# トーンマッピング
# その２

効率的な計算方法
及び
区分多項式を用いた
自由度の高い関数形状

# 効率的な計算方法

〜重複する計算を極力省くには〜

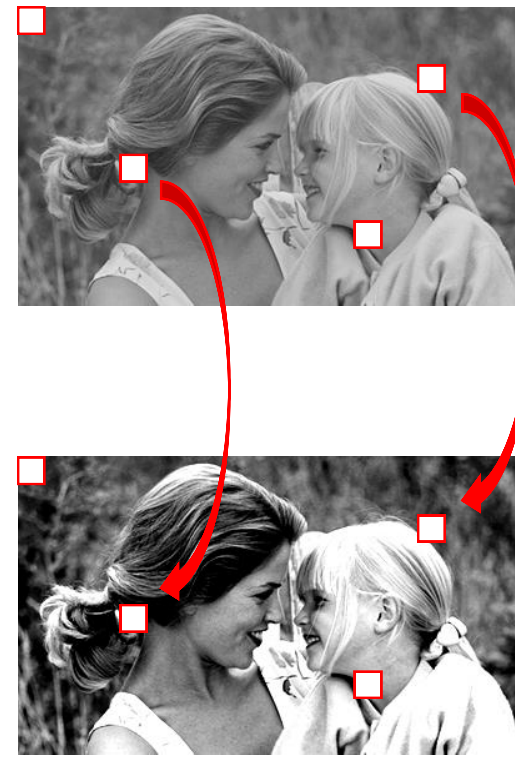
# 効率的な計算と任意関数の扱いのために

- 計算量
  - 各画素において，
    トーンマッピングの計算が行われている．
  - １画素あたり，$O$ の計算量がかかるとすると，
    全画素 $N$ では，　$O\times N$ の計算量となる．

    スマホのカメラでの最大画像サイズは，
    3000 × 4000 ＝ 12 × $10^6$ ピクセル．

    １秒で処理するには，
    １画素あたり，0.083 μs で処理する必要がある．

    60 フレーム／秒で処理するには，
    １画素あたり，0.0013 μs （1.38 ps）で処理する必要がある．



# 効率的な計算と任意関数の扱いのために

- 画像の特徴
  - まったく同じ輝度値をもつ画素が多数存在する．
  - 例えば，右の図では，0.3922 (＝100/255) の値をもつ画素は2710 個存在する．
  - 1 画素ずつトーンマッピングを行うと，同じ計算を 2710 回する必要がある．
  - 計算の無駄である．

輝度値 0.3922 (100/255) を持つ画素
2710 個

輝度値 0.7059 (180/255) を持つ画素
1006 個

# 高効率な計算（1／2）

- 画素をグルーピングする方法
  - 同一輝度値をもつ画素を
    リストアップする.
  - ヒストグラムを作成する際に,
    画素番号をスタックしていく.
  - 輝度値に対して
    トーンマッピングを施し,
    その結果を，同行の画素に反映させる.

| 階調番号 | 輝度値 | 画素番号のリスト |
|---|---|---|
| 1 | 0/255 | 13, 15, ... |
| 2 | 1/255 | 14, 18, 19, 30, ... |
| 3 | 2/255 | 28, 33, 78,... |
| 255 | 254/255 | 64, 98, 123,... |
| 256 | 255/255 | 63, 101,... |

| | |
|---|---|
| 1 | 0/255 |
| 2 | 1/255 |
| 3 | 2/255 |

トーン
マッピング

| |
|---|
| 0.1 |
| 0.13 |
| 0.17 |

画素に
割り当てる

| |
|---|
| 13, 15, ... |
| 14, 18, 19, 30, ... |
| 28, 33, 78,... |

# 高効率な計算（２／２）

- 更に簡略化
  - 入力輝度値の範囲が分かる場合
  - ステップ１
    - 予め，入力輝度値の候補を作成
    - トーンマッピングを行い，
      出力輝度値を生成
  - ステップ２
    - 画素値を走査し，
      注目画素値に最も近い
      入力輝度値の輝度値番号を探す．
    - 対応する出力輝度値を得る．

