電子制御工学実験報告書

実験題目　：信号処理プログラミング

報告者　　：4 年 32 番　平田 蓮

提出日　　：2020 年 6 月 18 日

実験日　　：2020 年 5 月 21 日, 5 月 28 日, 6 月 4 日, 6 月 11

実験班　　：

共同実験者：

※指導教員記入欄

評価項目	配点	一次チェック ・　・	二次チェック ・　・
記載量	20		
図・表・グラフ	20		
見出し，ページ番号，その他体裁	10		
その他の減点	–		
合計	50		

コメント：

1　目的

アナログ信号をディジタルデータに変換し, ディジタル機器で処理するために必要な起訴事項について学習し, C 言語で基本的な信号処理プログラムを作成する. また音声フォーマットの一つである WAVE ファイルの構造を理解し, 音声データの入出力プログラムを作成する.

2　周期関数の生成と可視化

正弦波のように一定周期ごとに同じ波形が繰り返される関数を**周期関数**と呼ぶ. よく知られている周期関数として, のこぎり波などがある. 本節では周期関数に関する演習を行う.

■**演習 1-1**　任意の孤度 r を区間 $[0:2\pi]$ に変換する関数 `rad` を作成せよ.

作成した関数をソースコード 1 に示す.

ソースコード 1　rad.c

```
1 double rad(double r) {
2     if (r >= 0) {
3         return fmod(r, 2 * PI);
4     } else {
5         return 2 * PI - fmod(-r, 2 * PI);
6     }
7 }
```

r が正のときは 2π との剰余を取る. r が負のときは, r を正数に変換し, 2π と剰余をとったものを 2π から引くことで変換している.

図 1 に横軸に r, 縦軸に `rad(r)` を取ったグラフを示す. この図から, 変換がうまく行われていることがわかる.

■**演習 1-2**　任意の孤度 r に対して, のこぎり波の振幅値を求める関数 `saw` を作成せよ.

作成した関数をソースコード 2 に示す.

ソースコード 2　saw.c

```
1 double saw(double r) {
2     return 1 - rad(r) / PI;
3 }
```

まず与えられた孤度 r を演習 1-1 で実装した `rad` を使い $[0:2\pi]$ の範囲に変換する.

r が 2π の倍数の場合に 1 を取り, そこから傾き $-\frac{1}{\pi}$ の形を繰り返すので, 上記のように実装することができた.

図 2 にグラフを示す. うまくのこぎり波が現れていることがみてとれる.

