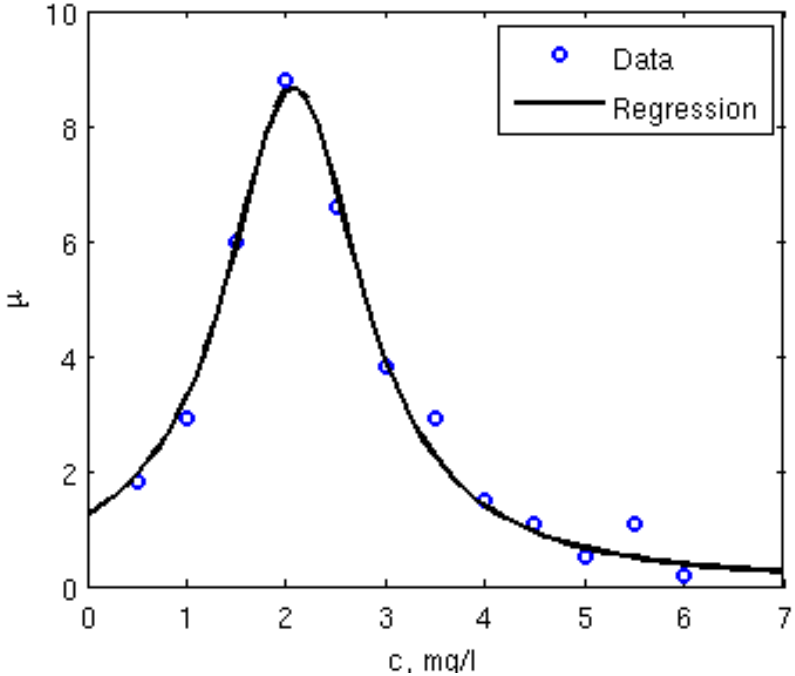
# Лекция 4. Нелинейная регрессия




## Краткое содержание

1. Нелинейная регрессия: постановка задачи
2. Методы Ньютона, Гаусса-Ньютона, Левенберга-Марквардта
3. Функции в языке GNU Octave
4. Практическая реализация нелинейной регрессии в GNU Octave
5. Проблемы глобальной оптимизации и генетические алгоритмы

# Виды регрессии

## 1. Линейная регрессия

$$y = b_0 + b_1 x_1 + \cdots + b_n x_n$$
$$X\beta = y;\ \hat{\beta} = (X^\top X)^{-1} X^\top y$$

## 2. Нелинейная регрессия, поддающаяся линеаризации

$$y = b_0 + b_1 f_1(x_1, \dots, x_n) + \cdots + b_n f_n(x_1, \dots, x_n)$$

<u>Примеры:</u> $k = k_0 \exp\left(-\frac{E_a}{RT}\right) \Rightarrow \ln k = \ln k_0 - \frac{E_a}{RT}$; $c_p(T) = a + bT + cT^2$;

## 3. Нелинейная регрессия, не поддающаяся линеаризации

$$y = f(\beta_1, \dots, \beta_m, x_1, \dots, x_n) = f(\vec{\beta}, \vec{x})$$

<u>Пример:</u> $y = \frac{1}{1+\exp(-\beta_1 - \beta_2 x)}$

Для решения таких задач используют численные методы: Ньютона, Гаусса-Ньютона, градиентного спуска, Левенберга-Марквардта

# Численные методы и МНК

**Нелинейная система уравнений**

$$\begin{cases} \varphi(\beta_1, \dots, \beta_k, x_{11}, x_{21}, \dots x_{m1}) = y_1 \\ \varphi(\beta_1, \dots, \beta_k, x_{12}, x_{22}, \dots, x_{m2}) = y_2 \\ \dots \\ \varphi(\beta_1, \dots, \beta_k, x_{1n}, x_{2n}, \dots, x_{mn}) = y_n \end{cases}$$

