# Assignment - 3

## FEM and CFD Theory
*ME3180*

**Name: Harshit Shambharkar**
**Roll No: ME21BTECH11019**

## 2D unsteady Heat Conduction

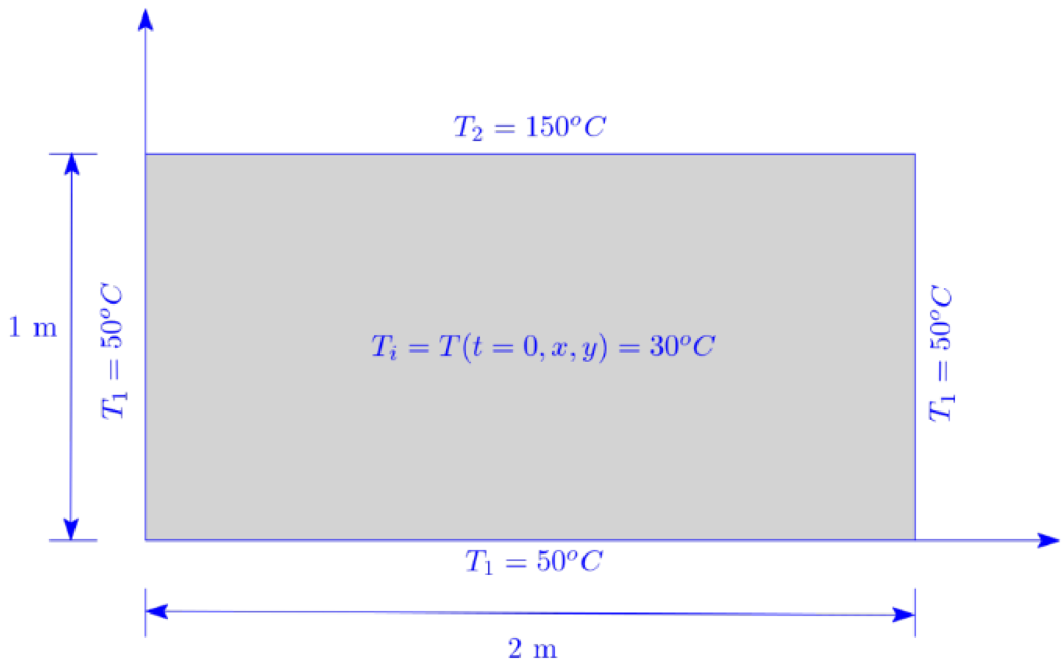


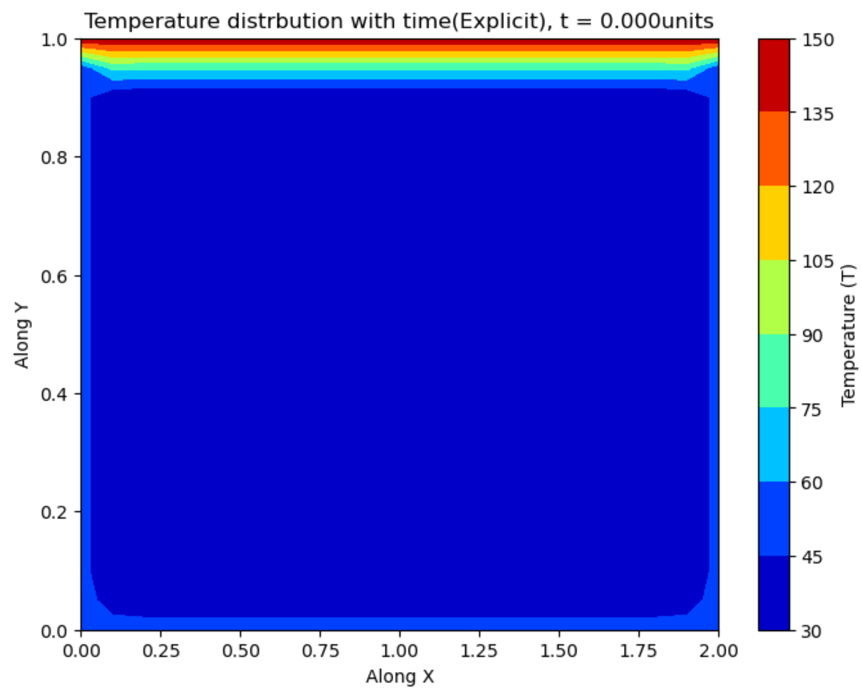

**Figure 1:** Physical domain of the plate with rectangular cross-section.

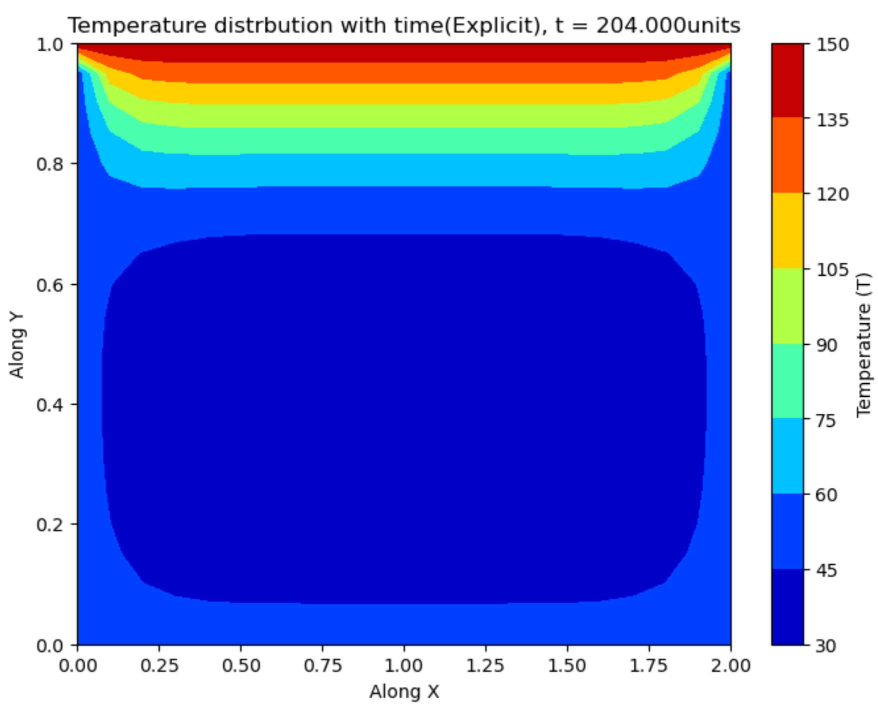
Snapshots of various methods to solve the equation are attached

