

MATLAB で Audio Plugin 開発

#03 ディストーション

松本 和樹 (早稲田大学, MATLAB Student Ambassador)

はじめに

自己紹介



氏名 : 松本 和樹
所属 : 早稲田大学
研究 : 音響信号処理
趣味 : 作曲
仕事 : MATLAB Student Ambassador
@km_MATLAB_Amb 

MATLAB で Audio Plugin 開発

Audio Plugin とは

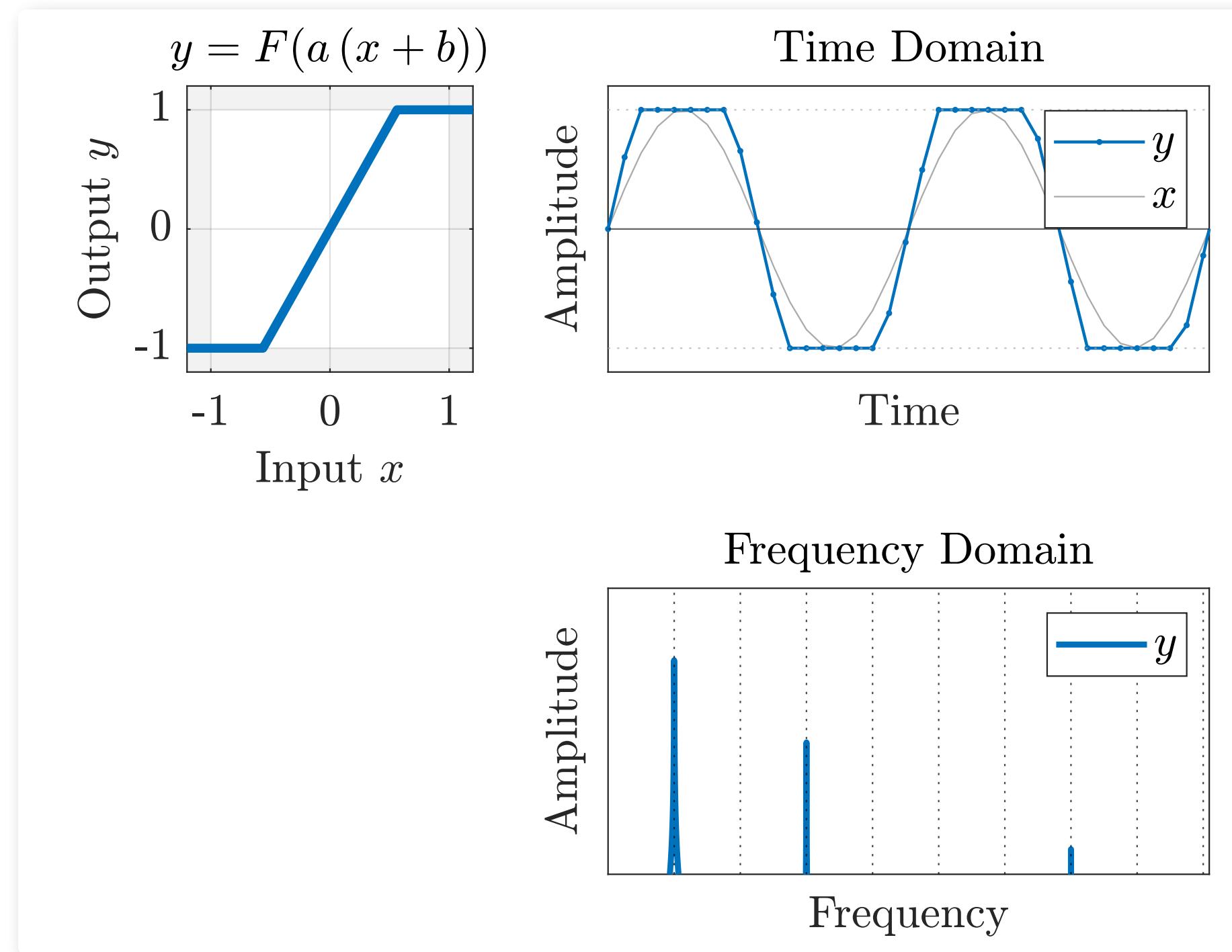
- DAW 等の音楽制作ソフトウェア上で動作する拡張機能
- MATLAB の **Audio Toolbox**  で手軽な開発が可能

