UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

SOLUÇÃO EXATA DE SISTEMAS DE EQUAÇÕES LINEARES DETERMINADOS UTILIZANDO A ARITMÉTICA RESIDUAL

Salete Maria Chalub Bandeira

Campina Grande - PB
1998

Salete Maria Chalub Bandeira

SOLUÇÃO EXATA DE SISTEMAS DE EQUAÇÕES LINEARES DETERMINADOS UTILIZANDO A ARITMÉTICA RESIDUAL

Dissertação submetida ao Curso de Pós-Graduação do Centro de Ciências e Tecnologia da Universidade Federal da Paraíba, como requisito parcial para a obtenção do grau de Mestre em Informática.
Área de Concentração: Ciência da Computação
Sub-Área: Matemática Computacional.

Orientadores: Mário Toyotaro Hattori (*In memorian*)
Bruno Correia da Nóbrega Queiroz

Campina Grande - PB
1998

519.6 Bandeira, Salete Maria Chalub.
B 214 Solução Exata de Sistemas de Equações Lineares Determinados utilizando a Aritmética Residual. Campina Grande: UFPB, setembro de 1998.
118p.

Dissertação (Mestrado) UFPB – COPIN

1. Matemática Computacional
2. Sistemas Lineares
3. Aritmética Residual

- CDU -

SOLUÇÃO EXATA DE SISTEMAS DE EQUAÇÕES LINEARES DETERMINADOS UTILIZANDO A ARITMÉTICA RESIDUAL

SALETE MARIA CHALUB BANDEIRA

DISSERTAÇÃO APROVADA EM 29.09.1998

PROF. MÁRIO TOYOTARO HATTORI, M.Sc
Orientador

PROF. BRUNO CORREIA DA NÓBREGA QUEIROZ, M.Sc
Co-orientador

PROF. JOÃO MARQUES DE CARVALHO, Ph.D
Examinador

PROF. MAURO CAVALCANTE PEQUENO, DS.c
- Examinador -

CAMPINA GRANDE - PB

"O Binômio de Newton é tão belo como
a Vênus de Milo
O que há é pouca gente para dar por isso".
(...)

Fernando Pessoa

Ao meu orientador In memorian, **Mário Toyotaro Hattori** *- presença sempre viva, desde a idéia inicial deste estudo, quando a pesquisa se constituia num rascunho, até a concretização da mesma - , Engenheiro Eletrônico egresso do ITA e mestre em ciência da computação pela University of Toronto, Ontario, Canadá; dono de um currículo brilhante, teve importante participação na criação do curso de ciência da computação da Universidade Federal da Paraíba – campus II – na cidade de Campina Grande. Homem simples, de perspicácia indefinível, demonstrou valentia mesmo quando seu corpo físico perdia as energias vitais, sua alma se mantinha forte, mesmo assim, enquanto esteve entre nós, fez questão de conduzir-me a todos os passos deste trabalho. Com ele, conheci a partir da disciplina Análise Numérica Aplicada, como estudar os métodos numéricos, erros em computação, objetivos do software matemático e as primeiras noções de programação em Fortran; aprendi ainda, a importância de sistematizar nossas experiências.*

Sua partida, em 08 de dezembro de 1997, deixou entre nós a saudade e a lembrança; mas nos deixou também sua rica experiência, seu grande exemplo de profissional e de cidadão.

Ao meu querido sobrinho, **Thiago Chalub Bandeira Teixeira** – *ao olhar para o céu, vejo através das estrelas a imagem de seu sorriso que permanecerá por toda a vida, a eterna e doce lembrança em nossos corações.*

" O amor é a poesia dos sentidos.
Quando existe, existe para todo o
sempre e aumenta cada vez mais."

AGRADECIMENTOS

À ***Deus****, Espírito de amor, paz e alegria, que me acompanha em todos os momentos da minha vida, me iluminando e me ajudando, e a cada dia me fazendo compreender e sentir o seu amor, dando-me aconchego, força e coragem nos momentos mais difíceis.*

Aos meus pais, Mercedes e Aldo Bandeira - pela dedicação, o amor, carinho, e especialmente, pela educação que me foi dada - , por seu incentivo e apoio incondicional que sempre me dispensaram rumo à concretização de minhas aspirações.

As minhas irmãs e companheiras, Solange e Simone, presentes em todos os momentos da minha vida, que apesar da distância, sempre me estimularam e acreditaram no meu sucesso.

A toda minha família: avós, tios, primos, cunhados e sobrinhos pelo carinho.

Ao meu Co-orientador, Prof. Ms. Bruno Correia da Nóbrega Queiroz, cujo carinho, atenção, dedicação, paciência e sobretudo, a solidariedade, tornaram possíveis a continuidade e concretização deste trabalho.

À Universidade Federal do Acre, através da Pró-Reitoria de Pós-Graduação e ao Departamento de Matemática e Estatística, pela colaboração e apoio.

Aos Professores e Funcionários do Curso de Pós-Graduação em Informática, da Universidade Federal da Paraíba - Campus II, em especial Prof. Edilson e Aninha (secretária), pela calorosa acolhida ao mestrado.

Aos colegas do mestrado, em especial: Malty, Silvino, Augusto, Cenidalva, Rute, Cassandra, Patrícia, Edjozane, Adeilton, Jean e Vera, pela vivência coletiva nas várias fazes do curso, e pelo carinho e apoio recebidos.

À Prof. Ms. Kíssia Carvalho pela colaboração e valiosas críticas à primeira versão deste trabalho.

Aos Professores, Dr. Mauro Cavalcante Pequeno - Universidade Federal do Ceará e Dr. João Marques - Universidade Federal da Paraíba, ambos, membros da comissão examinadora, que se dispuseram a colaborar conosco com suas valiosas avaliações.

À Família Santiago, pela calorosa acolhida, pelo apoio e carinho prestados ao longo deste curso.

Aos amigos, Lígia e Ruy Lino, pela colaboração prestada e pelo carinhoso incentivo.

À Prof. Ms. Francisca C. S. da Silva, por todo o incentivo, atenção e colaboração.

À Família Pessoa, em especial Cristiane, pelo carinho e dedicação atribuída ao final deste trabalho.

À Ana Rita e Idelfonso, pelo carinho, atenção e preocupação com o meu bem estar.

À amiga Aglaize Damasceno, por todo o apoio, atenção e carinho.

À Walter, pela competente digitação deste trabalho em latex.

À Alécio, pela contribuição em sua experiência com Fortran.

A todos que de certo modo deram alguma contribuição a este trabalho e cujos nomes foram omitidos na lista de agradecimentos.

RESUMO

O presente estudo, "***Solução Exata de Sistemas de Equações Lineares Determinados utilizando a Aritmética Residual***", consiste na aplicação da Aritmética Residual na Solução de Sistemas de Equações Lineares, cujos dados são números exatos (inteiros) e sabendo-se que existe uma solução exata (inteira), deseja-se encontrá-la. Para isto, estudamos implementações existentes e implementamos experimentalmente algumas variantes dos algoritmos encontrados nas referências. Fez-se um estudo comparativo entre os métodos diretos convencionais, tais como: Eliminação de Gauss e Gauss – Jordan e os algoritmos publicados pela ACM (Association for Computing Machinery), constituídos em dois estudo de casos: 1. Algoritmo 406 – *EXACT* [HOWELL, 1971] – Solução Exata de Equações Lineares Usando Aritmética Residual; 2. Algoritmo 522 – *ESOLVE* [CABAY e LAM, 1977] – Técnicas de Congruência para uma Solução Exata de Sistemas de Equações Lineares Inteiros. Finalizando, são apresentados os resultados obtidos.

ABSTRACT

The present study, "***Exact Solution of Systems Determined Linear Equations Using Residual Arithmetic*** ", consists of the application of residual arithmetic to the solution of systems of linear equations, whose data correspond to exact number (integers). Knowing that an exact solution exists, it was desirable to find it. In order to do this, various existent implementations were studied and some variants of the algorithms found in references were experimentally implemented. A comparative study was carried out using direct conventional methods likes: Elimination of Gauss and Gauss-Jordan and the algorithms published by ACM (Association for Computing Machinery), was carried out, constituted of two case studies: 1. *Algorithm 406*, *EXACT* [HOWELL, 1971]: Exact Solution of Linear Equations Using Residual Arithmetic; 2. *Algorithm 522*, *ESOLVE* [CABAY and LAM, 1977]: Congruence Techniques for the Exact Solution of Integer Systems of Linear Equations. Finally, the results obtained are presented.

SUMÁRIO

LISTA DE TABELAS

Capítulo 1

Introdução

Os problemas de matemática surgem ao se tentar explicar fenômenos da natureza e prever o seu comportamento futuro, procurando uma relação entre causa e efeito. Busca-se primeiro uma explicação para um fenômeno por meio de um modelo que estabeleça uma relação entre causas e efeitos, modelo esse que dá origem aos problemas de matemática como, por exemplo, sistemas de equações lineares.

Para encontrar a solução de um dado problema de matemática, existem vários métodos [HATTORI, 1992]:

1. *Método Dedutivo*: quando se quer provar alguma verdade matemática.

2. *Método Analítico*: procura-se obter uma relação entre as variáveis independentes e variáveis dependentes satisfazendo alguma(s) condição(ões).

3. *Método Numérico*: procura-se obter valores solução (variável dependente) de um problema para valores discretos das variáveis independentes, quando a solução

multiplicação, o tempo necessário para resolver um sistema 20×20 será

$$t = 0,5 \times 10^{13} \quad segundos.$$

Como um dia tem 86.400 segundos, então o tempo gasto em anos será de 158.548 anos. Para um sistema de 15 equações lineares ($n = 15$), requer 16! operações de multiplicação. Supondo que cada operação leve 100 nanosegundos, o tempo gasto será de 24 dias ou 0,066 anos.

A regra de Cramer é um exemplo de que a simples utilização de uma implementação direta de um método matemático no computador pode ser inviável. Com isso, verifica-se que nem sempre os métodos analíticos têm suas implementações viáveis. Busca-se então, novas maneiras de resolver sistemas lineares, surgindo assim a necessidade de desenvolver métodos numéricos.

Segundo, dado um problema e dispondo de um programa para resolvê-lo, basta aprender a usar o programa e fornecer os dados do problema. Na verdade, não é tão simples assim. Em computação numérica [1]não basta apenas saber utilizar um programa, é necessário verificar se os resultados obtidos são válidos e se o problema em si não apresenta dificuldades.

Exemplo 1.1 *[HATTORI, 1992]: O sistema linear*

$$37639840x - 46099201y = 0$$

$$29180479x - 35738642y = -1$$

[1]Computação Numérica: *É a resolução numérica de problemas de matemática utilizando o computador.* □

foi resolvido por um programa que sabidamente é de boa qualidade. Usando um microcomputador compatível com IBM PC, a resposta encontrada foi

$$x = -42587641,592475$$

e

$$y = -34772663,750032.$$

Embora a solução correta seja

$$x = 46099201$$

e

$$y = 37639840.$$

Será que o programa contém erros? Na verdade, o que tem de excepcional é o problema e não erros no programa. No caso, o sitema linear é quase indeterminado, levando a uma "solução" que tem pouca confiabilidade. □

O *exemplo 1.1* ilustra a necessidade de analisar os resultados computados e o problema. O sistema acima é dito mal-condicionado.

1.1 Terminologia

O termo *software matemático (mathematical software)* foi cunhado por John Rice ao convidar pesquisadores para participarem de um simpósio que haveria na

Pardue University em abril de 1970 [RICE, 1969]. Nesse convite Rice definiu um software matemático como sendo programas de computador que implementam procedimentos matemáticos amplamente aplicáveis. Nos anais do simpósio publicados em 1971 [RICE, 1971], Rice definiu o software matemático como sendo um conjunto de algoritmos na área de matemática. Essas duas definições ilustram a confusão inicial que se fazia entre programas de computador e algoritmos. A plena compreensão de que uma implementação é diferente do algoritmo básico marca o reconhecimento do software matemático como uma nova disciplina (assunto de ensino e pesquisa). Portanto, ao contrário da idéia de produto que Rice deu a entender nos primórdios da história do software matemático, hoje ele é considerado uma disciplina, uma atividade que envolve a implementação dos métodos computacionais, se preocupa com padrões de qualidade, com a melhoria da produtividade do processo de desenvolvimento de software de métodos numéricos e com o reconhecimento do hardware e do software do computador em que os programas são executados.

A terminologia usada é introduzida a seguir:

Definição 1.1.1 *[HATTORI, 1992] : Software matemático é um ramo da engenharia de software que concebe, implementa, testa e mantém software para solução de problemas de matemática.* □

Definição 1.1.2 *[HATTORI, 1992] : Software numérico é um software cuja finalidade é a solução numérica de problemas de matemática.* □

Um software combina diversos algoritmos para que cumpra suas finalidades.

Definição 1.1.3 *[HATTORI, 1992] : Algoritmo básico é aquele definido matematicamente.* □

Definição 1.1.4 *[HATTORI, 1992] : Algoritmo computacional (numérico, algébrico, ou gráfico) é aquele obtido de um algoritmo básico ou de uma combinação deles, levando em conta as características do computador com a finalidade de executar uma sequência de operações para resolver um problema.* □

Um software matemático pode se apresentar como:

- Um pacote, que é uma coleção de programas para resolver os problemas de uma área, por exemplo, sistemas de equações lineares.
- Uma biblioteca, uma coleção de programas para resolver diversas classes de problemas de matemática.
- Um sistema de software, constituído de um pacote ou uma biblioteca com uma interface de comunicação com o usuário.

1.2 Motivação

Os sistemas de equações lineares surgem em praticamente todas as áreas da física, biologia, engenharia, álgebra linear e ciências sociais. Como também, nas soluções numéricas de problemas em equações diferenciais ordinárias, equações diferenciais parciais, equações inteiras, problemas de otimização e de vários outros problemas que precisam da solução de sistemas de equações lineares.

Existem duas formas de resolver sistemas de equações lineares: por métodos diretos e por métodos iterativos.

Definição 1.2.1 : *Diz-se que o método é direto quando é possível chegar a uma solução exata após um número finito de operações aritméticas.* □

Todo método direto implica em algum procedimento de eliminação. A compreensão dos métodos diretos, Eliminação de Gauss e similares, foi possível graças aos trabalhos de J. Wilkinson [WILKINSON, 1963] no início da década de 60. A Eliminação de Gauss está definitivamente consagrada, graças aos trabalhos de Wilkinson e ao sucesso dos métodos dos elementos finitos que a utiliza para resolver sistemas de equações lineares contendo milhares de equações.

Definição 1.2.2 : *Diz-se que o método é iterativo quando partindo de uma aproximação inicial é possível se chegar a aproximações mais precisas que dependem, sempre, de valores anteriormente calculados.* □

Os métodos iterativos foram populares na década de 50 e uma das classes desses métodos foi plenamente analisada por David Young, cujo livro [YOUNG, 1971] é um tratado sobre o assunto. Tais métodos são atraentes para a solução de sistemas de equações lineares que resultam da discretização de equações diferenciais parciais pelo método das diferenças finitas, porque não destroem a estrutura da matriz dos coeficientes.

Na década de 70 houve uma renovação do interesse em métodos iterativos, graças a redescoberta, por John Reid em 1971, do método dos gradientes conjugados inventado por Hestenes e Stiefel em 1950 e a descoberta de métodos para acelerar

a convergência (pré-condicionamento). Contudo, deve-se chamar a atenção para o fato que não há regras rígidas para escolher um método direto ou iterativo [RALSTON, 1965]. Na verdade, existem controvérsias quanto a essas regras.

Nesta dissertação pretende-se analisar problemas de matemática, em particular solução de sistemas de equações lineares, em que a atenção especial será dada a uma classe de problema:

1. Sistemas de equações lineares determinados: encontrar um vetor $\mathbf{x}$ que satisfaça $A\mathbf{x} = \mathbf{b}$, em que A é uma matriz $n \times n$, não singular, $\mathbf{x}$ e $\mathbf{b}$ são vetores colunas com n componentes.

Resolvendo o problema (1) pelo método direto, conhecido como Eliminação de Gauss, se os dados são reais, surgem os seguintes problemas:

- A solução obtida é apenas aproximada porque a execução de operações aritméticas reais em ponto flutuante está sujeita a erros de arrendondamento.
- O problema do erro de arredondamento se agrava quando o problema é mal condicionado, ou seja, o problema é muito sensível a perturbações.

Para superar os problemas quando trabalhamos com aritmética real, existem algumas soluções:

- Usar aritmética real de múltipla precisão;
- Usar aritmética inteira (disponível no computador ou estendido) para evitar geração de erro de arredondamento.

Vamos analisar problemas de matemática em que os dados são números exatos (inteiros) e sabendo-se que existe uma solução exata (inteira), deseja-se encontrá-la. Como a solução para esse problema envolve números inteiros, seria natural usar a aritmética inteira disponível no computador ou aritmética inteira estendida, para evitar geração de erro de arredondamento. Porém, encontramos os seguintes problemas:

1. Usando a aritmética inteira disponível no computador, o limite superior é muito baixo (o maior inteiro representável em inteiro no computador é muito pequeno) e podemos nos deparar com o problema de *transbordamento* (*overflow*) que ocorre quando o resultado de uma operação for maior que o maior número representável como inteiro no computador.

2. Usando aritmética inteira estendida, expandimos o limite superior por software, isto é, simulamos o tamanho da palavra. Porém, ganhamos em termos de precisão no resultado, mas perdemos em eficiência (eficiência baixa), ou seja, aumenta o tempo de processamento.

Uma solução possível para superarmos o problema de transbordamento (overflow) descrito em 1, é usar a aritmética residual, que trabalha com inteiros escalados para o intervalo $[0, m-1]$, em que m é o módulo usado. Em qualquer caso, temos a necessidade de pequenas modificações na Eliminação de Gauss, que será mostrado no capítulo 3.

A utilização da aritmética residual apresenta duas dificuldades:

1. O resíduo de uma operação de divisão não é bem definido;

2. Tendo o resultado de uma operação em aritmética residual usando um módulo, não é possível recuperar univocamente o resultado da mesma operação em aritmética inteira com os dados originais.

Para superar a dificuldade 1 evita-se a execução da operação de divisão trabalhando no campo dos racionais ou algoritmicamente evitando a operação de divisão.

Exemplo 1.2 *: Sistema linear de duas equações*

$$a_{11}x_1 + a_{12}x_2 = b_1,$$

$$a_{21}x_1 + a_{22}x_2 = b_2.$$

Com divisão: a primeira equação do sistema permanece inalterada. Para eliminar x_1 *da* 2^a *equação, multiplicamos a* 1^a *equação por* a_{21}/a_{11}*, e subtraímos da* 2^a *equação obtendo,*

$$a_{11}x_1 + a_{12}x_2 = b_1,$$

$$(a_{22} - a_{21}/a_{11} \times a_{12})x_2 = b_2 - a_{21}/a_{11} \times b_1.$$

Sem divisão: a primeira equação do sistema permanece inalterada. Para eliminar x_1 *da* 2^a *equação, subtraímos da* 2^a *multiplicada por* a_{11}*, a* 1^a *multiplicada por* a_{21} *obtendo,*

$$a_{11}x_1 + a_{12}x_2 = b_1,$$

$$(a_{22}a_{11} - a_{12}a_{21})x_2 = a_{11}b_2 - a_{21}b_1. \quad \square$$

A dificuldade descrita em 2, no caso geral nem sempre é possível computar $d = det\, A$ e $\mathbf{y} = A^{adj} \times \mathbf{b}$ em ordem de selecionar um módulo m que satisfaça $(m,\, d) = 1$ e $m > 2\max(|d|, \max_i |y_i|)$. Para garantir a unicidade, trabalhamos com a aritmética dos resíduos usando mais de um módulo M, de modo que $M = m_1 m_2 m_3 ... m_r$ e

$$M \geq 2 \prod_{i=1}^{n} \left(\sum_{k=1}^{n} a_{ik}^2 \right)^{\frac{1}{2}} \sum_{j=1}^{n} |b_j|.$$

1.3 Objetivos da Dissertação

Os objetivos deste trabalho são:

- Aplicar a aritmética residual na solução exata de sistemas de equações lineares;
- Estudar implementações existentes, bem como, implementar experimentalmente algumas variantes dos algoritmos encontrados nas referências;
- E por fim, comparar os resultados obtidos com os resultados obtidos pelos métodos diretos: Eliminação de Gauss e Eliminação de Gauss-Jordan.

