Центр компьютерной физики
Кафедра общей физики и волновых процессов
Лаборатория Инженерной Физики

Параллельное программирование
для ресурсоёмких задач численного моделирования в физике

В.О. Милицин, Д.Н. Янышев, И.А. Буткарев

Решение краевоей задачи методом Якоби с использованием технологии MPI

Физический факультет МГУ им М.В. Ломоносова

Уравнение Лапласа

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad x, y \in [0,1] \tag{1}$$

Граничные условия:

$$\begin{aligned} u(x,0) &= \sin(\pi x) && 0 \le x \le 1 \\ u(x,1) &= \sin(\pi x) e^{-x} && 0 \le x \le 1 \\ u(0,y) &= u(1,y) = 0 && 0 \le y \le 1 \end{aligned} \tag{2}$$

Аналитическое решение:

$$u(x,y) = ?? \qquad x, y \in [0,1] \tag{3}$$

Численное решение

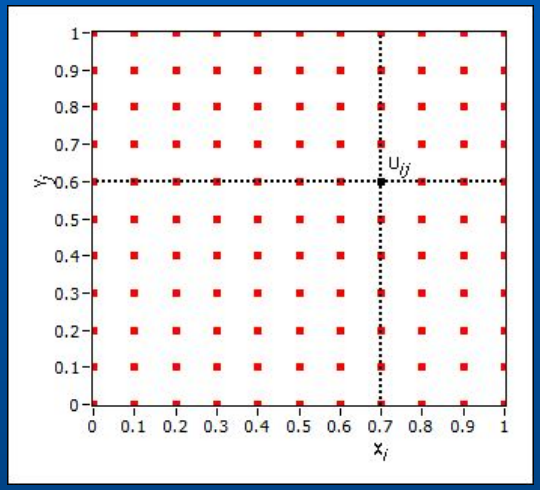
Применим итерационный метод Якоби

$$E\frac{u^{n+1}-u^{n}}{\tau}=\Lambda_x u^{n}+\Lambda_y u^{n}, \qquad (4)$$

где

$$\tau=\left(\frac{2}{\Delta x^2}+\frac{2}{\Delta y^2}\right)^{-1}, \quad \Lambda_x u^{n}=\frac{u^{n}_{i+1,j}-2u^{n}_{i,j}+u^{n}_{i-1,j}}{\Delta x^2},$$

$$u^{n}_{i,j}=u^{n}\left(x_i,y_j\right)$$



Для простоты мы возьмем $\Delta x = \Delta y = \frac{1}{m+1},$

тогда преобразовывая уравнение (4) получим

$$u_{i,j}^{n+1} = \frac{u_{i+1,j}^{n} + u_{i-1,j}^{n} + u_{i,j+1}^{n} + u_{i,j-1}^{n}}{4} \qquad i = 1,2,\ldots,m; \quad j = 1,2,\ldots,m \qquad (5)$$

где n и n+1 означают текущий и последующий шаг, соответственно, для

$$u_{i,j}^{n} = u^{n}(x_i, y_j) \qquad i = 0,1,2,\ldots,m+1; \quad j = 0,1,2,\ldots,m+1$$
$$= u^{n}(i\Delta x, j\Delta y) \qquad (6)$$

Последовательная программа

новое значение в каждой точке сетки равно среднему из предыдущих значений четырех ее соседних точек (слева, справа, сверху и снизу). Этот процесс повторяется, пока вычисление не завершится.

