Christopher Lum
lum@uw.edu

Lecture08e
Numerically Solving Partial Differential Equations

Lecture is on YouTube

The YouTube video entitled ‘Numerically Solving Partial Differential Equations’ that covers this lecture is located at https://youtu.be/ZSNl5crAvsw.

Outline

Introduction

In many cases, it is difficult or impossible to solve a given PDE analytically. In these situations, it becomes necessary to utilize numerical techniques to obtain approximate solutions to the problem.

Consider the Fokker-Planck equation (Wikipedia link) which is a partial differential equation that can be used to describes the time evolution of the probability density function of the velocity of a particle under the influence of drag forces and random forces, as in Brownian motion.

If there is one temporal dimension (t) and one spatial dimension (x), one can derive the simplest form of a diffusion equation from the Fokker-Planck equation to be a PDE as shown below

$$\frac{\partial p(t,x)}{\partial t} = \frac{1}{2}\frac{\partial^2 p(t,x)}{\partial x^2} \qquad \textbf{(Eq.1)}$$

Minor Note: The list of the independent variables with *t* before *x* is somewhat different from previous notes and other literature but this ordering is a matter of preference and the currently used ordering will prove useful when translating results to software algorithms.

Analytical Solution

For certain initial and boundary conditions, Eq.1 has an analytical solution. For example, if the initial condition is $p(0, x) = \delta(x)$ (Dirac delta function) and there are no explicit constraints on the boundary conditions, the analytical solution is given by

$$p(t, x) = \frac{1}{(2\pi t)^{1/2}} e^{-x^2/(2t)} \qquad \textbf{(Eq.2)}$$

```
In[•]:= p[t_, x_] = 1/(2 π t)^(1/2) Exp[-x^2/(2 t)]
```

```
Out[•]=
```
$$\frac{e^{-\frac{x^2}{2t}}}{\sqrt{2\pi}\ \sqrt{t}}$$

Note that this is a Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = t^{1/2}$.

```
In[•]:= (*Define a Gaussian distribution*)
        gaussian[x_, μ_, σ_] = 1/(σ (2 π)^(1/2)) Exp[-1/2 ((x - μ)/σ)^2];

        (*Check that the solution to the Fokker-Planck is a Gaussian with μ=0 and σ=t^(1/2)*)
        gaussian[x, 0, t^(1/2)] == p[t, x]
        Clear[f]
Out[•]= True
```

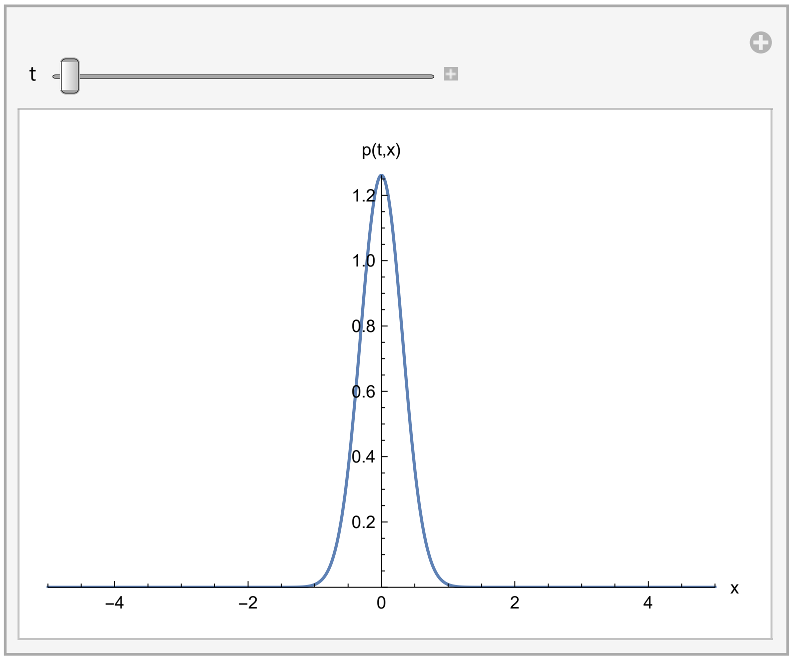
Let us verify that Eq.2 is a solution to the original PDE

```
In[•]:= D[p[t, x], t] == 1/2 D[p[t, x], {x, 2}] // Simplify
Out[•]= True
```

One can visualize this function over a range of times. Side note: See YouTube video entitled ‘Creating Movies and Animations in Mathematica’ for more information (https://youtu.be/S03e6dwM100?t=204)

In[ ]:=
```
(*Setup boundaries for the visualization*)
xMin = -5;
xMax  = -xMin;
tMin = 1 / 10;
tMax = 10;

pMax = p[tMin, 0];

(*Visualize at various times*)
Manipulate[
 Plot[p[t, x], {x, xMin, xMax},
  AxesLabel → {"x", "p(t,x)"},
  PlotRange → {{xMin, xMax}, {0, pMax}}],
 {t, tMin, tMax}]
```

Out[ ]=



Finite Difference Equations

Suppose that we are not aware of the analytical solution and instead we desire a numerical technique that would allow the distribution to be computed at future times.

