Tutorial Letter 202/1/2013

Numerical Methods II

APM3711

Semester 1

Department of Mathematical Sciences

Solutions to Assignment 2

BAR CODE

ONLY FOR SEMESTER 1 STUDENTS

ASSIGNMENT 02

FIXED CLOSING DATE: 16 APRIL 2013
Unique Number: 282179

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) the dominant eigenvalue and the associated eigenvector,

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

(c) 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

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

(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

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

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:

$\underline{\lambda = -4}$

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

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

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

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

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

$\underline{\lambda = 3}$

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

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

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

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

(b) **The power method**

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

Thus for $Ax = \lambda x$:

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

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

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.

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

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

$$= 0.4167 \begin{bmatrix} 0.8667 \\ -0.5877 \\ 1 \end{bmatrix}$$

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, λ_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 $(A+6I)^{-1}$:

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

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

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

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

$$\therefore \quad (A+6I)^{-1} = \begin{bmatrix} \frac{11}{18} & 0 & -\frac{1}{3} \\ -\frac{10}{54} & \frac{1}{15} & \frac{4}{45} \\ \frac{1}{6} & 0 & 0 \end{bmatrix}$$

Now the power method is applied

$$\begin{bmatrix} \frac{11}{18} & 0 & -\frac{1}{3} \\ -\frac{10}{54} & \frac{1}{15} & \frac{4}{45} \\ \frac{1}{6} & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.2777 \\ -0.0296 \\ 0.1667 \end{bmatrix}$$

$$= 0.2777 \begin{bmatrix} 1 \\ -0.1067 \\ 0.6000 \end{bmatrix}$$

$$\begin{bmatrix} \frac{11}{18} & 0 & -\frac{1}{3} \\ -\frac{10}{54} & \frac{1}{15} & \frac{4}{45} \\ \frac{1}{6} & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -0.1067 \\ 0.6000 \end{bmatrix} = \begin{bmatrix} 0.4111 \\ -0.1390 \\ 0.1667 \end{bmatrix}$$

$$= 0.4111 \begin{bmatrix} 1 \\ -0.3380 \\ 0.4054 \end{bmatrix}$$

After 2 iterations we obtain

$$\frac{1}{\lambda_3 + 6} = 0.4111 \qquad \therefore \ \lambda_3 = \frac{1}{0.4111} - 6 = -3.5676$$

$$x = (1, \ -0.3380, \ 0.4054) .$$

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

Consider the characteristic–value problem

$$y'' - x^2 y' + ky = 0, \quad y(0) = y(1) = 0.$$

Taking $h = 0.2$, derive an eigenvalue problem for determining the non-zero values of k for which the differential equation has non-trivial solutions. (Do not solve the eigenvalue problem.)

SOLUTION

We proceed as follows:

Let

$$x_i = i \, (0.2) , \qquad y_i = y \, (x_i) , \qquad i = 0, 1, \ldots, 5,$$

so that

$$y_0 = 0, \qquad y_5 = 0. \tag{2}$$

We approximate the derivatives in (1) by second–order central differences. With $h = 0.2$ this gives

$$\frac{1}{(0.2)^2} (y_{i+1} - 2y_i + y_{i-1}) - \frac{x_i^2}{2 \, (0.2)} (y_{i+1} - y_{i-1}) + ky_i = 0$$

$$\therefore -\left(\frac{5}{2}x_i^2 + 25\right) y_{i-1} + 50y_i + \left(\frac{5}{2}x_i^2 - 25\right) y_{i+1} = ky_i, \quad i = 1, \ldots, 4.$$

Notice that we have taken all the terms in which k appears to the right hand side. Simplification of the coefficients, using (2), yields

$$\begin{aligned} 50y_1 - 24.9y_2 &= ky_1 \\ -25.4y_1 + 50y_2 - 24.6y_3 &= ky_2 \\ -25.9y_2 + 50y_3 - 24.1y_4 &= ky_3 \\ -26.6y_3 + 50y_4 &= ky_4. \end{aligned}$$

Hence we have the following **eigenvalue problem:**

Find the (non–zero) eigenvalues k and the corresponding eigenvectors y of the matrix M, i.e. k and y such that $My = ky$, where

$$M = \begin{bmatrix} 50 & -24.9 & 0 & 0 \\ -25.4 & 50 & -24.6 & 0 \\ 0 & -25.9 & 50 & -24.1 \\ 0 & 0 & -26.6 & 50 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}.$$

Question 3

Solve the boundary–value problem

$$\begin{aligned} y'' + (y')^2 - 9xy &= -9x^3 + 36x^2 + 6x - 6, \\ y(1) &= -2, \quad y(2) = -4 \end{aligned}$$

by using the **shooting method.** Use the modified Euler method (with only one correction at each step), and take $h = 0.2$. Start with an initial slope of $y'(1) = -2.99$ as a first attempt and $y'(1) = -3.01$ as a second attempt. Then interpolate. Continue until the solutions corresponding to two consecutive estimates of $y'(1)$ agree in at least 2 decimal places. Compare the result with the analytical solution $y = x^3 - 3x^2$.

