MAP 2320 – MÉTODOS NUMÉRICOS EM EQUAÇÕES DIFERENCIAIS II

2º Semestre - 2020

Prof. Dr. Luis Carlos de Castro Santos

lsantos@ime.usp.br

(1) *Steady-state:*
(called the **Poisson equation**)

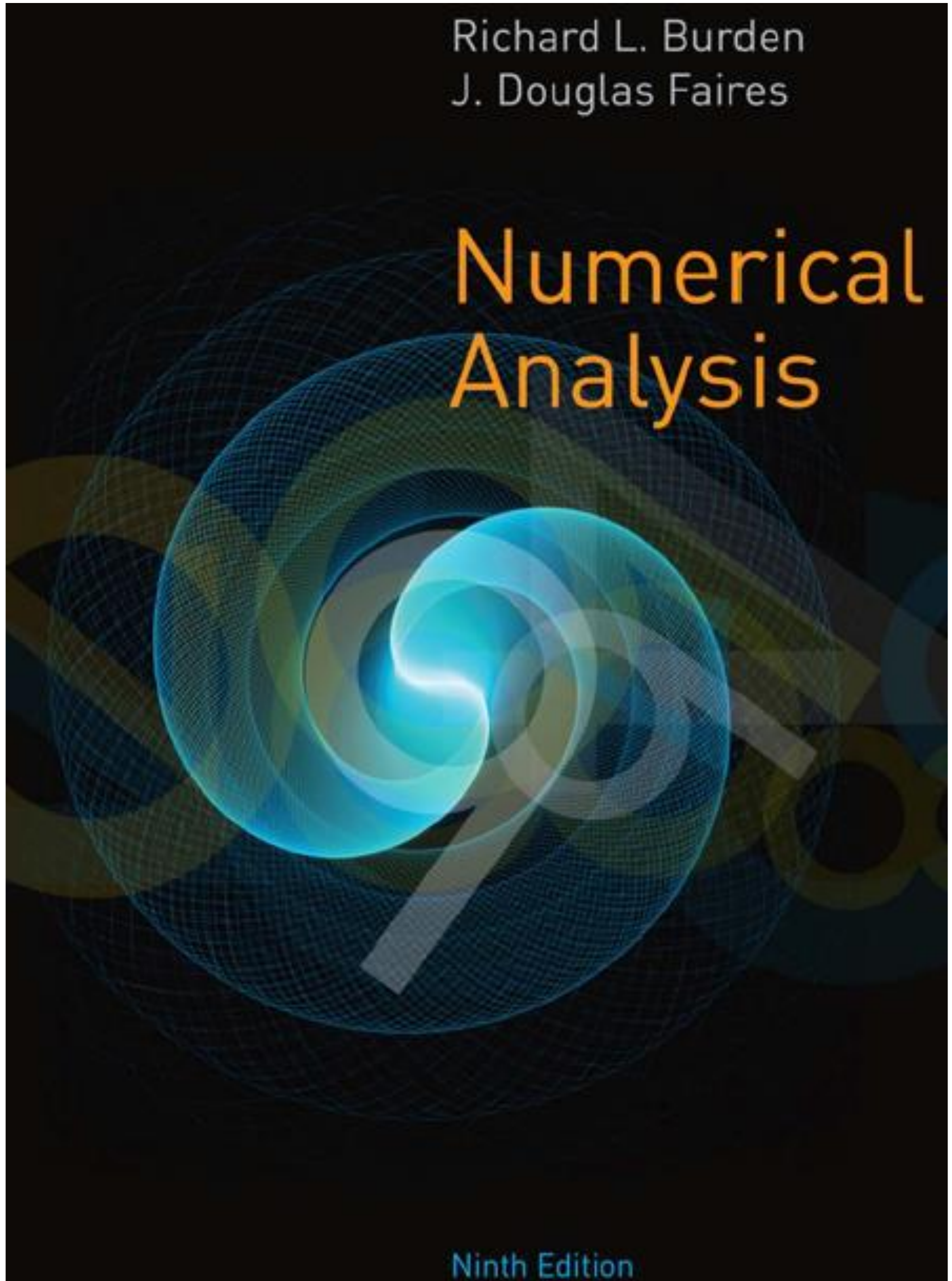
$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{\dot{e}_{\text{gen}}}{k} = 0$$

(2) *Transient, no heat generation:*
(called the **diffusion equation**)

(3) *Steady-state, no heat generation:*
(called the **Laplace equation**)

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = 0$$

MAP2320


Richard L. Burden
J. Douglas Faires
Numerical Analysis
Ninth Edition

Numerical Analysis

NINTH EDITION

Richard L. Burden
Youngstown State University

J. Douglas Faires
Youngstown State University

12 Numerical Solutions to Partial Differential Equations 713

12.1 Elliptic Partial Differential Equations

The *elliptic* partial differential equation we consider is the Poisson equation,

$$\nabla^2 u(x,y) \equiv \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y) \tag{12.1}$$

on $R = \{(x,y) \mid a < x < b, c < y < d\}$, with $u(x,y) = g(x,y)$ for $(x,y) \in S$, where S denotes the boundary of R. If f and g are continuous on their domains, then there is a unique solution to this equation.

