Biomatemática ***26*** *(2016), 65–80* ***ISSN 1679-365X***
Uma Publicação do Grupo de Biomatemática IMECC – UNICAMP

Uso de simulações computacionais para o ensino de condições de contorno homogêneas em problemas de difusão

Tiago Yuzo Miyaoka[1], João F. C. A. Meyer[2]
DMA, IMECC – UNICAMP, Campinas/SP.

Juliana Marta R. de Souza[3]
DM, IMECC – UNICAMP, Campinas/SP.

Resumo. Neste trabalho uma implementação de aproximação numérica para a solução de um Problema de Valor de Contorno e Inicial Difusivo, com condições de contorno homogêneas que podem ser determinadas pelo usuário, serve de ferramenta para introduzir, explicar ou esclarecer a relação entre os diferentes tipos de condições de contorno homogêneas que mais frequentemente aparecem na literatura, a saber, Dirichlet, Neumann e Robin. A expectativa é de que, diante da materialização das soluções dos PVIC's, os estudantes possam também compreender qual é o papel e o significado intuitivo de cada tipo de condição de contorno analisada. Para tanto, trazemos recortes gráficos do que, no ambiente da sala de aula pode ser reproduzido na forma de um filme. A rotina computacional, obtida via Método de Diferenças Finitas de segunda ordem e implementada em MATLAB, disponibilizada em anexo, permite que eles próprios modifiquem os parâmetros do PVIC e observem seu impacto nas soluções.

Palavras-chave*: Equações Diferenciais Parciais, Condições de Contorno, Métodos Numéricos, Ensino-Aprendizagem.*

[1]tiagoyuzo@gmail.com
[2]joni@ime.unicamp.br
[3]jumarta@gmail.com

1. Introdução

Problemas biológicos, ambientais, sociais e físicos são frequentemente modelados por meio de Equações Diferenciais Parciais (EDP's) ou sistemas de EDP's. Tais EDP's, ou sistemas de EDP's, quando acompanhadas(os) de condições de contorno e de condição inicial, resultam em um Problema de Valor Inicial e de Contorno (PVIC). Alunos de graduação da área de Ciências Exatas, portanto, deparam-se, ao longo de sua formação, com disciplinas acerca da resolução analítica ou numérica de PVIC's bem como se o deparam com aprendizado de técnicas de modelagem utilizando EDP's.

Uma das primeiras Equações Diferenciais Parciais que surgem na vida do estudante de Matemática Pura ou Aplicada, Física ou Engenharia é a Equação do Calor, ou Equação de Difusão (Butkov e de Carvalho, 1988). Mesmo em disciplinas mais avançadas, esta equação tem papel importante, como sua aplicação à modelos de dispersão populacional (Edelstein-Keshet, 1987), por exemplo. Neste trabalho apresentamos três PVIC's compostos por uma equação de difusão, condição inicial e condições de contorno: de Dirichlet, Neumann ou Robin, sempre homogêneas. Partimos deste problema como trampolim para, via aproximação numérica da solução, obtida pelo Método de Diferenças Finitas de segunda ordem, obter uma rotina computacional que, através de seus resultados gráficos, permita aos alunos se apropriarem dos significados e implicações de cada condição de contorno.

2. Modelagem Matemática

Seja $u(x, y, t)$ a densidade de uma população, ou concentração de uma substância, ou uma temperatura, que se difunde. Sua modelagem é feita a partir da equação de difusão (Cantrell e Cosner, 2004), isto é:

$$\frac{\partial u}{\partial t} = \nabla \cdot (\alpha(x, y)\nabla u) + f(x, y) \tag{1}$$

onde: o operador ∇ é calculado somente nas variáveis espaciais x e y; $\alpha(x, y)$ é o coeficiente de difusão, ou difusividade térmica, aqui tomada constante, e $f(x, y)$ é uma fonte de concentração externa ao meio que neste trabalho tomaremos como identicamente nula.

