Компьютерные технологии в математическом моделировании

Ассистент кафедры
математической физики
к.ф.-м.н.
Татьяна Евгеньевна Романенко

Содержание лекции

- Определение задач для УЧП в терминах вариационных задач
- Задание простых областей
- УЧП, зависящие от времени
- Граничные условия

Установка FEniCS через Docker

1. Установка Docker . https://www.docker.com
2. Terminal> curl -s https://get.fenicsproject.org | bash
3. fenicsproject run
4. docker run --rm -ti -v ‘pwd‘:/home/fenics/shared -w /home/fenics/shared quay.io/fenicsproject/stable:current ’/bin/bash -l -c "export TERM=xterm; bash -i"’

Установка FEniCS через пакеты Ubuntu

1. sudo add-apt-repository ppa:fenics-packages/fenics
2. sudo apt-get update
3. sudo apt-get install fenics
4. sudo apt-get dist-upgrade

Проверка установки:
Terminal> python -c ’import fenics’

Общий план решения

1. Определить расчетную область, уравнение, граничные условия
2. Переформулировать задачу для УЧП как конечно-элементную вариационную задачу
3. Определить конечно-элементную вариационную задачу средствами FEniCS
4. Решение вариационной задачи с помощью FEniCS, визуализация данных, вычисление погрешности

Переход к вариационной задаче для уравнения Пуассона

1. Уравнение Пуассона

$$\begin{cases} -\Delta u(x) = f(x), & (x) \in \Omega \\ u(x) = u_D(x), & (x) \in \partial\Omega \end{cases} \qquad (1)$$

2. Переходим к вариационной задаче:
   1. Домножение на тестовую функцию v:

      $$-\int_\Omega -\Delta u(x) v(x) dx = -\int_\Omega \left(\nabla^2 u(x)\right) v(x) dx = \int_\Omega f(x) v(x) dx \ (2)$$

   2. Интегрируем по частям:

      $$-\int_\Omega \left(\nabla^2 u(x)\right) v(x) dx = \int_\Omega \nabla u(x) \nabla v(x) dx - \int_{\partial\Omega} \frac{\partial u}{\partial n} \ v \, ds \ (3)$$

   3. Обнуление тестовой функции на границе $\partial\Omega$:

      $$\int_\Omega \nabla u(x) \nabla v(x) dx = \int_\Omega f(x) v(x) dx \qquad (4)$$

Переход к вариационной задаче для уравнения Пуассона

4. При выполнении (4) для всех тестовых функций v из некоторого $\hat{V}$, получаем вариационную форму задачи (1) для пробных функций u из пространства пробных функций V:

$$\int_\Omega \nabla u \nabla v dx = \int_\Omega f v dx \qquad \forall v \in \hat{V}.$$

5. Определяем пространства

$$V = \{v \in H^1(\Omega): v = u_D \text{ на } \partial\Omega\}$$

$$\hat{V} = \{v \in H^1(\Omega): \ v = 0 \text{ на } \partial\Omega\}$$

6. Нужен переход от непрерывной задаче к дискретной:

$$\int_\Omega \nabla \mathrm{u_h} \nabla v dx = \int_\Omega f v dx \qquad \forall v \in \hat{V}_h \subset \hat{V}.$$

Переход к вариационной задаче для уравнения Пуассона

7. Нужен переход к вариационной задаче по методу Галеркина:

$$\mathrm{a}(u, v) = L(v) \quad \forall v \in \hat{V}$$

8. Определяем билинейную форму для уравнения Пуассона:

$$\mathrm{a}(u, v) = \int_{\Omega} \nabla u \nabla v \, dx$$

9. Определяем линейный функционал для уравнения Пуассона:

$$L(v) = \int_{\Omega} f v \, dx$$

Переход к вариационной задаче для уравнения Пуассона

3. Выбор конкретных конечно-элементных пространств V и $\hat{V}$ в FEniCS:
   1. Задание расчетной области (сетки)
   2. Задание типа функциональных пространств (степень и тип)

   Задать в FEniCS задачу для УЧП как дискретную вариационную задачу.

