ПРИБЛИЖЕННОЕ

ВЫЧИСЛЕНИЕ ИНТЕГРАЛОВ

Формулы для вычисления интеграла

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

получают следующим образом.

Интервал [a, b] разбивают на n отрезков длиной $h = (b - a) / n$ (в общем случае разной длины), тогда значение интеграла по всей области равно сумме интегралов на этих отрезках.

На каждом отрезке [x_i, x_{i+1}] выбирают 1 – 5 узлов и по ним строят интерполяционный многочлен соответствующего порядка. Вычисляют интеграл от этого многочлена на отрезке.

В результате получают выражение интеграла (формулу численного интегрирования) через значения подынтегральной функции в выбранной системе точек.

Такие выражения называют ***квадратурными формулами.***

Рассмотрим наиболее часто используемые квадратурные формулы для отрезков равной длины:

$$h = (b - a) / n;$$

$$x_i = a + (i - 1) \cdot h; \quad i = 1, 2, \ldots, n.$$

Формула средних

Формула средних получается, если на каж-дом ***i***-м отрезке взять один центральный узел $x_{i+1/2} = (x_i + x_{i+1})/2$, соответствующий середине отрезка. Функция на каждом отрезке аппрокси-мируется многочленом нулевой степени (конста-нтой) $P_0(x) = y_{i+1/2} = f(x_{i+1/2})$. Заменяя площадь криволинейной фигуры площадью прямоуголь-ника высотой $y_{i+1/2}$ и основанием h, получим приближенную формулу для расчета (рис. 1):

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} P_0(x)dx = h\sum_{i=1}^{n} y_{i+1/2} = \quad . (1)$$

