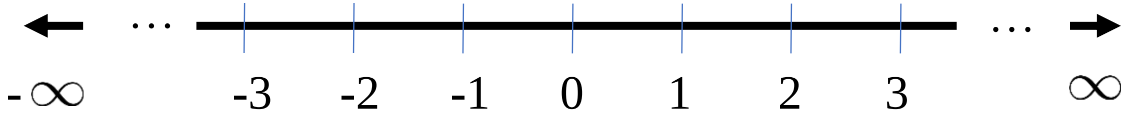
게임 수학 – 강의 9
복소수의 곱과 회전, 그리고 쿼터니언

동명대학교 게임공학과

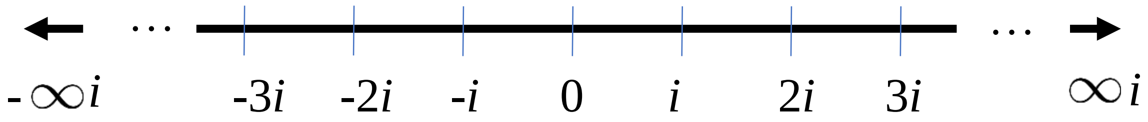
강영민

허수imaginary number

- 허수란?
  - 제곱해서 −1이 되는 수를 단위 i로 하는 수 체계
  - 실수 집합에서는 아무리 찾아도 제곱해서 음수가 되는 수는 찾을 수 없음
- 실수의 단위는 1, 허수의 단위는 i
- 실수는 수직선으로 표현 가능



- 허수도 수직선으로 표현 가능



복소수complex number

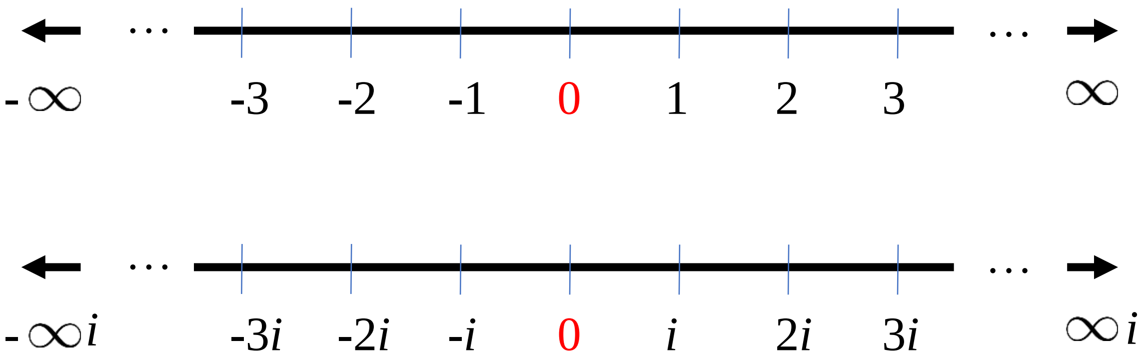
- 실수와 허수가 함께 만드는 수
  - 실수 + 허수

- 실수 + 실수 ➔ 실수 $\alpha + \beta = \gamma$
- 허수 + 허수 ➔ 허수 $\alpha i + \beta i = \gamma i$

- 실수와 허수의 덧셈 : $\alpha + \beta i = ?$
  - 실수 α와 허수의 크기 β는 서로 **연산이 되지 않음**
    - 더 줄일 수 없는 표현 : $\alpha + \beta i$ ← 복소수

복소수complex number는 어디에 있나?

- 실수와 허수 모두 수직선에 표시할 수 있음

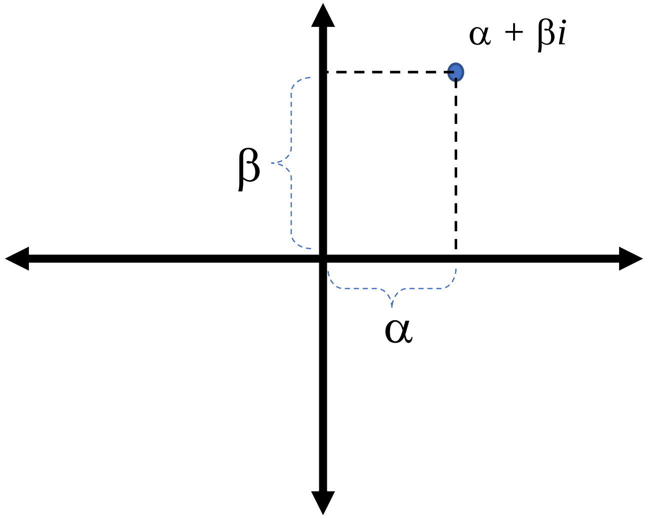


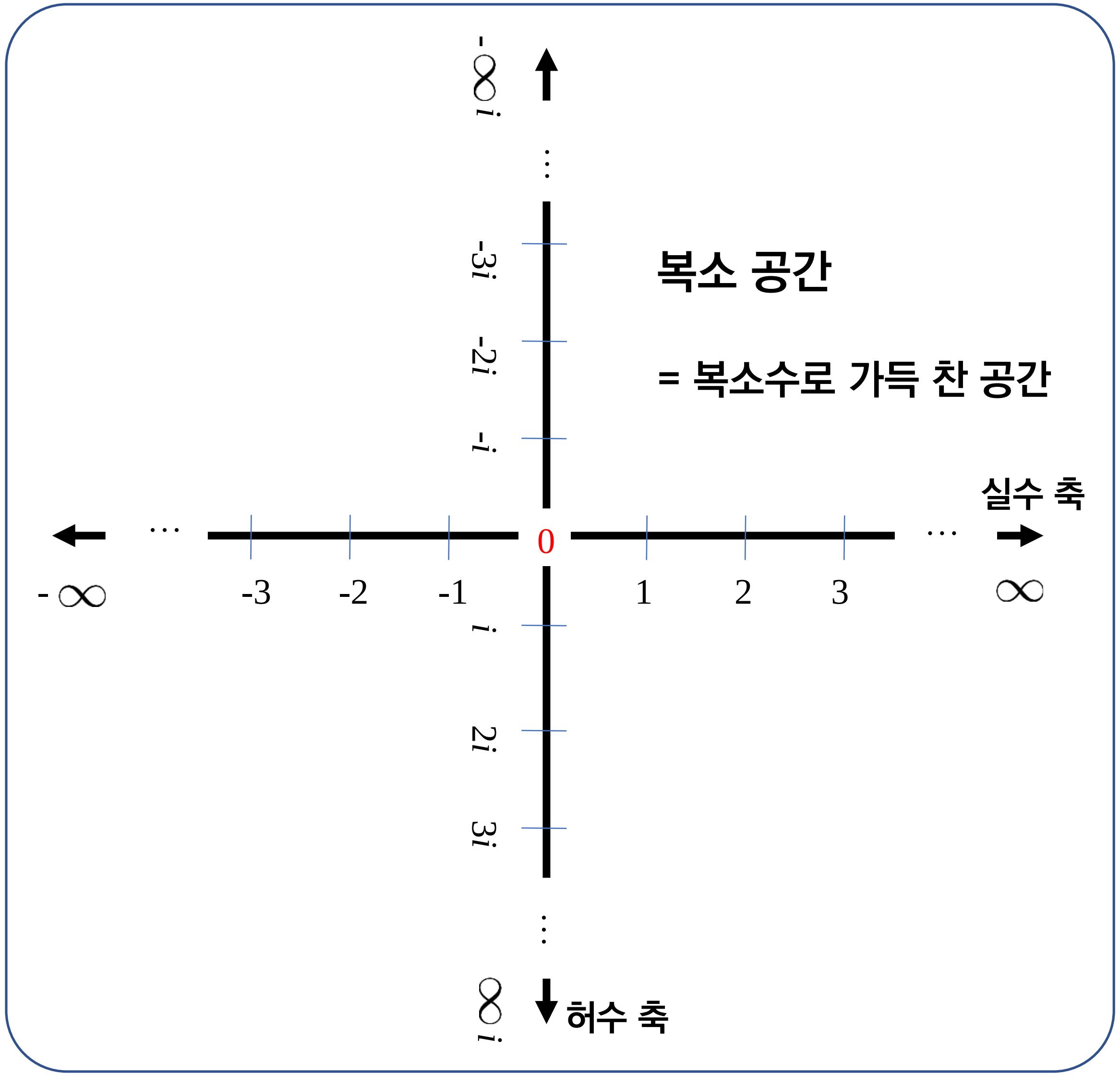
- 복소수는 어디에 있나?
  - 실수 수직선에도 없고, 허수 수직선에도 찍을 수 없음
  - 두 수직선의 밖에 있음
  - 두 수직선은 하나의 동일한 값을 가짐: 0

복소수complex number는 어디에 있나?

- 실수 수직선과 허수 수직선이 공유하는 값은 한 곳에
  - 실수는 가로축
  - 허수는 세로축
  - 실수와 허수도 아닌 어마어마하게 넓은 공간이 생성된다

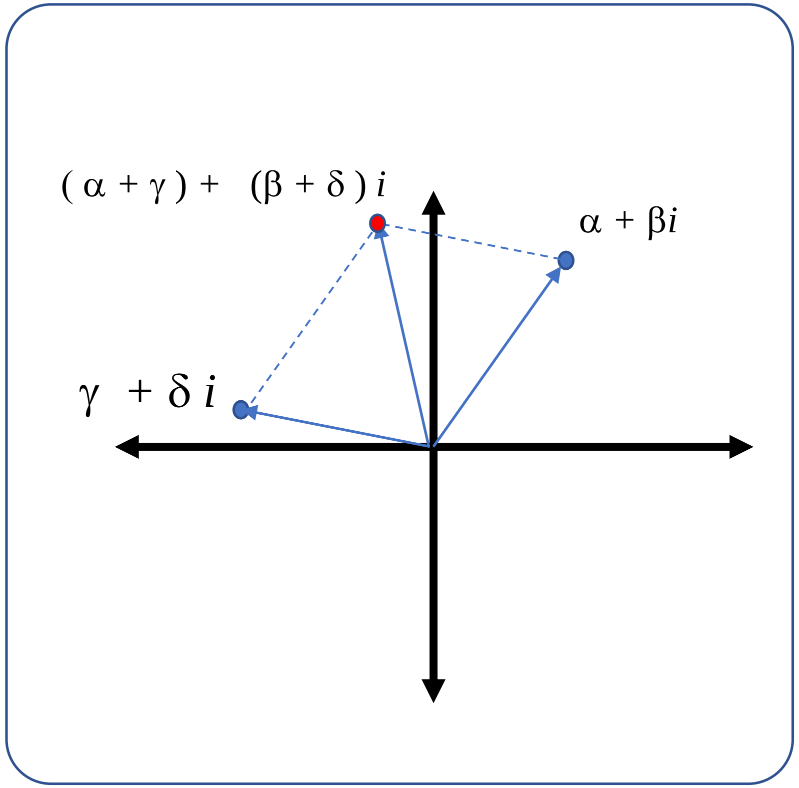
- (x, y) 좌표처럼 실수와 허수로 좌표 표현 가능
  - $(\alpha , \beta i)$
  - 이곳이 바로 $\alpha + \beta i$





복소수는 벡터와 비슷?

- 벡터처럼 덧셈과 뺄셈을 하면 각 축별로 이루어짐
  - 덧셈
    - 복소수 표현: $(\alpha + \beta i) + (\gamma + \delta i) = (\alpha + \gamma) + (\beta + \delta) i$
    - 벡터 표현 : $(\alpha, \beta) + (\gamma, \delta) = (\alpha + \gamma, \beta + \delta)$
  - 뺄셈
    - 복소수 표현: $(\alpha + \beta i) - (\gamma + \delta i) = (\alpha - \gamma) + (\beta - \delta) i$
    - 벡터 표현 : $(\alpha, \beta) - (\gamma, \delta) = (\alpha - \gamma, \beta - \delta)$



복소수의 곱셈은

- 복소수의 곱셈은?
  - 벡터는 스칼라 곱인 점곱과 벡터 곱인 가위곱이 존재
  - 복소수의 곱셈은
    - 실수와 허수는 곱셈이 가능 (허수를 실수배 하는 일)
    - 허수끼리 곱하면 음수
    - 두 복소수를 자연스럽게 곱할 수 있음
    - $(\alpha + \beta i)(\gamma + \delta i) = \alpha(\gamma + \delta i) + \beta i(\gamma + \delta i)$

$$= \alpha\gamma + \alpha\delta i + \beta\gamma i + \beta\delta i^2$$: 청색은 허수, 적색은 실수

$$= \boldsymbol{\alpha\gamma - \beta\delta} + \boldsymbol{(\alpha\delta + \beta\gamma) i}$$

복소수의 곱셈이 무슨 의미가 있을까?

- 복소수의 곱이 기하적으로 의미를 가질까?
  - 복소수 곱을 벡터의 변환으로 이해해 보자

- $(\alpha + \beta i)(\gamma + \delta i) = \alpha\gamma - \beta\delta + (\alpha\delta + \beta\gamma)i$

- $(\alpha + \beta i)(x + y i) = \alpha x - \beta y + (\alpha y + \beta x) i$
  $= x' + y' i$

$$x' = \alpha x - \beta y$$
$$y' = \beta x + \alpha y$$

선형변환

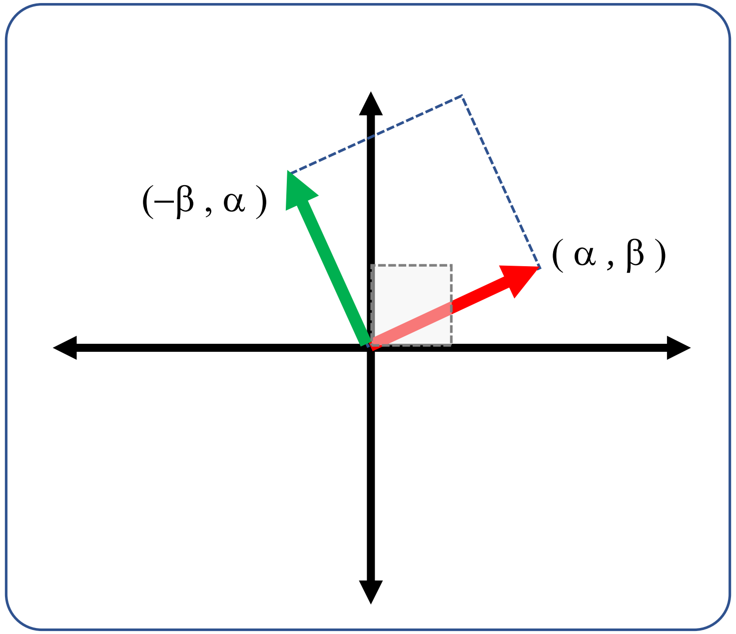
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$(\alpha + \beta i)$를 $(x + y i)$에 곱한다는 것은
복소공간의 점 (x, y) 에 행렬
$\begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix}$ 를 곱해서 변환하는 것과 같다

복소수의 곱셈은 어떤 변환을 하나

- 행렬을 그려보자

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \Leftrightarrow (\alpha + \beta i)(x + yi)$$

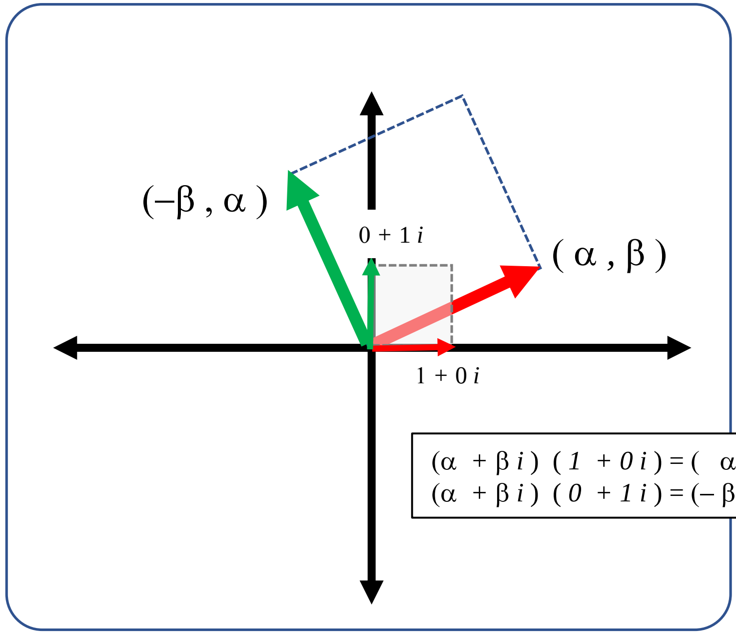


- 변환의 기하적 의미
  - 오른쪽 그림의 회색 사각형 = 축에 정렬된 단위 사각형
  - 이 단위 사각형이
    - 행렬의 1열 벡터를 표시하는 붉은 축과 2열 벡터인 녹색 축이 만드는 공간으로 옮겨짐
  - 이 붉은 색 축과 녹색 축은 언제나 직교 = 두 축의 내적은 언제나 0 $(-\alpha\beta + \alpha\beta)$
    - 공간이 크기만 커지거나 줄어들지 찌그러지지 않음
    - 축에 따라 크기가 바뀐 공간이 회전을 하게 됨

복소수의 곱셈은 어떤 변환을 하나

- 행렬을 그려보자

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \Leftrightarrow (\alpha + \beta i)(x + yi)$$

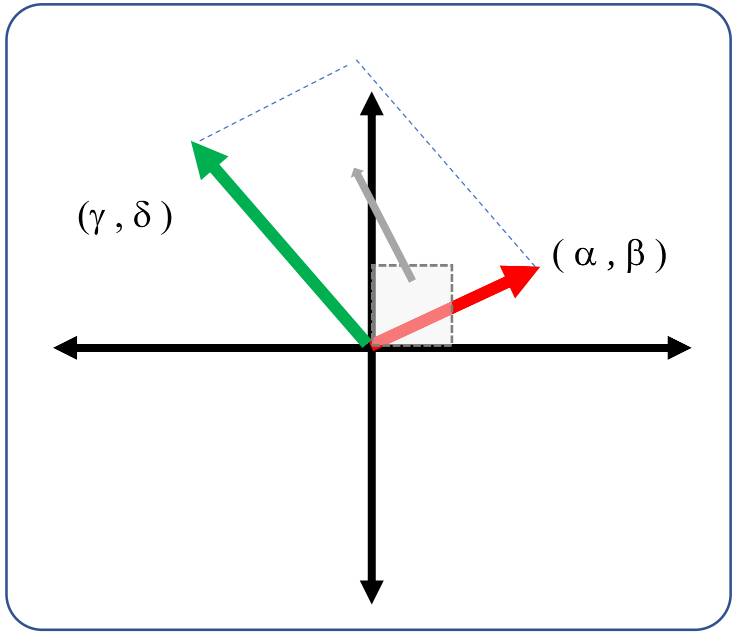


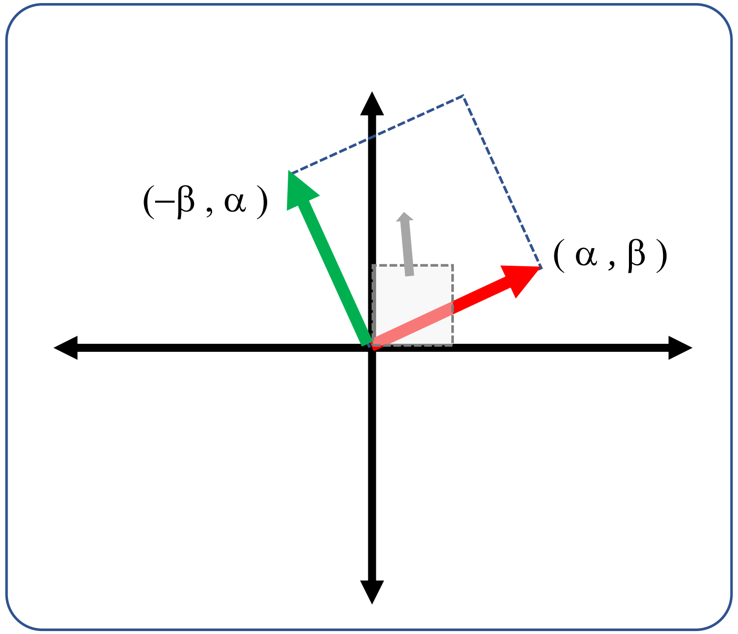
$(\alpha + \beta i)(1 + 0i) = (\alpha + \beta i)$
$(\alpha + \beta i)(0 + 1i) = (-\beta + \alpha i)$

행렬곱셈과 복소수 곱셈의 비교

$$\begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix} \times$$

$$(\alpha + \beta i) \times$$

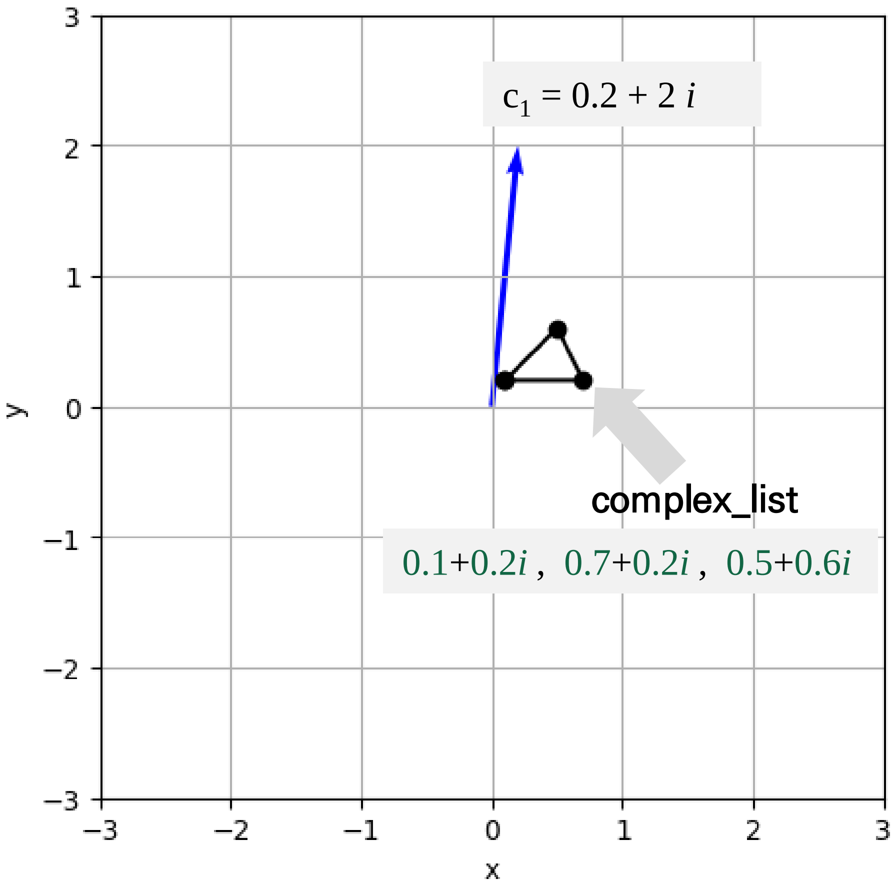


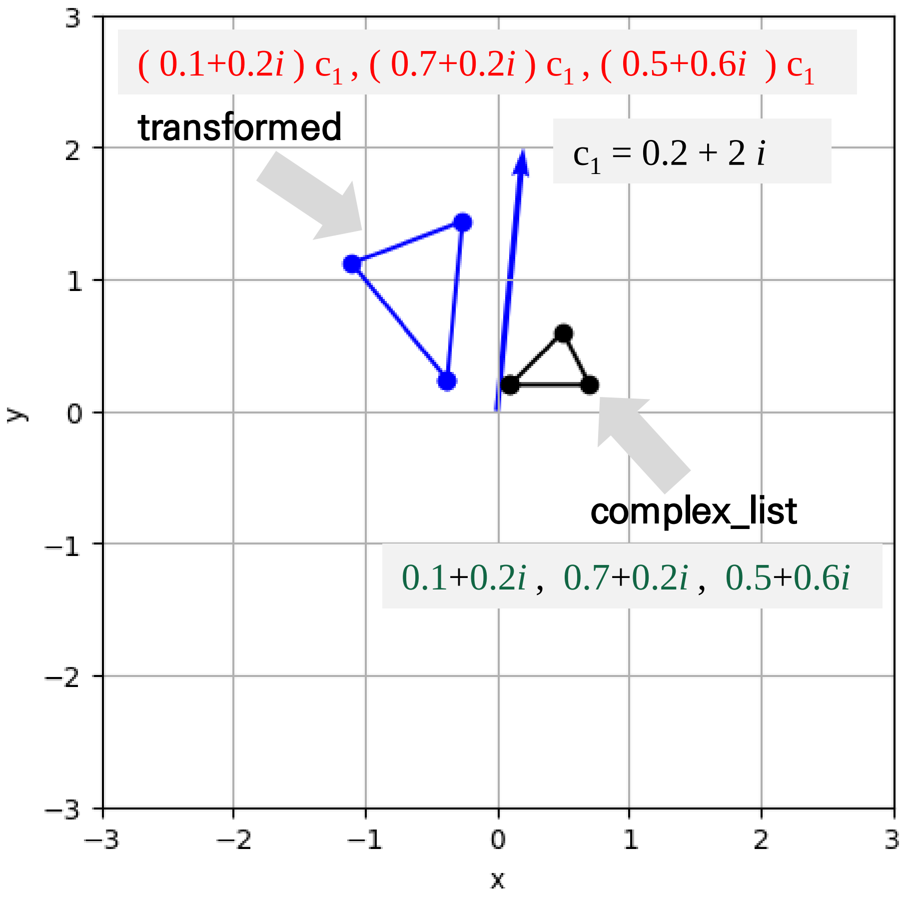


복소수를 곱하는 것은 복소평면 내의 변환이다.
행렬이 직교축을 임의의 축으로 옮겨 놓는다면,
복소수 곱하기는 회전과 스케일 둘만 적용되어
새로운 좌표축도 여전히 직교로 남게 변환된다.

```python
axis = prepareAxis2D(x=[-3, 3], y=[-3, 3])
c1 = 0.2 + 2j
complex_list = [ 0.1+0.2j, 0.7+0.2j, 0.5+0.6j]

visComplex(axis, c1)
drawComplex(axis, complex_list)
```

