КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт экологии и природопользования,

кафедра моделирования экологических систем

Институт математики и механики, кафедра общей математики

Зарипов Ш.Х., Абзалилов Д.Ф., Костерина Е.А.

Задачи математической экологии и пакет Maxima

Учебное пособие

Казань

2015

УДК 51-7

Печатается по решению Редакционно–издательского совета ФГАОУ ВПО "Казанский (Приволжский) федеральный университет и учебно-методической комиссии Института экологии и природопользования

Протокол N 5 от 12мая 2015 г.

заседания кафедры моделирования экологических систем

Протокол N 9 от 15 апреля 2015 г.

Авторы–составители

д.ф.м.н. Зарипов Ш.Х., д.ф.м.н. Абзалилов Д.Ф., к.ф.м.н. Костерина Е.А.

Научный редактор

д.ф.м.н., профессор Скворцов Э.В.

Задачи математической экологии и пакет Maxima: учебное пособие / Зарипов Ш.Х., Абзалилов Д.Ф., Костерина Е.А. – Казань: Изд–во Казанского федерального университета, 2015. – 120 с.

Учебное пособие содержит примеры решения задач математической экологии и краткое описание программы Maxima и. Пособие рекомендовано для бакалавров и магистров по направлению "Экология и природопользование".

Содержание

Введение

В настоящее время интегрированные математические пакеты активно применяются в науке и образовании, как в различных физико-математических дисциплинах, так и в области естественных наук. Это относится и к экологии, где часто возникает необходимость решения задач математического моделирования. В образовательном процессе математические пакеты успешно дополняют другие готовые программные средства. Студенты института экологии и природопользования знакомятся с элементами математического моделирования природных и экологических процессов на основе программного пакета Maxima, который является свободно распространяемым программным обеспечением и охватывает широкий спектр методов символьной и вычислительной математики: решение систем линейных и нелинейных алгебраических, трансцендентных и дифференциальных уравнений, интегрирование и дифференцирование, задачи векторной алгебры и теории матриц, исследование свойств функций, построение графиков и т.п. Имеется возможность производить арифметические вычисления и находить аналитические решения задач, выражаемые формулами. В тех случаях, когда точное решение математической задачи найти невозможно, применяются методы вычислительной математики, позволяющие найти приближенное решение задачи с определенной числовой погрешностью. Простота языка Maxima и доступность пакета делают его незаменимым инструментом в учебном процессе на естественнонаучных направлениях. Целью настоящего учебного пособия являются знакомство с пакетом Maxima и освоение решения типичных задач математической экологии.

ГЛАВА 1. ЗАДАЧИ МАТЕМАТИЧЕСКОЙ ЭКОЛОГИИ

§ 1. Дифференциальные уравнения в теории эпидемий (модели Бейли)

Рассмотрим задачу о распространении эпидемии инфекционного заболевания в рамках одной популяции [3,4]. Пренебрегая неоднородностью распределения популяции по пространству, введем две функции $x(t)$ и $y(t)$, характеризующие число незараженных и зараженных особей в момент времени t. В начальный момент времени $t=0$ известны начальные значения $x(0)=n$ и $y(0)=a$.

Для того чтобы построить математическую модель, воспользуемся гипотезой: инфекция передается при встрече зараженных особей с незараженными. Это означает, что число незараженных особей будет убывать с течением времени пропорционально количеству встреч между зараженными и незараженными особями, т.е. пропорционально произведению xy.

На основании принятого предположения выразим убыль Δx незараженных особей за промежуток времени Δt в виде

$$\Delta x = x(t+\Delta t) - x(t) = -\beta xy\Delta t \qquad (1.1)$$

Величина β представляет собой коэффициент пропорциональности. Перейдем в (1.1) к пределу при $\Delta t \to 0$

$$\lim_{\Delta t\to 0}\frac{\Delta x}{\Delta t} = \frac{dx}{dt} = -\beta xy \qquad (1.2)$$

Для замыкания модели будем считать, что болезнь не приводит к смертности, следовательно, можно написать условие баланса

$$a + n = x + y = \text{const} \qquad (1.3)$$

Учитывая (1.3), перепишем (1.2) и добавим начальное условие

$$\frac{dx}{dt} = -\beta x(n + a - x) \tag{1.4}$$

$$x(0)=n \tag{1.5}$$

Формулы (1.4), (1.5) представляют собой математическую модель динамики численности незараженных особей. Коэффициент пропорциональности β в модели характеризует вероятность передачи инфекции при встречах больных и здоровых особей. В общем случае значение параметра β зависит от вида особи и типа болезни.

Считая β постоянной величиной, найдем решение обыкновенного дифференциального уравнения (1.4). Разделив переменные, можем переписать (1.4) в виде

$$\frac{dx}{x(n + a - x)} = -\beta dt \tag{1.6}$$

Разложим левую часть (4.6) на простые дроби и проинтегрируем

$$\frac{1}{n + a}\left(\frac{dx}{x} + \frac{dx}{n - x + a}\right) = -\beta dt$$

$$\frac{1}{n + a}\left(\ln x - \ln(n - x + a)\right) = -\beta t + C$$

Потенцируя последнее выражение, придем к равенству

$$\frac{x}{n - x + a} = Ce^{-\beta(n+a)t} \tag{1.7}$$

Учитывая начальное условие (1.5), из (1.7) получим окончательное выражение для искомой функции

$$x(t) = \frac{n(n + a)}{n + ae^{\beta(n+a)t}} \tag{1.8}$$

При известном $x(t)$ число $y(t)$ зараженных особей определится из условия баланса (1.3)

$$y = a + n - x \tag{1.9}$$

Примеры графиков функций $x(t)$ и $y(t)$, вычисленных по формулам (1.8), (1.9) при нескольких значениях параметра β, приведены на рис. 1.1-

1.2. Начальные значения числа незараженных и зараженных особей приняты равными n=200, a=100 (программа **P1.1**). При увеличении β скорость передачи инфекции увеличивается, и численность незараженных особей падает быстрее.

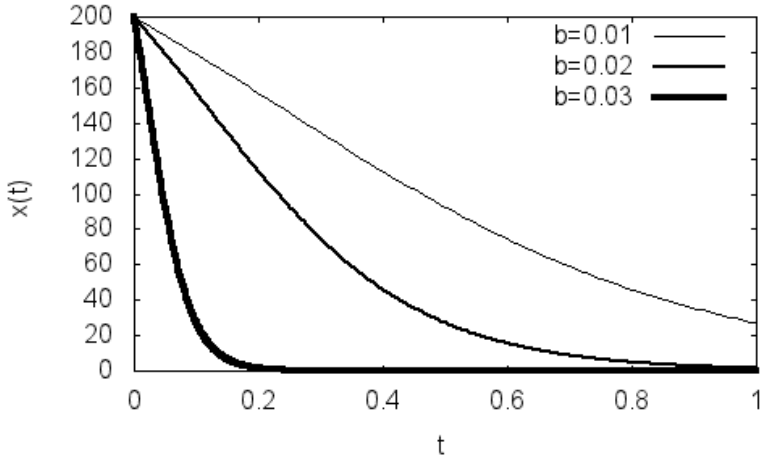
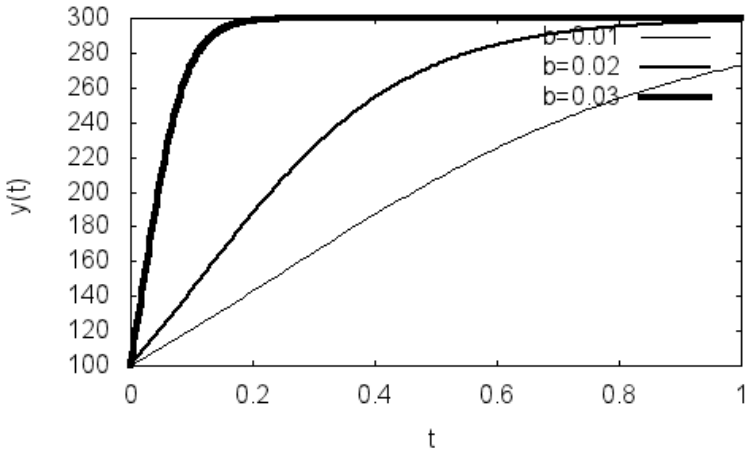


Рис. 1.1. Динамика численности незараженных особей $x(t)$ согласно модели (1.4) для n=200, a=100 и различных β



б

Рис. 1.2. Динамика численности зараженных особей $y(t)$ согласно модели (1.4) для n=200, a=100 и различных β

Текст программы P1.1

```
n:200$
a:100$
x:n*(n+a)/(n+a*exp(bt*(n+a)*t))$
x1:x, bt:0.01$
x2:x, bt:0.02$
x3:x, bt:0.1$
wxplot2d([x1,x2,x3],[t,0,1],[legend, "b=0.01","b=0.02","b=0.03"],[style,
[lines,1,5],[lines,2,5],[lines,4,5]],[ylabel, "x(t)"],[xlabel, "t"]);
```

```
wxplot2d([a+n–x1,a+n–x2,a+n–x3],[t,0,1],[legend,
"b=0.01","b=0.02","b=0.03"],[style,
[lines,1,5],[lines,2,5],[lines,4,5]],[ylabel, "y(t)"],[xlabel, "t"]);
```

Изменим приведенную модель, добавляя в нее еще один процесс – выздоровление больных особей. Для этого введем новую функцию $z(t)$, выражающую число выздоровевших особей. Новая математическая модель может быть представлена системой уравнений

$$\begin{cases} \dfrac{dx}{dt} = -\beta xy \\ \dfrac{dy}{dt} = \beta xy - \gamma y \\ \dfrac{dz}{dt} = \gamma y \end{cases} \tag{1.10}$$

где параметр γ характеризует степень выздоровления и определяется видом болезни и типом особи. Число выздоровевших особей в начальный момент времени равно нулю, поэтому начальные условия для системы (1.10) примут вид

$$x(0) = n, y(0) = a, z(0) = 0 \tag{1.11}$$

Условие баланса (1.3) перепишется как

$$x + y + z = n + a \tag{1.12}$$

Разделив второе уравнение (1.10) на первое, придем к уравнению

$$\frac{dy}{dx} = -1 + \frac{\gamma}{\beta}\frac{1}{x} \tag{1.13}$$

решение которого с учетом начальных условий (1.11) запишется так:

$$y + x - \frac{\gamma}{\beta}\ln x = a + n - \frac{\gamma}{\beta}\ln n \tag{1.14}$$

Исключая *y* из (1.12) и (1.14), получим связь *x* и *z* в виде

$$x = ne^{-\frac{\beta}{\gamma}z} \tag{1.15}$$

Выразив с помощью уравнений (1.12), (1.14) и (1.15) связь y через z и подставив в третье уравнение (1.10), придем к уравнению

$$\frac{dz}{dt} = \gamma \left[n + a - z - ne^{-\frac{\beta}{\gamma} z} \right] \tag{1.16}$$

Результаты решения уравнения (1.16) с учетом начального условия (1.11) приведены на рис. 1.3 (программа **P1.2**). Видно, что с ростом t все особи успевают заболеть, т.е. величина x падает до нуля. Численность зараженных особей y сначала растет, но дальше уменьшается в связи с их выздоровлением. При $t \to \infty$ модель предсказывает полное выздоровление всех особей.

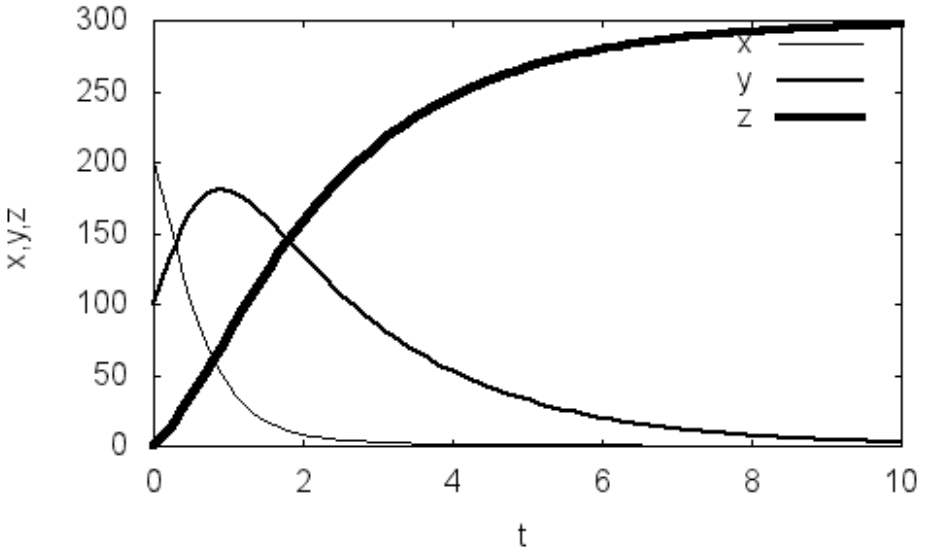


Рис. 1.3. Динамика численности незараженных $x(t)$, зараженных $y(t)$ и выздоровевших $z(t)$ особей согласно модели (1.10) для n=200, a=100, β=0.01, γ=0.5

Текст программы P1.2

```
load("dynamics")$
depends(z,t)$
g:0.5$
n:200$
a:100$
b:0.01$
sol:rk(g*(n+a-z-n*exp(-b*z/g)),z,0,[t,0,10,0.1])$
len:length(sol)$
tt:makelist(sol[k][1],k,1,len)$
```

```
z1:makelist(sol[k][2],k,1,len)$
x1:makelist(n*exp(-b*z1[k]/g),k,1,len)$
y1:makelist(n+a-z1[k]-x1[k],k,1,len)$
wxplot2d([[discrete,tt,x1],[discrete,tt,y1],[discrete,tt,z1]],[style,
[lines,1,5],[lines,2,5],[lines,4,5]],[legend, "x", "y","z"],[xlabel,"t"], [yla-
bel,"x,y,z"]);
```

Задания к параграфу

С использованием программы **P1.2** провести исследование динамики развития эпидемии согласно модели (1.10) для β=0.01 и γ=0.01, 0.1, 0.2.

§2. Динамика плотности одиночной популяции

2.1. Модель неограниченной одиночной популяции

Томас Роберт Мальтус, 1766 – 1834, английский священник и учёный, демограф и экономист. Автор теории, согласно которой неконтролируемый рост народонаселения должен привести к голоду на Земле. 1798 – Essay on the Principle of Population («Опыт о законе народонаселения»).

Для построения математических моделей динамики численности популяций, как правило, используются различные гипотезы. Одна из простейших гипотез: скорость изменения численности популяции пропорциональна самой численности. На ее основе Мальтусом в 1798 г. была сформулирована модель неограниченной одиночной популяции:

$$\frac{dx}{dt} = rx\,, \qquad (2.1)$$

где x – численность популяции, t – время, $r = \alpha - \beta$, α – коэффициент рождаемости, β – коэффициент смертности. Коэффициент прироста r называют мальтузианским параметром. Задача Коши для уравнения (2.1) с начальным условием

$$x = x_0 \text{ при } t_0 = 0 \tag{2.2}$$

может быть решена аналитически (программа **P2.1**):

$$x = x_0 \, \mathrm{e}^{rt} \tag{2.3}$$

Зависимость $x(t)$ для различных коэффициентов прироста приведена на рис. 2.1. Для положительных значений r модель Мальтуса предсказывает неограниченный рост популяции.

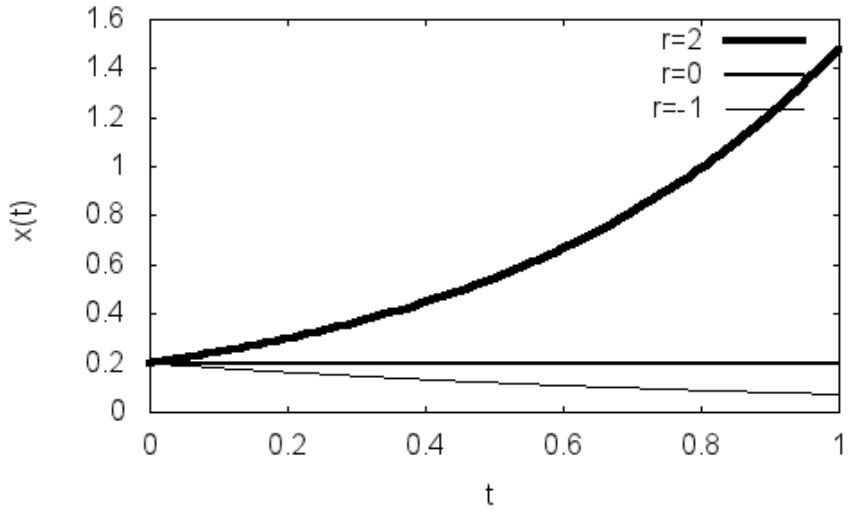


Рис. 2.1. Зависимость $x(t)$ при различных значениях r

Текст программы P2.1

```
kill(all)$
depends(x,t)$
eq:diff(x,t)=r*x$
sol:ode2(eq,x,t)$
sol:ic1(%,x=0.2,t=0)$
xt1:rhs(sol),r:2$
xt2:rhs(sol),r:0$
xt3:rhs(sol),r:–1$
wxplot2d([xt1,xt2,xt3],[t,0,1],[legend, "r=5","r=0","r=–1"],[style,
[lines,4,5],[lines,2,5],[lines,1,5]],[ylabel, "x(t)"],[xlabel, "t"]);
```

2.2. Модель ограниченной одиночной популяции

	**Пьер Франсуа Ферхюльст**, 1804 – 1849, бельгийский математик. Предложил модель ограниченной одиночной популяции.
	**Раймонд Пирл**, 1879 – 1940, американский биолог и математик. Pearl R. The biology of population growth, N.Y., 1925.

Модель неограниченной популяции может описывать динамику популяции в начальный период развития, когда популяция не испытывает нехватки пищевых ресурсов. В реальных условиях любой биологический вид существует в ограниченной пищевой среде. В 1838 г. бельгийским математиком Ферхюльстом и в 1928 г. американским биологом Пирлом модель Мальтуса была обобщена для случая ограниченной популяции.

Пусть коэффициент прироста r с увеличением плотности популяции уменьшается по линейному закону.

$$r \approx r_m - \gamma x$$

где r_m – коэффициент прироста при неограниченном количестве пищи и минимальной смертности, γ – коэффициент, показывающий насколько сильно уменьшается величина r с ростом численности популяции. Подставляем приведенное выражение в уравнение (2.1). Тогда получаем:

$$\frac{dx}{dt} = x(r_m - \gamma x) \tag{2.4}$$

Введя параметр $k = r_m / \gamma$ (емкость среды или равновесная плотность популяции), перепишем (2.4) в виде

$$\frac{dx}{dt} = r_m x\left(1 - \frac{x}{k}\right) \tag{2.5}$$

Уравнение (2.5) – обыкновенное дифференциальное уравнение с разделяющимися переменными. Разделим переменные в (2.5):

$$\frac{dx}{x^2 - kx} = -\frac{r_m}{k}\,dt \tag{2.6}$$

Интегрирование (2.6) с начальным условием (2.2) дает:

$$\ln \frac{(k - x)/x}{(k - x_0)/x_0} = -r_m t \tag{2.7}$$

Из (2.7) получим функцию $x(t)$ с параметрами k и r_m:

$$x = \frac{k}{1 + e^{-r_m t}(k - x_0)/x_0} \tag{2.8}$$

В решении модели динамики ограниченной популяции (2.8) наряду с коэффициентом прироста появляется новый параметр k – емкость среды или равновесная плотность популяции. Примеры зависимостей $x(t)$ для различных k, полученные по программе **P2.2,** даны на рис. 2.2. В отличие от модели Мальтуса численность популяции растет со временем до значения $x=k$, соответствующего равновесной плотности.

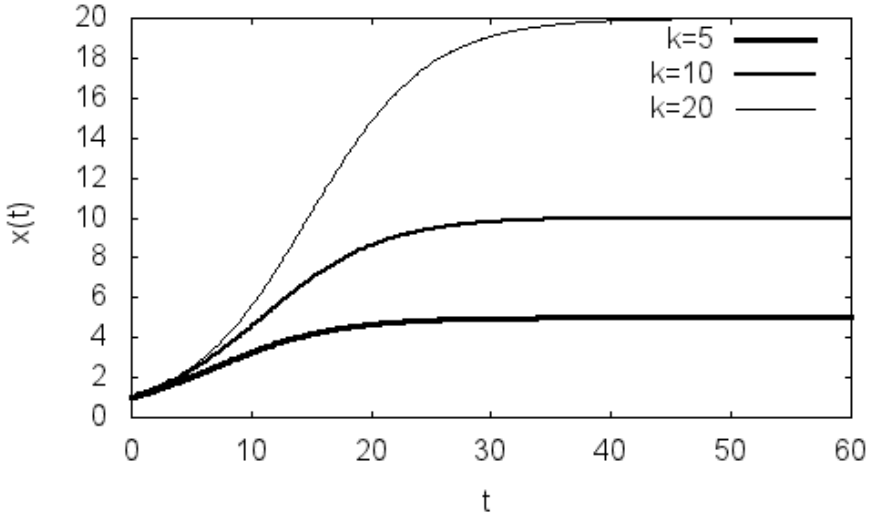


Рис. 2.2. Динамика плотности ограниченной одиночной популяции

Текст программы P2.2

```
k:10$
x0:1$
xa1:k/(1+(k-x0)*exp(-rm*t/x0)),rm:-0.1$
xa2:k/(1+(k-x0)*exp(-rm*t/x0)),rm:0.$
xa3:k/(1+(k-x0)*exp(-rm*t/x0)),rm:0.1$
wxplot2d([xa1,xa2,xa3],[t,0,100],[legend, "rm=-0.1","rm=0.","rm=0.1"],
[style, [lines,3,5],[lines,2,5],[lines,1,5]],[xlabel,"t,hour"],[ylabel,"x(t)"]);
```

Задания к параграфу

1. Найти аналитическое решение уравнения, описывающего неограниченный рост одиночной популяции (экспоненциальная модель):

$$\frac{dx}{dt} = rx, \tag{2.9}$$

где r – коэффициент естественного прироста плотности популяции, x – плотность популяции. В качестве начального условия для (2.10) принять

$$x = x_0 \quad \text{при} \quad t = t_0. \tag{2.10}$$

Построить графики функций $x=x(t)$ при $x_0 = 500$ и r=–0.5,0.25,0,25, 0.5.

2. Получить численное решение уравнения Ферхюльста – Пирла (логистическая модель)

$$\frac{dx}{dt} = xr_m\left(1-\frac{x}{k}\right) \tag{2.11}$$

с начальным условием (2.2) и сравнить с аналитическим решением (2.8). Построить графики функций $x=x(t)$ для значений параметров k, x_0 и r_m из таблицы 2.1.

Таблица 2.1

Значения параметров для задания 2 к § 2

x	1	2	3	4	5	6	7	8	9	10
k	100	200	300	400	500	600	700	800	900	1000
x_0	50	150	250	350	450	550	650	750	850	950
r_m	0.02	0.04	0.06	0.08	0.10	0.12	0.14	0.16	0.18	0.20

§3. Модели взаимодействия популяций: хищник–жертва

Альфред Джеймс Лотка, 1880 – 1949, американский математик. Один из первых ввел в биологию понятия физики.

Lotka A.J. Elements of physical biology. Baltimore: Williams and Wilkins, 1925, 460 p. (Элементы физической биологии).

	**Вито Вольтерра,** 1860–1940, итальянский математик, физик Volterra V. Lecóns sur la théorie mathematique de la lutte pour la vie. P.:Gauthiers–Villars, 1931 Вольтерра В. Математическая теория борьбы за существование. 1931 / Пер. с французского под ред. Ю.М.Свирежева. – М.: Наука, 1976.

Одной из классических задач математической экологии является модель популяционной системы «хищник–жертва», описываемая уравнениями Лотки и Вольтерры:

$$\begin{aligned} \frac{dx}{dt} &= r_1 x - \lambda_1 xy \\ \frac{dy}{dt} &= \lambda_2 xy - \beta_2 y \end{aligned} \qquad (3.1)$$

где x и y – плотности популяций жертвы и хищника, λ_1, λ_2 – коэффициенты, характеризующие скорость поедания жертвы хищником и обусловленную этим скорость изменения плотности хищника, β_2 – коэффициент смертности хищника, r_1 – коэффициент естественного прироста жертвы (без учета поедания ее хищником), $\lambda_2 = \gamma\lambda_1$, $\gamma -$ коэффициент показывающий, насколько увеличивается плотность популяции хищника при увеличении потребления пищи на единицу массы или численности. Уравнения (3.1), дополненные начальными условиями $x(0) = x_0, y(0) = y_0$, представляют собой задачу Коши для системы нелинейных обыкновенных дифференциальных уравнений. Такая задача может быть решена в программе Maxima методом Рунге–Кутты с помощью программы **Р3.1.**

Результатом выполнения приведенной программы будут зависимо-

сти численностей жертвы и хищника от времени (рис. 3.1). Наблюдается характерная периодическая динамика экологической системы «хищник–жертва», выражаемая в фазовой плоскости замкнутой кривой (рис. 3.2). Система уравнений (3.1) может быть дополнена новыми членами, учитывающими другие популяционные процессы, такие, например, как внутривидовая конкуренция. В случае учета внутривидовой конкуренции жертвы первое уравнение системы (3.1) примет вид

$$\frac{dx}{dt} = r_1 x - \lambda_1 xy - g_1 x^2 \qquad (3.2)$$

где g_1 – коэффициент внутривидовой конкуренции жертвы. Кривые $x(t)$ и $y(t)$ и фазовый портрет системы «хищник–жертва» при наличии внутривидовой конкуренции (g_1=0.0005) приведены на рис. 3.3 и рис. 3.4. Учет внутривидовой конкуренции меняет периодический характер поведения численности популяций жертвы и хищника. Наблюдаются затухающие колебания, отражением которых в фазовой плоскости является спиралевидная кривая.

