Решение систем линейных алгебраических уравнений методом минимальных невязок

Описание метода:

В данной задаче имеется матричное уравнение вида *Ax = f*. Буду решать его с помощью градиентного метода.

Также матрица A должна быть положительно определенной для того, чтобы по мере итераций решение сходилось.

Выберем начальное приближение x_0. Следующее приближение буду строить из предыдущего путем добавления к нему вектора невязки, умноженного на некоторый скаляр.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{r}_k$$

Вектор невязки будет искаться так:

$$\mathbf{r}_k = \mathbf{f} - \mathbf{A}\mathbf{x}_k$$

Коэффициент альфа будет искаться из следующего выражения:

$$\alpha_k = \frac{(\mathbf{A}\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{A}\mathbf{r}_k, \mathbf{A}\mathbf{r}_k)}$$

Тогда за k шагов получим некоторое приближение решения.

Необходимые формулы:

перечислены выше

Программная реализация:

```python
import numpy as np

eps = 10E-8


def lin_solve(a: np.array, f: np.array, it=1000):
    n = a.shape[0]
    x = np.zeros(n)
    for i in range(it):
        r = f - a.dot(x)
        ar = a.dot(r)
        al = ar.dot(r) / (ar.dot(ar))
        x += r * al
        if abs(r)<eps:
            print('Нужная точность была достигнута')
            print('Кол-во итераций: ', i)
            break
    return x


for _ in range(10):
    a = np.random.uniform(-12, 12, (5, 5))
    a = a.dot(a.transpose())
    f = np.random.uniform(-12, 12, a.shape[0])
    x = lin_solve(a, f)
    print(np.linalg.norm(a.dot(x) - f))
```

Проверка правильности реализации

Буду задавать случайную матрицу *A*. Затем для получения положительно определенной буду умножать *A* на саму себя, но транспонированную. Легко понять, что матрица AA^T будет положительно определенной.

Проверю работу алгоритма на случайных входных данных. Количество итераций установлю равным 1000. Буду выводить норму вектора невязки после каждого теста реализованной функции.

```
for _ in range(10):
    a = np.random.uniform(-12, 12, (5, 5))
    a = a.dot(a.transpose())
    f = np.random.uniform(-12, 12, a.shape[0])
    x = lin_solve(a, f)
    print(np.linalg.norm(a.dot(x) - f))
```

Результат:

```
2.498772674675193e-14
1.6280579355608324e-14
6.993523870246153e-15
2.1291704219146526e-13
3.0626061064194705e-14
2.0821154689797349e-13
2.831830528057311e-14
1.3322676295501878e-15
1.3732700395566711e-15
2.0689803639648097e-14
```

Как видно, алгоритм работает. Точность полученного решения близка к машинной, поэтому можно сказать, что здесь действительно наблюдается сходимость.

Теперь модифицирую функцию так, чтобы она останавливалась, когда достигается нужная точность. Результат:

```
Нужная точность была достигнута
Кол-во итераций:  4566
9.930809037278394e-08
Нужная точность была достигнута
Кол-во итераций:  94
8.00889333613461e-08
Нужная точность была достигнута
Кол-во итераций:  436
9.459181251491504e-08
Нужная точность была достигнута
Кол-во итераций:  905
9.770705312030976e-08
```