1.4 Organização da Dissertação

A princípio pretendeu-se estudar duas classes de problemas:

1. Sistemas de equações lineares determinados;

2. Sistema de equações lineares superdeterminados: encontrar um vetor $\mathbf{x}$ que satisfaça $A\mathbf{x} = \mathbf{b}$, em que A é uma matriz $m \times n$, com n colunas linearmente independentes, $m > n$, $\mathbf{x}$ é um vetor coluna com n componentes e $\mathbf{b}$ é um vetor de m componentes. Esse problema é equivalente a resolver $A^TA\mathbf{x} = A^T\mathbf{b}$, em que A^T é a transposta da matriz A [2]. Como A^TA é $n \times n$ e não singular porque as colunas de A são linearmente independentes e $A^T\mathbf{b}$ é um vetor coluna de n componentes, o problema pode ser transformado no problema (1).

Contudo, com o decorrer da pesquisa, optou-se pelo seguinte tema: *Solução Exata de Sistemas de Equações Lineares Determinados Utilizando a Aritmética Residual.*

Acreditando poder posteriormente dar continuidade ao trabalho pretendido inicialmente, o que certamente me apontaria outras exigências profissionais.

A dissertação foi assim organizada:

⋆ *Capítulo 2* - Fundamentos Matemáticos:

- Métodos diretos: Eliminação de Gauss, Eliminação de Gauss-Jordan com seus respectivos algoritmos, bem como a comparação entre estes métodos;
- Definições Básicas (Aritmética Residual): algumas definições e teoremas básicos para aplicação da aritmética residual tradicional e aritmética residual estendida à la Howell;

[2] A transposta da matriz A, $m \times n$ denotada por A^T é uma matriz $n \times m$ cujos elementos a_{ij}^T satisfazem a condição $a_{ij}^T = a_{ji}$, $i = 1, \cdots, m$ e $j = 1, \cdots, n$. □

- Conversão da representação residual para representação residual simétrica e conversão da representação residual simétrica para representação simétrica de base mista;

- Teorema Chinês do Resto.

⋆ *Capítulo 3* - Algoritmos modificados para a aplicação da aritmética residual:

- Eliminação de Gauss e Gauss-Jordan utilizando aritmética residual e seus algoritmos.

⋆ *Capítulo 4* - Estudo de Casos - "Detalhes de Implementação" São estudados dois algoritmos publicados pela ACM (Association for Computing Machinery).

1. Estudo de Caso 1: Algoritmo 406 - EXACT [HOWELL, 1971] - Solução Exata de Equações Lineares usando Aritmética Residual.

2. Estudo de Caso 2: Algoritmo 522 - ESOLVE [CABAY e LAM, 1977] - Técnicas de Congruência para uma Solução Exata de Sistemas de Equações Lineares Inteiros.

Apresenta uma descrição das matrizes dos coeficientes, vetor independente dos sistemas usados em testes, os resultados dos testes realizados e uma avaliação dos algoritmos.

⋆ *Capítulo 5* - Considerações finais, trabalho futuro.

Capítulo 2

Fundamentos Matemáticos

Neste capítulo apresentamos os fundamentos necessários para o entendimento dos algoritmos que serão descritos no capítulo 3.

2.1 Métodos Diretos

2.1.1 Eliminação de Gauss

Já conhecido da Análise Numérica, o método de Eliminação de Gauss, triangulariza a matriz A dos coeficientes, ou seja, reduz a matriz A a uma matriz triangular superior ($a_{ij} = 0$, para todo $i > j$), conforme o processo a seguir:

Consideremos um sistema de n equações lineares e n incógnitas, do tipo:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & \cdots & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array} \tag{2.1}$$

Podemos escrever o sistema (2.1) numa forma matricial:

$$A\mathbf{x} = \mathbf{b}$$

em que,

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

é a matriz dos coeficientes,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix}$$

é a matriz das incógnitas e

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_n \end{bmatrix}$$

é a matriz dos termos independentes.

Uma matriz que podemos associar ao sistema $A\mathrm{x} = \mathbf{b}$, é a matriz ampliada, denotada por $[A \mid \mathbf{b}]$ na qual

$$[A \mid \mathbf{b}] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right], \tag{2.2}$$

em que, $b_1 = a_{1(n+1)}$, $b_2 = a_{2(n+1)}$, $\cdots$, $b_n = a_{n(n+1)}$.

O primeiro passo é a eliminação de todos os elementos da primeira coluna de $[A \mid \mathbf{b}]$, exceto a_{11}. Vamos supor o pivô $a_{11} \neq 0$. Definimos os multiplicadores por

$$m_{i1} = \frac{a_{i1}}{a_{11}}$$

para $i = 2, 3, ..., n$.

Multiplicamos a 1^a linha por m_{i1}, e subtraímos da i-ésima linha, para $i = 2, 3, ..., n$, obtendo,

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22} - m_{21}a_{12} & \cdots & a_{2n} - m_{21}a_{1n} & b_2 - m_{21}b_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2} - m_{n1}a_{12} & \cdots & a_{nn} - m_{n1}a_{1n} & b_n - m_{n1}b_1 \end{bmatrix}$$

Reescrevemos a nova matriz como:

$$[A \mid \mathbf{b}]^{(1)} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix}, \tag{2.3}$$

em que (1) indica que uma eliminação foi realizada com

$$a_{ij}^{(1)} = a_{ij} - m_{i1}a_{1j} \text{ para } i, j = 2, 3, ..., n$$

e

$$b_i^{(1)} = b_i - m_{i1}b_1 \text{ para } i = 2, 3, ..., n.$$

Assumindo o pivô $a_{22}^{(1)} \neq 0$. No segundo passo, eliminamos todos os elementos da 2^a coluna de $[A \mid \mathbf{b}]^{(1)}$ abaixo de $a_{22}^{(1)}$. Definimos $m_{i2} = a_{i2}^{(1)}/a_{22}^{(1)}$ para $i = 3, 4, \cdots, n$. Multiplicamos a 2^a linha por m_{i2}, e subtraímos da i-ésima linha, para

$i = 3, 4, ..., n$, obtendo:

$$[A \mid \mathbf{b}]^{(2)} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{bmatrix}, \tag{2.4}$$

em que,

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i2}a_{2j}^{(1)} \text{ para } i, j = 3, 4, ..., n$$

e

$$b_i^{(2)} = b_i^{(1)} - m_{i2}b_2^{(1)} \text{ para } i = 3, 4, ..., n.$$

Repetindo este procedimento $(n-1)$ vezes, obtemos:

$$[A \mid \mathbf{b}]^{(n-1)} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} & b_1 \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & ... & a_{2n}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & ... & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & ... & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{bmatrix}, \tag{2.5}$$

no qual na i-ésima etapa os pivôs $a_{ii}^{(i-1)} \neq 0$, para $i = 1, 2, ..., n-1$, em que;

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - m_{ik}a_{kj}^{(k-1)},$$

$$b_i^{(k)} = b_i^{(k-1)} - m_{ik}b_k^{(k-1)},$$

$$m_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}},$$

para $k = 1, 2, ..., n-1$ e $i, j = k+1, k+2, ..., n$, e

$a_{ij}^{(0)} = a_{ij}$ para $i, j = 1, 2, ..., n$ e $b_i^{(0)} = b_i$ para $i = 1, 2, ..., n$.

Da equação (2.5),

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}},$$

se $a_{nn}^{(n-1)} \neq 0$ e

$$x_i = \frac{1}{a_{ii}^{(i-1)}} \left[b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j \right],$$

para $i = n-1, n-2, ..., 1$.

Eliminação Sem Divisão:

$$a_{ij}^{(k)} = a_{kk}^{(k-1)} a_{ij}^{(k-1)} - a_{kj}^{(k-1)} a_{ik}^{(k-1)},$$

$$b_i^{(k)} = a_{kk}^{(k-1)} b_i^{(k-1)} - b_k^{(k-1)} a_{ik}^{(k-1)}$$

para $k = 1, 2, ..., n-1$, e $i, j = k+1, k+2, ..., n$, e $a_{ij}^{(0)} = a_{ij}$ para $i, j = 1, 2, ..., n$ e $b_i^{(0)} = b_i$ para $i = 1, 2, ..., n$.

O sistema linear $A^{(n-1)} \cdot \mathbf{x} = \mathbf{b}^{(n-1)}$ (2.5) é equivalente [3] ao sistema linear original, $A\mathbf{x} = \mathbf{b}$ (2.2).

O método de Eliminação de Gauss descrito anteriormente, pode ser implementado utilizando o algoritmo 1.

[3] Dois sistemas lineares, $A\mathbf{x} = \mathbf{b}$ e $A'\mathbf{x} = \mathbf{b}'$, são equivalentes se qualquer solução de um é também solução do outro. □

Algoritmo 1:

1. **for** $k = 1$ **until** $n - 1$ **do**
2. **for** $i = k + 1$ **until** n **do**
3. $m_{ik} := a_{ik}/a_{kk}$
4. **for** $j = k + 1$ **until** $n + 1$ **do**
5. $a_{ij} := a_{ij} - m_{ik}a_{kj}$
6. **endfor**
7. **endfor**
8. **endfor** □

Após a transformação da matriz A em uma matriz triangular superior obtemos o vetor $\mathbf{x}$ a partir da equação (2.5).

Complexidade

Analizando a eficiência do método de Eliminação de Gauss, verificamos que:

No 1^o Passo:

- Quando multiplicamos a 1^a linha por $m_{21} = a_{21}/a_{11}$ e subtraímos desta linha a 2^a linha. Para isto, necessitamos de uma divisão, para obtermos m_{21}, e n multiplicações para obtermos $m_{21}a_{12}, m_{21}a_{13}, ..., m_{21}a_{1n}$ e $m_{21}b_1$. Subtraindo este produto da 2^a linha, isto exige n adições/subtrações. Desde então, o 1^o elemento da 2^a linha será zero. Isto requer uma divisão, n multiplicações e n adições / subtrações para obter a 2^a linha da equação (2.3) da equação (2.2).

- Para obtermos a 3^a, 4^a, ..., n-ésima linha da equação (2.3) de (2.2), precisamos de uma divisão, n multiplicações e n adições/subtrações para cada linha. Assim, necessitamos de $n-1$ divisões, $(n-1)n$ multiplicações, e $(n-1)n$ adições/subtrações para reduzirmos (2.2) a (2.3).

- Similarmente precisamos de $(n-2)$ divisões, $(n-2)(n-1)$ multiplicações, e $(n-2)(n-1)$ adições/subtrações para reduzirmos (2.3) a (2.4).

- Continuando este procedimento para reduzirmos a equação (2.2) a (2.5), realizamos $(n-1)+(n-2)+...+1$ divisões, $(n-1).n+(n-2).(n-1)+...+1.2$ multiplicações e $(n-1).n+(n-2).(n-1)+...+1.2$ adições/subtrações. Precisamos do mesmo número de multiplicações e adições/subtrações para reduzir (2.2) a (2.5). Para avaliarmos o número total de multiplicações, reescrevemo-as:

$$\begin{aligned} 1\cdot 2+2\cdot 3+...+(n-1)\cdot n &= \sum_{i=1}^{n}(i-1)\cdot i \\ &= \sum_{i=1}^{n} i^2-\sum_{i=1}^{n} i \\ &= \frac{n(n-1)(n+1)}{3}. \end{aligned}$$

Para reduzirmos a equação (2.2) a (2.5), necessitamos de $n(n-1)(n+1)/3$ multiplicações e $n(n-1)(n+1)/3$ adições/subtrações. O número total de divisões é dado por:

$$1+2+...+(n-1)=\sum_{i=1}^{n-1} i=\frac{n(n-1)}{2}.$$

Da equação (2.5), $a_{nn}^{(n-1)}x_n=b_n^{(n-1)}$. Portanto, para o processo completo, precisamos de:

$$\frac{n(n+1)}{2} \text{ divisões,}$$

$$\frac{n(n-1)(n+1)}{3} + \frac{n(n-1)}{2} = \frac{n(n-1)(2n+5)}{6} \text{ multiplicações e}$$

$$\frac{n(n-1)(2n+5)}{6} \text{ adições/subtrações.}$$

2.1.2 Eliminação de Gauss-Jordan

O método de Eliminação de Gauss-Jordan, diagonaliza A ($a_{ij} = 0$, para todo $i \neq j$), isto é, apenas os elementos da diagonal são não nulos. Caso não queiramos fazer nenhuma substituição, reduzimos os elementos da diagonal à unidade (transformamos A em uma matriz identidade).

Seja A uma matriz $n \times n$, não singular, e $\mathbf{b}$ um vetor não nulo de n componentes. Procuramos uma matriz J, $n \times n$, não singular, em que:

$$JA\mathbf{x} = J\mathbf{b}, \tag{2.6}$$

com $JA = I$. Assim,

$$J = A^{-1} \tag{2.7}$$

e

$$\mathbf{x} = J\mathbf{b} \tag{2.8}$$

é a solução.

Em Gauss-Jordan, assumimos que $a_{22}^{(1)} \neq 0$ em (2.3). Eliminamos todos os elementos da 2^a coluna de $[A \mid \mathbf{b}]^{(1)}$ em (2.3) exceto o elemento da diagonal $a_{22}^{(1)}$,

no qual $m_{i2} = a_{i2}^{(1)}/a_{22}^{(1)}$ para $i = 1, 3, 4, ..., n$, em que $a_{12}^{(1)} = a_{12}$ e subtraindo m_{i2} multiplicado pela 2^a linha da i-ésima linha, para $i = 1, 3, 4, ..., n$, reduzimos (2.3) a

$$\begin{bmatrix} a_{11} & 0 & a_{13}^{(2)} & ... & a_{1n}^{(2)} & b_1^{(2)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & ... & a_{2n}^{(1)} & b_2^{(1)} \\ 0 & 0 & a_{33}^{(2)} & ... & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & a_{n3}^{(2)} & ... & a_{nn}^{(2)} & b_n^{(2)} \end{bmatrix}. \tag{2.9}$$

Os elementos da 1^a linha são dados por:

$$a_{1j}^{(2)} = a_{1j} - \left(\frac{a_{12}}{a_{22}^{(1)}}\right) a_{2j}^{(1)} \quad \text{para} \quad j = 3, 4, ..., n$$

e

$$b_1^{(2)} = b_1 - \left(\frac{a_{12}}{a_{22}^{(1)}}\right) b_2^{(1)}.$$

A idéia é reduzir todos os elementos das colunas para zero, exceto os elementos da diagonal. Repetindo este procedimento, obtemos:

$$\begin{bmatrix} a_{11} & 0 & 0 & ... & 0 & b_1^{(n-1)} \\ 0 & a_{22}^{(1)} & 0 & ... & 0 & b_2^{(n-1)} \\ 0 & 0 & a_{33}^{(2)} & ... & 0 & b_3^{(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & ... & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{bmatrix}, \tag{2.10}$$

em que,

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - m_{ik} a_{kj}^{(k-1)},$$

$$b_i^{(k)} = b_i^{(k-1)} - m_{ik} b_k^{(k-1)},$$

$$m_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}},$$

para $k = 1, 2, ..., n-1$,

$i, j = 1, 2, ..., k-1, k+1, ..., n$, e

$a_{ij}^{(0)} = a_{ij}$, para $i, j = 1, 2, ..., n$ e

$b_i^{(0)} = b_i$, para $i = 1, 2, ..., n$.

A solução da equação (2.10) é dada por:

$$x_i = \frac{b_i^{(i-1)}}{a_{ii}^{(i-1)}} \tag{2.11}$$

para $i = 1, 2, ..., n$, se $a_{ii}^{(i-1)} \neq 0$.

Para reduzirmos A de (2.10) à matriz identidade, multiplicamos a 1^a linha, 2^a linha, ..., i-ésima linha por:

$$m_{ii} = \frac{1}{a_{ii}^{(i-1)}},$$

em que $a_{ii}^{(i-1)} \neq 0$, para $i = 1, 2, ..., n$ e obtemos:

$$\begin{bmatrix} 1 & 0 & 0 & ... & 0 & b_1^{(n-1)}.m_{11} \\ 0 & 1 & 0 & ... & 0 & b_2^{(n-1)}.m_{22} \\ 0 & 0 & 1 & ... & 0 & b_3^{(n-1)}.m_{33} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & ... & 1 & b_n^{(n-1)}.m_{nn} \end{bmatrix} \tag{2.12}$$

em que,

$$x_i = b_i^{(i-1)} m_{ii}, \quad \text{para} \quad i = 1, 2, ..., n.$$

O método de Eliminação de Gauss-Jordan, descrito anteriormente, pode ser implementado utilizando o algoritmo 2.

Algoritmo 2:

1. **for** $k = 1$ **until** n **do**
2. **for** $i = 1$ **until** n **do**
3. $m_{ik} := a_{ik}/a_{kk}$
4. **for** $j = 1$ **until** $n + 1$ **do**
5. $a_{ij} := a_{ij} - m_{ik} a_{kj}$
6. **endfor**
7. **endfor**
8. **endfor**□

Complexidade

Para resolvermos um sistema de n equações lineares e n incógnitas pela eliminação de Gauss-Jordan, precisamos de:

$$n^2 \quad \text{divisões,}$$

$$\frac{n(n^2-1)}{2} \quad \text{multiplicações e}$$

$$\frac{n(n^2-1)}{2} \quad \text{adições/subtrações.}$$

2.1.3 Comparação dos Métodos

A Tabela 2.1 ilustra o número total de operações de divisões, multiplicações e adições / subtrações requeridas para os métodos da Eliminação de Gauss e Gauss-Jordan, por comparação de diferentes n.

	Eliminação de Gauss			**Eliminação de Gauss-Jordan**		
n	Divisões	Multiplicações	Adições/ Subtrações	Divisões	Multiplicações	Adições/ Subtrações
3	6	11	11	9	12	12
5	15	50	50	25	60	60
10	55	375	375	100	495	495
100	5050	338250	338250	10000	499950	499950

Tabela 2.1: Comparação entre os métodos diretos.

Observando a Tabela 2.1 acima, fica evidente que o método de Eliminação de Gauss-Jordan requer mais operações que o método de Eliminação de Gauss. Diante disto, a Eliminação de Gauss é a mais recomendada, uma vez que para o n grande, o número de operações é proporcional a $2n^3/3$ e para Gauss-Jordan, n^3. Maiores detalhes, ver [PATEL, 1994].

