Astroinformatics 2022
Session 09: Blackbody Radiation

Kinoshita Daisuke

14 November 2022
publicly accessible version

About this file...

- Important information about this file
  - The author of this file is Kinoshita Daisuke.
  - The original version of this file was used for the course "Astroinformatics" (course ID: AS6095) offered at Institute of Astronomy, National Central University from September 2022 to January 2023.
  - The file is provided in the hope that it will be useful, but there is no guarantee for the correctness. Use this file at your own risk.
  - If you are willing to use this file for your study, please feel free to use. I'll be very happy to receive feedback from you.
  - If you are willing to use this file for your teaching, please contact to Kinoshita Daisuke. When you use this file partly or entirely, please mention clearly that the author of the original version is Kinoshita Daisuke. Please help me to improve the contents of this file by sending your feedback.
  - Contact address: `https://www.instagram.com/daisuke23888/`

For this session, we study blackbody radiation.

1 Sample Python scripts for this session

Sample Python scripts for this session can be downloaded from GitHub repository. Visit following GitHub repository.

- `https://github.com/kinoshitadaisuke/ncu_astroinformatics_202209`

1.1 Executing sample Python scripts on a terminal emulator

If you prefer to execute sample Python scripts for this session on a terminal emulator, download `.py` files from GitHub repository.

1.2 Executing sample Python scripts on JupyterLab

If you prefer to execute sample Python scripts for this session on JupyterLab (or Jupyter Notebook), download `.ipynb` file from GitHub repository.

1.3 Executing sample Python scripts using Binder

If you prefer to execute sample Python scripts for this session on Binder, visit following web page.

- `https://mybinder.org/v2/gh/kinoshitadaisuke/ncu_astroinformatics_202209/HEAD`

Start your favourite web browser and go to above web page. (Fig. 1) In a minute or two, you see JupyterLab working on your web browser. (Fig. 2) Go to the directory (folder) "`s09`". (Fig. 3) Choose the file "`ai202209_s09.ipynb`" (Fig. 4 and 5) and open it (Fig. 6).

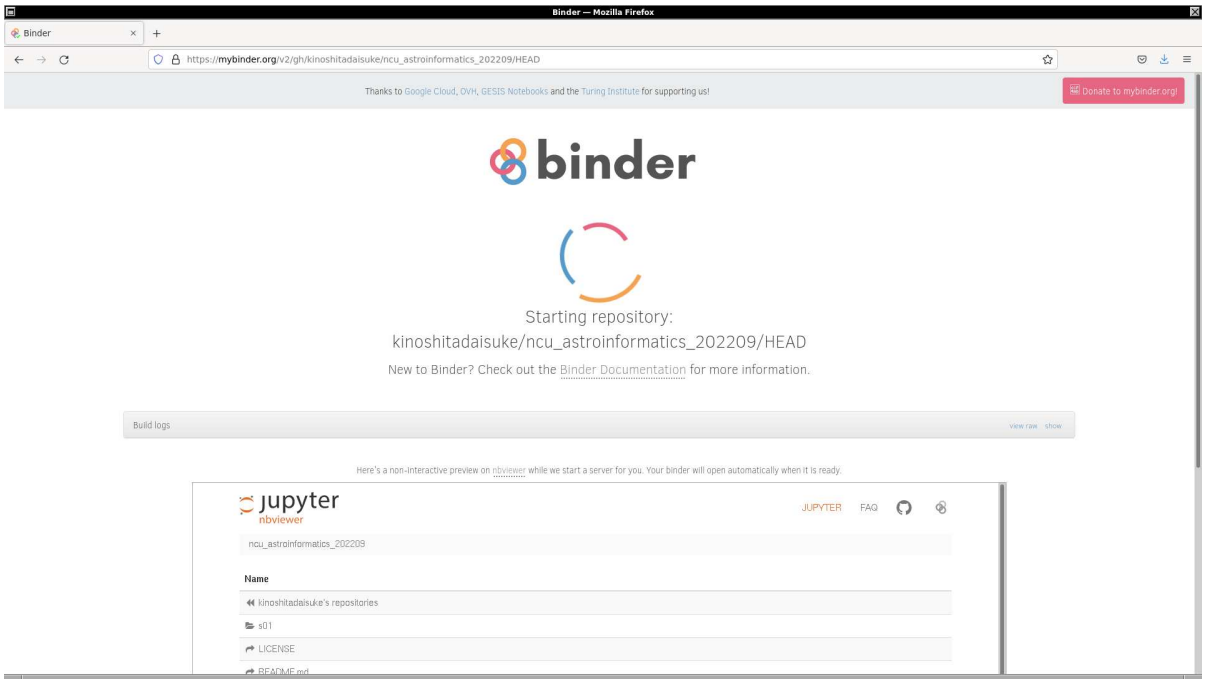

Figure 1: Using Binder to execute sample Python scripts for this session.

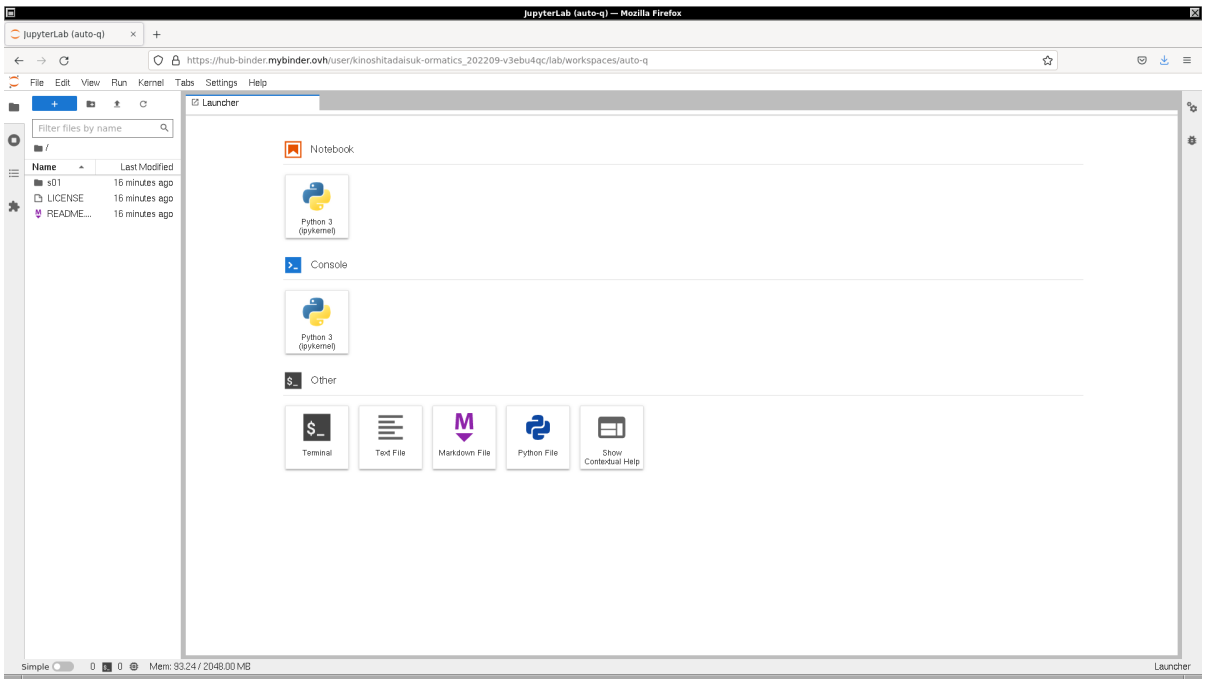

Figure 2: Using Binder to execute sample Python scripts for this session.

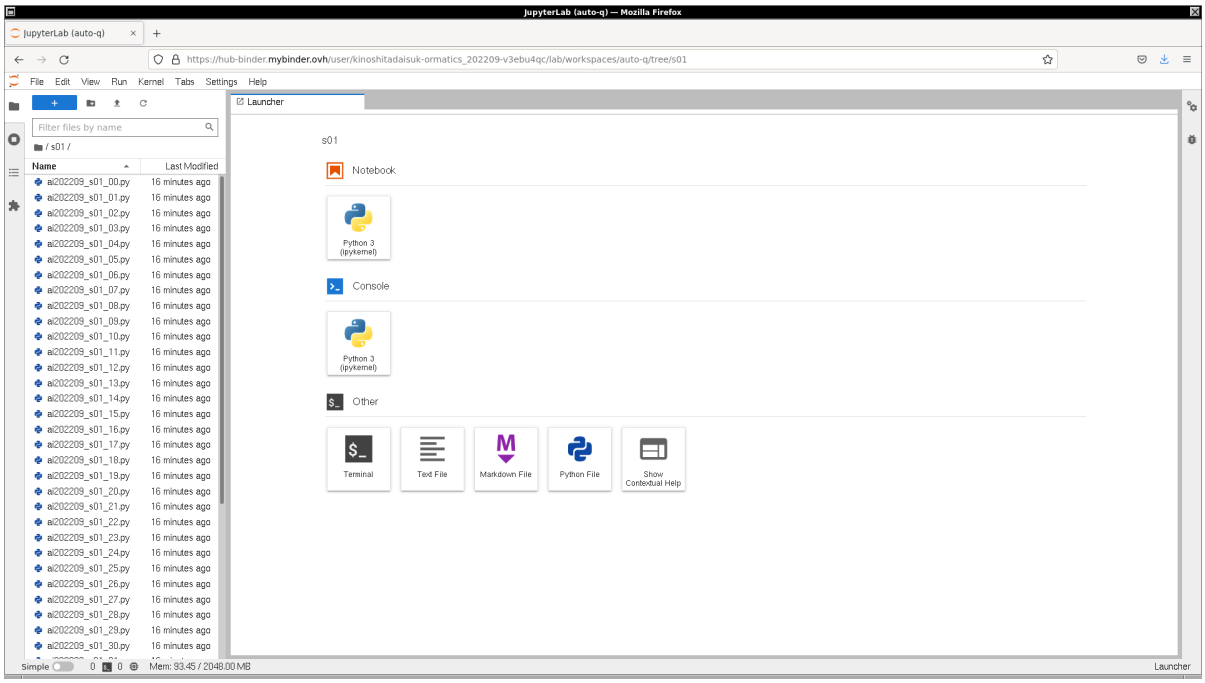

Figure 3: Using Binder to execute sample Python scripts for this session.

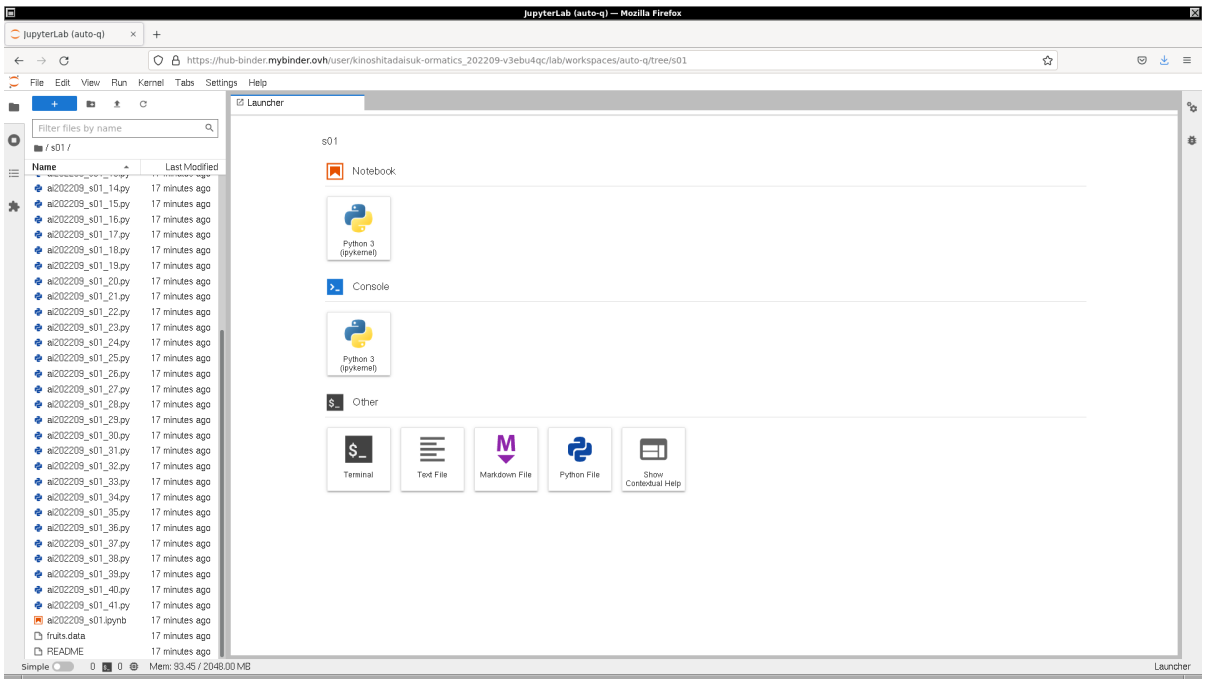

Figure 4: Using Binder to execute sample Python scripts for this session.

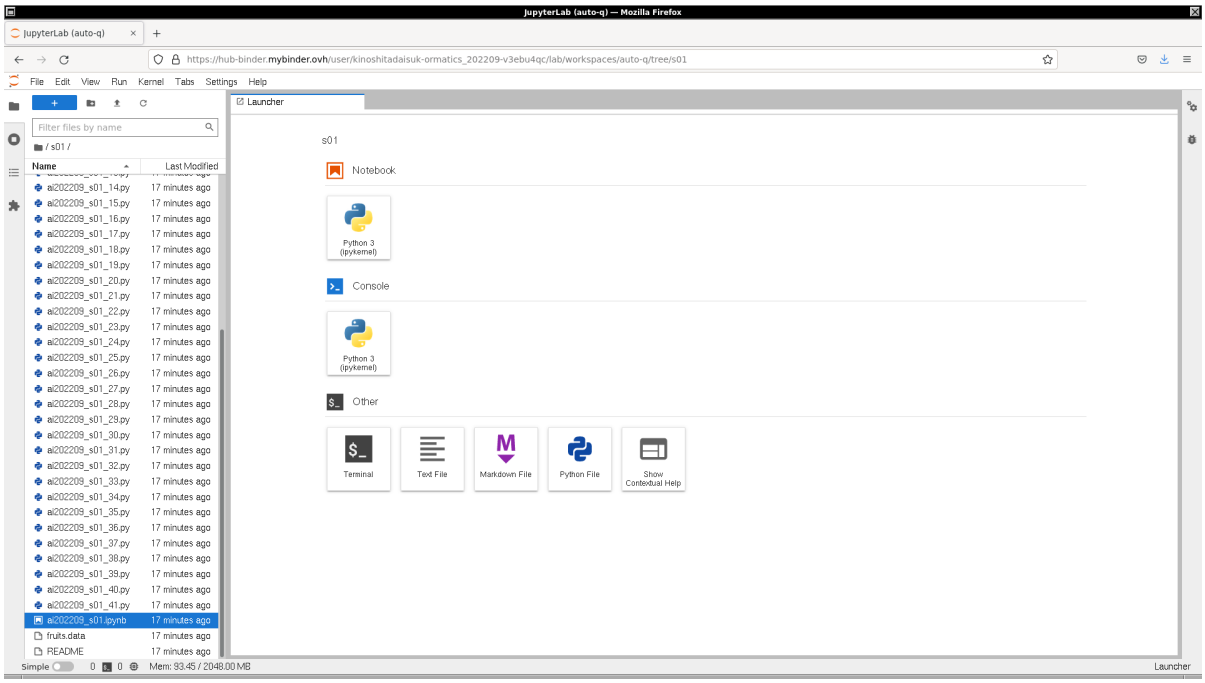

Figure 5: Using Binder to execute sample Python scripts for this session.

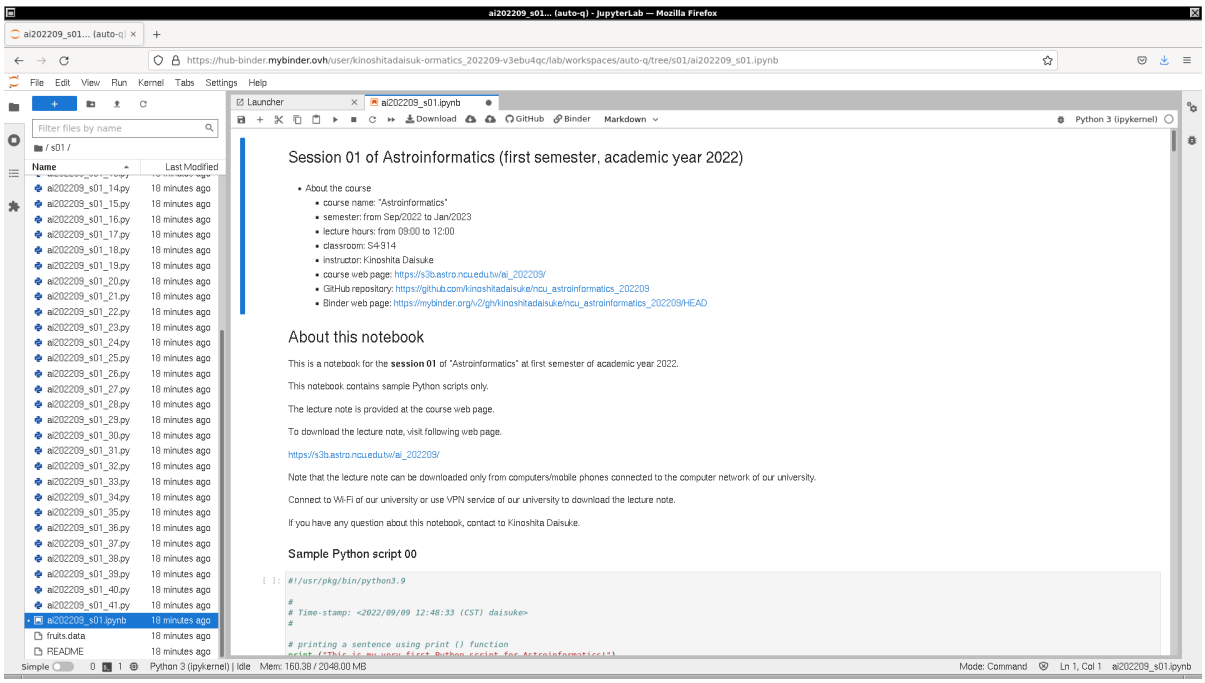

Figure 6: Using Binder to execute sample Python scripts for this session.

2 Blackbody spectrum of $T = 5800$ K

2.1 Physical constants for blackbody calculation

Get values and units of the speed of light in vacuum, Planck constant, and Boltzmann constant using SciPy.

Python Code 1: ai202209_s09_00_00.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/10 15:37:36 (CST) daisuke>
#

# importing scipy module
import scipy.constants

#
# constants
#

# speed of light
c = scipy.constants.physical_constants['speed of light in vacuum']

# Planck constant
h = scipy.constants.physical_constants['Planck constant']

# Boltzmann constant
k = scipy.constants.physical_constants['Boltzmann constant']

# printing values of constants
print (f'Constants:')
print (f'  c = {c[0]:g} [{c[1]}]')
print (f'  h = {h[0]:g} [{h[1]}]')
print (f'  k = {k[0]:g} [{k[1]}]')
```

Execute above script to check values and units of constants.

