

Лабораторная работа №2.

Оптимизация многомерных функций с помощью ГА

Цель работы: модификация представления хромосомы и операторов рекомбинации ГА для оптимизации многомерных функций. Графическое отображение результатов оптимизации.

Общие сведения

Представление хромосомы

При работе с оптимизационными задачами в непрерывных пространствах вполне естественно представлять гены напрямую вещественными числами. В этом случае хромосома есть вектор вещественных чисел. Их точность будет определяться исключительно разрядной сеткой той ЭВМ, на которой реализуется real-coded алгоритм. Длина хромосомы будет совпадать с длиной вектора-решения оптимизационной задачи, иначе говоря, каждый ген будет отвечать за одну переменную. Генотип объекта становится идентичным его фенотипу.

Вышесказанное определяет список основных преимуществ real-coded алгоритмов:

1. Использование непрерывных генов делает возможным поиск в больших пространствах (даже в неизвестных), что трудно делать в случае двоичных генов, когда увеличение пространства поиска сокращает точность решения при неизменной длине хромосомы.
2. Одной из важных черт непрерывных ГА является их способность к локальной настройке решений.

3. Использование RGA для представления решений удобно, поскольку близко к постановке большинства прикладных задач. Кроме того, отсутствие операций кодирования/декодирования, которые необходимы в BGA, повышает скорость работы алгоритма.

Появление новых особей в популяции канонического ГА обеспечивают несколько биологических операторов: отбор, скрещивание и мутация. В качестве операторов отбора особей в родительскую пару здесь подходят любые известные из BGA: рулетка, турнирный, случайный. Однако операторы скрещивания и мутации в классических реализациях работают с битовыми строками. Необходимы реализации, учитывающие специфику real-coded алгоритмов.

Также рекомендуется использовать стратегию элитизма – лучшая особь сохраняется отдельно и не стирается при смене эпох, принимая при этом участие в отборе и рекомбинации.

Операторы кроссинговера

Оператор скрещивания непрерывного ГА, или кроссовер, порождает одного или нескольких потомков от двух хромосом. Собственно говоря, требуется из двух векторов вещественных чисел получить новые векторы по каким-либо законам. Большинство real-coded алгоритмов генерируют новые векторы в окрестности родительских пар. Далее представлены простые и популярные кроссоверы.

Пусть $C1=(c11,c21,...,cn1)$ и $C2=(c12,c22,...,cn2)$ – две хромосомы, выбранные оператором селекции для скрещивания. После формулы для некоторых кроссоверов приводится рисунок – геометрическая интерпретация его работы. Предполагается, что $sk1 \leq sk2$ и $f(C1) \geq f(C2)$.

Плоский кроссовер (flat crossover): создается потомок $H=(h_1,...,h_k,...,h_n)$, где h_k , ($k=1,...,n$) – случайное число из интервала $[c_{k1},c_{k2}]$.

Простейший кроссовер (simple crossover): случайным образом выбирается число k из интервала $\{1,2,...,n-1\}$ и генерируются два потомка $H1=(c_{11},c_{21},...,c_{k1},c_{k+12},...,c_{n2})$ и $H2=(c_{12},c_{22},...,c_{k2},c_{k+11},...,c_{n2})$.

Арифметический кроссовер (arithmetical crossover): создаются два потомка $H1=(h_{11},...,h_{n1})$, $H2=(h_{12},...,h_{n2})$, где $h_{k1}=w*c_{k1}+(1-w)*c_{k2}$, $h_{k2}=w*c_{k2}+(1-w)*c_{k1}$, $k=1,...,n$, w либо константа (равномерный арифметический кроссовер) из интервала $[0;1]$, либо изменяется с увеличением эпох (неравномерный арифметический кроссовер).

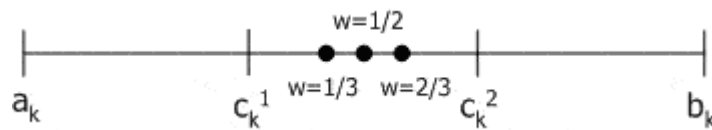


Рис. 2.1. Арифметический кроссовер.

Геометрический кроссовер (geometrical crossover): создаются два потомка $H1=(h_{11},...,h_{n1})$, $H2=(h_{12},...,h_{n2})$, где $h_{k1}=(c_{k1})w*(c_{k2})(1-w)$, $(c_{k2})w*(c_{k1})(1-w)$, где w – случайное число из интервала $[0;1]$.

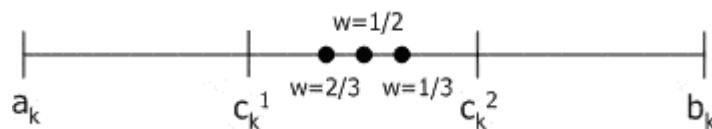


Рис. 2.2. Геометрический кроссовер.

Смешанный кроссовер (blend, BLX-alpha crossover): генерируется один потомок $H=(h_1,...,h_k,...,h_n)$, где h_k – случайное число из интервала $[c_{min}-l*\alpha, c_{max}+l*\alpha]$, $c_{min}=\min(c_{k1}, c_{k2})$, $c_{max}=\max(c_{k1}, c_{k2})$, $l=c_{max}-c_{min}$. BLX-0.0 кроссовер превращается в плоский.

