ANALELE ŞTIINŢIFICE ALE UNIVERSITĂŢII "AL.I. CUZA" DIN IAŞI (S.N.)
MATEMATICĂ, Tomul LIII, 2007, f.1

NUMERICAL ALGORITHMS FOR A SECOND ORDER ELLIPTIC BVP

BY

GINA DURA and RĂZVAN ŞTEFĂNESCU

Abstract. The aim of our paper is to verify by numerical experiments (computer programs) the estimated rate of convergence of some iterative numerical methods. The problem to be solved is a linear 2D second order elliptic BVP. The discretization is made by finite differences and the corresponding algebraic linear system is solved by iterative methods (Jacobi, Gauss-Seidel and SOR with Chebyshev acceleration), which are compared for efficiency.

Mathematics Subject Classification 2000: 65N06, 35J25, 65F10.

Key words: second order elliptic PDE, finite difference methods, iterative methods for linear systems, Matlab programs.

Introduction. Chapter 1 contains a short review on iterative methods for algebraic system. Chapter 2 is concerned with the discretization of the elliptic problem. Computer programs in Matlab, to compare the iterative methods by numerical experiments, are presented in Chapter 3.

1. Iterative methods for algebraic linear system. Let A be a nonsingular $n \times n$ matrix and the system of linear equations

$$Ax = b \ . \tag{1.1}$$

We introduce iterative methods of the form $x^{(i+1)} = \Phi(x^i), i = 0, 1, \ldots$. Using an arbitrary nonsingular $n \times n$ matrix B, we rewrite the equation (1.1) as

$$Bx + (A - B)x = b.$$

The iterative method is given by

$$x^{(i+1)} = (I - B^{-1}A)x^{(i)} + B^{-1}b, \tag{1.2}$$

where I is the corresponding identity matrix. We also introduce the following standard decomposition of A:

$$A = L + D + U,$$

where D is the diagonal part of A, L is the lower triangular part of A with zeros on the diagonal, U is the upper triangular part of A with zeros on the diagonal.

Denote $H = I - B^{-1}A$ and $\tilde{b} = B^{-1}b$. Then (1.2) becomes

$$x^{(i+1)} \;=\; Hx^{(i)} + \tilde{b}. \tag{1.3}$$

The iterative methods to be used in the sequel are: Jacobi, Gauss-Seidel and Successive Over-Relaxation (SOR). For the Jacobi method, $B = D$, and we obtain

$$H_J = -D^{-1}(L + U).$$

We therefore obtain the iteration

$$x_j^{(i+1)} = \left(b_j - \sum_{k \neq j} a_{jk}x_k^{(i)}\right)/a_{jj}, \quad j = 1, 2, \ldots, n. \tag{1.4}$$

For the Gauss-Seidel method, $B = D + L$, and therefore $H_{GS} = -(D + L)^{-1}U$ and we obtain

$$x_j^{(i+1)} = \left(b_j - \sum_{k=1}^{j-1} a_{jk}x_k^{(i+1)} - \sum_{k=j+1}^{n} a_{jk}x_k^{(i)}\right)/a_{jj}, \quad j = 1, 2, \ldots, n. \tag{1.5}$$

To get the relaxation method we rewrite the formula above as

$$x_j^{(i+1)} = x_j^{(i)} + \left(b_j - \sum_{k=1}^{j-1} a_{jk}x_k^{(i+1)} - a_{jj}x_j^{(i)} - \sum_{k=j+1}^{n} a_{jk}x_k^{(i)}\right)/a_{jj}, \;\; j = 1, 2, \ldots, n,$$

and we introduce the relaxation parameter

$$x_j^{(i+1)} \;=\; x_j^{(i)} + \frac{\omega}{a_{jj}}\left(b_j - \sum_{k=1}^{j-1} a_{jk}x_k^{(i+1)} - a_{jj}x_j^{(k)} - \sum_{k=j+1}^{n} a_{jk}x_k^{(i)}\right), \;\; j = 1, 2, \ldots, n,$$

and we obtain the corresponding formula for the relaxation method.