One approach is to discretize the governing PDE using a technique such as the finite difference method (Wikipedia link). The concept here is to replace the derivatives in PDE with algebraic approximations. We then solve the resulting algebraic equation to obtain an approximate solution to the PDE.

We first adopt the notation of

$$p(t_i, x_j) = p_{i,j} \qquad \textbf{(Eq.3)}$$

where i = temporal index
j = spatial index

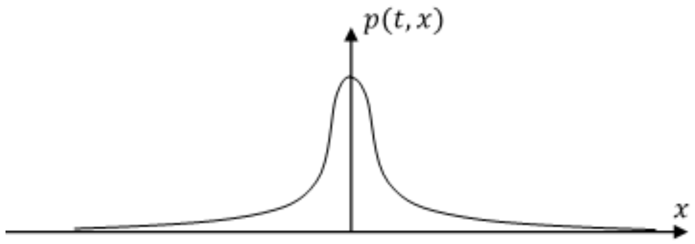
Using this notation, we can more easily write expressions such as

$$p(t_i + \Delta t, x_j) = p_{i+1,j}$$

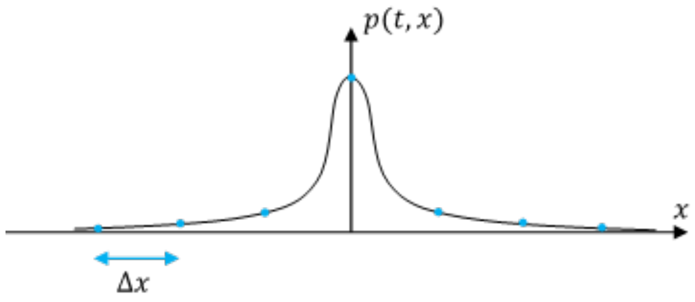
or

$$p(t_i, x_j - \Delta x) = p_{i,j-1}$$

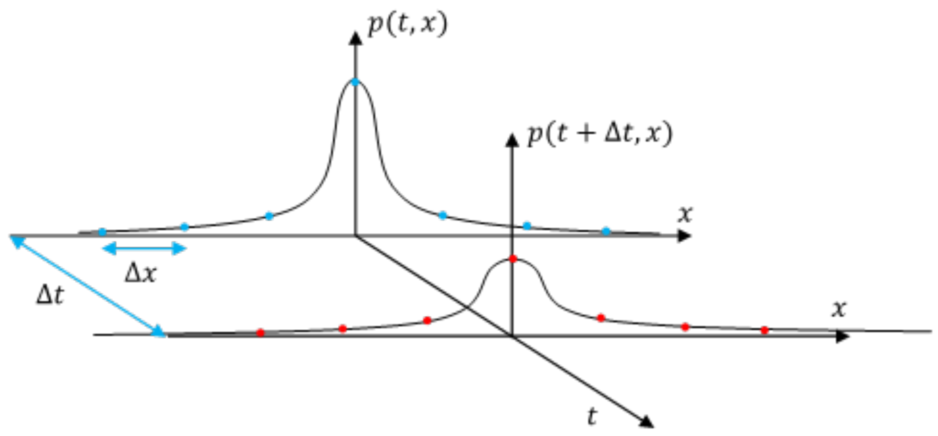
Assuming that the distribution is known at time t, one can visualize the continuous distribution as shown below



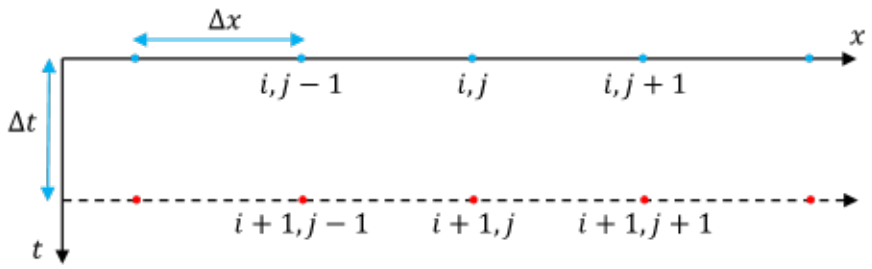
One can discretize this distribution at the time t by sampling the distribution at a finite number of spatial points as shown below as the blue dots.



The goal is to numerically calculate the distribution at a time $t + \Delta t$. These future points are shown below as red dots and represent the numerical approximation of the function in the future.