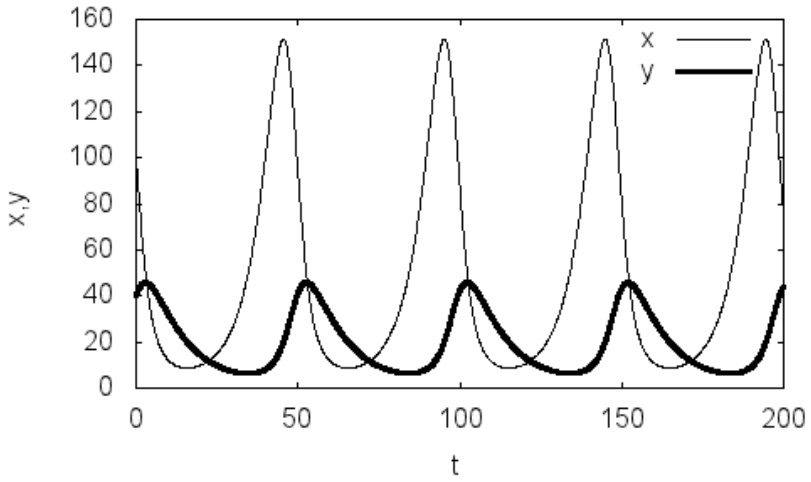


Рис. 3.1. Зависимости численностей популяций хищника и жертвы от времени

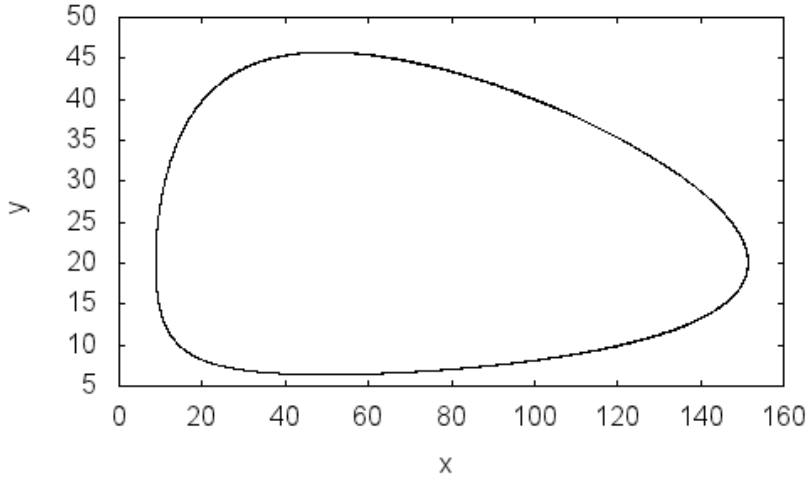


Рис. 3.2. Фазовый портрет системы «хищник–жертва»

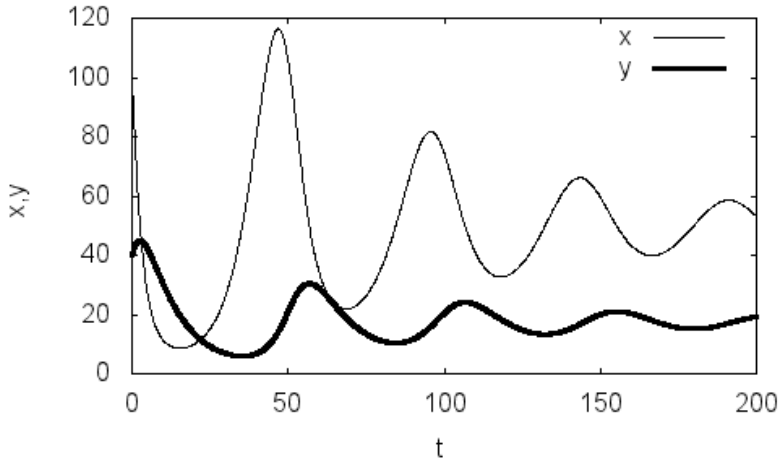


Рис. 3.3. Зависимости численностей популяций хищника и жертвы от времени с учетом внутривидовой конкуренции жертвы

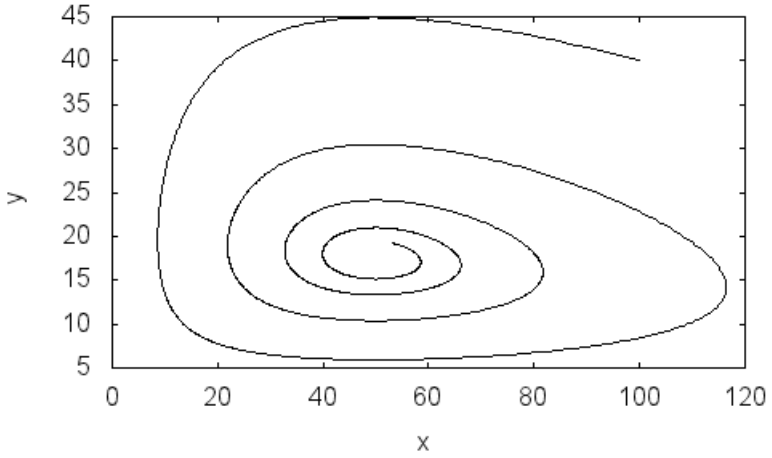


Рис. 3.4. Фазовый портрет системы «хищник–жертва» с учетом внутривидовой конкуренции жертвы

Текст программы P3.1

```
load("dynamics");
r1:0.2$
lb1:0.01$
gam:0.2$
lb2:gam*lb1$
bt:0.1$
sol:rk([r1*x–lb1*x*y,–bt*y+lb2*x*y],[x,y],[100.,40],[t,0,200,0.1]);
len:length(sol)$
tt:makelist(sol[k][1],k,1,len)$
n1:makelist(sol[k][2],k,1,len)$
n2:makelist(sol[k][3],k,1,len)$
wxplot2d([[discrete,tt,n1],[discrete,tt,n2]],[style, [lines,1,5], [lines,3,5]],
[legend, "x", "y"],[xlabel,"t"], [ylabel,"x,y"]);
wxplot2d([discrete,n1,n2],[xlabel,"x"], [ylabel,"y"],[style, [lines,1,5]]);
```

§4. Модель динамики биомассы микроорганизмов с учетом влияния освещенности

Циклические изменения условий окружающей среды воздействовали на биосферу в течение длительного времени. В результате многократных периодических воздействий выработался набор механизмов реагирования биосферы, которые сформировали ее устойчивость за все время существования. К таким периодическим внешним факторам относятся суточные колебания освещенности и температуры воздушной или водной среды обитания. Обеспечение биоты солнечной энергией осуществляется в циклическом режиме (день – ночь), когда освещенность меняется от нуля до максимума. Температура изменяется волнообразно в более узком диапазоне из-за влияния тепловой инерции земной поверхности или водной среды. Продолжительность дня и ночи измеряется часами, сезонные изменения протекают более медленно – в течение десятков суток. Амплитуда сезонного изменения среднесуточной освещенности меньше, чем суточного. В свою очередь температура в течение сезона варьируется в более широком диапазоне, так как тепловой инерции земной поверхности недостаточно

для длительного сглаживания температурных колебаний, обусловленных изменяющимся притоком солнечной энергии. Годовые циклы характеризуются еще более высокими изменениями внешних факторов для биосферы. В течение года могут проявиться экстремальные изменения факторов окружающей среды: сильные засухи, холодные зимы. Таким образом, с момента своего возникновения биосфера развивалась в условиях циклического изменения факторов внешней среды различных временных масштабов. Поэтому в биосфере сформировались сложные внутренние механизмы реагирования на суточные, сезонные и годовые циклы изменения условий окружающей среды.

Следуя работе [20], рассмотрим суточную динамику биомассы фототрофных микроорганизмов на основе модели Мальтуса неограниченной одиночной популяции:

$$\frac{dx}{dt} = (\mu - \varepsilon)x \tag{4.1}$$

где x – концентрация биомассы [г сух. вещества/л], μ – удельная скорость роста биомассы [$ч^{-1}$]; ε – удельная скорость расходования биомассы [$ч^{-1}$] за счет поедания растительноядными организмами и дыхания растений.

В общем случае величина μ зависит от концентрации субстратов, температуры среды, освещенности. Будем считать, что концентрация субстратов и температура воды в водоеме в течение суток изменяются несущественно, основным внешним фактором примем освещенность E [$Вт/м^2$].

Пусть продолжительность дня и ночи равны между собой (день равноденствия). Тогда освещенность E от момента восхода Солнца t=0 до его захода t=12 ч меняется по закону:

$$E = E_n \sin\left(2\pi t / 24\right) \tag{4.2}$$

где E_n – освещенность в полдень (максимальная), t – время (в часах). В период от 12 до 24 ч освещенность принята равной нулю E =0.

Полагая, что скорость процесса фотосинтеза пропорциональна осве-

щенности E, зависимость μ от E представим в виде гиперболической зависимости:

$$\mu = \mu_{\max} \frac{E}{K_E + E} \tag{4.3}$$

где K_E – коэффициент, равный освещенности, при которой μ=0.5μ_{max}. Величину ε будем считать не изменяющейся в течение суток. В начальный момент времени $x(0)=x_0$.

Примем следующие значения параметров математической модели: x_0=5 г/л, E_n=400 Вт/м^2, μ_{max}=0.3 ч$^{-1}$, K_E =100 Вт/м^2, ε=0.1 ч$^{-1}$. На рис. 4.1-4.3 приведены результаты расчетов по описанной модели (программа **P4.1**). На рис. 4.1 и рис. 4.2 даны зависимости $E(t)$ и $\mu(t)$ в течение трех суток. Периодический характер изменения освещенности вызывает периодическое изменение удельной скорости роста биомассы. В результате динамика роста биомассы также носит циклический характер (рис. 4.3). В дневное время, когда удельная скорость роста биомассы оказывается выше удельной скорости расходования биомассы, мальтузианский параметр модели положителен и наблюдается рост биомассы. В ночное время при нулевой скорости роста $\mu(t)$ правая часть уравнения (4.1) будет отрицательна, что приводит к уменьшению биомассы в результате ее расходования. С течением времени биомасса в среднем уменьшается. Это связано с принятым предположением о постоянстве скорости расходования биомассы ε.

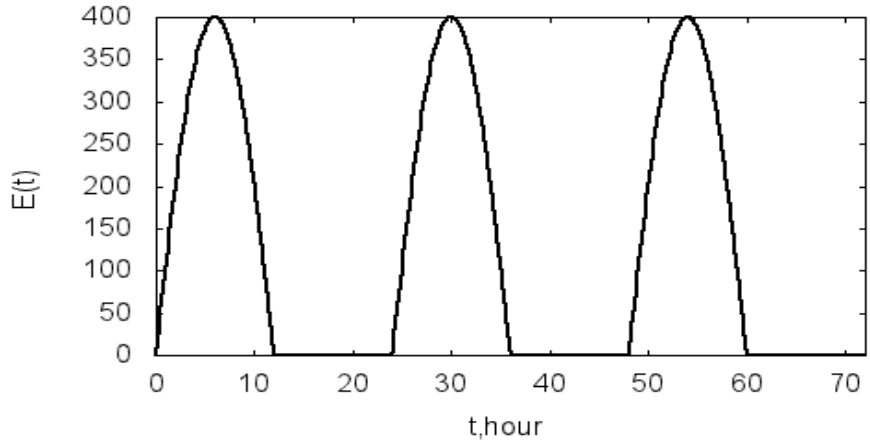


Рис. 4.1. Зависимость освещенности от времени

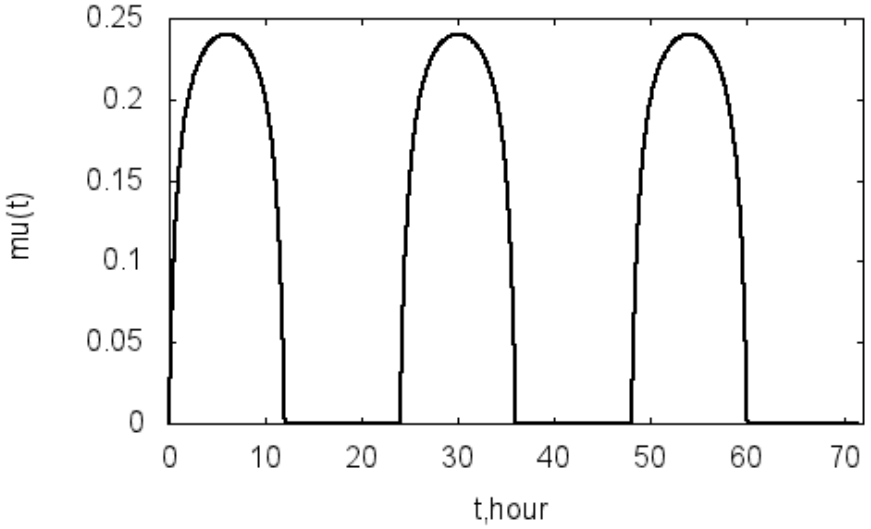


Рис. 4.2. Зависимость удельной скорости роста биомассы от времени

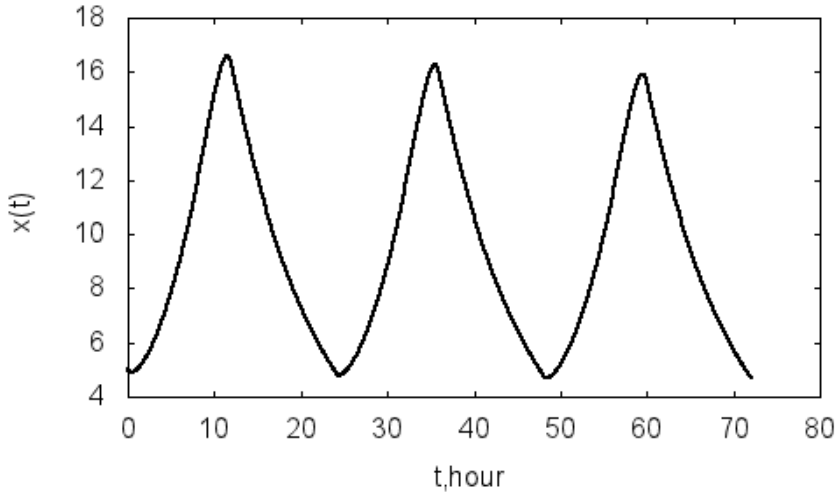


Рис. 4.3. Зависимость биомассы от времени

Текст программы P4.1

```
En:400$
mumax:0.3$
ke:100$
eps:0.1$
E:En*sin(2*%pi*t/24)$
Et:if t/24–floor(t/24)<0.5 then E else 0$
wxplot2d(Et,[t,0,72],[y,0,400],[xlabel,"t,h"],[ylabel,"E(t)"],[style,[lines,2,5]]);
mu:0.5*mumax;
mu:mumax*Et/(ke+Et)$
wxplot2d(mu,[t,0,72],[y,0,0.25],[xlabel,"t,h"],[ylabel,"mu"],[style,[lines,2,5]]
);
load("dynamics")$
```

```
eq:(mu–eps)*x$
sol:rk(eq,x,5,[t,0,72,0.1])$
len:length(sol);
tg:makelist(sol[k] [1],k,1,len)$
xg:makelist(sol[k] [2],k,1,len)$
wxplot2d([discrete,tg,xg],[xlabel,"t,hour"],[ylabel,"x(t)"],[style,[lines,2,5]]);
```

Задания к параграфу

Провести исследование динамики биомассы фототрофных организмов при различных значениях освещенности в полдень E_n, максимальной удельной скорости роста биомассы μ_{max}, коэффициенте K_E, удельной скорости расходования биомассы ε и длительности T_n.

Таблица 4.1.

Значения параметров модели (4.1)

Вариант	E_n, Вт/м^2	μ_{max}, ч$^{-1}$	K_E, Вт/м^2	ε, ч$^{-1}$	T_n, ч
1	300	0.05	200	0	48
2	400	0.1	100	0.1	72
3	500	0.2	300	0.3	96

§5. Дискретные модели популяций

В непрерывных моделях популяций численность или плотность расселения популяции считается непрерывной функцией времени и/или пространственных координат. В реальности численность популяции представляет собой дискретную величину, которая принимает определенные значения в фиксированные моменты времени. Дискретные значения численности популяции могут быть получены из экспериментальных данных по изучению реальных популяций (лабораторных или полевых) в дискретные моменты времени. Если при этом предположить, что численность популя-

ции x_n в момент времени t зависит от численностей в некоторые предшествующие моменты времени, то для описания динамики численности популяций можно применять аппарат разностных уравнений.

	**Леонардо Фибоначчи,** около 1180–1240, итальянский математик «Трактат о счете» (1202 год)

Исторически первой дискретной моделью биологической популяции в математической экологии принято считать модель динамики популяции в виде ряда чисел

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, \ldots \qquad (5.1)$$

приведенную в книге «Трактат о счете» (1202 год) итальянского ученого Леонардо Фибоначчи. Числовая последовательность (5.1) вошла в историю как ряд чисел Фибоначчи, а его члены – числа Фибоначчи. Числа в последовательности (5.1) выражают динамику роста численности кроликов от поколения к поколению согласно простому правилу: каждый последующий член последовательности равен сумме двух предыдущих. Приведенная модель выражает гипотезу, что количество воспроизводимых кроликов в данном поколении равно сумме кроликов в двух предыдущих поколениях.

В настоящее время дискретные модели широко применяются для исследования динамики популяций и относятся к важной группе математи-

ческих моделей в экологии. В той или иной степени дискретные модели популяций описаны в работах[6,8,9,11,12,14,15,21], которые были использованы при написании данного пособия.

В дискретных моделях время представляется как дискретная переменная, и наблюдения выполняются лишь через определенные фиксированные интервалы времени (ежечасно, ежегодно, через каждые 10 лет и т.п.). Введем дискретную переменную x_n, представляющую собой численность популяции к концу n-го периода времени. Тогда $x_0, x_1, x_2, \ldots, x_n$, $x_{n+1} \ldots$ – последовательность чисел, описывающая развитие популяции во времени. На практике обычно известна начальная численность популяции и скорость роста популяции в разные периоды времени. Для определения численности популяции x_n вводится понятие разностного уравнения.

Определение. Разностное уравнение – уравнение, которое связывает между собой значения численности популяции x_n при различных значениях индекса n. Если N_1 и N_2 представляют собой наибольший и наименьший из индексов n, встречающихся в записи уравнения, то порядок разностного уравнения будет равен $N_1 - N_2$. Различают линейные и нелинейные разностные уравнения. Разностные уравнения, содержащие x_n, x_{n+1} и т.д. в степени выше первой и/или их произведения, являются нелинейными.

Аппарат разностных уравнений широко используется в вычислительной математике при определении дискретных значений функций, получаемых в результате дискретизации задачи для дифференциальных уравнений, описывающих различные физические процессы. Разностные уравнения в задачах математической экологии могут быть записаны как математические выражения различных гипотез, формулируемых относительно динамики численности популяций.

5.1. Дискретная модель неограниченной одиночной популяции

Пусть скорость роста популяции в период времени n пропорциональна размеру популяции в начале этого периода

$$\Delta x_n = x_{n+1} - x_n = a\ x_n \qquad (5.2)$$

Тогда численность популяции в следующий момент времени определится по формуле

$$x_{n+1} = (1+a)\, x_n . \qquad (5.3)$$

Согласно (5.3) можно записать:

$$x_1 = (1+a)\, x_0$$

$$x_2 = (1+a)\, x_1 = (1+a)(1+a)\, x_0 = (1+a)^2\ x_0$$

$$x_3 = (1+a)\, x_2 = (1+a)(1+a)(1+a)\, x_0 = (1+a)^3\ x_0$$

….

$$x_n = (1+a)^n\ x_0 .$$

При известном начальном значении x_0 можно рассчитать динамику популяции во времени. В зависимости от коэффициента роста $r = a+1$ возможны следующие ситуации:

1) $a > 0$, $(1+a) > 1 \Rightarrow (1+a)^n \to \infty$ при $n \to \infty$ – неограниченный рост;

2) $a = 0$, $1+a = 1$ – численность популяции не меняется;

3) $-1 < a < 0$, $0 < 1+a < 1 \Rightarrow x_n \to 0$ – вымирание популяции;

4) $a = -1$ – вымирание за один период времени;

5) $a < -1$ – отрицательные численности (нереальная ситуация).

Уравнение (5.3) обычно записывается в форме

$$x_{n+1} = r\ x_n \tag{5.4}$$

Уравнение (5.4) называется дискретным аналогом модели одиночной неограниченной популяции.

5.2. Дискретная модель ограниченной популяции: логистическое уравнение

Одним из классических примеров дискретных моделей является логистическое разностное уравнение для одиночной ограниченной популяции. Получаемая при этом динамика системы включает в себя все многообразие типов поведения реальных популяционных систем, как простых или упорядоченных, так и сложных или хаотических. Переход от простого поведения к хаосу обладает схожими закономерностями, присущими различным моделям популяций или различным сложным динамическим системам. Следуя знаменитой работе [18], исследуем разностную модель, задаваемую логистическим уравнением.

Как было отмечено ранее, дискретная модель (5.4) при коэффициенте прироста $r > 1$ предсказывает неограниченный рост численности популяции. В реальности ни одна популяция не может увеличиваться бесконечно вследствие ограниченности пищевых ресурсов и других ограничивающих внешних факторов. Для учета этого обстоятельства введем условие ограничения роста. Пусть коэффициент прироста r будет зависеть от численности популяции, а именно, будет убывать по мере роста численности популяции по закону $r \sim r(1 - x_n)$. Тогда уравнение (5.4) примет вид

$$x_{n+1} = x_n r(1 - x_n),\ n = 0, 1, 2... \qquad (5.5)$$

Уравнение (5.5), называемое логистическим уравнением или дискретным аналогом модели Ферхюльста–Пирла, может описывать не только динамику популяций, но и многие другие явления в природе, экономике и обществе. Отметим, что величина x в уравнении (5.5) меняется от 0 до 1, а r – от 0 до 4. При других значениях x и r логистическое уравнение дает отрицательные значения численности популяции.

Задавая различные значения параметра r естественной скорости роста и начальной численности x_0 популяции, можно получить качественно различные типы поведения переменной, удовлетворяющей разностному уравнению (5.5). Разностное уравнение наряду с равновесием и циклами может иметь хаотические решения, не стремящиеся ни к какому притягивающему решению.

Приведем результаты численных исследований модели (5.5) по программе **Р5.1.** На рис. 5.1 показаны графики зависимости x_n от номера периода времени для разных значений параметра r $(x_0 = 0.2)$. Для значения $r = 0.5$ популяция за несколько периодов времени приходит к вымиранию (нулевое значение x_n при $n \to \infty$). При $r = 2.4$ получаем единственное равновесное значение численности $x_n = 0.5833$. Из рис. 5.1 видно, что при $r = 3.33$ конечное значение численности популяции начинает осциллировать между двумя уровнями, которые соответствуют значениям 0.829635 и 0.470666, то есть реализуется периодический режим – цикл с периодом 2. С ростом r динамика системы усложняется.

Для $r = 3.5$ динамика системы характеризуется устойчивыми периодическими колебаниями с периодом 4 (установившиеся значения численности x_n=0.874997; 0.500887; 0.826939; 0.382818). И, наконец, при $r = 3.9$ можно наблюдать, что процесс перестает быть периодическим. При увеличении значения номера периода времени численность популяции принима-

ет новые неповторяющиеся значения. Такое поведение называется нерегулярным или хаотическим.

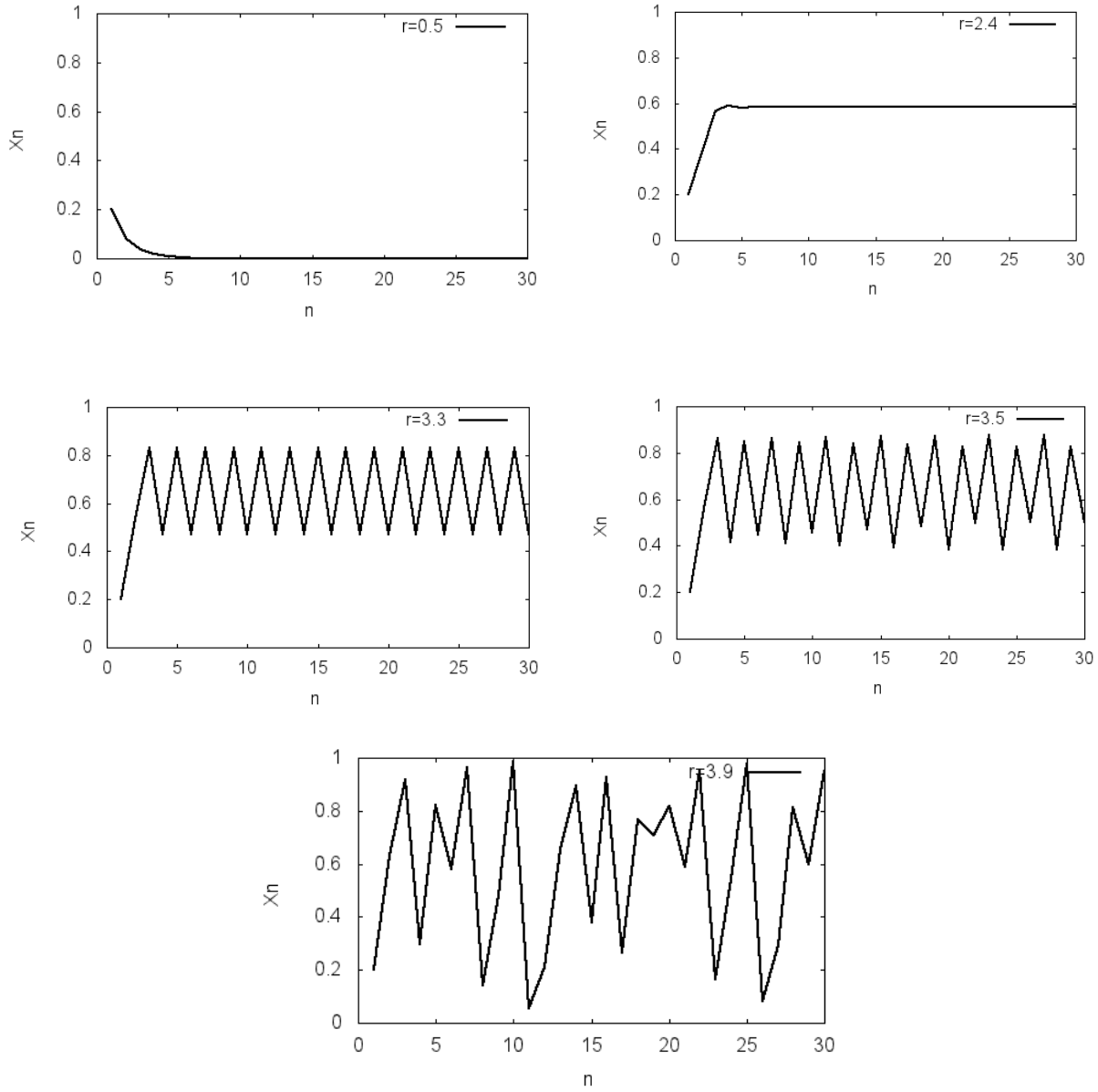


Рис. 5.1. Зависимость x_n от периода времени *n* для различных значений параметра r

Проанализируем подробнее причину появления такого поведения в этой, казалось бы, простой модели. Перепишем формулу (5.5) в виде

$$x_{n+1} = f(x_n),\ n = 0, 1, 2... \qquad (5.6)$$

Природа столь сложного поведения решения логистического уравнения заключена в нелинейности функции $f(x) = rx(1-x)$ в (5.6), которая явля-

ется квадратичной функцией *x*. Равновесным решением или неподвижной точкой уравнения (5.6) называется решение вида $x_n = x^* = const$, удовлетворяющее соотношению

$$x^* = f(x^*) \tag{5.7}$$

Построим графики функций $y = f(x)$ и $y = x$ при различных значениях *r* (рис. 5.2). В точках пересечения графиков $x = f(x)$, т.е. точки пересечения являются неподвижными точками. Для случая $r = 0.5$ графики пересекаются только в одной точке в начале координат $x = 0$, и мы имеем единственное нулевое предельное значение последовательности x_n в диапазоне $0 < r < 1$. При $r = 1$ происходит первая бифуркация, появляются два новых решения или решение удваивается. Наряду с $x^* = 0$ при $r \geq 1$ появляется решение $x^* = (r-1)/r$. Появляются две точки пересечения прямой $y = x$ и функции $y = f(x)$. Оба этих решения легко получаются из уравнения

$$x^* = x^* r(1 - x^*) \tag{5.8}$$

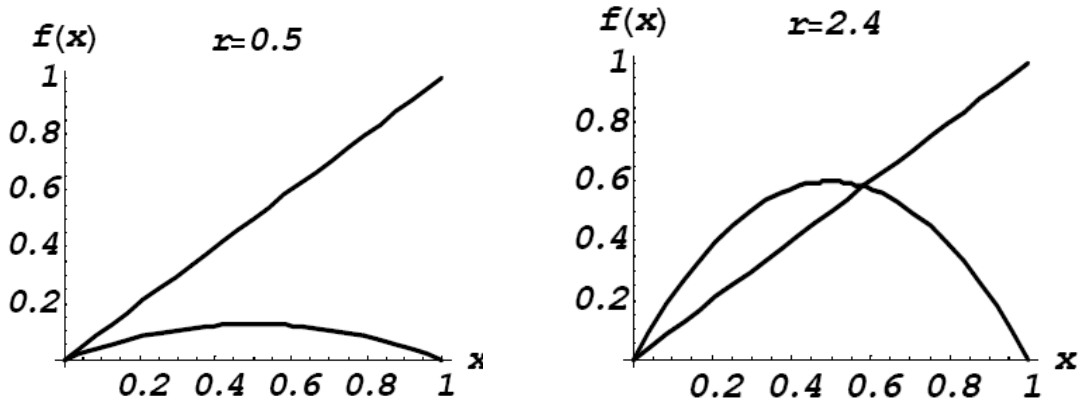


Рис. 5.2. Графики функций $y = x$ и $y = f(x)$

Неподвижная точка или точка равновесия может быть устойчивой и неустойчивой. Иначе говоря, итерации точки x_n могут и приближаться к

точке x^*, и удаляться от нее. Устойчивость процесса (5.6) зависит от угла наклона кривой $f(x)$ в неподвижной точке. Если угол наклона с осью x не превышает по модулю $\pi/4$, то неподвижная точка является устойчивой. Это означает также, что для устойчивой неподвижной точки производная функции $f(x)$ меньше единицы по модулю. Учитывая, что производная $f(x)$ равна

$$\frac{df}{dx} = r(1-2x),$$

получим, что неподвижная точка становится неустойчивой при $r = \pm 1/\left|1-2x^*\right|$. Таким значением является $r = 3$, при котором появляется новая бифуркация, то есть решение еще раз удваивается. Неподвижные точки для цикла с периодом 2 (рис. 5.1, $r = 3.33$) определяются уравнением

$$x^* = f(f(x^*)), \text{ или}$$

$$x^* = r^2 x^*(1-x^*)[1-rx^*(1-x^*)] \qquad (5.9)$$

На рис. 5.3 показан график функции $f(f(x))$. Видно, что в этом случае мы имеем четыре точки пересечения прямой $y = x$ и функции $y = f(f(x))$, а следовательно, четыре неподвижных точки, две из которых являются устойчивыми. Эти четыре точки являются корнями алгебраического уравнения четвертого порядка (5.9). При некотором новом значении r произойдет новое удвоение решения логистического уравнения, и мы будем иметь цикл с периодом 4 (рис. 5.1, $r = 3.5$), и т.д. Если производная по модулю в какой-либо точке становится больше 1, неподвижная точка расщепляется на две и возникает новый устойчивый цикл. Поэтому процесс удвоения периода будет происходить до бесконечности.

