Leo Carrico
Dec 19 2019

Final Lab

1. One Dimensional Heat Equation with Euler Explicit

For One Dimensional Heat Equation with Euler Explicit, we start by initializing our variables.

```
m = 2500
n = 25

L = 1.0
T = 1.0

delta_x = L/n
delta_t = T/m

alpha = .60
```

This sets our bounds up to L and T for our distance and time with m and n as our number of steps to get there.

Then using those numbers we find our delta_x and delta_t, or our distance between each step.

```
us = np.zeros([m+1,n+1])

for i in range(1, len(us[0,:])):
    us[0,i] = us[0,i-1] + delta_x
```

Then we initialize our 2 by 2 two matrix representing our values of u of x and t.

With x being our points in space, we set values u at our first point, x = 0, to the values of x at those points.
With that done, we can now use our initial condition to create our values of u at our initial point.

```
def u0(x):
    return np.sin(np.pi * x)

for i in range(len(us[0, :])):
    us[0, i] = u0(us[0, i])
```

This code generates our initial condition function and sets the initial values.

Now in order to solve the equation, we have to solve for the euler explicit using finite difference.

So with this given:
$du/dt = a * d^2u/dx^2$

We use finite difference to get:
$(u_{i,n+1} - u_{i,n})/\Delta t - \alpha\, (u_{i+1,n} - 2u_{i,n} + u_{i-1,n} / (\Delta x)^2) = 0$

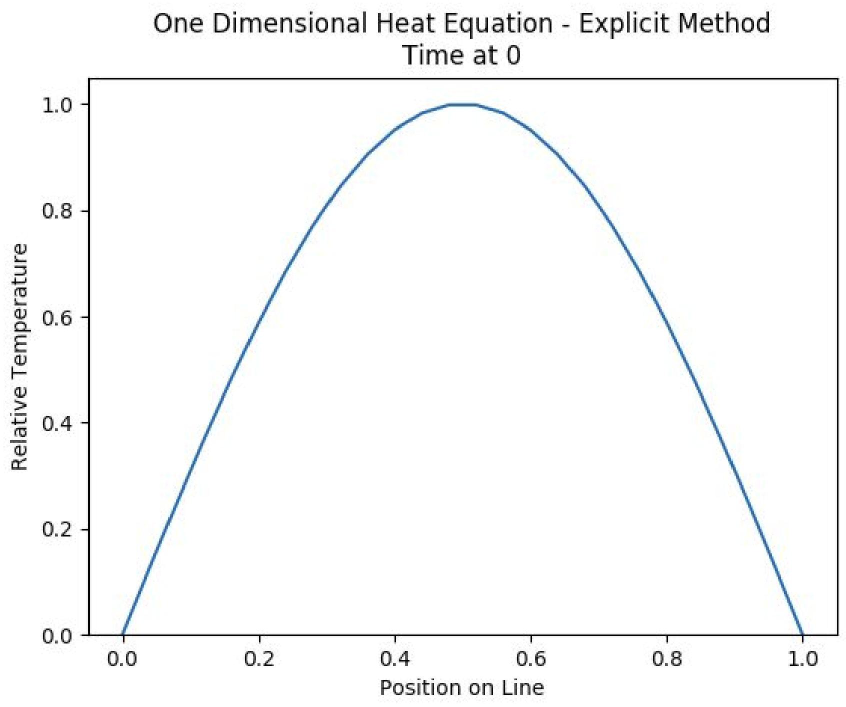
Separate the unknowns from the knowns and you get…
$u_{i,n+1} = (1 - 2 * h)u_{i,n} + h(u_{i+1,n} + u_{i-1,n})$
With h as $\alpha\Delta t / (\Delta x)^2$

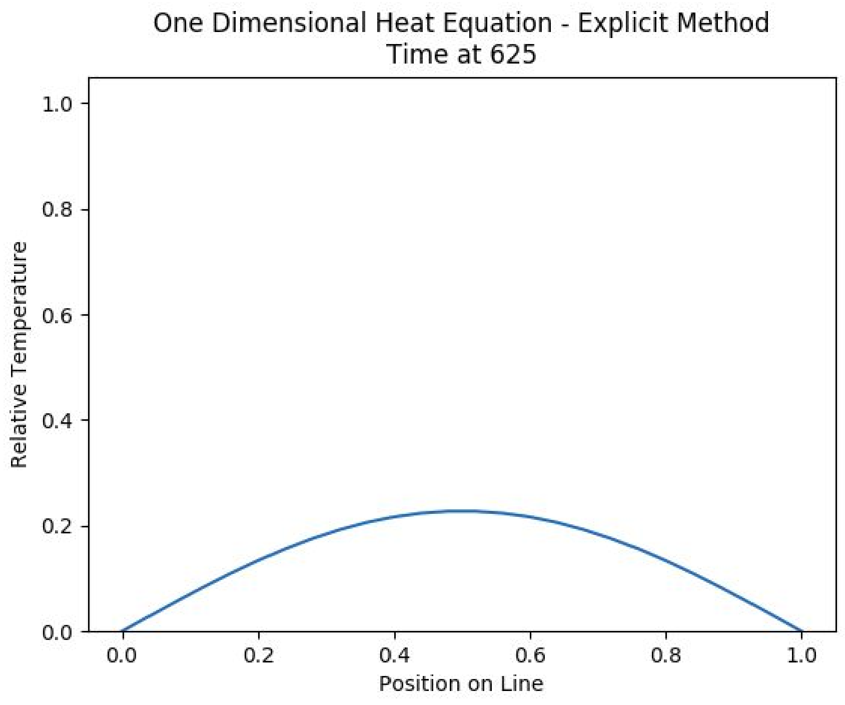
With this we can code our solver.

