разностями, все остальные *линейные члены* уравнений и краевые условия выбираются в моменты времени $t = t_{m+1}$ (неявная схема), а *нелинейные члены* $F(s,a)$ выбираются в момент времени $t = t_m$ (явная схема). Впрочем, имеются и иные варианты аппроксимации для функции $F(s,a)$. Например, числитель этой функции, т. е. sa, можно представлять в виде $s^{m+1}a^m$ для уравнения (6.5) и в виде $s^m a^{m+1}$ для уравнения (6.6).

Слабую формулировку задачи (6.5)–(6.8) получим, умножая уравнения (6.5), (6.6) на *независимые* тестовые функции $v_s(x,y)$, $v_a(x,y)$, а затем складывая уравнения и интегрируя по частям при помощи формулы Грина (с учетом краевых условий (6.7))

$$\begin{aligned} &\iint\limits_D \left(\frac{s^{m+1}-s^m}{\tau} v_s + \mu_s \nabla s \cdot \nabla v_s \right) dx\,dy - \\ - &\iint\limits_D \gamma(s_0 - s^{m+1} - \rho F(s^m, a^m)) v_s \, dx\,dy + \\ + &\iint\limits_D \left(\frac{a^{m+1}-a^m}{\tau} v_a + \mu_a \nabla a \cdot \nabla v_a \right) dx\,dy - \\ - &\iint\limits_D \gamma(\alpha(a_0 - a^{m+1}) - \rho F(s^m, a^m)) v_a \, dx\,dy = 0. \end{aligned} \tag{6.9}$$

6.3 Фрагменты кодов на языке FreeFem++

Полный код на языке FreeFem++ для решения задачи (6.5)–(6.8) дан в п. 6.4.2. Приведем фрагменты программы, позволяющие использовать результаты, полученные ранее при решении нестационарной задачи для уравнения Лапласа (см. с. 62).

Здесь и далее используем следующие идентификаторы

s^{m+1}	→	`s`	$g(s,a)$	→	`gsa`	α	→	`alpha`
a^{m+1}	→	`a`	$f(s,a)$	→	`fsa`	ρ	→	`rho`
s^m	→	`spr`	μ_s	→	`mus`	β	→	`beta`
a^m	→	`apr`	μ_a	→	`mua`	τ	→	`dt`
v_s	→	`vs`	K	→	`K`	t	→	`t`
v_a	→	`va`	γ	→	`gamma`			

При составлении алгоритма следует учесть, что необходимо ввести *две* неизвестные функции `s` и `a` и для *каждой из них* свои тестовые функции, например, `vs`, `va`

```
Vh s, a, vs, va;
```

Слабая формулировка задачи (6.9) записывается в виде

```
problem Diffusion(s, a, vs, va, solver=UMFPACK) =
             int2d(Th)( s*vs + dt*mus*(dx(s)*dx(vs) + dy(s)*dy(vs)) )
            -int2d(Th)( ( spr + dt*gsa )*vs )
```

```
            +int2d(Th)( a*va + dt*mua*(dx(a)*dx(va) + dy(a)*dy(va)) )
            -int2d(Th)( ( apr + dt*fsa )*va ) ;
```

Сравнивая `problem Diffusion(s, a, vs, va)` с `problem Heat(u, v)` на с. 62, видим, что в данном случае в качестве аргументов `Diffusion` следует использовать `s`, `a`, `vs`, `va`, т. е. перечислять все неизвестные и тестовые функции. Алгоритм решения нестационарной задачи имеет вид (ср. с аналогичным для `problem Heat` на с. 62)

```
for (int m=0; m<=1000; m++)
  {  gsa = gamma*(s0-spr - rho*spr*apr/(1+spr+K*spr^2));
     fsa = gamma*(alpha*(a0-apr) - rho*spr*apr/(1+spr+K*spr^2));
     t = t + dt;
     Diffusion;
     spr = s;    apr = a;
     plot(s);
   }
```

Обратим внимание, что функции `gsa`, `fsa` вычисляются внутри цикла на каждом шаге по времени и должны быть предварительно описаны до их появления в строках кода, т. е. до записи `problem Diffusion`

```
Vh gsa, fsa;
```

Наконец, укажем как на языке FreeFem++ в случае задачи с несколькими неизвестными функциями следует задавать краевые условия первого рода. Например, для задания однородных условий Дирихле на каком-либо участке границы с именем `Bn` в `problem Diffusion(...) =` необходимо добавить строку

```
+ on(Bn, s=0, a=0);
```

$$(s|_{\mathrm{Bn}} = 0, \quad a|_{\mathrm{Bn}} = 0).$$

Иными словами, краевые условия должны быть указаны для каждой неизвестной функции.