SOLUTION

Set

$$z = \frac{dy}{dx} = y',$$

then

$$z' = \frac{dz}{dx} = y''.$$

The second–order differential equation (1) can thus be written as a system of two coupled first–order differential equations:

$$y' = z \tag{2}$$

$$z' = 9xy - z^2 - 9x^3 + 36x^2 + 6x - 6 = g(x, y, z) \tag{3}$$

with boundary conditions

$$y(1) = -2, \quad y(2) = -4.$$

To solve (2)–(3) we must estimate $z(0) = y'(0)$. For a chosen estimate z_0 the algorithm is

Predictor	Corrector
$z_p = z_i + hz'_i$	$z_{i+1} = z_i + h\frac{1}{2}\left[z'_i + z'\left(x_i + h, y_p, z_p\right)\right]$
$y_p = y_i + hy'_i$	$y_{i+1} = y_i + h\frac{1}{2}\left[y'_i + y'\left(x_i + h, y_p, z_p\right)\right]$.

Sequence of calculations:

$$
\begin{aligned}
x_0 &= 1 \\
y_0 &= -2 \\
z_0 &= \text{estimate} \\
z'_i &= g(x_i, y_i, z_i) \\
z_p &= z_i + hz'_i \\
y'_i &= z_i \\
y_p &= y_i + hy'_i \\
z'_p &= g\left(x_i + h, y_p, z_p\right) \\
z_{i+1} &= z_i + h\left(z'_i + z'_p\right)/2 \\
y'_p &= z_{i+1} \\
y_{i+1} &= y_i + h\left(y'_i + y'_p\right)/2 \\
x_{i+1} &= x_i + h.
\end{aligned}
$$

Note that we have improved the algorithm slightly by using the corrected value z_{i+1} instead of z_p to calculate y'_p.
The question suggested the estimates

$$
\begin{aligned}
z_0 &= -2.99 \quad \text{(first attempt)} \\
z_0 &= -3.01 \quad \text{(second attempt)}
\end{aligned}
$$

The third estimate is calculated by using the extrapolation formula:

$$z_0 = G_1 + \frac{G_2 - G_1}{R_2 - R_1}(D - R_1)$$

where

$$\begin{aligned} G_1 &= -2.99 \\ G_2 &= -3.01 \\ D &= -4 \end{aligned}$$

$$\begin{aligned} R_1 &= y(2) \quad \text{calculated when } z_0 = -2.99 \\ R_2 &= y(2) \quad \text{calculated when } z_0 = -3.01. \end{aligned}$$