In many cases, it may be more useful to look at the points in the x, t plane. This representation of the mesh in both time and space is sometimes referred to as a **stencil** (link). For the above scenario, we can draw a stencil as shown below.

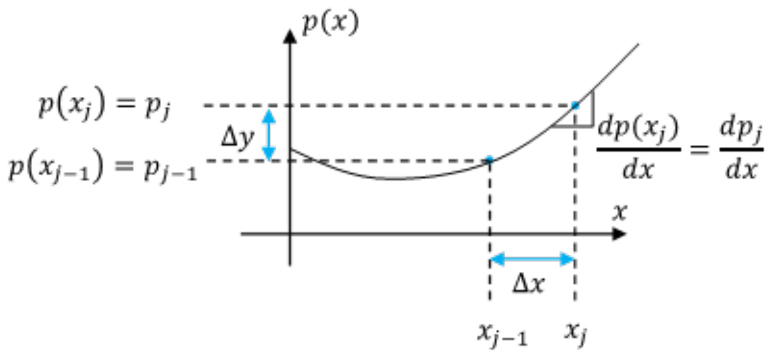


Using the above notation, the governing PDE can now be written as

$$\frac{\partial p_{i,j}}{\partial t} = \frac{1}{2}\frac{\partial^2 p_{i,j}}{\partial x^2} \qquad \textbf{\textit{(Eq.4)}}$$

Just like we did when numerically solving ODEs, the central idea is to now replace the various derivative terms with numerical, finite-difference approximations.

For example, consider a simple derivative with respect to a single independent variable



So we see that

$$\frac{d\,p_j}{d\,x} \approx \frac{\text{rise}}{\text{run}}$$

$$= \frac{\Delta y}{\Delta x}$$

$$\frac{d p_j}{d x} \approx \frac{p_j - p_{j-1}}{\Delta x}$$

We can extend this idea to a function of multiple independent variables. In the case we are interested in, we have x and t as independent variables. Let us focus on computing a derivative of the function with respect to the spatial variable x. In other words, we would like to numerically compute $\partial p_{i,j} / \partial x$. There are three common schemes to numerically approximate a derivative (link).

1. Forward Difference
2. Backward Difference
3. Central Difference

In each case, we seek to approximate the derivative with an algebraic expressing involving the function values at nearby points. The general idea is to approximate the instantaneous derivative by a simple rise/run formula. In essence, we ask how did the function change between points and use this information to approximate the derivative.

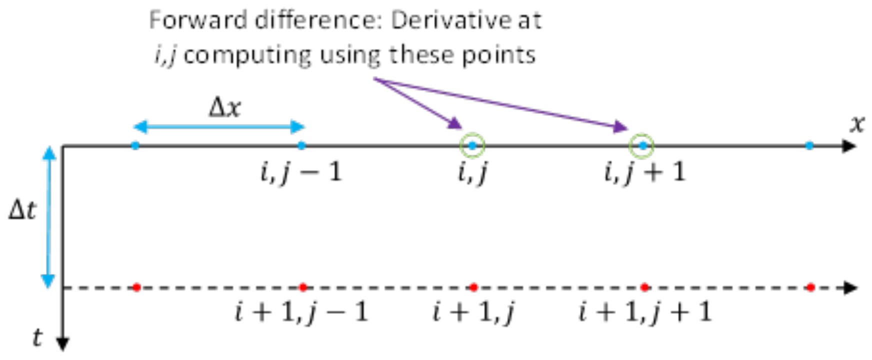
Forward Difference

With the forward difference scheme, the derivative at a given point is computed using the function value at the point in question as well as at the point forward from the point in question.

$$\frac{\partial p_{i,j}}{\partial x} \approx \frac{p_{i,j+1} - p_{i,j}}{\Delta x} \qquad \textbf{(Eq.5)}$$

where Δx = spatial step

The stencil associated with this is shown below

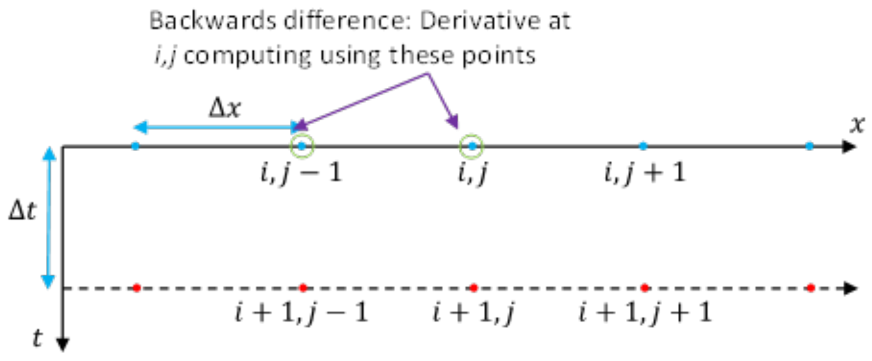

Forward difference: Derivative at
i,j computing using these points
Δx
x
i, j − 1
i, j
i, j + 1
Δt
i + 1, j − 1
i + 1, j
i + 1, j + 1
t

