Лабораторная работа №5
Ильин Денис, z3243

Задание

1. Объясните график ошибки. Почему ошибка сначала уменьшается, но потом растет? По какому закону происходит уменьшение и рост ошибки?
2. Для произвольной функции, оцените величину шага, при котором ошибка приближения производной минимальна. Какова минимальная ошибка такого метода приближения?

$$f^c(x) = f(x)(1 + \epsilon(x))$$

$$\frac{f^c(x+h) - f^c(x)}{h} = \frac{f(x+h) - f(x)}{h} + f\frac{\epsilon_1 - \epsilon_2}{h}.$$

$$f(x+h) = f + hf'(x) + \frac{h^2}{2}f''(x) + \mathrm{O}(h^3),$$

$$\frac{f^c(x+h) - f^c(x)}{h} = f'(x) + \mathrm{O}(f''(x)h) + \mathrm{O}\left(\frac{\epsilon f(x)}{h}\right)$$

$$\mathrm{O}(f''(x)h) = \frac{h}{2}f''(x) \qquad\qquad \mathrm{O}\left(\frac{\epsilon f(x)}{h}\right) = \frac{2\epsilon f(x)}{h}$$

Полная погрешность: $$\frac{h}{2}f''(x) + \frac{2\epsilon f(x)}{h}.$$

Получается, что при h>10^(-8) преобладает линейный член, при h<10^(-8) преобладает второй член.

Оптимальный шаг: $$h_0 \approx \sqrt{4\left|\frac{\epsilon f(x)}{f''(x)}\right|}.$$

Полная минимальная погрешность метода: $2\sqrt{\epsilon f(x) f''(x)}$

Задание

1. Объясните, почему прямая и обратная разделенные разности дают одинаковую погрешность, а центральная конечная разность дает более точный ответ?
2. Для произвольной функции оцените скорость уменьшения ошибка для центральной конечной разности. Как зависит скорость от гладкости функции?
3. Какова минимальная погрешность для вычисления центральной конечной разности?

По сути вычисление прямой и обратной разности ничем не отличается. Рассмотрим центральную разность:

$$f'(x) \approx \frac{f^c(x+h) - f^c(x-h)}{2h} = \frac{f(x+h) - f(x-h)}{2h} + \mathrm{O}\left(\frac{\epsilon}{h} f(x)\right)$$

$$\frac{f^c(x+h) - f^c(x-h)}{2h} = f'(x) + \frac{h^2}{3} f'''(x) + \mathrm{O}(h^3) + \mathrm{O}\left(\frac{\epsilon}{h} f(x)\right)$$

Оптимальный шаг: $(|3\epsilon f / f'''|)^{1/3}$

Полная минимальная погрешность: $(9\epsilon^2 f^2 f''')^{1/3}$

Центральная разность обладает меньшей погрешностью, так как в нее входит член epsilon^(2/3), тогда как в прямой (обратной) лишь epsilon^(1/2).

Задания

1. Реализуйте автоматическое дифференцирование для вычисления арктангенса.
2. Реализуйте автоматическое дифференцирование для двух переменных (можно ограничиться только арифметикой).
3. Какой ответ получается точнее: через автоматическое дифференцирование или через аналитическое выражение для производной, полученное через символьную алгебру? Что быстрее считается?

```
@staticmethod
    def arctan(x):
        return AG(np.arctan(x.v), x.d/(x.v^2+1))
```

```
class AG:
def __init__(self, a, b, c):
    """
    Инициализирует пару (f, df/dx, df/dy) = (a, b, c).
    """
    self.a = a
    self.b = b
    self.c = c

# Представление констант
@staticmethod
def const(x):
    return AG(x, 1, 1)

# Арифметические операции
def __add__(self, other):
    return AG(self.a+other.a, self.b+other.b, self.c+other.c)
def __sub__(self, other):
    return AG(self.a-other.a, self.b-other.b, self.c-other.c)
def __mul__(self, other):
    return AG(self.a*other.a, self.b*other.a + self.a+other.b, self.c*other.a + 
self.a+other.c)
def __truediv__(self, other):
    return AG(self.a/other.a, (self.b*other.a-self.a*other.b)/(other.a**2), 
(self.c*other.a-self.a*other.c)/(other.a**2))

# Возведение в степень
def __pow__(self, other):
    return AG(np.power(self.a, other.a), 
np.power(self.a,other.a-1.)*other.a*self.b 
                                            + 
np.power(self.a,other.a)*np.log(self.a)*other.b, 
np.power(self.a,other.a-1.)*other.a*self.c 
                                            + 
np.power(self.a,other.a)*np.log(self.a)*other.c)

# Основные функции
@staticmethod
def sin(x):
    return AG(np.sin(x.a), np.cos(x.a)*x.b, np.cos(x.a)*x.c)

@staticmethod
def cos(x):
    return AG(np.cos(x.a), -np.sin(x.a)*x.b, -np.sin(x.a)*x.c)
```

```
@staticmethod
def log(x):
    return AG(np.log(x.a), x.b/x.a, x.c/x.a)

@staticmethod
def arctan(x):
    return AG(np.arctan(x.a), x.b/(x.a^2+1), x.c/(x.a^2+1))
```

```
Через символьную алгебру лучше, так как можно контролировать
порядок выполнения операций в отличии от данного метода.
```

