实验报告

----------陈赞华

1 使用的python库

1. numpy
2. matplotlib
3. scipy
4. math

2 Euler 法和改进欧拉法

1. 实验题目

实习题 1：

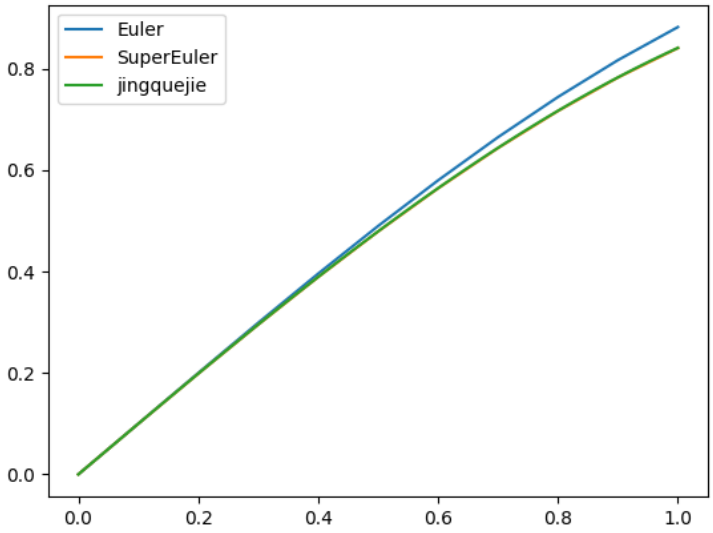
将 $u'' = -u(0 \le t \le 1),\ u(0) = 0,\ u'(0) = 1$ 化成一阶方程组，并用 Euler 法和改进的 Euler 法求解，步长 h = 0.1, 0.05。

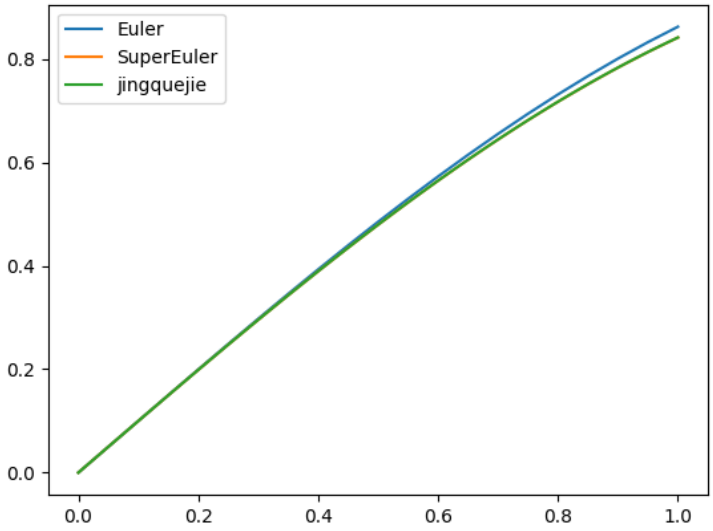
(1) 画出 Euler 法，Euler 改进法以及精确解的数值结果图形；

(2) 列表对比 Euler 法，Euler 改进法以及精确解，判断哪种方法的精度更高。

2. 实验结果

(1) 步长h=0.1 h = 0.05





3. 代码实现

```
# import module
import numpy as np
import matplotlib
from math import *
matplotlib.rcParams['axes.unicode_minus'] =False
import matplotlib.pyplot as plt


def jingquejie(a,b):
    '''
    input :
    a: 区间左端点
    b: 区间右端点
```

```
    output :

    '''
    x=np.linspace(a,b,100)
    y=np.sin(x)
    plt.plot(x,y)


#Euler法
def Euler(a,b,Mo,Uo,h):
    '''
    input:
    a,b :区间左右端点
    Mo,Uo: 初值
    h: 步长
    '''
    #节点个数
    num=int((b-a)/h+1)
    U=np.zeros((num,))
    M=np.zeros((num,))
    U[0]=Uo
    M[0]=Mo
    for n in range (num-1):
        if n>=1:
           M[n]=M[n-1]-h*U[n-1]
        U[n+1]=U[n]+h*M[n]

    x=np.linspace(a,b,num)
    plt.plot(x,U)


#改进欧拉法
def SuperEuler(a,b,Mo,Uo,h):
    '''
    input :
    a,b:区间左右端点
    Mo,Uo: 初值
    h:步长
    '''
    num=int((b-a)/h+1)
    U=np.zeros((num,))
    M=np.zeros((num,))
    U[0]=Uo
    M[0]=Mo
    L=np.zeros((num,2))
    L[0,0]=M[0]
    L[0,1]=U[0]
    A=np.array([[2,h],[-h,2]])
```

```
    #L[n+1,:]=[M[n+1],U[n+1]]
    #A*L[n+1,:].T=C*L[n,:].T
    C=A.T
    for n in range (num-1):
        L[n+1,:]=np.linalg.solve(A, np.dot(C,L[n,:].T)).T
    U=L[:,1]
    x=np.linspace(a,b,num)
    plt.plot(x,U)


def main():
    Mo=1
    Uo=0
    h=0.05
    a=0
    b=1
    Euler(a,b,Mo,Uo,h)
    SuperEuler(a,b,Mo,Uo,h)
    jingquejie(a,b)
    plt.legend(["Euler","SuperEuler","jingquejie"])
    plt.show()

main()
```

4. 算法比较

h = 0.05时

Euler法	精确解	改进Euler法
0	0	0
0.05	0.04996877	0.04997917
0.1	0.0998127	0.09983342
0.149875	0.14940724	0.14943813
0.1995	0.19862851	0.19866933
0.24875031	0.24735351	0.24740396
0.29750188	0.29546052	0.29552021
0.34563156	0.34282934	0.34289781
0.39301749	0.38934161	0.38941834
0.43953935	0.43488115	0.43496553
0.48507866	0.47933416	0.47942554
0.52951912	0.52258958	0.52268723
0.57274688	0.56453934	0.56464247
0.61465085	0.60507864	0.60518641
0.65512295	0.64410618	0.64421769
0.69405842	0.68152447	0.68163876
0.73135608	0.71724001	0.71735609
0.7669186	0.75116357	0.75128041
0.80065273	0.78321039	0.78332691
0.83246956	0.81330041	0.8134155
0.86228476	0.84135845	0.84147098

3 龙格库塔法

1. 实验题目

实验题 2

用四级四阶 Runge Kutta 法计算初值问题：

$$u' = 4tu^{\frac{1}{2}}, \qquad 0 \le t \le 2,$$

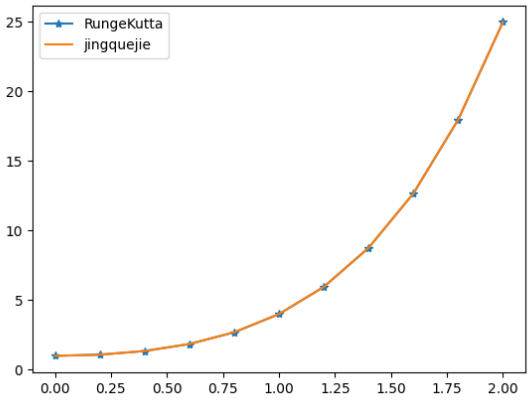
$$u(0)=1,$$

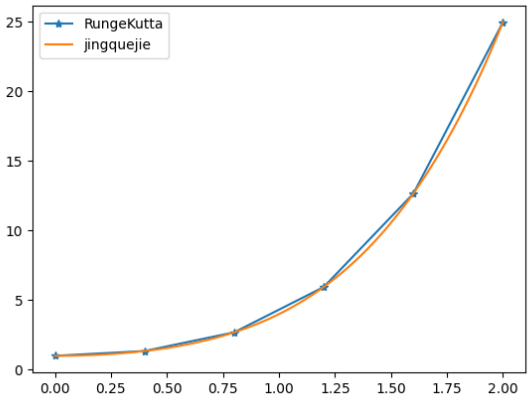
取 h=0.2，0.4，0.5.

