Chapter 4. Matrix

4.1. Matrix operations

Untitled.ipynb

In [1]:
```
from numpy import array
A = [[1, 2, 3], [4, 5, 6]]; A
```

Out[1]:
```
[[1, 2, 3], [4, 5, 6]]
```

In [2]:
```
A = array(A); A
```

Out[2]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [3]:
```
print(A)
```

```
[[1 2 3]
 [4 5 6]]
```

In [4]:
```
L = A.tolist(); L
```

Out[4]:
```
[[1, 2, 3], [4, 5, 6]]
```

In [5]:
```
B = A.copy(); B
```

Out[5]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [6]:
```
A == B
```

Out[6]:
```
array([[ True,  True,  True],
       [ True,  True,  True]])
```

In [7]:
```
(A == B).all()
```

Out[7]:
```
True
```

In [8]:
```
B[0, 1] = 1
(A == B).all()
```

Out[8]:
```
False
```

In [9]:
```
(A == B).any()
```

Out[9]:
```
True
```

Untitled1.ipynb

In [1]:
```
from numpy import array
A = array([1, 2, 3]); A
```

Out[1]:
```
array([1, 2, 3])
```

In [2]:
```
B = A.reshape((1, 3)); B
```

Out[2]:
```
array([[1, 2, 3]])
```

In [3]:
```
C = B.reshape((3, 1)); C
```

Out[3]:
```
array([[1],
       [2],
       [3]])
```

In [4]:
```
D = C.reshape((3,)); D
```

Out[4]:
```
array([1, 2, 3])
```

In [5]:
```
A[0] = 0; A
```

Out[5]:
```
array([0, 2, 3])
```

In [6]:
```
B[0, 0], C[0, 0], D[0]
```

Out[6]:
```
(0, 0, 0)
```

Untitled2.ipynb

In [1]:
```
from numpy import array
A = array([[1, 2, 3], [4, 5, 6]])
B = array([[1, 3, 5], [2, 4, 6]])
A + B
```

Out[1]:
```
array([[ 2,  5,  8],
       [ 6,  9, 12]])
```

In [2]:
```
2 * A
```

Out[2]:
```
array([[ 2,  4,  6],
       [ 8, 10, 12]])
```

In [3]:
```
0 * A
```

Out[3]:
```
array([[0, 0, 0],
       [0, 0, 0]])
```

In [4]:
```
-1 * A
```

Out[4]:
```
array([[-1, -2, -3],
       [-4, -5, -6]])
```

Program: latex1.py / latex1.ipynb

In [1]:
```
from numpy.random import seed, randint, choice
from sympy import Matrix, latex

seed(2021)
```

```
m, n = randint(2, 4, 2)
X = [-3, -2, -1, 1, 2, 3, 4, 5]
A = Matrix(choice(X, (m, n)))
B = Matrix(choice(X, (m, n)))
print(f'{latex(A)} + {latex(B)} = ')
```

```
\left[\begin{matrix}-2 & -3 & 3\\4 & 4 & 2\end{matrix}\right] + \left[\begi
n{matrix}5 & 4 & 1\\3 & 3 & 3\end{matrix}\right] =
```

$$\left[\begin{matrix}-2 & -3 & 3\\4 & 4 & 2\end{matrix}\right] + \left[\begin{matrix}5 & 4 & 1\\3 & 3 & 3\end{matrix}\right] =$$

In [2]:
```
A+B
```

Out[2]:
$$\left[\begin{matrix}3 & 1 & 4\\7 & 7 & 5\end{matrix}\right]$$

template.tex

template.pdf

4.2. Matrix and linear mapping

Untitled.py

In [1]:
```
from numpy import array, dot
y = dot([[1, 2], [3, 4]], [5, 6]); y
```

Out[1]:
```
array([17, 39])
```

In [2]:
```
A = array([[1, 2], [3, 4]])
A.dot([5, 6])
```

Out[2]:
```
array([17, 39])
```

In [3]:
```
A.dot([[5], [6]])
```

Out[3]:
```
array([[17],
       [39]])
```

Program: mypict5.py

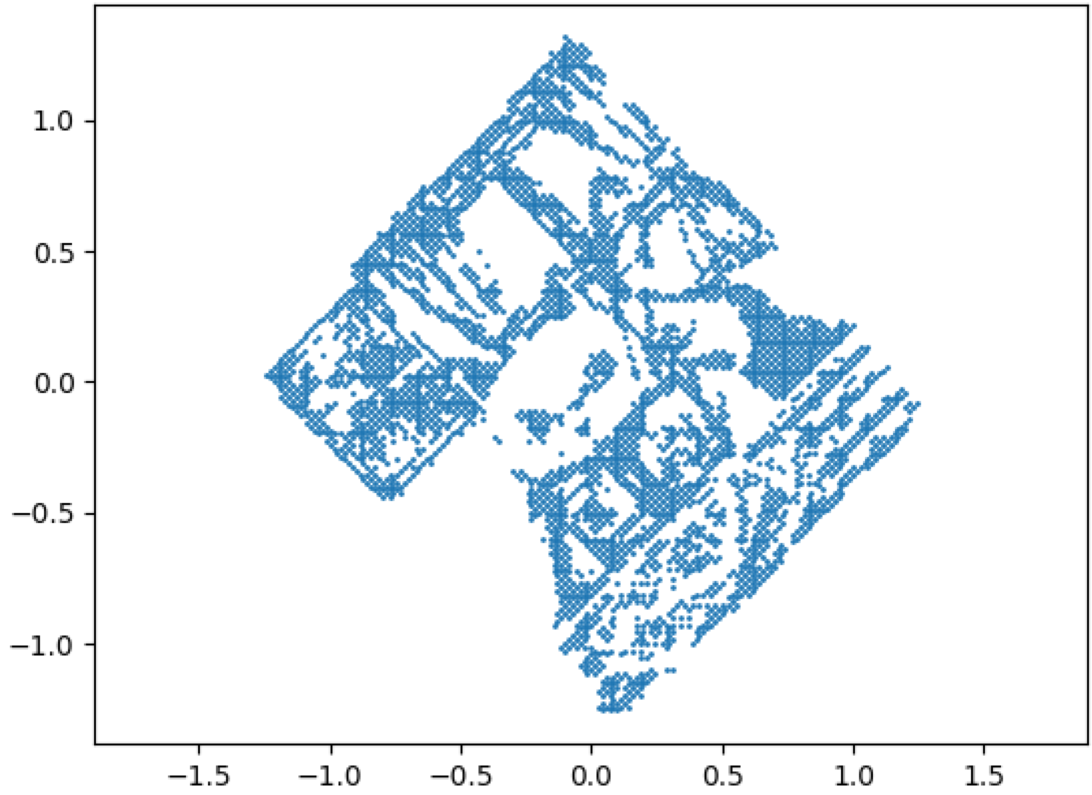
In [1]:
```
from numpy import array, pi, sin, cos
import matplotlib.pyplot as plt

t = pi / 4
A = array([[cos(t), -sin(t)], [sin(t), cos(t)]])

with open('mypict1.txt', 'r') as fd:
    P = eval(fd.read())

Q = [A.dot(p) for p in P]
x, y = zip(*Q)
plt.scatter(x, y, s=1)
plt.axis('equal')
```

Out[1]:
```
(-1.3679922752216784,
 1.367952868365186,
 -1.386564726686574,
 1.4437046686006383)
```



4.3. Composition of linear mappings and product of matrices

Untitled.ipynb

In [1]:
```python
import numpy as np
A = [[1, 2, 3], [4, 5, 6]]
B = [[1, 2], [3, 4], [5, 6]]
np.dot(A, B)
```

Out[1]:
```
array([[22, 28],
       [49, 64]])
```

In [2]:
```python
np.array(A).dot(B)
```

Out[2]:
```
array([[22, 28],
       [49, 64]])
```

In [3]:
```python
np.matrix(A) * np.matrix(B)
```

Out[3]:
```
matrix([[22, 28],
        [49, 64]])
```

Program: problems.py

In [1]:
```python
from numpy import array

A = array([[1, 2], [3, 4]])
B = array([[1, 2, 3], [4, 5, 6]])
```

```
C = array([[1, 2], [3, 4], [5, 6]])
D = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

for X in (A, B, C, D):
    for Y in (A, B, C, D):
        if X.shape[1] == Y.shape[0]:
            print(f'{X}\n{Y}\n= {X.dot(Y)}\n')
```

```
[[1 2]
 [3 4]]
[[1 2]
 [3 4]]
= [[ 7 10]
 [15 22]]

[[1 2]
 [3 4]]
[[1 2 3]
 [4 5 6]]
= [[ 9 12 15]
 [19 26 33]]

[[1 2 3]
 [4 5 6]]
[[1 2]
 [3 4]
 [5 6]]
= [[22 28]
 [49 64]]

[[1 2 3]
 [4 5 6]]
[[1 2 3]
 [4 5 6]
 [7 8 9]]
= [[30 36 42]
 [66 81 96]]

[[1 2]
 [3 4]
 [5 6]]
[[1 2]
 [3 4]]
= [[ 7 10]
 [15 22]
 [23 34]]

[[1 2]
 [3 4]
 [5 6]]
[[1 2 3]
 [4 5 6]]
= [[ 9 12 15]
 [19 26 33]
 [29 40 51]]

[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[1 2]
 [3 4]
 [5 6]]
= [[ 22  28]
 [ 49  64]
 [ 76 100]]

[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
= [[ 30  36  42]
 [ 66  81  96]
 [102 126 150]]
```

Untitled1.ipynb

In [1]:
```
from sympy import Matrix
from sympy.abc import a, b, c, d
A = Matrix([[1, 2], [3, 4]])
A/2
```

Out[1]: $$\begin{bmatrix} \frac{1}{2} & 1 \\ \frac{3}{2} & 2 \end{bmatrix}$$

In [2]:
```
B = Matrix([[a, b], [c, d]])
B/2
```

Out[2]: $$\begin{bmatrix} \frac{a}{2} & \frac{b}{2} \\ \frac{c}{2} & \frac{d}{2} \end{bmatrix}$$

In [3]:
```
A + B
```

Out[3]: $$\begin{bmatrix} a+1 & b+2 \\ c+3 & d+4 \end{bmatrix}$$

In [4]:
```
A * B
```

Out[4]: $$\begin{bmatrix} a+2c & b+2d \\ 3a+4c & 3b+4d \end{bmatrix}$$

Untitled2.ipynb

In [1]:
```
from sympy import Integer, Rational
2 / 3
```

Out[1]: 0.6666666666666666

In [2]:
```
Integer(2) / 3
```

Out[2]: $$\frac{2}{3}$$

In [3]:
```
Rational(2, 3)
```

Out[3]: $$\frac{2}{3}$$

Untitled3.ipynb

In [1]:
```
import numpy as np
import sympy as sp
np.zeros((2,3))
```

Out[1]:
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [2]:
```
sp.zeros(2,3)
```

Out[2]:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

In [3]:
```
np.eye(3)
```

Out[3]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [4]:
```
sp.eye(3)
```

Out[4]:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Program: mat_product1.py

