Задание на 30.01

Кириленко Андрей, ВШЭ

29 января 2019

$\boxed{\text{1a}}$

Считаем Лагранжа по формуле. Для этого нам потребуется генерировать массив коэффициентов, которые состоят из произведений, а также надо будет подставлять имеющиеся точки для интерполяции, итого получается две функции, которые вычисляют полином Лагранжа, они приведены ниже.

```
def gen_Lk(x, xi):
    xi = np.longdouble(xi)
    x = np.longdouble(x)
    res = np.longdouble([])

    for i, a in enumerate(xi):
        top = np.longdouble(1)
        bottom = np.longdouble(1)
        for j, b in enumerate(xi):
            if i != j:
                top *= (x - b)
                bottom *= (a - b)
        res = np.append(res, top / bottom)

    return res

def calc_lagrange(x, xvals, yvals):
    Lk = gen_Lk(x, xvals)
    result = np.longdouble(0)
    for i, value in enumerate(yvals):
        result += np.longdouble(value) * Lk[i]
    return result
```

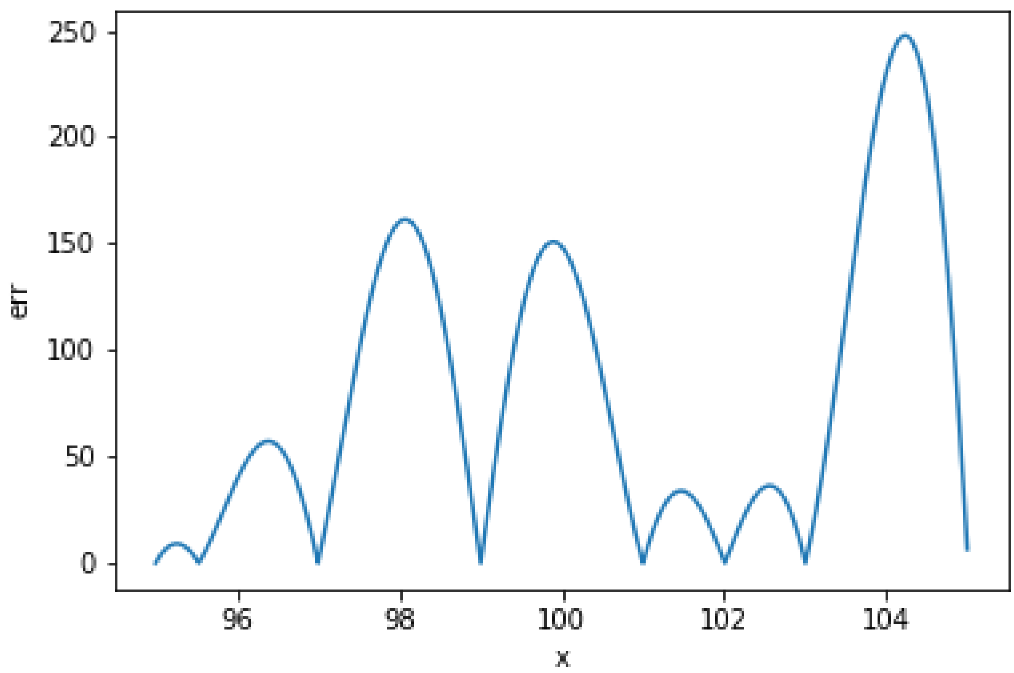
Далее генерируем deg+1 узел равномерно по формуле, для функции $f_S = xsin(2x)$, просто идем по отрезку от начала до конца, включая края, с одинаковым шагом, чтобы итоговое число точек было тем, которое необходимо:

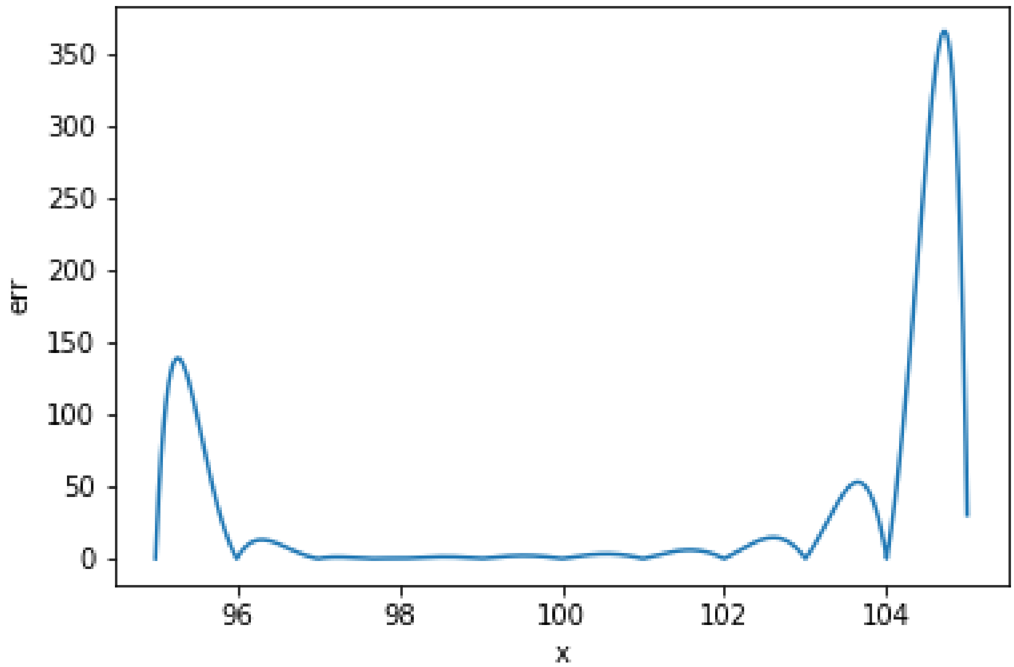
```
def gen_x_for_lagrange(x0, deg):
    return x0 - 5 + np.arange(0, deg + 1) * 10 / deg
```

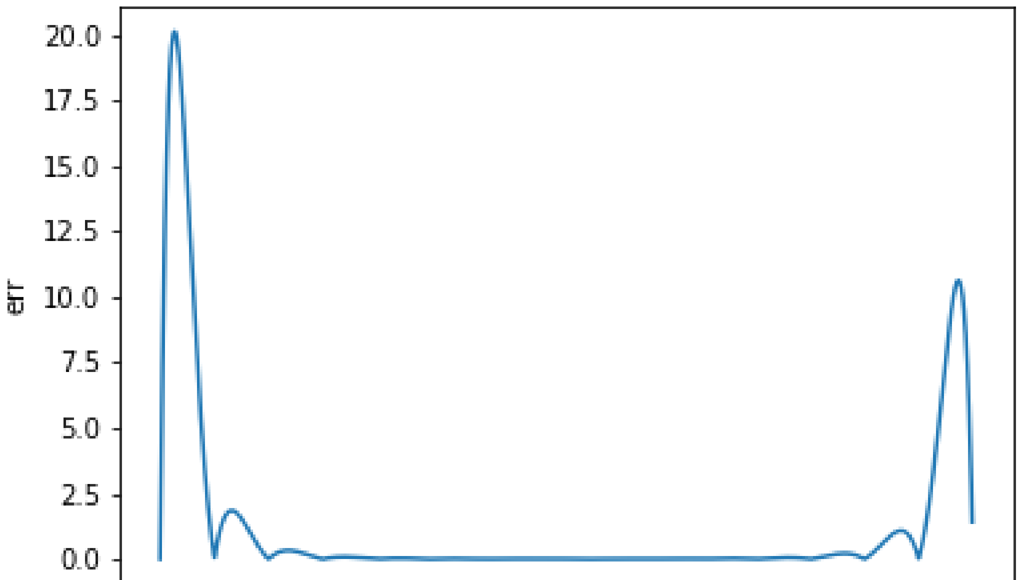
Далее строим графики для 1000 итераций(шаг это $\frac{1}{1000}$ длины отрезка, в коде $N = 1000$), при этом выясняя максимальную погрешность следующим образом:

```
def calc_err(xvals, yvals, a, b, N):
    ans = 0
    left = a
    step = np.longdouble((b - a) / N)
    while left <= b:
        ans = max(ans, np.abs(calc_lagrange(left, xvals, yvals) - func(left)))
        left += step
    return ans
```

Получим три графика (для N=5, 10, 15):







Максимальные ошибки при этом равны соответственно $N = 5$: 247.65768162102412, $N = 10$: 365.0420193178311,$N = 15$: 20.107935548939878.
Как и на лекции, ошибка растет на концах. Кроме того, с ростом N ошибка уменьшается($h = \frac{10}{N}$, после 10 уменьшается, до 10 растет), что соответствует оценке из теории $(O(h^{N+1}))$