Backward Difference

With the backward difference scheme, the derivative at a given point is computed using the function value at the point in question as well as at the point backwards from the point in question.

$$\frac{\partial p_{i,j}}{\partial x} \approx \frac{p_{i,j} - p_{i,j-1}}{\Delta x} \qquad \textbf{(Eq.6)}$$

The stencil associated with this is shown below

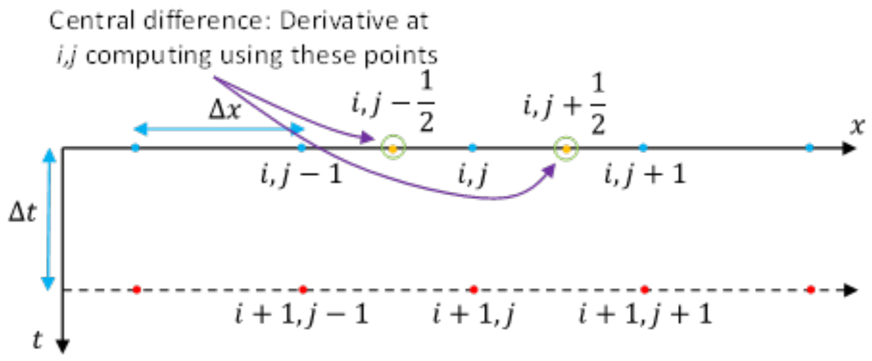


Central Difference

With the central difference scheme, the derivative at a given point is computed by first introducing intermediate, (halfway) points (shown in orange in the stencil below). The derivative at a given point is computed using the function values at these intermediate points.

$$\frac{\partial p_{i,j}}{\partial x} \approx \frac{p_{i,j+1/2} - p_{i,j-1/2}}{\Delta x} \qquad \textbf{(Eq.6)}$$

The stencil associated with this is shown below



We can extend this idea for higher order derivatives. For example, to compute $\partial^2 p_{i,j} / \partial x^2$, we first apply the central difference formula (Eq.6) for the derivative p at $x_{j+1/2}$ to obtain

$$\frac{\partial p_{i,j+1/2}}{\partial x} = \frac{p_{i,j+1} - p_{i,j}}{\Delta x} \qquad \textbf{(Eq.7)}$$

In a similar fashion, for the derivative p at $x_{j-1/2}$

$$\frac{\partial p_{i,j-1/2}}{\partial x} = \frac{p_{i,j} - p_{i,j-1}}{\Delta x} \qquad \textbf{(Eq.8)}$$

We now compute the second derivative by again applying the central difference formula to compute the second derivative of p at x_j as how much the first derivative changes between the points $j-1/2$ to $j+1/2$

$$\frac{\partial^2 p_{i,j}}{\partial x^2} \approx \frac{\frac{\partial p_{i,j+1/2}}{\partial x} - \frac{\partial p_{i,j-1/2}}{\partial x}}{\Delta x}$$

$$\approx \frac{\left(\frac{p_{i,j+1}-p_{i,j}}{\Delta x}\right) - \left(\frac{p_{i,j}-p_{i,j-1}}{\Delta x}\right)}{\Delta x}$$

$$\approx \frac{p_{i,j+1}-p_{i,j}-p_{i,j}+p_{i,j-1}}{\Delta x^2}$$

$$\frac{\partial^2 p_{i,j}}{\partial x^2} \approx \frac{p_{i,j+1}-2\,p_{i,j}+p_{i,j-1}}{\Delta x^2} \qquad \textbf{\textit{(Eq.9)}}$$

We now turn our attention to the temporal derivative, $\partial p_{i,j} / \partial t$. Applying a forward difference formula yields

$$\frac{\partial p_{i,j}}{\partial t} \approx \frac{p_{i+1,j}-p_{i,j}}{\Delta t} \qquad \textbf{\textit{(Eq.10)}}$$

