Численные методы (язык Си)

Основные понятия

Задача: решить уравнение

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

$$f(x) = 0$$

Типы решения:

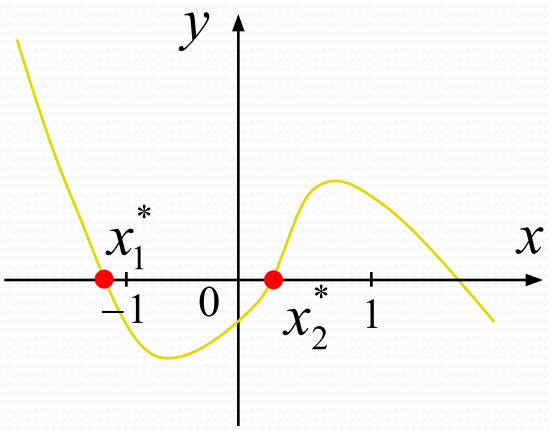
- **аналитическое** (точное, в виде формулы)

$$x^* = \ldots$$

? **Как?**

- **приближенное** (неточное)

графический метод



численные методы

$x_0 = -1$ **начальное приближение**

$x_1 = -1{,}102$

$x_2 = -1{,}215$

при $N \Rightarrow \infty$

$$x^* \approx -1{,}252\ldots$$

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

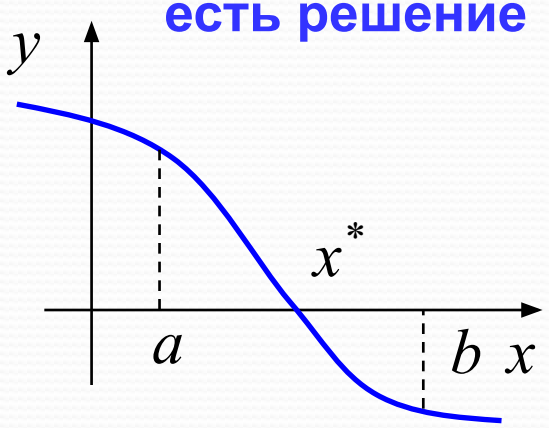
$$\sqrt{x+1}-4\sin(x-1)=0 \Rightarrow x=1,3974$$

$$x \approx 1,3974$$

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

Есть ли решение на $[a, b]$?

