PDB

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


#Syarat awal
u_1_0 = 3 # Nilai u'(0)
u_0 = 9 # Nilai u(0)

#Syarat batas
a = 0 # Batas bawah
b = 1 # Batas atas
delta_x = 0.01 # Panjang partisi
n = (b-a)/delta_x + 1 # Banyak titik
```

Analitik

+ Code + Text

```
analitik = [] # Variabel kosong yang akan diisi hasil perhitungan secara analitik

def u_x(x):
    # Mendefinisikan fungsi u_x yang merupakan hasil dari perhitungan analitik
    f_x = -np.sin(x)+4*x+9 # Solusi dari persamaan diferensial secara analitik
    return f_x

for i in range(int(n)):
    analitik.append(u_x(a+i*delta_x)) # Memasukkan nilai u(x) ke dalam variabel "analitik"



analitik_kolom = np.array(analitik).T # Mentranspose variabel "analitik" agar mudah dilihat outputnya
print(analitik_kolom)
```

```
[ 9.          9.03000017  9.06000133  9.0900045   9.12001067  9.15002083
  9.18003599  9.21005715  9.24008531  9.27012145  9.30016658  9.3302217
  9.36028779  9.39036586  9.42045689  9.45056187  9.48068179  9.51081765
  9.54097043  9.57114111  9.60133067  9.6315401   9.66177038  9.69202248
  9.72229737  9.75259604  9.78291945  9.81326856  9.84364435  9.87404777
  9.90447979  9.93494136  9.96543344  9.99595697 10.02651291 10.05710219
 10.08772577 10.11838457 10.14907953 10.17981158 10.21058166 10.24139067
 10.27223955 10.3031292  10.33406053 10.36503447 10.39605189 10.42711371
 10.45822082 10.48937411 10.52057446 10.55182275 10.58311986 10.61446666
 10.64586401 10.67731277 10.7088138  10.74036795 10.77197606 10.80363898
 10.83535753 10.86713254 10.89896484 10.93085524 10.96280456 10.99481359
 11.02688315 11.05901401 11.09120698 11.12346282 11.15578231 11.18816623
 11.22061533 11.25313036 11.28571209 11.31836124 11.35107855 11.38386476
 11.41672058 11.44964673 11.48264391 11.51571283 11.54885417 11.58206863
 11.61535688 11.64871959 11.68215744 11.71567106 11.74926112 11.78292825
 11.81667309 11.85049626 11.88439838 11.91838006 11.9524419  11.9865845
 12.02080843 12.05511429 12.08950263 12.12397402 12.15852902]
```

Numerik

```
numerik = [] # Variabel kosong yang akan diisi hasil perhitungan secara numerik
numerik.append(u_0)# Memasukan suku pertama
numerik.append(u_0+u_1_0*delta_x)# Memasukan suku kedua menggunakan ekspansi deret Taylor untuk u(x+delta_x)
def u_j_x(x,a):
    # Mendefinisikan fungsi u_j_x yang merupakan hasil dari perhitungan numerik. Variabel "x" adalah nilai x dan variabel "a" adalah indeks dar
    u_j = 2*numerik[a-1] +delta_x**2*np.sin(x)-numerik[a-2] # Fungsi aproksimasi turunan kedua dengan metode beda hingga
    return u_j

for i in range(2,int(n)):
    numerik.append(u_j_x(a+delta_x*(i-1),i)) # Memasukkan nilai u_j(x) ke dalam variabel "numerik"


numerik_kolom = np.array(numerik).T # Mentranspose variabel "numerik" agar mudah dilihat outputnya
print(numerik_kolom)
```