$$x_j^{(i+1)} = (1-\omega)x_j^{(i)} + \frac{\omega}{a_{jj}}\left(b_j - \sum_{k=1}^{j-1} a_{jk}x_k^{(i+1)} - \sum_{k=j+1}^{n} a_{jk}x_k^{(i)}\right), \tag{1.6}$$

$$j = 1, 2, \ldots, n.$$

Next, we multiply (1.6) by a_{jj} and get

$$(D + \omega L)x^{(i+1)} = [(1-\omega)D - \omega U]x^{(i)} + \omega b\ .$$

Comparing with (1.3), we have

$$\begin{aligned} H &= H(\omega) = (D+\omega L)^{-1}[(1-\omega)D - \omega U] \\ &= \left(\frac{1}{\omega}D + L\right)^{-1}\left(\frac{1-\omega}{\omega}D - U\right) = I - \left(\frac{1}{\omega}D + L\right)^{-1} A. \end{aligned}$$

Remark. When $\omega = 1$, the SOR method is actually the Gauss-Seidel method

We recall more results about the convergence of iterative methods

Definition. *We say that the iterative method is convergent if for all initial point $x^{(0)}$ the sequence $\{x^{(i)}\}_{i=0,1,\ldots,}$ converges to the exact solution $x^* = A^{-1}b$.*

We denote by $\rho(C)$the spectral radius of matrix C.

Theorem 1. (a) *The iterative method* (1.3) *converges if and only if* $\rho(H) < 1$.

(b) *A sufficient condition for the convergence of method* (1.3) *is the existence of a natural norm, such that* $\|H\| < 1$.

Theorem 2. (a) *Strong Row Sum Criterion: The Jacobi and Gauss-Seidel method are convergent for any matrix A which satisfy*

$$| a_{ii} | > \sum_{k \neq i} | a_{ik} |, \qquad i = 1, 2, \ldots, n.$$

(b) *Strong Column Sum Criterion: The Jacobi method is convergent for any matrix A which satisfy*

$$| a_{kk} | > \sum_{i \neq k} | a_{ik} | \ , \qquad i = 1, 2, \ldots, n\ .$$

Moreover $\|H_{GS}\| \leq \|H_J\| < 1$.

Theorem 3. (Ostrowski-Reich) *If A is a Hermitian (symmetric) and positive definite matrix and if $0 < \omega < 2$ then the SOR method is convergent.*

For more details about theory of iterative methods, see [3] Chapter 8.

2. Finite difference methods for elliptic equations

2.1 Poisson's equation and its numerical approximation. We consider first Poisson's equation:

$$(P) \qquad \begin{cases} \Delta(x,y) = f(x,y), (x,y) \in \Omega, u(x,y) = 0, (x,y) \in \partial\Omega, \end{cases}$$

where $\Omega = (x_0, x_0 + C) \times (y_0, y_0 + D)$ and $f \in C^2(\bar{\Omega})$.

Under these conditions there exists a unique solution $u \in C^4(\bar{\Omega})$.

We introduce the grid points $M_{i,j}(x_i, y_j)$, where:

$$x_i = x_0 + ih, i = 0, \ldots, N,$$

$$y_j = y_0 + jh, j = 0, \ldots, N,$$

and h is the grid step.

The finite difference method approximates the solution u of problem (P) at the points of the grid. So, we search for a double indexed vector $(u_{i,j})_{i=1,\ldots,N-1;j=1,\ldots,N-1}$, where $u_{i,j} = u(x_i, y_j)$.

From the boundary conditions we get:

$$\begin{cases} u(x_0, y_j) = 0, u(x_0 + C, y_j) & = 0 \\ u(x_i, y_0) = 0, u(x_i, y_j + D) & = 0, \end{cases} \text{for } i = 0, \ldots N, j = 0, \ldots, N.$$

Suppose that $u \in C^4(\bar{\Omega})$. Using a Taylor's series expansion, we obtain the following well-known formulas:

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \frac{u(x_{i+1}, y_j) + u(x_{i-1}, y_j) - 2u(x_i, y_j)}{h^2} + O(h^2),$$

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \frac{u(x_i, y_{j+1}) + u(x_i, y_{j-1}) - 2u(x_i, y_j)}{h^2} + O(h^2).$$

Replacing into the equation above we get:

$$\frac{u(x_{i+1}, y_j) + u(x_{i-1}, y_j) + u(x_i, y_{j+1}) + u(x_i, y_{j-1}) - 4u(x_i, y_j)}{h^2} + O(h^2) = f(x_i, y_j).$$