In [1]:
```
from sympy import Matrix, sin, cos, eye
from sympy.abc import theta

A = Matrix([[cos(theta), sin(theta)],
            [-sin(theta), cos(theta)]])
B = Matrix([[1,  0],
            [0, -1]])
C = Matrix([[cos(theta), -sin(theta)],
            [sin(theta), cos(theta)]])
D = C * B * A
E = (eye(2)+D) / 2
print(f'D = {D}')
print(f'E = {E}')
```

```
D = Matrix([[-sin(theta)**2 + cos(theta)**2, 2*sin(theta)*cos(theta)], [2*s
in(theta)*cos(theta), sin(theta)**2 - cos(theta)**2]])
E = Matrix([[-sin(theta)**2/2 + cos(theta)**2/2 + 1/2, sin(theta)*cos(thet
a)], [sin(theta)*cos(theta), sin(theta)**2/2 - cos(theta)**2/2 + 1/2]])
```

Program: mat_product2.py

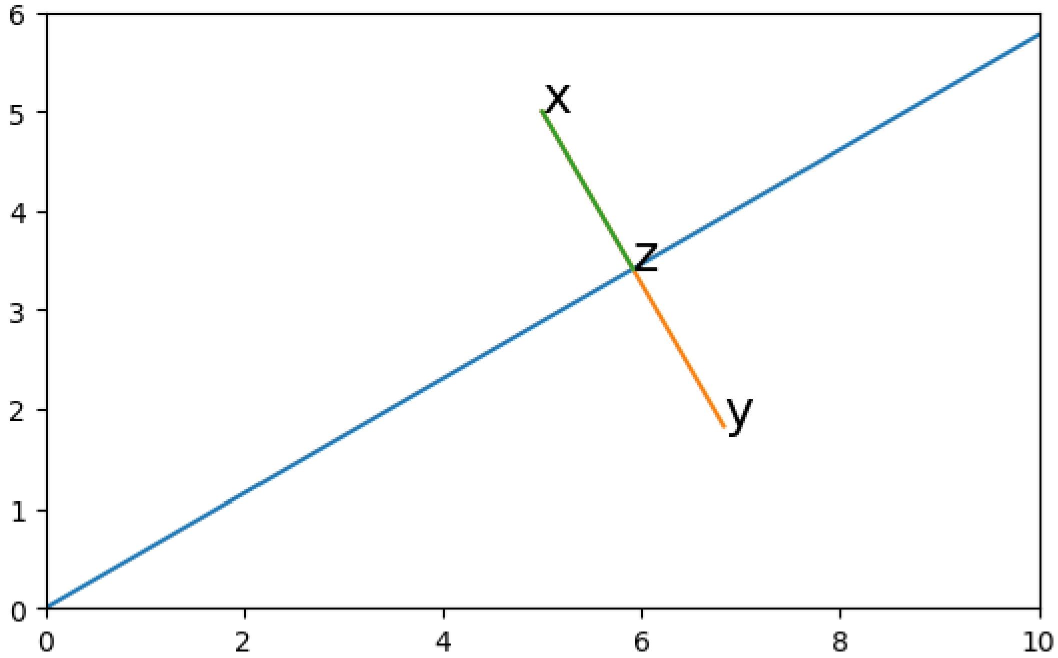
In [1]:
```
from numpy import matrix, sin, cos, tan, pi, eye
import matplotlib.pyplot as plt

t = pi / 6
A = matrix([[cos(t), sin(t)], [-sin(t), cos(t)]])
B = matrix([[1, 0], [0, -1]])
```

```
C = matrix([[cos(t), -sin(t)], [sin(t), cos(t)]])
D = C * B * A
E = (eye(2)+D) / 2
x = matrix([[5], [5]])
y = D * x
z = E * x

plt.plot([0, 10], [0, 10 * tan(t)])
plt.plot([x[0, 0], y[0, 0]], [x[1, 0], y[1, 0]])
plt.plot([x[0, 0], z[0, 0]], [x[1, 0], z[1, 0]])
plt.text(x[0, 0], x[1, 0], 'x',  fontsize=18)
plt.text(y[0, 0], y[1, 0], 'y',  fontsize=18)
plt.text(z[0, 0], z[1, 0], 'z',  fontsize=18)
plt.axis('scaled'), plt.xlim(0, 10), plt.ylim(0, 6)
```