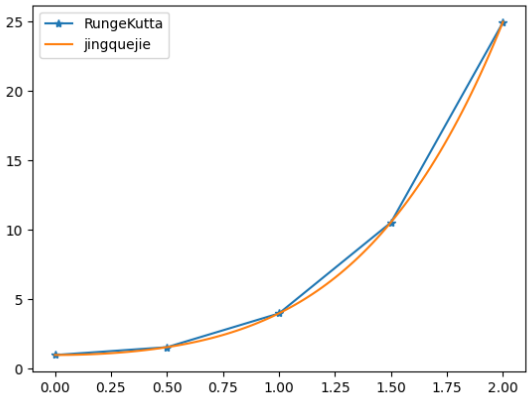
精确解为 $u(t) = (1+t^2)^2$

2. 实验结果

(1)　h = 0.2　h =0.4　h = 0.5







3. 代码实现

```
# import module
import numpy as np
import matplotlib
from math import *
```

```
matplotlib.rcParams['axes.unicode_minus'] =False
import matplotlib.pyplot as plt

def function(t,u):
    '''
    inpu
    t: 微分方程中的变量
    u: 微分方程的变量
    '''
    du=4*t*sqrt(u)
    return du
def CALk(t,Un,h):
    '''
    input :
    t:
    Un: U在t(n)处的取值
    h: 步长
    '''
    K=np.zeros((4,))
    K[0]=function(t,Un)
    K[1]=function(t+h/2,Un+h/2*K[0])
    K[2]=function(t+h/2,Un+h/2*K[1])
    K[3]=function(t+h,Un+h*K[2])
    return K

def RungeKutta(a,b,h,Uo):
    '''
    input
    a,b: 区间的做右端点
    h: 步长
    Uo: U的初值
    '''
    num=int((b-a)/h)+1
    U=np.zeros((num,))
    U[0]=Uo
    for n in range(num-1):
        t=a+n*h
        K=CALk(t,U[n],h)
        U[n+1]=U[n]+h/6*(K[0]+2*K[1]+2*K[2]+K[3])
    x=np.linspace(a,b,num)
    plt.plot(x,U,'-*')


def jingquejie(a,b):
    '''
    input
    a,b:区间做右端点
    '''
```

```
    x=np.linspace(a,b,100)
    y=(1+x*x)*(1+x*x)
    plt.plot(x,y)

def main():
    a=0
    b=2
    h=0.5
    Uo=1
    RungeKutta(a,b,h,Uo)
    jingquejie(a,b)
    plt.legend(["RungeKutta","jingquejie"])
    plt.show()

main()
```

4 五点差分

1. 实验题目

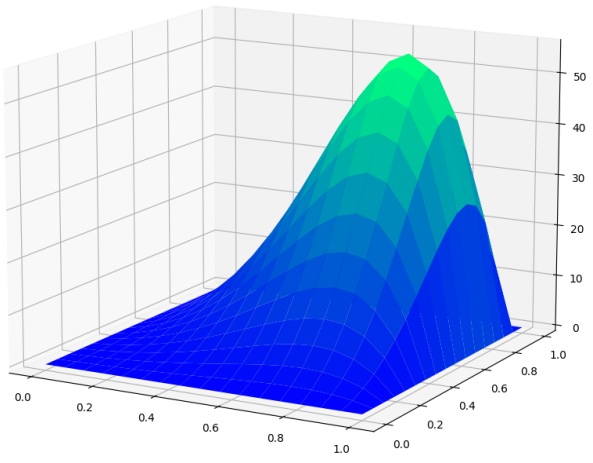
实习题 3：(P104)
求解边值问题

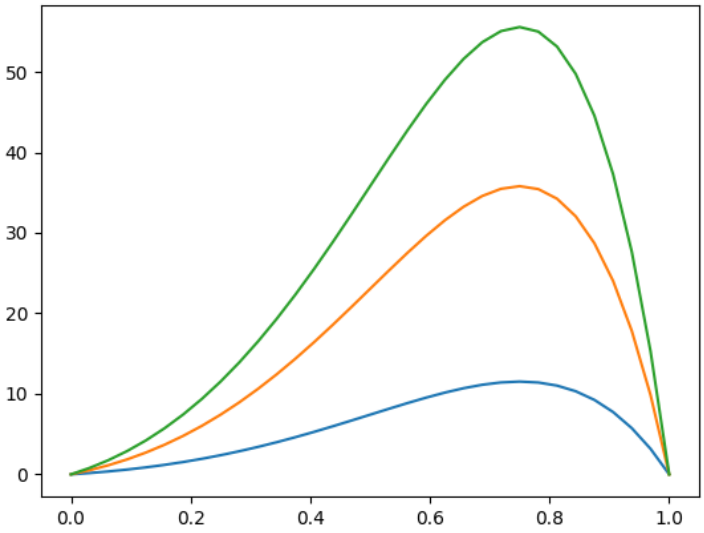
$$\Delta u = 2\pi^2 e^{\pi(x+y)}(\sin \pi x \cos \pi y + \cos \pi x \sin \pi y), \qquad (x,y) \in G = (0,1)\times(0,1)$$

$$u = 0,, \quad (x,y) \in \partial G$$

(1) 取步长 h=k=1/16；1/32；1/64 进行数值求解，并列表给出不同网格中(1/4,1/4)，(1/2,1/2)，(3/4,3/4)点处的数值解和解析解；
(2) 就 h=1/32，画出 y=1/4；1/2 和 3/4 及 i=0,1,2,…,32 时的差分解曲线。

