

Численные методы (язык Паскаль)

1. Решение уравнений
2. Вычисление площади
(интеграла)
3. Вычисление длины кривой
4. Оптимизация

Численные методы (язык Паскаль)

Тема 1. Решение уравнений

Основные понятия

Задача: решить уравнение

$$x^2 = 5 \cos x \Leftrightarrow x^2 - 5 \cos x = 0$$

$$f(x) = 0$$

Типы решения:

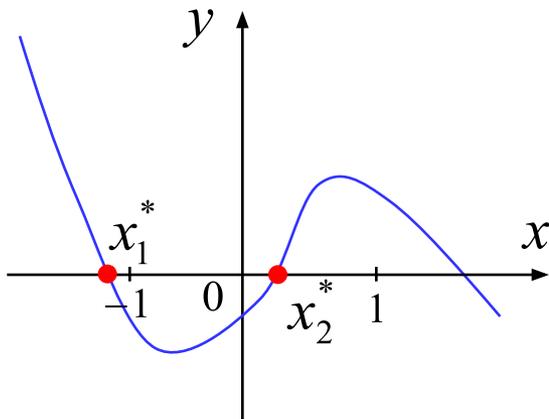
- **аналитическое** (точное, в виде формулы)

$$x^* = \dots$$



- **приближенное** (неточное)

графический метод



численные методы

$x_0 = -1$ начальное приближение

$$x_1 = -1,102$$

$$x_2 = -1,215$$



$$x^* \approx -1,252\dots$$

при $N \rightarrow \infty$

Численные методы

Идея: последовательное уточнение решения с помощью некоторого алгоритма.

Область применения: когда найти точное решение невозможно или крайне сложно.

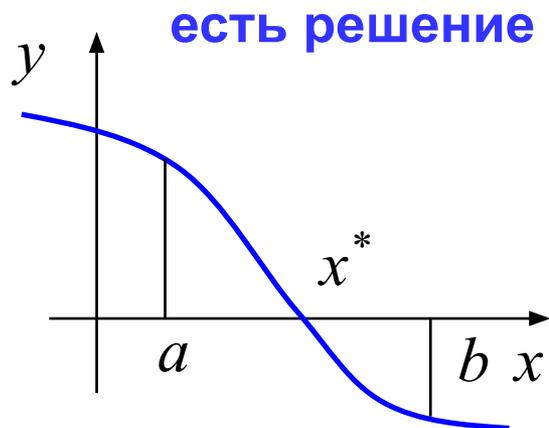
- ⊕ 1) можно найти хоть какое-то решение
- 2) во многих случаях можно оценить ошибку (то есть можно найти решение **с заданной точностью**)

- ⊖ 1) нельзя найти *точное* решение

$$\sqrt{x+1} - 4\sin(x-1) = 0 \quad \longrightarrow \quad x = \del{1,2974} \quad x \approx 1,3974$$

- 2) невозможно исследовать решение при изменении параметров
- 3) большой объем вычислений
- 4) иногда сложно оценить ошибку
- 5) нет универсальных методов

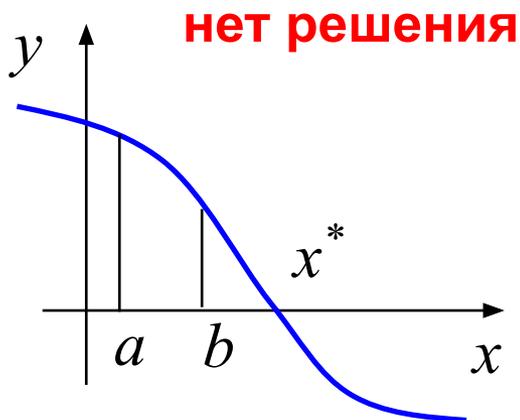
Есть ли решение на $[a, b]$?



$$f(a) > 0$$

$$f(b) < 0$$

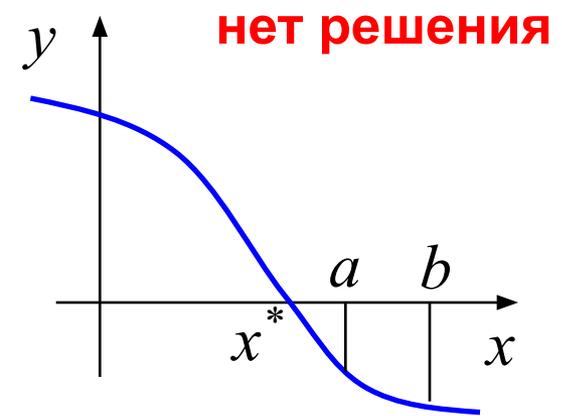
$$f(a)f(b) < 0$$



$$f(a) > 0$$

$$f(b) > 0$$

$$f(a)f(b) > 0$$



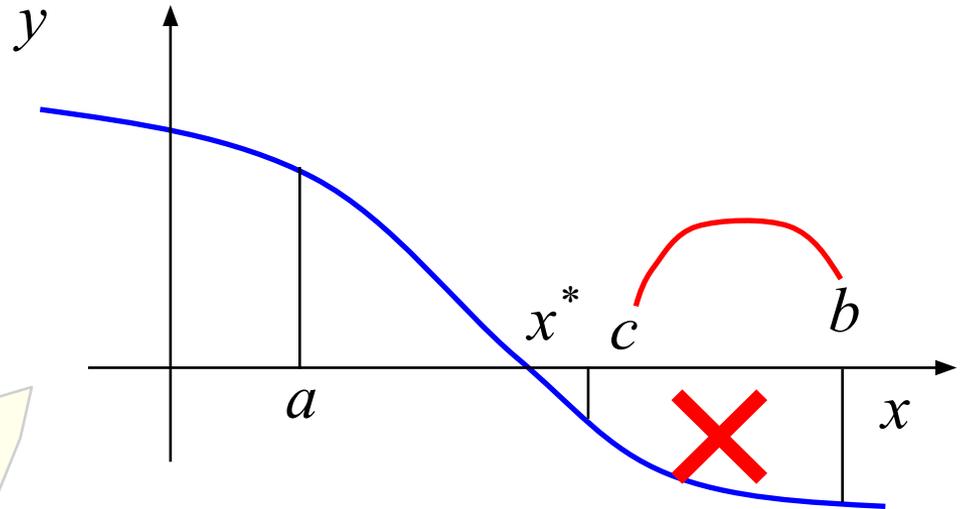
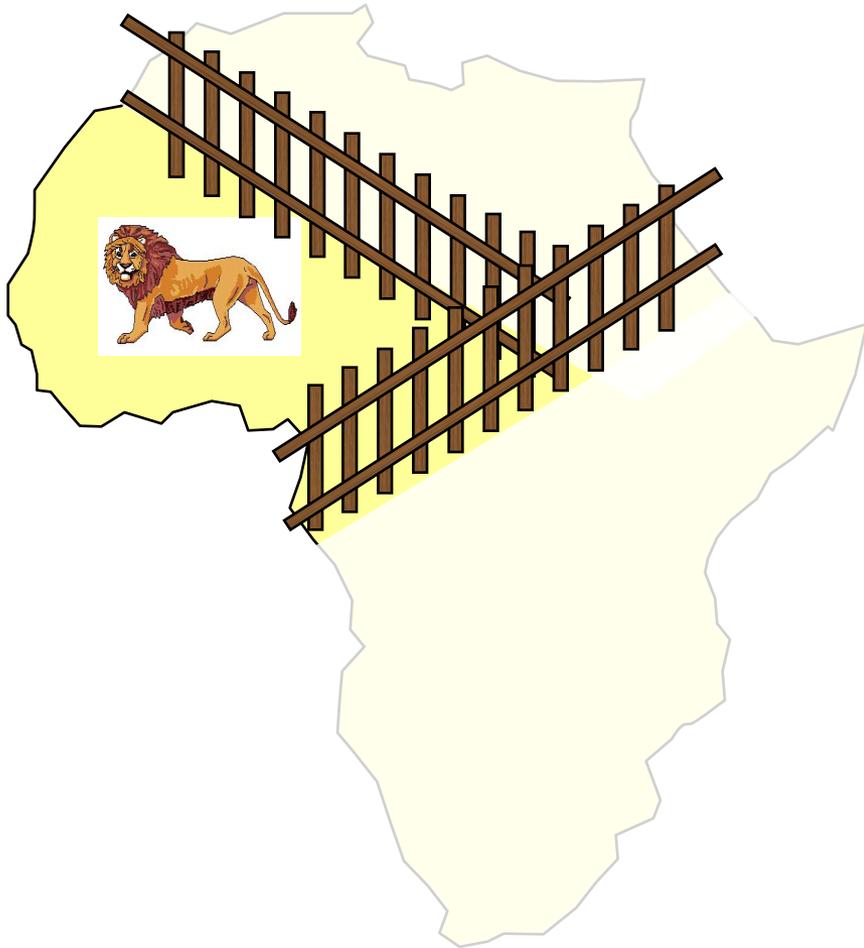
$$f(a) < 0$$

$$f(b) < 0$$



Если **непрерывная** функция $f(x)$ имеет разные знаки на концах интервала $[a, b]$, то в некоторой точке x^* внутри $[a, b]$ имеем $f(x^*) = 0$!

