コンピュータグラフィックス基礎

第 5 回 曲線・曲面の表現「ベジェ曲線」

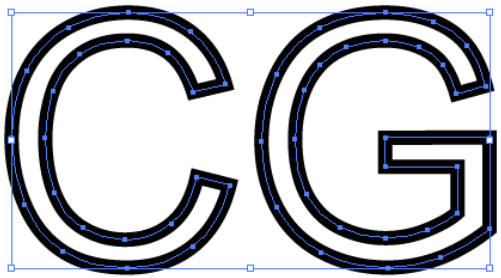
金森 由博

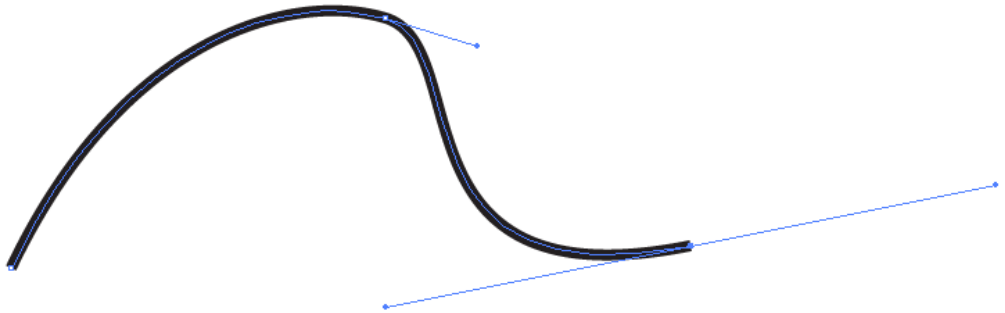
学習の目標

- 滑らかな曲線を扱う方法を学習する
- パラメトリック曲線について理解する
- 広く一般的に使われているベジェ曲線を理解する
- 制御点を入力することで、ベジェ曲線を描画するアプリケーションの開発を行えるようになる
- C++ 言語の便利な機能を使えるようになる
  - 要素数が可変な配列としての std::vector の活用

計算機による曲線の表現

- 求められるもの
  - 意図した曲線を直観的に入力できる
  - 曲線の品質がよい
    - 数学的に厳密に（任意の精度で）再現できる
    - 滑らかである（連続性、微分可能）
- 例：フォント、Illustratorなどのドローソフト

