Post_processing

April 11, 2023

1 Question (5) (a)

1.1 Perform numerical experiments with implicit Euler scheme. Use the same equation and parameters provided in questions 3 and 4. Solver the linear system at each time step using the Jacobi method (tolerance= 10^{-4}; do not use any libraries).

2 Answer (5) (a):

3 For the tolerance value 1e-4

4 Import the necessary libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

5 Reading the name of the files generated

[2]:
```
Output_files = pd.read_csv("Output_file_names.csv",delimiter=",",header=None).
↪to_numpy()
Output_files = np.squeeze(Output_files)
Output_files = Output_files.tolist()
```

5.1 Output files generated

[3]:
```
Output_files
```

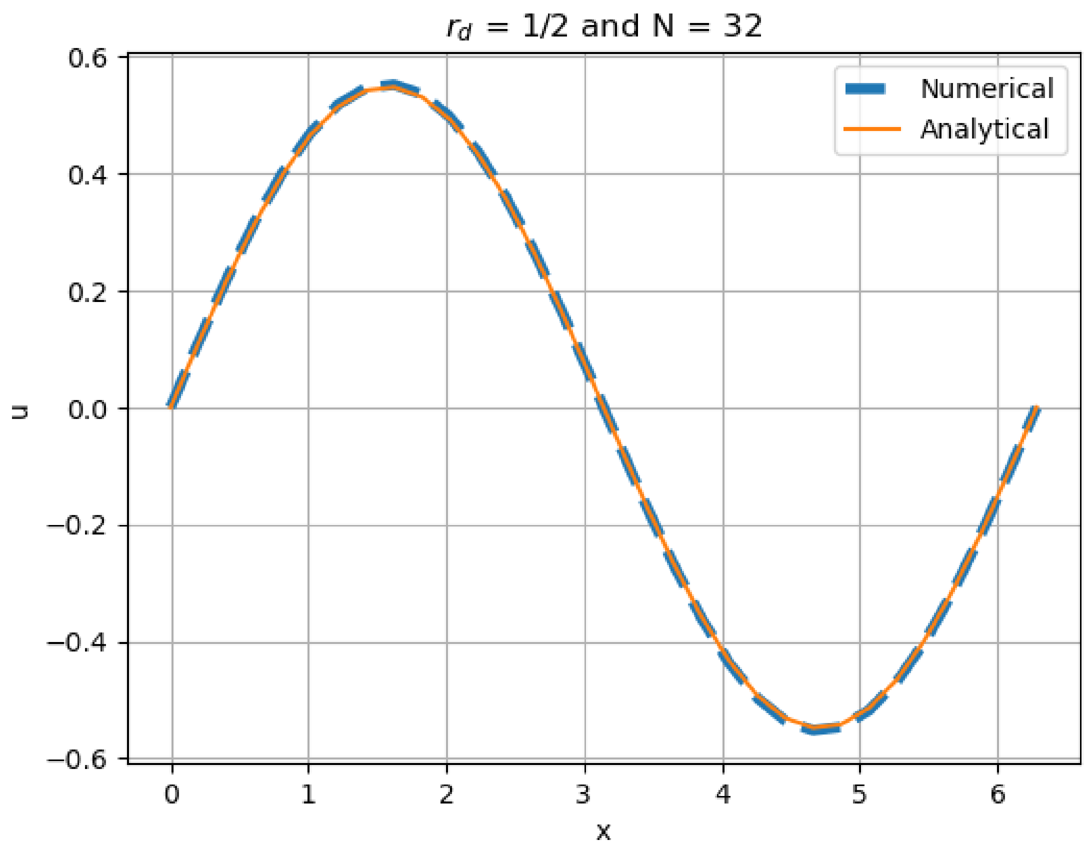
[3]:
```
['Question_5_Implicit_u_n_t_0.400000_N_32_rd_0.500000_.csv',
 'Analytical_Solution_u_n_t_0.400000_N_32_rd_0.500000_.csv',
 'Question_5_Implicit_Average_Error_u_n_t_0.400000_N_32_rd_0.500000_.csv',
 'Question_5_Implicit_u_n_t_0.400000_N_32_rd_0.166667_.csv',
 'Analytical_Solution_u_n_t_0.400000_N_32_rd_0.166667_.csv',
 'Question_5_Implicit_Average_Error_u_n_t_0.400000_N_32_rd_0.166667_.csv',
 'Question_5_Implicit_u_n_t_0.400000_N_64_rd_0.500000_.csv',
 'Analytical_Solution_u_n_t_0.400000_N_64_rd_0.500000_.csv',
```

```
'Question_5_Implicit_Average_Error_u_n_t_0.400000_N_64_rd_0.500000_.csv',
'Question_5_Implicit_u_n_t_0.400000_N_64_rd_0.166667_.csv',
'Analytical_Solution_u_n_t_0.400000_N_64_rd_0.166667_.csv',
'Question_5_Implicit_Average_Error_u_n_t_0.400000_N_64_rd_0.166667_.csv',
'Question_5_Implicit_u_n_t_0.400000_N_128_rd_0.500000_.csv',
'Analytical_Solution_u_n_t_0.400000_N_128_rd_0.500000_.csv',
'Question_5_Implicit_Average_Error_u_n_t_0.400000_N_128_rd_0.500000_.csv',
'Question_5_Implicit_u_n_t_0.400000_N_128_rd_0.166667_.csv',
'Analytical_Solution_u_n_t_0.400000_N_128_rd_0.166667_.csv',
'Question_5_Implicit_Average_Error_u_n_t_0.400000_N_128_rd_0.166667_.csv',
'Question_5_Implicit_u_n_t_0.400000_N_256_rd_0.500000_.csv',
'Analytical_Solution_u_n_t_0.400000_N_256_rd_0.500000_.csv',
'Question_5_Implicit_Average_Error_u_n_t_0.400000_N_256_rd_0.500000_.csv',
'Question_5_Implicit_u_n_t_0.400000_N_256_rd_0.166667_.csv',
'Analytical_Solution_u_n_t_0.400000_N_256_rd_0.166667_.csv',
'Question_5_Implicit_Average_Error_u_n_t_0.400000_N_256_rd_0.166667_.csv']
```

