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」

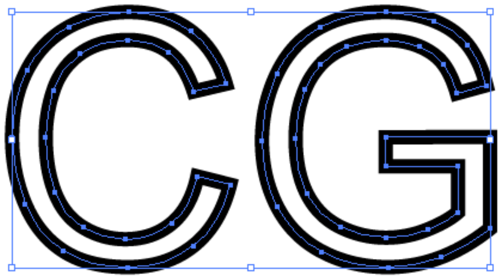
三谷 純

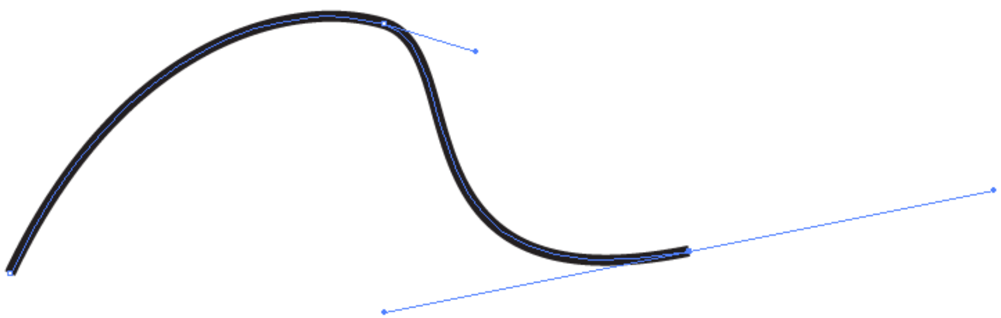
学習の目標

- 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

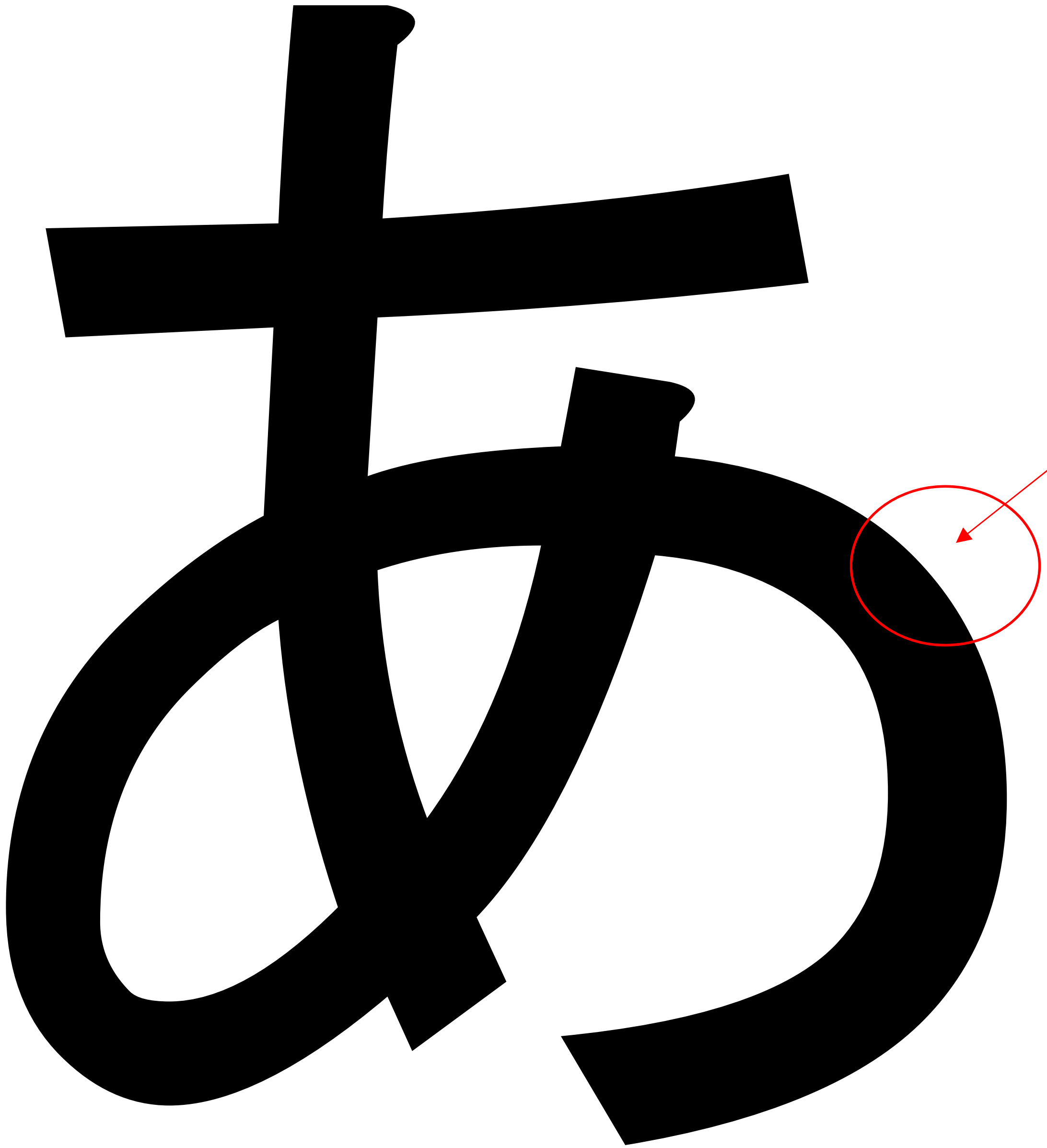
計算機による曲線の表現

- 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、どのようにデータを持ち、どのような方法で
画面に描画されるだろうか？


数学的に定義され
る形なので、どこま
で拡大しても滑らか

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

・実際に描画するときは、曲線上の点をサンプリングし、折れ線で近似

・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：　$y = x^2$　　$y = (x-1)^3$

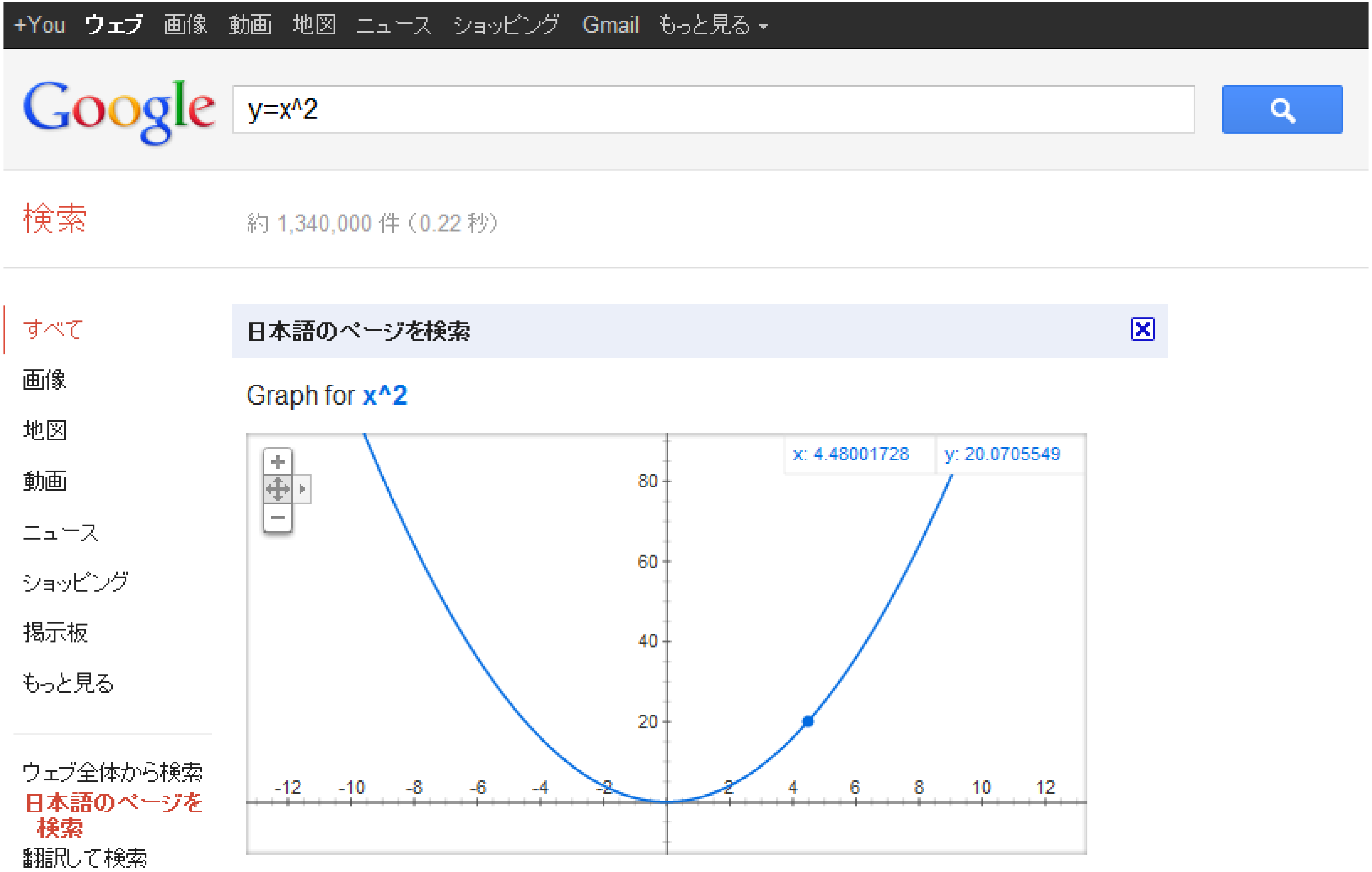
長所：実装が容易

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
x: 4.48001728
y: 20.0705549
80
60
40
20
-12
-10
-8
-6
-4
-2
2
4
6
8
10
12

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： $y = x^2 \quad \rightarrow \quad x^2 - y = 0$

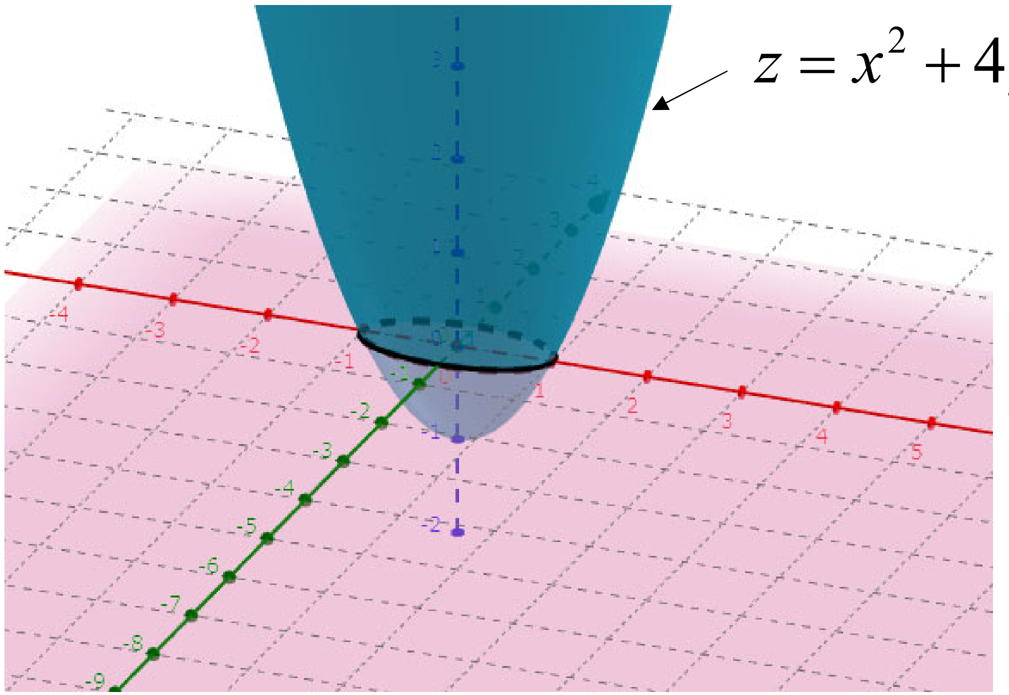
長所：複雑な曲線を表現できる
短所：方程式を解かなくてはならない
（または空間を分割して境界を探索する）
次数が高くなると解を求めるのが困難