4. Решить задачу, визуализировать решение, посчитать погрешность в узлах сетки.

Простой пример решения уравнения Пуассона в единичном квадрате

1. Уравнение Пуассона в единичном квадрате

$$\begin{cases} -\Delta u(x,y) = -6, & (x,y) \in [0,1]\times[0,1] \\ u(x,y) = 1 + x^2 + 2y^2 & (x,y) \in \partial\Omega \end{cases}$$

2. Переходим к вариационной задаче:

$$\mathrm{a}(u,v) = \int_\Omega \nabla u(\mathrm{x},\mathrm{y}) \nabla v(x,y)\, dxdy \quad \forall v \in \hat{V}$$

$$L(v) = \int_\Omega f(x,y) v(x,y) dxdy$$

Пример кода

```
m, k = 8, 8
mesh = UnitSquareMesh(m, k)

V = FunctionSpace(mesh, 'P', 1)
u_D = Expression('1 + x[0]*x[0] + 2*x[1]*x[1]', degree=2)

def boundary(x, on_boundary):
        return on_boundary

bc = DirichletBC(V, u_D, boundary)
u = TrialFunction(V)
v = TestFunction(V)
f = Constant(-6.0)
a = dot(grad(u), grad(v))*dx
L = f*v*dx
u = Function(V)
solve(a == L, u, bc)
```

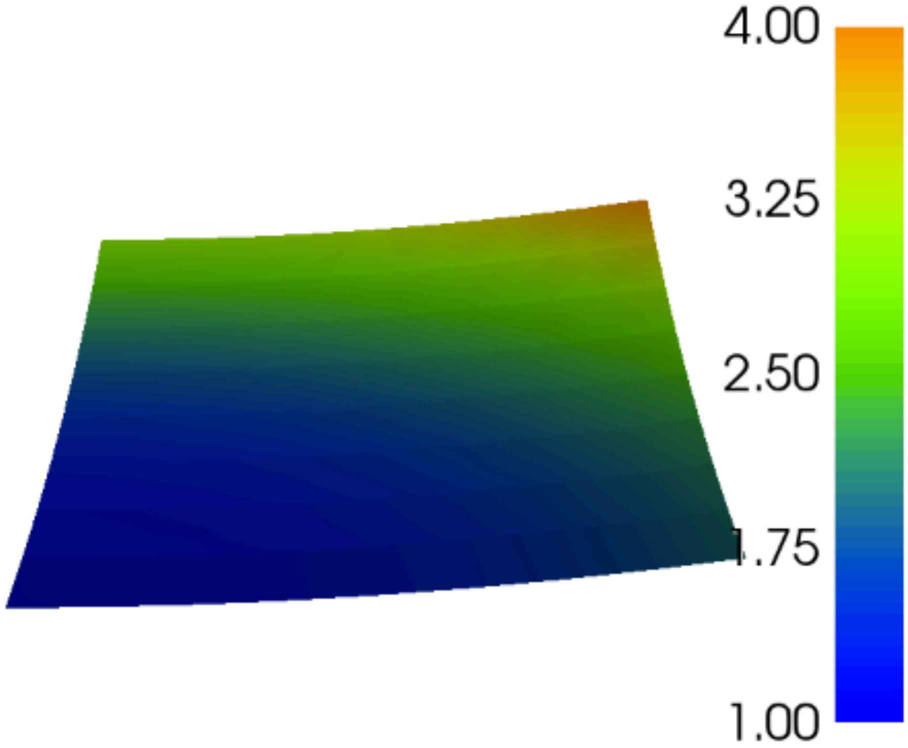
Вычисление погрешностей и стандартная визуализация

