BASIC プログラミング演習

山形大学理学部物質生命化学科

平成 20 年 4 月 10 日

目 次

0 N88 互換 Basic の使い方

0.0 N88 互換 Basic の使い方

図 0.1: N88Basic のショートカットアイコン

N88Basic を始めるには、デスクトップにある図 0.1 のアイコンをクリックする。

あるいは、画面左下の「スタート」ボタンをクリックし、「プログラム」→「N88 互換 BASIC for Windows95」→「N88 互換 BASIC for Windows95」を選ぶ（図 0.2）。

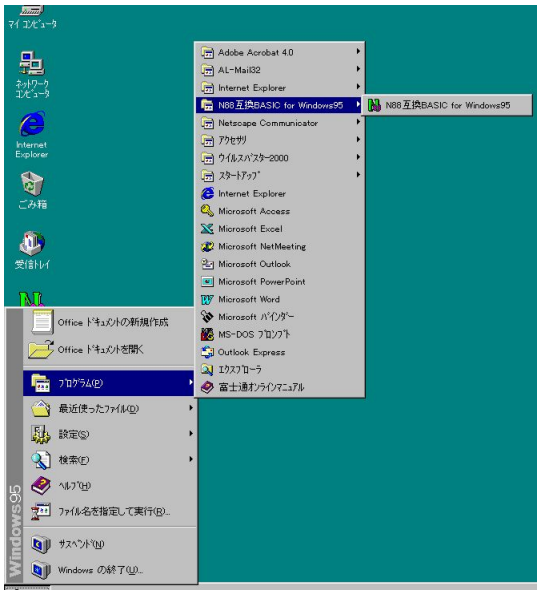

図 0.2: メニューから N88 互換 BASIC を選ぶ

N88 互換 BASIC が始まると、図 0.3 に示したように、白いウィンドウが前面に、黒いウィンドウ（実行画面）がその背後に表示される。プログラムの保存や呼び出し、編集は、白いウィンドウを前面に持ってきた状態で行う。黒いウィンドウには、プログラムの実行結果が表示される。

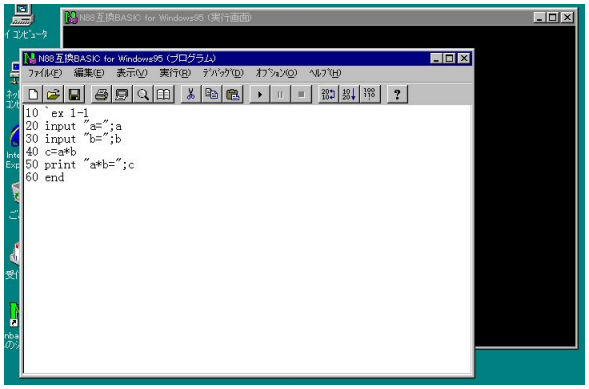

図 0.3: プログラムを入力する

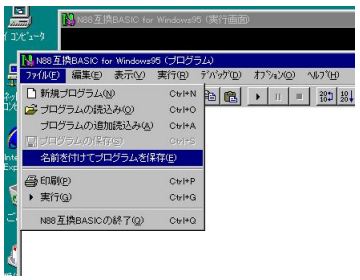

図 0.4: プログラムを保存する

プログラムを入力し終わったら、実行する前に保存する。保存するには、「ファイル (F)」メニューから「名前をつけてプログラムを保存」を選ぶ。BASIC のファイルは、ファイルの最後が拡張子「.bas」でなければならない。ファイル名の入力は大文字でも小文字でもかまわないが、適宜大文字に変換されるので、綴りが同じで大文字・小文字のパターンだけが違う名前をつけないように注意すること。

図 0.5: プログラムの保存場所を選び、名前をつける

プログラムはどこにでも保存できるが、混乱を防ぐために、C:ドライブの My Documents フォルダにまとめるか、デスクトップにフォルダを作ってその中に入れるようにすること。

図 0.6: プログラムを実行する

プログラムを保存したら、メニュー「実行 (R)」→「開始 (G)」を選ぶ。正しくプログラムが入力されていれば、背後にあった黒い画面が手前に出てきて、結果を表示する。例に示したプログラムでは、実行の時にキーボードから数値を入れるように作ってあるので、図 0.7 のように、数値の入力を待つ状態になる。