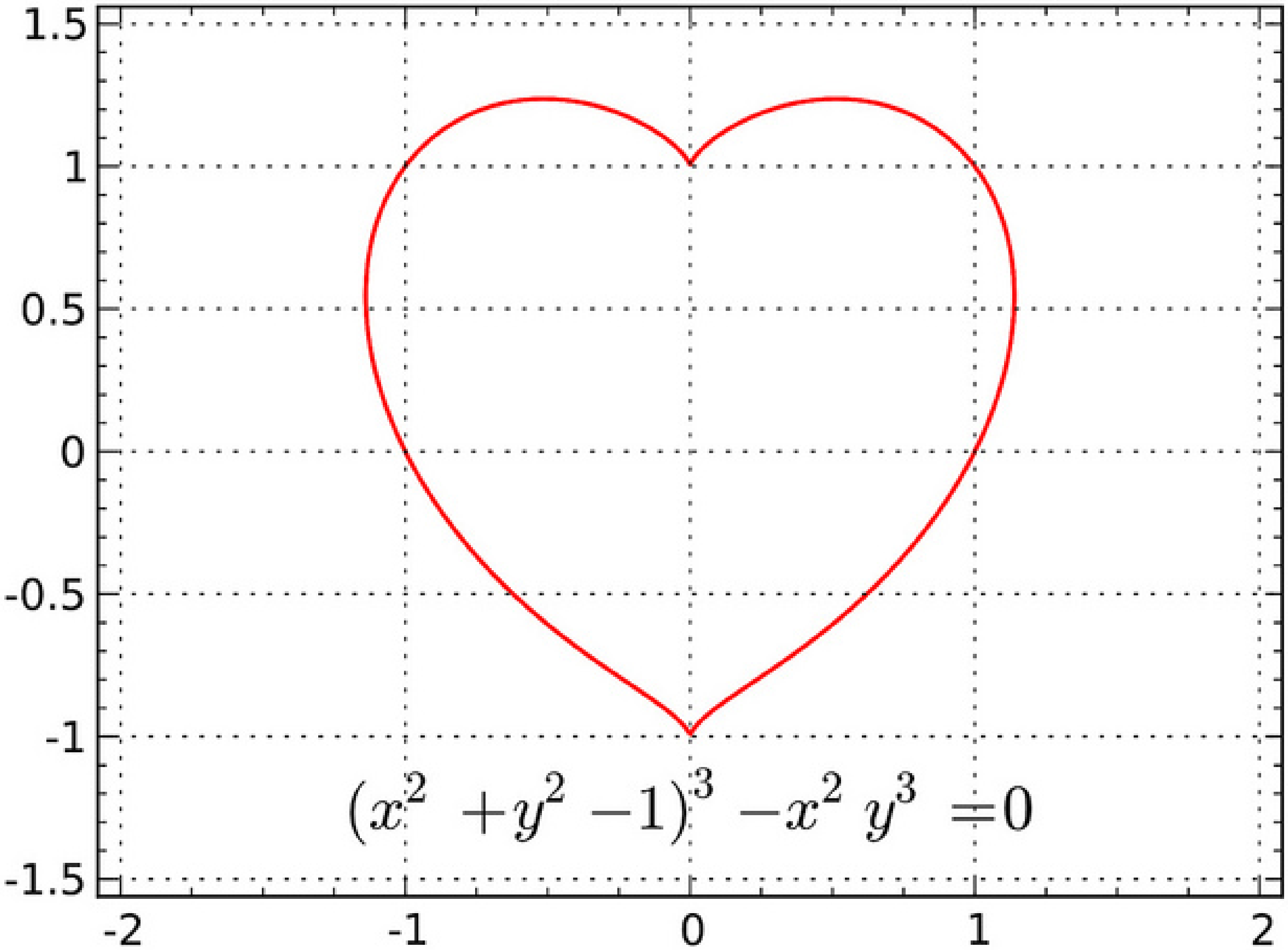
陰関数表現されたグラフの描画

例えば　$x^2+4y^2-1=0$　の場合

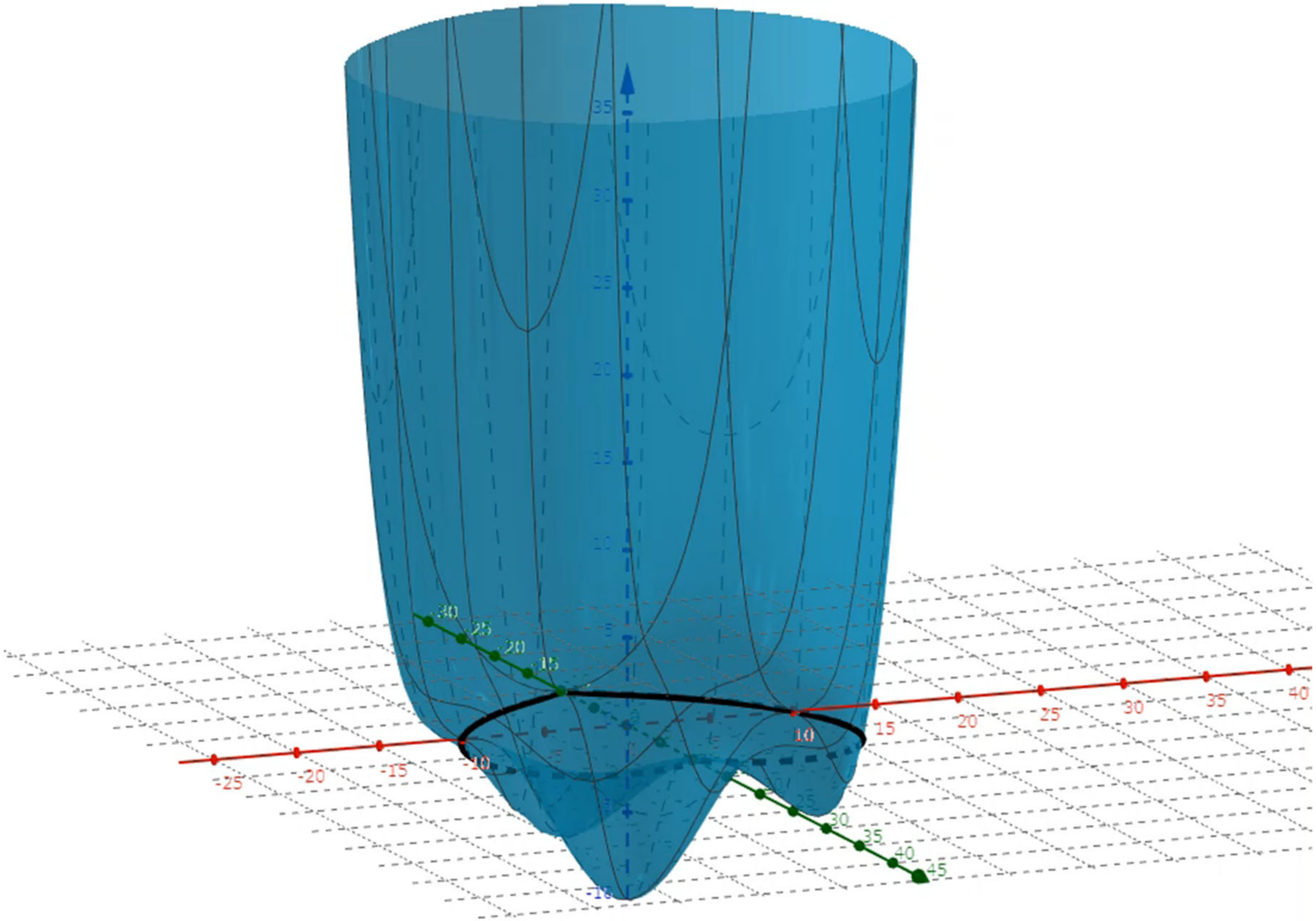
次元を１つ上げて　$z=x^2+4y^2-1$　とする

$z=0$となる点の集合が求めるグラフとなる





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


35
30
25
20
15
10
5
-10
-25
-20
-15
-10
10
15
20
25
30
35
40
30
25
20
15
30
35
40
45

陰関数表現の利点

- 内側と外側を定義できる

$f(x, y) < 0$ 内側

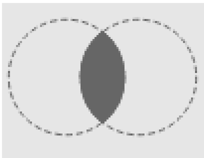
$f(x, y) = 0$ 境界

$f(x, y) > 0$ 外側

