コンピュータグラフィクス論

– モデリング (1) –

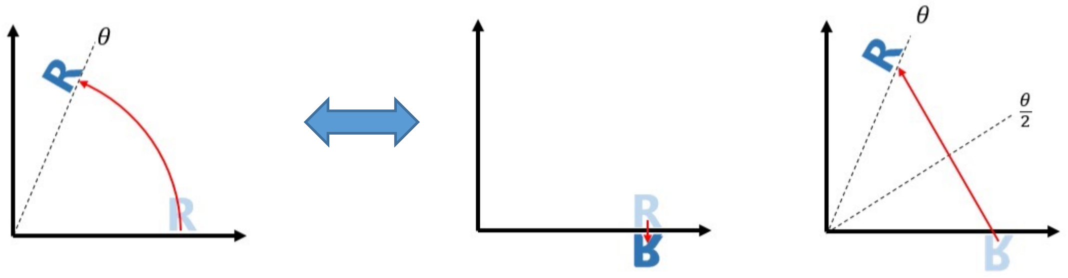
2021年4月15日

高山 健志

クォータニオンについての補足説明

クォータニオンの直感的な説明 (概要)

1. あらゆる回転操作は、偶数回の反転操作に分解できる



2. クォータニオンを使うと、3Dの反転操作を簡潔に記述できる

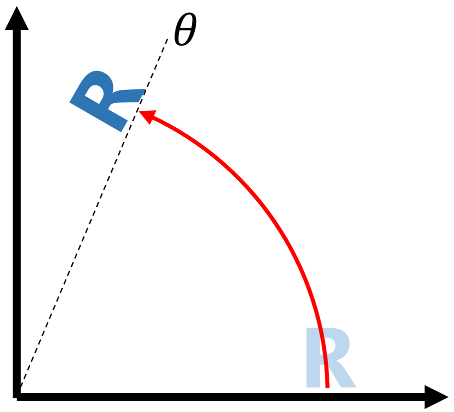
$$R_{\vec{f}}(\vec{x}) = -\vec{f}\,\vec{x}\,\vec{f}^{-1}$$

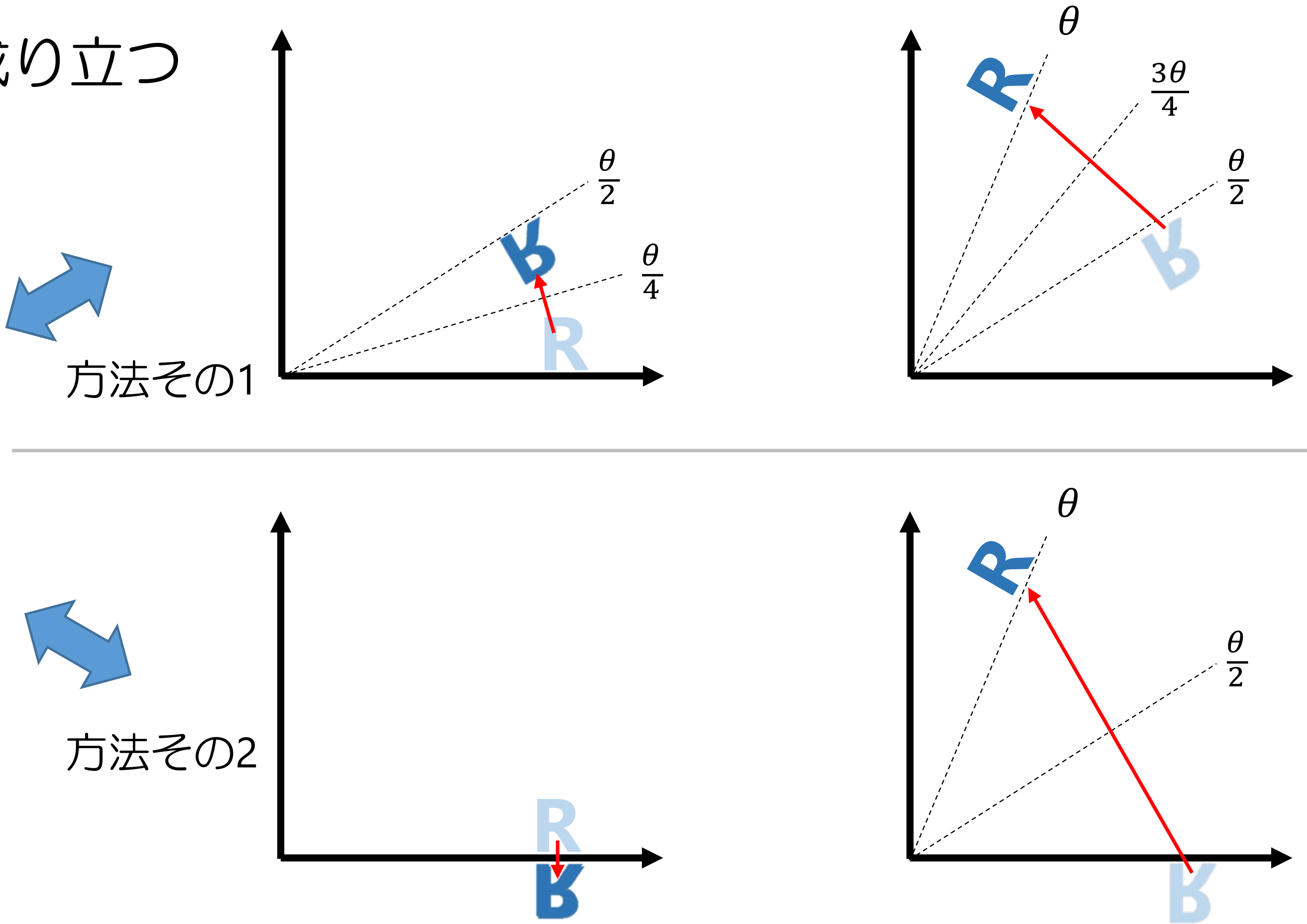
3. 回転操作と等価な反転操作に相当するクォータニオンを合成すると、公式の形が出てくる

$$R_{\vec{g}}\left(R_{\vec{f}}(\vec{x})\right) = \left(\cos\frac{\theta}{2} + \vec{\omega}\sin\frac{\theta}{2}\right)\vec{x}\left(\cos\frac{\theta}{2} - \vec{\omega}\sin\frac{\theta}{2}\right)$$

あらゆる回転は、偶数回の反転に分解できる

- 数学的に証明されている
  - 任意次元の空間に対して成り立つ
- もちろん、一意ではない





クォータニオン (四元数) のおさらい

- 複素数：実部と虚部のペア　$a + b\,\mathbf{i}$

- 四元数：スカラーとベクトルのペア　$a + \vec{v}$
- 四元数の積の定義：

$$(a_1 + \overrightarrow{v_1})(a_2 + \overrightarrow{v_2}) \coloneqq \overbrace{a_1 a_2 - \overrightarrow{v_1} \cdot \overrightarrow{v_2}}^{\text{スカラー部}} + \overbrace{a_1 \overrightarrow{v_2} + a_2 \overrightarrow{v_1} + \overrightarrow{v_1} \times \overrightarrow{v_2}}^{\text{ベクトル部}}$$

- 純粋なベクトルも、四元数と見なせば積を取れる：

$$\overrightarrow{v_1}\,\overrightarrow{v_2} = -\overrightarrow{v_1} \cdot \overrightarrow{v_2} + \overrightarrow{v_1} \times \overrightarrow{v_2}$$

- 特筆すべき性質：
  - (後で使う)

$$\boxed{\vec{v}\,\vec{v} = -\|\vec{v}\|^2}$$

$\vec{v} \times \vec{v}$ は必ずゼロ

$$\boxed{\vec{v}^{-1} = -\frac{\vec{v}}{\|\vec{v}\|^2}}$$

右辺に $\vec{v}$ を掛けると1になる

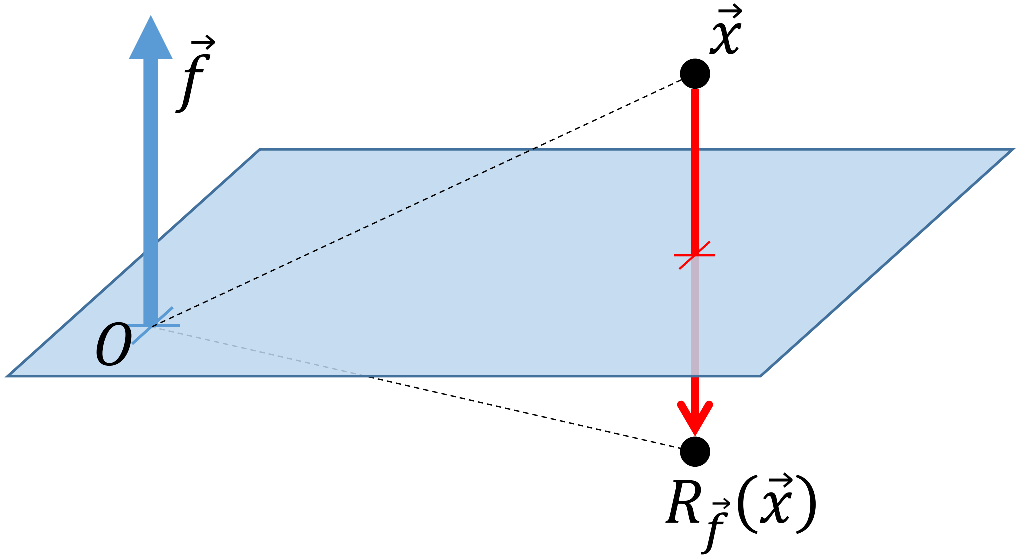
$$\boxed{\vec{v} \cdot \vec{w} = 0 \text{ の場合、} \vec{v}\,\vec{w} = -\vec{w}\,\vec{v}}$$

$\vec{v}\,\vec{w} = \vec{v} \times \vec{w} = -\vec{w} \times \vec{v} = -\vec{w}\,\vec{v}$

クォータニオンを使った反転の表現

- 原点を通りベクトル $\vec{f}$ に直交する面に関して、点 $\vec{x}$ を反転させる式：

$$R_{\vec{f}}(\vec{x}) \coloneqq -\vec{f}\,\vec{x}\,\vec{f}^{-1}$$



- 反転の性質を満たすことを確認：
  - 線形性：

    $$R_{\vec{f}}(a\,\vec{x} + b\,\vec{y}) = a\,R_{\vec{f}}(\vec{x}) + b\,R_{\vec{f}}(\vec{y})$$

  - $\vec{f}$ は $-\vec{f}$ へ写像される：

    $$R_{\vec{f}}(\vec{f}) = -\vec{f}\,\vec{f}\,\vec{f}^{-1} = -\vec{f}$$

  - 点 $\vec{x}$ が $\vec{x} \cdot \vec{f} = 0$ を満たす、つまり平面上にあるとき、$\vec{x}$ は不変：

    $$R_{\vec{f}}(\vec{x}) = -\vec{f}\,\vec{x}\,\vec{f}^{-1} = -\left(-\vec{x}\,\vec{f}\right)\vec{f}^{-1} = \vec{x}$$

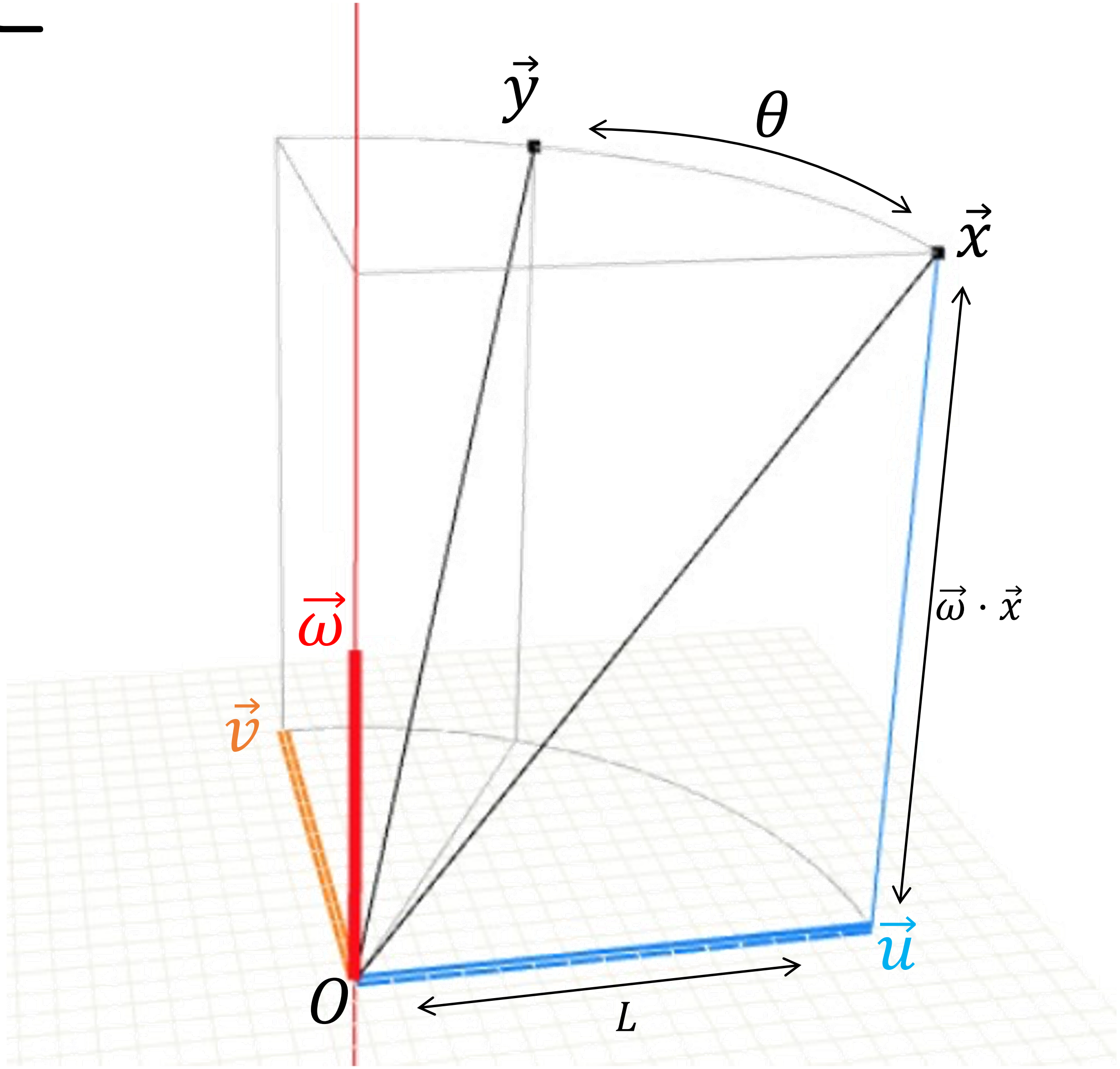
    $\vec{x} \cdot \vec{f} = 0$ のとき、$\vec{f}\,\vec{x} = -\vec{x}\,\vec{f}$ なので

https://math.stackexchange.com/a/7263

軸周りの回転の問題設定

- 回転軸 (単位ベクトル)： $\vec{\omega}$
- 回転角： θ
- 回転前の点： $\vec{x}$
- 回転後の点： $\vec{y} \coloneqq R_{\vec{\omega},\theta}(\vec{x})$

- 局所2D座標系的なものを考える：
  - 「右」ベクトル： $\vec{u} \coloneqq \vec{x} - (\vec{\omega} \cdot \vec{x})\vec{\omega}$
  - 「上」ベクトル： $\vec{v} \coloneqq \vec{\omega} \times \vec{x}$
    - $\|\vec{u}\| = \|\vec{v}\|$ であることに注意
    - (これを L とおく)