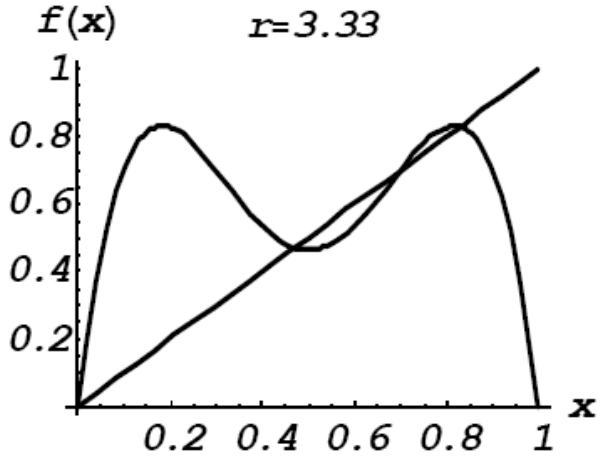


Рис.5.3. Графики функций $y = x$ и $y = f(f(x))$

Для наблюдения всех режимов поведения системы удобно строить бифуркационную диаграмму, представляющую собой зависимость равновесных значений популяции x^* от параметра r. Бифуркационная диаграмма для модели (5.5) приводится на рис. 5.4. На рис. 5.4 отчетливо прослеживается описанный выше порядок перехода к хаотическому поведению.

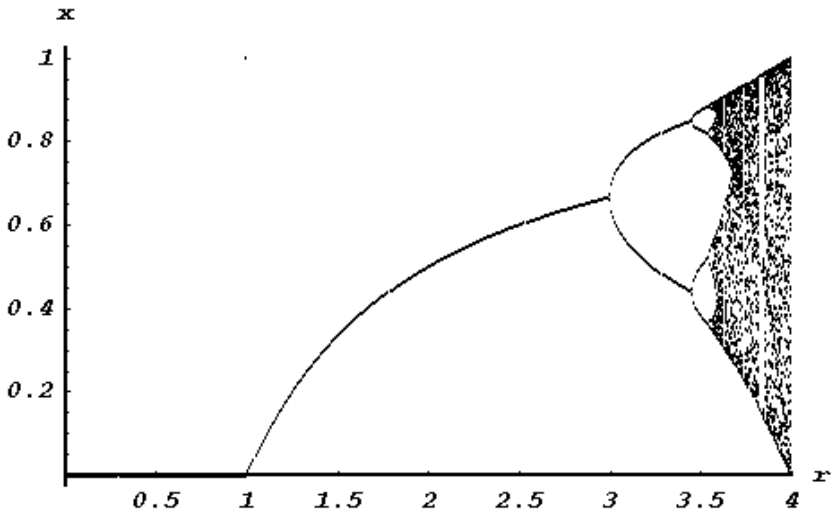


Рис. 5.4. Бифуркационная диаграмма для логистического уравнения (5.5)

	**Митчелл Фейгенбаум**, родился в 1944 г., американский математик. Открыл в 1976 переход к хаотическому поведению динамической системы через каскад удвоения периода. Открыл универсальную постоянную, названную его именем.

В работе [18] Фейгенбаумом было обнаружено, что последовательность значений параметра $\{r_n\}$, соответствующая точкам бифуркации (точкам удвоения решения), удовлетворяет соотношению

$$\lim_{n\to\infty} \frac{r_n - r_{n-1}}{r_{n+1} - r_n} = \delta = 4.6692... \qquad (5.10)$$

Фейгенбаум показал, что то же самое число δ возникает и в других процессах, отличных от процесса (5.5), и что это число является универсальной характеристикой сценария удвоения периода для целого класса одномерных дискретных моделей типа (5.5). Число δ носит название «числа Фейгенбаума». Обнаружение такого замечательного числа, присущего различных процессам, привело к высокой активности ученых в различных областях науки. Было поставлено множество вычислительных экспериментов, показавших, что сценарий удвоения периода действительно появляется во многих физических системах. Это и формирование турбулентности в потоке жидкости, и нелинейные колебания в электрических цепях и химических и биологических системах. Все указанные процессы имеют следующую общую черту: по мере изменения одного из параметров модели поведение системы меняется от простого к хаотическому. Причем переход идет по вполне закономерному сценарию через череду бифурка-

ций динамической системы.

Текст программы P5.1

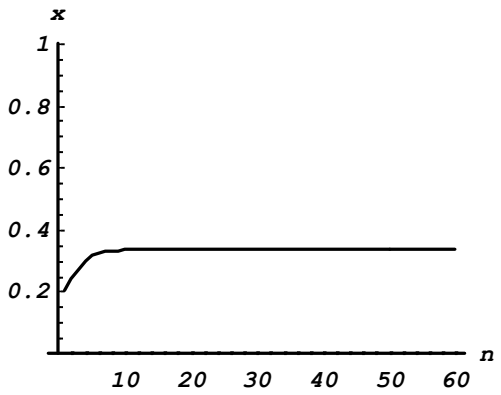
```
nx:30;
x:makelist(0.2,n,1,nx)$
r:0.5$
for n: 1 thru nx–1 do x[n+1]:r*x[n]*(1–x[n]);
xy:makelist([n,x[n]],n,1,nx)$
wxplot2d([discrete,xy],[y,0,1],[xlabel,"n"],[ylabel,"Xn"],[style,[lines,2,5]],[le
gend, "r=0.5"]);
```

5.3. Динамика одиночной популяции с учетом запаздывания

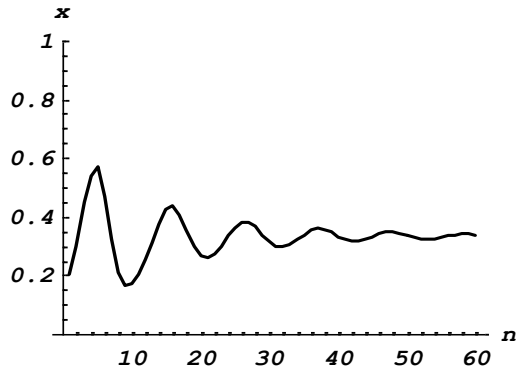
Модель (5.5) включает в себя внутривидовую конкуренцию за пищевые ресурсы, которая начинает сказываться на динамике популяции с первого момента времени. На самом деле существует некоторое время, в течение которого популяция может почувствовать нехватку пищевых ресурсов. В популяционной теории рассматриваются различные модели с учетом такого запаздывания. Запаздывание может быть учтено в рамках как непрерывных, так и дискретных моделей. Введем время τ запаздывания в (5.5)

$$x_{n+1} = x_n r(1 - x_{n-\tau}), \; n = 0, 1, 2... \; , \tau = 0,1,2... \qquad (5.11)$$

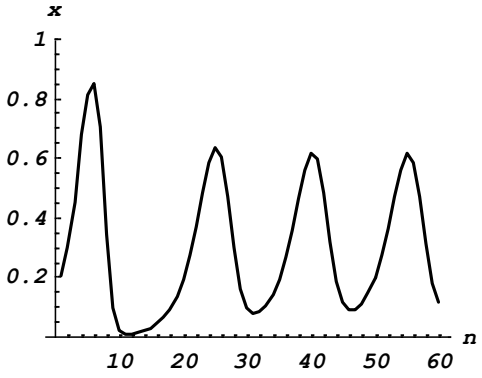
Результаты расчетов численности популяции при различных временах запаздывания τ, задаваемого в числе периодов времени, приведены на рис. 5.5. Из рисунков наглядно видно, что динамика системы существенно меняется при учете времени запаздывания. Численность популяции стремилась к постоянному значению при возрастании n для $\tau = 0$. Учет времени запаздывания приводит к появлению затухающих колебаний переменной x_n, при этом начальная амплитуда колебаний выше при больших временах запаздывания. Описанное поведение согласуется с поведением процессов в технических системах с обратной связью при учете запаздывания.



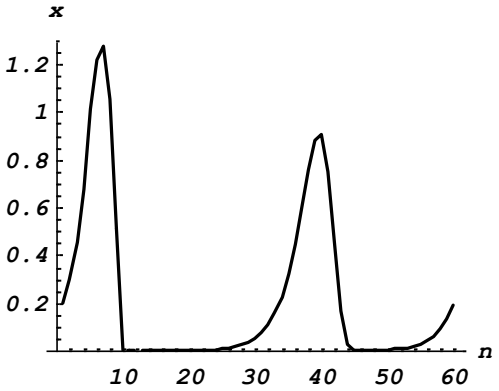
$\tau = 0$



$\tau = 2$



$\tau = 3$



$\tau = 4$

Рис. 5.5. Динамика численности популяции согласно модели (5.11) при различных временах запаздывания

5.4. Модель Рикера

Рассмотрим модель ограниченной популяции (модель Ферхюльста–Пирла)

$$\frac{dx}{dt} = rx\left(1 - \frac{x}{k}\right) \tag{5.12}$$

где x – численность популяции, k – равновесное значение численности, r – коэффициент прироста. Заменим производную в (5.12) dx/dt на $\Delta x/\Delta t = (\mathrm{x}_{t+1} - x_t)/\Delta t$. Полагая $\Delta t = 1$ из (5.12), получим разностное уравнение вида

$$x_{t+1} = x_t[1 + r(1 - x_t / k)] = x_t f(x_t) \qquad (5.13)$$

Уравнение (5.13) при $x_t > k(1+r)/k$ дает отрицательные значения численности популяции. Это связано с видом функции $f(x) = 1 + r(1 - x/k)$ (рис.5.6).

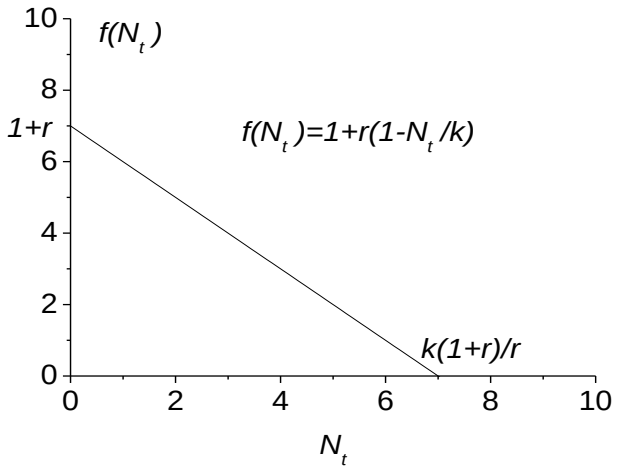


Рис. 5.6. Вид функции $f(x) = 1 + r(1 - x/k)$

В теории популяций часто используют исправленную модель одиночной ограниченной популяции, а именно модель Рикера, в которой в качестве функции $f(x_t)$ выбрана функция $f(x_t) = \exp[r(1 - x_t / k)]$, обеспечивающая асимптотическое стремление к нулю x_t при $t \to \infty$ (рис. 5.7):

$$x_{t+1} = x_t \exp[r(1 - x_t / k)] \qquad (5.14)$$

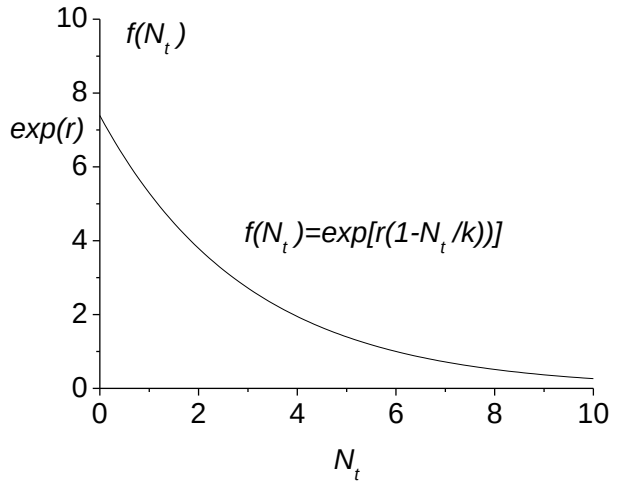


Рис. 5.7. Вид функции $f(x_t) = \exp[r(1 - x_t / k)]$

Опишем результаты численных исследований динамики популяции согласно модели Рикера (5.14) (программа **P5.2**). Начальное значение равновесной численности популяции во всех расчетах принято равным 0.1. Приведенные на рис. 5.8 зависимости численности популяции от времени дают представление о динамике системы в рамках модели Рикера при варьировании коэффициента прироста r. При r=0.5 численность популяции монотонно возрастает до максимума своей численности и остается дальше неизменной на этом уровне. При r=1.8 изменения численности представляют собой затухающие колебания. Причем, чем ближе значения r к 2, тем медленнее происходят затухания. При r=2.5 наблюдаются двухточечные циклы. При r=2.6 появляются циклы из четырех последовательно повторяющихся значений. И, наконец, при r>3 можно наблюдать, что процесс перестает быть периодическим. При увеличении значения номера периода времени численность популяции принимает новые неповторяющиеся значения. Такое поведение называется нерегулярным или хаотическим. Таким образом, численные расчеты по модели Рикера выявляют спектр типов поведения динамики численности популяции: выход на стабильный уровень, периодические циклы и хаотическое поведение. Такое многообразие решений в модели Рикера обусловлено нелинейностью модели.

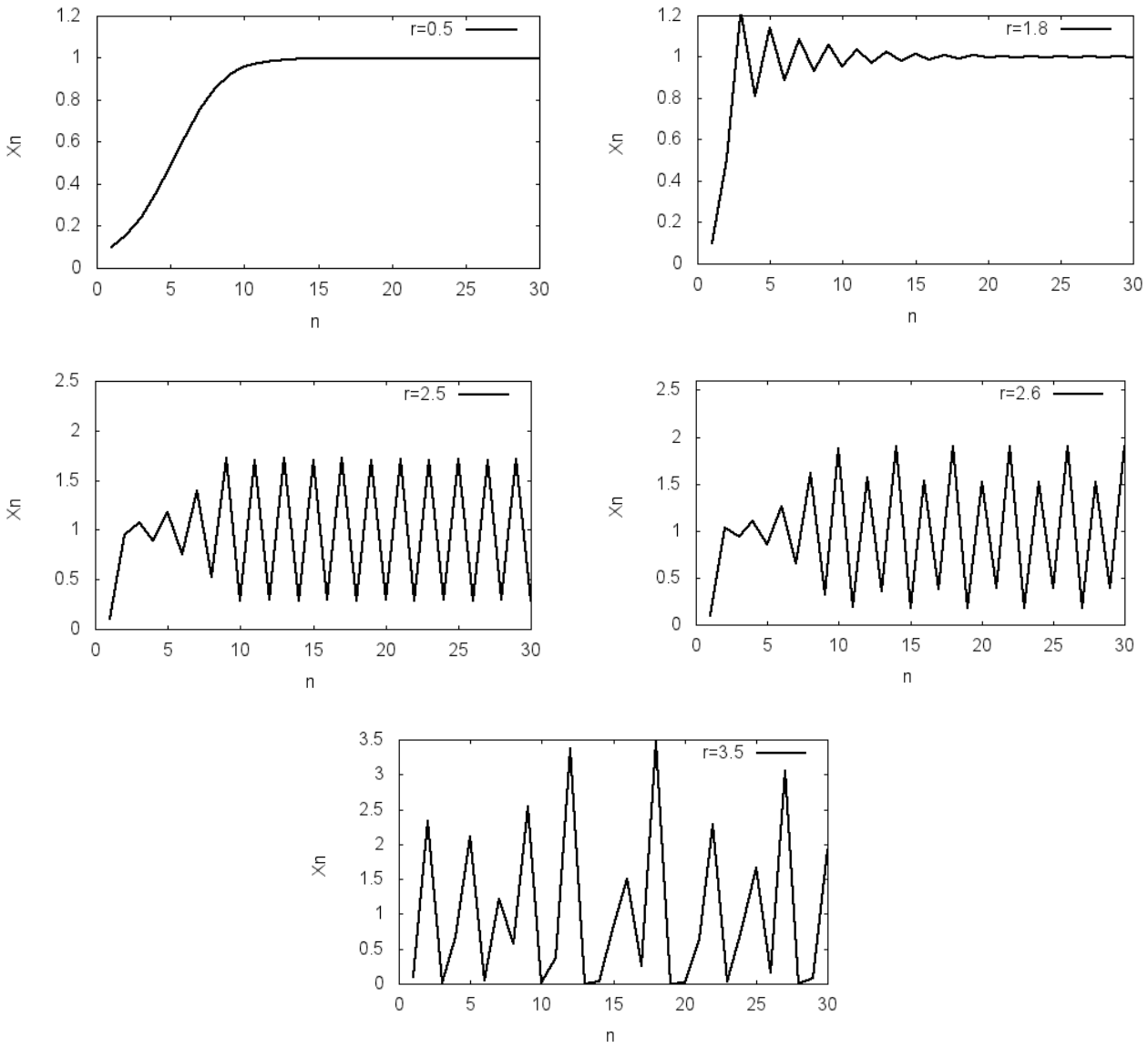


Рис. 5.8. Зависимость численности популяции от времени для модели Рикера

Текст программы P5.2

```
nx:30;
x:makelist(0.1,n,1,nx)$
r:2.6$
k:1$
for n: 1 thru nx–1 do x[n+1]:x[n]*exp(r*(1–x[n]/k));
xy:makelist([n,x[n]],n,1,nx)$
wxplot2d([discrete,xy],[y,0,r],[xlabel,"n"],[ylabel,"Xn"],[style,[lines,2,5]],[leg
end, "r=2.6"]);
```

5.5. Дискретная модель популяции с учетом возрастной структуры

Для учета возрастной структуры вводятся модели популяций с подразделением на дискретные возрастные классы (или стадии развития), численности которых зависят от численностей предшествующих (а в отдельных случаях, и всех остальных) возрастных классов. Задача описания динамики возрастных классов таких популяций приводит к дискретным матричным моделям или к системе разностных уравнений.

В [19] предложена и исследована модель динамики численности для популяции с возрастной структурой, которая может быть представлена совокупностью двух возрастных классов: младшего, включающего неполовозрелых особей, и старшего, состоящего из особей, участвующих в размножении. Обозначим численность младшего возраста в n–й сезон размножения через x_n, а численность репродуктивного поколения через y_n. Период размножения заканчивается появлением новорожденных особей нового поколения. Предполагается, что времени между двумя последовательными периодами размножения достаточно для полного развития младенцев до взрослого состояния, а новорожденных – до состояния младшего возраста. Коэффициенты выживаемости и плодовитости зрелых особей считаются постоянными. Принятое предположение характерно для организмов с небольшим периодом жизни, включающим два–три периода размножения: насекомые, рыбы, мелкие млекопитающие, двух – трехлетние растения и др.

Дискретная модель двухвозрастной популяции представляется в виде двух разностных уравнений

$$\begin{aligned} x_{n+1} &= by_n, \\ y_{n+1} &= x_n(1-x_n) + cy_n, \end{aligned} \tag{5.15}$$

где b – произведение коэффициентов рождаемости и выживаемости приплода на первом году жизни, а c – выживаемость половозрелых особей. В работе [19] показано, что все множество допустимых значений параметров b и с ($b>0$, $0<c<1$) системы (5.15) можно разбить на три области:

1) $b+c<1$ – в этой области для (5.15) существует только устойчивое положение нулевого равновесия $x=0$, $y=0$;

2) $b+c>1$, $b+2c<3$ – существует устойчивое ненулевое положение равновесия;

3) $b+2c>3$ – существуют неустойчивые нулевая и ненулевая стационарные точки системы (5.15).

Как и в [19], проводились численные исследования поведения системы (5.15) при $n \to \infty$ в области значений b,c, удовлетворяющих условию $b+2c>3$ (программа **P5.3**). Для расчетов, результаты которых приведены ниже, выбиралось $c=0.15$, а значение параметра b варьировалось, начальные значения численности двух поколений принимались равными $x_0=0.2$, $y_0=0.1$. Рассчитаны численности поколений в зависимости от времени и построены траектории системы в фазовом пространстве (x, y). Представление о динамике численности неполовозрелых и половозрелых особей дает рис. 5.9. Наблюдаются нерегулярные колебания численностей с изменением периода времени. Аттракторы системы (5.15) для значений $b=2.8;3.10;\ 3.22;\ 3.31$ приводятся на рис. 5.10. Расчеты проводились до $n=10000$, для представления на графике сохранялись последние 9000 значений численности популяции. Приведенные рисунки показывают равновесные состояния системы в фазовом пространстве, к которым стремятся переменные x,y при $n \to \infty$. По мере изменения параметра b характер линий значительно меняется. При $b=2.8$ аттрактор системы (5.15) представляет собой замкнутую кривую в фазовом пространстве – предельный цикл. С увеличением параметра b замкнутая кривая трансформируется в области со сложной структурой. Все линии утолщаются, и постепенно множество точек траектории более или менее плотно закрывает некоторую область

фазового пространства (рис. 5.10). Такое поведение системы позволяет предположить наличие сложной серии бифуркаций аттракторов. Таким образом, модель двухвозрастной популяции (5.15), как и модели одновозрастной популяции (5.5), (5.14), содержит в себе разнообразие поведения численности одной изолированной локальной популяции.

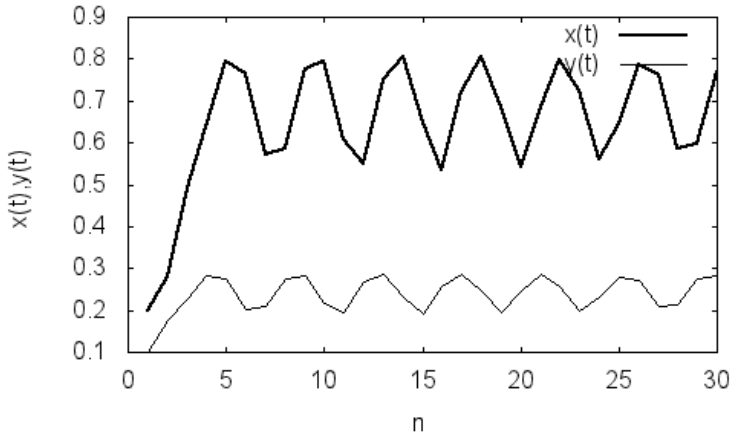
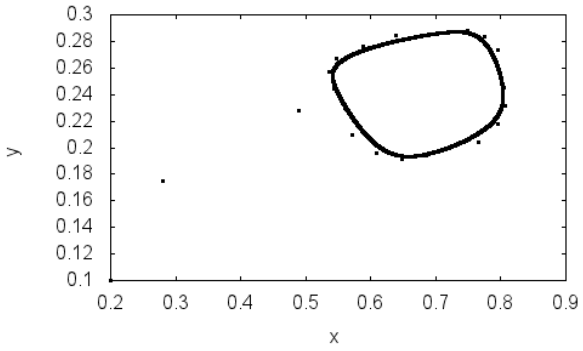
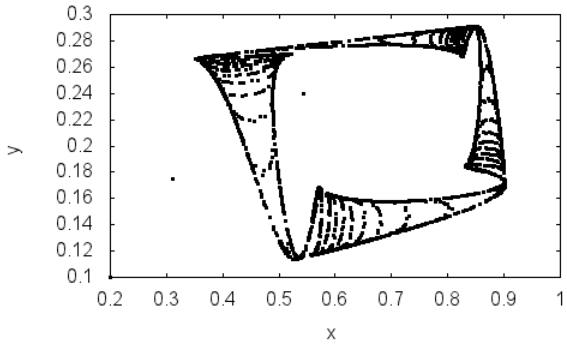


Рис. 5.9. Динамика численности двух поколений популяции согласно модели (5.15) при c=0.15 и b=2.8

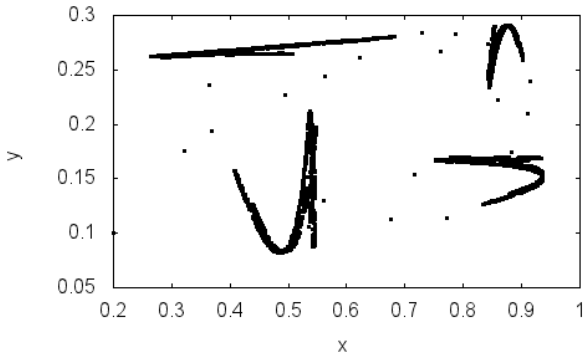
Из результатов анализа двух моделей (5.14) и (5.15) можно сделать один общий вывод: даже самые простые детерминированные дискретные модели динамики одиночных популяций могут приводить к сложному поведению, характеризуемому циклическими или нерегулярными хаотическими колебаниями численности популяции. В математическом плане причиной появления периодических или нерегулярных решений является нелинейность моделей. Косвенно параметры модели могут учитывать воздействие различных внешних факторов. Но даже при постоянных параметрах динамика рассматриваемых систем содержит различные колебания, в том числе и нерегулярные. То есть сложное поведение в динамической системе может быть связано с внутренней сущностью системы и может проявиться в отсутствие влияния внешних факторов.



