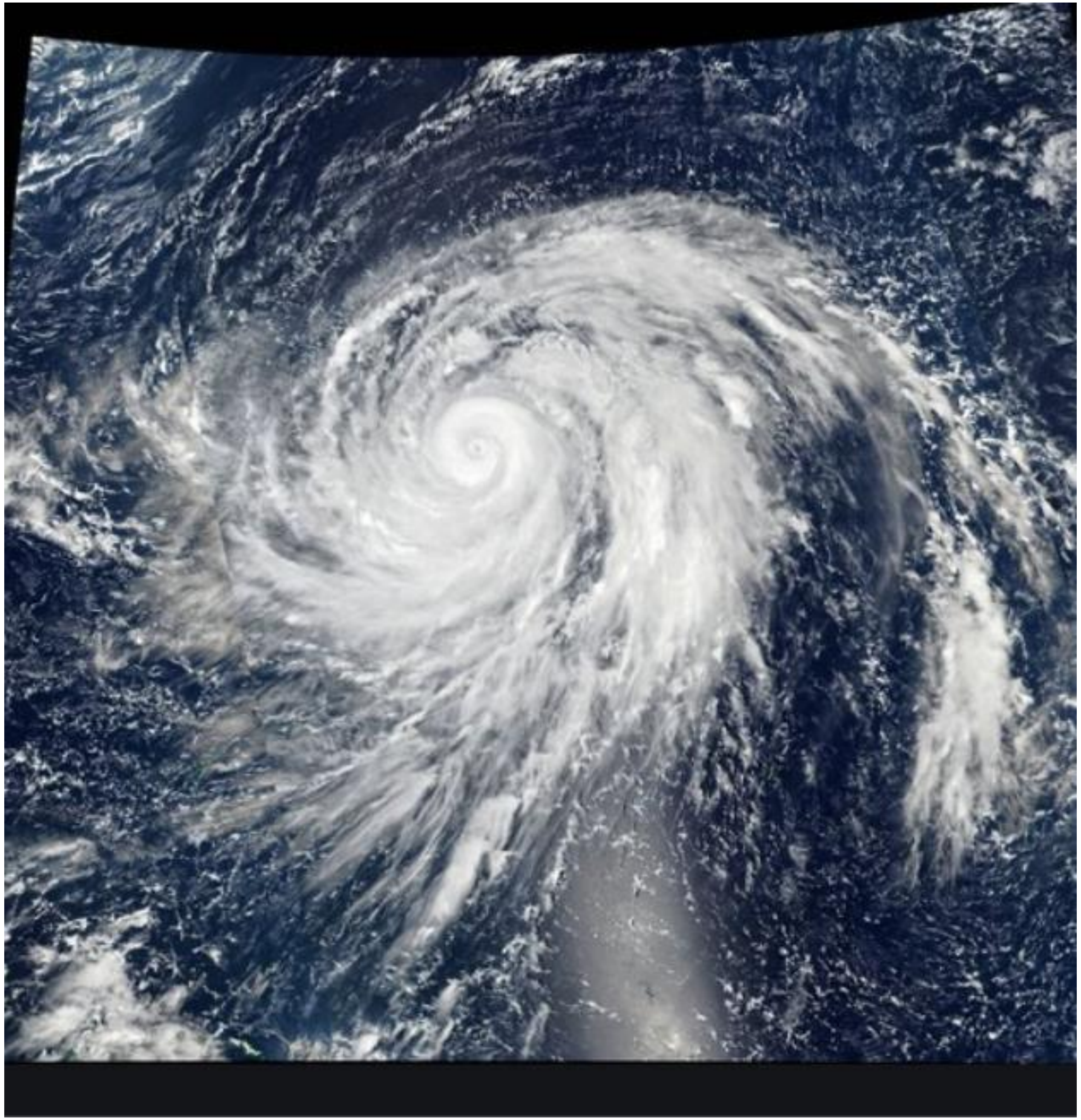
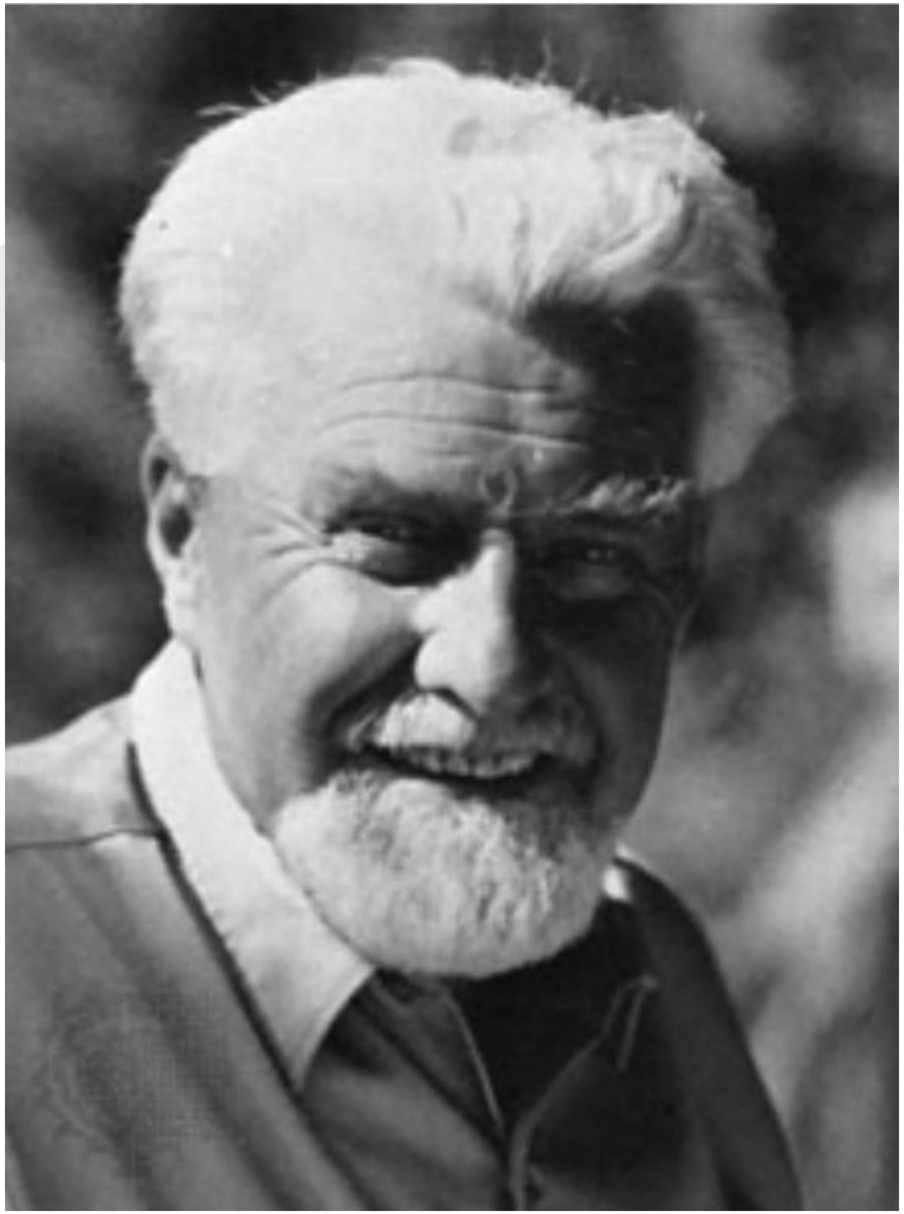
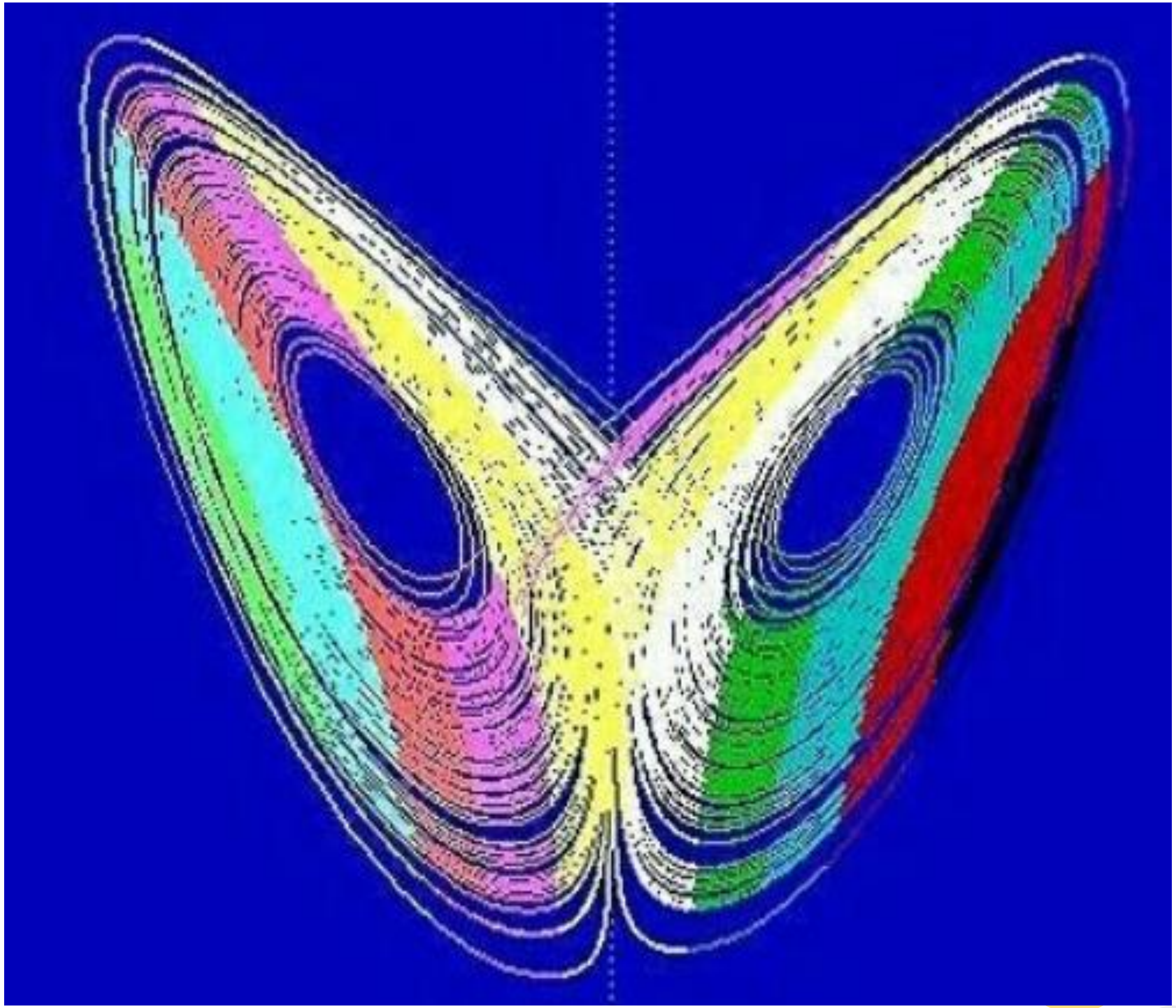
로렌츠 방정식,

