Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Matemática
Mestrado Profissional em Matemática

Ediney Laurindo Alves

Função Polinomial do Terceiro Grau com o Lumibot-CoppeliaSim

Teresina - 2024

Ediney Laurindo Alves

Dissertação de Mestrado:

Função Polinomial do Terceiro Grau com o Lumibot-CoppeliaSim

Dissertação submetida à Coordenação do Programa de Mestrado Profissional em Matemática - Profmat, da Universidade Federal do Piauí, como requisito parcial para obtenção do grau de Mestre em Matemática na modalidade profissional.

Orientador:
Prof. Dr. Cleidinaldo Aguiar Souza.

Teresina - 2024

Universidade Federal do Piauí - UFPI, Centro de Ciência da Natureza - CCN, Programa de Pós-Graduação em Matemática, Mestrado Profissional em Matemática. Cep 64049-550 - Teresina, PI.

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Biblioteca Comunitária Jornalista Carlos Castello Branco
Divisão de Representação da Informação

A474f Alves, Ediney Laurindo.
Função polinomial do terceiro grau com o Lumibot-CoppelianSim / Ediney Laurindo Alves. -- 2024.
64 f.

Dissertação (Mestrado) – Universidade Federal do Piauí, Programa de Pós-Graduação em Matemática, Teresina, 2024.
"Orientador: Prof. Dr. Cleidinaldo Aguiar Souza".

1. Função polinomial do terceiro grau. 2. Robótica educacional. 3. CoppelianSim. I. Souza, Cleidinaldo Aguiar. II. Título.

CDD 515.55

Bibliotecária: Francisca das Chagas Dias Leite – CRB3/1004

Ediney Laurindo Alves

Função Polinomial do Terceiro Grau com o Lumibot-CoppeliaSim

Dissertação submetida à banca examinadora abaixo discriminada em defesa pública e aprovada em 28/02/2024.

BANCA EXAMINADORA

Cleidinaldo Aguiar Souza (Orientador)

Universidade Federal do Piauí

gov.br Documento assinado digitalmente
VITALIANO DE SOUSA AMARAL
Data: 25/04/2024 14:59:33-0300
Verifique em https://validar.iti.gov.br

Vitaliano de Sousa Amaral

Universidade Federal do Piauí

Natã Firmino Santana Rocha

Universidade Estadual do Piauí

Teresina - 2024

Dedico esse trabalho à minha mãe Maria Alves de Jesus, a minha esposa Regina Lúcia ao meu filho Ediney L. A. Júnior e a meu irmão Edinaldo Germano Alves (in memorian), com muito amor e saudade.

Agradecimentos

Agradeço, a Deus, por tornar possível esta grande conquista.

A minha esposa Regina Lúcia do nascimento e ao meu filho Ediney Laurindo Alves Júnior que juntos foram meu alicerce, a minha força motivadora.

Ao meu irmão Edinaldo Germano Alves (in memorian), por sempre acreditar em mim.

Aos meus amigos do mestrado, que sempre estiveram ao meu lado, pela amizade incondicional e pelo apoio demonstrado ao longo de todo o período de tempo em que estivemos juntos.

Ao meu amigo e Professor Dr. Cleidinaldo Aguiar Souza, por ter sido meu orientador e ter desempenhado tal função com dedicação e amizade.

Enfim, agradeço a todos que participaram dessa jornada.

"*Se pude enxergar mais longe, foi porque me apoiei em ombros de gigantes*".

Isaac Newton.

Resumo

Com a ascensão das tecnologias aplicadas ao ensino, a robótica educacional vem despertando o interesse de docentes e acadêmicos que almejam aprimorar o desenvolvimento cognitivo dos educandos e atender a uma nova demanda educacional em busca de atividades mais significativas e inovadoras. Nesta perspectiva, podemos pensar na robótica educacional como um recurso capaz de integrar diversas áreas do conhecimento, podendo trazer mudanças na forma de agir, aprender e pensar dos alunos. Por isso, este trabalho se apresenta como uma oportunidade de aplicar a robótica educativa em conteúdos do ensino básico. Mais especificamente, utilizamos o Lumibot que é um dos robôs presentes no software CoppeliaSim. Abordando funções polinomiais do terceiro grau, visto que funções cúbicas podem ser aplicadas a situações do mundo real, como modelagem de fenômenos físicos, crescimento populacional, análise de dados e previsões, permitindo que os alunos apliquem conceitos matemáticos a contextos práticos. Com o uso da linguagem de programação Python, que será responsável por calcular o polinômio de terceiro grau que servirá de caminho para o robô Lumibot do CoppeliaSim. Os coeficientes deste polinômio serão responsáveis pelo trajeto do Lumibot, ou seja, o robô percorrerá o gráfico do polinômio gerado no Python. Isso possibilitará a visualização do gráfico e da trajetória do robô no ambiente de simulação do CoppeliaSim. Foram propostos vários desafios ao Lumibot no cenário de visualização do CoppeliaSim, utilizando apenas os conhecimentos de polinômios do terceiro grau, assim como de gráficos gerados por esses polinômios, conferindo um significado prático ao estudo desses conteúdos.

Palavras-chave: função polinomial do terceiro grau; robótica educacional; CoppeliaSim.

Abstract

With the rise of technology applied to education, educational robotics has been capturing the interest of educators and academics who aspire to enhance the cognitive development of learners and meet a new educational demand for more meaningful and innovative activities. In this perspective, educational robotics can be seen as a resource capable of integrating various areas of knowledge, potentially bringing changes to the way students act, learn, and think. Therefore, this work presents an opportunity to apply educational robotics to elementary education content. More specifically, we utilized the Lumibot, one of the robots present in the CoppeliaSim software. Addressing third-degree polynomial functions, as cubic functions can be applied to real-world situations such as modeling physical phenomena, population growth, data analysis, and predictions, allowing students to apply mathematical concepts to practical contexts. Using the Python programming language, which will be responsible for calculating the third-degree polynomial that will serve as a path for the Lumibot robot in CoppeliaSim. The coefficients of this polynomial will determine the Lumibot's trajectory, meaning the robot will traverse the graph of the polynomial generated in Python. This will enable the visualization of the graph and the robot's trajectory in the CoppeliaSim simulation environment. Various challenges were proposed to the Lumibot in the CoppeliaSim visualization scenario, utilizing only the knowledge of third-degree polynomials, as well as graphs generated by these polynomials, giving practical meaning to the study of these contents.

Keywords: third-degree polynomial function; educational robotics; CoppeliaSim.

Sumário

Introdução

Compreender as revoluções industriais que marcaram os últimos séculos é essencial para contextualizar a trajetória que nos trouxe até o momento atual.

Segundo Chiavenato [9], a primeira revolução industrial, que teve início por volta de 1765 na Inglaterra, representou uma mudança histórica, introduzindo a mecanização dos processos produtivos, ou seja, a substituição do trabalho humano por máquinas. Logo após a primeira revolução industrial, a tecnologia avançou rapidamente.

A segunda revolução industrial teve início por volta de 1870, quando a eletricidade e o petróleo surgiram como fontes alternativas de energia. Esse período marcou o desenvolvimento de indústrias químicas e siderúrgicas, resultando na criação de inovações como automóveis, telefones e rádios. Com o término da Segunda Guerra Mundial, os avanços tecnológicos abriram caminho para a descoberta de uma nova forma de energia, ainda mais poderosa que as anteriores: a energia nuclear.

Por volta de 1969, teve início a terceira revolução industrial. Essa fase se destacou pelo surgimento de equipamentos eletrônicos, telecomunicações e computadores. Essa nova onda tecnológica também possibilitou a exploração espacial e avanços notáveis na biotecnologia. Na indústria, a terceira revolução industrial trouxe a invenção de robôs e sistemas automatizados, além do modelo de produção conhecido como Toyotismo, caracterizado pela flexibilidade e eficiência. Esse conceito também influenciou os métodos de ensino e aprendizagem, promovendo abordagens mais flexíveis e adaptáveis às necessidades dos alunos, para mais informações consulte [20].

Segundo Macedo [13], atualmente estamos diante da quarta revolução industrial, também chamada de indústria 4.0, que engloba um amplo sistema de tecnologias tais como, a inteligência artificial, robótica, computação em nuvem, internet das coisas, realidade aumentada e realidade virtual, Big data, impressão 3D e 4D entre outras, que estão mudando a forma de produção e os modelos de negócios no Brasil e no mundo.

Para a atual revolução industrial, segundo Ahrens e Spöttl [4], cinco parâmetros são importantes para a qualificação exigida dos estudantes que ocuparão os postos de trabalhos: integração e comunicação abrangente, conhecimento em automação de sistemas de produção, tomada de decisão, alfabetização digital e gerenciamento interativo, flexibilização do trabalho.

Os estudantes deste século que ocuparão trabalhos modernos precisam dominar a interdisciplinaridade com habilidades em uma variedade de disciplinas teóricas e práticas (ELBESTAWI et al., [10]). Recentemente, Oliveira e Souza [17]-[18], iniciaram os estudos visando a interdisciplinaridade entre o conceito de matrizes com geometria, conseguindo resolver um problema de finança em [17]; e um problema de navegação autônoma de robôs

em [18]. Posteriormente estes trabalhos foram ampliados por Aguiar e Souza [1]-[2]. Em seguida, Aguiar e Souza [3], melhoraram estes trabalhos através de uma abordagem de matrizes de ordem três associada a sólidos geométricos, novamente com uma interdisciplinaridade para resolver desafios da vida prática.

Como podemos perceber, essas inovações tecnológicas estão mudando a dinâmica entre o trabalho humano e os processos controlados por sistemas computacionais, por isso é de fundamental importância que estejamos preparados para formar os estudantes da educação básica para enfrentar essa realidade, de modo que nossos futuros educandos sejam capazes de trabalhar com essas novas tecnologias e quem sabe em um futuro próximo, sejam capazes de desenvolver novas tecnologias importantes para a sociedade.

Ressaltamos que, devido aos custos elevados, muitas destas tecnologias ainda não estão presentes nas salas de aula, mas essa realidade vem mudando e hoje já contamos, por exemplo, com bons softwares de simulação robótica gratuitos, dentre eles o software educacional CoppeliaSim, que será a principal ferramenta deste trabalho. Com a ascensão das tecnologias aplicadas ao ensino, a robótica educacional, vem despertando o interesse de docentes e acadêmicos, que buscam melhorar o desenvolvimento cognitivo dos educandos e atender a uma nova demanda educacional que buscam por atividades mais significativas e inovadoras.

Segundo (Paulo Blikstein, apud Ferraz, [11]), que trabalha com o uso de tecnologias educacionais, acredita que a robótica pode ser usada para ensinar habilidades importantes, como pensamento crítico, resolução de problemas e trabalho em equipe. Nesta perspectiva, podemos pensar na robótica educacional como um recurso educacional capaz de integrar diversas áreas do conhecimento, podendo trazer mudanças na forma de agir, aprender e pensar dos alunos.

A robótica, ao desempenhar o papel de facilitadora na abordagem interdisciplinar de diversos conteúdos na educação básica, incluindo matemática, biologia, física, entre outros, não apenas quebra com o método tradicional, mas também capacita o aluno a adotar uma nova postura diante do conhecimento. Isso ocorre por meio da contextualização, expandindo as fronteiras do aprendizado para além do que é convencionalmente abordado em sala de aula.

> A robótica é um campo de conhecimento interdisciplinar, pois conecta as definições de muitos campos da tecnologia, sendo um campo de aplicação para uso no trabalho educacional. A interligação das capacidades mentais necessárias para a evolução da racionalidade na robótica possibilita que o aluno perceba que os conceitos aprendidos em diferentes disciplinas são utilizáveis e devem ser usados em conjunto para desobstruir obstáculos que não são apresentados no ensino convencional (Silva [21]).

Conforme evidenciado na fala de Silva [21], a robótica pode ser uma grande aliada no processo de ensino e aprendizagem dando significado prático ao que é visto em sala de

aula, sem contar que nas fases do processo, poderão surgir questionamentos e discussões, tornando o processo de ensino e aprendizagem mais efetivos.

Por outro lado, sabemos que as atividades que envolvem robótica em sala de aula podem ser bastante desafiadoras para os docentes, tanto por falta de conhecimento computacional como por falta de material de apoio que lhes sirvam de inspiração e também por necessitar do conhecimento pedagógico tecnológico do conteúdo.

> As atividades que envolvam a robótica na sala de aula podem ser desafiadoras para o professor, seja ele um docente especialista em computação, seja um docente polivalente com formação em educação, haja vista a articulação de diferentes aspectos e saberes relacionados ao uso dessa tecnologia (Campos [8]).

Recentemente, Macedo e Souza [13]-[14], conseguiram uma interdisciplinaridade entre funções polinomiais do primeiro e segundo grau com problemas do mundo real, através do simulador robótico, CoppeliaSim. Mais precisamente, os autores conseguiram resolver um problema de navegação autônoma de robôs, através do gráfico destas funções. Por isto, este trabalho vem como uma possibilidade de se aplicar a robótica educacional em conteúdos do ensino básico, onde utilizamos o software CoppeliaSim, no campo das simulações robóticas envolvendo o estudo de funções polinomiais do terceiro grau.

Abordamos funções polinomiais do terceiro grau, no ambiente de simulação robótica CoppeliaSim, mais precisamente utilizamos o Lumibot que é um dos robôs presentes nesta plataforma. Para isto, utilizaremos a linguagem de programação Python, que será responsável por calcular o polinômio de terceiro grau que servirá de caminho para o robô Lumibot; de tal modo que os estudantes determinarão os coeficientes deste polinômio responsável, influenciando o trajeto do Lumibot.

Neste trabalho propomos alguns desafios ao Lumibot, por meio de cenário de visualização do CoppeliaSim, onde os estudantes serão motivados a resolver estes desafios, apenas com os conhecimentos de polinômios do terceiro grau, bem como de gráficos gerados por estes polinômios, dando um significado prático ao estudo de tais conteúdos.

De acordo com a problematização que já foi esplanada anteriormente, definimos os seguintes objetivos.

Objetivo geral

Este trabalho apresenta como objetivo geral propor um ambiente simulado, fazendo uso da robótica educativa no estudo de funções polinomiais do terceiro grau.

Objetivos específicos

Para que o objetivo geral seja atingido, o trabalho deve alcançar os respectivos objetivos específicos:

• Capacitar o leitor a modelar uma função cúbica por meio da técnica de interpolação

de Lagrange.

• Investigar o método de Cardano-Tartaglia e sua aplicação na resolução de equações cúbicas.

• Criar situações-problema nas quais um robô deve cumprir tarefas específicas pré-determinados pelo usuário.

• Implementar um ambiente para que um robô industrial (Lumibot) percorra pontos pré-determinados no CoppeliaSim, através da linguagem de programação Python.

• Propor simulações robóticas e analise de resultados.

O presente documento foi dividido em cinco capítulos a fim de apresentar a fundamentação teórica, os materiais e métodos utilizados na construção metodológica e os resultados obtidos.

No Capítulo 1, exploramos o polinômio interpolador de Lagrange e sua interpretação geométrica como uma ferramenta de modelagem matemática no CoppeliaSim, com o objetivo de criar trajetórias para o Lumibot. A utilização da interpolação de Lagrange permite a geração de funções que direcionam o Lumibot através de pontos específicos no plano.

No Capítulo 2, estudamos o método de Cardano-Tartaglia para a resolução de funções polinomiais do terceiro grau, abordando tanto a fórmula de Cardano-Tartaglia como também a análise do discriminante gerado pela fórmula.

No Capítulo 3, destacamos algumas aplicabilidades de funções no ambiente CoppeliaSim, bem como os resultados obtidos através de testes executados em ambiente de simulação robótica entre o CoppeliaSim e os polinômios cúbicos escritos no ambiente de programação Python.

Por fim, o Capítulo 4, contém as conclusões deste trabalho.

Capítulo 1

Interpolação de Lagrange

Neste capítulo, exploraremos a interpolação de Lagrange como uma poderosa ferramenta de modelagem matemática no contexto do CoppeliaSim, focando na criação de trajetórias para o Lumibot. A interpolação de Lagrange nos permite gerar funções que guiam o Lumibot através de pontos determinados no plano. Ao estabelecer uma correspondência entre os pontos de referência desejados, podemos criar trajetórias precisas, garantindo que o Lumibot passe exatamente pelos pontos especificados. Ao entender e aplicar a interpolação de Lagrange, expandimos significativamente as capacidades de modelagem e controle do Lumibot no ambiente simulado do CoppeliaSim. Neste contexto, considerando a existência de trabalhos anteriores na literatura que abordam trajetórias de funções afins e quadráticas no CoppeliaSim, conforme Macedo e Souza [13]-[14]. Direcionaremos nosso foco para a modelagem de trajetórias utilizando funções cúbicas.

1.1 Introdução à Interpolação Polinomial

A interpolação polinomial é um método amplamente utilizado para encontrar uma função polinomial que passe exatamente por um conjunto de pontos dados. Uma das técnicas mais conhecidas para realizar essa interpolação é o polinômio interpolador de Lagrange.

Como nosso foco é a educação básica, optamos por não nos aprofundar em abordagens mais avançadas, que podem ser encontradas em trabalhos, como o citado em [12]. Em vez disso, concentraremos nossa atenção na compreensão e aplicação prática do polinômio interpolador de Lagrange, fornecendo uma base sólida para estudantes da educação básica.

1.2 Polinomios de Interpolação

A ideia da interpolação por meio de polinômios consiste em determinar um polinômio $P_n(x)$ de grau máximo n, onde são conhecidos $n+1$ pontos distintos (reais ou complexos) $x_0, x_1, x_2, \ldots, x_n$ e $n+1$ valores de uma função $y = f(x)$, $y_0, y_1, \ldots, y_n$, tal que:

$$P_n(x_0) = y_0;\ P_n(x_1) = y_1;\ \ldots;\ P_n(x_n) = y_n.$$

Teorema 1.2.1 *Dados $n+1$ pontos distintos $x_0, x_1, \ldots, x_n$ (reais ou complexos) e $n+1$ valores $y_0, y_1, \ldots, y_n$, existe um e somente um polinômio $P_n(x)$, de grau menor ou igual a n, tal que:*

$$P_n(x_k) = y_k;\ \text{para } k = 0, 1, 2, \ldots, n. \tag{1.1}$$

Demonstração:

Seja $P_n(x) = a_0 + a_1x + \ldots + a_nx^n$, um polinômio de grau máximo n, com $n+1$ coeficientes $a_0, a_1, \ldots, a_n$ a serem determinados.

Pela equação (1.1), tem-se que:

$$\begin{cases} a_0 + a_1x_0 + \ldots + a_nx_0^n &= y_0 \\ a_0 + a_1x_1 + \ldots + a_nx_1^n &= y_1 \\ \vdots & \\ a_0 + a_1x_n + \ldots + a_nx_n^n &= y_n \end{cases} \tag{1.2}$$

um sistema linear para os coeficientes $a_0, a_1, a_2, \ldots, a_n$ dados a partir de uma matriz V conhecida como matriz de Vandermonde, para mais detalhes consulte [12].

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix}$$

que tem determinante denotado por

$$det(V) = V(x_0, x_1, \ldots, x_n) \tag{1.3}$$

$\det(V(x))$ é um polinômio de grau menor ou igual a n. Além disso, $V(x)$ se anula em $x_0, x_1, \ldots, x_{n-1}$, uma vez que o cálculo desse tipo de determinante pode ser feito a partir do produto de todas as diferenças possíveis entre os elementos da segunda coluna.

Isto é,

$$V(x_0, x_1, \ldots, x_{n-1}, x) = A(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \tag{1.4}$$

onde A depende de $x_0, x_1, \ldots, x_{n-1}$.

Para calcular A, desenvolve-se (1.4) segundo os elementos da última linha e observa-se que o coeficiente de x^n é $V(x_0, x_1, \ldots, x_{n-1})$. Então:

$$V(x_0, x_1, \ldots, x_{n-1}, x) = V(x_0, x_1, \ldots, x_{n-1})(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \tag{1.5}$$

Substituindo x por x_n em (1.5) obtém-se a seguinte fórmula de recorrência:

$$V(x_0, x_1, \ldots, x_{n-1}, x_n) = V(x_0, x_1, \ldots, x_{n-1})(x_n - x_0)(x_n - x_1)\ldots(x_n - x_{n-1}) \tag{1.6}$$

De (1.3), temos que $V(x_0, x_1) = x_1 - x_0$, como em (1.6) pode-se escrever:

$$V(x_0, x_1, x_2) = (x_1 - x_0)(x_2 - x_0)(x_2 - x_1)$$

Por aplicações sucessivas de (1.6), obtém-se:

$$V(x_0, x_1, \ldots, x_n) = \prod_{i>j}(x_i - x_j)$$

Por hipótese, $x_0 \neq x_1 \neq \ldots \neq x_n$. Assim $V \neq 0$ e por Cramer o sistema (1.2) admite uma e somente uma solução $a_0, a_1, \ldots, a_n$. Terminando a prova do resultado .

Concluímos então que a partir de $n+1$ pontos distintos $x_0, x_1, \ldots, x_n$ e $n+1$ valores $f(x_0) = y_0, f(x_1) = y_1, \ldots, f(x_n) = y_n$, de uma função $y = f(x)$, existe apenas um polinômio $P_n(x)$ de grau no máximo n tal que

$$P_n(x_k) = f(x_k); \quad \text{onde } k = 0, 1, \ldots, n.$$

Em suma, como o foco do nosso trabalho é encontrar funções polinomiais cúbicas para modelar trajetorias do Lumibot no CoppeliaSim, este teorema garante que com quatro pontos distintos podemos determinar uma função polinomial cúbica, além disso, ele garante que existe apenas uma função cúbica que passe exatamente por esses quatro pontos específicos, proporcionando uma solução única para o problema da interpolação.

1.3 O Polinômio Interpolador de Lagrange

Agora que foi estabelecida a existência e a unicidade do Polinômio Interpolador, podemos apresentar a fórmula de Lagrange, que fornece um Polinômio Interpolador de uma função real de uma variável a partir de alguns pontos conhecidos.

Sejam dados $n+1$ pontos distintos, que seguem uma função $f(x_n)$, e são denotados por $(x_0, f(x_0))$, $(x_1, f(x_1))$, ..., $(x_n, f(x_n))$, onde $x_0 < x_1 < \ldots < x_n$. Primeiramente, observe os $n+1$ valores obtidos em x_0, x_1, ..., x_n, com $f(x_0) = y_0$, $f(x_1) = y_1$, ..., $f(x_n) = y_n$. Em seguida, para encontrar o polinômio interpolante desejado (que denotamos por $P(x)$), propomos o polinômio de grau n, apresentado da seguinte forma:

$$\begin{aligned} P(x) = {} & K_0 \cdot (x - x_1)(x - x_2)\ldots(x - x_n) \\ & + K_1 \cdot (x - x_0)(x - x_2)\ldots(x - x_n) \\ & \vdots \\ & + K_{n-1} \cdot (x - x_0)(x - x_1)\ldots(x - x_{n-2})(x - x_n) \\ & + K_n \cdot (x - x_0)(x - x_1)\ldots(x - x_{n-1}) \end{aligned} \tag{1.7}$$

e defina os coeficientes constantes K_0, K_1, K_2, K_3, ..., K_{n-1}, K_n para que satisfaçam as seguintes condições:

$$P(x_0) = y_0, \quad P(x_1) = y_1, \quad \ldots, \quad P(x_n) = y_n. \tag{1.8}$$

Assim, substituindo $x = x_0$ na equação (1.7) acima apresentada e, em seguida, ao levar em conta as condições impostas em (1.8), obtemos:

$$y_0 = K_0 \cdot (x_0 - x_1)(x_0 - x_2)\ldots(x_0 - x_n),$$

donde concluímos que

$$K_0 = \frac{y_0}{(x_0 - x_1)(x_0 - x_2)\ldots(x_0 - x_n)}$$

Em seguida, substituindo $x = x_1$ em (1.7) e, ao levar em conta as condições impostas em (1.8), obtemos:

$$y_1 = K_1 \cdot (x_1 - x_0)(x_1 - x_2)\ldots(x_1 - x_n),$$

donde concluímos que

$$K_1 = \frac{y_1}{(x_1 - x_0)(x_1 - x_2)\ldots(x_1 - x_n)}$$

Assim, de maneira análoga à feita para encontrarmos as expressões para as constantes K_0

e K_1, concluímos que

$$\begin{aligned}
K_2 &= \frac{y_2}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)\dots(x_2 - x_n)} \\
K_3 &= \frac{y_3}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)(x_3 - x_4)\dots(x_2 - x_n)} \\
&\vdots \\
K_{n-1} &= \frac{y_{n-1}}{(x_{n-1} - x_0)(x_{n-1} - x_1)\dots(x_{n-1} - x_{n-2})(x_{n-1} - x_n)} \\
K_n &= \frac{y_n}{(x_n - x_0)(x_n - x_1)(x_n - x_2)\dots(x_n - x_{n-1})}
\end{aligned}$$

Consecutivamente, ao substituir os valores dos coeficientes K_0, K_1, K_2, K_3, ..., K_{n-1}, K_n na equação (1.7), obtemos a seguinte fórmula para a expressão algébrica da função $P(x)$ proposta inicialmente:

$$\begin{aligned}
P(x) = &\frac{(x - x_1)(x - x_2)\dots(x - x_n)\cdot}{(x_0 - x_1)(x_0 - x_2)\dots(x_0 - x_n)} y_0 \\
&+ \frac{(x - x_0)(x - x_2)\dots(x - x_n)\cdot}{(x_1 - x_0)(x_1 - x_2)\dots(x_1 - x_n)} y_1 \\
&\vdots \\
&+ \frac{(x - x_0)(x - x_1)\dots(x - x_{n-2})(x - x_n)\cdot}{(x_{n-1} - x_0)(x_{n-1} - x_1)\dots(x_{n-1} - x_{n-2})(x_{n-1} - x_n)} y_{n-1} \\
&+ \frac{(x - x_0)(x - x_1)\dots(x - x_{n-1})\cdot}{(x_n - x_0)(x_n - x_1)(x_n - x_2)\dots(x_n - x_{n-1})} y_n
\end{aligned}$$

A expressão obtida para o polinômio interpolador, $P(x)$, é conhecida como "Fórmula de Interpolação Polinomial de Lagrange", em homenagem ao matemático Joseph Louis Lagrange, que a publicou em 1794. Dada sua relevância, apresentaremos agora sua definição formal.

Definição (Método de Interpolação Polinomial de Lagrange).

Seja um conjunto de $n+1$ dados discretos $y_0, y_1, \dots, y_n$ coletados, respectivamente, em $x_0, x_1, \dots, x_n$, onde $x_0 < x_1 < x_2 < \dots < x_n$ e $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ são $n+1$ pontos que seguem uma função f. Então, existe uma única função polinomial P, de grau menor ou igual que n, que interpola os pontos de f e que possui expressão algébrica dada por:

$$P(x) = \sum_{i=0}^{n} L_i(x) \cdot y_i, \text{ onde } L_i(x) = \frac{\prod_{j=0, j\neq i}^{n}(x - x_j)}{\prod_{j=0, j\neq i}^{n}(x_i - x_j)} \text{ e } 0 \leq i \leq n.$$

Logo, diz-se que a função polinomial $P : \mathbb{R} \to \mathbb{R}$, $P(x) = L_0(x)\cdot y_0 + L_1(x)\cdot y_1 + L_2(x)\cdot y_2 + \dots + L_{n-1}(x)\cdot y_{n-1} + L_n(x)\cdot y_n$ é a função obtida pelo método de interpolação polinomial de Lagrange dos $n+1$ pontos dados que seguem a função f, ou simplesmente, que P é

a função de interpolação polinomial de Lagrange. Em resumo, valendo-se da fórmula do método de interpolação polinomial de Lagrange, temos que o polinômio interpolador, que é a expressão algébrica da função interpolante P, é dado por:

$$P(x) = L_0(x) \cdot y_0 + L_1(x) \cdot y_1 + L_2(x) \cdot y_2 + \ldots + L_{n-1}(x) \cdot y_{n-1} + L_n(x) \cdot y_n,$$

onde

$$0 \leq i \leq n$$

e

$$L_i(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_{n-1})(x - x_n)}{(x_i - x_0)(x_i - x_1)(x_i - x_2)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_{n-1})(x_i - x_n)}.$$

Cabe-nos ressaltar que a verificação de que $P(x)$ é o polinômio interpolador segue diretamente da substituição dos pontos de interpolação e da observação de que

$$L_i(x_j) = \begin{cases} 1, & \text{se } j = i \\ 0, & \text{se } j \neq i \end{cases}$$

e que esse polinomio é único como demonstrado em (1.2).

1.4 Interpretação geométrica

Para demonstrar a eficácia do polinômio interpolador de Lagrange, consideremos um conjunto de quatro pontos dados $(x_0, y_0), (x_1, y_1), (x_2, y_2)$ e (x_3, y_3). Já sabemos que com quatro pontos dados iremos encontrar um polinômio $P(x)$ de grau três, que passe por todos esses pontos.

Usando a fórmula do polinômio interpolador de Lagrange, temos:

$$P(x) = L_0(x) \cdot y_0 + L_1(x) \cdot y_1 + L_2(x) \cdot y_2 + L_3(x) \cdot y_3,$$

Com,

$$
\begin{aligned}
L_0(x) &= \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} \\
L_1(x) &= \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} \\
L_2(x) &= \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} \\
L_3(x) &= \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}
\end{aligned}
$$

Onde temos que

$$
\begin{aligned}
P(x) = &\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}y_0 \\
&+ \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}y_1 \\
&+ \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}y_2 \\
&+ \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}y_3
\end{aligned}
\tag{1.9}
$$

Sejam os quatros pontos dados iguais a $(-1,2),(0,2),(1,-2)$ e $(3,2)$, respectivamente. Iremos encontrar o polinômio interpolador de Lagrande $P(x)$, donde temos

$$
\begin{aligned}
P(x) = &\frac{(x-0)(x-1)(x-3)}{(-1-0)(-1-1)(-1-3)}.2 \\
&+ \frac{(x+1)(x-1)(x-3)}{(0+1)(0-1)(0-3)}.2 \\
&+ \frac{(x+1)(x-0)(x-3)}{(1+1)(1-0)(1-3)}.(-2) \\
&+ \frac{(x+1)(x-0)(x-1)}{(3+1)(3-0)(3-1)}.2
\end{aligned}
$$

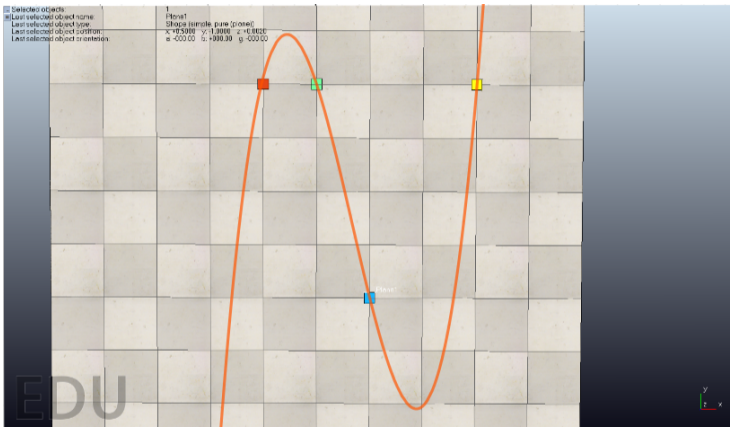
desenvolvendo temos

$$
\begin{aligned}
P(x) &= 2\frac{(x^3-4x^2+3x)}{(-8)} + 2\frac{(x^3-3x^2-x+3)}{3} + (-2)\frac{(x^3-2x^2-3x)}{(-4)} + 2\frac{(x^3-x)}{24} \\
&= \frac{-6(x^3-4x^2+3x)+16(x^3-3x^2-x+3)+12(x^3-2x^2-3x)+2(x^3-x)}{24} \\
&= \frac{24x^3-48x^2-72x+48}{24} \\
&= x^3-2x^2-3x+2
\end{aligned}
$$

Mostraremos o gráfico de $P(x)$ conforme a figura 1.1, no ambiente de simulação do

CoppeliaSim, onde o ponto $(-1, 2)$ será representado por um quadrado na cor vermelha, o ponto $(0, 2)$ por um quadrado na cor verde, o ponto $(1, -2)$ por um quadrado na cor azul e o ponto $(3, 2)$ por um quadrado na cor amarela. Portanto, o polinômio $P(x)$ interpolador

Figura 1.1: gráfico do polinômio $P(x) = x^3 - 2x^2 - 3x + 2$



Fonte: próprio autor.

de Lagrange passa por todos os pontos dados, demonstrando sua eficácia na interpolação polinomial.

Capítulo 2

Método de Cardano-Tartaglia para Equações Cúbicas

Ao nos debruçarmos sobre o estudo das equações polinomiais do Terceiro Grau, observamos que a mesma já era objeto de estudo dos babilônios entre 1800 e 1600 a.c mas, foram os matematicos Scipione Del Ferro (1465-1526), Gerônimo Cardano (1501-1576) e Nicoló Fontana (1499-1557) ‘‘mais conhecido como Tartaglia’’, que deram uma grande contribuição no estudo das equações polinomias do treceiro grau.

Dentre os variados métodos de resolução, daremos destaque a fórmula de Cardano–Tartaglia, por se destacar na resolução de equações polinomiais do terceiro grau sem se limitar a algumas particularidades encontradas em outros métodos algébricos, para mais informações sobre o método de Cardano-Tartaglia consulte [19].

2.1 Método de Cardano-Tartaglia para Equações Cúbicas

Considere a equação cúbica geral:

$$ax^3 + bx^2 + cx + d = 0, \tag{2.1}$$

onde $a \neq 0$, b, c, d são constantes reais. Divida a equação (2.1) por a para obter a forma monicêntrica:

$$x^3 + \frac{b}{a}x^2 + \frac{c}{a}x + \frac{d}{a} = 0 \tag{2.2}$$

A equação (2.2) é equivalente à

$$x^3 + Ax^2 + Bx + C = 0, \tag{2.3}$$

onde, $A = \frac{b}{a}$, $B = \frac{c}{a}$, e $C = \frac{d}{a}$.

Faremos agora uma mudança de variável de modo a eliminar o termo de segundo grau na equação (2.3). Tome,

$$x = y + k$$

e substituindo em

$$x^3 + Ax^2 + Bx + C = 0,$$

teremos:

$$\begin{aligned}(y+k)^3 + A(y+k)^2 + B(y+k) + C = \\ y^3 + 3y^2k + 3yk^2 + k^3 + A(y^2 + 2yk + k^2) + By + Bk + C = \\ y^3 + 3y^2k + 3yk^2 + k^3 + Ay^2 + 2Ayk + Ak^2 + By + Bk + C = 0.\end{aligned}$$

Donde

$$y^3 + (3k + A)y^2 + (3k^2 + 2Ak + B)y + (k^3 + Ak^2 + Bk + C) = 0 \tag{2.4}$$

Como o objetivo é eliminar o termo de segundo grau da equação (2.3), basta considerar $3k + A = 0$ na equação (2.4), e portanto

$$k = -\frac{A}{3}.$$

Dessa forma, substituindo $k = -\frac{A}{3}$ em (2.4), obtemos

$$y^3 + \left(\frac{A^2}{3} - \frac{2A^2}{3} + B\right)y + \left(-\frac{A^3}{27} + \frac{A^3}{9} - \frac{AB}{3} + C\right) = 0.$$

Daí

$$y^3 + \left(B - \frac{A^2}{3}\right)y + \left(\frac{2A^3}{27} - \frac{AB}{3} + C\right) = 0. \tag{2.5}$$

Sejam,

$$p = B - \frac{A^2}{3} \quad \text{e} \quad q = \frac{2A^3}{27} - \frac{AB}{3} + C$$

a equação (2.5) é equivalente a,

$$y^3 + py + q = 0 \tag{2.6}$$

Supondo que a solução da equação (2.6) seja a soma de duas parcelas u e v, ou seja, $y = u + v$, temos:

$$\begin{aligned}(u+v)^3 + p(u+v) + q = \\ u^3 + v^3 + 3uv(u+v) + p(u+v) + q = \\ (u^3 + v^3 + q) + (p + 3uv)(u+v) = 0.\end{aligned}$$

Para que a igualdade seja satisfeita, tomaremos

$$u^3 + v^3 + q = 0 \quad \text{e} \quad p + 3uv = 0.$$

Considerando u^3 e v^3 como raízes de uma equação do segundo grau, temos:

$$S = u^3 + v^3 = -q \quad \text{e,}$$
$$P = u^3 v^3 = -\frac{p^3}{27}.$$

Portanto, o problema se reduz a resolver uma equação do 2º grau da forma

$$w^2 - Sw + P = 0,$$

com a soma S, e o produto P das raízes definidas acima. Segue-se

$$w^2 - Sw + P =$$
$$w^2 - (-q)w + (-\frac{p}{3})^3 = 0.$$

Ou seja,

$$w^2 + qw + (-\frac{p}{3})^3 = 0. \tag{2.7}$$

Resolvendo a equação (2.7), obtemos

$$w = \frac{-q \pm \sqrt{q^2 - 4.1\left(-\frac{p}{3}\right)^3}}{2.1}$$
$$= -\frac{q}{2} \pm \sqrt{\frac{q^2 - 4\left(-\frac{p}{3}\right)^3}{4}}$$
$$= -\frac{q}{2} \pm \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}.$$

Logo

$$w_1 = -\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3} \quad \text{e} \quad w_2 = -\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3},$$

são soluções para a equação (2.7).

Da adaptação que fizemos de soma e produto, seja $u^3 = w_1$ e $v^3 = w_2$, temos:

$$u^3 = -\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}$$

e

$$v^3 = -\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}.$$

Donde

$$u = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}},$$

e

$$v = \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}.$$

Como $y = u + v$, então

$$y = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}.$$

Por fim, como $x = y + k$, com $k = -\frac{A}{3}$, temos:

$$x = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} - \frac{A}{3}.$$

Seja

$$\Delta = \frac{q^2}{4} + \frac{p^3}{27},$$

segue-se que

$$x = \sqrt[3]{-\frac{q}{2} + \sqrt{\Delta}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\Delta}} - \frac{A}{3}.$$

A grande dificuldade encontrada com a fórmula de Cardano-Tartaglia, foi observada por Cardano e denominada como "O caso irredutível", ou seja, caso em que $\Delta < 0$, onde a fórmula levava a raiz quadrada de números negativos; Cardano percebeu que ao resolver a equação $x^3 - 15x - 4 = 0$, onde $\Delta < 0$, tinha como solução $x = 4$, intrigando muitos matemáticos. É importante salientar que nesta época, não se tinha conhecimento de números complexos, ou seja, tanto Cardano como Tartaglia não tinham conhecimentos de números complexos nem de como realizar uma aproximação dos números, visto que em muitos casos a resposta será de valores tão aproximados quanto queremos.

Segundo Boyer [7], Coube a Rafael Bombelli propor uma nova abordagem ao problema de Cardano, considerando as raízes quadradas de números negativos como números verdadeiros e encontrando as outras raízes.

Euller ao determinar as raízes n-ésimas de um números complexo desvendou o grande enigma da fórmula de Cardano-Tartaglia nos casos em que $\Delta < 0$.

Na proxima seção, exploramos o descriminante da equação fornecida pela fórmula de Cardano-Tartaglia, afim de classifica-lá quanto as três raízes existentes.

2.2 Relação entre as raízes de uma equação cúbica e o discriminante Δ.

Nesta seção iremos abordar a classificação das raízes de uma equação cúbica em relação ao conjunto dos números reais ou complexos e para isto iremos utilizar o descriminante da fórmula de Cardano-Tartaglia, onde

$$\Delta = \frac{q^2}{4} + \frac{p^3}{27},$$

Assim como ocorre com uma equação do 2º grau, o valor do discriminante está diretamente relacionado com o número de raízes reais de uma equação cúbica. Como já vimos que toda equação cúbica da forma $ax^3 + bx^2 + cx + d = 0$ pode ser escrita na forma $y^3 + py + q = 0$, com $p, q \in \mathbb{R}$.

Sejam y_1, y_2 e y_3 as raízes da equação:

$$y^3 + py + q = 0, \quad com\ p, q \in \mathbb{R},$$

E usando as relações de Girard (para maiores informações consulte [19]), teríamos:

$$\begin{aligned} y_1 + y_2 + y_3 &= 0, \\ y_1y_2 + y_1y_3 + y_2y_3 &= p, \\ -y_1y_2y_3 &= q. \end{aligned}$$

Em $y_1 + y_2 + y_3 = 0$, temos que $y_2 + y_3 = -y_1$, e substituindo em $y_1y_2 + y_1y_3 + y_2y_3 = p$, teremos:

$$p = y_1(y_2 + y_3) + y_2y_3 = -y_1^2 + y_2y_3.$$

Substituindo em

$$\Delta = \frac{q^2}{4} + \frac{p^3}{27},$$

temos:

$$\Delta = \frac{y_1^2y_2^2y_3^2}{4} + \frac{(y_2y_3 - y_1^2)^3}{27},$$

2.2.1 A equação cúbica tem uma raiz real e duas raízes complexas, implica $\Delta > 0$

Seja $y_1 = m + ni$ uma raiz da equação $y^3 + py + q = 0$, logo seu conjugado também será raiz, ou seja, $y_2 = m - ni$, por fim seja a outra raiz $y_3 = k$, com $p, q, m, k \in \mathbb{R}$ e $n \in \mathbb{R}^*$

como $y_1 + y_2 + y_3 = 0$, implica em $2m + k = 0$ e consequentemente $y_3 = -2m$.

Em $p = y_1y_2 + y_1y_3 + y_2y_3 = y_1y_2 + y_3(y_1 + y_2)$, teremos que

$$(m^2 + n^2) - 2m(2m) = n^2 - 3m^2,$$

e consequentemente $(\frac{p}{3})^3 = (\frac{n^2}{3} - m^2)^3$.

Do fato de $q = -y_1y_2y_3$, teremos $q = 2m(m^2 + n^2)$ e consequentemente

$$(\frac{q}{2})^2 = m^2(m^2 + n^2)^2.$$

Por fim, temos que:

$$\begin{aligned}\Delta = \frac{q^2}{4} + \frac{p^3}{27} &= (\frac{q}{2})^2 + (\frac{p}{3})^3 \\ &= m^2(m^2 + n^2)^2 + (\frac{n^2}{3} - m^2)^2 \\ &= 3m^4n^2 + \frac{2}{3}m^2n^4 + \frac{n^6}{27}\end{aligned}$$

Como $3m^4n^2 \geq 0$, $\frac{2}{3}m^2n^4 \geq 0$ e $\frac{n^6}{27} > 0$, visto que $n \in \mathbb{R}^*$, concluimos que $\Delta > 0$. Para mais informações consulte [5].

O exemplo a seguir ilustra o caso em $\Delta > 0$, e a equação cúbica terá como solução uma raiz real e duas raízes complexas, faremos um exemplo.

Exemplo: Resolva a equação $x^3 - 6x - 40 = 0$.

Inicialmente note que $p = -6$ e $q = -40$, como $\Delta = \frac{q^2}{4} + \frac{p^3}{27} = \frac{(-40)^2}{4} + \frac{(-6)^3}{27} = 392 >$

0, pelo método de Cardano-Tartaglia sua raiz é dada por:

$$\begin{aligned}
x_1 &= \sqrt[3]{-\frac{q}{2}+\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}}+\sqrt[3]{-\frac{q}{2}-\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}}-\frac{A}{3}\\
&= \sqrt[3]{-\frac{(-40)}{2}+\sqrt{\frac{(-40)^2}{4}+\frac{(-6)^3}{27}}}+\sqrt[3]{-\frac{-40}{2}-\sqrt{\frac{(-40)^2}{4}+\frac{(-6)^3}{27}}}\\
&= \sqrt[3]{20+\sqrt{392}}+\sqrt[3]{20-\sqrt{392}}\\
&= \sqrt[3]{(2+\sqrt{2})^3}+\sqrt[3]{(2-\sqrt{2})^3}\\
&= 2+\sqrt{2}+2-\sqrt{2}\\
&= 4.
\end{aligned}$$

Para obter as raízes complexas, basta dividir $x^3-6x-40=0$ por $(x-4)$, e usar a fórmula quadrática, onde teremos como solução:

$$\begin{aligned}
&x_1 = 4 \quad \text{(Raiz real)},\\
&x_2 = -2+\sqrt{6i} \quad \text{(Raiz complexa conjugada 1)},\\
&x_3 = -2-\sqrt{6i} \quad \text{(Raiz complexa conjugada 2)}.
\end{aligned}$$

2.2.2 A equação cúbica possui três raízes reais, onde pelo menos duas são iguais implica que $\Delta = 0$

Sejam y_1, y_2 e y_3 as raízes da equação $y^3+py+q=0$, com $p, q, \in \mathbb{R}$. Das relações de Girard, temos:

$$\begin{aligned}
y_1+y_2+y_3 &= 0,\\
y_1y_2+y_1y_3+y_2y_3 &= p,\\
-y_1y_2y_3 &= q.
\end{aligned}$$

De $y_1+y_2+y_3=0$ e supondo $y_1=y_2$, teremos $y_3=-2y_1$. Substituindo no discriminante temos:

$$\begin{aligned}
\Delta &= \frac{y_1^2y_2^2y_3^2}{4}+\frac{(y_2y_3-y_1^2)^3}{27}\\
&= \frac{y_1^2y_1^2(-2y_1)^2}{4}+\frac{(-y_1y_1-y_1^2)^3}{27}\\
&= \frac{4y_1^6}{4}+\frac{(-3y_1^2)^3}{27}\\
&= y_1^6-y_1^6=0
\end{aligned}$$

Portanto, $\Delta = 0$. Para maiores informações consulte [5].

O exemplo a seguir ilustra o caso quando $\Delta = 0$, e a equação cúbica terá como solução três raízes reais com pelo menos duas iguais.

Exemplo: Resolva a equação $x^3 + 4x^2 - 91x - 490 = 0$.

Inicialmente note que $p = -\frac{289}{3}$ e $q = -\frac{9826}{27}$, como $\Delta = \frac{q^2}{4} + \frac{p^3}{27} = \frac{(-\frac{9826}{27})^2}{4} + \frac{(-\frac{289}{3})^3}{27} = 0$, pelo método de Cardano-Tartaglia sua raiz é dada por:

$$\begin{aligned}
x_1 &= \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} - \frac{A}{3} \\
&= \sqrt[3]{-\frac{(-\frac{9826}{27})}{2} + \sqrt{\frac{(-\frac{9826}{27})^2}{4} + \frac{(-\frac{289}{3})^3}{27}}} + \sqrt[3]{-\frac{(-\frac{9826}{27})}{2} - \sqrt{\frac{(-\frac{9826}{27})^2}{4} + \frac{(-\frac{289}{3})^3}{27}}} - \frac{4}{3} \\
&= \sqrt[3]{\frac{4913}{27} + \sqrt{0}} + \sqrt[3]{\frac{4913}{27} - \sqrt{0}} - \frac{4}{3} \\
&= \sqrt[3]{(\frac{17}{3})^3} + \sqrt[3]{(\frac{17}{3})^3} - \frac{4}{3} \\
&= \frac{17}{3} + \frac{17}{3} - \frac{4}{3} \\
&= 10.
\end{aligned}$$

Para obter as outras raízes, basta dividir $x^3 + 4x^2 - 91x - 490 = 0$ por $(x - 10)$, onde obteremos $x^3 + 4x^2 - 91x - 490 = (x - 10)(x^2 + 14x + 49)$ e usar a fórmula quadrática, onde teremos como solução:

$$\begin{aligned}
&x_1 = 10 \quad \text{(Raiz real)}, \\
&x_2 = x_3 = -7 \quad \text{(Raízes reais de multiplicidade dois)}.
\end{aligned}$$

2.2.3 A equação cúbica possui três raízes reais distintas implica que $\Delta < 0$

Sejam y_1, y_2 e y_3 as raízes reais e distintas da equação $y^3 + py + q = 0$, com $p, q, \in \mathbb{R}$. Por Girard, temos

$$y_1 + y_2 + y_3 = 0,$$

donde

$$(y_2 + y_3)^2 = y_1^2,$$

e através da substituição no discriminante obtemos:

$$\begin{aligned}\Delta &= \frac{y_1^2y_2^2y_3^2}{4} + \frac{(y_2y_3 - y_1^2)^3}{27} \\ &= \frac{((y_2+y_3)y_2y_3)^2}{4} + \frac{(y_2y_3 - (y_2+y_3)^2)^3}{27} \\ &= (\frac{y_2y_3(y_2+y_3)}{2})^2 + (\frac{y_2y_3 - (y_2+y_3)^2}{3})^3\end{aligned}$$

Desenvolvendo as potências e reagrupando os termos, teremos:

$$\Delta = -\frac{(y_2-y_3)^2(2y_2+y_3)^2(y_2+2y_3)^2}{108}$$

Donde, concluímos que $\Delta < 0$. Para mais informações consulte [5].

O exemplo a seguir ilustra o caso em que $\Delta < 0$, e a equação cúbica tem como solução três raízes reais distintas.

Exemplo: Resolva a equação $x^3 - 3x^2 - 3x + 9 = 0$.

Inicialmente note que $p = -6$ e $q = 4$, como $\Delta = \frac{q^2}{4} + \frac{p^3}{27} = \frac{4^2}{4} + \frac{(-6)^3}{27} = -4 < 0$, pelo método de Cardano-Tartaglia sua raiz é dada por:

$$\begin{aligned}x_1 &= \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} - \frac{A}{3} \\ &= \sqrt[3]{-\frac{4}{2} + \sqrt{\frac{4^2}{4} + \frac{(-6)^3}{27}}} + \sqrt[3]{-\frac{4}{2} - \sqrt{\frac{4^2}{4} + \frac{(-6)^3}{27}}} - \frac{(-3)}{3} \\ &= \sqrt[3]{-2 + \sqrt{-4}} + \sqrt[3]{-2 - \sqrt{-4}} + 1 \\ &= \sqrt[3]{-2+2i} + \sqrt[3]{-2-2i} + 1 \\ &= \sqrt[3]{(1+i)^3} + \sqrt[3]{(1-i)^3} + 1 \\ &= 2 + 1 \\ &= 3\end{aligned}$$

Para obter as outras raízes, basta dividir $x^3 - 3x^2 - 3x + 9 = 0$ por $(x - 3)$, onde obteremos $x^3 - 3x^2 - 3x + 9 = (x - 3)(x^2 - 3)$ e usar a fórmula quadrática, onde teremos como solução:

$$\begin{aligned}x_1 &= 3 \quad \text{(Raiz real 1)}, \\ x_2 &= \sqrt{3} \quad \text{(Raiz real 2)}, \\ x_3 &= -\sqrt{3} \quad \text{(Raiz real 3)}.\end{aligned}$$

Quando $\Delta < 0$, temos que a equação cúbica possui três raízes reais distintas, como queríamos demonstrar.

2.3 Lei de formação de uma função dado seus pontos

No estudo do gráfico de funções cúbicas é interessante frisarmos que dados alguns pontos no plano podemos definir sua lei de formação. Este método pode ser de grande valia visto que podemos modelar a rota do Lumibot, de modo a desviarmos de obstáculos bem como determinar os pontos de partida e chegada.

2.3.1 Dados três raízes e um ponto

É possível determinar a lei de formação de uma função cúbica quando as raízes são conhecidas, juntamente com um ponto adicional. Uma função cúbica geral pode ser representada na forma:

$$f(x) = a(x - r_1)(x - r_2)(x - r_3),$$

onde r_1, r_2 e r_3 são as raízes da função e a é o coeficiente líder (o coeficiente do termo x^3).

Se você também tem um ponto (x_0, y_0) no gráfico da função, você pode usar esse ponto para determinar o valor de a.

$$y_0 = a(x_0 - r_1)(x_0 - r_2)(x_0 - r_3)$$

Depois de encontrar o valor de a, você pode escrever a lei de formação completa da função cúbica.

2.3.2 Dados quatro pontos no gráfico da função

Dados quatro pontos no gráfico da função, podemos determinar os coeficientes de uma função cúbica. Pelo método da substituição.

A forma geral de uma função cúbica é $f(x) = ax^3 + bx^2 + cx + d$, Se tivermos quatro pontos $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ e (x_4, y_4), pode-se definir um sistema de quatro equações para encontrar os coeficientes a, b, c e d:

$$\begin{cases} ax_1^3 + bx_1^2 + cx_1 + d = y_1 \\ ax_2^3 + bx_2^2 + cx_2 + d = y_2 \\ ax_3^3 + bx_3^2 + cx_3 + d = y_3 \\ ax_4^3 + bx_4^2 + cx_4 + d = y_4 \end{cases}$$

Este é um sistema linear de quatro equações com quatro incógnitas. Resolver esse sistema fornecerá os valores de a, b, c e d, permitindo que você determine a função cúbica específica que passa pelos quatro pontos dados.

O Teorema a seguir, embora não consiga classificar todas as funções polinomiais do

terceiro grau que passa por três pontos, tem a função de produzir algumas funções, conhecendo-se uma única raiz e dois pontos quaisquer sobre o gráfico da função.

Teorema 2.3.1 *Sejam $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ e $P_3 = (x_3, y_3)$ pontos distintos sobre o gráfico da função $f(x) = ax^3 + bx^2 + cx + d$. Se $f(x_1) = 0$, então os coeficientes a, b , c e d ficam completamente determinados. Ou seja,*

(i)

$$a = \frac{y_2(x_3 - x_2)}{[x_2^3(x_3 - x_2) - x_1(x_3^3 - x_2^3) + x_1(x_3^3 - x_2^3) - x_1^3(x_3 - x_2)]}. \tag{2.8}$$

$$b = 0. \tag{2.9}$$

$$c = \frac{y_3}{(x_3 - x_2)} - \frac{y_2(x_3^3 - x_2^3)}{[x_2^3(x_3 - x_2) - x_1(x_3^3 - x_2^3) + x_1(x_3^3 - x_2^3) - x_1^3(x_3 - x_2)]} \tag{2.10}$$

$$d = \frac{y_2[x_1(x_3^3 - x_2^3) - x_1^3(x_3 - x_2)]}{[x_2^3(x_3 - x_2) - x_1(x_3^3 - x_2^3) + x_1(x_3^3 - x_2^3) - x_1^3(x_3 - x_2)]} - \frac{x_1 y_3}{(x_3 - x_2)}, \tag{2.11}$$

desde que

$$[x_2^3(x_3 - x_2) - x_1(x_3^3 - x_2^3) + x_1(x_3^3 - x_2^3) - x_1^3(x_3 - x_2)] \neq 0.$$

(ii)

$$a = \frac{(x_2^3 - 3x_1x_2^2 + 2x_1^3)(x_3 - x_2) + (x_2 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)}{y_2(x_3 - x_2) - (x_2 - 3x_1)(y_3 - y_2)}. \tag{2.12}$$

$$b = \frac{-3x_1[(x_2^3 - 3x_1x_2^2 + 2x_1^3)(x_3 - x_2) + (x_2 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)]}{y_2(x_3 - x_2) - (x_2 - 3x_1)(y_3 - y_2)}. \tag{2.13}$$

$$c = \frac{y_3 - y_2}{x_3 - x_2} + \frac{[(x_2^3 - 3x_1x_2^2 + 2x_1^3)(x_3 - x_2) + (x_2 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)][x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)]}{(y_2(x_3 - x_2) - (x_2 - 3x_1)(y_3 - y_2))(x_3 - x_2)} \tag{2.14}$$

$$d = \frac{2x_1^3[(x_2^3 - 3x_1x_2^2 + 2x_1^3)(x_3 - x_2) + (x_2 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)]}{y_2(x_3 - x_2) - (x_2 - 3x_1)(y_3 - y_2)} - \frac{x_1[y_3 - y_2 + (x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2))]}{x_3 - x_2}, \tag{2.15}$$

desde que $y_2(x_3 - x_2) - (x_2 - 3x_1)(y_3 - y_2) \neq 0$.

(iii)

$$a = \frac{(x_3^3 - 3x_1x_3^2 + 2x_1^3)(x_3 - x_2) + (x_3 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)}{y_3(x_3 - x_2) - (x_3 - 3x_1)(y_3 - y_2)}. \tag{2.16}$$

$$b = \frac{-3x_1[(x_3^3 - 3x_1x_3^2 + 2x_1^3)(x_3 - x_2) + (x_3 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)]}{y_3(x_3 - x_2) - (x_3 - 3x_1)(y_3 - y_2)}. \tag{2.17}$$

$$c = \frac{y_3 - y_2}{x_3 - x_2} + \frac{[(x_3^3 - 3x_1x_3^2 + 2x_1^3)(x_3 - x_2) + (x_3 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)][x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)]}{(y_3(x_3 - x_2) - (x_3 - 3x_1)(y_3 - y_2))(x_3 - x_2)} \tag{2.18}$$

$$d = \frac{2x_1^3[(x_3^3 - 3x_1x_3^2 + 2x_1^3)(x_3 - x_2) + (x_3 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)]}{y_3(x_3 - x_2) - (x_3 - 3x_1)(y_3 - y_2)} - \frac{x_1[y_3 - y_2 + (x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2))]}{x_3 - x_2}. \tag{2.19}$$

desde que $y_3(x_3 - x_2) - (x_3 - 3x_1)(y_3 - y_2) \neq 0$,

Prova

Dividiremos a prova em dois casos. Inicialmente consideraremos $b = 0$. Neste caso, obtemos um sistema linear com três equações e três incógnitas

$$\begin{cases} ax_1^3 + cx_1 + d = 0 \\ ax_2^3 + cx_2 + d = y_2 \\ ax_3^3 + cx_3 + d = y_3 \end{cases}$$

Resolvendo o sistema acima obtemos que,

$$a(x_3^3 - x_2^3) + c(x_3 - x_2) = y_3, \tag{2.20}$$

daí

$$c = \frac{y_3 - a(x_3^3 - x_2^3)}{(x_3 - x_2)}. \tag{2.21}$$

Substituindo o valor de c na primeira equação do sistema acima, temos que

$$d = \frac{a[x_1(x_3^3 - x_2^3) - x_1^3(x_3 - x_2)] - x_1 y_3}{(x_3 - x_2)}. \tag{2.22}$$

Substituindo c e d na segunda equação do sistema, segue-se

$$a = \frac{y_2(x_3 - x_2)}{[x_2^3(x_3 - x_2) - x_1(x_3^3 - x_2^3) + x_1(x_3^3 - x_2^3) - x_1^3(x_3 - x_2)]}, \tag{2.23}$$

desde que

$$[x_2^3(x_3 - x_2) - x_1(x_3^3 - x_2^3) + x_1(x_3^3 - x_2^3) - x_1^3(x_3 - x_2)] \neq 0.$$

Para o caso geral, temos

$$\sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} = \frac{A}{3} + x_1. \tag{2.24}$$

$$\tag{2.25}$$

Daí

$$\left[\sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}\right]^3 = [\frac{A}{3} + x_1]^3. \tag{2.26}$$

$$\tag{2.27}$$

Isto é

$$\begin{aligned} &-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}} + 3\left[\sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}\right]^2 \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} \\ &+3\left[\sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}\right]^2 \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \left[-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}\right] \\ &= [\frac{A}{3} + x_1]^3. \end{aligned} \tag{2.28}$$

Equivalente

$$-\frac{q}{2}+\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}-3\frac{p}{3}\left[\sqrt[3]{-\frac{q}{2}+\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}}\right]-3\frac{p}{3}\left[\sqrt[3]{-\frac{q}{2}-\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}}\right]+\left[-\frac{q}{2}-\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}\right]$$
$$=[\frac{A}{3}+x_1]^3 \tag{2.29}$$

Por (2.24), obtemos

$$-\frac{q}{2}+\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}-3\frac{p}{3}\left[\frac{A}{3}-x_1\right]+\left[-\frac{q}{2}-\sqrt{\frac{q^2}{4}+\frac{p^3}{27}}\right]=[\frac{A}{3}+x_1]^3 \tag{2.30}$$
$$\tag{2.31}$$

Ou seja

$$q+p\left[\frac{A}{3}-x_1\right]+\left[\frac{A}{3}+x_1\right]^3=0 \tag{2.32}$$
$$\tag{2.33}$$

é uma função polinominal do terceiro grau em

$$\left[\frac{A}{3}+x_1\right].$$

Temos três casos a considerar para a equação acima.
Caso A. $q=p=\frac{A}{3}-x_1=0$, donde caímos no caso inicial acima.
Caso B. $q=\frac{A}{3}+x_1=0$ e $p\neq 0$. Neste caso,

$$b=-3x_1a$$

e

$$d=-x_1c+2x_1^3a.$$

Por outro lado,

$$\begin{cases} ax_1^3 + bx_1^2 + cx_1 + d = 0 \\ ax_2^3 + bx_2^2 + cx_2 + d = y_2 \\ ax_3^3 + bx_3^2 + cx_3 + d = y_3 \end{cases} \tag{2.34}$$

Ou seja,

$$\begin{cases} ax_2^3 - 3x_1ax_2^2 + cx_2 + 2ax_1^3 - 3cx_1 = y_2 \\ ax_3^3 - 3x_1ax_3^2 + cx_3 + 2ax_1^3 - cx_1 = y_3 \end{cases} \tag{2.35}$$

Como $x_1 \neq 0$, obtemos que

$$c = \frac{y_3 - y_2 + a(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2))}{x_3 - x_2}$$

Donde, ou

$$a = \frac{(x_2^3 - 3x_1x_2^2 + 2x_1^3)(x_3 - x_2) + (x_2 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)}{y_2(x_3 - x_2) - (x_2 - 3x_1)(y_3 - y_2)},$$

desde que $y_2(x_3 - x_2) - (x_2 - 3x_1)(y_3 - y_2) \neq 0$,

$$b = \frac{-3x_1[(x_2^3 - 3x_1x_2^2 + 2x_1^3)(x_3 - x_2) + (x_2 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)]}{y_2(x_3 - x_2) - (x_2 - 3x_1)(y_3 - y_2)}$$

e

$$\begin{aligned} d = & \frac{2x_1^3[(x_2^3 - 3x_1x_2^2 + 2x_1^3)(x_3 - x_2) + (x_2 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)]}{y_2(x_3 - x_2) - (x_2 - 3x_1)(y_3 - y_2)} \\ & - \frac{x_1[y_3 - y_2 + (x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2))]}{x_3 - x_2}. \end{aligned} \tag{2.36}$$

ou

$$a = \frac{(x_3^3 - 3x_1x_3^2 + 2x_1^3)(x_3 - x_2) + (x_3 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)}{y_3(x_3 - x_2) - (x_3 - 3x_1)(y_3 - y_2)},$$

desde que $y_3(x_3 - x_2) - (x_3 - 3x_1)(y_3 - y_2) \neq 0$,

$$b = \frac{-3x_1[(x_3^3 - 3x_1x_3^2 + 2x_1^3)(x_3 - x_2) + (x_3 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)]}{y_3(x_3 - x_2) - (x_3 - 3x_1)(y_3 - y_2)}$$

e

$$\begin{aligned} d = & \frac{2x_1^3[(x_3^3 - 3x_1x_3^2 + 2x_1^3)(x_3 - x_2) + (x_3 - 3x_1)(x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2)]}{y_3(x_3 - x_2) - (x_3 - 3x_1)(y_3 - y_2)} \\ & - \frac{x_1[y_3 - y_2 + (x_2^3 - x_3^3 - 3x_1(x_2^2 - x_3^2))]}{x_3 - x_2}. \end{aligned} \tag{2.37}$$

Caso C. $\frac{A}{3} - x_1 \neq 0$ e $p \neq 0$. Neste caso, por Cardano-Tartaglia, temos

$$\left[\frac{A}{3} - x_1\right] = \sqrt[3]{-\frac{q}{2} + \sqrt{\Delta}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\Delta}}.$$

Assim, obtemos enumeras soluções.

Capítulo 3

Relação entre CoppeliaSim e Funções Cúbicas

Neste capítulo abordaremos a relação entre o CoppeliaSim e o gráfico das funções cúbicas bem como a aplicação de tais funções para modelar o trajeto do Lumibot, proporcionando movimentos mais naturais e realistas.

3.1 Relação entre CoppeliaSim e Funções Cúbicas

Apesar de o CoppeliaSim não ser uma ferramenta dedicada ao estudo de funções cúbicas, ele destaca-se quando o objetivo é a aplicação prática de tais funções em situações simuladas de controle de robôs. Isso inclui a definição de trajetórias suaves e controladas, tornando a simulação de robôs mais precisa e realista.

O software proporciona uma perspectiva visual sobre como as funções cúbicas são utilizadas para controlar movimentos, delinear trajetórias e modelar o comportamento de sistemas dinâmicos, sendo de grande valia em ambientes de engenharia, robótica e automação.

Aqui estão algumas maneiras como o estudo de funções cúbicas e o CoppeliaSim podem se relacionar:

1. **Visualização de Gráficos de Funções Cúbicas:**

   • O CoppeliaSim permite criar ambientes 3D interativos em que podemos modelar e visualizar gráficos de funções cúbicas, podendo alterar os parâmetros das funções e observar como essas mudanças afetam os gráficos e o comportamento dos objetos no ambiente virtual. Isso permite uma exploração ativa das relações entre as variáveis.

2. **Simulação de Máquinas e Robôs Controlados por Funções Cúbicas:**

• Alunos e professores podem criar robôs ou máquinas virtuais e programá-los para seguir trajetórias definidas por funções cúbicas. Isso envolve a aplicação prática dos conceitos matemáticos, tornando o aprendizado mais prático e significativo.

3. **Processo de ensino e aprendizagem**

   • O CoppeliaSim pode ser usado como uma ferramenta de ensino para demonstrar conceitos de matemática aplicada, como funções cúbicas, de maneira visual e interativa. Isso torna o aprendizado desses conceitos mais envolvente.

4. **Ensino interdisciplinar**

   • O CoppeliaSim é uma ferramenta interdisciplinar que pode ser usada para integrar conceitos de matemática com outras disciplinas, como Biologia, ciência da computação, física e engenharia.

3.2 Simulação de trajetórias do Lumibot por função cúbica

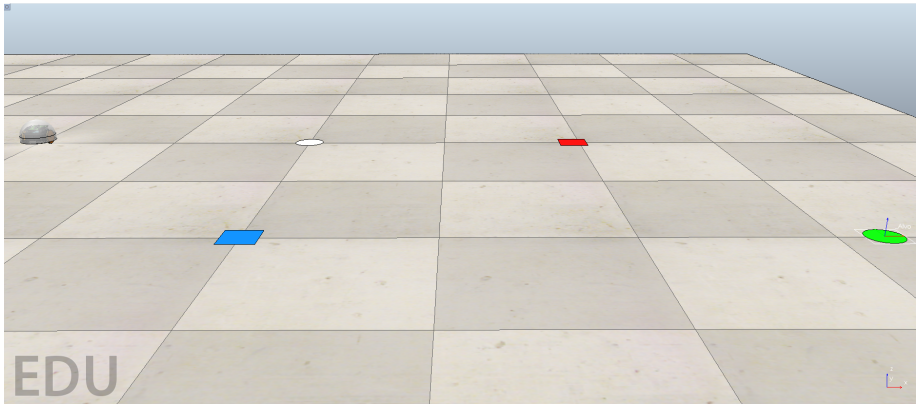
Nesta seção utilizaremos o robô móvel, Lumibot, do CoppeliaSim para percorrer trajetórias definidas por funções cúbicas, de modo a dar significado prático ao estudo de tais funções tornando o processo de ensino aprendizagem mais prático e significativo. Dividiremos esta seção em subseções, onde as três primeiras serão dedicada aos cenários que fornecem os nomes para as subseções.

3.2.1 Cenário 1

Para este cenário, propomos uma atividade simples. Inicialmente, selecionaremos quatro pontos no plano, e utilizando os conhecimentos adquiridos no capítulo 1, buscaremos identificar a lei de formação correspondente. O primeiro ponto será representado pela localização do robô Lumibot, o segundo ponto será associado a um quadrado de cor azul, o terceiro ponto estará relacionado a outro quadrado, desta vez de cor vermelha, e, por último, o Lumibot percorrerá um ponto correspondente a um disco de cor verde, conforme a Figura. 3.1.

Ao realizar essa atividade, pretendemos aplicar os conceitos aprendidos anteriormente para determinar a função que descreve a posição ou características desses elementos no plano. Este exercício não apenas consolidará os conhecimentos adquiridos, mas também proporcionará uma oportunidade prática de aplicação dos métodos discutidos no capítulo anterior.

Figura 3.1: Cenário com o Lumibot, dois quadrados e dois discos



Fonte: próprio autor.

Levando em consideração que as dimensões dos lados dos quadrados que compõe o piso medem 0,5 metros, e que o disco branco é o centro do plano cartesiano, temos que o robô Lumibot ocupa a posição $S_i = (-1, 0)$; o quadrado de cor azul ocupa a posição $Q = (0, -1)$; o quadrado de cor vermelha ocupa a posição $R = (1, 0)$ e o disco de cor verde ocupa a posição $S_f = (1.62, -1)$.

Queremos que o robô saia da sua posição inicial S_i, passe pelos quadrados azul e vermelho e pare sobre o disco verde, como ilustra a Figura 3.1. Como o objetivo é estimular os estudantes a utilizarem o pensamento matemático para resolverem este problema, então pensando matematicamente, propomos uma solução para este problema real, através do gráfico de uma função polinominal do terceiro grau. Ou seja, se o robô seguir a trajetória descrita pelo gráfico desta função, o mesmo consegue passar por todos os locais desejados até atingir o seu alvo.

Pelo método da interpolação de Lagrange, sabemos que dados quatro pontos no plano, que no nosso caso são os pontos: $S_i = (-1, 0)$, $Q = (0, -1)$, $R = (1, 0)$, $S_f = (1, 62, -1)$, respectivamente. Conseguimos modelar uma função cúbica utilizando a fórmula obtida em 1.9, donde:

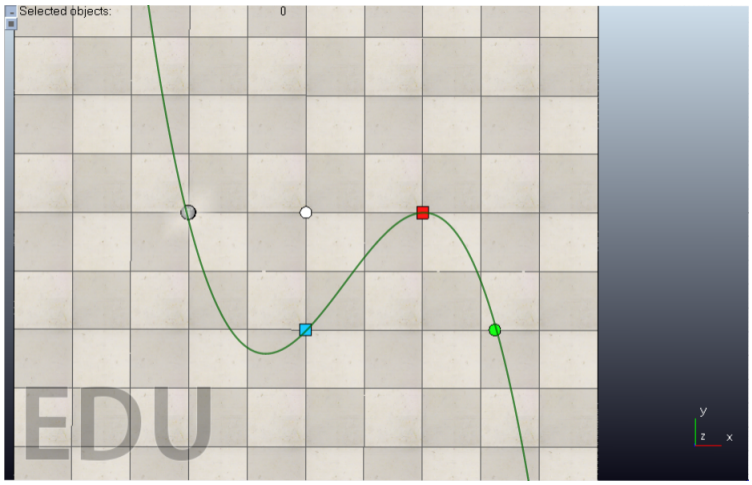
$$\begin{aligned} P(x) = &\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}y_0 + \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}y_1 \\ &+ \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}y_2 + \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}y_3 \end{aligned} \tag{3.1}$$

Substituindo os pontos dados, temos

$$P(x) = \frac{(x-0)(x-1)(x-1.62)}{(-1-0)(-1-1)(-1-1.62)}.0 + \frac{(x+1)(x-1)(x-1.62)}{(0+1)(0-1)(0-1.62)}.(-1) + \frac{(x+1)(x-0)(x-1.62)}{(1+1)(1-0)(1-1.62)}.0 + \frac{(x+1)(x-0)(x-1)}{(1.62+1)(1.62-0)(1.62-1)}.(-1) \quad (3.2)$$

Resolvendo, obtemos a seguinte função cúbica, $P(x) = -x^3+x^2+x-1$. Mostraremos o gráfico de $P(x)$, no ambiente de simulação do CoppeliaSim, conforme a Figura 3.2.

Figura 3.2: Gráfico da função que sera seguido pelo Lumibot



Fonte: próprio autor.

Abaixo apresentamos a linguagem de programação Python que foi escrita no editor de texto PyCharm, onde cada linha de comando foi numerada e em seguida acompanhada de cada frase do código utilizado, essa numeração não deve ser digitada junto com o código, serve apenas para nos localizarmos, e o código deve ser digitado da maneira como ele aparece, conforme em [13]-[14].

```
1 # coding=utf-8
2# Insert in a script in Coppelia
3# simRemoteApi.start(19999)
4 import symtable
5 try:
6     import sim
7 except:
8     print (")
9 import math
10 import time
11 print ('Programa Iniciado')
12 sim.simxFinish(-1)
13 clientID =sim.simxStart('127.0.0.1',19999,True,True,5000,5)
```

14 robotname = 'lumibot'
15 if clientID != -1:
16 time.sleep(2)
17# Coletar handles
18 [erro, robot] = sim.simxGetObjectHandle(clientID, robotname, sim.simx$_o$pmode$_o$neshot$_w$ait)
19 $[erro, robotLeftMotor] = sim.simxGetObjectHandle(clientID, robotname+'_l eftMotor', sim.simx_opmode_oneshot_wait)$
20$[erro, robotRightMotor] = sim.simxGetObjectHandle(clientID, robotname+'_r ightMotor', sim.simx_opmode_oneshot_wait)$
21$\#Criarstreamdedados$
22 $[erro, positionrobot] = sim.simxGetObjectPosition(clientID, robot, -1, sim.simx_opmode_streaming)$
23 $[erro, orientationrobot] = sim.simxGetObjectOrientation(clientID, robot, -1, sim.simx_opmode_streaming)$
24 $time.sleep(2)$
25$\#comandosdosmotores$
26 $sim.simxSetJointTargetVelocity(clientID, robotRightMotor, 0.0, sim.simx_opmode_oneshot)$
27 $sim.simxSetJointTargetVelocity(clientID, robotLeftMotor, 0.0, sim.simx_opmode_oneshot)$
28 $state =' stopped'$
29 $dist_set = 0.1$
30 $vref = 2.0$
31 $running = True$
32 $while(running) :$
33$\#Coletarascoordenadasdorob$
34 $[erro, [xr, yr, zr]] = sim.simxGetObjectPosition(clientID, robot, -1, sim.simx_opmode_buffer)$
35 $[erro, [alpha, beta, gamma]] = sim.simxGetObjectOrientation(clientID, robot, -1, sim.simx_opmode_buffer)$
36$\#Coletadedadosdafuno$
37 ap = 0.16
38 bp = 0
39 cp = -0.16
40 dp = 0
41 xi = -1
42 xf = 2
43 yi = 0
$44 * 0,5cm$yf = 1

```
45  xp = xr
46  yp = yr+0.84
47  dist = math.sqrt((xr - xf) * (xr - xf) + (yr - yf) * (yr - yf))
48  fxp = ap*xp*xp*xp+bp*xp*xp+cp*xp+dp
49  dr = 27.0
50#Transies
51  if(state=='stopped')&(dist > dist_set):
52   state = 'align'
53  elif (state == 'align') & (gamma < math.pi/2):
54  state = 'forward'
55   elif (state == 'forward') & (dist < dist_set):
56   state = 'stopped'
57   #Ações
58   if state == 'stopped':
59  vRightMotor = 0.0
60  vLeftMotor = 0.0
61  running = False
62  elif state == 'align':
63  vRightMotor = -vref
64  vLeftMotor = vref
65  elif state =='forward':
66  if (ap*bp*cp*dp == 0):
67      gxp = 1/(math.sqrt(abs((-3*ap*xp*xp-2*bp*xp-5*cp)*(-3*ap*xp*xp-2*bp*xp-
5*cp)+1)))
68      Gxp = -(xf-xi)/(abs(xf-xi))*(yp-ap*xp*xp*xp-bp*xp*xp-5*cp*xp-dp)
69      Hxp = -(xf-xi)/(abs(xf-xi))*math.sqrt(abs(1-(yp-ap*xp*xp*xp-bp*xp*xp-5*cp*xp-
dp)*(yp-ap*xp*xp*xp-bp*xp*xp-5*cp*xp-dp)))
70      vRightMotor = -0.3*(1/0.05)*gxp*((Gxp*(-3*ap*xp*xp-2*bp*xp-5*cp+0.18/2*dr)
+Hxp*((0.18*(-3*ap*xp*xp-2*bp*xp-5*cp))/(2*dr)-1))*math.cos(gamma) +(Gxp*(1-0.18*(-
3*ap*xp*xp-2*bp*xp-5*cp)/2*dr)+Hxp*(-3*ap*xp*xp-2*bp*xp-5*cp+0.18/2*dr))*math.sin
(gamma))
71   vLeftMotor = -0.3*(1/0.05)*gxp*((Gxp*(-3*ap*xp*xp-2*bp*xp-5*cp-0.18/2*dr) -
Hxp*((0.18*(-3*ap*xp*xp-2*bp*xp-5*cp))/(2*dr)+1))*math.cos(gamma)+(Gxp*(1+0.18*(-
3*ap*xp*xp-2*bp*xp-5*cp)/2*dr)+Hxp*(-3*ap*xp*xp-2*bp*xp-5*cp-0.18/2*dr))*math.sin
(gamma))
72  else:
73      gxp = 1/(math.sqrt(abs((-3*ap*xp*xp-2*bp*xp-0.5*cp)*(-3*ap*xp*xp-2*bp*xp-
0.5*cp)+1)))
74      Gxp = -(xf-xi)/(abs(xf-xi))*(yp-ap*xp*xp*xp-bp*xp*xp-0.5*cp*xp-dp)
75      Hxp = -(xf-xi)/(abs(xf-xi))*math.sqrt(abs(1-(yp-ap*xp*xp*xp-bp*xp*xp-0.5*cp*xp-
```

dp)*(yp-ap*xp*xp*xp-bp*xp*xp-0.5*cp*xp-dp)))
76 vRightMotor = -0.3*(1/0.05)*gxp*((Gxp*(-3*ap*xp*xp-2*bp*xp-0.5*cp+0.18/2*dr)
+Hxp*((0.18*(-3*ap*xp*xp-2*bp*xp-0.5*cp))/(2*dr)-1))*math.cos(gamma)+(Gxp*(1-0.18*(-
3*ap*xp*xp-2*bp*xp-0.5*cp)/2*dr)+xp*(-3*ap*xp*xp-2*bp*xp-0.5*cp+0.18/2*dr))*math.sin
(gamma))
77 vLeftMotor = -0.3*(1/0.05)*gxp*((Gxp*(-3*ap*xp*xp-2*bp*xp-0.5*cp-0.18/2*dr)-
Hxp*((0.18*(-3*ap*xp*xp-2*bp*xp-0.5*cp))/(2*dr)+1))*math.cos(gamma)+(Gxp*(1+0.18*(-
3*ap*xp*xp-2*bp*xp-0.5*cp)/2*dr)+Hxp*(-3*ap*xp*xp-2*bp*xp-5*cp-0.18/2*dr))*math.sin
(gamma))
78 # Comandos motores
79 sim.simxPauseCommunication(clientID, True)
80 sim.simxSetJointTargetVelocity(clientID, robotRightMotor, vRightMotor, sim.si
mx$_o pmode_o neshot$)
81 $sim.simxSetJointTargetVelocity(clientID, robotLeftMotor, vLeftMotor, sim.simx$
$_o pmode_o neshot$)
82 $sim.simxPauseCommunication(clientID, False)$
83 $sim.simxPauseSimulation(clientID, sim.simx_o pmode_o neshot_w ait) sim.simxFinish$
$(clientID)$
84$else$:
85 $print('Falhadeconexo')$
86$print('Programafinalizado')$

Escrevendo a função com a mesma notação do código em Python acima, temos que

$$Px_p = -x_p * x_p * x_p + x_p * x_p + x_p - 1.$$

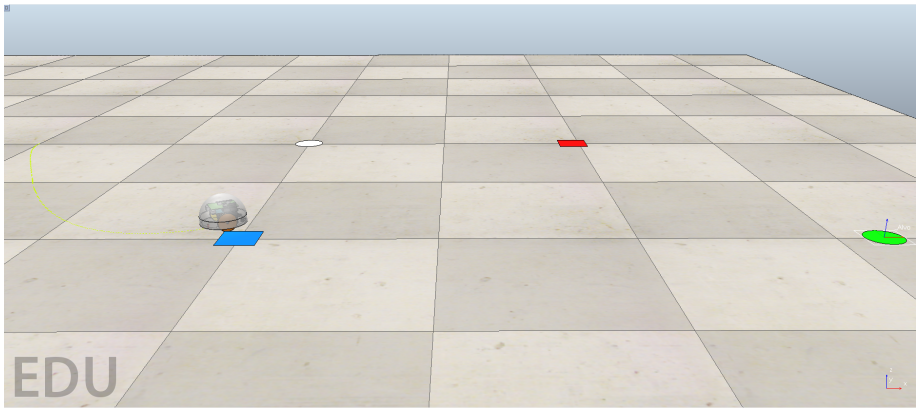
Logo:

- $a_p = -1$ será substituído na linha 38 do código em Python;
- $b_p = 1$ será substituído na linha 39 do código em Python;
- $c_p = 1$ será substituído na linha 40 do código em Python;
- $d_p = -1$ será substituído na linha 41 do código em Python;
- $x_i = -1$ será substituído na linha 42 do código em Python;
- $x_f = 1$ será substituído na linha 43 do código em Python;
- $y_i = 0$ será substituído na linha 44 do código em Python;
- $y_f = 0$ será substituído na linha 45 do código em Python;

- $d_r = 18$ que não faz parte da função, mas é o valor de correção da trajetória, será substituído na linha 50 do código em Python. Este valor pode ser modificado para melhorar a trajetória.

Temos que o Lumibot sairá de sua posição inicial e passará primeiramente pelo quadrado azul, como ilustra a Figura 3.3.

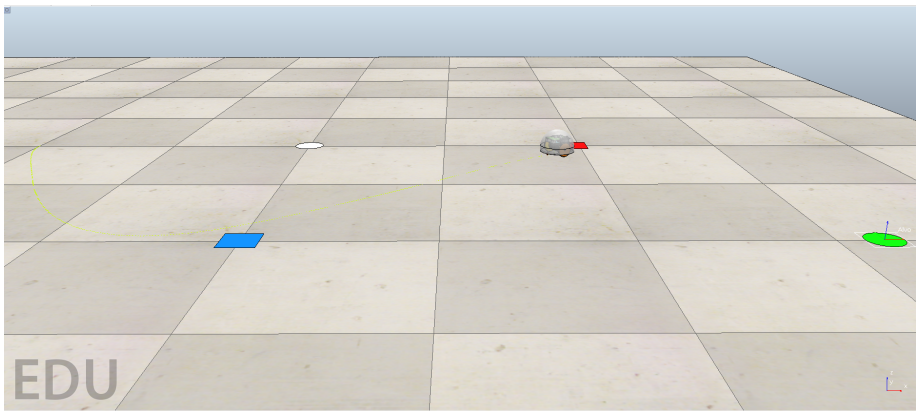
Figura 3.3: Lumibot passando pelo primeiro quadrado



Fonte: próprio autor.

Continuando o Lumibot passará em seguida pelo segundo quadrado, como ilustra a Figura 3.4.

Figura 3.4: Lumibot passando pelo segundo quadrado



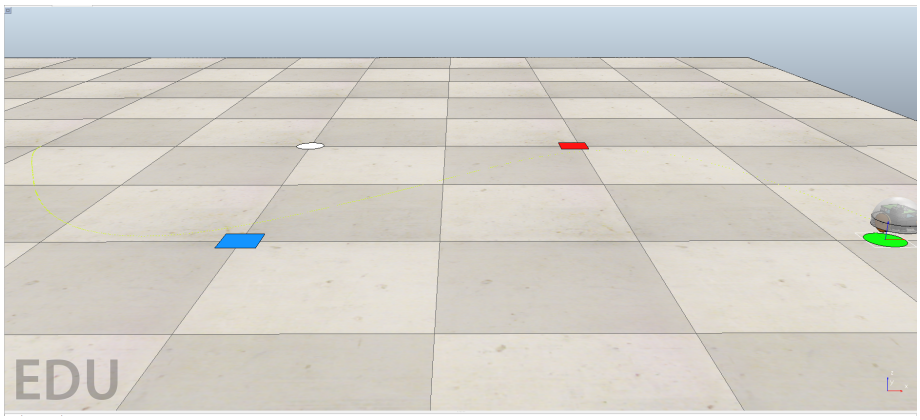
Fonte: próprio autor.

Por fim, o robô Lumibot chegará até o seu alvo que é o disco verde, onde o mesmo irá parar como ilustra a Figura 3.5

Caso tivessemos apenas a função cúbica $P(x) = -x^3 + x^2 + x - 1$, poderiamos identificar alguns pontos com os conhecimentos do método de Cardano-Tartaglia estudado no capítulo 2.

Pelo método de Cardano-Tartaglia, podemos encontrar suas raízes. Resolvendo a equação $-x^3 + x^2 + x - 1 = 0$.

Figura 3.5: Lumibot atingindo o seu alvo



Fonte: próprio autor.

Inicialmente note que $p = -\frac{4}{3}$ e $q = \frac{16}{27}$, como $\Delta = \frac{q^2}{4} + \frac{p^3}{27} = \frac{(\frac{16}{27})^2}{4} + \frac{(\frac{-4}{3})^3}{27} = 0$, pelo método de Cardano-Tartaglia a equação possui três raízes reais, onde pelo menos duas são iguais:

$$\begin{aligned}
x_1 &= \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} - \frac{A}{3} \\
&= \sqrt[3]{-\frac{(\frac{16}{27})}{2} + \sqrt{\frac{(\frac{16}{27})^2}{4} + \frac{(-\frac{4}{3})^3}{27}}} + \sqrt[3]{-\frac{(\frac{16}{27})}{2} + \sqrt{\frac{(-\frac{16}{27})^2}{4} + \frac{(-\frac{4}{3})^3}{27}}} - \frac{1}{3} \\
&= \sqrt[3]{-\frac{8}{27} + \sqrt{0}} + \sqrt[3]{-\frac{8}{27} - \sqrt{0}} + \frac{1}{3} \\
&= \sqrt[3]{(-\frac{2}{3})^3} + \sqrt[3]{(-\frac{2}{3})^3} + \frac{1}{3} \\
&= -\frac{2}{3} - \frac{2}{3} + \frac{1}{3} \\
&= -1.
\end{aligned} \tag{3.3}$$

Para obter as outras raízes, basta dividir $-x^3 + x^2 + x - 1 = 0$ por $(x + 1)$, onde obtemos $-x^3 + x^2 + x - 1 = (x + 1)(-x^2 + 2x - 1)$ e usar qualquer método para resolver equação do 2º grau, onde temos como solução:

$$\begin{aligned}
x_1 &= 1 \quad \text{(Raiz real)} \\
x_2 &= x_3 = -1 \quad \text{(Raízes reais de multiplicidade dois)}
\end{aligned}$$

Das raízes da função cúbica acima, obtemos os pontos $(-1, 0)$ e $(1, 0)$, que correspodem respectivamentes as posições do Lumibot e do quadrado de cor vermelha, conforme Figura 3.2. Sabemos que a interseção do gráfico com o eixo y estar relacionada ao valor de "d" bastar calcular $P(0)$, que obteremos o ponto $(0, d)$, donde

$P(0) = -(0)^3 + (0)^2 + (0) - 1 = -1$, obtendo o ponto $(0, -1)$.

Usando pontos no plano ou uma função cúbica, conseguimos criar um modelo preciso para o caminho do Lumibot. Essa abordagem flexível combina informações específicas dos pontos com uma fórmula matemática, permitindo não apenas definir o caminho com base em dados concretos, mas também fornecer uma representação matemática sólida por meio da função cúbica.

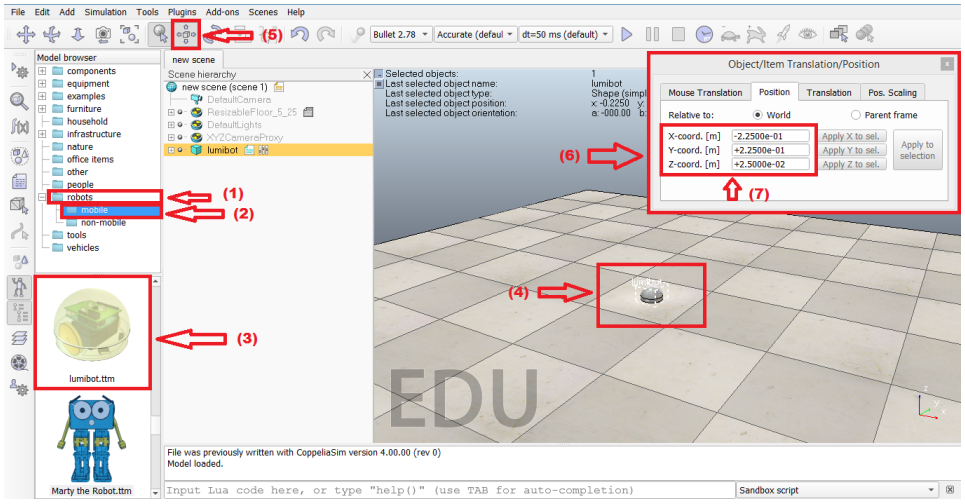
3.2.2 Cenário 2

Criamos um cenário mais realista, onde o Lumibot do CoppeliaSim seguirá uma trajetória modelada por uma função cúbica.

Com o CoppeliaSim aberto, escolheremos o robô que será utilizado em nossa apresentação. Na barra lateral esquerda, encontre a "Biblioteca de Objetos" ou "Library Browser", dentro da biblioteca procure pela categoria de robôs ou pesquise diretamente por "Lumibot". Clique e arraste o modelo Lumibot para a área de trabalho.

Uma vez adicionado à cena, podemos ajustar a posição inicial arrastando o Lumibot na janela de cena ou manipulando as coordenadas de posição na lista de propriedades. Localize as configurações de posição e altere as coordenadas x, y e z, altere esses valores para mover o Lumibot para a nova posição desejada. Alternativamente, podemos simplesmente arrastar o Lumibot para a nova posição diretamente na janela de cena usando as ferramentas de manipulação de objetos. Abaixo mostraremos o passo a passo utilizado, conforme ilustra a Figura 3.6.

Figura 3.6: Passos indicados por setas



Fonte: próprio autor.

Ou seja, temos os seguintes passos:

(1) Clique em robots;

(2) clique na pasta mobile;

(3) Escolha o Lumibot e arraste para área de trabalho do CoppeliaSim;

(4) Já estamos com Lumibot na área de trabalho como desejamos.

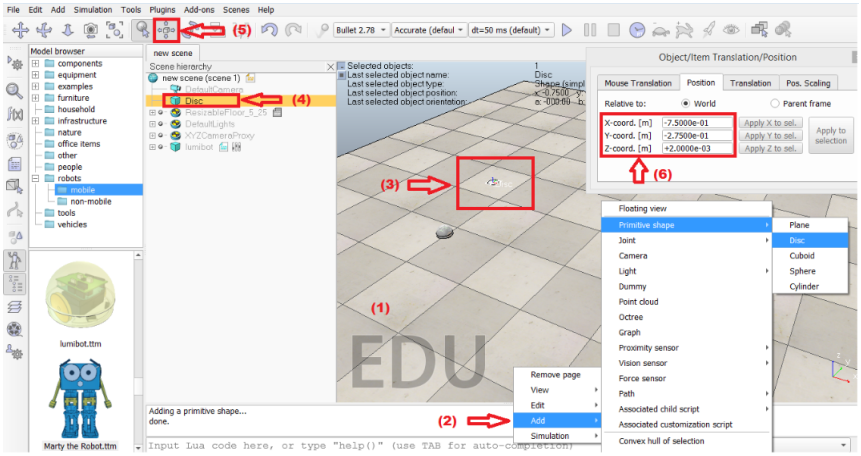
(5) Clicando sobre este ícone abriremos a caixa para alteração do objeto no plano;

(6) Através desta caixa podemos modicar a posição do objeto;

(7) Para o nosso cenário iremos utilizar a posição inicial do Lumibot $(-1, 0)$.

O nosso proximo passo consiste em adicionar o nosso alvo, ou seja, onde o nosso robô irá parar ao atingi-lo, e em seguida iremos ajustar a sua posição que será um ponto pertencente a função cúbica que iremos utilizar, com ilustra a Figura 3.7.

Figura 3.7: Passo para colocação do alvo



Fonte: próprio autor.

Isto é, temos os seguintes passos:

(1) Com o botão direito click na área de trabalho do CoppeliaSim;

(2) Click em **Add**, depois em **Primitive chape**, e por fim em **Disc**;

(3) Já estamos com o Disc na área de trabalho como desejamos;

(4) Com dois clicks sobre o Disc iremos renomea-lo para **Alvo**;

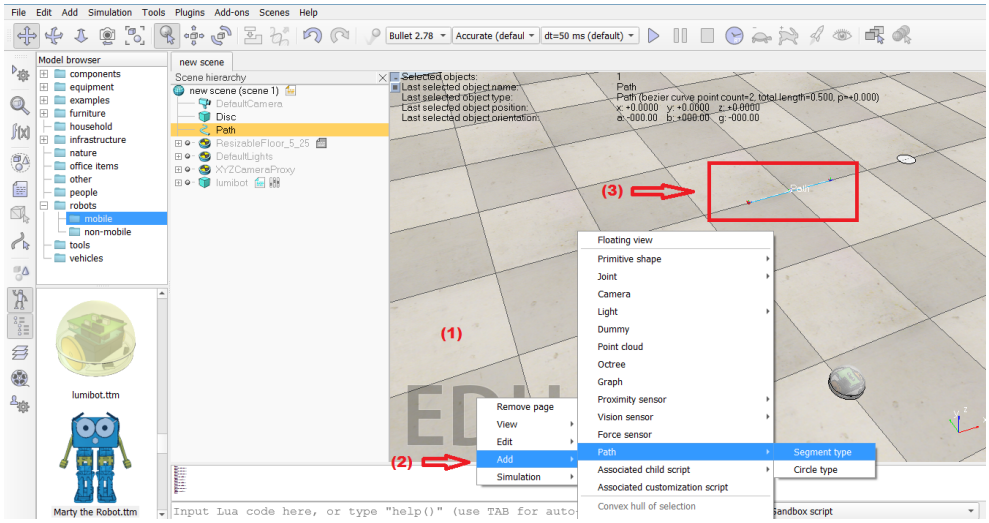
(5) Clicando sobre este ícone abriremos a caixa para alteração do objeto no plano;

(6) Através desta caixa podemos modicar a posição do objeto, para o nosso cenário iremos utilizar a posição $(2, 1)$.

Para o nosso cenário iremos criar um caminho (path) e em seguida configurá-lo para seguir uma trajetória cúbica. Para adicionar um caminho clique em menu "Scene Object"(Objeto de Cena) e escolha um objeto que representará o caminho. Você pode

optar por um objeto de trajetória ou uma spline, como ilustra a Figura 3.8.

Figura 3.8: Passos para criação da trajetória



Fonte: próprio autor.

Utilizamos os seguintes passos:

(1) Com o botão direito click na área de trabalho do CoppeliaSim;

(2) Click em **Add**, depois em **Path**, e por fim em **Segment type**;

(3) Já estamos com o Path na área de trabalho como desejamos.

Com o path na área de trabalho do CoppeliaSim podemos configurar algumas propriedades em "Path Shaping"como por exemplo, o tipo de segmento vertical, fator de escala, e componentes ambientais/difusas.

Basta abrir o diálogo de configurações de caminho (Path Shaping Dialog), procurar por uma opção ou menu relacionado a “Path Shaping” ou “Trajectory Configuration”, no diálogo de configuração, e ativar ou habilitar o “Path Shaping”, também podemos definir o tipo de segmento como “Vertical Segment”, isso define como os pontos do caminho se comportam verticalmente.

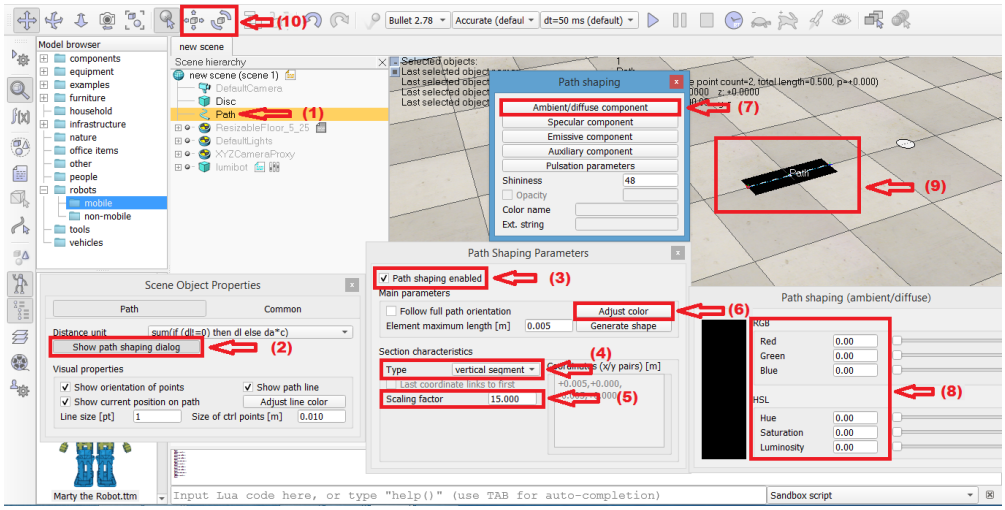
Dentro do diálogo de configuração, encontre a opção “Scaling Factor” para ajustar o tamanho ou escala do caminho, experimente diferentes valores para ver como afetam o caminho, já para alterar a aparência visual do caminho utilize a opção configurar componentes ambientais/difusas, como ilustra a Figura 3.9.

Para o nosso cenários utilizamos os seguintes passos:

(1) Clique em Path;

(2) Abrirar uma janela chamada "Scene Object Properties"basta clicar em Show path shaping dialog;

Figura 3.9: Configurando a trajetória



Fonte: próprio autor.

(3) Marque a opção Path shaping enable para alterar alguns parâmetros;

(4) Em Type selecione a opção vertical segment;

(5) Na opção Scaling factor altere para 15.000;

(6) Clique em ajust color;

(7) Clique em Ambient/diffuse component;

(8) Tanto em "RGB"quanto em "HSL"zere todos os valores;

(9) O Path já se encontra na área de trabalho com todas as alterações feitas;

(10) Através destas caixas podemos modificar a posição do Path.

Para criarmos um cenário mais realista iremos adicionar árvore, terreno e montanha. Basta no menu “Scene Object” (Objeto de Cena) procurarmos por Tree (árvore), Terrain (terreno) e Terrain bump (irregularidades do terreno).

Ao arrastarmos os objetos para a área de trabalho do CoppeliaSim podemos configurar propriedades tais como densidade, altura, e orientação para as árvores e para o terreno podemos configurar o tamanho, texturas que simulem diferentes tipos de terreno, de maneira a criar um efeito realista, como ilustra a Figura 3.10.

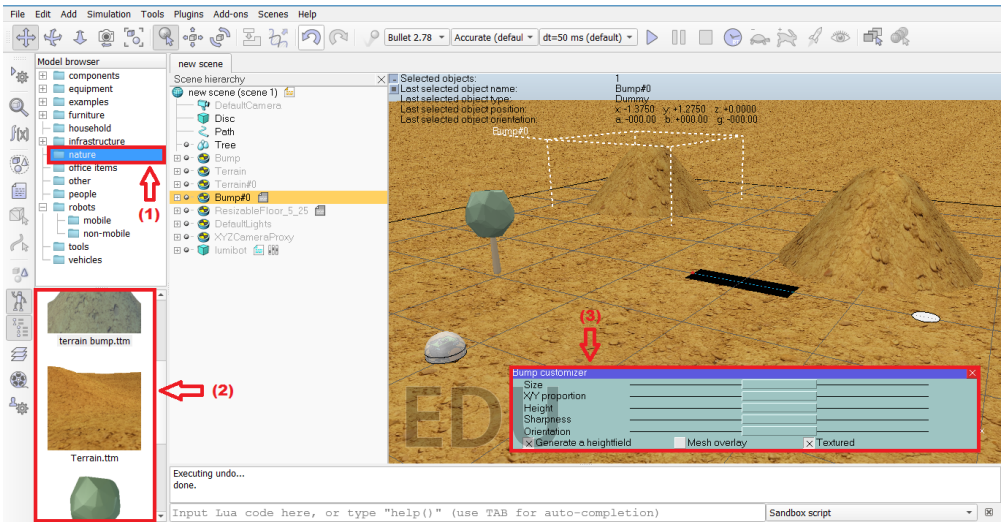
Utilizamos os seguintes passos:

(1) Clique em nature;

(2) Arraste para a área de trabalho do CoppeliaSim, os objetos "terrain bump", "terrain"e "tree", de modo a criar um cenário realista;

(3) Ao adicionar o objeto para a área de trabalho aparecerá a opção "bump custo-

Figura 3.10: Adicionando outros objetos ao cenário

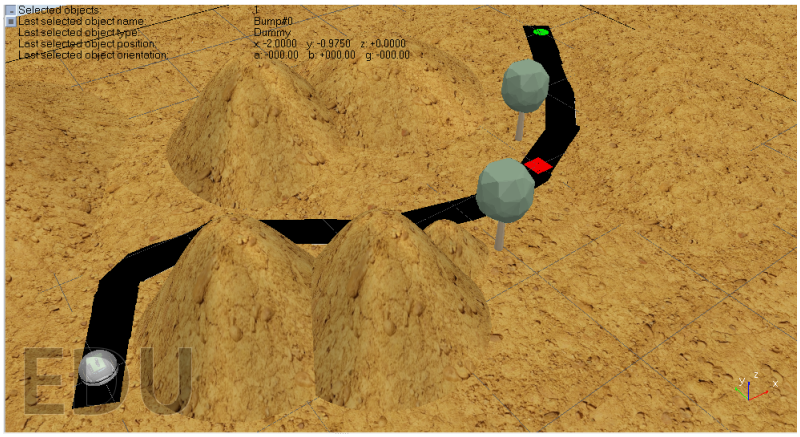


Fonte: próprio autor.

mizer", onde podemos configurar o objeto.