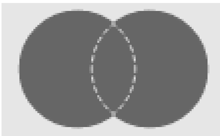
図形の面積の計算、塗りつぶし処理が必ずできる。

陰関数表現の利点

- 図形の論理演算が容易



図形の積　$A \cap B$　　$\max(F_A, F_B) \leq 0$



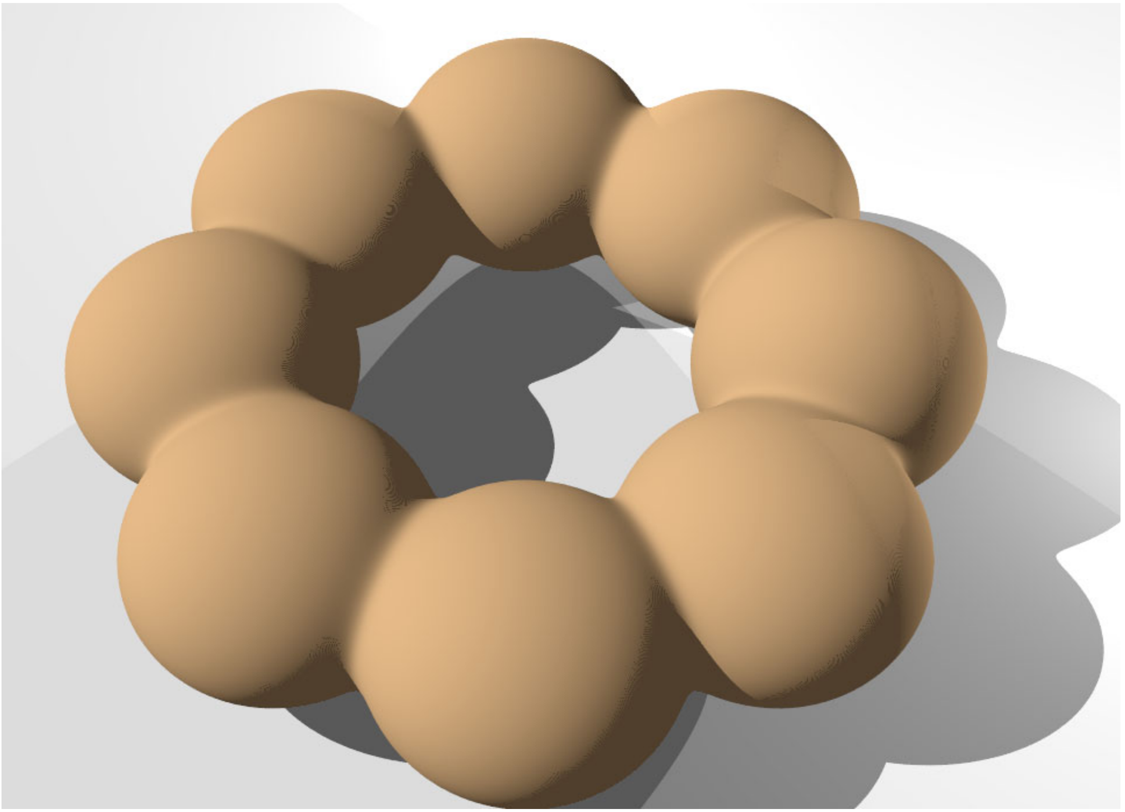
図形の和　$A \cup B$　　$\min(F_A, F_B) \leq 0$

- 立体への拡張が容易

$$f(x, y) = 0 \quad \rightarrow \quad f(x, y, z) = 0$$

陰関数表現の例（3次元形状）

$$\sum_{k=0}^{7} \exp\left\{-0.7\left(\left(x-6.75\cos\frac{2\pi}{k}\right)^2+\left(y-6.75\sin\frac{2\pi}{k}\right)^2+1.2z^2\right)\right\}-0.001=0$$



ポン・デ・リング

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

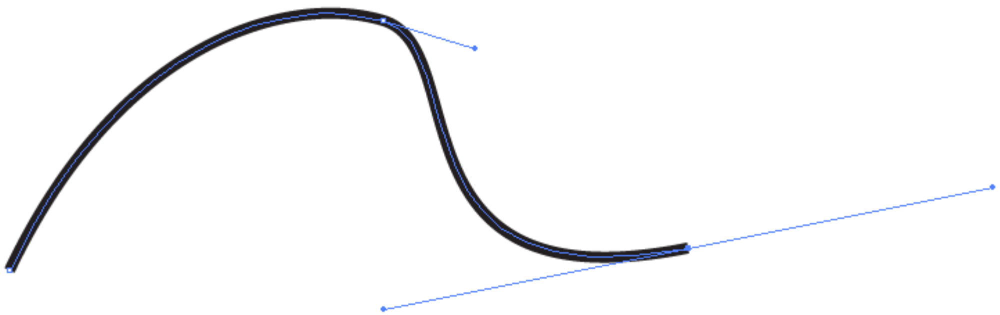
- 広く使われているパラメトリック表現による曲線

いちいち数式を記述していられない！
数か所マウスでクリックするだけで曲線を描きたい！

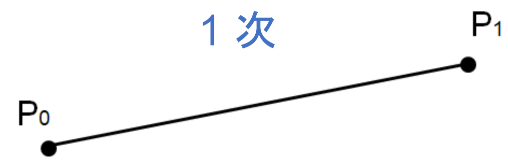
- ベジェ曲線（今週の内容）
- B スプライン曲線（来週の内容）

「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

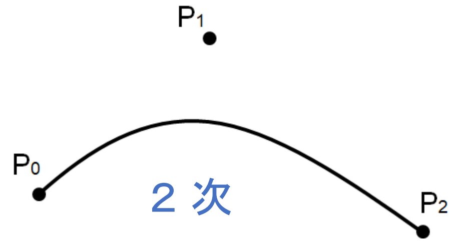


ベジェ曲線の数式表現



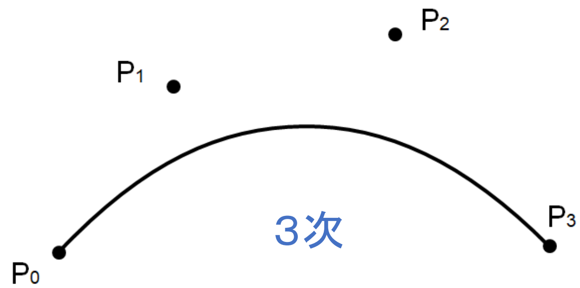
- 1 次

$$\mathbf{P}(t) = (1-t)\mathbf{P}_0 + t\mathbf{P}_1$$



- 2 次

$$\mathbf{P}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2$$



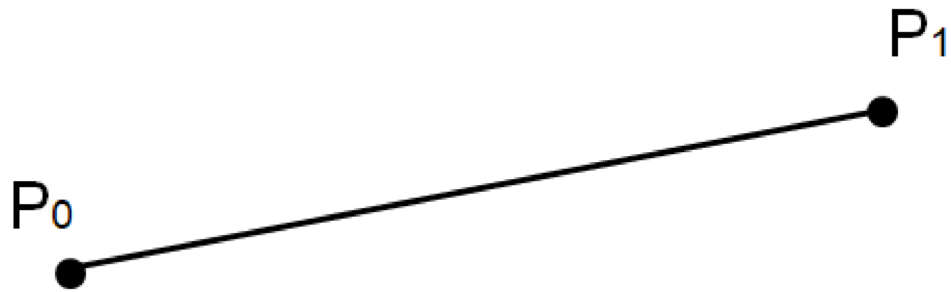
- 3 次

$$\mathbf{P}(t) = (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3$$

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

１次ベジェ曲線　（＝直線）

$$\mathbf{P}(t) = (1-t)\mathbf{P}_0 + t\mathbf{P}_1$$



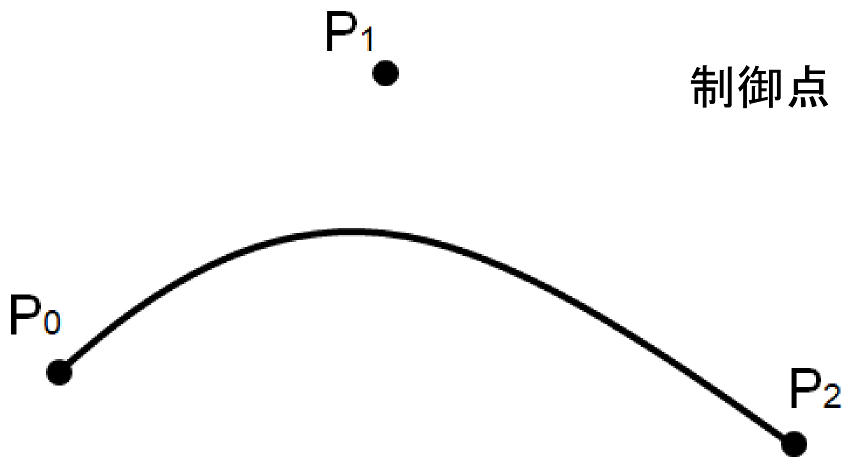
制御点 2つ

２次ベジェ曲線

$$\mathbf{P}(t) = (1-t)^2\mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2\mathbf{P}_2$$

