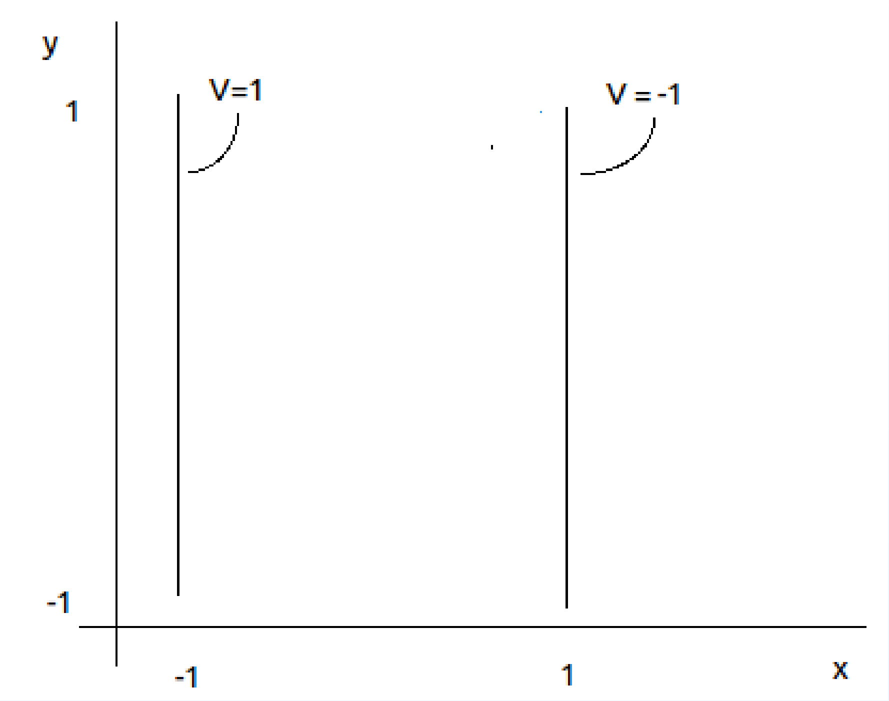
Potencial em uma caixa

João Medeiros (joaomedeiros@dfte.ufrn.br)

Descrição do problema

Potencial em uma caixa



As condições de contorno são dadas pelo valor do potencial nas paredes esquerda e direita.

Objetivo

- Determinar o valor do potencial $V(x, y)$

Objetivo

- Determinar o valor do potencial $V(x, y)$
- Determinar o campo elétrico $E = -\nabla V$

Solução do problema

Solução

Solução

- Iremos utilizar o método da relaxação

Solução

- Iremos utilizar o método da relaxação

  - Inicie com $u_0=u(x,y,z,0)$
  - Evolua $u_0 \longrightarrow u_1 \longrightarrow u_2$ ... no tempo...
  - Até atingir uma solução estacionária (independente do tempo) $u_n(x,y,z,t\to\infty)$.
    - ♣ Esta solução deve, obviamente, ter $\frac{\partial u_n}{\partial t}=0$
  - Então, $u_n(x,y,z,t\to\infty)$ também satisfaz a equação de Laplace!

Visão geral do programa

Visão geral do programa

1. Declaração das variáveis necessárias

Visão geral do programa

1. Declaração das variáveis necessárias

2. Inicialização das variáveis - Condições de contorno

Visão geral do programa

1. Declaração das variáveis necessárias
2. Inicialização das variáveis - Condições de contorno
3. Cálculo da solução do problema - usando o método da relaxação
    - calcula a matriz em cada passo
    - verifica se houve convergência
    - a partir da solução obtida, calcula o campo elétrico

Visão geral do programa

1. Declaração das variáveis necessárias
2. Inicialização das variáveis - Condições de contorno
3. Cálculo da solução do problema - usando o método da relaxação
   - calcula a matriz em cada passo
   - verifica se houve convergência
   - a partir da solução obtida, calcula o campo elétrico
4. Análise dos resultados
   - Gravação dos dados em formato adequado para software de visualização (gnuplot)
   - visualização da solução através de gráficos

