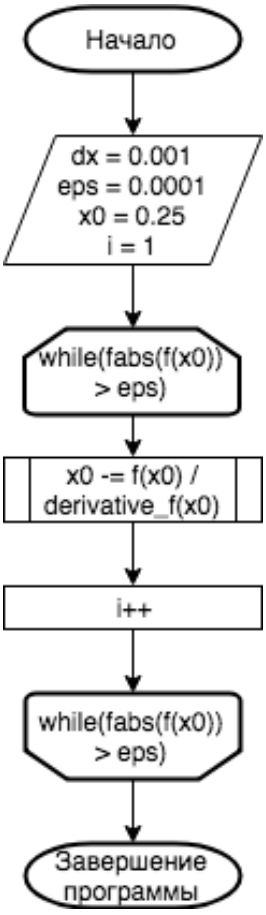
Задание 1

Найти корень уравнения f(x) = 0 методом Ньютона с определением производной функции методом конечных разностей.

Листинг программы

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
const double dx = 0.001, eps = 0.0001; //константы из условия задачи
double f(double x) {return pow(1.5, x) * pow(x - 0.5, 2) - 3;} //функция вычисляющая f(x)
double derivative_f(double x) {return (f(x + dx) - f(x - dx)) / (2 * dx);} //функция вычисляющая f'(x)
int main(){
	cout << "Решение уравнения f(x)=1.5^x(x-0.5)^2-3 методом Ньютона" << endl;
	cout.setf(ios_base::fixed); //фиксированное колличество знаков после запятой, равное...
	cout.precision(5); //5
	double x0 = 0.25; //начальное приближение будет изменяться по ходу рассчета, поэтому не константа
	int i = 1; //номер шага
	cout << setw(2) << "#"  << " |" << setw(10) << "f(x0)" << " |" << setw(10) << "x0" << endl; //выводим
