Лабораторная работа №4
Ильин Денис, z3243

1) Измените метод `__call__`, так чтобы он реализовывал схему Горнера. Чем эта схема лучше?

   Используя запись полинома n-ой степени в виде мономов требует максимум n операций сложения и (n^2+n)/2 операций умножения, если степени считаются путем последовательного перемножения и каждый моном оценивается индивидуально. Это может быть сведено к n операциям сложения и 2n-1 операции умножения, если считать степени x итеративно. Однако схема горенка требует только n операций сложения и умножения, что сильно понижает погрешность вычислений.

```
p = 0.
for pn in reversed(self.pn):
    p *= x
    p+= pn
return p
```

2) Почему нахождение коэффициентов интерполяционного многочлена через решение системы дает ошибочный ответ?

   Есть какая то погрешность начальных данных, и когда мы домножаем эту погрешность на обратную матрицу, в которой очень большие числа, потому что определитель почти 0, погрешность возрастает многократно. Система плохо обусловленна, особенно если брать логарифмическую решетку в качестве узлов. Многочлены почти линейно зависимы, из-за чего минимальное собственное число очень близко к 0, поэтому число обусловленности очень велико)

3) Найдите определитель матрицы Вандермонда теоретически и численно.

```
theor_det = 1
for i in range(len(x)):
     for j in range(i):
          theor_det *= x[i] - x[j]
print("Numerical determinant", np.linalg.det(vandermonde(x)))
print("Theoretical determinant", theor_det)

('Numerical determinant', 3.0764191717848722e-49)
('Theoretical determinant', 3.0749894740093671e-49)
```

4) Найдите числа обусловленности матрицы Вандермонда.

   ('Condition number', 1.3566225413987645e+17)

5) Реализуйте метод Эйткена вычисления интерполяционного многочлена.

```
def ayytken(xn, yn, i=0, j=N-1):
    if i == j:
        z = np.zeros(N)
        z[0] = yn[i]
        return z
    ayy1 = ayytken(xn, yn, i + 1, j)
    ayy2 = ayytken(xn, yn, i, j - 1)
    return 1 / (xn[j] - xn[i]) * ((np.roll(ayy1,1) - ayy1 * xn[i]) -
(np.roll(ayy2,1) - ayy2 * xn[j]))
```

6) Если мы попытаемся восстановить многочлен через его значения в точках, аналогично заданию 2, получим ли мы с помощью метода Эйткена ответ точнее, чем через решение системы?

Метод Эйткена на два-три порядка хуже, чем решение системы, потому что задача все равно плохо обусловленна при таком выборе узлов.

