Ph 22.0 – Finding Roots

-v20140107-

Introduction

The last numerical technique that you will meet in ph20 is *numerical root finding*, useful whenever you cannot get an analytical expression for the solutions of an equation. In this assignment you make your acquaintance with three simple root-finding techniques (the *bisection*, *Newton–Raphson*, and *secant* methods), and you put them to work to study the motion of two gravitationally bound bodies in an elliptical orbit.

Numerical root finding

To find the roots of a function $f(x)$ of a single real variable numerically, you almost always need to *bracket* them: that is, to find two abscissae x_1 and x_2 such that $f(x_1)$ and $f(x_2)$ have opposite signs. Then, if the function is continuous, the *intermediate value theorem* tells us that there must be a root $f(x_0) = 0$, with x_0 between x_1 and x_2. The bracket may come from analytical insight, or from numerical investigation; *Numerical recipes* describes two simple approaches to bracketing by expanding or shrinking an initial arbitrary interval. Without bracketing, root-finding methods can easily get lost in the vastness of the real line, far from the desired root.

The bisection method

There is little more than repeated bracketing to our first root-finding method, *bisection*.

1. Start with the initial bracket $[x_1^{(1)}, x_2^{(1)}]$ with $f(x_1^{(1)})f(x_2^{(1)}) < 0$ (assume $x_1^{(1)} < x_2^{(1)}$, without loss of generality).
2. Guess the location of the root as the midpoint $x_0^{(1)} = (x_1^{(1)} + x_2^{(1)})/2$.
3. If $f(x_0^{(1)})$ has the same sign as $f(x_1^{(1)})$, set the new bracket to $[x_1^{(2)}, x_2^{(2)}] \equiv [x_0^{(1)}, x_2^{(1)}]$, otherwise set it to $[x_1^{(2)}, x_2^{(2)}] \equiv [x_1^{(1)}, x_0^{(1)}]$.
4. At the end of step 3, the location of the root is known with precision[1] $|x_1^{(2)} - x_2^{(2)}|$. If this is not satisfactory, go back to step 2 to obtain the next bracket, $[x_1^{(3)}, x_2^{(3)}]$. Repeat until satisfied.

For continuous functions, this method is guaranteed to converge to the root, and at each step the error is halved. Therefore the method is said to be *linearly convergent*, because the number of correct digits in the answer (the location of the root) grows linearly with the number of iterations (one decimal digit every $\log_2 10$ iterations).

The Newton–Raphson method

The idea of the *Newton–Raphson method* is to use information about the derivative of the function to guide the choice of a sequence of points converging to the root. Look at Fig. 1 (left) while you go through the algorithm given below.

[1] What about the *accuracy*? That will depend mainly on the accuracy of the numerical computation of $f(x)$.

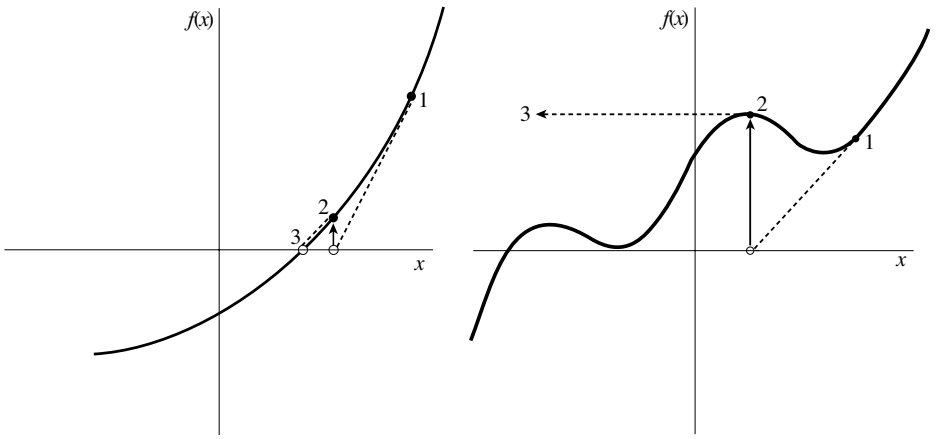

Figure 1: The Newton–Raphson method of numerical root finding. On the left, the method works correctly, converging to the root; on the right, the method is fooled by a local extremum. [Figures adapted from *Numerical Recipes*.]

1. Start from the initial guess $x^{(1)}$.

2. Approximate the function around $x^{(1)}$ by the first two terms of its Taylor expansion (i.e., as the straight line through $[x^{(1)}, f(x^{(1)})]$ with the same slope as the curve $f(x)$):

$$f(x) \approx f(x^{(1)}) + f'(x^{(1)})(x - x^{(1)}) + \cdots . \tag{1}$$

   An approximate guess ($x^{(2)}$) for the root is obtained by setting $f(x) = 0$, yielding

$$x^{(2)} = x^{(1)} - \frac{f(x^{(1)})}{f'(x^{(1)})}. \tag{2}$$

3. Compare the value of $f(x^{(2)})$ with a predetermined *tolerance* (a small number that represents our *working definition* of zero), to decide whether to stop the iteration or to go back to step 2 and obtain $x^{(3)}$. Repeat until satisfied.

