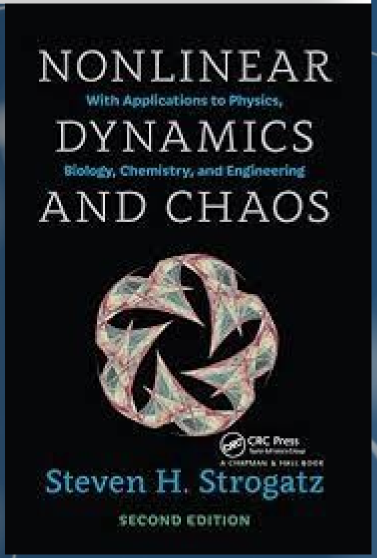
Module 3 (Solving Differential Equations: numerical techniques)



Module -3 (prerequisite)

- Module 1,2
- Ordinary Differential Equations
- Newtonian Mechanics

Module -3 (Learning outcome)

- Learning of linear and Non-linear Differential Equations
- Love affairs, Predator-Prey interactions and SIR model
- Numerical Methods-Euler, Runge-Kutta 4
- Double Pendulum, Lorenz Oscillator

  Chaos theory : A brief and general idea (sensitivity to IC)
- Fractal

Logistic equation

The logistic equation, also known as Verhulst equation, is a formula for approximating the evolution of an animal population over time.

$$x_{n+1} = r x_n (1 - x_n)$$
$$x_{n+1} = r x_n - r {x_n}^2$$

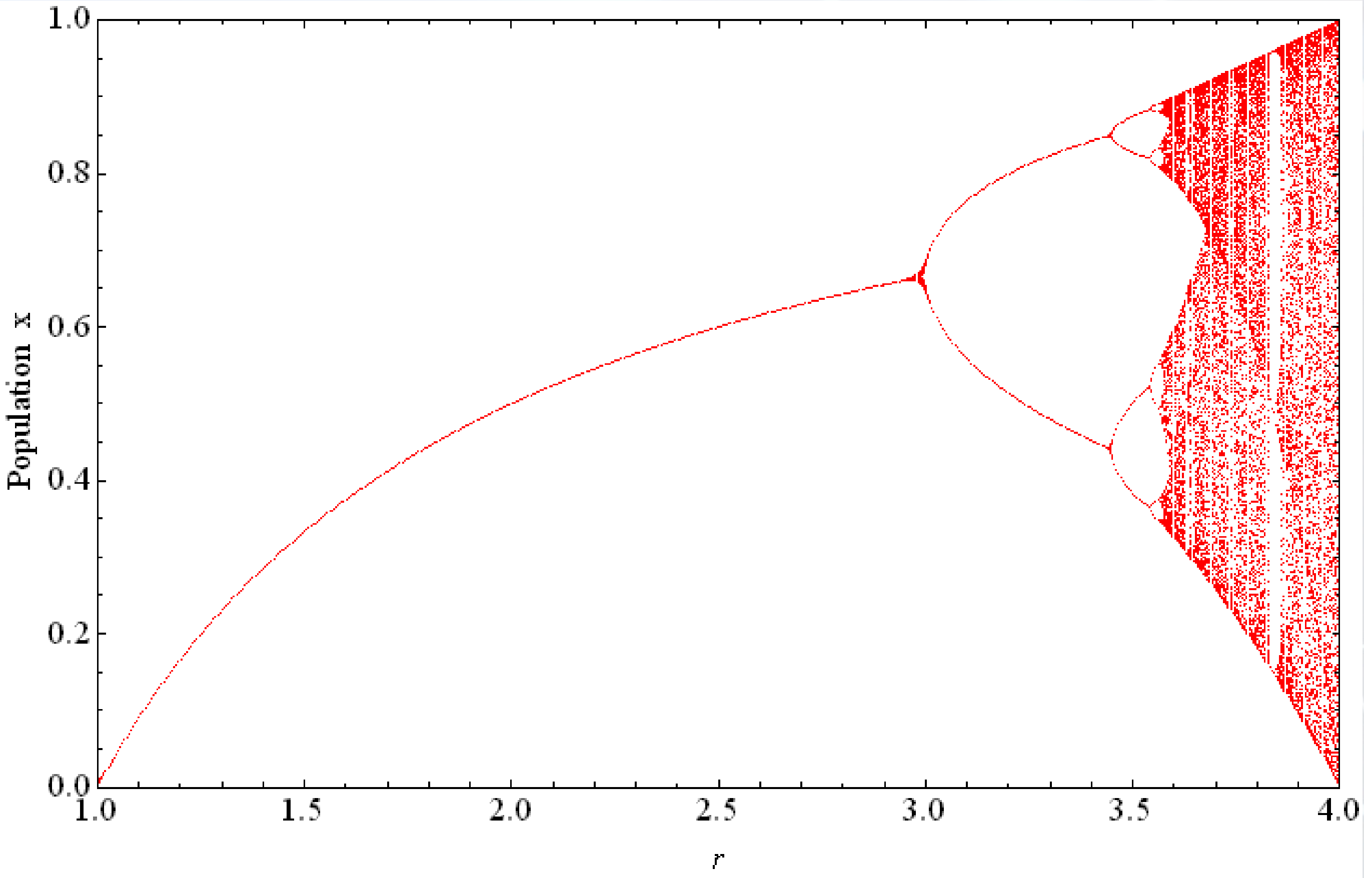
where x_n is an actual population in the current year, x_{n+1} is population in the next year and r is combined rate for reproduction and for starvation.

Zero value for the x means dead population and $x = 1$ means population on its limit.

Bifurcation diagram of Logistic equation

r = 0,5	r = 0,8	r = 1,5	r = 1,5	r = 2,5	r = 3,2	r = 3,5	r = 3,55
0,5	0,5	0,5	**0,2**	0,5	0,5	0,5	0,5
0,125	0,2	0,375	0,2400	0,6250	0,8000	0,8750	0,8875
0,0547	0,128	0,3516	0,2736	0,5859	0,5120	0,3828	0,3544
0,0258	0,0893	0,3419	0,2981	0,6065	0,7995	0,8269	0,8123
0,0126	0,0651	0,3375	0,3139	0,5966	0,5129	0,5009	0,5413
0,0062	0,0487	0,3354	0,3230	0,6017	**0,7995**	0,8750	0,8814
0,0031	0,037	0,3344	0,3280	0,5992	**0,5130**	0,3828	0,3710
0,0015	0,0285	0,3338	0,3306	0,6004	0,7995	0,8269	0,8284
0,0008	0,0222	0,3336	0,3320	0,5998	0,5130	0,5009	0,5047
0,0004	0,0173	0,3335	0,3327	0,6001	**0,7995**	0,8750	0,8874
0,0002	0,0136	0,3334	0,3330	0,5999	**0,5130**	0,3828	0,3547
0,0001	0,0108	0,3334	0,3332	0,6000	0,7995	**0,8269**	0,8125
0	0,0085	0,3333	0,3332	0,6000	0,5130	**0,5009**	0,5408
0	0,0068	0,3333	0,3333	0,6000	**0,7995**	**0,8750**	0,8816
0	0,0054	0,3333	0,3333	0,6000	**0,5130**	**0,3828**	0,3706
0	0,0043	0,3333	0,3333	0,6000	0,7995	0,8269	**0,8280**
0	0,0034	0,3333	0,3333	0,6000	0,5130	0,5009	**0,5055**
0	0,0027	0,3333	0,3333	0,6000	**0,7995**	0,8750	**0,8874**
0	0,0022	0,3333	0,3333	0,6000	**0,5130**	0,3828	**0,3547**
0	0,0017	0,3333	0,3333	0,6000	0,7995	**0,8269**	**0,8126**
0	0,0014	0,3333	0,3333	0,6000	0,5130	**0,5009**	**0,5406**
0	0,0011	0,3333	0,3333	0,6000	**0,7995**	**0,8750**	**0,8816**
0	0,0009	0,3333	0,3333	0,6000	**0,5130**	**0,3828**	**0,3704**

Bifurcation diagram of Logistic equation

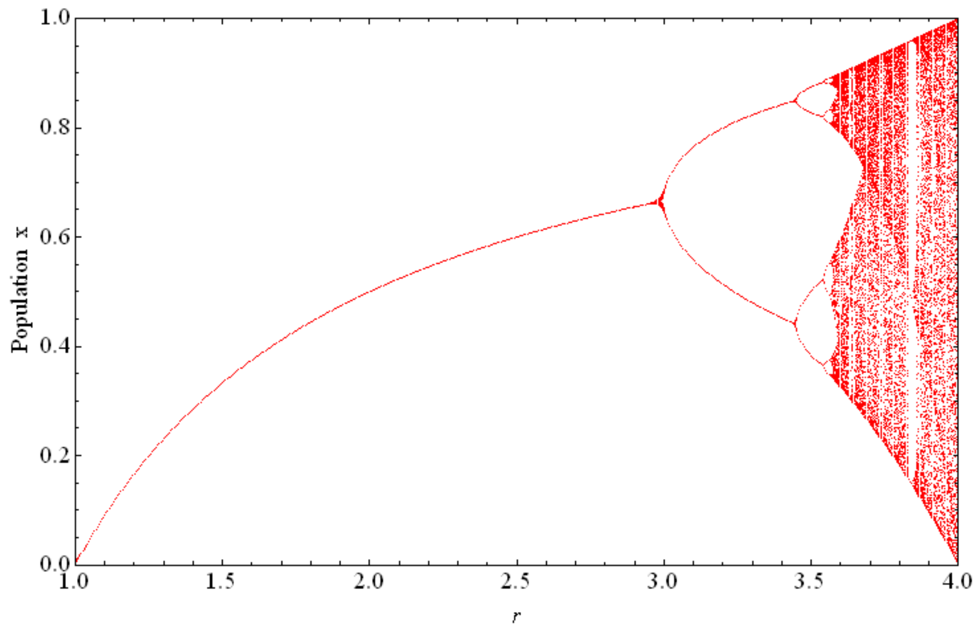


For the value r=3 we can observe the first **bifurcation** (doubling of the functional dependence). Another bifurcations follow for r=3.449, r=3.544 etc.

Bifurcation diagram of Logistic equation

1. Extinction ($r<1$): if the growth rate is less than 1 the system "dies„
2. Fixed point area ($1<r<3$): the series tends to a single value for any initial x_0
3. Oscillation area ($3<r<3.57$): The series jumps between two or more discrete states.
4. Chaos area ($3.57<r<4$): the system can evaluate to any position at all with no apparent order

For higher values of r ($r>4$) all solutions zoom to infinity and the modeling aspects of this function become useless.



