Matthew Evans, Andrew Flint, R. Robert Hentzel
Harvey Mudd College
1995 Outstanding Winner
数学中国提供 http://www.madio.net

A Specialized Root-Finding Method for Rapidly Determining the Intersections of a Plane and a Helix

Matthew Evans
Andrew Flint
Noah Kubow
Harvey Mudd College
Claremont, CA 91711
{mevans, aflint, nkubow}@hmc.edu

Advisor: David L. Bosley

Introduction

Our problem is to locate all of the intersections between a helix and a plane that are in general position.

The problem statement leaves several potentially significant parameters unspecified. In the most general case, solutions may be entirely intractable. Certainly, such cases would be computationally difficult and therefore inappropriate for real-time simulations. At the onset of our investigation, we made the following assumptions:

- **Software application.** The problem is motivated by a desire to predict intersections using computer software, and therefore any solution should be proposed as if it were the computational engine for a larger package. We further assume that all relevant information about the plane and helix is passed into this engine from the user interface.
- **Nonzero tolerance.** The engine should be expected to locate approximate points of intersection within a certain numerical tolerance. Exact solutions are not required for graphical applications. Particularly when a rapid sequence of solutions is desired, the tolerance should increase, to minimize computation time.
- **Frame-by-frame animation.** We approach the problem of real-time simulation as a finite sequence of discrete static instances of the general problem. For example, a 90° rotation of the plane is simulated by a handful of fixed relative orientations, which the engine solves sequentially. Each solution set is then used to construct a single frame in the sequence, which is animated for the user in real time.

更多数学建模资料请关注微店店铺“数学建模学习交流”
https://k.weidian.com/RHO6PSpA

- **Nondegenerate, regular finite helix.** We assume that the helix is circular, of finite height, and with a uniform pitch and nonzero radius at any fixed time.
- **Effectively infinite plane.** We consider only an infinite plane, since the possibility of a helix sneaking around the edge of a plane segment adds considerable difficulty to the problem, and we do not believe that is in the spirit of the problem as stated.
- **Relative coordinate system.** The positions of any intersections are to be generated by the engine in its own coordinate system and are then passed out to the calling function. If necessary, the calling function then rescales and translates these points to a coordinate system appropriate for the user interface, including projecting these points in three-space onto a two-dimensional screen.

Constructing the Model

On-screen, a helix and plane may be oriented in virtually any manner. In constructing a model, however, we are concerned only with the relative orientations and positions of the two objects. In light of this, we are free to choose an arrangement that is easiest to treat mathematically.

Assigning a Coordinate System

We begin by assigning a coordinate system to the model. First, we choose one end of the helix to be the initial point and the other end to be the terminal point. The z-axis is the axis of the helix, and the x-axis contains the initial point. The origin of the coordinate system is thereby fixed at the bottom of the helical axis. The orientation of the helix in this coordinate system is shown in **Figure 1**. For the purposes of discussion, we consider only right-handed helices, but the model can easily be extended to left-handed ones.

To locate the plane in this coordinate system, we take $(0, 0, z_0)$ to be the point of intersection between the plane and the z-axis. This intersection is guaranteed to occur provided that the vector normal to the cutting plane is not perpendicular to the z-axis (a special case, which we address later).

Since we have located the point $(0, 0, z_0)$, the plane is completely specified by the angles θ_0 and ϕ_0. In our representation, θ_0 is the angle formed by the line of intersection of the cutting plane with the plane $z = z_0$, as measured counterclockwise from the x-axis. The angle ϕ_0 is the angle of declination of the cutting plane from the z-axis. One could think of this as creating a plane specified by z_0, θ_0, and ϕ_0 by starting with a vertical

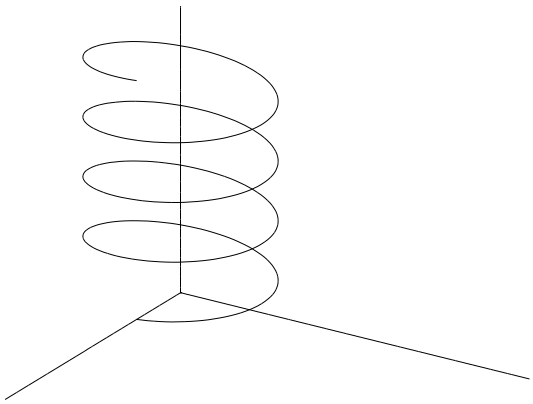

Figure 1. The helix-defined coordinate system.

plane along the xz-axis, then rotating the plane about the z-axis counter-clockwise through θ_0 radians, rotating back from the z-axis by ϕ_0 radians, and finally translating directly upward to z_0. The orientation of the plane in the coordinate system is shown in **Figure 2**.

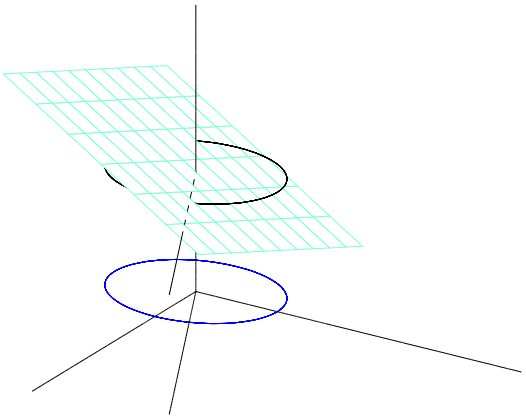

Figure 2. Fixing the plane in the coordinate system.

Parametrizing the Problem

By defining our coordinate system in this way, we can preserve any relative orientation of a helix and a cutting plane while fixing the helix in a vertical position. In these coordinates, we can easily describe these two objects with explicit equations. If the helix has radius R, pitch p, and length L, then it is given in rectangular coordinates by the parametric equations

$$x = R\cos 2\pi t, \qquad y = R\sin 2\pi t, \qquad z = pt, \qquad 0 \le t \le L/p. \qquad \textbf{(1)}$$

For $R > 0$, this generates a right-handed helix.

Similarly, the plane can be described directly in rectangular coordinates by

$$z = -\sin\theta_0 \tan\left(\tfrac{\pi}{2} - \phi_0\right) x + \cos\theta_0 \tan\left(\tfrac{\pi}{2} - \phi_0\right) y + z_0. \qquad \textbf{(2)}$$

For a derivation of this result, see **Appendix A**.

Now consider a right circular cylinder of radius R and height L, centered at the origin and resting on the xy-plane. This cylinder is given by

$$x = R\cos 2\pi t, \qquad y = R\sin 2\pi t, \qquad z = s, \qquad 0 \le s \le L. \qquad \textbf{(3)}$$

This cylinder contains the helix, and the intersection of the cylinder and the cutting plane must contain all of the intersections between the helix and the plane. The intersection of the cylinder and the plane is just the projection of the cylinder onto the cutting plane, which is the curve

$$z = R\tan\left(\frac{\pi}{2} - \phi_0\right)\sin(2\pi t - \theta_0) + z_0,$$

as determined by the simultaneous solution of the equations for z in **(2)** and **(3)**. See **Appendix A** for details.

A Visual Interpretation

Further insight into the problem of finding intersections between the helix and the cutting plane can be gained from the following illustration. Suppose that we place a bead on the top of the helix and allow it to slide downward along the helix at a constant rate. If there is a bright light directly above the bead, we will see the bead's shadow repeatedly tracing out an elliptical curve on the cutting plane. We further endow the bead with the magical ability to slide directly through the plane without stopping. Each time the bead comes into contact with its shadow, the bead must be passing through the plane.

In mathematical terms, the motion of the bead is described by **(1)**—if you invert time, or reverse the influence of gravity, anyway—and the motion of its shadow is given by **(3)**, simply allowing t to run on as in **(1)**. Clearly, then, since the x- and y-components of the bead and its shadow are equivalent for all t, we really search only for those instances when they share the same z-component.

Thus, we would like to find all solutions to the equation

$$pt = R\tan\left(\tfrac{\pi}{2} - \phi_0\right)\sin(2\pi t - \theta_0) + z_0, \qquad 0 \le t \le L/p. \qquad \textbf{(4)}$$

Unfortunately, this is a transcendental equation whose solutions cannot be found analytically for $\phi_0 \neq 0$. When $\phi_0 = 0$, this equation is not a valid description of the helical intersections with the cutting plane; we have found analytical solutions to the problem in this case (see **Appendix B**).

Despite the fact that **(4)** has no analytical solutions when $\phi_0 \neq 0$, we have developed a solution algorithm that takes advantage of the unusual characteristics of the equation. A presentation and discussion of our algorithm is given in the following section. Four views of the intersection of a sample helix and plane are shown in **Figure 3**.

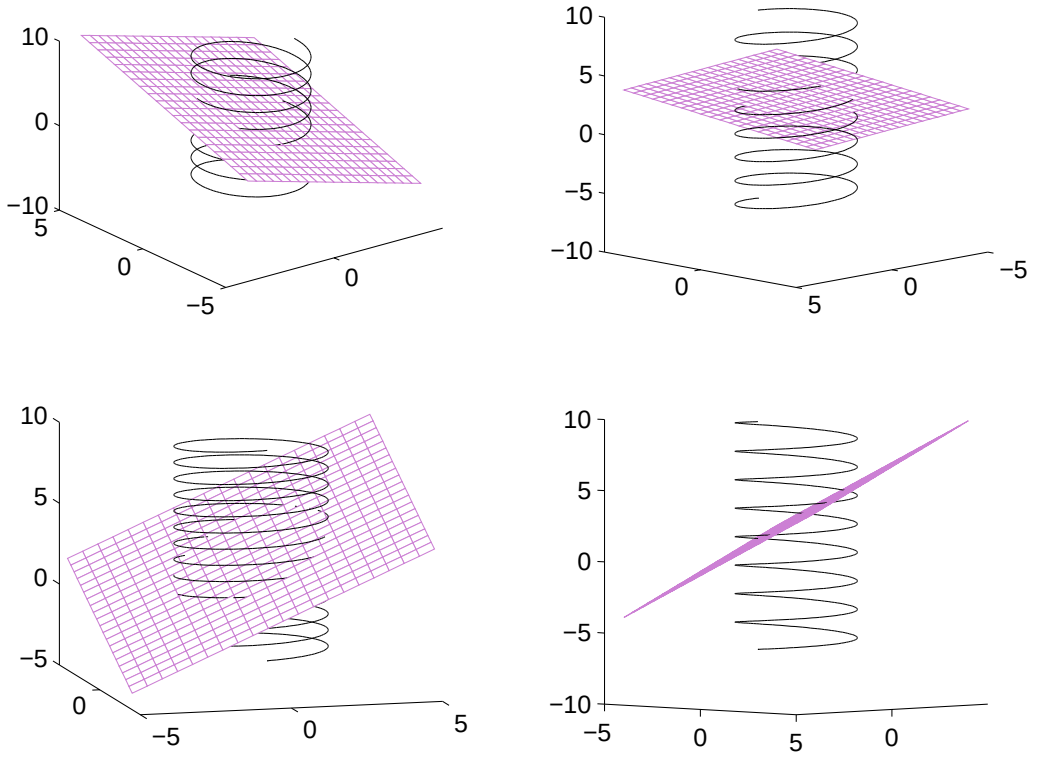

Figure 3. Four views of a typical plane-helix intersection.

Nondimensionalizing the Problem

Eq. **(4)** can be somewhat difficult to deal with. To allow for easier analysis, we nondimensionalize the equation using the definitions

$$\tau = 2\pi t, \qquad \beta = \frac{2\pi_0}{p}, \qquad \sigma = \frac{2\pi R \tan\left(\frac{\pi}{2} - \phi_0\right)}{p}.$$

This nondimensionalization allows us to consider the equation

$$\tau = \sigma \sin(\tau - \theta_0) + \beta.$$

We then define a function

$$f(\tau) = \sigma \sin(\tau - \theta_0) + \beta - \tau \qquad \textbf{(5)}$$

and attempt to solve $f(\tau) = 0$. By finding the roots of f, we find the points of intersection between the helix and the plane. Having found a root $\tau^*$of f, we simply divide this value by 2π and substitute into the parametric equations in **(1)** to locate the points of intersection in Cartesian coordinates.

Analysis of the Model

In proceeding to analyze this model, we first produced a plot of f, as shown in **Figure 4**. The curve represents the vertical separation between the bead and its shadow for any τ.

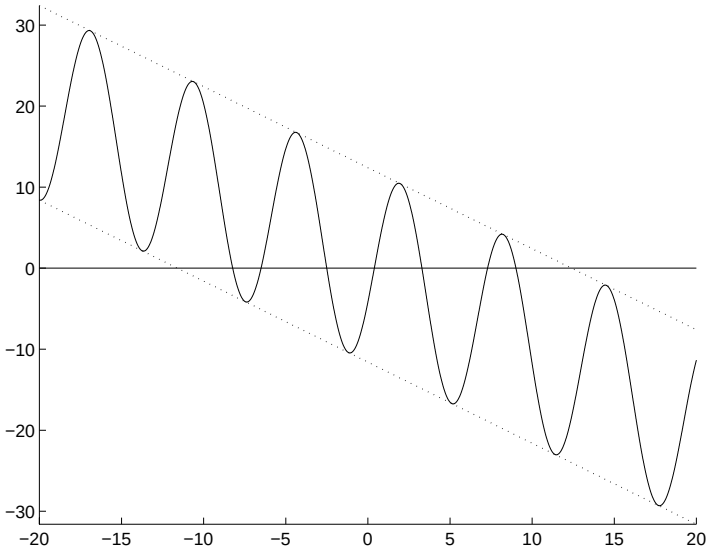

Figure 4. A plot of the function f, of vertical separation between the bead and its shadow, as a function of τ.

First note that the function f is bounded above by $\sigma \cdot 1 + \beta - \tau$ and below by $\sigma \cdot (-1) + \beta - \tau$, the dashed lines in **Figure 4**. These lines cross the τ-axis at $\beta \pm \sigma$, so we can limit our search for roots of f to τ in the interval $(\beta - \sigma, \beta + \sigma)$. Of course, for real-time applications, even limiting the search for intersections to this interval may not let us achieve sufficiently fast solutions. Furthermore, the size of this interval depends on variables controlled by the user and cannot be guaranteed to be small. Moreover, because of the rapid oscillations of the function, most standard root-finding algorithms, such as Newton's method or the bisection method, will not perform adequately on this function.

A Fast Approximation

As a first approximation, we devised a root-finding technique that constructs a linear sketch of f. The local maxima and minima of f can be found

analytically by locating the roots of its first derivative. The solutions of

$$\frac{df}{dt} = \sigma \cos(\tau - \theta_0) - 1 = 0$$

are given by

$$\tau = \theta_0 \pm \arccos \frac{1}{\sigma} + 2\pi n, \qquad \text{for } \sigma > 1.$$

(The special case when $|\sigma| \leq 1$ is addressed in **Appendix B**.) We use this family of points to create a jagged linear approximation g of f, by connecting each maximum to its immediately surrounding minimum, and vice versa. **Figure 5** illustrates g. From this we can easily (and quickly) estimate the roots of f by finding the roots of g. It is especially important to note that because g is hinged on the local maxima and minima of f, there will be precisely the same number of roots of g as of f. In other words, *this method of root approximation can never miss a real intersection, or introduce an artificial intersection, of the helix and the plane*.

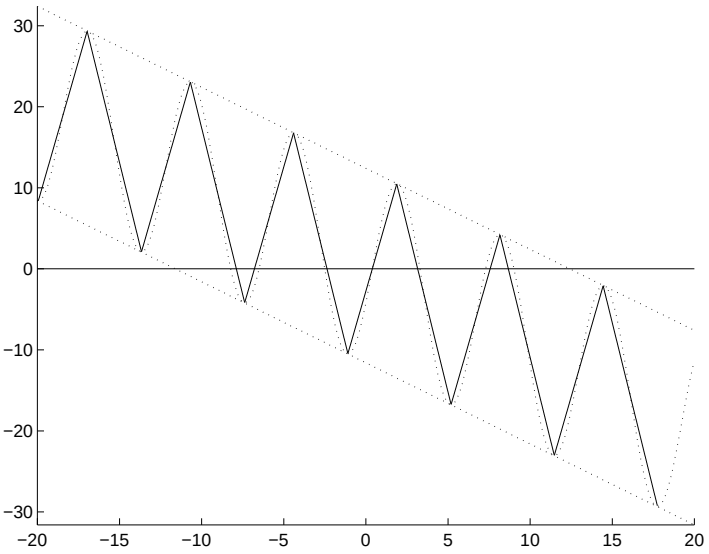

Figure 5. The function g, which approximates f.

From **Figure 4**, we see that the roots of f occur, in increasing τ, either between a maximum and the following minimum, or between a minimum and the following maximum. We refer to the former as a "descending" root and the latter as an "ascending" root. According to the derivation in **Appendix C**, all of the ascending roots are separated by a constant interval in τ. The constant interval is $w\pi(1 + 1/s)$, where s is the slope of the line connecting (in the case of ascending roots) a maximum to the following minimum. Descending roots are separated by a different constant interval in τ, with the same expression for τ but with s being the slope between a maximum and the following minimum.

For example, having located the first ascending root of g at τ_1, we know that the $(n+1)^{\text{st}}$ ascending root occurs at $\tau_1 + 2\pi n(1+1/s)$. In this way, we can generate the complete collection of g's roots almost immediately. Only those n roots corresponding to roots on the finite helix need to be considered, thus limiting n. The limits are derived in **Appendix C**.

This calculation of the roots of g constitutes a rough but very fast estimate of the roots of f and in some cases may actually suffice for real-time graphical applications. If the calling program allows a generous enough tolerance in the coordinates of the helical intersections, this initial collection of approximate roots will be adequate, and the problem can be considered solved for the current frame in the animation sequence.

It should further be noted that as more intersections occur, the linear approximation should generate increasingly accurate estimates. In terms of the model, the user may increase the number of intersections by either increasing R (the radius of the helix), decreasing p (the pitch of the helix), or decreasing ϕ_0 (the inclination of the cutting plane with the helical axis). Note that all of these changes cause an increase in the nondimensional parameter σ. Therefore, the vertical distance between adjacent maxima and minima increases, and the linear approximation g becomes more and more accurate as an indicator of the roots of f. Thus, as the apparent complexity of the problem increases, our algorithm experiences only a nominal increase in runtime and achieves more accurate first approximations.

A Rapid Root Search

In many cases, however, the roots of the linear approximation to f may not satisfy the accuracy requirements for the current frame. In this event, the algorithm engages in a more precise method for finding the roots of f.

Taking as seed values the roots of g, we use a modified Newton's method to zero in on the roots of f. The method is modified by taking an approximation of the derivative of f at the seed point. Essentially, we use two approximations to the derivative of f. The constant approximation to an upward slope is given by $\pi/2\sigma$, and $-\pi/2\sigma$ is used to approximate a negative slope. These constants are used for improving the location of descending and ascending roots, respectively. The determination of these approximations is given in **Appendix C**.

We use this alternative to a true Newton's method for two reasons.

- Each computation of the derivative entails computing a cosine function, which is orders of magnitude more time-consuming than a simple variable lookup. By using two constant values, we significantly reduce the computation time.

- Newton's method often has trouble locating the roots of functions with periodic derivatives, such as f.

As in Newton's method, we evaluate the function f at a seed point τ_s to determine the direction and magnitude of the error there. The next approximation is $f(\tau_s) \times \pi/2\sigma$ plus a small perturbation. This process is repeated until the approximation is sufficiently close to the root, yielding an estimated intersection within the desired tolerance. The perturbation has period three, so our linear method is unlikely to fall into an indefinite oscillation about a root. Additionally, the multiplicative factor $\pi/2\sigma$ is well below π/σ, which is the limit on jump size that prevents the algorithm from skipping too far from the approximate root and missing the actual root all together. A more detailed discussion of this technique and the parameters chosen in presented in **Appendix C**.

Testing the Model

We coded our algorithm in C++ and ran several test cases to confirm its root-finding capabilities for this particular problem. Our trials suggest not only that the engine is very rapid in its approximations of the roots of f, but also that it can attain a great level of accuracy with a nominal time penalty. The code itself is presented in **Appendix D**. [EDITOR'S NOTE: Omitted.]

Runtime

We programmed a series of ten frames, with each frame representing a 10° rotation of the cutting plane about the z-axis. The angle of declination ϕ_0 was fixed at 45°, and all parameters describing the helix were held constant. We feel this might represent a typical course of duties demanded by the user. On average, the algorithm calculated about ten points of intersection between the plane and helix in each frame and was able to generate all ten frames in 0.4 seconds. This indicates an average speed of about 25 frames per second.

To put the algorithm to a more demanding test, we then programmed a series of 100 frames. This time, all parameters were permitted to vary randomly (within appropriate bounds) between frames. We found basically the same runtime estimate: 20–25 frames per second.

Accuracy

In each of our trial runs, we specified a tolerance level that corresponds to an allowable error in the distance between an estimated intersection and an actual intersection in three-space. Within the algorithm, this tolerance is transformed into an upper bound on the allowable error in the root approximation. In every case tested, our algorithm was able to satisfy this accuracy

test. This is because the algorithm will always succeed in finding an actual root within the precision of the machine or that requested by the calling program: Because we use a period-three perturbation in the root-finding iterations, the algorithm cannot bounce indefinitely around the root.

Furthermore, our root-finding method outperformed the root-finding routines of Mathematica (which uses Newton's method), which gave approximations always farther from the true root than our algorithm's approximations.

A Graphical Test

The graphical results of a simulation are presented in **Figure 6**.

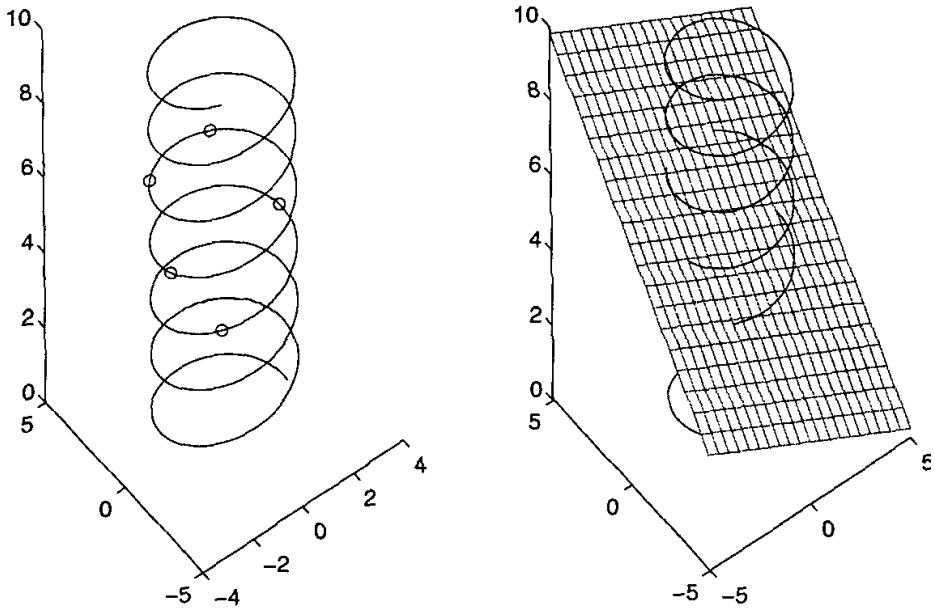

Figure 6. Intersection points as generated by our algorithm.

Critique of the Model

Strengths of the Model and Algorithm

By fixing the coordinate system about the helix, we can easily construct parametric equations to describe the helix in Cartesian coordinates. This representation in turn allows us to construct a relatively simple equation, the roots of which correspond to actual intersections between the helix and the cutting plane. These roots can easily be translated to give the locations of the intersections in Cartesian coordinates.

Our algorithm for finding the roots of **(5)** also has several notable features. First, by using a linearized approximation to f, we are guaranteed never to miss a root. That is, the number of approximate roots found from the linearization g will always equal the true number of roots of f. Furthermore, our root-searching technique can always improve on these estimates of the roots of f to within the desired accuracy. That is, for computational purposes, the algorithm always finds the correct roots.

In addition, since we compute all the parameters of **(5)** for each new frame, any parameter describing the system can vary arbitrarily between consecutive frames. For example, although the radius of the helix is assumed to be uniform along its entire length in one frame, the radius may increase or decrease in the successive frame with no performance penalty. Similarly, both the length and the pitch of the helix and the relative orientation of the plane to the helix can vary between frames without disturbing the root-finding capabilities of the engine.

Our algorithm can determine roots very quickly. In fact, as the number of intersections—and thus the number of roots—increases, the speed with which these roots are determined increases, because the first approximation of the roots is more likely to satisfy the accuracy criterion.

Finally, the algorithm exhibits very little sensitivity to the input parameters. That is, there are no pathological cases for which the algorithm fails. The model represented by **(5)** fails to describe the desired situation only when the cutting plane is parallel to the helical axis, for which case we give analytical results.

Weaknesses of the Model

Any weaknesses of the model are present in the assumptions of the model. The model assumes a helix of only finite length, with a uniform radius and pitch along its entire length. Provided the plane is not parallel to the helical axis, intersections can occur over only a finite stretch of the helix. More important, the model does not succeed in representing a helix that varies in radius or pitch, though such a case may not even qualify as a true helix to some.

Another drawback is that the model considers only a static relationship between the helix and the plane. No attempt has been made to incorporate, for example, a rate of change of declination of the plane from the helical axis. Instead, our model assumes that relative motion between the two objects occurs in consecutive discrete steps.

The algorithm to find the roots of equation **(5)** also has some drawbacks. As with any computational routine, runtime is bound to increase when greater accuracy is desired. For primarily graphical applications, however, extreme accuracy is seldom required. Furthermore, the runtime limitations are dramatic only in the case of σ approaching 1 from above.

In addition, computational techniques inherently introduce numerical errors in solutions; in some cases this may cause misleading results. For example, consider a situation where the cutting plane doesn't cut directly through the helix but instead contains only a single point where the helix is tangent to the plane. This would correspond to a point in **(5)** where $f = 0$ and $df/d\tau = 0$. In this case, our algorithm would find two roots, one corresponding to an ascending root and the other to a descending root. However, the roots would be identical within machine error (not the user-requested error) and thus would yield two identical points of intersection in three-space.

Appendix A: Derivations of Equations

Derivation of the Plane Equation

A plane is determined by a normal vector (a, b, c) and a point (x_0, y_0, z_0) in the plane. The z-coordinate of the plane can be written as a function of x and y as

$$z = \frac{a}{c}(x - x_0) + \frac{b}{c}(y_0 - y) + z_0.$$

The cutting plane is defined so that it contains $(0, 0, z_0)$. Furthermore, a vector normal to the cutting plane is given by the cross product

$$\vec{n} = (\cos\theta, \sin\theta, 0) \times \left(-\cos\left(\tfrac{\pi}{2} - \phi_0\right)\sin\theta, \cos\left(\tfrac{\pi}{2} - \phi_0\right)\cos\theta, \sin\left(\tfrac{\pi}{2} - \phi_0\right)\right).$$

This product reduces to

$$\vec{n} = \left(\sin\theta\sin\left(\tfrac{\pi}{2} - \phi_0\right), -\cos\theta\sin\left(\tfrac{\pi}{2} - \phi_0\right), \cos\left(\tfrac{\pi}{2} - \phi_0\right)\right).$$

Thus, the z-coordinate of the cutting plane as a function of x and y is

$$z = -\sin\theta_0\tan\left(\tfrac{\pi}{2} - \phi_0\right)x + \cos\theta_0\tan\left(\tfrac{\pi}{2} - \phi_0\right)y + z_0.$$

The Equation of the Elliptical Intersection

Combining the equations for x and y given in **(1)** and the equation for z in **(2)** yields the intersection of the cutting plane and the cylinder enclosing the helix:

$$\begin{aligned} z_e &= -\sin\theta_0\tan\left(\tfrac{\pi}{2} - \phi_0\right)R\cos 2\pi t + \cos\theta_0\tan\left(\tfrac{\pi}{2} - \phi_0\right)R\sin 2\pi t + z_0 \\ &= R\tan\left(\tfrac{\pi}{2} - \phi_0\right)(\cos\theta_0\sin 2\pi t - \sin\theta_0\cos 2\pi t) + z_0 \\ &= R\tan\left(\tfrac{\pi}{2} - \phi_0\right)\sin(2\pi t - \theta_0) + z_0. \end{aligned}$$

Appendix B: Special Cases

The Case of a Vertical Cutting Plane

When $\phi_0 = 0$, the cutting plane is parallel to the z-axis, and the parameter z_0 has no interpretation. To describe the plane, let the perpendicular distance from the plane to the z-axis be r_0. The equation of the plane becomes

$$r = \frac{r_0}{\sin(2\pi t - \theta_0)}.$$

Note that if $r_0 > R$, there are no intersections between the plane and the helix. Otherwise, finding the intersections of a vertical cutting plane with the helix is equivalent to finding all t that satisfy either of the following equations

$$R = \begin{cases} \dfrac{r_0}{\sin(2\pi t - \theta_0 - 2\pi n)} \\ \dfrac{r_0}{\sin(\pi - 2\pi t + \theta_0 - 2\pi m)}, \end{cases}$$

where $m, n \in \mathbb{Z}$. Solving for t yields infinite families of solutions

$$t = \begin{cases} \dfrac{\theta_0 + \arcsin \frac{r_0}{R}}{2\pi} + n \\ \dfrac{\theta_0 - \arcsin \frac{r_0}{R}}{2\pi} + m + \dfrac{1}{2}. \end{cases}$$

Having found these values of t, we need only substitute them back into the equations that define the helix to determine all of the intersection points exactly. Because the helix is finite, we expect that this will hold for only a finite selection of ms and ns, the bounds on which are given in **Appendix C**.

The Case $\sigma \leq 1$

For $\sigma \leq 1$, the derivative $df/d\tau = \sigma \cos(\tau - \theta_0) - 1$ is nonpositive for all τ. Thus, f is monotonically nonincreasing and crosses the axis exactly once. Since this root must occur between $\beta + \sigma$ and $\beta - \sigma$, a simple bisection search can be used. Our algorithm starts the bisection search at β and has an initial step size of $\sigma/2$.

Appendix C

Constructing a Linear Approximation of f

The Slope of g

Finding the slopes of the line segments from which g is constructed can be broken down into two parts: finding the slope of the ascending segments and finding the slope of the descending segments.

First we find the slope of the ascending segments. Since all of these segments are parallel, we may choose any one of them as the basis for our problem. Let the endpoints of the segment be denoted by τ_{M} (a maximum) and τ_{m} (the preceding minimum). From the roots of $df/d\tau$ we find

$$\tau_{\mathrm{M}} = \theta_0 - \arccos\frac{1}{\sigma}, \qquad \tau_{\mathrm{m}} = \theta_0 + \arccos\frac{1}{\sigma}.$$

The slope of the line connecting these two points is given by

$$s_d = \frac{f(\tau_{\mathrm{M}}) - f(\tau_{\mathrm{m}})}{\tau_{\mathrm{M}} - \tau_{\mathrm{m}}},$$

which, upon application of the definition of f (and a little algebra), reduces to

$$s_d = \frac{\sqrt{\sigma^2 - 1}}{\arccos\frac{1}{\sigma}} - 1.$$

A similar approach finds the slope of the descending segments, using the same τ_{M} but with τ_{m} being the minimum that follows it rather than precedes it. Again, from the roots of $df/d\tau$, we have

$$\tau_{\mathrm{M}} = \theta_0 - \arccos\frac{1}{\sigma} + 2\pi, \qquad \tau_{\mathrm{m}} = \theta_0 + \arccos\frac{1}{\sigma},$$

which, using the definitions of slope and of the function f, yields

$$s_d = \frac{-\sqrt{\sigma^2 - 1}}{\arccos\frac{-1}{\sigma}} - 1.$$

The Roots of g

Once the slopes for g have been found, finding the equations of the lines from which g is made is a simple matter of using a point that is known to be on one of the lines in combination with the slope. Evaluating f at $\tau = \theta_0$ yields the point $f(\theta_0) = \beta - \theta_0$ on the descending line segment used in finding the slope. Thus, this line has a root at $\tau_d = \theta_0 - (\beta - \theta_0)/s_d$. In the same vein, we have $f(\theta_0 + \pi) = \beta - \theta_0 - \pi$, this time on the ascending line segment used in

finding the slope. Thus, this line has a root at $\tau_d = \theta_0 + \pi - (\beta - \theta_0 - \pi)/s_d$. Finally, the equations of the lines are

$$g_{d_0}(\tau) = s_d(\tau - \tau_d), \qquad g_{a_0}(\tau) = s_a(\tau - \tau_a).$$

Since $f(\tau + 2\pi) = f(\tau) - 2\pi$, all of the lines used in the construction of g are given by

$$g_{d_n}(\tau) + 2\pi n = s_d(\tau - \tau_d - 2\pi n), \qquad g_{a_m}(\tau) + 2\pi m = s_d(\tau - \tau_a - 2\pi m),$$

where $m, n \in \mathbb{Z}$, which can be reduced to

$$\begin{aligned} g_{d_n}(\tau) &= s_d\left[\tau - \tau_d - 2\pi n\left(1 + \frac{1}{s_d}\right)\right], \\ g_{a_m}(\tau) &= s_d\left[\tau - \tau_a - 2\pi m\left(1 + \frac{1}{s_a}\right)\right]. \end{aligned}$$

This formulation shows that the roots of the lines from which g is constructed simply repeat in τ every $2\pi(1 + 1/s)$. Thus, the set of all of these roots is given by

$$\begin{aligned} \tau_{d_n} &= \theta_0 - \frac{\beta}{s_d} + 2\pi n\left(1 + \frac{1}{s_d}\right), \\ \tau_{a_m} &= \theta_0 - \frac{\beta}{s_a} + 2\pi m\left(1 + \frac{1}{s_a}\right) + \pi. \end{aligned} \qquad \textbf{(6)}$$

Since f can have roots only between $\beta \pm \sigma$, we need concern ourselves only with the roots of g that lie between these bounds. Further limits can be placed on the range of possible roots by recalling that the helix is of finite length and thus intersections can occur only in $0 \le \tau \le 2\pi L/p$. Calling the upper bound τ_h (the lesser of $2\pi L/p$ and $\beta + \sigma$) and the lower bound τ_ℓ (the greater of 0 and $\beta - \sigma$), the limits on the integers n and m are given by

$$\begin{aligned} \tau_\ell \le \quad & \theta_0 - \tfrac{\beta}{s_d} + 2\pi n\left(1 + \tfrac{1}{s_d}\right) && \le \tau_k \\ \tau_\ell \le \quad & \theta_0 - \tfrac{\beta}{s_d} + 2\pi m\left(1 + \tfrac{1}{s_a}\right) + \pi && \le \tau_k. \end{aligned}$$

Solving these equations for n and m yields

$$\begin{aligned} \frac{\frac{\beta}{s_d} - \theta_0 - \tau_\ell}{2\pi\left(1 + \frac{1}{s_d}\right)} &\le n \le \frac{\frac{\beta}{s_d} - \theta_0 + \tau_k}{2\pi\left(1 + \frac{1}{s_d}\right)} \\ \frac{\frac{\beta}{s_a} - \theta_0 - \tau_\ell - \pi}{2\pi\left(1 + \frac{1}{s_a}\right)} &\le n \le \frac{\frac{\beta}{s_a} - \theta_0 + \tau_k - \pi}{2\pi\left(1 + \frac{1}{s_a}\right)}. \end{aligned}$$

This range in n and m, when substituted back into **(6)**, produces all of the roots of g, and thus we have a set of approximations to all of the roots of f.

Quick and Safe Approximation of the Slope of f

Since the period of $df/d\tau$ is 2π, a change in τ that is greater than π could move us from the search for one ascending (or descending) root to another. To prevent this, we use $\pi/2$ as the limit for possible changes in our approximation of τ (for any single step). Since f is bounded by $g \pm \sigma$, the maximum value for a given root of g is $|\sigma|$. Thus, a search routine that produces successive approximations by $\tau_n = \pm(\pi/\sigma)f(\tau_{n-1})$ (positive when searching on an ascending segment of g, and negative when on a descending segment) would never move away from its intended target root. However, machine errors could cause this search method to converge to the wrong root, so we use one-half of that quantity.

Furthermore, we add a small (less than 10%) period-three oscillation to the constant to prevent the search algorithm from oscillating about a root with a period other than an integer multiple of three.

Reference

Edwards, C.H., and David E. Penney. 1990. *Calculus and Analytic Geometry.* Englewood Cliffs, NJ: Prentice-Hall.