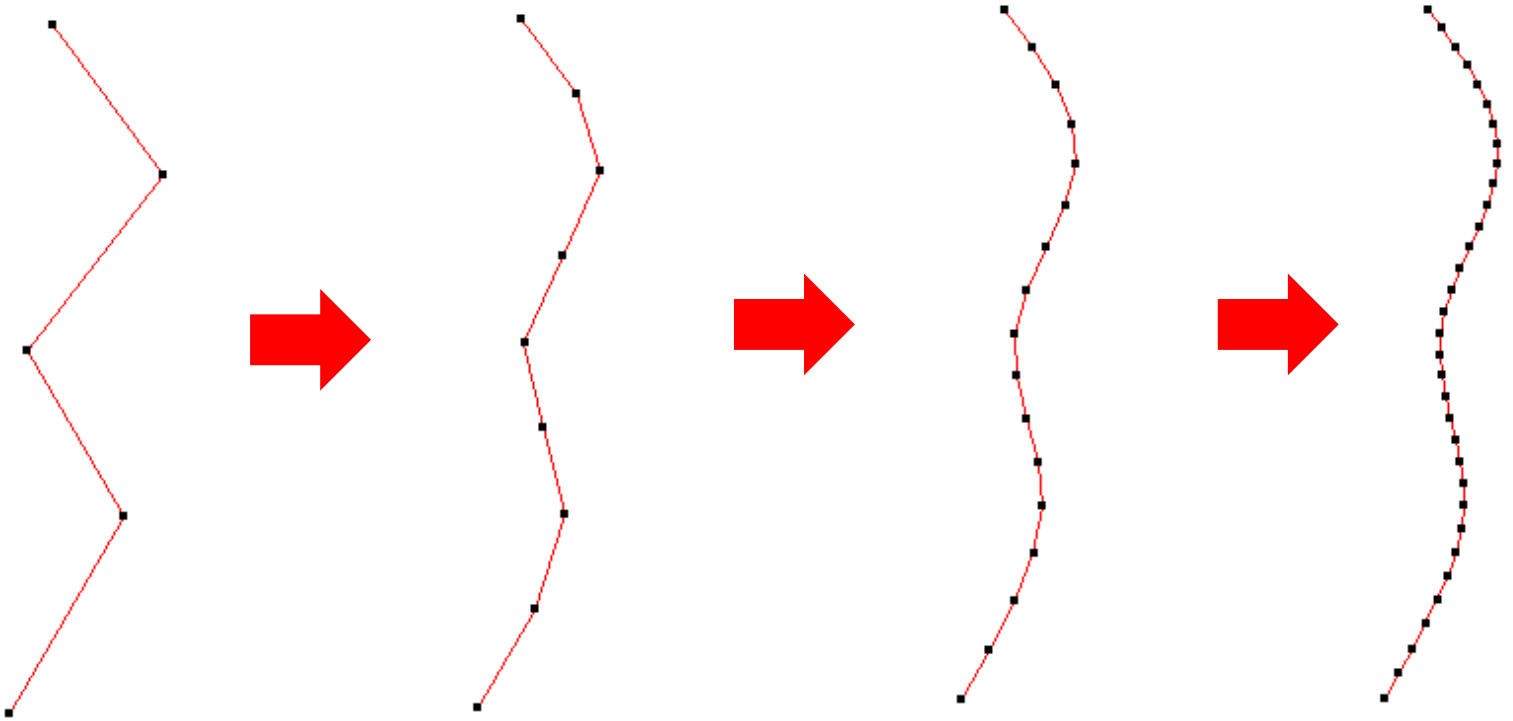




これらは、どのようのデータを持ち、どのような方法で
画面に描画されるだろうか？

折れ線による曲線の近似表現

- 折れ線（ポリライン）で近似表現する
- 細かく分割することで、曲線らしく見せる



曲線の数学的表現

- 陽関数表現
- 陰関数表現
- パラメトリック表現（媒介変数表現）

- 数式で表されるので、どこまで拡大しても滑らか
- 実際の描画は、点の集まり（折れ線近似）なので滑らかさの程度をプログラムでコントロールする

陽関数表現

$y = f(x)$ の形で表現される

例： $y = x^2$　　$y = (x-1)^3$

長所： 実装が容易

短所：表現力が乏しい

1 つの x の値に対して 1 つの y の値しか定まらない

陽関数表現の実装例

```
glBegin(GL_LINE_STRIP);
for(double x = 0; x < 1.0; x += 0.01) {
  glVertex2d(x, f(x));
}
glEnd();
```

+You ウェブ 画像 動画 地図 ニュース ショッピング Gmail もっと見る

Google y=x^2

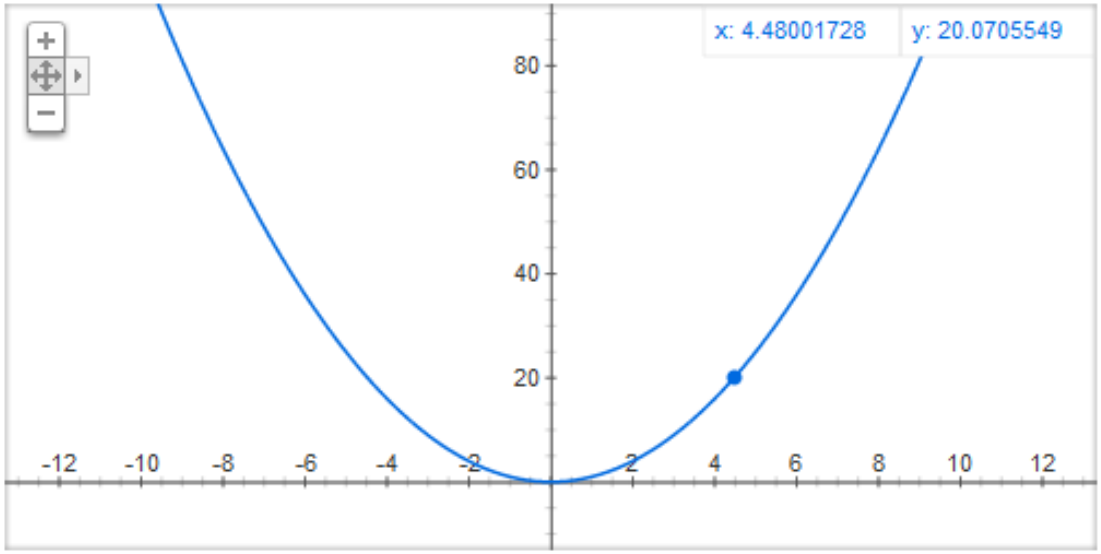
検索 約 1,340,000 件（0.22 秒）

すべて
画像
地図
動画
ニュース
ショッピング
掲示板
もっと見る

ウェブ全体から検索
日本語のページを検索
翻訳して検索

日本語のページを検索

Graph for x^2



陰関数表現

$f(x, y) = 0$ の形で表現される

- xとyの値が陽に求まらない。
- 数式で空間を＋領域と-領域に区分し、その境界を表現したもの。

例：$x^2 + y^2 - 1 = 0$

陽関数を陰関数で表現することもできる
例：$y = x^2 \quad \rightarrow \quad x^2 - y = 0$

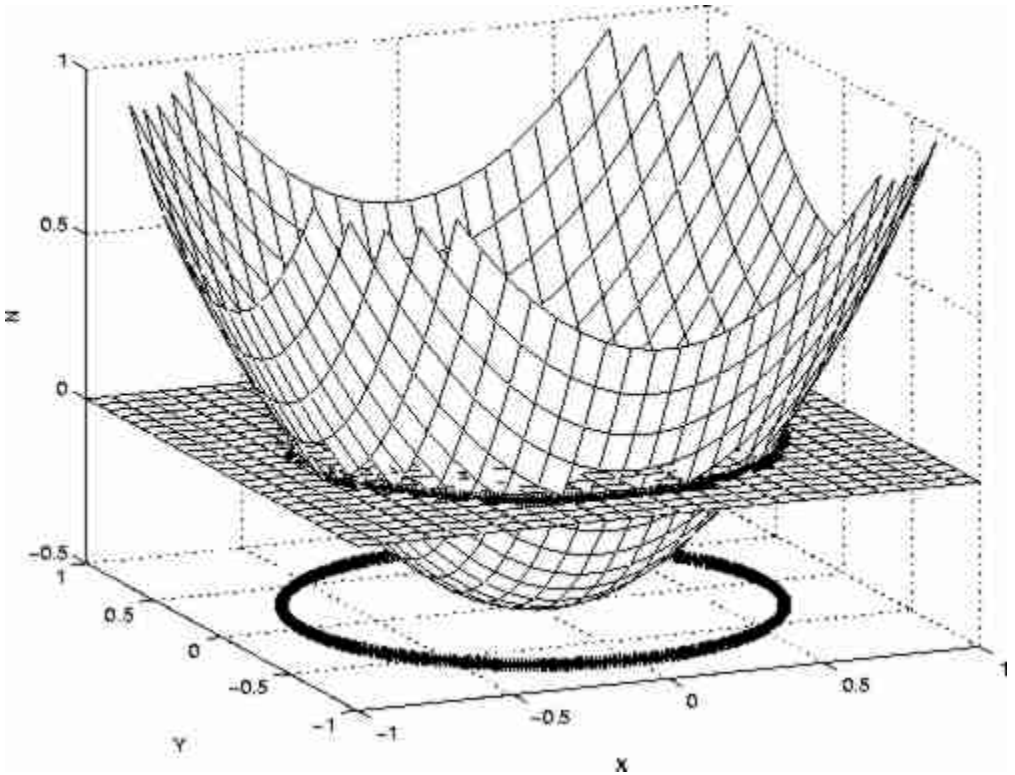
長所：複雑な曲線を表現できる
短所：方程式を解かなくてはならない
（次数が高くなると解を求めるのが困難）