O domínio espacial é uma região aberta e limitada $\Omega \in \mathbb{R}^2$ com contorno $\partial\Omega$ e o domínio temporal é dado por $I = (t_0, t_f]$. Além disso, $u(x, y, t_0) =$

$u_0(x, y)$ é uma condição inicial arbitrária (a utilizada neste trabalho pode ser encontrada na seção 4). A condição de contorno homogênea mais geral para esta equação é, para $(x, y) \in \partial\Omega$ e $t \in I$ (Cantrell e Cosner, 2004, p. 30):

$$\alpha \frac{\partial u}{\partial \vec{n}} + \beta u = 0 \tag{2}$$

onde $\vec{n}$ denota o vetor normal à fronteira do domínio, "para fora", e β é uma função relacionada ao fluxo de saída na fronteira, que será tomada positiva e constante. Esta condição que envolve a derivada e o próprio valor da função $u(x, y, t)$, é dita de Robin ou de terceira espécie. Para analisá-la melhor vamos escrever a equação (1) em função do fluxo de $u(x, y, t)$ à fronteira. Pela lei de Fick (Okubo e Levin, 2013) temos que este fluxo é dado por: $\vec{J}(u) = -\alpha \nabla u$, e assim a equação (1) se torna:

$$\frac{\partial u}{\partial t} = -\nabla \cdot \vec{J}(u) + f.$$

E a condição de contorno (2) é escrita, então, como:

$$\vec{J}(u) \cdot \vec{n} = \beta u. \tag{3}$$

Escrito dessa forma, podemos ver que a condição de contorno relaciona o fluxo que atravessa o bordo do domínio com a densidade ali presente, sendo β um parâmetro que regula a taxa de saída.

Se $\beta = 0$, temos:

$$\frac{\partial u}{\partial \vec{n}} = 0 \tag{4}$$

Esta condição nos diz que não há fluxo no bordo. Então $\partial\Omega$ barra totalmente o fluxo difusivo e a condição é dita de Neumann homogênea.

Voltando a (2) e (3), se aumentarmos o valor de β, o fluxo para fora do domínio também aumenta, fazendo com que haja uma variação de densidade para fora do domínio.

Escrevendo (3) como $u = (1/\beta)\vec{J} \cdot \vec{n}$ podemos ver que, se $\beta \to \infty$ então:

$$u(x, y, t) = 0 \tag{5}$$

A condição (5) é chamada de Dirichlet homogênea. Esta condição indica que toda densidade que chega à fronteira passa por ela imediatamente, mantendo o valor da densidade em zero no bordo.

Para melhor compreender os efeitos de cada tipo de condição de contorno, utilizamos o conceito de massa total do sistema, que é uma função

dependente do tempo, e é obtida integrando a densidade $u(x, y, t)$ em todo domínio Ω (Stewart, 2013):

$$M(t) = \iint_{\Omega} u(x, y, t)\ dydx. \tag{6}$$

Considerando a equação (1) com fonte nula, e integrando-a em Ω:

$$\iint_{\Omega} \frac{\partial u}{\partial t}\ dydx = \iint_{\Omega} \nabla \cdot (\alpha \nabla u)\ dydx.$$

Assumindo $u(x, y, t)$ uniformemente contínua podemos trocar a ordem da derivada temporal com a integral. Aplicando o Teorema da Divergência (Stewart, 2013) no termo à direita obtemos:

$$\frac{\partial}{\partial t} \iint_{\Omega} u\ dydx = \oint_{\partial\Omega} \alpha \nabla u \cdot \vec{n}\ dydx.$$

Observando que a integral do lado esquerdo é a massa do sistema e reescrevendo a derivada direcional $\nabla u \cdot \vec{n}$:

$$\frac{\partial M}{\partial t} = \oint_{\partial\Omega} \alpha \frac{\partial u}{\partial \vec{n}}\ dydx. \tag{7}$$

