

%%
%%

%

% PTC3213 - EC1 – 2019 – Método das Diferenças Finitas

% Solução da Equação de Laplace

% Turma 1 - Professora Viviane

%

% Guilherme Akira Alves dos Santos nUSP 11027484

% Jéssica Gonsalves Santos nUSP 10773864

% Letícia Mendonça Carraro nUSP 10374480

%

%%
%%

clear;

clf;

% define regioao:

% este dx e' a discretizacao utilizada (somente os valores abaixo são

% possíveis!)

%dx=0.05; % Tempo de execução longo!!

%dx=0.1;

%dx=0.25;

dx=0.5;

eps0= 8.854187817*(10^-12) ;

epsr= 2.5;

sigma= 3.5;

sigma_dual= 3.5;

dy=dx;

tol=1e-4; maxit=1e4; iter=0;

Vmin= 0 ;

Vmax= 100 ;

```

erro=0.0;

start= -1;

start_Dual=-1;


a= 11 ;

b= 5 ;

c= 4 ;

d= 3 ;

g= 3 ;

h= 1 ;

lx=a;

ly=b;

Nx=round(lx/dx)+1;

Ny=round(ly/dx)+1;


%

% figura 1 – Geometria do condutor

%

figure(1);

xv = [0 a a 0 0 NaN g g g+c g+c g];

yv = [0 0 b b 0 NaN h h+d h+d h h];

%

% Traçado do problema

%

plot(xv,yv,'LineWidth',2)

text(a/4,b+1,'EC1 - Condutor retangular vazado - Geometria','Color','r')

grid on

axis ([-1 a+2 -1 b+2])

%

% Discretização (geração da grade)

%
```

```

%
xgv=((1:Nx)-1)*dx;
ygv=((1:Ny)-1)*dx;
[x,y]=meshgrid(xgv,ygv);
[in,on] = inpolygon(x,y,xv,yv);
%
% figura 2 - Grade
%
figure(2)
plot(xv,yv)
text(a/3,b+1,'Grade Regular','Color','r')
axis ([-1 a+2 -1 b+2])
hold on
plot(x(in&~on),y(in&~on),'r+')

plot(x(on),y(on),'k*')

axis ([-1 a+2 -1 b+2])
grid on
hold off
%
% figura 3 – Pontos do Contorno
%

figure(3)
spy(on)
text((g+c/6)/dx,(h+d/3)/dx,'Nós - Contorno','Color','r')
%
% figura 4 - Nós internos
%
figure(4)

```

```

spy(in&~on)

text((g+c/8)/dx,(h+d/3)/dx,'Nós - Grade interna','Color','r')

%

% Atribui Condições de contorno

%

r=find(in); % tudo
p=find(in-on); %so' nós internos
q=find(on); %so' fronteira

iVmax=find(((x(q)>0.99*g) & (x(q)<1.01*(g+c)) & (y(q)>0.99*h) & (y(q) < 1.01*(h+d))));
iFuro=find(((x(:,.)>g) & (x(:,.)<(g+c)) & (y(:,.)>h) & (y(:,.) < (h+d))) );

Phi_prev=zeros(size(x));
Phi_new=zeros(size(x));
Phi_new(q(iVmax))= Vmax;
Phi_new(iFuro)= NaN;
Phi_new(p)= start;

%

%Contador de iterações

%

iter=0;

% Erro máximo entre duas iterações
erro=max(max(abs(Phi_new-Phi_prev)));

%Laço iterativo
while(erro > tol && iter < maxit)%Executa até convergir ou atingir o máximo de iterações
    iter=iter+1; % Incrementa iteração

    % Atualiza o potencial dos nós internos pela média dos 4 vizinhos - Eq. Laplace - M.D.F.
    for k=1:size(p);
        [i,j]=ind2sub(size(x),p(k));
        Phi_new(i,j)=(Phi_new(i-1,j)+Phi_new(i+1,j)+Phi_new(i,j-1)+Phi_new(i,j+1))/4;
    end

    % Calcula maximo erro entre Phi_atual e Phi_prev de todo o dominio
    erro=max(max(abs(Phi_new-Phi_prev)));