"загаловок" таблицы, с помощью setw() размер поля для вывода
	while(fabs(f(x0)) > eps){ //пока абсолютное значение f(x0) больше эпсилон
		x0 -= f(x0) / derivative_f(x0); //вычисляем x0
		cout << setw(2) << i << " |" << setw(10) << f(x0)  << " |" << setw(10) << x0 << endl; //выводим номер
шага, f(x0) и x0
		i++;  //увеличиваем номер шага
	}
	cout << x0 << " – корень уравнения f(x)=1.5^x*(x-0.5)^2-3" << endl; //последние значение x0 и есть корень
уравнения
	return 0;
}
```

Блок-схема программы



Полученные результаты и их анализ

График функции:

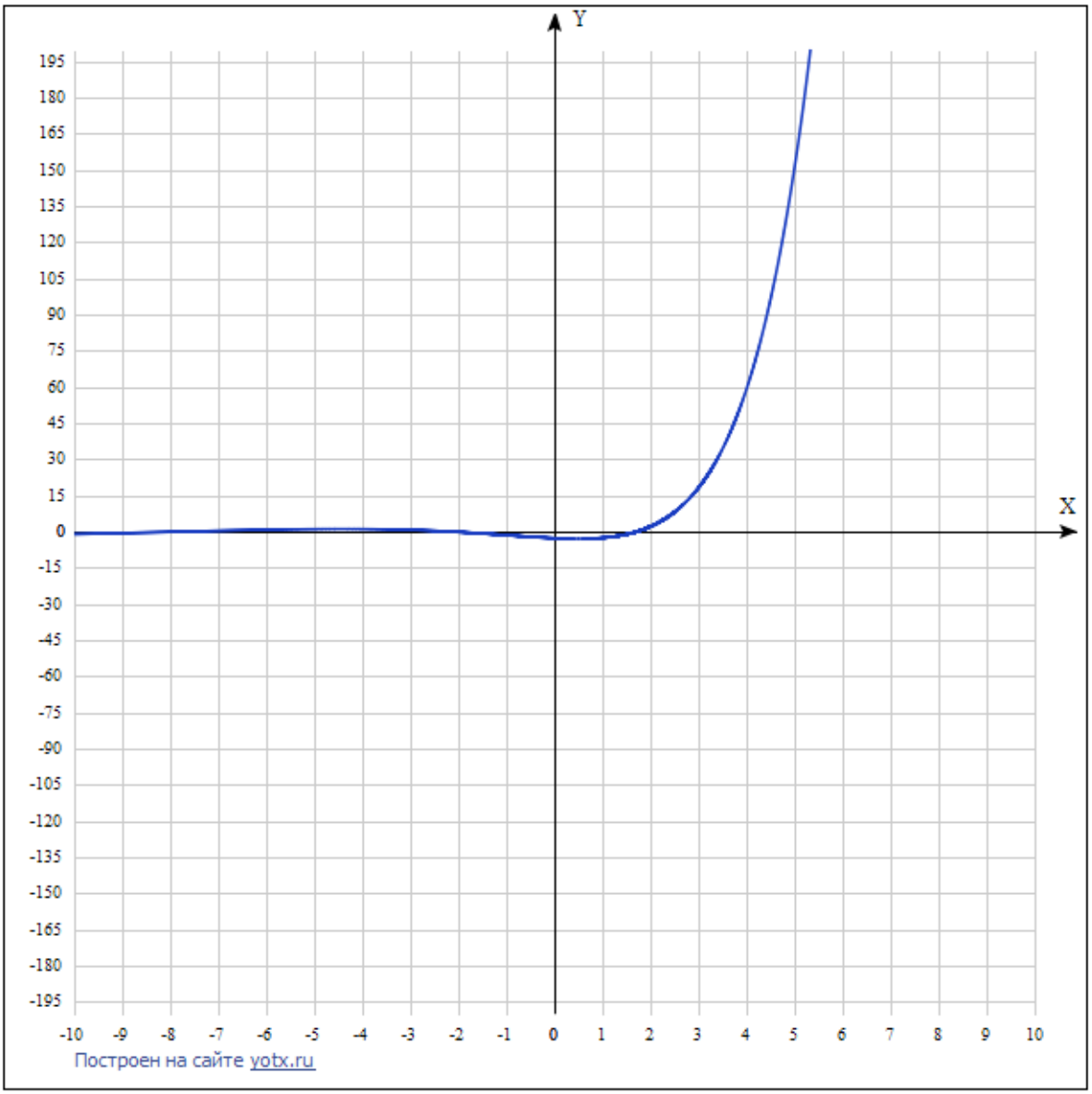
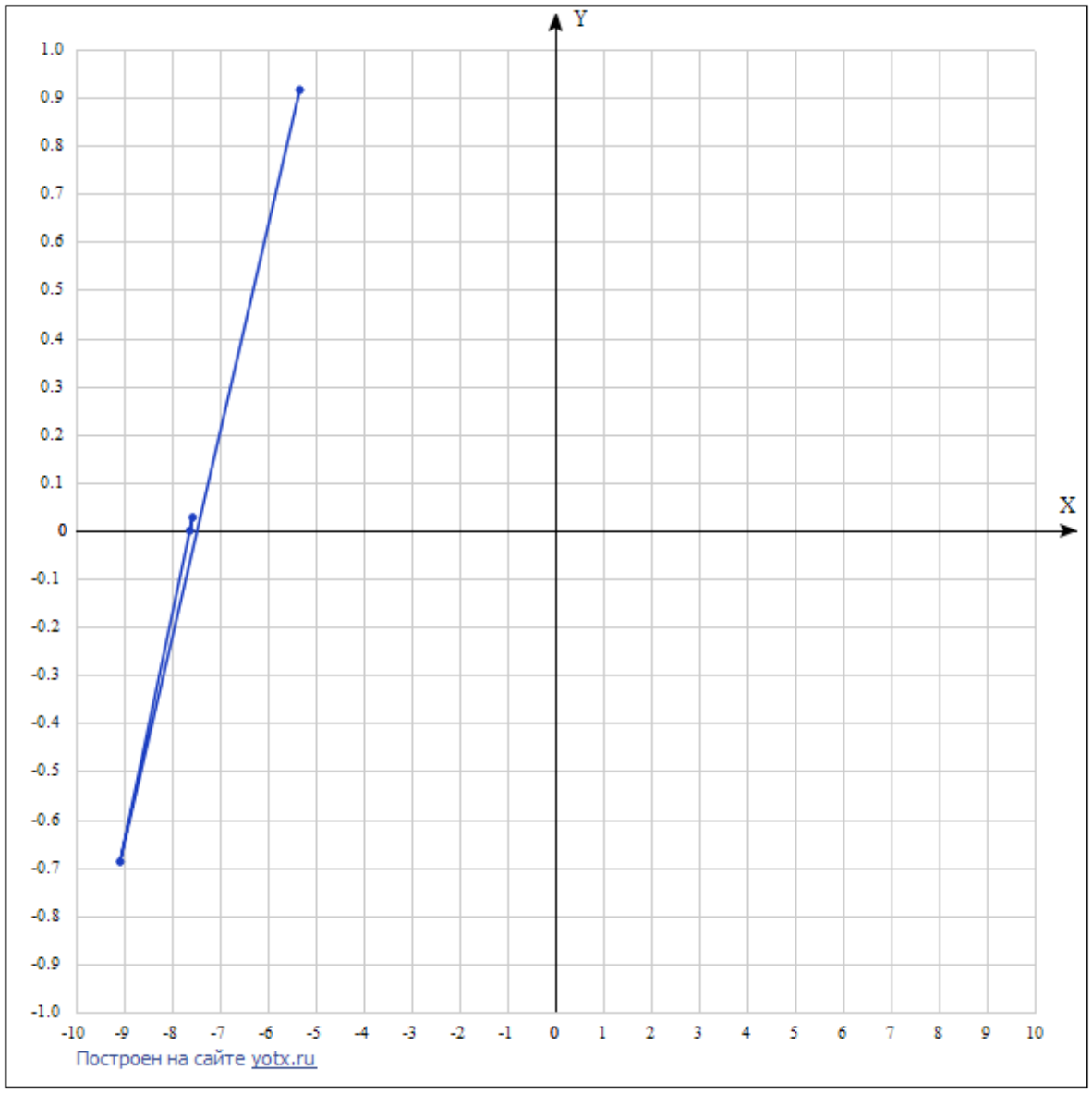


График решения:



Метод Ньютона основан на переборе и не позволяет найти корень уравнения со 100% точностью. Она зависит от шага (dx) и от точности (eps). При уменьшении шага и точности увеличивается близость f(x0) к 0, но при этом также увеличивается количество итераций, а следовательно и время расчета.