Tutorial Letter 202/1/2014

Numerical Methods II

APM3711

Semester 1

Department of Mathematical Sciences

IMPORTANT INFORMATION:

This tutorial letter contains solutions to assignment 02

BAR CODE

ONLY FOR SEMESTER 1 STUDENTS
ASSIGNMENT 02

FIXED CLOSING DATE: 16 APRIL 2014
Unique Number: 859053

Question 1

Consider the eigenvalue problem $Ax = \lambda x$ with

$$A = \begin{bmatrix} -6 & 0 & 6 \\ 4 & 9 & 2 \\ -3 & 0 & 5 \end{bmatrix} \quad \text{and} \quad A^{-1} = \begin{bmatrix} -\frac{5}{12} & 0 & \frac{1}{2} \\ \frac{13}{54} & \frac{1}{9} & -\frac{1}{3} \\ -\frac{1}{4} & 0 & \frac{1}{2} \end{bmatrix}.$$

(a) Apply Gerschgorin's theorem to find regions in the complex plane within which the eigenvalues must lie. Sketch these regions and give the number of eigenvalues within each.

Use the power method to estimate

(b) the dominant eigenvalue and the associated eigenvector,

(c) the eigenvalue with the smallest absolute value and the associated eigenvector,

(d) the remaining eigenvalue and the associated eigenvector.

In all cases, start with the vector $(1, 1, 1)$ and iterate three times. Use at least 4 decimal digits with rounding.

SOLUTION

Given

$$A = \begin{bmatrix} -6 & 0 & 6 \\ 4 & 9 & 2 \\ -3 & 0 & 5 \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} -\frac{5}{12} & 0 & \frac{1}{2} \\ \frac{13}{54} & \frac{1}{9} & -\frac{1}{3} \\ -\frac{1}{4} & 0 & \frac{1}{2} \end{bmatrix}.$$

(a) Gerschgorin's circle theorems can be formulated as follows:

Theorem I

Let A be an $n \times n$ matrix (with the $a_{ij}s$ real– or complex–valued), then all the eigenvalues of A lie in the union of the following n disks, D_i, in the complex plane:

$$D_i = \left\{ z \in \mathbf{C} : |z - a_{ii}| \leq \sum_{j=1,\, j \neq i}^{n} |a_{ij}| \right\}, \qquad i = 1, 2, \ldots, n.$$

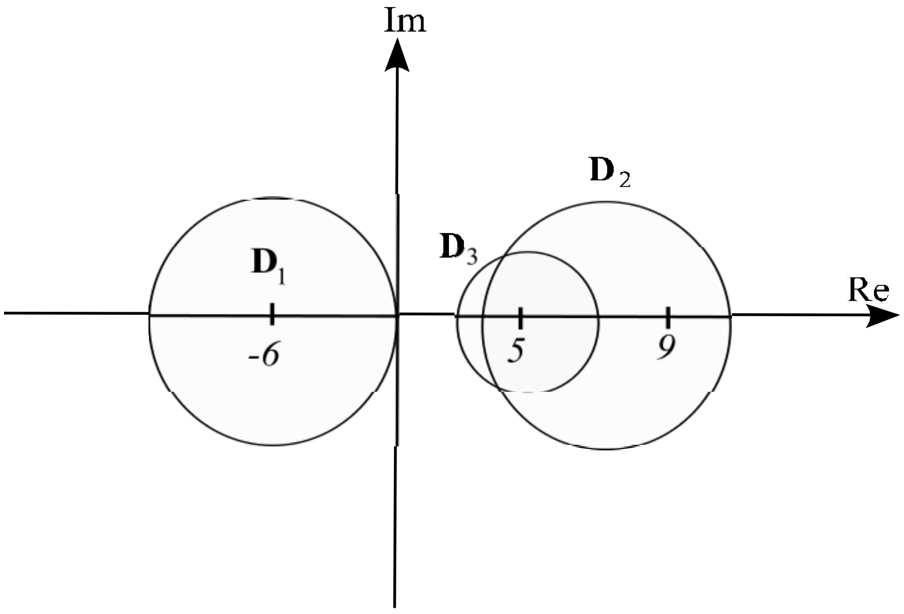
(D_i is simply the disk with centre a_{ii} and radius equal to the sum of the absolute values of the entries in row I which are not on the main diagonal.)

Theorem II

If k of these disks do not touch the other $n-k$ disks, then exactly k eigenvalues (counting multiplicities) lie in the union of those k disks.

For the matrix A above we have

$$\begin{aligned} D_1 &= \{z \in \mathbf{C} : |z-(-6)| \le |0| + |6| = 6\} \\ D_2 &= \{z \in \mathbf{C} : |z-9| \le |4| + |2| = 6\} \\ D_3 &= \{z \in \mathbf{C} : |z-5| \le |-3| + |0| = 3\}. \end{aligned}$$



According to Gerschgorin I and II one eigenvalues lies in D_1 and the other two lie in $D_2 \cup D_3$.

For the sake of interest we shall verify this by calculating the eigenvalues and the corresponding eigenvectors analytically. We have

$$\begin{aligned} Ax &= \lambda x \\ (A-\lambda I)\,x &= 0 \end{aligned}$$

where I denotes the 3×3 identity matrix. If $x \neq 0$ we must have $|A - \lambda I| = 0$. Now

$$\begin{aligned}
|A - \lambda I| &= \begin{vmatrix} -6-\lambda & 0 & 6 \\ 4 & 9-\lambda & 2 \\ -3 & 0 & 5-\lambda \end{vmatrix} \\
&= (9-\lambda)\begin{vmatrix} -6-\lambda & 6 \\ -3 & 5-\lambda \end{vmatrix} \\
&= (9-\lambda)\{(-6-\lambda)(5-\lambda) - 6(-3)\} \\
&= (9-\lambda)\left(\lambda^2 + \lambda - 12\right) \\
&= (9-\lambda)(\lambda+4)(\lambda-3).
\end{aligned}$$

Hence the **characteristic equation** is

$$(9-\lambda)(\lambda+4)(\lambda-3) = 0$$

with roots, i.e. the eigenvalues of A,

$$\lambda = -4,\ 3,\ 9.$$

This confirms our earlier conclusion, because -4 lies in D_1 while 3 and 9 lie in $D_2 \cup D_3$.