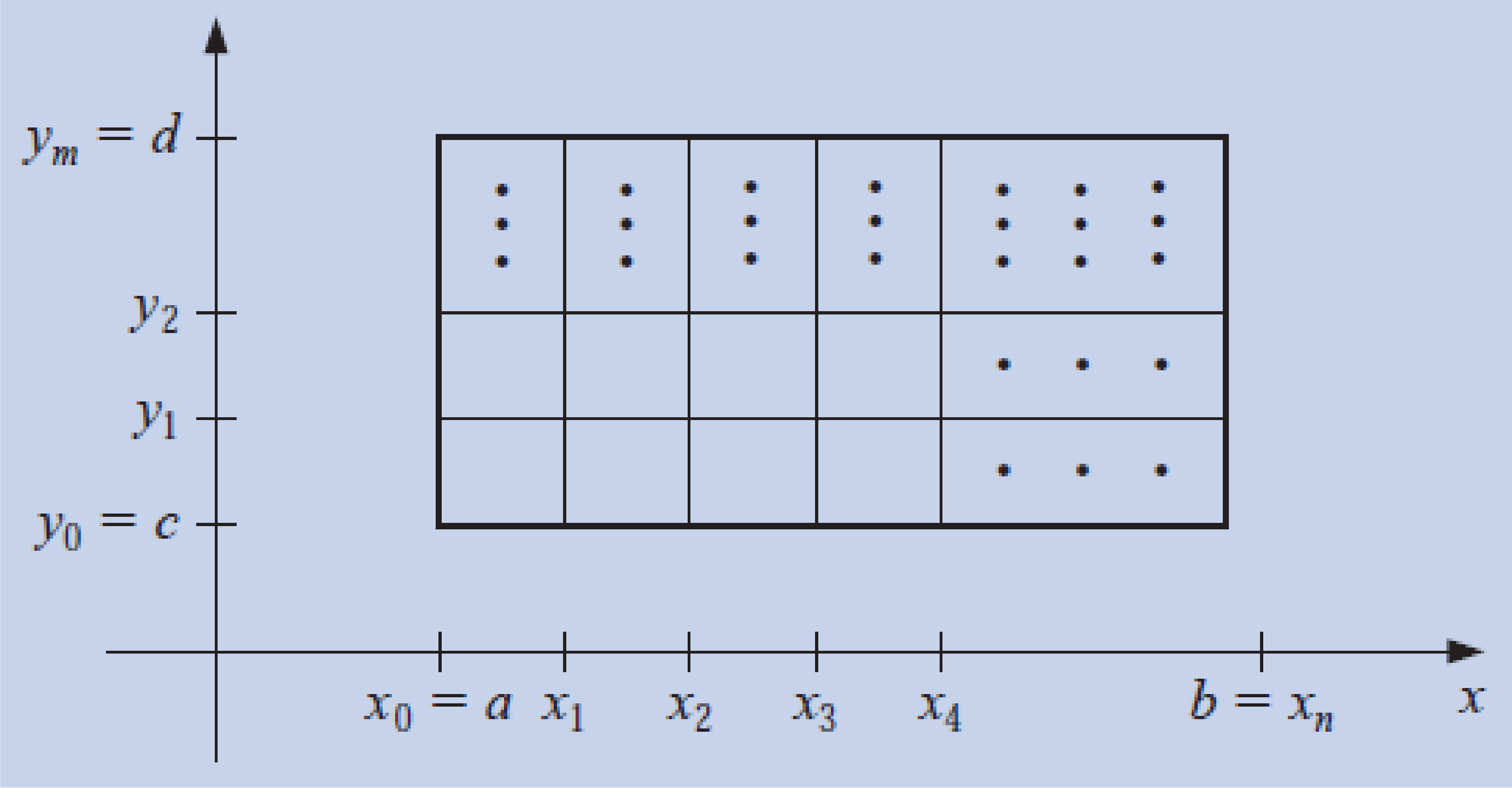
Por enquanto apenas C.C. de Dirichlet

Selecting a Grid

The method used is a two-dimensional adaptation of the Finite-Difference method for linear boundary-value problems, which was discussed in Section 11.3. The first step is to choose integers n and m to define step sizes $h = (b-a)/n$ and $k = (d-c)/m$. Partition the interval $[a,b]$ into n equal parts of width h and the interval $[c,d]$ into m equal parts of width k (see Figure 12.4).

$$h = \Delta x, \qquad k = \Delta y$$

Figure 12.4



Place a grid on the rectangle R by drawing vertical and horizontal lines through the points with coordinates (x_i, y_j), where

$$x_i = a + ih, \quad \text{for each } i = 0, 1, \ldots, n, \quad \text{and} \quad y_j = c + jk, \quad \text{for each } j = 0, 1, \ldots, m.$$

The lines $x = x_i$ and $y = y_j$ are **grid lines**, and their intersections are the **mesh points** of the grid. For each mesh point in the interior of the grid, (x_i, y_j), for $i = 1, 2, \ldots, n-1$ and $j = 1, 2, \ldots, m-1$, we can use the Taylor series in the variable x about x_i to generate the centered-difference formula

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h^2} - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j), \tag{12.2}$$

where $\xi_i \in (x_{i-1}, x_{i+1})$. We can also use the Taylor series in the variable y about y_j to generate the centered-difference formula

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{k^2} - \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, \eta_j), \tag{12.3}$$

where $\eta_j \in (y_{j-1}, y_{j+1})$.

Using these formulas in Eq. (12.1) allows us to express the Poisson equation at the points (x_i, y_j) as

$$\frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h^2} + \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{k^2}$$

$$= f(x_i, y_j) + \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) + \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, \eta_j),$$

for each $i = 1, 2, \ldots, n-1$ and $j = 1, 2, \ldots, m-1$. The boundary conditions are

$$u(x_0, y_j) = g(x_0, y_j) \quad \text{and} \quad u(x_n, y_j) = g(x_n, y_j), \quad \text{for each } j = 0, 1, \ldots, m;$$

$$u(x_i, y_0) = g(x_i, y_0) \quad \text{and} \quad u(x_i, y_m) = g(x_i, y_m), \quad \text{for each } i = 1, 2, \ldots, n-1.$$