```
% chmod a+x ai202209_s09_00_00.py
% ./ai202209_s09_00_00.py
Constants:
  c = 2.99792e+08 [m s^-1]
  h = 6.62607e-34 [J Hz^-1]
  k = 1.38065e-23 [J K^-1]
```

Try following practice.

Practice 09-01

Obtain the value of gravitational constant using SciPy and print it.

2.2 Planck's radiation law

Use Planck's radiation law to construct a model spectrum of $T = 5800$ K blackbody.

Python Code 2: ai202209_s09_00_01.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/10 15:50:55 (CST) daisuke>
```

```
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

#
# constants
#

# speed of light
c = scipy.constants.physical_constants['speed of light in vacuum']

# Planck constant
h = scipy.constants.physical_constants['Planck constant']

# Boltzmann constant
k = scipy.constants.physical_constants['Boltzmann constant']

# printing values of constants
print (f'Constants:')
print (f'  c = {c[0]:g} [{c[1]}]')
print (f'  h = {h[0]:g} [{h[1]}]')
print (f'  k = {k[0]:g} [{k[1]}]')

# temperature of blackbody
T = 5800.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of wavelength (from 10**-8 m = 10 nm to 10**-3 m = 1 mm)
wavelength_min = -8.0
wavelength_max = -3.0

# wavelength in metre
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001)

# calculation of Planck function
blackbody = 2.0 * h[0] * c[0]**2 / wavelength**5 \
    / (numpy.exp (h[0] * c[0] / (wavelength * k[0] * T) ) - 1.0 )

# printing Planck function
print (f'Wavelength:')
print (f'{wavelength}')
print (f'Planck function:')
print (f'{blackbody}')
```

Execute above script to make a model spectrum of $T = 5800$ K blackbody.

```
% chmod a+x ai202209_s09_00_01.py
% ./ai202209_s09_00_01.py
Constants:
  c = 2.99792e+08 [m s^-1]
  h = 6.62607e-34 [J Hz^-1]
  k = 1.38065e-23 [J K^-1]
```

```
Temperature:
  T = 5800.0 K
Wavelength:
[1.00000000e-08 1.00230524e-08 1.00461579e-08 ... 9.95405417e-04
 9.97700064e-04 1.00000000e-03]
Planck function:
[2.20128336e-84 3.84995411e-84 6.72458175e-84 ... 4.88450670e+01
 4.83973915e+01 4.79538187e+01]
```

Try following practice.

Practice 09-02

Use Planck's radiation law to construct a model spectrum of $T = 25000$ K blackbody.

Define a function to calculate blackbody spectrum and construct a model spectrum of $T = 3000$ K blackbody.

Python Code 3: ai202209_s09_00_02.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 09:12:39 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

#
# function to calculate blackbody curve
#
def bb_lambda (wavelength, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * c[0]**2 / wavelength**5 \
        / (numpy.exp (h[0] * c[0] / (wavelength * k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# temperature of blackbody
T = 3000.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of wavelength (from 10**-8 m = 10 nm to 10**-3 m = 1 mm)
wavelength_min = -8.0
```

```
wavelength_max = -3.0

# wavelength in metre
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001)

# T = 3000 K blackbody spectrum
bb_3000 = bb_lambda (wavelength, T)

# printing Planck function
print (f'Wavelength:')
print (f'{wavelength}')
print (f'Planck function:')
print (f'{bb_3000}')
```

Execute above script to

```
% chmod a+x ai202209_s09_00_02.py
% ./ai202209_s09_00_02.py
Constants:
  c = 2.99792e+08 [m s^-1]
  h = 6.62607e-34 [J Hz^-1]
  k = 1.38065e-23 [J K^-1]
Temperature:
  T = 3000.0 K
Wavelength:
[1.00000000e-08 1.00230524e-08 1.00461579e-08 ... 9.95405417e-04
 9.97700064e-04 1.00000000e-03]
Planck function:
[6.18929573e-185 1.84366410e-184 5.47798201e-184 ... 2.52353067e+001
 2.50040867e+001 2.47749849e+001]
```

Try following practice.

Practice 09-03

Use Planck's radiation law to construct a model spectrum of $T = 300$ K blackbody.

2.3 Plotting a model spectrum of blackbody radiation

Use Planck's radiation law to construct a model spectrum of $T = 5800$ K blackbody and plot calculated model spectrum.

Python Code 4: ai202209_s09_00_03.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/11 13:26:19 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg
```

```
# output file name
file_output = 'ai202209_s09_00_03.png'

#
# function to calculate blackbody curve
#
def bb_lambda (wavelength, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * c[0]**2 / wavelength**5 \
        / (numpy.exp (h[0] * c[0] / (wavelength * k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# temperature of blackbody
T = 5800.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of wavelength (from 10**-8 m = 10 nm to 10**-3 m = 1 mm)
wavelength_min = -8.0
wavelength_max = -3.0

# wavelength in metre
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001)

# T = 5800 K blackbody spectrum
bb_5800 = bb_lambda (wavelength, T)

# printing Planck function
print (f'Wavelength:')
print (f'{wavelength}')
print (f'Planck function:')
print (f'{bb_5800}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [$\mu$m]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-3}$]')

# plotting data
ax.plot (wavelength * 10**6, bb_5800, linestyle='-', color='r', \
         linewidth=3, label='Blackbody of T = 5800 K')
```

```
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to

```
% chmod a+x ai202209_s09_00_03.py
% ./ai202209_s09_00_03.py
Constants:
  c = 2.99792e+08 [m s^-1]
  h = 6.62607e-34 [J Hz^-1]
  k = 1.38065e-23 [J K^-1]
Temperature:
  T = 5800.0 K
Wavelength:
[1.00000000e-08 1.00230524e-08 1.00461579e-08 ... 9.95405417e-04
 9.97700064e-04 1.00000000e-03]
Planck function:
[2.20128336e-84 3.84995411e-84 6.72458175e-84 ... 4.88450670e+01
 4.83973915e+01 4.79538187e+01]
```

Display generated PNG file. (7)

```
% feh -dF ai202209_s09_00_03.png
```

The plot does not look nice. Adjust the range of wavelength, and re-plot the blackbody radiation curve for $T = 5800$ K.

Python Code 5: ai202209_s09_00_04.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/11 13:26:39 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_04.png'

#
# function to calculate blackbody curve
#
def bb_lambda (wavelength, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
```

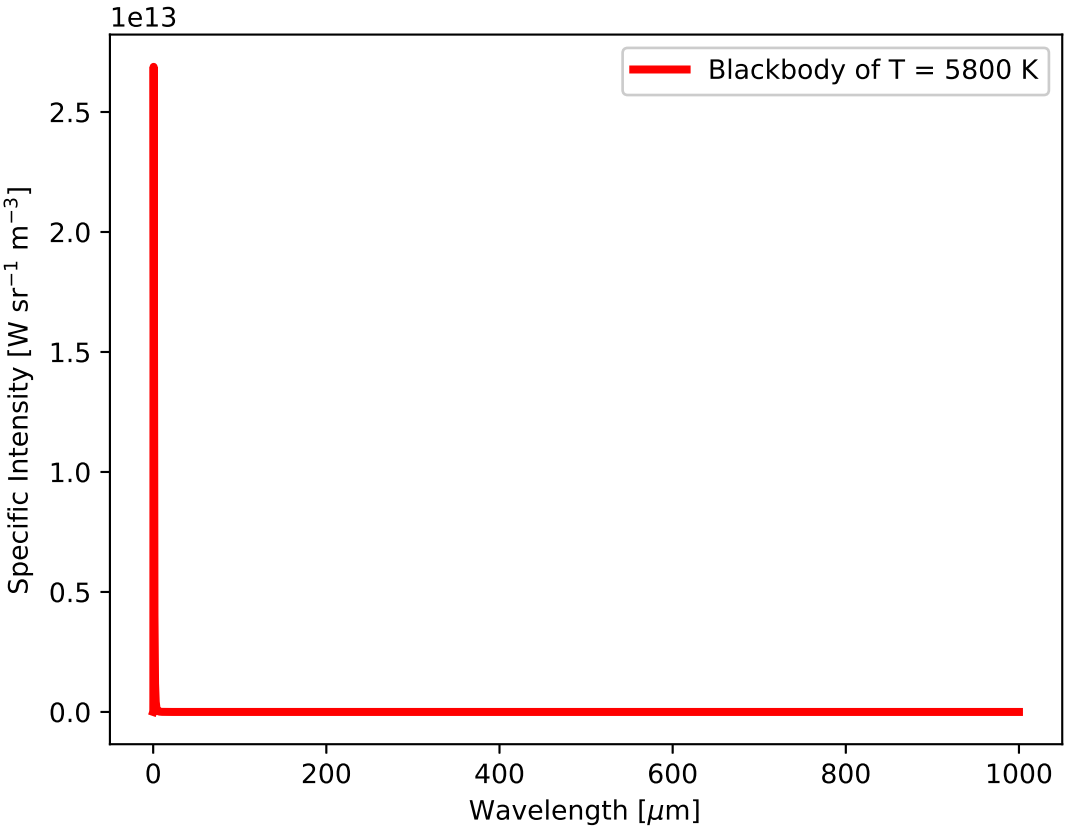

Figure 7: A model spectrum of $T = 5800$ K blackbody using Planck's radiation law.

```
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * c[0]**2 / wavelength**5 \
        / (numpy.exp (h[0] * c[0] / (wavelength * k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# temperature of blackbody
T = 5800.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of wavelength (from 10**-8 m = 10 nm to 10**-3 m = 1 mm)
wavelength_min = -8.0
wavelength_max = -3.0

# wavelength in metre
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001)

# T = 5800 K blackbody spectrum
bb_5800 = bb_lambda (wavelength, T)

# printing Planck function
print (f'Wavelength:')
print (f'{wavelength}')
print (f'Planck function:')
print (f'{bb_5800}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [$\mu$m]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-3}$]')

# axes
ax.set_xlim (0.03, 5.0)
ax.set_ylim (0.0, bb_5800.max () * 1.2)

# plotting data
ax.plot (wavelength * 10**6, bb_5800, linestyle='-', color='r', \
         linewidth=3, label='Blackbody of T = 5800 K')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to

```
% chmod a+x ai202209_s09_00_04.py
```

```
% ./ai202209_s09_00_04.py
Constants:
  c = 2.99792e+08 [m s^-1]
  h = 6.62607e-34 [J Hz^-1]
  k = 1.38065e-23 [J K^-1]
Temperature:
  T = 5800.0 K
Wavelength:
[1.00000000e-08 1.00230524e-08 1.00461579e-08 ... 9.95405417e-04
 9.97700064e-04 1.00000000e-03]
Planck function:
[2.20128336e-84 3.84995411e-84 6.72458175e-84 ... 4.88450670e+01
 4.83973915e+01 4.79538187e+01]
```

Display generated PNG file. (8)

```
% feh -dF ai202209_s09_00_04.png
```

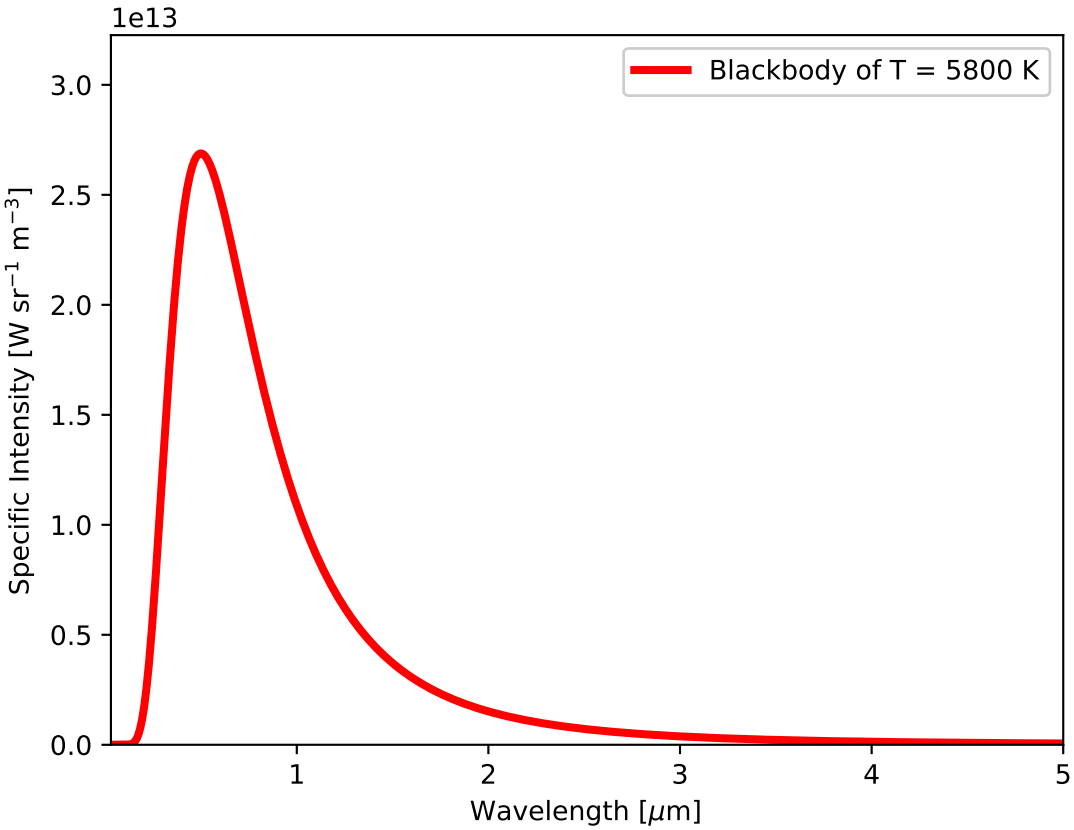

Figure 8: A model spectrum of $T = 5800$ K blackbody using Planck's radiation law from 30 nm to 5 μm.

Try following practice.

Practice 09-04

Use Planck's radiation law to construct a model spectrum of $T = 3000$ K blackbody and make a plot of it.

Make a log-log plot of a model spectrum of $T = 5800$ K blackbody.

Python Code 6: ai202209_s09_00_05.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/11 13:26:57 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_05.png'

#
# function to calculate blackbody curve
#
def bb_lambda (wavelength, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * c[0]**2 / wavelength**5 \
        / (numpy.exp (h[0] * c[0] / (wavelength * k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# temperature of blackbody
T = 5800.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of wavelength (from 10**-8 m = 10 nm to 10**-3 m = 1 mm)
wavelength_min = -8.0
wavelength_max = -3.0

# wavelength in metre
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001)

# T = 5800 K blackbody spectrum
bb_5800 = bb_lambda (wavelength, T)

# printing Planck function
print (f'Wavelength:')
```

```
print (f'{wavelength}')
print (f'Planck function:')
print (f'{bb_5800}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [$\mu$m]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-3}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (0.03, 1000.0)
ax.set_ylim (None, bb_5800.max () * 3)

# plotting data
ax.plot (wavelength * 10**6, bb_5800, linestyle='-', color='r', \
         linewidth=3, label='Blackbody of T = 5800 K')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to

```
% chmod a+x ai202209_s09_00_05.py
% ./ai202209_s09_00_05.py
Constants:
  c = 2.99792e+08 [m s^-1]
  h = 6.62607e-34 [J Hz^-1]
  k = 1.38065e-23 [J K^-1]
Temperature:
  T = 5800.0 K
Wavelength:
[1.00000000e-08 1.00230524e-08 1.00461579e-08 ... 9.95405417e-04
 9.97700064e-04 1.00000000e-03]
Planck function:
[2.20128336e-84 3.84995411e-84 6.72458175e-84 ... 4.88450670e+01
 4.83973915e+01 4.79538187e+01]
```

Display generated PNG file. (9)

```
% feh -dF ai202209_s09_00_05.png
```

Try following practice.

Practice 09-05

Use Planck's radiation law to construct a model spectrum of $T = 10000$ K blackbody and make a log-log plot of it.

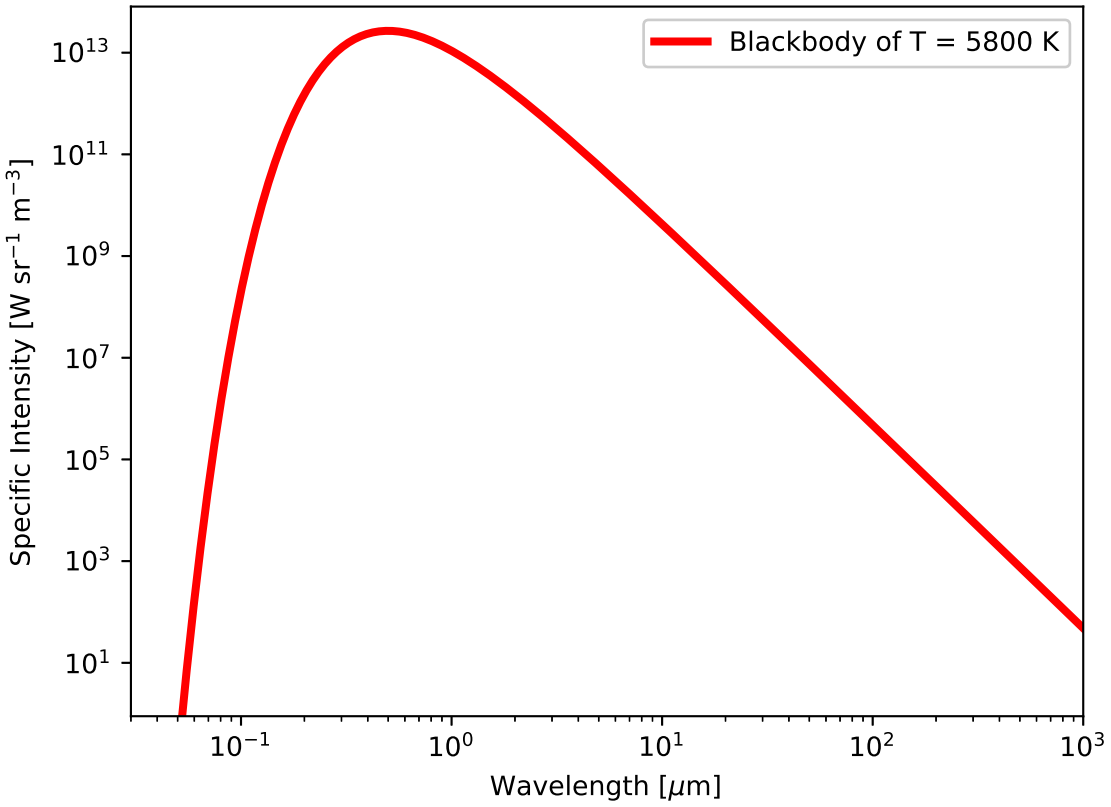

Figure 9: Log-log plot of a model spectrum of $T = 5800$ K blackbody using Planck's radiation law.

2.4 Finding peak wavelength of blackbody radiation

Find a peak wavelength of blackbody radiation of $T = 5800$ K.

Python Code 7: ai202209_s09_00_06.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 07:35:49 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants
import scipy.optimize

#
# function to calculate blackbody curve
#
def bb_lambda (wavelength, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * c[0]**2 / wavelength**5 \
        / (numpy.exp (h[0] * c[0] / (wavelength * k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody * -1.0)

# temperature of blackbody
T = 5800.0

# finding a peak of T=5800 K blackbody radiation spectrum
wavelength_peak = scipy.optimize.minimize_scalar (bb_lambda, \
                                                  bracket=(10**-8, 10**-3), \
                                                  args=(T), method='Brent')

# printing peak wavelength of black body radiation
print (f'peak wavelength of T={T} K blackbody = {wavelength_peak.x} m')
```

Execute above script to find a peak wavelength of blackbody radiation of $T = 5800$ K.

```
% chmod a+x ai202209_s09_00_06.py
% ./ai202209_s09_00_06.py
peak wavelength of T=5800.0 K blackbody = 4.996094419102113e-07 m
```

The peak of $T = 5800$ K blackbody spectrum is at $\lambda = 5.0 \times 10^{-7}$ m.
Find a peak wavelength of blackbody radiation of $T = 2500$ K.