シリーズの内容

- 第1回 : ゲイン (+プラグインの作りの基礎) 
- 第2回 : イコライザ 
- 第3回 : ディストーション 

ディストーション

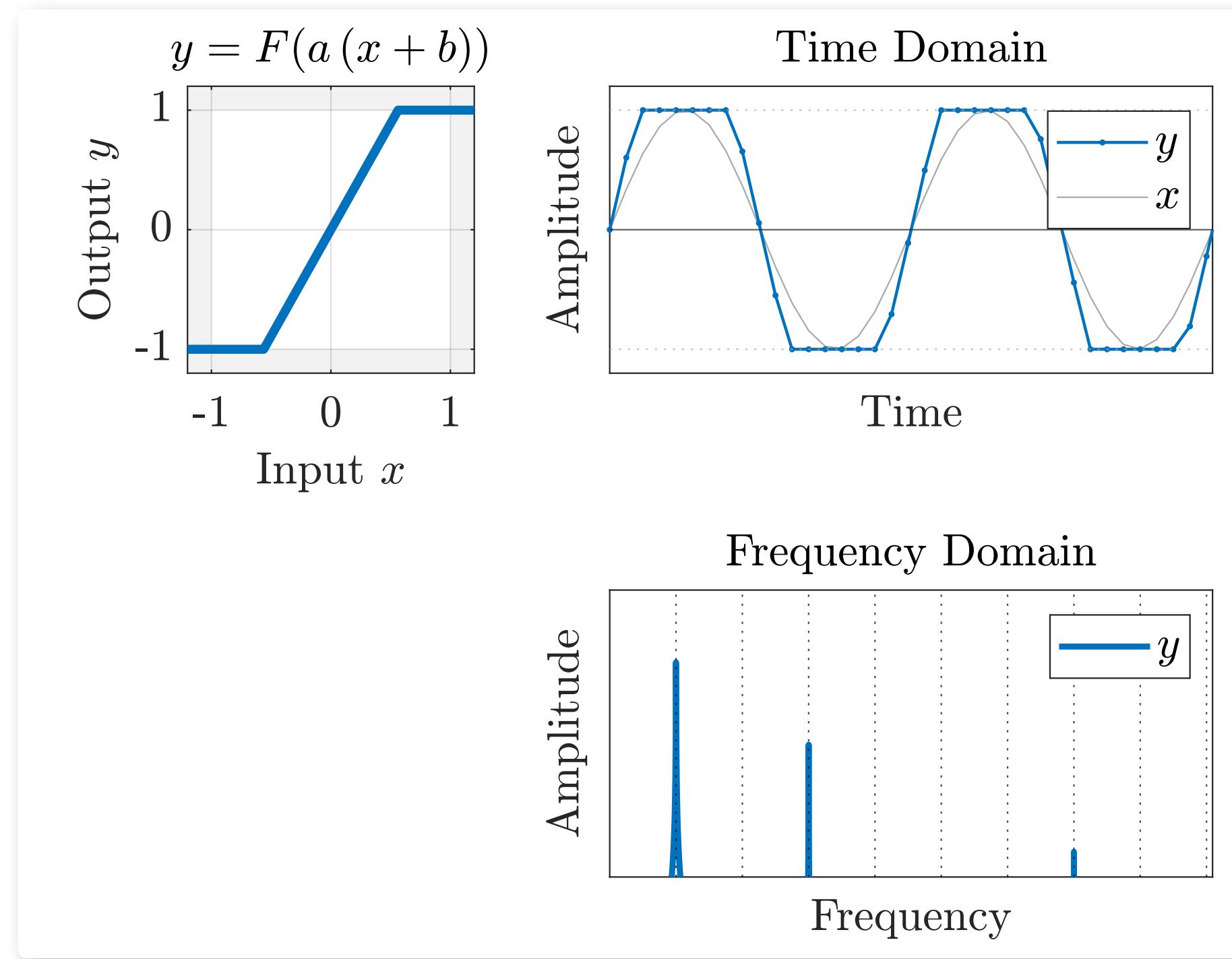
ディストーションとは



- 音を歪ませるエフェクタ
 - エレキギター
 - 音割れ
- 歪みによって倍音が付加される

ディストーションの効果 : `preGain = 5`

ディストーションの仕組み



ディストーションの効果 : `preGain = 5`

基本的な仕組みは単純

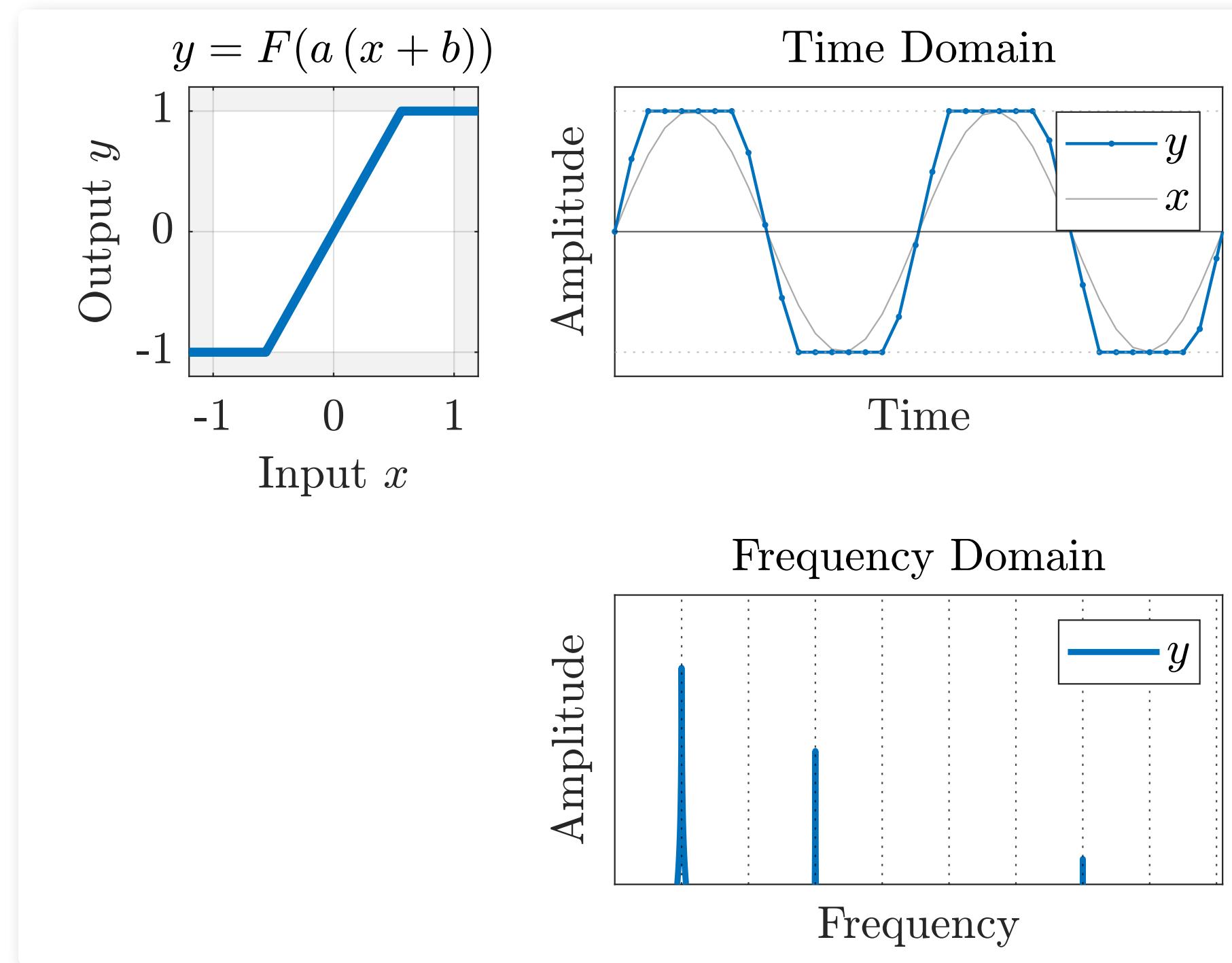
- 非線形関数 F を入力信号 $x(t)$ に適用
- 関数 F 適用前に平行移動（定数 b の加算） & 振幅を増幅（定数 a 倍）することで歪みを調整

$$y(t) = F(a(x(t) + b))$$

- Hardclipper が典型例：好みの関数を見つけよう！

$$F_{\text{Hardclip}}(x) = \begin{cases} 1 & (x > 1) \\ x & (-1 \leq x \leq 1) \\ -1 & (x < -1) \end{cases}$$

