UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CÂMPUS APUCARANA

CURSO DE ENGENHARIA DA COMPUTAÇÃO

VITOR LUIS DE QUEIROZ BATISTA

CÁLCULO DA INVERSA DE UMA MATRIZ

APUCARANA

JULHO, 2019

VITOR LUIS DE QUEIROZ BATISTA

CÁLCULO DA INVERSA DE UMA MATRIZ

Relatório elaborado na disciplina de Geometria Analítica do Curso de Engenharia de Computação do Campus Apucarana da Universidade Tecnológica Federal do Paraná - UTFPR.

Orientador: Ana Paula da Silveira Vargas

APUCARANA

JULHO, 2019

1 INTRODUÇÃO

Foi proposto pela professora de GA-AL o cálculo da matriz inversa pelo método de eliminação de Gauss-Jordan em linguagem C.

Foi indicado que tal matriz deveria ser recebida de um arquivo, depois disso o resultado das operações do programa devia ser gravado em outro arquivo.

1.1 MÉTODO DE ELIMINAÇÃO DE GAUSS-JORDAN

O método de eliminação de Gauss-Jordan segue em uma versão da eliminação de Gauss, sendo dessa forma zerando os elementos que estão acima e abaixo dos elementos chamados de pivots, transformando a matriz em sua forma reduzida por linhas.

No código é obtido a matriz por um arquivo de entrada chamado “matrix.txt”, dobrando o número de colunas da matriz, a fim de adicionar uma matriz identidade no lado esquerdo e assim resolvendo as operações de Gauss-Jordan e escalonando normalmente.

Dessa forma, após o escalonamento do lado esquerdo da matriz, a matriz identidade anteriormente adicionada no código à direita da matriz será manipulada igualmente, obtendo a matriz inversa.

Após isso, a matriz é consequentemente impressa em outro arquivo chamado “saída.txt” também com tabulação igual à matriz obtida por entrada.

2 PROCEDIMENTO EXPERIMENTAL

A plataforma Netbeans foi utilizada para a criação do programa e do código em linguagem C.

O código comentado segue abaixo.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * File:   main.c
 * Author: Vitor
 *
 * Created on 2 de Julho de 2019, 16:15
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

/*
 *
 */

int main(int argc, char** argv) {
    FILE * arquivo = fopen("matrix.txt", "r");
    if (arquivo == NULL)
        exit(-5);

    int numeroCOLUNAS;
```

```
int numeroLINHAS;

float **MATRIX;

float aux;

numeroLINHAS = 0;
numeroCOLUNAS = 0;

while (!feof(arquivo)) {
    aux = fgetc(arquivo);
    if (aux == '\n')
        numeroLINHAS++;
}

char linhaSTR[150];
char *ponteiroL;

rewind(arquivo);

fgets(linhaSTR, 150, arquivo);
ponteiroL = strtok(linhaSTR, " ");
for (;;) {
    double val;
    if (ponteiroL == "\n")
        break;
    if (ponteiroL != NULL)
        numeroCOLUNAS++;
    else
        break;
```

```
    ponteiroL = strtok(NULL, " ");
}

printf("Mij = (%i, %i)", numeroLINHAS, numeroCOLUNAS);

MATRIX = (float **) malloc(numeroLINHAS * sizeof (float *));
for (int i = 0; i < numeroLINHAS; i++) {
    MATRIX[i] = (float *) malloc(2 * numeroCOLUNAS * sizeof (float));
}

rewind(arquivo);

for (int i = 0; i < numeroLINHAS; i++) {
    for (int j = 0; j < numeroCOLUNAS; j++) {
        float valor;
        fscanf(arquivo, "%f", &valor);
        MATRIX[i][j] = valor;
    }
}

rewind(arquivo);

printf("Matriz: \n");
for (int i = 0; i < numeroLINHAS; i++) {
    for (int j = 0; j < numeroCOLUNAS; j++) {
        printf("%f ", MATRIX[i][j]);
    }
    printf("\n");
}
```

```
fclose(arquivo);

printf("\n\nEscolha uma opcao:\n"
       "1-Resolutor GAUSS JORDAN [matriz necessaria: Mij=(x,x+1)]\n"
       "2-Resolutor INVERSA [matriz necessaria: Mij=(x,x)]\n"
       "[DIGITAR ESCOLHA]\n");


int escolha;
scanf("%i", &escolha);
getchar();

switch (escolha) {
   case 1:
   {
      if (numeroCOLUNAS != numeroLINHAS + 1) {
         printf("\na matriz nao e Mij=(x,x+1).\n");
         exit(-2);
      }


   }
   case 2:
   {
      if (numeroCOLUNAS != numeroLINHAS) {
         printf("\na matriz nao e Mij=(x,x).");
         exit(-3);
      }
      /*
      MATRIX = (float*)realloc(MATRIX, numeroLINHAS * sizeof(float));
```

```
        for (int i = 0; i < numeroLINHAS; i++)
            MATRIX[i] = (float *) realloc(MATRIX, numeroCOLUNAS * 2 * sizeof
(float));
         */
        printf("Matriz: \n");
        for (int i = 0; i < numeroLINHAS; i++) {
            for (int j = 0; j < numeroCOLUNAS; j++) {
                printf("%f ", MATRIX[i][j]);
            }
            printf("\n");
        }

        for (int i = 0; i < numeroLINHAS; i++) {
            for (int j = 0; j < numeroCOLUNAS; j++) {
                if (i == j)
                    MATRIX[i][j + numeroCOLUNAS] = 1;
                else
                    MATRIX[i][j + numeroCOLUNAS] = 0;
            }
        }

        printf("Matriz: \n");
        for (int i = 0; i < numeroLINHAS; i++) {
            for (int j = 0; j < 2 * numeroCOLUNAS; j++)
                printf("%f ", MATRIX[i][j]);
            printf("\n");
        }

        double coeficiente;
        for (int i = 0; i < numeroLINHAS; i++) {
```

```
    for (int j = 0; j < numeroCOLUNAS; j++) {
        if (i != j) {
            coeficiente = MATRIX[j][i] / MATRIX[i][i];
            for (int k = 0; k < 2 * numeroCOLUNAS; k++)
                MATRIX[j][k] = MATRIX[j][k] - coeficiente * MATRIX[i][k];
        }
    }
}

printf("Matriz: \n");
for (int i = 0; i < numeroLINHAS; i++) {
    for (int j = 0; j < 2* numeroCOLUNAS; j++)
        printf("%f ", MATRIX[i][j]);
    printf("\n");
}

for (int i = 0; i < numeroLINHAS; i++) {
    coeficiente = MATRIX[i][i];
    for (int j = 0; j < 2 * numeroCOLUNAS; j++)
        MATRIX[i][j] = MATRIX[i][j] / coeficiente;
}

printf("Matriz: \n");
for (int i = 0; i < numeroLINHAS; i++) {
    for (int j = 0; j <2 * numeroCOLUNAS; j++)
        printf("%.3f\t", MATRIX[i][j]);
    printf("\n");
}

FILE * saida = fopen("saida.txt", "w");
```

```
            for (int i = 0; i < numeroLINHAS; i++) {
                for (int j = 0; j < numeroCOLUNAS; j++)
                    fprintf(saida, "%.5f\t", MATRIX[i][j]);
                fprintf(saida, "\n");
            }


        }
    }


    getchar();
    return (EXIT_SUCCESS);
}
```