Assignment 5

Given Information
We want to model the spread of a polluting gas in the atmosphere. The computational domain Ω is a $(L_x \times L_y)$ rectangle with the lower-left corner at $(0,0)$. The time interval of interest is $t \in [0,T]$. The gas is emitted by several localized sources at a constant rate in time. The emission rate density is described by the function:

$$f(x,y) = \sum_{i=1}^{3} e^{-\alpha(x-x_i)^2-\alpha(y-y_i)^2} \tag{1}$$

And as per values given in Table 1:

$$\alpha = 50, A = 5, x_i = 1, 1.5, 4, y_i = 2, 2.5, 3 \, for\, i = 1,2,3\, and\, (x,y) \in \vec{\Omega}$$

The transport of the pollutant $u(x,y,t)$ in the atmosphere is described by the convection-diffusion equation:

$$\frac{\partial u}{\partial t} - D\Delta u + \mathbf{w}.\nabla u = f \tag{2}$$

where D (here $D = 0.005$) is the effective diffusivity describing the (approximately) diffusive transport of gas caused by the atmospheric turbulence and $w(x,y,t) = w_x\hat{i} + w_y\hat{j}$ is the vector field of the mean wind velocity with its components given by:

$$w_x(x,y,t) = Bcos(\omega t),\ w_y(x,y,t) = Bsin(\omega t) \tag{3}$$

where $B = 0.5$, $\omega = \frac{2\pi}{4T}$, and $T = 10$

$\vec{\Omega} = [0,10] \times [0,10]$ is a square with the corner (0,0), (10,0), (10,10), and (0,10)

Problem 1
Formulation of the problem and its weak form.

Solution
(a) Considering the initial concentration of pollutant to be zero as well as assuming that the concentration of pollutant at the boundary is zero at all times, the boundary value problem is given by:

$$\frac{\partial u}{\partial t} - D\Delta u + \mathbf{w}.\nabla u = f,\ (x,y) \in \Omega = (0,10) \times (0,10),\ t \in [0,T]$$
$$u(x,y,t) = 0,\ (x,y) \in \partial\Omega,\ t \in [0,T]$$
$$u(x,y,t) = 0,\ (x,y) \in \Omega,\ t = 0$$
$$f(x,y) = \sum_{i=1}^{3} e^{-\alpha(x-x_i)^2-\alpha(y-y_i)^2},\ (x,y) \in \vec{\Omega} = [0,10] \times [0,10]$$
$$with\, \alpha = 50, A = 5, x_i = 1, 1.5, 4, y_i = 2, 2.5, 3\, for\, i = 1,2,3\, and\, (x,y) \in \vec{\Omega}$$
$$w_x(x,y,t) = Bcos(\omega t),\ w_y(x,y,t) = Bsin(\omega t)$$
$$where\, B = 0.5,\ \omega = \frac{2\pi}{4T},\ and\, T = 10$$

(b) Backward-Euler time integration method discretizes the time domain using finite difference method. Ignoring higher order terms, the formulation is given by:

$$\frac{{u_{i,j}}^{k+1} - {u_{i,j}}^{k}}{\Delta t} = D\Delta u - \mathbf{w}.\nabla u + f$$

$$u(t_{k+1}) = u(t_k) + \Delta t.(D\Delta u - \mathbf{w}.\nabla u + f)$$

where Δt is the time step, with superscripts k and $k+1$ denoting time increments and subscripts i, j indicating space grid points.