Por enquanto apenas C.C. de Dirichlet

Finite-Difference Method

In difference-equation form, this results in the **Finite-Difference method**:

$$2\left[\left(\frac{h}{k}\right)^2 + 1\right] w_{ij} - (w_{i+1,j} + w_{i-1,j}) - \left(\frac{h}{k}\right)^2 (w_{i,j+1} + w_{i,j-1}) = -h^2 f(x_i, y_j), \quad (12.4)$$

for each $i = 1, 2, \ldots, n-1$ and $j = 1, 2, \ldots, m-1$, and

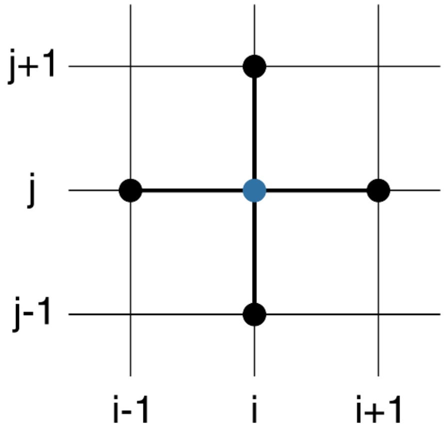
$$w_{0j} = g(x_0, y_j) \quad \text{and} \quad w_{nj} = g(x_n, y_j), \quad \text{for each } j = 0, 1, \ldots, m; \quad (12.5)$$

$$w_{i0} = g(x_i, y_0) \quad \text{and} \quad w_{im} = g(x_i, y_m), \quad \text{for each } i = 1, 2, \ldots, n-1;$$

where w_{ij} approximates $u(x_i, y_j)$. This method has local truncation error of order $O(h^2 + k^2)$

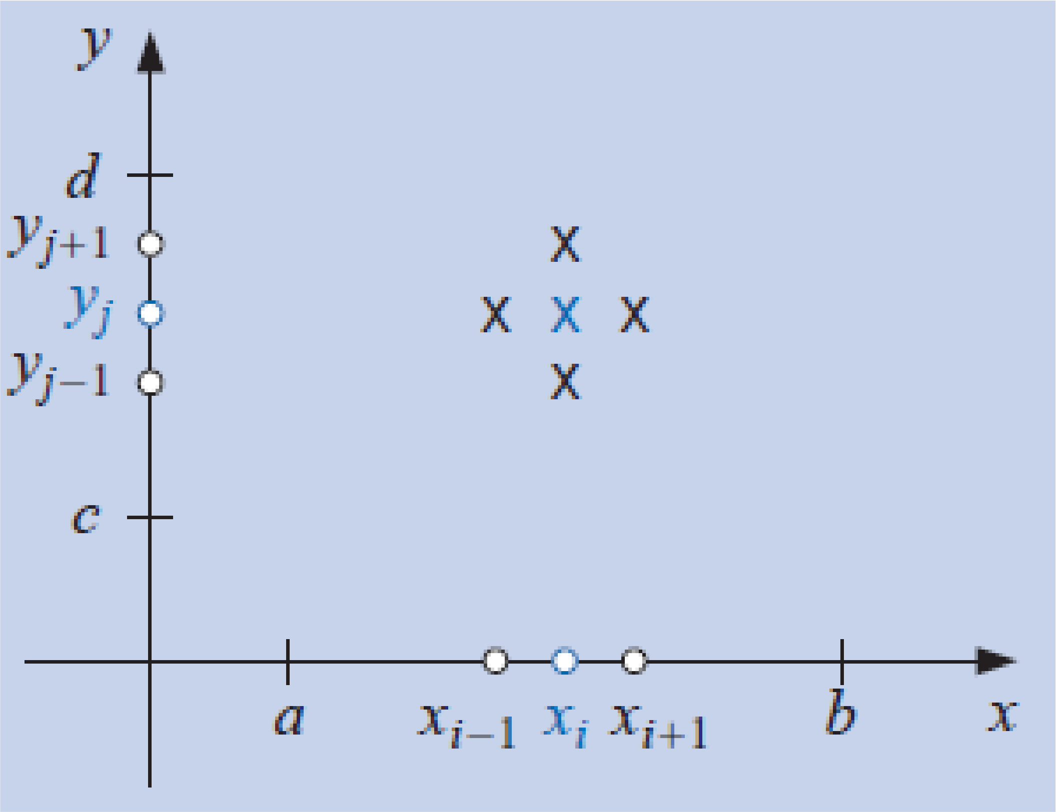
The typical equation in (12.4) involves approximations to $u(x, y)$ at the points

$$(x_{i-1}, y_j), \quad (x_i, y_j), \quad (x_{i+1}, y_j), \quad (x_i, y_{j-1}), \quad \text{and} \quad (x_i, y_{j+1}).$$



Reproducing the portion of the grid where these points are located (see Figure 12.5) shows that each equation involves approximations in a star-shaped region about the blue X at (x_i, y_j).

Figure 12.5



We use the information from the boundary conditions (12.5) whenever appropriate in the system given by (12.4); that is, at all points (x_i, y_j) adjacent to a boundary mesh point.

This produces an $(n-1)(m-1) \times (n-1)(m-1)$ linear system with the unknowns being the approximations $w_{i,j}$ to $u(x_i, y_j)$ at the interior mesh points.