ディストーションの実装



ディストーションの効果 : `preGain = 5`

実装

```
F = @(x) min(max(x,-1),1); % Hard Clipping の関数

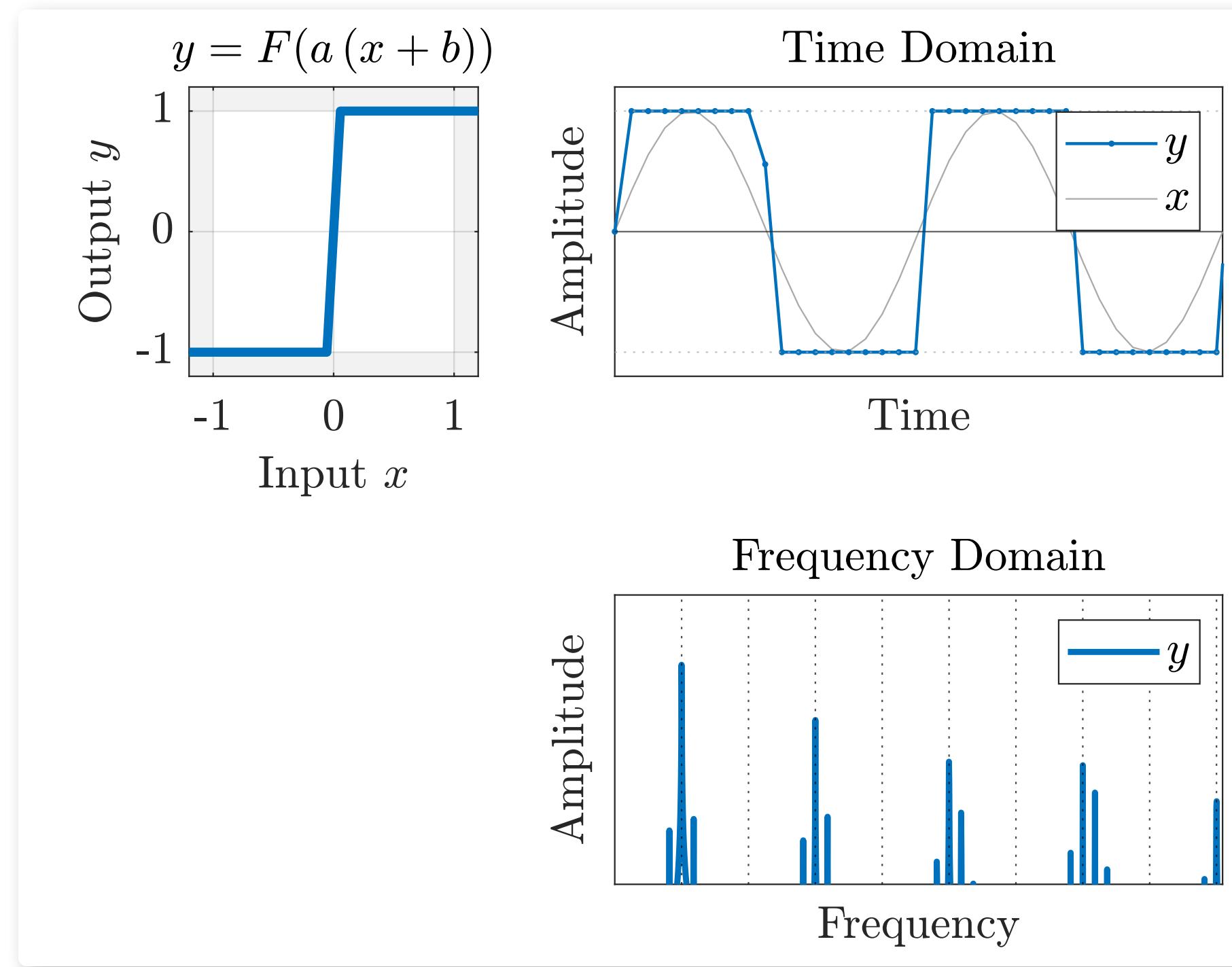
x = sin(2 * pi * freq * t); % 正弦波の作成

a = db2mag(preGain);           % preGain [dB] の単位変換

y = F(a.*(x + b));            % ディストーションの適用
```

✓ 元信号の周波数の整数倍の成分 (= 倍音) が現れた！

ディストーションの実装



ディストーションの効果 : `preGain = 25`

実装

```
F = @(x) min(max(x, -1), 1); % Hard Clipping の関数

x = sin(2 * pi * freq * t); % 正弦波の作成

a = db2mag(preGain);           % preGain [dB] の単位変換

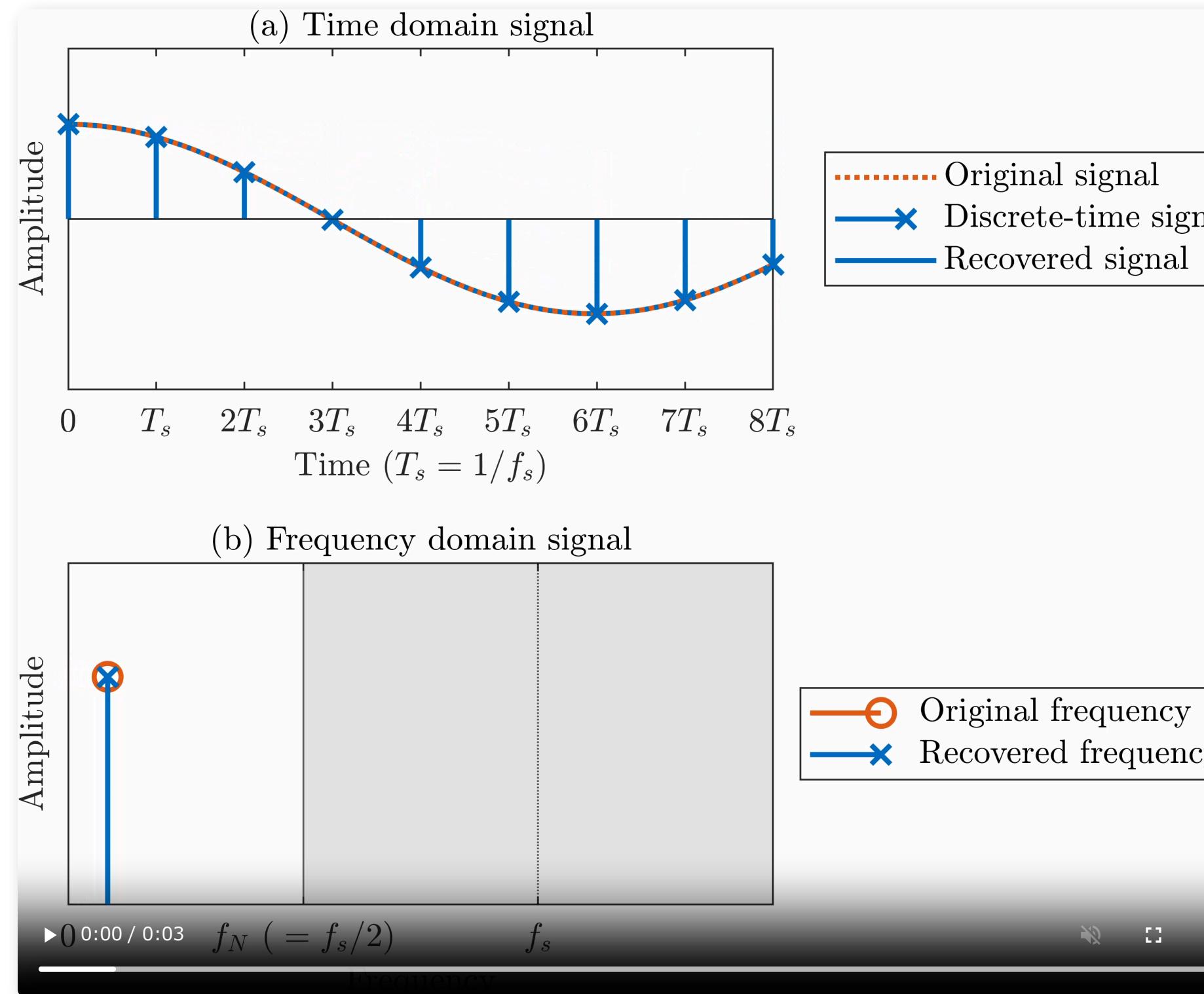
y = F(a.* (x + b));           % ディストーションの適用
```

しかし、ゲインをあげてみると？

- ザラザラした音がする 🤔
- スペクトルを確認 ▶ 整数次の倍音（点線）だけでなく
非整数次の倍音まで現れている 🤔

サンプリング定理とオーバーサンプリング

サンプリング定理

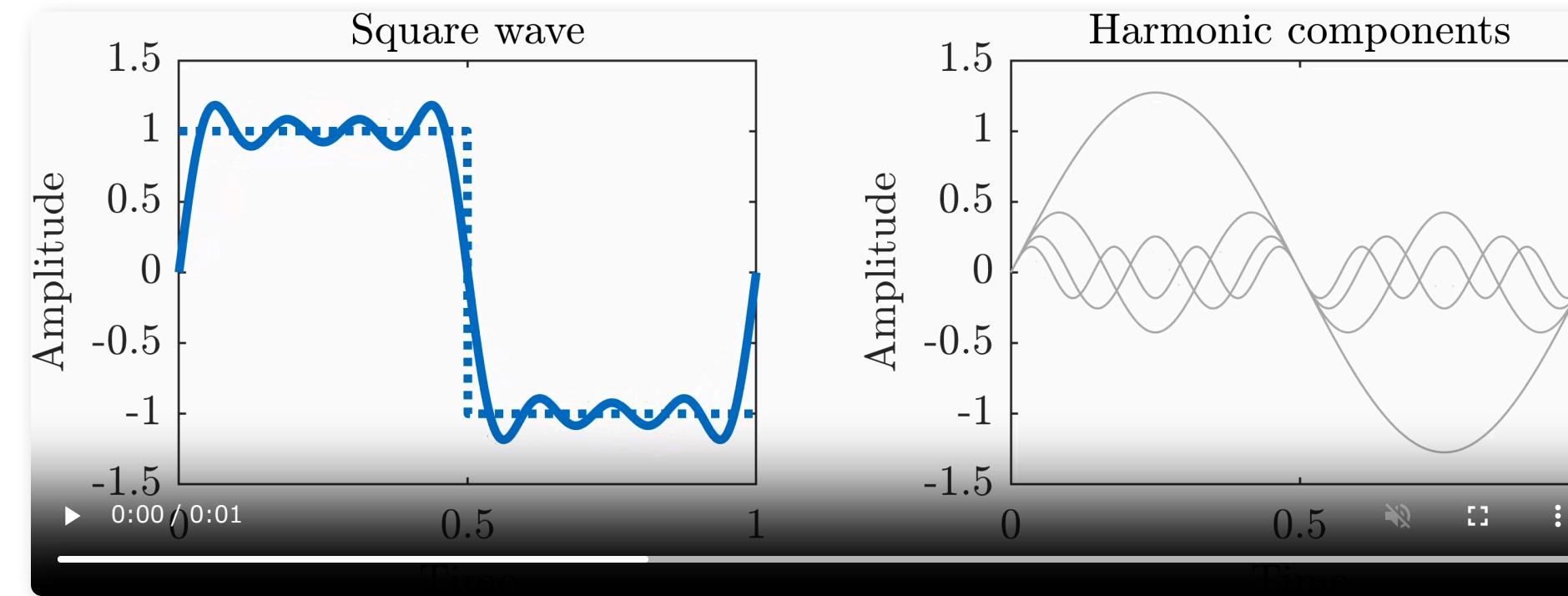


サンプリング定理の直観的理解

離散時間信号で扱える帯域幅には限りがある

- 音響信号は音波を一定間隔でサンプリングしたもの
- サンプリング周波数 f_s** ：一秒あたりのサンプル数
- オレンジ**がナイキスト周波数 $f_N = (f_s/2)$ 超えると、
 $0 \sim f_N$ の範囲に存在する別の正弦波（**青**）と
区別がつかない
- オレンジ**をサンプリングしたにもかかわらず、
周波数が折り返された別の音（**青**）として再生される
▶ **エイリアシング**と呼ばれる現象
- エイリアシングにより生じる雑音を**エイリアスノイズ**
（折り返し雑音） と呼ぶ

ディストーションは高周波数成分を生む



矩形波のフーリエ級数展開

- 以下のようなディストーションを考える

$$F(x) = \begin{cases} 1 & (x \geq 0) \\ -1 & (x < 0) \end{cases}$$

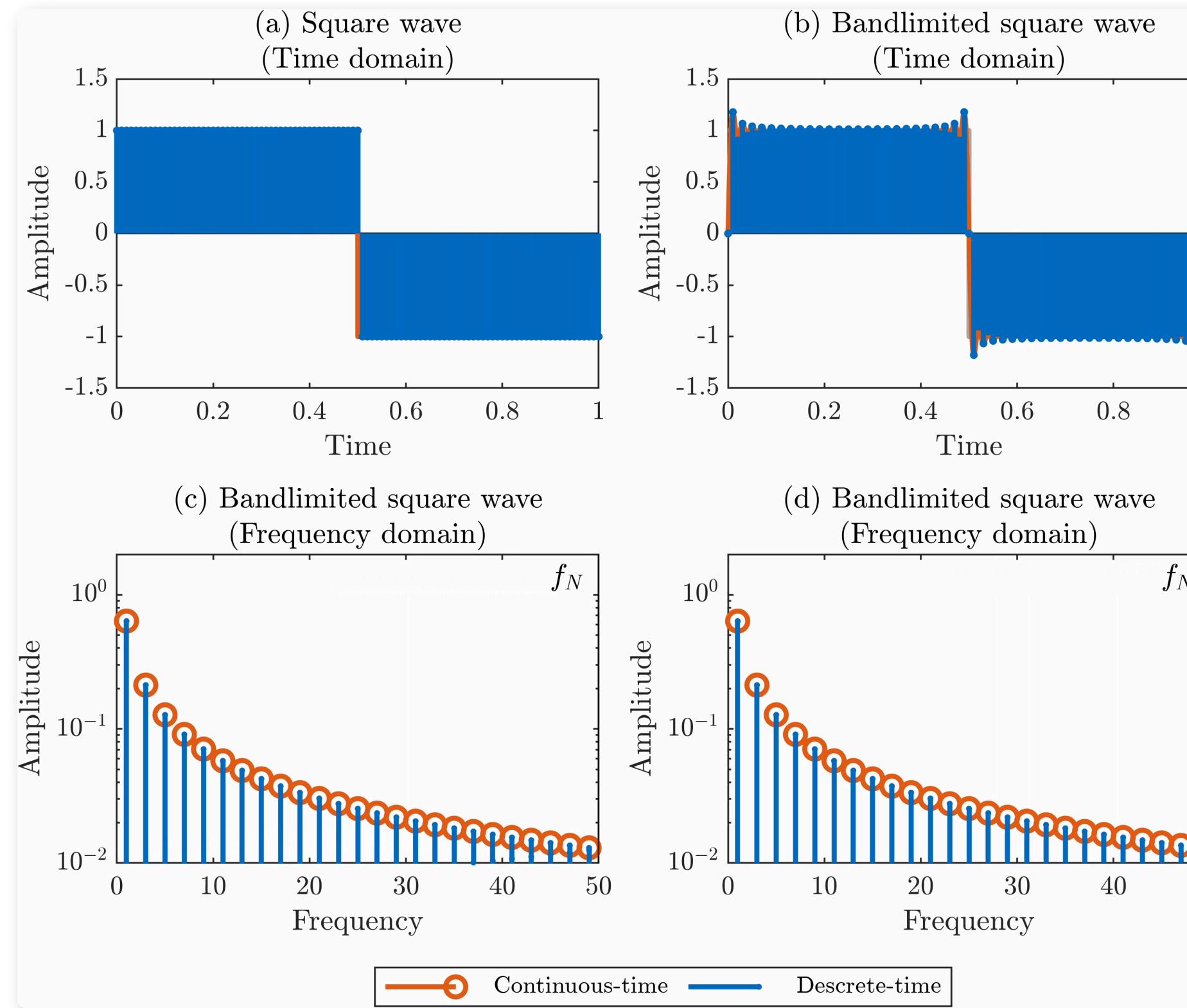
この関数は正弦波から**矩形波** $y_{\text{square}}(t)$ を生む

