Assignment 2: Θ-method for Heat equation

Manan Doshi

11 March 2018

1 Introduction

The θ method is a semi-implicit forward time centered space scheme to solve the heat equation. The heat equation in one dimension is given as follows:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

2 Θ Scheme

The scheme to solve the equation is as follows:

$$\partial_t^+ U_j^k = (\theta)\partial_x^+\partial_x^- U_j^{k+1} + (1-\theta)\partial_x^+\partial_x^- U_j^k$$

$$\left(\frac{U_j^{k+1} - U_j^k}{\Delta t}\right) = (\theta)\left(\frac{U_{j+1}^{k+1} - 2U_j^{k+1} + U_{j-1}^{k+1}}{\Delta x^2}\right) + (1-\theta)\left(\frac{U_{j+1}^{k} - 2U_j^{k} + U_{j-1}^{k}}{\Delta x^2}\right)$$

Setting $\lambda_1 = \left(\frac{\Delta t}{\Delta x^2}\right)\theta$ and $\lambda_2 = \left(\frac{\Delta t}{\Delta x^2}\right)(\theta - 1)$, we can write it in a matrix-vector form,

$$\begin{bmatrix} 1+2\lambda_1 & -\lambda_1 & 0 & \dots & 0 \\ -\lambda_1 & 1+2\lambda_1 & -\lambda_1 & \ddots & 0 \\ 0 & -\lambda_1 & 1+2\lambda_1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -\lambda_1 & 1+2\lambda_1 \end{bmatrix} \begin{bmatrix} U_1^{k+1} \\ U_2^{k+1} \\ \vdots \\ \vdots \\ U_{J-1}^{k+1} \end{bmatrix} = \begin{bmatrix} 1+2\lambda_2 & -\lambda_2 & 0 & \dots & 0 \\ -\lambda_2 & 1+2\lambda_2 & -\lambda_2 & \ddots & 0 \\ 0 & -\lambda_2 & 1+2\lambda_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -\lambda_2 & 1+2\lambda_2 \end{bmatrix} \begin{bmatrix} U_1^{k} \\ U_2^{k} \\ \vdots \\ \vdots \\ U_{J-1}^{k} \end{bmatrix}$$

Taking the inverse, we can write it as a linear equation

$$U^{k+1} = \mathbf{A}(\mu, \theta)U^k$$

The eigenvalues of the matrix $\mathbf{A}$ dictate the stability of the scheme.

3 Stability Analysis using eigenvalues

It can be clearly seen in Figure1 that the method is stable for $\theta \geq 0.5$ and for $\theta < 0.5$ when $\mu < 0.5$. This is what we obtained from Von-Neumann Stability analysis of the method.

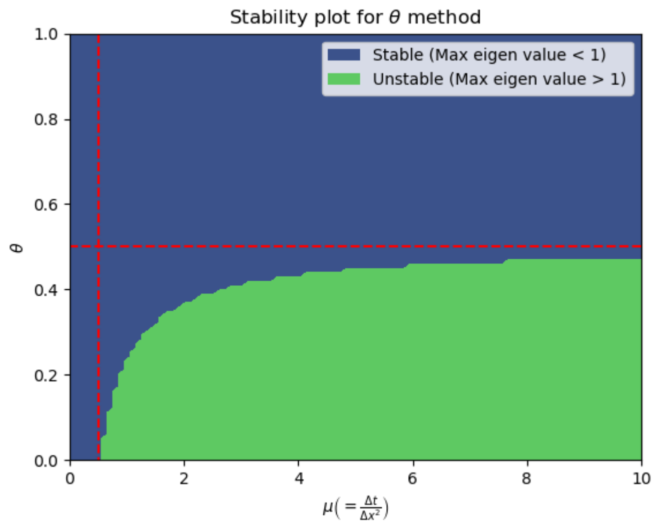

Figure 1: Stability plot fot θ method

4 Convergence study

The following set of plots show the solution for various values of μ and θ. The red line represents the exact solution and the blue line represents the numerical solution. The convergence plots are log log plotted against μ and Δt. The red line in this plot represents $\mu = 0.5$. The contour plot represents the solution in the x-t plane.