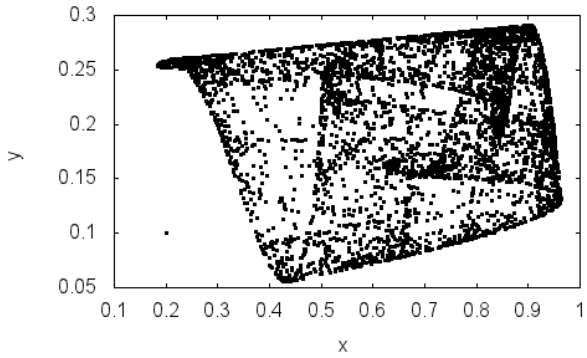
b=2.8



b=3.10



b=3.22



b=3.31

Рис. 5.10. Аттрактор системы (5.15) при c=0.15 и различных *b*

Текст программы P5.3

```
b:2.8$
c:0.15$
n1:30$
n:5000$
x:makelist(0.2,i,1,n)$
y:makelist(0.1,i,1,n)$
for i:1 thru n–1 do [x[i+1]:b*y[i],y[i+1]:x[i]*(1–x[i])+c*y[i]];
xx:makelist([i,x[i]],i,1,n1)$
yy:makelist([i,y[i]],i,1,n1)$
wxplot2d([[discrete,xx],[discrete,yy]],[style,[lines,2,5],[lines,1,5]]
,[legend,"x(t)","y(t)"],[ylabel, "x(t),y(t)"],[xlabel, "n"]);
xy:makelist([x[i],y[i]],i,1,n)$
wxplot2d([discrete,xy],[style,[points,0.4,5,6]]);
```

Задания к параграфу

1. Исследовать динамику популяции, задаваемую дискретными моделями. Построить зависимости $x_n = f(n)$ при варьировании параметров задачи.

Модель Рикера (1954)

$$x_{n+1} = x_n e^{-r(1-x_n/K)},\ n = 0, 1, 2... \tag{5.16}$$

Принять емкость среды равной $K = 100; 200$, коэффициент прироста $r = 0.1; 0.5; 1; 1.9; 2; 2.3; 2.6; 3.$

Модель Хасселя (1976)

$$x_{n+1} = \lambda x_n (1 - a x_n)^{-\beta},\ n = 0, 1, 2... \tag{5.17}$$

Принять $\lambda \in [1,1000]$, $\beta \in [1,10]$, $a = 1$.

2. Исследовать динамику двух популяций, задаваемую дискретными моделями. Построить зависимости $x_n = f_x(n)$, $y_n = f_y(n)$ и фазовые портреты в плоскости (x, y).

Модель хищник–жертва

$$\begin{aligned} x_{n+1} &= x_n + a_1 x_n - b_1 x_n y_n, \\ y_{n+1} &= y_n - a_2 y_n + b_2 x_n y_n \end{aligned} \tag{5.18}$$

Принять в качестве базовых значений $x_0 = 26,\ y_0 = 7$, $a_1 = 0.01, a_2 = 0.02,\ \ b_1 = 0.002, b_2 = 0.001.$

Модель конкуренции двух видов

$$\begin{aligned} x_{n+1} &= x_n + a_1 x_n - b_1 x_n y_n, \\ y_{n+1} &= y_n - a_2 y_n + b_2 x_n y_n \end{aligned} \tag{5.19}$$

Принять в качестве базовых значений $x_0 = 26,\ y_0 = 7$, $a_1 = 0.01, a_2 = 0.02,\ \ b_1 = 0.002, b_2 = 0.001.$

Модель конкуренции трех видов

$$
\begin{aligned}
x_{n+1} &= x_n + a_1 x_n - a_2 x_n y_n - a_3 x_n z_n, \\
y_{n+1} &= y_n + b_1 y_n - b_2 y_n x_n - b_3 y_n z_n, \\
z_{n+1} &= z_n + c_1 z_n - c_2 z_n x_n - c_3 z_n y_n.
\end{aligned} \tag{5.20}
$$

Принять в качестве базовых значений $x_0 = 5,\ y_0 = 7,\ z_0 = 7,$

$a_1 = 0.01,\ a_2 = 0.001,\ a_3 = 0.002,\ \ b_1 = 0.002,\ b_2 = 0.002,\ b_3 = 0.003,$
$c_1 = 0.03,\ c_2 = 0.03,\ c_3 = 0.004,$

Модель эпидемии (x – численность здоровых особей, y – численность больных особей)

$$
\begin{aligned}
x_{n+1} &= x_n - b x_n y_n, \\
y_{n+1} &= y_n + b x_n y_n
\end{aligned} \tag{5.21}
$$

Принять в качестве базовых значений начальной численности здоровых и больных особей $x_0 = 10,\ y_0 = 1$ и коэффициент передачи инфекции $\beta = 0.05; 0.03; 0.02; 0.01.$

Модель эпидемии с учетом смертности больных (x – численность здоровых особей, y – численность больных особей)

$$
\begin{aligned}
x_{n+1} &= x_n - b x_n y_n, \\
y_{n+1} &= y_n + b x_n y_n - \gamma y_n
\end{aligned} \tag{5.22}
$$

Принять в качестве базовых значений начальной численности здоровых и больных особей $x_0 = 10,\ y_0 = 1,$ коэффициент передачи инфекции $\beta = 0.01; 0.05; 0.1$ и коэффициент смертности $\gamma = 0.1; 0.2.$

Модель Николсона–Бейли (1935) (y – численность хозяина, x – численность паразита)

$$
\begin{aligned}
x_{n+1} &= y_n (1 - e^{-x_n}), \\
y_{n+1} &= y_n e^{r(1 - y_n / K) - x_n}
\end{aligned} \tag{5.23}
$$

Исследовать систему (5.23) при $r = 1,\ K = 1; 2; 3; 4; 5,$ $r = 1.8,\ K = 5,$
$r = 2.5,\ K = 3; 3.5; 4.03; 4.1; 4.15; 4.5; 5; 5.2; 5.5.$

§6. Модели переноса воздушных загрязнений

6.1. Гауссова модель распространения загрязнений от точечного источника

Для описания процессов распространения загрязняющих веществ в атмосферном воздухе используются различные модели, основанные на решении уравнения конвективно-диффузионного переноса [2,13]. Рассмотрим задачу переноса примеси от стационарного точечного источника. Пусть выброс загрязняющих веществ в атмосферу осуществляется через трубу высоты h. Примем за начало координат основание трубы. Тогда точечный источник загрязнения атмосферы имеет координаты $(0;0;h)$. Если высота трубы, через которую производится выброс в атмосферу, небольшая, при расчете пространственного распределения концентрации примеси необходимо учитывать эффект отражения распространяющихся в атмосфере взвесей от земной поверхности. Эффект отражения можно учесть, применяя метод виртуального источника загрязнения. Согласно данному методу концентрация в некоторой точке определяется суммой двух концентраций, одна из которых получается при рассмотрении реального источника в предположении отсутствия эффекта отражения, а другая – при рассмотрении виртуального источника, расположенного на высоте $(-h)$. В итоге пространственное распределение концентрации загрязняющей примеси $c(x,y,z)$ [мг/м^3] от описанного источника для постоянного ветра, направленного вдоль оси x, в рамках гауссовой модели рассеяния может быть представлено формулой [13]:

$$c(x,y,z)=\frac{q}{2\pi\sigma_y\sigma_z u}\exp\left(\frac{-y^2}{\sigma_y^2}\right)\left[\exp\left(\frac{-(z-h)^2}{\sigma_z^2}\right)+\exp\left(\frac{-(z+h)^2}{\sigma_z^2}\right)\right], \quad (6.1)$$

где q [г/с] – мощность источника, h [м] – высота источника, u [м/с] – сред-

няя скорость ветра, ось *y* – поперечно-горизонтальное направление, ось *z* направлена вертикально вверх. Величины σ_y, σ_z – горизонтальная и вертикальная функции рассеяния от источника, показывающие как меняется ширина гауссовой струи с увеличением расстояния *x* от источника загрязнения.

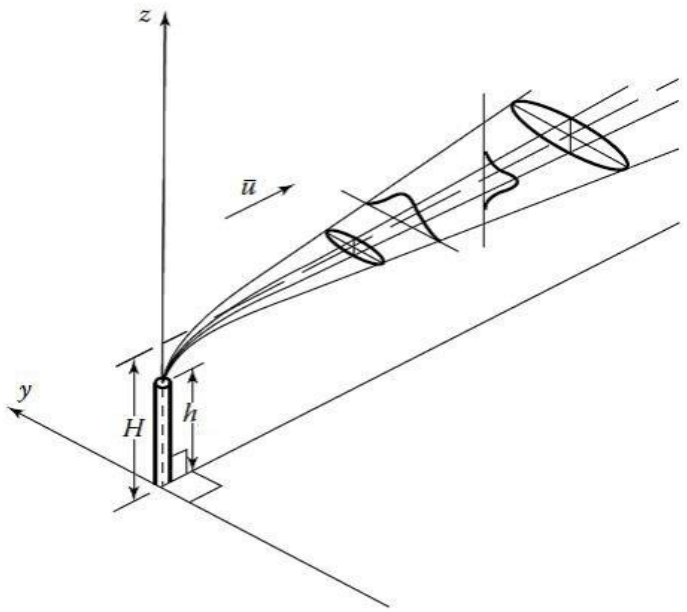


Рис. 6.1. Схема гауссовой струи

Известны гауссовы модели с различными функциями рассеяния σ_y, σ_z: модель Паскуилла-Бриггса, модель Паскуилла-Гиффорда и другие. В модели Паскуилла-Бриггса значения функций рассеяния задаются в зависимости от класса устойчивости атмосферы (табл. 6.1). Каждому классу устойчивости соответствуют определенные значения скорости ветра *u* и степени дневной инсоляции и ночной облачности (табл. 6.2).

Таблица 6.1

Формулы для σ_y, σ_z, рекомендованные Бриггсом для расстояний от 100 до 10000 м в условиях города и открытой местности

Класс устойчивости атмо-	Состояние устойчивости	σ_y (м)	σ_z (м)

сферы Паску-илла			
Открытая местность			
A	Сильно неустойчивое (1)	$0.22x(1+0.0001x)^{-1/2}$	$0.2x$
B	Неустойчивое (2)	$0.16x(1+0.0001x)^{-1/2}$	$0.12x$
C	Слабо неустойчивое (3)	$0.11x(1+0.0001x)^{-1/2}$	$0.08x(1+0.0002x)^{-1/2}$
D	Равновесное (4)	$0.08x(1+0.0001x)^{-1/2}$	$0.06x(1+0.0015x)^{-1/2}$
E	Слабоустойчивое (5)	$0.06x(1+0.0001x)^{-1/2}$	$0.03x(1+0.0003x)^{-1}$
F	Устойчивое (6)	$0.04x(1+0.0001x)^{-1/2}$	$0.016x(1+0.0003x)^{-1}$
Городская местность			
A-B	Неустойчивое (1-2)	$0.32x(1+0.0004x)^{-1/2}$	$0.24x(1+0.001x)$
C	Слабо неустойчивое (3)	$0.22x(1+0.0004x)^{-1/2}$	$0.2x$
D	Равновесное (4)	$0.16x(1+0.0004x)^{-1/2}$	$0.14x(1+0.0003x)^{-1/2}$
E-F	Устойчивое (5-6)	$0.11x(1+0.0004x)^{-1/2}$	$0.08x(1+0.0015x)^{-1/2}$

Таблица 6.2

Классы устойчивости атмосферы Паскуилла

Скорость ветра на высоте 10 м, м/с	Степень инсоляции днём			Облачность ночью, баллы	
	сильная	умеренная	слабая	10 (общая) или >5 (нижняя)	<4 (нижняя)
<2	A	A-B	B	—	—
2-3	A-B	B	C	E	F
3-5	B	B-C	D	D	E
5-6	C	C-D	D	D	D
>6	C	D	D	D	D

Приняв для σ_y, σ_z формулы, соответствующие классу устойчивости D для открытой местности и условий городской застройки (табл. 6.1)

$$\sigma_y = 0.08x(1+0.0001x)^{-1/2},\ \sigma_z = 0.06x(1+0.0015x)^{-1/2} \qquad (6.2)$$

$$\sigma_y = 0.16x(1+0.0004x)^{-0.5},\ \sigma_z = 0.14x(1+0.0003x)^{-0.5} \qquad (6.3)$$

найдем пространственное распределение приземной концентрации загряз-

няющей примеси согласно модели (6.1) для q=80 г/с, u=6 м/с, h=30 м (программа **P6.1**). На рис. 6.2 – 6.4 приведены изолинии приземной концентрации $c(x,y,0)$ и распределение концентрации загрязняющей примеси $c(x,0,0)$ вдоль ветра на оси симметрии. Видно, что различие функций рассеяния для открытой местности и условий городской застройки дает отличные картины распределения концентрации. Во втором случае максимум концентрации достигается ближе к источнику в связи с влиянием значительной шероховатости поверхности земли в условиях городской застройки.

Текст программы P6.1

```
qq:80$
u:3$
h:30$
sy:0.16*x/sqrt(1+0.0004*x)$
sz:0.14*x/sqrt(1+0.0003*x)$
sy:0.08*x/sqrt(1+0.0001*x)$
sz:0.06*x/sqrt(1+0.00015*x)$
c(x,y,z):=qq/(2*%pi*sz*sy*u)*exp(-y^2/sy^2)*(exp(-(z-h)^2/sz^2)+exp(-
(z+h)^2/sz^2))$
wxcontour_plot(c(x,y,0),[x,10,1000],[y,-200,200],[legend,false],
          [gnuplot_preamble, "set cntrparam levels 40"],[grid, 150, 150],
[xlabel, "x,m"],[ylabel, "y,m"]);
wxplot2d(c(x,0,0),[x,1,1000],[xlabel, "x,m"],[ylabel, "c(x,0,0)"]);
```

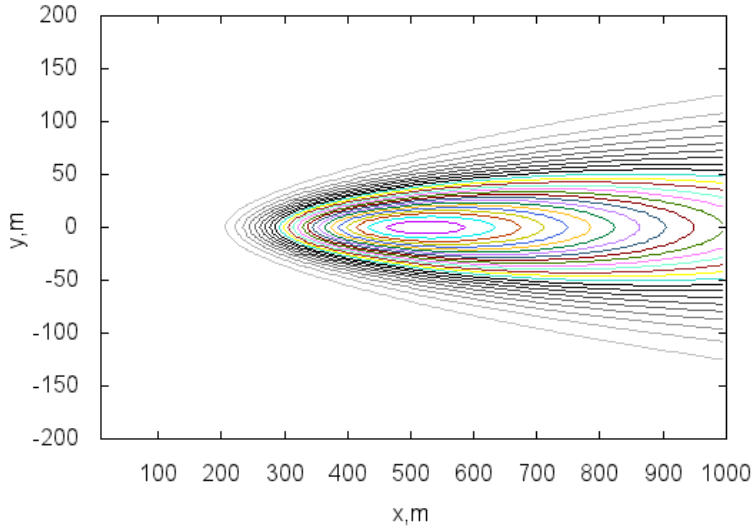


Рис. 6.2. Распределение приземной концентрации загрязняющей примеси для открытой местности

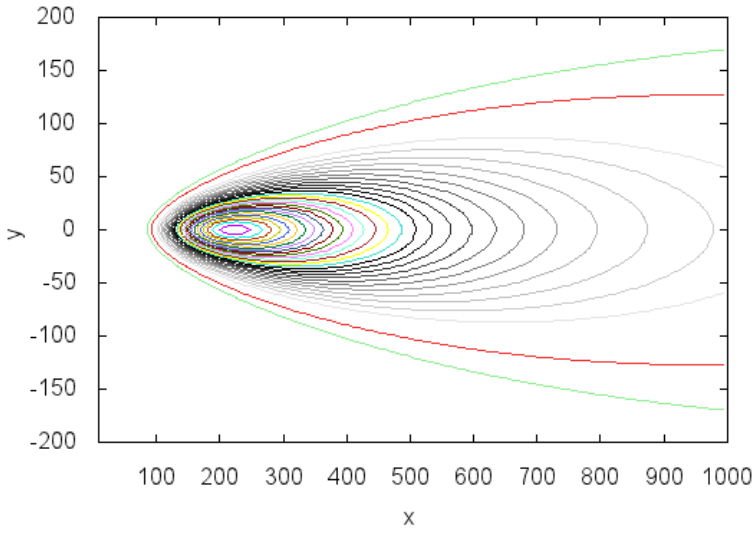
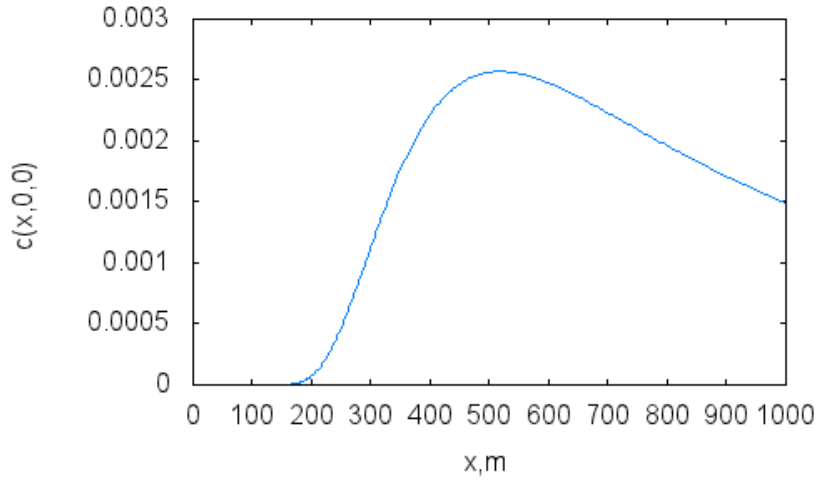
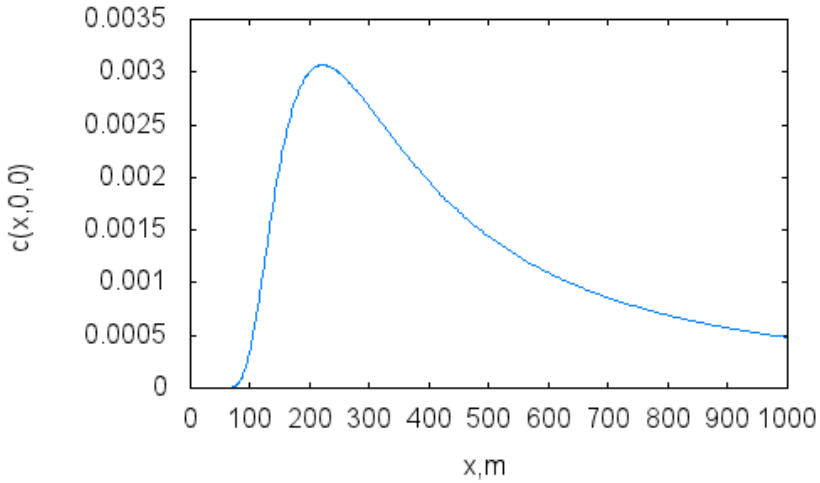


Рис. 6.3. Распределение приземной концентрации загрязняющей примеси для условий городской застройки



а



б

Рис. 6.4. Распределение концентрации загрязняющей примеси c(x,0,0) для открытой местности (а) и условий городской застройки (б)

6.2. Гауссова модель для *n* источников

Предположим, что вместо одного точечного источника имеется n источников выбросов загрязняющих веществ в атмосферу различной высоты h_i, каждый из которых характеризуется своей мощностью выбросов q_i [г/с] и положением в пространстве $\{x_i, y_i\}$, (i=1,2,…, n). Формула (6.1) дает распределение концентрации загрязняющей примеси для источника, расположенного в начале координат. Для того чтобы применить данную формулу к источнику, находящемуся в точке $\{x_i, y_i\}$, осуществим параллельный перенос координат по формулам $x^*=x-x_i$, $y^*=y-y_i$. Тогда суммарная

концентрация загрязнения может быть определена как сумма концентраций веществ от отдельных источников и вычислена по формуле

$$c(x,y,z)=\sum_{i=1}^{n}c(x-x_i,y-y_i,z)= \\ \sum_{i=1}^{n}\frac{q_i}{2\pi u\sigma_y\sigma_z}e^{-y_i^2/2\sigma_{yi}^2}\left(e^{-(z-h_i)^2/2\sigma_z^2}+e^{-(z+h_i)^2/2\sigma_z^2}\right) \tag{6.3}$$

На рис. 6.5 приведено рассчитанное по формуле (6.3) поле концентрации для случая поступления примесей от трех источников с координатами и высотами (в метрах): $x_1=0$ $y_1=0$, $h_1=40$; $x_2=100$ $y_2=100$, $h_2=50$; $x_3=250$, $y_3=-800$, $h_3=30$ (программа **P6.2**). За начало координат принято основание трубы первого источника.

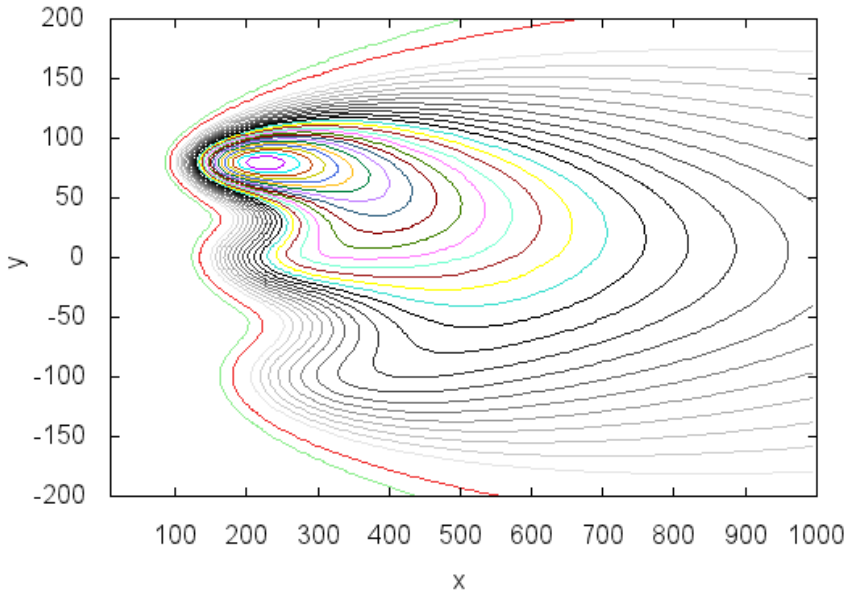


Рис. 6.5. Распределение приземной концентрации загрязняющей примеси от трех точечных источников для условий городской застройки

Текст программы P6.2

```
qq:80$
u:3$
```

```
sy:0.16*x/sqrt(1+0.0004*x)$
sz:0.14*x/sqrt(1+0.0003*x)$
c(x,y,z,h):=qq/(2*%pi*sz*sy*u)*exp(-y^2/sy^2)*(exp(-(z-h)^2/sz^2)+exp(-
(z+h)^2/sz^2));
csum(x,y,z):=c(x,y,z,40)+c(x+100,y+100,z,50)+c(x+250,y-80,z,30);
wxcontour_plot(csum(x,y,0),[x,10,1000],[y,-200,200],[legend,false],
[gnuplot_preamble, "set cntrparam levels 40"], [grid, 150, 150]);
```

Задания к параграфу

Рассчитать по формуле (6.3) распределение приземной концентрации загрязнений от четырех источников с координатами и высотами (в метрах), приведенными в таблице 6.3.

Таблица 6.3

	x_i, *м*	y_i, *м*	h_i, *м*
1	0	20	30
2	100	0	40
3	30	-50	50
4	120	80	30

§7. Модель загрязнения реки

Загрязнение водных систем является одной из важных экологических проблем. Рассмотрим простейшую модель водной системы, включающую в себя растворенные в воде кислород и органические отходы [6,17]. С течением времени отходы разлагаются под воздействием бактерий при потреблении кислорода. Концентрация отходов $c_p(t)$ [мг/л] определяется биохимической потребностью в кислороде, т.е. количеством кислорода на единицу объема воды, необходимым для разложения отходов. Предпола-

гая, что скорость разложения отходов пропорциональна их концентрации при условии, что в воде присутствует достаточно кислорода для поддержания процесса разложения, можем получить уравнение

$$\frac{dc_p}{dt} = -k_1 c_p \tag{7.1}$$

где k_1 [1/день] – коэффициент потребления кислорода. Решение уравнения (7.1) с начальным условием $c_p(0) = c_{p0}$ запишется в виде

$$c_p = c_{p0} e^{-k_1 t} \tag{7.2}$$

Введем величины c_0 и $c(t)$ как равновесную концентрацию кислорода в воде в отсутствие отходов и фактическую концентрацию кислорода с учетом его расходования на разложение загрязнений. Наряду с потреблением кислорода в загрязненной водной системе есть механизм его увеличения за счет поглощения кислорода из атмосферы водной поверхностью, называемый реаэрацией.

Учитывая два описанных процесса, влияющих на количество кислорода, запишем уравнение для разности концентраций (дефицит кислорода) $d_{o_2} = c_0 - c$

$$\frac{dd_{o_2}}{dt} = k_1 c_p - k_2 d_{o_2} \tag{7.3}$$

где k_2 [1/день] – коэффициент реаэрации. В начальный момент времени $d_{o_2}(0) = d_0$. Уравнение (7.3) может быть решено аналитически

$$d_{o_2}(t) = \frac{k_1 c_{p0}}{k_2 - k_1}\left(e^{-k_1 t} - e^{-k_2 t}\right) + d_0 e^{-k_2 t} \tag{7.4}$$

Зависимости величин c_p и d_{o_2} от времени для $k_1 = 0.25$, $k_2 = 0.4$, $c_{p0} = 100$ мг/л $d_0 = 20$ мг/л приведены на рис. 7.1 (программа **P7.1**). По мере разложения отходов их концентрация падает, в то время как дефицит кислорода растет до некоторого значения в связи с его расходованием. После дости-

жения своего максимума величина d_{o_2} падает из-из уменьшения начальной концентрации загрязнения.

Решения (7.2) и (7.4) дают зависимость величин c_p и d от времени. Считая в первом приближении, что скорость реки u постоянна, заменим время в (7.2) и (7.4) отношением расстояния к скорости течения реки по формуле $t=x/u$. Зависимости $c_p(x)$ и $d_{o_2}(x)$, полученные для тех же значений параметров, что и на рис. 7.1, приведены на рис. 7.2 (программа **P7.2**). Видно, что максимум дефицита кислорода достигается на расстоянии около десяти километров от точки выброса.

