Métodos Computacionais – Eletromagnetismo

Aula: Solução numérica da Equação de Laplace.

Eletromagnetismo: potenciais e campos

- Resolução numérica da equação de Laplace/Poisson.

$$\boxed{\nabla^2 V(\mathbf{r}) = 0}$$

$$\boxed{\nabla^2 V(\mathbf{r}) = -\frac{\rho(\mathbf{r})}{\epsilon_0}}$$

(**+ condições de contorno**)

- Campos magnéticos gerados por uma corrente

(Lei de Biot-Savard)

Aula: Eq. de Laplace - Objetivos

Ao final desta aula, você deverá ser capaz de:

1) Discretizar a Equação de Laplace homogênea (sem distribuição de cargas) para um potencial elétrico V(x,y) em 2D.
2) Determinar o potencial elétrico V(x,y) em todos os pontos de um "grid" (dadas as condições de contorno) através do método de relaxação de Jacobi.

Tarefa: Obter numericamente o potencial elétrico V(x,y) de duas placas paralelas mantidas a determinados valores de potencial.

Tempo aproximado: 20 a 30 min (lembrando que o *debug* é a maior parte disso!).

Solução numérica da Eq. de Laplace

Eq. de Laplace em 2D: $$\boxed{\frac{\partial^2 V(x,y)}{\partial x^2} + \frac{\partial^2 V(x,y)}{\partial y^2} = 0}$$

Condições de contorno: $V(x_p, y_p) = V_p$

Aproximação da 2a derivada.

$$\frac{\partial^2 V(x,y)}{\partial x^2} \approx \frac{V(x+\Delta x, y) - 2V(x,y) + V(x-\Delta x, y)}{(\Delta x)^2}$$

$$\frac{\partial^2 V(x,y)}{\partial y^2} \approx \frac{V(x, y+\Delta y) - 2V(x,y) + V(x, y-\Delta y)}{(\Delta y)^2}$$

A soma destas duas expressões deve ser igual a zero.

Solução numérica da Eq. de Laplace

Aproximação da 2a derivada.

$$(\Delta x)^2 \frac{\partial^2 V(x,y)}{\partial x^2} \approx V(x+\Delta x, y) - 2V(x,y) + V(x-\Delta x, y)$$

$$(\Delta y)^2 \frac{\partial^2 V(x,y)}{\partial y^2} \approx V(x, y+\Delta y) - 2V(x,y) + V(x, y-\Delta y)$$

Assumindo $\Delta x = \Delta y$ temos: a seguinte aproximação para V(x,y):

$$(\Delta x)^2 \left[\frac{\partial^2 V(x,y)}{\partial x^2} + \frac{\partial^2 V(x,y)}{\partial y^2}\right] = 0 \Rightarrow$$

$$\boxed{V(x,y) = \frac{1}{4}[V(x+\Delta x, y) + V(x-\Delta x, y) + V(x, y+\Delta y) + V(x, y-\Delta y)]}$$

Solução numérica da Eq. de Laplace

Assumindo $\Delta x = \Delta y$ temos a seguinte aproximação para V(x,y):

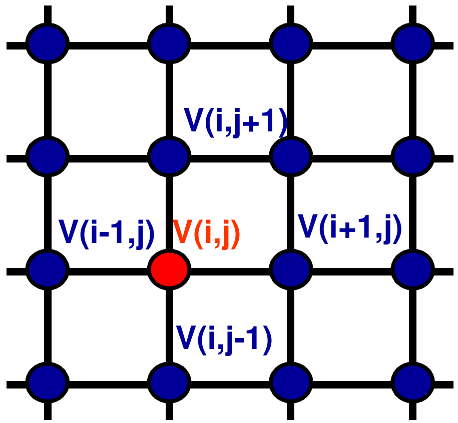
$$V(x,y) = \frac{1}{4}[V(x+\Delta x, y) + V(x-\Delta x, y) + V(x, y+\Delta y) + V(x, y-\Delta y)]$$

Forma discretizada:

$$x_i = (i-1)\,\Delta x \qquad y_j = (j-1)\,\Delta y \qquad V(x_i, y_j) \rightarrow V(i,j)$$

