科学技術計算用スクリプト電卓 AprocD 操作説明

1．はじめに

電卓を使って計算するとき不便に思うことはありませんか？

検算のとき、或いはミスしたとき同じ数を何度も入力しなければならなかったり
（複雑な式や、桁数が多い数値は厭になりますね）
別の計算結果と比べたいけれど、記録していなかったり
10進数と16進数の切り替えが煩わしかったり
本当に正しく入力したか心配で眠れなくなったり

でも、わざわざ表計算ソフトなどを使うほどのものではないし …

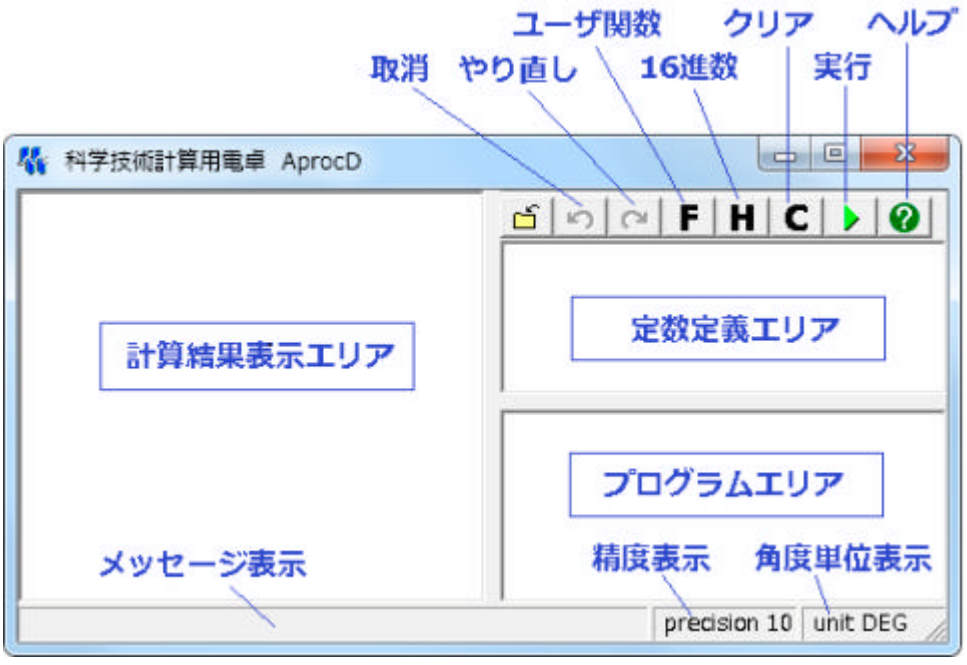
しょせん電卓だから、しょうがないと諦めていませんか？

AprocD は、そんなあなたのためのプログラム電卓です。以下の特徴があります。

入力した式や数値をずっと覚えていて、簡単に再利用できます。
ボタンを使わず数式で入力しますので、結果が読みやすく履歴も明快です。
異なるパラメータの値について一括計算できます。
条件分岐や繰返しを含む複雑な処理も可能です。
16種類の演算子と20種類の関数を備えています。
自分で定義すれば、関数の数はいくらでも増やせます。
10進数と16進数を併記表示できます。
グラフも簡単に描けます

2．AprocD の外観

この図は **AprocD** の起動画面です。３つのエリアに区切られていて、右側の２つが入力用エリア、左側が出力表示用エリアです。各エリアのサイズは、境界をドラッグして自由に変更できます。



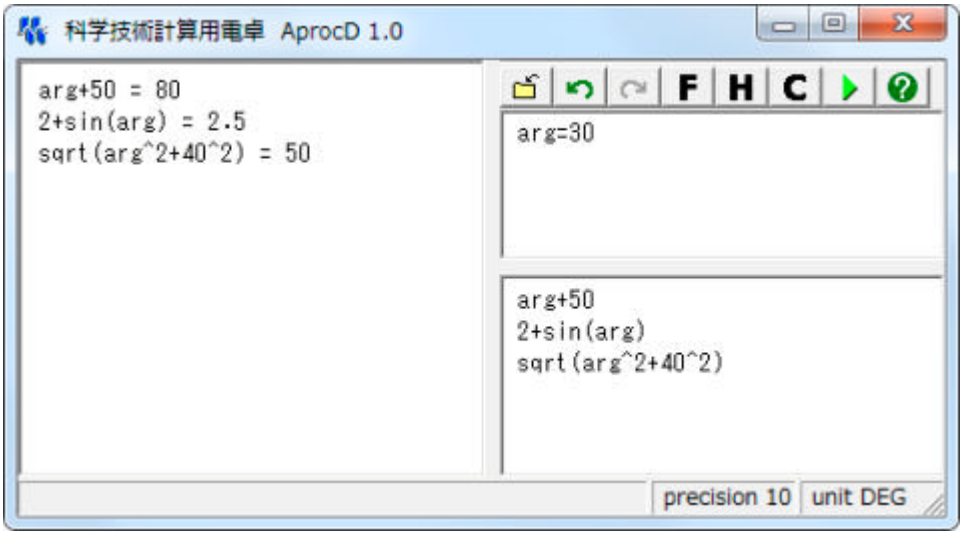
３．簡単な計算例

プログラムエリアに計算したい式を書いて実行ボタン ▶ またはファンクションキー **F9** を押します。たとえば、30+50 と入力して実行ボタン ▶ を押すと、出力エリアに 30+50=80 と表示されます。 いちどにたくさんの式を書いてもかまいません。この例は式を３つ書いて実行した結果です。[1]

