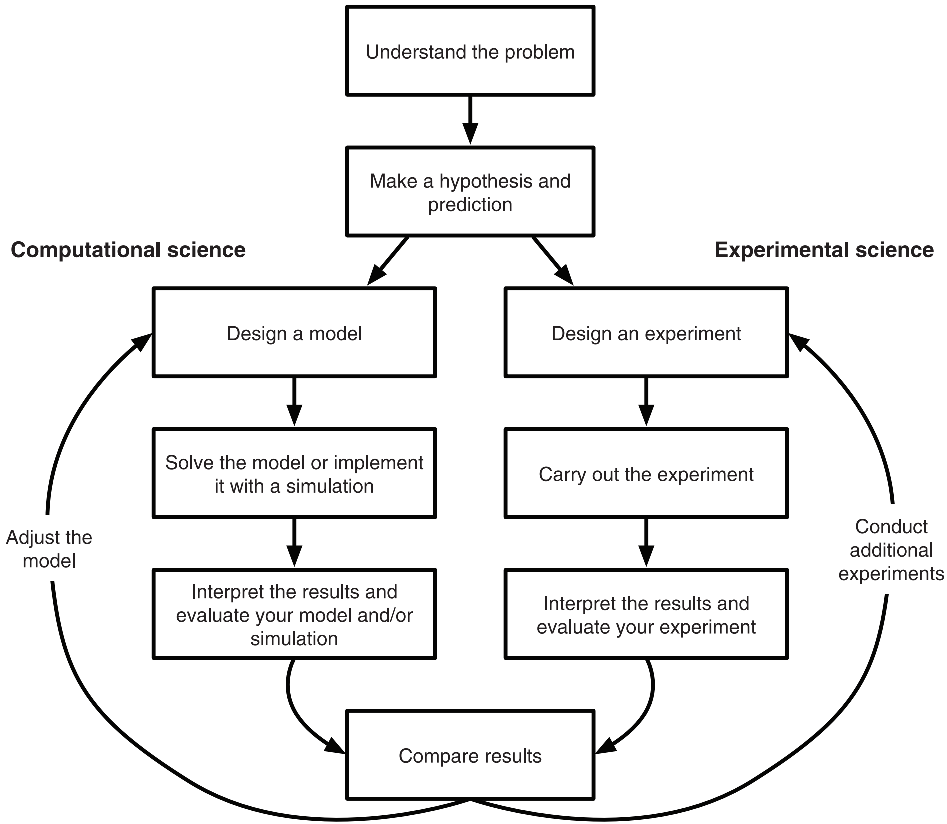

Figure 4.1 Parallel experimental and computational processes.

