Лабораторная работа 4, задание 1,2

Задания

1. Измените метод `__call__`, так чтобы он реализовывал схему Горнера. Чем эта схема лучше?
2. Почему нахождение коэффициентов интерполяционного многочлена через решение системы дает ошибочный ответ?
3. Найдите определитель матрицы Вандермонда теоретически и численно.
4. Найдите числа обусловленности матрицы Вандермонда. Сравните экспериментально полученные погрешности решения системы и невязку с теоретическим предсказанием.

Горнер

При использовании схемы Горнена нужно совершать меньше операций, уменьшает погрешность вычесления.

```
class Poly2():
    def __init__(self, pn):
        """
        Создает многочлен с коэффициентами pn:
            self(x) = sum_n pn[n] * x**n.
        Коэффициенты pn перечисляются в порядке 
возрастания степени одночлена.
        """
        self.pn = pn
    def __call__(self, x):
        """
        Вычисляет многочлен на векторе значений x.
        """
        p = self.pn[-1]
        for i in range(2,len(self.pn)+1):
            p = x*p+self.pn[len(self.pn)-i]
        return p

def interp_naive2(xn, fn):
    """
    Возвращает интерполяционный многочлен, принимающий в 
точках xp значение fp.
    """
    M = vandermonde(xn)
         pn = np.linalg.solve(M, fn)
    return Poly2(pn)
```

Обусловленность

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)

$$\mu(M) - \text{число обусловленности}$$

$$\Delta\theta = \mu(M)\Delta f \cdot \theta, \text{where} \Delta f - \text{error of polynom value}$$

$$\Delta\tilde{f} = (\mu(M)\Delta f \cdot \theta) \cdot (x\_k)\mathrm{N}, \text{where}((x\_k)\mathrm{N}) - \text{vector}, \Delta\tilde{f} - \text{error of value}$$

```
#Число обусловленности
M=vandermonde(x)
a= np.linalg.solve(M, y)
mu=np.linalg.cond(M)
print("Число обусловленности", mu)
print("Evaluated error of coefficients",
mu*epsilon*np.linalg.norm(a))
print("Evaluated error of coefficients",
mu*epsilon*np.linalg.norm(np.dot(M,a)))
```

```
Число обусловленности 2.6245217742490032e+17
Evaluated error of coefficients 14286.613072106418
Evaluated error of coefficients 520.2598313589505
```

Теоретическая погрешность слишком пессимистична

Определитель

Теоретически определитель матрицы Вандермонда считается по индукции (напишу только ответ):

$$\det(M) = \prod_{1 \leq m < k \leq n} (xk - xm)$$

Очевидна невырожденность при не совпадающих узлах.

```
def det_vandermond(x):
    n=len(x)
    d=1
    for m in range(n):
        for k in range(m+1,n):
            d*=(x[k]-x[m])
    return d
print(det_vandermond(x))
print(np.linalg.det(M))
```

```
3.074989474009368e-49
3.0753158985971445e-49
```

