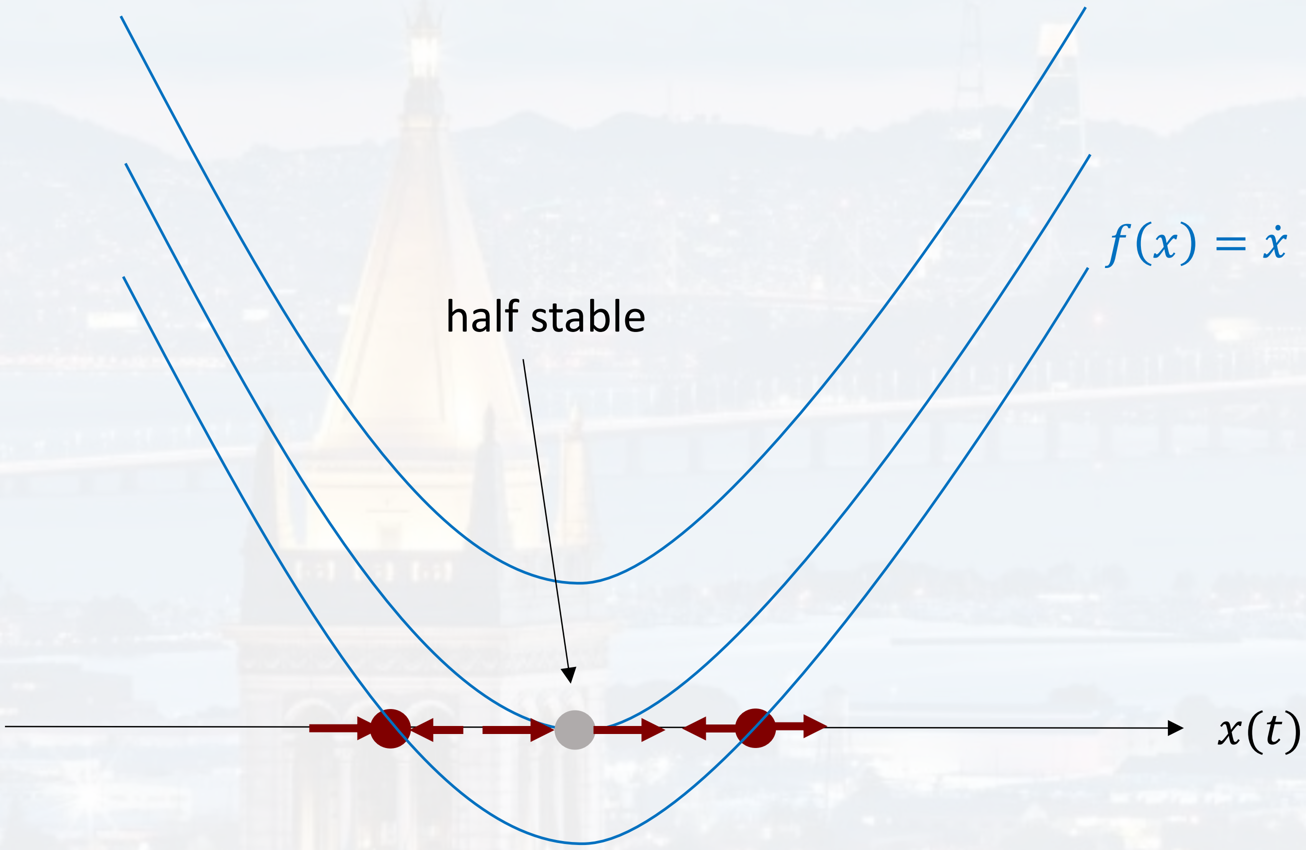
small perturbation $\varepsilon(t)$

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

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

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

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



fixed points x^*

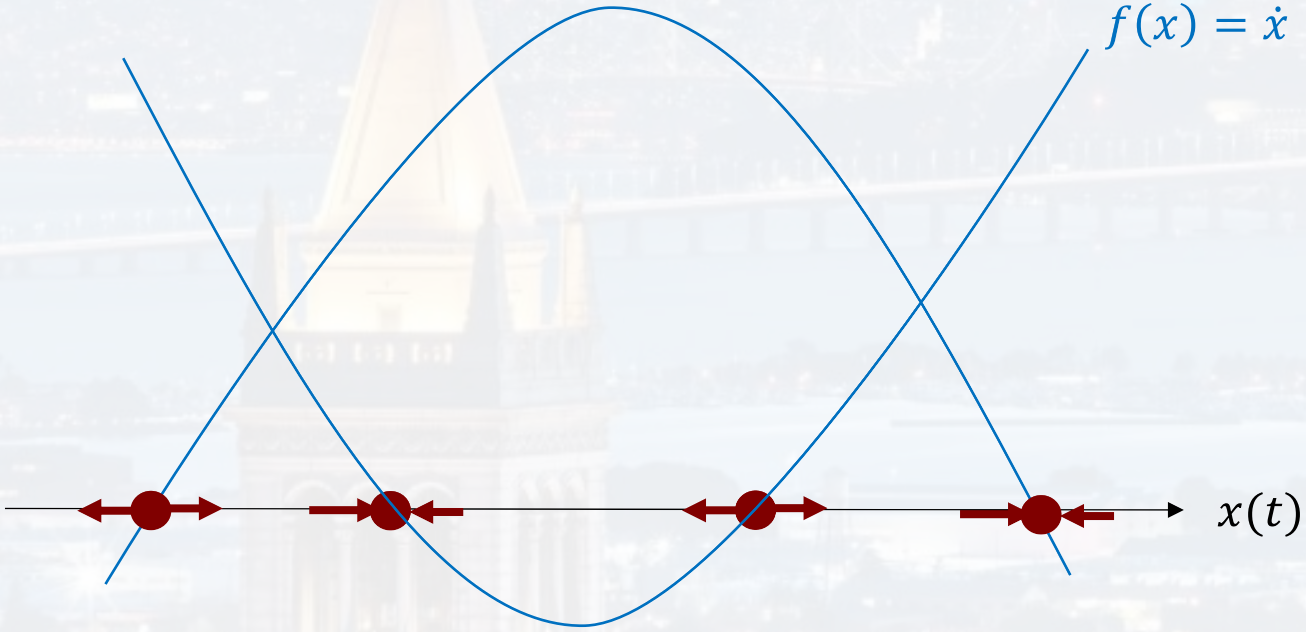
small perturbation $\varepsilon(t)$

$$\varepsilon(t) = \varepsilon_0 \, e^{\frac{df(x)}{dx}|_{x=x^*} t} \qquad \text{time scale } \tau = \frac{1}{\frac{df(x)}{dx}|_{x=x^*}}$$