This procedure of calculation and extrapolation is repeated, each time using the previous two estimates of z_0 and the corresponding calculated values of $y(2)$, until the difference of -4 and the value of $y(2)$ calculated for the last extrapolated estimate z_0, as well as the maximum difference on $[1, 2]$ of the solutions y corresponding to two consecutive estimates of z_0, are less than a chosen tolerance.
A disadvantage of the shooting method is that it does not always converge when the problem is nonlinear and the initial estimates of z are not sufficiently close to the exact value. In practice, i.e. when the exact solution is unknown, this can be a problem if additional information about y' is not available. This is illustrated by our problem when we use the initial estimates $z_0 = -2.9$ and $z_0 = -3.1$. Note the large values produced at $x = 2$ in iterations 2 and 5 where zinitial < -3. Note also that although the exact values for $y(2)$ and $y(4)$ are obtained the intermediate values and the values for z are not very accurate. This is due to the inaccuracy of the modified Euler method for the large step size $h = 0.2$.

```
PROGRAM AS_3_1(output);
  CONST
    xinitial = 1.0;
    xfinal = 2.0;
    yinitial = -2.0;
    yfinal = -4.0;
    zinit1 = -2.99;   zinit2 = -3.01;
  { zinit1 = -2.9;   zinit2 = -3.1; }
    h = 0.2;
    tol = 1e-3;
    imax = round((xfinal - xinitial)/h);
    itermax = 20;
  TYPE
    index = 1..imax;
    iter = 1..itermax;
    sol = array[index] of real;
  VAR
    i :  0..imax;
    j :  iter;
    x, y, z, diff, d :  real;
    zi, yf :  array[iter] of real;
    pre_y :  sol;
    fst :  text;

  FUNCTION fy(x,y,z :  real) :  real;
```

```
  (*   calculates y'  *)
    BEGIN
      fy := z;
    END; {fy}

  FUNCTION fz(x,y,z :  real) :  real;
  (*   calculates z'  *)
    BEGIN
      fz := - sqr(z) + 9*x*y
            + (36 - 9*x)*sqr(x) + 6*x - 6;
    END; {fz}
  PROCEDURE ModEuler(j :  integer;
                     zinitial :  real;
                     VAR yend, diff :  real;
                     VAR pre_y :  sol);
    VAR
      d, x, y, z, yp, zp, fy0, fz0 :  real;
      i :  index;
    BEGIN
      x := xinitial;
      y := yinitial;
      z := zinitial;
      diff := 0.0;
      writeln(fst);
      writeln(fst,'    iteration ',j,':',
             '  zinitial = ',zinitial:10:6);
      writeln(fst,
'         x           y           z');
      writeln(fst,x:12:6, y:12:6, z:12:6);
      FOR i := 1 to imax DO
        BEGIN
          fz0 := fz(x,y,z);
          zp := z + h*fz0;
          fy0 := fy(x,y,z);
          yp := y + h*fy0;
          x := x + h;
          z := z + h*(fz0 + fz(x,yp,zp))/2;
          y := y + h*(fy0 + fy(x,yp,z))/2;
          writeln(fst,x:12:6, y:12:6, z:12:6);
          IF j >= 2 THEN
            BEGIN
              d := abs(y - pre_y[i]);
              IF d > diff THEN
                diff := d;
            END; {if j}
          pre_y[i] := y;
        END; {for i}
```

```
      yend := pre_y[imax];
      writeln(fst,'    error in final y = ',
             (yfinal - yend):10:6);
      IF j >= 2 THEN
        writeln(fst,
'     max|y - pre_y| = ',diff:10:6);
    END; {ModEuler}

  BEGIN {program}
    assign(fst,'a:\as01_3_1.dat');
    rewrite(fst);
    writeln(fst); writeln(fst);
    writeln(fst,
'     *****  ASSIGNMENT 3, QUESTION 1',
          '  *****');
    writeln(fst);
    writeln(fst,'    tolerance = ',tol:6:3);
    writeln(fst,'    maximum iterations = ',itermax);

    zi[1] := zinit1;
    zi[2] := zinit2;
    FOR j := 1 to 2 DO
      ModEuler(j, zi[j], yf[j], diff, pre_y);

    REPEAT
      (*  Interpolate the previous two initial values of
          z to find the next estimate of zinitial:   *)
      d := yf[j] - yf[j - 1];
      (*  Test to avoid division by zero:   *)
      IF abs(d) >= 1.0E-10 THEN
        BEGIN
          zi[j + 1] := zi[j - 1] + (zi[j] - zi[j - 1])*
                       (yfinal - yf[j - 1])/d;
          j := j + 1;
          ModEuler(j, zi[j], yf[j], diff, pre_y);
        END {if abs(d)}
      ELSE
        writeln(fst,'    DIVISION BY ZERO');
    UNTIL ((diff < tol) and (abs(yf[j] - yfinal) < tol))
         or (j = itermax) or (abs(d) < 1.0E-10);
    writeln(fst);
    writeln(fst,'    Exact solution');
    writeln(fst,'        x           y           z');
    FOR i := 0 to imax DO
      BEGIN
        x := xinitial + i*h;
        y := (x - 3)*sqr(x);
```

```
      z := 3*x*(x - 2);
      writeln(fst,x:12:6, y:12:6, z:12:6);
    END; {for i}
  close(fst);
END.
```

```
  *****  ASSIGNMENT 3, QUESTION 1  *****

  tolerance =  0.001
  maximum iterations = 20

  iteration 1:   zinitial =  -2.990000
     x           y           z
  1.000000   -2.000000   -2.990000
  1.200000   -2.591791   -2.927910
  1.400000   -3.161089   -2.765066
  1.600000   -3.720067   -2.824720
  1.800000   -4.428752   -4.262123
  2.000000   -6.376087  -15.211230
  error in final y =   2.376087

  iteration 2:   zinitial =  -3.010000
     x           y           z
  1.000000   -2.000000   -3.010000
  1.200000   -2.600063   -2.990630
  1.400000   -3.197867   -2.987410
  1.600000   -3.863464   -3.668557
  1.800000   -5.080148   -8.498290
  2.000000  -14.635467  -87.054898
  error in final y =  10.635467
  max|y - pre_y| =   8.259380

  iteration 3:   zinitial =  -2.984246
     x           y           z
  1.000000   -2.000000   -2.984246
  1.200000   -2.589422   -2.909971
  1.400000   -3.150683   -2.702642
  1.600000   -3.681220   -2.602728
  1.800000   -4.279109   -3.376161
  2.000000   -5.512105   -8.953800
  error in final y =   1.512105
  max|y - pre_y| =   9.123362
  iteration 4:   zinitial =  -2.979978
     x           y           z
  1.000000   -2.000000   -2.979978
  1.200000   -2.587667   -2.896692
```

```
1.400000   -3.143013   -2.656767
1.600000   -3.653050   -2.443603
1.800000   -4.176625   -2.792149
2.000000   -5.039964   -5.841245
error in final y =   1.039964
max|y - pre_y| =   0.472141

iteration 5:   zinitial =  -2.970576
    x           y           z
1.000000   -2.000000   -2.970576
1.200000   -2.583811   -2.867534
1.400000   -3.126266   -2.557015
1.600000   -3.592870   -2.109026
1.800000   -3.972977   -1.692045
2.000000   -4.308304   -1.661223
error in final y =   0.308304
max|y - pre_y| =   0.731660

iteration 6:   zinitial =  -2.966614
    x           y           z
1.000000   -2.000000   -2.966614
1.200000   -2.582190   -2.855284
1.400000   -3.119269   -2.515509
1.600000   -3.568252   -1.974322
1.800000   -3.895126   -1.294417
2.000000   -4.082087   -0.575188
error in final y =   0.082087
max|y - pre_y| =   0.226217

iteration 7:   zinitial =  -2.965177
    x           y           z
1.000000   -2.000000   -2.965177
1.200000   -2.581602   -2.850844
1.400000   -3.116739   -2.500524
1.600000   -3.559424   -1.926329
1.800000   -3.867920   -1.158628
2.000000   -4.008374   -0.245911
error in final y =   0.008374
max|y - pre_y| =   0.073713

iteration 8:   zinitial =  -2.965013
    x           y           z
1.000000   -2.000000   -2.965013
1.200000   -2.581535   -2.850340
1.400000   -3.116452   -2.498824
1.600000   -3.558425   -1.920907
1.800000   -3.864863   -1.143475
```

```
2.000000   -4.000249   -0.210381
error in final y =   0.000249
max|y - pre_y| =   0.008125

iteration 9:   zinitial =  -2.965008
    x           y           z
1.000000   -2.000000   -2.965008
1.200000   -2.581533   -2.850325
1.400000   -3.116443   -2.498772
1.600000   -3.558394   -1.920741
1.800000   -3.864770   -1.143012
2.000000   -4.000001   -0.209300
error in final y =   0.000001
max|y - pre_y| =   0.000248

Exact solution
    x           y           z
1.000000   -2.000000   -3.000000
1.200000   -2.592000   -2.880000
1.400000   -3.136000   -2.520000
1.600000   -3.584000   -1.920000
1.800000   -3.888000   -1.080000
2.000000   -4.000000    0.000000
```