ある比率で各制御点の座標を混ぜ合わせている！

（混合比）

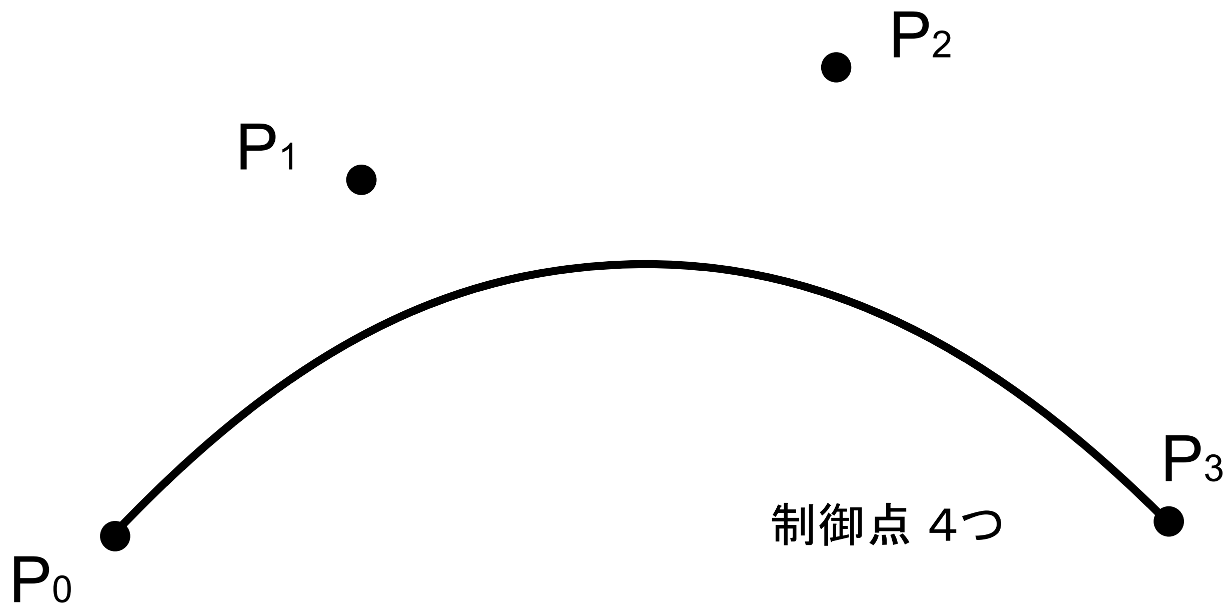


制御点 3つ

３次ベジェ曲線

$$\mathbf{P}(t) = (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3$$

ある比率で各制御点の座標を混ぜ合わせている！
（混合比）



3 次ベジェ曲線の性質

$$\mathbf{P}(t) = (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3$$

$$\mathbf{P}(0) = \mathbf{P}_0 \quad \mathbf{P}(1) = \mathbf{P}_3$$

最初と最後の制御点を通る

1階微分 → 接線ベクトル

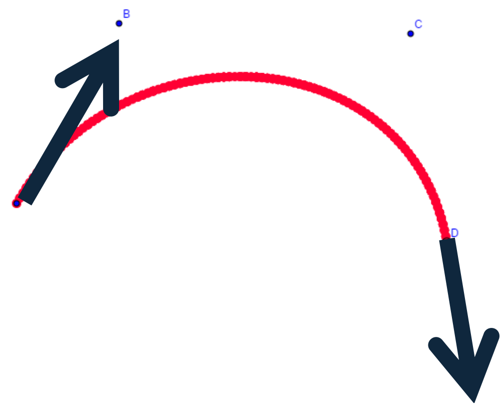
$$\frac{d\mathbf{P}(t)}{dt} = -3(1-t)^2\mathbf{P}_0 + (9t^2 - 12t + 3)\mathbf{P}_1 + (-9t^2 + 6t)\mathbf{P}_2 + 3t^2\mathbf{P}_3$$

$$\frac{d\mathbf{P}(0)}{dt} = -3\mathbf{P}_0 + 3\mathbf{P}_1 = 3\overrightarrow{\mathbf{P}_0\mathbf{P}_1}$$

第 1, 2 制御点で終点での接線が決まる

$$\frac{d\mathbf{P}(1)}{dt} = -3\mathbf{P}_2 + 3\mathbf{P}_3 = 3\overrightarrow{\mathbf{P}_2\mathbf{P}_3}$$

第 3, 4 制御点で終点での接線が決まる



2階微分 → 曲率

$$\frac{d^2\mathbf{P}(t)}{dt^2} = 6(1-t)\mathbf{P}_0 + (18t - 12)\mathbf{P}_1 + (-18t + 6)\mathbf{P}_2 + 6t\mathbf{P}_3$$

パラメトリック曲線の曲率（参考）

- パラメトリック曲線 $\boldsymbol{p}(t) = (x(t), y(t))$ を弧長パラメータsで表したときのtとsの関係

$$\frac{ds}{dt} = \frac{d}{dt}\int_0^t ||\dot{\mathbf{p}}(t)||dt = ||\dot{\mathbf{p}}(t)|| = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} = \sqrt{\dot{x}^2 + \dot{y}^2} \dots (\mathrm{i})$$

- 曲率（弧長パラメータの２階微分の絶対値）

2階微分の計算

$$\frac{d}{ds}\left(\frac{d\mathbf{p}}{ds}\right) = \frac{d}{ds}\left(\frac{1}{\sqrt{\dot{x}^2+\dot{y}^2}}\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix}\right) = \frac{d}{dt}\left(\frac{1}{\sqrt{\dot{x}^2+\dot{y}^2}}\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix}\right)\frac{dt}{ds} = \frac{1}{\sqrt{\dot{x}^2+\dot{y}^2}}\frac{d}{dt}\left(\frac{1}{\sqrt{\dot{x}^2+\dot{y}^2}}\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix}\right)$$