The eigenvectors:

$\underline{\lambda = -4}$

$$(A - \lambda I)\, x = 0$$

$$\Leftrightarrow \begin{bmatrix} -6-(-4) & 0 & 6 \\ 4 & 9-(-4) & 2 \\ -3 & 0 & 5-(-4) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{aligned}
\therefore \qquad -2x_1 + 6x_3 &= 0 && (1) \\
4x_1 + 13x_2 + 2x_3 &= 0 && (2) \\
-3x_1 + 9x_3 &= 0 && (3)
\end{aligned}$$

From (1) and (3) it follows that $x_1 = 3x_3$, so that (2) implies that

$$x_2 = -\frac{1}{13}(4x_1 + 2x_3) = -\frac{14}{13}x_3,$$

while an arbitrary value can be taken for x_3. Hence the eigenvectors for $\lambda = -4$ are given by

$$x = \left(3m,\ -\frac{14}{13}m,\ m\right) = m\left(3,\ -\frac{14}{13},\ 1\right), \quad m \text{ arbitrary.}$$

$\underline{\lambda = 3}$

$$(A-\lambda I)\,\underline{x}=\underline{0} \Leftrightarrow \begin{bmatrix} -6-3 & 0 & 6 \\ 4 & 9-3 & 2 \\ -3 & 0 & 5-3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{aligned} -9x_1 \qquad\quad + 6x_3 &= 0 \quad \Rightarrow x_3 = \frac{3}{2}x_1 \\ 4x_1 + 6x_2 + 3x_3 &= 0 \quad \Rightarrow x_2 = -\frac{1}{6}\left(4x_1 + 2x_3\right) = -\frac{7}{6}x_1 \\ -3x_1 \qquad\quad + 3x_2 &= 0 \quad \Rightarrow x_3 = \frac{3}{2}x_1 \end{aligned}$$

The eigenvector for $\lambda = 3$ is

$$x = \left(n,\ -\frac{7}{6}n,\ \frac{3}{2}n\right) = n\left(1,\ -\frac{7}{6},\ \frac{3}{2}\right), \qquad n \text{ arbitrary.}$$

$\underline{\lambda = 9}$

$$(A-\lambda I)\,\underline{x}=\underline{0} \Leftrightarrow \begin{bmatrix} -6-9 & 0 & 6-9 \\ 4 & 9-9 & 2 \\ -3 & 0 & 5-9 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{aligned} -15x_1 - 3x_3 &= 0 \quad \Rightarrow x_3 = -5x_1 \\ 4x_1 + 2x_3 &= 0 \quad \Rightarrow x_3 = 2x_1 \Rightarrow x_1 = x_3 = 0 \\ -3x_1 - 4x_3 &= 0 \end{aligned}$$

The eigenvector for $\lambda = 9$ is

$$\underline{x} = (0,p,0) = p\,(0,1,0)\,, \quad p \text{ arbitrary.}$$

(b) **The power method**

The dominant eigenvalue can be found by applying the power method to the matrix A.

Thus for $Ax = \lambda x$:

$$\begin{bmatrix} -6 & 0 & 6 \\ 4 & 9 & 2 \\ -3 & 0 & 5 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 15 \\ 2 \end{bmatrix} = 15\begin{bmatrix} 0 \\ 1 \\ 0.1333 \end{bmatrix}$$

$$\begin{bmatrix} -6 & 0 & 6 \\ 4 & 9 & 2 \\ -3 & 0 & 5 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 0.1333 \end{bmatrix} = \begin{bmatrix} 0.8000 \\ 9.2667 \\ 0.6667 \end{bmatrix} = 9.2667\begin{bmatrix} 0.0863 \\ 1 \\ 0.0719 \end{bmatrix}$$

After 2 iterations we obtain

$$\lambda = 9.2667, \qquad x = (0.0863,\ 1,\ 0.0719)\,.$$

After 10 iterations we obtain

$$\lambda = 8.9994, \qquad x = (0.0001,\ 1,\ 0.0000)\,.$$

Compare this with the result in (i) when $\lambda = 9$ and $p = 1$:

$$x = (0, 1, 0)\,.$$

(c) The smallest absolute eigenvalue can be obtained by computing the inverse of A and then using the power method because

$$Ax = \lambda x \Rightarrow x = A^{-1}Ax = A^{-1}\lambda x \Rightarrow A^{-1}x = \frac{1}{\lambda}x$$

and conversely

$$A^{-1}x = \mu x \Rightarrow x = AA^{-1}x = A\mu x \Rightarrow Ax = \frac{1}{\mu}x.$$

This proves that $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of A if and only if the eigenvalues of A^{-1} are $\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_n}$. Hence the eigenvalue of A with smallest absolute value is the inverse of the eigenvalue of A^{-1} with largest absolute value. So compute A^{-1} and apply the power method.