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

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

$$f(x) = \boldsymbol{a}x^2 + \boldsymbol{b}x + \boldsymbol{c}$$



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

2D system

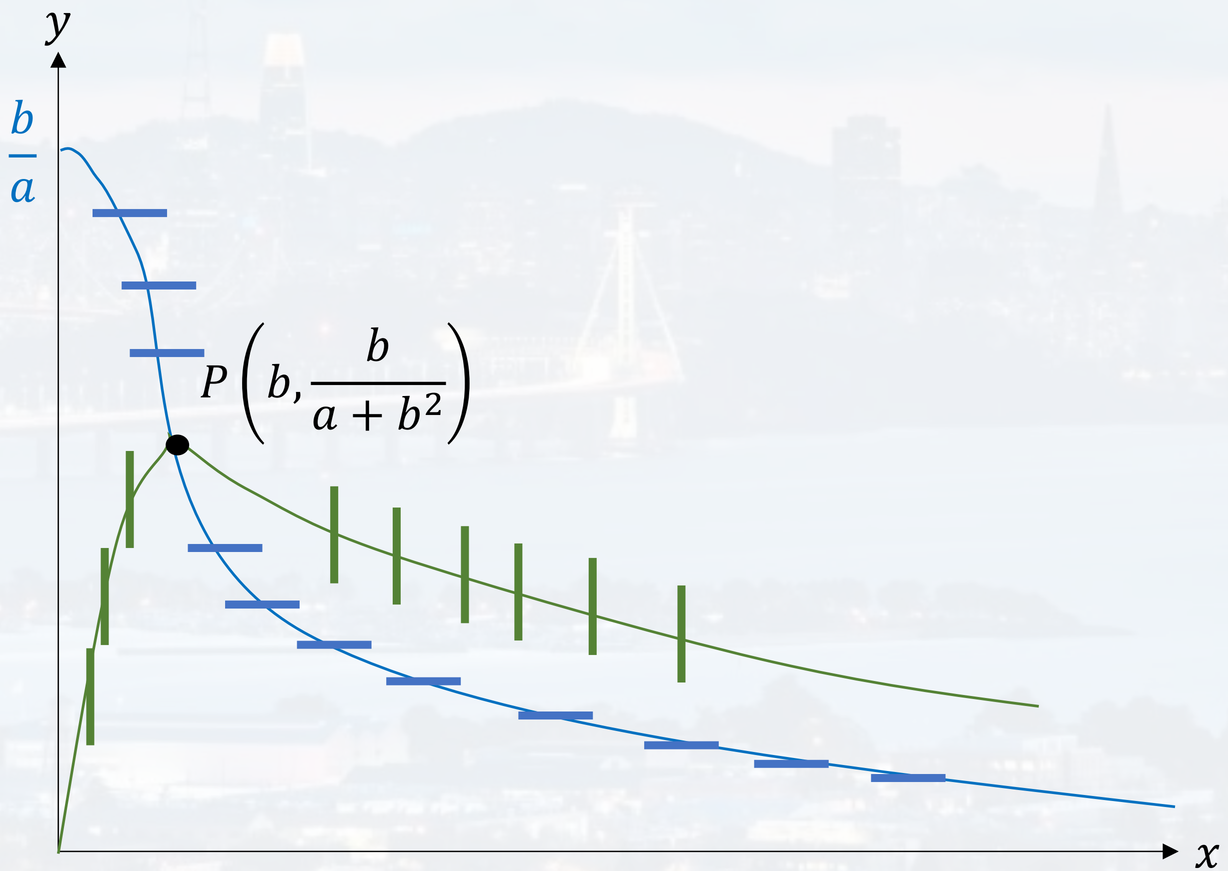
$f(x,y) = \dot{x}$ $\qquad \dot{x} = -x + a\,y + x^2 y$

$g(x,y) = \dot{y}$ $\qquad \dot{y} = b - a\,y - x^2 y$ $\qquad$ non-linear, coupled ODEs

null clines

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

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



Find out which way the system moves!

2D system

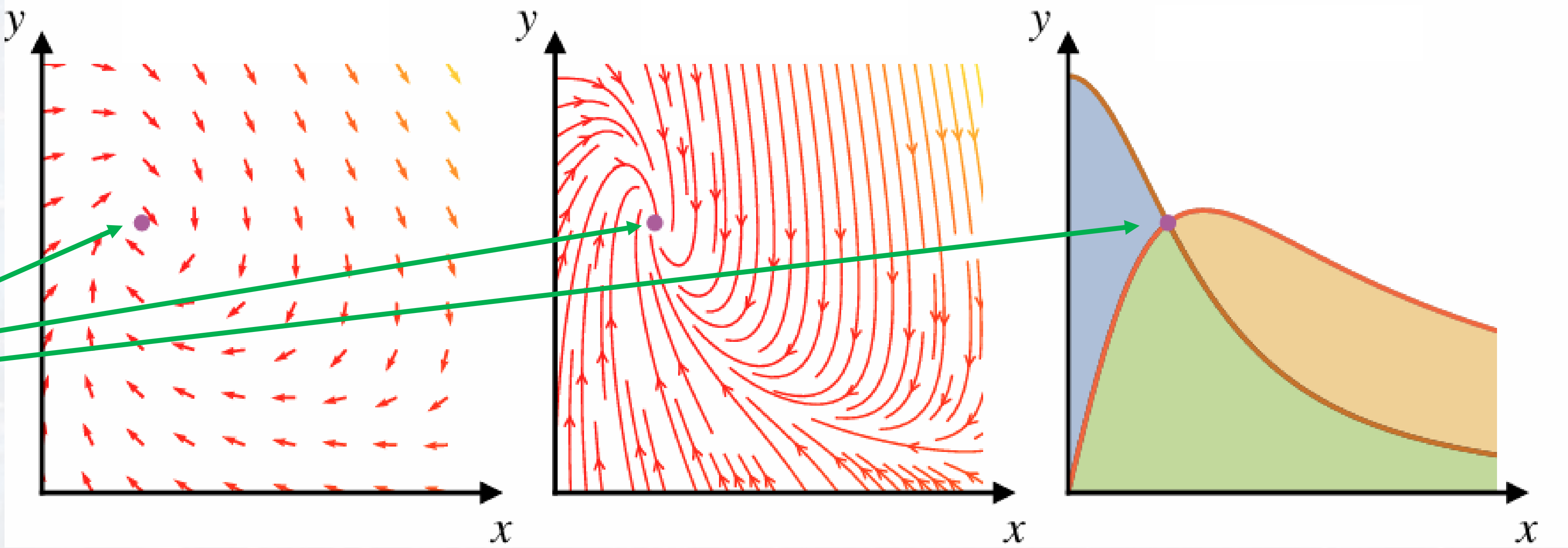
$$f(x,y) = \dot{x} \qquad \dot{x} = -x + a\,y + x^2 y$$

$$g(x,y) = \dot{y} \qquad \dot{y} = b - a\,y - x^2 y$$

non-linear, coupled ODEs

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

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



attractor or repeller?

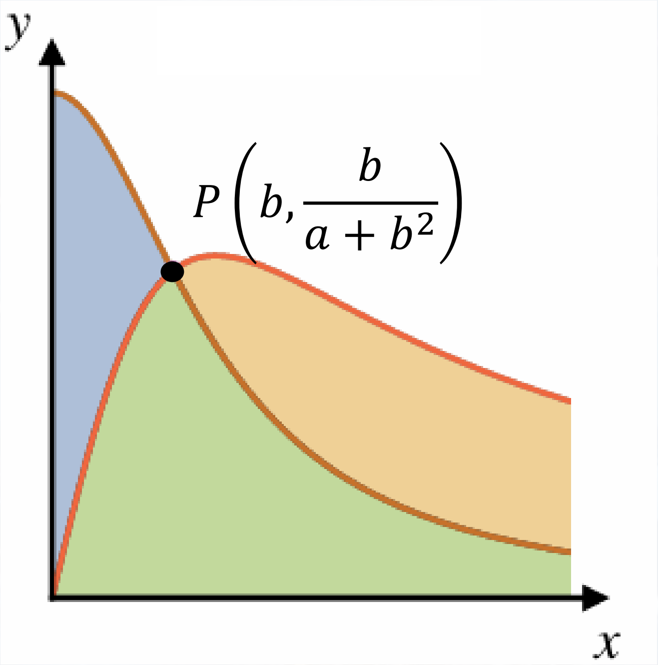
2D system

$$f(x,y) = \dot{x} \qquad \dot{x} = -x + a\,y + x^2 y$$

$$g(x,y) = \dot{y} \qquad \dot{y} = b - a\,y - x^2 y$$

non-linear, coupled ODEs

stability of P:



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

$$\underbrace{g(x^* + \varepsilon_x, y^* + \varepsilon_y)}_{\frac{d\,\varepsilon_y(t)}{dt}} \approx \underbrace{g(x^*, y^*)}_{=0} + \underbrace{\frac{\partial g(x,y)}{\partial x}\Big|_{x^*,y^*}}_{\gamma} \varepsilon_x + \underbrace{\frac{\partial g(x,y)}{\partial y}\Big|_{x^*,y^*}}_{\delta} \varepsilon_y$$

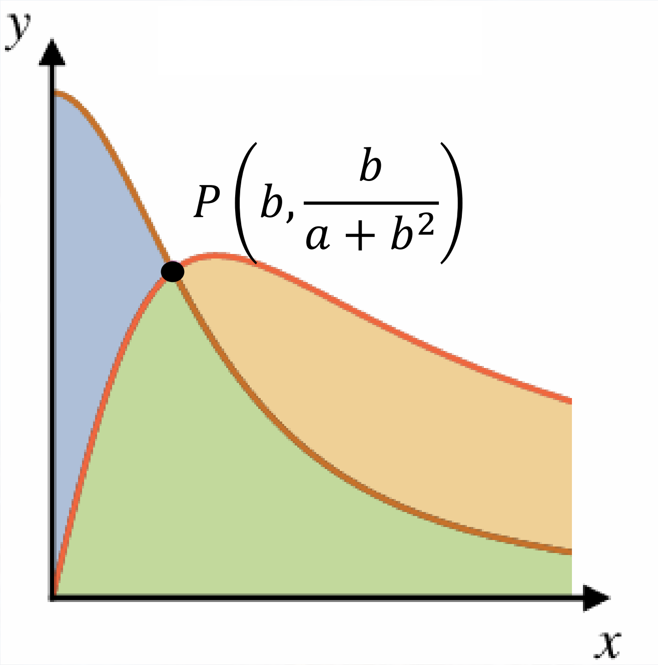
$$\dot{\vec{\varepsilon}} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \vec{\varepsilon}$$

2D system

$f(x,y) = \dot{x}$ $\quad\quad$ $\dot{x} = -x + a\,y + x^2 y$

$g(x,y) = \dot{y}$ $\quad\quad$ $\dot{y} = b - a\,y - x^2 y$ $\quad\quad$ non-linear, coupled ODEs

stability of P:



$$\dot{\vec{\varepsilon}} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \vec{\varepsilon} \qquad A = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \qquad \varepsilon(t) = \varepsilon_0\, e^{\frac{df(x)}{dx}|_{x=x^*}\, t}$$

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

λ: eigenvalue of A

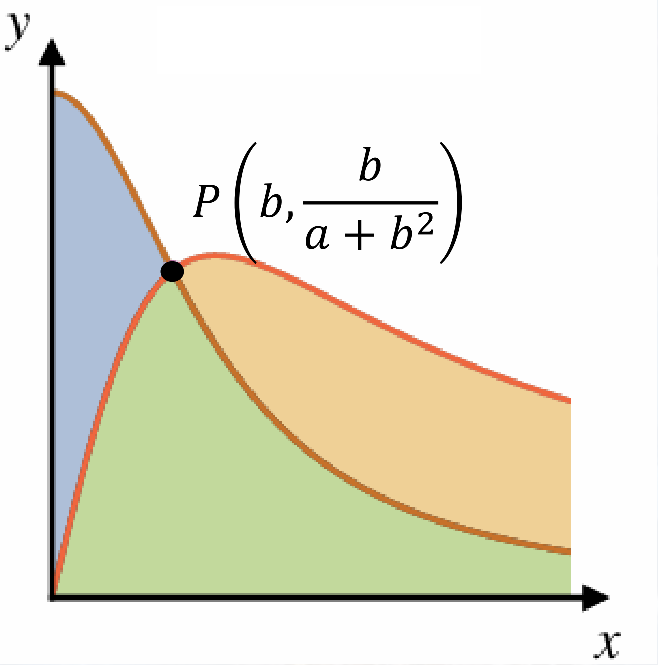
2D system

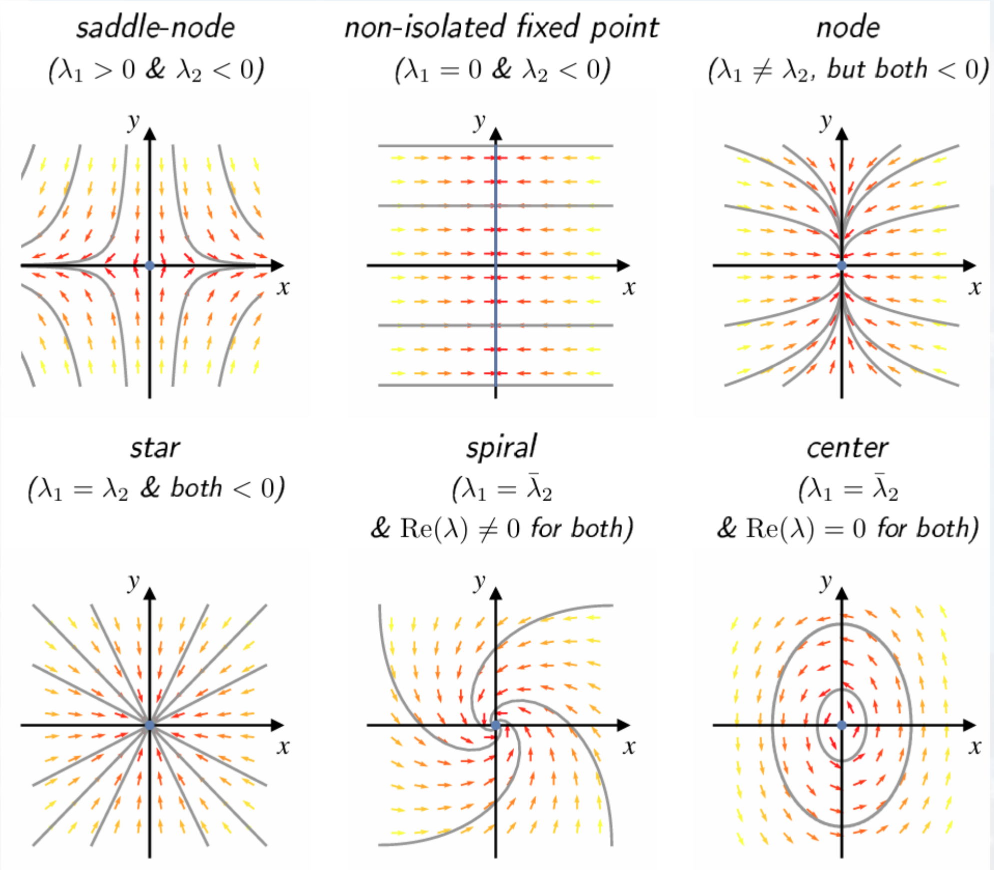
$$f(x,y) = \dot{x} \qquad \dot{x} = -x + a\,y + x^2 y$$