Метод дихотомии (деление пополам)



1. Найти середину отрезка $[a, b]$:
$$c = (a + b) / 2;$$
2. Если $f(c) * f(a) < 0$, сдвинуть правую границу интервала
$$b = c;$$
3. Если $f(c) * f(a) \geq 0$, сдвинуть левую границу интервала
$$a = c;$$
4. Повторять шаги 1-3, пока не будет $b - a \leq \epsilon$.

Метод дихотомии (деления пополам)

-  простота
 - можно получить решение с заданной **точностью** (в пределах точности машинных вычислений)
- 
 - нужно знать **интервал** $[a, b]$
 - на интервале $[a, b]$ должно быть только **одно** решение
 - **большое число шагов** для достижения высокой точности
 - только для функций **одной** переменной

Метод деления отрезка пополам

```
{-----
  BinSolve находит решение на [a,b]
           методом деления отрезка пополам
  Вход:   a, b - границы интервала,  a < b
           eps  - точность решения
  Выход:  x - решение уравнения f(x)=0
-----}
```

```
function BinSolve (a, b, eps: real): real;
var c:real;
begin
  while b - a > eps do begin
    c := (a + b) / 2;
    if f(a)*f(c) < 0 then
      b := c
    else a := c;
  end;
  BinSolve := (a + b) / 2;
end;
```

```
function f(x:real): real;
begin
  f := x*x - 5;
end;
```

Как подсчитать число шагов?

```

function BinSolve (a, b, eps: real;
                  var N: integer ): real;
var c:real;
begin
  N := 0;
  while b - a > eps do begin
    c := (a + b) / 2;
    if f(a)*f(c) < 0 then
      b := c;
    else a := c;
  end;
  BinSolve (a, b, eps, N);
end;

```

значение переменной
меняется внутри функции

Вызов в основной программе:

```

var x: real;
    N: integer;
...
x := BinSolve(1, 2, 0.0001, N);
writeln('Ответ: x = ', x:7:3);
writeln('Число шагов: N = ', N);

```

Метод итераций (повторений)

Задача: $f(x) = 0$ $x = ?$

Эквивалентные преобразования:

$b \cdot f(x) = 0$ имеет те же решения при $b \neq 0$

$$x + b \cdot f(x) = x$$

$$x = \varphi(x), \quad \varphi(x) = x + b \cdot f(x)$$

Идея решения:

x_0 – начальное приближение (например, с графика)

$$x_k = \varphi(x_{k-1}) = x_{k-1} + b \cdot f(x_{k-1}), \quad k = 1, 2, \dots$$

Проблемы:

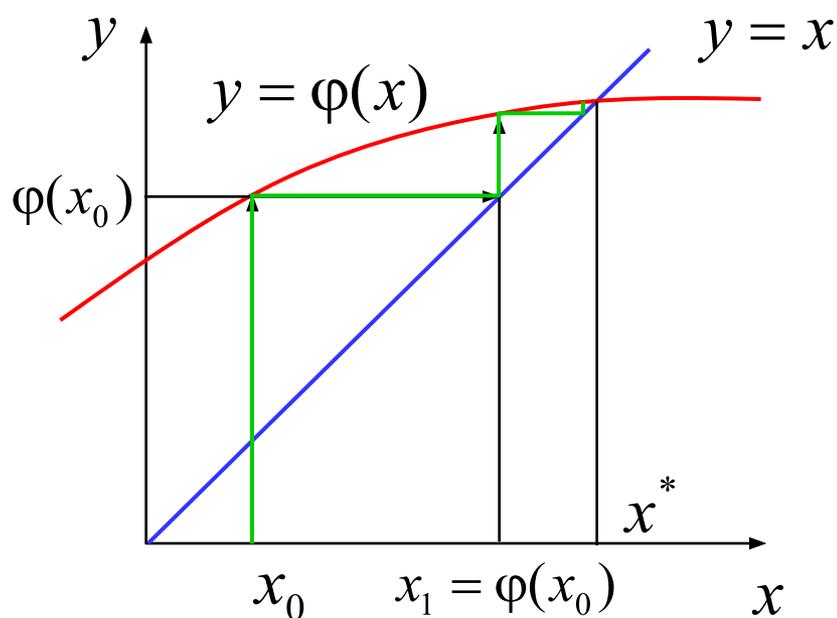
- 1) как лучше выбрать b ?
- 2) всегда ли так можно найти решение?

Сходимость итераций

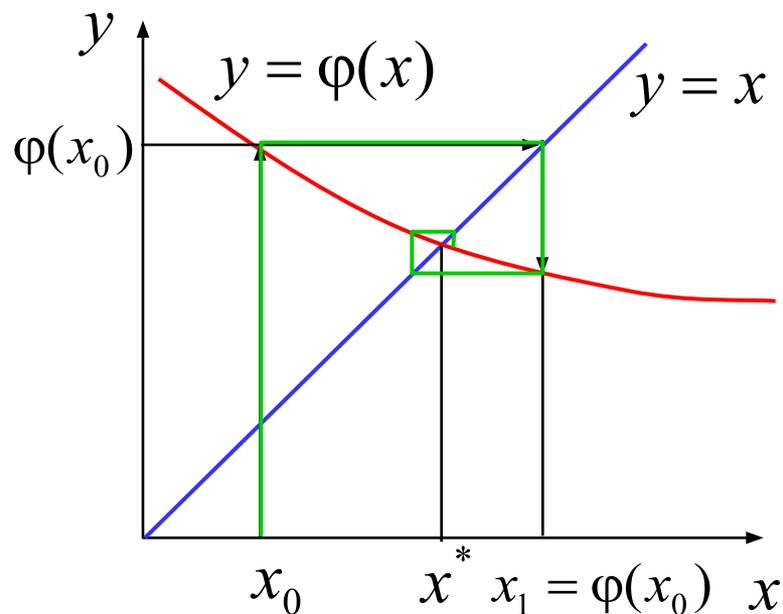
Сходящийся итерационный процесс:

последовательность x_0, x_1, \dots приближается (сходится) к точному решению.