４．パラメータの利用

前の例には、30という数が3回使われています。この値を変更したいとき、いちいち書き直すのは手間ですね。
そこで、変化させたい数値を**パラメータ**として文字[2]で書きます。

この例では、30 という数を arg という単語で置き換え、定数定義エリアに arg=30 と記しました。
こうすると 30という数字は1つだけになるので変更は容易です。

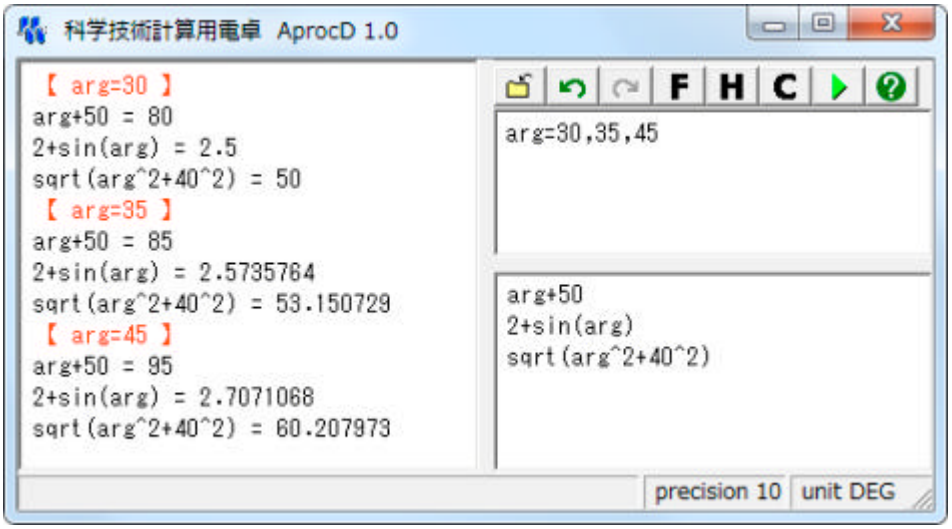


---

[1] sin( ) は三角関数、sqrt( ) は平方根関数、 記号 ^ はべき乗の演算子です。（付録参照）

[2] **パラメータ** や、これから説明する **変数** 、および**ユーザ関数**のスペルは、英字からはじまり、 英数字だけを含む文字列であれば自由です。文字数に制限はありません。ただし、予約語(付録参照)を除きます。

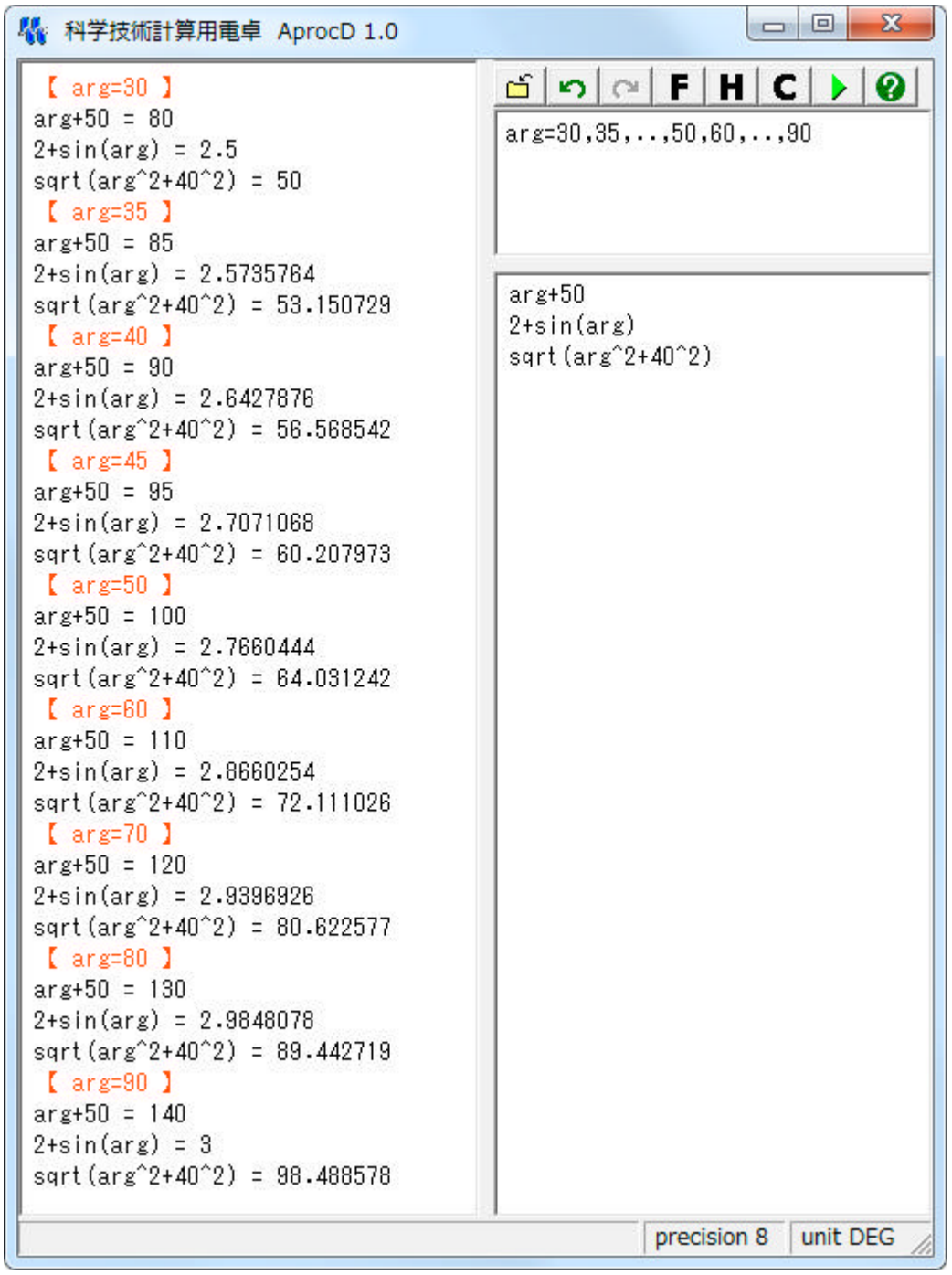
複数の数値をコンマで区切って指定すると、それぞれの計算結果がすべて表示されます。このように、複数の値を与えたパラメータを **くり返しパラメータ** と呼び、ひとつだけ使うことができます。



