Warren Keil

Path Tubes Method of Numerical Approximation of Partial Differential Equations

April 22, 2018

1 Introduction

In this presentation, I will present a newly proposed method [1, 2, 3, 4] of numerical solving partial differential equations called the path tubes method. Numerical techniques are very important for solving partial differential equations for a couple of reasons. For one, partial differential equations can be very difficult to solve analytically. A large group of PDEs, namely non-linear PDEs, can be incredibly hard to solve. Thus arriving at solutions numerically is very beneficial to those using these equations. The other reason numerical solutions are so important is because partial differential equations are fundamental in so many areas of science. PDEs provide the underlying structure for areas such as quantum mechanics, fluid mechanics, biological modeling and many others areas of research.

First, I will give an intuitive description of what path tubes are for some general problem. I will provide the derivation of the general path tubes method not applied to any particular partial differential equation.

Next, I will give a background review on the advection equation

$$\frac{\partial C}{\partial t} + \nabla \cdot (C\mathbf{v}) = 0$$

t

by describing each the terms in this equation. Since the papers I am presenting from from the field of engineering science, I will provide explanations for the differences of the notation from these papers and the equations presented in the text, *Applied Partial Differential Equations* by J. David Logan.

Subsequently, I will formulate the path tubes method as applied to the advection equation. This will involve taking an arbitrary point in the space and deriving the path it takes over an interval of time. After deriving the path tubes method, I will then provide the steps required to discretize the space and give us an algorithm to numerically solve the advection equation.

Next, I will describe the terms of the Navier-Stokes equation

$$\frac{\partial V}{\partial t} + (\nabla V) \cdot V - \frac{1}{Re}\Delta V + \nabla p = \mathcal{F}$$

I will then formulate the path tubes method applied to the Navier-Stokes equation and show its corresponding discretization method.

I then will show applications of both of these methods by coding these algorithms in python and testing it out on equations with easily accessible solutions so that the approximations may be compared to the known solutions.

2 Methods

2.1 The Path Tubes Method

The path tubes method was invented using the idea that when modeling phenomena over continuous media, we can take any arbitrary 'particle' of substance in our model and calculate the path it takes in the spatial dimensions as time t changes [1, 2]. To express this mathematically, let t be the time variable so that $t \in \mathbb{R}$. Let Ω be some region in 3-space, $\Omega \in \mathbb{R}^3$. Then we can look at the change of the spatial position of Ω by looking at some point in $\zeta \in \Omega$ and follow its change as t changes. We track this change by representing its position by a vector $\mathbf{x} = \mathbf{x}(\zeta, t)$. Thus, the path of the particle ζ is given by the vector $\mathbf{x}$ at each point in time t. The path written in set notation is

$$\mathfrak{T}_\zeta = \{(\mathbf{x}, t) \in \mathbb{R}^3 \times \mathbb{R} : \mathbf{x} = \mathbf{x}(\zeta, t) \in \Omega_t, \ \ t \in \mathbb{R}\}$$

To give an explanation of this set we have $\mathfrak{T}_\zeta$ is the set of all points from 3-space together with each time t such that the x_1, x_2, x_3 coordinates at a given time t are given by the position vector $\mathbf{x}$ where $\mathbf{x}$ is a function of ζ and time.

Next from vector calculus, the parametric equations of this line is given by the differential equations

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}$$
$$\Rightarrow \frac{dx_1}{dt} = v_{x_1}(x_1, x_2, x_3, t) \qquad \frac{dx_2}{dt} = v_{x_2}(x_1, x_2, x_3, t) \qquad \frac{dx_3}{dt} = v_{x_3}(x_1, x_2, x_3, t)$$

We can picture this equation as the velocity field $\mathbf{v}$ as consisting of a vector for each point in our space. The derivative of $\mathbf{x}$ for each spatial dimension with respect to time is the velocity vector for each point projected onto elementary vectors pointing in each of the n dimensions we are working in.

The additional piece of information used to solve this system of differential equations is the assumption that the position of a particle can be determined solely by the change in time.

$$\mathbf{x}(\zeta, \hat{t}) = \hat{\mathbf{x}}$$

Next the definition of path tubes [1, 2, 4] can now be given as the *region in spacetime formed by the union of path lines relative to all particles ζ of the given material body Ω* A path tube is denoted by big sigma Σ and the path tube of a specific region is written as

$$\Sigma = \bigcup_{\zeta \in \Omega} \mathfrak{T}_\zeta.$$

2.2 The Advection Equation

The advection equation used in our problem is

$$\frac{\partial C}{\partial t} + \nabla \cdot (C\mathbf{v}) = 0.$$

This version of the equation is slightly different the the advection equation found in the text by Logan. To describe each of the terms of this equation, we have:

$C = C(\mathbf{x}, t)$	State variable. Concentration of the substance.
t	Time
∇	Vector of first partial derivatives of each of the spatial variables
$\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$	Velocity field. Gives a constant of velocity for each direction for each point of space.