```

```

eps(iter)=erro;

%Atualiza a matriz de potenciais
Phi_prev=Phi_new;

%Exibe a progressão da solução
%(Execução lenta; use apenas com discretização grosseira)

figure (5);

imagesc(Phi_new);colorbar;

title(['Potenciais na iteracao no. ',int2str(iter),' Erro = ', num2str(erro)],'Color','k');

getframe;

end

niter1=iter;

if (niter1 == maxit && erro > tol)

    disp([' Número máximo de iterações atingido sem convergência :', num2stg(niter1), '
    iterações – Erro: \n', num2str(erro), 'Os resultados podem não ter significado!\n']);

end

%

%

% Problema Dual (para traçado dos Quadrados Curvilíneos

%

% Atribui Condições de Contorno

%

iyDual=find( (y(:, :) < b/1.999) & (y(:, :) > b/2.001) );

iVmaxdual=find( (x(iyDual) > (-0.01)) & (x(iyDual) < (1.0001*g)));

i0=find( (x(iyDual)> (0.9999*(g+c))) & (x(iyDual)< (1.0001*a)) );

xfe=find( x(q(iVmax))< 1.0001*min(x(q(iVmax))) );

xfd=find( x(q(iVmax))> 0.9999*max(x(q(iVmax))) );

yfa=find( y(q(iVmax))> 0.9999*max(y(q(iVmax))) );

yfb=find( y(q(iVmax))< 1.0001*min(y(q(iVmax))) );

for k=1:size(iVmax);

    if ( abs( x(q(iVmax(k)))-min(x(q(iVmax))) ) < tol && abs( y(q(iVmax(k)))-min(y(q(iVmax))) ) <
tol)

        [ieb,jeb]=ind2sub(size(x), q(iVmax(k)));

```

```

elseif (abs( x(q(iVmax(k)))-min(x(q(iVmax))) )< tol && abs( y(q(iVmax(k)))-max(y(q(iVmax)))
)< tol)

    [iea,jea]=ind2sub(size(x), q(iVmax(k)));

    elseif ( abs( x(q(iVmax(k)))-max(x(q(iVmax))) )< tol && abs( y(q(iVmax(k)))-min(y(q(iVmax)))
)< tol)

        [idb,jdb]=ind2sub(size(x), q(iVmax(k)));

        elseif (abs( x(q(iVmax(k)))-max(x(q(iVmax))) )< tol && abs( y(q(iVmax(k)))-max(y(q(iVmax)))
)< tol)

            [ida,jda]=ind2sub(size(x), q(iVmax(k)));

        end
    end

Dual_prev=zeros(size(x));

Dual_new=Dual_prev;

Dual_new(r)= -1;

Dual_new(iFuro)= NaN;

Dual_new(iyDual(iVmaxdual))=Vmax;

Dual_new(iyDual(i0))=Vmin;

p2=find(Dual_new(p) < 0);

Dual_new(r)= start_Dual;

Dual_new(iFuro)= NaN;

Dual_new(iyDual(iVmaxdual))=Vmax;

Dual_new(iyDual(i0))=Vmin;

%Contador de iterações - dual

iter2=0;

% Erro máximo entre Phi_new e Phi_prev (Dual)

erro2=max(max(abs(Dual_new-Dual_prev)));

%Laço iterativo (Problema Dual)

while(erro2 > 10*tol && iter2 < maxit)%Executa até convergir ou atingir o máximo de iterações

    iter2=iter2+1; % Incrementa iteração

    %Atualiza o potencial das fronteiras

    Dual_new(1,:)=Dual_prev(2,:);

    Dual_new(Ny,:)=Dual_prev(Ny-1,:);

```

```

Dual_new(:,1)=Dual_prev(:,2);
Dual_new(2:Ny-1,Nx)=Dual_prev(2:Ny-1,Nx-1);
for k=2:size(xfe)-1
    [ie,je]=ind2sub(size(Dual_new), q(iVmax(xfe(k))));
    Dual_new(ie,je)=Dual_new(ie,je-1);
end
for k=2:size(xfd)-1
    [id,jd]=ind2sub(size(Dual_new), q(iVmax(xfd(k))));
    Dual_new(id,jd)=Dual_new(id,jd+1);
end
for k=2:size(yfb)-1
    [ib,jb]=ind2sub(size(Dual_new), q(iVmax(yfb(k))));
    Dual_new(ib,jb)=Dual_new(ib-1,jb);
end
for k=2:size(yfa)-1
    [ia,ja]=ind2sub(size(Dual_new), q(iVmax(yfa(k))));
    Dual_new(ia,ja)=Dual_new(ia+1,ja);
end
Dual_new(iyDual(iVmaxdual))=Vmax;
Dual_new(iyDual(i0))=Vmin;
%
% Atualiza o potencial dos nós internos pela média dos 4 vizinhos - Eq. Laplace - M.D.F.
for k=1:size(p2);
    [i,j]=ind2sub(size(x),p(p2(k)));
    Dual_new(i,j)=(Dual_new(i-1,j)+Dual_new(i+1,j)+Dual_new(i,j-1)+Dual_new(i,j+1))/4;
end
%Cantos
Dual_new(ieb,jeb)=(Dual_new(ieb-1,jeb)+Dual_new(ieb+1,jeb)+Dual_new(ieb,jeb-1)+Dual_new(ieb,jeb+1))/4;
Dual_new(iea,jea)=(Dual_new(iea-1,jea)+Dual_new(iea+1,jea)+Dual_new(iea,jea-1)+Dual_new(iea,jea+1))/4;