Análise dos trechos do programa

Declaração das variáveis necessárias

Trecho inicial da função principal

```
main() {

double deltaV;

// matrizes utilizadas no cálculo da solução
double V[N][N];
double Vn1[N][N];

// ponteiro para o arquivo que será utilizado na gravação
// da solução
FILE *fp;

// indica se houve ou não convergência
int convergiu;
```

Correspondência entre as matrizes utilizadas no programa e a matriz do sistema de coordenadas

Matriz utilizada no programa

$$\begin{vmatrix} (0,0) & (0,1) & \dots & (0,N-1) \\ (1,0) & (1,1) & \dots & (1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ (N-1,0) & (N-1,1) & \dots & (N-1,N-1) \end{vmatrix}$$

Sistema de coordenadas, definida pelo problema

$$\begin{vmatrix} (-1,1) & (-1+dx,1) & \dots & (1,1) \\ (-1,1-dy) & (-1+dx,1-dy) & \dots & (1,1-dy) \\ \vdots & \vdots & \ddots & \vdots \\ (-1,-1) & (-1+dx,-1) & \dots & (1,-1) \end{vmatrix}$$

Correspondência entre as matrizes utilizadas no programa e a matriz do sistema de coordenadas

Matriz utilizada no programa

$$\begin{vmatrix} (0,0) & (0,1) & \dots & (0,N-1) \\ (1,0) & (1,1) & \dots & (1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ (N-1,0) & (N-1,1) & \dots & (N-1,N-1) \end{vmatrix}$$

$(0,0) \to (-1,1)$

Sistema de coordenadas, definida pelo problema

$$\begin{vmatrix} (-1,1) & (-1+dx,1) & \dots & (1,1) \\ (-1,1-dy) & (-1+dx,1-dy) & \dots & (1,1-dy) \\ \vdots & \vdots & \ddots & \vdots \\ (-1,-1) & (-1+dx,-1) & \dots & (1,-1) \end{vmatrix}$$

Correspondência entre as matrizes utilizadas no programa e a matriz do sistema de coordenadas

Matriz utilizada no programa

$$\begin{vmatrix} (0,0) & (0,1) & \dots & (0,N-1) \\ (1,0) & (1,1) & \dots & (1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ (N-1,0) & (N-1,1) & \dots & (N-1,N-1) \end{vmatrix}$$

$(0,0) \rightarrow (-1,1)$
$(1,1) \rightarrow (1+dx, 1-dy)$

Sistema de coordenadas, definida pelo problema

$$\begin{vmatrix} (-1,1) & (-1+dx,1) & \dots & (1,1) \\ (-1,1-dy) & (-1+dx,1-dy) & \dots & (1,1-dy) \\ \vdots & \vdots & \ddots & \vdots \\ (-1,-1) & (-1+dx,-1) & \dots & (1,-1) \end{vmatrix}$$

Correspondência entre as matrizes utilizadas no programa e a matriz do sistema de coordenadas

Matriz utilizada no programa

$$\begin{vmatrix} (0,0) & (0,1) & \dots & (0,N-1) \\ (1,0) & (1,1) & \dots & (1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ (N-1,0) & (N-1,1) & \dots & (N-1,N-1) \end{vmatrix}$$

Sistema de coordenadas, definida pelo problema

$$\begin{vmatrix} (-1,1) & (-1+dx,1) & \dots & (1,1) \\ (-1,1-dy) & (-1+dx,1-dy) & \dots & (1,1-dy) \\ \vdots & \vdots & \ddots & \vdots \\ (-1,-1) & (-1+dx,-1) & \dots & (1,-1) \end{vmatrix}$$

$(0,0) \rightarrow (-1,1)$

$(1,1) \rightarrow (1+dx, 1-dy)$

dx = (Xmax - Xmin) / (N-1)

Correspondência entre as matrizes utilizadas no programa e a matriz do sistema de coordenadas

