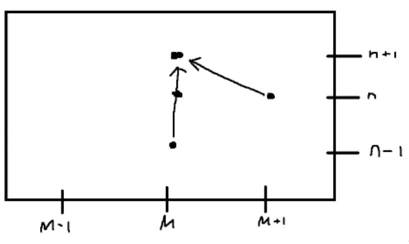
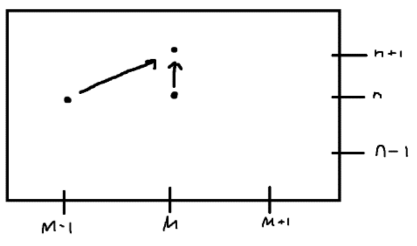
Forwards time backwards space; Central time forwards space

In the given equation, the initial time and initial x boundary conditions are given, meaning you can move from either side of the mesh. A FTBS scheme being viable due to the given u(0,t) condition, as well as a CTFS scheme being viable due to both u(x,0) and u(0,t) conditions. For both schemes the direction of the mesh will be as shown on the images bellow, with the addition of a u_m^n point due to the βu function in the equation. ($\beta = coefficent\ of\ u(x,t)$. Additionally, an FTBS scheme is explicit and 1 step, making it easy to implement. The CTFS scheme is explicit but multistep, which is considerably more stable.

FTBS



CTFS



Numerical scheme construction with consistency

Equation:

$$\mathrm{P}(\partial \mathrm{x}, \partial \mathrm{t}) = \partial \mathrm{t} + \alpha\, \partial \mathrm{x} + \mathrm{P}[\mathrm{u}], \qquad \mathrm{P}[\mathrm{u}] = \beta \mathrm{u}$$

$$FTBS\colon u_t = \frac{1}{k}[u_m^{n+1} - u_m^n], \quad u_x = \frac{1}{h}[u_m^n - u_{m-1}^n]$$

$$CTFS\colon u_t = \frac{1}{2k}[u_m^{n+1} - u_m^{n-1}], \quad u_x = \frac{1}{h}[u_{m+1}^n - u_m^n]$$

FTBS Scheme: $u_t + \alpha u_x + \beta u$

$$\frac{1}{k}[u_m^{n+1} - u_m^n] + \alpha\frac{1}{h}[u_m^n - u_{m-1}^n] + \beta u_m^n = 0$$

$$u_m^{n+1} = u_m^n\left(k - \frac{\alpha k}{h} - k\beta\right) + \left(\frac{\alpha k}{h}\right)u_{m-1}^n$$

$$= u_m^{n+1} = u_m^n(k - \alpha\lambda - k\beta) + (\alpha\lambda)u_{m-1}^n$$

CTFS Scheme

$$\frac{1}{2k}[u_m^{n+1} - u_m^{n-1}] + \frac{\alpha}{h}[u_{m+1}^n - u_m^n] + \beta u_m^n = 0$$

$$u_m^{n+1} = 2k\left[-\left[\frac{\alpha}{h}u_{m+1}^n - \frac{\alpha}{h}u_m^n\right] - \beta u_m^n\right] + u_m^{n-1}$$

$$u_m^{n+1} = -2k\frac{\alpha}{h}u_{m+1}^n + 2k\frac{\alpha}{h}u_m^n - 2k\,\beta u_m^n + u_m^{n-1}$$

$$u_m^{n+1} = -2\alpha\lambda u_{m+1}^n + [2\alpha\lambda - 2k\,\beta]u_m^n + u_m^{n-1}$$

Using test function $\varphi(\mathrm{x},\mathrm{t})$; $P_\varphi = \varphi_t + \alpha\varphi_x + \beta\varphi(\mathrm{x},\mathrm{t})$

Where the taylor expansion of $\varphi(x, t \pm k)$ *is*: $\varphi(x,t) \pm k\varphi t + \frac{k^2}{2!}\varphi tt \pm \cdots$

and the expansion of $\varphi(x \pm h)$ *is*: $\varphi(x,t) \pm h\varphi x + \frac{h2}{2!}\varphi xx \pm \frac{h3}{3!}\varphi xxx + \frac{h4}{4!}\varphi xxxx + \ldots.$