2. 实验结果





3. 代码实现

```
from math import *
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D #3d模块
def calculate(h,N,x,y):
    '''
    input :
    h : 步长
    N : 区间等分数
    x,y: 网格节点向量
    output:
    A: 系数矩阵
    U:  网格矩阵
    B:  微分方程中右端项向量
    '''
    A=np.zeros((N*N,N*N))
    U=np.zeros((N*N,1))
    B=np.zeros((N*N,1))
    #先处理A
    ## 处理边界条件。（上下）
    e=np.eye(N*N)
    A[0:N,:]=e[0:N,:]
    A[N*N-N:N*N,:]=e[N*N-N:N*N,:]
    #B[0:N,0]=np.zeros((N,))
    #B[N*N-N:N*N,0]=np.zeros((N,))
    ## 处理左右边界上的点
    for i in range (1,N):
        A[i*N,:]=e[i*N,:]
       # B[i*N,0]=0
        A[(i+1)*N-1,:]=e[(i+1)*N-1,:]
```

```
        # B[(i+1)*N-1,:]=0
    ##处理内点
    for j in range (N,N*N-N,1):
        if not j%N==0 and  not (j+1)%N==0:
            A[j][j]=-4
            A[j][j-1]=1
            A[j][j+1]=1
            A[j][j-N]=1
            A[j][j+N]=1
            xx=x[j//N] #取商
            yy=y[j%N]  #取余
            B[j,0]=h*h*(2*pi*pi)*exp(pi*(xx+yy))*
(sin(pi*xx)*cos(pi*yy)+cos(pi*xx)*sin(pi*yy))



    return A,U,B
def initialize():
    '''
    input:

    output:
    h: 步长
    N:  区间等分数
    k:  网格点坐标值向量（题目中要求求解的网格点坐标）
    x,y:网格节点向量
    '''
    a=b=0
    c=d=1
    h=1/32
    N=int((c-a)/h+1)
    x=[a+h*i for i in range(N)]
    y=[b+h*j for j in range(N)]
    k=[1/4,1/2,3/4]
    return h,N,k,x,y



def shuzhi(U,k,h):
    '''
    imput :
    U:  网格矩阵
    k:  网格点坐标值向量（题目中要求求解的网格点坐标）
    h:  步长
    '''
    xy1,xy2,xy3=int(k[0]/h),int(k[1]/h),int(k[2]/h)
    a1=U[xy1][xy1]
    a2=U[xy2][xy2]
```

```
    a3=U[xy3][xy3]
    print(f"a1:{a1}  a2: {a2} a3: {a3}")

def precise(k):
    '''
    input :
    k:网格点坐标值向量（题目中要求求解的网格点坐标）
    '''
    res=[]
    for x in k:
        y=x
        prec=exp(pi*(x+y))*sin(pi*x)*sin(pi*y)
        res.append(prec)
    print(f"{res[0]}   {res[1]}   {res[2]}")


def plotdata(x,y,U):
    '''
    input :
    x,y :网格节点
    U:  网格点矩阵
    '''
    fig=plt.figure()
    ax=Axes3D(fig)
    [X,Y]=np.meshgrid(x,y)
    # TODO 查清楚x, y 的位置
     # TODO 将精确解的图像话出来比较一下。
    ax.plot_surface(X,Y,U,cmap=plt.cm.winter)
    plt.show()


def main():
    #初始化
    h,N,k,x,y=initialize()
    #计算系数矩阵，和右端项
    A,U,B=calculate(h,N,x,y)

 #np.savetxt(r'C:\Users\c2752\Desktop\resources\weifenfangchengcode\wud
ianchafen\A.txt',A,fmt="%d")
    U=np.dot(np.linalg.inv(A),B)
    U_M=U.reshape(N,N)

 #np.savetxt(r'C:\Users\c2752\Desktop\resources\weifenfangchengcode\wud
ianchafen\U.txt',U)

 #np.savetxt(r'C:\Users\c2752\Desktop\resources\weifenfangchengcode\wud
ianchafen\U_M.txt',U_M)
    #print(U)
```

```
        #数值解
        shuzhi(U_M,k,h)
        #精确解
        precise(k)
        #画图
        plotdata(x,y,U_M)

main()
```

(1) 数据展示

取值	数值解(h=1/16)	数值解(h=1/32)	数值解(1/64)	精确解
(1/4,1/4)	2.2905368916404747	2.3765261471204164`	2.398058271014387	2.405238690482676
(1/2,1/2)	22.757421506639552	23.04493518121395	23.11675719574881	23.140692632779267
(3/4,3/4)	55.27675662281694	55.56368483155225	55.635108949328625	55.65888924492812

5 抛物方程的向后格式

1. 实验题目

实习期 4：

求解一维抛物方程的初边值问题：

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}+\sin t,\quad 0<x<1,\quad t>0$$
$$u_x(0,t)=u_x(1,t)=0,\quad t>0$$
$$u(x,0)=\cos \pi x,\quad 0<x<1$$

(精确解为：$u=e^{-\pi^2 t}\cos \pi x+(1-\cos t)$)

分别采用向后以及六点对称格式，通过以下两种方案：

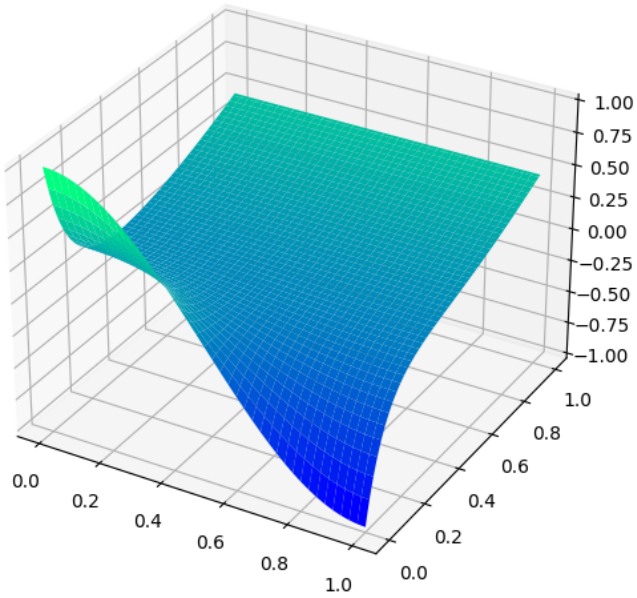
(1) h=1/40, τ= 1/1600；(2) h=1/80, τ= 1/3200