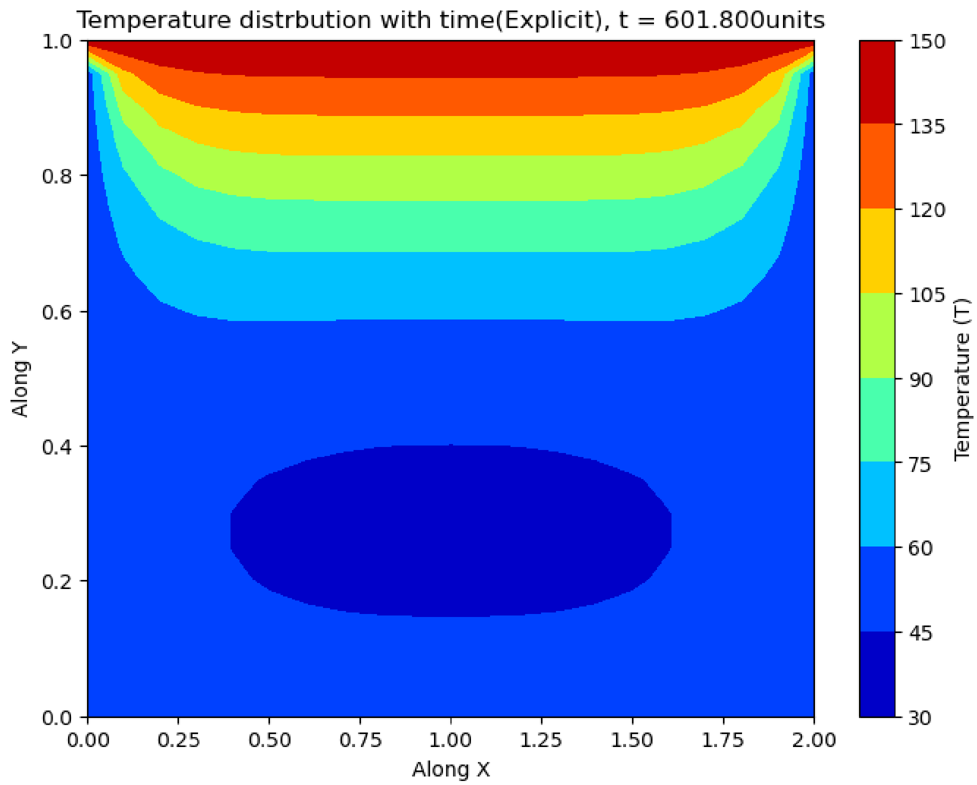
## Explicit Method

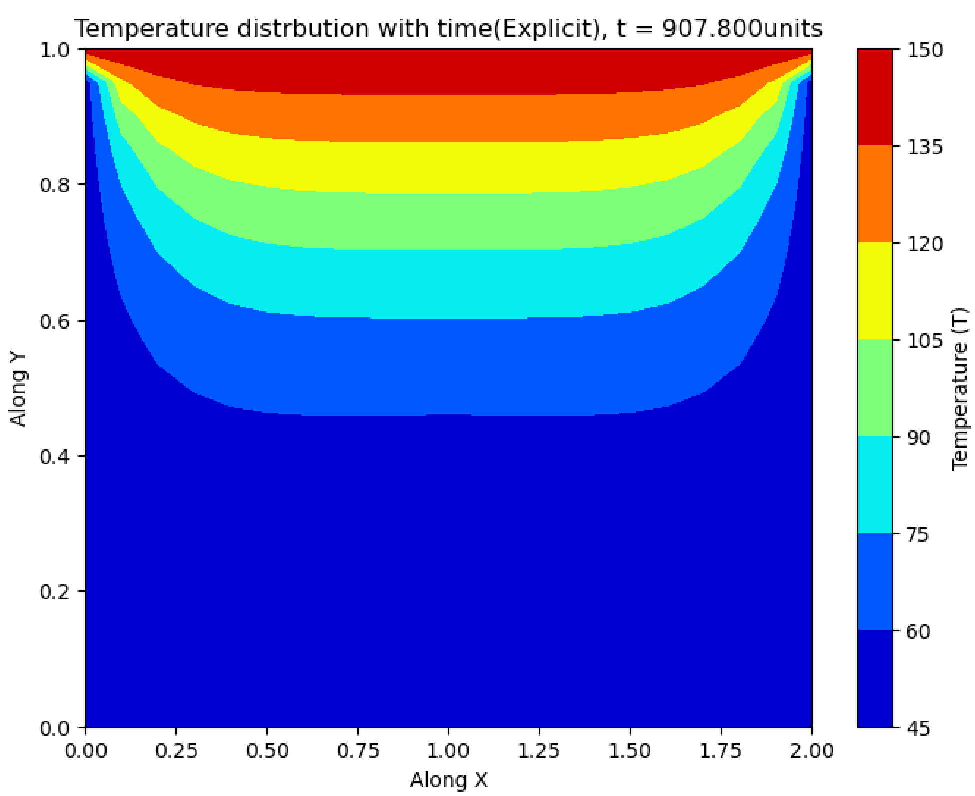
The contour is plotted at different intervals so as to showcase how it is changing with time:

```
In [18]: T_e = np.ones((nx, ny))

#Imposing Initial Conditions
T_e = T_init*T_e

# Initialising Boundary Conditions
T_e[0,:] = T_1
T_e[:,0] = T_1
T_e[:, ny-1] = T_1
T_e[nx-1,:] = T_2


t = 0.0 # time, initially it is 0
dt = 10.2 # time step

rx = alpha*dt/(dx**2)
ry = alpha*dt/(dy**2)

T_e_old = np.copy(T_e)

while t <= max_time:

    for i in range(1,nx-1):
        for j in range(1,ny-1):
            T_e[i,j] = rx*(T_e_old[i+1, j] + T_e_old[i-1, j]) + (1 - 2*rx - 2*ry)*T_e_old[i,j] + ry*(T_e_old[i, j+1]

    if(np.max(np.max(np.abs(T_e - T_e_old))) < Tolerance):
        print(f"Steady State reached at t = {t} units of time")
        break

    tt = int(t)
    if  0 <= (tt)%100 <= 8:
        # Plot for the Temperature Distribution
        X, Y = np.meshgrid(x, y)
        plt.figure(figsize=(8,6))

        contour = plt.contourf(X, Y, T_e, cmap=plt.cm.jet)
        #plt.contour(X, Y, T_e, levels = 20, cmap = plt.cm.jet)
        plt.colorbar(contour, label='Temperature (T)')
        plt.title(f'Temperature distrbution with time(Explicit), t = {t:.3f}units')
        plt.xlabel("Along X")
        plt.ylabel("Along Y")
        plt.show()

    t = t + dt
    T_e_old = np.copy(T_e)
```


Temperature distrbution with time(Explicit), t = 0.000units
Along Y
Along X
Temperature (T)
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
150
135
120
105
90
75
60
45
30


Temperature distrbution with time(Explicit), t = 204.000units
Along Y
Along X
Temperature (T)
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
150
135
120
105
90
75
60
45
30

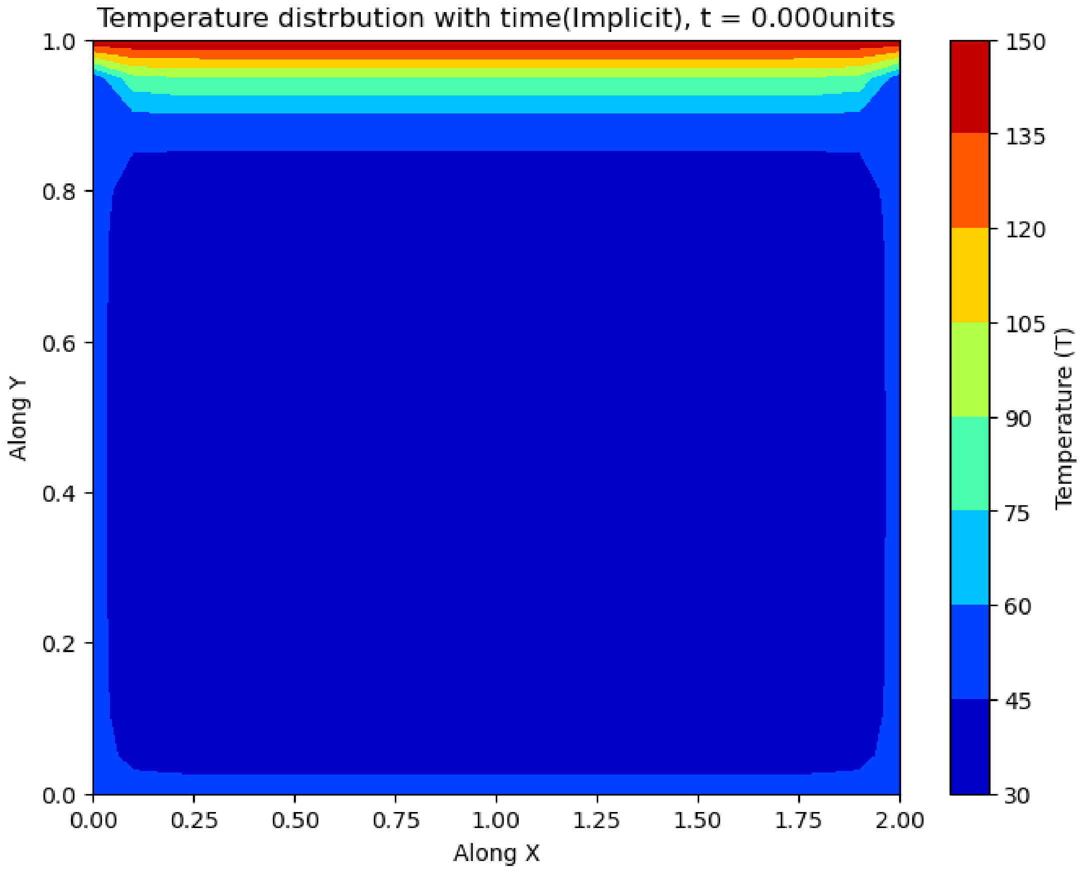

Temperature distrbution with time(Explicit), t = 601.800units
Along Y
Along X
Temperature (T)
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
150
135
120
105
90
75
60
45
30