4.1 Explicit ($\theta = 0$)

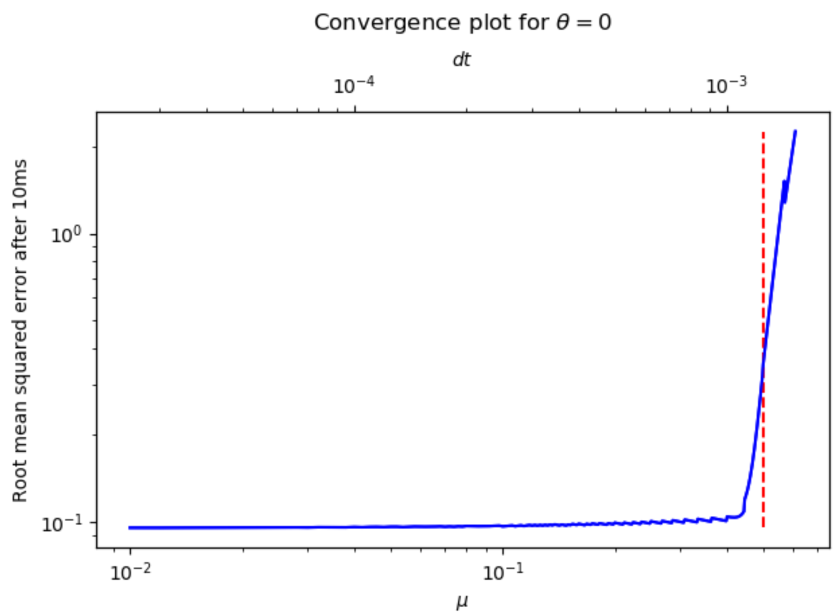

Figure 2: Convergence plot for $\theta = 0$

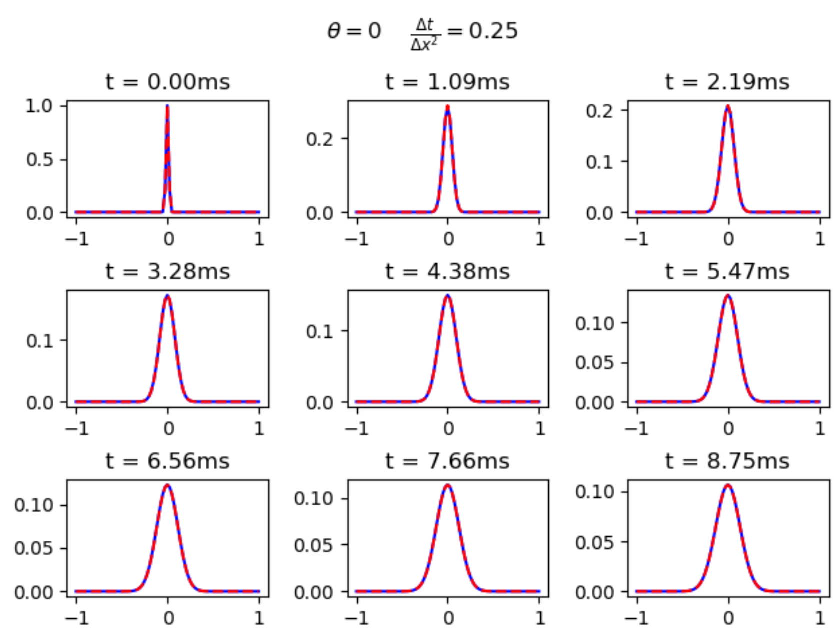

Figure 3: Solution for $\mu = 0.2$ and $\theta = 0$

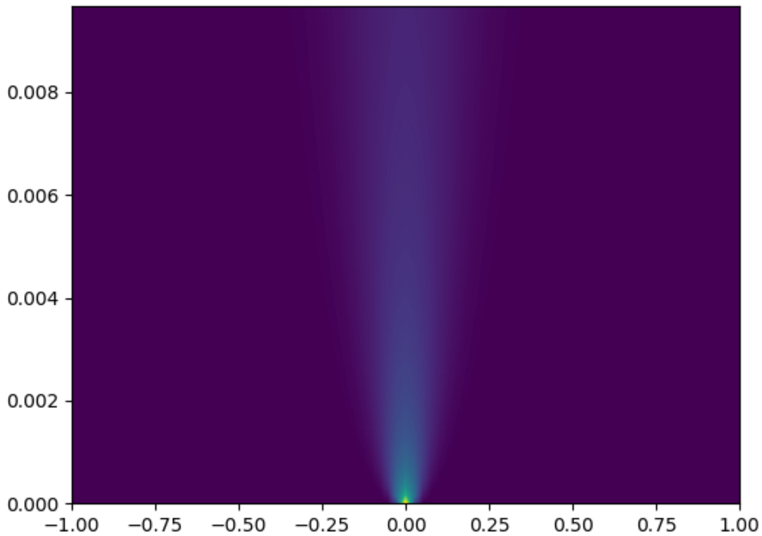

Figure 4: Solution for $\mu = 0.2$ and $\theta = 0$

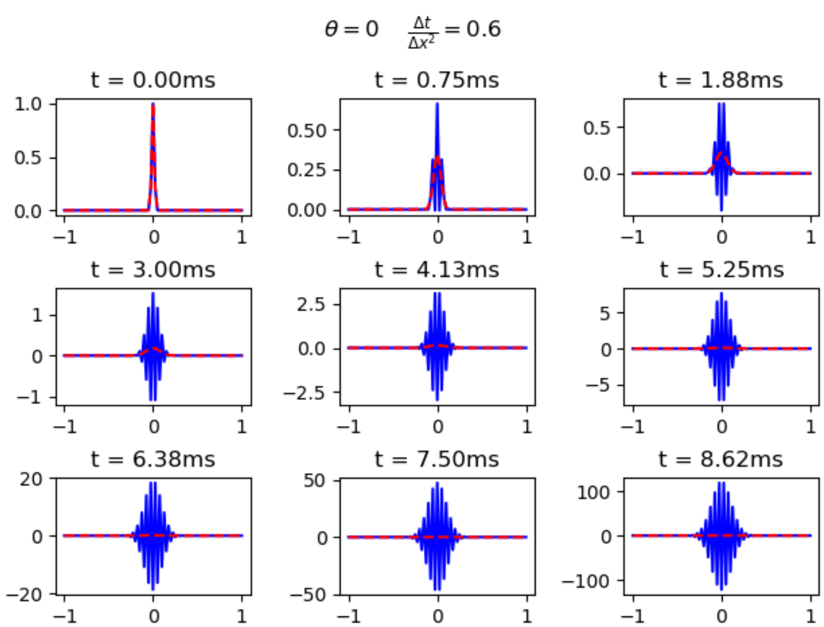

Figure 5: Solution for $\mu = 0.6$ and $\theta = 0$ (Blowup. Unstable)