5.2 Plotting the results for rd = 1/2 and N = 32 at $t_{end} = 0.4$

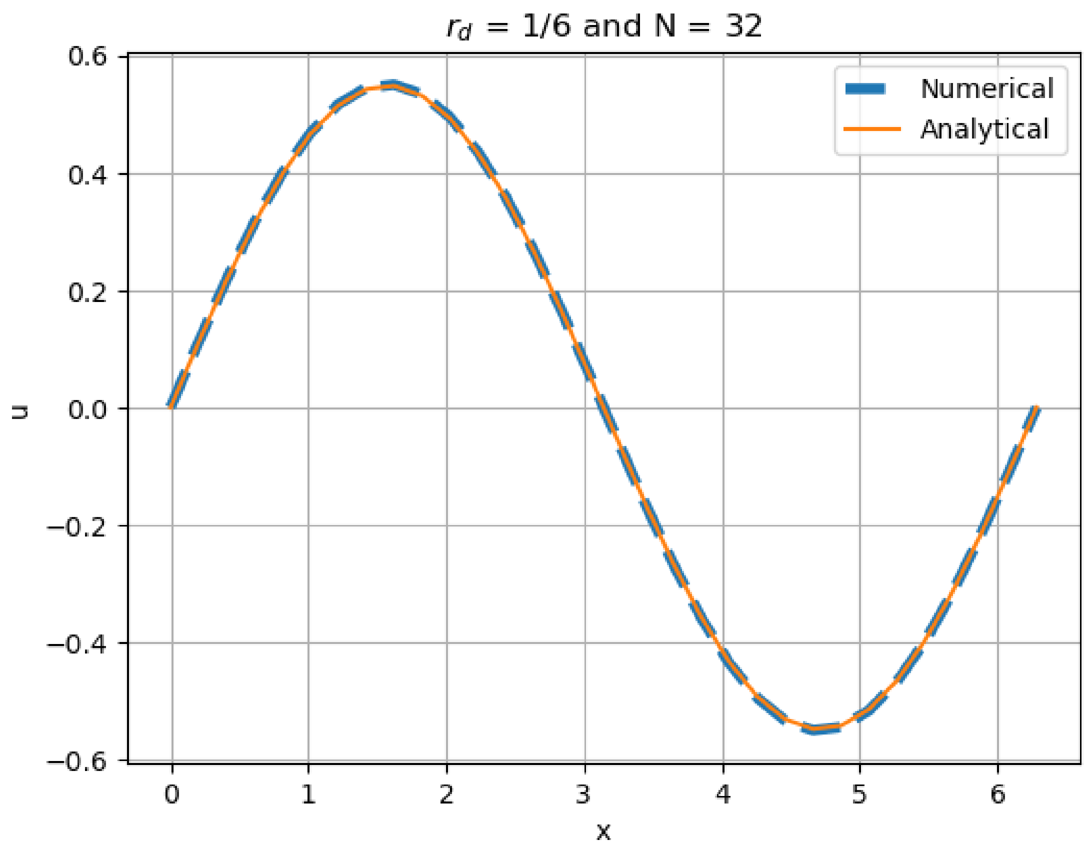
[4]:

```python
Numerical_Solution_N_32_rd_half = pd.read_csv(Output_files[0], delimiter =␣
  ↪",",header=None).to_numpy()
Analytical_Solution_N_32_rd_half = pd.read_csv(Output_files[1], delimiter =␣
  ↪",",header=None).to_numpy()
x = np.linspace(0,2*np.pi,Numerical_Solution_N_32_rd_half.shape[0])
plt.plot(x,Numerical_Solution_N_32_rd_half,linestyle='dashed',linewidth=4)
plt.plot(x,Analytical_Solution_N_32_rd_half)
plt.xlabel("x")
plt.ylabel("u")
plt.legend(["Numerical","Analytical"])
plt.title(R"$r_d$ = 1/2 and N = 32")
plt.grid()
plt.savefig("r_d_1_2_and_N_32.png",dpi = 1000)
plt.show()
```



