2° Trabalho de Cálculo Numérico

Profa. Vanessa Rolnik

Guilherme Martiniano de Oliveira - 11215765
Gustavo Fernandes Carneiro de Castro - 11369684
Mateus Miquelino da Silva - 11208412

Item a)

As equações químicas envolvem ao lado esquerdo, os reagentes e, como "resultado", os produtos. Existem casos onde as proporções de reagentes não estão de acordo com os produtos, ou vice-versa. Quando isso ocorre, as substâncias precisam ser "equilibradas". Para isso, os coeficientes estequiométricos* devem ser balanceados, ou seja, devemos seguir os seguintes passos:

1. Fazer o balanceamento** da equação química (corrigir os coeficientes estequiométricos);
2. Fazer a contagem de mol de cada substância;
3. Analisar o que é pedido;
4. Relacionar as grandezas;
5. Calcular a proporção;

*Coeficiente Estequiométrico: Quantos mols de uma substância devem ser utilizados em uma equação química.

**Balanceamento: utilizar as quantidades de cada átomo nas substâncias dos reagentes para obter um resultado equivalente nos produtos. Com isso essas quantidades são vistas matematicamente como soluções de um sistema de equações.

Fontes:
https://www.soq.com.br/conteudos/em/estequiometria/p4.php
https://mundoeducacao.bol.uol.com.br/quimica/equacoes-quimicas.htm

Item b)

Arquivo lido:

```
1  0 0 2 0 0 0 0 0 7
2  0 0 0 0 0 1 0 0 66
3  0 -2 0 0 0 0 0 2 96
4  0 0 0 0 1 0 0 0 42
5  -4 -4 7 4 2 3 4 1 24
6  -1 0 2 0 0 1 2 0 0
7  -1 0 0 1 0 0 0 0 0
8  0 -1 0 1 0 0 1 0 0
```

A matriz lida é a versão estendida, utilizando como separador, espaços (“ “).

Item c)

```c
//Guilherme Martiniano de Oliveira - 11215765
//Gustavo Fernandes Carneiro de Castro - 11369684
//Mateus Miquelino da Silva - 11208412

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

double** readMatrix(double **matrix, int *max);
void printMatrix(double **matrix, int max);
void escalonamento(double **matrix, int linhas, int colunas);
double* answer(double **matrix, int max);
double somatoria(double **matrix, int l, int max);
void printResposta(double *resposta, int max);

const char *endl = "\n";
const char *delim = " ";

int main()
{
    double **matrix;
    int *max; //ponteiro para receber o valor da funcao readMatrix
    double *respostas;

    matrix = readMatrix(matrix, max);

    printf("\nMatriz original estendida: \n");
    printMatrix(matrix, *max);

    escalonamento(matrix, *max, *max + 1);
    printf("\nMatriz escalonada estendida: \n");
    printMatrix(matrix, *max);

    respostas = answer(matrix, *max);
    printf("\nVetor dos valores de x: \n");
    printResposta(respostas, *max);


}

void escalonamento(double **matrix, int linhas, int colunas)
{
    int passo, i, j, flag = 0;
    double m, pivo = 0, aux;

    for(passo = 0; passo < linhas; passo++){ //Cada passo para o processo de escalonamento, ele termina no passo linhas - 1 pois a ultima linha ja estera escalonada;
        pivo = matrix[passo][passo];         //O elemento pivo eh previamento selecionado por ser o primeiro elemento da diagonal principal;

        for(i = passo; i < linhas; i++){
            if(fabs(matrix[i][passo]) > fabs(pivo)){ //O elemento pivo eh comparado com os outros elementos para ser entao o elemento de maior valor absoluto o pivo;
                pivo = matrix[i][passo];
                flag = i; //Indicar qual linha precisa ser permutada/substituida;
            }
        }

        if(pivo != 0){ //O elemento do pivo nao pode ser nulo;
            if(flag != 0){
                for(i = 0; i < colunas; i++){ //Faz a troca das linhas caso o pivo nao seja ja o elemento da diagonal principal;
                    aux = matrix[passo][i];
                    matrix[passo][i] = matrix[flag][i];
                    matrix[flag][i] = aux;
                }
            }

            for(i = passo + 1; i < linhas; i++){ //Formulas de escalonamento de matriz;
                m = matrix[i][passo] / pivo;
                matrix[i][passo] = 0;

                for(j = passo + 1; j < colunas; j++){
                    matrix[i][j] = matrix[i][j] - (m * matrix[passo][j]);
                }
            }
        }
        flag = 0; //Zera o indicador para que todo passo possa ser indicado a posicao do pivo na matriz;
    }
}

double* answer(double **original, int max)
{
    double *respostas; //vetor dos valores de x
    double **matrix; //criado para não alterar a matriz original

    respostas = (double *)malloc(max * sizeof(double));

    matrix = (double **)malloc(max * sizeof(double*));
    for(int i = 0; i < max; i++) matrix[i] = (double *)malloc((max+1) * sizeof(double));

    for(int i = 0; i < max; i++) //copia da matriz original
    {
        for(int j = 0; j < max + 1; j++)
        {
            matrix[i][j] = original[i][j];
        }
    }