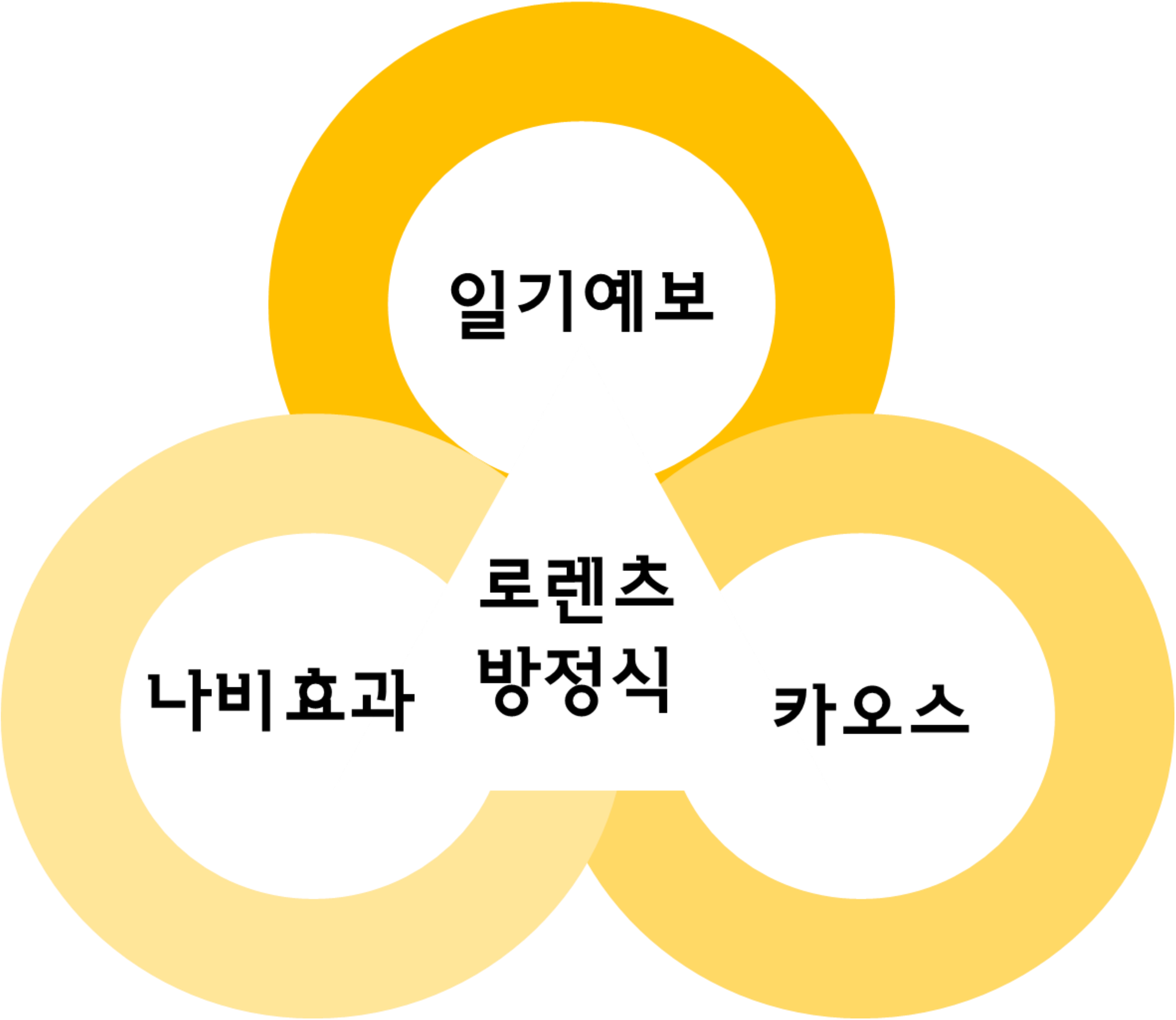
수치해석에서 딥러닝까지




내가 날갯짓을 하면
미국에서 토네이도가
일어나지 !
미국






일기예보
로렌츠
방정식
나비효과
카오스

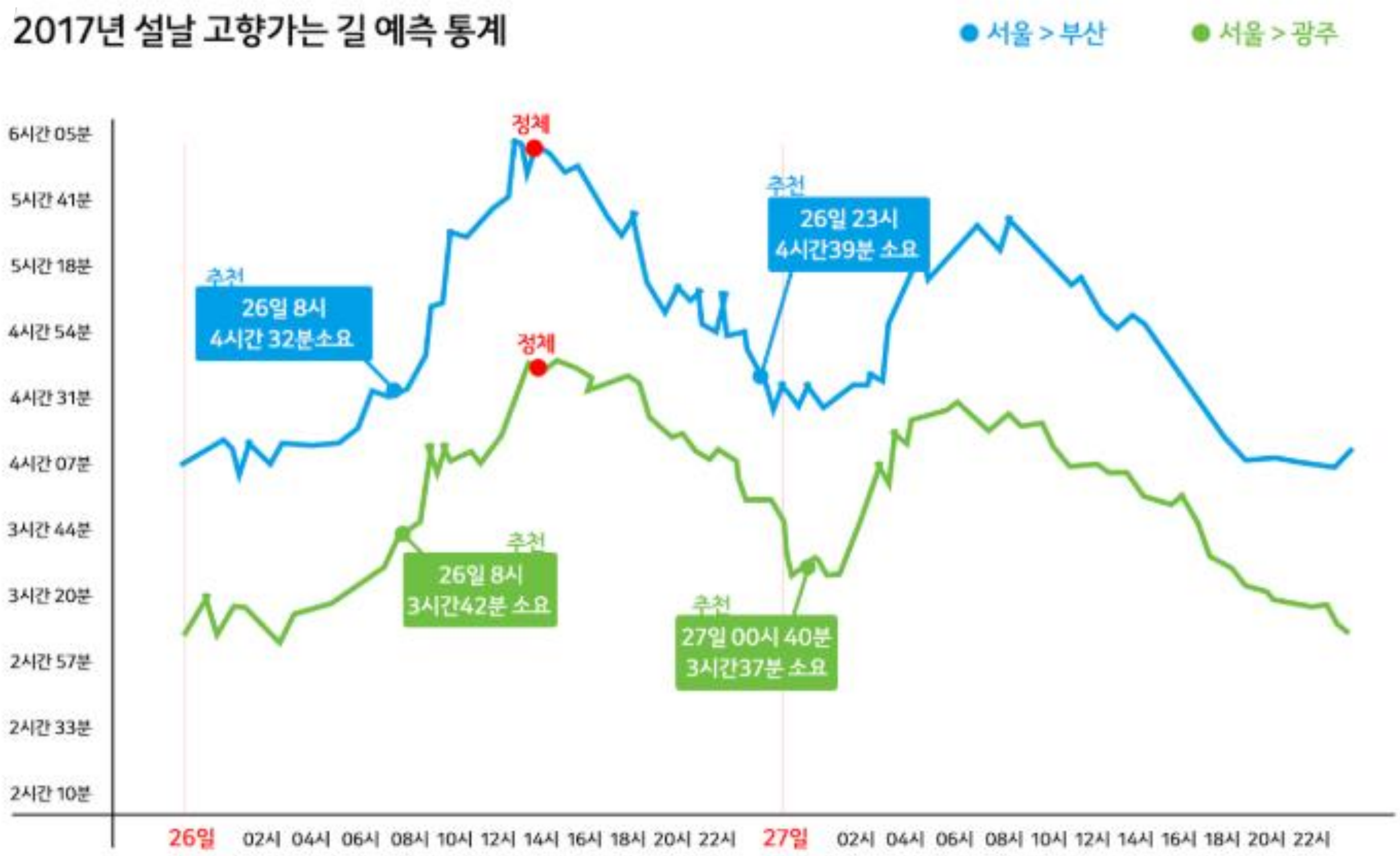
목차

Ⅰ. 연구 필요성 및 목적

연구 필요성 및 목적

연구 필요성 및 목적

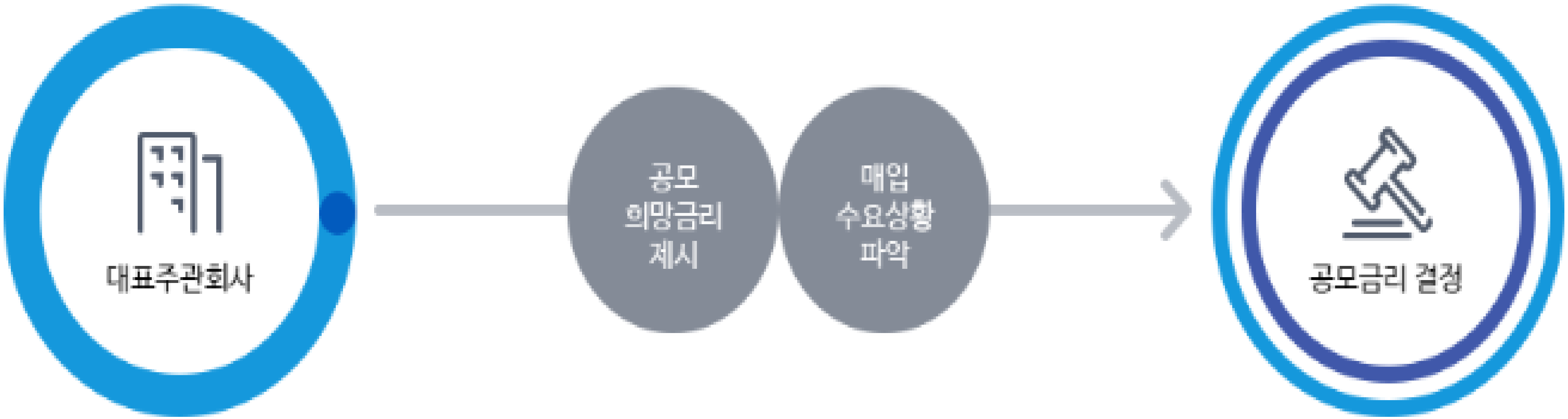
연구 필요성 및 목적



2017년 설날 고향가는 길 예측 통계
서울 > 부산
서울 > 광주
6시간 05분
5시간 41분
5시간 18분
4시간 54분
4시간 31분
4시간 07분
3시간 44분
3시간 20분
2시간 57분
2시간 33분
2시간 10분
정체
정체
추천
26일 8시
4시간 32분소요
추천
26일 23시
4시간39분 소요
추천
26일 8시
3시간42분 소요
추천
27일 00시 40분
3시간37분 소요
26일
02시 04시 06시 08시 10시 12시 14시 16시 18시 20시 22시
27일
02시 04시 06시 08시 10시 12시 14시 16시 18시 20시 22시

연구 필요성 및 목적

✓ 금융투자 – 수요예측



연구 필요성 및 목적

$$F\left(x, y, \frac{dy}{dx}, \cdots, \frac{d^n y}{dx^n}\right) = 0$$

종속변수