$$\begin{bmatrix} -\frac{5}{12} & 0 & \frac{1}{2} \\ \frac{13}{54} & \frac{1}{9} & -\frac{1}{3} \\ -\frac{1}{4} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.0833 \\ 0.0185 \\ 0.2500 \end{bmatrix} = 0.25 \begin{bmatrix} 0.3333 \\ 0.0741 \\ 1 \end{bmatrix}$$

$$\begin{aligned} \begin{bmatrix} -\frac{5}{12} & 0 & \frac{1}{2} \\ \frac{13}{54} & \frac{1}{9} & -\frac{1}{3} \\ -\frac{1}{4} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 0.3333 \\ 0.0741 \\ 1 \end{bmatrix} &= \begin{bmatrix} 0.3611 \\ -0.2449 \\ 0.4167 \end{bmatrix} \\ &= 0.4167 \begin{bmatrix} 0.8667 \\ -0.5877 \\ 1 \end{bmatrix} \end{aligned}$$

After 2 iterations we estimate the dominant eigenvalue of A^{-1} as

$$0.4167$$

and the corresponding eigenvector as

$$x = (0.8667,\ -0.5877,\ 1)\,.$$

The eigenvalue of least magnitude of A is therefore

$$\lambda = \frac{1}{0.4167} = 2.4000$$

and the corresponding eigenvector is the one given above.

After 10 iterations we obtain

$$\lambda = \frac{1}{0.3407} = 2.935, \qquad x = (0, 6884,\ -0.7805,\ 1)\,.$$

Compare this with the result in (i) when $\lambda = 3$ and $n = \frac{2}{3}$:

$$x = \left(\frac{2}{3},\ -\frac{7}{9},\ 1\right) = (0, 6667,\ -0.7778,\ 1)\,.$$

The convergence of the power method is slower than in (a) because the magnitude of the dominanet eigenvalue of A^{-1}, $\frac{1}{3}$, is not much larger than that of the eigenvalue with the second largest magnitude, $-\frac{1}{4}$.

(d) The two calculated eigenvalues, $\lambda_1 \approx 9.2667$ and $\lambda_2 \approx 2.4000$, both lie in the union of the disks D_2 and D_3. Hence we know from (i) that the remaining eigenvalue λ_3 must lie in D_1, i.e. λ_3 must be near -6. In order to use the inverse power method we must first shift the eigenvalues.

If λ is an eigenvalue of A with corresponding eigenvector x, then

$$\begin{aligned} Ax &= \lambda x \\ \therefore\quad Ax - (-6)\,Ix &= \lambda x - (-6)\,x \\ \therefore\quad (A + 6I)\,x &= (\lambda + 6)\,x \end{aligned}$$

Hence the eigenvalues of $A + 6I$ are

$$\lambda_1 + 6,\ \lambda_2 + 6,\ \lambda_3 + 6$$

and the eigenvalues of $(A + 6I)^{-1}$ are

$$\frac{1}{\lambda_1 + 6},\ \frac{1}{\lambda_2 + 6},\ \frac{1}{\lambda_3 + 6}.$$

Note that the eigenvectors reamin unchanged. Since λ_3 is near -6, $\lambda_3 + 6$ will be small. Thus $\frac{1}{\lambda_3+6}$ will be the dominant eigenvalue of $(A + 6I)^{-1}$ (compared to $\frac{1}{\lambda_1+6} \approx 0.0655$ and

$\frac{1}{\lambda_2+6} \approx 0.1190)$, and so the power method can be used to find it. First we use the Gauss–Jordan method to obtain $(A+6I)^{-1}$:

$$\left[\begin{array}{rrr|rrr} 0 & 0 & 6 & 1 & 0 & 0 \\ 4 & 15 & 2 & 0 & 1 & 0 \\ -3 & 0 & 11 & 0 & 0 & 1 \end{array}\right]$$

$$\sim \left[\begin{array}{rrr|rrr} 1 & 0 & -\frac{11}{3} & 0 & 0 & -\frac{1}{3} \\ \frac{4}{15} & 1 & \frac{2}{15} & 0 & \frac{1}{15} & 0 \\ 0 & 0 & 1 & \frac{1}{6} & 0 & 0 \end{array}\right]$$

$$\sim \left[\begin{array}{rrr|rrr} 1 & 0 & -\frac{11}{3} & 0 & 0 & -\frac{1}{3} \\ 0 & 1 & \frac{10}{9} & 0 & \frac{1}{15} & \frac{4}{45} \\ 0 & 0 & 1 & \frac{1}{6} & 0 & 0 \end{array}\right]$$

$$\sim \left[\begin{array}{rrr|rrr} 1 & 0 & 0 & \frac{11}{18} & 0 & -\frac{1}{3} \\ 0 & 1 & 0 & -\frac{10}{54} & \frac{1}{15} & \frac{4}{45} \\ 0 & 0 & 1 & \frac{1}{6} & 0 & 0 \end{array}\right]$$

$$\therefore \ (A+6I)^{-1} = \left[\begin{array}{rrr} \frac{11}{18} & 0 & -\frac{1}{3} \\ -\frac{10}{54} & \frac{1}{15} & \frac{4}{45} \\ \frac{1}{6} & 0 & 0 \end{array}\right]$$

Now the power method is applied

$$\begin{aligned} \left[\begin{array}{rrr} \frac{11}{18} & 0 & -\frac{1}{3} \\ -\frac{10}{54} & \frac{1}{15} & \frac{4}{45} \\ \frac{1}{6} & 0 & 0 \end{array}\right] \left[\begin{array}{r} 1 \\ 1 \\ 1 \end{array}\right] &= \left[\begin{array}{r} 0.2777 \\ -0.0296 \\ 0.1667 \end{array}\right] \\ &= 0.2777 \left[\begin{array}{r} 1 \\ -0.1067 \\ 0.6000 \end{array}\right] \end{aligned}$$

$$\begin{aligned} \left[\begin{array}{rrr} \frac{11}{18} & 0 & -\frac{1}{3} \\ -\frac{10}{54} & \frac{1}{15} & \frac{4}{45} \\ \frac{1}{6} & 0 & 0 \end{array}\right] \left[\begin{array}{r} 1 \\ -0.1067 \\ 0.6000 \end{array}\right] &= \left[\begin{array}{r} 0.4111 \\ -0.1390 \\ 0.1667 \end{array}\right] \\ &= 0.4111 \left[\begin{array}{r} 1 \\ -0.3380 \\ 0.4054 \end{array}\right] \end{aligned}$$

