OPERAÇÕES COM MATRIZES: UMA ABORDAGEM ENVOLVENDO LINGUAGEM COMPUTACIONAL

Manuel López Grijalva
Profª Deise Deolindo Silva

nicamlg@gmail.com
deisedeolindo@hotmail.com

Faculdade de Tecnologia de Garça (FATEC-Garça)
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Resumo: *Como parte do processo de aprendizado da disciplina Programação Linear e Aplicações, os alunos deveriam desenvolver um projeto envolvendo a programação de funções matemáticas. O objetivo seria elaborar um sistema computacional que executasse o cálculo de operações básicas sobre matrizes, especificamente seria necessário resolver a soma, a subtração, o produto, o determinante de matrizes e sistemas de equações lineares. O referido trabalho foi programado em Javascript e HTML, pois facilita o uso e também a programação, simplificando a entrada de dados e a abstração do software da plataforma. Como resultado observa que os alunos compreenderam melhor o processo matemático das operações requeridas, assim como treinaram a codificação de algoritmos matemáticos.*

Palavras-chave*: Programação Linear, Matrizes, Sistemas Lineares, Educação Matemática, Javascript.*

Abstract*: As part of Linear Programming and Applications learning process, the students should develop a project tha involves programming mathematical functions. Its goal would be to create a computational system able to execute the necessary calculations with matrixes, more specifically to solve basic operations such as addition, subtraction, multiplication and determination of matrixes and linear equations. This work has been programmed in Javascript and HTML, because they make it easier its use and also the programming, simplifying data entry, as well as to abstract the software platform. As result it was observed that the alumni did better understand the mathematical process of the required operations, and they also trained their programming skills of mathematical algorithms.*

Keywords*: Linear Programming, Matrixes, Linear Systems, Math Education, Javascript.*

1 INTRODUÇÃO

Ao estudar o projeto pedagógico do curso de Tecnologia em Análise e Desenvolvimento de Sistemas (ADS), observa-se que o tecnólogo deve analisar e manter sistemas computacionais de informação, trabalhar com ferramentas computacionais e equipamentos informáticos, ter bom raciocínio lógico e empregar linguagens de programação.

A disciplina Programação Linear e Aplicações, ministrada no quinto termo do curso, com 80 horas-aula, tem por objetivo conhecer e aplicar os conhecimentos de Pesquisa Operacional e desenvolver aplicativos para essa área. Sua ementa contempla tópicos matemáticos (matrizes, determinantes, sistemas lineares) e específicos de pesquisa operacional (método simplex e dos transportes).

Portanto, a disciplina sugere o desenvolvimento, em uma linguagem de programação, de conteúdos específicos da matéria, com o intuito de sondar os resultados do processo de ensino-aprendizagem de Pesquisa Operacional e Matemática, além de contemplar a interdisciplinaridade e a contextualização da própria disciplina.

A proposta deste trabalho é codificar de um programa capaz de realizar operações básicas com matrizes (adição, subtração, produto de duas matrizes, multiplicação de uma matriz por um escalar e o cálculo do determinante de uma matriz, a princípio de ordem 2, 3 ou 4) e a resolução de sistemas lineares pelo método de Crammer.

Observa-se que os tópicos a serem programados pertencem à área de Matemática designada Álgebra Matricial. Os conceitos deste campo muitas vezes são considerados complexos, pois diferem da aritmética convencional, e uma forma de compreendê-los é descompondo esses processos de forma lógica, criando algoritmos que mostrem aos estudantes os passos a serem dados para a resolução de cada problema. Em se tratando de um curso de Tecnologia da Informação, o desenvolvimento desses algoritmos pode resultar facilmente na elaboração de um programa de informática que solicite os dados do problema e mostre o resultado, além de tornar o processo do cálculo matricial mais claro.

Para desenvolver este projeto escolheu-se a linguagem *JavaScript* (especificamente, *ECMAScript v.5*) usando uma interface para a *web* (HTML e CSS3). Para chegar a esta escolha, consideraram-se vários critérios: usabilidade, portabilidade, acessibilidade e adaptação da linguagem ao problema considerado.