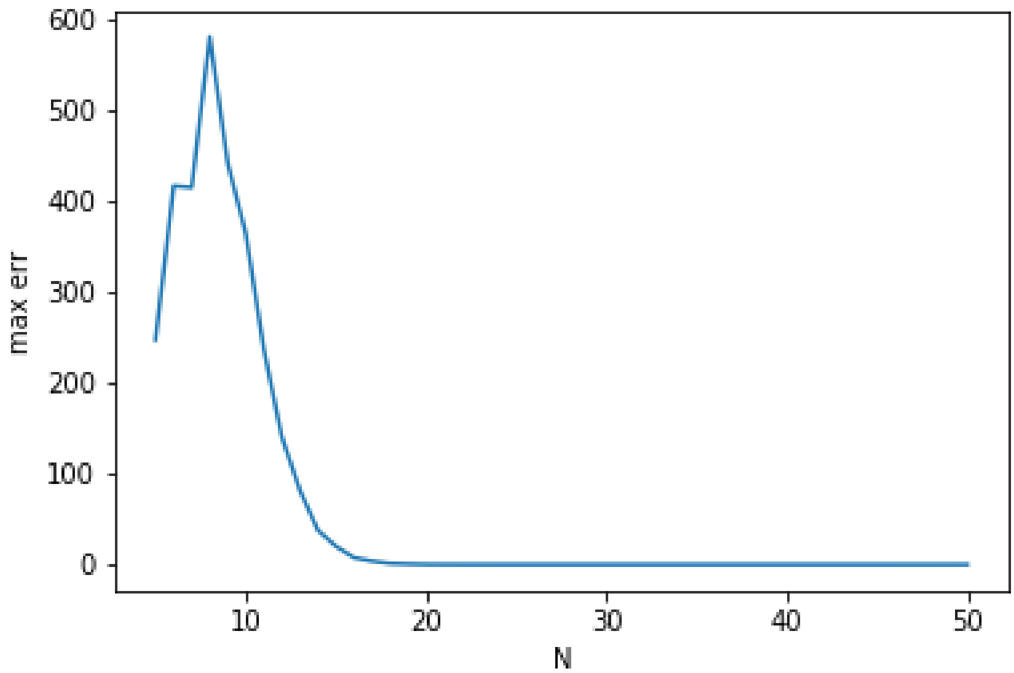
$\boxed{\text{1b}}$

Построим график, здесь ничего принципиально нового нет:

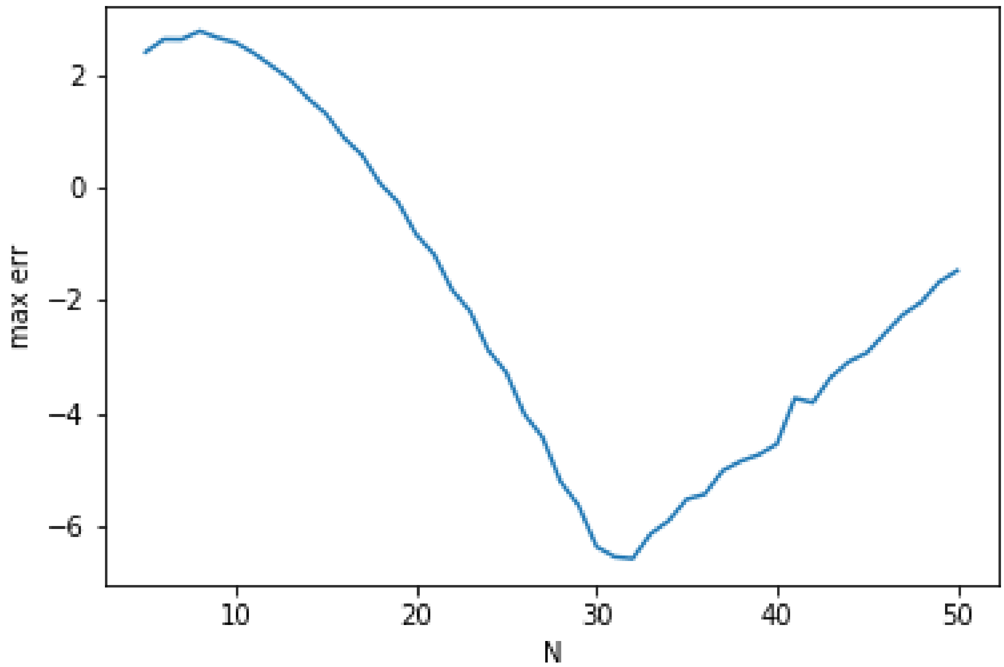
```
def task1b():
    xs = range(5, 51)
    x0 = 100
    ys = []
    for deg in range(5, 51):
        lagrange_x = gen_x_for_lagrange(x0, deg)
        lagrange_y = [func(pnt) for pnt in lagrange_x]
        ys.append(calc_err(lagrange_x, lagrange_y, x0 - 5, x0 + 5, 1000))
    graph.clf()
    graph.plot(xs, ys)
    graph.ylabel('max err')
    graph.xlabel('N')
    graph.savefig('1b.png')
```