Thus we obtain the following system of linear equations:

$$\begin{cases} u(x_{i+1},y_j)+u(x_{i-1},y_j)+u(x_i,y_{j+1})+u(x_i,y_{j-1})-4u(x_i,y_j) \\ \qquad\qquad = h^2 f(x_i,y_j)+O(h^2), \\ u(x_0,y_j)=u(x_I,j)=u(x_i,y_0)=u(x_i,y_J)=0, i=0,\dots,N, j=0,\dots,N. \end{cases}$$

For sufficiently small h, we can omit the $O(h^2)$ order terms and then we have:

$$(P_h) \qquad \begin{cases} u_{i,j}=\frac{1}{4}(u_{i+1,j}+u_{i-1,j}+u_{i,j+1}+u_{i,j-1}-h^2 f_{i,j}), \\ i=\overline{1,N-1}, j=\overline{1,N-1}, \\ u_{0,j}=u_{N,j}=u_{i,0}=u_{i,N}=0, i=0,\dots,N, j=0,\dots,N, \end{cases}$$

where $f_{i,j}=f(x_i,y_j)$.

We introduce the iteration sequence and we obtain Jacobi's method

$$\begin{aligned} & u_{i,j}^{(k+1)}=\frac{1}{4}\left(u_{i+1,j}^{(k)}+u_{i-1,j}^{(k)}+u_{i,j+1}^{(k)}+u_{i,j-1}^{(k)}-h^2 f_{i,j}\right), \\ & i=\overline{1,N-1}, j=\overline{1,N-1}, \end{aligned} \tag{2.1}$$

and Gauss-Seidel method

$$\begin{aligned} & u_{i,j}^{(k+1)}=\frac{1}{4}\left(u_{i+1,j}^{(k)}+u_{i-1,j}^{(k+1)}+u_{i,j+1}^{(k)}+u_{i,j-1}^{(k+1)}-h^2 f_{i,j}\right), \\ & i=\overline{1,N-1}, j=\overline{1,N-1}. \end{aligned} \tag{2.2}$$

Thus we have obtained the discrete problem associated with problem (P).

2.2 A general linear second-order elliptic equations. We consider the following problem:

$$\begin{cases} a\dfrac{\partial^2 u}{\partial x^2}(x,y)+b\dfrac{\partial u}{\partial x}(x,y)+c\dfrac{\partial^2 u}{\partial y^2}(x,y)+d\dfrac{\partial u}{\partial y}(x,y)+eu(x,y) \\ \qquad\qquad = f(x,y), (x,y)\in\Omega, \\ u(x,y)=0, (x,y)\in\partial\Omega, \end{cases}$$

where $\Omega=(x_0,x_0+C)\times(y_0,y_0+D)$ and $f\in C^2(\bar{\Omega})$. We consider the same grid of points as above to obtain the discrete problem.

We approximate the equation, in every inner node of the grid, using the numerical derivation formulas

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \frac{u(x_{i+1}, y_j) + u(x_{i-1}, y_j) - 2u(x_i, y_j)}{h^2} + O(h^2),$$
$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \frac{u(x_i, y_{j+1}) + u(x_i, y_{j-1}) - 2u(x_i, y_j)}{h^2} + O(h^2),$$
$$\frac{\partial u}{\partial x}(x_i, y_j) = \frac{u(x_{i+1}, y_j) - u(x_{i-1}, y_j)}{2h} + O(h^2),$$
$$\frac{\partial u}{\partial y}(x_i, y_j) = \frac{u(x_i, y_{j+1}) - u(x_i, y_{j-1})}{2h} + O(h^2).$$

Replacing these expressions into the equation we get

$$(2a + hb)u(x_{i+1}, y_j) + (2a - hb)u(x_{i-1}, y_j) + (2c + hd)u(x_i, y_{j+1})$$
$$+(2c - hd)u(x_i, y_{j-1}) + (2eh^2 - 4a - 4c)u(x_i, y_j) + O(h^2) = 2h^2 f(x_i, y_j),$$
$$\text{for } \; i = 1, \dots N-1, j = 1, \dots, N-1.$$

