ЛАБОРАТОРНАЯ РАБОТА №2.

МОДЕЛИРОВАНИЕ L-СИСТЕМ

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.2.2 Контекстно-зависимые L-системы

Каждый символ A может меняться в зависимости от своего *контекста* (окружения). В $2L$-системах контекст двухсторонний: от окружающих его слева и справа символов (A_l и A_r) зависит порождающее правило:

$$A_l < A > A_r \longmapsto newA.$$

Также используют дополнительные символы 0 и 1, отвечающие за контекст, но не интерпретируемые черепашкой, и указывается, какие символы игнорируются при нахождении контекста: $ignore\epsilon : + - F$ означает, что, например, в слове

$$F\underline{0}FF\underline{\underline{1}} + F\underline{0} - 1F$$

контекстом символа $A = 1$ в середине слова являются символы $A_l = 0$ и $A_r = 0$. Запись $* < A > * \longmapsto newA$ означает, что символ A меняется независимо от контекста. Как правило, для символа F правила не задаются. Примеры таких $2L$-систем см. в Таблице 1.3.

Если $2L$-система содержит ветвления, то задание контекста несколько усложняется: близкие символы могут быть разделены длинной последовательностью ветвей. В одном из самых простых случаев дочерние ветви не пересекаются с контекстом материнских ветвей. Например, в слове

$$ABC\underline{D}[EF][\underline{\underline{G}}[HI[JKL]M]\underline{N}OPQ]$$

левым контекстом символа G является D, а правым контекстом — символ N.

1.2.3 Стохастические L-системы

Случайность в построении L-систем можно вводить на двух этапах (см. Задачи 1.1 и 1.2):

- При построении слова вместо одного правила для символа F используется N правил $newF_1, newF_2, \ldots, newF_N$, выбираемых на каждом шаге итерационного процесса с соответствующими вероятностями $p_1, p_2, \ldots, p_N$ $(\sum_{n=1}^{N} p_n = 1)$. Обозначение:

$$p_1 \colon F \overset{p_1}{\longmapsto} newF_1,$$
$$\vdots$$
$$p_N \colon F \overset{p_N}{\longmapsto} newF_N.$$

- При интерпретации слова можно менять длину шага черепашки и угол поворота ее головы случайным образом в соответствии с некоторыми распределениями вероятностей.

1.2.5 L-системы с двумя символами шага

Для удобства построения некоторых систем вместо одно символа шага F полезно ввести два символа F_x и F_y. Например:

$$\begin{aligned} &axiom = F_x, \\ &F_x \longmapsto F_x + F_y+, \\ &F_y \longmapsto -F_x - F_y. \end{aligned}$$

Однако тот же результат можно получить, если символ шага F будет один, но добавятся еще два не интерпретируемых черепашкой символа X и Y:

$$\begin{aligned} &axiom = FX, \\ &X \longmapsto X + YF+, \\ &Y \longmapsto -FX - Y. \end{aligned}$$

Примеры таких L-систем см. в Таблице 1.2. См. также Задачу 1.9.

Алгоритм 1.1: **Реализация L-системы и черепашьей графики (без символа b)**

Вход: $axiom$; ✎ *слово-аксиома*
$newF$; ✎ *слово-правило $F \mapsto newF$*
$newb$; ✎ *слово-правило $b \mapsto newb$*
$level$; ✎ *количество итераций*
θ; ✎ *угол поворота черепашки*
Выход: W ✎ *результирующее кодовое слово*
и его графическая интерпретация.

1: $W = axiom$; ✎ *инициализация искомого слова*
2: **повторять** $level$ **раз**
3: строим слово T, получающееся из слова W заменами: $F \mapsto newF$, $b \mapsto newb$;
4: $W = T$;
✎ *интерпретация слова W:*
5: $\alpha = 0$; ✎ *инициализируем начальное направление черепашки*
6: $(x_0, y_0) = (0, 0)$; ✎ *инициализируем начальное положение черепашки*
7: $path = \{(x_0, y_0)\}$; ✎ *инициализируем массив точек ломаной пути черепашки*
8: $stack = \emptyset$; ✎ *инициализируем стек ветвления*

