Python_quiz_3

October 21, 2020

1 2020 Differential Equation Python Quiz 3

1.1 Question 1

1. At time t = 0, a tank contains Y_0 lb of salt dissolved in a gal of water. Assume that water containing b lb of salt/gal is entering tank at rate of r gal/min, and leaves at same rate. At time tim, $Y(t)$ is within $p\%$ of the limit value Y_∞}
   Please write a function $y_t, y_\infty, tim = \text{quiz\_3}(Y_0, a, b, r, p)$, where y_t is the solution of the first order differential equation.
   Your function name should be "*quiz_3*", it should take 5 parameters: (Y_0, a, b, r, p), and it should return 3 things: (y_t, y_∞, tim)

a) Solve ODE y_t **(15 points)**

b) Find the limit value Y_∞ **(15 points)**

c) Find the time tim **(15 points)**

d) Plot $y(t)$ **(15 points)**

1.1.1 Some hints for question 1

- Don't define symbolic function y with respect to value t during the declaration. Instead use syntax **y(t)** later on. Otherwise you won't be able to use **dsolve** with initial conditions.
- For initial condition use **ics** parameter inside **dsolve** function. It takes *dictionary* as input. So use this as input "**y(0): 0**". But **remember that it takes input as dictionary**, so make sure you parse it properly.
- For evaluating limit use **sympy limit** function. For infinity use following syntax: "**oo**". **oo** operator is still a part of sympy so make sure to use it properly.
- For plotting use **sympy plot** function. Use *xlim* and *ylim* parameters to define proper range. They take input in a form of tuple.

[14]:
```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
import sympy as sp
```

```python
from sympy.interactive import printing
printing.init_printing(use_latex=True) # For better representation
from quiz3 import quiz_3
```

[18]:

```python

```

[19]:

```python
a, b, r, y0, p = 80, 0.4, 35, 10, 0.03
```

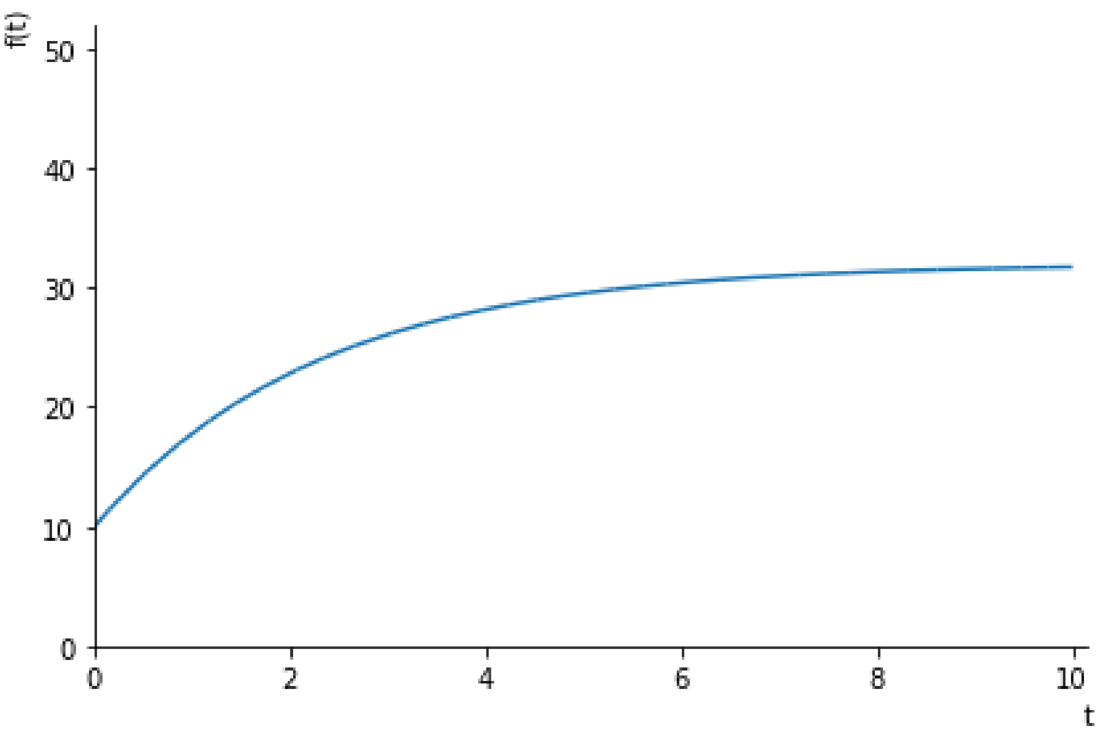
For finding time use the code defined in the cell below.
y_inf is the limit of your solution as it approaches infinity.