After 2 iterations we obtain

$$\begin{aligned} \frac{1}{\lambda_3+6} &= 0.4111 \qquad \therefore \ \lambda_3 = \frac{1}{0.4111} - 6 = -3.5676 \\ x &= (1, \ -0.3380, \ 0.4054) \,. \end{aligned}$$

After 10 iterations we obtain

$$\lambda_3 = \frac{1}{0.5000} - 6 = 4.000, \quad \underline{x} = (1, \ 0.3590, \ 0.3333) \,.$$

Compare this with the result in (a) when $\lambda = -4$ and $m = \frac{1}{3}$:

$$x = \left(1,\ \frac{14}{39},\ \frac{1}{3}\right) = (1,\ 0.3590,\ 0.3333)\,.$$

Question 2

Given the characteristic value problem

$$y'' - 3y' + 2k^2 y = 0,\ \ y(0) = 0,\ y(1) = 0, \tag{1}$$

using the method of finite differences, derive an eigenvalue problem for determining the non–zero values of k for which the differential equation has non–trivial solutions.

Solution

The problem (1) has the trivial solution $y \equiv 0$. For certain values of k, called characteristic values of the problem, it may also have non–trivial (non–zero) solutions. "Solving" the characteristic value problem consists of finding these characteristic values, and usually finding the corresponding solutions as well.

First, we set up the system of equations corresponding to the boundary–value problem, as usual, by replacing the differential equation in (1) with a finite–difference equation at the grid points. Let x_i denote the grid points, $h = x_{i+1} - x_i$ and $y_i = y(x_i)$. Replacing derivatives of y with respect to x by central differences, we see that the differential equation in (1) can be approximated at a grid point x_i by the difference equation

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} - 3\frac{y_{i+1} - y_{i-1}}{2h} + 2k^2 y_i = 0$$

which we rewrite as

$$y_{i-1}(1 + \frac{3}{2}h) - 2y_i(1 - k^2h^2) + y_{i+1}(1 - \frac{3}{2}h) = 0. \tag{2}$$

For each value of h we get a set of grid points x_i. We will need to apply (2) at each **interior** grid point x_i, $0 < x_i < 1$. The given boundary values $y(0) = y(1) = 0$ will give the values y_j at the two grid points x_j which coincide with the boundaries $x = 0$ and $x = 1$. This will give us a set of linear equations from which the y_i–values can be solved, providing us with the approximate values of the solution function $y(x)$ at the grid points x_i. The identification of characteristic values k, for which non–zero solutions y exist, can now be expressed as a problem of finding eigenvalues of a matrix.

(a) With $h = \frac{1}{2}$, the grid points are $x_0 = 0$, $x_1 = \frac{1}{2}$ and $x_2 = 1$. The boundary values give $y_0 = y(x_0) = 0$, $y_2 = y(x_2) = 0$. Applying (2) at the grid point $x_1 = \frac{1}{2}$ gives

$$\begin{aligned} &y_0(1 + \frac{3}{2}h) - 2y_1(1 - k^2h^2) + y_2(1 - \frac{3}{2}h) = 0 \\ \therefore\ \ &0 \cdot (1 + \frac{3}{2} \cdot \frac{1}{2}) = 2y_1(1 - k^2(\frac{1}{2})^2) + 0 \cdot (1 - \frac{3}{2} \cdot \frac{1}{2}) = 0 \\ \therefore\ &-2y_1(1 - \frac{k^2}{4}) = 0. \end{aligned} \tag{3}$$

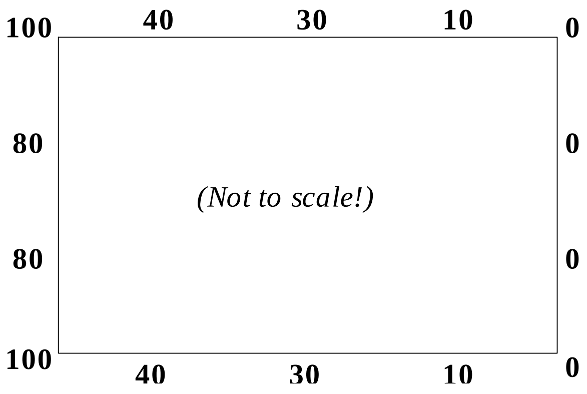
This is a very simple case, with one unknown variable and one equation. We see immediately the following: The equation (3) can have a non–zero solution y_1 if and only if

$$1 - \frac{k^2}{4} = 0$$

$$\therefore \quad k = \pm\, 2.$$

Question 3

We have a plate of 12×9 cm and the temperatures on the edges are held as shown in the sketch. Take $\Delta x = \Delta y = 3$ cm and use the **S.O.R. method** (successive overrelaxation method) to find the temperatures at all the grid–points. First calculate the optimal value of ω and then use this value in the algorithm.