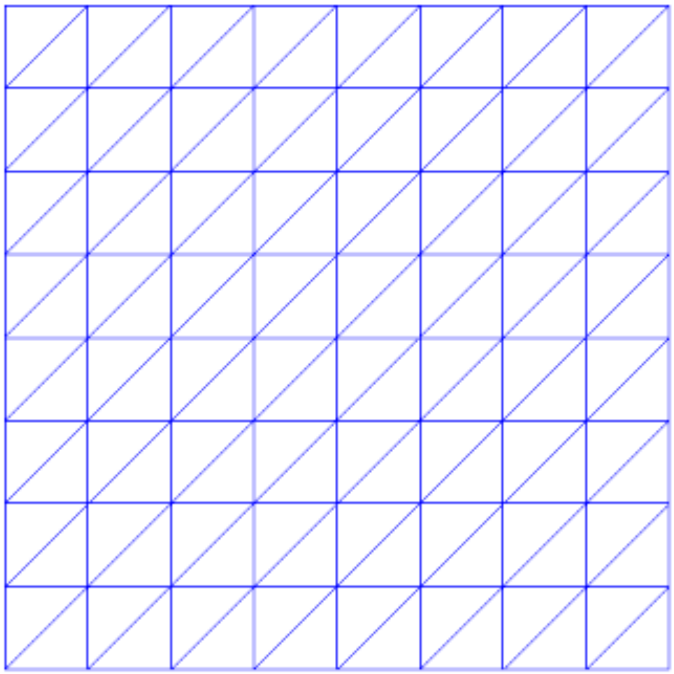
```
error_L2 = errornorm(u_D, u, 'L2')

vertex_values_u_D =
u_D.compute_vertex_values(mesh)
vertex_values_u = u.compute_vertex_values(mesh)
error_C = np.max(np.abs(
            vertex_values_u - vertex_values_u_D))
plot(u)
plot(mesh)
vtkfile = File('poisson1_solution.pvd')
vtkfile << u
interactive()
```