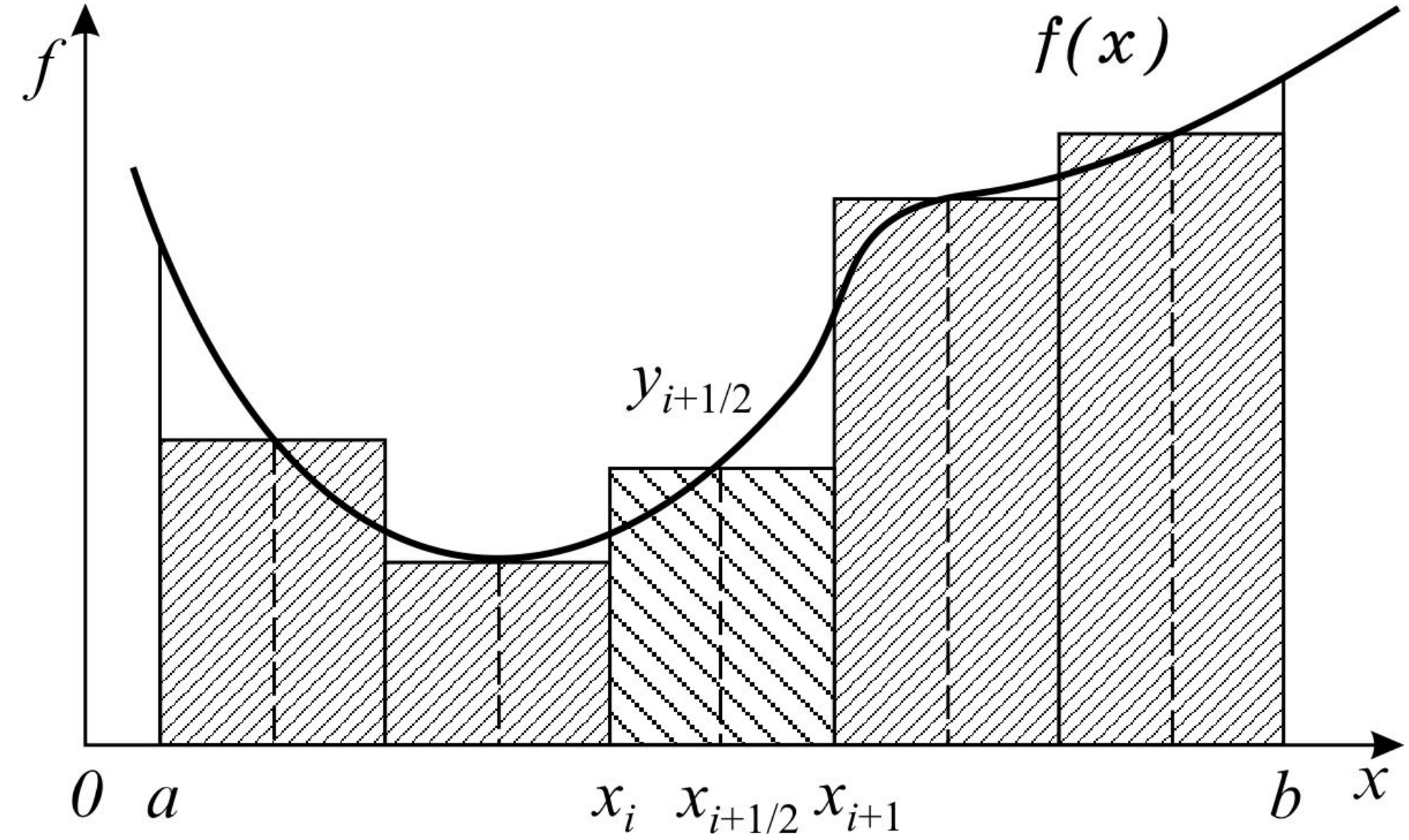


Рис. 1. Иллюстрация формулы средних

Формула трапеций

Формула трапеций получается при аппроксимации функции $f(x)$ на каждом отрезке $[x_i, x_{i+1}]$ интерполяционным многочленом первого порядка, т.е. прямой, проходящей через точки (x_i, y_i), (x_{i+1}, y_{i+1}). Площадь криволинейной фигуры заменяется площадью трапеции с основаниями y_i, y_{i+1} и высотой h (рис. 2):

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} P_1(x)dx = h\sum_{i=1}^{n} \frac{y_i + y_{i+1}}{2} = \tag{2}$$

