CHAPTER 7 - ROOTS AND MINIMIZATION

There are many situations in computational physics in which one wishes to drive a system towards a specific characteristic. Two mathematical aspects to these situations are

•*root-finding*. Example: determining the pressure needed to give a gas sample a particular average volume.

•*minimization*. Example: determining the lowest energy state (or *ground state*) of a system.

Both of these computational tasks have application in a broad spectrum of fields, so there is an extensive mathematical literature on algorithms that perform the tasks efficiently. Our purpose in this section is to introduce the reader to the issues, and present some concepts behind the algorithms. For more details on the algorithms, their mathematics and their accuracy, a good starting point is Press *et al*. (1992).

7.1 Root-finding

Consider the problem of finding the pressure P required to contain a gas sample to a particular average volume $\langle V \rangle$, as in projects 4 and 5. While it may be more conventional notation to use V instead of $\langle V \rangle$, we know from our earlier work that only one of P and V can be fixed. The volume is a function of several variables, but here we consider it only to be a function of pressure $f(P)$:

$$\langle V \rangle = f(P). \tag{7.1}$$

The requirement that the average volume be equal to a fixed value V_f can be stated as

$$f(P) = V_f$$

or

$$f(P) - V_f = 0. \tag{7.2}$$

Determining V_f is equivalent to finding the roots of Eq. (7.2).

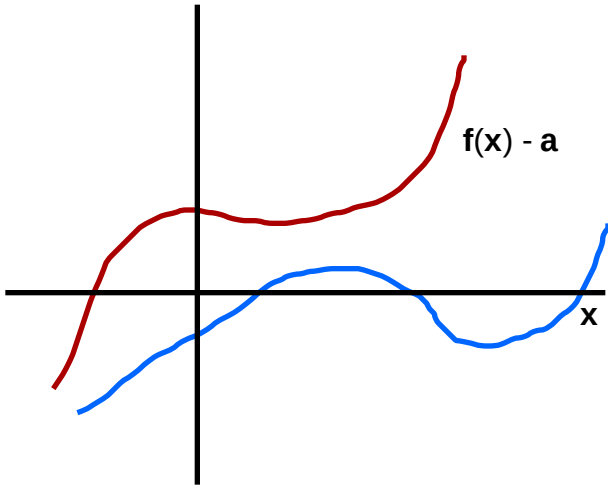


Fig. 7.1 Two sample functions with local extrema. The lower (blue) curve also has more than one root.

As long as one avoids phase changes, the pressure-volume relationship for gases is a continuous monotonic function, and there is only one root to Eq. (7.2). However, functions may not be monotonic, or may have more than one root, as in Fig. 7.1. Numerical methods for finding roots must take this kind of behavior into account. Most methods are iterative, in that they begin the search for a root with an educated guess for the region in which the root lies and then iteratively approach the root. We discuss two such methods - *bisection* and *Newton's rule.*

Bisection

The bisection method, although not the fastest computationally, will find at least one root or singularity of an equation. For each root in Fig. 7.1, there exists a domain in which the function $F(x) = f(x) - a$ changes sign around the root, and this statement is true in general unless the function has singularities. Let x_1 and x_2 be two values of x for which the corresponding values of the function $F(x)$ have different signs. Hence x_1 and x_2 *bracket* at least one root. If the region between x_1 and x_2 is bisected, then the root must lie on one side of the bisector. Defining x_b as the value of x that bisects the region, then the root must lie in (x_1, x_b) or (x_b, x_2). The algorithm obtains the sign of $F(x_b)$, then chooses one of these regions to bracket the root. This procedure is iterated until the bracketing region is smaller than a user-determined numerical range. At that point, the root is said to be found. The bisection method is obviously easy to code, and ultimately yields a root. There are several variations on this geometrical method of finding roots.

