Slope Fields.

Slope fields are used to visualize the shape of different solutions to a differential equation. This is particularly useful when a differential equation does not have an analytical solution. Here, we are given the system of differential equations:

$$\frac{dB}{dt} = r_b B(t) - kB(t)P(t)$$

and

$$\frac{dP}{dt} = -d_p P(t) + kB(t)P(t)$$

where r_b, d_p and k are constants.

We are asked to plot a slope field for B vs P. Let us take P as the $y - axis$ and B as the $x - axis$.

Using the chain rule, we have

$$\frac{dP}{dB} = \frac{\frac{dP}{dt}}{\frac{dB}{dt}} = \frac{-d_p P + kBP}{r_b B - kBP}$$

Now, let us run a code which will output our slope as a vector. When our slope is m, the corresponding vector is $\hat{i} + m \cdot \hat{j}$. We will consider $r_b = 0.8$, $d_p = 1.6$ and $k = 1.2$.

Let us find the slope fields at the grid $B \times P \in [-0.5, 0.5] \times [-0.5, 0.5]$, at every 0.2 interval. At each point, we will output the following:

1. slope m
2. direction $\hat{i} + m\hat{j}$
3. the $\arctan(m)$

This program can be cuztomized with different grid sizes, partitions on each axis , and different values of r_b, d_p, k.

```
import math

# Function to calculate the slope field at a given point (B, P)
def slope(B, P, r_b, d_p, k):
  numerator = (d_p * P) + (k * B * P)
  denominator = (r_b * B) - (k * B * P)
  try:
    return numerator / denominator
  except ZeroDivisionError:
    return math.nan
```

```
import numpy as np

def coordinates(lower_limit, upper_limit, partitions):
```

```
  gap = (upper_limit - lower_limit)/partitions
  output = []
  for i in range(0, partitions+1):
    output.append(lower_limit + i*gap)
  return output
```

```
from tabulate import tabulate
```

```
def print_slope_field(
    lower_limit_b = -0.5,
    upper_limit_b = 0.5,
    partitions_b = 5,
    lower_limit_p = -0.5,
    upper_limit_p = 0.5,
    partitions_p = 5,
    r_b = 0.8,
    d_p = 1.6,
    k = 1.2):

    output = []
    b_points = coordinates(lower_limit_b, upper_limit_b, partitions_b)
    p_points = coordinates(lower_limit_p, upper_limit_p, partitions_p)
    for b in b_points :
        for p in p_points:
            slp = slope(b, p, r_b, d_p, k)
            direction = (1, slp)
            # angle in degree
            angle = math.atan(slp)*180.0/math.pi
            output.append(["{0:.2f}".format(b), "{0:.2f}".format(p), direction, "{0
    print(tabulate(output, headers = ['B', 'P', 'vector', 'angle wrt B axis in degr
```

```
print_slope_field()
```

```
   B     P  vector                       angle wrt B axis in degree
----  ----  -------------------------  ----------------------------
-0.5  -0.5  (1, 0.7142857142857143)                           35.54
-0.5  -0.3  (1, 0.5172413793103448)                           27.35
-0.5  -0.1  (1, 0.21739130434782605)                          12.26
-0.5   0.1  (1, -0.2941176470588238)                         -16.39
-0.5   0.3  (1, -1.3636363636363638)                         -53.75
-0.5   0.5  (1, -4.999999999999998)                          -78.69
-0.3  -0.5  (1, 1.4761904761904765)                           55.89
-0.3  -0.3  (1, 1.0689655172413794)                           46.91
-0.3  -0.1  (1, 0.4492753623188406)                           24.19
-0.3   0.1  (1, -0.6078431372549026)                         -31.29
-0.3   0.3  (1, -2.8181818181818197)                         -70.46
-0.3   0.5  (1, -10.333333333333336)                         -84.47
-0.1  -0.5  (1, 5.285714285714288)                            79.29
-0.1  -0.3  (1, 3.8275862068965525)                           75.36
-0.1  -0.1  (1, 1.6086956521739133)                           58.13
-0.1   0.1  (1, -2.1764705882352966)                         -65.32
-0.1   0.3  (1, -10.090909090909095)                         -84.34
-0.1   0.5  (1, -37.0)                                       -88.45
```

```
0.1  -0.5  (1, -6.142857142857138)                       -80.75
0.1  -0.3  (1, -4.448275862068962)                       -77.33
0.1  -0.1  (1, -1.8695652173913024)                      -61.86
0.1   0.1  (1, 2.5294117647058827)                        68.43
0.1   0.3  (1, 11.727272727272721)                        85.13
0.1   0.5  (1, 42.999999999999964)                        88.67
0.3  -0.5  (1, -2.3333333333333335)                      -66.8
0.3  -0.3  (1, -1.6896551724137925)                      -59.38
0.3  -0.1  (1, -0.7101449275362317)                      -35.38
0.3   0.1  (1, 0.960784313725491)                         43.85
0.3   0.3  (1, 4.454545454545455)                         77.35
0.3   0.5  (1, 16.33333333333333)                         86.5
0.5  -0.5  (1, -1.5714285714285716)                      -57.53
0.5  -0.3  (1, -1.1379310344827585)                      -48.69
0.5  -0.1  (1, -0.4782608695652173)                      -25.56
0.5   0.1  (1, 0.6470588235294124)                        32.91
0.5   0.3  (1, 3.0000000000000004)                        71.57
0.5   0.5  (1, 10.999999999999996)                        84.81
```

✓ 0s completed at 22:58