Exercise 3 Problem 2

March 18, 2020

1 Problem statement:

a) What is the photon spectrum [1/keV/cm2/s] for a tungsten anode x-ray tube for a peak energy of 100 keV, 10 keV minimum energy, 30° anode angle at 1m (air) with a 2 mm aluminum filter and a current of 400 mA?
b) Describe briefly the effect of increasing the aluminum filter thickness on the x-ray flux.
c) Using the spectrum from a), calculate the spectrum averaged attenuation coefficient for steel (iron), aluminum and water.

Hint: Download "SpekCalc light" from spekcalc.weebly.com and use the NIST-library (https://physics.nist.gov).

- steel: density = 8.05 g/cm3; Standard atomic weight = 55.845 g/mol
- aluminum: density = 2.70 g/cm3; Standard atomic weight = 26.981 g/mol
- water: density = 0.998 g/cm3; Standard atomic weight = 18 g/mol

[4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os, sys, glob
import re
import matplotlib
from matplotlib import pyplot as plt
from matplotlib.ticker import AutoMinorLocator
from scipy.integrate import simps
from scipy import interpolate
```

[5]:
```
os.environ["PATH"] += os.pathsep + '/usr/local/texlive/2018/bin/x86_64-darwin'
plt.rc('text', usetex=True)
plt.rc('font', weight='bold')
matplotlib.rcParams['mathtext.fontset'] = 'custom'
matplotlib.rcParams['mathtext.rm'] = 'Arial'
matplotlib.rcParams['mathtext.it'] = 'Arial:italic'
matplotlib.rcParams['mathtext.bf'] = 'Arial:bold'
matplotlib.rcParams['mathtext.tt'] = 'Arial'
matplotlib.rcParams['mathtext.cal'] = 'Arial'
matplotlib.rcParams['text.latex.preamble'] = [r'\usepackage{sfmath} \boldmath']
```

2 a)

What is the photon spectrum [1/keV/cm2/s] for a tungsten anode x-ray tube for a peak energy of 100 keV, 10 keV minimum energy, 30° anode angle at 1m (air) with a 2 mm aluminum filter and a current of 400 mA?

Use SpekCalc light (note that you need to change the filter thickness to 2 mm and e.g finer energy binning to 0.1 keV) obtain the following spectrum:

Exporting the data from a) via "View Data" and importing into a software package (i.e. Matlab). Results as csv in "01_spectrum_raw.csv".

[6]:
```
# SpekCalcFile with 0.1 kev energy binning
spek_file = './01_spectrum_raw.csv'