[ ]:

```python
if y0 >= y_inf:
    tim = -a/r * sp.log(((1 + p) * y_inf - a * b)/ (y0 - a * b))
else:
    tim = -a/r * sp.log(((1 - p) * y_inf - a * b)/ (y0 - a * b))
```

[20]:

```python
y_t, y_infnt, time = quiz_3(y0, a, b, r, p)
```



[21]:

```python
y_t
```

[21]: $y(t) = 32.0 - 22.0e^{-0.4375t}$

[22]:

```python
y_infnt
```

[22]:

32.0

[23]:
```
time
```

[23]:

7.15854730943673

1.2 Question 2

2. Suppose you have a differential equation as follows: $y' = 4y + 5t$
   Plot the direction field with x and y between -1 and 1, and use Euler's method with increased interval h = 0.01 to plot two curves with *proper range*

a) $y(0) = -4/16$ (figure 1) **(15 points)**

b) $y(0) = -5/16$ (figure 2) **(15 points)**

1.2.1 Some hints for question 2

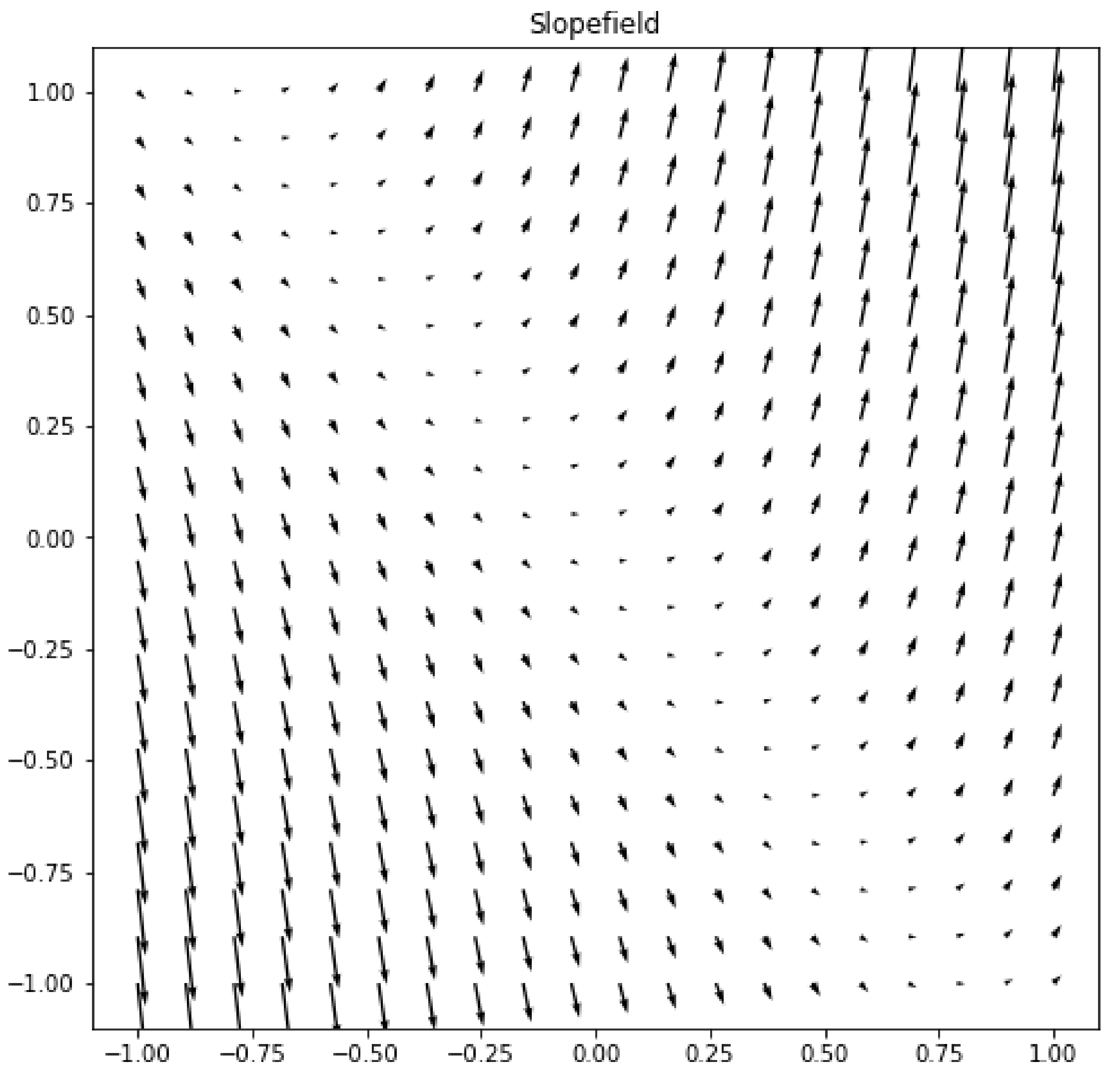
- Use of **numpy arange** function is preffered when defining range with certain interval.