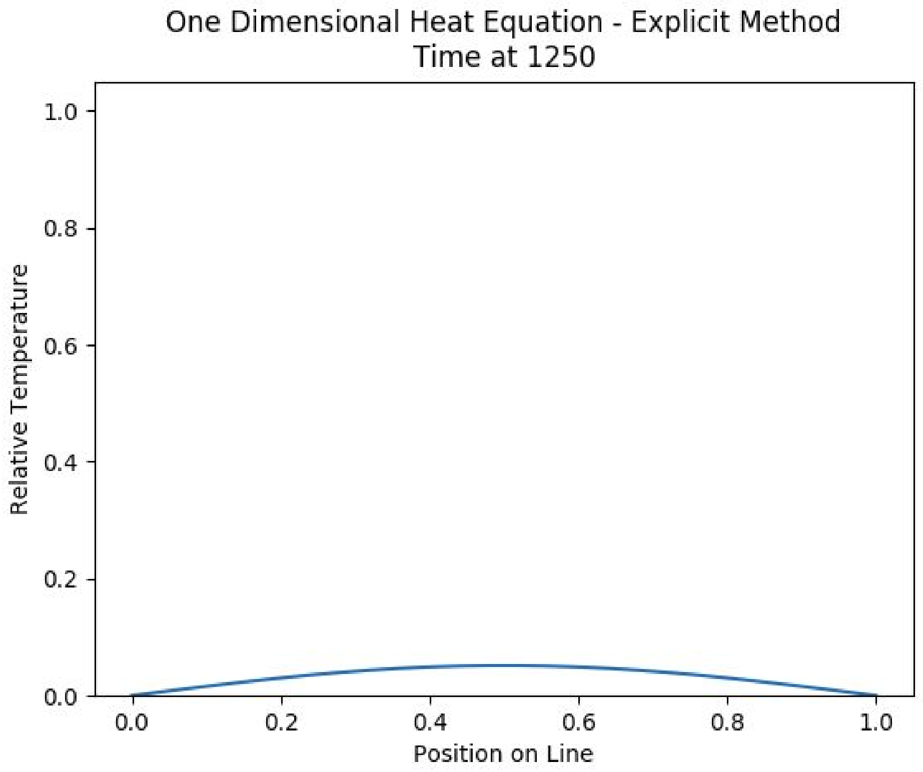
```
h = (alpha * delta_t / (delta_x * delta_x))

for k in range(1,m+1,1):
    for j in range (1,n,1):
        us[k][j] = h * us[k-1][j-1] + (1 - 2 * h) * us[k-1][j] + h *
us[k-1][j+1]
```

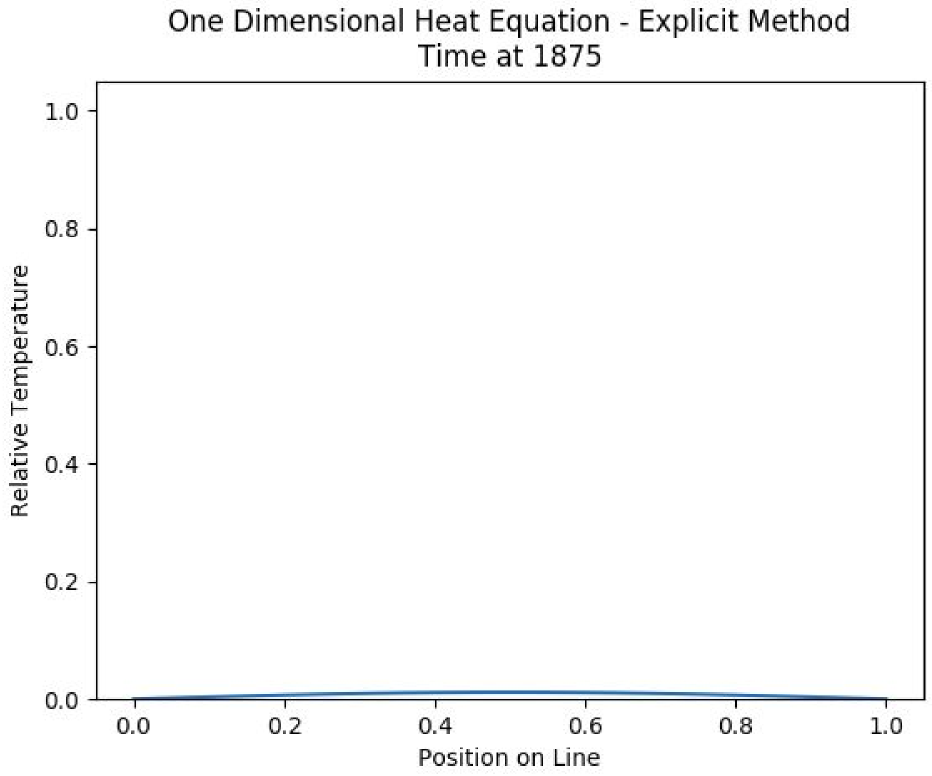
We initialize h to be our $\alpha\Delta t / (\Delta x)^2$
And then solve for every next iteration of u until the end of our bounds.

Here are the solutions for the problem given:






One Dimensional Heat Equation - Explicit Method
Time at 1250
Relative Temperature
Position on Line
0.0
0.2
0.4
0.6
0.8
1.0


One Dimensional Heat Equation - Explicit Method
Time at 1875
Relative Temperature
Position on Line
0.0
0.2
0.4
0.6
0.8
1.0

2. One Dimensional Heat Equation with Euler Implicit

In the case of the One Dimensional Heat Equation with Euler Implicit, all the initialization steps are the same as with the Euler Explicit, the difference being the Solver.