SOLUTION

According to Section 7.7 [Section 7.2] of Gerald, the steady–state heat flow is modelled by Laplace's equation:

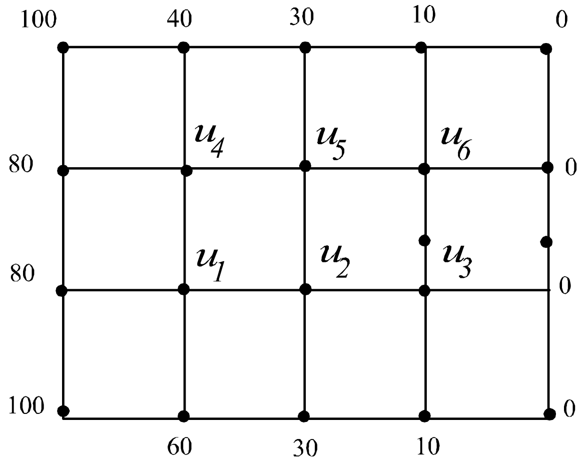
$$\nabla^2 u = 0 \tag{1}$$

The S.O.R. method for the solution of (1) is discussed in Gerald on pp. 556-557 [pp. 559–564]. The iteration formula for S.O.R. can be written as

$$\begin{aligned} u_{ij}^{(k+1)} &= u_{ij}^{(k)} + \frac{\omega}{4}[u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} - 4u_{ij}^{(k)}] \\ &= (1-\omega)u_{ij}^{(k)} + \frac{\omega}{4}[u_{i,j-1}^{(k+1)} + u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j+1}^{(k)}] \end{aligned} \tag{2}$$

where $u_{ij}^{(k)}$ denotes the $k-th$ approximation of $u(x_i, y_j)$ and ω is the overrelaxation factor.

Since $u_{i,j-1}^{(k+1)}$ and $u_{i-1,j}^{(k+1)}$ must be available for the calculation of $u_{ij}^{(k+1)}$, formula (2) must always be applied at points (x_i, y_{j-1}) and (x_{i-1}, y_j) before it is applied at (x_i, y_j). For the conventional numbering of the coordinates, where i increases from left to right and j increases in the upward direction, there are two possible orderings of the mesh points that will ensure this: (a) column–wise, starting at the left most column and moving upwards in every column, or (b) row–wise, starting at the bottom row and moving to the right in every row – as in the following sketch:



Applying (2) in this order, we obtain the following iteration scheme:

$$\begin{aligned}
u_1^{(k+1)} &= (1-\omega)u_1^{(k)} + \frac{\omega}{4}[60 + 80 + u_2^{(k)} + u_4^{(k)}] \\
u_2^{(k+1)} &= (1-\omega)u_2^{(k)} + \frac{\omega}{4}[30 + u_1^{(k+1)} + u_3^{(k)} + u_5^{(k)}] \\
u_3^{(k+1)} &= (1-\omega)u_3^{(k)} + \frac{\omega}{4}[10 + u_2^{(k+1)} + 0 + u_6^{(k)}] \\
u_4^{(k+1)} &= (1-\omega)u_4^{(k)} + \frac{\omega}{4}[u_1^{(k+1)} + 80 + u_5^{(k)} + 40] \\
u_5^{(k+1)} &= (1-\omega)u_5^{(k)} + \frac{\omega}{4}[u_2^{(k+1)} + u_4^{(k+1)} + u_6^{(k)} + 30] \\
u_6^{(k+1)} &= (1-\omega)u_6^{(k)} + \frac{\omega}{4}[u_3^{(k+1)} + u_5^{(k+1)} + 0 + 10]
\end{aligned}$$

Observe that the boundary values prescribed at the corners of the plate do not appear in any of the equations; in other words the S.O.R. method does not permit us to prescribe these values. This is due to the fact that (2) is based on the five–point formula for the Laplace operator.

In the program below the solution is represented as a matrix $u[i,j]$, so that (2) can be applied directly. Hence it is not really necessary to write down and copy to the program the equations above. The average of the boundary values at all the points, excluding the corners, is used as the initial estimate of u.

An estimate of the optimum overrelaxation factor ω is given on p. 557 [p. 561] of Gerald: We get that

$$\omega_{opt} = \frac{4}{2+\sqrt{4-c^2}}, \qquad c = \cos(\frac{\pi}{p}) + \cos(\frac{\pi}{q})$$

where p and q are the number of mesh divisions on each side of the rectangular domain. Thus $p = 3$ and $q = 4$, so that $\omega_{opt} \approx 1.1128$. For the sake of interest we verified this by executing the following program with different values of ω :