Stability and Fixed points of dynamical systems

- Daniel Arovas , Phys Dept., University of California, San Diego
- Steven Strogatz : Nonlinear Dynamics and Chaos,

Stability and Fixed points of dynamical systems

$$\frac{du}{dt} = f(u)$$

Example : Suppose $f(u) = a - bu$, with a and b constant. Then

$$dt = \frac{du}{a - bu} = -b^{-1}\, d\ln(a - bu)$$

Stability and Fixed points of dynamical systems

$$\frac{du}{dt} = f(u)$$

Example : Suppose $f(u) = a - bu$, with a and b constant. Then

$$dt = \frac{du}{a - bu} = -b^{-1}\, d\ln(a - bu)$$

whence

$$t = \frac{1}{b}\ln\left(\frac{a - b\,u(0)}{a - b\,u(t)}\right) \quad \Longrightarrow \quad u(t) = \frac{a}{b} + \left(u(0) - \frac{a}{b}\right)\exp(-bt)\ .$$

Stability and Fixed points of dynamical systems (Model 1)

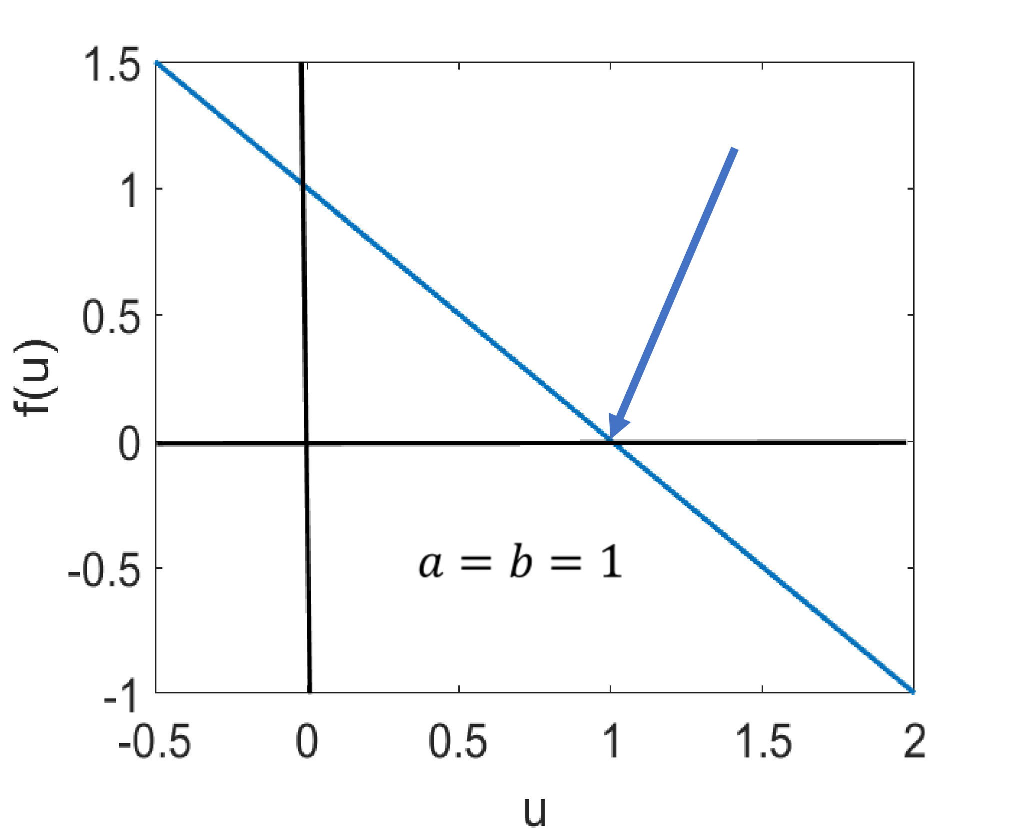
$$\frac{du}{dt} = f(u)$$

Example : Suppose $f(u) = a - bu$, with a and

$$dt = \frac{du}{a - bu} = -b^{-1}$$

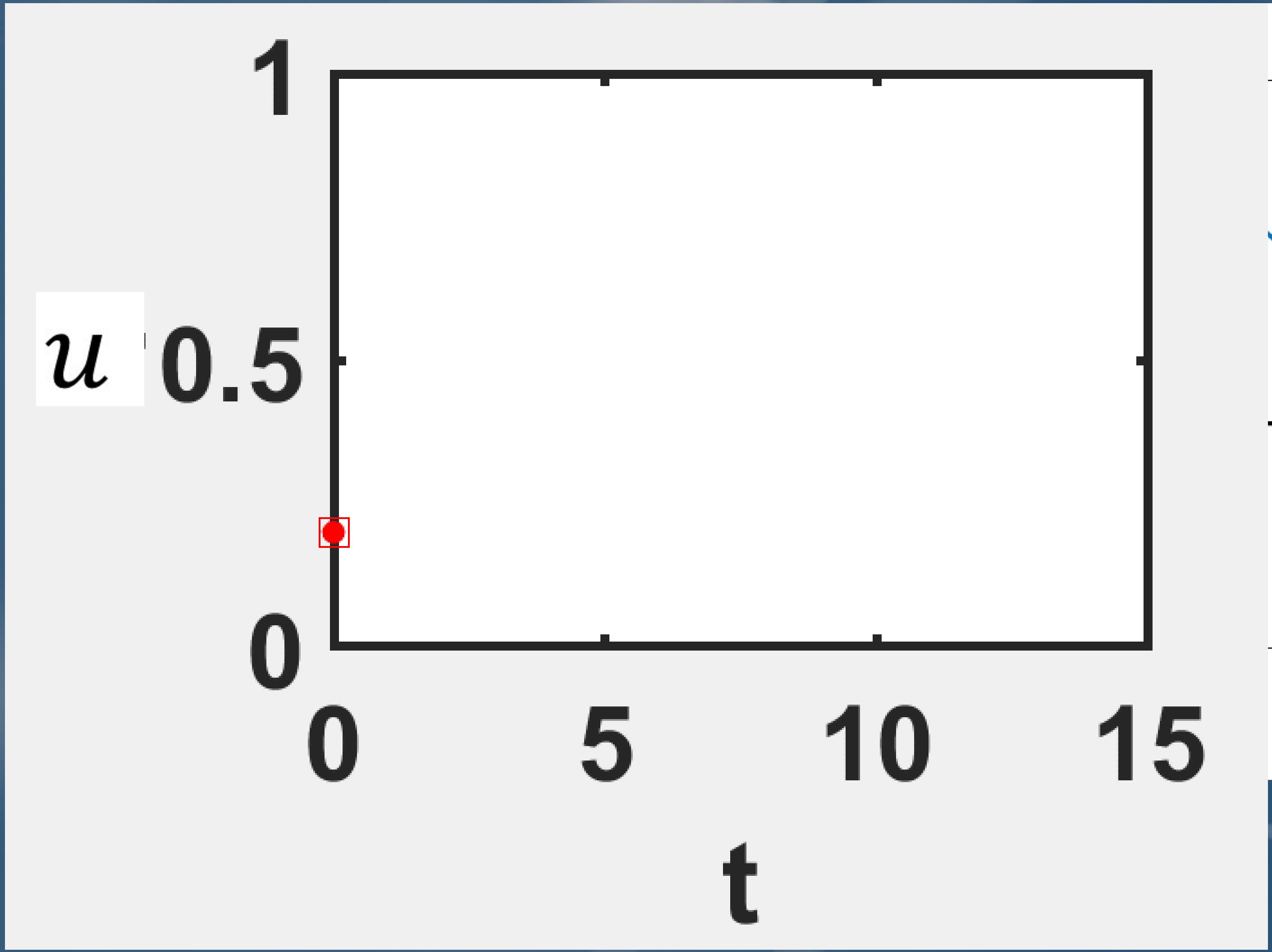
whence

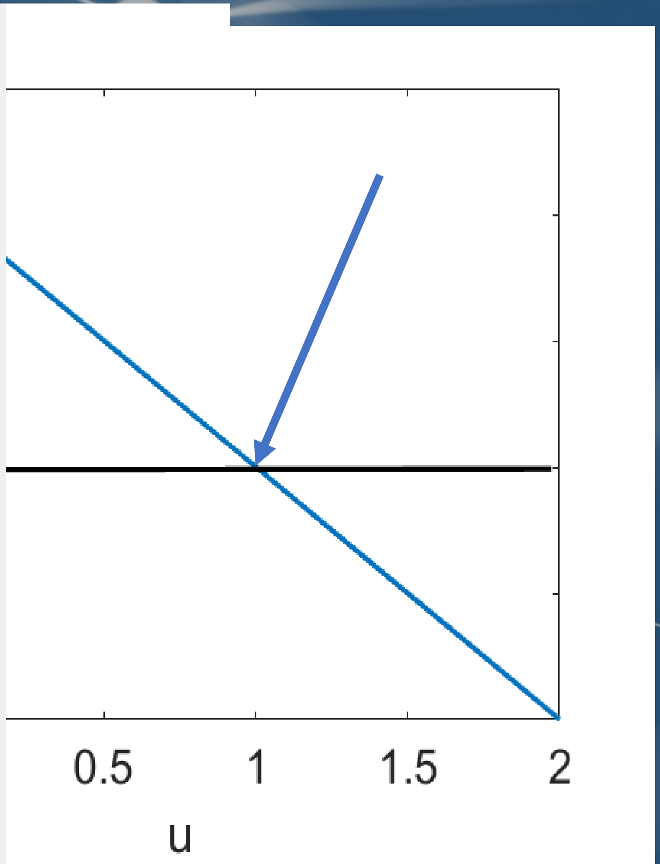
$$t = \frac{1}{b}\ln\left(\frac{a - b\,u(0)}{a - b\,u(t)}\right) \quad \Longrightarrow \quad u(t) = \frac{a}{b}$$



Stability and Fixed points of dynamical systems (Model 1)

$$\frac{du}{dt} = f(u)$$





Numerical Solution with Euler Method

Euler's Method: Tangent Line Approximation

- For our first order initial value problem

$$y' = f(t, y), y(t_0) = y_0,$$

  an alternative is to compute approximate values of the solution $y = \phi(t)$ at a selected set of t-values.
- Ideally, the approximate solution values will be accompanied by error bounds that ensure the level of accuracy.
- In this section, we examine the **tangent line method**, which is also called **Euler's Method**.