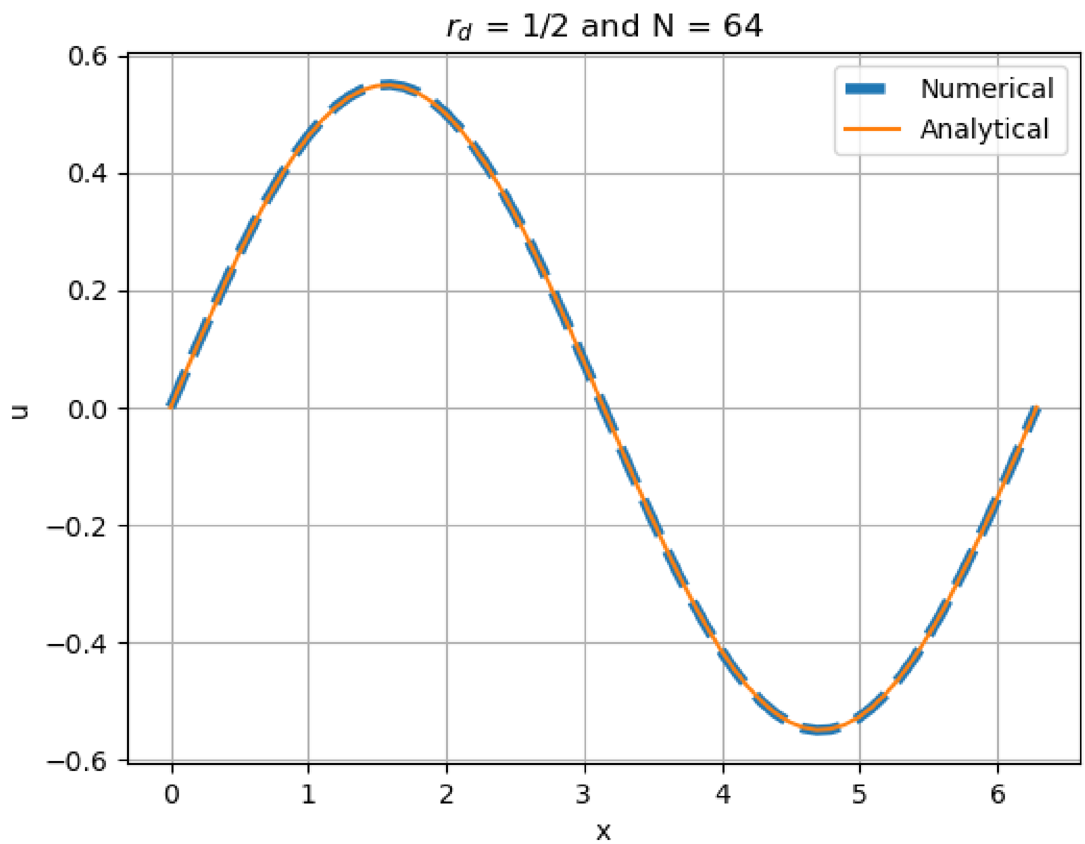
5.3 Plotting the results for rd = 1/6 and N = 32 at $t_{end} = 0.4$

[5]:
```
Numerical_Solution_N_32_rd_1_6 = pd.read_csv(Output_files[3], delimiter =␣
 ↪",",header=None).to_numpy()
Analytical_Solution_N_32_rd_1_6 = pd.read_csv(Output_files[4], delimiter =␣
 ↪",",header=None).to_numpy()
x = np.linspace(0,2*np.pi,Numerical_Solution_N_32_rd_1_6.shape[0])
plt.plot(x,Numerical_Solution_N_32_rd_1_6,linestyle='dashed',linewidth=4)
plt.plot(x,Analytical_Solution_N_32_rd_1_6)
plt.xlabel("x")
plt.ylabel("u")
plt.legend(["Numerical","Analytical"])
plt.title(R"$r_d$ = 1/6 and N = 32")
plt.grid()
plt.savefig("r_d_1_6_and_N_32.png",dpi = 1000)
plt.show()
```



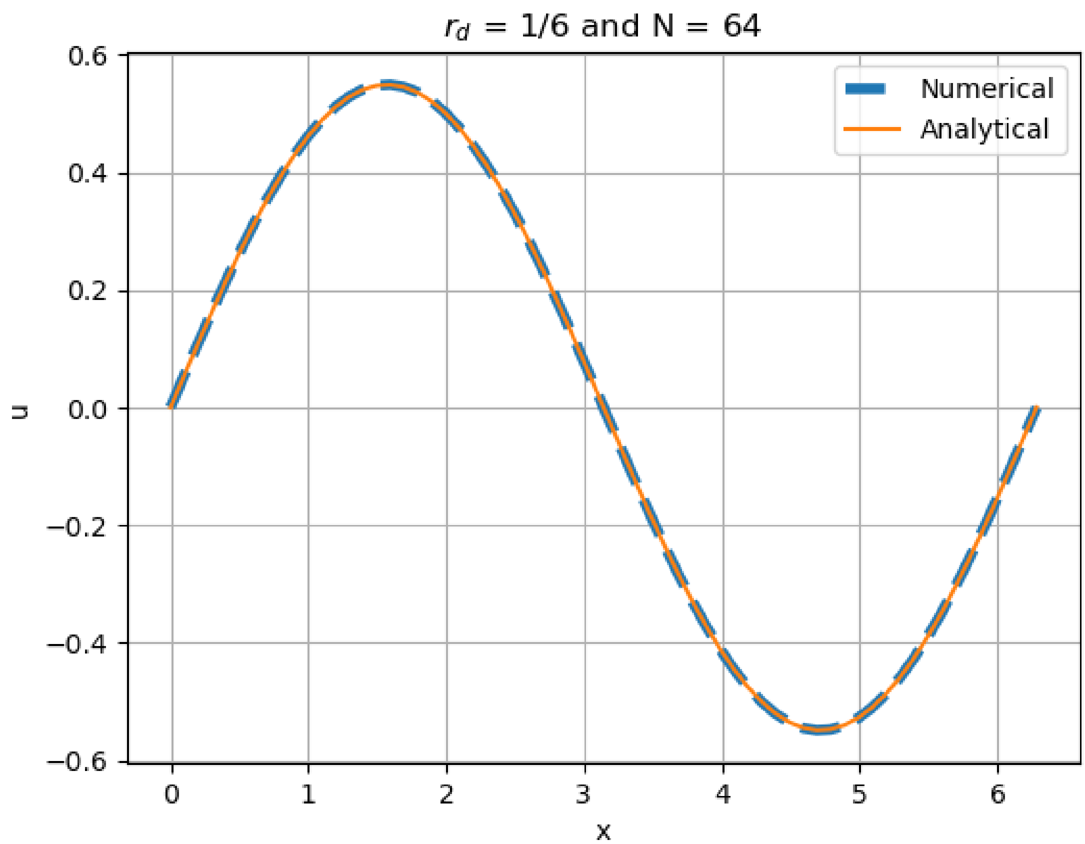
5.4 Plotting the results for rd = 1/2 and N = 64 at $t_{end} = 0.4$