A implementação do método de Eliminação de Gauss encontra-se no *LINPACK* em [DONGARRA, 1979] e Gauss-Jordan em [LUDOVICE, 1998].

2.1.4 Estratégias de Pivoteamento

Vimos que o algoritmo para o método de Eliminação de Gauss e Gauss-Jordan requer o cálculo dos multiplicadores m_{ik}, em cada etapa k do processo. Perguntaremos então:

O que acontece se o pivô for nulo? E se o pivô estiver próximo de zero? Primeiramente, é impossível trabalhar com um pivô nulo. E trabalhar com um pivô próximo de zero pode conduzir a resultados totalmente imprecisos. Isto porque em qualquer calculadora ou computador os cálculos são efetuados com aritmética de precisão finita, e pivôs próximos de zero dão origem a multiplicadores bem maiores que a unidade que, por sua vez, origina uma aplicação dos erros de arredondamento.

Para se contornar estes problemas devemos adotar uma estratégia de pivoteamento, ou seja, adotar um processo de escolha da linha e/ou coluna pivotal.

1. **Estratégia de Pivoteamento Parcial.**

   Esta estratégia consiste em:

   (a) no início da etapa k da fase de eliminação, escolher para pivô o elemento de maior módulo entre os coeficientes: $a_{ik}^{(k-1)}$, $i = k, k+1, \cdots, n$;

   (b) trocar as linhas k e i se for necessário.

   Vejamos um exemplo a seguir:

$n = 4$ e $k = 2$

$$[A|\mathbf{b}]^{(1)} = \left(\begin{array}{cccc|c} 3 & 2 & -1 & 1 & 5 \\ 0 & 1 & 0 & 3 & 6 \\ 0 & -3 & -4 & 7 & 7 \\ 0 & 2 & 3 & 0 & 13 \end{array}\right)$$

Início da etapa 2:

(a) escolher pivô

$$\max_{j=2,3,4} |a_{j2}^{(1)}| = |a_{32}^{(1)}| = 3 \Rightarrow \text{pivô} = -3$$

(b) trocar linhas 2 e 3.

Assim,

$$[A|\mathbf{b}]^{(1)} = \left(\begin{array}{cccc|c} 3 & 2 & -1 & 1 & 5 \\ 0 & -3 & -4 & 7 & 7 \\ 0 & 1 & 0 & 3 & 6 \\ 0 & 2 & 3 & 0 & 13 \end{array}\right)$$

e os multiplicadores desta etapa são:

$$m_{32} = \frac{1}{-3} = -1/3,$$
$$m_{42} = \frac{2}{-3} = -2/3.$$

Observamos que a escolha do maior elemento em módulo entre os candidatos a pivô faz com que os multiplicadores, em módulo, estejam entre zero e um, o que evita a ampliação dos erros de arredondamento.

2. **Estratégia de Pivoteamento Completo.**

   Nesta estratégia, no início da etapa k é escolhido para pivô o elemento de maior módulo, entre todos os elementos que ainda atuam no processo de eliminação:

$$\max_{\forall i,j \geq k} |a_{ij}^{(k-1)}| = |a_{rs}^{(k-1)}| \Rightarrow \text{pivô} = a_{rs}^{(k-1)}.$$

   Adotando esta estratégia para o exemplo anterior, o pivô da etapa 2 seria $a_{34}^{(1)} = 7$, o que acarretaria a troca das colunas 2 e 4, em seguida, das linhas 2 e 3, donde:

$$[A|\mathbf{b}]^{(1)} = \left(\begin{array}{cccc|c} 3 & 1 & -1 & 2 & 5 \\ 0 & 7 & -4 & -3 & 7 \\ 0 & 3 & 0 & 1 & 6 \\ 0 & 0 & 3 & 2 & 13 \end{array}\right)$$

Esta estratégia não é muito empregada, pois envolve uma comparação extensa entre os elementos $a_{ij}^{(k-1)}$, $i, j \geq k$ e troca de linhas e colunas, conforme vimos no exemplo anterior; é evidente que todo este processo acarreta em esforço computacional maior que a estratégia de pivoteamento parcial. Maiores detalhes ver: [RUGGIERO & LOPES, 1996] e [HATTORI, 1992].

2.2 Definições Básicas (Aritmética Residual)

Definição 2.2.1 *[HOWELL, 1971]: Dado qualquer inteiro x e qualquer móulo inteiro m, se*

1. $r \equiv x \,(\mathrm{mod} m)$,

2. $| r | < m$ *e*

3. $sgn\,(r) = sgn\,(x)$, *então escrevemos* $r = | x |_m$ [4] *e dizemos que r é um resíduo de x módulo m.* □

Exemplo 2.1 : *Consideremos x = -12, m = 7 e vamos encontrar r.*

De acordo com a definição 2.2.1, temos:

1. $r \equiv$ *-12* $(\mathrm{mod}\, 7)$,

2. $| r | < 7$ *e*

3. $sgn(r) = sgn(-12)$, *então* $r = | -12 |_7$ = *-5 e dizemos que -5 é um resíduo de -12 módulo 7.* □

Exemplo 2.2 : *Consideremos r = 12, m = 7 e vamos encontrar r.*

Análogo ao exemplo 2.1, temos:

1. $r \equiv 12(\mathrm{mod}7)$,

2. $| r | < 7$ *e*

3. $sgn(r) = sgn(12)$, *então* $r = | 12 |_7 = 5$ *e dizemos que 5 é um resíduo de 12 módulo 7.* □

---

[4] as barras | | denotam resíduo módulo *m*.

Mostramos que estes resíduos são únicos, observando a tabela a seguir:

⋮	⋮	⋮	⋮	⋮	⋮	⋮	
-21	-20	-19	-18	-17	-16	-15	
-14	-13	-12	-11	-10	-9	-8	
-7	$\mathbf{-6}$	-5	$\mathbf{-4}$	$\mathbf{-3}$	$\mathbf{-2}$	$\mathbf{-1}$	resíduos
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
7	8	9	10	11	12	13	
14	15	16	17	18	19	20	
21	22	23	24	25	26	27	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{3}$	$\mathbf{4}$	5	$\mathbf{6}$	resíduos $(m = 7)$.

Vejamos agora, uma outra forma de definir resíduo módulo m:

Definição 2.2.2 *[HOWELL & GREGORY, 1969]: Dados quaisquer inteiros x e $m > 0$, se $r \equiv x(\mathrm{mod}\, m)$ e se $0 \leq r < m$, então escrevemos $r = \mid x \mid_m$ e dizemos que r é um resíduo de x módulo m.* □

Voltando ao *exemplo 2.1*, usando a definição 2.2.2, notamos que:

Como $m = 7 > 0$, se $r \equiv -12(\mathrm{mod}7)$ e se $0 \leq r < 7$, então escrevemos $r = \mid -12 \mid_7 = 2$.

De forma semelhante, no *exemplo 2.2*, se $r \equiv 12(\mathrm{mod}7)$ e se $0 \leq r < 7$, então escrevemos $r = \mid 12 \mid_7 = 5$, que de fato difere da definição 2.2.1.

⋮	⋮	⋮	⋮	⋮	⋮	⋮	
−21	−20	−19	−18	−17	−16	−15	
−14	−13	−12	−11	−10	−9	−8	
−7	−6	−5	−4	−3	−2	−1	
0	**1**	*2*	**3**	**4**	*5*	**6**	resíduos $m = 7$
7	8	9	10	11	12	13	
14	15	16	17	18	19	20	
21	22	23	24	25	26	27	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Todos inteiros em uma coluna constituem uma *classe residual.*

A definição 2.2.1 é usada no Estudo do Caso 1[5], no capítulo 4.

Definição 2.2.3 *[HOWELL & GREGORY, 1969] : Chamamos de Sistemas de Números Residuais, um sistema em que mais de um módulo é usado.*□

Definição 2.2.4 *[YOUNG & GREGORY, 1972] : Uma representação residual de um inteiro x, para as bases* m_1, m_2, m_3, ..., m_s *é a s-tupla* $x \sim \{| x |_{m_1}, | x |_{m_2}, ..., | x |_{m_s}\}$.

□

Definição 2.2.5 *[YOUNG & GREGORY, 1972] : Seja A uma matriz inteira* $n \times n$, *então a matriz adjunta de A módulo m,* $| A^{adj} |_m$, *é definida como sendo* $| A^{adj} |_m =$

[5]Estudo de Caso 1: Algoritmo 406 - *EXACT* - Solução Exata de Equações Lineares usando a Aritmética Residual.

$| A_{ji} |_m$, *em que* A_{ij} *é o cofator de* a_{ij}. *(*A_{ij} *é a transposta da matriz dos cofatores).* □

Definição 2.2.6 *[YOUNG & GREGORY, 1972] : Sejam A e C matrizes inteiras* $n \times n$ *e um inteiro* $m > 1$, *se:*

1. $| AC |_m = I = | CA |_m$
2. $| C |_m = C$, *então escrevemos* $C = A^{-1}(\mathrm{mod}\, m)$ *e chamamos C um inverso multiplicativo de A módulo m.* □

Definição 2.2.7 *[YOUNG & GREGORY, 1972] : Uma matriz inteira A,* $n \times n$, *é não singular módulo m, se e somente se,*

1. $|d\,^6|_m \neq 0$,
2. $(d,\ m) = 1$.

Caso contrário, A é singular módulo m. □

Teorema 2.2.1 *[YOUNG & GREGORY, 1972] :* $A^{-1}(\mathrm{mod}\, m)$ *existe se e somente se, A é não singular módulo m, isto é, se e somente se,*

1. $| d |_m \neq 0$,
2. $(d, m) = 1$. *Neste caso,*

---

[6] $d = \det$ A.

$$A^{-1}(\mathrm{mod}m) = \mid d^{-1}(\mathrm{mod}m) \mid A^{adj} \mid_m \mid_m . \ \square$$

Teorema 2.2.2 *[YOUNG & GREGORY, 1972] : Se*

1. *A é uma matriz inteira,* $n \times n$,

2. *A é não singular módulo* m,

3. $\mathbf{b}$ *é um vetor inteiro de* n *componentes,*

4. $\mid A\overline{\mathbf{x}} \mid_m = \mid \mathbf{b} \mid_m$,

então, $\mid \overline{\mathbf{x}} \mid_m = \mid A^{-1}(\mathrm{mod}m) \mid \mathbf{b} \mid_m \mid_m$. $\square$

Os teoremas 2.2.3 e 2.2.4, servem para escolhermos o módulo necessário para obter uma solução para o sistema de equações lineares.

Teorema 2.2.3 *[NEWMAN, 1967] : Se o módulo* m *for escolhido de maneira que*

1. $m > 2 \mid d \mid$, *e se* d' *for formado a partir de* $\mid d \mid_m$ *tal que satisfaça*

2. $\mid d' \mid_m = \mid d \mid_m$,

3. $\mid d' \mid < m/2$, *então* $d' = d$. $\square$

Teorema 2.2.4 *[NEWMAN, 1967] : Se* $(m, d) = 1$ *e o módulo* m *for escolhido de maneira que*

1. $m > 2\, max_i \mid y_i \mid$, *e se* $\mathbf{y}'$ *for formado a partir de* $\mid \mathbf{y} \mid_m$ *tal que satisfaça*

2. $| \mathbf{y}' |_m = | \mathbf{y} |_m$,

3. $max_i \mid y_i' \mid < m/2$, *então* $\mathbf{y}' = \mathbf{y}$. □

Segundo [NEWMAN, 1967], se trabalhamos com um único módulo m, escolhemos m, de maneira que:

$$m > 2max(\mid d \mid, max_i \mid y_i \mid). \tag{2.13}$$

Os teoremas 2.2.3 e 2.2.4, garantem a inequação (2.13).

Para sistemas lineares simples, em que a matriz A é de ordem 2×2 e o vetor $\mathbf{b}$ é de ordem 2×1, é possível computar d e $\mathbf{y}$, de maneira que m satisfaça a inequação (2.13) e $(m, d) = 1$, e assim, encontrar $\mathbf{x} = (1/d) \times \mathbf{y}$. Porém, nem sempre é possível acharmos a solução, pois, tendo um resultado de uma operação em aritmética residual usando um módulo, não é possível recuperar univocamente o resultado da mesma operação em aritmética inteira com os dados originais. Para garantirmos a unicidade, trabalhamos com a aritmética residual usando mais de um módulo.

Seja $M = m_1\ m_2\ m_3\ ...m_s$ e $M > 1$, em que $(m_i, m_j) = 1$, para $i \neq j$. Podemos assumir que existem suficientes módulos (de tamanho suficiente) para que M satisfaça

$$(d, M) = 1 \tag{2.14}$$

e

$$M > 2max(\mid d \mid, max_i \mid y_i \mid), \tag{2.15}$$

em que $d = \det A$ e $\mathbf{y} = A^{adj}\ \mathbf{b}$.

Os teoremas 2.2.5 e 2.2.6 computam $| d |_m$ e $| \mathbf{y} |_m$.

Teorema 2.2.5 *[YOUNG & GREGORY, 1972] : Se* $a_{11}^{(1)}, a_{22}^{(2)}, ..., a_{nn}^{(n)}$ *forem os pivôs na Eliminação de Gauss-Jordan, então*

$$| d |_m = | a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)} |_m \, . \, \square$$

Teorema 2.2.6 *[YOUNG & GREGORY, 1972] : Podemos computar* $| \mathbf{y} |_m$ *usando a equação*

$$| \mathbf{y} |_m = || d |_m | \overline{\mathbf{x}} |_m |_m \, . \, \square$$

Definição 2.2.8 *[SZABÓ & TANAKA, 1967] : Sejam* a, b *e* m *inteiros e* $m > 1$, *se*

1. $0 < b < m$,
2. $| ab |_m = | ba |_m = 1$,

então escrevemos $b = a^{-1} (\bmod m)$ *e dizemos* b *é um inverso multiplicativo de* a *módulo* m. $\square$

Teorema 2.2.7 (Teorema de Fermat): *Se* a *e* $m > 1$ *são inteiros, e se,*

1. m *é primo,*
2. $a^{-1} (\bmod m)$ *existe,*

então, $a^{-1} (\bmod m) = | a^{m-2} |_m$. $\square$

Definição 2.2.9 *[YOUNG & GREGORY, 1972]: Se o candidato a pivô* $a_{kk}^{(k-1)}$ *(chamando de* p_k*), em que* $a_{kk}^{(k-1)} \neq 0$*, para* $k = 1, 2, ..., n-1$*, tem um inverso multiplicativo módulo* m*, então*

$$\overline{m}_{kk} = p_k^{-1}(\mathrm{mod} m),$$

e para $i \neq k$*,*

$$\overline{m}_{ik} = \left| p_k^{-1}(\mathrm{mod} m) a_{ik}^{(k-1)} \right|_m \text{ ou } \overline{m}_{ik} = \left| \overline{m}_{kk} a_{ik}^{(k-1)} \right|_m .\square$$

Para calcularmos o inverso multiplicativo de a módulo m, denotado por a^{-1} (mod$m$), usamos o Algoritmo Estendido de Euclides[7].

Definição 2.2.10 (Algoritmo Estendido de Euclides - AEE)*[KNUTH, 1968]: Dado dois inteiros positivos* u *e* m*, calculamos o máximo divisor comum* d = m.d.c. (u, m) *e dois inteiros a e v, tal que* $au + vm = d$.□

Vejamos como se dá este processo:

AEE1 - (Inicializamos): $a' \leftarrow v \leftarrow 1$, $a \leftarrow v' \leftarrow 0$, $c \leftarrow u$, $d \leftarrow m$.

AEE2 - (Dividimos): *Dados q e r, como sendo o quociente e o resto de c dividido por d, em que* $c = qd + r$, $0 \leq r < d$.

AEE3 - (O resto é zero?): *Se* $r = 0$*, o algoritmo termina. Temos neste caso,* $au + vm = d$ *como desejado.*

[7]Euclides [KOOGAN & HONAISS, 1992]: matemático grego que ensinava em Alexandria durante o reinado de Ptolomeu I (séc. III a.C.). Autor de Elementos, base da geometria elementar, que contém o famoso postulado de Euclides.

AEE4 - (Permutamos): $c \leftarrow d$, $d \leftarrow r$, $t \leftarrow a'$, $a' \leftarrow a$, $a \leftarrow t - qa$, $t \leftarrow v'$, $v' \leftarrow v$, $v \leftarrow t - qv$ *e retornamos para AEE2.*

O Algoritmo Estendido de Euclides resolve $au + vm = 1$, *para* u *e* m *primos entre si. Então,* $u \equiv a^{-1} \pmod{m}$. *Ver [BAREISS, 1972].*

O algoritmo 3 que descreveremos neste momento, calcula o inverso multiplicativo de a módulo m.

Algoritmo 3:

1. $a[0] := m;$
2. $a[1] := a;$
3. $q[0] := a[0]/a[1];$
4. $a[2] := a[0] - q[0] \times a[1];$
5. $i := 0;$
6. **while** $(a[i+2] \neq 1)$ **do**
7. $\quad i := i + 1;$
8. $\quad q[i] := a[i]/a[i+1];$
9. $\quad a[i+2] := a[i] - q[i] \times a[i+1]$;
10. **endwhile**
11. $n := i;$
12. $u[0] := 0;$
13. $u[1] := 1;$
14. **for** $j = 0$ **until** n **do**
16. $\quad u[j+2] := u[j] - q[j] \times u[j+1];$

17. $u[n+2] := a^{-1}$;

18. **endfor**

19. **if** $(u[n+2] \geq 0)$ **then**

20. *return*

21. **else**

22. $u[n+2] := m + a^{-1}$.

23. **endif** □

Exemplo 2.3 *: Ilustrando a computação do inverso multiplicativo, calcular o inverso multiplicativo de* $a = 8$ *módulo* $m = 11$*, usando o algoritmo 3.*

Assim,

$i := 0$;

$a[0] := 11$;

$a[1] := 8$;

$q[0] := a[0]/a[1] = 11/8 = 1$;

$a[2] := a[0] - q[0] \times a[1] = 11 - 1 \times 8 = 3$;

while $(a[i+2] \neq 1)$ **do**

$i := 1$;

$q[1] := a[1]/a[2] = 8/3 = 2$;

$a[3] := a[1] - q[1] \times a[2] = 8 - 2 \times 3 = 2$;

$i := 2$;

$q[2] := a[2]/a[3] = 3/2 = 1$;

$a[4] := a[2] - q[2] \times a[3] = 3 - 1 \times 2 = 1;$

endwhile

$n := 2;$

$u[0] := 0;$

$u[1] := 1;$

for $j := 0$ **until** 2 **do**

$j = 0;$

$u[2] := u[0] - q[0] \times u[1] = 0 - 1 \times 1 = -1;$

$u[4] := a^{-1};$

$j = 1;$

$u[3] := u[1] - q[1] \times u[2] = 1 - 2 \times (-1) = 3;$

$j = 2;$

$u[4] := u[2] - q[2] \times u[3] = (-1) - (1 \times 3) = -4;$

endfor

if $(u[4] \geq 0)$ **then**

return

else

$u[4] = m + a^{-1} = 11 + (-4) = 7.$

endif

Portanto o inverso multiplicativo de $a = 8$ módulo $m = 11$ é 7, ou seja,

$8^{-1}(\text{mod}11) = 7.$

Definição 2.2.11 *[HOWELL & GREGORY, 1970] : Dado qualquer inteiro x e qualquer módulo inteiro m, se*

1. $r \equiv x(\mathrm{mod} m)$,

2. $-\frac{1}{2}m < r < \frac{1}{2}m$,

então escrevemos $r = /x/_m$ [8] e dizemos que r é um resíduo simétrico de x módulo m. □

Definição 2.2.12 *[HOWELL & GREGORY, 1970] : Uma representação residual simétrica de um inteiro x, para as bases $m_1, m_2, ..., m_s$, é a s-tupla $x \sim \{/x/_{m_1}, /x/_{m_2}, ..., /x/_{m_s}\}$.* □

Teorema 2.2.8 *[HOWELL & GREGORY, 1970] : Dados quaisquer inteiros x e m, $m > 1$, $/x/_m$ é único.* □

2.3 Conversão da Representação Residual ($|x|_{m_i}$) para Representação Residual Simétrica ($/x/_{m_i}$):

Em virtude de simplificar alguma de nossas análises, vamos considerar em nosso sistema residual, a base como sendo constituída de primos ímpares.