Euler's Method: Tangent Line Approximation

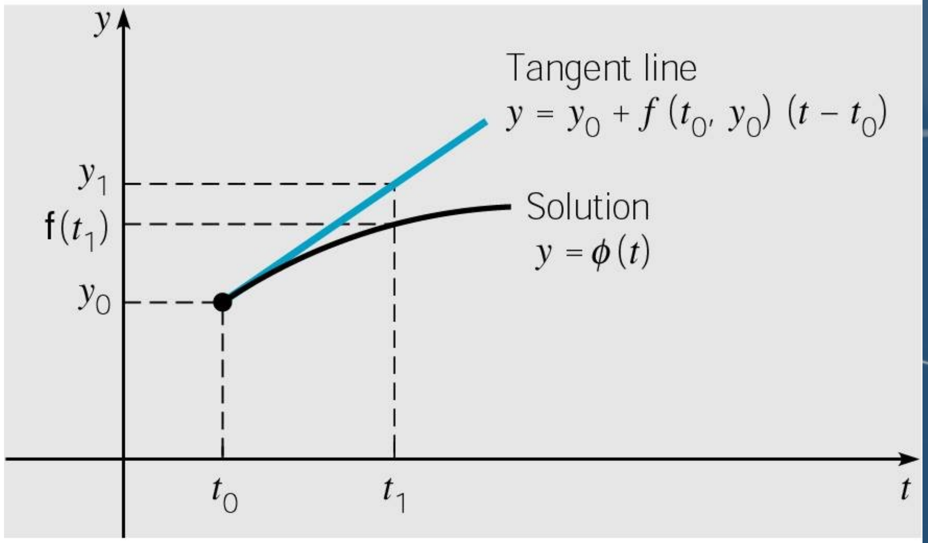
- For the initial value problem

$$y' = f(t, y), y(t_0) = y_0,$$

approximating solution: $y = \phi(t)$ at initial point t_0.

- The line tangent to solution at initial point is thus

$$y = y_0 + f(t_0, y_0)(t - t_0)$$



- Thus, if t_1 is close enough to t_0, we can approximate $\phi(t_1)$ by

$$y_1 = y_0 + f(t_0, y_0)(t_1 - t_0)$$

Euler’s Method: Tangent Line Approximation

- For a point t_2 close to t_1, we approximate $\phi(t_2)$ using the line passing through (t_1, y_1) with slope $f(t_1,y_1)$:

$$y_2 = y_1 + f(t_1, y_1)(t_2 - t_1)$$

- Thus we create a sequence y_n of approximations to $\phi(t_n)$:

$$\begin{aligned} y_1 &= y_0 + f_0 \cdot (t_1 - t_0) \\ y_2 &= y_1 + f_1 \cdot (t_2 - t_1) \\ &\vdots \\ y_{n+1} &= y_n + f_n \cdot (t_{n+1} - t_n) \end{aligned}$$

where $f_n = f(t_n, y_n)$.

- For a uniform step size $h = t_n - t_{n-1}$, Euler’s formula becomes

$$y_{n+1} = y_n + h f_n, \quad n = 0, 1, 2, \ldots$$

Euler Method: Model 1 (pseudo code)

$$\frac{du}{dt} = f(u) = a - bu$$

```
Inputs: a, b, u0, t0, tf, dt  // Initial conditions and step
size
N = (tf - t0) / dt            // Number of time steps
t = t0
u = u0

For i = 1 to N:
    udot = a - b * u          // Compute derivative
    u = u + udot * dt         // Euler update step
    t = t + dt                // Update time
    Print t, u                // Output time and state

End For
```

Stability and Fixed points of dynamical systems (linear stability analysis)

Stability and Fixed points of dynamical systems (linear stability analysis)

$\dot{u} = f(u)$

Perturbing around the fixed points:

$$\dot{u} + \dot{\epsilon} = f(u+\epsilon)\Big|_{u=u^*} = f(u^*) + \epsilon f'(u^*) + \frac{\epsilon^2}{2!} f''(u^*) + O(\epsilon^3)$$

Stability and Fixed points of dynamical systems (linear stability analysis)

$\dot{u} = f(u)$

Perturbing (small and linear) around the fixed points:

$$\dot{u} + \dot{\epsilon} = f(u+\epsilon)\Big|_{u=u^*} = f(u^*) + \epsilon f'(u^*) + \frac{\epsilon^2}{2!} f''(u^*) + O(\epsilon^3)$$

$\dot{\epsilon} = \epsilon f'(u^*)$

The perturbation grows if $f'(u^*) > 0$ and decreases if $f'(u^*) < 0$

Dropping higher order terms we obtain

$$\epsilon(t) = \exp[f'(u^*)t]\,\epsilon(0)$$

Or $t(\epsilon) = \frac{1}{f'(u^*)} \ln(\frac{\epsilon}{\epsilon(0)})$

so the approach to a stable fixed point takes a logarithmically infinite time. For the unstable case, the deviation grows exponentially, until eventually the linearization itself fails.

Stability and Fixed points of dynamical systems (vector flow)

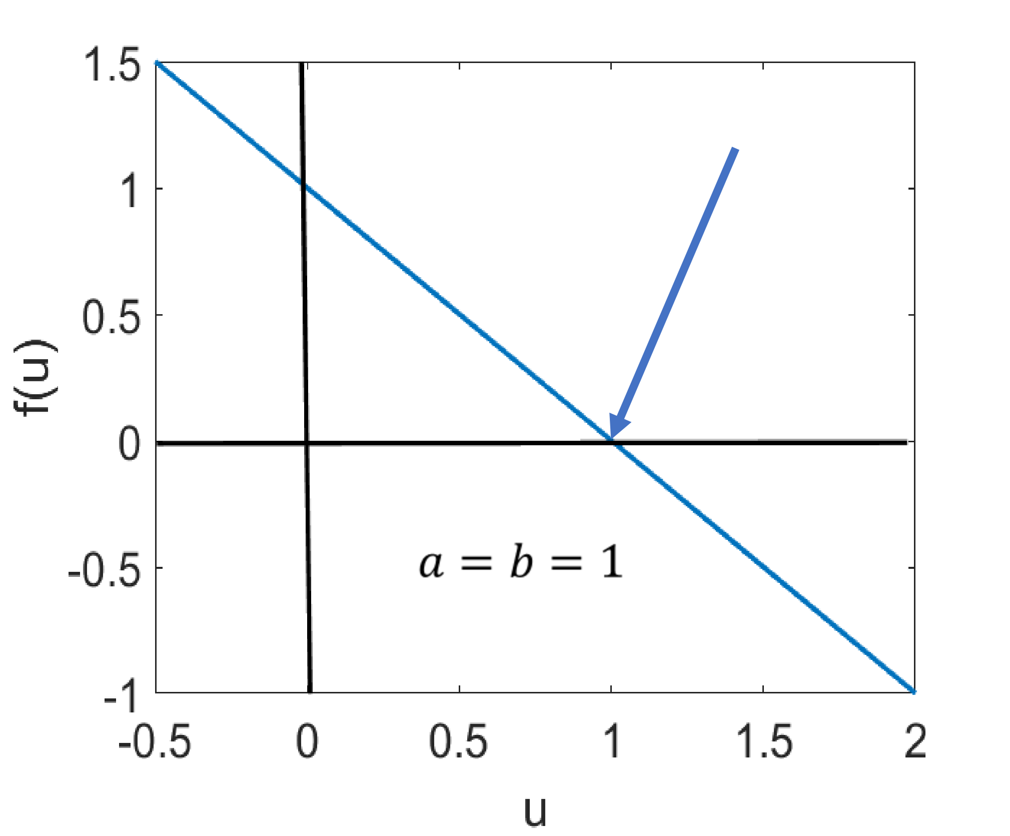
$$\frac{du}{dt} = f(u)$$

Example : Suppose $f(u) = a - bu$, with a and

$$dt = \frac{du}{a - bu} = -b^{-1}$$

whence

$$t = \frac{1}{b}\ln\left(\frac{a - b\,u(0)}{a - b\,u(t)}\right) \quad \Longrightarrow \quad u(t) = \frac{a}{b}$$

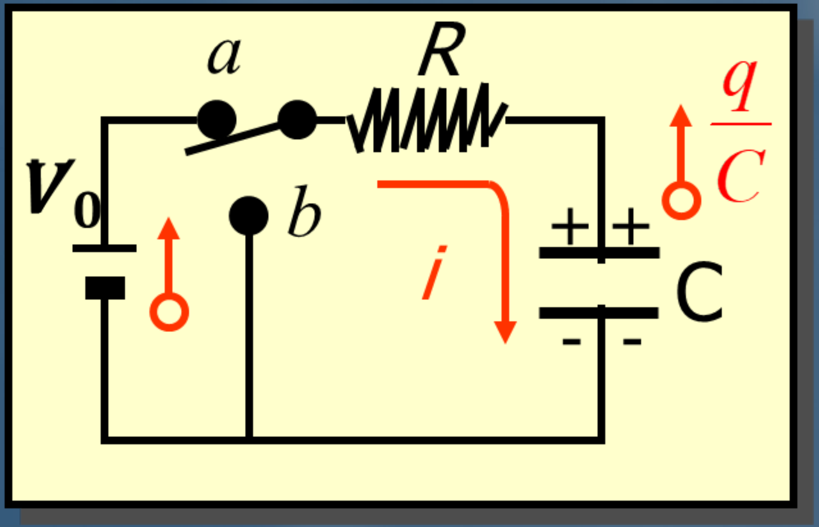


$$\dot{u} = f(u) \quad \Longrightarrow \quad \begin{cases} f(u) > 0 & \dot{u} > 0 & \Rightarrow & \text{move to right} \\ f(u) < 0 & \dot{u} < 0 & \Rightarrow & \text{move to left} \\ f(u) = 0 & \dot{u} = 0 & \Rightarrow & \text{fixed point} \end{cases}$$

The perturbation grows if $f'(u^*) > 0$ and decreases if $f'(u^*) < 0$

RC Circuit: Charging Capacitor

RC Circuit: Charging Capacitor

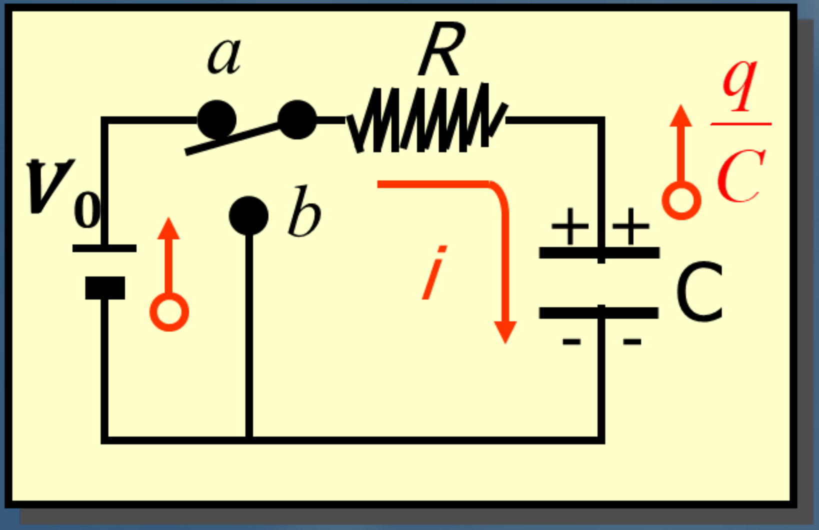


$$V_0 - \frac{q}{C} = iR$$

$$R\frac{dq}{dt} = V_0 - \frac{q}{C}$$

Q: What will be the asymptotic value of q?