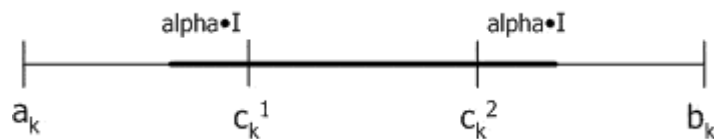


Рис. 2.3. Смешанный кроссовер.

Линейный кроссовер (linear crossover): создаются три потомка $H_q=(h_{1q},...,h_{kq},...,h_{nq})$, $q=1,2,3$, где $h_{k1}=0.5*c_{k1}+0.5*c_{k2}$, $h_{k2}=1.5*c_{k1}-0.5*c_{k2}$, $h_{k3}=-0.5*c_{k1}+1.5*c_{k2}$. На этапе селекции в этом кроссовере отбираются два наиболее сильных потомка.

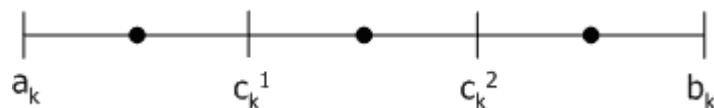


Рис. 2.4. Линейный кроссовер.

Дискретный кроссовер (discrete crossover): каждый ген h_k выбирается случайно по равномерному закону из конечного множества $\{c_{k1}, c_{k2}\}$.

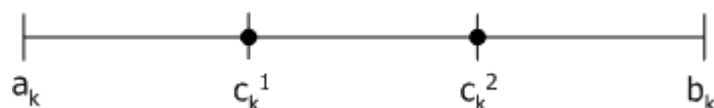


Рис. 2.5. Дискретный кроссовер.

Расширенный линейный кроссовер (extended line crossover): ген $h_k = c_{k1} + w * (c_{k2} - c_{k1})$, w – случайное число из интервала $[-0.25; 1.25]$.



Рис. 2.6. Расширенный линейный кроссовер.

Эвристический кроссовер (Wright's heuristic crossover). Пусть $C1$ – один из двух родителей с лучшей приспособленностью. Тогда $h_k = w * (c_{k1} - c_{k2}) + c_{k1}$, w – случайное число из интервала $[0; 1]$.

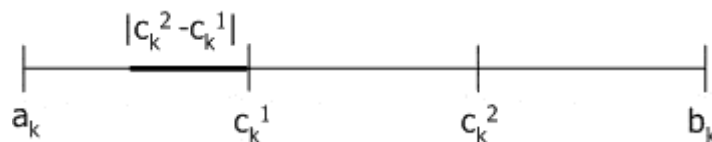


Рис. 2.7. Эвристический кроссовер.

Нечеткий кроссовер (fuzzy recombination, FR-d crossover): создаются два потомка $H1 = (h_{11}, \dots, h_{n1})$, $H2 = (h_{12}, \dots, h_{n2})$. Вероятность того, что в i -том гене появится число v_i , задается распределением $p(v_i) \in \{F(c_{k1}), F(c_{k2})\}$, где $F(c_{k1}), F(c_{k2})$ – распределения вероятностей треугольной формы (треугольные

нечеткие функции принадлежности) со следующими свойствами ($c_{k1} \leq c_{k2}$ и $l = |c_{k1} - c_{k2}|$) таб.2.1.

Параметр d определяет степень перекрытия треугольных функций принадлежности, по умолчанию $d=0.5$.

Таблица 2.1.

Распределение вероятностей	Минимум	Центр	Максимум
$F(c_k^1)$	$c_k^1 - d * l$	c_k^1	$c_k^1 + d * l$
$F(c_k^2)$	$c_k^2 - d * l$	c_k^2	$c_k^2 + d * l$

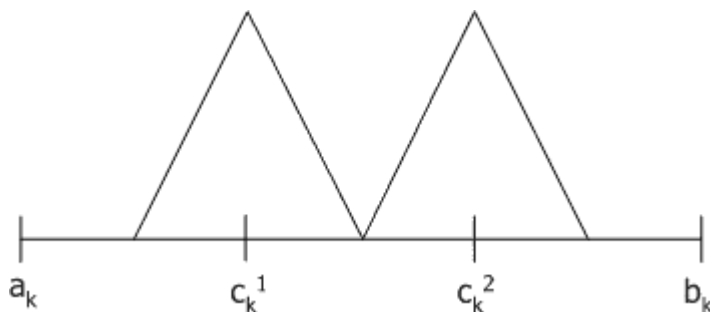


Рис. 2.8. Нечеткий кроссовер.

Мин-максный кроссинговер

Для представления хромосомы числами с плавающей точкой используется практически только мин-максный кроссовер или его модификации:

Для родителей $C_w^t = (c_1, \dots, c_k, \dots, c_{3*(n+1)})$ и $C_v^t = (c'_1, \dots, c'_k, \dots, c'_{3*(n+1)})$

получаются 4 потомка

$$C_1^{t+1} = \alpha * C_w^t + (1 - \alpha) * C_v^t,$$

$$C_2^{t+1} = \alpha * C_v^t + (1 - \alpha) * C_w^t,$$

$$C_3^{t+1} = \min\{c_k, c'_k\},$$

$$C_4^{t+1} = \max\{c_k, c'_k\}.$$

Выбираются два лучших из этих четырех потомков.