L2-error = 0.00823509807336

C-error = 2.22044604925e-15

Стандартная визуализация

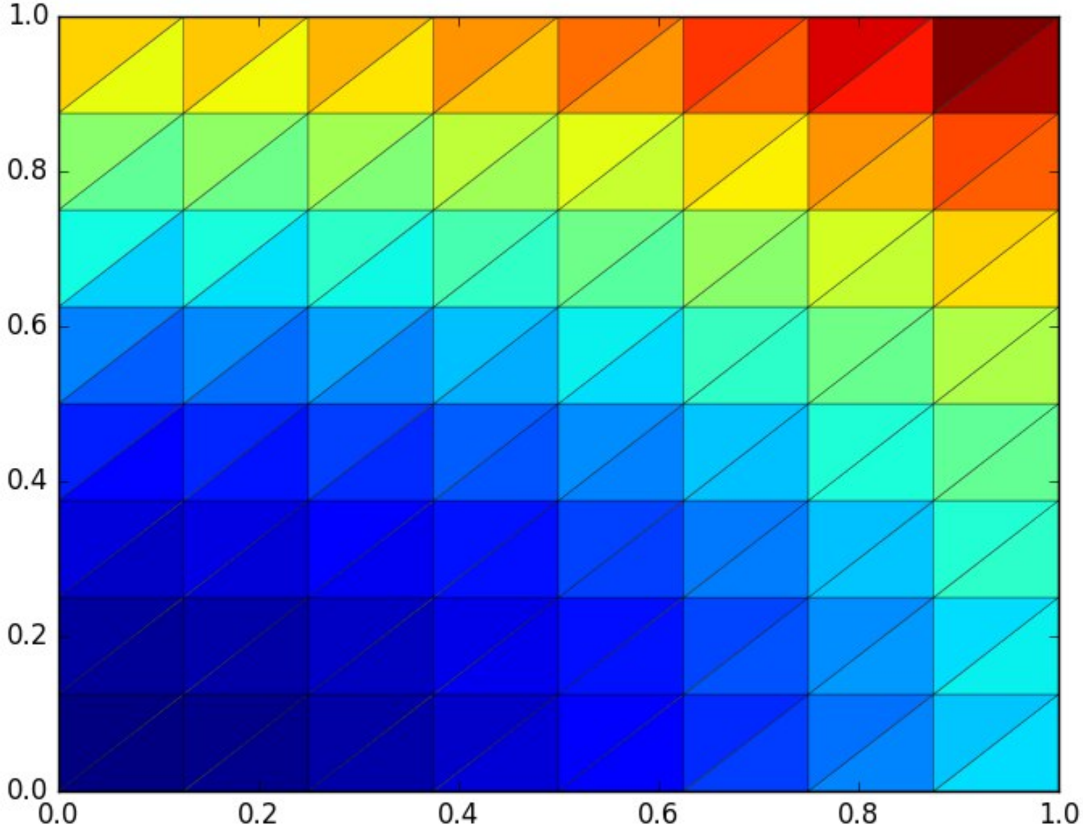


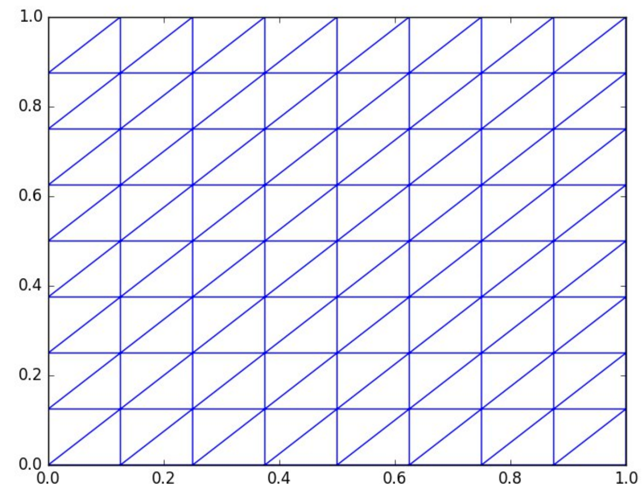


Визуализация с помощью matplotlib

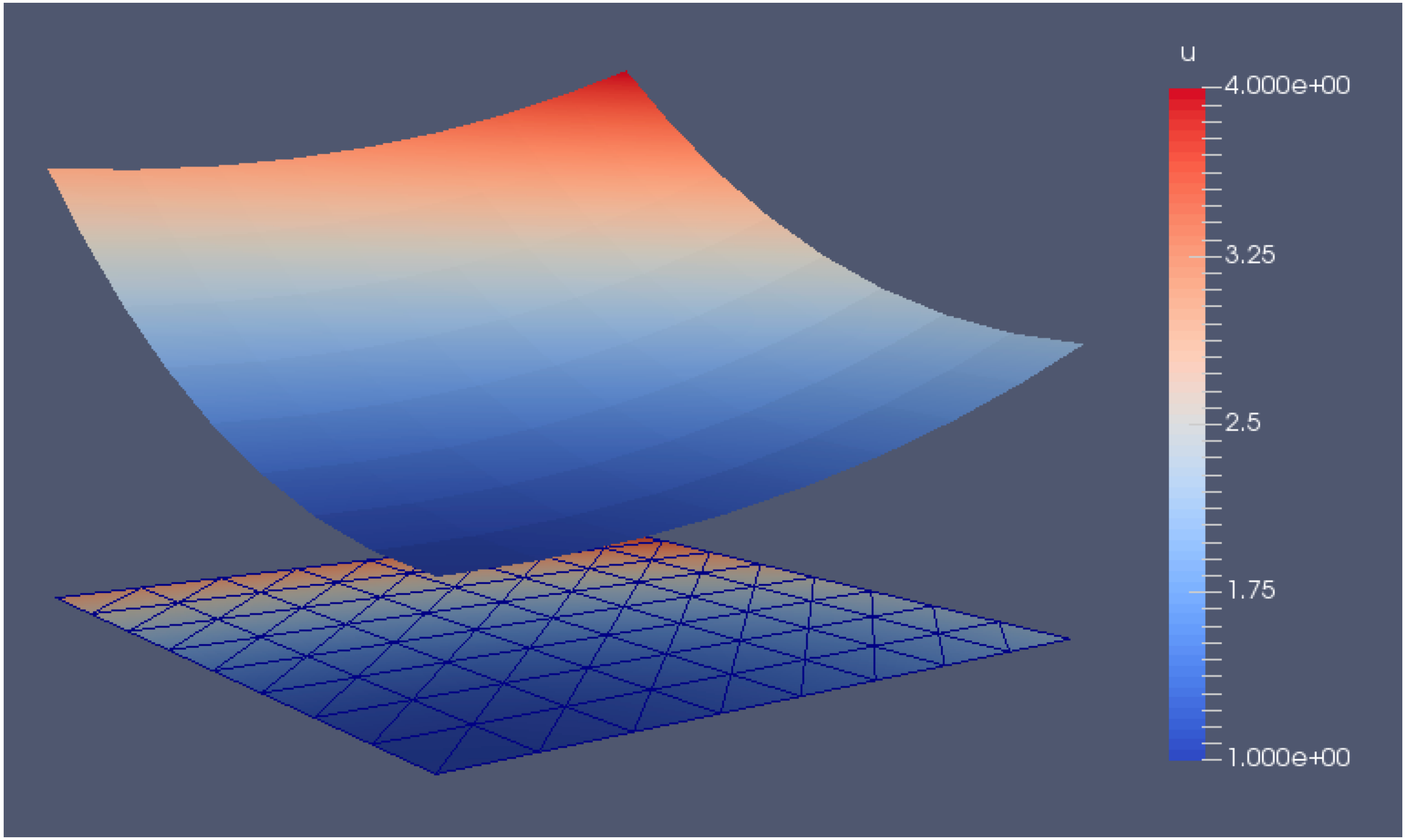
```
n = mesh.num_vertices()
d = mesh.geometry().dim()
mesh_coordinates = mesh.coordinates().reshape((n, d))
triangles = np.asarray([cell.entities(0) for cell in
cells(mesh)])
triangulation = tri.Triangulation(mesh_coordinates[:, 0],
mesh_coordinates[:, 1],                               triangles)

plt.figure()
zfaces = np.asarray([u(cell.midpoint()) for cell in
cells(mesh)])
plt.tripcolor(triangulation, facecolors=zfaces, edgecolors='k')
plt.savefig('u_midpoint.png')
```

Визуализация с помощью matplotlib





Визуализация с помощью ParaView



Граничные условия 2 рода

1. Постановка задачи:

$$\begin{cases} -\Delta u(x) = f(x), & (x) \in \Omega \\ \dfrac{\partial u}{\partial n} = g_N(x), & (x) \in \partial\Omega \end{cases}$$

2. Слабая постановка задачи:

$$a(u,v) := \int_\Omega \nabla u \nabla v \, dx = \int_\Omega f v dx + \int_{\partial\Omega} g_N v ds, v \in \hat{V}$$

3. Решение проблемы с единственностью:
   1. Фиксирование значения на границе
   2. Метод множителей Лагранжа
   3. Метод Крылова