Out[1]: ((-0.5, 10.5, -0.28867513459481287, 6.06217782649107), (0.0, 10.0), (0.0, 6.0))



Program: latex2.py

In [1]:
```
from numpy.random import seed, choice
from sympy import Matrix, latex

seed(2021)
template = r'''
\begin{array}{ll}
(1) &%s%s =\\[0.5cm]
(2) &%s%s =\\[0.5cm]
(3) &%s%s =\\[0.5cm]
(4) &%s%s =\\[0.5cm]
(5) &%s%s =\\
\end{array}
'''

matrices= ()
for no in range(5):
    m, el, n = choice([2, 3], 3)
    X = [-3, -2, -1, 1, 2, 3, 4, 5]
```

```
    A = Matrix(choice(X, (m, el)))
    B = Matrix(choice(X, (el, n)))
    matrices += (latex(A), latex(B))
print(template % matrices)
```

```
\begin{array}{ll}
(1) &\left[\begin{matrix}-3 & 3 & 4\\4 & 2 & 5\end{matrix}\right]\left[\begin{matrix}4 & 1 & 3\\3 & 3 & -3\\2 & 4 & 4\end{matrix}\right] =\\[0.5cm]
(2) &\left[\begin{matrix}3 & 5\\-2 & 4\end{matrix}\right]\left[\begin{matrix}-2 & 2 & 3\\-1 & -1 & -3\end{matrix}\right] =\\[0.5cm]
(3) &\left[\begin{matrix}-3 & -1 & 4\\1 & 2 & 3\\-3 & 3 & -2\end{matrix}\right]\left[\begin{matrix}4 & -1 & 4\\5 & 3 & 4\\-2 & 3 & 4\end{matrix}\right] =\\[0.5cm]
(4) &\left[\begin{matrix}2 & 3\\-1 & 1\\-2 & -1\end{matrix}\right]\left[\begin{matrix}-1 & 5\\-3 & -1\end{matrix}\right] =\\[0.5cm]
(5) &\left[\begin{matrix}1 & -2 & 4\\-2 & -2 & -1\end{matrix}\right]\left[\begin{matrix}5 & 3 & 1\\1 & 2 & 5\\5 & 1 & -2\end{matrix}\right] =\\
\end{array}
```

template.tex

template.pdf

$$\begin{array}{ll}
(1) &\left[\begin{matrix}-3 & 3 & 4\\4 & 2 & 5\end{matrix}\right]\left[\begin{matrix}4 & 1 & 3\\3 & 3 & -3\\2 & 4 & 4\end{matrix}\right] =\\[0.5cm]
(2) &\left[\begin{matrix}3 & 5\\-2 & 4\end{matrix}\right]\left[\begin{matrix}-2 & 2 & 3\\-1 & -1 & -3\end{matrix}\right] =\\[0.5cm]
(3) &\left[\begin{matrix}-3 & -1 & 4\\1 & 2 & 3\\-3 & 3 & -2\end{matrix}\right]\left[\begin{matrix}4 & -1 & 4\\5 & 3 & 4\\-2 & 3 & 4\end{matrix}\right] =\\[0.5cm]
(4) &\left[\begin{matrix}2 & 3\\-1 & 1\\-2 & -1\end{matrix}\right]\left[\begin{matrix}-1 & 5\\-3 & -1\end{matrix}\right] =\\[0.5cm]
(5) &\left[\begin{matrix}1 & -2 & 4\\-2 & -2 & -1\end{matrix}\right]\left[\begin{matrix}5 & 3 & 1\\1 & 2 & 5\\5 & 1 & -2\end{matrix}\right] =
\end{array}$$