A equação acima nos diz que a variação temporal da massa depende do que ocorre no contorno, o que condiz com a intuição, visto que como a fonte f é nula, a solução $u(x, y, t)$ depende apenas da condição inicial e da condição de contorno. Substituindo a condição geral (2) em (7):

$$\frac{\partial M}{\partial t} = -\oint_{\partial\Omega} \beta u\ dydx.$$

como β e u assumem somente valores positivos, a equação acima nos diz que a variação temporal da massa é negativa, isto é: a massa decai com o tempo. Quanto maior o valor de β, maior é o decaimento, ou seja, maior é a saída de concentração pelo contorno, de acordo com a análise em (3). Se $\beta = 0$, a variação da massa é nula, o que condiz com a condição (4), fluxo de saída nulo. Quanto maior o valor de β, maior a variação da massa e do fluxo à fronteira, condizendo no limite à condição de Dirichlet (5).

3. Métodos Numéricos

De forma a obter soluções numéricas para o problema (1) utilizamos o Método de Diferenças Finitas, tanto na variável espacial, com fórmulas centradas de segunda ordem; quanto na temporal, com o Método de Crank-Nicolson

(Cunha, 2003; LeVeque, 2007). Por simplicidade consideramos um domínio retangular $\Omega = [0, L] \times [0, H]$, e um intervalo de tempo $I = [0, t_f]$. As fórmulas de diferenças utilizadas são:

$$\begin{aligned} \frac{\partial u_{i,j}^n}{\partial x} &\approx \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2\Delta x}, \ \frac{\partial^2 u_{i,j}^n}{\partial x^2} \approx \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} \\ \frac{\partial u_{i,j}^{n+\frac{1}{2}}}{\partial t} &\approx \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t}, \ u_{i,j}^{n+\frac{1}{2}} \approx \frac{u_{i,j}^{n+1} + u_{i,j}^n}{2} \end{aligned} \tag{8}$$

onde $u_{i,j}^n = u(x_i, y_j, t_n)$, $i = 1, \ldots, n_x$, $j = 1, \ldots, n_y$, $n = 1, \ldots, n_t$, sendo n_x, n_y e n_t os números de nós nas malhas discretizadas em x, y e t, respectivamente, e com $u(x, y, t) \in \mathcal{C}^4(\Omega)$.

Para os pontos do contorno do domínio, aplicamos as fórmulas de diferenças diretamente nas condições (2) de modo a obter equações complementares (Cunha, 2003; LeVeque, 2007). Com essas fórmulas temos então um sistema linear algébrico de equações a ser resolvido para cada passo no tempo $n = 1, \ldots, n_t$, que pode ser escrito na forma matricial como:

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{b} \tag{9}$$

Este sistema pode ser resolvido pelo método da fatoração *LU* (Cunha, 2003), fatorando a matriz $\mathbf{A}$ uma única vez e utilizando a fatoração em todas a iterações temporais.

Para calcular aproximadamente a massa total do sistema (6), utilizamos uma fórmula de Simpson repetida (Cunha, 2003) nos mesmos pontos da malha espacial do Método de Diferenças Finitas.

4. Simulações Computacionais

Com base no método descrito na seção anterior, implementamos um algoritmo em Matlab, por seu caráter didático, de modo a obter soluções numéricas para o problema (1) – (2) que ilustrem bem os comportamentos dos diferentes tipos de condições de contorno mencionados anteriormente. O algoritmo está disponível em anexo.

Os parâmetros utilizados nos experimentos que mostramos aqui foram: $\alpha = 0.2$ km^2/dia, $f(x, y) = 0$, $\beta = 0$ km/dia, $\beta = 0.5$ km/dia ou $\beta = 100$ km/dia, $L = 1$ km, $H = 1$ km, $nx = ny = 2^6$, $nt = 2^{10}$, $t_f = 1$ dia e como condição inicial $u_0(x, y) = \kappa \exp(-64((x-0.25)^2+(y-0.25)^2))$, uma gaussiana,

ilustrada na Figura 1, onde κ é uma constante de forma a manter a massa inicial do sistema unitária. Em um ambiente de ensino, estes parâmetros podem, e devem, ser modificados pelo professor e pelos alunos a seu gosto.