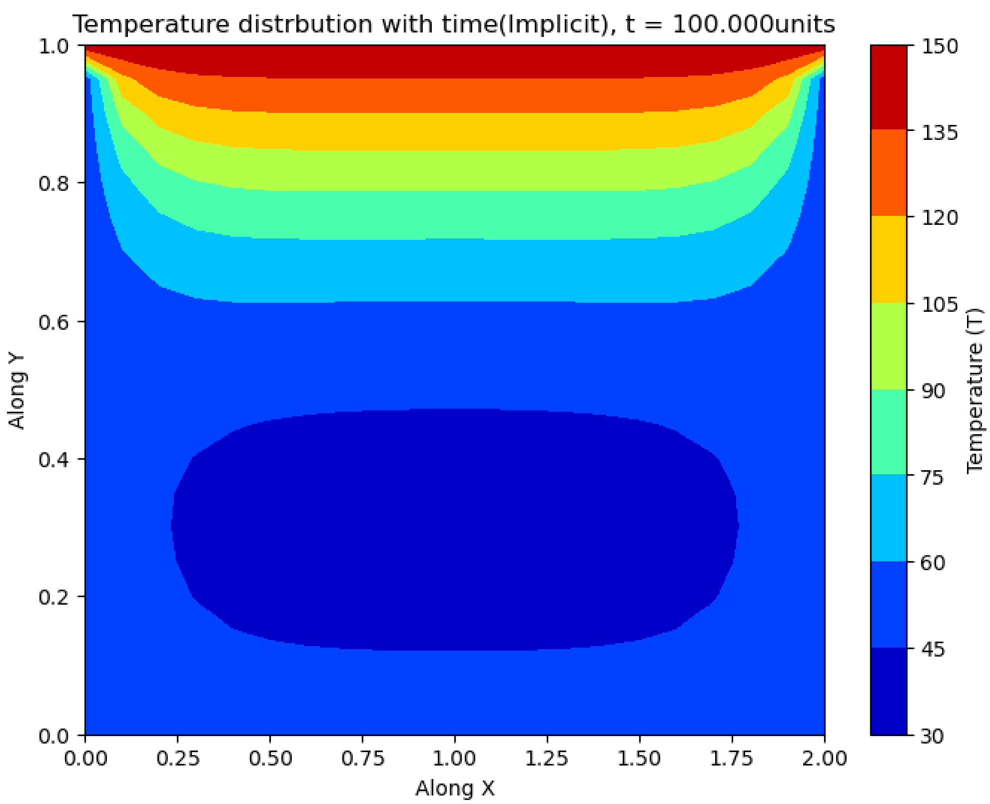

Temperature distrbution with time(Explicit), t = 907.800units
Along Y
Along X
Temperature (T)
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
150
135
120
105
90
75
60
45

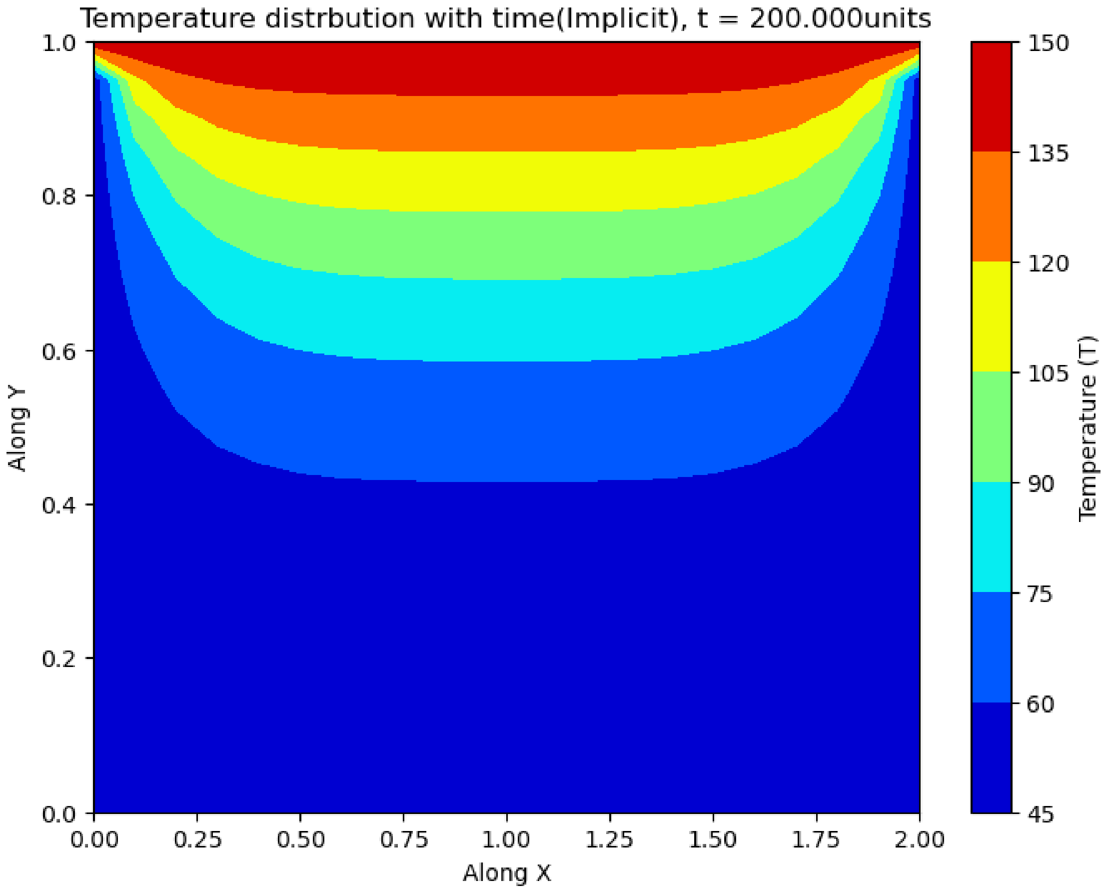
## Implicit Method

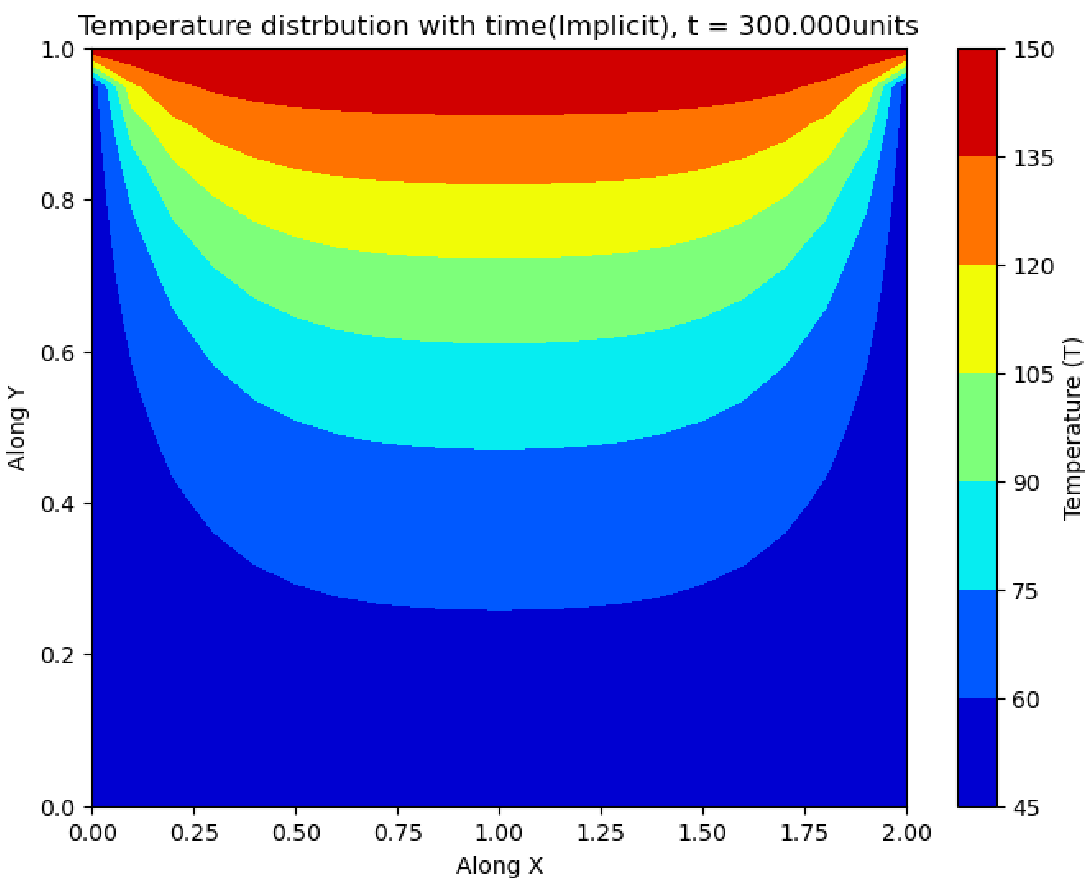
The contour is plotted at different intervals so as to showcase how it is changing with time:

In [20]:
```
#Imposing Initial Conditions
t = 0.0
dt = 10 # Since Impilcit Scheme is Unconditionally Stable, we can use any time step

rx = alpha*dt/(dx**2)
ry = alpha*dt/(dy**2)

T_i = np.ones((nx, ny))
T_i = T_init*T_i


T_i[0,:] = T_1
T_i[:,0] = T_1
T_i[:, ny-1] = T_1
T_i[nx-1,:] = T_2

T_i_old = np.copy(T_i)
T_prev = np.copy(T_i)

while t <= max_time:

    Error = 1
    while Error > Tolerance:
        for i in range(1,nx-1):
            for j in range(1, ny-1):
                T_i[i, j] = (T_prev[i,j] + rx*(T_i[i-1, j] + T_i[i+1, j]) + ry*(T_i[i, j-1] + T_i[i, j+1]))/ (1 + 2*


        Error = np.max(np.max(np.abs(T_i - T_i_old)))
        T_i_old = np.copy(T_i)

    if(np.max(np.max(np.abs(T_i - T_prev))) < Tolerance):
        print(f"Steady State reached at t, {t} units of time")
        break


    tt = int(t)
    if  0 <= (tt)%100 <= 8:
        # Plot for the Temperature Distribution
        X, Y = np.meshgrid(x, y)
        plt.figure(figsize=(8,6))

        contour = plt.contourf(X, Y, T_i, cmap=plt.cm.jet)
        #plt.contour(X, Y, T_e, levels = 20, cmap = plt.cm.jet)
        plt.colorbar(contour, label='Temperature (T)')
        plt.title(f'Temperature distrbution with time(Implicit), t = {t:.3f}units')
        plt.xlabel("Along X")
        plt.ylabel("Along Y")
        plt.show()
    T_prev = np.copy(T_i)
    t = t + dt
```


Temperature distrbution with time(Implicit), t = 0.000units
Along Y
Along X
Temperature (T)
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
30
45
60
75
90
105
120
135
150


Temperature distrbution with time(Implicit), t = 100.000units
Along Y
Along X
Temperature (T)
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
30
45
60
75
90
105
120
135
150


Temperature distrbution with time(Implicit), t = 200.000units
Along Y
Along X
Temperature (T)
150
135
120
105
90
75
60
45
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Temperature distrbution with time(Implicit), t = 300.000units
Along Y
Along X
Temperature (T)
150
135
120
105
90
75
60
45
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

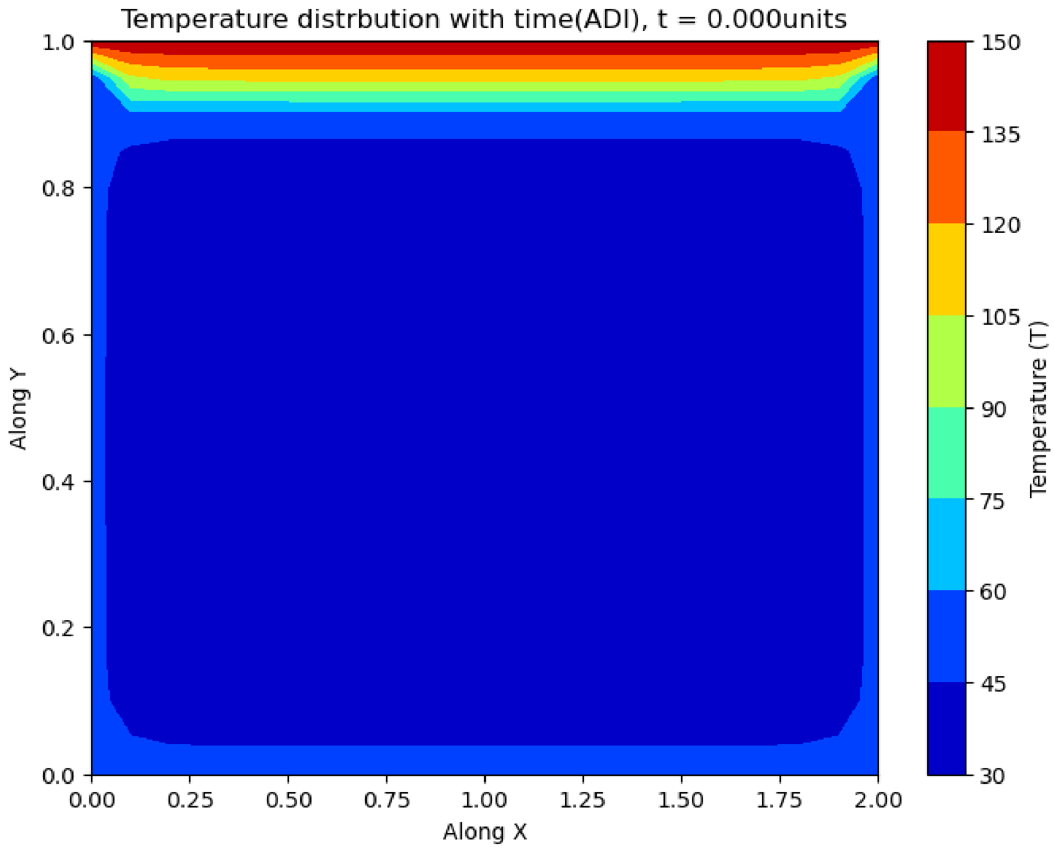
**Alternate Direct Implicit:**

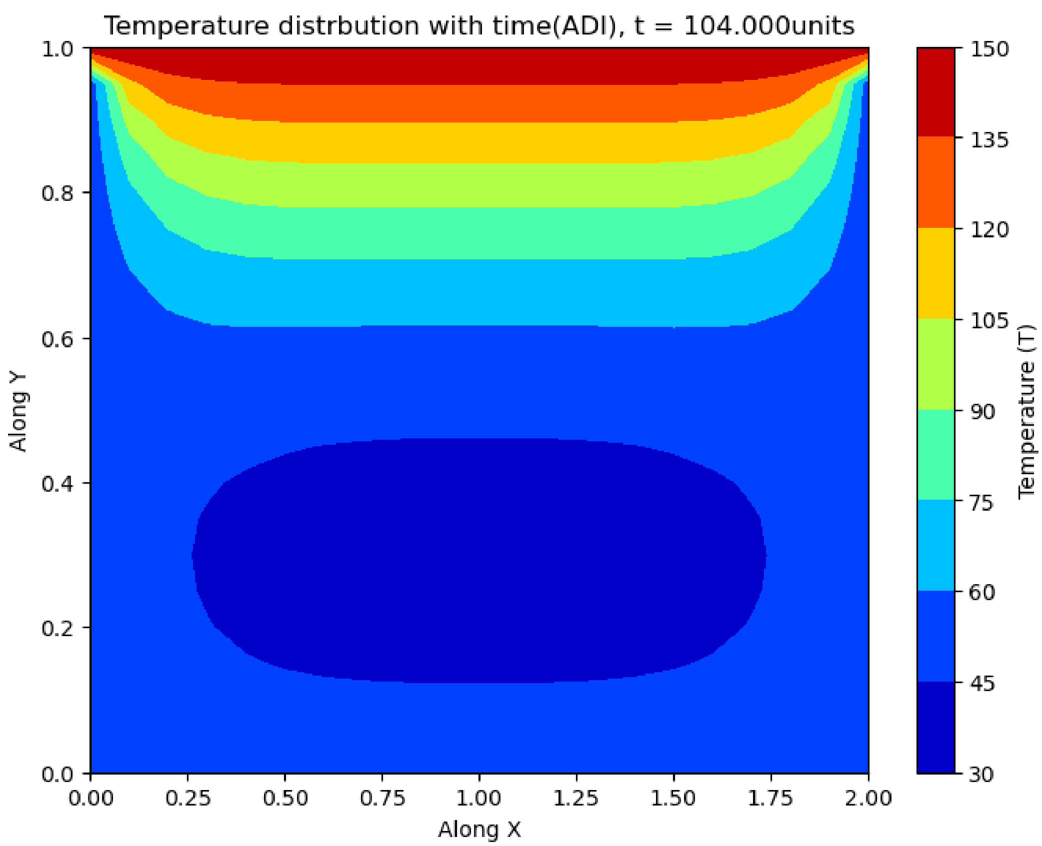
The contour is plotted at different intervals so as to showcase how it is changing with time:

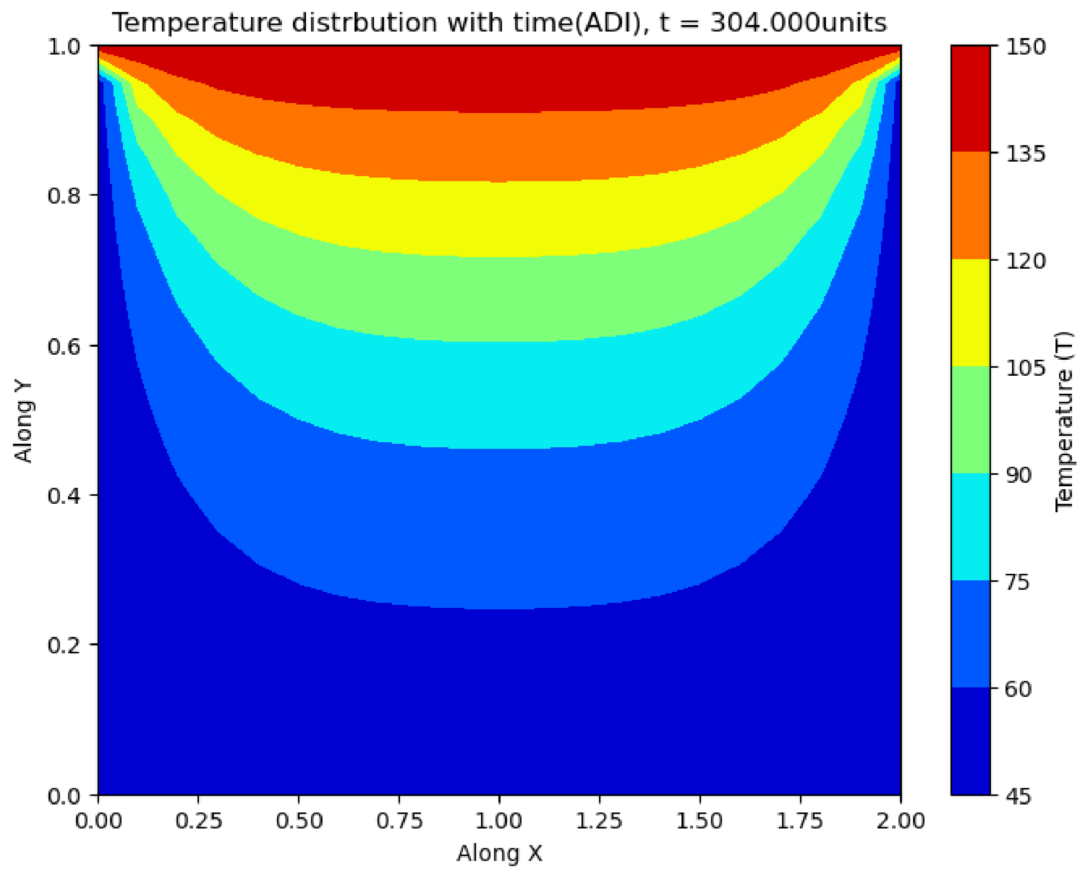
Alternate Direction Implicit

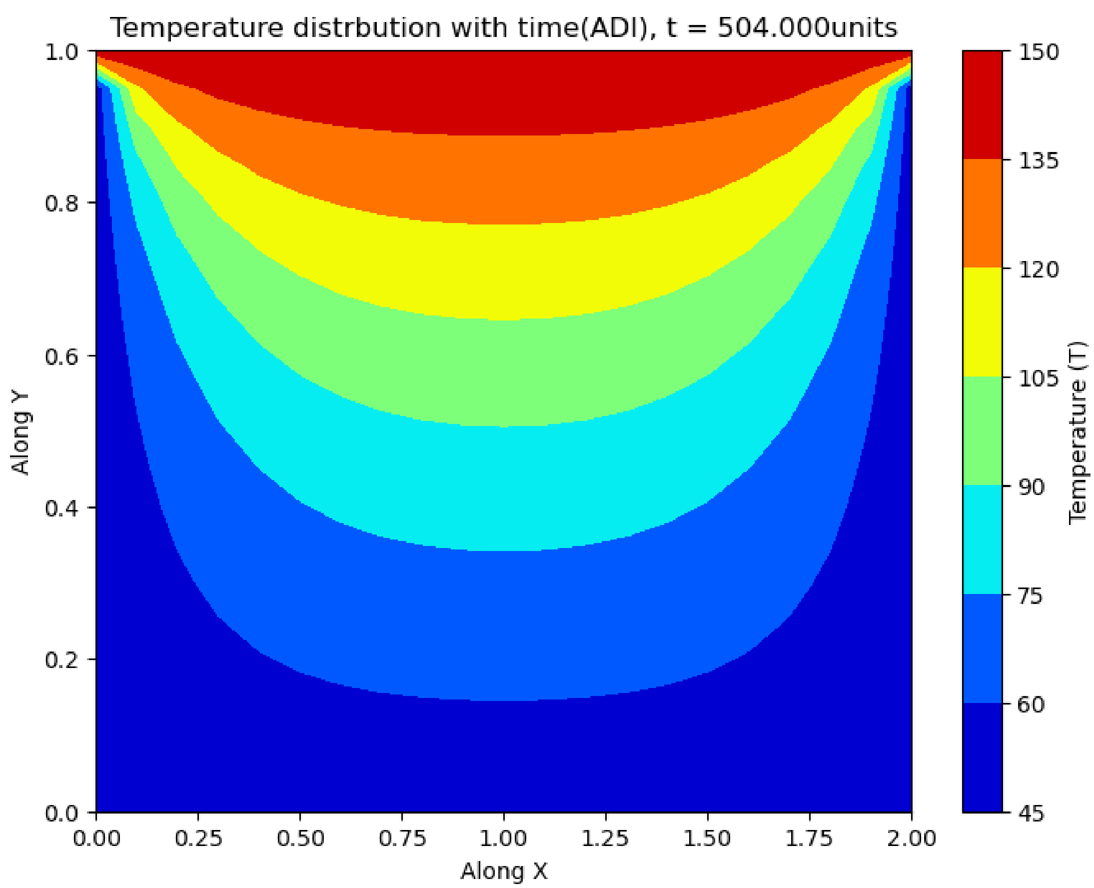
```
In [22]: t = 0.0
dt = 8

rx = alpha*dt/(dx**2)
ry = alpha*dt/(dy**2)

T_adt = np.ones((nx, ny))

T_adt = T_init*T_adt

T_adt[0,:] = T_1
T_adt[:,0] = T_1
T_adt[:, ny-1] = T_1
T_adt[nx-1,:] = T_2

T_adt_old = np.copy(T_adt)

while t <= max_time:

    Temp = np.copy(T_adt_old)

    # Sweeping in X - Direction
    a = (1 + ry)
    b = 0.5*ry
    c = 0.5*ry

    T0 = T_1
    Tn = T_1
    for i in range(1,nx-1):
        d = np.zeros(ny)
        for j in range(1, ny-1):
            d[j] = 0.5*rx*T_adt_old[i-1,j] + (1 - rx)*T_adt_old[i,j] + 0.5*rx*T_adt_old[i+1,j]

        Temp[i,:] = Thomas_algo(a,b,c,d, T0, Tn,ny)

    T_adt = np.copy(Temp)
    # Sweeping in Y Direction
```

```
    # Sweeping in Y Direction

    a = 1 + rx
    b = 0.5*rx
    c = 0.5*rx

    T0 = T_1
    Tn = T_2
    for j in range(1, ny-1):
        d = np.zeros(nx)
        for i in range(1, nx-1):
            d[i] = 0.5*ry*Temp[i,j-1] + (1 - ry)*Temp[i,j] + 0.5*ry*Temp[i,j+1]

        T_adt[:,j] = Thomas_algo(a,b,c,d,T0,Tn,nx)

    if(np.max(np.max(np.abs(T_adt - T_adt_old))) < Tolerance):
        print(f"Steady State reached at t, {t} units of time")
        break

    tt = int(t)
    if  0 <= (tt)%100 <= 8:
        # Plot for the Temperature Distribution
        X, Y = np.meshgrid(x, y)
        plt.figure(figsize=(8,6))

        contour = plt.contourf(X, Y, T_adt, cmap=plt.cm.jet)
        #plt.contour(X, Y, T_e, levels = 20, cmap = plt.cm.jet)
        plt.colorbar(contour, label='Temperature (T)')
        plt.title(f'Temperature distrbution with time(ADI), t = {t:.3f}units')
        plt.xlabel("Along X")
        plt.ylabel("Along Y")
        plt.show()

    t = t + dt
    T_adt_old = np.copy(T_adt)
```