Учет краевых условий третьего рода и неоднородных условий второго рода, как обычно, осуществляется при записи слабой (вариационной) формулировки задачи, т. е. при помощи интегралов по контуру `int1d` и интегралов по области `int2d` (для краевых условий третьего рода).

6.4 Окраска шкур животных

Как уже говорилось при задании функций (6.4), задача (6.1)–(6.4) может моделировать процесс формирования окраски шкур животных [19]. Эта прекрасная задача как нельзя лучше позволяет продемонстрировать широкие возможности метода конечных элементов для построения решения в областях сложной формы.

Приведем минимально необходимые сведения, требующиеся для понимания результатов дальнейших расчетов. Предположим, что имеется шкура зародыша животного (двумерная область D сложной формы), на которой в процессе развития эмбриона формируется окраска, например, в виде

чередующихся полос, как у зебры, или пятен, как у леопарда. В простейшей модели считается, что на шкуре животного имеется начальное распределение двух веществ, одно из которых субстрат (красящий пигмент), а другое — косубстрат. В результате химических взаимодействий в присутствии фермента (катализатора) и процессов диффузии происходит перераспределение концентрации красящего вещества.

Система уравнений (6.1), (6.4) как раз и описывает двухкомпонентную смесь, в которой происходит превращение тирозина в меланин при наличии ферментов в эпидерме и (или) шерсти. Вещество с концентрацией s является субстратом (меланин), а вещество с концентрацией a — косубстратом (тирозин). Предполагается, что на шкуре имеется равномерное распределение субстрата s_0 и косубстрата a_0. Члены $\gamma(s_0 - s)$ и $\gamma\alpha(a_0 - a)$ в плотностях источников концентраций (функции g и f) моделируют приток веществ к поверхности шкуры из тела эмбриона.

Функция $F(s, a)$ (см. (6.4)) моделирует реакцию, которую называют реакцией с ингибированием субстратом (K — коэффициент ингибирования). Термин «ингибирование субстратом» означает, что с ростом концентрации s скорость химической реакции $F(s, a)$ стремится к нулю — большое количество субстрата подавляет (ингибирует) процесс протекания реакций.

Процесс возникновения сложной окраски качественно можно объяснить следующим образом. Если в результате химических реакций между компонентами s и a в окрестности какой-либо точки (x, y) области D количество субстрата s становится достаточно большим, то протекание химической реакции полностью подавляется. В такой ситуации решающую роль в распределении концентрации играют диффузионные процессы. В результате диффузии концентрация субстрата s в точке (x, y) начинает уменьшаться и вновь «включаются» химические процессы, приводящие к увеличению концентрации субстрата. При некоторых значениях параметров могут возникнуть сложные пространственно-временные структуры, соответствующие полосам, пятнам и другим рисункам окраски шкуры животного.

Описаный механизм возможного образования структур называется диффузионной неустойчивостью и особенно ярко проявляется в окрестности точки равновесия $(\tilde{s}, \tilde{a})$, которая является решением системы уравнений $f(s, a) = 0$, $g(s, a) = 0$ (и всей задачи в целом).

6.4.1 Вычислительный эксперимент

На рис. 6.1 показано образование пространственной структуры распределения субстрата в области D, которая имитирует шкуру домашней длинношвостой кошки[1]. Для проведения расчетов использован один из наборов параметров, предложенный в [19] (к сожалению, размеры области D там не указаны)

$$K = 0{,}125; \quad \rho = 13; \quad s_0 = 103; \quad a_0 = 77;$$

$$\alpha = 1{,}5; \quad \mu_a = 1; \quad \mu_s = \beta = 7; \quad \gamma = 510.$$

[1] Прототипом области D послужила шкура кошки учебной компьютерной лаборатории кафедры вычислительной математики и математической физики `htpp://vmmf.math.rsu.ru`

Равновесному решению соответствуют следующие значения

$$s = \tilde{s} = 24{,}95939605, \quad a = \tilde{a} = 24{,}97293070$$

(заметим, что в [19] приведены ошибочные значения $\tilde{s} = 24$ и $\tilde{a} = 23$).