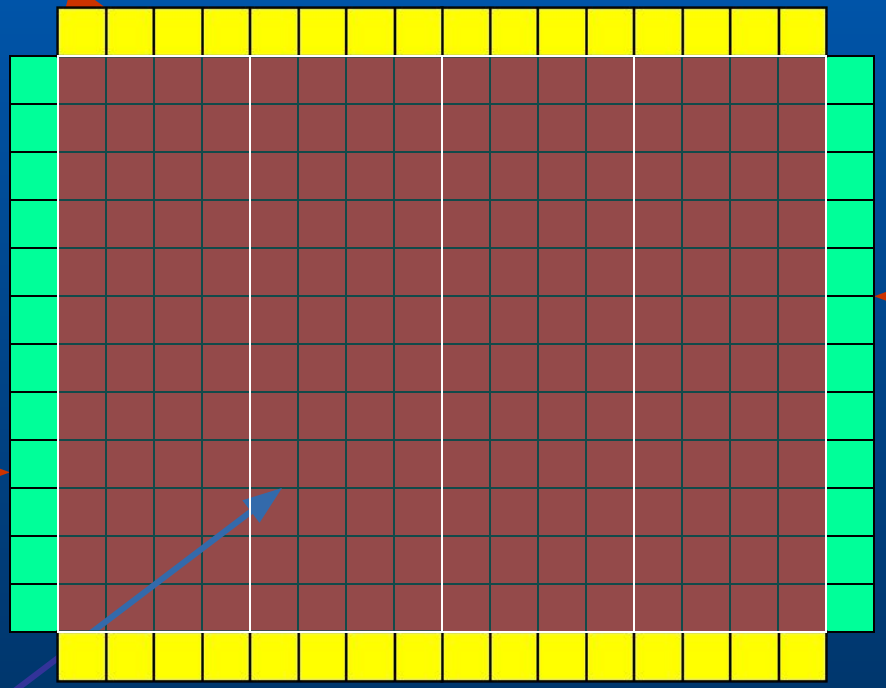
```
while (true) {
  # вычислить новые значения во всех внутренних точках
  for [i = 1 to n, j = 1 to n]
    new[i,j] = (grid[i-1,j] + grid[i+1,j] +
                grid[i,j-1] + grid[i,j+1] / 4;
  iters++;
  # вычислить максимальную разность
  maxdiff = 0.0;
  for [i = 1 to n, j = 1 to n]
    maxdiff = max(maxdiff, abs(new[i,j]-grid[i,j]));
  # проверить условие завершения вычислений
  if (maxdiff < EPSILON)
    break;
  # скопировать new в grid, чтобы подготовить следующие обновления
  for [i = 1 to n, j = 1 to n]
    grid[i,j] = new[i,j];
}
```

Оптимизированный вариант

```
for [ iter = 1 to MAXITERS] {
  #вычислить новые значения во всех внутренних точках
  for [ i = 1 to n, j = 1 to n]
    new[i,j] = (grid[i-1,j] + grid[i+1,j] +
            grid[i,j-1] + grid[i,j+1] - h2*f[i,j])*0.25;
  #скопировать new в grid, чтобы подготовить следующие
    обновления
  for [ i = 1 to n, j = 1 to n]
    grid[i,j] = new[i,j];
}
# вычислить максимальную разность, или среднеквадратичное
    отклонение
maxdiff = 0.0;
for [ i = 1 to n, j = 1 to n]
  maxdiff = max(maxdiff, abs (new[i,j] - grid[i,j]));
```

