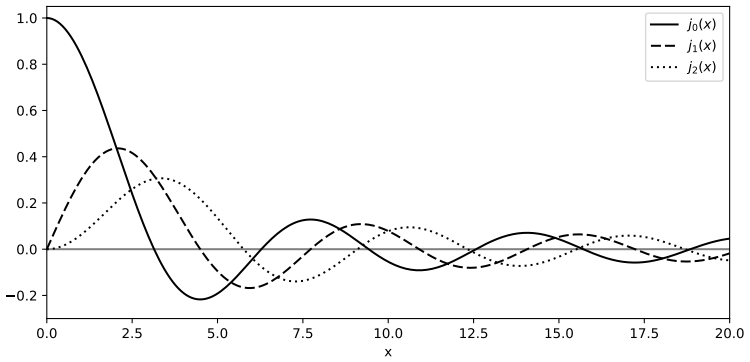
7.6 Exercises

1. Write a function that can return each of the first three spherical Bessel functions $j_n(x)$:

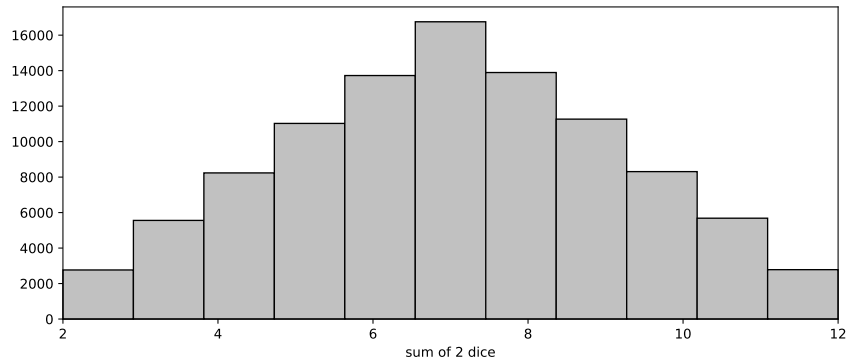
$$
\begin{aligned}
j_0(x) &= \frac{\sin x}{x} \\
j_1(x) &= \frac{\sin x}{x^2} - \frac{\cos x}{x} \\
j_2(x) &= \left(\frac{3}{x^2} - 1\right)\frac{\sin x}{a} - \frac{3\cos x}{x^2}
\end{aligned}
\tag{7.17}
$$

   Your function should take as arguments a NumPy array x and the order n, and should return an array of the designated order n spherical Bessel function. Take care to make sure that your functions behave properly at $x = 0$.

   Demonstrate the use of your function by writing a Python routine that plots the three Bessel functions for $0 \leq x \leq 20$. Your plot should look like the one below. Something to think about: You might note that $j_1(x)$ can be written in terms of $j_0(x)$, and that $j_2(x)$ can be written in terms of $j_1(x)$ and $j_0(x)$. Can you take advantage of this to write a more efficient function for the calculations of $j_1(x)$ and $j_2(x)$?



2. (a) Write a function that simulates the rolling of n dice. Use the NumPy function `random.random_integers(6)`, which generates a random integer between 1 and 6 with equal probability (like rolling fair dice). The input of your function should be the number of dice thrown each roll and the output should be



the sum of the n dice. See §9.2 for a description of NumPy's module random, which has a number of useful functions for generating arrays of random numbers.

(b) "Roll" 2 dice 10,000 times keeping track of all the sums of each set of rolls in a list. Then use your program to generate a histogram summarizing the rolls of two dice 10,000 times. The result should look like the histogram plotted above. Use the matplotlib function `hist` (see http://matplotlib.org/api/pyplot_summary.html) and set the number of bins in the histogram equal to the number of different possible outcomes of a roll of your dice. For example, the sum of two dice can be anything between 2 and 12, which corresponds to 11 possible outcomes. You should get a histogram that looks like the one above.

(c) Repeat part (b) using 3 dice and plot the resulting histogram.

3. In §7.5, we showed that the best fit of a line $y = a + bx$ to a set of data $\{(x_i, y_i)\}$ is obtained for the values of a and b given by Eq. (7.10). Those formulas were obtained by finding the values of a and b that minimized the sum in Eq. (7.5). This approach and these formulas are valid when the uncertainties in the data are the same for all data points. The Python function `lineFit(x, y)` in §7.5.1 implements Eq. (7.10).

(a) Write a new fitting function `lineFitWt(x, y)` that implements the formulas given in Eq. (7.14)) that minimize the χ^2 function give by Eq. (7.12). This more general approach is valid when the individual data points have different weight-

ings *or* when they all have the same weighting. You should also write a function to calculate the reduced chi-squared χ_r^2 defined by Eq. (7.12).