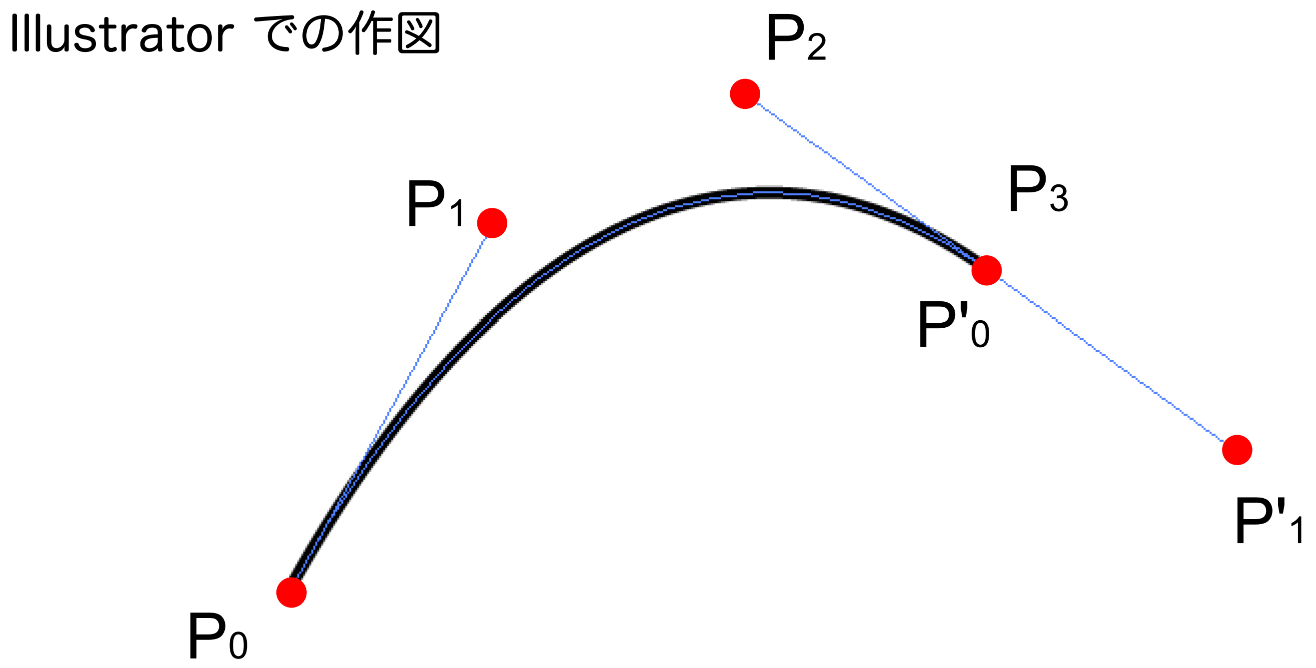
$$= \frac{1}{(\dot{x}^2+\dot{y}^2)^2}\begin{pmatrix}-\dot{x}(\dot{x}\ddot{x}+\dot{y}\ddot{y}) + \ddot{x}(\dot{x}^2+\dot{y}^2)\\ -\dot{y}(\dot{x}\ddot{x}+\dot{y}\ddot{y}) + \ddot{y}(\dot{x}^2+\dot{y}^2)\end{pmatrix} = \frac{1}{(\dot{x}^2+\dot{y}^2)^2}\begin{pmatrix}-\dot{y}(\dot{x}\ddot{y}-\dot{y}\ddot{x})\\ \dot{x}(\dot{x}\ddot{y}-\dot{y}\ddot{x})\end{pmatrix}$$

絶対値の計算

$$k = \left\|\frac{d}{ds}\left(\frac{d\mathbf{p}}{ds}\right)\right\| = \left(\frac{\dot{y}^2(\dot{x}\ddot{y}-\dot{y}\ddot{x})^2 + \dot{x}^2(\dot{x}\ddot{y}-\dot{y}\ddot{x})^2}{(\dot{x}^2+\dot{y}^2)^4}\right)^{1/2} = \left(\frac{(\dot{x}\ddot{y}-\dot{y}\ddot{x})^2}{(\dot{x}^2+\dot{y}^2)^3}\right)^{1/2} = \boxed{\frac{\dot{x}\ddot{y}-\dot{y}\ddot{x}}{(\dot{x}^2+\dot{y}^2)^{3/2}}}$$

