Homework 5

Bray Moll

October 22, 2023

1 1)

1.1 Problem Setup

```
import pandas as pd
from solver import solver
from solver import mesher
import plotnine as pn
```

In this problem we are using a spatial domain $x\epsilon[0, 1]$

We are using an intial condition $T(x, t = 0) = 0°C$ a left dirichlet boundary $T(0, t) = 50°C$

On the right size we will apply a zero Neuiman Boundary Condition $\frac{\partial T}{\partial x}(x = 1, t) = 0$

We will set the thermal diffusivity to $\alpha^2 = 0.0001\frac{m^2}{s}$

Additionally we can set paramaters of interest such as the maximum, the number of cells as well as the mesh type. The only difference for the mesh type in 1d is if the node is directly on the boundarys (as would be seen in the $\text{finite}_{\text{difference}}$ or if the edge cell face is on the boundary

```
def create_mesh(n_cells, mesh_type):
    mesh = mesher.create_1Dmesh(x=[0, 1], n_cells=n_cells, mesh_type=mesh_type)
    mesh.set_cell_temperature(0)  # set initial conditions to 0 celcicus
    mesh.set_dirichlet_boundary("left", 50)  # Left to 50 c
    mesh.set_neumann_boundary("right")
    mesh.set_thermal_diffusivity(0.0001)  # m^2/s
    return mesh

n_cells = 20
time_max = 30000
```

1.2 Explicit method Unstable