**Как правило, полученную систему уравнений нельзя решить аналитическими методами**

Нужно использовать численные методы

**Минимизируемая сумма квадратов**

$$F(\vec{\beta}) = \sum_i (\hat{y}_i - y_i)^2 = \sum_i \left(\varphi(\vec{\beta}, \vec{x}_i) - y_i\right)^2 = \sum_i f_i^2$$

**Система уравнений для поиска минимума (ср. с линейным МНК)**

$$\begin{cases} \dfrac{\partial F(\vec{\beta})}{\partial \beta_1} = 0 \\ \dots \\ \dfrac{\partial F(\vec{\beta})}{\partial \beta_m} = 0 \end{cases}$$

**Рассматриваемые численные методы**

1. Метод Ньютона
2. Метод Гаусса-Ньютона
3. Метод Левенберга-Марквардта

# Метод Ньютона

**Одномерный случай (нелинейное уравнения)**

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) = 0$$

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

**Многомерный случай (система нелинейных уравнений)**

$$F(\vec{\beta}) \approx F(\vec{\beta}_0) + \sum_i \frac{\partial F(\vec{\beta}_0)}{\partial \beta_i}\left(\beta_i - \beta_i^{(0)}\right) = 0 \Rightarrow F(\vec{\beta}_0) + \nabla F(\vec{\beta}_0)\vec{p} = 0$$

**Особенности метода Ньютона**

1. Быстро сходится (квадратичная сходимость)
2. В случае нескольких корней может быть найден любой из них (зависит от начального приближения)
3. Скорость сходимости (и сама сходимость) может зависеть от начального приближения

# Метод Ньютона: практическая реализация

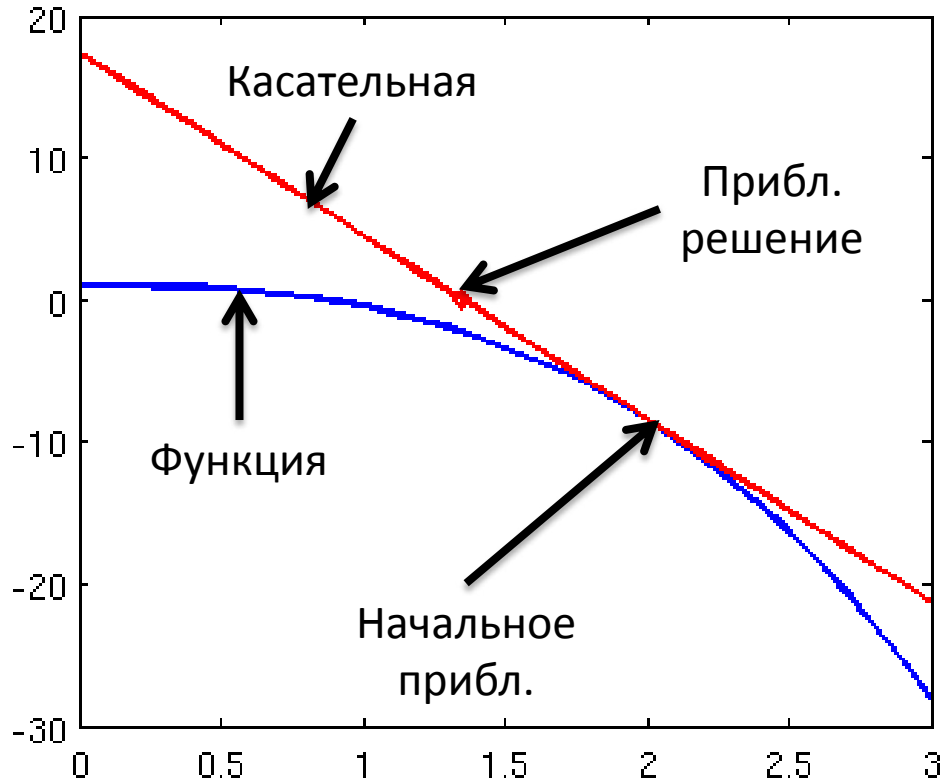
**Уравнение**

$$f(x) = \cos x - x^3 = 0$$

$$f'^{(x)} = -\sin x - 3x^2 = 0$$

**Программа**

```
format long;
delta = 1e-15;
x0 = 0.5;
x = x0 + 2*delta;
while abs(x-x0) >= delta
    x0 = x;
    f = cos(x0)-x0^3;
    df = -sin(x0)-3*x0^2;
    x = x0 - f/df;
    disp(x);
end
```




```
2.000000000000000
1.348055393079852
1.001262924310922
0.880591138294078
0.865691456412747
0.865474078978736
0.865474033101617
0.865474033101614
```