[6]:
```
Numerical_Solution_N_64_rd_half = pd.read_csv(Output_files[6], delimiter =␣
  ↪",",header=None).to_numpy()
Analytical_Solution_N_64_rd_half = pd.read_csv(Output_files[7], delimiter =␣
  ↪",",header=None).to_numpy()
x = np.linspace(0,2*np.pi,Numerical_Solution_N_64_rd_half.shape[0])
plt.plot(x,Numerical_Solution_N_64_rd_half,linestyle='dashed',linewidth=4)
plt.plot(x,Analytical_Solution_N_64_rd_half)
plt.xlabel("x")
plt.ylabel("u")
plt.legend(["Numerical","Analytical"])
plt.title(R"$r_d$ = 1/2 and N = 64")
plt.grid()
plt.savefig("r_d_1_2_and_N_64.png",dpi = 1000)
plt.show()
```



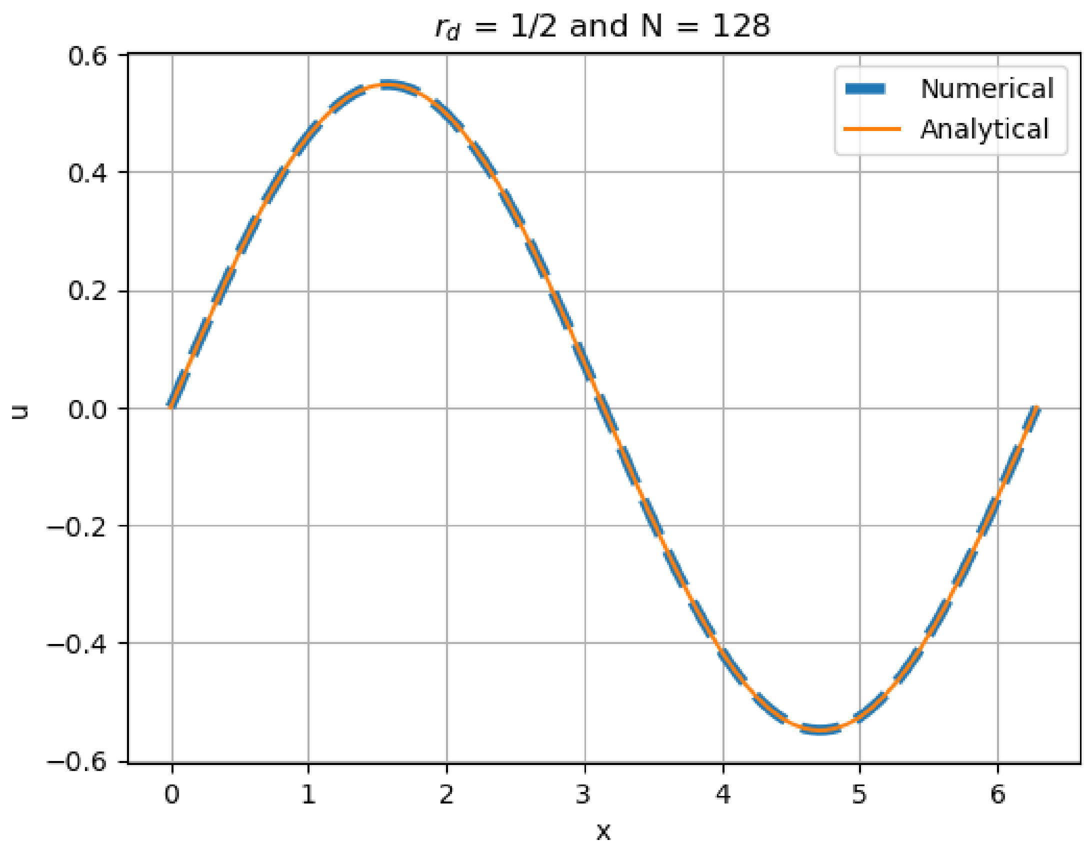
5.5 Plotting the results for rd = 1/6 and N = 64 at $t_{end} = 0.4$

[7]:
```
Numerical_Solution_N_64_rd_1_6 = pd.read_csv(Output_files[9], delimiter =␣
  ↪",",header=None).to_numpy()
Analytical_Solution_N_64_rd_1_6 = pd.read_csv(Output_files[10], delimiter =␣
  ↪",",header=None).to_numpy()
x = np.linspace(0,2*np.pi,Numerical_Solution_N_64_rd_1_6.shape[0])
plt.plot(x,Numerical_Solution_N_64_rd_1_6,linestyle='dashed',linewidth=4)
plt.plot(x,Analytical_Solution_N_64_rd_1_6)
plt.xlabel("x")
plt.ylabel("u")
plt.legend(["Numerical","Analytical"])
plt.title(R"$r_d$ = 1/6 and N = 64")
plt.grid()
plt.savefig("r_d_1_6_and_N_64.png",dpi = 1000)
plt.show()
```



