UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE MATEMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA
EM REDE NACIONAL

HUGO VICTOR SILVA

SOLUÇÃO DE SISTEMAS LINEARES ATRAVÉS DE MÉTODO COMPUTACIONAL PARA ALUNOS DA EDUCAÇÃO BÁSICA

FORTALEZA
2013

HUGO VICTOR SILVA

SOLUÇÃO DE SISTEMAS LINEARES ATRAVÉS DE MÉTODO COMPUTACIONAL PARA ALUNOS DA EDUCAÇÃO BÁSICA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Matemática em Rede Nacional (PROFMAT), do Departamento de Matemática da Universidade Federal do Ceará, como requisito parcial para obtenção do título de Mestre em Matemática. Área de Concentração: Ensino de Matemática.

Orientador: Prof. Dr. José Fábio Bezerra Montenegro

FORTALEZA

2013

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Curso de Matemática

S58s Silva, Hugo Victor
Solução de sistemas lineares através de método computacional para alunos da educação básica / Hugo Victor Silva. – 2013.
41 f. : il., enc.; 31 cm

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Departamento de Matemática, Programa de Pós-Graduação em Matemática em Rede Nacional, Fortaleza, 2013.
Área de Concentração: Ensino de Matemática.
Orientação: Prof. Dr. José Fábio Bezerra Montenegro.

1. Matrizes (Matemática). 2. Lógica computacional. I. Título.

CDD 512.9434

HUGO VICTOR SILVA

SOLUÇÃO DE SISTEMAS LINEARES ATRAVÉS DO MÉTODO COMPUTACIONAL PARA ALUNOS DA EDUCAÇÃO BÁSICA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Matemática em Rede Nacional, do Departamento de Matemática da Universidade Federal do Ceará, como requisito parcial para a obtenção do Título de Mestre em Matemática. Área de concentração: Ensino de Matemática.

Aprovada em: 08 / 08 / 2013.

BANCA EXAMINADORA

José Fabio Bezerra Montenegro

Prof. Dr. José Fábio Bezerra Montenegro (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Marcelo Ferreira de Melo
Universidade Federal do Ceará (UFC)

Prof. Dr. Luiz Antonio Caetano Monte
Universidade de Fortaleza (UNIFOR)

AGRADECIMENTOS

À CAPES, pelo apoio financeiro com a manutenção da bolsa de auxílio.

Ao Prof. Dr. José Fábio Bezerra Montenegro, pela excelente orientação.

Aos professores participantes da banca examinadora: Prof. Dr. Marcelo Ferreira de Melo e Prof. Dr. Luiz Antônio Caetano Ponte pelo tempo e pelas valiosas sugestões.

Aos colegas de turma de mestrado, em especial à Profa. Ms. Carina Brunehilde Pinto da Silva, pelos momentos de discussão e reflexões.

À Profa. Ms. Patrícia Batista, por todas as importantes dicas e criteriosas correções.

RESUMO

Sistemas de equações lineares é um tema abordado no ensino médio de forma limitada principalmente por conta do excesso de cálculos a serem feitos para encontrar seu conjunto solução. A presente dissertação tem como objetivo criar e disponibilizar uma ferramenta aos professores que têm interesse em ensinar a encontrar a solução de sistemas lineares diversos aos seus alunos da educação básica utilizando a lógica computacional já que este trabalho contém ferramentas para a introdução do estudo de lógica computacional aliada a uma linguagem de programação simples e acessível aos estudantes tanto para o ensino médio regular como para a educação tecnológica. Além disso, apresentamos uma interessante aplicação em um dos problemas tradicionais da química, o balanceamento de reações. Nesta dissertação, também faremos uma abordagem de algumas definições necessárias envolvendo matrizes e suas operações. Além disso, veremos como utilizar uma linguagem de programação simples que é usada como ferramenta de aprendizagem inicial na maioria dos cursos básicos de linguagem de programação, o Português Estruturado. Pontuaremos alguns elementos da sintaxe do VisualG 2.5, compilador gratuito e autoexecutável que irá nos ajudar a atingir nosso objetivo. Veremos como encontrar a solução de um sistema de equações lineares com um número finito de variáveis usando o método do escalonamento de sistemas, a substituição retroativa e seguindo um algoritmo computacional para chegar a tal solução.

Palavras-chave: Matrizes. Programação. Sistemas Lineares.

ABSTRACT

Systems of linear equations are a topic covered in high school so limited mainly due to the excess calculations to be made to find a whole solution. This dissertation aims to create and provide a tool for teachers who are interested in teaching find the solution of several linear systems to their basic education students using computational logic as this work contains tools for introducing the study of computational logic ally a simple programming language, accessible to students both as a regular high school for technology education. Furthermore, we present an interesting application in one of the traditional problems of chemistry, balancing reactions. In this thesis we will also address some necessary definitions and operations involving matrices. Moreover, we will see how to use a simple programming language that is used as a tool for initial learning in most courses basic programming language, the Structured Portuguese. We will point some elements of syntax VisuAlg 2.5, free compiler and auto executable that will help us achieve our goal. We will see how to find the solution of a system of linear equations with a finite number of variables using the method of scheduling systems, the backward substitution and following an algorithm for finding such a solution.

Keywords: Matrices. Programming. Linear Systems.

SUMÁRIO

1 INTRODUÇÃO

Historicamente, os orientais já representavam sistemas por meio de seus coeficientes escritos sobre os quadrados de um tabuleiro. Assim, acabaram descobrindo o método de resolução por eliminação, que consiste em anular coeficientes por meio de operações elementares. É atribuído ao astrônomo, matemático e físico alemão Karl Fredrich Gauss (1777-1855) o método de solução de sistemas por escalonamento.

O conteúdo de sistemas lineares, comum nos ensinos fundamental e médio, é ensinado, na maioria das vezes, com o auxílio de situações-problema limitado à quantidade de, no máximo, três equações com três variáveis. Durante o ensino médio, esse processo é visto como trabalhoso e cansativo pelos alunos. Nessa perspectiva, o objetivo deste trabalho é propor mais uma ferramenta para o ensino de sistemas lineares na educação básica utilizando o computador, equipamento de total importância para esse método e que torna o aprendizado mais atrativo para o aluno.

Esse trabalho traz uma breve abordagem sobre conceitos envolvendo matrizes e suas operações, já que estes serão úteis para resolver o problema de forma computacional. Veremos a sintaxe de uma linguagem de programação, o Português Estruturado (também chamado de Portugol), que por ser de fácil aprendizagem, possuir uma sintaxe simples e por ser copilado pelo VisuAlg 2.5, um compilador gratuito, nos ajudará a não desfocar do nosso objetivo principal. Além disso, também são apresentadas ferramentas para que possamos escalonar sistemas lineares e, após serem escalonados, podermos afirmar se tal sistema linear possui ou não solução e, se tal sistema possuir solução única, exibir esta solução. Objetivou-se que a resolução não se limitasse a apenas sistemas com três equações com três variáveis, mas se estendesse a sistemas com mais equações e variáveis.

