ボーズ - アインシュタイン積分の数値計算

2016.10.7　　鈴木　実

1　はじめに

フェルミ - ディラック分布関数を含むフェルミ - ディラック積分と同様にボーズ - アインシュタイン分布関数にもボーズ - アインシュタイン積分が存在し，ボーズ - アインシュタイン分布関数を含む計算に頻繁に登場するだけでなく，しばしば数値計算を強いられる．そこでここでは，ボーズ - アインシュタイン積分の数値計算の C プログラムを載せておくことにする．関数の形に書いてあるので，種々の計算にそのまま使用することができる．

2　ボーズ - アインシュタイン積分

ボーズ-アインシュタイン分布関数を含む次の積分をボーズ-アインシュタイン積分という．

$$F(s,\alpha)=\frac{1}{\Gamma(s)}\int_0^\infty \frac{x^{1-s}}{\mathrm{e}^{x+\alpha}-1}\mathrm{d}x \tag{1}$$

ここで，$\alpha\geq 0$ である．この式は次のように展開式に変形できる．

$$\begin{aligned}F(s,\alpha)&=\frac{1}{\Gamma(s)}\sum_{n=1}^\infty\int_0^\infty\left(\frac{1}{n}\right)^s t^{s-1}\mathrm{e}^{-t-n\alpha}\mathrm{d}t=\frac{1}{\Gamma(s)}\sum_{n=1}^\infty\Gamma(s)\left(\frac{1}{n}\right)^s\mathrm{e}^{-n\alpha}\\&=\sum_{n=1}^\infty n^{-s}\mathrm{e}^{-n\alpha}\end{aligned} \tag{2}$$

この展開式は $\alpha\simeq 0$ で収束が非常に遅くなる．そこで，$0\leq\alpha<1$ では Robinson [1] が報告した次の収束の速い展開式を用いる．

$$F(s,\alpha)=\Gamma(1-s)\alpha^{s-1}+\sum_{n=0}^\infty\frac{(-1)^n}{n!}\zeta(s-n)\alpha^n \tag{3}$$

Robinson の展開式にはリーマンのツェータ関数$\zeta(z)$ が含まれている．通常ツェータ関数の定義に用いられる無限級数は収束が極めて遅い．そこで，次に示す Hasse [2] の展開式を用いる．この展開式は $s=1$ を除く複素平面上の全ての点で速く収束する．

$$\zeta(s)=\frac{1}{1-2^{1-s}}\sum_{n=0}^\infty\frac{1}{2^{n+1}}\sum_{k=0}^n(-1)^k\binom{n}{k}(k+1)^{-s} \tag{4}$$

参考文献

[1] J. E. Robinson, “Note on the Bose-Einstein Integral Functions”, Phys. Rev. **83**, 678-679 (1951).
[2] Wolfram MathWorld, http://mathworld.wolfram.com/RiemannZetaFunction.html

プログラムソース

1. ボーズ-アインシュタイン積分の数値計算プログラム

```
/* bose_einstein_integral_f.c  -o  be_int_f */
/* 2016.10.7 by M. Suzuki */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define    NMAX    50
#define    NMAX1    200
#define    NMAX2    10
int factorial(int n)
{
    int i;
    if(n==0) return 1;
    i=1;
    while(n>0)
    {
        i*=n;
        n--;
    }
    return i;
}

double binom(int n, int k)
{
    int i;
    double x, z;
    if(k==0) return z=n;
    z=1;
    i=k;
    while(k>0)
    {
        z*=n;
        n--;
        k--;
        if(n==0) n=1;
    }
    while(i>0)
    {
        z/=i;
        i--;
    }
    return z;
}

double zeta(double s)
{
    double t, x, y, z;
    int i, n, k;

    if(s==1) exit(0);
```

```
    z=0;
    for(n=0; n<NMAX+1; n++)
    {
        t=1/pow(2.0, n+1);
        y=0;
        for(k=0;k<n+1;k++)
        {
            x=pow(k+1.0, -s)*binom(n, k);
            if((k % 2)==1) x*=-1;
            y+=x;
        }
        t*=y;
        z+=t;
    }
    z/=(1-pow(2.0, 1-s));
    return z;
}

double BE_integral(double s, double alpha)
{
    int i;
    double t, y, z;

    if(alpha > 1.0)
    {
        z=0;
        for(i=1;i<NMAX1;i++)
        {
            t=i;
            y=pow(t, -s)*exp(-i*alpha);
            z+=y;
        }
    }
    else if(alpha==0) z=zeta(s);
    else
    {
        z=0;
        for(i=0;i<NMAX2;i++)
        {
            t=i;
            y=zeta(s-i)*pow(alpha, t);
            y/=factorial(i);
            if((i % 2)==1) y*=-1;
            z+=y;
        }
        z+=tgamma(1-s)*pow(alpha, s-1);
    }
    return z;
}

int main(int argc, char *argv[])
{
    double alpha, s, z;

    if(argc<3)
    {
        printf("!\n!  Usage - a.out [alpha] [s] \n!\n");
```

```
        exit(0);
    }

    s=atof(argv[1]);
    alpha=atof(argv[2]);
    z=BE_integral(s, alpha);
    printf("%lf\t%lf\t%16.12lf\n", alpha, s, z);
}
```