SHIFT YOUR PARADIGM

ゲームプログラマのための数学の歩き方 – デュアルクォータニオン編

鍛冶静雄(九州大学)
落合啓之(九州大学)
長谷川勇(スクウェア・エニックス)
Rodolphe Vaillant (スクウェア・エニックス)

九州大学マス・フォア・インダストリ研究所(IMI)

Institute of Mathematics for Industry
Kyushu University

2011年 アジア初の産業数学の研究所として創設

CG・ゲームとの関わり

- 研究集会
  - デジタル映像表現のための数理的手法(MEIS2014--2018)
  - CG技術の実装と数理2017 など
- CEDEC x IMI コラボセッション
  - 大規模グラフ解析と都市 OS の開発（藤澤克樹, 2017年）
  - 暗号の安全性はどのように評価するのか（高木剛, 2016年）
  - データ解析における学習理論と統計学の特徴と生かし方　（西井龍映, 2015年）
  - CGにおける運動や変形の記述とその数理（落合啓之, 2014年）
  - 行列の数理と運動の記述（落合啓之, 2013年）
- スクエニ x IMI @ CEDEC
  - ゲームプログラマのための数学の歩き方
- ラプラシアン編　（長谷川勇, 太田友, 2020年）
- クォータニオンとリー群編　（長谷川勇, 朴炯基, 2021年）
- デュアルクォータニオン編（長谷川勇, 鍛冶静雄, 落合啓之, 2021年）

活動

- ワークショップ
- チュートリアル
- 技術相談 (AIMaP)
- 共同研究
- インターンシップ
- 国際交流 (APCMfI)
- IMI宣言2021

Outline

- 導入
  - 「ゲームプログラマのための数学の歩き方」でやりたいこと
  - デュアルクォータニオンの導入・利用例（DQS、自動微分）
- 剛体変換
- デュアル実数・デュアル複素数・デュアルクォータニオン

SHIFT YOUR PARADIGM

解決したい課題：数学の独学の難しさ

次はラプラス作用素を調べると

- 増えていくワード
  - 多様体
  - テンソル
  - 外微分
  - ホッジ双対
  - …

一般化 [編集]

ラプラス＝ベルトラミ作用素 [編集]

詳細は「ラプラス＝ベルトラミ作用素 (英語版)」を参照

ラプラス作用素の概念は、リーマン多様体上で定義されたラプラス＝ベルトラ
ることができる。同様にダランベール作用素は擬リーマン多様体上の双曲型
函数に適用すれば、その函数のヘッセ行列のトレース

$$\Delta f = \operatorname{tr}(H(f))$$

が得られる。ただし、トレースは計量テンソルの逆に関して取るものとする。
に作用する作用素（これもまたラプラス＝ベルトラミ作用素と呼ばれる）に

ラプラス作用素の別な一般化として、擬リーマン多様体上で定義される外微

$$\Delta f = d^* df$$

を考えることもできる。ここで d^* は余微分 (英語版) で、ホッジ双対を使って
ラシアン」とは異なるものであることには注意すべきである。そのことは大
い。より一般に、微分形式に対して定義される「ホッジ」ラプラシアン α は

$$\Delta\alpha = d^* d\alpha + dd^* \alpha$$

と書ける。これはまたラプラス＝ドラーム作用素 (英語版) とも呼ばれ、ヴァ
トラミ作用素と関係する。

ダランベール作用素 [編集]

ラプラシアンを適当な仕方によって非ユークリッド空間に一般化することが
どが可能である。

出典: https://ja.wikip

ホッジ双対？

数学において、**ホッジスター作用素**(ホッジスターさようそ、Hodge star operator)、もしくは、**ホッジ双対**(ホッジそうつい、Hodge dual)は、ウィリアム・ホッジにより導入された線型写像である。ホッジ双対は、有限次元の向き付けられた内積空間の外積代数の上で定義されるk -ベクトルのなす空間からn-k-ベクトルのなす空間への線形同型である。

他のベクトル空間に対する多くの構成と同様に、ホッジスター作用素は多様体の上のベクトルバンドルへの作用に拡張することができる。たとえば余接束の外積代数（すなわち、多様体上の微分形式の空間）に対して、ホッジスター作用素を用いてラプラス＝ド・ラーム作用素を定義し、コンパクトなリーマン多様体上の微分形式のホッジ分解を導くことができる。

- 外積代数
- ベクトルバンドル
- 微分形式
- …

出典: https://ja.wikipedia.org/wiki/ホッジ双対

CEDEC2020

SHIFT YOUR PARADIGM

引用: https://cedec.cesa.or.jp/2020/session/detail/s5e79a3e9c953a.html
CEDEC2020「ゲームプログラマのための数学の歩き方 - ラプラシアン編」

解決したい課題: 数学の行間

行間を読むのが難しい(球面調和関数)

調和多項式 p が k 次の斉次多項式であるとき、p を単位球面

$$S^{n-1} = \{(x_1, \ldots, x_n) \in \mathbf{R}^n \mid \sum_i {x_i}^2 = 1\}$$

に制限した制限写像

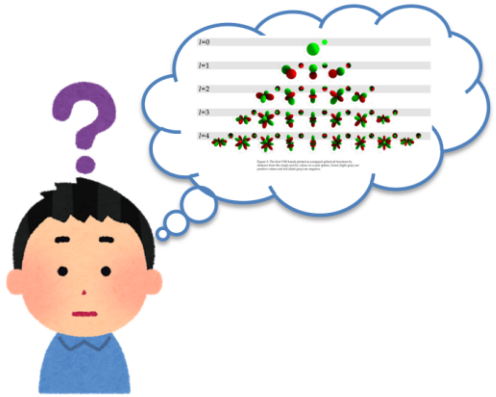
$$Y = p|_{S^{n-1}} : S^{n-1} \to \mathbf{R}$$

を k 次の**球面調和関数**という[5]。

3次元空間における球面調和関数 [編集]

3次元空間 $\mathbf{R}^3$ の場合、$\mathbf{R}^3$ を球面座標 (r, θ, φ) で表すと、下記の $Y_k^m(\theta, \varphi)$ が球面調和関数になる事が知られている。

$$Y_k^m(\theta, \phi) = (-1)^{(m+|m|)/2} \sqrt{\frac{2k+1}{4\pi} \frac{(k-|m|)!}{(k+|m|)!}} P_k^{|m|}(\cos\theta)\, e^{im\phi}. \quad \textbf{(B1)}$$



出典: https://ja.wikipedia.org/wiki/球面調和関数

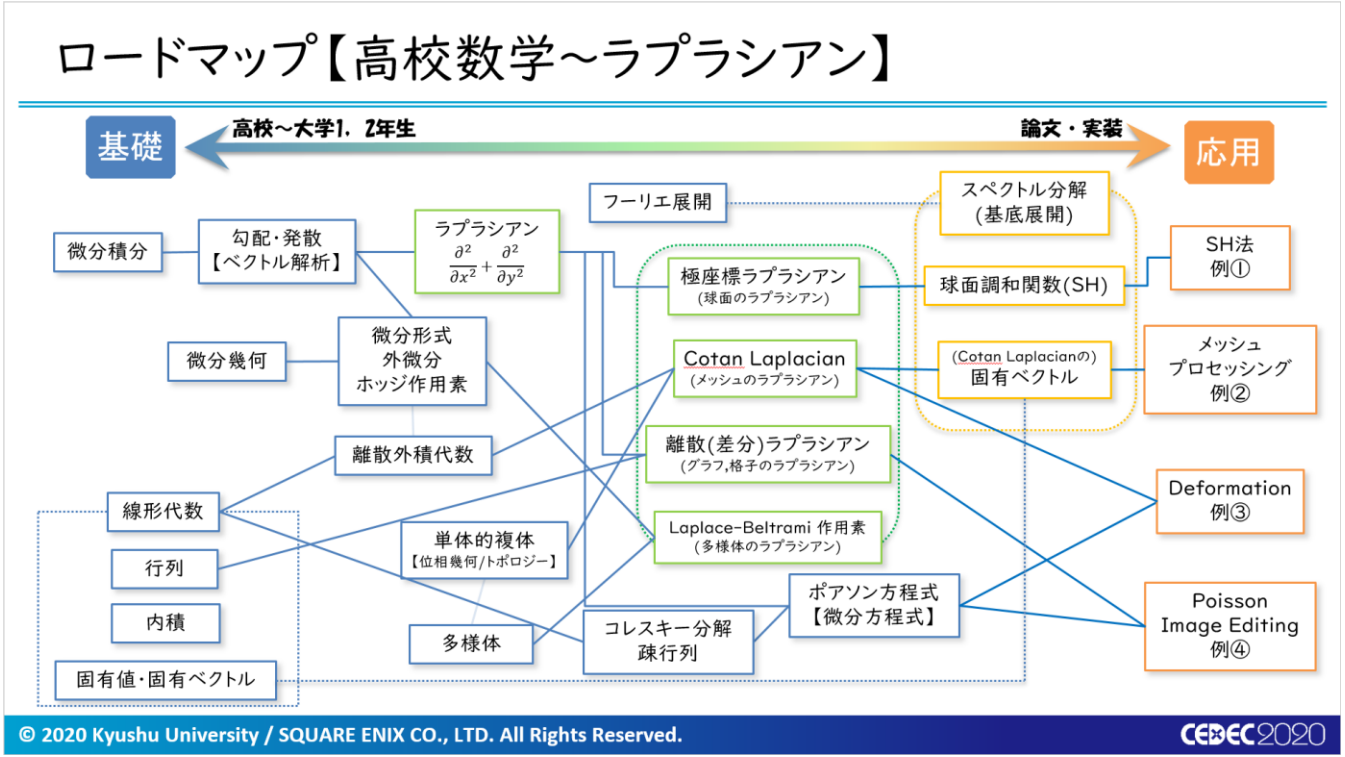
CEDEC2020

SHIFT YOUR PARADIGM

引用: https://cedec.cesa.or.jp/2020/session/detail/s5e79a3e9c953a.html
CEDEC2020「ゲームプログラマのための数学の歩き方 - ラプラシアン編」