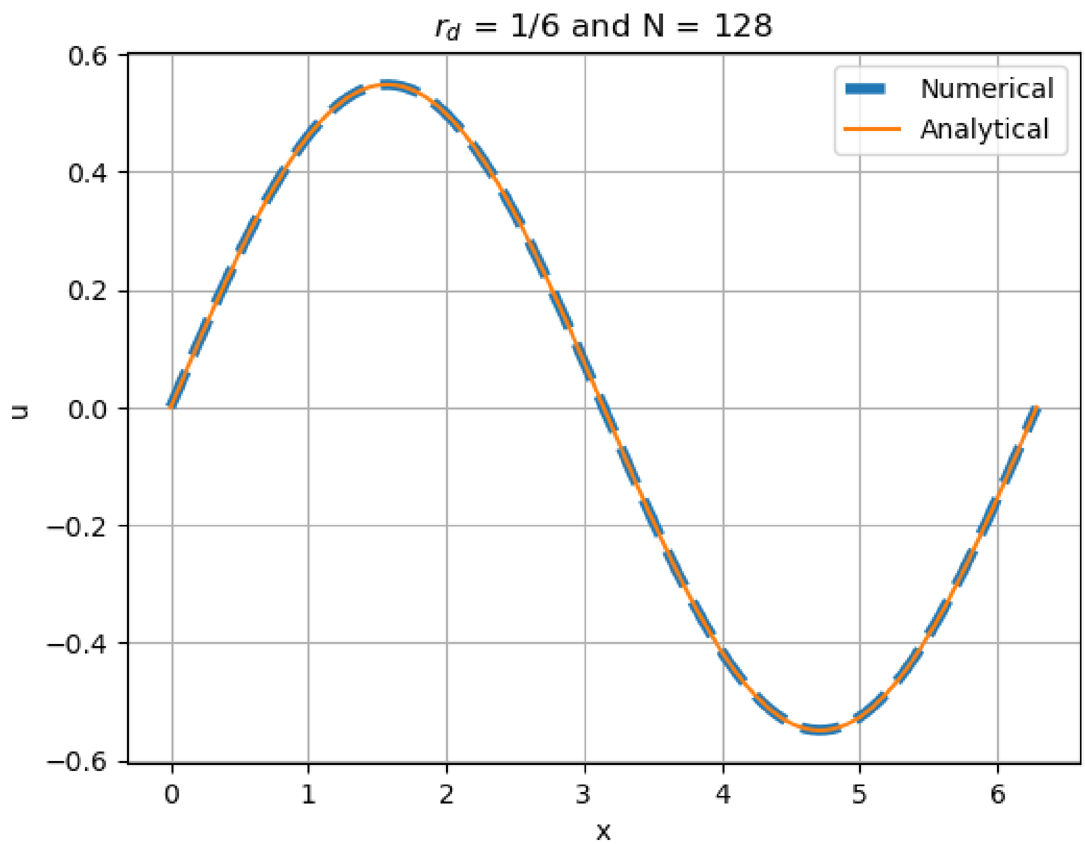
5.6 Plotting the results for rd = 1/2 and N = 128 at $t_{end} = 0.4$

[8]:
```
Numerical_Solution_N_128_rd_half = pd.read_csv(Output_files[12], delimiter =␣
  ↪",",header=None).to_numpy()
Analytical_Solution_N_128_rd_half = pd.read_csv(Output_files[13], delimiter =␣
  ↪",",header=None).to_numpy()
x = np.linspace(0,2*np.pi,Numerical_Solution_N_128_rd_half.shape[0])
plt.plot(x,Numerical_Solution_N_128_rd_half,linestyle='dashed',linewidth=4)
plt.plot(x,Analytical_Solution_N_128_rd_half)
plt.xlabel("x")
plt.ylabel("u")
plt.legend(["Numerical","Analytical"])
plt.title(R"$r_d$ = 1/2 and N = 128")
plt.grid()
plt.savefig("r_d_1_2_and_N_128.png",dpi = 1000)
plt.show()
```