A usabilidade considera o uso de um ambiente e sua interface. O conhecimento prévio do usuário facilita a adaptação do uso do *software*, pois ele já sabe como usar botões, áreas de texto e outros elementos HTML. Com as ajudas incorporadas, o usuário não precisa ter instruções para começar a usar o *software*.

O critério portabilidade garante que o acesso ao programa seja independente do sistema operacional. Pois, devido ao uso da Internet, os computadores e outros dispositivos com conexão à rede têm a possibilidade de abrir os arquivos e executar o sistema, inclusive sem a presença de um servidor *web*.

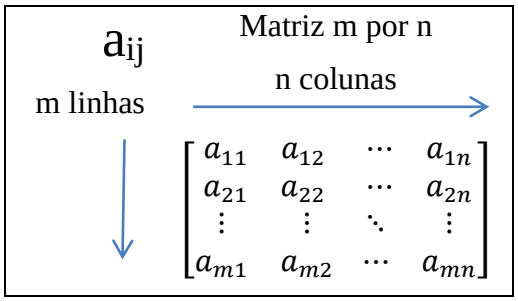
A acessibilidade além de facilitar o acesso a pessoas com alguma deficiência (uso do padrão HTML 5), permite um acesso universal, por conta de sua simplicidade.

O *framework* (neste caso, o conjunto HTML+CSS+JS) facilita a programação ao utilizar linguagens de programação como C e derivados. Percebe-se que não é trivial o manejo de *arrays* (matrizes) ou a conversão da entrada de dados em formato de texto para matrizes. Ao trabalhar com uma linguagem como *JavaScript*, a entrada de dados é simplificada, assim como a conversão e o processamento. Utilizar HTML e CSS3 resulta numa interface gráfica amigável e com os conhecimentos adequados garante uma apresentação esteticamente agradável. Mas também facilita a aplicação do padrão MVC, fazendo com que a lógica do programa esteja separada do restante, facilitando a reutilização do código.

2 CÁLCULOS MATRICIAIS

Na Matemática, uma matriz é um conjunto de números ou outros valores matemáticos (símbolos ou expressões), disposto numa distribuição cartesiana (*m* linhas e *n* colunas). As matrizes são usadas na resolução de sistemas lineares e outras operações, como transformações de espaços lineares (IEZZI; HAZZAN, 2002). A representação de uma matriz com m linhas e n colunas está disposta na Figura 1.

Figura 1. Representação de uma matriz A_{mn}.



Fonte: elaborada pelos autores.

As letras maiúsculas do alfabeto latino são usadas para se referir à matriz, enquanto as minúsculas representam seus elementos. O par de letras *mn* representam o tamanho ou limite da matriz, e usa-se o par *i,j* para referenciar os índices de um elemento específico, que é representado pela letra minúscula da matriz, eles também são chamados de subíndices e representam o "endereço" do elemento, em que o primeiro valor refere-se à linha e a segunda à coluna.

De acordo com Iezzi e Hazzan (2002), as matrizes podem ser somadas, subtraídas ou multiplicadas. Tanto a soma quanto a multiplicação têm propriedades associativas:

$$A + B + C = (A + B) + C \text{ ou } A \times B \times C = A \times (B \times C)$$

Além desta propriedade, a soma tem a propriedade comutativa, enquanto o produto não:

$$A + B = B + A, \text{ mas } A \times B \neq B \times A.$$

A soma (e a subtração) de duas matrizes só é possível se as dimensões das matrizes forem iguais. O processo consiste na adição (ou a subtração) consiste em:

$$A_{mn} + B_{mn} = C_{mn} \mid c_{ij}=a_{ij} + b_{ij}$$

A multiplicação apresenta dois cenários: a multiplicação de um escalar por uma matriz, ou o produto entre duas matrizes. O primeiro caso é considerado mais simples, basta multiplicar cada elemento da matriz pela constante. Assim, $k \times A = k \cdot a_{i,j}$.

