Australian Government
Department of Defence
Defence Science and
Technology Organisation

Constant Speed Interpolating Paths

Jason R. Looker

Air Operations Division

Defence Science and Technology Organisation

DSTO–TN–0989

ABSTRACT

Constant speed paths have been used as interpolating functions for highway and railway design, motion planning for robots, and path planning for military aircraft. A simple algorithm is developed in this note to compute a two-dimensional path that interpolates between ordered data points, consisting of Cornu-spiral, straight-line and circular sub-paths. The interpolation algorithm determines the sub-path parameters such that the interpolating path has constant speed, continuous heading and bounded curvature, while endeavouring to sequentially minimise the sub-path arc lengths.

The most significant factor to influence the performance of the interpolation algorithm is the number of data points that require the interpolating path to execute a hard turn. If no hard turns are required, then the interpolation algorithm quickly returns an interpolating path with minimal arc length (in a heuristic sense). However, hard turns can substantially increase both the CPU time of the algorithm and arc length of the path.

Published by

DSTO Defence Science and Technology Organisation
506 Lorimer St,
Fishermans Bend, Victoria 3207, Australia

Telephone: (03) 9626 7000
Facsimile: (03) 9626 7999

AR No. AR 014–939
March, 2011

Constant Speed Interpolating Paths
Executive Summary

Constant speed paths have been used as interpolating functions for highway and railway design, motion planning for robots, and path planning for unmanned aerial vehicles. This note emerged from a study of path planning for military aircraft conducting missions in hostile environments, where the interpolating path represents the trajectory of the aircraft as it flies through waypoints.

A simple algorithm is developed to compute a two-dimensional path that interpolates between ordered data points. The heading of each sub-path is represented by a quadratic polynomial, resulting in Cornu-spiral, straight-line and circular sub-paths. The interpolation algorithm determines the sub-path parameters such that the interpolating path has constant speed, continuous heading and bounded curvature, while endeavouring to sequentially minimise the sub-path arc lengths.

Stoer[1] used Lagrange multipliers and Newton's method to calculate an interpolating path with constant speed, continuous heading and curvature, and minimal total arc length. This incurs a significant computational cost. A simpler approach is taken in this note, where the curvature of the interpolating path is permitted to have a finite jump discontinuity at the data points, and the arc length of each sub-path is attempted to be minimised in sequence.

The interpolation algorithm developed in this note begins by returning the straight-line sub-path between the first two data points. Then for each of the remaining data points:

1. Update the initial heading.
2. If the initial heading and the heading of the straight-line sub-path are equal, then return the straight-line sub-path.
3. If the straight-line sub-path was not returned, then:
   (i) Calculate the circular sub-path.
   (ii) Attempt to minimise the arc length of the Cornu-spiral sub-path.
   (iii) If a Cornu-spiral sub-path was not able to be calculated or the arc length of the circular sub-path is less than the arc length of the Cornu-spiral sub-path, then return the circular sub-path, otherwise return the Cornu-spiral sub-path.
4. Return to the first step and repeat until the interpolating path is complete.

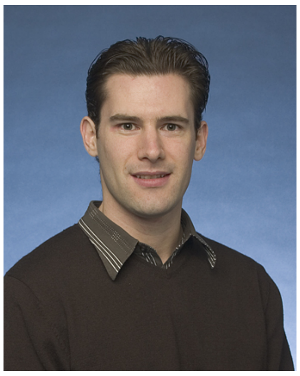
The most significant factor to influence the performance of this interpolation algorithm is the number of data points that require the interpolating path to execute a hard turn. If no hard turns are required, then the interpolation algorithm quickly returns an interpolating path with minimal arc length (in a heuristic sense). However, hard turns can substantially increase both the CPU time of the algorithm and arc length of the path.

Although this note emerged from a study of path planning for military aircraft, the interpolation algorithm is generic and potentially has a broad range of applications. The utility of the interpolation algorithm will be determined by the context in which the data is generated and the intended use of the interpolating path.

[1]Stoer, J. (1982) Curve fitting with clothoidal splines, *Journal of Research of the National Bureau of Standards* **87**(4), 317–346.