2 MATRIZES

2.1 Definição

Chamamos de *matriz* $m \times n$(sobre $\mathbb{R}$), com *m* e *n* inteiros, qualquer lista ordenada de *m.n* números reais, dispostos em *m* linhas e *n* colunas. Os números que constituem uma matriz são chamados de *termos* da matriz.

2.2 Notação

Uma matriz A, $m \times n$, pode ser denotada da seguinte forma:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

ou, simplesmente, $A = (a_{ij})$, onde $1 \leq i \leq m$ e $1 \leq j \leq n$. Os índices *i* e *j* indicam a posição em que o termo ocupa na matriz, ou seja, o termo a_{ij} está na *i*-ésima linha e na *j*-ésima coluna.

2.3 Igualdade de Matrizes

Duas matrizes $A = (a_{ij})$ e $B = (b_{ij})$ serão ditas *iguais* se, e somente se, $a_{ij} = b_{ij}$ para quaisquer que sejam *i* e *j*. Vale lembrar que a condição anterior faz com que tenhamos as matrizes A e B com a mesma quantidade de linhas e de colunas.

2.4 Tipos especiais de Matrizes

2.4.1 Matriz Linha

Uma matriz $m \times n$ é chamada de *matriz linha* se $m = 1$, ou seja, se ela for constituída de uma só linha.

2.4.2 Matriz Coluna

Uma matriz $m \times n$ é chamada de *matriz coluna* se $n = 1$, ou seja, se ela for constituída de uma só coluna.

2.4.3 Matriz Nula

Uma matriz $m \times n$ é chamada de *matriz nula* se todos os seus termos são nulos.

2.4.4 Matriz Quadrada

Uma matriz $m \times n$ é chamada de *matriz quadrada* se $m = n$, ou seja, se a quantidade de linhas for igual à quantidade de colunas. Dizemos que uma matriz quadrada $n \times n$ é de *ordem* n.

Definição 1

Seja $A = (a_{ij})$ uma matriz quadrada de ordem n. Chamamos de *diagonal principal*, ou simplesmente *diagonal* da matriz A, a lista ordenada $(a_{11}, a_{22}, \ldots, a_{nn})$ e de *diagonal secundária* da matriz A a lista ordenada $(a_{1n}, a_{2(n-1)}, \ldots, a_{(n-1)2}, a_{n1})$ (termos cuja soma dos índices é sempre igual a $n + 1$).

2.4.5 Matriz Triangular Superior

Uma matriz quadrada $A = (a_{ij})$ é dita *triangular superior* se todos os termos que ficam abaixo da diagonal principal são iguais a zero, ou seja, $a_{ij} = 0$ sempre que $i > j$.

2.4.6 Matriz Triangular Inferior

Uma matriz quadrada $A = (a_{ij})$ é dita *triangular inferior* se todos os termos que ficam acima da diagonal principal são iguais a zero, ou seja, $a_{ij} = 0$ sempre que $i < j$.

2.4.7 Matriz Diagonal

Uma matriz quadrada $A = (a_{ij})$ é chamada *matriz diagonal* se todos os termos tal que $i \neq j$ forem nulos. Note que toda matriz quadrada nula também é matriz diagonal.

2.4.8 Matriz Identidade

Uma matriz quadrada é dita *matriz identidade* se ela foi uma matriz diagonal cujos termos da diagonal principal são todos iguais a 1. Denotaremos a matriz identidade de ordem n por I_n.

2.5 Operações com matrizes

2.5.1 Adição

Sejam $A = (a_{ij})$ e $B = (b_{ij})$ matrizes $m \times n$. Definimos a soma das matrizes A e B como sendo a matriz $C = (c_{ij})$ em que$c_{ij} = a_{ij} + b_{ij}$, ou seja, somar A com B consiste em somar termos correspondentes.

Observação 1: Podemos somar as matrizes A e B apenas se ambas forem $m \times n$, ou seja, ambas têm a mesma quantidade de linhas e de colunas.

Observação 2: A adição de matrizes possui as seguintes propriedades: associatividade, comutatividade, elemento neutro, elemento inverso. Essas propriedades são facilmente demonstradas por consequência das propriedades dos números reais.

2.5.2 Multiplicação

Sejam $A = (a_{ij})$ uma matriz $m \times n$ e $B = (b_{jk})$ uma matriz $n \times p$. Definimos o *produto escalar* da i-ésima linha de A pela k-ésima coluna de B como sendo

$$A_i.B^k = a_{i1}.b_{1k} + a_{i2}.b_{2k} + \cdots + a_{in}b_{nk} = \sum_{j=1}^{n} a_{ij}b_{jk}$$

Observe que A_i e B_k possuem a mesma quantidade de termos (n), o que torna possível definir o produto escalar entre A_i e B_k.

Definimos o *produto* de A por B como sendo a matriz $m \times p$

$$A.B = \begin{bmatrix} A_1.B^1 & A_1.B^2 & \dots & A_1.B^p \\ A_2.B^1 & A_2.B^2 & \dots & A_2.B^p \\ \vdots & \vdots & \ddots & \vdots \\ A_m.B^1 & A_m.B^2 & \dots & A_m.B^p \end{bmatrix}$$

Observação 3: O produto $A.B$ é uma matriz $m \times p$.

Observação 4: O termo de $A.B$ que se situa na i-ésima linha e k-ésima coluna é $A_i.B_k$.

3 INTRODUÇÃO À PROGRAMAÇÃO – VISUALG

Neste capítulo, apresentamos uma introdução à lógica computacional e a uma linguagem de programação muito simples de usar que é o *VISUALG 2.5*, focado apenas nas palavras e funções que usaremos no capítulo seguinte.

3.1 Algoritmos

Algoritmo é uma sequência de instruções finitas e ordenadas para a resolução de uma tarefa ou problema. São exemplos de algoritmos: instruções de montagem, receitas, manuais de uso de um equipamento. Para que um algoritmo possa ser útil, é necessário que quem for fazer uso dele conheça os termos utilizados nas instruções. No nosso caso, será o computador, por isso trabalharemos com *algoritmos computacionais*.

Para que o algoritmo possa ser utilizado por uma máquina, é importante que as instruções sejam corretas e sem ambiguidades. Portanto, a forma especial de linguagem que utilizaremos é bem mais restrita que o Português e com significados bem definidos para todos os termos utilizados nas instruções. Essa linguagem é conhecida como *Português Estruturado* (ou *Portugol).* O Português Estruturado é, na verdade, uma simplificação do Português, limitada a poucas palavras e estruturas que têm um significado bem definido. Ao conjunto de palavras e regras que definem o formato das sentenças válidas chamamos de *sintaxe da linguagem*.

3.2 Operadores

Os operadores utilizados no Português estruturado são:

3.2.1 Operadores Aritméticos

Quadro 1 - Operadores aritméticos do VisualG

OPERADORES ARITMÉTICOS	PORTUGUÊS ESTRUTURADO
Adição	+
Subtração	-
Multiplicação	*
Divisão	/

Fonte: TONET; KOLIVER, s/d.