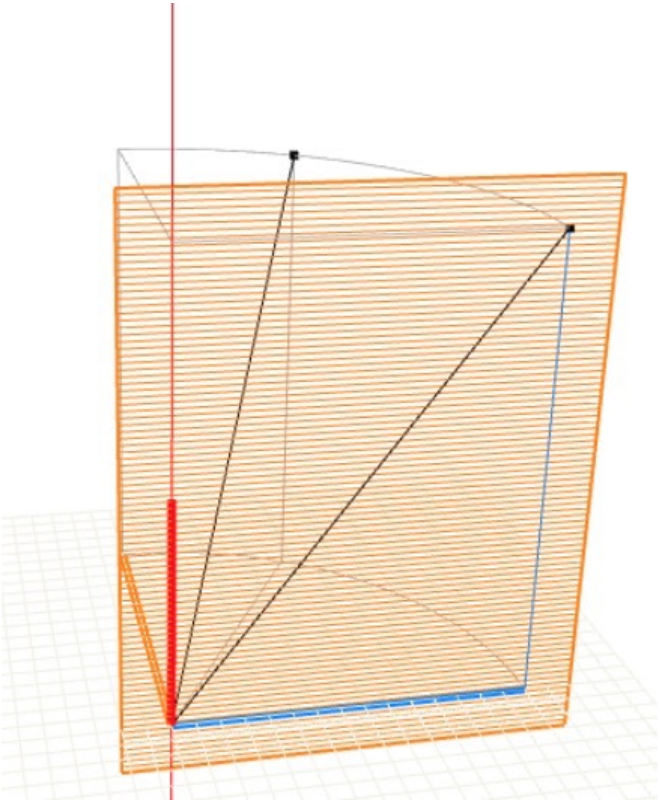


https://legacygl-js.glitch.me/demo/quaternion-schematic.html

軸周りの回転を、2回の反転に分解する

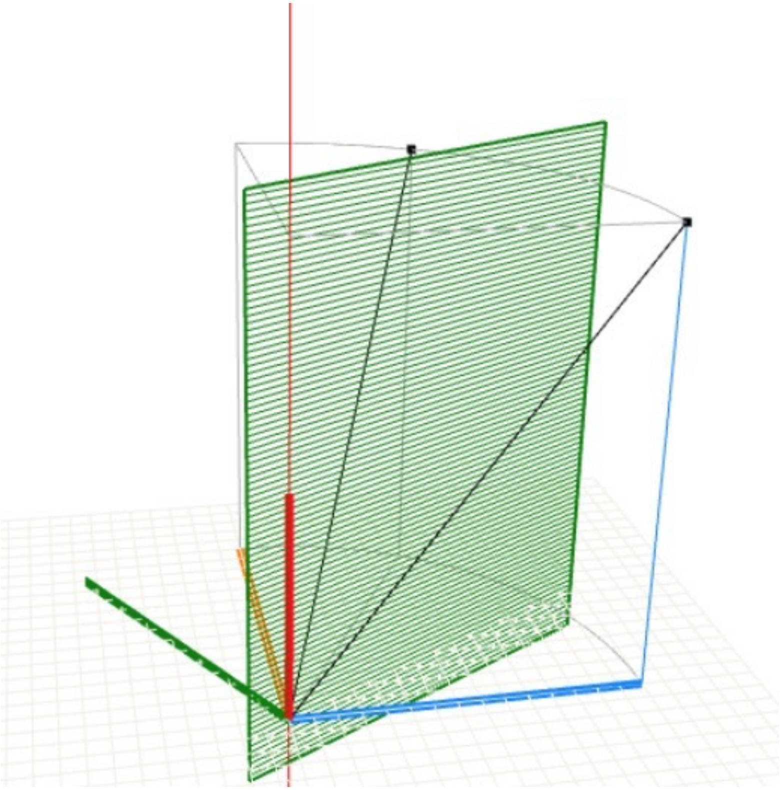
1回目の反転：

$$\vec{f} \coloneqq \vec{v}$$

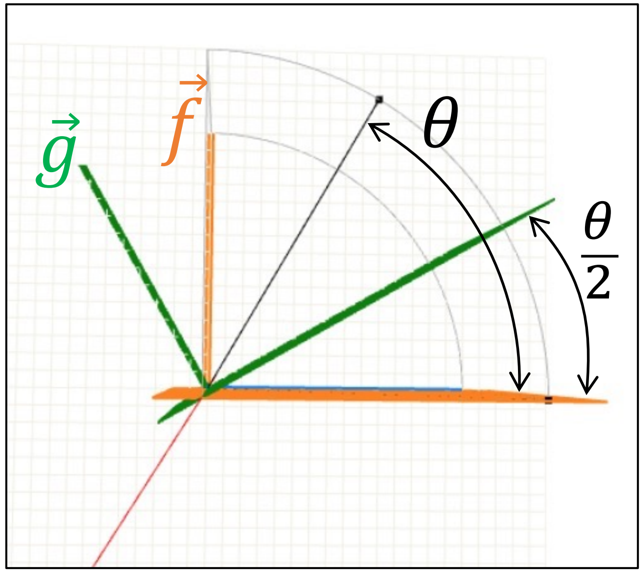


2回目の反転：

$$\vec{g} \coloneqq -\sin\frac{\theta}{2}\,\vec{u} + \cos\frac{\theta}{2}\,\vec{v}$$



上から見た図



https://legacygl-js.glitch.me/demo/quaternion-schematic.html

2回の反転を合成する

- 合成の式： $R_{\vec{g}}\left(R_{\vec{f}}(\vec{x})\right) = R_{\vec{g}}(-\vec{f}\,\vec{x}\,\vec{f}^{-1}) = -\vec{g}(-\vec{f}\,\vec{x}\,\vec{f}^{-1})\vec{g}^{-1} = (\vec{g}\,\vec{f})\,\vec{x}\,(\vec{f}^{-1}\,\vec{g}^{-1})$
  - これに $\vec{f} \coloneqq \vec{v},\quad \vec{g} \coloneqq -\sin\frac{\theta}{2}\,\vec{u} + \cos\frac{\theta}{2}\,\vec{v}$ を代入
- 左側 $\vec{g}\,\vec{f}$ について：

$$\vec{g}\cdot\vec{f} = \left(-\sin\tfrac{\theta}{2}\,\vec{u} + \cos\tfrac{\theta}{2}\,\vec{v}\right)\cdot\vec{v} = L^2\cos\tfrac{\theta}{2}$$

$$\vec{g}\times\vec{f} = \left(-\sin\tfrac{\theta}{2}\,\vec{u} + \cos\tfrac{\theta}{2}\,\vec{v}\right)\times\vec{v} = -L^2\sin\tfrac{\theta}{2}\,\vec{\omega}$$

($\vec{u}\cdot\vec{v} = 0$ より)
($\vec{u}\times\vec{v} = L^2\vec{\omega}$ より)

より、

$$\vec{g}\,\vec{f} = -\vec{g}\cdot\vec{f} + \vec{g}\times\vec{f} = -L^2\left(\cos\tfrac{\theta}{2} + \vec{\omega}\,\sin\tfrac{\theta}{2}\right)$$

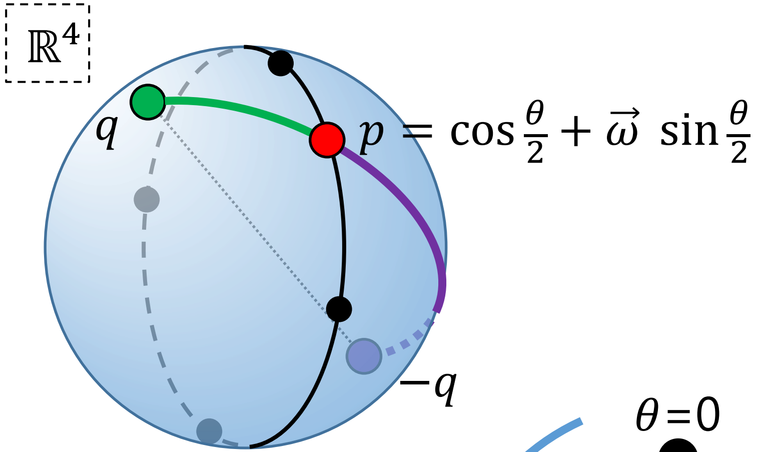
- 右側 $\vec{f}^{-1}\,\vec{g}^{-1} = \frac{\vec{f}\,\vec{g}}{L^4}$ についても同様： $\vec{f}^{-1}\,\vec{g}^{-1} = -L^{-2}\left(\cos\frac{\theta}{2} - \vec{\omega}\,\sin\frac{\theta}{2}\right)$
- 以上から、公式が導かれる：

$$R_{\vec{\omega},\theta}(\vec{x}) = R_{\vec{g}}\left(R_{\vec{f}}(\vec{x})\right) = \left(\cancel{-L^2}\left(\cos\tfrac{\theta}{2} + \vec{\omega}\,\sin\tfrac{\theta}{2}\right)\right)\vec{x}\left(\cancel{-L^{-2}}\left(\cos\tfrac{\theta}{2} - \vec{\omega}\,\sin\tfrac{\theta}{2}\right)\right)$$

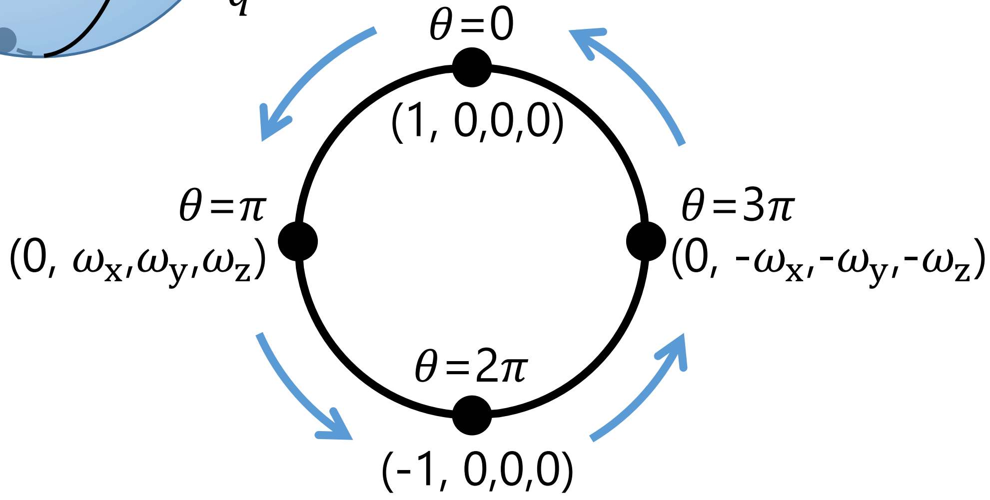
$$= \boxed{\left(\cos\tfrac{\theta}{2} + \vec{\omega}\,\sin\tfrac{\theta}{2}\right)\vec{x}\left(\cos\tfrac{\theta}{2} - \vec{\omega}\,\sin\tfrac{\theta}{2}\right)}$$

クォータニオンによる姿勢の表現と補間

- 任意の回転 (姿勢) は、単位クォータニオンとして表せる
  - 4D空間の単位超球面 (hypersphere) 上の点



- $\vec{\omega}$ を固定して θ を変化させると、4D空間の単位円が得られる



- ある姿勢を、ある軸の周りで一回転させた姿勢は、別のクォータニオンとして表される
  - 一つの姿勢に、二つのクォータニオンが対応する (double cover)
- Hypersphere上の2点 p, q を結ぶ測地線は、姿勢の補間を表す
  - q と $-q$ のうち、p と近い方 (4D内積が正となる方) を選ぶべし

クォータニオンを正規化する？しない？

- 任意のクォータニオンは、単位クォータニオンのスカラー倍として表せる

$$q = r\left(\cos\frac{\theta}{2} + \vec{\omega}\ \sin\frac{\theta}{2}\right), \qquad q^{-1} = r^{-1}\left(\cos\frac{\theta}{2} - \vec{\omega}\ \sin\frac{\theta}{2}\right)$$

- 回転の式では、スカラー倍の効果がキャンセルされる

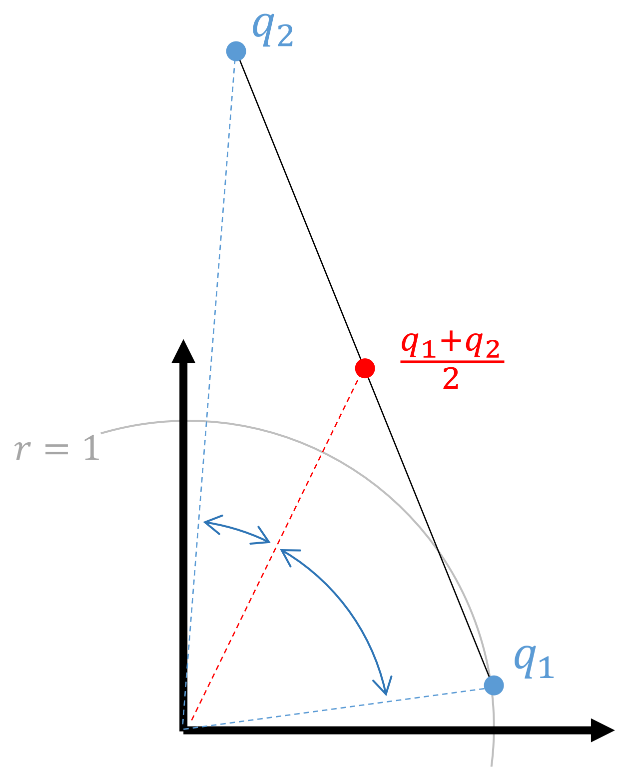
$$q\,\vec{x}\,q^{-1} = \cancel{r}\left(\cos\frac{\theta}{2} + \vec{\omega}\ \sin\frac{\theta}{2}\right)\vec{x}\,\cancel{r}^{-1}\left(\cos\frac{\theta}{2} - \vec{\omega}\ \sin\frac{\theta}{2}\right) = \left(\cos\frac{\theta}{2} + \vec{\omega}\ \sin\frac{\theta}{2}\right)\vec{x}\left(\cos\frac{\theta}{2} - \vec{\omega}\ \sin\frac{\theta}{2}\right)$$

➔ では、正規化しなくても良い？

- 実際の座標変換の計算に、クォータニオン積をそのまま使うことは無い (計算効率が悪い)
  - 普通に回転軸と回転角を使って直接計算する

    $$(\vec{x} - (\vec{\omega}\cdot\vec{x})\vec{\omega})\cos\theta + (\vec{\omega}\times\vec{x})\sin\theta + (\vec{\omega}\cdot\vec{x})\vec{\omega}$$

  - 回転軸と回転角は、正規化して初めて得られる
- 補間をする際も、正規化されていないと望ましくない結果になり得る

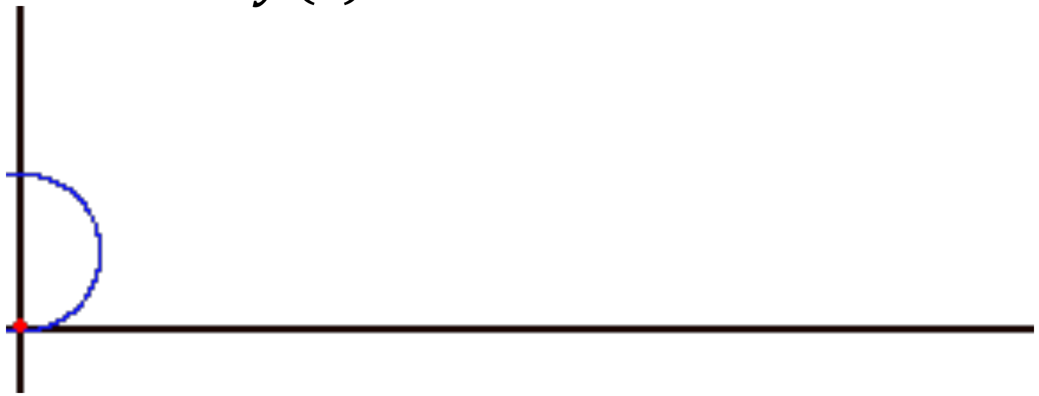


曲線のモデリング

パラメトリック曲線

- X座標とY座標がパラメタ t (≒時刻) によって決まるもの
  - 例：サイクロイド

$$x(t) = t - \sin t$$
$$y(t) = 1 - \cos t$$

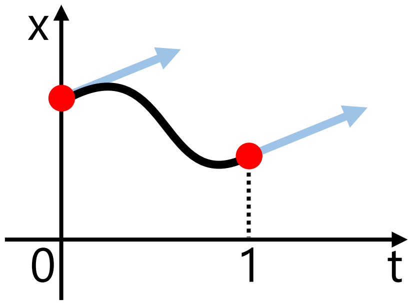


- 接線ベクトル：$\left(x'(t), y'(t)\right)$
- 多項式曲線： $x(t) = \sum_i a_i t^i$

3次エルミート曲線

- 両端での値と微分の制約を満たす
  (＝エルミート補間) ような、多項式曲線

$$
\begin{aligned}
x\,(0) &= x_0 \\
x\,(1) &= x_1 \\
x'(0) &= x_0' \\
x'(1) &= x_1'
\end{aligned}
$$