9: **для** $j = 1, \ldots,$ длина слова W
10: **если** $W[j] = +$, **то** $\alpha = \alpha + \theta$;
11: **если** $W[j] = -$, **то** $\alpha = \alpha - \theta$;
12: **если** $W[j] = F$, **то**
13: $x = x_0 + \cos\alpha$;
14: $y = y_0 + \sin\alpha$;
15: $path = path \cup (x, y)$;
16: $(x_0, y_0) = (x, y)$;
17: **если** $W[j] = [$, **то** $stack = stack \cup (x_0, y_0, \alpha)$;
18: **если** $W[j] =]$, **то**
19: $(x_0, y_0, \alpha) = stack(-1)$; ✎ *достаем последний элемент из стека*
20: удаляем последний элемент из стека;
21: соединяем массив точек $path$ ломаной и выводим на экран;

Программируем на Maxima (листинг № 1.1):

Реализация L-системы и черепашьей графики (без символа b)

```
load(stringproc);                              /* загрузка пакета работы со строками! */
axiom: "F++F++F"$                                    /* начальное слово-аксиома */
newF: "F-F++F-F"$                                  /* порождающее правило для F */
newb: "b"$                                         /* порождающее правило для b */
level: 4$                                                /* количество итераций */

W: axiom$                                /* реализация L-системы, результат - слово W */
thru level do
    (T: "",
    for j:1 thru slength(W) do
       (if sequal(charat(W,j),"+") then T:simplode([T,"+"]),
        if sequal(charat(W,j),"-") then T:simplode([T,"-"]),
        if sequal(charat(W,j),"[") then T:simplode([T,"["]),
        if sequal(charat(W,j),"]") then T:simplode([T,"]"]),
        if sequal(charat(W,j),"F") then T:simplode([T,newF]),
        if sequal(charat(W,j),"b") then T:simplode([T,newb])
        ),
    W:T)$
      /* реализация черепашьей графики: интерпретация слова, результат - чертеж */
                                                  /* символ b здесь не реализован! */
pict(word,theta):=(

    alpha: 0,                                          /* угол начального направления */
    [x0,y0]: [0,0],                                        /* начальное положение */
    path: [[x0,y0]],                  /* массив последовательных точек пути черепашки */
    stack: [],                                         /* стек ветвлений, пока пустой */

    for j:1 thru slength(word) do
      (if sequal(charat(word,j),"+") then alpha: alpha+theta,
       if sequal(charat(word,j),"-") then alpha: alpha-theta,
       if sequal(charat(word,j),"F") then (

                                        x: x0+float(cos(alpha)),
                                        y: y0+float(sin(alpha)),
                                        path: endcons([x,y],path),
                                        x0: x,
                                        y0: y
                                        ),
       if sequal(charat(word,j),"[") then stack: endcons([x0,y0,alpha],stack),
       if sequal(charat(word,j),"]") then (   /* вызов последнего эл-та стека */
          [x0,y0,alpha]:last(stack),
          stack:rest(stack,-1)                          /* его удаление из стека */
                                          )
       ),
    plot2d([discrete,path],[box,false],[plot_format,xmaxima],[xlabel,""],[ylabel,""])
)$

pict(W,%pi/3)$                                  /* вызов команды построения чертежа */
```

Комментарий.

Алгоритм состоит из двух частей: в строках 7–18 происходит построение кодового слова, а в строках 21–46 — его интерпретация и графическая реализация.

Напомним некоторые сведения о синтаксисе и командах в Maxima (дальнейшие сведения можно найти непосредственно в руководстве Maxima).