Python Code 8: ai202209_s09_00_07.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 07:35:33 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants
import scipy.optimize

#
# function to calculate blackbody curve
#
def bb_lambda (wavelength, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * c[0]**2 / wavelength**5 \
        / (numpy.exp (h[0] * c[0] / (wavelength * k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody * -1.0)

# temperature of blackbody
T = 2500.0

# finding a peak of T=5800 K blackbody radiation spectrum
wavelength_peak = scipy.optimize.minimize_scalar (bb_lambda, \
                                                  bracket=(10**-8, 10**-3), \
                                                  args=(T), method='Brent')

# printing peak wavelength of black body radiation
print (f'peak wavelength of T={T} K blackbody = {wavelength_peak.x} m')
```

Execute above script to find a peak wavelength of blackbody radiation of $T = 5800$ K.

```
% chmod a+x ai202209_s09_00_07.py
% ./ai202209_s09_00_07.py
peak wavelength of T=2500.0 K blackbody = 1.1591097728957567e-06 m
```

The peak of $T = 2500$ K blackbody spectrum is at $\lambda = 1.2 \times 10^{-6}$ m.
Try following practice.

Practice 09-06

Use `scipy.optimize` to find a peak wavelength of $T = 300$ K blackbody spectrum.

Find the peak wavelength of a blackbody spectrum using Wien's displacement law.

Python Code 9: ai202209_s09_00_08.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 08:20:13 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants
import scipy.optimize

# speed of light
c = scipy.constants.physical_constants['speed of light in vacuum']

# Planck constant
h = scipy.constants.physical_constants['Planck constant']

# Boltzmann constant
k = scipy.constants.physical_constants['Boltzmann constant']

# temperature of blackbody
T = 5800.0

# function for solving the equation (x-5) * exp(x) + 5 = 0
def peak (x):
    y = ( (x - 5.0) * numpy.exp (x) + 5 )**2
    return (y)

# finding the minimum x corresponding to the peak of blackbody radiation
x_min = scipy.optimize.minimize_scalar (peak, bracket=(2.0, 5.0), \
                                        method='Brent')

# printing the minimum x
print (x_min)

# finding a peak of T=5800 K blackbody radiation spectrum
wavelength_peak = h[0] * c[0] / (x_min.x * k[0] * T)

# printing peak wavelength of black body radiation
print (f'peak wavelength of T={T} K blackbody = {wavelength_peak} m')
```

Execute above script to find a peak wavelength of blackbody radiation of $T = 5800$ K using Wien's displacement law.

```
% chmod a+x ai202209_s09_00_08.py
% ./ai202209_s09_00_08.py
     fun: 1.3243140920601533e-15
 message: '\nOptimization terminated successfully;\nThe returned value satisf
ies the termination criteria\n(using xtol = 1.48e-08 )'
    nfev: 19
     nit: 15
 success: True
       x: 4.965114232007361
peak wavelength of T=5800.0 K blackbody = 4.996158543157982e-07 m
```

The peak of $T = 2500$ K blackbody spectrum is at $\lambda = 5.0 \times 10^{-7}$ m.

Try following practice.

Practice 09-07

Use Wien's displacement law to find a peak wavelength of $T = 2500$ K blackbody spectrum.

2.5 Showing regions of UV, visible, and IR

Plot a model spectrum of $T = 5800$ K blackbody, and superimpose regions of UV, visible, and IR.

Python Code 10: ai202209_s09_00_09.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 08:58:29 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_10.png'

#
# function to calculate blackbody curve
#
def bb_lambda (wavelength, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * c[0]**2 / wavelength**5 \
        / (numpy.exp (h[0] * c[0] / (wavelength * k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# temperature of blackbody
T = 5800.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of wavelength (from 10**-8 m = 10 nm to 10**-3 m = 1 mm)
wavelength_min = -8.0
```

```
wavelength_max = -3.0

# wavelength in metre
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001)

# T = 5800 K blackbody spectrum
bb_5800 = bb_lambda (wavelength, T)

# printing Planck function
print (f'Wavelength:')
print (f'{wavelength}')
print (f'Planck function:')
print (f'{bb_5800}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [$\mu$m]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-3}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (0.03, 1000.0)
ax.set_ylim (10**0, bb_5800.max () * 3)

# showing UV region
ax.fill_between ([0.03, 0.3], 10**-2, 10**18, color='violet', alpha=0.1)
ax.text (x=0.1, y=10**1, s='UV')

# showing visible region
ax.fill_between ([0.3, 1.0], 10**-2, 10**18, color='green', alpha=0.1)
ax.text (x=0.35, y=10**1, s='Visible')

# showing IR region
ax.fill_between ([1.0, 1000.0], 10**-2, 10**18, color='red', alpha=0.1)
ax.text (x=3.0, y=10**1, s='IR')

# plotting data
ax.plot (wavelength * 10**6, bb_5800, linestyle='-', color='r', \
         linewidth=3, label='Blackbody of T = 5800 K')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to plot a model spectrum of $T = 5800$ K blackbody, and superimpose regions of UV, visible, and IR.

```
% chmod a+x ai202209_s09_00_09.py
% ./ai202209_s09_00_09.py
Temperature:
  T = 5800.0 K
Wavelength:
[1.00000000e-08 1.00230524e-08 1.00461579e-08 ... 9.95405417e-04
 9.97700064e-04 1.00000000e-03]
```

```
Planck function:
[2.20128336e-84 3.84995411e-84 6.72458175e-84 ... 4.88450670e+01
 4.83973915e+01 4.79538187e+01]
```

Display generated PNG file. (10) The peak of the spectrum is in visible wavelength.

```
% feh -dF ai202209_s09_00_09.png
```

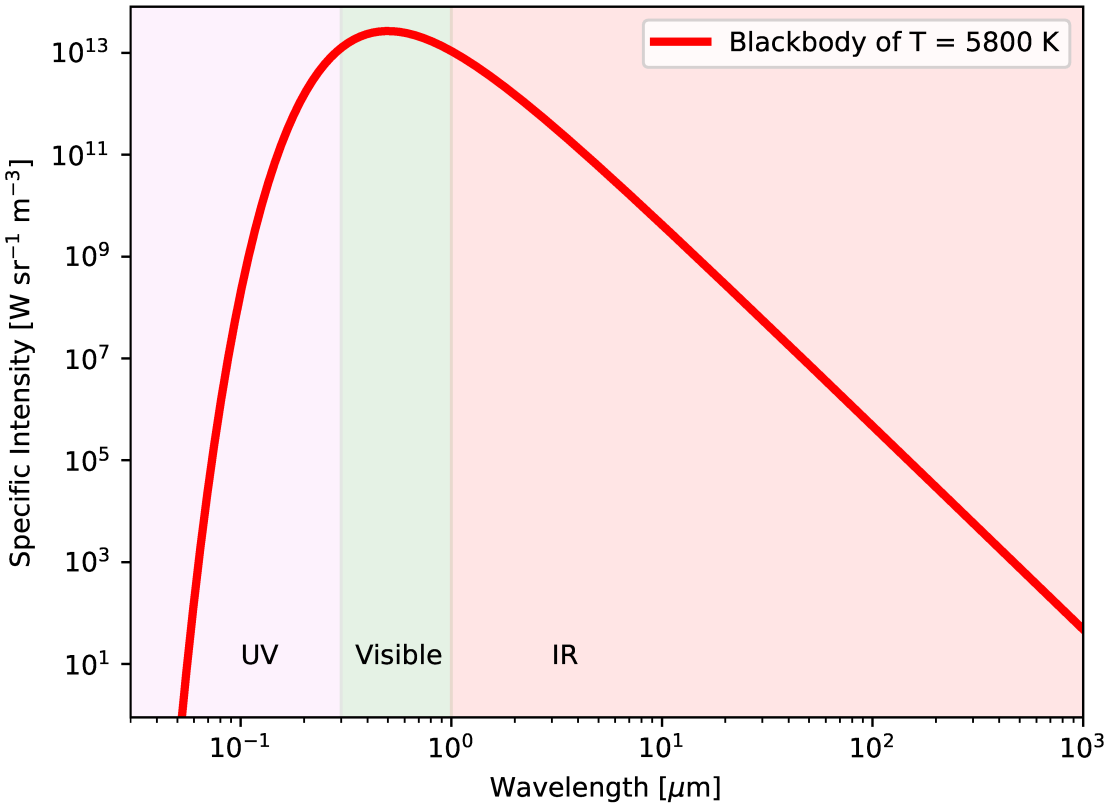

Figure 10: Log-log plot of a model spectrum of $T = 5800$ K blackbody using Planck's radiation law. UV, visible, and IR regions are superimposed on a plot.

Try following practice.

Practice 09-08

Make a plot of $T = 10000$ K blackbody model spectrum using Planck's radiation law, and show regions of UV, visible, and IR wavelengths on the same plot.

Plot a model spectrum of $T = 30000$ K blackbody, and superimpose regions of UV, visible, and IR.

Python Code 11: ai202209_s09_00_10.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 09:01:11 (CST) daisuke>
#
```

```
# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_10.png'

#
# function to calculate blackbody curve
#
def bb_lambda (wavelength, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * c[0]**2 / wavelength**5 \
        / (numpy.exp (h[0] * c[0] / (wavelength * k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# temperature of blackbody
T = 30000.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of wavelength (from 10**-8 m = 10 nm to 10**-3 m = 1 mm)
wavelength_min = -8.0
wavelength_max = -3.0

# wavelength in metre
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001)

# T = 30000 K blackbody spectrum
bb_30000 = bb_lambda (wavelength, T)

# printing Planck function
print (f'Wavelength:')
print (f'{wavelength}')
print (f'Planck function:')
print (f'{bb_30000}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
```

```
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [$\mu$m]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-3}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (0.03, 1000.0)
ax.set_ylim (10**0, bb_30000.max () * 3)

# showing UV region
ax.fill_between ([0.03, 0.3], 10**-2, 10**18, color='violet', alpha=0.1)
ax.text (x=0.1, y=10**1, s='UV')

# showing visible region
ax.fill_between ([0.3, 1.0], 10**-2, 10**18, color='green', alpha=0.1)
ax.text (x=0.35, y=10**1, s='Visible')

# showing IR region
ax.fill_between ([1.0, 1000.0], 10**-2, 10**18, color='red', alpha=0.1)
ax.text (x=3.0, y=10**1, s='IR')

# plotting data
ax.plot (wavelength * 10**6, bb_30000, linestyle='-', color='r', \
         linewidth=3, label='Blackbody of T = 30000 K')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to plot a model spectrum of $T = 30000$ K blackbody, and superimpose regions of UV, visible, and IR.

```
% chmod a+x ai202209_s09_00_10.py
% ./ai202209_s09_00_10.py
Temperature:
  T = 30000.0 K
Wavelength:
[1.00000000e-08 1.00230524e-08 1.00461579e-08 ... 9.95405417e-04
 9.97700064e-04 1.00000000e-03]
Planck function:
[1768.06737833 1951.65418766 2153.75722834 ...  252.90103631  250.58256452
  248.28534702]
```

Display generated PNG file. (11) The peak of the spectrum is in UV wavelength.

```
% feh -dF ai202209_s09_00_10.png
```

Try following practice.

Practice 09-09

Make a plot of $T = 50000$ K blackbody model spectrum using Planck's radiation law, and show regions of UV, visible, and IR wavelengths on the same plot.

Plot a model spectrum of $T = 2000$ K blackbody, and superimpose regions of UV, visible, and IR.

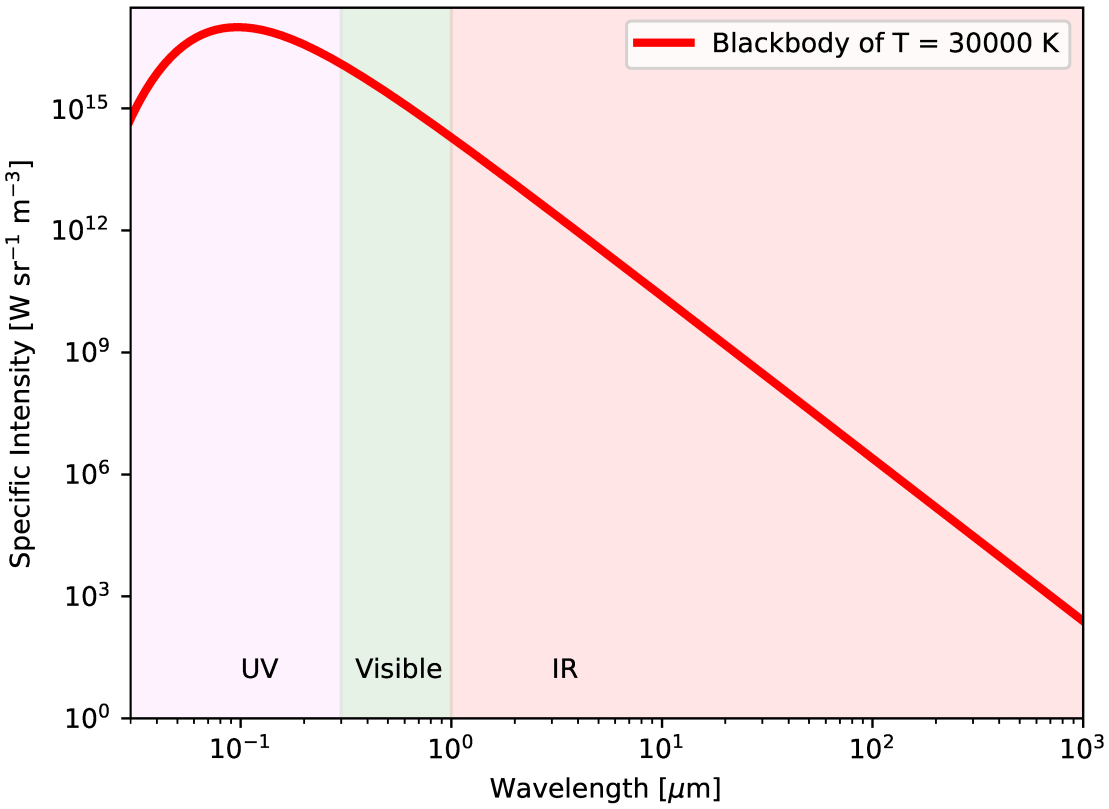

Figure 11: Log-log plot of a model spectrum of $T = 30000$ K blackbody using Planck's radiation law. UV, visible, and IR regions are superimposed on a plot.

Python Code 12: ai202209_s09_00_11.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 09:08:59 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_11.png'

#
# function to calculate blackbody curve
#
def bb_lambda (wavelength, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * c[0]**2 / wavelength**5 \
        / (numpy.exp (h[0] * c[0] / (wavelength * k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# temperature of blackbody
T = 2000.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of wavelength (from 10**-8 m = 10 nm to 10**-3 m = 1 mm)
wavelength_min = -8.0
wavelength_max = -3.0

# wavelength in metre
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001)

# T = 2000 K blackbody spectrum
bb_2000 = bb_lambda (wavelength, T)

# printing Planck function
print (f'Wavelength:')
```

```
print (f'{wavelength}')
print (f'Planck function:')
print (f'{bb_2000}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [$\mu$m]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-3}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (0.03, 1000.0)
ax.set_ylim (10**0, bb_2000.max () * 3)

# showing UV region
ax.fill_between ([0.03, 0.3], 10**-2, 10**18, color='violet', alpha=0.1)
ax.text (x=0.1, y=10**1, s='UV')

# showing visible region
ax.fill_between ([0.3, 1.0], 10**-2, 10**18, color='green', alpha=0.1)
ax.text (x=0.35, y=10**1, s='Visible')

# showing IR region
ax.fill_between ([1.0, 1000.0], 10**-2, 10**18, color='red', alpha=0.1)
ax.text (x=3.0, y=10**1, s='IR')

# plotting data
ax.plot (wavelength * 10**6, bb_2000, linestyle='-', color='r', \
         linewidth=3, label='Blackbody of T = 2000 K')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to plot a model spectrum of $T = 2000$ K blackbody, and superimpose regions of UV, visible, and IR.

```
% chmod a+x ai202209_s09_00_11.py
% ./ai202209_s09_00_11.py
Temperature:
  T = 2000.0 K
Wavelength:
[1.00000000e-08 1.00230524e-08 1.00461579e-08 ... 9.95405417e-04
 9.97700064e-04 1.00000000e-03]
Planck function:
[ 0.          0.          0.         ... 16.80326549 16.64935076
 16.49684555]
```

Display generated PNG file. (12) The peak of the spectrum is in IR wavelength.

```
% feh -dF ai202209_s09_00_11.png
```

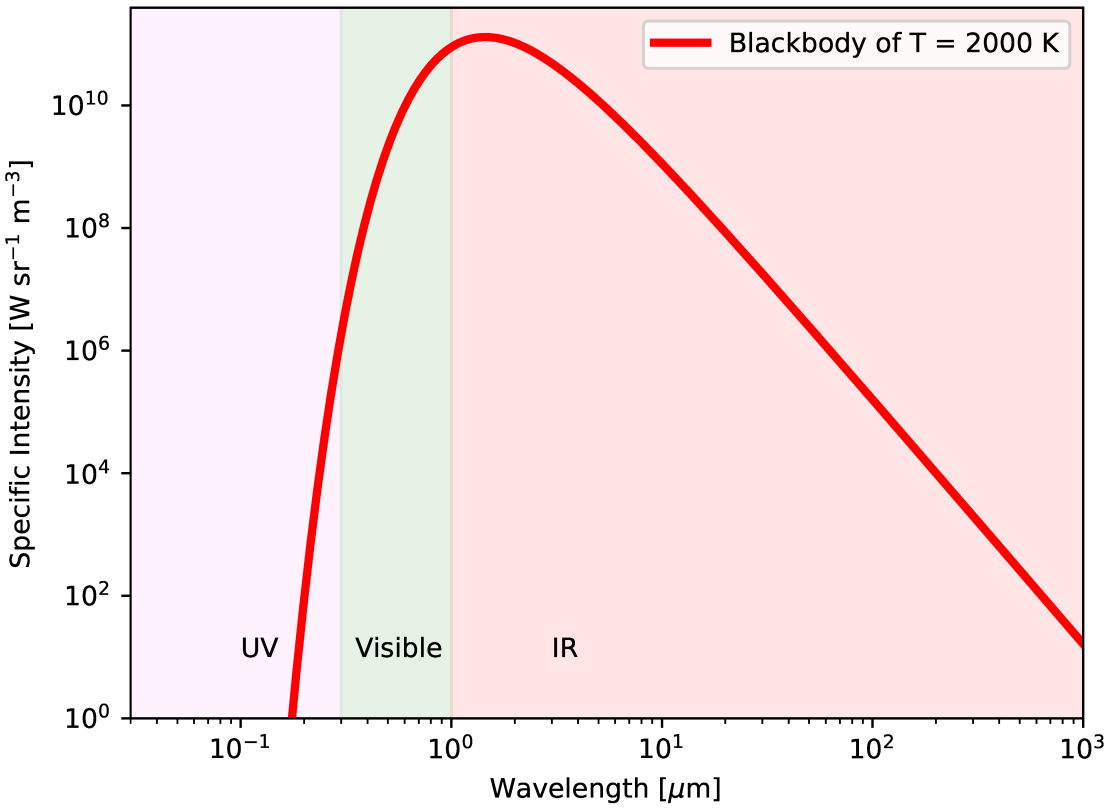

Figure 12: Log-log plot of a model spectrum of $T = 2000$ K blackbody using Planck's radiation law. UV, visible, and IR regions are superimposed on a plot.