- 矩形波を正弦波の和に分解 (フーリエ級数展開)
 - 無限に高い周波数の成分が含まれている

$$y_{\text{square}}(t) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin(2\pi(2k+1)t)}{2k+1}$$

- ディストーション適用後の波形は**帯域幅が無限大**
 - エイリアシングが発生してしまう

アンチエイリアシング

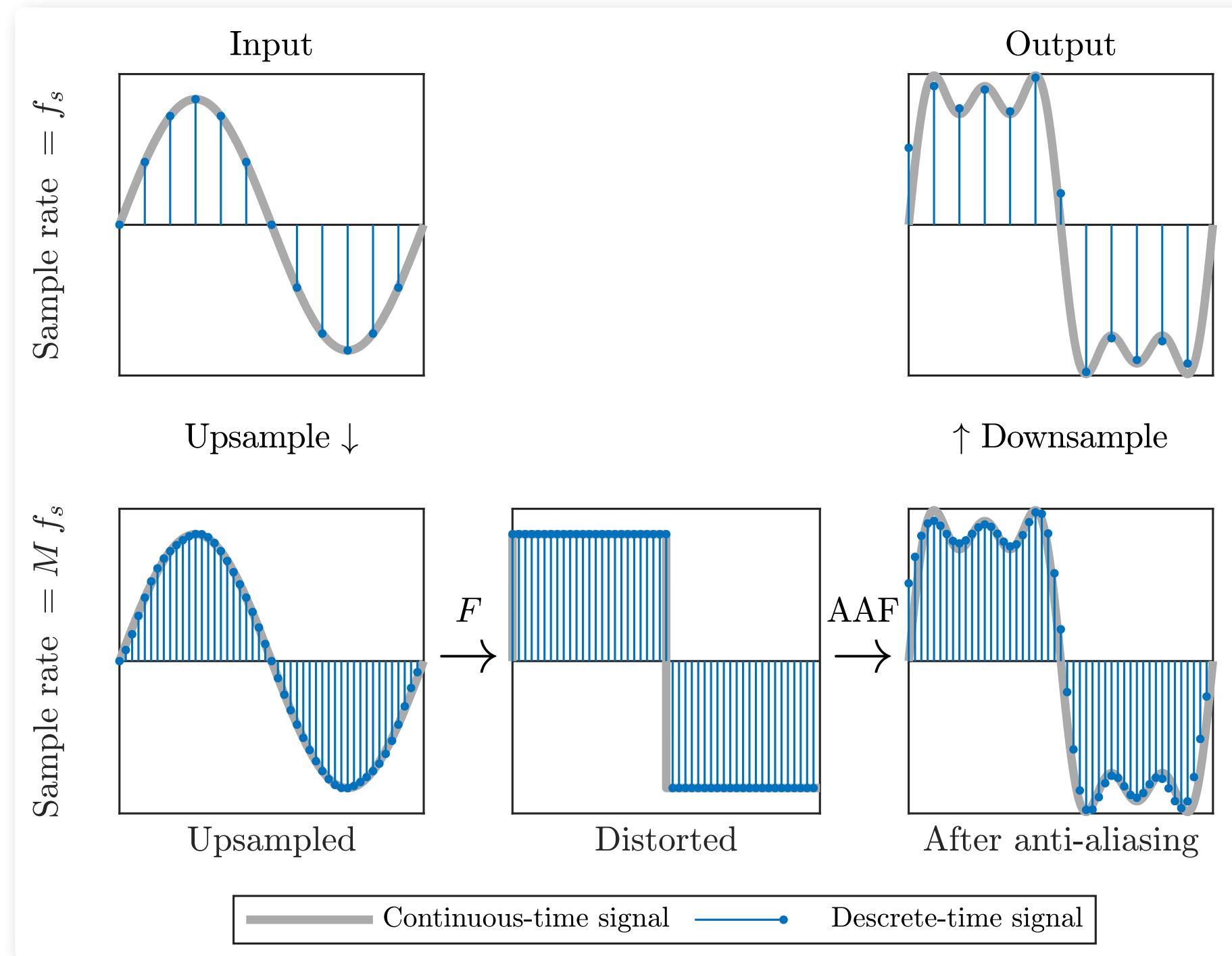


矩形波に対するアンチエイリアシング

- アンチエイリアシング：
サンプリングの前にナイキスト周波数 f_N よりも高い周波数の成分を取り除く (= **帯域幅を制限する**) ことでエイリアシングを防止
- 理想的に帯域制限された矩形波は、矩形波のフーリエ級数展開を有限の k で打ち切ったものに対応する