# МНК и метод Гаусса-Ньютона

## Система уравнений и метод Ньютона

$$\begin{cases} \dfrac{\partial F(\vec{\beta})}{\partial \beta_1} = 0 \\ \cdots \\ \dfrac{\partial F(\vec{\beta})}{\partial \beta_m} = 0 \end{cases} \Rightarrow \frac{\partial F(\vec{\beta})}{\partial \beta_i} \approx \frac{\partial F(\vec{\beta}^{(0)})}{\partial \beta_i} + \sum_j \frac{\partial F(\vec{\beta}^{(0)})}{\partial \beta_i \partial \beta_j}\left(\beta_j - \beta_j^{(0)}\right) = 0$$

## Матричная запись и метод Гаусса-Ньютона

$$\nabla F(\vec{\beta}^{(0)}) + H(\vec{\beta}^{(0)})\vec{p} = 0 \Rightarrow J^\top(\vec{\beta}^{(0)}) f(\vec{\beta}^{(0)}) + J^\top(\vec{\beta}^{(0)}) J(\vec{\beta}^{(0)})\vec{p} = 0$$

## Градиент, якобиан и гессиан

$$(\nabla F)_i = \frac{\partial F}{\partial \beta_i} = \frac{\partial}{\partial \beta_i}\left(\sum_j f_j^2\right) = 2\sum_j f_j \frac{\partial f_j}{\partial \beta_i} = 2J^\top f; J_{ij} = \frac{\partial f_i}{\partial \beta_j} = \frac{\partial \varphi(\vec{\beta}, \vec{x_i})}{\partial \beta_j}$$

$$H_{ij} = \frac{\partial F(\beta)}{\partial \beta_i \partial \beta_j} = \frac{\partial}{\partial \beta_j}\left[2\sum_k f_k \frac{\partial f_k}{\partial \beta_i}\right] = 2\sum_k \left[\frac{\partial f_k}{\partial \beta_i}\frac{\partial f_k}{\partial \beta_j} + f_k \frac{\partial f_k}{\partial \beta_i \partial \beta_j}\right] \approx 2\sum_k \frac{\partial f_k}{\partial \beta_i}\frac{\partial f_k}{\partial \beta_j} = 2J^\top J$$

# Метод Левенберга-Марквардта

$$(J^\top J + \lambda I)\vec{p} = -\nabla F = -J^\top f \text{ (вариант 1)}$$

$$(J^\top J + \lambda \mathrm{diag}[J^\top J])\,\vec{p} = -\nabla F = -J^\top f \text{ (вариант 2)}$$

**Каждая итерация включает в себя подбор шага $\lambda$:**

1. Взять начальное (очень малое) значение $\lambda$
2. Выполнить итерацию, найдя $\vec{p}$
3. Если значение $F$ возросло, то увеличить шаг $\lambda$ и вернуться к п.1
4. Если значение $F$ уменьшилось, то принять полученное значение $\vec{p}$ и перейти к следующей итерации

*Сходится медленнее метода Гаусса-Ньютона, но менее требователен к начальному приближению. Широко применяется на практике.*

**Метод Л.-М. – комбинация методов Гаусса-Ньютона и градиентного спуска**

а) метод градиентного спуска: $\vec{p} = -\lambda \nabla F\left(\vec{\beta}^{(0)}\right)$

б) метод Гаусса-Ньютона: $J^\top J\vec{p} = -\nabla F$

Обозначения: $\vec{p} = \vec{\beta} - \vec{\beta}^{(0)}$ , $\vec{\beta}^{(0)}$ – начальное приближение; $\vec{\beta}$ – результат итерации,$\lambda$ – шаг, $F$ – минимизируемая функция

# О функциях в GNU Octave

Функция – фрагмент программного кода, который можно вызывать из других частей программы. Имеет входные и выходные параметры (аргументы), а также свою область видимости для переменных.

