Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

**Лабораторная работа № 7**

**«Численное дифференцирование и интегрирование функции»**по учебной дисциплине «Методы численного анализа»

**Выполнили:**

студент гр. 153504 Климкович Н. В.  
студент гр. 153504 Тиханёнок И. А.  
 студент гр. 153504 Тарасенко Ф. П.

**Проверила:**

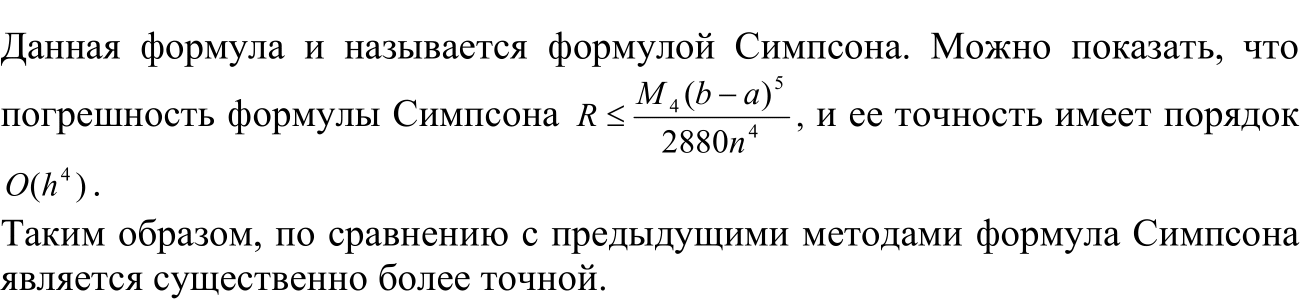
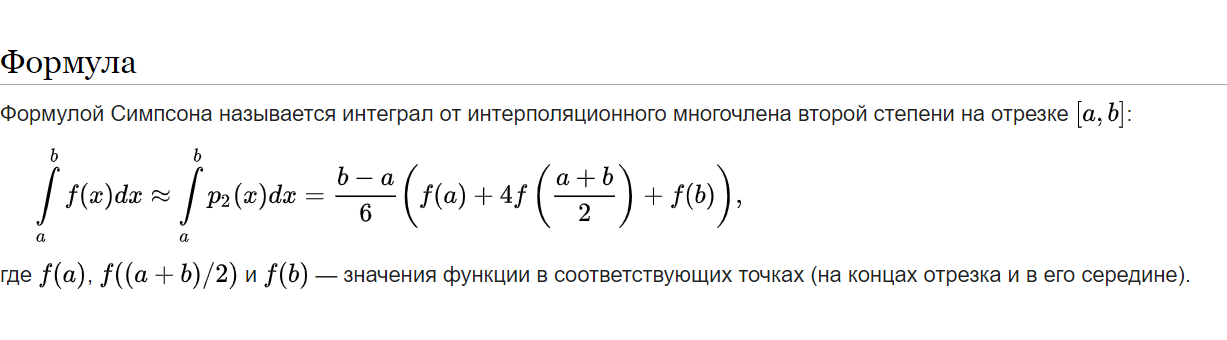
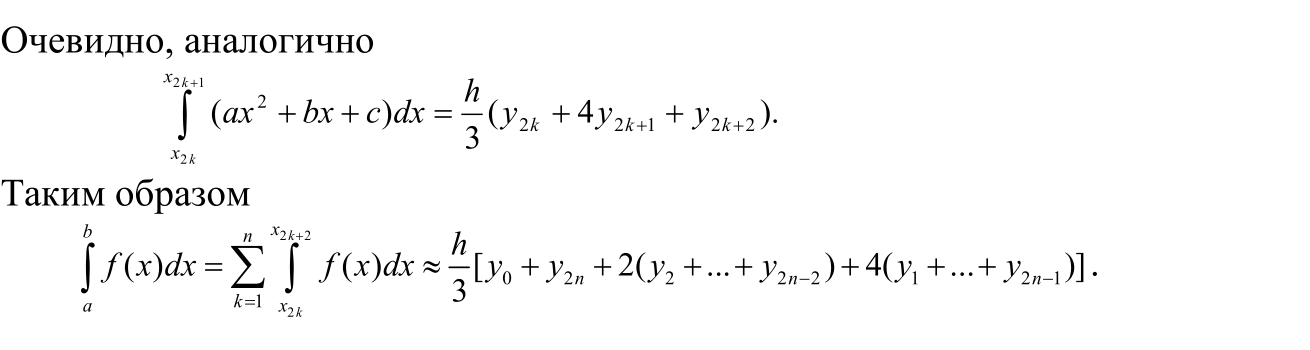
