第2章

四則計算で音が出せる！

それでは，さっそくDSP（ディジタル信号処理）シミュレータを活用していきましょう．本章では，F君，U君の生徒二人と先生との会話を通して，ディジタル信号処理のようすを感覚的につかんでもらいます．“読んでわかる，ためになる”内容をふんだんに取り入れ，わかりやすく解説を進めていきます（なお，F君は本書の著者の一人である藤村諒，U君は内田瑛一かも？）．

2.1　数値の並びがディジタル信号になる?!

F君とU君の二人の生徒を前に，先生はチョークで黒板に次のように書き始めました．

『ディジタル信号は，数値を並べたものである』

（先生）黒板に書いた文章の意味，お二人さん，なんとなくわかるかな？

（F君）はい．ディジタル信号の**ディジタル**という言葉は，ディジタル時計で時刻を表示する数字のことかな，という気がします．

（U君）僕もそう思います．でも，数値と信号との関わりあいがよくわかりません．先生，目で見て確かめてみたいのですが，どうすればいいんですか．

（先生）よし，わかった．君たちのパソコンにはすでにDSPシミュレータが入っているから，そのソフトウェアを動かして信号波形を確認してみよう．**図1.19**に示す起動画面が表示されていない人は，デスクトップ上のScilabのアイコン（  ）をダブル・クリックしてScilabを起動してから，DSPシミュレータを立ち上げてください．その際，**図1.17**の『⑩ディレクトリの変更』と**図1.18**の『⑪ディジタル信号処理ライブラリの選択，実行』は忘れないようにしてください．うっかり忘れちゃ，絶対にだめだよ．DSPシミュレータの機能を利用するときに，隠れた力を発揮するようになっているからね．

（F君，U君）DSPシミュレータが起動しました．入力はどうすればいいんですか．

（先生）起動画面の最後に矢印みたいな記号，すなわち，

```
-->
```

があるよね．このプロンプト記号に続けて，**プログラム2.1**の①，②，③の3行をスペルをミスしないように入力してもらいたいんだ．1行分を入力したら，必ず Enterキー（改行キー，あるいは ↵ と表記）を押すことを忘れないようにね．なお，③の「%pi」は円周率のπを表します．そのほか，虚数単位（$j=\sqrt{-1}$）は「%i」，自然対数$\log_e(x)$の底$e=2.71828\cdots$（ネピア数）は「%e」と表記します．

［プログラム2.1］

```
--> f = 3 ; sfrq = 20 ; T= 1/sfrq ; tmax = 1 ;↵   ..... ①
--> k = 0 : 1 : (tmax/T) ;↵                        ..... ②
--> y = cos(2 * %pi * f * k * T) ;↵                ..... ③
```

（F君）　ミスなく入力できました．簡単ですね．でも，何にも見えませんよ，変だなぁ．

（U君）　まだ何か，入力しないといけないんですか．早く，教えてください．

（先生）　そんなにあせらなくていいよ．もし入力ミスしたときは，矢印カーソル・キーの ↑ あるいは ↓ を押すと，すべての文字列を再入力しなくても，以前に入力した文字列を再利用して楽に入力できるからね．

それでは，**プログラム2.2**のplot命令を入力してみてください．plot命令は，信号波形を観測するために使用するオシロスコープのようなものです．さて，さて，お二人さん，どうなりましたか？

［プログラム2.2］

```
-->  plot2d3(k, y, 20) ;↵   ..... ④
-->  plot2d(k, y, -5) ;↵    ..... ⑤
```

（F君）　おおっと，とびとびの縦棒が現れました．〔**図2.1**には，画面には縦棒を滑らかにつないだ信号（時間的に連続で，アナログ信号という）も点線で示してあります．〕