Temperature distrbution with time(ADI), t = 0.000units
Along Y
Along X
Temperature (T)
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
30
45
60
75
90
105
120
135
150


Temperature distrbution with time(ADI), t = 104.000units
Along Y
Along X
Temperature (T)
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
30
45
60
75
90
105
120
135
150

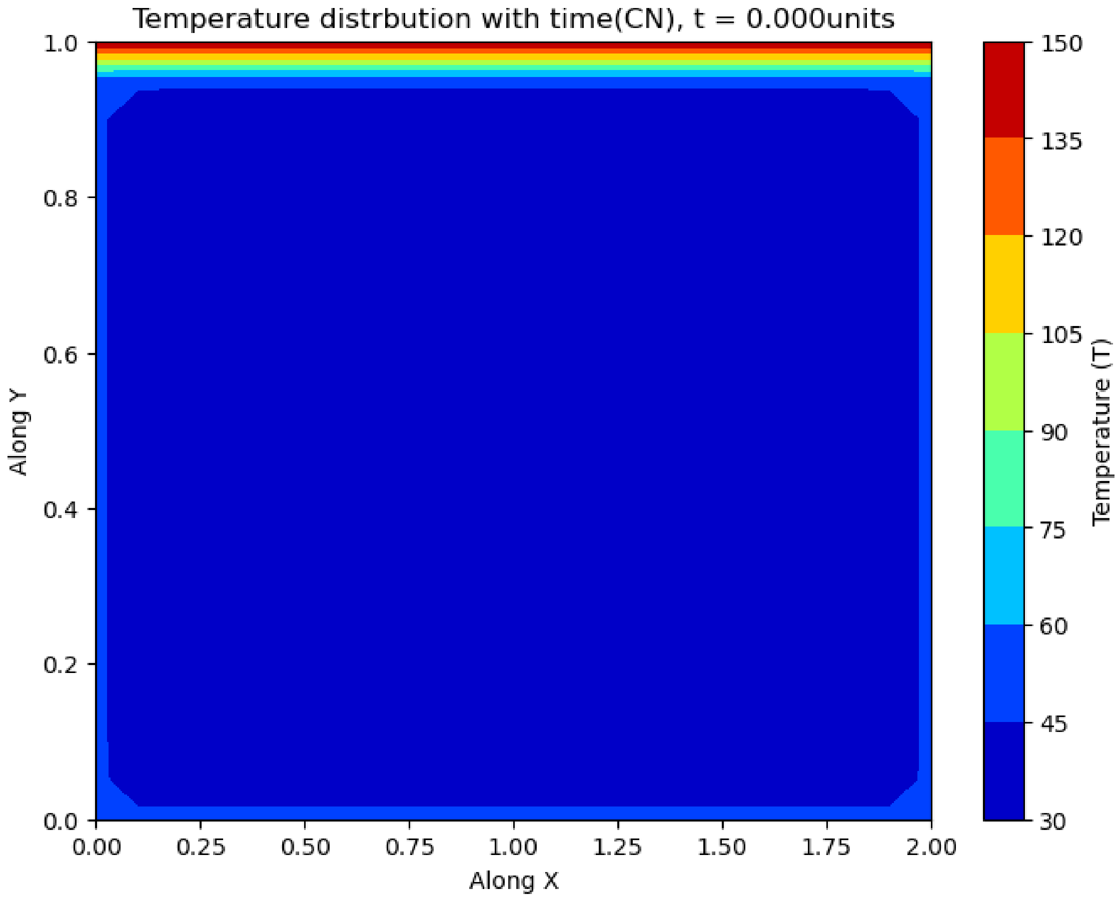

Temperature distrbution with time(ADI), t = 304.000units
Along Y
Along X
Temperature (T)
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
150
135
120
105
90
75
60
45

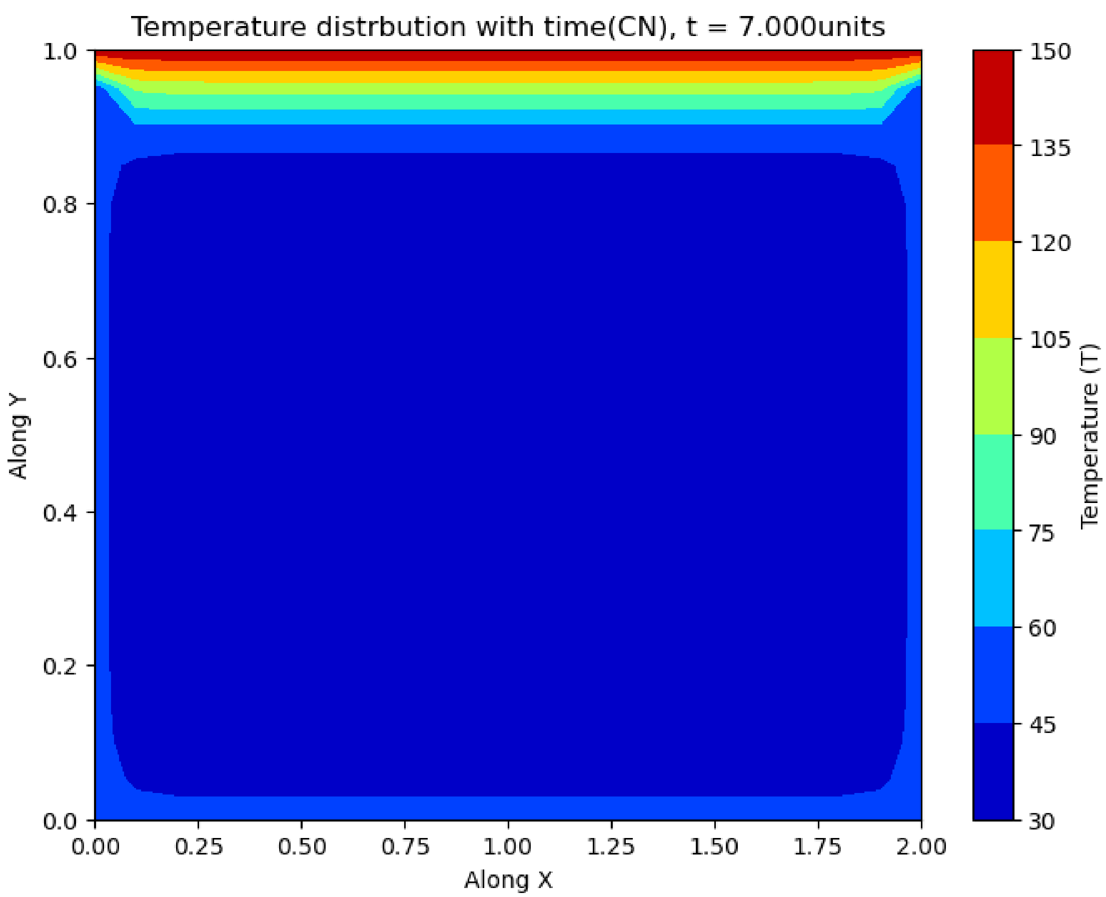

Temperature distrbution with time(ADI), t = 504.000units
Along Y
Along X
Temperature (T)
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
150
135
120
105
90
75
60
45