RC Circuit: Charging Capacitor



$$V_0 - \frac{q}{C} = iR$$

$$R\frac{dq}{dt} = V_0 - \frac{q}{C}$$

$$\int_0^q \frac{dq}{(CV_0 - q)} = \int_o^t \frac{dt}{RC}$$

$$\ln(CV_0 - q) - \ln(CV_0) = \frac{-t}{RC}$$

$$\ln\frac{(CV_0 - q)}{CV_0} = \frac{-t}{RC}$$

At time $t = 0$: $q = CV_0(1 - 1)$; $q = 0$

At time $t = \infty$: $q = CV_0(1 - 0)$; $q_{max} = CV_0$

$$q = CV_0\left(1 - e^{-t/RC}\right)$$

The charge q rises from zero initially to its maximum value $q_{max} = CV_0$

RC Circuit: Charging Capacitor

Rearranging for Euler's Method

Rewriting the equation for the rate of change of charge:

$$\frac{dq}{dt} = \frac{V_0 - q/C}{R}$$

Using Euler's method:

$$q(t + \Delta t) = q(t) + \Delta t \cdot \frac{V_0 - q/C}{R}$$

Since $V_C = q/C$, we can compute the capacitor voltage at each step using:

$$V_C = q/C$$

1. Initialize parameters:
   - Set R = [resistance value]
   - Set C = [capacitance value]
   - Set Δt = [time step size]
   - Set t_final = [final simulation time]
   - Set V_0 = [input voltage]
   - Set q = 0 (initial charge on the capacitor)

2. Compute the number of iterations:
   - N = t_final / Δt

3. Loop over each time step:
   for n = 0 to N-1:
      a. Compute the rate of change of charge:
         dq_dt = (V_0 - q / C) / R

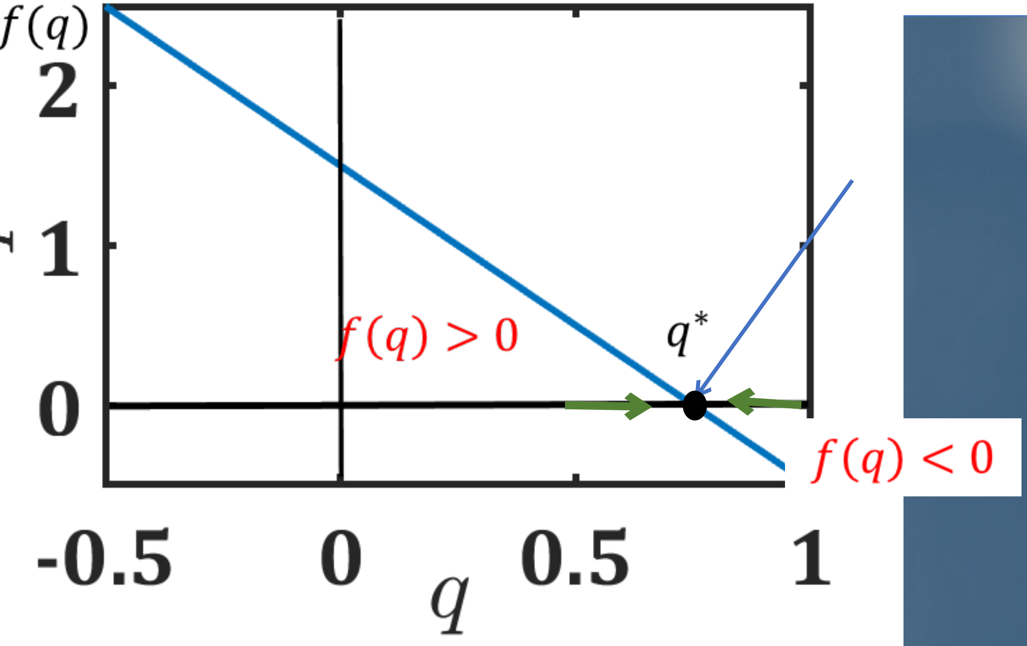
      b. Update the charge:
         q = q + Δt * dq_dt

      c. Compute the capacitor voltage:
         V_C = q / C

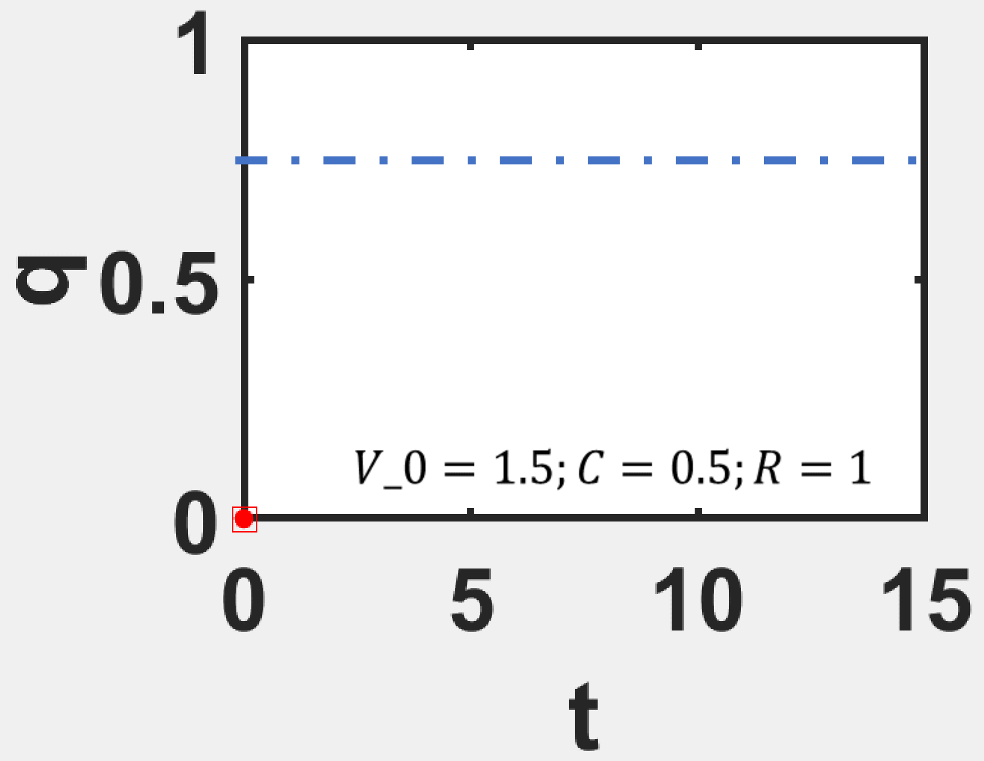
      d. Update time:
         t = (n + 1) * Δt

      e. (Optional) Store or print results:
         print("Time:", t, "Charge:", q, "Voltage across capacitor:", V_C)

Interpreting a differential equation as a vector field Flows on the Line



Solution
$q = CV_0\left(1 - e^{-t/RC}\right)$



'he perturbation grows if $f'(u^*) > 0$
nd decreases if $f'(u^*) < 0$

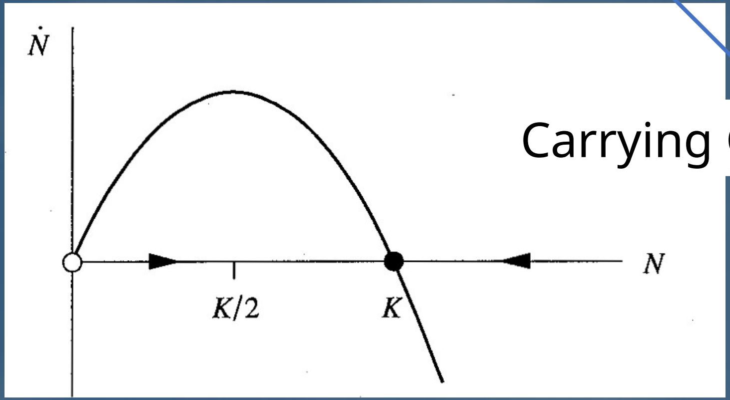
Interpreting a differential equation as a vector field Flows on the Line

Logistic equation (Model 3)

Analysis of Fixed points

$$\dot{N} = rN\left(1 - \frac{N}{K}\right)$$

Per Capita Growth Rate $\dot{N}/N$



Carrying Capacity

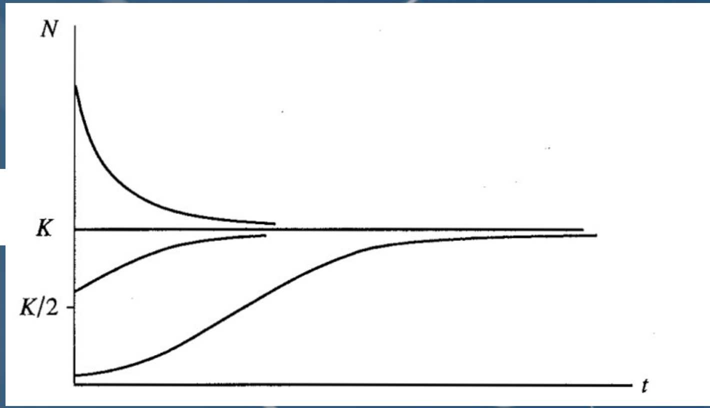


Figure 2.3.3 also allows us to deduce the qualitative shape of the solutions. For example, if $N_0 < K/2$, the phase point moves faster and faster until it crosses $N = K/2$, where the parabola in Figure 2.3.3 reaches its maximum. Then the phase point slows down and eventually creeps toward $N = K$. In biological terms, this means that the population initially grows in an accelerating fashion, and the graph of $N(t)$ is concave up. But after $N = K/2$, the derivative $\dot{N}$ begins to decrease, and so $N(t)$ is concave down as it asymptotes to the horizontal line $N = K$ (Figure 2.3.4). Thus the graph of $N(t)$ is S-shaped or *sigmoid* for $N_0 < K/2$.