陰関数表現されたグラフの描画

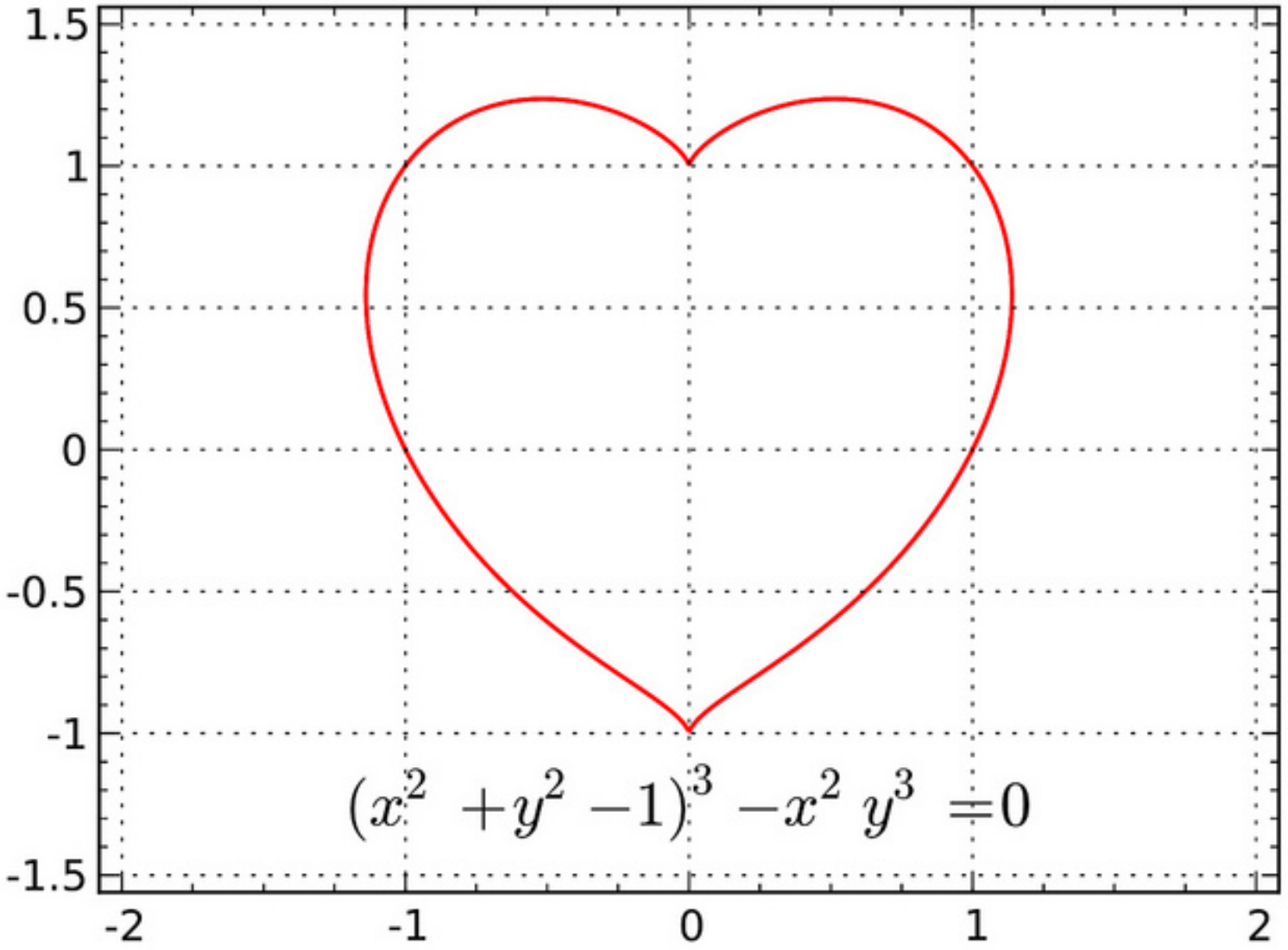
例えば　$x^2+4y^2-1=0$　の場合

次元を１つ上げて $z=x^2+4y^2-1$ とする

$z>0, z<0$ の領域の境界が求めるグラフとなる



http://www.lems.brown.edu/~tt/2Dfitting/ipcurves.html



http://www.flickr.com/photos/dullhunk/6233240407/

パラメトリック表現

$$\begin{cases} x = f_x(t) \\ y = f_y(t) \end{cases}$$ の形で表される関数

個々の座標値がパラメータ（媒介変数）で表現される。
パラメータ t の値が与えられれば、x, y 座標が求まる。

例： $$\begin{cases} x = r\cos(t) \\ y = r\sin(t) \end{cases} \quad (0 \le t \le 2\pi)$$