- 制約が4つなので、4自由度が必要
  ➔ 4個の係数 ➔ 3次多項式
  - $x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
  - $x'(t) = a_1 + 2a_2 t + 3a_3 t^2$

$$
\begin{aligned}
x\,(0) &= a_0 &&= x_0 \\
x\,(1) &= a_0 + a_1 + a_2 + a_3 &&= x_1 \\
x'(0) &= a_1 &&= x_0' \\
x'(1) &= a_1 + 2\,a_2 + 3\,a_3 &&= x_1'
\end{aligned}
$$

➔

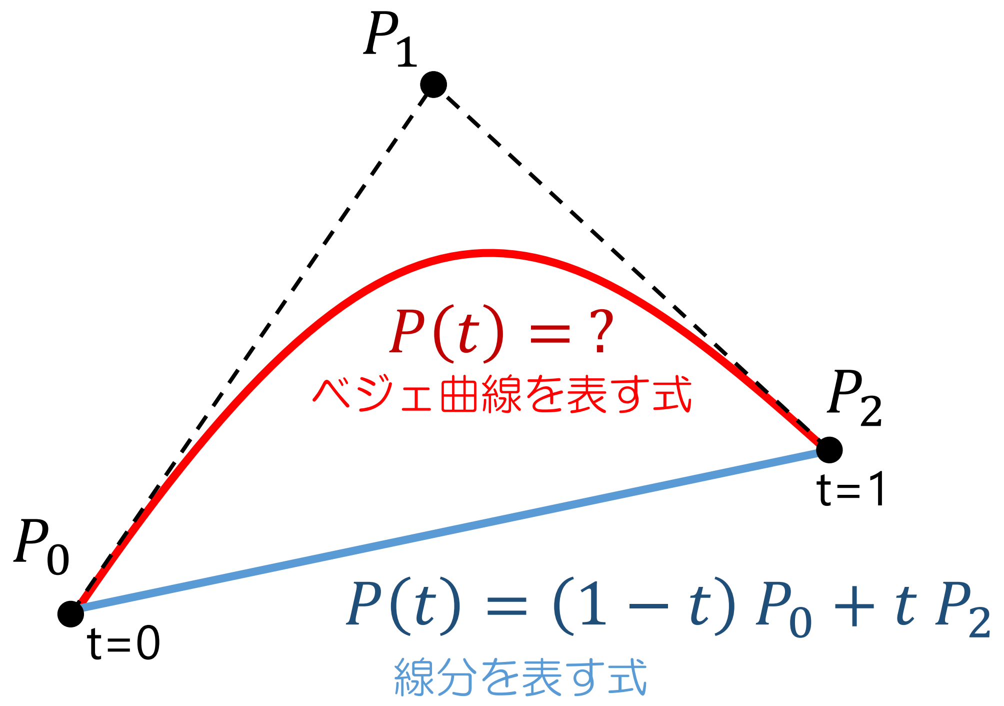
$$
\begin{aligned}
a_0 &= x_0 \\
a_1 &= x_0' \\
a_2 &= -3\,x_0 + 3\,x_1 - 2\,x_0' - x_1' \\
a_3 &= 2\,x_0 - 2\,x_1 + x_0' + x_1'
\end{aligned}
$$

- 制約値を代入すれば係数が求まる

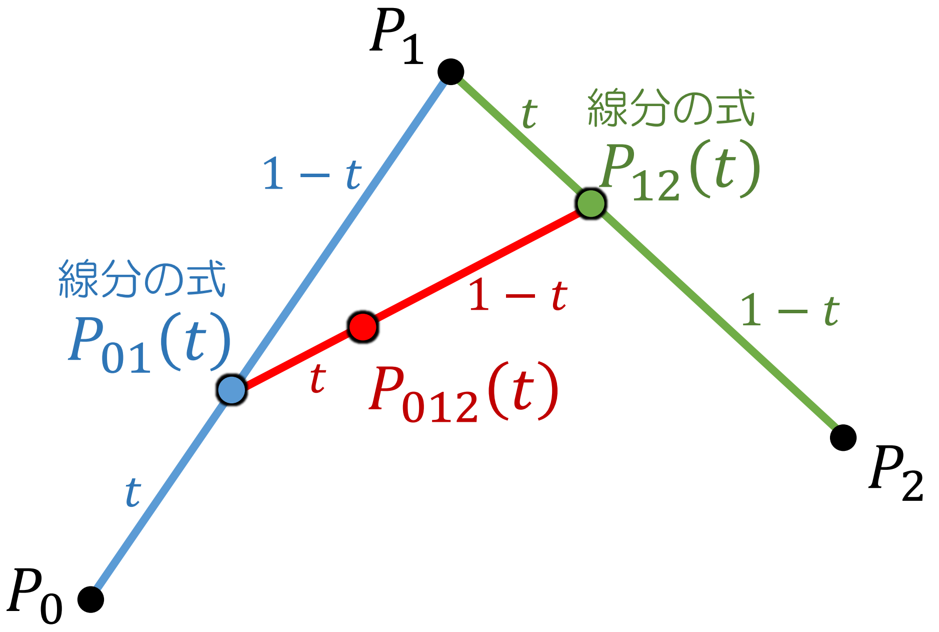
ベジェ (Bezier[ベズィエ]) 曲線

- 入力：3点P_0, P_1, P_2 (制御点)
  - 任意空間 (2D, 3D) の座標

- 求めたいもの：曲線$P(t)$で
  $$P(0) = P_0$$
  $$P(1) = P_2$$
  を満たしつつ、
  P_1に “引っ張られる” もの



ベジェ曲線



- $P_{01}(t) = (1-t)P_0 + t\,P_1$
- $P_{12}(t) = (1-t)P_1 + t\,P_2$
  - $P_{01}(0) = P_0$
  - $P_{12}(1) = P_2$

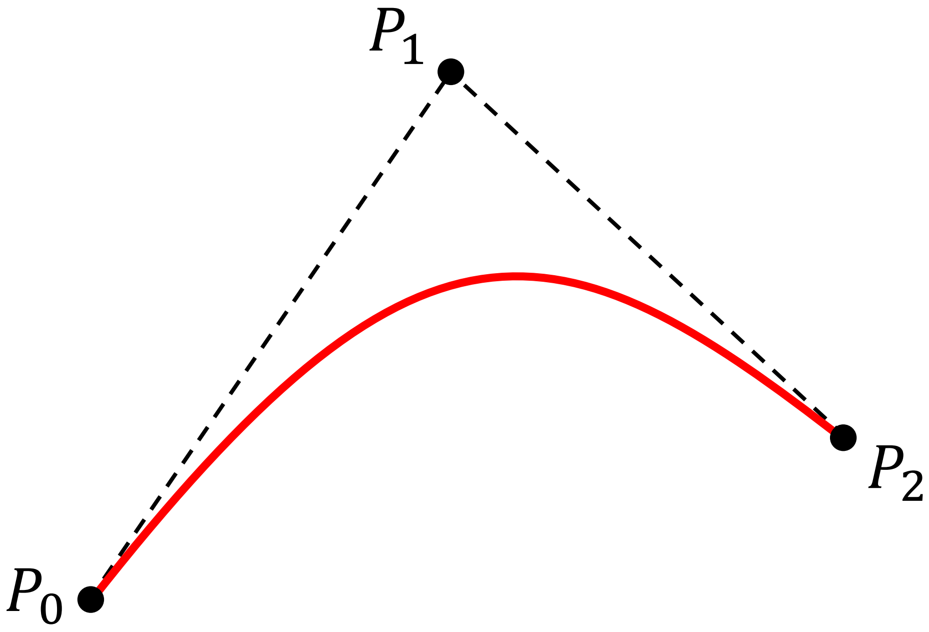
- アイディア： "補間を補間"
  $t: 0 \to 1$ のとき $P_{01} \to P_{12}$ となるように線形補間

- $P_{012}(t) = (1-t)P_{01}(t) + t\,P_{12}(t)$

$$= (1-t)\{(1-t)P_0 + t\,P_1\} + t\,\{(1-t)P_1 + t\,P_2\}$$

$$= \underbrace{(1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2}_{\text{2次ベジェ曲線}}$$

ベジェ曲線



- $P_{01}(t) = (1-t)P_0 + t\,P_1$
- $P_{12}(t) = (1-t)P_1 + t\,P_2$
    - $P_{01}(0) = P_0$
    - $P_{12}(1) = P_2$

- アイディア： "補間を補間"
  $t : 0 \rightarrow 1$ のとき $P_{01} \rightarrow P_{12}$ となるように線形補間

- $P_{012}(t) = (1-t)P_{01}(t) + t\,P_{12}(t)$

$$= (1-t)\{(1-t)P_0 + t\,P_1\} + t\,\{(1-t)P_1 + t\,P_2\}$$

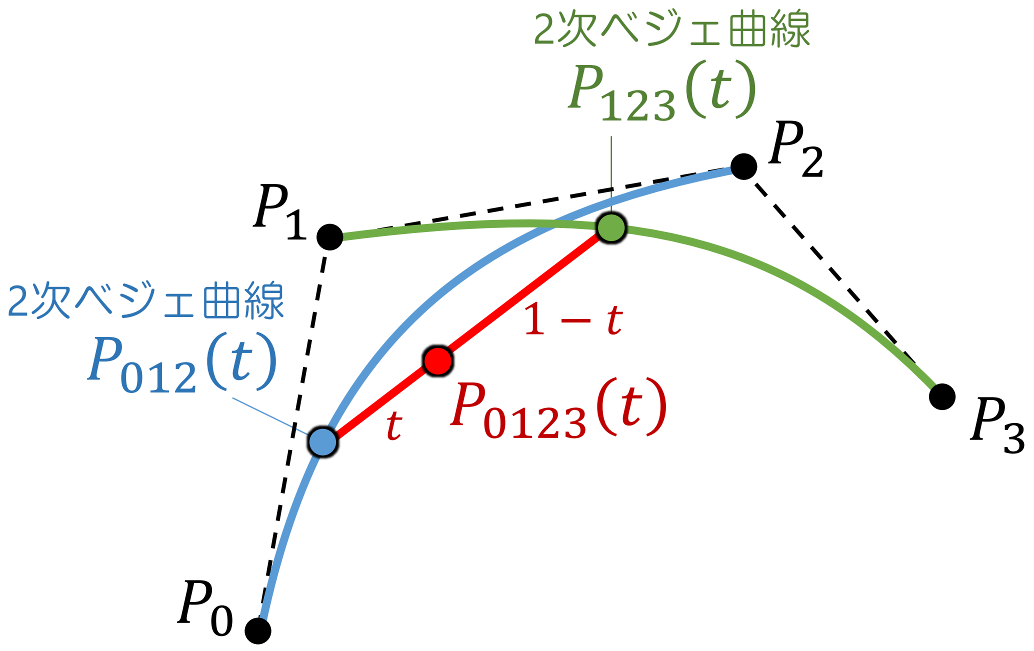
$$= \underbrace{(1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2}_{\text{2次ベジェ曲線}}$$

3次ベジェ曲線

- 全く同じ考え方を4点$P_0, P_1, P_2\ P_3$に適用：

  $t : 0 \to 1$ のとき $P_{012} \to P_{123}$

  となるように線形補間



- $P_{0123}(t) = (1-t)P_{012}(t) + t\, P_{123}(t)$

$$= (1-t)\{(1-t)^2P_0 + 2t(1-t)P_1 + t^2P_2\} + t\,\{(1-t)^2P_1 + 2t(1-t)P_2 + t^2P_3\}$$

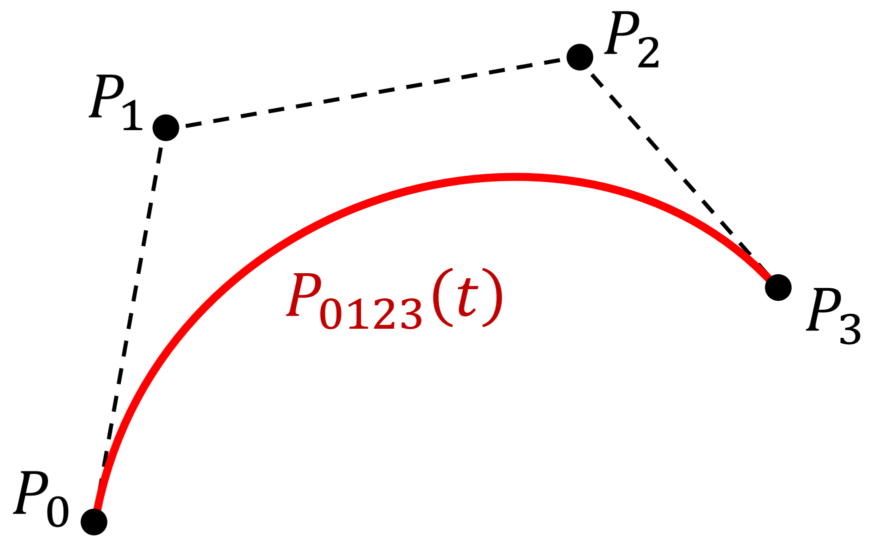
$$= \underbrace{(1-t)^3P_0 + 3t(1-t)^2P_1 + 3t^2(1-t)P_2 + t^3P_3}_{\text{3次ベジェ曲線}}$$

3次ベジェ曲線

- 全く同じ考え方を4点P_0, P_1, P_2 P_3に適用：

  $t : 0 \rightarrow 1$ のとき $P_{012} \rightarrow P_{123}$

  となるように線形補間



- $P_{0123}(t) = (1-t)P_{012}(t) + t\,P_{123}(t)$

$$= (1-t)\{(1-t)^2P_0 + 2t(1-t)P_1 + t^2P_2\} + t\,\{(1-t)^2P_1 + 2t(1-t)P_2 + t^2P_3\}$$

$$= \underbrace{(1-t)^3P_0 + 3t(1-t)^2P_1 + 3t^2(1-t)P_2 + t^3P_3}_{\text{3次ベジェ曲線}}$$

- 両端における接線の制御がしやすい➔CGで頻繁に使われる

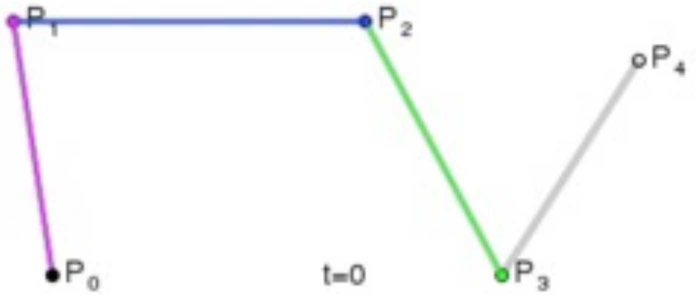
n次ベジェ曲線

- 入力：n+1個の制御点$P_0, \cdots, P_n$

$$P(t) = \sum_{i=0}^{n} \underbrace{{}_n\mathrm{C}_i\, t^i\, (1-t)^{n-i}}_{b_i^n(t)} P_i$$

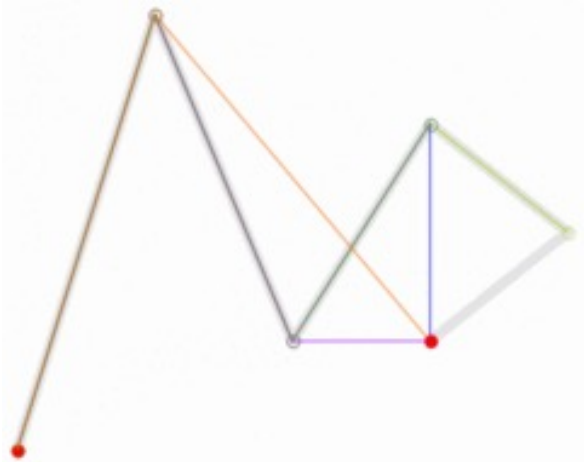
バーンスタイン基底関数

4次ベジェ曲線



$$\begin{aligned}&(1-t)^4 P_0 &+\\ &4t(1-t)^3 P_1 &+\\ &6t^2(1-t)^2 P_2 &+\\ &4t^3(1-t) P_3 &+\\ &t^4 P_4\end{aligned}$$

