|  |  |
| --- | --- |
| |  | | --- | | МИНОБРНАУКИ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | |

Институт Информационных технологий

Кафедра МОСИТ

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5**

по дисциплине «Теория принятия решений»

**Студент группы** ИКБО-11-17 Алиев Ю. А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись студента)*

**Руководитель работы** Петренко А. А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись руководителя)*

Москва, 2020

# Постановка задачи

Найти: *Argmin* *f*(*x*), *x* ∈ [-5; 5] при Δ*x* ≤ 0.1 перечисленными ниже методами в соответствии с вариантом.

а) методом средней точки;

б) методом хорд;

в) методом Ньютона;

г) методом ломаных.

Вариант 1: *f*(*x*) = 2x2 - 5

# Метод параболы

Для решения данным методом была написана следующая программа на языке программирования JavaScript:

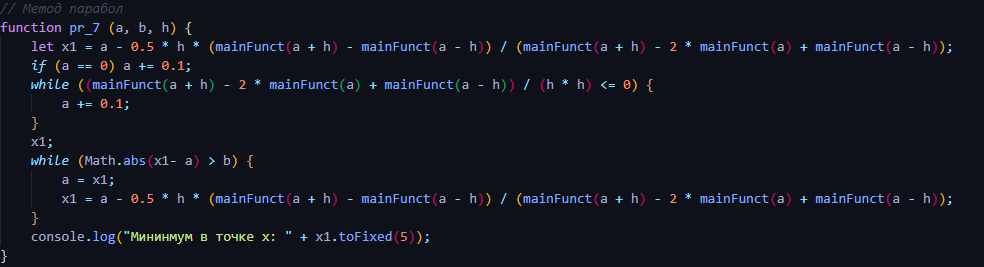


Рисунок 1. Метод параболы

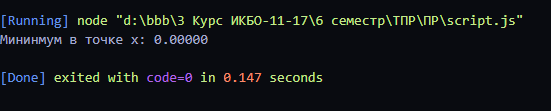


Рисунок 2. Ответ, найденный методом параболы

# Метод средней точки

Для решения данным методом была написана следующая программа на языке программирования JavaScript:

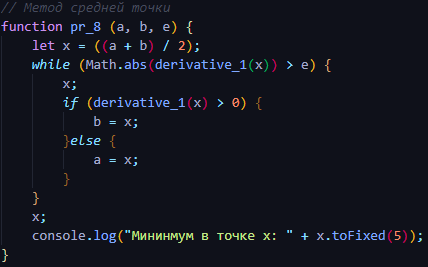


Рисунок 3. Метод средней точки

Алгоритм написан на основе лекций.

Для варианта №1 алгоритм выдает следующий ответ.

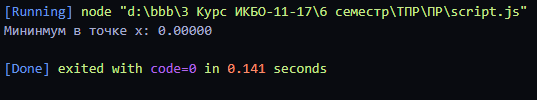


Рисунок 4. Ответ, найденный методом средней точки

Ответ совпадает с решением из предыдущих номеров.

# Метод хорд

Метод золотого сечения также описан на языке программирования JavaScript.

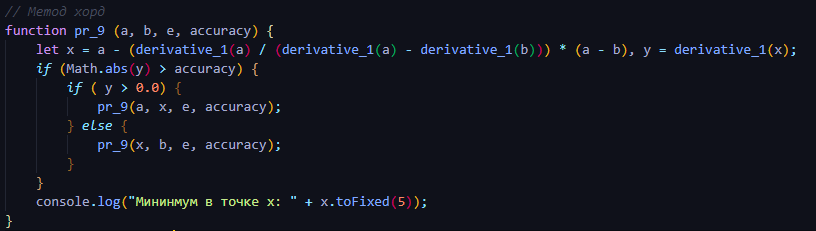


Рисунок 5. Алгоритм метода хорд

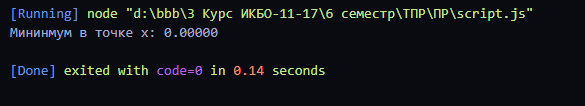
 **\***

Рисунок 6. Ответ, найденный методом хорд

Ответ совпадает с решением из предыдущих номеров.

# Метод Ньютона

Для данного метода также был написан алгоритм на языке JavaScript.