ω	iterations / iterasies
0.9	20
1.0	16
1.1	11
1.1128	10
1.2	11
1.3	14
1.4	18

The program and results follow.

```
PROGRAM AS02_5_3(output);
  CONST
    nx = 3; (* number of nodes in x-direction *)
    ny = 2; (*  "     "    "   "  y-    "     *)
    tolerance = 1.0E-5;
    itermax = 100;
  TYPE
    idim = 0..(nx + 1);
    jdim = 0..(ny + 1);
    solution = array[idim,jdim] of real;
  VAR
    i :  idim;  j :  jdim;
    k :  0..itermax;
    omega :  real;
    old_u, u :  solution;
    f :  text;

  FUNCTION MaxDif(u,v :  solution) :  real;
    VAR
      d :  real;
      i :  idim;  j :  jdim;
    BEGIN
      d := 0.0;
      FOR i := 1 to nx DO
        FOR j := 1 to ny DO
          IF abs(u[i,j] - v[i,j]) > d THEN
            d := abs(u[i,j] - v[i,j]);
      MaxDif := d;
    END; {MaxDif}

  PROCEDURE Initialize(VAR omega :  real;
                       VAR old_u, u :  solution);
    VAR
      c, sum :  real;
      i :  idim;  j :  jdim;
    BEGIN
      c := cos(pi/(nx + 1)) + cos(pi/(ny + 1));
      omega := 4/(2 + sqrt(4 - c*c));
      (* omega := 1.0 *)

      FOR i := 0 to nx + 1 DO
        FOR j := 0 to ny + 1 DO
          u[i,j] := 0.0;
      (* Nonzero boundary values:  *)
      u[0,1] :=  80.0;  u[0,2] :=  80.0;
      u[1,3] :=  40.0;  u[2,3] :=  30.0;
      u[3,3] :=  10.0;  u[1,0] :=  60.0;
      u[2,0] :=  30.0;  u[3,0] :=  10.0;

      (* Find the sum of the boundary values:  *)
      sum := 0.0;
      FOR i := 1 to nx DO
        sum := sum + u[i,0] + u[i,ny + 1];
      FOR j := 1 to ny DO
        sum := sum + u[0,j] + u[nx + 1,j];
      (* Define u at the interior points:  *)
      FOR i := 1 to nx DO
        FOR j := 1 to ny DO
          u[i,j] := sum/(2*nx + 2*ny);

      FOR i := 0 to nx + 1 DO
        FOR j := 0 to ny + 1 DO
          old_u[i,j] := u[i,j];
    END; {Initialize}

  BEGIN {Program}
    assign(f, 'work\AS02_5_3.DAT');
    rewrite(f);
    Initialize(omega,old_u,u);
    writeln(f);
    writeln(f,'    ********  ASSIGNMENT 2,  ',
              'QUESTION 3  ********');
    writeln(f);
    writeln(f,'    S.O.R. Method for the Laplace',
              ' equation:');
    writeln(f,'    omega = ',omega:6:4);
    writeln(f,'    tolerance = ',tolerance:8:6);
    writeln(f,'    max.  iterations = ',itermax:3);
    k := 0;
    REPEAT
      k := k + 1;
      FOR i := 1 to nx DO
        FOR j := 1 to ny DO
          old_u[i,j] := u[i,j];
      FOR i := 1 to nx DO
        FOR j := 1 to ny DO
          u[i,j] := (1 - omega)*old_u[i,j]
            + omega*(u[i,j - 1] + u[i - 1,j]
            + old_u[i + 1,j] + old_u[i,j + 1])/4;
    UNTIL (MaxDif(u,old_u) < tolerance) OR
          (k >= itermax);

    writeln(f);
    writeln(f,'    Solution and boundary values:');
    writeln(f);
    write(f,'            ');
    FOR i := 1 to nx DO
      write(f,' ',u[i,ny + 1]:9:4);
    writeln(f);
    FOR j := ny downto 1 DO
      BEGIN
        write(f,'  ');
        FOR i := 0 to nx + 1 DO
          write(f,' ',u[i,j]:9:4);
        writeln(f);
      END; {for j}
    write(f,'            ');
    FOR i := 1 to nx DO
    write(f,' ',u[i,0]:9:4);
    writeln(f);  writeln(f);
    writeln(f,'    iterations = ',k:3);
    writeln(f,'    max |u - old_u| = ',
              MaxDif(u,old_u):10:6);
    close(f);
  END.
```

```
  ********  ASSIGNMENT 2,  QUESTION 3  ********

  S.O.R. Method for the Laplace equation:
  omega = 1.1128
  tolerance = 0.000010
  max.  iterations = 100

  Solution and boundary values:
```

```
            40.0000    30.0000    10.0000
  80.0000   52.1988    32.4224    14.1988     0.0000
  80.0000   56.3727    33.2919    14.3727     0.0000
            60.0000    30.0000    10.0000

iterations =   10
max |u - old_u| =    0.000006
```

Question 4

Solve the problem in Question 3 by using the **A.D.I. method** (alternating–direction–implicit method) with $\Delta x = \Delta y = 3$ cm and $\rho = 1.0$.

SOLUTION

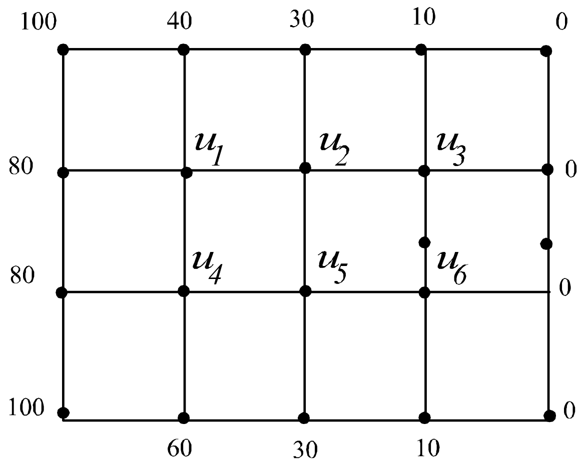
The A.D.I. method for solving Laplace's equation is discussed in Section 7.8 [7.10] of Gerald. The equations for the horizontal and vertical traverses, respectively, are given by

$$\begin{aligned} & -u_L^{(k+1)} + \left(\frac{1}{\rho} + 2\right) u_0^{(k+1)} - u_R^{(k+1)} \\ = \; & u_A^{(k)} + \left(\frac{1}{\rho} - 2\right) u_0^{(k)} + u_B^{(k)} \end{aligned} \tag{1}$$

and

$$\begin{aligned} & -u_A^{(k+2)} + \left(\frac{1}{\rho} + 2\right) u_0^{(k+2)} - u_B^{(k+2)} \\ = \; & u_L^{(k+1)} + \left(\frac{1}{\rho} - 2\right) u_0^{(k+1)} + u_R^{(k+1)}, \end{aligned} \tag{2}$$

where ρ is the acceleration factor.