Scheme consistency FTBS

The initial boundary value problem becomes:

$$\frac{1}{k}[\varphi(x,t+k) - \varphi(x,t)] + \frac{\alpha}{h}[\varphi(x,t) - \varphi(x,t-k)] + \beta\varphi(x,t)$$

Inputting this into our new equation yields:

$$P_{k,h,\varphi} = \frac{1}{k}\left[\varphi(x,t) + k\varphi t + \frac{k^2}{2!}\varphi tt - \varphi(x,t)\right] + \frac{\alpha}{h}\left[\varphi(x,t) - \varphi(x,t) - h\varphi x + \frac{h2}{2!}\varphi xx - \frac{h3}{3!}\varphi xxx\right] + \beta\varphi(x,t)$$

This reduces to: $P_{k,h,\varphi} = \left[\varphi t + \frac{k}{2!}\varphi tt\right] + \alpha\left[\varphi x + \frac{h}{2!}\varphi xx - \frac{h^2}{3!}\varphi xxx\right] + \beta\varphi(x,t)$

Taking the difference between these schemes yield:

$$P_\varphi - P_{k,h,\varphi} = \varphi_t + \alpha\varphi_x + \beta\varphi(x,t) - \frac{1}{k}\left[\varphi(x,t) + k\varphi t + \frac{k^2}{2!}\varphi tt - \varphi(x,t)\right]$$

$$-\frac{\alpha}{h}\left[\varphi(x,t) - \varphi(x,t) - h\varphi x + \frac{h2}{2!}\varphi xx - \frac{h3}{3!}\varphi xxx\right] - \beta\varphi(x,t)$$

$= \frac{k}{2!}\varphi tt - \alpha\frac{h}{2!}\varphi xx - \frac{h^2}{3!}\varphi xxx$. We can see that this will tend to 0 as $h, k \to \infty$,

deducing that this finite difference scheme is consistent with the governing PDE.

Scheme consistency CTFS

Using test function $\varphi(x,t)$; $P_\varphi = \varphi(x,t+k) + \alpha\varphi(x+h,t) + \beta\varphi(x,t)$

Where the initial boundary value problem becomes:,

$$\frac{1}{2k}[\varphi(x,t+k) - \varphi(x,t-k)] + \frac{\alpha}{h}[\varphi(x+h,t) - \varphi(x,t)] + \beta\varphi(x,t)$$

Where the taylor expansion of $\varphi(x, t \pm k)$ *is*: $\varphi(x,t) \pm k\varphi t + \frac{k^2}{2!}\varphi tt \pm \cdots$

and the expansion of $\varphi(x \pm h)$ *is*: $\varphi(x,t) \pm h\varphi x + \frac{h2}{2!}\varphi xx \pm \frac{h3}{3!}\varphi xxx + \frac{h4}{4!}\varphi xxxx + \ldots.$

Inputting this into our new equation yields:

$$P_{k,h,\varphi} = \frac{1}{2k}\left[\varphi(x,t) + k\varphi t + \frac{k^2}{2!}\varphi tt - \varphi(x,t) + k\varphi t - \frac{k^2}{2!}\varphi tt\right]$$
$$+ \frac{\alpha}{h}\left[\varphi(x,t) \pm h\varphi x + \frac{h2}{2!}\varphi xx + \frac{h3}{3!}\varphi xxx - \varphi(x,t)\right] + \beta\varphi(x,t)$$

This reduces to: $P_{k,h,\varphi} = [\varphi t] + \alpha\left[\varphi x + \frac{h}{2!}\varphi xx + \frac{h^2}{3!}\varphi xxx\right] + \beta\varphi(x,t)$