004x001.eps

Courtesy of CRC Press/Taylor & Francis Group

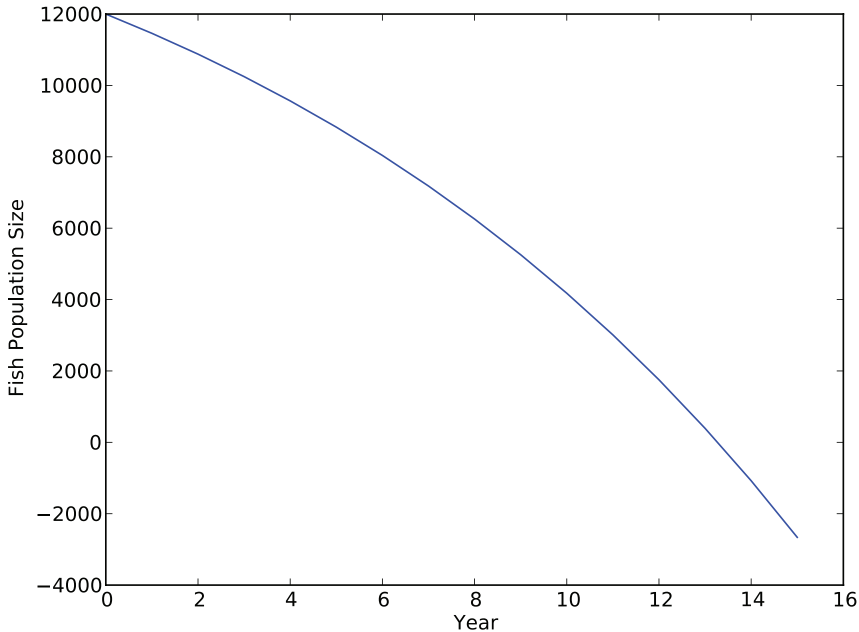

Figure 4.2 Plot of population size in our fishing pond model with years = 15.

004x002.eps

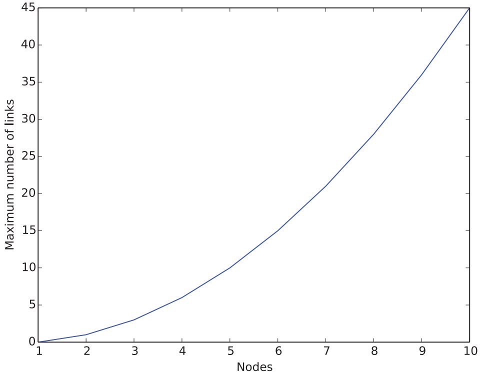

Figure 4.3 Plot for Exercise 4.2.1.

004x003.eps

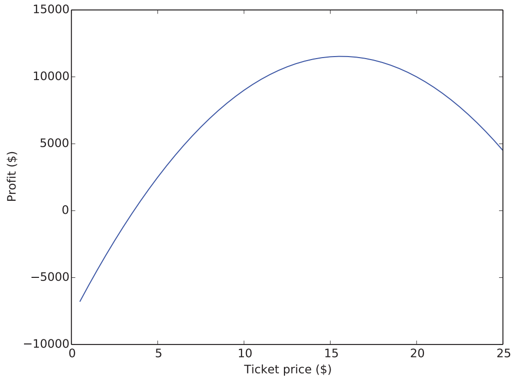

Figure 4.4 Plot for Exercise 4.2.2.

004x004.eps

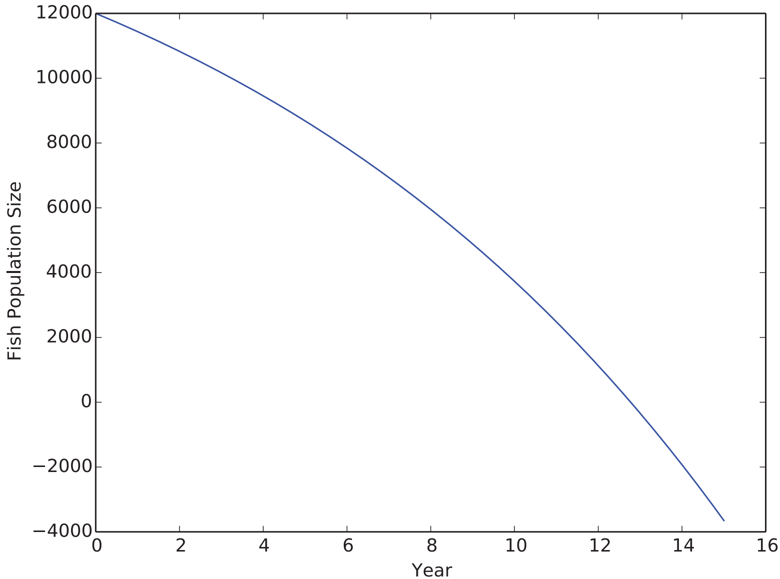

Figure 4.5 The plot produced by calling pond (15, 12000, 1500, 0.01).

