Atmospheric Dispersion

The problem considered in this report is the emission of a gaseous pollutant from a chimney stack, and how the distribution of the concentration of the pollutant is affected by both diffusion (movement down a concentration gradient) and advection (movement under the influence of a force). I considered a steady state for both the diffusion and advection terms, in other words there is no change in the local atmosphere (*I also do not consider a topographical change, the chimney in the model is placed on a perfectly flat plain*)

The idea behind creating this model is to be able to accurately predict the concentration of pollutants released from a chimney when sampled at and around ground level, it is observed that low windspeed conditions create the highest ground level concentration of pollutants (F.B. Smith, Cited in Sharan et al 1996). Using a model from (Sharan et al. 1996) gives an atmospheric diffusion and advection equation, as follows

$$\frac{\partial C}{\partial t} + u\frac{\partial C}{\partial x} + v\frac{\partial C}{\partial y} + w\frac{\partial C}{\partial z} = \frac{\partial}{\partial x}\left[K_x\frac{\partial C}{\partial x}\right]$$

$$+ \frac{\partial}{\partial y}\left[K_y\frac{\partial C}{\partial y}\right] + \frac{\partial}{\partial z}\left[K_z\frac{\partial C}{\partial z}\right] + S + R \qquad (1)$$

Here u,v,w define wind speed components, C the pollutants concentration, S the source term, R the removal term, $K_xK_yK_z$ the eddy diffusivity coefficients

Now I make the following assumptions to simplify/rearrange the above equation to provide something which I can use to create a numerical solution,

-There is no pollutant removal (No settling, No removal via chemical reaction, etc.) R = 0.

-The windspeed in the vertical direction is much smaller than either horizontal windspeed (and this model deals with a non-particulate pollutant so settling speed is also not a factor) hence I disregard advection in the vertical direction.

-I take the x-axis for the model in the direction of greatest mean wind speed, this lets me similarly disregard advection in the y direction.

-The value for windspeed in the x direction is large enough that diffusion in the x direction can be disregarded giving only the advection term in the x direction

- Again using (Sharan et al. 1996),

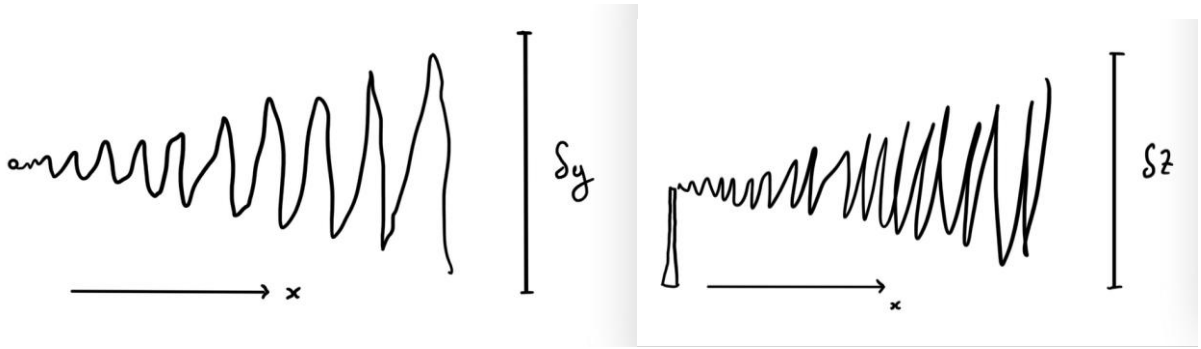
The source term is defined, $S = q \cdot \delta(x)\delta(y)\delta(z)$

I later disregard the Dirac's delta and instead implement a singularity in the python code I use to model the problem

-I built up different models for different windspeeds and different values of the eddy diffusivity co-efficients, but within each model I treated each of these values as constant (as mentioned above).

With the model there is diffusion in both the z & y direction and advection in the x direction, whilst I would like to create a 3-dimensional model I will settle (for convenience and time's sake) to create two models one for diffusion in the z direction and one for diffusion in the y direction (these will both be 2-dimensional models).

I've started by drawing some sketches of what I'm looking to model,



(Of course, the actual distributions of the pollutants should not look like perfect cones)

The equations for the models are as follows (The Dirac functions that do not correspond to the selected co-ordinate systems have been eliminated),

$\frac{\partial C}{\partial t} + u \cdot \frac{\partial C}{\partial x} = K_y \frac{\partial^2 C}{\partial y^2} + q \cdot \delta(x)\delta(y)$ and then $\frac{\partial C}{\partial t} + u \cdot \frac{\partial C}{\partial x} = K_z \frac{\partial^2 C}{\partial z^2} + q \cdot \delta(x)\delta(z)$

Before going any further I will mention that in my code and notation I use i,j,k,n as the indices for x,y,z,t respectively.

