Modelling a Road Using Spline Interpolation

Kendall Atkinson
Dept of Computer Science
Dept of Mathematics
The University of Iowa
Iowa City, Iowa 52242

February 26, 2002

Abstract

We study the use of cubic spline interpolation to represent the centerline of a road, for curves in both $\mathbb{R}^2$ and $\mathbb{R}^3$. We look at algorithms to create a representation based on arc length and evenly spaced nodes along the centerline. We also consider methods for moving between rectangular coordinates and coordinates based on distance along the centerline and the offset from that centerline (in $\mathbb{R}^2$) and a related decomposition in $\mathbb{R}^3$.

1 Introduction

In road vehicle simulation, we need a way to describe the road. The usual procedure is to give the center line of the road and to then relate the position of the vehicle to this center-line. In the simulation it is necessary to go back and forth between rectangular coordinates and coordinates based on distance along the road and an offset from the centerline. To do this most conveniently, we need a parameterization of the center line based on arc length along that line. In the following, we give a way to create such a parameterization when the given parameterization does not use arc length. We create cubic spline interpolates with nodes that are evenly spaced. Using these and arc length, we can rapidly calculate the position of a vehicle.

The converse problem is to find the arc length and offset from the centerline of a point that is given in rectangular coordinates. To do this, we need to minimize a function of the arc length s that represents the distance of the given point from the point on the centerline with arc length s.

In §§2 and 3, we consider these problems when the road is located entirely within the plane $\mathbb{R}^2$. We generalize these ideas to §§4 and 5. In §6 we discuss the error as it is related to the meshsize used in constructing the approximating spline curve

2 Fitting a planar curve

Let Γ be a planar curve with the parameterization $(x, y) = (f(t), g(t))$, $0 \le t \le b$. We want to produce an accurate cubic spline fit to this curve Γ with the parameterization variable the arc length s along the curve. To begin we must calculate t as a function of s, and we begin by finding the arc length of Γ.

Let $n > 0$ and $h = b/n$; define $t_j = jh$ for $j = 0, 1, \ldots, n$. We produce a curve Γ_{sp} with the parameterization $(x, y) = (f_{sp}(t), g_{sp}(t))$, $0 \le t \le b$, in which the components are cubic spline interpolates that satisfy

$$(f_{sp}(t_j), g_{sp}(t_j)) = (f(t_j), g(t_j)), \qquad j = 0, 1, ..., n \tag{1}$$

This could be based on using either *not-a-knot* boundary conditions or first derivative boundary conditions, if the latter are known. Also, one could use one type of boundary condition at one end of the curve and another type at the other end. With our examples, we use the *not-a-knot* boundary conditions as that is the default used with the *Matlab* function for computing a cubic spline interpolate. If $f, g \in C^4[0, b]$, then

$$\begin{aligned} \|f - f_{sp}\|_\infty, \ \|g - g_{sp}\|_\infty &= O\left(h^4\right) \\ \left\|f' - f'_{sp}\right\|_\infty, \ \left\|g' - g'_{sp}\right\|_\infty &= O\left(h^3\right) \end{aligned} \tag{2}$$

See [4, p. ??]. In further calculations we use Γ_{sp} to replace Γ. Usually we choose a very large choice of n to ensure high accuracy in our computations, to ensure that $\Gamma_{sp} \approx \Gamma$ is an accurate approximation.

The arc length of Γ is given by

$$\begin{aligned} L &= \int_0^b \sqrt{f'(t)^2 + g'(t)^2}dt \\ &\approx \int_0^b \sqrt{f'_{sp}(t)^2 + g'_{sp}(t)^2}dt \equiv L_{sp} \end{aligned} \tag{3}$$

with the latter integral the arc length of Γ_{sp}. Compute an approximation to L_{sp} by using numerical integration, calling it $\widehat{L}_{sp}$. We assume n is even and we use Simpson's numerical integration with n subdivisions. Other integration rules could be used. From (2), it follows that

$$L - L_{sp} = O\left(h^3\right) \tag{4}$$

The arc length to an arbitrary point $(x, y) = (f_{sp}(t), g_{sp}(t))$ on Γ_{sp} is given by

$$s(t) = \int_0^t \sqrt{f'_{sp}(\tau)^2 + g'_{sp}(\tau)^2}d\tau, \qquad 0 \le t \le b. \tag{5}$$

We would like to know t as a function of s, and we would like to find the values of t corresponding to m equally spaced values of s on $\left[0, \widehat{L}_{sp}\right]$, calling them

$\{s_0, s_1, \ldots, s_m\}$ with $s_0 = 0$ and $s_m = \widehat{L}_{sp}$. From (5),

$$\frac{ds}{dt} = \sqrt{f'_{sp}(t)^2 + g'_{sp}(t)^2}, \qquad s(0) = 0$$

But we know that

$$\frac{dt}{ds} = \left[\frac{ds}{dt}\right]^{-1}$$

Thus $t(s)$ satisfies

$$t'(s) = \frac{1}{\sqrt{f'_{sp}(t)^2 + g'_{sp}(t)^2}}, \qquad 0 \le s \le \widehat{L}_{sp}, \qquad t(0) = 0 \tag{6}$$

Use an ODE solver to find $T_j = t(s_j),\ j = 0, \ldots, m$. The accuracy with which the points $\{T_j\}$ are computed will depend on the choice of the ODE solver and the accuracy requested of it. The choice of m will be determined by the accuracy with which Γ is to be approximated.