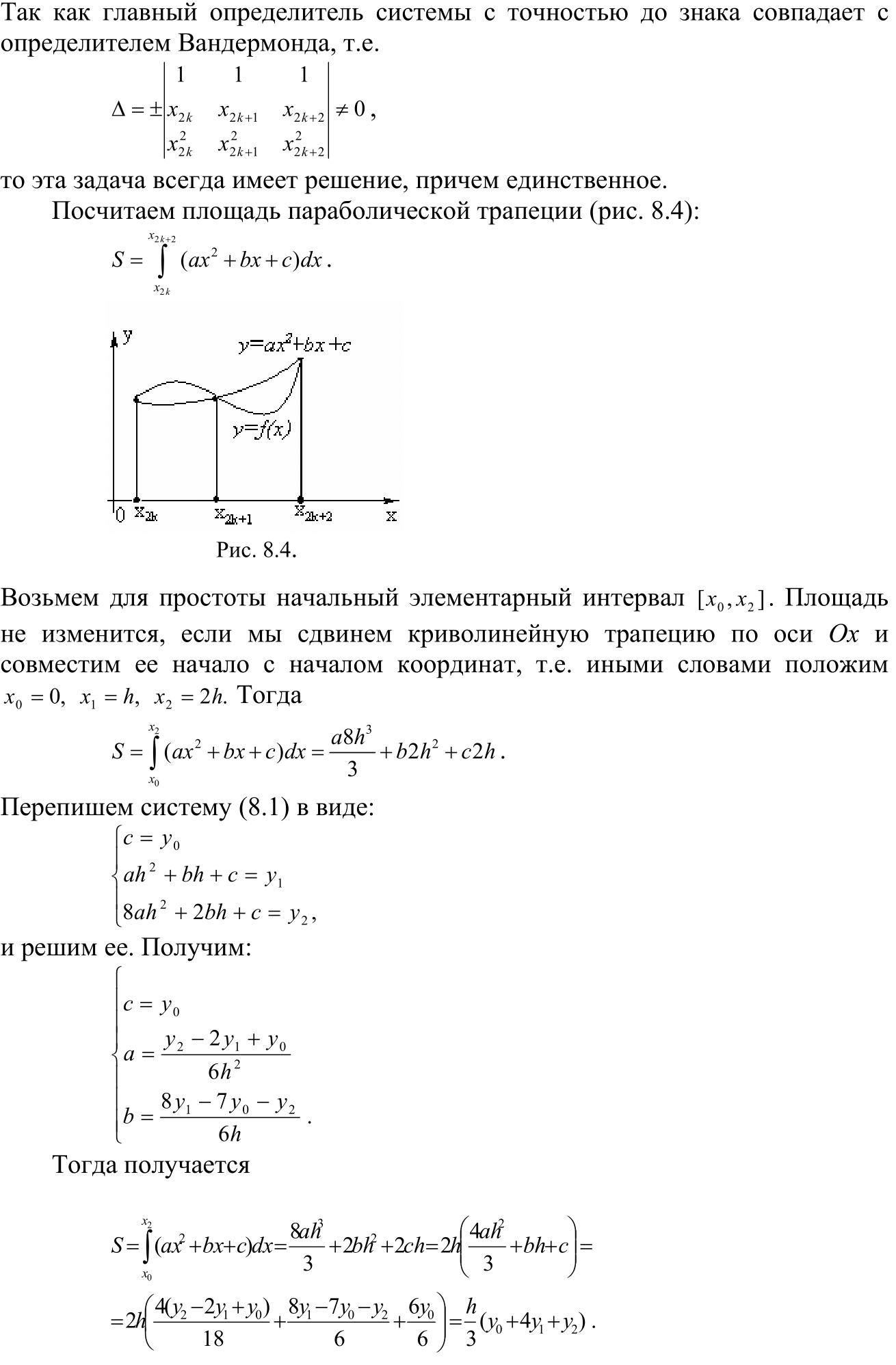
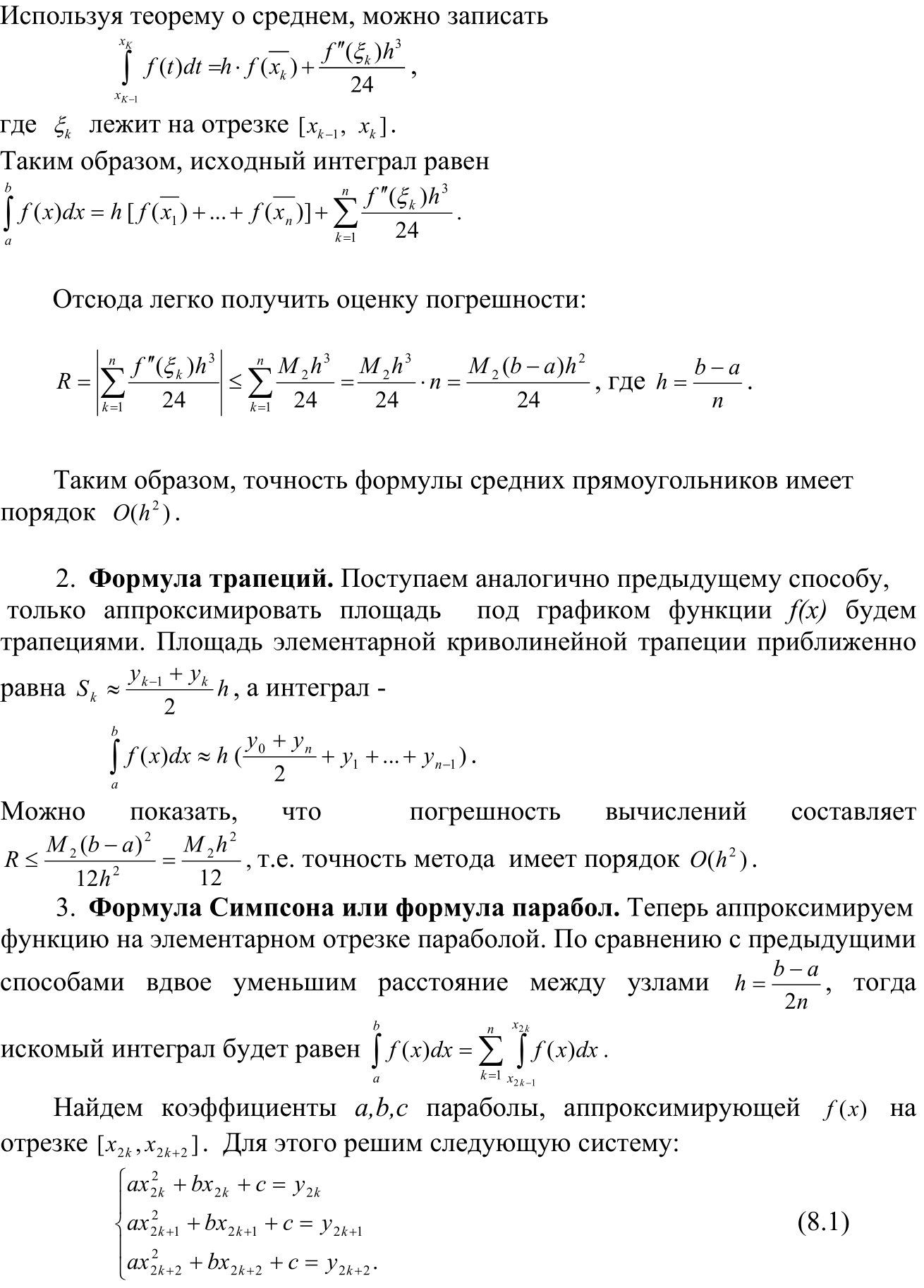
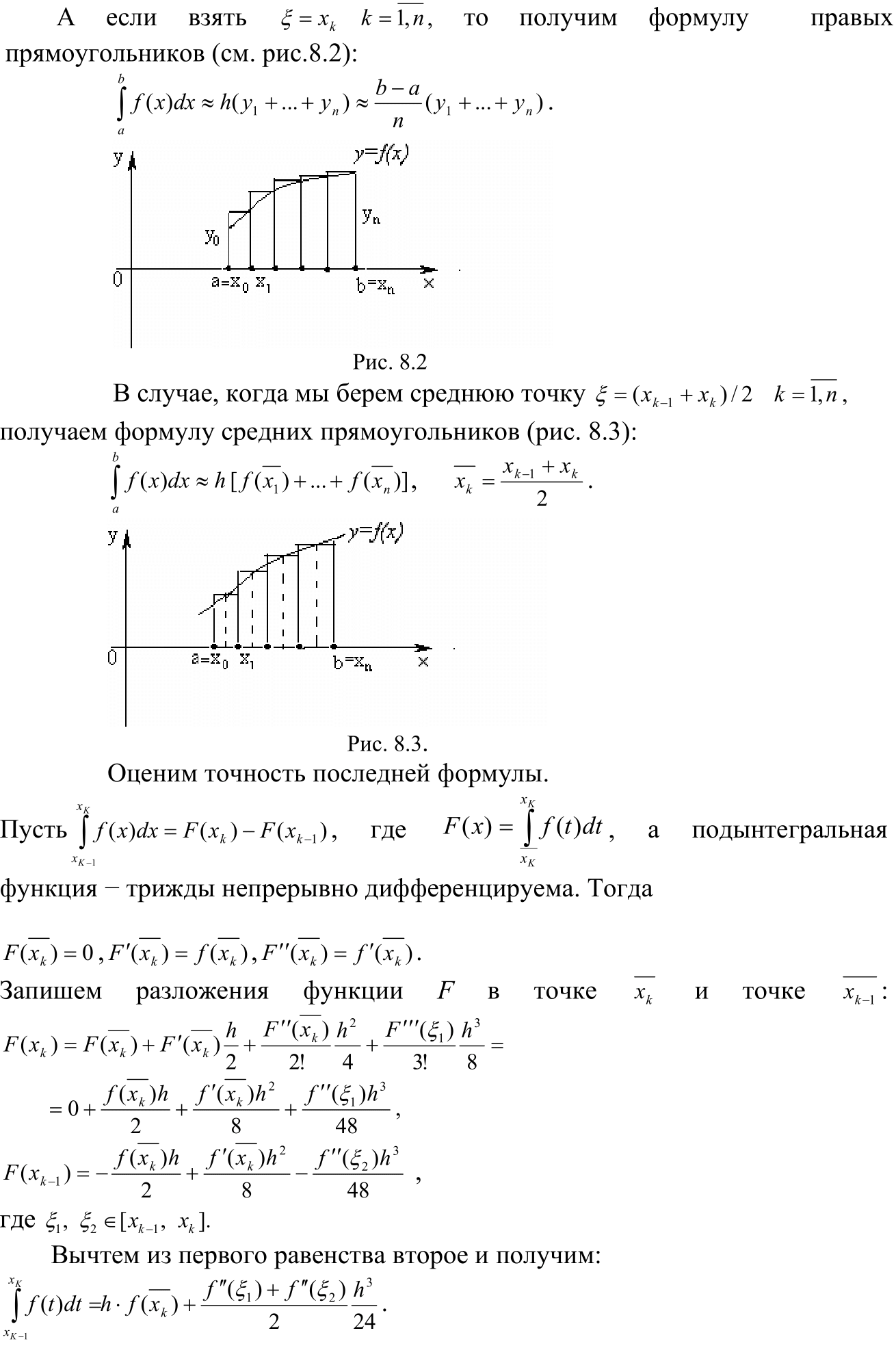
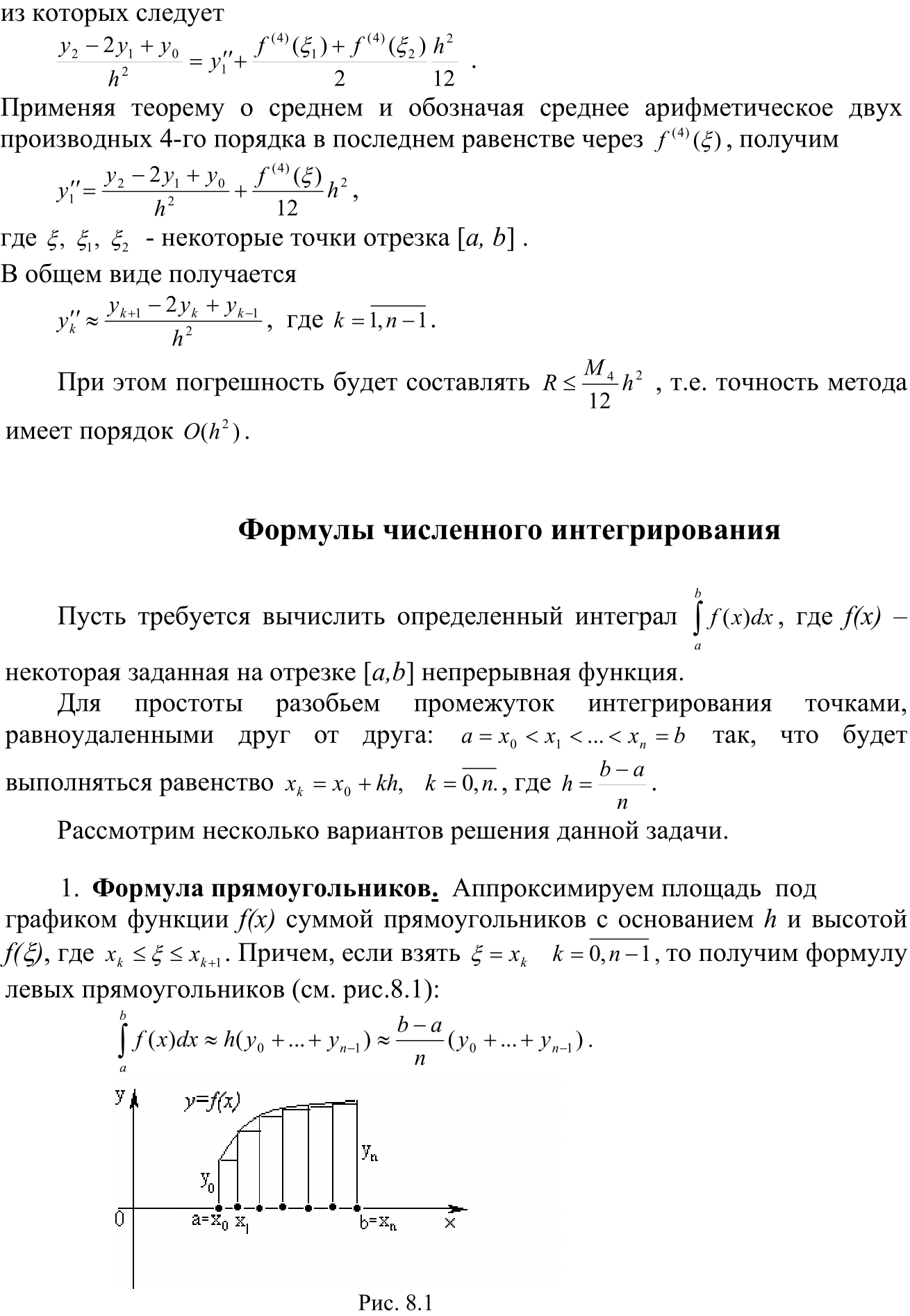