4.4. Inverse matrix, basis change and similarity of matrices

Program:mat_product3.py

In [1]:
```
from sympy import Matrix, solve, eye
from sympy.abc import a, b, c, d, e, f

A = Matrix([[1, 2, 3], [2, 3, 4]])
B = Matrix([[a, b], [c, d], [e, f]])
```

```
ans = solve(A*B - eye(2), [a, b, c, d, e, f])
print(ans)
```

```
{a: e - 3, c: 2 - 2*e, b: f + 2, d: -2*f - 1}
```

In [2]:
```
C = B.subs(ans); C
```

Out[2]:
$$\begin{bmatrix} e-3 & f+2 \\ 2-2e & -2f-1 \\ e & f \end{bmatrix}$$

In [3]:
```
A * C
```

Out[3]:
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Untitled.ipynb

A = [[1, 2], [2, 1]] from numpy.linalg import inv inv(A)

In [2]:
```
from numpy import matrix
matrix(A)**(-1)
```

Out[2]:
```
matrix([[-0.33333333,  0.66666667],
        [ 0.66666667, -0.33333333]])
```

In [3]:
```
matrix(A)**2
```

Out[3]:
```
matrix([[5, 4],
        [4, 5]])
```

Untitled1.ipynb

In [1]:
```
from sympy import Matrix, S
Matrix([[1, 2], [2, 1]]) ** (-1)
```

Out[1]:
$$\begin{bmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{bmatrix}$$

In [2]:
```
A = Matrix([[S('a'), S('b')], [S('c'), S('d')]])
A**(-1)
```

Out[2]:
$$\begin{bmatrix} \frac{d}{ad-bc} & -\frac{b}{ad-bc} \\ -\frac{c}{ad-bc} & \frac{a}{ad-bc} \end{bmatrix}$$

In [3]:
```
A**2
```

Out[3]:
$$\begin{bmatrix} a^2+bc & ab+bd \\ ac+cd & bc+d^2 \end{bmatrix}$$

4.5. Adjoint matrix

Untitled.ipynb

In [1]:
```
from numpy import *
A = array([[1 + 2j, 2 + 3j, 3 + 4j],
           [2 + 3j, 3 + 4j, 4 + 5j]])
A.T
```

Out[1]:
```
array([[1.+2.j, 2.+3.j],
       [2.+3.j, 3.+4.j],
       [3.+4.j, 4.+5.j]])
```

In [2]:
```
A.conj()
```

Out[2]:
```
array([[1.-2.j, 2.-3.j, 3.-4.j],
       [2.-3.j, 3.-4.j, 4.-5.j]])
```

In [3]:
```
A = matrix(A); A
```

Out[3]:
```
matrix([[1.+2.j, 2.+3.j, 3.+4.j],
        [2.+3.j, 3.+4.j, 4.+5.j]])
```

In [4]:
```
A.H
```

Out[4]:
```
matrix([[1.-2.j, 2.-3.j],
        [2.-3.j, 3.-4.j],
        [3.-4.j, 4.-5.j]])
```

Untitled1.ipynb

In [1]:
```
from sympy import Matrix
A = Matrix([[1 + 2j, 2 + 3j, 3 + 4j],
            [2 + 3j, 3 + 4j, 4 + 5j]]); A
```

Out[1]:
$$\begin{bmatrix} 1.0 + 2.0i & 2.0 + 3.0i & 3.0 + 4.0i \\ 2.0 + 3.0i & 3.0 + 4.0i & 4.0 + 5.0i \end{bmatrix}$$

In [2]:
```
A.T
```

Out[2]:
$$\begin{bmatrix} 1.0 + 2.0i & 2.0 + 3.0i \\ 2.0 + 3.0i & 3.0 + 4.0i \\ 3.0 + 4.0i & 4.0 + 5.0i \end{bmatrix}$$

In [3]:
```
A.C
```

Out[3]:
$$\begin{bmatrix} 1.0 - 2.0i & 2.0 - 3.0i & 3.0 - 4.0i \\ 2.0 - 3.0i & 3.0 - 4.0i & 4.0 - 5.0i \end{bmatrix}$$