I first start by looking at the simpler of the two problems, that being the diffusion advection system in x,y,t. I first consider just the diffusion in y. I use a FTCS scheme (*with a lagged term, usage explained later*) which is conditionally stable to solve this equation,

$$\frac{\partial C}{\partial t} = K_y \frac{\partial^2 C}{\partial y^2}$$

Which gives,

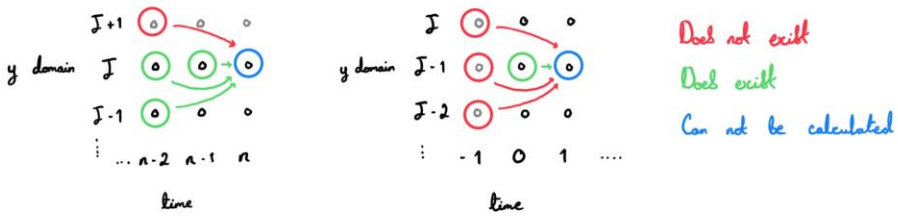
$$\frac{C_j^{n+1} - C_j^n}{\Delta t} = \left(K_y \cdot \left(\frac{C_{j+1}^{n-1} - 2C_j^{n-1} - C_{j+1}^{n-1}}{\Delta y^2} \right) \right)$$

Re-arranging,

$$C_j^{n+1} = C_j^n + \Delta t \cdot \left(K_y \cdot \left(\frac{C_{j+1}^{n-1} - 2C_j^{n-1} - C_{j+1}^{n-1}}{\Delta y^2} \right) \right)$$

I coded this into python and immediately noticed a problem. Since arrays are necessarily limited in size, I cannot calculate values for the very ends of my arrays because I will encounter out of bounds errors. And because the problem does not have boundary conditions there is no way of fixing this numerically. Instead, I have a y domain large enough that within the specified time the concentration does not reach the edges of the arrays. Then by limiting the calculations to remain inside the array I will not be affecting the model.

It's also worth noting that the index for time steps must start at the second time step for similar reasons. I have drawn a quick diagram to visually demonstrate the issue, *(This also will apply to the x domain)*



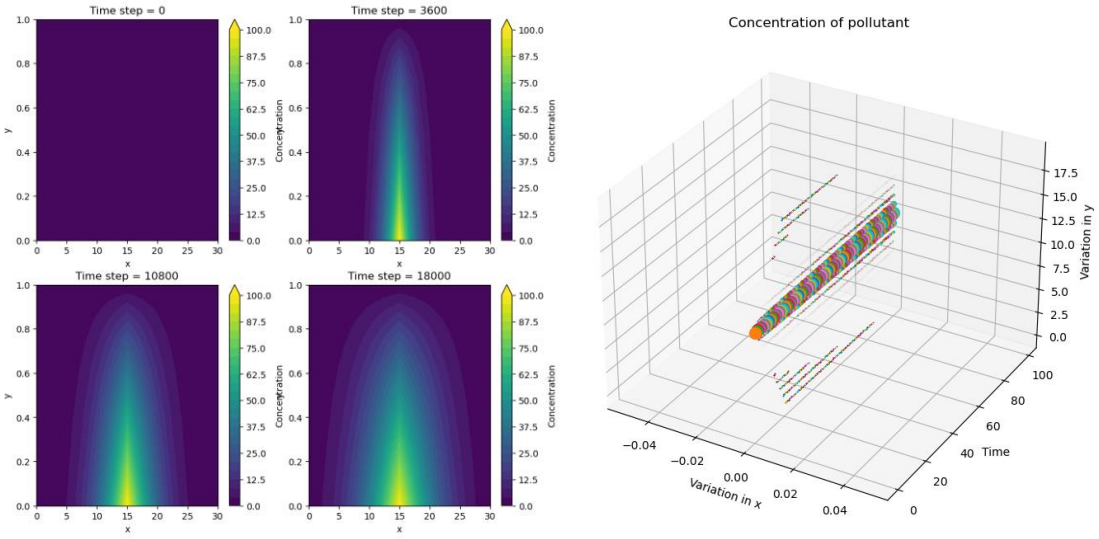
Using both contour and scatter plots to graph the distribution of the pollutant for just diffusion the expected graph slowly fills out in the first row of y (bearing in mind that the chimney is set to be constantly emitting so it is expected for concentration to strictly increase outwards). Before modelling the solution, I check that I have conditional stability for the chosen constants those being,

```
#Set up initiual values
EMISSION_RATE = 100
DIFFUSIVITY_COEFFICIENT_Y = 0.1
DIFFUSIVITY_COEFFICIENT_Z = 0.01
WIND_SPEED = 0.1

TIME_STEP = 1
SECOND = 1
HOUR = 60 * 60 * SECOND
DAY = 24 * HOUR
WEEK = 7 * DAY

NUM_TIME_STEPS = HOUR

X_DOMAIN = 2
Y_DOMAIN = 20
Z_DOMAIN = 10


X_STEPS = X_DOMAIN
Y_STEPS = Y_DOMAIN
Z_STEPS = Z_DOMAIN
#Setting our steps equal to our do

Dx = X_DOMAIN/X_STEPS
Dy = Y_DOMAIN/Y_STEPS
Dz = Z_DOMAIN/Z_STEPS
```

From the lecture materials that 2-Dimensional Diffusion can be checked with Fourier analysis, using the term,

$$\frac{\Delta t \cdot K_y}{\Delta y^2} = 1$$

So this solution should be numerically stable.