予め入力値に対する出力値を計算した表のことを
ルックアップテーブルと呼ぶ

トーンマッピング

| 階調番号 | 入力輝度値 | 出力輝度値 |
|---|---|---|
| 1 | 0/255 | 0.1 |
| 2 | 1/255 | 0.13 |
| 3 | 2/255 | 0.17 |

# プログラム（1／4）

- 入力画像の輝度値を N 階調に分割
  - ```
    I = im2double( imread('../images/family.png') );
    Iycc = rgb2ycbcr( I );
    G = Iycc(:,:,1);
    % 輝度番号の生成
    N = 256; % 階調数
    Tone = round( G * (N-1) ) + 1;
    % G は [0,1] までの値
    % Tone は階調番号 [1,N] までの N 個の値
    ```
- 入力輝度値を別途用意し，トーンマッピングする．
  - ```
    X = (0:N-1)/(N-1);  % [0,1] までの N 個の値
    Y = 1.5 * X - 0.1;  % 例えば，線形濃度変換
    Y = max(0,min(1,Y));
    ```

# プログラム（２／４）

- 輝度値の探索
  - 例えば，１番目の画素であれば，

```
i = 1;
tone = Tone(i); % 輝度値に紐付いた階調番号を得る

g = G(i)     % 入力画像の画素値
x = X(tone)  % 入力画素値
y = Y(tone)  % 出力画素値

% g と x の値は，完全一致はしないが，
% ほぼ同値となる．
% g ≒ x
% G(i) ≒ X(tone) == X(Tone(i))
```

# プログラム（３／４）

- 全ての画素値に，トーンマッピング結果を反映させる．
  - ```
    G_out = G;  % 出力画像を用意

    for i = 1:numel(G) % numel() で要素数を取得
      tone = Tone(i);  % 輝度値に紐付いた階調番号を得る

      G_out(i) = Y(tone); % トーンマップ結果を出力
      % 入力は
      % G(i) ≒ X(tone) ＝ X(Tone(i))
      % 出力は
      % G_out(i) ＝ Y(tone) ＝ ToneMap( X(tone) )
      %           ≒ ToneMap( G(i) )
    end
    ```
    ```

# プログラム（４／４）

- トーンマップした輝度成分をカラー画像へと反映
  - ```
    Jycc = Iycc;  % 出力画像を用意
    Jycc(:,:,1) = G_out;
    % Jycc = cat( 3, G_out, Iycc(:,:,1:2) ); でも良い

