Федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский государственный университет

информационных технологий, механики и оптики

Факультет прикладной информатики и компьютерных технологий

ITMO UNIVERSITY

# Вычислительная Математика

## Лабораторная работа №3

### Интерполяция

### Метод Ньютона


Группа: P3218

Студент: Петкевич Константин Вячеславович

Преподаватель: Кучер Алексей Владимирович


г. Санкт-Петербург

2016 г.

## Метод Ньютона

В лабораторной работе, я генерировал узлы интерполяции случанйым образом. Поэтому шаг имеет переменной значение и вместо конечных разностей мы будем использыват раздельную .

Разделенная разность — обобщение понятия производной для дискретного набора точек. Разделенная разность нулевого порядка функции f(x) — сама функция f(x). Разделённая разность порядка n определяется через разделённую разность порядка n− 1 по формуле

$$f(x_0;\ x_1;\ \ldots;\ x_n) = \frac{f(x_1;\ \ldots;\ x_n) - f(x_0;\ \ldots;\ x_{n-1})}{x_n - x_0}.$$

Формулу интерполяционного полинома в форме Ньютона можно записать для любого способа упорядочивания узлов. Так, например, при упорядочивании узлов в обратном порядке ($x_n$, $x_{n-1}$, $x_{n-2}$, …, $x_2$, $x_0$) примет вид

$$P_n(x) = f(x_n) + f(x_n, x_{n-1})\times(x - x_n) + f(x_n, x_{n-1}, x_{n-2})\times(x - x_n)(x - x_{n-1}) + \ldots$$
$$\ldots + f(x_n, x_{n-1}, x_{n-2}, \ldots x_1, x_0)\times(x - x_n)(x - x_{n-1})\ldots(x - x_2)(x - x_1).$$

Такой интерполяционный полином называют интерполяционным полиномом для интерполирования назад.