Calculate the points $(X_j, Y_j) = (f_{sp}(T_j), g_{sp}(T_j)),\ j = 0, \ldots, m$ on Γ_{sp}. These are spaced equally as regards arc length, with an arc length of $\delta \equiv \widehat{L}_{sp}/m$ between each successive pair of points on Γ_{sp}. Construct a new fitting function $(x, y) = (\xi_{n,m}(s), \eta_{n,m}(s))$ on $\left[0, \widehat{L}_{sp}\right]$ using spline functions $\xi_{n,m}(s)$, $\eta_{n,m}(s)$ that interpolate as follows:

$$(\xi_{n,m}(s_j), \eta_{n,m}(s_j)) = (X_j, Y_j), \qquad j = 0, 1, \ldots, m.$$

The parameterization $(\xi_{n,m}(s), \eta_{n,m}(s))$, $0 \le s \le \widehat{L}_{sp}$, is to be used as the model for the center-line of the road. Finding a position on the road at an arc length of s is a simple matter of first determining the appropriate subinterval $[s_{j-1}, s_j]$ in which s is located. The index is given by $j = 1 + [s/\delta]$, where $[\cdot]$ denotes the greatest integer function. Calculating $(\xi_{n,m}(s), \eta_{n,m}(s))$ amounts to evaluating two cubic polynomials, determined by j, and this is a simple calculation involving 8 arithmetic operations for each polynomial.

For extra accuracy in solving for the cubic spline functions $\xi_{n,m}(s), \eta_{n,m}(s)$ on $\left[0, \widehat{L}_{sp}\right]$, we recommend the following when using the *not-a-knot* boundary conditions. In addition to the node points (X_j, Y_j) defined above, also include the additional two points

$$(f_{sp}(t(\delta/2)), g_{sp}(t(\delta/2))), \quad \left(f_{sp}\left(t\left(\widehat{L}_{sp} - \delta/2\right)\right), g_{sp}\left(t\left(\widehat{L}_{sp} - \delta/2\right)\right)\right). \tag{7}$$

With $\{(X_j, Y_j)\}$ and these two additional points, solve for the spline functions $\xi_{n,m}(s), \eta_{n,m}(s)$. Including these two additional points will increase the accuracy of the cubic spline interpolates when s is near to the endpoints of $\left[0, \widehat{L}_{sp}\right]$. With the *not-a-knot* boundary condition, the cubic polynomial on $[0, \delta/2]$ and that

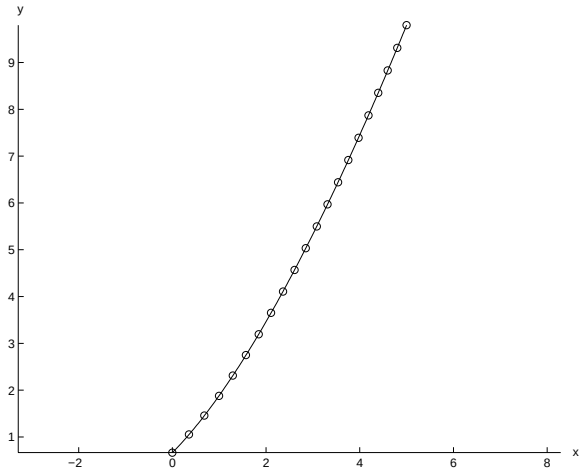

Figure 1: Fitting the curve of (8) with $n = 80$ and $m = 20$.

on $[\delta/2, \delta]$ are the same. Therefore when calculating $(\xi_{n,m}(s), \eta_{n,m}(s))$ on $[0, \delta]$, simply use the polynomial produced for $[0, \delta/2]$. Proceed analogously when calculating $(\xi_{n,m}(s), \eta_{n,m}(s))$ on $\left[\widehat{L}_{sp} - \delta, \widehat{L}_{sp}\right]$.

Example. Consider the curve Γ given by

$$(x, y) = \left(t, \tfrac{2}{3}(t+1)^{\frac{3}{2}}\right), \qquad t \geq 0. \tag{8}$$

The arc length is given by

$$s = \frac{2}{3}\left[(t+2)^{\frac{3}{2}} - \sqrt{8}\right] \tag{9}$$

or equivalently,

$$t = \left(\frac{3}{2}s + \sqrt{8}\right)^{\frac{2}{3}} - 2 \tag{10}$$

Using $n = 80$, $m = 20$ with the construction described above for $0 \leq t \leq 5$, we obtain the cubic spline function shown in Figure 1. The circles are equidistant in arc length along Γ. The true arc length of Γ is given by (9) with $t = 5$; using our schema described above for computing the arc length $\widehat{L}_{sp}$, the error is -1.83×10^{-8}. Note that only n is of importance as a parameter in this computation.