    J = ycbcr2rgb( Jycc );
    figure(1), imshow( [I,J] );
    ```

入力画像

効率的な計算による
トーンマッピング結果

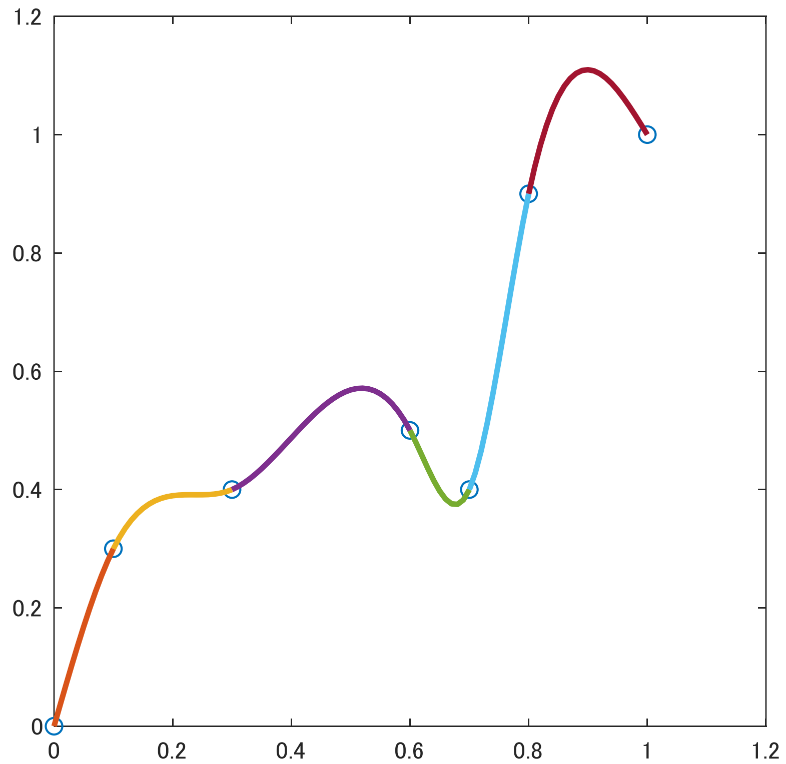
全画素において
トーンマッピングを行う
ナイーブな手法の結果
    ```

# 区分多項式を用いた
# 自由度の高い関数形状

〜曲線の数式を用いた描き方〜

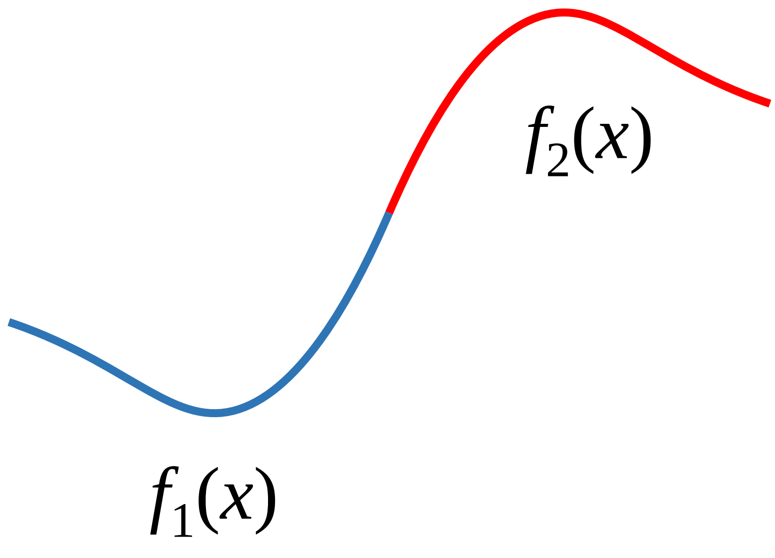
# 複雑なトーンマッピング関数の作成

- 区分多項式（スプライン）補間を用いる方法
  - ユーザーが与えた点群を通るような
    ３次関数をつなぎ合わせることで,
    複雑な形状の曲線を作り出す方法.



# 連続する区分多項式とその条件（１／５）

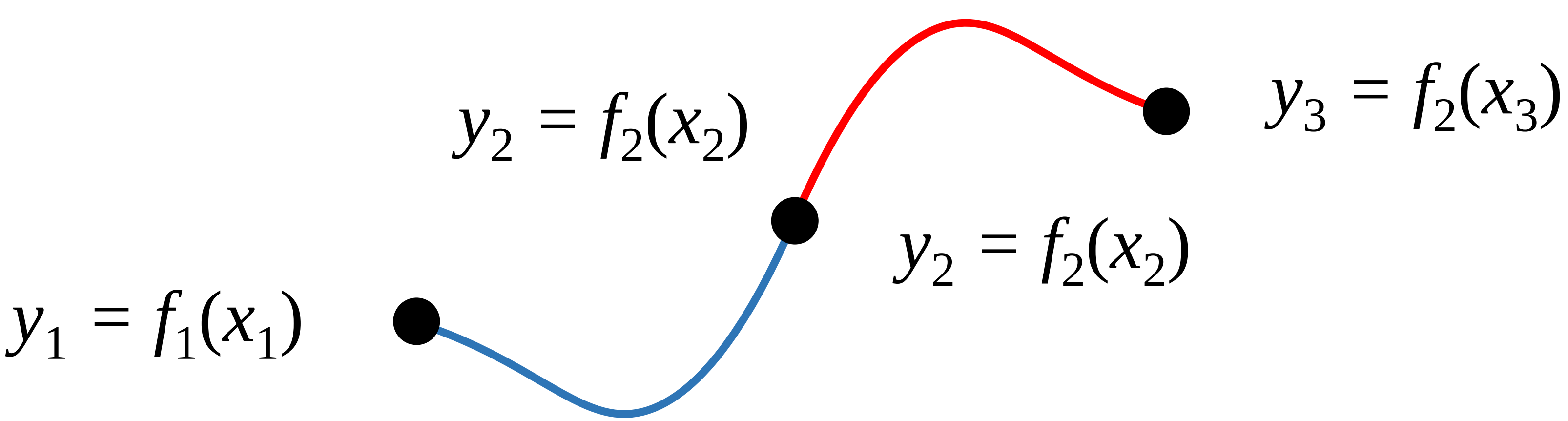
- ２つの隣あう３次多項式を定義
  - $f_1(x) = a_1 x^3 + b_1 x^2 + c_1 x + d_1$
  - $f_2(x) = a_2 x^3 + b_2 x^2 + c_2 x + d_2$
    - $\{a_1, b_1, c_1, d_1\}$ と $\{a_2, b_2, c_2, d_2\}$ の値を求めたい.