```
[ 9.          9.03        9.060001    9.090004    9.12001     9.15002
  9.18003499  9.21005599  9.24008397  9.27011995  9.30016492  9.33021987
  9.3602858   9.39036369  9.42045456  9.45055937  9.48067913  9.51081482
  9.54096743  9.57113795  9.60132735  9.63153661  9.66176672  9.69201866
  9.72229339  9.7525919   9.78291514  9.81326409  9.84363972  9.87404298
  9.90447483  9.93493624  9.96542815  9.99595152 10.0265073  10.05709642
 10.08771983 10.11837847 10.14907327 10.17980517 10.21057508 10.24138393
 10.27223265 10.30312214 10.33405332 10.36502709 10.39604436 10.42710602
 10.45821298 10.48936611 10.5205663  10.55181443 10.58311139 10.61445803
 10.64585522 10.67730383 10.70880471 10.7403587  10.77196666 10.80362942
 10.83534782 10.86712268 10.89895483 10.93084508 10.96279425 10.99480313
```

```
 11.02687254 11.05900325 11.09119607 11.12345176 11.15577111 11.18815488
 11.22060383 11.25311872 11.2857003  11.31834931 11.35106648 11.38385254
 11.41670822 11.44963422 11.48263126 11.51570004 11.54884124 11.58205556
 11.61534367 11.64870625 11.68214396 11.71565744 11.74924736 11.78291436
 11.81665906 11.8504821  11.88438408 11.91836563 11.95242734 11.9865698
 12.02079361 12.05509933 12.08948754 12.1239588  12.15851367]
```

Galat Relatif

```
galat_relatif = [] # Variabel kosong yang akan diisi hasil galat relatif

for i in range(int(n)):
  galat_relatif.append(((analitik[i]-numerik[i])/analitik[i])*100) # Memasukkan nilai galat relatif ke dalam variabel "galat_relatif"


galat_relatif_kolom = np.array(galat_relatif).T # Mentranspose variabel "galat_relatif" agar mudah dilihat outputnya
print(galat_relatif_kolom)
```

```
[0.00000000e+00 1.84569026e-06 3.67906499e-06 5.50015322e-06
 7.30898286e-06 9.10558072e-06 1.08899725e-05 1.26621830e-05
 1.44222360e-05 1.61701540e-05 1.79059592e-05 1.96296724e-05
 2.13413139e-05 2.30409029e-05 2.47284581e-05 2.64039973e-05
 2.80675377e-05 2.97190958e-05 3.13586875e-05 3.29863280e-05
 3.46020320e-05 3.62058137e-05 3.77976868e-05 3.93776645e-05
 4.09457594e-05 4.25019840e-05 4.40463502e-05 4.55788695e-05
 4.70995532e-05 4.86084123e-05 5.01054575e-05 5.15906990e-05
 5.30641472e-05 5.45258119e-05 5.59757031e-05 5.74138304e-05
 5.88402032e-05 6.02548310e-05 6.16577232e-05 6.30488890e-05
 6.44283376e-05 6.57960783e-05 6.71521203e-05 6.84964729e-05
 6.98291452e-05 7.11501467e-05 7.24594868e-05 7.37571750e-05
 7.50432211e-05 7.63176348e-05 7.75804260e-05 7.88316050e-05
 8.00711819e-05 8.12991674e-05 8.25155723e-05 8.37204074e-05
 8.49136841e-05 8.60954138e-05 8.72656085e-05 8.84242801e-05
 8.95714412e-05 9.07071045e-05 9.18312830e-05 9.29439903e-05
 9.40452401e-05 9.51350466e-05 9.62134245e-05 9.72803887e-05
 9.83359546e-05 9.93801380e-05 1.00412955e-04 1.01434423e-04
 1.02444558e-04 1.03443379e-04 1.04430903e-04 1.05407149e-04
 1.06372136e-04 1.07325883e-04 1.08268412e-04 1.09199741e-04
 1.10119893e-04 1.11028888e-04 1.11926750e-04 1.12813501e-04
 1.13689164e-04 1.14553762e-04 1.15407321e-04 1.16249865e-04
 1.17081420e-04 1.17902011e-04 1.18711666e-04 1.19510411e-04
 1.20298274e-04 1.21075284e-04 1.21841470e-04 1.22596861e-04
 1.23341487e-04 1.24075380e-04 1.24798570e-04 1.25511089e-04
 1.26212970e-04]
```

Tabel Gabungan

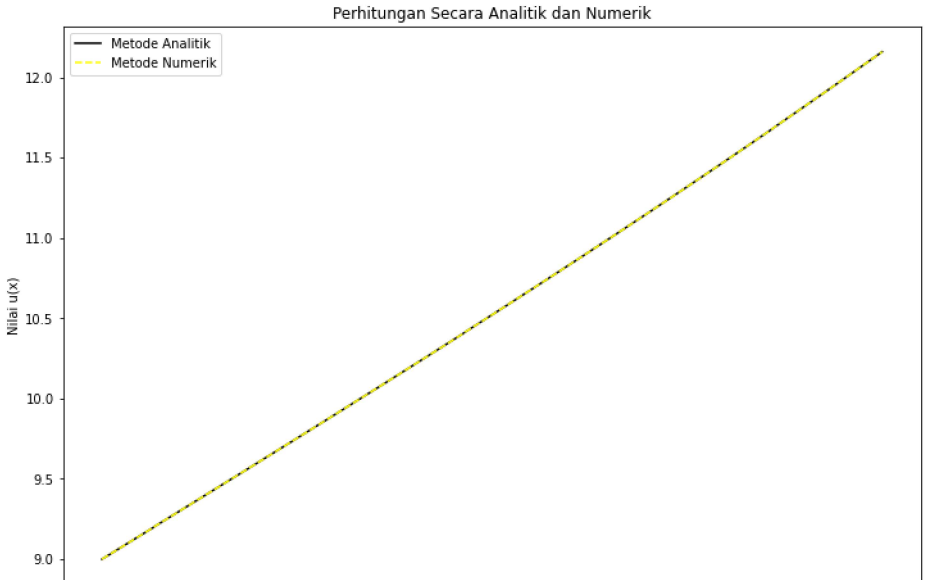
```
import pandas as pd


