PROFESSOR DE MATEMÁTICA ONLINE
Revista eletrônica da Sociedade Brasileira de Matemática

PMO v.11, n.2, 2023
ISSN: 2319-023X
https://doi.org/10.21711/2319023x2023/pmo1118

Função polinomial do segundo grau com o Lumibot-CoppeliaSim

Cássio Lima Macedo Cleidinaldo Aguiar Souza

Resumo

Neste trabalho abordaremos funções polinomiais do segundo grau através do ambiente de simulação robótica, CoppeliaSim. Mais precisamente utilizaremos o Lumibot, que é um dos robôs presentes no ambiente de simulação CoppeliaSim. Todo esse processo faz parte do modo de aprendizagem STEM, em que problemas matemáticos são resolvidos através da interdisciplinaridade com outras áreas apresentadas aos estudantes da educação básica. Apresentaremos o passo a passo para a utilização do ambiente de simulação CoppeliaSim, assim como um código em linguagem Python que pode ser alterado de modo adequado, possibilitando a criação e resolução de novos problemas envolvendo funções polinomiais do segundo grau. O método utilizado neste trabalho evidencia como as novas profissões necessitam cada vez mais de criatividade matemática. Consequentemente, este trabalho pode despertar a criatividade matemática necessária aos estudantes de hoje, para que os mesmos possam ocupar os novos postos de trabalho.

Palavras-chave: Função Polinomial do Segundo Grau; Robótica; CoppeliaSim; Ensino; STEM.

Abstract

In this work we will approach second degree polynomial functions through the robotic simulation environment, CoppeliaSim. More precisely, we will use Lumibot, which is one of the robots present in the CoppeliaSim simulation environment. This whole process is part of the STEM learning mode, in which mathematical problems are solved through interdisciplinarity with other areas presented to basic education students. We will present step by step the use of the CoppeliaSim simulation environment, as well as a code in Python language that can be changed accordingly, allowing the creation and resolution of new problems involving second degree polynomial functions. The method used in this work shows how new professions increasingly need mathematical creativity. Consequently, this work can wake up the mathematical creativity needed by today's students, so that they can occupy new jobs.

Keywords: Polynomial Function of the Second Degree; Robotics; CoppeliaSim; Teaching; STEM.

1. Introdução

Em educação básica, um dos maiores desafios do ensino de matemática é apresentar aos alunos conteúdos que se complementam e que se desenvolvam, exigindo que o educando, para além de

SBM SOCIEDADE BRASILEIRA DE MATEMÁTICA

reproduzir fórmulas aparentemente isoladas, desenvolva a capacidade de ler, escrever, interpretar e produzir os elementos, durante o processo de alfabetização matemática, de maneira eficaz. Recentemente, (OLIVEIRA e SOUZA, 2020) apresentaram uma alternativa para esse desafio, abordando um conteúdo que inúmeras vezes é trabalhado em educação básica de forma abstrata, sem apresentar qualquer relação com outras áreas do conhecimento, ou até mesmo com outros conteúdos dentro da matemática, que são as matrizes.

Atualmente estamos diante da quarta revolução industrial, chamada de indústria 4.0, que é caracterizada pela digitalização e pela robótica no processo de produção, ou seja, é baseada em inovações tecnológicas digitais que alteram as interfaces entre o trabalho humano e os processos controlados por sistemas computacionais. Como consequência, novas profissões estão surgindo e algumas profissões de décadas estão desaparecendo. Isto é uma inerência das revoluções industriais, que influenciam para além da produção o sistema educacional dos países. A indústria 4.0 tem feito mudanças no sistema educacional, porém ao contrário das revoluções industriais anteriores, a nova indústria tem feito com que novas tecnologias sejam desenvolvidas em uma velocidade várias vezes à frente do ritmo das mudanças do sistema de educação. Isso tem feito com que vários países priorizem o processo de capacitação para atender à indústria 4.0.

A indústria 4.0 exige que os humanos tenham, além do conhecimento, habilidades para colaborar, resolver problemas, pensar criticamente e trabalhar em equipe (ROSA e OREY, 2018). É cada vez mais evidente que as novas tecnologias estão fundindo os mundos físico, digital e biológico, influenciando toda sociedade e modificando todas disciplinas. É um desafio fazer com que os estudantes de hoje tenham criatividade e habilidade necessárias para trabalhar com as novas tecnologias, e no futuro possam desenvolver novas tecnologias importantes para a sociedade. Uma alternativa para preparar gerações para os desafios que as novas profissões exigem, é o modo de aprendizagem STEM, que em inglês significa *Science, Technology, Engineering, and Mathematics.* Tal modo de aprendizagem nos anos de 1990 foi originalmente chamado pelo National Science Foundation (NSF) de SMET (Science, Mathematics, Engineering, and Technology). A sigla SMET foi reordenada em 2001 para a sigla STEM pela bióloga americana Judith Ramaley (TSUPROS, *et al.*, 2009).

O modo de aprendizagem STEM, é uma abordagem educacional que integra ciência, tecnologia, engenharia e matemática no processo de ensino e aprendizagem. O modo de aprendizagem STEM é uma interdisciplinaridade entre ciência, tecnologia, engenharia e matemática (BYBEE, 2013; GONZALEZ e KUENZI, 2012). Outra definição para STEM é o processo de aprendizagem através de problemas da vida real (YILDIRIM e SELVI, 2016). Além disso, o uso de STEM é adequado para o ensino e aprendizagem, desenvolvendo um pensamento criativo e crítico dos alunos (PRATAMA, *et. al*, 2016). Por sua vez, o STEM é uma excelente abordagem para promover o envolvimento dos alunos na aprendizagem, pois os alunos são fisicamente e emocionalmente envolvidos no ambiente de aprendizagem (STRUYF, 2019).

O modo de aprendizagem STEM tem sido desenvolvido em muitos países como, por exemplo: Taiwan (CHEN e LIN, 2019), Estados Unidos (GONZALEZ e KUENZI, 2012), Suiça (HINOJO-LUCEMA, *et. al*, 2020), Japão (YATA, *et. al*, 2020) e muitos outros. No trabalho de (OLIVEIRA e SOUZA, 2020) os autores fizeram uma abordagem teórica de tal método de aprendizagem, combinando a teoria de matrizes com geometria e navegação autônoma de robôs.

Uma maneira de combinar todas as áreas do modo de aprendizagem STEM é utilizando robótica no processo de ensino e aprendizagem (NAYA, *et. al*, 2017; TAKACS, *et. al*, 2016). Dessa forma, o modo de aprendizagem STEM enquadra-se no Construtivismo de Jean Piaget, em que materiais favorecem ao aluno aprender envolvendo-os com o que está sendo construindo. Em toda educação básica podemos utilizar robôs como ferramenta educacional. Em particular, no ensino médio o professor em colaboração com o aluno podem aprofundar a capacidade robótica e as apli-

cações.

A robótica como ferramenta educacional está em rápida expansão, onde professores, pesquisadores e empresas estão caminhando em conjunto para criar um novo ambiente de aprendizagem nas escolas. As plataformas robóticas educacionais mais populares são as seguinte: AlphaBot2, Lego Ev3, Dash&Dot, Edison, EUROPA, Ranger, Mbot, ThymioII. Existem ainda simuladores robóticos voltados para a indústria, que, embora não sejam plataformas educacionais, podem ser facilmente adaptadas para o modo de aprendizagem STEM, como, por exemplo: CoppeliaSim e RobotStudio.

Neste trabalho utilizaremos a robótica como ferramenta educacional, onde utilizaremos o modo de aprendizagem STEM no processo de ensino e aprendizagem de funções polinomiais do segundo grau. Mais precisamente, utilizaremos o simulador robótico CoppeliaSim, onde apresentaremos uma maneira de os professores trabalharem funções polinomiais do segundo grau resolvendo problemas atuais do mundo real, através dessa ferramenta. Este trabalho está organizado da seguinte maneira: a seção (1) é formada por esta breve introdução; na seção (2) apresentamos o ambiente de simulação CoppeliaSim, assim como um código que será alterado ao longo do trabalho; na seção (3) através da função polinomial do segundo grau, daremos uma funcionalidade para o robô, trabalhando diretamente com problemas que ilustram situações reais para um bom funcionamento de um robô; a seção (4) é dedicada para a conclusão do trabalho.

2. CoppeliaSim

Nesta seção faremos uma breve apresentação do ambiente de simulação CoppeliaSim, onde passaremos uma ideia do seu funcionamento e apresentaremos o robô que utilizaremos ao longo do trabalho, assim como o código em linguagem Python.

CoppeliaSim, antigamente chamado de V-REP, é um simulador robótico para dinâmica de corpos rígidos, que simula com um alto de grau de precisão as partes que compõem um robô e sua interação com o ambiente. O CoppeliaSim foi desenvolvido pela Toshiba e atualmente pertence a empresa Coppelia Robotics AG, localizada em Zurique na Suíça. Existem três versões do *software* CoppeliaSim: player, edu e pró. A versão pró é de uso puramente comercial, enquanto que as versões player e edu são versões livres. Neste trabalho utilizaremos a versão edu, que é a versão do *software* gratuita para estudantes, professores e pesquisadores.

O CoppeliaSim é um simulador multiplataforma que suporta um ambiente de desenvolvimento integrado, podendo se comunicar com vários ambientes de programação, ou seja, o *software* possui uma flexibilidade e uma portabilidade que facilita o ajuste do código de programação. O usuário tem a possibilidade de escolher entre utilizar a linguagem de programação do Coppelasim, chamada de Lua, ou utilizar outras linguagens, como, por exemplo Python.

Neste trabalho utilizaremos a linguagem de programação Python, onde o código será escrito e compilado através do ambiente de desenvolvimento integrado, PyCharm. Ou seja, integraremos o Python ao CoppeliaSim para sistema operacional windows. Para que a integração aconteça, os seguintes *softwares* devem estar instalados no PC que será utilizado:

- CoppeliaSim Edu 4.0, que pode ser baixado através do *link*:
  https:// www.coppeliarobotics.com/downloads.
  A escolha desta versão deve-se ao fato de que a mesma possui o robô que iremos trabalhar;
- Python, versão 2.7.18, que pode ser baixada através do *link*:
  https://www.python.org/downloads/release/python-270/ .
  A escolha desta versão é justamente para o bom funcionamento com a versão do CoppeliaSim escolhida;

- PyCharm, que pode ser baixado através do *link*:
  https://www.jetbrains.com/pycharm/download/#section=windows.
  Sugiro escolher a versão comunidade, que é mais leve que a profissional. Após baixar o PyCharm crie uma conta como professor ou aluno para utilizar o *software*.

Com os três *softwares* já instalados, integraremos o CoppeliaSim ao Python, obtendo assim um ambiente adequado para o nosso trabalho. O procedimento para integrar o Python ao CoppeliaSim é feito de modo bem simples seguindo os passos abaixo:

(1) vá até a pasta onde está instalado o CoppeliaSim, abra a pasta *programming*, depois abra a pasta remoteApiBindings, em seguida abra a pasta python, copie os arquivos: sim.py e simConst.py;

(2) procure a pasta PycharmProjects, abra a pasta Project e cole dentro dessa pasta os arquivos copiados no passo anterior;

(3) retorne para a pasta remoteApiBindings, abra a pasta lib, depois a pasta windows copie o arquivo remoteApi.dll e cole este arquivo junto aos demais arquivos do passo anterior na pasta Project do PycharmProjects;

(4) finalmente estão interligados o CoppeliaSim com o python, agora é só abrir o CoppeliaSim e o PyCharm para que possamos trabalhar.

O CoppeliaSim é uma plataforma dinâmica, bastante interativa e muito elegante. Agora teremos contato com o Lumibot, robô que será utilizado ao longo deste trabalho. A Figura 1, ilustra o passo a passo para utilização desta plataforma. Embora esteja numerado cada passo, não significa que esta deva ser a ordem adota.

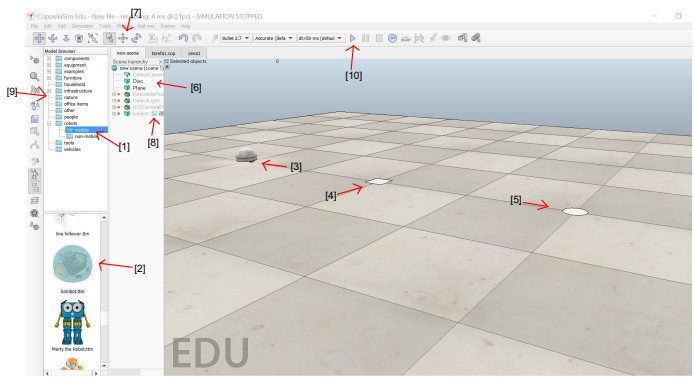

Figura 1: CoppeliaSim

(1) Na barra de botão modelo as pastas são divididas por categorias, selecionamos na pasta robot a subpasta mobile.

(2) Na subpasta robot escolhemos o robô Lumibot e arrastamos com o *mouse* para a área de trabalho.

(3) Temos o Lumibot na área de trabalho.

(4) Clicando com o botão direito do *mouse* em qualquer local da área de trabalho, aparacerá uma caixa, onde podemos clicar em add, em seguida escolher a opção primitive shape e escolher o plane. Automaticamente o plano e qualquer outra forma primitiva será colocada na origem do plano que representa a área de trabalho.

(5) Fazemos o mesmo que o passo anterior e escolhemos a opção disc.

(6) Podemos alterar o nome dos objetos que estão na área de trabalho. Além disso, podemos selecionar os objetos diretamente clicando sobre os seus nomes.

(7) Após selecionar os objetos que estão na área de trabalho, clicando sobre esse ícone podemos modificar a posição dos objetos na área de trabalho.

(8) Clicando sobre tal ícone abriremos o código em linguagem Lua do Lumibot, apagaremos todo o código Lua que está dentro dessa caixa e digitaremos: simRemoteApi.start(19999). Em seguida fechamos a caixa.

(9) Na barra de botão modelo, podemos selecionar diversos objetos, como por exemplo garras, plantas, cadeiras, janelas etc.

(10) Este é o ícone que inicia o processo que o Lumibot irá executar

Para os cenários que iremos trabalhar na próxima seção, o processo de construção é semelhante a esse.

O Lumibot é um robô móvel autônomo com duas rodas, envolvido por uma carenagem de formato esférico com 0,2 metro de diâmetro em sua base por 0,2 metro de altura. Sob sua carenagem transparente que se assemelha a um concha, podem ser visto seus equipamentos eletrônico que se iluminam quando o robô está em funcionamento. Tal robô é parecido com criaturas que moram no fundo do mar ou águas-vivas, como ilustra a Figura 2. O Lumibot, quando está em movimento, deixa rastros brilhantes que desaparecem lentamente.

Figura 2: Lumibot real

A versão virtual do Lumibot produzida dentro da plataforma de simulação CoppeliaSim, reproduz com fidelidade a versão real do robô, como ilustra Figura 3.

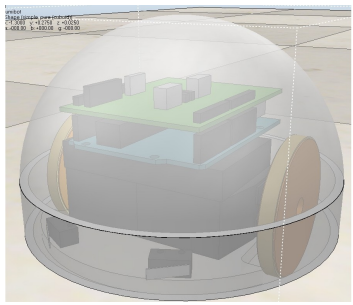

Figura 3: Lumibot versão virtual

Por sua vez, a Figura 4 ilustra os passos que serão aplicados neste trabalho para a utilização do PyCharm.

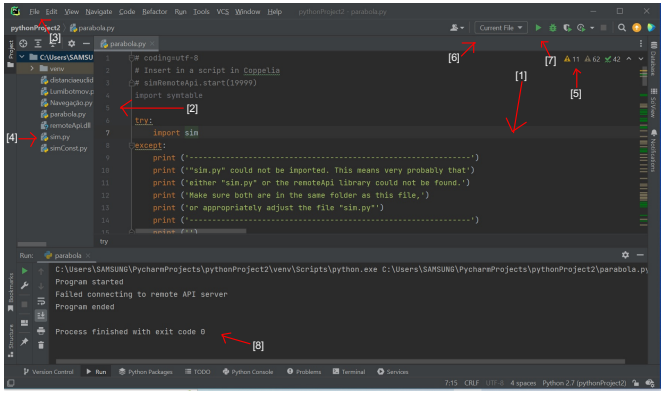

Figura 4: PyCharm

(1) Neste ambiente digitamos o código em linguagem Python que será utilizado com o CoppeliaSim.

(2) O código é numerado por linhas e colunas, cada linha dessa chama-se linha de comando.

(3) Através desse ícone podemos abrir um código que esteja salvo, bem como salvar o código que acabamos de criar.

(4) Nessa parte ficam todos os códigos que estão salvos dentro da pasta Project, assim como os arquivos que copiamos da pasta onde está instalado o CoppeliaSim.

(5) Quando esse símbolo fica vermelho é uma aviso de que o código tem erros, além de avisar a quantidade de erros que temos.

(6) Quando mais de um código estiverem abertos podemos escolher qual o código será compilado. Sugiro que a opção Current File esteja sempre selecionada.

(7) Este é o botão compilar.

(8) Quando a compilação tiver sido um sucesso esta será a mensagem que aparecerá.

Para finalizar a seção apresentaremos o código em Python que será utilizado neste trabalho. Cada linha de comando será acompanhada por m ↦, em que o natural m representa a linha onde a frase deve iniciar quando digitada no ambiente PyCharm. Essa notação, m ↦, não deve ser digitada junto com o código, sua utilidade é apenas para nos localizarmos.

Iniciamos o código com a comunicação entre o Python e o CoppeliaSim:

```
1 ↦#coding = utf − 8
2 ↦# Insert in a script in Coppelia
3 ↦#simRemoteApi.start(19999)
```

Em seguida importamos as funções matemática e de tempo:

```
4 ↦import symtable
5 ↦try:
6 ↦      import sim
7 ↦except:
8 ↦      print ('——————————————————————————————')
9 ↦      print('"sim.py"could not be imported. This means very probably that')
10 ↦     print('either "sim.py"or the remoteApi library could not be found.')
11 ↦     print('Make sure both are in the same folder as this file,')
12 ↦     print ('or appropriately adjust the file "sim.py"')
13 ↦     print('——————————————————————————————')
14 ↦     print (")
15 ↦import math
16 ↦import time
17 ↦print ('Program started')
18 ↦sim.simxFinish(−1)
19 ↦clientID = sim.simxStart('127.0.0.1', 19999, True, True, 5000, 5)
```

Agora declaramos os nomes do robô e do alvo, respectivamente:

```
20 ↦robotname = 'lumibot'
21 ↦targetname = 'Alvo'
22 ↦if clientID != −1 :
23 ↦     time.sleep(2)
```

Logo abaixo coletamos e armazenamos os valores do robô e do alvo:

```
24 ↦ [erro, robot] =sim.simxGetObjectHandle(clientID, robotname,sim.simx_opmode_oneshot_wait)
25 ↦ [erro, target] =sim.simxGetObjectHandle(clientID,targetname, sim.simx_opmode_oneshot
                    _wait)
26 ↦ [erro, robotLeftMotor] =sim.simxGetObjectHandle(clientID,robotname +'_leftMotor',
                              sim.simx_opmode_oneshot_wait)
27 ↦ [erro, robotRightMotor] =sim.simxGetObjectHandle(clientID, robotname +'_rightMotor',
                              sim.simx_opmode_oneshot_wait)
```

Depois coletamos as posições do robô e do alvo, assim como o ângulo de orientação do robô:

```
28 ↦ [erro, positionrobot] =sim.simxGetObjectPosition(clientID, robot, −1, sim.simx_opmode
                           _streaming)
29 ↦ [erro, positiontarget] =sim.simxGetObjectPosition(clientID, target, −1, sim.simx_opmode
                           _streaming)
30 ↦ [erro, orientationrobot] = sim.simxGetObjectOrientation(clientID,robot,−1,sim.simxopmode_
```

opmode_streaming)

```
31 ↦ time.sleep(2)
```

Através dos comandos dos motores abaixo, declaramos as velocidades nas rodas direita e esquerda, assim como valores de referência para a distância do alvo e para a velocidade:

```
32 ↦ sim.simxSetJointTargetVelocity(clientID, robotRightMotor, 0.0, sim.simx_opmode_oneshot)
33 ↦ sim.simxSetJointTargetVelocity(clientID, robotLeftMotor, 0.0, sim.simx_opmode_oneshot)
34 ↦ state = 'stopped'
35 ↦ dist_set = 0.1
36 ↦ vref = 2.0
37 ↦ running = True
38 ↦ while(running):
```

Declaramos as variáveis que representarão as posições do robô, do alvo e do ângulo. Além de defirmos a distância que será utilizada:

```
39 ↦  [erro, [xr, yr, zr]] = sim.simxGetObjectPosition(clientID, robot, -1, sim.simx_opmode_buffer)
40 ↦ [erro, [xt, yt, zt]] =sim.simxGetObjectPosition(clientID, target, –1, sim.simx_opmode_buffer)
41 ↦  [erro, [alpha, beta, gamma]] =sim.simxGetObjectOrientation(clientID, robot, -1, sim.simx
                                    _opmode_buffer)
42 ↦ dist = math.sqrt((xt − xr) ∗ (xt − xr) + (yt − yr) ∗ (yt − yr))
```

Inserimos os valores dos coeficientes da função, as posições inicial x_i e final x_f do robô durante a trajetória, assim como a própria função:

```
43 ↦        xp = xr
44 ↦        yp = yr + 0.5
45 ↦        ap = –1
46 ↦        bp = 0
47 ↦        cp = 0
48 ↦        xi = –1
49 ↦        xf = 1
50 ↦        dr = 4.5
51 ↦        fxp = ap ∗ xp ∗ xp + bp ∗ xp + cp
```

Em seguida faremos as transições entre os estágio, (parado, alinhando e avançando):

```
52 ↦        if(state ==′ stopped′)&(dist > dist_set) :
53 ↦            state =′ align′
54 ↦        elif(state ==′ align′)&(gamma < (ap/abs(ap)) ∗ math.pi/2) :
55 ↦            state =′ forward′
56 ↦        elif(state ==′ forward′)&(dist < dist_set) :
57 ↦            state =′ stopped′
```

Agora podemos introduzir as velocidades nas rodas direita e esquerda em cada estágio:

```
58 ↦        if state == 'stopped':
59 ↦            vRightMotor = 0.0
60 ↦            vLeftMotor = 0.0
61 ↦            running = False
62 ↦        elif state == 'align':
63 ↦            vRightMotor = (ap/abs(ap)) ∗ –vref
64 ↦            vLeftMotor = (ap/abs(ap)) ∗ vref
65 ↦        elif state =='forward':
66 ↦        vRightMotor = (ap/abs(ap))∗((xi–xf)/(abs(xi–xf)))∗(1/(0.05))∗((1/(math.sqrt(abs((2∗ap∗xp+
bp)∗(2∗ap∗xp+bp)+1)))∗(ap/abs(ap))∗(–(yp–fxp))∗(2∗ap∗xp+bp) +1/(math.sqrt(abs((2∗ap∗xp+bp)∗(2∗
ap ∗ xp + bp) + 1))) ∗ (xi − xf)/(abs(xi − xf)) ∗ math.sqrt(abs(1 − (ap/abs(ap)) ∗ (−(yp − fxp)) ∗ ((ap/abs(ap)) ∗
```

```
(−(yp − fxp)))))) ∗((0.18 ∗ math.sin(gamma))/(2 ∗ dr) − math.cos(gamma)) + (1/(math.sqrt(abs((2 ∗ ap ∗
xp + bp) ∗ (2 ∗ ap ∗ xp + bp) + 1))) ∗ (ap/abs(ap)) ∗ (−(yp − fxp)) − 1/(math.sqrt(abs((2 ∗ ap ∗ xp + bp)
∗(2∗ap∗xp+bp)+1)))∗(xi−xf)/(abs(xi−xf))∗math.sqrt(abs(1−(ap/abs(ap))∗(−(yp−fxp))∗((ap/abs(ap))∗
(−(yp − fxp))))) ∗ (2 ∗ ap ∗ xp + bp)) ∗ ((0.18 ∗ math.cos(gamma))/(2 ∗ dr) + math.sin(gamma)))
71 ↦ vLeftMotor = (ap/abs(ap))∗((xi−xf)/(abs(xi−xf)))∗(1/(0.05))∗((1/(math.sqrt(abs((2∗ap∗xp+
bp)∗(2∗ap∗xp+bp)+1)))∗(ap/abs(ap))∗(−(yp−fxp))∗(2∗ap∗xp+bp)+1/(math.sqrt(abs((2∗ap∗xp+bp)
∗(2∗ap∗xp+bp)+1)))∗(xi−xf)/(abs(xi−xf))∗math.sqrt(abs(1−(ap/abs(ap))∗(−(yp−fxp))∗((ap/abs(ap))∗
(−(yp − fxp)))))) ∗ ((−0.18 ∗ math.sin(gamma))/(2 ∗ dr) − math.cos(gamma)) +(1/(math.sqrt(abs((2 ∗ ap ∗
xp + bp) ∗ (2 ∗ ap ∗ xp + bp) + 1))) ∗ (ap/abs(ap)) ∗ (−(yp − fxp)) − 1/(math.sqrt(abs((2 ∗ ap ∗ xp + bp) ∗ (2 ∗
ap ∗ xp + bp) + 1))) ∗ (xi − xf)/(abs(xi − xf)) ∗math.sqrt(abs(1 − (ap/abs(ap)) ∗ (−(yp − fxp)) ∗ ((ap/abs(ap)) ∗
(−(yp − fxp))))) ∗ (2 ∗ ap ∗ xp + bp)) ∗ (math.sin(gamma) − (0.18 ∗ math.cos(gamma))/(2 ∗ dr)))
```

Por fim, encerramos o código:

```
76 ↦ sim.simxPauseCommunication(clientID, True)
77 ↦ sim.simxSetJointTargetVelocity(clientID, robotRightMotor, vRightMotor,
         4im.simx_opmode_oneshot)
78 ↦ sim.simxSetJointTargetVelocity(clientID, robotLeftMotor, vLeftMotor, sim.simx_opmode_oneshot)
79 ↦ sim.simxPauseCommunication(clientID, False)
80 ↦ sim.simxPauseSimulation(clientID,sim.simx_opmode_oneshot_wait)
81 ↦ sim.simxFinish(clientID)
82 ↦ else:
83 ↦   print('Falha de conexão')
84 ↦ print('Programa finalizado')
```

Fique atento, pois cada frase do código que apresentamos acima deve ser digitada, rigorosamente, da mesma maneira obedecendo a ordem para linhas e os espaço para as colunas. Tudo ocorrendo bem, o código já digitado iremos colocar o CoppeliaSim para funcionar através do PyCharm. Para isso, inicie o processo no CoppeliaSim (passo (10) ilustrado na Figura 1), em seguida execute o código com PyCharm (passo (7) ilustrado na Figura 4).

3. Resultado

Nesta seção utilizaremos funções polinomiais do segundo grau para que possamos dar funcionalidade para o Lumibot, apresentando uma maneira de resolver um problema do mundo real que consiste em determinar a trajetória de um robô móvel.

Função polinomial do segundo grau, também chamada de função quadrática, é um dos principais conteúdos ensinado no ensino médio. É de grande importância que os estudantes dominem completamente a função quadrática; isto significa entender suas características e possíveis aplicações.

O método de ensino, comumente, utilizado nas escolas de ensino médio consiste em explicar a definição de função quadrática, fazer exemplos e apresentar suas diferentes formas. A forma mais comum utilizada é dada da seguinte maneira:

$$f(x) = ax^2 + bx + c \qquad a \neq 0. \tag{1}$$

Assim, partindo de (1) exploramos as raízes, o gráfico que é uma parábola, vértices com o valor mínimo e máximo, sinal, monotonicidade, e os casos em que a parábola tem concavidade para cima e para baixo.