Вычислительная сетка

$u(x,1) = \sin(\pi x)e^{-x}$



y, j

x, i

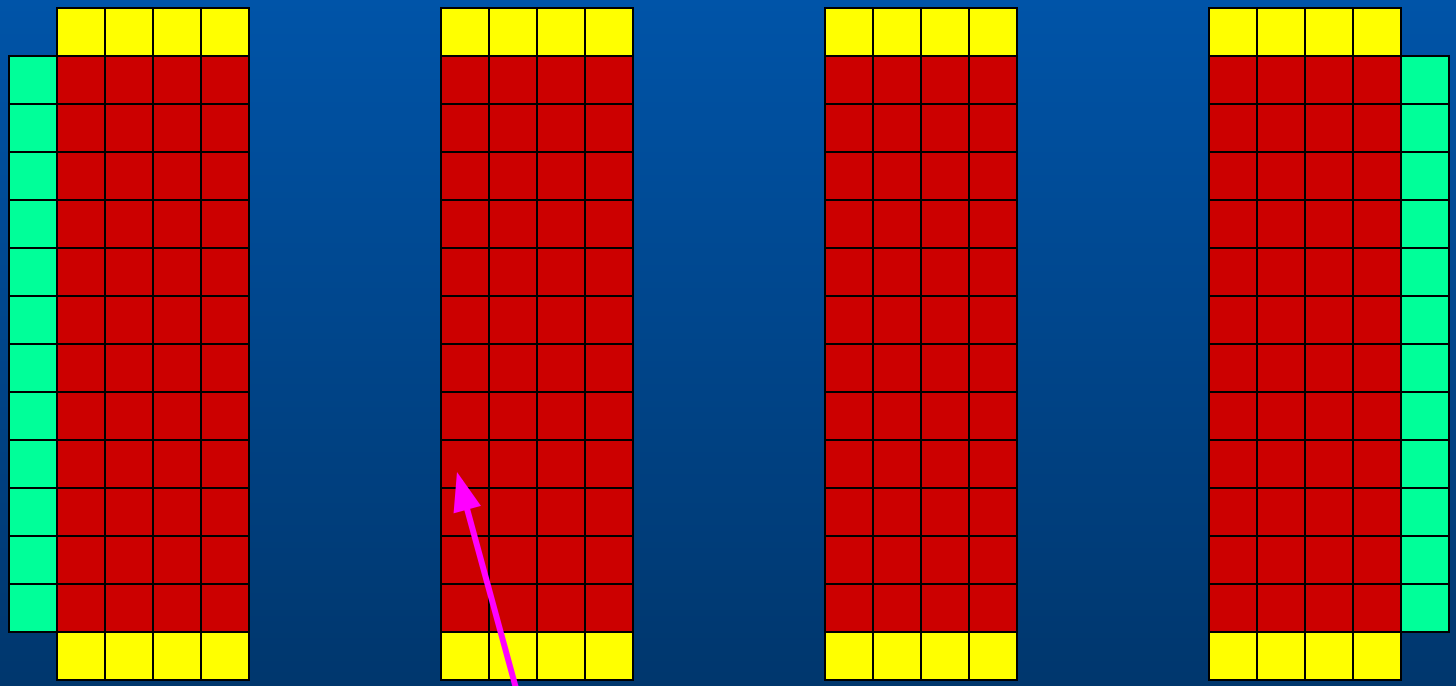
$u(1, y) = 0$

$u(0, y) = 0$

$u(x,0) = \sin(\pi x)$

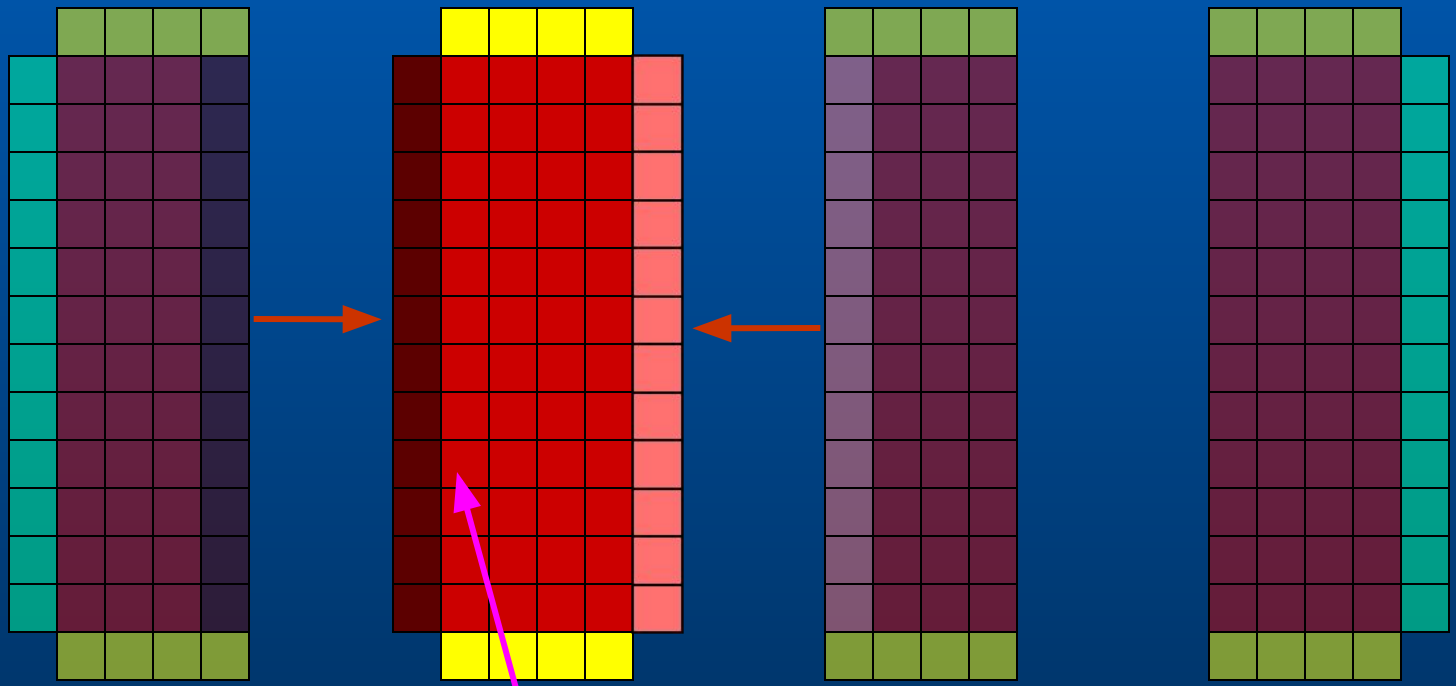
$$u_{i,j}^{n+1} \cong \frac{u_{i+1,j}^{n} + u_{i-1,j}^{n} + u_{i,j+1}^{n} + u_{i,j-1}^{n}}{4} \qquad i = 1,2,\ldots,m; \quad j = 1,2,\ldots,m$$