data = {'Analitik' : analitik, 'Numerik (Beda Hingga)' : numerik, 'Galat Relatif (%)' : galat_relatif}
df = pd.DataFrame(data=data) # Membentuk data frame dari array analitik, numerik, dan galat relatif
print(df)
```

```
      Analitik  Numerik (Beda Hingga)  Galat Relatif (%)
0     9.000000               9.000000           0.000000
1     9.030000               9.030000           0.000002
2     9.060001               9.060001           0.000004
3     9.090004               9.090004           0.000006
4     9.120011               9.120010           0.000007
..         ...                    ...                ...
96   12.020808              12.020794           0.000123
97   12.055114              12.055099           0.000124
98   12.089503              12.089488           0.000125
99   12.123974              12.123959           0.000126
100  12.158529              12.158514           0.000126

[101 rows x 3 columns]
```

Grafik

```
x_a = np.arange(a,b+delta_x,delta_x) # Menghasilkan angka dari 0 sampai 1 dengan beda antar angkanya 0.01
y_a = analitik # Menggunakan hasil variabel analitik

x_n = np.arange(a,b+delta_x,delta_x) # Menghasilkan angka dari 0 sampai 1 dengan beda antar angkanya 0.01
y_n = numerik # Menggunakan hasil variabel numerik


# Menghasilkan grafik
plt.figure(figsize=(12, 8))
plt.plot(x_a,y_a, label ="Metode Analitik",color="black")
plt.plot(x_n,y_n, label ="Metode Numerik", linestyle ="--",color="yellow")
plt.xlabel('Nilai x')
```