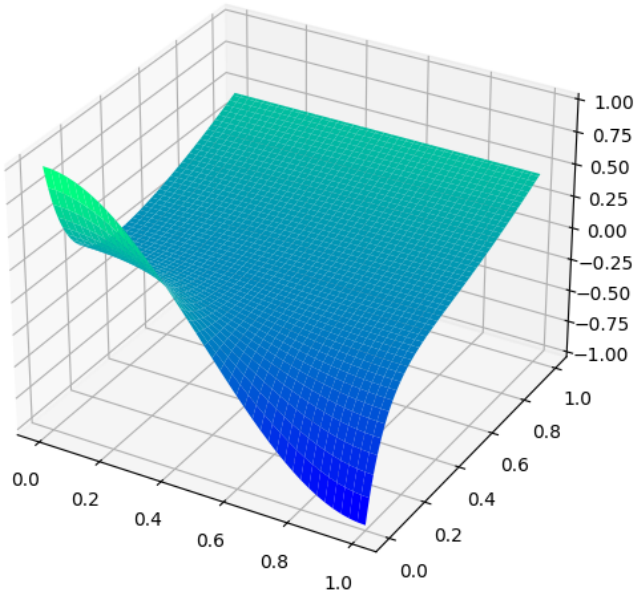
计算到时间 t=1。

并列表给出该时刻，两种数值方法以及解析解在 x=1/5,2/5,3/5,4/5,1处的值。

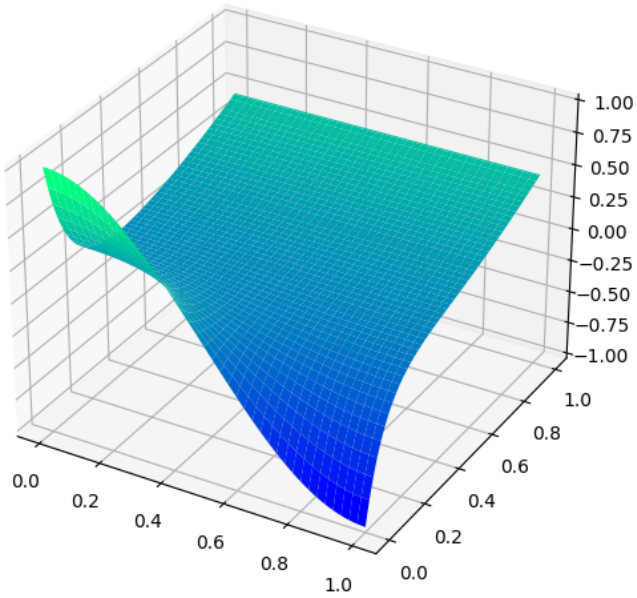
2. 实验结果

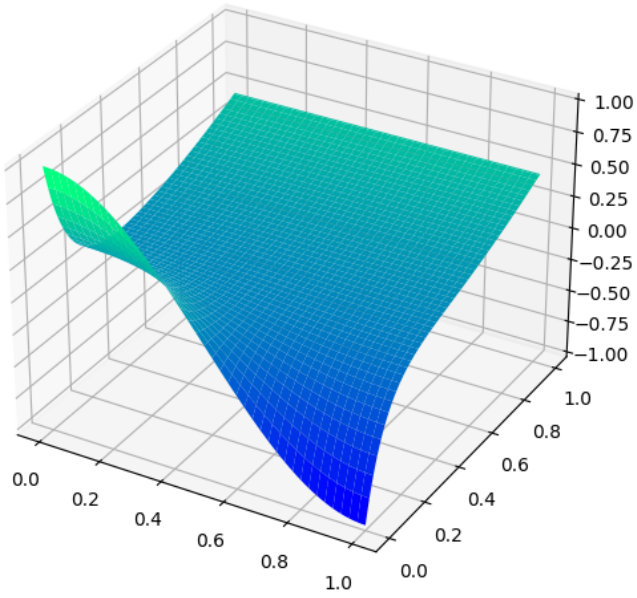
r = 1 数值解 精确解


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

r = 2数値解　精確解





3. 代码实现

```
import numpy as np
from math import *
from scipy import sparse
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D #3d模块
def init():
    '''
```

```
    input:
    output:
    A: 系数矩阵
    U: 网格矩阵
    x,t: 网格节点 x为空间层,t 为时间层
    h,tao: 空间和时间上的步长
    J:  空间上的区间等分数
    N:  时间上的区间等分数
    '''
    x_0, x_J = 0, 1
    t_0, t_N =0, 1
    #时间和空间上的步长
    h, tao =1/80, 1/3200
    r= ceil((1/(h*h))/(1/tao))
    #列数
    J=int((x_J-x_0)/h)
    #行数
    N=int((t_N-t_0)/tao)

    x=[x_0+j*h  for j in range(J+1)]
    t=[t_0+n*tao for n in range(N+1)]
    # U
    U = np. zeros((N+1,J+1))
    U[0,:] = np.cos((np.multiply(pi,x)))

    # 设计A矩阵
    v1 = [1+2*r]* (J+1)
    v2 = [-r]*(J+1)
    matrix = np.array([v2,v1,v2])
    #三对角矩阵
    A = sparse.spdiags(matrix,[-1,0,1],J+1,J+1).T.A
    A[0][1], A[J][J-1] = -2*r, -2*r
    #TODO 这里的A[J][J-1]中的下标有没有错误
    return A,U,x,t,h,tao,J,N
def F(tao,t,J):
    '''
    input:
    tao: 时间上的步长
    t:   某一时刻
    J:    空间上的等分数
    output:
    B:   抛物方程中的右端项向量
    '''
    T = np.array([t]*(J+1))
    B = np.multiply(tao,np.sin(T))
    return B

def backforward(A,U,T,tao,J,N):
```

```
    '''
    input :
    A: 系数矩阵
    U:  网格矩阵
    T:  时间网格节点
    tao: 时间上的步长
    J:  空间上的区间等分数
    N:  时间上的区间等分数
    output:
    U: 网格矩阵
    '''
    Un = np.array(U[0,:]).T
    A_inv = np.linalg.inv(A)

    for i in range(N):
        B = F(tao,T[i+1],J)
        Un_1 = np.dot(A_inv,Un+B)
        U[i+1,:] = Un_1.T.tolist()
        Un=np.array(U[i+1,:]).T

    return U
   #
np.savetxt(r"C:\Users\c2752\Desktop\resources\weifenfangchengcode\paowu
fangcheng\U.txt",U)


def jingquejie(x,t):
    '''
    input:
    x,t: 网格节点
    output:
    U : 精确解矩阵
    '''
    fig = plt.figure()
    ax = Axes3D(fig, auto_add_to_figure =False)
    fig.add_axes(ax)
    [X,Y] = np.meshgrid(x,t)
    U=np.exp(-pi*pi*Y)*np.cos(pi*X)+(1-np.cos(Y))
    ax.plot_surface(X,Y,U,cmap=plt.cm.winter)
    plt.show()
    return U




def plotdata(x,t,U):
    '''
    input:
```

```
    x,t : 网格节点
    U : 网格矩阵
    '''
    fig = plt.figure()
    ax = Axes3D(fig, auto_add_to_figure =False)
    fig.add_axes(ax)
    [X,Y] = np.meshgrid(x,t)
    # TODO 查清楚x, t 的位置关系。
    # TODO 将精确解的图像话出来比较一下。
    ax.plot_surface(X,Y,U,cmap=plt.cm.winter)
    plt.show()

# def collectdata(U,U1):
#     x = [16*i for i in range(1,6,1)]
#     a = []
#     b = []
#     for i in x:
#         a.append(U[-1,i])
#         b.append(U1[-1,i])
#     print(a,"\n",b)

def main():
    A,U,x,t,h,tao,J,N =init()

    U=backforward(A,U,t,tao,J,N)
    print(type(U))
    plotdata(x,t,U)
    U1 = jingquejie(x,t)
    #collectdata(U,U1)


main()
```

(1) 数据

r=1	0.2	0.4	0.6	0.8	1
数值解	0.46000398976485324	0.45997719742633003	0.45994408027463757	0.4599172879361126	0.4599070541734343
精确解	0.45973953906850235	0.45971367747540043	0.45968171078832004	0.4596558491952181	0.4596459709456564
r=2	0.2	0.4	0.6	0.8	1
数值解	0.4598717093526007	0.4598454187303609	0.459812921734098	0.45978663111185136	0.4597765889877391
精确解	0.45973953906850235	0.45971367747540043	0.45968171078832004	0.4596558491952181	0.4596459709456564

6　抛物方程的六点对称格式

1. 题目

实习期 4：

求解一维抛物方程的初边值问题：

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}+\sin t,\quad 0<x<1,\quad t>0$$
$$u_x(0,t)=u_x(1,t)=0,\quad t>0$$
$$u(x,0)=\cos \pi x,\quad 0<x<1$$

(精确解为：$u=e^{-\pi^2 t}\cos \pi x+(1-\cos t)$)