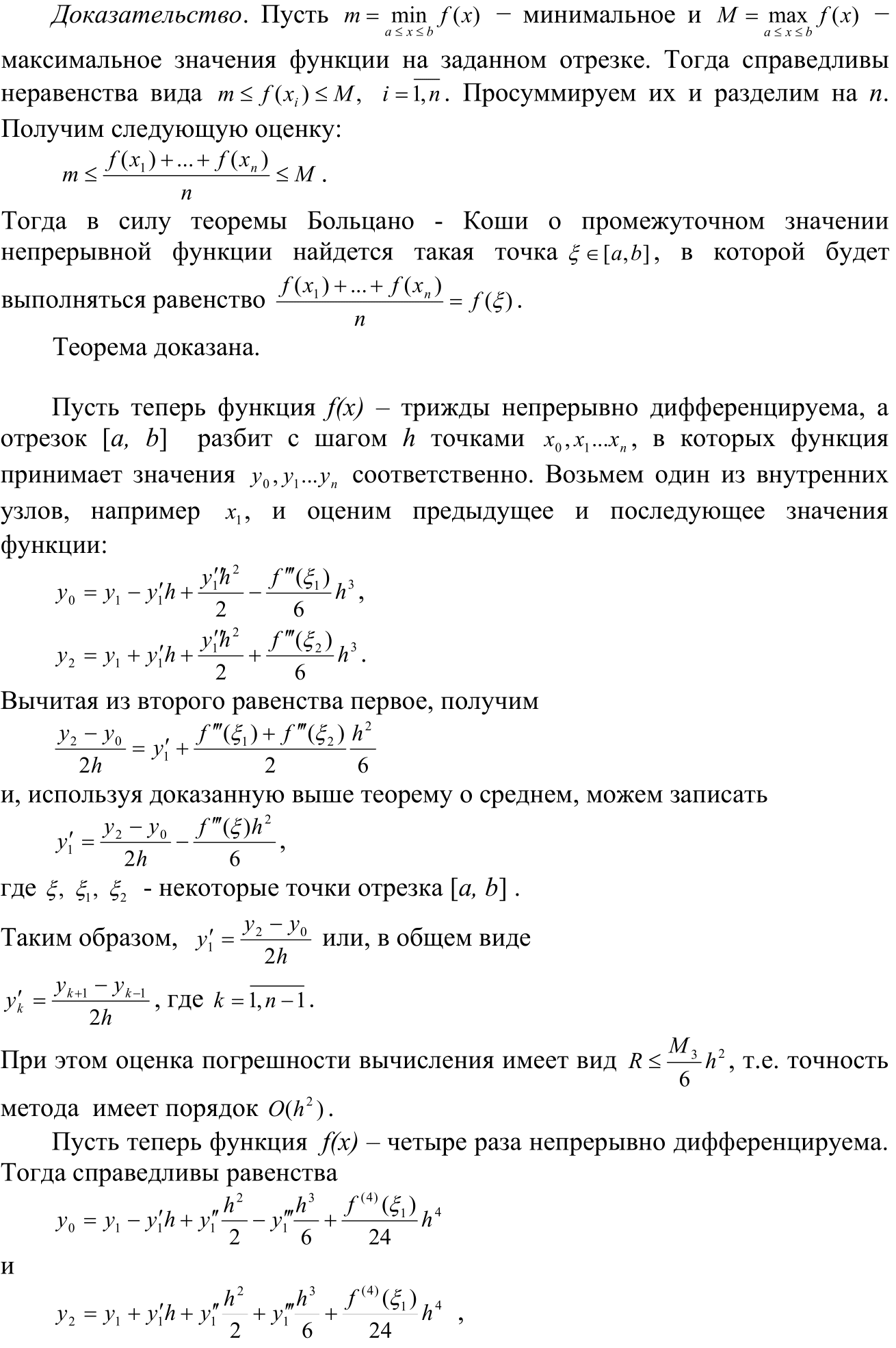
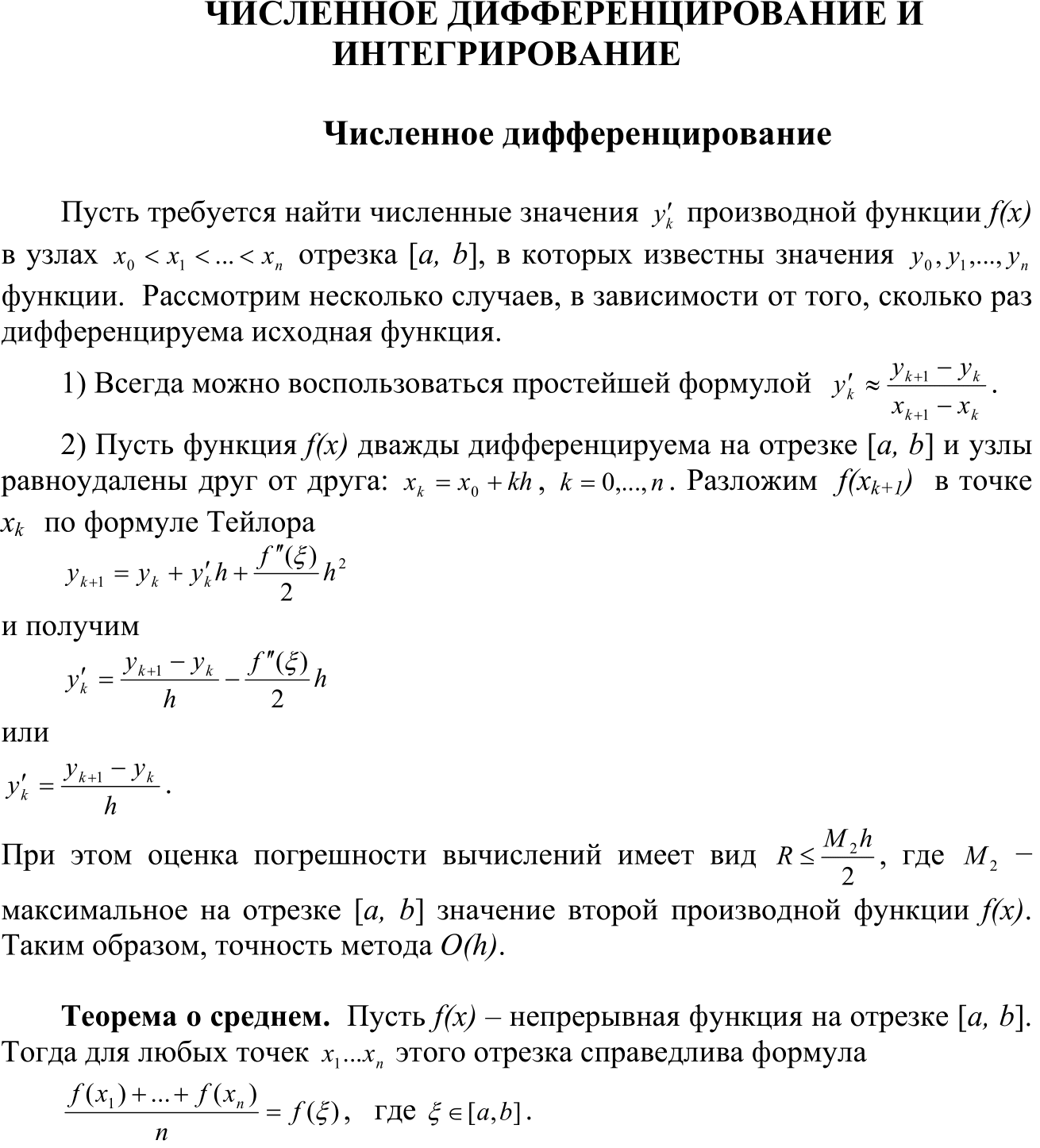
ст. преподаватель кафедры информатики Стройникова Е. Д.