図 0.7: 数値を入力

適当な数字を入れて「enter」または「return」キーを押すと、結果が表示されて（図 0.8）プログラムが終了する。

図 0.8: 実行結果の表示

プログラムを変えて何度も実行する場合、前の結果に続いて表示されるため、画面が見づらくなることがある。実行画面は、「表示 (V)」→「実行画面クリア」を選ぶことで、内容を消すことができる。

図 0.9: 実行画面を消去する

プログラムに間違いがあった場合には、実行の途中で図 0.10 のようなエラーが出て止まってしまう。どこが違っているか行番号が出るので、その行を修

正し、保存した後、再度実行する。こまめに保存するように心がけること。

図 0.10: 実行エラー

新しくプログラムを作り始める場合は「ファイル」メニューから「新規プログラム」を選ぶ。プログラム編集用の白いウィンドウの内容がすべて消去され、新しいプログラムを入力することができる。

既に作ったプログラムを読み込んで実行したい時は、「ファイル」メニューから「プログラムの読み込み」を選ぶ。

図 0.11: プログラムを読み込む

プログラムファイルを保存してあるフォルダに移動し、ファイルを選んで「開く」をクリックする。

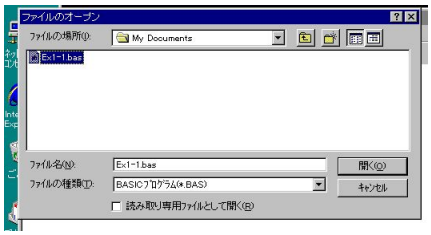

図 0.12: 読み込むプログラムファイルを選ぶ

プログラムの中に無限ループがあると、プログラムが終了しなくなる。この時は、メニューから「実行 (R)」→「終了 (H)」を選ぶか、メニューの下の「■」ボタンをクリックする。

図 0.13: プログラムの強制終了

実行の途中で、マウスカーソルが砂時計になったまま止まってしまって、BASIC が全く動かなくなることがある。この場合は、画面左下の「スタート」をクリックし、「Windows の終了」を選ぶ。

図 0.14: 終了メニュー

「アプリケーションを終了し、Windows にログオンし直す」を選ぶと、動かなくなった BASIC を強制的に終了させてよいか訊いてくるので、終了を選ぶ。このあと Windows にログオンし、再度 N88 互換 BASIC を動かせば、作業を続けることができる。ただし、保存していなかったプログラムは消えてしまう。

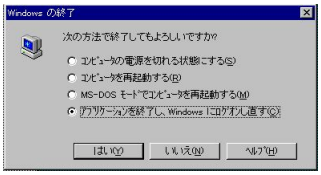

図 0.15: 終了の方法

これらの操作をしても N88BASIC が終了できなかったり止まってしまったりする場合は、「コンピュータを再起動する」を選ぶ。それでもだめな時は、本体のリセットスイッチを押すか、いったん電源を切ってもう一度最初からやり直す。

0.1 その他

「ヘルプ」→「ヘルプの表示」を使うと、N88 互換 BASIC のマニュアルを読むことができる。わからないことがあったらまずマニュアルを読むこと。

1　簡単な計算と結果の表示

例 1.1 (簡単な計算) a と b の値を入力すると $a \times b$ を計算し結果を表示するプログラム

プログラムとして入力するのは、左側の部分です。行番号も含めて入力してください。行番号は 10 刻みにつけておくと、途中に追加する時に楽です。また、行番号をつけ直すこともできます。

`10 'ex 1-1`	先頭に' をつけると、メモ書きできる
`20 input "a=";a`	a=?と表示したあと、変数a に代入する数値をきいてくる
`30 input "b=";b`	b=?と表示したあと、変数b に代入する数値をきいてくる
`40 c=a*b`	$a \times b$ の計算
`50 print "a*b=";c`	画面に結果を表示（たとえばa*b=60）
`60 end`	プログラムの最後はend と書く

print

変数の内容や文字を画面に表示する

`print a`	変数a が覚えている数を画面に表示
`print "hello!!"`	画面にhello!!という文字を表示
`print "a=";a`	a=という文字に続いて変数a の内容を表示

input

変数の値をキーボードから入力する

`input a`	変数a に代入する数値を訊いてくる。
`input "a=";a`	画面にa=?と表示して数値a をきいてくる。

例 1.2 (簡単な計算) a と b の値を入力すると $a \times b$ を計算し結果を表示するプログラム。read,data を利用。

`10 'ex 1-2`	
`20 read a, b`	変数a とb の値をdata 文の先頭から順番に読み込む
`30 data 5, 2`	変数データを, で区切りながら順番に書いておく。（つまりa に5 が、b に2 が読み込まれる）
`40 c=a*b`	
`50 print "a*b=";c`	
`60 end`	

read, data

data 文に書いてある数値を read 文で順番に読み込む。

```
read a, b, c, d        a、b、c、d の値をdata 文から順番に読み込む
data 5, 12, 50, -90    read 文で読み込む順番に書いておく。
                       a=5 : b=12 : c=50 : d=-90 と同じこと。
```