# import data
df_spek = pd.read_csv(spek_file, skiprows=1, index_col=0, header=None,␣
 ↪delimiter=',')
df_spek.columns=['Energy_keV', 'phi_cm2_s_mA_keV']
df_spek.head()
```

[6]:

	Energy_keV	phi_cm2_s_mA_keV
0		
0	10.0	13.44400
1	10.1	18.10739
2	10.2	24.40184
3	10.3	32.94153
4	10.4	44.54384

2.0.1 Multiply for each bin with the current of 400 mA to compute the spectrum in units of [1/keV/cm2/s]:

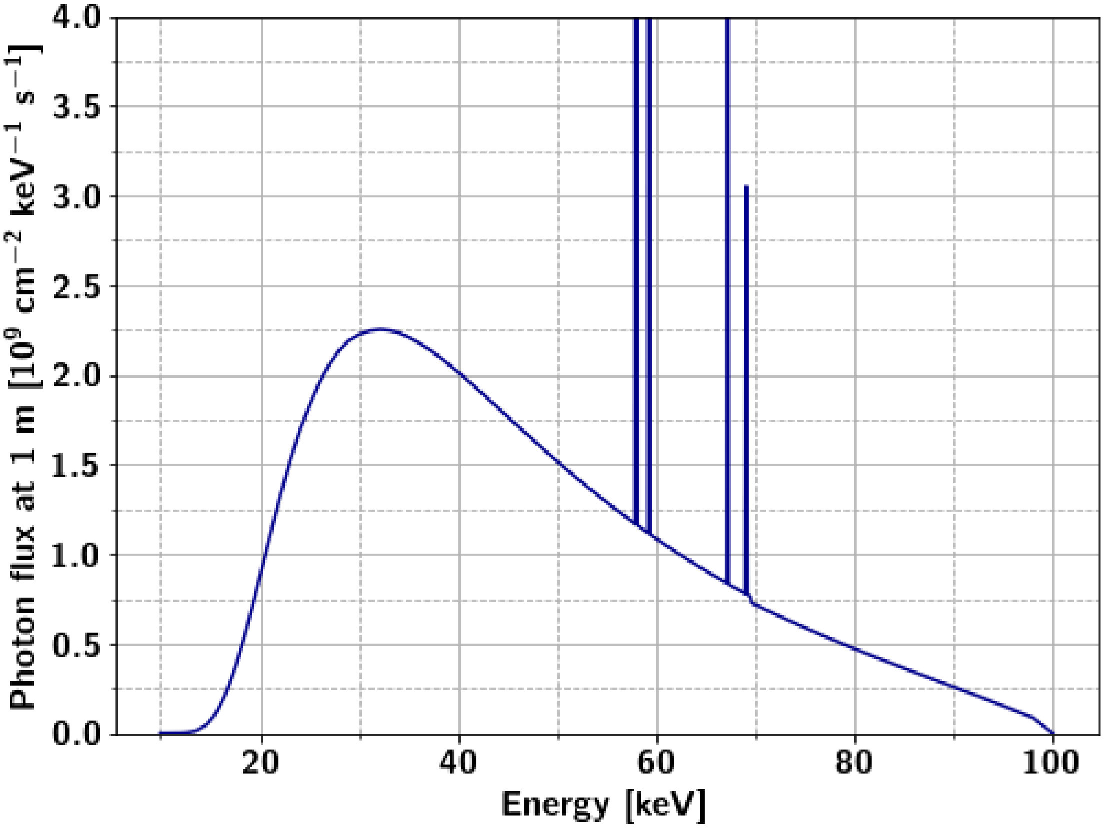
[7]:
```
# account for mA
tube_current = 400  # mA
df_spek['phi_cm2_s_keV'] = df_spek['phi_cm2_s_mA_keV'] * tube_current
df_spek.head()
```

[7]:

	Energy_keV	phi_cm2_s_mA_keV	phi_cm2_s_keV
0			
0	10.0	13.44400	5377.600
1	10.1	18.10739	7242.956
2	10.2	24.40184	9760.736
3	10.3	32.94153	13176.612
4	10.4	44.54384	17817.536

[23]:

```
fs = 16
fig, ax = plt.subplots(figsize=(8, 6))
plt.plot(df_spek['Energy_keV'].values, df_spek['phi_cm2_s_keV'].values/1e9,␣
 ↪color='darkblue')
plt.ylabel(r'\textbf{Photon flux at 1 m [10$^9$ cm$^{-2}$ keV$^{-1}$␣
 ↪s$^{-1}$]}', fontsize=fs)
plt.xlabel(r'\textbf{Energy [keV]}', fontsize=fs)
plt.ylim(0, 4)

ax.tick_params(axis = 'both', which = 'major', labelsize = fs)
# minor ticks x
minor_locator = AutoMinorLocator(2)
ax.xaxis.set_minor_locator(minor_locator)
# minor ticks y
minor_locator = AutoMinorLocator(2)
ax.yaxis.set_minor_locator(minor_locator)
# grid
ax.grid(b=True, which='major', linestyle='-')#, color='gray')
ax.grid(b=True, which='minor', linestyle='--')#, color='gray')
plt.show()
plt.close()