Matriz utilizada no programa

$$\begin{vmatrix} (0,0) & (0,1) & \dots & (0,N-1) \\ (1,0) & (1,1) & \dots & (1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ (N-1,0) & (N-1,1) & \dots & (N-1,N-1) \end{vmatrix}$$

$(0,0) \rightarrow (-1,1)$

$(1,1) \rightarrow (1+dx, 1-dy)$

dx = (Xmax - Xmin) / (N-1)

Sistema de coordenadas, definida pelo problema

$$\begin{vmatrix} (-1,1) & (-1+dx,1) & \dots & (1,1) \\ (-1,1-dy) & (-1+dx,1-dy) & \dots & (1,1-dy) \\ \vdots & \vdots & \ddots & \vdots \\ (-1,-1) & (-1+dx,-1) & \dots & (1,-1) \end{vmatrix}$$

Trecho de programa para realizar a correspondência entre as matrizes

```
dx = (Xmax - Xmin) / (double) (N-1);
dy = dx;
y = Xmax;
for(i=0; i<N; i++) {
  x = -Xmin;
  for(j=0; j<N; j++) {
        ...
        x += dx;
  }
  ...
  y -= dy;
}
```

Inicialização das variáveis - Condições de contorno

```
// inicializa as matrizes V e Vn1 com as
// condicoes de contorno
void inicia(double V[N][N], double Vn1[N][N]) {
        int i,j;
        for(i=0; i<N; i++) {
                for(j=0; j<N; j++) {
                        V[i][j]=Vn1[i][j]=0.0;
                }
        }
        // atribuir o valor 1 na parede esquerda
        for(i=0; i<N; i++) {
                V[i][0]=Vn1[i][0]=1.0;
        }
        // atribuir o valor -1 na parede direita
        for(i=0; i<N; i++) {
                V[i][N-1]=Vn1[i][N-1]=-1.0;
        }
}
```

Cálculo da solução do problema - usando o método da relaxação

Atualiza a matriz em cada passo, observe que além de atualizar a matriz, essa função também calcula a diferença entre a nova e a antiga matriz.

```
double atualiza(double V[N][N], double Vn1[N][N]) {
        double deltaV = 0;
        int i,j;
        // calcula inicialmente a parte interior da matriz
        for(i=1; i<N-1; i++) {
                for(j=1; j<N-1; j++) {
                        Vn1[i][j] = 0.25*( V[i+1][j] + V[i-1][j] +
                                        V[i][j+1] + V[i][j-1] );
                        deltaV += fabs(V[i][j] - Vn1[i][j]);
                }
        }
        // calcula os elementos da borda superior
        for(j=1; j<N-1; j++) {
                Vn1[0][j] = 1.0/3.0*( V[1][j] +
                                V[0][j+1] + V[0][j-1] );
                deltaV += fabs(V[0][j] - Vn1[0][j]);
        }
        // calcula os elementos da borda inferior
        for(j=1; j<N-1; j++) {
                Vn1[N-1][j] = 1.0/3.0*( V[N-2][j] +
                                V[N-1][j+1] + V[N-1][j-1] );
                deltaV += fabs(V[N-1][j] - Vn1[N-1][j]);
        }
        return deltaV;
}
```

Cálculo da solução do problema - usando o método da relaxação

```
/*
 Calcula o valor de V
 Parametros de entrada:
  V - matriz com a solucao inicial
  Vn1 - matriz calculada
  eps - criterio de convergencia
  itMax - numero maximo de iteracoes
 Parametros de retorno
  A funcao retornara 1 se houver convergencia e zero se nao houver
  convergencia. Se houve convergencia as matrizes V e Vn1 serao a solucao
  do problema.
*/

int calcula(double V[N][N], double Vn1[N][N], double eps, int itMax) {
     int i;
     double deltaV;
     int convergiu = 0;
     for(i=0; i< itMax; i++) {
        deltaV = atualiza(V, Vn1);
        deltaV = atualiza(Vn1, V);
        if(deltaV < eps) {
           convergiu=1;
           break;
        }
        printf("i=%d deltaV = %lf\n",i, deltaV);
    }
    printf("i=%d deltaV = %lf\n",i, deltaV);
    return convergiu;
}
```