Получится такой график:

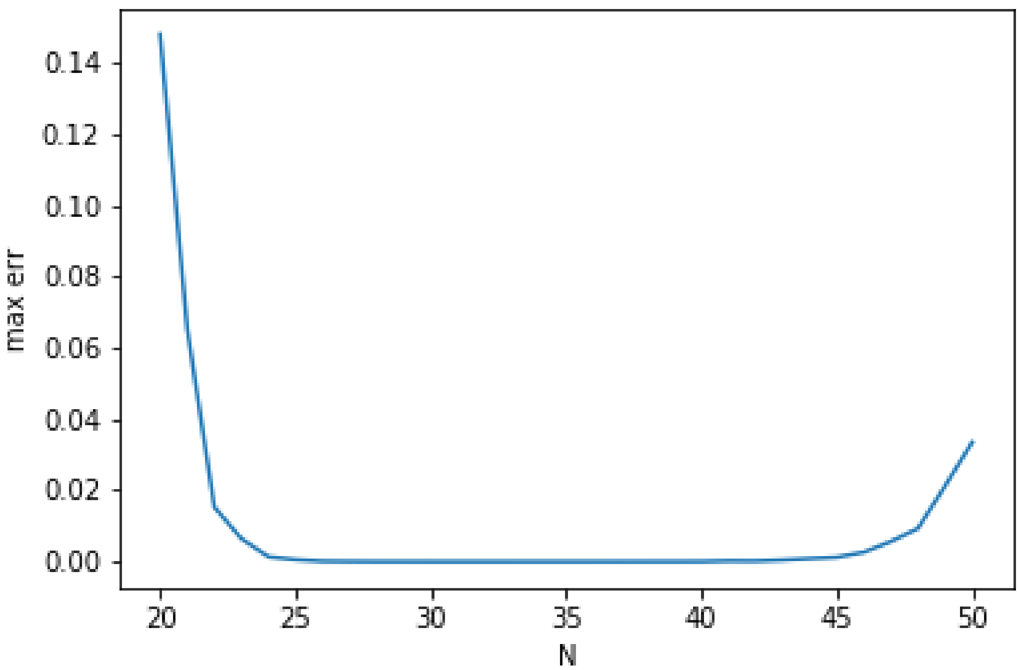
Объясняется он так: ошибка $O(h^{N+1})$, $h = \frac{10}{N}$, до 10 быстро растем, а потом быстро убываем. То есть около значения 10 изменение погрешности(локальная погрешность) наиболее сильное.



Можно также построить логарифм ошибки:



А еще рассмотреть отдельно середину графика для наглядности(можно заметить начавшийся рост при интерполяции высокими степенями, на логарифме это даже нагляднее):