$$g(x,y) = \dot{y} \qquad \dot{y} = b - a\,y - x^2 y$$

non-linear, coupled ODEs

stability of P: $\vec{\varepsilon}(t) = \vec{\varepsilon}(t=0)\, e^{\lambda t}$





2D system

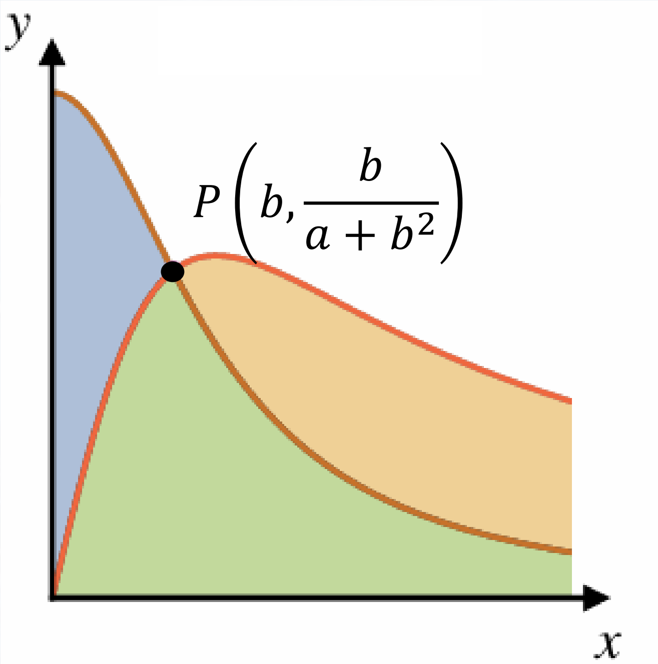
$$f(x,y) = \dot{x} \qquad \dot{x} = -x + a\,y + x^2 y$$

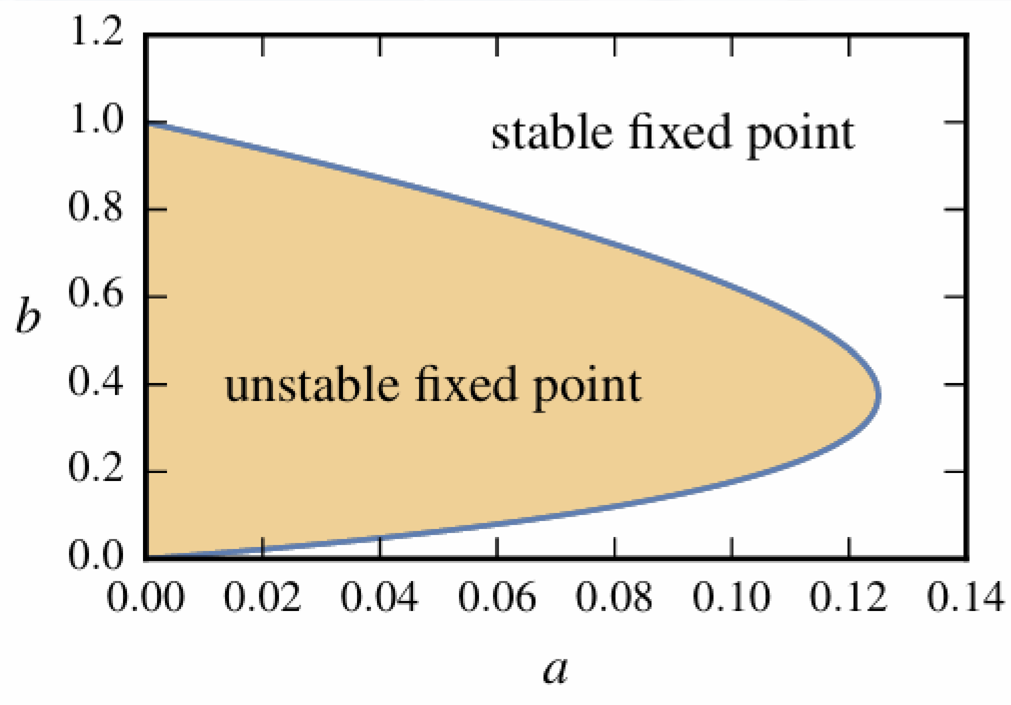
$$g(x,y) = \dot{y} \qquad \dot{y} = b - a\,y - x^2 y$$

non-linear, coupled ODEs

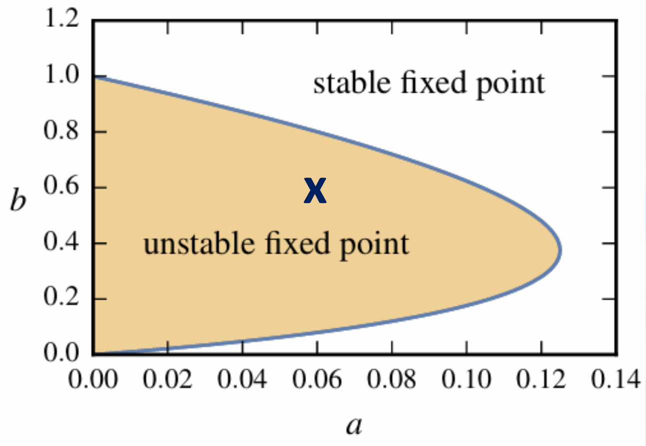
stability of P: $\vec{\varepsilon}(t) = \vec{\varepsilon}(t=0)\, e^{\lambda t}$

for this particular system:

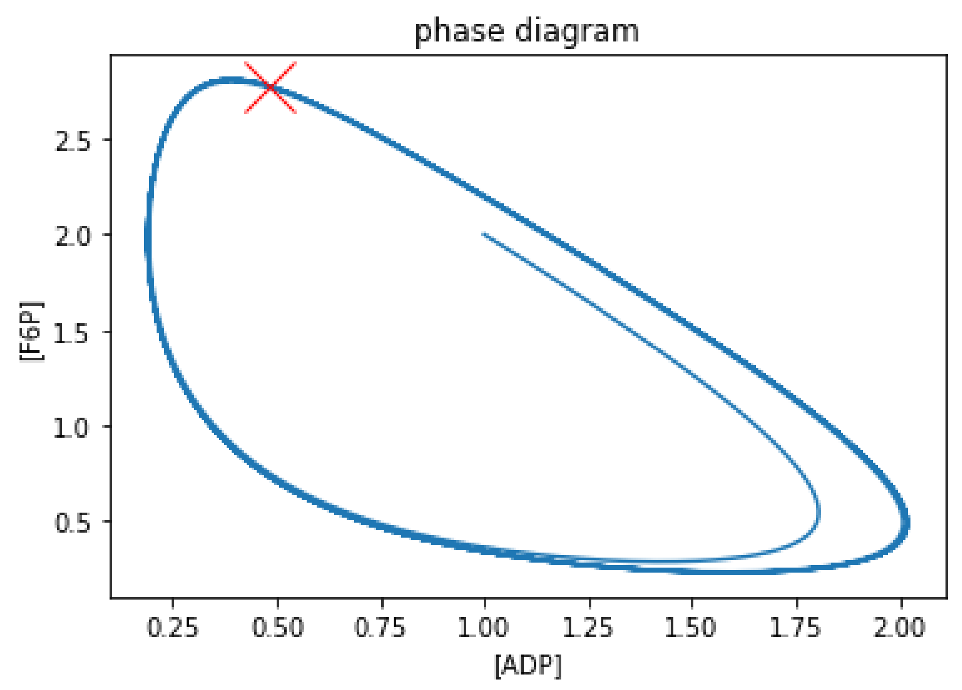


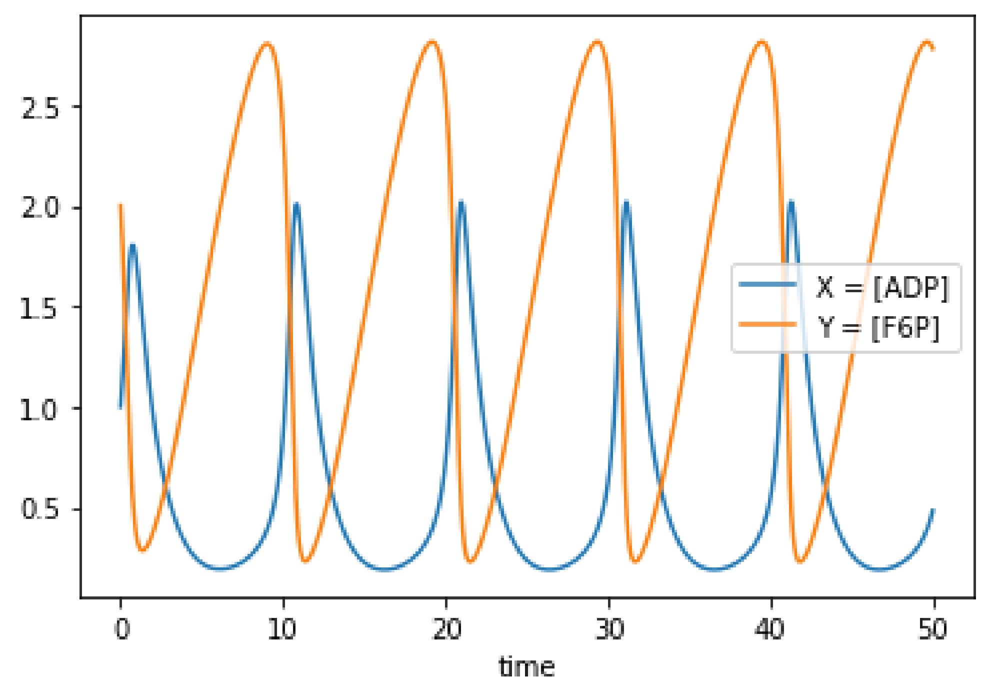


2D system

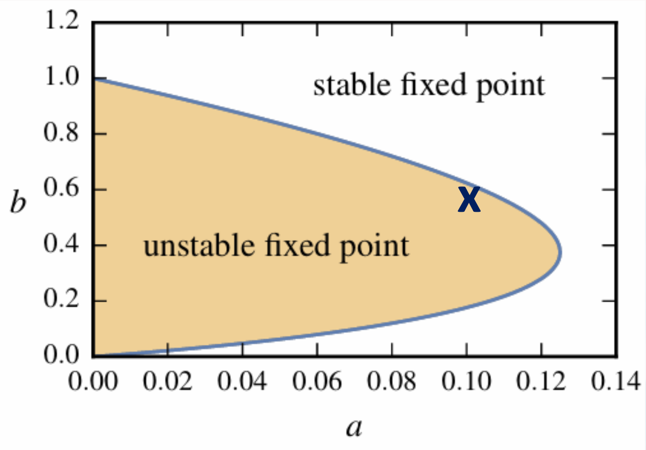


$$P\left(b, \frac{b}{a+b^2}\right) = (0.6, 1.42)$$

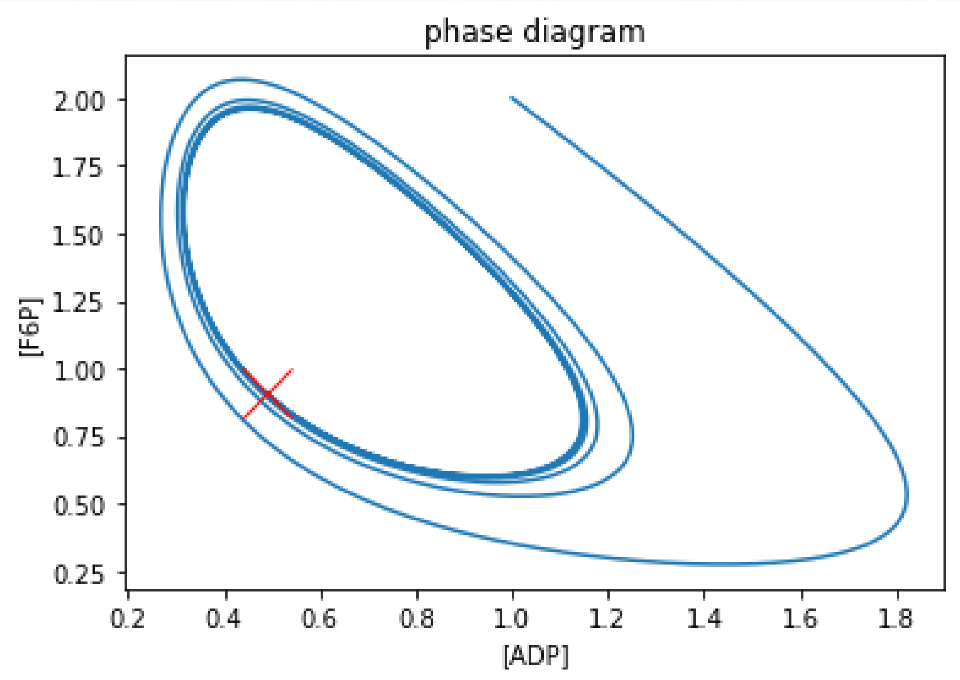


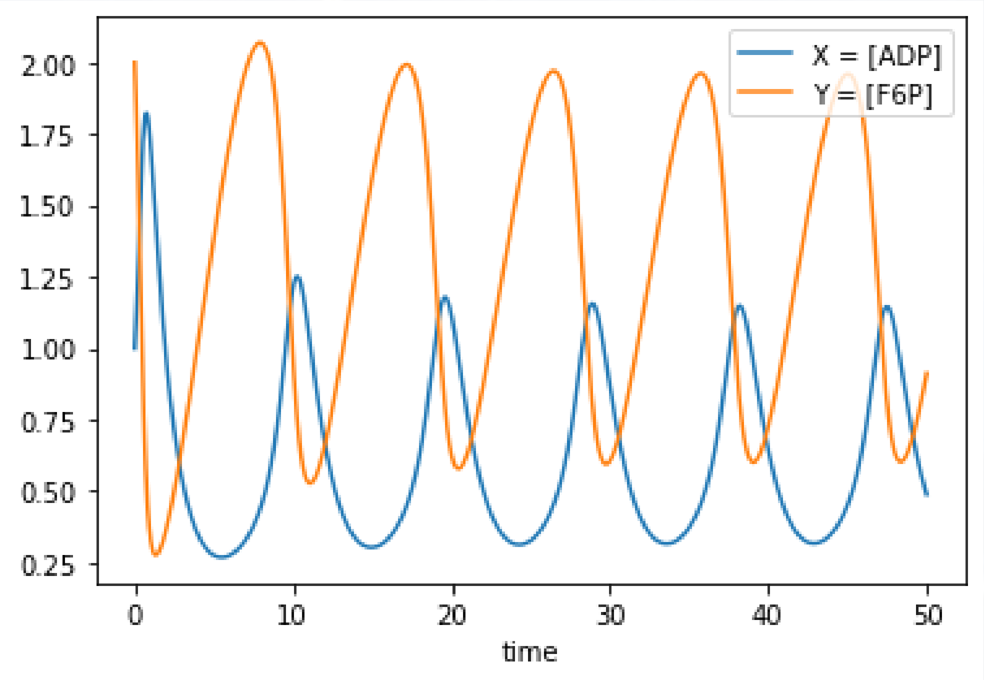


2D system

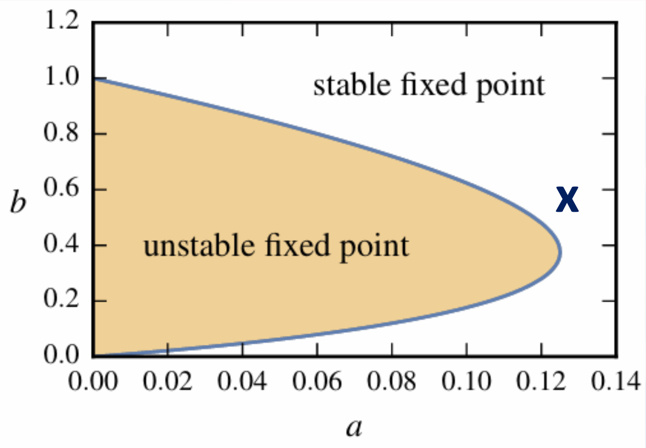


$$P\left(b, \frac{b}{a+b^2}\right) = (0.6, 1.30)$$

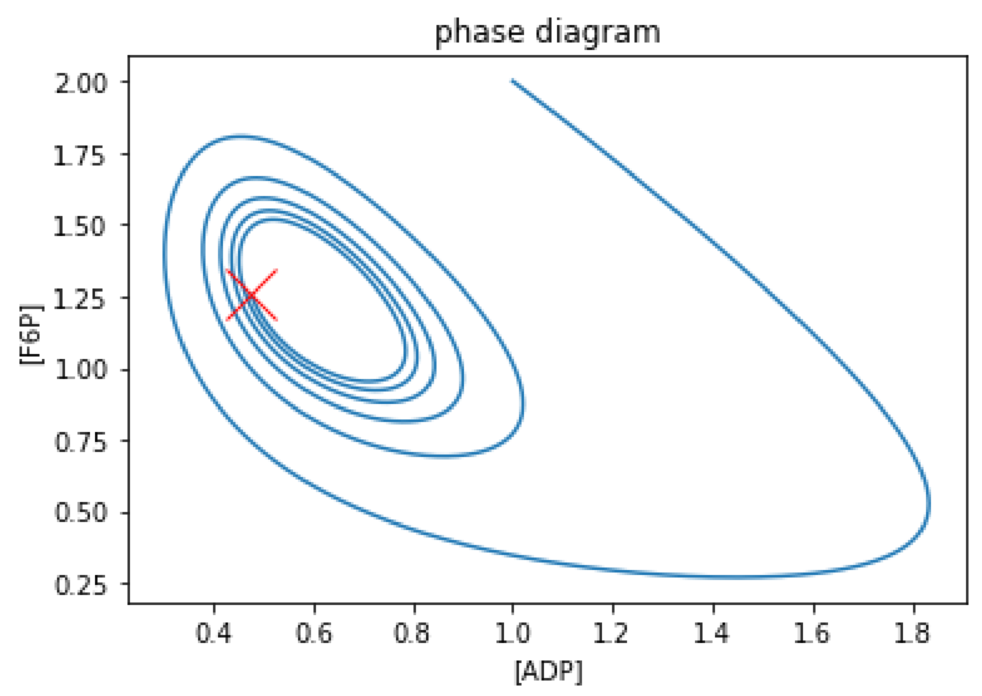


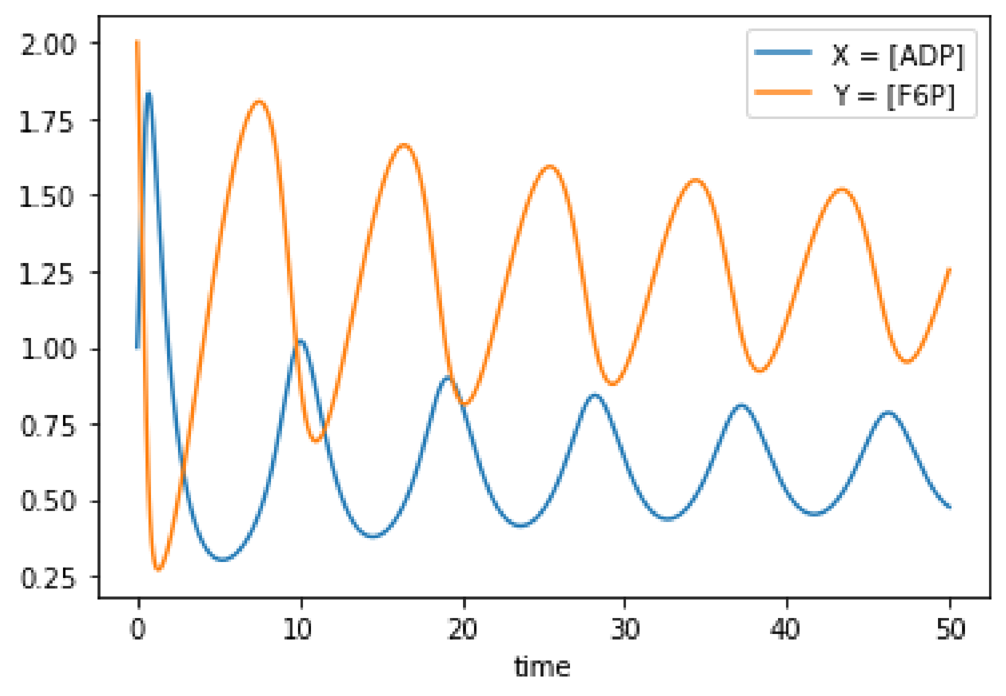


2D system

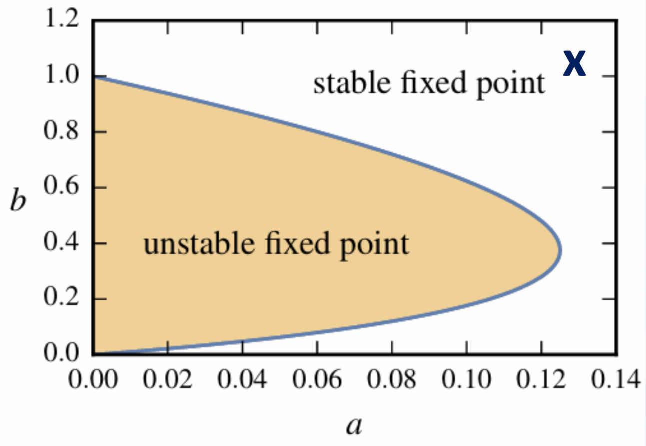


$$P\left(b, \frac{b}{a+b^2}\right) = (0.6, 1.24)$$

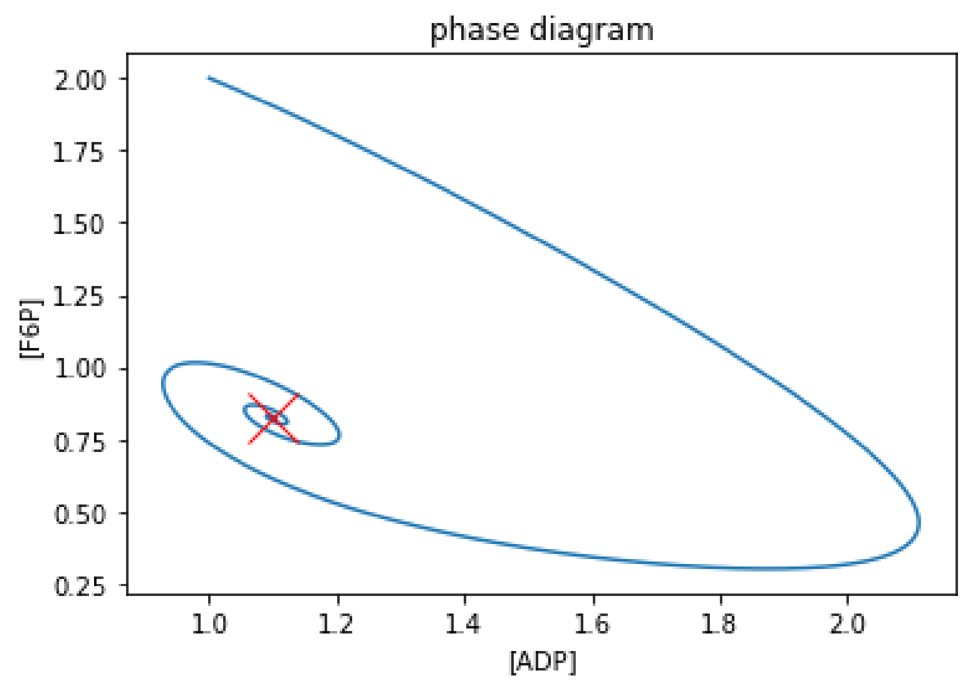


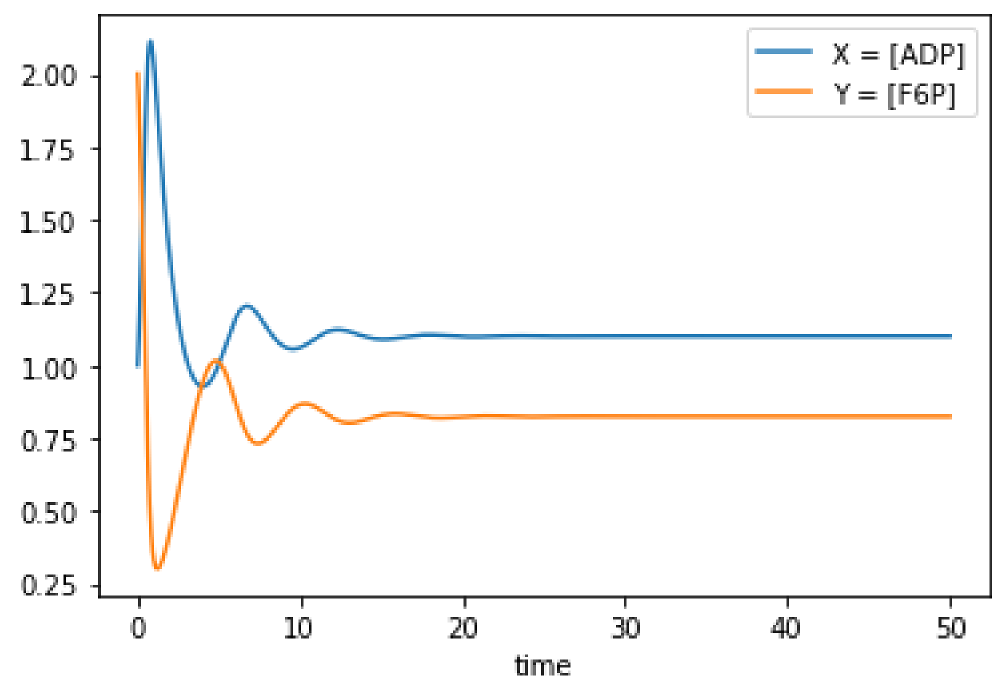


2D system



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





Runge-Kutta-Heun

$\frac{dy}{dt} = f(x(t), t)$ initial condition: $y_0 = y(t_0)$ goal: $y(t)$

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

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

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

note: more general $\frac{dy}{dx} = f(x(t), y(x(t)))$

Runge-Kutta-Heun

$$\left.\begin{aligned}\frac{dy}{dx_-} &= f(x(t), y(x(t))) \\ \frac{dy}{dx_+} &= f(x(t)+\Delta x, y + \Delta x\ y(x(t)))\end{aligned}\right\} \quad \frac{dy}{dx} = \frac{1}{2}\left[\frac{dy}{dx_-} + \frac{dy}{dx_+}\right]$$

updating:

$$x(new) = x(t) + \Delta x$$

$$y(new) = y(t) + \Delta x \frac{1}{2}\left[\frac{dy}{dx_-} + \frac{dy}{dx_+}\right] = y(t) + \Delta x \frac{1}{2}[f(x(t), y(x(t))) + f(x(t)+\Delta x, y + \Delta x\ y(x(t)))]$$