Cálculo da solução do problema - usando o método da relaxação

Utilizando as funções criadas até agora

```
// inicializa as matrizes com as condicoes iniciais
inicia(V, Vn1);

convergiu = calcula(V, Vn1, 1.0e-5, 100);
if(convergiu ==1 ) {
  // se houve convergengia, imprime os resultados

}
```

Cálculo do campo elétrico

- Determinar o campo elétrico $E = -\nabla V$

$$\vec{E} = -\vec{\nabla} V \rightarrow \left.\begin{aligned} E_x &= -\frac{\partial V}{\partial x} \\ E_y &= -\frac{\partial V}{\partial y} \end{aligned}\right\} \quad E_x = \frac{V(i+1,j) - V(i-1,j)}{2\Delta x}$$

Cálculo do campo elétrico

```
void calculaE(double V[N][N], double Ex[N][N], double Ey[N][N]) {
    int i,j;
    double V1, V2;
    for(i=0;i<N;i++) {
        for(j=0;j<N;j++) {
            Ex[i][j]=Ey[i][j]=0.0;
        }
    }
    // nao vamos nos preocupar com as bordas por enquanto
    for(i=1;i<N-1;i++) {
        for(j=1;j<N-1;j++) {
                V1 = V[i+1][j];
                V2=V[i-1][j];
            Ex[i][j] = - 0.5*(V1 - V2)/dx;
            // calcula Ey
                V1 = V[i][j+1];
                V2=V[i][j-1];
            Ey[i][j] = - 0.5*(V1 - V2)/dx;
        }
    }
}
```

Exibição dos resultados

Gravação dos dados em formato adequado para software de visualização (gnuplot)

Potencial elétrico

- Utilizaremos o comando *splot* do gnuplot
- O gnuplot lê os dados a partir de um arquivo
- Formato do arquivo

```
x1 y1 z1
x1 y2 z2
\\linha em branco
x2 y1 z3
x2 y2 z4
...
```

- Deveremos escrever a nossa matriz solução no formato acima

Gravação dos dados em formato adequado para software de visualização (gnuplot)

Potencial elétrico

```
// recebe a matriz V e um ponteiro para um arquivo
// e imprime a matriz nesse arquivo usando
// o formato que podera ser utilizado pelo
// gnuplot

void imprimeGrafico(double V[N][N], FILE *fp) {
 int i,j;
 double x, y;
 double dx,dy;
  Xmin = -1;
  Xmax = 1;
  x = Xmin;
  y = Xmax;
  dx = (Xmax - Xmin) / (double) (N-1);
  dy = dx;
  for(i=0; i<N; i++) {
      x = Xmin;
      for(j=0; j<N; j++) {
         fprintf(fp, "%5.2lf %5.2lf %5.2lf\n",x, y,V[i][j]);
         x += dx;
       }
       // pula uma linha para separar as linhas da matriz
       fprintf(fp,"\n");
       y -= dy;
   }
}
```

Gravação dos dados em formato adequado para software de visualização (gnuplot)

Campo elétrico

- Utilizaremos o comando *plot* do gnuplot com o estilo $vector$
- Formato do arquivo, para o caso de um campo vetorial, caso 2D

```
x y dx dy
...
```

- O estilo $vector$ irá desenhar vetores de (x, y) para $(x + dx, y + dy)$
- Deveremos escrever a nossa matriz do campo elétrico no formato acima

Gravação dos dados em formato adequado para software de visualização (gnuplot)

Campo elétrico

```
void imprimeCampo (double Ex[N][N], double Ey[N][N], FILE *fp) {
    int i,j;
    double x,y;
    y=Xmax;
    for(i=0;i<N;i++) {
        x=Xmin;
        for(j=0;j<N;j++) {
            fprintf(fp,"%6.2lf %6.2lf %6.2lf %6.2lf\n",x, y,
                    Ey[i][j], Ex[i][j]);
            x+=dx;
        }
        y -=dx;
    }
}
```