くり返しの回数が多い場合、途中を2個以上連続するピリオドで代用できます。 次の例では、

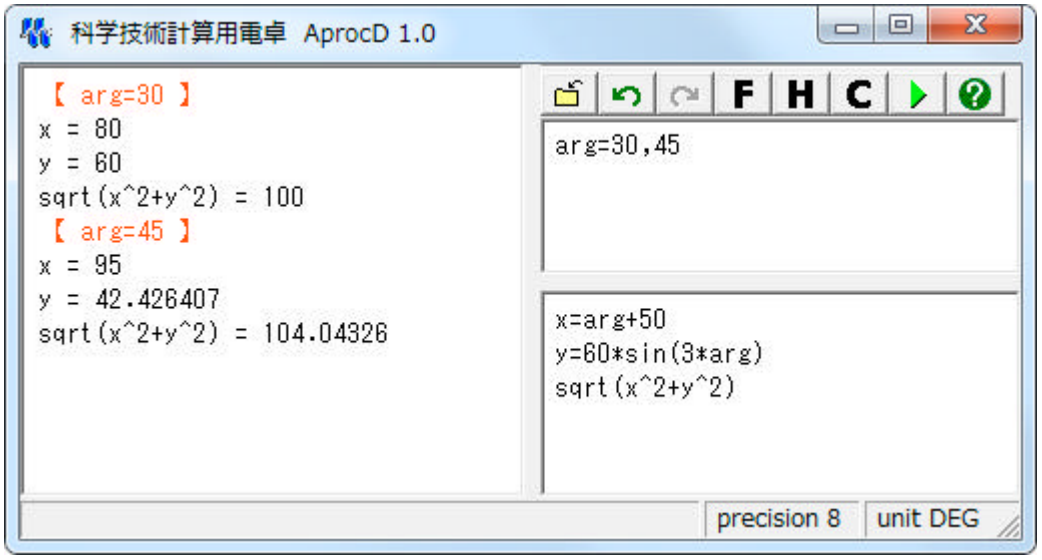
arg = 30, 35, … , 50, 60, …, 90

と書いて、arg の値を 30から50までは5毎に、50から90までは10毎に変化させています。


科学技術計算用電卓 AprocD 1.0
【 arg=30 】
arg+50 = 80
2+sin(arg) = 2.5
sqrt(arg^2+40^2) = 50
【 arg=35 】
arg+50 = 85
2+sin(arg) = 2.5735764
sqrt(arg^2+40^2) = 53.150729
【 arg=40 】
arg+50 = 90
2+sin(arg) = 2.6427876
sqrt(arg^2+40^2) = 56.568542
【 arg=45 】
arg+50 = 95
2+sin(arg) = 2.7071068
sqrt(arg^2+40^2) = 60.207973
【 arg=50 】
arg+50 = 100
2+sin(arg) = 2.7660444
sqrt(arg^2+40^2) = 64.031242
【 arg=60 】
arg+50 = 110
2+sin(arg) = 2.8660254
sqrt(arg^2+40^2) = 72.111026
【 arg=70 】
arg+50 = 120
2+sin(arg) = 2.9396926
sqrt(arg^2+40^2) = 80.622577
【 arg=80 】
arg+50 = 130
2+sin(arg) = 2.9848078
sqrt(arg^2+40^2) = 89.442719
【 arg=90 】
arg+50 = 140
2+sin(arg) = 3
sqrt(arg^2+40^2) = 98.488578
F
H
C
arg=30,35,..,50,60,..,90
arg+50
2+sin(arg)
sqrt(arg^2+40^2)
precision 8
unit DEG