Runge-Kutta-Heun

$$x(new) = x(t) + \Delta x$$

$$y(new) = y(t) + \Delta x \frac{1}{2}\left[\frac{dy}{dx_-} + \frac{dy}{dx_+}\right] = y(t) + \Delta x \frac{1}{2}[f(x(t), y(x(t))) + f(x(t) + \Delta x, y + \Delta x\, y(x(t)))]$$

more precise (here shown for for 1D *y = y(t)*):

$$k_1 = f(t_n, y_n)$$

$$k_2 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t k_1}{2})$$

$$k_3 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t k_2}{2})$$

$$k_4 = f(t_n + \Delta t, y_n + \Delta t k_3)$$

$$t(new) = t_{n+1} = t_n + \Delta t$$

$$\mathrm{y}(new) = y_{n+1} = y_n + \frac{\Delta t}{6}\underbrace{[k_1 + 2k_2 + 2k_3 + k_4]}_{\text{recall: Simpson \& Simpson 3/8}}$$

Runge-Kutta-Heun

$$x(new) = x(t) + \Delta x$$

$$y(new) = y(t) + \Delta x \frac{1}{2}\left[\frac{dy}{dx_-} + \frac{dy}{dx_+}\right] = y(t) + \Delta x \frac{1}{2}[f(x(t), y(x(t))) + f(x(t) + \Delta x, y + \Delta x\, y(x(t)))]$$

```
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

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

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

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

[t_start, t_end]

parameter of the function

contains the actual ODEs

```
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###########################

    #...
```

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

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