```
plt.ylabel('Nilai u(x)')
#plt.xlim([a,b])
#plt.ylim([0,13])
plt.title('Perhitungan Secara Analitik dan Numerik')
plt.legend()
plt.show()
```



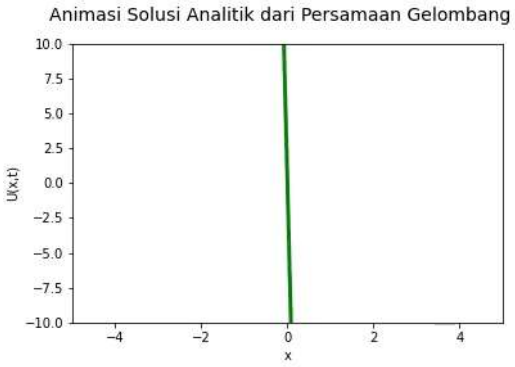
PDP

Animasi Analitik

Sumber: https://www.geeksforgeeks.org/how-to-save-matplotlib-animation/

```
# importing required libraries
from matplotlib import pyplot as plt
import numpy as np
import math as mt
import matplotlib.animation as animation
from IPython import display

# initializing a figure
fig = plt.figure()

# labeling the x-axis and y-axis
axis = plt.axes(xlim=(-5, 5),  ylim=(-10, 10))

# initializing a line variable
line, = axis.plot([], [], lw=3)

def animate(frame_number):
    x = np.linspace(-5, 5, 1000)

    # plots a graph
    y = ((-1/6)*(x)*(frame_number**3))
    line.set_data(x, y)
    line.set_color('green')
    return line,


anim = animation.FuncAnimation(fig, animate, frames=20,
                               interval=100, blit=True) # Naikkin frames biar tambah lambat
fig.suptitle('Animasi Solusi Analitik dari Persamaan Gelombang', fontsize=14)
plt.xlabel('x')
plt.ylabel('U(x,t)')

# converting to an html5 video
video = anim.to_html5_video()

# embedding for the video
html = display.HTML(video)

# draw the animation
```

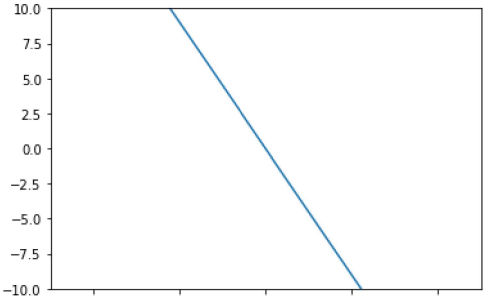
```
display.display(html)
plt.close()
```



2D Analitik

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

t = int(input('Masukkan waktu (t) (bilangan bulat) yang ingin diperhitungkan: '))

axis = plt.axes(xlim=(-5, 5),  ylim=(-10, 10))
x = np.linspace(-5, 5, 1000)

y = ((-1/6)*(x)*(t)**3)
plt.plot(x, y)
print('Grafik Solusi Analitik pada saat t = ', t, 'detik')
```

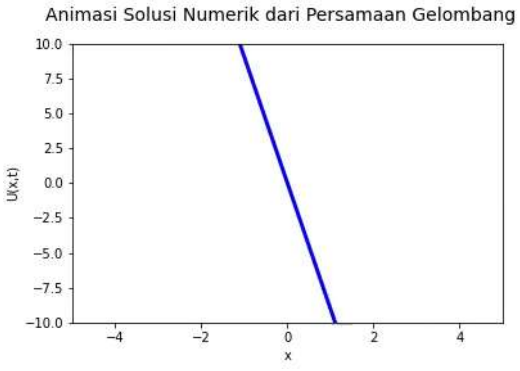
```
Masukkan waktu (t) (bilangan bulat) yang ingin diperhitungkan: 3
Grafik Solusi Analitik pada saat t =  3 detik
```