In [4]:
```
A.H
```

Out[4]:
$$\begin{bmatrix} 1.0 - 2.0i & 2.0 - 3.0i \\ 2.0 - 3.0i & 3.0 - 4.0i \\ 3.0 - 4.0i & 4.0 - 5.0i \end{bmatrix}$$

4.6. Measuring matrix computation time

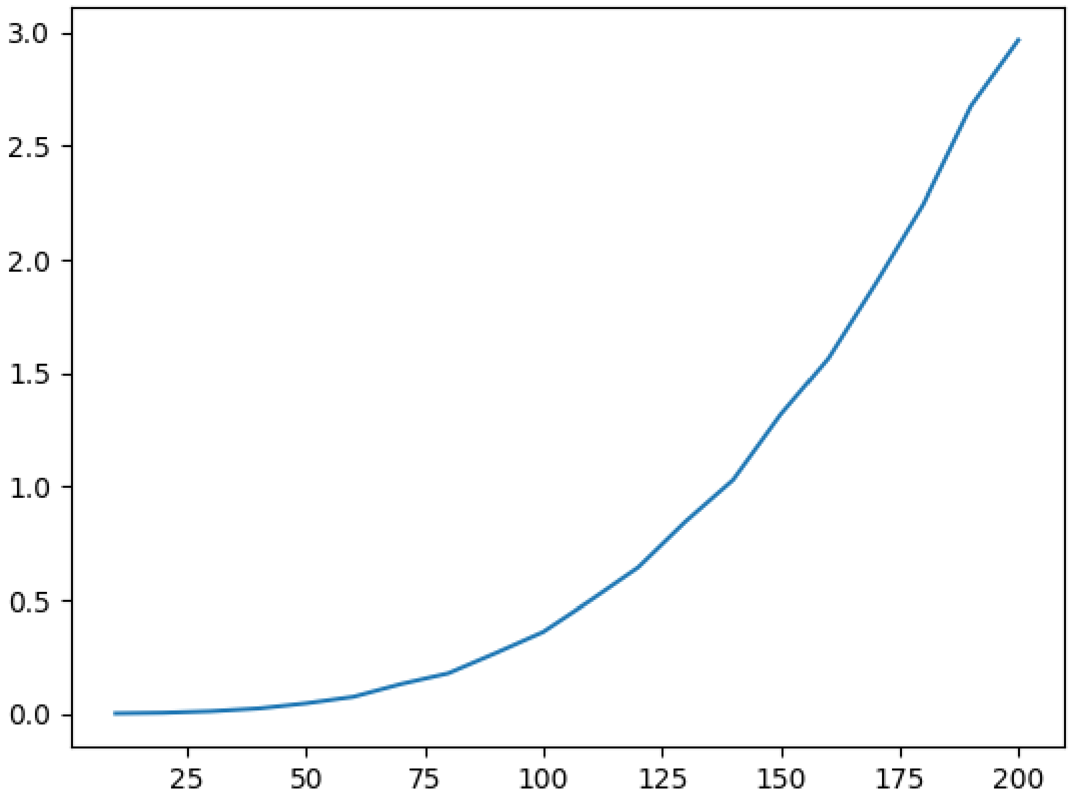
Program: mat_product4.py

In [1]:
```
def matrix_multiply(A, B):
    m, el, n = len(A), len(A[0]), len(B[0])
    C = [[sum([A[i][k] * B[k][j] for k in range(el)])
          for j in range(n)] for i in range(m)]
    return C

if __name__ == '__main__':
    from numpy.random import normal
    import matplotlib.pyplot as plt
    from time import time

    N = range(10, 210, 10)
    T = []
    for n in N:
        A = normal(0, 1, (n, n)).tolist()
        t0 = time()
        matrix_multiply(A, A)
        t1 = time()
        print(n, end=', ')
        T.append(t1 - t0)
    plt.plot(N, T)
```

```
10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170,
180, 190, 200,
```