Um dos requisitos mais importantes para os alunos é que eles entendam corretamente a conexão de dependência dos coeficientes a, b e c em (1). De certa forma, conhecendo esses coeficientes, a função quadrática fica completamente determinada. Isto é, sejam x_0, x_1, x_2 núme-

ros reais ou complexos distintos, queremos determinar uma função f(x) que satisfaça (1) tal que $f(x_0) = y_0, f(x_1) = y_1, f(x_2) = y_2$. Ou seja, o sistema linear

$$\begin{cases} ax_0^2 + bx_0 + c = y_0 \\ ax_1^2 + bx_1 + c = y_1 \\ ax_2^2 + bx_2 + c = y_2 \end{cases} \tag{2}$$

admite uma única solução $\{a \neq 0, b, c\}$. Como x_0, x_1, x_2 são distintos, então multiplicando a primeira linha em (2) por $(x_1 - x_2)$, a segunda por $-(x_0 - x_2)$ e a terceira por $(x_0 - x_1)$, obtemos:

$$\begin{cases} a(x_1 - x_2)x_0^2 + b(x_1 - x_2)x_0 + c(x_1 - x_2) = y_0(x_1 - x_2) \\ -a(x_0 - x_2)x_1^2 - b(x_0 - x_2)x_1 - c(x_0 - x_2) = -y_1(x_0 - x_2) \\ a(x_0 - x_1)x_2^2 + b(x_0 - x_1)x_2 + c(x_0 - x_1) = y_2(x_0 - x_1) \end{cases} \tag{3}$$

Somando membro a membro as três equações acima, segue-se que

$$a = \frac{y_0(x_1 - x_2) - y_1(x_0 - x_2) + y_2(x_0 - x_1)}{(x_1 - x_2)x_0^2 - (x_0 - x_2)x_1^2 + (x_0 - x_1)x_2^2}.$$

Continuando com o processo, multiplicamos a primeira, segunda e terceira linhas em (2) por $-(x_1^2 - x_2^2)$, $(x_0^2 - x_2^2)$ e $-(x_0^2 - x_1^2)$, respectivamente, e em seguida somando membro a membro, obtemos que

$$b = \frac{y_0(x_1^2 - x_2^2) - y_1(x_0^2 - x_2^2) + y_2(x_0^2 - x_1^2)}{(x_1 - x_2)x_0^2 - (x_0 - x_2)x_1^2 + (x_0 - x_1)x_2^2}.$$

Finalmente, multiplicamos a primeira, segunda e terceira linhas em (2) por $(x_2x_1^2 - x_1x_2^2)$, $-(x_2x_0^2 - x_0x_2^2)$ e $(x_1x_0^2 - x_0x_1^2)$, respectivamente, e em seguida somando membro a membro, obtemos que

$$c = \frac{y_0(x_2x_1^2 - x_1x_2^2) - y_1(x_2x_0^2 - x_0x_2^2) + y_2(x_1x_0^2 - x_0x_1^2)}{(x_1 - x_2)x_0^2 - (x_0 - x_2)x_1^2 + (x_0 - x_1)x_2^2},$$

desde que $(x_1 - x_2)x_0^2 - (x_0 - x_2)x_1^2 + (x_0 - x_1)x_2^2 \neq 0$.

Portanto, existem e são únicos os coeficientes $a \neq 0, b, c$. Além disso, o processo acima permite determinar esses coeficientes. Segue-se que conhecendo três valores para x e f(x) em (1), obtemos os coeficientes a, b e c, e consequentemente toda a função f(x).

Utilizaremos o Construtivismo como recurso pedagógico, tendência essa que estimula os alunos a participarem ativamente do processo de aprendizagem utilizando materiais que propiciam ao aluno a aprender e a pensar sobre o aprender (PAPERT, 2020).

Dividiremos esta seção em quatro subseções que serão chamadas de cenários. Após apresentar a plataforma e os cenários aos alunos, o professor pode dividir a turma em grupos de no máximo quatro componentes, onde dois grupos podem ficar com o mesmo cenário. O professor propõe aos grupos que encontrem a trajetória descrita pelo robô em seu, respectivo, cenário. Após os alunos encontrarem a função polinomial do segundo grau que será utilizada para determinar a trajetória do robô Lumibot, o professor pode projetar as respectivas trajetórias encontradas por cada grupo.

3.1. Cenário 1

Inicialmente consideraremos um cenário bem simples criado através do CoppeliaSim, formado apenas por um piso de centro na origem, que coincide com o centro do círculo branco. Além

disso, sobre o piso temos um quadrado branco de lados medindo 0.01 metro e um símbolo em X de cor branca e o robô Lumibot, como ilustra a Figura 5. Partindo do círculo que é a origem do plano, obtemos que o robô Lumibot localiza-se na posição $P_i = (-2, 0.24)$. Por sua vez, o quadrado está localizado na posição $P = (-1, -0.75)$, e o alvo que o Lumibot deve atingir representado pelo símbolo X está localizado na posição $P_f = (0, 0.25)$, como ilustra a Figura 5. Todas essas posições são consideradas em relação ao centro de massa dos objetos e são dadas em metros.

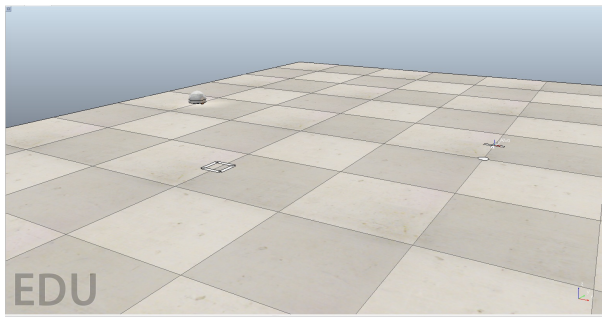

Figura 5: Plano centrado sobre o círculo

Queremos encontrar uma trajetória para que o Lumibot saia de sua posição inicial, passe pelo quadrado branco e chegue até o alvo. Considerando a expressão (1), quando o robô estiver em sua posição inicial obtemos que a função f satisfaz $f(-2) = 0.24$, ou seja, $4a - 2b + c = 0.24$. No momento em que o robô passar pelo quadrado, a função f satisfaz $f(-1) = -0.75$, isto é, $a - b + c = -0.75$. Por sua vez, no momento em que o robô atingir o alvo, a função f satisfaz $f(0) = 0.25$ e assim obtemos o sistema linear:

$$\begin{cases} 4a - 2b + c = 0.24 \\ a - b + c = -0.75 \\ c = 0.25 \end{cases} \tag{4}$$