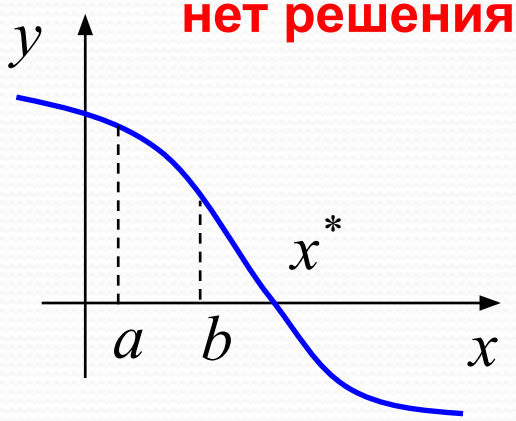
есть решение



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

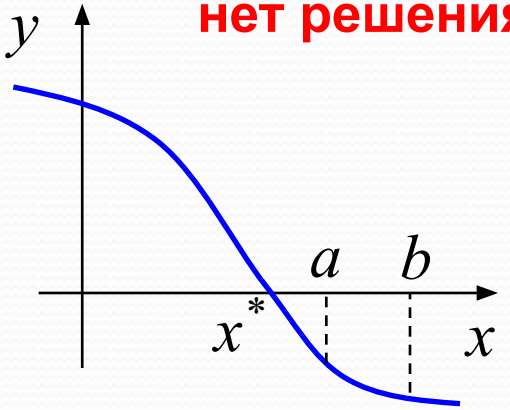
$$f(a)\,f(b) < 0$$

нет решения



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

нет решения

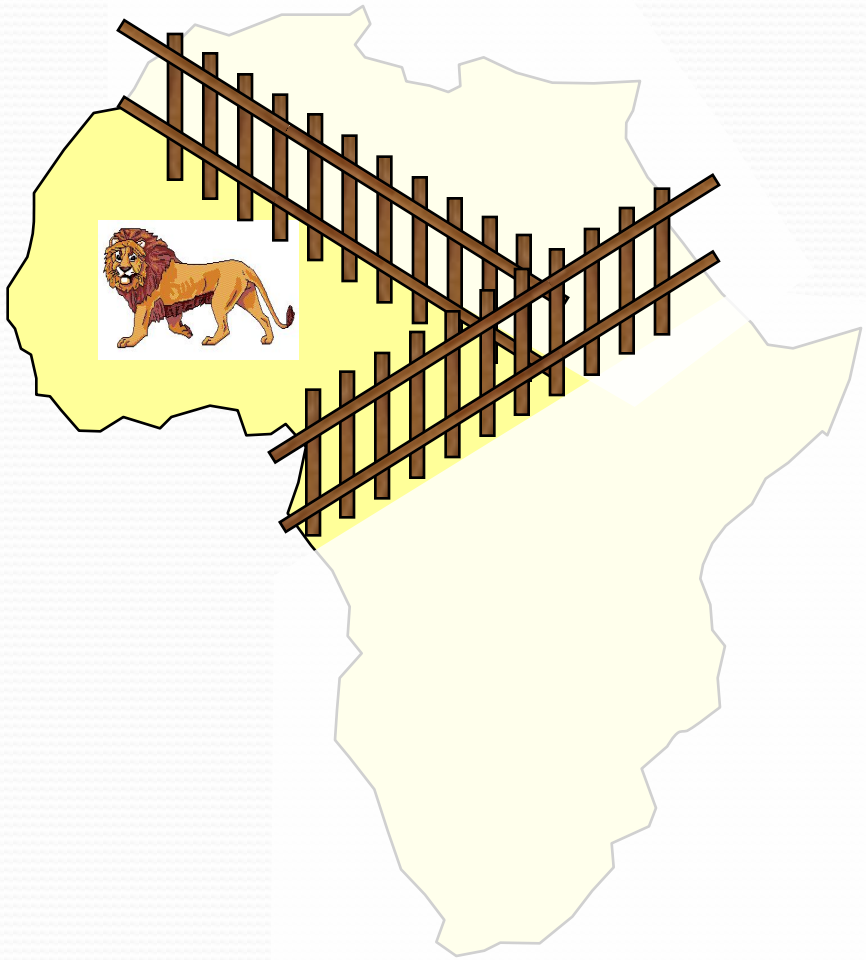


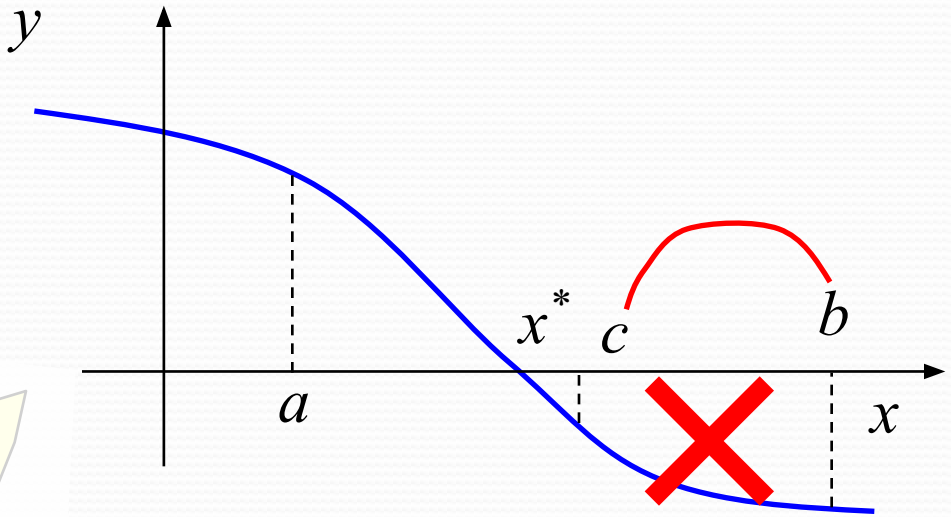
$$f(a) < 0$$
$$f(b) < 0$$

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

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)≥0`, сдвинуть левую границу интервала
   `a = c;`
4. Повторять шаги 1-3, пока не будет `b – a ≤ ε`.

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
//-----------------------------------------
float BinSolve ( float a, float b, float eps )
{
  float c;
  while ( b - a > eps )
    {
    c = (a + b) / 2;
    if ( f(a)*f(c) < 0 )
         b = c;
    else a = c;
    }
  return (a + b) / 2;
}
```

```
float f ( float x )
{
  return x*x – 5;
}
```

Как подсчитать число шагов?

```
float BinSolve ( float a, float b,
                 float eps, int &n )
{
  float c;
  n = 0;
  while ( b - a > eps )
    {
    c = (a + b) / 2;
    if ( f
         b
    else a
    n ++;
    }
  return (
}
```

значение переменной меняется внутри функции

Вызов в основной программе:

```
float x;
int N;
...
x = BinSolve ( 2, 3, 0.0001, N );
printf("Ответ: x = %7.3f", x);
printf("Число шагов: %d", N);
```

Метод итераций (повторений)

Задача: $f(x) = 0 \qquad x = ?$

Эквивалентные преобразования:

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

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

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

Идея решения:

x_0 – начальное приближение (например, с графика)

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

Проблемы:

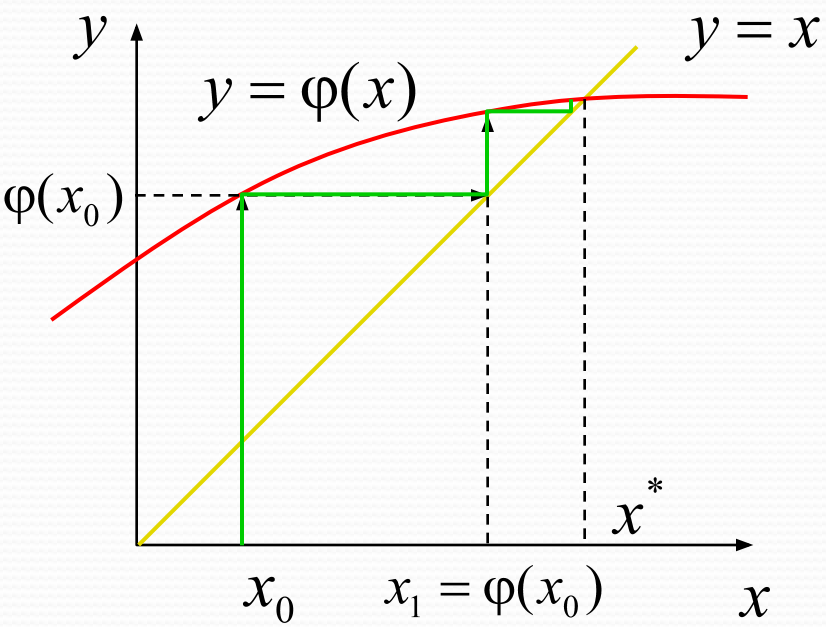
1) как лучше выбрать b?
2) всегда ли так можно найти решение?

Сходимость итераций

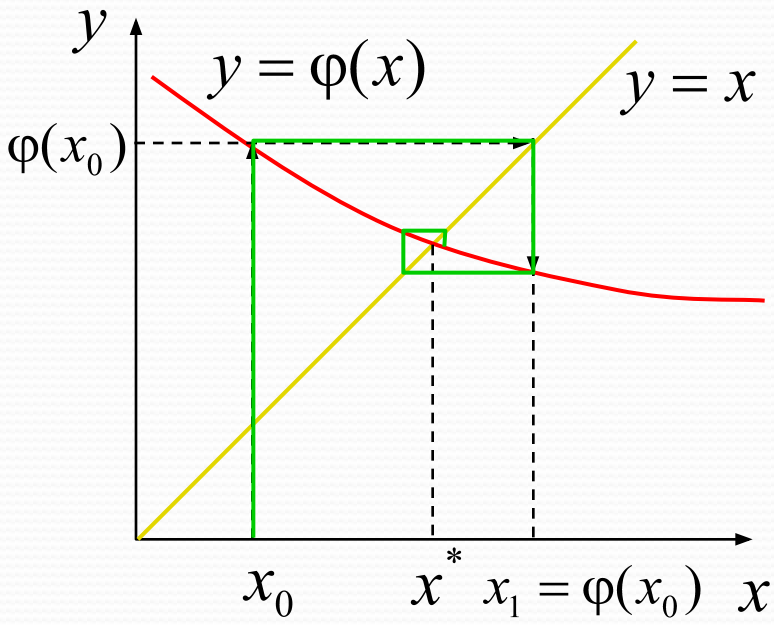
Сходящийся итерационный процесс:

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

$$x^* = \varphi(x^*) \qquad x_0, x_1, x_2, \ldots \to x^*$$



односторонняя сходимость

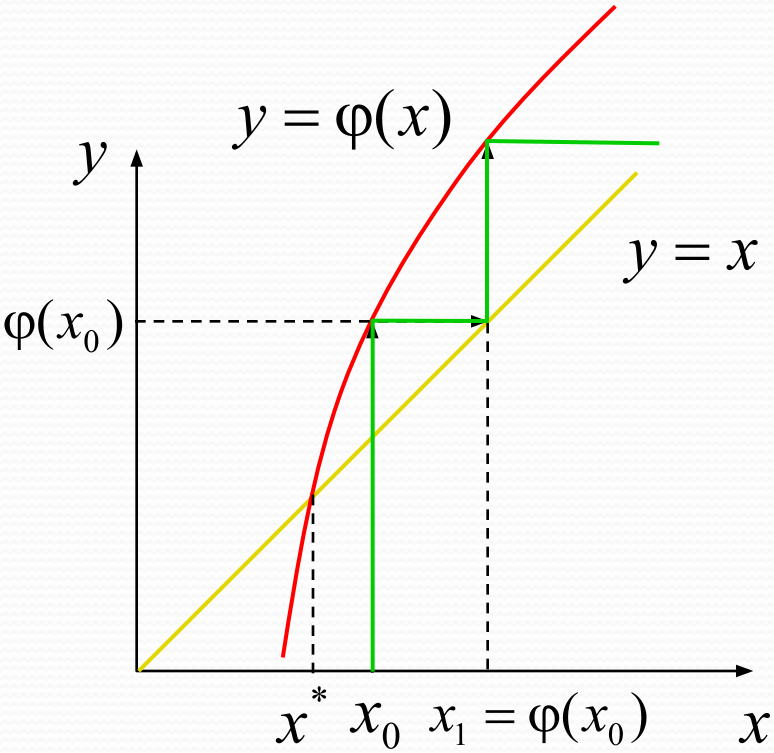


двусторонняя сходимость

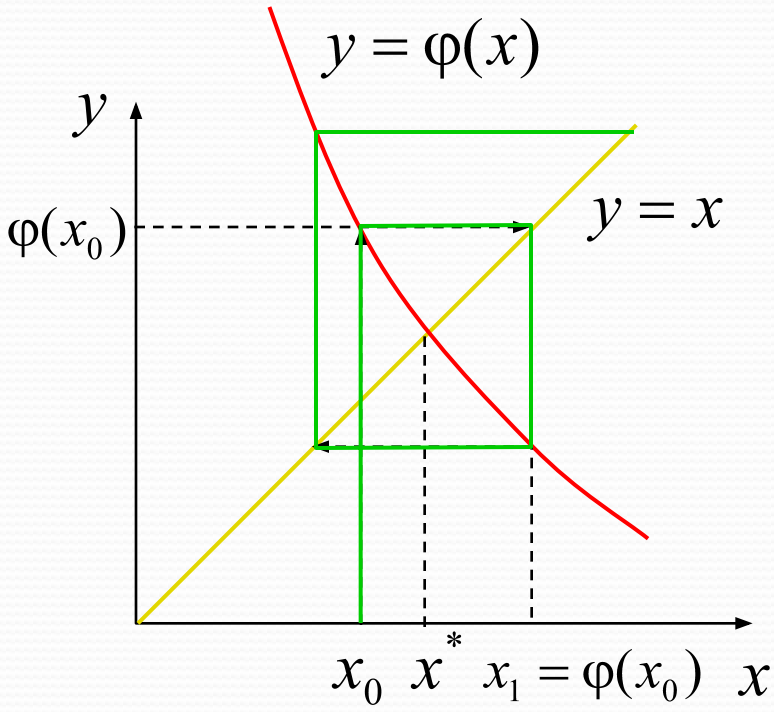
Расходимость итераций

Расходящийся итерационный процесс:

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



односторонняя расходимость

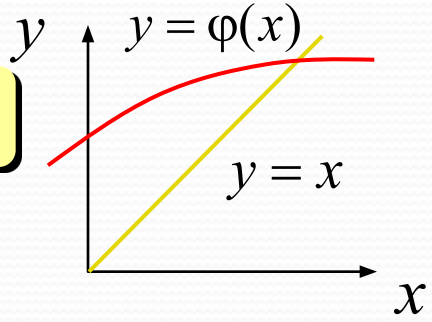


двусторонняя расходимость

От чего зависит сходимость?

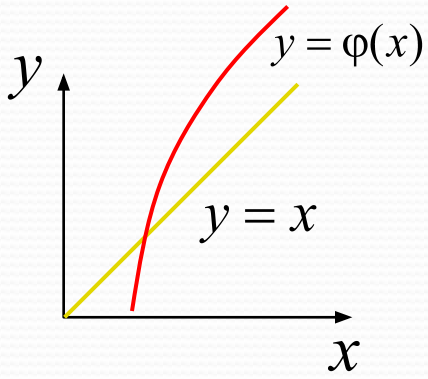
сходится

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

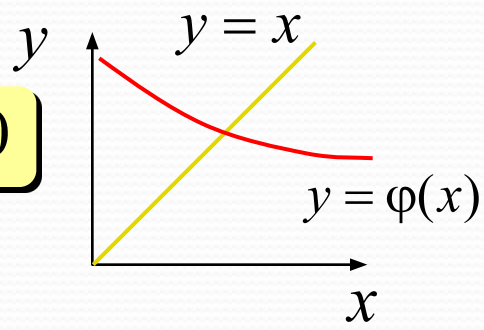