CEDEC2021

解決策: ロードマップ



引用: https://cedec.cesa.or.jp/2020/session/detail/s5e79a3e9c953a.html
CEDEC2020「ゲームプログラマのための数学の歩き方 - ラプラシアン編」

解決策: ラプラシアンの「心」

ラプラシアンとは？

【答えの一例】

いろいろな場所(定義域)で
フーリエ展開をするために必要なもの

(今回の講演では)

⇩

ラプラシアンの「心」

引用: https://cedec.cesa.or.jp/2020/session/detail/s5e79a3e9c953a.html
CEDEC2020「ゲームプログラマのための数学の歩き方 - ラプラシアン編」

クォータニオンの導入

- クォータニオンは次で定義される

$$q = q_0 + q_1 i + q_2 j + q_3 k \ (q_0, q_1, q_2, q_3 \text{は実数})$$
$$i^2 = j^2 = k^2 = ijk = -1$$

$\Leftrightarrow$

(x軸周りの)回転行列

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

- 3次元座標の任意軸回転を表せる
  - 3次元ベクトル(p_1, p_2, p_3) を$p = p_1 i + p_2 j + p_3 k$
  - 長さ1の(w_1, w_2, w_3) を回転軸とする角度θの回転

$$q = \cos\frac{\theta}{2} + (w_1 i + w_2 j + w_3 k)\sin\frac{\theta}{2}$$

  - pをqで回転するには $p' = qpq^{-1}$

$\Leftrightarrow$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

SHIFT YOUR PARADIGM

クォータニオンの導入(プログラマ観点)

- 3次元座標の任意軸回転
- クォータニオンによる回転(DirectXMath)

```
XMVECTOR q = XMQuaternionRotationRollPitchYaw(pitch, yaw, roll);
XMVECTOR vout = XMVector3Rotate(vin, q);
```

  - 回転行列と同じように使える

```
XMMATRIX mat = XMMatrixRotationRollPitchYaw(pitch, yaw, roll);
XMVECTOR vout = XMVector3Transform(vin, mat);
…
XMVECTOR q = XMQuaternionRotationMatrix(mat); // 回転行列から変換
XMMATRIX m = XMMatrixRotationQuaternion(q);    // 回転行列へ変換
```

SHIFT YOUR PARADIGM

クォータニオンの導入(メリット)

- データサイズが小さい
  - 回転行列: 3x3(あるいは4x4)行列
  - クォータニオン: Vector4
- (回転操作自体の)計算量が少ない
- (オイラー角の)ジンバルロックがない
- 球面線形補間が簡単

```
XMVECTOR q = XMQuaternionSlerp(q0, q1, t);
```

SHIFT YOUR PARADIGM

デュアルクォータニオンの導入

- クォータニオン
  - 回転行列の便利な代替
  - Rigid transform(剛体変換)の代替ではない
- クォータニオンと統一してscale, translateも扱いたい(rigid transformの代替にしたい)
  ⇒ デュアルクォータニオン

SHIFT YOUR PARADIGM

デュアルクォータニオンの導入

- デュアルクォータニオンは次で定義

$$\hat{q} = q_0 + q_1\epsilon \ (q_0, q_1 \text{はクォータニオン})$$
$$\epsilon^2 = 0$$

- 回転を表す: $\hat{q}_R = q_R + 0\epsilon$
- 移動を表す: $\hat{q}_T = 1 + \frac{1}{2} q_T \epsilon$

- なぜ今度はϵ^2は-1でなく0？
  ⇒ 後半で

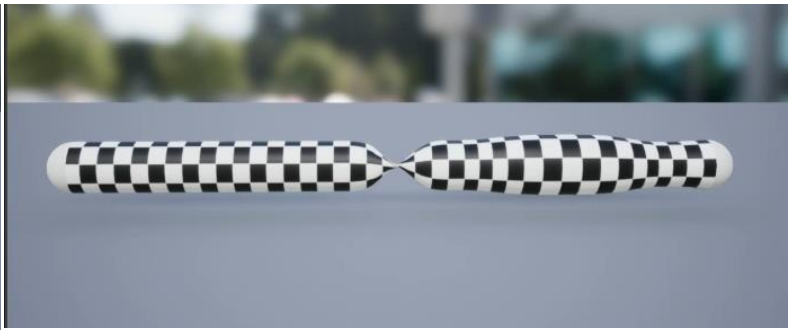
SHIFT YOUR PARADIGM

アニメーションでの利用

- Kavan et al.
  “Skinning with Dual Quaternions” I3D 2007
  - “candy-wrapper artifact”を解決
  - Linear Blend Skinningの1.5倍程度のコスト

Dual Quaternion Skinning

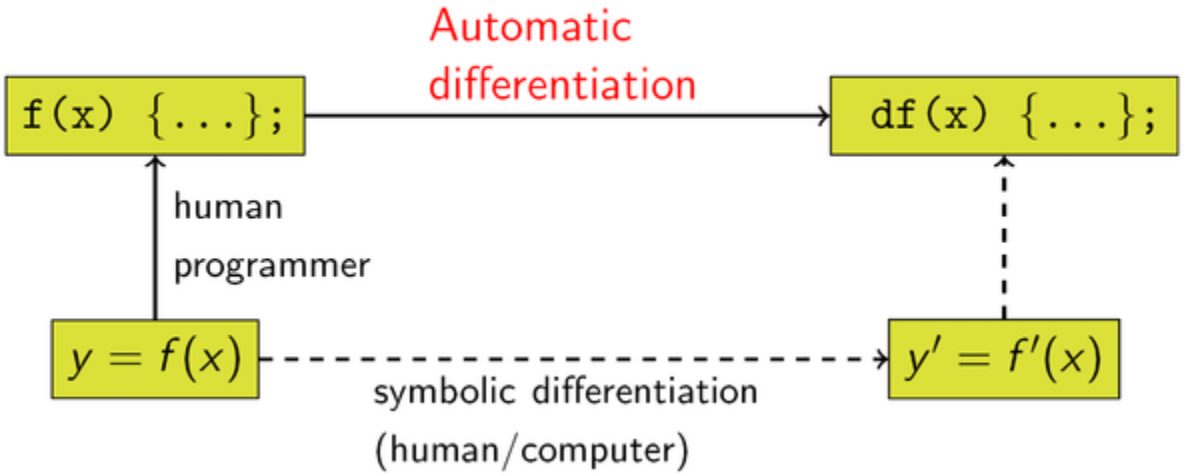
Linear Blend Skinning



SHIFT YOUR PARADIGM

自動微分

- プログラムで定義された関数の偏導関数の値を計算するプログラムを導出する技術

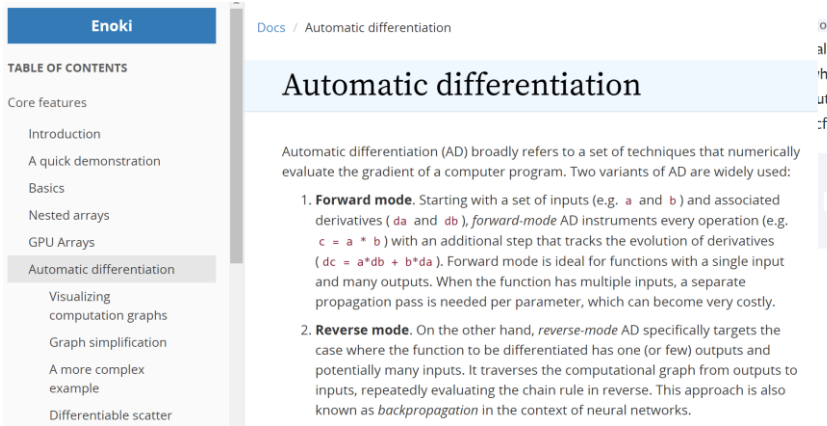


https://ja.wikipedia.org/wiki/自動微分

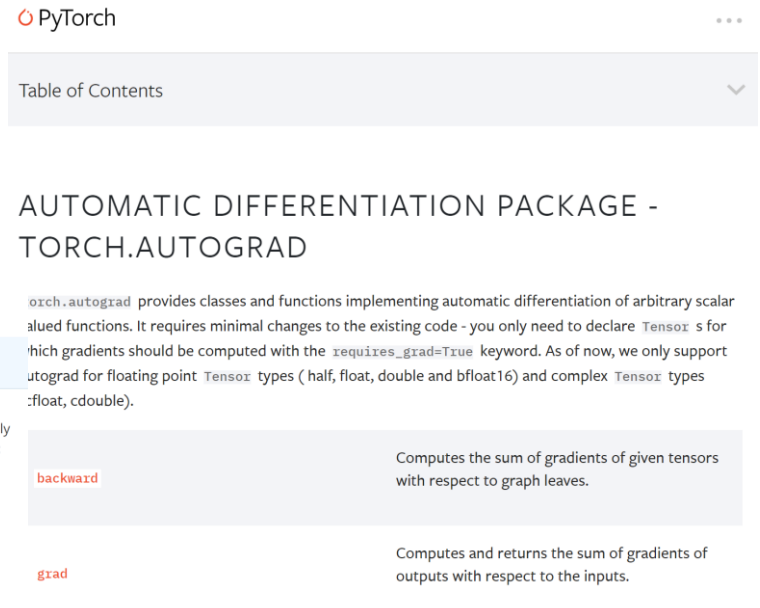
SHIFT YOUR PARADIGM

自動微分の利用例

- ゲーム開発と関連する利用例
  - 微分可能レンダラ
  - 機械学習
  - …



https://enoki.readthedocs.io/en/master/autodiff.html



https://pytorch.org/docs/stable/autograd.html

SHIFT YOUR PARADIGM

自動微分での(デュアル実数の)利用

二重数を用いた自動微分 [編集]