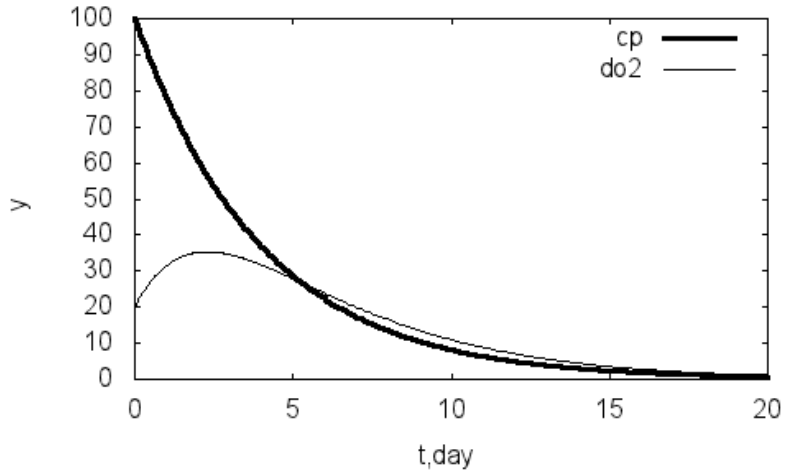


Рис. 7.1. Зависимости величин c_p и d от времени

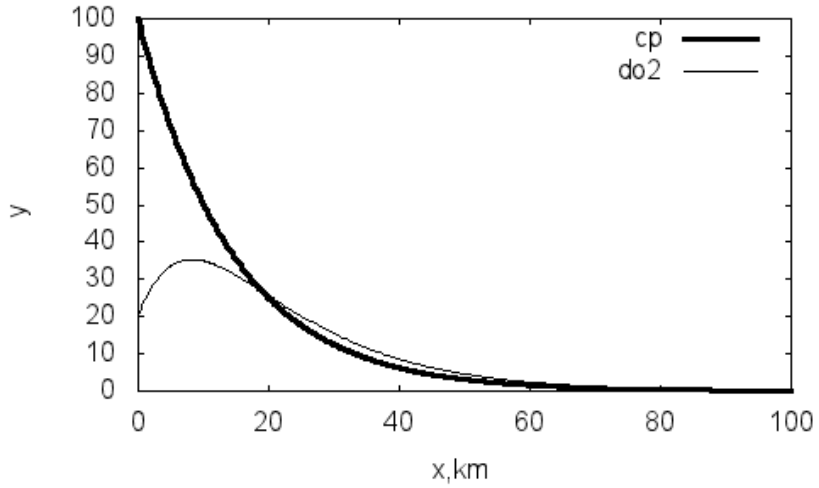


Рис. 7.2. Зависимости величин c_p и d от расстояния до точки выброса

Текст программы P7.1

```
k1:0.25$
k2:0.4$
cp0:100$
d0:20$
cpt:cp0*exp(-k1*t);
dt:k1*cp0/(k2-k1)*(exp(-k1*t)-exp(-k2*t))+d0*exp(-k2*t);
wxplot2d([cpt,dt],[t,0,20],[style, [lines,3,5], [lines,1,5]],
[legend, "cp", "do2"],[xlabel,"t,day"], [ylabel,"y"]);
```

Текст программы P7.2

```
us:0.001$
uday:us*3600$
k1:0.25$
k2:0.4$
cp0:100$
d0:20$
cpt:cp0*exp(-k1*x/uday);
dt:k1*cp0/(k2-k1)*(exp(-k1*x/uday)-exp(-k2*x/uday))+d0*exp(-k2*x/uday);
wxplot2d([cpt,dt],[x,0,100],[legend, "cp", "d"],[xlabel,"x,km"], [ylabel,"y"],
[style, [lines,3,5],[lines,1,5]]);
```

Задания к параграфу

Решить задачу Коши для уравнений (7.1) и (7.3) численно с помощью процедуры rk (метод Рунге-Кутта) (программа **P7.3**). Сравнить кривые $c_p(t)$ и $d_{o_2}(t)$, полученные численно и из формул (7.2) и (7.4).

Текст программы P7.3

```
load("dynamics");
k1:0.25$
k2:0.4$
cp0:100$
d0:20$
sol:rk([-k1*cp,k1*cp-k2*d],[cp,d],[cp0,d0],[t,0,20,0.1])$
len:length(sol)$
tt:makelist(sol[k][1],k,1,len)$
y1:makelist(sol[k][2],k,1,len)$
y2:makelist(sol[k][3],k,1,len)$
wxplot2d([[discrete,tt,y1],[discrete,tt,y2]],[style, [lines,3,5], [lines,1,5]],
[legend, "cp", "d02"],[xlabel,"t"], [ylabel,"cp,d"]);
```

§ 8. Знакомство с программой Maxima

8.1. Простейшие операции. Ввод любой команды в Maxima заканчивается символом “;” или “$”. Первый символ используется, если результат выполнения команды надо вывести на экран, а второй – когда команда выполняется без вывода (также этот символ используется при выводе графиков). Выполнение команды происходит при нажатии комбинации “Shift+Enter” или “Ctrl+Enter”.

Вычислим сумму дробей $\frac{1}{3}+\frac{3}{7}$. Запишем в программе команду

```
--> 1/3+3/7;
```

и нажмем “Shift+Enter”. В результате получим ответ:

(%) $\frac{16}{21}$

Если результат надо получить в десятичной форме, после команды следует дописать “, numer”:

```
--> 1/3+3/7, numer;
```

```
(%) 0.76190476190476
```

Программа выводит 16 знаков числа. Изменить это число (например, когда требуется меньшая точность) можно командой fpprintprec, указав, сколько знаков числа следует выводить:

```
--> fpprintprec:5;
```

Теперь, при выводе числа в десятичной записи Maxima будет выдавать лишь 5 знаков числа:

```
--> 11/3-3/7, numer;
```

```
(%) 3.2381
```

Для ввода чисел, заданных в экспоненциальной форме, используется буква ‘‘e’’. Найдем произведение чисел $3.4 \cdot 10^{-5}$ и $6.7 \cdot 10^{8}$:

```
--> 3.4e-5*6.7e8;
```

```
(%) 22780.0
```

Для четырёх основных математических операций используются символы “+”, “-”, “*”, “/”. Отметим, что если в обычной записи знак умножения иногда опускается, в программе Maxima его следует писать всегда. Для указания приоритета операций используются круглые скобки (символы “(” и “)”). Так, для того, чтобы вычислить, чему равна дробь $\dfrac{6(3+4)}{7-3}$, надо использовать следующую команду

```
--> 6*(3+4)/(7-3);
```

Для возведения в степень используется символ “^”. Для того, чтобы вычислить 2^{10}, 5^{-2}, $\sqrt[3]{27}$ следует писать команды:

```
--> 2^10; 5^(-2); 27^(1/3);
```

```
(%) 1024
(%) 1/25
(%) 3
```

Для квадратного корня можно также использовать функцию `sqrt()`. Найдем $\sqrt{169}$ и $\sqrt{170}$:

```
--> sqrt(169); sqrt(170)
```

```
(%) 13
(%) sqrt(170)
```

Найдем $\sqrt{170}$ в десятичной форме:

```
--> sqrt(170), numer;
```

```
(%) 13.038
```

8.2. Переменные и постоянные. Постоянные в Maxima начинаются с символа "%". Так, числа π, e, i следует писать так: "%pi", "%e", "%i". Найдем численное значение π и возведем i в квадрат:

```
--> %pi, numer;

(%) 3.1416

--> %i^2;

(%) -1
```

Буквы латинского алфавита программа понимает как переменные. Заглавные и строчные буквы считаются различными переменными. Программа понимает и русские буквы, но из-за того, что многие из них имеют одинаковое написание с латинскими, во избежание путаницы, лучше их не использовать. Заметим также, что при записи латинскими буквами названий греческих букв, на выводе программа запишет результат греческими:

```
--> beta-alpha

(%) β-α
```

Для присваивания переменным значений используется символ ":" (хотя в обычной записи для этого используется символ "="). Зададим $a = 5$ и $b = 10$. Присвоим переменной c значение $a + b$ и переменной d значение $c \cdot b$:

```
--> a:5;b:10;

(%) 5

(%) 10

--> c:a+b; d:c*b;

(%) 15
```

(%) 150

8.3. Списки. Для хранения набора из нескольких числовых данных в Maxima используются списки. С их помощью можно сформировать векторы, матрицы и другие подобные объекты. Объекты списка задаются в квадратных скобках через запятую:

```
--> p:[1, 2, 3]; q:[4, 5, 6];
```

Со списками можно производить некоторые операции, как и с обычными числами. Maxima выполняет соответствующее действие с каждым элементом списка. Например:

```
--> p^2+1;
```

(%) $[2, 5, 10]$

```
--> p+q;p*q;
```

(%) $[5, 7, 9]$

(%) $[4, 10, 18]$

Также существует аналог скалярного произведения векторов, выполняемый командой `"."`:

```
--> p.q;
```

(%) 32

Команды `append` и `join` объединяют два списка, команда `reverse` создаёт список с элементами, записанными в обратной последовательности:

```
--> r:append(p,q); s:join(p,q); t:reverse(p);
```

(%) $[1, 2, 3, 4, 5, 6]$

(%) $[1, 4, 2, 5, 3, 6]$

(%) $[3, 2, 1]$

Длину списка определяем с помощью команды `length`:

```
--> length(r);
```

(%) 6

Для получения нужного элемента списка указываем его номер в квадратных скобках после имени списка:

```
--> s1:s[5];
```

(%) 3

Для создания списков используется команда `makelist`. Работу этой команды проиллюстрируем на нескольких примерах:

```
--> makelist(i^2, i, 5);
```

(%) $[1, 4, 9, 16, 25]$

```
--> makelist(-i, i, 3, 7);
```

(%) $[-3, -4, -5, -6, -7]$

```
--> makelist(10*i, i, -5, 5, 2);
```

(%) $[-50, -30, -10, 10, 30, 50]$

8.4. Основные математические функции. В таблице приведен список основных математических функций.

Отметим, что при записи функции в программе Maxima аргумент следует брать в круглые скобки.

Найдем $|\mathrm{arctg}(\ln e)| + \sqrt{e^{\sin\frac{\pi}{3}}}$:

```
--> abs(atan(log(%e)))+sqrt(exp(sin(%pi/3)));
```

(%) $\dfrac{\pi}{4} + e^{\frac{\sqrt{3}}{4}}$

Запись в Maxima	Функция	Описание
`abs(x)`	$\|x\|$	модуль числа
`sqrt(x)`	$\sqrt{x}$	квадратный корень
`exp(x)`	e^x	экспонента
`log(x)`	$\ln x$	натуральный логарифм
`sin(x)`	$\sin x$	тригонометрические функции
`cos(x)`	$\cos x$	
`tan(x)`	$\operatorname{tg} x$	
`cot(x)`	$\operatorname{ctg} x$	
`asin(x)`	$\arcsin x$	обратные тригонометрические функции
`acos(x)`	$\arccos x$	
`atan(x)`	$\operatorname{arctg} x$	
`acot(x)`	$\operatorname{arcctg} x$	

Функциям можно присваивать имена (командой присваивания ":") и находить их числовые значения при заданном аргументе. Например, функции $\ln 3x + e^{\sqrt{x}}$ присвоим имя `func` и найдем её точное и приближённое (в десятичной записи) значение при $x = 5$:

```
--> func: log(3*x)+exp(sqrt(x));
```

(%) $\log(3\,x) + e^{\sqrt{x}}$

```
--> func, x=5;
```

(%) $\log(15) + e^{\sqrt{5}}$

```
--> func, x=5, numer;
```

(%) 12.065

Заметим, что в приведённых выше примерах `func` является не функцией, а символьным выражением. Для создания собственной функции в Maxima

используется команда :=.

```
--> fun(x,y):=x^3+y^2;

--> fun(u,v); fun(2,3);
```

(%) u^3+v^2

(%) 17

При создании функции Maxima не вычисляет и не преобразует выражение, стоящее после знака :=, а берет его в неизменном символьном виде. В некоторых случаях (например, при определении производной функции) требуется вычислить данное выражение. Для определения функции в этом случае используется команда `define`. Задание той же самой функции с помощью этой команды имеет вид:

```
--> define(g(x,y), x^3+y^2);
```

8.5. Задания к теме.

1. Вычислить $\dfrac{\sqrt{25}+1}{8^{2/3}-1}$.

2. Найти значение выражения $\dfrac{\pi^2}{1+\sqrt{e-1}}$ в десятичной записи.

3. Задать $a=2$, $b=a+\dfrac{1}{a}$, $c=b^a$. Найти сумму $a+b+c$.

4. Присвоить функции $\dfrac{e^x-e^{-x}}{e^x+e^{-x}}$ имя `th` и вычислить значения этой функции при *а*) $x=1$, *б*) $x=\ln(2)$, *в*) $x=-4$.

Ответы: **1.** 2; **2.** 4.2710; **3.** 43/4 **4.** *а*) 0.7616, *б*) 0.6, *в*) -0.9993.

§ 9. Преобразование арифметических выражений

Познакомимся с основными командами, служащими для обработки математических выражений, т.е. для представления результата в нужном для пользователя виде.

9.1. Раскрытие скобок и разложение на множители. Для раскрытия скобок в выражении используется команда `expand()`. Раскроем скобки в выражении $(x+y)^5$

```
--> expand((x+y)^5);
```

(%) $y^5 + 5\,x\,y^4 + 10\,x^2\,y^3 + 10\,x^3\,y^2 + 5\,x^4\,y + x^5$

Для разложения на множители в программе Maxima используется команда `factor()`. Разложим на множители $x^6 - 1$:

```
--> factor(x^6-1);
```

(%) $(x-1)\,(x+1)\,\left(x^2-x+1\right)\,\left(x^2+x+1\right)$

9.2. Упрощение арифметических выражений. Для приведения выражений к простому виду существуют команды `ratsimp()` и `radcan()`. Первая команда работает с арифметическими выражениями, а вторая упрощает выражения с дробными степенями, логарифмами и экспонентами.

Упростим дробь $\dfrac{x+t}{x^2-t^2}$:

```
--> ratsimp((x+t)/(x^2-t^2));
```

(%) $\dfrac{1}{x-t}$

Упростим выражение $f = \ln \dfrac{e^{4w}}{z^6}$. Запишем вначале его под именем `f`:

```
--> f:log(exp(4*w)/z^6);
```

Попробуем преобразовать командой `ratsimp()`:

```
--> ratsimp(f);
```

(%) $\log\left(\dfrac{e^{4\,w}}{z^6}\right)$

Как мы видим, команда `ratsimp()` упростить это выражение не смогла. Выполним упрощение командой `radcan()`:

```
--> radcan(f);
```

(%) $4\,w - 6\log(z)$

Для разложения дроби на сумму простых дробей используется команда `partfrac`. Разложим дробь $\dfrac{ax^2+b}{x^3+x^4}$:

```
--> partfrac((a*x^2+b)/(x^3+x^4), x);
```

(%) $\dfrac{-b-a}{x+1} + \dfrac{b+a}{x} - \dfrac{b}{x^2} + \dfrac{b}{x^3}$

9.3. Упрощение тригонометрических выражений. Для преобразований тригонометрических выражений существуют команды `trigexpand()`, `trigreduce()`, `trigsimp()`. Первая команда раскладывает все тригонометрические функции от сумм и кратных углов через функции одинарного угла.

Запишем $\sin 4x$ через функции аргумента x:

```
--> trigexpand(sin(4*x));
```

(%) $4\cos(x)^3\sin(x) - 4\cos(x)\sin(x)^3$

Запишем $\mathrm{tg}(a+b-c)$ через функции от аргументов a, b, c:

```
--> trigexpand(tan(a+b-c));
```

(%) $-\dfrac{-\tan(a)\tan(b)\tan(c) + \tan(c) - \tan(b) - \tan(a)}{\tan(b)\tan(c) + \tan(a)\tan(c) - \tan(a)\tan(b) + 1}$

Команда `trigreduce()` выполняет свертывание всех произведений тригонометрических функций в тригонометрические функции от сумм. Запишем $\sin(a+b)\sin(a)\sin(a-b)$ в виде суммы:

```
--> trigreduce(sin(a-b)*sin(a)*sin(a+b));
```

(%) $\dfrac{\sin(2\,b+a)}{4} - \dfrac{\sin(2\,b-a)}{4} - \dfrac{\sin(3\,a)}{4} + \dfrac{\sin(a)}{4}$

Команда `trigsimp()` пытается упросить выражение, применяя к нему простейшие тригонометрические тождества типа $\sin^2 x + \cos^2 x = 1$. Упростим выражение

```
--> trigsimp(1-cos(x)^2);
```

(%) $\sin(x)^2$

Наилучшего результата в преобразовании тригонометрических выражений можно добиться, комбинируя `trigsimp()`, `trigreduce()` и `ratsimp()`/`radcan()`.

9.4. Задания к теме.

1. Задать функцию $f(t) = \dfrac{e^{\sin 2t + \cos t} - 1}{1 + \ln^2 t}$ и найти ее значение при $t = \pi/3$.
2. Разложить на множители полином $x^5 - x^4 + 2x^3 - 2x^2 + x - 1$.
3. Упростить $\dfrac{16r(x+r)^3 - (x-r)^4 + x^4}{(x+r)^2 - (x-r)^2}$.
4. Упростить $\dfrac{\sqrt{x-a}(x+a) - (x-a)^{\frac{3}{2}}}{\sqrt{x^2-a^2}}$.
5. Упростить $\cos^3\alpha \sin 3\alpha + \sin^3\alpha \cos 3\alpha$.

Ответы: **1.** 2.9135; **2.** $(x-1)(x^2+1)^2$; **3.** $6x^2 + 6rx + 20r^2$ **4.** $\frac{2a}{\sqrt{x+a}}$; **5.** $\frac{3}{4}\sin 4\alpha$.

§ 10. Операции с матрицами

10.1. Задание матриц. Матрицы вводятся с помощью команды `matrix`, каждая строка пишется в квадратных скобках. Например, зададим матрицу A:

```
--> A: matrix([1,2,3], [4,5,6], [7,8,9]);
```

(%) $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

Можно получить доступ к любому элементу матрицы, записав его индек-

сы в квадратных скобках. Если написать лишь один индекс, Maxima выведет заданную строку.

```
--> A[2,3];A[2];
```

(%) 6

(%) $[4, 5, 6]$

Команда `transpose()` транспонирует матрицу.

```
--> transpose(A);
```

(%)
$$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

С помощью этой команды можно вывести заданный столбец:

```
--> transpose(A)[3];
```

(%) $[3, 6, 9]$

Команды `addrow`/`addcol` добавляют к матрице дополнительную строку/ряд. Заметим, что эти команды не изменяют исходную матрицу (то есть после выполнения этих команд матрица A так и останется исходной матрицей 3×3), а создают новую матрицу. Чтобы использовать полученную матрицу в последующих расчетах, ей необходимо дать имя. Так, матрицу $A1$, равную матрице A с добавленной четвертой нулевой строкой, можно задать командой

```
--> A1: addrow(A, [0,0,0]);
```

(%)
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 0 & 0 & 0 \end{pmatrix}$$

Создадим матрицу $A2$, добавим к A новый столбец:

```
--> A2: addcol(A, [9,9,9]);
```

$$(\%) \quad \begin{pmatrix} 1 & 2 & 3 & 9 \\ 4 & 5 & 6 & 9 \\ 7 & 8 & 9 & 9 \end{pmatrix}$$

Убрать ненужные строки или столбцы из матрицы можно командой `submatrix`. Убираемые номера строк надо писать через запятую до имени исходной матрицы, а номера столбцов – после. Например, удалим из матрицы A первую строку и третий столбец. Полученной матрице дадим имя $A3$

```
--> A3: submatrix(1, A, 3);
```

$$(\%) \quad \begin{pmatrix} 4 & 5 \\ 7 & 8 \end{pmatrix}$$

Для создания нулевой матрицы существует команда `zeromatrix(n,m)`. Создадим нулевую матрицу R, состоящую из 3 строк и 5 столбцов:

```
--> R: zeromatrix(3, 5);
```

10.2. Простейшие операции с матрицами. Матрица A у нас уже введена, зададим еще одну матрицу B:

```
--> B: matrix([1,1,1], [0,1,2], [1,0,0]);
```

$$(\%) \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 0 \end{pmatrix}$$

Команды “+” и “-” выполняют сложение и вычитание матриц

```
--> A+B; A-B;
```

(%) $\begin{pmatrix} 2 & 3 & 4 \\ 4 & 6 & 8 \\ 8 & 8 & 9 \end{pmatrix}$ $\begin{pmatrix} 0 & 1 & 2 \\ 4 & 4 & 4 \\ 6 & 8 & 9 \end{pmatrix}$

Команда “*” выполняет поэлементное умножение. Для матричного умножения есть команда “.”:

```
--> A*B; A.B;
```

(%) $\begin{pmatrix} 1 & 2 & 3 \\ 0 & 5 & 12 \\ 7 & 0 & 0 \end{pmatrix}$ $\begin{pmatrix} 4 & 3 & 5 \\ 10 & 9 & 14 \\ 16 & 15 & 23 \end{pmatrix}$

Есть также команды поэлементного “^” и матричного “^^” возведения в целую степень:

```
--> A^2; A^^2;
```

(%) $\begin{pmatrix} 1 & 4 & 9 \\ 16 & 25 & 36 \\ 49 & 64 & 81 \end{pmatrix}$ $\begin{pmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{pmatrix}$

Определитель находится командой `determinant()`:

```
--> determinant(A);
```

```
(%) 0
```

```
--> determinant(B);
```

```
(%) 1
```

Обратная матрица находится возведением в степень -1:

```
--> C: B^^(-1);
```

(%)
$$\begin{pmatrix} 0 & 0 & 1 \\ 2 & -1 & -2 \\ -1 & 1 & 1 \end{pmatrix}$$

Проверяем

```
--> B.C; C.B;
```

(%)
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

10.3. Нахождение собственных значений и векторов матриц. Для аналитического нахождения собственных значений матрицы существуют команда `eigenvalues`:

```
--> eigenvalues(A);
```

(%) $[[-\frac{3\cdot\sqrt{33}-15}{2}, \frac{3\cdot\sqrt{33}+15}{2}, 0], [1, 1, 1]]$

Результат является списком, состоящим из двух списков. В первом списке содержатся сами собственные значения, а во втором – кратность соответствующего собственного значения. Для нахождения собственных векторов используется команда `eigenvectors`.

```
--> eigenvectors(A);
```

Заметим, что эти команды находят собственные числа и векторы в аналитическом виде. Если матрицы имеют большие размерности, они могут не дать результат. В этом случае возможно нахождение собственных чисел и векторов численно, с помощью пакета `lapack`. Вначале следует загрузить этот пакет:

```
--> load(lapack)$
```

Зададим требуемую точность:

```
--> fpprintprec: 4;
```

Теперь найдем собственные значения `L`, правые `U` и левые `V` матрицы собственных векторов командой

```
--> [L, V, U]: dgeev(A, true, true);
```

(%) $[16.12, -1.117, -1.3037 \cdot 10^{-15}]$

(%)
$$\begin{pmatrix} -0.232 & -0.7858 & 0.4082 \\ -0.5253 & -0.08675 & -0.8165 \\ -0.8187 & 0.6123 & 0.4082 \end{pmatrix}$$

(%)
$$\begin{pmatrix} -0.4645 & -0.8829 & 0.4082 \\ -0.5708 & -0.2395 & -0.8165 \\ -0.677 & 0.4039 & 0.4082 \end{pmatrix}$$

Если матрицы собственных векторов не нужны, можно в аргументах команды написать `false` или вовсе их опустить.

10.4. Задания к теме.

1. Задать матрицы

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 2 & 2 \\ -1 & 1 & 1 \end{pmatrix} \quad \text{и} \quad B = \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix}$$

Найти $\det(A)$ и матрицу $X = A^{-1} \cdot B$.

2. Получить матрицы $A1$ и $A2$ добавлением к матрице A строчки/столбца элементов матрицы B.

3. Из матрицы A получить матрицу $A3$, заменив ее второй столбец элементами матрицы B.

Ответ: **1.** -8, $[2, 3, 0]$.

§ 11. Решение уравнений и систем уравнений

11.1. Аналитическое нахождение корней уравнений. Для решения уравнения используется команда `solve()`.

Решим квадратное уравнение $x^2 - ax + 5 = 0$, записав его вначале под именем `eq`:

```
--> eq:x^2-a*x+5=0; solve(eq, x);
```

(%) $[x = a - \sqrt{a^2 - 5}, x = \sqrt{a^2 - 5} + a]$

Второй аргумент в команде `solve` указывает, что надо найти. Например, найдем из того же уравнения переменную a:

```
--> solve(eq, a);
```

(%) $[a = \frac{x^2 + 5}{2x}]$

Программа находит также комплексные корни. Найдем все три корня уравнения $x^3 = 1$. Найденные корни запишем под именем `roots`:

```
--> roots: solve(x^3=1, x);
```

(%) $[x = \frac{\sqrt{3}\,i - 1}{2}, x = -\frac{\sqrt{3}\,i + 1}{2}, x = 1]$

Команда `solve` результат выдает в виде списка (матрицы с одной строкой). Если нам нужен лишь второй корень, то его можно получить командой

```
--> roots[2];
```

(%) $x = -\frac{\sqrt{3}\,i + 1}{2}$

Этот корень записан в виде выражения. Если для дальнейших расчетов нам требуется лишь его числовое значение (то есть лишь правая часть выражения, после знака `=`), то для этого используется команда `rhs()`:

```
--> rhs(roots[2]);
```

$$(\%) \quad -\frac{\sqrt{3}\,i+1}{2}$$

11.2. Аналитическое решение систем уравнений. Команда `solve` может решать и системы уравнений. Уравнения и переменные пишутся в квадратных скобках через запятую. Решим систему

$$\begin{cases} 2x+5y=9, \\ x^2+y^2=5. \end{cases}$$

Для этого вначале запишем исходные уравнения под именами `eq1` и `eq2`:

```
--> eq1:2*x+5*y=9; eq2:x^2+y^2=5;
```

Далее, для решения системы используем команду `solve()`:

```
--> solve([eq1, eq2], [x, y]);
```

$$(\%) \quad [[x=2, y=1], [x=-\frac{22}{29}, y=\frac{61}{29}]]$$

Если система уравнений линейна, можно решать и недоопределённые системы. Решим систему

$$\begin{cases} x+y+z=3, \\ x+2y+3z=6. \end{cases}$$

```
--> eq1:x+y+z=3; eq2:x+2*y+3*z=6;

--> solve([eq1, eq2], [x, y, z]);
```

$$(\%) \quad [x=\%r1, y=3-2\%r1, z=\%r1]$$

Мы видим, решение нашлось с точностью до постоянной `%r1`.

11.3. Численное нахождение корней уравнений. Точное решение удается найти не всегда. Попробуем найти корни уравнения $x^5-6x+2=0$:

```
--> eq:x^5-6*x+2=0; solve(eq, x);
```

$$(\%) \quad [0=x^5-6\,x+2]$$

В этом случае Maxima решить уравнение не смогла. Корни этого уравнения можно найти численно. Если требуется найти корни полинома (как в нашем случае) можно использовать команду `allroots()`. Найдем все корни уравнения `eq`:

```
--> allroots(eq);
```

(%) $[x = 0.33402, x = -1.63921, x = 1.57561\,i - 0.08112, x = -1.57561\,i - - 0.08112, x = 1.46744]$

Так как наше уравнение было пятой степени, программа нашла все пять корней, три из них – вещественные, а два – комплексные.

Для поиска корней произвольной функции используется команда `find_root()`. Этой команде надо указать отрезок (то есть наименьшее и наибольшее значение x), на котором расположен корень уравнения. Если на этом отрезке корней нет, Maxima выдаст ошибку. Если на отрезке несколько корней, то Maxima найдет лишь один из корней или выдаст ошибку. Поэтому перед использованием команды `find_root()` необходимо провести дополнительное исследование, например, построить график функции и убедиться, что на задаваемом отрезке расположен лишь один корень.

Найдем корень уравнения $\cos(x) = x^2 + x$ на отрезке $x \in [0, 3]$:

```
--> find_root(cos(x)=x^2+x, x, 0, 3);
```

(%) 0.55

На отрезке $[-5, 0]$ найдем ещё один корень:

```
--> find_root(cos(x)=x^2+x, x, -5, 0);
```

(%) -1.2512

11.4. Численное решение систем уравнений. Для численного решения систем уравнений в программе Maxima используется метод Ньютона. Для этого необходимо вначале загрузить пакет `mnewton` командой:

```
--> load(mnewton)$
```

Решим систему

$$\begin{cases} x + 3\ln x - y^2 & = 0, \\ 2x^2 - xy - 5x + 1 & = 0. \end{cases}$$

Запишем исходные уравнения под именами `eq1` и `eq2`:

```
--> eq1:x+3*log(x)-y^2; eq2:2*x^2-x*y-5*x+1;
```

Команда `solve` данную систему решить не может и поэтому найдем решение численно. Для этого используется команда `mnewton`. Этой команде необходима начальная точка. Если корней у системы несколько, численно найдется лишь один корень, обычно ближайший к начальной точке. Если начальная точка расположена далеко от корней, решение может и не найтись. Для решения нашей системы в качестве начальной точки возьмем $x = 5$ и $y = 5$:

```
--> mnewton( [eq1,eq2], [x,y], [5,5]);
```

(%) $[[x = 3.7568, y = 2.7798]]$

Сменив начальную точку на $x = 1$ и $y = -1$, найдем другое решение системы:

```
--> mnewton( [eq1,eq2], [x,y], [1,-1]);
```

(%) $[[x = 1.3735, y = -1.525]]$

11.5. Задания к теме.