[51]:
```
from scipy.integrate import odeint
```

First check if you can plot slopefield of this differential equation

[96]:
```

```

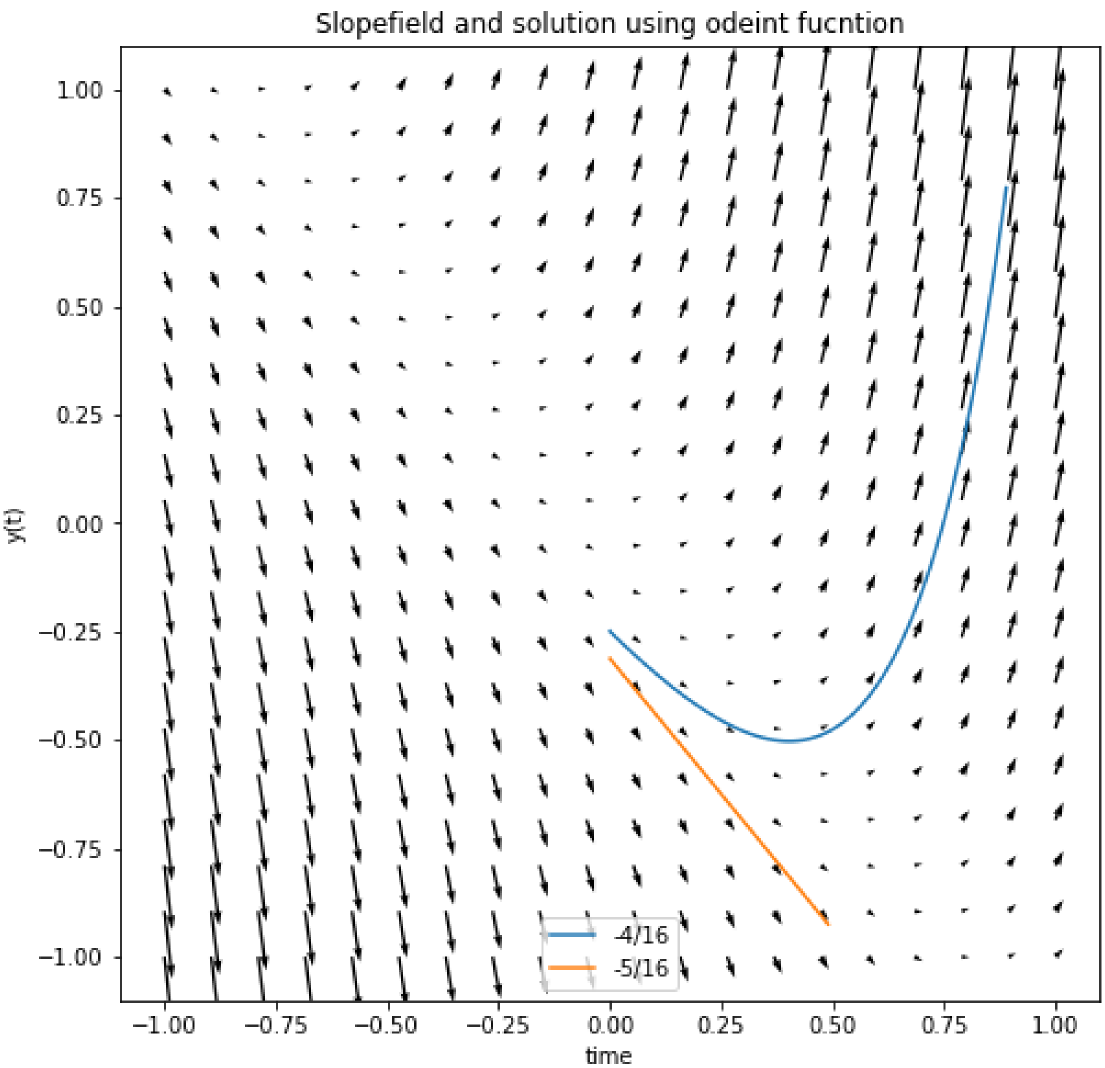


Then find the solution using **odeint** function. Remember to use proper range.

