Solving Diffusion Problems

Diffusion phenomena are ubiquitous in science and engineering. For example, diffusion describes the spread of particles through random motion from regions of higher concentration to regions of lower concentration. Consider for instance Oxygen molecules diffusing across cell membranes into cells, and carbon dioxide molecules diffusing out, the diffusion of sugar in a cup or the spread of perfume in a room. The basic equation for the diffusion of a species u is given by:

$$\frac{\partial u}{\partial t} = D\Delta u + S, \tag{1}$$

where D is the diffusion constant, u is the temperature and S is the source term. The same equation is also valid to describe heat conduction through metals as well as other phenomena such as the effect of viscosity in a fluid.

It is possible to solve the heat equation analytically for very special cases, but more often than not, it is necessary to use computer simulations to solve a typical problem in science and engineering. This module describes the numerical approximations of diffusion problems in one and two spatial dimensions and their implementation in MATLAB:

1 The Steady-State Diffusion Equation in 1D

Figure 1: Discretization of a one dimensional domain with $m = 6$ nodes. The solution u at the red nodes will be given the value of the boundary conditions through the function BC, while at the blue nodes the solution will be approximated using (3).

Consider the heat equation in one spatial dimensions:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + S,$$

where D is the diffusion constant, u is the temperature and S is the source term. At steady-state, we have:

$$-D\frac{\partial^2 u}{\partial x^2} = S, \tag{2}$$

which is called the **Poisson** equation. This equation simply gives the temperature distribution when the system is untouched for a very long time, i.e. $t \to \infty$. We assume that the value of the temperature is given on the walls of the domain by a function called BC. The source term is also given by a function S. In order to find a numerical solution, we discretize the computational domain into m nodes, at which we can write an approximation to equation (2), as illustrated in figure 1. Such a numerical approximation can be written as:

$$-D\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} = S_i^n, \tag{3}$$

where $u_i = u(x_i)$ and Δx is the distance between two adjacent grid nodes. This approximation allows to write an equation for each node of the grid i. More precisely, the approximation (3) will give an equation for all interior nodes, whereas for the nodes on the domain's walls, the values of u are simply given by the boundary conditions. Writing these equations gives a linear system:

$$Au = rhs.$$

In the example of the grid in figure 1, if we define the coefficients:

$$C = D\frac{2}{\Delta x^2}, \quad R = -D\frac{1}{\Delta x^2}, \quad L = -D\frac{1}{\Delta x^2},$$

then the linear system is written as:

$$\underbrace{\begin{pmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 & 0 \\ 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 \\ 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 \\ 0 & 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix}}_{u} = \underbrace{\begin{pmatrix} BC_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ BC_6 \end{pmatrix}}_{rhs}$$

A MATLAB code to solve the Poisson equation in one spatial dimension is given next:

```
function outvar=Steady_State_Heat(a,b,BC_a,BC_b,n)
    x=linspace(a,b,n);
    dx=(b-a)/(n-1);

    A=zeros(n,n);
    A(1,1)=1; A(n,n)=1;
    rhs(1)=BC_a; rhs(n)=BC_b;
    for i=2:n-1
        A(i,i)=2/dx/dx; A(i,i-1)=-1/dx/dx; A(i,i+1)=-1/dx/dx;
        rhs(i)=-(x(i)+3)*exp(x(i));
    end

    outvar=A\rhs';

    exact=(x+1).*exp(x);
    plot(x,exact,'r',x,outvar,'bo');
    legend('Exact','Numerical');
    xlabel('x');ylabel('Temperature');
    title('Numerical Solution of the Temperature distribution');
end
```

2 The Diffusion Equation in 1D

Figure 2: Discretization of a one dimensional domain with $m = 6$ nodes. The solution u at the red nodes will be given the value of the boundary conditions through the function BC, while at the blue nodes the solution will be approximated using (5).

Consider the heat equation in one spatial dimensions:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + S, \tag{4}$$

where D is the diffusion constant, u is the temperature and S is the source term. We assume that the value of the temperature is given on the walls of the domain by a function called BC. The source term is also given by a function S. In order to find a numerical solution, we discretize the computational domain into m nodes, at which we can write an approximation to equation (4), as illustrated in figure 2. Such a numerical approximation can be written as:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = D\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} + S_i^n,$$

where $u_i^n = u(x_i, t^n)$, Δx is the given space step and Δt is the given time step. We can rewrite this approximation as:

$$u_i^{n+1} + D\frac{\Delta t}{\Delta x^2}\left(-u_{i+1}^{n+1} + 2u_i^{n+1} - u_{i-1}^{n+1}\right) = u_i^n + \Delta t S_i^n. \tag{5}$$

This approximation allows to write an equation for each grid node i. More precisely, the approximation (5) will give an equation for all interior nodes, whereas for the nodes on the domain's walls, the values of u are simply given by the boundary conditions. Writing these equations gives a linear system:

$$Au^{n+1} = rhs.$$

In the example of the grid in figure 2, if we define the coefficients:

$$C = 1 + 2D\frac{\Delta t}{\Delta x^2}, \quad R = -D\frac{\Delta t}{\Delta x^2}, \quad L = -D\frac{\Delta t}{\Delta x^2},$$

then the linear system is written as:

$$\underbrace{\begin{pmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 & 0 \\ 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 \\ 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 \\ 0 & 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} u_1^n \\ u_2^n \\ u_3^n \\ u_4^n \\ u_5^n \\ u_6^n \end{pmatrix}}_{u^{n+1}} = \underbrace{\begin{pmatrix} BC_1^{n+1} \\ u_2^{n+1} + \Delta t S_2^{n+1} \\ u_3^{n+1} + \Delta t S_3^{n+1} \\ u_4^{n+1} + \Delta t S_4^{n+1} \\ u_5^{n+1} + \Delta t S_5^{n+1} \\ BC_6^{n+1} \end{pmatrix}}_{rhs}$$

A MATLAB code to solve the heat equation in one spatial dimension is given next:

```
function outvar=Heat(a,b,n,dt,t_final)

    aviobj = avifile('My_Movie.avi');
    aviobj.Quality = 100;

    x=linspace(a,b,n);
    dx=(b-a)/(n-1);

    A=zeros(n,n);
    t=0;    Un=cos(x); % Initial Guess.
    while t<t_final % & count<...
        if t+dt>t_final
            dt=t_final-t;
        end
        t=t+dt;
        A(1,1)=1; rhs(1)=cos(a)*exp(-t);
        A(n,n)=1; rhs(n)=cos(b)*exp(-t);
        for i=2:n-1
            A(i,i)=1+2*dt/dx/dx; A(i,i-1)=-dt/dx/dx; A(i,i+1)=-dt/dx/dx;
            rhs(i)=Un(i);
        end

        Unp1=A\rhs';
        Un=Unp1;
        exact=cos(x).*exp(-t);
        plot(x,exact,'r',x,Unp1,'bo'); axis([a b, 0 1]);
        legend('Exact','Numerical');
        xlabel('x');ylabel('Temperature');
        s=sprintf('Numerical Solution of the Temperature distribution at t =%4.2f',t);
        title(s);
%        title('Numerical Solution of the Temperature distribution');
        F=getframe(gca);
        aviobj=addframe(aviobj,F);
        pause(dt);
```

```
    end
    aviobj=close(aviobj);
    outvar=Unp1;
end
```