$$\frac{4}{\pi} \sum_{k=1}^K \frac{\sin(2\pi(2k+1)t)}{2k+1}$$

オーバーサンプリング



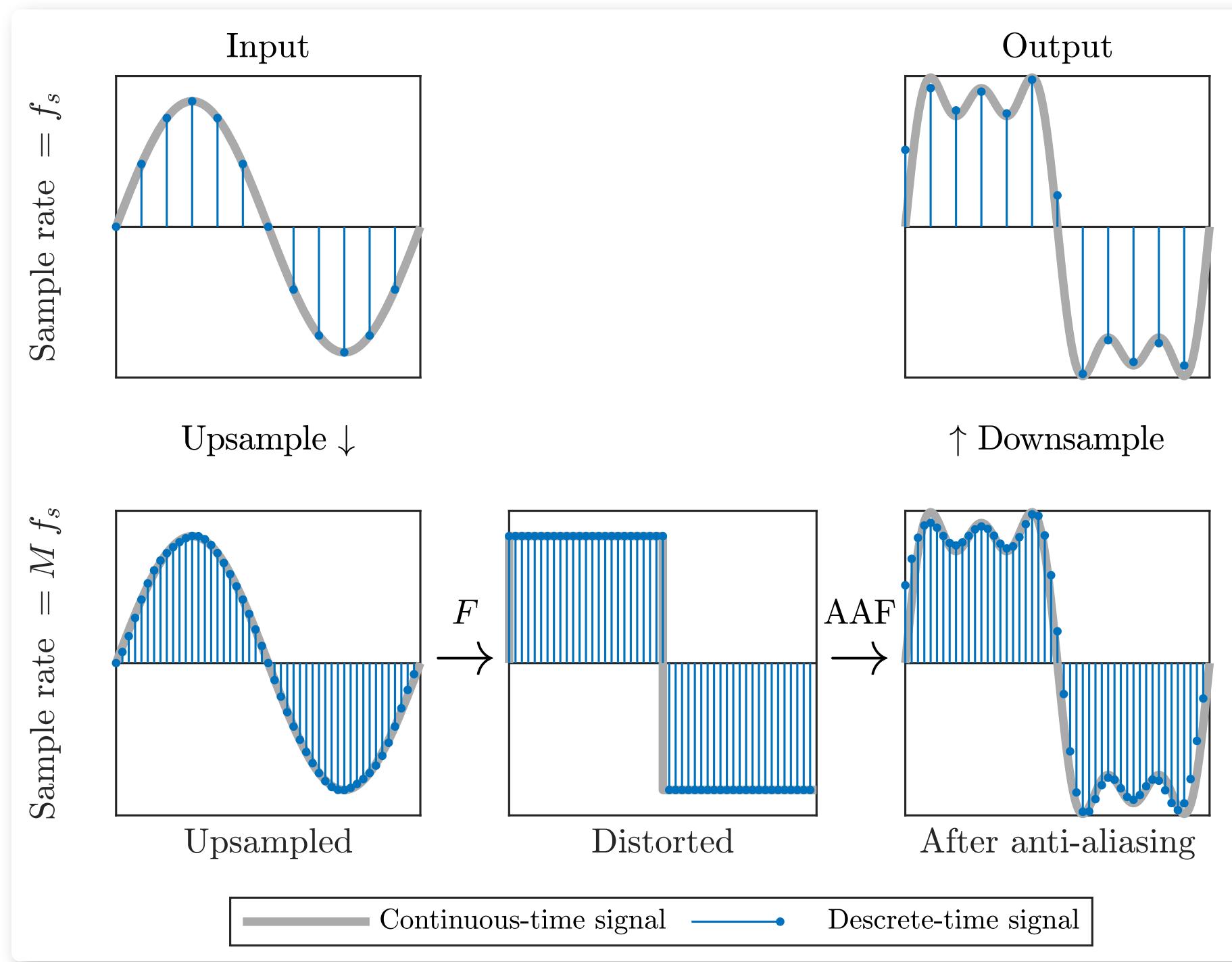
オーバーサンプリングを用いるディストーション

エイリアスノイズの発生を抑制するには
オーバーサンプリングが有効

1. サンプリング周波数を M 倍に引き上げる
2. ディストーションの関数 F を適用
3. アンチエイリアシングフィルタ (AAF) を適用し,
ナイキスト周波数 f_N 以下の成分を除去
4. サンプリング周波数をもとに戻す