3.2.2 Operadores Relacionais

Quadro 2 - Operadores relacionais do VisualG

OPERADORES RELACIONAIS	PORTUGUÊS ESTRUTURADO
Maior	>
Menor	<
Maior ou igual	>=
Menor ou igual	<=
Igual	=
Diferente	<>

Fonte: TONET; KOLIVER, s/d.

3.2.3 Operadores Lógicos

Quadro 3 - Operadores lógicos do VisualG

OPERADORES LÓGICOS	PORTUGUÊS ESTRUTURADO	SIGNIFICADO
Multiplicação lógica	E	Resulta VERDADEIRO se ambas as partes forem verdadeiras.
Adição lógica	Ou	Resulta VERDADEIRO se uma das partes é verdadeira.
Negação	Nao	Nega uma afirmação, invertendo o seu valor lógico: se for VERDADEIRO torna-se FALSO, se for FALSO torna-se VERDADEIRO.

Fonte: TONET; KOLIVER, s/d.

A princípio, a execução é da esquerda para a direita, mas existem prioridades entre os operadores envolvidos na expressão. Tais prioridades são mostradas abaixo.

Quadro 4 - Prioridade dos operadores aritméticos do VisualG

OPERADOR ARITMÉTICO	PRIORIDADE
Multiplicação	4 (maior)
Divisão	3
Adição	2
Subtração	1 (menor)

Fonte: TONET; KOLIVER, s/d.

Quadro 5 - Prioridade dos operadores lógicos do VisualG

OPERADOR LÓGICO	PRIORIDADE
e	3
ou	2
nao	1

Fonte: TONET; KOLIVER, s/d.

Quadro 6 - Prioridade dos operadores do VisualG

OPERADOR	PRIORIDADE
Operadores aritméticos	3
Operadores relacionais	2
Operadores lógicos	1

Fonte: TONET; KOLIVER, s/d.

De acordo com a necessidade, as operações podem ser unidas pelos operadores lógicos.

Exemplos:

$(3 + 3)/3$ resulta em 2;

$3 + 3/3$ resulta em 4;

$(4 > 6)$ ou $(6 < 4)$ e $(4 < 6)$ resulta em FALSO

$(4 > 6)$ e $(6 < 4)$ ou $(4 < 6)$ resulta em VERDADEIRO

Observação 5: O software VisuAlg 2.5 não possui relacionamento de categorias. $3 * 6 > 4$ ou $6 + 2 < 3$ e $3 < 9 - 3$ resulta em erro. O correto seria: $(3 * 6 > 4)$ou$(6 + 2 < 3)$ e $(3 < 9 - 3)$.

3.3 Linearização de expressões

Para a construção de algoritmos que realizam cálculos matemáticos, todas as expressões aritméticas devem ser linearizadas, além de possuírem os operadores próprios do Português Estruturado. Utilizaremos somente parênteses "( )" para dividir uma expressão em partes. Essa prática é chamada de *modularização*. Na sintaxe do Português Estruturado,

podemos ter parênteses dentro de parênteses. Os parênteses indicam quais expressões secundárias, dentro de uma expressão maior, serão executadas primeiro.

Exemplo:

Tradicional: $\left\{\left[\frac{2}{3} - (5-3)\right] + 1\right\}.5$

Linear: $((2/3 - (5-3)) + 1) * 5$

3.4 Forma Geral de um Algoritmo

A estrutura de um algoritmo é:

Algoritmo "< nome do algoritmo >"
var
< declaração de variáveis >
inicio
< lista de comandos >
fimalgoritmo

onde as palavras *algoritmo* e *fimalgoritmo* fazem parte da sintaxe da linguagem e delimitam o início e o fim de um algoritmo; a <*declaração de variáveis*> é a seção onde descrevemos os tipos de dados que serão usados na lista de comando; *inicio* indica o fim das declarações de variáveis e o início da seção de comandos; <*lista de comandos*> é apenas uma indicação que entre as palavras *inicio* e *fimalgoritmo* podemos escrever uma lista com uma ou mais instruções ou comandos. As palavras que fazem parte da sintaxe da linguagem são chamadas *palavras reservadas*, ou seja, não podem ser usadas para outro propósito em um algoritmo que não seja aquele previsto nas regras da sintaxe. São elas:

Quadro 8 - Palavras reservadas do VisualG

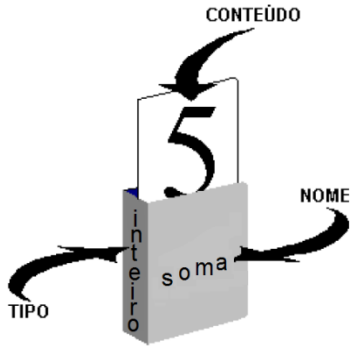
PALAVRAS RESERVADAS			
aleatorio	e	grauprad	passo
abs	eco	inicio	pausa
algoritmo	enquanto	int	pi
arccos	entao	interrompa	pos
arcsen	escolha	leia	procedimento
arctan	escreva	literal	quad
arquivo	exp	log	radpgrau
asc	faca	logico	raizq
ate	falso	logn	rand
caracter	fimalgoritmo	maiusc	randi
caso	fimenquanto	mensagem	repita
compr	fimescolha	minusc	se
copia	fimfuncao	nao	sen
cos	fimpara	numerico	senao
cotan	fimprocedimento	numpcarac	timer
cronometro	fimrepita	ou	tan
debug	fimse	outrocaso	verdadeiro
declare	função	para	xou

Fonte: TONET; KOLIVER, s/d.

3.5 Variáveis

Uma *variável* pode ser vista como uma caixa que armazena um *conteúdo* (que varia ao longo da execução do programa) de certo *tipo* e possui um *nome*.

Figura 1 - Noção intuitiva de uma variável



Fonte: TONET; KOLIVER, s/d.

A declaração de variáveis segue a seguinte estrutura:

var

$< nome\ 1 >, < nome\ 2 >, \ldots , < nome\ n > : < tipo >$

Em Português Estruturado temos quatro tipos de variáveis: *INTEIRO*; *REAL*; *LITERAL* ou *CARACTERE*; *LÓGICO (*que recebe apenas os valores VERDADEIRO ou FALSO).

Exemplo:

var

$k, soma, m, max, temp : real$

$i, j, n : inteiro$

Nesse exemplo, definimos as variáveis"$k, soma, m, max, temp$", cinco variáveis do mesmo tipo (número real), e as variáveis "i, j, n" do mesmo tipo (número inteiro).

3.6 Linhas de Comentário

Os comentários são declarações não copiladas que podem conter qualquer informação textual que queiramos adicionar ao código-fonte. Os comentários devem ser sucedidos por duas barras (//).

3.7 Operador de Atribuição

Serve para "colocar" um valor em uma variável dentro de um algoritmo. Ele é representado por uma seta apontando para a esquerda (na verdade é a justaposição do símbolo de menor que com o de subtração). É importante lembrar que só pode atribuir valores do mesmo tipo das variáveis.