Once we have our formula
ui,n+1 = (1 − 2 * h)ui,n + h(ui+1,n + ui−1,n)
With h as αΔt / (Δx)^2

We can see that this can be expressed as a matrix:

$$\begin{bmatrix} u_{1,n+1} \\ u_{2,n+1} \\ \vdots \\ u_{M-1,n+1} \end{bmatrix} = \begin{bmatrix} a & b & 0 & \dots & 0 \\ b & a & b & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & b & a \end{bmatrix} \begin{bmatrix} u_{1,n} \\ u_{2,n} \\ \vdots \\ u_{M-1,n} \end{bmatrix}$$

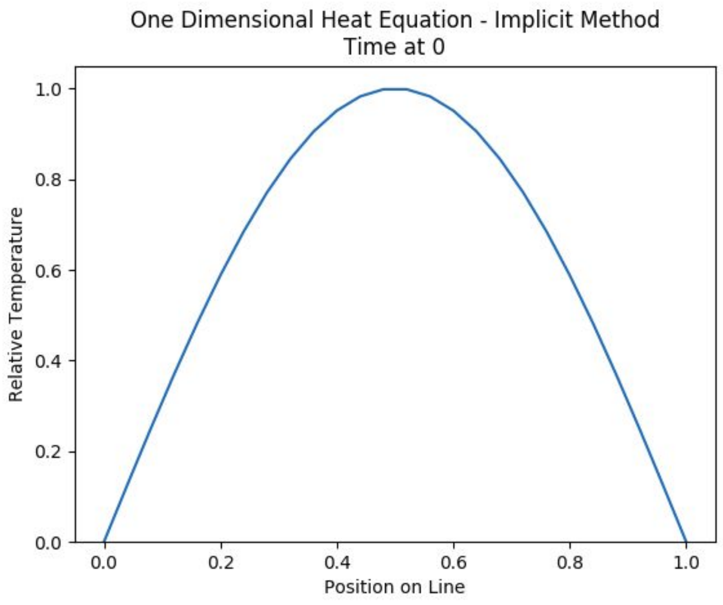
Where a = 2 * h + 1 and b = -h

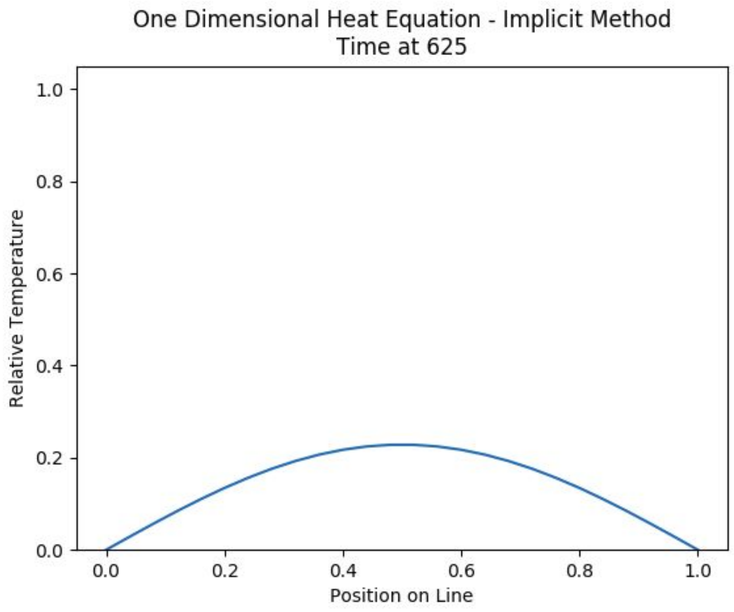
With this knowledge, we can implicitly solve our values of u.

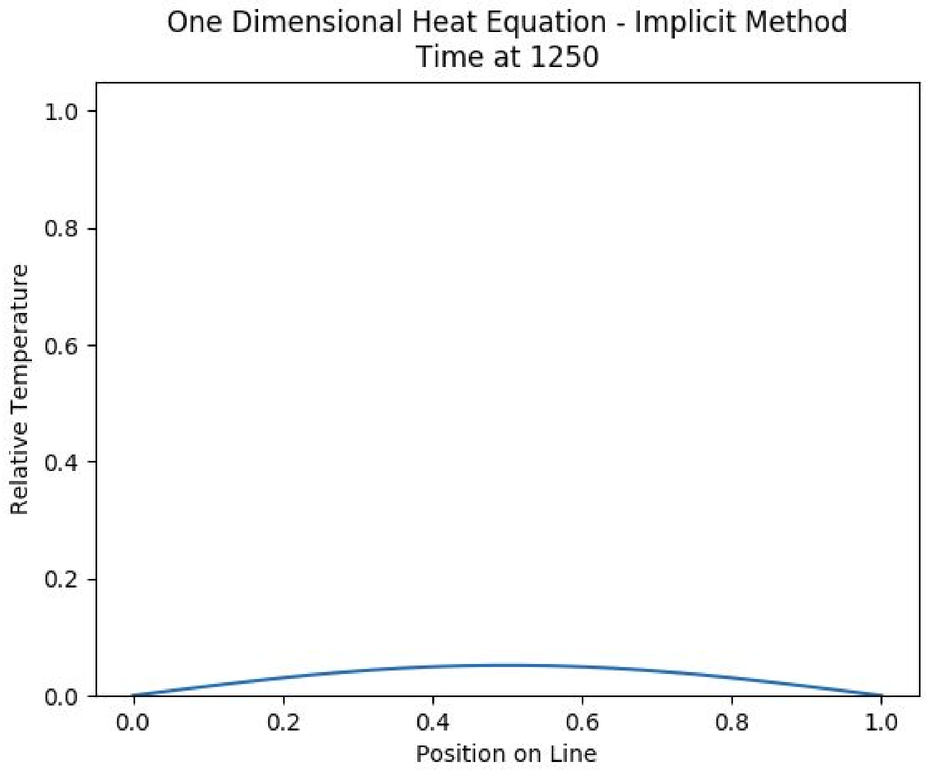
```
for k in range(0, m, 1):
    mat = np.zeros([n+1,n+1])
    for i in range(1,n,1):
        mat[i,i] = 2 * h + 1.0
        mat[i,i-1] = -h
        mat[i,i+1] = -h
    mat[0,0] = 1
    mat[n,n] = 1

    temp = us[k,:]

    x = linalg.solve(mat, temp)
    us[k + 1, :] = x
```