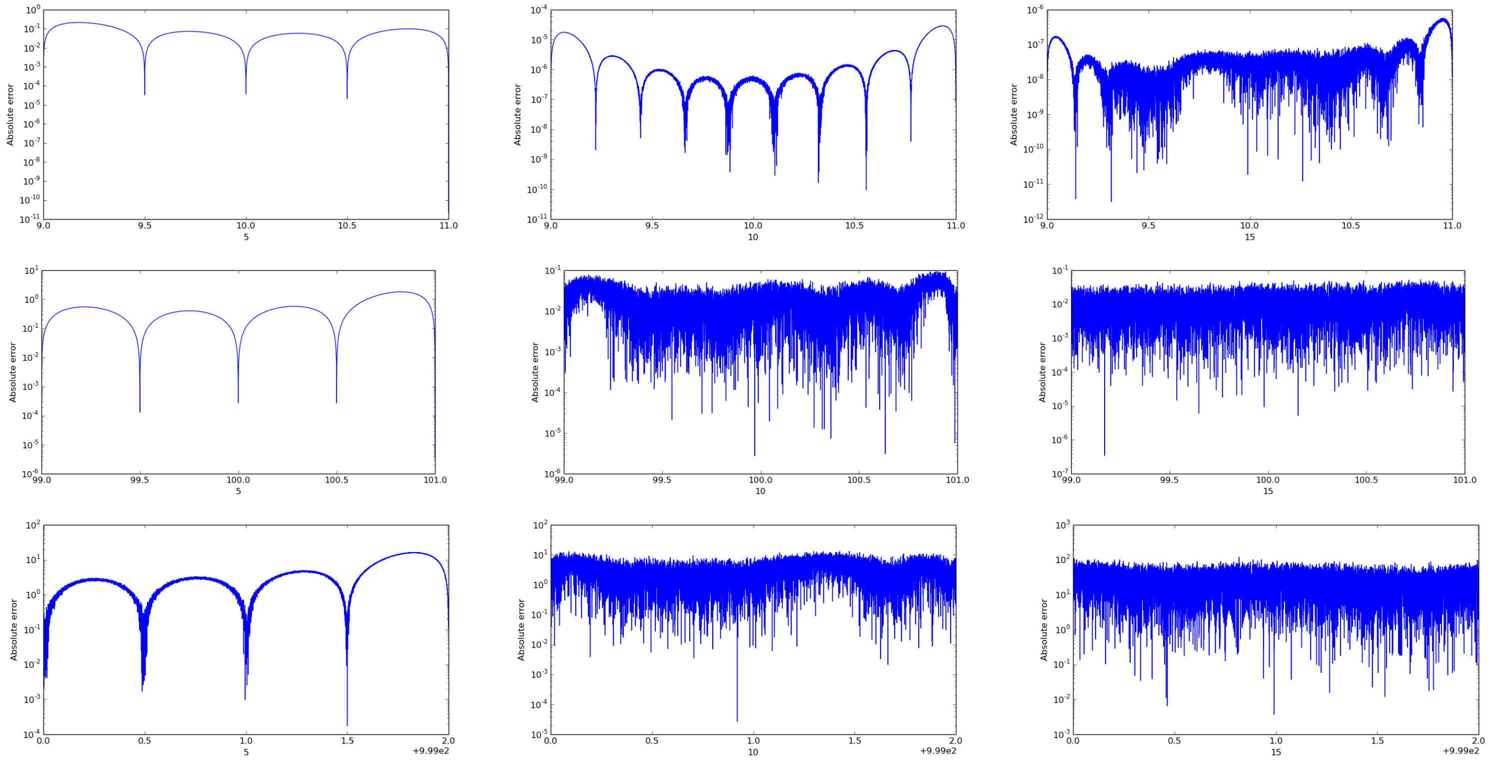
7) Scipy содержит готовую реализацию интерполяционного многочлена Лагранжа [scipy.interpolate.lagrange](). В документации отмечается, что метод численно неустойчив. Что это означает?

Численная неустойчивость - это высокая чувствительность к возмущениям в исходных данных.

8) Ошибки в исходных данных для построения интерполяционного многочлена вызывают ошибки при вычислении интерполяционного многочлена в промежуточных точках. При каком расположении узлов интерполяция многочленом Лагранжа имеет наименьшую ошибку? Как это связано с численной устойчивостью?

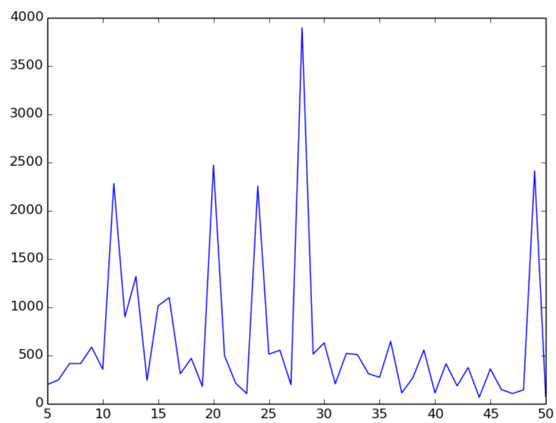
Наилучшее расположение узлов - нули многочлена Чебышёва, так как для них есть наилучшая оценка константы Лебега. От выбора узлов зависит обусловленность задачи, это в свою очередь связано с устойчивостью, чем лучше обусловлена задача, тем более устойчивым будет для нее метод.

9) Найдите погрешность приближения функции f интерполяционным многолченом p для x0=10,100,1000 и N=5,10,15. Объясните получающиеся результаты.