Рассмотренные кроссоверы исторически были предложены первыми, однако во многих задачах их эффективность оказывается невысокой. Исключение составляет BLX-кроссовер с параметром $\alpha=0.5$ – он превосходит по эффективности большинство простых кроссоверов. Позднее были разработаны улучшенные операторы скрещивания, аналитическая формула которых и эффективность обоснованы теоретически. Один из таких кроссоверов – SBX.

SBX кроссовер

SBX (англ.: Simulated Binary Crossover) – кроссовер, имитирующий двоичный. Был разработан в 1995 году исследовательской группой под руководством К. Deb'а. Как следует из его названия, он моделирует принципы работы двоичного оператора скрещивания.

SBX кроссовер был получен следующим способом. У двоичного кроссовера было обнаружено важное свойство – среднее значение функции приспособленности оставалось неизменным у родителей и их потомков, полученных путем скрещивания. Затем автором было введено понятие силы поиска кроссовера (search power). Это количественная величина, характеризующая распределение вероятностей появления любого потомка от двух произвольных родителей. Первоначально была рассчитана сила поиска для одноточечного двоичного кроссовера, а затем был разработан вещественный SBX кроссовер с такой же силой поиска. В нем сила поиска характеризуется распределением вероятностей случайной величины β :

$$P(\beta) = \begin{cases} 0.5(n+1)\beta^n, & \text{при } \beta \leq 1, \\ 0.5(n+1)\beta^{-(n+2)}, & \text{при } \beta > 1. \end{cases}$$

Для генерации потомков используется следующий алгоритм, использующий выражение для $P(\beta)$. Создаются два потомка $H_k = (h_{1k}, \dots, h_{jk}, \dots, h_{nk})$, $k=1,2$, где $h_{j1} = 0.5[(1 - \beta_k)c_{j2} + (1 + \beta_k)c_{j2}]$, $h_{j2} = 0.5[(1 + \beta_k)c_{j1} + (1 - \beta_k)c_{j2}]$, $\beta_k \geq 0$ – число, полученное по формуле:

$$\beta(u) = \begin{cases} (2u)^{\frac{1}{n+1}}, & \text{при } u(0,1) \leq 0.5, \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{n+1}}, & \text{при } u(0,1) > 0.5. \end{cases}$$

В формуле $u(0,1)$ – случайное число, распределенное по равномерному закону, $n \in [2,5]$ – параметр кроссовера.

На рисунке приведена геометрическая интерпретация работы SBX кроссовера при скрещивании двух хромосом, соответствующих вещественным числам 2 и 5. Видно, как параметр n влияет на конечный результат: увеличение n влечет за собой увеличение вероятности появления потомка в окрестности родителя и наоборот.

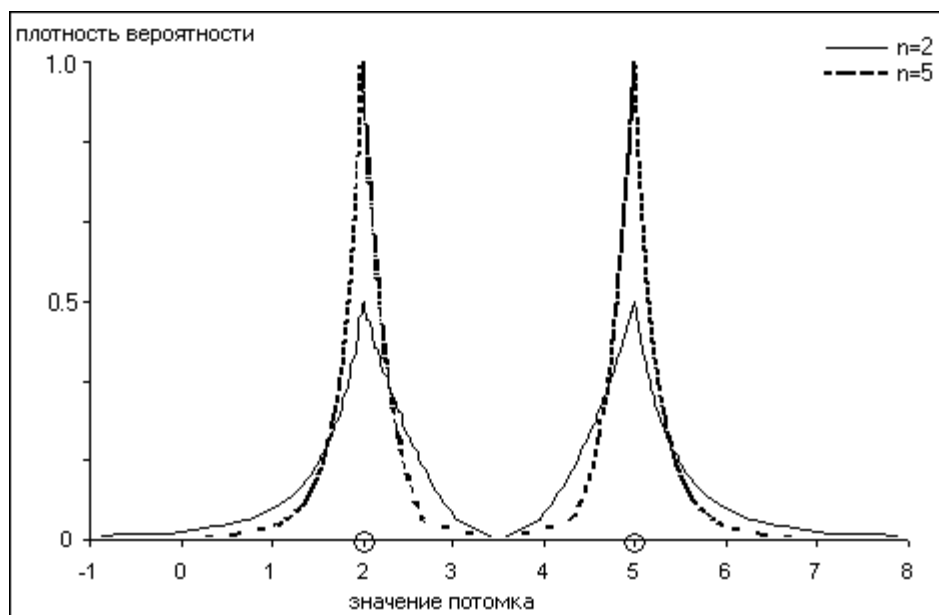


Рис. 2.9. SBX кроссовер.

Эксперименты автора SBX кроссовера показали, что он во многих случаях эффективнее BLX, хотя, очевидно, что не существует ни одного кроссовера, эффективного во всех случаях. Исследования показывают, что использование нескольких различных операторов кроссовера позволяет уменьшить вероятность преждевременной сходимости, т.е. улучшить эффективность алгоритма оптимизации в целом. Для этого могут использоваться специальные стратегии, изменяющие вероятность применения отдельного эволюционного оператора в зависимости от его «успешности», или использование гибридных кроссоверов, которых в настоящее время насчитывается несколько десятков.