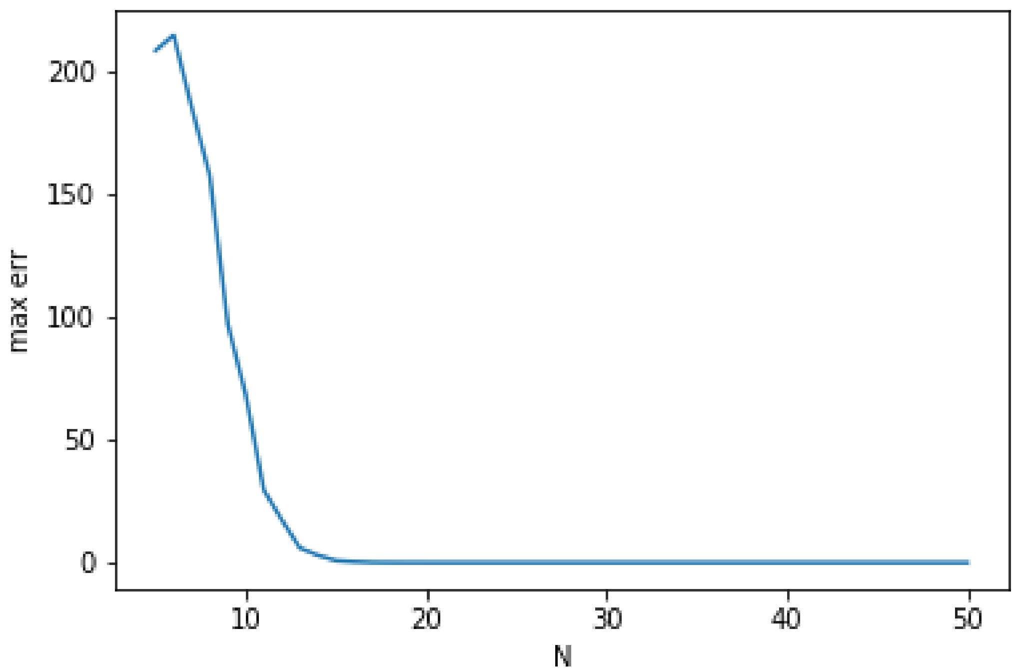


$\boxed{\text{1c}}$

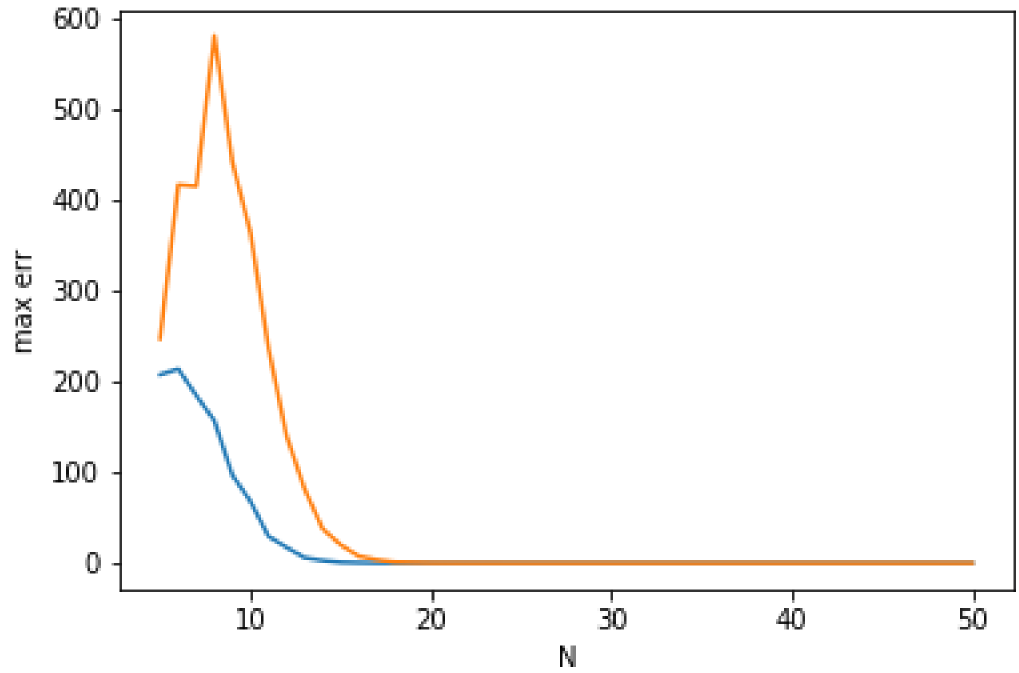
Рассмотрим код для Чебышева, генерируем узлы по форумлам, линейно отображая в нужный отрезок:

```
def cheb(k, deg):
    return np.longdouble(math.cos(np.longdouble(math.pi) / 2 * (2 * k - 1) /

def move_segment(a, b, t):
    return np.longdouble(0.5) * (a + b) + np.longdouble(0.5) * (b - a) * t

def gen_arr(N):
   return np.longdouble(np.array([np.cos((np.pi * (2 * k - 1)) / (2 * N)) for

def task1c():
    xs = range(5, 51)
    x0 = 100
    ys = []
    for deg in range(5, 51):
        cheb_arr = gen_arr(deg+1)
        lagrange_x = [move_segment(95, 105, t) for t in cheb_arr]
        lagrange_y = [func(pnt) for pnt in lagrange_x]
        ys.append(calc_err(lagrange_x, lagrange_y, x0 - 5, x0 + 5, 1000))
    graph.clf()
    graph.plot(xs, ys)
    graph.ylabel('max err')
    graph.xlabel('N')
    graph.savefig('1c.png')
```