Вычислительная сетка II



$$u_{i,j}^{n+1} \cong \frac{u_{i+1,j}^{n} + u_{i-1,j}^{n} + u_{i,j+1}^{n} + u_{i,j-1}^{n}}{4} \qquad i = 1,2,\ldots,m; \quad j = 1,2,\ldots,m$$

Вычислительная сетка III



$$u_{i,j}^{n+1} \cong \frac{u_{i+1,j}^{n} + u_{i-1,j}^{n} + u_{i,j+1}^{n} + u_{i,j-1}^{n}}{4} \qquad i = 1,2,\ldots,m; \quad j = 1,2,\ldots,m$$

Пример программы

```
PROGRAM JACOBI
IMPLICIT NONE
INCLUDE 'mpif.h'
INTEGER n,m,npmin,nps,itmax
PARAMETER (n = 400, npmin=1, nps=n/npmin+1, itmax = 1000)

!массивы
REAL*8 A(0:n+1,0:nps+1), B(n,nps)
REAL*8 diffnorm, gdiffnorm, eps, time

!индексы
INTEGER left, right, i, j, k, itcnt, status(MPI_STATUS_SIZE ), tag
INTEGER IAM, NPROCS, ierr
LOGICAL converged

!start MPI
CALL MPI_INIT(IERR)
CALL MPI_COMM_SIZE(MPI_COMM_WORLD, NPROCS, ierr)
CALL MPI_COMM_RANK(MPI_COMM_WORLD, IAM, ierr)
```

Начальные и граничные условия

```
eps = 0.01

!распределение сетки
m = n/NPROCS
IF (IAM.LT.(n-NPROCS*m)) THEN
      m = m+1
END IF

!начальное состояние
time = MPI_Wtime()
do j = 0,m+1
  do i = 0,n+1
     A(i,j) = 0.0
  end do
end do

!граничные условие верх/низ
do j = 0,m+1
      A(0,j) = sin(3.14*(j+m*IAM)/n)
      A(n+1,j) = sin(3.14*(j+m*IAM)/n) *exp(-(j+m*IAM)/n)
end do
```

Начальные и граничные условия (продолжение)

```
!граница слева
IF(IAM.EQ.0) then
     do i = 0,n+1
       A(i,0) = 0.
     end do
end if

!граница справа
IF(IAM.EQ.NPROCS-1) then
     do i = 0,n+1
       A(i,m+1) = 0.
     end do
end if
```

Определение номеров процессов

```
!левый сосед
IF (IAM.EQ.0) THEN
      left = MPI_PROC_NULL
ELSE
      left = IAM - 1
END IF

!правый сосед
IF (IAM.EQ.NPROCS-1) THEN
      right = MPI_PROC_NULL
ELSE
      right = IAM+1
END IF


tag = 100
itcnt = 0
converged = .FALSE.
```

Вычисление новых значений функций по 5-точечной схеме

```
DO k = 1,itmax

      diffnorm = 0.0
      itcnt = itcnt + 1

      DO j = 1, m
         DO i = 1, n
            B(i,j)=0.25*(A(i-1,j)+A(i+1,j)+A(i,j-1)+A(i,j+1))
            diffnorm = diffnorm + (B(i,j)-A(i,j))**2
         END DO
      END DO

      DO j = 1, m
         DO i = 1, n
            A(i,j) = B(i,j)
         END DO
      END DO
```