We cancel the $O(h^2)$ terms and we obtain the corresponding discrete problem

$$\begin{cases} (2a + hb)u_{i+1,j} + (2a - hb)u_{i-1,j} + (2c + hd)u_{i,j+1} \\ +(2c - hd)u_{i,j-1}) + (2eh^2 - 4a - 4c)u_{i,j} = 2h^2 f_{i,j}, \\ i = 1, \dots N-1, j = 1, \dots N-1, \\ u_{0,j} = u_{I,j} = u_{i,0} = u_{i,J} = 0, i = 0, \dots, N, j = 0, \dots, N. \end{cases}$$

Remark. We point out that the method can also be applied for the more general case in which a, b, c, d and e depend on (x, y).

For more detail about finite difference methods see [4].

3. The application of iterative methods. In order to illustrate the application of the iterative methods above, we consider the problem (P), together with the corresponding problem (P_h).

Denoting $u_{ij} = u(x_i, x_j)$ and $f(x_i, y_j) = f_{ij}$ for $i = 0, 1, \dots, N$ and $j = 0, 1, \dots, N$, we rewrite (P_h) as

$$\begin{cases} u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = h^2 f_{i,j}, \\ i = \overline{1, N-1}\ , \ j = \overline{1, N-1}, \\ u_{0,j} = u_{I,j} = u_{i,0} = u_{i,J} = 0, i = \overline{0, N}, j = \overline{0, N}. \end{cases}$$

To write this system of linear equations in matrix form we need to make a vector containing all $u_{i,j}$. Let us number the two dimensions of grid points in a single one-dimensional sequence by defining: $l = i(N+1) + j$ for $i = 0, 1, \ldots, N, j = 0, 1, \ldots, N$. Thus we get

$$\begin{cases} u_l = 0, \text{ for } \{i = 0, i = N \text{ and } j = 0, \ldots, N\}, \\ \{j = 0, j = N \text{ and } i = 0, \ldots, N\}; \\ u_{l+N+1} + u_{l-(N+1)} + u_{l+1} + u_{l-1} - 4u_l = h^2 f_l, \\ \text{for } i = \overline{1, N-1};\ j = \overline{1, N-1}. \end{cases}$$

Now we are able to write our system in matrix form: $Ax = b$, where A is a $N^2 \times N^2$ matrix. The matrix A is sparse. In each row, at most five elements are different from zero. The matrix A almost satisfies the strong row sum criterion (see Theorem 2 in Section 1). There are several lines where we get equality, but the numerical methods work in practice.

The iteration matrix in the Jacobi method has the following form:

$$H_J = -D^{-1}(L+U) = \frac{1}{4}(L+U),$$

ant the k-th step iteration is

$$X^{(k)} = \frac{1}{4}X^{(k-1)} - \frac{1}{4}b$$

The eigenvalues of H_J can be determined explicitly, and the spectral radius of H_J is

$$\rho(H_J) = \cos\frac{\pi}{N},$$

(see [2], Chapter 17). The number k of iterations required to reduce the overall error by a factor 10^{-p} (we want to obtain $\|X^* - X^{(k)}\| \leq 10^{-p}\|X^* - X^{(0)}\|$, where X^* represents the exact solution and $X^{(0)}$ is the initial vector) is thus estimated by:

$$k \approx \frac{p\ \ln 10}{-\ln\ \rho(H_J)}.$$

For large N the spectral radius becomes

$$\rho(H_J) \simeq 1 - \frac{\pi^2}{2N^2}. \tag{3.1}$$

The number of iterations k required to reduce the overall error by a factor 10^{-p} is thus

$$k \simeq \frac{2pN^2 \ln 10}{\pi^2} \simeq \frac{1}{2} pN^2 \ .$$

In other words, the number of iteration is proportional to N^2 (the number of mesh points). It is clear that the Jacobi method is not useful in practice.