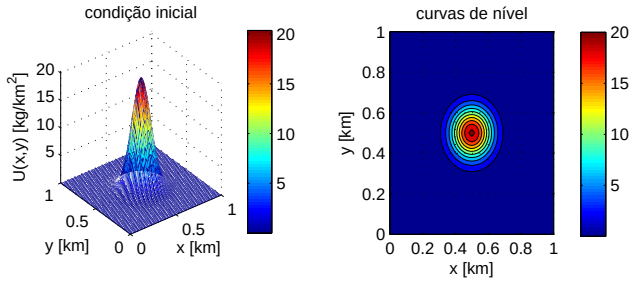

Figura 1: Condição inicial para o problema (1) - (2), $u_0(x, y) = \kappa \exp(-64((x - 0.25)^2 + (y - 0.25)^2))$.

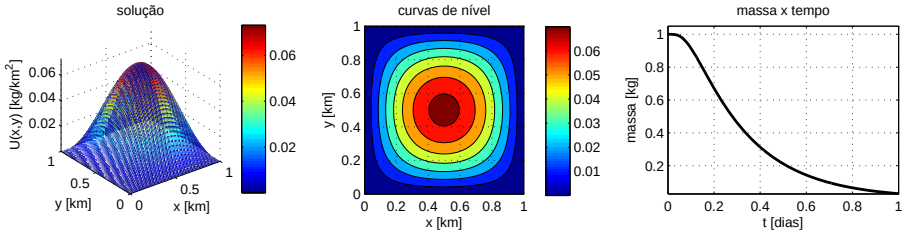

Figura 2: Solução numérica para o problema (1) - (2), $t_f = 1$ dia, $\beta = 100$ km/dia, Condição de Dirichlet homogênea.

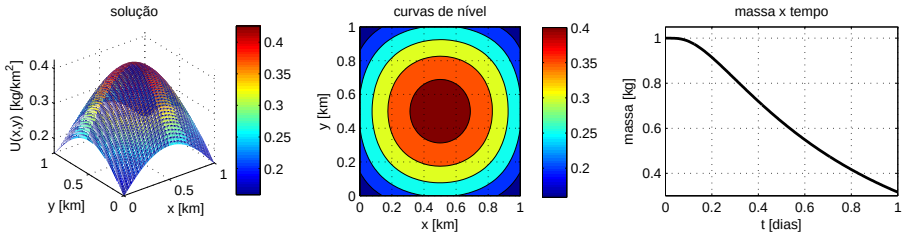

Figura 3: Solução numérica para o problema (1) - (2), $t_f = 1$ dia, $\beta = 0.5$ km/dia, Condição de Robin com fluxo de saída médio.

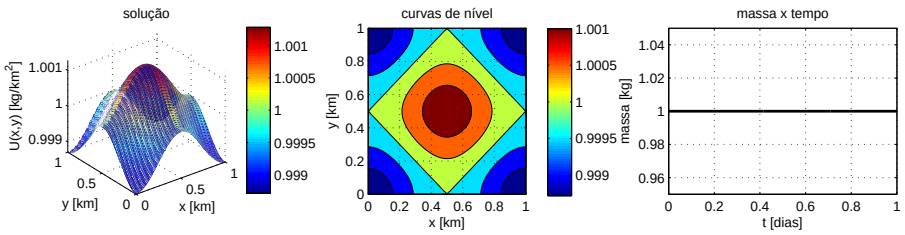

Figura 4: Solução numérica para o problema (1) - (2), $t_f = 1$ dia, $\beta = 0$ km/dia, Condição de Neumann com fluxo de saída nulo.

Apresentamos os resultados nas Figuras 2 a 4: à esquerda das figuras temos as superfícies das soluções, ao centro suas curvas de nível, ambos em $t = 1$ dia, e à direita gráficos da massa dada por (6) pelo tempo t. Para a análise dos resultados, é importante observar a diferença nas escalas dos gráficos.

