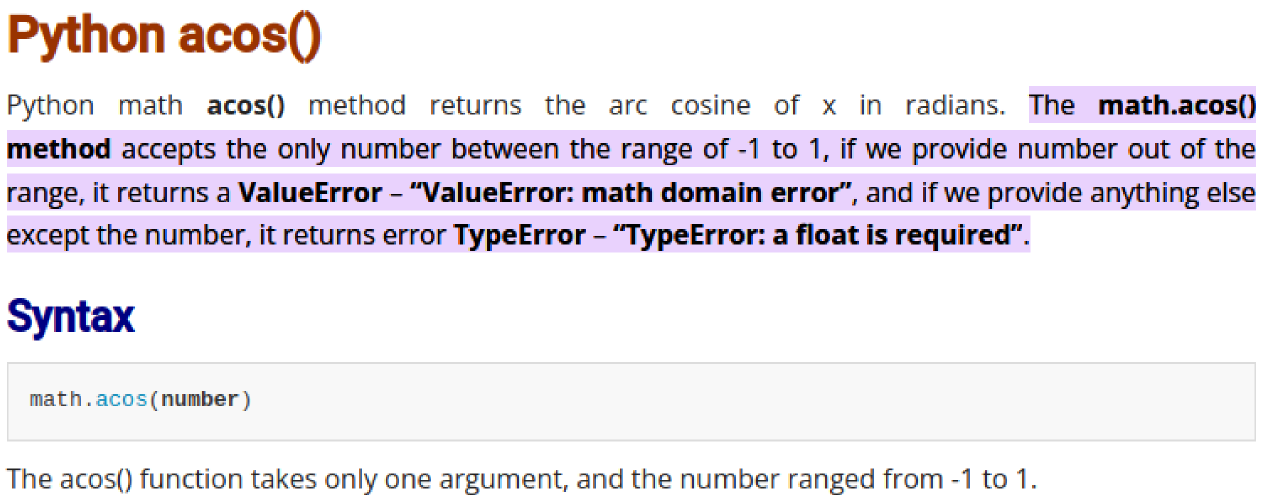
April 5, 2022 YY: Calculation Error debugging note

Error

```
  File "/home/yusuke/Documents/CTI_One_Corp/2_Work/Unity-Python/Unity-Python-OpenCV/PythonProgram/CAPP_Kart_4_Observation.py", line 94, in getAngle
    theta = math.acos(numv / denv)
ValueError: math domain error
```

The error happens if numv / denv is not between from -1 to 1.



Python acos()

Python math **acos()** method returns the arc cosine of x in radians. The **math.acos() method** accepts the only number between the range of -1 to 1, if we provide number out of the range, it returns a **ValueError – "ValueError: math domain error"**, and if we provide anything else except the number, it returns error **TypeError – "TypeError: a float is required"**.

Syntax

```
math.acos(number)
```

The acos() function takes only one argument, and the number ranged from -1 to 1.

https://appdividend.com/2020/02/11/python-acos-function-math-acos-in-python-example/#:~:text=x%20in%20radians.-,The%20math.,%3A%20a%20float%20is%20required%E2%80%9D.

Error Data
Point A(x,z) =(-25.80288,10.33637)
Point B(x,z) =(-9.802876,23.33637)
RobotPosition(x,z) = (-17.03665,17.45893)

Equation
v1x = Point A(x) - Point B(x)
v1y = Point A(z) - Point B(z)
v2x = Point A(x) - RobotPosition(x)
v2y = Point A(z) - RobotPosition(z)

numv = v1x*v2x + v1y*v2y
denv = √(v1x**2 + v1y**2) * √(v2x**2 + v2y**2)

num/dev =

Handcalcuation (Ubuntu calculator)
v1x = Point A(x) - Point B(x) = -25.80288 - (-9.802876)= −16.000004
v1y = Point A(z) - Point B(z) = 10.33637 - 23.33637= −13

v2x = Point A(x) - RobotPosition(x) = -25.80288 - (-17.03665) = −8.76623
v2y = Point A(z) - RobotPosition(z) = 10.33637 - 17.45893 = −7.12256

numv = v1x*v2x + v1y*v2y
= (−16.000004)*(−8.76623) + (−13) *(−7.12256)
= 232.852995065

denv = √(v1x**2 + v1y**2) * √(v2x**2 + v2y**2)
= √((−16.000004)**2 + (−13)**2) * √((−8.76623)**2 + (−7.12256)**2)
= √(256.000128 + 169) * √(76.846788413 +50.730860954)
= 20.615531233 * 11.295027639
= 232.852995069

num/dev = 232.852995065/232.852995069 is less than 1

Python Code (this part is from CAPP_Kart_4_Observation.py)

```
import math
import traceback

pointA = (-25.80288, 10.33637)
pointB = (-9.802876, 23.33637)
currentPosition = (-17.03665, 17.45893)

v1x = pointA[0] - pointB[0]
v1y = pointA[1] - pointB[1]

v2x = pointA[0] - currentPosition[0]
v2y = pointA[1] - currentPosition[1]

numv = v1x * v2x + v1y * v2y
denv = math.sqrt(v1x ** 2 + v1y ** 2) * math.sqrt(v2x ** 2 + v2y ** 2)
print("pointA[0]:", pointA[0], "pointA[1] :", pointA[1], "pointB[0]:", pointB[0], "pointB[1]:", pointB[1],
      "currentPosition[0]:", currentPosition[0], "currentPosition[1]:", currentPosition[1])
print("v1x:", v1x, "v1y:", v1y, "v2x:", v2x, "v2y:", v2y)
print("numv:", numv, "denv:", denv)

# 2022-03-22 YY Add to check if denv is zero to avoid division by zero
if denv == 0:
    theta = 0
else:
    # 2022-04-04 YY Add to check math domain error
    try:
        print("numv / denv:", numv / denv)
        theta = math.acos(numv / denv)
        theta = round(math.degrees(theta))
    except Exception as e:
        print("Error #####: ", e)
        print("pointA: ", pointA, " pointB:", pointB, " currentPosition:", currentPosition)
        print("Error #####: ", traceback.format_exc())

    # theta = math.acos(numv / denv)
```

Result

```
pointA[0]: -25.80288 pointA[1] : 10.33637
pointB[0]: -9.802876 pointB[1]: 23.33637
currentPosition[0]: -17.03665 currentPosition[1]: 17.45893

v1x: -16.000003999999997
v1y: -12.999999999999998
v2x: -8.766229999999997
v2y: -7.122559999999998

numv: 232.8529950649199
denv:  232.85299506491987
numv / denv: 1.0000000000000002 is greater than 1
```

Hand Calculation	Py
$v1x = -16.000004$	v1x: -16.000003999999997
$v1y = -13$	v1y: -12.999999999999998
$v2x = -8.76623$	v2x: -8.766229999999997
$v2y = -7.12256$	v2y: -7.122559999999998

There is a rounding error.

Solution

Add if statement set the result of “numv / denv” as 1 or -1

```
    if numvdenv > 1:
        numvdenv = 1
    elif numvdenv < -1:
        numvdenv = -1
```

Modified code

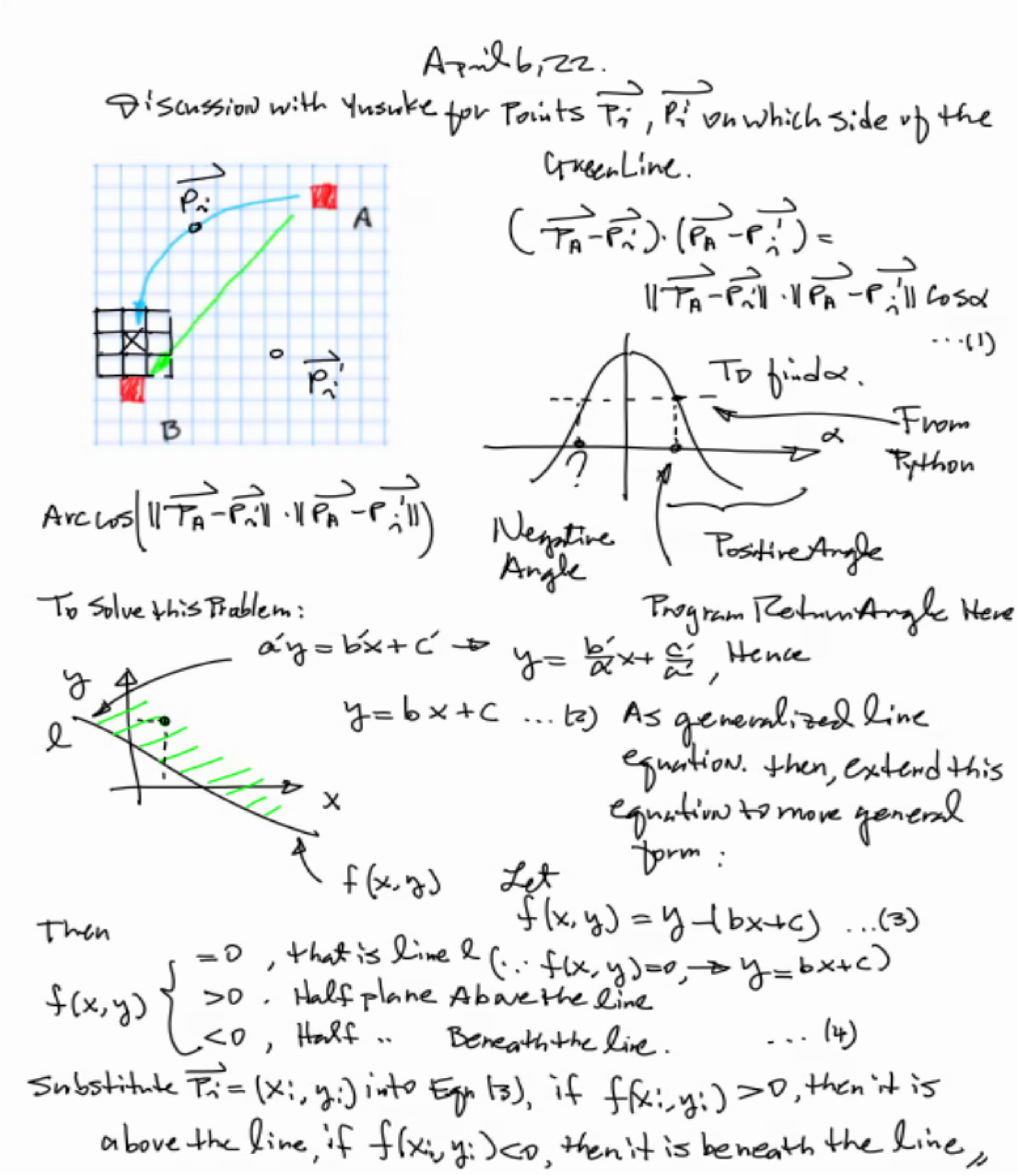
```
# 2022-04-04 YY Add to check math domain error
try:
    numvdenv = numv / denv
    print("numv / denv:", numv / denv)
    if numvdenv > 1:
        numvdenv = 1
    elif numvdenv < -1:
        numvdenv = -1

    theta = math.acos(numvdenv)
    theta = round(math.degrees(theta))
except Exception as e:
    print("Error #####: ", e)
    print("pointA: ", pointA, " pointB:", pointB, " currentPosition:", currentPosition)
    print("Error #####: ", traceback.format_exc())
```

if numvdenv > 1 - epsilon

(set epsilon = 0.00001)

if numvdenv < -1 + epsilon



April 6, 22.

Discussion with Yusuke for Points $\vec{P_i}$, $\vec{P_i'}$ on which side of the GreenLine.

$\vec{P_i}$
A
B
$\vec{P_i'}$

$$(\vec{P_A}-\vec{P_i})\cdot(\vec{P_A}-\vec{P_i'}) = \|\vec{P_A}-\vec{P_i}\|\cdot\|\vec{P_A}-\vec{P_i'}\|\cos\alpha$$ ...(1)

To find α.

From Python
α
?
Negative Angle
Positive Angle

$\mathrm{ArcCos}\left(\|\vec{P_A}-\vec{P_i}\|\cdot\|\vec{P_A}-\vec{P_i'}\|\right)$

Program Return Angle Here

To Solve this Problem:

$a'y = b'x + c' \rightarrow y = \frac{b'}{a'}x + \frac{c'}{a'}$, Hence

$y = bx + c$ ...(2) As generalized line equation. then, extend this equation to more general form:

Let

$f(x, y) = y - (bx + c)$ ...(3)

Then

$$f(x,y)\begin{cases} =0, & \text{that is line } l \ (\because f(x,y)=0, \rightarrow y = bx + c) \\ >0. & \text{Half plane Above the line} \\ <0, & \text{Half .. Beneath the line.} \end{cases}$$ ...(4)

Substitute $\vec{P_i} = (x_i, y_i)$ into Eqn (3), if $f(x_i, y_i) > 0$, then it is above the line, if $f(x_i, y_i) < 0$, then it is beneath the line,,