$$= h\left[\frac{y_1 + y_{n+1}}{2} + \sum_{i=2}^{n} y_{iTP}\right] = \quad .$$

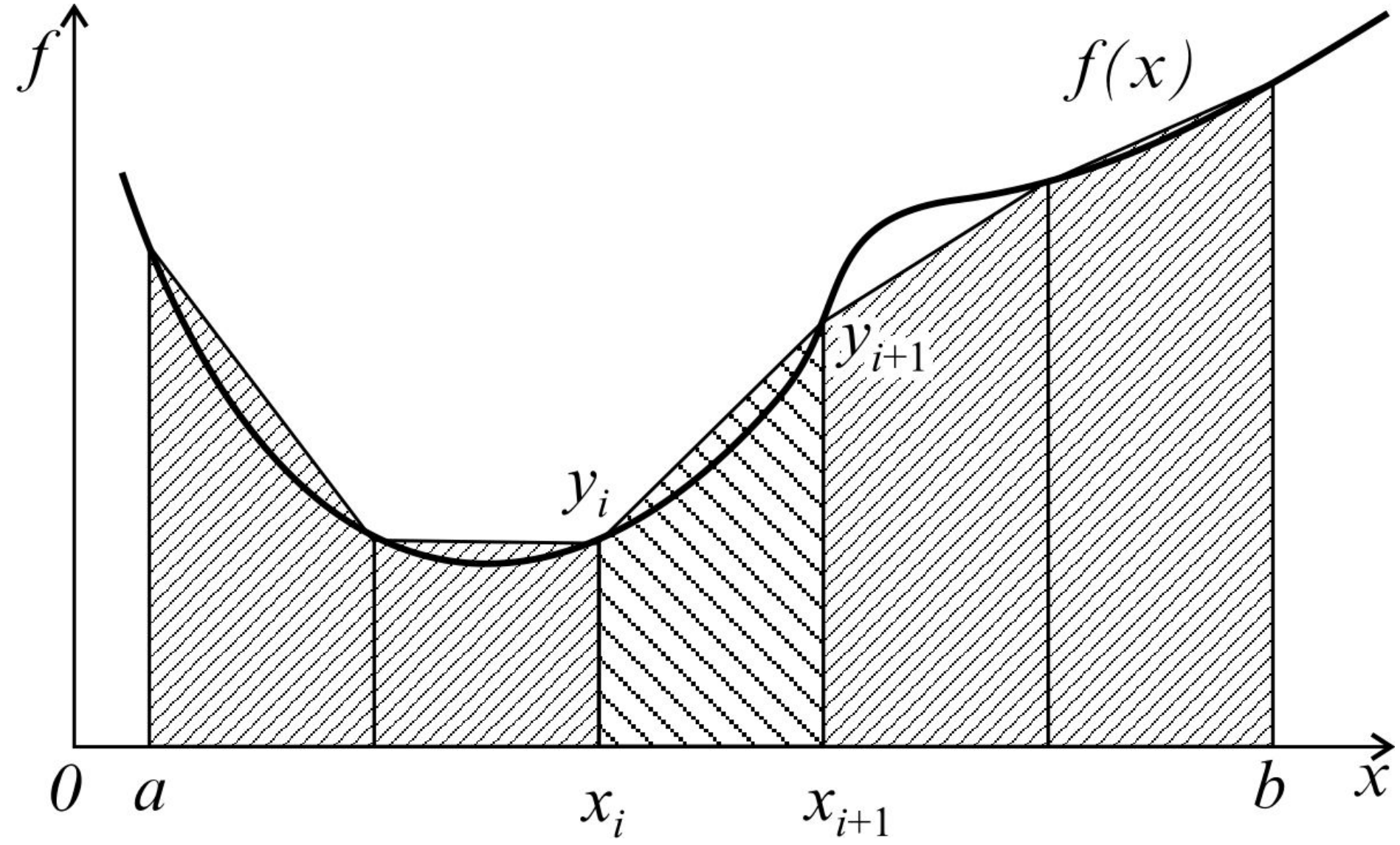


Рис. 2. Иллюстрация формулы трапеций

Формула Симпсона

Эта формула получается при аппроксима-ции функции $f(x)$ на каждом отрезке $[x_i, x_{i+1}]$ интерполяционным многочленом второго порядка (параболой) с узлами x_i, $x_{i+1/2}$, x_{i+1}. После интегрирования параболы получаем (рис. 3):

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} P_2(x)dx = \tag{3}$$

$$= \frac{h}{6} \sum_{i=1}^{n} (y_i + 4y_{i+1/2} + y_{i+1}) = \quad .$$

После приведения подобных членов полу-чаем более удобный для программирования вид:

$$\Phi_{СИМ} = \frac{h}{3} \cdot \left[\frac{y_1 + 4y_{1+1/2} + y_{n+1}}{2} + \sum_{i=2}^{n} (2y_{i+1/2} + y_i) \right].$$