df_spek[['Energy_keV', 'phi_cm2_s_keV']].to_csv('02_spectrum.csv',␣
 ↪header=['Energy [keV]', 'Photon flux at 1 m [1/cm2/keV/s]'])
```



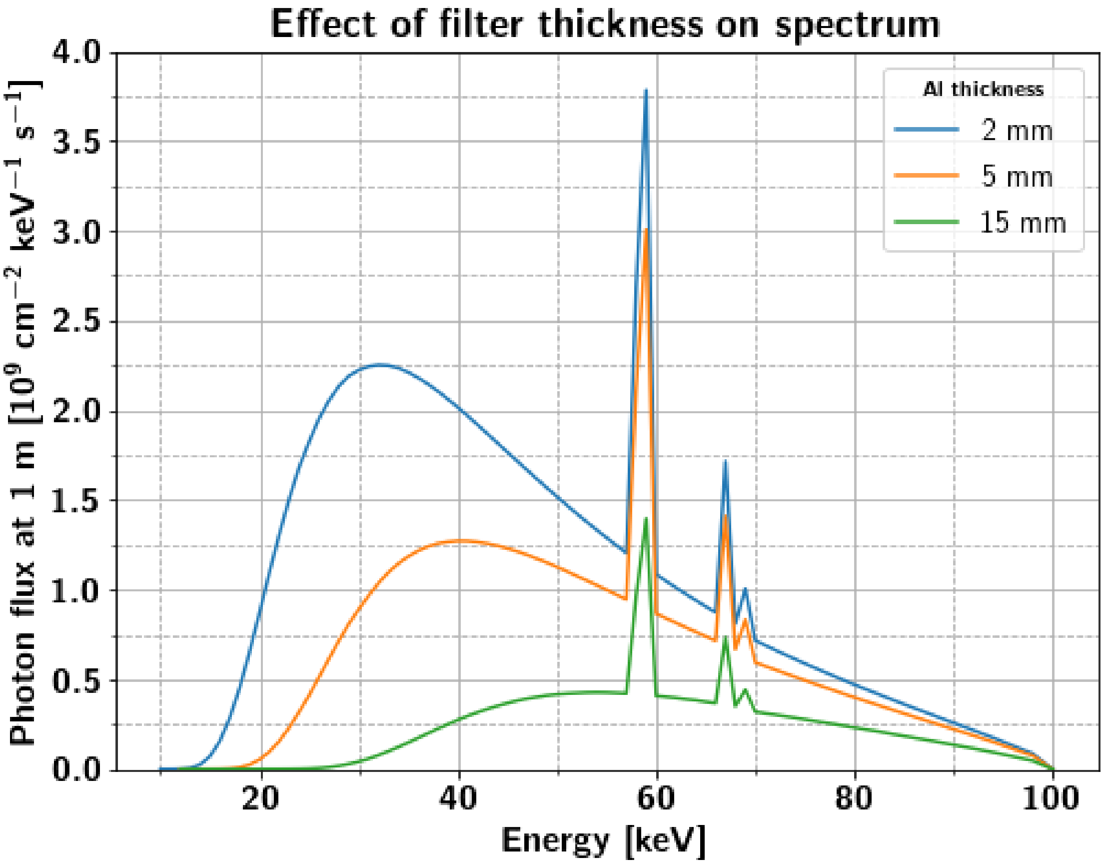
Spectrum is exported as "02_spectrum.csv".

3 b)

Describe briefly the effect of increasing the aluminum filter thickness on the x-ray flux.

[29]:
```
files = ['spectrum_Al_2mm.txt', 'spectrum_Al_5mm.txt', 'spectrum_Al_15mm.txt']
lst_df = []
for file in files:
    al_thickness = re.findall(r'(\d*)mm', file)[0]
    # import data
    df = pd.read_csv(file, skiprows=2, header=None, delimiter='\s+')
    df.columns=['Energy_keV', 'phi_cm2_s_mA_keV']
    # account for mA
    tube_current = 400  # mA
    df['phi_cm2_s_keV'] = df['phi_cm2_s_mA_keV'] * tube_current
    df['label'] = f'{al_thickness} mm'
    lst_df.append(df)
```