$$x^* = \varphi(x^*) \quad x_0, x_1, x_2, \dots \rightarrow x^*$$



односторонняя сходимость

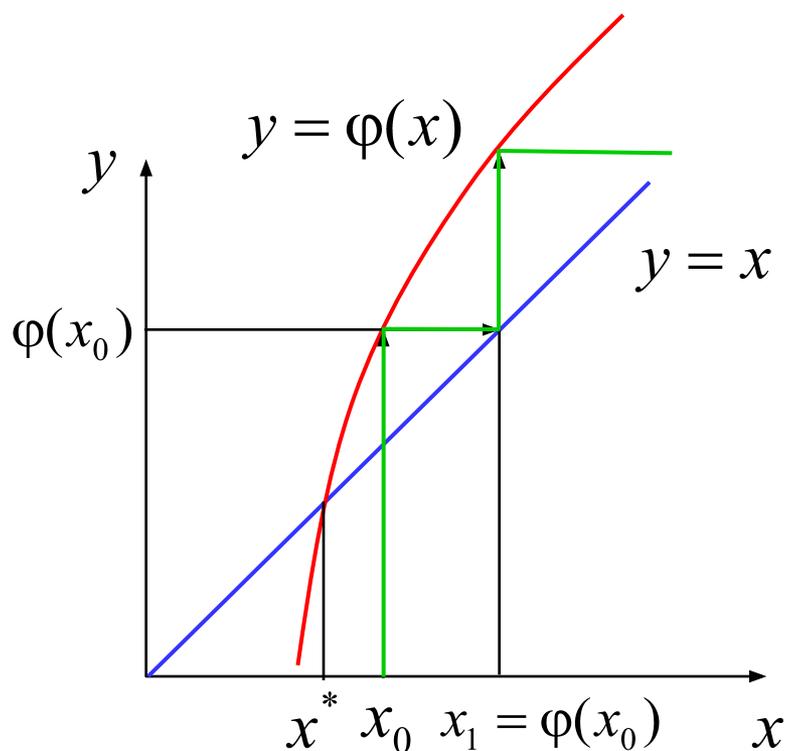


двусторонняя сходимость

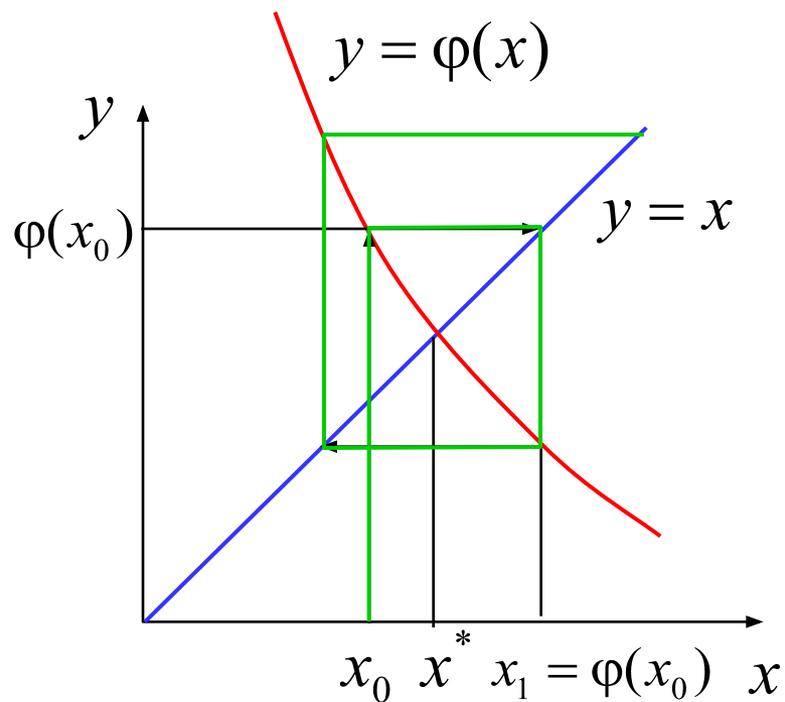
Расходимость итераций

Расходящийся итерационный процесс:

последовательность x_0, x_1, \dots неограниченно возрастает или убывает, не приближается к решению.



односторонняя расходимость

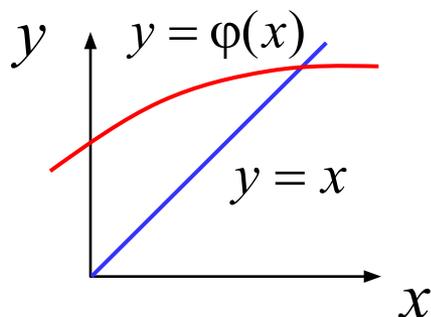


двусторонняя расходимость

От чего зависит сходимость?

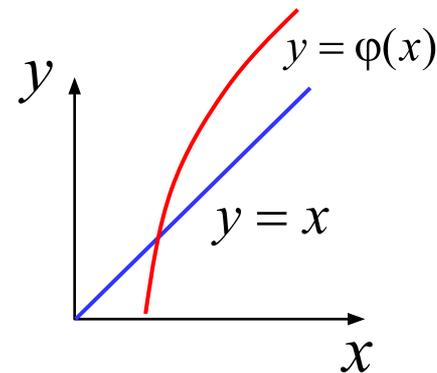
сходится

$$0 < \varphi'(x) < 1$$

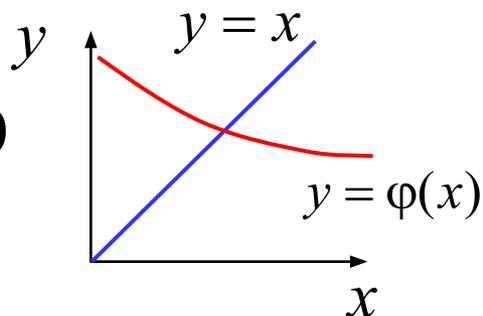


расходится

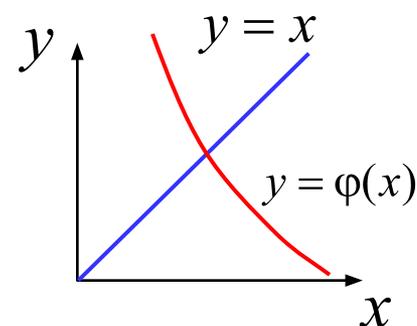
$$\varphi'(x) > 1$$



$$-1 < \varphi'(x) < 0$$



$$\varphi'(x) < -1$$



Выводы:

- сходимость итераций зависит от производной $\varphi'(x)$
- итерации сходятся при $|\varphi'(x)| < 1$ и расходятся при $|\varphi'(x)| > 1$
- сходимость определяется выбором параметра b

$$\varphi(x) = x + b \cdot f(x) \quad \Rightarrow \quad \varphi'(x) = 1 + b \cdot f'(x)$$

Как выбрать b ?

- наугад, попробовать разные варианты
- для начального приближения x_0

$$-1 < 1 + b \cdot f'(x_0) < 1 \quad \Rightarrow \quad -2 < b \cdot f'(x_0) < 0$$

$$f'(x_0) > 0 \quad \Rightarrow \quad -\frac{2}{f'(x_0)} < b < 0$$

$$f'(x_0) < 0 \quad \Rightarrow \quad 0 < b < -\frac{2}{f'(x_0)}$$

- пересчитывать на каждом шаге, например:

$$1 + b \cdot f'(x_k) = 0 \quad \Rightarrow \quad b = -\frac{1}{f'(x_k)}$$



Какие могут быть проблемы?

Метод итераций (программа)

```

{-----
Iter решение уравнения методом итераций
Вход:  x - начальное приближение
        b - параметр
        eps - точность решения
Выход: решение уравнения f(x)=0, n - число шагов
-----}

function Iter (x, b, eps: real; var N: integer): real;
var dx: real;
    OK: boolean;
begin
    N := 0;
    OK := False; {еще не нашли}
    while not OK and (N < 100) do begin
        dx := b*f(x);
        x := x + dx;
        N := N + 1;
        if abs(dx) < eps then OK := True;
    end;
    Iter := x;
end;

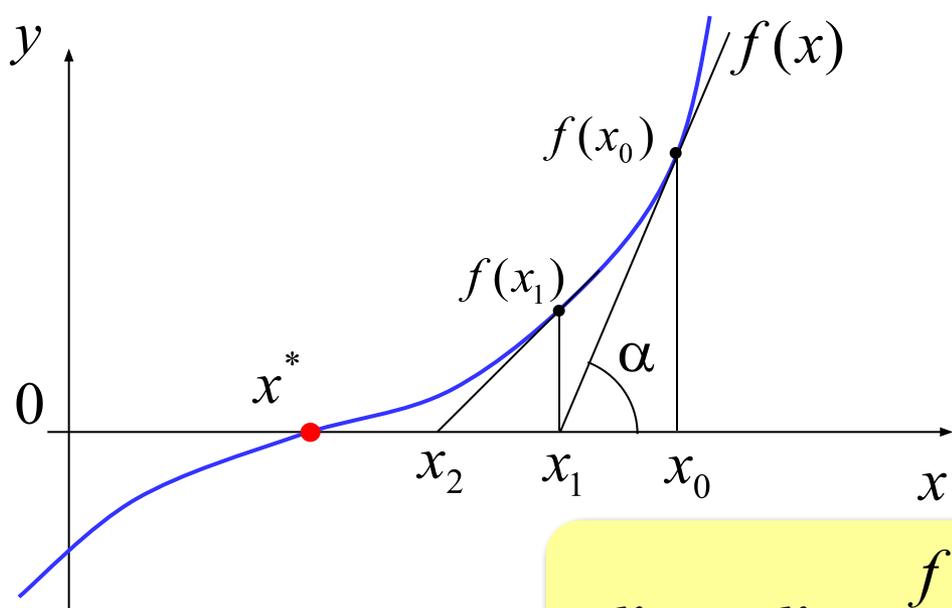
```