3 The Steady-State Diffusion Equation in 2D

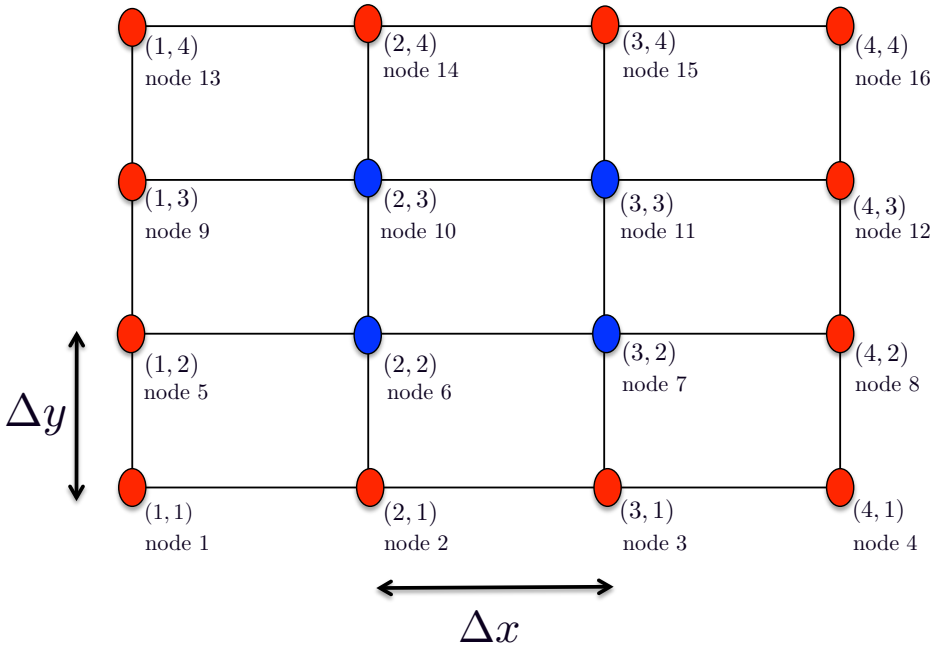

Figure 3: Discretization of a two dimensional domain with $m = 4$ points in the x direction and $n = 4$ points in the y direction. A grid node is referenced by its i and j indices. The i index references the i^{th} location in the x direction, while the j index references the j^{th} location in the y direction. The solution u at red nodes will be given the value of the boundary conditions given by the function BC, while at the blue nodes the solution will be approximated using (7).

Consider the heat equation in two spatial dimensions:

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + S,$$

where D is the diffusion constant, u is the temperature and S is the source term. At steady-state, we have:

$$-D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = S, \tag{6}$$

which is called the **Poisson** equation. We assume that the value of the temperature is given on the walls of the domain by a function called BC. The source term is also given by a function S. In order to find a numerical solution, we discretize the computational domain into m points in the x-direction and n points in the y-direction, as illustrated in figure 3. This gives a grid with $m \times n$ grid nodes, at which we can write an approximation to equation (6). Such a numerical approximation can be written as:

$$-D\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}\right) = S^n_{i,j}, \tag{7}$$

where $u_{i,j} = u(x_i, y_j)$ and Δx and Δy are the given space steps in the x- and y- directions, respectively. This approximation allows to write an equation for each grid node (i, j). We will write these equations, node after node, with the convention that the first equation is written for $(1, 1)$ (we will call this node 1), then for $(1, 2)$ (we will call this node 2), and so on from left to right and bottom to top (see figure 3). The approximation (7) is valid for all interior nodes, whereas for the nodes on the domain's walls the values of u are simply given by the boundary conditions. Writing these equations gives a linear system:

$$Au = rhs.$$

In the example of the grid in figure 3, if we define the coefficients:

$$C = D\left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right), \quad R = -D\frac{1}{\Delta x^2}, \quad L = -D\frac{1}{\Delta x^2}, \quad T = -D\frac{1}{\Delta y^2}, \quad B = -D\frac{1}{\Delta y^2},$$

then the linear system is written as:

$$
\underbrace{\begin{pmatrix}
\mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{B} & 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 & \mathbf{T} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{B} & 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 & \mathbf{T} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{B} & 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 & \mathbf{T} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{B} & 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 & \mathbf{T} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1}
\end{pmatrix}}_{A}
\underbrace{\begin{pmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{4,1} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ u_{4,2} \\ u_{1,3} \\ u_{2,3} \\ u_{3,3} \\ u_{4,3} \\ u_{1,4} \\ u_{2,4} \\ u_{3,4} \\ u_{4,4} \end{pmatrix}}_{u}
=
\underbrace{\begin{pmatrix} BC_{1,1} \\ BC_{2,1} \\ BC_{3,1} \\ BC_{4,1} \\ BC_{1,2} \\ S_{2,2} \\ S_{3,2} \\ BC_{4,2} \\ BC_{1,3} \\ S_{2,3} \\ S_{3,3} \\ BC_{4,3} \\ BC_{1,4} \\ BC_{2,4} \\ BC_{3,4} \\ BC_{4,4} \end{pmatrix}}_{rhs}
$$

Remarks:

1. For each node, one row of the linear system is filled. More precisely, for node p, the p^{th} row of the linear system is filled.

2. Given (i, j), the node number (hence the row of the linear system filled) is $p = (j-1)m + i$.

3. In general, when filling the p^{th} row, the column associated with the coefficient T is $p + m$ while the column associated with the coefficient B is $p - m$.

A MATLAB code to solve the Poisson equation in two spatial dimensions is given next. In particular, note how we code the general linear system associated with the discretization of the Poisson equation in two spatial dimensions:

```
function outvar=Poisson_2D(D,S,BC,a,b,c,d,m,n)

    tic;    % To compute the execution time.

    % Discretize the domain:
    x=linspace(a,b,m);
    y=linspace(c,d,n);
    dx=(b-a)/(m-1);
    dy=(d-c)/(n-1);
    mn=m*n;

    % Initialize the variables:
    A=sparse(mn,mn);    % This is more computationally efficient than A=zeros(mn,mn).

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Assemble the linear system: %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Treat the interior points:
    for i=2:m-1
        for j=2:n-1
            r=(j-1)*m+i;
            A(r,r  )= 2*D/dx/dx + 2*D/dy/dy;
            A(r,r-1)=  -D/dx/dx;
            A(r,r+1)=  -D/dx/dx;
            A(r,r-m)=              -D/dy/dy;
            A(r,r+m)=              -D/dy/dy;
            rhs(r)  = S(x(i),y(j));
        end
    end

    % Treat the boundary conditions:
```

```
for j=1:n
    i=1;        % Left  boundary condition
    r=(j-1)*m+i;
    A(r,r)=1; rhs(r)=BC(a,y(j));
    i=m;        % Right boundary condition
    r=(j-1)*m+i;
    A(r,r)=1; rhs(r)=BC(b,y(j));
end

for i=1:m
    j=1;        % Bottom boundary condition
    r=(j-1)*m+i;
    A(r,r)=1; rhs(r)=BC(x(i),c);
    j=n;        % Top    boundary condition
    r=(j-1)*m+i;
    A(r,r)=1; rhs(r)=BC(x(i),d);
end

% Solve the linear system:
U=A\rhs';

toc;    % To compute the execution time.

% Plot the solution:
for i=1:m
    for j=1:n
        exact(i,j)=BC(x(i),y(j));
    end
end
subplot(2,1,1);
surfl(x,y,exact); shading interp; colormap(summer);
axis([a b, c d, min(min(exact)) max(max(exact))]);
xlabel('x');ylabel('y'); zlabel('Temperature');
title('Exact Solution of the Temperature distribution');
```

```
    subplot(2,1,2);
    % Need to put Un, which is now a vector, into a grid form.
    for i=1:m
        for j=1:n
            r=(j-1)*m+i;
            U_grid(i,j)=U(r);
        end
    end
    surfl(x,y,U_grid); shading interp; colormap(summer);
    axis([a b, c d, min(min(exact)) max(max(exact))]);
    xlabel('x');ylabel('y'); zlabel('Temperature');
    title('Numerical Solution of the Temperature distribution');

    outvar=U;
end
```

An example of how this function is called is:

```
>> BC = @(x,y) cos(2*pi*x)+sin(2*pi*y);
>> S  = @(x,y) 4*pi*pi*cos(2*pi*x)+4*pi*pi*sin(2*pi*y);
>> result=Poisson_2D(1,S,BC,0,1,0,1,80,80);
Elapsed time is 0.750840 seconds.
```

The function will returns the solution of the Poisson equation and store it in the vector `result` and will plot the figure 4.

4 The Diffusion Equation in 2D

Consider the heat equation in two spatial dimensions:

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + S,$$

where D is the diffusion constant, u is the temperature and S is the source term. We assume that the value of the temperature is given on the walls of the domain by a function called BC. In order to find a numerical solution, we discretize the computational domain into m points in the x direction and n

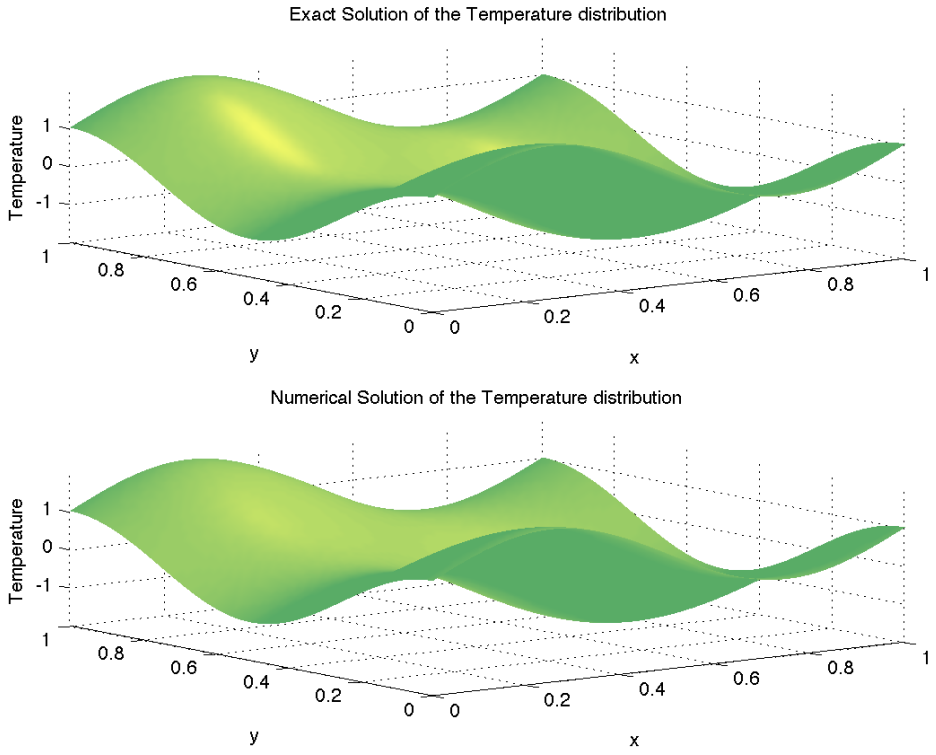

Figure 4: Graphical representation of the solution of the Poisson equation in two spatial dimensions.

points in the y direction, as illustrated in figure 5. This gives a grid with $m \times n$ grid nodes, at which we can write an approximation to equation (8). Such a numerical approximation can be written as:

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = D\left(\frac{u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2}\right) = S_{i,j}^{n},$$

where $u_{i,j}^{n} = u(x_i, y_j, t^n)$, Δx and Δy are the given space steps in the x- and y- directions, respectively and Δt is the given time step. We can rewrite this approximation as:

$$u_{i,j}^{n+1} + D\frac{\Delta t}{\Delta x^2}\left(-u_{i+1,j}^{n+1} + 2u_{i,j}^{n+1} - u_{i-1,j}^{n+1}\right) + D\frac{\Delta t}{\Delta y^2}\left(-u_{i,j+1}^{n+1} + 2u_{i,j}^{n+1} - u_{i,j-1}^{n+1}\right) = u_{i,j}^{n} + \Delta t S_{i,j}^{n}. \quad (8)$$

This approximation allows to write an equation for each grid node (i, j). We will write these