# 連続する区分多項式とその条件（２／５）

- 条件１：ユーザーが与えた点を通る
  - ３点 $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ を与えたとき
  - 端点
    - $f_1(x_1) = a_1 x_1^3 + b_1 x_1^2 + c_1 x_1 + d_1 = y_1$
    - $f_2(x_3) = a_2 x_3^3 + b_2 x_3^2 + c_2 x_3 + d_2 = y_3$
  - 中間点
    - $f_1(x_2) = a_1 x_2^3 + b_1 x_2^2 + c_1 x_2 + d_1 = y_2$
    - $f_2(x_2) = a_2 x_2^3 + b_2 x_2^2 + c_2 x_2 + d_2 = y_2$

$f_1(x_2) = f_2(x_2)$
でもある



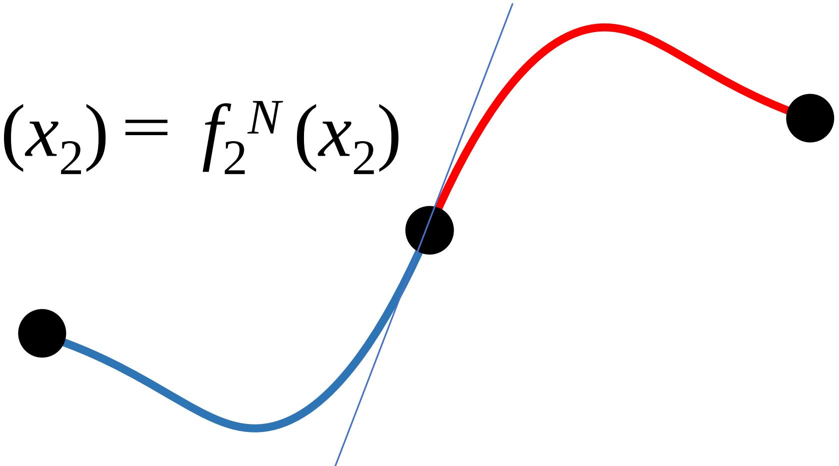
# 連続する区分多項式とその条件（３／５）

- 条件２：連続条件　２つの関数は滑らかに接続する
  - 「変化が滑らか」とは「微分値が同値」ということ
  - 中間点で１階微分～２階微分が等しい
    - $f_1'(x_2) = f_2'(x_2) \Rightarrow a_1 3x_2^2 + b_1 2x_2 + c_1 = a_2 3x_2^2 + b_2 2x_2 + c_2$
    - $f_1''(x_2) = f_2''(x_2) \Rightarrow a_1 6x_2 + b_1 2 = a_2 6x_2 + b_2 2$