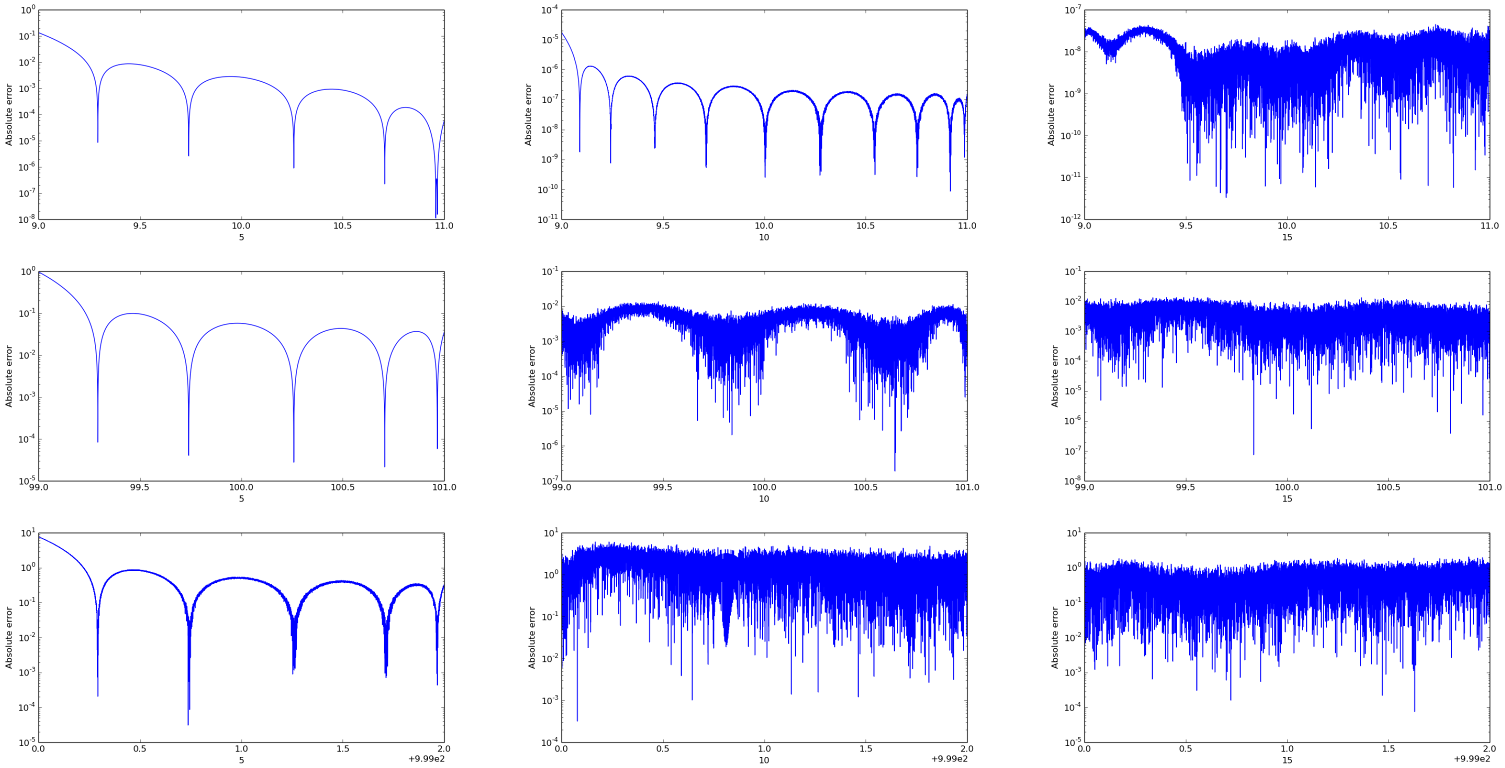


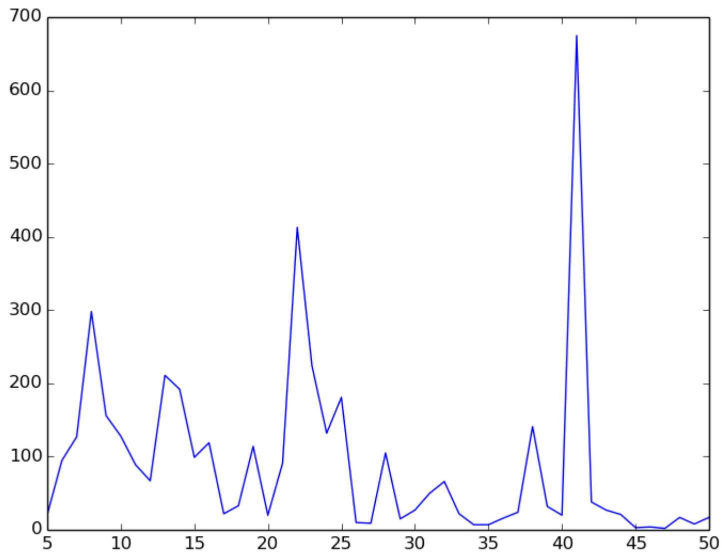
С увеличением N уменьшается погрешность от интерполяции, но число обусловленности задачи растет. При малых x0 мы наблюдаем первое, при большом - второе.