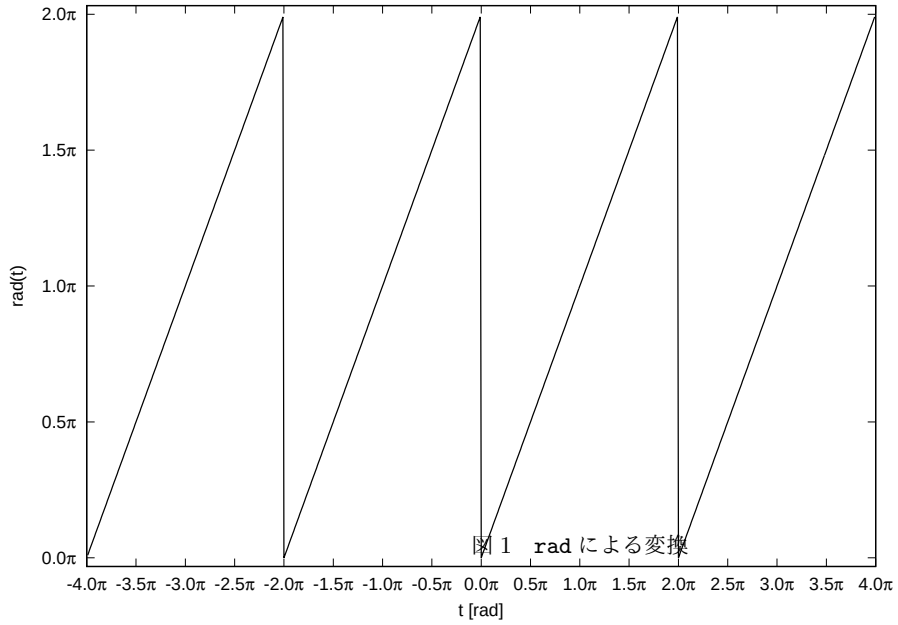

図 1　`rad` による変換

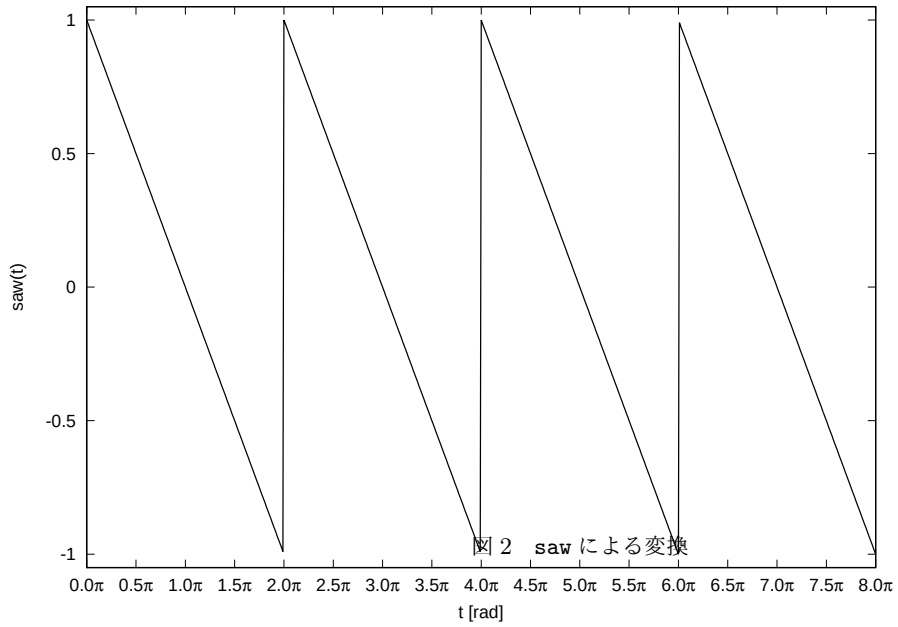

図 2　`saw` による変換

3　サンプリング

アナログ信号波形をディジタルデータに変換する手法に**サンプリング**がある．本節では演習問題を通して実際にサンプリングの様子を観察する．

■**演習 2-1**　リスト 2 のコードを解析し, 完成させよ. 振幅 100, 周波数 4Hz のデータを出力リダイレクションを使って sin100f4.csv に出力せよ. この sin100f4.csv を gnuplot でグラフ化し, サンプリングの効果を確認せよ.

まず, 完成させたリスト 2 のコード, sin10af1.c を出力部分を抜粋して掲載する.

ソースコード 3　sin10af1.c

```
1  for (t = 0; t <= TEND; t += DT) {
2      r = 2 * PI * frq * t / 1000.0;
3      vin = amp * sin(r) + BIAS;
4      if (vin > 255) {
5          vout = 255;
6      } else if (vin < 0) {
7          vout = 0;
8      } else {
9          vout = vin;
10     }
11     printf("%4f, %4d\n", t / 1000.0, vout);
12 }
```

4 行目から 10 行目に渡って, クリッピングという処理を施してある. クリッピングをすることで, 出力用の変数 (今回は 8bit の char 型) に収まらない値を切り捨て, 波形を維持することができる.

図 3 にサンプリングしたデータを示す. 実際に正弦波をサンプリングできていることがわかる.