    - ３次を含めると，自由度が減るため，含めない．

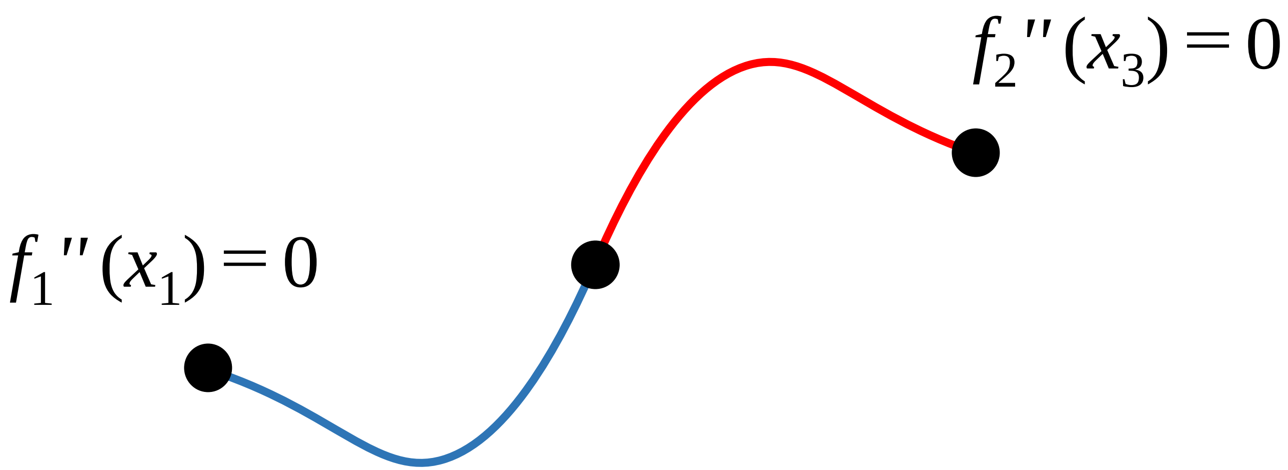
      $f_1'''(x_1) = f_2'''(x_1) \qquad a_1 = a_2$

$$\forall_N \ f_1^N(x_2) = f_2^N(x_2)$$



# 連続する区分多項式とその条件（４／５）

- 条件３：境界条件　端点での条件
  - 端点で２階微分が 0 となる
    - $f_1''(x_1) = a_1 6x_1 + b_1 2 = 0$
    - $f_2''(x_3) = a_2 6x_3 + b_2 2 = 0$



# 連続する区分多項式とその条件（５／５）

- 全ての条件
- 8 個の未知数に対して，8 本の式があるため，方程式を解くことができる

端点

- $a_1 x_1^3 + b_1 x_1^2 + c_1 x_1 + d_1 = y_1$
- $a_1 6x_1 + b_1 2 = 0$

中間点

- $a_1 x_2^3 + b_1 x_2^2 + c_1 x_2 + d_1 = y_2$
- $a_2 x_2^3 + b_2 x_2^2 + c_2 x_2 + d_2 = y_2$
- $a_1 3x_2^2 + b_1 2x_2 + c_1 = a_2 3x_2^2 + b_2 2x_2 + c_2$
- $a_1 6x_2 + b_1 2 = a_2 6x_2 + b_2 2$

端点

- $a_2 x_3^3 + b_2 x_3^2 + c_2 x_3 + d_2 = y_3$
- $a_2 6x_3 + b_2 2 = 0$

# 数式の整理

- 全ての条件
- 8 個の未知数に対して，8 本の式があるため，方程式を解くことができる

端点

- $a_1\, x_1^3 + b_1\, x_1^2 + c_1\, x_1 + d_1 \ = \ y_1$
- $a_1\, 3x_1 + b_1 \ = \ 0$

中間点

- $a_1\, x_2^3 + b_1\, x_2^2 + c_1\, x_2 + d_1 \ = \ y_2$
- $a_1\, x_2^3 + b_1\, x_2^2 + c_1\, x_2 + d_1 - a_2\, x_2^3 - b_2\, x_2^2 - c_2\, x_2 - d_2 \ = \ 0$
- $a_1\, 3x_2^2 + b_1\, 2x_2 + c_1 - a_2\, 3x_2^2 - b_2\, 2x_2 - c_2 = 0$
- $a_1\, 3x_2 + b_1 - a_2\, 3x_2 - b_2 = 0$