4.2 Implicit ($\theta = 1$)

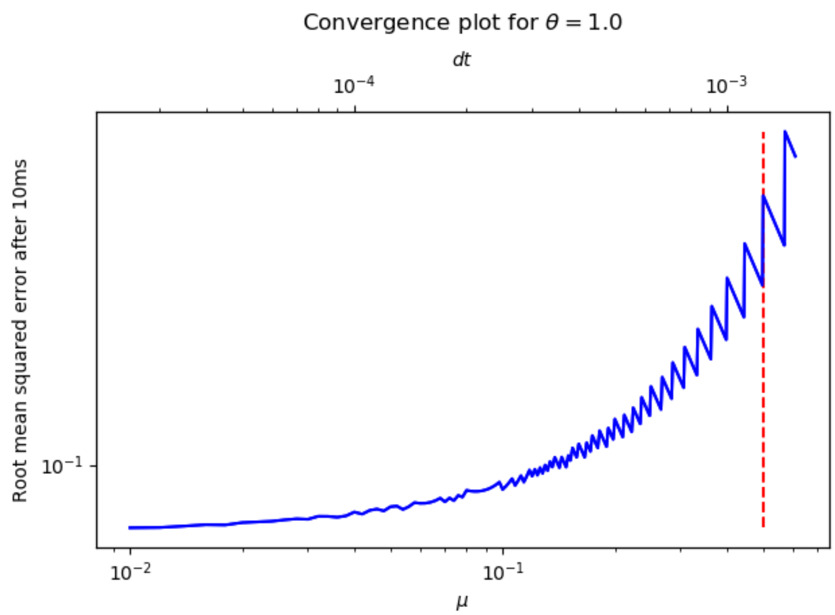

Figure 6: Convergence plot for $\theta = 1$

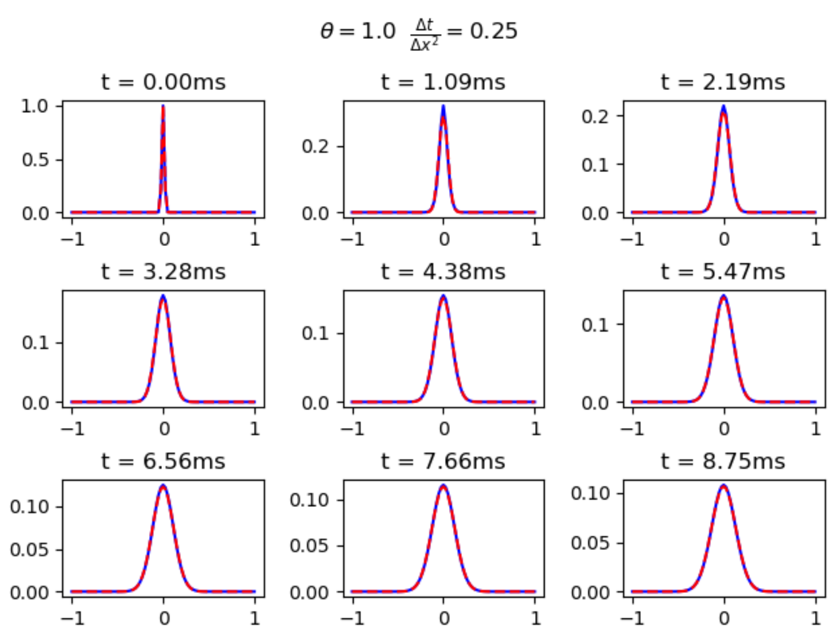

Figure 7: Solution for $\mu = 0.2$ and $\theta = 1$

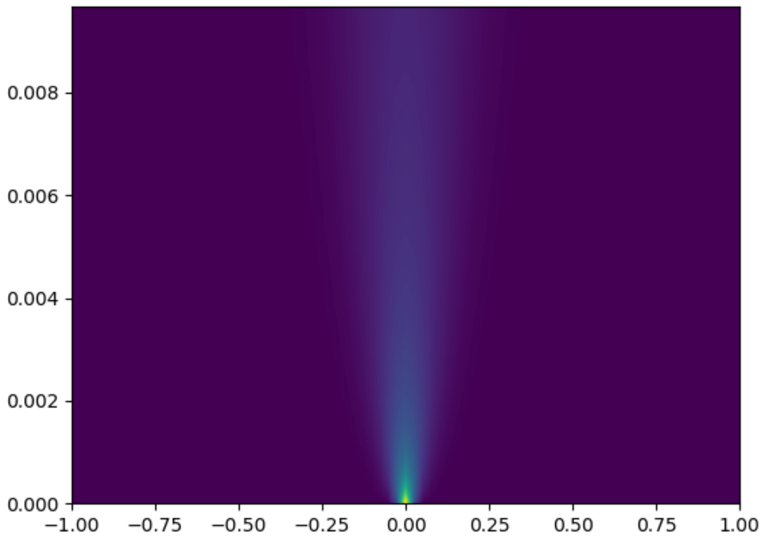

Figure 8: Solution for $\mu = 0.2$ and $\theta = 1$