長所：実装が容易
　　t の値の刻み幅で曲線の正確さを制御できる

パラメトリック表現の実装例

一般に、t の値は 0 から 1 とすることが多い。

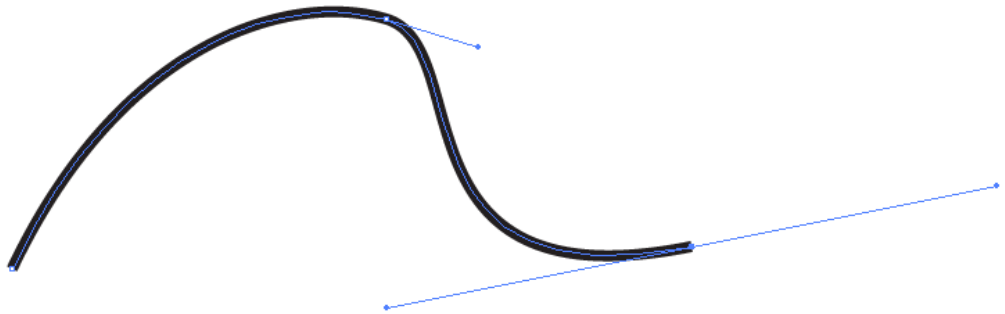
$$\begin{cases} x = r\cos(t) \\ y = r\sin(t) \end{cases} \quad (0 \le t \le 2\pi) \Rightarrow \begin{cases} x = r\cos(2\pi t) \\ y = r\sin(2\pi t) \end{cases} \quad (0 \le t \le 1)$$

```
glBegin(GL_LINE_STRIP);
for(double t = 0; t <= 1.0; t += 0.01) {
  glVertex2d(fx(t), fy(t));
}
glEnd();
```

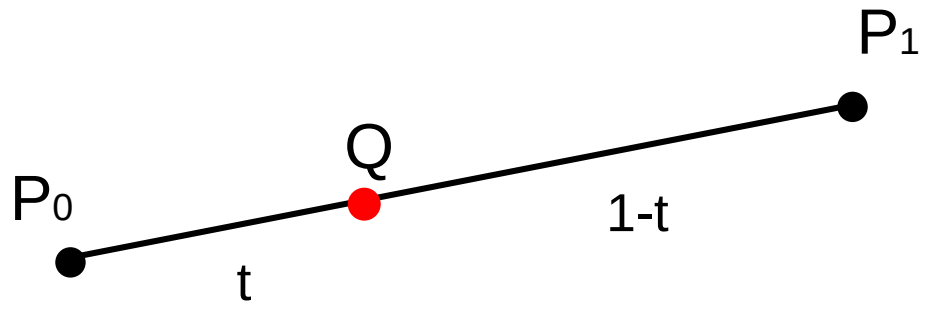
パラメトリック曲線

- 広く使われているパラメトリック表現による曲線
  - ベジェ曲線（今週の内容）
  - Bスプライン曲線（来週の内容）
  - 「制御点」という概念を用いて形状を容易にコントロール（制御）できる

