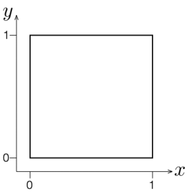
PDE python exercise 3

In this exercise we will apply an iterative method to solve an elliptic PDE. The problem we consider is a square plate and in particular the distribution of the temperature for a given set of boundary conditions. This plate is illustrated in the figure below as well as its boundary conditions.



$$T = 20 \text{ for } 0 \leq x \leq 1 \text{ and } y = 1$$
$$T = 10 \text{ for } 0 \leq x \leq 1 \text{ and } y = 0$$
$$T = 20 \text{ for } 0 < y \leq 1 \text{ and } x = 0$$
$$T = 10 \text{ for } 0 \leq y < 1 \text{ and } x = 1$$

The plate is of size 1 in x and 1 in y. Dirichlet boundary conditions are applied to each edge of the plate. The temperature distribution in the plate is governed by the steady-state heat equation which reads:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

We want to solve this problem using a centered finite difference for both partial derivatives and the Gauss-Seidel's method.

In this exercise, you need to :

On paper:

1. Discretize the numerical scheme using the chosen finite difference
2. Formulate the numerical scheme using the Gauss-Seidel's method

Within the python notebook:

1. Discretize the independent variables x and y using a `numpy meshgrid` , first you can use a step size of $1/20$
2. Write a function which takes in the discretized independent variables as well as the number of iteration for the Gauss-Seidel's method and return the temperature distribution. The boundary conditions will be enforced within the function.
3. Plot the resulting temperature distribution using a 3D surface plot when using 200 iterations.
4. What happens when you refined the mesh grid (decrease by a factor of 2 the step size)?
5. What happens when you solve the heat equation using only 20 iterations?

Question 1:

We express our finite difference approximation of $\frac{\partial^2 T}{\partial x^2}$ and $\frac{\partial^2 T}{\partial y^2}$ as:

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{h^2}$$

and

$$\frac{\partial^2 T}{\partial y^2} \approx \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{h^2}$$

where we have assumed that the discretization along x Δx and along y Δy are the same and defined as h.

When incorporating the finite difference approximations in our PDE, we can obtain the following numerical scheme driving our problem:

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i,j} = 0$$

Question 2

The Gauss-Seidel's method is defined as:

$$T_{i,j}^{m+1} = T_{i,j}^{m} + \delta T_{i,j}$$

where m an iteration counter.

The correction term $\delta T_{i,j}$ has then to be expressed as function of the temperatures across our plate. According to the Gauss-Seidel's method, the correction term will contain values at iteration m and $m+1$ and can be expressed as:

$$\delta T_{i,j} = \frac{1}{4}\left(T_{i-1,j}^{m+1} + T_{i,j-1}^{m+1} + T_{i+1,j}^{m} + T_{i,j+1}^{m} - 4T_{ij}^{m}\right)$$

To determine which terms are computed at iteration m and $m+1$ it was assumed that the numerical scheme will be screened starting from the lower left of the plate towards the higher right corner.

First we import the mandatory modules:

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

We first discretize our independent variables x and y using a `numpy meshgrid`

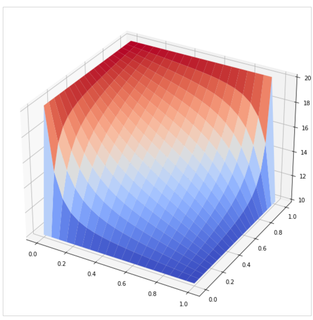
In [ ]:
```
N = 20
x = np.linspace(0, 1.0, N)
y = np.linspace(0, 1.0, N)

[X,Y] = np.meshgrid(x,y)
```

In [ ]:
```
def num_scheme(X,niter,N):
#
    T = np.zeros_like(X)
#
    Ttop, Tleft = 20.0,20.0
    Tbottom, Tright = 10.0,10.0
#
    T[-1,:] = Ttop
    T[:,0]  = Tleft
    T[:,-1] = Tright
    T[0,:] = Tbottom
#
    for iteration in range(0,niter):
        for j in range(1,N-1):
            for i in range(1, N-1):
                R = (T[j,i-1]+T[j-1,i]+T[j,i+1]+T[j+1,i]-4.0*T[j,i])
                dT = 0.25*R
                T[j,i]+=dT
    return T
```

We can then execute our function and plot the resulting temperature in a 3D surface.

In [ ]:
```
T = num_scheme(X,200,N)
ax = plt.figure(figsize=(10,10)).add_subplot(projection='3d')
ax.plot_surface(X, Y, T,cmap='coolwarm')
```

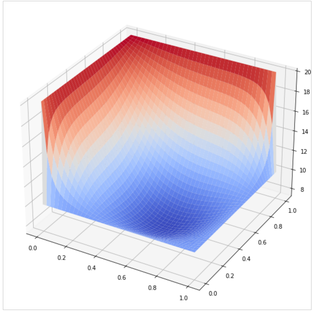
Out[ ]:
```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x10fd29d30>
```



We can decrease the mesh size by a factor of 2 and re-execute our function. The same temperature distribution is obtained but we have a better representation of the temperature gradient close to the boundaries.

In [ ]:
```
N = 40
x = np.linspace(0, 1.0, N)
y = np.linspace(0, 1.0, N)

[X,Y] = np.meshgrid(x,y)
T = num_scheme(X,200,N)
ax = plt.figure(figsize=(10,10)).add_subplot(projection='3d')
ax.plot_surface(X, Y, T,cmap='coolwarm')
```

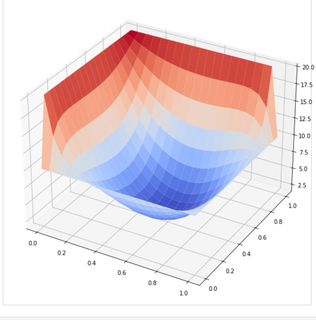
Out[ ]:
```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x10ffdc340>
```



When we carry out the same computation using a lower number of iteration we can observe that the temperature distribution is quite different from before with having very low temperatures in the center of the plate. Here we have not iterated enough leading to the wrong temperature distribution.

In [ ]:
```
N = 20
x = np.linspace(0, 1.0, N)
y = np.linspace(0, 1.0, N)

[X,Y] = np.meshgrid(x,y)
T = num_scheme(X,20,N)
ax = plt.figure(figsize=(10,10)).add_subplot(projection='3d')
ax.plot_surface(X, Y, T,cmap='coolwarm')
```

Out[ ]:
```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x11e981eb0>
```



In [ ]: