Instituto Federal de Ciência, Educação e Tecnologia de São Paulo
***Campus* São Paulo**

Bacharelado em Sistemas de Informação
Disciplina de Vetores, Geometría Analítica e Álgebra Linear

ALEX PASSOS DA SILVA
ERIK JONATAN MARTINS VIANA
LARISSA LUMY HIGUE

Aplicações de Vetores, GA e Álgebra Linear em TI: Jogos de Estratégia com Vetores

São Paulo
2025

Sumário

1. INTRODUÇÃO

Em diversas situações do mundo real, sejam elas estratégias militares, negociações econômicas ou competições esportivas, com frequência nos deparamos com cenários de conflito nos quais a decisão de um indivíduo depende diretamente da ação de um oponente. Nessas situações, o participante além de escolher uma ação, também deve prever como o adversário reagirá e, a partir disso, deve escolher a melhor estratégia possível para minimizar perdas ou maximizar ganhos.

Para analisar esses cenários de forma lógica, utiliza-se a *Teoria dos Jogos*, um ramo da matemática aplicada. Especificamente, este trabalho focará em um jogo de matriz de dois jogadores com soma zero, que apresenta uma situação na qual o lucro de um jogador resulta, necessariamente, no prejuízo do outro. Será desenvolvido uma aplicação que analisa os elementos da matriz de pagamento para identificar uma solução estratégica, verificando a existência de um "Ponto de Sela" e, na sua ausência, calculando as "Estratégias Mistas" por meio de vetores de probabilidade.

A relevância deste projeto para a área de TI está na implementação prática de algoritmos de tomada de decisão em problemas reais. Implementando conceitos da matéria de Vetores, Geometria Analítica e Álgebra Linear em código, nota-se a sua relação direta com áreas como Inteligência Artificial e Pesquisa Operacional, demonstrando como estruturas de dados matriciais são utilizadas para resolver determinados conflitos e automatizar soluções.

2. FUNDAMENTAÇÃO TEÓRICA

A base matemática deste projeto se baseia na Teoria dos Jogos, especificamente na análise de jogos de dois jogadores com soma zero. Para a resolução destes problemas, utilizaremos ferramentas centrais da Álgebra Linear, como matrizes para a representação das jogadas possíveis e vetores para a definição das estratégias utilizadas.

2.1 Introdução à Teoria dos Jogos

A Teoria dos Jogos é um ramo da matemática aplicada que estuda situações de interação estratégica, nas quais os resultados/*payoffs* dependem não apenas

das próprias escolhas do indivíduo, mas também das ações tomadas por outros participantes. Ou seja, depende das decisões de múltiplos agentes inteligentes (Fei et al., 2021, p.23). O objetivo dessa teoria é analisar o comportamento racional em cenários de conflito ou cooperação, buscando encontrar uma explicação matemática dessas situações de forma a otimizar as estratégias de jogo.

De acordo com Fei et al. (2021), um jogo consiste de pelo menos três elementos: o grupo de jogadores, o grupo de ações que jogadores podem tomar e a função de recompensa que descreve quanto um jogador pode ser recompensado sobre diferentes estados do mundo e das ações tomadas em conjunto pelos jogadores. Os jogos podem ser classificados em diferentes tipos baseado em suas diferenças nesses elementos.

Com isto em mente, dentro deste campo de estudo, são estudados com maior destaque os jogos de soma zero, nos quais dois jogadores competem e o benefício de um é diretamente o prejuízo do outro (Marcia, 2012). Para permitir uma análise matemática desses cenários, é necessário transformar as interações e instruções verbais para uma estrutura numérica. Essa etapa justifica a Álgebra Linear como uma ferramenta essencial para o caso, pois possibilita a criação de uma representação matricial e vetorial nas quais são aplicados algoritmos para a resolução do problema.

2.2. Jogos de soma zero e representação matricial

Um jogo de soma zero descreve uma situação competitiva na qual o ganho de um participante resulta obrigatoriamente na perda do outro, mantendo a soma de todos os resultados igual a zero (Marcia, 2012).