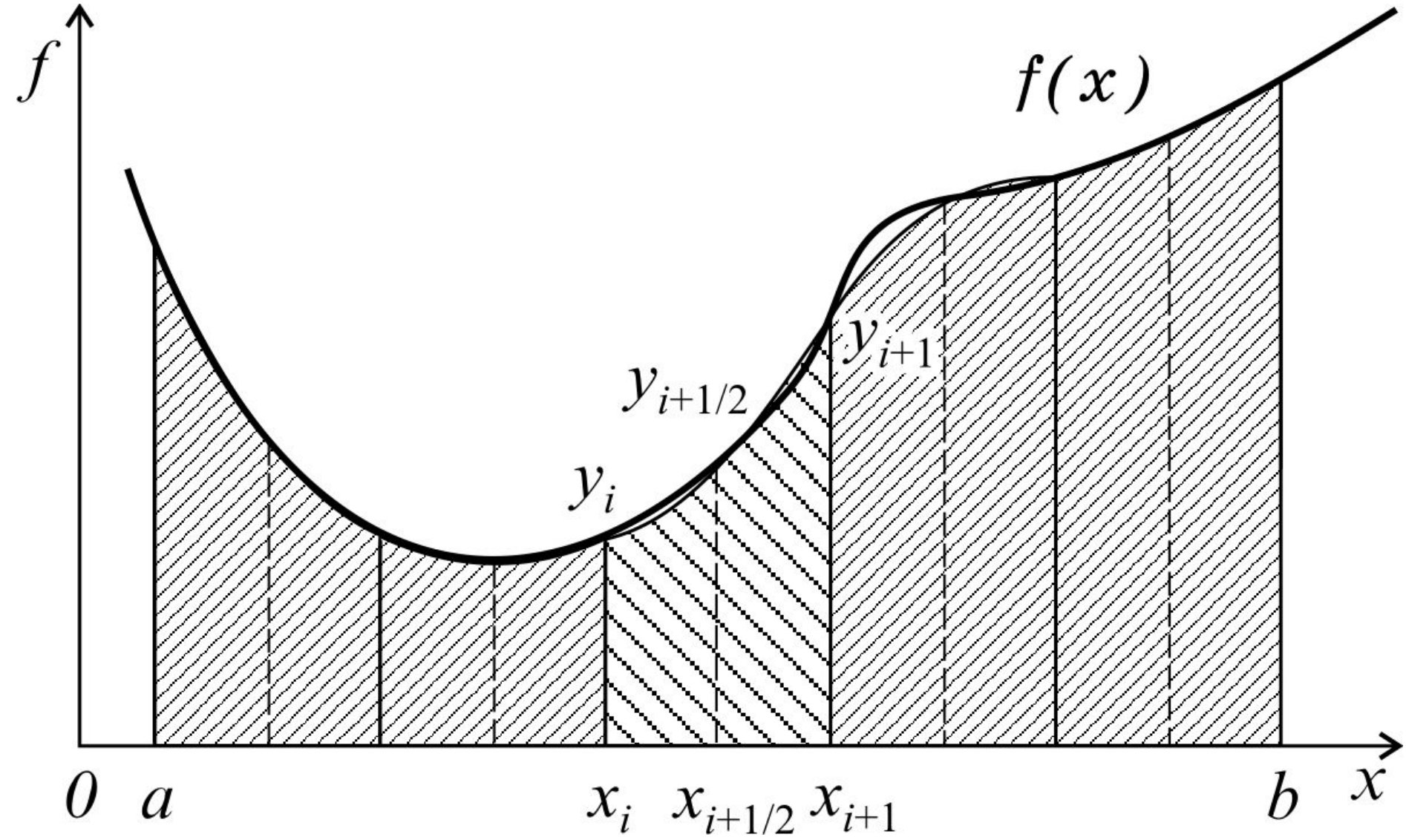


Рис. 3. Иллюстрация формулы Симпсона

Погрешность формулы трапеций в два раза больше, чем погрешность формулы средних:

$$\varepsilon_{TP} = \max \left| \int_a^b f(x)\,dx - \quad _{TP} \right| \le \frac{h^2}{12} \int_a^b x''( )\,dx \quad .$$

Погрешность формулы Симпсона имеет четвертый порядок по $\boldsymbol{h}$:

$$\varepsilon_{CИМ} = \max \left| \int_a^b f(x)\,dx - \quad _{CИМ} \right| \le \frac{h^4}{2880} \int_a^b x^{(4)}( )\,dx \quad .$$

Расчет интеграла по заданной точности ε

***Метод* 1**. Один из вариантов вычисления интеграла с заданной точностью:

1) задают начальное число разбиений n и вычисляют приближенное значение интеграла $I1$ выбранным методом;

2) число интервалов удваивают $n = 2n$;

3) вычисляют новое значение интеграла $I2$;

4) если $|I1 - I2| \geq \varepsilon$, то $I1 = I2$ и расчет повторяют – переход к п. 2; иначе ($|I1 - I2| < \varepsilon$) – заданная точность достигнута, выводят результаты: $I2$ – найденное значение интеграла с заданной точностью ε и n – количество интервалов.

Метод 2 – классическая схема автоматического выбора шага.