フォワードモード自動微分は実数の代数に（元を）添加して新しい算術を導入することによって可能である。全ての数（通常の実数）に対して、その数における関数の微分を表現する追加の成分が足され、全ての算術演算がこの添加代数に拡張される。すなわち二重数の代数である。このアプローチはプログラミング空間上の演算子法（英語版）の理論（つまり双対空間のテンソル代数）によって一般化される（解析的プログラミング空間（英語版）を見よ）。

各数 x を数 $x + x'\varepsilon$ に置き換える。ここで x' は実数だが ε は $\varepsilon^2 = 0$ を満たす抽象的数（英語版）である（無限小；滑らかな無限小解析も参照）。ちょうどこれだけを用いて通常の演算が得られる：

$$(x + x'\varepsilon) + (y + y'\varepsilon) = x + y + (x' + y')\varepsilon$$
$$(x + x'\varepsilon) \cdot (y + y'\varepsilon) = xy + xy'\varepsilon + yx'\varepsilon + x'y'\varepsilon^2 = xy + (xy' + yx')\varepsilon$$

引き算と割り算についても同様である。

いまやこの拡張算術のもとで多項式を計算できる。もし $P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n$ ならば、

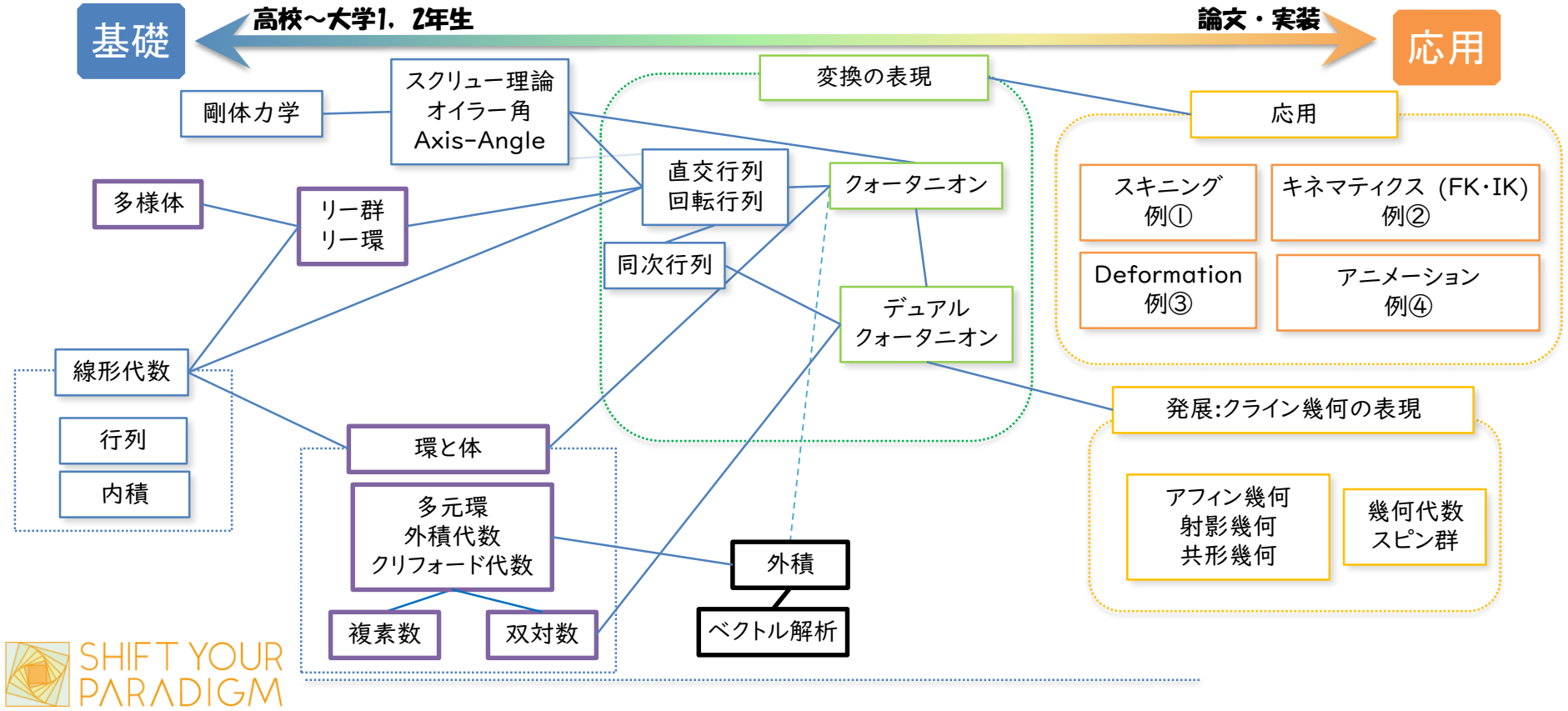
$$\begin{aligned} P(x + x'\varepsilon) &= p_0 + p_1(x + x'\varepsilon) + \cdots + p_n(x + x'\varepsilon)^n \\ &= p_0 + p_1 x + \cdots + p_n x^n + p_1 x'\varepsilon + 2p_2 x x'\varepsilon + \cdots + n p_n x^{n-1} x'\varepsilon \\ &= P(x) + P^{(1)}(x)x'\varepsilon \end{aligned}$$

ここで $P^{(1)}$ は P の最初の引数に対する微分であり、x'（種と呼ばれる）は任意に取れる。

https://ja.wikipedia.org/wiki/自動微分

SHIFT YOUR PARADIGM

ロードマップ【高校数学～デュアルクォータニオン】



SHIFT YOUR PARADIGM

(デュアル)クォータニオンの「心」

複素数平面と(デュアル)クォータニオンは同じ仲間
その心を知れば
どの技術も同じ発想で生まれることが分かる!

- 数によって変換を表す
- 数の演算が図形的な意味に対応 (積=合成，線型和=混ぜ合わせ)
- 変換には種類・階層がある
  必要なところに制限して，ちょうどよく表すのが大事
- 混ぜ合わせによって，非線型で複雑な変換も作り出せる

回転 ⊂ 剛体変換 ⊂ 相似変換 ⊂ アフィン変換 ⊂ 非線型変換

複素数
クォータニオン

デュアル複素数
DQ

同次行列

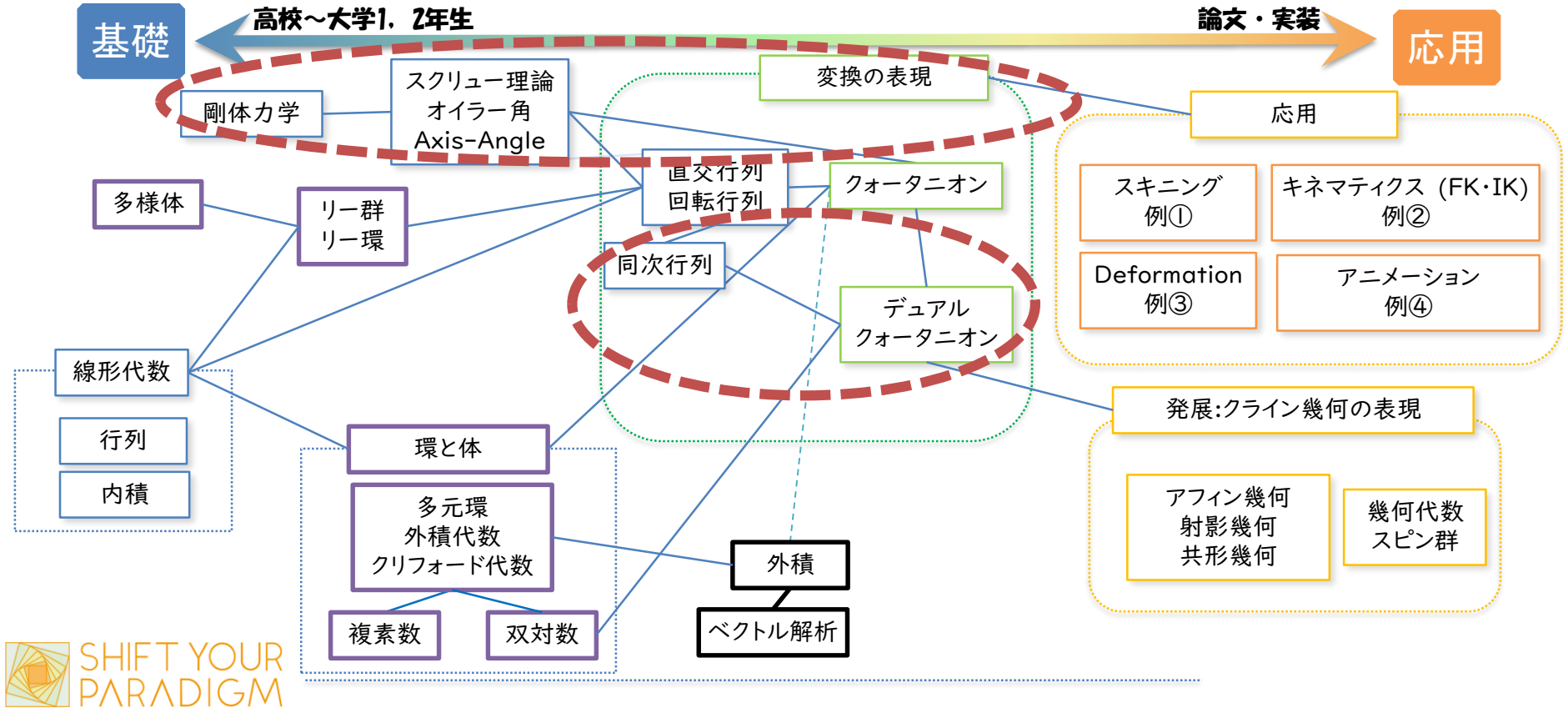
混ぜ合わせ

SHIFT YOUR PARADIGM

剛体変換 ＝ 回転+平行移動

スピーカー：落合啓之(九州大学)

ロードマップ【高校数学～デュアルクォータニオン】



SHIFT YOUR PARADIGM

剛体運動と剛体変換

- 剛体運動：
  運動の前後で形が変わらない。
  長さや角度が保たれる。
- ３次元空間の間の写像で、長さや角度を保つものを剛体変換と呼ぶ。
- 剛体運動は剛体変換で表せる。
- 頂点の位置の座標ではなく、変換をコントロールすることで運動を制作・制御する。

剛体変換　(１つの変換の定義と性質)

- 定義：長さを保つ変換
- 性質１：角度も保つ
- 性質２：回転と平行移動の合成で書ける
  しかも１回ずつで良い。
  回転して平行移動して回転して平行移動して…を１回の回転と１回の平行移動で書ける
- 性質３：回転してから平行移動と書けるし
  平行移動してから回転とも書ける。(半直積)
  その時、回転は２通りの表示で同じ、
  平行移動は２通りの表示で一般には異なる。

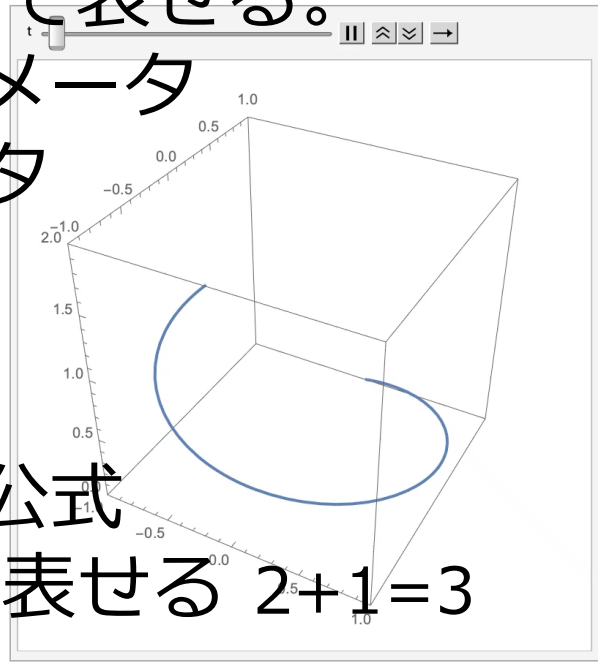
SHIFT YOUR PARADIGM

剛体変換　(集団としての性質)

- nDの回転は n(n-1)/2 個のパラメータで記述
- nDの平行移動は n 個のパラメータで記述
- nDの剛体変換はn(n+1)/2個のパラメータで記述
- 問：(n+1)Dの回転変換とnDの剛体変換のパラメータの個数が一致しているのは偶然？
  答：カルタン運動群と見ると関係がある(二重数)。
- 2Dの場合は 1+2=3 パラメータ
- 3Dの場合は 3+3=6 パラメータ

剛体変換(スクリュー理論)[axis-angle]

- 定理(Euler-Chasles)
  3D剛体変換は「ネジを回す」動きで表せる。
  ネジの方向：3D内の直線：4パラメータ
  ネジの向きに進む量：１パラメータ
  ネジを回す角度：１パラメータ
- 4+1+1=6 パラメータ
- おもちゃのモデル：ロドリゲスの公式
  3D回転は回転軸と軸周りの角度で表せる 2+1=3



SHIFT YOUR PARADIGM

剛体変換(行列による表示)

- nD剛体変換は (n+1) 次の行列で表せる

$$\left(\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & s_1 \\ r_{21} & r_{22} & r_{23} & s_2 \\ r_{31} & r_{32} & r_{33} & s_3 \\ 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \\ 1 \end{pmatrix}$$

- 同次座標(homogeneous coordinates)の考え方
  非同次座標(通常の座標)を射影幾何で拡張

SHIFT YOUR PARADIGM

剛体変換の表示の特性

- 行列は便利ではあるが、補間や平均にはあまり向いていない。なぜなら、複数の回転や剛体変換を行列と思って平均したものが、回転や剛体変換ではないから。
- 補間操作で閉じた体系が欲しい。
  その要求への一つの答えが DQ
- 剛体運動＝「回転+平行移動」という表し方は座標の取り方に依存する。
- 回転と平行移動に分けずに一緒に扱う方法は？ ➡ DQ

可換・非可換

- 可換 : AしてからBするのとBしてからAするのが同じ
- 平行移動どうしは可換
- 2D回転どうしは可換
- 3D回転どうしは非可換(可換とは限らない)
- 回転と平行移動は非可換(2D,3D,nD)
- 3D回転や2D3D剛体変換らを表すのには非可換な体系が不可避。一つの回答が四元数やDQ。

小ネタ：複素数は可換なので2D剛体変形を表すには一工夫必要

数の集まり(数学では体(たい)と言います)

- 四則演算(加減乗除)ができる：直感が効く
- 結合則、分配則が成り立つ
- 掛け算の交換則は犠牲にする(仮定しない)
- 割り算ができる、特に逆数が高速かつ誤差なしに計算できることが四元数やDQの利点
- (行列の逆行列は成分の非線形な式。明示的ではあるが分配法則などとの見通しが(あまり)良くない)

剛体変換 まとめ

- 剛体変換を表す方法は色々(回転+平行移動、行列、スクリュー、デュアルクォータニオン)ある。
- それぞれに利点や欠点がある。
- どの方式でも非可換性を反映する仕組みは必須。
- 四元数やDQの特徴：
  2D,3Dしか表せないけれど。
  補間や平均に向いている。
  数で表せる。直感的な計算に向いている

点も回転も関数もみんな数。

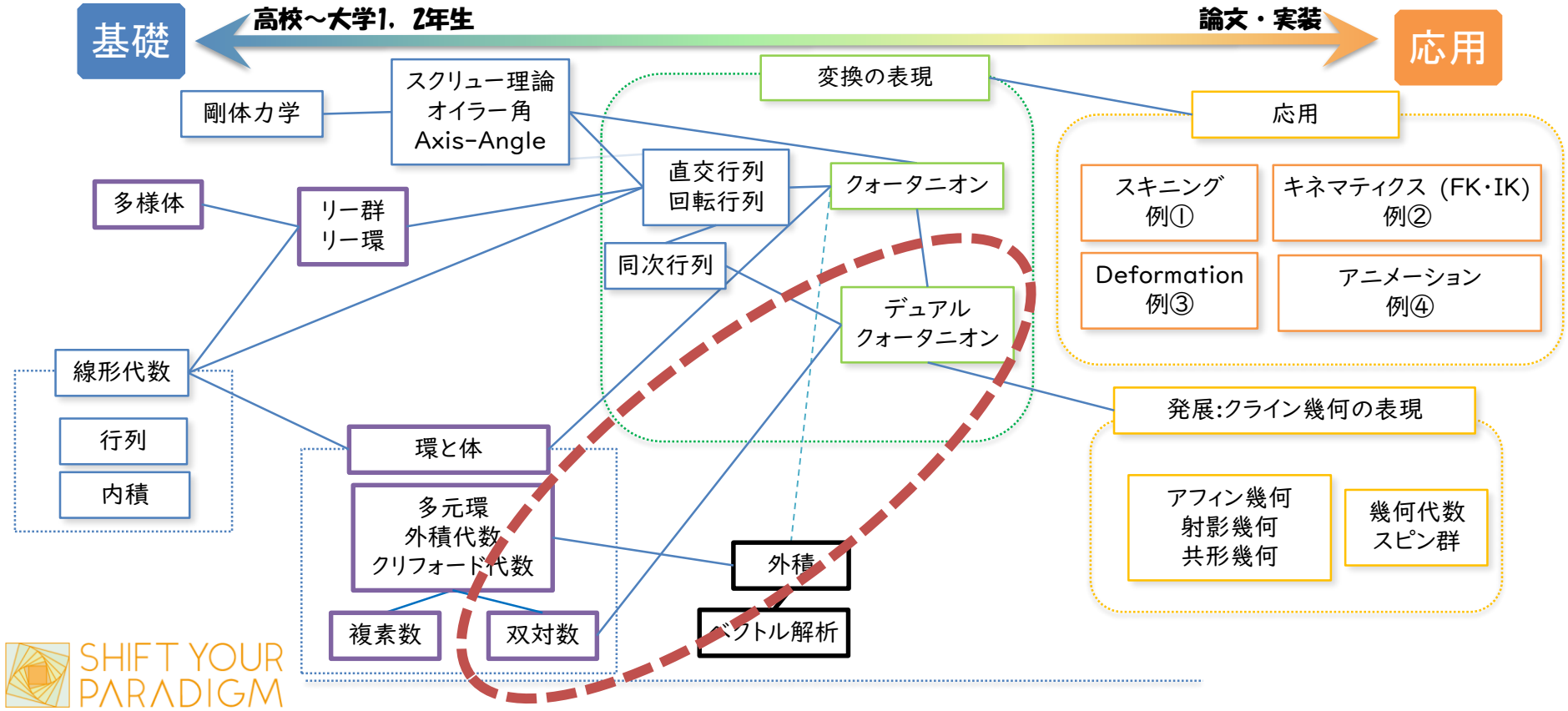
SHIFT YOUR PARADIGM

デュアル数(二重数/双対数)

スピーカー：鍛冶静雄(九州大学)