[97]:

```

```

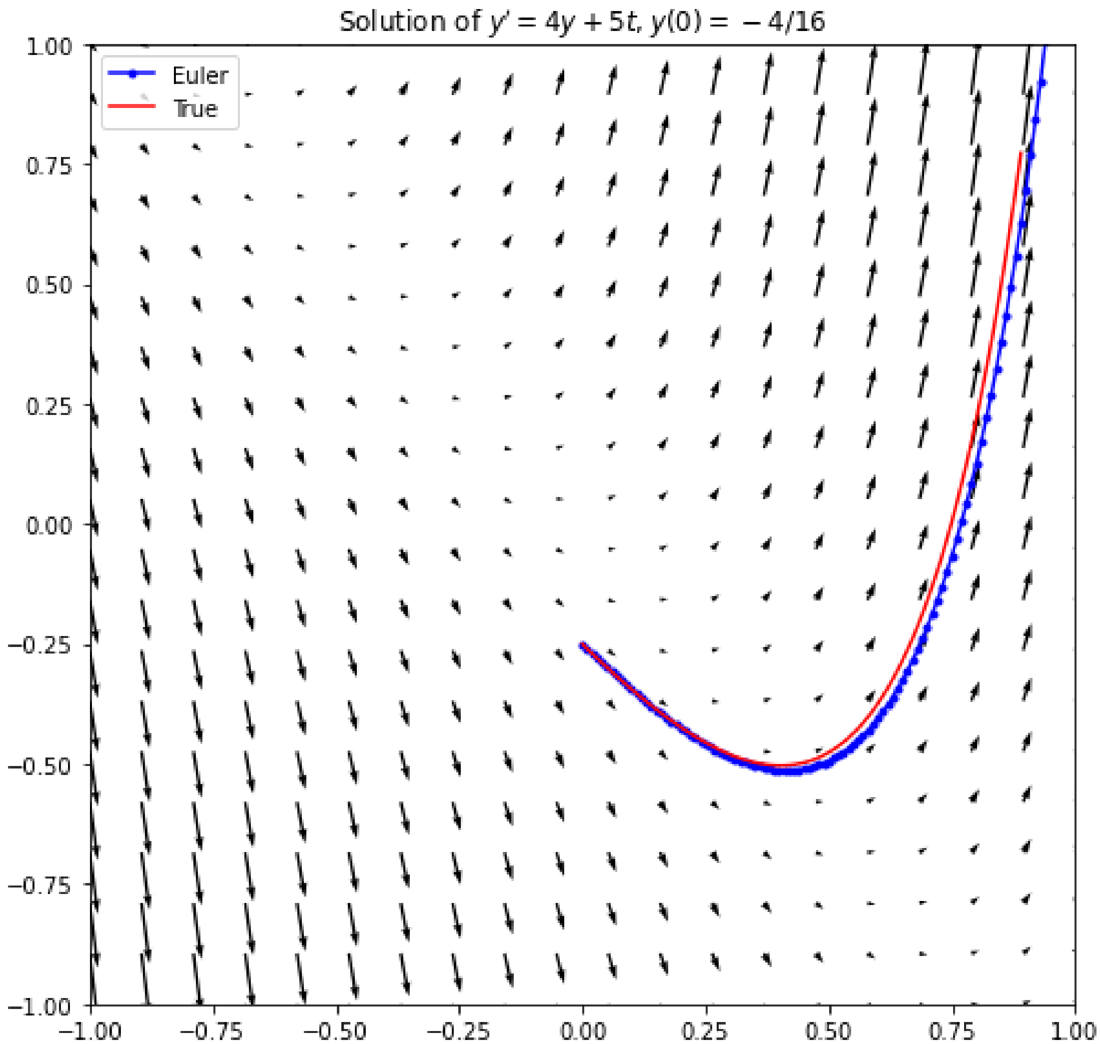


Then define your **custom function** that will find **numerical approximation** of the solution using Euler's method.
After you've done with that. Check your function's result against the **odeint** solution
In the end we will only check the figures of your custom function against **odeint** method. Previous steps are meant to help you debug your program.