4. If the function is sufficiently linear near the last guess, x^{last} (as it should be if the tolerance is sufficiently small with respect to the scale of variation of the function), the precision to which we know the location of the root is approximately $|f(x^{\text{last}})/f'(x^{\text{last}})|$.

The advantage of the Newton–Raphson method is that it *converges quadratically* (the number of correct digits in the answer approximately doubles with each iteration, as proved analytically in *Numerical Recipes*, Sec. 9.4). However, unlike the bisection method, the Newton–Raphson method is not surefire: it can be fooled by local extrema [see Fig. 1 (right)], which can send the next guess for the location of the root way off toward infinity. We could have guessed that this would happen from Eq. (2), where the correction to x_1 becomes very large when the local value of the derivative is very small.

There are two ways to address this problem: one is to use a hybrid method that alternates Newton steps [Eq. (2)] with bisection steps whenever the Newton step would not maintain the

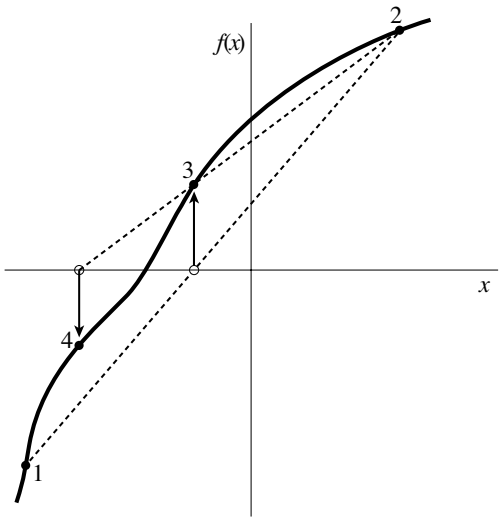

Figure 2: The secant method for numerical root finding. [Figure adapted from *Numerical Recipes*.]

bracket (see for instance the `rtsafe` algorithm in *Numerical recipes*); the other is to know beforehand (by analytical means, or by empirical observation) that the function has no local extrema in the region where we are looking for the root.

The secant method

The *secant method* is a close relative of the Newton–Raphson method, but it does not require the explicit knowledge of the derivative $f'(x)$; instead, the derivative is approximated by tracing a straight line through the last two points examined. The algorithm would run as follows (look at Fig. 2).

1. Start from the *two* initial guesses $x^{(1)}$ and $x^{(2)}$.
2. Approximate the derivative at $x^{(2)}$ by the slope of the line that joins $(x^{(1)}, f(x^{(1)}))$ to $(x^{(2)}, f(x^{(2)}))$:
$$f'(x^{(2)}) \approx \frac{f(x^{(2)}) - f(x^{(1)})}{x^{(2)} - x^{(1)}}. \tag{3}$$
3. Take a Newton–Raphson step to find $x^{(3)}$:
$$x^{(3)} = x^{(2)} - f(x^{(2)}) \frac{x^{(2)} - x^{(1)}}{f(x^{(2)}) - f(x^{(1)})}. \tag{4}$$
4. Compare $f(x^{(3)})$ with the tolerance to decide whether to continue the iteration. Estimate precision as in the Newton–Raphson method.

The secant method converges more slowly than the Newton–Raphson method: its order of convergence is the *golden ratio* $\phi = 1.618\cdots$, so the number of correct digits in the location of the root is approximately multiplied by ϕ after each iteration. The secant method, too, can suffer from fly-to-infinity problems, and requires cautions similar to those outlined for the Newton–Raphson method.

Elliptic Keplerian orbits

Before proceeding, you should review the theory of Keplerian orbits in your favorite general-physics textbook. While for circular orbits it is easy to write down the trajectory of two gravitationally bound bodies as an *explicit* function of time, this is not the case for elliptical orbits, where the trajectory can only be expressed *implicitly* by two *parametric equations* for the orbital separation r and for the time t,

$$r = a(1 - e\cos\xi), \quad t = \frac{T}{2\pi}(\xi - e\sin\xi), \tag{5}$$

where a is the semimajor axis of the orbit, e is the eccentricity, and T is the orbital period, given by Kepler's law as

$$\frac{2\pi}{T} = \sqrt{\frac{G(m_1 + m_2)}{a^3}}, \tag{6}$$

with G Newton's gravitational constant, and m_1 and m_2 the two masses. Last, the parameter ξ is the *eccentric anomaly* of the orbit (i.e., the angle about the center of the ellipse). After some algebra, Cartesian coordinates for the orbital separation are found as

$$x = a(\cos\xi - e), \quad y = a\sqrt{1 - e^2}\sin\xi. \tag{7}$$

To obtain r (or x and y) as a function of t, we must first numerically obtain the value of ξ for a given value of t, and then calculate the value of r with ξ. To find the relation between ξ and t, we pick a specific value of t, say t^*, and then we find (numerically!) the roots of the equation

$$\frac{T}{2\pi}\big(\xi - e\sin\xi\big) - t^* = 0. \tag{8}$$

Similar equations exist for hyperbolic orbits. Having completed this assignment, you will therefore be able to handle all standard two-body orbits: circular, elliptical, and hyperbolic.