```

```c
    for(int i = max - 1; i >= 0; i--) //indo na diagonal principal da matriz do sistema de baixo para cima
    {
        respostas[i] = (matrix[i][max] - somatoria(matrix, i, max)) / matrix[i][i];
        //valores de x = resposta da linha - somatoria das: constantes da linha multiplicados pelos x já encontrados e substituidos, tudo dividido pelo valor atual da diagonal

        for(int j = i - 1; j >= 0; j--) //o valor do x achado acima é multiplicado na coluna inteira de baixo para cima
        {
            matrix[j][i] = matrix[j][i] * respostas[i];
        }
    }

    return respostas;
}


double somatoria(double **matrix, int l, int max)
{
    double soma = 0;
    for(int c = l + 1; c < max; c++) //a somatoria e de todos os valores entre i e a ultima coluna, sem contar ambos
    {
        soma = soma + matrix[l][c];
    }

    return soma;

}


double** readMatrix(double **matrix, int *max) // a funcao apenas le matrizes estendidas
{
    FILE *arq;
    char fileStream[100000];
    int count = 0;

    arq = fopen("matriz.txt", "r");

    if (arq == NULL)
    {
        printf("O arquivo não existe\n");
    }

    for (char c = getc(arq); c != EOF; c = getc(arq)) //contador do numero de linhas do arq
        if (c == '\n')
            count++;
    count++;

    //por ser uma matriz estendida de um sistema, no numero de col = num lin + 1

    rewind(arq);

    fread(fileStream, sizeof(char), 100000, arq);

    matrix = (double **)malloc(count * sizeof(double*));
    for(int i = 0; i < count; i++) matrix[i] = (double *)malloc((count+1) * sizeof(double));


    matrix[0][0] = atoi(strtok(fileStream, delim));

    for(int c = 1; c < count; c++)
        matrix[0][c] = atoi(strtok(NULL, delim));

    matrix[0][count] = atoi(strtok(NULL, endl));


    for(int l = 1; l < count; l++)
    {
        for(int c = 0; c < count; c++)
        {
            matrix[l][c] = atoi(strtok(NULL, delim));
        }
        matrix[l][count] = atoi(strtok(NULL, endl));
    }


    fclose(arq);

    *max = count; //retornando o valor das linhas
    return matrix;

}

void printMatrix(double **matrix, int max)
{

    for(int i = 0; i < max; i++)
    {
        for(int j = 0; j < (max+1); j++)
        {
            printf("%7.4lf ", matrix[i][j]);
        }
        printf("\n");
    }
    printf("\n\n");
}
```

```c
void printResposta(double *resposta, int max)
{
    for(int i = 0; i < max; i++)
        if(i != max - 1){
            printf("X%d=%5.4lf, ",i + 1 , resposta[i]);
        }else{
            printf("X%d=%5.4lf",i + 1 , resposta[i]);
        }
}
```

Item d)

Respostas:

```
Matriz original extendida:
 0.0000  0.0000  2.0000  0.0000  0.0000  0.0000  0.0000  0.0000  7.0000
 0.0000  0.0000  0.0000  0.0000  0.0000  1.0000  0.0000  0.0000 66.0000
 0.0000 -2.0000  0.0000  0.0000  0.0000  0.0000  0.0000  2.0000 96.0000
 0.0000  0.0000  0.0000  0.0000  1.0000  0.0000  0.0000  0.0000 42.0000
-4.0000 -4.0000  7.0000  4.0000  2.0000  3.0000  4.0000  1.0000 24.0000
-1.0000  0.0000  2.0000  0.0000  0.0000  1.0000  2.0000  0.0000  0.0000
-1.0000  0.0000  0.0000  1.0000  0.0000  0.0000  0.0000  0.0000  0.0000
 0.0000 -1.0000  0.0000  1.0000  0.0000  0.0000  1.0000  0.0000  0.0000



Matriz escalonada extendida:
-4.0000 -4.0000  7.0000  4.0000  2.0000  3.0000  4.0000  1.0000 24.0000
 0.0000 -2.0000  0.0000  0.0000  0.0000  0.0000  0.0000  2.0000 96.0000
 0.0000  0.0000  2.0000  0.0000  0.0000  0.0000  0.0000  0.0000  7.0000
 0.0000  0.0000  0.0000 -1.0000 -0.5000  0.2500  1.0000  0.7500 41.1250
 0.0000  0.0000  0.0000  0.0000  1.0000  0.0000  0.0000  0.0000 42.0000
 0.0000  0.0000  0.0000  0.0000  0.0000  1.0000  0.0000  0.0000 66.0000
 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  2.0000 -0.2500 -2.3750
 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.6250 117.4375



Vetor dos valores de x:
X1=117.6000, X2=139.9000, X3=3.5000, X4=117.6000, X5=42.0000, X6=66.0000, X7=22.3000, X8=187.9000
```

Equação química balanceada:

$(Cr(N_2H_4CO)_6)_4(Cr(CN)_6)_3 + 117.6\ KMnO_4 + 139.0\ H_2SO_4 \rightarrow 3.5\ K_2Cr_2O_7 + 117.6\ MnSO_4 + 42.0\ CO_2 + 66.0\ KNO_3 + 22.3\ K_2SO_4 + 187.9\ HO_2$