曲線曲面の曲率 http://takashiijiri.com/study/miscs/Curvature.htm

複数セグメントの連結



・ 2 つのセグメントで端点を共有する
（制御点位置の共有, G^0連続→位置連続）

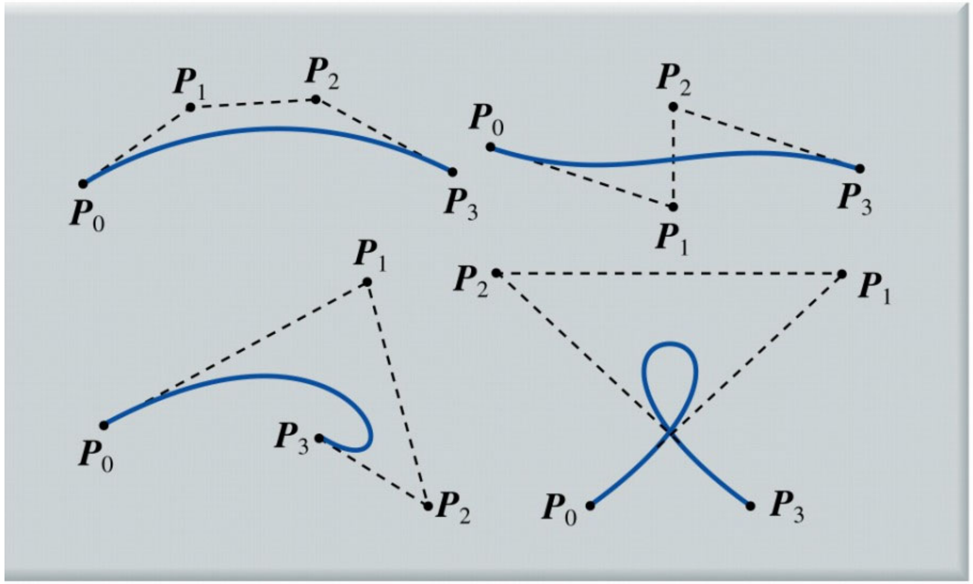
接続点で接線方向が同じ

・ 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，G^1連続→接線連続）

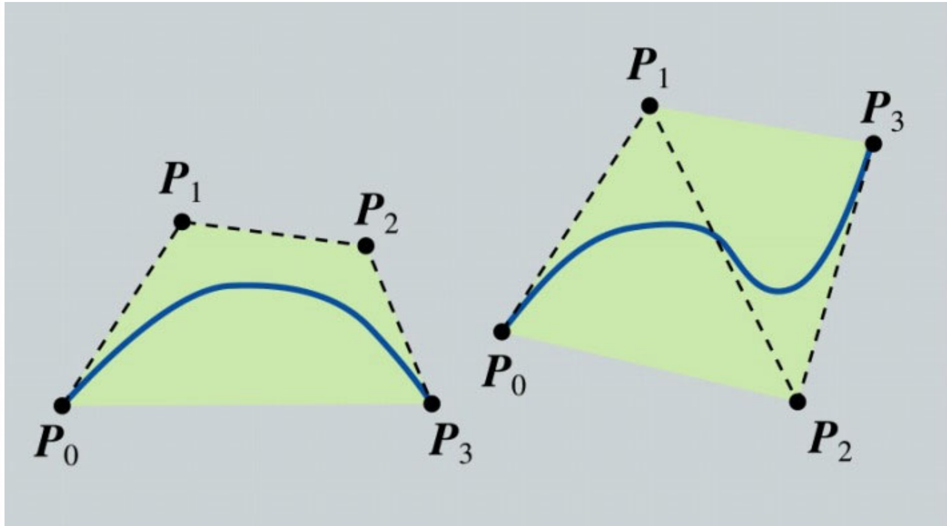
3 次ベジェ曲線の凸包性

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

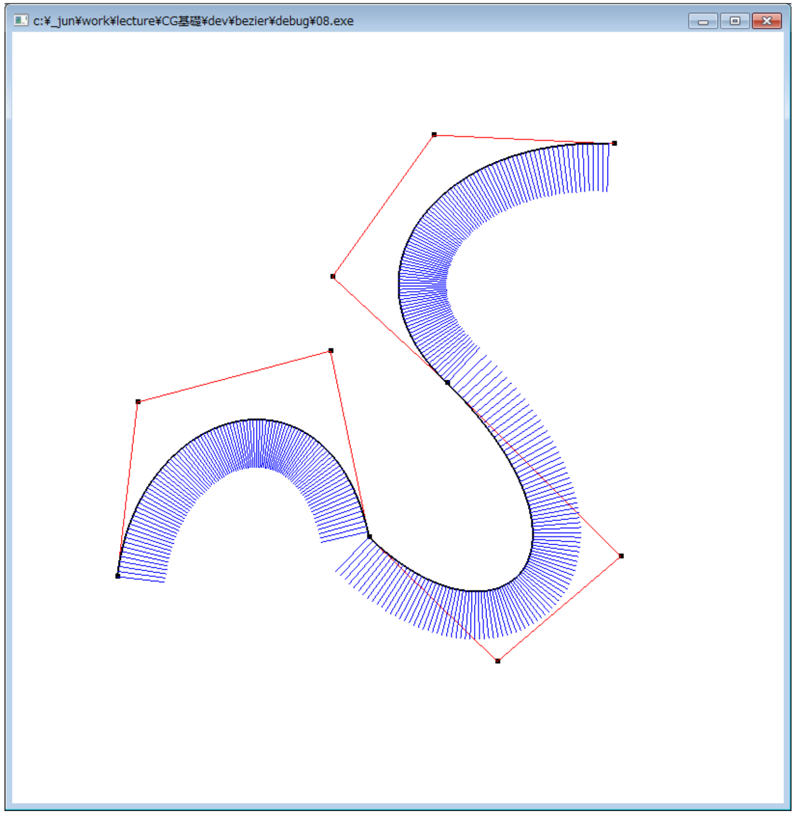


「コンピュータグラフィックス」2004年 / 財団法人画像情報教育振興協会（CG-ARTS協会）



課題

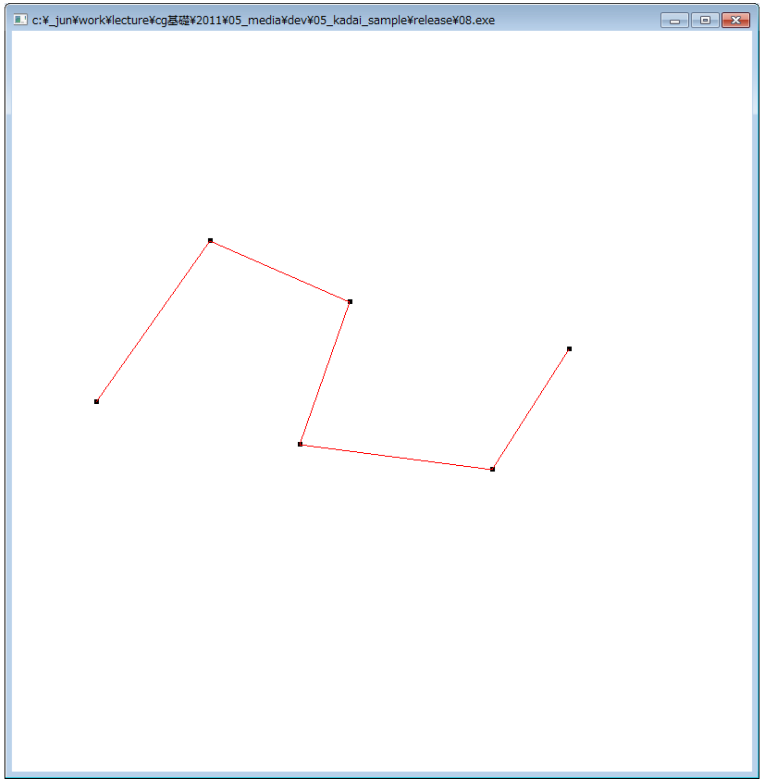
- 入力された制御点を使って、3 次の Bezier 曲線を描画する
- 法線（接線に垂直な線）を描画する



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



課題のヒント

- 3次ベジェ曲線の式

$$\mathbf{P}(t) = (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3$$

$\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ は制御点（マウスクリックした位置）

t の値を0～1の間で少しずつ大きくすると、点$\mathbf{P}(t)$がベジェ曲線上を移動する。それを線分で結べばベジェ曲線が描ける

接線方向は1階微分した次式で求まる

$$\frac{d\mathbf{P}(t)}{dt} = -3(1-t)^2\mathbf{P}_0 + (9t^2 - 12t + 3)\mathbf{P}_1 + (-9t^2 + 6t)\mathbf{P}_2 + 3t^2\mathbf{P}_3$$

法線は接線ベクトルを90度回転した方向
法線ベクトルを$(v'_x\ v'_y)$、接線ベクトルを(v_x, v_y)とすると　$v'_x = -v_y,\ v'_y = v_x$

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();
```