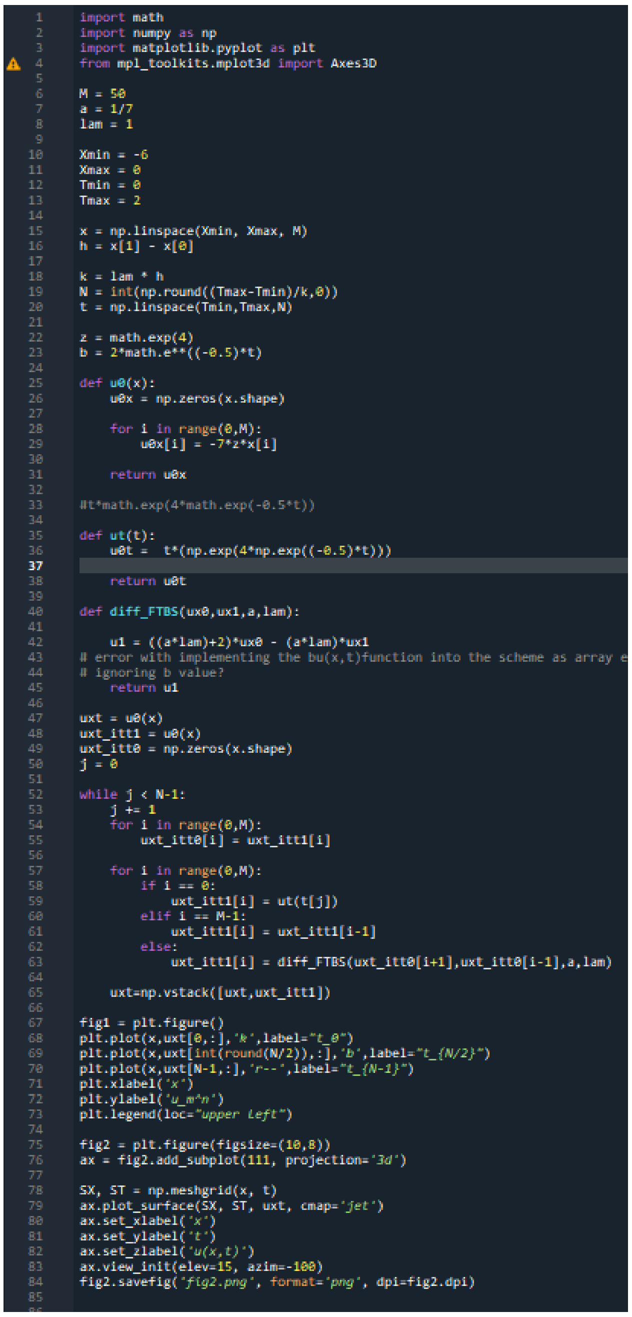
Taking the difference between these schemes yield:

$$P_\varphi - P_{k,h,\varphi} = \varphi(x,t+k) + \alpha\varphi(x+h,t) + \beta\varphi(x,t) - [\varphi t] + \alpha\left[\varphi x + \frac{h}{2!}\varphi xx + \frac{h^2}{3!}\varphi xxx\right] + \beta\varphi(x,t)$$

$= -\left[\frac{k}{2!}\varphi tt\right] - \alpha\left[\frac{\mathrm{h}}{2!}\varphi xx - \frac{h^2}{3!}\varphi xxx\right]$: We can see that this will tend to 0 as $h, k \to \infty$,

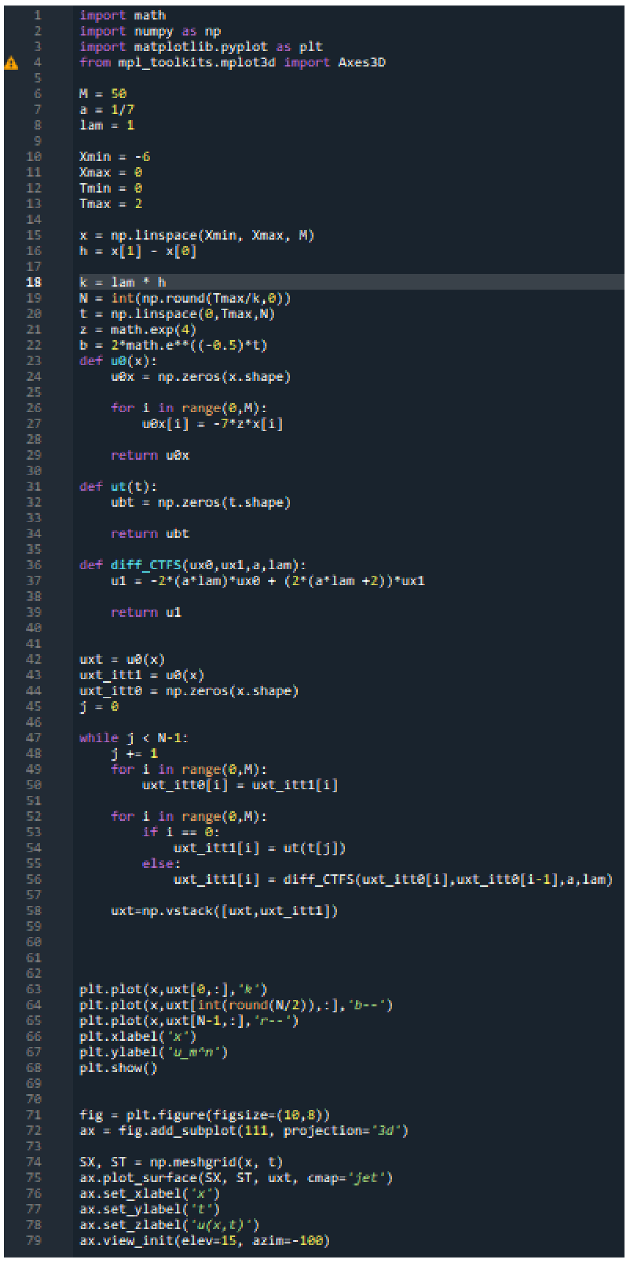
deducing that this finite difference scheme is consistent with the governing PDE