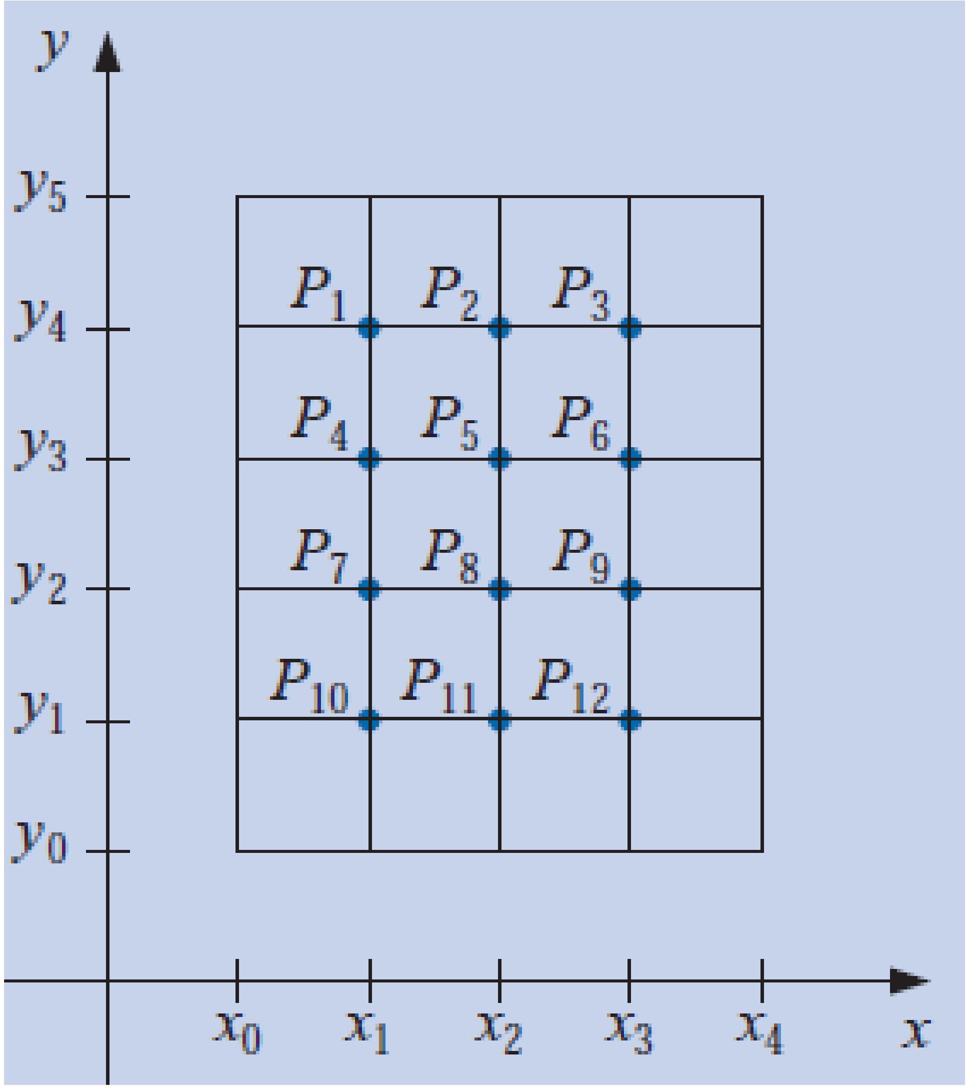
The linear system involving these unknowns is expressed for matrix calculations more efficiently if a relabeling of the interior mesh points is introduced. A recommended labeling of these points (see [Var1], p. 210) is to let

$$P_l = (x_i, y_j) \quad \text{and} \quad w_l = w_{i,j},$$

where $l = i + (m-1-j)(n-1)$, for each $i = 1, 2, \ldots, n-1$ and $j = 1, 2, \ldots, m-1$. This labels the mesh points consecutively from left to right and top to bottom. Labeling the points in this manner ensures that the system needed to determine the $w_{i,j}$ is a banded matrix with band width at most $2n-1$.

For example, with $n = 4$ and $m = 5$, the relabeling results in a grid whose points are shown in Figure 12.6.

Figure 12.6



Example 1

Determine the steady-state heat distribution in a thin square metal plate with dimensions 0.5 m by 0.5 m using $n = m = 4$. Two adjacent boundaries are held at 0°C, and the heat on the other boundaries increases linearly from 0°C at one corner to 100°C where the sides meet.

Solution Place the sides with the zero boundary conditions along the x- and y-axes. Then the problem is expressed as

$$\frac{\partial^2 u}{\partial x^2}(x, y) + \frac{\partial^2 u}{\partial y^2}(x, y) = 0,$$

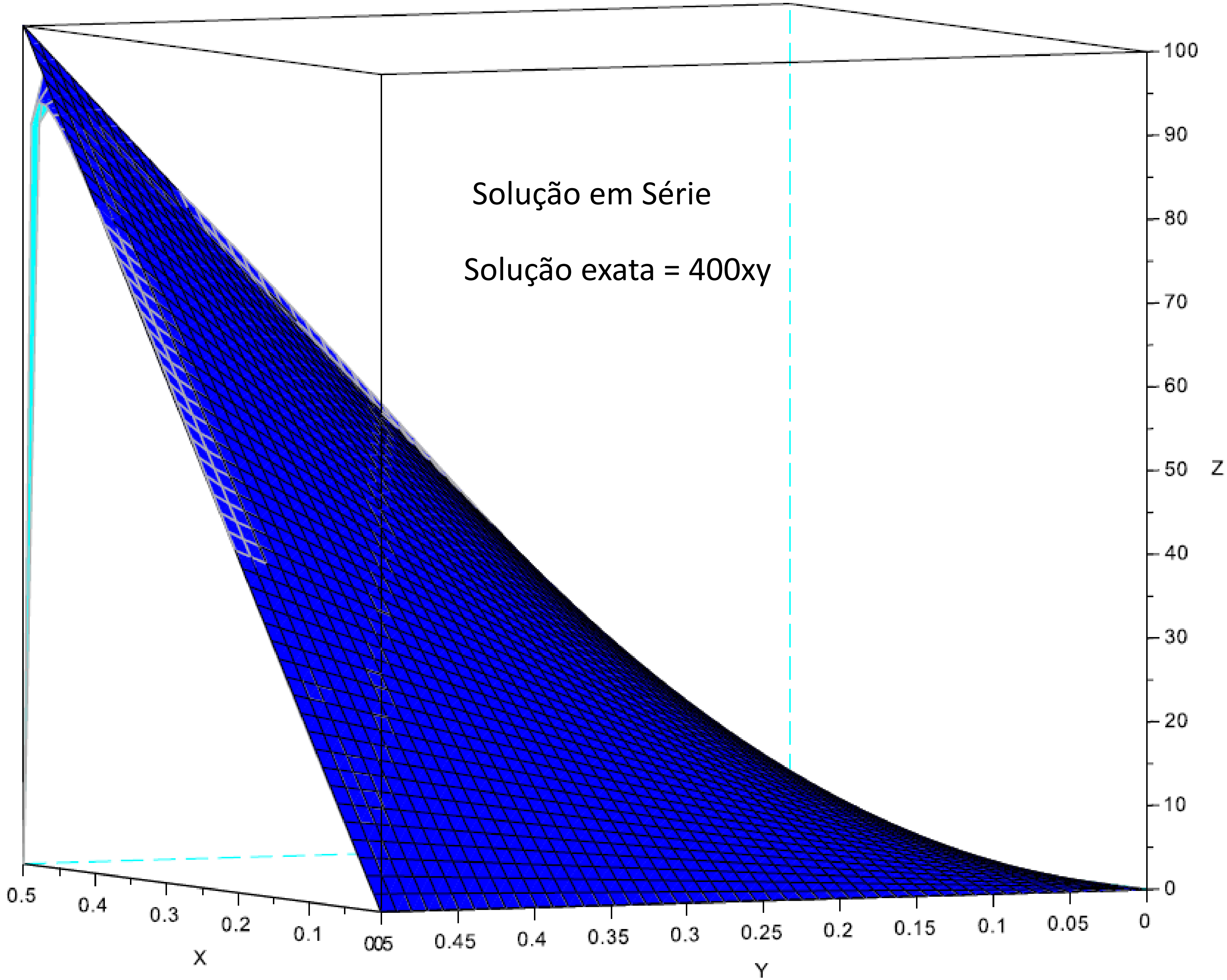
for (x, y) in the set $R = \{\, (x, y) \mid 0 < x < 0.5,\ 0 < y < 0.5 \,\}$. The boundary conditions are