Анализ формул (1) – (3) показал, что точное значение интеграла находится между значения-ми $\boldsymbol{\Phi_{СР}}$ и $\boldsymbol{\Phi_{ТР}}$ и выполняется соотношение

$$\boldsymbol{\Phi_{СИ} = (\Phi_{ТР} + 2 \cdot \Phi_{СР}) / 3}. \qquad (4)$$

Расчет начинается с $\boldsymbol{n = 2}$ и производится по двум методам $\boldsymbol{\Phi_{СР}}$ и $\boldsymbol{\Phi_{ТР}}$.

Если $\boldsymbol{|\Phi_{СР} - \Phi_{ТР}| \geq \varepsilon}$, увеличивают $\boldsymbol{n = 2n}$ и расчет повторяют.

Если точность достигнута, то окончательное значение интеграла находят по формуле (4).

Формулы Гаусса

В рассмотренных формулах в качестве узлов многочлена выбирались середины и (или) концы интервала разбиения.

Оказывается, что увеличение количества узлов не всегда приводит к уменьшению погре-шности, т.е. за счет удачного расположения узлов можно значительно увеличить точность.

Суть методов Гаусса порядка ***n*** состоит в таком расположении ***n*** узлов интерполяционного многочлена на отрезке $[x_i, x_{i+1}]$, при котором достигается минимум погрешности квадратурной формулы.

Анализ показал, что узлами, удовлетворяющими такому условию, являются нули ортогонального многочлена Лежандра степени n :

$$\{ \varphi_1(x) = 1; \quad \varphi_2(x) = x;$$

$$\varphi_{k+1}(x) = [\, (2k + 1)\, x\, \varphi_k(x) - k\, \varphi_{k-1}(x)],$$

$$k = 2, 3, \ldots, n \} .$$

Так, для $n = 1$ один узел должен быть выбран в центре.

Следовательно, метод средних является методом ***Гаусса 1-го порядка***.

Для $n = 2$ узлы должны быть выбраны следующим образом:

$$x_i^1 = x_{i+1/2} - h/2 \cdot 0{,}5773502692;$$
$$x_i^2 = x_{i+1/2} + h/2 \cdot 0{,}5773502692;$$

и формула ***Гаусса 2-го порядка*** имеет вид:

$$\int_a^b f(x)dx \approx \frac{h}{2}\sum_{i=1}^{n}[f(x_i^1) + f(x_i^2)].$$

Погрешность этой формулы при $h \to 0$ такой же, как у метода Симпсона, хотя используется только 2 узла.

Для $n = 3$ выбираются узлы:

$$x_i^0 = x_{i+1/2};$$

$$x_i^1 = x_i^0 - h/2 \cdot 0{,}7745966692;$$

$$x_i^2 = x_i^0 + h/2 \cdot 0{,}7745966692;$$

и формула ***Гаусса 3-го порядка*** имеет вид:

$$\int_a^b f(x)dx \approx \frac{h}{2}\sum_{i=1}^{n}\left(\frac{8}{9}f(x_i^0) + \frac{5}{9}f(x_i^1) + \frac{5}{9}f(x_i^2)\right).$$

Погрешность этой формулы при $h \to 0$ имеет 7-й порядок, что значительно выше, чем у формулы Симпсона, практически при одинаковых вычислительных затратах.

Рассмотрим пример

Написать и отладить программу вычисления значения интеграла от функции $f(x) = 4x - 7\sin x$ на интервале [a, b] методом ***Симпсона*** с выбором: по заданному количеству разбиений n и заданной точности ε (***метод*** **1**).

На интервале [–2, 3] интеграл равен 5,983.