Exemplo:

$peso < -85{,}9$ // $peso$ deve ser do tipo $real$

$nome < -Pedro\ Henrique$ // $nome$ deve ser do tipo $literal$

$localizado < -falso$ // $localizado$ deve ser do tipo $logico$

3.8 Comandos de Entrada e Saída

Para obter um resultado do usuário, o Português Estruturado usa o comando *leia* e, para exibir alguma expressão ou valor de uma variável, ele usa os comandos *escreva* ou *escreval* (após escrever o que for pedido ele passa para uma próxima linha).

Exemplo:

$escreval("\text{Digite o valor da variável k}")$//aparecerá na tela a frase entre (" ");

$leia(k)$// abaixo da linha anterior aparecerá um espaço para digitar o valor de k;

$k <- k + 1$ //o operador de atribuição implementará na variável k seu sucessor;

$escreva("o\ sucessor\ de\ k\ é", k)$ //além da frase será exibido o novo valor de k.

3.9 Estrutura Condicional

Expressões lógicas que podem ser respondidas apenas como "isso é *verdadeiro*" ou "isso é *falso*" podem ser utilizadas no Português Estruturado e são chamadas de *estruturas condicionais*. Vejamos a seguinte Estrutura Condicional:

se< condição a ser avaliada >$entao$

< ações a serem realizadas se a condição for verdadeira >

$senao$

< ações a serem realizadas se a condição for verdadeira >

$fimse$

3.10 Estrutura de Repetição

O conjunto de estruturas sintáxicas que permitem a repetição de uma lista de comandos do algoritmo é chamada de *Estruturas de Repetição*. Utilizaremos a seguinte Estrutura de Repetição:

$para$<variável de controle>de< valor inicial >ate< valor final >$faca$

< lista de comandos >

fimpara

onde a variável de controle será um número inteiro (senão teremos erro de sintaxe). Estruturas de Repetição podem causar um laço infinito se não forem bem utilizadas. Caso isso aconteça, o programa VisuAlg 2.5 irá travar e deveremos apertar Ctrl+Alt+Del e finalizar o programa.

3.11 Variáveis Compostas

Para utilizar diversas variáveis de um mesmo tipo, podemos usar o recurso chamado *variáveis indexadas* (ou *compostas)* que podem ser *unidimensionais(vetores)* ou *bidimensionais (matrizes)*. Essas variáveis indexadas correspondem a um identificador que difere apenas por um *índice.*

3.11.1 Variáveis Indexadas Unidimensionais (Vetores)

São variáveis referenciadas por um único índice. A sintaxe da declaração é:

< nome da variável>: *vetor* [<I..F>] *de*< tipo >.

Onde I significa *Valor Inicial* dos índices e F significa *Valor Final.*

3.11.2 Variáveis Indexadas Bidimensionais (Matrizes)

São variáveis referenciadas por dois índices. Podemos interpretar como se o primeiro índice representasse as linhas de uma matriz e o segundo índice representasse as colunas. A sintaxe da declaração é:

<nome da variável> :*vetor*[<I_1..F_1 , I_2..F_2>] *de* < tipo >

Onde I_1 significa *Valor Inicial do índice 1*, F_1 significa *Valor Final do índice 1*, I_2 significa *Valor Inicial do índice 2 e* F_2 *Valor Final do índice 2.*

Exemplo: Algoritmo que lê os termos de uma matriz e de um vetor e exibe o maior termo do vetor e seu índice.

algoritmo

var

```
x : vetor [1..5] de real//5 variáveis tipo real representadas por um vetor;
A : vetor [1..5,1..6] de real  // 30 variáveis tipo real representadas por
      //   uma matriz com 5 linhas e 6 colunas;
i,j,t: inteiro   //3 variáveis tipo inteiro que podem ser usadas como índices;
max: real//uma variável do tipo real;
inicio
Para i de 1 ate 5 faca//implementa valores na variável i de 1 até 5;
   Para j de 1 ate 6 faca//implementa valores na variável i de 1 até 6;
     Leia (A[i,j])// ler o termo da matriz de 1° índice i e 2° índice j;
   fimpara//finaliza o segundo para, ou seja, i continua igual a 1 enquanto
            //  j vai de 1 até 6;
 fimpara//finaliza o primeiro para, ou seja, após ler toda a linha 1 vai
 //iniciar  a ler a linha 2 e assim por diante até ler toda a linha 6;
 para i de 1 ate 5 faca//implementa valores na variável i de 1 até 5;
 leia(x[i]) //o mesmo i pode ser usado para ler o vetor x;
 fimpara
max < −x[1]
para i de 1 ate 5 faca
     se (x[i] > max) entao // identificar o termo de maior valor no vetor x ;
      max < − x[i]
      t < − i
     fimse
   fimpara
escreva("um dos maiores termos do vetor x é", max, "e seu índice é", t)
fimalgoritmo
```

3.12 Funções

São instrumentos que têm como objetivo retornar um valor ou uma informação. As funções podem ser predefinidas pela linguagem ou criadas pelo programador. O VisuAlg

2.5 vem com *bibliotecas* de funções que podem ser usadas nos programas, basta pressionar CTRL+J que aparece uma lista de funções na tela do VisuAlg 2.5.

Exemplo: Lê um vetor e exibe o termo de maior valor absoluto e seu índice.

```
algoritmo
var
x : vetor [1..5] de real
i, t: inteiro
max: real
inicio
max <- -abs(x[1])// a variável max recebe o valor absoluto de x[1]
para i de 1 ate 5 faca
se (abs(x[i]) > max) entao // identifica o maior valor absoluto no vetor x;
      max <- abs(x[i])
      t <- i
    fimse
  fimpara
escreval("o maior valor absoluto do vetor x é",max)
escreva(" e um de seus índices é", t)
fimalgoritmo
```

4 SISTEMAS LINEARES

4.1 Introdução

Os sistemas lineares trabalham com equações do tipo: $a_1x_1 + a_2x_2 + ... + a_nx_n = b$, onde a_i, com $i = 1, 2, ..., n$ são os coeficientes, x_i são as variáveis e b é o termo independente.

Em um sistema linear há n equações lineares com n incógnitas:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \qquad (1)$$

que na forma matricial pode ser escrito como $\boldsymbol{AX = B}$, onde $\boldsymbol{A}$ é a matriz dos coeficientes, $\boldsymbol{B}$ é o vetor dos termos independentes e $\boldsymbol{X}$ é o vetor das incógnitas.

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} B = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Para solucionar o sistema linear (1), deve-se encontrar um vetor $X' = \begin{bmatrix} x'_1 \\ \vdots \\ x'_n \end{bmatrix}$, chamado de vetor solução, tal que a igualdade $AX' = B$ seja satisfeita. O conjunto de todos os vetores X'encontrados é chamado de *conjunto solução do sistema.*

Definição 2

Dizemos que o sistema (1) é, respectivamente, *impossível*, *possível e determinado* ou *possível e indeterminado*, conforme o conjunto solução seja vazio, unitário ou possua pelo menos dois elementos.