$$u(0, y) = 0, \quad u(x, 0) = 0, \quad u(x, 0.5) = 200x, \text{ and } u(0.5, y) = 200y.$$

```
t=linspace(0,0.5,50);
T0=200;
function z=my_surface(x, y)
n=100;
z=0;
for k=1:n
cg = -T0/(k*%pi*sinh(k*%pi))*cos(k*%pi);
ck = cg;
ug =cg*sin(2*k*%pi*x)*sinh(2*k*%pi*y);
uk =ck*sin(2*k*%pi*y)*sinh(2*k*%pi*x);
z=z + ug + uk;
end
endfunction

function zz=fexata(x, y)
zz=2*T0*x*y
endfunction

z=feval(t,t,my_surface);
zz=feval(t,t,fexata);

plot3d(t,t,z);
```

Solução em Série

Solução exata = 400xy

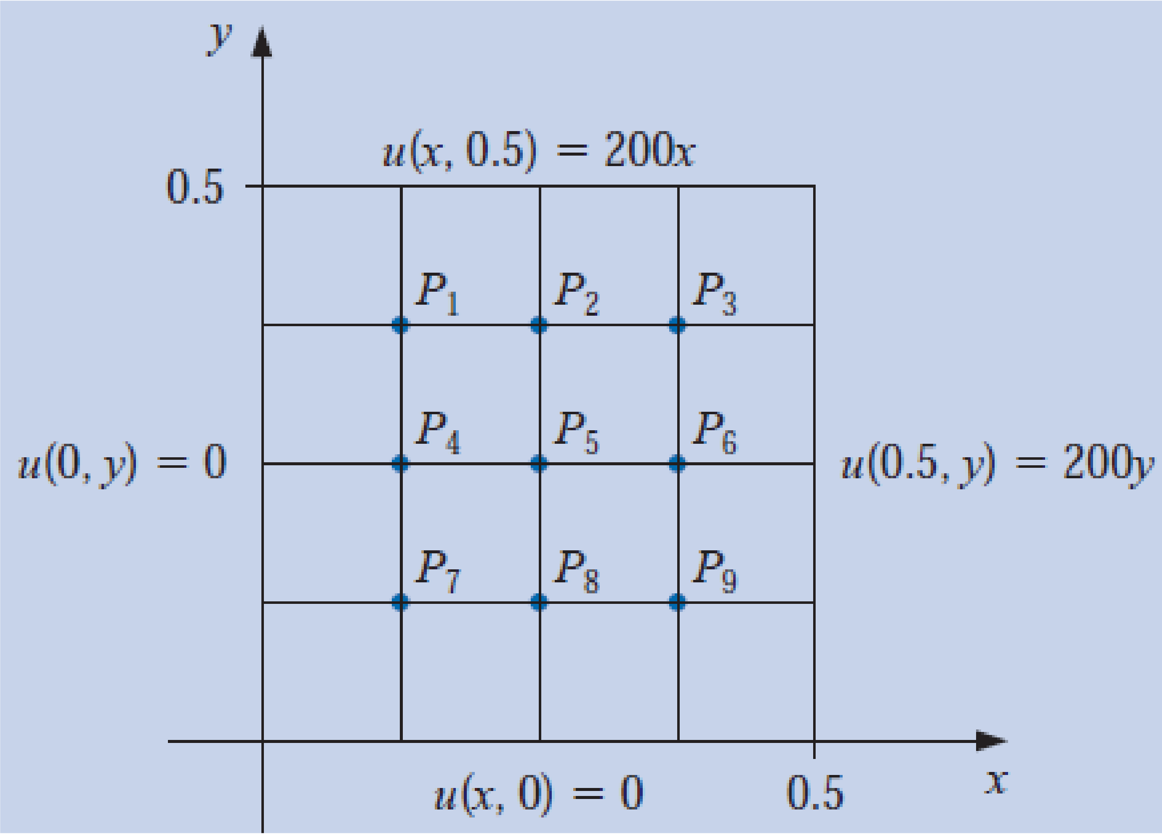

Solução em Série
Solução exata = 400xy
100
90
80
70
60
50
40
30
20
10
0
Z
0.5
0.4
0.3
0.2
0.1
X
005
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
Y

If $n = m = 4$, the problem has the grid given in Figure 12.7, and the difference equation (12.4) is

$$4w_{i,j} - w_{i+1,j} - w_{i-1,j} - w_{i,j-1} - w_{i,j+1} = 0,$$

for each $i = 1, 2, 3$ and $j = 1, 2, 3$.