5次ベジェ曲線

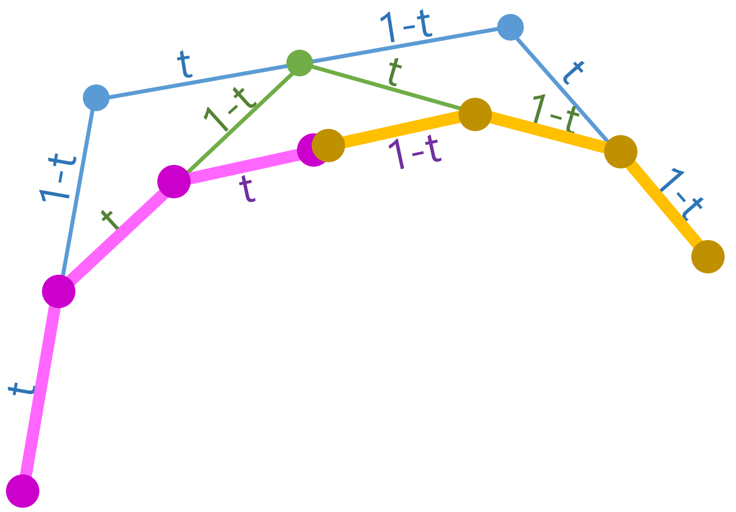


$$\begin{aligned}&(1-t)^5 P_0 &+\\ &5t(1-t)^4 P_1 &+\\ &10t^2(1-t)^3 P_2 &+\\ &10t^3(1-t)^2 P_3 &+\\ &5t^4(1-t) P_4 &+\\ &t^5 P_5\end{aligned}$$

ベジェ曲線の評価方法

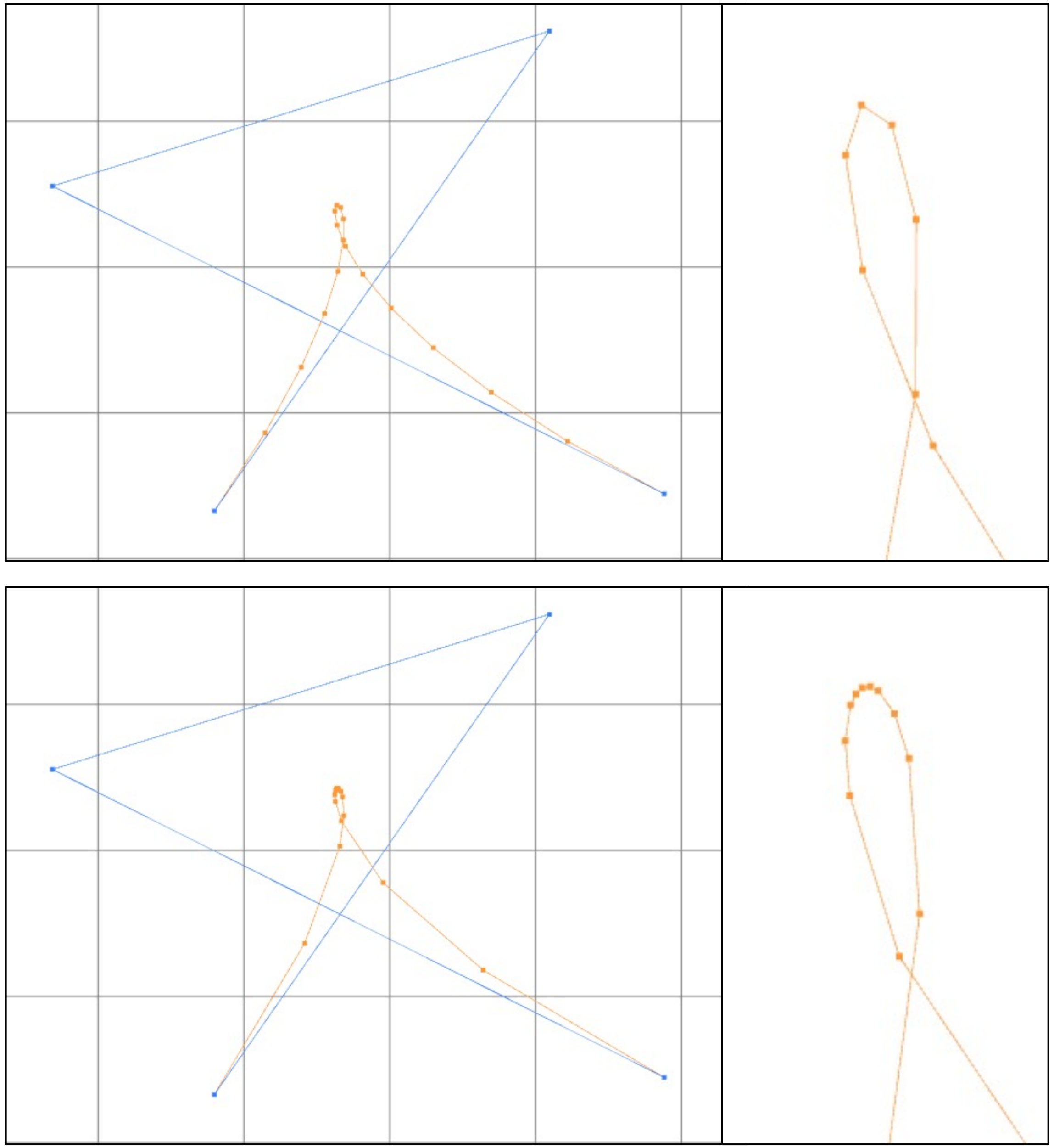
- 方法1：多項式をそのまま評価する
  - 単純で速いが、数値的に不安定になる場合も

- 方法2：ド・カステリョ [de Casteljau] のアルゴリズム
  - ベジェ曲線の再帰的な定義そのものに倣う
  - 計算手順は増えるが、数値的に安定
  - ベジェ曲線の分割にも使える



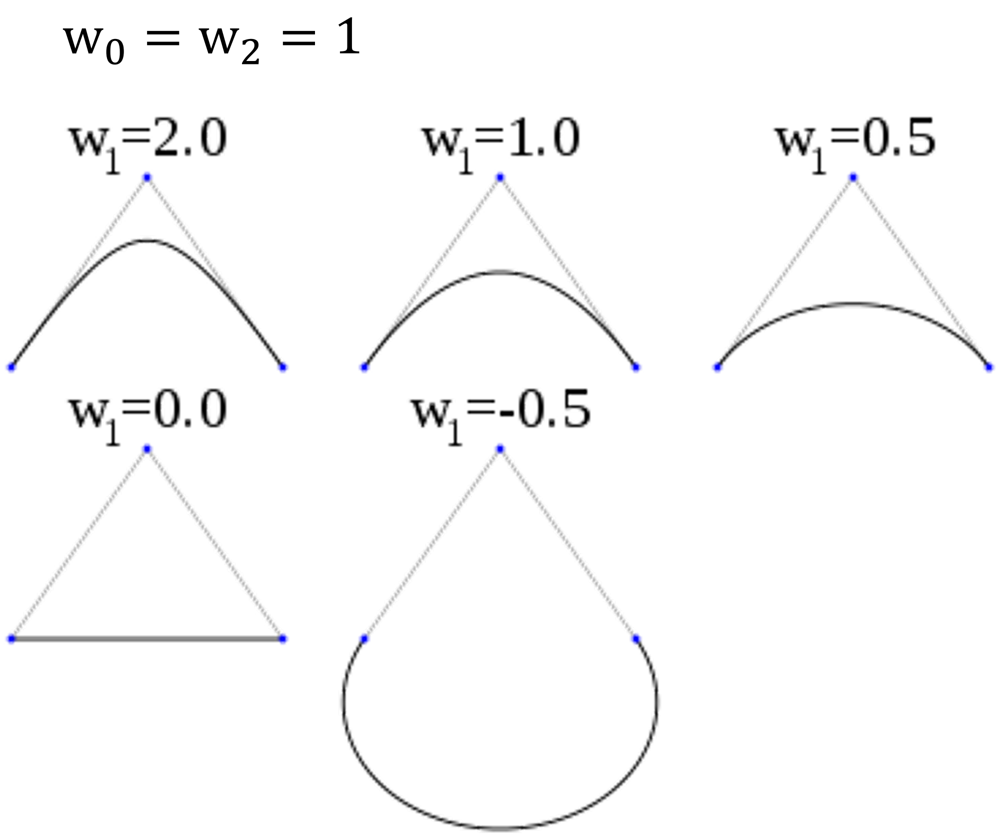
ベジェ曲線の描画方法

- 最終的には折れ線で近似的に描く
  - パラメタtをどうサンプリングするかが問題
- 方法1：一定間隔でサンプリング
  - 実装が簡単
  - サンプル点の密度が必要十分でなくなる？
- 方法2：適応的なサンプリング
  - 制御点列が直線状でなければde Casteljauの方法で分割する



さらなる制御法：有理ベジェ曲線

- ベジェ曲線は、制御点の "重み付き平均" と見ることができる
  - $P_{012}(t) = (1-t)^2 P_0 + 2t(1-t) P_1 + t^2 \; P_2$
    $\qquad = \lambda_0(t) \; P_0 + \lambda_1(t) \; P_1 + \lambda_2(t) P_2$
  - 重要な性質：**partition of unity**
    $$\lambda_0(t) + \lambda_1(t) + \lambda_2(t) = 1 \qquad \forall t$$
- 各制御点の重み$\lambda_i(t)$に任意係数w_iを掛ける：
  $$\xi_i(t) = w_i \, \lambda_i(t)$$
- 正規化して新しい重みを得る：
  $$\lambda_i'(t) = \frac{\xi_i(t)}{\sum_j \xi_j(t)}$$



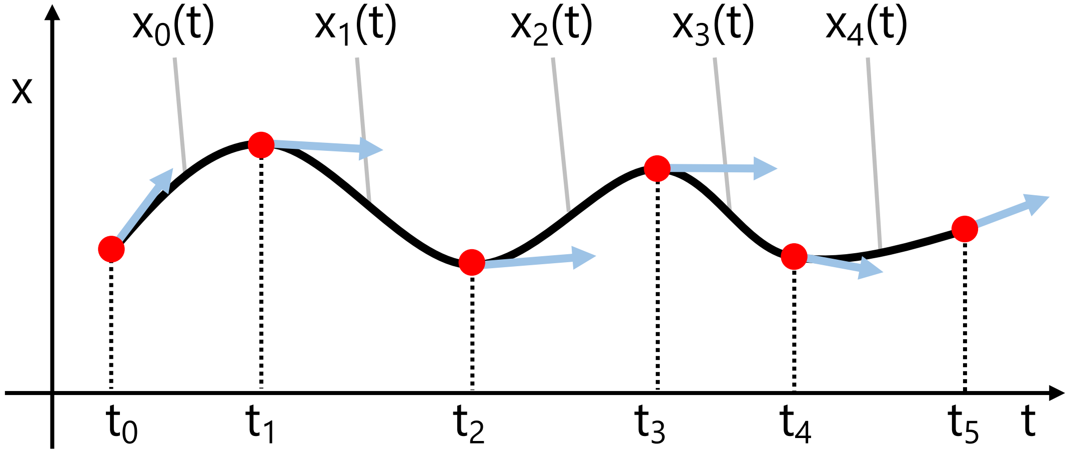
多項式曲線ではない ➔ 円弧なども表現可能

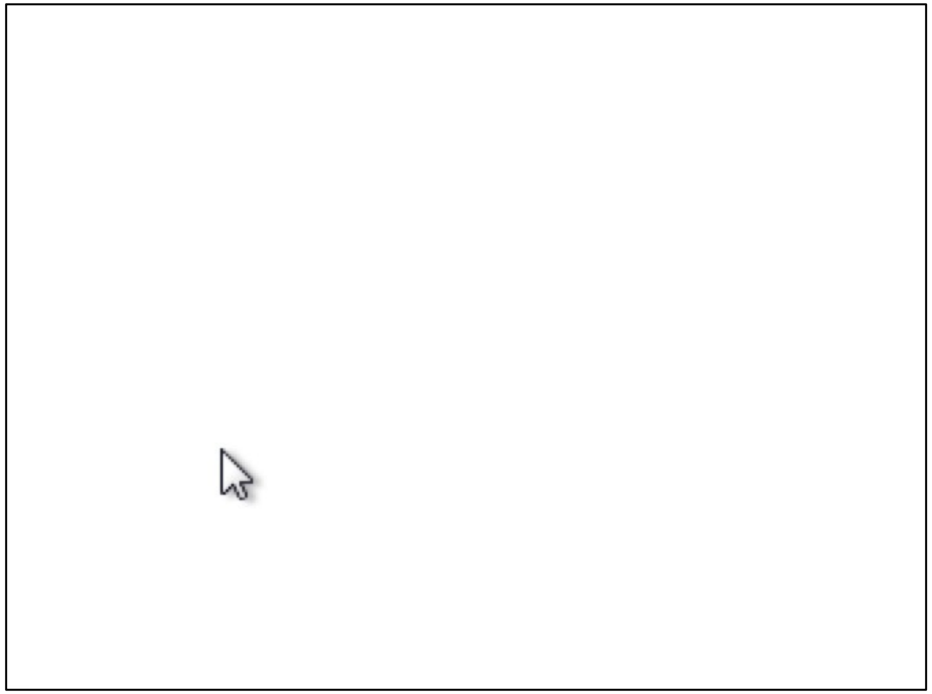
3次スプライン

- 複数の3次曲線を滑らかに繋げたもの
  - 区分的多項式
  - 区分の境目で値と微分値が共通 (C^1連続)