分别采用向后以及六点对称格式，通过以下两种方案：

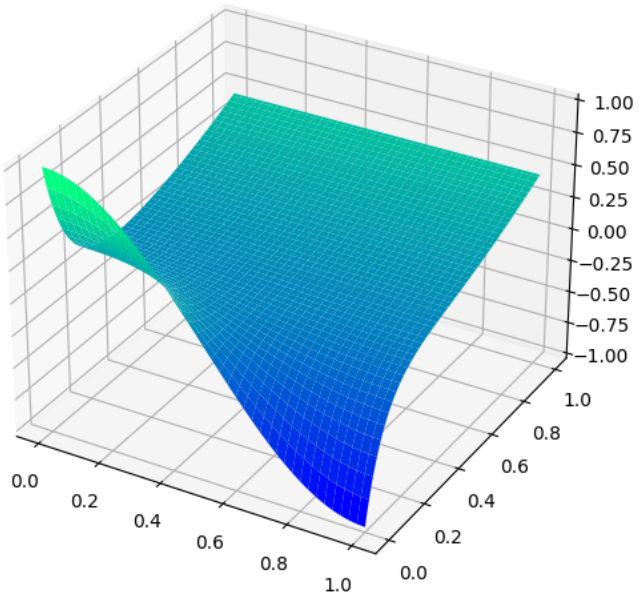
(1) h=1/40, $\tau=1/1600$；(2) h=1/80, $\tau=1/3200$