► 帯域制限された矩形波に近い結果が得られる

オーバーサンプリング



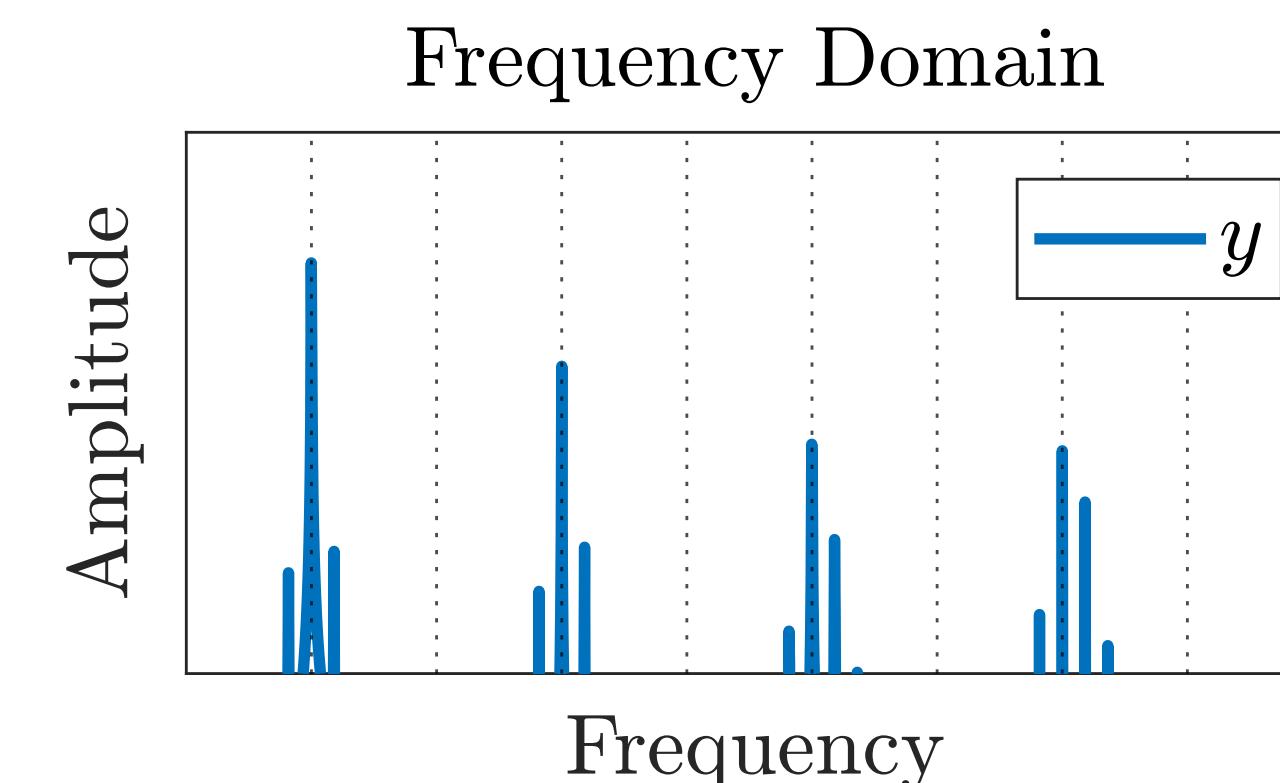
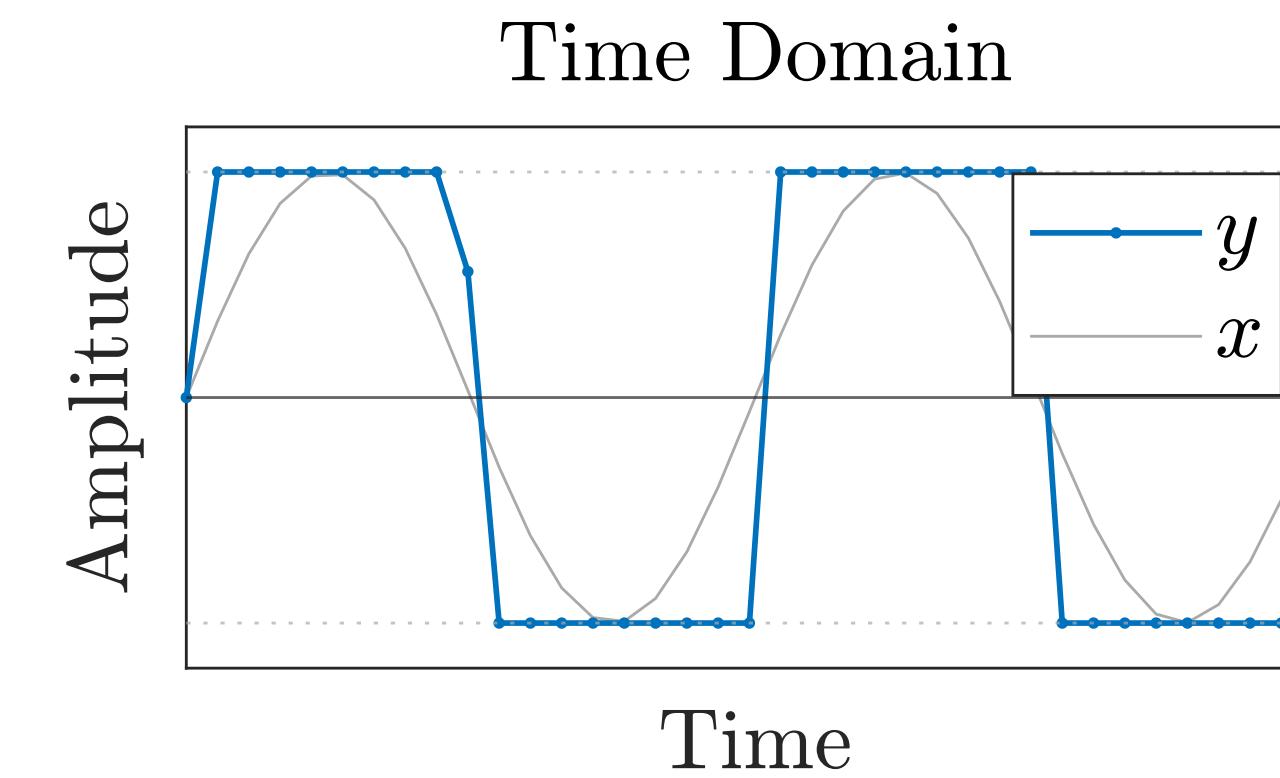
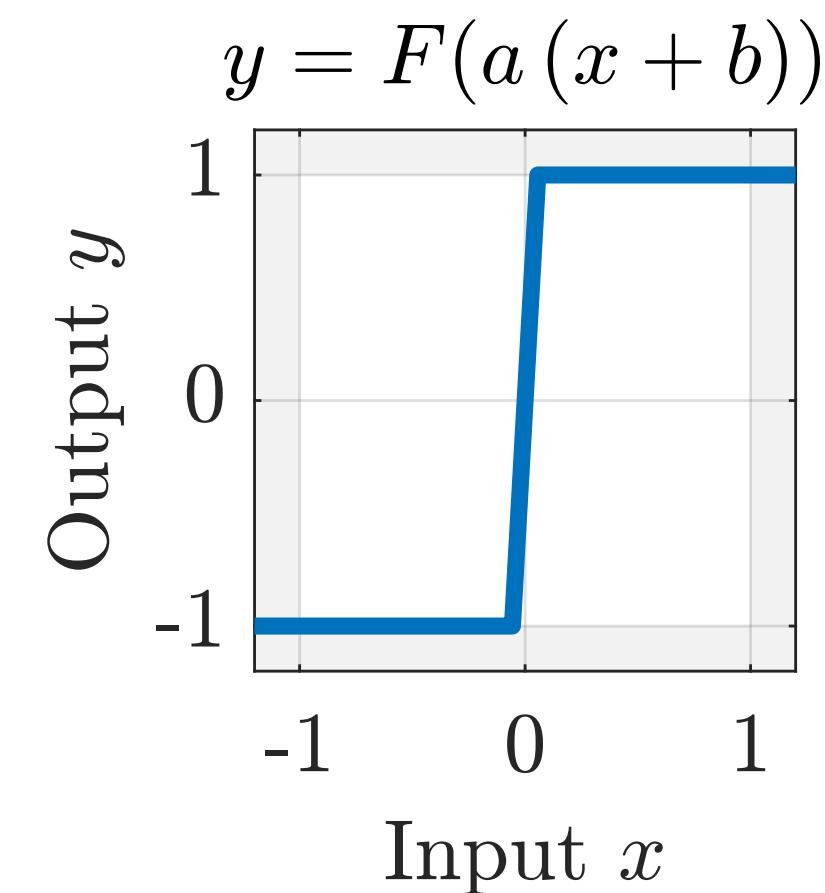
オーバーサンプリングを用いるディストーション