10) Постройте график зависимости ошибки от числа узлов интерполяции для N от 5 до 50.



11) Повторите задания 9 и 10 для узлов интерполяции Чебышева

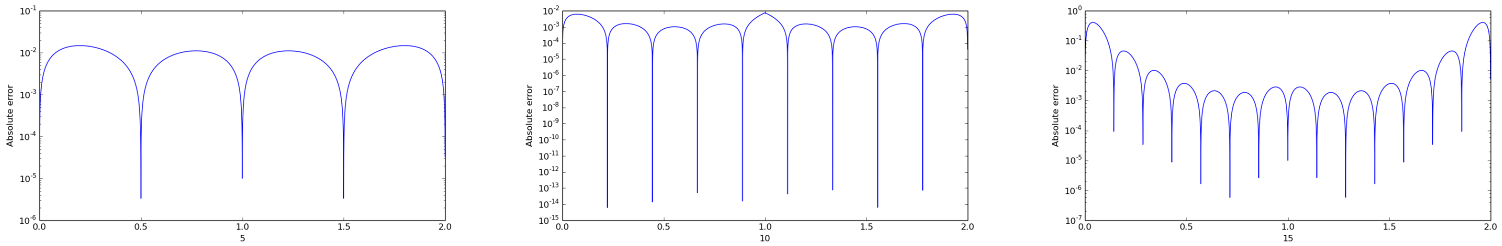


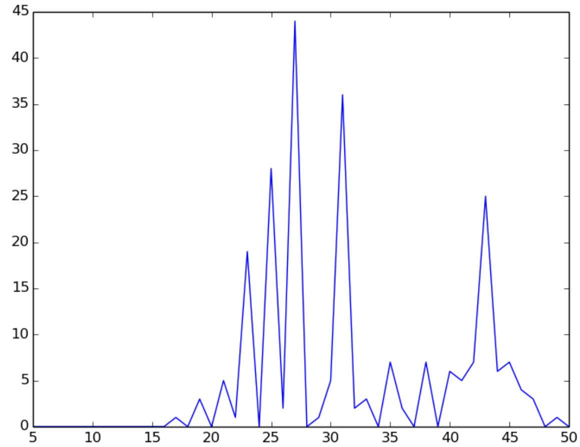


12) Сравните распределение ошибки внутри интервала для равномерно расположенных узлов и для узлов Чебышева.

Многочлены с узлами Чебышёва интерполируют исходный многочлен лучше, чем при равномерно расположенных узлах.

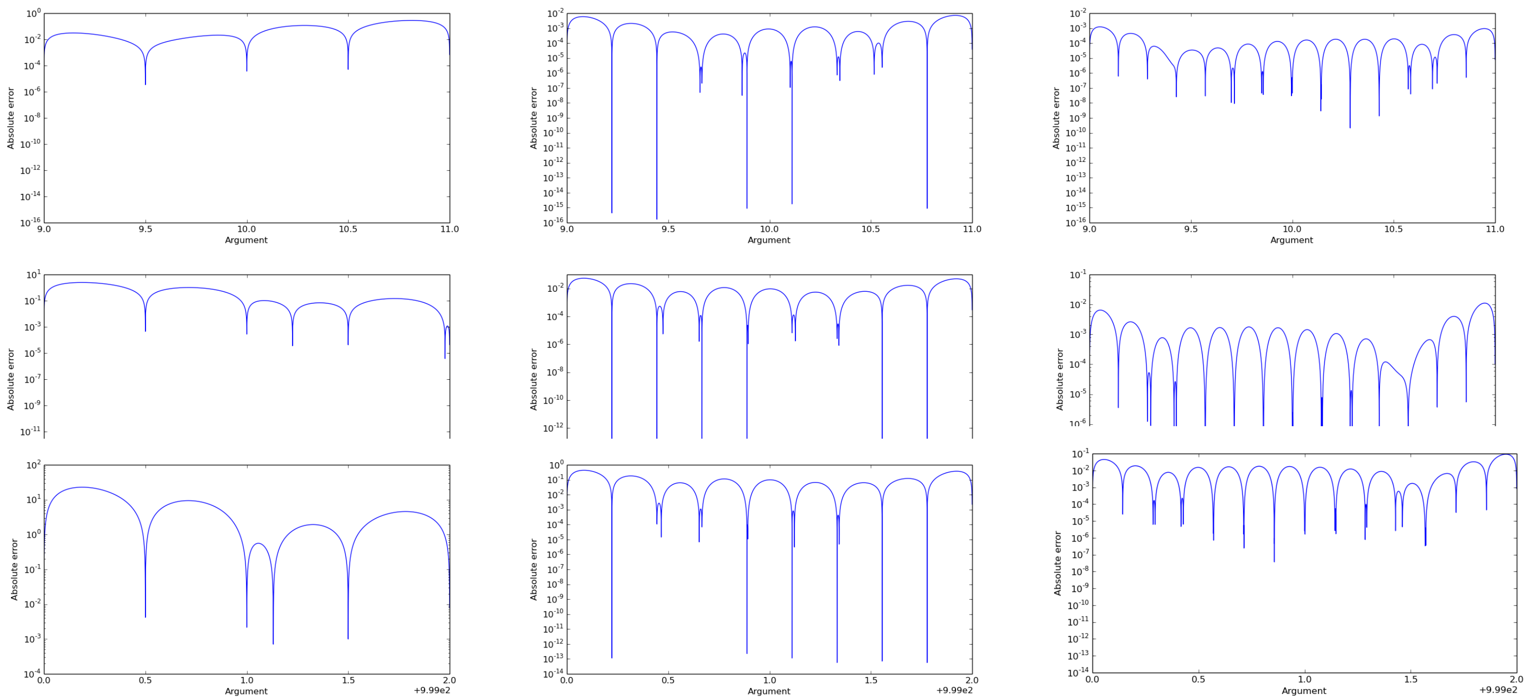
13) Повторите задания 9 и 10 для функции f=|x-1|, r=1, x0=1. Объясните наблюдающиеся различия.