Resolvendo o sistema (4), segue-se que $a = 1$, $b = 2$ e $c = 0.25$, donde

$$f(x) = x^2 + 2x + 0.25$$

é a função polinomial do segundo grau obtida, cujo gráfico representa a trajetória procurada, como ilustra a Figura 6. Seguindo as notações do código em Python dado na seção 2, temos que

$$fxp = xp * xp + 2 * xp + 0.25$$

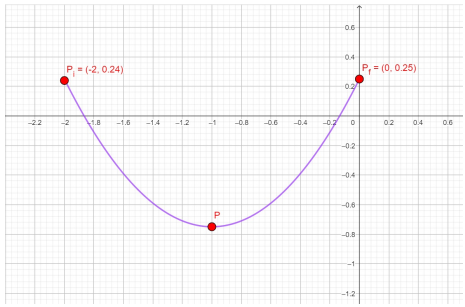

Figura 6: Parábola determinada por P_i, P e P_f

Substituiremos os valores diretamente no código dado na seção 2. Iniciamos com o valor da constante ap = 1, que será substituído na linha 45; depois o valor da constante bp = 2, que será substituído na linha 46; por sua vez o valor da constante cp = 0.25, que será substituído na linha 47; em seguida o valor inicial de x, xi = –2, será substituído na linha 48; continuando, substituímos o valor final de x, xf = 0, que será substituído na linha 49; e, por último, o valor de controle de trajetória dr = 4.5 será substituído na linha 50. Segue-se, que o Lumibot percorrerá o trajeto descrito por uma parábola passando pelo quadrado até atingir o alvo, como ilustra a Figura 7. Toda a trajetória pode ser acompanhada na íntegra através do *link* https://youtu.be/i1cv8HZYD5w

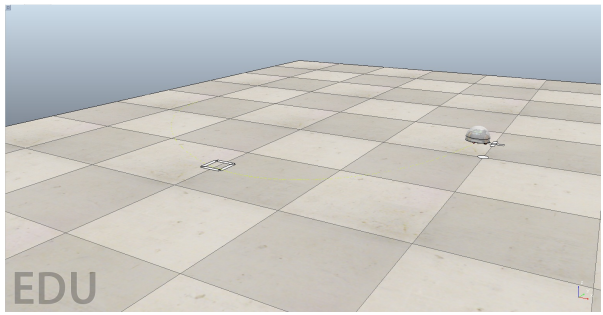

Figura 7: Trajeto descrito até atingir o alvo

O valor do ponto de controle pode ser escolhido entre 1 e 10, sendo que a escolha desse valor influencia diretamente no percurso do robô.

3.2. Cenário 2

Considere o cenário de uma sala de estar, composta por uma poltrona, uma mesa de centro, um *notebook*, um copo de vidro, uma jarra de vidro, uma planta, uma caixa verde (escondida pela

mesa de centro) e o Lumibot, como ilustra a Figura 8

Figura 8: O Lumibot à direita da poltrona, e alvo embaixo da mesa

Sabendo que o centro da poltrona é exatamente a origem do plano, e que cada quadrado que compõe o piso tem lados medido 0.5 metro, obtemos que o Lumibot situa-se na posição $P_i = (-1, 0)$ sobre o plano, e o alvo está situado na posição $P_f = (0, -1)$. Queremos encontrar a trajetória que será seguida pelo robô para sair de sua posição inicial e chegar até o seu alvo, representado pelo círculo embaixo da mesa de centro. Como entre o robô e o alvo temos uma caixa, como ilustra a Figura 9, então não podemos utilizar uma reta como trajetória.

Figura 9: Caixa verde como obstáculo

Uma alternativa para esse problema é dada através da expressão (1), pois quando o robô encontra-se na posição inicial, a função f satifaz $f(-1) = 0$, ou seja, $a - b + c = 0$; por sua vez, quando o robô atingir o alvo, f satisfaz $f(0) = -1$, isto é, $c = -1$. Afim de determinarmos completamente a função polinomial f necessitamos de pelo menos mais um ponto sobre o gráfico da função. Para isso, note que o robô deve desviar da caixa em verde centrada no ponto $Q = (-0.5, -0.5)$, com dimensões da base dadas por 0.10 metro de largura por 0.10 metro de comprimento. Assim, levando em consideração as dimensões da caixa e do robô, a posição $P = (-\frac{\sqrt{10}}{5}, -\frac{6}{10})$ é um local

seguro para o robô passar durante seu percurso. Para essa posição P, f satisfaz $f(-\frac{\sqrt{10}}{5}) = -\frac{6}{10}$, ou seja, $\frac{2}{5}a - \frac{\sqrt{10}}{5}b + c = \frac{-3}{5}$, e assim obtemos o sistema linear:

$$\begin{cases} a & - & b & + & c & = & 0 \\ \frac{2}{5}a & - & \frac{\sqrt{10}}{5}b & + & c & = & -\frac{3}{5} \\ & & & & c & = & -1 \end{cases} \tag{5}$$

Resolvendo o sistema (5), segue-se que $a = 1$, $b = 0$ e $c = -1$, donde a função quadrática cujo gráfico ilustrado na Figura 10 representa o trajeto que será seguido pelo Lumibot, até atinge o seu alvo, dada da seguinte maneira:

$$f(x) = x^2 - 1$$

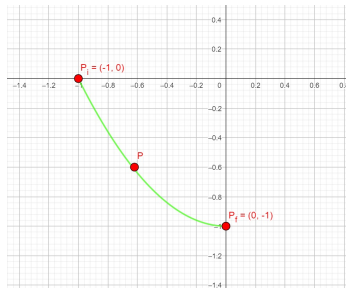

Figura 10: Parábola determinada por P_i, P e P_f

Ou seja, seguindo as notações do código obtemos a função quadrática

$$fxp = xp * xp - 1.$$

Substituiremos os valores diretamente no código. Iniciamos com o valor da constante $ap = 1$ que será substituído na linha 45; depois o valor da constante $bp = 0$ que será substituído na linha 46; por sua vez o valor da constante $cp = -1$, que será substituído na linha 47. Continuando, substituímos o valor inicial de x, $xi = -1$, na linha 48, e por sua vez o valor final de x, $xf = 0$, que será substituído na linha 49. Finalmente o valor de controle de trajetória $dr = 4.5$ será substituído na linha 50. Segue-se que o Lumibot percorrerá o trajeto descrito por uma parábola desviando do obstáculo até atingir o alvo, como ilustra a Figura 11.