Минск 2022

**Цель работы:**

Изучить методы численного вычисления производных и методы численного интегрирования. Сравнить методы по трудоёмкости, точности. Выполнить тестовое задание по численному дифференцированию и интегрированию.

**Краткие теоретические сведения:**



**Программная реализация:**

print("Numerical Differentiation and Integration \n")  
  
np.random.seed(42)  
  
  
L, R, DerXdot = 0, 2, 1  
def f(x): return np.arctan(np.sqrt(x))  
def F(x): return x \* np.arctan(np.sqrt(x)) - np.sqrt(x) + np.arctan(np.sqrt(x))  
def fd1(x): return 1/(2\*(x\*\*(1/2))\*(1+x))  
def fd2(x): return -1/(4\*x\*\*(3/2)\*(1+x))-1/(2\*(x\*\*(1/2))\*(1+x)\*\*2)  
def fd3(x): return 3/(8\*x\*\*(5/2)\*(1+x))+1/(2\*x\*\*(3/2)\*(1+x)\*\*2)+1/((x\*\*(1/2))\*(1+x)\*\*3)  
def fd4(x): return -15/(16\*x\*\*(7/2)\*(1+x))-9/(8\*x\*\*(5/2)\*(1+x)\*\*2)-3/(2\*x\*\*(3/2)\*(1+x)\*\*3)-3/((x\*\*(1/2))\*(1+x)\*\*4)  
M2deLR, M4deLR = 0, 0  
  
# L, R, DerXdot = -1, 1, 0  
# def f(x): return np.exp(x)  
# def F(x): return np.exp(x)  
# def fd1(x): return np.exp(x)  
# def fd2(x): return np.exp(x)  
# def fd3(x): return np.exp(x)  
# def fd4(x): return np.exp(x)  
# M2deLR, M4deLR = np.exp(R), np.exp(R)  
# M2deLR, M4deLR = 0, 0  
  
# L, R, DerXdot = -1, 1, 1/2  
# def f(x): return np.sqrt(1 - x\*\*2)  
# def F(x): return (1/2)\*x\*np.sqrt(1-x\*\*2)+(1/2)\*np.arcsin(x)  
# def fd1(x): return -x/np.sqrt(1-x\*\*2)  
# def fd2(x): return -x\*\*2/(-x\*\*2+1)\*\*(3/2)-1/np.sqrt(-x\*\*2+1)  
# def fd3(x): return -3\*x\*\*3/(-x\*\*2+1)\*\*(5/2)-3\*x/(-x\*\*2+1)\*\*(3/2)  
# def fd4(x): return -15\*x\*\*4/(-x\*\*2+1)\*\*(7/2)-18\*x\*\*2/(-x\*\*2+1)\*\*(5/2)-3/(-x\*\*2+1)\*\*(3/2)  
# M2deLR, M4deLR = 0, 0  
  
# L, R, DerXdot = 0, np.pi, np.pi  
# BIG\_CONST = np.pi  
# def f(x): return BIG\_CONST \* np.sin(BIG\_CONST \* x)  
# def F(x): return -np.cos(BIG\_CONST \* x)  
# def fd1(x): return BIG\_CONST\*\*2 \* np.cos(BIG\_CONST \* x)  
# def fd2(x): return -BIG\_CONST\*\*3 \* np.sin(BIG\_CONST \* x)  
# def fd3(x): return -BIG\_CONST\*\*4 \* np.cos(BIG\_CONST \* x)  
# def fd4(x): return BIG\_CONST\*\*5 \* np.sin(BIG\_CONST \* x)  
# M2deLR, M4deLR = BIG\_CONST\*\*3, BIG\_CONST\*\*5  
# M2deLR, M4deLR = 0, 0  
  
  
# from sympy import \*  
# from mpmath import \*  
# L, R = 0, 1  
# def f(x): return Float(besselj(0, 100\*x))  
# def F(x): return 0  
# M2deLR, M4deLR = 0, 0  
  
  
  
IntEps = 0.000001  
IntFormatString = "{:.7f}"  
  
DerEps = 0.01  
DerFormatString = "{:.3f}"  
  
  
  
def DerivativeFirst(f, x, d):  
 return (f(x + d) - f(x - d)) / (2 \* d)  
   
def DerivativeFirstViaEstimation(f, x):  
 M2 = abs(fd2(x))  
 df = 2 \* DerEps / M2  
 M3 = abs(fd3(x))  
 ds = (6 \* DerEps / M3) \*\* (1 / 2)  
 return DerivativeFirst(f, x, min(df, ds))  
  
def DerivativeFirstViaTenInMinus5(f, x):  
 d = 10.0 \*\* -5  
 return DerivativeFirst(f, x, d)  
  
  
def DerivativeSecond(f, x, d):  
 return (f(x + d) - 2 \* f(x) + f(x - d)) / (d \*\* 2)  
   
def DerivativeSecondViaEstimation(f, x):  
 M4 = abs(fd4(x))  
 d = (12 \* DerEps / M4) \*\* (1 / 2)  
 return DerivativeSecond(f, x, d)  
   
def DerivativeSecondViaTenInMinus4(f, x):  
 d = 10.0 \*\* -4  
 return DerivativeSecond(f, x, d)  
  