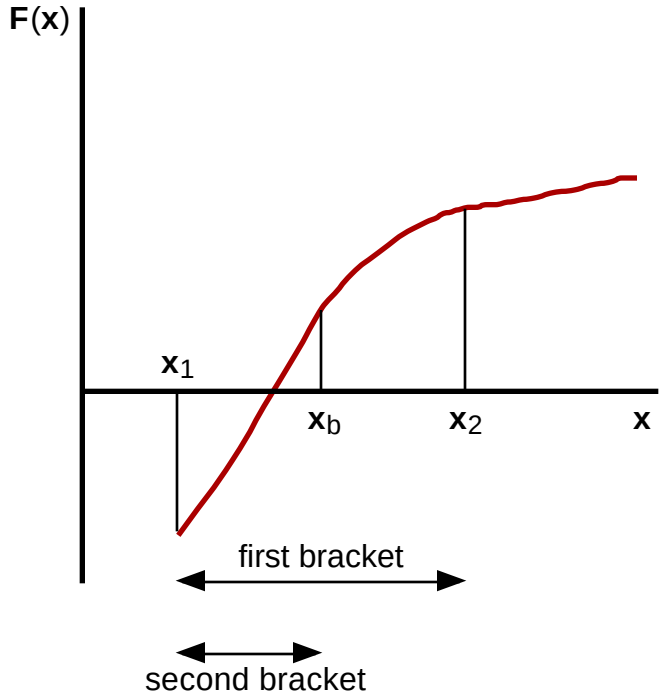


Fig. 7.2. An iterative procedure is used in the *bisection* method to provide a nested sequence of intervals to bracket a root.

Newton's rule

The bisection method gradually encloses the root through a sequence of intervals. An alternate approach that requires the evaluation of a derivative, may lead one from a starting point to a root more quickly, although its success is not guaranteed. This iterative procedure is called *Newton's rule*, or *Newton-Raphson* method, and assumes that the starting value of x is moderately close to the root. The method has its origins in a Taylor series expansion of the function whose root one is attempting to find.

Suppose that the starting value of the search is at x, and the root is at $x + \delta$. Then, by Taylor series

$$F(x + \delta) = F(x) + F'(x)\delta + (1/2)F''(x)\delta^2 + \ldots \qquad (7.3)$$

where the primes indicate derivatives with respect to x. If δ is small, then terms of order δ^2 and higher may be neglected. The condition that the root is at $x + \delta$ is equivalent to

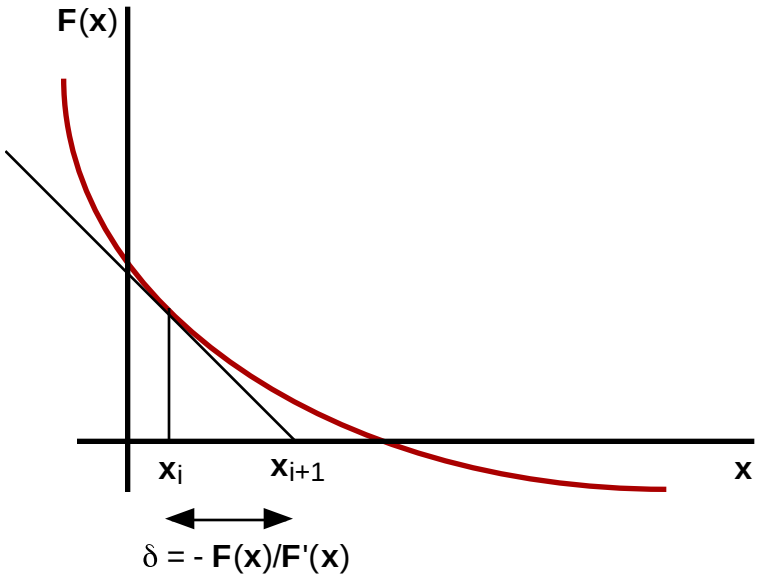


Fig. 7.3. Newton's rule for iteratively approaching a root using derivatives of the function $F(x)$. The procedure is repeated until succesive values of x_i change by less than a user-specified amount.

$$F(x + \delta) = 0,$$

for which Eq. (7.3) yields

$$\delta = - F(x) / F'(x) \qquad (7.4)$$

In other words, if one knows the first derivative of the function, either analytically or numerically, then Eq. (7.4) provides an approximation for determining the value of x_{i+1} (at iteration i+1) from x_i:

$$x_{i+1} = x_i - F(x) / F'(x) \qquad (7.5)$$

Eq. (7.5) is obviously exact if $F(x)$ is linear in x, but then one usually doesn't resort to a computer to solve a linear equation with one unknown. The procedure is illustrated in Fig. 7.3.