課題 1.1 (計算・表示プログラム) 三角形の底辺と高さが与えられた時、面積を求め表示するプログラムを作成せよ。

課題 1.2 (計算・表示プログラム) 三角形の *2* 辺の長さ a、b とその挟む角 θ*(theta)* が与えられた時、残りの 1 辺の長さ c を求め表示するプログラム（input 文を使って作ってみよう）
（ヒント）余弦定理より $c^2 = a^2 + b^2 - 2ab\cos\theta$。$a^2$ の計算は a^2 又は a*a と書く。$\cos\theta$ の計算は cos(theta*3.14159/180)（コンピュータはラジアン単位)。$\sqrt{x}$ の計算は sqr(x)。

2　繰り返しと条件分岐

例 2.1 (繰り返し) 繰り返し命令 for···next を使って、例題 *1.1* を *10* 回繰り返すプログラム

10 'ex 2-1	
20 for n = 1 to 10 step 1	n を1 から10 まで1 ずつ変化させ、30-60 行の計算を繰り返す。
30 input "a=";a	for とnext で囲まれた部分が繰り返される。
40 input "b=";b	繰り返される部分を少し字下げして書いておくとわかりやすい。
50 c=a*b	
60 print"a*b=";c	
70 next n	n が10 になるまで20 行目に戻る。
80 end	

for ··· next

for と next で挟まれた部分を繰り返し実行する。step は変化量で、＋、ー両方 OK。for と next はセットで必要、対応していないとエラーが出る。

```
for n = 0 to 6 step 2   i を0 から6 まで2 ずつ変化させる。
 (計算や命令)
next i                  i が6 になるまでfor 文に戻り、繰り返す。
```

例 2.2 (繰り返し) 条件分岐命令 if を使って、例題 *1.1* を *10* 回繰り返すプログラム

10 'ex 2-2	
20 n=0	n の初期値を0 にする
30 n=n+1	n を1 増やす。ここに戻るたびにn が1 増える。
40 input "a=";a	
50 input "b=";b	
60 c=a*b	
70 print "a*b=";c	
80 if n<10 then 30 else 90	もしn<10 なら30 行に戻る。そうでなければ90 行に飛ぶ。
90 end	

if ··· then ··· else

条件を満たした時と満たさなかった時の制御移動先を指定する。行番号のところには、移動先ではなく計算式や命令を入れることもできる。

`if` 条件 `then 30 else 60`	条件を満たした時30 行に飛ぶ。満たさなかった時は60 行に飛ぶ。

課題 2.1 (繰り返し) 繰り返し命令を使って、*2* の *1* 乗、*2* 乗、*3* 乗、···、*15* 乗を計算するプログラムを作成せよ。

課題 2.2 (繰り返し) 二次方程式 $ax^2 + bx + c = 0$ の解を判別し、実数解なら解を求め、虚数解なら Kyosuu-kai!!と表示するプログラムを作成せよ（判別式を使って条件分岐させる）。

3　配列変数の活用

大量のデータの平均値を求めるなどの統計的な計算は、コンピュータの最も得意とする処理といって良い。この時、繰り返し命令や配列変数が威力を発揮する。

例 3.1 (配列) *12* 人に対して行った試験の平均点を求めるプログラム。

得点は以下の通り。
50, 92, 78, 32, 10, 60, 88, 36, 75, 62, 59, 99

```
10 'ex 3-1
20 dim tokuten(12)
30 goukei = 0
40 for n = 1 to 12
50   read tokuten(n)
60   print n, "tokuten";tokuten(n)
70   goukei = goukei + tokuten(n)
80 next n
90 heikin = goukei / 12
100 print "goukei=";goukei
110 print "heikin=";heikin
120 '
130 data 50, 92, 78, 32, 10, 60
140 data 88, 36, 75, 62, 59, 99
150 end
```

tokuten として12 個の配列を確保
最初に合計を 0 にしておく
12 個のデータを読む繰り返し。
data から 1 つずつ順番にデータを読み込む
読み込んだ得点の表示
得点を順次合計する。