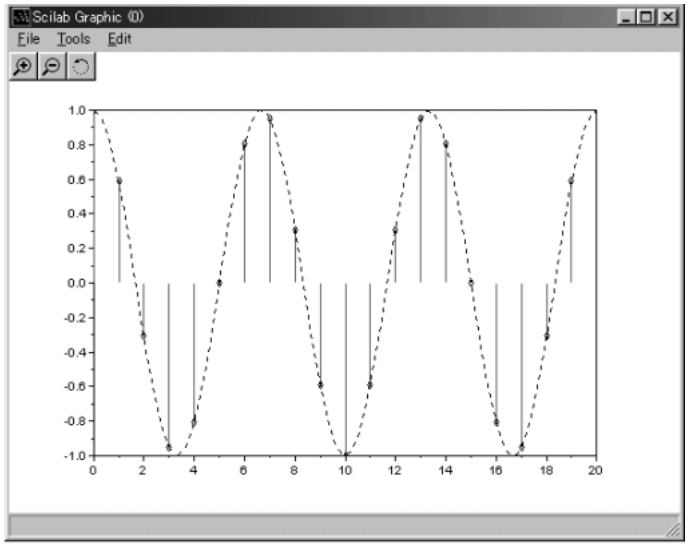

図2.1　ディジタル信号のグラフ表示

（U君）やったぁー．僕も同じ結果になりました．ところで，縦棒がディジタル信号なのかな？

（先生）U君の言うとおり，縦棒がディジタル信号で，横方向は時間軸としてサンプル時刻（変数kで整数）を表し，**離散時間信号** $y[k]$（$k=0, 1, 2, \cdots, 20$）と表します．“とびとびに縦棒がある”――――**離散化**（discrete）されているのは，信号の値ではなく時間軸であることに注意してください．**図2.1**の点線で示すcos波形（**アナログ信号**に相当），すなわち，

$$y(t)=\cos(2\pi\times 3\times t) \quad \cdots\cdots (2.1)$$

を一定の時間間隔 $T=1/20$［秒］ごとに1秒間（$=t_{\max}$）だけ取り出した信号として，式(2.1)で$t=kT$を代入すれば，

$$y[k]=y(kT)=\cos\left(2\pi\times 3\times k\times\frac{1}{20}\right) \quad \cdots\cdots (2.2)$$

となります（サンプリング処理，あるいはサンプリング操作という）．このように，アナログ信号から一定の時間間隔 T［秒］で離散化したディジタル信号は，時間軸上で「とびとび」にしか定義されていないのです．このときのディジタル信号のサンプル点（整数k）は，0～20（$=t_{\max}/T$）となります．

（F君）先生，一つお聞きしたいことがあります．それは，先ほど先生が黒板に書かれたことですが，**ディジタル信号は数値**だという証拠を見せてほしいんです．

（先生）それじゃ，証拠をお見せしよう．先ほど，**プログラム2.2**のplot命令（④，⑤）を入力すると縦棒が表示されたけれど，実は変数yという箱（配列）の中に格納されている数値をグラフ化してディジタル信号の波形として見たわけです．つまり，変数yという箱の中身がディジタル信号そのものになるので，変数yの箱の中身を調べてみればよいのです（簡単でしょ）．パソコンの画面に数値を表示させて確認してみることにしましょう．それには，

［プログラム2.3］

```
--> y ↵                                    ..... ⑥
```

とただ1文字‘y’と入力して Enter キーを押すだけで，変数yの箱の中身を画面に表示することができるのです．

（U君）変数yの箱の中身が見えたぞー，21個の数値が表示されてる（**図2.2**）．ホントだぁー，ディジタル信号は数値に順番をつけて横に並べたもの（行ベクトル），いわゆる順序集合なんですね．よくわかりました．

（F君）そうか，式(2.1)を**プログラム2.1**の③のように計算プログラムとして入力したわけだから，当たり前なんですね．先生，**図2.1**のグラフをながめていて気がついたんですが，山と谷がそれぞれ3個ずつありますね．

（先生）F君，いいところに目をつけたね．山と谷はそれぞれ3個ずつある．これは，周波数と呼ばれるもので，**プログラム2.1**の①の$f=3$が相当します．周波数の単位はHz（ヘルツ）です．

（U君）あれっ，ちょっと待ってください．周波数と言えば，1秒間あたりの振動数ではなかったでしたっけ？

（先生）そのとおりです．変数kの0から20までが実は1秒間を表し，1秒を20等分して1/20秒ごと

```
-->y
 y  =

         column 1 to 5

!   1.    0.5877853  - 0.3090170  - 0.9510565  - 0.8090170 !

         column  6 to 10

! - 1.837D-16    0.8090170    0.9510565    0.3090170  - 0.5877853 !

         column 11 to 15

! - 1.  - 0.5877853    0.3090170    0.9510565    0.8090170 !

         column 16 to 20

! - 1.225D-15  - 0.8090170  - 0.9510565  - 0.3090170    0.5877853 !

         column 21

!   1. !
```