SHIFT YOUR PARADIGM

ロードマップ【高校数学～デュアルクォータニオン】



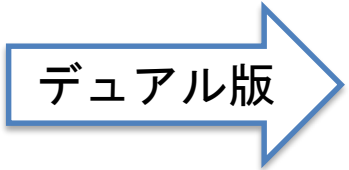
SHIFT YOUR PARADIGM

“デュアル化”による数のパワーアップ

実数： $\mathbb{R}$
微積分

複素数： $\mathbb{C}$
二次元回転

クォータニオン： $\mathbb{H}$
三次元回転



デュアル実数： $\mathbb{R}\langle\epsilon\rangle$
自動微分

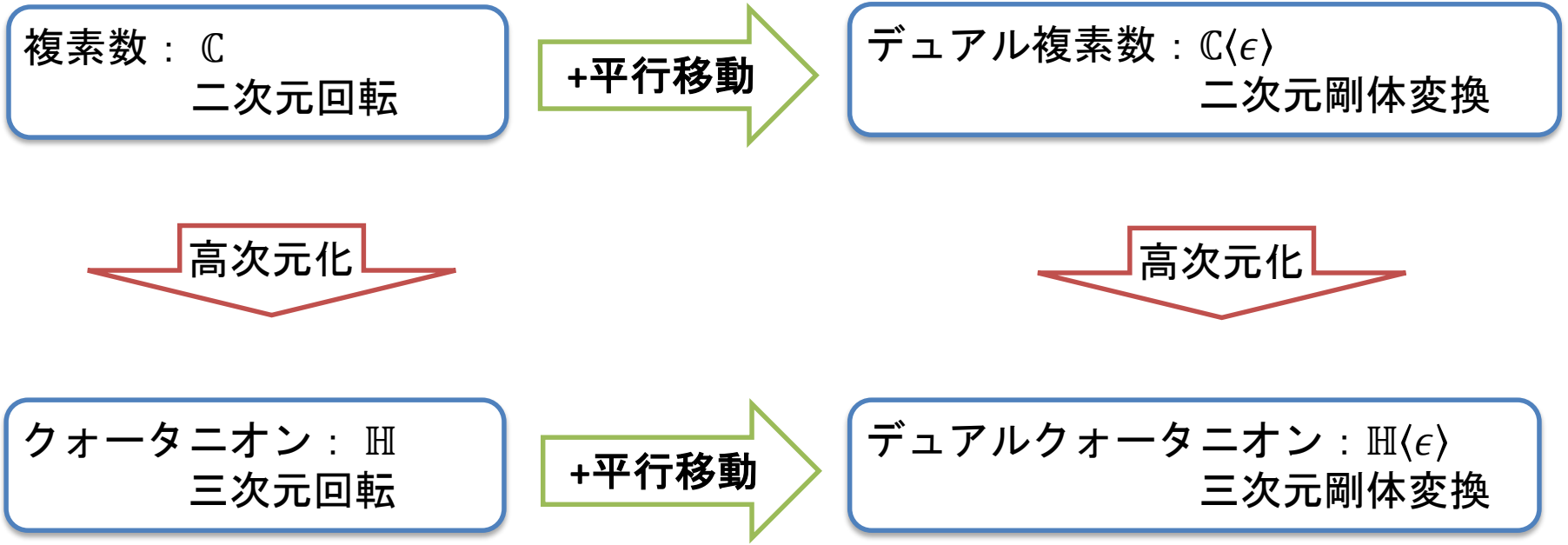
デュアル複素数： $\mathbb{C}\langle\epsilon\rangle$
二次元剛体変換

デュアルクォータニオン： $\mathbb{H}\langle\epsilon\rangle$
三次元剛体変換

今日は $\mathbb{H}\langle\epsilon\rangle$ を説明する代わりに，
本質は同じでより簡単な $\mathbb{C}\langle\epsilon\rangle$ を主に紹介する

SHIFT YOUR PARADIGM

二つの軸による数の拡張



SHIFT YOUR PARADIGM

数の拡張方法 (多元環)

普通は基底の 1 は書かない

複素数 $\mathbb{C}$

二次元ベクトル $a\,1 + b\,i\ (a, b \in \mathbb{R})$ — 足し算構造

$i^2 = -1$ という関係式 — 掛け算構造

クォータニオン $\mathbb{H}$

四次元ベクトル $a + bi + cj + dk$

$i^2 = j^2 = k^2 = -1$ という関係式

デュアル数 $\mathbb{R}\langle\epsilon\rangle, \mathbb{C}\langle\epsilon\rangle, \mathbb{H}\langle\epsilon\rangle$

2 倍次元ベクトル $q_0 + q_1\epsilon\ (q_0, q_1 \in \mathbb{R}\ \text{or}\ \mathbb{C}\ \text{or}\ \mathbb{H})$

$\epsilon^2 = 0$ という関係式

いずれも，新しい文字を導入して
掛け算のルールを決めた形になっている

SHIFT YOUR PARADIGM

デュアル実数 $\mathbb{R}\langle\epsilon\rangle$

実数に ϵ を付け足したもの。(実２次元ベクトル空間)

$$\mathbb{R}\langle\epsilon\rangle := \{x + y\,\epsilon \mid x, y \in \mathbb{R}\}$$

足し算　は普通に ϵ をただの文字と思う

掛け算は ϵ を文字と思って計算し、$\epsilon^2 = 0$ に置き換える

微小数のイメージ

練習：

- $(3 + \sqrt{2}\epsilon)(-1 + 2\epsilon) = -3 + (6 - \sqrt{2})\epsilon + 2\sqrt{2}\epsilon^2$

実数 θ に対して

- $\cos(\theta + \epsilon) = 1 - \frac{(\theta+\epsilon)^2}{2!} + \frac{(\theta+\epsilon)^4}{4!} + \cdots$

$$= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} + \cdots\right) - \left(\theta\epsilon - \frac{\theta^3\epsilon}{3!} + \cdots\right)$$

$$= \cos(\theta) - \sin(\theta)\epsilon$$

テイラー展開を使って
関数の値も計算可能

$(\cos(\theta))' = -\sin(\theta)$に注意

デュアル実数による自動微分

(実数値)関数 f に対して，そのテイラー展開

$$f(t) = f(0) + f'(t)t + \frac{f''(t)}{2!}t^2 + \cdots$$

に $t = x + \epsilon$ を”代入”すると (ここで x は実数)

$$f(x + \epsilon) = f(x) + f'(x)\epsilon$$

デュアル実数の足し算と掛け算は，微分に整合的になっている

関数の引数・戻り値をデュアル実数に拡張することで，自動微分が実装可能！

デュアル複素数(反可換双対複素数)

複素数に ϵ を付け足したもの。(複素２次元ベクトル空間)

$$\mathbb{C}\langle\epsilon\rangle := \{\, p_0 + p_1\,\epsilon \mid p_0, p_1 \in \mathbb{C}\}$$

足し算　は ϵ をただの文字と思う

掛け算は $(p_0 + p_1\epsilon)(q_0 + q_1\epsilon) = p_0q_0 + (p_1\bar{q}_0 + p_0q_1)\epsilon$

つまり、　$\epsilon^2 = 0$

かつ　$\epsilon i = -i\epsilon$

とひっくり返すときにマイナスをつける。

この辺が反可換
デュアル複素数に特有

ちょっと計算練習

$$(3+5i\epsilon)-(1+i+2\epsilon+3i\epsilon)$$
$$=(2-i)+(-2+2i)\epsilon$$

$$(3+5i\epsilon)(1+i+2\epsilon+3i\epsilon)$$
$$=(3+3i+6\epsilon+9i\epsilon)+(5i\epsilon+5i\epsilon i+10i\epsilon^2+15(i\epsilon)^2)$$
$$=(3+3i+6\epsilon+9i\epsilon)+(5i\epsilon+5\epsilon)$$
$$=(3+3i)+(11+14i)\epsilon$$

結合法則や分配法則は成り立つので、
交換法則が成り立たないことだけ注意して計算

平面上の点への作用

平面上の点とデュアル複素数を次のように対応付ける

$$(x, y) \leftrightarrow 1 + (x + iy)\epsilon \qquad (x, y \in \mathbb{R})$$

今、この点を、別のデュアル複素数 $p_0 + p_1\epsilon \quad (p_0, p_1 \in \mathbb{C})$

をつかって動かしたい。

デュアル複素数を平面上の点に作用させるという

ここで，単位デュアル複素数，つまり，$|p_0|^2 = p_0\overline{p_0} = 1$ の場合に限定する．

次で定める両側から挟んだ掛け算を，新しい記号 $\diamond$ で表すことにする

$$(p_0 + p_1\epsilon) \diamond (1 + (x + iy)\epsilon) := (p_0 + p_1\epsilon)(1 + (x + iy)\epsilon)(\bar{p}_0 + p_1\epsilon)$$

変換を表す
デュアル複素数

平面の点を表す
デュアル複素数

クォータニオン同様
“サンドイッチ”

右辺を計算すると、 $= 1 + (p_0^2(x + iy) + 2p_0p_1)\epsilon$

作用の式: 回転も平行移動もこれ一つで

作用の式：$(p_0 + p_1\epsilon) \diamond (1 + (x + iy)\epsilon) = 1 + (p_0^2(x + iy) + 2p_0p_1)\epsilon$

$1 + (a + bi)\epsilon$　を作用させると

$(1 + (a + bi)\epsilon) \diamond (1 + (x + iy)\epsilon) = 1 + ((x + 2a) + (y + 2b)i)\epsilon$

なので、$(x, y) \mapsto (x + 2a, y + 2b)$　と平行移動

$\cos(\theta) + i\sin\theta + 0\epsilon$　を作用させると

$$(\cos(\theta) + i\sin\theta) \diamond (1 + (x + iy)\epsilon)$$
$$= 1 + (x\cos 2\theta - y\sin 2\theta) + i(y\cos 2\theta + x\sin 2\theta))\epsilon$$

なので、２θの回転

一般には、作用の式により回転と平行移動が同時に(=剛体変換)行われる!

デュアル複素数のメリット

なんか計算ややこしいけど…

うれしい点

1. 複数の変換の合成を計算できる

結合法則を使って回転の合成(加法定理)を計算した類似

2. 複数の変換を混ぜ合わすことができる

掛け算が作用に対応するので，”余った”演算である足し算を混ぜ合せに利用

デュアル複素数が環になっているのがうれしい

うれしい点 その１　　変換の合成

二つのデュアル複素数

$q_0 + q_1\epsilon \quad p_0 + p_1\epsilon$

で表される二つの変換を、
この順番で続けて作用させる変換は、それらの関

$$(p_0 + p_1\epsilon)(q_0 + q_1\epsilon)$$

で表される.