FTBS



```python
 1  import math
 2  import numpy as np
 3  import matplotlib.pyplot as plt
 4  from mpl_toolkits.mplot3d import Axes3D
 5
 6  M = 50
 7  a = 1/7
 8  lam = 1
 9
10  Xmin = -6
11  Xmax = 0
12  Tmin = 0
13  Tmax = 2
14
15  x = np.linspace(Xmin, Xmax, M)
16  h = x[1] - x[0]
17
18  k = lam * h
19  N = int(np.round((Tmax-Tmin)/k,0))
20  t = np.linspace(Tmin,Tmax,N)
21
22  z = math.exp(4)
23  b = 2*math.e**((-0.5)*t)
24
25  def u0(x):
26      u0x = np.zeros(x.shape)
27
28      for i in range(0,M):
29          u0x[i] = -7*z*x[i]
30
31      return u0x
32
33  #t*math.exp(4*math.exp(-0.5*t))
34
35  def ut(t):
36      u0t =  t*(np.exp(4*np.exp((-0.5)*t)))
37
38      return u0t
39
40  def diff_FTBS(ux0,ux1,a,lam):
41
42      u1 = ((a*lam)+2)*ux0 - (a*lam)*ux1
43  # error with implementing the bu(x,t)function into the scheme as array e
44  # ignoring b value?
45      return u1
46
47  uxt = u0(x)
48  uxt_itt1 = u0(x)
49  uxt_itt0 = np.zeros(x.shape)
50  j = 0
51
52  while j < N-1:
53      j += 1
54      for i in range(0,M):
55          uxt_itt0[i] = uxt_itt1[i]
56
57      for i in range(0,M):
58          if i == 0:
59              uxt_itt1[i] = ut(t[j])
60          elif i == M-1:
61              uxt_itt1[i] = uxt_itt1[i-1]
62          else:
63              uxt_itt1[i] = diff_FTBS(uxt_itt0[i+1],uxt_itt0[i-1],a,lam)
64
65      uxt=np.vstack([uxt,uxt_itt1])
66
67  fig1 = plt.figure()
68  plt.plot(x,uxt[0,:],'k',label="t_0")
69  plt.plot(x,uxt[int(round(N/2)),:],'b',label="t_{N/2}")
70  plt.plot(x,uxt[N-1,:],'r--',label="t_{N-1}")
71  plt.xlabel('x')
72  plt.ylabel('u_m^n')
73  plt.legend(loc="upper left")
74
75  fig2 = plt.figure(figsize=(10,8))
76  ax = fig2.add_subplot(111, projection='3d')
77
78  SX, ST = np.meshgrid(x, t)
79  ax.plot_surface(SX, ST, uxt, cmap='jet')
80  ax.set_xlabel('x')
81  ax.set_ylabel('t')
82  ax.set_zlabel('u(x,t)')
83  ax.view_init(elev=15, azim=-100)
84  fig2.savefig('fig2.png', format='png', dpi=fig2.dpi)
85
```