Create a solver object with time paramaters, desired method, and a mesh and solve using .solve(t_{final})

```
explicit_solution_unstable = solver.solver_1d(
    mesh=create_mesh(n_cells, mesh_type),
    initial_time=0,
    time_step_size=15,
    method="explicit",
)
explicit_solution_unstable.solve(600)
```

The data will be stored in a pandas data frame in the self.saved$_{data}$ atribute. For example, as we have 20 nodes, the first 20 rows will be the initial condition, followed by will be the first time step.

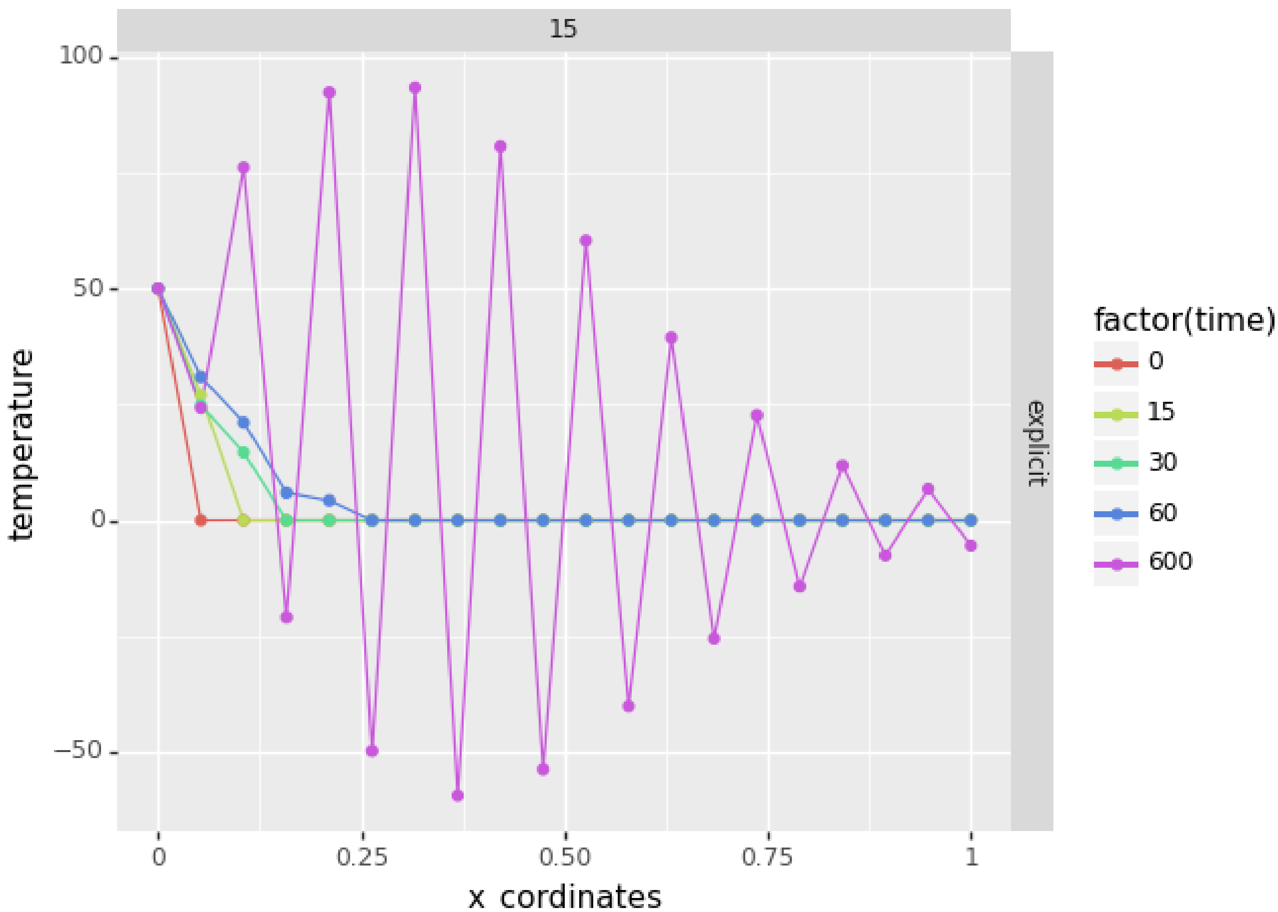
```
print(explicit_solution_unstable.saved_data.head(10))
```

```
     method  time_step_size  time  x_cordinates  temperature
0  explicit              15     0      0.000000         50.0
1  explicit              15     0      0.052632          0.0
2  explicit              15     0      0.105263          0.0
3  explicit              15     0      0.157895          0.0
4  explicit              15     0      0.210526          0.0
5  explicit              15     0      0.263158          0.0
6  explicit              15     0      0.315789          0.0
7  explicit              15     0      0.368421          0.0
8  explicit              15     0      0.421053          0.0
9  explicit              15     0      0.473684          0.0
```