аварийный выход
(итерации расходятся)

$$x = x + b \cdot f(x)$$

нормальный
выход

Метод Ньютона (метод касательных)



$$\operatorname{tg} \alpha = \frac{f(x_0)}{x_0 - x_1}$$

$$\operatorname{tg} \alpha = f'(x_0)$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$



Какая связь с методом итераций?

$$x_k = x_{k-1} + b \cdot f(x_{k-1}) \quad \Rightarrow \quad b = -\frac{1}{f'(x_{k-1})}$$

Метод Ньютона (программа)

```

{-----}
Newton решение уравнения методом Ньютона
Вход:  x - начальное приближение
      eps - точность решения
Выход: решение уравнения  $f(x)=0$ , n - число шагов
-----}
function Newton (x, eps: real; var N: integer): real;
var dx: real;
    OK: boolean;
begin
  N := 0; OK := False;
  while not OK and (N < 10)
  begin
    dx := f(x) / df(x);
    x := x - dx;
    N := N + 1;
    OK := abs(dx) < eps;
  end;
  Newton := x;
end;

```

```

{ функция }
function f(x:real): real;
begin
  f := 3*x*x*x+2*x+5;
end;
{ производная }
function df(x:real): real;
begin
  df := 9*x*x + 2;
end;

```

Метод Ньютона



- быстрая (квадратичная) сходимость – ошибка на k -ом шаге обратно пропорциональна k^2
- не нужно знать интервал, только начальное приближение
- применим для функция нескольких переменных



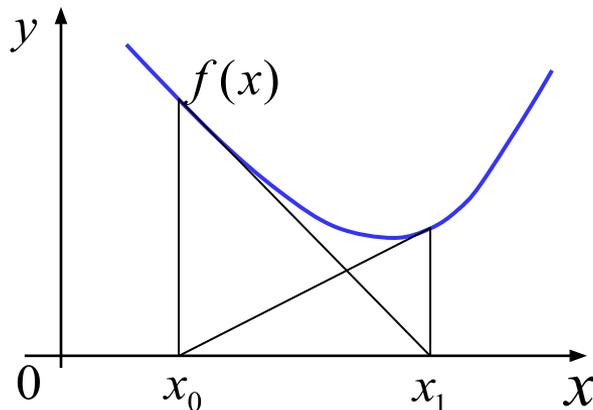
- нужно уметь вычислять производную (по формуле или численно)
- производная не должна быть равна нулю

$$x^3 = 0 \Rightarrow f'(x) = 3x^2$$

- может зацикливаться

$$f(x) = x^3 - 2x + 2$$

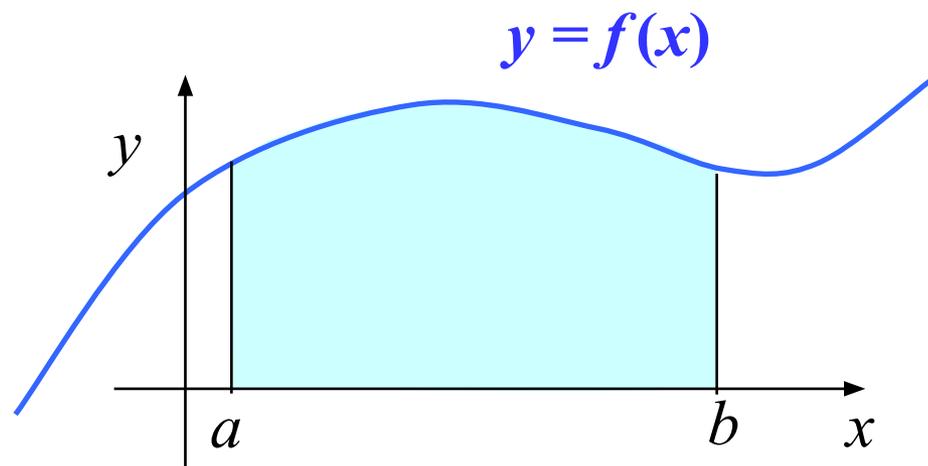
$$x_0 = 0$$



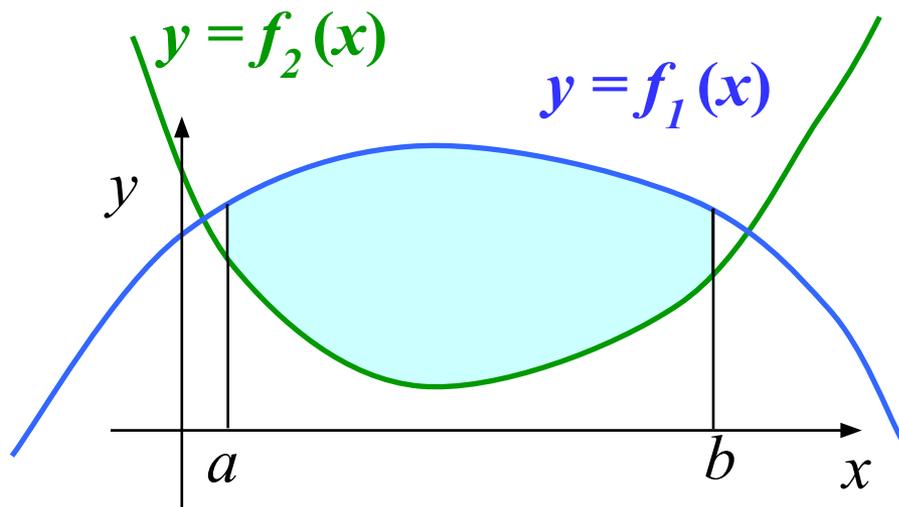
Численные методы (язык Паскаль)

Тема 2. Вычисление площади (интеграла)

Площадь криволинейной трапеции

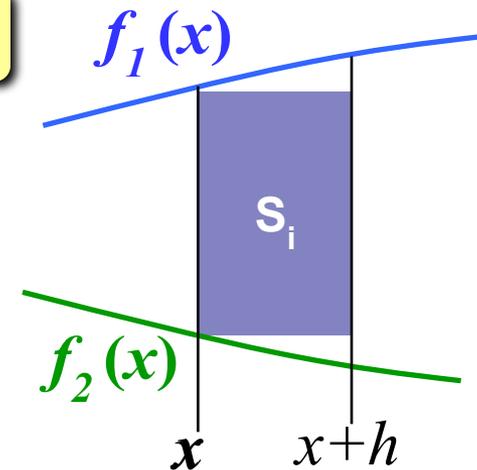
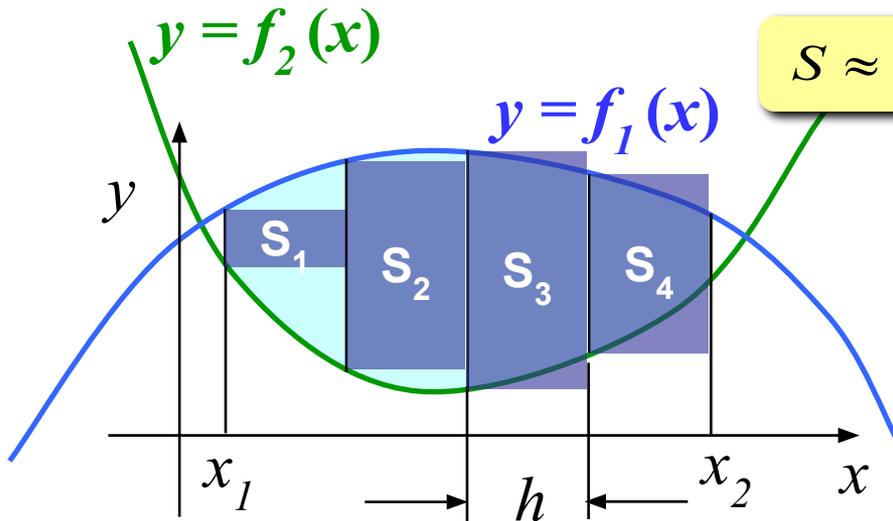