The iteration matrix in the Gauss-Seidel method has the form

$$H_{GS} = -(L + D)^{-1} U,$$

ant the $k - th$ step iteration is

$$(L + D)X^{(k)} = -UX^{(k-1)} + b.$$

The eigenvalues of H_{GS} can be determined explicitly and the spectral radius of H_{GS} is:

$$\rho(H_{GS}) = \cos^2 \frac{\pi}{N},$$

(see [2], Chapter 17). For large N the spectral radius is

$$\rho(H_{GS}) \simeq 1 - \frac{\pi^2}{N^2}.$$

The number of iteration k required to reduce the overall error by a factor 10^{-p} is thus

$$k \simeq \frac{pN^2 \ln 10}{\pi^2} \simeq \frac{1}{4} pN^2.$$

This shows that the Gauss-Seidel method is faster than the Jacobi method.

The Simultaneous Over-Relaxation method derives from the Gauss-Seidel method, where we are introducing a relaxation parameter ω. It has been proved that the method is convergent only for $\omega \in (0, 2)$ (see Theorem Ostrowski-Reich) and for $\omega \in (1, 2)$ (over-relaxation case) the convergence can be faster then Gauss-Seidel method. The optimal choice for ω is given by:

$$\omega = \frac{2}{1 + \sqrt{1 - \rho(H_J)^2}},$$

(see [2], Chapter 17).

For this optimal choice, the spectral radius for SOR is

$$\rho_{SOR} = \left(\frac{\rho(H_J)}{1 + \sqrt{1 - \rho(H_J)^2}} \right)^2.$$

Replacing $\rho(H_J)$ from (3.1) we obtain $\omega \simeq 2/(1+\frac{\pi}{N})$, and for large N we have: $\rho(H_{SOR}) \simeq 1 - \frac{2\pi}{N}$. The number of iterations required to reduce the initial error by a factor of 10^{-p} is therefore

$$k \simeq \frac{pN \ln 10}{2\pi} \simeq \frac{1}{3} pN.$$

Comparing this results, we see that optimal SOR requires only O(N) iterations instead of $O(N^2)$ iterations in case of Jacobi and Gauss-Seidel methods.

In the sequel, we describe numerical algorithms corresponding to the iterative methods presented before and numerical results that we have obtained after implementation with Matlab.

In the Jacobi's method (see (2.1)), the algorithm consists of using the average of u at its four nearest neighbor points of the grid (plus the contribution from the boundary conditions). Thus we need only two vectors to stock the information at every step and the stopping criterion is

$$\|u^{(k)} - u^{(k-1)}\| < 10^{-p}.$$

In Gauss-Seidel's method, we make use of updated values on the right side of equation (see (2.2)) as soon as they become available. Thus, the averaging is done in place, instead of being copied from an earlier step to a later one. Similarly with Jacobi's method, we need two vectors to save the intermediate values of u and the stopping criterion is the same.

Next, we consider a general second order elliptic equation:

$$au_{i+1,j} + bu_{i-1,j} + cu_{i,j+1} + du_{i,j-1} + eu_{i,j} = f_{i,j}. \tag{3.2}$$

For problem (P), we have $a = b = c = d = \frac{1}{h^2}, e = -\frac{4}{h^2}$.

To get a practical implementation of SOR algorithm we need explicit formulas. First we solve equation (3.2) for $u_{i,j}$: $u^*_{i,j} = \frac{1}{e}(f_{i,j} - au_{i+1,j} - bu_{i-1,j} - cu_{i,j+1} - du_{i,j-1})$. Then, we introduce ω and obtain

$$u^{new}_{i,j} = \omega u^*_{i,j} + (1-\omega) u^{old}_{i,j}.$$

Denote $\xi_{i,j} = au_{i+1,j} + bu_{i-1,j} + cu_{i,j+1} + du_{i,j-1} + eu_{i,j} - f_{i,j}$ (the residual vector at any stage), and the components are *old* or *new* depending on the nods numbering. So, we get a practical algorithm for SOR

$$u^{new}_{i,j} = u^{old}_{i,j} - \omega \frac{\xi_{i,j}}{e}.$$

The norm of the residual vector $\xi_{i,j}$ can be used for stopping the algorithm.

Another problem is the order in which mesh points are processed. A simply strategy is to proceed in order down the rows. Alternatively, suppose we divide the mesh into "odd" and "even" points, like the black and white squares of chessboard(by "odd" and "even" we understand the sum of the indices of $u_{i,j}$). Then equation (3.2) shows that the odd points depend only on the even ones and inversely. Accordingly, we can carry out one half updating of the odd points, and then another half updating of the even points. We used odd-even ordering for the version of SOR implemented below.