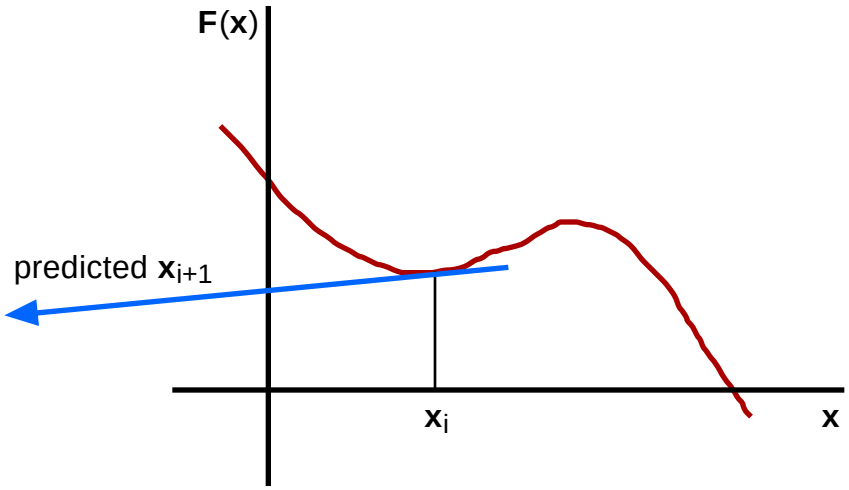


Fig. 7.4. Application of Newton's rule (for root-finding) near a local extremum can fail to converge. The predicted x_{i+1} in this example of root-finding is not the same as that obtained from the Newton-Raphson method for minimization in Sec. 7.2.

While Newton's rule is faster than bisection, there are pitfalls in using it. Divisions are dangerous, especially divisions by derivatives. If there are local minima or maxima to the function, as in Fig. 7.4, then Newton's rule will produce extraordinarily large and usually incorrect updates if $F'(x)$ is near zero. These difficulties, and the application of Newton's rule and other procedures to functions of more than one variable, are discussed at length by Press *et al.* (1992).

7.2 Minimization

A related, but not identical, problem to root-finding is the problem of finding extrema of functions. For example, the ground state of a system at zero temperature corresponds to a minimum of the energy. Another example of minimization is a system of equations that is overconstrained and has no solution. In such a situation, an acceptable procedure is to find values for the variables that minimize a statistical measure such as χ^2 (see Chap. 8). Again, many algorithms have been developed for minimization. In this section, we cover a traditional approach - *Newton-Raphson* - as well as a more recent approach - *simulated annealing*. Descriptions of an alternate minimization routine that has biological roots - genetic algorithms - can be found in the references at the end of Appendix A.

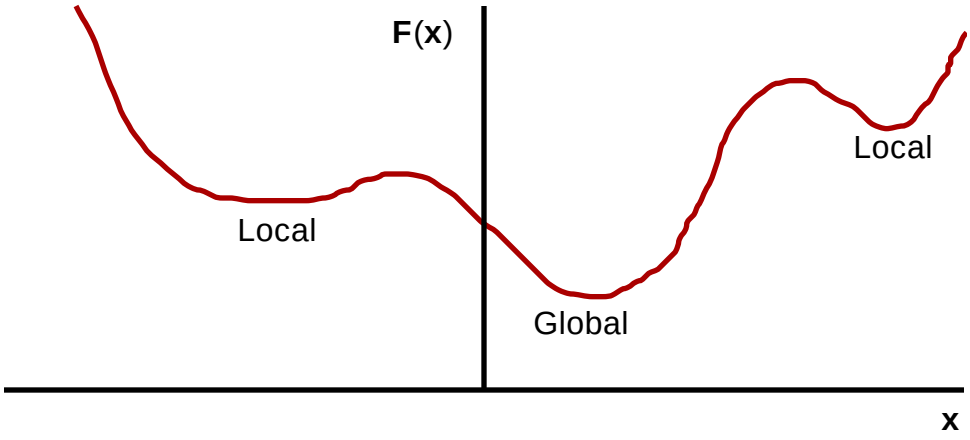


Fig. 7.5. Minimization of a function with several local minima, but only one global minimum.