Figure 12.7



Expressing this in terms of the relabeled interior grid points $w_i = u(P_i)$ implies that the equations at the points P_i are:

$$\begin{aligned}
P_1: &\quad 4w_1 - w_2 - w_4 = w_{0,3} + w_{1,4},\\
P_2: &\quad 4w_2 - w_3 - w_1 - w_5 = w_{2,4},\\
P_3: &\quad 4w_3 - w_2 - w_6 = w_{4,3} + w_{3,4},\\
P_4: &\quad 4w_4 - w_5 - w_1 - w_7 = w_{0,2},\\
P_5: &\quad 4w_5 - w_6 - w_4 - w_2 - w_8 = 0,\\
P_6: &\quad 4w_6 - w_5 - w_3 - w_9 = w_{4,2},\\
P_7: &\quad 4w_7 - w_8 - w_4 = w_{0,1} + w_{1,0},\\
P_8: &\quad 4w_8 - w_9 - w_7 - w_5 = w_{2,0},\\
P_9: &\quad 4w_9 - w_8 - w_6 = w_{3,0} + w_{4,1},
\end{aligned}$$

where the right sides of the equations are obtained from the boundary conditions.

In fact, the boundary conditions imply that

$$w_{1,0} = w_{2,0} = w_{3,0} = w_{0,1} = w_{0,2} = w_{0,3} = 0,$$

$$w_{1,4} = w_{4,1} = 25, \quad w_{2,4} = w_{4,2} = 50, \quad \text{and} \quad w_{3,4} = w_{4,3} = 75.$$

So the linear system associated with this problem has the form

$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \end{bmatrix} = \begin{bmatrix} 25 \\ 50 \\ 150 \\ 0 \\ 0 \\ 50 \\ 0 \\ 0 \\ 25 \end{bmatrix}.$$

The values of $w_1, w_2, \ldots, w_9$, found by applying the Gauss-Seidel method to this matrix, are given in Table 12.1.

Table 12.1

i	w_i
1	18.75
2	37.50
3	56.25
4	12.50
5	25.00
6	37.50
7	6.25
8	12.50
9	18.75

These answers are exact, because the true solution, $u(x, y) = 400xy$, has

$$\frac{\partial^4 u}{\partial x^4} = \frac{\partial^4 u}{\partial y^4} \equiv 0,$$

and the truncation error is zero at each step.

$$\frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h^2} + \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{k^2}$$

$$= f(x_i, y_j) + \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) + \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, \eta_j),$$

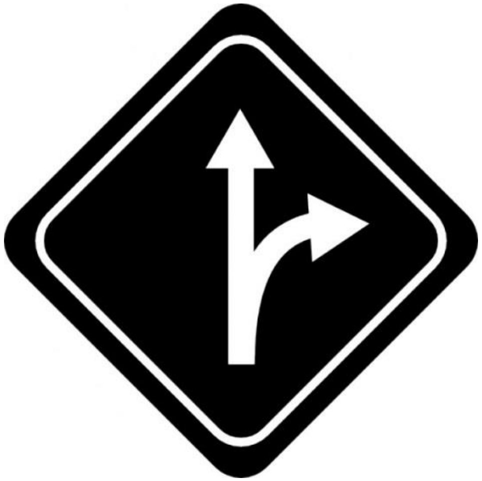
The problem we considered in Example 1 has the same mesh size, 0.125, on each axis and requires solving only a 9×9 linear system. This simplifies the situation and does not introduce the computational problems that are present when the system is larger.

9 x 9 = 81 coeficientes

Nesse caso 48 nulos, ou seja 59%

O que ocorre a medida que a matriz cresce ?


ATENÇÃO
DESVIO A
DIREITA



7.3 The Jacobi and Gauss-Siedel Iterative Techniques

In this section we describe the Jacobi and the Gauss-Seidel iterative methods, classic methods that date to the late eighteenth century. Iterative techniques are seldom used for solving linear systems of small dimension since the time required for sufficient accuracy exceeds that required for direct techniques such as Gaussian elimination. For large systems with a high percentage of 0 entries, however, these techniques are efficient in terms of both computer storage and computation. Systems of this type arise frequently in circuit analysis and in the numerical solution of boundary-value problems and partial-differential equations.

An iterative technique to solve the $n \times n$ linear system $A\mathbf{x} = \mathbf{b}$ starts with an initial approximation $\mathbf{x}^{(0)}$ to the solution $\mathbf{x}$ and generates a sequence of vectors $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ that converges to $\mathbf{x}$.

The Gauss-Seidel Method

A possible improvement in Algorithm 7.1 can be seen by reconsidering Eq. (7.5). The components of $\mathbf{x}^{(k-1)}$ are used to compute all the components $x_i^{(k)}$ of $\mathbf{x}^{(k)}$. But, for $i > 1$, the components $x_1^{(k)}, \ldots, x_{i-1}^{(k)}$ of $\mathbf{x}^{(k)}$ have already been computed and are expected to be better approximations to the actual solutions $x_1, \ldots, x_{i-1}$ than are $x_1^{(k-1)}, \ldots, x_{i-1}^{(k-1)}$. It seems reasonable, then, to compute $x_i^{(k)}$ using these most recently calculated values. That is, to use

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1}(a_{ij}x_j^{(k)}) - \sum_{j=i+1}^{n}(a_{ij}x_j^{(k-1)}) + b_i\right], \tag{7.8}$$

for each $i = 1, 2, \ldots, n$, instead of Eq. (7.5). This modification is called the **Gauss-Seidel iterative technique** and is illustrated in the following example.

Example 3

Use the Gauss-Seidel iterative technique to find approximate solutions to

$$\begin{aligned} 10x_1 - x_2 + 2x_3 \quad &= 6, \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25, \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11, \\ 3x_2 - x_3 + 8x_4 &= 15 \end{aligned}$$

starting with $\mathbf{x} = (0, 0, 0, 0)^t$ and iterating until

$$\frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_\infty}{\|\mathbf{x}^{(k)}\|_\infty} < 10^{-3}.$$

Solution The solution $\mathbf{x} = (1, 2, -1, 1)^t$ was approximated by Jacobi's method in Example 1. For the Gauss-Seidel method we write the system, for each $k = 1, 2, \ldots$ as

$$\begin{aligned} x_1^{(k)} &= \frac{1}{10}x_2^{(k-1)} - \frac{1}{5}x_3^{(k-1)} + \frac{3}{5}, \\ x_2^{(k)} &= \frac{1}{11}x_1^{(k)} + \frac{1}{11}x_3^{(k-1)} - \frac{3}{11}x_4^{(k-1)} + \frac{25}{11}, \\ x_3^{(k)} &= -\frac{1}{5}x_1^{(k)} + \frac{1}{10}x_2^{(k)} + \frac{1}{10}x_4^{(k-1)} - \frac{11}{10}, \\ x_4^{(k)} &= -\frac{3}{8}x_2^{(k)} + \frac{1}{8}x_3^{(k)} + \frac{15}{8}. \end{aligned}$$

When $\mathbf{x}^{(0)} = (0, 0, 0, 0)^t$, we have $\mathbf{x}^{(1)} = (0.6000, 2.3272, -0.9873, 0.8789)^t$. Subsequent iterations give the values in Table 7.2.

Table 7.2

k	0	1	2	3	4	5
$x_1^{(k)}$	0.0000	0.6000	1.030	1.0065	1.0009	1.0001
$x_2^{(k)}$	0.0000	2.3272	2.037	2.0036	2.0003	2.0000
$x_3^{(k)}$	0.0000	−0.9873	−1.014	−1.0025	−1.0003	−1.0000
$x_4^{(k)}$	0.0000	0.8789	0.9844	0.9983	0.9999	1.0000

Because

$$\frac{\|\mathbf{x}^{(5)} - \mathbf{x}^{(4)}\|_\infty}{\|\mathbf{x}^{(5)}\|_\infty} = \frac{0.0008}{2.000} = 4 \times 10^{-4},$$

$\mathbf{x}^{(5)}$ is accepted as a reasonable approximation to the solution. Note that Jacobi's method in Example 1 required twice as many iterations for the same accuracy. ■

ALGORITHM **7.2**

Gauss-Seidel Iterative

To solve $A\mathbf{x} = \mathbf{b}$ given an initial approximation $\mathbf{x}^{(0)}$:

INPUT the number of equations and unknowns n; the entries a_{ij}, $1 \leq i, j \leq n$ of the matrix A; the entries b_i, $1 \leq i \leq n$ of $\mathbf{b}$; the entries XO_i, $1 \leq i \leq n$ of $\mathbf{XO} = \mathbf{x}^{(0)}$; tolerance TOL; maximum number of iterations N.

OUTPUT the approximate solution $x_1, \ldots, x_n$ or a message that the number of iterations was exceeded.

Step 1 Set $k = 1$.

Step 2 While $(k \leq N)$ do Steps 3–6.

Step 3 For $i = 1, \ldots, n$

$$\text{set } x_i = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}XO_j + b_i\right].$$

Step 4 If $||\mathbf{x} - \mathbf{XO}|| < TOL$ then OUTPUT $(x_1, \ldots, x_n)$;
(*The procedure was successful.*)
STOP.

Step 5 Set $k = k + 1$.

Step 6 For $i = 1, \ldots, n$ set $XO_i = x_i$.

Step 7 OUTPUT ('Maximum number of iterations exceeded');
(*The procedure was successful.*)
STOP.

MAP2320

ALGORITHM
12.1

Poisson Equation Finite-Difference

To approximate the solution to the Poisson equation

$$\frac{\partial^2 u}{\partial x^2}(x, y) + \frac{\partial^2 u}{\partial y^2}(x, y) = f(x, y), \quad a \le x \le b, \quad c \le y \le d,$$

subject to the boundary conditions

$$u(x, y) = g(x, y) \quad \text{if } x = a \text{ or } x = b \quad \text{and} \quad c \le y \le d$$

and

$$u(x, y) = g(x, y) \quad \text{if } y = c \text{ or } y = d \quad \text{and} \quad a \le x \le b:$$

ALGORITHM 12.1

INPUT endpoints a, b, c, d; integers $m \geq 3$, $n \geq 3$; tolerance TOL; maximum number of iterations N.

OUTPUT approximations $w_{i,j}$ to $u(x_i, y_j)$ for each $i=1, \ldots, n-1$ and for each $j = 1, \ldots, m-1$ or a message that the maximum number of iterations was exceeded.

Step 1 Set $h = (b - a)/n$;
$k = (d - c)/m$.

Step 2 For $i = 1, \ldots, n-1$ set $x_i = a + ih$. (*Steps* 2 *and* 3 *construct mesh points.*)

Step 3 For $j = 1, \ldots, m-1$ set $y_j = c + jk$.

Step 4 For $i = 1, \ldots, n-1$
for $j = 1, \ldots, m-1$ set $w_{i,j} = 0$.

Step 5 Set $\lambda = h^2/k^2$;
$\mu = 2(1 + \lambda)$;
$l = 1$.

Step 6 While $l \leq N$ do Steps 7–20. (*Steps* 7–20 *perform Gauss-Seidel iterations.*)

Step 7 Set $z = \left(-h^2 f(x_1, y_{m-1}) + g(a, y_{m-1}) + \lambda g(x_1, d) + \lambda w_{1,m-2} + w_{2,m-1}\right)/\mu$;
$NORM = |z - w_{1,m-1}|$;
$w_{1,m-1} = z$.

ALGORITHM
12.1

Step 8 For $i = 2, \dots, n-2$

set $z = \big(-h^2 f(x_i, y_{m-1}) + \lambda g(x_i, d) + w_{i-1,m-1}$
$+ w_{i+1,m-1} + \lambda w_{i,m-2}\big)/\mu;$

if $|w_{i,m-1} - z| > NORM$ then set $NORM = |w_{i,m-1} - z|;$

set $w_{i,m-1} = z.$

Step 9 Set $z = \big(-h^2 f(x_{n-1}, y_{m-1}) + g(b, y_{m-1}) + \lambda g(x_{n-1}, d)$
$+ w_{n-2,m-1} + \lambda w_{n-1,m-2}\big)/\mu;$

if $|w_{n-1,m-1} - z| > NORM$ then set $NORM = |w_{n-1,m-1} - z|;$

set $w_{n-1,m-1} = z.$

Step 10 For $j = m-2, \dots, 2$ do Steps 11, 12, and 13.