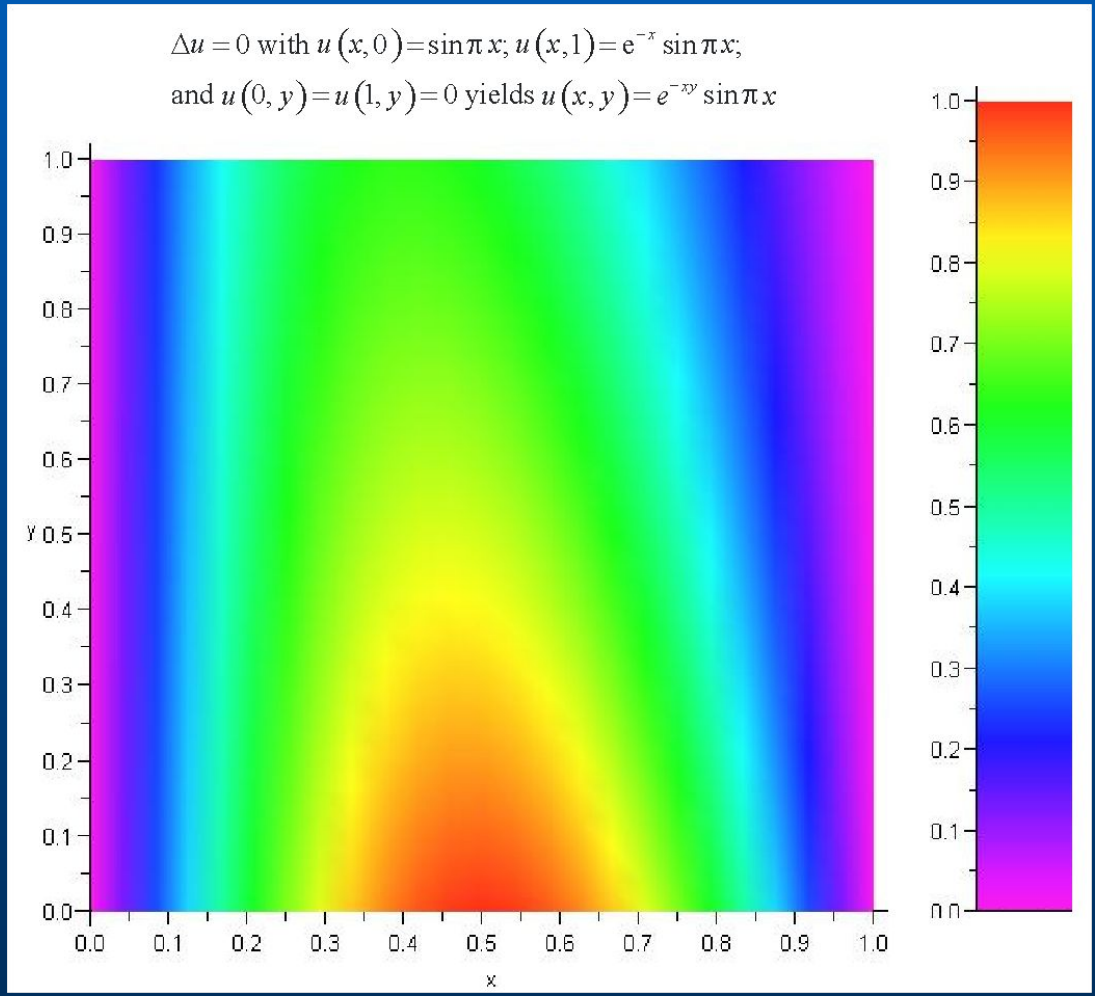
Пересылка граничных значений в соседние процессы

```
!справа налево
CALL MPI_SENDRECV( B(1,1), n, MPI_DOUBLE_PRECISION, left, tag,
     A(1,0), n, MPI_DOUBLE_PRECISION, left, tag,
     MPI_COMM_WORLD, status, ierr)

!слева направо
CALL MPI_SENDRECV( B(1,m), n, MPI_DOUBLE_PRECISION, right, tag,
     A(1,m+1), n, MPI_DOUBLE_PRECISION, right, tag,
     MPI_COMM_WORLD, status, ierr)

!условие выхода
CALL MPI_Allreduce( diffnorm, gdiffnorm, 1, MPI_DOUBLE_PRECISION,
     MPI_SUM, MPI_COMM_WORLD, ierr )

gdiffnorm = sqrt( gdiffnorm)
converged = gdiffnorm.LT.eps
if(converged) goto 777


END DO


777 continue
```

