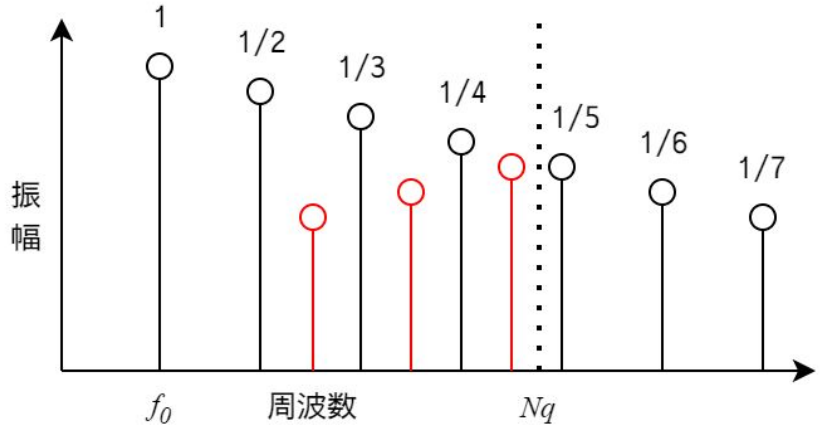
オシレーターの改良

オシレーター

- 教材アプリのオシレーターについてどう思いますか
  - 普段使っている同様のオシレーターと違いはありますか

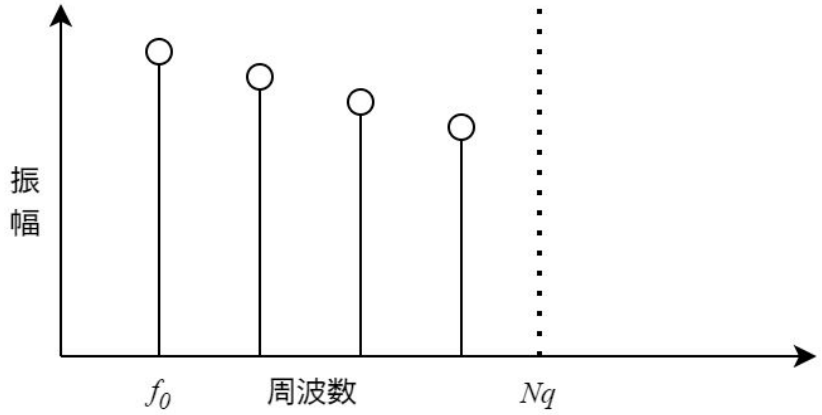
ディジタルオシレーター

- ディジタル領域で生成した信号には折り返しノイズが含まれる
  - ナイキスト周波数を超える成分が折り返されて発生する
  - アナログオシレーターでは発生しない



バーチャルアナログオシレーター

- ナイキスト周波数以下の成分のみを持つ信号を生成するオシレーター
  - Band-Limited Oscillator と呼ばれる
  - どうすればディジタル領域で実現できると思いますか



バーチャルアナログオシレーター

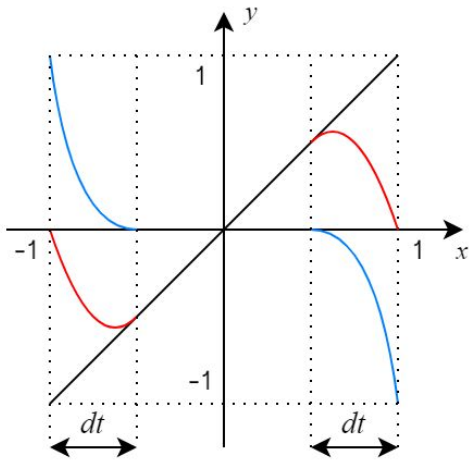
- 考えられる実現方法
  - 正弦波の加算合成
  - ダウンサンプリングの利用
    - 例：96 kHz で生成した鋸歯状波を 48kHz に変換
  - 軽量なアルゴリズムである程度の成果を出す

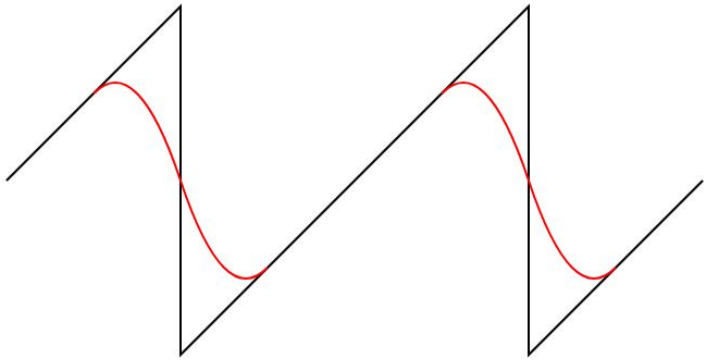
Poly BLEP

- Polynomial Band-Limited stEP function
  - 信号で生じる大きなギャップを丸くするアルゴリズム
    - 大きなギャップが折り返しノイズの主な原因になる
  - 鋸歯状波や矩形波に適用する
  - 今回は簡易な二次関数で実現する

簡易な二次関数を用いたPoly BLEP

- 黒 (鋸歯状波) + 青 (二次関数) = 赤 (帯域制限された鋸歯状波)
  1. 下のグラフから二次関数の式を導出
  2. 生成した鋸歯状波に導出した二次関数の値を足す





二次関数の導出

- 1 - dt < x <= 1 の二次関数：$y = -\left(\frac{x - 1 + dt}{dt}\right)^2$
  - x = 1 - dt でx軸に接して (1, -1) を通る関数
- -1 <= x < -1 + dt の二次関数：$y = \left(\frac{x + 1 - dt}{dt}\right)^2$
  - x = -1 + dt でx軸に接して (-1, 1) を通る関数
- dtの計算式：$dt = \frac{2 \times \mathrm{Frequency}}{\mathrm{SampleRate}}$
  - ただしプログラム中では dt = 2 × this.frequency とする