Try following practice.

Practice 09-10

Make a plot of $T = 1000$ K blackbody model spectrum using Planck's radiation law, and show regions of UV, visible, and IR wavelengths on the same plot.

2.6 Rayleigh-Jeans law

Plot a model spectrum of $T = 5800$ K blackbody as a function of frequency using Planck's radiation law.

Python Code 13: ai202209_s09_00_12.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 14:23:57 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_12.png'

#
# function to calculate blackbody curve
#
def bb_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * frequency**3 / c[0]**2 \
        / (numpy.exp (h[0] * frequency / (k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# temperature of blackbody
T = 5800.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of frequency (from 10**0 Hz to 10**16 Hz)
```

```
frequency_min = 0.0
frequency_max = 16.0

# frequency in Hz
frequency = numpy.logspace (frequency_min, frequency_max, num=16001)

# T = 5800 K blackbody spectrum
bb_5800 = bb_nu (frequency, T)

# printing Planck function
print (f'Frequency:')
print (f'{frequency}')
print (f'Planck function:')
print (f'{bb_5800}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Frequency [Hz]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-2}$ Hz$^{-1}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (10**1, 10**18)
ax.set_ylim (10**-30, 10**-3)

# plotting data
ax.plot (frequency, bb_5800, linestyle='-', color='r', \
         linewidth=3, label='Blackbody of T = 5800 K')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to plot a model spectrum of $T = 5800$ K blackbody as a function of frequency.

```
% chmod a+x ai202209_s09_00_12.py
% ./ai202209_s09_00_12.py
Temperature:
  T = 5800.0 K
Frequency:
[1.00000000e+00 1.00230524e+00 1.00461579e+00 ... 9.95405417e+15
 9.97700064e+15 1.00000000e+16]
Planck function:
[1.79474466e-36 1.80718524e-36 1.81971204e-36 ... 2.46496487e-38
 2.05281799e-38 1.70883475e-38]
```

Display generated PNG file. (13)

```
% feh -dF ai202209_s09_00_12.png
```

Try following practice.

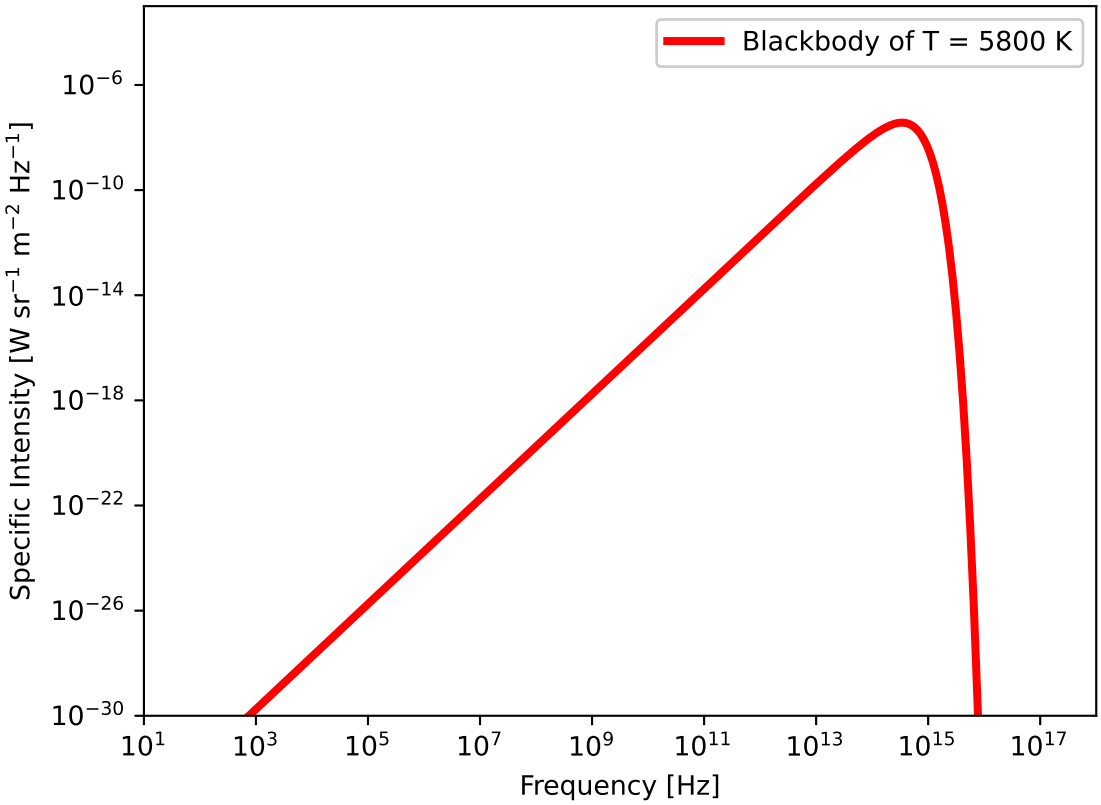

Figure 13: Log-log plot of a model spectrum of $T = 5800$ K blackbody as a function of frequency using Planck's radiation law.

Practice 09-11

Make a plot of $T = 8000$ K blackbody model spectrum as a function of frequency using Planck's radiation law.

Plot a model spectrum of $T = 5800$ K blackbody as a function of frequency using Planck's radiation law. Use `.secondary_xaxis ()` method to add a secondary X-axis for wavelength.

Python Code 14: ai202209_s09_00_13.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 14:54:37 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_13.png'

#
# function to calculate blackbody curve
#
def bb_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * frequency**3 / c[0]**2 \
        / (numpy.exp (h[0] * frequency / (k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# temperature of blackbody
T = 5800.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of frequency (from 10**0 Hz to 10**16 Hz)
frequency_min = 0.0
frequency_max = 16.0

# frequency in Hz
```

```
frequency = numpy.logspace (frequency_min, frequency_max, num=16001)

# T = 5800 K blackbody spectrum
bb_5800 = bb_nu (frequency, T)

# printing Planck function
print (f'Frequency:')
print (f'{frequency}')
print (f'Planck function:')
print (f'{bb_5800}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Frequency [Hz]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-2}$ Hz$^{-1}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (10**1, 10**18)
ax.set_ylim (10**-30, 10**-3)

# make secondary X-axis
c   = scipy.constants.physical_constants['speed of light in vacuum'][0]
ax2 = ax.secondary_xaxis (location='top', \
                          functions=(lambda x: c/x, lambda x: c/x) )
ax2.set_xlabel ('Wavelength [m]')

# plotting data
ax.plot (frequency, bb_5800, linestyle='-', color='r', \
         linewidth=3, label='Blackbody of T = 5800 K')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to plot a model spectrum of $T = 5800$ K blackbody as a function of frequency.

```
% chmod a+x ai202209_s09_00_13.py
% ./ai202209_s09_00_13.py
Temperature:
  T = 5800.0 K
Frequency:
[1.00000000e+00 1.00230524e+00 1.00461579e+00 ... 9.95405417e+15
 9.97700064e+15 1.00000000e+16]
Planck function:
[1.79474466e-36 1.80718524e-36 1.81971204e-36 ... 2.46496487e-38
 2.05281799e-38 1.70883475e-38]
```

Display generated PNG file. (14)

```
% feh -dF ai202209_s09_00_13.png
```

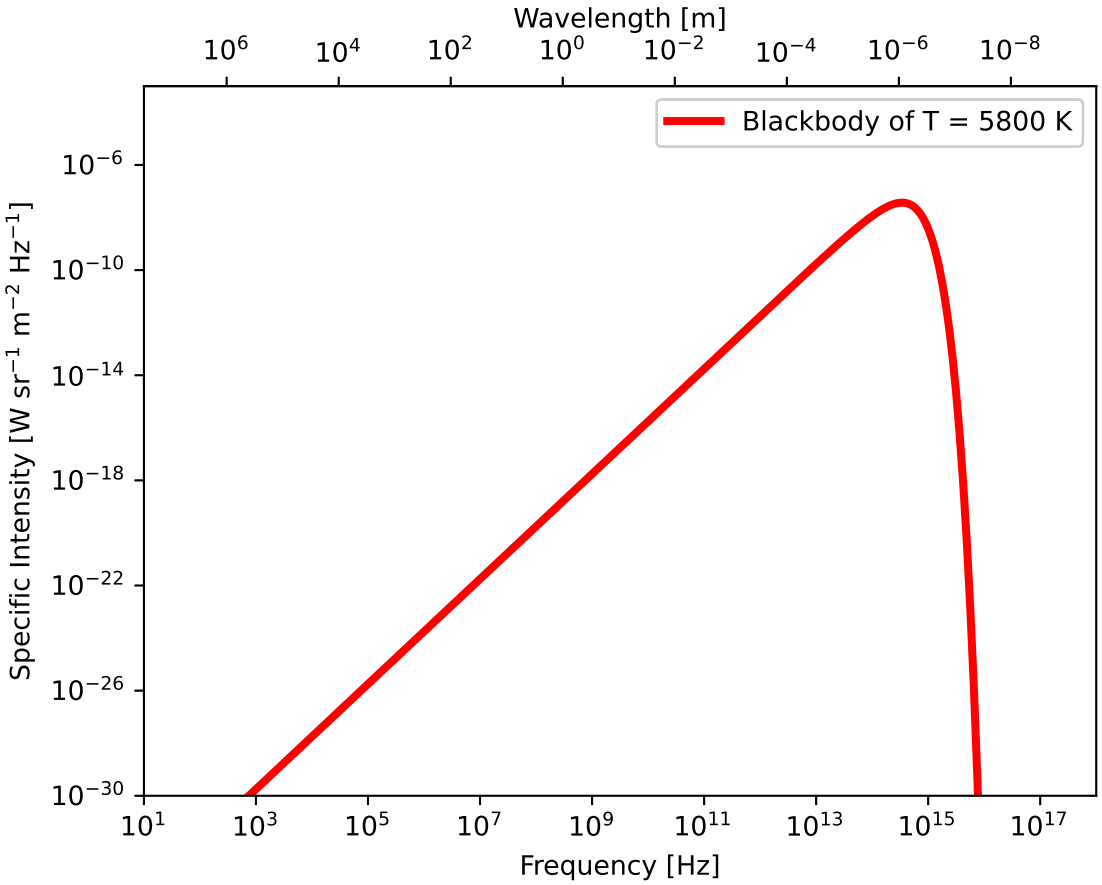

Figure 14: Log-log plot of a model spectrum of $T = 5800$ K blackbody as a function of frequency using Planck's radiation law.

Try following practice.

Practice 09-12

Make a plot of $T = 20000$ K blackbody model spectrum as a function of frequency using Planck's radiation law, and make a secondary X-axis showing wavelength.

Plot a model spectrum of $T = 5800$ K blackbody as a function of frequency using Planck's radiation law. Superimpose a model spectrum of $T = 5800$ K blackbody by Rayleigh-Jeans law.

Python Code 15: ai202209_s09_00_14.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 15:04:06 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_14.png'

#
# function to calculate blackbody curve
#
def bb_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * frequency**3 / c[0]**2 \
        / (numpy.exp (h[0] * frequency / (k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# function to calculate Rayleigh-Jeans law
def rj_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Rayleigh-Jeans law
    blackbody = 2.0 * frequency**2 * k[0] * T / c[0]**2
```

```
    # returning blackbody curve by Rayleigh-Jeans law
    return (blackbody)

# temperature of blackbody
T = 5800.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of frequency (from 10**0 Hz to 10**16 Hz)
frequency_min = 0.0
frequency_max = 16.0

# frequency in Hz
frequency = numpy.logspace (frequency_min, frequency_max, num=16001)

# T = 5800 K blackbody spectrum
bb_5800 = bb_nu (frequency, T)
rj_5800 = rj_nu (frequency, T)

# printing Planck function
print (f'Frequency:')
print (f'{frequency}')
print (f'Planck function:')
print (f'{bb_5800}')
print (f'Rayleigh-Jeans law:')
print (f'{rj_5800}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Frequency [Hz]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-2}$ Hz$^{-1}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (10**1, 10**18)
ax.set_ylim (10**-30, 10**-3)

# make secondary X-axis
c   = scipy.constants.physical_constants['speed of light in vacuum'][0]
ax2 = ax.secondary_xaxis (location='top', \
                          functions=(lambda x: c/x, lambda x: c/x) )
ax2.set_xlabel ('Wavelength [m]')

# plotting data
ax.plot (frequency, bb_5800, linestyle='-', color='r', \
         linewidth=5, label='Blackbody of T = 5800 K')
ax.plot (frequency, rj_5800, linestyle='--', color='b', \
         linewidth=3, label='Rayleigh-Jeans law for T = 5800 K')
ax.legend ()

# saving the plot into a file
```

```
fig.savefig (file_output, dpi=225)
```

Execute above script to plot model spectra of $T = 5800$ K blackbody as a function of frequency.

```
% chmod a+x ai202209_s09_00_14.py
% ./ai202209_s09_00_14.py
Temperature:
  T = 5800.0 K
Frequency:
[1.00000000e+00 1.00230524e+00 1.00461579e+00 ... 9.95405417e+15
 9.97700064e+15 1.00000000e+16]
Planck function:
[1.79474466e-36 1.80718524e-36 1.81971204e-36 ... 2.46496487e-38
 2.05281799e-38 1.70883475e-38]
Rayleigh-Jeans law:
[1.78196786e-36 1.79019305e-36 1.79845620e-36 ... 1.76563068e-04
 1.77378046e-04 1.78196786e-04]
```

Display generated PNG file. (15)

```
% feh -dF ai202209_s09_00_14.png
```

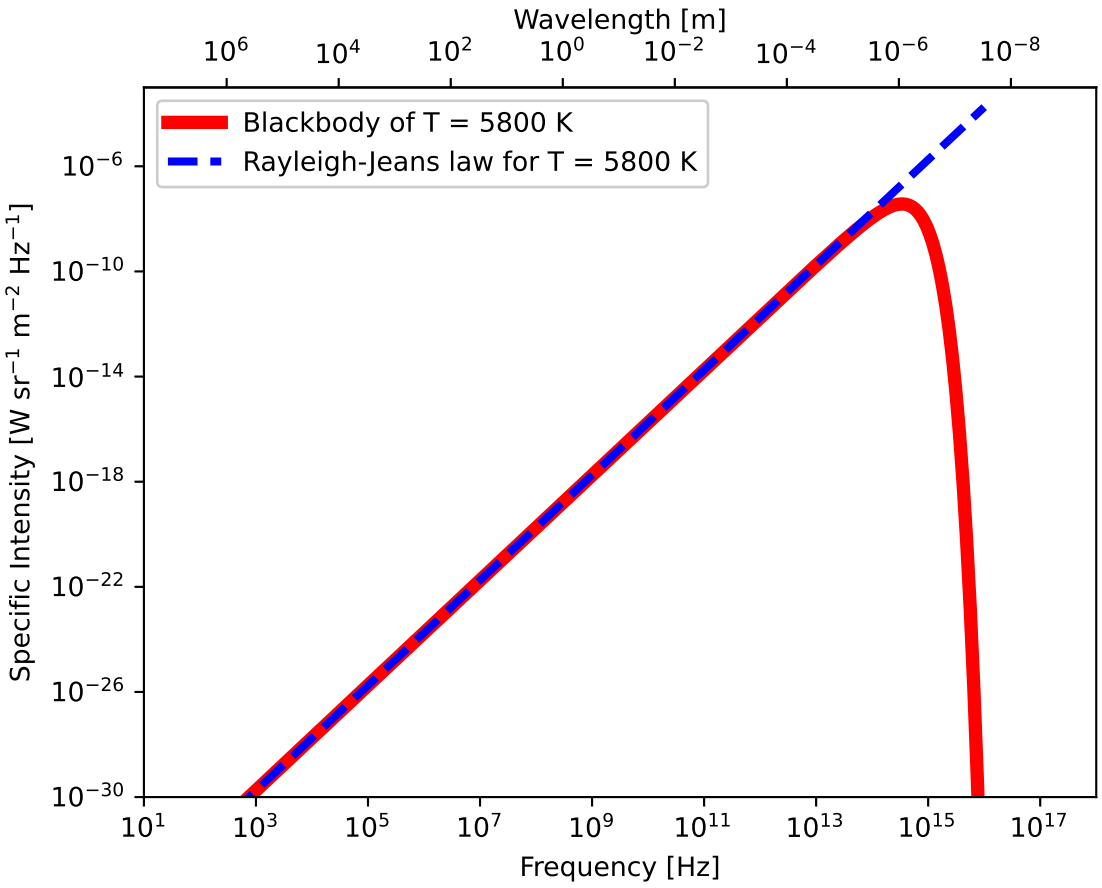

Figure 15: Log-log plot of model spectra of $T = 5800$ K blackbody as a function of frequency using Planck's radiation law and Rayleigh-Jeans law.

Try following practice.

Practice 09-13

Make a plot of $T = 20000$ K blackbody model spectrum as a function of frequency using Planck's radiation law, and make a secondary X-axis showing wavelength. Superimpose a model spectrum of $T = 20000$ K blackbody using Rayleigh-Jeans law.

2.7 Wien's law

Plot a model spectrum of $T = 5800$ K blackbody as a function of frequency using Planck's radiation law. Superimpose model spectra of $T = 5800$ K blackbody by Rayleigh-Jeans law and Wien's law.

Python Code 16: ai202209_s09_00_15.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 16:31:51 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_15.png'

#
# function to calculate blackbody curve
#
def bb_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * frequency**3 / c[0]**2 \
        / (numpy.exp (h[0] * frequency / (k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# function to calculate Rayleigh-Jeans law
def rj_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']
```

```
    # calculation of Rayleigh-Jeans law
    blackbody = 2.0 * frequency**2 * k[0] * T / c[0]**2

    # returning blackbody curve by Rayleigh-Jeans law
    return (blackbody)

# function to calculate Wien's law
def wi_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Wien's law
    blackbody = 2.0 * h[0] * frequency**3 / c[0]**2 \
        * numpy.exp (-h[0] * frequency / (k[0] * T) )

    # returning blackbody curve by Wien's law
    return (blackbody)

# temperature of blackbody
T = 5800.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of frequency (from 10**0 Hz to 10**16 Hz)
frequency_min = 0.0
frequency_max = 16.0

# frequency in Hz
frequency = numpy.logspace (frequency_min, frequency_max, num=16001)

# T = 5800 K blackbody spectrum
bb_5800 = bb_nu (frequency, T)
rj_5800 = rj_nu (frequency, T)
wi_5800 = wi_nu (frequency, T)

# printing Planck function
print (f'Frequency:')
print (f'{frequency}')
print (f'Planck function:')
print (f'{bb_5800}')
print (f'Rayleigh-Jeans law:')
print (f'{rj_5800}')
print (f'Wien\'s law:')
print (f'{wi_5800}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
```

```
ax.set_xlabel ('Frequency [Hz]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-2}$ Hz$^{-1}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (10**1, 10**18)
ax.set_ylim (10**-30, 10**-3)

# make secondary X-axis
c   = scipy.constants.physical_constants['speed of light in vacuum'][0]
ax2 = ax.secondary_xaxis (location='top', \
                          functions=(lambda x: c/x, lambda x: c/x) )
ax2.set_xlabel ('Wavelength [m]')

# plotting data
ax.plot (frequency, bb_5800, linestyle='-', color='r', \
         linewidth=5, label='Blackbody of T = 5800 K')
ax.plot (frequency, rj_5800, linestyle='--', color='b', \
         linewidth=3, label='Rayleigh-Jeans law for T = 5800 K')
ax.plot (frequency, wi_5800, linestyle=':', color='g', \
         linewidth=3, label='Wien\'s law for T = 5800 K')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to plot model spectra of $T = 5800$ K blackbody as a function of frequency.