$$\frac{d^2 y}{dx^2} + 8y = 0$$

독립변수

Ⅱ. 배경이론 소개

배경이론

미분방정식 유한차분법 딥러닝

배경이론

✓ **미분방정식**

Differential equation Derivative form

$$y'' + y' - y = -2x \qquad \cdots (2.1)$$

Differential form

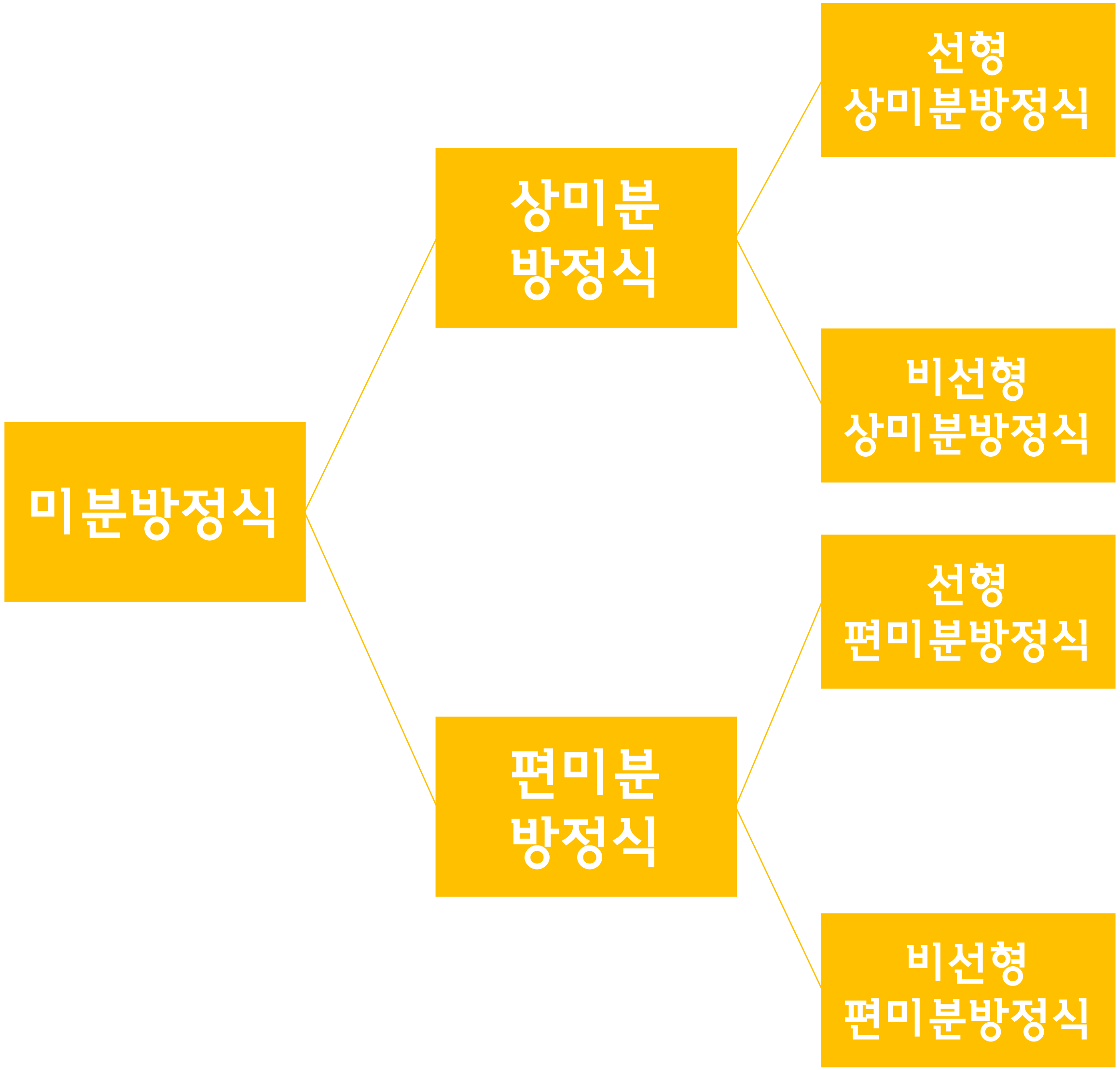
$$\frac{d^2y}{dx^2} + \frac{dy}{dx} - y = -2x \qquad \cdots (2.2)$$

Differential Operator

$$D^2y + Dy - y = -2x \qquad \cdots (2.3)$$

배경이론

✓ 미분방정식


미분방정식
상미분
방정식
편미분
방정식
선형
상미분방정식
비선형
상미분방정식
선형
편미분방정식
비선형
편미분방정식

배경이론

✓ 미분방정식

Ordinary Differential Equation

$$\frac{d^2y}{dx^2} + \frac{dy}{dx} - y = -2x \qquad \cdots (3.1)$$

Partial Differential Equation

$$\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial x^2} = 0 \qquad \cdots (3.2)$$