ベジェ曲線



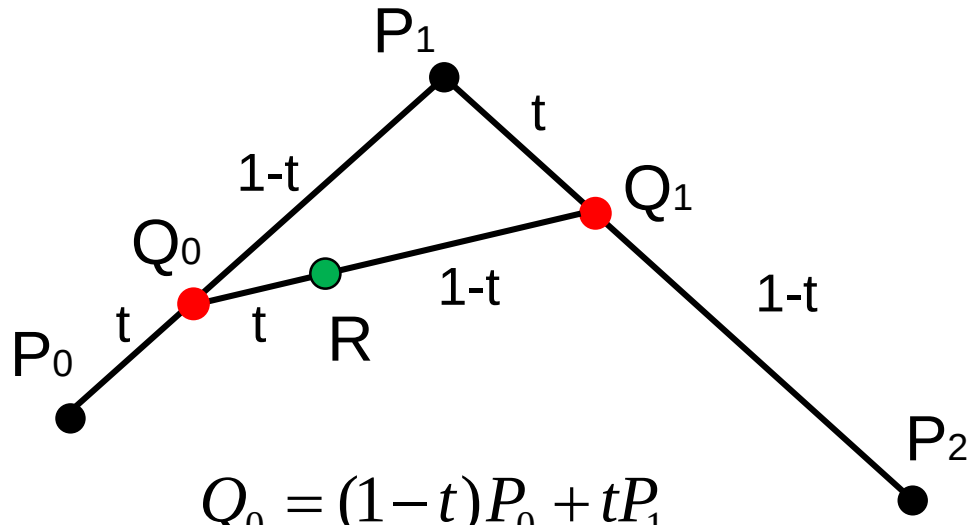
ベジェ曲線の図形的理解（1 次）



$$Q = (1-t)P_0 + tP_1 \quad (0 \le t \le 1)$$

- t の 1 次式
- 2 つの制御点
- 直線

ベジェ曲線の図形的理解（2 次）



$$Q_0 = (1-t)P_0 + tP_1$$

$$Q_1 = (1-t)P_1 + tP_2 \qquad (0 \le t \le 1)$$

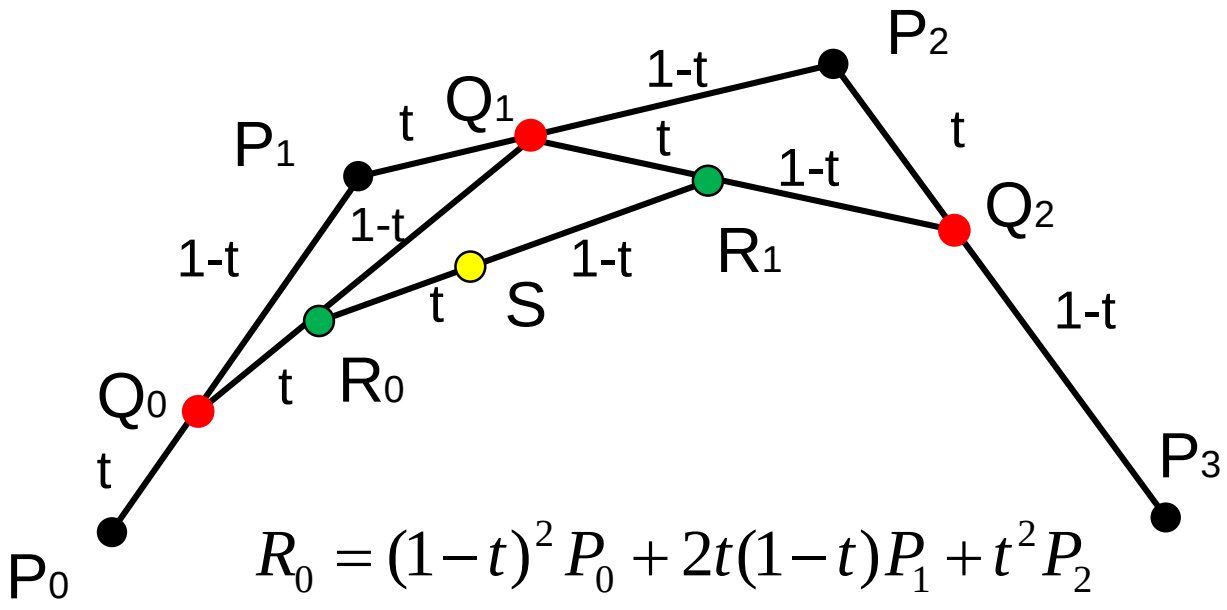
$$R = (1-t)Q_0 + tQ_1$$

$$R = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$

- t の 2 次式
- 3 つの制御点
- 2 次曲線

（問 t = 0, 0.5, 1 のときの位置は？）

ベジェ曲線の図形的理解（3 次）



$$R_0 = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$

$$R_1 = (1-t)^2 P_1 + 2t(1-t)P_2 + t^2 P_3$$

$$S = (1-t)R_0 + tR_1$$

$$(0 \le t \le 1)$$

$$S = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

- t の 3 次式
- 4 つの制御点
- 3 次曲線

ベジェ曲線の数式表現（まとめ）

- 1 次　$P(t) = (1-t)P_0 + tP_1$

- 2 次　$P(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$

- 3 次　$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$

一般的な CAD 系ソフトウェアで用いられている（Adobe Illustrator も）

3 次ベジェ曲線の性質

$$S(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

$$S(0) =$$

$$S(1) =$$

$$\frac{dS(t)}{dt} =$$

$$\frac{dS(0)}{dt} =$$

$$\frac{dS(1)}{dt} =$$

問： 上記の値を求めよ。また赤線で示した各制御点の係数の和を求めよ。

3 次ベジェ曲線の性質

$$S(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

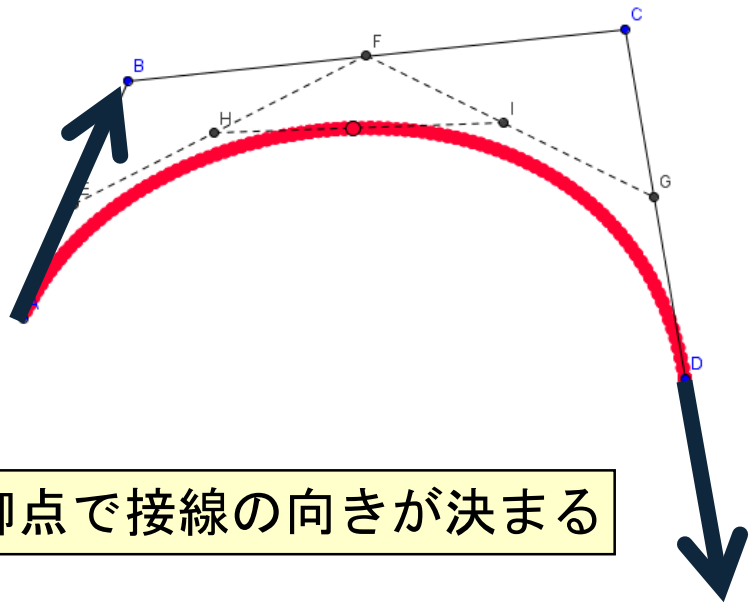
$S(0) = P_0 \quad S(1) = P_3$ ← 最初と最後の制御点を通る

$$\frac{dS(t)}{dt} = -3(1-t)^2 P_0 + (9t^2 - 12t + 3)P_1 + (-9t^2 + 6t)P_2 + 3t^2 P_3$$