端点

- $a_2\, x_3^3 + b_2\, x_3^2 + c_2\, x_3 + d_2 \ = \ y_2$
- $a_2\, 3x_3 + b_2 \ = \ 0$

# 線形代数を用いた表現

$$
\begin{array}{rl}
 & \qquad\qquad\qquad \text{関数 1} \qquad\qquad\qquad\qquad\qquad\qquad \text{関数 2} \\
\begin{array}{c} \text{端点} \\ \\ \\ \text{中間点} \\ \\ \\ \\ \text{端点} \end{array} &
\left(\begin{array}{rrrr|rrrr}
x_1^3 & x_1^2 & x_1 & 1 & 0 & 0 & 0 & 0 \\
3x_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
x_2^3 & x_2^2 & x_2 & 1 & 0 & 0 & 0 & 0 \\
x_2^3 & x_2^2 & x_2 & 1 & -x_2^3 & -x_2^2 & -x_2 & -1 \\
3x_2^2 & 2x_2 & 1 & 0 & -3x_2^2 & -2x_2 & -1 & 0 \\
3x_2 & 1 & 0 & 0 & -3x_2 & -1 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & x_3^3 & x_3^2 & x_3 & 1 \\
0 & 0 & 0 & 0 & 3x_3 & 1 & 0 & 0
\end{array}\right)
\left(\begin{array}{c}
a_1 \\ b_1 \\ c_1 \\ d_1 \\ \hline a_2 \\ b_2 \\ c_2 \\ d_2
\end{array}\right)
=
\left(\begin{array}{r}
y_1 \\ 0 \\ \hline y_2 \\ 0 \\ 0 \\ 0 \\ \hline y_3 \\ 0
\end{array}\right)
\end{array}
$$

$$\mathbf{A}\mathbf{p} = \mathbf{b}$$

$$\mathbf{p} = \mathbf{A}^{-1}\mathbf{b}$$

行列 **A** とベクトル **b** を
作れば,
関数のパラメータは
求まる

# 点を増やした場合

- 4 点 $(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$ を与えたとき

$$
\begin{array}{c c}
 & \begin{array}{cccc} \text{関数 1} & \text{関数 2} & \text{関数 3} & \end{array} \\
\begin{array}{c} \text{端点} \\ \\ \text{中間点} \\ \\ \\ \\ \text{中間点} \\ \\ \\ \\ \text{端点} \\ \\ \end{array}
&
\begin{pmatrix}
x_1^3 & x_1^2 & x_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3x_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
x_2^3 & x_2^2 & x_2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
x_2^3 & x_2^2 & x_2 & 1 & -x_2^3 & -x_2^2 & -x_2 & -1 & 0 & 0 & 0 & 0 \\
3x_2^2 & 2x_2 & 1 & 0 & -3x_2^2 & -2x_2 & -1 & 0 & 0 & 0 & 0 & 0 \\
3x_2 & x_2 & 0 & 0 & -3x_2 & -x_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & x_3^3 & x_3^2 & x_3 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & x_3^3 & x_3^2 & x_3 & 1 & -x_3^3 & -x_3^2 & -x_3 & -1 \\
0 & 0 & 0 & 0 & 3x_3^2 & 2x_3 & 1 & 0 & -3x_3^2 & -2x_3 & -1 & 0 \\
0 & 0 & 0 & 0 & 3x_3 & x_3 & 0 & 0 & -3x_3 & -x_3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_4^3 & x_4^2 & x_4 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3x_4 & 1 & 0 & 0
\end{pmatrix}
\begin{pmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \\ a_2 \\ b_2 \\ c_2 \\ d_2 \\ a_3 \\ b_3 \\ c_3 \\ d_3 \end{pmatrix}
=
\begin{pmatrix} y_1 \\ 0 \\ y_2 \\ 0 \\ 0 \\ 0 \\ y_3 \\ 0 \\ 0 \\ 0 \\ y_4 \\ 0 \end{pmatrix}
\end{array}
$$