배경이론

✓ **초기값 문제 (Initial Value Problem, IVP)**

$$\begin{cases} x' = 3t^2 - 4t^{-1} + (1+t^2)^{-1} \\ x(5) = 17 \end{cases}$$

$$\mathrm{x(t)} = t^3 - 4lnt + arctant + C$$

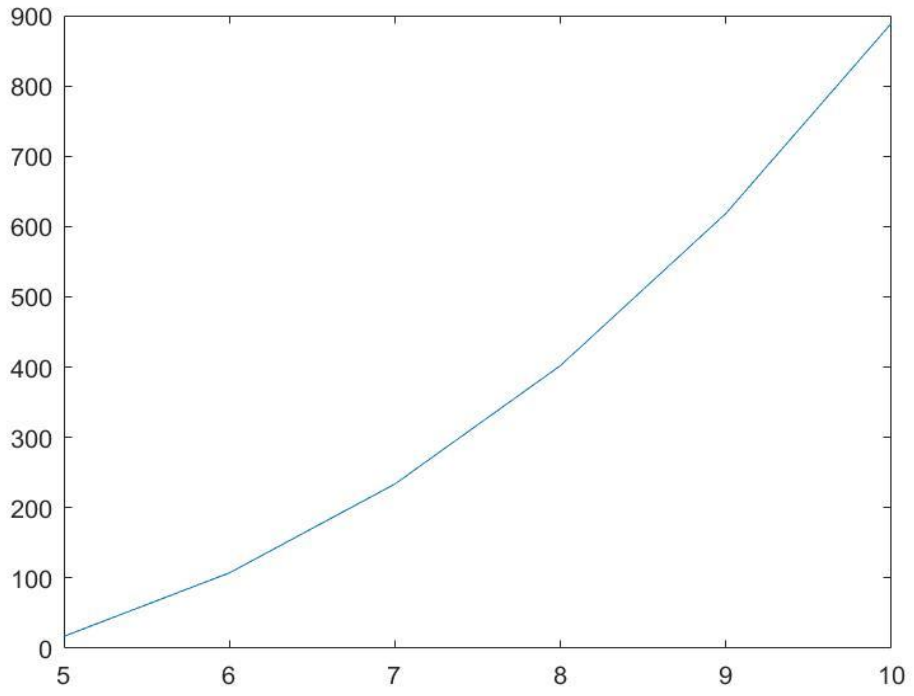
$$\mathrm{x}(5) = 17$$

$$\mathrm{C} = 4\ln(5) - \arctan(5) - 108$$

배경이론

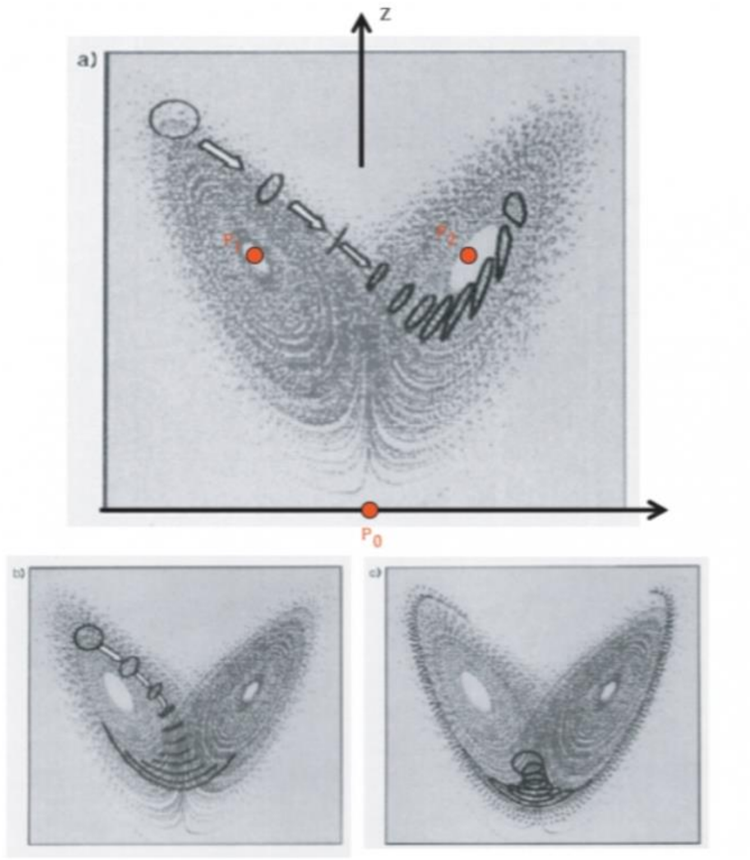
✓ **초기값 문제 (Initial Value Problem, IVP)**

```
function dydt = f(t,y)
dydt = 3*t^2 - 4*t^(-1) +(1+t^2)^(-1)

[t y] = ode45(@dydt, [5:10],17)
plot(t,y)
```



배경이론

✓ 로렌츠방정식(Lorenz equation)



배경이론

✓ **로렌츠방정식(Lorenz equation)**

$$\frac{dx}{dt} = -\sigma x + \sigma y$$

$$\frac{dy}{dt} = -xz + \rho x - y$$

$$\frac{dz}{dt} = xy - \beta z$$

배경이론

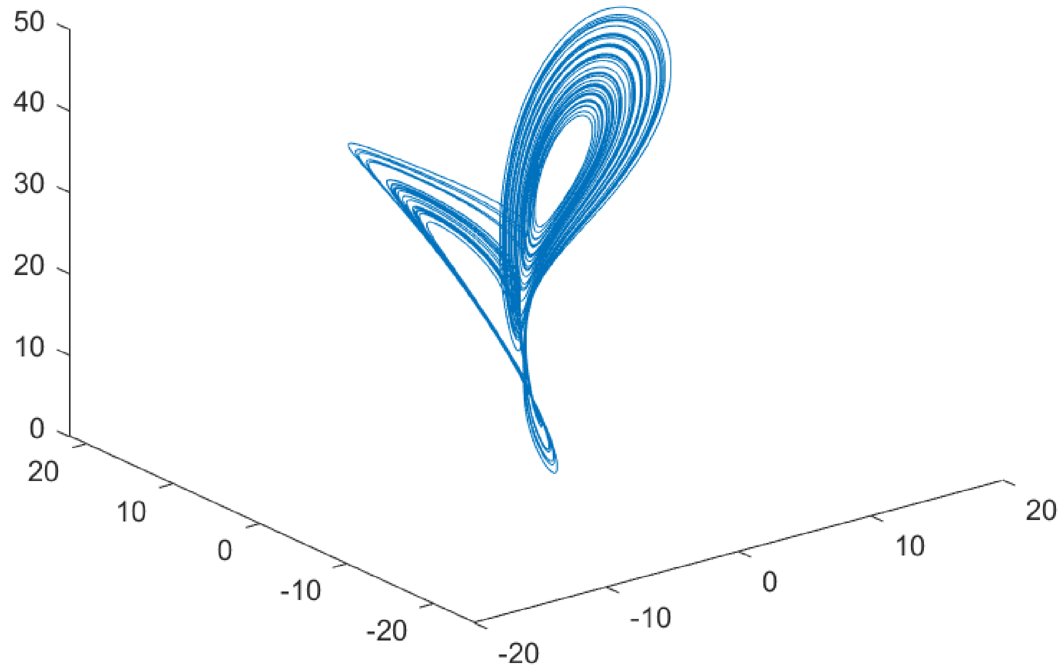
✓ **로렌츠방정식(Lorenz equation)**

```
Beta = [10;28;8/3];
x0 = [ 0; 1; 20];
dt = 0.001;
tspan = dt:dt:50

[t,x] = ode45(@(t,x) Lorenz(t,x,Beta),tspan,x0);
plot3(x(:,1),x(:,2),x(:,3));
```

```
function dx = Lorenz(t,x,Beta)
dx = [
Beta(1)*(x(2)-x(1)) ;
x(1)*(Beta(2) - x(3)) - x(2);
x(1)*x(2) - Beta(3) *x(3);
];
end
```

배경이론

✓ ODE45로 구현해본 로렌츠방정식(Lorenz equation)



배경이론

✓ 유한차분법(Finite Difference Method)

- 미분방정식의 미분을 유한한 차분식으로 근사시켜 해를 구하는 방법

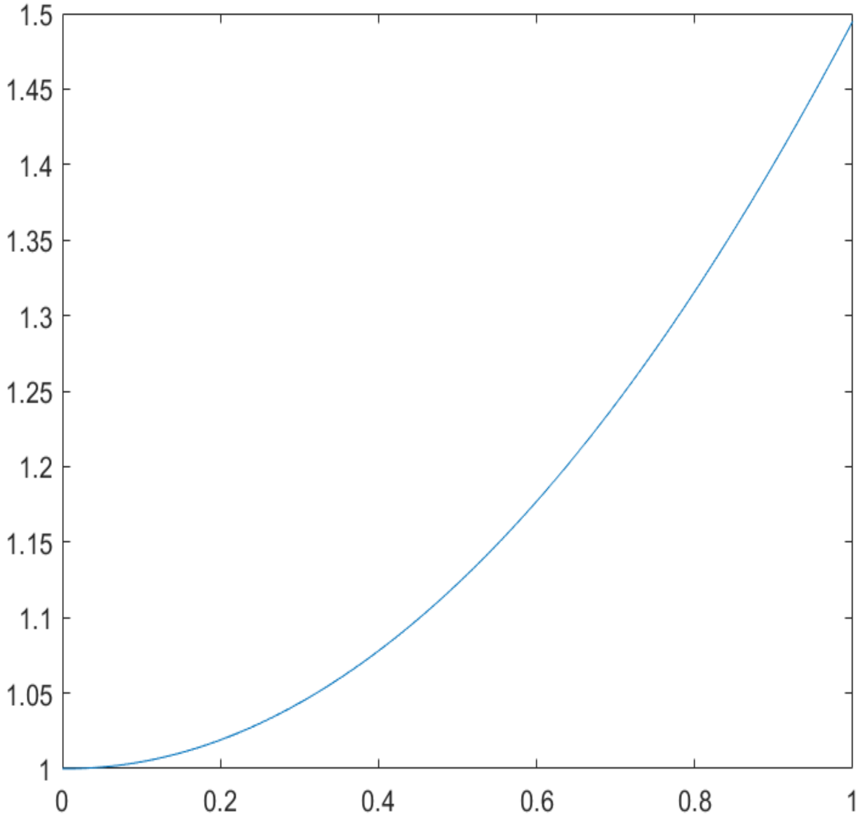
미분식	차분식
$\frac{du}{dx} = \lim_{\Delta x \to 0} \frac{\Delta u}{\Delta x}$	$\frac{\Delta u}{\Delta x} = \frac{u(x + \Delta x) - u(x)}{\Delta x}$

➡ 미분식은 무한의 개념 차분식은 유한의 개념!