## Именованные функции

Хранятся в файлах. Синтаксис:

```
% Справочная информация
function [o1,…,om]=funcname(i1,…,in)
% Тело функции
end
```

Пример:

```
% SQREQ_ROOTS  Finds the roots
% of square equation
function [x1,x2] = sqreq_roots(a,b,c)
D = b.^2 - 4*a.*c;
x1 = (-b + sqrt(D)) ./ (2*a);
x2 = (-b - sqrt(D)) ./ (2*a);
end
```

## Анонимные функции

Хранятся в переменных. Синтаксис:

```
func = @(arg1,…,argn) expression
```

Примеры:

```
sqr = @(x) x.^2
len = @(x,y) sqrt(x.^2 + y.^2)
```

См. также: вложенные функции (nested function), MEX-файлы, объектно-ориентированное программирование

# Нелинейная регрессия: практическая реализация

**Шаг 1. Задание аппроксимирующей функции**

$$y = \beta_1 + \beta_2 \exp(-\beta_3 x)$$

**Шаг 2. Задание якобиана**

$$\begin{cases} \dfrac{\partial y_i}{\partial \beta_1} = 1 \\ \dfrac{\partial y_i}{\partial \beta_2} = \exp(-\beta_3 x_i) \\ \dfrac{\partial y_i}{\partial \beta_3} = -x\beta_2 \exp(-\beta_3 x_i) \end{cases}$$

**Шаг 3. Запись на языке Octave**

```
b0 = [10 10 10];
[b, res, J] = ...
  lsqfit_lm(X,Y,@func, b0);
[db, b_lb, b_ub, sb] = ...
  lsqfit_ci(b, res, J);

function [F, J] = func(b, x)
F = b(1) + b(2)*exp(-b(3)*x);
if nargout == 2
  df_db1 = ones(size(x));
  df_db2 = exp(-b(3)*x);
  df_db3 = -b(2)*exp(-
b(3)*x).*x;
  J = [df_db1 df_db2 df_db3];
end
```

**Шаг 4. Подбор начального приближение и визуализация результатов**

# Нелинейная регрессия: доверительные интервалы

**Линейная регрессия**

$$s^2_{\hat{\boldsymbol{\beta}}} = \hat{\sigma}^2 (X^\top X)^{-1}$$

$$X = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_m^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_m^{(n)} \end{pmatrix}$$

**Нелинейная регрессия**

$$s^2_{\hat{\boldsymbol{\beta}}} = \hat{\sigma}^2 (J^\top J)^{-1}$$

При этом якобиан и отклонения рассчитываются в точке $\hat{\boldsymbol{\beta}}$

**Исходная система уравнений**

$$\begin{cases} \varphi(\beta, x_1) = y_1 \\ \cdots \\ \varphi(\beta, x_n) = y_n \end{cases}$$