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

    XY     = ode_solver(Glycolysis, Init, t_span, method = 'RK45',\
                        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

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

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

the actual solver

```
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################################

    #...
```

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

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

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

    XY      = ode_solver(Glycolysis, Init, t_span, method = 'RK45',\
                         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),\
                           t_span = t_span, y0 = Init, method = method,\
                           rtol = 1e-9, atol = 1e-9, max_step = 0.01)


        return result
```

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

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

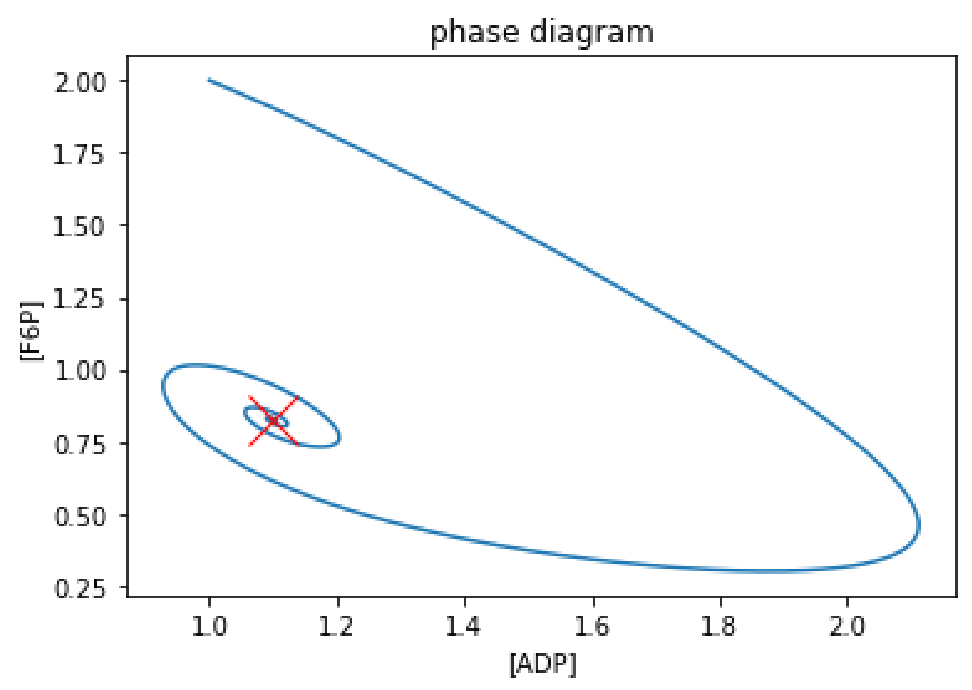
```
def SolveGlycolysis(Init, t_span, a, b):

    XY      = ode_solver(Glycolysis, Init, t_span, method = 'RK45',\
                         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),\
                       t_span = t_span, y0 = Init, method = method,\
                       rtol = 1e-9, atol = 1e-9, max_step = 0.01)

    return result
```