1. Решить уравнение $x^3 - 2a^2x + a^3 = 0$.

2. Численно найти оба корня уравнения $e^x = x + 3$.

3. Найти решение систем уравнений:

а) $\begin{cases} y & = x^2 - 1, \\ x & = y^2 - 1. \end{cases}$ б) $\begin{cases} 2x - 4y + 3z & = R, \\ x - 2y + 4z & = 3, \\ 3x - y + 5z & = 2. \end{cases}$

4. Найти численное решение системы: $\begin{cases} 3^x - \dfrac{y}{x} = 5, \\ 2^y + x = 4. \end{cases}$

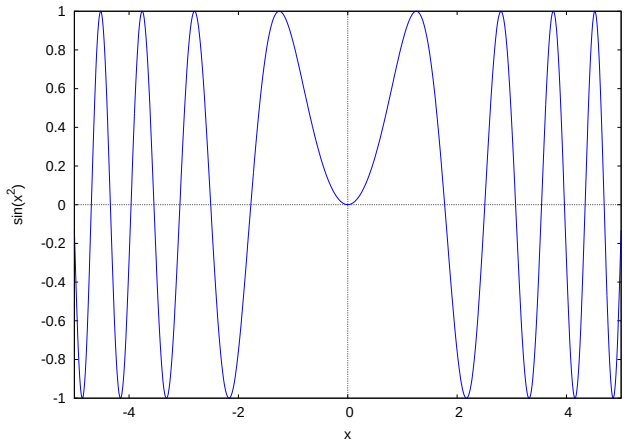
Ответы: **1.** Уравнение имеет три корня: $x_1 = -\frac{(\sqrt{5}+1)\,a}{2}$, $x_2 = \frac{(\sqrt{5}-1)\,a}{2}$, $x_3 = a$; **2.** $x_1 = -2.9475$, $x_2 = 1.5052$; **3.** а) система имеет 4 решения: $x_1 = -1$, $y_1 = 0$; $x_2 = 0$, $y_2 = -1$; $x_3 = y_3 = -\frac{\sqrt{5}-1}{2}$; $x_4 = y_4 = \frac{\sqrt{5}+1}{2}$; б) $x = \frac{6\,R-31}{25}$, $y = -\frac{7\,R-7}{25}$, $z = -\frac{R-6}{5}$; **4.** $x = 1.5986$, $y = 1.2639$.

§ 12. Построение графиков (plot2d)

12.1. Построение графиков явно заданных функций. Для построения графиков есть команды `plot2d()` и `wxplot2d()`. Первая строит график в отдельном окне, вторая – встраивает в лист вычислений. Заметим, при открытом командой `plot2d` окне с графиком дальнейшие вычисления в программе Maxima невозможны, поэтому это окно после просмотра графика необходимо закрыть.

Построим график функции $y = \sin(x^2)$ на отрезке $x \in [-5, 5]$:

```
--> plot2d(sin(x^2), [x, -5, 5])$

--> wxplot2d(sin(x^2), [x, -5, 5])$
```

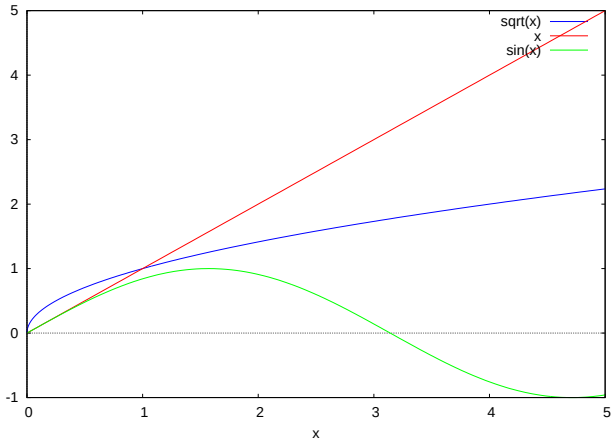


Интервал изменения ординаты программа выбирает сама, исходя из минимальных и максимальных значений функции. Этот интервал можно задать и самому. Построим график $y = \frac{1}{x^2}$ на отрезке $x \in [-5, 5]$ для интервала изменения $y \in [0, 5]$:

```
--> wxplot2d(1/(x^2), [x, -5, 5], [y, 0, 5])$
```

Для построения на одном чертеже нескольких графиков исходные функции записывают в виде списка через запятую в квадратных скобках:

```
--> wxplot2d([sqrt(x), x, sin(x)], [x, 0, 5])$
```



12.2. Построение графиков параметрически заданных функций. Если функция задана в параметрическом виде, используется опция `parametric`. Построим график функции $x(t) = \cos 3t$, $y(t) = \sin 4t$ в интервале изменения параметра $t \in [-\pi, \pi]$:

```
--> wxplot2d([parametric, cos(3*t), sin(4*t),
    [t,-%pi,%pi], [nticks,100]])$
```

Параметр `nticks` задает количество точек, по которым строится график. Чем больше это значение, тем более гладким будет построенная кривая, но при этом увеличивается время, необходимое для ее построения.

Частным случаем параметрически задания функции является задание в полярной системе координат. Построим график кардиоиды $r(\varphi) = 1 - \sin\varphi$:

```
--> r:1-sin(t);

--> wxplot2d([parametric, r*cos(t), r*sin(t),
    [t,-%pi,%pi], [nticks,100]])$
```

12.3. Построение графиков дискретных множеств. Еще одной опцией команды `plot2d` является `discrete`. Она строит график по заданному набору точек. Зададим координаты шести точек в виде списка под именем pts и построим график соединяющей их линии:

```
--> pts:[[0,0],[1,6],[2,9],[3,11],[4,13],[5,14]];

--> wxplot2d([discrete, pts]);
```

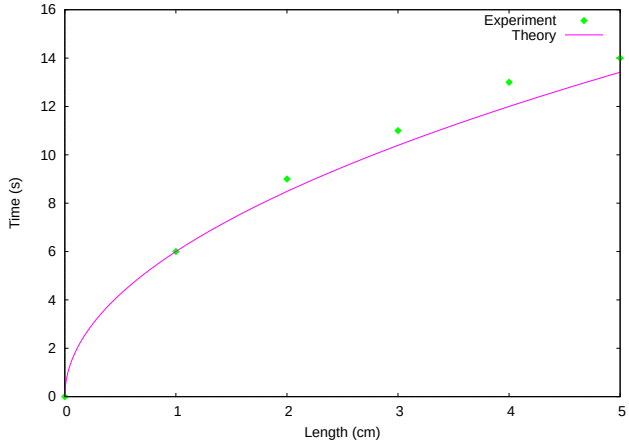
Координаты точек могут быть заданы и в другом виде, в виде двух списков, содержащих координаты x и y соответственно. Так, тот же самый график можно построить командой:

```
--> xpts:[0,1,2,3,4,5]; ypt:[0,6,9,11,13,14];

--> wxplot2d([discrete, xpt, ypt]);
```

12.4. Опции команды plot2d. Команда `plot2d` имеет множество опций, позволяющих настроить внешний вид чертежа. Для знакомства с некоторыми из них наберем команду:

```
--> plot2d([[discrete,pts], 6*sqrt(x)], [x,0,5],
    [y,0,16], [style, [points,4,9,12], [lines,3,4]],
    [legend, "Experiment", "Theory"],
    [xlabel, "Length (cm)"], [ylabel, "Time (s)"])$
```

Эта команда строит график двух функций, первая задана дискретным набором точек `pts`, вторая функцией $6\sqrt{x}$.



Опции, которые были использованы при построении:

`style` – задает стиль линии. Возможные значения `lines`, `points`, `linespoints`. Команда `lines` имеет две дополнительные числовые опции, задающие толщину линии и ее цвет. У команды `points` три опции, задающие размер символа, его цвет и его форму. Команда `linespoints` имеет 4 опции: толщина линии, размер символа, цвет, форма.

`legend` – задает подписи к линиям графиков. Команда `[legend, false]` убирает окно с подписями линий графиков.

`xlabel` – задает подпись к оси абсцисс.

`ylabel` – задает подпись к оси ординат.

Другие возможные опции:

`[box, false]` – отменяет построение рамки вокруг рисунка с графиками.

`[axes, false]` – отменяет построение осей координат.

`[logx]` – ось абсцисс будет логарифмической.

`[logy]` – ось ординат будет логарифмической.

Для сохранения графика в виде EPS файла используются следующие опции: `[gnuplot_term,ps]`, `[gnuplot_out_file,"name.eps"]`

12.5. Задания к теме.

1. На одном чертеже постройте графики функций $y = \operatorname{arctg} x$ и $y = e^{-x^2}$ ($x \in [-4, 4]$).

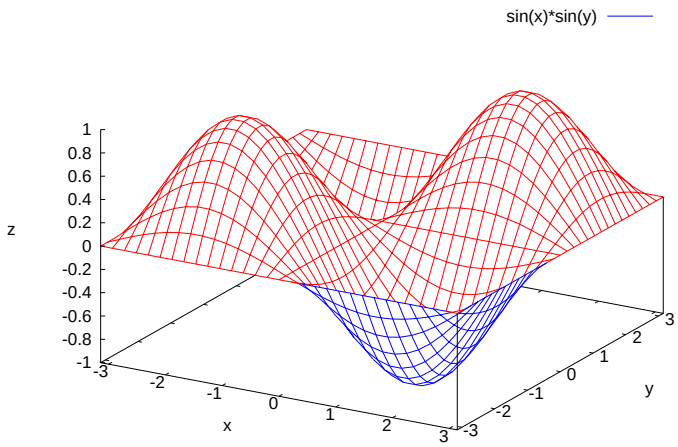
2. Постройте график функции: $\begin{cases} x(t) = \cos t + \cos 5t, \\ y(t) = \sin t - \sin 5t, \end{cases} \quad t \in [0, 2\pi].$

3. Постройте пятиконечную звезду, задав координаты ее вершин в виде набора точек.

§ 13. Построение поверхностей (plot3d)

13.1. Построение явно заданных поверхностей. Для построения трехмерной поверхности функции двух аргументов есть команды `plot3d()` и `wxplot3d()`. Если для построения использовалась команда `plot3d()`, то нарисованную поверхность можно изучить с разных сторон, вращая её с помощью мышки. Построим с помощью этой команды график функции $z = \sin x \sin y$ на прямоугольнике $x \in [-\pi, \pi]$, $y \in [-\pi, \pi]$:

```
--> plot3d(sin(x)*sin(y),[x,-%pi,%pi],[y,-%pi,%pi])$
```

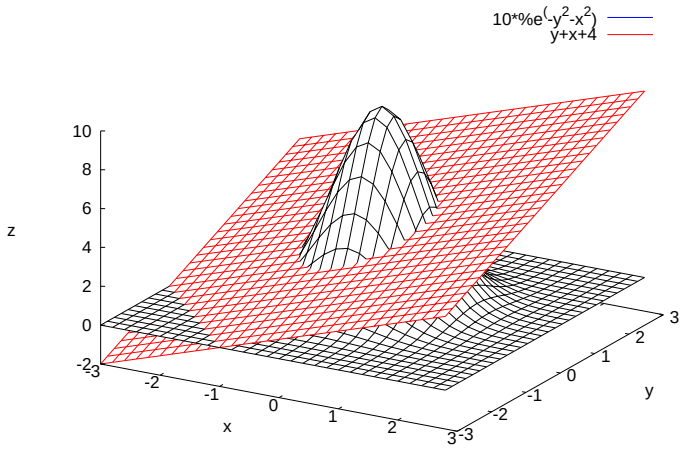


Диапазон изменения значений функции $z \in [z_0, z_1]$ можно задавать, для этого следует к аргументам команды дописать опцию "`[z, z0, z1]`".

На одном чертеже можно разместить графики двух функций, если задать их в виде списка. Построим две поверхности, заданные функциями $f_1(x, y) = x + y + 4$ и $f_2(x, y) = 10e^{-(x^2+y^2)}$:

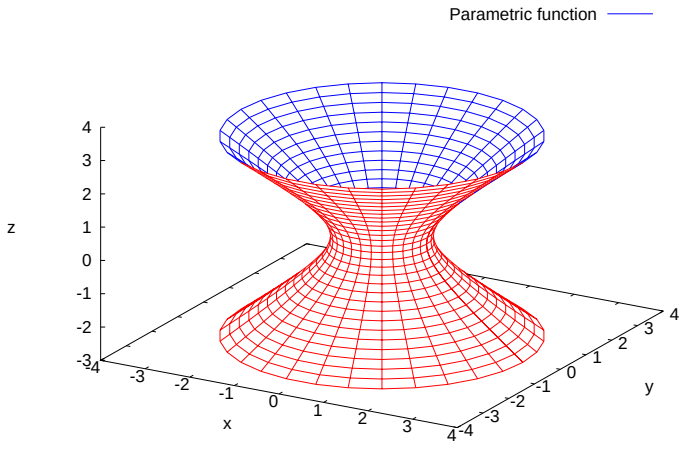
```
--> plot3d([x+y+4, 10*exp(-(x^2+y^2)), [x,-3,3],
    [y,-3,3]], [palette,false])$
```

Использованная здесь опция `[palette, false]` отключает закраску поверхности.



13.2. Построение параметрически заданной поверхности. Если поверхность задана параметрически (от двух параметров u и v), зависимости $x(u,v)$, $y(u,v)$ и $z(u,v)$ следует писать в квадратных скобках через запятую. В качестве примера построения параметрически заданной поверхности рассмотрим пример построения однополостного гиперболоида:

```
--> plot3d([sqrt(1+v^2)*cos(u),sqrt(1+v^2)*sin(u),v],
    [u,0,2*%pi], [v,-3,3])$
```



13.3. Опции команды plot3d. Опция `grid` задает число точек разбиения по каждой переменной. Чем больше задаваемое число, тем более гладкой

будет построенная поверхность, но увеличивается время ее построения. Пример построения первой поверхности с разбиением 100×100:

```
--> plot3d(sin(x)*sin(y), [x,-%pi,%pi], [y,-%pi,%pi],
    [grid,100,100])$
```

Опция `color` задает два цвета, в которую следует окрасить нижнюю и верхнюю сетку поверхности. Для того, чтобы она сработала, необходима отключить закраску опцией `[palette,false]`:

```
--> plot3d(sin(x)*sin(y), [x,-%pi,%pi], [y,-%pi,%pi],
    [palette,false],[color,red,green])$
```

Опция `[mesh_lines_color,false]` отключает прорисовку сетки:

```
--> plot3d(sin(x)*sin(y), [x,-%pi,%pi], [y,-%pi,%pi],
    [mesh_lines_color,false])$
```

13.4. Построение линий уровня. Вместо трехмерной поверхности функцию двух переменных можно изобразить на плоскости с помощью линий уровня. Для этого используется команда

`contour_plot(`функция, границы_x, границы_y, опции`)$`

Опции такие же, как и для команды `plot2d`, например, опция `[legend, false]` отменяет прорисовку легенды. Построим линии уровня функции $z = x^3 + y^2$:

```
--> contour_plot(x^3+y^2, [x,-5,5], [y,-5,5],
    [grid,100,100], [gnuplot_preamble,"set cntrparam levels
    increment -10, 2, 10"])$
```

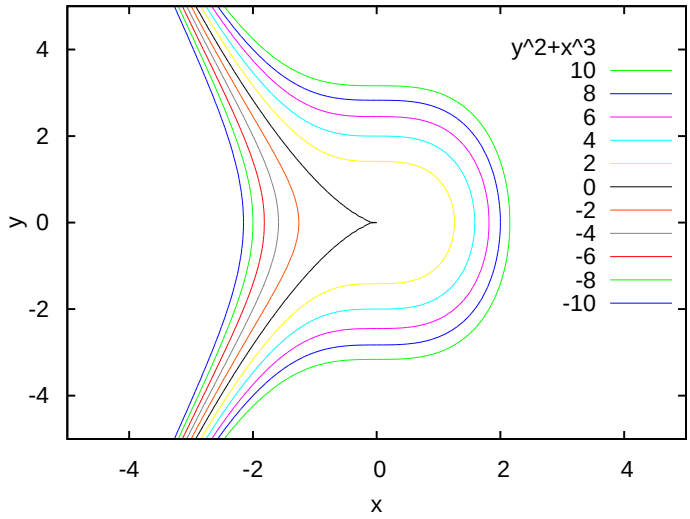
Последняя опция позволяет управлять числом и значениями линий уровня. Возможны следующие три варианта:

`set cntrparam levels 10` – будет показано 10 линий уровня, распределённых равномерно на интервале от наименьшего до наибольшего значения

функции в заданной области

`set cntrparam levels increment i1, istep, i2` – будут показаны уровни с i1 до i2, распределённые с шагом istep.

`set cntrparam levels discrete i1, i2, ..., in` – будут показаны уровни i1, i2, . . . , in.



13.5. Задания к теме.

1. На одном чертеже постройте обе части двухполостного гиперболоида $z = \pm\sqrt{1 + x^2 + y^2}$ ($x \in [-3, 3]$, $y \in [-3, 3]$).

2. Постройте тор: $\begin{cases} x = & (3 - \cos v)\cos u, \\ y = & (3 - \cos v)\sin u, \\ z = & \sin v \end{cases}$ в пределах $u, v \in [0, 2\pi]$.

3. Постройте линии уровня для функции $f(x, y) = e^{-x^2-(y-2)^2} + 3e^{-(x-1)^2-y^2}$

§ 14. Вычисление пределов

14.1. Команда `limit`. Для вычисления пределов в программе Maxima есть команда `limit()`. Найдем первый замечательный предел $\lim\limits_{x\to 0} \dfrac{\sin x}{x}$:

```
--> limit(sin(x)/x, x, 0);
```

(%) 1

Для обозначения плюс/минус бесконечности используются символы `inf/minf`. Найдем второй замечательный предел $\lim\limits_{x\to\infty}\left(1+\dfrac{1}{x}\right)^x$:

```
--> limit((1+1/x)^x, x, inf);
```

(%) e

Можно находить и односторонние пределы. Для этого в аргументах команды `limit()` надо дописать `plus` для правосторонних и `minus` для левосторонних. Вычислим $\lim\limits_{x\to0+0}\dfrac{1}{x}$, $\lim\limits_{x\to0-0}\dfrac{1}{x}$:

```
--> limit(1/x, x, 0, plus);
```

(%) ∞

```
--> limit(1/x, x, 0, minus);
```

(%) $-\infty$

14.2. Задания к теме.

1. Найти пределы:

$$\text{а)}\lim_{x\to0}\frac{\sin4x}{\sqrt{x+1}-1},\quad \text{б)}\lim_{x\to\pi/2}(\sin x)^{\operatorname{tg}^2 x}$$

2. Вычислить

$$\text{а)}\lim_{x\to\pi/2-0}\frac{\sqrt{1+\cos2x}}{\sqrt{\pi}-\sqrt{2x}},\quad \text{б)}\lim_{x\to-\infty}\left(\sqrt{x^2-ax}-\sqrt{x^2+ax}\right).$$

Ответы: **1.** а) 8; б) $\frac{1}{\sqrt{e}}$; **2.** а) $\sqrt{2\pi}$; б) a.

§ 15. Дифференцирование.

15.1. Вычисление производной явной функции. Для нахождения производной в программе Maxima есть команда `diff`. Найдем y' и y'' функции

$y = x^5$:

```
--> f: x^5;
```

(%) x^5

```
--> diff(f, x);
```

(%) $5\,x^4$

```
--> diff(f, x, 2);
```

(%) $20\,x^3$

Второй аргумент этой команды определяет переменную дифференцирования, а третий – порядок производной. Команда `diff` работает и в случае функции многих переменных для нахождения частных производных. Вычислим $\dfrac{\partial^3 f(x,y)}{\partial x^2 \partial y}$ для функции $f(x,y) = x^5 y^3$:

```
--> f: x^5*y^3;
```

(%) $x^5\,y^3$

```
--> diff(f, x, 2, y, 1);
```

(%) $60\,x^3\,y^2$

Вычисленную производную программа Maxima выводит в непреобразованном виде. Поэтому для записи производной в удобном виде полученную производную преобразовывают с помощью команд § 9.

15.2. Нахождение производной неявной функции. По умолчанию все переменные в Maxima считаются независимыми. Поэтому результат выполнения команды

```
--> diff(y, x);
```

будет нулевой

(%) 0

Чтобы декларировать, что одна переменная зависит от другой используется команда `depends`:

```
--> depends(y,x);
```

Теперь результат выполнения команды

```
--> diff(y, x);
```

будет другой:

(%) $\frac{d}{d\,x}\,y$

Это используется при нахождении производной неявной функции. Найдем производную y' неявно заданной функции $x^2+y^2=1$. Зададим вначале ее под именем `f`:

```
--> f: x^2+y^2=1;
```

(%) $y^2+x^2=1$

Производную `f` запишем под именем `g`:

```
--> g: diff(f, x);
```

(%) $2\,y\left(\frac{d}{d\,x}\,y\right)+2\,x=0$

Осталось из равенства `g` выразить производную. Для этого используем команду `solve()`:

```
--> solve(g, diff(y,x));
```

(%) $[\frac{d}{d\,x}\,y=-\frac{x}{y}]$

15.3. Задания к теме.

1. Найти производные функций:

$$y=\sqrt{1+\sin 6x},\ y=\arcsin\frac{x-1}{x},\ s=\ln(\sqrt{e^{2t}+1})-\operatorname{arctg}(e^t).$$

2. Найти производную 6 порядка для функции $y = e^{-x}\sin x$.

3. Найти y' для неявно заданной функции $\operatorname{arctg} y = x + y$.

Ответы: **1.** $\frac{3\cos(6x)}{\sqrt{\sin(6x)+1}}$; $\frac{1}{\sqrt{2x-1}|x|}$; $\frac{(e^t-1)e^t}{e^{2t}+1}$; **2.** $8e^{-x}\cos x$; **3.** $-\frac{y^2+1}{y^2}$.

§ 16. Интегрирование

16.1. Вычисление неопределённых интегралов. Для вычисления интегралов используется команда `integrate()`. Вычислим $\int \ln^3 x dx$:

```
--> integrate(log(x)^3, x);
```

$$(\%)\quad x\left(\log(x)^3 - 3\log(x)^2 + 6\log(x) - 6\right)$$

Также, как и в случае дифференцирования, результат интегрирования Maxima выводит в непреобразованном виде. Поэтому для получения результата в более удобном виде полученную функцию преобразовывают с помощью команд § 9. Вычислим, например, $\int \frac{\sin^3 x}{\cos^3 x} dx$:

```
--> f: sin(x)^3/cos(x)^3;
```

```
--> F: integrate(f, x);
```

$$(\%)\quad \frac{\log\left(\sin(x)^2 - 1\right)}{2} - \frac{1}{2\sin(x)^2 - 2}$$

```
--> F1: trigsimp(F);
```

$$(\%)\quad \frac{\cos(x)^2\log\left(-\cos(x)^2\right) + 1}{2\cos(x)^2}$$

```
--> F2: expand(F1);
```

$$(\%)\quad \frac{\log\left(-\cos(x)^2\right)}{2} + \frac{1}{2\cos(x)^2}$$

Если результат зависит от значений постоянных, Maxima спросит об этом

пользователя. Так, в следующем примере, необходимо ввести “p” (positive) в случае a>0 или “n” (negative) в случае a<0.

```
--> integrate(1/(x^2-a), x);

Is a positive or negative? p;
```

$$(\%)\quad \frac{\log\left(\frac{2\,x-2\,\sqrt{a}}{2\,x+2\,\sqrt{a}}\right)}{2\,\sqrt{a}}$$

Выбираем другой вариант:

```
--> integrate(1/(x^2-a), x);

Is a positive or negative? n;
```

$$(\%)\quad \frac{\operatorname{atan}\left(\frac{x}{\sqrt{-a}}\right)}{\sqrt{-a}}$$

16.2. Аналитическое вычисление определённых интегралов. В случае определённого интеграла в команде `integrate` дописываем пределы интегрирования. Найдем $\int\limits_0^2 \frac{1}{x^3+1}dx$:

```
--> integrate(1/(x^3+1), x, 0, 2);
```

$$(\%)\quad \frac{\log(3)}{6}+\frac{\pi}{2\,\sqrt{3}}$$

Пределы интегрирования могут быть и бесконечными. Вычислим интеграл $\int\limits_{-\infty}^{\infty} e^{-x^2}dx$:

```
--> integrate(exp(-x^2), x, minf, inf);
```

$$(\%)\quad \sqrt{\pi}$$

Некоторые интегралы Maxima может записать через специальные функции. Для вычисления численного значения таких интегралов используется команда `numer`. Вычислим $S=\int\limits_2^3 \frac{1}{\ln x}dx$:

```
--> S: integrate(1/log(x), x, 2, 3);
```

$$\text{(\%)}\quad \text{gamma\_incomplete}\,(0, -\log(2)) - \text{gamma\_incomplete}\,(0, -\log(3))$$

```
--> S, numer;
```

$$\text{(\%)}\quad 1.1184$$

16.3. Численное вычисление определённых интегралов. Если определённый интеграл не вычисляется, то Maxima просто запишет его в символьном виде. Можно найти приближённое значение интеграла численными методами. Это можно сделать командой `quad_qags()`. Вычислим $\int_1^2 \ln x\, e^{x^2} dx$:

```
--> quad_qags(log(x)*exp(x^2), x, 1, 2);
```

$$\text{(\%)}\quad [8.057,\quad 8.945\,10^{-14},\quad 21,\quad 0]$$

Maxima выведет список, состоящий из четырёх чисел. Первое число, 8.057 – приближённое значение интеграла, второе, $8.945 \cdot 10^{-14}$ – точность вычисления, третье, 21 – число использованных разбиений, четвёртое, 0 – код ошибки. Если код ошибки равен нулю, значит проблем при вычислении интеграла не возникло.

В Maxima есть также пакет `romberg` для вычисления определённого интеграла по методу Ромберга. Продемонстрируем его работу на том же примере.

```
--> romberg(log(x)*exp(x^2), x, 1, 2);
```

$$\text{(\%)}\quad 8.057$$

16.4. Задания к теме.

1. Вычислить неопределённые интегралы:

$$\int \frac{dx}{x^4 + ax^3}, \qquad \int \frac{b^2 - x^2}{(x^2 + b^2)^4} dx, \qquad \int \sin^6 x\, dx.$$

2. Вычислить определённые интегралы:

$$\int_a^{a\sqrt{3}} \frac{dx}{a^2+x^2}, \qquad \int_1^{\infty} \frac{1}{x(1+x^2)}dx, \qquad \int_0^{\pi} \ln(1+\sin^2 x)dx.$$

Ответы: **1.** $\frac{1}{a^3}\ln\frac{x}{x+a} + \frac{2x-a}{2a^2x^2}$; $\frac{1}{4b^5}\operatorname{arctg}\frac{x}{b} + \frac{x(3x^4+8b^2x^2+9b^4)}{12b^4(x^2+b^2)^3}$; $\frac{5}{16}x - \frac{15}{64}\sin 2x + \frac{3}{64}\sin 4x - \frac{1}{192}\sin 6x$; **2.** $\frac{\pi}{12a}$; $\frac{\ln 2}{2} \approx 0.3465$; 1.1827.