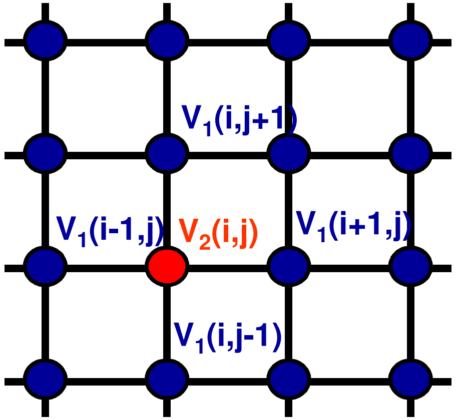
$$\boxed{V(i,j) = \frac{1}{4}[V(i+1,j) + V(i-1,j) + V(i,j+1) + V(i,j-1)]}$$

Mas todos os pontos dependem dos vizinhos!
("iterando" a partir da borda não é possível)



Como resolver?

Método de relaxação de Jacobi



"Chute" inicial para V(x,y): $V_1(i,j)$

Calculamos uma nova aproximação $V_2(x,y)$

$$V_2(i,j) = \frac{1}{4}\left[V_1(i+1,j) + V_1(i-1,j) + V_1(i,j+1) + V_1(i,j-1)\right]$$

Usamos $V_2(x,y)$ para calcular uma nova aproximação $V_3(x,y)$...

... e assim por diante!

$$\boxed{V_{n+1}(i,j) = \frac{1}{4}\left[V_n(i+1,j) + V_n(i-1,j) + V_n(i,j+1) + V_n(i,j-1)\right]}$$

Critério de convergência: $$\sum_{i,j} |V_{n+1}(i,j) - V_n(i,j)| < \epsilon$$

Importante: todos os $V_n(x,y)$ devem satisfazer as **condições de contorno.**

Eq. de Laplace – Tarefa (Fazer upload!)

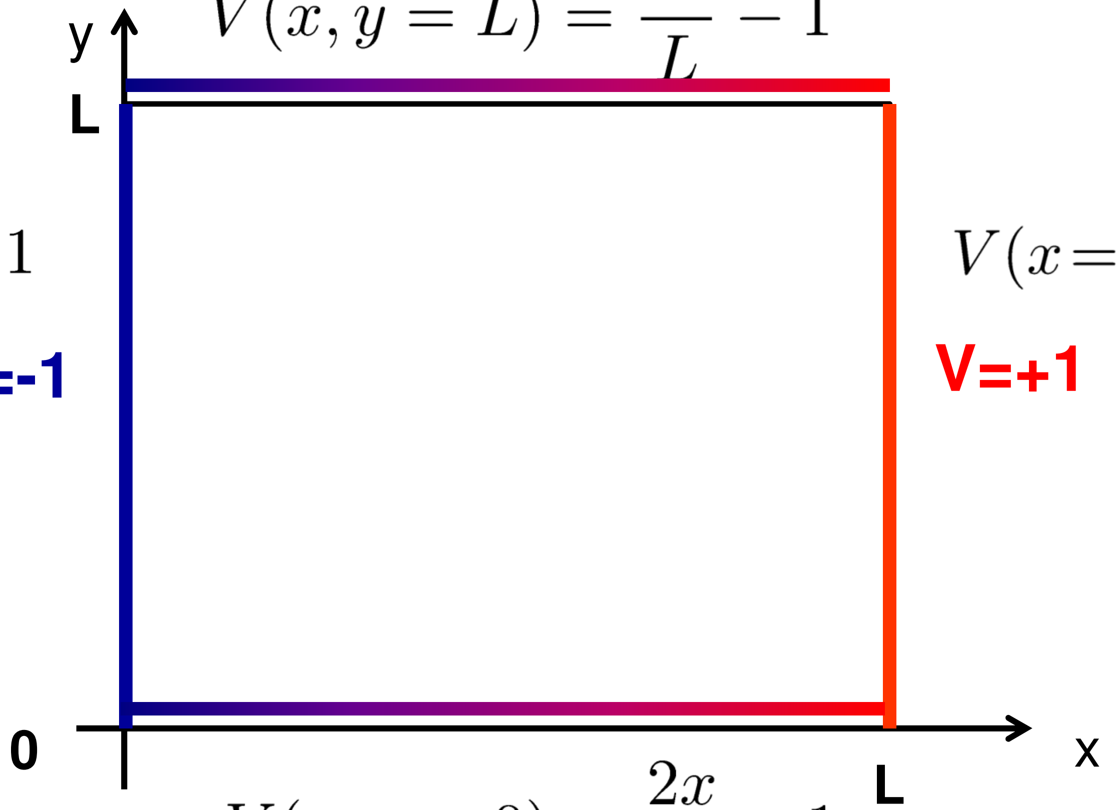
Resolva a equação de Laplace 2D para um sistema de duas placas paralelas mantidas a -1 e +1 V. Use as condições de contorno:

$$V(x, y = L) = \frac{2x}{L} - 1$$

$$V(x=0, y) = -1$$

$$V(x=L, y) = +1$$

$$V(x, y = 0) = \frac{2x}{L} - 1$$



Eq. de Laplace – Tarefa

- *Utilize como condição inicial* $\boxed{V_1(i,j)=0}$ *.*

(exceto nas bordas onde valem as condições de contorno)

- *Use* $\Delta x = \Delta y = 0.05L$ *ou menor.*
- *Obtenha a convergência* $\sum_{i,j} |V_{n+1}(i,j) - V_n(i,j)| < 10^{-3}$ *e* $n_{max}=500$*.*
- *Faça um gráfico de V(x,y) utilizando o comando* **contour** ou **plot3**

(Vide "Dicas" à frente)

Responda (comentário no script):

1) O potencial final tem alguma simetria espacial? Qual?

2) Você consegue pensar em algum "chute" inicial compatível com as condições de contorno? Seria mais rápida a convergência?

Eq. de Laplace - Tarefa – Dicas

- *Escolha L=1 e inicie no ponto (i,j)=(2,2).*
- *Para representar $V_{n+1}(x,y)$ e $V_n(x,y)$, basta utilizar* V(i,j) e V_{old}(i,j)*.*
- *A cada passo, atualize* V_{old}(i,j)=V(i,j) e calcule V(i,j).
- *Definindo os grids em x e y (xarray, yarray) utilize a função:*

contour(xarray,yarray,V',50)

para plotar o perfil do potencial em um plot com 50 cores.

- *Debug:* V_n(i, j) para i,j <=4

```
 n=2 DeltaV=14.00000
-1.0000 -1.0000 -1.0000 -1.0000
-0.9000 -0.4750 -0.2500 -0.2500
-0.8000 -0.2000 0.0000 0.0000
-0.7000 -0.1750 0.0000 0.0000
 n=3 DeltaV=10.00000
-1.0000 -1.0000 -1.0000 -1.0000
-0.9000 -0.5875 -0.4313 -0.3750
-0.8000 -0.3625 -0.1125 -0.0625
-0.7000 -0.2625 -0.0437 0.0000
```

```
 n=4 DeltaV=8.12500
-1.0000 -1.0000 -1.0000 -1.0000
-0.9000 -0.6734 -0.5188 -0.4672
-0.8000 -0.4406 -0.2250 -0.1375
-0.7000 -0.3328 -0.1031 -0.0266
 n=5 DeltaV=6.93750
-1.0000 -1.0000 -1.0000 -1.0000
-0.9000 -0.7148 -0.5914 -0.5273
-0.8000 -0.5078 -0.3000 -0.2109
-0.7000 -0.3789 -0.1648 -0.0664
```

Dica – Exemplo usando **contour/plot3**

```
yarray=-1:0.05:1;
xarray=-1:0.05:1;
Nx=length(xarray);
Ny=length(yarray);
Rsq=zeros(Nx,Ny);
for ii=1:Nx
  for jj=1:Ny
    Rsq(ii,jj)=xarray(ii)^2 + yarray(jj)^2;
  end
end
%usando contour
contour(xarray,yarray,Rsq',50);
colorbar;
%usando Plot3D
figure;
plot3(xarray, yarray, Rsq');
```

Note o operador de transposição!