Let $u_1, ..., u_6$ be as in the sketch above. Let $\rho = 1$ and apply equation (1) at each of the interior mesh points, proceeding row–wise. We get

$$\begin{array}{lll}
(r1) & -80 + 3u_1^{(k+1)} - u_2^{(k+1)} & = 40 - u_1^{(k)} + u_4^{(k)} \\
(r2) & -u_1^{(k+1)} + 3u_2^{(k+1)} - u_3^{(k+1)} & = 30 - u_2^{(k)} + u_5^{(k)} \\
(r3) & -u_2^{(k+1)} + 3u_3^{(k+1)} - 0 & = 10 - u_3^{(k)} + u_6^{(k)} \\
(r4) & -80 + 3u_4^{(k+1)} - u_5^{(k+1)} & = u_1^{(k)} - u_4^{(k)} + 60 \\
(r5) & -u_4^{(k+1)} + 3u_5^{(k+1)} - u_6^{(k+1)} & = u_2^{(k)} - u_5^{(k)} + 30 \\
(r6) & -u_5^{(k+1)} + 3u_6^{(k+1)} - 0 & = u_3^{(k)} - u_6^{(k)} + 10
\end{array}$$

It only remains to take all the constants in equations (r1) – (r6) to the right hand side. Similarly, by applying (2) at every mesh point, proceeding column–wise, we obtain (here we have already moved the constants to the right):

$$\begin{array}{lll}
(c1) & 3u_1^{(k+2)} - u_4^{(k+2)} & = 80 - u_1^{(k+1)} + u_2^{(k+1)} + 40 \\
(c2) & -u_1^{(k+2)} + 3u_4^{(k+2)} & = 80 - u_4^{(k+1)} + u_5^{(k+1)} + 60 \\
(c3) & 3u_2^{(k+2)} - u_5^{(k+2)} & = u_1^{(k+1)} - u_2^{(k+1)} + u_3^{(k+1)} + 30 \\
(c4) & -u_2^{(k+2)} + 3u_5^{(k+2)} & = u_4^{(k+1)} - u_5^{(k+1)} + u_6^{(k+1)} + 30 \\
(c5) & 3u_3^{(k+2)} - u_6^{(k+2)} & = u_2^{(k+1)} - u_3^{(k+1)} + 0 + 10 \\
(c6) & -u_3^{(k+2)} + 3u_6^{(k+2)} & = u_5^{(k+1)} - u_6^{(k+1)} + 0 + 10
\end{array}$$

Observe that, as in the case of the S.O.R. method, the boundary values given at the corners of the plate do not appear in any of the equations. This is due to the fact that equations (1) and (2) are based on the five–point formula (1).

We have written out the row and column equations using only one numbering, u_1 to u_6, for the nodes. In the program, the two tridiagonal matrices are generated automatically, and the user only needs to type in the boundary values at the appropriate places.
As explained in Tutorial letter 102, we can also introduce the double notation for the nodes (u_i for rowwise numbering, and v_i for columnwise numbering), write down the two sets of equations as matrix equations $Au = c$, $Bv = d$ and then provide these matrices A, B and vectors c, d as input to the program. The downside is that the user has to set up the equations, whereas in the program that follows, that is done automatically!
The algorithm is as follows:

1. Define ϵ and $\max\ k$.

2. Let $k = 0$ and define the initial estimate $u_i^{(0)}$, $i = 1, 2, ..., 12$.

Repeat 3 – 6 until 7 is satisfied:

3. Use $u_1^{(k)}, ..., u_6^{(k)}$ to define the right hand sides of equations (r1),...,(r6), then solve for $u_1^{(k+1)}, ..., u_6^{(k+1)}$.

4. Use $u_1^{(k+1)}, ..., u_6^{(k+1)}$ to define the right hand sides of equations (c1),...,(c6), then solve for $u_1^{(k+2)}, ..., u_6^{(k+2)}$.

5. Print $u_1^{(k+2)}, ..., u_6^{(k+2)}$.

6. Let $k := k + 2$.
7. $\max\limits_i \left| u_i^{(k)} - u_i^{(k-2)} \right| < \epsilon$ OR $k = \max\ k$

The program and results follow. The program is based on the program on pages 585–589 of Gerald [program ADIELL, pp. 606–609]. The following table shows the effect of varying ρ :