В качестве начального распределения (6.3) взято малое возмущение стационарного решения (имитация случайного возмущения)

$$s = \tilde{s} + 10^{-2}\cos(12{,}765\pi x)\sin(12\pi y),$$

$$a = \tilde{a} + 10^{-5}\cos(10\pi x)\sin(16{,}786\pi y).$$

На рис. 6.1 хорошо видно, как с течением времени формируется пространственно-периодическая структура изолиний распределения концентрации s. На самом деле, на рис. 6.1 для лучшей визуализации приведена разность $s(x, y, t) - \tilde{s}$. Дело в том, что в задаче очень быстро устанавливается решение, близкое к стационарному, т. е. диффузионная неустойчивость возникает в окрестности точки $(\tilde{s}, \tilde{a})$.

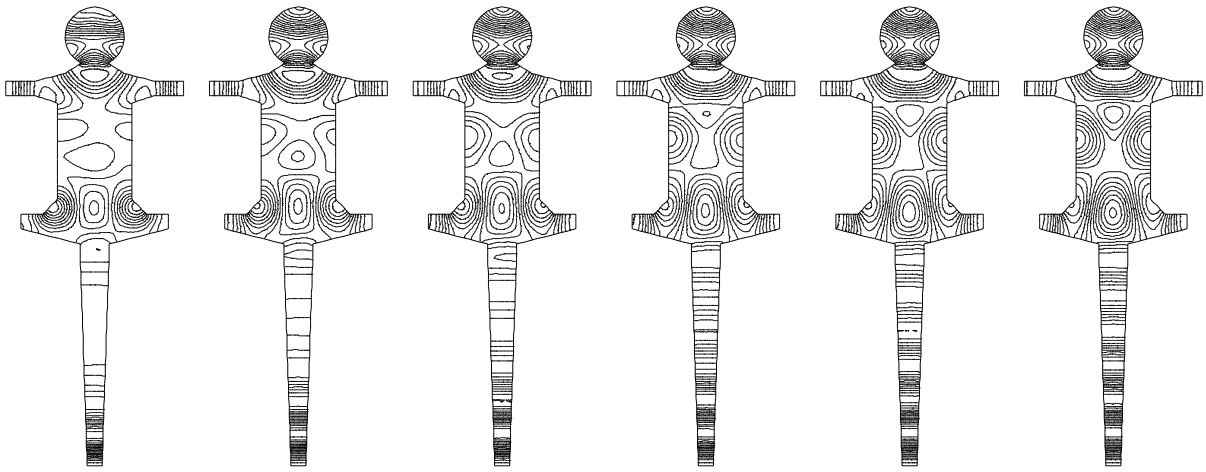

Рис. 6.1. Изолинии функции $s(x, y, t) - \tilde{s}$ при $t = 0{,}025m$, $m = 1, \ldots, 6$

Рис. 6.1 показывает, что с течением времени формируется ярко выраженная полосатость протяженных областей (хвост, лапы). В областях с соизмеримыми размерами (туловище, голова) полосатость не так заметна.

Подчеркнем, что модель (6.1)–(6.4) реально описывает лишь двухцветный механизм формирования полос или пятен. Значениям $(s - \tilde{s}) > 0$ соответствует черный цвет, а значениям $(s - \tilde{s}) < 0$ — белый. На рис. 6.2 (второй слева) показано распределение черно-белых полос на шкуре. Для реализации такой визуализации был использован следующий код (конечно, необходимо описать массив `rr` — `Vh rr`)

```
for (int j=0; j<sdif[].n; j++)
  {
   if (!(sdif[][j]>0))  { rr[][j]=1; }
   if (!(sdif[][j]<0))  { rr[][j]=0; }
  }
plot(rr, bw=1);
```

6.4.1.1 Деформация области

В языке FreeFem++ имеется ключевое слово `movemesh` для деформации исходной области. Пример использования `movemesh`

```
func uu = sin(y*pi)/10,    vv = cos(x*pi)/10;
// коэффициент деформации сетки
real coef = 1.5;
// деформация сетки
Th = movemesh(Th0,[x+coef*uu,y+coef*vv]);
```

На рис. 6.2 показана триангуляция сложной области, использованной для расчетов, показанных на рис. 6.1, и триангуляция области, полученной в результате деформации.