Nestes tipos de jogos, cada um dos jogadores escolhe um entre seus movimentos sem que o outro jogador saiba a sua escolha. Uma vez tomadas as suas decisões, uma tabela de pagamento é usada para determinar a compensação de um jogador ao outro. Matematicamente, esse cenário é composto por uma matriz de pagamentos A de dimensão $m*n$, em que as m linhas representam as estratégias (movimentos) disponíveis para o Jogador X e as n colunas representam as estratégias disponíveis para o Jogador Y. Cada elemento da matriz representa o “payoff” (pagamento) resultante quando o Jogador X escolhe a linha i e o Jogador Y escolhe a coluna j. Valores positivos indicam pagamentos de Y para X, enquanto valores negativos indicam pagamentos de X para Y.

A tabela abaixo apresenta um exemplo de uma destas matrizes, em que o jogador X possui as estratégias A, B e C e o jogador Y possui as estratégias D, E e F. Nesta simulação, se o jogador X escolhesse a estratégia A e o jogador Y escolhesse a estratégia D, o jogador Y teria que pagar o jogador X em 1 ponto.

	D	E	F
A	1	2	-1
B	3	0	1
C	-3	-2	1

2.3 Ponto de sela e estratégias

A primeira etapa na análise do jogo é a verificação da existência de um equilíbrio estável, conhecido como *Ponto de Sela*. Este conceito representa um estado de equilíbrio em jogos de matriz de soma zero, caracterizado por algum dos elementos da matriz A que é, ao mesmo tempo, o valor mínimo de sua linha e o valor máximo de sua coluna. A existência desse ponto mostra que as estratégias de ambos os jogadores convergem (*Maximin* do Jogador X = *Minimax* do Jogador Y), por isso é possível criar uma solução do jogo, na qual sabe-se o resultado da partida e não há vantagens para nenhum jogador em mudar suas ações (Wilheim, 2017).

Neste caso, o Jogador X (que deseja maximizar seu ganho) identifica o valor mínimo de cada linha e escolhe qual contém o maior desses mínimos (estratégia Maximin). Enquanto isso, o Jogador Y (que deseja minimizar a perda) identifica o valor máximo de cada coluna e escolhe qual contém o menor desses máximos (estratégia Minimax). Nessa situação, o *ponto de sela* ocorre na posição (i,j) se, e somente se, o elemento for simultaneamente o mínimo de sua linha i e o máximo de sua coluna j. Caso essa condição seja satisfeita, o jogo possui uma solução em *estratégias puras*, e já se sabe o seu resultado. Considerando que o jogador X escolha sempre a linha *i* e o jogador Y escolha sempre a coluna *j*, há um equilíbrio e não é necessário pensar em diferentes jogadas ou estratégias.

Esse equilíbrio corresponde diretamente a um conceito central na teoria dos jogos, chamado *Equilíbrio de Nash*, que descreve um estado estável em que

nenhum jogador tem incentivo para mudar sua estratégia unilateralmente. Nesse ponto, a escolha de cada jogador é a sua melhor resposta às escolhas dos outros participantes, pois qualquer alteração na sua estratégia, mantendo as dos outros inalteradas, não resultaria em um benefício para ele (Cadilhac, 2021).

2.4 Estratégias mistas com vetores de probabilidade (2x2 e MxN)

No entanto, nos casos em que a matriz de pagamentos não apresenta um ponto de sela (*Maximin≠Minimax*), não há uma única jogada pré estabelecida que satisfaça ambos os jogadores. Assim, se um jogador optar sempre por usar uma mesma estratégia, o oponente pode prever esses movimentos para obter vantagem.

Para resolver isso, utiliza-se agora o conceito de *estratégias mistas*. Nela, a solução não é uma escolha única e melhor possível (como era definido nas estratégias puras), mas passa a ser uma distribuição de probabilidades sobre todas as possíveis ações a serem realizadas, sempre com o objetivo de maximizar o valor esperado do retorno para ambas as partes. Neste novo modelo, as estratégias disponíveis serão representadas por vetores de probabilidade: um *vetor p* para o Jogador X, que terá dimensão *m*, onde cada um de seus componentes representa a probabilidade de X escolher a linha *i*; e um *vetor q* para o jogador Y, com dimensão *n*, onde cada componente representa a probabilidade de Y escolher a coluna *j*. Além disso, para ambos os vetores serem válidos, devem cumprir simultaneamente duas regras: todos os elementos devem ser não negativos (≥ 0) e a soma dos elementos de cada vetor deve ser igual a 1 (para chegar aos 100%) (Wilheim, 2017).

Vale ressaltar também que a probabilidade calculada por esses vetores não define uma sequência fixa de jogadas em um pequeno conjunto, mas sim a chance independente de escolha a cada rodada. Embora as porcentagem sejam definidas matematicamente, a sua aplicação prática deve ser aleatória de modo que seja mantida uma imprevisibilidade, evitando que o oponente consiga deduzir suas jogadas finais por eliminação.

Nos cálculos de probabilidade dos vetores, nota-se que há uma diferença de complexidade dependendo da dimensão da matriz e dos números fornecidos pela matriz. Nas matrizes menores (2x2), a solução pode ser encontrada através de fórmulas algébricas derivadas de sistemas lineares, calculando probabilidades com o objetivo de igualar os pagamentos esperados. Já em jogos maiores (MxN), nem todas as estratégias podem ser úteis, considerando que algumas linhas ou colunas

podem nunca ser jogadas, o que faz com que o algoritmo utilize “suportes” (sub-matrizes dentro da matriz maior) ou métodos iterativos para encontrar o ponto no qual o ganho mínimo de X é maximizado, independentemente das jogadas de Y.

2.5 Estratégia mista via programação linear (simplex aproximado)

Conforme mencionado, uma das estratégias para o cálculo dos vetores é a busca por submatrizes, no entanto, isso pode se tornar computacionalmente mais complexa em matrizes maiores, considerando que possíveis “subjogos” irão crescer de forma exponencial. Não sendo encontradas estratégias mistas otimizadas, o problema se torna um de otimização matemática, relacionado a programação linear.

Nesse caso, o objetivo do cálculo passa a ser maximizar ou minimizar uma função, deixando de ser uma resolução de equação simples (como na matriz 2x2). O Jogador X precisa encontrar o vetor de possibilidades *p* que maximize o valor do jogo otimizado *v*, garantindo que seu ganho seja pelo menos o valor *v*, independente do que seu oponente fizer.

Para solucionar esse sistema de inequações lineares (considerando que o ganho de X deve ser $\geq v$) utiliza-se o método Simplex, que interpreta as restrições do problema como as faces de um poliedro convexo, que seria a região viável de soluções. O método vai navegando pelos vértices dessa estrutura, até localizar um vértice (elemento) que maximize a função, o que é justamente o necessário para resolver o problema.

A utilização dessa abordagem no algoritmo aumenta o sucesso da aplicação, pois enquanto as fórmulas mais diretas resolvem as matrizes 2x2 e o método de encontrar suportes resolvem casos intermediários, a utilização de um Simplex (ainda que de forma simplificada) serve como uma garantia de que o software encontrará alguma maneira de resolver o problema, não importando a complexidade ou tamanho da matriz (Cadilhac, 2021).

3. APLICAÇÃO E DESENVOLVIMENTO

O desenvolvimento da solução foi realizado em JavaScript, utilizando o ambiente de execução Node.js para permitir a interação direta via terminal. A escolha dessa tecnologia se deve à sua simplicidade, portabilidade e capacidade de lidar com entrada e saída de dados de forma eficiente.

3.1 Planejamento e estrutura do código

O código foi planejado de forma modular, com funções independentes responsáveis por cada etapa da resolução do jogo. Essa abordagem facilita a manutenção e compreensão da lógica, além de permitir que diferentes métodos de solução sejam aplicados conforme o tipo de matriz. A interação com o usuário é feita por meio da biblioteca *prompt-sync,* que possibilita a leitura de dados diretamente no terminal.

3.2 Leitura da Matriz

Primeiramente o programa requisita o tamanho da matriz através da quantidade de movimentos de X (representado pelas linhas) e dos movimentos de Y (colunas). A matriz de pagamentos, então, é lida linha por linha, garantindo que o usuário insira corretamente os valores correspondentes às estratégias dos jogadores. O usuário é instruído a inserir os valores separados por um caractere de espaço “ “.

O programa valida a entrada, verificando se o número de elementos por linha corresponde ao número de colunas informado anteriormente, também é verificado se todos os valores inseridos são numéricos.

3.3 Estratégias

Como destacado no tópico anterior, são utilizadas algumas estratégias para a resolução de Jogos de Soma Zero, como a estratégia pura com ponto de sela ou algumas opções de estratégias mistas, nesta solução estão implementadas 3:

- Matriz fixa 2x2;
- Suporte Duplo; e
- Programação Linear Aproximada.

Cada função foi implementada com cálculos específicos, como operações de soma ponderada, normalização de vetores de probabilidade e verificação de restrições de indiferença.

3.3.1 Ponto de Sela

Como já foi explicado anteriormente, o ponto de sela é o estado de equilíbrio em Jogos de Soma Zero, onde o elemento mínimo de uma linha é o máximo de sua

coluna. Segue o exemplo: $A = \left|\frac{4\ 6}{5\ 7}\right|$, nesta matriz o ponto de sela é definido pelo vetor (2, 1), pois o elemento $A_{2,1}$ é simultâneamente a melhor escolha para ambos os jogadores.

A função de ponto de sela foi implementada de forma modular, se utilizando de duas funções auxiliares para identificar os valores mínimos e máximos. A primeira delas, *minIndicesLinha(),* percorre uma linha da matriz e retorna o menor valor encontrado junto com todos os índices de coluna em que este valor ocorre. A segunda, *maxIndicesColuna()*, faz o mesmo processo em uma coluna específica, retornando o maior valor e os índices de linha correspondentes.

Com essas informações, a função principal percorre cada linha da matriz, identifica os elementos mínimos e verifica se algum deles também corresponde ao valor máximo da respectiva coluna. Caso essa condição seja satisfeita, significa que o elemento é um ponto de sela, e a função retorna sua posição e valor. Se nenhum elemento atender simultâneamente às duas condições, a função conclui que não há ponto de sela na matriz.

3.3.2 Matriz Fixa 2x2

Quando o jogo é representado por uma matriz 2x2, é possível encontrar diretamente o equilíbrio em estratégias mistas utilizando fórmulas fechadas. Esse caso é especial porque as condições de indiferença entre as estratégias dos jogadores geram um sistema simples de equações lineares, cuja solução fornece as probabilidades ótimas de cada movimento.

Nesse exemplo, as probabilidades são calculadas a partir dos elementos da matriz $A = \left|\frac{a\ b}{c\ d}\right|$. O jogador X escolhe sua primeira linha com probabilidade $\frac{d-c}{a-b-c+d}$, enquanto o jogador Y escolhe sua primeira coluna com probabilidade $\frac{d-b}{a-b-c+d}$. O valor do jogo é dado por $\frac{a*d-b*c}{a-b-c+d}$.

A função foi implementada de forma direta, extraindo os quatro elementos da matriz e aplicando essas fórmulas. Caso o denominador seja nulo, o jogo é considerado degenerado, pois não há solução mista válida. Quando as probabilidades calculadas estão entre [0, 1], a função retorna os vetores de estratégias mistas *p* e *q*, juntamente do valor do jogo *v*. Desta forma, o programa

consegue resolver jogos 2x2 sem necessidade de iterações, aproveitando a simplicidade da estrutura e garantindo uma solução exata para o equilíbrio Nash.

3.3.3 Suporte Duplo

Quando não existe ponto de sela e a matriz não é 2x2, uma alternativa é verificar equilíbrios mistos em subconjuntos de estratégias, chamados de suporte duplo. Nesse método, selecionam-se duas linhas *r* e *s,* e duas colunas *c* e *d,* e o programa tenta encontrar probabilidades que tornem os jogadores indiferentes entre essas escolhas.

A função foi implementada para realizar esse processo. Primeiramente, ela calcula as condições de indiferença: encontra os valores de q_c e q_d que igualam os payoffs das linhas *r* e *s,* e em seguida determina p_r e p_s que igualam os payoffs das colunas *c* e *d*. Se os denominadores das equações forem nulos, significa que não há solução válida para aquele suporte.

Depois, a função verifica se as probabilidades obtidas são válidas, isto é, se estão dentro do intervalo [0, 1] com tolerância numérica. Em seguida, calcula o valor do jogo *v*, que deve ser o mesmo para as duas linhas do suporte, garantindo consistência. Para validar a solução, o algoritmo checa se nenhuma linha fora do suporte oferece payoff maior que *v* contra *q*, e se nenhuma coluna fora do suporte oferece payoff menor que *v* contra *p.*

Se todas as condições forem satisfeitas, a função constroi os vetores completos de probabilidades *p* e *q*, normaliza-os para garantir que somem 1 e retorna o equilíbrio encontrado junto com o valor do jogo. Caso contrário, retorna *null,* indicando que não há solução válida para aquele suporte.

3.3.4 Programação Linear Aproximada (Simplex Iterativo)

Quando não há ponto de sela, nem solução direta em matrizes 2×2 ou por suporte duplo, o programa utiliza um método geral de aproximação inspirado em algoritmos de otimização, chamado aqui de simplex iterativo. A ideia é ajustar gradualmente a estratégia mista do jogador Y até que se aproxime do equilíbrio de Nash, garantindo que o valor do jogo seja minimizado para X e maximizado para Y.

A função inicia atribuindo probabilidades uniformes às colunas de Y e define o valor do jogo *v* como infinito. Em cada iteração, calcula os payoffs esperados de

todas as linhas contra a estratégia atual de Y, identifica a linha mais favorável a X (a de maior payoff) e utiliza essa linha como gradiente para atualizar o vetor de probabilidades *q*. O ajuste é feito com um passo de aprendizado $(\alpha = 0.01)$, seguido de projeção para eliminar valores negativos e normalização para garantir que a soma das probabilidades seja igual a 1. Esse processo é repetido milhares de vezes para garantir uma maior precisão, aproximando *q* da estratégia ótima de Y.

Após as iterações, o programa recalcula os payoffs contra *q* e identifica quais linhas de X estão no suporte ótimo, atribuindo probabilidades a elas para formar o vetor *p*. O valor do jogo *v* é calculado como a média entre o menor e o maior payoff obtidos contra a estratégia final de Y. Dessa forma, a função retorna as estratégias mistas *p* e *q,* juntamente com o valor do jogo, oferecendo uma solução aproximada para matrizes de qualquer dimensão.

3.5 Função Principal e Resultados

A função *main()* é responsável por iniciar a análise do jogo de soma zero, organizar o fluxo de execução e apresentar os resultados ao usuário. Ela atua como controlador central, chamando os métodos apropriados conforme o tipo de solução que o jogo permite.

Logo no início, o programa exibe um cabeçalho informativo no terminal e chama a função *lerMatriz()*, que coleta os dados de entrada do usuário e retorna a matriz de pagamentos A, junto com suas dimensões *m* (linhas) e *n* (colunas).

Em seguida, o programa verifica se existe um ponto de sela na matriz, utilizando a função *pontoDeSela()*. Se um ponto de sela for encontrado, significa que há uma estratégia pura ótima para ambos os jogadores. Nesse caso, a função *interpretarPuro()* é chamada para imprimir os movimentos fixos e o valor do jogo, e a execução é encerrada com return.

Caso não exista ponto de sela, o programa chama a função *estrategiasMistasGenerico()*, que tenta encontrar uma solução mista por meio das três abordagens:

- Fórmula direta para matrizes 2×2;
- Suporte duplo; e
- Programação linear aproximada.

Se alguma dessas abordagens retornar uma solução válida, a função *interpretarMisto()* é chamada para imprimir os vetores de probabilidades *p* e *q*, além do valor esperado do jogo *v*. Caso nenhuma estratégia mista seja encontrada, o programa informa ao usuário que não foi possível resolver o jogo com os métodos disponíveis.

Essa estrutura modular e condicional garante que o programa seja capaz de lidar com diferentes tipos de matrizes, aplicando o método mais eficiente e interpretando os resultados de forma clara e acessível.

4. EXECUÇÃO E TESTES

4.1 Ponto de Sela

Para testar o algoritmo com o Ponto de Sela podemos utilizar do exemplo a seguir: $A = \begin{vmatrix} 4 & 6 \\ 5 & 7 \end{vmatrix}$. O esperado é o programa encontrar o elemento do ponto de sela (o número 5 em $A_{2,1}$) e exibir a estratégia pura, sendo ela o movimento 2 para X e o movimento 1 para Y. O valor do jogo é igual ao valor do ponto de sela.

```
==== Análise de Jogo de Matriz (Soma Zero) ====
Número de movimentos do jogador X (linhas): 2
Número de movimentos do jogador Y (colunas): 2
Digite a matriz A linha por linha (valores separados por espaço):
Linha 1: 4 6
Linha 2: 5 7

Resultado: Estratégia pura (ponto de sela).
- Jogador X deve escolher permanentemente o movimento 2.
- Jogador Y deve escolher permanentemente o movimento 1.
- Valor do jogo (pagamento de Y para X): 5.000000
```

4.2 Matriz 2x2

Para testar a Estratégia Mista com Matriz Fixa 2x2 podemos utilizar o exemplo a seguir: $A = \begin{vmatrix} 2 & 0 \\ 0 & 1 \end{vmatrix}$. O esperado pode ser indicado pela aplicação da fórmula:

1. Cálculo do denominador: $D = a - b - c + d = 2 - 0 - 0 + 1 = 3.$
2. Probabilidade de X: $p_1 = \frac{d-c}{D} = \frac{1-0}{3} = \frac{1}{3}\ ou\ \approx 0.333333$ e $p_2 = 1 - \frac{1}{3} = \frac{2}{3}\ ou\ \approx 0.666667.$

3. Probabilidade de Y: $q_1 = \frac{d-b}{D} = \frac{1-0}{3} = \frac{1}{3}\ ou\ \approx 0.333333$ e $q_2 = 1 - \frac{1}{3} = \frac{2}{3}\ ou\ \approx 0.666667.$
4. Valor do Jogo: $v = \frac{ad-bc}{D} = \frac{2*1-0}{3} = \frac{2}{3}\ ou\ \approx 0.666667.$

```
==== Análise de Jogo de Matriz (Soma Zero) ====
Número de movimentos do jogador X (linhas): 2
Número de movimentos do jogador Y (colunas): 2
Digite a matriz A linha por linha (valores separados por espaço):
Linha 1: 2 0
Linha 2: 0 1

Resultado: Estratégias mistas (equilíbrio).
- Jogador X deve escolher os movimentos com as seguintes probabilidades:
  * Movimento 1: 0.333333
  * Movimento 2: 0.666667
- Jogador Y deve escolher os movimentos com as seguintes probabilidades:
  * Movimento 1: 0.333333
  * Movimento 2: 0.666667
- Valor do jogo (pagamento esperado de Y para X): 0.666667
```

4.3 Suporte duplo

Para testar a Estratégia Mista com Suporte Duplo podemos utilizar o exemplo a seguir:

$$A = \begin{vmatrix} 2 & 0 & 1 \\ 1 & 3 & 0 \\ 0 & 2 & 4 \end{vmatrix}$$

O suporte testado é (em vermelho):

- Linhas: r = 1, s = 2; e
- Colunas: c = 1, d = 2.

1. Indiferença para X (resolve q):
   a. Linha 1: $2q_1 + 0q_2 = 2q_1$.
   b. Linha 2: $1q_1 + 3q_2$.
   c. Com $q_1 + q_2 = 1$, igualando:

$$2q_1 = 1q_1 + 3(1 - q_1) \Rightarrow 2q_1 = q_1 + 3 - 3q_1$$

$$2q_1 = 3 - 2q_1 \Rightarrow 4q_1 = 3 \Rightarrow q_1 = 0,75,\ q_2 = 0,25\ e\ q_3 = 0$$

2. Indiferença para Y (resolve p):
   a. Coluna 1: payoff = $2p_1 + 1p_2$.
   b. Coluna 2: payoff = $0p_1 + 3p_2$.

c. Com $p_1 + p_2 = 1$, igualando:

$$2p_1 + p_2 = 3p_2 \Rightarrow 2p_1 = 2p_2 \Rightarrow p_1 = p_2$$

$$p_1 = 0,5,\ p_2 = 0,5\ e\ p_3 = 0$$

3. Valor do jogo:
   a. Linha 1: $2(0,75) + 0(0,25) = 1,5$.
   b. Linha 2: $1(0,75) + 3(0,25) = 1,5$.
   c. Linha 3 contra q: $0(0,75) + 2(0,25) = 0,5 \leq 1,5$ (coerente).

4. Resultado esperado:
   a. Jogador X: (0.5, 0.5, 0);
   b. Jogador Y: (0.75, 0.25, 0); e
   c. Valor do Jogo: 1,5.

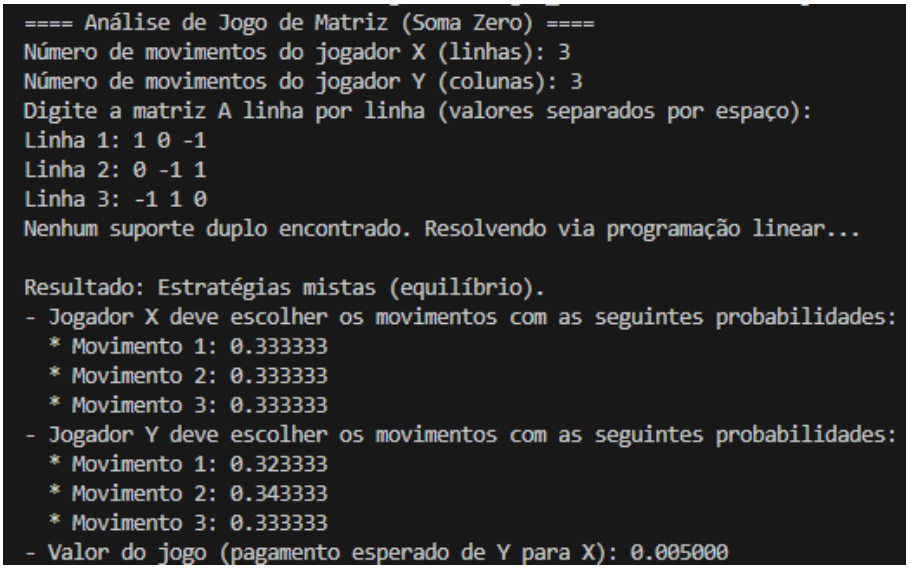
```
==== Análise de Jogo de Matriz (Soma Zero) ====
Número de movimentos do jogador X (linhas): 3
Número de movimentos do jogador Y (colunas): 3
Digite a matriz A linha por linha (valores separados por espaço):
Linha 1: 2 0 1
Linha 2: 1 3 0
Linha 3: 0 2 4

Resultado: Estratégias mistas (equilíbrio).
- Jogador X deve escolher os movimentos com as seguintes probabilidades:
  * Movimento 1: 0.500000
  * Movimento 2: 0.500000
  * Movimento 3: 0.000000
- Jogador Y deve escolher os movimentos com as seguintes probabilidades:
  * Movimento 1: 0.750000
  * Movimento 2: 0.250000
  * Movimento 3: 0.000000
- Valor do jogo (pagamento esperado de Y para X): 1.500000
```

4.4 Programação Linear

Para esse teste, o exemplo usado é o de um jogo de Pedra, Papel e Tesoura, nele o equilíbrio esperado é totalmente simétrico: ambos os jogadores misturam uniformemente entre as três estratégias, ou seja, p = ($\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{3}$) e q = ($\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{3}$). O valor do jogo é 0, indicando que nenhum jogador tem vantagem estrutural.