  
print()  
def delta(derappr): return np.ceil(abs(derappr - fd1(DerXdot)) \* (1 / (DerEps / 10))) \* (DerEps / 10)  
print("First Derivative = " + DerFormatString.format( fd1(DerXdot) ))  
print("ViaEstimation = " + DerFormatString.format( DerivativeFirstViaEstimation(f, DerXdot) ), \  
 " | delta = " + DerFormatString.format( delta(DerivativeFirstViaEstimation(f, DerXdot)) ))  
print("ViaTenInMinus5 = " + DerFormatString.format( DerivativeFirstViaTenInMinus5(f, DerXdot) ), \  
 " | delta = " + DerFormatString.format( delta(DerivativeFirstViaTenInMinus5(f, DerXdot)) ))  
print()  
def delta(derappr): return np.ceil(abs(derappr - fd2(DerXdot)) \* (1 / (DerEps / 10))) \* (DerEps / 10)  
print("Second Derivative = " + DerFormatString.format( fd2(DerXdot) ))  
print("ViaEstimation = " + DerFormatString.format( DerivativeSecondViaEstimation(f, DerXdot) ), \  
 " | delta = " + DerFormatString.format( delta(DerivativeSecondViaEstimation(f, DerXdot)) ))  
print("ViaTenInMinus4 = " + DerFormatString.format( DerivativeSecondViaTenInMinus4(f, DerXdot) ), \  
 " | delta = " + DerFormatString.format( delta(DerivativeSecondViaTenInMinus4(f, DerXdot)) ))  
print()  
  
  
  
def IntegralViaMiddleRectangles(f, L, R, N):  
 h = (R - L) / N  
 x = L + h / 2  
 s = 0.0  
 while x < R:  
 s += f(x) \* h  
 x += h  
 return s  
   
def IntegralViaTrapezoids(f, L, R, N):  
 h = (R - L) / N  
 x = L + h / 2  
 s = 0.0  
 while x < R:  
 s += ((f(x - h / 2) + f(x + h / 2)) / 2) \* h  
 x += h  
 return s  
  
def IntegralViaSimpson(f, L, R, N):  
 h = (R - L) / N  
 x = L + h / 2  
 s = 0.0  
 while x < R:  
 fa = f(x - h / 2)  
 fm = f(x)   
 fb = f(x + h / 2)   
 s += (fa + 4 \* fm + fb) \* h / 6  
 x += h  
 return s  
  
  
def IntegralViaRandomSegments(method, f, L, R):  
 LeftCoeff, RightCoeff = 1/3, 1/2  
 h\_prev = R - L  
 ans\_prev = method(f, L, R, 1)  
 while True:  
 h\_new = h\_prev \* (LeftCoeff + (RightCoeff - LeftCoeff) \* np.random.rand())  
 N = math.floor((R - L) / h\_new)  
 M = L + h\_new \* N  
 ans\_new = method(f, L, M, N) + method(f, M, R, 1)  
 if abs(ans\_new - ans\_prev) < IntEps:  
 print("\nRandom intervals: N =",N)  
 return ans\_new  
 ans\_prev = ans\_new  
 h\_prev = h\_new  
  
  
def IntegralViaMiddleRectanglesViaEstimation(f, L, R):  
 if M2deLR > 0.0:  
 M2 = M2deLR  
 h = (24 \* IntEps / (R - L) / M2) \*\* (1 / 2)  
 N = np.ceil((R - L) / h)  
 return IntegralViaMiddleRectangles(f, L, R, N)  
 else:  
 return IntegralViaRandomSegments(IntegralViaMiddleRectangles, f, L, R)  
  
def IntegralViaTrapezoidsViaEstimation(f, L, R):  
 if M2deLR > 0.0:  
 M2 = M2deLR  
 h = (12 \* IntEps / (R - L) / M2) \*\* (1 / 2)  
 N = np.ceil((R - L) / h)  
 return IntegralViaTrapezoids(f, L, R, N)  
 else:  
 return IntegralViaRandomSegments(IntegralViaTrapezoids, f, L, R)  
  
def IntegralViaSimpsonViaEstimation(f, L, R):  
 if M4deLR > 0.0:  
 M4 = M4deLR  
 h = (180 \* IntEps / (R - L) / M4) \*\* (1 / 4)  
 N = np.ceil((R - L) / h)  
 return IntegralViaSimpson(f, L, R, N)  
 else:  
 return IntegralViaRandomSegments(IntegralViaSimpson, f, L, R)  
  
  
print()  
intprecised = F(R) - F(L)  
print("Integral = " + IntFormatString.format( intprecised ))  
def delta(intappr): return np.ceil(abs(intappr - intprecised) \* (1 / (IntEps / 10))) \* (IntEps / 10)  
intappr = IntegralViaMiddleRectanglesViaEstimation(f, L, R)  
print(("ViaMiddleRectangles = " + IntFormatString + " | delta = " + IntFormatString).format( intappr , delta(intappr) ))  
intappr = IntegralViaTrapezoidsViaEstimation(f, L, R)  
print(("ViaTrapezoids = " + IntFormatString + " | delta = " + IntFormatString).format( intappr , delta(intappr) ))  
intappr = IntegralViaSimpsonViaEstimation(f, L, R)  
print(("ViaSimpson = " + IntFormatString + " | delta = " + IntFormatString).format( intappr , delta(intappr) ))  
print()

Найти численное значение первой и второй производной в точке. Найти численное значение интеграла по формулам прямоугольников, трапеций, Симпсона.