THIS PAGE IS INTENTIONALLY BLANK

Author



Jason Looker
Air Operations Division

Dr Jason Looker joined DSTO in 2006 as an Operations Research Scientist after completing a BSc (Honours) and PhD in Mathematics at The University of Melbourne. He has undertaken operations analysis in support of Air Force Headquarters and AIR 9000 Phase 8 (Future Naval Aviation Combat System), and is leading a research project on the path planning of low-observable aircraft through hostile environments.

THIS PAGE IS INTENTIONALLY BLANK

Contents

Figures

Tables

Notation

$\mathbf{x}$	a generic 2-d vector, $\mathbf{x} = (x_1, x_2)$
$\lvert\mathbf{x}\rvert$	magnitude of $\mathbf{x}$, $\lvert\mathbf{x}\rvert = \sqrt{x_1^2 + x_2^2}$
$\mathbf{x} \cdot \mathbf{y}$	scalar (dot) product, $\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2$
$\dot{f}(z)$	$\frac{df}{dz}$, the derivative of a function f with respect to z
$\mathbf{p}$	interpolating path
∞	infinity
$\mathbb{R}$	real numbers
N	number of data points
$\mathbf{x}_i$	$(x_i,\, y_i)$, the ith data point
s	arc length as a parameter
θ	heading of $\mathbf{p}$
$\mathbf{p}_i$	sub-path of $\mathbf{p}$ between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$
ℓ_i	arc length of $\mathbf{p}_i$
ℓ	arc length of $\mathbf{p}$
θ_i	heading of $\mathbf{p}_i$, $\theta_i(s) = \theta_0^i + a_i s + b_i s^2$
θ_0^i	initial heading of $\mathbf{p}_i$
a_i	initial curvature of $\mathbf{p}_i$
b_i	controls the rate at which the curvature of $\mathbf{p}_i$ changes
$C_i(s)$	first component of the Cornu-spiral representation of $\mathbf{p}_i$
$S_i(s)$	second component of the Cornu-spiral representation of $\mathbf{p}_i$
$F_C(s)$	cosine Fresnel Integral
$F_S(s)$	sine Fresnel Integral
$C_i^+(s)$, $C_i^-(s)$	value of $C_i(s)$ when $b_i > 0$ and $b_i < 0$, respectively
$S_i^+(s)$, $S_i^-(s)$	value of $S_i(s)$ when $b_i > 0$ and $b_i < 0$, respectively
$\Delta\mathbf{x}_i$	$\mathbf{x}_{i+1} - \mathbf{x}_i$
$\mathbf{x}_i^c$	centre of the ith circular sub-path
φ_i	angle between $\mathbf{x}_{i+1} - \mathbf{x}_i^c$ and $\mathbf{x}_i - \mathbf{x}_i^c$
κ_i	curvature of $\mathbf{p}_i$
$(\bar{a}_i,\, \bar{\ell}_i)$	initial guess for a solution (a_i, ℓ_i) to $(C_i(\ell_i), S_i(\ell_i)) = \mathbf{x}_{i+1} - \mathbf{x}_i$
$\mathcal{N}(0, \sigma_a)$	normal probability density function with zero mean and standard deviation σ_a
ϵ_a	$\lvert\bar{a}_i\rvert > \epsilon_a$ where $\epsilon_a \ll 1$
$\emptyset$	empty set

1 Introduction

In a typical interpolation problem, a finite set of discrete data points is given and a continuous interpolating function that passes through each of the data points must be determined [Atkinson 1989]. The distinguishing features of a good interpolating function depend on the context in which the data was generated and the intended use of the interpolating function. Constant speed paths have been used as interpolating functions for highway and railway design [Meek & Walton 1989, Meek & Walton 1992], motion planning for robots [Kelly & Nagy 2003], and path planning for unmanned aerial vehicles [Dai & Cochran Jr. 2010]. This note emerged from a study of path planning for military aircraft conducting missions in hostile environments, where the interpolating path represents the trajectory of the aircraft as it flies through waypoints.

The aim of this note is to develop a simple algorithm to compute a path $\mathbf{p}\colon[0,\infty)\to\mathbb{R}^2$ that interpolates between N ordered data points $\{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^2$. This interpolating path must satisfy the following properties:

1. **Interpolation**; the path passes through all the data points.
2. **Constant speed**; the path has a given constant speed at each point on the path.
3. **Continuous heading**; the heading of the path is continuous at each point on the path.[1]
4. **Bounded curvature**; the curvature of the path is bounded at each point on the path.

It is also desirable for the path have minimal length, subject to the constraints imposed by Properties 1 to 4.