`load(`*имя пакета*`)` — загрузка дополнительного пакета команд;
`a:b` — присваивание переменной `a` значения `b`;
`;` — завершает одиночную команду (ставится в Maxima по умолчанию);
`$` — завершает одиночную команду, запрещая вывод на экран результата выполнения команды;
`/* комментарий */` — запись комментария;
`"`*символы в двойных кавычках*`"` — обозначение строк;
`%pi` — константы (например π) начинаются со знака `%`;
`[x1,x2,...]` — массив значений `x1,x2,...`;
`a[i]` — i-й элемент массива `a`;
`thru l do` — упрощенная конструкция цикла `for`: повторять цикл `l` раз; пропущена инициализация переменной — такое сокращение удобно, когда сама переменная цикла не используется внутри него (таким образом, ее можно не определять);
`length(a)` — количество элементов в массиве `a`;
`slength(w)` — количество символов в строке `w`;
`sequal(w1,w2)` — проверка равенства двух слов `w1` и `w2`;
`charat(w1,k)` — `k`-й символ в строке `w1`;

`simplode(w1,w2)` — конкатенация двух строк `w1` и `w2`;
`endcons(elem,list)` — добавить элемент `elem` к концу массива `list`. Аналог этой операции для массивов:
`append(l1,l2)` — присоединение массива `l2` к концу массива `l1`;
`last(a)` — последний элемент массива `a`;
`rest(a,n)` — удалить из массива `a` (а также из матрицы или другого выражения) `n` первых элементов, если `n`>0, или `n` последних элементов, если `n`<0.

`plot2d([discrete,[[x1,y1],[x2,y2],...]])` — график дискретного множества точек;

параметры графики команды plot:

`[box,false]` — скрыть обрамляющий графику прямоугольник и оси;
`[plot_format, xmaxima]` — выбор графического формата (`gnuplot` по умолчанию);
`[xlabel,w]` — пометить ось x строкой `w` (пустая строка `""` для скрытия этой пометки, нужно в формате xmaxima).
`[gnuplot_preamble, "set size ratio -1"]` — управление соотношением сторон графики;
Более удобным для отображения графики является пакет **draw**[3], тогда строчку 46 можно заменить командами

```
load(draw)$
draw2d(points(path), point_type=dot, points_joined=true, axis_left=false,
    axis_right=false, axis_top=false,axis_bottom=false, xtics=false,ytics
    =false, user_preamble="set size ratio -1")$
```

Отображение дискретного массива точек задается командой

```
draw2d(points([[x1,y1],[x2,y2],...]))
```

параметры графики команды draw:

`point_type=dot` — отображать точки «обычными» точками;
`points_joined=true` — соединять последовательные точки отрезками;
`axis_left=false,axis_right=false,axis_top=false,axis_bottom=false` — не отображать соответствующие оси координат;
`xtics=false,ytics=false` — не отображать засечки на осях координат;

`user_preamble="set size ratio -1"` — установить масштаб сторон 1 : 1. Эта команда эквивалентна опции

`proportional_axes=xy`

Заметим, гораздо проще собрать все однотипные опции в пользовательский массив:

`noframe:[axis_left=false,axis_right=false,axis_top=false,axis_bottom=false, xtics=false,ytics=false,user_preamble="set size ratio -1"]$`

и использовать его как опцию команды: `draw2d(...,points(...),noframe)`.

Также, часто используемые опции графики можно сохранить глобально командой `set_draw_defaults(`*опции*`)`. Вернуться к опциям по умолчанию можно «пустой» командой `set_draw_defaults()`.

Заметим, что опции `draw` могут быть глобальными (тогда в команде `draw` они стоят после графического объекта) и локальными (тогда их нужно писать перед графическим объектом, поскольку, если таковых несколько, у каждого могут быть свои опции).

Пользователи wxMaxima могут воспользоваться модифицированными командами `wxplot2d` и `wxdraw2d` для вывода графики в рабочую область приложения, а не в отдельное окно (попробуйте!).

Поскольку итеративные вычисления часто являются ресурсоёмкими, полезно оценивать количество итераций по статистике затраченного времени;

измерение затраченного времени:

`showtime: true` — включает слежение за временем выполнения команд;

`timer(f)` — помещает функцию `f` в таймер на слежение;

`untimer(f)` — убирает функцию `f` из таймера;

`timer_info()` — вызывает статистику вызова функций, помещенных в таймер;

`elapsed_run_time()` — количество секунд в данной сессии Maxima, затраченных на подсчёты.

Также отметим, для отлаживания программы полезна команда

`kill(all)` — стирает все пользовательские определения функций и переменных;

`kill(z)` — стирает определение функции или переменной с именем `z`.