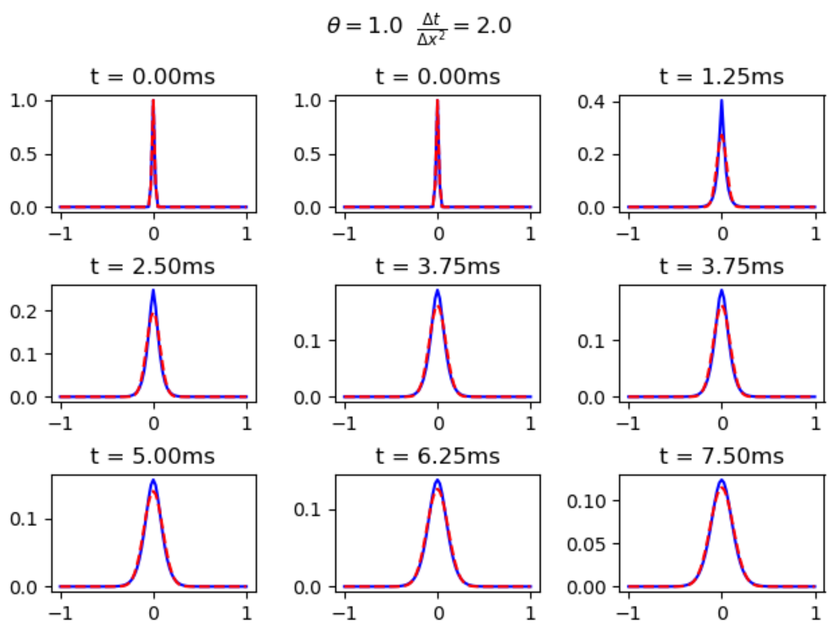

Figure 9: Solution for $\mu = 2.0$ and $\theta = 1$

4.3 CN ($\theta = 0.5$)

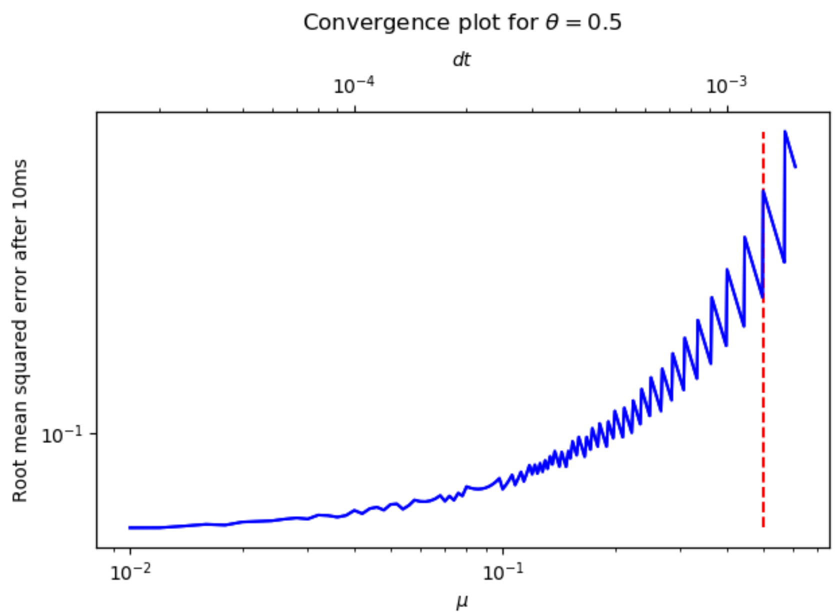

Figure 10: Convergence plot for $\theta = 0.5$

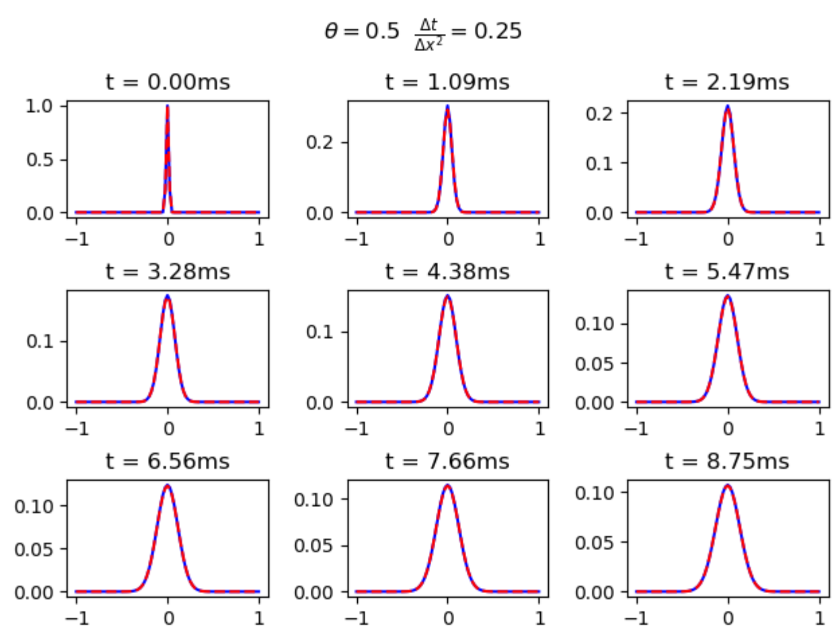

Figure 11: Solution for $\mu = 0.2$ and $\theta = 0.5$

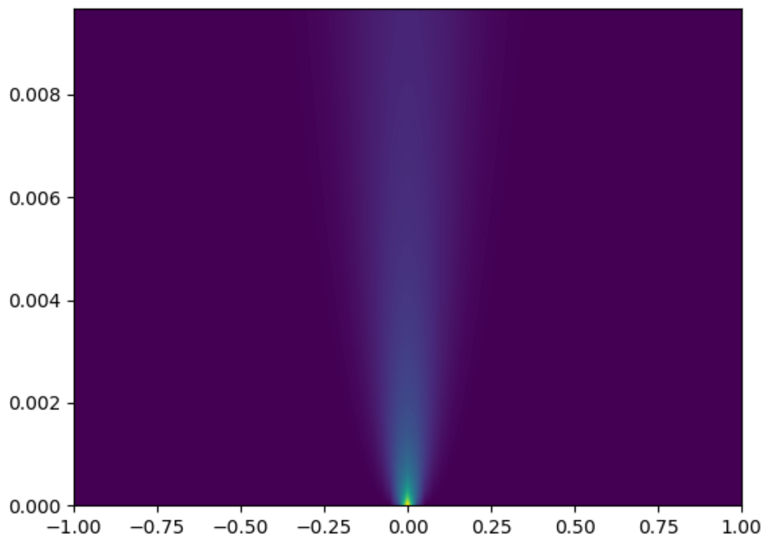