The Assignment

1. Prove that the order of convergence of the secant method is the golden ratio.

   Hint: work in analogy to the argument given for the Newton–Raphson method in Sec. 9.4 of *Numerical Recipes*. You should obtain a recurrence relation similar to Eq. (9.4.6), approximating $1/(1 + \text{small})$ as $1 - \text{small}$. Then assume that $\epsilon_{i+1} = C\epsilon_i^r$ for all values of i, where C and r are constants independent of i. Plug this assumption into your recursion relation and solve for C and r. The value of r should be the golden ratio.

2. Implement the bisection, Newton–Raphson, and secant methods to solve the generic problem $f(x) = 0$, writing functions that take the function f (and possibly its derivative) and one or two initial x's as parameters. For your Ph20 Beautiful Plot$^{\text{TM}}$ of the week, compare the convergence rates of the three methods on the simple function $f_c(x) = \sin(x) - c$, with $|c| < 1$.

3. The binary pulsar 1913+16 is a famous binary system where a rotating neutron star is in orbit around another object, probably another neutron star. The rotating neutron star is a pulsar: that is, it emits a beam of radio waves in a direction that changes with the rotation of the star; the radio emmision is seen on earth as a series of 'pulses'—one pulse whenever the radio beam points at the earth. This pulsed radiation is so regular it can be used as a very

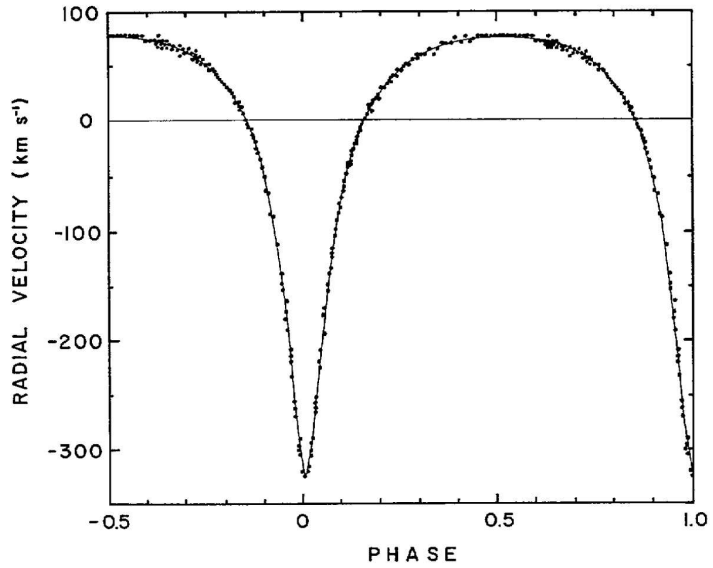

Figure 3: Radial velocity vs. phase diagram for the pulsar 1916+13, from R. A. Hulse and J. H. Taylor, *Astrophys. J.* **195**, L51 (1975).

accurate clock, so accurate that it can be used to measure the Doppler shifts caused by the orbital motion of the two neutron stars about each other. These Doppler shift measurements yield a very small orbital period corresponding to a very tight orbit (the orbital period is $\sim 8\,\mathrm{hr}$). In such an orbit gravitational radiation becomes important, and indeed this system provided the first proof of the existence of gravitational radiation. The 1993 Nobel Prize in physics was awarded to Hulse and Taylor for the discovery of this system.

Using your favorite method, solve for the elliptical orbit of this system using the equation for ξ in terms of t, and the equations for $\{x, y\}$ in terms of ξ; then plot the orbit. The relevant parameters are: $e = 0.617139$, $T = 27906.98161$ seconds, and $a = 2.34186\,\mathrm{s} \times c$, where c is the velocity of light. You may use a as if it were the semi-major axis, although actually it is the *projected* semi-major axis.

4. The velocity of the pulsar along the line-of-sight to the earth (measured using the Doppler shifts of the incoming pulsar radiation) is plotted in Fig. 3. You should see if your orbit agrees with this data for some orientation of the orbit with respect to earth. To do this, obtain the velocities $x'(t/T)$ and $y'(t/T)$ by the approximate finite-difference formulas

$$x'(t) \approx [x(t + \Delta t) - x(t)]/\Delta t, \quad y'(t) \approx [y(t + \Delta t) - y(t)]/\Delta t, \tag{9}$$

(for a small enough Δt) then project (dot-product) the vector $\{x'(t), y'(t)\}$ along the unit vector $\{\cos\phi, \sin\phi\}$. Can you find an angle ϕ that gives a good agreement with Fig. 3? (Qualitative agreement is enough.)

Hint: Plot velocities in units of km/s, and time in units of one orbital period; you may have to shift your velocity curve horizontally to account for the *initial phase* of the binary (this phase is not the same thing as the angle ϕ, which models the orientation of the binary with respect to the receiver).