Задания

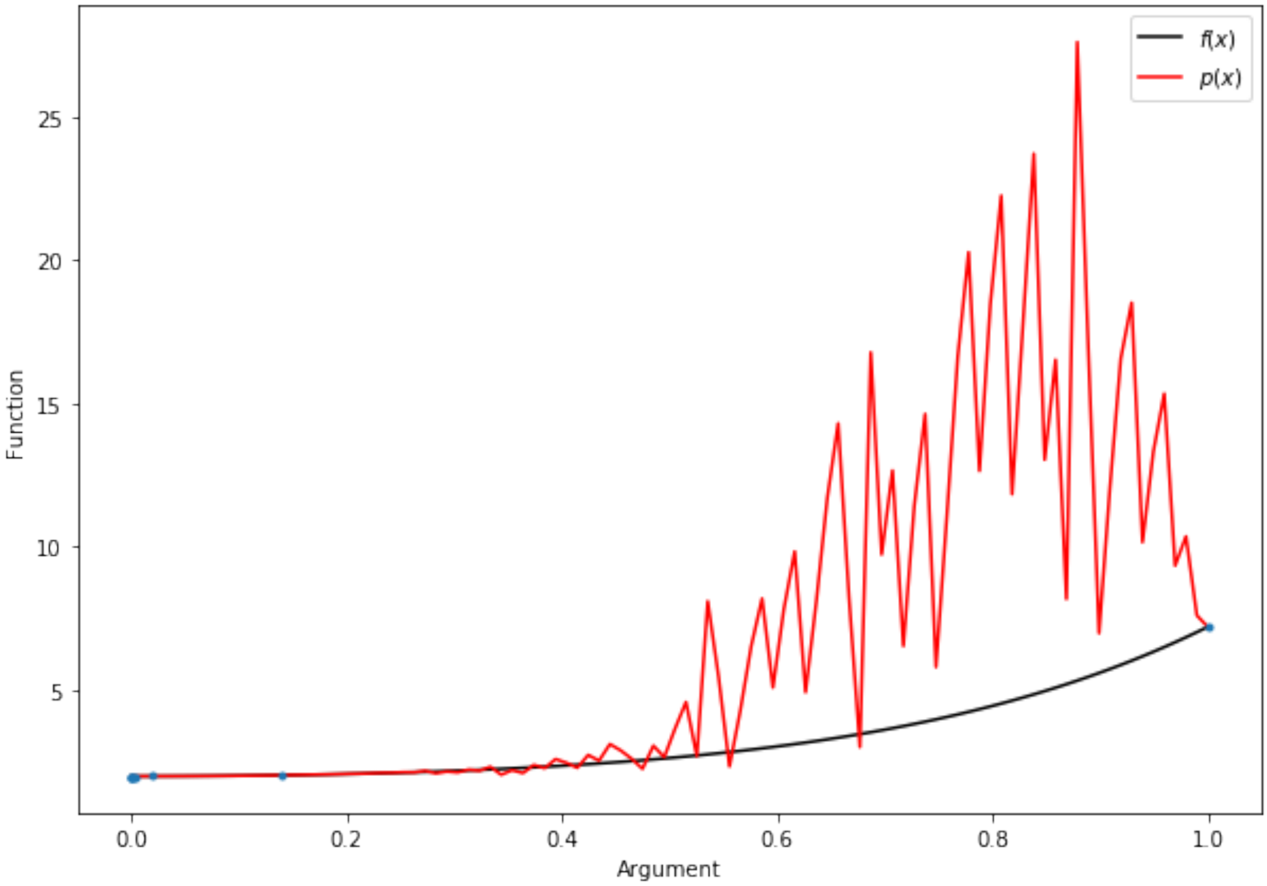
1. Реализуйте метод Эйткена вычисления интерполяционного многочлена.
2. Если мы попытаемся восстановить многочлен через его значения в точках, аналогично заданию 2, получим ли мы с помощью метода Эйткена ответ точнее, чем через решение системы?
3. Scipy содержит готовую реализацию интерполяционного многочлена Лагранжа `scipy.interpolate.lagrange`. В документации отмечается, что метод численно неустойчив. Что это означает?
4. 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?

```
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]))
```

```
t = np.linspace(0,1,100)
```

```
_, ax = plt.subplots(figsize=(10,7))
ax.plot(t, f(t), '-k')
ax.plot(t, l(t), '-r')
ax.plot(x, f(x), '.')
ax.set_xlabel("Argument")
ax.set_ylabel("Function")
ax.legend(["$f(x)$", "$p(x)$"])
plt.show()
d = l(x) # Значения интерполяционного многочлена на
решетке.
print("Absolute error of values", np.linalg.norm(d-y))
```

```
Absolute error of values 1.6764000044290905e-15
```



Как можно заметить мы не получили более точное решение, используя метод Эйткена, потому что задача все равно плохо обусловленна при таком выборе узлов.
Численная неустойчивость - это высокая чуствительность к возмущениям в исходных данных.
Оптимальными являются Чебышевские узлы, так как для них наилучшая ассимптотика константы Лебега. От располажения узлов зависит обусловленность задачи, что напрямую связанно с устойчивостью, чем хуже обусловленна задача, тем более вероятно метод окажется неустойчивым для нее.

```
g=scipy.interpolate.lagrange(x,y)
t = np.linspace(0,1,100)

_, ax = plt.subplots(figsize=(10,7))
ax.plot(t, f(t), '-k')
ax.plot(t, g(t), '-r')
ax.plot(x, f(x), '.')
ax.set_xlabel("Argument")
ax.set_ylabel("Function")
```

```
ax.legend(["$f(x)$", "$p(x)$"])
plt.show()
d = g(x) # Значения интерполяционного многочлена на решетке.
print("Absolute error of values", np.linalg.norm(d-y))
```

```
Absolute error of values 232.49175879773261
```