Figure 12: Solution for $\mu = 0.2$ and $\theta = 0.5$

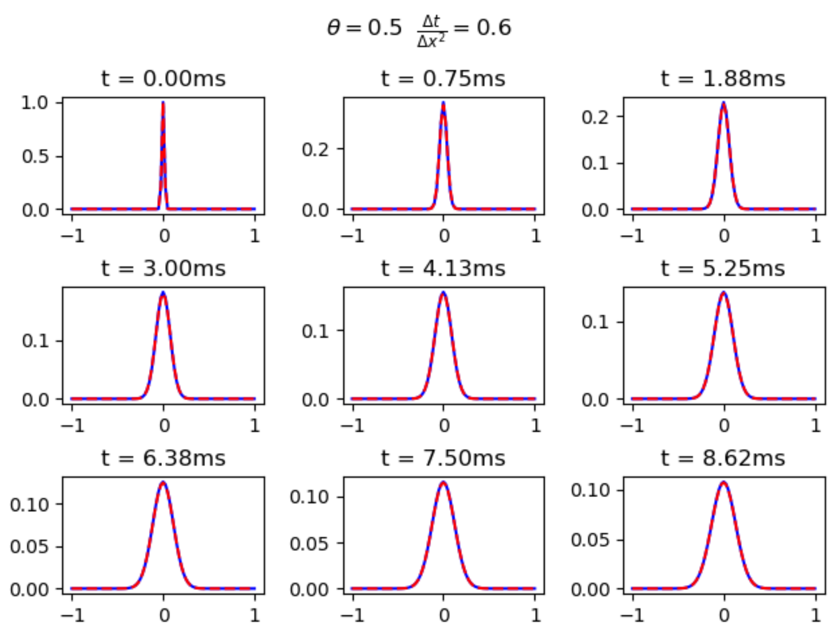

Figure 13: Solution for $\mu = 0.6$ and $\theta = 0.5$

4.4 $\theta = 0.6$

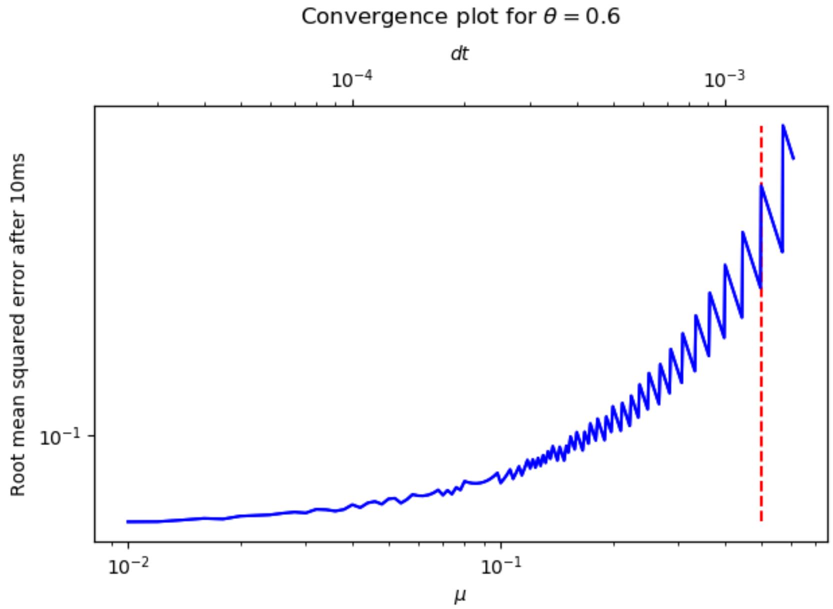

Figure 14: Convergence plot for $\theta = 0.6$

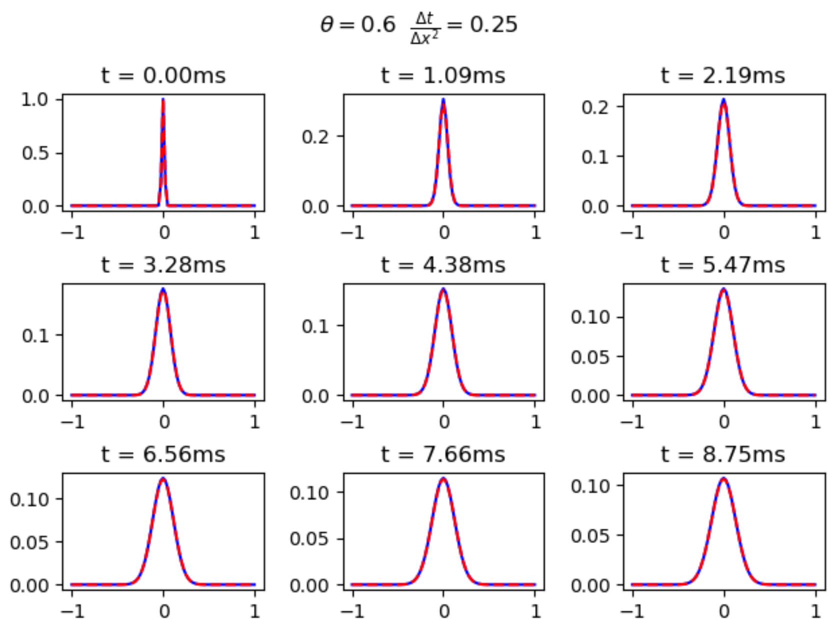