(For the purposes of graphing I implemented a function that takes only the positive values in the array, this is not necessary for just diffusion as everything is strictly positive but becomes necessary for advection where negative terms can appear in the array)

The scatter graph above is not the one from the exact initial values listed, due to how intensive the creation of this graph was I was unable to generate more figures after the initial figure without crashing my IDE (spyder). For the same reason, the scatter graph is for far fewer time steps.

Regardless the two graphs illuminate distinct parts of the model. As would be expected diffusion is a very slow process. Notice also from the scatter graph the introduction of an error. Having checked that my solution is numerically stable the other most obvious possibility is that my model has some error relating to the way python handles numbers. If I had time, I could attempt to implement code to try and avoid this but given it is not having a visible impact on the contour graph and that a fix for this might require considerable adjustments to the code. I will proceed without addressing the error.

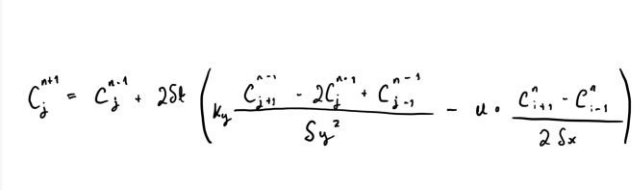
Solving now with the advection term the problem becomes more difficult because an FTCS scheme is conditionally stable for diffusion and unstable for advection, and similarly a CTCS scheme is conditionally stable for advection and unstable for diffusion. This means for there to be a stable solution to the problem a mixed scheme must be employed. The first attempt at this will be using a CTCS term for advection and a lagged CTCS term for diffusion (*as seen in the lecture materials but extended into the 2-dimensional case*). Solving the equation,

$$\frac{\partial C}{\partial t} + u \cdot \frac{\partial C}{\partial x} = K_y \frac{\partial^2 C}{\partial y^2} + q \cdot \delta(x)\delta(y)$$

Producing,

$$\frac{C_j^{n+1} - C_j^n}{2 \cdot \Delta t} + u \cdot \frac{C_{i+1}^n - C_{i-1}^n}{2 \cdot \Delta x} = \left(K_y \cdot \left(\frac{C_{j+1}^{n-1} - 2C_j^{n-1} - C_{j+1}^{n-1}}{\Delta y^2} \right) \right)$$

Re-arranging,



(Had to insert as an image because word was not including brackets)

As with previous before graphing the results Fourier analysis is used to check stability for each term separately for the chosen starting conditions below,

```
#Set up initiual values
EMISSION_RATE = 100
DIFFUSIVITY_COEFFICIENT_Y = 1
DIFFUSIVITY_COEFFICIENT_Z = 0.01
WIND_SPEED = 0.1

TIME_STEP = 1
SECOND = 1
HOUR = 60 * 60 * SECOND
DAY = 24 * HOUR
WEEK = 7 * DAY

NUM_TIME_STEPS = 5 * HOUR

X_DOMAIN = 60
Y_DOMAIN = 30
Z_DOMAIN = 10

#Setting the steps equal to the do
X_STEPS = X_DOMAIN
Y_STEPS = Y_DOMAIN
Z_STEPS = Z_DOMAIN

Dx = X_DOMAIN/X_STEPS
Dy = Y_DOMAIN/Y_STEPS
Dz = Z_DOMAIN/Z_STEPS
```

So, the conditions are

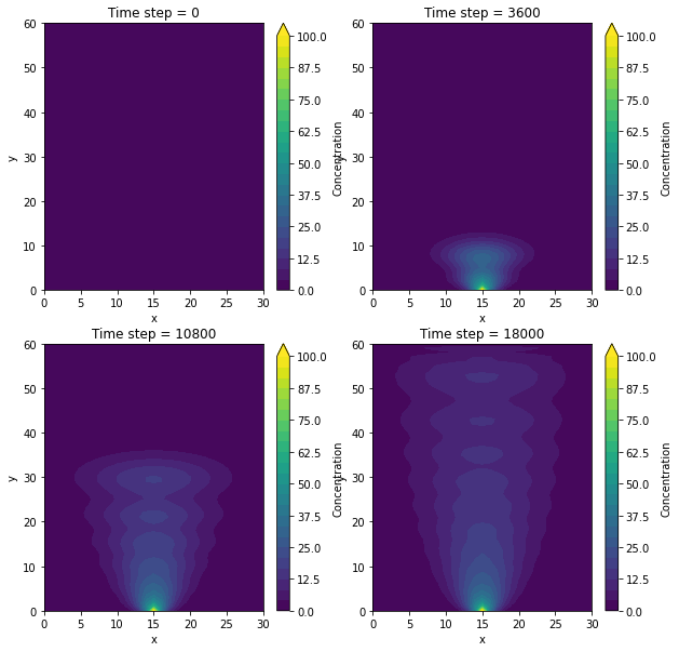
$$\frac{\Delta t \cdot K_y}{\Delta y^2} = 1 \, , \frac{|u| \cdot \Delta t}{\Delta x} = 0.1$$