004x005.eps

Courtesy of CRC Press/Taylor & Francis Group

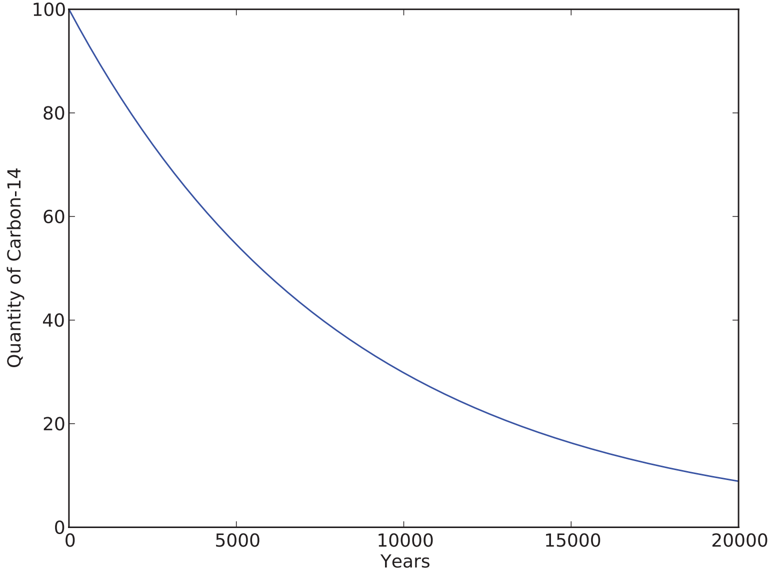

Figure 4.6 Plot of carbon-14 decay generated with decayC14(100, 20000, 0.01).

004x006.eps

```
import matplotlib.pyplot as pyplot

def SIR(population, days, dt):
    """Simulates the SIR model of infectious disease and
       plots the population sizes over time.

    Parameters:
        population: the population size
        days: number of days to simulate
        dt: the value of "Delta t" in the simulation
            (fraction of a day)

    Return value: None
    """

    susceptible = population - 1  # susceptible count = S(t)
    infected = 1.0                # infected count = I(t)
    recovered = 0.0               # recovered count = R(t)
    recRate = 0.25                # recovery rate r
    infRate = 0.0004              # infection rate d
    SList = [susceptible]
    IList = [infected]
    RList = [recovered]
    timeList = [0]

    # Loop using the difference equations goes here.

    pyplot.plot(timeList, SList, label = 'Susceptible')
    pyplot.plot(timeList, IList, label = 'Infected')
    pyplot.plot(timeList, RList, label = 'Recovered')
    pyplot.legend(loc = 'center right')
    pyplot.xlabel('Days')
    pyplot.ylabel('Individuals')
    pyplot.show()
```

Figure 4.7 Output from the SIR model with 2,200 individuals over 30 days with $\Delta t = 0.01$.