In SOR with Chebyshev acceleration (using odd-even ordering), we change ω at each half-sweep step according to the following formulas:

$$\begin{aligned}
\omega^{(0)} &= 1, \\
\omega^{\frac{1}{2}} &= 1/\left(1 - \frac{\rho^2_{Jacobi}}{2}\right), \\
\omega^{(k+\frac{1}{2})} &= 1/\left(1 - \frac{\rho^2_{Jacobi}\omega^{(k)}}{4}\right), \quad k = \frac{1}{2}, 1, \ldots, \\
\lim_{k\to\infty} \omega^{(k)} &= \omega_{optimal}.
\end{aligned}$$

The beauty of Chebysev acceleration is that the norm of the residual error always decreases with each iteration.

In the sequel, we give a routine for Jacobi, Gauss-Seidel, and SOR with Chebyshev acceleration and odd-even ordering for problem (P). Here $\bar{\Omega} = [x_0, x_0 + q] \times [y_0, y_0 + q]$ is a square.

The main part is:

```
clear;
global x0 x1 y0 y1 h2;
global x y;
global N iter;
global u unew  uold;
N=input('N: ');
x0=input('x0: ');
y0=input('y0: ');
q=input('length: ');
tol=input('precision: ');
maxit=input('maxiter: ');
```

```
disp('1=Jacobi, 2=Gauss-Seidel, 3=SOR');
disp('What do you want to choose: ');
M=input(' ');
x1=x0+q;
y1=y0+q;
h=q/(N-1);
h2=h^2;
for j =1:N
    temp =(j-1)* h;
    x(j)   =x0+temp;
    y(j)   =y0+temp;
end
a=1.0;
b=1.0;
c=1.0;
d=1.0;
e=-4.0;
rjac=1.0-0.5*(pi/N)^2;
ro=rjac^2;
u=zeros(N,N) ;
uold=zeros(N,N) ;
unew=uold ;
if M==3
     kod=relax(a,b,c,d,e,ro,maxit,tol);
     if kod~ =0
          error('Sor failed');
     end
end
if M==1
     Jacobi(a,b,c,d,e,ro,maxit,tol);
     disp('Solution obtained');
end
if M==2
     Gauss-Seidel(a,b,c,d,e,ro,maxit,tol);
     disp('Solution obtained');
end
mesh(u)
u
```

iter

The Jacobi method's function is:

```
function Jacobi(a,b,c,d,e,ro,maxit,tol)
global x y
global N iter
global u unew uold
flag=0 ;
iter=0 ;
while ~ flag
      iter =iter+1;
      for i=2:N-1
          for j=2:N-1
              unew(i,j)=(a*uold(i-1,j)+b*uold(i+1,j)+c*uold(i,j-1)+
              +d*uold(i,j+1))/4.0-Fbvp(x(i),x(j));
          end
      end
      if all (abs(unew-uold) <= tol)
          flag=1 ;
          u=unew ;
      end
      if  (iter>maxit)
          disp('Jacobi failure');
          break
      end
      uold=unew ;
end
```

The Gauss-Seidel method's function is:

```
function Gauss-Seidel1(a,b,c,d,e,ro,maxit,tol)
global x y
global N iter
global u unew uold
flag=0;
iter=0;
while ~ flag
     for i=1:N
```

```
        unew(i,1)=0;
        unew(i,N)=0;
    end
    for j=1:N
        unew(1,j)=0;
        unew(N,j)=0;
    end
    iter=iter+1;
    for i=2:N-1
        for j=2:N-1
            unew(i,j)=(unew(i-1,j)+uold(i+1,j)+unew(i,j-1)
            +uold(i,j+1))/4.0-Fbvp(x(i),y(j));
        end
    end
    if all(abs(unew-uold)<=tol)
      flag=1;
      u=unew;
    end
    if (iter>maxit)
      error('Gauss Seidel failure');
    end
    uold=unew;
end
```