расходится

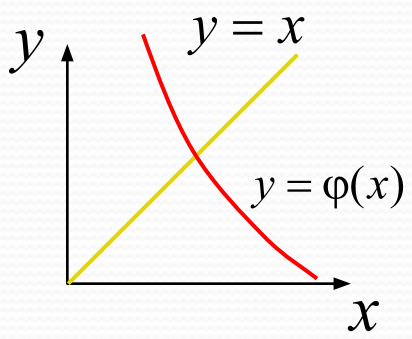
$\varphi'(x) > 1$



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



$\varphi'(x) < -1$



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, пробовать 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)

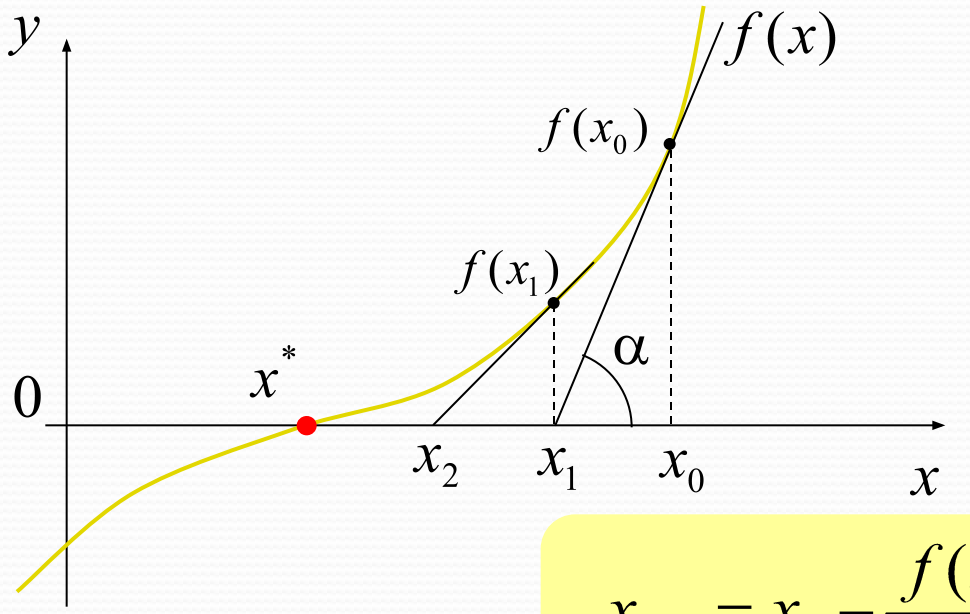
```
//-----------------------------------------------
// Iter решение уравнения методом итераций
// Вход:  x - начальное приближение
//        b - параметр
//        eps  - точность решения
// Выход: решение уравнения f(x)=0
//        n - число шагов
////---------------------------------------------
float Iter ( float x, float b, float eps, int &n)
{
  float dx;
  n = 0;
  while ( 1 ) {
     dx = b*f(x);
     x = x + dx;
     if ( fabs(dx) < eps ) break;
     n ++;
     if ( n > 100 ) break;
     }
  return x;
}
```

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

нормальный выход

аварийный выход (итерации расходятся)

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
//-----------------------------------------
float Newton ( float x, float eps, int &n)
{
  float dx;
  n = 0;
  while ( 1 ) {
     dx = f(x) / df(x);
     x = x - dx;
     if ( fabs(dx) < eps ) break;
     n ++;
     if ( n > 100 ) break;
     }
  return x;
}
```

```
float f ( float x ) {
  return 3*x*x*x+2*x+5;
}
float df ( float x ) {
  return 9*x*x + 2;
}
```