Задание

1. Объясните различия в точностях приближения центральной и прямой разделенными разностями.
2. Точность приближения можно увеличить, уменьшая шаг решетки , что приводит к увеличению числа узлов и пропорциональному увеличению времени работы. Сколько памяти и времени можно сэкономить, используя центральную разность?
3. Как изменится результат, если шаги решетки не будут постоянными?
4. Чтобы получить на произвольной решетке такой же аккуратный результат, как центральная разность на равномерной решетке, необходимо использовать три соседних узла , и , чтобы получить производную в . Выведите соответствующую формулу и проведите численный эксперимент для оценки погрешности.
5. Выведите формулу для оценки производной с погрешностью на равномерной решетке, используя значения функции в четырех узлах.

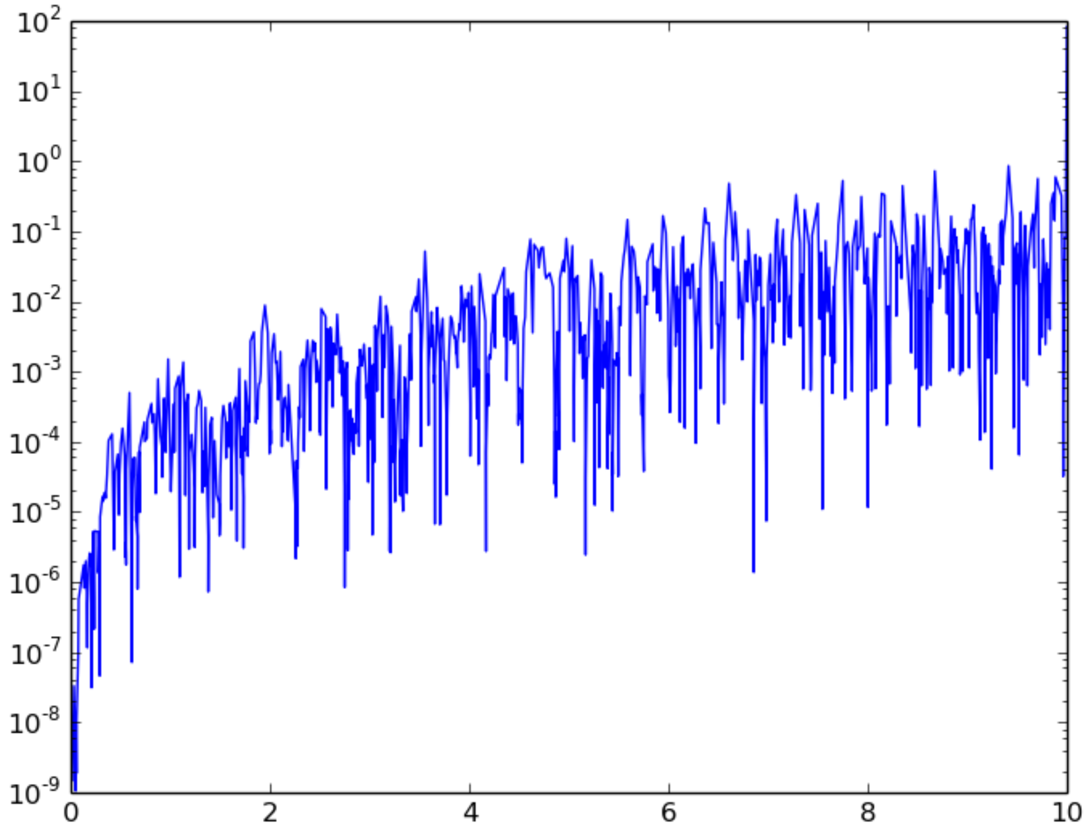
Первый пункт - был рассмотрен ранее.
Шаг нужен в 1000 раз больше, следовательно памяти в 1000 раз больше. Для прямой разности принципиально ничего не поменяется. Для центральной нужно будет ввести следующие поправки:

$$\frac{h_1^2(f(x_{k+1})-f(x_k))-h_2^2(f(x_{k-1})-f(x_k))}{h_1h_2^2+h_2h_1^2}$$

Где h1=x_k - x_k-1, h2=x_k+1 - x_k

```
xk = np.random.uniform(0,10,1000); xk=np.sort(xk)
fk = f(xk)

new_dfdx = np.empty_like(xk)
new_dfdx[0] = dfdx(xk[0])
for i in range(1,999):
    h1 = xk[i] - xk[i-1]
    h2 = xk[i+1] - xk[i]
```




```
    new_dfdx[i] = (h1*h1*(f(xk[i+1])-f(xk[i]))-h2*h2*(f(xk[i-1])-f(xk[i])))/
(h1*h2*(h1+h2))
```

Оценка производной для четырех узлов:

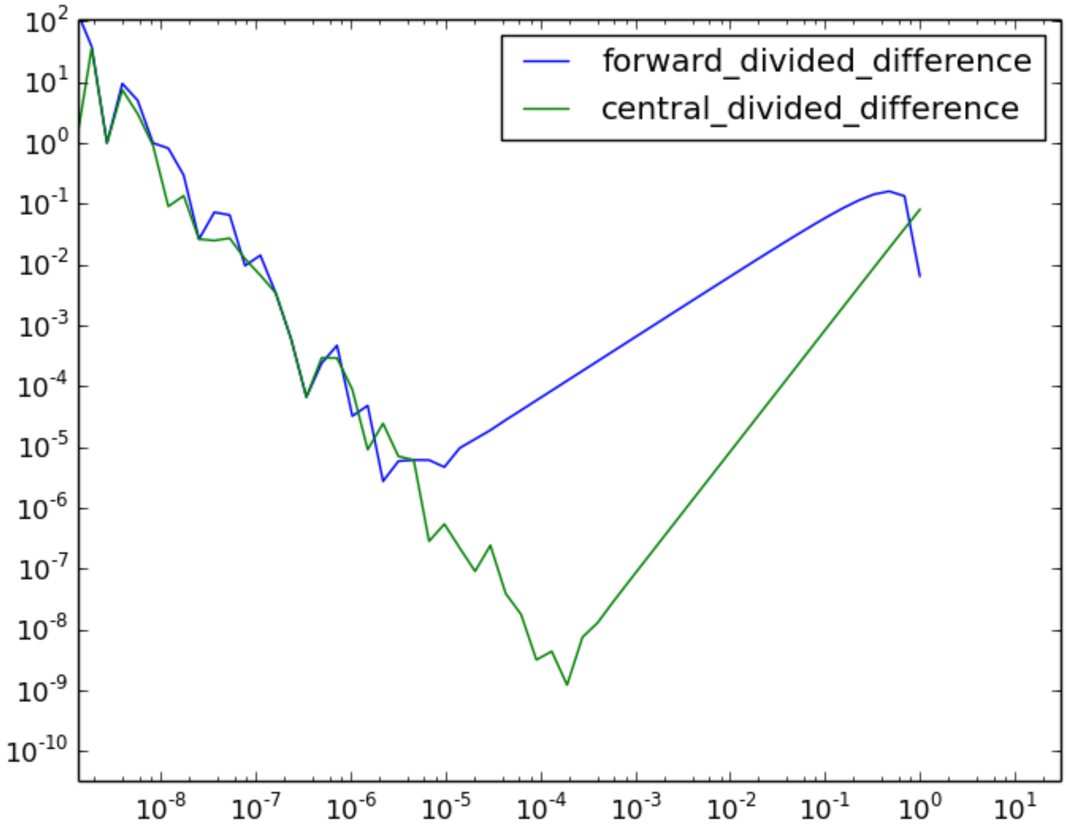
$$\frac{6f(x+h)-f(x+2h)-2f(x-h)-3f(x)}{6h}$$