This should mean that both terms are numerically stable and thus so is the solution. Now unfortunately I did not have the processing power to plot scatter graphs like the previous, So I opted to use contour graphs. I also ran into a runtime warning problem with my function.

To try and work around this I change my equation (I remove the dx,dy terms and replace them with the domains of x and y, this should fix the runtime error and because this is just adjusting constants the actual model that the equation creates should still be correct and because the adjustment will make the terms that determine stability smaller it should also preserve stability) while this new formulation is not perfect it lets me obtain a plot for advection/diffusion.

Using these conditions, I obtain the graphs (*as mentioned previously I filter out negative terms before plotting*),

Luke Kelly 100227404

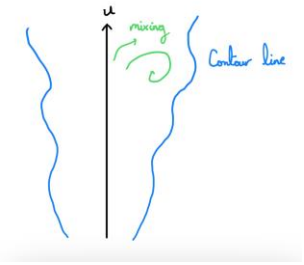


The model solution I obtained looks somewhat like what I would expect. The notable differences from what I would expect without comparison to materials or analytic solutions are as follows

-In the last time step the model has reached the end of the x domain and there is seemingly an artifact of this at the very top of the graph

-The graph contains oscillations in the concentration

The first of these problems is minor and can easily be resolved by increasing the size of the x domain, the second is more complicated to resolve using the previous analysis even accounting for changes to the equation there should still be numerically stable solutions. It is most likely roll over from the previously observed error. However, it also corresponds to an actual expected result if you consider mixing from eddies and how this may look when plotted by a contour. I drew a quick sketch of how this would visually appear.

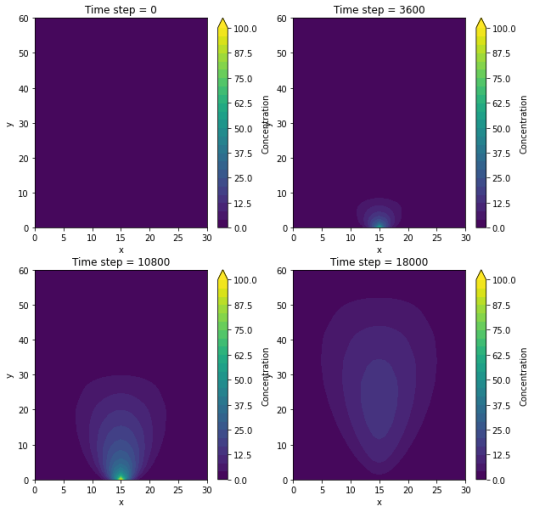


An expected result when comparing this result to the one for just diffusion is the fact that the introduction of advection has caused diffusion in a single layer to act much slower because there is smaller concentration of pollutant in any given layer because of advection.

To simulate the effect of the chimney starting to produce pollutants at the start of the model's run time, and gradually increasing emissions until midway through the model before decreasing to nothing at the end of the model's run time. I have set the emission rate equal to this function (See visualisation here),

$q = 100 \cdot \sin\left(\frac{t \cdot \pi}{Num\ time\ steps}\right)$ (*Do not need to use the abs function because in this domain sin is strictly positive*)