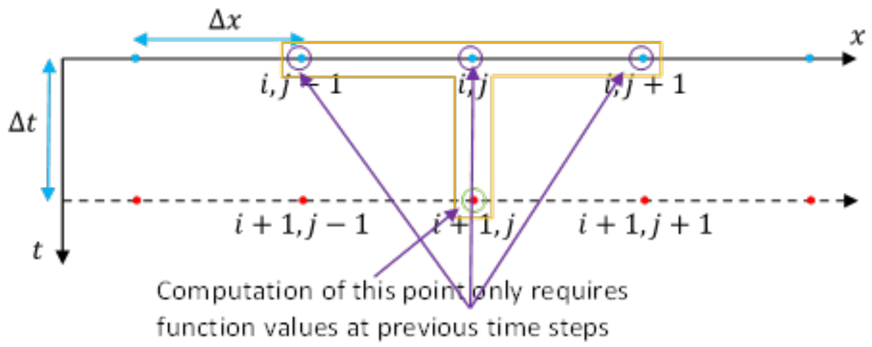
Substituting Eq. 9 and Eq.10 into the original PDE (Eq.4) yields

$$\frac{\partial p_{i,j}}{\partial t} = \frac{1}{2}\frac{\partial^2 p_{i,j}}{\partial x^2} \qquad \text{note: } \frac{\partial p_{i,j}}{\partial t} \approx \frac{p_{i+1,j}-p_{i,j}}{\Delta t} \text{ and } \frac{\partial^2 p_{i,j}}{\partial x^2} \approx \frac{p_{i,j+1}-2\,p_{i,j}+p_{i,j-1}}{\Delta x^2}$$

$$\frac{p_{i+1,j}-p_{i,j}}{\Delta t} = \frac{1}{2}\frac{p_{i,j+1}-2\,p_{i,j}+p_{i,j-1}}{\Delta x^2}$$

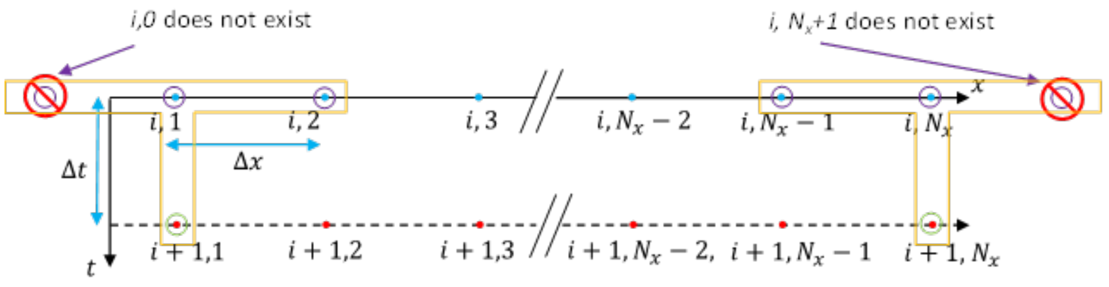
$$p_{i+1,j} = \frac{1}{2}\frac{\Delta t}{\Delta x^2}\left(p_{i,j+1} - 2\,p_{i,j} + p_{i,j-1}\right) + p_{i,j} \qquad \textbf{\textit{(Eq.11)}}$$

Eq.11 is a convenient formulation as it allows the distribution at a future time step (green circled point) to be computed using only points from the previous time step (purple circled points)


Δx
x
Δt
t
$i, j-1$
i, j
$i, j+1$
$i+1, j-1$
$i+1, j$
$i+1, j+1$
Computation of this point only requires function values at previous time steps

Boundary Conditions

Consider that the spatial domain of the function is discretized into N_x points. Therefore, $j = 1, 2, \ldots, N_x$. A more complete stencil is shown below

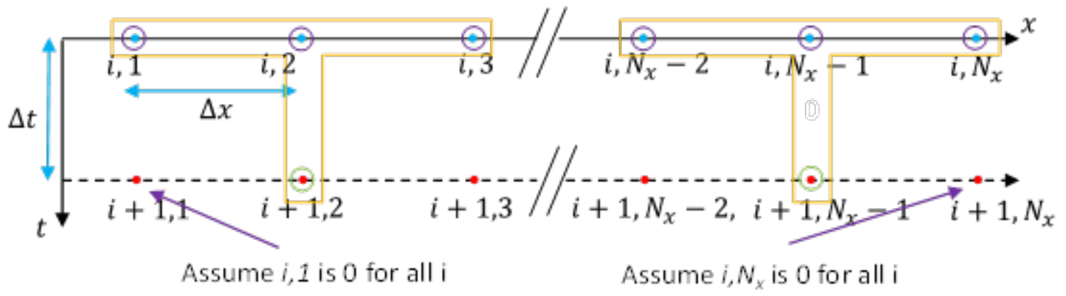


Notice at Eq.11 cannot be directly applied to compute $p_{i+1,1}$ because $p_{i,0}$ does not exist. A similar problem exists when attempting to compute p_{i+1,N_x} because p_{i,N_x+1} does not exist.

These are boundary condition problems and there are several approaches to handling these.

Zero Boundary Conditions

A simple approach is to assume that the function is zero at the left and right boundary for all j.