Animasi Numerik

```
from matplotlib import pyplot as plt
import numpy as np
import math as mt
import matplotlib.animation as animation
from IPython import display

# Numerik
c=1**2

batas_bawah = 0
batas_atas_l = 1
batas_atas_t = 1

#Diskritisasi
dx = 0.05
dt = 0.05

xx = np.arange(batas_bawah,batas_atas_l,dx)
tt = np.arange(batas_bawah,batas_atas_t,dt)

nx = len(xx)
nt = len(tt)

r =c*dt**2/dx**2

u=np.zeros((nx,nt))
```

```
# iterasi
for TT in range(1,nt-1):
    for XX in range(1,nx-1):
        u[XX,TT+1]=r*(u[XX+1,TT]-2*u[XX,TT]+u[XX-1,TT])+2*u[XX,TT]-u[XX,TT-1]-dt**2*XX*TT

# Puncak
maks = 0
for TT in range(1,nt):
    for XX in range(1,nx):
        if maks < u[XX,TT]:
            maks = u[XX,TT]
            x = xx[XX]
            t = tt[TT]

# initializing a figure
fig = plt.figure()

# labeling the x-axis and y-axis
axis = plt.axes(xlim=(-5, 5),  ylim=(-10, 10))

# initializing a line variable
line, = axis.plot([], [], lw=3)

def animate(frame_number):
    x = np.linspace(-5, 5, 1000)

    # plots a graph
    #y = ((-1/6)*(x)*(frame_number**3))
    U = []
    for i in range(0,20):
      U.append(i)
    y = -U[frame_number]*x
    line.set_data(x, y)
    line.set_color('blue')
    return line,


anim = animation.FuncAnimation(fig, animate, frames=20,
                               interval=100, blit=True) # Naikkin frames biar tambah lambat
fig.suptitle('Animasi Solusi Numerik dari Persamaan Gelombang', fontsize=14)
plt.xlabel('x')
plt.ylabel('U(x,t)')

# converting to an html5 video
video = anim.to_html5_video()

# embedding for the video
html = display.HTML(video)

# draw the animation
display.display(html)
plt.close()
```

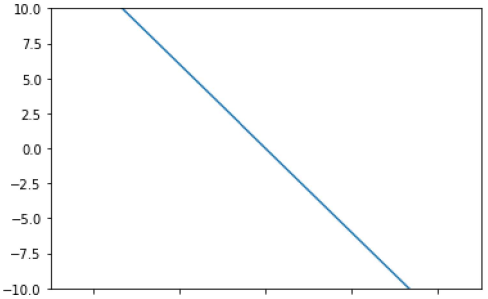


2D Numerik

```
from matplotlib import pyplot as plt
import numpy as np
# Numerik
c=1**2

batas_bawah = 0
batas_atas_l = 1
batas_atas_t = 1

#Diskritisasi
dx = 0.05
dt = 0.05
```

```
xx = np.arange(batas_bawah,batas_atas_l,dx)
tt = np.arange(batas_bawah,batas_atas_t,dt)

nx = len(xx)
nt = len(tt)

r =c*dt**2/dx**2

u=np.zeros((nx,nt))

# iterasi
for TT in range(1,nt-1):
    for XX in range(1,nx-1):
        u[XX,TT+1]=r*(u[XX+1,TT]-2*u[XX,TT]+u[XX-1,TT])+2*u[XX,TT]-u[XX,TT-1]-dt**2*XX*TT

# Puncak
maks = 0
for TT in range(1,nt):
    for XX in range(1,nx):
        if maks < u[XX,TT]:
            maks = u[XX,TT]
            x = xx[XX]
            t = tt[TT]

t = int(input('Masukkan waktu (t) (bilangan bulat) yang ingin diperhitungkan: '))

x = np.linspace(-5, 5, 1000)

    # plots a graph
    #y = ((-1/6)*(x)*(frame_number**3))


axis = plt.axes(xlim=(-5, 5),  ylim=(-10, 10))
U = []
for i in range(0,20):
  U.append(i)
y = -U[t]*x
plt.plot(x, y)
print('Grafik Solusi Numerik pada saat t = ', t, 'detik')
```