平均点の計算
合計の表示
平均の表示

データを順番に記述。
２行以上に分けてもかまわない。

dim

配列変数の使用を宣言する。

```
dim x(100), y(100)
```
x として100 個、y として100 個の配列変数を確保する

配列変数の個数は、10 個までなら省略できるが、プログラムが読みにくくなるので必ず明示的に宣言すること。

課題 3.1 (配列) ある試験問題の回答率及び正解率の平均値を求めるプログラム。

設問番号	1	2	3	4	5	6	7	8
回答率 (%)	85.2	98.4	54.7	99.0	98.5	45.1	87.8	81.4
正解率 (%)	67.5	90.2	13.7	84.1	45.6	42.2	65.0	57.6

4 グラフの表示

BASIC では、簡単な命令で画面に線を引いたり点を出したりできるので、図形やグラフを表示することができる。ただし、コンピュータの画面では上下が反転しているため、y 座標の指定の仕方に工夫が必要となる。

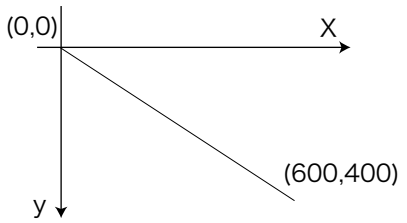

図 4.1: コンピュータの画面上の座標系

コンピュータでは、左上隅が常に (0,0) で、右下に行くと (x,y) の値が正になる。

画面上の表示サイズは、小さなドット 1 つ分が表示の単位となる。ドットは小さいからある程度の大きさがないとグラフが見えないが、大きすぎても一部しか見えない。目安としては、縦横およそ 400〜650 程度にすれば、ほどよいサイズで表示できる。

例 4.1 (グラフの表示) $y = x^2(\sin x)^2$ のグラフを、$-10 \leq x \leq 10$、$-15 \leq y \leq 100$ の範囲で表示する。

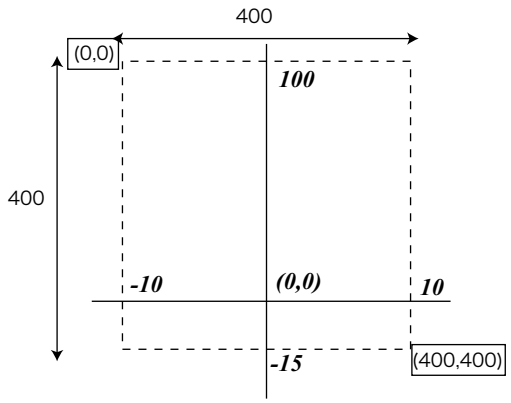

図 4.2: 画面上の座標と実際の座標

まず、画面上で 400×400 の範囲にグラフを表示することを考える。この範囲の中に、x 軸と y 軸を図 4.2 のように置く。斜体字で示したのが、表示したいグラフの座標系である。□で囲ったのが画面上の座標系である。

まず x 軸の画面上でのサイズは 400、これが ± 10 に対応するから、グラフ上の $x = 0$ の点は、画面上では半分のところで、`x0 = 200` となる[1]。y については、画面上の 400 がグラフでは $100 - (-15) = 115$ に対応する。さらに、下向きが正であるので、$y = 0$ はグラフ上では `y0 = 400 / 115 * 100` となる。

次に、グラフ上の大きさ 1 が画面上でいくつになるかを考える。x については、グラフ上の 20 が画面上の 400 に相当するので、比率は `dx = 400 / 20` となる。y については、グラフ上の 115 が画面上の 400 に相当するので、比率は `dy = 400 / 115` となる。

従って、グラフ上の点 `(x,y)` は、画面上ではそれぞれ `px = x0 + x *dx`、`py = y0 + (-y)*dy` となる。y の符号が逆になっているのは、画面上の座標の向きと描きたいグラフの向きが逆だからである。