(c) The weak form of Galerkin FEM is given by:

$$F_{k+1}(u,v) = u - \Delta t D\nabla^2 u + \Delta t\vec{w}.\nabla u - (u^k + \Delta t f^{k+1}) = 0$$

$$\int_\Omega F_{k+1}(u,v).v.dx = 0$$

$$\int_\Omega u.v.dx - (u^k + \Delta t f^{k+1}).v.dx + \Delta t\vec{w}.\nabla u.v.dx - \int_\Omega \Delta t.D\nabla^2 u.v.dx = 0$$

Solving the last part of the equation by parts:

$$\int_\Omega u.v.dx - (u^k + \Delta t f^{k+1}).v.dx + \Delta t\vec{w}.\nabla u.v.dx - (v.\nabla u|_\Omega - \int_\Omega \nabla v.\nabla u.dx)$$

u is zero at the boundary and therefore the ∇u term also goes to zero. The final expression is given by:

$$\int_\Omega u.v.dx - (u^k + \Delta t f^{k+1}).v.dx + \Delta t\vec{w}.\nabla u.v.dx + \int_\Omega \nabla v.\nabla u.dx$$

(d) The Backward-Euler method used for time integration is inherently stable and hence the numerical solution obtained is by extension also stable.

Problem 2
Implementation in FEniCS.

Solution

```
from __future__ import print_function
from dolfin import *

import matplotlib.pyplot as plt
import matplotlib.tri as mtri

import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import math

# Time step definition
T = 10
num_steps = 20
dt = T/num_steps

# Rectangle domain with mesh
Lx = Ly = 10
nx = ny = 100

mesh = RectangleMesh(Point(0,0),Point(Lx,Ly),nx,ny)
```

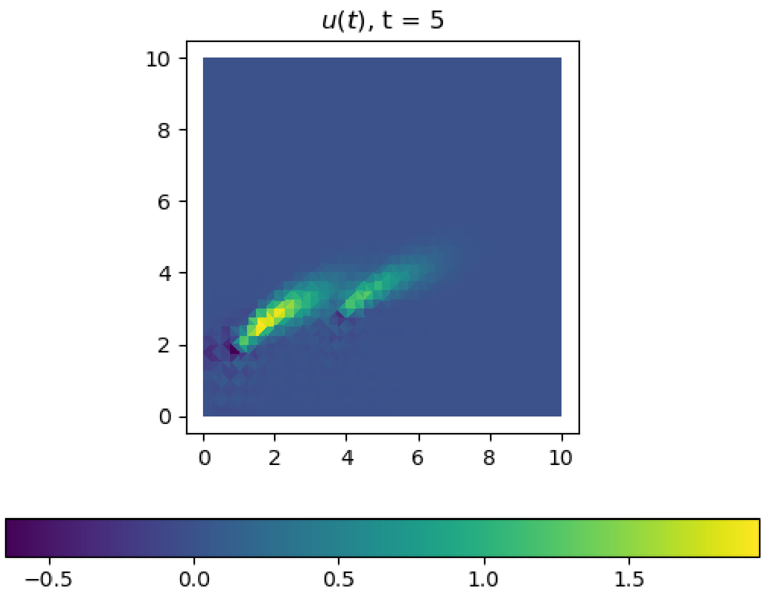
```
# Finite element with Lagrange basis function
P1 = FiniteElement('Lagrange',triangle,1)

# Function space V for scalar functions
V = FunctionSpace(mesh, P1)

# Function space W for vector functions
W = FunctionSpace(mesh, MixedElement([P1,P1]))

# Velocity vector field
B = 0.5; omega = (2*np.pi)/(4*T)
velocity = Expression(('B*cos(omega*t)','B*sin(omega*t)'),degree=2,B=B,omega=
    ↪ omega,t=0.0)
w = Function(W)
w.interpolate(velocity)

# Define Boundary Condition (Given u=0 at all times at the boundary)
u_D = Constant(0.0)

# Dirichlet (left) boundary
def boundary(x,on_boundary):
    return on_boundary

# The essential boundary condition
bc = DirichletBC(V, u_D, boundary)

# Initial value of u_k
u_k = interpolate(u_D, V)

# Source function
A = 5
alpha = 50
x_1,x_2,x_3 = [1,1.5,4] # Given in Table 1
y_1,y_2,y_3 = [2,2.5,3] # Given in Table 1

# Variational problem
u = TrialFunction(V)
v = TestFunction(V) # Automatically assumes zero Dirichlet BC at V
f = Expression('A * exp(-alpha*(x[0]-x_1)*(x[0]-x_1) - alpha*(x[1]-y_1)*(x[1]-
    ↪ y_1)) + A * exp(-alpha*(x[0]-x_2)*(x[0]-x_2) - alpha*(x[1]-y_2)*(x[1]-
    ↪ y_2)) + A * exp(-alpha*(x[0]-x_3)*(x[0]-x_3) - alpha*(x[1]-y_3)*(x[1]-
    ↪ y_3))',degree=2,alpha=alpha,A=A,x_1=x_1,y_1=y_1,x_2=x_2,y_2=y_2,x_3=x_3
    ↪ ,y_3=y_3)

D = 0.005
F = u*v*dx + D*dt*dot(grad(u),grad(v))*dx + dt*dot(w,grad(u))*v*dx - (u_k + dt
    ↪ *f)*v*dx

a,L = lhs(F),rhs(F)

# Time stepping loop
u = Function(V)
```