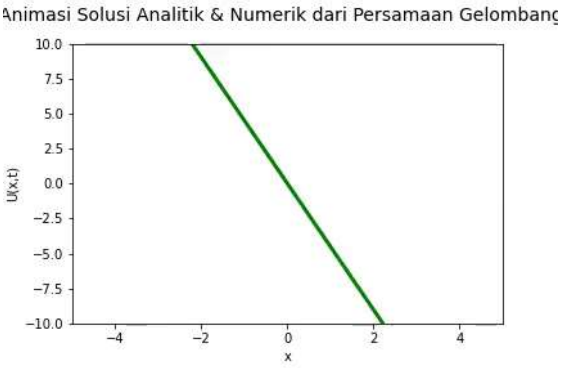
```
Masukkan waktu (t) (bilangan bulat) yang ingin diperhitungkan: 3
Grafik Solusi Numerik pada saat t =  3 detik
```

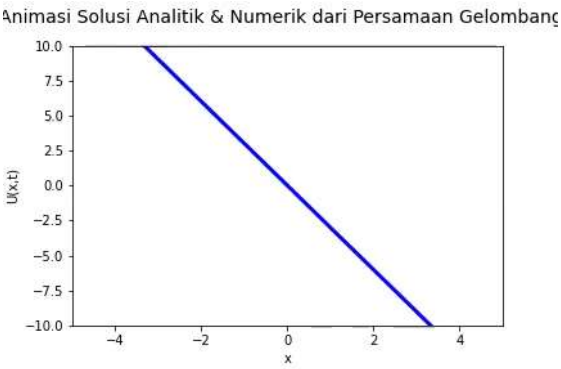


Animasi Gabungan

```
# importing required libraries
from matplotlib import pyplot as plt
import numpy as np
import math as mt
import matplotlib.animation as animation
from IPython import display

# initializing a figure
fig = plt.figure()

# labeling the x-axis and y-axis
axis = plt.axes(xlim=(-5, 5),  ylim=(-10, 10))

# initializing a line variable
line, = axis.plot([], [], lw=3)

def animate_1(frame_number):
    x = np.linspace(-5, 5, 1000)

    # plots a graph
    y = ((-1/6)*(x)*(frame_number**3))
    line.set_data(x, y)
    line.set_color('green')
```

```
    return line,

def animate_2(frame_number):
    x = np.linspace(-5, 5, 1000)

    # plots a graph
    #y = ((-1/6)*(x)*(frame_number**3))
    U = []
    for i in range(0,20):
      U.append(i)
    y = -U[frame_number]*x
    line.set_data(x, y)
    line.set_color('blue')
    return line,


anim1 = animation.FuncAnimation(fig, animate_1, frames=20,
                                interval=100, blit=True) # Naikkin frames biar tambah lambat


anim2 = animation.FuncAnimation(fig, animate_2, frames=20,
                                interval=100, blit=True) # Naikkin frames biar tambah lambat
fig.suptitle('Animasi Solusi Analitik & Numerik dari Persamaan Gelombang', fontsize=14)
plt.xlabel('x')
plt.ylabel('U(x,t)')

# converting to an html5 video
video1 = anim1.to_html5_video()
video2 = anim2.to_html5_video()
# embedding for the video
html1 = display.HTML(video1)
html2 = display.HTML(video2)
# draw the animation
display.display(html1)
display.display(html2)
plt.close()
```





Array Analitik dan Numerik

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


t1 = int(input('Masukkan waktu (t) (bilangan bulat) yang ingin diperhitungkan: '))
```

```
    Masukkan waktu (t) (bilangan bulat) yang ingin diperhitungkan: 3
```

Analitik

```
#Analitik
```

```
array_a = []

for i in range(0,20):
  x = np.linspace(-5, 5, 20)
  y = ((-1/6)*(x)*(i)**3)
  array_a.append(y)

print(array_a[t1]) #Nilai-nilai x ketika t1
```

```
[ 22.5         20.13157895  17.76315789  15.39473684  13.02631579
  10.65789474   8.28947368   5.92105263   3.55263158   1.18421053
  -1.18421053  -3.55263158  -5.92105263  -8.28947368 -10.65789474
 -13.02631579 -15.39473684 -17.76315789 -20.13157895 -22.5       ]
```

Numerik