$$S = \int_a^b f(x) dx$$



$$S = \int_a^b f_1(x) dx - \int_a^b f_2(x) dx$$

Метод (левых) прямоугольников



```
function Area(x1, x2:real): real;
var x, S, h: real;
begin
  S := 0; h := 0.001; x := x1;
  while x < x2 do begin
    S := S + f1(x) - f2(x);
    x := x + h;
  end;
  Area := S * h;
end;
```

$$S_i = (f_1(x) - f_2(x)) \cdot h$$

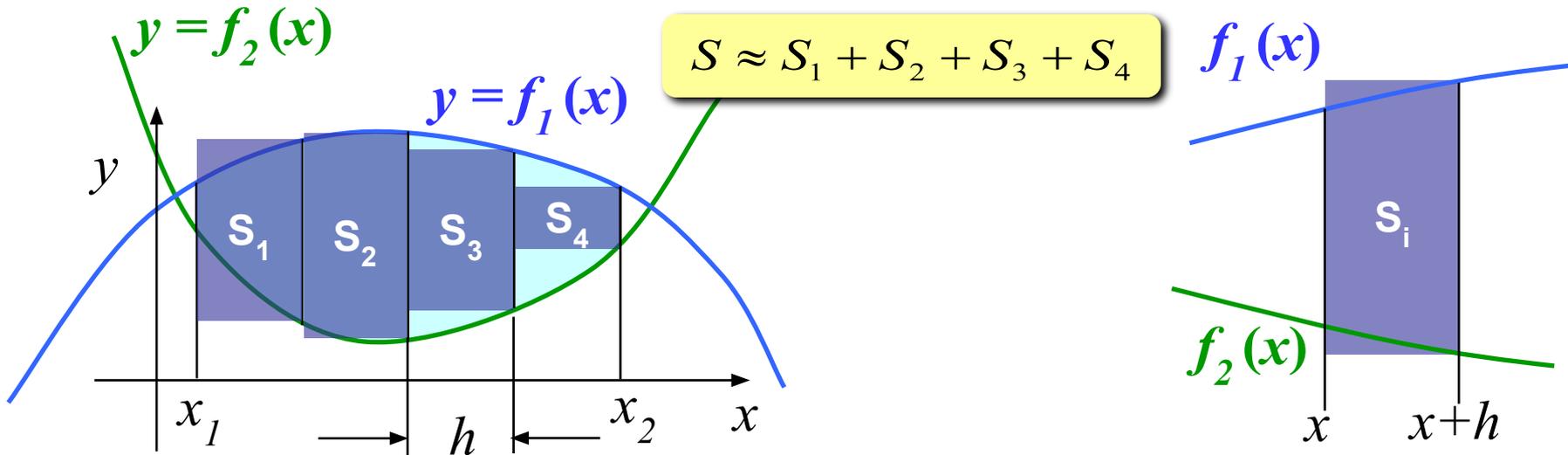


Почему не
 $x \leq x_2$?



Как улучшить
решение?

Метод (правых) прямоугольников



$$S \approx S_1 + S_2 + S_3 + S_4$$

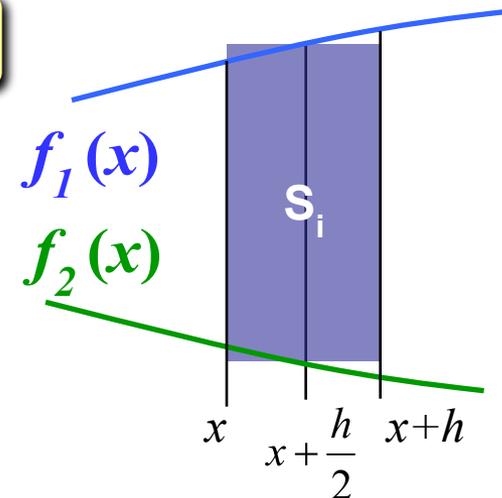
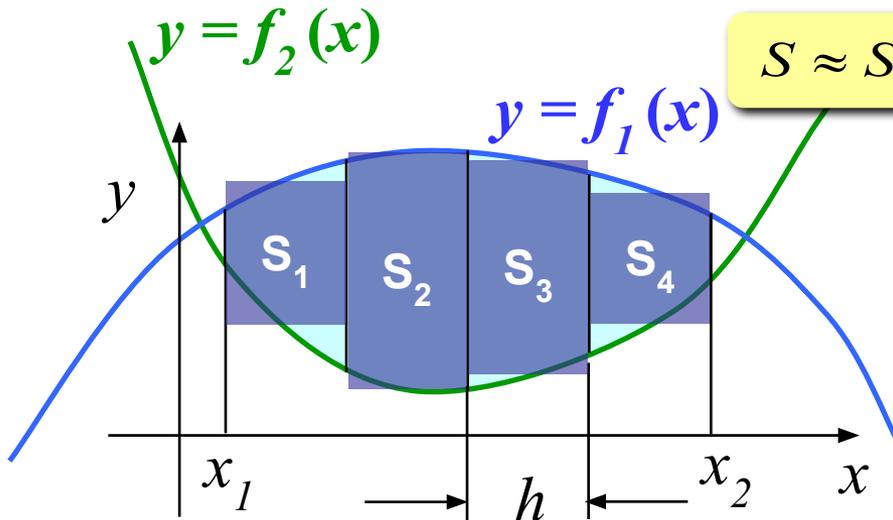
```
function Area(x1, x2:real): real;
var x, S, h: real;
begin
  S := 0; h := 0.001; x := x1;
  while x < x2 do begin
    S := S + f1(x+h) - f2(x+h);
    x := x + h;
  end;
  Area := S * h;
end;
```

$$S_i = (f_1(x+h) - f_2(x+h)) \cdot h$$



Какой метод точнее?

Метод (средних) прямоугольников



```
function Area(x1, x2:real):
var x, S, h: real;
begin
  S := 0; h := 0.001; x := x1;
  while x < x2 do begin
    S := S + f1(x+h/2) - f2(x+h/2);
    x := x+h;
  end;
  Area := S*h;
end;
```

$$S_i = \left[f_1\left(x + \frac{h}{2}\right) - f_2\left(x + \frac{h}{2}\right) \right] \cdot h$$



Какой метод точнее?

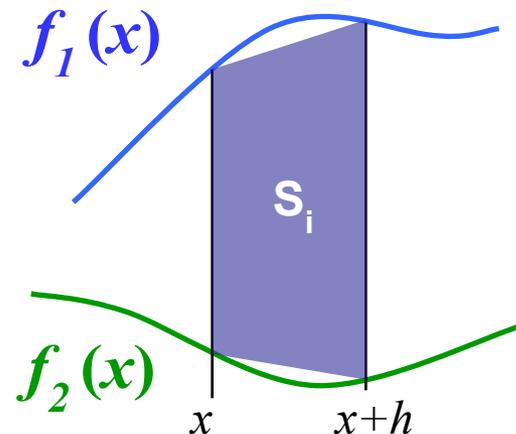
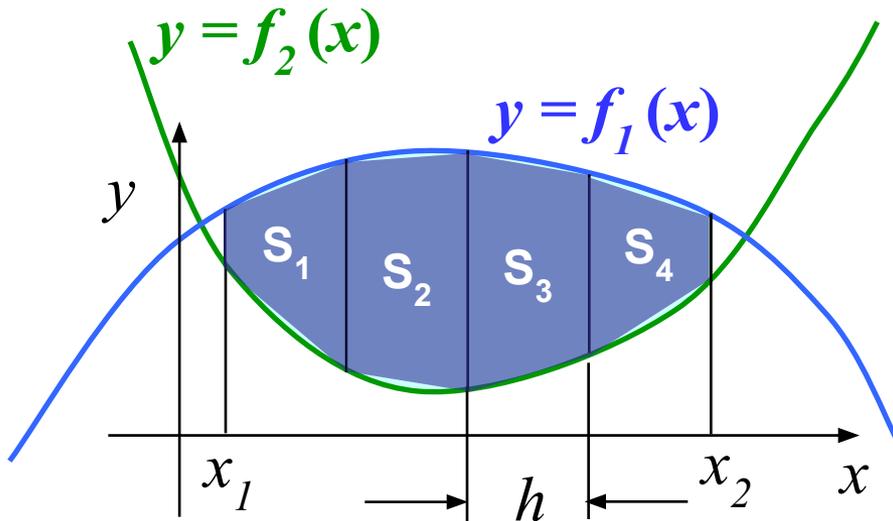
левые (правые):