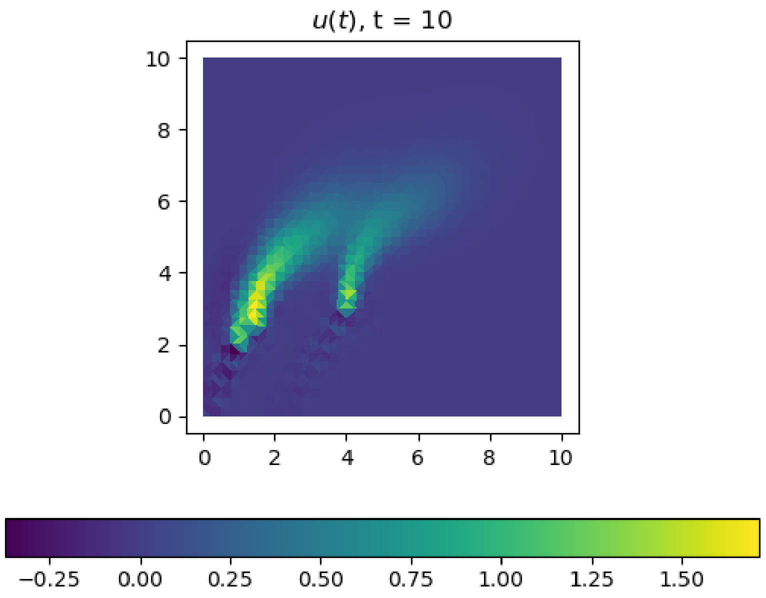
```
t = 0

# Setting up for plotting the u values
plt.ion()
plt.figure(1)
plt.clf()
fig, ax = plt.subplots(nrows=1, ncols=1, num=1)
points = mesh.coordinates()

unow = u_k.compute_vertex_values(mesh)
im = ax.tripcolor(points[:,0],points[:,1],unow)
ax.set_aspect('equal', adjustable='box')
cbar = fig.colorbar(im,ax=ax, orientation='horizontal')
ax.set_title(r'$u(t)$, t = '+str(np.round(t,2)))
plt.savefig('u_plot_t='+str(t)+'.png')

unow_5 = 0; unow_10 = 0

for n in range(num_steps):

    # Updating time
    t += dt
    u_D.t = t
    velocity.t = t # updating time inside velocity
    w.interpolate(velocity) # updating w

    # Solve variational problem
    solve(a == L,u,bc)

    # Update previous solution
    u_k.assign(u)

    if t == 5:
        unow_5 = u_k.compute_vertex_values(mesh)
    elif t == 10:
        unow_10 = u_k.compute_vertex_values(mesh)

im.remove()
im = ax.tripcolor(points[:,0],points[:,1],unow_5)
cbar.update_normal(im)
ax.set_title(r'$u(t)$, t = 5')
plt.savefig('u_plot_t=5.png')

im.remove()
im = ax.tripcolor(points[:,0],points[:,1],unow_10)
cbar.update_normal(im)
ax.set_title(r'$u(t)$, t = 10')
plt.savefig('u_plot_t=10.png')
```

Problem 3
Numerical experiments.

Solution
(a) We used the time step $h_t = T/20$ and the other problem parameters to setup our computation in the script above.

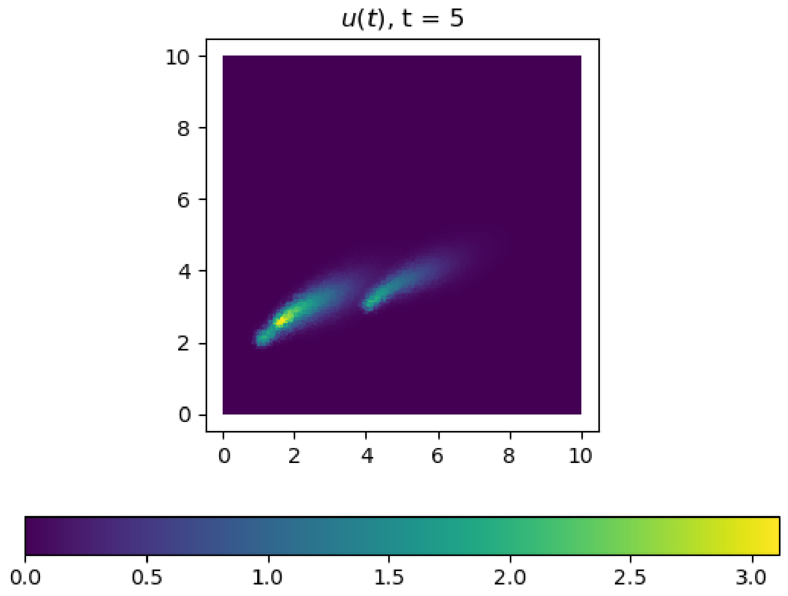
(b) Setting $N_x = N_y = 40$, we obtain the following solution plots for $t = 5$ and $t = 10$

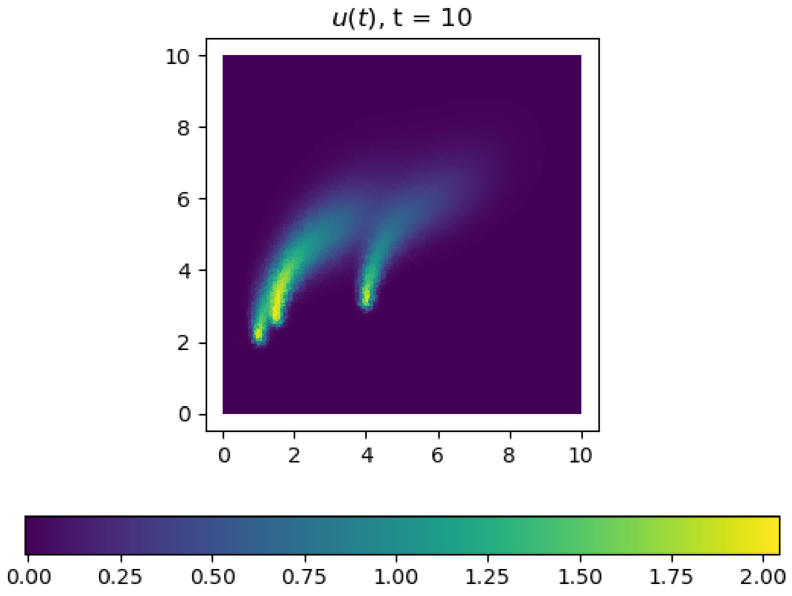




The results show how the pollutant is transported from the source in our domain. We have 3 source points, with a high concentration of pollutant. The difference in concentration and the wind drive the transportation process. We can see how the wind velocity has a strong influence on the spread, as the y-component wind velocity starts to dominate as time passes.

(c) Setting $N_x = N_y = 100$ is providing a result with high enough resolution to distinguish the 3 sources and providing a better color bar. The resulting plots are:


$u(t)$, t = 5
0
2
4
6
8
10
0.0
0.5
1.0
1.5
2.0
2.5
3.0


$u(t)$, t = 10
0
2
4
6
8
10
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00