Операторы мутации

В качестве оператора мутации наибольшее распространение получили: случайная и неравномерная мутация (random and non-uniform mutation).

При **случайной мутации** ген, подлежащий изменению, принимает случайное значение из интервала своего изменения.

В **неравномерной мутации** из особи случайно выбирается точка c_k (с разрешенными пределами изменения $[c_{kl} c_{kr}]$). Точка меняется на

$$c'_k = \begin{cases} c_k + \Delta(t, c_{kr} - c_k), & \text{при } a = 0, \\ c_k - \Delta(t, c_k - c_{kl}), & \text{при } a = 1, \end{cases} \text{ где}$$

a – случайно выбранное направление изменения,

$\Delta(t, y)$ - функция, возвращающая случайную величину в пределах $[0..y]$ таким образом, что при увеличении t среднее возвращаемое значение увеличивалось:

$$\Delta(t, y) = y(1 - r^{(1-\frac{t}{T})^b}), \text{ где}$$

r – случайная величина на интервале $[0..1]$

t – текущая эпоха работы генетического алгоритма

T – общее разрешенное число эпох алгоритма

b – задаваемый пользователем параметр, определяющий степень зависимости от числа эпох.

Тестовые примеры

Для данного вида задачи существует большое число тестовых примеров – Benchmark-ов. С некоторыми из них можно познакомиться, например, в **[Ошибка! Источник ссылки не найден.]**. Для данных тестов произведено большое число исследований на скорость алгоритма, количество эпох для достижения результата и пр. С результатами этих исследований можно ознакомиться в научной литературе, доступной в Internet.

Многочисленные исследования доказывают, что непрерывные ГА не менее эффективно, а часто гораздо лучше справляются с задачами оптимизации в многомерных пространствах, при этом более просты в реализации из-за отсутствия процедур кодирования и декодирования хромосом.

Порядок выполнения лабораторной работы

1. Создать программу, использующую ГА для нахождения оптимума функции согласно таблице вариантов, приведенной в приложении А. Для всех Benchmark-ов оптимумом является минимум. Программу выполнить на встроенном языке пакета Matlab.
2. Для $n=2$ вывести на экран график данной функции с указанием найденного экстремума, точек популяции. Для вывода графиков использовать стандартные возможности пакета Matlab. Предусмотреть возможность пошагового просмотра процесса поиска решения.
3. Повторить нахождение решения с использованием стандартного Genetic Algorithm toolbox. Сравнить полученные результаты.
4. Исследовать зависимость времени поиска, числа поколений (генераций), точности нахождения решения от основных параметров генетического алгоритма:
 - число особей в популяции
 - вероятность кроссинговера, мутации.Критерий остановки вычислений – повторение лучшего результата заданное количество раз или достижение популяцией определенного возраста (например, 100 эпох).
5. Повторить процесс поиска решения для $n=3$, сравнить результаты, скорость работы программы.

Содержание отчета.

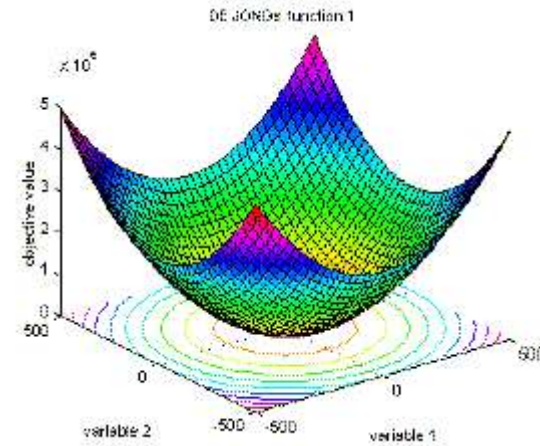
1. Титульный лист установленной формы.
2. Условие задания с вариантом.
3. Распечатанный листинг программы.
4. Распечатка результатов выполнения программы (графиков);
5. Диаграммы исследованных зависимостей.

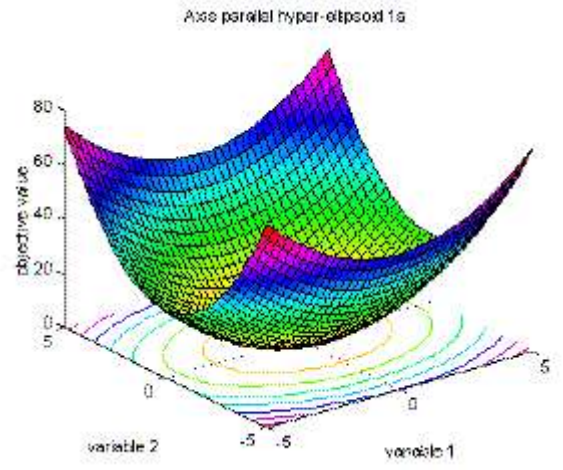
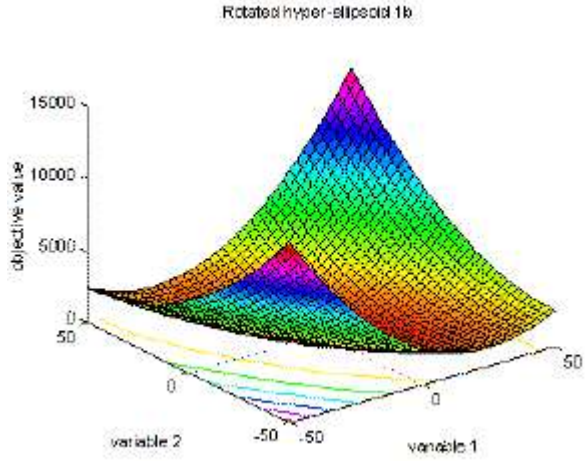
Контрольные вопросы