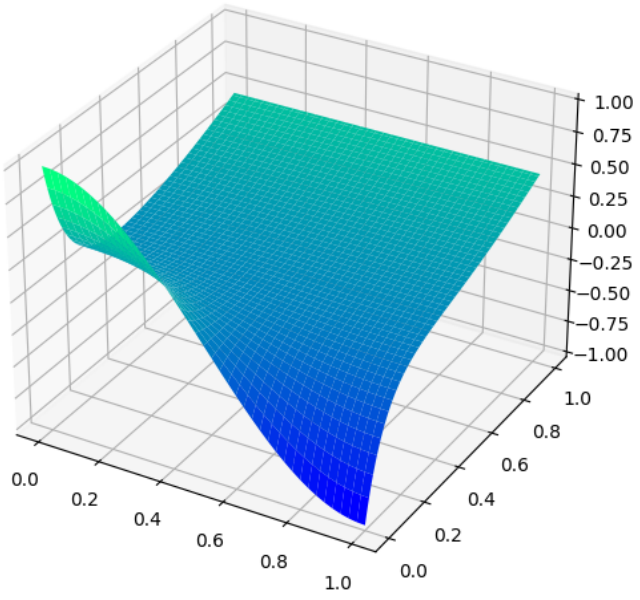
计算到时间 t=1。

并列表给出该时刻，两种数值方法以及解析解在 x=1/5,2/5,3/5,4/5,1处的值。

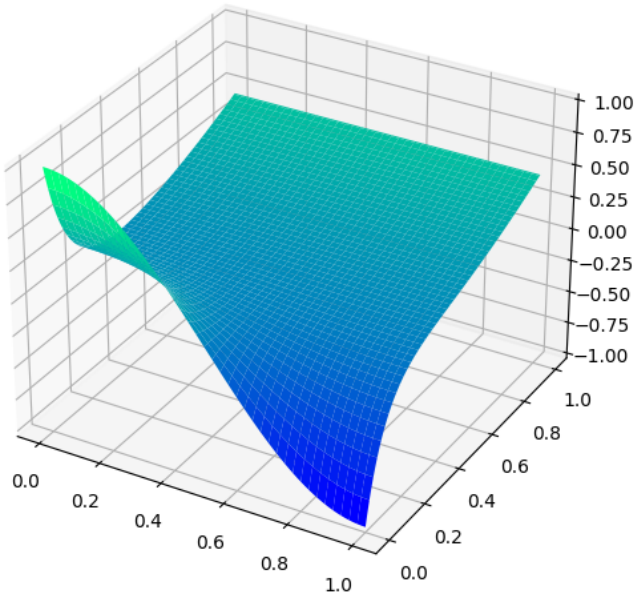
(1)　实验结果

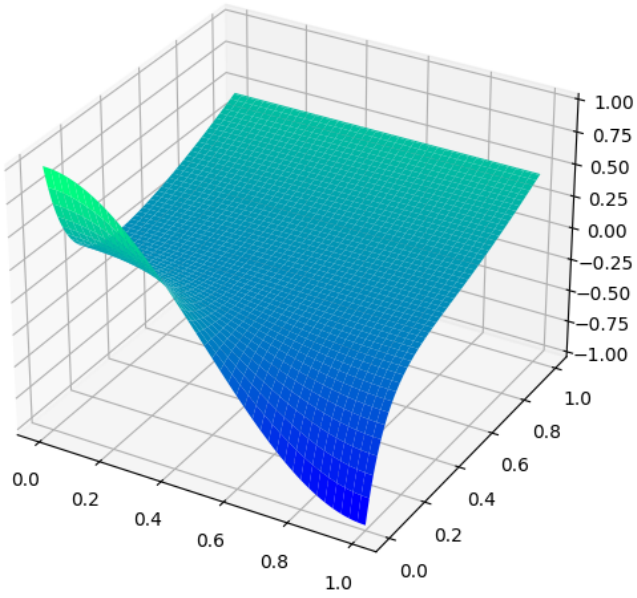
r ＝ 1 数值解　精确解


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

r=2 数値解　　精確解





3. 代码实现

```
import numpy as np
from math import *
from scipy import sparse
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D #3d模块

def init():
    '''
    input:
    output:
    A, B: 系数矩阵
    U: 网格矩阵
    x,t: 网格节点 x为空间层,t 为时间层
    h,tao: 空间和时间上的步长
```

```
    J:  空间上的区间等分数
    N:  时间上的区间等分数
    '''
    x_0, x_J = 0, 1
    t_0, t_N =0, 1
    #时间和空间上的步长
    h, tao =1/40, 1/1600
    r= ceil((1/(h*h))/(1/tao))
    #列数
    J=int((x_J-x_0)/h)
    #行数
    N=int((t_N-t_0)/tao)

    x=[x_0+j*h  for j in range(J+1)]
    t=[t_0+n*tao for n in range(N+1)]
    # U
    U = np. zeros((N+1,J+1))
    U[0,:] = np.cos((np.multiply(pi,x)))

    # 设计A矩阵
    v1 = [1/tao+1/(h*h)]* (J+1)
    v2 = [-1/(2*h*h)]*(J+1)
    matrix = np.array([v2,v1,v2])
    #三对角矩阵
    A = sparse.spdiags(matrix,[-1,0,1],J+1,J+1).T.A
    A[0][1] = -1/(h*h)
    A[J][J-1]= -1/(h*h)
    # 设计B矩阵
    # 设计A矩阵
    v1 = [1/tao-1/(h*h)]* (J+1)
    v2 = [1/(2*h*h)]*(J+1)
    matrix = np.array([v2,v1,v2])
    #三对角矩阵
    B = sparse.spdiags(matrix,[-1,0,1],J+1,J+1).T.A
    B[0][1], B[J][J-1]= 1/(h*h), 1/(h*h)

 #np.savetxt(r"C:\Users\c2752\Desktop\resources\weifenfangchengcode\pao
wufangcheng\A.txt",A)

 #np.savetxt(r"C:\Users\c2752\Desktop\resources\weifenfangchengcode\pao
wufangcheng\B.txt",B)

    return A,B,U,x,t,h,tao,(J,N)


def jingquejie(x,t):
    '''
    input:
```

```
    x,t: 网格节点
    output:
    U : 精确解矩阵
    '''
    fig = plt.figure()
    ax = Axes3D(fig, auto_add_to_figure =False)
    fig.add_axes(ax)
    [X,Y] = np.meshgrid(x,t)
    U=np.exp(-pi*pi*Y)*np.cos(pi*X)+(1-np.cos(Y))
    ax.plot_surface(X,Y,U,cmap=plt.cm.winter)
    plt.show()
    return U

def plotdata(x,t,U):
    '''
    input:
    x,t : 网格节点
    U : 网格矩阵
    '''
    fig = plt.figure()
    ax = Axes3D(fig, auto_add_to_figure =False)
    fig.add_axes(ax)
    [X,Y] = np.meshgrid(x,t)
    # TODO 查清楚x, t 的位置关系。
    # TODO 将精确解的图像话出来比较一下。
    ax.plot_surface(X,Y,U,cmap=plt.cm.winter)
    #plt.show()


def F(ti,ti1,J):
    '''
    input:
    ti,ti1 : 时刻t(i) t(i+1)
    J: 空间上区间的等分数
    output:
    b : 抛物方程中的右端项
    '''
    temp  = (sin(ti)+sin(ti1))/2
    b = np.array([temp]*(J+1))
    b = b.reshape(J+1,1)

    return b

def CrankNicolson(A,B,U,T,J,N):
    '''
    input :
    A,B: 系数矩阵
    U: 网格矩阵
```

```
    T:  网格节点（时间）
    J, N:  空间和时间上的区间的等分数
    output:
    U : 网格矩阵
    '''
    Un = np.array(U[0,:]).T# (J+1) by 1
    A_inv = np.linalg.inv(A) # (J+1) by (J+1)
    temp = np.zeros((J+1,1)) # (J+1) by 1
    for i in range(N):
        b = F(T[i],T[i+1],J) # (J+1) by 1
        temp = np.dot(B,Un).reshape((J+1),1) + b
        Un_1 = np.dot(A_inv,temp)
        temp1 = Un_1.T.reshape(-1,)
        U[i+1,:] = temp1
        Un=np.array(U[i+1,:]).T
    return U
   #
np.savetxt(r"C:\Users\c2752\Desktop\resources\weifenfangchengcode\paowu
fangcheng\U.txt",U)


# def collectdata(U,U1):
#     x = [16*i for i in range(1,6,1)]
#     a = []
#     b = []
#     for i in x:
#         a.append(U[-1,i])
#         b.append(U1[-1,i])
#     print(a,"\n",b)


def main():
    A, B, U, x, t, h, tao, D= init()
    U = CrankNicolson(A,B,U,t,D[0],D[1])
    plotdata(x,t,U)
    U1 = jingquejie (x,t)
    #collectdata(U,U1)

main()
```

(1)　数据

r=1	0.2	0.4	0.6	0.8	1
数值解	0.45973973558143955	0.459713743288304	0.45968161504709587	0.45965562275395916	0.4596456945814267
精确解	0.45973953906850235	0.45971367747540043	0.45968171078832004	0.4596558491952181	0.4596459709456564
r=2	0.2	0.4	0.6	0.8	1
数值解	0.45973958810451144	0.45971369389331646	0.4596816868880529	0.4596557926768566	0.4596459019682915
精确解	0.45973953906850235	0.45971367747540043	0.45968171078832004	0.4596558491952181	0.4596459709456564

7 迎风格式

(1) 实验题目

1. (实习题) 求解

$$
\begin{aligned}
&u_t - u_x = 0, \quad x \in (0,1),\\
&u(x,0) = \sin^{40} \pi x, \quad x \in [0,1],\\
&u(0,t) = u(1,t), \quad t \geqslant 0.\\
&(\text{精确解 } u(x,t) = \sin^{40} \pi(x+t))
\end{aligned}
$$

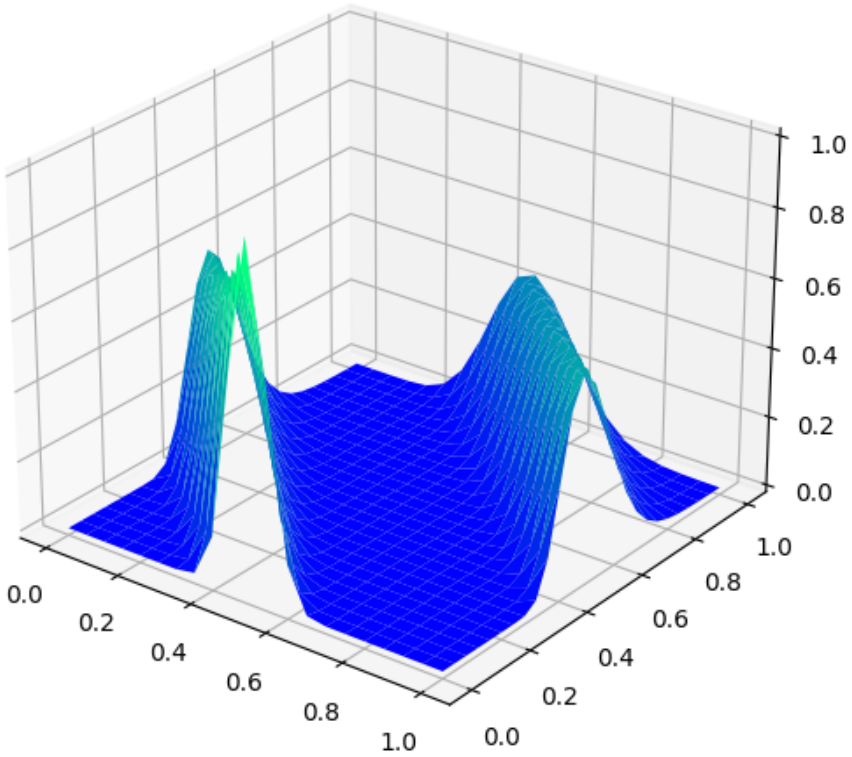
用迎风格式计算.

(a) 取 $h = 0.05, \tau = 0.04$, 算出 $t = 0.00,\ 0.12,\ 0.20,\ 0.80$ 的值.

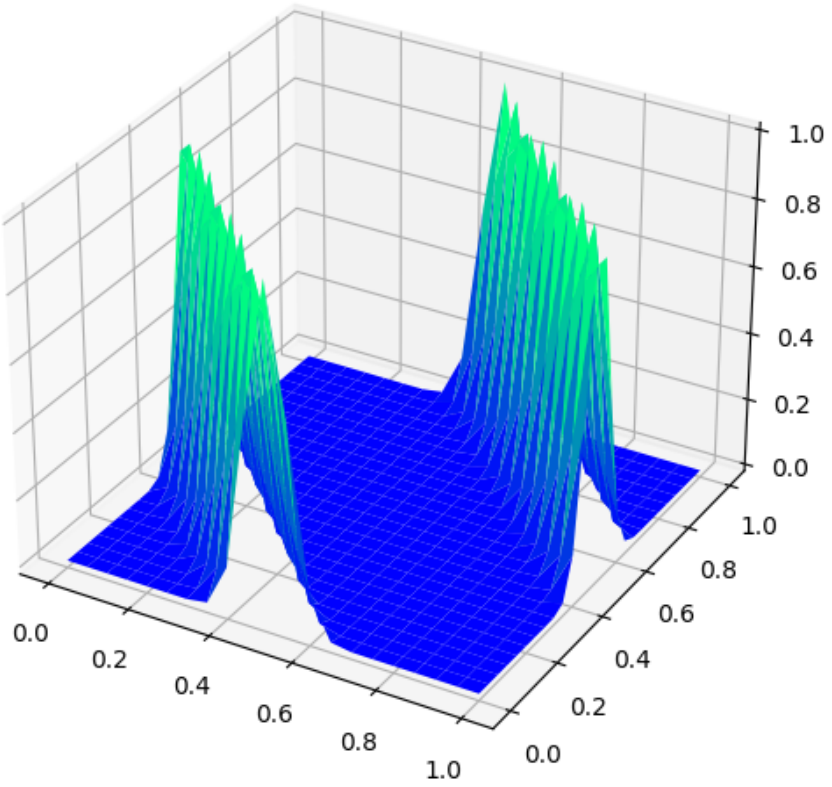
(b) 画出 (a) 的图形, 观察峰值位置的变化, 与精确解峰值位置比较.

