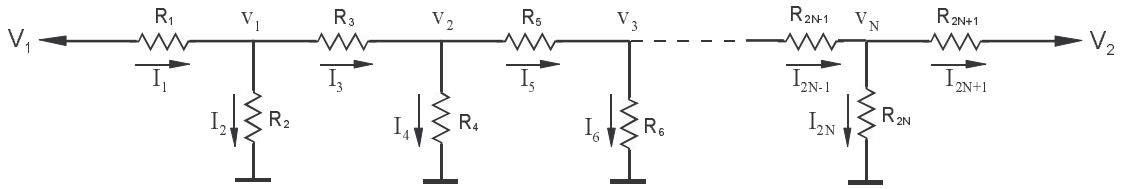
Problemas em Engenharia : Solução de equações algébricas

7ª série de exercícios

EXEMPLO: Array de resistores (ponte)



Problema: Conhecidos os valores dos resistores e as tensões V_1 e V_2, encontrar as voltagens nos nós v_1, v_2, ..., v_N.

SOLUÇÃO 1: Podemos facilmente montar um sistema de equações algébricas para v_1, .. v_N.

Basta escrever a conservação da corrente em cada nó e usar a lei de Ohm para expressar essa relação em termos das voltagens:

$$\begin{cases} I_1 = I_2 + I_3 \\ I_3 = I_4 + I_5 \\ \dots \\ I_{2N-1} = I_{2N} + I_{2N+1} \end{cases} \qquad \begin{cases} \dfrac{V_1 - v_1}{R_1} = \dfrac{v_1}{R_2} + \dfrac{v_1 - v_2}{R_3} \\ \dfrac{v_1 - v_2}{R_3} = \dfrac{v_2}{R_4} + \dfrac{v_2 - v_3}{R_5} \\ \dots \\ \dfrac{v_{N-1} - v_N}{R_{2N-1}} = \dfrac{v_N}{R_{2N}} + \dfrac{v_N - V_2}{R_{2N+1}} \end{cases}$$

$$\begin{cases} \left(\dfrac{1}{R_1} + \dfrac{1}{R_2} + \dfrac{1}{R_3}\right) v_1 - \dfrac{1}{R_3} v_2 = \dfrac{V_1}{R_1} \\ -\dfrac{1}{R_3} v_1 + \left(\dfrac{1}{R_3} + \dfrac{1}{R_4} + \dfrac{1}{R_5}\right) v_2 - \dfrac{1}{R_5} v_3 = 0 \\ \dots \\ -\dfrac{1}{R_{2N-1}} v_{N-1} + \left(\dfrac{1}{R_{2N-1}} + \dfrac{1}{R_{2N}} + \dfrac{1}{R_{2N+1}}\right) v_N = \dfrac{V_2}{R_{2N+1}} \end{cases}$$

Esse é um sistema tridiagonal simétrico para as incógnitas v_1, v_2, ... v_N, que tem a forma matricial

$$\begin{pmatrix} a_1 & -b_1 & & & & \\ -b_1 & a_2 & -b_2 & & 0 & \\ & -b_2 & a_3 & -b_3 & & \\ & & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot & \\ & 0 & & -b_3 & a_{N-1} & -b_{N-1} \\ & & & & -b_{N-1} & a_N \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \cdot \\ \cdot \\ v_{N-1} \\ v_N \end{pmatrix} = \begin{pmatrix} V_1/R_1 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \\ V_2/R_{2N+1} \end{pmatrix}$$

É fácil fazer um algoritmo para resolver esse sistema *(veja, por exemplo, o Numerical Recipes)*.

SOLUÇÃO 2: Vamos resolver o problema por um método iterativo. A idéia é "chutar" valores iniciais para as voltagens v_1, v_2, ... v_N, calcular as correntes e corrigir os valores. Essa iteração é aplicada até que as correções sejam muito pequenas.

Há várias maneiras de se fazer isso, e encontrar um esquema eficiente exige certa "arte" por parte do programador.