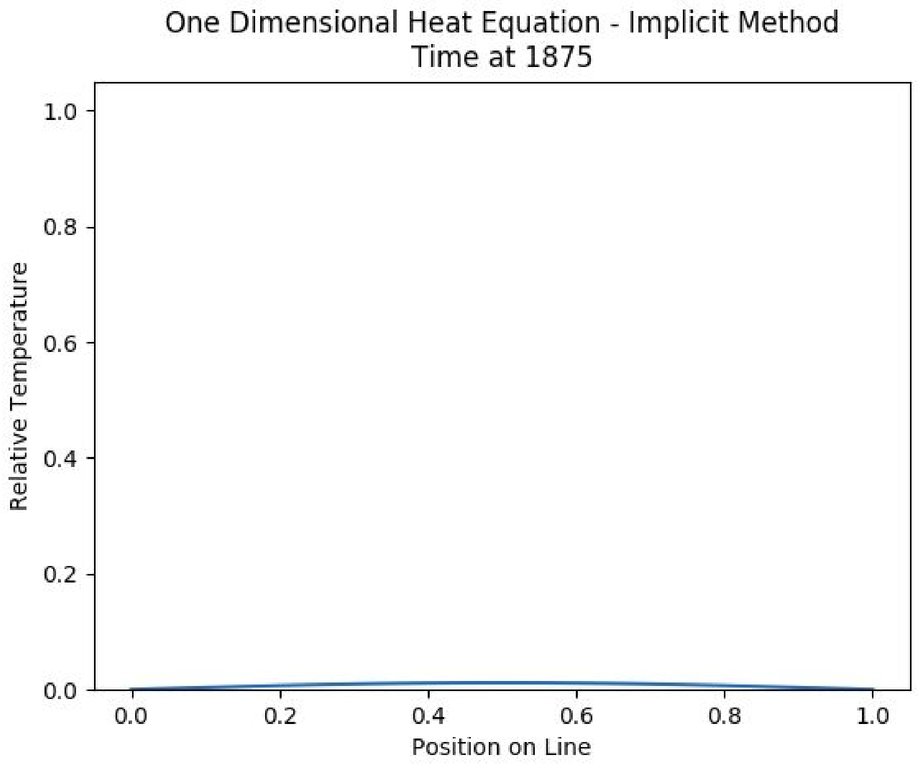
Our code here initializes our matrices accordingly and then runs the solver to find our next values.

Here are our solutions:






One Dimensional Heat Equation - Implicit Method
Time at 1250
Relative Temperature
Position on Line
0.0
0.2
0.4
0.6
0.8
1.0


One Dimensional Heat Equation - Implicit Method
Time at 1875
Relative Temperature
Position on Line
0.0
0.2
0.4
0.6
0.8
1.0

3. One Dimensional Heat Equation with Runge Kutta 4

In the case of the One Dimensional Heat Equation with Runge Kutta 4, all the initialization steps are the same as with the Euler Explicit and Euler Implicit, the difference being the solver.

Once we have our formula
ui,n+1 = (1 − 2 * h)ui,n + h(ui+1,n + ui−1,n)
With h as α∆t / (∆x)^2

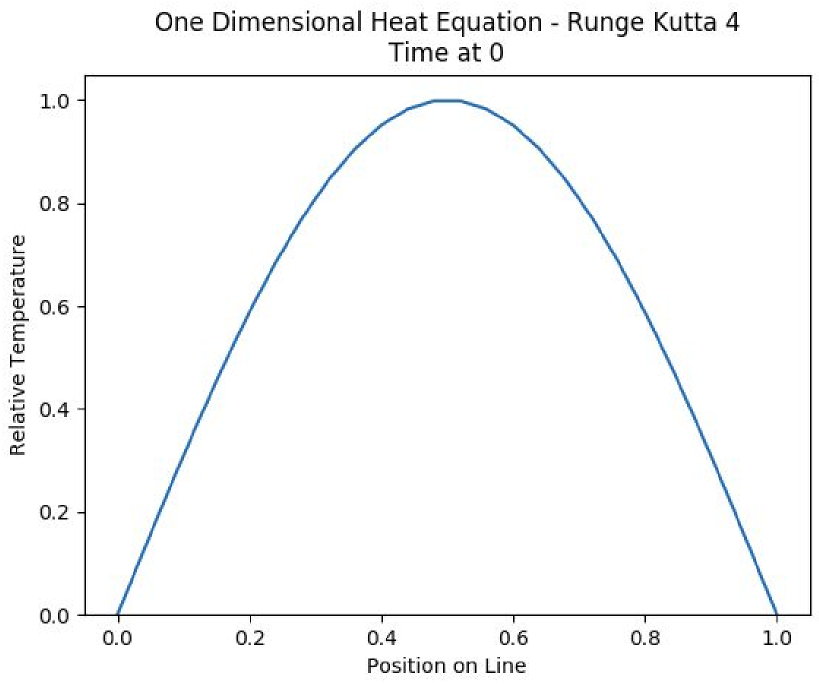
Now we have our formula for Runge Kutta being:
yn+1 = yn + (⅙)(k1 + 2k2 + 2k3 + k4)
Where
k1 = h * f(xn, yn)
k2 = h * f(xn + h/2, yn + k1/2)
k3 = h * f(xn + h/2, yn + k2/2)
K4 = h * f(xn + h, yn + k3)