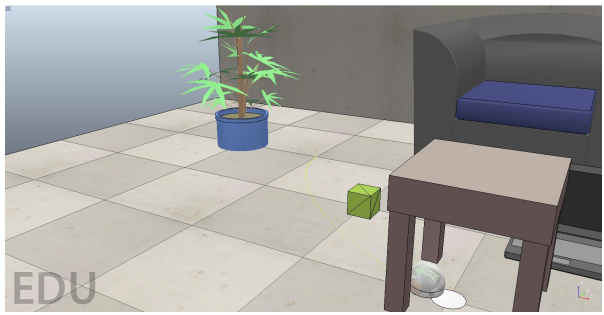

Figura 11: Trajetória desviando da caixa verde

Toda a trajetória pode ser acompanhada na íntegra através do link https://youtu.be/wvAcMiRbHSg

3.3. Cenário 3

Considere o mesmo cenário ilustrado anteriormente, com apenas uma mudança, o alvo que estava embaixo da mesa agora localiza-se ao lado da poltrona, como ilustra a Figura 12

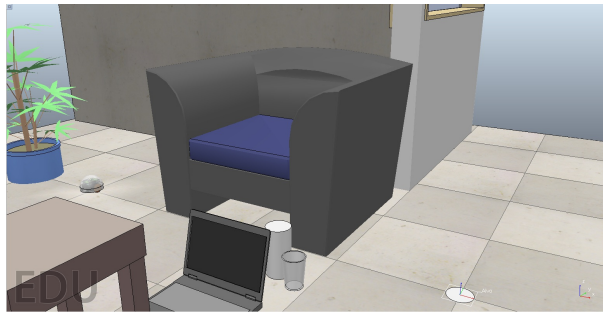

Figura 12: Alvo à esquerda da poltrona

Como cada quadrado que forma o piso tem lado medindo 0,5 metro, obtemos que a posição do alvo no plano é dada pelo par ordenado $P_f = (1, 0)$. Queremos encontrar a trajetória que será seguida pelo robô para sair de sua posição inicial P_i e atingir o seu alvo P_f. Uma alternativa para esse problema é considerar a trajetória descrita no cenário anterior, prolongando a parábola, passando embaixo da mesa de centro até atingir P_f. Assim, obtemos a mesma função quadrática $f(x) = x^2 - 1$ cujo gráfico ilustrado na Figura 13, representa a trajetória que o robô irá seguir até atingir o seu alvo.

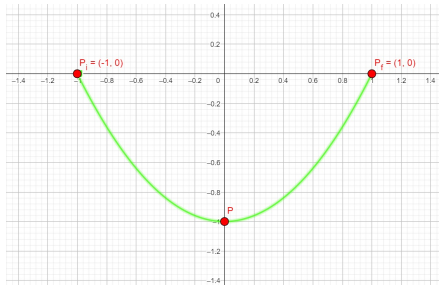

Figura 13: Parábola determinada por P_i, P e P_f

Trocando apenas os valor final xf = 1 no Cenário 2, segue que o Lumibot seguirá a trajetória descrita por uma parábola, como ilustra a Figura 14.

Figura 14: Trajetória descrita pelo robô até atingir o alvo

Toda a trajetória pode ser acompanhada na íntegra através do *link* https://youtu.be/ZCTPnD80Anw

3.4. Cenário 4

Considere uma sala de jantar conjugada com uma sala de estar e uma pequena cozinha. A sala de estar é composta por uma planta, duas poltronas de cor azul, dois copos de vidro bem próximos as poltronas, uma mesa de centro sobre um tapete; em cima da mesa de centro temos um projetor de imagem apontando para uma tela de projeção retrátil sobre a parede ao fundo da sala. A sala de jantar é composta por uma mesa e seis cadeiras. Por sua vez, a cozinha tem uma bancada para a sala de jantar e uma porta de abertura e fechamento automático. Todo esse cenário é ilustrado de modo panorâmico na Figura 15

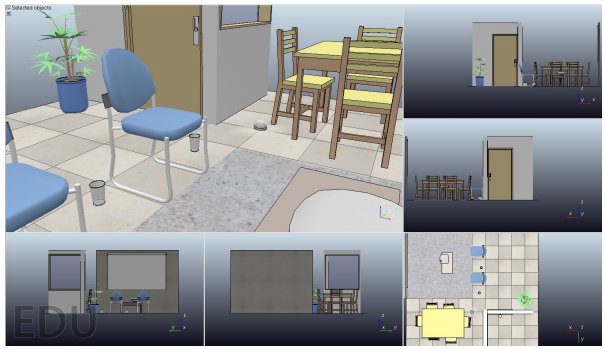

Figura 15: Vista panorâmica das salas com a cozinha

Queremos que o robô Lumibot situado na sala de jantar, na posição $P_i = (0, 1)$, atinja o alvo dado por um círculo branco situado na cozinha, na posição $P_f = (-1.07, 1.15)$, como ilustra a Figura 16.

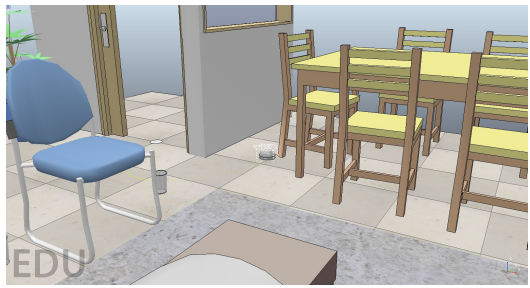

Figura 16: O alvo na cozinha

Mais uma vez, uma alternativa para este problema é dada através da expressão (1). Conhecemos as posições P_i e P_f sobre gráfico dessa função; daí para que a função fique completamente determinada precisamos encontrar mais um ponto sobre o gráfico da função f. Considerando as dimensões do robô, considere o ponto $P = (-0.5, 0.5)$ por onde o robô deve passar durante seu percurso até atingir o seu alvo. Segue-se que quando o robô encontra-se na posição inicial, f satisfaz $f(0) = 1$, ou seja, $c = 1$; no instante em que o robô atingir a posição P, f satisfaz $\frac{1}{4}a + \frac{1}{2}b + c = \frac{1}{2}$; por sua vez, quando o robô encontra-se sobre a posição final, f satisfaz $f(-1.07) = 1.15$, isto é, $\frac{11449}{10000}a + \frac{107}{100}b + c = \frac{115}{100}$. Dessa forma obtemos o sistema linear:

$$\begin{cases} \frac{11449}{10000}a + \frac{107}{100}b + c = 0 \\ \frac{1}{4}a + \frac{1}{2}b + c = \frac{1}{2} \\ c = 1 \end{cases} \tag{6}$$

Resolvendo o sistema acima, obtemos a função polinomial do segundo grau:

$$f(x) = 2x^2 + 2x + 1,$$

cujo gráfico ilustrado na Figura 17

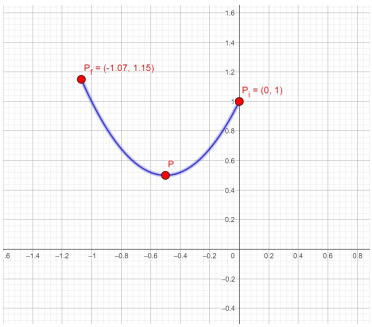

Figura 17: Parábola determinada por P_i, P e P_f

representa a trajetória que o robô irá seguir até atingir o seu alvo. Ou seja, seguindo as notações do código obtemos a função

$$fxp = 2 * xp * xp + 2 * xp + 1.$$

Substituindo no código os valores ap = 2, bp = 2, cp = 1, xi = 0 , xf = –1.07 e kp = 4.5, nas linhas 45, 46, 47, 48, 49, 50, respectivamente, segue-se que o Lumibot percorre o trajeto passando pelo quadrado até atingir o obstáculo, como ilustra a Figura 15.