Figure 15: Solution for $\mu = 0.2$ and $\theta = 0.6$

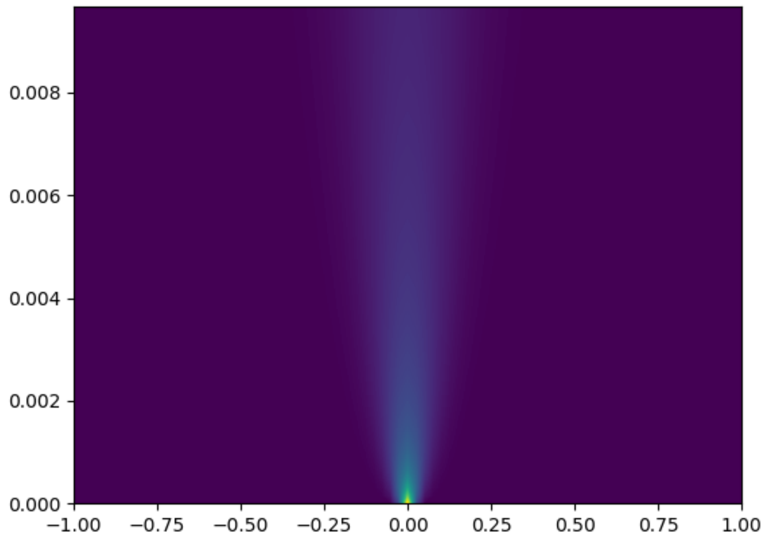

Figure 16: Solution for $\mu = 0.2$ and $\theta = 0.6$

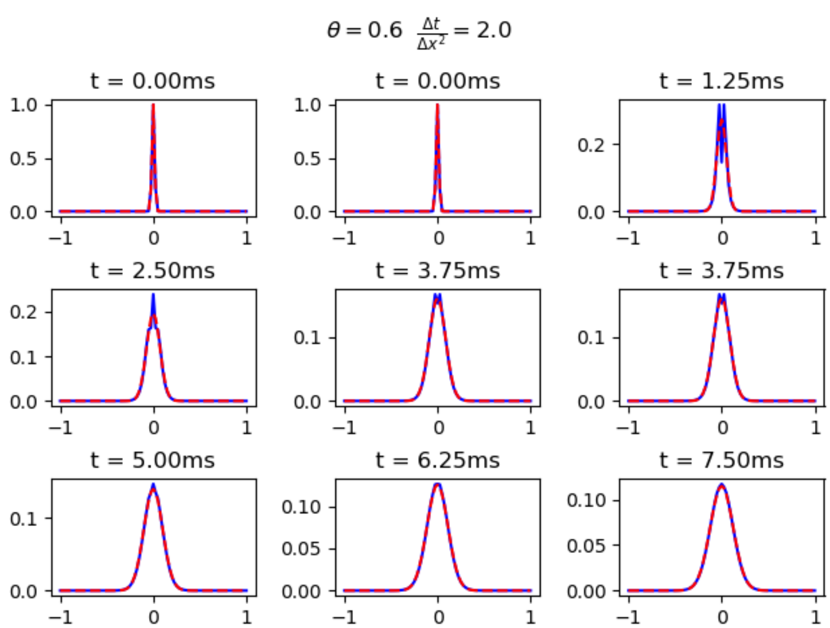

Figure 17: Solution for $\mu = 2.0$ and $\theta = 0.6$

4.5 $\theta = 0.4$

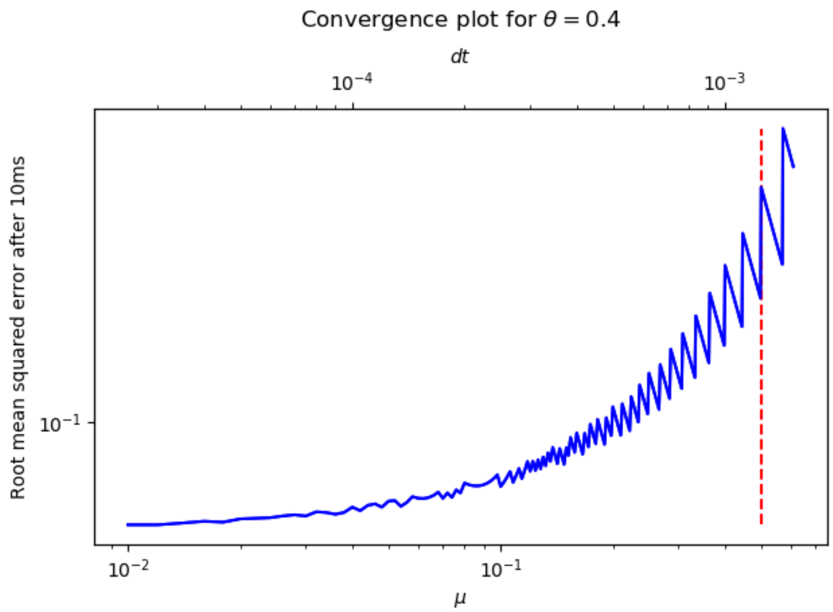

Figure 18: Convergence plot for $\theta = 0.4$