Como ensejado, quanto maior o valor de β maior o fluxo à fronteira, assim na Figura 2, onde $\beta = 100$ km/dia a concentração se anula no bordo e a massa do sistema decai rapidamente conforme a solução atinge a fronteira.

Na Figura 3 o valor de β está bastante menor do que 100, $\beta = 0.5$ km/dia mas ainda é positivo, observamos, então uma saída à fronteira, mas esta não é total, logo a concentração não se anula no bordo e o decrescimento da massa é menos acentuado.

Por fim, na Figura 4, $\beta = 0$ km/dia, logo não há saída de concentração pela fronteira, vemos as concentrações no bordo aumentarem bastante e a massa total do sistema se mantém constante.

5. Conclusão

Este trabalho se apoia em simulações numéricas como meio para explicar e mostrar, na prática, enquanto a simulação acontece, o impacto de diferentes Condições de Contorno homogêneas sobre o Problema de Difusão. Neste formato de artigo, só é possível ver retratos estáticos das soluções, como muitos livros podem trazer, mas quem dispuser da rotina pode modificar os parâmetros e executá-la a qualquer momento, em qualquer ambiente de ensino que disponha de um computador com Matlab; pode distribuir o código aos alunos e pedir trabalhos que requeiram variações em PVIC's e interpretações das diferenças

acarretadas na solução. Por exemplo, após uma apresentação como esta o professor pode perguntas à turma: *O que acontecerá se* $\beta < 0$*?* Claro que é possível fazer o mesmo em outras linguagens computacionais, bastando para tanto adequar o código, o que pode exigir um pouco mais de conhecimento em linguagens de programação. Uma possibilidade menos interativa mas que também não requer o uso do *software* Matlab é gravar os vídeos referentes aos diferentes tipos de condições de contorno. Disponibilizamos alguns vídeos *online* (Miyaoka, 2016), que podem ser distribuídos para e/ou compartilhados com os alunos. Entendemos que este é um método eficiente de inserir conceitos matemáticos pertinentes ao nível de Bacharelado ou Licenciatura no cotidiano, até mesmo virtual dos alunos, e isto pode tornar a Matemática mais palpável, mais atraente e mais interessante.

Agradecimentos

Os autores agradecem à CAPES.

Referências

Butkov, E. e de Carvalho, J. B. P. F. (1988). *Física matemática*. Livros Técnicos e Científicos, Rio de Janeiro.

Cantrell, R. S. e Cosner, C. (2004). *Spatial ecology via reaction-diffusion equations*. John Wiley & Sons.

Cunha, M. C. C. (2003). *Métodos numéricos*. Editora da UNICAMP.

Edelstein-Keshet, L. (1987). *Mathematical models in biology*, volume 46. Siam.

LeVeque, R. J. (2007). *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, volume 98. Siam.

Miyaoka, T. Y. (2016). Vídeos *online*. `http://www.youtube.com/playlist?list=PL9vXgVs5TAkyRND7vJwPesgLQ0VCxZRDo`. Acessado pela última vez em: 18/05/2016.

Okubo, A. e Levin, S. A. (2013). *Diffusion and ecological problems: modern perspectives*, volume 14. Springer Science & Business Media.

Stewart, J. (2013). *Cálculo – Volume 2.* Cengage Learning, São Paulo.