After making this assumption, only the values of $j = 2, 3, \ldots, N_x - 1$ are computed using Eq.11.

$$p_{i+1,j} = \begin{cases} 0 & j = 1 \text{ or } j = N_x \\ \frac{1}{2}\frac{\Delta t}{\Delta x^2}\left(p_{i,j+1} - 2\,p_{i,j} + p_{i,j-1}\right) + p_{i,j} & \text{otherwise} \end{cases} \qquad j = 1, 2, \ldots, N_x \qquad \textbf{(Eq.12)}$$

Alternatively, we can change the labeling of the indexing to write this as

$$p_{i,j} = \begin{cases} 0 & j = 1 \text{ or } j = N_x \\ \frac{1}{2}\frac{\Delta t}{\Delta x^2}\left(p_{i-1,j+1} - 2\,p_{i-1,j} + p_{i-1,j-1}\right) + p_{i-1,j} & \text{otherwise} \end{cases} \qquad j = 1, 2, \ldots, N_x$$

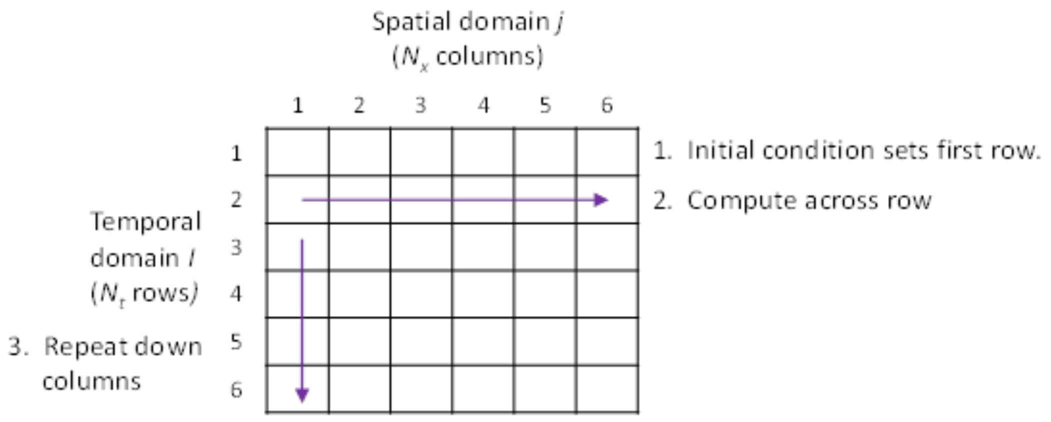
(Eq.13)

Periodic Boundary Conditions

Another approach is to apply periodic boundary conditions. In this scenario, the missing point on the left boundary is taken as the value on the right boundary $(p_{i,0} = p_{i,N})$ and vice versa $(p_{i,N+1} = p_{i,1})$. This method is outside the scope of this class.

Numerical Implementation

At this point, we can develop an algorithm to solve Eq.13 over a discretized temporal/spatial domain. During numerical implementation, our choice of indexing order allows development of an algorithm that directly mirror the diagrams and concepts outlined above. In other words, if we discretize the temporal domain into N_t sections, we can setup a matrix with N_t rows and N_x columns.



This results in a double for loop that applies Eq.13 at each point in the matrix. Pseudo-code for this procedure is shown below. You will implement this code in a homework assignment.

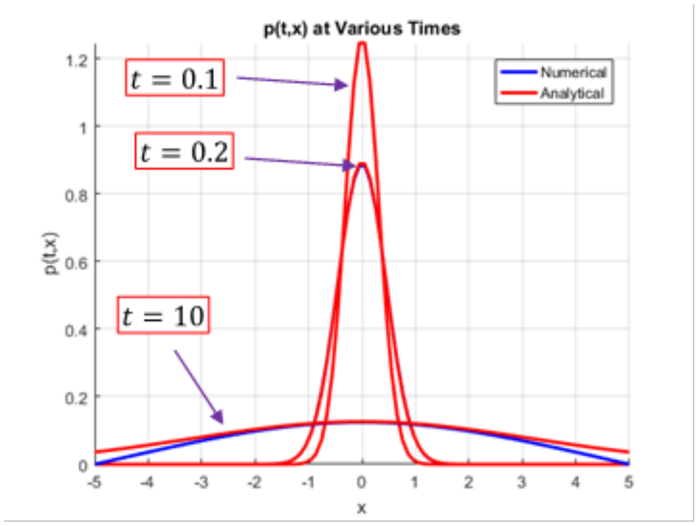
```
Nt = 1000;       %number of temporal steps
Nx = 100;        %number of spatial steps

tMin = 0.1;
tMax = 10;

xMin = -5;
xMax = 5;

t = linspace(tMin, tMax, Nt);
x = linspace(xMin, xMax, Nx);

%Compute secondary parameters
DeltaT = t(2) - t(1);
DeltaX = x(2) - x(1);

%% Numerically Solve PDE
%Initialize a container for all the p values.
p = zeros(Nt, Nx);

%Set initial condition
p(1,:) = AnalyticalSolution(t(1), x);

%iterate through time values
for i = 2:Nt

    %iterate through spatial values
    for j = 1:Nx
        if(j==1 || j==Nx)
            %Hold boundary condition of zero
            p(i,j) = 0;
        else
            %Compute function value
            p_im1_jp1   = p(i-1,j+1);   %previous time point and forward spatial point
            p_im1_j     = p(i-1,j);     %previous time point and current spatial point
            p_im1_jm1   = p(i-1,j-1);   %previous time point and backwards spatial point

            p(i,j) = 0.5*(DeltaT/(DeltaX^2))*(p_im1_jp1 - 2*p_im1_j + p_im1_jm1) + p_im1_j;
        end
    end
end
```