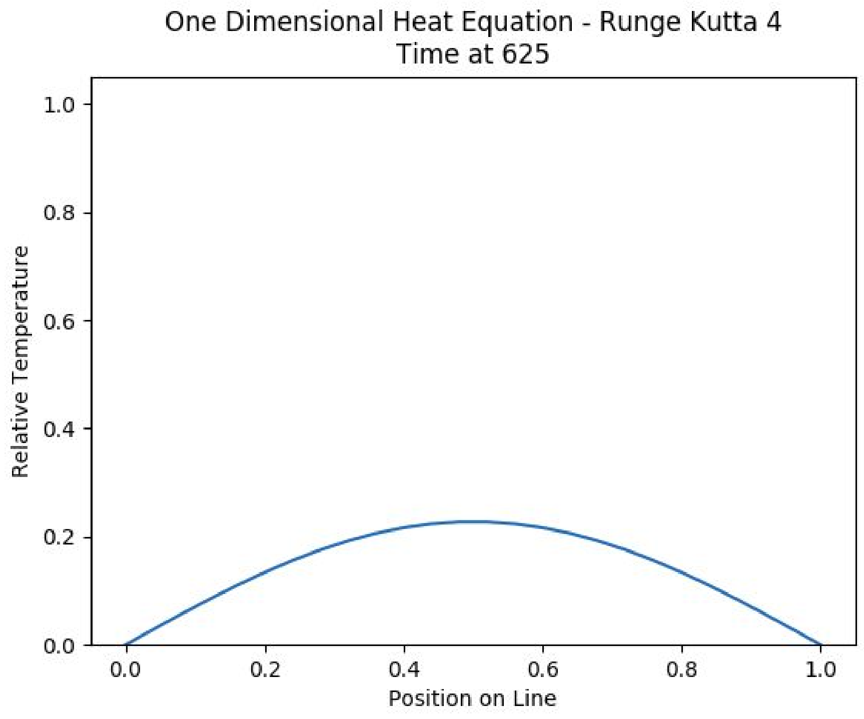
Applying our original function, we get this with our code:

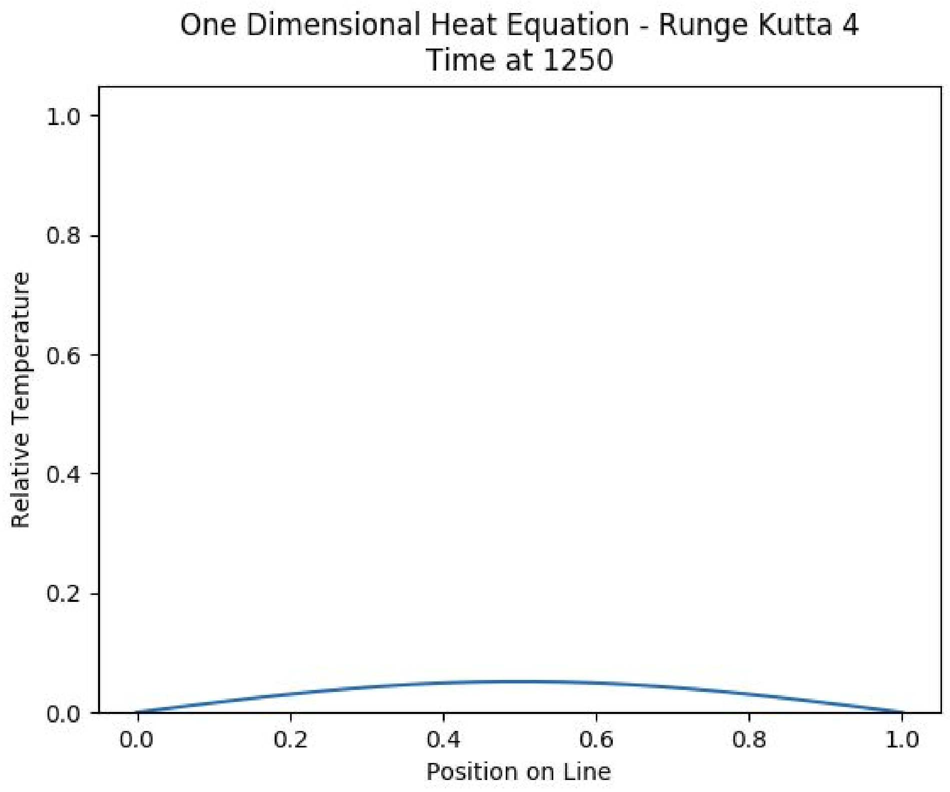
```
for k in range(1,m+1,1):
    for j in range (1,n,1):
        # us[k][j] = h * us[k-1][j-1] + (1 - 2 * h) * us[k-1][j] + h
* us[k-1][j+1]
        k1 = h * (us[k-1][j+1] - 2*us[k-1][j] + us[k-1][j-1])
        k2 = h * ((us[k-1][j+1]+
0.5*k1)-2*(us[k-1][j]+0.5*k1)+(us[k-1][j-1]+0.5*k1))
        k3 = h *
((us[k-1][j+1]+0.5*k2)-2*(us[k-1][j]+0.5*k2)+(us[k-1][j-1]+0.5*k2))
        k4 = h *
((us[k-1][j+1]+k3)-2*(us[k-1][j]+k3)+(us[k-1][j-1]+k3))
        us[k][j] = us[k-1][j] + (1/6) * (k1 + 2*k2 + 2*k3 + k4)
```