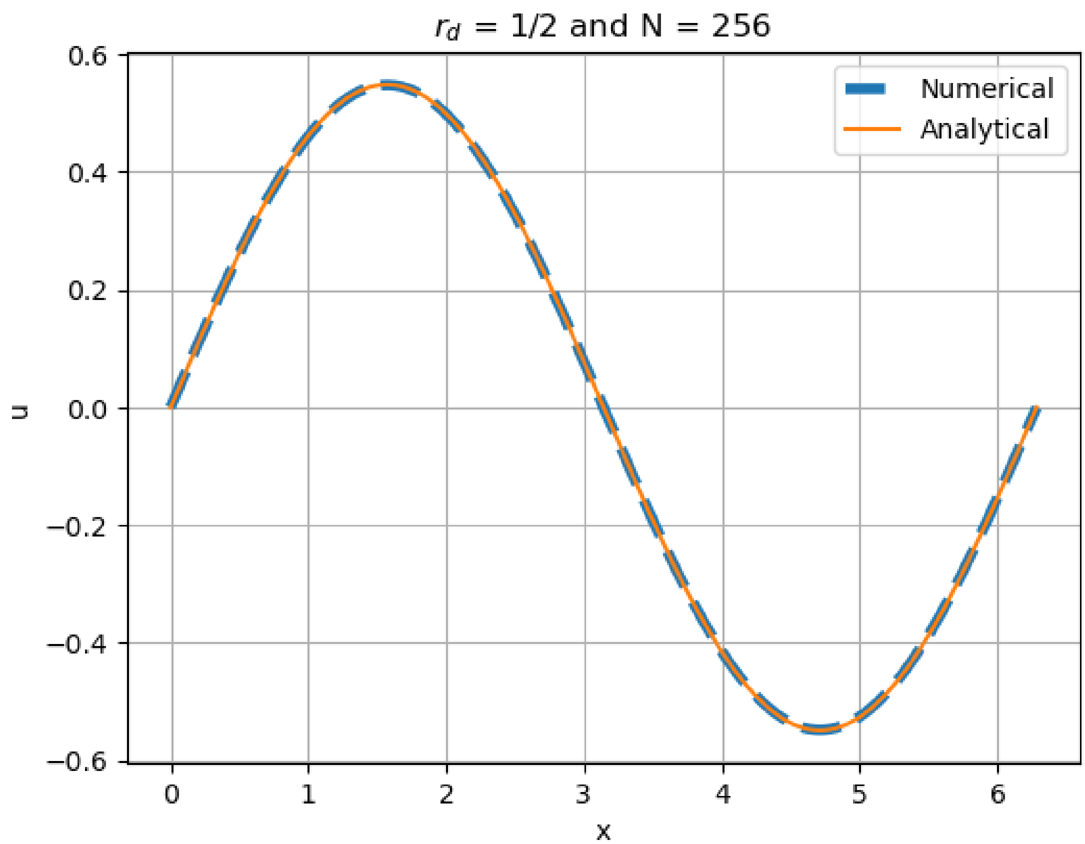
5.7 Plotting the results for rd = 1/6 and N = 128 at $t_{end} = 0.4$

[9]:
```
Numerical_Solution_N_128_rd_1_6 = pd.read_csv(Output_files[15], delimiter =␣
 ↪",",header=None).to_numpy()
Analytical_Solution_N_128_rd_1_6 = pd.read_csv(Output_files[16], delimiter =␣
 ↪",",header=None).to_numpy()
x = np.linspace(0,2*np.pi,Numerical_Solution_N_128_rd_1_6.shape[0])
plt.plot(x,Numerical_Solution_N_128_rd_1_6,linestyle='dashed',linewidth=4)
plt.plot(x,Analytical_Solution_N_128_rd_1_6)
plt.xlabel("x")
plt.ylabel("u")
plt.legend(["Numerical","Analytical"])
plt.title(R"$r_d$ = 1/6 and N = 128")
plt.grid()
plt.savefig("r_d_1_6_and_N_128.png",dpi = 1000)
plt.show()
```



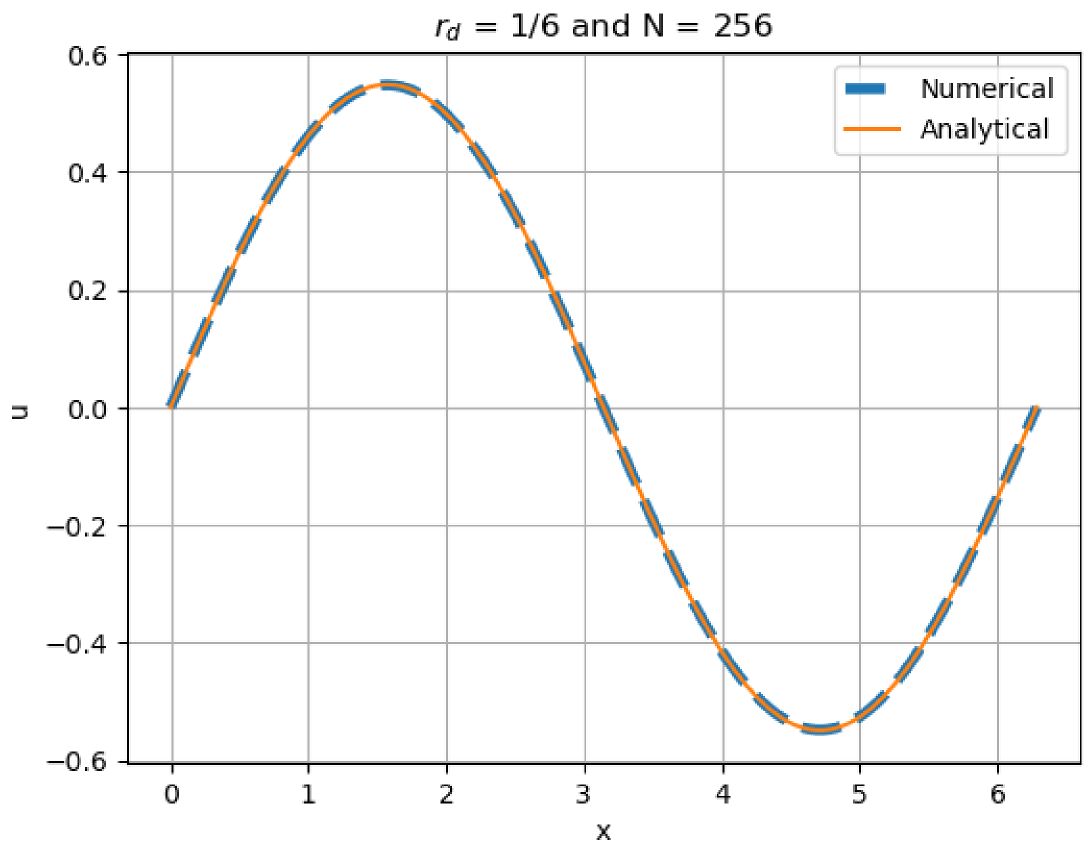
5.8 Plotting the results for rd = 1/2 and N = 256 at $t_{end} = 0.4$