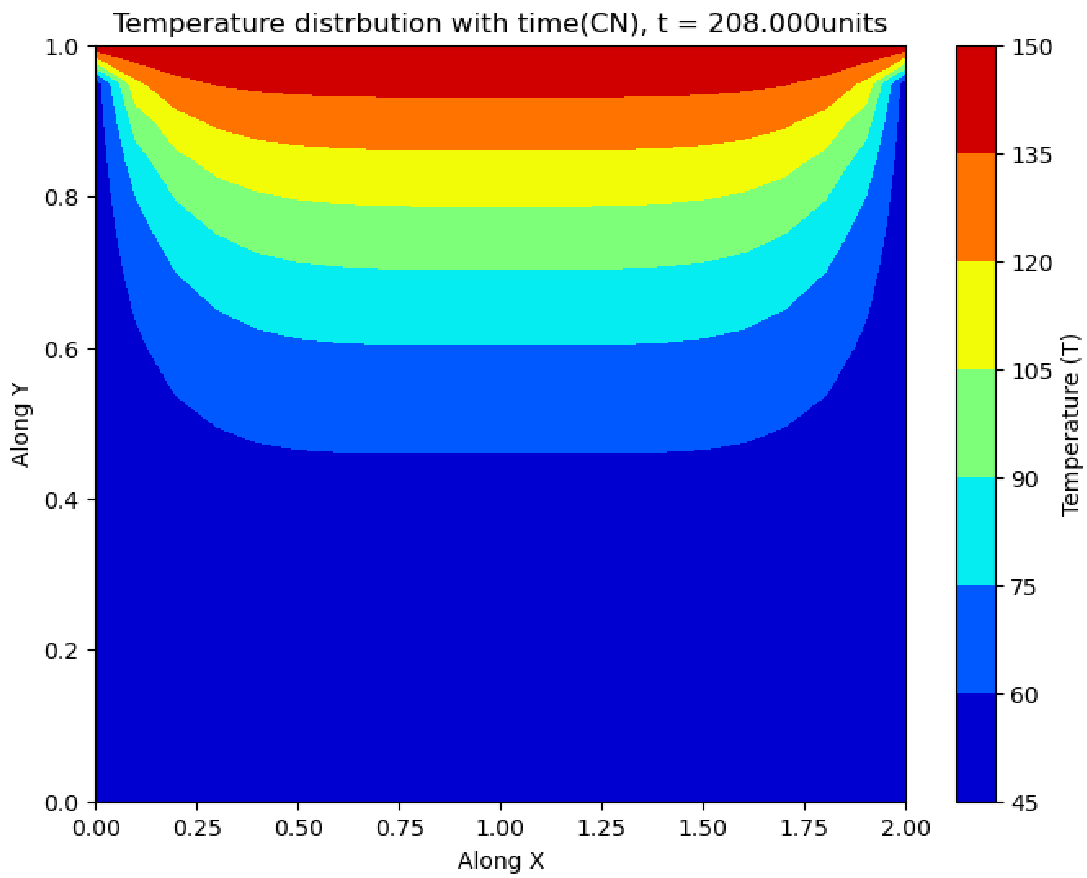
## Nickolson Method:

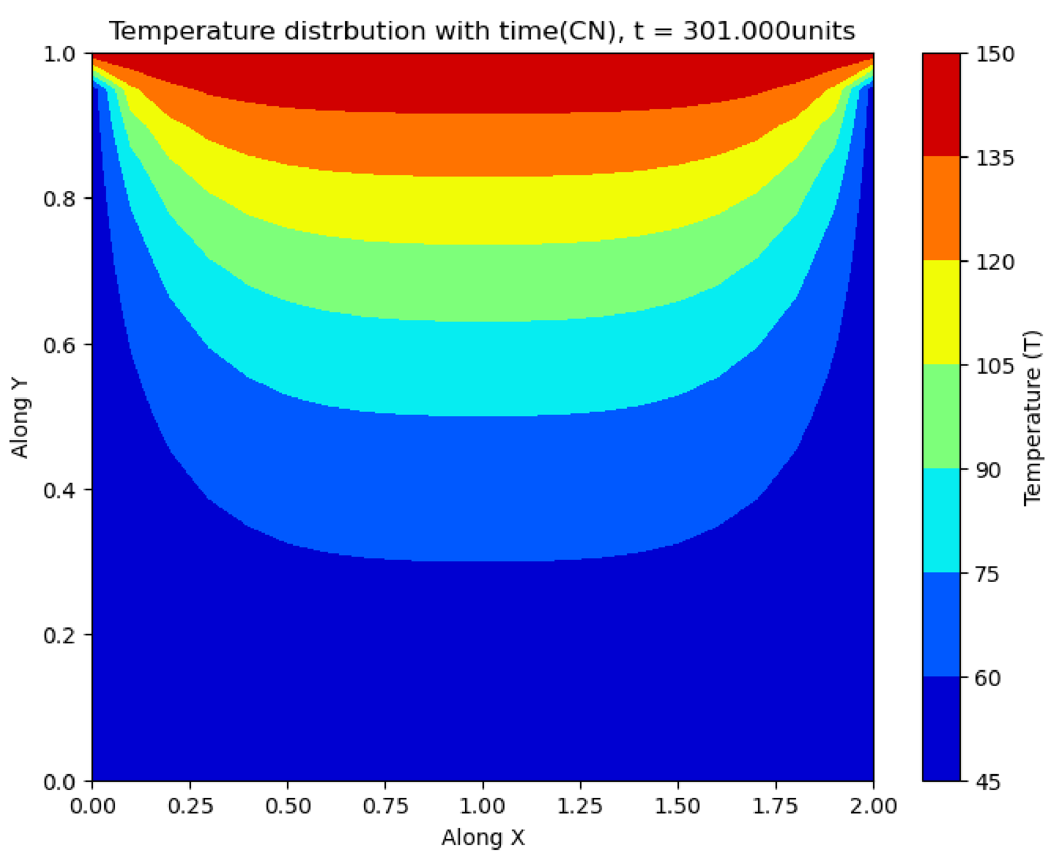
The contour is plotted at different intervals so as to showcase how it is changing with time:

In [27]:
```
T_CN = np.ones((nx, ny))

t = 0.0
dt = 1.0

rx = alpha*dt/(dx**2)
ry = alpha*dt/(dy**2)

T_CN = T_CN*T_init

T_CN[0,:] = T_1
T_CN[:,0] = T_1
T_CN[:, ny-1] = T_1
T_CN[nx-1,:] = T_2

T_CN_old = np.copy(T_CN)

while t <= max_time:
    a = (1 + ry + rx)
    b = 0.5*ry
    c = 0.5*ry
    T0 = T_1
    Tn = T_1

    for i in range(1,nx-1):
        d = np.zeros(ny)
        for j in range(1,ny-1):
            term1 = 0.5*rx*(T_CN_old[i-1,j] - 2*T_CN_old[i, j] + T_CN_old[i+1,j])
            term2 = 0.5*ry*(T_CN_old[i,j-1] - 2*T_CN_old[i, j] + T_CN_old[i,j+1])
            term3 = 0.5*rx*(T_CN[i-1,j] + T_CN[i+1, j])
            d[j] = term1 + term2 + term3 + T_CN_old[i,j]


        T_CN[i,:] = Thomas_algo(a,b,c,d,T0,Tn,ny)


    if(np.max(np.max(np.abs(T_CN - T_CN_old))) < Tolerance):
        print(f"Steady State reached at t, {t} units of time")
        break

    tt = t
    if  0 <= int(tt)%100 <= 8:
        # Plot for the Temperature Distribution
        X, Y = np.meshgrid(x, y)
        plt.figure(figsize=(8,6))

        contour = plt.contourf(X, Y, T_CN, cmap=plt.cm.jet)
        #plt.contour(X, Y, T_e, levels = 20, cmap = plt.cm.jet)
        plt.colorbar(contour, label='Temperature (T)')
```