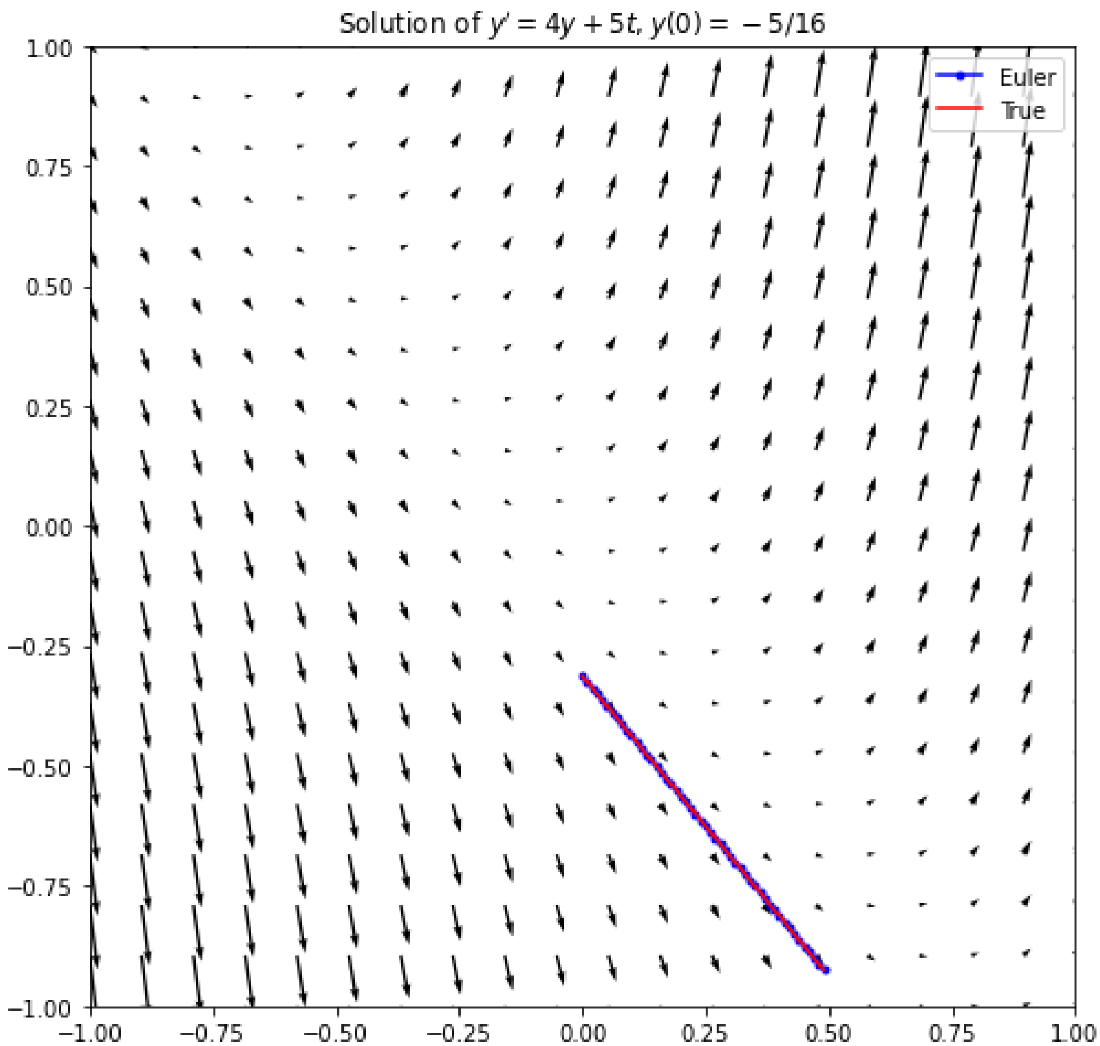
[94]:

```

```


Solution of $y' = 4y + 5t$, $y(0) = -4/16$
Euler
True
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

[95]:



1.3 Question 3

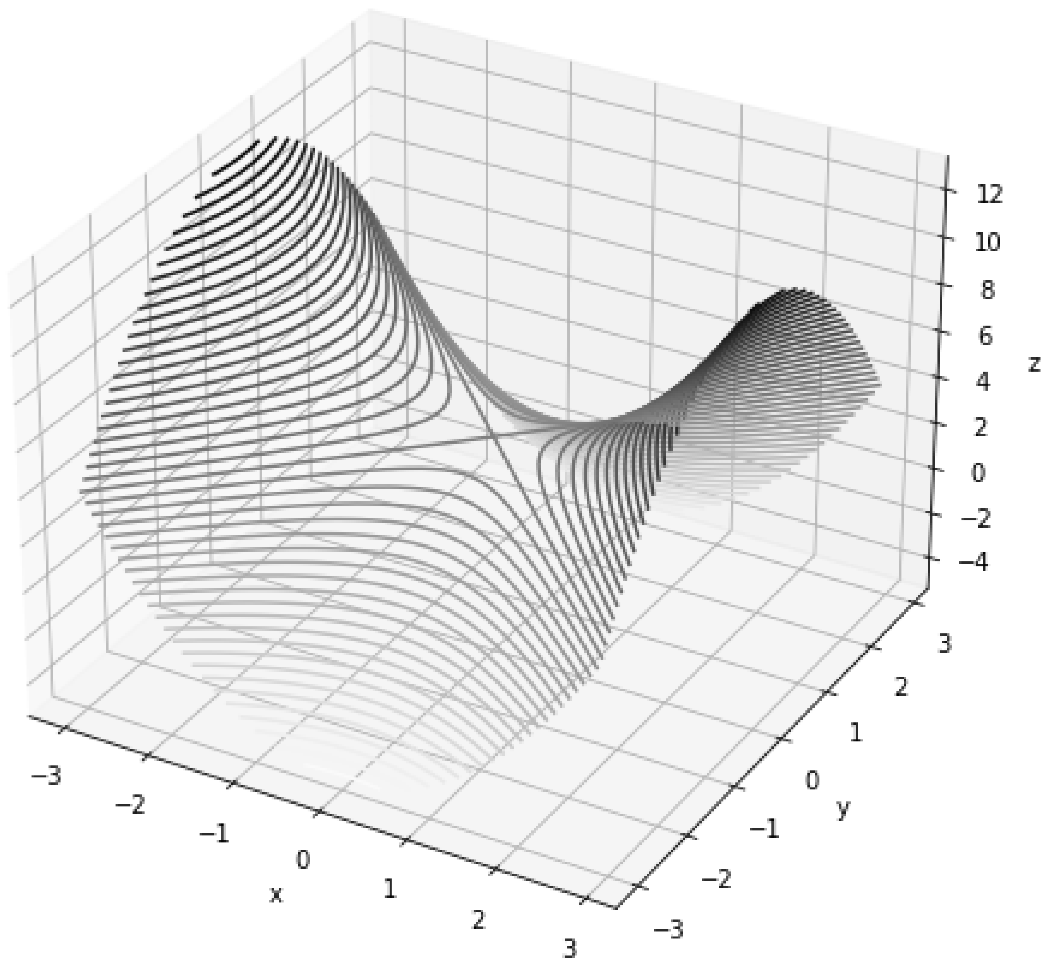
3. Complex Parabola: For $z = a^2 + 4, \quad a = x + iy$, is a complex number

a) Plot three 3D figures with axis x, y, and $\mathbb{R}$(z) **(10 points)**
Refer to this website for help with plotting
For complex numbers reference you can take a look at this link