Códigos

```
%-----------------------------------------------------------------------
function condcont
%-----------------------------------------------------------------------
%   obtem uma solucao numerica para o problema:
%
%   du/dt = alpha div(grad u)
%
%   - alpha du/dn = beta u
%
%   onde o dominio e  um retangulo [0,Lx] x [0,Ly]
%-----------------------------------------------------------------------
global dx dy dt x y t alpha beta
%-----------------------------------------------------------------------
%   parametros
%-----------------------------------------------------------------------
[nx,ny,nt,Lx,Ly,Tf,alpha,beta] = parametros;
%-----------------------------------------------------------------------
%   malha
%-----------------------------------------------------------------------
[x,y,t,dx,dy,dt,nnx,nny,nn] = malha(nx,ny,nt,Lx,Ly,Tf);
%-----------------------------------------------------------------------
%   construcao do sistema
%-----------------------------------------------------------------------
[A,B] = sistema(nn,nnx,nny,nx);
%-----------------------------------------------------------------------
%   condicoes iniciais
%-----------------------------------------------------------------------
u = zeros(nn,1);
veru = zeros(nny,nnx);
massa = zeros(nt+1,1);
%
for i = 1:nnx
    for j = 1:nny
        k = (i - 1)*nny + j;
        u(k) = 100*exp(-64*((0.5-x(i))^2+(0.5-y(j))^2))/4.9087383698276;
        veru(j,i) = u(k);
    end
end
%
massa(1) = simpson2d(veru,0,Lx(2),0,Ly(2));
%-----------------------------------------------------------------------
%   grafico da condicao inicial
%-----------------------------------------------------------------------
figure
subplot(1,2,1)
surf(x,y,veru);
%
```

```
subplot(1,2,2)
contourf(x,y,veru);
%-----------------------------------------------------------------------
%   fatoracao LU de A
[L,U] = lu(A);
%-----------------------------------------------------------------------
%   iteracoes temporais
%-----------------------------------------------------------------------
for it = 1:nt
    c = B*u;
    u = U\(L\c);
%   construcao do grafico
    for j = 1:nny
        for i = 1:nnx
            k = (i - 1)*nny + j;
            veru(j,i) = u(k);
        end
    end
    massa(it+1) = simpson2d(veru,0,Lx(2),0,Ly(2)); % para o graf da massa
end
%-----------------------------------------------------------------------
figure
subplot(1,3,1)
surf(x,y,veru);
%
subplot(1,3,2)
contourf(x,y,veru);
%
subplot(1,3,3)
plot(t,massa,  k- ,  LineWidth , 2)
%-----------------------------------------------------------------------
%   definicao das funcoes
%-----------------------------------------------------------------------
%   parametros do modelo
%-----------------------------------------------------------------------
function [nx,ny,nt,Lx,Ly,Tf,alpha,beta] = parametros
%-----------------------------------------------------------------------
nx = 2^6; % numero de divisoes em x
ny = nx; % numero de divisoes em y
%
Lx(1) = 0; % lado do retangulo na direcao x
Lx(2) = 1; %lado do retangulo na direcao y
Ly(1) = Lx(1);
Ly(2) = Lx(2);
%
nt = 2^10; % numero de divisoes em t
Tf = 1; % tempo final
%
alpha = 0.2; % coeficiente de difusao
beta = 100; % constante do contorno
%-----------------------------------------------------------------------
end % function parametros
%-----------------------------------------------------------------------