Figure 2 contains the error in the arc length function with $n = 20$. Note that the error in computing the solution to (6) is dependent on only n and the

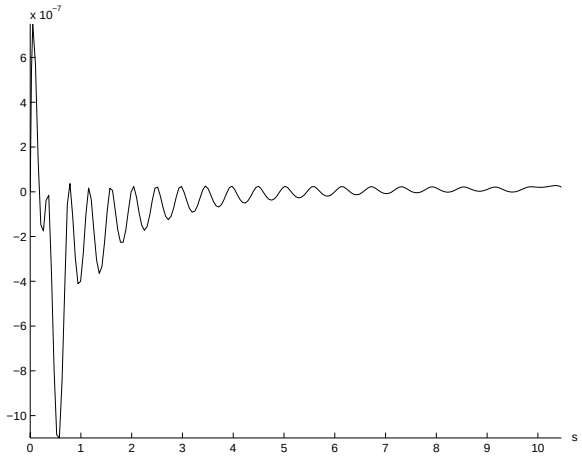

Figure 2: The error in the arc-length function for the computation of (8) using $n = 20$.

n	*Error in* $\widehat{L}_{sp}$	*Ratio*	*Error in* $t(s)$	*Ratio*
10	$-5.72E-5$		$1.00E-5$	
20	$-4.26E-6$	13.4	$1.13E-6$	8.85
40	$-2.85E-7$	14.9	$1.02E-7$	11.1
80	$-1.83E-8$	15.6	$7.70E-9$	13.2

Table 1: Errors in computing arc-length function $t(s)$

accuracy of the ODE solver (and we used the *Matlab* solver *ode45* with a very stringent error tolerance). In Table 1, we give the error in the computation of $\widehat{L}_{sp}$ and the maximum norm of the error in approximating the arc length function for varying values of n. The true arc length of Γ is given by (9) with $t = 5$; and the true formula for the original parameter t as a function of the arc length s is given in (9). The rate of convergence for $\widehat{L}_{sp}$ appears to be $O\left(h^4\right)$, better than predicted by (4), but consistent with the theory given in [2] for a closely related method for computing arc length. The rate of convergence for the approximation of the arc length function $t(s)$ also appears to be $O\left(h^4\right)$; and again this is better than would be expected when considering (2).

As a convenient test of accuracy when the true solution is unknown, which is the usual case, we look at the expression

$$\sqrt{\left(\xi'_{n,m}(s)\right)^2 + \left(\eta'_{n,m}(s)\right)^2} - 1, \qquad 0 \le s \le \widehat{L}_{sp}. \tag{11}$$

This should equal zero. For the above example, the graph of this is shown in

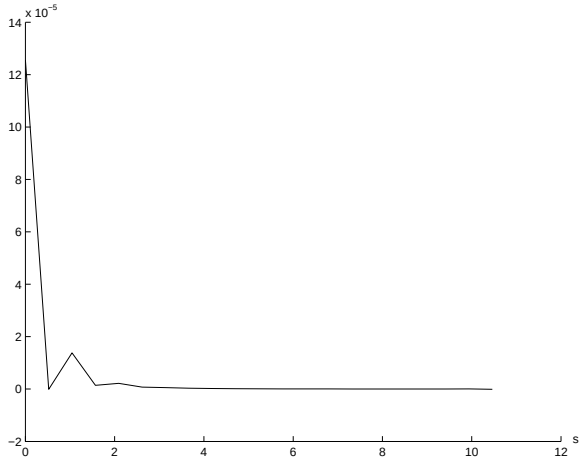

Figure 3: The error test function (11) using $n = 80$ and $m = 20$.

Figure 3 for $n = 80$ and $m = 20$. The error around $t = 0$ is much larger than elsewhere in the interval, although for practical purposes it is still acceptably small (the maximum is 1.26×10^{-4}). Without the use of the extra interpolation points in (7), the error around $t = 0$ is even worse, both for this example and in general (the maximum is then 6.13×10^{-4} for this example).

3 Finding a closest point - Planar case

At each point of the center line of the road, given here by the cubic spline representation

$$(\xi(s), \eta(s)), \qquad 0 \le s \le \widehat{L}_{sp} \tag{12}$$

there is a curvature which describes the reciprocal of the radius (called the *radius of curvature*) of the largest circle that is tangent to the curve at that point. For the example (8) of the preceding section, Figure 4 gives the curvature $\kappa(s)$ as a function of arc length. For a general parameterization with s not necessarily arc length,

$$\kappa(s) = \frac{\xi'(s)\eta''(s) - \eta'(s)\xi''(s)}{\left(\sqrt{(\xi'(s))^2 + (\eta'(s))^2}\right)^3} \tag{13}$$

With s the arc length, the denominator of this fraction is identically 1.