O segundo caso tem um processo mais algébrico, o produto de duas matrizes pela Regra de Caley. A mais importante restrição é que para que a multiplicação seja possível, o número de colunas da primeira matriz deve ser igual ao número de linhas da segunda. Para $A_{p,n} \times B_{m,q}$, se $n = m$, então é possível realizar a operação. A formulação matemática do produto é:

$$A = (a_{ij})_{m\times n} \;\; e \;\; B = (b_{ik})_{n\times p} \;\; então \;\; A.B = C = (c_{ik})_{m\times p} \mid c_{ik} = \sum_{j=1}^{n} a_{ij} \cdot b_{jk}$$

$$\forall\, i \in \{1, 2, \ldots, m\} \;\; e \;\; \forall\, k \in \{1, 2, \ldots, p\}$$

Além destas operações matriciais, os sistemas lineares compõem um conteúdo pertinente à temática deste trabalho. Para resolver sistemas lineares, por meio da regra de Cramer, é necessário recorrer aos cálculos de determinantes (D).

O determinante da matriz D é formada pelos coeficientes das variáveis. Posteriormente, calcula-se o determinante Dx_i formada pela substituição da coluna de cada variável (x_i) pela coluna que contém os valores independentes. O resultado do sistema é dado pela seguinte expressão.

$$\boldsymbol{x_i = \frac{Dx_i}{D}}$$

3 DESCRIÇÃO DO PROGRAMA

O ambiente apresentado ao usuário foi projetado para uma aplicação *web* e o *design* foi desenvolvido e testado no intuito de ser o mais simples possível, sem precisar usar manuais ou treinamento sobre seu uso.

As Figuras 1 e 2 apresentam, respectivamente, a tela principal do programa e a resolução de um sistema linear com três variáveis e três equações.

Figura 2. Tela principal do programa

Fonte: elaborada pelos autores.

Figura 3. Exemplificação da resolução de sistemas lineares com três equações e três incógnitas.

Fonte: elaborada pelos autores.

O padrão *Model-View-Controller* (MVC) é um paradigma de interface que supõe a separação lógica e "física" (em diferentes arquivos ou seções) da lógica e processamento de dados (*model*), a visualização e entrada e saída de dados (*view*) e o controle e intercâmbio de informações entre *model* e *view* (*controller*) (KRASNER; POPE, 1988).

Além de esclarecer as responsabilidades, o autor Martin (2002), usou a divisão de preocupações, por meio do *SoC* ou *Separation of Concerns*, de cada seção do código do programa, no intuito do modelo melhorar a legibilidade e a reutilização do código. Como a combinação de HTML, CSS e *JavaScript* supõem uma separação de

preocupações, fica mais fácil aplicar o MVC, deixando a visualização destinada ao HTML (estrutura) e o CSS (estilo), a lógica e o comportamento ao *JavaScript*.

No caso deste programa, implementou-se uma "classe" (um objeto *JavaScript*) chamada Matrix, a qual manipula e processa matrizes. Esta classe converte de *string* de entrada em *array*.

Um segundo arquivo contém o código que manipula e processa os sistemas lineares. A "classe" SL se responsabiliza por interpretar os dados de entrada já no construtor, extraindo os "nomes" das variáveis do sistema linear, os valores dos coeficientes e dos termos independentes.

O sistema linear após ser digitado é recuperado por meio do método **`SL.parseEquations()`**. Para resolver um sistema linear, pela da regra de Cramer, é necessário utilizar o método **`SL.solve()`**.

Um terceiro código *JavaScript* controla o comportamento da interface, respondendo às ações do usuário, enviando os dados inseridos no *model* e mostrando o retorno em tela.

A inserção de dados é realizada em duas áreas de texto `<TEXTAREA>` para as matrizes, e uma para o sistema linear. Os campos de texto HTML que permitem inserir várias linhas de texto de forma direta. A matriz deve ser formatada separando cada coluna com espaço, ponto-e-vírgula ou *pipe* (|), e cada linha da matriz é uma designada pela própria linha.

Os sistemas lineares podem ser digitados normalmente, no entanto, quando os coeficientes forem zero, esses devem ser digitados igualmente e os termos independentes devem estar sempre no lado direito da igualdade.

Para matrizes, o "método" **`Matrix.str2array()`** interpreta o texto gerado, separando as linhas e depois as colunas, usando expressões regulares, como proposto por Jeffrey, (2006). O procedimento é simples e rápido, e não requer o conhecimento prévio do tamanho da matriz, como ocorreria em outras linguagens de programação. Cada item é processado e armazenado como número inteiro ou decimal (*float*) dentro da matriz.