[10]:
```
Numerical_Solution_N_256_rd_half = pd.read_csv(Output_files[18], delimiter =␣
 ↪",",header=None).to_numpy()
Analytical_Solution_N_256_rd_half = pd.read_csv(Output_files[19], delimiter =␣
 ↪",",header=None).to_numpy()
x = np.linspace(0,2*np.pi,Numerical_Solution_N_256_rd_half.shape[0])
plt.plot(x,Numerical_Solution_N_256_rd_half,linestyle='dashed',linewidth=4)
plt.plot(x,Analytical_Solution_N_256_rd_half)
plt.xlabel("x")
plt.ylabel("u")
plt.legend(["Numerical","Analytical"])
plt.title(R"$r_d$ = 1/2 and N = 256")
plt.grid()
plt.savefig("r_d_1_2_and_N_256.png",dpi = 1000)
plt.show()
```



5.9 Plotting the results for rd = 1/6 and N = 256 at $t_{end} = 0.4$

[11]:
```
Numerical_Solution_N_256_rd_1_6 = pd.read_csv(Output_files[21], delimiter =␣
  ↪",",header=None).to_numpy()
Analytical_Solution_N_256_rd_1_6 = pd.read_csv(Output_files[22], delimiter =␣
  ↪",",header=None).to_numpy()
x = np.linspace(0,2*np.pi,Numerical_Solution_N_256_rd_1_6.shape[0])
plt.plot(x,Numerical_Solution_N_256_rd_1_6,linestyle='dashed',linewidth=4)
plt.plot(x,Analytical_Solution_N_256_rd_1_6)
plt.xlabel("x")
plt.ylabel("u")
plt.legend(["Numerical","Analytical"])
plt.title(R"$r_d$ = 1/6 and N = 256")
plt.grid()
plt.savefig("r_d_1_6_and_N_256.png",dpi = 1000)
plt.show()
```



6 Average error vs step size for different values of rd in log-log scale

[12]:
```
rd = []
for i in range(2,len(Output_files),3):
    temp = pd.read_csv(Output_files[i], delimiter = ",",header=None).to_numpy().
 ↪squeeze()
    rd.append(temp.item())
rd_half = []
for i in range(0,len(rd),2):
    temp = rd[i]
    rd_half.append(temp)
rd_one_sixth = []
for i in range(1,len(rd),2):
    temp = rd[i]
    rd_one_sixth.append(temp)
```

[13]:
```
rd
```

[13]:
```
[0.00268342,
 0.00135759,
 0.000577935,
 0.000241834,
 3.84373e-05,
 3.5107e-05,
 0.000375391,
 0.000379308]
```

[14]:
```
rd_half
```

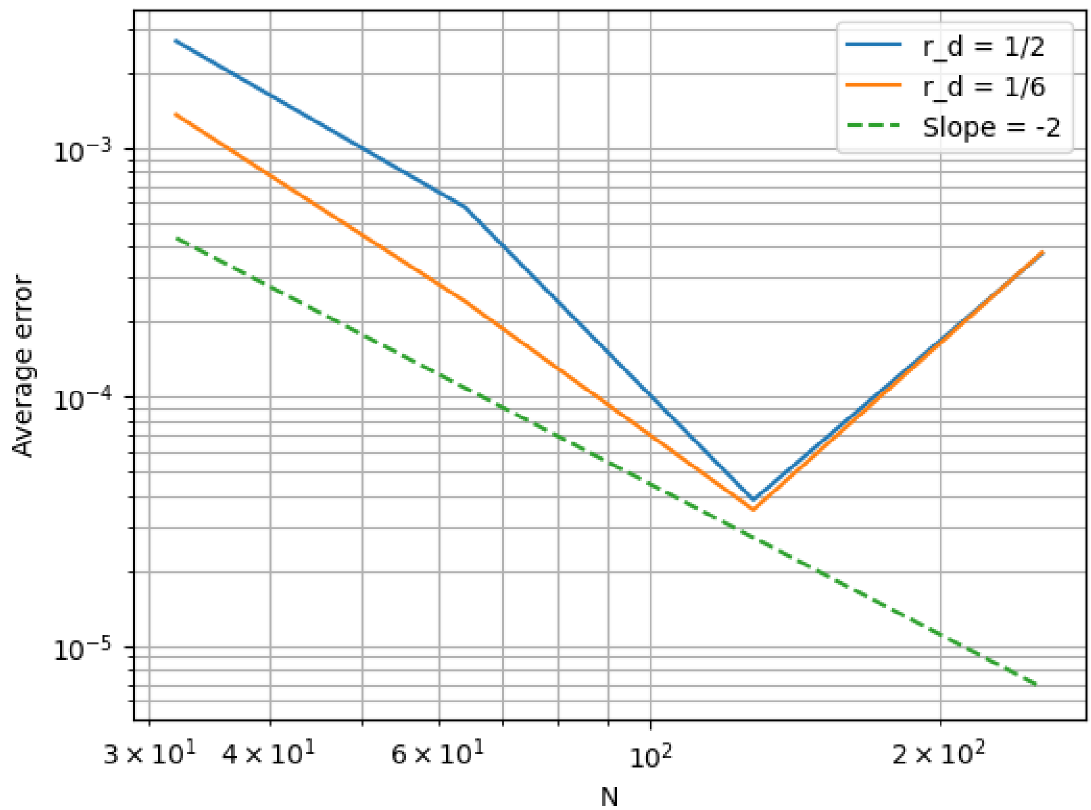
[14]:
```
[0.00268342, 0.000577935, 3.84373e-05, 0.000375391]
```