```

```

Dual_new(idb,jdb)=(Dual_new(idb-1,jdb)+Dual_new(idb+1,jdb)+Dual_new(idb,jdb-
1)+Dual_new(idb,jdb+1))/4;

Dual_new(ida,jda)=(Dual_new(ida-1,jda)+Dual_new(ida+1,jda)+Dual_new(ida,jda-
1)+Dual_new(ida,jda+1))/4;

% Calcula maximo erro entre Phi_atual e Phi_prev de todo o dominio
erro2=max(max(abs(Dual_new-Dual_prev)));

eps2(iter2)=erro2;

%Atualiza a matriz de potenciais
Dual_prev=Dual_new;

%
%Exibe a progressão da solução (execução do programa mais lenta!)

figure (6);

imagesc(Dual_new);colormap(cool);colorbar;

title(['Potenciais na iteracao no. ',int2str(iter2),' Erro = ', num2str(erro2)],'Color','k');

getframe;

end

niter2=iter2;

if (niter2 == maxit && erro2 > 10*tol)

    disp([' Número máximo de iterações atingido sem convergência: ', num2str(niter2), '
    iterações – Erro: \n', num2str(erro2), 'Interprete este resultado com ressalvas!\n']);

end

%
%Evolução das Iterações

figure (7)

% Traça a evolução das iterações

grid on

semilogy(eps, '*'); xlabel('Iterações'); ylabel('Erro');xlim([0,niter1]);

title('Evolução das Iterações')

hold on

semilogy(eps2, 'ro');xlim([0,max(niter1,niter2)]);

legend ('Original','Dual');

hold off

```



```

%
% Corrente Total
%
Somat=sum(Phi_new(2,:))+sum(Phi_new(Ny-1,:))+sum(Phi_new(:,2))+sum(Phi_new(:,Nx-1));
I= (sigma*(10^-3)*Somat)/2;

%
% Resistencia
%
R= (Vmax-Vmin)/I;
%
% Capacitancia
%
Cap= (eps0*epsr)/(sigma*(10^-3)*R) ;
%
% Resistencia dual
%
Rdual= 1/(2*sigma*sigma_dual*(10^-6)*R) ;
% Densidade de carga:
Dn=[Phi_new(2,1:Nx-1),Phi_new(1:Ny-1,Nx-1)',Phi_new(Ny-1,1:Nx-1),Phi_new(1:Ny-
1,2)']*epsr*eps0/dx*100;
ol=(1:length(Dn))-1;
figure(8), plot(ol*dx,Dn), xlabel('l (m)'), ylabel(' Dn (C/m^2) ');
%
% Densidade Superficial de Carga Mínima
%
Rho_s_min = min(-Dn);
%
figure (7);
%
% Traçado dos vetores de campo eletrico (apenas para visualização!)

```

```

[ex,ey]=gradient(Phi_new);
scale=2;
Q=quiver(x,y,-ex,-ey, scale);
color = Q.Color;
Q.Color = 'red';
%
% Equipotenciais
%
V=0:10:Vmax;
V(1)=1e-6; V(length(V))=V(length(V))-1e-6;
[C,H]=contour( Phi_new, length(V) );
clabel(C,V);
axis('equal');
hold on
%
% Equipotencias - Problema Dual (para traçado dos quadrados curvilíneos)
%
sp= 1/(R*sigma*(10^-3)) ;
ntubos = 5/sp;
disp(['Num. tubos = ',num2str(ntubos)]);
dV = Vmax/ntubos;
V = 0:dV:Vmax;
V(1)=1e-6; V(length(V))=V(length(V))-1e-6;
[C,H]= contour( Dual_new, length(V) );
axis('equal');
hold off
%
% Impressão de resultados >>> Atenção para as unidades !!!!<<<<
%
disp('EP - 1 : ');
fprintf('b= %d c= %d d= %d g= %d h= %1.1g (valores em cm)\n', b,c,d,g,h);

```

```

fprintf('eps_r= %1.1g  Sigma = %1.1g mS/m  Sigma_dual = %1.1g mS/m \n',
epsr,sigma,sigma_dual);

disp(['Densidade superficial de Carga mínima = ',num2str( Rho_s_min/(10^-9)), ' nC/m^2']);

disp(['Resistência = ',num2str( R ), ' ohms  ']);

disp(['Capacitância = ',num2str( Cap/(10^-12) ), ' pF  ']);

disp(['Resistência Dual = ',num2str( Rdual ), ' ohms  ']);

%

% FIM

%
```