Já para os sistemas de equações, foi programado um *parser* ou interpretador, que extrai de cada termo o coeficiente, em seguida interpreta-o e o adiciona aos dados da equação, posteriormente ordena a equação pelas variáveis. O resultado do *parser* é lançado no **`<TEXTAREA>`** de entrada de dados, substituindo os valores, digitados pelo usuário, pela resolução do sistema.

Existem várias funções que auxiliam os cálculos matriciais:

1. `Matrix.transpose()`: transpõe a matriz.
2. `Matrix.getDim()` lê a matriz e devolve um objeto *JavaScript* com as dimensões (linhas (*m*) e colunas (*n*)) da matriz informada.
3. `Matrix.getValue()` retorna o valor da posição da matriz informada, as posições começam por 1 e não em 0, para facilitar a leitura, sendo que a nomeação dos elementos de uma matriz são com base 1 (a_{11}, a_{12},…, a_{mn}).

4. Cálculo e manipulação de determinantes, a saber: `cofactor()`, `removeRow()`, `updateRow()`, `updateCol()` e `removeCol()`.

Para somar duas matrizes elas devem ter as mesmas dimensões. O método `Matrix.sum()` percorre as linhas e colunas da primeira matriz, e soma o correspondente elemento da segunda, armazenando o resultado numa *string* que tem a mesma formatação que a entrada manual, e finalmente esta *string* é convertida novamente em matriz usando o método `Matrix.str2array()`. O Quadro 1 apresenta o procedimento utilizado.

Quadro 1. Procedimento para calcular a soma de duas matrizes.

```
sum: function(A,B) {
  var S;
  var dimA = this.getDim(A);
  var dimB = this.getDim(B);

  if (dimA.n == dimB.n && dimA.m == dimB.m) {
    S = new Array(dimA.n);
    for (var n=0; n<dimA.n; n++) {
      S[n] = new Array(dimA.m);
      for (var m=0; m < dimA.m; m++) {
        S[n][m] = A[n][m] + B[n][m];
      }
    }
    return S;
  }
}
```

Fonte: elaborada pelos autores.

Para subtrair duas matrizes elas também devem possuir as mesmas dimensões. Neste caso, aproveitou-se o código descrito no Quadro 1 e multiplicou-se a segunda matriz por -1. O Quadro 2 apresenta o procedimento para subtração de matrizes.

Quadro 2. Subtração de duas matrizes.

```
subtract: function(A, B) {
  return this.sum(A, this.product(-1, B)); // inverte B e soma A+B-
}
```

Fonte: elaborada pelos autores.

A função `Matrix.product()` calcula o produto (multiplicação) de duas matrizes. Para realiza-lo é necessário que o número de colunas da primeira matriz seja igual ao número de linhas da segunda. Esta condição deve ser conferida antes de começar o processamento.

O programa também calcula o produto de uma matriz por um número escalar. Inicia o processo verificando se um dos parâmetros é um escalar. Se for, utiliza o método `Matrix.newArray()` para gerar uma nova matriz, na qual todos os elementos tem o mesmo valor do escalar, com as dimensões da matriz transposta, para garantir que cumpre a necessidade do produto de duas matrizes. Com esta nova matriz, o método continua normalmente, sendo assim que sempre é calculado o produto de duas matrizes.

O determinante de uma matriz é calculado pela função `Matrix.det()`. Para matrizes de ordem 2, utiliza-se a regra prática e para as demais dimensões o Teorema de Laplace. Este Teorema afirma que o determinante de uma matriz é igual à soma dos produtos dos elementos de uma linha ou coluna pelos respectivos cofatores (complementos algébricos).

Seja a matriz $A \in M_{m\times n}(K)$, o cofator $a_{i,j}$é o escalar $A_{i,j} = a_{ij}.(-1)^{i+j} \cdot |M_{ij}|$, em que M_{ij} representa a matriz resultante de retirar a *i*-ésima linha e a *j*-ésima coluna. O determinante de A é então a soma dos cofatores (IEZZI; HAZZAN, 2002).