プログラミング時には，疎行列に関する関数（spalloc や sparse など）を用いて作成する必要があるため，今回の演習では 4 点以上は行わない．

# プログラム（１／３）

- 簡単のため，３点 $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ を与える
  $\mathbf{x} = \{x_1, x_2, x_3\}$，$\mathbf{y} = \{y_1, y_2, y_3\}$ として与える．
- 行列 A もフル行列（0 の部分も値として保持する）で表す．
  - ```
    px = [0,0.3,1]; py = [0, 0.4,1];

    b = [py(1); 0; py(2); 0; 0; 0; py(3); 0];

    x1 = px(1);
    A1 = [ x1^3, x1^2, x1, 1, 0, 0, 0, 0;
           3*x1,    1,  0, 0, 0, 0, 0, 0;];

    x3 = px(3);
    A3 = [0, 0, 0, 0, x3^3, x3^2, x3, 1;
          0, 0, 0, 0, 3*x3,    1,  0, 0;];
    ```
    ```

# プログラム（２／３）

```
x2 = px(2);
A2 = [
      x2^3, x2^2, x2, 1,        0,      0,   0,  0;
      x2^3, x2^2, x2, 1,    -x2^3, -x2^2, -x2, -1;
    3*x2^2, 2*x2,  1, 0, -3*x2^2, -2*x2,  -1,  0;
      3*x2,    1,  0, 0,    -3*x2,    -1,   0,  0;
];

A = [A1; A2; A3];
p = inv(A)*b;

% 関数形状の表示
pori = @(X,P) P(1)*X.^3 + P(2)*X.^2 + P(3)*X + P(4);
figure(1), plot( X, pori(X,p(1:4)), X, pori(X,p(5:8)) );
```

多項式を定義
簡単な関数であれば,
ラムダ形式で定義

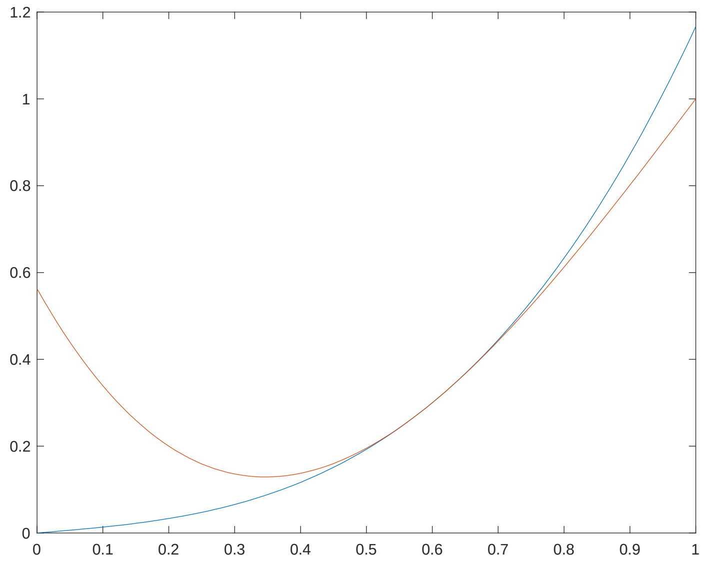
# プログラム（３／３）

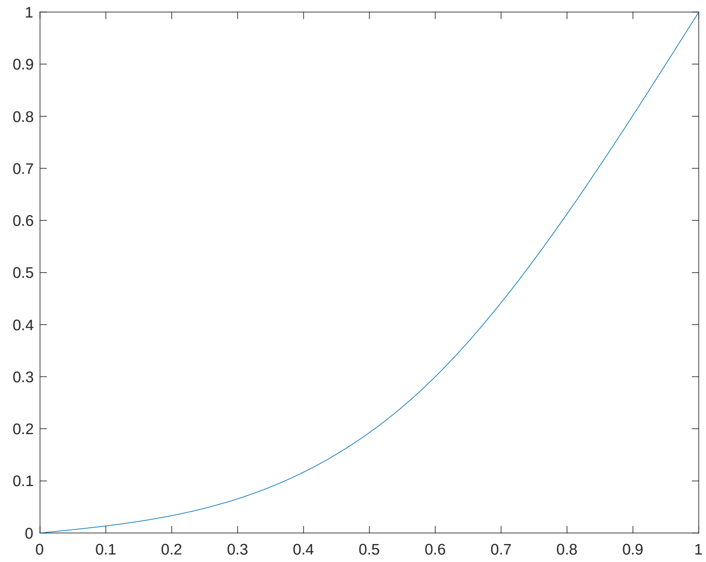
- 入力輝度値が，どの区間に対応するかを調べ，パラメータを切り替える．
  - ```
    N = 256;
    X = (0:N-1)/(N-1);     % 入力輝度値
    Y = zeros( size(X) ); % 出力輝度値