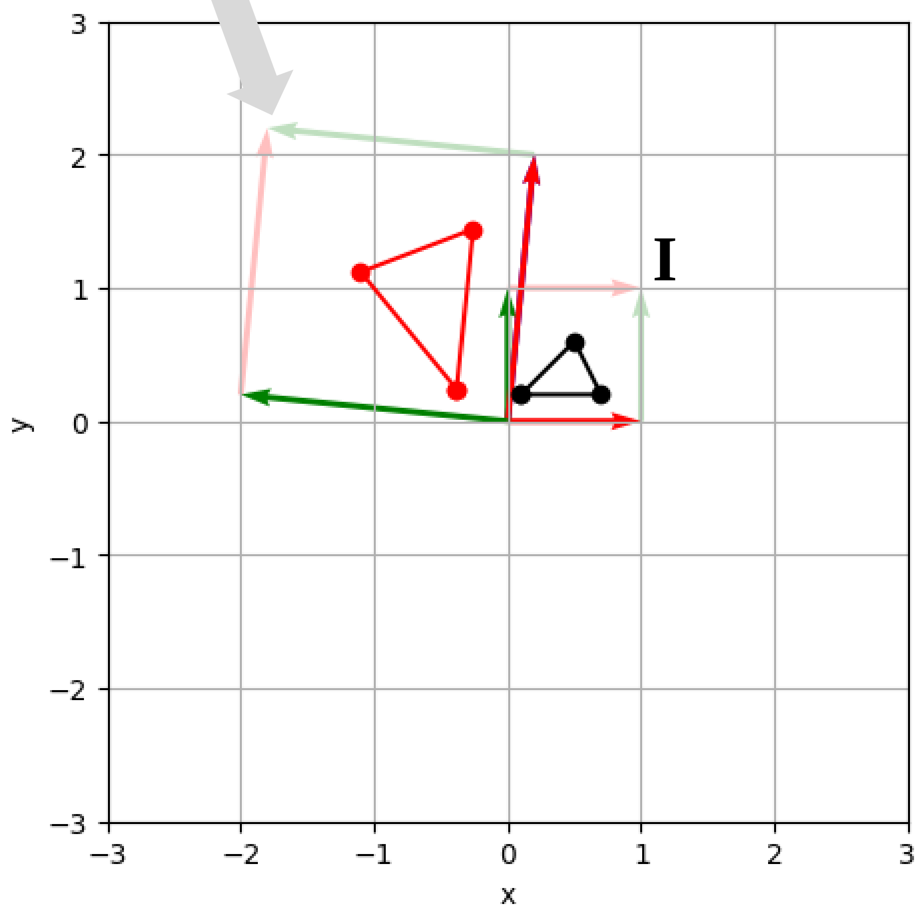
```python
axis = prepareAxis2D(x=[-3, 3], y=[-3, 3])
c1 = 0.2 + 2j
complex_list = [ 0.1+0.2j, 0.7+0.2j, 0.5+0.6j]
transformed = [ c1 * c for c in complex_list ]

visComplex(axis, c1)
drawComplex(axis, complex_list)
drawComplex(axis, transformed, color='red')
```





```
axis = prepareAxis2D(x=[-3, 3], y=[-3, 3])

c1 = 0.2 + 2j
complex_list = [ 0.1+0.2j, 0.7+0.2j, 0.5+0.6j]
transformed = [ c1 * c for c in complex_list ]

visComplex(axis, c1, color='blue')

# original space
visMat22(axis, np.eye(2))
drawComplex(axis, complex_list)

# transformed space
M = np.array([[c1.real, -c1.imag ],
              [c1.imag,  c1.real ]])
visMat22(axis, M)
drawComplex(axis, transformed, color='red')
```

$$\mathbf{M} = \begin{pmatrix} \text{c1.real} & \text{-c1.imag} \\ \text{c1.imag} & \text{c1.real} \end{pmatrix}$$

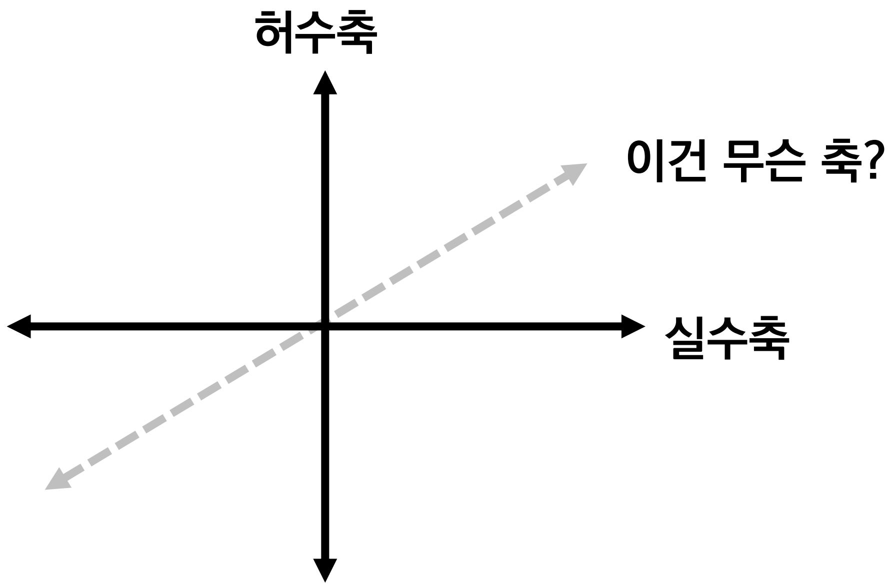


복소수 곱하기로 표현하는 회전 변환

- 회전 행렬

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad \Leftrightarrow \quad (\cos\theta + \sin\theta\, i)(x + y\, i)$$

3차원 공간에서는?