(2) 实验结果

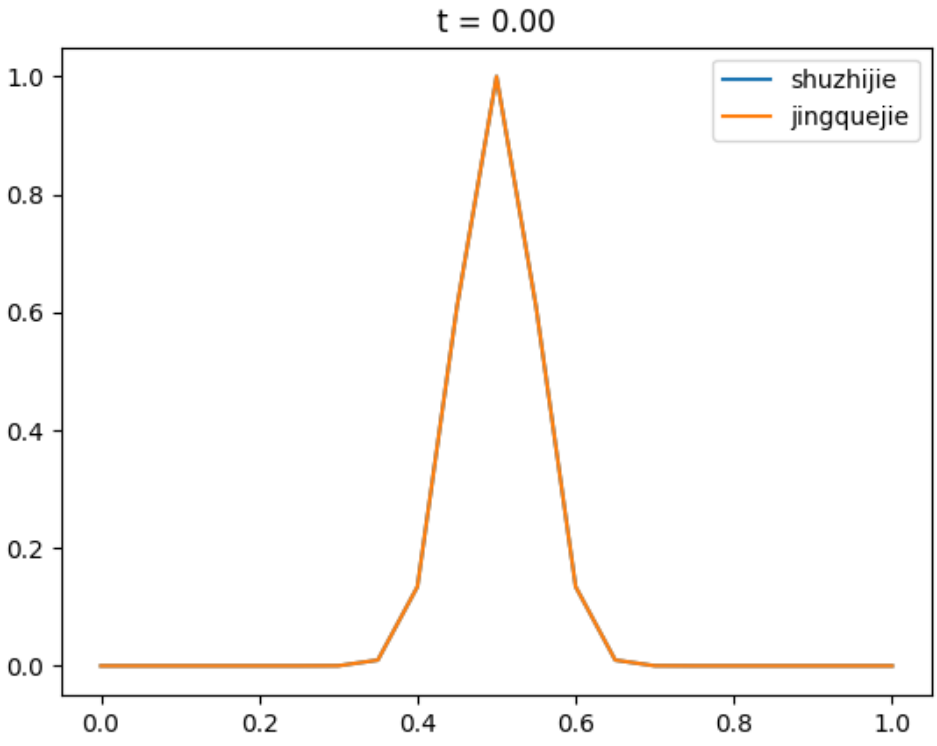
数值解:



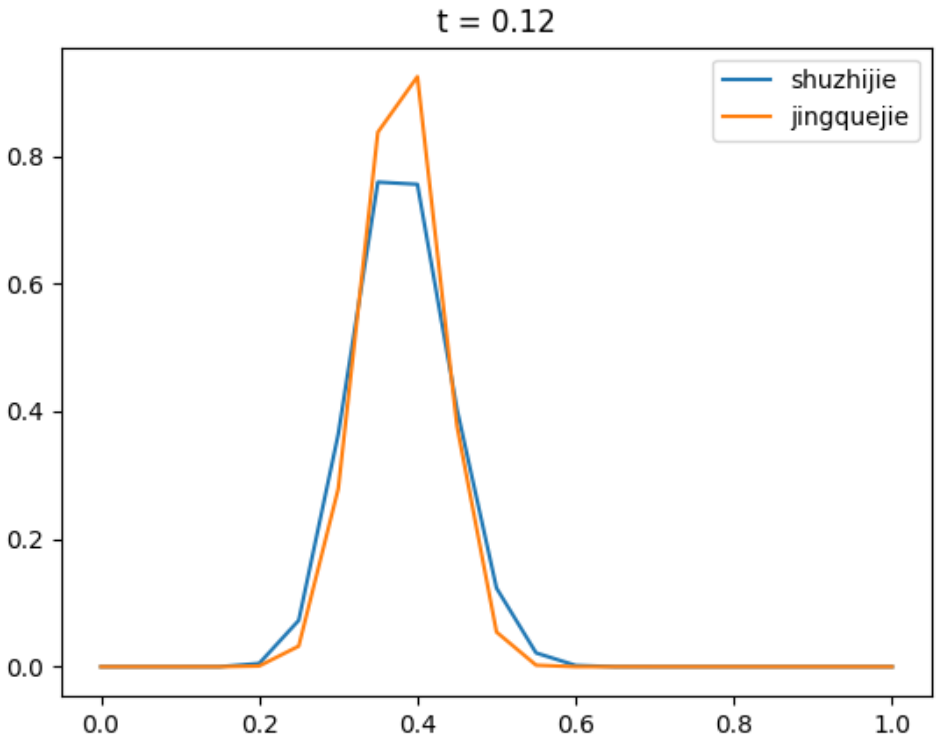
精确解:


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

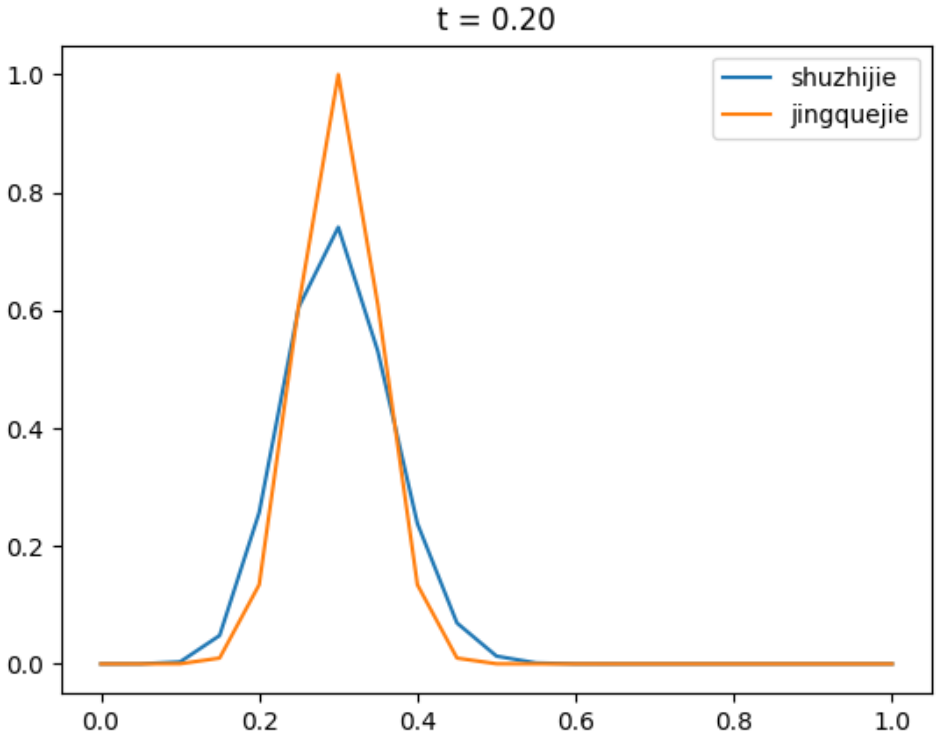
t = 0.00


t = 0.00
shuzhijie
jingquejie
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

