Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Sistemas e Computação
Programa de Educação Tutorial (PET)

A computação aplicada à resolução de sistemas lineares

Izabela Vanessa
(izabela@dsc.ufcg.edu.br)

Agenda

- Motivação
- Aplicações da Matemática
- Sistemas Lineares
- Resolução de Sistemas Lineares
  - Método de Gauss
  - Método de Jacobi
- Considerações Finais
- Bibliografia

Motivação

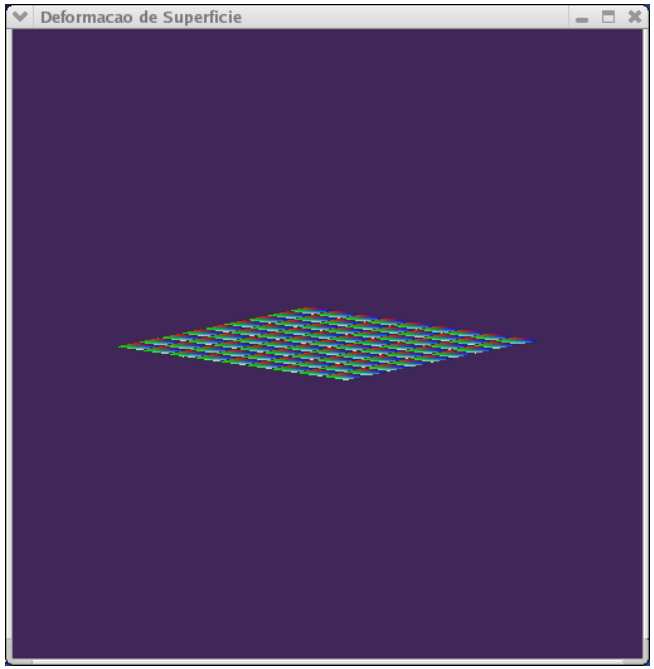
- "Qual a importância da Matemática num curso de Ciência da Computação?"
- "Para que estudar equações diferenciais?"
- "Não vou usar sistemas lineares no meu curso!"
- "Não quero saber de integral e somatório!"
- "Probabilidade? Para quê?"

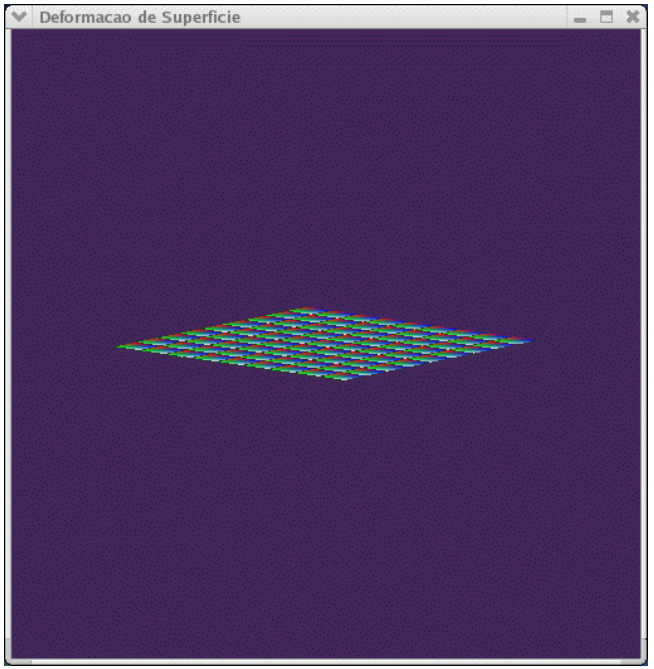
Aplicações da Matemática

- Sistema binário
- Lógica Matemática
  - PROLOG
- Métodos Estatísticos
  - Análise de algoritmos
- Matrizes
  - Transformações 3D
- Matemática discreta
  - Teoria da Computação

Aplicações da Matemática

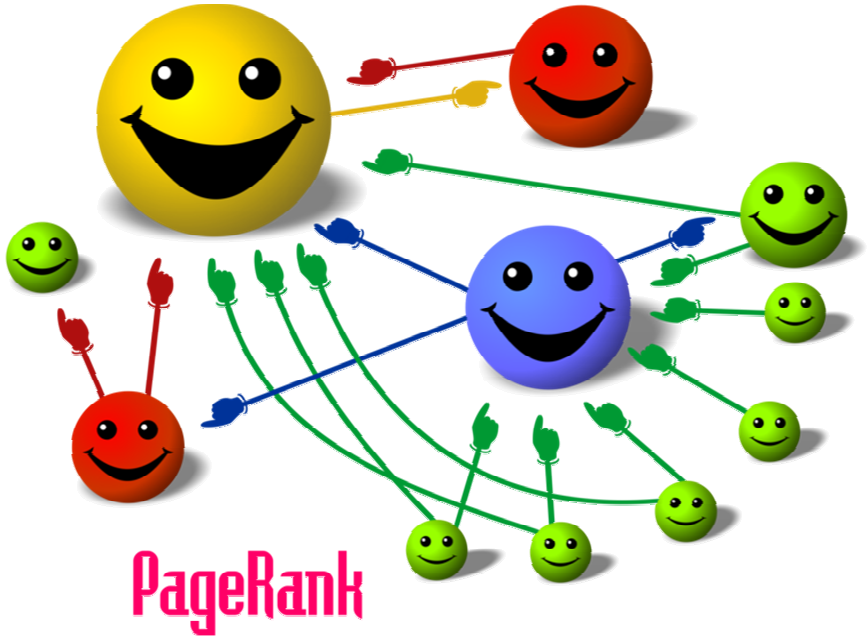
- Sistemas Lineares
  - Interpolação de cores RGB
  - Caimento de um pano sobre uma mesa





Aplicações da Matemática

- Sistemas Lineares
  - Determinação do PageRank



Introdução a sistemas lineares

- Sistema linear Sn de *m* equações com *n* incógnitas

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

- A forma matricial Sn pode ser escrita como AX = b:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_m \end{bmatrix}$$

- Foco do seminário: Matrizes *N* x *N*

Introdução a sistemas lineares

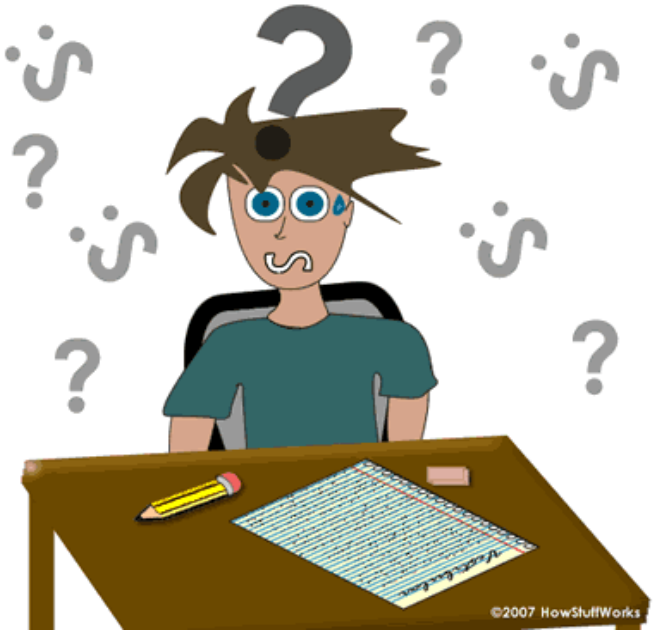
- Resolver o sistema:

$$\begin{cases} x + y = 1 \\ x - y = 3 \end{cases}$$

Introdução a sistemas lineares

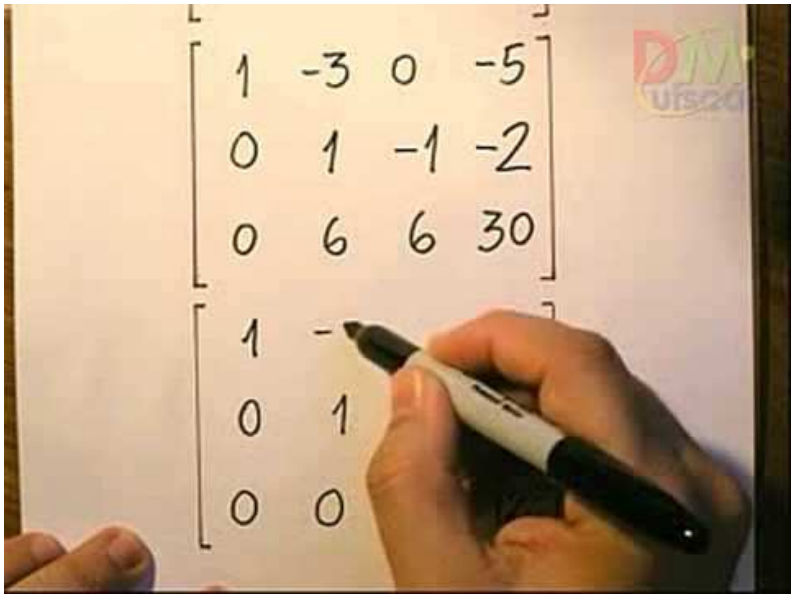
- Resolver o sistema:

$$\begin{cases} 8{,}7x + 3y + 9{,}3z + 11{,}0w = 16{,}4 \\ 24{,}5x - 8{,}8y + 11{,}5z - 45{,}1w = -49{,}7 \\ 52{,}3x - 84{,}0y - 23{,}5z + 11{,}4w = -80{,}8 \\ 21{,}0x - 81{,}0y - 13{,}2z + 21{,}5w = -106{,}3 \end{cases}$$



Introdução a sistemas lineares

Resolução de sistemas lineares



DSC/CEEI/UFCG

PET computação

Resoluções numéricas

- Dois caminhos:
  - Métodos diretos
    - Determinam a solução de um sistema linear com um número finito de operações
  - Métodos iterativos
    - Determinam a solução de um sistema linear com um número não determinado de operações

Método de Gauss

- Método direto
- Com (n-1) passos, o sistema linear AX = b é transformado num sistema triangular equivalente UX = c
- UX = c é resolvido por substituição retroativa.

Método de Gauss

$$\begin{cases} 2x + 3y - z = 5 \\ 4x + 4y - 3z = 3 \\ 2x - 3y + z = -1 \end{cases}$$

- Após 2 passos:

$$\begin{cases} 2x + 3y - z = 5 \\ -2y - z = -7 \\ 5z = 15 \end{cases}$$

Implementação do Método de Gauss

```
/**
 * Método de Gauss: resolução de sistemas lineares de nxn
 */
public static double[] metodoDeGauss(double[][] sistema){
    double[] resultadoFinal = new double[sistema.length];
    double[][] sistemaTriangularSuperior = sistema;
    for (int i = 0; i <= sistema.length - 2; i++) {
        double[] linhaAtual = sistema[i];
        if(i == 0){
            sistemaTriangularSuperior[i] = linhaAtual;
        }
        for (int j = i + 1; j < sistema.length; j++) {
            double m = - sistema[j][i]/sistema[i][i];
            double[] outraLinha = sistemaTriangularSuperior[j];
            sistemaTriangularSuperior[j] = transformacaoElementar(linhaAtual, outraLinha, m);
        }
    }
    resultadoFinal = substituicaoRetroativa(sistemaTriangularSuperior);
    return resultadoFinal;
}
```

Implementação do Método de Gauss

```
/**
 * Realiza as transformações elementares entre as linhas do sistema.
 * Método auxiliar ao metodoDeGauss().
 */
private static double[] transformacaoElementar(double[] linhaAtual,
                                               double[] outraLinha, double m) {
    double[] resultado = new double[outraLinha.length];
    //linhaAtual * m + outraLinha = resultado
    for (int i = 0; i < resultado.length; i++) {
        resultado[i] = m * linhaAtual[i] + outraLinha[i];
    }
    return resultado;
}
```

Implementação do Método de Gauss

```java
/**
 * Método que resolve um sistema linear triangular superior chamado de
 * Substituição Retroativa.
 */
public static double[] substituicaoRetroativa(double[][] sistema){
    double[] resultadoFinal = new double[sistema.length];
    for (int k = sistema.length - 1; k >= 0 ; k--) {
        double[] equacaoAtual = sistema[k];
        double resultadoEquacao = equacaoAtual[equacaoAtual.length - 1];
        double somatorioCoeficientes = 0;
        int posicao = sistema.length - 1;
        for (int l = equacaoAtual.length - 2; l > k; l--) {
            somatorioCoeficientes += equacaoAtual[l] * resultadoFinal[posicao];
            posicao -= 1;
        }
        resultadoFinal[k] = (resultadoEquacao - somatorioCoeficientes)/equacaoAtual[k];
    }
    return resultadoFinal;
}
```

Demonstração

- Resolver o sistema

$$\begin{cases} 2x + 3y - z = 5 \\ 4x + 4y - 3z = 3 \\ 2x - 3y + z = -1 \end{cases}$$

a partir do Método de Gauss.

Noções básicas para os métodos iterativos

- O sistema

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

pode ser escrito como

$$\begin{cases} x_1 = [b_1 - (a_{12}x_2 + \ldots + a_{1n}x_n)] / a_{11} \\ \ldots \\ x_n = [b_n - (a_{n1}x_1 + \ldots + a_{n,n-1}x_{n-1})] / a_{nn} \end{cases}$$

em que $a_{ii} \neq 0$, para todo i.

Noções básicas para os métodos iterativos

- Esse sistema pode ser colocado na forma

$$X = FX + d$$

em que:

$$X = \begin{pmatrix} x_1 \\ \dots \\ x_n \end{pmatrix} \qquad d = \begin{pmatrix} b_1/a_{11} \\ \dots \\ b_n/a_{nn} \end{pmatrix} \qquad F = \begin{pmatrix} 0 & -a_{12}/a_{11} & \dots & -a_{1n}/a_{11} \\ -a_{21}/a_{22} & 0 & \dots & -a_{2n}/a_{22} \\ & & \dots & \\ -a_{n}/a_{nn} & & \dots & 0 \end{pmatrix}$$

Método de Jacobi

- Método iterativo
- Funcionamento:
  - Escolher uma aproximação inicial $x^{(0)}$
  - Gerar aproximações sucessivas de $x^{(k)}$ a partir da iteração
    - $X^{(k+1)} = FX^{(k)} + d, \quad k = 0,1,2...$
  - Gerar novas aproximações até que uma das condições abaixo seja satisfeita
    - Máx $|X_i^{(k+1)} - X_i^{(k)}| <= E$, E é a tolerância e $1<= i <=n$
    - K > M, M é o número máximo de iterações

Método de Jacobi

- Exemplo:

$$\begin{cases} 2x - y = 1 \\ x + 2y = 3 \end{cases}$$

com $E <= 10^{-2}$ ou $k > 10$.

Implementação do Método de Jacobi

```
/**
 * Implementação do Método de Jacobi: resolução de sistemas lineares de nxn,
 * com tolerancia t e um máximo de iteracoes k
 */
public static double[] metodoDeJacobi(double[][] sistema, double tolerancia, int iteracoes){
    double[] resultadoFinal = inicializeListaComZeros(sistema.length);
    double[] resultadoParcial = inicializeListaComZeros(sistema.length);
    int numeroIteracoes = 0;
    boolean toleranciaAceitavel = true;
    while(toleranciaAceitavel && numeroIteracoes <= iteracoes){
        for (int i = 0; i < resultadoFinal.length; i++) {
            double[] equacao = sistema[i];
            resultadoFinal[i] = (equacao[equacao.length - 1]
                                 + somatorio(equacao, resultadoParcial, i))/equacao[i];
        }
        resultadoParcial = resultadoFinal;
        numeroIteracoes++;
        toleranciaAceitavel = verificaTolerancia(resultadoFinal, tolerancia);
    }
    return resultadoFinal;
}
```

Implementação do Método de Jacobi

```
/**
 * Realiza o somatório entre xi*ai, para i <= numero de variaveis.
 * Método auxiliar ao metodoDeJacobi().
 */
private static double somatorio(double[] equacao, double[] resultadoParcial,
        int i) {
    double resultado = 0;
    for (int j = 0; j < equacao.length - 1; j++) {
        if(j != i){
            resultado += (- equacao[j])*resultadoParcial[j];
        }
    }
    return resultado;
}
```

Demonstração

- Resolver o sistema

$$\begin{cases} 2x - y = 1 \\ x + 2y = 3 \end{cases}$$

com E <= 10^{-3} ou k > 10 pelo programa do método de Jacobi.

Considerações finais

- Diferentemente do que escutamos pelos corredores do DSC, a matemática tem sim um papel importante para o nosso curso de Ciência da Computação;
- É importante a automatização da resolução de sistemas lineares complexos para evitar erros e/ou perda de tempo.

Dúvidas?

Bibliografia

- BARROSO, Leônidas Conceição; BARROSO, Magali Maria de Araújo; FILHO, Frederico Ferreira Campos; CARVALHO, Márcio Luiz Bunte de; MAIA, Miriam Lourenço. *Cálculo Numérico (com aplicações)*, 2ª edição. 1987 – Editora Harbra Ltda.
- ASANO, Claudio Hirofume; COLLI, Eduardo. *Cálculo Numérico — Fundamentos e Aplicações*. 2007. IME-USP
- http://www.dca.fee.unicamp.br/projects/desmo/. Acesso em 22/04/10.
- http://www.atractor.pt/mat/pagerank/exemplos.htm. Acesso em 13/05/10.
- http://pt.wikipedia.org/wiki/PageRank. Acesso em 13/05/10.