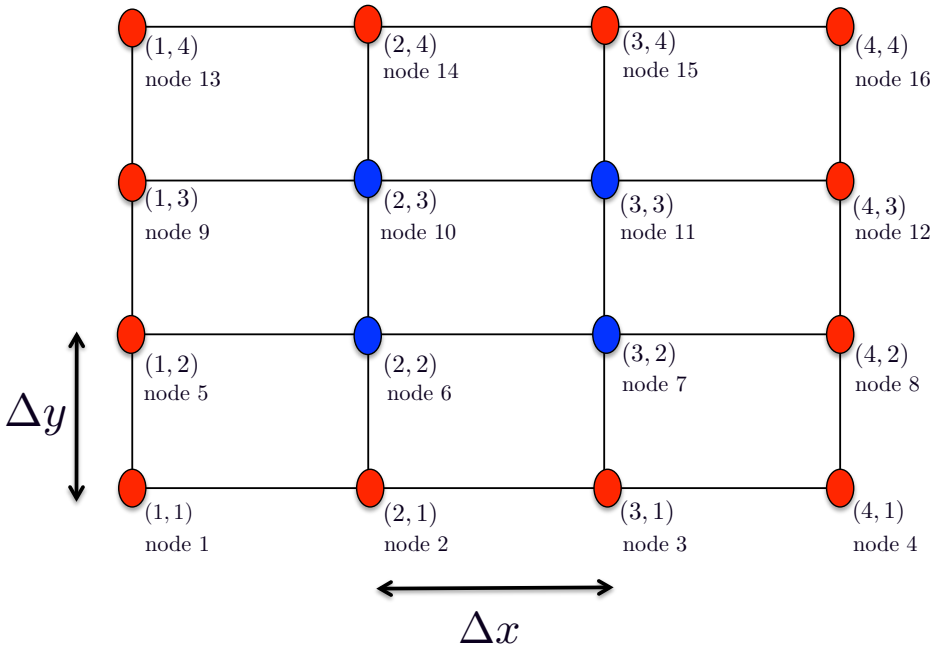

Figure 5: Discretization of a two dimensional domain with $m = 4$ points in the x direction and $n = 4$ points in the y direction. A grid node is referenced by its i and j indices. The i index references the i^{th} location in the x direction, while the j index references the j^{th} location in the y direction. The solution u at red nodes will be given the value of the boundary conditions given by the function BC, while at the blue nodes the solution will be approximated using (8).

equations, node after node, with the convention that the first equation is written for $(1, 1)$ (we will call this node 1), then for $(1, 2)$ (we will call this node 2), and so on from left to right and bottom to top (see figure 3). The approximation (8) is valid for all interior nodes, whereas for the nodes on the domain's walls the values of u are simply given by the boundary conditions. Writing these equations gives a linear system:

$$Au^{n+1} = rhs.$$

In the example of the grid in figure 3, if we define the coefficients:

$$C = 1 + D\left(\frac{2\Delta t}{\Delta x^2} + \frac{2\Delta t}{\Delta y^2}\right), \quad R = -D\frac{\Delta t}{\Delta x^2}, \quad L = -D\frac{\Delta t}{\Delta x^2}, \quad T = -D\frac{\Delta t}{\Delta y^2}, \quad B = -D\frac{\Delta t}{\Delta y^2},$$

then the linear system is written as:

$$\underbrace{\begin{pmatrix}
\mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{B} & 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 & \mathbf{T} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{B} & 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 & \mathbf{T} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{B} & 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 & \mathbf{T} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{B} & 0 & 0 & \mathbf{L} & \mathbf{C} & \mathbf{R} & 0 & 0 & \mathbf{T} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1}
\end{pmatrix}}_{A}
\underbrace{\begin{pmatrix}
u_{1,1}^{n+1} \\ u_{2,1}^{n+1} \\ u_{3,1}^{n+1} \\ u_{4,1}^{n+1} \\ u_{1,2}^{n+1} \\ u_{2,2}^{n+1} \\ u_{3,2}^{n+1} \\ u_{4,2}^{n+1} \\ u_{1,3}^{n+1} \\ u_{2,3}^{n+1} \\ u_{3,3}^{n+1} \\ u_{4,3}^{n+1} \\ u_{1,4}^{n+1} \\ u_{2,4}^{n+1} \\ u_{3,4}^{n+1} \\ u_{4,4}^{n+1}
\end{pmatrix}}_{u^{n+1}}
=
\underbrace{\begin{pmatrix}
BC_{1,1}^{n} \\ BC_{2,1}^{n} \\ BC_{3,1}^{n} \\ BC_{4,1}^{n} \\ BC_{1,2}^{n} \\ u_{2,2}^{n} + \Delta t S_{2,2}^{n} \\ u_{3,2}^{n} + \Delta t S_{3,2}^{n} \\ BC_{4,2}^{n} \\ BC_{1,3}^{n} \\ u_{2,3}^{n} + \Delta t S_{2,3}^{n} \\ u_{3,3}^{n} + \Delta t S_{3,3}^{n} \\ BC_{4,3}^{n} \\ BC_{1,4}^{n} \\ BC_{2,4}^{n} \\ BC_{3,4}^{n} \\ BC_{4,4}^{n}
\end{pmatrix}}_{rhs}$$

Remarks:

1. For each node, one row of the linear system is filled. More precisely, for node p, the p^{th} row of the linear system is filled.
2. Given (i, j), the node number (hence the row of the linear system filled) is $p = (j-1)m + i$.

3. In general, when filling the p^{th} row, the column associated with the coefficient T is $p+m$ while the column associated with the coefficient B is $p-m$.

A MATLAB code to solve the Poisson equation in two spatial dimensions is given next. In particular, note how we code the general linear system associated with the discretization of the Poisson equation in two spatial dimensions:

```
function outvar=Heat_2D(D,S,BC,Initial_Data,a,b,c,d,m,n,dt,t_final)

    aviobj         = avifile('My_Movie.avi');
    aviobj.Quality = 100;

    % Discretize the domain:
    x=linspace(a,b,m);
    y=linspace(c,d,n);
    dx=(b-a)/(m-1);
    dy=(d-c)/(n-1);
    mn=m*n;

    % Initialize the variables:
    A=sparse(mn,mn);
    t=0;
    % Initial condition:
    for i=1:m
        for j=1:n
            r=(j-1)*m+i;
            Un(r)=Initial_Data(x(i),y(j),t);
        end
    end

    % Time stepping:
    while t<t_final % & count<...
        if t+dt>t_final
            dt=t_final-t;
        end
        t=t+dt;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Assemble the linear system: %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
 % Treat the interior points:
 for i=2:m-1
     for j=2:n-1
         r=(j-1)*m+i;
         A(r,r  )=1 + 2*dt*D/dx/dx + 2*dt*D/dy/dy;
         A(r,r-1)=     -dt*D/dx/dx;
         A(r,r+1)=     -dt*D/dx/dx;
         A(r,r-m)=                   -dt*D/dy/dy;
         A(r,r+m)=                   -dt*D/dy/dy;
         rhs(r)=Un(r)+dt*S(x(i),y(j),t);
     end
 end

 % Treat the boundary conditions:
for j=1:n
     i=1;        % Left  boundary condition
     r=(j-1)*m+i;
     A(r,r)=1; rhs(r)=BC(a,y(j),t);
     i=m;        % Right boundary condition
     r=(j-1)*m+i;
     A(r,r)=1; rhs(r)=BC(b,y(j),t);
end

for i=1:m
     j=1;        % Bottom boundary condition
     r=(j-1)*m+i;
     A(r,r)=1; rhs(r)=BC(x(i),c,t);
     j=n;        % Top    boundary condition
     r=(j-1)*m+i;
     A(r,r)=1; rhs(r)=BC(x(i),d,t);
end

% Solve the linear system:
Unp1=A\rhs';
```

```
        Un=Unp1;

        % Need to put Un, which is now a vector, into a grid form.
        for i=1:m
            for j=1:n
                r=(j-1)*m+i;
                U_grid(i,j)=Un(r);
            end
        end
        mesh(x,y,U_grid); axis([a b, c d, 0 2]);
        xlabel('x');ylabel('y'); zlabel('Temperature');
        s=sprintf('Numerical Solution of the Temperature distribution at t =%4.2f',t);
        title(s);
%        title('Numerical Solution of the Temperature distribution');

        F=getframe(gca);
        aviobj=addframe(aviobj,F);
        pause(dt/10);
    end
    aviobj=close(aviobj);

    outvar=Un;
end
```