CTFS

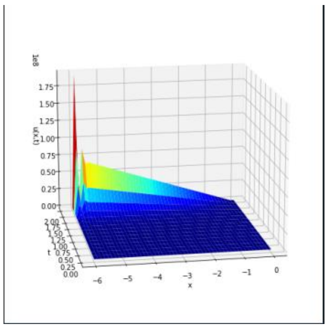


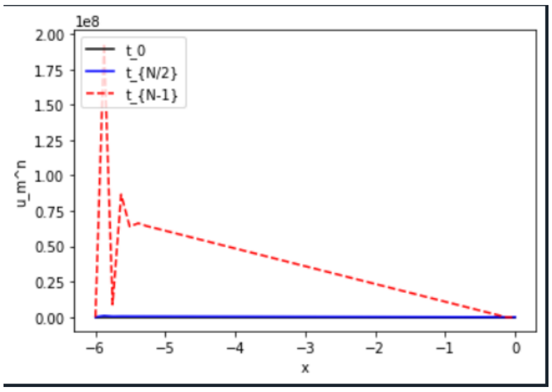
```python
 1  import math
 2  import numpy as np
 3  import matplotlib.pyplot as plt
 4  from mpl_toolkits.mplot3d import Axes3D
 5
 6  M = 50
 7  a = 1/7
 8  lam = 1
 9
10  Xmin = -6
11  Xmax = 0
12  Tmin = 0
13  Tmax = 2
14
15  x = np.linspace(Xmin, Xmax, M)
16  h = x[1] - x[0]
17
18  k = lam * h
19  N = int(np.round(Tmax/k,0))
20  t = np.linspace(0,Tmax,N)
21  z = math.exp(4)
22  b = 2*math.e**((-0.5)*t)
23  def u0(x):
24      u0x = np.zeros(x.shape)
25
26      for i in range(0,M):
27          u0x[i] = -7*z*x[i]
28
29      return u0x
30
31  def ut(t):
32      ubt = np.zeros(t.shape)
33
34      return ubt
35
36  def diff_CTFS(ux0,ux1,a,lam):
37      u1 = -2*(a*lam)*ux0 + (2*(a*lam +2))*ux1
38
39      return u1
40
41
42  uxt = u0(x)
43  uxt_itt1 = u0(x)
44  uxt_itt0 = np.zeros(x.shape)
45  j = 0
46
47  while j < N-1:
48      j += 1
49      for i in range(0,M):
50          uxt_itt0[i] = uxt_itt1[i]
51
52      for i in range(0,M):
53          if i == 0:
54              uxt_itt1[i] = ut(t[j])
55          else:
56              uxt_itt1[i] = diff_CTFS(uxt_itt0[i],uxt_itt0[i-1],a,lam)
57
58      uxt=np.vstack([uxt,uxt_itt1])
59
60
61
62
63  plt.plot(x,uxt[0,:],'k')
64  plt.plot(x,uxt[int(round(N/2)),:],'b--')
65  plt.plot(x,uxt[N-1,:],'r--')
66  plt.xlabel('x')
67  plt.ylabel('u_m^n')
68  plt.show()
69
70
71  fig = plt.figure(figsize=(10,8))
72  ax = fig.add_subplot(111, projection='3d')
73
74  SX, ST = np.meshgrid(x, t)
75  ax.plot_surface(SX, ST, uxt, cmap='jet')
76  ax.set_xlabel('x')
77  ax.set_ylabel('t')
78  ax.set_zlabel('u(x,t)')
79  ax.view_init(elev=15, azim=-100)
```

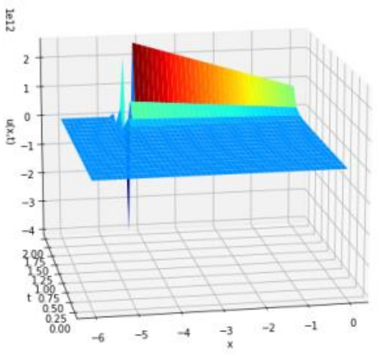
Relevant libraries are imported, variables which we can changed based on other information relating to the PDE and its conditions are defined, as well as the creation of functions which are the initial boundary conditions and the adjusted scheme which I will be using (FTBS and CTFS). For the left image, lines 42-65 create a mesh which (based on the defined scheme) moves, maps and layers values under given boundary conditions. For the FTBS scheme the time step moves forwards where the X step moves backwards (as shown in the stencil in part a), the code then computes points in space using the defined FTBS scheme derived from the given PDE, this follows a loop which returns the final result after all timesteps have been calculated. As this is an explicit one step scheme, the code goes up a time step, but down a space step making it simple, but potentially unstable and less accurate vs an implicit scheme.