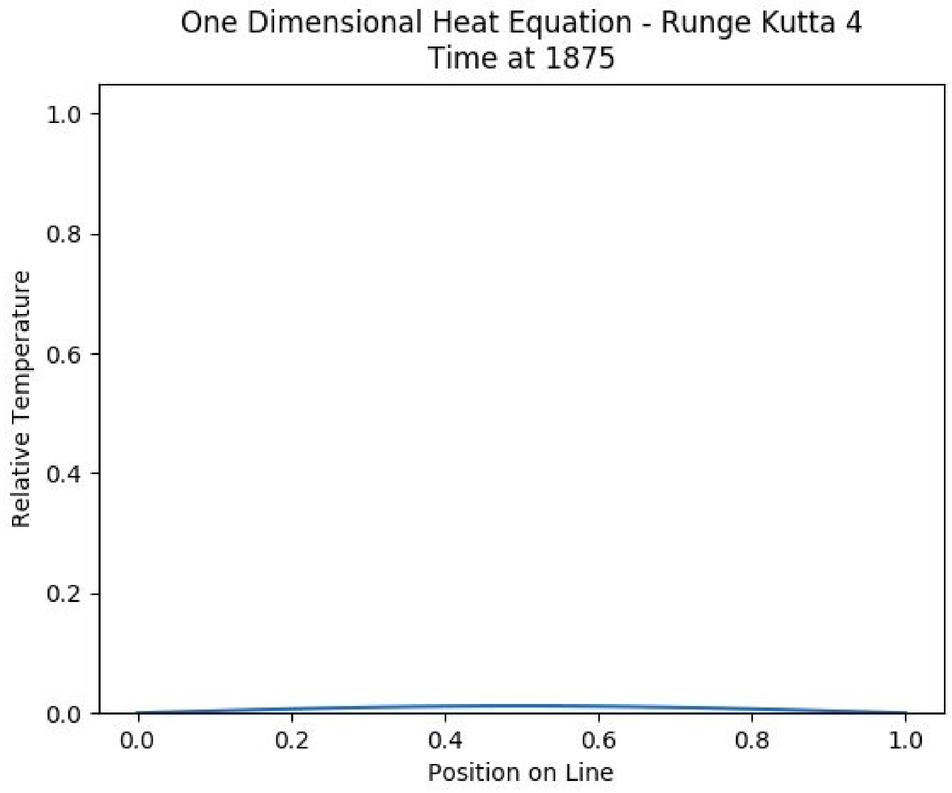
So now with function f replaced by our actual function, we get our functional solver.

Solution:


One Dimensional Heat Equation - Runge Kutta 4
Time at 0
Relative Temperature
Position on Line
0.0
0.2
0.4
0.6
0.8
1.0


One Dimensional Heat Equation - Runge Kutta 4
Time at 625
Relative Temperature
Position on Line
0.0
0.2
0.4
0.6
0.8
1.0


One Dimensional Heat Equation - Runge Kutta 4
Time at 1250
Relative Temperature
Position on Line
0.0
0.2
0.4
0.6
0.8
1.0


One Dimensional Heat Equation - Runge Kutta 4
Time at 1875
Relative Temperature
Position on Line
0.0
0.2
0.4
0.6
0.8
1.0

4. One Dimensional Wave Equation with Euler Explicit

For One Dimensional Wave Equation with Euler Explicit, we start by initializing our variables.

```
m = 1000
n = 100

L = np.pi
T = 100

delta_x = L/n
delta_t = T/m

alpha = 0.01
```

This sets our bounds up to L and T for our distance and time with m and n as our number of steps to get there.

Then using those numbers we find our delta_x and delta_t, or our distance between each step.

```
us = np.zeros([m+1,n+1])

for i in range(1, len(us[0,:])):
    us[0,i] = us[0,i-1] + delta_x
```

Then we initialize our 2 by 2 two matrix representing our values of u of x and t.

With x being our points in space, we set values u at our first point, x = 0, to the values of x at those points.

With that done, we can now use our initial condition to create our values of u at our initial point. In order to begin doing so, since our equation says:

$d^2u/dt^2 = a * d^2u/dx^2 + f(x,t)$

Using finite difference we get:
$(u_{i,n+1} - 2u_{i,n} + u_{i,n-1}) / \Delta t^2 = a * ((u_{i+1,n} - 2u_{i,n} + u_{i-1,n}) / \Delta x^2)$

And in order to get our next iteration, we have to have both the previous iteration, and the one before it. Thus we must start by initializing our first two points in our initial time.

Then our initial points can be initialized like so:

```python
def u0(x):
    return np.sin(x)

def v0(x):
    return -np.sin(x)

for i in range(1, len(us[0,:])):
    us[0,i] = us[0,i-1] + delta_x

for i in range(len(us[0, :]) - 1):
    us[0, i] = u0(us[0, i])

for i in range(len(us[1, :]) - 1):
    us[1, i] = us[0,i] + v0(xs[i]) * delta_t
```

This code generates our initial condition function and sets the initial values for the first two points. Now we can begin to generate the solver.