Newton-Raphson method

The *Newton-Raphson* iteration algorithm starts with a Taylor series expansion for the *derivative* of the function $F(x)$:

$$F'(x+\delta) = F'(x) + F''(x)\delta + \ldots \qquad (7.6)$$

Imposing the condition that $F'(x) = 0$ at the extremum leads to a similar form to the Newton-Raphson method for finding $F(x) = 0$:

$$x_{i+1} = x_i - F'(x) / F''(x) \qquad (7.7)$$

Note the similarities and differences between the Newton-Raphson iteration rules for root-finding and minimization:

	Condition	Iteration
Root-finding	$F(x) = 0$	$x_{i+1} = x_i - F(x) / F'(x)$
Minimization	$F'(x) = 0$	$x_{i+1} = x_i - F'(x) / F''(x)$

Note further that we have considered only functions of one variable. Multivariable functions require more attention.

Simulated annealing

One of the challenges of finding global minima using the Newton-Raphson method is the problem of being trapped in a local minimum, as in Fig. 7.5. The presence of false minima may increase with the number of variables in the problem, since the function becomes embedded in a multi-dimensional space.

One way to avoid being trapped in local minima is to incorporate a mechanism for temporarily reversing the "downhill slide" that we have been so careful to construct. Of course, one does not want to reverse the direction of the moves *too* easily, or one will never find the minimum. The situation resembles the constant temperature ensemble that we investigated with the Monte Carlo algorithm: there are many energy states, and one can move in a direction that *increases* the energy, but only with a probability less than unity.

The idea behind simulated annealing as a means to find minima is to assign a "temperature" to the function and allow the function variables to explore a range of values according to the Metropolis algorithm. For a given parameter range *dx*, the system (*i.e.*, function) will spend more time near the global minimum than near other regions, but it will not be completely restricted in its movement. There is always a finite probability that the system will move from one region to another, even though the probability may be very small.

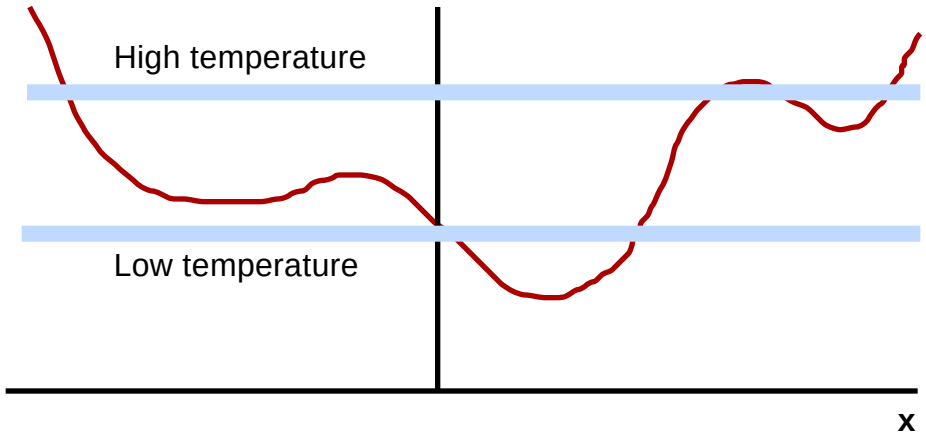


Fig. 7.6 In simulated annealing, the system can explore a large parameter space at high temperature, and a smaller space at low temperature. While the probability of moving from one region to another may be small, it is never zero.

One implementation of simulated annealing is the following:

1. Start the system off at a temperature high enough so that the system explores a reasonable region of parameter space.

2. Allow the system to relax at this temperature, then lower the temperature to a new value, whose starting configuration is the same as the last configuration at the previous temperature.

3. Repeat step 2 as the system is cooled towards zero temperature.

4. Repeat 1-3 as necessary with different initial configurations to check if the same final configuration is reached.

An extension of this approach is to evaluate quantities of interest, such as the expectation of the energy $\langle E \rangle$, at each temperature, and then extrapolate their value to $T = 0$.

References

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd edition (Cambridge, 1992) .

7.3 Project 7 - Angels on a ring

Methods for finding the ground state of a system of mutually repelling particles in a confined geometry have long been of interest in physics and mathematics. For objects with hard-core interactions, the problem is commonly called the Tammes problem, after a Dutch botanist who was interested in the pattern of orifices in spherical pollen grains. The Tammes problem is equivalent to decorating a sphere of fixed size with N identical, non-overlapping disks of radius R: the object is to find the maximum allowable value of R compared to the sphere radius.