Текст программы в ***консольном приложе-нии*** может иметь вид:

```
. . .
double fun (double);
double Metod (double (*f )(double), double, double, int);
// Декларации прототипов функций Пользователя
```

```
void  main ()  {
    double  a,  b,  x,  eps,  h,  I1,  I2,  pogr;
    int  n,  n1,  kod;
    cout << " If a = -2, b = 3, Int = 5,983" << endl;
// Цикл, организующий повторение расчетов
    do {
        cout << " Input a, b : ";
        cin >> a >> b;
/* Выбор расчета: 1 – по разбиению на n, иначе по
точности eps  */
        cout << "\n\t Input 1 – n,  Else – eps : ";
        cin >> kod;
```

```
    if ( kod == 1 )   {
// Выполняем расчет по числу разбиений n
        cout << " Input n : ";
        cin >> n;
                I1 = Metod ( fun, a, b, n );
    }
    else {
// Иначе, выполняем расчет по точности eps
        cout << " Input eps : ";
        cin >> eps;
                n1 = 2;
// Начальное число разбиений n1, интеграл – I1
                I1 = Metod ( fun, a, b, n1 );
```

```
    do {
/* Увеличиваем число разбиений и находим новое
значение интеграла  I2  */
                n1 *= 2;
                I2 = Metod ( fun, a, b, n1);
                pogr = fabs (I2 – I1);
                I1 = I2;
        } while ( pogr > eps );
/* Цикл выполняем пока разница между предыду-
щим значением интеграла I1  и найденным  I2  не
станет меньше eps  */
```

```
        cout << "\t n = " << n1 << endl;
/* Выводим количество разбиений n1, при котором была достигнута заданная точность  */
        }       // Конец  else
    cout << "\n\t Integral = " << I1 << endl;
// Выводим найденное значение интеграла I1
    cout << "\n Repeat - 1, Else - EXIT " << endl;
/* Для повторения (Repeat) введите 1, чтобы условие getch() == '1' в следующем операторе while было истинным, иначе – конец программы  */
    }   while ( getch() == '1' );
}
```

Функция метода Симпсона

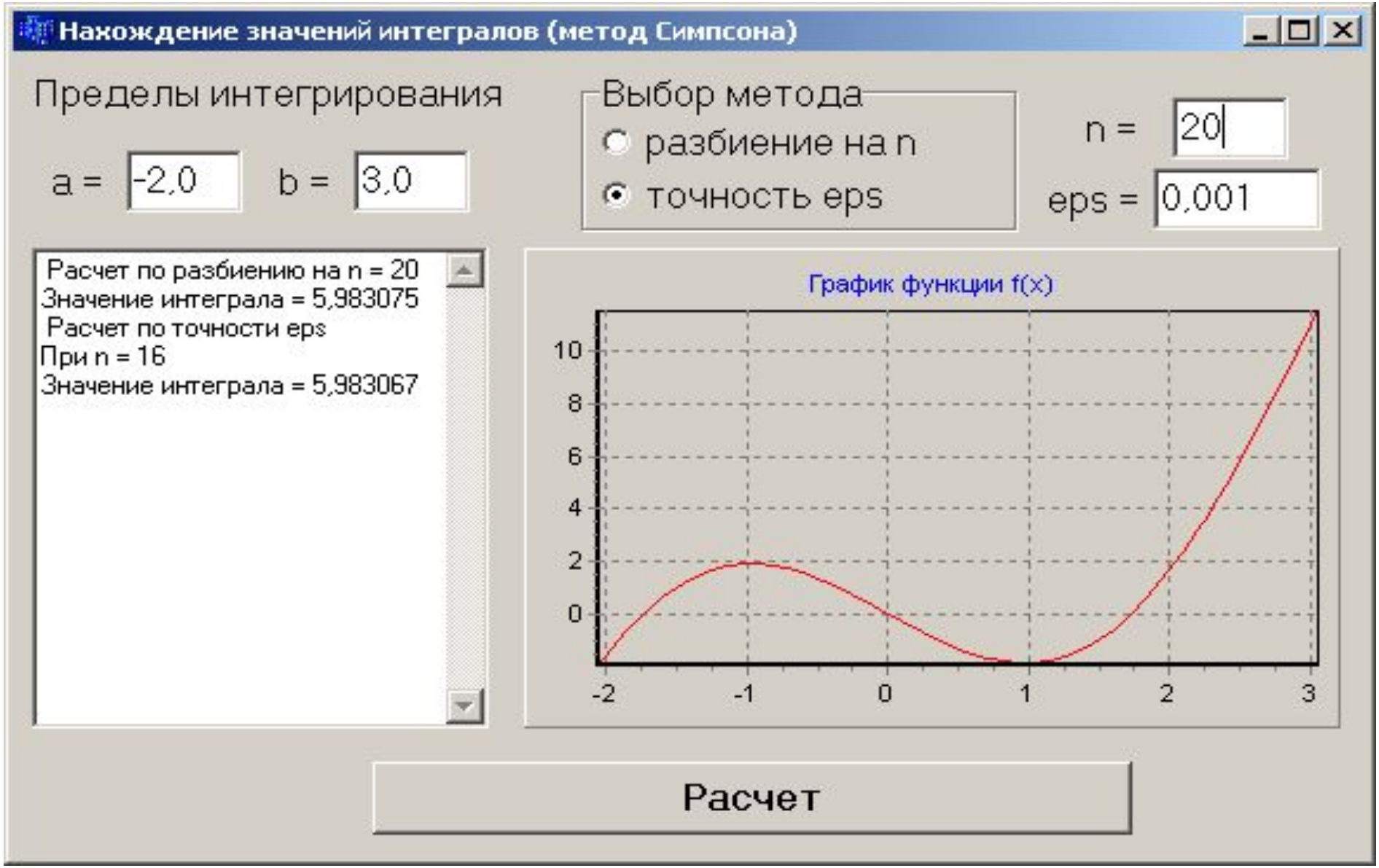
```
double Metod (double ( *f ) (double x), double a,
                        double b, int n)
{
    double s = 0, h, x;
    h = (b - a) / n;          // Находим шаг
    x = a;
// Выполняем расчеты согласно формуле (3)
   for ( int i = 1; i <= n; i++) {
     s += f (x) + 4 * f (x + h/2) + f (x + h);
     x += h;
     }
    return  s * h / 6;
}
```