Step 11 Set $z = \big(-h^2 f(x_1, y_j) + g(a, y_j) + \lambda w_{1,j+1} + \lambda w_{1,j-1} + w_{2,j}\big)/\mu;$

if $|w_{1,j} - z| > NORM$ then set $NORM = |w_{1,j} - z|;$

set $w_{1,j} = z.$

Step 12 For $i = 2, \dots, n-2$

set $z = \big(-h^2 f(x_i, y_j) + w_{i-1,j} + \lambda w_{i,j+1} + w_{i+1,j} + \lambda w_{i,j-1}\big)/\mu;$

if $|w_{i,j} - z| > NORM$ then set $NORM = |w_{i,j} - z|;$

set $w_{i,j} = z.$

Step 13 Set $z = \big(-h^2 f(x_{n-1}, y_j) + g(b, y_j) + w_{n-2,j}$
$+ \lambda w_{n-1,j+1} + \lambda w_{n-1,j-1}\big)/\mu;$

if $|w_{n-1,j} - z| > NORM$ then set $NORM = |w_{n-1,j} - z|;$

set $w_{n-1,j} = z.$

ALGORITHM 12.1

Step 14 Set $z = \left(-h^2 f(x_1, y_1) + g(a, y_1) + \lambda g(x_1, c) + \lambda w_{1,2} + w_{2,1}\right) / \mu$;
if $|w_{1,1} - z| > NORM$ then set $NORM = |w_{1,1} - z|$;
set $w_{1,1} = z$.

Step 15 For $i = 2, \ldots, n-2$
set $z = \left(-h^2 f(x_i, y_1) + \lambda g(x_i, c) + w_{i-1,1} + \lambda w_{i,2} + w_{i+1,1}\right) / \mu$;
if $|w_{i,1} - z| > NORM$ then set $NORM = |w_{i,1} - z|$;
set $w_{i,1} = z$.

Step 16 Set $z = \left(-h^2 f(x_{n-1}, y_1) + g(b, y_1) + \lambda g(x_{n-1}, c) + w_{n-2,1} + \lambda w_{n-1,2}\right) / \mu$;
if $|w_{n-1,1} - z| > NORM$ then set $NORM = |w_{n-1,1} - z|$;
set $w_{n-1,1} = z$.

Step 17 If $NORM \le TOL$ then do Steps 18 and 19.

Step 18 For $i = 1, \ldots, n-1$
for $j = 1, \ldots, m-1$ OUTPUT $(x_i, y_j, w_{i,j})$.

Step 19 STOP. *(The procedure was successful.)*

Step 20 Set $l = l + 1$.

Step 21 OUTPUT ('Maximum number of iterations exceeded');
(The procedure was unsuccessful.)
STOP.

Example 2

Use the Poisson finite-difference method with $n = 6$, $m = 5$, and a tolerance of 10^{-10} to approximate the solution to

$$\frac{\partial^2 u}{\partial x^2}(x, y) + \frac{\partial^2 u}{\partial y^2}(x, y) = xe^y, \quad 0 < x < 2, \quad 0 < y < 1,$$

with the boundary conditions

e^x

$$u(0, y) = 0, \quad u(2, y) = 2e^y, \quad 0 \leq y \leq 1,$$
$$u(x, 0) = x, \quad u(x, 1) = ex, \quad 0 \leq x \leq 2,$$

and compare the results with the exact solution $u(x, y) = xe^y$.

Solution Using Algorithm 12.1 with a maximum number of iterations set at $N = 100$ gives the results in Table 12.2. The stopping criterion for the Gauss-Seidel method in Step 17 requires that

$$\left| w_{ij}^{(l)} - w_{ij}^{(l-1)} \right| \leq 10^{-10},$$

for each $i = 1, \ldots, 5$ and $j = 1, \ldots, 4$. The solution to the difference equation was accurately obtained, and the procedure stopped at $l = 61$. The results, along with the correct values, are presented in Table 12.2. ■

Table 12.2

i	j	x_i	y_j	$w_{i,j}^{(61)}$	$u(x_i, y_j)$	$\left\|u(x_i, y_j) - w_{i,j}^{(61)}\right\|$
1	1	0.3333	0.2000	0.40726	0.40713	1.30×10^{-4}
1	2	0.3333	0.4000	0.49748	0.49727	2.08×10^{-4}
1	3	0.3333	0.6000	0.60760	0.60737	2.23×10^{-4}
1	4	0.3333	0.8000	0.74201	0.74185	1.60×10^{-4}
2	1	0.6667	0.2000	0.81452	0.81427	2.55×10^{-4}
2	2	0.6667	0.4000	0.99496	0.99455	4.08×10^{-4}
2	3	0.6667	0.6000	1.2152	1.2147	4.37×10^{-4}
2	4	0.6667	0.8000	1.4840	1.4837	3.15×10^{-4}
3	1	1.0000	0.2000	1.2218	1.2214	3.64×10^{-4}
3	2	1.0000	0.4000	1.4924	1.4918	5.80×10^{-4}
3	3	1.0000	0.6000	1.8227	1.8221	6.24×10^{-4}
3	4	1.0000	0.8000	2.2260	2.2255	4.51×10^{-4}
4	1	1.3333	0.2000	1.6290	1.6285	4.27×10^{-4}
4	2	1.3333	0.4000	1.9898	1.9891	6.79×10^{-4}
4	3	1.3333	0.6000	2.4302	2.4295	7.35×10^{-4}
4	4	1.3333	0.8000	2.9679	2.9674	5.40×10^{-4}
5	1	1.6667	0.2000	2.0360	2.0357	3.71×10^{-4}
5	2	1.6667	0.4000	2.4870	2.4864	5.84×10^{-4}
5	3	1.6667	0.6000	3.0375	3.0369	6.41×10^{-4}
5	4	1.6667	0.8000	3.7097	3.7092	4.89×10^{-4}

MAP 2320 – MÉTODOS NUMÉRICOS EM EQUAÇÕES DIFERENCIAIS II

2º Semestre - 2020

Roteiro do curso

- Introdução
- Séries de Fourier
- **Método de Diferenças Finitas**
- Equação do calor transiente (parabólica)
- **Equação de Poisson (elíptica)**
- Equação da onda (hiperbólica)