A importância dos métodos iterativos é que podemos aplicá-los mesmo quando as equações envolvidas não são lineares.
(isto iria acontecer se um dos elementos da ponte fosse não-linear, tal como um lead, um diodo ou um varistor)

Uma maneira é a seguinte:

passo 1: chute valores iniciais para as tensões $v_1^0, v_2^0, \dots, v_N^0$

passo 2 : encontre as correntes pelos resistores da ponte (com índice ímpar), I_1, I_3, I_5, ... I_{2N+1}

Se definirmos $v_0^0 = V_1$ e $v_{N+1}^0 = V_2$, basta fazer um loop para $I_i = \dfrac{v^0_{\frac{i-1}{2}} - v^0_{\frac{i+1}{2}}}{R_i}$, i = 1, (2N+1), 2

passo 3 : calcule as correntes pelos resistores de shunt (com índice par), I_2, I_4, I_6, ..., I_{2N}

Basta fazer $I_i = I_{i-1} - I_{i+1}$, i = 2, (2N), 2

passo 4: encontre os novos valores das voltagens v_1, v_2, ... v_N usando as correntes pelos resistores de shunt

$$v_i = R_{2i}.I_{2i} \text{ , } i = 1, N, 1$$

passo 5: atualize os valores das voltagens fazendo uma combinação entre os valores antigos e os valores corrigidos (isto é chamado de "relaxação")

$$v_i = \omega v_i + (1-\omega)v_i^0$$

O parâmetro ω deve ser escolhido, via de regra, entre 0 e 1. O melhor valor é encontrado por tentativa. Dependendo do problema, um valor de ω maior do que 1 ou mesmo negativo pode funcionar.

passo 6: repita os passos 2 a 5 até que as correções feitas em cada iteração sejam suficientemente pequenas, isto é, até que $\|v - v^0\| < \varepsilon$, onde ε é a máxima tolerância admitida. A norma $\| \ \|$ utilizada usualmente é a da máxima diferença absoluta *(se algum valor da solução for muito pequeno, é melhor usar a máxima diferença absoluta relativa)*.

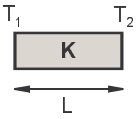
Um programa em C implementando esses passos é o seguinte:

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

/* flag para imprimir as iteracoes */
#define DEBUG 1
/* numero maximo de resistores - deve ser um numero impar */
#define NRMax 21
/* Tamanho do array dos resistores - deve ser (NRMax+1) */
#define DRMax 22
/* array dos resistores */
double R[DRMax];
/* array das correntes */
double I[DRMax];
/* Numero maximo de nos de voltagens - deve ser (NRMax-1)/2 */
#define VMax 10
/* Tamanho do array de voltagens - deve ser (VMax+1) + 2 */
#define DVMax 12
double V[DVMax], V0[DVMax];
double EPS = 0.001; // tolerancia minima desejada
double w = 0.1; // fator de relaxacao
int ITMAX = 2000; // numero maximo de iteracoes

int main(){
   int N, NR, ok, i, it, j, N2;
   double dv, Vesq, Vdir, dif, err_V;

   N = 0;
   while(N < 1 || N > VMax){
     printf("Entre com o numero de nos de voltagens (de 1 a %d):\n", VMax);
     scanf("%d", &N);
     if(N > VMax)printf("\n ---> No maximo %d !!!\n", VMax);
     if(N < 1)printf("\n ---> No minimo um !!!\n");
   }
   printf("\n N = %d nos\n", N);

   printf("\nEntre com os valores da fontes esquerda (Vesq) e direita (Vdir):\n");
   scanf("%lf %lf", &Vesq, &Vdir);
   printf("\n Vesq = %g  Vdir = %g\n", Vesq, Vdir);
   /* colocar esses valores no inicio e no final do array das voltagens */
   V[0] = Vesq;     V[N+1] = Vdir;

   NR = 2*N +1; // numero de resistores
```

```
 ok = 0;
   while(!ok){ ok = 1;
     printf("\nEntre com os valores dos resistores, na ordem da rede:\n");
     for(i=1; i<=NR; i++){ scanf("%lf", &R[i]);
                           if(R[i] <= 0){ printf("\n ---> Valores invalidos !!!\n");
                                          ok = 0; }
                         }
   }
   printf("\nValores dos resistores : \n   ");
   for(i=1; i<=NR; i++)printf("  %g", R[i]);
   printf("\n");