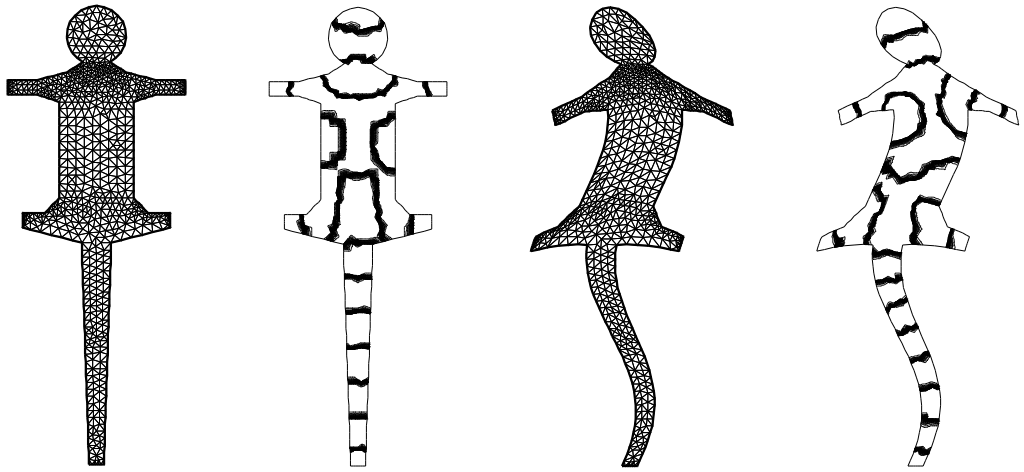

Рис. 6.2. Полосы окраски в момент времени $t = 0{,}1$

6.4.1.2 Различные модификации кода

Подчеркнем, что во всех приводимых программах контроль точности вычислений не производится. Результаты расчетов могут довольно сильно зависеть от количества точек, выбираемых для триангуляции области

```
mesh Th0 = buildmesh(...);
```

от выбора конечных элементов (`P1`, `P2`,...) в операторе

```
fespace Vh(Th, P2);
```

от способа вычисления интегралов `int2d`, `int1d`, от алгоритма решения алгебраических уравнений (`solver`)

```
problem Diffusion(s,a,vs,va, solver=UMFPACK) = ...;
```

Обо всем этом подробно рассказано в гл. 19. Приведем здесь для примера лишь результаты расчета с конечными элементами `P1` и `P2`.

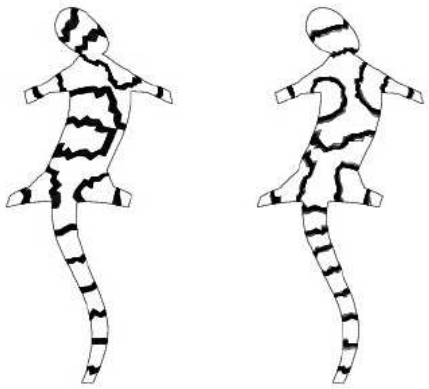

Рис. 6.3. Полосы окраски при $t = 0{,}1$ для конечных элементов `P1` (слева) и `P2`

Как указано в [19], структура полос сильно зависит от параметра γ. При малых γ в области будет формироваться лишь один черный цвет, при увеличении γ возникнет двухцветный окрас (черный верх, белый низ). При дальнейшем увеличении γ будет возникать более частое чередование черного и белого цветов и, наконец, при очень больших γ чередование будет столь велико, что практически останется лишь один черный цвет (кошка повышенной черной пятнистости).

Заметим, что все параметры, входящие в задачу (K, β, α, s_0, a_0,...), могут быть функциями координат и времени. Очевидно, это будет имитировать ситуацию пространственно-неоднородной шкуры и изменения параметров, например, константы ингибирования K с течением времени.