- パラメタ範囲は $[0, 1]$ とは限らない
  - ただし $t_k < t_{k+1}$

- 値のみを入力として、
  微分値を適切に自動推定したい

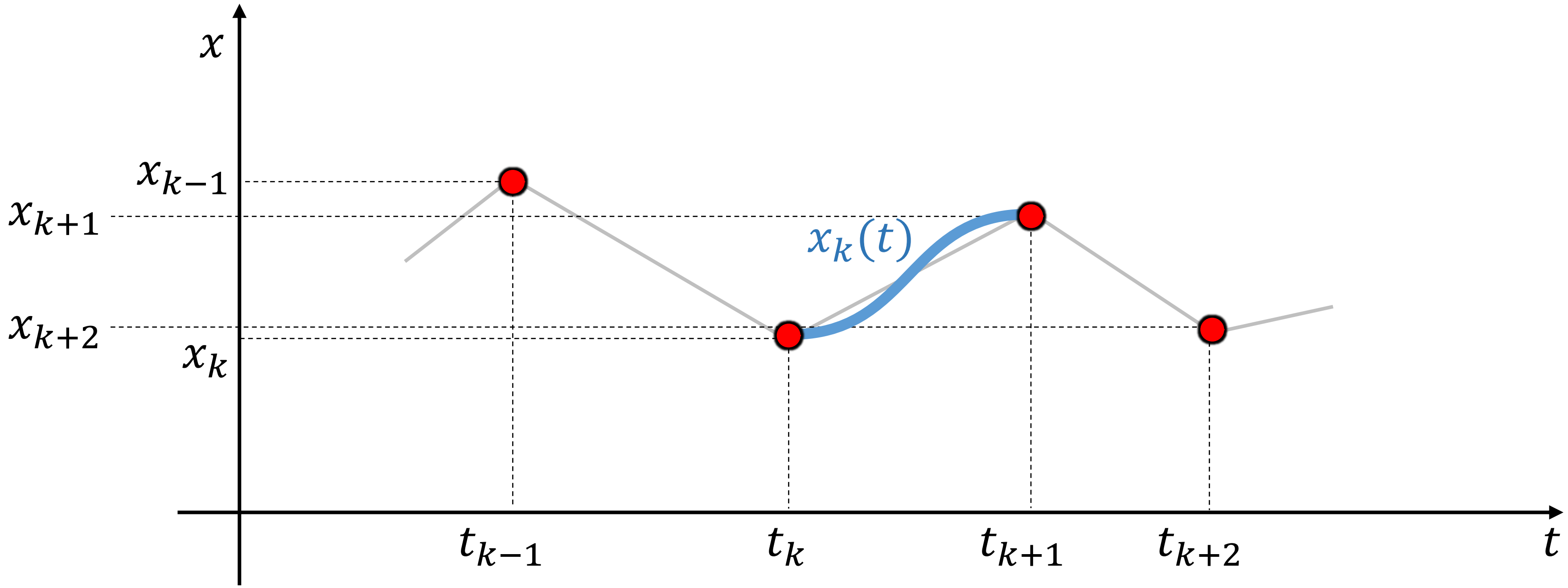




PowerPointの曲線ツール

3次Catmull-Romスプライン

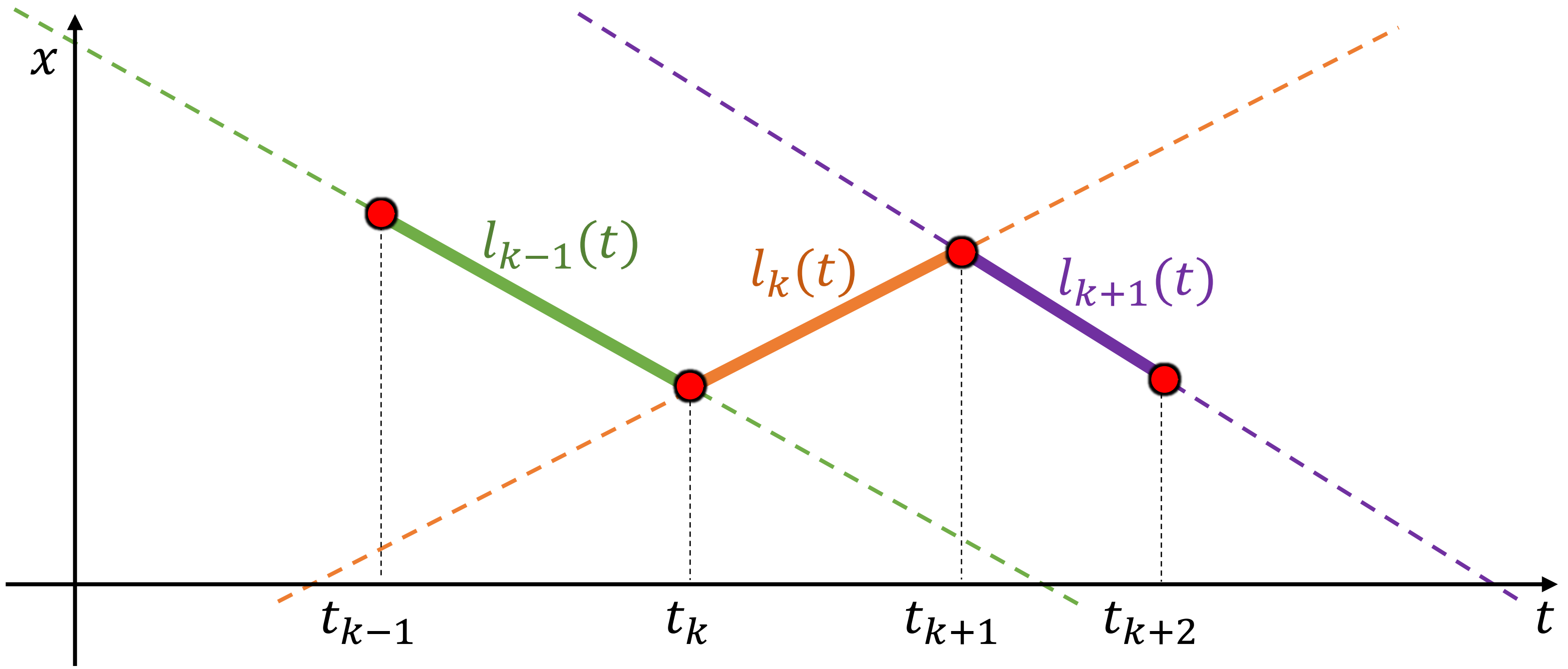
- 区間 $t_k \leq t \leq t_{k+1}$ における3次関数 $x_k(t)$ を、
  前後の制約値 $x_{k-1}, x_k, x_{k+1}, x_{k+2}$ から決定



3次Catmull-Romスプライン：ステップ1

- $t_k \rightarrow t_{k+1}$ のとき $x_k \rightarrow x_{k+1}$ となるように補間 ➔ 直線

$$l_k(t) = \left(1 - \frac{t \; - t_k}{t_{k+1} - t_k}\right) x_k + \frac{t \; - t_k}{t_{k+1} - t_k} x_{k+1}$$

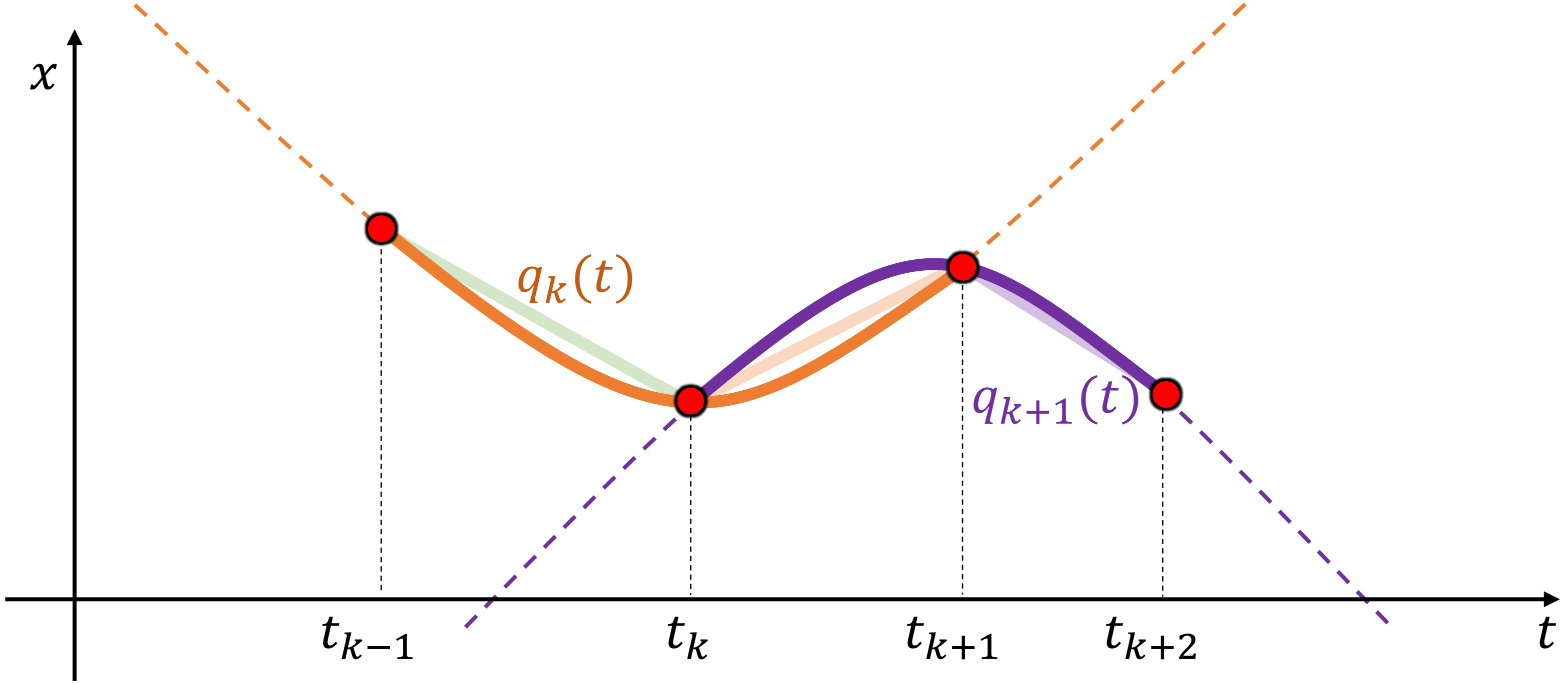


3次Catmull-Romスプライン：ステップ2

- $t_{k-1} \to t_{k+1}$ のとき $l_{k-1} \to l_k$ となるように補間 ➔ 2次曲線

$$q_k(t) = \left(1 - \frac{t - t_{k-1}}{t_{k+1} - t_{k-1}}\right) l_{k-1}(t) + \frac{t - t_{k-1}}{t_{k+1} - t_{k-1}} l_k(t)$$

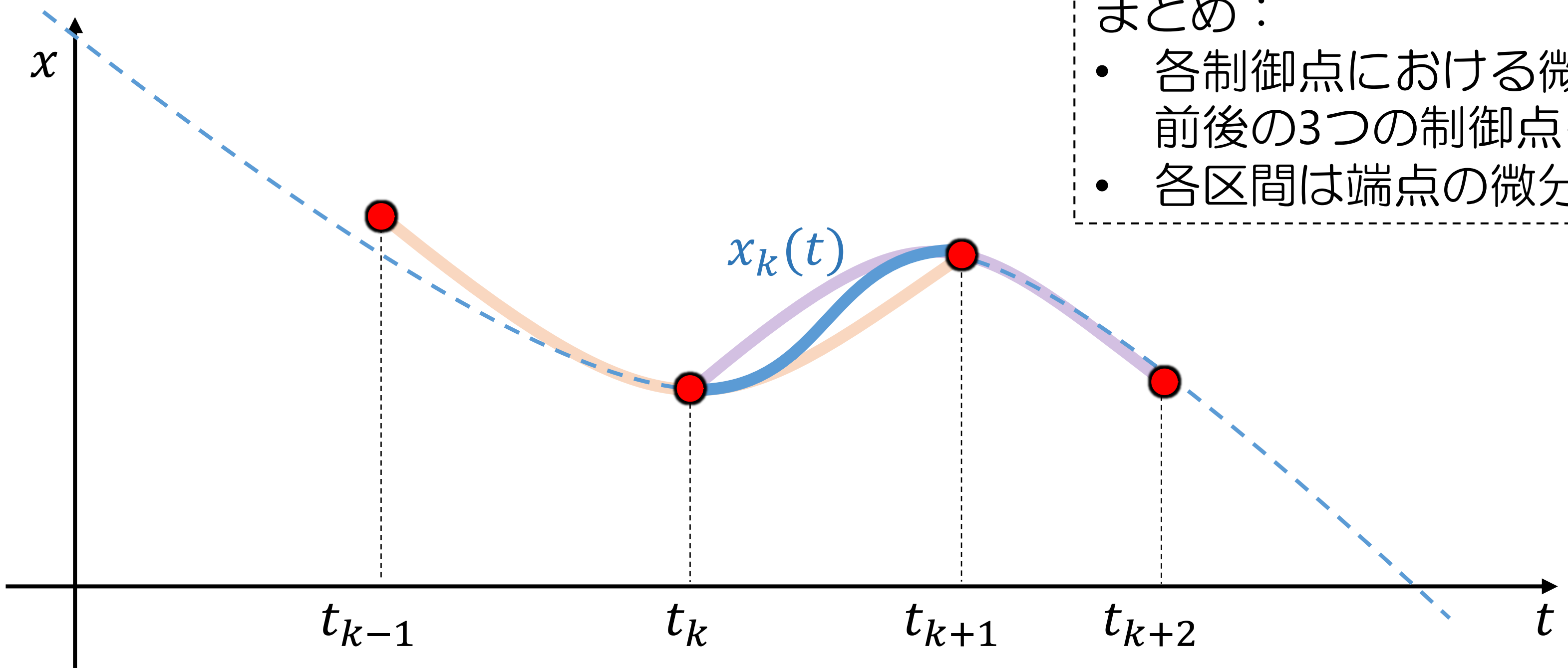
  - 3点 $(t_{k-1}, x_{k-1}), (t_k, x_k), (t_{k+1}, x_{k+1})$ を通る



3次Catmull-Romスプライン：ステップ3

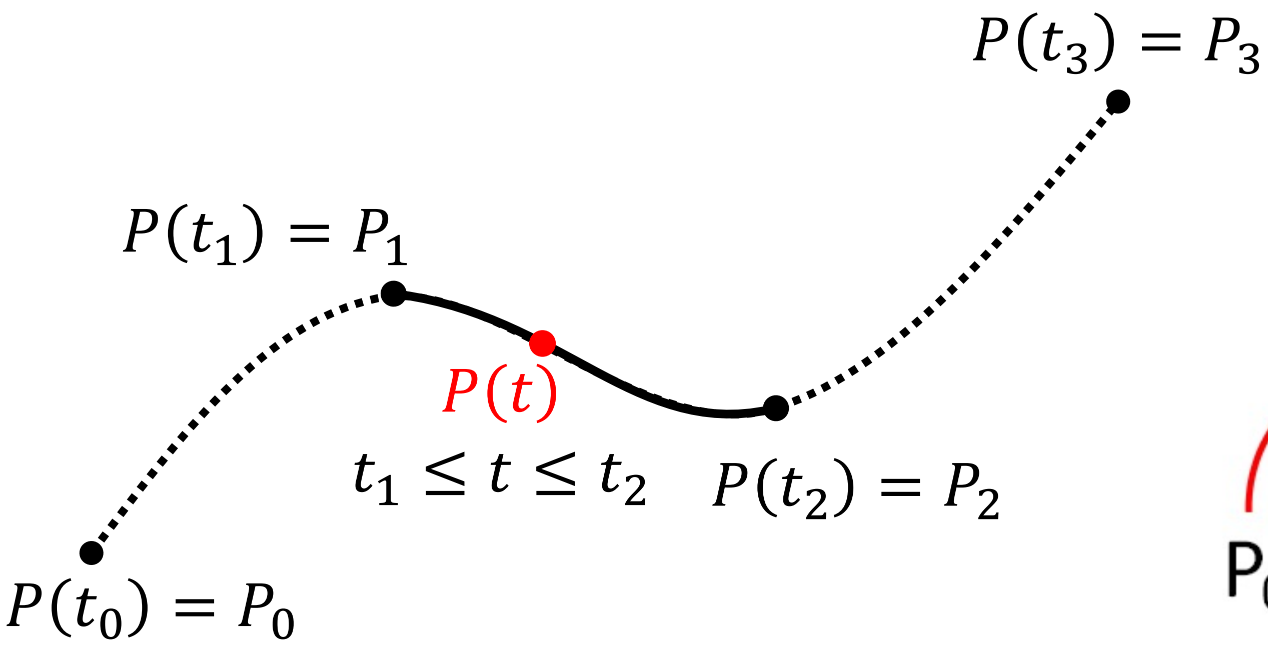
- $t_k \to t_{k+1}$ のとき $q_k \to q_{k+1}$ となるように補間 ➔ 3次曲線

