常微分方程式の解を計算するC言語のプログラム

山本昌志*

2005年10月21日

1　1階の常微分方程式

1.1　4次のルンゲ・クッタ法

1.1.1　概要

前回の授業で学習した4次のルンゲ・クッタ法のプログラムの例をリスト1に示す．このプログラムでは，微分方程式

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \sin x \cos x - y \cos x \tag{1}$$

を解く．このプログラムは，4つの関数から構成され，それぞれの役割は以下の通りである．

- `main()` 関数
  - 計算条件をキーボードより，読み込む
  - 4次のルンゲ・クッタ法で微分方程式を計算する．
  - 計算結果は，配列 `x[]` と `y[]` に格納される．
- `func()` 関数
  - 導関数 $\frac{\mathrm{d}y}{\mathrm{d}x}$ を計算する．
- `mk_plot_data()` 関数
  - gnuplotを用いて解をプロットするために，計算結果 (`x[],y[]`) をファイルに書き出す．
- `mk_graph` 関数
  - gnuplotを呼び出して，解をプロットする．

*国立秋田工業高等専門学校　電気工学科

1.1.2 使い方

このプログラムの使用方法は，以下の通りである．もし，異なった微分方程式を計算したければ，func()関数を書き直せばよい．

- 初期条件 x_0 と y_0 を聞いてくるので，それぞれをキーボードから入力する．
- 計算終了の x の値を聞いてくるので，それをキーボードから入力する．
- 計算ステップ数を聞いてくるので，それをキーボードから入力する．

リスト 1: 4 次のルンゲ・クッタ法

```
1  #include <stdio.h>
2  #include <math.h>
3  #define IMAX 100001
4  double func(double x, double y);
5  int mk_plot_data(char *a, double x[], double y[], int n);
6  int mk_graph(char *f);
7
8  /*=============================================================*/
9  /*        main function                                        */
10 /*=============================================================*/
11 int main(){
12   double x[IMAX], y[IMAX];
13   double final_x, h;
14   double k1, k2, k3, k4;
15   char temp;
16   int ncal, i;
17
18
19   /*--- set initial condition and cal range ---*/
20
21   printf("\ninitial value x0 = ");
22   scanf("%lf%c", &x[0], &temp);
23
24   printf("\ninitial value y0 = ");
25   scanf("%lf%c", &y[0], &temp);
26
27   printf("\nfinal x = ");
28   scanf("%lf%c", &final_x, &temp);
29
30   do{
31     printf("\nNumber of calculation steps( <= %d ) = ",IMAX-1);
32     scanf("%d%c", &ncal, &temp);
33   }while(ncal > IMAX-1);
34
35
36   /* --- size of calculation step --- */
37
38   h=(final_x-x[0])/ncal;
39
40
41   /* --- 4th Runge Kutta Calculation --- */
42
43   for(i=0; i < ncal; i++){
44     k1=h*func(x[i],y[i]);
45     k2=h*func(x[i]+h/2.0, y[i]+k1/2.0);
46     k3=h*func(x[i]+h/2.0, y[i]+k2/2.0);
```

```
47      k4=h*func(x[i]+h, y[i]+k3);
48
49      x[i+1]=x[i]+h;
50      y[i+1]=y[i]+1.0/6.0*(k1+2.0*k2+2.0*k3+k4);
51    }
52
53
54    /* --- make a graph --- */
55
56    mk_plot_data("out.txt", x, y, ncal);
57    mk_graph("out.txt");
58
59    return 0;
60  }
61
62
63  /*=============================================================*/
64  /*        define function                                      */
65  /*=============================================================*/
66  double func(double x, double y){
67    double dydx;
68
69    dydx=sin(x)*cos(x)-y*cos(x);
70
71    return(dydx);
72  }
73
74
75  /*=============================================================*/
76  /*    make a data file                                         */
77  /*=============================================================*/
78  int mk_plot_data(char *a, double x[], double y[], int n){
79    int i;
80    FILE *out;
81
82    out = fopen(a, "w");
83
84    for(i=0; i<=n; i++){
85      fprintf(out, "%e\t%e\n", x[i], y[i]);
86    }
87
88    fclose(out);
89
90    return 0;
91  }
92
93  /*=============================================================*/
94  /*    make a graph                                             */
95  /*=============================================================*/
96  int mk_graph(char *f){
97    FILE *gp;
98
99    gp = popen("gnuplot -persist","w");
100
101   fprintf(gp, "reset\n");
102   fprintf(gp, "set terminal postscript eps color\n");
103   fprintf(gp, "set output \"graph.eps\"\n");
104   fprintf(gp, "set grid\n");
105
106
107   /* --------  plat graph ----------*/
108
```

```
109     fprintf(gp, "plot \"%s\" using 1:2 with line\n",f);
110     fprintf(gp, "set terminal x11\n");
111     fprintf(gp, "replot\n");
112
113     pclose(gp);
114
115     return 0;
116 }
```