§ 17. Аналитическое решение дифференциальных уравнений и систем

17.1. Решение дифференциального уравнения первого порядка. По умолчанию все переменные в Maxima являются независимыми. Поэтому, перед тем как приступить к заданию и решению дифференциального уравнения $y' = f(x, y)$, необходимо указать, что переменная y зависит от x:

```
--> depends(y,x);
```

(%) [y (x)]

Решим дифференциальное уравнение $y' = 2 - y$. Запишем его под именем `eqn`:

```
--> eqn: diff(y,x)=2-y;
```

(%) $\dfrac{d}{d\,x}\,y = 2 - y$

Для решения дифференциального уравнения используется команда `ode2()`. Решение запишем под именем `sol`:

```
--> sol: ode2(eqn, y, x);
```

(%) $y = e^{-x}\,(2\,e^{x} + \%c)$

Постоянную c можно найти, если даны начальные условия. Для этого

есть команда `ic1()`. Решение с начальными условиями ($x = 0$, $y = 0$) запишем под тем же именем `sol`:

```
--> sol: ic1(sol, x=0, y=0);
```

(%) $y = e^{-x}\,(2\,e^{x} - 2)$

Построим график полученной функции на отрезке $x \in [0, 5]$. Команда `rhs(sol)` выдает только правую часть выражения `sol` после знака "=":

```
--> wxplot2d(rhs(sol), [x,0,5])$
```

17.2. Решение дифференциального уравнения второго порядка. Для решения дифференциального уравнения второго порядка используется та же команда `ode2()`. Две постоянные находятся из начальных условий ($x = x_0$, $y = y_0$, $y' = y'_0$) командой `ic2()`. Приведем процесс решения уравнения $y'' = y$ с начальными условиями $y(0) = 2$, $y'(0) = -1$:

```
--> eqn: diff(y,x,2)=y;
```

(%) $\dfrac{d^2}{d\,x^2}\,y = y$

```
--> sol: ode2(eqn, y, x);
```

(%) $y = \%k1\,e^{x} + \%k2\,e^{-x}$

```
--> sol: ic2(sol, x=0, y=2, diff(y,x)=-1);
```

(%) $y = \dfrac{e^{x}}{2} + \dfrac{3\,e^{-x}}{2}$

Строим график полученной функции:

```
--> wxplot2d(rhs(sol), [x,0,2])$
```

17.3. Решение линейных дифференциальных уравнений и систем с помощью преобразования Лапласа. Для решения линейного

дифференциального уравнения (или системы линейных дифференциальных уравнений) можно использовать команду `desolve()`. Предварительно необходимо задать начальные условия с помощью команды `atvalue()`. Схему решения продемонстрируем на двух примерах.

Пример 1. Решить уравнение $y''' + y'' = 6x + e^x$ при начальных условиях $y(0) = 1$, $y'(0) = 2$, $y''(0) = 3$.

Задаем исходное уравнение под именем `eqn`:

```
--> eqn: diff(f(x),x,3)+diff(f(x),x,2)=6*x+exp(x);
```

Искомую функцию обозначаем как `f(x)`. Заметим, что аргумент в скобках писать в данном случае обязательно. Теперь зададим начальные условия:

```
--> atvalue(f(x), x=0, 1);

--> atvalue(diff(f(x),x), x=0, 2);

--> atvalue(diff(f(x),x,2), x=0, 3);
```

Далее, находим `f(x)` командой `desolve()`:

```
--> desolve(eqn, f(x));
```

(%) $\mathrm{f}(x) = \dfrac{e^x}{2} + \dfrac{17e^{-x}}{2} + x^3 - 3x^2 + 10x - 8$

Пример 2. Решить систему уравнений

$$\begin{cases} x' = y - x + e^t, \\ y' = x - y + e^t, \end{cases} \qquad \begin{cases} x(0) = a, \\ y(0) = b. \end{cases}$$

Задаем уравнения:

```
--> eqn1: diff(x(t),t)=y(t)-x(t)+exp(t);

--> eqn2: diff(y(t),t)=x(t)-y(t)+exp(t);
```

И начальные условия:

```
--> atvalue(x(t), t=0, a);
```

```
--> atvalue(y(t), t=0, b);
```

Решаем систему:

```
--> desolve([eqn1, eqn2], [x(t),y(t)]);
```

$$(\%)\quad [\mathrm{x}\,(t) = e^t - \frac{(b-a)\,e^{-2\,t}}{2} + \frac{b+a-2}{2},$$
$$\mathrm{y}\,(t) = e^t + \frac{(b-a)\,e^{-2\,t}}{2} + \frac{b+a-2}{2}]$$

17.4. Задания к теме.

1. Решить уравнение $y' + xy = xy^2$, если $y(0) = 2$.

2. Решить уравнение $y''y + (y')^2 = 0$ при начальных условиях: $y(0) = 2$, $y'(0) = 1$.

3. Решить уравнение $y''' - 4y' = 16x^3$ при начальных условиях: $y(0) = 0$, $y'(0) = 0$, $y''(0) = 2$.

Ответы: **1.** $y = \frac{2}{2-e^{x^2/2}}$; **2.** $y = 2\sqrt{x+1}$; **3.** $y = e^{2\,x} + e^{-2\,x} - x^4 - 3x^2 - 2$.

§ 18. Численное решение дифференциальных уравнений и систем

Для численного решения дифференциальных уравнений и систем необходимо предварительно загрузить пакет `dynamics`:

```
--> load(dynamics);
```

18.1. Численное решение ДУ первого порядка. Численное интегрирование дифференциального уравнения методом Рунге – Кутта выполняется командой `rk()`. Предварительно дифференциальное уравнение необходимо записать в виде $y' = f(x, y)$, то есть выразить производную в явном виде.

Численное решение рассмотрим на примере. Проинтегрируем уравнение $y' = 1 - 2y$. Правую часть уравнения сохраним под именем `eqn`:

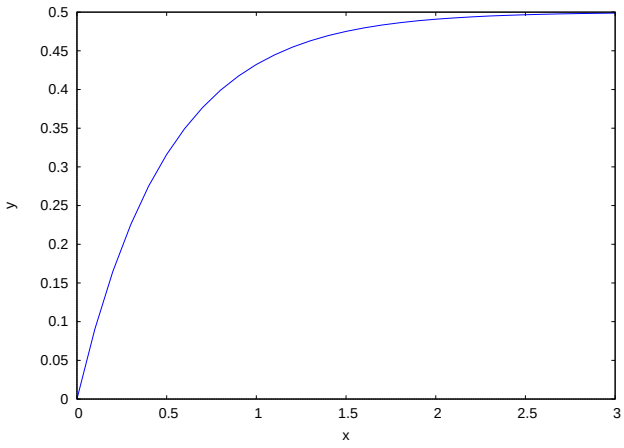
```
--> eqn: 1-2*y;
```

Интегрирование проведем на отрезке от 0 до 3 с шагом 0.1 при начальном условии $y(0) = 0$. Результат запишем под именем `pts`:

```
--> pts: rk(eqn, y, 0, [x, 0, 3, 0.1]);
```

В результате получим 31 пару чисел $[x_i, y_i]$. Их можно изобразить в виде графика командой `plot2d` с опцией `discrete`:

```
--> wxplot2d([discrete, pts])$
```



18.2. Численное решение систем дифференциальных уравнений. Системы дифференциальных уравнений решаются с использованием той же команды `rk()`. Уравнения, искомые переменные и начальные условия к ним перечисляются в квадратных скобках.

Пример. Решить систему

$$\begin{cases} x' = x - xy, \\ y' = xy - y. \end{cases} \qquad \begin{cases} x(0) = 2, \\ y(0) = 1. \end{cases}$$

на отрезке $t \in [0, 5]$.

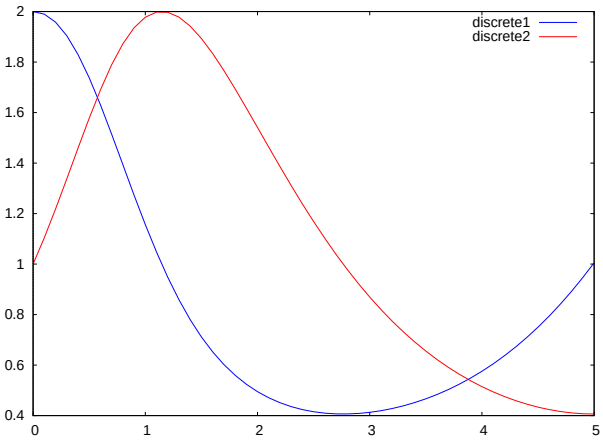
```
--> eq1: x-x*y; eq2: -y+x*y;

--> pts: rk([eq1,eq2], [x,y], [2,1], [t,0,5,0.1]);
```

В списке `pts` элементами являются тройки чисел $[t_i, x_i, y_i]$. Для того, чтобы построить графики функций $x(t)$ и $y(t)$, необходимо создать списки, состоящие из пар чисел $[t_i, x_i]$ и $[t_i, y_i]$. Для этого используем команду `makelist`. Сохраним списки таких пар под именами `xt` и `yt`:

```
--> xt: makelist([pts[i][1], pts[i][2]],
    i, 1, length(pts))$

--> yt: makelist([pts[i][1], pts[i][3]],
    i, 1, length(pts))$
```

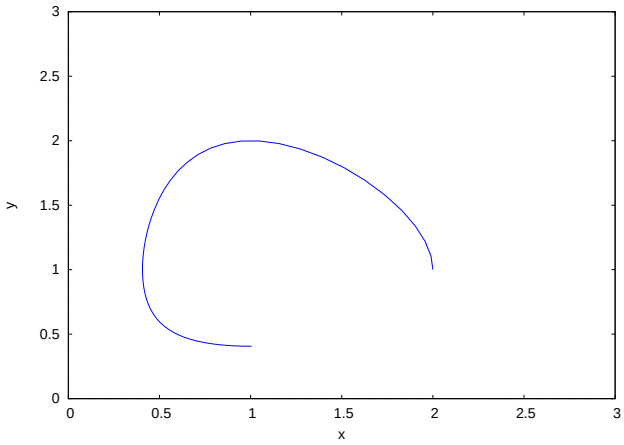
Теперь построим оба графика на одном чертеже:

```
--> wxplot2d([[discrete, xt], [discrete, yt]])$
```



Можно построить график в фазовой плоскости. Для этого предварительно создадим набор пар точек $[x_i, y_i]$ под именем `yx`:

```
--> yx: makelist([pts[i][2], pts[i][3]],
    i, 1, length(pts))$

--> wxplot2d([[discrete, yx]], [x,0,3], [y,0,3])$
```



18.3. Построение векторного поля направлений траекторий в фазовой плоскости. В случае системы из двух дифференциальных уравнений первого порядка с помощью программы Maxima можно построить векторное поле направлений в фазовой плоскости. Это делается с использованием команды `plotdf`.

Возьмем ту же систему дифференциальных уравнений

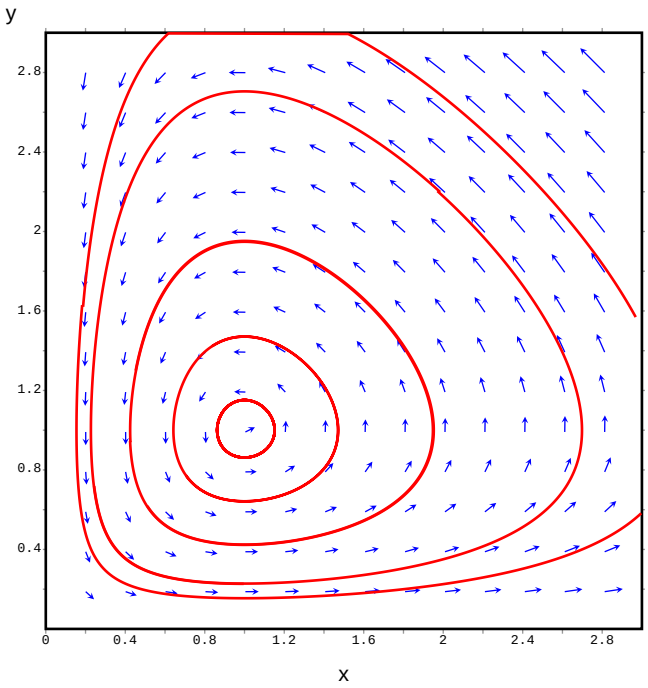
```
--> eq1: x-x*y; eq2: -y+x*y;
```

В фазовой плоскости нарисуем траекторию, начинающуюся в точке $x = 2, y = 1$ в направлении роста t:

```
--> plotdf([eq1, eq2], [x,y], [x, 0, 3], [y, 0, 3],
    [trajectory_at, 2, 1], [direction, forward])$
```

С использованием мыши в окне с графиком фазовой плоскости можно нарисовать и другие траектории.

Если в исходной системе уравнений содержится один или несколько параметров, можно проводить исследование влияния этих параметров на получаемые поля траекторий в фазовой плоскости.



Рассмотрим систему с двумя параметрами:

$$\begin{cases} x' = x - xy - ax^2, \\ y' = xy - y - c. \end{cases}$$

```
--> eq1:x-x*y-a*x^2; eq2:-y+x*y-c;

--> plotdf([eq1,eq2], [x,y], [x, 0, 3], [y, 0, 3],
    [sliders,"a=-1:1,c=-1:1"]);
```

В нижней части экрана с фазовой плоскостью появляется два бегунка, позволяющие изменять значения параметров a и c.

18.4. Задания к теме

1. Проинтегрировать уравнение $y' = x \sin y - 1$ на отрезке $x \in [0, 10]$ при начальном условии $y(0) = 2$.

2. Проинтегрировать систему на отрезке $t \in [0, 15]$:

$$\begin{cases} x' = x - xy - 0.1x^2, \\ y' = -y + xy - 0.1y^2, \end{cases} \qquad \begin{cases} x(0) = 2, \\ y(0) = 1. \end{cases}$$

§ 19. Элементы программирования

19.1. Оператор цикла for. Синтаксис оператора имеет вид:

`for` *имя*`:` *нач_знч* `thru` *кон_знч* `step` *шаг* `while` *услов* `do` *выражение*`$`

Если шаг равен 1, конструкцию `step` можно опустить. Также в зависимости от условий окончания цикла обычно используется одна из конструкций `thru` или `while`, а другая опускается.

Примеры: Необходимо вывести на экран квадраты чисел от 1 до 10:

```
--> for i:1 thru 10 do print(i^2)$
```

Необходимо вывести на экран числа от 1 до 150, являющиеся полными квадратами:

```
--> for i:1 while i^2<=150 do print(i^2)$
```

В качестве условий конструкции `while` можно использовать операторы сравнения =, <, >, <= (меньше или равно), >=, # (не равно) и логические операторы `and`, `or`, `not`.

Для того, чтобы в теле цикла выполнить несколько выражений, необходимо их записать в круглых скобках через запятую. Найдем, например, первые 10 чисел Фибоначчи.

```
--> a:1;b:1;
```

```
--> for i:1 thru 10 do (print(a), c:a+b, a:b, b:c)$
```

Можно вкладывать циклы друг в друга. Создадим матрицу A размера 5×5, у которой элементы $a_{ij} = i^2 - j$. Вначале создадим нулевую матрицу заданного размера:

```
--> A: zeromatrix(5,5)$
```

Теперь, используя два вложенных цикла, изменим элементы матрицы:

```
--> for i: 1 thru 5 do for j: 1 thru 5 do A[i,j]:i^2-j$
```

Выведем результат на экран:

```
--> A;
```

$$(\%)\quad \begin{pmatrix} 0 & -1 & -2 & -3 & -4 \\ 3 & 2 & 1 & 0 & -1 \\ 8 & 7 & 6 & 5 & 4 \\ 15 & 14 & 13 & 12 & 11 \\ 24 & 23 & 22 & 21 & 20 \end{pmatrix}$$

Сосчитаем сумму элементов полученной матрицы. Для этого используем следующие команды в пакете Maxima:

```
--> s:0$

--> for i: 1 thru 5 do for j: 1 thru 5 do s:s+A[i,j]$

--> s;

(%) 200
```

19.2. Условный оператор if. Синтаксис оператора имеет следующий вид:

`if` *условие* `then` *выражение1* `else` *выражение2* `$`

Конструкцию `else` можно опустить. В условии используются операторы сравнения и логические операторы, как и в конструкции `while`. Вместо одного выражения можно использовать несколько, взяв их в круглые скобки и записав через запятые.

С использованием оператора `if`, присвоим переменной с наибольшее из значений переменных $a = 10$ и $b = \pi^2$:

```
--> a:10; b:%pi^2;
```

```
--> if a>b then c:a else c:b;
```

Для полученной ранее матрицы A найдем число отрицательных элементов. Для этого используем счётчик k.

```
--> k:0$

--> for i: 1 thru 5 do for j: 1 thru 5 do
    if A[i,j]<0 then k:k+1$

--> k;

(%) 5
```

Найдем наибольший amax элемент матрицы A и соответствующие ему индексы imax и jmax:

```
--> amax:A[1,1];imax:1;jmax:1$

--> for i: 1 thru 5 do for j: 1 thru 5 do
    if A[i,j]>amax then (amax:A[i,j],imax:i,jmax:j)$

--> amax;imax;jmax;

(%) 24

(%) 5

(%) 1
```

19.3. Задания к теме

1. Вычислить сумму членов ряда, меньших 10^{-6}:

$$\frac{1}{1+1}+\frac{1}{4+1}+\frac{1}{9+1}+\frac{1}{16+1}+\dots$$

2. Создать матрицу B размера 9×9, элементы которой $b_{ij}=\frac{i-j^2}{i+j}$. Вычислить сумму квадратов элементов матрицы B.

3. Найти произведение положительных элементов матрицы B.

4. Чему равна разность между наибольшим и наименьшим элементами матрицы B?

Ответы: **1.** ≈ 1.0756735; **2.** ≈ 852.06; **3.** $\approx 4.81 \cdot 10^{-5}$; **4.** $\frac{44}{5}$.

§ 20. Построение в пакете рисования draw

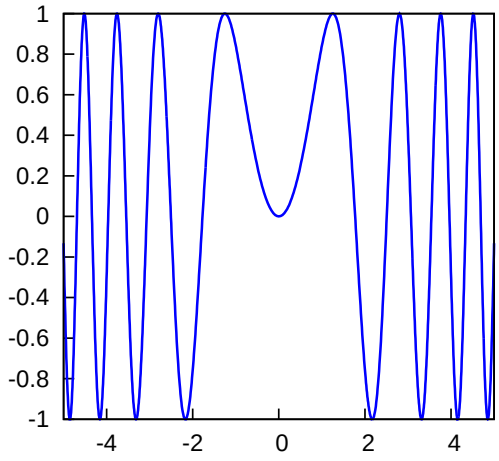
Для построения графиков и поверхностей ранее были рассмотрены команды `plot2d`, `plot3d` и `contour_plot`. Кроме этих команд в Maxima есть другой, более гибкий для рисования пакет `draw`. Перед использованием любой команды из этого пакета необходимо один раз загрузить его командой

```
--> load(draw)$
```

20.1. Построение графиков функций. Для построения графиков функций и двумерных чертежей используются `draw2d()` и `wxdraw2d()`. Первая строит график в отдельном окне, вторая – встраивает в лист вычислений.

Для явно заданных функций есть опция `explicit()`. Так, команда для построения графика функции $y = \sin(x^2)$ на отрезке $x \in [-5, 5]$ будет выглядеть так:

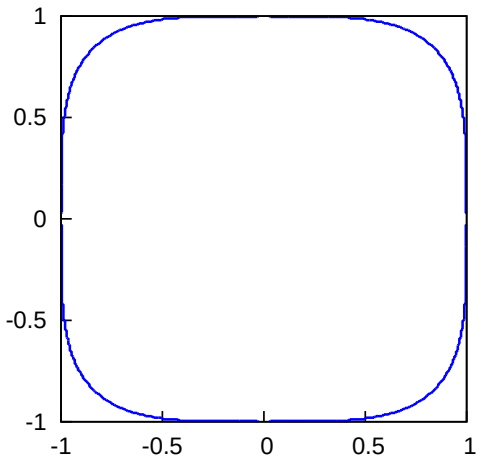
```
--> wxdraw2d(explicit(sin(x^2),x,-5,5))$
```



Интервал изменения ординаты программа выбирает сама, исходя из минимальных и максимальных значений функции. Этот интервал можно задать и самому, с помощью опций, об этом будет сказано позднее.

Для неявно заданных функций используется опция `implicit()`. Так, команда для построения графика функции $x^4 + y^4 = 1$ будет иметь вид:

```
--> wxdraw2d(implicit(x^4+y^4=1, x, -1, 1, y, -1, 1))$
```



Для построения графика функции, заданной в полярной системе, в пакете `draw` есть опция `polar()`. Построим график функции $\rho = 1 - \sin\varphi$:

```
--> wxdraw2d(polar(1-sin(t),t,-%pi,%pi))$
```

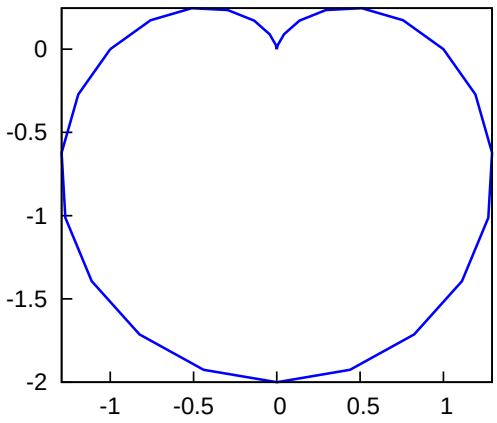
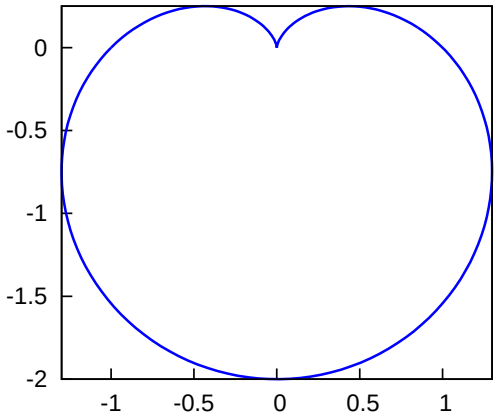


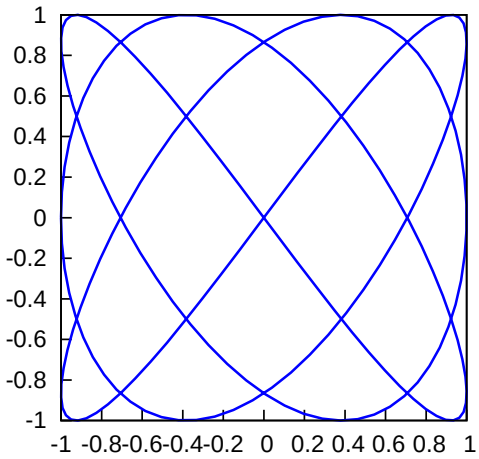
График функции, заданной в полярной системе (а также параметрически), строится по умолчанию по 30 точкам. Для более точного графика это число опорных точек может быть увеличено опцией `nticks`. Так, команда для построения графика той же самой функции по 200 точкам имеет вид:

```
--> wxdraw2d(nticks=200, polar(1-sin(t),t,-%pi,%pi))$
```



Для построения графика параметрически заданной функции есть опция `parametric()`. Построим график функции $x = \cos 3t$, $y = \sin 3t$ для границ изменения параметра $t \in [-\pi, \pi]$:

```
--> wxdraw2d(nticks=200,
    parametric(cos(3*t),sin(4*t),t,-%pi,%pi))$
```

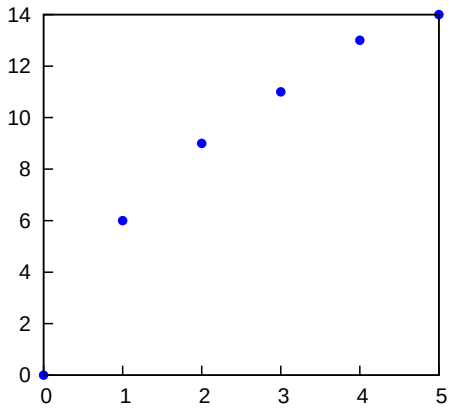


Для построения графика по дискретным точкам есть опция points. Как и для команды `plot2d`, исходные данные могут быть заданы двумя способами: в виде списка, каждый элемент которого представляет собой пару координат $[x_i, y_i]$, или в виде двух списков, содержащих абсциссы и ординаты точек.

Команда рисования для первого способа:

```
--> pts:[[0,0],[1,6], [2,9], [3,11], [4,13], [5,14]];
```

```
--> wxdraw2d(point_type=circle, points(pts)$
```



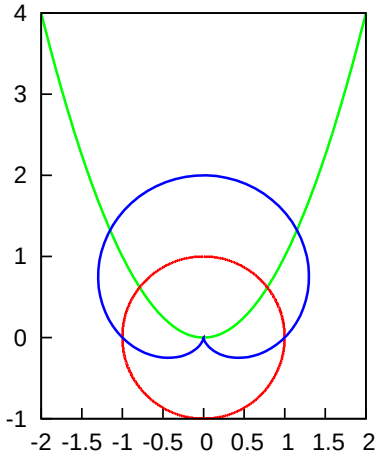
И для второго способа:

```
--> xpt:[0,1,2,3,4,5]; ypt:[0,6,9,11,13,14];

--> wxdraw2d(point_type=circle, points(xpt, ypt))$
```

По умолчанию точки не соединяются линиями, их можно соединить с помощью опции `points_joined=true`.

Для построения на одном чертеже нескольких графиков исходные функции записывают через запятую. Пример:

```
--> wxdraw2d(line_width=2, color=green,
    explicit(x^2,x,-2,2), color=red,
    implicit(x^2+y^2=1,x,-1,1,y,-1,1),
    color=blue, nticks=100, polar(1+sin(t),t,-%pi,%pi))$
```



20.2. Опции команды draw2d. Команда `draw2d` имеет множество опций, позволяющих настроить внешний вид чертежа. Опции, меняющие вид графика функции, необходимо записывать перед командой рисования. Вот некоторые из таких опций:

`key="`*имя*`"` – задает имя графика функции.

`line_width=`*число* – задает толщину линии. По умолчанию равна 1.

`line_type=`dots/solid – задает тип линии. По умолчанию имеет значение `solid` (сплошная линия).

`color=`*цвет* – задает цвет графика. Возможные цвета:

Также цвет можно задать в виде стандартного шестнадцатеричного кода `#rrggbb`.

`nticks=`*число* – задает число контрольных точек, по которым строится график функции. Используется для функций, заданных неявно, параметрически и в полярной системе координат.

`point_size=`*число* – задает размер точки при построении по опции `points`.

`point_type=`*число или значение* – задает вид символа, которым рисуются точки. Возможные числа и соответствующие им значения:

−1 none	0 dot	1 plus
2 multiply	3 asterisk	4 square
5 filled_square	6 circle	7 filled_circle
8 up_triangle	9 filled_up_triangle	10 down_triangle
11 filled_down_triangle	12 diamant	13 filled_diamant

`points_joined=true` – рисуемые точки соединяются линиями.

Следующие опции используются для оформления всего чертежа:

`title="`*имя*`"` – задается название чертежа.

`grid=true` – рисуется сетка. По умолчанию она отключена.

`background_color=`*цвет* – задает цвет фона.

`file_name="`*имя*`"` – задает имя выходного файла. Используется совместно с командой `terminal`.

`terminal = '`*тип* – используется для сохранения графика в файл. Возможные типы файлов: png, jpg, gif, eps, pdf и svg.

`dimensions = [`*число, число*`]` – определяет размеры изображения в выходном файле. Указывается число пикселей при выводе в растровые форматы или сотые доли сантиметра при выводе в векторные форматы. По умолчанию имеет значение `[600,500]`.

`proportional_axes=xy` – масштаб осей координат будет одинаковым.

Следующие опции используются для оформления оси x. Аналогичные команды есть и для оси y.

`xaxis=true` – рисуется ось x. По умолчанию она отключена.

`xlabel="`*имя*`"` – задает имя оси x.

`logx=true` – ось x будет логарифмической.

`xaxis_color=`*цвет* – задается цвет оси x. По умолчанию `black`.