[44]:
```
x = np.arange(-3, 3, 0.1)
y = np.arange(-3, 3, 0.1)
```

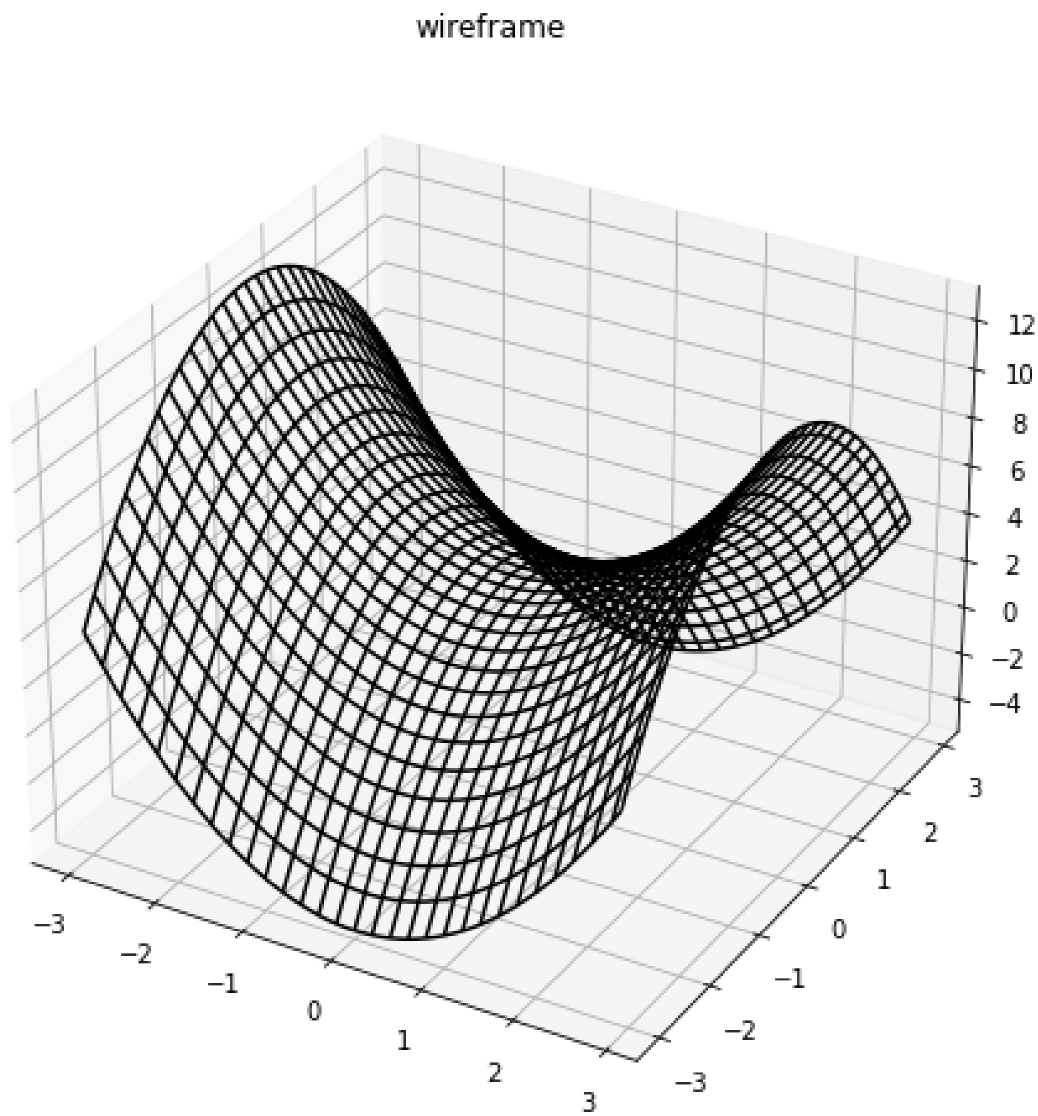
[35]:
```
```

[35]: Text(0.5, 0, 'z')