Задание

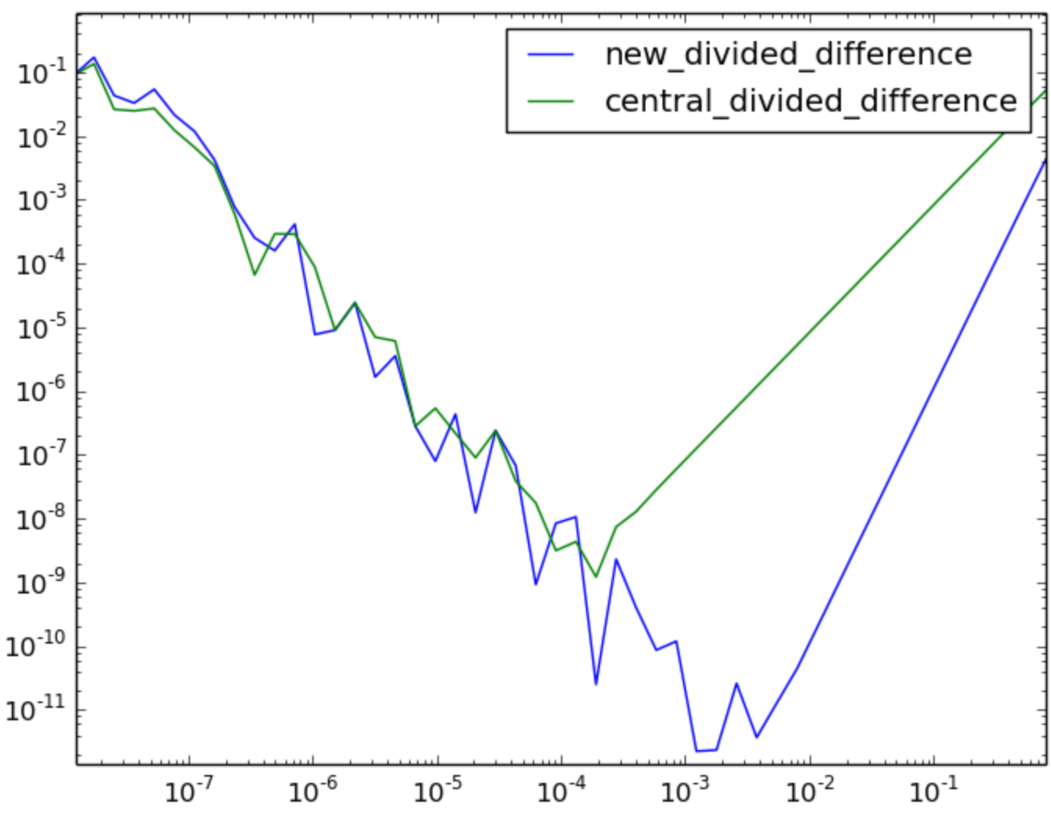
1. Сравните теоретически и экспериментально погрешности для оценки второй производной через прямую и центральную вторые разделенные разности.
2. Получите и проверьте формулу для оценки второй производной с точностью . Сколько узлов для этого нужно использовать?

Полная погрешность для центральной разности

$$\frac{h^2}{12}f''''(x)+\frac{4\epsilon}{h^2}|f(x)|$$



$$y''(x) = (-y(x+2h) + 16y(x+h) - 30y(x) + 16y(x-h) - y(x-2h))/(12h^2) + O(h^4)$$



Оценка второй производной с точностью o(h^4).

```
def experiment(method, f=np.sin, dfdx=np.sin, x0=1, dx = np.logspace(-16, 0, 
100)):

    approx_dfdx = method(f, x0, dx)
    exact_dfdx = -dfdx(x0)
    relative_error = np.abs(1.0-approx_dfdx/exact_dfdx)

    plt.loglog(dx, relative_error, label=method.__name__)
    plt.xlabel("Приращение аргумента")
    plt.ylabel("Относительная погрешность")
    return relative_error

def forward_divided_difference(f, x0, dx):

    return (f(x0+2*dx)-2*f(x0+dx)+f(x0))/(dx*dx)

def new_divided_difference(f, x0, dx):

    return (-f(x0+2*dx)+16*f(x0+dx)-30*f(x0)+16*f(x0-dx)-f(x0-2*dx))/(12*dx*dx)

def central_divided_difference(f, x0, dx):

    return (f(x0+dx)-2*f(x0)+f(x0-dx))/(dx*dx)
```