$$\frac{d^2 S(t)}{dt^2} = 6(1-t)P_0 + (18t - 12)P_1 + (-18t + 6)P_2 + 6tP_3$$

$$\frac{dS(0)}{dt} = -3P_0 + 3P_1 = 3\overrightarrow{P_0 P_1}$$

$$\frac{dS(1)}{dt} = -3P_2 + 3P_3 = 3\overrightarrow{P_2 P_3}$$

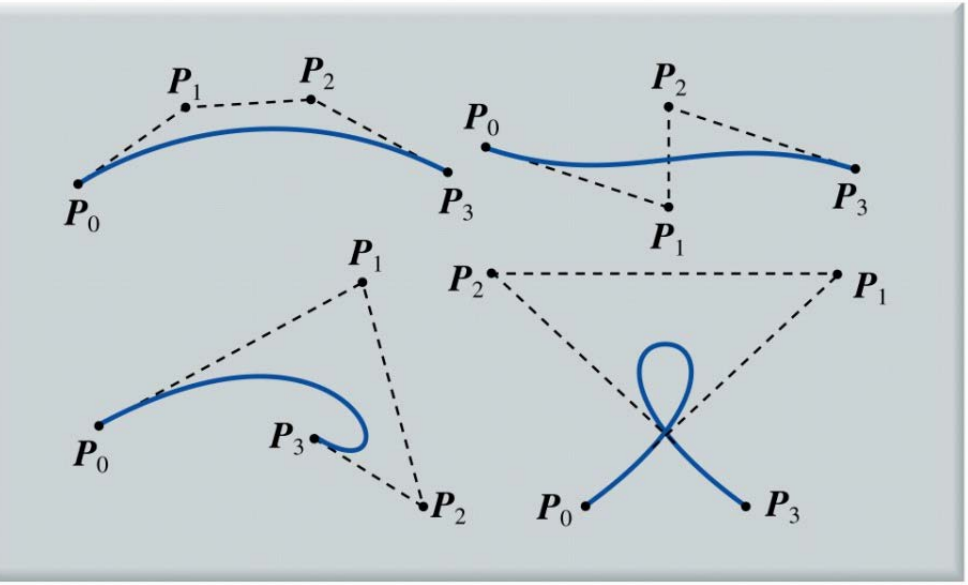


第 1, 2 制御点と第 3, 4 制御点で接線の向きが決まる

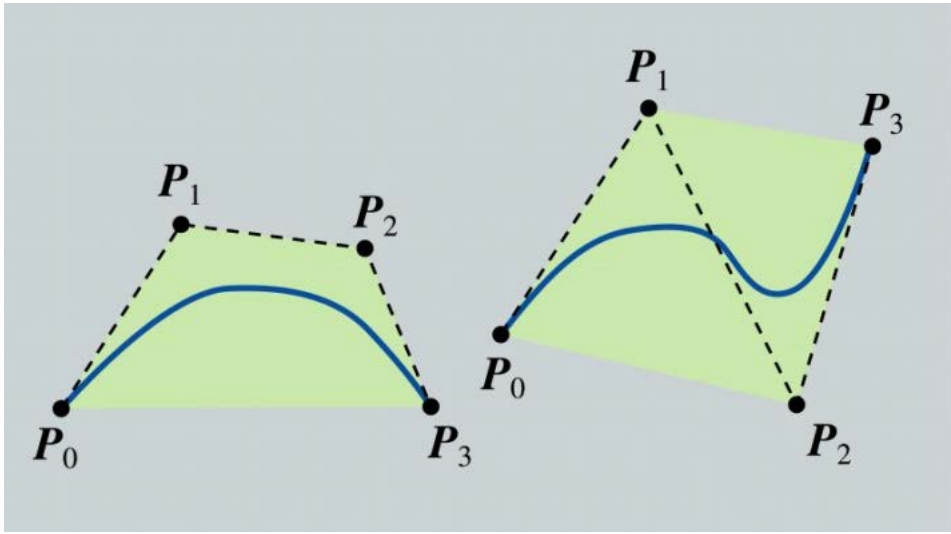
3 次ベジェ曲線の凸包性

制御点の凸包の内部に含まれる

■図3.23——3次ベジエ曲線の例

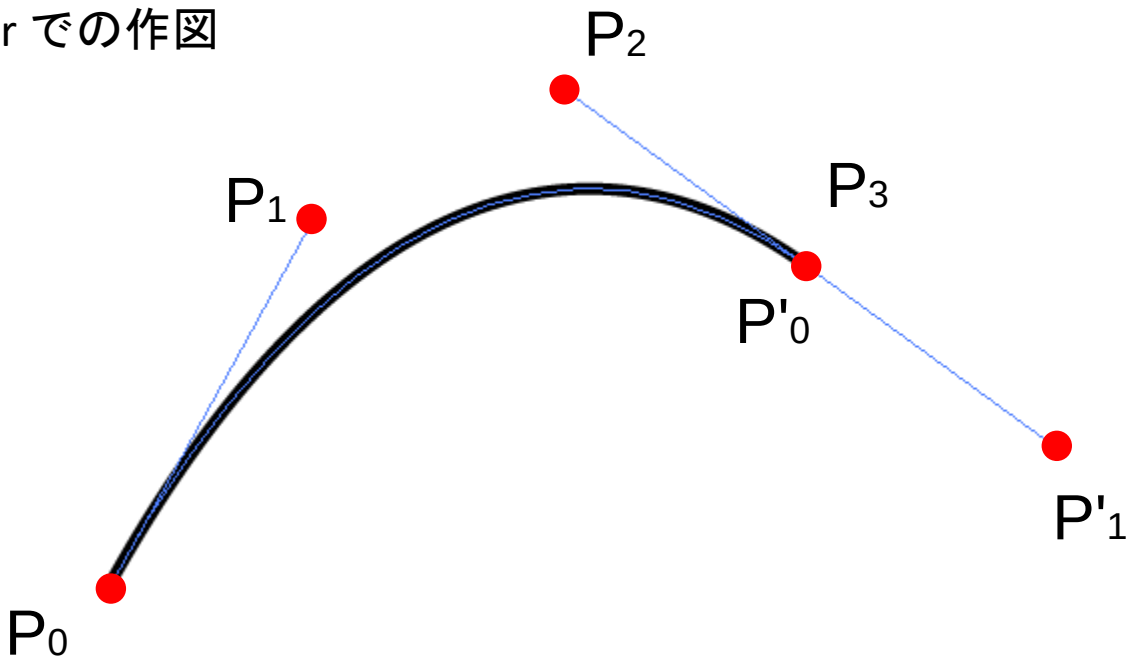


「コンピュータグラフィックス」2004年 / 財団法人画像情報教育振興協会（CG-ARTS協会）



複数セグメントの連結

Illustrator での作図



- 2 つのセグメントで端点を共有する
  （制御点位置の共有, C^0連続）
- 2 つのセグメントで接線を共有する
  （制御点が同一直線上で等距離，C^1連続）