Dos métodos de solução de sistemas ensinados no ensino médio regular no estado do Ceará, iremos nos concentrar no *método do escalonamento* que consiste em encontrar um *sistema equivalente* ao sistema dado usando *operações básicas* de tal forma que o sistema encontrado seja um *sistema escalonado*.

Definição 3

São chamadas *operações básicas* o conjunto de operações realizadas em um sistema de equações com o intuito de obter um novo sistema que possua o mesmo conjunto solução. As operações básicas são:

1. Trocar a posição de duas equações do sistema;
2. Multiplicar uma equação do sistema por uma constante não nula;
3. Substituir uma equação pela soma membro a membro dela com outra.

Observação 6: Podemos até realizar as operações básicas 2 e 3 simultaneamente, ou seja, substituir uma equação E_i de um sistema linear por $k.\mathrm{E}_i + l.\mathrm{E}_j$, onde ke l são constantes reais, k é não nula e E_j é outra equação do mesmo sistema linear.

Observação 7: É evidente que, aplicando a operação 1 em um sistema de equações, nós obteremos um novo sistema que possui o mesmo conjunto solução do sistema inicial. Quanto a segunda e terceira operações (mostraremos a observação 1 que é mais geral) temos que os sistemas

$$\begin{cases} E_1: a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ E_2: a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ E_i: a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i \\ \vdots \\ E_j: a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jn}x_n = b_j \\ \vdots \\ E_n: a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{2}$$

e

$$\begin{cases} E_1: a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ E_2: a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ E_i: (k.a_{i1} + l.a_{j1})x_1 + (k.a_{i2} + l.a_{j2})x_2 + \cdots + (k.a_{in} + l.a_{jn})x_n = k.b_i + l.b_j \\ \vdots \\ E_j: a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jn}x_n = b_j \\ \vdots \\ E_n: a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{3}$$

possuem o mesmo conjunto solução, pois se $X' = (x'_1, x'_2, \dots, x'_n)$ é solução de (2) então X' também é solução das equações E_t de (3) com $1 \leq t \leq n$ e $t \neq i$, $t, n,\ i$ inteiros positivos. Multiplicando a equação E_i de (2) por k e E_j por l e somando-as veremos que X' também satisfaz a equação E_i de (3). Reciprocamente se X' é solução do sistema (3) então também satisfaz as equações $E_t$de (2) com $1 \leq t \leq n$ e $t \neq i$, $t, n,\ i$ inteiros positivos. Multiplicando a equação E_j de (3) por l e substitui E_i por $E_i - l.\ E_j$. Agora multiplique a nova equação E_i por $1/k$ e veremos que X' também satisfaz a equação E_i de (2).

Definição 4

Dizemos que dois sistemas lineares são *equivalentes* se ambos tiverem o mesmo *conjunto solução*.

Exemplo: Os sistemas $\begin{cases} x_1 + x_2 = 3 \\ x_1 - x_2 = 1 \end{cases}$ e $\begin{cases} 2x_1 - x_2 = 3 \\ x_1 - 2x_2 = 0 \end{cases}$ são equivalentes, pois apresentam o conjunto $\{(2,1)\}$ como conjunto solução.

Definição 5

O sistema (1) é dito *escalonado* se o número de coeficientes nulos, antes do primeiro coeficiente não nulo, não diminui de equação para equação.

Exemplos:

$$\begin{cases} x_1 + x_2 + 3x_3 = 1 \\ x_2 - x_3 = 4 \\ 2x_3 = 5 \end{cases}$$

$$\begin{cases} 4x_1 + x_2 - x_3 + x_4 = 1 \\ x_2 - x_3 - x_4 = 0 \\ x_4 = 5 \\ x_4 = 7 \end{cases}$$

4.2 Métodos de Solução de Sistemas

4.2.1 Sistemas Escalonados

Iniciaremos com a técnica de encontrar a solução de sistemas possíveis e determinados que já estejam escalonados. Usaremos o método da *substituição retroativa*.

Exemplo: Resolver o sistema a seguir por substituição retroativa.

$$\begin{cases} x_1 + x_2 - 2x_3 = 0 & (i) \\ 2x_2 - x_3 = 4 & (ii) \\ 3x_3 = 6 & (iii) \end{cases}$$

Por(iii) é possível determinar o valor de x_3. Assim, temos:

$$3x_3 = 6 \Rightarrow x_3 = \frac{6}{3} \Rightarrow x_3 = 2$$

Este valor de x_3 é substituído em (ii) para a determinação de x_2:

$$2x_2 - x_3 = 4 \Rightarrow 2x_2 - 2 = 4 \Rightarrow 2x_2 = 6 \Rightarrow x_2 = \frac{6}{2} \Rightarrow x_2 = 3$$

Substituindo agora os valores de x_2 e x_3 em (i) chegaremos ao valor de x_1:

$$x_1 + x_2 - 2x_3 = 0 \Rightarrow x_1 + 3 - 2.2 = 0 \Rightarrow x_1 = 1$$

O vetor solução encontrado é:

$$X' = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

4.2.2 Algoritmo para Solucionar Sistemas Escalonados

Para realizar numericamente o procedimento para calcular o vetor solução de sistemas escalonados, utilizaremos a *matriz aumentada*$[AB]$. Ela consiste em uma matriz $n \times (n+1)$ onde a $(n+1)$-ésima coluna é o vetor dos termos independentes, enquanto os outros elementos são da própria matriz dos coeficientes.

$$[AB] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & & a_{2n} & b_2 \\ \vdots & & \ddots & \vdots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{bmatrix}$$

Exemplo: Dado o sistema

$$\begin{cases} 2x_1 - 2x_2 - x_3 = -7 \\ 4x_1 - 4x_2 + x_3 = -5 \\ x_1 + 3x_2 - 2x_3 = 4 \end{cases}$$

A matriz aumentada do sistema é:

$$[AB] = \begin{bmatrix} 2 & -2 & -1 & -7 \\ 4 & -4 & 1 & -5 \\ 1 & 3 & -2 & 4 \end{bmatrix}$$

Para solucionar um sistema linear, devem-se seguir os procedimentos do seguinte algoritmo:

$X[n] < -A\ [n, n+1]/A[n,n]$ // calculando o valor de $X[n]$

$para\ r\ de\ 1\ ate\ (n-1)\ faca$

$soma < -0$

$s < -(n-r)$ // variável que vai ajudar no método de substituição retroativa.

$para\ j\ de\ (s+1)\ ate\ n\ faca$

$soma < -soma + A[s,j] * X[j]$ // calculando o somatório do produto entre os coeficientes // e os valores das variáveis abaixo da linha *s* (já descobertos)

$fimpara$

$X[s] < -(A[s, n+1] - soma)/A[s,s]$ // calculando o valor de $X[s]$

$fimpara$

Como dados de entrada, é necessária a matriz aumentada dos coeficientes A com os termos independentes e a ordem de A_n.

4.2.3 Escalonando Sistemas

Usaremos o Método da Eliminação Gaussiana com escolha do *elemento pivô* conveniente. De forma simples, esse método consiste em transformar um sistema linear da forma (1) em um sistema escalonado para o qual possa ser usado o método 2.1.