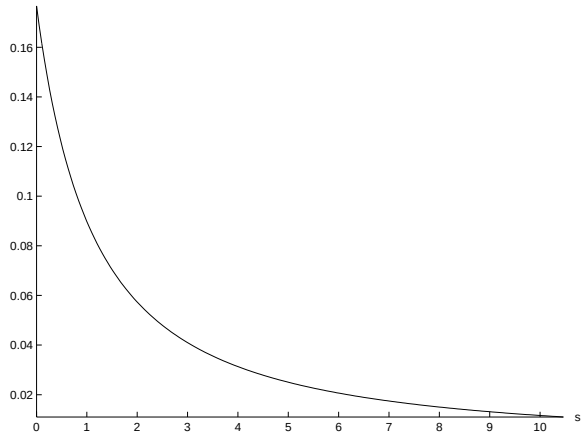

Figure 4: Curvature of (8) as a function of arc length s

If a given point (x, y) is sufficiently close to the center line (12), then we would like to find the closest point on the center line to the given point. Then the arc length of this center line point and its offset from (x, y) is another way to describe the location of (x, y). If the distance from the curve is within the region defined by the radius of curvature, then there is a unique such closest point on the center line.

In practice, the driver of a vehicle will be constantly monitored as to his or her location relative to the center line of the road. The driver will have xy-coordinates (α, β), and it necessary to determine the point (γ, δ) on the center line that is closest to (α, β). This point on the center line will be defined using the arc length parameter s^* at that point; and the distance from (α, β) to (γ, δ) is called the *offset distance* (denote it by ω). Given (α, β), we can usually assume that we know an approximation of the closest subinterval, call it $[s_{j-1}, s_j]$, or a good initial guess, call it σ_0.

Let

$$D(s) = (\alpha - \xi(s))^2 + (\beta - \eta(s))^2, \qquad 0 \le s \le \widehat{L}_{sp} \tag{14}$$

with $(\xi(s), \eta(s)) \equiv (\xi_{n,m}(s), \eta_{n,m}(s))$ as determined earlier in §2 following (7). We need to minimize $D(s)$, with s^* denoting the point at which the minimum is attained (and then $\omega = D(s^*)$). There are three procedures we have used tested in order to determine s^*.

1. **Quadratic fit method**

2. **Newton's method**

3. **Brent's single variable minimization method**

Method 1: Quadratic fit. Given three points $\sigma_1, \sigma_2, \sigma_3$, we fit $D(s)$ with a quadratic polynomial $p(s)$, determined by interpolation to $D(s)$ at $\sigma_1, \sigma_2, \sigma_3$. We then use the minimum of this polynomial as a new estimate σ_4 of s^*. We let $(\sigma_2, \sigma_3, \sigma_4) \to (\sigma_1, \sigma_2, \sigma_3)$ and repeat the process. It can be shown that with a sufficiently good set of initial guesses $\sigma_1, \sigma_2, \sigma_3$, the iterates will converge to s^*.The formula for σ_4 is given by

$$\sigma_4 = \frac{1}{2} \frac{y_{2,3} D(\sigma_1) + y_{3,1} D(\sigma_2) + y_{1,2} D(\sigma_3)}{\sigma_{2,3} D(\sigma_1) + \sigma_{3,1} D(\sigma_2) + \sigma_{1,2} D(\sigma_3)} \tag{15}$$

$$\sigma_{i,j} = \sigma_i - \sigma_j, \qquad y_{i,j} = \sigma_i^2 - \sigma_j^2, \qquad i, j = 1, 2, 3$$

This has a superlinear rate of convergence. For a further discussion, see [6, p. 206].

Method 2: Newton's method. We solve the rootfinding problem

$$D'(s) \equiv -2 \left[(\alpha - \xi(s)) \xi'(s) + (\beta - \eta(s)) \eta'(s) \right] = 0$$

using Newton's method:

$$\sigma_{k+1} = \sigma_k - \frac{D'(\sigma_k)}{D''(\sigma_k)}, \qquad k = 0, 1, 2, \ldots$$

In this case, we need an initial guess $\sigma_0 \approx s^*$; and since the driver is being monitored along the road, it is not difficult to supply such an initial guess. Because $D(s)$ uses the spline functions $\xi(s), \eta(s)$, both $D'(s)$ and $D''(s)$ can be calculated quite efficiently. The method is, of course, quadratically convergent.

Method 3: Brent's single variable minimization method. The original form of this method can be found in [3]. The version used here is the *Matlab* function *fminbnd* and is based on that given in [5, Chap. 9].The method assumes the function $D(s)$ to be minimized is continuous on a given interval $[s_{low}, s_{up}]$, and it finds a local minimizing point s^* within $[s_{low}, s_{up}]$. If there is a unique minimum within the given interval (including possibly an endpoint), then s^* is that point. The method is guaranteed to converge, and the error bound is also guaranteed. The method uses a combination of the quadratic fit method given above and the *golden section search* (cf. [6, p. 199]).