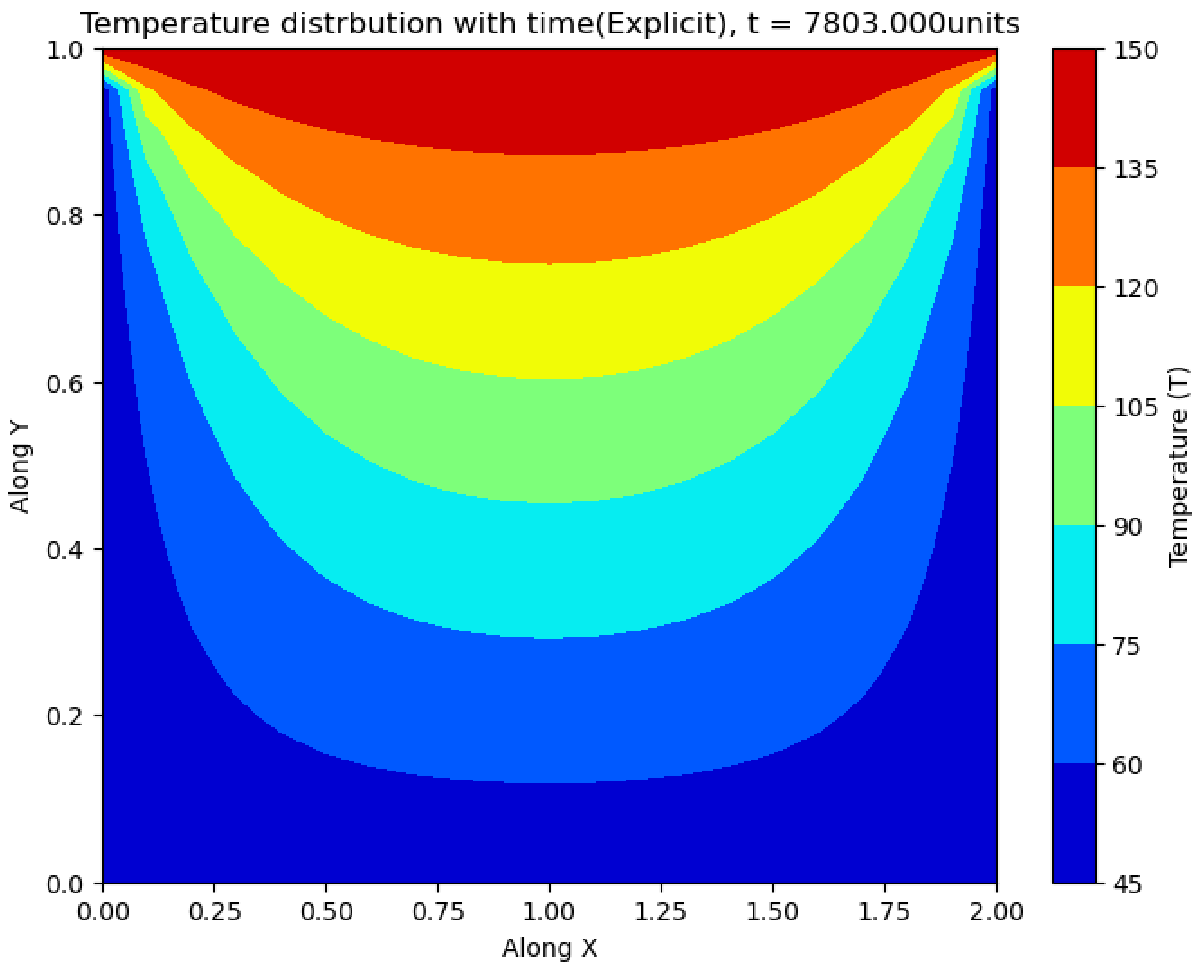

Temperature distrbution with time(CN), t = 0.000units
Along Y
Along X
Temperature (T)
150
135
120
105
90
75
60
45
30
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Temperature distrbution with time(CN), t = 7.000units
Along Y
Along X
Temperature (T)
150
135
120
105
90
75
60
45
30
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

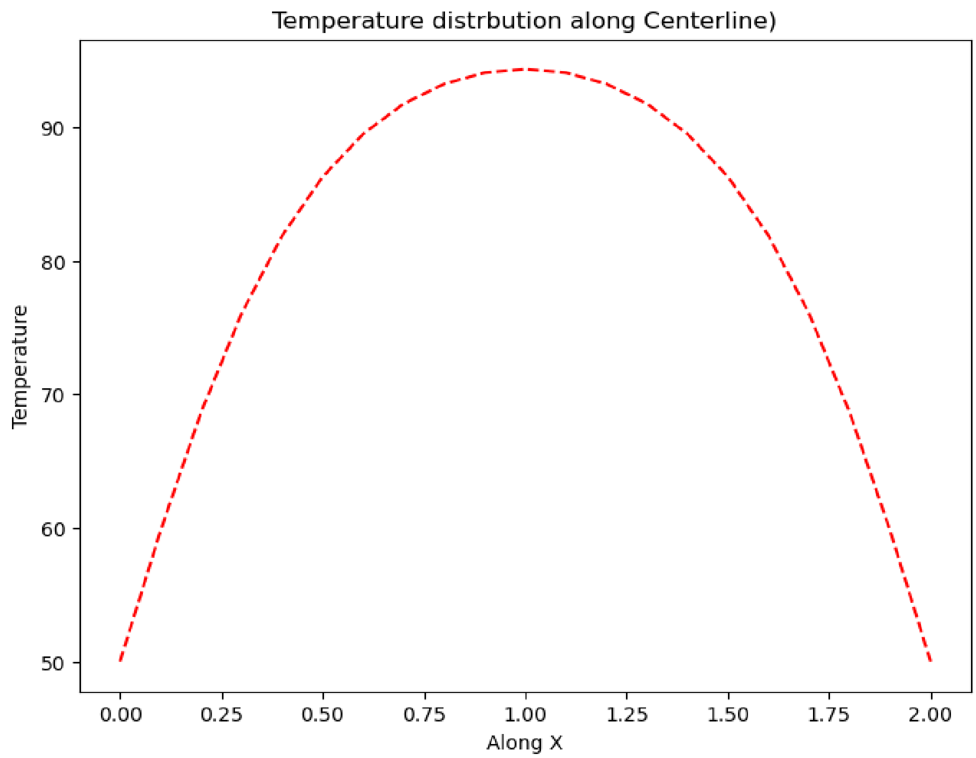

Temperature distrbution with time(CN), t = 208.000units
Along Y
Along X
Temperature (T)
150
135
120
105
90
75
60
45

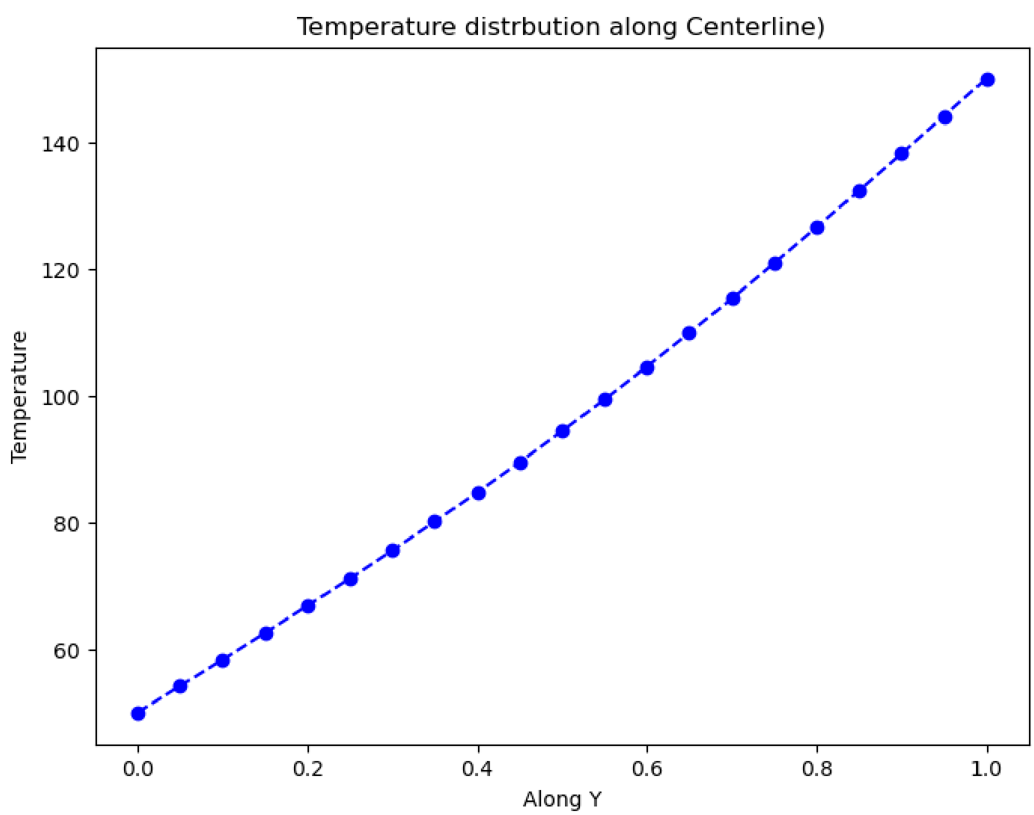

Temperature distrbution with time(CN), t = 301.000units
Along Y
Along X
Temperature (T)
150
135
120
105
90
75
60
45

## Plot for Steady state



Temperature distribution along Centerlines(along X and Y is plotted)



Grid Independence Test :

Temperature distribution along centerline is plotted for different grid sizes.