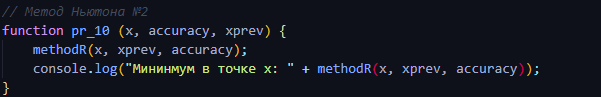


Рисунок 7. Алгоритм метода Ньютона

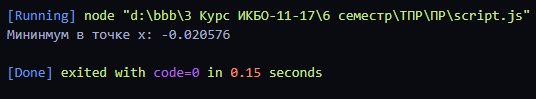


Рисунок 8. Ответ, найденный методом Ньютона

# Метод ломаных

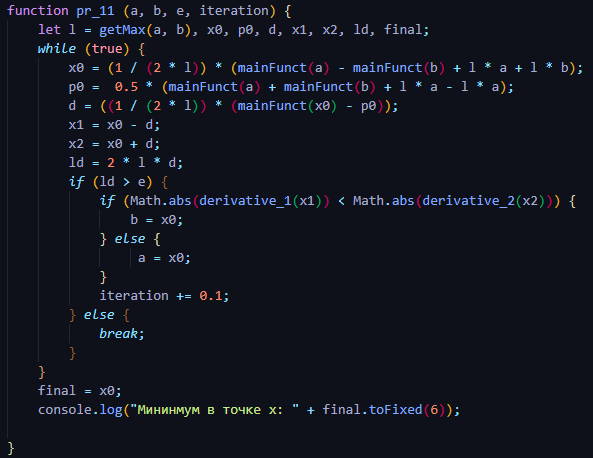


Рисунок 9. Алгоритм метода ломаных

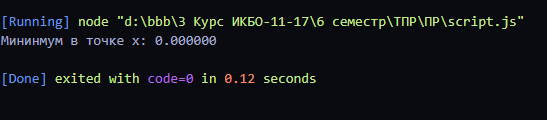


Рисунок 10. Ответ, найденный методом ломаных

# Приложение

Исходный код программы:

|  |
| --- |
| // Основная функция  function mainFunct (x) {  return (2 \* (x\*\*2)) - 5;  }  // Первая производная  function derivative\_1 (x) {  return 4 \* x;  }  // Вторая производная  function derivative\_2 (x) {  return 6;  }  // Числа Фибоначчи  function fib (i) {  if (i == 1) return 1;  if (i == 2) return 1;  return fib(i - 1) + fib(i - 2);  }  // Метод Ньютона №1  function methodR (x, prev, accuracy) {  if (Math.abs(x - prev) > accuracy) {  return methodR(x - derivative\_1(x) / derivative\_2(x), x, accuracy);  }else {  return x;  }  }  // Поиск максимума на концах отрезка  function getMax (a, b) {  if (Math.abs(derivative\_1(a)) > Math.abs(derivative\_1(b))) {  return Math.abs(derivative\_1(a));  } else {  return Math.abs(derivative\_1(b));  }  }  // Метод полного перебора  function pr\_2 () {  let x = -5, min = 999, argmin = 0, y;  while (x <= 5) {  y = mainFunct(x);  // console.log("x = " + x.toFixed(1) + " y = " + y.toFixed(2));  x += 0.001;  if (y <= min){  min = y;  argmin = x;  }  }  console.log("Мининмум в точке x: " + argmin.toFixed(4));  }  // Метод поразрядного поиска;  function pr\_3 () {  let x = -5, min = 999, argmin = 0, y;  while (x <= 5) {  y = mainFunct(x);  // console.log("x = " + x.toFixed(1) + " y = " + y.toFixed(2));  x += 0.3;  if (y <= min){  min = y;  argmin = x;  }  }  console.log("Мининмум в точке x: " + argmin.toFixed(4));  }  // Метод дихотомии;  function pr\_4 (a, b, delta, sigma) {  let result = 0, sr, x1, x2;  while (Math.abs(b - a) >= delta) {  sr = (a + b) / 2;  x1 = sr - sigma;  x2 = sr + sigma;  if (mainFunct(x1) <= mainFunct(x2)) {  b = x2;  } else {  a = x1;  }  result = (b + a) / 2;  }  console.log("Мининмум в точке x: " + result.toFixed(2));  }  // Метод золотого сечения;  function pr\_5 (a, b, e) {  let phi = (1 + Math.sqrt(5)) / 2, x1, x2;  do {  x1 = b - (b - a) / phi;  x2 = a + (b - a) / phi;  if (mainFunct(x1) >= mainFunct(x2)) {  a = x1;  } else {  b = x2;  }  } while (!(Math.abs(b - a) < e));  console.log("Мининмум в точке x: " + ((a + b) / 2));  console.log("Мининмум в точке x: " + ((a + b) / 2).toFixed(5));  }  // Метод Фибоначчи;  function pr\_6 (a, b, e) {  let iterationCount = ((b - a) / e), s = 0, x1, x2, l,  fibList = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946];  for (index = 0; index < fibList.length; ++index) {  if (iterationCount >= fibList[index]) {  s = index;  }  }  do {  l = (b - a) / fib(s + 2);  x1 = a + l \* fib(s);  x2 = b - l \* fib(s);  if (mainFunct(x1) < mainFunct(x2)) {  b = x2;  } else {  a = x1;  }  } while (!(Math.abs(b - a) < e));  console.log("Мининмум в точке x: " + ((a + b) / 2));  console.log("Мининмум в точке x: " + ((a + b) / 2).toFixed(5));  }  // Метод парабол  function pr\_7 (a, b, h) {  let x1 = a - 0.5 \* h \* (mainFunct(a + h) - mainFunct(a - h)) / (mainFunct(a + h) - 2 \* mainFunct(a) + mainFunct(a - h));  if (a == 0) a += 0.1;  while ((mainFunct(a + h) - 2 \* mainFunct(a) + mainFunct(a - h)) / (h \* h) <= 0) {  a += 0.1;  }  x1;  while (Math.abs(x1- a) > b) {  a = x1;  x1 = a - 0.5 \* h \* (mainFunct(a + h) - mainFunct(a - h)) / (mainFunct(a + h) - 2 \* mainFunct(a) + mainFunct(a - h));  }  console.log("Мининмум в точке x: " + x1.toFixed(5));  }  // Метод средней точки  function pr\_8 (a, b, e) {  let x = ((a + b) / 2);  while (Math.abs(derivative\_1(x)) > e) {  x;  if (derivative\_1(x) > 0) {  b = x;  }else {  a = x;  }  }  x;  console.log("Мининмум в точке x: " + x.toFixed(5));  }  // Метод хорд  function pr\_9 (a, b, e, accuracy) {  let x = a - (derivative\_1(a) / (derivative\_1(a) - derivative\_1(b))) \* (a - b), y = derivative\_1(x);  if (Math.abs(y) > accuracy) {  if ( y > 0.0) {  pr\_9(a, x, e, accuracy);  } else {  pr\_9(x, b, e, accuracy);  }  }  console.log("Мининмум в точке x: " + x.toFixed(5));  }  // Метод Ньютона №2  function pr\_10 (x, accuracy, xprev) {  methodR(x, xprev, accuracy);  console.log("Мининмум в точке x: " + methodR(x, xprev, accuracy).toFixed(6));  }  function pr\_11 (a, b, e, iteration) {  let l = getMax(a, b), x0, p0, d, x1, x2, ld, final;  while (true) {  x0 = (1 / (2 \* l)) \* (mainFunct(a) - mainFunct(b) + l \* a + l \* b);  p0 = 0.5 \* (mainFunct(a) + mainFunct(b) + l \* a - l \* a);  d = ((1 / (2 \* l)) \* (mainFunct(x0) - p0));  x1 = x0 - d;  x2 = x0 + d;  ld = 2 \* l \* d;  if (ld > e) {  if (Math.abs(derivative\_1(x1)) < Math.abs(derivative\_2(x2))) {  b = x0;  } else {  a = x0;  }  iteration += 0.1;  } else {  break;  }  }  final = x0;  console.log("Мининмум в точке x: " + final.toFixed(6));    }  // Вызов всех функций  function init () {  pr\_2();  pr\_3();  pr\_4(-5, 5, 0.1, 0.04);  pr\_5(-5, 5, 0.001);  pr\_6(-5, 5, 0.001);  pr\_7(0, 1, 0.001);  pr\_8(-5, 5, 0.001);  pr\_9(-5, 5, 0.1, 0.001);  pr\_10(-5, 0.1, 0.1);  pr\_11(-5, 5, 0.1, 1);  }  init(); |

Листинг 1. Исходный код файла pr1

# Функции из вариантов задания

def function(x):

    y=2\*x\*x + 6\* x

    return y

# Производная от функции варианта задания

def function\_d(x):

    y=4\*x +6

    return y

# Производная второго порядка от функции варианта задания

def function\_dd(x):

    y=4

    return y

def BrokenLine(a, b, e0):

    iteration = 1

    L = get\_max(a, b)

    while True:

        x0 = (1/(2\*L))\*(function(a)-function(b)+L\*a+L\*b)

        p0 = 0.5\*(function(a)+function(b) + L\*a - L\*b)

        d = ((1/(2\*L))\*(function(x0)-p0))

        x1 = x0 - d

        x2 = x0 + d

        ld=2 \* L \* d

        if ld > e0:

            if abs(function\_d(x1)) < abs(function\_d(x2)):

                b=x0

                x0=x1

            else:

                a=x0

                x0=x2

            iteration += 1

        else:

            break

    final\_x=x0

    return final\_x

# Определить максимальное значение функции в окраинных точках

def get\_max(a, b):

    if abs(function\_d(a)) > abs(function\_d(b)):

        return abs(function\_d(a))

    else:

        return abs(function\_d(b))

print("Ответ для метода ломаных", BrokenLine(-5,5,0.1))

Листинг 4. Алгоритм ломаных