(b) Write a Python program that reads in the data below, plots it, and fits it using the two fitting functions `lineFit(x, y)` and `lineFitWt(x, y)`. Your program should plot the data with error bars and with *both* fits with and without weighting, that is from `lineFit(x, y)` and `lineFitWt(x, y, dy)`. It should also report the results for both fits on the plot, similar to Fig. 7.3, as well as the values of χ_r^2, the reduce chi-squared value, for both fits. Explain why weighting the data gives a steeper or less steep slope than the fit without weighting.

```
Velocity  vs time datafor a falling  mass
time (s) velocity (m/s) uncertainty (m/s)
 2.23         139              16
 4.78         123              16
 7.21         115               4
 9.37          96               9
11.64          62              17
14.23          54              17
16.55          10              12
18.70          -3              15
21.05         -13              18
23.21         -55              10
```

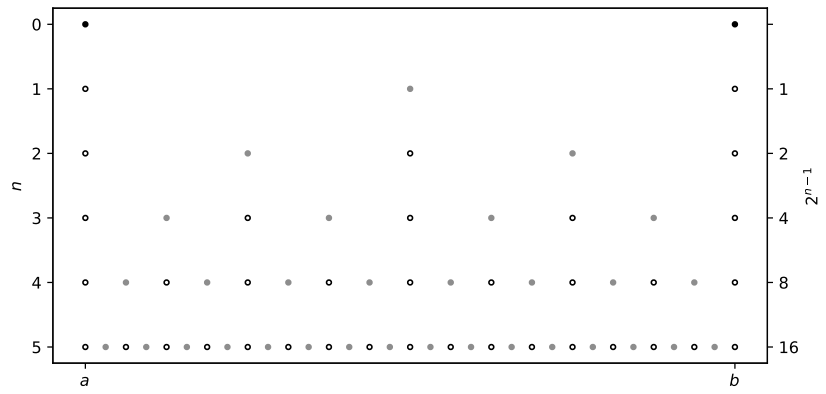
4. Modify the function `lineFitWt(x, y)` that you wrote in Exercise 4 above so that in addition to returning the fitting parameters a and b, it also returns the uncertainties in the fitting parameters σ_a and σ_b using the formulas given by Eq. (7.16). Use your new fitting function to find the uncertainties in the fitted slope and y-intercept for the data provided with Exercise 4.

5. Write a function to that numerically estimates the integral

$$A = \int_a^b f(x)\,dx$$

using the *trapezoid rule*. The simplest version of the trapezoid rule, which generally gives a very crude estimate, is

$$A_0 = \tfrac{1}{2}h_0[f(a) + f(b)]\,, \quad h_0 = b - a\,.$$



This estimate for the integral can be refined by dividing the interval from a to b in two and performing the trapezoid rule on each interval. This process can be repeated as many times as needed until you get the desired precision, which you can estimate by requiring that the fractional difference between successive estimates $(A_i - A_{i-1})/A_i < \epsilon$, where ϵ might be some small number like 10^{-8}. Repeatedly applying the trapezoid rule gives the following succession of estimates

$$
\begin{aligned}
A_1 &= \tfrac{1}{2}h_1[f(a)+f(a+h_1)] + \tfrac{1}{2}h_1[f(a+h_1)+f(b)]\,, \quad h_1 = \tfrac{1}{2}h_0 \\
&= \tfrac{1}{2}h_1[f(a)+2f(a+h_1)+f(b)] \\
&= \tfrac{1}{2}A_0 + h_1 f(a+h_1) \\
A_2 &= \tfrac{1}{2}A_1 + h_2[f(a+h_2)+f(b-h_2)]\,, \quad h_2 = \tfrac{1}{2}h_1 \\
A_3 &= \tfrac{1}{2}A_2 + h_3[f(a+h_3)+f(a+3h_3)+f(a+5h_5)+f(b-h_3)]\,, \\
&\qquad h_3 = \tfrac{1}{2}h_2 \\
&\vdots \\
A_n &= \tfrac{1}{2}A_{n-1} + h_n \sum_{i=1,3,\ldots}^{2^{n-1}} f(a+ih_n)\,, \quad h_n = \tfrac{1}{2}h_{n-1}\,, \text{ for } n \geq 1
\end{aligned}
$$

Write a function that implements the trapezoid rule by first evaluating A_0, then A_1, ... until ϵ is less than some preset tolerance. Note that to calculate A_i, by using the previous result A_{i-1}, you need only to evaluate the function to be integrated $f(x)$ at the open

circles in the preceding diagram, saving a great deal of computation.

Try your trapezoid integration function on the following integrals and show that you get an answer within the specified tolerance of the exact value.

(a) $\int_2^5 x^2\,dx = 39$

(b) $\int_0^\pi \sin x\,dx = 2$

(c) $\int_0^{3.5} e^{-x^2}\,dx = \frac{\sqrt{\pi}}{2}\text{erf}(3.5) \simeq 0.8862262668989721$