```
#Numerik
c=1**2

batas_bawah = 0
batas_atas_l = 1
batas_atas_t = 1

#Diskritisasi
dx = 0.05
dt = 0.05

xx = np.arange(batas_bawah,batas_atas_l,dx)
tt = np.arange(batas_bawah,batas_atas_t,dt)

nx = len(xx)
nt = len(tt)

r =c*dt**2/dx**2

u=np.zeros((nx,nt))

# iterasi
for TT in range(1,nt-1):
    for XX in range(1,nx-1):
        u[XX,TT+1]=r*(u[XX+1,TT]-2*u[XX,TT]+u[XX-1,TT])+2*u[XX,TT]-u[XX,TT-1]-dt**2*XX*TT

# Puncak
maks = 0
for TT in range(1,nt):
    for XX in range(1,nx):
        if maks < u[XX,TT]:
            maks = u[XX,TT]
            x = xx[XX]
            t = tt[TT]
#for i in range(0,20):
    #U.append(u[i])
#for i in range(0,20):
    #y.append(-U[i]*x)
print(u[t1]) #Nilai-nilai x ketika t1
```

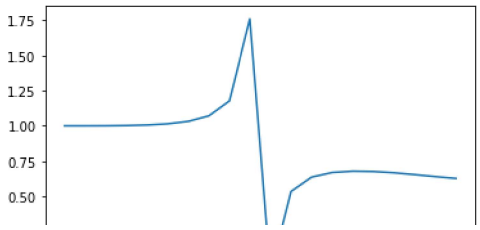
```
[ 0.0000e+00  0.0000e+00 -7.5000e-03 -3.0000e-02 -7.5000e-02 -1.5000e-01
 -2.6250e-01 -4.2000e-01 -6.3000e-01 -9.0000e-01 -1.2375e+00 -1.6500e+00
 -2.1450e+00 -2.7300e+00 -3.4125e+00 -4.2000e+00 -5.1000e+00 -6.1200e+00
 -7.2200e+00 -8.3600e+00]
```

Galat Relatif

Masukin t satu2

```
# Galat
galat = []
for i in range(0,20):
  galat.append((array_a[i]-u[i])/array_a[i])
print(galat[t1])

a_x = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]
a_y = galat[t1] # Galat ketika t nya 2
plt.plot(a_x, a_y)
```