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

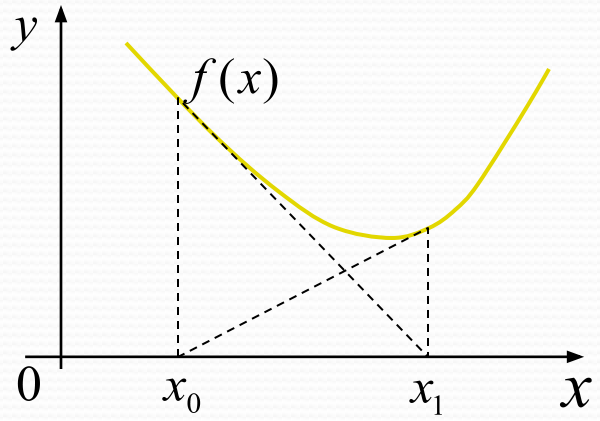
- нужно уметь вычислять производную (по формуле или численно)
- производная не должна быть равна нулю

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

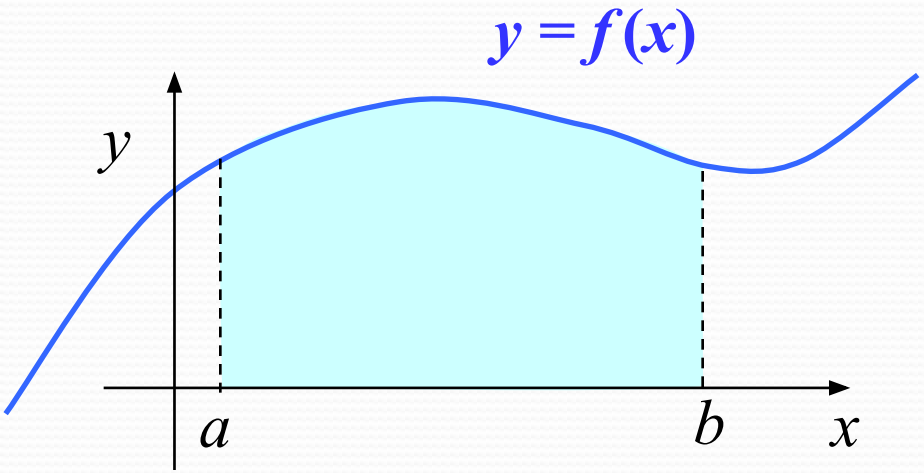
- может зацикливаться

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

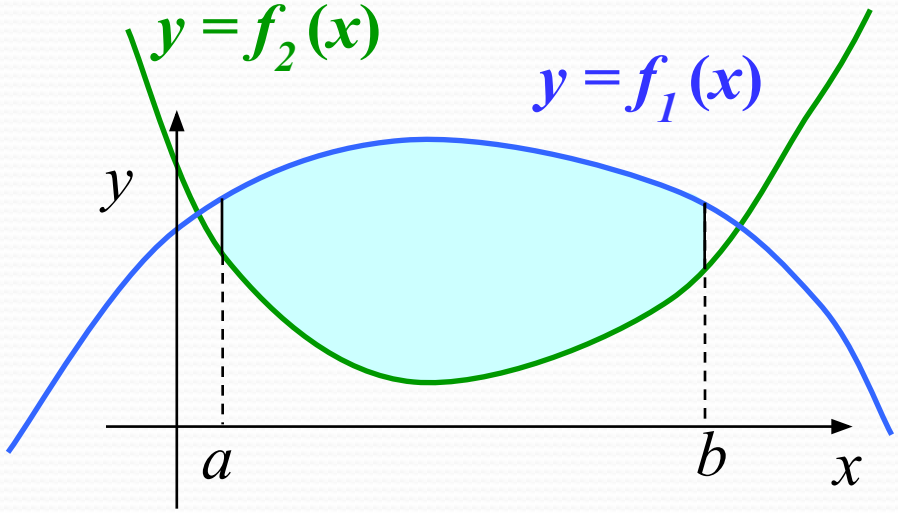
$$x_0 = 0$$



Площадь криволинейной трапеции