Let $\mathbf{p}$ be parametrised by arc length $s \in [0,\infty)$. Then Property 2 implies that $\mathbf{p}$ has the form

$$\mathbf{p}(s) = \mathbf{x}_1 + \int_0^s (\cos\theta(z), \sin\theta(z))\, dz, \tag{1}$$

where $\theta(s)$ is the heading of the path when its arc length is s and $|\dot{\mathbf{p}}| = 1$, that is, $\mathbf{p}$ has unit speed.[2] It remains to choose $\theta(s)$ such that Properties 1 to 4 are satisfied. If the heading is represented by a polynomial in s, then the resulting paths are known as Cornu spirals (see Figure 1), which have been employed as interpolating paths with minimal length [Stoer 1982] and satisfying a least-squares criteria [Davis 1999], for highway and railway design [Meek & Walton 1989, Meek & Walton 1992], motion planning for robots [Kelly & Nagy 2003], and path planning for unmanned aerial vehicles [Dai & Cochran Jr. 2010].

The algorithm developed in this note to compute $\mathbf{p}$ is based on the work of Stoer [1982]. The interpolating path computed by Stoer has continuous curvature at each point on the path and the integral of the curvature squared over the length of the entire path

[1] For a path with continuous heading, infinitesimal changes in position on the path result in infinitesimal changes in heading.

[2] Any constant speed path can be re-parametrised by arc length to yield a unit speed path.

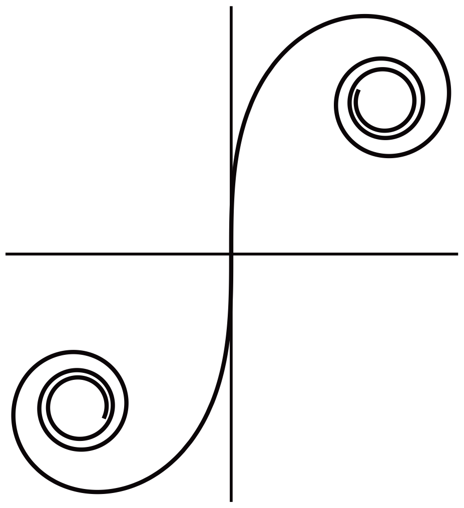


Figure 1: *An example of a Cornu spiral.*

is minimised, which is equivalent to minimal total arc length. This incurs a significant computational cost. Here the emphasis is on a simple algorithm to compute $\mathbf{p}$, and as a result these two properties are relaxed:

- The curvature of $\mathbf{p}$ is permitted to have a finite jump discontinuity at the data points.
- The integral of the curvature squared over the length of $\mathbf{p}$ between each of the data points is attempted to be minimised in sequence.

These properties are consistent with Properties 1 to 4. Note that $\mathbf{p}$ is not guaranteed to have minimal total arc length.

In Section 2 a representation of the interpolating path in terms of Cornu-spiral, straight-line and circular sub-paths is stated and discussed. The interpolation algorithm is developed in Section 3, including a derivation of the sub-path arc length minimisation heuristic that is central to the algorithm. In Section 4 the interpolation algorithm is specified using pseudo code and its performance with respect to CPU time and total arc length is illustrated.

2 Representation of the Interpolating Path

Let $\mathbf{p}_i : [0, \ell_i] \to \mathbb{R}^2$ be the sub-path of $\mathbf{p}$ between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ with arc length ℓ_i for $i = 1, \ldots, N-1$. Denote the total arc length of $\mathbf{p}$ by ℓ, then

$$\ell = \sum_{i=1}^{N-1} \ell_i.$$

Stoer [1982] represents the heading of each sub-path $\theta_i : [0, \ell_i] \to \mathbb{R}$ by a quadratic polynomial,

$$\theta_i(s) = \theta_0^i + a_i s + b_i s^2, \tag{2}$$

for $i = 1, \ldots, N-1$, resulting in sub-paths that are Cornu spirals (see Figure 1). The sub-path parameters θ_0^i, a_i and b_i determine the shape of the path: θ_0^i is the initial heading, a_i is the initial curvature, and b_i controls the rate at which the curvature changes as s increases. Special cases include straight-line ($a_i = b_i = 0$) and circular ($b_i = 0$) sub-paths. Unlike straight-line and circular paths, Cornu-spiral paths can perform one or two turns. Although Properties 1 to 4 can be satisfied using only straight-line and circular sub-paths, this will often generate interpolating paths with excessively long arc lengths.