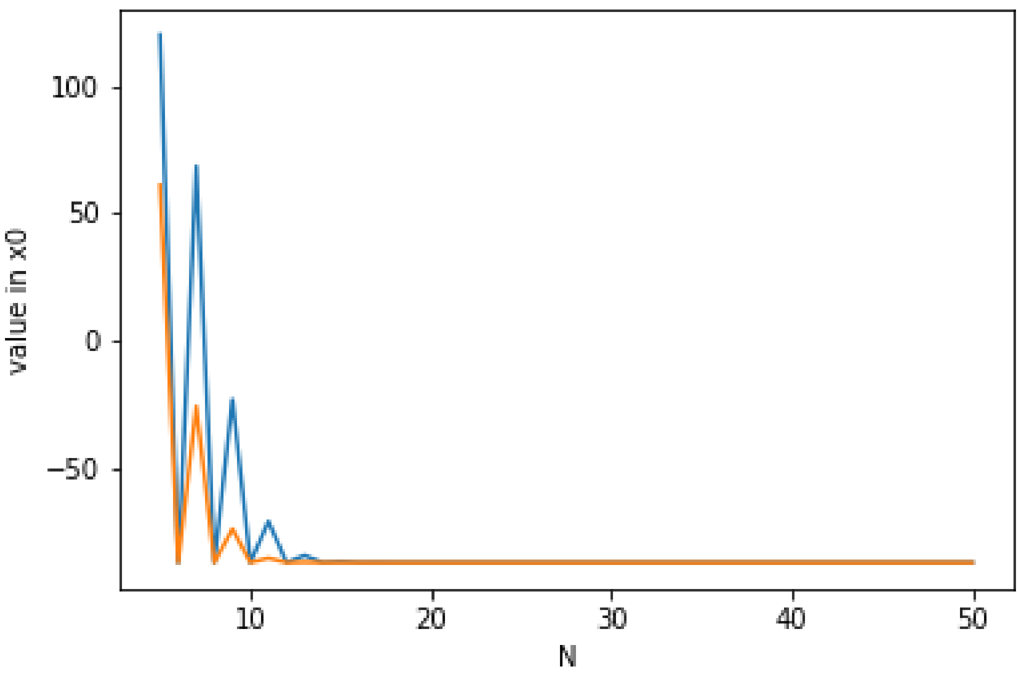
Это дает график:



А вместе с равномерными узлами(оранжевые):



Из графиков следует что более точные Чебышевские узлы имеют меньшую погрешность.
Графики с логарифмами будут приведены в конце.
Сравнение значений:



$\boxed{\text{1d}}$

Обновим функцию расчета ошибки(заменим на |x-1|), далее ничего нового.

```
def fm(x):
```

```
    return abs(x - 1)

def gen_x_for_lagrange_fm(x0=1, deg):
    return x0 - 1 + np.arange(0, deg + 1) * 2 / deg

def calc_err_fm(xvals, yvals, a, b, N):
    ans = 0
    left = a
    step = np.longdouble((b - a) / N)
    while left <= b:
        ans = max(ans, np.abs(calc_lagrange(left, xvals, yvals) - fm(left)))
        left += step
    return ans
```

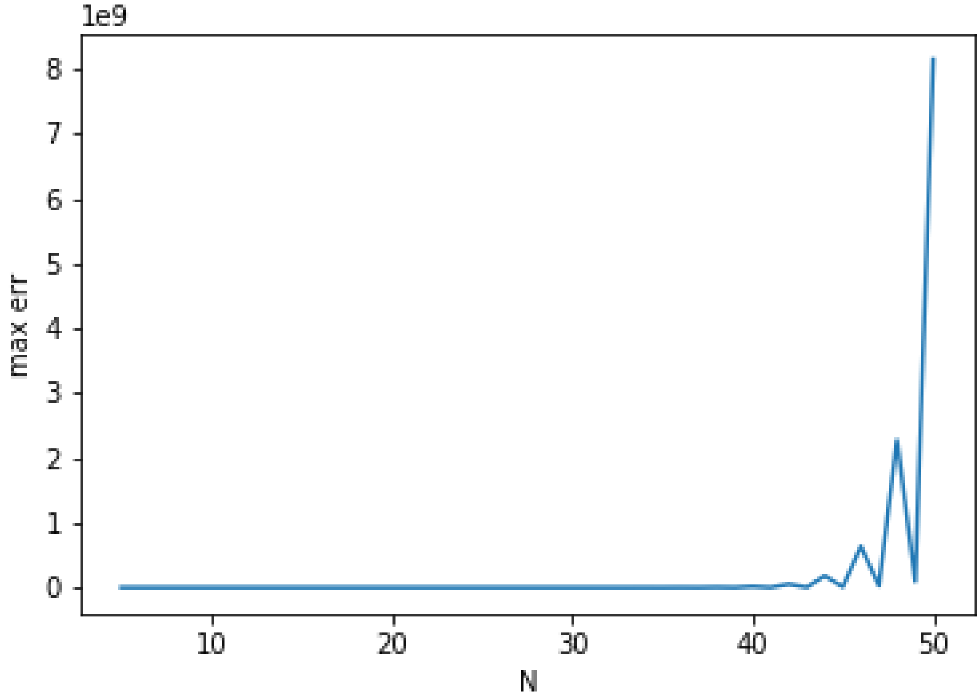
Затем строим графики погрешностей для двух методов выбора узлов:

```
def task1d_std():
    xs = range(5, 51)
    x0 = 1
    ys = []
    for deg in range(5, 51):
        lagrange_x = gen_x_for_lagrange_fm(x0, deg)
        lagrange_y = [fm(pnt) for pnt in lagrange_x]
        ys.append(calc_err_fm(lagrange_x, lagrange_y, x0 - 1, x0 + 1, 1000))
    graph.clf()
    graph.plot(xs, ys)
    graph.ylabel('max err')
    graph.xlabel('N')
    graph.savefig('1d_std.png')

def task1d_cheb():
    xs = range(5, 51)
    ys = []
    for deg in range(5, 51):
        cheb_arr = gen_arr(deg+1)
        lagrange_x = [move_segment(0, 2, t) for t in cheb_arr]
        lagrange_y = [fm(pnt) for pnt in lagrange_x]
        ys.append(calc_err_fm(lagrange_x, lagrange_y, 0, 2, 1000))
    graph.clf()
    graph.plot(xs, ys)
    graph.ylabel('max err')
    graph.xlabel('N')
```