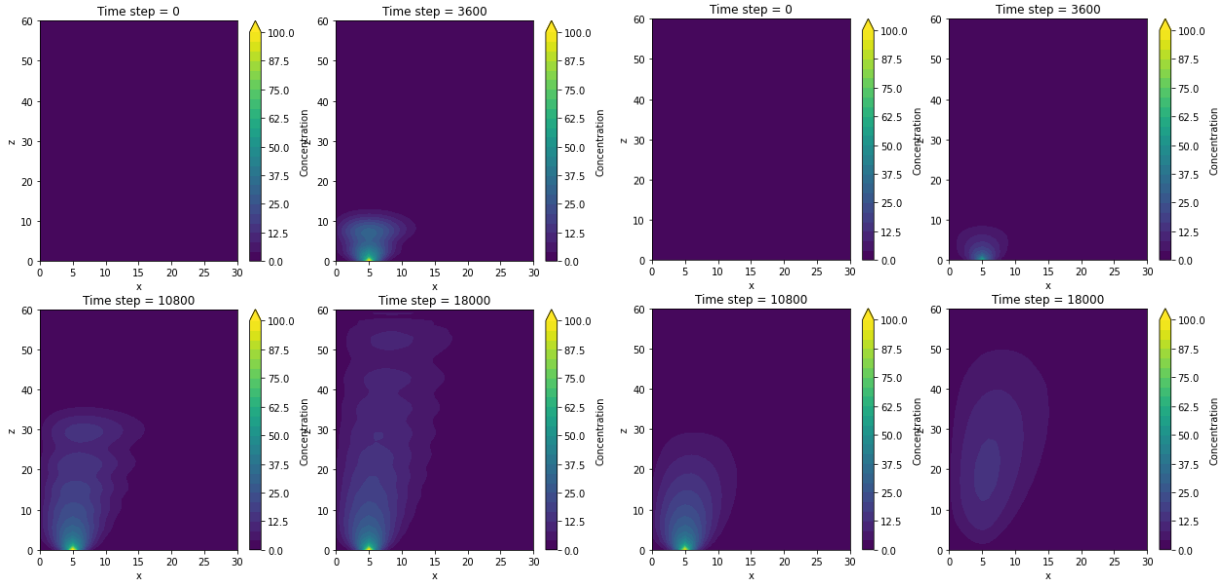
Obtaining the following graph,



What is noticeably different here is the complete lack of oscillations. If it is, as I put forward earlier that the oscillations are the result of a number handling error with python, then the considerably smaller amount of pollution introduced into this model over the same time steps would have this exact result, since as with the purely diffusion model above the handling error has not had the chance to impact the contours.

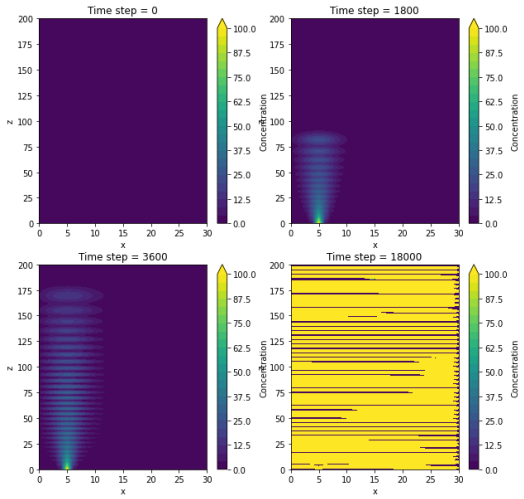
I now move onto considering the second equation I want to model, that being the diffusion advection system in x,z,t. The initial modeling and reasoning are skipped over since without the implementation of any conditions the model will be identical to the previous model.

I first consider that the change of introducing chimney height and a fixed boundary condition at the ground, since in the code the Dirac's delta is disregarded the implementation of the chimney height is done similarly to how the midpoint of y is introduced into the code. When it comes to the boundary condition at the ground the model considers a pollutant that does not settle (i.e. there is no diffusion process occurring over the boundary) hence the boundary condition can be taken as a row of zeros (*happily this is already a factor in the previous calculations*). Setting the chimney height to 5 and using all previous initial values produces the graphs.



As expected, diffusion is affected by the boundary condition creating a limiting factor on the spread of the pollutant at the ground. It also noticeably reduces the size of the oscillations previously observed this is most certainly just due to the compression of the available space for the oscillations to occur in and for the contours to map on to.

Now with this somewhat stable solution I examine what happens if the windspeed is increased (without changing the diffusivity coefficient). I set the windspeed to be 5 and increase the size of the x domain to 200 account for the expected results. Generating the graphs,



This is what would be expected from a much higher windspeed and corresponds to materials on the dispersion of pollutants from a chimney with this being much closer to the process known as fanning than the earlier lower windspeed models which were closer to the description of coning (Lew, J). Thus, backing up the validity of the model. And despite expanding the domain of x to try and account for the faster advective spread by the 5th hour the model has collapsed, demonstrating just how fast advection can act.

The last modification I model in this report is the re-inclusion of diffusion in the x direction. Taking the equation,

$$\frac{\partial C}{\partial t} + u \cdot \frac{\partial C}{\partial x} = K_z \frac{\partial^2 C}{\partial z^2} + K_x \frac{\partial^2 C}{\partial x^2}$$

And solving with an FTCS/CTCS mixed scheme as with previous yields,

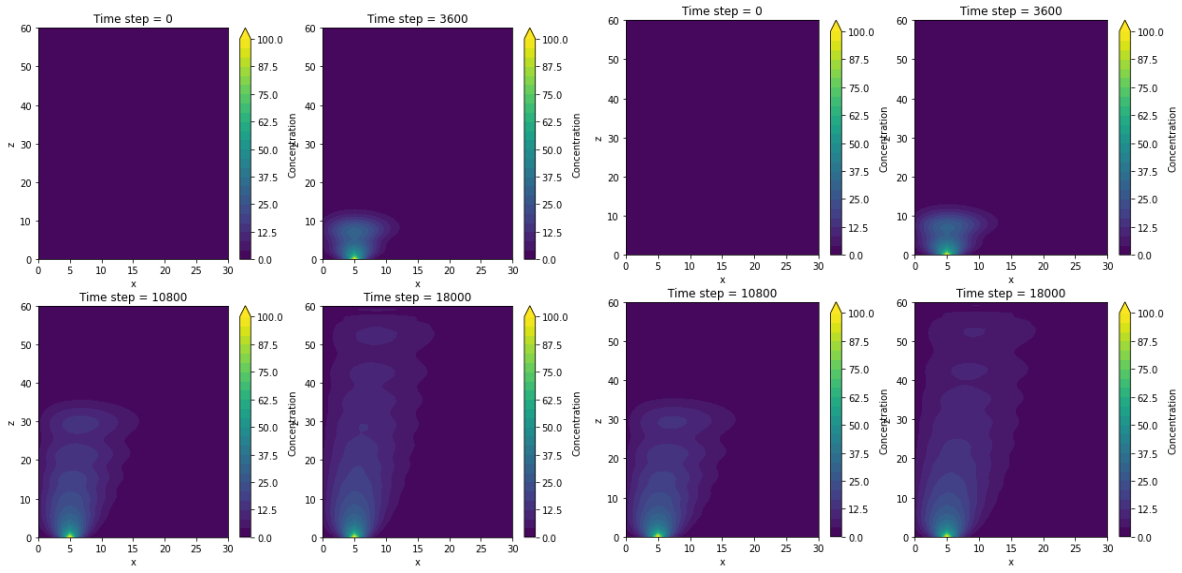
$$\frac{C_j^{n+1} - C_j^n}{2 \cdot \Delta t} + u \cdot \frac{C_{i+1}^n - C_{i-1}^n}{2 \cdot \Delta x} = K_y \cdot \left(\frac{C_{j+1}^{n-1} - 2C_j^{n-1} - C_{j+1}^{n-1}}{\Delta y^2} \right) + K_x \left(\frac{C_{i+1}^{n-1} - 2C_i^{n-1} + C_{i-1}^{n-1}}{\Delta x^2} \right)$$

Re-arranging,

$$C_j^{n+1} = C_j^{n-1} + 2\delta t \cdot \left(K_y \frac{C_{j+1}^{n-1} - 2C_j^{n-1} + C_{j-1}^{n-1}}{\delta y^2} + K_x \frac{C_{i+1}^{n-1} - 2C_i^{n-1} + C_{i-1}^{n-1}}{\delta x^2} - u \cdot \frac{C_{i+1}^n - C_{i-1}^n}{2\delta x} \right)$$

(Had to insert as an image because word was not including brackets)

The graph generated from this equation confirms the validity of making the earlier assumption that given advection in the x direction, and since advection is a first order derivative and diffusion is a second order derivative the inclusion of diffusion in the x direction has little to no impact on the model and increases the computation time significantly.



For comparison, the graph on the left is without diffusion in the x direction (seen previously) and the graph on the right has diffusion in the x direction

Due to computational limitations, I cannot extend either of my models much further in terms of domain size, or in terms of time steps. I do believe that given that my models correspond to the shape of a gaussian plumes (Bhattacharya, R / Uknown author) and previously stated reasoning I do believe they are reasonably accurate.

To conclude I will mention some avenues for future investigation that are not covered in the scope of this report. These in general, take one of two forms either examining an assumption I made in the course of constructing this model or an expansion of my model solution to improve some part of it.

Some of the first form are as follows;

-Assuming the local atmosphere is completely fixed. In the real world a pollutant from a chimney will certainly experience temperature change upon contact with the air and this temperature change will affect the rate of diffusion. Likewise but less dramatically, the windspeed and direction is going to be affected by the chimney itself acting as an obstacle so the windspeed will not be equal everywhere and will not be a field of perfectly parallel vectors.

-Assuming there is no pollutant removal. This is a safer assumption than the previous since this can often be the case when modelling with real pollutants. But this is not always the case and being able to explore this creates a more complete model and opens the potential for alternative boundary conditions from the ones used in the models above.

-Assuming there is no advective process in the z direction. This is again a safer assumption than the first but when advection does exist in the z direction it can have dramatic effects on models (Lew J) and change the shape of the dispersion entirely so it is worth investigating. There are also several different types of advective processes that can occur in z (deposition, vertical wind components, areas of updraft, settling, etc.) and comparing the impacts of these would be an interesting exploration.

-Assuming no topographical change on the plain which the chimney resides. The topography of the local area would have an effect on all 3 of the above-mentioned assumptions. Notably it would make setting a boundary condition for z a complicated endeavor.

-Assuming the chimney is a singularity. Whilst this assumption should not have a significant impact the effect of different types of chimney outlet and different numbers of chimneys would both be worth considering if you were comparing the model to real world chimney outputs. See source types (Bhattacharya, R)

Some of the second form are as follows;

- To model advection an FTCS for the first step and a CTCS term for all proceeding steps. This would mean instead of taking 2 initial time steps the model would only have to take one which could be useful but also considering the model runs for 1000 steps the impact of this adjustment is going to be minor.

-Runga-Kutta methods could be used to try and deal with the oscillatory error in the graphs. This would massively increase computation time and would likely not have much effect on the overall shape of the graphs but would be something worth doing if you were going to run the simulation for larger domains where the error will become more significant.

-If you had access to more computational power, increasing the fidelity of the graphs or employing scatter graphs as I attempted earlier in the report would give a more complete picture of the processes occurring in the model.

References:

Sharan, et al. 1996: A mathematical model for dispersion of air pollutants in low wind conditions, *Atmospheric Environment*, **30,** 1209-1220.

Bhattacharya, R. *Atmospheric Dispersion* [PowerPoint presentation] Available at: https://ansn.iaea.org/Common/Topics/OpenTopic.aspx?ID=13012 (Last Accessed: 22 December 2022)

Lew, J. *Chapter 18 - Air Pollution* [Website / lecture notes collection] Available at: https://apollo.nvu.vsc.edu/classes/met130/notes/chapter18/index.html (Last Accessed: 22 December 2022)

Uknown author. *Gaussian Plumes* [Self study notes] Available at: https://www.eng.uwo.ca/people/esavory/gaussian%20plumes.pdf (Last Accessed: 22 December 2022)

~~~~~~~~~~~~~~~

Code:

```
# -*- coding: utf-8 -*-
"""
Created on Wed Dec 21 17:42:12 2022

@author: LODKe

"""