Exemplo: Dado o sistema

$$\begin{cases} 2x_1 - 2x_2 - x_3 = -7 \\ 4x_1 - 4x_2 + x_3 = -5 \\ x_1 + 3x_2 - 2x_3 = 4 \end{cases}$$

aplicar o Método da Eliminação Gaussiana para encontrar um sistema escalonado equivalente.

Solução: Como vimos sua matriz aumentada é:

$$[AB] = \begin{bmatrix} 2 & -2 & -1 & -7 \\ 4 & -4 & 1 & -5 \\ 1 & 3 & -2 & 4 \end{bmatrix}$$

O primeiro passo é determinar o elemento de maior módulo da *primeira coluna*. Neste caso, este elemento é $a_{21} = 4$. Utilizaremos este elemento como o pivô para proceder

com a Eliminação Gaussiana, ou seja, faremos com o valor 4 esteja na posição do elemento a_{11}. Assim, realizaremos uma permutação entre a primeira e a segunda linha da matriz aumentada. O resultado é:

$$[AB] = \begin{bmatrix} 4 & -4 & 1 & -5 \\ 2 & -2 & -1 & -7 \\ 1 & 3 & -2 & 4 \end{bmatrix}$$

Escolhendo o elemento a_{11}da nova matriz $[AB]$ como primeiro pivô, encontraremos agora o multiplicado que fará com que os elementos da *coluna* 1 que estejam abaixo do elemento a_{11} sejam anulados. Iniciaremos anulando o elemento da *linha 2* logo o multiplicador será nomeado como m_{21} e seu valor é dado por :

$$m_{21} = \frac{a_{21}}{a_{11}} \Rightarrow m_{21} = \frac{2}{4} \Rightarrow m_{21} = 0{,}5$$

Para alterar os elementos da segunda linha, zerando assim o elemento a_{21}, executamos $\mathrm{L}_2 \leftarrow \mathrm{L}_2 - m_{21}.\mathrm{L}_1$, assim a matriz aumentada fica igual a:

$$[AB] = \begin{bmatrix} 4 & -4 & 1 & -5 \\ 0 & 0 & -1{,}5 & -4{,}5 \\ 1 & 3 & -2 & 4 \end{bmatrix}$$

Passaremos agora para a eliminação do elemento a_{31}. Encontrando o multiplicador correspondente, temos:

$$m_{31} = \frac{a_{31}}{a_{11}} \Rightarrow m_{31} = \frac{1}{4} \Rightarrow m_{31} = 0{,}25$$

A operação sobre a terceira linha é $\mathrm{L}_3 \leftarrow \mathrm{L}_3 - m_{31}.\mathrm{L}_1$, resultando em:

$$[AB] = \begin{bmatrix} 4 & -4 & 1 & -5 \\ 0 & 0 & -1{,}5 & -4{,}5 \\ 0 & 4 & -2{,}25 & 5{,}25 \end{bmatrix}$$

O passo agora é encontrar o elemento de maior módulo na *segunda coluna*. Neste caso, não é possível escolher o elemento $a_{12} = -4$,pois a regra geral que se aplica é a busca pelo elemento de maior módulo, que vai ser o pivô no processo de Eliminação Gaussiana, é realizada da diagonal principal para baixo. Nesse exemplo específico devemos realizar a busca entre os elementos $a_{22} = 0$ e $a_{32} = 4$. Sendo a_{32} com o maior módulo, ele será escolhido como pivô. Permutaremos então as linhas 2 e 3 a fim de colocar o pivô na diagonal principal. Nossa matriz aumentada fica na seguinte forma:

$$[AB] = \begin{bmatrix} 4 & -4 & 1 & -5 \\ 0 & 4 & -2{,}25 & 5{,}25 \\ 0 & 0 & -1{,}5 & -4{,}5 \end{bmatrix}$$

Como nesse caso o termo a_{32} é nulo não há necessidade de encontrarmos o multiplicador

$$m_{32} = \frac{a_{32}}{a_{22}}$$

muito menos de fazer a operação $L_3 \leftarrow L_3 - m_{32}.L_2$, pois já temos o nosso sistema na forma escalonada. Para encontrar o vetor solução X' basta aplicar o método 2.1. A solução é:

$$X' = \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix}$$

4.2.4 Algoritmo para Escalonar Sistemas

Para escalonar um sistema linear, devemos seguir o seguinte algoritmo:

para s de 1 ate (n − 1) faca

max < − abs (a[s, s])

t < −s

para i de (s + 1)ate n faca//identificando o maior valor absoluto na coluna *s*

se abs (a[i,s]) > max entao

max < − abs(a[i, s])

t < − i

fimse

fimpara

se (t <> s) entao //verificando se é necessário permutar linhas da matriz aumentada

para j de s ate (n + 1) faca //colocando o pivô na diagonal principal

temp < −a[s, j]

a[s, j] < −a[t, j]

a[t, j] < − temp

fimpara

fimse

se a[s, s] <> 0 entao // verificando se há valores nulos na diagonal principal

para r de (s + 1) ate n faca

m < − a[r,s]/a[s, s] //calculando o multiplicador da *linha r coluna s*

$para\ c\ de\ s\ ate\ (n+1) faca$ // aplicando a operação $L_r \leftarrow L_r - m_{rs}.L_s$

$a[r,c] < -a[r,c] - m * a[s,c]$

$fimpara$

$fimpara$

$senao$

$escreva("Divisão\ por\ zero")$

$fimse$

$fimpara$

5 CONSIDERAÇÕES FINAIS

A partir do que foi exposto nos capítulos anteriores, recomendamos ao profissional que for trabalhar com uma abordagem computacional em sala de aula que siga a ordem inversa do que foi apresentado nesse material. Inicialmente, pode-se trabalhar com sistemas lineares pequenos (duas equações e duas variáveis) e aumentando gradativamente a complexidade dos problemas. Um problema interessante que pode ser abordado para motivar o estudo de sistemas (além dos problemas clássicos dos livros didáticos) é o problema do *balanceamento de uma equação química*, baseado na lei de conservação das massas de Lavoisier, segundo a qual "em um sistema químico isolado, a massa permanece constante, quaisquer que sejam as transformações que nele se processam" ou de forma mais simples: "em uma reação química, a soma das massas dos reagentes é igual à soma das massas dos produtos resultantes". Por exemplo:

(ENEM 2012) "No Japão, um movimento nacional para a promoção da luta contra o aquecimento global leva o slogan: 1 pessoa, 1 dia, 1 kg de CO_2 *a menos! A ideia é cada pessoa reduzir em 1 kg a quantidade de* CO_2 *emitida todo dia, por meio de pequenos gestos ecológicos, como diminuir a queima de gás de cozinha"*

Um hamburguer ecológico? É pra já! *Disponível em: http://lqes.iqm.unicamp.br. Acesso em: 24 fev. 2012 (adaptado).*

Considerando um processo de combustão completa de um gás de cozinha composto exclusivamente por butano (C_4H_{10}*), a mínima quantidade desse gás que um japonês deve deixar de queimar para atender à meta diária, apenas com esse gesto, é de Dados:* CO_2 *(44 g/mol);* C_4H_{10} *(58 g/mol)*

A) 0,25 kg. B) 0,33 kg. C) 1,0 kg. D) 1,3 kg. E) 3,0 kg.