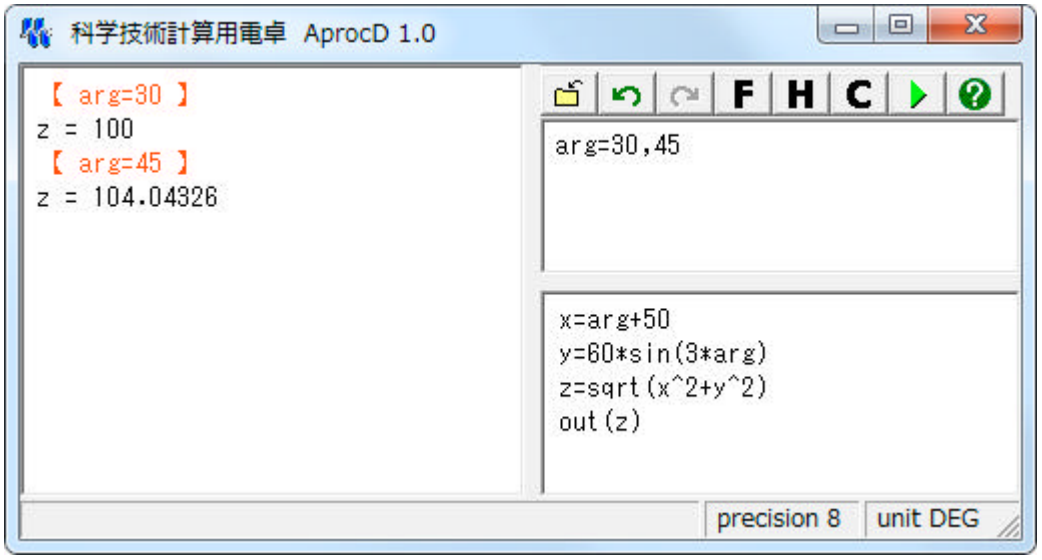
５．変数の利用

式の計算値を **変数** へ代入しておくと、その値を継続する行で使うことができます。 次の例では、第1行の計算結果を変数 x へ　第2行の計算結果を変数 y へ代入し、第3行で x と y の2乗和の平方根を計算させました。変数を使うと、ややこしい計算を簡潔に記述することができます。



特定の結果だけ表示させたい場合は、関数 out を使って変数を指定します。下の例では、3番目の式を 変数 z に代入し、out(z) としました。 以下の構文で複数の変数も指定できます。

out(**変数1，変数2，…**)