```
fs = 16
plt.figure(figsize=(8, 6))
for df in lst_df:
    plt.plot(df['Energy_keV'], df['phi_cm2_s_keV'].values/1e9,␣
 ↪label=df['label'].unique()[0])
plt.title(r'\textbf{Effect of filter thickness on spectrum}', fontsize=fs+2)
plt.ylabel(r'\textbf{Photon flux at 1 m [10$^9$ cm$^{-2}$ keV$^{-1}$␣
 ↪s$^{-1}$]}', fontsize=fs)
plt.xlabel(r'\textbf{Energy [keV]}', fontsize=fs)
plt.ylim(0, 4)
plt.legend(title=r'\textbf{Al thickness}', fontsize=fs-2)
ax = plt.gca()
ax.tick_params(axis = 'both', which = 'major', labelsize = fs)
ax.tick_params(axis = 'both', which = 'major', labelsize = fs)
# minor ticks x
minor_locator = AutoMinorLocator(2)
ax.xaxis.set_minor_locator(minor_locator)
# minor ticks y
minor_locator = AutoMinorLocator(2)
ax.yaxis.set_minor_locator(minor_locator)
# grid
ax.grid(b=True, which='major', linestyle='-')#, color='gray')
ax.grid(b=True, which='minor', linestyle='--')#, color='gray')
plt.show()
plt.close()
```



—> The spectrum gets harder with increasing filter thickness, but at the same time the total intensity also decreases. There is a tradeoff between hardness and total intensity.

4 c)

Using the spectrum from a), calculate the spectrum averaged attenuation coefficient for steel (iron), aluminum and water.

—> Use the spectrum from a).

- Normalize the spectrum $\phi(E)$ to obtain a distribution function $\phi_{norm}(E)$:

$$\phi_{norm}(E) = \frac{\phi(E)}{\int_0^\infty \phi(E)dE}$$

- Check the normalization, so that $\int_0^\infty \phi_{norm}(E)dE = 1$

[30]:
```
# normalization by the area under the curve, energy binning is 0.1 keV!
X = df_spek['Energy_keV'].values
Y = df_spek['phi_cm2_s_keV'].values
```

```python
area_tot = simps(Y, X)
# print(area_tot, X)
df_spek['phi_norm'] = df_spek['phi_cm2_s_keV']/area_tot

# check the normalization
Y = df_spek['phi_norm'].values
print(simps(Y, X))
df_spek.head()
```

1.0

[30]:

	Energy_keV	phi_cm2_s_mA_keV	phi_cm2_s_keV	phi_norm
0				
0	10.0	13.44400	5377.600	5.526965e-08
1	10.1	18.10739	7242.956	7.444132e-08
2	10.2	24.40184	9760.736	1.003184e-07
3	10.3	32.94153	13176.612	1.354260e-07
4	10.4	44.54384	17817.536	1.831243e-07

Spectrum averaged photon energy (not required):

$$<E>=\int_0^\infty E*\phi_{norm}(E)dE$$

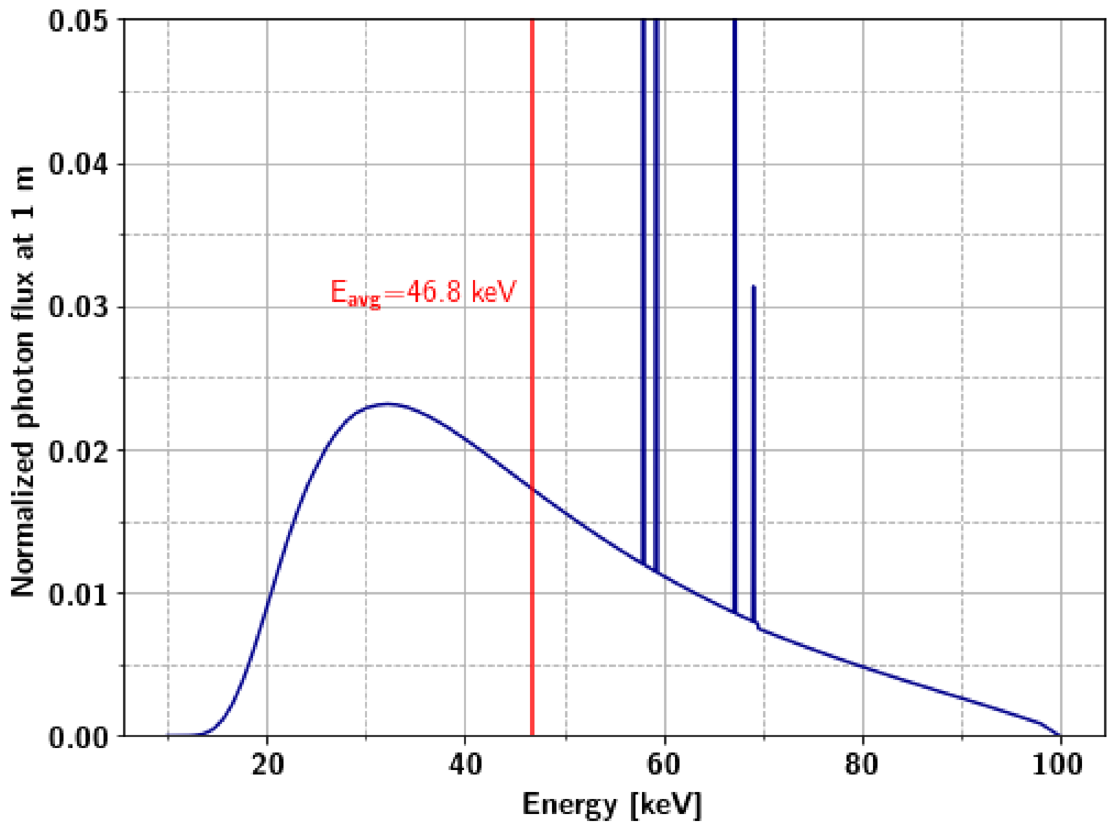
[37]:

```python
# switch energies to the center bin energy
df_spek['CenterBinEnergy_keV'] = df_spek['Energy_keV'] + 0.05

# compute E times intensity: <E> = int_0^inf(E * phi * dE)
df_spek['ExI'] = np.multiply(df_spek['CenterBinEnergy_keV'],␣
 ↪df_spek['phi_norm'])
# average photon energy
E_avg = simps(df_spek['ExI'], df_spek['CenterBinEnergy_keV'] )
```

[38]:

```python
fs = 14
plt.figure(figsize=(8, 6))
plt.plot(df_spek['Energy_keV'], df_spek['phi_norm'], color='darkblue')

plt.plot([E_avg, E_avg], [0, 0.05], color='red')
ax = plt.gca()
ax.text(26.5,0.03, r'E$_{avg}$' + r'={:.1f} keV'.format(E_avg), color='red',␣
 ↪fontsize=fs)

plt.ylabel(r'\textbf{Normalized photon flux at 1 m}', fontsize=fs)
plt.xlabel(r'\textbf{Energy [keV]}', fontsize=fs)
plt.ylim(0, 0.05)

ax.tick_params(axis = 'both', which = 'major', labelsize = fs)
```

```
# minor ticks x
minor_locator = AutoMinorLocator(2)
ax.xaxis.set_minor_locator(minor_locator)
# minor ticks y
minor_locator = AutoMinorLocator(2)
ax.yaxis.set_minor_locator(minor_locator)
# grid
ax.grid(b=True, which='major', linestyle='-')#, color='gray')
ax.grid(b=True, which='minor', linestyle='--')#, color='gray')
plt.show()
plt.close()