図2.2　ディジタル信号 y の中身を行ベクトルで表示した例

```
-->y'
 ans  =

!   1.         !
!   0.5877853 !
! - 0.3090170 !
! - 0.9510565 !
! - 0.8090170 !
! - 1.837D-16 !
!   0.8090170 !
!   0.9510565 !
!   0.3090170 !
! - 0.5877853 !
! - 1.         !
! - 0.5877853 !
!   0.3090170 !
!   0.9510565 !
!   0.8090170 !
! - 1.225D-15 !
! - 0.8090170 !
! - 0.9510565 !
! - 0.3090170 !
!   0.5877853 !
!   1.         !
```

図2.3　ディジタル信号 y の中身を列ベクトルで表示した例

のディジタル信号が得られています．また，1秒間に山と谷が3組あるので，振動数は3回，すなわち周波数は3 Hzとなります．周期は周波数の逆数に等しく，1/3秒ですね．

（F君）それで，0から20までの21個のディジタル信号の数値が，**図2.2**のように表示されるんですね．だったら，5秒間のディジタル信号のときは，**プログラム2.1**の②のtmaxを5にすればいいのか．少しずつディジタル信号がわかってきた，見えてきました．

（先生）すばらしいねぇ．F君，やるじゃないか．ちなみに，変数sfrqは**サンプリング周波数**（あるいは**標本化周波数**）とよばれ，1秒間あたりのディジタル信号の個数に相当します．また，その逆数（1/sfrq）を表す変数Tは**サンプリング間隔**（あるいは**標本化間隔**）といい，単位は［秒］です．詳しくは，後で説明します．なお，一定の時間間隔1/20秒はプログラム中の変数Tでサンプリング間隔，分母の20は変数sfrqでサンプリング周波数（1秒間に含まれるディジタル信号，すなわち縦棒の総個数に相当）とよばれます．

ところで，**図2.2**の信号表示は少しわかりにくいかもしれないので，

［プログラム2.4］

```
-->  y' ↵                                                    ..... ⑦
```

と，信号変数の後ろにアポストロフィ「'」を付けてみてください．見やすくなったでしょう（**図2.3**）．ディジタル信号が縦に順番に並んで，いわゆる列ベクトル（縦ベクトル）の形式で表示したんですね．

（U君）ハーイ，わかりました．あと，①～⑤のDSPプログラムの書き方，その処理内容の解説を簡単でいいですからお願いします．

（先生）承知しました．おさらいしながら，DSPシミュレータを利用するための基本的な機能や文法的なことをまとめておきたいと思います．

● DSPプログラミングのまとめ（その1）

DSPシミュレータを起動すると，対話的にコマンド（命令）を実行するウィンドウ（**図1.19**）が開きます．ウィンドウ上に表示されている記号-->に続けて文字列を入力し，Enterキーを押すことで対話的にコマンドを実行することができます．なお，**大文字と小文字は区別される**ので，細心の注意を払って文字入力してください．

（**処理例2.1**） 変数aに数値7.3を代入します．

```
--> a = 7.3 ↵
  a =
     7.3
```

（**処理例2.2**） 変数aに数値7.3を代入して実行した後に，結果を出力させたくないときは，行末にセミコロン「;」を付けます．結果は表示されず，次の命令の入力待ち状態になります．

```
--> a = 7.3 ; ↵
-->
```

このとき，変数aに格納された値を確認したいときは，変数名を入力します．

```
--> a ↵
  a =
     7.3
```

（**処理例2.3**） 変数名には，大文字と小文字を区別して用いることができます．以下のように，変数名としてaとAを混在させることもできます．

```
--> a = 7.3 ; ↵
--> A = 2.5 ; ↵
--> a + A ↵
  ans =
      9.8
```

```
--> wa = a + A ↵
  wa =
      9.8
```

(**処理例2.4**)　1行に複数の命令を書くときには，カンマ「,」あるいはセミコロン「;」で区切って表記します．ただし，カンマのときは結果が出力されますが，セミコロンのときは出力されません．

```
--> a = 3, b = 5, c = a + b ↵
  a =
     3.
  b =
     5.
  c =
     8.
--> a = 3 ; b = 5, c = a + b; ↵
  b =
     5.
```