An example of how this function is called is:

```
>> BC=@(x,y,t) 0;
>> Initial_Data=@(x,y,t) 0;
>> result=Heat_2D(.1,@Source_Term,BC,-1,1,-1,1,50,50,.1,4);
```

The function will returns the solution of the heat equation at time $t = 4$ and store it in the vector `result` and will plot the figure 6. Here the function `Source_Term` is given by:

```
function outvar = Source_Term(x,y,t)
if t<1
    if sqrt((x-.5)*(x-.5)+(y-.75)*(y-.75)) <.1
        outvar = 5;
```

```
    else
        if sqrt((x+.5)*(x+.5)+(y+.75)*(y+.75)) <.2
            outvar=-2;
        else
            outvar = 0;
        end
    end
else
    if sqrt((x-.5)*(x-.5)+(y-.75)*(y-.75)) <.1
        outvar = -3;
    else
        if sqrt((x+.5)*(x+.5)+(y+.75)*(y+.75)) <.2
            outvar=1;
        else
            outvar = 0;
        end
    end
end
```

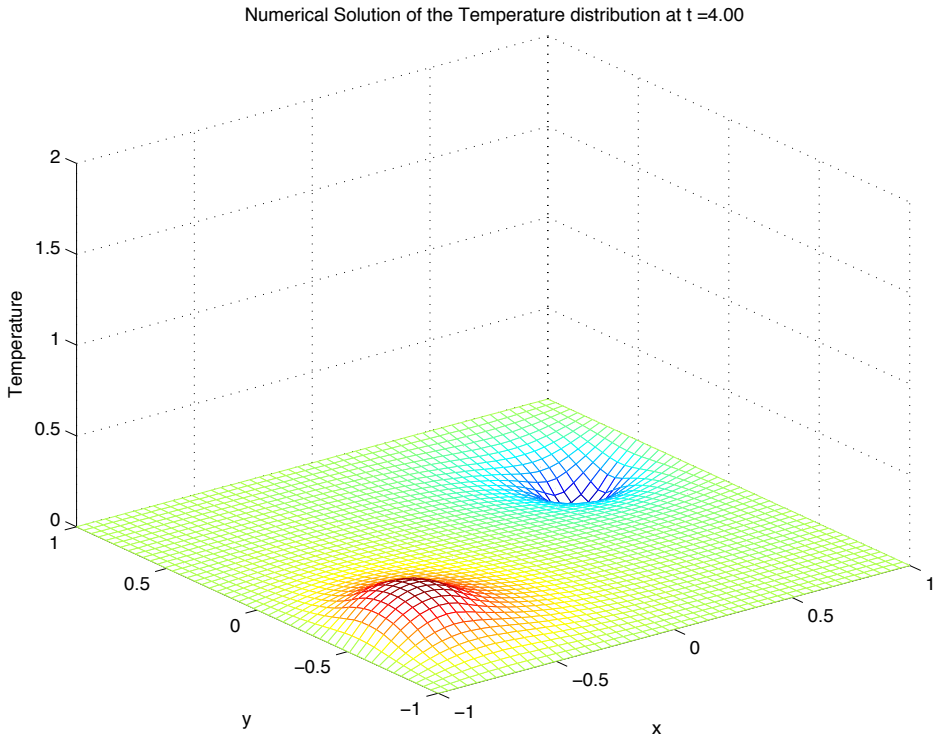

Figure 6: Graphical representation of the solution of the Heat equation in two spatial dimensions at $t = 4$.