A conversão de $| \ x \ |_{m_i}$ para $/x/_{m_i}$, quando os módulos são primos ímpares, é como segue:

[8] as barras / / denotam resíduo simétrico módulo m.

Se

$$0 \leq |x|_{m_i} < \frac{1}{2}m_i, \tag{2.16}$$

então,

$$/x/_{m_i} = |x|_{m_i}.$$

Por outro lado, se (2.16), não é satisfeito, então

$$/x/_{m_i} = |x|_{m_i} - m_i.$$

Se

$$-\frac{1}{2}m_i \leq |x|_{m_i} < 0, \tag{2.17}$$

então,

$$/x/_{m_i} = |x|_{m_i}.$$

Por outro lado, se (2.17) não é satisfeito, então

$$/x/_{m_i} = |x|_{m_i} + m_i.$$

O que estamos fazendo, é mapear os m_i inteiros no intervalo $[0, m_i - 1]$, para os inteiros no intervalo simétrico $[-\frac{1}{2}(m_i - 1), \frac{1}{2}(m_i - 1)]$. Maiores detalhes ver [HOWELL & GREGORY, 1970].

No Estudo de Caso 1, a função inteira *residu*, definida como *residu*[mod (k, m)], em que k e m são inteiros e m é o módulo; faz a conversão da representação residual de d e $\mathbf{y}$ para as suas representações residuais simétricas.

O processo de conversão descrito anteriormente, pode ser implementado pelo algoritmo 6.

Algoritmo 6:

1. **if** $(residu > 0)$ **then**
2. **if** $(-2 \times residu + m \geq 0)$ **then**
3. return
4. **else**
5. $residu = residu - m$
6. **endif**
7. **else**
8. **if** $(residu = 0)$ **then**
9. return
10. **else**
11. **if** $(2 \times residu + m \geq 0)$ **then**
12. return
13. **else**
14. $residu = residu + m$
15. **endif**
16. **endif**
17. **endif** □

Assim, convertemos as representações residuais de d e $\mathbf{y}$ para suas representações residuais simétricas.

$$d \sim \{/d/_{m_1}, /d/_{m_2}, ..., /d/_{m_s}\}$$

e

$$y_i \sim \{/y_i/_{m_1}, /y_i/_{m_2}, ..., /y_i/_{m_s}\}$$

Veremos agora como converter a representação residual simétrica de d e **y** para a sua representação simétrica de base mista.

2.4 Conversão da Representação Residual Simétrica para a Representação Simétrica de Base Mista

Antes de mostrarmos como se dá o processo de converter uma representação residual simétrica para sua representação simétrica de base mista, vamos definir um Sistema Numérico Simétrico de Base Mista. Maiores detalhes ver [HOWELL & GREGORY, 1970].

Definição 2.4.1 : *Seja o conjunto de inteiros $R_1, R_2, ..., R_t$ chamado de base. Então todo inteiro x, no intervalo $(-R/2, R/2)$, pode ser expresso da forma:*

$$x = \delta_1 + \delta_2 R_1 + \delta_3 R_1 R_2 + ... + \delta_t R_1 R_2 ... R_{t-1}, \tag{2.18}$$

em que $\delta_1, \delta_2, ..., \delta_t$ são dígitos simétricos de base mista satisfazendo, para todo i,

$$| \delta_i | < \frac{1}{2} R_i . \square \tag{2.19}$$

Teorema 2.4.1 : *Se $R = \prod_{i=1}^{t} R_i$, então qualquer inteiro x no intervalo $(-\frac{1}{2}R, \frac{1}{2}R)$ tem uma única representação na forma 2.18.* $\square$

É possível construir um Sistema Numérico Simétrico de Base Mista usando os módulos $m_1, m_2, ..., m_s$ como base. Em outras palavras, se

$$M = m_1, m_2, ..., m_s, \tag{2.20}$$

então podemos expressar qualquer inteiro x no intervalo $(-\frac{1}{2}M, \frac{1}{2}M)$, unicamente, na forma

$$x = \alpha_1 + \alpha_2 m_1 + \alpha_3 m_1 m_2 + ... + \alpha_s m_1 m_2 ... m_{s-1}, \tag{2.21}$$

no qual, para todo i,

$$\mid \alpha_i \mid < \frac{1}{2} m_i. \tag{2.22}$$

Assim, pelo teorema 2.4.1, para um dado conjunto de módulos, a representação simétrica de base mista de um inteiro x, que é denotada por

$$x \sim \langle \alpha_1, \alpha_2, ..., \alpha_s \rangle, \tag{2.23}$$

pode ser unicamente determinada.

O sistema residual simétrico com base $m_1, m_2, ..., m_s$ e o sistema simétrico de base mista com $R_i = m_i$, para $i = 1, ..., s$ são chamados Sistemas Numéricos Associados. Estes dois sistemas associados proporcionam a representação do mesmo número de inteiros M, e que dado um inteiro x no intervalo $(-\frac{1}{2}M, \frac{1}{2}M)$, podemos determinar unicamente as representações em cada um dos dois sistemas associados. Além disso, dada uma representação em cada um dos dois sistemas, podemos reconstruir o único inteiro x em $(-\frac{1}{2}M, \frac{1}{2}M)$ que corresponde a representação.

Para reconstruir o único inteiro da representação de base mista é simples. Usamos a equação (2.21), que podemos expressá-la como segue:

$$\begin{cases} P_1 = \alpha_s \\ P_2 = P_1 m_{s-1} + \alpha_{s-1} \\ \vdots \\ P_i = P_{i-1} m_{s-i+1} + \alpha_{s-i+1} \\ \vdots \\ x = P_s = P_{s-1} m_1 + \alpha_1 \end{cases} \tag{2.24}$$

Vejamos como este processo de conversão é descrito em [LINDAMOOD, 1964].

O processo de conversão de base mista que vai ser descrito, é usado para determinar os dígitos simétricos de base mista, α_i, dos dígitos residuais simétricos, $/x/_{m_i}$. Inspeção da equação (2.21) mostra que se tomarmos o residuo simétrico de x módulo m_1, temos

$$/x/_{m_1} = \alpha_1. \tag{2.25}$$

Conseqüentemente, m_1 divide $x - \alpha_1$, e deste modo, se deixamos $x_1 = x$ e

$$x_2 = \frac{x_1 - \alpha_1}{m_1}, \tag{2.26}$$

então x_2 é um inteiro. Portanto, de (2.21) e (2.26), temos:

$$x_2 = \alpha_2 + \alpha_3 m_2 + ... + \alpha_s m_2 ... m_{s-1}, \tag{2.27}$$

Claramente,

$$/x_2/_{m_2} = \alpha_2. \tag{2.28}$$

Desta maneira, repetindo este processo, obtemos

$$x_i = \frac{x_{i-1} - \alpha_{i-1}}{m_{i-1}}, i = 2, ..., s. \tag{2.29}$$

De (2.29) e (2.21), temos:

$$x_i = \alpha_i + \alpha_{i+1}m_i + \alpha_{i+2}m_i m_{i+1} + ... + \alpha_s m_i m_{i+1}...m_{s-1}, \tag{2.30}$$

para $i = 1, ..., s$, e assim

$$/x_i/_{m_i} = \alpha_i. \tag{2.31}$$

Mostramos acima como os dígitos simétricos de base mista α_i, estão relacionados aos inteiros x_i, definidos em (2.29). A seguir será visto, como os α_i são construídos, usando a aritmética residual, da representação residual simétrica de x_i.

Vamos supor que $x = x_1$ tem representação residual simétrica.

$$x_1 \sim \{/x_1/_{m_1}, /x_1/_{m_2}, ..., /x_1/_{m_s}\} \tag{2.32}$$

De (2.31) e teorema 2.2.8, podemos escrever

$$x_1 \sim \{\alpha_1, /x_1/_{m_2}, ..., /x_1/_{m_s}\}. \tag{2.33}$$

Pela definição (2.31),

$$\alpha_1 \sim \{/\alpha_1/_{m_1}, /\alpha_1/_{m_2}, ..., /\alpha_1/_{m_s}\}. \tag{2.34}$$

De (2.26),

$$m_1 x_2 = x_1 - \alpha_1. \tag{2.35}$$

Portanto,

$$x_1 - \alpha_1 \sim \{0, /x_1 - \alpha_1/_{m_2}, ..., /x_1 - \alpha_1/_{m_s}\}. \tag{2.36}$$

Considerando os módulos primos ímpares, podemos calcular o inverso multiplicativo de m_1 módulo m_i, $i = 2, 3, ..., s$. Assim, de (2.26) podemos calcular

$$/x_2/_{m_i} = /\frac{x_1 - \alpha_1}{m_1}/_{m_i},$$

$$/x_2/_{m_i} = /(x_1 - \alpha_1)m_1^{-1}(\mathrm{mod} m_i)/_{m_i}. \tag{2.37}$$

Então, computamos a representação residual simétrica para x_2.

$$x_2 \sim \{/x_2/_{m_2}, /x_2/_{m_3}, ..., /x_2/_{m_s}\}.$$

De (2.31) e teorema 2.2.8, esta representação tornou-se

$$x_2 \sim \{\alpha_2, /x_2/_{m_3}, ..., /x_2/_{m_s}\}.$$

Descrevemos assim, os dois primeiros passos do algoritmo, encontrando α_1 e α_2. Vamos descrever o passo genérico $(j+1)$-ézimo que produz α_{j+1}.

Pela definição (2.31),

$$\alpha_j \sim \{/\alpha_j/_{m_j}, /\alpha_j/_{m_{j+1}}, ..., /\alpha_j/_{m_s}\}, \tag{2.38}$$

em que, $2 \leq j \leq s - 1$.

De (2.29),

$$m_j x_{j+1} = x_j - \alpha_j. \tag{2.39}$$

Assim,

$$x_j - \alpha_j \sim \{0, /x_j - \alpha_j/_{m_{j+1}}, ..., /x_j - \alpha_j/_{m_s}\}. \tag{2.40}$$

Como antes, para $i = j+1, ..., s$, calculamos

$$/x_{j+1}/_{m_i} = /\frac{x_j - \alpha_j}{m_j}/_{m_i},$$

$$/x_{j+1}/_{m_i} = /(x_j - \alpha_j)m_j^{-1}(\mathrm{mod}\, m_i)/_{m_i}. \tag{2.41}$$

Assim,

$$x_{j+1} \sim \{/x_{j+1}/_{m_{j+1}}, /x_{j+1}/_{m_{j+2}}, ..., /x_{j+1}/_{m_s}\}. \tag{2.42}$$

De (2.31) e teorema 2.2.8,

$$x_{j+1} \sim \{\alpha_{j+1}, /x_{j+1}/_{m_{j+2}}, ..., /x_{j+1}/_{m_s}\}. \tag{2.43}$$

Por indução em j e aplicação de (2.31) provamos o teorema seguinte.

Teorema 2.4.2 : *Os inteiros $/x_j/_{m_j}$, para $j = 1, 2, ..., s$ são os coeficientes simétricos de base mista $\alpha_1, \alpha_2, ..., \alpha_s$, respectivamente aparecendo em (2.21).* □

O procedimento visto anteriormente, descreve completamente um algoritmo para converter uma representação residual simétrica para a associada representação simétrica de base mista. Vejamos o exemplo a seguir:

Exemplo 2.4 : *Seja $m_1 = 3$, $m_2 = 5$ e $m_3 = 7$, de maneira que $M = 3.5.7 = 105$. Calcular a representação simétrica de base mista, $\langle \alpha_1, \alpha_2, \alpha_3 \rangle$ e o único inteiro x no intervalo $(-52, 52)$ de $x \sim \{2, 0, 6\}$.*

Aplicando o procedimento descrito na seção 2.4, temos:

$\alpha_1 = x_1 = 2,$

$\alpha_2 = /[m_1^{-1}(x_2 - \alpha_1)]/_{m_2}$, *assim*

$\alpha_2 = /[3^{-1}(0-2)]/_5 = 1$, *então*

$\alpha_2 = 1$. *E por fim,*

$\alpha_3 = /m_2^{-1}[m_1^{-1}(x_3 - \alpha_1) - \alpha_2]/_{m_3},$

$\alpha_3 = /5^{-1}[3^{-1}(6-2) - 1]/_7 = /3.[5(6-2) - 1]/_7,$

$\alpha_3 = /3.[5(4) - 1]/_7 = /3.[6-1]/_7 = /15/_7,$

$\alpha_3 = 1.$

Portanto, $\langle 2, 1, 1 \rangle$ é a representação residual simétrica de base mista. O único inteiro no intervalo $(-52, 52)$ é:

$x = \alpha_1 + \alpha_2 m_1 + \alpha_3 m_1 m_2$, por (2.21).

Assim,

$x = 2 + 1.3 + 1.3.5$

$x = 20$

Vimos, portanto, como converter a representação residual simétrica de d e $\mathbf{y}$ para a sua associada representação simétrica de base mista. Com a representação simétrica de base mista, calculamos d e $\mathbf{y}$, por (2.21).

2.5 Teorema Chinês do Resto

Existem vários algoritmos para obtenção de $\mid d \mid_M$ e $\mid \mathbf{y} \mid_M$. Talvez o mais conhecido procedimento faz uso do teorema clássico da teoria dos números chamado *Teorema Chinês do Resto.*

Teorema 2.5.1 : *Seja* $m_1, m_2, ..., m_s$ *a base para um sistema numérico residual em que* $(m_i, m_j) = 1$, *para* $i \neq j$, *e* $M = m_1 m_2 ... m_s$. *Além disso,*

$$m_j^{'} = \frac{M}{m_j}.$$

Seja q *uma representação residual, da forma*

$q \sim \{r_1, r_2, ..., r_s\}$, *no qual* $r_i = \mid q \mid_{m_i}$, *para* $i = 1, 2, ..., s$.

Então,

$$\mid q \mid_M \;=\; \mid \sum_{j=1}^{s} m_j^{'} \mid r_j m_j^{'-1} (\mathrm{mod} m_j) \mid_{m_j} \mid_M,$$

$$|q|_M = \mid m_1^{'} \mid r_1 m_1^{'-1} (\mathrm{mod} m_1) \mid_{m_1} + \cdots + m_s^{'} \mid r_s m_s^{'-1} (\mathrm{mod} m_s) \mid_{m_s} \mid_M .$$

Exemplo 2.5 : *Se* $m_1 = 7$ *e* $m_2 = 11$, *encontre* $\mid q \mid_M$ *da representação residual* $q \sim \{4, 8\}$.

Solução:

Primeiramente calculamos:

$m_1 = 7;$

$m_2 = 11;$

$M = 7 \times 11 = 77;$

$m_1^{'} = \frac{77}{7} = 11;$

$m_2^{'} = \frac{77}{11} = 7;$

Em seguida,

$| m_1^{'} |_{m_1} = | 11 |_7 = 4;$

$| m_2^{'} |_{m_2} = | 7 |_{11} = 7.$

Assim,

$m_1^{'-1}(\mathrm{mod} m_1) = | 4^{-1} |_7 = 2,$

$m_2^{'-1}(\mathrm{mod} m_2) = | 7^{-1} |_{11} = 8.$

Portanto,

$| q |_{77} = | 11 | (4)(2) |_7 + 7 | (8)(8) |_{11} |_{77} = | 11(1) + 7(9) |_{77} = 74.$

Capítulo 3

Algoritmos Modificados para Aplicação da Aritmética Residual

Neste capítulo, apresentamos os algoritmos de Eliminação de Gauss e Gauss-Jordan estudados no capítulo 2, que são modificados para se adequarem à utilização da aritmética residual. Estes algoritmos são aplicados nos Estudo de Casos, que será visto no capítulo 4.

3.1 Eliminação de Gauss para Aritmética Residual

3.1.1 Eliminação Ordinária

$$| a_{ij}^{(k)} |_m = | a_{ij}^{(k-1)} |_m - \overline{m}_{ik} | a_{kj}^{(k-1)} |_m,$$

no qual, (k) indica que k eliminações foram realizadas.

3.1.2 Eliminação sem Divisão

$$| a_{ij}^{(k)} |_m = ||| (a_{kk}^{(k-1)})^{-1} |_m | a_{ij}^{(k-1)} |_m |_m - || a_{kj}^{(k-1)} |_m | a_{ik}^{(k-1)} |_m |_m |_m,$$

$$| a_{ij}^{(k)} |_m = || (a_{kk}^{(k-1)})^{-1} | a_{ij}^{(k-1)} |_m |_m - | a_{kj}^{(k-1)} | a_{ik}^{(k-1)} |_m |_m |_m,$$

$$| a_{ij}^{(k)} |_m = ||| (a_{kk}^{(k-1)})^{-1} |_m a_{ij}^{(k-1)} |_m - || a_{kj}^{(k-1)} |_m a_{ik}^{(k-1)} |_m |_m .$$

Observe que $| (a_{kk}^{(k-1)})^{-1} |_m = (a_{kk}^{(k-1)})^{-1} \bmod m = \overline{m}_{kk}$. [9]

O algoritmo seguinte, é a implementação do método de Eliminação de Gauss usando a Aritmética Residual.

Algoritmo 4:

1. **for** $k = 1$ **until** $n - 1$ **do**
2. **for** $i = k + 1$ **until** n **do**
3. $\overline{m}_{ik} :=| p_k^{-1} (\bmod m) a_{ik} |_m$
4. **for** $j = k + 1$ **until** $n + 1$ **do**
5. $| a_{ij} |_m :=|| a_{ij} |_m - \overline{m}_{ik} | a_{kj} |_m |_m$
6. **endfor**
7. **endfor**
8. **endfor** □

Antes de aplicarmos o algoritmo 4 para Eliminação de Gauss, calculamos $| [A \mid \mathbf{b}] |_m$, pela definição 2.2.1, vista na seção 2.2, do capítulo 2.

[9] Ver definição 2.2.9, na seção 2.2, do capítulo 2.

3.2 Eliminação de Gauss-Jordan para Aritmética Residual

O algoritmo 5, descrito a seguir, é a implementação do método de Eliminação de Gauss-Jordan usando a Aritmética Residual.

Algoritmo 5:

1. **for** $k = 1$ **until** n **do**
2. **for** $i = 1$ **until** n **do**
3. $\overline{m}_{ik} :=\mid p_k^{-1} (\mathrm{mod}\, m) a_{ik} \mid_m$
4. **for** $j = 1$ **until** $n + 1$ **do**
5. $\mid a_{ij} \mid_m :=\mid\mid a_{ij} \mid_m - \overline{m}_{ik} \mid a_{kj} \mid_m \mid_m$
6. **endfor**
7. **endfor**
8. **endfor** □

Vejamos um exemplo, em que são aplicados os algoritmos 4 e 5, descritos anteriormente.

Exemplo 3.1 *: Seja*

$$A = \begin{bmatrix} 12 & 3 \\ -3 & -1 \end{bmatrix}$$

e

$$\mathbf{b} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

Vamos resolver $| A\bar{\mathbf{x}} |_{m_i} = | \mathbf{b} |_{m_i}$, *pelos algoritmos 4 e 5.*

Solução:

Primeiramente escolhemos os módulos $m_1 = 7$ *e* $m_2 = 11$, *em que* $M = 77$ (*satisfazendo a inequação 2.15*); *pois com um só módulo não garantimos a unicidade na solução.*

Antes de aplicarmos o algoritmo 4 (*Eliminação de Gauss para aritmética residual*) *e o algoritmo 5* (*Eliminação de Gauss-Jordan para aritmética residual*), *calculamos* $| A |_7$, $| \mathbf{b} |_7$, $| A |_{11}$ *e* $| \mathbf{b} |_{11}$ *pela definição 2.2.1, vista na seção 2.2, do capítulo 2.*

Assim,

$$| A |_7 = \begin{bmatrix} 5 & 3 \\ -3 & -1 \end{bmatrix}$$

e

$$| \mathbf{b}' |_7 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}.$$

A matriz aumentada

$$| [A \mid \mathbf{b}] |_7 = \begin{bmatrix} 5 & 3 & -1 \\ -3 & -1 & -2 \end{bmatrix}.$$

$p_1 = a_{11} = 5$ (*pivô*). *Calculemos* $5^{-1}(\text{mod}7)$, *isto é, o inverso multiplicativo de* 5 *módulo* 7 (*pelo algoritmo 3, descrito na seção 2.2 do capítulo 2*), *obtemos* $5^{-1}(\text{mod } 7) = 3$.

Utilizando a algoritmo 4:

$$1^a linha(\times 3) \begin{bmatrix} 5 & 3 & -1 \\ -3 & -1 & -2 \end{bmatrix}$$

$$\sim \begin{bmatrix} |\,15\,|_7 & |\,9\,|_7 & |\,-3\,|_7 \\ -3 & -1 & -2 \end{bmatrix}$$

$$2^a linha - 1^a linha[\times(-3)] \begin{bmatrix} 1 & 2 & -3 \\ -3 & -1 & -2 \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & 2 & -3 \\ -3+3 & -1+6 & -2-|\,9\,|_7 \end{bmatrix}$$

$$2^a linha(\times 3) \begin{bmatrix} 1 & 2 & -3 \\ 0 & 5 & -4 \end{bmatrix}$$

$p_2 = a_{22} = 5$ (*pivô*). *Já vimos que* $5^{-1}(\mathrm{mod}7) = 3$. *Assim,*

$$\begin{bmatrix} 1 & 2 & -3 \\ 0 & |\,15\,|_7 & |\,-12\,|_7 \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & 2 & -3 \\ 0 & 1 & -5 \end{bmatrix}.$$

Portanto,

$|\,\overline{x}_2\,|_7 = |\,-5\,|_7 = -5,$

$|\,\overline{x}_1\,|_7 = ||\,-3\,|_7 - |\,2 \times (-5)\,|_7\,|_7 = |\,-3 - |\,-10\,|_7|_7 = |\,-3-(-3)\,|_7 = 0$ e $|\,\overline{x}\,|_7$ é $\{0, -5\}$.

Usando o algoritmo 5, temos: (continuando o processo anterior)

$$1^a linha - 2^a linha(\times 2) \begin{bmatrix} 1 & 2 & -3 \\ 0 & 1 & -5 \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & \mid 2-2 \mid_7 & \mid -3- \mid -10 \mid_7 \mid_7 \\ 0 & 1 & -5 \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -5 \end{bmatrix}.$$

Portanto, a solução do sistema $\mid A\overline{\mathbf{x}} \mid_7 = \mid \mathbf{b} \mid_7$ *é* $\{0, -5\}$.

Da mesma forma calculamos $\mid A\overline{\mathbf{x}} \mid_{11} = \mid \mathbf{b} \mid_{11}$.

$$\mid A \mid_{11} = \begin{bmatrix} 1 & 3 \\ -3 & -1 \end{bmatrix}$$

e

$$\mid \mathbf{b}' \mid_{11} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}.$$

A matriz aumentada

$$\mid [A \mid \mathbf{b}] \mid_{11} = \begin{bmatrix} 1 & 3 & -1 \\ -3 & -1 & -2 \end{bmatrix}.$$

Aplicando o algoritmo 4, obtemos:

$$2^a linha - 1^a linha[\times(-3)] \begin{bmatrix} 1 & 3 & -1 \\ -3 & -1 & -2 \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & 3 & -1 \\ |-3+3|_{11} & |-1-|-9|_{11}|_{11} & |-2-|3|_{11}|_{11} \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & 3 & -1 \\ 0 & 8 & -5 \end{bmatrix}.$$

$p_2 = a_{22} = 8$ (*pivô*). *Calculemos* $8^{-1}(\text{mod}11)$, *pelo algoritmo 3,* $8^{-1}(\text{mod}11) = 7$.

$$2^a linha(\times 7) \begin{bmatrix} 1 & 3 & -1 \\ 0 & 8 & -5 \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & 3 & -1 \\ 0 & |56|_{11} & |-35|_{11} \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & 3 & -1 \\ 0 & 1 & -2 \end{bmatrix}.$$

Assim,

$|\overline{x}_2|_{11} = -2,$

$|\overline{x}_1|_{11} = |-1-|3 \times (-2)|_{11}|_{11} = 5.$

Aplicando o algoritmo 5 (continuando o método), obtemos:

$$1^a linha - 2^a linha(\times 3) \begin{bmatrix} 1 & 3 & -1 \\ 0 & 1 & -2 \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & |\,3-3\,|_{11} & |\,-1+6\,|_{11} \\ 0 & 1 & -2 \end{bmatrix}$$

$$\sim \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & -2 \end{bmatrix}.$$

Portanto, a solução do sistema $|\,A\overline{\mathbf{x}}\,|_{11} = |\,\mathbf{b}\,|_{11}$ *é* $\{5, -2\}$.

3.3 Quadro de Complexidade segundo E. H. Bareiss

A Tabela 3.1 apresenta uma comparação da complexidade dos algoritmos de Eliminação de Gauss e Gauss-Jordan × a Aritmética Residual. Maiores detalhes ver [BAREISS, 1972].

Métodos Diretos	**Operações (Multiplicação)**	**Aritmética Residual**
Elimanação de Gauss	$O(n^3/3)$	$O\left(n^4 \frac{\delta R}{\omega}\right)$
Eliminação de Gauss-Jordan	$O(n^3/2)$	$O\left(\frac{3}{2} n^4 \frac{\delta R}{\omega}\right)$

Tabela 3.1: Comparação dos Métodos Diretos × Aritmética Residual.

Explanação

β: Base do Sistema Numérico;

α: número máximo de dígitos no elemento a_{ij} da matriz A;

a_{ij}: elemento da matriz A, um inteiro;

w: número de dígitos em precisão simples da palavra do computador.

Usando a Desigualdade de Hadamard [10], *podemos estimar um limite absoluto, para cada elemento $a_{ij}^{(k)}$, em termos dos elementos a_{ij} de A. Para simplificar, assumimos que:*

$$| a_{ij} | < \beta^{\alpha} : \textit{limite do } max_{ij} | a_{ij} | . \tag{3.1}$$

Assim, para algum subdeterminante de ordem n,

$| \det(a_{ij}) | < (\beta^{\alpha}\sqrt{n})^{n} = \beta^{\alpha\delta_n}$: *limite absoluto para* $\det A$ *e pela inequação (3.1),*

$$\delta_n = \alpha + \frac{1}{2}\log_{\beta}^{n}.$$

Portanto,

$$\delta_n = \alpha + \frac{1}{2}\log_{\beta}^{n} \geq \log | \det(a_{ij}) |^{\frac{1}{n}},$$

$$\delta_R = \frac{\delta_n + (\log_{\beta}^{2n+\alpha})}{n}.$$

---

[10]HADAMARD, Jaques *[KOOGAN & HONAISS, 1992]*: matemático francês (Versalhes, 1865 - Paris, 1963), com importantes trabalhos sobre análise infinitesimal.

Desigualdade de Hadamard *[GRADSTEYN & RYZHIK, 1979]*: Seja $A = a_{ii}$, uma matriz não singular, $n \times n$, com elementos reais e determinante $| A |$, então:

$$| A |^2 \leq \prod_{i=1}^{n} \left(\sum_{k=1}^{n} a_{ik}^2 \right). \square$$

Exemplo 3.2 : *Para* $n = 100$

Métodos Diretos	**Operações (Multiplicação)**	**Aritmética Residual**
Eliminação de Gauss	$\frac{1}{3} \times 10^6$	25×10^6
Eliminação de Gauss-Jordan	$\frac{1}{2} \times 10^6$	150×10^6

$\alpha \sim \delta_n \sim \delta_R \sim \omega/4$

Capítulo 4

Estudos de Casos: Detalhes de Implementação

Neste capítulo, são estudados dois algoritmos publicados na *ACM* [11] *Transactions on Mathematical Software*, que estão organizados em Estudos de Casos:

1. Estudo de Caso 1: *Algoritmo 406 - EXACT* - Solução Exata de Equações Lineares usando Aritmética Residual.

2. Estudo de Caso 2: *Algoritmo 522 - ESOLVE* - Técnicas de Congruência para uma Solução Exata de Sistemas Inteiros.

Será apresentado os detalhes de implementação, tais como: a descrição da matriz A dos coeficientes, do vetor independente $\mathbf{b}$ e módulos m (números primos); o resultado obtido nos testes, o tempo de execução e a solução encontrada.

[11] Association for Computing Machinery.

Os Estudos de Casos encontravam-se anteriormente implementados na linguagem Fortran-66, porém foram modificados para Fortran-77, sendo para isto utilizado o compilador *WATFOR77* (Coschi & Schueler, 1989), *versão 3.0* para PC, pois dispõe de recursos de diagnóstico muito bons, tanto na compilação como na execução. Na compilação faz verificação de aderência de um programa em Fortran-77, detecta erros com bastante exatidão, aponta transferências para dentro de um domínio de DO, verifica o tamanho dos COMMON's nas unidades de programa que o utilizam e verifica a precisão de subprogramas FUNCTION definidos e usados em diferentes unidades de programa. Na execução detecta variáveis indefinidas, verifica se os tipos de parâmetros dos subprogramas estão de acordo com a declaração, verifica limites de conjuntos e outros erros comuns. Num ambiente PC-DOS, a velocidade de compilação e imbatível. Utilizamos também, o *Microsoft Fortran PowerStation 4.0*, que é o Sistema de Desenvolvimento Fortran um dos melhores para ambiente Windows 95 e Windows NT, em que podemos criar e rodar programas Fortran de variados tamanhos no PC, sendo interativo com vários produtos da Microsoft, tais como: Microsoft Visual C++, Microsoft Visual Test, Microsoft Visual Basic. O *Microsoft Fortran PowerStation 4.0* incorpora todas as características do Fortran 90, no qual proporciona significativa melhora do Fortran-77 e acrescenta as extensões e flexibilidade de novas linguagens. E por fim, fizemos uso do compilador *Fortran77* para o ambiente AIX, versão 4.1.

Os testes de implementação foram realizados no IBM PowerPC, na Universidade Federal da Paraíba - UFPB, Campus II em Campina Grande.

Os Estudos de Casos podem ser usados para calcular as soluções de duas classes de problemas:

1. Problemas envolvendo matrizes com elementos aleatórios de quatro dígitos ($| a_{ij} | \leq 10^4$);

2. Problemas que envolvem as matrizes de Pascal.

4.1 Descrição das Matrizes e Vetores

As matrizes usadas nos testes são do tipo banda simétrica, simétrica e de Pascal. Vejamos suas definições.

Definição 4.1.1 : *Matriz Banda Simétrica - Uma matriz tem estrutura de banda se os elementos não nulos se concentram em torno da diagonal principal, constituída de elementos a_{ii}, ocupando algumas subdiagonais e superdiagonais. Formalmente, se A for de ordem $m \times n$, tem largura de banda inferior p se $a_{ij} = 0$ para $i > j + p$ e tem largura de banda superior q se $a_{ij} = 0$ para $j > i + q$. No caso geral $p \neq q$. Se $p = q$, $m = n$ e $a_{ij} = a_{ji}$, A é banda simétrica.* □

Em cada linha da matriz simétrica, o elemento da diagonal principal, $d_{i,i}$, $1 \leq i \leq n$, é escolhido de maneira que a matriz A seja diagonal dominante [12] ou

[12]Uma matriz A de ordem $n \times n$ é diagonal dominante se

1) $| a_{ii} | \geq \sum_{j=1 j \neq 1}^{n} | a_{ij} | = r_i, \quad i = 1,2,\cdots,n$ e

2) se existe, pelo menos um k tal que $| a_{kk} | > r_k$.□

estritamente diagonal dominante [13].

Definição 4.1.2 : *Matriz Simétrica - Uma matriz é dita simétrica se* $A = A^T$. □

Definição 4.1.3 : *Matriz de Pascal - Uma matriz é conhecida como matriz de Pascal se for uma matriz simétrica positiva definida* [14], *com entradas inteiras, formando o triângulo de Pascal, cujos elementos são numéricos binomiais* [15]. □

No *Estudo de Caso 1*, foi realizado o teste com as matrizes e vetores do exemplo a seguir.

Exemplo 4.1 : *A matriz A é do tipo banda simétrica, de ordem* $n \times n$, *em que tem bandas superior e inferior de largura 1. As bandas superiores e inferiores são compostas de elementos* $'-1'$ *e os elementos da diagonal principal são escolhidos para que torne a matriz A diagonal dominante ou estritamente diagonal dominante. No nosso exemplo* $(d_{i,i} = 2)$, $1 \leq i \leq n$. *O vetor* $\mathbf{b}$ *é um vetor inteiro, em que seus componentes*

[13] A é estritamente diagonal dominante se

$$| a_{ii} | > \sum_{j=1 j \neq 1} | a_{ij} | \ , \ i = 1, 2, \cdots, n. \square$$

[14] Uma matriz A, $n \times n$, é simétrica positiva definida, se e somente se, A for simétrica e $\mathbf{v}^T A\mathbf{v} > 0$, para qualquer vetor de ordem $n \times 1$, $\mathbf{v} \neq 0$.

[15] Sendo n e p números inteiros não negativos, com $n \geq p$, definimos binomial de ordem n e classe p, como sendo:

$$\begin{pmatrix} n \\ p \end{pmatrix} = \frac{n!}{p!(n-p)!}. \square$$

alternam entre -1 e 1, ou seja, $b_i = (-1)^i$, $i = 1, 2, \cdots, n$.

$$A = \begin{bmatrix} d_{1,1} & -1 & 0 & 0 & 0 & 0 & \cdot & \cdot & 0 & 0 \\ -1 & d_{2,2} & -1 & 0 & 0 & 0 & \cdot & \cdot & 0 & 0 \\ 0 & -1 & d_{3,3} & -1 & 0 & 0 & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & -1 & d_{n-2,n-2} & -1 & 0 \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & 0 & -1 & d_{n-1,n-1} & -1 \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & 0 & 0 & -1 & d_{n,n} \end{bmatrix}. \square$$

Para $n = 4$, *temos:*

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}. \square$$

Exemplo 4.2 *: A matriz A é simétrica, de ordem* $n \times n$, *em que* $A_{n\times n} = (a_{ij})_{n\times n} = \begin{cases} n, & \text{para } i = j \\ n-1, & \text{para } i \neq j \end{cases}$. *O vetor* $\mathbf{b}$ *é a soma dos elementos* a_{ij} *da linha* i, *para* $i = 1, 2, \cdots, n$. *Então,*

$$b_i = \sum_{j=1}^{n} a_{ij}.$$

Assim, para $n = 5$, *temos:*

$$A = \begin{bmatrix} 5 & 4 & 4 & 4 & 4 \\ 4 & 5 & 4 & 4 & 4 \\ 4 & 4 & 5 & 4 & 4 \\ 4 & 4 & 4 & 5 & 4 \\ 4 & 4 & 4 & 4 & 5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 21 \\ 21 \\ 21 \\ 21 \\ 21 \end{bmatrix}. \square$$

Exemplo 4.3 : *A matriz A é conhecida como matriz de Pascal de ordem* 6×6. *O vetor* **b** *é escolhido de maneira que forme o triângulo de Pascal.*

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 6 & 10 & 15 & 21 \\ 1 & 4 & 10 & 20 & 35 & 56 \\ 1 & 5 & 15 & 35 & 70 & 126 \\ 1 & 6 & 21 & 56 & 126 & 252 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 7 \\ 28 \\ 84 \\ 210 \\ 462 \end{bmatrix}. \square$$

Para fazermos um estudo comparativo entre os *Estudo de Casos* e os métodos diretos convencionais (Eliminação de Gauss e Gauss-Jordan), escolhemos estes três tipos de matrizes, descritas acima.

4.2 Estudo de Caso 1: Algoritmo 406 - EXACT - Solução Exata de Equações Lineares usando a Aritmética Residual

A subrotina *EXACT* resolve o sistema de equações lineares $A\mathbf{x} = \mathbf{b}$, usando a Aritmética Residual, em que a matriz A dos coeficientes, é inteira de ordem $n \times n$, não singular, $\mathbf{x}$ e $\mathbf{b}$ são vetores colunas com n componentes. Os componentes do vetor $\mathbf{x}$ são números racionais e os do vetor $\mathbf{b}$, números inteiros.

O *Estudo de Caso 1* pode ser usado para resolver sistemas de equações lineares algébricas, obter matrizes inversas, computar determinantes e matrizes adjuntas.

4.2.1 Procedimento Computacional

A solução formal, para resolver $A\mathbf{x} = \mathbf{b}$, pode ser escrita:

$$\mathbf{x} = A^{-1}\mathbf{b}, \text{ como } A^{-1} = (1/d) \times A^{adj}, \text{ temos:}$$

$$\mathbf{x} = (1/d) \times A^{adj} \times \mathbf{b},$$

$$\mathbf{x} = (1/d) \times \mathbf{y}, \text{ em que, } d = \det A \text{ e } \mathbf{y} = A^{adj} \times \mathbf{b}.$$

O algoritmo 406 [16] usa a definição 2.2.1 de resíduo, vista no capítulo 2, que é refletida pela função FORTRAN mod, para resolver $\mid A\overline{\mathbf{x}} \mid_{m_i} = \mid \mathbf{b} \mid_{m_i}$, $i =$

[16] *Estudo de Caso 1*: Algoritmo 406: EXACT - Solução Exata de Equações Lineares usando a Aritmética Residual.

$1, 2, \cdots, s$. A computação é realizada pelo método de Eliminação de Gauss-Jordan (*algoritmo 5*), descrito no capítulo 3.

Em seguida, encontramos as representações residuais de d e $\mathbf{y}$, pelos teoremas 2.2.5 e 2.2.6, obtendo:

$$d \sim \{| d |_{m_1}, | d |_{m_2}, \cdots, | d |_{m_s}\} \tag{4.1}$$

e

$$\mathbf{y} \sim \{| \mathbf{y} |_{m_1}, | \mathbf{y} |_{m_2}, \cdots, | \mathbf{y} |_{m_s}\}. \tag{4.2}$$

Dando prosseguimento ao método descrito pelo *algoritmo 406*, convertemos as representações residuais de d e de $\mathbf{y}$, para as suas representações residuais simétricas, em que os detalhes podem ser encontrados na seção 2.3, do capítulo 2, em que:

$$d \sim \{/d/_{m_1}, /d/_{m_2}, \cdots, /d/_{m_s}\} \tag{4.3}$$

e

$$\mathbf{y} \sim \{/\mathbf{y}/_{m_1}, /\mathbf{y}/_{m_2}, \cdots, /\mathbf{y}/_{m_s}\}. \tag{4.4}$$

Dando continuidade, convertemos as representações residuais simétricas de d e de $\mathbf{y}$, para as suas representações simétricas de base mista. Maiores detalhes, ver seção 2.4, do capítulo 2, no qual:

$$d \sim \langle \alpha_1, \alpha_2, \cdots, \alpha_s \rangle$$

e

$\mathbf{y} \sim \langle \beta_1, \beta_2, \cdots, \beta_s \rangle$, cujos elementos de d e $\mathbf{y}$ são obtidos como segue:

$$d = \alpha_1 + \alpha_2 m_1 + \alpha_3 m_1 m_2 + \cdots + \alpha_s (m_1, m_2, \cdots, m_{s-1}) \tag{4.5}$$

e

$$\mathbf{y} = \beta_1 + \beta_2 m_1 + \beta_3 m_1 m_2 + \cdots + \beta_s(m_1, m_2, \cdots, m_{s-1}) \tag{4.6}$$

Calculado d e $\mathbf{y}$ pelas equações 4.5 e 4.6, podemos encontrar $\mathbf{x}$ imediatamente de $\mathbf{x} = (1/d) \times \mathbf{y}$. □

Aplicando este procedimento ao *exemplo 3.1*, visto no capítulo 3, encontramos as representações de $d \sim \{|\ d\ |_7, |\ d\ |_{11}\}$ e de $\mathbf{y} \sim \{|\ \mathbf{y}\ |_7, |\ \mathbf{y}\ |_{11}\}$, tais como:

$$d \sim \{4, 8\}$$

e

$$\mathbf{y} \sim \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 7 \\ -5 \end{bmatrix} \right\}.$$

Em que convertemos para as suas representações residuais simétricas, obtendo:

$$d \sim \{-3, -3\}$$

e

$$\mathbf{y} \sim \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -4 \\ -5 \end{bmatrix} \right\}.$$

Neste ponto é mais apropriado chamar $\mathbf{y}$, de $y_1 = \{0, -4\}$ e $y_2 = \{1, -5\}$.

Em seguida, convertemos as representações residuais simétricas para as representações simétricas de base mista, em que:

$$d \sim \langle -3,\ 0 \rangle$$

e

$$y_1 = \langle 0, 1 \rangle,\ y_2 = \langle 1, -4 \rangle.$$

Neste momento, computamos d e $\mathbf{y}$, por 4.5 e 4.6. Então:

$$d = -3 + 0 \times 7 = -3$$

e

$$y_1 = 0 + 1 \times 7 = 7,\ y_2 = 1 + (-4) \times 7 = -27.$$

Portanto, podemos encontrar $\mathbf{x} = (1/d) \times \mathbf{y}$.

Assim,

$$d = -3,$$

$$\mathbf{y} = \begin{bmatrix} 7 \\ -27 \end{bmatrix},$$

e,

$$\mathbf{x} = \begin{bmatrix} -\frac{7}{3} \\ 9 \end{bmatrix}.\square$$

Existe uma outra maneira de resolvermos o sistema residual $| A\overline{\mathbf{x}} |_{m_i} = | \mathbf{b} |_{m_i}$, semelhante ao apresentado, que vamos chamar de *ALGORITMO I.*

Primeiramente, usamos a definição 2.2.2 de resíduo, vista na seção 2, do capítulo 2, que difere da definição de resíduo usada pelo *algoritmo 406.* O mesmo esquema é usado para encontrar as representações residuais de d e de $\mathbf{y}$. Prosseguindo, usamos o Teorema Chinês do Resto, descrito na seção 2.5, para computar $| d |_M$ e $| \mathbf{y} |_M$, pelos teoremas 2.2.5 e 2.2.6, em que $M = m_1 m_2 \cdots m_s$. Usando os teoremas 2.2.3 e 2.2.4, computamos d e $\mathbf{y}$ e a solução é obtida. Maiores detalhes em relação ao *ALGORITMO I*, consultar [HOWELL & GREGORY, 1969].

Aplicando o *ALGORITMO I*, para o *exemplo 3.1*, temos que primeiramente resolver o sistema residual $| A\overline{\mathbf{x}} |_7 = | \mathbf{b} |_7$, para $| \overline{\mathbf{x}} |_7$, $|d|_7$ e $| \mathbf{y} |_7$.

Assim, a matriz ampliada

$$| [A \mid \mathbf{b}] |_7 = \begin{bmatrix} 5 & 3 & 6 \\ 4 & 6 & 5 \end{bmatrix},$$

pode ser reduzida à:

$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix}$, usando o mesmo procedimento descrito pelo *Algoritmo 406.*

Como os dois pivôs são 5 e 5, então:

$$| \overline{\mathbf{x}} |_7 = \begin{bmatrix} 0 \\ 2 \end{bmatrix},$$

$$| d |_7 = | 5 \times 5 |_7 = 4$$

e

$$| \mathbf{y} |_7 = || d |_7 | \overline{x} |_7 |_7 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Resolvendo o sistema residual $| A\overline{x} |_{11} = | \mathbf{b} |_{11}$, para $| \overline{x} |_{11}$, $| d |_{11}$ e $| \mathbf{y} |_{11}$, temos a matriz ampliada,

$$| [A \mid \mathbf{b}] |_{11} = \begin{bmatrix} 1 & 3 & 10 \\ 8 & 10 & 9 \end{bmatrix},$$

sendo reduzida à:

$\begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 9 \end{bmatrix}$, em que os dois pivôs são 1 e 8.

Então,

$$| \overline{x} |_{11} = \begin{bmatrix} 5 \\ 9 \end{bmatrix},$$

$$| d |_{11} = | 1 \times 8 |_{11} = 8$$

e

$$| \mathbf{y} |_{11} = || d |_{11} | \overline{x} |_{11} |_{11} = \begin{bmatrix} 7 \\ 6 \end{bmatrix}.$$

Logo as representações residuais de d e de $\mathbf{y}$, são:

$$d \sim \{4, 8\}$$

e

$$\mathbf{y} \sim \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 7 \\ 6 \end{bmatrix} \right\}, \text{ sendo mais apropriado escrever:}$$

$$y_1 \sim \{0, 7\}$$

e

$$y_2 \sim \{1, 6\}.$$

Usando o *Teorema do Chinês do Resto*, a princípio computamos:

$$m_1 = 7 \quad M = 77 \quad m_1^{'} = 11$$

$$m_2 = 11 \qquad\qquad m_2^{'} = 7,$$

e,

$$| m_1^{'} |_{m_1} = 4,$$

$$| m_2^{'} |_{m_2} = 7.$$

Assim,

$$m_1^{'-1} (\mathrm{mod}\, m_1) = | 4^{-1} |_7 = 2,$$

$$m_2^{'-1} (\mathrm{mod}\, m_2) = | 7^{-1} |_{11} = 8.$$

De modo que,

$$| d |_{77} = | 11 | (4) \times (2) |_7 + 7 | (8) \times (8) |_{11} |_{77} = 74,$$

$$| y_1 |_{77} = | 11 | (0) \times (2) |_7 + 7 | (7) \times (8) |_{11} |_{77} = 7$$

e,

$$| y_2 |_{77} = | 11 | (1) \times (2) |_7 + 7 | (6) \times (8) |_{11} |_{77} = 50.$$

Usamos os teoremas 2.2.3 e 2.2.4 para computar d e $\mathbf{y}$, respectivamente. Procuramos números menores que $\frac{77}{2}$ em magnitude que são congruentes módulo 77, a 74, 7 e 50. Estes números são -3, 7 e -27.

Portanto,

$$d = -3,$$

$$\mathbf{y} = \begin{bmatrix} 7 \\ \\ -27 \end{bmatrix},$$

e, obtemos a solução

$$\mathbf{x} = \begin{bmatrix} -\frac{7}{3} \\ \\ 9 \end{bmatrix}. \square$$

4.2.2 Descrição dos Módulos

Na prática, os módulos são escolhidos como números primos. Essa escolha aumenta a probabilidade que $(d, m_i) = 1$ e $| d |_{m_i} \neq 0$. Se estas duas condições são satisfeitas, então A é não singular módulo m_i e o sistema residual $| A\bar{\mathbf{x}} |_{m_i} = | \mathbf{b} |_{m_i}$ pode ser resolvido para $| d |_{m_i}$ e $| \mathbf{y} |_{m_i}$. Se as duas condições não são satisfeitas, então simplemente selecionamos outro primo para o módulo. Além disso, por escolha

dos módulos como números primos, garantimos que $(m_i, m_j) = 1$ para $i \neq j$ e assim, pelo teorema 4.1 [17] existe um único inteiro no intervalo $(0, m-1)$ com uma dada representação residual.

A escolha dos módulos necessários para obter uma solução, é prevista pela subrotina *LOGBND*, cujas informações mais detalhadas ver [YOUNG & GREGORY, 1972].

Um procedimento para a escolha de M é operar com um conjunto de módulos $m_1, m_2, \cdots, m_s$ e escolhermos $M = m_1 m_2 \cdots m_s (s \leq r)$, tal que

$$M \geq 2 \prod_{i=1}^{n} \left(\sum_{k=1}^{n} a_{ik}^2 \right)^{\frac{1}{2}} \sum_{j=1}^{n} \mid b_j \mid$$

e satisfaz a inequação (2.15), vista na seção 2.2, do capítulo 2.

O programa calcula o *BOUND* pela fórmula:

$$BOUND = \log \left(2 \left[\prod_{i=1}^{n} \left(\sum_{j=1}^{n} a_{ij}^2 \right)^{\frac{1}{2}} \prod_{k=1}^{n} \mid b_k \mid \right] \right),$$

[17] **Teorema 4.1** *[YOUNG & GREGORY, 1972]*: Dois inteiros x e y têm a mesma representação residual para a base $m_1, m_2, \cdots, m_s$ se, e somente se $x \equiv y (\mathrm{mod}\, L)$, em que L é o mínimo múltiplo comum dos módulos. □

Exemplo 4.1: Dados $m_1 = 7$, $m_2 = 9$. Assim, *m.m.c.*(7,9) = 63 e pelo teorema anterior, segue que:

$$138 \equiv 12 (\mathrm{mod}\, 63), \text{ em que } x = 138 \text{ e } y = 12.$$

Portanto, 138 e 12 têm a mesma representação residual para a base 7 e 9.

$$138 \sim \{\mid 138 \mid_7, \mid 138 \mid_9\} \sim \{5, 3\}$$

$$12 \sim \{\mid 12 \mid_7, \mid 12 \mid_9\} \sim \{5, 3\}. \ \square$$

Corolário 4.2 *[YOUNG & GREGORY, 1972]*: A representação residual dos inteiros w no intervalo $0 \leq w \leq L - 1$ são distintas. □

($| b_k | \geq 0$) e mais de um módulo é escolhido de modo que:

$$BOUND \leq SUMLOG = \log(m_1) + \cdots + \log(m_s).$$

Antes de nos reportar ao *Estudo de Caso 2*, introduziremos uma linguagem especial de congruências, que é extremamente útil em teoria dos números. Esta liguagem foi desenvolvida no século XIX, por Karl Friedrich Gauss [18], um dos mais famosos matemáticos da história.

4.3 Congruências

Definição 4.3.1 : *Se a, b e m são inteiros e se $m \neq 0$ divide $a - b$, $(m \mid a - b)$, escrevemos $a \equiv b (\mathrm{mod}\, m)$ e dizemos a é congruente a b módulo m.* □

Teorema 4.3.1 : *Se a e b são inteiros, então $a \equiv b (\mathrm{mod}\, m)$ se e somente se existe um inteiro k tal que $a = b + km$* □.

Teorema 4.3.2 : *Sejam m um inteiro positivo e a, b, c, d, x e y inteiros quaisquer. Então tem-se:*

1. $a \equiv a (\mathrm{mod}\, m)$, *propriedade reflexiva.*

2. $a \equiv b (\mathrm{mod}\, m)$, *então* $b \equiv a (\mathrm{mod}\, m)$, *propriedade simétrica.*

3. $a \equiv b (\mathrm{mod}\, m)$ *e* $b \equiv c (\mathrm{mod}\, m)$, *então* $a \equiv c (\mathrm{mod}\, m)$, *propriedade transitiva.*

[18] GAUSS, Carl Friedrich *[KOOGAN & HONAISS, 1992]*: astrônomo, matemático e físico alemão (Brunswick, 1777 - Götttingen, 1855), autor de trabalhos sobre mecânica celeste, a teoria dos erros, o magnetismo, o eletromagnetismo e a óptica.

4. *Se* $a \equiv b(\mathrm{mod}\, m)$ *e* $c \equiv d(\mathrm{mod}\, m)$, *então* $a+c \equiv b+d(\mathrm{mod}\, m)$, $a-c \equiv b-d(\mathrm{mod}\, m)$ *e* $ac \equiv bd(\mathrm{mod}\, m)$.

5. *Se* $a \equiv b(\mathrm{mod}\, m)$ *e* $c \equiv d(\mathrm{mod}\, m)$, *então* $ax + cy \equiv bx + dy(\mathrm{mod}\, m)$.

6. *Se* $d > 0$ *divide* m *e se* $a \equiv b(\mathrm{mod}\, m)$, *então* $a \equiv b(\mathrm{mod}\, d)$.

7. *Se* $ac \equiv bc(\mathrm{mod}\, m)$ *e* c *e* m *são primos entre si, isto é,* $(c, m) = 1$, *então* $a \equiv b(\mathrm{mod}\, m)$.

8. *Se* $a \equiv b(\mathrm{mod}\, m)$, *então* $a^k \equiv b^k(\mathrm{mod}\, m)$, *para todos inteiros* $k > 0$. □

Definição 4.3.2 : *Um Sistema Completo de Restos Módulos* m *é um conjunto de inteiros tal que todo inteiro é congruente módulo* m *a exatamente um inteiro do conjunto.* □

Exemplo 4.4 : *O algoritmo da divisão mostra que o conjunto de inteiros* $0, 1, 2, \cdots, m-1$ *é um sistema completo de restos módulo* m. *Este conjunto é chamado de menor resto (resíduo) não negativo módulo* m. □

Teorema 4.3.3 : *Se* $r_1, r_2, \cdots, r_m$ *é um sistema completo de restos (resíduo) módulo* m, *e se* a *é um inteiro positivo com* $(a, m) = 1$, *então* $ar_1 + b, ar_2 + b, \cdots, ar_m + b$ *é um sistema completo de restos módulo* m. □

Teorema 4.3.4 : *Se* $a \equiv b(\mathrm{mod}\, m_1)$, $a \equiv b(\mathrm{mod}\, m_2), \cdots, a \equiv b(\mathrm{mod}\, m_k)$, *em que* a, b, $m_1, m_2, \cdots, m_k$ *são inteiros, com* $m_1, m_2, \cdots, m_k$ *positivos, então* $a = b(\mathrm{mod}\, [m_1, m_2, \cdots, m_k])$, *em que* $[m_1, m_2, \cdots, m_k]$ *é o mínimo múltiplo comum de* $m_1, m_2, \cdots, m_k$. □

Corolário 4.3.1 : *Se* $a \equiv b(\mathrm{mod}\, m_1)$, $a \equiv b(\mathrm{mod}\, m_2), \cdots, a \equiv b(\mathrm{mod}\, m_k)$, *em que* a, b *são inteiros e* $m_1, m_2, \cdots, m_k$ *são pares de inteiros positivos relativamente primos entre si, então* $a \equiv b(\mathrm{mod}\, m_1 m_2 \cdots m_k)$. □

Definição 4.3.3 : *Uma congruência da forma* $ax \equiv b(\mathrm{mod}\, m)$, *em que* a, b *e* $m > 0$ *são inteiros e* x *um inteiro desconhecido, é chamada de um congruência linear de uma variável. O estudo de tal congruência e similar ao estudo das equações diofantinas de duas variáveis* [19]. □

Teorema 4.3.5 : *Sejam* a, b *e* c *são inteiros com* $d = (a, b)$, *isto é,* d *é o máximo divisor comum entre* a *e* b. *A equação* $ax + by = c$ *não tem solução inteira se* d *não divide* c. *Se* $d \mid c$, *então existe infinitas soluções inteiras. Além disso, se* $x = x_0$, $y = y_0$ *é uma solução particular da equação, então todas as soluções são dadas por* $x = x_0 + (b/d)n$, $y = y_0 - (a/d)n$, *em que* n *é um inteiro.* □

Teorema 4.3.6 : *Sejam* a, b, *e* $m > 0$ *inteiros e* $(a, m) = d$. *Se* d *não divide* b, *então* $ax \equiv b(\mathrm{mod}\, m)$ *não tem solução. Se* $d \mid b$, *então* $ax \equiv b(\mathrm{mod}\, m)$ *tem exatamente* d *soluções incongruentes módulo* m. □

Vamos considerar congruências da forma especial $ax \equiv 1(\mathrm{mod}\, m)$. Pelo teorema 4.3.6 existe uma solução para esta congruência se e somente se $(a, m) = 1$, e então todas as soluções são congruentes módulo m.

Definição 4.3.4 : *Dado um inteiro* a *com* $(a, m) = 1$, *uma solução de* $ax \equiv 1(\mathrm{mod}\, m)$ *é chamada um inverso de* a *módulo* m. □

[19] *Equações Diofantinas*: São equações polinomiais a um certo número de variáveis, nas quais os coeficientes são inteiros e onde se procuram soluções em inteiros.

Teorema 4.3.7 : *(Pequeno Teorema de Fermat) Seja m um primo positivo. Então:*

1. *Se a é um inteiro não divisível por m, então $a^{m-1} \equiv 1(\mathrm{mod}\, m)$.*

2. *Se a um inteiro qualquer, então $a^m \equiv 1(\mathrm{mod}\, m)$.* □

Pequeno Teorema de Fermat possibilita soluções de certas congruências lineares:

Proposição 4.3.1 : *Sejam m um primo positivo, e a e b inteiros tais que $(a, m) = 1$. Então a congruência linear $ax \equiv b(\mathrm{mod}\, m)$ admite uma solução. Além do mais a solução é única módulo m.* □

Maiores detalhes ver [ROSEN, 1988] e [VISWANATHAN, 1979].

4.4 Estudo de Caso 2: Algoritmo 522 - ESOLVE, Técnicas de Congruência para uma Solução Exata de Sistemas de Equações Lineares Inteiros

A subrotina *ESOLVE* resolve exatamente o sistema de equações lineares $A\mathbf{x} = \mathbf{b}$, em que A uma matriz inteira de múltipla precisão, $n \times n$, não singular, $\mathbf{x}$ e $\mathbf{b}$ são vetores colunas com n componentes. As componentes do vetor $\mathbf{b}$ são números inteiros, e de $\mathbf{x}$, são números racionais. A subrotina retorna um d [20] e um vetor $\mathbf{y} = A^{adj} \times \mathbf{b}$, ambos inteiros de múltipla precisão. Se $d \neq 0$, obtemos a solução do sistema,

[20] $d = \det A$.

computando $\mathbf{x} = \mathbf{y}/d$. Por outro lado, se $d = 0$, o sistema de equações lineares é compatível e indeterminado, ou é incompatível, isto é, não tem solução. *ESOLVE* não pode ser usado para distinguir entre estes dois casos, pois detecta incompatibilidade $rank(A)$[21] $= n - 1$, como também, não pode gerar famílias paramétricas de soluções, no caso do sistema ser indeterminado.

O *Estudo de Caso 2* pode ser usado para resolver sistemas de equações lineares algébricas, computar determinantes e matrizes adjuntas.

4.4.1 Procedimento Computacional

Em *ESOLVE*, a forma de base mista, apresentada na seção 2.4 do capítulo 2, é usada para a representação de todos os inteiros de múltipla precisão. Um sistema numérico de base mista, para as bases ordenadas $m_1, m_2, \cdots, m_s$, é um conjunto de seqüências de dígitos para o inteiro x, no qual é denotado em 2.23. Um caso especial, ocorre se $B = m_1 = m_2 = \cdots = m_s$, o conhecido sistema numérico de base fixa. Assim, se x é um inteiro, podemos expressá-lo unicamente na forma

$$x = c_1 + c_2 B + \cdots + c_n B^{(n-1)},$$

no qual os números $c_1, c_2, \cdots, c_n$ são os coeficientes de base fixa, em que $\mid c_i \mid < B$, para $i = 1, 2, \cdots, n$. Para maiores esclarecimentos ver [SODERSTRAND & JENKINS, 1986].

[21] *Definição 4.2.1*: Dada uma matriz A, $m \times n$, seja C, $m \times n$, a matriz-linha reduzida à forma escada linha equivalente a A. O posto de A, denotado por $rank(A)$, é o número de linhas não nulas de C. □

No *Estudo de Caso 2*, primeiramente convertemos a matriz ampliada, $[A \mid \mathbf{b}]$, da representação de base fixa ($B = 10000$) para a representação de base mista, usando a subrotina *MRADIX*. Caso queiramos converter a representação de base mista para a representação de base fixa, usamos a subrotina *FRADIX*.

Dando continuidade ao Algoritmo 522, a subrotina *SUBBND*, usa a Desigualdade de Hadamard , para obtermos um limite mais rigoroso - *BOUND* - para d e $\mathbf{y}$, no qual,

$$BOUND \geq \begin{cases} \prod_{j=1}^{n} \left(\sum_{i=1}^{n} a_{ij}^2\right)^{\frac{1}{2}}, \ \overline{b} \leq \overline{a} \\ \prod_{j=1}^{n} \left(\sum_{i=1}^{n} a_{ij}^2\right)^{\frac{1}{2}} \ \overline{b}/\overline{a}, \ \overline{b} > \overline{a}, \end{cases}$$

$$\text{em que, } \overline{a} = \min_{1 \leq j \leq n} \left\{ \left(\sum_{i=1}^{n} a_{ij}^2 \right)^{\frac{1}{2}} \right\}, \ \overline{b} = \max_{1 \leq 1 \leq m} \left\{ \left(\sum_{k=1}^{n} b_{kl}^2 \right)^{\frac{1}{2}} \right\}.$$

A subrotina *SUBBND* é complexa para computar o *BOUND* porque os elementos da matriz A e os componentes do vetor $\mathbf{b}$, são inteiros de múltipla precisão. Esta subrotina, também calcula o número máximo de primos (*MAXPRM*) necessários para representar d e $\mathbf{y}$ e obtermos a solução do sistema $A\mathbf{x} = \mathbf{b}$.

Como entrada para *ESOLVE*, o usuário precisa especificar um valor para o parâmetro *MAXPRM* (número máximo de primos) para que cada componente y_i de $\mathbf{y}$ e d satisfaça

$$\mid d \mid, \mid y_i \mid \leq [m_1 \times \cdots \times m_{MAXPRM} - 1]/2.$$

O menor inteiro *MAXPRM* é usado tal que,

$$BOUND \leq [m_1 \times \cdots \times m_{MAXPRM} - 1]/2.$$

O número de primos é escolhido de modo que:

$$BOUND \leq SUMLOG = \log(m_1) + \log(m_2) + \cdots + \log(m_s).$$

A escolha do *MAXPRM*, pode ou não ser crucial, dependendo se o usuário atribuir ou não, o *RTEST* sendo falso ou *RTEST* verdadeiro. Se for falso, *ESOLVE* continua a computar os coeficientes da representação de base mista de d e $\mathbf{y}$ (isto é, continua a repetir) até o *MAXPRM*, cujos coeficientes foram encontrados. Se *RTEST* for verdadeiro, *ESOLVE* continua a repetir somente até um ou outro coeficiente *MAXPRM* ter sido computado ou até os coeficientes consecutivo zero - *TR* na representação de base mista de d e $\mathbf{y}$ serem encontrados, o que acontecer primeiro. *TR* é definido como o menor inteiro para o qual,

$$|| [A \mid \mathbf{b}] ||_\infty \leq 2 \times m_1 \times \cdots \times m_{TR},$$

em que, $|| [A \mid \mathbf{b}] ||_\infty$ [22] é a norma infinita da matriz ampliada do sistema $A\mathbf{x} = \mathbf{b}$. Maiores detalhes, vejamos o teorema 4.4.1 e corolário 4.4.1.

Teorema 4.4.1 *[BAREISS, 1972]: Seja uma matriz quadrada A, vetores $\mathbf{y}$ e $\mathbf{b} \neq 0$ tal que $A\mathbf{y} = \mathbf{b}d$, e a representação de base mista*

$$\begin{bmatrix} y \\ d \end{bmatrix} \sim [\alpha_{jk}]$$

Correspondendo a um conjunto de primos $\{p_i > 2, i = 1, 2, \cdots\}$. Se em $[\alpha_{ij}]$, coluna m contém pelo menos um elemento não nulo α_{im}, e é seguido de pelo menos t colunas zero tal que

$$\pi_{m+t} > \frac{1}{2} || [A \mid \mathbf{b}] ||_\infty \sum_{k=1}^{m} \pi_k, \text{ em que } \pi_m = \prod_{k=1}^{m} p_k.$$

[22] A norma infinita de A é definida por:

$$|| A ||_\infty = max_i \sum_{j=1}^{n} | a_{ij} | \ . \ \square$$

Então,

1. $A\mathbf{y}^{(m)} = \mathbf{b}d^{(m)}$,

2. $A\mathbf{x} = \mathbf{b}$, $\mathbf{x} = \mathbf{y}^{(m)}/d^{(m)})$, *se* $d^{(m)} \neq 0$,

3. $d = 0$, *se* $d^{(m)} = 0$ *e* $\mathbf{y}^{(m)} \neq 0$. □

Corolário 4.4.1 *[BAREISS, 1972]: Se no teorema 4.4.1 os primos são ordenados em ordem crescente então, um suficiente limite inferior para t é dado por $2\pi_t > || [A, \mathbf{b}] ||_\infty$.*

□

A validade do teste recursivo é provada por [BAREISS, 1972] e [CABAY, 1971]. Se o teste recursivo for satisfeito, $A \times \mathbf{y} = d \times \mathbf{b}$ será verdadeiro, mesmo se em várias ocasiões, $\mathbf{y}$ possa não ser igual a $A^{adj} \times \mathbf{b}$ e d possa não mais ser o determinante de A. Portanto, quando $d \neq 0$, $\mathbf{x} = \mathbf{y}/d$ ainda dará a solução requisitada.

Dando prosseguimento ao algoritmo 522, resolvemos o sistema residual $| A\overline{\mathbf{x}} |_{m_i} = | \mathbf{b} |_{m_i}$, para $m_i = 1, 2, \cdots, MAXPRM$, usando o método de Eliminação de Gauss (algoritmo 4), descrito no capítulo 3. Para calcularmos o inverso multiplicativo de a módulo m, aplicamos o Algoritmo Estendido de Euclides - algoritmo 3, visto na seção 2.2 do capítulo 2. Aqui, computamos os termos da representação de base mista de d e de $\mathbf{y}$ usando o Teorema Chinês do Resto (descrito no capítulo 2). E, finalizando o *Estudo de Caso 2*, obtemos a solução $\mathbf{x} = \mathbf{y}/d$.

4.4.2 Escolha dos Módulos

Uma lista ótima de números primos para *ESOLVE* é definida como sendo uma lista $m_1, m_2, \cdots$ em ordem crescente, escolhida de forma a torná-la a maior possível, mas sujeita a condição de que o resultado da operação $m_i \times m_j$, $\forall_{i,j}$ não acarreta overflow. Com isso, uma lista ótima de números primos será gerada, uma de cada vez, para cada instalação diferente e armazenda em um único bloco de nome *PRIMEB*. A maneira como esta lista é gerada fica transparente ao usuário. O comprimento da lista é determinado de acordo com o tamanho do problema e fornecido ao usuário. Para o IBM 360/67, em que a palvra simples de computador é 32 bits, a lista ótima de primos é dada como segue:

KPRIME(100)/ 45233, 45247, 45259, 45263, 45281, 45289, 45293, 45307, 45317, 45319, 45329, 45337, 45341, 45343, 45361, 45377, 45389, 45403, 45413, 45427, 45433, 45439, 45481, 45491, 45497, 45503, 45523, 45533, 45541, 45553, 45557, 45569, 45587, 45589, 45599, 45613, 45631, 45641, 45659, 45667, 45673, 45677, 45691, 45697, 45707, 45737, 45751, 45757, 45763, 45767, 45779, 45817, 45821, 45823, 45827, 45833, 45841, 45853, 45863, 45869, 45887, 45893, 45943, 45949, 45953, 45959, 45971, 45979, 45989, 46021, 46027, 46049, 46051, 46061, 46073, 46091, 46093, 46099, 46103, 46133, 46141, 46147, 46153, 46171, 46181, 46183, 46187, 46199, 46219, 46229, 46237, 46261, 46271, 46273, 46279, 46301, 46307, 46309, 46327, 46337/.

É também necessário durante a instalação, gerar uma lista correspondente $Im_1, Im_2, Im_3, \cdots$ satisfazendo $Im_i \times m_1 \times m_2 \times \cdots \times m_{i-1} = 1 \bmod m_i$, $i = 2, 3, 4, \cdots$ e armazená-la no mesmo bloco comum, *PRIMEB*. O Algoritmo Estendido de Euclides descrito na seção 2.2.1, do capítulo 2, pode ser usado para computar Im_i,

isto é, *IPRIME(i)*. Veja a seguir:

IPRIME(100)/ 00000, 42015, 28577, 01108, 29342, 16641, 10405, 19447, 26685, 39525, 14116, 12753, 32178, 01043, 08857, 27911, 15049, 07079, 33425, 00804, 23175, 23886, 44779, 41942, 10171, 16606, 10638, 17371, 27195, 35827, 42639, 01829, 24558, 09023, 37958, 30638, 06339, 41270, 40538, 10157, 11783, 00457, 32947, 42170, 17910, 33474, 20017, 25086, 36508, 37444, 35543, 06993, 10326, 16328, 26765, 42083, 37223, 30711, 09408, 06635, 38421, 11397, 32683, 17333, 34245, 15748, 35735, 23492, 19302, 20076, 45620, 44978, 09864, 14832, 16092, 19457, 24045, 44950, 32872, 24309, 15726, 43057, 37766, 14046, 41826, 19946, 41363, 23967, 39791, 29237, 18085, 12952, 36850, 02213, 30023, 34871, 42667, 40410, 32615, 46136/.

4.5 Observações sobre Implementação

4.5.1 Estudo de Caso 1

Na implementação do *Algoritmo 406 - EXACT*:

1. Privilegiamos o processamento de matrizes por colunas.

2. Os módulos usados no teste são os números primos: 2, 3, $\cdots$, 113.

3. O programa retorna um código de erro *IER*:

   (a) 0 se o sistema é satisfatoriamente solucionável e apresenta a solução.

   (b) 1 se não existem módulos suficientes para avaliar a solução do sistema.

(c) 2 se a matriz A dos coeficientes é singular (det $= 0$), e $\mathbf{x}$ e $\mathbf{y}$ não são computados.

4. Permite apenas uma entrada de precisão simples.

5. Diagonaliza A: método de Eliminação de Gauss-Jordan - algoritmo 5, visto no capítulo 3.

6. Dispensa maus primos, isto é, primos pelos quais $\mid \det A \mid_{m_i} = 0$. Não pode pre-computar a lista *IPRIME*, nem o número de primos necessários para determinar unicamente a solução.

7. Usa a Desigualdade de Hadamard para obter um limite *BOUND*, para d e $\mathbf{y}$ de maneira que:

$$BOUND \geq \prod_{i=1}^{n} \left(\sum_{j=1}^{n} a_{ij}^2 \right)^{\frac{1}{2}} \prod_{k=1}^{n} \mid b_k \mid .$$

8. Um menor inteiro *IS* é usado tal que,

$$BOUND \leq [m_1 \times m_2 \times \cdots \times m_{IS} - 1]/2.$$

4.5.2 Estudo de Caso 2

Na implementação do *Algoritmo 522 - ESOLVE*:

1. Privilegiamos o processamento de matrizes por colunas.

2. Os módulos usados no teste são os vistos na seção 4.4.2.

3. O programa retorna um código de erro *IER*:

(a) 0 se $\mathbf{y}/d$ é a solução para $A\mathbf{x} = \mathbf{b}$.

(b) 1 se os coeficientes *MAXPRM* na representação de base mista de d e $\mathbf{y}$ terem sido computados. Já que o usuário escolhe o valor de *MAXPRM*, o programa não pode garantir que $\mathbf{y}/d$ é a solução para $A\mathbf{x} = \mathbf{b}$.

(c) 2 se A é singular.

(d) 3 se A é singular, ou d e $\mathbf{y}$ são múltiplos diferentes de $PRIME(1) \times \cdots \times PRIME(MAXPRM)$.

(e) 4 se os parâmetros de entrada são incorretos.

4. Permite entrada de múltipla precisão.

5. Triangulariza A: método de Eliminação de Gauss - algoritmo 4, visto no capítulo 3.

6. É capaz de reter todos os primos.

7. Usa a Desigualdade de Hadamard para obter um limite - *BOUND* mais rigoroso, para d e $\mathbf{y}$, de maneira que:

$$BOUND \geq \begin{cases} \prod_{j=1}^{n} \left(\sum_{i=1}^{n} a_{ij}^2\right)^{\frac{1}{2}}, \bar{b} \leq \bar{a} \\ \prod_{j=1}^{n} \left(\sum_{i=1}^{n} a_{ij}^2\right)^{\frac{1}{2}} \bar{b}/\bar{a}, \bar{b} > \bar{a}, \end{cases}$$

$$\text{em que } \bar{a} = \min_{1 \leq j \leq n} \left\{ \left(\sum_{i=1}^{n} a_{ij}^2 \right)^{\frac{1}{2}} \right\}, \bar{b} = \max_{1 \leq l \leq m} \left\{ \left(\sum_{k=1}^{n} b_{kl}^2 \right)^{\frac{1}{2}} \right\}.$$

8. Um menor inteiro *MAXPRM* é usado tal que,

$$BOUND \leq [m_1 \times m_2 \times \cdots \times m_{MAXPRM} - 1]/2.$$

4.6 Processamento de Matrizes por Coluna

Na implementação dos algoritmos foi utilizada a linguagem de programação Fortran. Nas operações entre matrizes e vetores optou-se pelo processamento por coluna sempre que possível, já que em Fortran uma matriz é armazenada por coluna; isso significa que os elementos consecutivos de uma coluna ocupam posições consecutivas de memória e os elementos consecutivos de uma linha ficam a uma distância igual a um múltiplo do número de linhas que for declarado para o conjunto que armazena a matriz. Por exemplo, se o conjunto A for declarado com L linhas, a distância entre os elementos consecutivos de uma linha ficam separados por um múltiplo de L (se 4 bytes formam uma palavra como nos microcomputadores, a distância será de $4L$ bytes em precisão simples e de $8L$ bytes em precisão dupla).

4.7 Resultados dos Testes

Apresentamos a seguir, os resultados dos testes realizados com os Estudo de Casos 1 e 2, e os métodos de Eliminação de Gauss e Gauss-Jordan.

Tabela 4.1: **Soluções de Sistemas Lineares**
Exemplo 4.1 – Matriz Banda Simétrica

N° de Equações *n*	**Gauss** Maior Raiz	**Gauss - Jordan** Maior Raiz	**ESOLVE: Gauss Aritmética Residual** Raízes Exatas	**EXACT: Gauss - Jordan Aritmética Residual** Raízes Exatas
05	0,500.000.000.000.000.00	0,499.999.992.549.419.79	3/6	3/6
10	0,454.545.454.545.454.58	0,454.454.480.301.413.28	5/11	5/11
20	0,476.190.476.190.476.22	0,476.190.494.676.656.97	10/21	10/21
40	0,487.804.878.048.780.31	0,487.804.853.124.571.44	20/41	20/41
50	0,490.196.078.431.372.69	0,490.196.022.778.526.91	25/51	25/51
80	0,493.827.160.493.827.24	0,493.827.556.256.077.59	40/81	40/81
100	0,495.049.504.950.494.93	0,495049.508.148.907.63	50/101	50/101
150	0,496.688.741.721.854.45	0,496.688.762.982.165.18	75/151	75/151

Teste realizado no IBM PowerPC, utilizando o compilador *Fortran77*, para o ambiente AIX, versão 4.1. Para exibir os erros, foi necessário utilizar precisão dupla.

Tabela 4.2: **Erro Absoluto × Número de Equações**
Exemplo 4.1 – Matriz Banda Simétrica

N° de Equações n	**Gauss** Tolerância: $\times 10^{-20}$	**Gauss-Jordan** Tolerância: $\times 10^{-20}$	**ESOLVE: Gauss** **Aritmética Residual**	**EXACT: Gauss - Jordan** **Aritmética Residual**
05	0	$0{,}745.058.021.000.010.000 \times 10^{-8}$	0	0
10	$0{,}345.456.0 \times 10^{-16}$	$0{,}909.742.440.412.654.544 \times 10^{-4}$	0	0
20	$0{,}295.238.0 \times 10^{-16}$	$0{,}184.861.807.795.238.000 \times 10^{-7}$	0	0
40	$0{,}177.805.0 \times 10^{-15}$	$0{,}249.242.090.478.050.000 \times 10^{-7}$	0	0
50	$0{,}140.980.5 \times 10^{-15}$	$0{,}556.528.456.390.195.000 \times 10^{-7}$	0	0
80	$0{,}795.064.0 \times 10^{-16}$	$0{,}395.762.250.429.506.400 \times 10^{-6}$	0	0
100	$0{,}119.504.8 \times 10^{-15}$	$0{,}319.841.258.049.520.000 \times 10^{-8}$	0	0
150	$0{,}145.364.4 \times 10^{-15}$	$0{,}212.603.108.753.644.000 \times 10^{-7}$	0	0

Os cálculos realizados na tolerância de 10^{-20} foram feitos utilizando a calculadora *CALCON*.

Tabela 4.3: **Tempo de Execução, em segundos, com tolerância de 10^{-3}.**
Exemplo 4.1 – Matriz Banda Simétrica

N° de Equações n	Gauss	Gauss-Jordan	ESOLVE : Gauss Aritmética Residual	MAXPRM	EXACT: Gauss - Jordan Aritmética Residual	PRIMOS
05	30	31	31	2	41	2
10	31	32	36	2	43	2
20	43	45	63	3	69	3
40	85	98	197	5	287	5
50	108	149	315	6	479	6
80	226	431	851	8	1281	5
100	340	659	1460	10	2330	5
150	710	1666	3960	14	7213	5

Teste realizado no IBM PowerPC, utilizando o compilador *Fortran77*, para o ambiente AIX, versão 4.1.

Tabela 4.4: **Soluções de Sistemas Lineares**
Exemplo 4.2 – Matriz Simétrica

N° de Equações *n*	**Gauss** Maior Raiz	**Gauss - Jordan** Maior Raiz	**ESOLVE: Gauss Aritmética Residual** Raízes Exatas	**EXACT: Gauss - Jordan Aritmética Residual** Raízes Exatas
05	1,000.000.000.000.000.22	1,000.000.034.922.425.17	1	1
10	1,000.000.000.000.000.67	1,000.000.034.807.036.59	1	1
20	1,000.000.000.000.002.22	1,000.000.067.316.341.83	1	1
30	1,000.000.000.000.004.00	1,000.000.071.219.957.26	1	1
40	1,000.000.000.000.007.77	1,000.000.063.386.520.03	1	1
50	1,000.000.000.000.006.44	1,000.000.051.204.810.39	1	1
60	1,000.000.000.000.020.21	1,000.000.051.001.561.19	1	1
70	1,000.000.000.000.011.77	1,000.000.082.962.157.58	1	1
80	1,000.000.000.000.021.98	1,000.000.091.474.371.63	1	1
90	1,000.000.000.000.047.07	1,000.000.060.885.111.17	1	1
100	1,000.000.000.000.093.92	1,000.000.055.568.823.54	1	1
120	1,000.000.000.000.080.82	1,000.000.073.361.672.29	1	1
140	1,000.000.000.000.136.56	1,000.000.057.224.541.99	1	1
200	1,000.000.000.000.220.49	1,000.000.070.680.627.57	-	1

Tabela 4.5: **Erro Absoluto × Número de Equações**
Exemplo 4.2 – Matriz Simétrica

N° de Equações n	**Gauss** Tolerância: $\times 10^{-20}$	**Gauss-Jordan** Tolerância: $\times 10^{-20}$	**ESOLVE: Gauss Aritmética Residual**	**EXACT: Gauss - Jordan Aritmética Residual**
05	$0{,}220.000 \times 10^{-15}$	$0{,}349.224.251.7 \times 10^{-7}$	0	0
10	$0{,}670.000 \times 10^{-15}$	$0{,}348.070.365.9 \times 10^{-7}$	0	0
20	$0{,}222.000 \times 10^{-14}$	$0{,}673.163.418.3 \times 10^{-7}$	0	0
30	$0{,}400.000 \times 10^{-14}$	$0{,}712.199.572.6 \times 10^{-7}$	0	0
40	$0{,}777.000 \times 10^{-14}$	$0{,}633.865.200.3 \times 10^{-7}$	0	0
50	$0{,}644.000 \times 10^{-14}$	$0{,}512.048.103.9 \times 10^{-7}$	0	0
60	$0{,}202.100 \times 10^{-13}$	$0{,}510.015.611.9 \times 10^{-7}$	0	0
70	$0{,}117.700 \times 10^{-13}$	$0{,}829.621.575.8 \times 10^{-7}$	0	0
80	$0{,}219.800 \times 10^{-13}$	$0{,}914.743.716.3 \times 10^{-7}$	0	0
90	$0{,}470.700 \times 10^{-13}$	$0{,}608.851.111.7 \times 10^{-7}$	0	0
100	$0{,}939.200 \times 10^{-13}$	$0{,}555.688.235.4 \times 10^{-7}$	0	0
120	$0{,}808.200 \times 10^{-13}$	$0{,}733.616.722.9 \times 10^{-7}$	0	0
140	$0{,}136.560 \times 10^{-12}$	$0{,}572.245.419.9 \times 10^{-7}$	0	0
200	$0{,}220.490 \times 10^{-12}$	$0{,}706.806.275.7 \times 10^{-7}$	-	0

Tabela 4.6: **Tempo de Execução, em segundos, com tolerância de 10^{-3}.**
Exemplo 4.2 – Matriz Simétrica

N° de Equações n	Gauss	Gauss-Jordan	ESOLVE : Gauss Aritmética Residual	MAXPRM	EXACT: Gauss - Jordan Aritmética Residual	PRIMOS
05	27	27	33	3	41	3
10	33	35	41	5	57	5
20	39	50	79	10	92	6
30	46	76	130	16	220	6
40	50	97	221	23	342	6
50	55	140	332	29	433	6
60	62	210	452	36	624	5
70	71	280	653	44	913	5
80	85	390	876	51	1292	5
90	95	460	1156	59	1782	5
100	110	610	1478	66	2359	5
120	156	961	2305	82	3901	5
140	223	1453	3352	99	5970	5
200	585	5858	-	-	16535	5

Tabela 4.7: **Soluções de Sistemas Lineares**
Exemplo 4.3 – Matriz Pascal

Métodos \ Raízes	x_1	x_2	x_3	x_4	x_5	x_6
Gauss – Jordan	−1,000.001.142.382.936.03	5,999.997.219.403.300.44	−15,000.068.430.268.607.1	19,999.939.154.800.564.7	−14,999.977.559.053.032.6	5,999.993.617.392.079.59
Gauss	− 0,999.999.999.998.554.93	5,999.999.999.993.614.89	−14,999.999.999.988.686.4	19,999.999.999.989.924.5	−14,999.999.999.995.480.9	5,999.999.999.999.182.88
ESOLVE Gauss Aritmética Residual	−1	6	−15	20	−15	6
EXACT Gauss – Jordan Aritmética Residual	−1	6	−15	20	−15	6

Tabela 4.8: **Erro Absoluto**
Exemplo 4.3 – Matriz Pascal

Métodos \ Raízes	x_1	x_2	x_3	x_4	x_5	x_6
Gauss – Jordan	$0{,}114.238.293.603 \times 10^{-5}$	$0{,}278.059.669.956 \times 10^{-5}$	$0{,}684.302.686.071 \times 10^{-4}$	$0{,}608.451.994.353 \times 10^{-4}$	$0{,}224.409.469.674 \times 10^{-4}$	$0{,}638.260.792.041 \times 10^{-5}$
Gauss	$0{,}144.507 \times 10^{-11}$	$0{,}638.511 \times 10^{-11}$	$0{,}113.136 \times 10^{-10}$	$0{,}100.755 \times 10^{-10}$	$0{,}451.910 \times 10^{-11}$	$0{,}200.008.171.26 \times 10^{-7}$
ESOLVE Gauss Aritmética Residual	0	0	0	0	0	0
EXACT Gauss – Jordan Aritmética Residual	0	0	0	0	0	0

Tabela 4.9: Tempo de Execução, em segundos, com tolerância de 10^{-3}.
Exemplo 4.3 – Matriz Pascal

Gauss	Gauss - Jordan	ESOLVE: Gauss Aritmética Residual MAXPRM: 4	EXACT: Gauss -Jordan Aritmética Residual PRIMOS: 4
29	30	33	37

Foram analisadas nos testes, três tipos de matrizes: a *Matriz Banda Simétrica*, em que resolvemos o sistema linear $A\mathbf{x} = \mathbf{b}$, para $n = 150$ equações. Com os métodos diretos (Eliminação de Gauss e Gauss-Jordan), as soluções foram aproximadas (ver Tabela 4.1), cujos erros absolutos cometidos se encontram na Tabela 4.2. Com os algoritmos 406 (EXACT) e 522 (ESOLVE), utilizando a Aritmética Residual, verificou-se que a solução foi exata nos racionais. Com a *Matriz Simétrica*, resolvemos o sistema linear $A\mathbf{x} = \mathbf{b}$ até $n = 200$ equações, com exceção de ESOLVE (até $n = 140$ equações). Para Gauss e Gauss-Jordan as soluções foram aproximadas (ver Tabela 4.4), seus respectivos erros absolutos estão na Tabela 4.5. Em ESOLVE e EXACT as soluções foram exatas (nos inteiros). Em ESOLVE, para $n > 140$, não foi possível encontrar a solução esperada, devido a lista de primos conter um total de 100 números (KPRIME = 100). Podemos solucionar esse problema ampliando essa lista, ou seja, KPRIME > 100. Nossa última matriz analisada, foi a *Matriz de Pascal* (mal-condicionada) para $n = 6$. Verificamos que com os métodos diretos as soluções foram aproximadas (ver Tabela 4.7), os erros absolutos se encontram na Tabela 4.8. Em ESOLVE e EXACT as soluções foram exatas (nos inteiros).

Capítulo 5

Considerações Finais

Após estudar os métodos diretos com a aplicação da Aritmética Residual, chegamos as conclusões a seguir:

- ESOLVE [CABAY & LAM, 1977], possui código fonte mais legível, visto que se encontra melhor estruturado, uma vez que EXACT [HOWELL, 1971], não foi escrito levando em consideração essa metodologia de programação;

- ESOLVE utiliza a Eliminação de Gauss (triangularização), uma vez que EXACT usa Gauss-Jordan (diagonalização). O método de Eliminação de Gauss, nas operações de multiplicação é proporcional a $O(n^4\frac{\delta R}{\omega})$ e Gauss-Jordan a $O(\frac{3}{2}n^4\frac{\delta R}{\omega})$, veja [BAREISS, 1972];

- EXACT dispensa maus primos, isto é, primos pelos quais $|\det A|_{m_i} = 0$. Desde que $\mathbf{y} = A^{adj}\mathbf{b}$ possa ainda ser encontrado neste caso [23], ESOLVE é capaz de reter todos os primos. EXACT não pode precomputar a lista *IPRIME* (lista de

[23] Veja [SHAPIRO, 1963] ou [CABAY, 1971].

número primos), nem o número de primos requeridos para determinar unicamente a solução;

- Tanto ESOLVE quanto EXACT usam a Desigualdade de Hadamard, para obter um limite para d e $\mathbf{y}$. Vale salientar que em ESOLVE é usada de forma mais rigorosa;

- EXACT e ESOLVE (se RTEST = *.FALSE.*), resolvem o sistema $|A\overline{\mathbf{x}}|_{m_i} = |\mathbf{b}|_{m_i}$, para todo i, $1 \leq i \leq MAXPRM$, em que $MAXPRM$ é o número máximo de primos para representar d e $\mathbf{y}$ na forma de base mista. Por outro lado, se no ESOLVE, RTEST = *.TRUE.*, continua a repetir somente até um ou outro MAXPRM ou até os coeficientes consecutivos zero - TR na representação de base mista de d e $\mathbf{y}$ serem encontrados, o que acontecer primeiro.

Para resolver um sistema linear não singular, utlizando a Aritmética Residual, os elementos da matriz ampliada do sistema linear devem ser número inteiros. A vantagem da utilização da Aritmética Residual em relação aos métodos diretos é a ausência de divisões com resultados não exatos, implicando na eliminação dos erros de arrendondamento. Isso favorece a aplicação da Aritmética Residual na obtenção da solução exata do sistema linear, veja Tabelas 4.1, 4.4 e 4.7.

Em nossos estudos, observamos que a Aritmética Residual apresenta duas dificuldades:

- O resíduo de uma operação de divisão não é bem definido: para superarmos esta dificuldade, evita-se a execução da operação de divisão trabalhando no campo dos racionais ou algoritmicamente evitando a operação de divisão.

- Tendo o resultado de uma operação em Aritmética Residual usando um módulo, não é possível recuperar univocamente o resultado da mesma operação em aritmética inteira com os dados originais: para evitarmos esta dificuladade, precisamos trabalhar com dois ou mais módulos.

Segundo Chung e Yau [24] "*A Aritmética Residual tem a vantagem de não gerar transporte (carry-free) nas operações de adição, subtração e multiplicação. Desta maneira usando o Sistema de Número Residual podemos, em princípio, aumentar a velocidade das operações no computador*".

Observamos que a Aritmética Residual tem algumas restrições: os elementos da matriz ampliada devem ser inteiros e o tempo de processamento é maior que o tempo obtido pelos métodos diretos: Eliminação de Gauss e Gauss-Jordan (veja Tabelas: 4.3, 4.6 e 4.9).

Neste sentido, verificamos que a Aritmética Residual é um método viável para resolver Sistemas de Equações Lineares corretamente, sendo este fato de grande importância para os profissionais envolvidos em ciência e tecnologia, apesar das restrições vistas anteriormente.

Como proposta para trabalhos futuros, sugerimos:

- A aplicação da Aritmética Residual para Sistema de Equações Lineares Superdeterminados e a expansão dos algoritmos 406 e 522, para resolver sistemas lineares acima de 200 equações.

---

[24] Veja [SODERSTRAND & JENKINS, 1986].

REFERÊNCIAS

ALBRECHT, Peter. *Análise Numérica: Um Curso Moderno*. Rio de Janeiro: Livros Técnicos e Científicos. Pontifícia Universidade Católica do Rio de Janeiro, 1973.

BAREISS, Erwin H. *Computacional Solutions of Matrix Problems Over na Integral Domais*. J. Inst. Maths. Applics. V. 10, p. 68-104, 1972.

BOLDRINI, José Luiz et alli. *Álgebra Linear*. 3ª ed. São Paulo: Harper & Rom. Do Brasil, 1980.

CABAY, S. *Exact Solution of Linear Equations*. Proc. Second Symp. pm Symbolic and Algebraic Manipulation. AC, New York, p. 392-398, 1971.

CABAY, S. and LAM, T.P.L. *Algorithm 522, ESOLVE: Congruence Techniques for the Exact Solution of Integer Systems of Linear Equations*. ACM Trans. Math. Software 3, V. 4, p. 404-410, dec/1977.

CARVALHO, Kíssia. Computação Algébrica: Sistemas de Software, Estrutura e Algoritmo de Riseh. (Dissertação de Mestrado) Universidade Federal da Paraíba, Campina Grande, 1996.

COSCHI, G. and SCHUELER, J.B. *Watfor 77 User's Guide for the IBM PC with DOS*. Waterloo, Canadá: Watcom Publication Limited, 1989.

COSTA, Adeilton Fernandes da. Estudo e Implementação de Métodos Diretos para Solução Exata de Sistemas Lineares. (Dissertação de Mestrado) Universidade Federal da Paraíba, Campina Grande, 1998.

DONGARRA, J.J. at alli. *LINPACK USER'S Guide*. Philadelphia: Siam, 1979.

FIGUEIREDO, Mário Augusto B. de. Um pacote de Aritmética de Múltipla Precisão – CALCON – Calculadora On-Line. (Dissertação de Mestrado) Universidade Federal da Paraíba, Campina Grande, 1989.

GOLUB, Gene H. and VAN LOAN, Charles F. *Matriz Computation*. 2 th ed. Baltimore: Johns University Press, 1989.

GRADSTEYN, I.S: and RYZHIK, I.M. *Tables of Intetgrals, Séries and Products*. 4 th ed. San Diego, CA: Academic Press, 1979.

HEHL, Maximilian Emil. *Linguagem de Programação Estruturada Fortran 77*. São Paulo: McGraw Hill, 1986.

HOSTETTER, Gene H. and MOHAMMED, S. et alli. *Analytical, Numerical, and Computational Methods for Science and Engineering*. EnglewoodCliffs: Prentica-hall, 1991.

HOWELL, Jo and GREGORY, R.T. *An Algarithm for Solving Linear Algebraic Equations Using Residue Arithmetic* I. BIT 9. p. 200-224, março/ 1969.

_____. *An Algarithm for Solving Linear Algebraic Equations Using Residue Arithmetic II*. BIT 9. p. 324-337, abril/ 1969.

_____. *An Algarithm for Solving Linear Algebraic Equations Using Residue Arithmetic II*. BIT 10. p. 23-27, jan./ 1970.

HOWELL, Jo Ann. *Algorithm 406: Exact Solution of Linear Equations Using Residue Arithmetic*. Communications of the ACM. V. 14, p. 180-184, março/1971.

HATTORI, Mário Toyotaro. *Software para Computação Numérica*. Relatório Técnico, Departamento de Sistemas e Computação. Universidade Estadual de Paraíba, Campus II, Campina Grande, 1992.

_____. Mário T. Solução de Problemas de Matemática usando o Computador. Relatório Técnico. Departamento de sistemas e Computação. Universidade Federal da Paraíba, Campina Grande, 1992.

KNUTH, D.E. *The Art of Computer Programming*. V. 1: Fundamental Algorithms. Addison – Wesley, Reading, Mass., 1968.

_____. *The Art of Computer Programming*. V. 2: Seminumerical Algorithms. Addison – Wesley, Reading, Mass., 1969.

KOOGAN, Abrahão & HONAISS, Antônio. *Enciclopédia e Dicionário Ilustrado*. Rio de Janeiro: Edições Delta, 1992.

LINDAMODD, G.E. *Numeral Analysis in Residue Number Systems*. University of Maryland Computer Science Center Report, TR 64-7, College Park: Maryland, 1964.

LUDOVICE, Dr. *Algoritmo para Solução de Sistema Linear pelo Método de Gauss-Jordan*. Georgia Institute of Technology, Atlanta, 1998.

http://www.chemse.gatech.edu/chezzio/proj595/hwsol4.html.

NETO, Veríssimo. *Cálculo Numérico.* 2ª ed. Recife, 1979.

NEWMAN, Morris. *Solving Equation Exactly.* Jour. Res. N. B. S. , 71 B, nº 4, p. 171-179, oct – dec/1967.

PATEL, Vidal A. *Numerical Analysis for Werth.* Sauders College, 1994.

PEREIRA FILHO, Jorge da Cunha & LOYOLA, Caetano Roberto A. *Fortran Anci – 77 e Watfiv* – S: Um texto Universitário. Rio de Janeiro: Editora Campus, 1987.

RALSTON, A. *A First Course in Numerical Analysis.* New York: McGraw-Hill. Book Company, 1965.

RICE, F. R. *Annoucement and Call for Papers.* Mathematical Software. SIGNUM Newsletter. V. 4, p. 7, 1969.

ROSEN, Kenneth H. *Elementary Number Theory and Its Applications.* Canadá: Addison Wesley, 1988.

RUGGIERO, Márcia A. Gomes e LOPES, Vera Lúcia de Rocha. *Cálculo Numérico: Aspectos Teóricos e Computacionais.* 2ª ed. São Paulo: MAKRON Books, 1996.

SHAPIRO, G. *Gauss Elimination for Singular Matrices.* Math. Comput. V. 17, p. 441-445, 1963.

SODERSTRAND, Michael and JENKINS, Kenneth W. et alli. *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing.* New York: IEEE Press, 1986.

SZABÓ, S. and TANAKA, R. *Residue Arithmetic Applications to Computer Technology.* New York: McGraw-Hill, 1967.

VISWANATHAN, T. M. *Introdução a Álgebra e a Aritmética.* Instituto de Matemática Pura e Aplicada. Rio de Janeiro, 1979.

YOUNG, David M. and GREGORY, Robert T. *A Survey of Numerical Mathematics.* V. 2. p. 779-886, Reading, Mass.: Addison – Wesley, 1972.

WILKINSON, J. H. *Rounding Errors in Algebraic Processes.* Englewood Cliffs: PrenticeHall, 1963.