(**処理例2.5**)　//から後ろはコメントとみなされるので，そこに書いてあることは実行結果に影響しません．なお，日本語文字は文字化けするので，英数字と記号の入力が基本になります．

```
--> a = 7    // real constant ↵
  a =
     7.
```

(**処理例2.6**)　行末にピリオドを . . . のように3個続けて書くと，次の行が継続しているものとみなし，連続した1行として記述できます．

```
--> wa = 1 + 2 + 3 + 4 + 5 + 6 + ... ↵
--> 7 + 8 + 9 + 10 ↵
   wa =
      55.
```

(**処理例2.7**)　繰り返し計算のための書き方です．いま，変数aの値を，たとえば0(開始値)から10(終了値)まで，2ずつ(ステップ値)増加させたいときは，開始値，ステップ値，終了値の順にコロン「:」で区切って次のように記述します．

```
--> a = 0 : 2 : 10 ↵
  a =
    ! 0.    2.    4.    6.    8.    10. !
```

また，変数bを20(開始値)から−10(終了値)まで，5ずつ減らしたいときは，ステップ値を(−5)に設定することになります．

```
--> b = 20 : -5 : -10; ↵
-->
```

ここで，実行命令の文末にセミコロン「;」を付けたので，変数bの値が表示されません．変数bの内

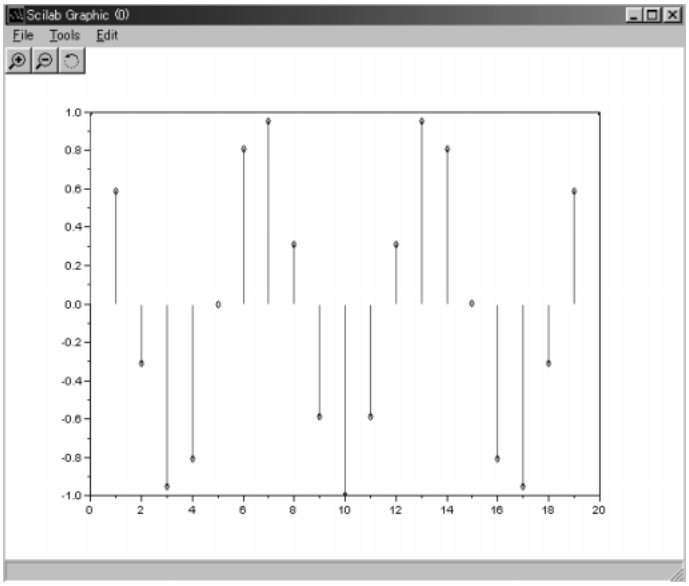

図2.4　ディジタル信号のグラフ表示
（plot2d3命令，plot2d命令による）

容を確認するには，変数名を入力すればよいのです(**処理例2.2**を参照)．実行結果を以下に示します．

```
--> b = 20 : -5 : -10 ; ↵
--> b ↵
  b =
     ! 20.   15.   10.   5.   0.  -5.  -10. !
```

なお，ステップ値が1の場合はステップ値を省略して開始値，終了値の順にコロン「:」で区切って簡略形で記述できます．

(**処理例2.8**)　ディジタル信号を時間離散の赤い縦棒(棒線の先端に白抜きのひし形「◇」を付加)として，**図2.1**のようにグラフ表示するための表記は，

```
--> plot2d3(k, y, 20) ; ↵
--> plot2d(k, y, -5) ; ↵
```

であり，グラフ表示は**図2.4**のようになります(**図2.1**の点線は，別途書き加えてあることに注意)．もちろん，1行にまとめて，

```
--> plot2d3(k, y, 20) ; plot2d(k, y, -5) ; ↵
```

と入力してもOKです．ただし，**プログラム2.1**の①～③がすでに実行されているものとし，繰り返し変数kに対するディジタル信号の変数yをグラフ表示する場合を想定しています．

このように，plot2d3命令はディジタル信号を縦棒で，plot2d命令はマーカー(たとえば，○，◇，＊，×など)で表すためのコマンドということになり，重ねて描かれます．いずれのグラフ表示命令も小カッコ(　)の中の引数パラメータ(これ以後，単にパラメータと記す)として，左から「繰り返し変数」，「グラフ表示する変数」，「そのほかのオプション」の順で表記します．