$$\varepsilon = O(h)$$

средние

$$\varepsilon = O(h^2)$$

Метод трапеций



$$S_i = \frac{f_1(x) - f_2(x) + f_1(x+h) - f_2(x+h)}{2} \cdot h$$

```
x = x1;
```

```
v S := ( f1(x1) - f2(x1) + f1(x2) - f2(x2) ) / 2;
```

```
x := x1 + h;
```

```
while x < x2 do begin
```

```
  S := S + f1(x) - f2(x);
```

```
  x := x + h;
```

```
end;
```

```
S := S*h;
```



Как улучшить?

Ошибка

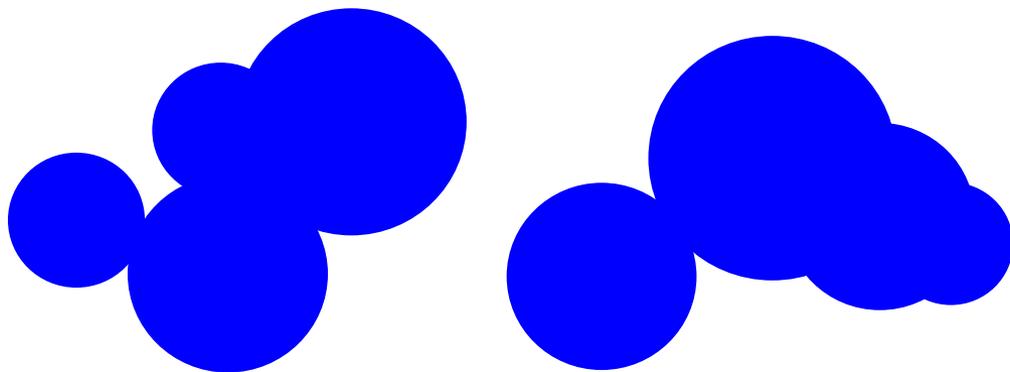
$$\varepsilon = O(h^2)$$

Метод Монте-Карло

Применение: вычисление площадей сложных фигур (трудно применить другие методы).

Требования: необходимо уметь достаточно просто определять, попала ли точка (x, y) внутрь фигуры.

Пример: заданы 100 кругов (координаты центра, радиусы), которые могут пересекаться. Найти площадь области, перекрытой кругами.

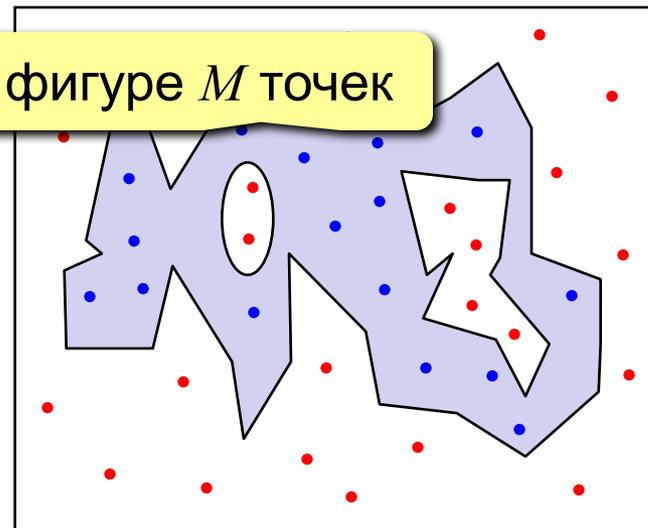


Как найти s ?

Метод Монте-Карло

1. Вписываем сложную фигуру в другую фигуру, для которой легко вычислить площадь (прямоугольник, круг, ...).
2. **Равномерно** N точек со случайными координатами внутри прямоугольника.
3. Подсчитываем количество точек, **попавших на фигуру**: M .
4. Вычисляем **площадь**: $\frac{S}{S_0} \approx \frac{M}{N} \Rightarrow S \approx S_0 \cdot \frac{M}{N}$

На фигуре M точек



Всего N точек

$$S \approx S_0 \cdot \frac{M}{N}$$

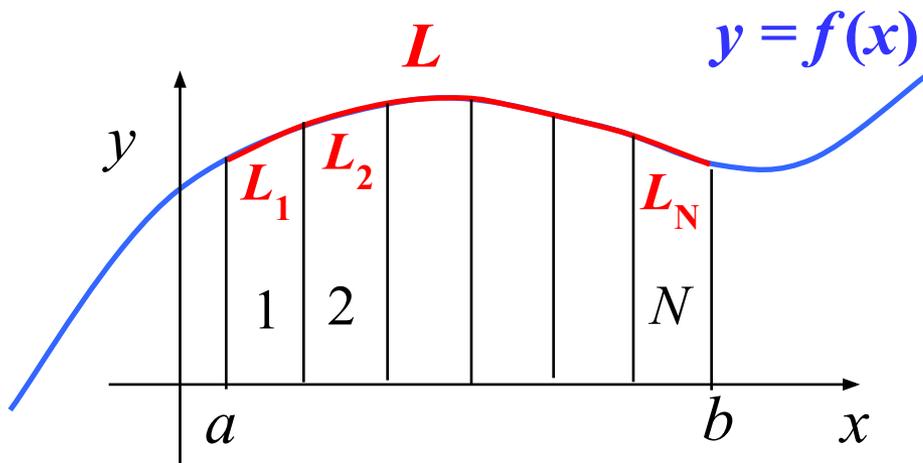


1. Метод приближенный.
2. Распределение должно быть равномерным.
3. Чем больше точек, тем точнее.
4. Точность ограничена датчиком случайных чисел.

Численные методы (язык Паскаль)

Тема 3. Вычисление длины кривой

Длина кривой



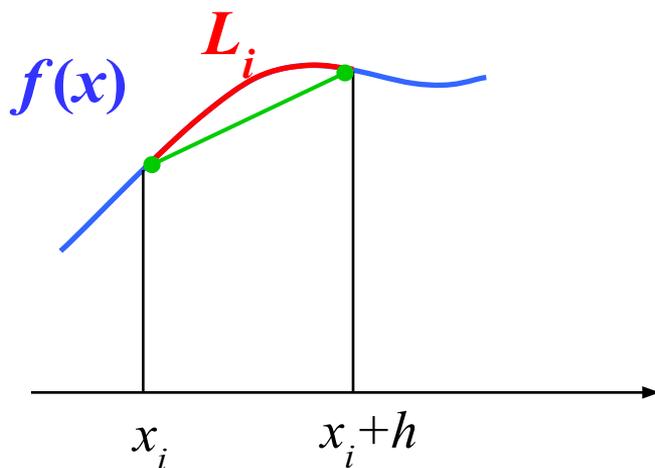
Точное решение:

$$L = \int_a^b \sqrt{1 + [f'(x)]^2} dx$$



- нужна формула для производной
- сложно взять интеграл

Приближенное решение:



$$L = L_1 + L_2 + \dots + L_N = \sum_{i=1}^N L_i$$

$$L_i \approx \sqrt{h^2 + [f(x_i + h) - f(x_i)]^2}$$

Длина кривой

```
{-----  
CurveLen вычисление длины кривой  
Вход:  a, b – границы интервала  
Выход: длина кривой  $y = f(x)$  на интервале [a,b]  
-----}
```

```
function CurveLen(a, b: real): real;  
var x, dy, h, L: real;  
begin  
  h := 0.001; L := 0;  
  x := a;  
  while x < b do begin  
    dy := f(x+h) - f(x);  
    L := L + sqrt(h*h + dy*dy);  
    x := x + h;  
  end;  
  CurveLen := L;  
end;
```

Численные методы

Тема 4. Оптимизация

Основные понятия

Оптимизация – поиск оптимального (наилучшего в некотором смысле) решения.

