計算物理学 第一単元レポート

61908697 佐々木良輔

2-A.

プログラムについて

図 1 に PAD 図を示す. PAD 図は PadTools (`https://naoblo.net/misc/padtools/`) を用いて作図した. このプログラムでは double precision 型の OnePlusEpsilon 変数に $1 + 2^{-n}$ を順次代入し, これが 1 と一致したときにループを抜ける. このとき $2^{-(n-1)}$ が計算機イプシロンになる. ソースコード A.1 に fortran によるソースコードを示す.

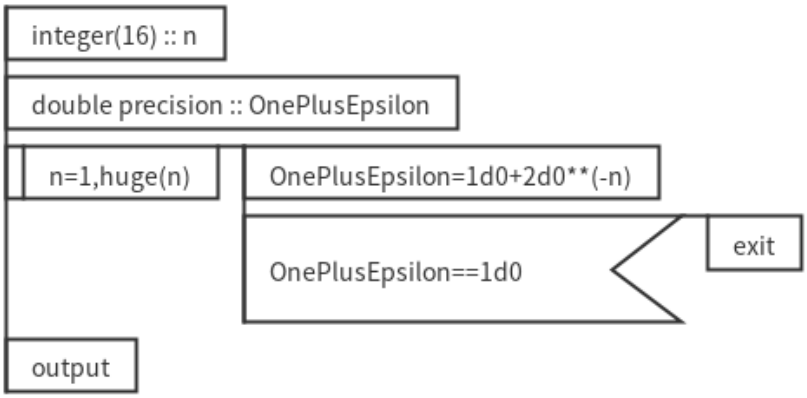

図 1　2-A. の PAD 図

出力結果

実行すると $n = 52$, $\varepsilon = 2.2 \times 10^{-16}$ という結果を得た. また OnePlusEpsilon を real 型に変えると $n = 23$, $\varepsilon = 1.2 \times 10^{-7}$ となった. これらは倍精度の仮数部が 53 bit, 単精度の仮数部が 24 bit であることとも整合し, プログラムが正常に動作していると考えられる.[2]

2-B.

公式の無次元化

各公式を無次元数 $\tilde{\omega} = \beta\hbar\omega$ で無次元化する. まず Planck の公式について

$$\begin{aligned} u(\omega, T) &= \frac{\omega^2}{\pi^2 c^3} \frac{\hbar\omega}{\mathrm{e}^{\beta\hbar\omega} - 1} \\ &= \frac{1}{\pi^2 c^3} \frac{\tilde{\omega^2}}{\beta^2\hbar^2} \frac{\tilde{\omega}}{\beta(\mathrm{e}^{\tilde{\omega}} - 1)} \end{aligned}$$

$$f_{\mathrm{planck}}(\tilde{\omega}) := \beta^3\pi^2\hbar^2c^3u(\omega, T) = \frac{\tilde{\omega}^3}{\mathrm{e}^{\tilde{\omega}} - 1}$$

同様に Rayleigh-Jeans の公式, Wien の公式について

$$\begin{aligned} f_{\mathrm{rayleigh-jeans}}(\tilde{\omega}) &:= \tilde{\omega}^2 \\ f_{\mathrm{wien}}(\tilde{\omega}) &:= \tilde{\omega}^3\mathrm{e}^{-\tilde{\omega}} \end{aligned}$$

として無次元化できた.

プログラムについて

図 2 に PAD 図を示す. このプログラムでは x_start から x_end までを num_samples 分割し, それぞれの点での planck(x), rayleigh_jeans(x), wien(x) 関数の値を出力している. これらの関数はそれぞれ上で無次元化した Planck の公式, Rayleigh-Jeans の公式, Wien の公式の値を返す関数である. ソースコード A.2 に fortran によるソースコードを示す.

