

## Метод SOR (продолжение)

Рассмотрим эллиптическое уравнение второго порядка относительно двух переменных  $x$  и  $y$  и напишем разностную схему для квадратной области. Каждой строке матрицы  $A$  в матричном уравнении  $Ax=b$  соответствует выражение

$$a_{j,l}u_{j+1,l} + b_{j,l}u_{j-1,l} + c_{j,l}u_{j,l+1} + d_{j,l}u_{j,l-1} + e_{j,l}u_{j,l} = f_{j,l}$$

Для рассматриваемой нами задачи  $a = b = c = d = 1$ ,  $e = 4$ .

Итерационную формулу можно написать в виде

$$u_{j,l}^* = \frac{1}{e_{j,l}} (f_{j,l} - a_{j,l}u_{j+1,l} - b_{j,l}u_{j-1,l} - c_{j,l}u_{j,l+1} - d_{j,l}u_{j,l-1})$$

Средневзвешенное значение имеет вид

$$u_{j,l}^{\text{new}} = \omega u_{j,l}^* + (1 - \omega)u_{j,l}^{\text{old}}$$

Невязку для нашей задачи можно записать:

$$\xi_{j,l} = a_{j,l}u_{j+1,l} + b_{j,l}u_{j-1,l} + c_{j,l}u_{j,l+1} + d_{j,l}u_{j,l-1} + e_{j,l}u_{j,l} - f_{j,l}$$

Используя формулу метода SOR (см. предыдущую лекцию) можно записать

$$u_{j,l}^{\text{new}} = u_{j,l}^{\text{old}} - \omega \frac{\xi_{j,l}}{e_{j,l}}$$

Эту формулу можно очень легко запрограммировать и норма невязки может служить критерием окончания итерационного процесса.

Узлы сетки можно разбить на четные и нечетные, при этом окажется, что значения в четных узлах зависят только от значений в нечетных узлах и наоборот. В программе, приведенной ниже, такое разбиение используется.

Асимптотическая оценка скорости сходимости метода SOR может служить лишь для оценки порядка количества итераций; в реальности количество итераций может быть в 20 раз больше асимптотической оценки.

В методе SOR с ускорением Чебышева используется динамическое вычисление коэффициента релаксации

$$\omega^{(0)} = 1$$

$$\omega^{(1/2)} = 1/(1 - \rho_{\text{Jacobi}}^2/2)$$

$$\omega^{(n+1/2)} = 1/(1 - \rho_{\text{Jacobi}}^2 \omega^{(n)}/4), \quad n = 1/2, 1, \dots, \infty$$

$$\omega^{(\infty)} \rightarrow \omega_{\text{optimal}}$$

Важным достоинством этой реализации метода SOR является уменьшение нормы ошибки на каждой итерации (имеется в виду ошибка при вычислении решения)

Текст программы на языке C приведен ниже

```

#include <math.h>
#define MAXITS 1000
#define EPS 1.0e-5

void sor(double **a, double **b, double **c, double **d, double **e,
         double **f, double **u, int jmax, double rjac)
{
    void nrerror(char error_text[]);
    int ipass,j,jsw,l,lsw,n;
    double anorm,anormf=0.0,omega=1.0,resid;
    Double precision is a good idea for jmax bigger than about 25.

    for (j=2;j<jmax;j++)
    Compute initial norm of residual and terminate iteration when norm
    a factor EPS.
        for (l=2;l<jmax;l++)
            anormf += fabs(f[j][l]);           Assumes initial u is
    for (n=1;n<=MAXITS;n++) {
        anorm=0.0;
        jsw=1;
        for (ipass=1;ipass<=2;ipass++) {      Odd-even ordering.
            lsw=jsw;
            for (j=2;j<jmax;j++) {
                for (l=lsw+1;l<jmax;l+=2) {
                    resid=a[j][l]*u[j+1][l]
                        +b[j][l]*u[j-1][l]
                        +c[j][l]*u[j][l+1]
                        +d[j][l]*u[j][l-1]
                        +e[j][l]*u[j][l]
                        -f[j][l];
                    anorm += fabs(resid);
                    u[j][l] -= omega*resid/e[j][l];
                }
                lsw=3-lsw;
            }
            jsw=3-jsw;
            omega=(n == 1 && ipass == 1 ? 1.0/(1.0-0.5*rjac*rjac)
                1.0/(1.0-0.25*rjac*rjac*omega));
        }
        if (anorm < EPS*anormf) return;
    }
    nrerror("MAXITS exceeded");
}

```

Главным достоинством метода является простота использования, а главным недостатком – недостаточно быстрая сходимость при решении больших задач.