高音質な実装

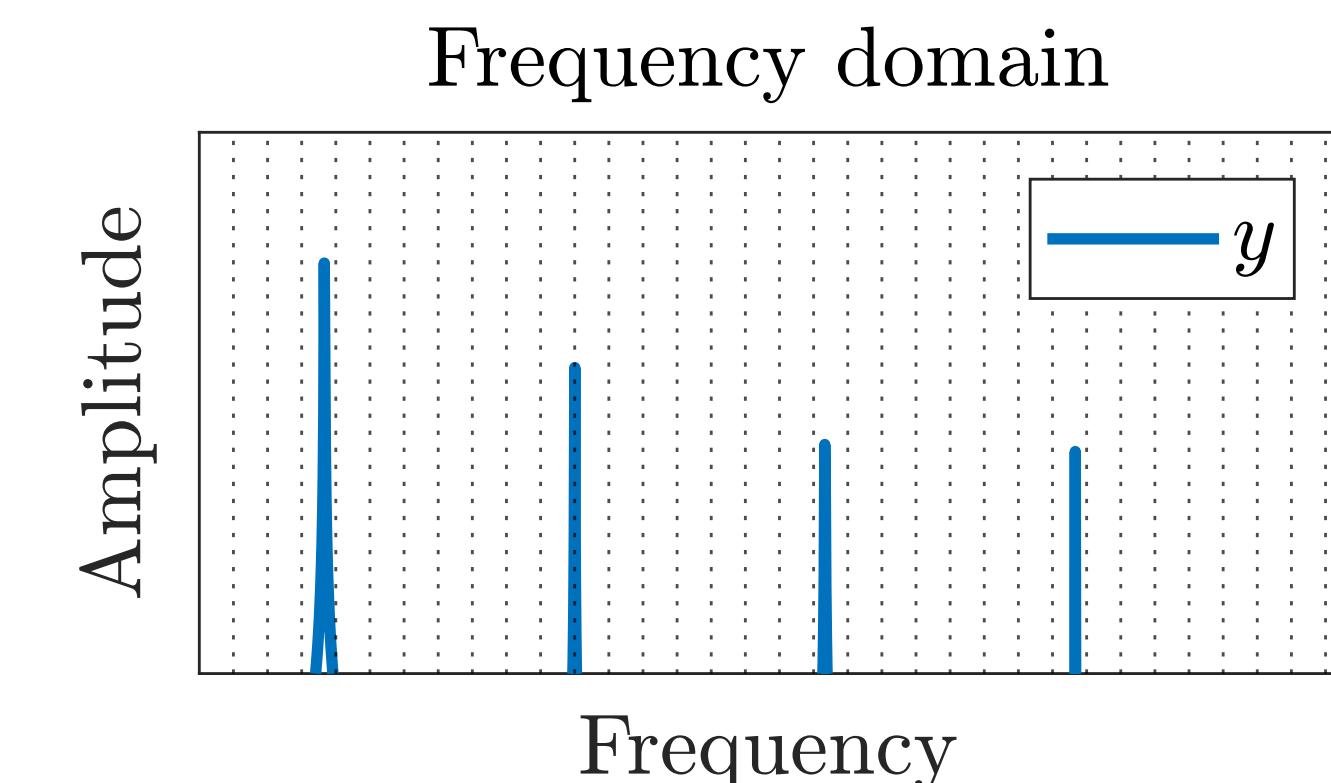
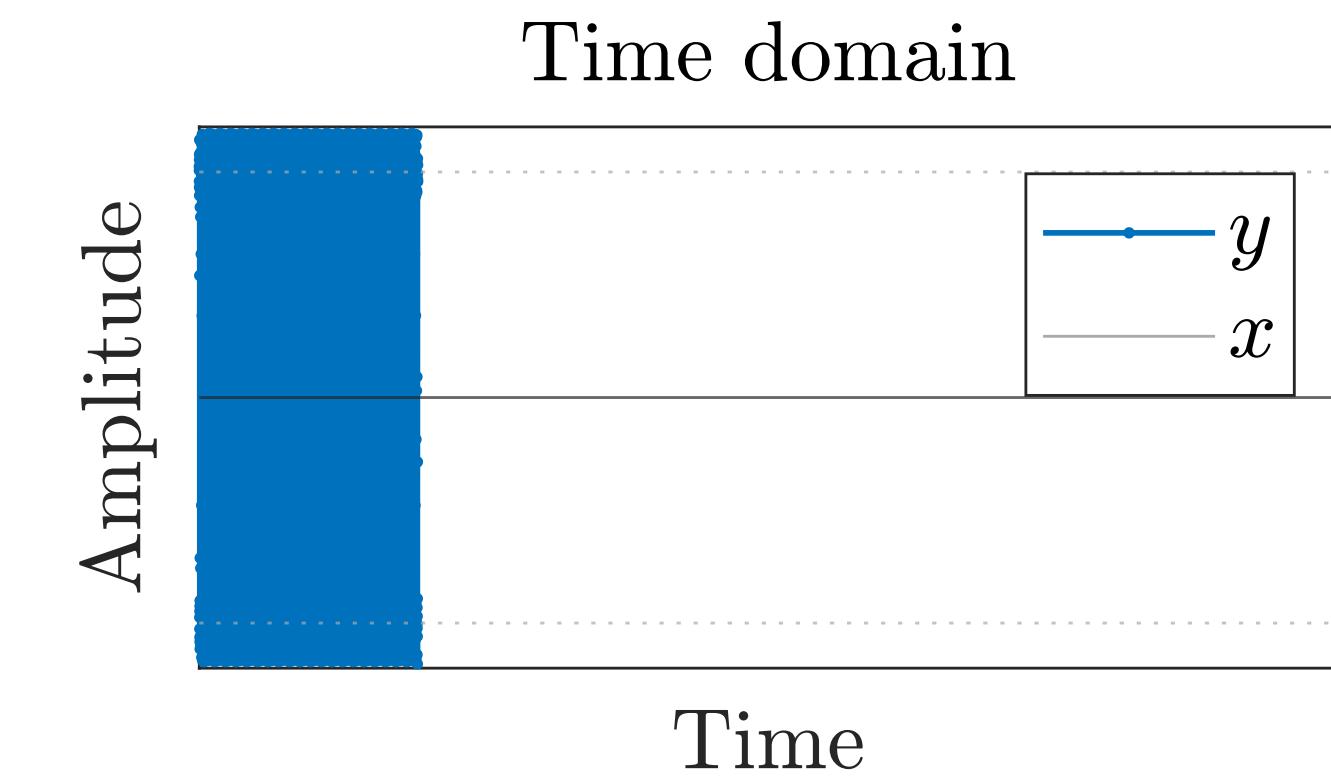
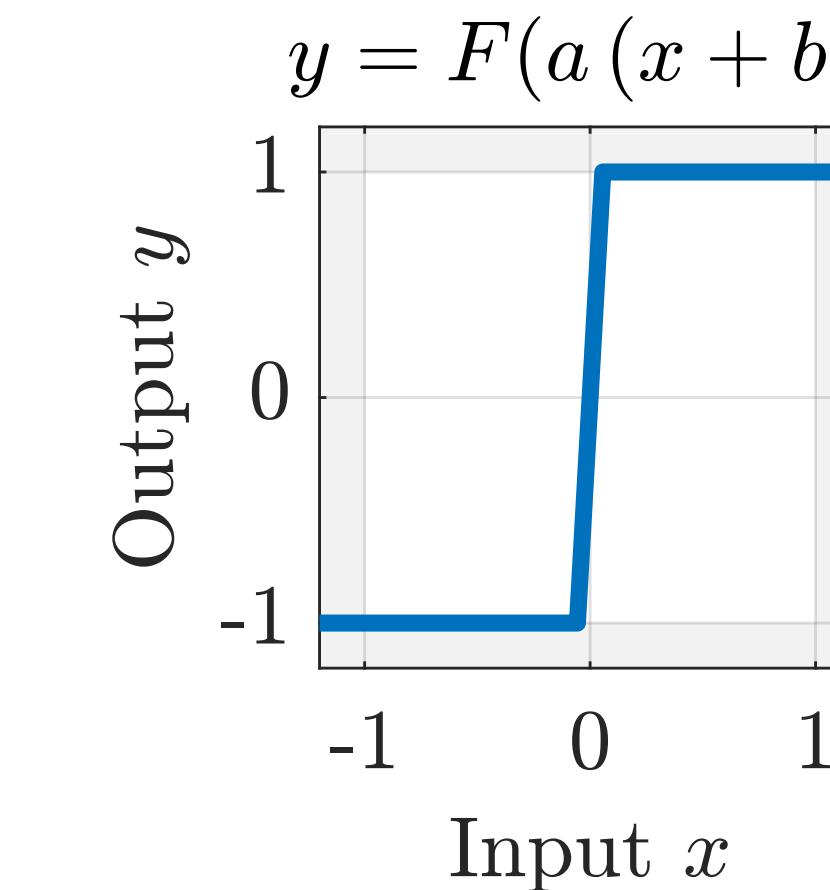
実装上は `interp` や `decimate` を利用
アンチエイリアシングフィルタの適用が実装されている

```
M = 8; % オーバーサンプリングの次数
x2 = interp(x,M); % アップサンプリング
y2 = f(a.* (x2 + b)); % ディストーションの適用
y = decimate(y2,M); % ダウンサンプリング
```

オーバーサンプリング



オーバーサンプリングなし



オーバーサンプリングあり

プラグインの実装

おわりに

まとめ

- ディストーションの実装
- サンプリング定理とエイリアシング
- オーバーサンプリングによるエイリアシングの抑制

Further Practice

- 関数 F をいじってより面白い歪みを探求する

役立つ資料

- やる夫で学ぶディジタル信号処理  : 信号処理全般