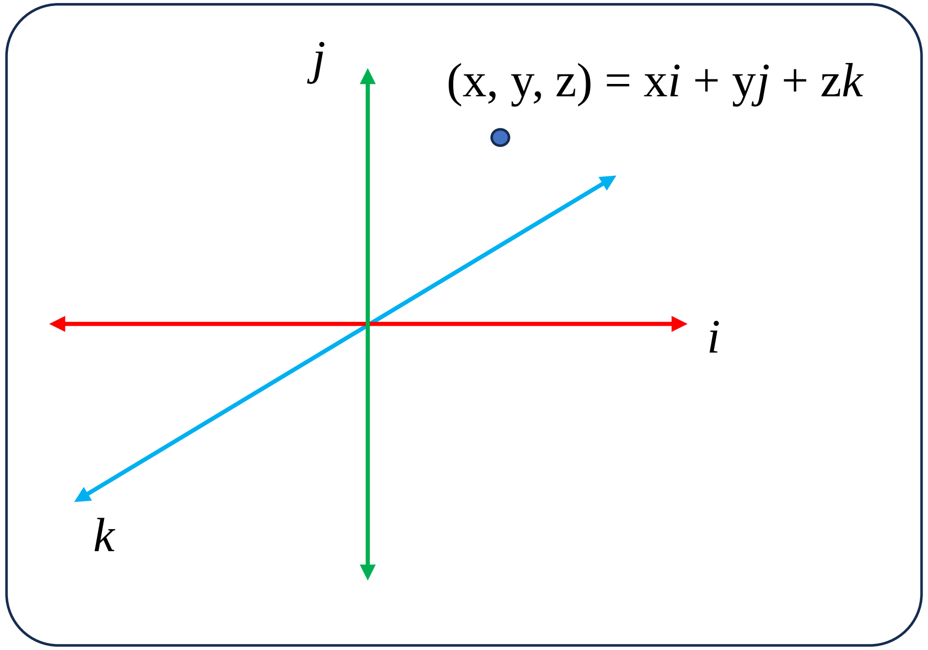


새로운 허수를 도입하자

- 2차원 복소 평면은
  - 두 개의 축 (실수, 허수)
    - 각각의 기저는 1과 i
    - 기저라는 것은 그 축의 모든 값이 이 기저를 늘이거나 줄인 값으로 표현할 수 있는 단위
  - 새로운 축이 필요하다면 i 말고 j라는 새로운 허수를 추가하면 어떨까?
- 모든 축을 허수로 다룬다면…
  - i, j, k 세 종류의 허수가 3차원 공간의 세 축의 기저가 되자

3차원 공간의 좌표

- 세 허수를 가진 복소수 표현
  - $(x, y, z) = xi + yj + zk$
- 실수는 사라진 것인가?
  - 3차원 공간 좌표를 표현하는 공간은 허수만 사용
  - 다른 용도로 실수축 사용
    - 벡터와 좌표의 구분 등
- 실수 + 허수 표현
  - 기저
    - $(1, (i, j, k))$: 1은 실수부의 기저, i, j, k는 허수축 각각의 기저



3차원 공간의 좌표를 곱할 수 있다?

- 복소수는 곱할 수 있다
  - 허수인 i, j, k의 제곱은 모두 −1
  - 실수와 허수의 곱은 허수 크기를 변경
  - 서로 다른 허수 사이의 곱은?
    - 다른 허수를 만든다

$ij = k$

$jk = i$

$ki = j$

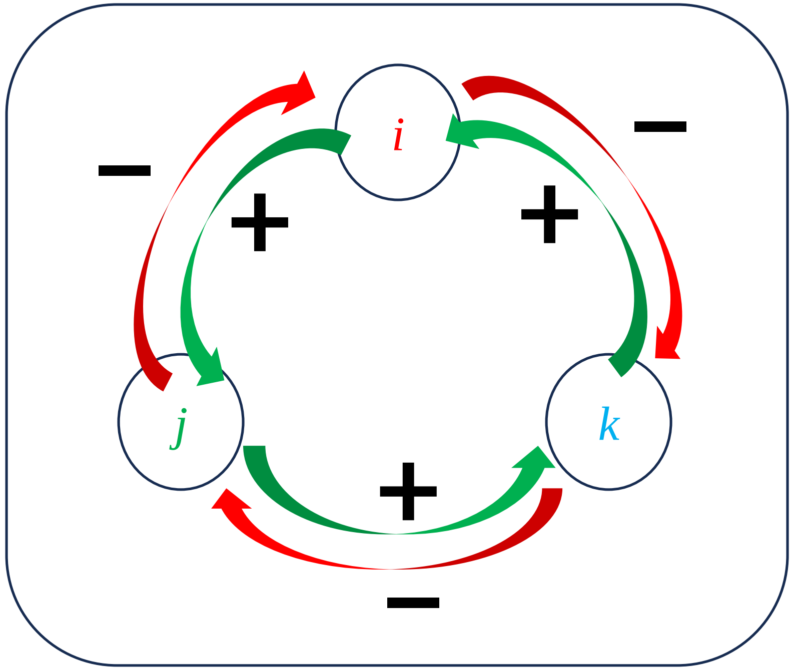
$ji = -k$

$kj = -i$

$ik = -j$

$$(x_1, y_1, z_1)\ (x_2, y_2, z_2)$$
$$= (x_1 i + y_1 j + z_1 k)\ (x_2 i + y_2 j + z_2 k)$$

이렇게 곱하면
무엇을 얻을까?



3차원 공간의 좌표를 곱할 수 있다?