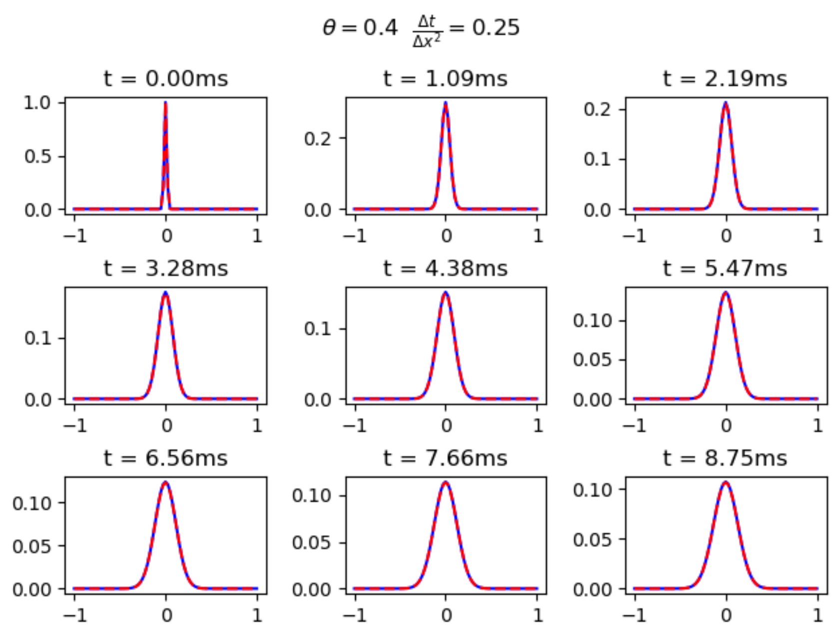

Figure 19: Solution for $\mu = 0.2$ and $\theta = 0.4$

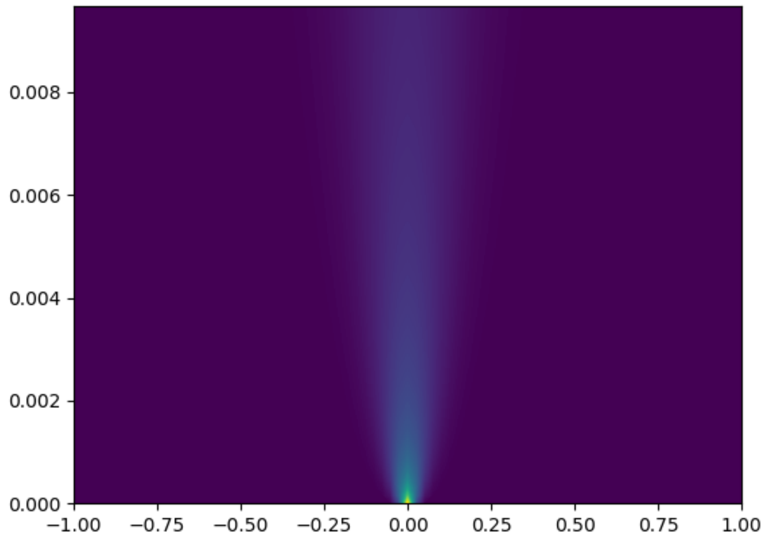

Figure 20: Solution for $\mu = 0.2$ and $\theta = 0.4$

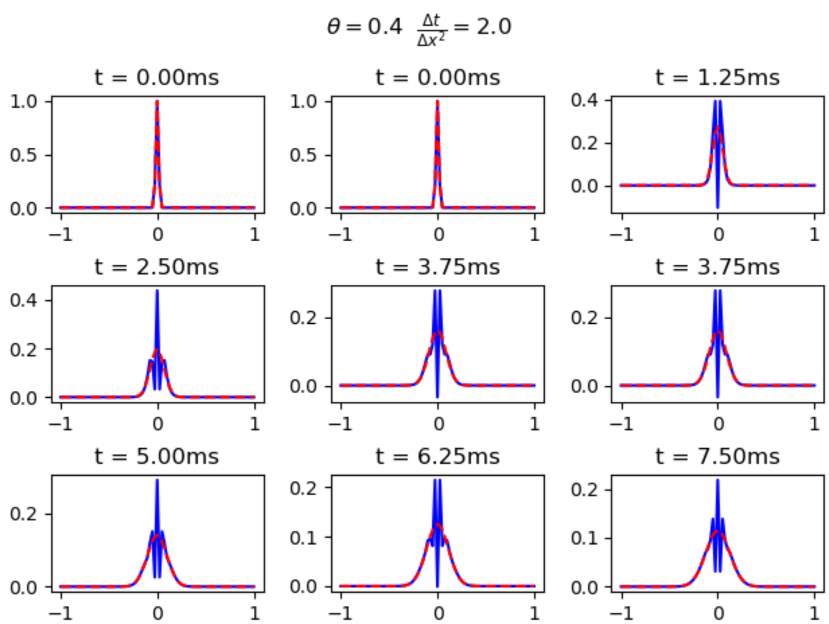

Figure 21: Solution for $\mu = 2.0$ and $\theta = 0.4$ (Unstable)