Question 4

Consider the partial differential equation

$$yu - 2\nabla^2 u = 12, \quad 0 < x < 4, \quad 0 < y < 3$$

with boundary conditions

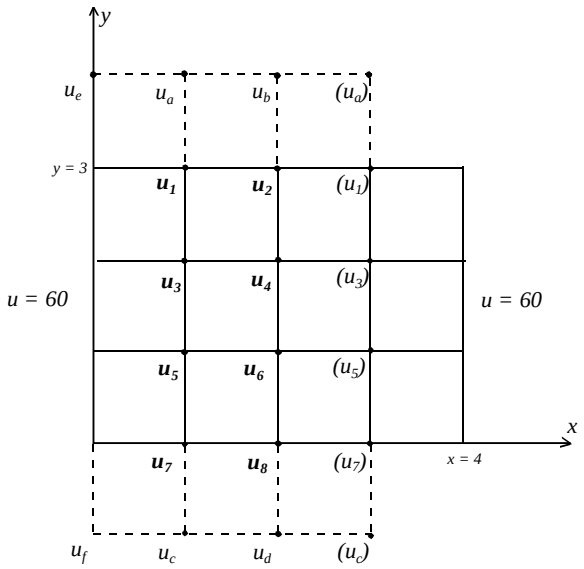
$$\begin{aligned} x &= 0 \text{ and } x = 4: \quad u = 60 \\ y &= 0 \text{ and } y = 3: \quad \frac{\partial u}{\partial y} = 5. \end{aligned}$$

(a) Taking $h = 1$, sketch the region and the grid points. Use symmetry to minimize the number of unknowns u_i that have to be calculated and indicate the u_i in the sketch.

(b) Use the 5-point difference formula for the Laplace operator to derive a system of equations for the u_i.

SOLUTION

(a) The line $x = 2$ is symmetry–line because the domain, boundary conditions and all the terms in the differential equation are symmetric with respect to this line. The unknowns that have to be determined, are $u_1, u_2, \ldots, u_8$. The numbers in brackets indicate how the symmetry is used to determine u at the other mesh points. Fictitious points $u_a, \ldots, u_f$, have been added to the mesh in order to deal with the derivative boundary conditions. Note that u_e and u_f will only be needed in the 9–point formula.



The line $y = 1.5$ is not a symmetry–line because:

(1) The coefficient y of u in the differential equation is not a symmetric function with respect to this line, and

(2) the boundary conditions at $y = 0$ and $y = 3$ cannot be satisfied by a function which is symmetric about $y = 1.5$. To see this, consider a function $v\,(x, y)$ which we assume is symmetric about this line, i.e.

$$v\,(x, 1.5 - z) = v\,(x, 1.5 + z)\,, \quad 0 \leq x \leq 4, \quad 0 \leq z \leq 1.5.$$

Denote the functions in this equation by $w\,(x, z)$, i.e. $w\,(x, z) = v\,(x, 1.5 - z) = v\,(x, 1.5 + z)$. From the two definitions of w and the chain rule we get

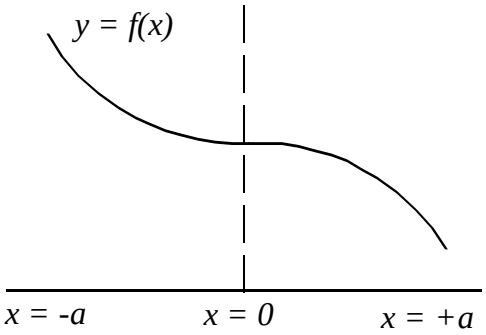
$$\frac{\partial w}{\partial z} = \frac{\partial}{\partial z} v\,(x, 1.5 - z) = -\frac{\partial v}{\partial y}\bigg|_{y=1.5-z}$$

$$\frac{\partial w}{\partial z} = \frac{\partial}{\partial z} v\,(x, 1.5 + z) = +\frac{\partial v}{\partial y}\bigg|_{y=1.5+z}$$

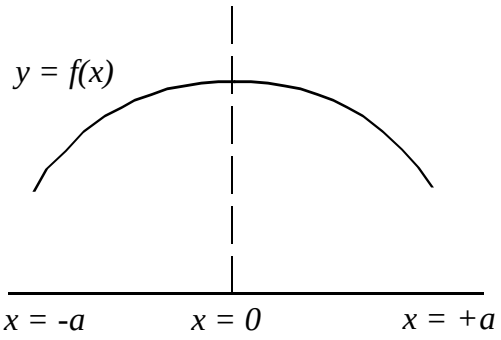
With $z = 1.5$ this gives

$$-\frac{\partial v}{\partial y}\bigg|_{y=0} = \frac{\partial v}{\partial y}\bigg|_{y=3}$$

Thus for symmetry the derivatives must have opposite signs- see the following figure.



f(x) is not symmetric about *x=0*,
f'(-a) = f'(a) = -1



f(x) is symmetric about *x=0*,
f'(-a) = 1, f'(a) = -1

This also applies to boundary conditions with higher–order derivatives of odd order. If the order of differentiation is even, the boundary values must be equal for symmetry to be possible.

