数値計算法・数値解析

補間：ラグランジュ補間

宮崎大学 工学部

第 3 回

補間

点群

$$\{(x_i, y_i) | i = 0, \ldots, n-1\}$$

が与えられた時，これらの点を全て通る曲線 $y = P(x)$ を推定して，$P(x_i)$ 以外の値を求めること．

n 個の点を通る曲線

⇒n-1 次多項式で補間することを考える

n-1 次多項式

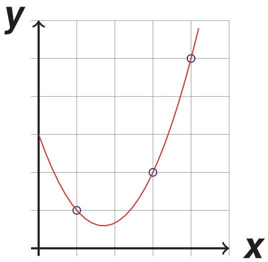
$$y = P(x) = \sum_{j=0}^{n-1} w_j x^j, \qquad w_j \text{ は n 個の係数}$$

n 個の未知変数に対して n 個の点が与えられる

⇒ 係数 w_j は一意に定まる

例

点群 **(1, 1), (3, 2), (4, 5)** を補間
点の数：３
　⇒２次多項式により補間
上記の３点を通る２次式を求めよ



ラグランジュ補間

補間多項式 $P(x)$ を求めるために毎回連立方程式を解くのは大変

$\Rightarrow$ プログラムで実装しやすい規則的な係数計算法

【ラグランジュ補間】

$$P(x) = \sum_{k=0}^{n-1} L_k(x) y_k$$

$$L_k(x) = \prod_{j=0, j \neq k}^{n-1} \frac{x - x_j}{x_k - x_j}$$

例題・プログラム

例題 1

点群 $(1, 1), (3, 2), (4, 5)$ のラグランジュ補間多項式を求める

例題 2

点群 $(1,1),(3,2),(4,5)$ をラグランジュ補間して,
$x = 6$ の時の値を求める

ラグランジュ補間のプログラム(1/2)

```
int main(int argc, char *argv[])
{
  double px[] = {1, 3, 4};
  double py[] = {1, 2, 5};
  int    size = sizeof(px)/sizeof(px[0]);

  double x = 3.0;
  printf("%f␣%.15f\n", x,
         lagrange(px, py, size, x));

  return 0;
}
```

ラグランジュ補間のプログラム (2/2)

```
double lagrange(double *px, double *py,
                int points, double x)
{ double        y = 0.0;
  for(int k = 0; k < points; k++)
  {
    double  l_k = 1.0;
    for(int j = 0; j < points; j++)
    {
      if (k == j) continue;
      l_k *= (x - px[j]) / (px[k] - px[j]);
    }
    y += l_k * py[k];
  }

  return y;
}
```

ラグランジュ補間の性質

【補間多項式 $P(x)$ の持つべき性質】

- n-1 次多項式
- $P(x_i) = y_i$ (点 (x_i, y_i) を通る)

$L_k(x)$ は x の n-1 回の掛け算：x の高々 n-1 次多項式

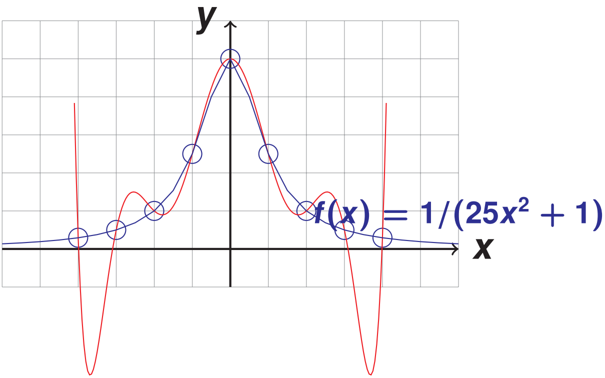
$$L_k(x_i)y_k = \begin{cases} y_i & (k = i\text{の時}) \\ 0 & (k \neq i\text{の時}) \end{cases}$$

従って，$P(x_i) = \sum_k L_k(x_i)y_k = y_i$

⇒ ラグランジュ補間により補間多項式が得られる
※n 点を通る n-1 次多項式は一意

ラグランジュ補間と誤差

x	-0.8	-0.6	-0.4	-0.2	0.0	0.2	0.4	0.6	0.8
y	0.059	0.1	0.2	0.5	1.0	0.5	0.2	0.1	0.059



非多項式関数を高次元ラグランジュ補間すると
大きな誤差が生じる場合がある