A significant advantage of this representation of the heading is that an analytical form of Equation (1) is available in terms of Fresnel Integrals [Stoer 1982]:

$$\mathbf{p}_i(s) = \mathbf{x}_i + (C_i(s), S_i(s))\,, \tag{3}$$

for $i = 1, \ldots, N-1$, where the functions $C_i(s)$ and $S_i(s)$ are given by

$$C_i(s) = \sqrt{\frac{\pi}{2b_i}}\left[\cos\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_C\left(\frac{a_i + 2b_i s}{\sqrt{2\pi b_i}}\right) - F_C\left(\frac{a_i}{\sqrt{2\pi b_i}}\right)\right)\right.$$
$$\left. + \sin\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_S\left(\frac{a_i + 2b_i s}{\sqrt{2\pi b_i}}\right) - F_S\left(\frac{a_i}{\sqrt{2\pi b_i}}\right)\right)\right],$$

$$S_i(s) = \sqrt{\frac{\pi}{2b_i}}\left[\cos\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_S\left(\frac{a_i + 2b_i s}{\sqrt{2\pi b_i}}\right) - F_S\left(\frac{a_i}{\sqrt{2\pi b_i}}\right)\right)\right.$$
$$\left. - \sin\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_C\left(\frac{a_i + 2b_i s}{\sqrt{2\pi b_i}}\right) - F_C\left(\frac{a_i}{\sqrt{2\pi b_i}}\right)\right)\right],$$

and $F_C(z)$ and $F_S(z)$ are the Fresnel Integrals[3] [Abramowitz & Stegun 1972]

$$F_C(z) = \int_0^z \cos\left(\frac{\pi}{2}t^2\right) dt,$$

$$F_S(z) = \int_0^z \sin\left(\frac{\pi}{2}t^2\right) dt.$$

It can be shown that $C_i(s)$ and $S_i(s)$ are bounded and real for all parameter values, including $b_i \leqslant 0$. However, this relies on the symmetry relations of the Fresnel Integrals to eliminate complex values, and the limiting behaviour of $C_i(s)$ and $S_i(s)$ as $b_i \to 0$ is highly oscillatory. Therefore numerical computations involving $C_i(s)$ and $S_i(s)$ can be problematic. To overcome these numerical difficulties, $C_i(s)$ and $S_i(s)$ are given the following equivalent definitions:[4]

$$C_i(s) = \begin{cases} C_i^+(s)\,, & b_i > 0 \\ C_i^-(s)\,, & b_i < 0, \end{cases} \tag{4}$$

[3]An example of code to compute the Fresnel Integrals can be found at MATLAB Central: `http://www.mathworks.com/matlabcentral/fileexchange/6580-fresnel-integrals` (accessed December 2010).

[4]These definitions were obtained using the symmetry relations of the Fresnel Integrals [Abramowitz & Stegun 1972].

where

$$C_i^+(s) = \sqrt{\frac{\pi}{2b_i}}\left[\cos\!\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_C\!\left(\frac{a_i + 2b_i s}{\sqrt{2\pi b_i}}\right) - F_C\!\left(\frac{a_i}{\sqrt{2\pi b_i}}\right)\right)\right.$$
$$\left. + \sin\!\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_S\!\left(\frac{a_i + 2b_i s}{\sqrt{2\pi b_i}}\right) - F_S\!\left(\frac{a_i}{\sqrt{2\pi b_i}}\right)\right)\right],$$

$$C_i^-(s) = -\sqrt{\frac{\pi}{2\,|b_i|}}\left[\cos\!\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_C\!\left(\frac{a_i + 2b_i s}{\sqrt{2\pi\,|b_i|}}\right) - F_C\!\left(\frac{a_i}{\sqrt{2\pi\,|b_i|}}\right)\right)\right.$$
$$\left. - \sin\!\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_S\!\left(\frac{a_i + 2b_i s}{\sqrt{2\pi\,|b_i|}}\right) - F_S\!\left(\frac{a_i}{\sqrt{2\pi\,|b_i|}}\right)\right)\right],$$

and

$$S_i(s) = \begin{cases} S_i^+(s)\,, & b_i > 0 \\ S_i^-(s)\,, & b_i < 0, \end{cases} \tag{5}$$