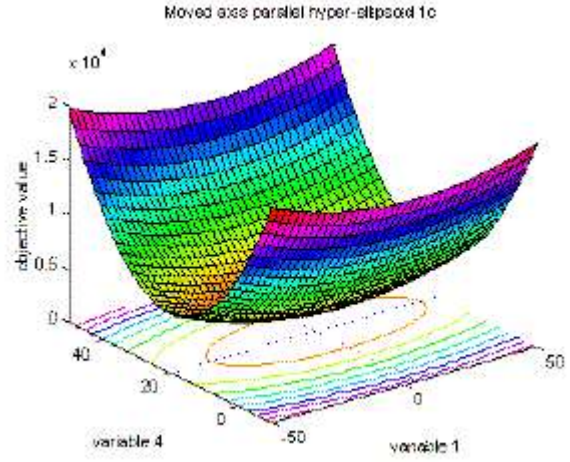
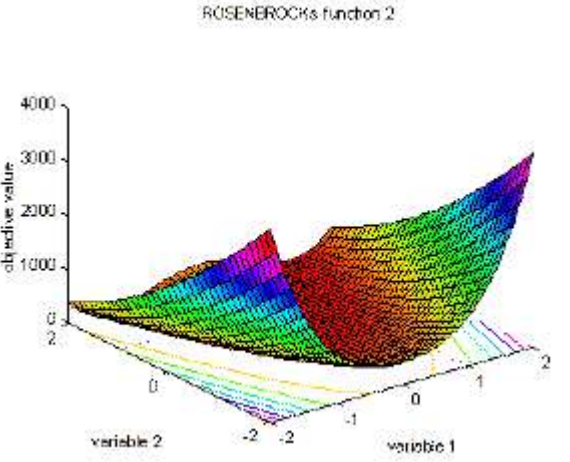
1. Что такое «оптимальность»?
 2. Опишите понятие «оптимизационная задача».
 3. Что такое «критерий оптимизации»?
 4. Что является целью оптимизационной задачи?
 5. Что такое целевая функция в генетических алгоритмах?
 6. Каким образом строится целевая функция?
 7. Дайте понятие экстремума и оптимума целевой функции.
 8. Что такое «локальный и глобальный оптимум»?
 9. Каким образом в генетических алгоритмах осуществляется выбор способа представления решения?
 10. Каким образом определяется эффективность генетического алгоритма?
 11. Приведите основные цели и задачи генетических алгоритмов.
 12. Выделите основные отличительные особенности генетических алгоритмов?
 13. Поясните, как создается начальная популяция альтернативных решений?
 14. Приведите основные понятия и определения генетических алгоритмов.
 15. Опишите методику
- ГА.

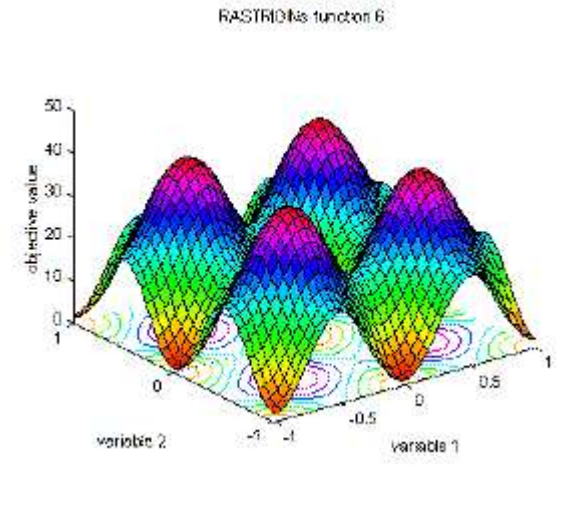
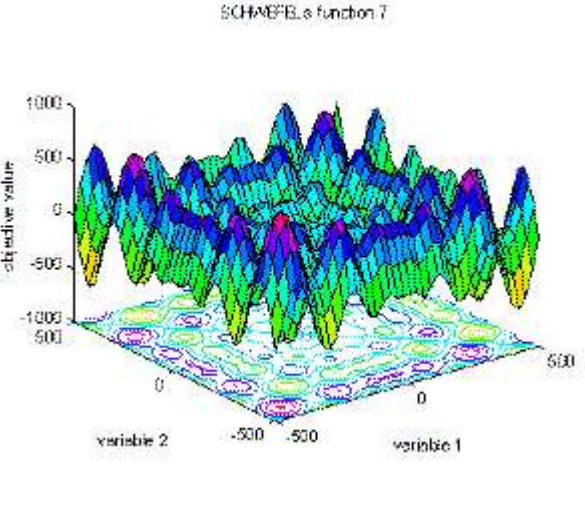
Приложение 1.