```
% chmod a+x ai202209_s09_00_15.py
% ./ai202209_s09_00_15.py
Temperature:
  T = 5800.0 K
Frequency:
[1.00000000e+00 1.00230524e+00 1.00461579e+00 ... 9.95405417e+15
 9.97700064e+15 1.00000000e+16]
Planck function:
[1.79474466e-36 1.80718524e-36 1.81971204e-36 ... 2.46496487e-38
 2.05281799e-38 1.70883475e-38]
Rayleigh-Jeans law:
[1.78196786e-36 1.79019305e-36 1.79845620e-36 ... 1.76563068e-04
 1.77378046e-04 1.78196786e-04]
Wien's law:
[1.47449946e-50 1.48472021e-50 1.49501180e-50 ... 2.46496487e-38
 2.05281799e-38 1.70883475e-38]
```

Display generated PNG file. (16)

```
% feh -dF ai202209_s09_00_15.png
```

Try following practice.

Practice 09-14

Make a plot of $T = 3500$ K blackbody model spectrum as a function of frequency using Planck's radiation law, and make a secondary X-axis showing wavelength. Superimpose model spectra of $T = 3500$ K blackbody using Rayleigh-Jeans law and Wien's law.

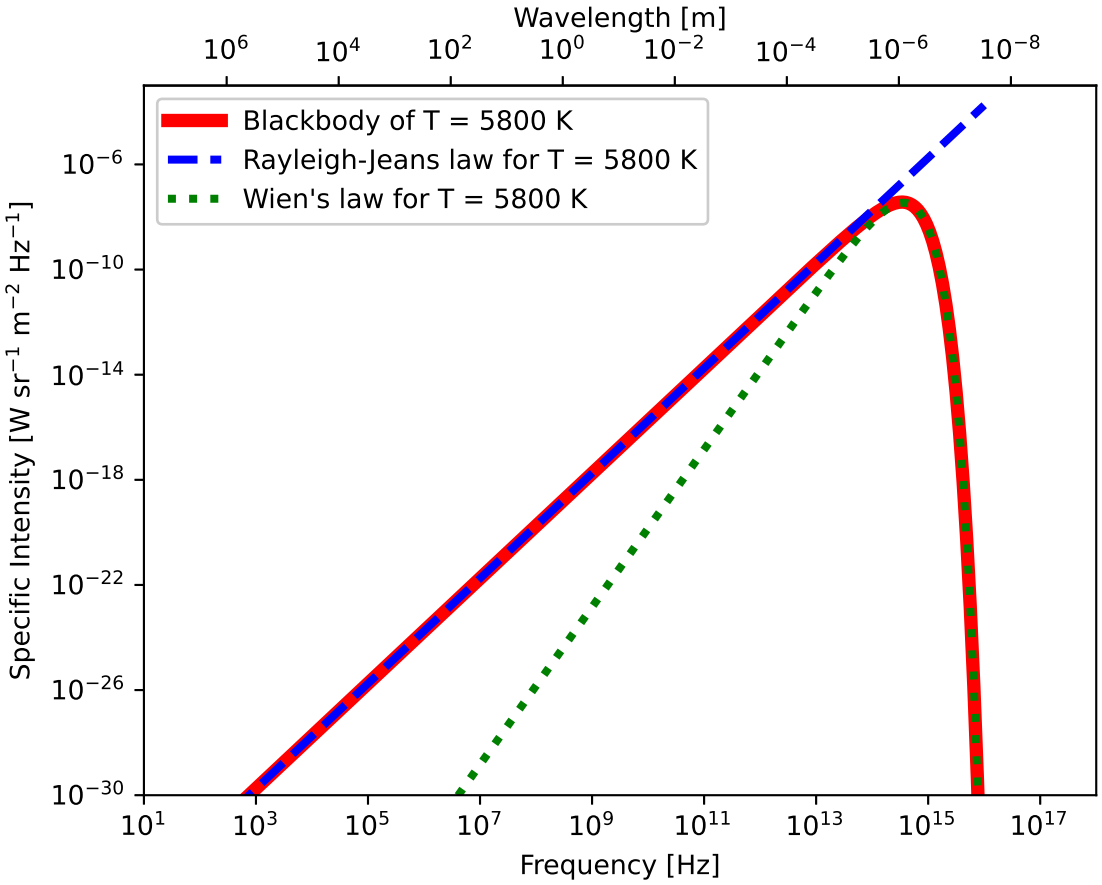

Figure 16: Log-log plot of model spectra of $T = 5800$ K blackbody as a function of frequency using Planck's radiation law, Rayleigh-Jeans law, and Wien's law.

Plot a model spectrum of $T = 5800$ K blackbody as a function of frequency using Planck's radiation law. Superimpose model spectra of $T = 5800$ K blackbody by Rayleigh-Jeans law and Wien's law. Superimpose ranges of X-ray, UV, visible, IR, and radio frequencies.

Python Code 17: ai202209_s09_00_16.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 18:43:56 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_16.png'

#
# function to calculate blackbody curve
#
def bb_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * frequency**3 / c[0]**2 \
        / (numpy.exp (h[0] * frequency / (k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# function to calculate Rayleigh-Jeans law
def rj_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Rayleigh-Jeans law
    blackbody = 2.0 * frequency**2 * k[0] * T / c[0]**2

    # returning blackbody curve by Rayleigh-Jeans law
    return (blackbody)

# function to calculate Wien's law
```

```
def wi_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Wien's law
    blackbody = 2.0 * h[0] * frequency**3 / c[0]**2 \
        * numpy.exp (-h[0] * frequency / (k[0] * T) )

    # returning blackbody curve by Wien's law
    return (blackbody)

# temperature of blackbody
T = 5800.0

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of frequency (from 10**0 Hz to 10**16 Hz)
frequency_min = 0.0
frequency_max = 16.0

# frequency in Hz
frequency = numpy.logspace (frequency_min, frequency_max, num=16001)

# T = 5800 K blackbody spectrum
bb_5800 = bb_nu (frequency, T)
rj_5800 = rj_nu (frequency, T)
wi_5800 = wi_nu (frequency, T)

# printing Planck function
print (f'Frequency:')
print (f'{frequency}')
print (f'Planck function:')
print (f'{bb_5800}')
print (f'Rayleigh-Jeans law:')
print (f'{rj_5800}')
print (f'Wien\'s law:')
print (f'{wi_5800}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Frequency [Hz]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-2}$ Hz$^{-1}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (10**1, 10**18)
```

```
ax.set_ylim (10**-30, 10**-3)

# make secondary X-axis
c   = scipy.constants.physical_constants['speed of light in vacuum'][0]
ax2 = ax.secondary_xaxis (location='top', \
                          functions=(lambda x: c/x, lambda x: c/x) )
ax2.set_xlabel ('Wavelength [m]')

# showing X-ray region
ax.fill_between ([3*10**16, 10**18], 10**-30, 10**-3, \
                 color='cyan', alpha=0.1)
ax.text (x=4*10**16, y=10**-29, s='X-ray')

# showing UV region
ax.fill_between ([10**15, 3*10**16], 10**-30, 10**-3, \
                 color='violet', alpha=0.1)
ax.text (x=1.1*10**15, y=10**-29, s='UV')

# showing visible region
ax.fill_between ([3*10**14, 10**15], 10**-30, 10**-3, \
                 color='green', alpha=0.1)
ax.text (x=10**14, y=3*10**-28, s='Visible')

# showing IR region
ax.fill_between ([10**12, 3*10**14], 10**-30, 10**-3, \
                 color='red', alpha=0.1)
ax.text (x=10**13, y=10**-29, s='IR')

# showing radio region
ax.fill_between ([10**0, 10**12], 10**-30, 10**-3, color='yellow', alpha=0.1)
ax.text (x=10**8, y=10**-29, s='Radio')

# plotting data
ax.plot (frequency, bb_5800, linestyle='-', color='r', \
         linewidth=5, label='Blackbody of T = 5800 K')
ax.plot (frequency, rj_5800, linestyle='--', color='b', \
         linewidth=3, label='Rayleigh-Jeans law for T = 5800 K')
ax.plot (frequency, wi_5800, linestyle=':', color='g', \
         linewidth=3, label='Wien\'s law for T = 5800 K')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to plot model spectra of $T = 5800$ K blackbody as a function of frequency.

```
% chmod a+x ai202209_s09_00_16.py
% ./ai202209_s09_00_16.py
Temperature:
  T = 5800.0 K
Frequency:
[1.00000000e+00 1.00230524e+00 1.00461579e+00 ... 9.95405417e+15
 9.97700064e+15 1.00000000e+16]
Planck function:
[1.79474466e-36 1.80718524e-36 1.81971204e-36 ... 2.46496487e-38
 2.05281799e-38 1.70883475e-38]
Rayleigh-Jeans law:
[1.78196786e-36 1.79019305e-36 1.79845620e-36 ... 1.76563068e-04
 1.77378046e-04 1.78196786e-04]
```

```
Wien's law:
[1.47449946e-50 1.48472021e-50 1.49501180e-50 ... 2.46496487e-38
 2.05281799e-38 1.70883475e-38]
```

Display generated PNG file. (17)

```
% feh -dF ai202209_s09_00_16.png
```

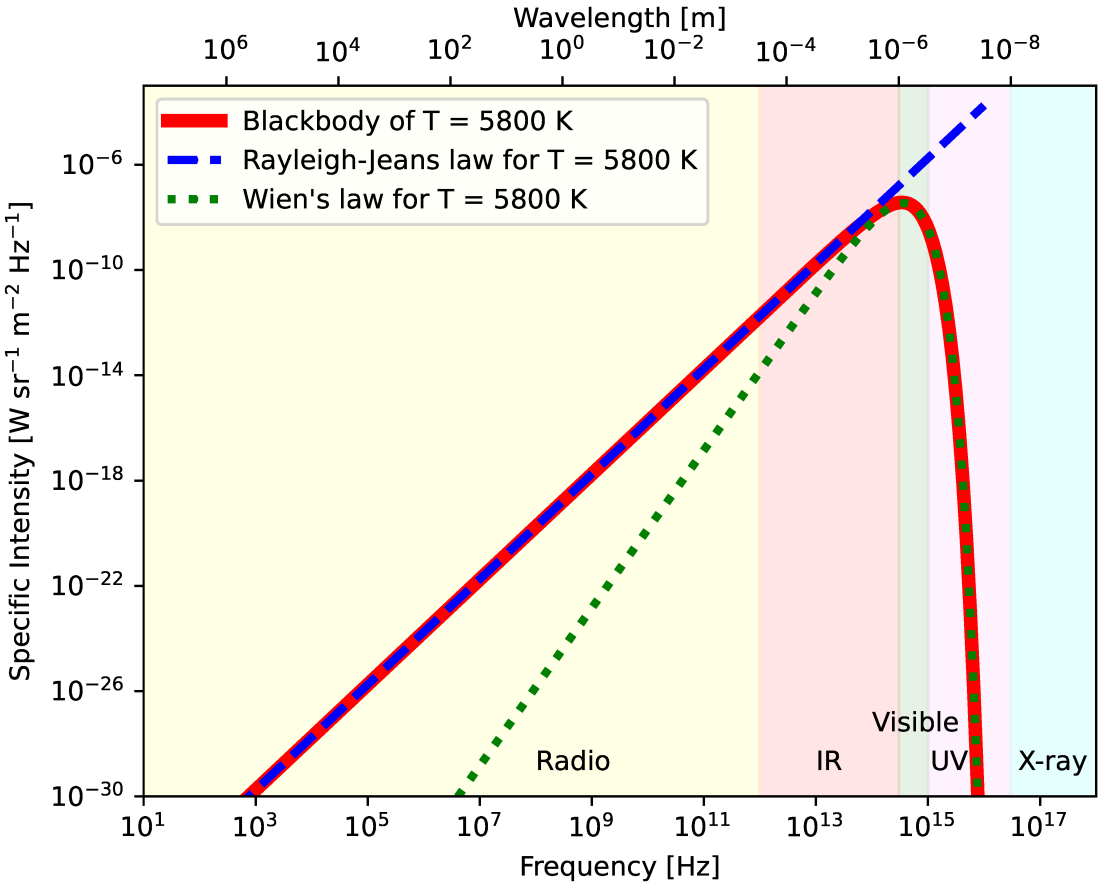

Figure 17: Log-log plot of model spectra of $T = 5800$ K blackbody as a function of frequency using Planck's radiation law, Rayleigh-Jeans law, and Wien's law. Frequency ranges of X-ray, UV, visible, IR, and radio are superimposed on the plot.

Try following practice.

Practice 09-15

Make a plot of $T = 25000$ K blackbody model spectrum as a function of frequency using Planck's radiation law, and make a secondary X-axis showing wavelength. Superimpose model spectra of $T = 25000$ K blackbody using Rayleigh-Jeans law and Wien's law. Superimpose frequency ranges of X-ray, UV, visible, IR, and radio.

2.8 Blackbody radiation of different temperature

Make a plot of blackbody radiation corresponding to temperature of 1, 10, 100, 1000, 10^4, 10^5, 10^6, 10^7, 10^8 K.

Python Code 18: ai202209_s09_00_17.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 20:09:09 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_00_17.png'

#
# function to calculate blackbody curve
#
def bb_nu (frequency, T):
    # speed of light
    c = scipy.constants.physical_constants['speed of light in vacuum']

    # Planck constant
    h = scipy.constants.physical_constants['Planck constant']

    # Boltzmann constant
    k = scipy.constants.physical_constants['Boltzmann constant']

    # calculation of Planck function
    blackbody = 2.0 * h[0] * frequency**3 / c[0]**2 \
        / (numpy.exp (h[0] * frequency / (k[0] * T) ) - 1.0 )

    # returning blackbody curve
    return (blackbody)

# temperature of blackbody
T = numpy.logspace (0.0, 8.0, num=9, dtype=numpy.longdouble)

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T} K')

# range of frequency (from 10**0 Hz to 10**16 Hz)
frequency_min = 2.0
frequency_max = 20.0

# frequency in Hz
frequency = numpy.logspace (frequency_min, frequency_max, num=16001, \
                            dtype=numpy.longdouble)

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)
```

```
# labels
ax.set_xlabel ('Frequency [Hz]')
ax.set_ylabel ('Specific Intensity [W sr$^{-1}$ m$^{-2}$ Hz$^{-1}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (10**3, numpy.array ([10**20], dtype=numpy.longdouble))
ax.set_ylim (10**-30, 10**6)

# make secondary X-axis
c   = scipy.constants.physical_constants['speed of light in vacuum'][0]
ax2 = ax.secondary_xaxis (location='top', \
                          functions=(lambda x: c/x, lambda x: c/x) )
ax2.set_xlabel ('Wavelength [m]')

# showing gamma-ray region
ax.fill_between (numpy.array ([3*10**19, 10**21], dtype=numpy.longdouble), \
                 10**-30, 10**8, \
                 color='magenta', alpha=0.1)
ax.text (x=4*10**19, y=10**-29, s='$\gamma$-ray')

# showing X-ray region
ax.fill_between (numpy.array ([3*10**16, 3*10**19], dtype=numpy.longdouble), \
                 10**-30, 10**8, \
                 color='cyan', alpha=0.1)
ax.text (x=3*10**17, y=10**-29, s='X-ray')

# showing UV region
ax.fill_between ([10**15, 3*10**16], 10**-30, 10**8, \
                 color='violet', alpha=0.1)
ax.text (x=2*10**15, y=10**-29, s='UV')

# showing visible region
ax.fill_between ([3*10**14, 10**15], 10**-30, 10**8, \
                 color='green', alpha=0.1)
ax.text (x=10**14, y=3*10**-28, s='Visible')

# showing IR region
ax.fill_between ([10**12, 3*10**14], 10**-30, 10**8, \
                 color='red', alpha=0.1)
ax.text (x=10**13, y=10**-29, s='IR')

# showing radio region
ax.fill_between ([10**0, 10**12], 10**-30, 10**8, \
                 color='yellow', alpha=0.1)
ax.text (x=10**8, y=10**-29, s='Radio')

# plotting data
for i in range (len (T)):
    # making model spectrum for given temperature using Planck's radiation law
    bb = bb_nu (frequency, T[i])
    # label
    text_label = f'T = {T[i]:g} K'
    ax.plot (frequency, bb, linestyle='-', linewidth=3, label=text_label)

# legend
ax.legend ()
```

```
# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to plot model spectra of 1, 10, 100, 1000, 10^4, 10^5, 10^6, 10^7, 10^8 K blackbody as a function of frequency.