Вид подынтегральной функции ***f (x)***

```
double fun (double x)
{
    return 4*x - 7*sin(x);
}
```

Пример в оконном приложении

Панель диалога:



Текст программы:

. . .

typedef double (***type_f***) (double);

type_f – тип указателя на функцию с одним параметром ***double*** и результатом ***double***

// Декларации прототипов функций Пользователя

double ***fun*** (double);
double ***Simps*** (***type_f***, double, double, int);

```
//---------- Кнопка РАСЧЕТ --------------------
 Заголовок функции (например, Button1Click() )
 {
    double a, b, x, eps, h, Int1, Int2, pogr;
    int n, n1;
    a = StrToFloat(Edit1->Text);
    b = StrToFloat(Edit2->Text);
    eps = StrToFloat(Edit3->Text);
    n = StrToInt(Edit4->Text);
    Chart1->Series[0]->Clear();
    for(x = a - h; x< b + h; x += 0.001)
              Chart1->Series[0]->AddXY (x, fun(x));
```

```
switch ( RadioGroup1->ItemIndex )   {

   case 0:
            Memo1->Lines->Add
       ("Расчет по разбиению на n = "
        + IntToStr(n));
            Int1 = Simps(fun,a,b,n);
      break;
```

```
case 1:
   n1=2;
      Memo1->Lines->Add ( "Расчет по eps" );
      Int1 = Simps (fun, a, b, n1);
      do {
                  n1*=2;
                  Int2 = Simps (fun, a, b, n1);
                  pogr = fabs ( Int2 - Int1 );
                  Int1 = Int2;
       } while ( pogr > eps );
       Memo1->Lines->Add("n = " +IntToStr(n1));
  break;
}      // Конец оператора switch
```

```
Memo1->Lines->Add("Значение интеграла = " +
FloatToStrF(Int1,ffFixed,8,6));
}   // Конец функции-обработчика
```

```
//------------ Функция Метода Симпсона ----------
double Simps ( type_f f, double a, double b, int n)
{
   double s = 0, h, x;
   h = (b - a) / n;
   x = a;
   for ( int i = 1; i<=n; i++)  {
    s += f (x) + 4 * f (x + h/2) + f (x + h);
    x += h;
   }
   return s * h / 6;
}
```

```
//---------- Функция f(x) ----------------

  double fun(double x)
  {
     return 4*x - 7*sin(x);
// На интервале [-2, 3] значение 5,983
  }
```