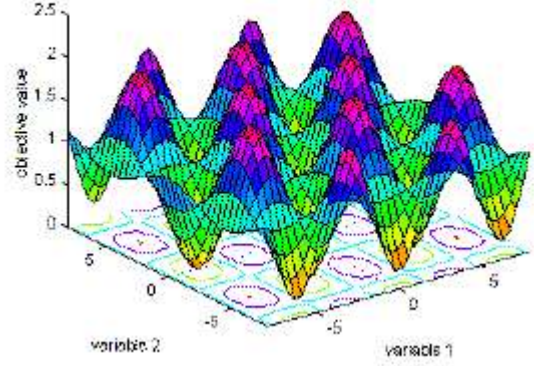
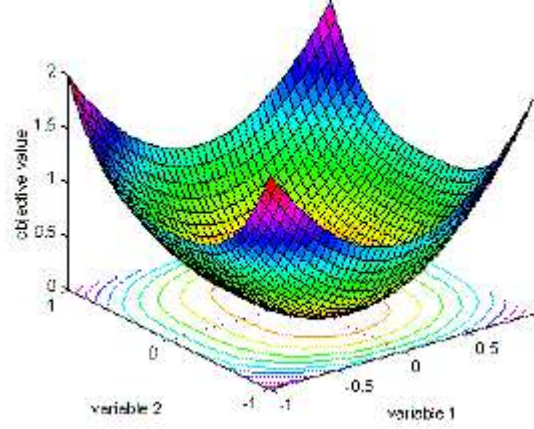
Индивидуальные задания на лабораторную работу №2.

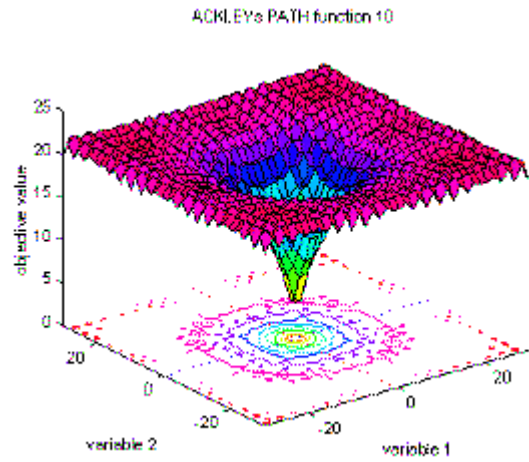
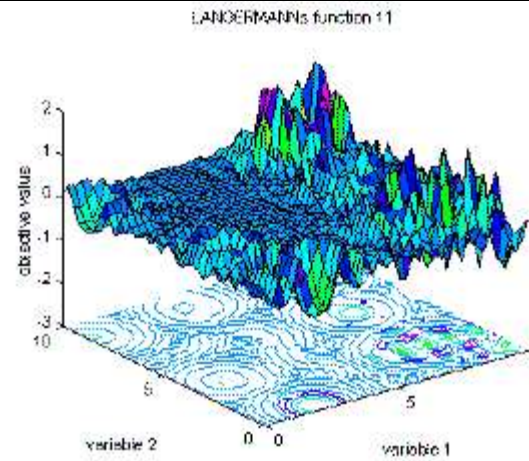
№ вв.	Название	Оптимум	Вид функции	График функции
1	De Jong's function 1	global minimum f(x)=0; x(i)=0, l=1:n.	$f_1(x) = \sum_{i=1}^n x_i^2 \quad -5.12 \leq x_i \leq 5.12$ f1(x)=sum(x(i)^2), i=1:n;	 <p>DE JONG's function 1</p>

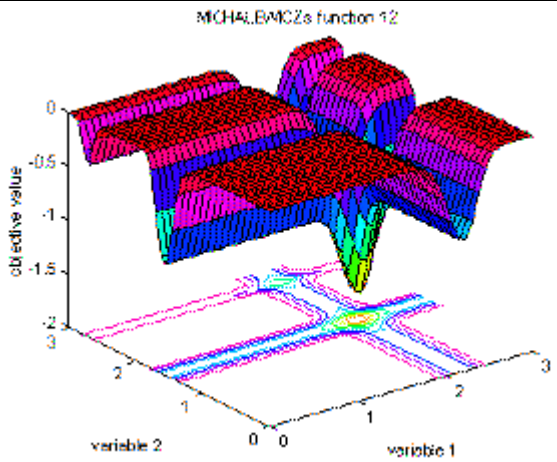
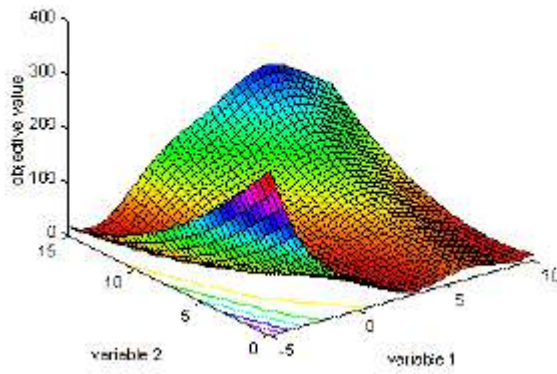
2	Axis parallel hyper-ellipsoid function	global minimum f(x)=0; x(i)= 0, i=1:n.	$f_{1a}(x) = \sum_{i=1}^n i \cdot x_i^2 \quad -5.12 \leq x_i \leq 5.12$ $f1a(x)=\sum(i \cdot x(i)^2),$ $i=1:n;$	 <p>Axis parallel hyper-ellipsoid 1a</p>
3	Rotated hyper-ellipsoid function	global minimum f(x)=0; x(i)=0, i=1:n	$f_{1b}(x) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2 \quad -65.536 \leq x_i \leq 65.536$ $f1b(x)=\sum(\sum(x(j)^2),$ $j=1:i), i=1:n;$	 <p>Rotated hyper-ellipsoid 1b</p>