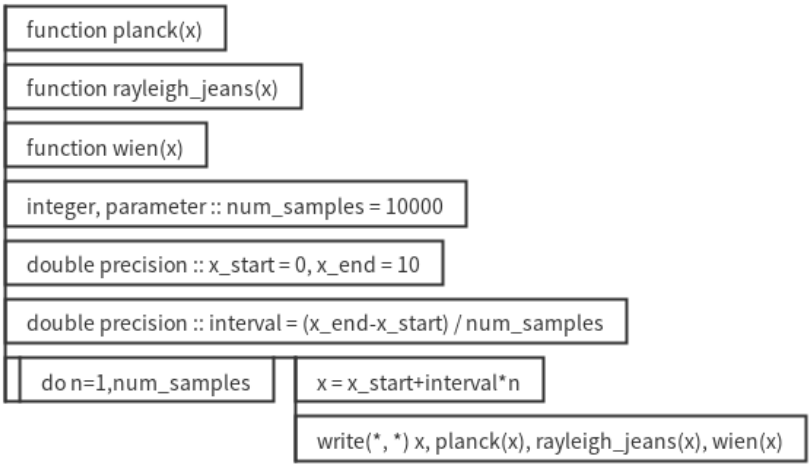

図 2　2-B. の PAD 図

出力結果

図 3 にプロットしたグラフを示す．Rayleigh-Jeans の公式, Wien の公式はそれぞれ Planck の公式の低周波数, 高周波数での近似である．図 3 ではたしかにそれぞれの公式が低周波数, 高周波数での Planck の公式をよく近似していることがわかる．

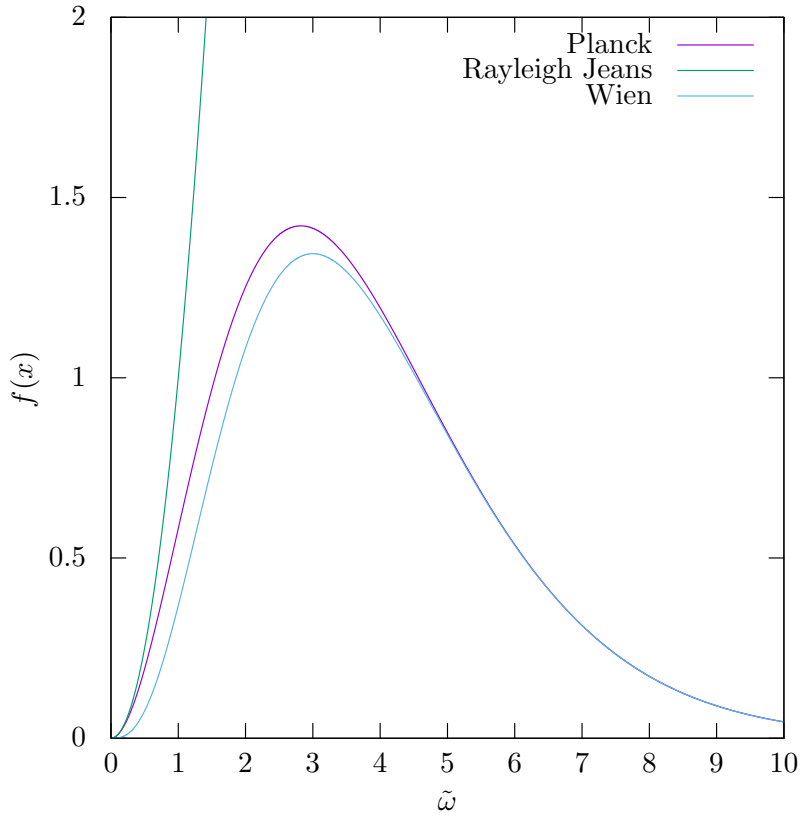

図 3　出力結果

2-C.

公式の無次元化

各公式を無次元数 $\tau = T/\Theta_D$ で無次元化する．まず Debye の公式は

$$\begin{aligned} C_V &= 3Nk_B \cdot 3\left(\frac{T}{\Theta_D}\right)^3 \int_0^{\Theta_D/T} \frac{x^4 \mathrm{e}^x}{(\mathrm{e}^x-1)^2}\mathrm{d}x \\ &= 3Nk_B \cdot 3\tau^3 \int_0^{1/\tau} \frac{x^4 \mathrm{e}^x}{(\mathrm{e}^x-1)^2}\mathrm{d}x \end{aligned}$$

$$f_{\mathrm{debye}}(\tau) := \frac{C_V}{3Nk_B} = 3\tau^3 \int_0^{1/\tau} \frac{x^4 \mathrm{e}^x}{(\mathrm{e}^x - 1)^2} \mathrm{d}x$$

同様に Dulong-Petit の公式及び低温での式も無次元化すると

$$f_{\mathrm{dulong-petit}}(\tau) := 1$$
$$f_{\mathrm{lowtemp}}(\tau) := \frac{4}{5}\pi^4 \tau^3$$

として無次元化できた.