ρ	iterations
0.3	24
0.4	18
0.5	16
0.6	14
0.7	16
0.8	16
0.9	18
1.0	20
1.1	22

```
PROGRAM AS02_5_4(output);
  CONST
    rho = 1.0;
    rows = 2;
    cols = 3;
    size = rows*cols;
    tol = 1.0E-5;
    itermax = 100;
    c = 1.0/rho - 2.0;
  TYPE
    vector = array[1..size] of real;
    matrix = array[1..size,1..3] of real;
    rowvector = array[1..cols] of real;
    colvector = array[1..rows] of real;
  VAR
    u_coef, v_coef :  matrix;
    u, v, old_u, u_bcnd, v_bcnd :  vector;
    top, bottom :  rowvector;
    left, right :  colvector;
    i, j, k, l :  integer;
    sum :  real;
    f :  text;

  FUNCTION MaxDif(x, y :  vector) :  real;
    VAR
      i :  integer;
      d :  real;
    BEGIN
      d := 0.0;
      FOR i := 1 to size DO
        IF abs(x[i] - y[i]) > d THEN
          d := abs(x[i] - y[i]);
      MaxDif := d;
    END; {MaxDif}

  PROCEDURE Solve(coef :  matrix;
                  bcnd, y :  vector;
                  m, n :  integer;
                  VAR x :  vector);
  (* Solves (coef)x = b with the LU form of coef
     given in coef, b determined by bcnd and y.  *)
    VAR
      i, j, k :  integer;
    BEGIN
      (* Compute r.h.s.  vector b, store it in x:  *)
      FOR i := 1 to n DO BEGIN
        j := (i - 1)*m + 1;
        x[i] := c*y[j] + y[j + 1] + bcnd[i];
        k := size - n + i;
        j := i*m;
        x[k] := y[j - 1] + c*y[j] + bcnd[k];
        END; {for i}
      FOR i := 2 to (m - 1) DO
        FOR j := 1 to n DO BEGIN
          k := (i - 1)*n + j;
          l := i + (j - 1)*m;
          x[k] := y[l - 1] + c*y[l] + y[l + 1]
                  + bcnd[k];
          END; {for j}

      (* Forward substitution to get z = L(-1)b:  *)
      x[1] := x[1]/coef[1,2];
      FOR i := 2 to size DO
        x[i] := (x[i] - coef[i,1]*x[i-1])/coef[i,2];

      (* Backward substitution to get x = U(-1)z:  *)
      FOR j := (size - 1) downto 1 DO
        x[j] := x[j] - coef[j,3]*x[j + 1];
    END; {Solve}

  BEGIN
    (* Define the boundary values:  *)
    top[1] := 40.0;  top[2] := 30.0;
    top[3] :=  10.0;
```

```
bottom[1] := 60.0;  bottom[2] := 30.0;
bottom[3] := 10.0;
left[1] := 80.0;  left[2] := 80.0;
FOR j := 1 to rows DO
  right[j] := 0.0;

(* Find the average of the boundary values
   and define the initial estimate of u:  *)
sum := 0.0;
FOR i := 1 to rows DO
  sum := sum + left[i] + right[i];
FOR i := 1 to cols DO
  sum := sum + top[i] + bottom[i];
FOR i := 1 to size DO
  u[i] := sum/(2*rows + 2*cols);

(* Establish the coefficient matrices:  *)
FOR i := 1 to size DO BEGIN
  u_coef[i,1] := -1.0;
  u_coef[i,2] := 1.0/rho + 2.0;
  u_coef[i,3] := -1.0;
  FOR j := 1 to 3 DO
    v_coef[i,j] := u_coef[i,j];
  END; {for i}
FOR i := 1 to (rows - 1) DO BEGIN
  u_coef[i*cols,3] := 0.0;
  u_coef[i*cols + 1,1] := 0.0;
  END; {for i}
u_coef[1,1] := 0.0;
u_coef[size,3] := 0.0;
FOR i := 1 to (cols - 1) DO BEGIN
  v_coef[i*rows,3] := 0.0;
  v_coef[i*rows + 1,1] := 0.0;
  END; {for i}
v_coef[1,1] := 0.0;
v_coef[size,3] := 0.0;
 (* Get boundary values into bcnd vectors:  *)
FOR i := 1 to size DO BEGIN
  u_bcnd[i] := 0.0;
  v_bcnd[i] := 0.0;
  END; {for i}
FOR i := 1 to cols DO BEGIN
  u_bcnd[i] := top[i];
  u_bcnd[size - cols + i] := bottom[i];
  END; {for i}
FOR i := 1 to rows DO BEGIN
  j := (i-1)*cols + 1;
  u_bcnd[j] := u_bcnd[j] + left[i];
  j := i*cols;
  u_bcnd[j] := u_bcnd[j] + right[i];
  END; {for i}
FOR i := 1 to rows DO BEGIN
  v_bcnd[i] := left[i];
  v_bcnd[size - rows + i] := right[i];
  END; {for i}
FOR i := 1 to cols DO BEGIN
  j := (i-1)*rows + 1;
  v_bcnd[j] := v_bcnd[j] + top[i];
  j := i*rows;
  v_bcnd[j] := v_bcnd[j] + bottom[i];
  END; {for i}

(* Replace the coefficient matrices by their
   LU decompositions:  *)
FOR i := 2 to size DO BEGIN
  u_coef[i-1,3] := u_coef[i-1,3]/u_coef[i-1,2];
  u_coef[i,2] := u_coef[i,2]
               - u_coef[i,1]*u_coef[i-1,3];
  v_coef[i-1,3] := v_coef[i-1,3]/v_coef[i-1,2];
  v_coef[i,2] := v_coef[i,2]
               - v_coef[i,1]*v_coef[i-1,3];
  END; {for i}

  k := 0;
REPEAT
  k := k + 2;
  FOR i := 1 to size DO
    old_u[i] := u[i];
  Solve(v_coef, v_bcnd, u, cols, rows, v);
  Solve(u_coef, u_bcnd, v, rows, cols, u);
UNTIL (MaxDif(old_u,u) < tol) OR (k >= itermax);

assign(f,'work\AS02_5_4.DAT');
rewrite(f);
writeln(f);  writeln(f);
writeln(f,'    ********  ASSIGNMENT 2, ',
          'QUESTION 4  ********');    writeln(f);
writeln(f,'    A.D.I. Method for the Laplace',
          ' equation:');
writeln(f,'    rho = ',rho:4:2);
writeln(f,'    tolerance = ',tol:10:6);
writeln(f,'    max.  iterations = ',itermax);    writeln(f);
writeln(f,'    Solution:');    writeln(f);
(* Print solution, 3 values per line:  *)
```

```
  FOR i := 1 to (size div 3) DO BEGIN
    FOR j := 1 to 3 DO BEGIN
      l := (i - 1)*3 + j;
      write(f,'    u',l:1,' = ',u[l]:8:4);
      END; {for j}
    writeln(f);
    END; {for i}
  FOR l := (size div 3)*3 + 1 to size DO
    write(f,'    u',l:1,' = ',u[l]:8:4);
  writeln(f);
  writeln(f,'    iterations = ',k);
  writeln(f,'    max |u - old_u| = ',
            MaxDif(old_u,u):8:6);
  close(f);
END.
```

```
  ********  ASSIGNMENT 2, QUESTION 4  ********

  A.D.I. Method for the Laplace equation:
  rho = 1.00
  tolerance =   0.000010
  max.  iterations = 100

  Solution:

  u1 =  52.1988    u2 =  32.4224    u3 =  14.1988
  u4 =  56.3727    u5 =  33.2919    u6 =  14.3727

  iterations = 20
  max |u - old_u| = 0.000003
```