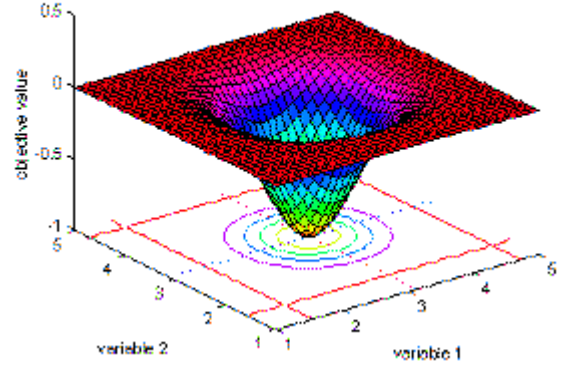
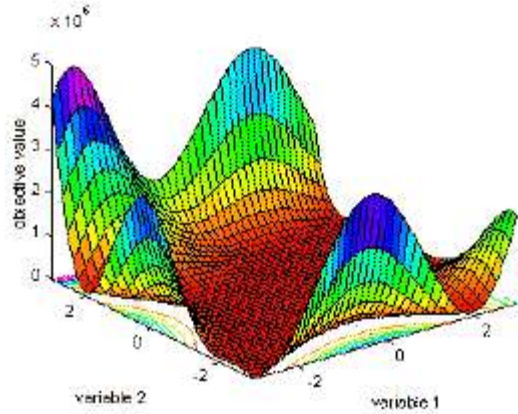
4	Moved axis parallel hyper-ellipsoid function	global minimum $f(x)=0$; $x(i)=5 \cdot i$, $i=1:n$	$f_k(x) = \sum_{i=1}^n 5i \cdot x_i^2 \quad -5.12 \leq x_i \leq 5.12$ $f1c(x)=\text{sum}(5 \cdot i \cdot x(i)^2),$ $i=1:n;$	 <p>Moved axis parallel hyper-ellipsoid 1c</p>
5	Rosenbrock's valley (De Jong's function 2)	global minimum $f(x)=0$; $x(i)=1$, $i=1:n$.	$f_2(x) = \sum_{i=1}^{n-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \quad -2.048 \leq x_i \leq 2.048$ $f2(x)=\text{sum}(100 \cdot (x(i+1)-x(i)^2)^2 + (1-x(i))^2),$ $i=1:n-1;$	 <p>ROSENBR0CK's function 2</p>

6	Rastrigin's function 6	<p>global minimum</p> <p>$f(x)=0$; $x(i)=0$, $i=1:n$.</p>	$f_6(x) = 10 \cdot n + \sum_{i=1}^n (x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i)) \quad -5.12 \leq x_i \leq 5.12$ <p>$f_6(x)=10 \cdot n + \sum_{i=1}^n (x(i)^2 - 10 \cdot \cos(2 \cdot \pi \cdot x(i)))$, $i=1:n$;</p>	<p>RASTRIGIN's function 6</p> 
7	Schwefel's function 7	<p>global minimum</p> <p>$f(x)=n \cdot 418.982$ 9; $x(i)=420.9687$, $i=1:n$.</p>	$f_7(x) = \sum_{i=1}^n -x_i \cdot \sin(\sqrt{ x_i }) \quad -500 \leq x_i \leq 500$ <p>$f_7(x)=\sum_{i=1}^n (-x(i) \cdot \sin(\sqrt{\text{abs}(x(i))}))$, $i=1:n$;</p>	<p>SCHWEFEL's function 7</p> 

8	Griewangk's function 8	global minimum f(x)=0; x(i)=0, i=1:n	$f_8(x) = \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad -600 \leq x_i \leq 600$ f8(x)=sum(x(i)^2/4000)-prod(cos(x(i)/sqrt(i)))+1, i=1:n;	<p>GRIEWANGK's function 8</p> 
9	Sum of different power function 9	global minimum f(x)=0; x(i)=0, i=1:n.	$f_9(x) = \sum_{i=1}^n x_i ^{(i+1)} \quad -1 \leq x_i \leq 1$ f9(x)=sum(abs(x(i))^(i+1)), i=1:n;	<p>Sum of different power function 9</p> 