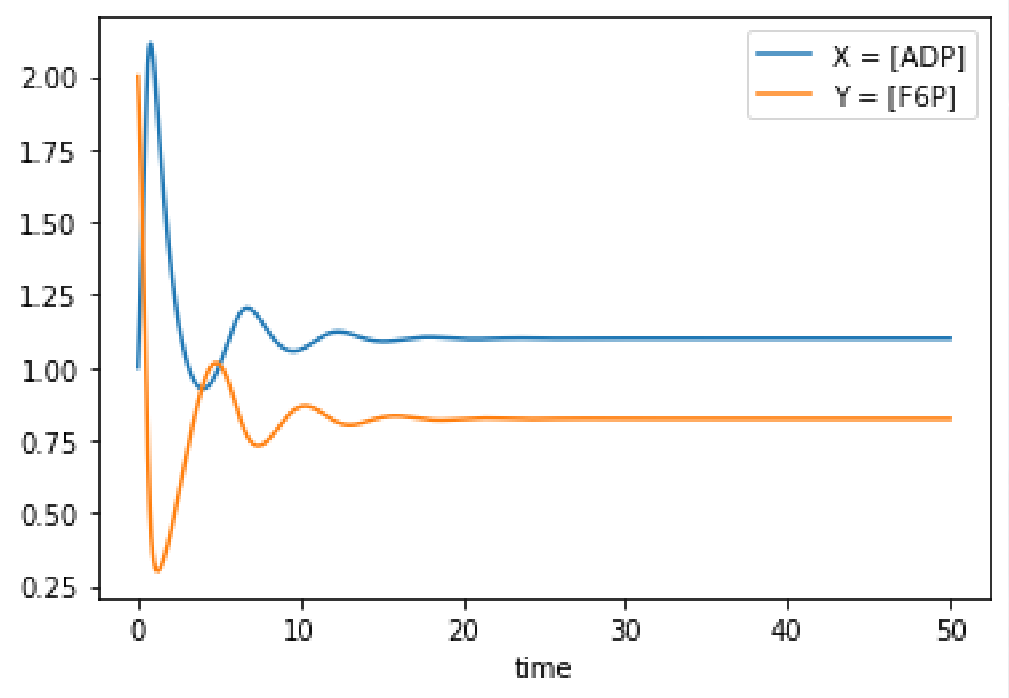
$$\dot{x} = -x + a\,y + x^2 y$$

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

<u>run:</u>

```
a = 0.125
b = 1.1
SolveGlycolysis([1, 2], [0, 50], a, b)
```





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

$$\dot{B} = \boxed{\overbrace{rB(1-\frac{B}{K_C})}^{\text{Growth}}} - \overbrace{\phi BP}^{\text{Lysis}} - \overbrace{\frac{\epsilon IB}{1+B/K_D}}^{\text{Immune killing}},$$

$$\dot{P} = \overbrace{\beta\phi BP}^{\text{Replication}} - \overbrace{\omega P}^{\text{Decay}},$$

$$\dot{I} = \overbrace{\boxed{\alpha I(1-\frac{I}{K_I})}\frac{B}{B+K_N}}^{\text{Immune stimulation}}.$$

B:	bacteria
P:	phages
I:	immune cells

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

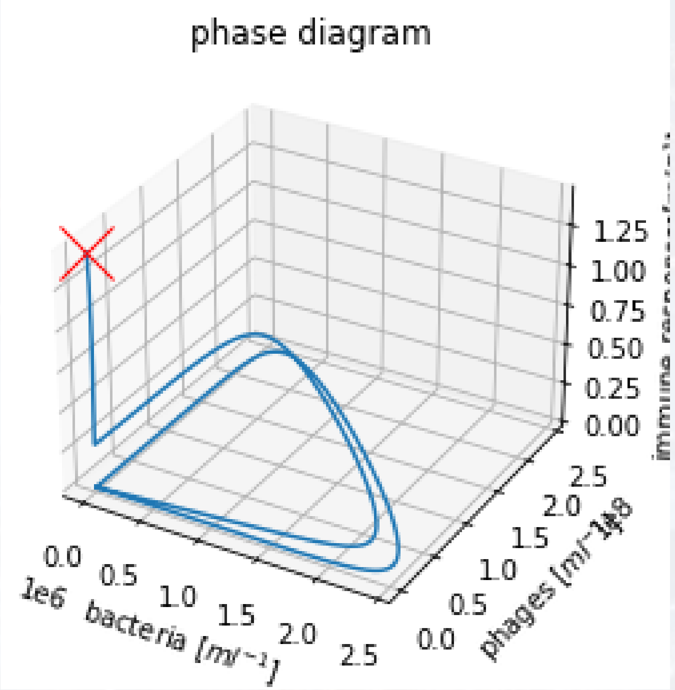
recall:
Verhulst Equation

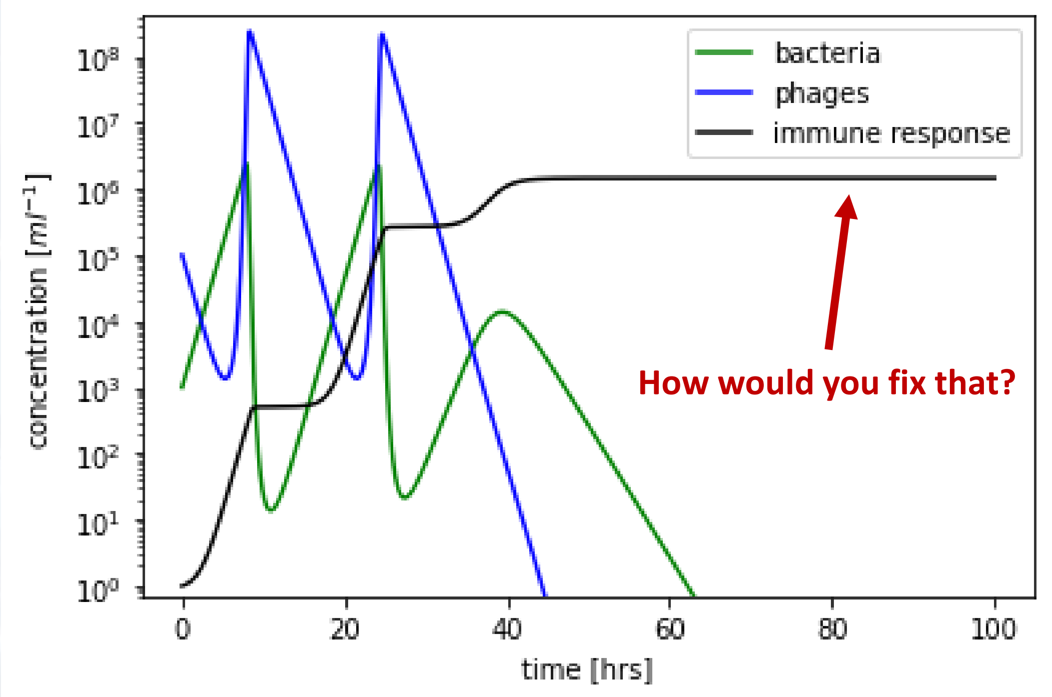
run:

```
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!