where

$$S_i^+(s) = \sqrt{\frac{\pi}{2b_i}}\left[\cos\!\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_S\!\left(\frac{a_i + 2b_i s}{\sqrt{2\pi b_i}}\right) - F_S\!\left(\frac{a_i}{\sqrt{2\pi b_i}}\right)\right)\right.$$
$$\left. - \sin\!\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_C\!\left(\frac{a_i + 2b_i s}{\sqrt{2\pi b_i}}\right) - F_C\!\left(\frac{a_i}{\sqrt{2\pi b_i}}\right)\right)\right],$$

$$S_i^-(s) = \sqrt{\frac{\pi}{2\,|b_i|}}\left[\cos\!\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_S\!\left(\frac{a_i + 2b_i s}{\sqrt{2\pi\,|b_i|}}\right) - F_S\!\left(\frac{a_i}{\sqrt{2\pi\,|b_i|}}\right)\right)\right.$$
$$\left. + \sin\!\left(\frac{a_i^2}{4b_i} - \theta_0^i\right)\left(F_C\!\left(\frac{a_i + 2b_i s}{\sqrt{2\pi\,|b_i|}}\right) - F_C\!\left(\frac{a_i}{\sqrt{2\pi\,|b_i|}}\right)\right)\right].$$

Straight-line ($a_i = b_i = 0$) and circular ($b_i = 0$) sub-paths are considered separately due to the singular behaviour of $C_i(s)$ and $S_i(s)$ as $b_i \to 0$, and the availability of explicit analytic formulae for these paths. Straight-line sub-paths have the form

$$\mathbf{p}_i(s) = \mathbf{x}_i + \frac{s}{\ell_i}(\mathbf{x}_{i+1} - \mathbf{x}_i), \tag{6}$$

$$\ell_i = |\mathbf{x}_{i+1} - \mathbf{x}_i|\,, \tag{7}$$

for $s \in [0, \ell_i]$. Circular sub-paths are defined by

$$\mathbf{p}_i(s) = \mathbf{x}_i + \frac{1}{a_i}\big(\sin\!\big(\theta_0^i + a_i s\big) - \sin\theta_0^i,\, \cos\theta_0^i - \cos\!\big(\theta_0^i + a_i s\big)\big)\,, \tag{8}$$

for $s \in [0, \ell_i]$ where

$$a_i = -2\frac{\Delta \mathbf{x}_i \cdot \left(\sin\theta_0^i,\, -\cos\theta_0^i\right)}{|\Delta \mathbf{x}_i|^2}, \tag{9}$$

$$\Delta \mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{x}_i, \tag{10}$$

$$\ell_i = \frac{\varphi_i}{|a_i|}, \tag{11}$$

$$\varphi_i = \begin{cases} \arccos\left(a_i^2(\mathbf{x}_i - \mathbf{x}_i^c) \cdot (\mathbf{x}_{i+1} - \mathbf{x}_i^c)\right), & \Delta \mathbf{x}_i \cdot \dot{\mathbf{p}}_i(0) > 0 \\ 2\pi - \arccos\left(a_i^2(\mathbf{x}_i - \mathbf{x}_i^c) \cdot (\mathbf{x}_{i+1} - \mathbf{x}_i^c)\right), & \Delta \mathbf{x}_i \cdot \dot{\mathbf{p}}_i(0) \leqslant 0, \end{cases} \tag{12}$$

$$\mathbf{x}_i^c = \mathbf{x}_i - \frac{1}{a_i}\left(\sin\theta_0^i,\, -\cos\theta_0^i\right), \tag{13}$$

$$\dot{\mathbf{p}}_i(0) = \left(\cos\theta_0^i,\, \sin\theta_0^i\right). \tag{14}$$

Note that $\mathbf{p}_i(\ell_i) = \mathbf{x}_{i+1}$ by construction.

3 Algorithm Development

The purpose of the interpolation algorithm is to determine the sub-path parameters $(\theta_0^i, a_i, b_i, \ell_i)$ such that Properties 1 to 4 (Interpolation, Constant Speed, Continuous Heading, and Bounded Curvature) are satisfied, while endeavouring to sequentially minimise the sub-path arc length ℓ_i.