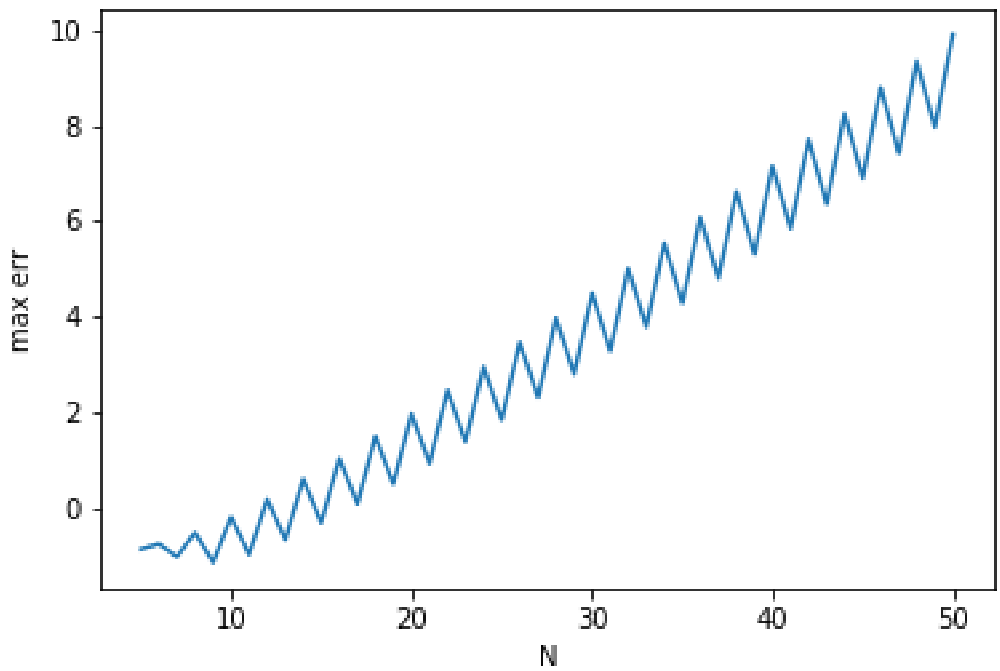
```
graph.savefig('1d_cheb.png')
```

Получим:

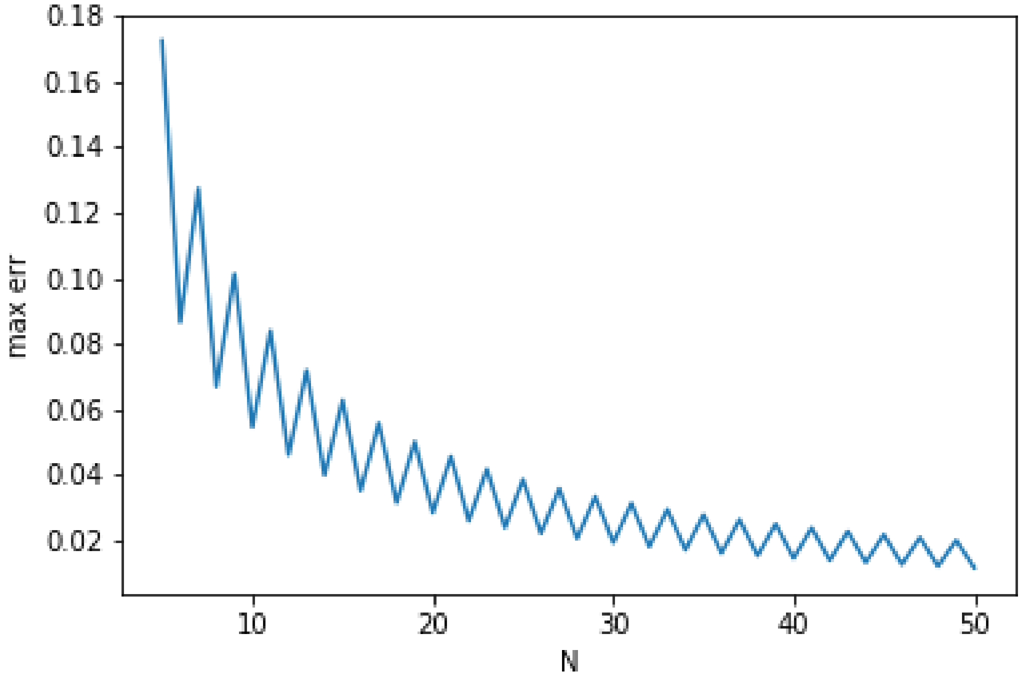
Равномерно, при больших N погрешность слишком велика(напрашивается экспонента, пробуем взять десятичный логарифм):



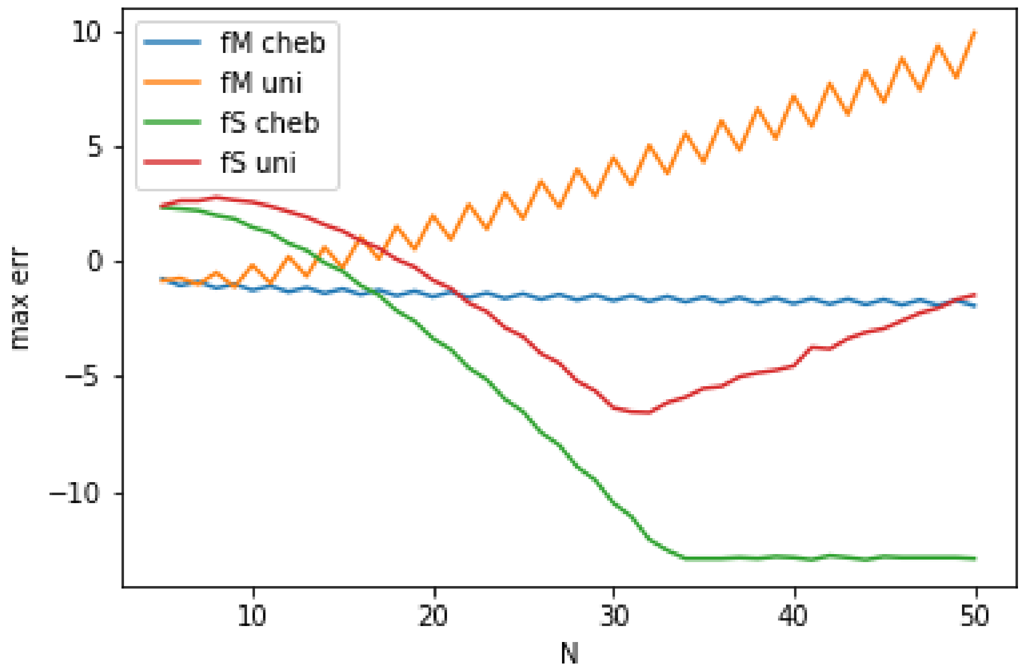
Логарифм, сразу видно экспоненту(у самой погрешности, а тут линейный график):



Чебышев, по графику погрешность имеет порядок $\frac{1}{n}$, а логарифм давал бы очевидно логарифмическую скорость:



Таким образом, на этом примере явно видны оценки из лекции на два вида выбора узлов – в первом он экспоненциальный, а для Чебышева логарифмический.
Сравним результаты с первой функцией(f_S), так как различия в графиках огромны, сразу построим логарифмы всего:



Какие выводы? Во-первых, видно что для первой функции погрешность растет до 10, согласно оценке с лекции. Во-вторых, экспоненциальная оценка на погрешность равномерных узлов имеет хорошую точность и достигается на примере второй функции. Наконец, с ростом N также начинается экспоннциальный рост ошибки и для первой функции при равномерном выборе узлов. Итого, Чебышевские узлы действительно оптимальнее.