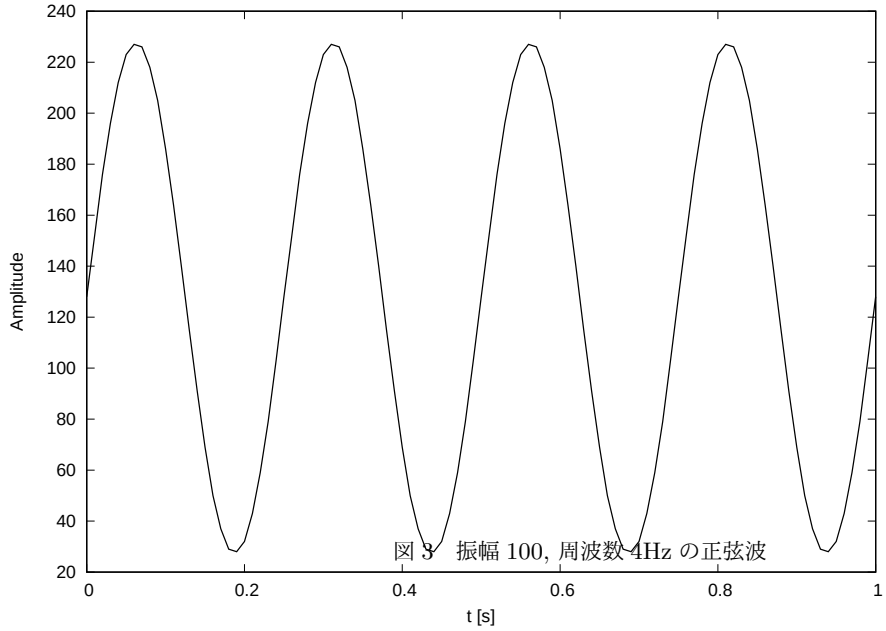

図 3　振幅 100, 周波数 4Hz の正弦波

■**演習 2-2**　振幅 150, 周波数 4Hz のデータ sin150f4.csv を生成しその波形を確認せよ. 波形に不具合があればその原因を考えて不具合を軽減するような修正を行え.

今回は振幅が 150 なので, char 型に収まらない数値をサンプリングすることになる. そこで, 演習 2-1 で実装したクリッピングが働く.

図 4, **??**にクリッピングを施した前と後の波形を示す. この図からクリッピングの効果がみて取れる. クリッピングをしていない波形では, オーバーフローした値が波形の逆側に飛んでしまい, 波形が崩れているが, クリッピングを施した方ではオーバーフローした値が切り捨てられ, 波形が維持できている.

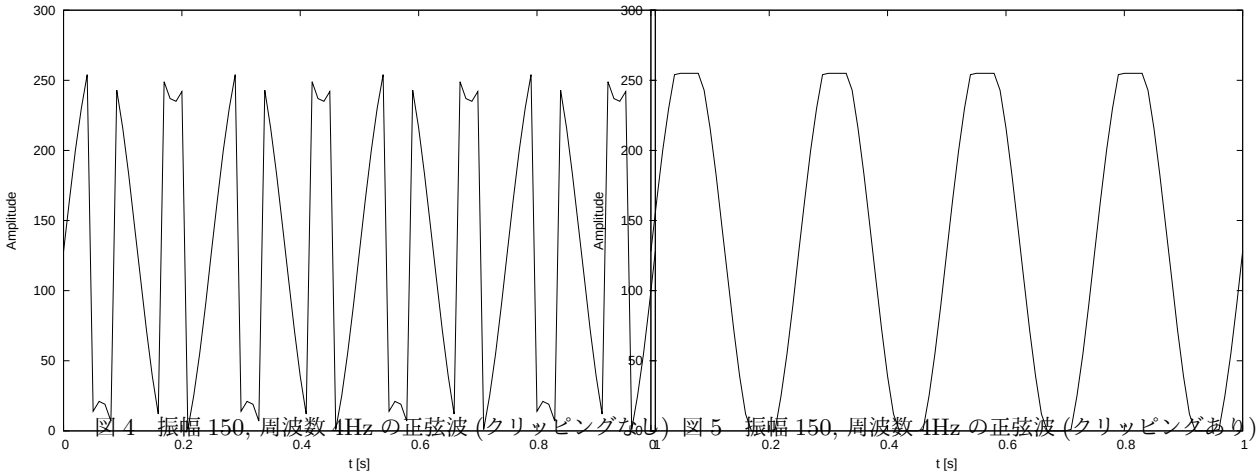

図 4　振幅 150, 周波数 4Hz の正弦波 (クリッピングなし)　図 5　振幅 150, 周波数 4Hz の正弦波 (クリッピングあり)

参考文献

[1] Electro-ワンチップマイコン http://laboratory.sub.jp/ele/13.html/