Universidade de Brasília

Relatório de atividade do módulo 2

Métodos Numéricos para Engenharia

Sistemas de Equações Lineares

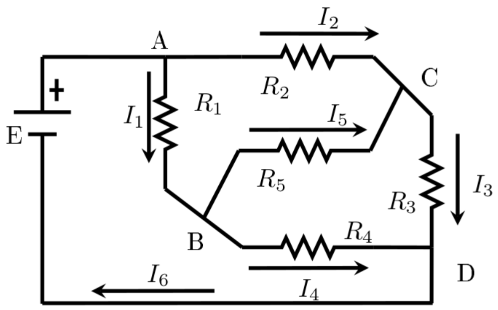
Aluno:
Wilton Rodrigues

Matrícula:
13/0049212

25 de setembro de 2016

1 Introdução

O objetivo deste relatório é exercitar os conceitos aprendidos em aula, com relação ao tópico: Sistemas de Equações Lineares. Que tem como objetivo prover métodos matemáticos capazes de solucionar esses sistemas que aparecem frequentemente em matemática aplicada, economia e na modelagem de fenônemos na engenharia. O problema a ser solucionado é o circuito mostrado abaixo, que é conhecido como Ponte de Wheatstone, e é frequentemente usado em medidas eletrônicas.



Neste circuito, $R_1 = 10\Omega$, $R_2 = R_3 = R_4 = R_5 = 100\Omega$ e $E = 20V$.

Figura 1: Ponte de Wheatstone

De acordo com os dados do problema, as equações que governam o problema são obtidas a partir da Lei de Kirchoff. Sendo assim, para as seguintes malhas temos as seguintes equações:

$$\begin{aligned} Para\ ABD: &\quad I_1R_1 + I_4R_4 - E = 0 \\ Para\ ABCA: &\quad I_1R_1 + I_5R_5 - I_2R_2 = 0 \\ Para\ BCDB: &\quad I_5R_5 + I_3R_3 - I_4R_4 = 0 \\ Para\ o\ A: &\quad I_6 = I_1 + I_2 \\ Para\ o\ B: &\quad I_1 = I_5 + I_4 \\ Para\ o\ C: &\quad I_3 = I_2 + I_5 \end{aligned} \tag{1}$$

O objetivo do trabalho é solucionar o sistema de equações lineares acima para determinar as correntes $I1, I2, ..., I6$ através do método de Gauss-Jordan.

2 Metodologia

A primeira coisa a se fazer é passar o sistema de equação (1) para o formato matricial, sendo assim obtemos a seguinte relação:

$$R * I = E \tag{2}$$

Onde R é a matriz dos resistores, I é o vetor das Correntes e E o vetor das tensões. A partir da relação expressa na equação (2) obtemos a seguinte matriz:

$$\begin{bmatrix} R_1 & R_2 & R_3 & R_4 & R_5 & R_6 \\ R_1 & R_2 & R_3 & R_4 & R_5 & R_6 \\ R_1 & R_2 & R_3 & R_4 & R_5 & R_6 \\ R_1 & R_2 & R_3 & R_4 & R_5 & R_6 \\ R_1 & R_2 & R_3 & R_4 & R_5 & R_6 \\ R_1 & R_2 & R_3 & R_4 & R_5 & R_6 \end{bmatrix} * \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \\ E_6 \end{bmatrix} \tag{3}$$

Baseando-se nos valores iniciais informados na figura (1) para as resistências e a tensão ao fazermos as devidas substituições obtemos a seguinte expressão:

$$\begin{bmatrix} 10 & 0 & 0 & 100 & 0 & 0 \\ 10 & -100 & 0 & 0 & 100 & 0 \\ 0 & 0 & 100 & -100 & 100 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 & -1 & 0 \\ 0 & -1 & 1 & 0 & -1 & 0 \end{bmatrix} * \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{bmatrix} = \begin{bmatrix} 20 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

Pela qual é possível expressar qualquer uma das equações informadas inicialmente no sistema (1). Ao pegarmos a matriz R e o vetor E teremos a matriz aumentada MA:

$$\begin{bmatrix} 10 & 0 & 0 & 100 & 0 & 0 & \vdots & 20 \\ 10 & -100 & 0 & 0 & 100 & 0 & \vdots & 0 \\ 0 & 0 & 100 & -100 & 100 & 0 & \vdots & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 & \vdots & 0 \\ 1 & 0 & 0 & -1 & -1 & 0 & \vdots & 0 \\ 0 & -1 & 1 & 0 & -1 & 0 & \vdots & 0 \end{bmatrix} \tag{5}$$

3 Diagrama esquemático de execução

Nesta seção, encontra-se o fluxo de execução da solução da equação (**??**) utilizando a linguagem C. Que é apresentada na próxima sessão.

A solução elaborada neste relatório funciona da seguinte maneira. Tanto a função, quanto a precisão desejada são inseridas diretamente no código fonte. Apenas o intervalo no qual se deseja verificar a existencia da raíz é solicitado ao usuário em tempo de execução. Caso o intervalo informado não possua uma raíz, de acordo com teorema **??**, uma mensagem é apresentada ao usuário e a execução encerra. Caso o intervalo seja válido a raíz correspondente é apresentada e então o programa se encerra.

4 Código Fonte

```
#include<stdio.h>
#include <locale.h>

double M[20][20], X[10], multiplier;
int n;

void read_elements(){
  printf("Insira a quantidade de linhas da matriz aumentada: \n");
  scanf("%d",&n);
  printf("Insira os elementos da matriz aumentada:\n");
  for(int i = 1; i <= n; i++){
    for(int j = 1; j <= (n + 1); j++){
      printf("M[%d][%d]:", i,j);
      scanf("%lf",&M[i][j]);
    }
  }
}

void diagonalize_matrix(){
  for(int j = 1; j <= n; j++){
    for(int i = 1; i <= n; i++){
      if(i != j){
        multiplier = M[i][j] / M[j][j];
        for(int k = 1; k <= (n + 1); k++){
          M[i][k] = M[i][k] - multiplier * M[j][k];
        }
      }
    }
  }
}

void show_results(){
  printf("A solução para o sistemas de equações lineares é: \n");
  for(int i = 1; i <= n; i++){
    X[i] = M[i][n+1] / M[i][i];
    printf("I%d = %.10lf\n",i,X[i]);
  }
}

int main()
{

  setlocale(LC_ALL,"");
  read_elements();
  diagonalize_matrix();
  show_results();
```

```

   return(0);
}
```