確認：$(p_0 + p_1\epsilon) \diamond ((q_0 + q_1\epsilon) \diamond (1 + (x + iy)\epsilon))$

$= ((p_0 + p_1\epsilon)(q_0 + q_1\epsilon)) \diamond (1 + (x + iy)\epsilon)$

変換を順番に作用させることなく、まとめて一つのデュアル複素数で表せる(**結合法則**のご利益)

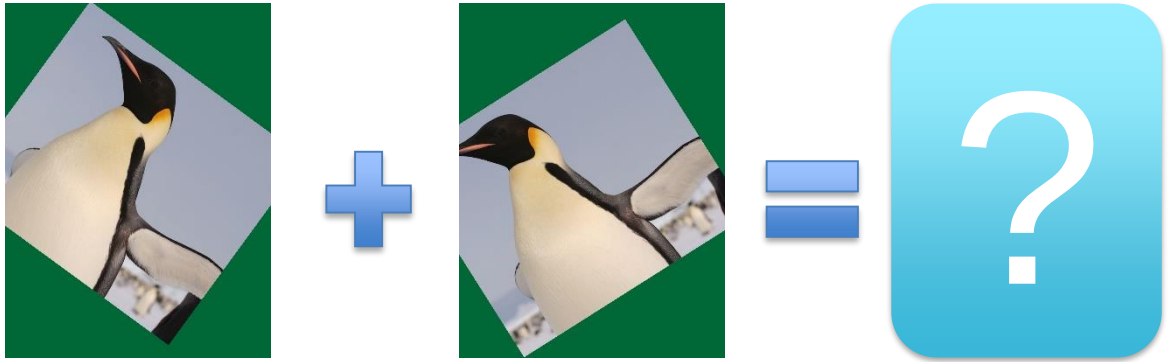
うれしい点 その２　変換の混ぜ合わせ

二つのデュアル複素数で表される変換

$q_0 + q_1\epsilon \quad p_0 + p_1\epsilon$　を例えば 70% : 30% で混ぜ合わせた変換を

$$0.7(q_0 + q_1\epsilon) + 0.3(p_0 + p_1\epsilon)$$

と単純な線型和で定めることができる。これを発展させると、
上の方では右回転を多めに、下の方では左回転を多めにブレンドで、
ということができる。

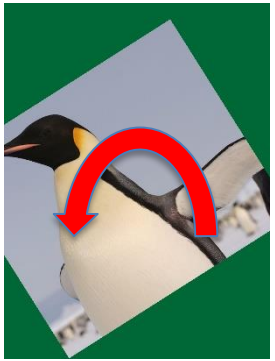


Spheniscidae - Photo (BY-SA 3.0) Dbush on Wikimedia Commons

うれしい点 その２　変換の混ぜ合わせ

$q_0 + q_1\epsilon$

$p_0 + p_1\epsilon$

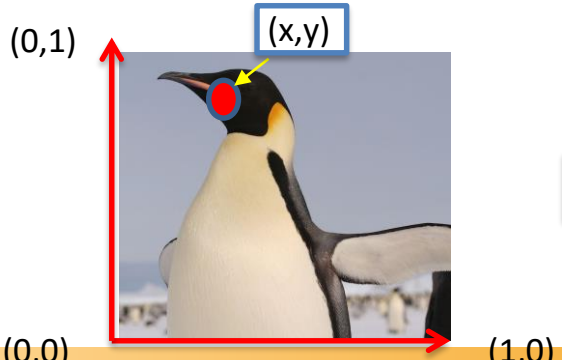


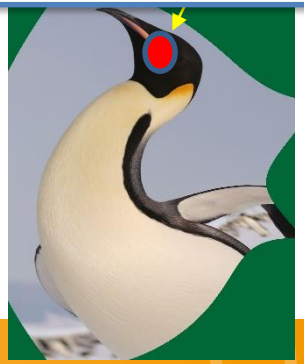
のニつを y 座標に応じてブレンド

点$\mathbb{R}^2 \ni (x, y) \mapsto 1 + x\epsilon + yi\epsilon$

$y(q_0 + q_1\epsilon) + (1 - y)(p_0 + p_1\epsilon)$ を各点$1 + (x + iy)\epsilon$ に作用させると

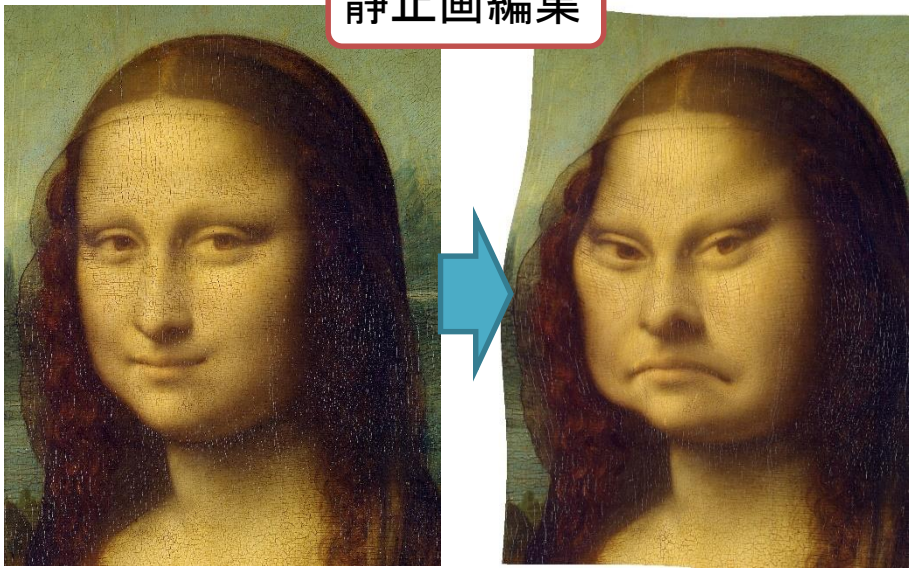
$(y(q_0 + q_1\epsilon) + (1 - y)(p_0 + p_1\epsilon)) \diamond (1 + (x + iy)\epsilon)$





インタラクティブな画像変形の例

静止画編集



カメラ入力のリアルタイム変形

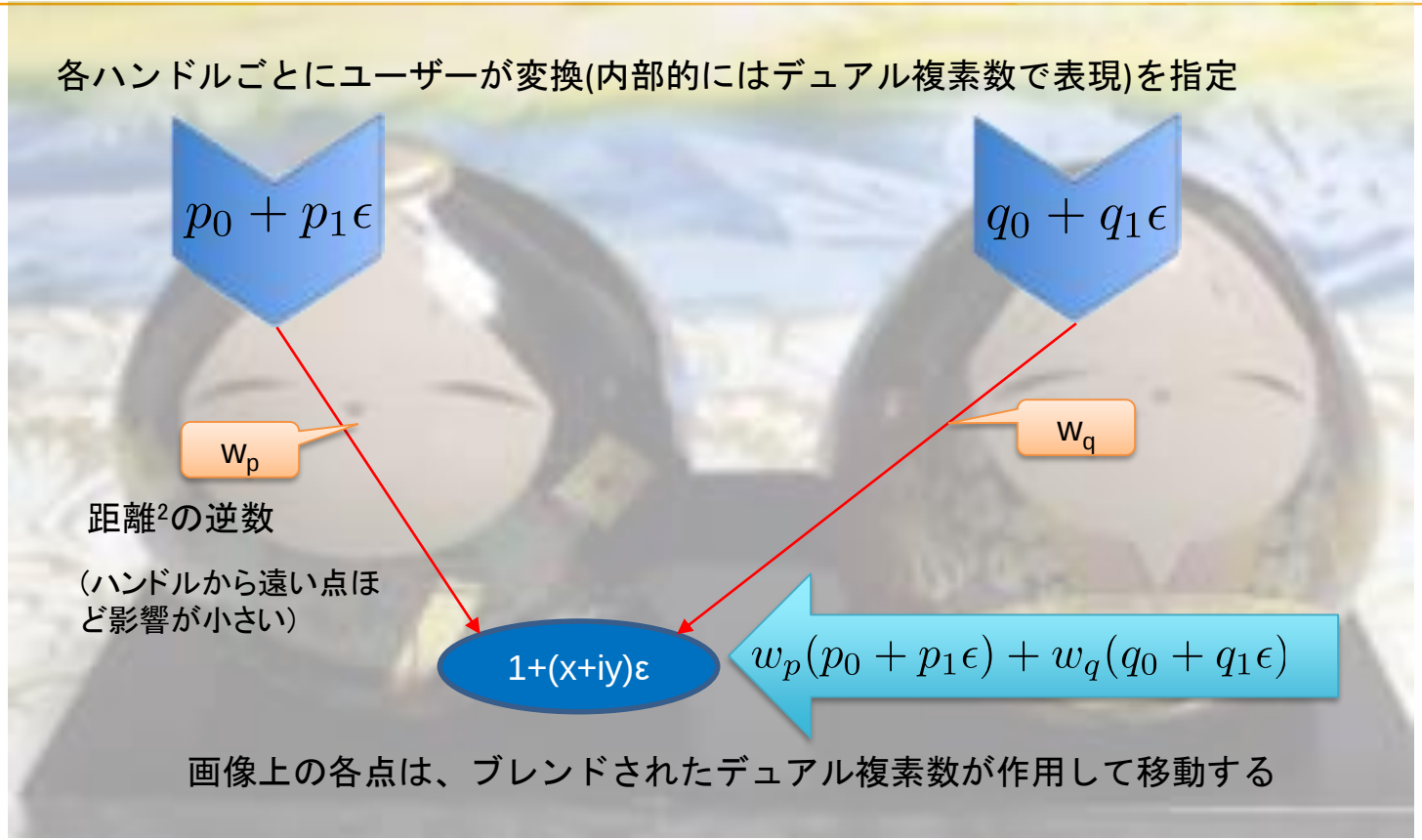
デュアル複素数の詳細・実装は以下を

Matsuda-Kaji-Ochiai, *Anti-commutative Dual Complex Numbers and 2D Rigid Transformation*

https://github.com/shizuo-kaji/DualComplexNumbers

https://github.com/KyushuUniversityMathematics/iPad-ProbeDeformer

各ハンドルごとにユーザーが変換(内部的にはデュアル複素数で表現)を指定



画像上の各点は、ブレンドされたデュアル複素数が作用して移動する

(線型和による)混ぜ合わせ

n 個の剛体変換 A_k $(k = 1..n)$ があり(例えばボーン)
各点 $x \in \mathbb{R}^3$ に，重み $w_k(x) \in \mathbb{R}_{\geq 0}$ が定まっている時