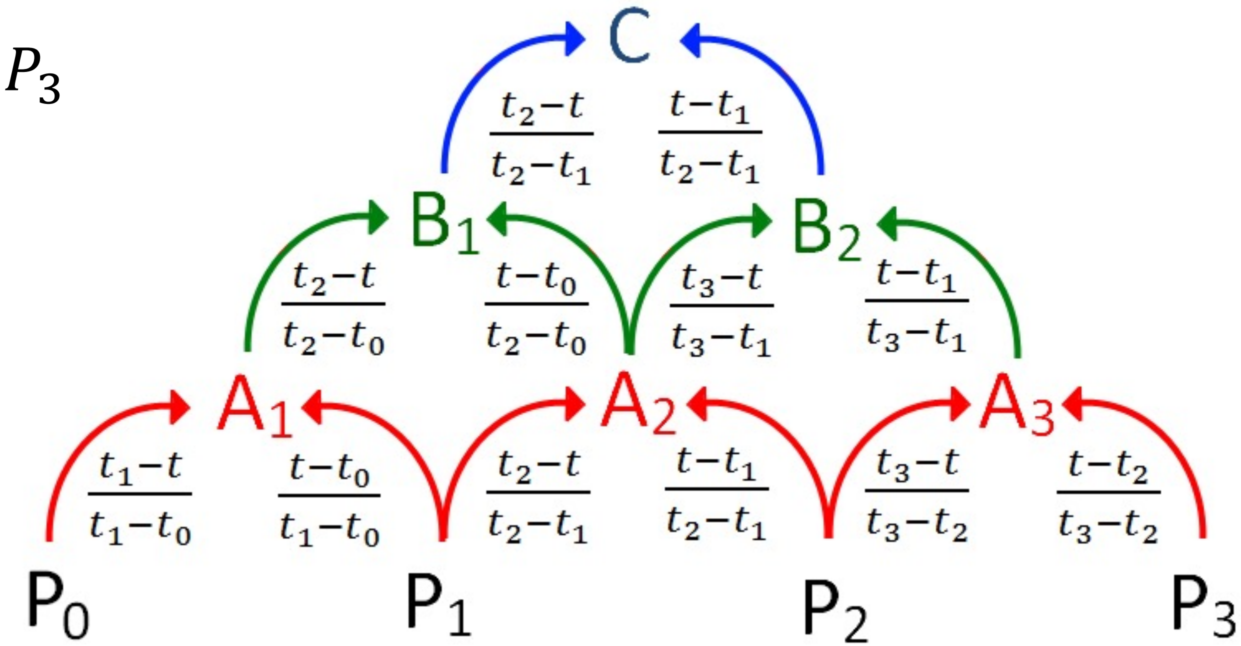
$$x_k(t) = \left(1 - \frac{t - t_k}{t_{k+1} - t_k}\right) q_k(t) + \frac{t - t_k}{t_{k+1} - t_k} q_{k+1}(t)$$



まとめ：
- 各制御点における微分値が、それ自身とその前後の3つの制御点を通る2次曲線から定まる
- 各区間は端点の微分値を満たす3次曲線

3次Catmull-Romスプラインの計算方法

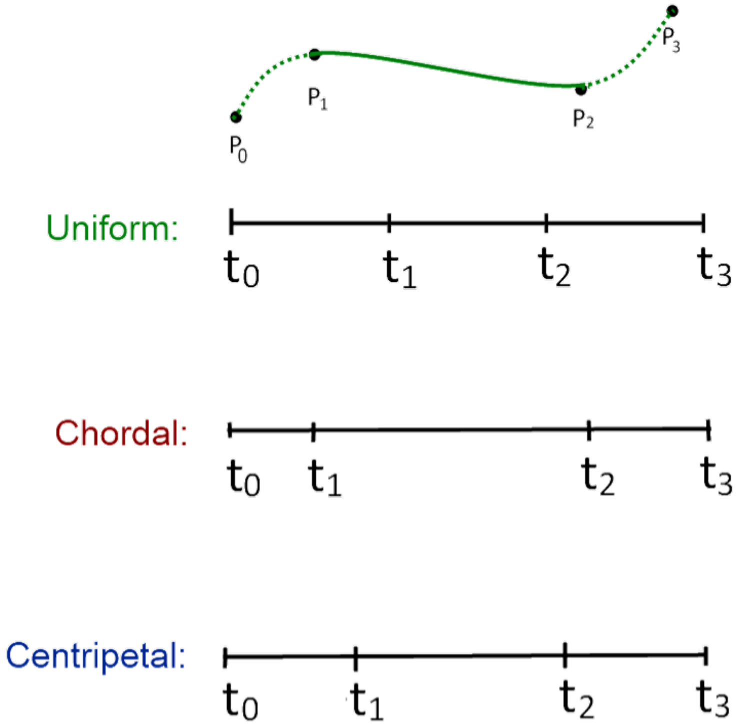


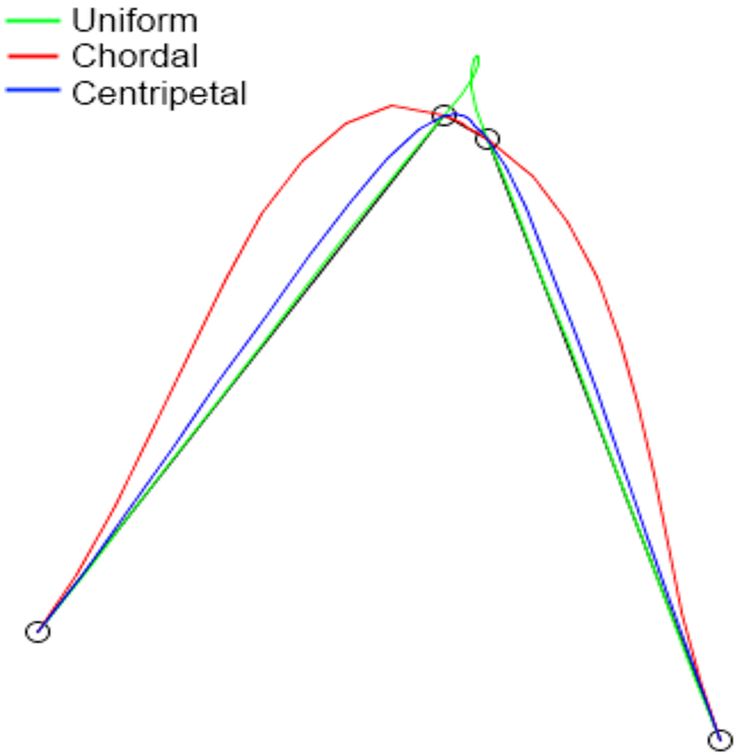


A recursive evaluation algorithm for a class of Catmull-Rom splines [Barry,Colgman,SIGGRAPH88]

パラメタ区分 t_k (ノット列) の決め方

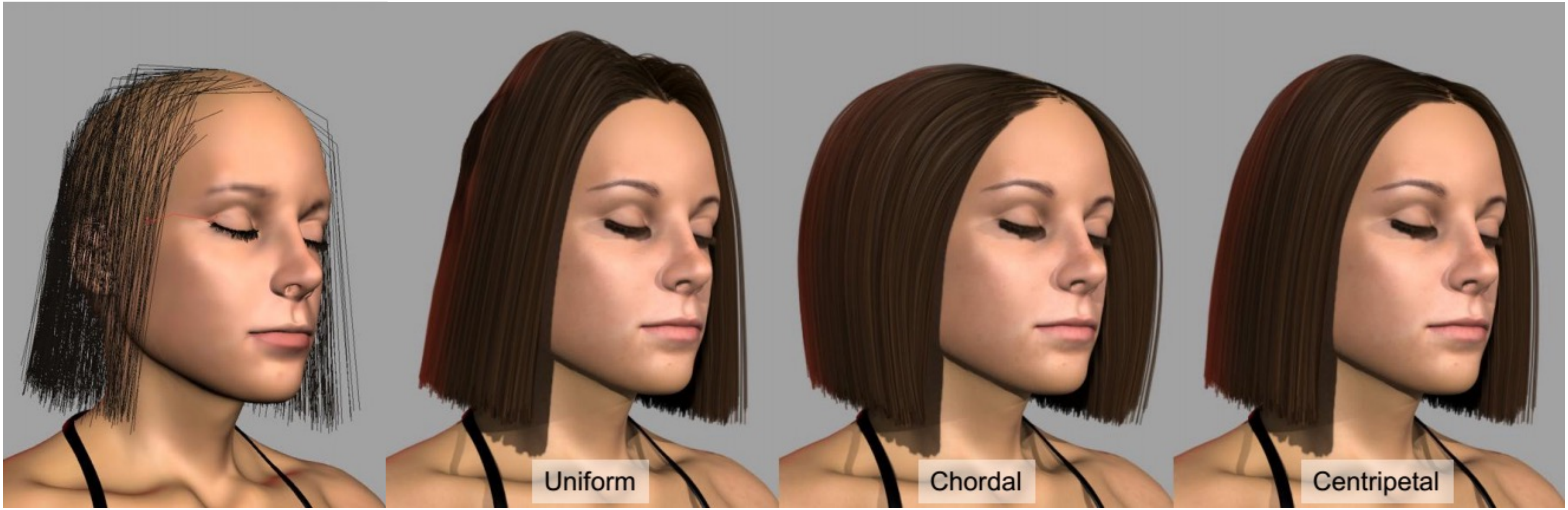
- 前提：$t_0 = 0$
- Uniform (一様)
  $$t_k = t_{k-1} + 1$$
- Chordal (弧長に基づく)
  $$t_k = t_{k-1} + |P_{k-1} - P_k|$$
- Centripetal (求心性？)
  $$t_k = t_{k-1} + \sqrt{|P_{k-1} - P_k|}$$





Parameterization of Catmull-Rom Curves [Yuksel,Schaefer,Keyser,CAD11]

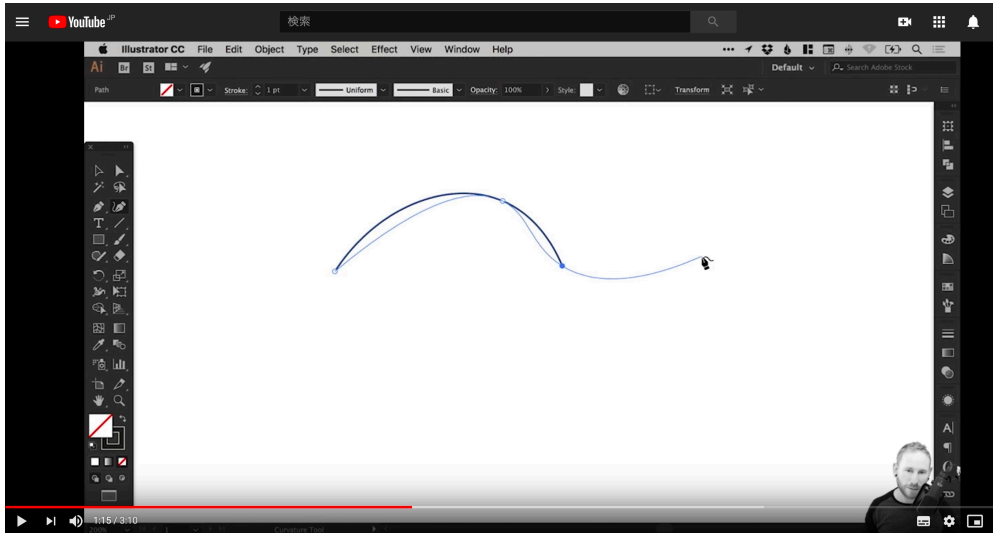
3次Catmull-Romスプラインの応用



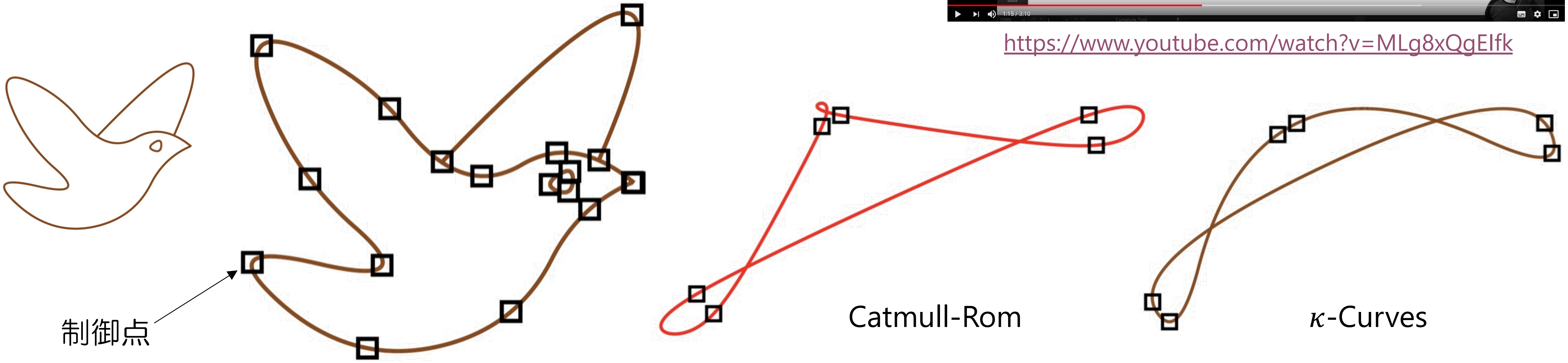
Parameterization of Catmull-Rom Curves [Yuksel,Schaefer,Keyser,CAD11]

最近の論文 (1): κ-Curves

- 大学と民間企業 (Adobe) の共同研究
- 特長
  - C^2 連続 (より滑らか)
  - 曲率最大の箇所が必ず制御点の上に来る

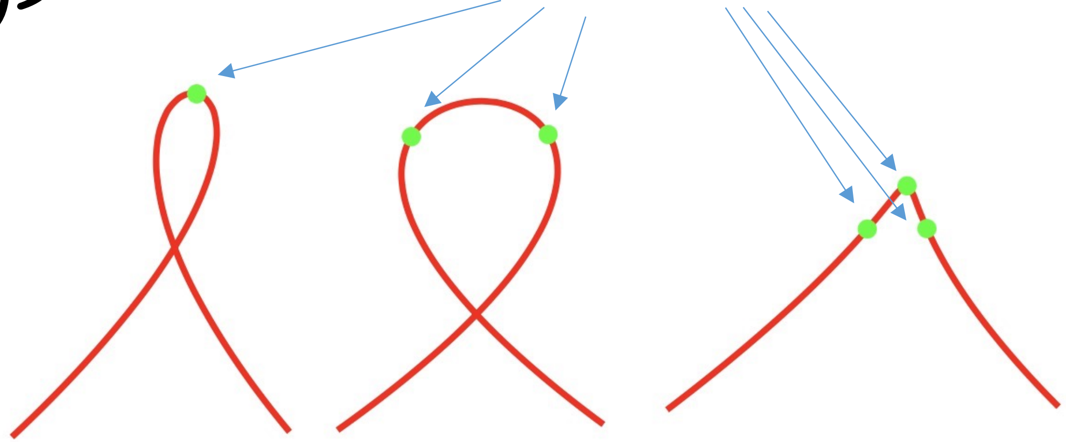


https://www.youtube.com/watch?v=MLg8xQgElfk



κ-Curves: Interpolation at Local Maximum Curvature [Yan, Schiller, Wilensky, Carr, Schaefer, SIGGRAPH 2017]