1 階微分が接続点で同値

N 次ベジェ曲線（ベジェ曲線の一般化）

$$P(t)=\sum_{i=0}^{n} {}_nC_i\, t^i (1-t)^{n-i} P_i$$

$${}_nC_i = \frac{n!}{i!(n-i)!}$$

2項係数

さらなる一般化

$$P(t)=\sum_{i=0}^{n} B_i^n P_i$$

$$B_i^n = {}_nC_i\, t^i (1-t)^{n-i}$$

$$\binom{n}{i}$$ と表記することもある

ある**比率**で各制御点の座標を混ぜ合わせる！

混合比（和は 1 になる）

混合比を関数で表したものを**「基底関数」**とよぶ

ベジェ曲線の基底関数

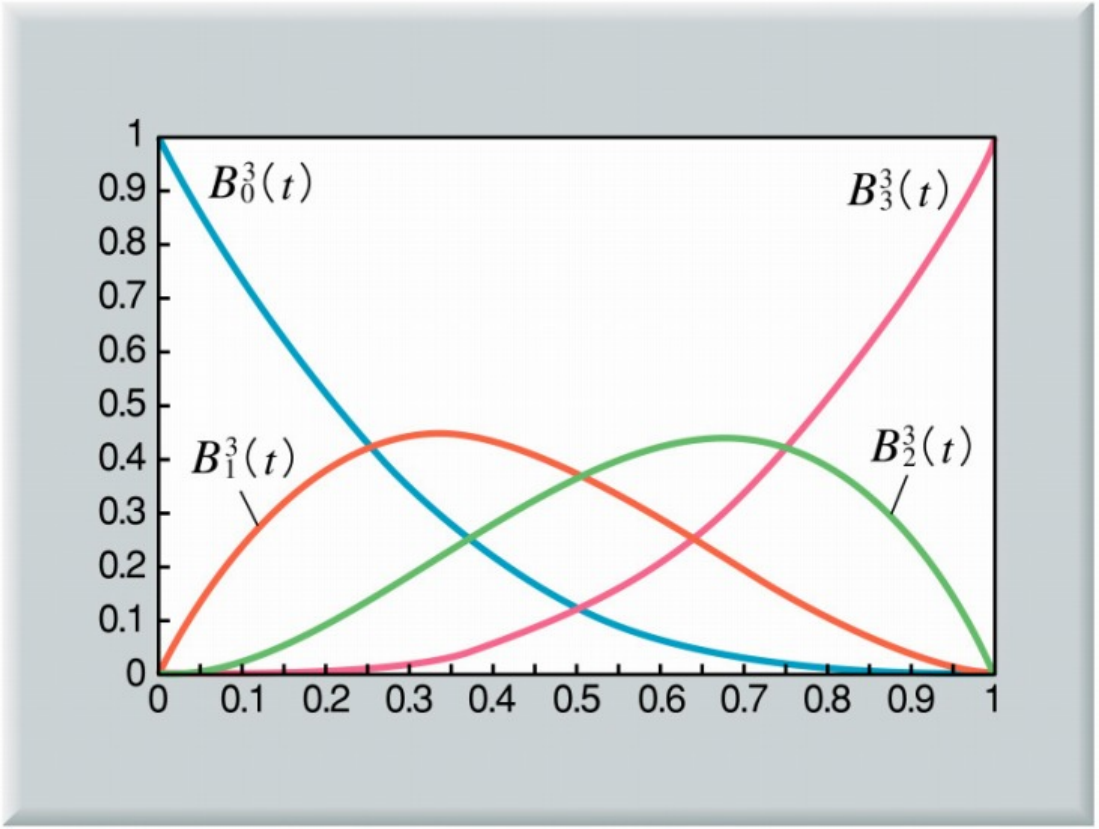
- バーンスタイン基底関数

$$B_i^n = {}_n C_i \, t^i (1-t)^{n-i}$$ n は次数を表す

3 次の場合

$$B_0^3 = (1-t)^3$$
$$B_1^3 = 3t(1-t)^2$$
$$B_2^3 = 3t^2(1-t)$$
$$B_3^3 = t^3$$

■図3.24——3次バーンスタイン基底関数のグラフ

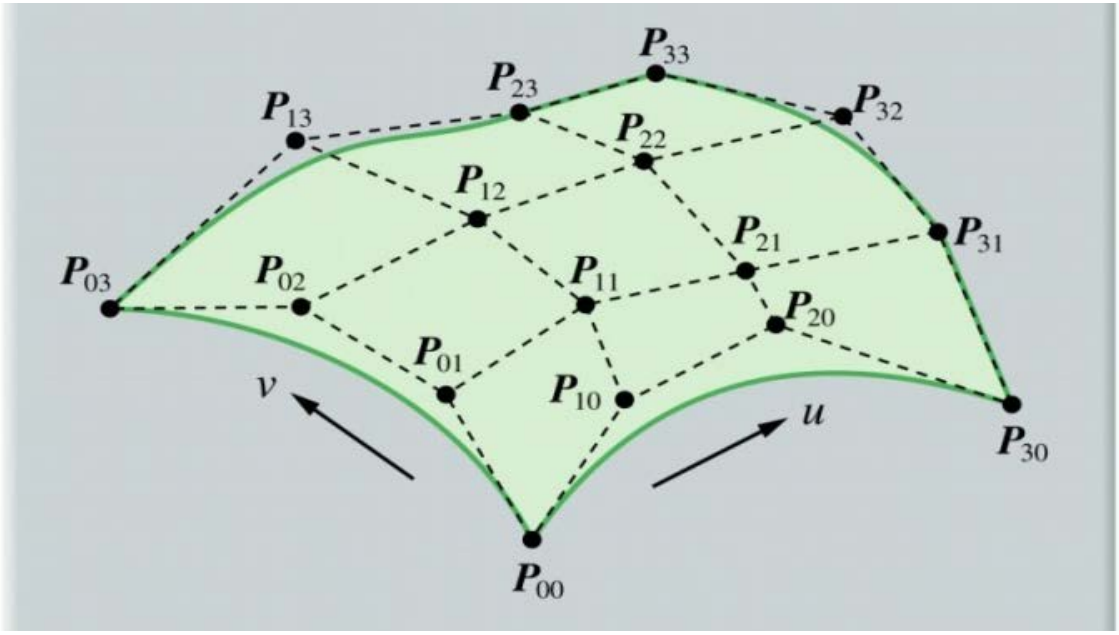


「コンピュータグラフィックス」2004年 / 財団法人画像情報教育振興協会（CG-ARTS協会）

3 次ベジェ曲面： 双 3 次ベジェ曲面

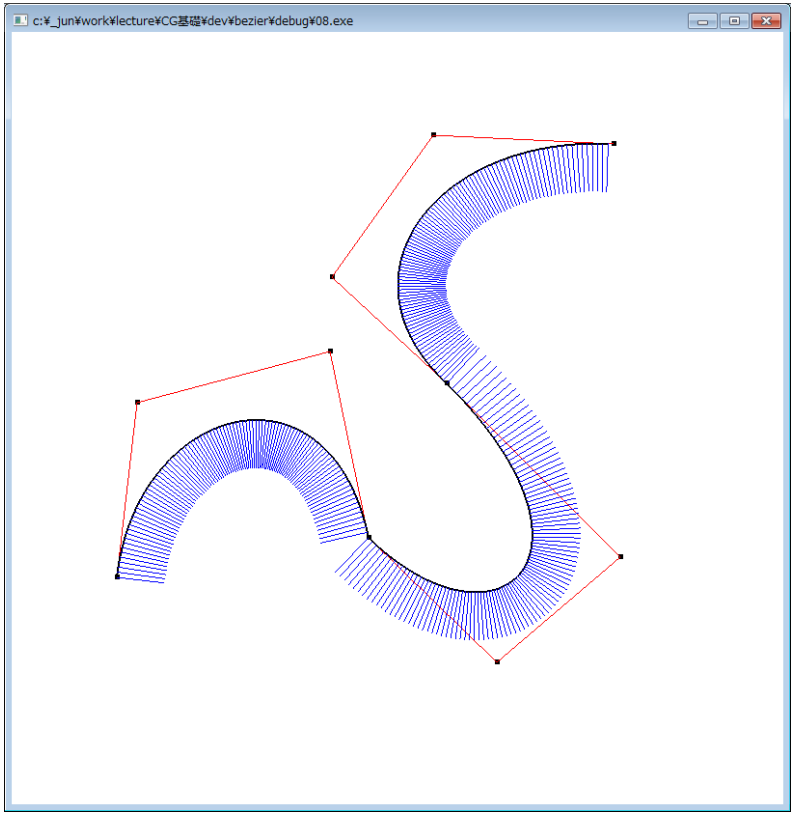
- 4×4 の格子状に並んだ 16 個の制御点 P_{ij} と 2 つのパラメータ u, v によって定義される。
- 4 隅の位置は制御点と一致する。

$$S(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} P_{ij} B_i^3(u) B_j^3(v)$$



課題

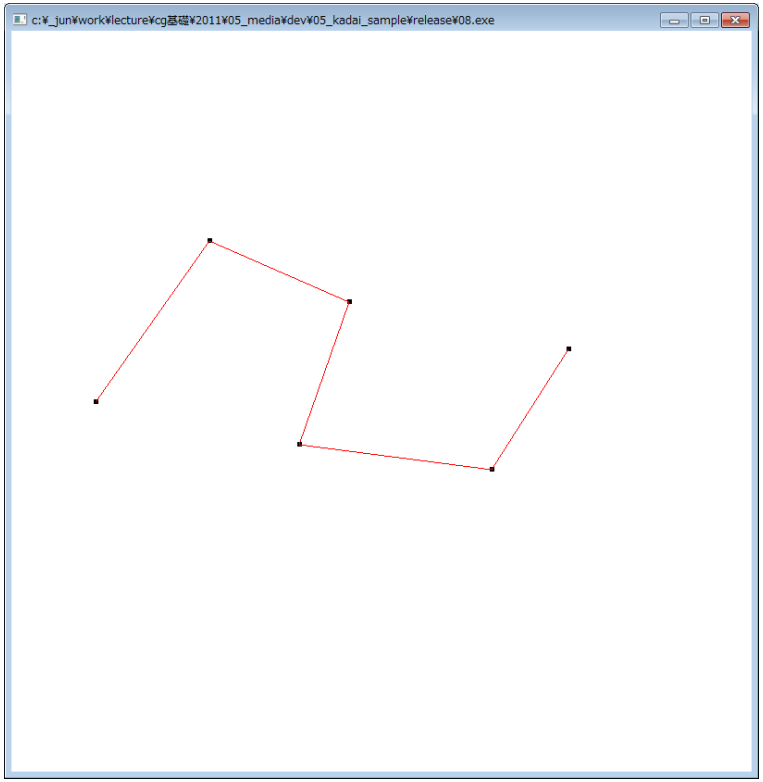
- 入力された制御点を使って、3 次の Bezier 曲線を描画する