```
% chmod a+x ai202209_s09_00_17.py
% ./ai202209_s09_00_17.py
Temperature:
  T = [1.e+00 1.e+01 1.e+02 1.e+03 1.e+04 1.e+05 1.e+06 1.e+07 1.e+08] K
```

Display generated PNG file. (18)

```
% feh -dF ai202209_s09_00_17.png
```

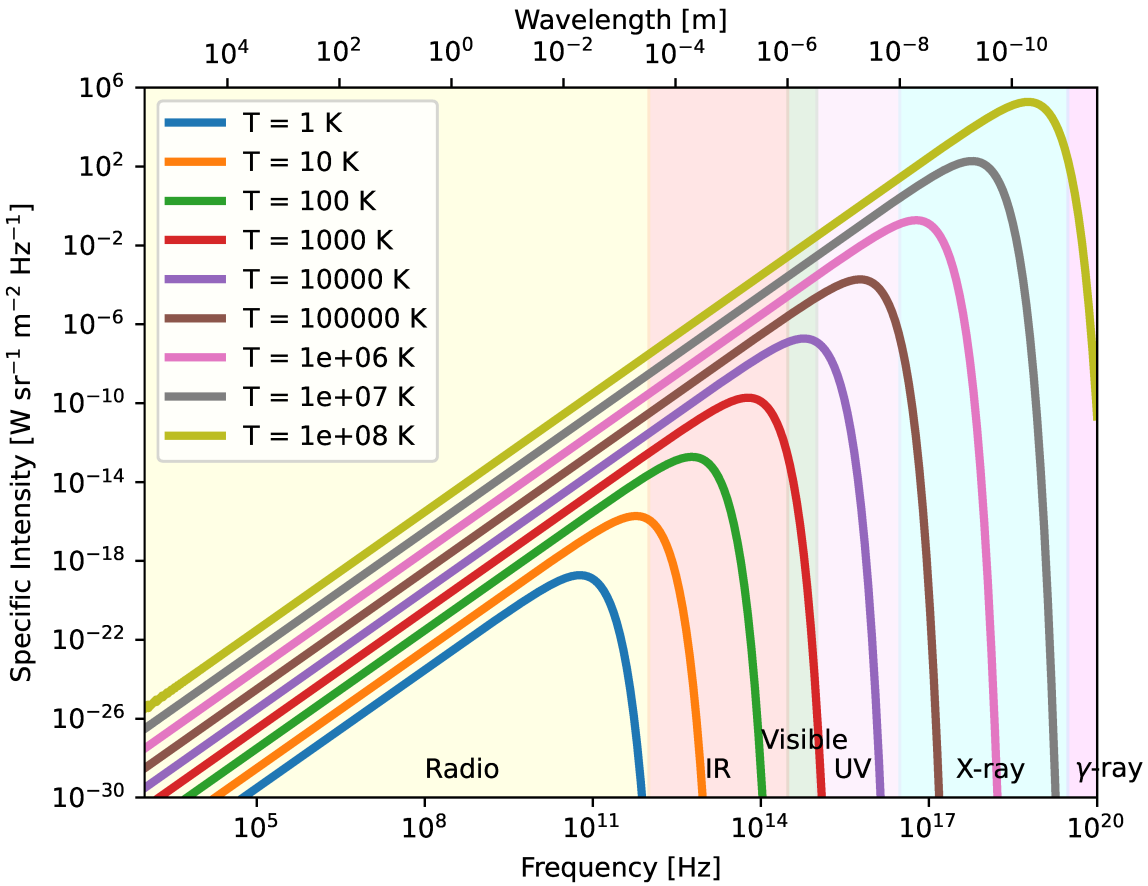

Figure 18: Log-log plot of model spectra of 1, 10, 100, 1000, 10^4, 10^5, 10^6, 10^7, 10^8 K blackbody as a function of frequency using Planck's radiation law, Rayleigh-Jeans law, and Wien's law. Frequency ranges of γ-ray, X-ray, UV, visible, IR, and radio are superimposed on the plot.

Try following practice.

Practice 09-16

Make a plot of 1, 10, 100, 1000, 10^4, 10^5, 10^6, 10^7, 10^8 K blackbody model spectrum as a function of wavelength using Planck's radiation law, and make a secondary X-axis showing frequency. Superimpose wavelength ranges of γ-ray, X-ray, UV, visible, IR, and radio.

3 Blackbody calculation using Astropy

Calculate blackbody radiation for temperature $T = 3000$ K using Astropy.

Python Code 19: ai202209_s09_01_00.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 20:37:35 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing astropy module
import astropy.modeling.models
import astropy.units

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# units
unit_K = astropy.units.K
unit_micron = astropy.units.micron

# temperature of blackbody
T = 3000.0 * unit_K

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T}')

# range of wavelength (from 10**-8 m to 10**-3 m)
wavelength_min = -8.0
wavelength_max = -3.0

# wavelength in micron
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001, \
                             dtype=numpy.longdouble) * 10**6 * unit_micron

# blackbody radiation
bb3000  = astropy.modeling.models.BlackBody (temperature=T)
bb_data = bb3000 (wavelength)

# printing blackbody radiation
print (f'Wavelength:')
print (f'{wavelength}')
print (f'Blackbody radiation:')
print (f'{bb_data}')
```

Execute above script to calculate blackbody radiation of temperature $T = 3000$ K.

```
% chmod a+x ai202209_s09_01_00.py
% ./ai202209_s09_01_00.py
Temperature:
```

```
  T = 3000.0 K
Wavelength:
[1.00000000e-02 1.00230524e-02 1.00461579e-02 ... 9.95405417e+02
 9.97700064e+02 1.00000000e+03] micron
Blackbody radiation:
[2.06452683e-206 6.17818767e-206 1.84416552e-205 ... 8.34041930e-011
 8.30214459e-011 8.26404542e-011] erg / (cm2 Hz s sr)
```

Try following practice.

Practice 09-17

Calculate blackbody radiation for temperature $T = 800$ K using Astropy.

Calculate blackbody radiation for temperature $T = 3000$ K using Astropy and make a plot.

Python Code 20: ai202209_s09_01_01.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 23:11:34 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing astropy module
import astropy.modeling.models
import astropy.units

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_01_01.png'

# units
unit_K      = astropy.units.K
unit_micron = astropy.units.micron

# temperature of blackbody
T = 3000.0 * unit_K

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T}')

# range of wavelength (from 10**-8 m to 10**-3 m)
wavelength_min = -8.0
wavelength_max = -3.0

# wavelength in micron
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001, \
                             dtype=numpy.longdouble) * 10**6 * unit_micron
```

```
# T = 3000 K blackbody radiation
bb3000  = astropy.modeling.models.BlackBody (temperature=T)
bb_data = bb3000 (wavelength)

# printing blackbody radiation
print (f'Wavelength:')
print (f'{wavelength}')
print (f'Blackbody radiation:')
print (f'{bb_data}')

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [$\mu$m]')
ax.set_ylabel ('Specific Intensity [erg sec$^{-1}$ sr$^{-1}$ cm$^{-2}$ Hz$^{-1}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (0.01, 100)
ax.set_ylim (10**-12, 10**-2)

# plotting data
ax.plot (wavelength, bb_data, linestyle='-', linewidth=3, \
         label='3000 K blackbody')

# legend
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to calculate blackbody radiation of temperature $T = 3000$ K and make a plot.

```
% chmod a+x ai202209_s09_01_01.py
% ./ai202209_s09_01_01.py
Temperature:
  T = 3000.0 K
Wavelength:
[1.00000000e-02 1.00230524e-02 1.00461579e-02 ... 9.95405417e+02
 9.97700064e+02 1.00000000e+03] micron
Blackbody radiation:
[2.06452683e-206 6.17818767e-206 1.84416552e-205 ... 8.34041930e-011
 8.30214459e-011 8.26404542e-011] erg / (cm2 Hz s sr)
```

Display generated PNG file. (19)

```
% feh -dF ai202209_s09_01_01.png
```

Try following practice.

Practice 09-18

Calculate blackbody radiation for temperature $T = 15000$ K using Astropy and make a plot.

Calculate blackbody radiation for temperature 3000, 6000, 12000, and 24000 K using Astropy and make a plot.

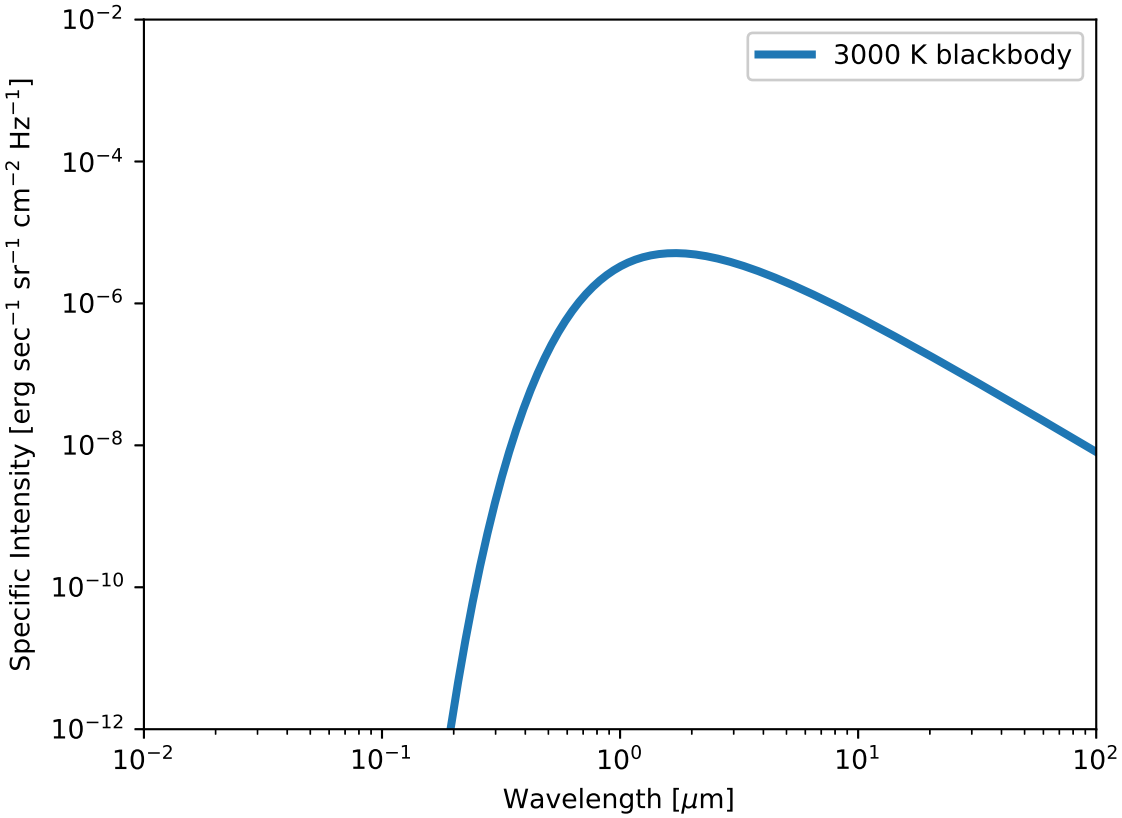

Figure 19: Log-log plot of model spectra of 3000 K blackbody as a function of wavelength using Astropy.

Python Code 21: ai202209_s09_01_02.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 23:11:51 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants

# importing astropy module
import astropy.modeling.models
import astropy.units

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# output file name
file_output = 'ai202209_s09_01_02.png'

# units
unit_K      = astropy.units.K
unit_micron = astropy.units.micron

# temperature of blackbody
T = [3000.0, 6000.0, 12000.0, 24000.0] * unit_K

# printing temperature of blackbody
print (f'Temperature:')
print (f'  T = {T}')

# range of wavelength (from 10**-8 m to 10**-3 m)
wavelength_min = -8.0
wavelength_max = -3.0

# wavelength in micron
wavelength = numpy.logspace (wavelength_min, wavelength_max, num=5001, \
                             dtype=numpy.longdouble) * 10**6 * unit_micron

# making objects "fig", "canvas", and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [$\mu$m]')
ax.set_ylabel ('Specific Intensity [erg sec$^{-1}$ sr$^{-1}$ cm$^{-2}$ Hz$^{-1}$]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
ax.set_xlim (0.01, 100)
ax.set_ylim (10**-12, 10**-2)

for i in range (len (T)):
```

```
    # blackbody radiation model
    bb_model = astropy.modeling.models.BlackBody (temperature=T[i])
    # blackbody radiation
    bb_data  = bb_model (wavelength)

    # label
    text_label = f'T = {T[i]}'

    # plotting data
    ax.plot (wavelength, bb_data, linestyle='-', linewidth=3, \
             label=text_label)

# legend
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to calculate blackbody radiation of temperature of 3000, 6000, 12000, and 24000 K and make a plot.

```
% chmod a+x ai202209_s09_01_02.py
% ./ai202209_s09_01_02.py
Temperature:
  T = [ 3000.  6000. 12000. 24000.] K
```

Display generated PNG file. (20)

```
% feh -dF ai202209_s09_01_02.png
```

Try following practice.

Practice 09-19

Calculate blackbody radiation for temperature of 2000, 4000, 8000, and 16000 K using Astropy and make a plot.

4 The solar spectrum

4.1 Downloading data

Download solar spectrum.

Python Code 22: ai202209_s09_02_00.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 23:25:06 (CST) daisuke>
#

# importing urllib module
import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context
```

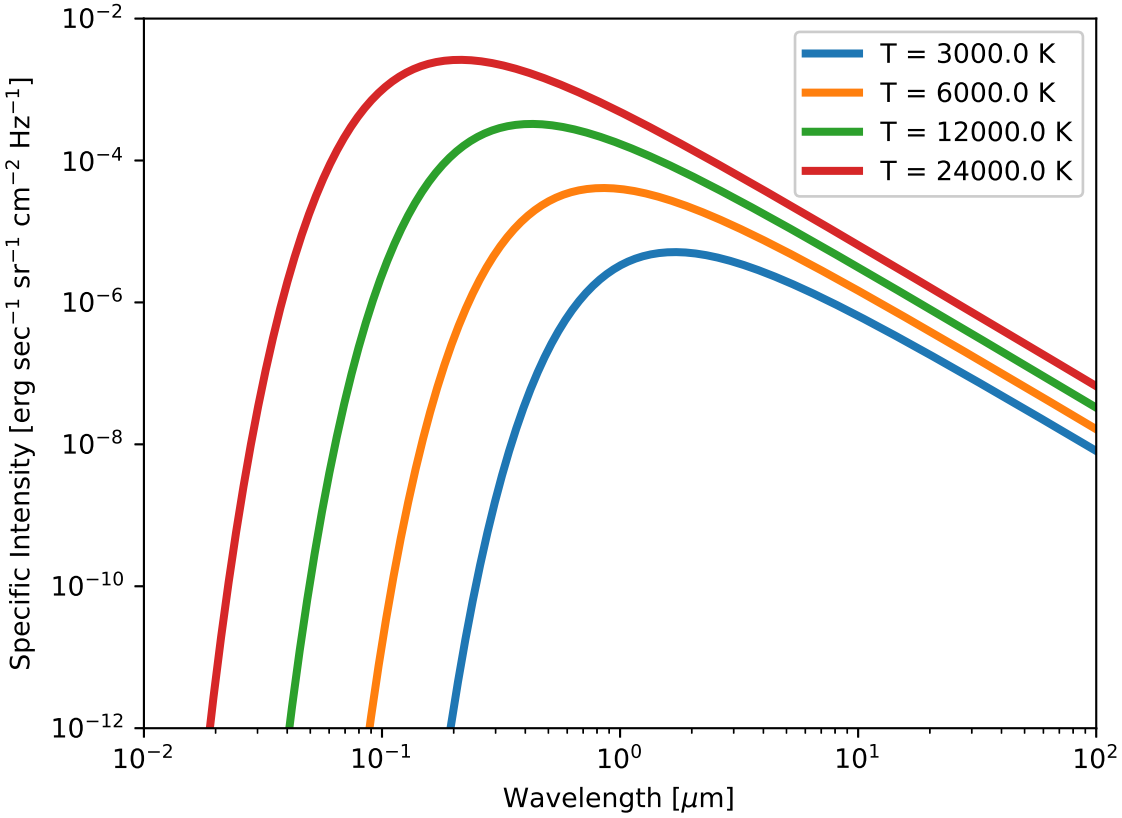

Figure 20: Log-log plot of model spectra of 3000, 6000, 12000, and 24000 K blackbody as a function of wavelength using Astropy.

```
# URL of data file
url_data = 'https://www.nrel.gov/grid/solar-resource/assets/data/newguey2003.txt'

# output file name
file_output = 'solar_spec.data'

# printing status
print (f'Fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()

# printing status
print (f'Fetched {url_data}!')

# converting raw byte data into string
data_str = data_byte.decode ('utf-8')

# printing status
print (f'Now, writing data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'w') as fh_write:
    # writing data
    fh_write.write (data_str)

# printing status
print (f'Finished writing data into file "{file_output}"!')
```

Execute above script to download solar spectrum.

```
% chmod a+x ai202209_s09_02_00.py
% ./ai202209_s09_02_00.py
Fetching https://www.nrel.gov/grid/solar-resource/assets/data/newguey2003.txt...
Fetched https://www.nrel.gov/grid/solar-resource/assets/data/newguey2003.txt!
Now, writing data into file "solar_spec.data"...
Finished writing data into file "solar_spec.data"!
% ls -lF solar_spec.data
-rw-r--r--  1 daisuke  taiwan  37030 Nov 13 23:25 solar_spec.data
```

4.2 Visualising solar spectrum

Make a Python script to visualise solar spectrum.

Python Code 23: ai202209_s09_02_01.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 23:33:09 (CST) daisuke>
#

# importing numpy module
import numpy

# importing matplotlib module
```

```
import matplotlib.figure
import matplotlib.backends.backend_agg

# data file name
file_input = 'solar_spec.data'

# figure file name
file_output = 'ai202209_s09_02_01.png'

# initialisation of numpy arrays for storing data
wl = numpy.array ([])
irradiance = numpy.array ([])

# opening file
with open (file_input, 'r') as fh:
    # initialisation of the parameter "i" for counting lines
    i = 0
    # reading data line-by-line
    for line in fh:
        # incrementing line number
        i += 1
        # skipping first 9 lines
        if (i < 10):
            continue
        # splitting data into wavelength and irradiance
        line = line.strip ()
        (wl_str, irradiance_str) = line.split ()
        # converting string into float
        wl_float = float (wl_str)
        irradiance_float = float (irradiance_str)
        # appending data to numpy arrays
        wl = numpy.append (wl, wl_float)
        irradiance = numpy.append (irradiance, irradiance_float)

# making objects "fig" and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [nm]')
ax.set_ylabel ('Irradiance [W m$^{-2}$ nm$^{-1}$]')

# axes
ax.set_xlim (100, 10000)
ax.set_ylim (0.0, 2.5)
ax.set_xscale ('log')

# plotting data
ax.plot (wl, irradiance, linestyle='-', color='r', linewidth=2, label='Sun')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to make a Python script to visualise solar spectrum.