Условия Неймана для уравнения Пуассона

1. Уравнение Пуассона в единичном квадрате

$$\begin{cases} -\Delta u(x,y) = -6, & (x,y) \in [0,1]\times[0,1] \\ -\dfrac{\partial u}{\partial n} = g = 4y & (x,y) \in \Gamma_N: y = 0, y = 1 \\ u(x,y) = 1 + x^2 + 2y^2 & (x,y) \in \Gamma_D: x = 0, x = 1 \end{cases}$$

$$-\int_\Omega (\nabla^2 u) v \, dx = \int_\Omega \nabla u \cdot \nabla v \, dx - \int_{\partial\Omega} \frac{\partial u}{\partial n} v \, ds$$

$$-\int_{\partial\Omega} \frac{\partial u}{\partial n} v \, ds = -\int_{\Gamma_N} \frac{\partial u}{\partial n} v \, ds = \int_{\Gamma_N} g v \, ds$$

2. Переходим к вариационной задаче:

$$\mathrm{a}(u,v) = \int_\Omega \nabla u(\mathrm{x},\mathrm{y}) \nabla v(x,y) \, dxdy \quad \forall v \in \hat{V}$$

$$L(v) = \int_\Omega f(x,y) v(x,y) dxdy - \int_{\Gamma_N} g(x,y) v(x,y) ds$$

Пример кода

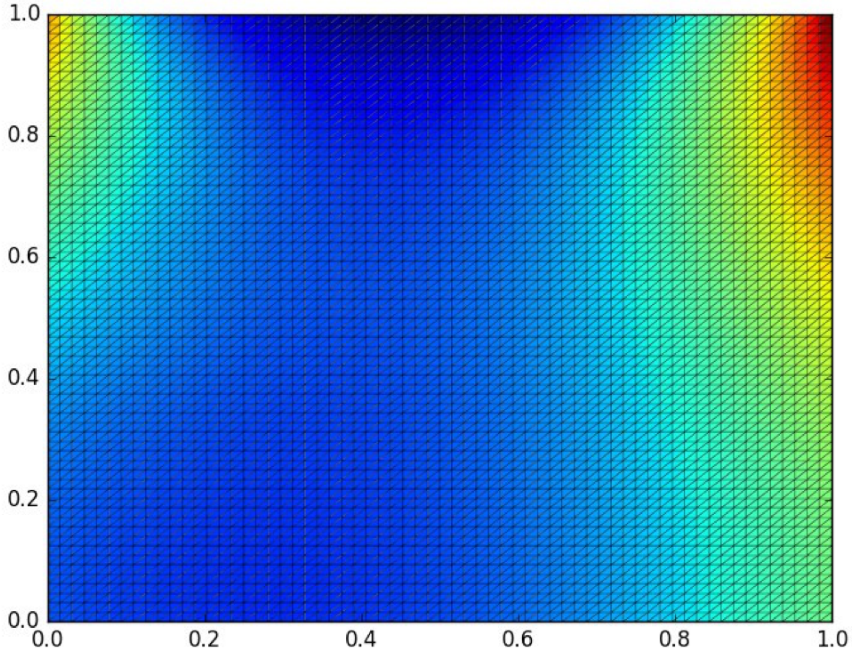
1. Изменение функции граничных условий

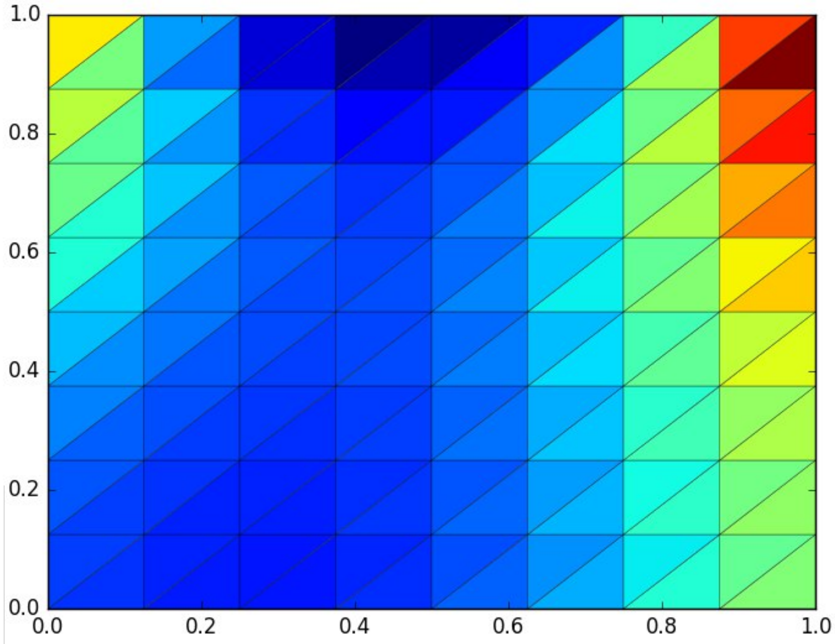
```
tol = 1E-14
def boundary_D(x, on_boundary):
        if on_boundary:
                if near(x[0], 0, tol) or near(x[0], 1, tol):
                        return True
                else:
                        return False
        else:
                return False
```

