경희대학교
KYUNG HEE UNIVERSITY

Radioactive Material Dispersion Modeling using Physics Informed Neural Network

경희대학교

김기범
허균영

2020.12.17

Introduction

Background

- 대기 확산의 지배 방정식은 2차 편미분 방정식 형태를 가짐
- 미분 방정식의 해를 구하는 방법으로 해석적/수치적 방법의 다양한 방법이 사용될 수 있음
  - 해석적 방법: Gaussian Plume, Gaussian Puff
  - 수치적 방법: FDM (Finite Difference Method), Spectral analysis, CFD (Computational Fluid Dynamics), PINN (Physics Informed Neural Network)
- 현재 확산 거동을 예측하는 전산 코드는 대부분 해석적 방법인 Gaussian Plume/Puff 방법을 사용함 (MACCS, RASCAL, COSYMA, OSCAAR)
- 최근 인공신경망을 활용하여 물리적 모델을 학습하는 PINN 방법이 제시된 바 있음

Purpose

- PINN 방법을 이용한 확산 모델링 방법 제시
- 사례 연구를 통한 PINN 방법의 확산 모델 적용 가능성 확인

Methodology

▪ Air dispersion model

– 방사성 물질과 같은 오염물질의 대기 확산에 대해 시간 $t \geq 0$, 위치 (x, y, z)에서의 오염물질 농도는 **질량보존법칙**에 의해 유도됨

– **2차 편미분 방정식** 형태를 가짐

$$\frac{\partial C}{\partial t} + \nabla \cdot \vec{J} = S$$

Where,
$C(x, y, z, t)$: 농도 $[kg/m^3]$
$\vec{J}(x, y, z, t)$: mass flux $[kg/m^2 s]$
S: Source & Sink 항

$$\vec{J} = \vec{J}_D + \vec{J}_A$$
$$\vec{J}_D = -K\nabla C$$
$$\vec{J}_A = C\vec{u}$$

$K = diag(K_x, K_y, K_z)$ $[m^2/s]$: 확산 계수
$\vec{u} = (u_x, u_y, u_z)$ $[m/s]$: 바람장

$$\boxed{\frac{\partial C}{\partial t} + \nabla \cdot (C\vec{u}) = \nabla \cdot (K\nabla C) + S}$$

Methodology

▪ Air dispersion model

– Gaussian model

$$\frac{\partial C}{\partial t} + \nabla \cdot (C\vec{u}) = \nabla \cdot (K\nabla C) + S$$

- Source는 일정한 방출율을 갖는 $(0,0,h)$에 위치한 point source
  → $S = Q\delta(x)\delta(y)\delta(z-h)$
- 바람의 방향과 속도는 일정함 → $\vec{u} = (u,0,0)$
- 바람 방향 (x 방향)으로는 advection이 지배적이므로 바람 방향의 diffusion effect 무시
  → $K_x = 0$
- Time scale is long enough
- …

$$\bar{C}(x,y,z) = \frac{Q}{2\pi\sigma_y\sigma_z u}\exp\left(\frac{-y^2}{2\sigma_y^2}\right)\left(\exp\left(\frac{-(z-h)^2}{2\sigma_z^2}\right) + \exp\left(\frac{-(z+h)^2}{2\sigma_z^2}\right)\right)$$

➔ **Gaussian Plume model**

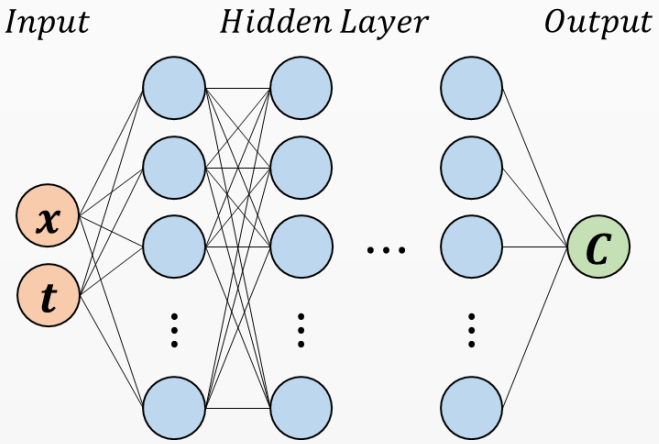
Methodology

$$\frac{\partial C}{\partial t} + \nabla \cdot (C\vec{u}) = \nabla \cdot (K\nabla C) + S$$

▪ PINN (Physics Informed Neural Network)

– 미분 방정식(물리적 모델)의 해를 인공 신경망으로 가정하고 이를 학습

$$\frac{\partial C}{\partial t} + u\frac{\partial C}{\partial x} = K\frac{\partial^2 C}{\partial x^2} \qquad (for\ 1D, when\ u\ and\ K\ is\ const.)$$



$assume\ that\ C = NN(x,t)$

PDE

$$C_t = -uC_x + KC_{xx}$$

$$\left.\begin{matrix} C(-\infty, t) = 0 \\ C(\infty, t) = 0 \end{matrix}\right\}$$ Boundary condition

Optimization

$$C_t + uC_x - KC_{xx} = f \quad \leftarrow \text{Loss function}$$

$$\underset{w}{\operatorname{argmin}}\left(MSE_f + (NN(-\infty, t) - 0)^2 + (NN(\infty, t) - 0)^2\right)$$

Loss function — Initial & boundary condition

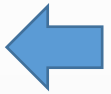
Methodology

▪ PINN (Physics Informed Neural Network)

① $assume\ that\ C = NN(x,t)$

```
self.C_model = tf.keras.Sequential()
self.C_model.add(tf.keras.layers.InputLayer(input_shape=(layers[0],)))
```

```
def net_C(self):
  X_C = tf.stack([self.x_C, self.t_C], axis=1)
  C = self.C_model(X_C)
  return C
```



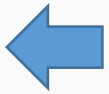
$boundary\ condition$
x, t

② $f = C_t + uC_x - KC_{xx}$

```
def net_f(self):
  with tf.GradientTape(persistent = True) as tape:
    tape.watch(self.x_f)
    tape.watch(self.t_f)
    X_f = tf.stack([self.x_f, self.t_f], axis=1)

    C = self.C_model(X_f)
    C_t = tape.gradient(C, self.t_f)
    C_x = tape.gradient(C, self.x_f)
    C_xx = tape.gradient(C_x, self.x_f)

  del tape
  return C_t + wind * C_x - K * C_xx
```



$randomly\ sampled$
x, t

③ $\underset{w}{\operatorname{argmin}}\left(MSE_f + (NN(-\infty,t) - 0)^2 + (NN(\infty,t) - 0)^2\right)$

```
def loss(self):
  f_pred = self.net_f()
  C_pred = self.net_C()
  return tf.reduce_mean(tf.square(self.C - C_pred)) + tf.reduce_mean(tf.square(f_pred))
```

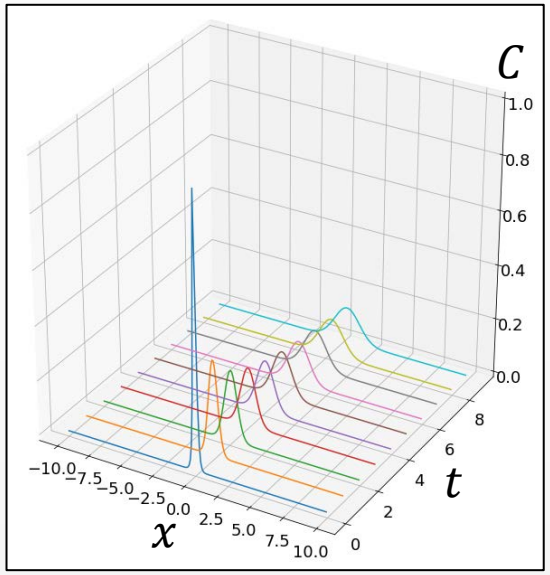
Result

▪ 1-D / constant wind & dispersion parameter

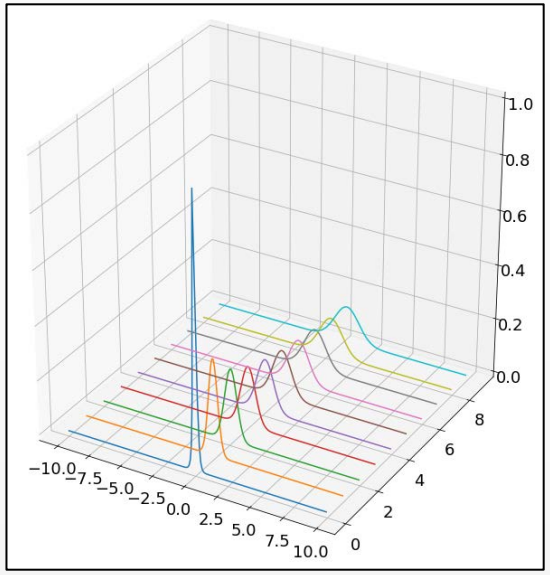
$$\frac{\partial C}{\partial t} + u\frac{\partial C}{\partial x} = K\frac{\partial^2 C}{\partial x^2} \qquad (for\ 1D, when\ u\ and\ K\ is\ const.)$$