Program: mat_product5.py

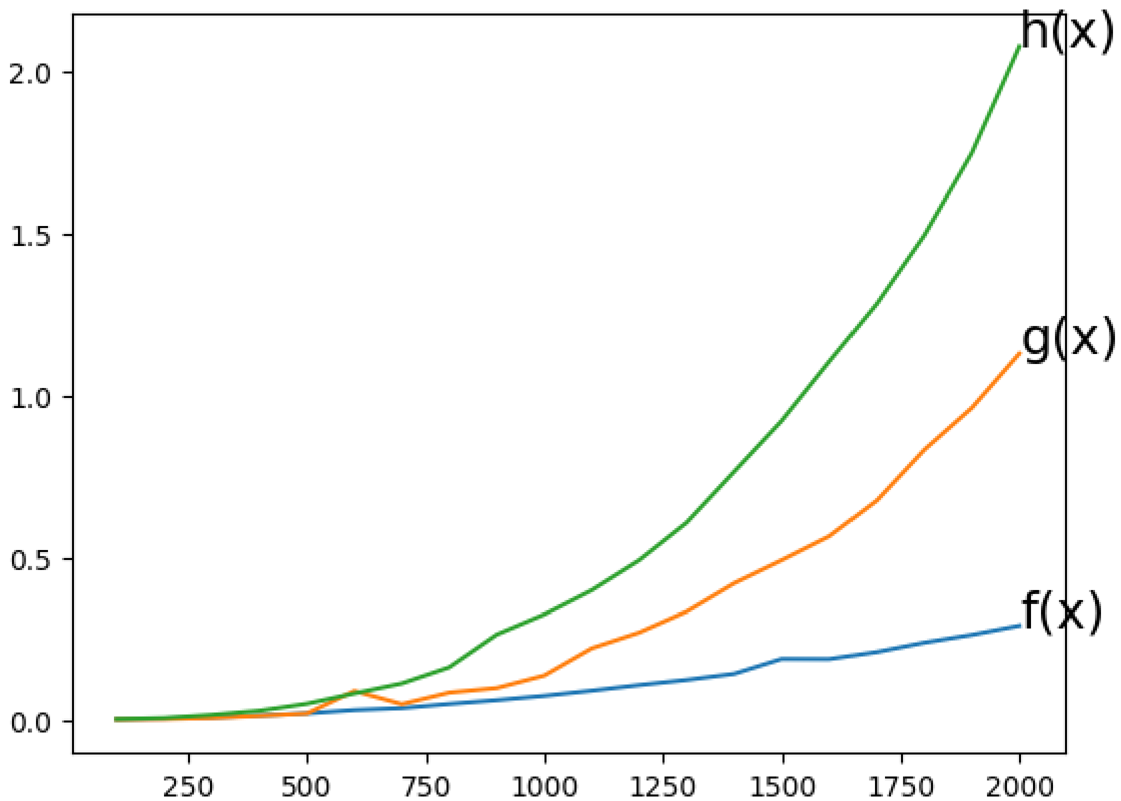
In [1]:
```
from numpy.random import normal
from numpy.linalg import inv
import matplotlib.pyplot as plt
from time import time
```

```
N = range(100, 2100, 100)
T = [[], [], []]

for n in N:
    t0 = time()
    A = normal(0, 1, (n, n))
    t1 = time()
    A.dot(A)
    t2 = time()
    inv(A)
    t3 = time()
    print(n, end=', ')
    t = (t0, t1, t2, t3)
    for i in range(3):
        T[i].append(t[i + 1] - t[i])

label = ['f(x)', 'g(x)', 'h(x)']
for i in range(3):
    plt.plot(N, T[i])
    plt.text(N[-1], T[i][-1], label[i],  fontsize=18)
```

```
100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400,
1500, 1600, 1700, 1800, 1900, 2000,
```



In [ ]: