전산물리 기말프로젝트 기획안

정준상, 김영훈

주제 : Visualizing partial differential equations with time-dependent case using matplotlib

설명

- 주어진 편미분 방정식에 initial condition / boundary condition 을 입력하여 시간에 따른 해를 얻고 시각화 시킨다.

- 편미분 방정식

Schrodinger equation : $i\hbar\frac{\partial\psi}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\psi + V\psi$

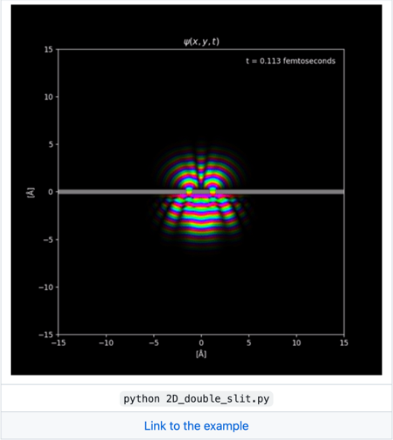
Heat diffusion equation : $\nabla^2 T + \frac{\dot{q}}{k} = \frac{\rho C_p}{k}\frac{\partial T}{\partial t}$

Wave equation (vibrating membrane) : $\frac{\partial^2 u}{\partial t^2} = c^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$

- 이미 구현된 예시

1. qmsolve : https://github.com/quantum-visualizations/qmsolve

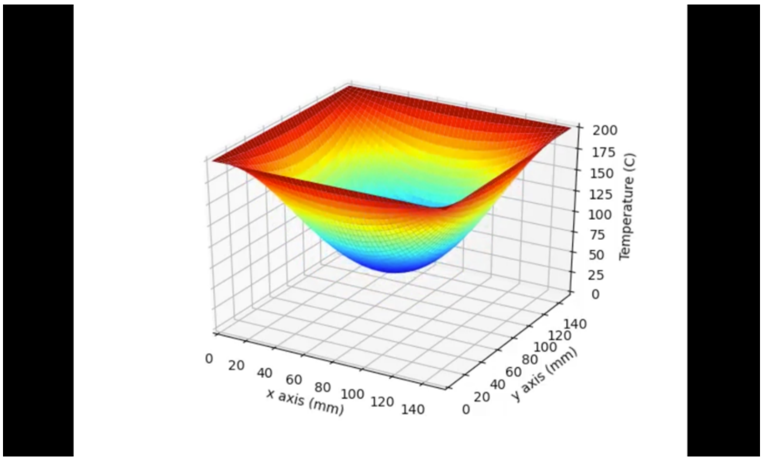
   a module for solving and visualizing the schrodinger equation



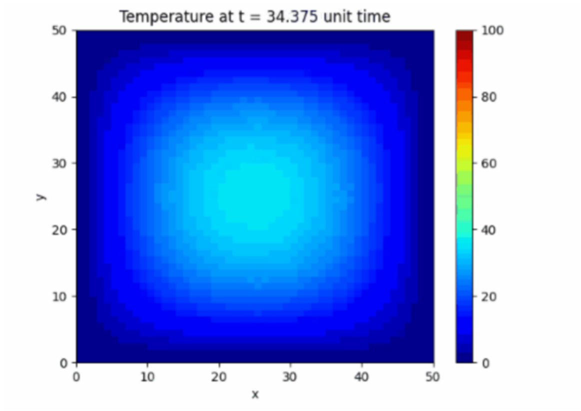
`python 2D_double_slit.py`

2. 3D Heat Equation Numerical simulation : https://www.aeroodyssey.org/3d-heat-equation

   Visualizing heat diffusion using finite element method

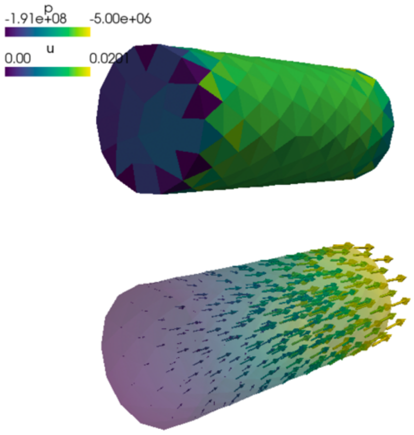


3. Solving 2D Heat Equation Numerically using Python : https://levelup.gitconnected.com/solving-2d-heat-equation-numerically-using-python-3334004aa01a
   Visualizing heat diffusion using finite element method

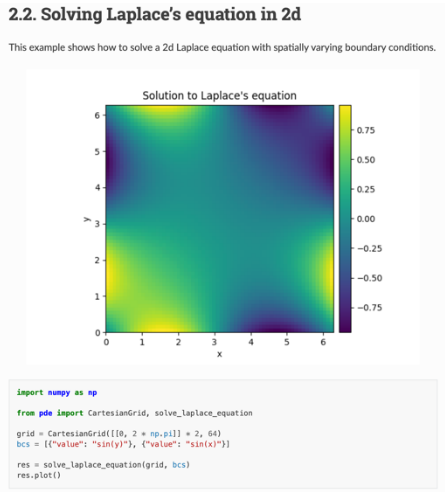


The numeric solution where all boundary conditions are 0 with randomized initial condition inside the grid

4. SfePy : https://sfepy.org/doc-devel/index.html
   SfePy is software for solving systems of coupled partial differential equations by finite element method in 1D, 2D, 3D



5. py-pde : https://github.com/zwicker-group/py-pde
   py-pde is a Python package for solving partial differential equations (PDEs). The package provides c lasses for grids on which scalar and tensor fields can be defined.



2.2. Solving Laplace's equation in 2d

This example shows how to solve a 2d Laplace equation with spatially varying boundary conditions.

```
import numpy as np

from pde import CartesianGrid, solve_laplace_equation

grid = CartesianGrid([[0, 2 * np.pi]] * 2, 64)
bcs = [{"value": "sin(y)"}, {"value": "sin(x)"}]

res = solve_laplace_equation(grid, bcs)
res.plot()
```