# Преобразование матрицы с помощью матриц отражений

1. Постройте элементарную матрицу отражений (для ), проверьте автоматически её свойства: ортогональность, симметричность, собственные числа

* x.shape[0] – кол-во элементов
* np.identity(n) – единичная матрица
* A.transpose() – транспонирование
* np.linalg.eig(A) – собственные числа
* np.linalg.norm – норма вектора или матрицы (спектральная)
* np.linalg.norm(A,1) – макс. столбцовая матрицы
* Умножение матриц - A@B
* для преобразования вектора в матрицу используйте x.reshape((1,n)) (в строку) и x.reshape((n,1)) (в столбец)
* x.dot(y) – скалярное произведение
* np.random.rand(5,10) – случайная матрица, np.random.rand(5) – случайный вектор

1. Постройте оптимизированную процедуру для домножения на слева: без вычисления матрицы
   * – столбец матрицы
   * -это простое скалярное произведение (число).
   * умножение вектора на матрицу в Numpy – это домножение столбцов на коэффициенты

np.array([1,2,3])\* np.array([[5,6,7],[8,9,10]])   
array([[ 5, 12, 21],  
 [ 8, 18, 30]])

* + можно использовать x.dot(A) для вычисления строки скалярных произведений
  + Замер времени:

%time (код вычисления)

Проверьте, что новая функция выполняется значительно быстрее стандартного умножения (вместо , заметно на больших матрицах)

1. Реализуйте метод отражений: на шаге метода столбец должен будет после преобразования отражения принять вид c нулями под главной диагональю:

Для этого матрица A умножается слева на матрицу отражения ( с вектором

где

*-й* столбец матрицы

*-й* столбец матрицы (её текущего состояния после предыдущих шагов)  
с нулями на первых местах

Умножение на матрицу U происходит по ускоренной процедуре, первые и строк матрицы в умножении не участвуют (операции можно сэкономить).

В результате должна получиться верхнетреугольная матрица на месте матрицы A.

Не забываем преобразовывать вектор b (правую часть).

1. Применить для решения системы

с заранее известным случайным вектором и вычисленным b.

(можно проверить, сравнив scipy.linalg.solve\_triangular)

Отобразить погрешность решения системы.

1. Проверить успешную работу алгоритма для , сравнить время и точность решения со стандартным алгоритмом для решения систем.
2. Модифицировать код метода отражений, добавив хранение матрицы отражений:

соответствует разложению

Матрицу можно вычислить домножением текущей матрицы справа на каждом шаге:

(умножение справа также должно быть эффективным с использованием скалярных произведений)

Проверить

1. Сравнить QR-разложение с вычисленным стандартной процедурой

Q,R=np.linalg.qr(A)

1. \*Применить отражения для приведения матрицы к почти треугольному виду (форме Хессенберга) преобразованием подобия

* на первом шаге
* на втором шаге обнуляется , меняются 3-4 строки и 3-4 столбцы

1. \*Проверить, что собственные числа матрицы остались неизменными после преобразования.