```
% chmod a+x ai202209_s09_02_01.py
% ./ai202209_s09_02_01.py
```

Display generated PNG file. (21)

```
% feh -dF ai202209_s09_02_01.png
```

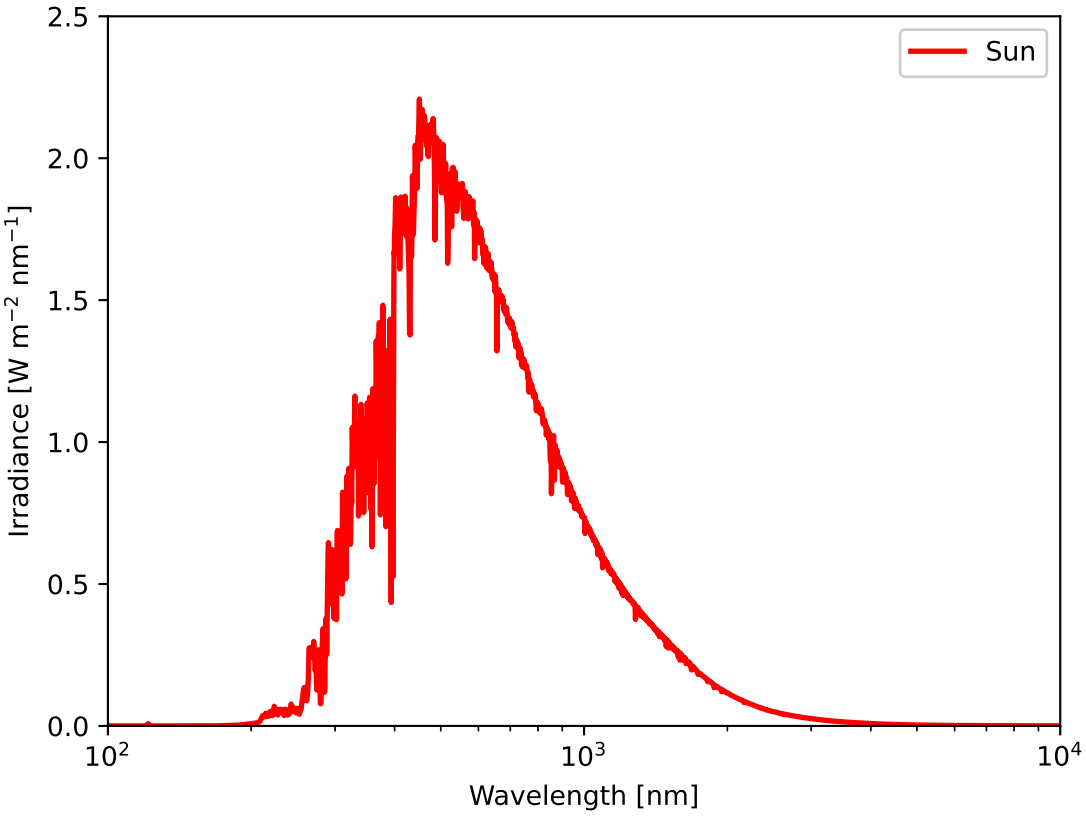

Figure 21: The solar spectrum.

4.3 Comparison with blackbody spectrum

Make a Python script to visualise solar spectrum and compare it with blackbody spectrum of $T = 5780$ K.

Python Code 24: ai202209_s09_02_02.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 23:43:24 (CST) daisuke>
#

# importing numpy module
import numpy

# importing astropy module
import astropy.modeling.models
import astropy.units

# importing matplotlib module
```

```
import matplotlib.figure
import matplotlib.backends.backend_agg

# data file name
file_input = 'solar_spec.data'

# figure file name
file_output = 'ai202209_s09_02_02.png'

# units
unit_W  = astropy.units.W
unit_m  = astropy.units.m
unit_nm = astropy.units.nm
unit_sr = astropy.units.sr
unit_K  = astropy.units.K

# initialisation of numpy arrays for storing data
wl = numpy.array ([])
irradiance = numpy.array ([])

# opening file
with open (file_input, 'r') as fh:
    # initialisation of the parameter "i" for counting lines
    i = 0
    # reading data line-by-line
    for line in fh:
        # incrementing line number
        i += 1
        # skipping first 9 lines
        if (i < 10):
            continue
        # splitting data into wavelength and irradiance
        line = line.strip ()
        (wl_str, irradiance_str) = line.split ()
        # converting string into float
        wl_float = float (wl_str)
        irradiance_float = float (irradiance_str)
        # appending data to numpy arrays
        wl = numpy.append (wl, wl_float)
        irradiance = numpy.append (irradiance, irradiance_float)

# wavelength
wl2_min = -7.0
wl2_max = -5.0
wl2 = numpy.logspace (wl2_min, wl2_max, 10**4) * 10**9 * unit_nm

# temperature
T = 5780.0 * unit_K

# blackbody radiation
bb = astropy.modeling.models.BlackBody \
    (temperature=T, \
     scale=7.0 * 10**-5 * unit_W / (unit_m**2 * unit_nm * unit_sr) )
bb_data = bb (wl2)

# making objects "fig" and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)
```

```
# labels
ax.set_xlabel ('Wavelength [nm]')
ax.set_ylabel ('Irradiance [W m$^{-2}$ nm$^{-1}$]')

# axes
ax.set_xlim (100, 10000)
ax.set_ylim (0.0, 2.5)
ax.set_xscale ('log')

# plotting data
ax.plot (wl, irradiance, linestyle='-', color='r', linewidth=2, label='Sun')
ax.plot (wl2, bb_data, linestyle=':', color='b', linewidth=2, \
         label='T=5780 K Blackbody')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to make a Python script to visualise solar spectrum and cmpare it with blackbody spectrum of $T = 5780$ K.

```
% chmod a+x ai202209_s09_02_02.py
% ./ai202209_s09_02_02.py
```

Display generated PNG file. (22)

```
% feh -dF ai202209_s09_02_02.png
```

5 Spectrum of HD 61005

5.1 Downloading data

Download Table 2 of following paper.

- "Resolving the Moth at Millimeter Wavelengths"
  - Ricarte et al., 2013, AJ, 774, 80.
  - `https://iopscience.iop.org/article/10.1088/0004-637X/774/1/80`

Python Code 25: ai202209_s09_03_00.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/13 23:47:14 (CST) daisuke>
#

# importing urllib module
import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context
```

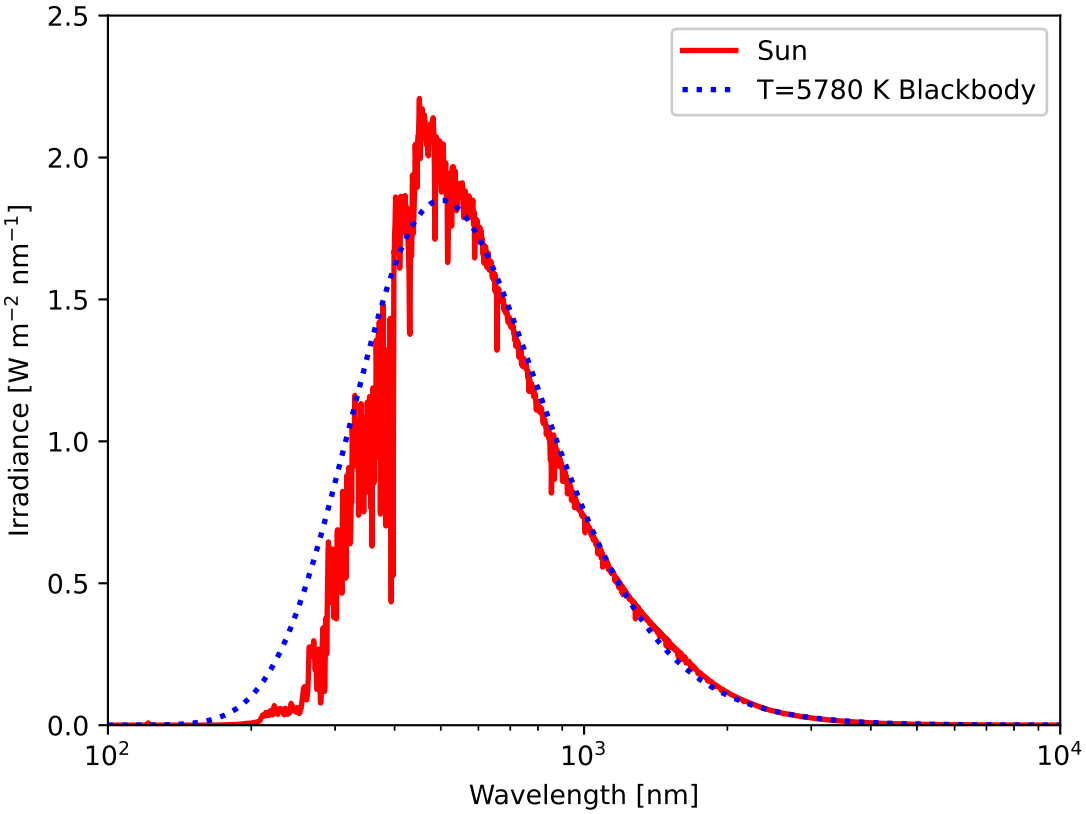

Figure 22: Comparison of the solar spectrum and $T = 5780$ K blackbody spectrum.

```
# URL of data file
url_data = 'https://iopscience.iop.org/0004-637X/774/1/80/suppdata/' \
    + 'apj480431t2_ascii.txt?doi=10.1088/0004-637X/774/1/80'

# output file name
file_output = 'hd61005_spec.data'

# printing status
print (f'Fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()

# printing status
print (f'Fetched {url_data}!')

# converting raw byte data into string
data_str = data_byte.decode ('utf-8')

# printing status
print (f'Now, writing data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'w') as fh_write:
    # writing data
    fh_write.write (data_str)

# printing status
print (f'Finished writing data into file "{file_output}"!')
```

Execute above script to download spectrum of HD 61005.

```
% chmod a+x ai202209_s09_03_00.py
% ./ai202209_s09_03_00.py
Fetching https://iopscience.iop.org/0004-637X/774/1/80/suppdata/apj480431t2_a
scii.txt?doi=10.1088/0004-637X/774/1/80...
Fetched https://iopscience.iop.org/0004-637X/774/1/80/suppdata/apj480431t2_as
cii.txt?doi=10.1088/0004-637X/774/1/80!
Now, writing data into file "hd61005_spec.data"...
Finished writing data into file "hd61005_spec.data"!
% head hd61005_spec.data
Table 2
Observed Flux Points for HD 61005


lambda  Flux    Source
(mum)   (Jy)
0.436   1.04 +or- 0.02  Tycho-2 (Hoslashg et al. 2000)
0.545   1.82 +or- 0.02  Tycho-2 (Hoslashg et al. 2000)
1.220   2.77 +or- 0.07  2MASS (Cutri et al. 2003)
1.630   2.39 +or- 0.10  2MASS (Cutri et al. 2003)
```

5.2 Visualising spectrum of HD 61005

Read the data file and visualise spectrum of HD 61005.

Python Code 26: ai202209_s09_03_01.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/14 00:02:58 (CST) daisuke>
#

# importing numpy module
import numpy

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# input file name
file_input = 'hd61005_spec.data'

# output file name
file_output = 'ai202209_s09_03_01.png'

# making empty numpy arrays
data_wl       = numpy.array ([])
data_flux     = numpy.array ([])
data_flux_err = numpy.array ([])

# opening data file
with open (file_input, 'r') as fh:
    # reading data line-by-line
    for line in fh:
        # if the word '+or-' is found, then we process the line
        if ('+or-' in line):
            # splitting data
            data = line.split ('+or-')
            # wavelength and flux
            (wl_str, flux_str) = data[0].split ()
            # error of flux
            flux_error_str = data[1].split ()[0]
            # conversion from string into float
            wl         = float (wl_str)
            flux       = float (flux_str)
            flux_error = float (flux_error_str)
            # appending data into numpy arrays
            data_wl       = numpy.append (data_wl, wl)
            data_flux     = numpy.append (data_flux, flux)
            data_flux_err = numpy.append (data_flux_err, flux_error)

# making objects "fig" and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [$\mu$m]')
ax.set_ylabel ('Flux [Jy]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
```

```
# plotting data
ax.errorbar (data_wl, data_flux, yerr=data_flux_err, \
             linestyle='', color='r', marker='o', markersize=5, \
             ecolor='black', capsize=3, label='HD61005')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to visualise spectrum of HD 61005.

```
% chmod a+x ai202209_s09_03_01.py
% ./ai202209_s09_03_01.py
```

Display generated PNG file. (23)

```
% feh -dF ai202209_s09_03_01.png
```

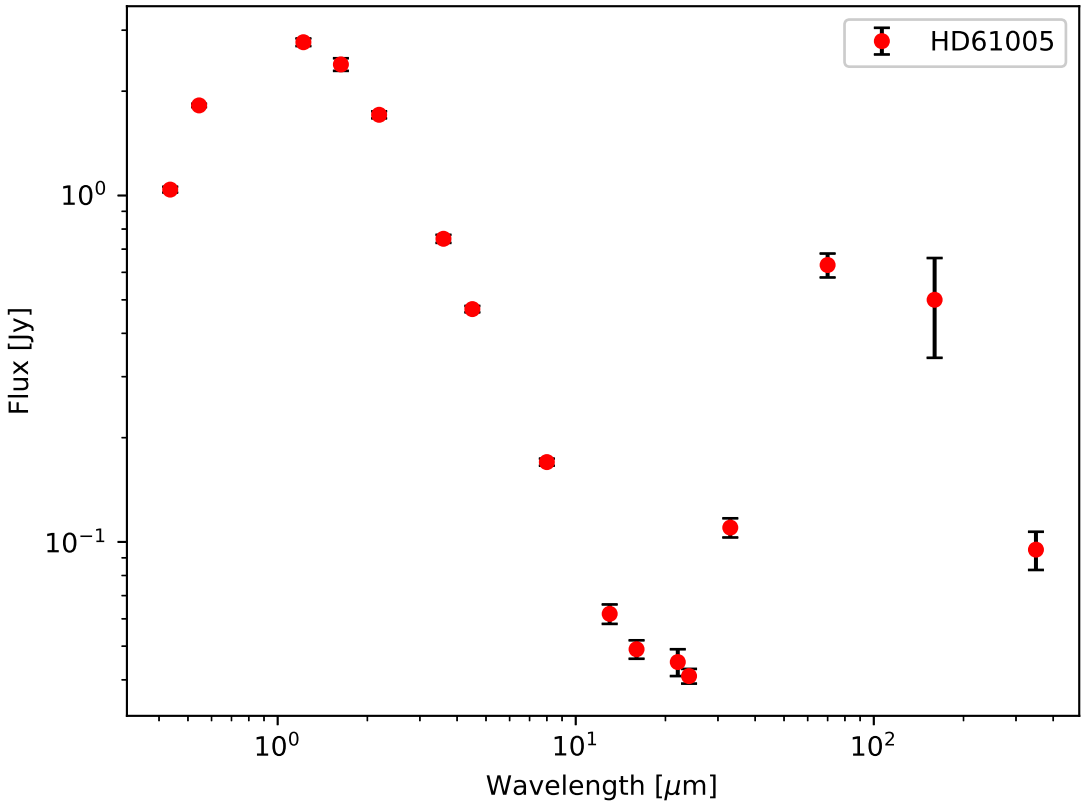

Figure 23: The spectrum of HD 61005.

5.3 SED fitting of HD 61005

Try SED fitting of HD 61005 using a combination of two blackbody curves.

Python Code 27: ai202209_s09_03_02.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/14 00:17:42 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants
import scipy.optimize

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# input file name
file_input = 'hd61005_spec.data'

# output file name
file_output = 'ai202209_s09_03_02.png'

# speed of light
c = scipy.constants.c

# Planck constant
h = scipy.constants.h

# Boltzmann constant
k = scipy.constants.k

# making empty numpy arrays
data_wl        = numpy.array ([])
data_flux      = numpy.array ([])
data_flux_err  = numpy.array ([])
data_wl1       = numpy.array ([])
data_flux1     = numpy.array ([])
data_flux_err1 = numpy.array ([])
data_wl2       = numpy.array ([])
data_flux2     = numpy.array ([])
data_flux_err2 = numpy.array ([])

# opening data file
with open (file_input, 'r') as fh:
    # reading data line-by-line
    for line in fh:
        # if the word '+or-' is found, then we process the line
        if ('+or-' in line):
            # splitting data
            data = line.split ('+or-')
            # wavelength and flux
            (wl_str, flux_str) = data[0].split ()
            # error of flux
            flux_error_str = data[1].split ()[0]
            # conversion from string into float
            wl         = float (wl_str)
            flux       = float (flux_str)
```

```
            flux_error = float (flux_error_str)
            # appending data into numpy arrays
            data_wl       = numpy.append (data_wl, wl)
            data_flux     = numpy.append (data_flux, flux)
            data_flux_err = numpy.append (data_flux_err, flux_error)
            if (wl < 20.0):
                data_wl1       = numpy.append (data_wl1, wl)
                data_flux1     = numpy.append (data_flux1, flux)
                data_flux_err1 = numpy.append (data_flux_err1, flux_error)
            if (wl > 30.0):
                data_wl2       = numpy.append (data_wl2, wl)
                data_flux2     = numpy.append (data_flux2, flux)
                data_flux_err2 = numpy.append (data_flux_err2, flux_error)

# initial values of coefficients of fitted function
T1 = 5000.0
T2 = 100.0
a1 = 10**6
a2 = 10**6
init1 = [T1, a1]
init2 = [T2, a2]

# function
def func (x, T, a):
    x_m = x * 10**-6
    f = c / x_m
    y = a * 2.0 * h * f**3 / c**2 / (numpy.exp (h * f / (k * T) ) - 1.0 )
    return (y)

# least-squares method
popt1, pcov1 = scipy.optimize.curve_fit (func, data_wl1, data_flux1, \
                                         p0=init1, sigma=data_flux_err1)
popt2, pcov2 = scipy.optimize.curve_fit (func, data_wl2, data_flux2, \
                                         p0=init2, sigma=data_flux_err2)

print (f"T1 = {popt1[0]} K")
print (f"T2 = {popt2[0]} K")

# fitted curve
wl_min = -0.4
wl_max = 2.7
n      = 10**4
fitted_x = numpy.logspace (wl_min, wl_max, n)
fitted_y = func (fitted_x, popt1[0], popt1[1]) \
    + func (fitted_x, popt2[0], popt2[1])

# making objects "fig" and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavelength [$\mu$m]')
ax.set_ylabel ('Flux [Jy]')

# axes
ax.set_xscale ('log')
ax.set_yscale ('log')
```

```
# plotting data
ax.plot (fitted_x, fitted_y, linestyle='-', color='b', linewidth=3, \
         label='Blackbody fitting')
ax.errorbar (data_wl, data_flux, yerr=data_flux_err, \
             linestyle='', color='r', marker='o', markersize=5, \
             ecolor='black', capsize=3, label='HD61005')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to visualise spectrum of HD 61005 and show the result of SED fitting.

```
% chmod a+x ai202209_s09_03_02.py
% ./ai202209_s09_03_02.py
T1 = 5478.437152088909 K
T2 = 61.8668397211255 K
```

Display generated PNG file. (24) The spectrum is well fitted by a combination of $T \sim 5500$ K and $T \sim 60$ K blackbody.

```
% feh -dF ai202209_s09_03_02.png
```

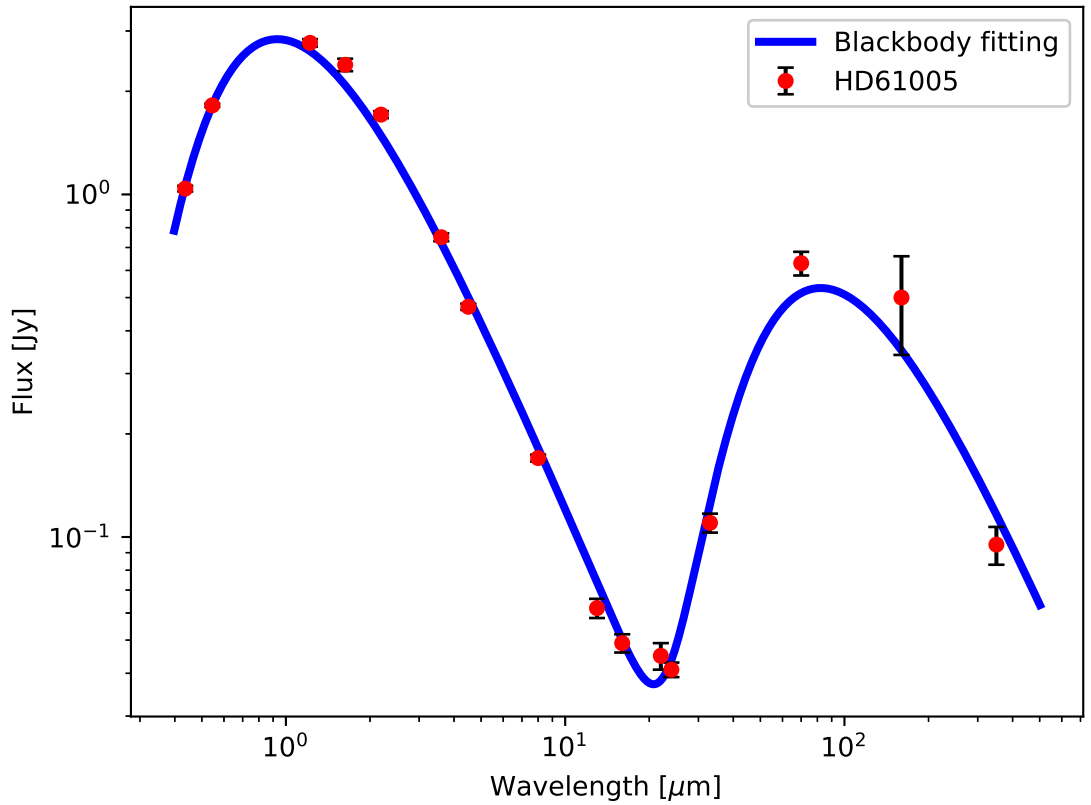

Figure 24: The spectrum of HD 61005 and result of SED fitting.