Já temos todos os comandos necessários para montarmos o nosso cenário, como ilustra a Figura 3.11.

Figura 3.11: Finalização de cenário



Fonte: próprio autor.

Temos que o Lumibot está localizado na posição $(-1, 0)$ e o alvo em que o Lumibot deve atingir está localizado na posição $(2, 1)$, todas estas posições são consideradas em relação ao centro de massa dos objetos. Queremos que o Lumibot também passe pelos pontos $(0, 0)$ e $(1, 0)$. É conhecido que podemos modelar uma função utilizando a interpolação de Lagrange, a qual passará exatamente por todos esses pontos.

Pelo método da interpolação de Lagrange, dados quatro pontos no plano, podemos modelar uma função cúbica. Conforme a fórmula obtida em 1.9, donde:

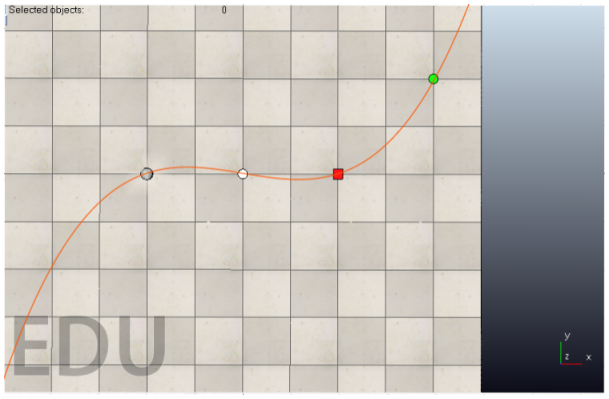
$$P(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}y_0 + \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}y_1 + \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}y_2 + \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}y_3.$$

Substituindo os pontos (x_0, y_0), (x_1, y_1), (x_2, y_2) e (x_3, y_3) da fórmula pelos pontos $(-1, 0)$, $(0, 0)$, $(1, 0)$ e $(2, 1)$, respectivamente. Obtemos:

$$P(x) = \frac{(x-0)(x-1)(x-2)}{(-1-0)(-1-1)(-1-2)}.0 + \frac{(x+1)(x-1)(x-2)}{(0+1)(0-1)(0-2)}.0 + \frac{(x+1)(x-0)(x-2)}{(1+1)(1-0)(1-2)}.0 + \frac{(x+1)(x-0)(x-1)}{(2+1)(2-0)(2-1)}.1$$

Resolvendo, temos a seguinte função cúbica $P(x) = \frac{1}{6}(x+1).x.(x-1)$, ou seja, $P(x) = \frac{1}{6}x^3 - \frac{1}{6}x$. Mostraremos o gráfico de $P(x)$, no ambiente de simulação do CoppeliaSim, conforme a Figura 3.12.

Figura 3.12: Gráfico da função que será seguida pelo Lumibot



Fonte: próprio autor.

Escrevendo a função com a mesma notação do código em Python acima, temos que

$$Px_p = \frac{1}{6} * x_p * x_p * x_p + x_p * x_p - \frac{1}{6} * x_p.$$

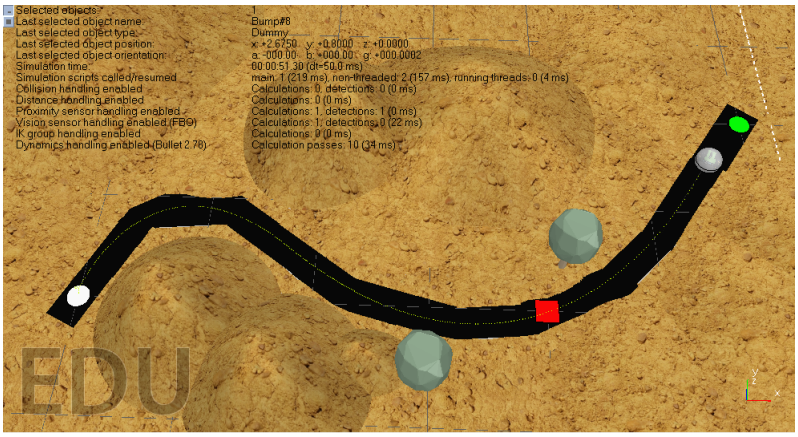
Logo:

- $a_p = \frac{1}{6}$ será substituído na linha 38 do código em Python;
- $b_p = 0$ será substituído na linha 39 do código em Python;
- $c_p = -\frac{1}{6}$ será substituído na linha 40 do código em Python;
- $d_p = 0$ será substituído na linha 41 do código em Python;

- $x_i = -1$ será substituído na linha 42 do código em Python;
- $x_f = 2$ será substituído na linha 43 do código em Python;
- $y_i = 0$ será substituído na linha 44 do código em Python;
- $y_f = 1$ será substituído na linha 45 do código em Python;
- $d_r = 27$ que não faz parte da função, mas é o valor de correção da trajetória, será substituído na linha 50 do código em Python. Este valor pode ser modificado para melhorar a trajetória.

Após digitar o código acima no editor de texto PyCharm e compilar, temos que o Lumibot percorrerá a trajetória descrita pela função cúbica $P(x) = \frac{1}{6}x^3 - \frac{1}{6}x$, conforme a Figura 3.13.

Figura 3.13: Lumibot percorrendo a função cúbica



Fonte: próprio autor.

Como virmos acima, o Lumibot executou com precisão a tarefa que nos propusemos, que consistiu em percorrer uma função cúbica dados quatro de seus pontos.

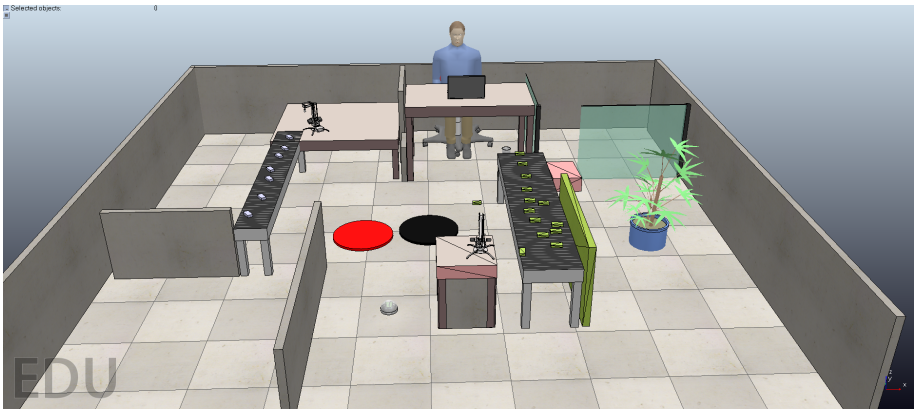
No próximo cenário, buscamos uma abordagem mais realista, onde os conhecimentos adquiridos sobre funções polinomiais do terceiro grau serão aplicados para resolver problemas do mundo real. Este ambiente mais desafiador proporcionará uma oportunidade valiosa para aplicar de maneira prática as habilidades adquiridas até o momento.

3.2.3 Cenário 3

Neste cenário consideraremos uma empresa do ramo de empacotamento de medicamentos, composta por duas esteiras automáticas uma que leva caixas para embalar os medicamentos, situada à esquerda com as caixas vazias na cor cinza; a outra situada à direira devolve os medicamentos já embalados, neste caso as caixas estão na cor verde. Todo o processo da empresa é feito de modo automatizado com apenas um funcionário

controlando todo o sistema, como ilustra a Figura 3.14.

Figura 3.14: Visão geral da empresa



Fonte: próprio autor.

Frequentemente algumas caixas caem das esteiras, por isso a empresa ainda conta com um robô móvel, o Lumibot, que tem a função de recolher as caixas que caem e levar para um local seguro dentro da empresa, como ilustra a Figura 3.15, o robô está à esquerda da mesa que contém o braço robótico que coloca as caixas com medicamentos e a caixa com medicamento na cor verde, sobre o chão, está ao lado da esteira bem próximo ao funcionário.

Figura 3.15: Lumibot e o alvo

Fonte: próprio autor.

Entre o Lumibot e a caixa com medicamentos que está no chão, temos uma abertura para o subsolo e a tampa na cor vermelha, desta passagem. Queremos que o robô saia da sua posição inicial, desvie dos obstáculos e retire a caixa em segurança levando para um local próximo ao rapaz que o opera todo sistema, deixando-a a uma distância de 0,50 metros de distância dele.

Esta situação apresentada neste cenário, ilustra bem a necessidade de criatividade matemática para resolver problemas do mundo real. O objetivo é estimular que os estudantes

utilizem o pensamento matemático para apresentarem uma solução para este problema.

Levando que consideração que todo o cenário da empresa está sobre um plano cartesinano OXY, onde a origem do plano OXY é exatamente o centro do círculo que forma o buraco, sendo que o diâmetro deste buraco mede 0,50 metros, como consequência sua tampa na cor vermelha também possue as mesmas dimensões. Nosso objetivo é fazer com que o Lumib saia da posição $L_a = (-0.29, -1.2)$, passe pelo ponto $L_b = (0, -0.8)$ para desviar do obstáculo e em seguida leve a caixa de medicamento localizada na posição $L_c = (0.44, 0.5)$ com segurança até o alvo, localizado na posição $L_d = (0.51, 1)$.

Para resolver este problema utilizaremos o gráfico da função polinomial do terceiro grau como trajetória para o Lumibot. Como já conhecemos as coordenadas dos pontos L_a, L_b, L_c e L_d, basta aplicar a fórmula de Lagrange obtida em 1.9, donde:

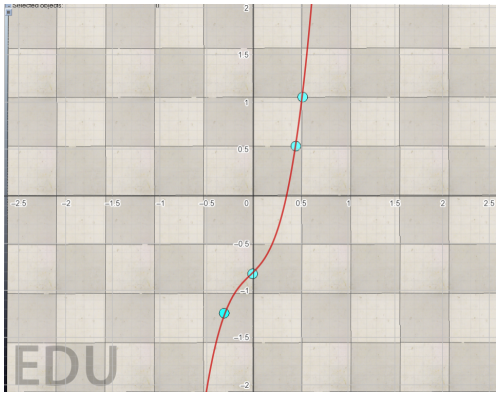
$$P(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}y_0 + \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}y_1 + \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}y_2 + \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}y_3$$

Substituindo os pontos (x_0, y_0), (x_1, y_1), (x_2, y_2) e (x_3, y_3) da fórmula pelos pontos $(-0.29, -1.2)$, $(0, -0.8)$, $(0.44, 0.5)$ e $(0.51, 1)$, respectivamente. Obtemos:

$$P(x) = \frac{(x-0)(x-0.44)(x-0.51)}{(-0.29-0)(-0.29-0.44)(-0.29-0.51)}.(-1.2) + \frac{(x+0.29)(x-0.44)(x-0.51)}{(0+0.29)(0-0.44)(0-0.51)}.(-0.8) + \frac{(x+0.29)(x-0)(x-0.51)}{(0.44+0.29)(0.44-0)(0.44-0.51)}.(0.5) + \frac{(x+0.29)(x-0)(x-0.44)}{(0.51+0.29)(0.51-0)(0.51-0.44)}.(1)$$

Resolvendo, temos a seguinte função cúbica $P(x) = 8x^3 + x^2 + x - 0.8$, onde o seu gráfico acaba sendo uma execelente escolha para a trajetória descrita pelo robô, como ilustra a Figura (3.16).

Figura 3.16: gráfico da função polinomial



Fonte: próprio autor.

Reescrevendo a função obtida de acordo com as notações do código em Python, temos:

$$Px_p = 8 * x_p * x_p * x_p + x_p * x_p + x_p * x_p + x_p - 0.8.$$

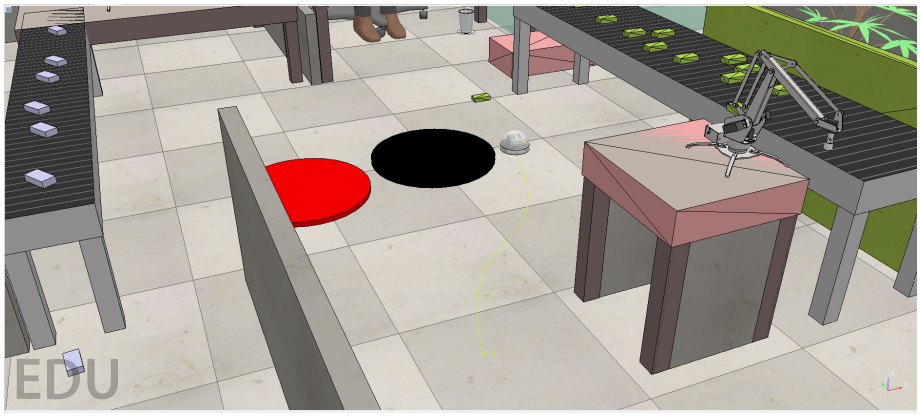
Logo:

- $a_p = 8$ será substituído na linha 38 do código em Python;
- $b_p = 1$ será substituído na linha 39 do código em Python;
- $c_p = 1$ será substituído na linha 40 do código em Python;
- $d_p = -0.8$ será substituído na linha 41 do código em Python;
- $x_i = -0.29$ será substituído na linha 42 do código em Python;
- $x_f = 0.51$ será substituído na linha 43 do código em Python;
- $y_i = -1.2$ será substituído na linha 44 do código em Python;
- $y_f = 1$ será substituído na linha 45 do código em Python;
- $d_r = 35$ que não faz parte da função, mas é o valor de correção da trajetória, será substituído na linha 50 do código em Python. Este valor pode ser modificado para melhorar a trajetória.

$$Px_p = 8x_p^3 + x_p^2 + x_p - 0.8.$$

Isto é, $a_p = 8$, $b_p = c_p = 1$, $d_p = -0.8$, assim levando estes dados para o código, obtemos que o robô percorre o trajeto escolhido desviando do buraco e da tampa que cobre o buraco, como ilustra a Figura 3.17.

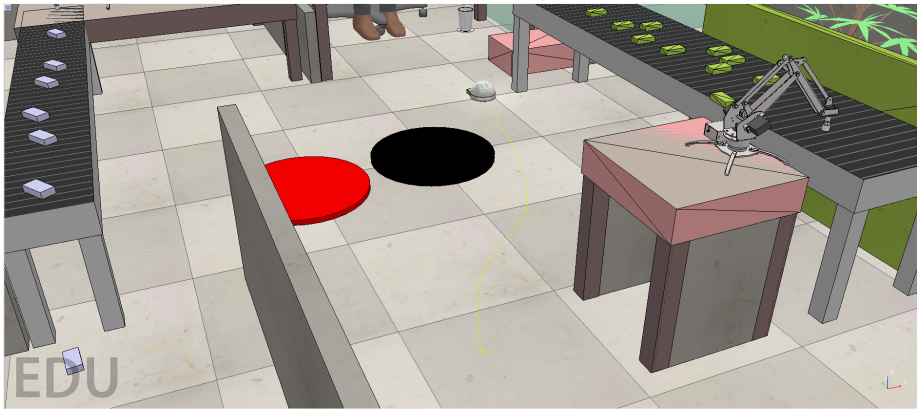
Figura 3.17: Lumibot desviando dos obstáculos



Fonte: próprio autor.

Continuando o trajeto, o Lumibot consegue atingir o alvo, como ilustra a Figura 3.18.

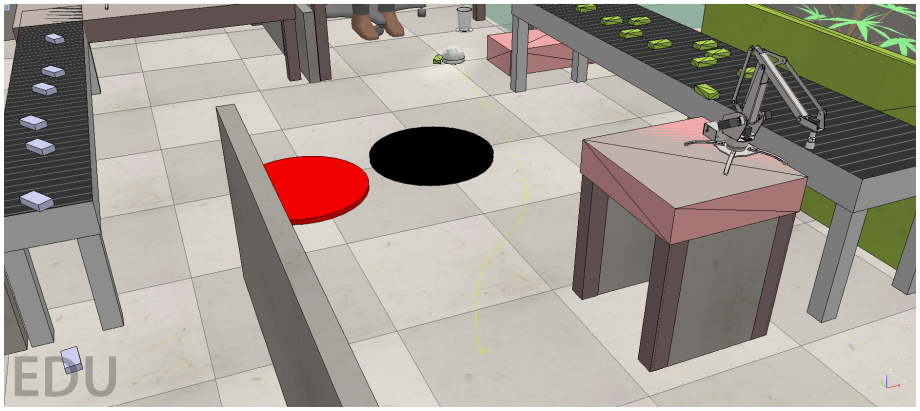
Figura 3.18: Lumibot atigindo o alvo por meio da função cúbica



Fonte: próprio autor.