Помните, при осуществлении вывода графики на экран процесс вычислений в Maxima не завершается, пока не закрыты графические окна gnuplot, xmaxima.

2. ЗАДАНИЯ ПО ЛАБОРАТОРНОЙ РАБОТЕ

Все задания выполняются в системах MAXIMA, MathCAD и на языке программирования выбранном студентом.

Задание 1.

Реализовать алгоритм L-системы со случайным выбором порождающих правил (см. параграф 1.2.3). Испытать алгоритм на примере: `axiom: F`,

`newF1: F[+F]F[-F]F`, `newF2: F[+F]F`, `newF3: F[-F]F`, `p1: 0.33`, `p2: 0.33`, `p3: 0.34`.

Задание 2.

Реализовать алгоритм рандомизированной L-системы. Угол поворота и длина шага могут быть случайными числами. Подберите их распределение так, чтобы

добиться наилучшего (эстетического) результата.

Задание 3.

При выводе на экран Maxima автоматически подгоняет размеры изображения так, чтобы оно полностью уместилось. Заметьте, что, вообще говоря, длина ин-

терпретируемого слова постоянно и неограниченно растет. Соответственно экспоненциально увеличиваются и линейные размеры графики. Как сделать процедуру нормировки так, чтобы картинка всегда была расположена в заданной области (например, квадрате $[-2,2] \times [-2,2]$)?

Задание 4.

Построить фигуры, задающиеся правилами из Таблицы 1.1.

Куст axiom: F newF: -F+F+[+F-F-]-[-F+F+F] theta: %pi/8	**Сорняк** axiom: F newF: F[+F]F[-F]F theta: %pi/7
Кривая Пеано axiom: F newF: F-F+F+F+F-F-F-F+F alpha: %pi/4 theta: %pi/2	**Цветок** axiom: F[+F+F][-F-F][++F][--F]F newF: FF[++F][+F][F][-F][--F] alpha: %pi/2 theta: %pi/16
Мозаика (острова и озера) axiom: F-F-F-F newF: F-b++FF-F-FF-Fb-FF+b-FF+F+FF+Fb+FFF newb: bbbbbb theta: %pi/2	**Цепочка** axiom: F+F+F+F newF: F+b-F-FFF+F+b-F newb: bbb theta: %pi/2
Снежинка axiom: [F]+[F]+[F]+[F]+[F]+[F] newF: F[+FF][-FF]FF[+F][-F]FF theta: %pi/3	**Остров** axiom: F+F+F+F newF: F+F-F-FFF+F+F-F theta: %pi/2
Снежинка Коха axiom: [F]+[F]+[F]+[F]+[F]+[F] newF: F[+FF][-FF]FF[+F][-F]FF theta: %pi/3	**Треуголка** axiom: F newF: F+F-F-F+F theta: %pi/2

<table>
<tr><td>Остров Коха
axiom: F-F-F-F
newF: F-F+F+FF-F-F+F
theta: %pi/2</td><td>Цепочка еще
axiom: F-F-F-F
newF: FF-F-F-F-F-F+F
theta: %pi/2</td></tr>
<tr><td>Коврик
axiom: F-F-F-F
newF: FF-F-F-F-FF
theta: %pi/2</td><td>Кирпичи
axiom: F-F-F-F
newF: FF-F+F-F-FF
theta: %pi/2</td></tr>
<tr><td>Кристаллы
axiom: F-F-F-F
newF: FF-F-F-F
theta: %pi/2</td><td>Заполненный остров
axiom: F-F-F-F
newF: F-FF--F-F
theta: %pi/2</td></tr>
<tr><td colspan="2">axiom: AAAA, theta: %pi/12 Спиральное покрытие
newA: X+X+X+X+X+X+
newX: [F+F+F+F[--X-Y]+++++F++++++++F-F-F-F]
newY: [F+F+F+F[--Y] +++++F++++++++F-F-F-F]</td></tr>
</table>

Таблица 1.1: Правила для черепашьей графики.

Задание 5.

Реализовать алгоритм *L*-системы с двумя символами шага (см. параграф 1.2.5) и опробовать его на примерах из Таблицы 1.2.