`xaxis_type=dots/solid` – задает тип оси x. По умолчанию: `dots`.

`xaxis_width=`*число* – толщина оси x. Значение по умолчанию: 1.

`xrange=[xmin,xmax]` – задаются границы изменения оси x. По умолчанию эти границы выбираются автоматически.

`xtics` – опция, меняющая внешний вид отсечек на оси x. По умолчанию имеет значение `auto`. Примеры использования:

`xtics=none` – отсечки не рисуются.

`xtics=0.5` – отсечки рисуются с шагом 0.5.

`xtics=[0.2,0.1,0.6]` – отсечки будут на отрезке $[0.2, 0.6]$ с шагом 0.1.

`xtics={0.2,0.3,0.6}` – рисуются отсечки в точках $0.2, 0.3, 0.6$.

`xtics_axis=true` – отсечки рисуются на оси x. По умолчанию отсечки рисуются на границе.

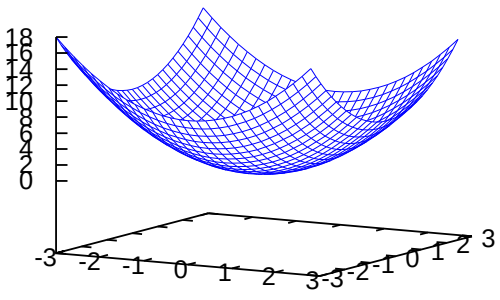
`xtics_rotate=true` – подписи к отсечкам поворачиваются на 90°.

`axis_bottom=false` – не рисуется ось по нижней границе. Аналогичные действия имеют команды `axis_top=false`, `axis_left=false`, `axis_right=false`.

20.3. Построение поверхностей и линий в пространстве. Для построения трёхмерных графиков используется команда `draw3d()`.

Для построения графика явно заданной функции двух переменных $z = f(x, y)$ используется опция `explicit`:

```
--> draw3d(explicit(x^2+y^2, x, -3, 3, y, -3, 3))$
```



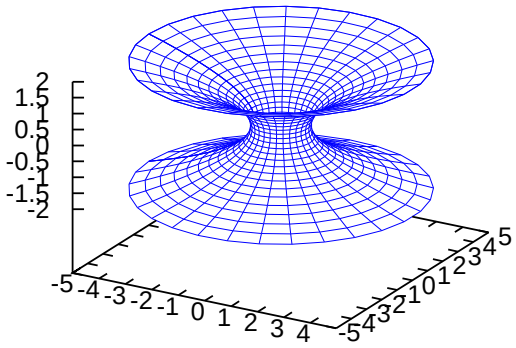
Если функция заданно неявно, т.е. в виде $F(x, y, z) = 0$, то для ее построения пишем опцию `implicit`:

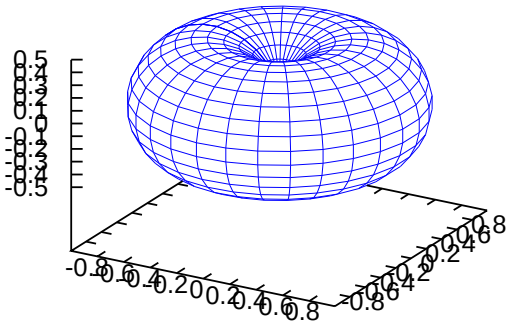
```
--> draw3d(nticks=100, implicit(x^2+y^2+z^2=1,
    x, -1, 1, y, -1, 1, z, -1, 1))$
```

Точность построения будет зависеть от опции `nticks`, задающей число опорных точек.

Для построения графика функции, заданной в цилиндрической или сферической системах координат, есть опции `cylindrical` и `spherical`. Построим график функций $\rho(z,t) = 1 + z^2$ в цилиндрической и $\rho(t,h) = \sin(h)$ в сферической системах:

```
--> draw3d(cylindrical(1+z^2,z,-2,2,t,-%pi,%pi))$

--> draw3d(spherical(sin(h),t,-%pi,%pi,h,0,%pi))$
```



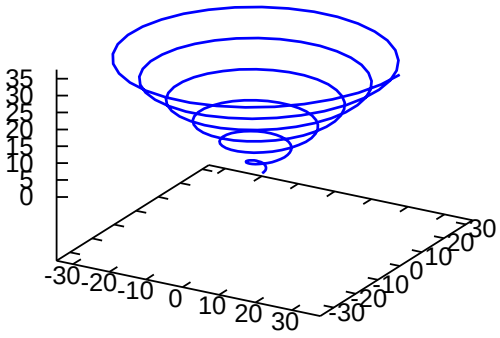


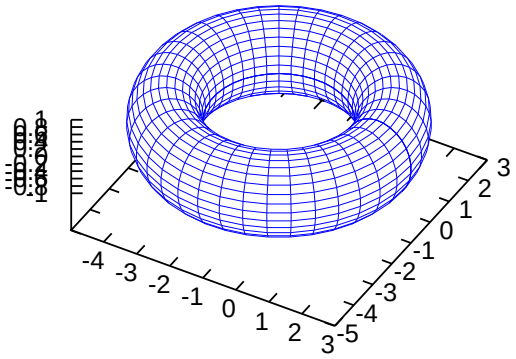
Можно строить и параметрически заданные трёхмерные линии и поверхности. Для этого есть опции `parametric()` и `parametric_surface()`. Построим линию и поверхность, заданную в виде

$$\begin{cases} x = t\cos t, \\ y = t\sin t, \\ z = t, \end{cases} \quad t \in [0, 12\pi] \qquad \begin{cases} x = (3 - \cos v)\cos u, \\ y = (3 - \cos v)\sin u, \\ z = \sin v, \end{cases} \quad u, v \in [0, 2\pi]$$

```
--> draw3d(nticks=200, line_width=2,
    parametric(t*cos(t),t*sin(t),t,t,0,12*%pi))$
```

```
--> draw3d(xu_grid=40, yv_grid=30, parametric_surface(
    (3-cos(v))*cos(u),(3-cos(v))*sin(u), sin(v),
    u, 0, 2*%pi, v, 0, 2*%pi))$
```





Опции `xu_grid`, `yv_grid` задают число точек, по которым находится функция при построении графика.

Для отображения дискретного набора точек есть опция points. Она имеет те же самые параметры, что и для команды `draw2d`. Ниже приведены примеры отображения набора точек для двух разных способов задания данных: в виде списка, каждый элемент которого представляет собой тройку координат $[x_i, y_i, z_i]$, или в виде трёх списков, содержащих абсциссы, ординаты и аппликаты точек.

Первый способ:

```
--> pts:[[0,0,0], [3,6,1], [8,2,2], [2,3,7]];

--> draw3d(point_type=circle, points(pts)$
```

Второй способ:

```
--> xpt:[0,3,8,3]; ypt:[0,6,2,3]; zpt:[0,1,2,7];

--> draw2d(point_type=circle, points(xpt, ypt, zpt))$
```

20.4. Опции команды draw3d. Как уже было сказано, опции `xu_grid=`*число*, `yv_grid=`*число* задают число точек разбиения точек по x и y (для явных функций) и по u и v (для параметрически заданных). Также действуют опции, описанные выше для команды `draw2d`. Другие опции, которые могут использоваться в команде `draw3d`:

`enhanced3d=true/false` – определяет, будет ли поверхность закрашенной или нет. Также можно выбрать более сложный способ закраски поверхности, например, с учётом света, падающего на поверхность. Такой пример приведен в конце параграфа.

`palette=[цвет, цвет, ..., цвет]` – определяет цвета раскраски поверхности. Названия цветов представлены выше.

`wired_surface=true/false` – отвечает за прорисовку сетки.

`surface_hide=true/false` – отвечает за показ/скрытие невидимой для данного угла обзора частей поверхности.

`view=[angle1, angle2]` – исходный угол обзора. Углы измеряются в градусах в диапазоне от $0°$ до $360°$. Заметим, что после показа графика угол обзора можно менять с помощью мышки.

`proportional_axes=xyz` – равный масштаб всех осей координат.

`contour=`*значение* – отвечает за прорисовку линий уровня. По умолчанию имеет значение `none`. Другие возможные значения следующие: `base` – линии рисуются в плоскости xy, `surface` – линии рисуются на самой поверхности, `both` – линии рисуются и там и там. Если значение равно `map`, то будет нарисована лишь плоскость xy с линиями уровня (аналог `contour_plot`).

`contour_levels=`*значение* – позволяет выбирать отображаемые линии уровня. Примеры с вариантами использования этой опции:

`contour_levels=10` – будет нарисовано 10 линий уровня с равномерным шагом между минимальным и максимальным значениями функции.

`contour_levels=[-8, 2, 8]` – будут нарисованы уровни для значений функции от -8 до 8 с шагом 2.

`contour_levels={-8, 2, 5, 8}` – будут нарисованы уровни для значе-

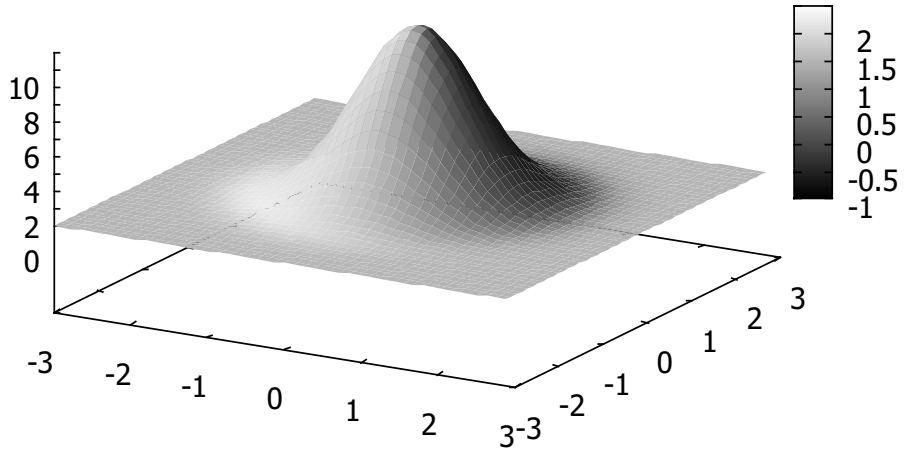
ний функции -8, 2, 5 и 8.

`colorbox=true/false` – отвечает за показ цветовой схемы.

`axis_3d=true/false` – отвечает за прорисовку на чертеже осей координат.

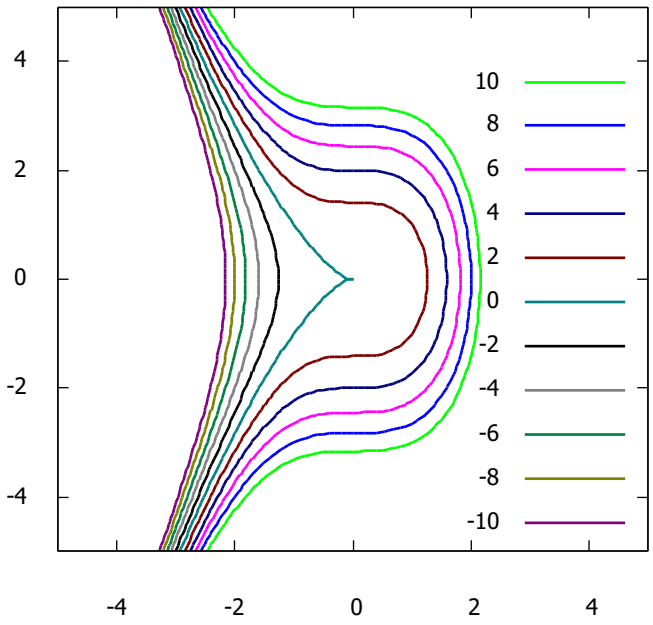
В следующем примере показан график, раскрашенной с учётом отражения света, падающего на поверхность.

```
--> f:10*exp(-x^2-y^2)$

--> fn:(diff(f,x) + diff(f,y) + 1)/sqrt(diff(f,x)^2 +
    diff(f,y)^2 + 1)$

--> draw3d(palette=[black, white], xu_grid=50, yv_grid=50,
    enhanced3d=fn, explicit(f, x ,-3, 3, y, -3, 3))$
```



Пример построения линий уровня функции $f(x, y) = x^3 + y^2$:

```
--> draw3d(xu_grid=100, yv_grid=100, line_width=2,
    explicit(x^3+y^2, x ,-5, 5, y, -5, 5), contour=map,
    contour_levels=[-10,2,10])$
```



20.5. Задания к теме.

Выполните задания параграфов 12.5 и 13.5, пользуясь командами draw.

§ 21. Аппроксимация числовых данных

В программе Maxima есть пакет `lsquares`, с помощью которого можно производить аппроксимацию числовых данных методом наименьших квадратов. Как и остальные пакеты, вначале его необходимо загрузить:

```
--> load(lsquares)$
```

Также загрузим пакет `draw` для графического представления получаемых результатов и зададим требуемую точность вычислений.

```
--> fpprintprec:3; load(draw)$
```

21.1. Линейная зависимость от одной переменной. Введем исходные данные задачи. Для пакета `lsquares` они должны быть представлены в виде матрицы, каждая строчка которой представляет собой пару чисел (x_i, y_i):

```
--> P: matrix([0,3], [3,6], [6,16], [8,17], [10,47],
    [11,56], [13,56], [14,59], [16,85], [17,90], [18,116],
    [20,146], [22,189], [23,202], [25,251])$
```

Предположим, что между ними есть линейная зависимость, т. е. $y = Ax + B$. Найдем коэффициенты A и B, аппроксимирующие исходные данные наилучшим образом:

```
--> lsquares_estimates(P, [x,y], y = A*x+B, [A,B]);
```

(%) $[[A = \frac{112171}{11594}, B = -\frac{252762}{5797}]]$

Первый аргумент команды `lsquares_estimates` – исходная матрица данных, второй аргумент – список переменных, третий – аппроксимирующее уравнение, содержащее переменные и параметры, четвертый аргумент – список искомых параметров. Результат выполнения данной команды является списком.

Немного изменим данную команду, во-первых, добавим в конец опцию `numer` для записи результатов в десятичном виде, во-вторых, результат выполнения сохраним под именем D:

```
--> D: lsquares_estimates(P, [x,y], y = A*x+B, [A,B]),
    numer;
```

Теперь "вытащим" из списка D сами числовые значения параметров. Для этого используем команду `rhs`, которая оставляет от выражения только правую часть после знака ''='':

```
--> A0:rhs(D[1][1]); B0:rhs(D[1][2]);
```

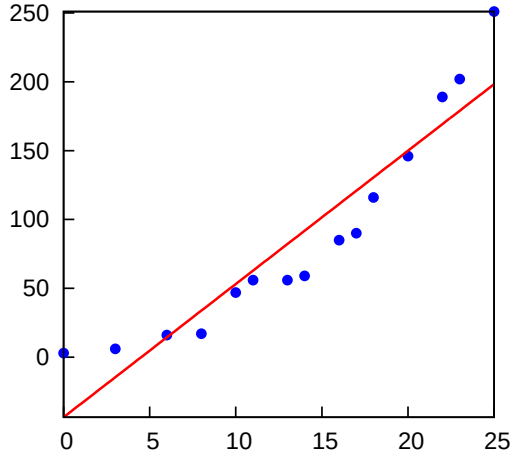
(%) 9.67

(%) -43.6

Исходные данные и найденную линейную зависимость представим в гра-

фическом виде:

```
--> draw2d(point_type=circle, points(P), color=red,
    line_width=3, explicit(A0*x+B0, x, 0, 25));
```



21.2. Другие фунциональные зависимости от одной переменной. По рисунку видно, что исходные данные плохо аппроксимируются линейной зависимостью. Рассмотрим другие виды зависимостей: квадратичную $y = Ax^2 + Bx + C$ и показательную $y = Ae^{Bx}$. Вид функции зависит от рассматриваемой задачи. Например, если исходные данные представляют собой численность популяции, то лучше использовать показательную функцию.

Для нахождения параметров квадратичной зависимости выполним команды:

```
--> D: lsquares_estimates(P, [x,y], y = A*x^2+B*x+C,
    [A,B,C]), numer;

--> A1:rhs(D[1][1]); B1:rhs(D[1][2]); C1:rhs(D[1][3]);
```

(%) 0.473

(%) $-$ 2.48

(%) 9.6

```
--> draw2d(point_type=circle, points(P), color=red,
    line_width=3, explicit(A1*x^2+B1*x+C1, x, 0, 25));
```

Для нахождения параметров показательной зависимости напишем:

```
--> D: lsquares_estimates(P, [x,y], y = A*exp(B*x), [A,B],
    initial=[1,0.2], iprint=[-1,0], tol=1e-5);
```

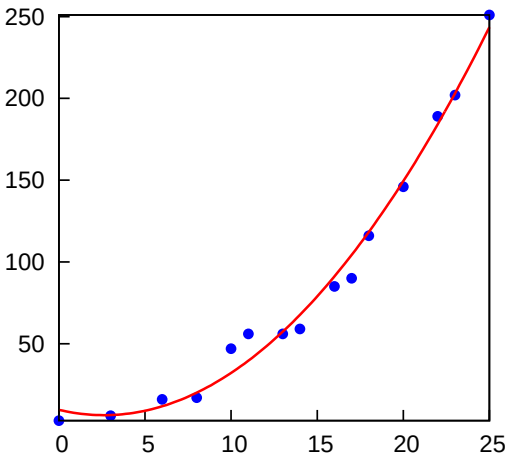
Для данной зависимости параметры находятся численно, в исполняемой команде могут понадобиться дополнительные параметры. Опция `initial` задает начальное приближение для искомых параметров задачи ($A = 1, B = 0.2$), опция `iprint=[n,m]` управляет выводимой на экран информацией. Если число n положительно, будет выводиться результат оптимизации после каждой n-ой итерации, если $n = -1$, то результаты итераций выводиться не будут. От числа m зависит объем вывода другой служебной информация, при $m = 3$ она будет выведена в полном объеме, при $m = 0$ вывода на экран не будет. Последняя опция `tol` задает требуемую точность, в нашем примере она равна 10^{-5}

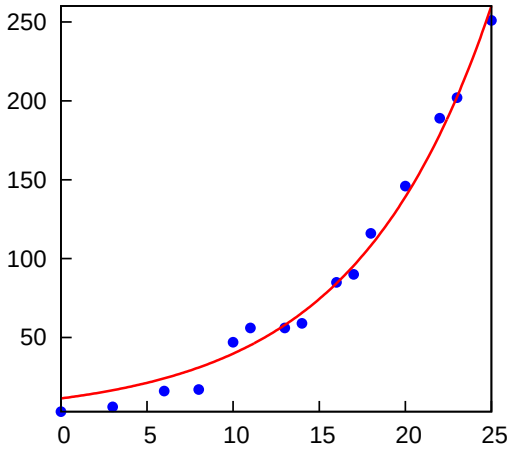
```
--> A2:rhs(D[1][1]); B2:rhs(D[1][2]);
```

```
(%) 11.4
```

```
(%) 0.125
```

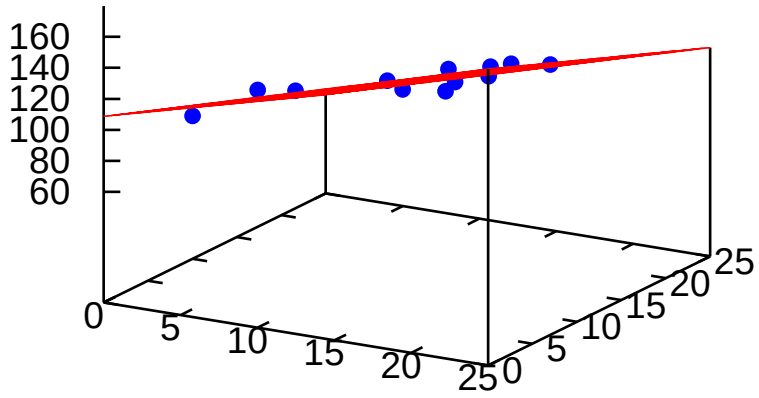
```
--> draw2d(point_type=circle, points(P), color=red,
    line_width=3, explicit(A2*exp(B2*x), x, 0, 25))$
```





21.3. Зависимости от нескольких переменных. Решение задачи аппроксимации в случае нескольких переменных проводится по той же схеме. Рассмотрим решение задачи нахождения линейной зависимости $z = Ax + By + C$ для исходных данных, записанных в виде троек чисел (x_i, y_i, z_i).

```
--> M: matrix([4,25,68], [13,16,101], [8,25,82], [9,6,123],
    [0,10,81], [19,13,137], [24,2,174], [10,0,142],
    [9,24,78], [5,25,64], [21,7,149], [25,7,163])$

--> D: lsquares_estimates(M, [x,y,z], z = A*x+B*y+C,
    [A,B,C]), numer;

--> A3:rhs(D[1][1]); B3:rhs(D[1][2]); C3:rhs(D[1][3]);

--> draw3d(point_type=circle, points(M), color=red,
    explicit(A3*x+B3*y+C3, x, 0, 25, y, 0, 25))$
```



§ 22. Основные команды программы Maxima

Список основных математических функций

Запись в Maxima	Функция	Описание
`abs(x)`	$\|x\|$	модуль числа
`sqrt(x)`	$\sqrt{x}$	квадратный корень
`exp(x)`	e^x	экспонента
`log(x)`	$\ln x$	натуральный логарифм
`sin(x)`	$\sin x$	тригонометрические функции
`cos(x)`	$\cos x$	
`tan(x)`	$\operatorname{tg} x$	
`cot(x)`	$\operatorname{ctg} x$	
`asin(x)`	$\arcsin x$	обратные тригонометрические функции
`acos(x)`	$\arccos x$	
`atan(x)`	$\operatorname{arctg} x$	
`acot(x)`	$\operatorname{arcctg} x$	

Команды преобразования выражений

`expand(`*выражение*`);` – раскрытие скобок.

`factor(`*выражение*`);` – разбиение на множители.

`ratsimp(`*выражение*`);`
`radcan(`*выражение*`);`
`trigsimp(`*выражение*`);`
– упрощение выражения.

Решение уравнений

`solve(`$f(x) = 0$`, x);` – решение уравнения $f(x) = 0$.

`solve([`$f(x, y) = 0$`,` $g(x, y) = 0$`], [x, y]);` – решение системы уравнений.

`find_root(`$f(x) = 0$`, x,` x_a`,` x_b`);` – численное решение уравнения $f(x) = 0$ на отрезке $x \in [x_a, x_b]$.

Построение графиков

`plot2d(`$f(x)$`, [x,`x_a`,`x_b`], [y,`y_a`,`y_b`])$` – построение графика функции $y = f(x)$ в прямоугольнике $x \in [x_a, x_b]$, $y \in [y_a, y_b]$.

`plot2d([`$f(x)$`, `$g(x)$`], [x,`x_a`,`x_b`])$` – построение графиков двух функций $y = f(x)$ и $y = g(x)$ для $x \in [x_a, x_b]$.

`plot2d([discrete, pts])$` – построение графика по набору пар чисел $[x_i, y_i]$, записанных под именем `pts`.

`plot3d(`$f(x,y)$`, [x,`x_a`,`x_b`], [y,`y_a`,`y_b`], [z,`z_a`,`z_b`])$` – построение поверхности $z = f(x, y)$ для $x \in [x_a, x_b]$, $y \in [y_a, y_b]$, $z \in [z_a, z_b]$.

Математический анализ

`limit(`$f(x)$`, x, `x_a`);` – нахождение предела $f(x)$ при $x \to x_a$.

`diff(`$f(x)$`, x);` – нахождение производной функции $f(x)$.

`diff(`$f(x)$`, x, `k`);` – нахождение k-той производной $f(x)$.

`integrate(`$f(x)$`, x);` – нахождение интеграла от $f(x)$.

`integrate(`$f(x)$`, x, `x_a`, `x_b`);` – нахождение определенного интеграла от функции $f(x)$ по отрезку $[x_a, x_b]$.

`quad_qags(`$f(x)$`, x, `x_a`, `x_b`);` – численное нахождение определенного интеграла.

Дифференциальные уравнения

`sol: ode2(`$f(x, y, \texttt{diff}(y, x)) = 0$`, y, x);` – нахождение решения `sol` дифференциального уравнения $f(x, y, y') = 0$.

`ic1(sol, x=`x_0`, y=`y_0`);` – нахождение постоянной в решении `sol` из начального условия $y(x_0) = y_0$.

`pts: rk(`$f(x,y)$`, y, `y_0`, [x,`x_0`,`x_1`,`h`]);` – численное решение уравнения $y' = f(x, y)$ на отрезке $x \in [x_0, x_1]$ при условии $y(x_0) = y_0$. Решение записывается под именем списка `pts`, его элементами являются пары чисел $[x_i, y_i]$, где x_i меняется с x_0 до x_1 с шагом h.

ЛИТЕРАТУРА

1. Берков Н. Математический практикум с применением пакета Maxima: Учебное пособие. М.: РИЦ МГИУ, 2008, 89 с.
2. Берлянд М.Е. Прогноз и регулирование загрязнений атмосферы. Л.: Гидрометеоиздат, 1985, 272 с.
3. Бэйли Р. Математика в биологии и медицине. М.: Мир, 1970.
4. Гильдерман Ю.И. Лекции по высшей математике для биологов. Новосибирск: Наука, 1974.
5. Гроссман С., Тернер Дж. Математика для биологов. М.: Высшая школа, 1983. 384 с.
6. Семенова Е. Е., Кудрявцева Е.В. Математические методы в экологии: Сборник задач и упражнений, Петрозаводск: Изд. ПетрГУ, 2005, 130 с.
7. Жегалов В.И. Обыкновенные дифференциальные уравнения в научных теориях, Казанское математическое общество, 2003, 100 с.
8. Жигулев В.Н. Динамика неустойчивостей. М.: МФТИ,1996, 344 с.
9. Мун Ф. Хаотические колебания. М.: Мир, 1990, 312 с.
10. Пайтген Х.–О., Рихтер П.Х. Красота фракталов: образы динамических систем. М.: Мир, 1993, 176 с.
11. Ризниченко Г. Ю., Рубин А. Б. Математические модели биологических продукционных процессов. М.: Изд–во МГУ, 1993, 302 с.
12. Ризниченко Г.Ю. Математические модели в биофизике и экологии. Иж.: ИКИ, 2003, 184 с.
13. Замай С.С., Якубайлик О.Э. Модели оценки и прогноза загрязнения атмосферы промышленными выбросами в информационно-аналити

ческой системе природоохранных служб крупного города: Учеб. пособие. Красноярск: Краснояр. гос. ун-т., 1998, 109 с.

14. Свирежев Ю.М., Логофет Д.О. Устойчивость биологических сообществ. М.: Наука, 1978, 352 с.

15. Скалецкая Е.И., Фрисман Е.Я., Шапиро А.П. Дискретные модели динамики численности популяций и оптимизация промысла. М.: Наука, 1979, 166 с.

16. Стахин Н. Основы работы с системой аналитических (символьных) вычислений Maxima: Учебное пособие– [Электронный ресурс], 2008, 86 с.

17. Страшкраба М., Гнаук А. Пресноводные экосистемы. Математическое моделирование. М.: Мир, 1989.

18. Фейгенбаум М. Универсальность поведения нелинейных систем. // Успехи физических наук, 1983. Т.141. №2. С.343–374.

19. Фрисман Е. Я. Странные аттракторы в простейших моделях динамики численности популяции с возрастной структурой. // Доклады Академии Наук, 1994. Т.338. №2. С.282–286.

20. Фролов Ю.П. Введение в математическое моделирование биологических процессов.Часть 2. Организмы и популяции. Самара: Изд.Самарский университет, 1994.

21. Шапиро А.П., Луппов С.П. Рекуррентные уравнения в теории популяционной биологии. М.: Наука, 1983, 134 с.