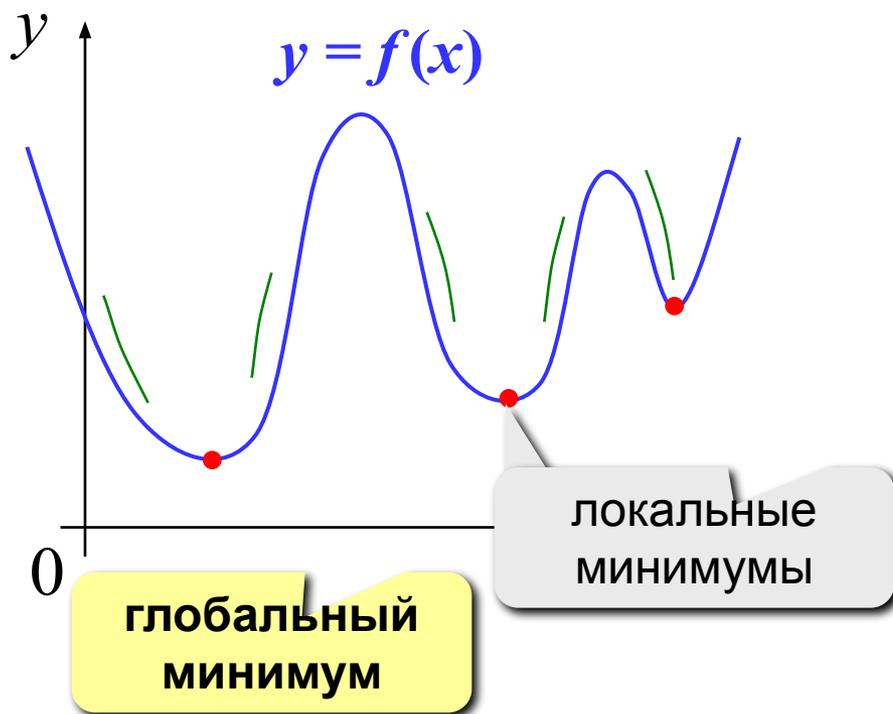
Цель: определить значения неизвестных параметров, при которых заданная функция достигает **минимума** (затраты) или **максимума** (доходы).

$$f(x) \rightarrow \min \quad \text{или} \quad f(x) \rightarrow \max$$

Ограничения – условия, которые делают задачу осмысленной.

Найти x , при котором $f(x) \rightarrow \min$ или $f(x) \rightarrow \max$ при заданных ограничениях.

Локальные и глобальные минимумы



Задача: найти глобальный минимум.

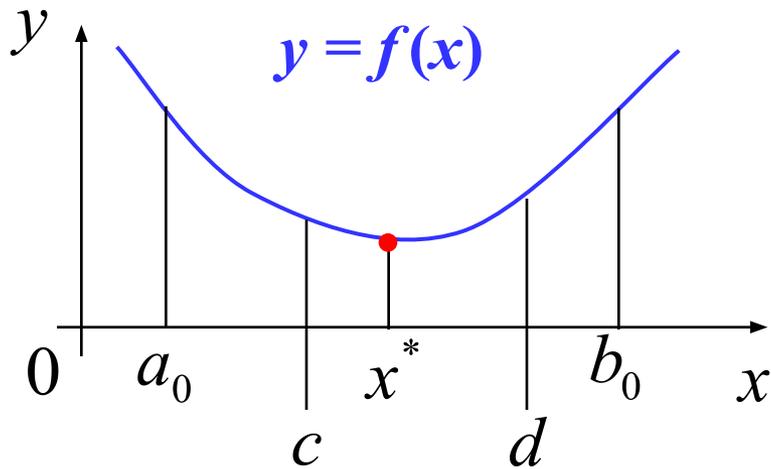
Реальность:

- большинство известных алгоритмов находят только *локальный* минимум вблизи начальной точки
- алгоритмы поиска *глобального* минимума в общем случае неизвестны

Что делать:

- для функций **одной переменной** начальная точка определяется по **графику**
- случайный выбор начальной точки
- запуск алгоритма поиска с нескольких разных точек и выбор наилучшего результата

Минимум функции одной переменной



Дано: на интервале $[a, b]$ функция непрерывна и имеет единственный минимум.

Найти: x^*

Принцип сжатия интервала:

$$[a_0, b_0] \rightarrow [a_1, b_1] \rightarrow \dots \rightarrow [a_n, b_n]$$

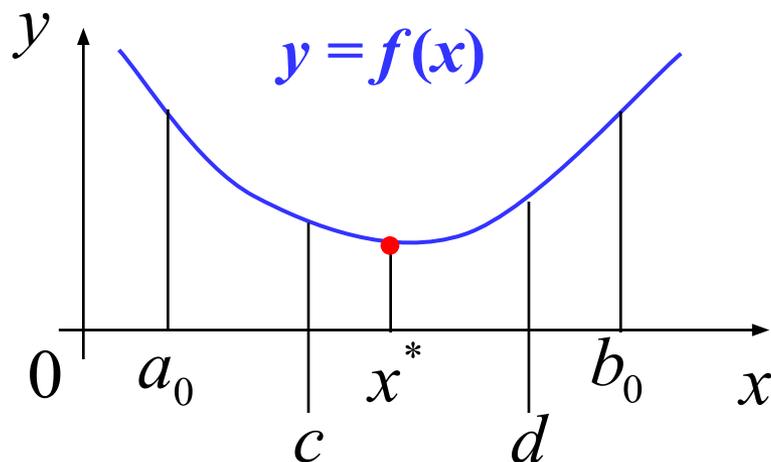
$$c < d \quad f(c) < f(d) \quad \Rightarrow [a_0, d]$$

$$f(c) > f(d) \quad \Rightarrow [c, b_0]$$



Как выбрать c и d наилучшим образом?

Минимум функции одной переменной



Постоянное сжатие в обоих случаях:

$$d - a_0 = b_0 - c$$

Коэффициент сжатия:

$$\mu = \frac{d - a_0}{a_0 - b_0} = \frac{b_0 - c}{a_0 - b_0} \rightarrow \min$$

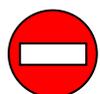
Самое быстрое сжатие:

$$\mu = 0,5 \quad \text{при} \quad c = d = \frac{a_0 + b_0}{2}$$

ДОЛЖНО БЫТЬ $c \neq d$

Метод «почти половинного» деления:

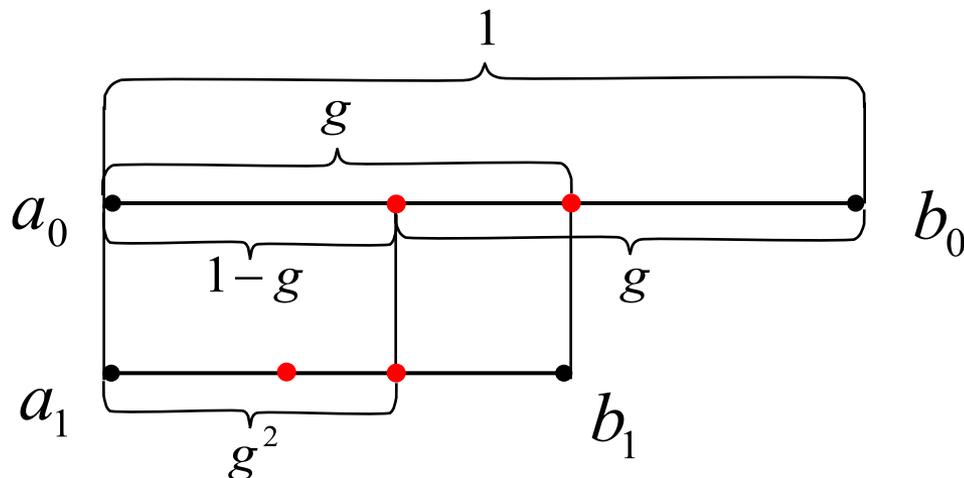
$$c = \frac{a_0 + b_0}{2} - \varepsilon, \quad d = \frac{a_0 + b_0}{2} + \varepsilon \quad \varepsilon - \text{малое число}$$



нужно искать два значения функции на каждом шаге

Отношение «золотого сечения»

Идея: выбрать c и d так, чтобы на каждом шаге вычислять только одно новое значение функции.