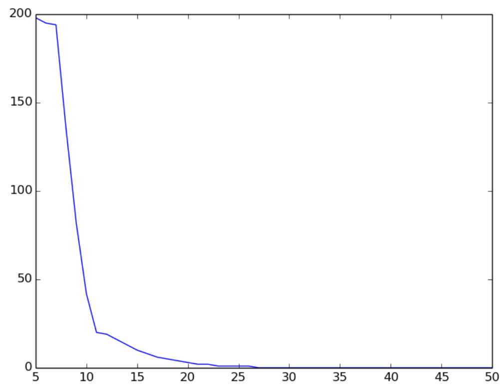


Производная функции |x-1| имеет особенность в точке x=1, что плохо для полиномиальной интерполяции, поэтому интерполяция второй функции дает большую погрешность.

14) Для функции из задания 9 постройте кубический сплайн дефекта 1 с узлами из задания 9.



15) Изучите зависимость погрешности приближения функции сплайном от числа узлов интерполяции. Сравните с результатом из задания 10.



Превосходство кубического сплайна видно сразу.

```
_p = scipy.interpolate.splrep(xn, f(xn))
def p(x):
     return scipy.interpolate.splev(x, _p)
```

16) Как можно обобщить изученные методы интерполяции на кривые в многомерном пространстве?

Можно создавать гладкие сплайны любого порядка, формы, а также осуществлять локальный контроль над кривой. Кривая представляется в параметрической форме, и для управления формой кривой можно использовать контрольные точки и весовые коэффициенты узлов. Параметрическое представление кривой позволяет использовать её для интерполяции данных в многомерных пространствах. Имеет смысл разбить кривую на регулярные части, и дальше либо интерполировать координатные отображения, либо интерполировать кривизну, кручение и дальше решать диффуры в базисе Френе, не только для 2,3-мерного случая, но и для n-мерного случая, так как базис Френе можно построить для любой размерности.

17) Как можно интерполировать функции нескольких переменных?

Можно использовать билиниейную интерполяцию. Сумма по всем переменым в случае 3-х и более переменных. Коэфициенты тогда можно искать, составляя слагаемые по принципу интерполяции Лагранжа, а затем перейти к системе линейных алгебраических уравнений. Определить там получится некоторое «обобщение» определителя Вандермонда. Можно использовать бикубическую интерполяцию или бикубическую интерполяцию сплайнами.

18) Какие еще способы интерполяции существуют?
    Существуют такие методы интерполяции, как тригонометрическая интерполяция, рациональная интерполяция, обратное интерполирование. Можно интерполировать дробно-рациональными функциями, тогда будет возможность неплохо интерполировать функции с особеностями, однако оценить ошибку теоретически весьма сложно. Можно интерполировать периодические функции синусами и косинусами, минимизируя ошибку в пространстве квадратично-интегрируемых функций.