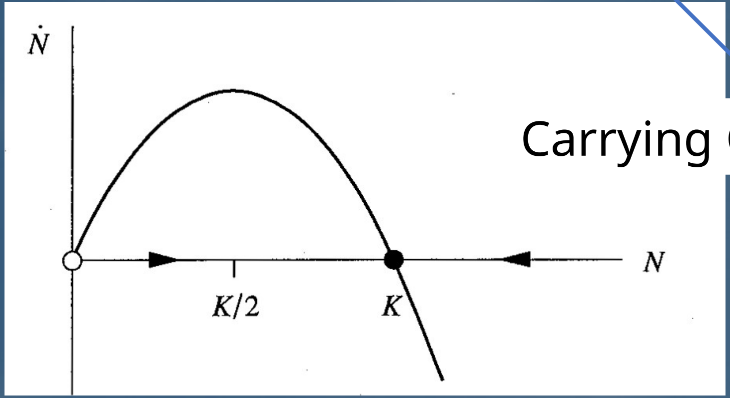
$N(t)$ is S-shaped or *sigmoid* for $N_0 < K/2$.

Interpreting a differential equation as a vector field Flows on the Line

Logistic equation (Model 3)

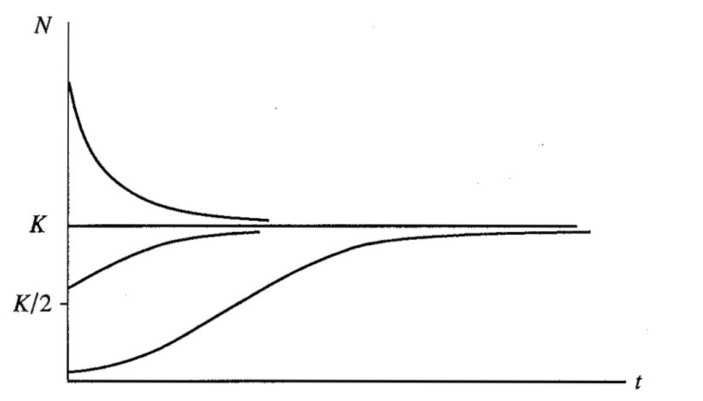
$$\dot{N} = rN\left(1 - \frac{N}{K}\right)$$

Carrying Capacity



Analysis of Fixed points

Per Capita Growth Rate $\dot{N}/N$



Analytical Calculation

Interpreting a differential equation as a vector field Flows on the Line

Logistic equation (Model 3)

$$\dot{N} = rN\left(1 - \frac{N}{K}\right)$$

Input: r (growth rate), x0 (initial value), dt (time step), N (number of iterations)
Output: x values over time

1. Initialize:
   x ← x0   # Initial condition
   t ← 0   # Initial time

2. Loop for N steps:
   a. Compute dx/dt = r * x * (1 - x)
   b. Update x using Euler's method:
      x ← x + dt * (r * x * (1 - x))
   c. Update time: t ← t + dt
   d. Store (t, x) for plotting (optional)

3. End loop

4. Return x values

- **Fixed Points and stability**
  - **Linear stability in 1-d model: Logistic equations, CR circuit**
  - **2-d linear model (concept of star, node, line of fixed points, and saddle points) *****
  - **General description (Role of trace and determinant)**
  - **Love affairs Romeo and Juliet**

- **Fixed points and Linearization of 2-d systems *(concept of Jacobian)***
  - **Rabit vs Sheep**
  - **Lotka-Volterra competition model**
  - **SIR model**

- **High dimensional systems**
  - **Double pendulum**
  - **Lorenz attractor**

***** *Numerical method: Euler, RK-4***

Interpreting differential eqn as vector field

$$\dot{x} = \sin x.$$

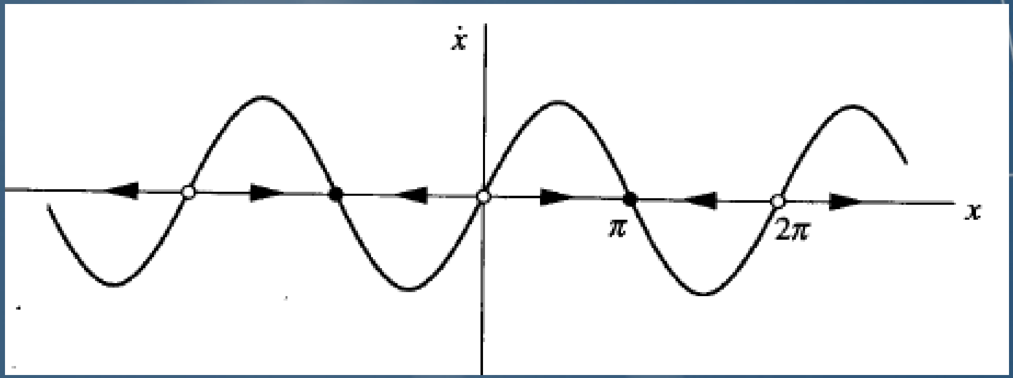
$$t = \ln\left|\frac{\csc x_0 + \cot x_0}{\csc x + \cot x}\right|.$$

Interpreting differential eqn as vector field (Model 4)

$$\dot{x} = \sin x.$$

$$t = \ln\left|\frac{\csc x_0 + \cot x_0}{\csc x + \cot x}\right|.$$

If we think x as position of a particle moving along real line and $\dot{x}$ as velocity vector, then the differential equation represents the vector-field: it dictates the velocity $\dot{x}$ at each x.



Elementary Physics Problems :vibrations of a mass hanging from a spring (Model 5)

$$m\frac{d^2\boldsymbol{x}}{dt^2} + k\boldsymbol{x} = \boldsymbol{0}$$

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = -\frac{k}{m}x = -\omega^2 x$$

- As we have to deal with nonlinear ode , we have to understand the vector field.
- To find the vector field we will detect the **state** of the systems in each time i.e current position and velocity.
- The systems (left) assign a vector $(\dot{x}, \dot{v}) = (v, -\omega^2 x)$ at each point (x, v), and therefore represents a **vector field** on the **phase plane.**
- **Plot it**

Strogatz, Chapter 5

Elementary Physics Problems :vibrations of a mass hanging from a spring

$$m\frac{d^2\boldsymbol{x}}{dt^2} + k\boldsymbol{x} = \mathbf{0}$$

$$\frac{dx}{dt} = v \qquad \frac{dv}{dt} = -\frac{k}{m}x = -\omega^2 x$$

- As we have to deal with nonlinear ode , we have to understand the vector field.
- To find the vector field we will detect the **state** of the systems in each time i.e current position and velocity.
- The systems (left) assign a vector $(\dot{x}, \dot{v}) = (v, -\omega^2 x)$ at each point (x, v), and therefore represents a **vector field** on the **phase plane ($\boldsymbol{x}$ vs $\boldsymbol{y}$, here $\boldsymbol{x}$ vs $\boldsymbol{v}$).**
- **Plot it**

- Start with when you are on the x-axis:

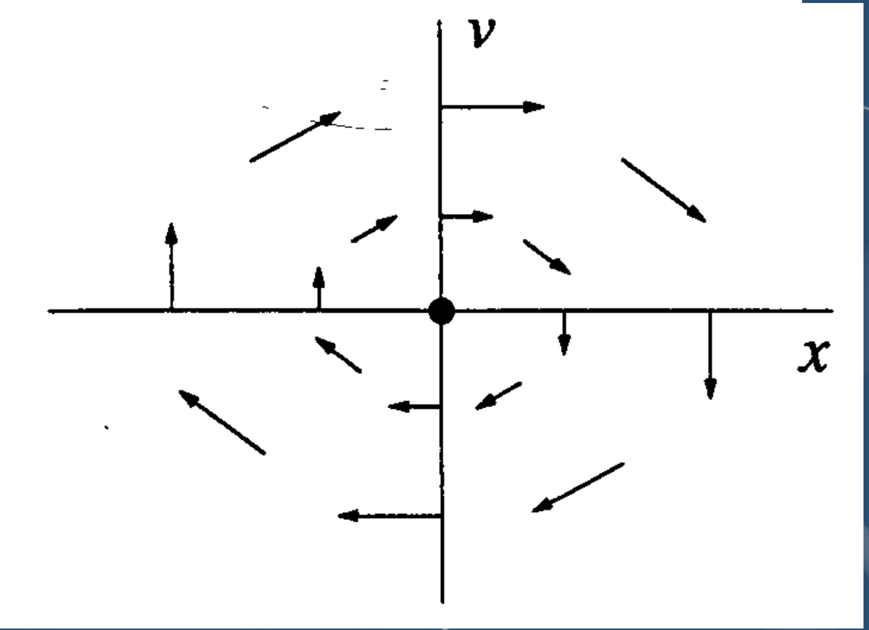
Then $v = 0$, $(\dot{x}, \dot{v}) = (0\ -\omega^2 x)$

Thus for positive x the vectors points vertically downward, oppo

- Now, you are on the v-axis:

Then $x = 0$, $(\dot{x}, \dot{v}) = (v, 0)$

Thus for positive v, the vector filed points to the right, and to the



Elementary Physics Problems :vibrations of a spring hanging from a spring

$$m\frac{d^2\boldsymbol{x}}{dt^2} + k\boldsymbol{x} = \boldsymbol{0}$$

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = -\frac{k}{m}x = -\omega^2 x$$

- Start with when you are on the x-axis:

Then $v = 0, (\dot{x}, \dot{v}) = (0\ -\omega^2 x)$

Thus for positive x the vectors points vertically downward, opposite effect in the negative x axis.

- Now, you are on the v-axis:

Then $x = 0, (\dot{x}, \dot{v}) = (v, 0)$

Thus for positive v, the vector filed points to the right, and to the left if $v < 0$

- The imaginary flow swirls about the origin.
- The origin is like an eye of hurricane, if someone starts there will remain there: FP (0,0)

(static equilibrium in Physics !!!)

- Someone starts not from origin will circulate about the origin.
- When the x is most negative, v is zero, the spring is fully compresses, and then v becomes positive and the spring comes to the equilibrium position ($x = 0$), with a large positive v, thus it overshoots the spring.

Elementary Physics Problems :vibrations of a mass hanging from a spring

$$m\frac{d^2\boldsymbol{x}}{dt^2} + k\boldsymbol{x} = \boldsymbol{0}$$

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = -\frac{k}{m}x = \omega^2 x$$

- The imaginary flow swirls about the origin.
- The origin is like a eye of hurricane, if someone starts there will remain there: FP (0,0) (static equilibrium in Physics !!!)

- Someone starts not from origin will circulate about the origin, eventually reaching to that IC:
- When the x is most negative, v is zero, the spring is fully compresses, and then v becomes positive and the spring comes to the equilibrium position ($x = 0$), with a large positive v, thus it overshoots the spring.

- The shape of the orbit: not a circle.
- It is actually an ellipse. Why????

$w^2x^2 + v^2 = c$ is constant (conservation of energy)

2-d linear model (concept of star, node, line of fixed points, and saddle points)

2-d linear model (concept of star, node, line of fixed points, and saddle points)

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = A\boldsymbol{x}$$

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$A = \begin{pmatrix} a & 0 \\ 0 & -1 \end{pmatrix}$$

2-d linear model (concept of star, node, line of fixed points, and saddle points) (Model 6)

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = A\boldsymbol{x}$$

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

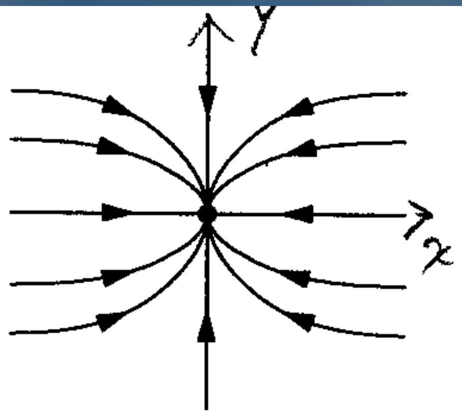
$$A = \begin{pmatrix} a & 0 \\ 0 & -1 \end{pmatrix}$$

$$\frac{\mathrm{d}x}{\mathrm{d}t} = ax \qquad \frac{\mathrm{d}y}{\mathrm{d}t} = -\mathrm{y}$$

$$x = x_0 e^{at} \qquad y = y_0 e^{-t}$$

- When $a < 0$, x decays (actually both) exponentially. Therefore, both will reach to zero.

2-d linear model (concept of star, node, line of fixed points, and saddle points)

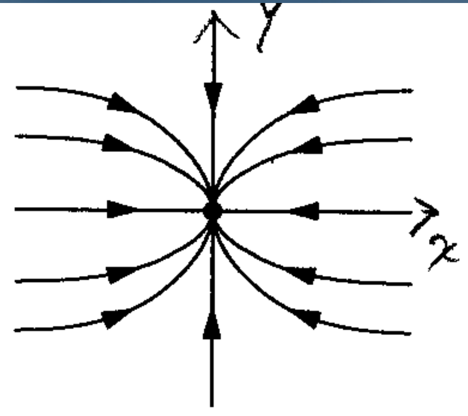


(a) $a < -1$

- When $a < 0$, x decays (actually both) exponentially. Therefore, both will reach to zero.

- **Case 1:** $\boldsymbol{a < -1}$ Along x , it decays faster compared to y. Starting from far away point, the trajectory approaches to the origin tangent to the slower direction (y-direction) .
- **Stable Node**

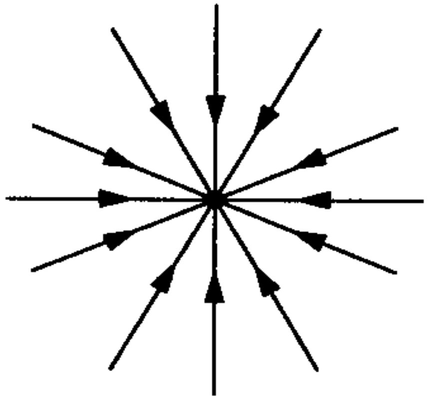
2-d linear model (concept of star, node, line of fixed points, and saddle points)



(a) $a < -1$

- When $a < 0$, x decays (actually both) exponentially. Therefore both will reach to zero.

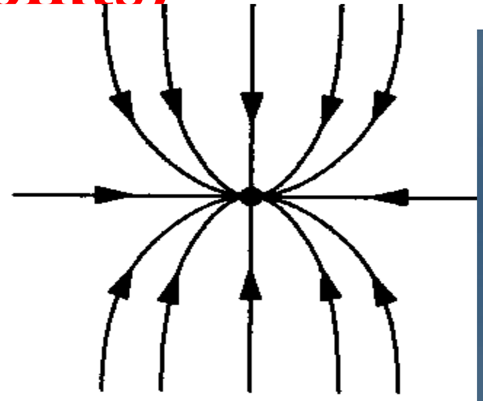
- **Case 1:** $\boldsymbol{a < -1}$ Along x , it decays faster compared to y. Starting from far away point, the trajectory approaches to the origin tangent to the slower direction (y-direction) .
- **Stable Node**



(b) $a = -1$

- When $a < 0$, x decays (actually both) exponentially. Therefore, both will reach to zero.
- **Case 2:** $\boldsymbol{a = -1}$ The decay rates are precisely equal for both. Thus trajectories are straight line passing through origin.
- Called as **star.**

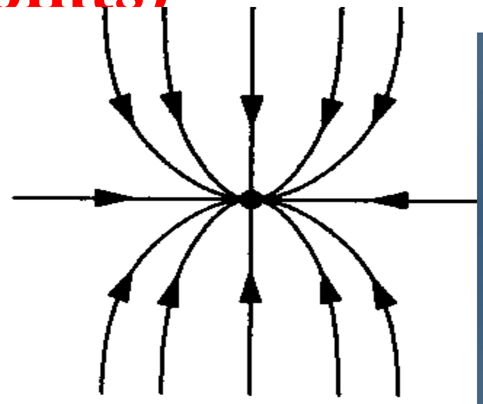
2-d linear model (concept of star, node, line of fixed points, and saddle points)



(c) $-1 < a < 0$

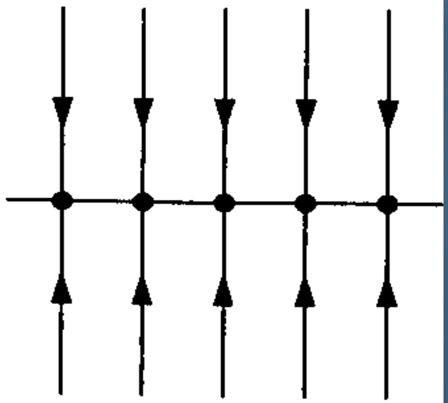
- **Case 3: -1<a < 0** Along y , it decays faster compared to x. Starting from far away point, the trajectory approaches to the origin tangent to the slower direction (x-direction) .
- **Stable Node**

2-d linear model (concept of star, node, line of fixed points, and saddle points)



(c) $-1 < a < 0$

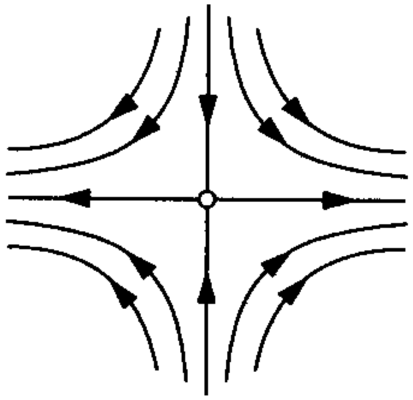
- **Case 3: -1<a < 0** Along y , it decays faster compared to x. Starting from far away point, the trajectory approaches to the origin tangent to the slower direction (x-direction) .
- **Stable Node**



(d) $a = 0$

- **Case 4: a= 0**, $x(t) = x0$ for long term evaluation.
- **So lines of fixed points**

2-d linear model (concept of star, node, line of fixed points, and saddle points)



(e) $a > 0$

Finally when $a > 0$ (Figure 5.1.5e), $\mathbf{x}^*$ becomes unstable, due to the exponential growth in the x-direction. Most trajectories veer away from $\mathbf{x}^*$ and head out to infinity. An exception occurs if the trajectory starts on the y-axis; then it walks a tightrope to the origin. In forward time, the trajectories are asymptotic to the x-axis; in backward time, to the y-axis. Here $\mathbf{x}^* = \mathbf{0}$ is called a ***saddle point***. The

Classification of linear systems

A ***two-dimensional linear system*** is a system of the form

$$\dot{x} = ax + by$$
$$\dot{y} = cx + dy$$

where a, b, c, d are parameters. If we use boldface to denote vectors, this system can be written more compactly in matrix form as

$$\dot{\mathbf{x}} = A\mathbf{x},$$

where

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ and } \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}.$$

Classification of linear systems

A ***two-dimensional linear system*** is a system of the form

$$\dot{x} = ax + by$$
$$\dot{y} = cx + dy$$

where a, b, c, d are parameters. If we use boldface to denote vectors, this system can be written more compactly in matrix form as

$$\dot{\mathbf{x}} = A\mathbf{x},$$

where

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ and } \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}.$$

$$\boldsymbol{x}(\boldsymbol{t}) = e^{\lambda t}\boldsymbol{v}$$

Classification of linear systems

A ***two-dimensional linear system*** is a system of the form

$$\dot{x} = ax + by$$
$$\dot{y} = cx + dy$$

where a, b, c, d are parameters. If we use boldface to denote vectors, this system can be written more compactly in matrix form as

$$\dot{\mathbf{x}} = A\mathbf{x},$$

where

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \text{ and } \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}.$$

$$\boldsymbol{x}(\boldsymbol{t}) = e^{\lambda t}\boldsymbol{v}$$

To find the conditions on $\mathbf{v}$ and λ, we substitute $\mathbf{x}(t) = e^{\lambda t}\mathbf{v}$ into $\dot{\mathbf{x}} = A\mathbf{x}$, and obtain $\lambda e^{\lambda t}\mathbf{v} = e^{\lambda t}A\mathbf{v}$. Canceling the nonzero scalar factor $e^{\lambda t}$ yields

$$A\mathbf{v} = \lambda\mathbf{v},$$

Classification of linear systems

$$\boldsymbol{x}(\boldsymbol{t}) = e^{\lambda t}\boldsymbol{v}$$

To find the conditions on $\mathbf{v}$ and λ, we substitute $\mathbf{x}(t) = e^{\lambda t}\mathbf{v}$ into $\dot{\mathbf{x}} = A\mathbf{x}$, and obtain $\lambda e^{\lambda t}\mathbf{v} = e^{\lambda t}A\mathbf{v}$. Canceling the nonzero scalar factor $e^{\lambda t}$ yields

$$A\mathbf{v} = \lambda\mathbf{v},$$

which says that the desired straight line solutions exist if $\mathbf{v}$ is an ***eigenvector*** of A with corresponding ***eigenvalue*** λ. In this case we call the solution (2) an ***eigen-solution***.