6.4.2 Реализация алгоритма на языке FreeFem++

Приведем полный код программы на языке FreeFem++, которая использовалась для расчета окраски шкуры кошки. Заметим, что большую часть кода (строки 5–65) занимает задание области D (имитирующей форму шкуры), в которой решается задача.

```
1  // описание параметров задачи
2  int n=2;
3  real alpha, beta, gamma, rho, a0, s0, K;
4  real t, dt;
5  // параметры для задания сложной области -- шкуры животного
6  real xx, yy;
7  real w1,w2,w3,w4,w5,w6,w7,w8,w9;
8  real h1,h2,h3,h4,h5,h6,h7,h8,h9,h10;
9  real x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14;
10 real y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12,y13,y14;
11 real x15,x16,x17,x18,x19,x20,x21,x22,x23,x24,x25,x26,x1;
12 real y15,y16,y17,y18,y19,y20,y21,y22,y23,y24,y25,y26,y1;
13 real delta,R,y00;
14 // задание координат для построения границ области
15 // голова
16 delta = 1.1;    R = 0.2;    y00 = 1.0;
17 w1=R*cos(-delta);   h1=y00+R*sin(-delta);
```

```
// параметры для координат точек контура границы
//w1=0.15;
w2=0.20; w3=0.40; w4=0.60; w5=0.25; w6=0.35; w7=0.50; w8=0.10; w9=0.05;
//h1=1.0;
h2=0.75;  h3=0.7;   h4=0.6;   h5=0.55;  h6=-0.1;
h7=-0.2;  h8=-0.3;  h9=-0.4;  h10=-1.9;
// координаты точек левого контура
x2=-w1;  y2=h1;   x3=-w2;  y3=h2;   x4=-w3;  y4=h3;   x5=-w4;  y5=h3;
x6=-w4;  y6=h4;   x7=-w3;  y7=h4;   x8=-w5;  y8=h5;   x9=-w5;  y9=h6;
x10=-w6; y10=h7;  x11=-w7; y11=h7;  x12=-w7; y12=h8;  x13=-w8; y13=h9;
x14=-w9; y14=h10;
// координаты точек правого контура
x1=w1;   y1=h1;   x26=w2;  y26=h2;  x25=w3;  y25=h3;  x24=w4;  y24=h3;
x23=w4;  y23=h4;  x22=w3;  y22=h4;  x21=w5;  y21=h5;  x20=w5;  y20=h6;
x19=w6;  y19=h7;  x18=w7;  y18=h7;  x17=w7;  y17=h8;  x16=w8;  y16=h9;
x15=w9;  y15=h10;
// задание линий контура
// окружность для головы
border L1(t=-delta,pi+delta){ x=R*cos(t);         y=y00+R*sin(t); };
// отрезки на границе для левого контура
//border L1(t=0,1){ x=(1-t)*x1+t*x2;      y=(1-t)*y1+t*y2; };
border L2(t=0,1){ x=(1-t)*x2+t*x3;        y=(1-t)*y2+t*y3; };
border L3(t=0,1){ x=(1-t)*x3+t*x4;        y=(1-t)*y3+t*y4; };
border L4(t=0,1){ x=(1-t)*x4+t*x5;        y=(1-t)*y4+t*y5; };
border L5(t=0,1){ x=(1-t)*x5+t*x6;        y=(1-t)*y5+t*y6; };
border L6(t=0,1){ x=(1-t)*x6+t*x7;        y=(1-t)*y6+t*y7; };
border L7(t=0,1){ x=(1-t)*x7+t*x8;        y=(1-t)*y7+t*y8; };
border L8(t=0,1){ x=(1-t)*x8+t*x9;        y=(1-t)*y8+t*y9; };
border L9(t=0,1){ x=(1-t)*x9+t*x10;       y=(1-t)*y9+t*y10; };
border L10(t=0,1){ x=(1-t)*x10+t*x11;     y=(1-t)*y10+t*y11; };
border L11(t=0,1){ x=(1-t)*x11+t*x12;     y=(1-t)*y11+t*y12; };
border L12(t=0,1){ x=(1-t)*x12+t*x13;     y=(1-t)*y12+t*y13; };
border L13(t=0,1){ x=(1-t)*x13+t*x14;     y=(1-t)*y13+t*y14; };
// нижний горизонтальный отрезок
border L14(t=0,1){ x=(1-t)*x14+t*x15;     y=(1-t)*y14+t*y15; };
// отрезки на границе для правого контура
border L15(t=0,1){ x=(1-t)*x15+t*x16;     y=(1-t)*y15+t*y16; };
border L16(t=0,1){ x=(1-t)*x16+t*x17;     y=(1-t)*y16+t*y17; };
border L17(t=0,1){ x=(1-t)*x17+t*x18;     y=(1-t)*y17+t*y18; };
border L18(t=0,1){ x=(1-t)*x18+t*x19;     y=(1-t)*y18+t*y19; };
border L19(t=0,1){ x=(1-t)*x19+t*x20;     y=(1-t)*y19+t*y20; };
border L20(t=0,1){ x=(1-t)*x20+t*x21;     y=(1-t)*y20+t*y21; };
border L21(t=0,1){ x=(1-t)*x21+t*x22;     y=(1-t)*y21+t*y22; };
border L22(t=0,1){ x=(1-t)*x22+t*x23;     y=(1-t)*y22+t*y23; };