$\mathbf{u} = (x_1, y_1, z_1)$
$\mathbf{v} = (x_2, y_2, z_2)$

이렇게 곱하면
무엇을 얻을까?

$$(x_1, y_1, z_1)\ (x_2, y_2, z_2) = (x_1 i + y_1 j + z_1 k)\ (x_2 i + y_2 j + z_2 k)$$

$$= x_1 i\,(x_2 i + y_2 j + z_2 k) + y_1 j\,(x_2 i + y_2 j + z_2 k) + z_1 k\,(x_2 i + y_2 j + z_2 k)$$
$$= x_1 i\, x_2 i + x_1 i\, y_2 j + x_1 i\, z_2 k + y_1 j\, x_2 i + y_1 j\, y_2 j + y_1 j\, z_2 k + z_1 k\, x_2 i + z_1 k\, y_2 j + z_1 k\, z_2 k$$
$$= -x_1 x_2 + x_1 y_2 k - x_1 z_2 j - y_1 x_2 k - y_1 y_2 + y_1 z_2 i + z_1 x_2 j - z_1 y_2 i - z_1 z_2$$
$$= -x_1 x_2 - y_1 y_2 - z_1 z_2 + y_1 z_2 i - z_1 y_2 i + z_1 x_2 j - x_1 z_2 j + x_1 y_2 k - y_1 x_2 k$$
$$= -x_1 x_2 - y_1 y_2 - z_1 z_2 + (y_1 z_2 - z_1 y_2) I + (z_1 x_2 - x_1 z_2) j + (x_1 y_2 - y_1 x_2) k$$
$$= -\mathbf{u} \cdot \mathbf{v} + \mathbf{u} \times \mathbf{v}$$

스칼라곱　　벡터곱

그런데 이 결과는 3개의 숫자로
표현된 원래의 좌표 방식으로 표현이 불가능
$\mathbf{u} \times \mathbf{v}$ 만 i, j, k 축 성분으로 표현 가능하고
$\mathbf{u} \cdot \mathbf{v}$ 는 실수축이 필요 ➔ 네 개의 기저 필요
$(1, (i, j, k))\ :\ (-\mathbf{u} \cdot \mathbf{v}, (\mathbf{u} \times \mathbf{v}))$

3차원 좌표를 곱하니 4개의 숫자가 필요

➔ 3차원 좌표는 4개의 숫자로 표현되는 수의 집합에서 실수 부분만 0인 부분집합

$\mathbf{u} = (x_1, y_1, z_1)$

$\mathbf{v} = (x_2, y_2, z_2)$

$\mathbf{u} = (0, x_1, y_1, z_1) = 0 + x_1 i + y_1 j + z_1 k$

$\mathbf{v} = (0, x_2, y_2, z_2) = 0 + x_2 i + y_2 j + z_2 k$

사원수(四元數)quaternion

사원수의 표현 방법: $(s, \mathbf{v})$

$s \in \mathbb{R}$

$\mathbf{v} \in \mathbb{R}^3$

쿼터니언 연산: 덧셈과 뺄셈

$$\mathbf{p} + \mathbf{q} = (s_p, \mathbf{v}_p) + (s_q, \mathbf{v}_q) = (s_p + s_q , \mathbf{v}_p + \mathbf{v}_q)$$

$$\mathbf{p} = (s_p, \mathbf{v}_p)$$
$$\mathbf{q} = (s_q, \mathbf{v}_q)$$

4차원 벡터로 이해하면 됨

$$\mathbf{p} - \mathbf{q} = (s_p, \mathbf{v}_p) - (s_q, \mathbf{v}_q) = (s_p - s_q , \mathbf{v}_p - \mathbf{v}_q)$$

쿼터니언의 곱셈: 일반적 경우

$$\mathbf{p} = (s_p, \mathbf{v}_p)$$
$$\mathbf{q} = (s_q, \mathbf{v}_q)$$

$$\begin{aligned}
\mathbf{p}\,\mathbf{q} &= (s_p + \mathbf{v}_p)(s_q + \mathbf{v}_q) \\
&= s_p s_q + s_p \mathbf{v}_q + s_q \mathbf{v}_p + \mathbf{v}_p \mathbf{v}_q \\
&= s_p s_q + s_p \mathbf{v}_q + s_q \mathbf{v}_p + (-\mathbf{v}_p \bullet \mathbf{v}_q + \mathbf{v}_p \times \mathbf{v}_q) \\
&= s_p s_q - \mathbf{v}_p \bullet \mathbf{v}_q + s_p \mathbf{v}_q + s_q \mathbf{v}_p + \mathbf{v}_p \times \mathbf{v}_q
\end{aligned}$$

$$\begin{aligned}
\mathbf{p}\,\mathbf{q} &= (s_p, \mathbf{v}_p)(s_q, \mathbf{v}_q) \\
&= (s_p s_q - \mathbf{v}_p \bullet \mathbf{v}_q,\; s_p \mathbf{v}_q + s_q \mathbf{v}_p + \mathbf{v}_p \times \mathbf{v}_q)
\end{aligned}$$

쿼터니언의 곱셈: 스칼라 원소가 0이면

$$\mathbf{p} = (0, \mathbf{v}_p)$$
$$\mathbf{q} = (0, \mathbf{v}_q)$$

$$\begin{aligned}\mathbf{p}\,\mathbf{q} &= (0 + \mathbf{v}_p)\,(0 + \mathbf{v}_q) \\ &= \mathbf{v}_p\,\mathbf{v}_q \\ &= (-\mathbf{v}_p \bullet \mathbf{v}_q + \mathbf{v}_p \times \mathbf{v}_q) \\ &= -\mathbf{v}_p \bullet \mathbf{v}_q + \mathbf{v}_p \times \mathbf{v}_q\end{aligned}$$

$$\begin{aligned}\mathbf{p}\,\mathbf{q} &= (0\,,\, \mathbf{v}_p)\,(0\,,\, \mathbf{v}_q) \\ &= (-\mathbf{v}_p \bullet \mathbf{v}_q\,,\, \mathbf{v}_p \times \mathbf{v}_q)\end{aligned}$$