Pequenas variações numéricas como q ≈ (0.323, 0.343, 0.333) e v ≈ 0.005 são típicas de aproximação iterativa e convergem para o equilíbrio uniforme.



5. CONCLUSÃO

O desenvolvimento deste projeto permitiu que fosse estabelecida uma relação prática entre a Álgebra Linear e as aplicações matemáticas teóricas de seus conceitos na TI, demonstrando como modelos matemáticos abstratos (no caso, os jogos de soma zero e suas especificações) podem ser traduzidos de forma direta para algoritmos de tomada de decisão. O objetivo principal, definido como a criação desta aplicação capaz de solucionar esses jogos, foi atingido, de forma que fossem oferecidas soluções e métodos de resolução adaptáveis para diferentes cenários.

Sob uma análise teórica, a aplicação prática do algoritmo evidenciou a importância das estruturas matriciais tanto como forma de armazenar dados do problema quanto como o “local” onde esse conflito dos oponentes ocorre. Na primeira verificação da existência do *ponto de sela*, comprovou-se que em situações de equilíbrio perfeito, a melhor estratégia é previsível facilmente com fórmulas algébricas. No entanto, há um aumento de complexidade e mudanças no algoritmo ao se acrescentar a manipulação de vetores de probabilidade nas estratégias mistas, justificado principalmente pelo crescimento da dimensão da matriz (mais elementos a serem testados).

Do ponto de vista da área de Tecnologia, o projeto ilustrou os fundamentos da Inteligência Artificial e Pesquisa Operacional. A implementação de diferentes métodos de resolução do caso destacou a necessidade de uma eficiência nos algoritmos, pois o software não se limita a apenas calcular números e probabilidades simples, como também busca simular um agente racional que deseja

maximizar seus ganhos enquanto limita suas perdas. Essa simulação e automação dos cálculos que um jogador teria que fazer por conta própria representa um conceito central no desenvolvimento de sistemas autônomos e negociações automatizadas.

Com isso, conclui-se que a integração entre Vetores, Geometria Analítica, Álgebra Linear e Programação forma uma combinação ideal para a resolução de problemas complexos. O estudo dos conceitos teóricos e sua aplicação no código demonstrou que a matemática pode fornecer uma base lógica essencial para encontrar soluções e construir softwares progressivamente mais eficientes.

6. REFERÊNCIAS

CADILHAC, Igor Tokuichi Kikuchi. Uma análise da relação entre a programação linear e a teoria dos jogos. 2021. Trabalho de Conclusão de Curso (Bacharelado em Ciências Econômicas) - Universidade Federal do Paraná, Curitiba, 2021. Disponível em: https://acervodigital.ufpr.br/handle/1884/76556 Acesso em: 01 dez. 2025.

FANG, F. et al. Introduction to Game Theory. Game Theory and Machine Learning for Cyber Security, p. 21–46. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119723950.ch2 Acesso em: 12 set. 2021

RUGGIERO, Márcia Aparecida Gomes. Estratégias para um jogo de matriz de dois jogadores com soma zero. Álgebra Linear e Aplicações. Campinas: IME-UNICAMP, [s.d.]. Disponível em: https://www.ime.unicamp.br/~marcia/AlgebraLinear/aplicacao_jogos_estrategia.html. Acesso em: 01 dez. 2025.

WILHELM, Volmir Eugênio. Introdução à Teoria dos Jogos. Curitiba: Universidade Federal do Paraná, [s.d.]. Disponível em: https://docs.ufpr.br/~volmir/Jogos_0.pdf Acesso em: 01 dez. 2025