Gravação dos dados em formato adequado para software de visualização (gnuplot)

Trecho do programa principal, utilizando as funções criadas

```
main() {
double deltaV;
double V[N][N];
double Vn1[N][N];
double Ex[N][N];
double Ey[N][N];
FILE *vfp, *efp;
int convergiu;
dx = (Xmax - Xmin) / (double) (N-1);
dy = dx;
// inicializa as matrizes com as condicoes iniciais
inicia(V, Vn1);
convergiu = calcula(V, Vn1, 1.0e-5, 100);
if(convergiu ==1 ) {
  calculaE(V, Ex, Ey);
  vfp = fopen("Vxy.dat","w");
  imprimeGrafico(V, vfp);
  fclose(vfp);
  efp = fopen("Exy.dat","w");
  imprimeCampo(Ex, Ey, efp);
  fclose(efp);
} else {
  printf("Nao houve convergencia\n");
}
}
```

Gravação dos dados em formato adequado para software de visualização (gnuplot)

Exemplo do arquivo Vxy.dat gerado pelo programa anterior

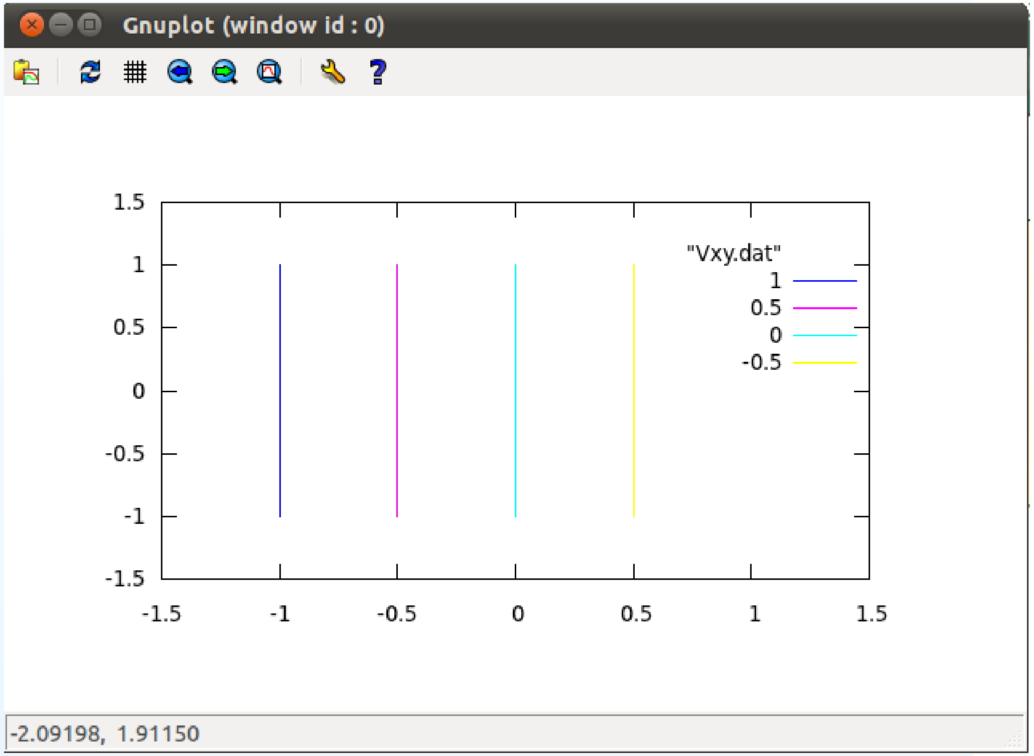
```
$ more Vxy.dat
-1.00  1.00  1.00
-0.75  1.00  0.75
-0.50  1.00  0.50
-0.25  1.00  0.25
 0.00  1.00  0.00
 0.25  1.00 -0.25
 0.50  1.00 -0.50
 0.75  1.00 -0.75
 1.00  1.00 -1.00

-1.00  0.75  1.00
-0.75  0.75  0.75
-0.50  0.75  0.50
-0.25  0.75  0.25
 0.00  0.75 -0.00
 0.25  0.75 -0.25
 0.50  0.75 -0.50
 0.75  0.75 -0.75
 1.00  0.75 -1.00
```