- 法線（接線に垂直な線）を描画する



解答例デモ

サンプルコード

- マウスの左クリックで制御点を追加、右クリックで削減する
- 制御点を連結した折れ線を表示する



C++ 言語で使用できる STL

- STL の vector クラスが便利
  - 配列の代わりに使える
  - 最初にサイズを指定する必要がない
  - 要素の追加と削除が簡単

[C言語]

```
#define MAX_ELEMENT_NUM 100
int numPoint = 0; // 要素の数を記録
double x[MAX_ELEMENT_NUM ];
double y[MAX_ELEMENT_NUM ];
x[0] = 1.0;
y[0] = 2.0;
x[1] = 3.0;
y[1] = 4.0;
numPoint = 2;
```

[C++]

```
#include <vector>
std::vector<Vector2d> points;
points.push_back(Vector2d(1.0, 2.0));
points.push_back(Vector2d(3.0, 4.0));

// C++11 なら
// points.emplace_back(1.0, 2.0);
// points.emplace_back(3.0, 4.0);
```

STL の vector の使用例

```
#include <vector>

std::vector<Vector2d> points;

// 追加
points.push_back(Vector2d(1.0, 2.0));
points.push_back(Vector2d(3.0, 4.0));
points.push_back(Vector2d(5.0, 6.0));

// 末尾の削除
points.pop_back();

// 要素数の確認
unsigned int n = points.size();

// 要素の取得
Vector2d v0 = points[0];
Vector2d v1 = points[1];

// 要素の全削除
points.clear();
```