For the purposes of this paper, we are going to be working in one spatial dimension. Thus the ∇ will be reduced to $\nabla = \frac{\partial}{\partial x}$, $\mathbf{x}$ will be reduced to x, and the velocity field will become $\mathbf{v} = v(x, t)$. We can now write the advection equation as

$$\frac{\partial C}{\partial t} + \frac{\partial}{\partial x}(Cv) = 0.$$

Next, notice that the set notation of a path tube in one dimension becomes

$$\mathfrak{T}_\zeta = \{(x,t) \in \mathbb{R} \times \mathbb{R}; x = x(\zeta, t) \in \Omega_t, t \in \mathbb{R}\}.$$

We then integrate the 1-D advection equation $\frac{\partial C}{\partial t} + \frac{\partial}{\partial x}(Cv)$ along the path tube of a time interval $[t, t+\Delta t]$

$$\Sigma = \bigcup_{\zeta \in \Omega} \{(x, \hat{t}) \in \mathbb{R} \times \mathbb{R}; x = (x(\zeta, \hat{t}) \in \Omega_{\hat{t}} = [x_1(\hat{t}), x_2(\hat{t})], t \leq, \hat{t} \leq t + \Delta t\}$$

To explain each of these terms of the path tube, we have that Σ is the set of all ordered pairs of x and $\hat{t}$ such that x is a real number and t is in the closed interval $[t, t+\Delta t]$. We also explicitly stated that the region where $\hat{t}$ lives is $\Omega_{\hat{t}} = [x_1(\hat{t}), x_2(\hat{t})]$ This makes sense that Ω is a line segment since we are in one dimension. So continuing on to integrate the advection equation along the path tube, we get

$$\int_t^{t+\Delta t} \int_{x_1(t)}^{x_2(t)} \left[\frac{\partial C}{\partial t} + \frac{\partial (vC)}{\partial x}\right] dxdt = 0$$

Expanding this out, we get

$$\int_t^{t+\Delta t} \int_{x_1(t)}^{x_2(t)} \left[\frac{\partial C}{\partial t} + \frac{\partial (vC)}{\partial x}\right] dxdt = 0 \tag{1}$$

$$\int_t^{t+\Delta t} \left[\int_{x_1(t)}^{x_2(t)} \frac{\partial C}{\partial t} dx + \int_{x_1(t)}^{x_2(t)} \frac{\partial (vC)}{\partial x} dx\right] dt = 0 \tag{2}$$

$$\int_t^{t+\Delta t} \left[\int_{x_1(t)}^{x_2(t)} \frac{\partial C}{\partial t} dx + [(vC)\Big|_{x_1(t)}^{x_2(t)}\right] dt = 0 \tag{3}$$

$$\int_t^{t+\Delta t} \left[\int_{x_1(t)}^{x_2(t)} \frac{\partial C}{\partial t} dx + [v(x_2(t),t)C(x_2(t),t) - v(x_1(t),t)C(x_1(t),t)]\right] dt = 0 \quad \text{since } v,\ C \text{ are functions of } x \tag{4}$$

Next, before proceeding, recall Leibniz's Integration formula,

$$\frac{d}{dx}\left(\int_a^b f(x,t)dt\right) = f(x,b(x)) \cdot \frac{d}{dx}b(x) - f(x,a(x)) \cdot \frac{d}{dx}a(x) + \int_{a(x)}^{b(x)} \frac{\partial}{\partial x} f(x,t)dt.$$

Rewriting Leibniz's formula by swapping xs and ts we get,

$$\frac{d}{dt}\left(\int_a^b f(x,t)dx\right) = f(b(t),t) \cdot \frac{d}{dt}b(t) - f(a(t),t) \cdot \frac{d}{dt}a(t) + \int_{a(t)}^{b(t)} \frac{\partial}{\partial t} f(x,t)dx \qquad (\star)$$

Next we observe the following things about equation (4). The limits of integration $x_1(t), x_2(t)$ are functions of time since the path tube is not necessarily a straight tube. Thus, when we differentiate these limits of integration with respect to time, we get that their derivative is $\frac{d}{dt}x_1(t) = v(x_1(t),t)$ and $\frac{d}{dt}x_2(t) = v(x_2(t),t)$ since x_1, x_2 are determined by the velocity field v. So now if we let $f(x,t)$ in equation $(\star)$ become $C(x,t)$ then Leibniz formula applied to the inside of equation (4) becomes

$$\int_{x_1(t)}^{x_2(t)} \frac{\partial C}{\partial t} dx + [v(x_2(t),t)C(x_2(t),t) - v(x_1(t),t)C(x_1(t),t)] = \frac{d}{dt}\int_{x_1(t)}^{x_2(t)} C(x,t)dx$$

Piecing this together with the rest of equation (4), we get

$$\int_t^{t+\Delta t}\left[\frac{d}{dt}\int_{x_1(t)}^{x_2(t)} C(x,t)dx\right]dt = 0$$

$$\int_{x_1(t)}^{x_2(t)} C(x,t)dx\Big|_t^{t+\Delta t} = 0$$

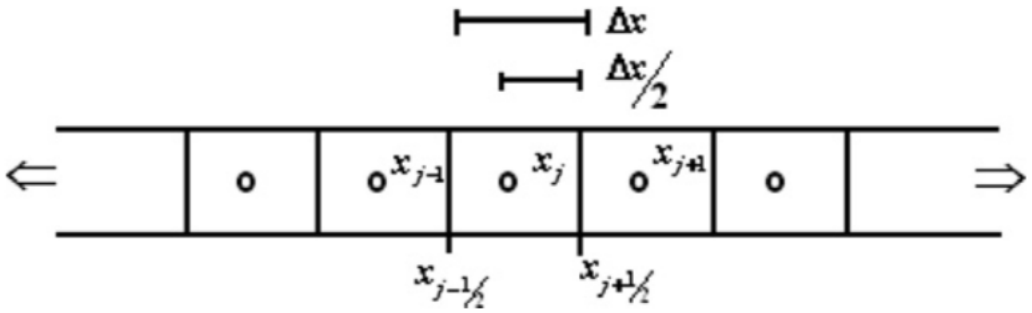
$$\int_{x_1(t+\Delta t)}^{x_2(t+\Delta t)} C(x,t+\Delta t)dx - \int_{x_1(t)}^{x_2(t)} C(x,t)dx = 0 \qquad \text{by FTC}$$

$$\int_{x_1(t+\Delta t)}^{x_2(t+\Delta t)} C(x,t+\Delta t)dx = \int_{x_1(t)}^{x_2(t)} C(x,t)dx \qquad \clubsuit$$

Now we observe that this last equation shows that C represents a concentration with units, mass/volume, and thus when we integrate out the spatial variable x, we are left with just mass. Thus, the equation $\clubsuit$ show that the mass is equal for any two different times, therefore the path tubes method conserves mass. The equation $\clubsuit$ is called the Langrangian formulation.

2.3 Discretization

To discretize the 1-dimensional space, we set a step length Δx and form evenly spaced cells across the x axis for some time t. This is denoted $I_j = [x_{j-1/2}, x_{j+1/2}]$. It should be noted that since we are in 1 dimension, then it follows that the sideways length of these cells will be along the x-axis and the vertical length is along the t-axis. We also will describe the adjacent x cells along the t axis as $x(t+\Delta t) = \tilde{x}$.



It follows that the edges of these cells can be found deterministically by observing the line integrals *backswards through time* of the preceding cells denoted x_- and x_+. Each of these cell walls on the x axis are found by solving $dx/dt = v(x,t)$ with the condition $\tilde{x} = x_{i-1/2}$ and $\tilde{x} = x_{i+1/2}$. The Langrangian formulation for this discretization now becomes

$$\int_{x_{j-1/2}}^{x_{j+1/2}} C(x,t+\Delta t)dx = \int_{x_-}^{x_+} C(x,t)dx.$$

It follows that for any time instant $t+\Delta t$, we can find the concentration C_j^{n+1}

$$C_j^{n+1} = \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} C(x,t+\Delta t)dx.$$

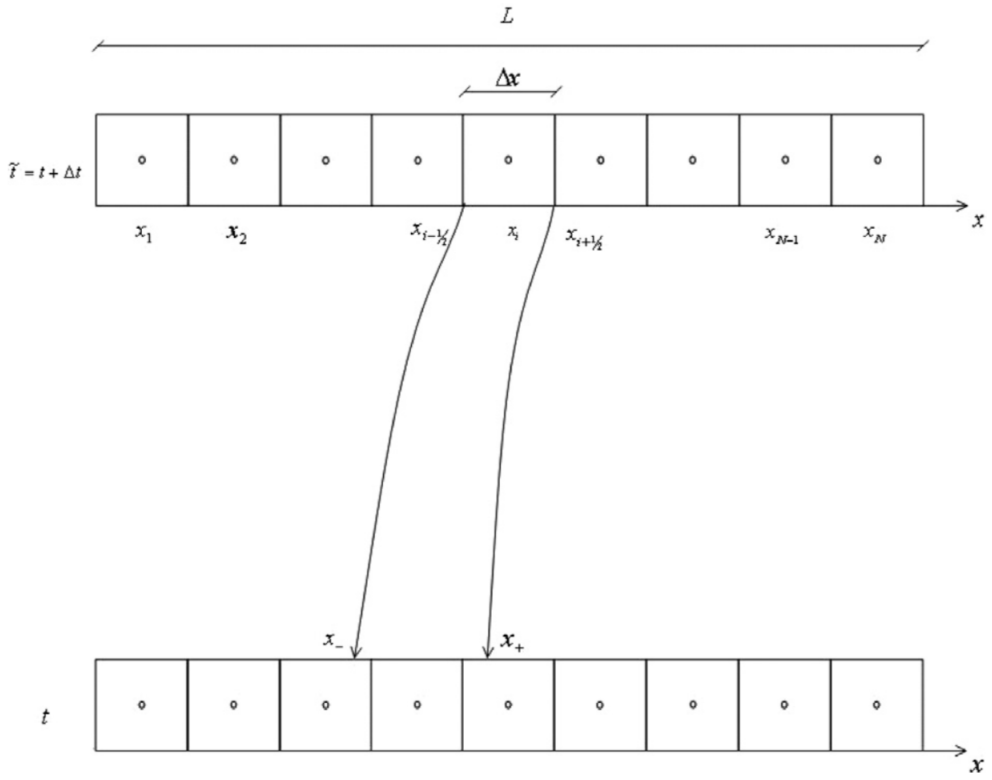


Fig. 3. A path tube in $\mathbb{R} \times \mathbb{R}$.

And by the discretized Langrangian formulation, we obtain

$$C_j^{n+1} = \frac{1}{\Delta x} \int_{x_-}^{x_+} C(x,t)dx.$$

To evaluate this integral, the authors used the Gaussian quadrature method of numerically evaluating integrals. This method can be found in a most texts on numerical methods. The numerical approximation of the integral is (using the weights that the authors deemed appropriate)

$$\int_{x_-}^{x_+} C(x,t)dx \approx \frac{(x_+ - x_-)}{18}[5(C_a + C_b) + 8C_c].$$

Filling the right hand side of this equation, we have $C_a \equiv C(x_a,t), C_b \equiv C(x_b,t), C_c \equiv C(x_c,t)$ with

$$\begin{aligned} x_a &= (x_+ + x_-)/2 - [(x_+ - x_-)/2)]\sqrt{3/5} \\ x_b &= (x_+ + x_-)/2 - [(x_+ + x_-)/2)]\sqrt{3/5} \\ x_c &= (x_+ + x_-)/2 \end{aligned}$$

To summarize this discretization scheme so far, the above mentioned iterative calculations are executed with only knowing the discrete points in the middle of each cell. We then calculate x_a, x_b and x_c. We then can calculate the next row of cells using the Gaussian quadrature method of approximating our discretized Langrangian formulation.

One last step needed before implementing this method is that the term C_a need a slightly tweaking in order to avoid potential oscillatory behavior. The authors credited their own experiences for noticing and solving this problem [2]. Since this is not part of the main ideas of this paper, we will not go into detail about the analysis of the cause and solution for this erroneous behavior, but the solution of this sub-problem is as follows.

If $x_a \in I_a = [x_{q-1/2}, x_{q+1/2}]$, then $C(x_a, t) = C_a$ is found by the following

$$C_a = \begin{Bmatrix} m, & \text{if } P_3(x_a) < m \\ P_3(x_a), & m \leq P_3(x_a) \leq M \\ M, & \text{if } P_3(x_a) > M \end{Bmatrix}$$

with

$$P_3(x_a) = \sum_{i=q-1}^{q+2} C_i^{(n)} \prod_{\substack{k=q-1 \\ k \neq i}}^{q+2} \frac{x - x_k}{x_i - x_k}$$

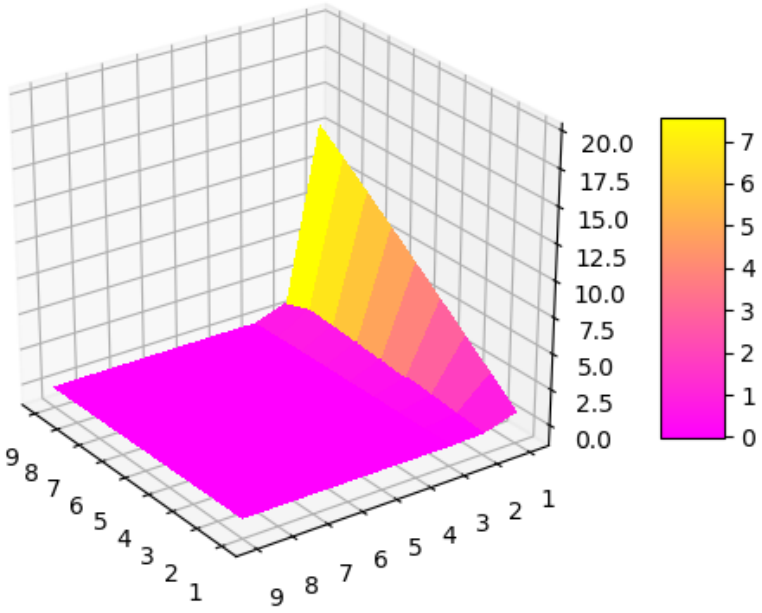
and m and M are given by

$$m = \min\{C_{q-1}^{(n)}, C_q^{(n)}, C_{q+1}^{(n)}, C_{q+2}^{(n)}\},$$
$$M = \max\{C_{q-1}^{(n)}, C_q^{(n)}, C_{q+1}^{(n)}, C_{q+2}^{(n)}\}.$$

The error analysis of both this intermediate interpolation and the entire approximation scheme can be found in the following papers [1, 2, 3, 4]. The stability analysis can be found in these articles as well.

2.4 Implementation

After spending quite some time coding this implementation of this discretization scheme on both python and mathematica, I decided to continue with my python code. While the steps in producing the algorithm were simple enough, the output I received did not appear to match known properties of advection only surface plots. One reason for this is the authors did mention that they took one additional step with the piecewise function mentioned above to reduce oscillations in the plot. However, I did not pursue adding this to my code because the problems I experience were that my C values rapidly decreased as time increase. And since I do not have a diffusive term in this equation, then it was totally unexpected.



3 Future Work

There are several follow up tasks and projects regarding these papers that can be performed in the future. For one the path tubes method can be extended to be applied to other types of PDEs. According to the four papers I have reviewed, the creators of the path tubes method have applied their technique to the advection equation, and the Navier-Stokes equation so far [1, 2, 4]. It is natural to ask would this method apply nicely to other PDEs such as the wave equation or the heat equation? Another area of future work is to further analyze how this approximation technique compares with other methods. Is it computationally costly when tackling large problem? Is there any scenarios when the path tubes method become unstable and produces nonsensical results? The authors have included a good bit of this type of analysis in the papers but by no means have they exhausted this subject. And furthermore, one could spend time writing stable software libraries for this technique in various programming languages. Languages such as R, python, and matlab are widely used in academia and in industry and people could benefit greatly if someone were to write stable functions in these languages.

For my personal future work, I plan on coding the algorithm from scratch again with the goal of getting a more expected result. The authors were very brief and concise in their description of the code by I am sure I can reproduce their plots if I invest more time in the project.

4 Conclusion

The path tubes method has shown to be a very interesting method of numerically approximating certain partial differential equations. It was interesting since this method took a totally different approach in its derivation than the finite difference approximation method as shown in the text by Logan. It was also notable that the path tube method has led its creators to publish four papers on the subject which implies that (at least according to them) this method holds good potential to be useful in the future. It was also enlightening to read so many academic papers while choosing this topic. I printed off and read 11 papers in total and learned a lot. I got the chance to appreciate how much information is in a mathematical journal article. It is refreshing to read these articles knowing a great deal of hard work and ingenuity went in to producing these works. While some of these papers can be daunting at first, they make sense as soon as you take the time to break each part of the paper down. And lastly, I got to see just how vast the subject of PDEs really is. When searching for these papers, I saw hundreds of different papers about using PDEs for a wide variety of subjects. These papers on PDEs ranges from pure theoretical standpoints to all kinds of numerical topics and also a wide variety of modeling topics. The subject of partial differential equations of definitely at forefront of a bunch of different math-related topics and is worth spending time to study.

References

[1] Nèlio Henderson, Marcelo Sampaio, Luciana Pena *Path Tubes Method: A Semi-Langrangian Approach for Linear Advection Equations*. Elsevier Journal of Chemical Engineering Science, `www.elsevier.com/locate/ces`

[2] Nèlio Henderson, Marcelo Sampaio, Luciana Pena *Developing New Approaches for the Path Tubes Method*. Elsevier Journal of Applied Mathematical Modeling, `www.elsevier.com/locate/apm`

[3] Nèlio Henderson, Luciana Pena *The Inverse Distance Weighted Interpolation Applied to a Particular Form of the Path Tubes Method: Theory and Computation for Advection in Incompressible Flow*. Elsevier Journal of Applied Mathematics and Computation, `www.elsevier.com/locate/amc`

[4] Nèlio Henderson, Fàbio Pacheco, Mauricio Kschinhevsky *Numerical Solutions of the Navier-Stokes Equations Using the Path Tubes Method*. Elsevier Journal of Chemical Engineering Science, `www.elsevier.com/locate/ces`

5 Python Code

```
"""
@author: warrenkeil
"""
import random
import sys
import os
import math
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import IPython
import scipy as sci
import pylab
import sympy as sym
import dionysus as d
import csv
import networkx as nx
from mpl_toolkits.mplot3d import Axes3D
  #   starting this model on [0,1] interval  !!

bigC = 1      # advection constant
bigN = 9     # number of time steps
bigJ = 9    # number of intervals to break space
h = (1/bigJ)   # space step size
k = .005     #  time step size
r = (k*bigC)/(h**2)   # stability, not sure it works here.

x = np.zeros(((bigN),(bigJ+1)))  # these are edges of each cell


mid =  [0]*(bigJ)   #This is the x values of mid point of cells
mid[0] = (1/(2*bigJ))

for i in range(1,bigJ):
    mid[i]=mid[i-1] + (1/bigJ)



u = np.zeros((bigN, bigJ))  #  matrix  This is state var C!!!

# set initial condition: starting at zero. can change later.

for i in range(0,np.shape(u)[1]):
    u[0,i] =  3*np.sin(mid[i])+mid[i]*(11)



# Set up our x edgepoints based on initial condition
for i in range(0,np.shape(x)[1]):
    if i==0:
        x[0,i] = u[0,0]
    elif i==(np.shape(x)[1]-1):
```

```
        x[0,i] = u[0,bigN-1]
    else:
        x[0,i]= (.5)*(u[0,i-1]+u[0,i])


# Set Gauss quadrature approx

for i in range(1,np.shape(u)[0]):  #### i    row
    for j in range(0,np.shape(u)[1]):     ####   j column
        a=(x[i-1,j+1]+x[i-1,j])*(.5)-(x[i-1,j+1]-x[i-1,j])*(.5)*np.sqrt(3/5)
        b=(x[i-1,j+1]+x[i-1,j])*(.5)-(x[i-1,j+1]+x[i-1,j])*(.5)*np.sqrt(3/5)
        c=(x[i-1,j+1]+x[i-1,j])*(.5)


        u[i,j]=(1/bigJ)*(1/18)*(x[i-1,j+1]-x[i-1,j])*(5*(a+b)+8*c)


        if j==0:
            x[i,j] = u[i,0]
        else:
            x[i,j]= (.5)*(u[i,j-1]+u[i,j])

    x[i,bigJ] = u[i,bigJ-1]

# PLOTTING

X = np.arange(1, 10)
Y = np.arange(1, 10)
X, Y = np.meshgrid(X, Y)
#R = np.sqrt(X**2 + Y**2)
#Z = np.sin(R)
fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(X, Y, u, rstride=1, cstride=1, cmap='spring', linewidth=0, antialiased=False)
ax.set_zlim(-1.01, 20.01)

fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()

'''
X = np.arange(1, 10)
Y = np.arange(1, 10)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='hot', linewidth=0, antialiased=False)
ax.set_zlim(-1.01, 1.01)

fig.colorbar(surf, shrink=0.5, aspect=5)
plt.show()
```