Para resolver esse problema um aluno deve se lembrar de que as reações de combustão necessitam de um comburente, nesse caso possivelmente será o oxigênio (O_2) e libera, além de gás carbônico (CO_2), vapor de água (H_2O). Podemos representar a reação de combustão do butano pela equação:

$$C_4H_{10} + O_2 \rightarrow CO_2 + H_2O$$

a qual não está balanceada. Podemos atribuir coeficientes literais x_i às substâncias que aparecem na equação, aplicar a lei de Lavoisier e comparar os elementos membro a membro,

construindo assim um sistema de equações algébricas lineares onde as incógnitas são os coeficientes estequiométricos x_i da reação química. Fazendo isso temos:

$x_1C_4H_{10} + x_2O_2 \rightarrow x_3CO_2 + x_4H_2O$

C :$4.x_1 = x_3$

H : $10.x_1 = 2.x_4$

O :$2.x_2 = 2.x_3 + x_4$

que é equivalente ao sistema:

$$\begin{cases} 4.x_1 + 0.x_2 - 1.x_3 + 0.x_4 = 0 \\ 10.x_1 + 0.x_2 + 0.x_3 - 2.x_4 = 0 \\ 0.x_1 + 2.x_2 - 2.x_3 - x_4 = 0 \end{cases}$$

Esse sistema possui apenas 3 equações porém possui 4 incógnitas. Atribuindo um valor a uma das variáveis (x_1 = 1, por exemplo) temos agora o seguinte sistema:

$$\begin{cases} 0.x_2 - 1.x_3 + 0.x_4 = -4 \\ 0.x_2 + 0.x_3 - 2.x_4 = -10 \\ 2.x_2 - 2.x_3 - x_4 = 0 \end{cases}$$

Pronto, conseguimos modelar um problema envolvendo balanceamento de equação química transformando-o num problema que consiste em encontrar a solução de um sistema de equações. Esse sistema é um sistema simples que apresenta

$$\begin{cases} 2.x_2 - 2.x_3 + 1.x_4 = 0 \\ 0.x_2 - 1.x_3 + 0.x_4 = -4 \\ 0.x_2 + 0.x_3 - 2.x_4 = -10 \end{cases}$$

como sistema equivalente obtido apenas através da permutação entre suas equações. Usando o método de substituição retroativa e acrescentando o valor x_1 = 1, obtemos o vetor $(1, \frac{13}{2}, 4, 5)$ como solução e como coeficientes estequiométricos. A equação balanceada fica:

$$C_4H_{10} + \frac{13}{2}O_2 \rightarrow 4CO_2 + 5H_2O$$

Para concluir o problema proposto, basta verificar que para a reação acontecer gastaremos 58g de butano e obteremos 4.44 = 176g de gás carbônico e que para atingir a meta de 1kg de gás carbônico a menos por dia seriam necessários 329,54g de butano (basta

verificar que a razão entre suas quantidades será constante) que equivale a 0,32954 kg ou mais simplificadamente 0,33 kg.

A princípio trata-se de um problema simples que não precisaria de métodos computacionais para solucioná-lo, porém ele pode servir como problema motivador para balanceamento de equações mais extensas como:

$$K_2Cr_2O_7 + FeSO_4 + HCl \rightarrow KCl + CrCl_3 + FeCl_3 + Fe_2(SO_4)_3 + H_2O$$

Após apresentado os métodos de solução de problemas o profissional pode introduzir a parte computacional. Introduziria inicialmente os conceitos básicos como *algoritmo, variável e operadores.* Posteriormente iniciaria uma abordagem sobre os *comandos de entrada e saída, estruturas de condição e de repetição, variáveis indexadas e funções.*

Ao falar sobre variáveis indexadas o profissional pode relembrar conceitos envolvendo matrizes e suas operações além de representar um sistema de equações lineares por meio de uma equação matricial, pois serão úteis quando for elaborar o programa para solucionar sistemas.

Com isso o profissional tem todas as ferramentas para solucionar os mais diversos tipos de sistemas, inclusive o que se origina da equação apresentada.

Utilizando o mesmo raciocínio teremos os coeficientes estequiométricos da equação química como incógnitas e um sistema de equações envolvendo tais coeficientes. Observe:

$$x_1K_2Cr_2O_7 + x_2FeSO_4 + x_3HCl \rightarrow x_4KCl + x_5CrCl_3 + x_6FeCl_3 + x_7Fe_2(SO_4)_3 + x_8H_2O$$

K: $2.x_1 = x_4$

Cr: $2.x_1 = x_5$

O: $7.x_1 + 4.x_2 = 12.x_7 + x_8$

Fe:$x_2 = x_6 + 2.x_7$

S:$x_2 = 3.x_7$

H: $x_3 = 2.x_8$

Cl:$x_3 = x_4 + 3.x_5 + 3.x_6$

que é equivalente ao sistema:

$$\begin{cases} 2.x_1 - x_4 = 0 \\ 2.x_1 - x_5 = 0 \\ 7.x_1 + 4.x_2 - 12.x_7 - x_8 = 0 \\ x_2 - x_6 - 2.x_7 = 0 \\ x_2 - 3.x_7 = 0 \\ x_3 - 2.x_8 = 0 \\ x_3 - x_4 - 3.x_5 - 3.x_6 = 0 \end{cases}$$

Um sistema com 7 equações e 8 variáveis. O processo será o mesmo que o aplicado anteriormente, porém será mais trabalhoso.

Os algoritmos computacionais vistos podem e vão facilitar a solução desse sistema.

Atribuindo um valor a uma das variáveis (x_1 = 1, por exemplo) obtemos por meio do algoritmo o seguinte sistema:

$$\begin{cases} 4.x_2 - 12.x_7 - x_8 = -7 \\ x_3 - 2.x_8 = 0 \\ -x_4 = -2 \\ -3.x_5 - 3.x_6 + 2.x_8 = 2 \\ x_6 - \left(\frac{2}{3}\right).x_8 = -\frac{8}{3} \\ x_7 - 0{,}417.x_8 = -0{,}917 \\ 0{,}25.x_8 = 1{,}75 \end{cases}$$

como sistema equivalente. Usando o método de substituição retroativa e acrescentando o valor x_1 = 1 obtemos o vetor $(1{,}6, 14, 2, 2, 2, 2{,}7)$ como solução e como coeficientes estequiométricos. A equação balanceada fica:

$$K_2Cr_2O_7 + 6FeSO_4 + 14HCl \rightarrow 2KCl + 2CrCl_3 + 2FeCl_3 + 2Fe_2(SO_4)_3 + 7H_2O$$

REFERÊNCIAS

AZEVEDO FILHO, Manoel Ferreira de. **Geometria analítica e álgebra linear**. 2 ed. Fortaleza: Edições Livro Técnico, 2003.