df_spek[['Energy_keV', 'phi_norm']].to_csv('03_spectrum.csv', header=['Energy␣
↪[keV]', 'Photon flux at 1 m [1/cm2/keV/s] (normalized)'])
```



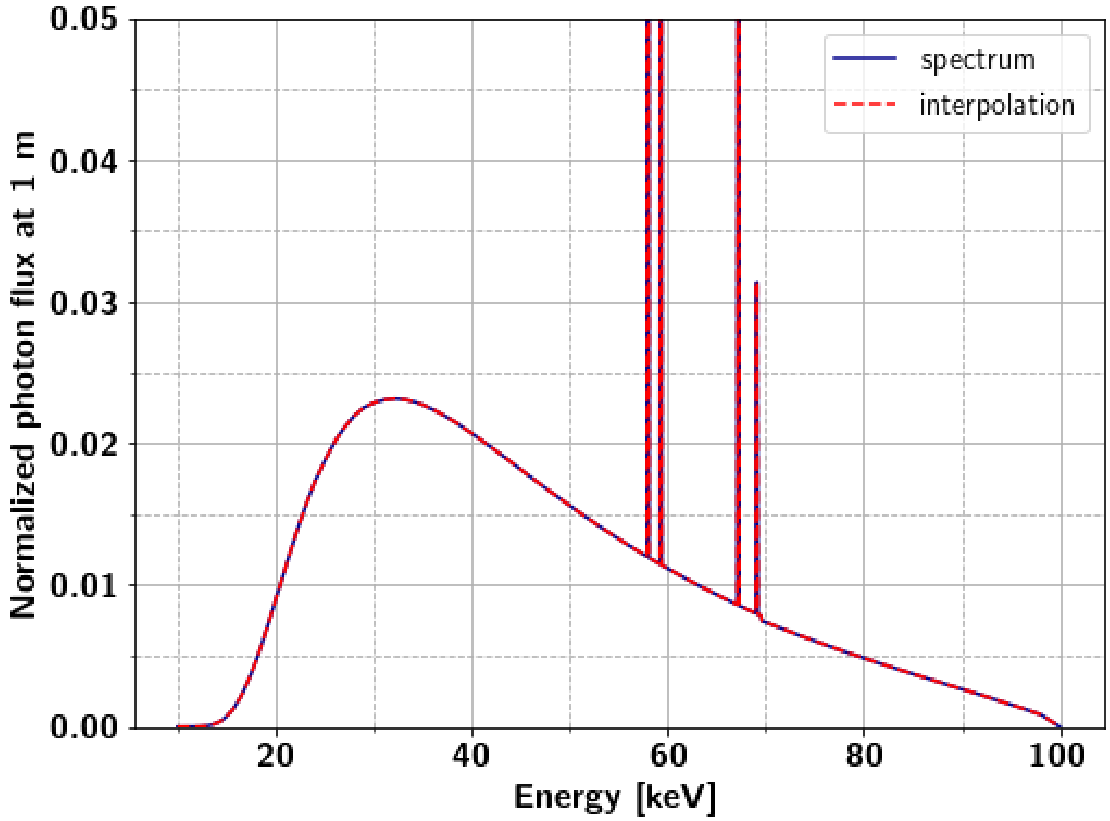
Create an interpolation function using $Y = \phi_{norm}(E)$ and $X = E$ for energies between minimum and maximum photon energies (E_{min} and E_{max}).

[39]:
```
# interpolate the flux
interpPhi_norm = interpolate.interp1d(df_spek['Energy_keV'].values,␣
↪df_spek['phi_norm'].values, fill_value='extrapolate')
```

```python
# minimum and maximum spectrum photon energy
E_max = np.max(df_spek['Energy_keV'])
E_min = np.min(df_spek['Energy_keV'])
```

[42]:

```python
fs = 16
E_vec = np.arange(E_min, E_max+0.1, 0.1)
plt.figure(figsize=(8, 6))
plt.plot(df_spek['Energy_keV'], df_spek['phi_norm'], color='darkblue',␣
 ↪label='spectrum')
plt.plot(E_vec, interpPhi_norm(E_vec), color='red', linestyle='--',␣
 ↪label='interpolation')