In this project, we will consider objects with soft-core repulsive interactions. Allowing the objects to move on the surface of a sphere is equivalent to the historically important problem of finding the number of angels on the head of a pin. However, minimizing the energy for a system with two translational degrees of freedom per angel involves a number of coding issues beyond the minimization routines presented in Sec. 7.1-7.2. We therefore content ourselves with solving the simpler problem of

finding the ground state of angels on a flat ring. The solution will be obtained in two ways:
•Newton-Raphson method to find ground state energies as a function of the number of angels
•simulated annealing for a fixed number of angels.

Physical system

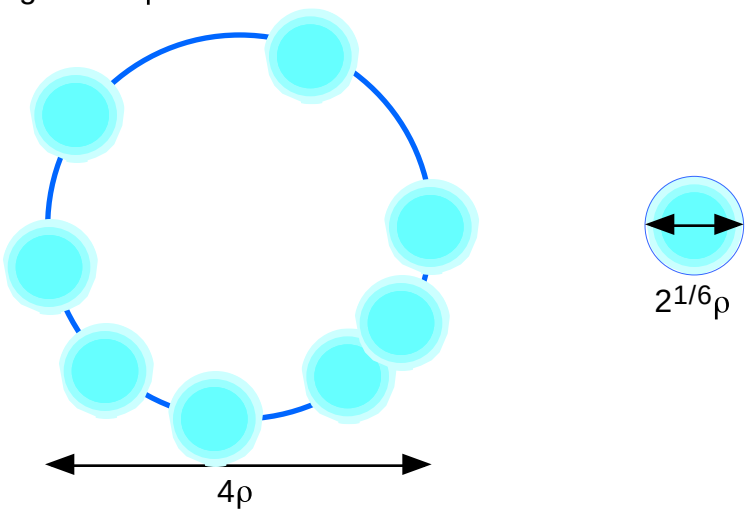
The angels are placed on a planar ring of fixed radius. The angels themselves have a central-force interaction with the other angels, given by the repulsive part of the Lennard-Jones interaction, but can otherwise move freely on the ring.

Simulation parameters

The interangel potential is the repulsive Lennard-Jones interaction:

$$v_{rep}(r) = \begin{cases} 4\varepsilon[(\rho / r)^{12} - (\rho / r)^{6}] + \varepsilon & \text{for } r < 2^{1/6}\rho \\ 0 & \text{for } r \geq 2^{1/6}\rho \end{cases} \qquad (7.8)$$

where the constants ρ and ε fix the length and energy scales respectively. Choose the diameter of the ring to be 4ρ.



Rather than use *xy* coordinates to represent the angel positions, work with angel angles θ, and express the chord r_{ij} in Eq. (7.8) in terms of $\theta_{ij} = \theta_i - \theta_j$, where *i* and *j* are angel labels. While you undoubtedly will use $\cos\theta_{ij}$ in your code, note that random positional moves must involve changes in θ_i to obtain the correct Boltzmann weight.

The ground state energy E_{gs} of this system can be determined analytically as a function of the number of angels N. Evaluate E_{gs} for $2 \le N \le 30$, paying close attention to the range of N for which $E_{gs} = 0$. Show that for N in the 20-30 range,

$$E_{gs} / N \sim 4\varepsilon(N/4\pi)^{12}. \qquad (7.9)$$

Code 1 - Newton's rule

1. Start with a copy of a previous code. For this project, you will need:
   - a random number generator
   - a routine to evaluate first and second derivatives of the potential analytically
   - a routine to make deterministic moves in the angel angle.

2. The energy of the system is just the potential energy given by the contributions from the Lennard-Jones interaction in Eq. (7.8), being careful not to double count. Because the potential involves even powers in the chord r_{ij}, it is very easy to express derivatives of the energy in terms of $\cos\theta_{ij}$.

3. Fix the radius $R = 2\rho$ of the ring, and follow steps 4 - 5 for a sequence of values of $N \le 30$. You don't need to treat small values of N where $E_{gs} = 0$.