これらを考慮すると、プログラムは次のようになる。

[1] y 座標と同様に考えると、$x0 = 400/20 * 10 = 200$

10 'ex 4-1	
20 screen 3,,0,1 : cls 3	画面設定と表示内容の消去
30 x0 = 200 : y0 = 400 / 115 * 100	(0,0) の画面上での位置を計算
40 dx = 400 / 20 : dy = 400 / 115	グラフ上での 1 が画面上でいくつになるか計算
50 line (x0 + (-10)*dx, y0)-(x0 + 10*dx, y0),4	x 軸（直線）を描く
60 line (x0, y0 + 15*dy)-(x0, y0 + (-100)*dy),4	y 軸 (直線）を描く
70 for x=-10 to 10 step 1	1 刻みで x 軸上に目盛りを描く
80 line (x0 + x*dx, y0+3)- (x0 + x*dx, y0-3),4	短い直線を繰り返し描く
90 next x	
100 for y=100 to -15 step -10	10 刻みで y 軸上に目盛りを描く
110 line(x0-3, y0 + (-y)*dy)-(x0+3, y0 + (-y)*dy),4	
120 next y	
130 for x = -10 to 10 step .05	0.05 刻みで表示すべき関数を計算
140 y = x^2*sin(x)^2	
150 pset(x0 + x*dx, y0 + (-y)*dy),5	(x,y) に点を描く
160 next x	
170 end	

line

指定した 2 点間を結ぶ直線を描く。

`line(x1,y1)-(x2,y2),c`	(x1,y1) と (x2,y2) を結ぶ直線を描く。c は色指定番号。青:1、赤:2、紫:3、緑:4、水色:5、黄:6、白:7。

pset

指定した座標に点を打つ。

`pset(x,y),c`	(x,y) に点を打つ。c は色指定番号。青:1、赤:2、紫:3、緑:4、水色:5、黄:6、白:7。

例 4.2 (グラフの表示) 水素原子の *1s* 軌道の動径分布関数 $4\pi r^2\psi^2$ を $0 \leq r \leq 3$ の範囲で表示する。

1s 軌道の波動関数は、

$$\psi = \frac{1}{\sqrt{\pi a_0^3}} \exp\left\{-\frac{r}{a_0}\right\} \tag{1}$$

ただし、a_0=0.529Å（Bohr 半径)。

10 'ex 4-2	
20 screen 3,,0,1 : cls 3	画面設定と表示内容の消去
30 x0 = 0 : y0 = 400	(0,0) の画面上での位置を計算
40 dx = 400 / 3 : dy = 400 / 2	グラフ上での 1 が画面上でいくつになるか計算。y 軸の範囲は 2 にした。
50 line(x0,y0)-(x0 + 3*dx, y0 + (-2)*dy), 4, B	外枠を描く。最後の,B は box を描けという意味。
60 for x = 0 to 3 step .5	外枠に 0.5 刻みで x 目盛りを描く
70 line(x0 + x*dx, y0)-(x0 + x*dx, y0-10), 4	
80 next x	
90 for y = 0 to 2 step .1	外枠に 0.1 刻みで y 目盛りを描く
100 line(x0, y0 + (-y)*dy)-(x0+10, y0+(-y)*dy), 4	
110 next y	
120 pi = 3.14159 : a = .529	
130 for r = 0 to 3 step .005	0.05 刻みで関数を計算する
140 p = exp(-r/a)/sqr(pi*a^3)	ψ の計算
150 y = 4 * pi * r^2 * p^2	$4\pi r^2\psi^2$ の計算
160 pset(x0 + r * dx, y0 + (-y)*dy),5	(r,y) に点を描く
170 next r	
180 end	

課題 4.1 (グラフ) 水素原子の *3s* 軌道の動径分布関数 $4\pi r^2\psi^2$ を $0 \leq r \leq 3$ の範囲で表示する。

3s 軌道の波動関数は、

$$\psi = \frac{1}{81}\frac{1}{\sqrt{3\pi a_0^3}}\left\{27 - 18\left(\frac{r}{a_0}\right) + 2\left(\frac{r}{a_0}\right)^2\right\}\exp\left(-\frac{r}{3a_0}\right) \tag{2}$$

課題 4.2 (グラフ) 例題 *3.1* の得点を棒グラフにする。

12 人分を横に並べたイメージ。line 文を用いて縦棒を描き、それを横に並べればいい。

課題 4.3 (グラフ) 練習 *3.1* の回答率を折れ線グラフにする。

line 文を用いて折れ線の 1 つ 1 つを描く時に、始点と終点をどのように指定するかがポイント。

5 アニメーション

グラフィックコマンドや分岐コマンドをうまく利用すれば、アニメーションを作ることができる。いろいろと試してみよう。

例 5.1 (グラフの表示) ボールを動かして模様を描く。

10 'ex 5-1	
20 screen 3,,0,1:cls 3	画面設定と表示内容の消去
30 dx = 400 : dy = 400	表示場所の画面上でのサイズ
40 line(0,0)-(dx,dy),4,B	枠を描く
50 if x <= 0 then xs = 1	x の折り返し（壁に当たった時のみ）
60 if x >= dx then xs = -1.1	x の折り返し
70 if y <= 0 then ys = 1.2	y の折り返し
80 if y >= dy then ys = -1	y の折り返し
90 circle(x, y),7,0	円を描く（黒色）
100 x = x + xs : y = y + ys	円の中心を動かす
110 circle(x, y),7,5	円を描く
120 goto 50	50 行目に戻る（無限に繰り返す）
130 end	