10	Ackley's Path function 10	global minimum f(x)=0; x(i)=0, i=1:n.	$f_{10}(x) = -a \cdot e^{-b \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}} - e^{-\frac{\sum_{i=1}^n \cos(c \cdot x_i)}{n}} + a + e^1 \quad -1 \leq x_i \leq 1$ <p> f10(x)=-a·exp(-b·sqrt(1/n·sum(x(i)^2)))- exp(1/n·sum(cos(c·x(i))))+a+exp(1); a=20; b=0.2; c=2·pi; i=1:n; </p>	
11	Langermann's function 11	global minimum f(x)=-1.4 (for m=5); x(i)=???, i=1:n.	$f_{11}(x) = -\sum_{i=1}^m c_i \left(e^{-\frac{\ x-A(i)\ ^2}{\pi}} \cdot \cos\left(\pi \cdot \left\ \bar{x} - A(i) \right\ ^2\right) \right) \quad i = 1: m, 2 \leq m \leq 10, 0 \leq x_i \leq 10$ <p> f11(x)=-sum(c(i)·(exp(-1/pi·sum((x-A(i))^2))·cos(pi·sum((x-A(i))^2)))), i=1:m, m=5; A(i),C(i)<>0, m=5 </p>	

12	Michalewicz's function 12	<p>global minimum</p> <p>$f(x)=-4.687$ ($n=5$); $x(i)=???$, $i=1:n$.</p> <p>$f(x)=-9.66$ ($n=10$); $x(i)=???$, $i=1:n$.</p>	$f_{12}(x) = -\sum_{i=1}^n \sin(x_i) \cdot \left(\sin\left(\frac{i \cdot x_i^2}{\pi}\right) \right)^{2m} \quad i = 1:n, m = 10, 0 \leq x_i \leq \pi$ <p>$f_{12}(x)=-\text{sum}(\sin(x(i)) \cdot (\sin(i \cdot x(i)^2/\pi))^{2 \cdot m}),$ $i=1:n, m=10$; проверить для $n=5,10$</p>	
13	Branins's rcos function	<p>global minimum</p> <p>$f(x_1,x_2)=0.397887$; $(x_1,x_2)=(-\pi, 12.275), (pi, 2.275), (9.42478, 2.475)$.</p>	$f_{Bran}(x_1, x_2) = a \cdot (x_2 - b \cdot x_1^2 + c \cdot x_1 - d)^2 + e \cdot (1 - f) \cdot \cos(x_1) + e \quad -5 \leq x_1 \leq 10, 0 \leq x_2 \leq 15$ $a = 1, \quad b = \frac{5.1}{4 \cdot \pi^2}, \quad c = \frac{5}{\pi}, \quad d = 6, \quad e = 10, \quad f = \frac{1}{8 \cdot \pi}$ <p>$f_{Bran}(x_1, x_2) = a \cdot (x_2 - b \cdot x_1^2 + c \cdot x_1 - d)^2 + e \cdot (1 - f) \cdot \cos(x_1) + e$; $a=1, b=5.1/(4 \cdot \pi^2), c=5/\pi, d=6, e=10, f=1/(8 \cdot \pi)$;</p>	

14	Easom's function	global minimum $f(x_1, x_2) = -1$; $(x_1, x_2) = (\pi, \pi)$.	$f_{Easo}(x_1, x_2) = -\cos(x_1) \cdot \cos(x_2) \cdot e^{-((x_1-\pi)^2 + (x_2-\pi)^2)} \quad -100 \leq x_i \leq 100, i = 1:2$ $fEaso(x_1, x_2) = -\cos(x_1) \cdot \cos(x_2) \cdot \exp(-((x_1-\pi)^2 + (x_2-\pi)^2));$	<p>EASOMs function</p> 
15	Goldstein-Price's function	global minimum $f(x_1, x_2) = 3$; $(x_1, x_2) = (0, -1)$.	$f_{Gold}(x_1, x_2) = \left(1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right) \cdot \left(30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right)$ $-2 \leq x_i \leq 2, i = 1:2$ $fGold(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14 \cdot x_1 + 3 \cdot x_1^2 - 14 \cdot x_2 + 6 \cdot x_1 \cdot x_2 + 3 \cdot x_2^2)] \cdot [30 + (2 \cdot x_1 - 3 \cdot x_2)^2 \cdot (18 - 32 \cdot x_1 + 12 \cdot x_1^2 + 48 \cdot x_2 - 36 \cdot x_1 \cdot x_2 + 27 \cdot x_2^2)]$	<p>GOLDSTEIN-PRICE function</p> 

16	Six-hump camel back function	<p>global minimum</p> <p>$f(x_1, x_2) = -1.0316;$</p> <p>$(x_1, x_2) = (-0.0898, 0.7126),$ $(0.0898, -0.7126).$</p>	$f_{Sixh}(x_1, x_2) = (4 - 2.1x_1^2 + x_1^4/3) \cdot x_1^2 + x_1x_2 + (-4 + 4x_2^2) \cdot x_2^2 \quad -3 \leq x_1 \leq 3, -2 \leq x_2 \leq 2$ $fSixh(x_1, x_2) = (4 - 2.1 \cdot x_1^2 + x_1^4/3) \cdot x_1^2 + x_1 \cdot x_2 + (-4 + 4 \cdot x_2^2) \cdot x_2^2;$	
----	------------------------------	---	---	--

Для остальных вариантов берется строчка, соответствующая остатку от деления номера варианта на 16 (для 17-1, 18-2 и т.д.)