plt.ylabel(r'\textbf{Normalized photon flux at 1 m}', fontsize=fs)
plt.xlabel(r'\textbf{Energy [keV]}', fontsize=fs)
plt.ylim(0, 0.05)
plt.legend(fontsize=fs-2)
ax = plt.gca()
ax.tick_params(axis = 'both', which = 'major', labelsize = fs)
# minor ticks x
minor_locator = AutoMinorLocator(2)
ax.xaxis.set_minor_locator(minor_locator)
# minor ticks y
minor_locator = AutoMinorLocator(2)
ax.yaxis.set_minor_locator(minor_locator)
# grid
ax.grid(b=True, which='major', linestyle='-')#, color='gray')
ax.grid(b=True, which='minor', linestyle='--')#, color='gray')
plt.show()
plt.close()
```



Import the X-Ray Mass Attenuation Coefficients for iron and aluminum from https://physics.nist.gov/PhysRefData/XrayMassCoef/tab3.html (As an alternative, one can use https://physics.nist.gov/PhysRefData/Xcom/html/xcom1.html)

Import the X-Ray Mass Attenuation Coefficients for water from https://physics.nist.gov/PhysRefData/XrayMassCoef/tab4.html

Using the densities for the respective material (given in the exercise or use the values in the NIST tables https://physics.nist.gov/PhysRefData/XrayMassCoef/tab1.html and https://physics.nist.gov/PhysRefData/XrayMassCoef/tab2.html) to compute the attenuation coefficient for each of the energies in the table:

$$\mu = \frac{\mu}{\rho} * \rho$$

Convert the energies from MeV into keV.

Interpolate the attenuation coefficient and resample in smaller energy steps (i.e. 0.1 keV). Should be the same as the SpekCalc-data. If the alternative attenuation coefficient website is used, this is not necessary as the energies requested can be put in manually.

Results for the attenuation coefficients are in "04_attenuation_coefficient_XXX.csv", where XXX is the material.

The spectrum averaged attenuation coefficient is:

$$\mu_{avg} = \int_0^\infty \mu(E) * \phi_{norm}(E) dE$$

Numerical integration (Simpson's rule or trapezoidal rule) yields: **47.39 1/cm for steel, 2.35 1/cm for aluminum and 0.32 1/cm for water.**

Note, a more general formula for the spectrum averaged attenuation coefficient is:

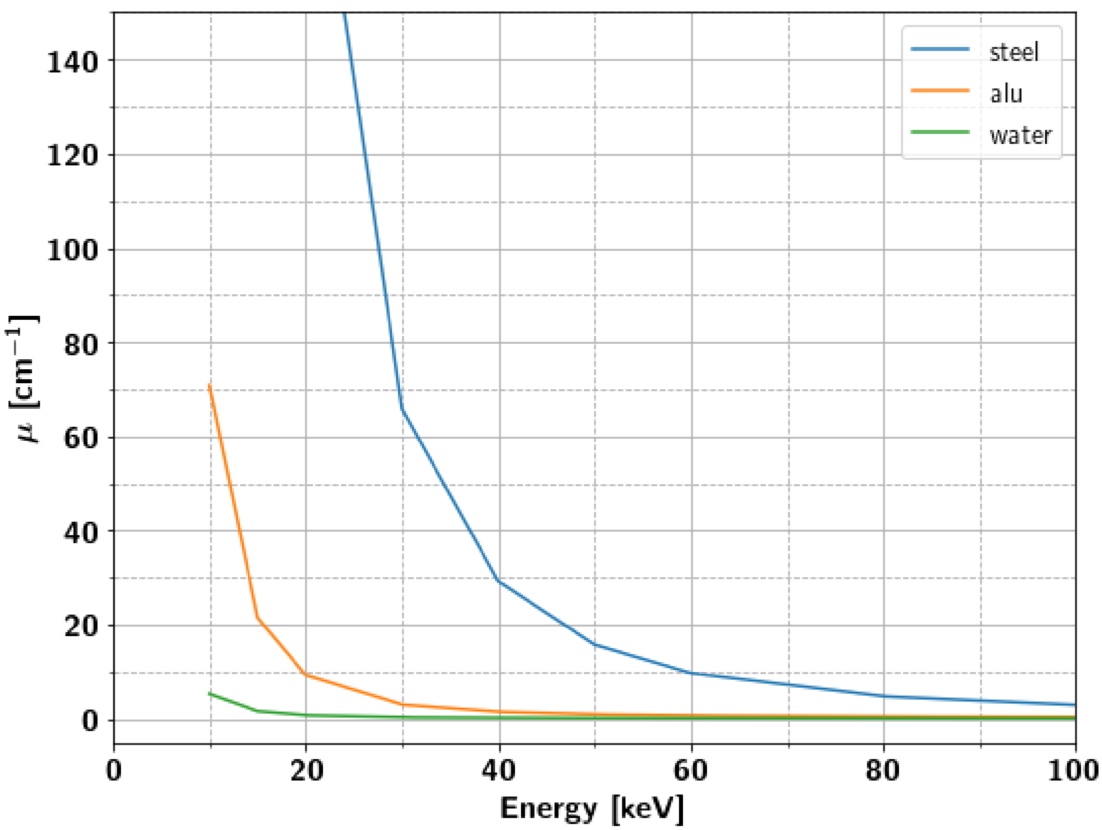
$$\mu_{avg} = \frac{\int_0^\infty \mu(E) * \phi(E) dE}{\int_0^\infty \phi(E) dE}$$

[47]:
```
# -------------------------------------------------------------------
# NIST
# -------------------------------------------------------------------
# energy in MeV, mass att coeff in cm2/g, some other value
material = ['steel', 'alu', 'water']