circle

指定した座標を中心として円を描く。

circle(x, y),r,c	(x,y) を中心として、半径r の円を描く。c は色指定番号。青:1、赤:2、紫:3、緑:4、水色:5、黄:6、白:7。

例 5.2 (グラフの表示) テニスゲームっぽいグラフィックス。

見やすくするため、プログラムの一部を改行していますが、入力するときは 1 行で入れること。

```
10 'ex 5-2                                                    
20 screen 3,,0,1:cls 3                                        画面設定と表示内容の消去
30 dx = 640 : dy = 400                                        使用する画面のサイズ
40 x = 40 : y = 40 : z = 300                                  ボールとラケットの初期位置
50 xs = 3 : ys = 3 : zs = 10 : s = 0                          ボールとラケットの移動速度
60 while 1                                                    無限ループ。wend までの間を繰り返す。

70   j = -j + 1 : screen 3,,j,16*(1-j)+1 : cls 2              2 枚の画面を交互に使う
80   if x <= 30 or x >= dx-30 then xs = -xs                   x 方向の折り返し
90   if y <= 30 then ys = -ys                                 y 方向の折り返し
100   if y >= dy-30 and abs(z-x) <= 30
          then ys = -ys : s = s + 1                           ラケットに当たっているか判別
110   if y >= 410 then 180                                    失敗したらループを抜ける
120   k$ = inkey$                                             ラケットを動かすためのキー入力

130   if k$ = "7" then z = z - zs                             7 なら左へ移動
140   if k$ = "9" then z = z + zs                             9 なら右へ移動
150   line(z-20, 377)-(z+20, 385),6,B                         新しいラケットを描く
160   x = x + xs : y = y + ys : circle(x,y),5,5               新しいボールを描く
170 wend                                                      繰り返しの範囲
180 screen 3,,0,1:cls 3                                       画面を消去
190 print "GAME OVER !! SCORE:";s                            スコアを表示
200 end
```

while ··· wend

条件を満たす間、処理を繰り返す。

```
while 条件    条件には、たとえばn <= 10 などのような条件式を入れる。
  (処理)      この部分が繰り返される
wend          while は必ずwend とセットで使う。
```

コンピュータの世界では、条件式の値が真のとき 1、偽のとき 0 となっている。n <= 10 という条件式があったとき、もし n=5 ならこの式の値は 1、もし n=12 ならこの式の値は 0 である。while 1 という表記は、条件式が常に真、ということになるので、無限に処理が繰り返される。

例 5.1 では、処理を無限に繰り返すために goto を使って始めの方に戻った。例 5.2 では、繰り返しを while 文で書いておき、条件を満たした時に goto で後に抜けた。goto でプログラムの実行場所を変えるのは便利だが、はじめの方に戻る処理は、プログラムが少し長く複雑になってくると、処理の全体の構造がわかりにくくなり、間違いのもとである。処理を繰り返すために goto を使うのは推奨できない。処理を終了するために繰り返し部分の後方に抜けるように書くと、混乱が生じない。

課題 5.1 (自由課題) オリジナルなアニメーションを作ってみよう。

6　数値計算その１：最小２乗法

実験をしたあと、測定した結果をグラフに表すと、測定誤差が必ず含まれるので、ばらつきが生じる。本来直線になるはずのものであれば、ばらついているデータを目で見て、大体真ん中を通る直線を定規で引くことになる。しかし、目分量に頼っていると、人によって直線の描き方が違うこともあるので、どれが一番それらしい直線かを決める必要がある。

図 6.1 の○で示したような測定結果を得たとしよう（説明のために、ばらつきを多少オーバーに表現している)。点と直線の距離は、点から y 軸に沿って線を引いたときの、直線までの長さで測ることにする。長さをどちら向きに測ったかで場合分けすると複雑になるので、点と直線がどれだけ隔たっているかは、長さの２乗で評価することにする。