1.2 プログラムの説明

このプログラムの大まかな構成は，

- main 関数は，計算条件の設定と 4 次のルンゲ・クッタ法による計算，グラフ作成のルーチンの呼び出しを行っている．
- 関数 func は，解くべき微分方程式 $dy/dx = f(x, y)$ の $f(x, y)$ の計算を行っている．
- 関数 mk_plot_data は，計算結果の x と y の配列の値を，ファイル (ハードディスク) に保管している．
- 関数 mk_graph は，ファイルからグラフを作成している．

である．

各行の役割は，以下の通りである．

2 行 数学関数を使っているので，math.h をインクルードしている．

3 行 計算に使う配列の大きさ IMAX を定義している．IMAX は最大計算ステップ数に 1 以上を加えた値にしなくてはならない．

4-7 行 プログラマーが作成した関数のプロトタイプ宣言．

11 行 main 関数の始まり．

12-16 行 main 関数で使う変数や配列の宣言．

21-28 行 メッセージを書き出し，計算に必要な初期値と終値を取り込む．

30-33 行 計算ステップ数の入力．ただし，配列の範囲を超えた場合，再度，入力を促すようになっている．

38 行 計算ステップのサイズ h の計算．

43-51 行 4 次のルンゲ・クッタ法の計算.

$$
\begin{aligned}
k_1 &= hf(x_i, y_i) && \Rightarrow \quad \texttt{k1=h*func(x[i],y[i])} \\
k_2 &= hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right) && \Rightarrow \quad \texttt{k3=h*func(x[i]+h/2.0, y[i]+k2/2.0)} \\
k_3 &= hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right) && \Rightarrow \quad \texttt{k2=h*func(x[i]+h/2.0, y[i]+k2/2.0)} \\
k_4 &= hf\left(x_i + h, y_i + k_3\right) && \Rightarrow \quad \texttt{k4=h*func(x[i]+h, y[i]+k3)} \\
x_{i+1} &= x_i + h && \Rightarrow \quad \texttt{x[i+1]=x[i]+h} \\
y_{i+1} &= y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) && \Rightarrow \quad \texttt{y[i+1]=y[i]+1.0/6.0*(k1+2.0*k2+2.0*k3+k4)}
\end{aligned}
$$

56 行 計算結果をファイルに格納する関数 (サブルーチン) の呼び出し.

57 行 ファイルに書かれたデータを gnuplot でグラフにする関数 (サブルーチン) の呼び出し.

59 行 `main` 関数の戻り値. このプログラムでは, 使われていない.

60 行 `main` 関数の終わり.

---

66 行 関数 func の始まり.

- 関数名は, `func`
- 入力引数は, 倍精度実数 `x` と倍精度実数 `y`
- 戻り値は, 倍精度実数

67 行 関数 `func` で使う変数宣言.

69 行 解くべき微分方程式 $dy/dx = f(x, y)$ の $f(x, y)$ の計算. 計算結果は, 変数 `dydx` に格納.

71 関数 `func` の戻り値を, 変数 `dydx` の値として与えている.

72 関数 `func` の終わり.

---

78 行 関数 mk_plot_data の始まり.

- 関数名は, `mk_plot_data`
- 入力引数は, 文字列のポインター `a`, 倍精度実数配列 `x` と y
- 戻り値は, 整数

79-80 行 関数 `mk_plot_data` で使う変数宣言. 型名 `FILE` は, ファイルを使うときに必要である.

82 行 ポインター `a` が示す文字列の名前で, 書き込みモードでファイルを開く. この後, ファイルは, `out` というポインターをつかってアクセスする.

84-86 行 ファイルへデータの書き込み.

88 行 ファイルを閉じる.

59 行 `mk_plot_data` 関数の戻り値. このプログラムでは, 使われていない.

60 行 mk_plot_data 関数の終わり．

---

96 行 関数 mk_graph の始まり．

- 関数名は，mk_graph
- 入力引数は，文字列のポインター a．これは，データファイル名を表す．
- 戻り値は，整数

97 行 関数 mk_graph で使う変数宣言．型名 FILE は，パイプを使うときに必要である．

99 行 書き込みモードでファイルでパイプを開く．これは，ターミナルで,「gnuplot -persist」と打ち込んだのと同じである．

101 -111 行 gnuplot にダブルクォーテションで囲んだ命令を与えている．

113 行 パイプを閉じている．

115 行 mk_graph 関数の戻り値．このプログラムでは，使われていない．

116 行 mk_graph 関数の終わり．