Por fim, o trajeto é concluído com o robô deixando a caixa de medicamentos em segurança próximo ao funcionário que opera a empresa, como ilustra a Figura 3.19

Figura 3.19: Lumibot deixando a caixa em segurança



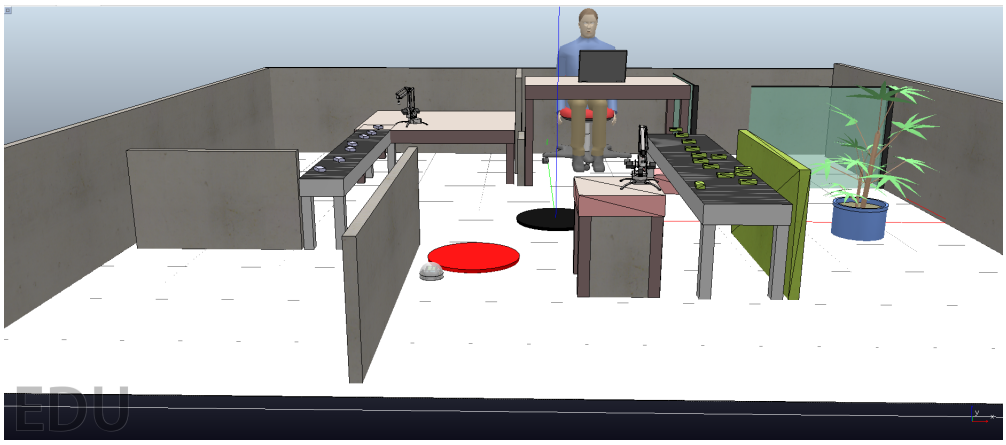
Fonte: próprio autor.

Podemos perceber que a tarefa foi execultada conforme planejado.

3.2.4 Cenário 4

Este cenário será uma reformulação do Cenário anterior, onde temos uma empresa do ramo de empacotamento de medicamentos, satisfazendo as mesmas condições do Cenário 3. Porém, neste Cenário apresenta um pouco mais de dificuldade, como o piso da empresa é composto por divisórias em formas de quadrados, que auxiliar na localização dos objetos sobre o piso, como ilustra a Figura 3.20. Além disso o robô está posicionado bem atrás da tampa em vermelho.

Figura 3.20: A empresa com o piso na cor braca



Fonte: próprio autor.

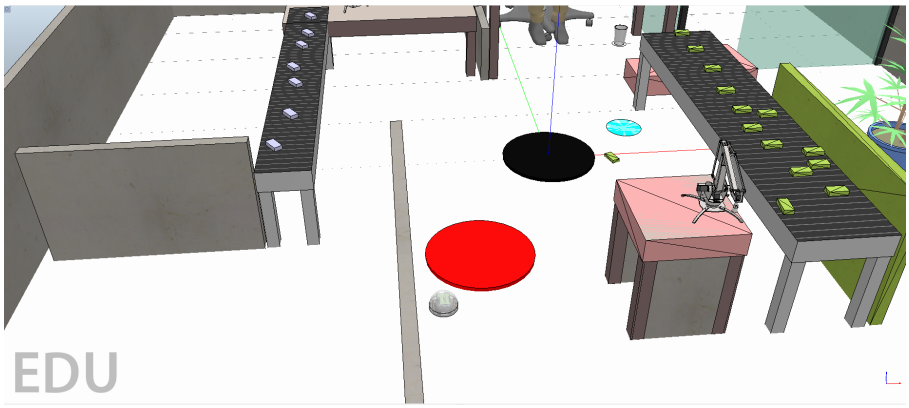
Frequentemente as caixas de medicamentos caem das esteiras, como a caixa verde ao lado do buraco indicado na cor preta, como ilustra a Figura 3.21. Para auxiliar o funcionário que opera todo o sistema, o Lumibot fica a postos em sua posição $P_i = (-0.67, -1.17)$ sobre o piso, para a retirada das caixas levando até um local seguro, que neste caso, é o disco que cor azul, como ilustra a Figura 3.21.

Para fazer com que o Lumibot saia de sua posição inicial, pegue o alvo (caixa de medicamento), e leve até o disco azul, precisamos de uma função para que o robô siga o seu gráfico como trajetória. Note que seguindo apenas uma reta, o robô não consegue atingir a posição final desejada. Seria necessário ao menos dois seguimentos de reta, e não dispomos destes trabalhos na literatura. Desta forma, teremos que encontrar outra função para o Lumibot seguir o seu gráfico.

Considerando o piso com um sistema de eixos ortogonais OXY, temos que o centro do buraco na cor preta é exatamente a origem do plano OXY, temos ainda, um reta na cor vermelha que representa o eixo OX e uma reta na cor verde que representa o eixo OY, como ilustra a Figura 3.21. Como o diâmetro do buraco mede 0,50 metros, obtemos uma boa aproximação para o nosso alvo, que neste caso é a caixa na cor verde, que ocupa aproximadamente a posição $P_a = (0.34, 0)$. Sabemos que o centro do disco na cor azul ocupa a posição $P_f = (0.5, 0.29)$, desta forma obtemos a posição de três pontos sobre o plano OXY, e por Interpolação de Lagrange o máximo que podemos obter é uma parábola. Como a parábola não resolve o nosso problema, não poderemos aplicar Interpolação de Lagrange.

Olhando para a trajetória que será seguida pela robô, percebemos que seguindo o gráfico de uma função polinomial do terceiro grau, o robô consegue sair de sua posição inicial, tocar na caixa de medicamentos e arrastar a até o disco azul, em segurança. Através de três pontos o Teorema (2.3.1) - item (ii), garante que a função polinomial do

Figura 3.21: Lumibot atrás da tampa em vermelho e caixa verde próximo ao buraco



Fonte: próprio autor.

terceiro grau cuja o gráfico serve como trajetória para o Lumibot é dada por:

$$f(x) = -0.62x^3 + 0.63x^2 + 1.58x - 0.58,$$

Reescrevendo a função obtida de acordo com as notações do código em Python, temos:

$$fx_p = -0.62 * x_p * x_p * x_p + 0.63x_p * x_p + 1.58x_p * x_p + x_p - 0.58.$$

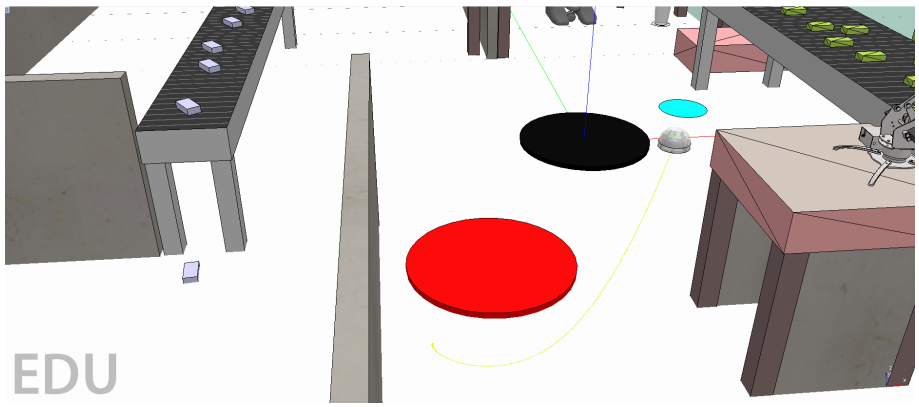
Logo:

- $a_p = -0.62$ será substituído na linha 38 do código em Python;
- $b_p = 0.63$ será substituído na linha 39 do código em Python;
- $c_p = 1.58$ será substituído na linha 40 do código em Python;
- $d_p = -0.58$ será substituído na linha 41 do código em Python;
- $x_i = -0.67$ será substituído na linha 42 do código em Python;
- $x_f = 0.51$ será substituído na linha 43 do código em Python;
- $y_i = -1.17$ será substituído na linha 44 do código em Python;
- $y_f = 0.29$ será substituído na linha 45 do código em Python;
- $d_r = 6$ que não faz parte da função, mas é o valor de correção da trajetória, será substituído na linha 50 do código em Python. Este valor pode ser modificado para melhorar a trajetória.

Assim levando estes dados para o código, obtemos que o robô percorre o trajeto escolhido desviando do buraco e da tampa que cobre o buraco e do buraco, como ilustra a Figura

3.22.

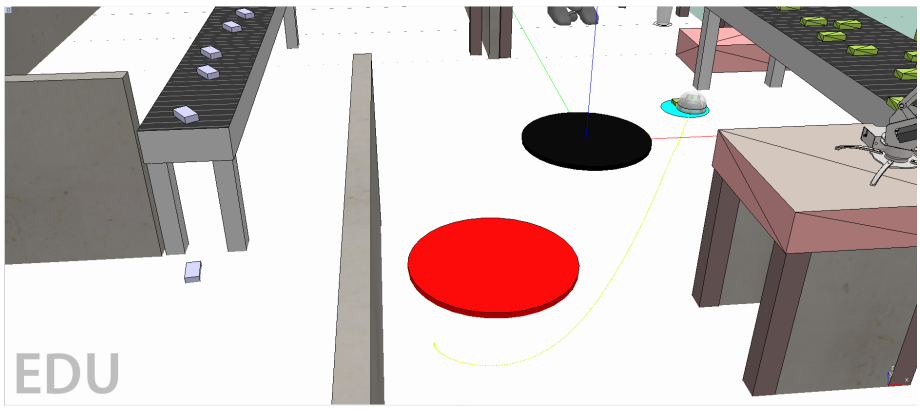
Figura 3.22: Lumibot pegando a caixa de medicamentos



Fonte: próprio autor.

Por fim, o trajeto é concluído com o robô deixando a caixa de medicamentos em segurança no local indicado, como ilustra a Figura 3.23

Figura 3.23: Lumibot deixando a caixa em segurança



Fonte: próprio autor.

Finalizando o trabalho.

Capítulo 4

Considerações Finais

Neste estudo, exploramos o potencial interdisciplinar do ambiente de simulação robótica, CoppeliaSim, ao utilizar o Lumibot como uma ferramenta pedagógica. Essa abordagem visa despertar a criatividade matemática dos alunos na educação básica. O ambiente CoppeliaSim, aliado ao código editável fornecido, capacita os professores a criar cenários dinâmicos e personalizados em sala de aula, estimulando o interesse dos alunos pela matemática.

Destacamos a versatilidade do Lumibot, capaz de lidar com situações de contexto mais complexo, proporcionando uma experiência educacional enriquecedora. A utilização dessa ferramenta não só desperta o interesse dos alunos pela matemática, mas também promove a criatividade necessária para enfrentar desafios do mundo real, como os apresentados neste trabalho.

A aplicação prática da fórmula de interpolação polinomial de Lagrange transcende o mero cálculo matemático. Ela capacita o Lumibot a navegar de forma precisa e eficiente por pontos específicos no plano, oferecendo uma solução concreta para a resolução de problemas do mundo real.

Em resumo, a integração da fórmula de Lagrange, o estudo do comportamento gráfico das funções cúbicas e a aplicação prática desses conceitos proporcionam uma abordagem interdisciplinar, unindo matemática, robótica e análise de dados. Este trabalho não apenas contribui para a compreensão teórica, mas também estabelece uma base sólida para a modelagem e previsão do movimento do Lumibot em suas atividades, ampliando as possibilidades de ensino e aprendizagem na interface entre matemática e robótica.

Referências Bibliográficas

[1] AGUIAR, Gerson Bruno Pinto de . A circunferência Escrita Como Soma e Produto de matrizes: uma nova abordagemdo ensino de matrizes ensino médio. Dissertação de mestrado. Orientador: Cleidinaldo Aguiar Souza. UFPI. 2021.

[2] AGUIAR, Gerson Bruno Pinto de; SOUZA, Cleidinaldo Aguiar. Representação da circunferência como o produto e soma de matrizes. Revista piauiense de matemática. 2022.

[3] AGUIAR, Gerson Bruno Pinto de; SOUZA, Cleidinaldo Aguiar. *Pragmatismo matemátic: matrizes* Edufpi. 2023.

[4] AHRENS, Daniela; SPÖTTL, Georg. Industrie 4.0 und Herausforderungen für die Qualifizierung von Fachkräften. In: Digitalisierung industrieller Arbeit. Nomos Verlagsgesellschaft mbH & Co. KG, 2018. p. 173-194.

[5] ALVES, Diego Koeningkam. Resolução da equação de terceiro grau no ensino médio. 2023. 61 f. Dissertação (Mestrado). Programa de Pós-Graduação em Matemática – ProfMat. Universidade Federal de Juiz de Fora. Minas Gerais, 2023.

[6] BONJOVANI, Evandro Luís. Construção de gráficos de funções polinomiais de grau 3. 2015

[7] BOYER, C. B. História da matemática. São Paulo: Edgard Blucher, 1996.

[8] CAMPOS, Flávio Rodrigues. A robótica para uso educacional. 1. ed. São Paulo: Editora SENAC-SP, 2019.

[9] CHIAVENATO, Idalberto. Introdução à Teoria Geral da Administração. 9ª edição. São Paulo: Manole, 2014.

[10] ELBESTAWI, Mo et al. SEPT learning factory for industry 4.0 education and applied research. Procedia , v. 23, p. 249-254, 2018.

[11] FERRAZ, Dalva de Oliveira. Robótica educacional para formação de professores do curso técnico em agropecuária. 2023. 140 f. Dissertação (Mestrado). Programa de Pós-Graduação em Educação Profissional e Tecnológica. Instituto Federal do Espírito Santo. Vitória, 2023.

[12] GABETTA JÚNIOR, Antônio Marcos. Aproximação de Funções por Interpolação: Método de Lagrange. 2015. 168 f. Dissertação (Mestrado). Programa de Pós-Graduação em Educação Profissional e Tecnológica. Universidade Estadual de Campinas. Campinas-SP, 2015.

[13] MACEDO, Cássio Lima. Funções Polinomiais do Primeiro e Segundo Grau como Lumibot-CoppeliaSim. 2022. 66 f. Dissertação (Mestrado). Programa de Pós-Graduação em Matemática – ProfMat. Universidade Federal do Piauí. Teresina, 2023.

[14] MACEDO, Cássio Lima; Souza, Cleidinaldo Aguiar. Funções Polinomiais do Segundo Grau com o Lumibot-CoppeliaSim. Revista do professor de matemática. 2022.

[15] MARTINS, Daniel Felipe Neves; GOMES, Raphael Martins. O USO DO POLINÔMIO DERIVADO COMO ALTERNATIVA PARA RESOLVER PROBLEMAS INTERESSANTES RELATIVOS À FUNÇÃO POLINOMAL DO 3º GRAU COM AUXÍLIO DE TECNOLOGIA DIGITAL NO ENSINO MÉDIO: ATIVIDADES QUE GERAM MOVIMENTAÇÃO EM AULAS DE MATEMÁTICA. PESQUISAS MULTIDISCIPLINARES EM CIÊNCIAS EXATAS, VOLUME 2., p. 254, 2022.

[16] Nascimento, Luís B. P.; Pereira, Diego S.; Santos, Vitor G.; Fernandes, Daniel H. S.; Alsina, Pablo J. Introdução ao V-REP: Uma Plataforma Virtual para Simulação de Robôs. Sociedade Brasileira de Computação (SBC) - Porto Alegre. 2019.

[17] OLIVEIRA, Marina França. A circunferência de centro na origem como produto de matrizes. Dissertação de mestrado. Orientador: Cleidinaldo Aguiar Souza. UFPI. 2019.

[18] OLIVEIRA, Marina França; SOUZA, Cleidinaldo Aguiar. A circunferência de centro na origem como produto de matrizes. Revista do professor de matemática. 2020.

[19] OLIVEIRA, Wagner Vieira. Métodos de resolução de equações cúbicas e quadráticas no percurso histórico / Wagner Vieira Oliveira. – Ponta Grossa - PR: Atena, 2023.

[20] O TOYOTISMO NAS ESCOLAS: O PROFESSOR MULTIFUNCIONAL, Campina Grande, Vol. 1 Ed. 4, ISSN 2316-1086, Realize editora, 2015.

[21] SILVA, L. S.; OLIVEIRA, R. N. Robótica Educacional: Perspectivas e Desafios no Ensino de Ciências e Matemática. 2022. 56 f. Monografia (Bacharelado). Curso de Bacharelado em Engenharia Elétrica. Instituto Federal de Educação, Ciência e Tecnologia de Goiás. Jataí, 2022.

[22] YAMAOKA, Luís Cláudio. CARACTERIZAÇÃO DAS RAÍZES DO POLINÔMIO DO TERCEIRO GRAU E ALGUNS RESULTADOS. Cadernos do IME-Série Matemática, n. 17, p. 50-71, 2021