3.1 Strategy

The sub-path parameters $(\theta_0^i, a_i, b_i, \ell_i)$ must be determined for each of the $N-1$ sub-paths $\mathbf{p}_i$ to completely define the interpolating path for a given set of ordered points $\{\mathbf{x}_i\}_{i=1}^N$. Stoer [1982] used Lagrange multipliers and Newton's method to find $(\theta_0^i, a_i, b_i, \ell_i)$ such that the curvature of the interpolating path is continuous at the data points, and the total arc length of the path is minimal. This incurs a significant computational cost.

A simpler approach is taken in this note, where the arc length of each sub-path is attempted to be minimised in sequence. If continuity of the curvature at the data points is also required, as in Stoer [1982], then substantial oscillations in the path can be introduced. This can be seen by the following argument. The shortest distance between two points is a straight line, implying that $a_1 = b_1 = 0$. Continuity of the curvature at $\mathbf{x}_2$ then yields $a_2 = 0$.[5] It follows that the heading of $\mathbf{p}_2$ can only change in one direction that is determined by the sign of b_2, which may require $\mathbf{p}_2$ to form a spiral to ensure that

[5]The curvature at the data points will be continuous if $a_i = a_{i-1} + 2b_{i-1}\ell_{i-1}$ for $i = 2, \ldots, N-1$, which follows from the derivative of Equation (2) with respect to arc length.

$\mathbf{p}_2(\ell_2) = \mathbf{x}_3$. Therefore, allowing a finite jump discontinuity in the curvature at the data points will ameliorate the formation of spirals.[6]

This strategy will typically result in an asymmetric interpolating path, that is, if the order of the data points is reversed then a different path will be generated. This is a consequence of sequentially minimising the sub-path arc lengths.

3.2 Satisfying the Interpolating Path Properties

Interpolation

The Interpolation property implies that

$$\mathbf{p}_i(0) = \mathbf{x}_i, \tag{15}$$

$$\mathbf{p}_i(\ell_i) = \mathbf{x}_{i+1}, \tag{16}$$

for $i = 1, \ldots, N-1$. The functional form of $\mathbf{p}_i$ was chosen to satisfy Equation (15) and hence this equation does not yield any additional information regarding the sub-path parameters. Observe that Equation (16) provides two independent equations for the sub-path parameters.[7]

Constant Speed

The sub-paths satisfy the Constant Speed property by construction and hence this property also does not yield any additional information regarding the sub-path parameters.

Continuous Heading

Since the shortest distance between two points is a straight line, the first sub-path is simply a straight-line path between $\mathbf{x}_1$ and $\mathbf{x}_2$, and hence

$$\theta_0^1 = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right), \tag{17}$$

where $\mathbf{x}_i = (x_i, y_i)$. For $i = 2, \ldots, N-1$, the Continuous Heading property implies that

$$\theta_0^i = \theta_0^{i-1} + a_{i-1}\ell_{i-1} + b_{i-1}\ell_{i-1}^2, \tag{18}$$

which follows from Equation (2).

[6]Reducing the number of constraints in any optimisation problem will lead to an improved optimal value for the objective function.

[7]Explicit solutions to Equation (16) are provided by Equation (7) for straight-line paths and Equations (9) to (14) for circular paths.

Bounded Curvature

The (signed) curvature of $\mathbf{p}_i$ is denoted by $\kappa_i : [0, \ell_i] \to \mathbb{R}$ and

$$\kappa_i(s) = \dot{\theta}_i(s) = a_i + 2b_i s.$$

Therefore

$$\max_{s\in[0,\ell_i]} |\kappa_i(s)| = \max\left\{|a_i|, |a_i + 2b_i\ell_i|\right\},$$

and so κ_i is bounded for finite values of a_i, b_i and ℓ_i.

3.3 Arc Length Minimisation

In this section, a method for calculating the sub-path parameters $(\theta_0^i, a_i, b_i, \ell_i)$ of Cornu-spiral sub-paths is discussed. Explicit formulae for the sub-path parameters of straight-line and circular paths are given by Equations (6) to (14) and Equations (17) and (18).

Strategy

Equation (16) provides two implicit equations to define $(\theta_0^i, a_i, b_i, \ell_i)$ and θ_0^i is given explicitly by Equations (17) and (18). To completely determine each $\mathbf{p}_i$, another equation or condition is required to specify one of a_i, b_i or ℓ_i. Requiring the curvature to be continuous at the data points will not be successful in the context of sequential minimisation of the sub-path arc lengths. Furthermore, $\mathbf{p}_i$ is singular as $b_i \to 0$ and hence varying b_i to solve Equation (16) may be problematic.[8] As a result it is advantageous to fix b_i then employ Equation (16) to specify a_i and ℓ_i.