- リニアスキニング $A(x) = \dfrac{\sum_{k=1}^{n} w_k(x)\, A_k}{\sum_{k=1}^{n} w_k(x)}$

剛体変換とは限らない

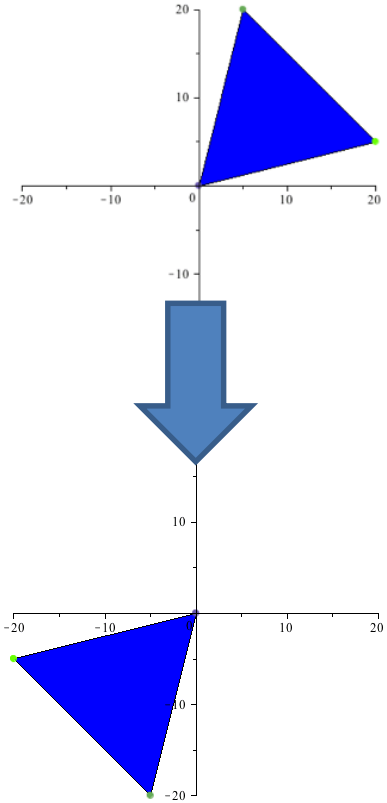
- デュアル数スキニング $Q(x) = \dfrac{\sum_{k=1}^{n} w_k(x)\, Q_k}{\left|\sum_{k=1}^{n} w_k(x) Q_k\right|}$

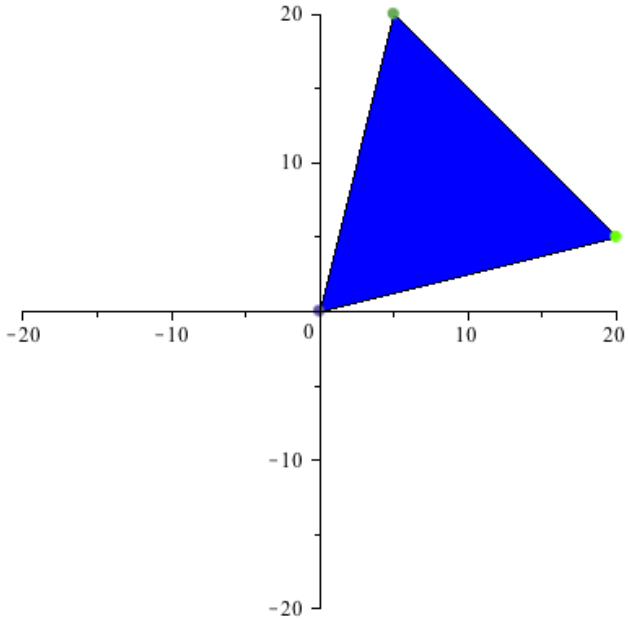
常に剛体変換

剛体変換だと体積が保たれる

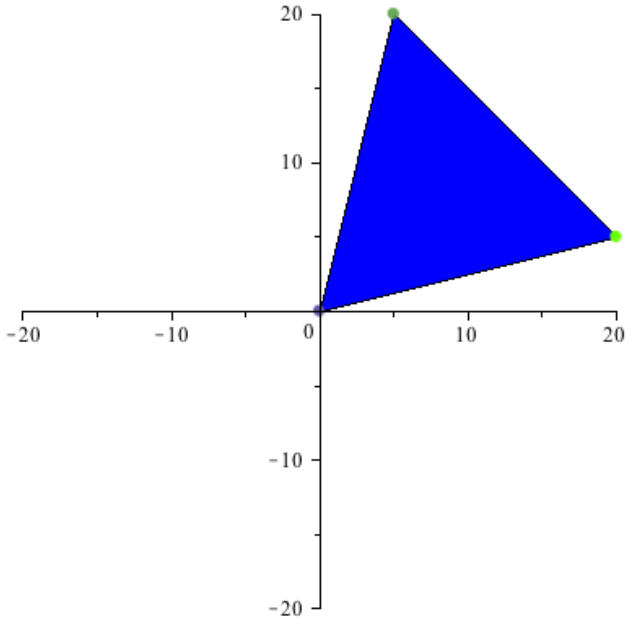
各点では剛体変換だが，重み w_k の定め方により
全体では非線型な複雑な変換が得られることに注意

行列 vs デュアル複素数　線型和補間





途中で潰れる



いつでも剛体変換

Candy wrapper の原因！

デュアルクォータニオンを１スライドで

- Q=$q_0 + q_1\epsilon \ (q_0, q_1 \in \mathbb{H})$
- 共役 $\overline{q_0 + q_1\epsilon} := q_0^* - q_1^*\epsilon$
- ノルム：$|q_0 + q_1\epsilon|^2 = (q_0 + q_1\epsilon)(q_0^* + q_1^*\epsilon) \in \mathbb{R}\langle\epsilon\rangle$
- 単位(unit)デュアルクォータニオン ⇔ ノルムが１
- 空間の点：$\mathbb{R}^3 \ni (x, y, z) \mapsto (1 + (xi + yj + zk)\epsilon)$
- 単位DQの点への作用：$Q \cdot (1 + (xi + yj + zk)\epsilon) \cdot \overline{Q}$

基本的には，デュアル複素数の係数をクォータにオンに取り替えるだけ

SHIFT YOU
PARADIGM

補足：単位デュアルクォータニオン

ノルム $|q_0 + q_1\epsilon|^2$ はデュアル実数になる.

単位デュアルクォータニオンは

$$|q_0 + q_1\epsilon|^2 = (q_0 + q_1\epsilon)(q_0^* + q_1^*\epsilon)$$
$$= q_0 q_0^* + (q_0 q_1^* + q_1 q_0^*)\epsilon = 1 + 0\epsilon$$

つまり $q_0 q_0^*$=1 かつ $q_0 q_1^* + q_1 q_0^* = 0$ を満たす

発展的補足：上式後者は，四次元ベクトルとしてみた時に q_0, q_1 の直交を意味することに注意
回転を幾何的に表す単位クォータニオンは三次元球面をなすが，
単位デュアルクォータニオンはその接束をなす．接ベクトル成分が平行移動に対応する.

補足：単位DQの逆数の計算は楽 $(q_0 + q_1\epsilon)^{-1} = q_0^* + q_1^*\epsilon$

球面補間 (SLERP の拡張)

軸は一定で
その周りを等各速度で
回転させる

- 任意の単位デュアルクォータニオンは

$$\hat{\theta} \in \mathbb{R}\langle\epsilon\rangle$$

$$\hat{s} = \hat{p}i + \hat{q}j + \hat{r}k, \qquad (\hat{p}, \hat{q}, \hat{r} \in \mathbb{R}\langle\epsilon\rangle, \qquad |\hat{s}| = 1)$$

を用いて以下の形に書ける：

$$\exp(\hat{\theta}\,\hat{s}) = \cos(\hat{\theta}) + \hat{s}\sin(\hat{\theta})$$

オイラーの公式！

- この形では，実数べき t 乗が簡単に計算できる

$$(\cos(\hat{\theta}) + \hat{s}\sin(\hat{\theta}))^t = \exp(\hat{\theta}\,\hat{s})^t = \exp(t\hat{\theta}\,\hat{s}) = \cos(t\hat{\theta}) + \hat{s}\sin(t\hat{\theta})$$

- 二つの $Q_1, Q_2 \in \mathbb{H}\langle\epsilon\rangle$ の球面補間は次で定義

$$t \mapsto Q_1(Q_1^* Q_2)^t$$

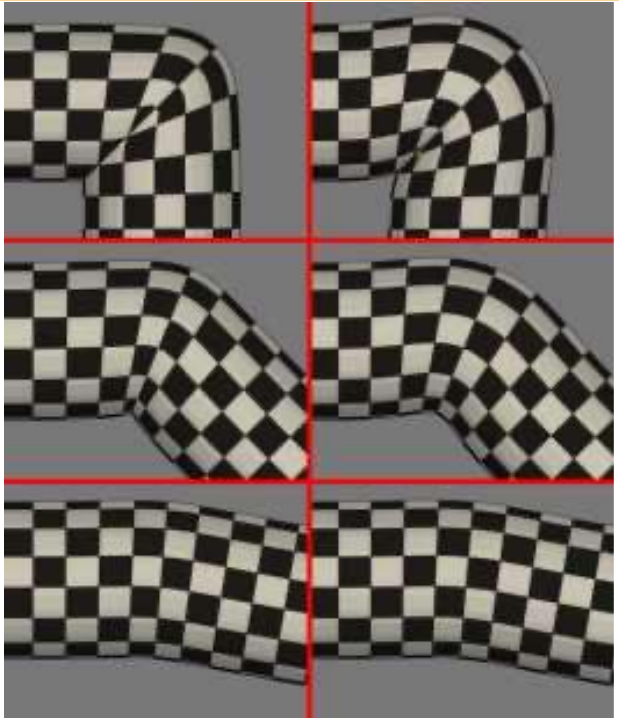
3つ以上の補間は
この方法では難しい

SHIFT YOUR PARADIGM

補足 ：Bulding アーティファクト

- DQスキニングでは，右図のように関節が膨らむ不具合が出る.
- 軽減方法が提案されている

  "A Method Of Artifact Compensation For Dual Quaternion Skinning And Its Application In Digital Twin Models”
  Y. Sulema and C. Rudenko