(b) Since $\triangle x = \triangle y = 1$ in the mesh above, we can approximate the Laplace operator by the 5–point difference formula on p. 554 [p. 555] of *Gerald*:

$$\begin{aligned} \nabla^2 u\left(x_i, y_j\right) &= \frac{1}{1^2}\begin{Bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{Bmatrix} u_{ij} \\ &= u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}. \end{aligned}$$

Substitution of this into the differential equation yields the difference equation

$$\left(y_j + 8\right) u_{i,j} - 2\left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}\right) = 12. \qquad (*)$$

Apply (*) at each of the eight internal mesh points. Then we obtain the following system of linear equations:

(1) $(3+8)\,u_1 - 2\,(u_2 + 60 + u_a + u_3) = 12$
(2) $(3+8)\,u_2 - 2\,(u_1 + u_1 + u_b + u_4) = 12$
(3) $(2+8)\,u_3 - 2\,(u_4 + 60 + u_1 + u_5) = 12$
(4) $(2+8)\,u_4 - 2\,(u_3 + u_3 + u_2 + u_6) = 12$
(5) $(1+8)\,u_5 - 2\,(u_6 + 60 + u_3 + u_7) = 12$
(6) $(1+8)\,u_6 - 2\,(u_5 + u_5 + u_4 + u_8) = 12$
(7) $(0+8)\,u_7 - 2\,(u_8 + 60 + u_5 + u_c) = 12$
(8) $(0+8)\,u_8 - 2\,(u_7 + u_7 + u_6 + u_d) = 12$

we determine u_a, u_b, u_c and u_d by means of the central–difference formula

$$\left.\frac{\partial u}{\partial y}\right|_{\substack{x=x_i \\ y=y_i}} = \frac{u_{i,j+1} - u_{i,j-1}}{2h}.$$

This gives

$$\frac{u_a - u_3}{2 \cdot 1} = \left.\frac{\partial u}{\partial y}\right|_{\substack{x=1 \\ y=3}} = 5 \qquad \therefore \quad u_a = u_3 + 10 \tag{9}$$

and similarly

$$\frac{u_b - u_4}{2} = 5 \qquad \therefore \quad u_b = u_4 + 10 \tag{10}$$

$$\frac{u_5 - u_c}{2} = 5 \qquad \therefore \quad u_c = u_5 - 10 \tag{11}$$

$$\frac{u_6 - u_d}{2} = 5 \qquad \therefore \quad u_d = u_6 - 10 \tag{12}$$

Here one can clearly see that the above boundary conditions are not symmetric with respect to the line $y = 1.5$, since otherwise the equations (9) and (10) would have been the same form as (11) and (12), respectively.

Substituting (9) – (12) in (1) – (8), we obtain a system of equations which looks as follows in matrix notation:

$$\begin{bmatrix} 11 & -2 & -4 & & & & & \\ -4 & 11 & & -4 & & & & \\ -2 & & 10 & -2 & -2 & & & \\ & -2 & -4 & 10 & & -2 & & \\ & & -2 & & 9 & -2 & -2 & \\ & & & -2 & -4 & 9 & & -2 \\ & & & & -4 & & 8 & -2 \\ & & & & & -4 & -4 & 8 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{bmatrix} = \begin{bmatrix} 152 \\ 32 \\ 132 \\ 12 \\ 132 \\ 12 \\ 112 \\ -8 \end{bmatrix}$$

For the sake of interest we calculate the solution. Since the size of the matrix is small, the system of equations should be solved by a direct method, e.g. Gaussian elimination with partial pivoting. See *Gerald*, sections 2.4 and 2.5. The solution below was found by using a Pascal program which applies Gaussian elimination with partial pivoting and back–substitution.

$$\begin{aligned} u_1 &= 29.4371 & u_2 &= 22.6047 \\ u_3 &= 31.6497 & u_4 &= 24.7259 \\ u_5 &= 38.0855 & u_6 &= 31.7254 \\ u_7 &= 42.0096 & u_8 &= 35.8675 \end{aligned}$$

(c) The difference equation based on the 9–point formula (7.13) [(7.8)] is

$$y_j u_{i,j} - \frac{2}{6 \cdot 1^2} \begin{Bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{Bmatrix} u_{i,j} = 12$$

i.e.

$$\begin{aligned} &\left(3y_j + 20\right) u_{i,j} - 4\left(u_{i-1,j} + u_{i+1,j} + u_{i,j+1} + u_{i,j-1}\right) \\ &- \left(u_{i-1,j+1} + u_{i+1,j+1} + u_{i+1,j-1} + u_{i-1,j-1}\right) = 36. \qquad ((**)) \end{aligned}$$

In order to apply (**) at u_1 and u_7, the value of u at the two additional fictitious points, u_e at $(0, 4)$ and u_f at $(0, -1)$, must first be found. As in (b) we have

$$\frac{u_e - 60}{2} = \left.\frac{\partial u}{\partial y}\right|_{\substack{x=0 \\ y=3}} = 5 \qquad \therefore \quad u_e = 70 \tag{13}$$

On the other hand, the condition $u = 60$ at $x = 0$ implies that

$$\left.\frac{\partial u}{\partial y}\right|_{\substack{x=0 \\ y=3}} = 0 \qquad \therefore \quad \frac{u_e - 60}{2} = 0 \qquad \therefore \quad u_e = 60 \tag{14}$$