5 Code

```
import numpy as np
import matplotlib.pyplot as plt

#Following is the code for theta method to solve the heat equation

#Initialization function
def init_f(x, t=0):
    return np.sqrt(1e-4/(t+1e-4))*np.exp(-x**2/(4*(t+1e-4)))

#Function to construct matrix for 1 side (see report)
def construct_matrix(mu, theta, size):
    l = mu*theta
    mat = np.zeros([size,size])
    mat[np.arange(size),np.arange(size)] = 1+2*l
    mat[np.arange(size-1)+1,np.arange(size-1)] = -1*l
    mat[np.arange(size-1),np.arange(size-1)+1] = -1*l
    return mat

#Function to create main 'A' matrix'
def construct_mat(mu,theta,size):
    m1 = construct_matrix(mu,theta,size)
    m2 = construct_matrix(mu,theta-1,size)
    return np.linalg.inv(m1).dot(m2)

#Main theta method code
def theta_method(theta, mu, num_points, tsteps, init_fn = init_f):
    dx = 2.0/(num_points-1)
    dt = mu*(dx**2)
    x  = np.linspace(-1,1,num_points)
    t  = np.arange(tsteps)*dt
    X,T = np.meshgrid(x,t)
    u_0  = init_f(x)

    U  = np.zeros([num_points,tsteps])

    U[:,0] = u_0
    mat = construct_mat(mu, theta, num_points-2)

    for i in range(tsteps-1):
        U[1:-1,i+1] = mat.dot(U[1:-1,i])

    return U.T, X, T

#Plot convergence plots for a given theta
def convergence_study(theta):
    num_points = 41
    mu = np.linspace(0.01,0.61,301)
    dx = 2.0/(num_points-1)
    dt = mu*(dx**2)
    errors = np.zeros_like(mu)
    num_steps = (0.01/(mu*dx**2)).astype(int)

    for i,m in enumerate(mu):
        U, X, T      = theta_method(theta, m, num_points,num_steps[i])
        err          = U[-1,:] - init_f(X[0,:],num_steps[i]*dt[i])
        errors[i]    = np.sqrt(np.sum(err**2))

    fig, ax = plt.subplots()
    ax.set_title(r'Convergence plot for $\theta = {}$'.format(theta)+'\n\n\n')
    ax.loglog(mu, errors,'b')
    ax.plot([0.5,0.5],[np.min(errors),np.max(errors)],'r--')
    ax.set_xlabel(r'$\mu$')
    ax.set_ylabel('Root mean squared error after 10ms')
    new_t = [xt*(dx**2) for xt in ax.get_xticks()]
    xmin,xmax = ax.get_xlim()
```

```
    ax2 = ax.twiny()
    ax2.set_xlim([xmin*(dx**2), xmax*(dx**2)])
    ax2.loglog(dt, errors,'b')
    ax2.set_xlabel('$dt$')
    plt.tight_layout()
    plt.savefig('theta{:.0f}.png'.format(theta*10))

#Eigenval stability analysis for various mu and theta values
def stability_analysis():
    mu = np.linspace(0,10,100)
    theta = np.linspace(0,1,100)
    mu,theta = np.meshgrid(mu,theta)
    eigval = np.zeros_like(mu)
    for m,t,e in zip(np.nditer(mu), np.nditer(theta), np.nditer(eigval, op_flags=['readwrite'])):
            e[...] = np.max(abs(np.linalg.eigvals(construct_mat(m,t,5))))

    eigval = (eigval>1)+0
    fig, ax = plt.subplots()
    cs = ax.contourf(mu,theta,eigval,1)
    proxy = [plt.Rectangle((0,0),1,1,fc = pc.get_facecolor()[0])
                for pc in cs.collections]

    plt.legend(proxy, ["Stable (Max eigen value < 1)", "Unstable (Max eigen value > 1)"])

    ax.plot([10,0],[0.5,0.5],'r--')
    ax.plot([0.5,0.5],[1,0],'r--')
    ax.set_xlabel(r'$\mu \left( = \frac{\Delta t}{\Delta x^2}\right)$')
    ax.set_ylabel(r'$\theta$')
    ax.set_title(r"Stability plot for $\theta$ method")
    plt.savefig('eig.png')
    plt.close()

#Code to generate plots for 1 instance of the theta method
def plot_instance(theta, mu):
    num_points = 81
    dx = 2.0/(num_points-1)
    dt = mu*(dx**2)
    numsteps = int(0.01/(mu*dx**2))
    U, X, T      = theta_method(theta, mu, num_points,numsteps)
    fig, axs = plt.subplots(3,3)
    fig.suptitle(r'$\theta = {}$'.format(theta)+'\t'+r'$\frac{\Delta t}{\Delta x^2} ='+ '{}$'.format(mu))

    for i,ax in enumerate(axs.flatten()):
        ax.plot(X[0,:],U[i*numsteps/9],'b-')
        ax.plot(X[0,:],init_f(X[0,:],T[i*numsteps/9,0]),'r--')
        ax.set_title("t = {:.2f}ms".format(T[i*numsteps/9,0]*1000))

    plt.tight_layout()
    fig.subplots_adjust(top=0.85)
    plt.savefig('mu{:.0f}theta{:.0f}.png'.format(mu*10.0,theta*10.0))
    plt.close()
    plt.figure()
    plt.contourf(X,T,U,200)
    plt.savefig('c_mu{:.0f}theta{:.0f}.png'.format(mu*10.0,theta*10.0))
    plt.close()

if __name__=='__main__':
    stability_analysis()
    for theta in [0,0.4,0.5,0.6,1.0]:
        convergence_study(theta)
    for mu in [0.25,0.6,2.0]:
        for theta in [0, 0.4, 0.5, 0.6, 1.0]:
            plot_instance(theta,mu)
```

6 Conclusion

We have successfully checked the convergence and stability of the θ method for the heat equation for various values of μ and θ. We have shown that the sceme is stable for large time steps when $\theta \geq 0.5$ and is conditionally stable otherwise.