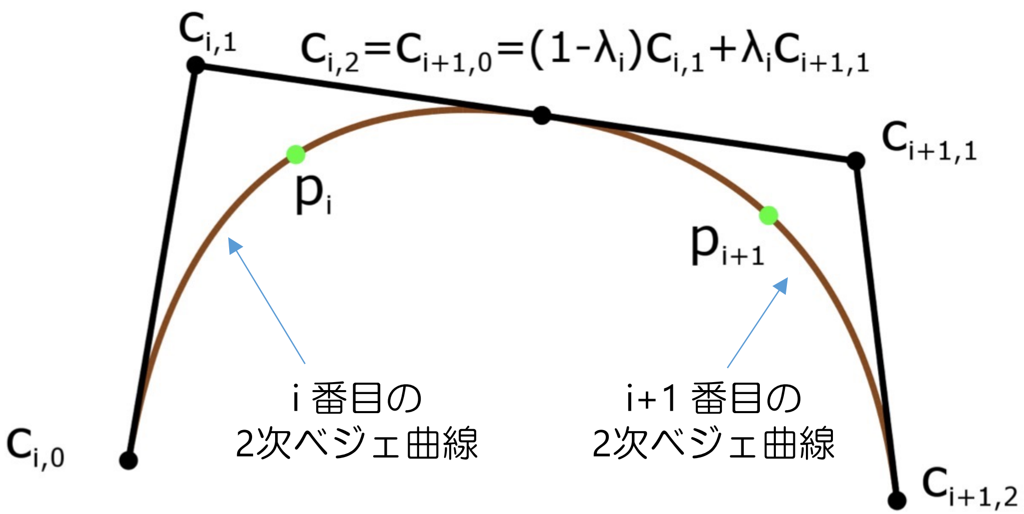
κ-Curves の基本アイディア

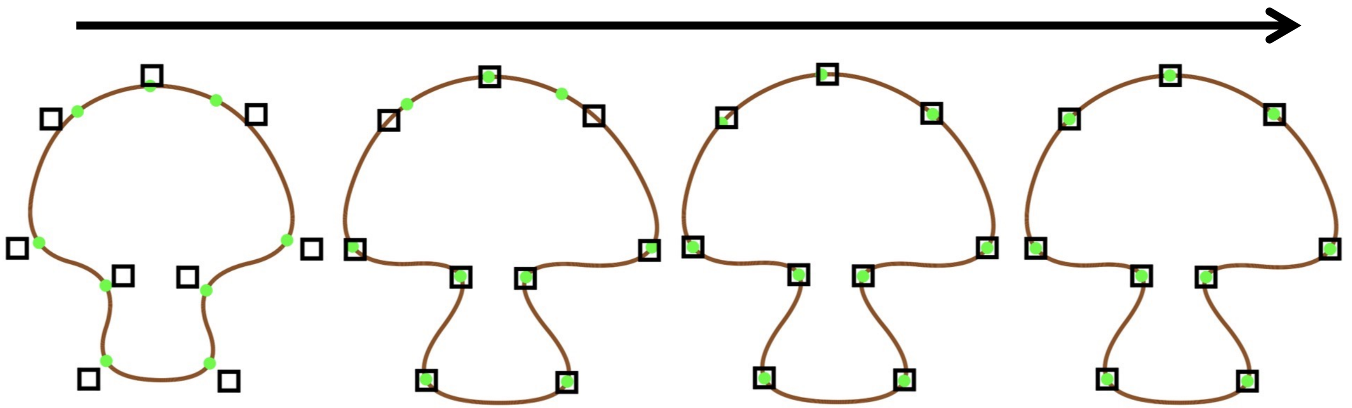


3次ベジェ曲線で起こり得る状態

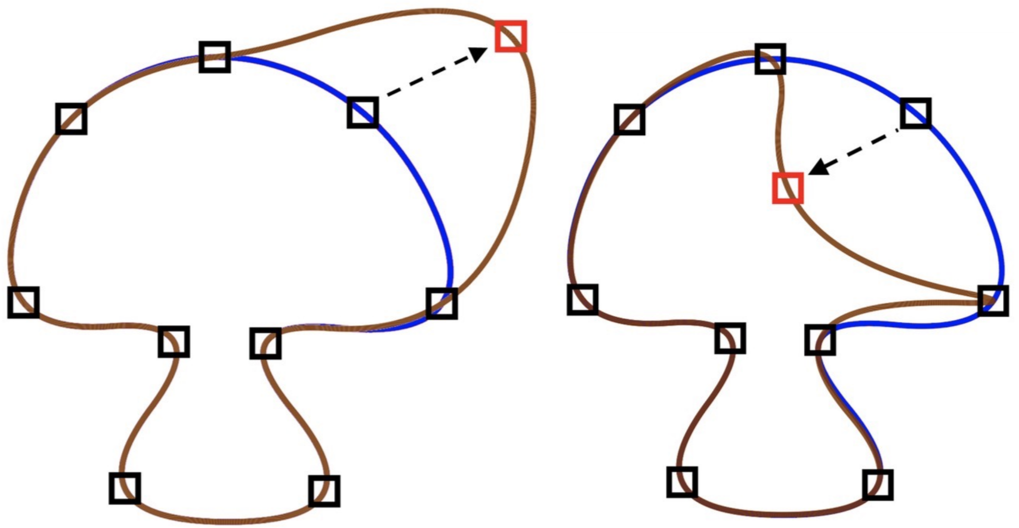
- 3次ベジェ曲線だと、
  曲率のコントロールが難しい

- 2次ベジェ曲線の方が実は使いやすい！
  - 曲率極大となるのが必ず高々1箇所
  - 曲率が極大となる箇所をユーザが指定し、ベジェ曲線の制御点の位置を逆算する

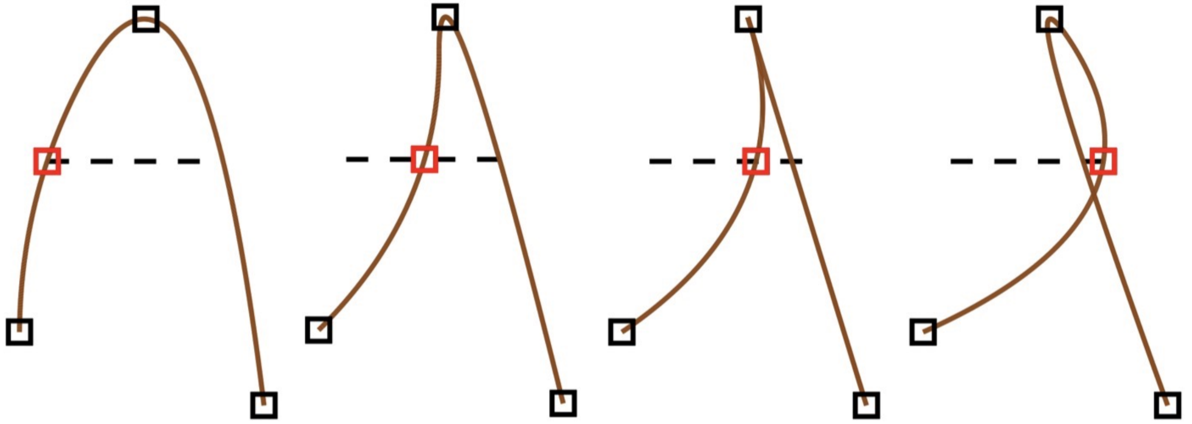




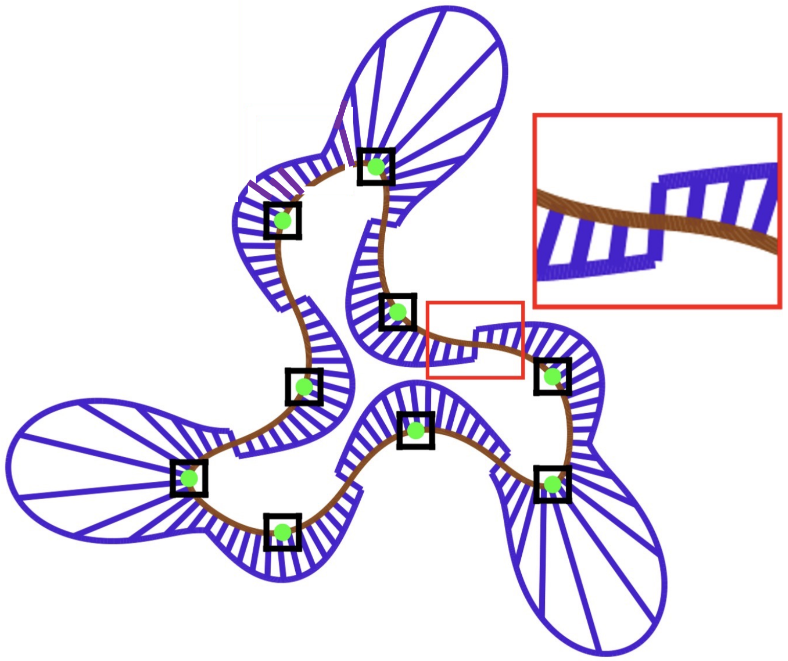
グローバルで非線形な最適化なので、繰り返し計算で解く



1個の制御点を動かすと、曲線全体が影響を受ける

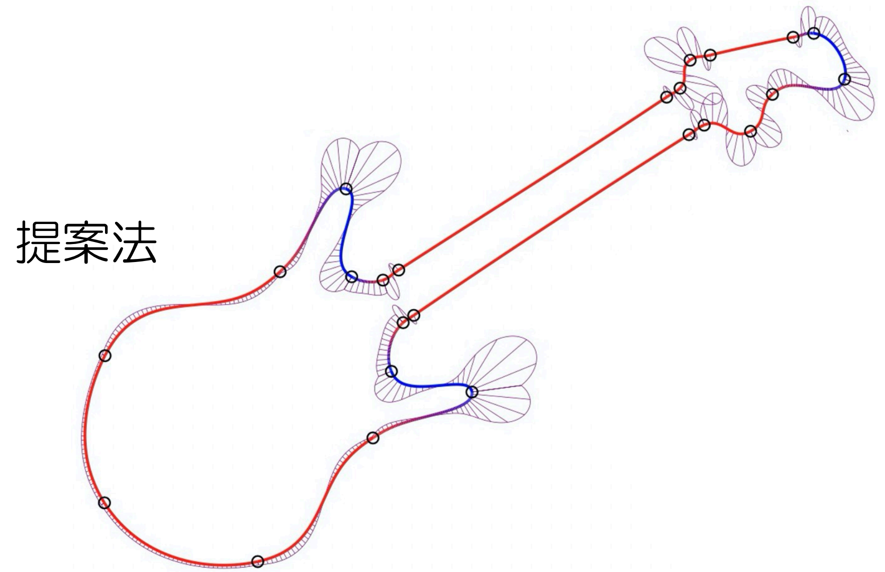


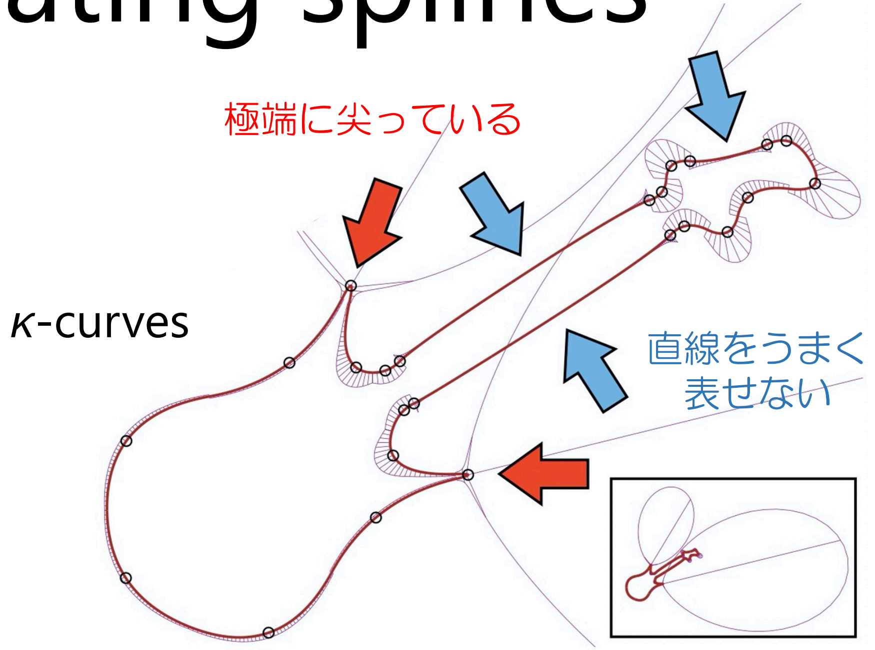
「ひっくり返り」は、必ず制御点の上で起こる



凹と凸の境目では、曲率は不連続になる

最近の論文 (2): C^2 interpolating splines





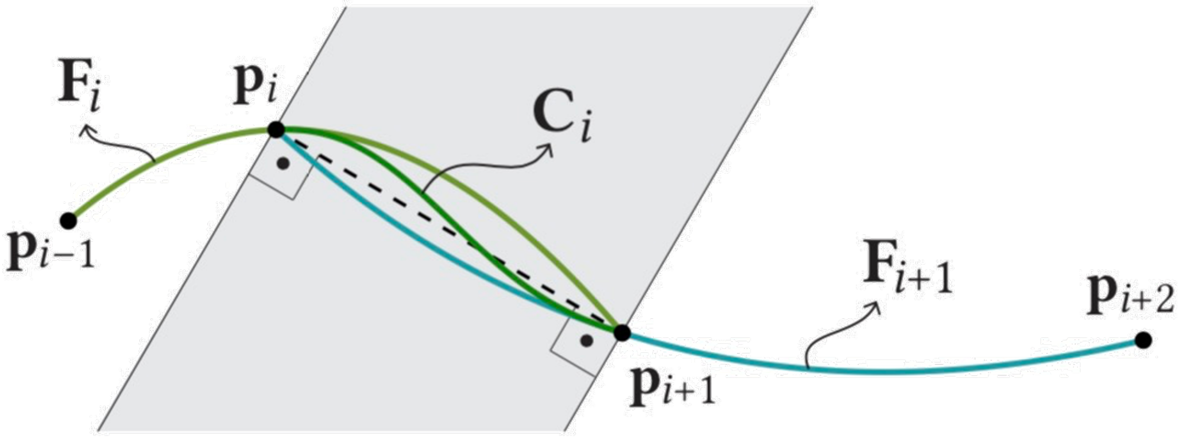
- κ-curvesの欠点：
  ☹ グローバルな最適化 (計算コスト高い)
  ☹ Global support (一点を動かしたら全体が動く)
  ☹ 円弧や直線を表現できない
  ☹ 3Dに拡張できない
- これらを克服するシンプルな手法を提案

Hair modelingに応用

A Class of C2 Interpolating Splines [Yuksel, SIGGRAPH 2020]

キーアイディア：

3点を通る補間関数 $\mathbf{F}_i$ と三角関数を組み合わせて、
各セグメント $\mathbf{C}_i(\theta)$ を定義



$$\boxed{\begin{array}{l} \mathbf{F}_i(0) = \mathbf{p}_{i-1}\ , \qquad \mathbf{F}_i(\tfrac{\pi}{2}) = \mathbf{p}_i\ , \qquad \mathbf{F}_i(\pi) = \mathbf{p}_{i+1} \\ \mathbf{C}_i(\theta) = \cos^2\theta\,\mathbf{F}_i(\theta + \tfrac{\pi}{2}) + \sin^2\theta\,\mathbf{F}_{i+1}(\theta) \qquad 0 \le \theta \le \dfrac{\pi}{2} \end{array}}$$

➔ $\mathbf{C}_i$ が直近の4点のみによって定まる

一階微分

$$\begin{aligned} \mathbf{C}_i'(\theta) &= 2\cos\theta\sin\theta\ \left(\mathbf{F}_{i+1}(\theta) - \mathbf{F}_i(\theta + \tfrac{\pi}{2})\right) \\ &\quad + \cos^2\theta\,\mathbf{F}_i'(\theta + \tfrac{\pi}{2}) + \sin^2\theta\,\mathbf{F}_{i+1}'(\theta)\ , \end{aligned}$$

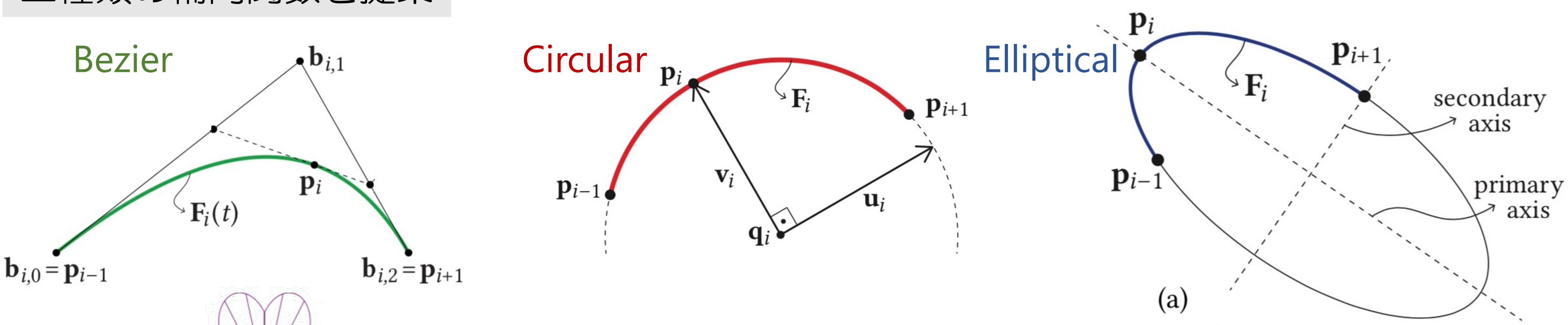
➡ $\mathbf{C}_i'(\tfrac{\pi}{2}) = \mathbf{C}_{i+1}'(0)$