The latter method was the only one that was completely reliable. To test the methods, we generated random choices of s^* and the offset distance ω, and then we produced (α, β) from s^* and ω. Then we attempted to retrieve the given (s^*, ω) and compared it to the original randomly generated values. We also randomly generated nearby initial intervals (needed for methods 1 and 3) and initial guesses (needed for method 2). As an illustration, consider the case illustrated in Figure 1. We generated 100 random values of s^* within the arc

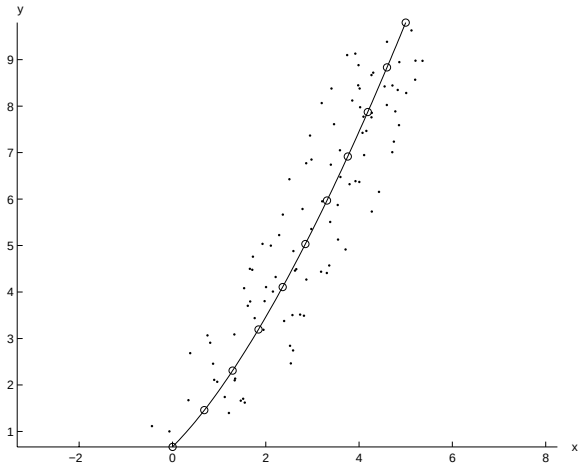

Figure 5: Random test cases for finding a nearest point on the centerline from a given point (α, β); 100 such points (α, β) are given here.

length for the curve, and for each s^* we generated random offsets ω using a maximum offset of size 1. If $s^* \in [s_j, s_{j+1}]$, then we randomly generated an initial guess $[s_i, s_{i+1}]$ for this subinterval with $|i - j| \leq 1$. We formed the point (α, β) and then attempted to recalculate (s^*, ω). We used a requested error tolerance of 10^{-5}, which is probably more than would be needed in practice. With *fminbnd,* we gave to it the subinterval $[s_{i-1}, s_{i+2}]$ as its requested initial interval that was to contain the desired minimizing point. Figure 5 contains the curve of Figure 1, together with the randomly generated points for our test. For a similar test using a 1000 random values, the range of function evaluations for the minimization in *fminbnd* was $[8, 17]$, with 10.1 as the average number of such evaluations. The function being evaluated was the distance of the given point (α, β) from the curve; this involved evaluating the two spline functions $(\xi_{n,m}(s), \eta_{n,m}(s))$ and their first derivatives.

The first two methods worked well in most cases; but there were always cases, seemingly innocuous ones, where the iteration did not converge. The program *fminbnd* was completely reliable, and we recommend its use in the future.

4 Fitting a curve in three dimensions

Let Γ be a curve in $\mathbb{R}^3$ with the parameterization $(x, y, z) = (f(t), g(t), k(t))$, $0 \leq t \leq b$. We want to produce an accurate cubic spline fit to this curve Γ with the parameterization variable the arc length s along the curve. We proceed as

with the planar case in §2, although later we consider some differences. To begin we must calculate t as a function of s, and we begin by finding the arc length of Γ.

Let $n > 0$ and $h = b/n$; define $t_j = jh$ for $j = 0, 1, \ldots, n$. We produce a curve Γ_{sp} with the parameterization $(x, y, z) = (f_{sp}(t), g_{sp}(t), k_{sp}(t))\,, 0 \le t \le b$, in which the components are cubic spline interpolates that satisfy

$$(f_{sp}(t_j), g_{sp}(t_j), k_{sp}(t_j)) = (f(t_j), g(t_j), k(t_j))\,, \qquad j = 0, 1, ..., n \tag{16}$$

This could be based on using either *not-a-knot* boundary conditions or first derivative boundary conditions, if the latter are known. Also, one could use one type of boundary condition at one end of the curve and another type at the other end. With our examples, we use the *not-a-knot* boundary conditions as that is the default used with the *Matlab* function for computing a cubic spline interpolate.

We calculate an approximation $\widehat{L}_{sp}$ to the arc length of Γ as before, by numerical integration. To find $t(s)$, we solve the differential equation

$$t'(s) = \frac{1}{\sqrt{f'_{sp}(t)^2 + g'_{sp}(t)^2 + k'_{sp}(t)^2}}, \qquad 0 \le s \le \widehat{L}_{sp}, \qquad t(0) = 0 \tag{17}$$

We proceed as described above, between (6) and (7), solving at the points

$$\{s_j\} = \{j\delta : 0 \le j \le m\} \cup \left\{\delta/2, \widehat{L}_{sp} - \delta/2\right\}$$

with $\delta \equiv \widehat{L}_{sp}/m$. Using $\{t\,(s_j)\}$, produce cubic spline functions $(\xi_{n,m}(s), \eta_{n,m}(s), \zeta_{n,m}(s))$, $0 \le s \le \widehat{L}_{sp}$, which interpolate Γ at the points $\{s_j\}$.

Example. Consider the curve given by

$$(x, y, z) = (a\cos t, b\sin t, ct)\,, \qquad 0 \le t \le d \tag{18}$$

for some $a, b, c, d > 0$. Figure 6 shows the resulting spline curve with $m = 50$ subdivisions on the original interval $0 \le t \le 4\pi$ and $(a, b, c) = (1, 2, 0.2)$; we used $n = 500$ in the construction of the first spline interpolates.

The comments made for the planar case regarding accuracy all apply here as well.

5 Finding a closest point - 3D case

In the planar case we were given a point (α, β) close to Γ and we needed to find the point on Γ that was closest to the given point. Doing so yielded the arc length parameter s^* and the offset distance ω of this closest point on Γ. In the three dimensional case, we are given a closest point (α, β, γ) and we must calculate three parameters: s^*, the arc length for the closest point on the center line; ω, the offset for the plane of the road; and λ, the *loft* of (α, β, γ) from that plane of the road. We begin by looking at the specification of the road in three dimensions.

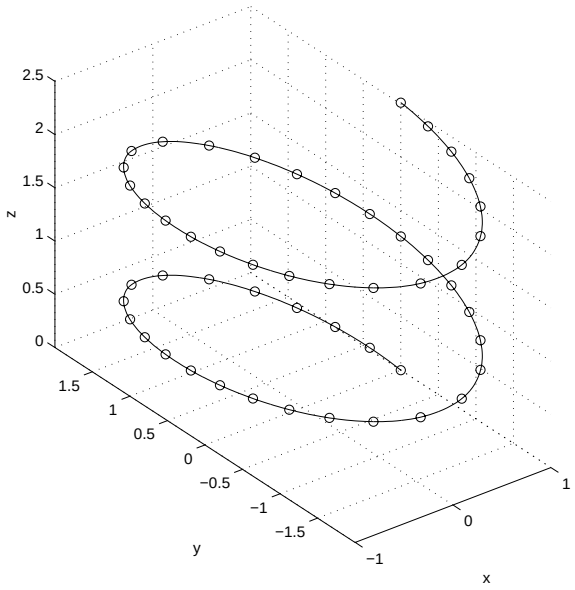

Figure 6: Subdivisions of (18) with $m = 50$ and $(a, b, c) = (1, 2, 0.2)$.

5.1 Specifying the road

The center line can be approximated by cubic splines, as is described in the preceding section. However, we give a development of the specification of the road that is more general, and it can then be applied to the approximating spline approximation $(\xi_{n,m}(s), \eta_{n,m}(s), \zeta_{n,m}(s))$ developed in the preceding section. We give a moving coordinate system along the center line.

Let $\mathbf{v}(t)$ be the unit vector that is tangent to the center line at the position $\mathbf{r}(t) = (f(t), g(t), k(t))$, directed according to increasing t; and let $\mathbf{n}(t)$ denote the unit normal to the road at this point, directed upwards. Let $\mathbf{u}(t)$ denote the unit normal to the center line directed tangent to the road and to the driver's left when driving along the road in the direction of increasing s. Easily, $\mathbf{n}(t) = \mathbf{v}(t) \times \mathbf{u}(t)$, and

$$\mathbf{v}(t) = \frac{\mathbf{r}'(t)}{|\mathbf{r}'(t)|}$$

To specify $\mathbf{u}(t)$ as simply as possible, we give the angle $\theta(t)$ between the road and the horizon as the driver looks to his or her left. When we know $t(s)$, we can also regard θ as a function of s.

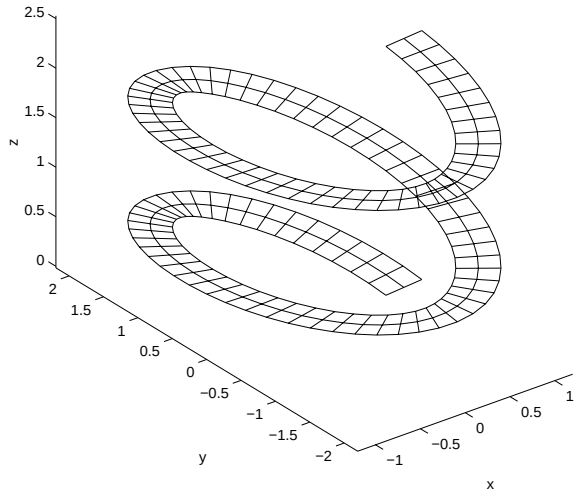

Figure 7: A roadbed based on (18) and (19)

Example. Using the center line (18), define the function

$$\theta(t) = -\frac{\pi}{20}\left(\frac{1+\sin t}{2}\right) \tag{19}$$

which gives a slope downward to the left varying from $\theta = 0$ to $\theta = \pi/20$. Figure 7 contains the road; and the slight slope of the road to the left side of the driver when ascending the road should be apparent.

How do we construct $\mathbf{u}(t)$ from the given $\theta(t)$? Let $\mathbf{u}(t)$ be orthogonal to the center line and let it be located in the road in a direction to the left of the mid-line; and let $\mathbf{u}(t)$ have unit length. To specify $\mathbf{u}(t)$, write it as

$$\mathbf{u}(t) = (\cos\theta\cos\phi, \cos\theta\sin\phi, \sin\theta)\,, \qquad -\frac{\pi}{2} < \theta < \frac{\pi}{2}, \quad 0 \le \phi \le 2\pi \tag{20}$$

The angle θ is the angle that $\mathbf{u}$ makes with the horizon. For $\theta < 0$, the vector $\mathbf{u}$ is directed downward, and for $\theta > 0$, the vector $\mathbf{u}$ is directed upward.

The case $\theta = 0$ corresponds to $\mathbf{u}$ being parallel to the xy-plane. In this case,

$$\mathbf{u} = \left(\frac{-v_y}{\sqrt{v_x^2+v_y^2}}, \frac{v_x}{\sqrt{v_x^2+v_y^2}}, 0\right)$$

Consider now the cases of $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$ with $\theta \neq 0$. We want to choose $\mathbf{u}$ by showing how to find $\cos\phi$ and $\sin\phi$. Note that $\mathbf{u} \cdot \mathbf{v} = 0$, or

$$u_x v_x + u_y v_y + u_z v_z = 0$$

Then

$$\begin{gathered} v_x \cos\theta \cos\phi + v_y \cos\theta \sin\phi + v_z \sin\theta = 0 \\ v_x \cos\phi + v_y \sin\phi = -v_z \tan\theta \end{gathered} \tag{21}$$

Introduce

$$(\cos\psi, \sin\psi) = \left(\frac{v_x}{\sqrt{v_x^2 + v_y^2}}, \frac{v_y}{\sqrt{v_x^2 + v_y^2}} \right)$$

which defines ψ uniquely in $[0, 2\pi)$. Then (21) becomes

$$(\cos\psi, \sin\psi) \cdot (\cos\phi, \sin\phi) = \frac{-v_z \tan\theta}{\sqrt{v_x^2 + v_y^2}}$$

$$\cos(\psi - \phi) = \frac{-v_z \tan\theta}{\sqrt{v_x^2 + v_y^2}}$$

This yields

$$\begin{aligned} \phi - \psi &= \pm \arccos\left(\frac{-v_z \tan\theta}{\sqrt{v_x^2 + v_y^2}} \right) \\ \phi &= \psi \pm \arccos\left(\frac{-v_z \tan\theta}{\sqrt{v_x^2 + v_y^2}} \right) \end{aligned} \tag{22}$$

Which sign and which value of ϕ should be chosen? Choose one and then check if the resulting vector (u_x, u_y) is on the correct side of (v_x, v_y). If

$$(u_x, u_y) \cdot (-v_y, v_x) > 0$$

then we have the correct side. Otherwise choose the other possible value for ϕ.

5.2 Finding the nearest point

Given a point (α, β, γ) near to the centerline, we find a nearest point

$$(\xi_{n,m}(s^*), \eta_{n,m}(s^*), \zeta_{n,m}(s^*))$$

by minimizing

$$D(s) = (\alpha - \xi(s))^2 + (\beta - \eta(s))^2 + (\gamma - \zeta(s))^2, \qquad 0 \leq s \leq \widehat{L}_{sp} \tag{23}$$

We use method 3 from §3, implemented in *Matlab* as *fminbnd*. After finding s^*, we find the offset ω and the loft λ using

$$\begin{aligned} \omega &= \mathbf{u}\cdot\left[(\alpha,\beta,\gamma) - (\xi_{n,m}(s^*),\eta_{n,m}(s^*),\zeta_{n,m}(s^*))\right] \\ \lambda &= \mathbf{n}\cdot\left[(\alpha,\beta,\gamma) - (\xi_{n,m}(s^*),\eta_{n,m}(s^*),\zeta_{n,m}(s^*))\right] \end{aligned} \tag{24}$$

To test the method, we proceed similarly to the test procedure of §3 as illustrated in Figure 5. We generated random values of s^*, ω, λ within specified limits, and using these we generated the corresponding values of (α, β, γ). We then used *fminbnd* to find s^*, and we used (24) to reconstruct ω and λ. We then compared the recalculated values of (s^*, ω, λ) to the original values to ensure the desired accuracy was being obtained. For an actual numerical example, consider the curve illustrated in Figure 6, but with $m = 100$. We generated 1000 random values, and we used a desired error tolerance of $\varepsilon = 10^{-5}$ for the minimization. The maximum sizes allowed for ω and λ were 0.5 and 0.05, respectively. The range of function evaluations for the minimization in *fminbnd* was $[7, 16]$, with 9.1 as the average number of such evaluations.

6 Effect of curvature on meshsize

Consider the approximation $(\xi_{n,m}(s), \eta_{n,m}(s))$ of the curve Γ as developed in §2 The error in the final approximation depends on both n and m. We assume that n is chosen so large that the error due to n is insignificant compared to that due to m, and therefore we ignore the effects due to n. The curvature $\kappa(s)$ is given by (13), and the radius of curvature is $1/\kappa(s)$.

Let Γ_r be the circle of radius r about the origin, and denote the approximation to Γ_r by $(\xi_{m,r}(s), \eta_{m,r}(s))$. A circle of radius r has a radius of curvature of r. We analyze the accuracy of $(\xi_{m,r}(s), \eta_{m,r}(s))$ as a function first of m and r, and then as a function of the meshsize

$$h_r = \frac{2\pi r}{m},$$

used in constructing $(\xi_{m,r}(s), \eta_{m,r}(s))$.