# densities for steel, alu, water
rhos = [8.05, 2.70, 0.998] # g/cm3

lst_df_NIST = []
ii = 0
for mat in material:
        # read file
        df_t = pd.read_csv('NIST_' + mat, delimiter='\s+', header = None)
        # rename columns
        df_t.columns = ['energy_MeV', 'mass_att_coeff', 'x']
        # convert to keV
        df_t['energy_keV'] = np.multiply(df_t['energy_MeV'], 1e3)
        # calculate attenuation coefficient 1/cm
        df_t['mu_1_p_cm'] = np.multiply(df_t['mass_att_coeff'], rhos[ii])

        # interpolate the attenuation coefficient
        interpNu = interpolate.interp1d(df_t['energy_keV'].values,␣
 ↪df_t['mu_1_p_cm'].values, fill_value='extrapolate')
        # energies for the attenuation coefficient
        energies = np.arange(E_min, E_max+0.1, 0.1)

        # output DF
        df_out = pd.DataFrame(columns=['energy_keV', 'nu_1_p_cm', 'phi_norm'])
        df_out['energy_keV'] = energies
        df_out['mu_1_p_cm'] = interpNu(energies)
        df_out['phi_norm'] = interpPhi_norm(energies)
        df_out['mu_x_phi_norm'] = np.multiply(df_out['mu_1_p_cm'],␣
 ↪df_out['phi_norm'])
```

```
        mu_avg = simps(df_out['mu_x_phi_norm'], df_out['energy_keV'])

        print(r'mu_avg for {} is {:.2f} 1/cm'.format(mat, mu_avg))

        ii = ii + 1


        lst_df_NIST.append(df_out)


# -------------------------------------------------------------------
# plot mu
# -------------------------------------------------------------------
plt.rc('text', usetex=True)
plt.rc('font', weight='bold')
matplotlib.rcParams['mathtext.fontset'] = 'custom'
matplotlib.rcParams['mathtext.rm'] = 'Arial'
matplotlib.rcParams['mathtext.it'] = 'Arial:italic'
matplotlib.rcParams['mathtext.bf'] = 'Arial:bold'
matplotlib.rcParams['mathtext.tt'] = 'Arial'
matplotlib.rcParams['mathtext.cal'] = 'Arial'
matplotlib.rcParams['text.latex.preamble'] = [r'\usepackage{sfmath} \boldmath']

fig = plt.figure(figsize=(8,6))

####################
# axis 1
####################
ax1 = fig.add_subplot(1, 1, 1)

# plot
ii = 0
for this_df in lst_df_NIST:
        ax1.plot(this_df['energy_keV'], this_df['mu_1_p_cm'],␣
 ↪label=material[ii])
        ii = ii + 1
# ax1.plot([E_avg,E_avg], [0,0.1], color='red')
# ax1.text(24.5,0.0005, r'E$_{avg}$' + r'={:.1f} keV'.format(E_avg),␣
 ↪color='red')
ax1.tick_params('x', colors='black', labelsize=fs)
ax1.tick_params('y', colors='black', labelsize=fs)
# grid
ax1.set_xlabel(r'\textbf{Energy [keV]}', fontsize=fs)
ax1.set_ylabel(r'\textbf{$\mu$ [cm$^{-1}$]}', fontsize=fs)

ylims = ax1.get_ylim()
ax1.set_ylim(-5, 150)
```

```python
ax1.set_xlim(0, 100)

# minor ticks x
minor_locator = AutoMinorLocator(2)
ax1.xaxis.set_minor_locator(minor_locator)
# minor ticks y
minor_locator = AutoMinorLocator(2)
ax1.yaxis.set_minor_locator(minor_locator)
# grid
ax1.grid(b=True, which='major', linestyle='-')#, color='gray')
ax1.grid(b=True, which='minor', linestyle='--')#, color='gray')
# fig.subplots_adjust(left=0.15, right=0.97, top=0.88, bottom=0.18)
plt.legend(loc='best', fontsize=fs-2)
plt.tight_layout()
plt.savefig('./mu_vs_energy.png', dpi=600)

plt.show()
```

```
mu_avg for steel is 47.40 1/cm
mu_avg for alu is 2.35 1/cm
mu_avg for water is 0.32 1/cm
```



```
[ ]:
```