```
[ 1.          1.          1.00042222  1.00194872  1.00575758  1.01407407
  1.03166667  1.07093333  1.17733333  1.76       -0.045       0.53555556
  0.63773333  0.67066667  0.67981481  0.67757576  0.66871795  0.65546667
  0.64135948  0.62844444]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: RuntimeWarning: invalid value encountered in true_divide
  after removing the cwd from sys.path.
[<matplotlib.lines.Line2D at 0x7f97741158d0>]
```

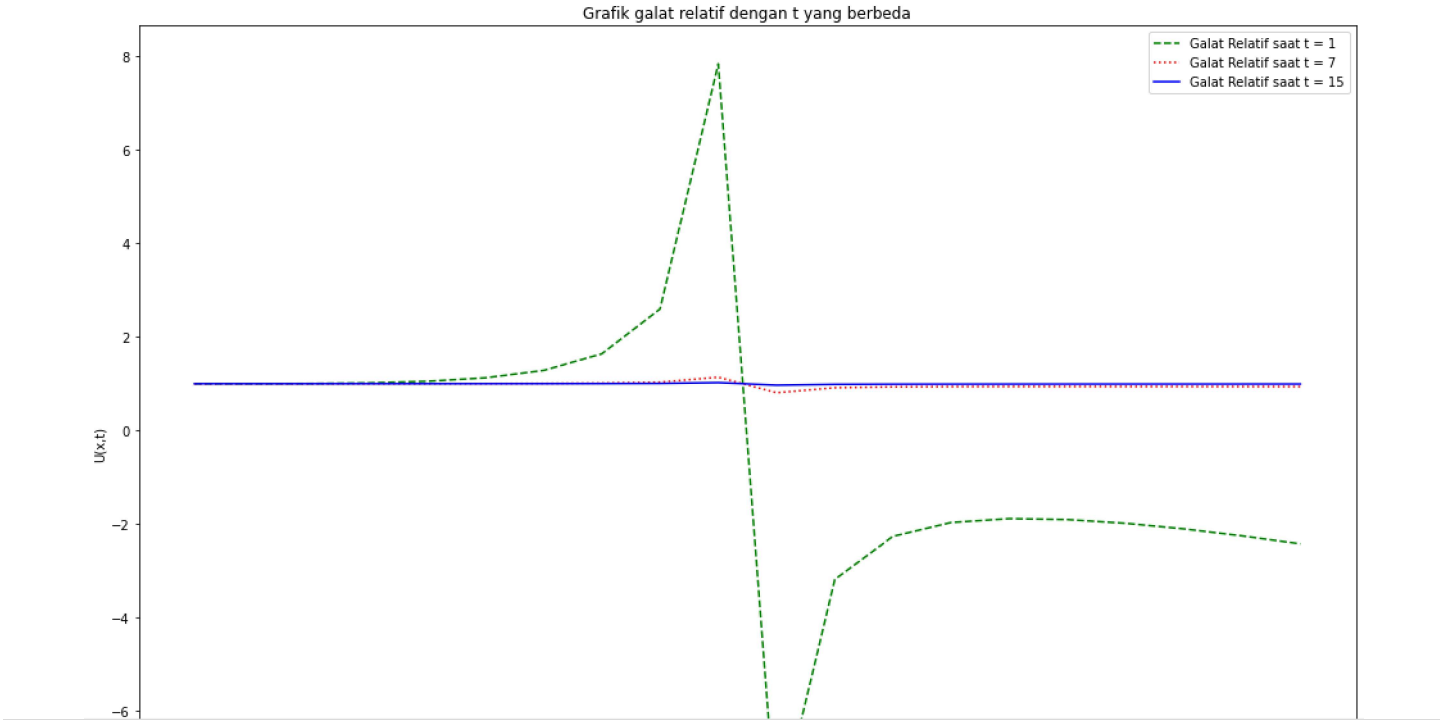


Grafik Galat Combine t=1 & t=15


```
galat1 = []
for i in range(0,20):
  galat.append((array_a[i]-u[i])/array_a[i])

a_x = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]
a_y1 = galat[1]

galat7 = []
for i in range(0,20):
  galat.append((array_a[i]-u[i])/array_a[i])

a_y7 = galat[7]

galat15 = []
for i in range(0,20):
  galat.append((array_a[i]-u[i])/array_a[i])

a_y15 = galat[15]
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: RuntimeWarning: invalid value encountered in true_divide
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: RuntimeWarning: invalid value encountered in true_divide
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:16: RuntimeWarning: invalid value encountered in true_divide
  app.launch_new_instance()
```

```
plt.figure(figsize=(17, 12))
plt.plot(a_x,a_y1, label ="Galat Relatif saat t = 1", linestyle ="--",color="green")
plt.plot(a_x,a_y7, label ="Galat Relatif saat t = 7", linestyle ="dotted",color="red")
plt.plot(a_x,a_y15, label ="Galat Relatif saat t = 15", linestyle ="solid",color="blue")
plt.xlabel('x')
plt.ylabel('U(x,t)')
plt.title('Grafik galat relatif dengan t yang berbeda')
plt.legend()
plt.show()
```



✓ 0s completed at 9:32 AM