Hence the two boundary conditions are inconsistent. As a compromise we use the average of the values (13) and (14) (or equivalently, $\frac{\partial u}{\partial y} = \frac{5}{2}$ at $(0, 3)$):

$$u_e = 65 \tag{15}$$

An analogous argument yields

$$y_f = 55 \tag{16}$$

The applications of (**), (9) – (12) and (15) – (16) produces the following system of equations:

$$\begin{bmatrix} 29 & -4 & -8 & -2 & & & & \\ -8 & 29 & -4 & -8 & & & & \\ -4 & -1 & 26 & -4 & -4 & -1 & & \\ -2 & -4 & -8 & 26 & -2 & -4 & & \\ & & -4 & -1 & 23 & -4 & -4 & -1 \\ & & -2 & -4 & -8 & 23 & -2 & -4 \\ & & & & -8 & -2 & 20 & -4 \\ & & & & -4 & -8 & -8 & 20 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{bmatrix} = \begin{bmatrix} 451 \\ 96 \\ 396 \\ 36 \\ 396 \\ 36 \\ 341 \\ -24 \end{bmatrix}$$

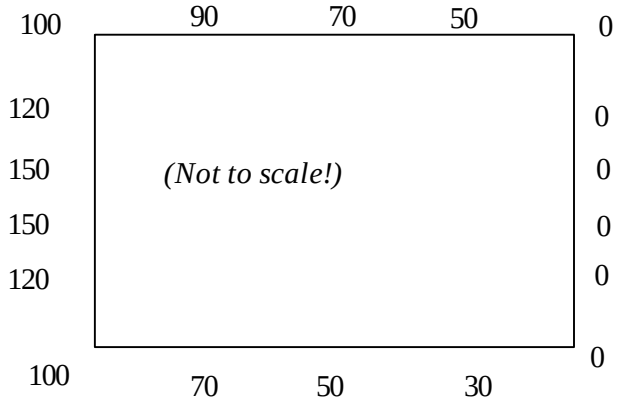
The solution is

$$\begin{aligned} u_1 &= 29.0340 & u_2 &= 22.4585 \\ u_3 &= 31.4853 & u_4 &= 24.6353 \\ u_5 &= 38.3708 & u_6 &= 31.9979 \\ u_7 &= 42.8835 & u_8 &= 36.4267 \end{aligned}$$

Question 5

We have a plate of 12×15 cm and the temperatures on the edges are held as shown in the sketch below. Take $\Delta x = \Delta y = 3$ cm and use the **S.O.R. method** (successive overrelaxation method) to find the temperatures at all the gridpoints. First calculate the optimal value of ω and then use this value in the algorithm. Start with

all grid values equal to the arithmetic average of the given boundary values.



SOLUTION

According to section 7.6 [7.2] of *Gerald,* steady–state heat flow is modelled by Laplace's equation:

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

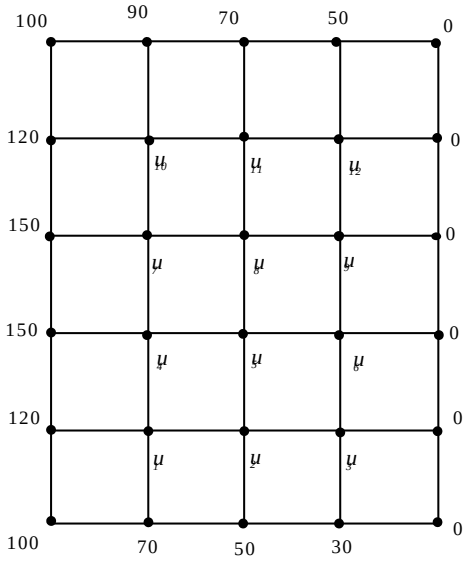
The iteration formula for S.O.R. can be written as

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

(1)

$$= (1-\omega)u_{ij}^{(k)} + \frac{\omega}{4}[u_{i,j-1}^{(k+1)} + u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j+1}^{(k)}]$$

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:

$$
\begin{aligned}
u_1^{(k+1)} &= (1-\omega)u_1^{(k)} + \frac{\omega}{4}[70 + 120 + u_2^{(k)} + u_4^{(k)}] \\
u_2^{(k+1)} &= (1-\omega)u_2^{(k)} + \frac{\omega}{4}[50 + u_1^{(k+1)} + u_3^{(k)} + u_5^{(k)}] \\
u_3^{(k+1)} &= (1-\omega)u_3^{(k)} + \frac{\omega}{4}[30 + u_2^{(k+1)} + 0 + u_6^{(k)}] \\
u_4^{(k+1)} &= (1-\omega)u_4^{(k)} + \frac{\omega}{4}[u_1^{(k+1)} + 150 + u_5^{(k)} + u_7^{(k)}] \\
u_5^{(k+1)} &= (1-\omega)u_5^{(k)} + \frac{\omega}{4}[u_2^{(k+1)} + u_4^{(k+1)} + u_6^{(k)} + u_8^{(k)}] \\
u_6^{(k+1)} &= (1-\omega)u_6^{(k)} + \frac{\omega}{4}[u_3^{(k+1)} + u_5^{(k+1)} + 0 + u_9^{(k)}] \\
u_7^{(k+1)} &= (1-\omega)u_7^{(k)} + \frac{\omega}{4}[u_4^{(k+1)} + 150 + u_8^{(k)} + u_{10}^{(k)}] \\
u_8^{(k+1)} &= (1-\omega)u_8^{(k)} + \frac{\omega}{4}[u_5^{(k+1)} + u_7^{(k+1)} + u_9^{(k)} + u_{11}^{(k)}] \\
u_9^{(k+1)} &= (1-\omega)u_9^{(k)} + \frac{\omega}{4}[u_6^{(k+1)} + u_8^{(k+1)} + 0 + u_{12}^{(k)}] \\
u_{10}^{(k+1)} &= (1-\omega)u_{10}^{(k)} + \frac{\omega}{4}[u_7^{(k+1)} + 120 + u_{11}^{(k)} + 90] \\
u_{11}^{(k+1)} &= (1-\omega)u_{11}^{(k)} + \frac{\omega}{4}[u_8^{(k+1)} + u_{10}^{(k+1)} + u_{12}^{(k)} + 70] \\
u_{12}^{(k+1)} &= (1-\omega)u_{12}^{(k)} + \frac{\omega}{4}[u_9^{(k+1)} + u_{11}^{(k+1)} + 0 + 50]
\end{aligned}
$$