[15]:
```
rd_one_sixth
```

[15]:
```
[0.00135759, 0.000241834, 3.5107e-05, 0.000379308]
```

[16]:
```
N = [32,64,128,256]
```

[17]:
```
fig = plt.figure()
plt.loglog(N,rd_half)
plt.loglog(N,rd_one_sixth)
plt.plot(N,(1.50*np.array(N))**(-2),"--")
plt.legend(["r_d = 1/2", "r_d = 1/6", "Slope = -2"])
plt.xlabel("N")
plt.ylabel("Average error")
plt.grid(which='both')
plt.show()
fig.savefig("Average_Error_log.png",dpi = 500, bbox_inches="tight")
```



7 Question (5) (b):

7.1 In a N vs E(N) graph, compare the errors with explicit method for rd = 1/2. Provide an explanation for your observations.

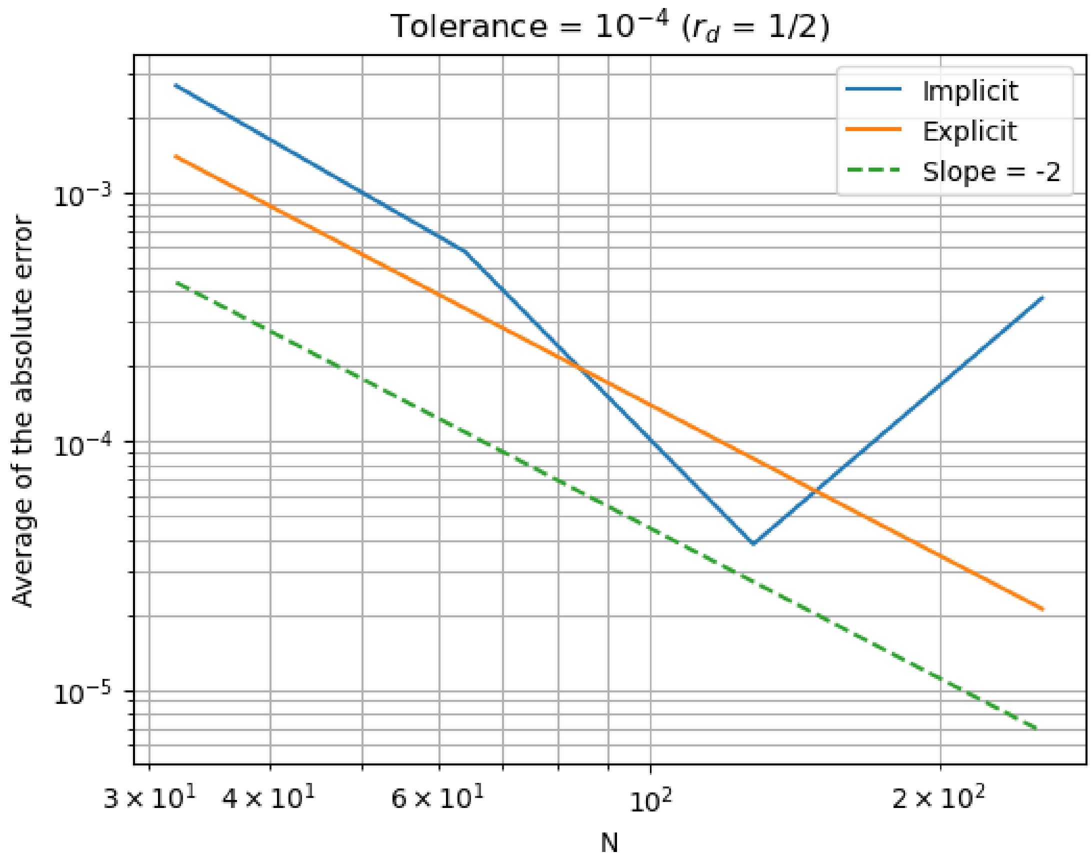
8 Answer (5) (b):

[18]:
```
Q_4_Error_Data = pd.read_csv("Question_4_Error_Data.csv",header=None)
```

9 Average error vs step size for different methods in log-log scale

[19]:
```
fig = plt.figure()
plt.loglog(N,rd_half)
plt.loglog(N,Q_4_Error_Data)
plt.plot(N,(1.50*np.array(N))**(-2),"--")
plt.legend(["Implicit", "Explicit", "Slope = -2"])
plt.xlabel("N")
plt.ylabel("Average of the absolute error")
plt.grid(which='both')
```

```
plt.title(R"Tolerance = $10^{-4}$ ($r_{d}$ = 1/2)")
plt.show()
fig.savefig("Average_Error_log_for_Different_Methods.png",dpi = 500,␣
 ↪bbox_inches="tight")
```



9.1 **In case of the implicit method, the truncation error dominates, when we are using lesser number of grid points but when we are using a finer grid the propagation error keep on accumulating over iterations and eventually dominates the truncation error. Hence, the error increases on using a finer grid, for a fixed tolerance.**

9.2 **We have found that on decreasing the tolerance from the 10^{-4} to 10^{-6} the error keeps on reducing.**