The SOR method's function is :

```
function koderr=relax(a,b,c,d,e,ro,maxit,tol)
global x y
global N
global u iter
koderr=1;
anormf=0.0;
for j=2:N-1
   for l=2:N-1
       anormf=anormf+abs(Fbvp(x(j),y(l)));
   end
end
omg=1.0;
for i=1:maxit
```

```
    anorm=0.0;
    for j=2:N-1
        for l=2:N-1
            if mod(j+l,2) ==mod(i,2)
                resid=a*u(j+1,l)+b*u(j-1,l)+c*u(j,l+1)+d*u(j,l-1)+
                e*u(j,l)-Fbvp(x(j),y(l));
                anorm=anorm+abs(resid);
                u(j,l)=u(j,l)-omg*resid/e;
            end
        end
    end
    if i==1
       omg=1.0/(1.0-ro/2.0);
    else
        omg=1.0/(1.0-ro*omg/4.0);
    end
    iter=i;
    if (i>1)&(anorm<(tol*anormf))
        koderr=0;
        break
    end
end
```

Finally, for the Fbvp function we have:

```
function z=Fbvp(a,b)
global x0 x1 y0 y1 h2
temp=(a-x0)*(x1-a)*(b-y0)*(y1-b);
z=h2*temp/4.0;
```

The graphics of the numerical solution is given in Figure 1.

The summary of our experiments is the following one:

Jacobi	$O\left(\frac{1}{2}pN^2\right)$	1172
Gauss-Seidel	$O\left(\frac{1}{4}pN^2\right)$	694
SOR	$O\left(\frac{1}{3}pN\right)$	98

The first column gives the mathematically estimated number of iterations, while the second one the number of iterations needed by the computer program.

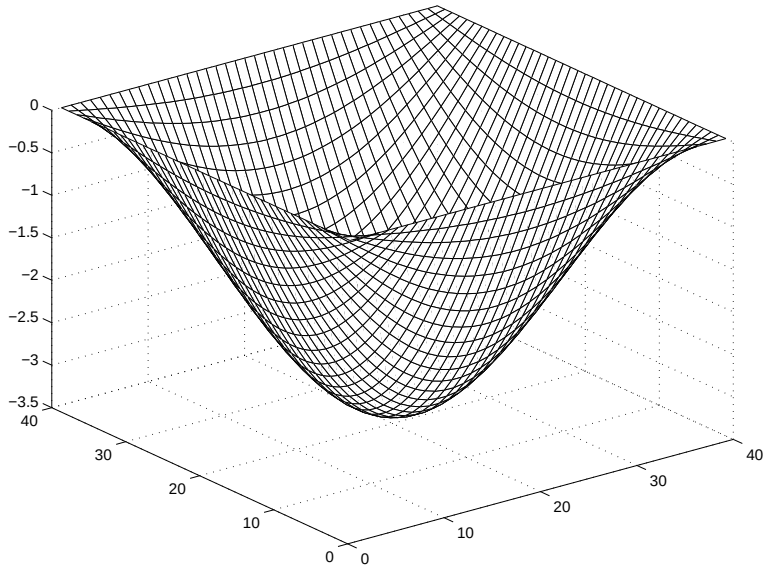

Figure 1

Recall that N is the number of subintervals for each edge of the square $\bar{\Omega}$ and p (tol in the program, actually $tol = 10^{-p}$) is the precision for convergence. In our experiments $\bar{\Omega} = [0, 4] \times [0, 4]$, N=40 and $p = 3$.

We may conclude that SOR is much faster also practically, confirming the mathematical evaluation.

We also refer to [1], where numerical results for a problem similar to (P) are presented. There the domain is an ellipse and the discretization is made by FEM.

REFERENCES

1. Arnautu, V.; Langmach, H.; Sprekels, J.; Tiba, D. – *On the approximation and the optimization of plates,* Numerical Functional Analysis and Optimization, vol. 21, nr. 3-4, 2000, pp. 337-354.

2. Press et al., W.H. – *Numerical Recipes in C,* Cambridge University Press, New York, 1990.

3. Stoer, J.; Bulirsch, R. – *Introduction to Numerical Analysis,* Springer, New York, 1980.

4. Thomas, J.W. – *Numerical Partial Differential Equations,* Springer, New York, 1995.

Received: 16.II.2007

"Al.I. Cuza" University,
Faculty of Mathematics,
Bd. Carol I, no 11, Iaşi,
ROMÂNIA
str8zv@yahoo.com