プログラムについて

図 4 に PAD 図を示す. このプログラムでは 1 つ目のループで 2-B. と同様に x_start から x_end までを num_samples 分割し, それぞれの点で debye(x), dulong_petit(x), low_temperature_behavior(x) 関数の値を出力している. それぞれの関数は上で無次元化した各公式である. また数値積分には simpson(arg, a, b, n) 関数を用いており, arg(x) の区間 [a, b] を n 分割してシンプソン則により積分を実行している. また 2 つ目のループではシンプソン則の分割数 n を変えつつ誤差を記録した. また被積分関数は 0 近傍で zero divison が発生するが, テイラー展開すると 0 近傍で 0 になるため NaN になる場合は関数値として 0 を返している. ソースコード A.3 に fortran によるソースコードを示す. 各関数と積分関数はモジュール化されており, 積分関数は被積分関数を引数として受けている.

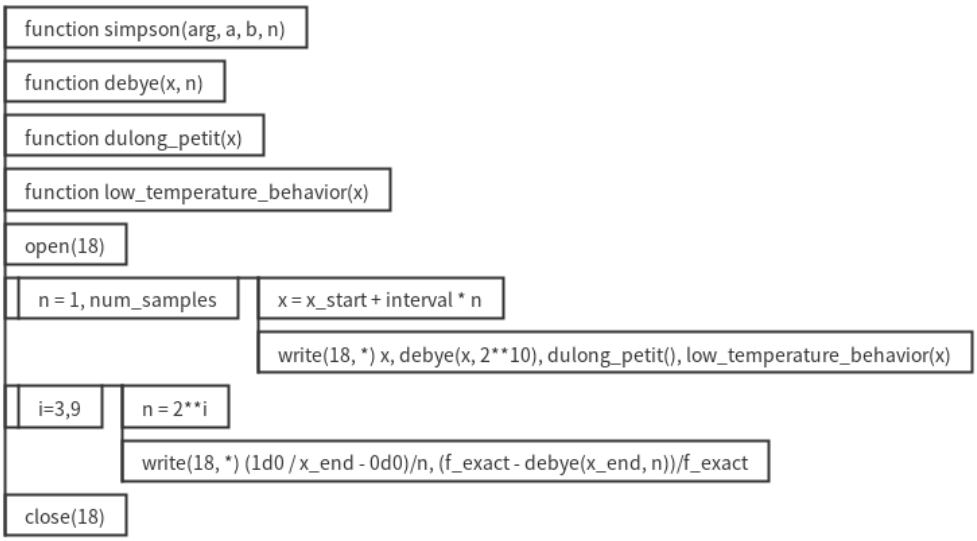

図 4　2-C. の PAD 図

出力結果

図 5 に出力結果を示す. この図からたしかに Dulong-Petit の公式や低温での振る舞いを示す公式は高温域及び低温域で Debye の公式をよく近似していることがわかる. すなわち定積比熱は高温域で一定値 ($= 3Nk_B$), 低温域では 3 次関数 ($3Nk_B \cdot 4\pi^4/5(T/\Theta_D)^3$) のように振る舞う.

また図 6 に刻み幅 h の 4 乗 (h^4) と誤差の関係を示す. 図からシンプソン則の誤差は刻み幅の 4 乗に比例することがわかる. また [1] によるとシンプソン則はその導出過程からも確かに h^4 の精度であり, 妥当な結果だと言える.