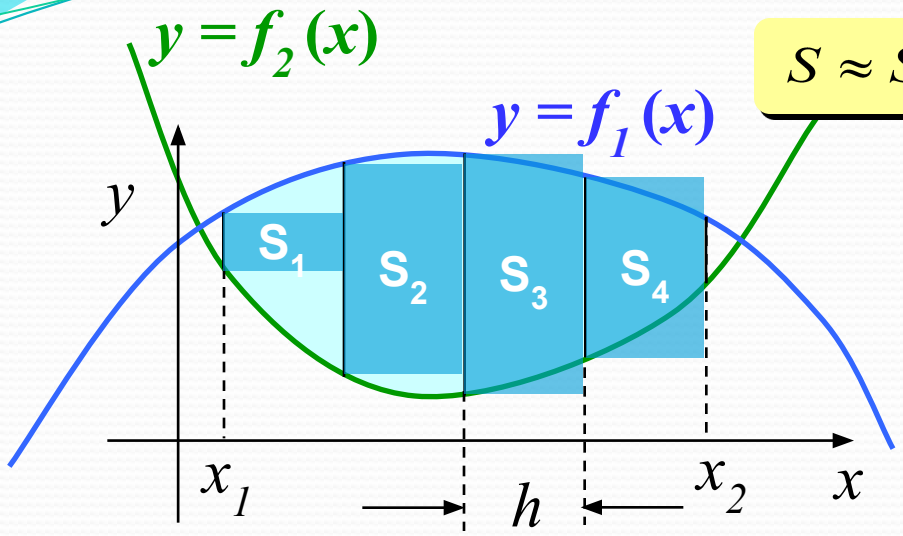


$$S = \int_a^b f(x)\,dx$$

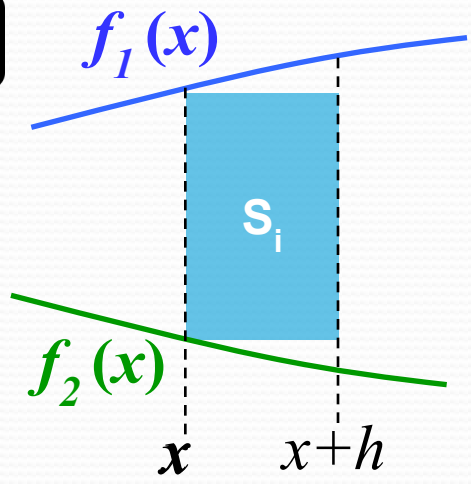


$$S = \int_a^b f_1(x)\,dx - \int_a^b f_2(x)\,dx$$

Метод (левых) прямоугольников



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



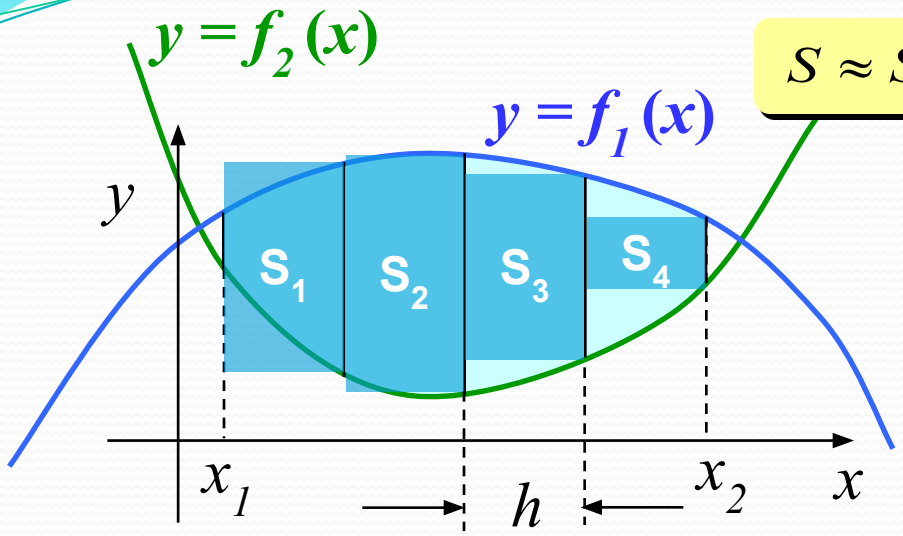
$$S_i = (f_1(x) - f_2(x)) \cdot h$$

```
float Area()
{
   float x, S = 0, h=0.001;
   for ( x = x1; x < x2; x += h)
      S += f1(x) - f2(x);
   return S * h;
}
```

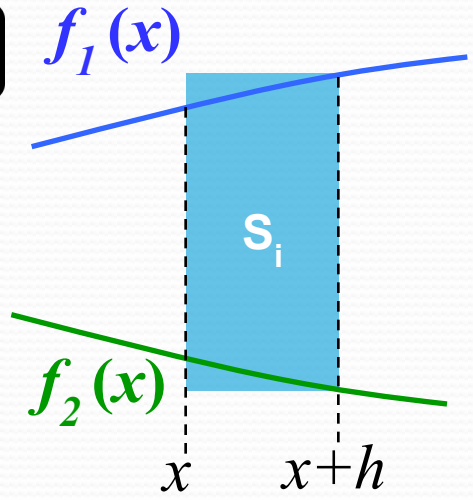
Почему не x <= x2?

Как улучшить решение?

Метод (правых) прямоугольников



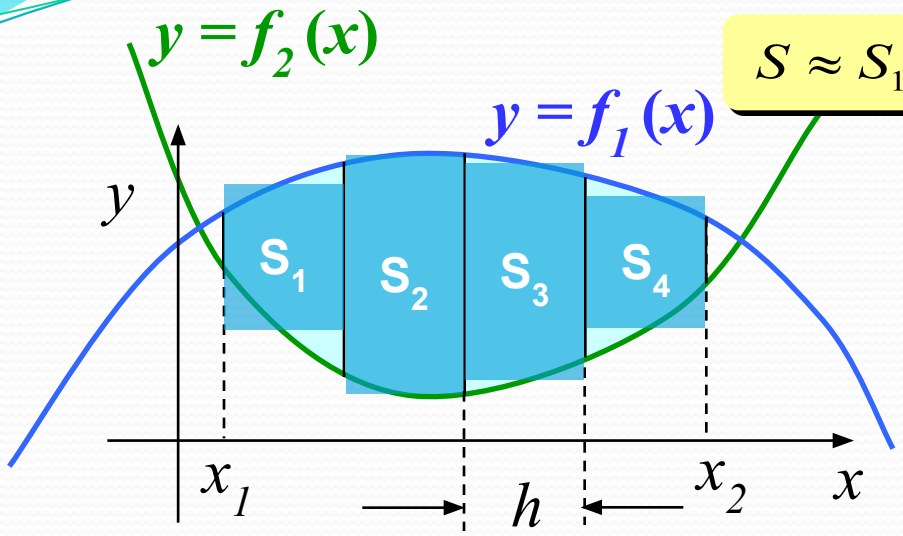
$$S \approx S_1 + S_2 + S_3 + S_4$$



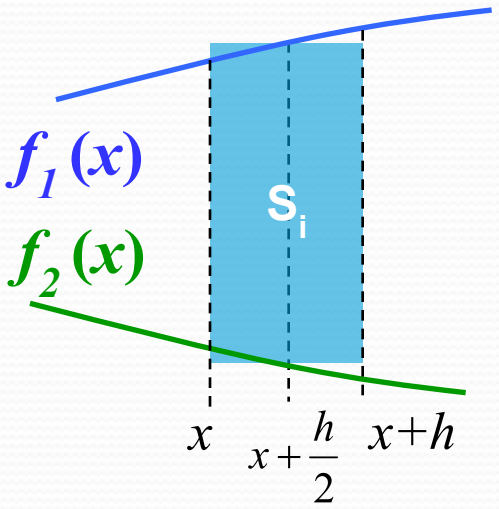
$$S_i = (f_1(x+h) - f_2(x+h)) \cdot h$$

```
float Area()
{
   float x, S = 0, h=0.001;
   for ( x = x1; x < x2; x += h )
      S += f1(x+h) - f2(x+h);
   S *= h;
   return S;
}
```

Метод (средних) прямоугольников



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



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

```
float Area()
{
   float x, S = 0, h=0.001;
   for ( x = x1; x < x2; x += h )
      S += f1(x+h/2) - f2(x+h/2);
   S *= h;
   return S;
}
```

Какой метод точнее?

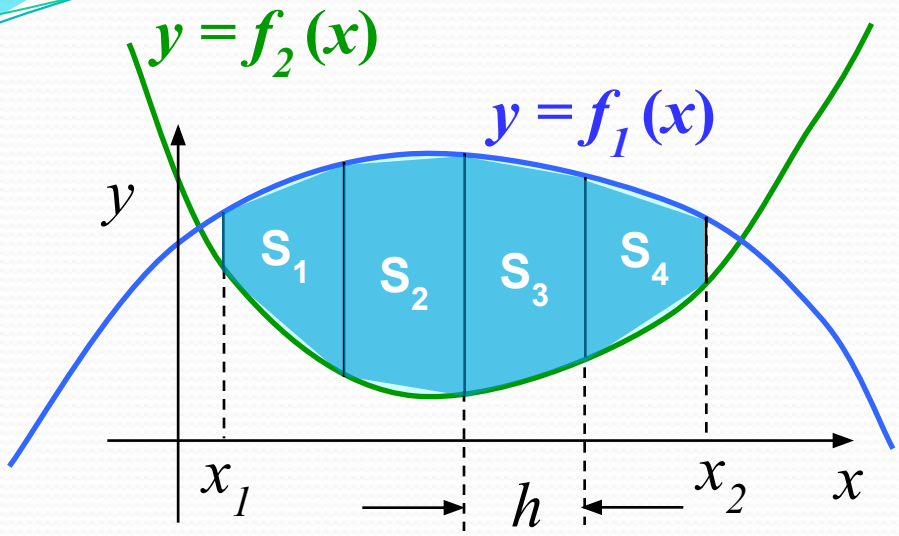
левые (правые):

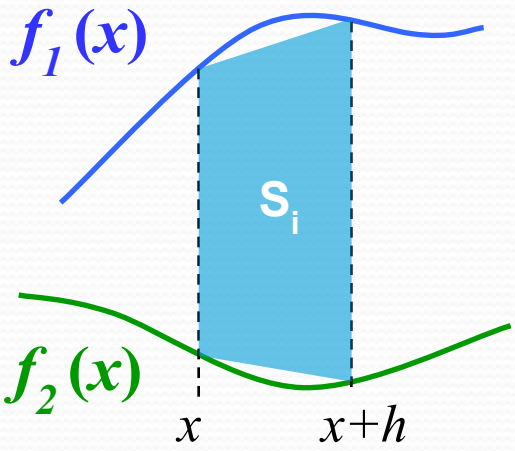
$$\varepsilon = O(h)$$

средние

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

Метод трапеций





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

```
for ( x = x1; x < x2; x += h )
S
```

```
S = ( f1(x1) - f2(x1)
    + f1(x2) - f2(x2) )/2.;
for ( x = x1+h; x < x2; x += h )
    S += f1(x) - f2(x);
S *= h;
```

Как улучшить?

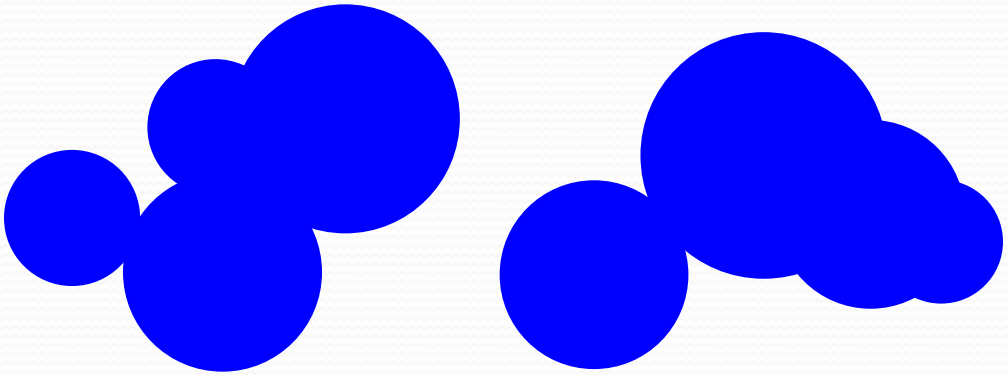
Ошибка $\varepsilon = O(h^2)$

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.

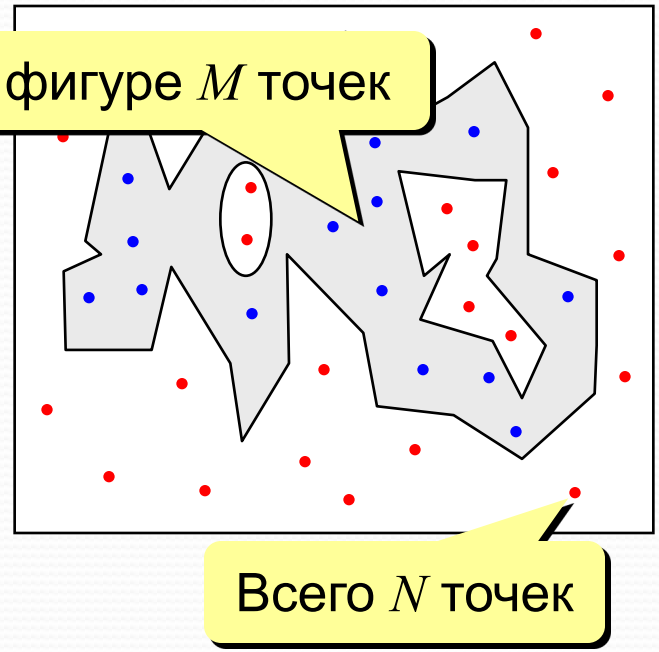
Пример: заданы 100 кругов (координаты центра, радиусы), которые могу пересекаться. Найти площадь области, перекрытой кругами.



? 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**: $\frac{S}{S_0} \approx \frac{M}{N} \Rightarrow S \approx S_0 \cdot \frac{M}{N}$

!

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