border L23(t=0,1){ x=(1-t)*x23+t*x24;     y=(1-t)*y23+t*y24; };
border L24(t=0,1){ x=(1-t)*x24+t*x25;     y=(1-t)*y24+t*y25; };
border L25(t=0,1){ x=(1-t)*x25+t*x26;     y=(1-t)*y25+t*y26; };
border L26(t=0,1){ x=(1-t)*x26+t*x1;      y=(1-t)*y26+t*y1; };
// формирование сетки - на различных участках границы, в зависимости от
// их длины, выбирается различное количество отрезков разбиения
mesh Th0 =
```

```
69      buildmesh(L1(10*n)+L2(3*n)+L3(3*n)+L4(3*n)+L5(3*n)+L6(3*n)+L7(3*n)
70          +L8(6*n)+L9(3*n)+L10(3*n)+L11(3*n)+L12(3*n)+L13(16*n)+L14(3*n)
71          +L15(16*n)+L16(3*n)+L17(3*n)+L18(3*n)+L19(3*n)+L20(6*n)
72          +L21(3*n)+L22(3*n)+L23(3*n)+L24(3*n)+L25(3*n)+L26(3*n));
73  // для деформации
74  mesh Th=Th0;
75  // функции для деформации сетки
76  //func uu = sin(2*y*pi)/10;
77  //func vv = cos(2*x*pi)/10;
78  func uu = sin(y*pi)/10;
79  func vv = cos(x*pi)/10;
80  // коэффициент деформации сетки
81  real coef=1.5;
82  // деформация сетки (убрать комментарии, если понадобится деформация)
83  //Th = movemesh(Th0,[x+coef*uu,y+coef*vv]);
84  // рисование сетки
85  plot(Th, wait=1);
86  // задание пространства конечных элементов
87  fespace Vh(Th, P2);
88  // на  Vh определяем искомые функции s, a и тестовые функции vs, va
89  Vh s, vs, a, va;
90  // на пространстве Vh определяем вспомогательные функции
91  Vh gsa, fsa;
92  Vh spr, apr;
93  // запись слабой (вариационной) формулировки задачи
94  problem Diffusion(s,a,vs,va, solver=UMFPACK) =
95      int2d(Th)(s*vs + dt*(dx(s)*dx(vs) + dy(s)*dy(vs)) )
96     -int2d(Th)((spr + dt*gsa)*vs)
97     +int2d(Th)(a*va + beta*dt*(dx(a)*dx(va) + dy(a)*dy(va)) )
98     -int2d(Th)((apr + dt*fsa)*va)  ;
99  // задание параметров для исходной задачи
100 alpha=1.5;  K=0.125;  rho=13;  s0=103;  a0=77;  beta=7;  gamma=510;
101 // описание переменных для стационарных состояний
102 real sinit, ainit;
103 // задание вспомогательной функции для вывода информации
104 Vh sdif;
105 // стационарное решение -- корни системы уравнений gsa=0, fsa=0
106 sinit = 24.95939605;  ainit = 24.97293070;
107 // возмущения стационарного решения
108 spr = sinit + 0.01*cos(12.765*pi*x)*sin(12*pi*y);
109 apr = ainit + 0.00001*cos(10*pi*x)*sin(16.786*pi*y);
110 // организуем пошаговое решение задачи
111 t = 0;   dt = 0.001;
112 for (int m=0; m<=1000; m++)
113   { // определение функций gsa, fsa
114     gsa = gamma*(s0-spr - rho*spr*apr/(1+spr+K*spr^2));
115     fsa = gamma*(alpha*(a0-apr) - rho*spr*apr/(1+spr+K*spr^2));
116     t = t + dt;
117     // вызов problem Diffusion
118     Diffusion;
119     // перенесение решения с одного временного слоя на другой
```

```
120       spr = s;
121       apr = a;
122       // формирование разности между решением и стационарным состоянием
123       sdif = s - sinit;
124       // вывод результатов
125       plot(sdif, fill=!(m % 5), //  заливка через пять шагов
126            wait=!(m % 25),      // ожидание "щелчка мыши" для
127                                 // продолжения расчета через 25 шагов
128            value=!(m % 25),     // вывод легенды через 25 шагов
129            cmm=m);
130       cout << 10000*s[].min << " " << 10000*s[].max << endl;
131     }
```

✓. *В языке FreeFem++ имеются встроенные генераторы случайных чисел (см. [1]), которые, в принципе, можно использовать для задания возмущения стационарного решения, изменив соответствующим образом строки 108, 109.*