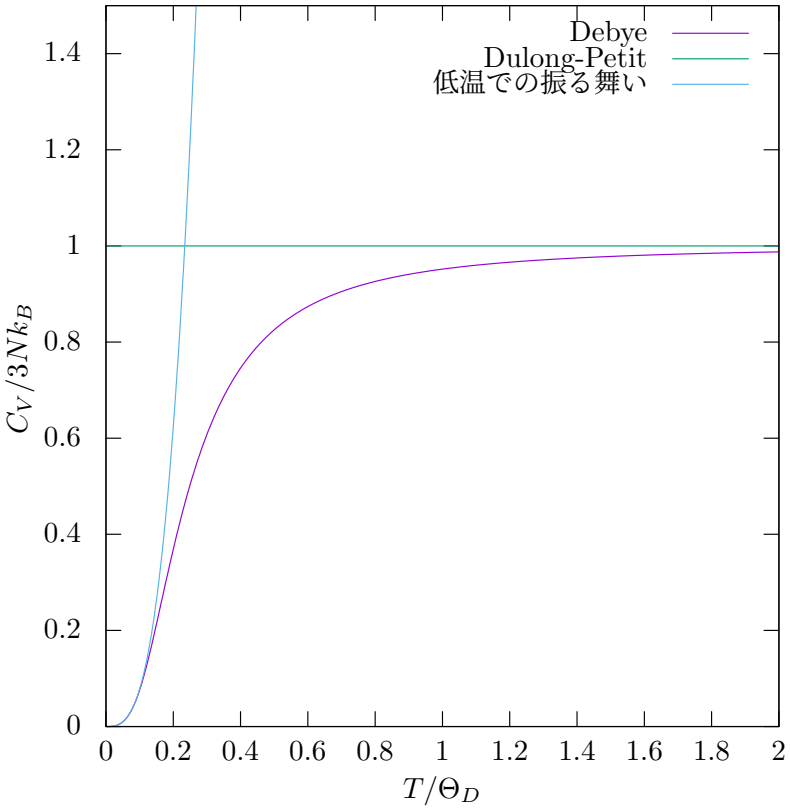

図 5　出力結果

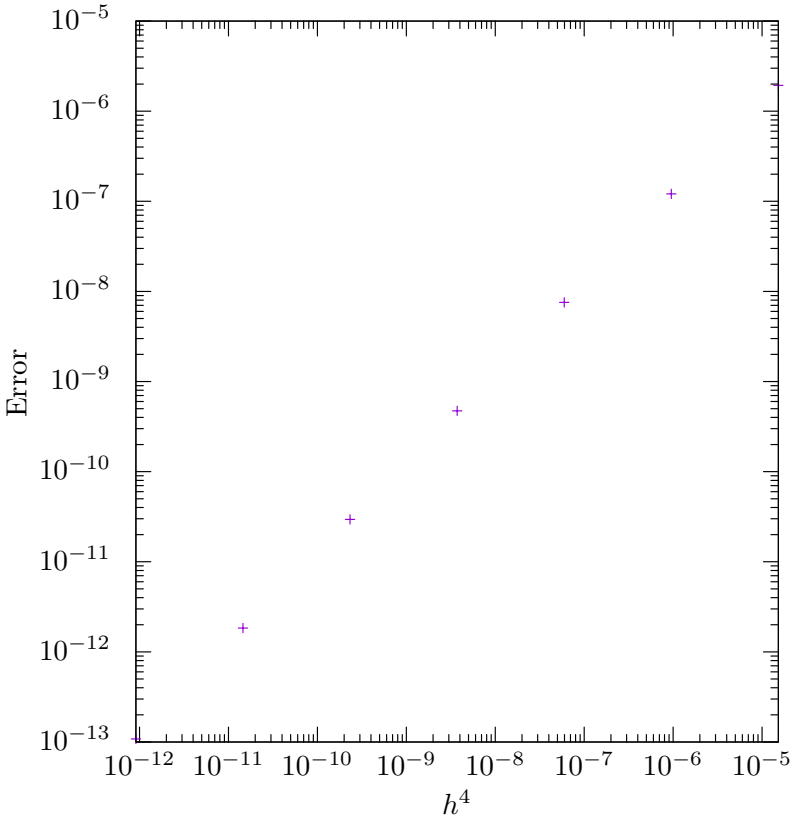
Error
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-11}
10^{-12}
10^{-13}
10^{-12} 10^{-11} 10^{-10} 10^{-9} 10^{-8} 10^{-7} 10^{-6} 10^{-5}
h^4

図 6　h^4 と誤差の関係

参考文献

[1] 東京都立大学海岸・海洋工学研究室. シンプソン公式. https://www.comp.tmu.ac.jp/shintani/classes/information_processing_2/integration_2/simpson.html. (Accessed on 10/11/2021).

[2] 電気通信大学情報基盤センター. 浮動小数点数の表現と誤差 - floating point number. http://www.edu.cc.uec.ac.jp/~ta113003/htsecure/ref/FloatingPoint.html. (Accessed on 10/10/2021).