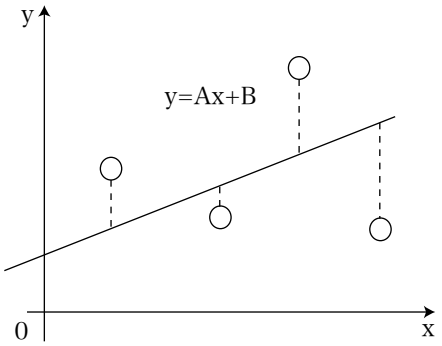

図 6.1: 一番それらしい直線とは？

直線の引き方が、特定の点のどれかに近づきすぎると、他の点からは遠ざかることになる。どの点からもそれなりの距離にあるほどよい直線であるには、それぞれの点から直線までの距離の２乗の和が最小になるように線を引けばよい。

目的とする直線を、

$$y = Ax + B \tag{3}$$

とする。N 個の点 $(x_i, y_i)(i = 1, 2, \cdots, N)$ から、最適な A、B の値を推定する方法について説明する。

各々の点から y 軸に沿っておろした線分の長さの 2 乗の和を $E(A, B)$ とする。

$$E(A, B) = \sum_{i=1}^{N} (y_i - Ax_i - B)^2 \tag{4}$$

E は、一般に複雑な振る舞いをするが、最小になるところでは、A や B を変えたときの、変化の傾きが 0 になる。つまり、

$$\left\{ \begin{array}{ll} \dfrac{\partial E}{\partial A} & = 2\left\{ A\displaystyle\sum_{i=1}^{N} x_i^2 + B\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} x_i y_i \right\} = 0 \\ \dfrac{\partial E}{\partial B} & = 2\left\{ A\displaystyle\sum_{i=1}^{N} x_i + BN - \sum_{i=1}^{N} y_i \right\} = 0 \end{array} \right. \tag{5}$$

一見複雑に見えるが、A、B に関する連立方程式の形をしているので、解くことができる。

$$\left\{ \begin{array}{ll} A & = \left\{ N\displaystyle\sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i \right\} / \left\{ N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i \right)^2 \right\} \\ B & = \left\{ \displaystyle\sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} x_i y_i \right\} / \left\{ N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i \right)^2 \right\} \end{array} \right. \tag{6}$$

課題 6.1 (最小２乗法) 次の 5 点について、最小２乗法により理論式 $y = Ax + B$ の A、B を推定し、点と直線をグラフとして表示せよ。

x_i	y_i
1	0.8
2	1.95
3	2.3
4	2.64
5	3.12

ここで説明したのは、直線をデータに合わせるという最も簡単な最小２乗法である。もっと複雑な一般の関数についても、曲線と点の間の距離を同じように決めて、２乗和の最小値を計算することができ、効率のよい計算方法がいくつかわかっているが、プログラムはだいぶ複雑になる。