   /* O programa agora coloca valores iniciais, igualmente espacados, para as N voltagens */
   /* Os valores devem estar entre Vesq e Vdir */
   printf("\n Valores iniciais:\n");
   dv = (Vdir-Vesq)/(N+1);
   V[1] = Vesq + dv;
   printf("  %g", V[1]);
   for(i=2; i<=N; i++){ V[i] = V[i-1] + dv;
                        printf(" %g", V[i]); }
   printf("\n");
   system("pause");

   it = 0; // inicializacao do contador de iteracoes
   /* guardar o valor atual das voltagens */
   for(i=1; i<=N; i++)V0[i] = V[i];
   do {
      if(++it > ITMAX){
            printf("\n NAO CONVERGIU : w = %g  ITMAX = %d\n", w, it);
            system("pause"); return 1; }
      /* atualizar os valores das voltagens, relaxados */
      for(i=1;i<=N;i++)V[i] = w*V[i] + (1.0-w)*V0[i];
      /* guardar o valor atual das voltagens */
      for(i=1; i<=N; i++)V0[i] = V[i];
      /* calcular as correntes nos resistores impares */
      for(i=1; i<=NR; i=i+2){
            j = (i-1)/2;
            I[i] = (V[j] - V[j+1])/R[i]; }
      /* calcular as correntes nos resistores pares */
      N2 = 2*N;
      for(i=2; i<=N2; i=i+2)I[i] = I[i-1] - I[i+1];
      /* calcular os novos valores das voltagens, usando os resistores pares */
      for(i=1; i<=N; i++)V[i] = R[2*i]*I[2*i];
      /* obter a diferenca maxima entre os valores novos e velhos das voltagens */
      err_V = 0;
      for(i=1;i<=N;i++){dif = fabs(V[i]-V0[i]);
                        if(err_V < dif)err_V = dif; }
      if(DEBUG){
               printf("\n [%d : %g]", it, err_V);
               for(i=1;i<=N;i++)printf("  %g", V[i]);
               printf("\n");
               system("pause"); }

     } while (err_V > EPS);

   /* convergindo, imprimir o resultado */
   printf("\n SOLUCAO (apos %d iteracoes) : \n   ", it);
   printf("\nVoltagens : "); for(i=1;i<=N;i++){ printf("  %-+9.5lf", V[i]);
                                                 if(i%5 == 0)printf("\n           "); }
   printf("\n");
   printf("\nCorrentes : "); for(i=1;i<=NR;i++){ printf("  %-+9.5lf", I[i]);
                                                  if(i%5 == 0)printf("\n           "); }
   printf("\n\n");

   system("pause");
   return 0;
}
```

EXERCÍCIO : O fluxo de calor por uma barra homogênea de comprimento L e condutividade térmica K é dado por

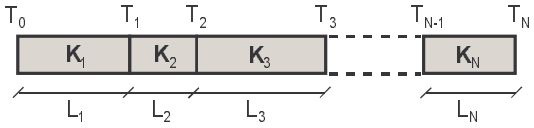
$$\phi = K\frac{T_2 - T_1}{L}$$



T_1 e T_2 são as temperaturas nos extremos da barra.

(estamos supondo que não há perdas durante a condução, e que o fluxo é unidimensional)

Considere uma montagem de N barras em contato, uma após a outra, e que as temperaturas nas extremidades esquerda e direita do conjunto sejam conhecidas (T_0 e T_N).



O problema consiste em encontrar as temperaturas em cada uma das junções, T_1, T_2, ... T_N.

Ora, se a temperatura em cada ponto não variar com o tempo, então o fluxo deve ser o mesmo através de todas as barras:

$$\phi = K_1\frac{T_1 - T_0}{L_1} = K_2\frac{T_2 - T_1}{L_2} = \ldots = K_N\frac{T_N - T_{N-1}}{L_N}$$

Essas relações equivalem a um sistema linear nas N incógnitas T_1, T_2, ..., T_N, que pode ser escrito como:

$$\begin{cases} T_1 - T_0 = \dfrac{L_1}{K_1}\phi \\ T_2 - T_1 = \dfrac{L_2}{K_2}\phi \\ \ldots\ldots \\ T_N - T_{N-1} = \dfrac{L_N}{K_N}\phi \end{cases}$$

(a) Some as equações acima e mostre que $\phi = \dfrac{T_N - T_0}{\left(\dfrac{L_1}{K_1} + \dfrac{L_2}{K_2} + \ldots\dfrac{L_N}{K_N}\right)}$. Portanto, as temperaturas podem ser encontradas facilmente.

(b) Escreva um programa em C para encontrar as N temperaturas. O programa deve ler o número de barras, o comprimento e a condutividade de cada uma, e as temperaturas nas extremidades esquerda e direita do conjunto. O programa deve também calcular a condutividade térmica equivalente $\overline{K}$ do conjunto, que é definida como $\phi = \overline{K}\dfrac{T_N - T_0}{\sum L_i}$.