**Результат линеаризации в векторной форме**

$$J(\hat{\beta})(\beta - \hat{\beta}) = y - \hat{y} = e$$

**Разложение в ряд Тейлора**

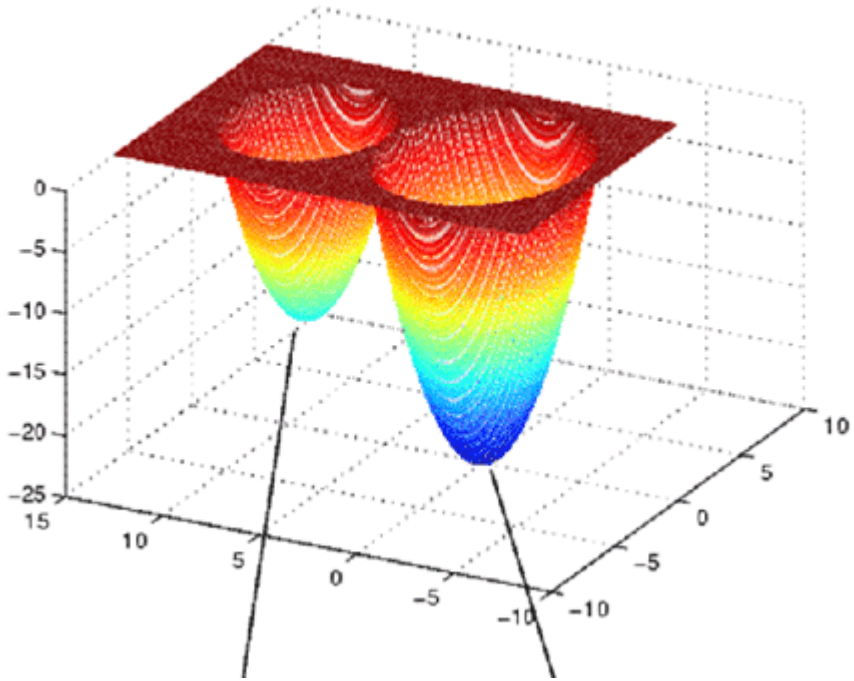
$$\varphi(\beta, x) = \varphi(\hat{\beta}, x) + \sum_i \frac{\partial \varphi(\hat{\beta}, x)}{\partial \hat{\beta}_i} (\beta_i - \hat{\beta}_i)$$

$J(\hat{\beta})$ - аналог матрицы $X$ в линейной регрессии

# Проблемы глобальной оптимизации

**Локальный минимум:** $c$ – локальный минимум функции $y = f(x)$, если $\exists \delta > 0$ такое, что $f(x) - f(c) \geq 0$ для $\forall x \in (c - \delta, c + \delta)$

**Глобальный минимум:** $c$ – глобальный минимум функции $y = f(x)$, если $f(x) - f(c) \geq 0$ для $\forall x \in D(f)$




Локальный минимум Глобальный минимум

**При оптимизации параметров модели важно избегать локальных минимумов и искать глобальный**

**Методы глобальной оптимизации**

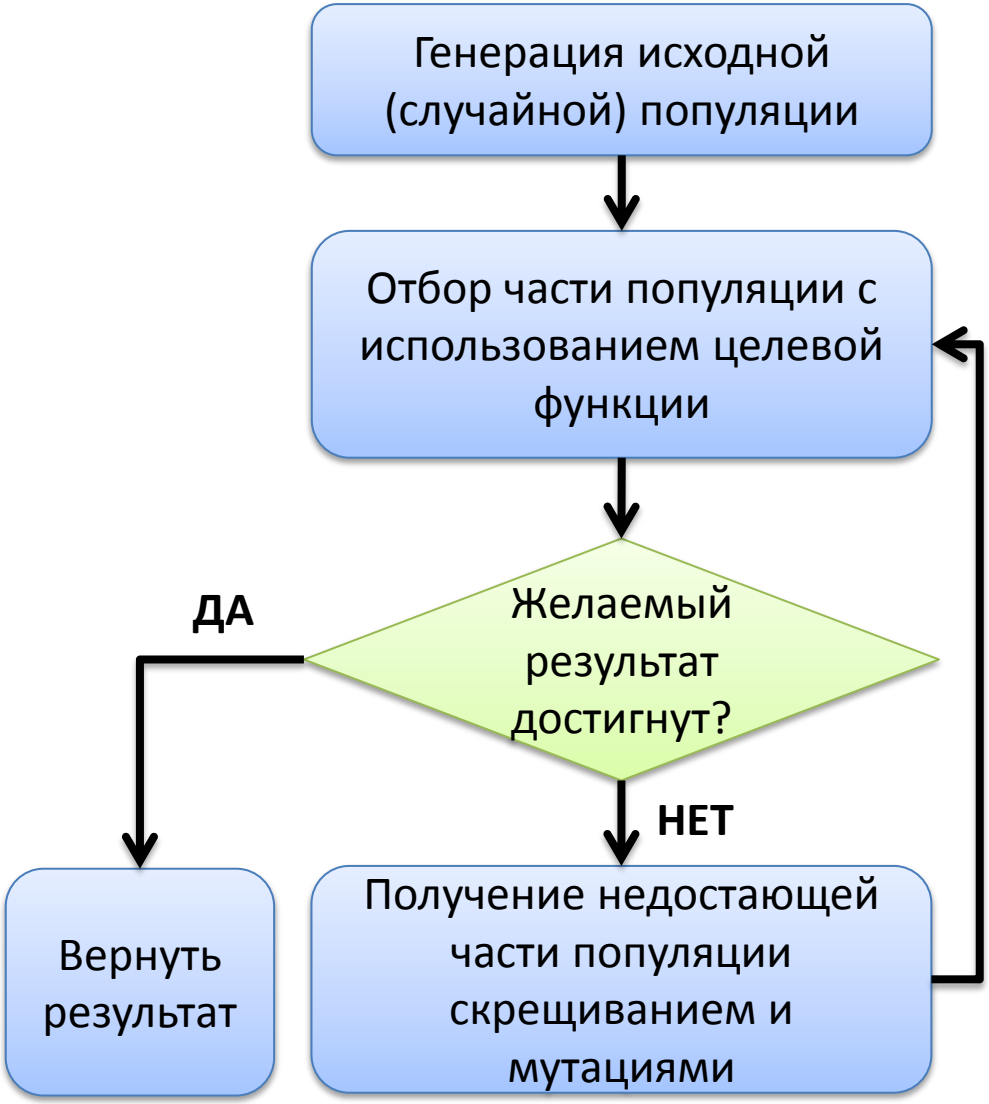
- Генетические алгоритмы
- Метод отжига
- Комбинированные методы
- Символьная регрессия