2. Изменение линейного функционала

```
g = Expression('4*x[1]', degree=1)
L = f*v*dx - g*v*ds
```

Решение





Задача для уравнения теплопроводности

1. Уравнение Пуассона

$$\begin{cases} \dfrac{\partial u}{\partial t} = \nabla^2 u + f \text{ в } \Omega\times[0,T] \\ u = u_D \qquad\qquad \text{на } \partial\Omega\times[0,T] \\ u = u_0 \qquad\qquad\qquad \text{при } t = 0 \end{cases}$$

2. Дискретизация по времени:

$$\frac{u^{n+1} - u^n}{\Delta t} = \nabla^2 u^{n+1} + f^{n+1}$$

3. Приходим к уравнению на каждом шаге:

$$u^0 = u_0,$$
$$u^{n+1} - \Delta t \nabla^2 u^{n+1} - u^n - \Delta t f^{n+1} = 0, n = 0, 1, 2, \dots$$

Задача для уравнения теплопроводности

4. Аналогично приходим к вариационной задаче
$$a(u,v) = L_{n+1}(v)$$
5. Билинейная форма:
$$a(u,v) = \int_\Omega (uv + \Delta t \nabla u \nabla v) dx$$
6. Приходим к уравнению на каждом шаге:
$$L_{n+1}(v) = \int_\Omega (u^n + \Delta t f^{n+1}) dx$$
7. Дополнительная аппроксимация начального условия:
$$\mathrm{a}_0(u,v) = \int_\Omega uv dx = \int_\Omega u_0 v dx = L_0(v)$$
8. Приходим к последовательности вариационных задач

9. $u_e(x,y,y) = 1 + x^2 + \alpha y^2 + \beta t$

Пример кода

```
T = 2.0
num_steps = 10
dt = T / num_steps
alpha = 3
beta = 1.2
nx = ny = 8
mesh = UnitSquareMesh(nx, ny)
V = FunctionSpace(mesh, 'P', 1)
u_D = Expression('1 + x[0]*x[0] + alpha*x[1]*x[1] + beta*t',
degree=2, alpha=alpha, beta=beta, t=0)

def boundary(x, on_boundary):
        return on_boundarybc = DirichletBC(V, u_D,
boundary)
u_n = interpolate(u_D, V)
u = TrialFunction(V)
v = TestFunction(V)
```

Пример кода

```
f = Constant(beta - 2 - 2*alpha)
F = u*v*dx + dt*dot(grad(u), grad(v))*dx - (u_n + dt*f)*v*dx
a, L = lhs(F), rhs(F)

u = Function(V)
t = 0
for n in range(num_steps):
        t += dt
        u_D.t = t
        solve(a == L, u, bc)
        plot(u)
        u_e = interpolate(u_D, V)
        error = np.abs(u_e.vector().array()-
                        u.vector().array()).max()
        print(' t = ', t, ', error = ', error)
        u_n.assign(u)

interactive()
```

Решение