– 초기값: $C(x,0) = \frac{1}{\cosh(10x)}$

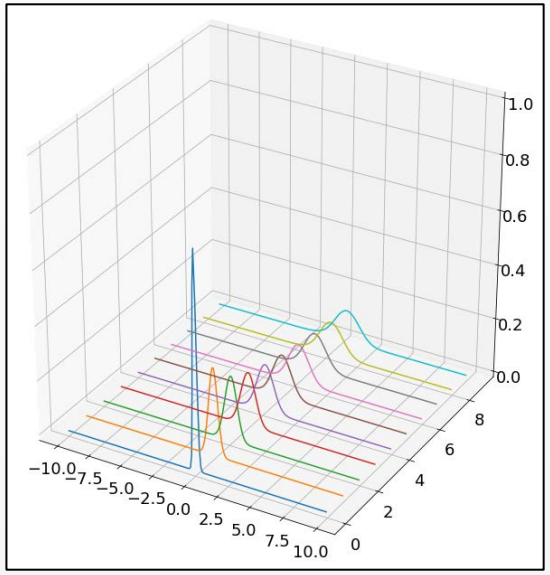
– $u = 5,\ K = 5$



<FDM>



<FFT>



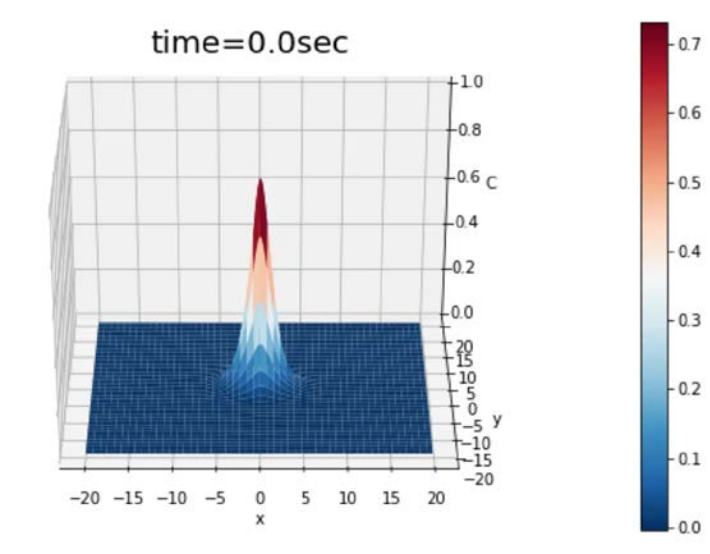
<PINN>

<평균 제곱근 편차(Root Mean Square Error>

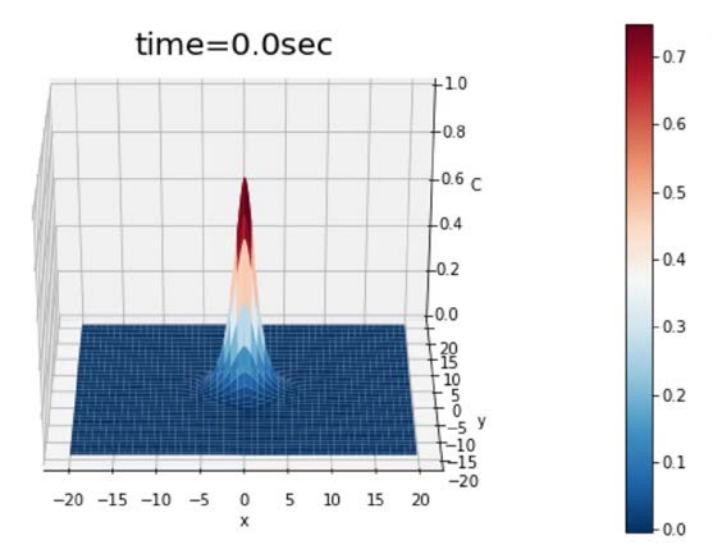
RMSE	FDM	FFT	PINN
FDM	-	1.11E-6	2.66E-5
FFT	1.11E-6	-	3.09E-5
PINN	2.66E-5	3.09E-5	-

Result

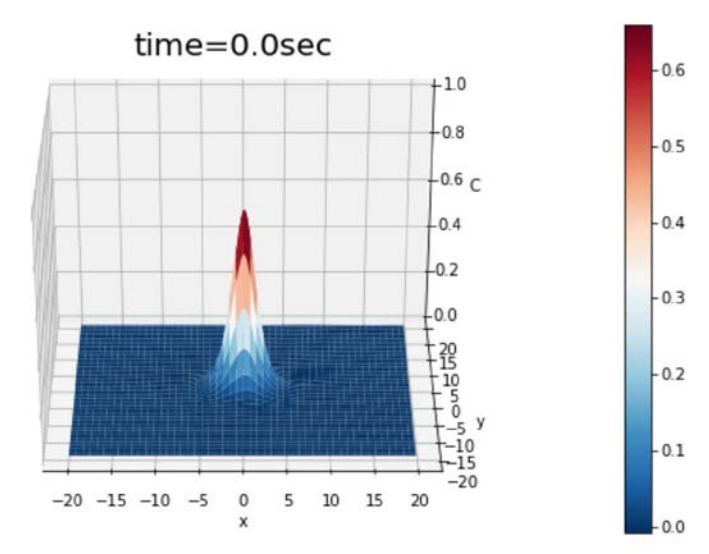
- **2-D**
  - 초기값: $C(x,0) = \frac{1}{\cosh\left(10\sqrt{x^2+y^2}\right)}$
  - $\vec{u} = (u,u), u = 2,5,10$
  - K: 거리에 따라 선형적으로 증가