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 copy out 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 ω is given by

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

where p and q are the number of mesh divisions on each side of the rectangular domain. Thus $p = 4$ and $q = 5$, so that $\omega_{opt} \approx 1.2105$. For the sake of interest we verified this by executing the following program

with different values of ω :

ω	iterations / iterasies
1.0	26
1.1	21
1.2	15
1.2105	14
1.3	15
1.4	19
1.5	25
1.6	33
1.7	47
1.8	70
1.9	153

```
PROGRAM AS01_5_2(output);
  CONST
    nx = 3; (* number of nodes in x-direction *)
    ny = 4; (*  ″     ″    ″   ″  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));

    FOR i := 0 to nx + 1 DO
      FOR j := 0 to ny + 1 DO
        u[i,j] := 0.0;
    (* Nonzero boundary values:  *)
    u[1,0] :=  70.0;  u[2,0] :=  50.0;
    u[3,0] :=  30.0;  u[1,5] := 90.0;
    u[2,5] :=  70.0;  u[3,5] :=  50.0;
    u[0,1] := 120.0;  u[0,2] := 150.0;
    u[0,3] := 150.0;  u[0,4] := 120.0;

    (* 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, 'AS01_5_2.DAT');
  rewrite(f);
  Initialize(omega,old_u,u);
  writeln(f);
  writeln(f,'    ********  ASSIGNMENT 5,  ',
          'QUESTION 2  ********');
  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 5,  QUESTION 2  ********

  S.O.R. Method for the Laplace equation:
  omega = 1.2105
  tolerance = 0.000010
  max.  iterations = 100

  Solution and boundary values:
```

```
          90.0000   70.0000   50.0000
120.0000   95.3613   67.7965   38.0886    0.0000
150.0000  103.6489   67.7359   34.5580    0.0000
150.0000  101.4984   64.9402   32.4074    0.0000
120.0000   87.4043   58.1190   30.1316    0.0000
          70.0000   50.0000   30.0000

iterations =  14
max |u - old_u| =   0.000004
```

Question 6

Solve the problem in question 5 by using the **A.D.I method** (alternating-direction-implicit method) without overrelaxation.

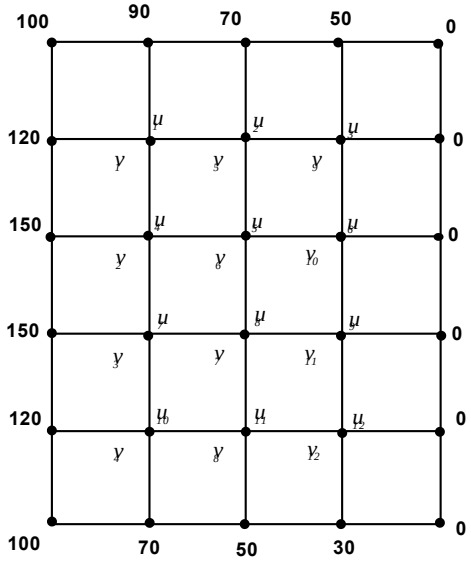
SOLUTION

The A.D.I. method for solving Laplace's equation is discussed in section 7.8 [7.10] of *Gerald*. The iteration formulas are

$$-u_{i-1,\,j}^{(k+1)} + \left(\frac{1}{\rho} + 2\right) 2u_{i,\,j}^{(k+1)} - u_{i+1,\,j}^{(k+1)} \;=\; u_{i,\,j-1}^{(k)} + \left(\frac{1}{\rho} - 2\right) u_{i,\,j}^{(k)} + u_{i,\,j+1}^{(k)} \tag{3}$$

$$-u_{i,\,j-1}^{(k+2)} + \left(\frac{1}{\rho} + 2\right) u_{i,\,j}^{(k+2)} - u_{i,\,j+1}^{(k+2)} \;=\; u_{i-1,\,j}^{(k+1)} + \left(\frac{1}{\rho} + 2\right) u_{i,\,j}^{(k+1)} + u_{i+1,\,j}^{(k+1)} \tag{4}$$

Here $u_{i,\,j}^{(k)}$ denotes the k–th estimate of $u\left(x_i,\, y_j\right)$. Note that the terms in the left hand side of (4) correspond to points in the same column, while the terms in the right hand side of (4) correspond to points in the same row. In (3) it is the other way round. It will be convenient to use u for the temperatures when ordering the points row–wise and v for the temperatures when ordering the points column–wise, as shown below:



Let us choose the acceleration factor $\rho = 1$; then if we apply equation (3) at every mesh point, proceeding row–wise, we obtain

$$\begin{aligned}
(u_1) \quad & -120 + 3u_1 - u_2 = 90 - v_1 + v_2 \\
(u_2) \quad & -u_1 + 3u_2 - u_3 = 70 - v_5 + v_6 \\
(u_3) \quad & -u_2 + 3u_3 - 0 = 50 - v_9 + v_{10} \\
(u_4) \quad & -150 + 3u_4 - u_5 = v_1 - v_2 + v_3 \\
(u_5) \quad & -u_4 + 3u_5 - u_6 = v_5 - v_6 + v_7 \\
(u_6) \quad & -u_5 + 3u_6 - 0 = v_9 - v_{10} + v_{11} \\
(u_7) \quad & -150 + 3u_7 - u_8 = v_2 - v_3 + v_4 \\
(u_8) \quad & -u_7 + 3u_8 - u_9 = v_6 - v_7 + v_8 \\
(u_9) \quad & -u_8 + 3u_9 - 0 = v_{10} - v_{11} + v_{12} \\
(u_{10}) \quad & -120 + 3u_{10} - u_{11} = v_3 - v_4 + 70 \\
(u_{11}) \quad & -u_{10} + 3u_{11} - u_{12} = v_7 - v_8 + 50 \\
(u_{12}) \quad & -u_{11} + 3u_{12} - 0 = v_{11} - v_{12} + 30
\end{aligned}$$

Similarly, by applying equation (4) at each of the interior mesh points, proceeding column–wise. We get

$$\begin{aligned}
(v_1) \quad & -90 + 3v_1 - v_2 = 120 - u_1 + u_2 \\
(v_2) \quad & -v_1 + 3v_2 - v_3 = 150 - u_4 + u_5 \\
(v_3) \quad & -v_2 + 3v_3 - v_4 = 150 - u_7 + u_8 \\
(v_4) \quad & -v_3 + 3v_4 - 70 = 120 - u_{10} + u_{11} \\
(v_5) \quad & -70 + 3v_5 - v_6 = u_1 - u_2 + u_3 \\
(v_6) \quad & -v_5 + 3v_6 - v_7 = u_4 - u_5 + u_6 \\
(v_7) \quad & -v_6 + 3v_7 - v_8 = u_7 - u_8 + u_9 \\
(v_8) \quad & -v_7 + 3v_8 - 50 = u_{10} - u_{11} + u_{12} \\
(v_9) \quad & -50 + 3v_9 - v_{10} = u_2 - u_3 + 0 \\
(v_{10}) \quad & -v_9 + 3v_{10} - v_{11} = u_5 - u_6 + 0 \\
(v_{11}) \quad & v_{10} + 3v_{11} - v_{12} = u_8 - u_9 + 0 \\
(v_{12}) \quad & -v_{11} + 3v_{12} - 30 = u_{11} - u_{12} + 0
\end{aligned}$$

Observe that the boundary values given at the corners of the plate do not appear in any of the equations; in other words the A.D.I. method does not permit us to prescribe these values. This is due to the fact that the iteration formulas are based on the five-point formula.
The algorithm is as follows:

1. Define ε and $\max k$.
2. Let $k = 0$ and define the initial estimate $u_i^{(0)}, \quad i = 1, 2, \ldots, 12$.
   Repeat $3 - 6$ until 7 is satisfied:
3. Let $k := k + 2$.
4. Use $u_1^{(k-2)}, \ldots, u_{12}^{(k-2)}$ to define the right hand sides of equations $(v_1), \ldots, (v_{12})$, then solve for $v_1^{(k-1)}, \ldots, v_{12}^{(k-1)}$.

5. Use $v_1^{(k-1)}, \ldots, v_{12}^{(k-1)}$ to define the right hand sides of equations $(u_1), \ldots (u_{12})$, then solve for $u_1^{(k)}, \ldots, u_{12}^{(k)}$.
6. Print $u_1^{(k)}, \ldots, u_{12}^{(k)}$.
7. $\max_i \left| u_i^{(k)} - u_i^{(k-2)} \right| < \varepsilon$ OR $k \geq \max k$.

The program and results follow. Note that in the program, the values of the function at the boundary are given, and the program will then derive the necessary equations to be solved. The following table shows the effect of varying ρ :

ρ	iterations
0,5	26
0,6	22
0,7	20
0,8	20
0,9	20
1,0	20
1,1	22
1,2	22
1,3	24
1,4	26
1,5	28

```
PROGRAM AS01_5_3(output);
  CONST
    rho = 1.0;
    rows = 4;
    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] := 90.0;  top[2] := 70.0;
  top[3] :=  50.0;
  bottom[1] := 70.0;  bottom[2] := 50.0;
  bottom[3] := 30.0;
  left[1] := 120.0;  left[2] := 150.0;
  left[3] := 150.0;  left[4] := 120.0;
  FOR j := 1 to cols 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,'AS01_5_3.DAT');
rewrite(f);
writeln(f);  writeln(f);
writeln(f,'    ********  ASSIGNMENT 5, ',
        'QUESTION 3  ********');
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:2,' = ',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:2,' = ',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 5, QUESTION 3  ********

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

  Solution:

  u 1 =  95.3613    u 2 =  67.7965    u 3 =  38.0886
  u 4 = 103.6489    u 5 =  67.7359    u 6 =  34.5580
  u 7 = 101.4984    u 8 =  64.9402    u 9 =  32.4074
  u10 =  87.4043    u11 =  58.1190    u12 =  30.1316

  iterations = 20
  max |u - old_u| = 0.000006
```