Classification of linear systems

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

the characteristic equation becomes

$$\det\begin{pmatrix} a-\lambda & b \\ c & d-\lambda \end{pmatrix} = 0 .$$

Expanding the determinant yields

$$\lambda^2 - \tau\lambda + \Delta = 0$$

where

$$\tau = \text{trace}(A) = a + d ,$$
$$\Delta = \det(A) = ad - bc .$$

Then

$$\lambda_1 = \frac{\tau + \sqrt{\tau^2 - 4\Delta}}{2} , \qquad \lambda_2 = \frac{\tau - \sqrt{\tau^2 - 4\Delta}}{2}$$

$\mathbf{x}(t) = e^{\lambda t}\mathbf{v}$ into $\dot{\mathbf{x}} = A\mathbf{x}$, and
ctor $e^{\lambda t}$ yields

ist if $\mathbf{v}$ is an ***eigenvector*** of A
call the solution (2) an ***eigen-***

$$\mathbf{x}(t) = c_1 e^{\lambda_1 t}\mathbf{v}_1 + c_2 e^{\lambda_2 t}\mathbf{v}_2 .$$

Solving Initial Value Problem (Model 7)

Solve the initial value problem $\dot{x} = x + y$, $\dot{y} = 4x - 2y$, subject to the initial condition $(x_0, y_0) = (2, -3)$.

Solution: The corresponding matrix equation is

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 4 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Solving Initial Value Problem (Model 7)

Solve the initial value problem $\dot{x} = x + y$, $\dot{y} = 4x - 2y$, subject to the initial condition $(x_0, y_0) = (2, -3)$.

Solution: The corresponding matrix equation is

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 4 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

$$\lambda_1 = 2, \quad \lambda_2 = -3.$$

$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \mathbf{v}_2 = \begin{pmatrix} 1 \\ -4 \end{pmatrix}.$$

Solving Initial Value Problem

Solve the initial value problem $\dot{x} = x + y$, $\dot{y} = 4x - 2y$, subject to the initial condition $(x_0, y_0) = (2, -3)$.

Solution: The corresponding matrix equation is

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 4 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

$$\lambda_1 = 2, \quad \lambda_2 = -3.$$

$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \mathbf{v}_2 = \begin{pmatrix} 1 \\ -4 \end{pmatrix}.$$

$$\mathbf{x}(t) = c_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix} e^{2t} + c_2 \begin{pmatrix} 1 \\ -4 \end{pmatrix} e^{-3t}.$$

$$(x_0, y_0) = (2, -3).$$

$$x(t) = e^{2t} + e^{-3t},$$
$$y(t) = e^{2t} - 4e^{-3t}.$$

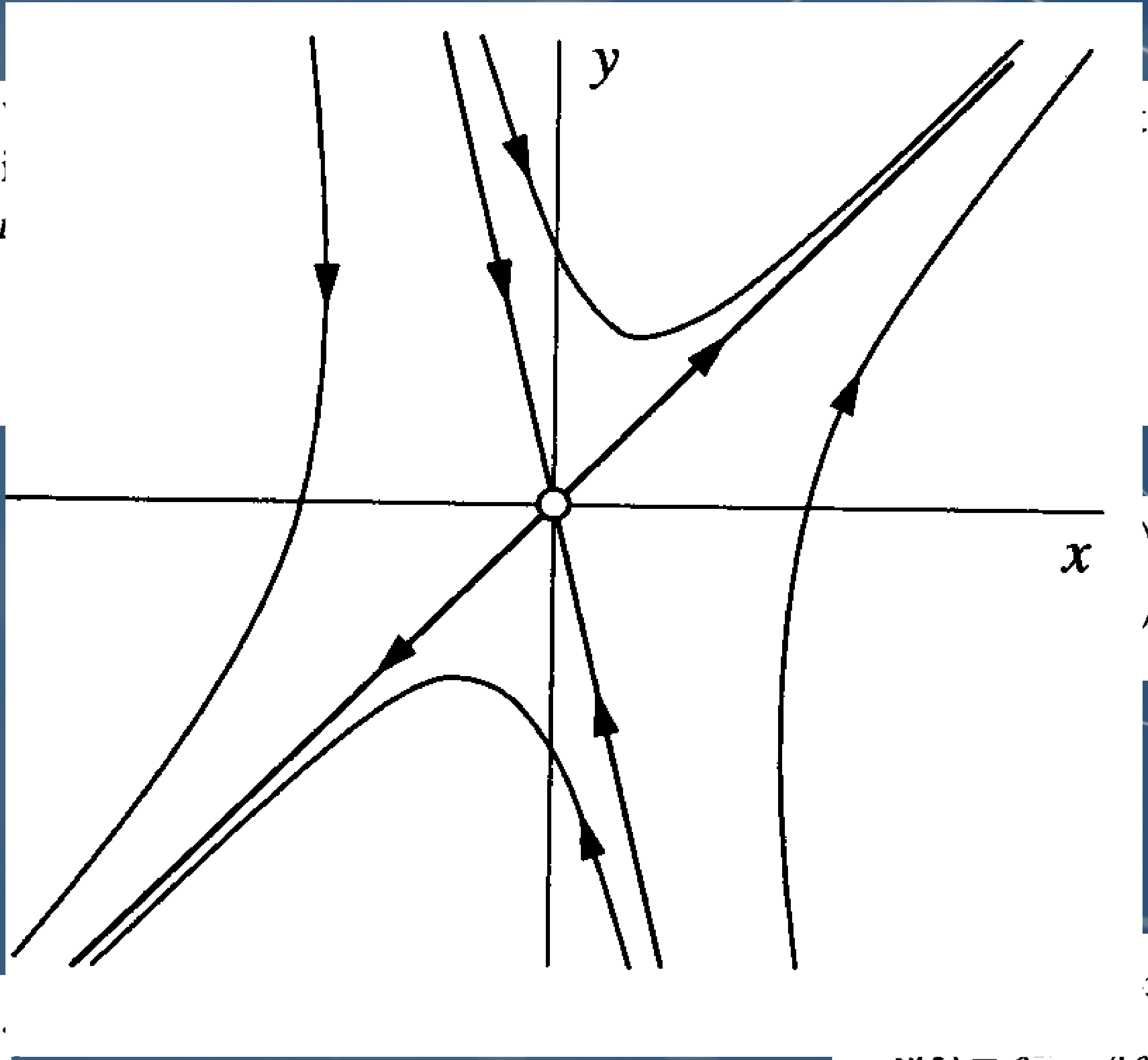
Solving Initial Value Problem

Sol

conditi

Soli

to the initial



$(x_0, y_0) = (2, -3)$.

$y(t) = e^{-} - 4e^{-3t}$

Love Affairs (Model 8)

Romeo is in love with Juliet, but in our version of this story, Juliet is a fickle over. The more Romeo loves her, the more Juliet wants to run away and hide. But vhen Romeo gets discouraged and backs off, Juliet begins to find him strangely atractive. Romeo, on the other hand, tends to echo her: he warms up when she loves iim, and grows cold when she hates him.

Romeo: *If I profane with my unworthiest hand*

Romeo: *O, then, dear saint, let lips do what hands do;*
They pray, grant thou, lest faith turn to despair.

Juliet: *Saints do not move, though grant for prayer's sake*

Romeo: *Then move not, while my prayer's effect I take.*
Thus from my lips, by yours, my sin is purged.

and
r kiss.
ur hand too much,
nds do touch,

Romeo: *Have not saints lips, and holy palmers too?*

Juliet: *Ay, pilgrim, lips that they must use in prayer.*

Love Affairs

Let

$$R(t) = \text{Romeo's love/hate for Juliet at time } t$$
$$J(t) = \text{Juliet's love/hate for Romeo at time } t\,.$$

Positive values of R, J signify love, negative values signify hate. Then a model for their star-crossed romance is

$$\dot{R} = aJ$$
$$\dot{J} = -bR$$

Love Affairs

Let

$R(t)$ = Romeo's love/hate for Juliet at time t

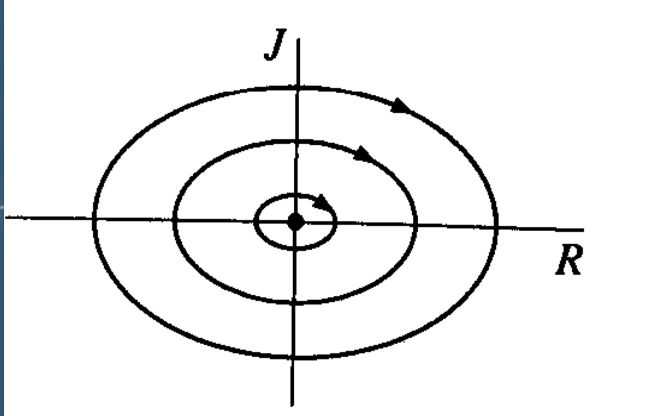
$J(t)$ = Juliet's love/hate for Romeo at time t.

Positive values of R, J signify love, negative values signify hate. Then a model for their star-crossed romance is

$$\dot{R} = aJ$$

$$\dot{J} = -bR$$

- **Never ending cycles of love and hate**
- **Governing system has center at (0,0)**
- **They manage love simultaneously one quarter of time**



Numerical techniques (Derivative of a function)

$$f(x_0+h) = f(x_0) + h\,f'(x_0) + \frac{h^2}{2!} f''(x_0) + \cdots + \frac{h^n}{n!}\, f^n(x_0) + R_n(x)$$

Denotes the difference between Talylor Polynomial of degree n and the original function