Visualização da solução através de gráficos

Visualizando resultado com o gnuplot

```
$ gnuplot
Terminal type set to 'wxt'
gnuplot> set xrange [-1.5:1.5]
gnuplot> set yrange [-1.5:1.5]
gnuplot> set view map
gnuplot> unset surface
gnuplot> set hidden3d
gnuplot> set contour base
gnuplot> splot "Vxy.dat" with lines
```

Gráfico das equipotenciais



Uma maneira de lembrar como gerar o gráfico, é criar um arquivo com os comando utilizados, por exemplo "Vxy.gnu"

Visualização da solução através de gráficos

Conteúdo do arquivo “Vxy.gnu”

```
set xrange [-1.5:1.5]
set yrange [-1.5:1.5]
set view map
unset surface
set hidden3d
set contour base
splot "Vxy.dat" with lines
```

Executando o gnuplot, passando como argumento o nome do arquivo a ser utilizado

```
$ gnuplot --persist Vxy.gnu
```

A opção –persist informa ao gnuplot para deixar a janela ativa após a execução.

Visualização da solução através de gráficos

Gerando uma figura, para ser utilizada em um artigo, trabalho, etc, com o resultado.

Conteúdo do arquivo “Vxy2.gnu”

```
set xrange [-1.5:1.5]
set yrange [-1.5:1.5]
set view map
unset surface
set hidden3d
set contour base
set term png
set output "Vxy.png"
splot "Vxy.dat" with lines
```

Execute o gnuplot, passando como argumento o nome do arquivo a ser utilizado

```
$ gnuplot Vxy.gnu
```

O comando acima deve gerar o arquivo ”Vxy.png”com as equipotenciais. O comando set term pode ser utilizado para gerar figuras em uma quantidade enorme de formatos.

Gravação dos dados em formato adequado para software de visualização (gnuplot)

Exemplo do arquivo Exy.dat gerado

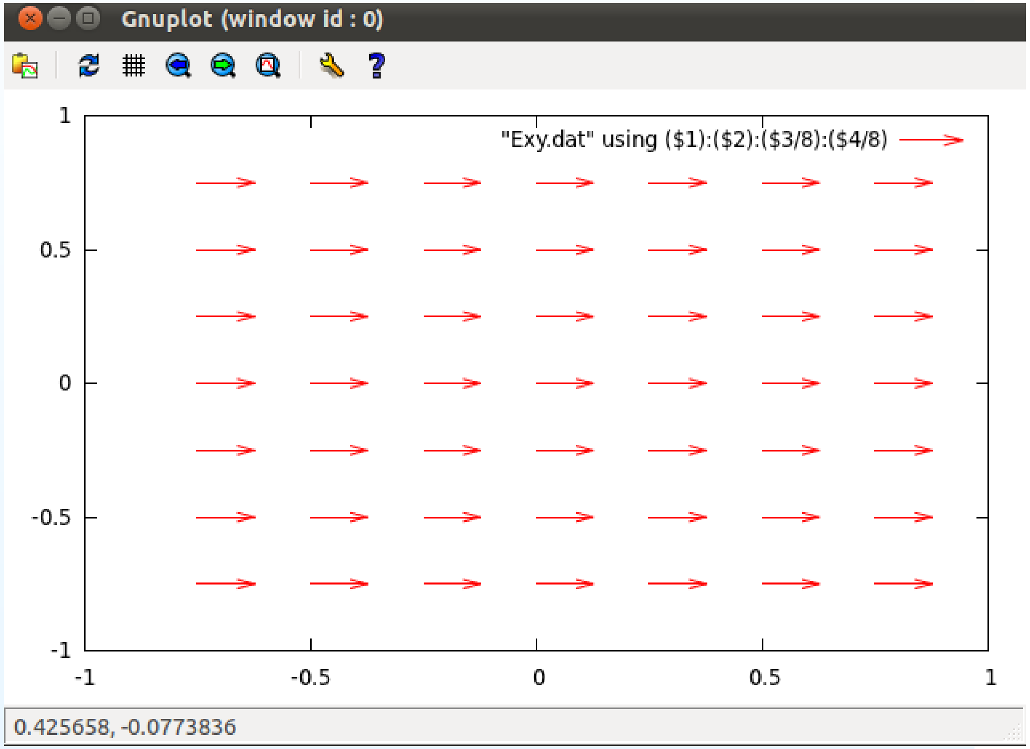
```
$ more Exy.dat
 -1.00   1.00   0.00   0.00
 -0.75   1.00   0.00   0.00
 -0.50   1.00   0.00   0.00
 -0.25   1.00   0.00   0.00
  0.00   1.00   0.00   0.00
  0.25   1.00   0.00   0.00
  0.50   1.00   0.00   0.00
  0.75   1.00   0.00   0.00
  1.00   1.00   0.00   0.00
 -1.00   0.75   0.00   0.00
 -0.75   0.75   1.00   0.00
 -0.50   0.75   1.00   0.00
 -0.25   0.75   1.00   0.00
  0.00   0.75   1.00  -0.00
```