We will see that the choices of temporal and spatial spacing affect the stability of the numerical algorithm.

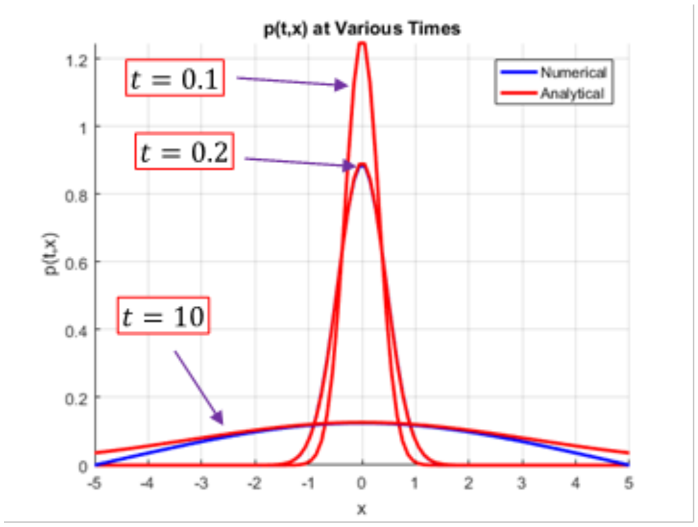
Stable Case

For example, if we choose $t \in [0.1, 10]$ with $N_t = 1000$ and $x \in [-5, 5]$ with $N_x = 100$ we obtain the results as shown below

In[∘]:=



Out[∘]=

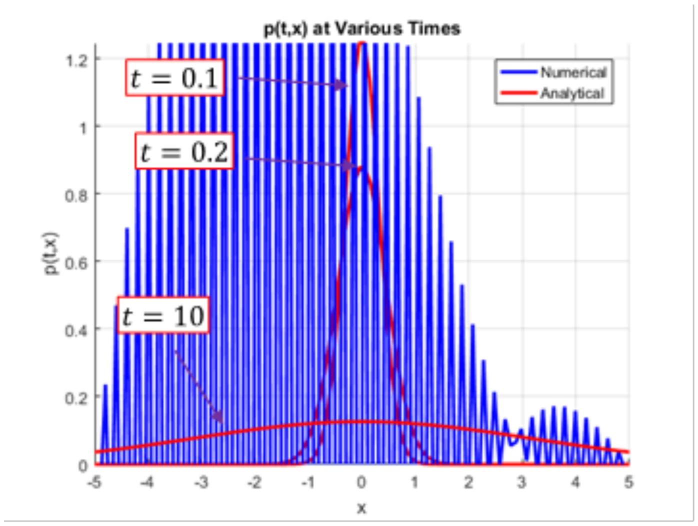


As can be seen, the numerical solution approximates the analytical solution fairly well. Notice that at higher values of t, the numerical solution maintains the boundary condition of $p = 0$ which makes it diverge from the analytical solution.

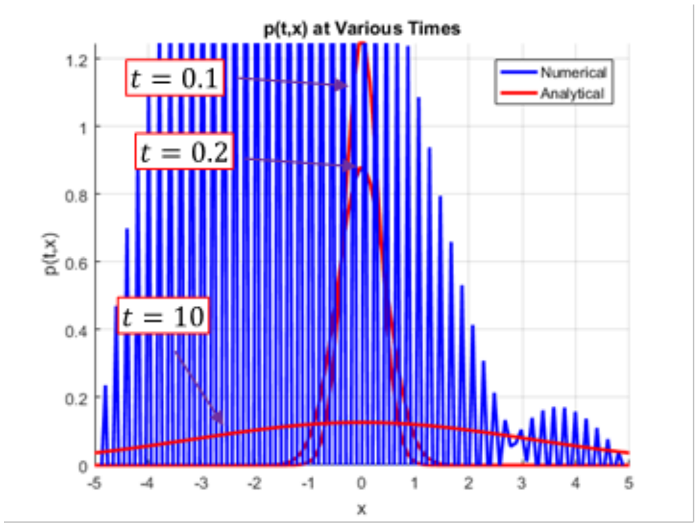
Unstable Case

We we use the same parameters as above but slightly increase the temporal spacing by reducing $N_t = 950$ we obtain the following results.