4. Randomly assign angels to positions on the ring. Update the angle of each angel sequentially by an amount $\Delta\theta_i$ according to the Newton-Raphson method to minimize the total energy E:

$$\Delta\theta_i = - E'(\theta_i) / E''(\theta_i), \qquad (7.10)$$

where $E'(\theta_i)$ and $E''(\theta_i)$ are the derivatives of E with respect to variation in θ_i, all other θ_j being held fixed. The predicted update given by Newton-Raphson may be too large, and you may have to introduce a damping parameter μ into Eq. (7.10) to reduce the step size:

$$\Delta\theta_i = - \mu E'(\theta_i) / E''(\theta_i). \qquad (7.11)$$

5. In general, it's a good idea to repeat the minimization with several different initializations, just to check that E_{gs} obtained from your code is not just a local minimum in E.

Analysis 1

1. For each N:
   - •follow the energy of the system and declare the local ground state to be reached when the energy changes by less than a fixed amount from one iteration to the next.

2. Combine the data for E_{gs} as a function of N and:
   - •plot E_{gs} / N to see how the average energy per angel increases with N.
   - •find the addition energy $A_N = E_{gs,N} - E_{gs,N-1}$, which measures how much energy it takes to add the N^{th} angel.

3. Are the observables E_{gs} / N and A_N power law functions of N? If so, what are the exponents?

4. Compare your results with your analytic expressions.

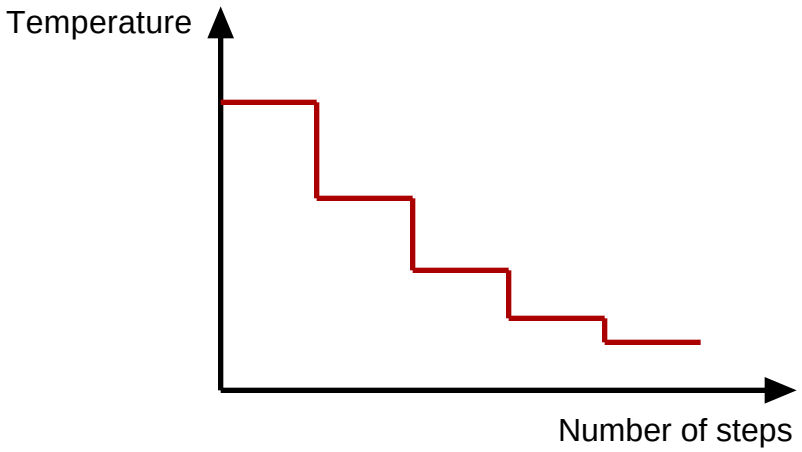
Code 2 - Simulated annealing

1. Start with a copy of code 1, and:
   - •throw out the derivative calculation
   - •add a Monte Carlo routine based upon the angel interaction potential.

2. Fix the radius of the ring at $R = 2\rho$, and add $N = 12$ angels.

3. Randomly assign angels to positions on the ring, then perform trial moves according to the Metropolis Monte Carlo technique. Allow the system to relax, then evaluate the average energy $\langle E \rangle$ for an ensemble of 100 statistically-independent configurations.

4. Perform step 3 for five values of the reduced temperature $(\beta\varepsilon)^{-1}$, where large values of $\beta\varepsilon$ correspond to low temperature. Each time you lower the temperature, use the last configuration at the higher temperature as the starting configuration at the lower temperature. Your "cooling curve" should look like:



Analysis 2

1. Find the energy <*E*> of the annealed state once the cooling process is complete.

2. Using the data at non-zero temperature, extrapolate the value of <*E*> to the zero temperature limit of $(\beta\varepsilon)^{-1} \rightarrow 0$. Compare the resulting energy with your analytical calculation and the result from the Newton-Raphson method of code 1. Estimate the uncertainty of your extrapolation.

Report

Begin the report with an analytical calculation of E_{gs} for $2 \leq N \leq 30$, and obtain the large-*N* expression (7.9) with only nearest neighbor interactions. Then report your results for Newton-Raphson and simulated annealing simulations. For each method, include:
•a one-paragraph conceptual outline of the technique
•a short description of how your code is organized
•results for your ground state search, including your extensive analysis of the results as outlined above.
Compare your data to the analytic calculations, then attach copies of your two codes after the report's conclusion.