付録 A　ソースコード

ソースコード A.1　2-A のソースコード

```
1  program main
2    integer(16) :: n
3    double precision :: OnePlusEpsilon
4    !real :: OnePlusEpsilon
5    do n = 1, huge(n)
6      OnePlusEpsilon = 1d0 + 2d0**(-n)
7      if (OnePlusEpsilon == 1d0) exit
8    end do
9
10   write(*, '(a)', advance='no') "n = "
11   write(*, '(i0)', advance='no') (n - 1)
12   write(*, '(a)', advance='no') ", epsilon = "
13   write(*, '(E8.2)') (2d0**(-(n - 1)))
14
15 end
```

ソースコード A.2　2-B のソースコード

```
program main
  implicit none
  integer :: n
  integer, parameter :: num_samples = 10000
  double precision :: x
  double precision, parameter :: x_start = 0, x_end = 10
  double precision, parameter :: interval = (x_end - x_start) / num_samples

  do n = 1, num_samples
      x = x_start + interval * n
      write(*, *) x, planck(x), rayleigh_jeans(x), wien(x)
  end do

  contains
  function planck(x)
      implicit none
      double precision :: planck, x
      planck = x**3d0 / (exp(x) - 1)
  end

  function rayleigh_jeans(x)
      implicit none
      double precision :: rayleigh_jeans, x
      rayleigh_jeans = x**2d0
  end

  function wien(x)
      implicit none
      double precision :: wien, x
      wien = x**3d0 * exp(-x)
  end
end
```

ソースコード A.3　2-C のソースコード

```
1  module integral
2    implicit none
3
4    contains
5    function simpson(arg, a, b, n)
6        implicit none
7        interface
8            function arg(x)
9                double precision :: arg
10               double precision, intent(in) :: x
11           end function arg
12       end interface
13       double precision :: simpson, h, sum, x
14       double precision, intent(in) :: a, b
15       integer :: i
16       integer, intent(in) :: n
17       h = (b - a) / n
18
19       sum = arg(a) + arg(b)
20       do i = 1, n - 1, 2
21           x = a + i * h
22           sum = sum + 4d0*arg(x)
23       end do
24       do i = 2, n - 1, 2
25           x = a + i*h
26           sum = sum + 2d0*arg(x)
27       end do
28       simpson = sum*h / 3d0
29   end function simpson
30 end module integral
31
32 module funcs
33   use integral
34   implicit none
35   double precision, parameter :: pi = 3.14159265358979323846264
36
37   contains
38   function debye(x, n)
39       implicit none
40       double precision :: debye
41       double precision, intent(in) :: x
```

```
42        integer, intent(in) :: n
43
44        debye = 3d0*x**3d0*simpson(integrand, 0d0, 1d0 / x, n)
45
46        contains
47        function integrand(x)
48            implicit none
49            double precision :: integrand, f
50            double precision, intent(in) :: x
51            f = x**4d0*exp(x) / (exp(x) - 1)**2d0
52            if (isnan(f)) then
53                integrand = 0
54            else
55                integrand = f
56            end if
57        end function integrand
58    end function debye
59
60    function dulong_petit()
61        implicit none
62        double precision :: dulong_petit
63        dulong_petit = 1d0
64    end function dulong_petit
65
66    function low_temperature_behavior(x)
67        implicit none
68        double precision :: low_temperature_behavior
69        double precision, intent(in) :: x
70        low_temperature_behavior = 4d0*pi**4d0*x**3d0 / 5d0
71    end function low_temperature_behavior
72  end module funcs
73
74  program main
75    use funcs
76    implicit none
77    integer :: n, i
78    integer, parameter :: num_samples = 1000
79    double precision :: x, f_exact, f
80    double precision, parameter :: x_start = 0d0, x_end = 2d0
81    double precision, parameter :: interval = (x_end - x_start) / num_samples
82    open(18, file='output2c-1', status='replace')
83    do n = 1, num_samples
```

```
84        x = x_start + interval * n
85        write(18, *) x, debye(x, 2**10), dulong_petit(),
              low_temperature_behavior(x)
86    end do
87    close(18)
88    open(18, file='output2c-2', status='replace')
89    f_exact = debye(x_end, 2**10)
90    do i = 3, 9
91        n = 2**i
92        write(18, *) (1d0 / x_end - 0d0)/n, (f_exact - debye(x_end, n))/
              f_exact
93    end do
94    close(18)
95
96  end program main
```