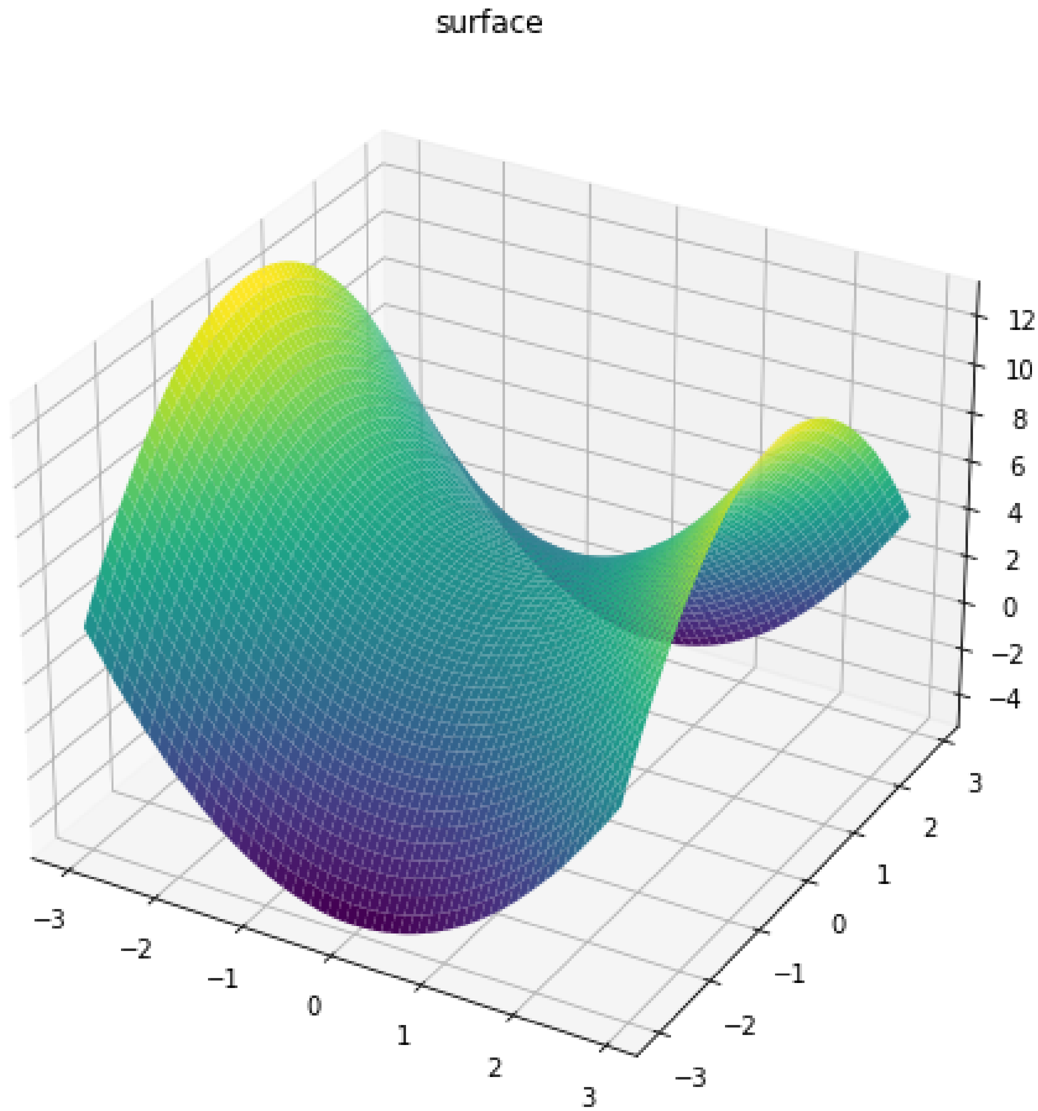
[36]:

[36]: Text(0.5, 0.92, 'wireframe')



[32]:

[32]: Text(0.5, 0.92, 'surface')


surface
12
10
8
6
4
2
0
−2
−4
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3