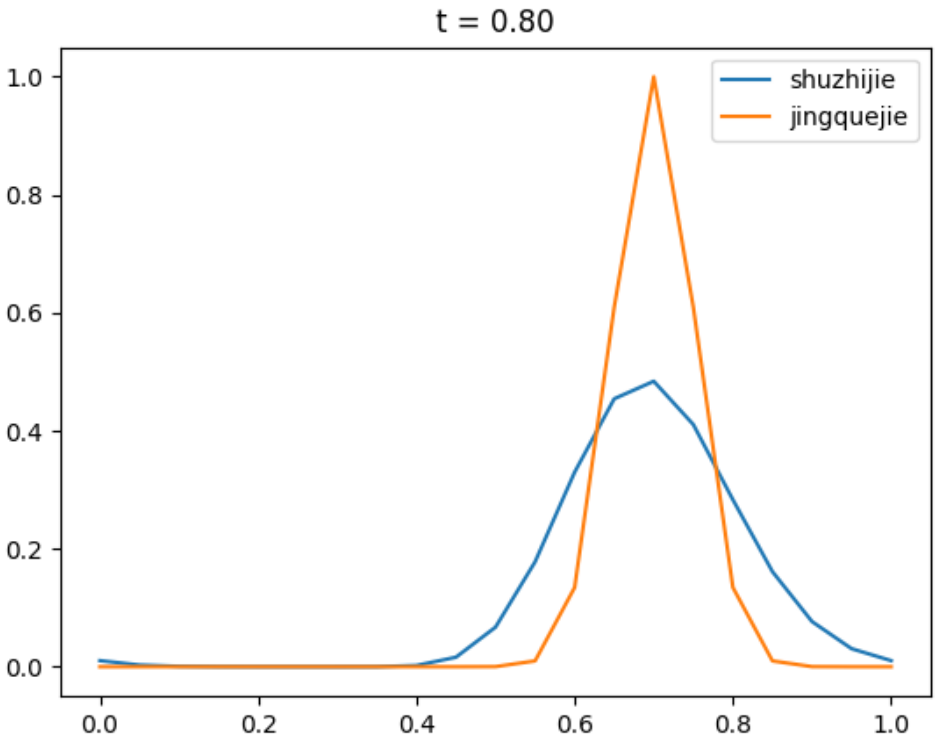
t = 0.12


t = 0.12
shuzhijie
jingquejie
0.0
0.2
0.4
0.6
0.8
0.0
0.2
0.4
0.6
0.8
1.0

t = 0.20


t = 0.20
shuzhijie
jingquejie
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

t = 0.80



(3) 代码实现

```
from math import *
from re import U
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D #3d模块
from scipy import sparse

def init():
    a ,b = 0, 1
    c ,d = 0, 1
    alpha = -1
    h ,tao = 0.05, 0.04
    J ,N = ceil((b-a)/h), ceil((d-c)/tao)
    x = np.linspace(a,b,J+1)
    t = np.linspace(c,d,N+1)
    U = np.zeros((N+1,J+1))
    temp = np.sin(np.multiply(pi,x))
    U[0,:] = np.power(temp, 40)
    r = alpha*tao/h


    v1 = [1+r]* (J)
    v2 = [-r]*(J)
    matrix = np.array([v2,v1])
    #三对角矩阵
```

```
    B = sparse.spdiags(matrix,[-1,0],J,J).T.A
    B[J-1][0] = -r

 #np.savetxt(r"C:\Users\c2752\Desktop\resources\weifenfangchengcod
e\yingfenggeshi\data1.txt",B)
    return B, U, J, N, x, t

def yingfeng(B,U,J,N):
    #print(U[0,:])
    Un = np.array(U[0,:-1]).T
    #print(Un)
    for i in range(N):
        Un1 = np.dot(B, Un)
        U[i+1,:-1] = Un1.T
        Un = Un1

    U[:,J] =U[:,0]

 #np.savetxt(r"C:\Users\c2752\Desktop\resources\weifenfangchengcod
e\yingfenggeshi\data.txt",U)

    return U

def plotdata(x,t,U):
    '''
    input:
    x,t : 网格节点
    U : 网格矩阵
    '''
    fig = plt.figure()
    ax = Axes3D(fig, auto_add_to_figure =False)
    fig.add_axes(ax)
    [X,Y] = np.meshgrid(x,t)

 np.savetxt(r"C:\Users\c2752\Desktop\resources\weifenfangchengcode
\yingfenggeshi\data2.txt",X)

 np.savetxt(r"C:\Users\c2752\Desktop\resources\weifenfangchengcode
\yingfenggeshi\data3.txt",Y)
    # TODO 查清楚x, t 的位置关系。
    # TODO 将精确解的图像话出来比较一下。
    ax.plot_surface(X,Y,U,cmap=plt.cm.winter)
    plt.title(["shuzhijie"])
    plt.show()

def precise(x,t):
    '''
    input:
```

```
    x,t : 网格节点
    U : 网格矩阵
    '''
    fig = plt.figure()
    ax = Axes3D(fig, auto_add_to_figure =False)
    fig.add_axes(ax)
    [X,Y] = np.meshgrid(x,t)

    # TODO 查清楚x, t 的位置关系。
    # TODO 将精确解的图像话出来比较一下。
    temp = np.sin(np.multiply(pi,X+Y))
    U = np.power(temp,40)
    ax.plot_surface(X,Y,U,cmap=plt.cm.winter)
    plt.title(["jingquejie"])
    plt.show()
    return U

def plotdata2(U,U1,x):
    T = [0.00 ,0.12, 0.20, 0.80]
    tn = [ceil(i/0.04) for i in T]
    plt.plot(x,U[tn[0],:])
    plt.plot(x,U1[tn[0],:])
    plt.title("t = 0.00")
    plt.legend(["shuzhijie","jingquejie"])
    plt.show()
    plt.plot(x,U[tn[1],:])
    plt.plot(x,U1[tn[1],:])
    plt.legend(["shuzhijie","jingquejie"])
    plt.title("t = 0.12")
    plt.show()
    plt.plot(x,U[tn[2],:])
    plt.plot(x,U1[tn[2],:])
    plt.legend(["shuzhijie","jingquejie"])
    plt.title("t = 0.20")
    plt.show()
    plt.plot(x,U[tn[3],:])
    plt.plot(x,U1[tn[3],:])
    plt.legend(["shuzhijie","jingquejie"])
    plt.title("t = 0.80")
    plt.show()
def main():
    B, U, J, N, x, t = init()
    U = yingfeng(B,U,J,N)
    plotdata(x,t,U)
    U1 = precise(x, t)
    plotdata2(U,U1,x)
main()
```