# Генетические алгоритмы

**Генетические алгоритмы** – алгоритмы нелинейной оптимизации. Ключевые характеристики:

- Работа с большим количеством потенциальных решений («популяцией»)
- Использование целевой функции для отбора членов популяции
- Получение новых членов популяции операциями «скрещивания» (комбинированием) и «мутациями» (случайные изменения)

Ср. с процессом эволюции в живой природе (идея генетического алгоритма была взята именно из биологии)

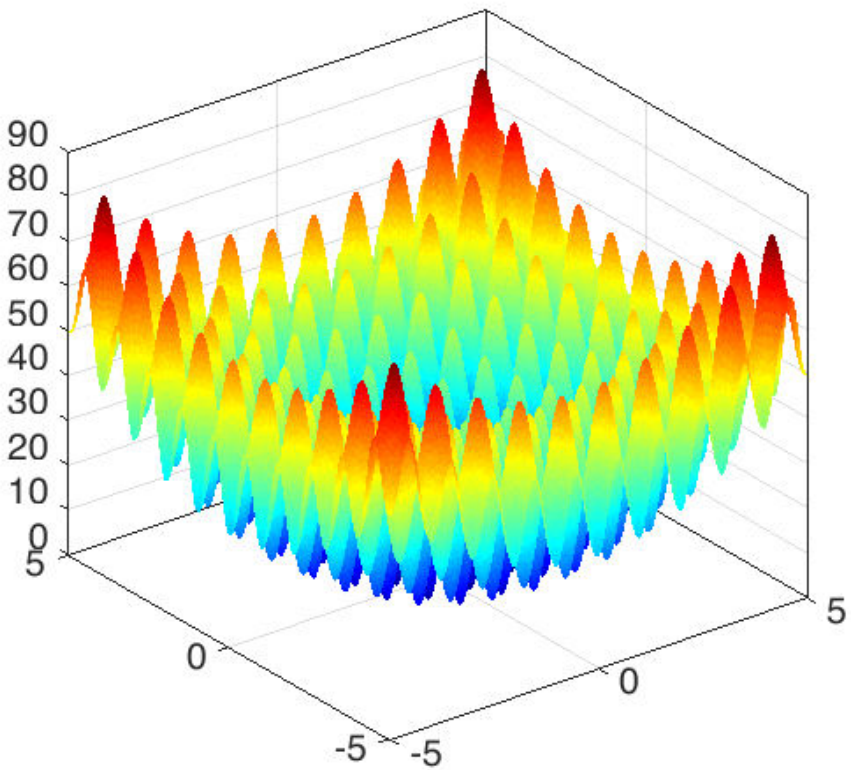


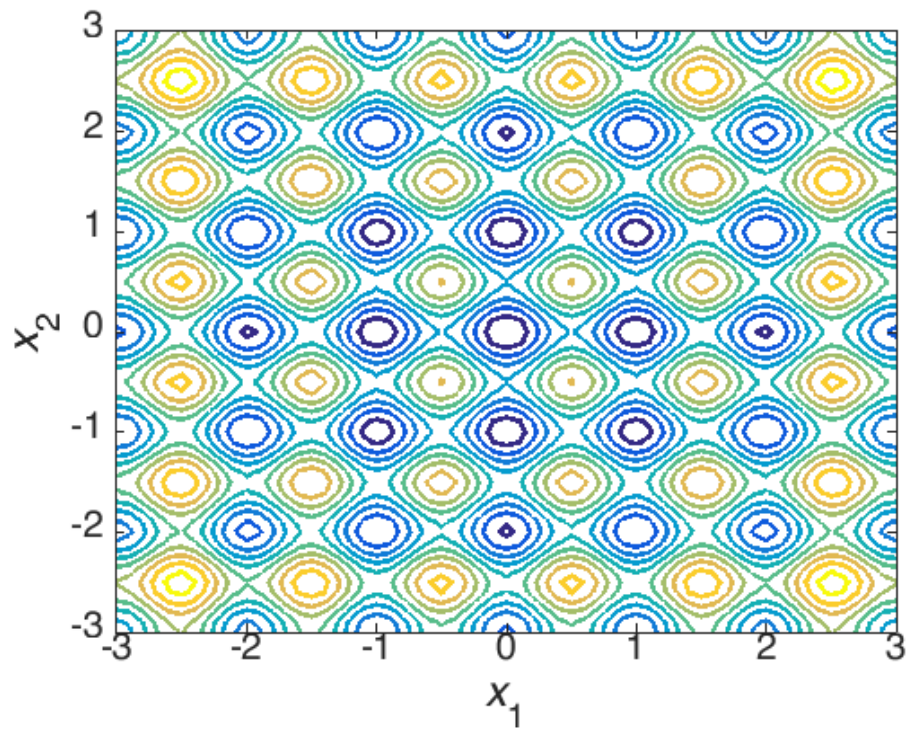
# Пример: функция Растригина

$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10\cos(2\pi x_1) - 10\cos(2\pi x_2)$$

Имеет множество локальных минимумов и один глобальный

Метод Левенберга-Марквардта и ему подобные зачастую находит локальный минимум



# Пример: функция Растригина

**Шаг 1. Создание начальной популяции**

$$x_1^{(0)} = x_2^{(0)} = 10$$
$$x_1^{(i)} = x_1^{(0)} + N_i(0; \sigma^2)$$
$$x_2^{(i)} = x_2^{(0)} + N_i(0; \sigma^2)$$

**Шаг 2. Отбор**

Целевая функция – значение функции Растригина для $x_1^{(i)}$ и $x_2^{(i)}$

**Шаг 3. Скрещивание и мутации**

Ведётся на основе лучших 5-10% членов популяции

Скрещивание: $\begin{cases} x_1^{(i)} = (1 - w_1)x_1^{(m)} + w_1 x_1^{(n)}; w_1 \in [0; 1] \\ x_2^{(i)} = (1 - w_1)x_2^{(m)} + w_2 x_2^{(n)}; w_2 \in [0; 1] \end{cases}$

Мутация: $\begin{cases} x_1^{(i),*} = x_1^{(i)} + N(0; \sigma^2) \\ x_2^{(i),*} = x_2^{(i)} + N(0; \sigma^2) \end{cases}$

# Пример: функция Растригина