Let $E(r, m)$ denote the maximum error when using m subdivisions of Γ_r:

$$E(r,m) = \max_{0\le s\le 2\pi r} \left| r\left(\cos\frac{s}{r}, \sin\frac{s}{r}\right) - (\xi_{m,r}(s), \eta_{m,r}(s))\right|$$

It is not difficult to see that

$$(\xi_{m,r}(rs), \eta_{m,r}(rs)) = r\left(\xi_{m,1}(s), \eta_{m,1}(s)\right), \qquad 0 \le s \le 2\pi$$

Thus

$$E(r,m) = rE(1,m) \tag{25}$$

m	$\varepsilon \equiv E(1,m)$	*Ratio*	h_1
5	$1.0494E-2$		1.257
10	$5.4932E-4$	19.10	0.6283
20	$3.2752E-5$	16.77	0.3142
40	$2.0224E-6$	16.19	0.1571
80	$1.2602E-7$	16.05	0.0785

Table 2: Errors in approximation of the unit circle

Suppose we calculate numerically the true error $E(1,m)$ for a particular value of m, say m_1: $\varepsilon = E(1,m_1)$. Based on results in [4], assume

$$E(r,m) = \frac{c_r}{m^4} \tag{26}$$

This is true only asymptotically, but is essentially correct for practical purposes. Then solving

$$\frac{c_1}{m_1^4} = \varepsilon$$

yields

$$c_1 = \varepsilon m_1^4 \tag{27}$$

Now choose m so that $E(r,m) = \varepsilon$. Combining (25) and (26) results in

$$c_r = rc_1 \tag{28}$$

Using (26)-(28) with $E(r,m) = \varepsilon$,

$$\begin{aligned} m &= \sqrt[4]{\frac{c_r}{\varepsilon}} = \sqrt[4]{\frac{rc_1}{\varepsilon}} = \sqrt[4]{\frac{r\varepsilon m_1^4}{\varepsilon}} = \sqrt[4]{rm_1^4} \\ &= \sqrt[4]{r}m_1 \equiv m_r \end{aligned} \tag{29}$$

We are interested in the meshsize associated with the grid on the circle of radius r. It is defined by

$$\begin{aligned} h_r &= \frac{2\pi r}{m_r} = \frac{2\pi r}{\sqrt[4]{r}m_1} \\ &= h_1\sqrt[4]{r^3} \end{aligned} \tag{30}$$

We use this last result when dealing with a general curve Γ with varying curvature, using it to give a "rule of thumb" for choosing a stepsize in the approximation of Γ. When the radius of curvature is r, we should use a local meshsize of h_r in order to preserve an accuracy of ε in the approximation of $(\xi(s),\eta(s))$.

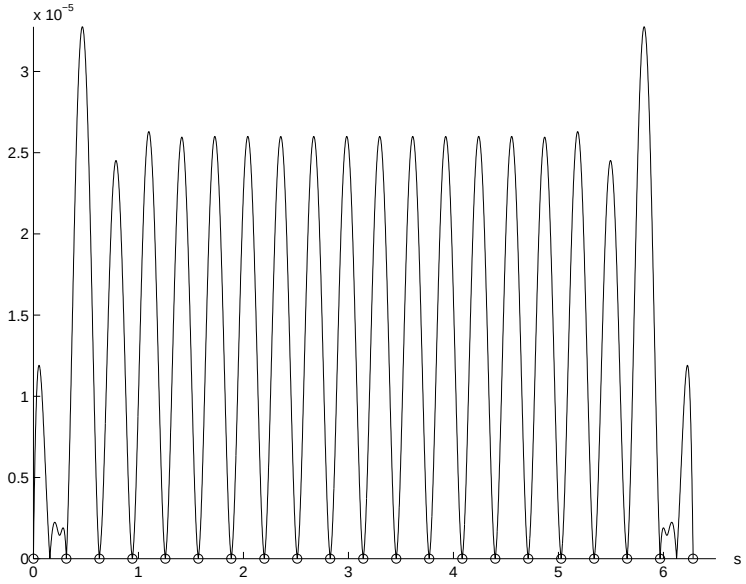

Figure 8: The error (31) when approximating the unit circle with $m = 20$.

Figure 8 gives the error

$$|(\cos s, \sin s) - (\xi_{m,1}(s), \eta_{m,1}(s))|\,, \qquad 0 \le s \le 2\pi \tag{31}$$

for $m = 20$ and $\Gamma = \Gamma_1$, the unit circle. The points marked by 'o' give the evenly spaced node points on the interval $[0, 2\pi]$, with $h \equiv h_1 = 2\pi/m$. Note that the error is also zero at the two points $h/2$ and $2\pi - h/2$, reflecting the use of the not-a-knot interpolating boundary conditions at these points, as suggested in and following (7). In Table 2 we give the values of ε and h_1 for varying values of m. The ratios in the table confirm empirically that (26) is true asymptotically as m increases.

References

[1] K. Atkinson, *An Introduction to Numerical Analysis*, John Wiley, New York, 1989.

[2] K. Atkinson and E. Venturino, Numerical evaluation of line integrals, *SIAM J. Numerical Analysis* **30** (1993), pp. 882-888.

[3] R .Brent, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, 1972.

[4] C. De Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[5] D. Kahaner, C. Moler, and S. Nash, *Numerical Methods and Software*, Prentice-Hall, 1989.

[6] D. Luenberger, *Linear and Nonlinear Programming*, 2^{nd} ed., Addison-Wesley, 1984.