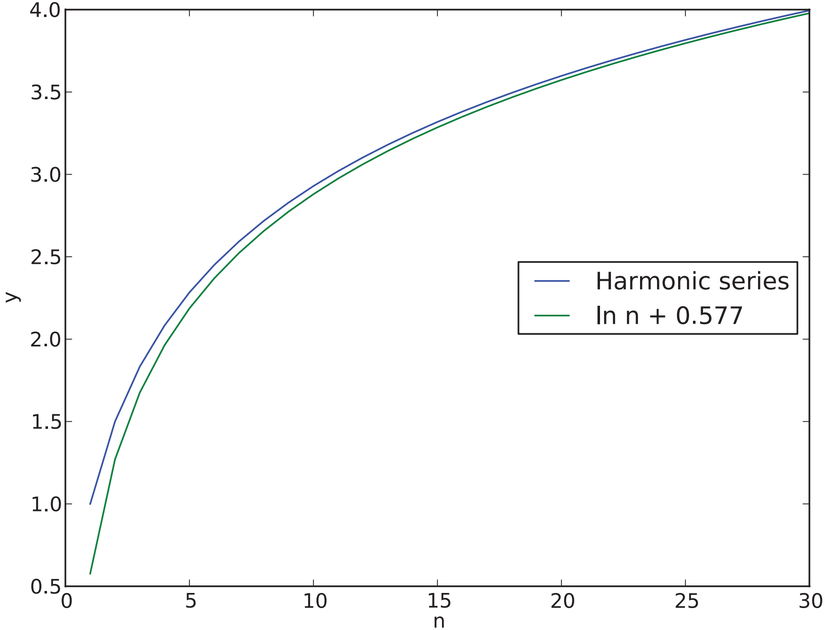

Figure 4.8 Harmonic series approximation of the natural logarithm (ln).

004x008.eps

Courtesy of CRC Press/Taylor & Francis Group

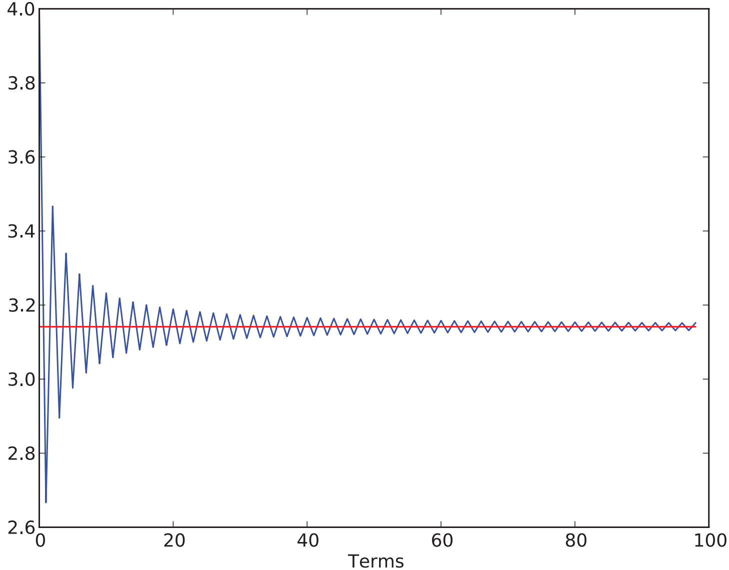

Figure 4.9 The Leibniz series converging to π.

004x009.eps

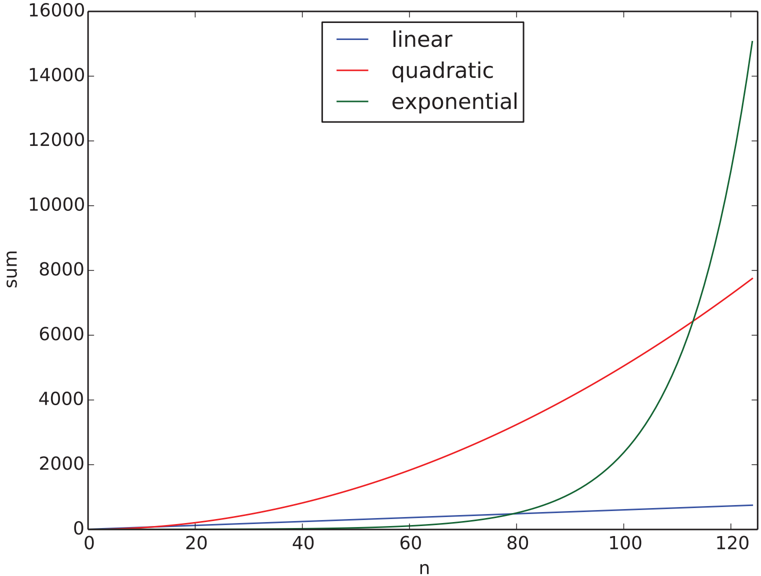

Figure 4.10 An illustration of linear, quadratic, and exponential growth. The curves are generated by accumulators adding 6, the index variable, and 1.08 times the accumulator, respectively, in each iteration.

004x010.eps