%   malha - construcao da malha
```

```
%-----------------------------------------------------------------------
function [x,y,t,dx,dy,dt,nnx,nny,nn] = malha(nx,ny,nt,Lx,Ly,Tf)
%-----------------------------------------------------------------------
dx = (Lx(2) - Lx(1))/nx; % delta x
dy = (Ly(2) - Ly(1))/ny; % delta y
dt = Tf/nt; % delta t
%
nnx = nx + 1; % numero de nos na direcao x
nny = ny + 1; % numero de nos na direcao y
%
nn = nnx*nny; % numero de nos totais na malha
%-----------------------------------------------------------------------
x = zeros(nnx,1); % vetor com os valores de x
y = zeros(nny,1); % vetor com os valores de y
t = zeros(nt,1); % vetor com os valores de t
%
x(1,1) = Lx(1);
y(1,1) = Ly(1);
%-----------------------------------------------------------------------
for i = 2:nnx
    x(i,1) = x(i-1,1) + dx;
end
%
for i = 2:nny
    y(i,1) = y(i-1,1) + dy;
end
%
for i = 2:nt+1
    t(i,1) = t(i-1,1) + dt;
end
%-----------------------------------------------------------------------
end % function malha
%-----------------------------------------------------------------------
%   sistema
%-----------------------------------------------------------------------
function [A,B] = sistema(nn,nnx,nny,nx)
%-----------------------------------------------------------------------
A = zeros(nn,nn); % matriz do lado esquerdo do sistema
B = zeros(nn,nn); % matriz do lado direito do sistema
%
cteA = zeros(5,1);
cteB = zeros(5,1);
%
c1 = (alpha/dy)/(2*dy)*dt;
c2 = (alpha/dx)/(2*dx)*dt;
c3 = (-alpha/(dx*dx) -alpha/(dx*dx))*dt;
c4 = (alpha/dx)/(2*dx)*dt;
c5 = (alpha/dy)/(2*dy)*dt;
%
cteA(1) = -c1;
cteA(2) = -c2;
cteA(3) = 1 - c3;
cteA(4) = -c4;
cteA(5) = -c5;
```

```
%
cteB(1) = c1;
cteB(2) = c2;
cteB(3) = 1 + c3;
cteB(4) = c4;
cteB(5) = c5;
%-------------------------------------------------------------------------
%   sistema - pontos internos do dominio
%-------------------------------------------------------------------------
for i = 1:nn
    A(i,i) = cteA(3);
    B(i,i) = cteB(3);
end
%
for i = 1:nn-1
    A(i+1,i) = cteA(1);
    A(i,i+1) = cteA(5);
    B(i+1,i) = cteB(1);
    B(i,i+1) = cteB(5);
end
%
for i = 1:nn-nny
    A(i+nny,i) = cteA(2);
    A(i,i+nny) = cteA(4);
    B(i+nny,i) = cteB(2);
    B(i,i+nny) = cteB(4);
end
%-------------------------------------------------------------------------
%   contorno
%-------------------------------------------------------------------------
c = -2*dx/alpha*beta;
A(1,1) = cteA(3) + cteA(2)*c;
B(1,1) = cteB(3) + cteB(2)*c;
for i = 2:nx
    A(i,i) = cteA(3) + cteA(2)*c;
    A(i,i+nny) = cteA(4) + cteA(2);
    B(i,i) = cteB(3) + cteB(2)*c;
    B(i,i+nny) = cteB(4) + cteB(2);
end
A(nnx,nnx+1) = 0;
B(nnx,nnx+1) = 0;
%-------------------------------------------------------------------------
c = -2*dy/alpha*beta;
A(1,1) = cteA(3) + cteA(1)*c;
A(1,2) = cteA(5) + cteA(1);
B(1,1) = cteB(3) + cteB(1)*c;
B(1,2) = cteB(5) + cteB(1);
for i = 1:nx
    j = i*nny + 1;
    A(j,j) = cteA(3) + cteA(1)*c;
    A(j,j+1) = cteA(5) + cteA(1);
    A(j,j-1) = 0;
    B(j,j) = cteB(3) + cteB(1)*c;
    B(j,j+1) = cteB(5) + cteB(1);
```

```
    B(j,j-1) = 0;
end
%-------------------------------------------------------------------------
c = -2*dx/alpha*beta;
for i = 1:nny
    j = nx*nny + i;
    A(j,j) = cteA(3) + cteA(4)*c;
    A(j,j-nny) = cteA(2) + cteA(4);
    B(j,j) = cteB(3) + cteB(4)*c;
    B(j,j-nny) = cteB(2) + cteB(4);
end
