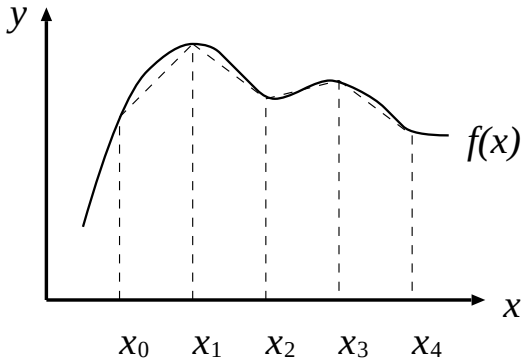
数値計算法概論：No.5 数値積分 (台形公式)

1 数値積分 (台形公式)

微分とは異なり、一般に積分は解析的には求まらないので、数値計算に頼る必要がでてくる。特異点のない場合、次の (閉じた) 台形公式により、数値積分ができる。

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{i=0}^{i=N-1} h(f(x_i)+f(x_{i+1}))/2 + O\left(\frac{(b-a)^3 f''}{N^2}\right) \\ &= h\left[\frac{1}{2}f(a) + \sum_{i=1}^{i=N-1} f(x_i) + \frac{1}{2}f(b)\right] + O\left(\frac{(b-a)^3 f''}{N^2}\right), \\ x_i &= a+hi, \quad h \equiv \frac{(b-a)}{N} \end{aligned} \tag{1}$$



[解説] 積分区間 $[a,b]$ を N 等分し $x_0 = a, x_1, x_2, \cdots, x_N = b$ とし、、各点での関数値を $f(x_0), f(x_1), \cdots, f(x_N)$ で表す。図のように関数 $f(x)$ を折れ線で近似すると、各台形の面積は、$h(f(x_i)+f(x_{i+1}))/2$ で、(積分値 $\approx$ 台形の総面積) から上の台形公式が導ける。

課題 1：次の問題を数値積分して見よ。分割数 N(刻み幅 h) に対する収束性を確認せよ。

$$\int_0^{\pi} \sin(x)dx = 2 \tag{2}$$

$$\int_0^1 e^x dx = e - 1 = 1.718281828\cdots \tag{3}$$

$$\int_{-1}^1 \frac{2}{1+x^2}dx = \pi = 3.141592653589793\cdots \tag{4}$$

$$\int_0^1 \exp(-x^2)dx \tag{5}$$

$$4\int_0^1 \sqrt{1-x^2}dx = \pi \tag{6}$$

2 プログラム

2.1 関数副プログラム

プログラムが複雑になると、プログラム全体を１つの単位としておくのが難しくなるので、プログラムを機能ごとの単位に分割し、それらを組み合わせていけばよい。こうすると、プログラム作成、デバック (エラー修正) が容易になるばかりでなく、過去のプログラムの蓄積が利用できる。そのようなものとしてまず関数副プログラムをあげる。

1) 関数副プログラムの定義：
[t] function $f(d_1, d_2, ...)$
.....
f= ...
return
end
t : 型宣言子 f : 関数名 d_i : 仮引数
2) 引用法：
$f(a_1, a_2, ...)$
f : 関数名 a_i : 実引数

注意 1: 仮引数の有効範囲は関数副プログラムの中だけである。したがって仮引数と同じ名前の変数をプログラム中の他の箇所で別の意味に使用することもできる。
注意 2: 実引数と仮引数の型は一致していなければならない。一方が単精度で他方が倍精度だったりすると、全く異なる結果になる。仮引数の宣言は関数副プログラムの中で行なう。
例：

```
c     Main routine
      implicit real*8 (a-h,o-z)
c
      read(*,*) a,b,n
      h=(b-a)/n
      sum=0.5*(func(a)+func(b))
      do i=1,n-1
         x=a+h*i
         sum = sum + func(x)
      end do
```

```
      write(*,*) sum*h
c
      end
c
c     Function subprogram
      real*8 function func(x)
c
      implicit real*8 (a-h,o-z)
c
      func=cos(x)
c
      return
      end
```

2.2 配列

複数個のデータをセットとして取り扱うと見通しが良くなることがある。例えば線形代数でのベクトルや、行列などである。コンピューター上で同じ機能を果たすものが配列である。配列は、複数個の同じ種類のデータを取り扱う形式である。

配列の宣言
型宣言子 $a_1(l_1:u_1), a_2(l_2:u_2, l_3:u_3), \cdots$
または
dimension $a_1(l_1:u_1), a_2(l_2:u_2, l_3:u_3), \cdots$
ただし、a_i : 配列名 l_i : 添字の下限 u_i : 添字の上限

配列は integer,real 等の型宣言子あるいは dimension 文で宣言する。配列の次元数は添字の上下限の組の数に等しく、$a_1(l_1:u_1)$ は 1 次元配列、$a_2(l_2:u_2, l_3:u_3)$ は 2 次元配列である。添字の下限が 1 の時は、下限を省略し $a_1(u_1), a_2(u_2, u_3)$ の様に書ける。

```
      implicit real*8 (a-h,o-z)
      dimension a(10),k(10)
c
      do i=1,5
         k(i)=2*i+1
         a(i)=1/dble(3*i-1)
      end do
c
      write(*,*) (k(i),i=1,5)
      write(*,*) (a(i),i=1,5)
c
      end
```

なお、ここで dble() は整数型を実数型に変換する組み込み関数である。

2.3 サブルーチン

関数副プログラムは出力できる値が１つに限られていた。サブルーチン副プログラムを使うと、複数の値を出力できる。また、配列の引渡しなども自由に行なえる。

1) サブルーチンの定義
subroutine $s(d_1, d_2, ...)$
.....
return
end
s : サブルーチン名 d_i : 引数

2) 引用法：
call $s(a_1, a_2, ...)$
s : サブルーチン名 a_i : 実引数

引数には変数名の他、配列名も書くことができる。

3 一般的注意

3.1 ディレクトリ及びファイルシステム (利用の手引 8.4.1)

ファイルを沢山作ると整理が大変になる。通常の事務作業と同様に、目的ごとに 1 連のファイルをまとめておくと整理し易くなる。これはディレクトリを使うと容易にできる。ディレクトリを作るには mkdir, ディレクトリを移動するには cd というコマンドを使う。また、現在どのディレクトリにいるのか確認するためには pwd というコマンドを使う。

```
# ls
test1.f test2.f test20.f test21.f
# mkdir task1
# cp test20.f task1
# cp test21.f task1
# pwd
# cd task1
# ls
test20.f test21.f
# cd ..
```