CAMPOS FILHO, Frederico Ferreira. **Algoritmos numéricos**. 2 ed. Rio de Janeiro: LTC, 2010.

IEZZI, Gelson. *et al*. **Fundamentos de matemática elementar** :sequências, matrizes, determinantes e sistemas lineares. 2 ed. São Paulo: Atual, 1977. v. 4, p. 35-64, 115-158.

LIMA, Elon Lages. *et al*. **A Matemática do ensino médio.** 6 ed. Rio de Janeiro: SBM, 2006. v. 3, p. 193-238.

LIMA, Elon Lages. **Geometria analítica e álgebra linear**. 2 ed. Rio de Janeiro: IMPA, 2011. p. 193-238.

PEDROSA, Diogo Pinheiro Fernandes. **Sistemas lineares**. UFRN-CT-DCA. Disponível em: < http://www.dca.ufrn.br/~diogo/FTP/dca0304/sistemaslineares.pdf> Acesso em: mar. 2013.

TONET, Bruno; KOLIVER, Cristian. **Introdução aos algoritmos.** UCS-NAPRO. Disponível em:<http://pt.scribd.com/doc/52626280/UCS-NAPRO-Visualg2-IntroducaoAlgoritmos> Acesso em: mar. 2013.

APÊNDICE

PROGRAMA COMPLETO PELO VisualG

```
algoritmo "Solução de sistemas"

// Autor : Hugo

// Data : 17/02/2013

// Seção de Declarações

var

A : vetor [1..15,1..16] de real

x : vetor [1..15] de real

k, soma, m, max, temp  : real

i, j, r, s, n, c, t  : inteiro

inicio

k <- 1

repita

Escreval("Esse programa soluciona sistemas de n equações com n variáveis (máximo n = 15 e
INTEIRO.")

escreval ()

Escreval ("Inicialmente indique quantas equações. Saiba que a quantidade de variáveis")

escreval ("será a mesma quantidade de equações")

leia (n)

se ((n > 15) ou (n < 1)) entao

escreval ("Valor inválido. Vamos tentar novamente. Leia atentamente as instruções do
programa")

escreval()

fimse
```

```
ate ((n < 16) e (n > 0))


repita
 k <- 0
Escreval("Digite os coeficientes do seu sistema de equações (Ao digitar um")
escreval ("coeficiente aperte enter. Escreva primeiro os coeficientes da primeira linha,")
escreval ("depois os da segunda linha e assim por diante")
escreval ("para termos um sistema da seguinte forma:")
escreval ()
escreval ("   a11   a12   ...   a1n          x1")
escreval ("   a21   a22   ...   a2n          x2")
escreval ("   a31   a32   ...   a3n          x3")
escreval ("    .     .    ...    .      *      . ")
escreval ("    .     .    ...    .           . ")
escreval ("    .     .    ...    .           . ")
escreval ("   an1   an2   ...   annxn")
escreval ("Após digitar os coeficientes, digite os valores de b1, b2, ...,bn ")
escreval ("respectivamente de modo que tenhamos o seguinte sistema:")
escreval ()
escreval ("   a11   a12   ...   a1n          x1           b1")
escreval ("   a21   a22   ...   a2n          x2           b2")
escreval ("   a31   a32   ...   a3n          x3           b3")
escreval ("    .     .    ...    .      *      .       =     . ")
escreval ("    .     .    ...    .           .            . ")
escreval ("    .     .    ...    .           .            . ")
escreval ("   an1   an2   ...   annxnbn")
```

```
 Para i de 1 ate n faca
    Para j de 1 ate n faca
       Leia (a[i,j])
fimpara
fimpara
para i de 1 ate n faca
leia(a[i,n+1])
fimpara
escreval()
escreval("o sistema de equações na forma matricial fica:")
escreval()

para i de 1 ate n faca
para j de 1 ate n faca
se a[i,j] < 0 entao
escreva(a[i,j]:4:3,"   ")
senao
escreva(" ",a[i,j]:4:3,"   ")
fimse
fimpara
escreva ("          x",i,"          ",a[i,n+1]:4:3)
escreval()
fimpara
escreval()
escreval("  É realmente esse o sistema que desejas resolver? Se quiser reescrever o")
escreval("sistema completamente digite 1 e aperte ENTER")
```

```
escreval("senão digite outro número qualquer")

leia(k)
escreval()
ate k <> 1

para s de 1 ate (n-1) faca
max<- abs(a[s,s])
   t <- s
para i de (s+1) ate n faca  //identificar termo de maior valor absoluto na coluna s
se (abs(a[i,s]) >max) entao
max<- abs(a[i,s])
        t <- i
fimse
fimpara
se (t <> s) entao     //verificando se é necessário permutar linhas da matriz aumentada
para j de s ate (n+1) faca   //colocando o pivô na diagonal principal
temp<- a[s,j]
a[s,j] <- a[t,j]
a[t,j] <- temp
fimpara
fimse
se (a[s,s] <> 0) entao
para r de (s+1) ate n faca
          m <- a[r,s]/a[s,s]
//calculando o multiplicador da linha r coluna s (posteriormente chamado de m[r,s])
para c de s ate (n+1) faca
```

```
// aplicando a operação Linha r <- Linha r - m[r,s]*Linha s
a[r,c] <- a[r,c] - m*a[s,c]
fimpara
fimpara
senao
escreval(" VAI DAR ERRO!!! Haverá uma divisão por zero se continuar.")

fimse
fimpara

escreval()
escreval("o sistema de equações pedido possui o seguinte sistema triangular equivalente:")
escreval()

para i de 1 ate n faca
para j de 1 ate n faca
escreva(a[i,j]:4:3,"   ")
fimpara
escreva ("          x",i,"          ",a[i,n+1]:4:3)
escreval()
fimpara

escreval ("Para continuar digite um número e aperte ENTER!")
leia (k)
temp<- 1

para i de 1 ate n faca
```

```
para j de 1 ate n faca
se i = j entao
se a[i,j] = 0 entao
temp<- 0
fimse
fimse
fimpara
fimpara

setemp = 0 entao

escreval ("O sistema não possui solução ou possui infinitas soluções")

senao

 x[n] <- a[n,n+1]/a[n,n]          // calculando o valor de X[n]
para r de 1 ate (n - 1) faca
soma<- 0
   s <- n - r        // variável que vai ajudar no método de substituição retroativa.
para j de (s+1) ate n faca
soma<- soma + a[s,j]*x[j]     // calculando o somatório do produto entre os coeficientes e os valores das variáveis abaixo da linha s (já descobertos)
fimpara
 x[s] <- (a[s,n+1] - soma)/ A[s,s]      // calculando o valor de X[s]
fimpara

escreval(" A solução do sistema é :")
```

```
para i de 1 ate n faca
se x[i]<>0 entao
escreval("O valor de x",i," é ", x[i]:4:3)
senao
escreval("O valor de x",i," é ", abs(x[i]:4:3))
fimse
fimpara
fimse
escreva("Foi um prazer ajudar. Bons estudos")
fimalgoritmo
```