$<u = 2>$



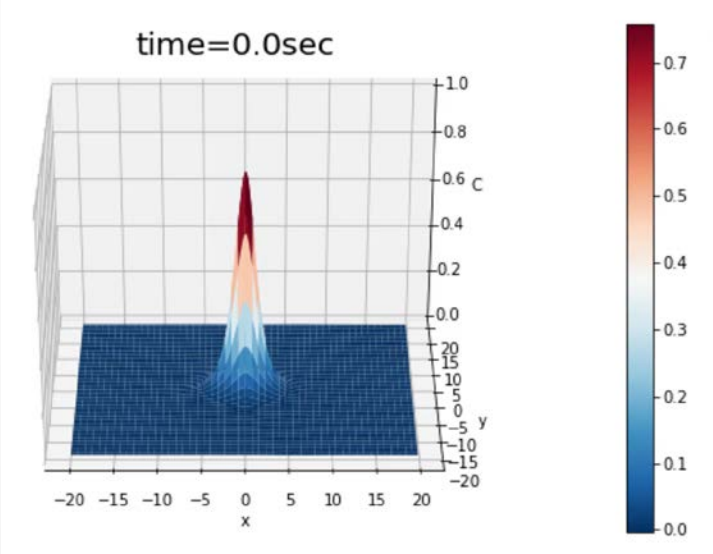
$<u = 5>$



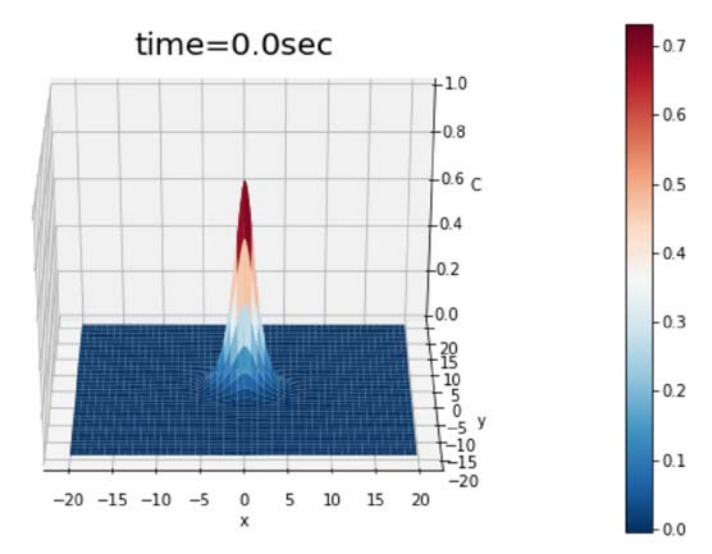
$<u = 10>$

Result

- 2-D
  - 초기값: $C(x,0) = \frac{1}{\cosh\left(10\sqrt{x^2+y^2}\right)}$
  - $\vec{u} = (5,0)$
  - K: 거리에 따라 선형적으로 증가



  - 초기값: $C(x,0) = \frac{1}{\cosh\left(10\sqrt{x^2+y^2}\right)}$
  - $\vec{u} = (5,5)$
  - Smaller K



Conclusion

- 간단한 사례 연구를 통해 PINN을 이용한 확산 모델링 수행 방법을 제시함
- 현재 대기 확산 예측을 위한 전산 코드들은 주로 Gaussian Plume/Puff model을 사용하고 있음
- Gaussian model은 계산이 간단하고 빠르지만, 많은 가정사항이 필요하고 따라서, 복잡한 상황에 적용되기 어려움
- PINN은 물리적 모델을 인공 신경망으로 학습하는 방법

  가정 사항을 필요로 하지 않으며, 물리적 모델의 원래 형태를 그대로 사용함
- 향후, 확산에 관여하는 보다 다양한 파라미터들을 고려하여 확산 모델링 수행할 예정

Q&A

감사합니다.

김기범 (rlqja2177@gmail.com)
허균영 (gheo@khu.ac.kr)