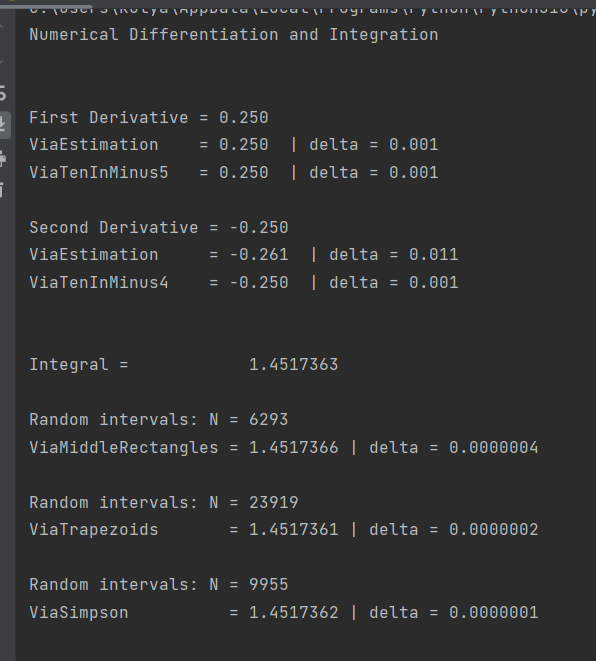
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Тестовый пример 1   |  |  |  | | --- | --- | --- | | *Функция* | *Точка* | *Интервал* | |  |  |  | | *Значение* | *Округл.* | *Приближ.* | |  | 1.000 | 1.000 | |  | 1.000 | 1.010 | | Метод оценки погрешностей | | | | *Вид* | *Округл.* |  | | *точный* | 2.3504024 | – | | *ср. прям.* | 2.3504020 | 0.0000005 | | *трапец.* | 2.3504028 | 0.0000005 | | *Симпсон* | 2.3504024 | 0.0000001 | | Метод остаточного отрезка | | | | *Вид* | *Округл.* |  | | *точный* | 2.3504024 | – | | *ср. прям.* | 2.3504021 | 0.0000003 | | *трапец.* | 2.3504025 | 0.0000001 | | *Симпсон* | 2.3504024 | 0.0000001 | | Тестовый пример 2   |  |  |  | | --- | --- | --- | | *Функция* | *Точка* | *Интервал* | |  |  |  | | *Значение* | *Округл.* | *Приближ.* | |  | –0.577 | –0.577 | |  | –1.540 | –1.550 | | Метод оценки погрешностей | | | | *Вид* | *Округл.* |  | | *точный* | 1.5707963 | – | | *ср. прям.* | – | – | | *трапец.* | – | – | | *Симпсон* | – | – | | Метод остаточного отрезка | | | | *Вид* | *Округл.* |  | | *точный* | 1.5707963 | – | | *ср. прям.* | 1.5707965 | 0.0000002 | | *трапец.* | 1.5707963 | 0.0000001 | | *Симпсон* | 1.5707962 | 0.0000002 | |
| Тестовый пример 3   |  |  |  | | --- | --- | --- | | *Функция* | *Точка* | *Интервал* | |  |  |  | | *Значение* | *Округл.* | *Приближ.* | |  | 0.540 | 0.540 | |  | –0.841 | –0.832 | | Метод оценки погрешностей | | | | *Вид* | *Округл.* |  | | *точный* | 2.0000000 | – | | *ср. прям.* | 2.0000006 | 0.0000007 | | *трапец.* | 1.9999994 | 0.0000007 | | *Симпсон* | 2.0000000 | 0.0000001 | | Метод остаточного отрезка | | | | *Вид* | *Округл.* |  | | *точный* | 2.0000000 | – | | *ср. прям.* | 2.0000001 | 0.0000002 | | *трапец.* | 1.9999999 | 0.0000002 | | *Симпсон* | 2.0000000 | 0.0000001 | | Тестовый пример 4   |  |  |  | | --- | --- | --- | | *Функция* | *Точка* | *Интервал* | |  |  |  | | *Значение* | *Округл.* | *Приближ.* | |  | –8.909 | –8.909 | |  | 13.342 | 13.332 | | Метод оценки погрешностей | | | | *Вид* | *Округл.* |  | | *точный* | 1.9026854 | – | | *ср. прям.* | 1.9026856 | 0.0000002 | | *трапец.* | 1.9026852 | 0.0000002 | | *Симпсон* | 1.9026854 | 0.0000001 | | Метод остаточного отрезка | | | | *Вид* | *Округл.* |  | | *точный* | 1.9026854 | – | | *ср. прям.* | 1.9026856 | 0.0000002 | | *трапец.* | 1.9026852 | 0.0000002 | | *Симпсон* | 1.9026854 | 0.0000001 | |

Вариант 7

В каждом варианте найти численное значение первой и второй производной в точке, являющейся серединой заданного интервала, с точностью до 0,01. Вычислить с точностью 0,000001 интегралы по формулам прямоугольников, трапеций, Симпсона. Сравнить методы по точности.

Ответ:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Функция | | Интервал | | Производная в точке | |
|  | |  | |  | |
| Значение производных | | | | | |
| Первой | | | Второй | | |
| Округлённое | Приближённое | | Округлённое | | Приближённое |
| ≈ 0.250 | ≈ 0.250 | | ≈ – 0.250 | | ≈ – 0.261 |
| Значение интеграла | | | | | |
| Точное в точке x | | | Округлённое на интервале [0, 2] | | |
|  | | | ≈ 1.4517363 | | |
| Приближённое по методу остаточного отрезка через метод | | | | | |
| Средних | | Трапеций | | Симпсона | |
| ≈ 1.4517366 | | ≈ 1.4517361 | | ≈ 1.4517362 | |
| |Δ| < 0.0000004 | | |Δ| < 0.0000002 | | |Δ| < 0.0000001 | |
| N = 6293 | | N = 23919 | | N = 9955 | |

****

# **Выводы:**

Таким образом, в ходе выполнения лабораторной работы были изучены и сравнены по трудоёмкости и точности методы численного вычисления производных и методы численного интегрирования. Составлена компьютерная программа, на тестовых примерах проверена правильность её работы. Для функции заданного варианта найдено численное значение первой и второй производной в точке, вычислены с заданной точностью интегралы по формулам прямоугольников, трапеций, Симпсона.