Complex Numbers in CS
Some Applications
Quaternions and Graphics

What are quaternions?

- This approach is not so much an application of Complex Numbers *per se* but rather a mechanism arising by *extending* some *geometric features* of the *Complex Plane* and *2-vectors*.
- In particular from the question "*How do we extend the idea of the Complex Plane to 3-dimensional settings*?"
- The answer may seem somewhat surprising.
- By considering a "*vector algebra*" with *four* components.
- This solution provides *very effective* computation in the case of *3-dimensional Computer Graphics* effects.

Quaternion Structure

- One form for Complex Numbers was through $z = x + iy$ giving us 2-vectors $\langle x, y\rangle$ in the Complex Plane and a convention $i^2 = -1$. Both x and y were Real numbers.
- Quaternions use points in R^4 of the form $q = (w, x, y, z)$.
- Matching $z = x + iy$ we now have

$$q = w + ix + jy + kz$$

- where

$$i^2 = j^2 = k^2 = -1$$
$$jk = -kj = i$$
$$ki = -ik = j$$
$$ij = -ji = k$$

Quaternion Notational Conventions

- H : the set of all quaternions;
- $q \in H$ an arbitrary quaternion.
- The forms $q = w + ix + jy + kz$ and $q = [\alpha , \underline{v}]$ are both used.
- In the second of these: $w \equiv \alpha \in R$; $\underline{v} = \langle x, y, z \rangle \in R^3$
- The notation, H, is in honour of William Hamilton (1805-65) who is (usually) credited with the development of Quaternion Algebra (discovered 1843, published 1848).
- [Although Rodrigues (in 1840) and Gauss (1819, unpublished) had already developed similar methods.]

Quaternion Operations

- Addition: Add *corresponding components*.
- Scalar Multiple by $\varepsilon \in R$: *multiply individual components* by ε.
- Conjugate: if $q = (w, x, y, z)$: $\bar{q} = (w, -x, -y, -z)$; *equivalently* if $q = \left[\alpha, \underline{v}\right]$: $\bar{q} = \left[\alpha, -\underline{v}\right]$.
- Modulus (size): $\|(w, x, y, z)\| \stackrel{\text{def}}{=} \sqrt{w^2 + x^2 + y^2 + z^2}$

  equivalently: $\|[\alpha, \underline{v}]\| \stackrel{\text{def}}{=} \sqrt{\alpha^2 + \|\underline{v}\|^2}$
- If $\|q\| = 1$, q is called a *unit quaternion*.
- Division: $q^{-1} = \dfrac{\bar{q}}{\|q\|^2}$

Quaternion Products

- If $s, t \in H$, how do we form the result $s \cdot t$ of *multiplying* these two quaternions, ie how is *quaternion product* defined?
- We *could* expand $(w + ix + jy + kz) \cdot (a + ib + jc + kd)$ (similar to Complex case): but this is "*messy*" and its *geometric form* in 3-dimensions is *opaque*. Writing $s = [\alpha, \underline{u}]\,; t = [\beta, \underline{v}]$ we define $s \cdot t$ as

$$[\alpha\beta - \underline{u} \cdot \underline{v}, \alpha\underline{v} + \beta\underline{u} + \underline{u} \times \underline{v}]$$

- $\underline{u} \cdot \underline{v}$ is *vector scalar* (dot) *product* (textbook pp 69–71)
- $\underline{u} \times \underline{v}$ is *3-vector cross product* (textbook pp 72–75, 90)

Quaternion Product is not commutative ($s \cdot t \neq t \cdot s$)

Quaternions and 3-D Graphics

- Earlier we met the idea of using matrix-vector products to realize graphical effects.
- In 2 dimensions this works "*reasonably well*".
- In 3 dimensions *rotational effects* become ***VERY*** *awkward* and *computationally costly* using matrix-vector products *directly*.
- One reason is because of the phenomenon called *Gimbal Lock*.
- The matrices used are defined in terms of *trigonometric functions* but it can happen that these have *undefined values* when *combining rotations* about *some angles*.
- As a result graphical simulation "*freezes*".
- *Quaternions* offer a *solution*: these *cannot* lead to *Gimbal Lock*.

Rotation about an axis using Quaternions

- Suppose we are looking to rotate a point (x, y, z) about some *axis of rotation* (ie infinite line in 3-dimensions) $\underline{v} = \langle \alpha, \beta, \gamma \rangle$.
- We wish to rotate through some angle $\theta°$.
- Choose the quaternion $q_\theta = [\cos(\theta/_2), \sin(\theta/_2)\, \underline{v}]$
- Since we are dealing with a line we can *always* choose $\underline{v}$ so that q_θ is a *unit quaternion*.
- The computation "*rotate* $\underline{w} = \langle x, y, z \rangle$ *by* $\theta°$ *about* $\underline{v}$" is just the quaternion product:

$$q_\theta \cdot [0, \underline{w}] \cdot q_\theta^{-1} = q_\theta^{-1} \cdot [0, \underline{w}] \cdot q_\theta$$

Quaternions – Summary

- Although *not directly* using *Complex Numbers*, the form of *Quaternion Algebra* would, arguably, *not have been discovered independently* of *Complex Algebra*.
- Rotation in 3-dimensions is just *one* application in *Graphics*.
- Quaternion *Graphics Engines* underpin many *popular video games*, eg *Tomb Raider*.
- Other uses will be seen in COMP222 – Principles of Computer Game Design next year.
- Other ideas from *Complex Numbers* (eg "*fractal art*" which we look at later in this section) extend to the *Quaternion domain*.
- Quaternions are *not* the "*end of process*": an *8-component form* operating in 7-dimensional space – the *Octonions* – has proved to be of importance in *Advanced Physics*.