"""#Signature
Print("Started making it,")
Print("Had a breakdown")
Print("Bon Appétit")
"""
```
~~~~~~~~~~~~~~~

```
#Import neccessary modules
import math as m
import numpy as np
import matplotlib.pyplot as plt

#Used All caps for constants, and snake_case for variables

#Set up initiual values
EMISSION_RATE = 100                #Set up  Emmision rate
DIFFUSIVITY_COEFFICIENT_X = 1      #Diffusivity co-effiecent in x
DIFFUSIVITY_COEFFICIENT_Y = 1      #Diffusivity co-effiecent in y
DIFFUSIVITY_COEFFICIENT_Z = 1      #Diffusivity co-effiecent in z
WIND_SPEED = 5                     #Windspeed in direction of x-axis

TIME_STEP = 1                      #How many seconds in each time step
SECOND = 1
HOUR = 60 * 60 * SECOND
DAY = 24 * HOUR
WEEK = 7 * DAY

NUM_TIME_STEPS = 5 * HOUR          #The number of steps we model for

X_DOMAIN = 200                     #Set up domain for x
Y_DOMAIN = 30                      #Set up domain for y
Z_DOMAIN = 30                      #Set up domain for z

#Setting the steps equal to the domains to simplify our equations
X_STEPS = X_DOMAIN
Y_STEPS = Y_DOMAIN
Z_STEPS = Z_DOMAIN
```

```
Dx = X_DOMAIN/X_STEPS
Dy = Y_DOMAIN/Y_STEPS
Dz = Z_DOMAIN/Z_STEPS

Mid_point_y = int(Y_DOMAIN / 2)     #Find a place for the chimney in y
CHIMNEY_HEIGHT = 5                  #Set a height for the chimney



#Create an array for concentration
C = np.zeros((NUM_TIME_STEPS+1, X_DOMAIN+1, Y_DOMAIN+1))

"""# Our Equation for working out diffusion/advection for x,y
for n in range(1,NUM_TIME_STEPS):
    #loop for 1=<t
    EMISSION_CHANGE = EMISSION_RATE * m.sin((n*m.pi)/(NUM_TIME_STEPS))
    for i in range(X_DOMAIN):
        #loop for 0=<x
        for j in range(Y_DOMAIN):
            #loop for 0=<y
            #print("C[",n,",",i,",",j,"]=",C[n,i,j]) #reads our array element by element


                #Set boundary "condition" for source of pollution
                #C[n,0,Mid_point_y] = EMISSION_RATE
                C[n,0,Mid_point_y] = EMISSION_CHANGE


                #Just diffusion in y
                #C[n+1,i,j] = C[n,i,j] + TIME_STEP * DIFFUSIVITY_COEFFICIENT_Y * ((C[n-1,i,j+1]-2*C[n-
1,i,j]+C[n-1,i,j-1])/(Y_DOMAIN**2))
```

```
            #Input our equation for advection/diffusion

            C[n+1,i,j] = (C[n,i,j] + 2 * TIME_STEP * ((DIFFUSIVITY_COEFFICIENT_Y * ((C[n-1,i,j+1]-2*C[n-
1,i,j]+C[n-1,i,j-1])/(Y_DOMAIN**2))) - WIND_SPEED*((C[n,i+1,j]-C[n,i-1,j])/(X_DOMAIN*2))))

"""


"""# Our Equation for working out diffusion/advection for x,z

for n in range(1,NUM_TIME_STEPS):

   #loop for 1=<t

   EMISSION_CHANGE = EMISSION_RATE * m.sin((2*n*m.pi)/(NUM_TIME_STEPS))

   for i in range(X_DOMAIN):

      #loop for 0=<x

      for j in range(Z_DOMAIN):

         #loop for 0=<y

         #print("C[",n,",",i,",",j,"]=",C[n,i,j]) #reads our array element by element


            #Set boundary "condition" for source of pollution

            C[n,0,CHIMNEY_HEIGHT] = EMISSION_RATE

            #C[n,0,CHIMNEY_HEIGHT] = EMISSION_CHANGE


            #Just diffusion in z

            #C[n+1,i,j] = C[n,i,j] + TIME_STEP * DIFFUSIVITY_COEFFICIENT_Z * ((C[n-1,i,j+1]-2*C[n-
1,i,j]+C[n-1,i,j-1])/(Z_DOMAIN**2))


            #Input our equation for advection/diffusion

            C[n+1,i,j] = (C[n,i,j] + 2 * TIME_STEP * ((DIFFUSIVITY_COEFFICIENT_Z * ((C[n-1,i,j+1]-2*C[n-
1,i,j]+C[n-1,i,j-1])/(Z_DOMAIN**2))) - WIND_SPEED*((C[n,i+1,j]-C[n,i-1,j])/(X_DOMAIN*2))))

"""


"""# Our Equation for working out diffusion/advection for x,z with x diffusion

for n in range(1,NUM_TIME_STEPS):

   #loop for 1=<t
```

```
    EMISSION_CHANGE = EMISSION_RATE * m.sin((2*n*m.pi)/(NUM_TIME_STEPS))
    for i in range(X_DOMAIN):
        #loop for 0=<x
        for j in range(Z_DOMAIN):
            #loop for 0=<y
            #print("C[",n,",",i,",",j,"]=",C[n,i,j]) #reads our array element by element


                #Set boundary "condition" for source of pollution
                C[n,0,CHIMNEY_HEIGHT] = EMISSION_RATE
                #C[n,0,CHIMNEY_HEIGHT] = EMISSION_CHANGE


                #Just diffusion in z
                #C[n+1,i,j] = C[n,i,j] + TIME_STEP * DIFFUSIVITY_COEFFICIENT_Z * ((C[n-1,i,j+1]-2*C[n-
1,i,j]+C[n-1,i,j-1])/(Z_DOMAIN**2))


                #Input our equation for advection/diffusion
                C[n+1,i,j] = (C[n,i,j] + 2 * TIME_STEP * ((DIFFUSIVITY_COEFFICIENT_Z * ((C[n-1,i,j+1]-2*C[n-
1,i,j]+C[n-1,i,j-1])/(Z_DOMAIN**2))) + (DIFFUSIVITY_COEFFICIENT_X * ((C[n-1,i,j+1]-2*C[n-1,i,j]+C[n-
1,i,j-1])/(X_DOMAIN**2))) - WIND_SPEED*((C[n,i+1,j]-C[n,i-1,j])/(X_DOMAIN*2))))
"""



C_copy = C.clip(min=0)          #This eliminates any posible negative values in our array which makes graphing
easier



"""#Scatter Graph Function
fig = plt.figure()
ax = fig.add_subplot(projection='3d')


for x in range(X_DOMAIN):
```

```
    for y in range(Y_DOMAIN):
        for t in range(NUM_TIME_STEPS):
            ax.scatter(x,t,y, s=C_copy[t,x,y])
            ax.set_xlabel("Variation in x")
            ax.set_ylabel("Time")
            ax.set_zlabel("Variation in y")
            ax.set_title("Concentration of pollutant")
plt.show()
"""

"""#Contour graph funciton
time_steps = [0,1800,HOUR,5 * HOUR]

fig, axes = plt.subplots(2,2,figsize=(10,10))
v = np.linspace(0, 100, 25)

for idx, ax in enumerate(axes.ravel()):

    cont = ax.contourf(C_copy[time_steps[idx],:,:], v, extend="max")
    plt.colorbar(cont, ax=ax,label="Concentration")
    ax.set_xlabel("x")
    ax.set_ylabel("z")
    ax.set_title(f"Time step = {time_steps[idx]}")

plt.show()

#plot.plot(C[15,0,:],z,'b-o',)
#for i in time_steps:
#    print(C[i,:,:])
"""
```