For the first derivative

$$f(x_0+h) = f(x_0) + h\,f'(x_0) + R_1(x)$$

Saml
l

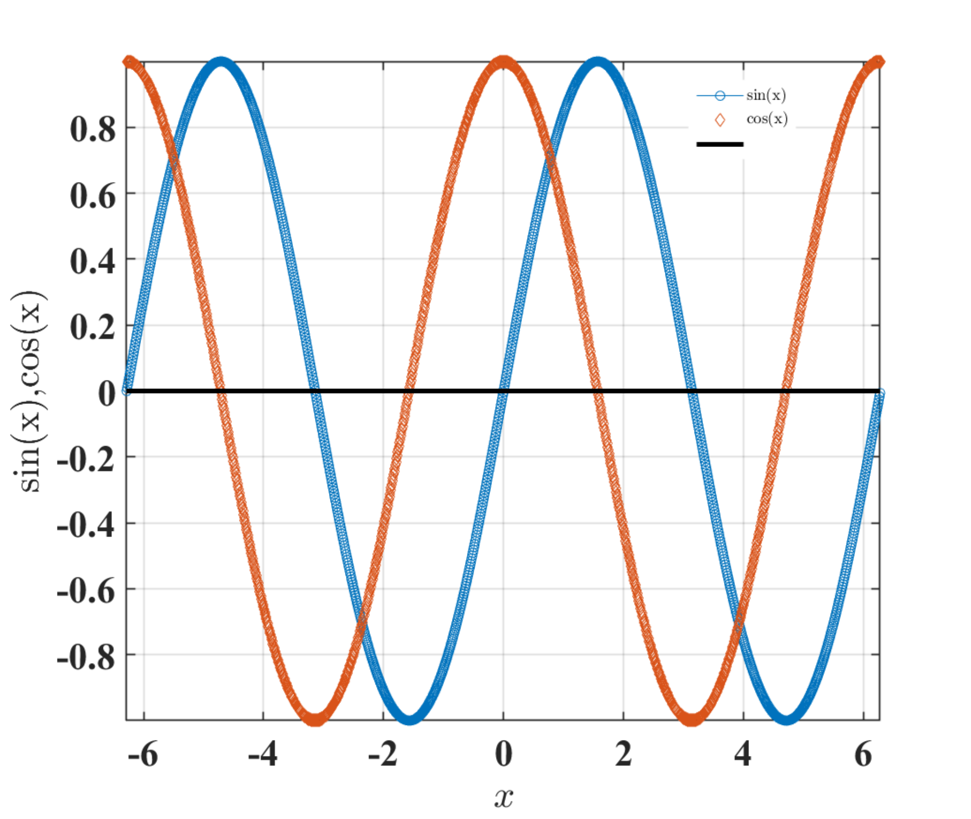
$$Lt_{h\to 0}\frac{f(x_0+h)-f(x_0)}{h} = f'(x_0)$$

Numerical techniques

$$f(x_0+h) = f(x_0) + h\,f'(x_0) + \frac{h^2}{2!}f''(x_0) + \cdots + \frac{h^n}{n!}\,f^n(x_0) + R_n(x)$$

$$Lt_{h\to 0}\frac{f(x_0+h)-f(x_0)}{h} = f'(x_0)$$

```
x=-2*pi:0.01:2*pi;
h=0.01;
f=sin(x);
g= (sin(x+h)-sin(x))./h;
plot(x,f,'-o');
hold on;
plot(x,g,'d');
legend 'sin(x)' 'cos(x)'
z=zeros(1,length(x));
hold on;
plot(x,z,'k','linewidth',3);
```



$$f(x_0 + h) = f(x_0) + h\, f'(x_0) + \frac{h^2}{2!} f''(x_0) + \cdots + \frac{h^n}{n!}\, f^n(x_0) + R_n(x)$$

We can use central differences as well

$$f(x + h) = f(x) + h\, f'(x)$$

$$f(x - h) = f(x) - h\, f'(x)$$

$$\frac{f(x + h) - f(x - h)}{2h} = f'(x)$$

Here we can obtain second order of accuracy as h^2 term will be cancelled.

Discretization & round-off errors

- Suppose the exact solution at a given value of x_n is ($y=F(x_n)$), the (accumulated) discretization error is defined as

$$E_n = F(x_n) - y_n$$

  - Caused by
    - approximate formula to estimate y_{n+1}
    - Input data at start of step do not necessarily correspond to exact soln
- Accumulated round-off error is defined as

$$R_n = y_n - Y_n$$

  - Where Y_n is the value we actually calculate after round-off rather than the true value y_n
- So the absolute value of the total error is given by the inequality $|F(x_n) - Y_n| \leq |E_n| + |R_n|$

Problems with the forward Euler method

- This method relies upon the derivative at the *beginning* of each interval to predict forward to the end of each interval
- Any errors in the solution tend to get amplified quickly
- Calculated solution quickly diverges away from the true solution as the error grows
- We can precisely calculate the error in a single step as follows

Local discretization error in Euler Method

The easiest way to calculate the local error is to use a Taylor expansion:

$$y_{i+1} = y_i + \left.\frac{dy}{dx}\right|_{x_i,y_i}(x_{i+1}-x_i) + \frac{1}{2!}\left.\frac{d^2y}{dx^2}\right|_{x_i,y_i}(x_{i+1}-x_i)^2 + \frac{1}{3!}\left.\frac{d^3y}{dx^3}\right|_{x_i,y_i}(x_{i+1}-x_i)^3 + \ldots$$

$$\boxed{y_{i+1} = y_i + f(x_i,y_i)(x_{i+1}-x_i)} + \frac{1}{2!}f'(x_i,y_i)(x_{i+1}-x_i)^2 + \frac{1}{3!}f''(x_i,y_i)(x_{i+1}-x_i)^3 + \ldots$$

These three terms correspond to Euler's method $y_{i+1} = y_i + f(x_i,y_i)h$

The local discretization error in the approximation at one step is thus given by

$$e_i = \frac{f'(x_i,y_i)}{2!}h^2 + O(h^3)$$

$$\therefore\ e_i \propto h^2$$

Improved Euler Method

- **Similar in concept to the trapezoid rule in quadrature, by utilizing an average of the start and end point derivatives we can better approximate the change in y over the interval**

$$\tilde{y}_1 = y_0 + hf(x_0, y_0), \qquad \tilde{f}_1 = f(x_1, \tilde{y}_1) \quad \text{Prediction step}$$

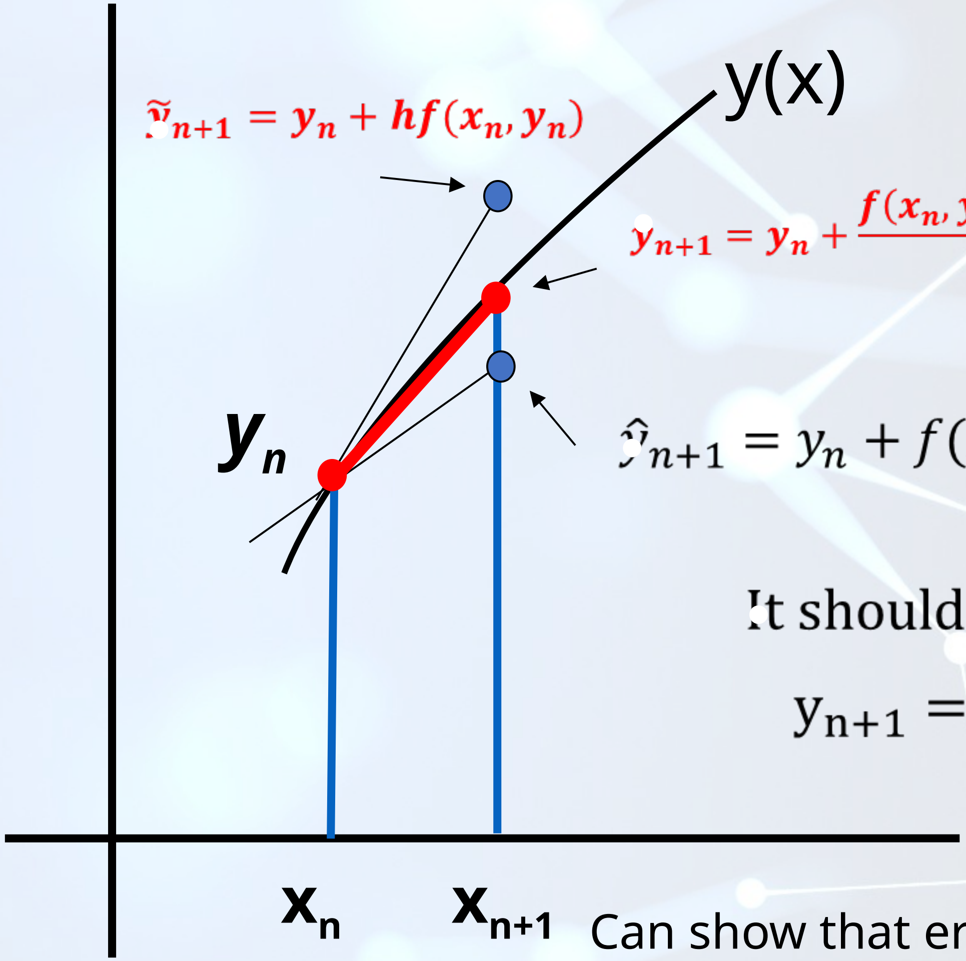
$$y_1 = y_0 + \frac{h}{2}(f_0 + \tilde{f}_1), \qquad f_1 = f(x_1, y_1) \quad \text{Correction step}$$

$$\tilde{y}_2 = y_1 + hf_1, \qquad \tilde{f}_2 = f(x_2, \tilde{y}_2) \quad \text{Prediction step}$$

$$y_2 = y_1 + \frac{h}{2}(f_1 + \tilde{f}_2), \qquad f_2 = f(x_2, y_2) \quad \text{Correction step}$$

- **This is the simplest example of so called "*Predictor-Corrector*" methods, (which includes Runge-Kutta methods)**

Improved Euler method graphically



$$\tilde{y}_{n+1} = y_n + hf(x_n, y_n)$$

$$y_{n+1} = y_n + \frac{f(x_n, y_n) + f(x_{n+1}, \tilde{y}_{n+1})}{2} h$$

$$\hat{y}_{n+1} = y_n + f(x_{n+1}, \tilde{y}_{n+1})h$$

It should be clear that

$$y_{n+1} = \frac{1}{2}(\tilde{y}_{n+1} + \hat{y}_{n+1})$$

Can show that errors in this method are also

$$e_i \propto O(h^3) \quad E_n \propto O(h^2)$$

This method is more accurate than the Euler method, in the sense that it tends to make a smaller ***error*** $E = |x(t_n) - x_n|$ for a given ***stepsize*** Δt. In both cases, the error $E \to 0$ as $\Delta t \to 0$, but the error decreases *faster* for the improved Euler method. One can show that $E \propto \Delta t$ for the Euler method, but $E \propto (\Delta t)^2$ for the improved Euler method (Exercises 2.8.7 and 2.8.8). In the jargon of numerical analysis, the Euler method is first order, whereas the improved Euler method is second order.

Methods of third, fourth, and even higher orders have been concocted, but you should realize that higher order methods are not necessarily superior. Higher order methods require more calculations and function evaluations, so there's a computational cost associated with them. In practice, a good balance is achieved by the ***fourth-order Runge–Kutta method***. To find x_{n+1} in terms of x_n, this method first requires us to calculate the following four numbers (cunningly chosen, as you'll see in Exercise 2.8.9):

Overall Target

- *Numerical s*
  differential
- *Modelling*
  chaos in p

Me after every single differential equations test.