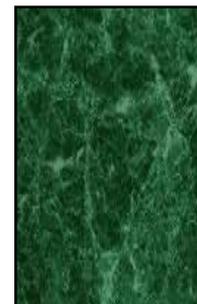


Уравнение для определения g :

$$1 - g = g^2 \Rightarrow g = \frac{1 + \sqrt{5}}{2} \approx 0,618$$

Отношение «золотого сечения»:

$$g \approx 0,618$$



Метод «золотого сечения»

```
{-----
Gold поиск минимума функции («золотое сечение»)
Вход:  a, b – границы интервала, eps – точность
Выход: x, при котором f(x) имеет минимум на [a,b]
-----}
```

```
function Gold(a, b, eps:real): real;
const g = 0.618034;
var x1, x2, R: real;
begin
  R := g*(b - a);
  while abs(b-a) > eps do begin
    x1 := b - R; x2 := a + R;
    if f(x1) > f(x2) then a := x1
    else b := x2;
    R := R * g;
  end;
  Gold := (a + b) / 2;
end;
```



Как вычислять только одно значение на каждом шаге?

Функции нескольких переменных

Найти $\{x_1, x_2, \dots, x_n\}$, для которых $f(x_1, x_2, \dots, x_n) \rightarrow \min$ при заданных ограничениях.

Проблемы:

- нет универсальных алгоритмов поиска глобального минимума
- неясно, как выбрать начальное приближение (зависит от задачи и интуиции)

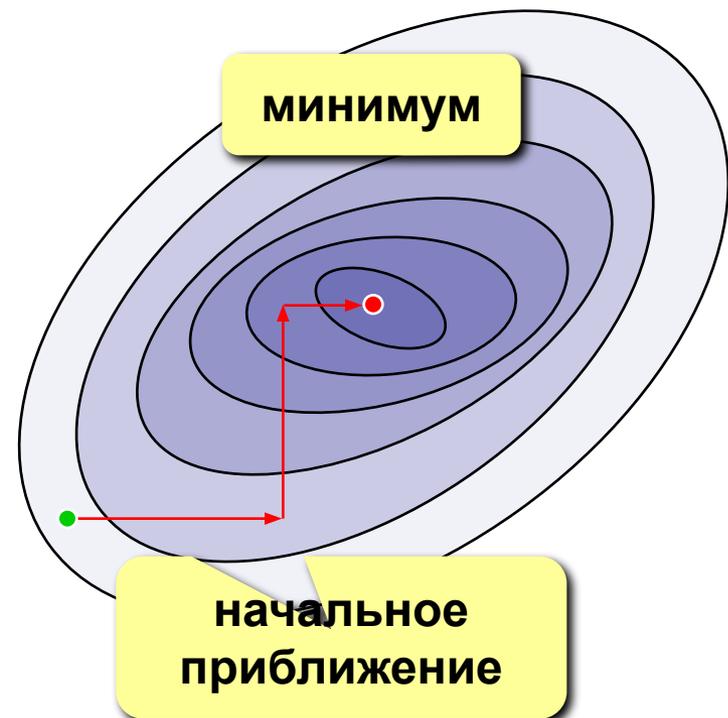
Подходы:

- методы локальной оптимизации (результат зависит от выбора начального приближения)
- случайный поиск (без гарантии)
- методы глобальной оптимизации (для особых классов функций)

Метод покоординатного спуска

Идея:

- выбираем начальную точку
- будем менять только x_1 , а остальные переменные «заморозим», находим минимум по x_1
- теперь будем менять только x_2 , а остальные переменные «заморозим», ...



-  • простота, сводится к нескольким задачам с одной переменной
-  • можно двигаться к минимуму быстрее
- большой объем вычислений
- может не найти решение для сложных функций

Градиентные методы

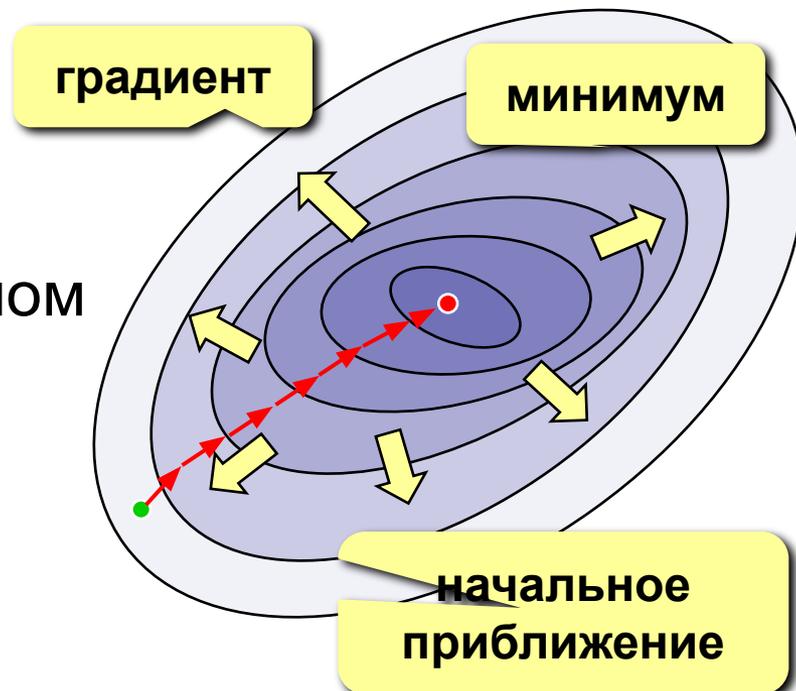
Градиент – это вектор, показывающий направление наискорейшего возрастания функции.

Идея:

- выбираем начальную точку
- на каждом шаге движемся в направлении, противоположном градиенту

 • быстрая сходимость

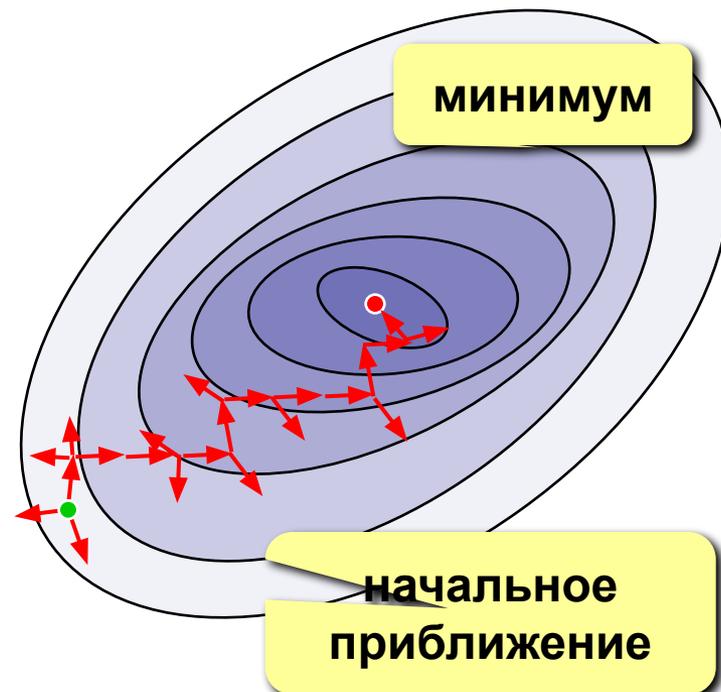
-  • необходимо считать производные (по формуле или численно)
- плохо работает для быстро меняющихся функций



Метод случайного поиска

Идея:

- выбираем начальную точку
- пробуем сделать шаг в случайном направлении
- если значение функции уменьшилось, шаг удачный (запоминается)



- простота реализации
- не требует вычисления производных
- много вариантов с самообучением
- хорошо работает для функций с многими локальными минимумами



- очень большой объем вычислений

Конец фильма