The data can be plotted using your favorite tool for Pandas dataframes

```
time_points = [0,15,30,60,600]
plot_data_filterd_bool = explicit_solution_unstable.saved_data["time"].isin(time_points)
plot_data_filtered = explicit_solution_unstable.saved_data[plot_data_filterd_bool]

plot = (
    pn.ggplot(
        plot_data_filtered,
        pn.aes("x_cordinates", "temperature", color="factor(time)"),
    )
    + pn.geom_line()
    + pn.geom_point()
    + pn.facet_grid("method~time_step_size")
```

```
  )
plot.save("explicit_unstable.png")
```



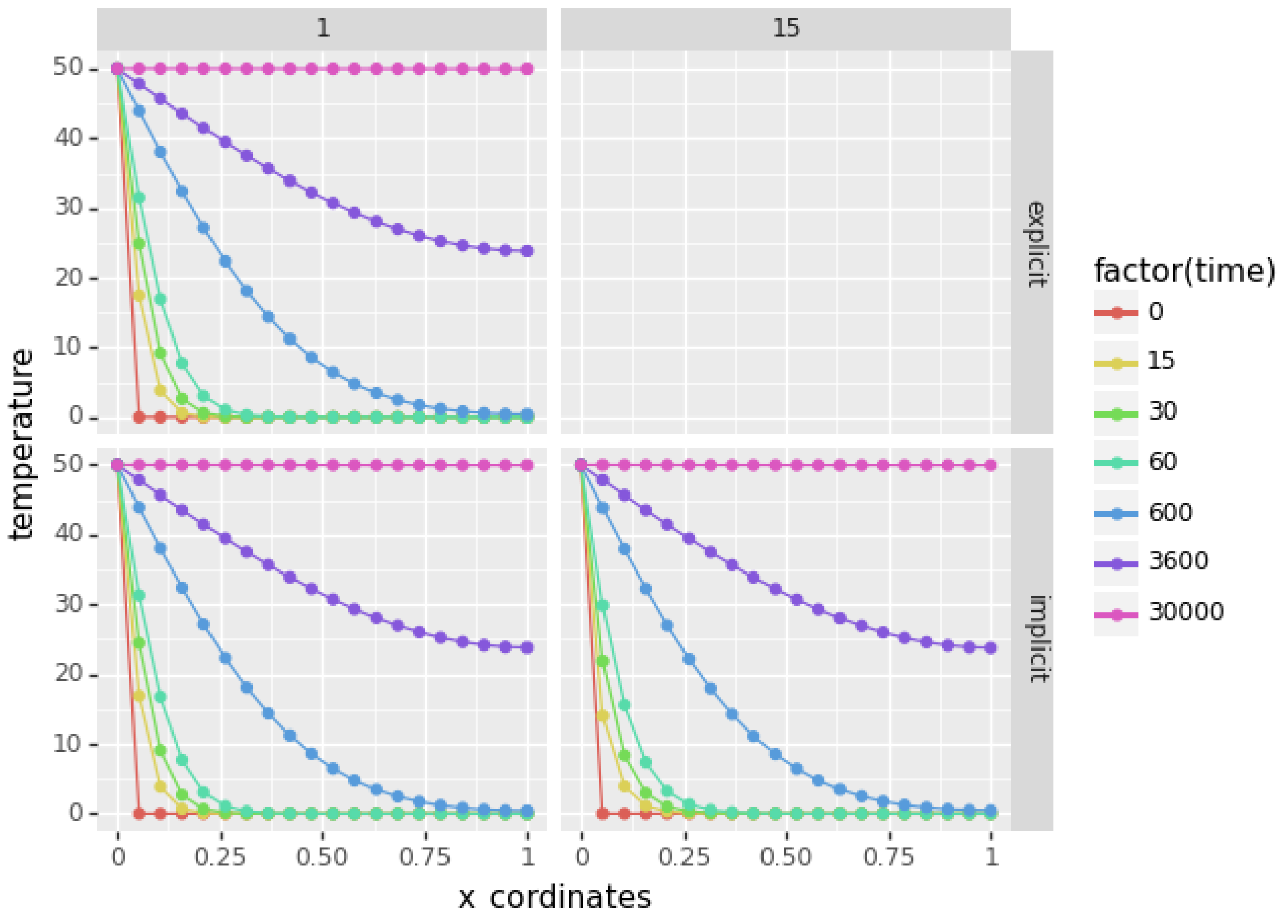
As demonstrated, the explicit method will demonstrate numerical unstability when the time step it too large.

1.3 Finite Difference Discritization

```
mesh_type = "finite_difference"
explicit_solution_stable = solver.solver_1d(
    mesh=create_mesh(n_cells, mesh_type),
    initial_time=0,
    time_step_size=1,
    method="explicit",
)
explicit_solution_stable.solve(time_max)

implicit_solution_15sec = solver.solver_1d(
```

```
    mesh=create_mesh(n_cells, mesh_type),
    initial_time=0,
    time_step_size=15,
    method="implicit",
)
implicit_solution_15sec.solve(time_max)

implicit_solution_1sec = solver.solver_1d(
    mesh=create_mesh(n_cells, mesh_type),
    initial_time=0,
    time_step_size=1,
    method="implicit",
)
implicit_solution_1sec.solve(time_max)

plot_data_finite_difference = pd.concat(
    [

        explicit_solution_stable.saved_data,
        implicit_solution_15sec.saved_data,
        implicit_solution_1sec.saved_data,
    ]
)
time_points = [
    0,
    15,
    30,
    60,
    600,
    3600,
    time_max,
]  # time points that you want to plot

plot_data_filterd_bool = plot_data_finite_difference["time"].isin(time_points)
plot_data_filtered = plot_data_finite_difference[plot_data_filterd_bool]

plot = (
    pn.ggplot(
        plot_data_filtered,
        pn.aes("x_cordinates", "temperature", color="factor(time)"),
    )
    + pn.geom_line()
    + pn.geom_point()
    + pn.facet_grid("method~time_step_size")
```

```
)

plot.save("finite_difference.png")
```



1.4 Finite Volume Discritization

Alternatly, the finite pvolume discritization can be used

```
mesh_type = "finite_volume"
```