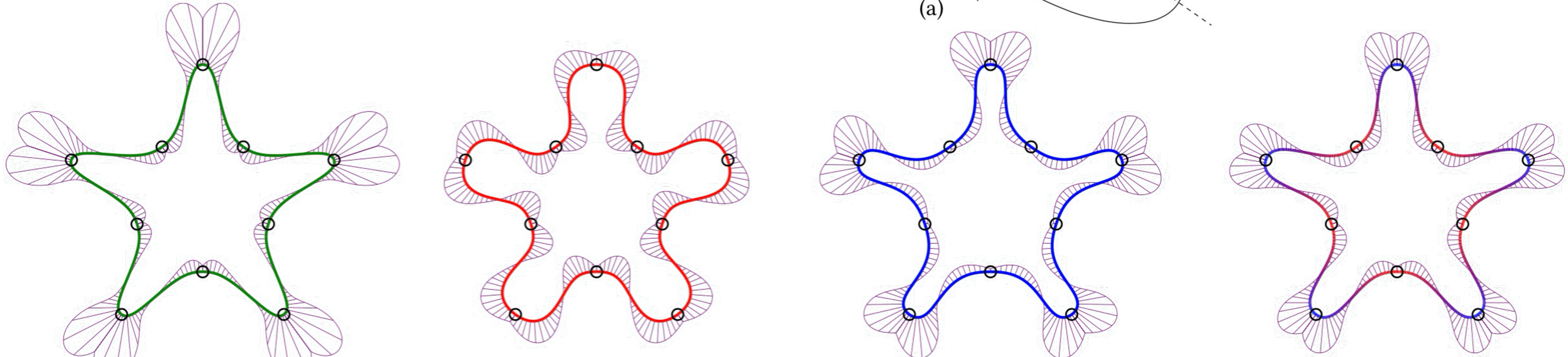
二階微分

$$\begin{aligned} \mathbf{C}_i''(\theta) &= 2\left(\cos^2\theta - \sin^2\theta\right)\ \left(\mathbf{F}_{i+1}(\theta) - \mathbf{F}_i(\theta + \tfrac{\pi}{2})\right) \\ &\quad + 4\cos\theta\sin\theta\ \left(\mathbf{F}_{i+1}'(\theta) - \mathbf{F}_i'(\theta + \tfrac{\pi}{2})\right) \\ &\quad + \cos^2\theta\,\mathbf{F}_i''(\theta + \tfrac{\pi}{2}) + \sin^2\theta\,\mathbf{F}_{i+1}''(\theta)\ . \end{aligned}$$

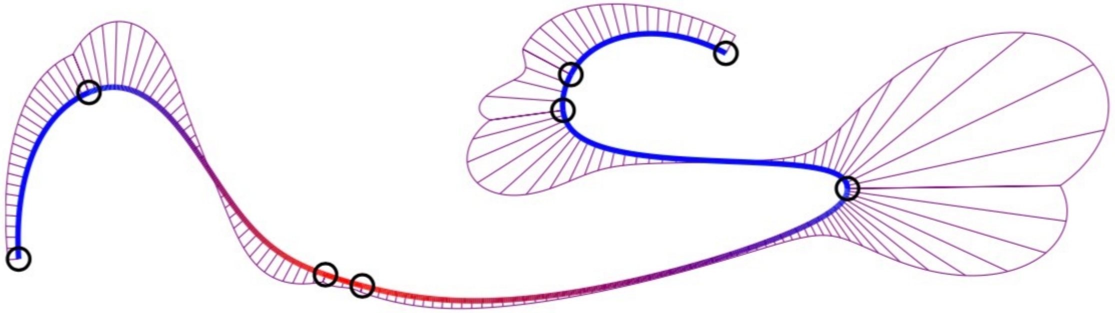
➡ $\mathbf{C}_i''(\tfrac{\pi}{2}) = \mathbf{C}_{i+1}''(0)$

三種類の補間関数を提案





(a) Bézier (b) Circular (c) Elliptical (d) Hybrid



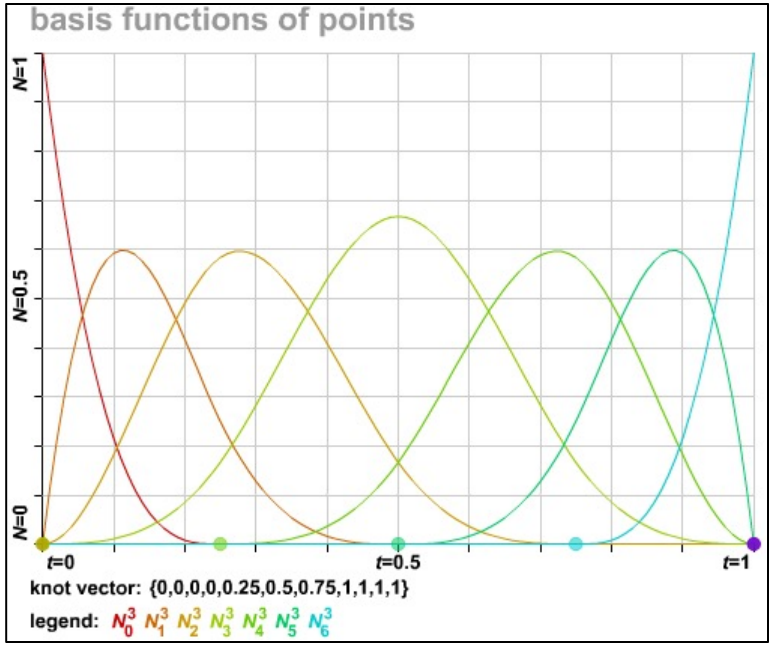
ハイブリッド型：

大きく曲がる場所 ➔ Elliptic

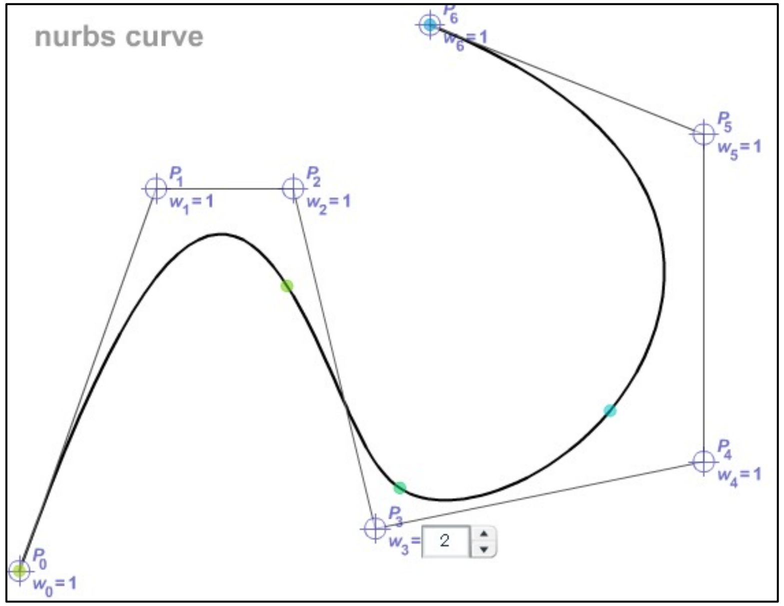
平坦な場所 ➔ Circular

Bスプライン曲線

- 多項式スプラインを表すもう一つの方法
  - 曲線を基底関数 (**b**asis) の重ね合わせで表す
  - 3次の基底関数が実用上一般的
  - サブディビジョンサーフェスと深い関係
    ➔ 次回講義で説明

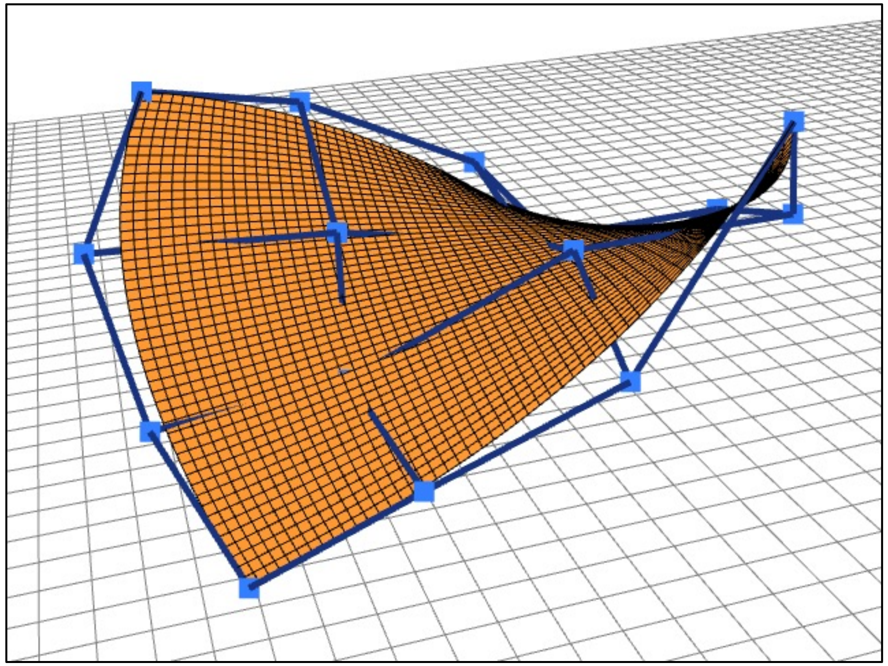


- **N**on-**U**niform **R**ational **B**-**S**pline
  - Non-Uniform = ノット列 (t_k) の間隔が一様でない
  - Rational = 制御点が重みを持つ (有理式曲線)
  - (ややこしいので本講義では説明しない)



- 秀逸なFlashデモ: http://geometrie.foretnik.net/files/NURBS-en.swf
  - 再生ツール: https://ruffle.rs/demo/

パラメトリック曲面



- パラメタが1個 ➔ 曲線 $P(t)$
- パラメタが2個 ➔ 曲面 $P(s,t)$

- 3次ベジェ曲面：
  - 入力：4×4=16個の制御点 P_{ij}

$$P(s,t) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i^3(s) b_j^3(t) P_{ij}$$

バーンスタイン基底関数

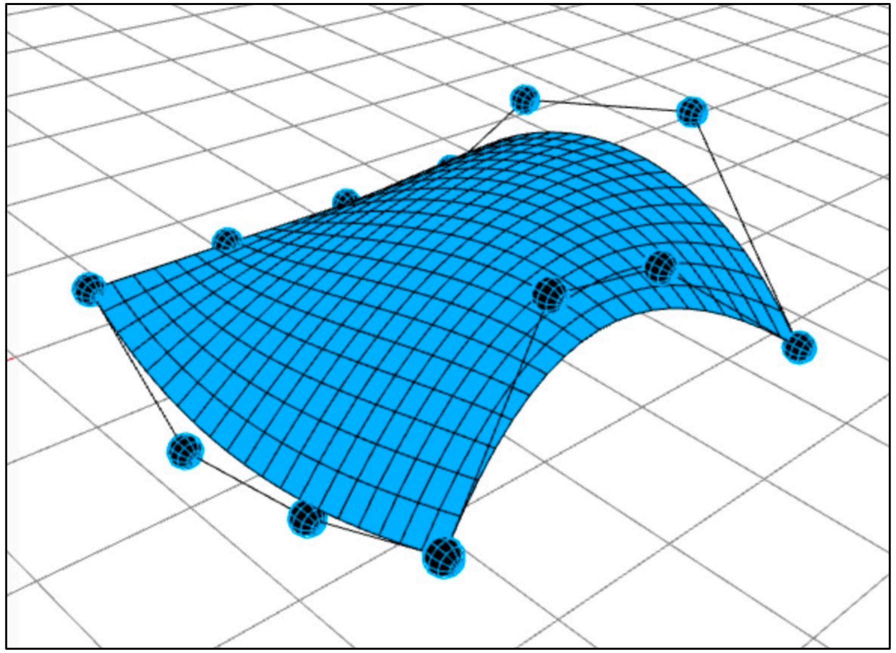
$b_0^3(t) = (1-t)^3$

$b_1^3(t) = 3t(1-t)^2$

$b_2^3(t) = 3t^2(1-t)$

$b_3^3(t) = t^3$

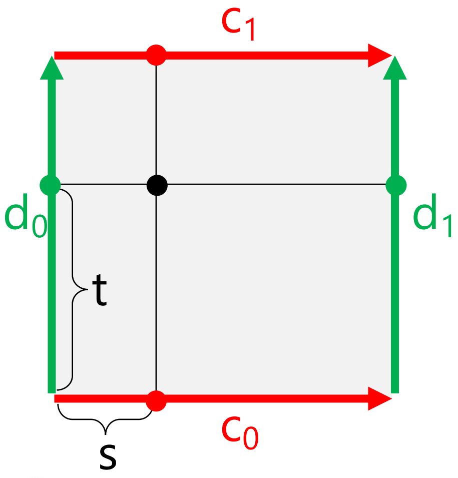
Coons 曲面



- 端点が一致する四つの曲線が定義されているとき、向かい合う曲線同士をそれぞれ評価し、その線形補間を取ることで曲面が定義できる

$$L_c(s,t) = (1-t)c_0(s) + tc_1(s)$$

$$L_d(s,t) = (1-s)d_0(t) + sd_1(t)$$



曲面の式：

$$\boxed{C(s,t) = L_c(s,t) + L_d(s,t) - B(s,t)}$$

ここで

$$B(s,t) = c_0(0)(1-s)(1-t) + c_0(1)s(1-t) + c_1(0)(1-s)t + c_1(1)st.$$

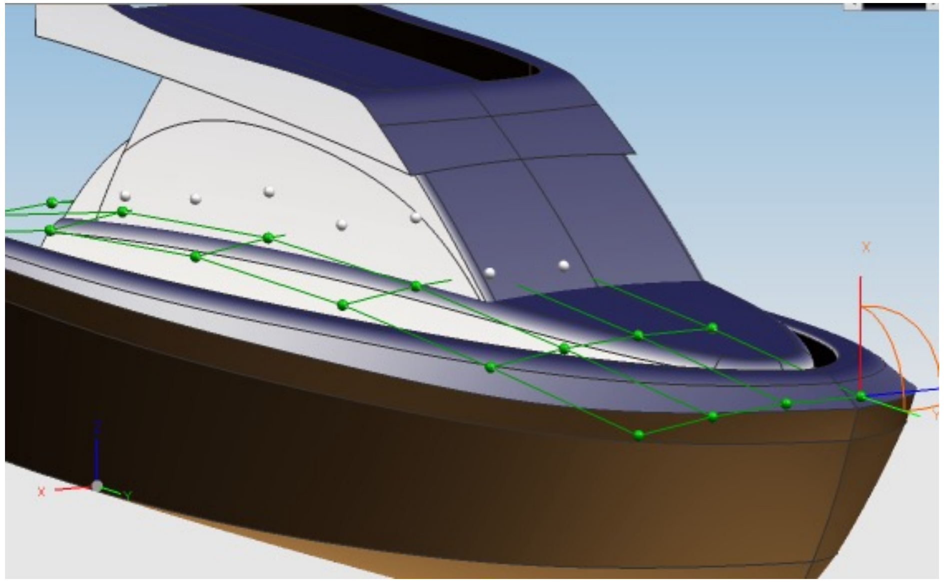
は、4つのコーナーのバイリニア補間

https://en.wikipedia.org/wiki/Coons_patch

パラメトリック曲面を用いた3Dモデリング

- 長所
  - 滑らかな曲面をコンパクトに表現できる
  - 球や円錐面などを正確に表現できる

- 短所
  - 複数のパッチをうまく配置するのが難しい
  - 複数のパッチ間の連続性を保つのが難しい



- シンプルなパーツの組み合わせで
  人工物をデザインするのによく使われる

参考

- http://en.wikipedia.org/wiki/Bezier_curve
- http://agg.sourceforge.net/antigrain.com/research/adaptive_bezier/index.html
- http://en.wikipedia.org/wiki/Cubic_Hermite_spline
- http://en.wikipedia.org/wiki/Centripetal_Catmull%E2%80%93Rom_spline
- https://qiita.com/Rijicho_nl/items/05ee4c8d77e99e29daa5
  - UX最強のベジェ曲線「κ-Curves」を完全に理解する