In[ ]:=



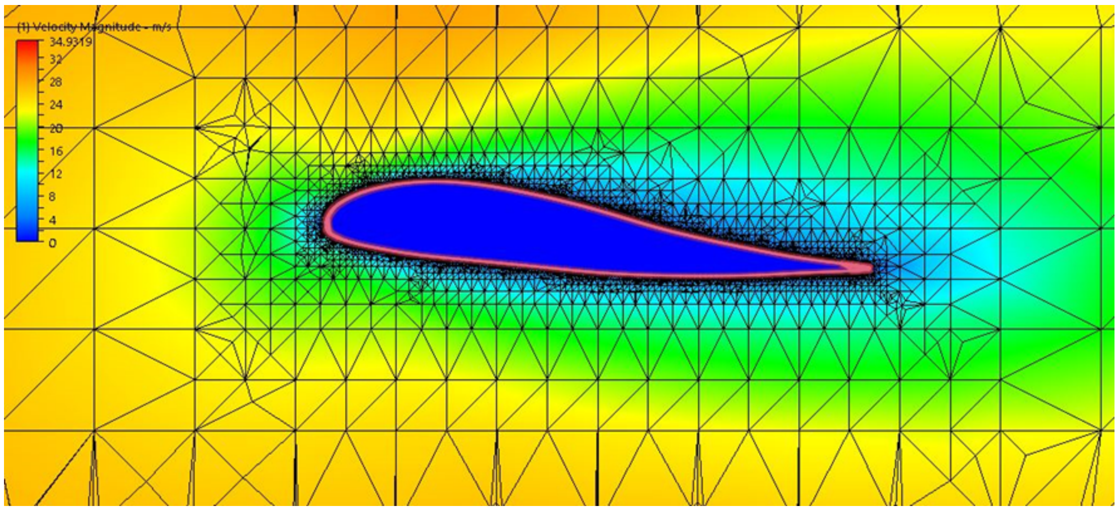
Out[ ]=



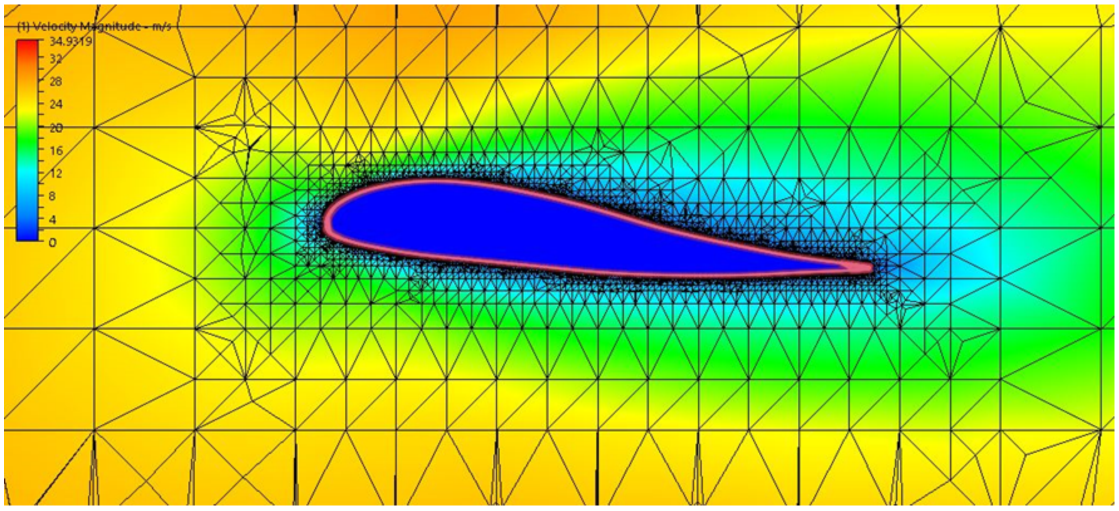
In this case, the numerical algorithm begins to break down as *t* increases. Near the end of the simulation the system begins to diverge ***<watch movie/animation of this phenomenon>***

This type of approach can be applied to finite element modeling (FEM), computational fluid dynamics (CFD), and other applications.

In[ ]:=


(1) Velocity Magnitude - m/s
34.9319
32
28
24
20
16
12
8
4
0

Out[ ]=


(1) Velocity Magnitude - m/s
34.9319
32
28
24
20
16
12
8
4
0