

## Atividade 1: Método dos Mínimos Quadrados

Método dos mínimos quadrados é uma técnica utilizada para aproximar uma curva a um conjunto de pontos, a partir da minimização da diferença dos quadrados para a curva estimada. Este trabalho apresenta uma implementação do método em linguagem Java 1.8, para a disciplina de tópicos especiais em aprendizagem.

### Introdução

Este artigo mostra como a aplicação que implementa o método de mínimos quadrados foi codificada. Para isto, uma breve explicação é feita sobre cada método desenvolvido, que compõe o exercício. O objetivo do trabalho é implementar um método simples que atenda os requisitos da disciplina.

### Método

O método dos mínimos quadrados busca encontrar a curva que melhor se aproxima de um conjunto de pontos distribuídos sobre um plano. Para tanto, o algoritmo busca minimizar a somatória das diferenças entre o valor medido e o valor estimado pela função escolhida:

$$\min \sum (y_i - \hat{y}_i)^2$$

A mesma equação em sua forma matricial pode ser expressada da seguinte forma:

$$\beta = (X^T X)^{-1} X^T y$$

### Matrizes

A forma matricial é mais adequada para a resolução do problema. Com isso, temos que encontrar a matriz inversa em um dos passos da equação:

$$A.A^{-1} = A^{-1}.A = I$$

Neste caso há uma restrição, apenas é possível calcular a matriz inversa para matrizes quadradas. Para isso, a seguinte fórmula deve ser utilizada:

$$A^{-1} = \frac{1}{\det(A)} \text{adj}(A)$$

O determinante da matriz  $\det(A)$  é igual a um escalar que representa a matriz. Já a matriz adjunta  $\text{adj}(A)$  representa a transposta dos cofatores.

Os cofatores são calculados a partir do determinante de uma sub-matriz obtida por meio da eliminação de uma linha e uma coluna da matriz A.

$$\tilde{a}_{ij} = (-1)^{i+j} D_{ij}$$

### Desenvolvimento

O método dos mínimos quadrados foi implementado na linguagem de programação Java em sua versão 8. Diferentes métodos foram criados que em

conjunto implementam o método dos mínimos quadrados. Neste programa são consideradas apenas matrizes de no máximo 3x3.

## Multiplicação de Matrizes

```
public static double[][] multiplicaMatriz(double[][] matriz_1,
double[][] matriz_2, int tamanho){

    double[][] r = new double[tamanho][tamanho];

    double v = 0;

    for (int i = 0; i < tamanho; i++) {

        // for (int j = 0; j < matriz_2[0].length; j++){

        for (int j = 0; j < tamanho; j++){

            for (int k = 0; k < tamanho; k++) {

                v = v + (matriz_1[i][k] * matriz_2[k][j]);

            }

            r[i][j] = v;

            v = 0;

        }

    }

    return r;
}
```

## Calculo da Matriz Adjunta

```
public static double[][] calculaAdjunta(double[][] matriz, int
tamanho) {

    double[][] adjunta = new double[tamanho][tamanho];

    if (tamanho == 2) {

        adjunta[0][0] = matriz[1][1];

        adjunta[0][1] = -matriz[1][0];

        adjunta[1][0] = -matriz[0][1];

        adjunta[1][1] = matriz[0][0];

    } else if (tamanho == 3) {

        double[][] temp = new double[2][2];

        for (int i = 0; i < tamanho; i++) {

            for (int j = 0; j < tamanho; j++) {

                temp[0][0] = matriz[(i + 1) % 3][(j + 1) % 3];

                temp[0][1] = matriz[(i + 1) % 3][(j + 2) % 3];

                temp[1][0] = matriz[(i + 2) % 3][(j + 1) % 3];

                temp[1][1] = matriz[(i + 2) % 3][(j + 2) % 3];

                adjunta[i][j] = calculaDeterminante(temp, 2);

            }

        }

    }

}
```

```

    }

    }

    return calculaTransposta(adjunta, tamanho);
}

```

## Calculo da Matriz Transposta

```

public static double[][] calculaTransposta(double[][] matriz, int
tamanho) {

    double[][] transp = new double[tamanho][tamanho];

    for (int i = 0; i < tamanho; i++) {

        for (int j = 0; j < tamanho; j++) {

            transp[j][i] = matriz[i][j];

        }

    }

    return transp;
}

```

## Calculo da Matriz Inversa

```

public static double[][] calculaInversa(double[][] matriz, int
tamanho) {

    double[][] inversa = new double[tamanho][tamanho];

    double[][] adjunta = calculaAdjunta(matriz, tamanho);

    double determinante = calculaDeterminante(matriz,
tamanho);

    for (int i = 0; i < tamanho; i++){

        for (int j = 0; j < tamanho; j++){

            inversa[j][i] = (1/determinante) * adjunta[i][j];

        }

    }

    return inversa;
}

```

## Calculo do Determinante

```

public static double calculaDeterminante(double[][] matriz, int
tamanho) {

    double det = 0;

    if (tamanho == 2)

        det = (matriz[0][0] * matriz[1][1]) - (matriz[0][1] *

```

```
matriz[1][0]);

if (tamanho == 3) {

    for (int i = 0; i < tamanho; i++) {

        det = det + (matriz[0][i] * (matriz[1][(i + 1) % 3] *

matriz[2][(i + 2) % 3] - matriz[1][(i + 2) % 3] * matriz[2][(i + 1) %

3]));

    }

}

return det;

}
```

do número de variáveis de entrada, fazendo com que uma matriz de ordem maior que 3x3 seja gerada, na qual não foi suportada por esta implementação.

Para os demais, o programa pode ser executado seguindo as instruções de entrada de dados, tendo sucesso em suas execuções.

### Conclusão

A partir do algoritmo implementado foi possível executar os exemplos indicados com sucesso.

- US Census
- Books x Grades
- Boiling point at the Alps
- Height x Shoe size

Para os dados do “Books x Grades”, não foi possível calcular o quadrático por conta