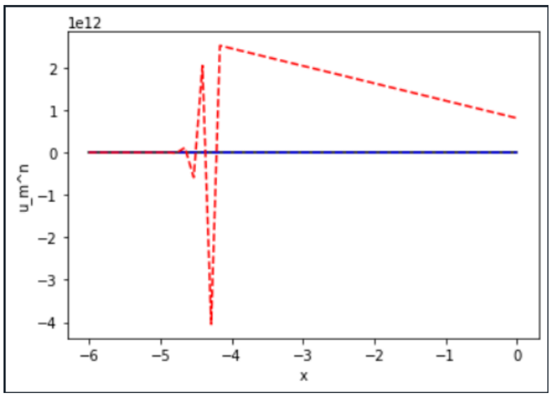
The second code is the CTFS scheme, which works in a similar way to the FTBS coding but uses the aspect of going forward extra steps using both boundary conditions. This scheme is explicit but multistep meaning an additional loop section in the code had to be added in order to account for this, additionally adjusting the scheme parameters and accounting for the u(x,o) boundary.

Figures displaying numerical solutions of the PDE









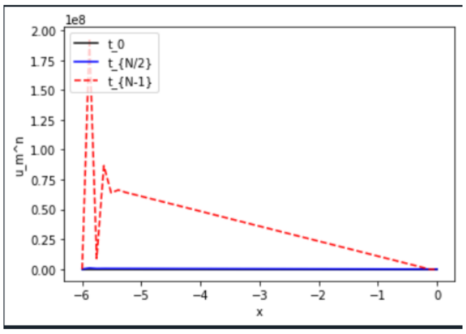
Stability of each scheme

A consistent finite difference scheme (assuming PDE is well posed) is only convergent if its stable. Schemes are considered explicit if an approximation of $yk + 1$ in $xk + 1$ can be calculated in already computed values yi,i≤k, otherwise implicit.

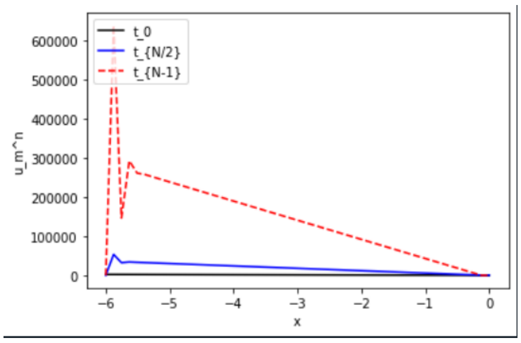
FTBS

Changing the value of lambda between 1 and 1.9 retains instability at around u(x,t) = 0-2, where increasing to 2.0 and above dramatically changes the range between 0 and 600,000.

Lambda = 1



Lambda = 2

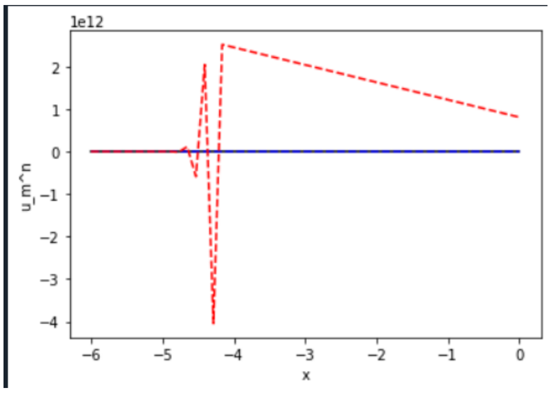


Keeping h and k equal, thus making lambda = 1 appears to be dramatically less unstable than higher Lambda values (for when h < k) . setting the number of spacial boundary nodes to below 30 also seems to achieve the same result as increasing lambda above 2 for this FTBS scheme, meaning M values too low, and lambda values too high significantly reduce the stability of the scheme.

CTBS

Reducing M to 17 and below with lambda >3 this time yields the same outcome as the FTBS scheme (when FTBS M value is < 30)

Lambda =1



Lambda = 3