6 Cosmic Microwave Background

6.1 Downloading COBE/FIRAS data

Make a Python script to download CMB data from COBE/FIRAS measurements.

Python Code 28: ai202209_s09_04_00.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/14 00:20:31 (CST) daisuke>
#

# importing urllib module
import urllib.request

# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# URL of data file
url_data = 'https://lambda.gsfc.nasa.gov/data/cobe/firas/monopole_spec/' \
    + 'firas_monopole_spec_v1.txt'

# output file name
file_output = 'cmb_cobe.data'

# printing status
print (f'Fetching {url_data}...')

# opening URL
with urllib.request.urlopen (url_data) as fh_read:
    # reading data
    data_byte = fh_read.read ()

# printing status
print (f'Fetched {url_data}!')

# converting raw byte data into string
data_str = data_byte.decode ('utf-8')

# printing status
print (f'Now, writing data into file "{file_output}"...')

# opening file for writing
with open (file_output, 'w') as fh_write:
    # writing data
    fh_write.write (data_str)

# printing status
print (f'Finished writing data into file "{file_output}"!')
```

Execute above script to download CMB data from COBE/FIRAS measurements.

```
% chmod a+x ai202209_s09_04_00.py
% ./ai202209_s09_04_00.py
Fetching https://lambda.gsfc.nasa.gov/data/cobe/firas/monopole_spec/firas_mon
```

```
opole_spec_v1.txt...
Fetched https://lambda.gsfc.nasa.gov/data/cobe/firas/monopole_spec/firas_mono
pole_spec_v1.txt!
Now, writing data into file "cmb_cobe.data"...
Finished writing data into file "cmb_cobe.data"!
```

6.2 Visualising CMB spectrum

Visualise CMB spectrum.

Python Code 29: ai202209_s09_04_01.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/14 00:38:06 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants
import scipy.optimize

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# input file name
file_input = 'cmb_cobe.data'

# output file name
file_output = 'ai202209_s09_04_01.png'

#
# constants
#

# speed of light
c = scipy.constants.c
# Planck constant
h = scipy.constants.h
# Boltzmann constant
k = scipy.constants.k

# numpy arrays for storing data
data_freq_kayser = numpy.array ([])
data_intensity   = numpy.array ([])
data_residual    = numpy.array ([])
data_uncertainty = numpy.array ([])
data_galspec     = numpy.array ([])

# opening file
with open (file_input, 'r') as fh:
    # reading data file
    for line in fh:
        # skip if the line starts with '#'.
        if (line[0] == '#'):
```

```
            continue
        # splitting the data
        (freq_kayser_str, intensity_str, residual_str, uncertainty_str,
         galspec_str) = line.split ()
        # conversion from string into float
        freq_kayser = float (freq_kayser_str)
        intensity   = float (intensity_str)
        residual    = float (residual_str)
        uncertainty = float (uncertainty_str)
        galspec     = float (galspec_str)
        # appending data to numpy arrays
        data_freq_kayser = numpy.append (data_freq_kayser, freq_kayser)
        data_intensity   = numpy.append (data_intensity, intensity)
        data_residual    = numpy.append (data_residual, residual)
        data_uncertainty = numpy.append (data_uncertainty, uncertainty)
        data_galspec     = numpy.append (data_galspec, galspec)

# conversion from wavenumber into wavelength
data_wavelength_mm = 10.0 / data_freq_kayser
data_wavelength_m  = data_wavelength_mm / 10**3

# making objects "fig" and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
ax.set_xlabel ('Wavenumber [cm$^{-1}$]')
ax.set_ylabel ('Intensity [MJy sr$^{-1}$]')

# axes
ax.set_xlim (0.0, 23.0)
ax.set_ylim (0.0, 500.0)

# plotting data
ax.errorbar (data_freq_kayser, data_intensity, yerr=data_uncertainty, \
             linestyle='', marker='o', color='r', markersize=5, \
             ecolor='black', capsize=3, label='CMB measured by COBE/FIRAS')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to visualise CMB spectrum.

```
% chmod a+x ai202209_s09_04_01.py
% ./ai202209_s09_04_01.py
```

Display generated PNG file. (25)

```
% feh -dF ai202209_s09_04_01.png
```

6.3 Blackbody fitting to CMB spectrum

Use least-squares method to fit the CMB spectrum with blackbody curve.

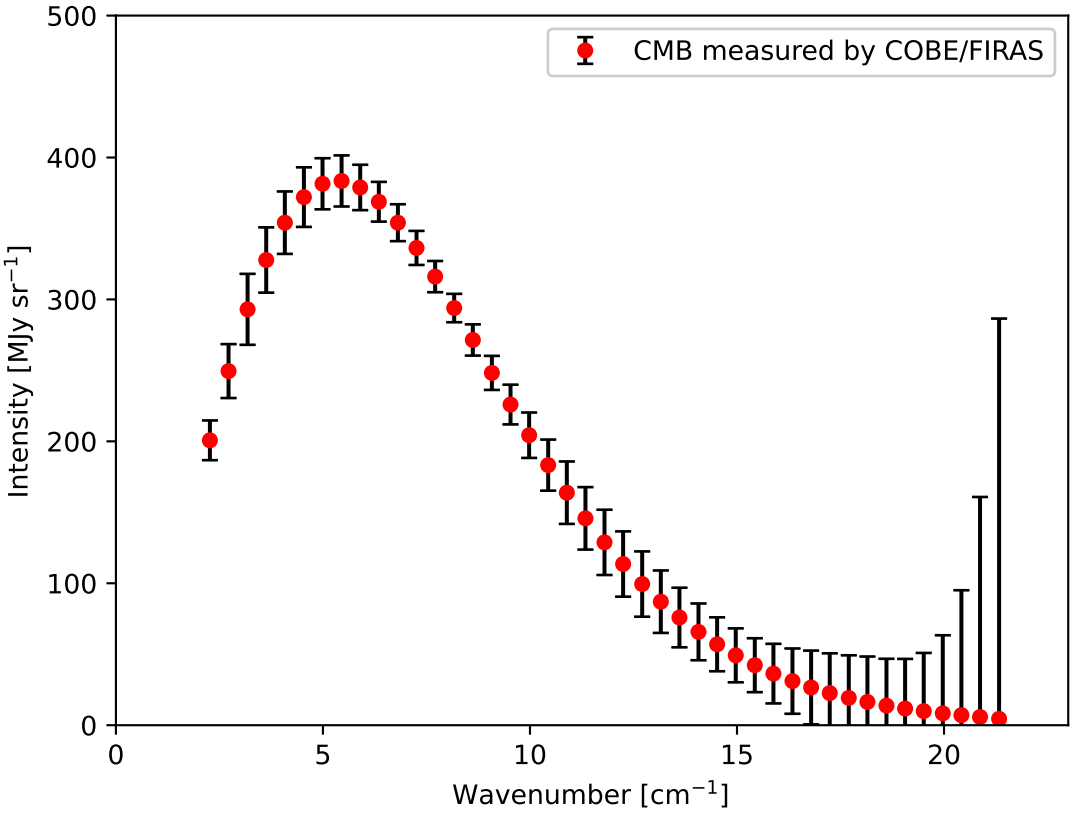

Figure 25: The CMB spectrum.

Python Code 30: ai202209_s09_04_02.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/11/14 00:36:15 (CST) daisuke>
#

# importing numpy module
import numpy

# importing scipy module
import scipy.constants
import scipy.optimize

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg

# input file name
file_input = 'cmb_cobe.data'

# output file name
file_output = 'ai202209_s09_04_02.png'

#
# constants
#

# speed of light
c = scipy.constants.c
# Planck constant
h = scipy.constants.h
# Boltzmann constant
k = scipy.constants.k

# numpy arrays for storing data
data_freq_kayser = numpy.array ([])
data_intensity   = numpy.array ([])
data_residual    = numpy.array ([])
data_uncertainty = numpy.array ([])
data_galspec     = numpy.array ([])

# opening file
with open (file_input, 'r') as fh:
    # reading data file
    for line in fh:
        # skip if the line starts with '#'.
        if (line[0] == '#'):
            continue
        # splitting the data
        (freq_kayser_str, intensity_str, residual_str, uncertainty_str,
         galspec_str) = line.split ()
        # conversion from string into float
        freq_kayser = float (freq_kayser_str)
        intensity   = float (intensity_str)
        residual    = float (residual_str)
        uncertainty = float (uncertainty_str)
        galspec     = float (galspec_str)
        # appending data to numpy arrays
```

```
        data_freq_kayser = numpy.append (data_freq_kayser, freq_kayser)
        data_intensity   = numpy.append (data_intensity, intensity)
        data_residual    = numpy.append (data_residual, residual)
        data_uncertainty = numpy.append (data_uncertainty, uncertainty)
        data_galspec     = numpy.append (data_galspec, galspec)

# conversion from wavenumber into wavelength
data_wavelength_mm = 10.0 / data_freq_kayser
data_wavelength_m  = data_wavelength_mm / 10**3

# initial values of coefficients of fitted function
T    = 10.0
a    = 10**10
init = [T, a]

# blackbody function
def func (x, T, a):
    x_mm = 10.0 / x
    x_m = x_mm * 10**-3
    f = c / x_m
    y = a * 2.0 * h * f**3 / c**2 / (numpy.exp (h * f / (k * T) ) - 1.0 )
    return (y)

# least-squares method
popt, pcov = scipy.optimize.curve_fit (func, data_freq_kayser, \
                                       data_intensity, \
                                       p0=init, sigma=data_uncertainty)

print ("popt:")
print (popt)

print ("pcov:")
print (pcov)

# dof
dof = len (data_freq_kayser) - len (init)
print ("dof =", dof)

# residual
residual = data_intensity - func (data_freq_kayser, popt[0], popt[1])
reduced_chi2 = (residual**2).sum () / dof
print ("reduced chi^2 =", reduced_chi2)

# errors of T and a
T_err = numpy.sqrt (pcov[0][0])
a_err = numpy.sqrt (pcov[1][1])
print (f"T = {popt[0]:g} +/- {T_err:g} ({T_err / popt[0] * 100.0} %)")
print (f"a = {popt[1]:g} +/- {a_err:g} ({a_err / popt[1] * 100.0} %)")

# fitted curve
fitted_x = numpy.linspace (0.1, 23.0, 10000)
fitted_y = func (fitted_x, popt[0], popt[1])

# making objects "fig" and "ax"
fig    = matplotlib.figure.Figure ()
canvas = matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax     = fig.add_subplot (111)

# labels
```

```
ax.set_xlabel ('Wavenumber [cm$^{-1}$]')
ax.set_ylabel ('Intensity [MJy sr$^{-1}$]')

# axes
ax.set_xlim (0.0, 23.0)
ax.set_ylim (0.0, 500.0)

# plotting data
ax.plot (fitted_x, fitted_y, linestyle='-', color='b', linewidth=3, \
         label='Blackbody')
ax.errorbar (data_freq_kayser, data_intensity, yerr=data_uncertainty, \
             linestyle='', marker='o', color='r', markersize=5, \
             ecolor='black', capsize=3, label='CMB measured by COBE/FIRAS')
ax.legend ()

# saving the plot into a file
fig.savefig (file_output, dpi=225)
```

Execute above script to fit the CMB spectrum with blackbody curve using least-squares method.

```
% chmod a+x ai202209_s09_04_02.py
% ./ai202209_s09_04_02.py
popt:
[2.72500853e+00 1.00001009e+20]
pcov:
[[ 9.59223968e-10 -1.32931264e+11]
 [-1.32931264e+11  1.97186404e+31]]
dof = 41
reduced chi^2 = 0.00537259435992668
T = 2.72501 +/- 3.09713e-05 (0.0011365594172638243 %)
a = 1.00001e+20 +/- 4.44057e+15 (0.00444052278368339 %)
```

Display generated PNG file. (26)

```
% feh -dF ai202209_s09_04_02.png
```

7 For your further reading

Read following document to learn more about Astropy.

- Astropy: `https://docs.astropy.org/en/stable/`
  - Models and Fitting: `https://docs.astropy.org/en/stable/modeling/`
    - Physical Models: `https://docs.astropy.org/en/stable/modeling/physical_models.html`

8 Assignment

1. Planck's radiation law is written as

$$B_\nu(T) = \frac{2h\nu^3}{c^2}\frac{1}{\exp\left(\frac{h\nu}{kT}\right)-1}.$$

   Show $B_\lambda(T)$ is written as

$$B_\lambda(T) = \frac{2hc^2}{\lambda^5}\frac{1}{\exp\left(\frac{hc}{\lambda kT}\right)-1}.$$

2. Show the derivation of the Wien's displacement law from Planck's radiation law.

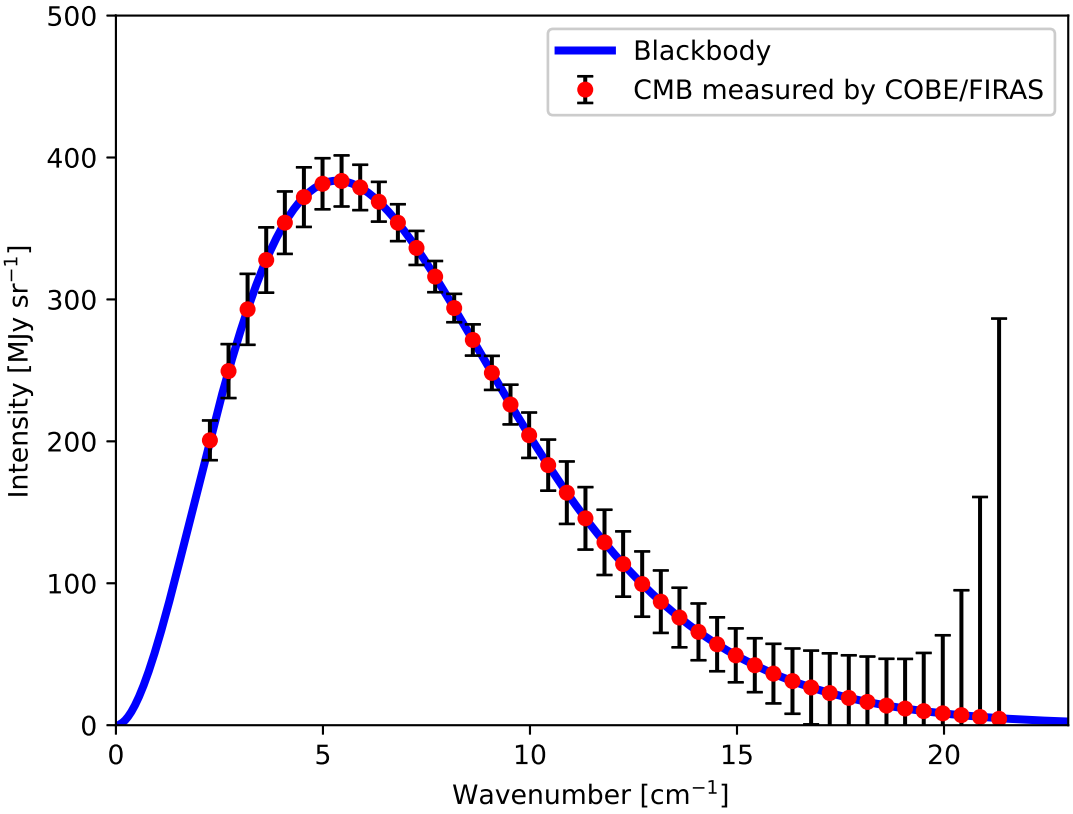

Figure 26: The CMB spectrum and result of blackbody fitting.

3. Show the derivation of Rayleigh-Jeans law from Planck's radiation law.
4. Show the derivation of Wien's law from Planck's radiation law.
5. HD61005
   (a) What is the spectral type of HD61005?
   (b) Explain why we see two peaks in the spectral energy distribution of HD61005?
   (c) Give one more example of stars that exhibit similar infrared excess in the spectral energy distribution. Give basic description of the star.
6. Fomalhaut
   (a) Read following paper.
      - "Herschel images of Fomalhaut: An extrasolar Kuiper belt at the height of its dynamical activity", Acke et al., 2012, A&A, 540, A125.
   (b) Plot the SED of Fomalhaut.
   (c) Carry out least-squares method to fit the data using Planck's radiation law.
   (d) What are your temperature estimates for the photosphere of the star and debris disk?
   (e) Show all the Python codes you made.