Кривая Серпинского	**Кривая Гильберта**	**Кривая Пеано**
axiom: F+X+F+XF	axiom: X	axiom: X
newF: F	newF: F	newF: F
newX: XF-F+F-XF+F+XF-F+F-X	newX: -YF+XFX+FY-	newX: XFYF+F+YFXFY-F-XFYFX
alpha: %pi/4	newY: +XF-YFY-FX+	newY: YFXFY-F-XFYFX+F+YFXFY
theta: %pi/2	theta: %pi/2	theta: %pi/2
Кривая Госпера	**Дракон Хартера-Хайтвея**	**Наконечник Серпинского**
axiom: XF	axiom: FX	axiom: YF
newF: F	newF: F	newF: F
newX: X+YF++YF-FX--FXFX-YF+	newX: X+YF+	newX: YF+XF+Y
newY: -FX+YFYF++YF+FX--FX-Y	newY: -FX-Y	newY: XF-YF-X
theta: %pi/3	theta: %pi/2	theta: %pi/3

Таблица 1.2: Правила для черепашьей графики с двумя символами шага.

Задание 6.

Реализовать алгоритм $2L$-системы (см. параграф 1.2.2) и опробовать его на примерах из Таблицы 1.3.

Куст 1	**Куст 2**	**Куст 3**
$\theta = 22.5°$	$\theta = 22.5°$	$\theta = 25.75°$
ignore: +-F	ignore: +-F	ignore: +-F
axiom: F1F1F1	axiom: F1F1F1	axiom: F1F1F1
0 < 0 > 0 ⟼ 0	0 < 0 > 0 ⟼ 1	0 < 0 > 0 ⟼ 0
0 < 0 > 1 ⟼ 1[+F1F1]	0 < 0 > 1 ⟼ 1[-F1F1]	0 < 0 > 1 ⟼ 1
0 < 1 > 0 ⟼ 1	0 < 1 > 0 ⟼ 1	0 < 1 > 0 ⟼ 0
0 < 1 > 1 ⟼ 1	0 < 1 > 1 ⟼ 1	0 < 1 > 1 ⟼ 1[+F1F1]
1 < 0 > 0 ⟼ 0	1 < 0 > 0 ⟼ 0	1 < 0 > 0 ⟼ 0
1 < 0 > 1 ⟼ 1F1	1 < 0 > 1 ⟼ 1F1	1 < 0 > 1 ⟼ 1F1
1 < 1 > 0 ⟼ 0	1 < 1 > 0 ⟼ 1	1 < 1 > 0 ⟼ 0
1 < 1 > 1 ⟼ 0	1 < 1 > 1 ⟼ 0	1 < 1 > 1 ⟼ 0
* < + > * ⟼ -	* < + > * ⟼ -	* < + > * ⟼ -
* < - > * ⟼ +	* < - > * ⟼ +	* < - > * ⟼ +

Куст 4	**Куст 5**	**Куст 6**
$\theta = 25.75°$	$\theta = 22.5°$	
ignore: +-F	ignore: +-F	
axiom: F0F1F1	axiom: F1F1F1	
0 < 0 > 0 ⟼ 1	0 < 0 > 0 ⟼ 0	
0 < 0 > 1 ⟼ 0	0 < 0 > 1 ⟼ 1[-F1F1]	
0 < 1 > 0 ⟼ 0	0 < 1 > 0 ⟼ 1	Ваш
0 < 1 > 1 ⟼ 1F1	0 < 1 > 1 ⟼ 1	
1 < 0 > 0 ⟼ 1	1 < 0 > 0 ⟼ 0	пример
1 < 0 > 1 ⟼ 1[+F1F1]	1 < 0 > 1 ⟼ 1F1	
1 < 1 > 0 ⟼ 1	1 < 1 > 0 ⟼ 1	
1 < 1 > 1 ⟼ 0	1 < 1 > 1 ⟼ 0	
* < + > * ⟼ -	* < + > * ⟼ -	
* < - > * ⟼ +	* < - > * ⟼ +	

Таблица 1.3: Правила для $2L$-системы (рекомендуется 30 итераций).