Unity implementation:
https://github.com/ConstantineRudenko/DQ-skinning-for-Unity

SHIFT YOUR PARADIGM

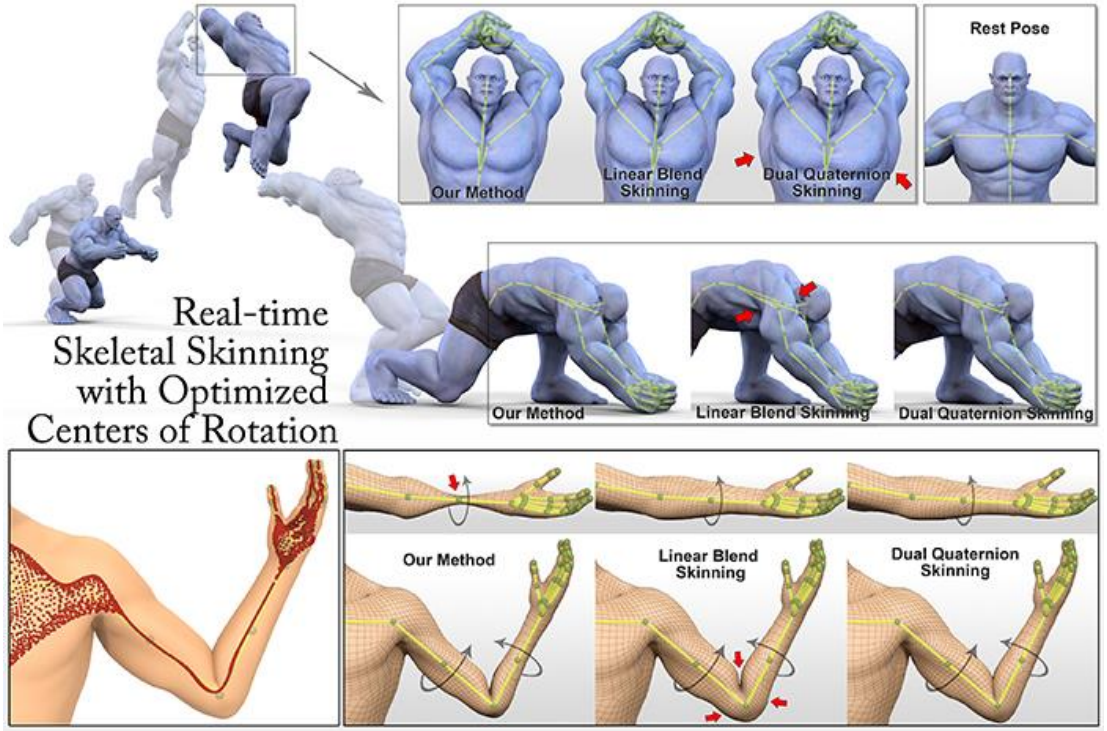
補足：スケーリングの取り扱い

- デュアルクォータニオンでは，スケーリングは扱えない.
- 解決法１：極分解を利用し，スケーリングを独立に扱う
- 解決法２：スケーリングも同時に扱える幾何代数を用いる

Geometric Skinning with Approximate Dual Quaternion Blending
L. Kavan, S. Collins, J. Žára, and C. O'Sullivan (2008) Section 4.2

SHIFT YOUR PARADIGM

補足： Skinning with Optimized Centers of Rotation



- DQ を使わない方法
- 実行時は DQ より高速
- ウェイト塗り替えのたびに再計算が必要

B. Le and J. Hodgins (2016)

SHIFT YOUR PARADIGM

参考資料

- 実装を自分でやりたい方は

  http://rodolphe-vaillant.fr/?e=29

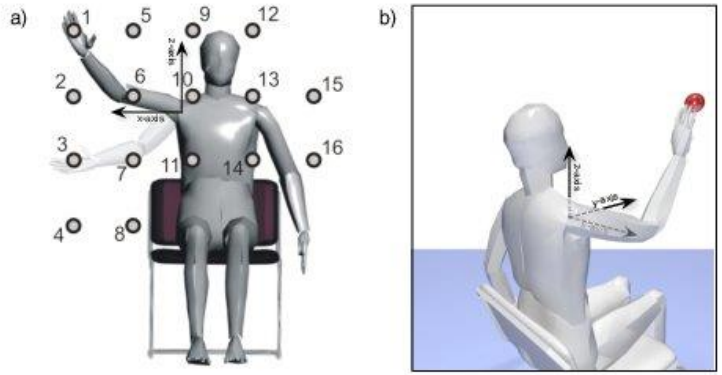
- **幾何代数**はデュアルクォータニオンを置き換えてより広い用途に用いることができる

  https://github.com/jeremyong/klein

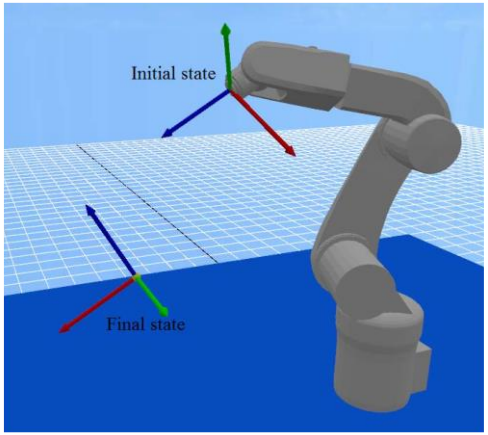
SHIFT YOUR PARADIGM

参考：スキニング以外の DQ 利用例

アニメーション，キネマティックス，ロボティクス，姿勢制御・推定 etc.



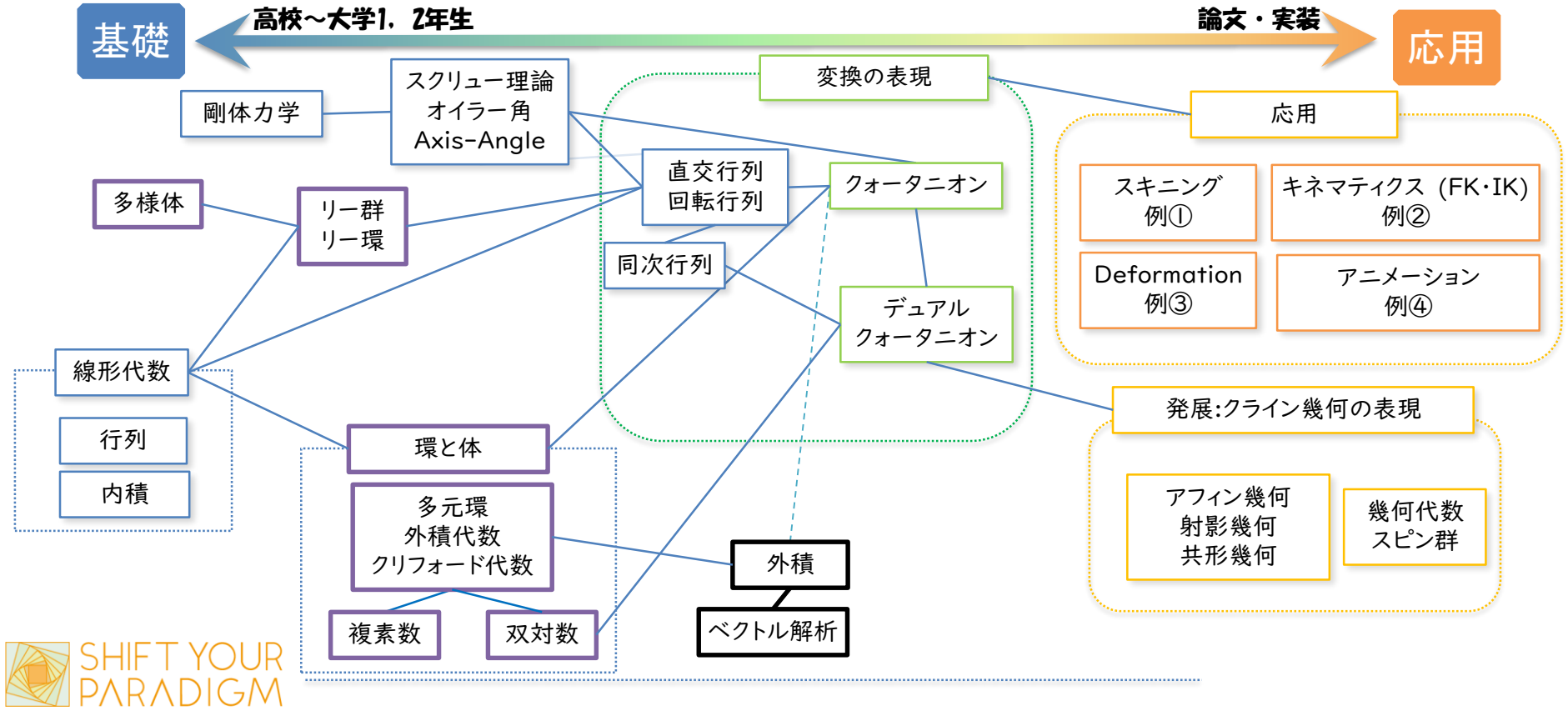
[Schilling et al., 2011]



[Pham et al, 2010]

SHIFT YOUR PARADIGM

ロードマップ【高校数学～デュアルクォータニオン】



SHIFT YOUR PARADIGM

(デュアル)クォータニオンの「心」

複素数平面と(デュアル)クォータニオンは同じ仲間
その心を知れば
どの技術も同じ発想で生まれることが分かる!

- 数によって変換を表す
- 数の演算が図形的な意味に対応 (積=合成，線型和=混ぜ合わせ)
- 変換には種類・階層がある
  必要なところに制限して，ちょうどよく表すのが大事
- 混ぜ合わせによって，非線型で複雑な変換も作り出せる

回転 ⊂ 剛体変換 ⊂ 相似変換 ⊂ アフィン変換 ⊂ 非線型変換

複素数
クォータニオン

デュアル複素数
DQ

同次行列

混ぜ合わせ