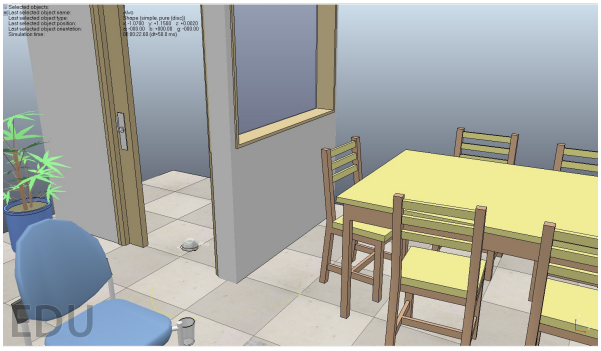

Figura 18: Trajeto descrito até atingir o alvo

Toda a trajetória pode ser acompanhada na íntegra através do *link* https://youtu.be/hD7qcXIcCcc
.

4. Conclusão

Neste trabalho, através do ambiente de simulação robótica, CoppeliaSim, conseguimos uma interdisciplinaridade envolvendo o conteúdo matemático de funções polinomiais do segundo grau. Mais precisamente, através do gráfico das funções polinomiais do segundo grau, conseguimos apresentar uma funcionalidade para o robô Lumibot. Dessa forma, obtemos uma importante ferramenta que permite despertar criatividade matemática dos alunos da educação básica, ou do público em geral, para atuarem com as novas profissões. Tudo isso está em comum acordo com o modo de aprendizagem STEM.

Apresentamos o ambiente de simulação CoppeliaSim acompanhado de um código editável, isso permite aos professores criarem novos cenários que podem ser trabalhados com os alunos em sala de aula, como, por exemplo explorando trajetórias para o Lumibot descritas por parábolas com a concavidade voltada para baixo.

Em geral, utilizando o Lumibot podemos ainda trabalhar com situações de contexto mais complexos, estimulando os alunos a trabalharem com a noção de pontos de mínimo e máximo, ou situações diversas. De certa forma, tal ferramenta desperta tanto o interesse dos alunos por matemática, assim como a criatividade matemática necessária para resolver problemas reais como os apresentados neste trabalho.

Referências

[1] Bybee, R. W. (2013). The Case for STEM Education: Challenges and Opportunities. Washington DC: National STEM Teachers Association. Disponível em: https://www.nsta.org/resources/case-stem-education-challenges-and-opportunities Acessado em: 15 jul. 2022.

[2] Chen, C. S., Lin, J. W. (2019). A Practical Action Research Study of the Impact of Maker-Centered STEM-PjBL on a Rural Middle School in Taiwan. *International Journal of Science and Mathematics Education*, 17(1), 85–108. Disponível em: https://eric.ed.gov/?id=EJ1220538 Acessado em: 20 mar. 2022.

[3] Gonzalez, H. B. and Kuenzi, J. (2012). Science, technology, engineering, and mathematics (STEM):A Primer. Congressional Research Service, August, 1–15. Disponível em: http://www.stemedcoalition.org/wp-content/uploads/2010/05/STEM-Education-Primer.pdf. Acessado em: 06 fev. 2022.

[4] Hinojo-Lucena, F. J., Dúo-Terrón, P., Navas-Parejo, M. R., Rodríguez-Jiménez, C., Moreno-Guerrero, A. J. (2020). Scientific performance and mapping of the term STEM in education on the web of science. Sustainability (Switzerland), 12(6), 1–20. Disponível em: https://www.mdpi.com/665100. Acessado em: 12 mai. 2022.

[5] Naya, M. et al. (2017). A versatile robotic plataform for educational interaction. *In Proceedigns of the 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications(IDAACS)*, Bucharest, Romania, 21–23; Volume 1, pp. 138–144. Disponível em: http : //idaacs.net/storage/conferences/2/abstracts/i17 – 052 – camera$_r$eady.pdf
Acessado em: 18 abr. 2022.

[6] Oliveira, M. F. e Souza, C. A. (2020). A Circunferência de Centro na Origem como Produto de Matrizes. *Professor de Matemática online-PMO*, volume 8, 535-550. Dísponível em:

https://scholar.archive.org/work/5nizl5is5zhznljm5zgmmys7oi/access/wayback/http://pmo.sbm.c
content/uploads/sites/16/dlm$_{u}$ploads/2020/10/art39$_{v}$ol8$_{P}$MO$_{S}$BM$_{2}$020b.pdf
Acessado em: 20 jul. 2022.

[7] Papert, S. A. (2020). Mindstorms: *Children, computers, and powerful ideas. Basic books.*

[8] Pratama, I., Permanasari, A. E., Ardiyanto, I., & Indrayani, R. (2016). A review of missing values handling methods on time-series data. *In 2016 international conference on information technology systems and innovation* (ICITSI), 1-6.

[9] Rosa, M., Orey, D. C. (2018). STEM Education in the Brazilian Context: An Ethnomathematical Perspective. In R. Jorgensen & K. Larkin (Eds.), STEM Education in the Junior Secondary: The State of Play (pp. 221–247). Springer Singapore. Disponível em: https : //link.springer.com/chapter/10.1007/978 – 981 – 10 – 5448 – 8$_{1}$1
Acessado em: 14 mar. 2022.

[10] Struyf, A., Loof, H. De, Pauw, J. B., & Petegem, P. Van. (2019). Students ' engagement in different STEM learning environments : integrated STEM education as promising practice ? *International Journal of Science Education*, 41(10), 1387–1407. Disponível em: https://www.tandfonline.com/doi/abs/10.1080/09500693.2019.1607983
Acessado em: 14 mar. 2022

[11] Takacs, A. G. et al. (2016). Teachers kit: development, usability and communities of modular robotic kits for classroom education. *IEEE Robotics & Automation Magazine* 23(2): 30-39.

[12] Tsupros, N., Kohler, R., & Hallinen, J. (2009). STEM education: A project to identify the missing components. Pittsburgh, PA: Intermediate Unit 1 and Carnegie Mellon.

[13] Yildirim, B. and Selvi, M. (2016) J. Examination of the effects of STEM education integrated as a part of science, technology, society and environment courses. *Journal of Human Science*, 13(2), 684-3695. Disponível em:
https://avesis.gazi.edu.tr/yayin/f3a35b99-cc1d-4f50-b75c-ba09f4501b5b/examination-of-the-effects-of-stem-education-integratedas-a-part-of-science-technology-society-and-environmentcourses/document.pdf
Acessado em: 18 jul. 2022.

[14] Yata, C., Ohtani, T., Isobe, M., (2020). Conceptual framework of STEM based on Japanese subject principles. *International Journal of STEM Education*, 7,1-10. Disponível em:
https://stemeducationjournal.springeropen.com/articles/10.1186/s40594-020-00205-8
Acessado em: 20 mai. 2022.

Cássio Lima Macedo
Secretaria de Educação de Timon Maranhão
<kassiooolima@gmail.com>

Cleidinaldo Aguiar Souza
Universidade Federal do Piauí-UFPI
<aguiarnaldo@ufpi.edu.br>

Recebido: 25/03/2023
Publicado: 02/08/2023