배경이론

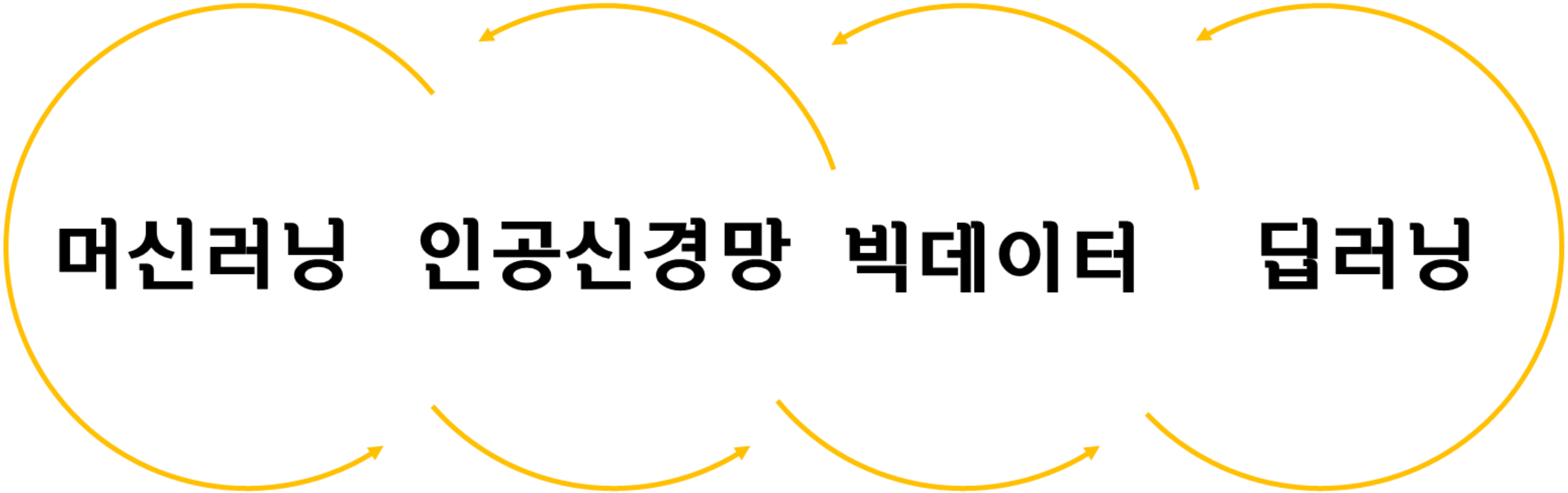
✓ **유한차분법을 이용한 미분방정식 $x' = t$ 의 풀이**

```
t = linspace(0,1,101); dt = t(2)-t(1)
x =zeros(1,101);
x(1) = 1;
for n = 1:100
    x(n+1) = x(n) + dt*t(n);
end
plot(t,x)
```



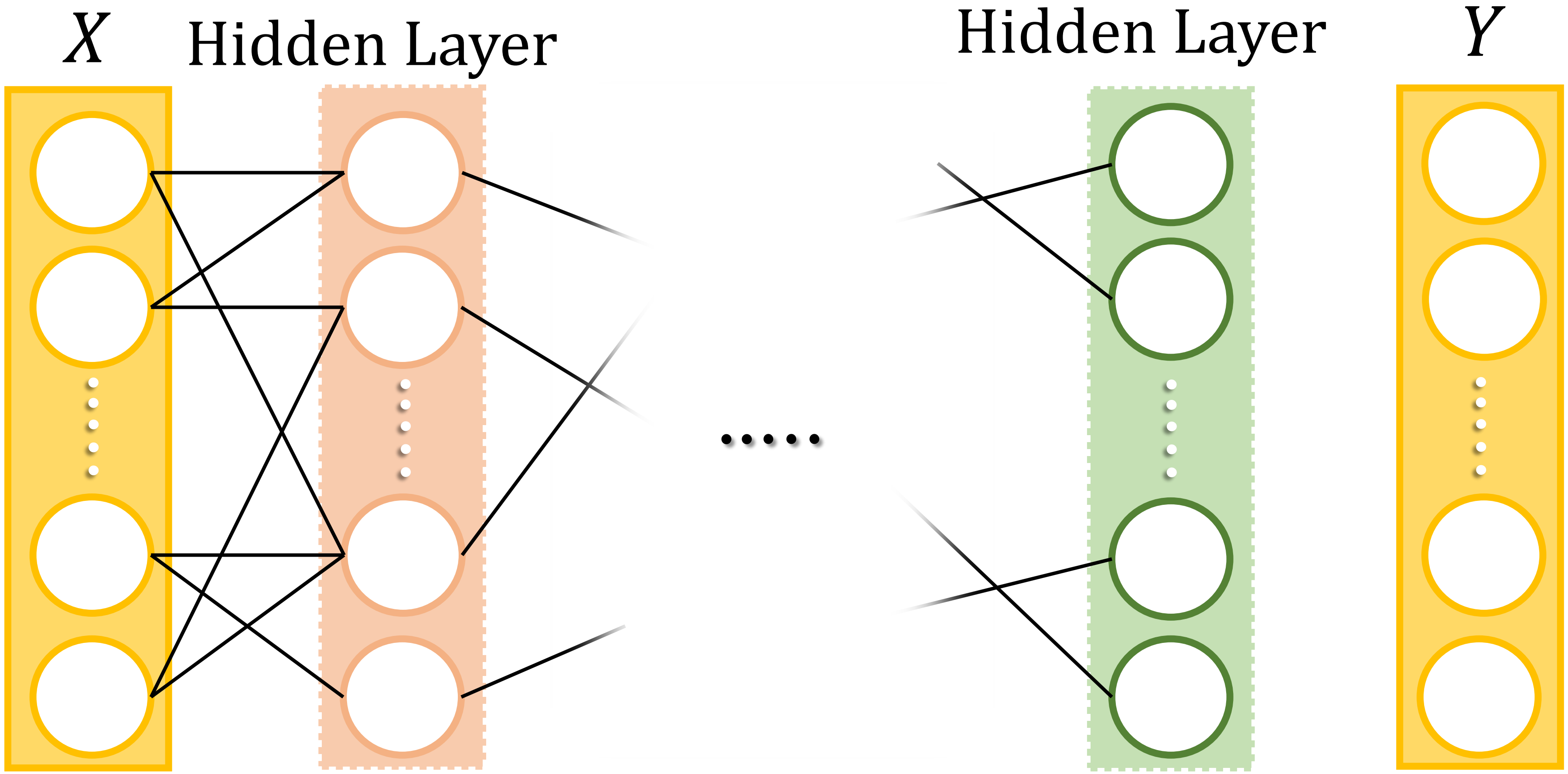
배경이론

✓ 딥러닝이란?



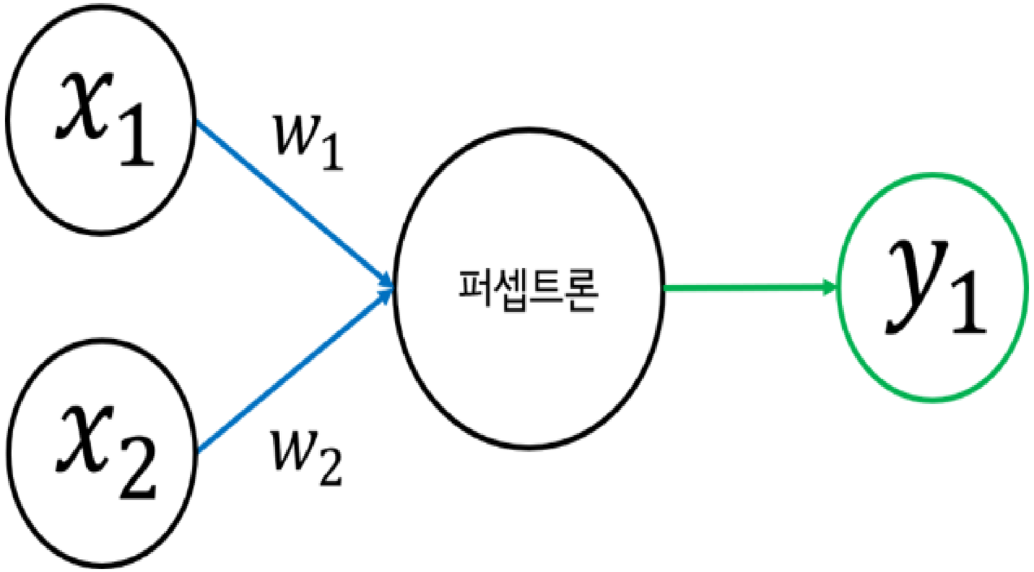
배경이론

✓ 딥러닝이란?