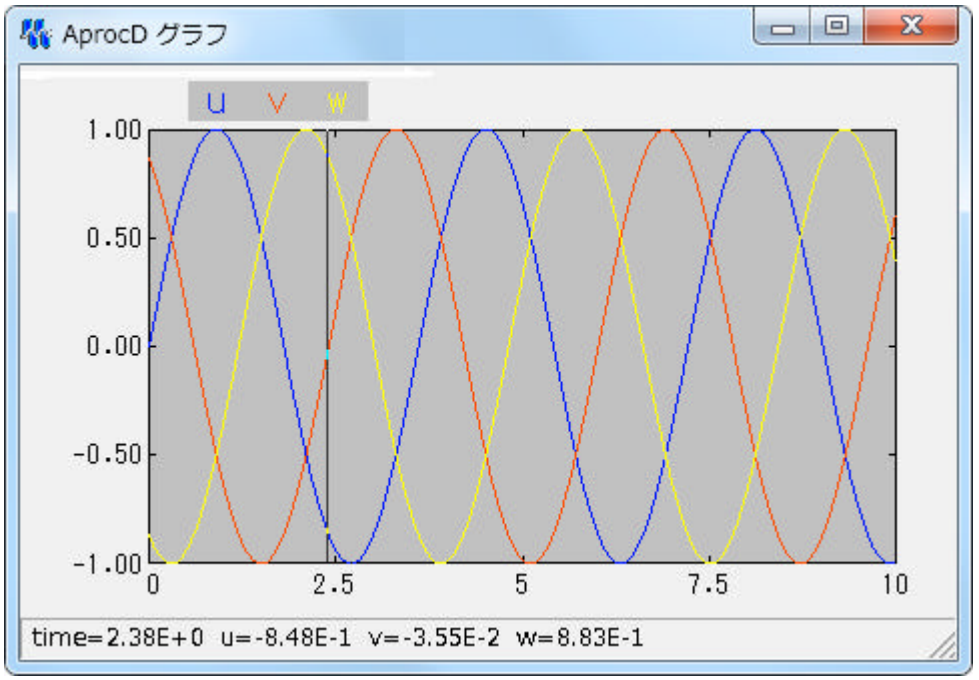
６．グラフの表示

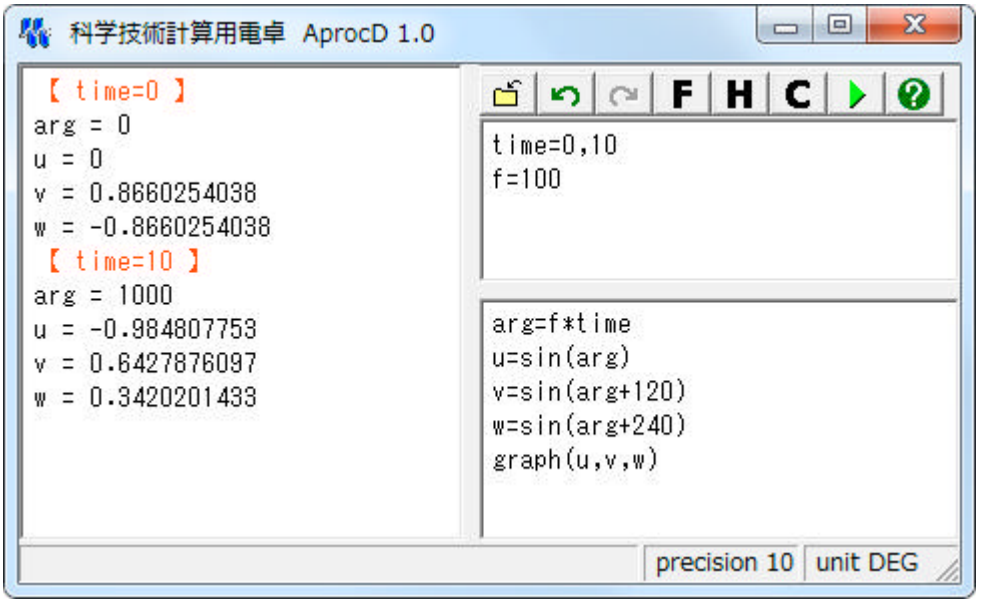
出力の指定と同じようにして、下記の構文で指定した変数をグラフ化することができます。

graph（ **変数**1，**変数**2，… ）

以下の例は、位相が120度ずつ異なる正弦関数を u, v, w という変数へ代入し、それらを指定しました。
くり返しパラメータ time の範囲を横軸にしてグラフが描かれます。

グラフをクリックすると **カーソル** が現れ、カーソル位置の数値がステータスバーに表示されます。 右クリックするとカーソルは消えます。





7．プログラム

変数を使った処理は既にプログラムになっている訳ですが、 さらに、**条件分岐** や **くり返し** を含む処理も記述できます。 次の例は、自然対数の底を次式のくり返し加算で計算しました。

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

加算回数は ∞ ではなく くり返しパラメータ Repeat に 10回、20回、30回と与えました。
比較のため、指数関数 exp(1) も計算しています。

プログラムエリアの記述で while という単語がくり返しを意味しています。以下の構文になります。

```
while 条件文
{
    1行以上の文
}
```

この構文は、**条件文**[3]が満たされる限り、 { } で囲まれた文を繰り返します。 例では、変数 n が毎回1ずつ増えていくので、Repeat=10の場合、10回目に n<Repeat という条件が満たされなくなって終了します。
最後の行に「//確認用」とあるのは **コメント** です。 "// "から行末までには何を書いても無視されます。