7　数値計算その２：微分方程式を数値的に解く

1 次反応

$$A + B \longrightarrow \text{反応物} \tag{7}$$

では、$[A]$ の濃度の時間的な変化が

$$v(t) = -\frac{d[A]}{dt} = k[A] \tag{8}$$

となる。$t = 0$ での $[A]$ の濃度が $[A]_0$ であったとすると、この式を積分して、

$$\ln \frac{[A]}{[A]_0} = -kt \tag{9}$$

$$[A] = [A]_0 e^{-kt} \tag{10}$$

となる。

この微分方程式を数値的に計算する。

一般に、1 階の微分方程式は、$f(t, y)$ を任意の関数とし

$$\begin{cases} \dfrac{dy}{dt} & = f(t, y) \\ y(0) & = a \end{cases} \tag{11}$$

の形で表される。ただし、a は定数で、解 $y(t)$ は $t > 0$ の範囲で求めるものとする。

右辺の $f(t, y)$ は、解となる関数の、各点での接線の傾きになっている。

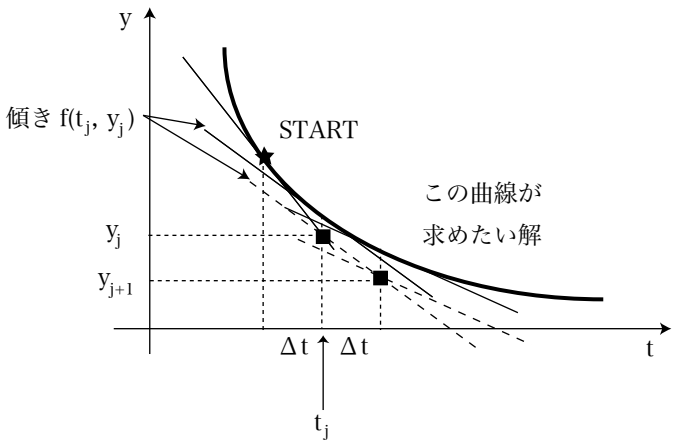

図 7.1: 次々に傾きを求めながら近似する

ある点 t_j での接線の傾きを計算する。接線の傾きは、微分方程式の右辺に $t = t_j$ を代入して計算するだけでよいから、$f(t_j, y_j)$ となる。ただし、y_j の値はわかっているものとする。t_j から Δt 離れた場所で、y の値が y_{j+1} であるとする。微分方程式から得られた接線の傾きと、$t = t_j$ での変化量が同じである、と近似して、

$$\frac{y_{j+1} - y_j}{\Delta t} = f(t_j, y_j) \tag{12}$$

となる。このように近似して微分方程式を数値的に解く方法を、オイラー法という。

オイラー法を計算するプログラムを作る。$0 \leq t \leq T$ の範囲で計算することにして、計算する範囲を N 分割する。また、$t = 0$ で $y(0) = a$ とする。

1. $\Delta t := T/N$
   $y_{old} := a$
2. $j = 0, 1, \cdots N-1$ で、

   $t := j\Delta t$
   $y_{new} := y_{old} + \Delta t f(t, y)$

   を繰り返す。

:= は、通常の等号ではなく、右辺を左辺に代入するという意味である。

課題 7.1 (オイラー法) 式 *8* で表される微分方程式をオイラー法で計算し、$[A]$ が時間的にどのように変化するかグラフを描け。

$[A]_0 = 1$、計算は $0 \leq t \leq 200$ まで行うものとする。速度定数は、$k = 0.0125$、$k = 0.0250$、$k = 0.0500$、$k = 0.1000$、の 4 種類について計算せよ。N の値は大きすぎるとうまく曲線を求めることができなくなるが、小さすぎると計算がなかなか終わらない。ほどよい N を探してみよう。

オイラー法よりも精度の高い計算方法として、ルンゲ・クッタ法が知られている。計算方法は次の通り。

1. $\Delta t := T/N$
   $y_{old} := a$
2. $j = 0, 1, \cdots N-1$ で、

   $t := j\Delta t$
   $k_1 := f(t, y)$
   $k_2 := f\left(t + \frac{\Delta t}{2}, y + \frac{\Delta t}{2} k_1\right)$
   $k_3 := f\left(t + \frac{\Delta t}{2}, y + \frac{\Delta t}{2} k_2\right)$
   $k_4 := f\left(t + \Delta t, y + \Delta t k_3\right)$
   $y_{new} := y_{old} + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

   を繰り返す。

課題 7.2 (ルンゲ・クッタ法) 問題 *7.1* を、ルンゲ・クッタ法で計算し、同じようにグラフを描け。

例 7.1 (サブルーチン) 繰り返して使う計算は、1 個所にまとめると便利である。

与えられた x、y について、x+y を表示するプログラム。

10 'ex 6-1	
20 x=1: y=3	サブルーチンに渡すパラメータを設定
30 gosub *test	サブルーチン*test に飛ぶ 。行番号を指定しても飛べる。
40 x=5 : y=-2	別の値をパラメータとして設定
50 gosub *test	
60 end	
70 '	もしn<10 なら30 行に戻る。そうでなければ90 行に飛ぶ。
80 *test	サブルーチンを呼び出すためのラベル。
90 print "x+y=";x+y	サブルーチン本体
100 return	サブルーチンの最後はreturn と書く

n88BASIC では、サブルーチン内で使っている変数 x、y はプログラムのどこからでも見えるので、他の場所で忘れて値を変えたりしないように注意が必要である。

gosub にさしかかると、指定されたラベルあるいは行番号にジャンプする。ジャンプした先で行番号の通りに実行し、return に出会うと、ジャンプした直後に戻る。サブルーチンの最後には return が必要である。

GOSUB ··· RETURN

サブルーチンにジャンプする。

```
gosub 100    行番号100 から始まるサブルーチンに飛ぶ。
gosub *sub   ラベル*sub で始まるサブルーチンに飛ぶ。ラベルは、*で始めること。
```

サブルーチンの最後には、必ず return を書くこと。書かないとエラーになる。