배경이론

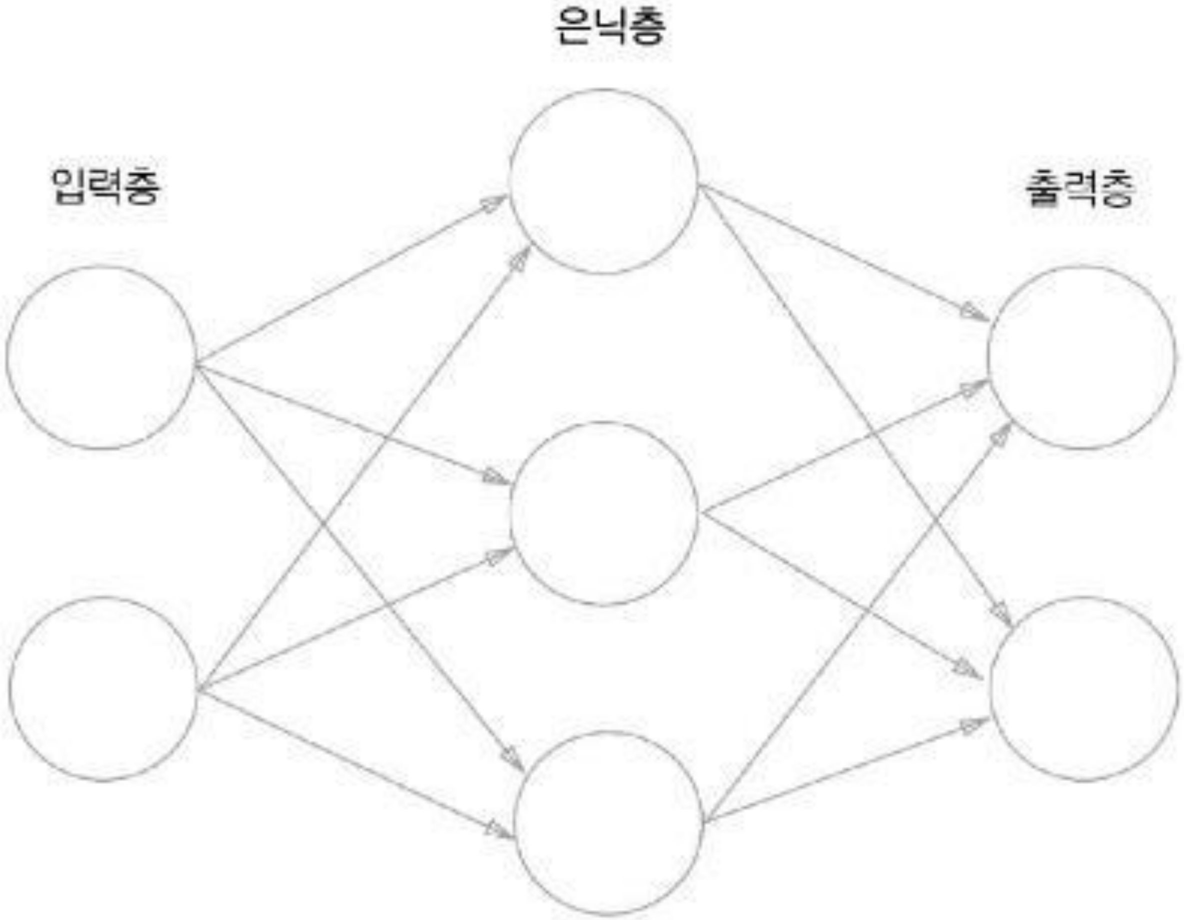
✓ **퍼셉트론**



$$y = \begin{cases} 0 & (x_1 w_1 + x_2 w_2 \le \theta) \\ 1 & (x_1 w_1 + x_2 w_2 > \theta) \end{cases}$$

배경이론

✓ 인공 신경망



배경이론

✓ 활성화 함수

❶ 시그모이드 함수

$$h(x) = \frac{1}{1 + e^{-x}}$$

❷ ReLU 함수

$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

배경이론

✓ 활성화 함수

Regression(회귀) → • 항등 함수

$$f(x) = x$$

Classification(분류) → • 소프트맥스 함수

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

Ⅲ. 로렌츠 방정식의 수치적 풀이법

로렌츠 방정식의 수치적 풀이법

✓ 오일러법

– 테일러 급수법 : 미분방정식의 해를 테일러 급수의 몇 개의 항으로 표현

$$x(t+h) = x(t) + hx'(t) + \frac{1}{2!}h^2x''(t) + \cdots + \frac{1}{m!}h^mx^{(m)}(t) + \cdots$$

이때, $\frac{1}{n!}h^nf^n(x)$ 까지의 항이 테일러 급수에 포함되었을 때를 n차 테일러 급수법이라 한다.

로렌츠 방정식의 수치적 풀이법

✓ 오일러법

– 오일러법 : 1차 테일러 급수법

$$\begin{cases} x' = f(t, x(t)) \\ x(a) = x_a \end{cases}$$

– 테일러 급수법의 처음 두 개의 항만 이용하면

$$x(t+h) = x(t) + hf(t, x(t))$$

로렌츠 방정식의 수치적 풀이법

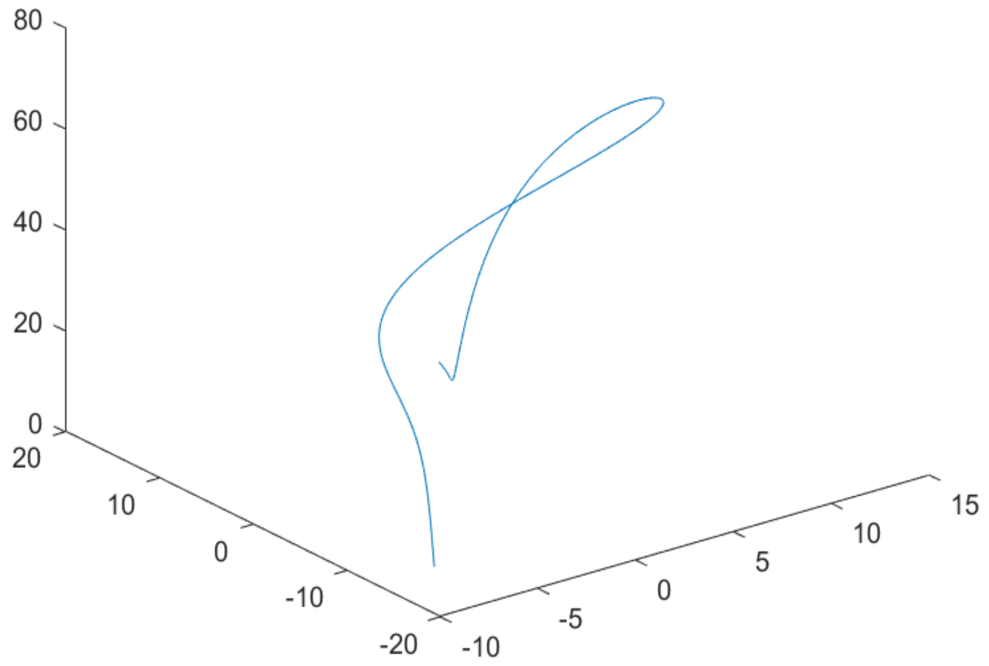
✓ 오일러법을 이용한 로렌츠 방정식 구현

```
clc; clear;
%Parameter
sigma = 10;
rho = 28;
beta = 8/3;

%Initial Data
x0 = [0;1;20];
xn = x0(1); yn = x0(2) ; zn = x0(3); x = x0;

%Forward Euler
t = 0; dt = 0.001;
maxT = 1;
```

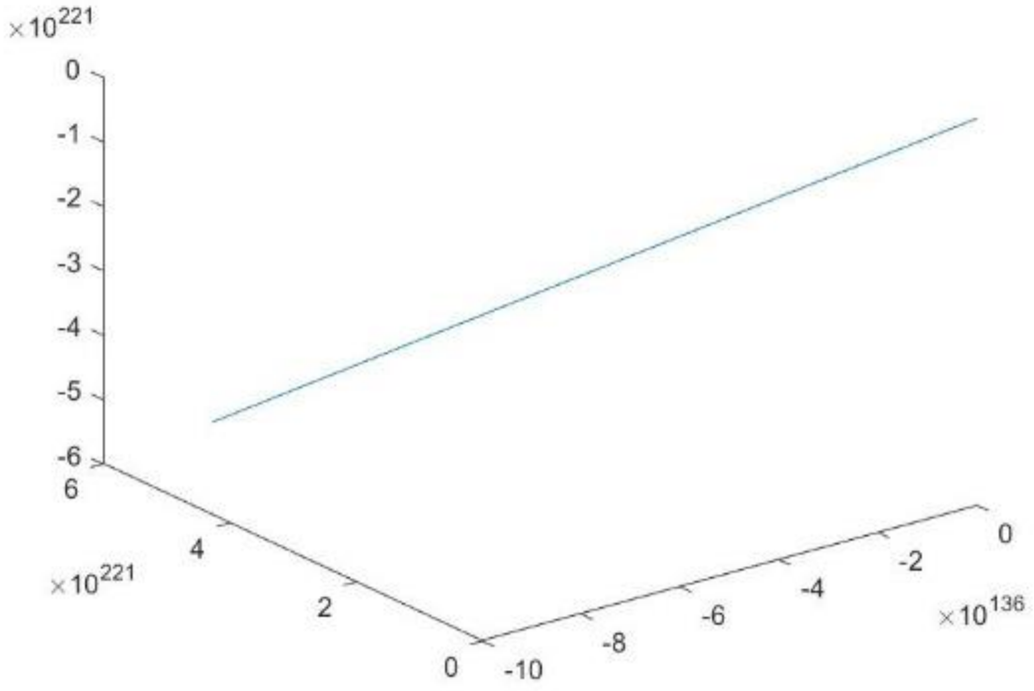
```
while true
    xnp1 = xn + dt*sigma*(yn - xn);
    ynp1 = yn + dt*(xn*(rho - zn) - yn);
    znp1 = zn + dt*(xn*zn - beta*zn);

    x = [x [xnp1; ynp1; znp1]];
    xn = xnp1 ; yn = ynp1; zn = znp1;
    t = t + dt;
    if t >= maxT
        break
    end
end
plot3(x(1,:),x(2,:),x(3,:));
```

로렌츠방정식의 수치적 풀이법

✓ maxT = 1



✓ maxT = 10



로렌츠방정식의 수치적 풀이법

✓ 룽게쿠타법

● RK2

$$x(t+h) = x(t) + \frac{1}{2}(K_1 + K_2)$$

(

여기서 K_1, K_2는 $\begin{cases} K_1 = hf(t,x) \\ K_2 = hf(t+h, x+K_1) \end{cases}$ 이다.)

● RK4

$$x(t+h) = x(t) + \frac{1}{6}(K_1 + 2K_2 + 2K_{3+K_4})$$

(여기서 K_1, K_2, K_3, K_4는

$\begin{cases} K_1 = hf(t,x) \\ K_2 = hf(t+\frac{1}{2}h, x+\frac{1}{2}K_1) \end{cases}$ $\begin{cases} K_3 = hf(t+\frac{1}{2}h, x+\frac{1}{2}K_2) \\ K_4 = hf(t+h, x+K_3) \end{cases}$ 이다.)