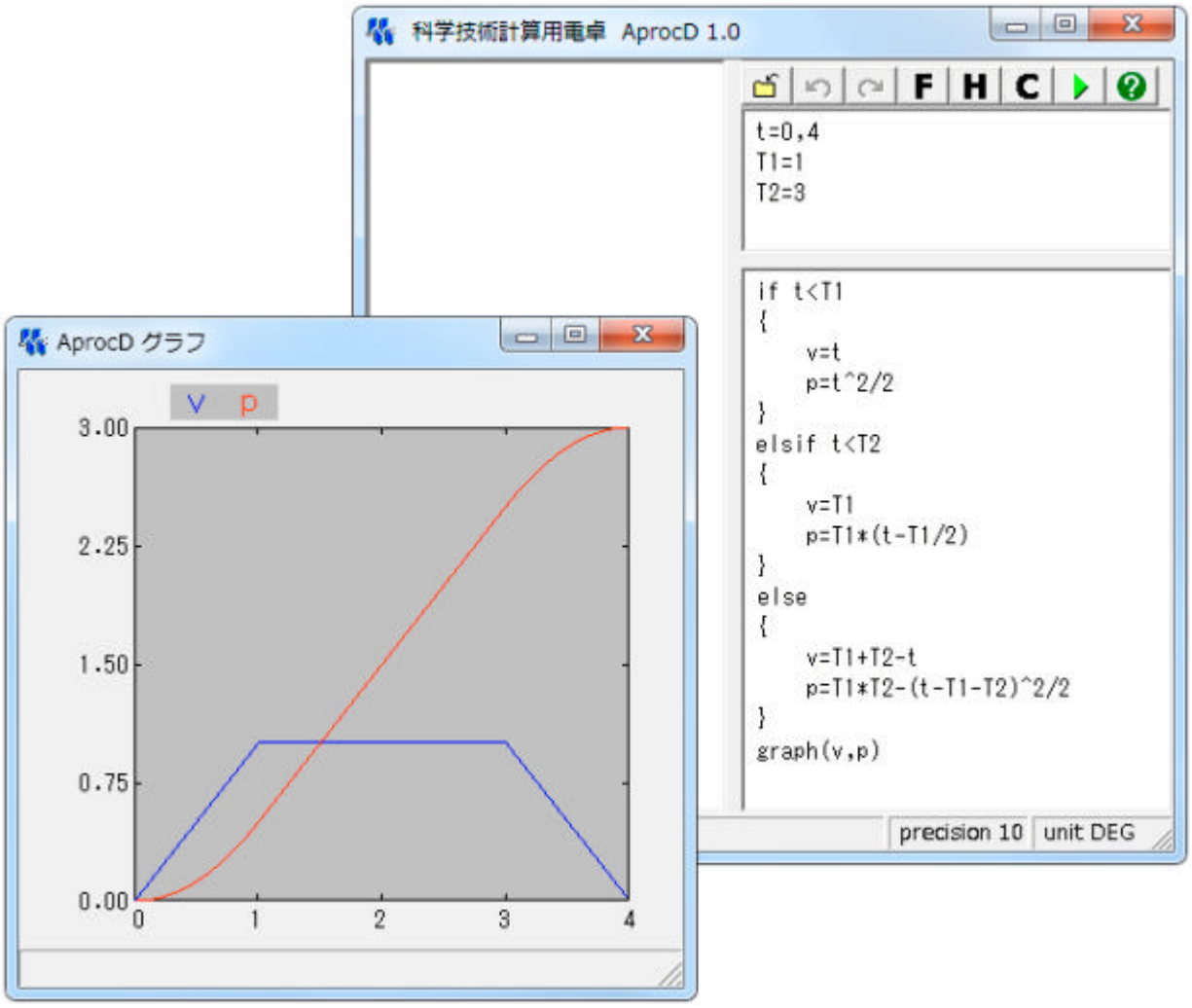
[3] 条件文とは、パラメータ、変数、関数 または、 それらを演算子で接続した式のことです。 そして、その値が正なら **真**（true）、0または負なら **偽**（false）となります。 while や if に続く条件文が真（つまり正）の場合だけ、その直後の { … } の内容が実行されます。

条件分岐用の制御文として if 、elsif 、else があります。以下の構文になります。

```
if  条件文
{
      1行以上の文
}
elsif 条件文
{
      1行以上の文
}
 else
{
      1行以上の文
}
```

”elsif　…{　…}　”　と　”else　{　…}　”は省略可能です。 また、”elsif　…　{　…}　”は複数行を並べて書くことができます。

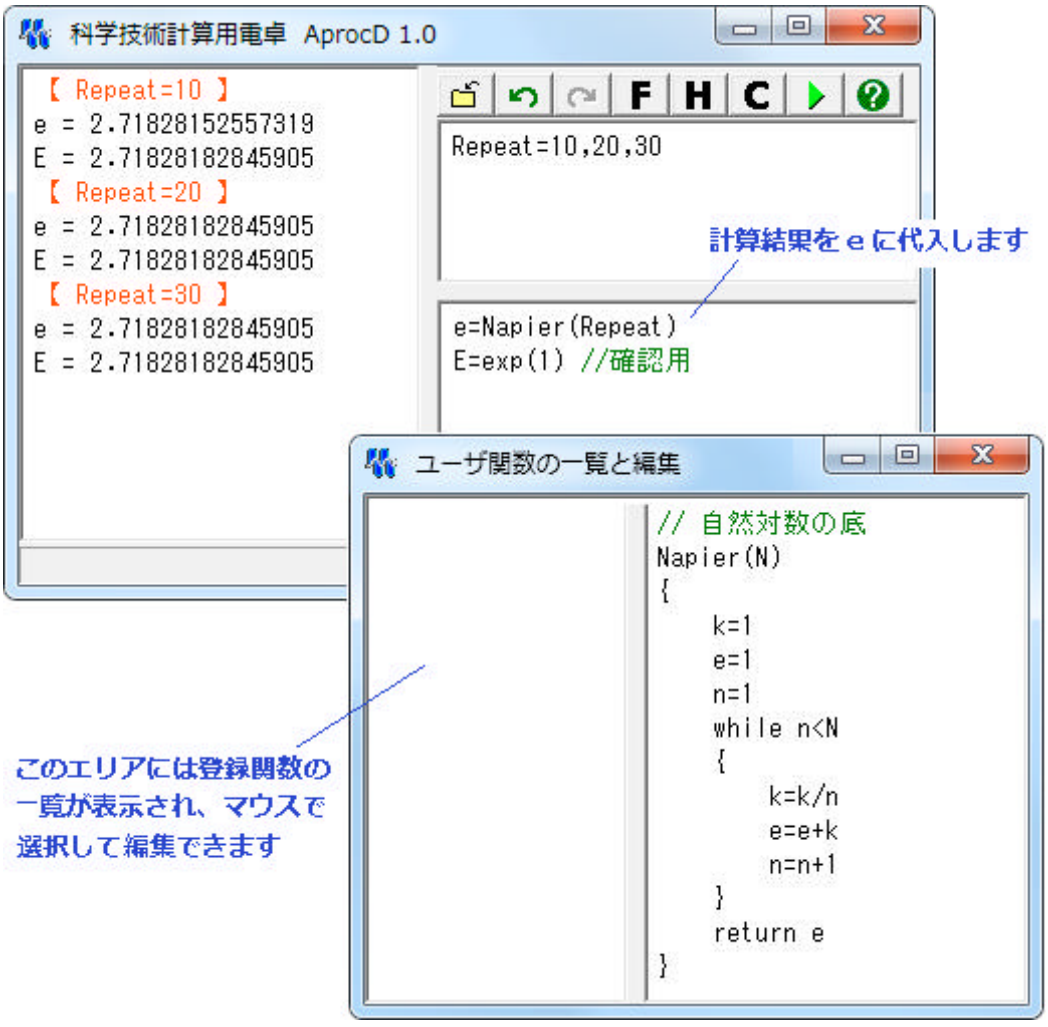
プログラムの例を以下に示します。くり返しパラメータ t が 0 から 4 まで変化するとして、 0≦t＜1, 1≦t＜3,　3≦t≦4　という 3 つの場合に異なる計算を行っています。