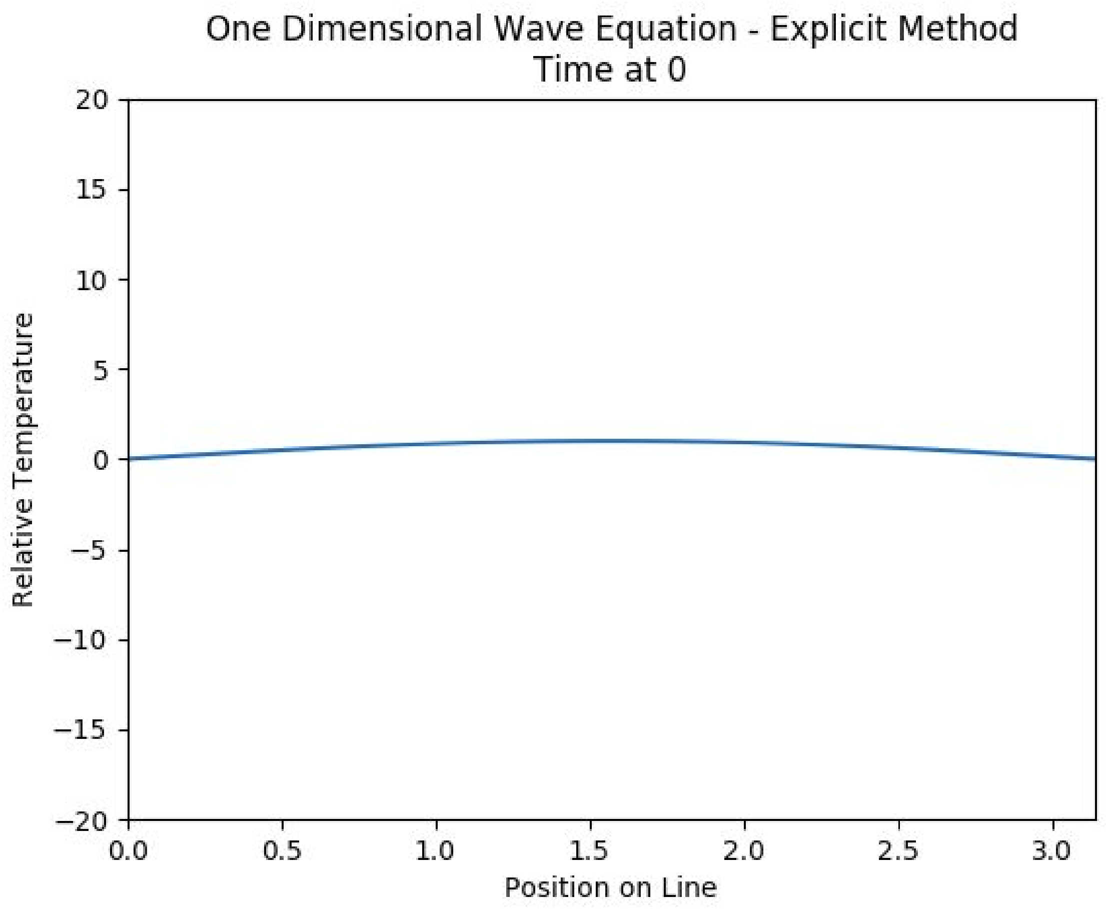
In order to do so, we move all unknowns to one side where we now have:
$u_{i,n+1} = -u_{i,n-1} + 2u_{i,n} + a*\Delta t^2/\Delta x^2 (u_{i+1,n} -2u_{i,n} +u_{i-1,n}) + \Delta t^2*f(x,t)$

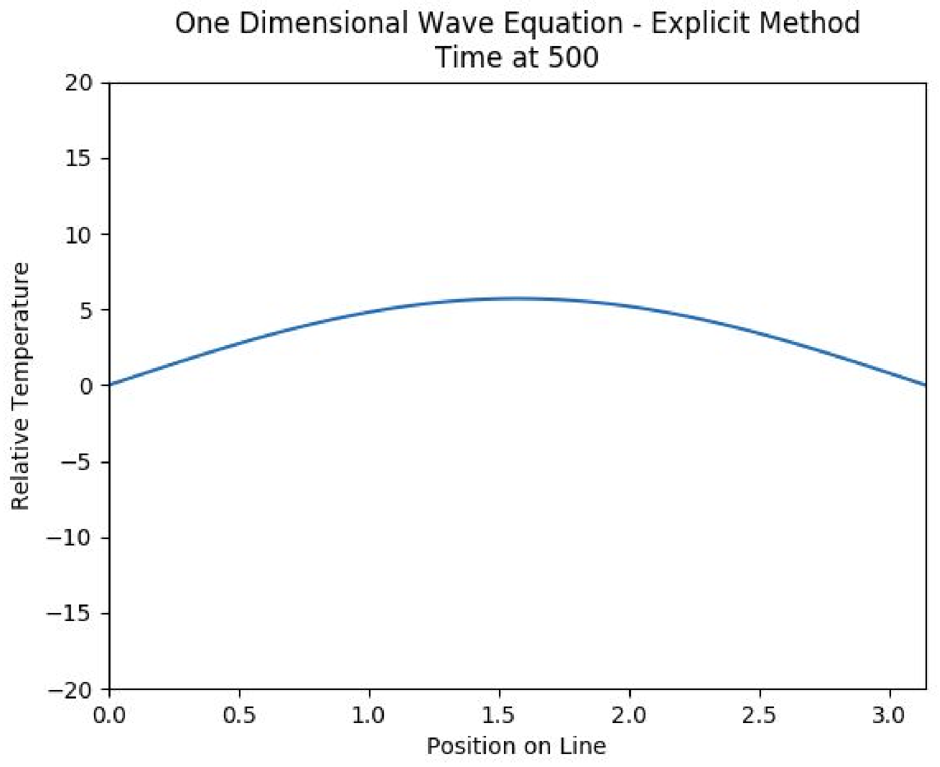
With this we generate our code as:

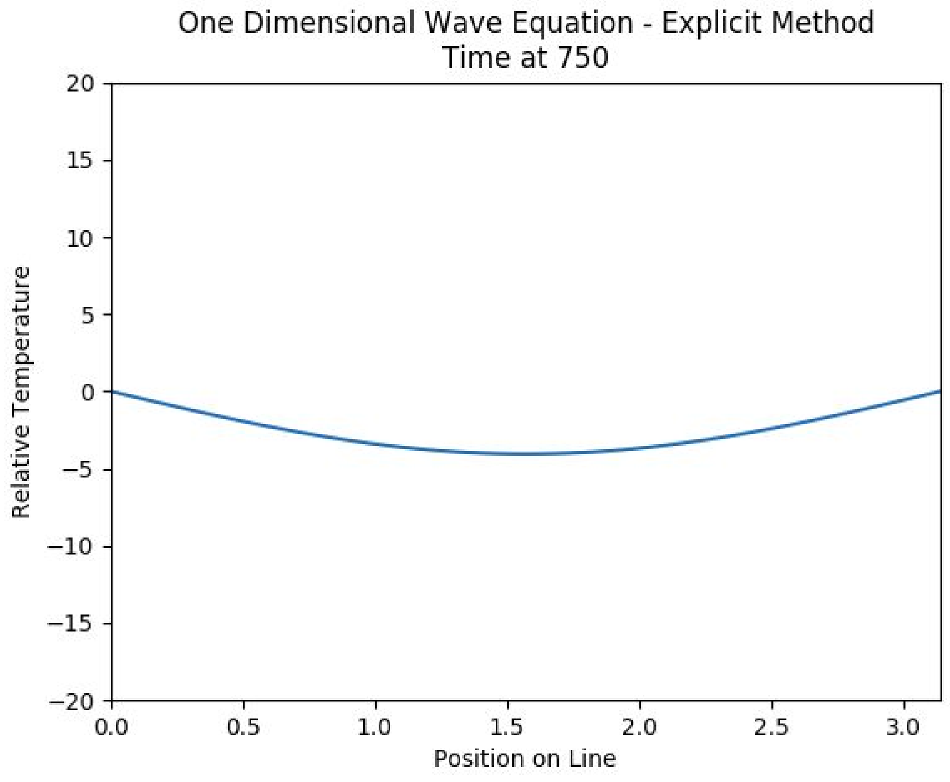
```
for k in range(2,m+1):
    for j in range (1,n):
        us[k,j] = -us[k-2,j] + 2 * us[k-1,j] + h * (us[k-1,j+1] - 2 *
(us[k-1,j]) + us[k-1,j-1]) + delta_t**2 * f(xs[j], k * delta_t)
    us[k,0], us[k,-1] = 0, 0
```

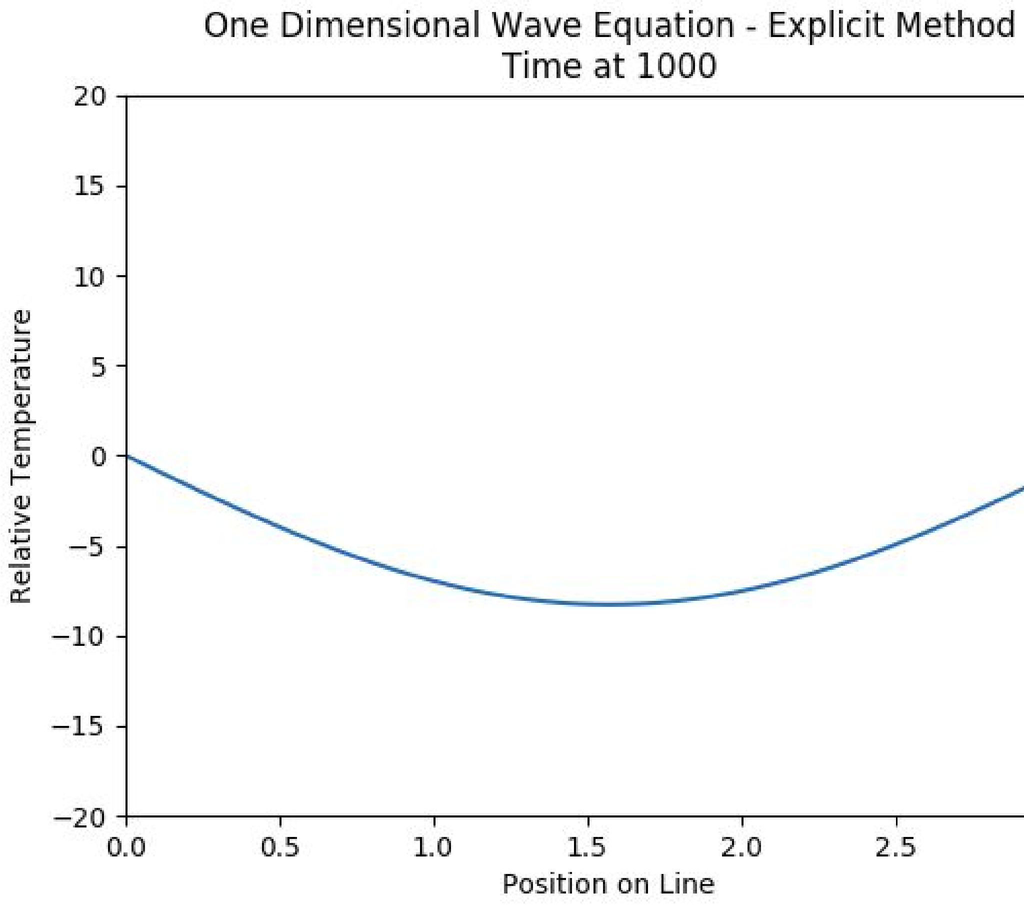
With our h as
αΔt^2 / (Δx)^2

This solver will now go through all our iterations and generate our values of u.
Here they are:




One Dimensional Wave Equation - Explicit Method
Time at 500
Relative Temperature
Position on Line
20
15
10
5
0
−5
−10
−15
−20
0.0
0.5
1.0
1.5
2.0
2.5
3.0


One Dimensional Wave Equation - Explicit Method
Time at 750
Relative Temperature
Position on Line
20
15
10
5
0
−5
−10
−15
−20
0.0
0.5
1.0
1.5
2.0
2.5
3.0


One Dimensional Wave Equation - Explicit Method
Time at 1000
Relative Temperature
Position on Line
20
15
10
5
0
−5
−10
−15
−20
0.0
0.5
1.0
1.5
2.0
2.5
3.0