    for i = 1:numel(X)
    x = X(i);
    % px 内で x を超える位置を探索
    ind = find( px >= x, 1 ) - 1;  % px >= x となる最初の位置の１つ前
    ind = max( 1, ind ); % x=0 で i=0 となるため修正

    Y(i) = pori( x, p((4*ind-3):(4*ind)) );
    end
    Y = max(0,min(1,Y));    % Octave では ycbcr2rgb 用に必要
    figure(2), plot( X, Y );
    ```
    ```

# 実行結果




２つの関数を表示したもの

途中で切り替えたもの

# 前半のプログラムへの組み込み（１／２）

- 後半のプログラムを関数化する.
  - ```
    function Y = tone_mapping_cubic_spline_LUT( X, px, py )

    b = [py(1); 0; py(2); 0; 0; 0; py(3); 0];

    x1 = px(1);  A1 = [x1^3, x1^2, x1, 1, 0, 0, 0, 0; 3*x1, 1, 0, 0, 0, 0, 0, 0;];
    x3 = px(3);  A3 = [0, 0, 0, 0, x3^3, x3^2, x3, 1; 0, 0, 0, 0, 3*x3, 1,  0, 0;];
    x2 = px(2);
    A2 = [x2^3, x2^2, x2, 1, 0, 0, 0, 0; x2^3, x2^2, x2, 1, -x2^3, -x2^2, -x2, -1;
          3*x2^2, 2*x2, 1, 0, -3*x2^2, -2*x2, -1, 0; 3*x2, 1, 0, 0, -3*x2, 1, 0, 0;];
    A = [A1; A2; A3];
    p = inv(A)*b;

    pori = @(X,P) P(1)*X.^3 + P(2)*X.^2 + P(3)*X + P(4);
    figure(2), plot( X, pori(X,p(1:4)), X, pori(X,p(5:8)) );

    Y = zeros( size(X) );
    for i = 1:numel( X )
      x = X(i);
      ind = find( px >= x, 1 ) - 1;
      ind = max( 1, ind );
      Y(i) = pori( x, p((4*ind-3):(4*ind)) );
    end
    Y = max(0,min(1,Y));
    figure(3), plot( X, Y );
    end
    ```
    ```

# 前半のプログラムへの組み込み（２／２）

- 線形濃度変換の箇所を置き換える.

```
I = im2double( imread('../images/family.png') );
Iycc = rgb2ycbcr( I );
G = Iycc(:,:,1);

N = 256;
Tone = round( G * (N-1) ) + 1;

X = (0:N-1)/(N-1);

%Y = 1.5 * X - 0.1;  % 例えば, 線形濃度変換
%Y = max(0,min(1,Y));
Y = tone_mapping_cubic_spline_LUT( X, [0,0.3,1], [0,0.4,1] );

G_out = G;
for i = 1:numel(G)
  tone = Tone(i);
  G_out(i) = Y(tone);
end

Jycc = Iycc;
Jycc(:,:,1) = G_out;

J = ycbcr2rgb( Jycc );
figure(1), imshow( [I,J] );
```