８．ユーザ関数

作成したプログラムを **ユーザ関数** として **登録** しておくと、他の用途にも利用できて便利です。 ボタン  を押すとユーザ関数編集用のウィンドウが現れますので、その右側のエリアに作成したい関数を定義し、**右クリックメニュー** で登録します。

この例は、自然対数の底を求めるプログラムを Napier というユーザ関数に書き直して実行した結果です。
プログラムエリアの記述はこの関数を参照するだけになりました。



ユーザ関数の記述内容は、前の例で Repeat を N に替え、先頭にコメントと関数名を、最後に計算結果を返す return **文** を追加しました。 ユーザ関数の構文は以下です。

```
// 一覧表に表示されるコメント（省略可）
関数名（引数1, 引数2, ...）
{
      0行以上の文
      ......
    return 変数名または式
}
```

作成したユーザ関数は、右クリックメニューから **登録または更新** します。　登録すると、その関数は通常の関数と 全く同じように使用できます[4]。また、ファイルへの保存や、保存ファイルからの登録も可能です。

ユーザ関数利用して定数を登録することもできます。例えば、引数のない関数を

```
PI(
{
        return  3.14159265358979
}
```

として登録しておくと、PI() を円周率として使うことができます。

９．入力の再利用

定数定義エリアとプログラムエリアに記述された（５文字以上の）文や数値を、実行時に記憶[5]しています。

再び同じ内容を入力しようとすると、最初の１文字を入力したとき、 同じ文字から始まる過去の入力が
別ウィンドウに現れます。

その中の適当な文をクリックすれば、入力が完了します。

２文字まで入力すると、２文字目が異なるものは消えますので候補は少なくなります。
３文字まで入力すると、３文字目が異なるものは消えますので候補はさらに少なくなります。

・・・・・・・・・　（以下同文）

これで説明は終わりです。 **AprocD** をぜひご活用ください。

[4] あるユーザ関数から別のユーザ関数を呼び出すこともできます。 ただし、循環（お互いを直接、間接に呼び合うこと）があるとエラーになります。

[5] 実行しなかったり、エラーが発生した場合は記憶しません。
不要な記憶は削除できます 。
この機能がうるさいと思われる方のため、無効化のメニューがあります

付録１．ボタンとメニュー

⑴ メインウインドウ （ 起動時のウィンドウ ） のボタン

項目	ボタン	説 明	ショートカット
終了		現在の表示と設定をファイルに保存します	Ctrl+S
アンドゥ／リドゥ		入力を（取消し／やり直し）します	Ctrl+**Y**/**Z**
ユーザ関数	**F**	ユーザ関数を作成・編集するウィンドウを開きます	
16進数表示	**H**	演算結果の数値を16進数で併記します	
クリア	**C**	出力エリアの内容を消去します	
演算実行		演算を開始し、結果を表示します	**F 9**
ヘルプ		このページを開きます	**F 1**
角度単位	unit ###	⑴三角関数で使用する角度の単位を変更します	
表示精度	precision ##	結果の表示桁数を1～15の数で指定します 演算精度とは関係ありません	

⑵ メインウインドウ（ 起動時のウィンドウ ） の右クリックメニュー

メニュー	説明または同じ機能のボタン	ショートカット
実行		**F 9**
ヘルプ		**F 1**
ファイルから読む	表示と設定内容をファイルから読み、或いは保存します	
新規作成	エリアをクリアします	
上書き保存する		Ctrl＋S
名前をつけて保存する	別名で保存します	
関数一覧を表示する	**F**	
入力再利用を無効にする	無効になるとメニューにチェックがつきます 再度実行するとチェックが消え、有効になります	
フォントの指定	フォントの種類とサイズを指定します	
元に戻す（アンドゥ）		Ctrl＋**Z**
やり直す（リドゥ）		Ctrl＋**Y**