%-------------------------------------------------------------------------
c = -2*dy/alpha*beta;
A(nn,nn) = cteA(3) + cteA(5)*c;
A(nn,nn-1) = cteA(1) + cteA(5);
B(nn,nn) = cteB(3) + cteB(5)*c;
B(nn,nn-1) = cteB(1) + cteB(5);
for i = 1:nx
    j = i*nny;
    A(j,j) = cteA(3) + cteA(5)*c;
    A(j,j-1) = cteA(1) + cteA(5);
    A(j,j+1) = 0;
    B(j,j) = cteB(3) + cteB(5)*c;
    B(j,j-1) = cteB(1) + cteB(5);
    B(j,j+1) = 0;
end
%-------------------------------------------------------------------------
%   casos particulares - pontos nos cantos do retangulo
%-------------------------------------------------------------------------
c2 = -2*dx/alpha*beta;
%
A(1,1) = cteA(3) + cteA(2)*c2;
A(1,1+nny) = cteA(4) + cteA(2);
A(1,2) = cteA(5) + cteA(1);
B(1,1) = cteB(3) + cteB(2)*c2;
B(1,1+nny) = cteB(4) + cteB(2);
B(1,2) = cteB(5) + cteB(1);
%-------------------------------------------------------------------------
c1 = -2*dy*beta/alpha;
%
ind = nny*nx+1;
A(ind,ind-nny) = cteA(2) + cteA(4);
A(ind,ind) = cteA(3) + cteA(1)*c1;
A(ind,ind+1) = cteA(5) + cteA(1);
B(ind,ind-nny) = cteB(2) + cteB(4);
B(ind,ind) = cteB(3) + cteB(1)*c1;
B(ind,ind+1) = cteB(5) + cteB(1);
%-------------------------------------------------------------------------
c1 = -2*dx/alpha*beta;
%
A(nn,nn-1) = cteA(1) + cteA(5);
A(nn,nn-nny) = cteA(2) + cteA(4);
A(nn,nn) = cteA(3) + cteA(4)*c1;
B(nn,nn-1) = cteB(1) + cteB(5);
```

```
B(nn,nn-nny) = cteB(2) + cteB(4);
B(nn,nn) = cteB(3) + cteB(4)*c1;
%-----------------------------------------------------------------------
c1 = -2*dx/alpha*beta;
%
A(nny,nny-1) = cteA(1) + cteA(5);
A(nny,nny) = cteA(3) + cteA(2)*c1;
A(nny,nny+nny) = cteA(4) + cteA(2);
B(nny,nny-1) = cteB(1) + cteB(5);
B(nny,nny) = cteB(3) + cteB(2)*c1;
B(nny,nny+nny) = cteB(4) + cteB(2);
%-----------------------------------------------------------------------
A = sparse(A);
B = sparse(B);
%-----------------------------------------------------------------------
end % function sistema
%-----------------------------------------------------------------------
function integral = simpson2d(f,ax,bx,ay,by)
%-----------------------------------------------------------------------
num = length(f);
%
hx = (bx-ax)/(num-1);
hy = (by-ay)/(num-1);
h = hx * hy / 9;
%-----------------------------------------------------------------------
% avalia os coeficientes
%-----------------------------------------------------------------------
sc = 2*ones(num,1);
%
for i = 2:2:(num-1)
    sc(i) = 4;
end
%
sc(1) = 1;
sc(num) = 1;
%
scx = zeros(num,num);
for i = 1:num
    for j = 1:num
        scx(i,j) = sc(j);
    end
end
%
scxy = ones(num,num);
%
for i = 2:2:(num-1)
    for j = 1:num
    scxy(i,j) = sc(2)*scx(i,j);
    end
end
%
for i = 3:2:(num-2)
    for j = 1:num
    scxy(i,j) = sc(3)*scx(i,j);
```

```
    end
end
%
for j = 1:num
    scxy(1,j) = sc(j);
    scxy(num,j) = sc(j);
end
%-------------------------------------------------------------------------
% avalia a integral
%-------------------------------------------------------------------------
integral = 0;
for i = 1:num
    for j = 1:num
        integral = integral + h*scxy(i,j)*f(i,j);
    end
end
%-------------------------------------------------------------------------
end % function simpson2d
%-------------------------------------------------------------------------
end % function condcont
%-------------------------------------------------------------------------
```