로렌츠 방정식의 수치적 풀이법

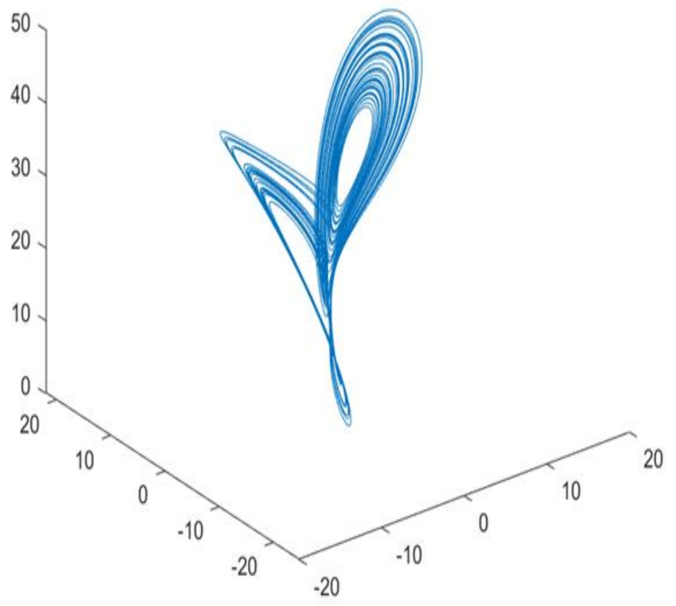
✓ RK4를 이용한 로렌츠 방정식의 풀이

```
clc; clear;
% Parameter Setup
sigma = 10;
rho = 28;
beta = 8/3;

% Initial data
x0 = [0; 1; 20];
xn = x0(1); yn = x0(2); zn = x0(3);
x  = x0;

% Forward Euler
t = 0; dt = 0.001;
maxT = 50;

Lorenz = @(t,x)([sigma*(x(2)-x(1)); ...
                 x(1)*(rho-x(3)) - x(2); ...
                 x(1)*x(2) - beta*x(3)]);
xnp1 = xn; ynp1 = yn; znp1 = zn;
x_old = [xn; yn; zn];
```

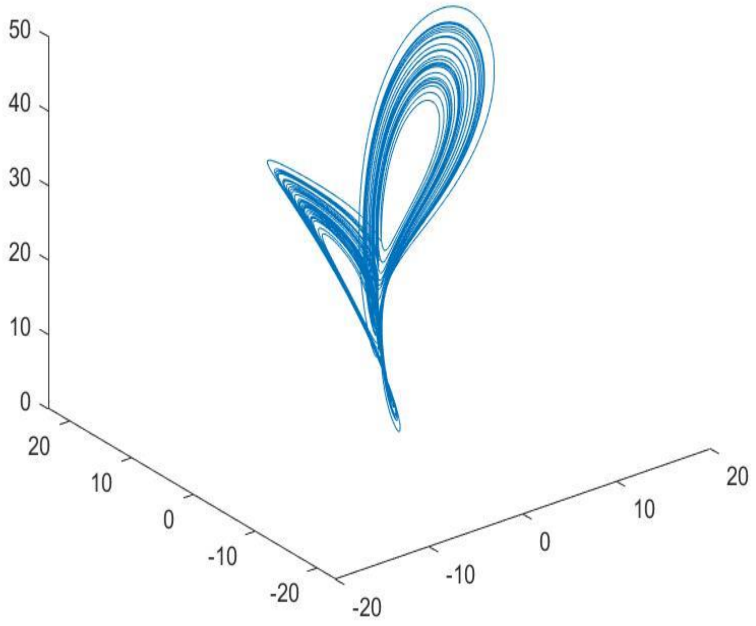
```
while true
    k1 = Lorenz(t,x_old);
    k2 = Lorenz(t+0.5*dt,x_old+0.5*k1*dt);
    k3 = Lorenz(t+0.5*dt,x_old+0.5*k2*dt);
    k4 = Lorenz(t+dt,x_old+k3*dt);

    % Main Uptate
    x_new = [xnp1; ynp1; znp1];
    x_new = x_old + dt*(k1+2*k2+2*k3+k4)/6;

    x = [x x_new];

    x_old = x_new;

    t = t + dt;
    if t >= maxT
        break;
    end
end

plot3(x(1,:), x(2,:), x(3,:));
```

로렌츠방정식의 수치적 풀이법

✓ ODE45로 구현해본 로렌츠방정식(Lorenz equation)



✓ RK4를 이용한 로렌츠 방정식의 풀이



Ⅳ. 딥러닝 기반의 새로운 접근법

딥러닝 기반의 새로운 접근법

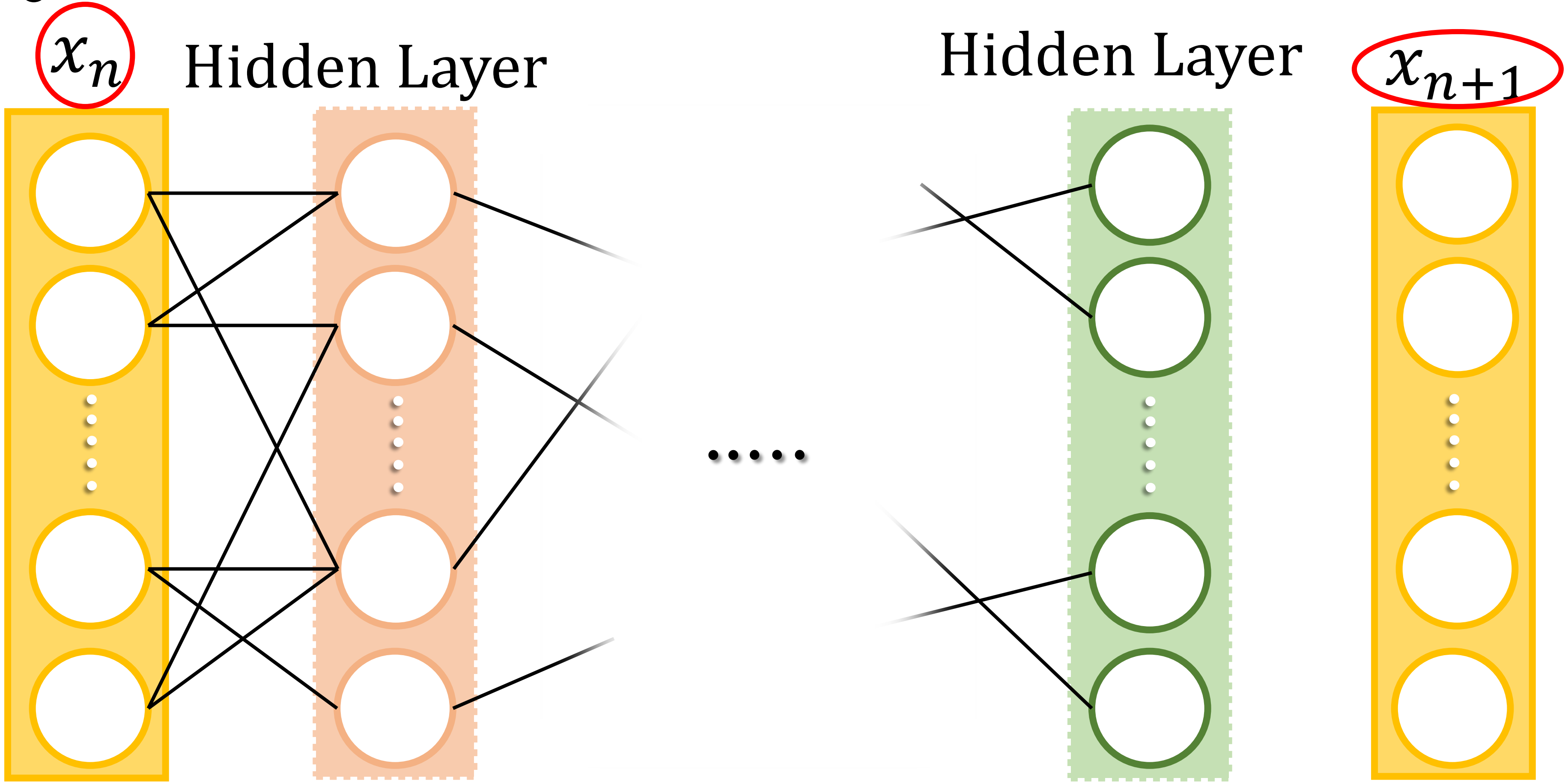
✓ **새로운 접근**

$$x^{n+1} = x^n + \Delta t \sigma (y^n - x^n)$$
$$y^{n+1} = y^n + \Delta t (x^n (\rho - z^n) - y^n)$$
$$z^{n+1} = z^n + \Delta t (x^n y^n - \beta z^n)$$

$$\underrightarrow{x^n} = (x^n, y^n, z^n)$$
$$\underrightarrow{x^1} = f\left(\underrightarrow{x^0}\right)$$
$$\underrightarrow{x^2} = f\left(\underrightarrow{x^1}\right)$$
$$\vdots$$
$$\underrightarrow{x^{n+1}} = f(\underrightarrow{x^n})$$

딥러닝 기반의 새로운 접근법

✓ 딥러닝의 목적



딥러닝 기반의 새로운 접근법

✓ 딥러닝 구현

학습 데이터 구축

```
clc; clear;
% Simulate Lorenz system
dt = 0.01; T = 8; t = 0:dt:T;
b = 8/3; sig = 10; r = 28;

Lorenz = @(t,x)([sig*(x(2)-x(1)); ...
                 x(1)*(r - x(3)) - x(2); ...
                 x(1)*x(2) - b*x(3)]);

ode_options = odeset('RelTol', 1e-10, 'AbsTol', 1e-11);
```

```
input = []; output = [];
for j = 1:100 % traninig trajectories
    x0 = 30*(rand(3,1)-0.5);
    [t,y] = ode45(Lorenz,t,x0);
    input = [input; y(1:end-1,:)];
    output = [output; y(2:end,:)];
    plot3(y(:,1),y(:,2),y(:,3)), hold on
    plot3(x0(1),x0(2),x0(3),'ro');
end

% Build a neural network for Lorenz system
net = feedforwardnet([10 10 10]);
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'radbas';
net.layers{3}.transferFcn = 'purelin';
net = train(net, input.', output.');
```