Завершение программы

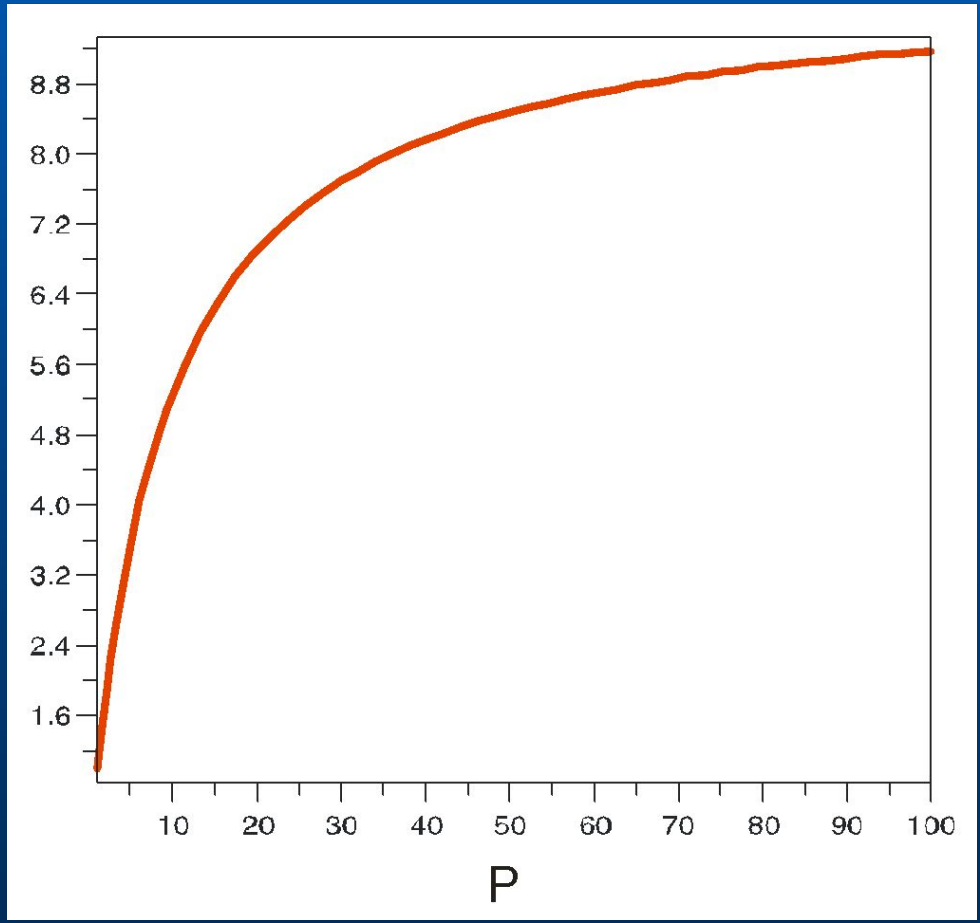
```
!время
time = MPI_Wtime() - time

!вывод результата
IF(IAM.EQ.0) then
      WRITE(*,*) ' At iteration ', itcnt, 'a diff is ', gdiffnorm
      WRITE(*,*) ' Time calculation: ', time
END IF

!завершение MPI
CALL MPI_Finalize(ierr)
stop

END PROGRAM
```

Решение уравнения Лапласа



Эффективность распараллеленных алгоритмов

$$s \le \frac{1}{d + \frac{1-d}{P}}$$



Пример

Рассмотрим задачу решения уравнения Лапласа (1)-(2) с измененными граничными условиями:

$$u(x,0) = a \sin \pi x, \quad 0 \le x \le 1$$

$$u(x,1) = b\, \mathrm{e}^{-x} \sin \pi x, \quad 0 \le x \le 1$$

Тогда можно поставить задачу синтеза таких параметров a и b, которые обеспечивают заданную температуру в некоторой точке (x,y) внутри области.

$$F[u] = \left(u(x,y) - u_0\right)^2$$