Visualização da solução através de gráficos

Visualizando o campo elétrico com o gnuplot

plot "Exy.dat"using ($1):($2):($3/8):($4/8) with vec

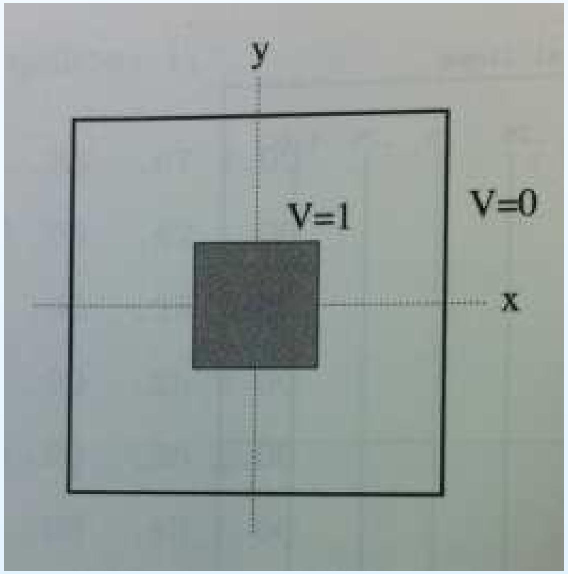
Gráfico do vetor campo elétrico



- Observe que dividimos o valor do campo por 8 de maneira a que os vetores "caibam" dentro da nossa matriz. Isso foi feito através das expressões ($3/8) e ($4/8) que significam: divida o valor da coluna 3 por 8 e divida o valor da coluna 4 por 8.
- Execute mesmo comando, sem a divisão vara visualizar como ficaria a representação do campo vetorial.
- Podemos usar o mesmo procedimento utilizado para o potencial elétrico e criar um arquivo Exy.gnu com os comando necessários para criar o gráfico.

Exercícios

1. Modifique o programa desenvolvido nessa aula para resolver a equação de laplace para a situação mostrada na figura abaixo. Que esquematiza um prisma infinito, na direção z, com uma parte central condutora. O potencial nas paredes externas do prisma é nulo e na parte interna é mantida em $V = 1$. Considere que os lados do prisma tem dimensão de 2 unidades e a parte interna dimensão de 0.6 unidades. Utilize $dx = 0.1$.



Exercícios

2. Modifique o programa desenvolvido nessa aula para resolver a equação de laplace para a situação mostrada na figura abaixo. Considere que a caixa tem largura de 2 unidades e as placas estão colocadas nas posições $x = \pm 0.3$ e têm comprimento de 0.6 unidades cada uma, posicionadas de tal maneira que suas extremidades estão em $y = \pm 1$. Utilize $dx = 0.1$.