딥러닝 기반의 새로운 접근법

✓ 딥러닝 구현

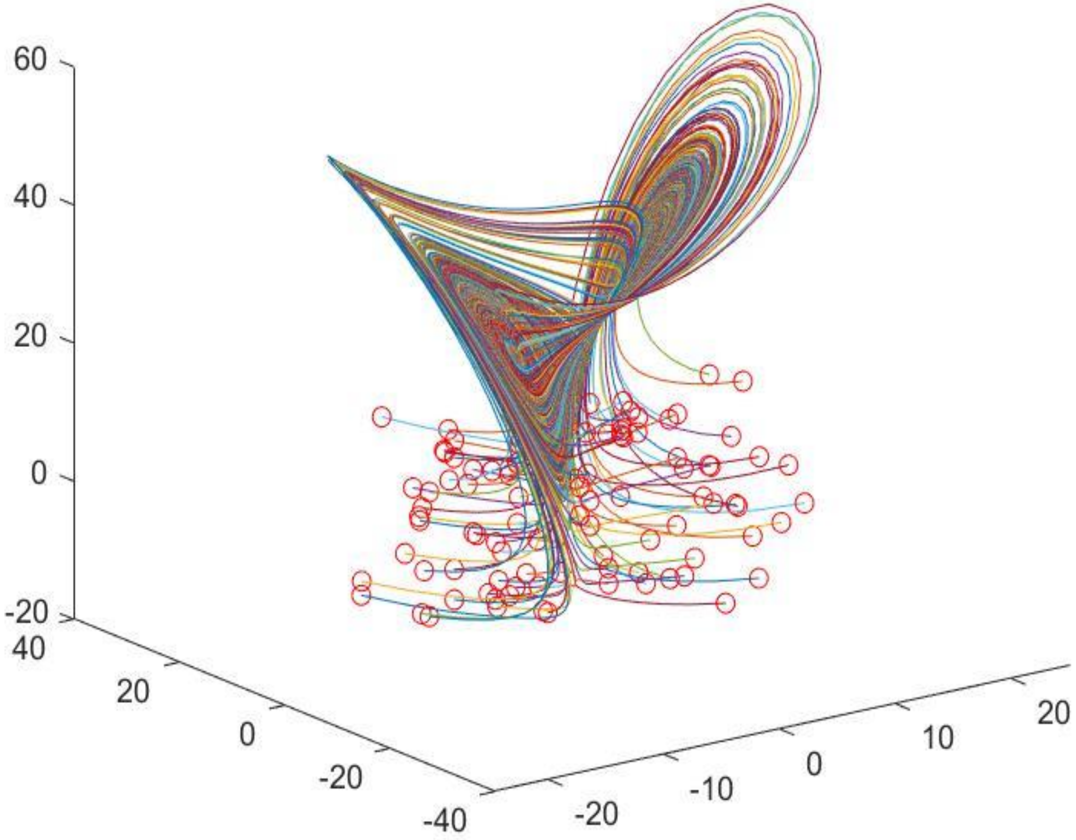
예측 모델링

```
% Neural network for prediction
xtmp = x0;
ynn(1,:) = x0;
for jj = 2:length(t)
    y0 = net(x0);
    ynn(jj,:) = y0.'; x0=y0;
end

plot3(ynn(:,1),ynn(:,2),ynn(:,3),':')
hold on;
[t,ytrue] = ode45(Lorenz,t,xtmp);
plot3(ytrue(:,1), ytrue(:,2), ytrue(:,3));
```

Ⅴ. 결과

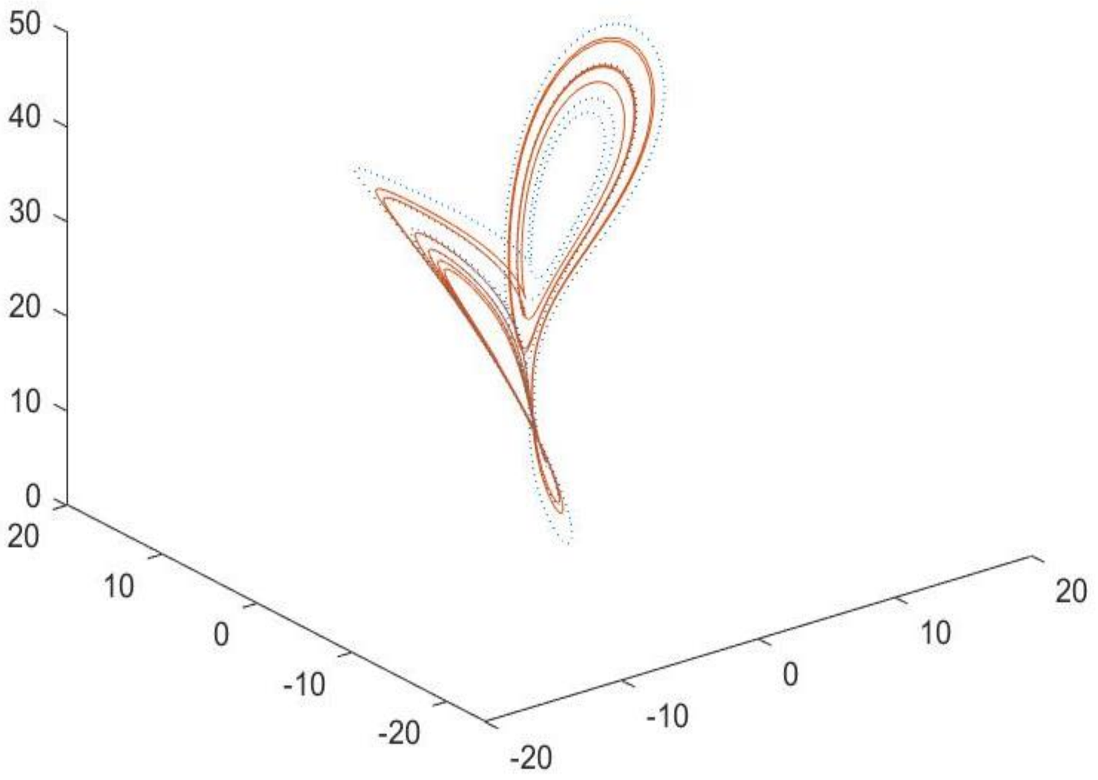
결과

✓ 딥러닝 구현결과

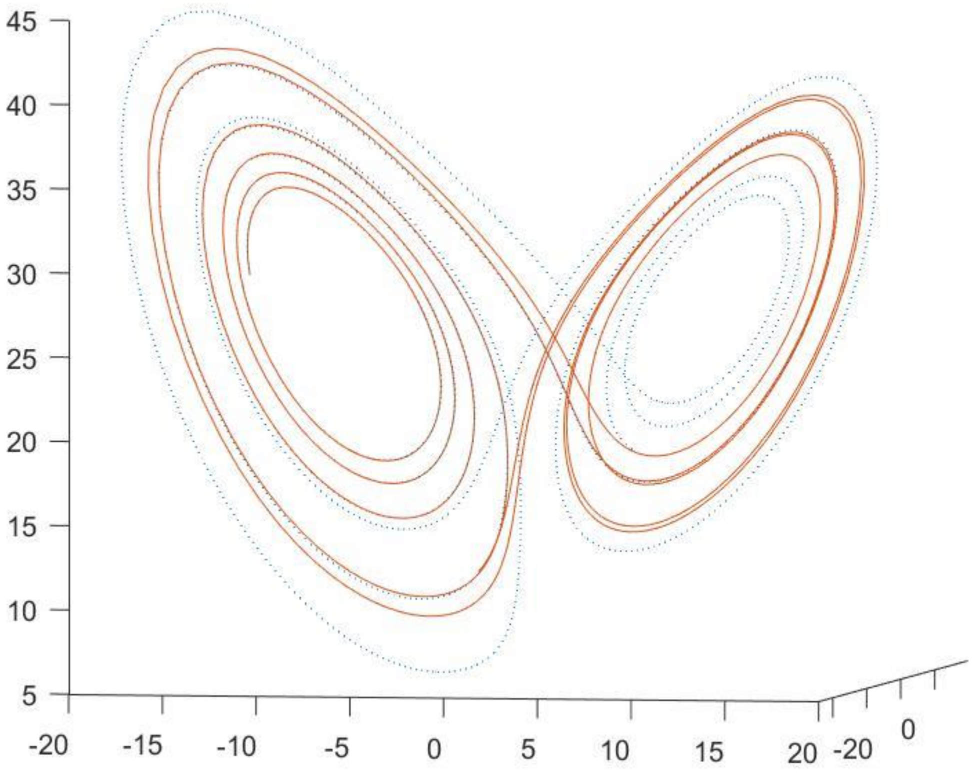


결과

✓ 딥러닝 구현결과



50
40
30
20
10
0
20
10
0
-10
-20
-20
-10
0
10
20



45
40
35
30
25
20
15
10
5
-20
-15
-10
-5
0
5
10
15
20
-20
0

Ⅵ. 결론

결론

- ✓ 오일러법 : 정확도가 떨어지는 한계
- ✓ 룽게-쿠타법 : 속도가 느리다는 한계
- ✓ 딥러닝 기반의 접근법에서의 한계 :

  ① 활성화 함수

  ② 적은 데이터로 예측도가 낮음

결론

구현된 그래프들의
비슷한 양상

의의 및 한계

- ✓ 수치해석적 접근
- ✓ 딥러닝
- ✓ 상미분방정식

Thank you