Foram criados métodos para calcular o cofator, transpor e excluir linhas e colunas. Separar o código, criando funções mais específicas, agiliza a programação de novas funções, e melhora a legibilidade do programa. O Quadro 3 resume esse procedimentos.

Quadro 3. Desenvolvimento do Teorema de Laplace.

```
for (var j=1, detC=0, detJ=0, detM=0; j<dims.n+1; j++) {
  detC = this.cofactor(matrix, 1, j);
  detJ += this.getValue(matrix, 1, j) * Math.pow(-1, j+1) * detC;
}
return detJ;
```

Fonte: elaborada pelos autores.

Para a resolução do Sistema Linear pela Regra de Cramer utilizou-se o seguinte algoritmo:

Quadro 4. Algoritmo para a resolução de Sistemas Lineares.

```
function solve() {
  var m      = this.matrix;
  var detJ={}, detJ={}; // Faz cópia da matriz
  var ind    = Matrix.str2array(Matrix.transpose(m)[m.length].join("\n"));
  var Mi     = Matrix.removeCol(m, order+1);
  var D.detD = Matrix.det(Mi);
  for (var Dv=0, ctrl=0, i=0; i < order;) {
    var v = this.vars[i++]; // nome da variável
    var submatrix = Matrix.updateCol(Mi, ind, i);
    D[v] = Matrix.det(submatrix);
    ctrl += Math.abs(D[v]); // Para saber se tudo é 0, tem que usar abs()
    if (D.detD !== 0) {
      res[v] = D[v]/D.detD;
    }
  }
  if (D.detD === 0 && ctrl !== 0)
    return "Sistema sem solução!";
  if (ctrl === 0)
    return "Sistema INDETERMINADO, com soluções infinitas!";
  return res;
}
```

Fonte: elaborada pelos autores.

Para cada coluna de coeficientes é criada uma nova matriz, substituindo a coluna da variável com a dos termos independentes (`Matrix.updateCol(matriz, ind, col)`), e calculamos o determinante da matriz resultante. Depois é calculado o valor de cada incógnita (D_{xi}/D) e armazenado num objeto com o nome da variável.

O programa realiza algumas características dos sistemas lineares

1. Se o determinante dos coeficientes das incógnitas for diferente de zero o sistema é dito Sistema Possível e Determinado (SPD) e tem somente uma solução;
2. Se todos os determinantes forem zero o sistema linear é possível e indeterminado (SPI) e possui infinitas soluções.
3. Se pelo menos uma variável tiver determinante diferente de 0 e o determinante da matriz de incógnitas for zero, tem-se um sistema impossível (SI), ou seja, não existe solução.

Finalmente o *controller* se encarregará de mostrar uma mensagem ou a solução do sistema, finalizando o processo.

4. CONSIDERAÇÕES FINAIS

Durante a codificação, para organizar adequadamente cada função, foi necessário analisar cada processo, separar todos os subprocessos do cálculo e as ações de manipulação que seriam necessárias.

Cada uma dessas partes teve de ser programada de forma que pudesse ser utilizada por outras funções. Após reduzir cada método em diferentes passos, a programação ficou concentrada em resolver o problema menor de cada passo, criando ferramentas (funções auxiliares) que resolvessem esses problemas. Em seguida procede-se os passos que necessitam de maior complexidade.

Observa-se que o sistema computacional desenvolvido atingiu os objetivos propostos pelo docente da disciplina e possibilitou a interdisciplinaridade entre disciplinas da área de Exatas e da Informática.

5 BIBLIOGRAFIA

KRASNER, G. E.; POPE, S. T. **"A description of the model-view-controller user interface paradigm in the smalltalk-80 system".** Journal of object oriented programming v.1, p. 26-49, 1988.

MARTIN, R. C. **Agile Software Development, Principles, Patterns, and Practices.** Prentice Hall, 2002.

IEZZI, G.; HAZZAN, S. **Fundamentos de Matemática Elementar**. São Paulo: SARAIVA, S.A. vol. 4, 2002.

JEFFREY, F. **Mastering regular expressions**. O'Reilly Media, Inc., 2006

MOZILLA FOUNDATION. **Mozilla Developer Network.** Disponível em: https://developer.mozilla.org. Acesso em: abril de 2015.