⑶ 関数編集用ウィンドウの右クリックメニュー

メニュー	説 明
新しい関数の作成	ユーザ関数のテンプレートを表示します
登録または更新	編集エリアの内容を登録します 既に同じ名前の関数が登録されている場合は上書きします
登録済関数の保存	登録されている全ての関数をファイルに保存します
ファイルから登録	保存したファイルから、未登録の関数だけを登録します
アルファベット順 （登録・更新の）新しい順	登録済関数のリストを並べ替えます
元に戻す／やり直す	ショートカットは Ctrl＋Z ／ Ctrl＋Yです
選択した関数の削除	選択されている関数を登録から削除します

⑷ 過去の入力ウインドウの右クリックメニュー

メニュー	説 明
全て表示	全ての記憶を表示します
全て選択	表示されているすべての文を選択します
選択部の削除	選択されている文を削除します 選択するときにＡＬＴキーを押しておくと、文の入力を禁止できます

付録２　演算子

⑴ 単項演算子

記号	意 味	結 果	例	
-	符号反転	続く式の値の符号を反転させます	x = y * -z[注]	x は y と -z の積
!	否定	続く式が真なら0、偽なら1になります	if !(x=3) { ... }	x が3未満なら...

【注】 符号反転には最も高い優先度がありますので、必ずしも -z を（ ）で囲む必要はありません。

⑵ ２項演算子

記号	意 味	結 果	例	
+	加算	2項の和	z = x + y	記述どおりの意味になります。なお、乗除算は加減算に優先します。
-	減算	2項の差	z = x - y	
*	乗算	2項の積	z = x * y	
/	除算	2項の商	z = x / y	
%	剰余	整数部[注1]で除算した余り	z = 5.5 % 3.7	z は 5÷3 の余りで 2
^	べき乗	べき計算[注2]	z = x^3 + 2*y^3 [注3]	$z = x^3 + 2 \cdot y^3$
>	大	記述された関係が 真なら1 偽なら0	y = x >0	x が正ならy=1、それ以外ならy=0
>=	非小		if x = 0 { ... }	x が負でなければ...
<	小		if x < 0 { ... }	x が負なら...
<=	非大		if x <= 0 { ... }	x が正でなければ...
= =[注4]	等しい		if x = = 1 { ... }	x = １.0 なら...
!=	等しくない		if x != 5 { ... }	x が 5.0 でないなら...
&&	論理積	前後の値が正なら真、0または負なら偽[注5]として演算し、その関係が真なら1、偽なら0	if (x=0) && (x<=9) { ... }	xが0以上9以下なら...
| |	論理和		if (x0) | | (y0) { ... }	x、yのいずれかが正なら...

【注1】 ある数の整数部とは、その符号によらず、小数点以下を無視した値です。
【注2】 べき数が整数でないとき、x^y = exp(y * ln(x)) と計算しますので x≦0 はエラーになります。
【注3】 べき乗の演算は乗除算よりも優先されます。
【注4】 この表では、フォントの都合で= =の等号の間にスペースを入れていますが、実際はありません。
【注5】 Ｃ言語などの論理（0以外は真）とは異なります。

付録３　関数

関数名	記号	引数	説明
絶対値	abs(x)	実数	絶対値を演算します
平方根	sqrt(x)	正数	平方根を演算します
指数関数	exp(x)	正数	e^x を演算します
対数	log(x)	正数	自然対数を演算します
	log10(x)	正数	常用対数を演算します
三角関数	sin(x)	実数	正弦値 sin を演算します
	cos(x)	実数	余弦値 cos を演算します
	tan(x)	実数	正接値 tan を演算します
逆三角関数	asin(x)	|x|<1	逆正弦値 $\sin^{-1}$ を演算します
	acos(x)	|x|<1	逆余弦値 $\cos^{-1}$ を演算します
	atan(x)	実数	逆正接値 $\tan^{-1}$ を演算します
	atan2(x,y)	実数	逆正接値 $\tan^{-1}(y/x)$ を４象限(-π～π)で演算します
双曲線関数	sinh(x)	実数	sinh を演算します
	cosh(x)	実数	cosh を演算します
	tanh(x)	実数	tanh を演算します
リミッタ	limit(min,max, x)	上下限と入力	限界値で飽和させます
正規雑音	gauss(x)	実数	平均 0、標準偏差 x の正規白色乱数を発生します
一様乱数	random()	なし	0～1の一様な白色乱数を発生します
ユーザ関数補助	aux(var)	変数名	ユーザ関数内で定義された変数の値を返します AprocSの auxfunc() と同じです
出力指定	out(v1,v2,...)	変数名	計算結果の表示を指定した変数だけにします
グラフ指定	graph(v1,v2,...)	変数名	指定した変数のグラフを表示します

付録４　予約語

あらかじめ決められた単語を **予約語** と言います。その一覧を以下に示します。

記述用の予約語	else, elsif, if, loops, return, while
関数名としての予約語	abs, acos, asin, atan, atan2, aux, cos, cosh, exp, gauss, graph, int, limit, log, log10, out, random, sin, sinh, sqrt, tan,

パラメータ名、変数名、およびユーザ関数名として、これらを使うことはできません。 ただし、
大文字と小文字を区別しますので、同じスペルでも大文字を含んでいれば使用可能です。