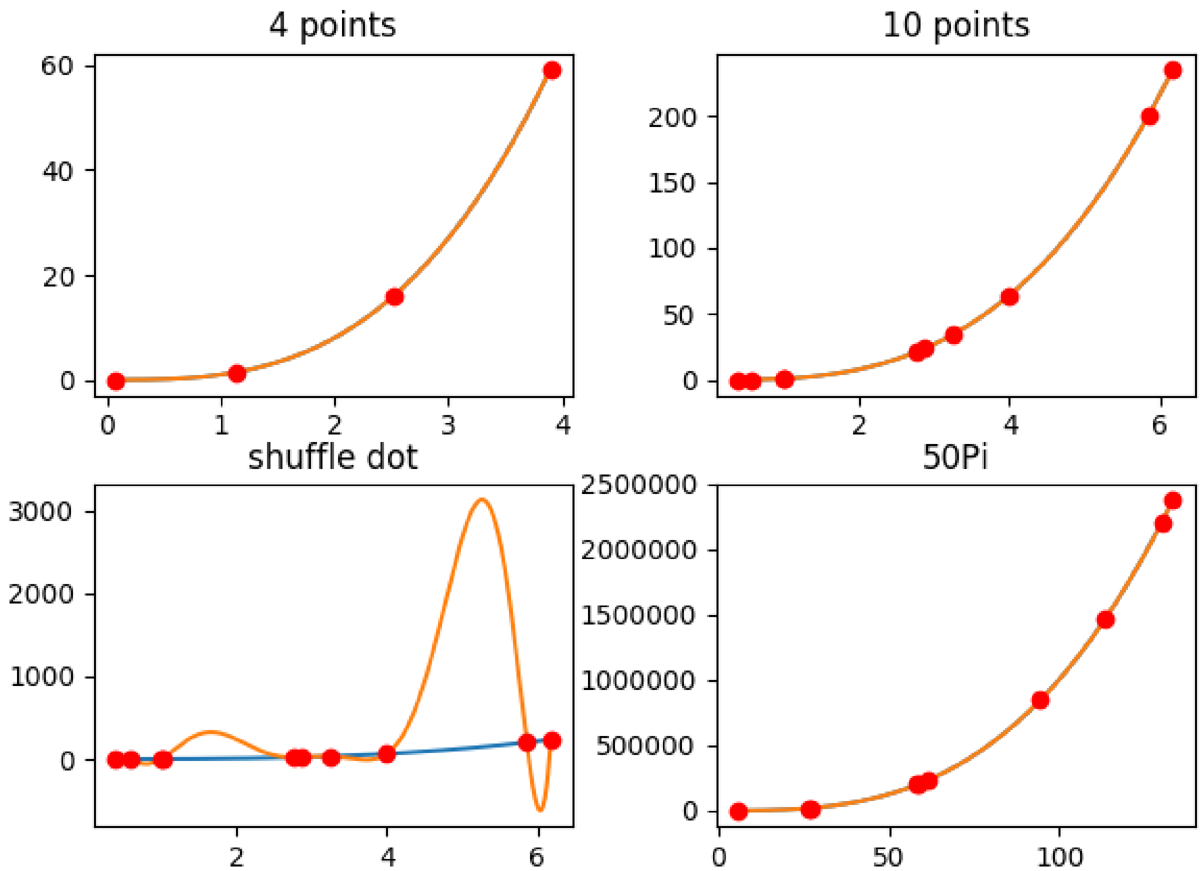
## Листинг программы

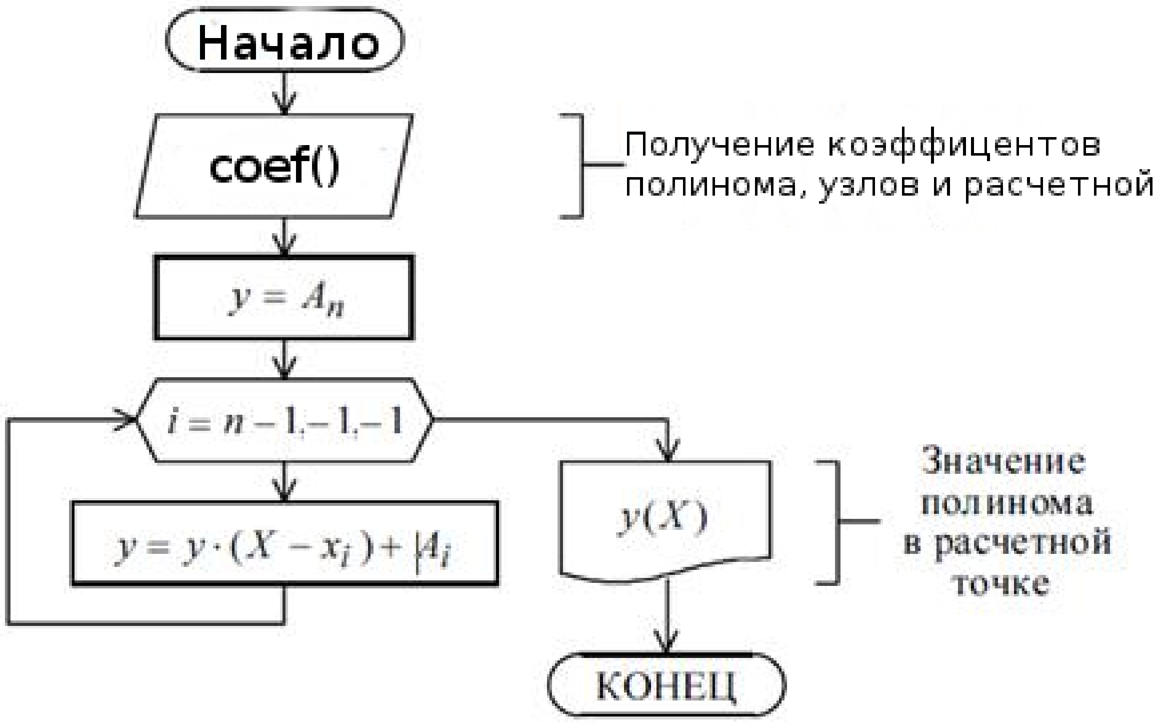
```
#Divided differences
def coef(points):
    '''x : array of data points
      y : array of f(x)  '''
    n = len(points)
    a = []
    for i in range(n):
        a.append(points[i][1])
    for j in range(1, n):
        for i in range(n-1, j-1, -1):
            a[i] = float(a[i]-a[i-1])/float(points[i][0]-points[i-j][0])
    return np.array(a) # return an array of divided differences
```

```
def eval(a, points, r):
    '''     a : array returned by function coef()
       points : array of data points
            r : the node to interpolate at  '''
    n = len( a ) - 1
    temp = a[n]
    for i in range( n - 1, -1, -1 ):
        temp = temp * ( r - points[i][0] ) + a[i]
    return temp # return the y_value interpolation
```

## Пример работы программы

Функция (x*x*x)






**Вывод:** интерполирование сильно зависит от количества и места расположения точек на интерполируемом отрезке функции. Одна точка с сильно измененным значением сводит на нет практический результат интерполяции. Использование интерполяционного полинома в форме Ньютона оказывается удобным, например, когда появляются дополнительные узлы. В формулах Нютона в случае добавления узла все найденные члены сохраняются и появляется новое слагаемое, представляющее собой ни что иное, как поправку к уже вычисленному значению. В формуле Лагранжа же добавление узла повлечет за собой не только появление нового слагаемого, но и потребует исправления ранее найденных членов суммы.