For each b_i there may be multiple a_i and ℓ_i that solve Equation (16); the solution with minimal ℓ_i is sought. Equation (16) is solved using a root-finding algorithm such as Newton's method, which begins with an initial guess for the solution. There are three factors that can influence the convergence of the root-finding algorithm to a solution of Equation (16) with minimal ℓ_i:

- proximity of the initial guess for a_i to a solution that corresponds to minimal ℓ_i,
- proximity of the initial guess for ℓ_i to a solution that corresponds to minimal ℓ_i, and
- a condition for b_i that favours solutions with minimal ℓ_i.

A Condition for b_i

A condition for b_i is obtained by seeking to minimise

$$\int_0^{\ell_i} \kappa_i^2(s)\,ds = \frac{4}{3}\ell_i\left[\left(b_i\ell_i + \frac{3}{4}a_i\right)^2 + \frac{3}{16}a_i^2\right],$$

[8] If $a_i = 0$ then the interpolating path is not singular as $b_i \to 0$.

which is equivalent to minimising ℓ_i.[9] Since $\int_0^{\ell_i} \kappa_i^2(s)\,ds$ is a sum of two positive terms, this suggests that

$$b_i = -\frac{3}{4}\frac{a_i}{\ell_i}, \tag{19}$$

will result in solutions of Equation (16) with minimal ℓ_i.[10]

Initial Guesses for a_i and ℓ_i

Let $\bar{a}_i$ and $\bar{\ell}_i$ be initial guesses for a solution (a_i, ℓ_i) to Equation (16). A lower bound for ℓ_i is the arc length of the straight-line path between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, for which $a_i = b_i = 0$. This suggests $\bar{\ell}_i = |\mathbf{x}_{i+1} - \mathbf{x}_i|$ and $\bar{a}_i = 0$. However, since $\mathbf{p}_i$ is singular as $b_i \to 0$, Equation (19) implies that $\bar{a}_i \neq 0$. Despite this, it is still necessary for $|\bar{a}_i|$ to be (in some sense) small to guide the root-finding algorithm to a solution of Equation (16) with minimal ℓ_i.

The root-finding algorithm is not guaranteed to converge to a solution of Equation (16) for every choice of $\bar{a}_i$ and $\bar{\ell}_i$, and so $\bar{a}_i$ and/or $\bar{\ell}_i$ must be modified for each attempt to solve Equation (16). Setting $\bar{\ell}_i = |\mathbf{x}_{i+1} - \mathbf{x}_i|$ and varying $\bar{a}_i$ has proven to yield the best results. Since a small value of a_i corresponds to a small value of ℓ_i, $\bar{a}_i$ is selected from a normal probability distribution with zero mean:

$$\bar{a}_i \sim \mathcal{N}(0, \sigma_a) \text{ such that } |\bar{a}_i| > \epsilon_a,$$

where $\mathcal{N}(0, \sigma_a)$ denotes the normal distribution probability density function with zero mean and standard deviation $\sigma_a > 0$ and $\epsilon_a \ll 1$. Both σ_a and ϵ_a are given fixed parameters.

4 The Interpolation Algorithm

In this section, the interpolation algorithm is specified using pseudo code and its performance with respect to CPU time and total arc length is illustrated.

4.1 Description

Cornu-spiral sub-paths

The sequential sub-path arc length minimisation strategy described in Section 3.3 is implemented in Algorithm 1 to determine (a_i, b_i, ℓ_i) of Cornu-spiral sub-paths, for $i = 2, \ldots, N-1$.[11] Note that:

- σ_a, ϵ_a, `lUpperBound` and `MaxAttempts` are independent of the sub-paths, that is, they remain constant for every call to Algorithm 1,

[9] $\int_0^{\ell_i} \kappa_i^2(s)\,ds$ is a monotone increasing function of ℓ_i.

[10] An alternative condition is $b_i = -a_i/\ell_i$, in which case $\int_0^{\ell_i} \kappa_i^2(s)\,ds = a_i^2\ell_i/3$.

[11] The first sub-path is given by Equations (6) and (7) and θ_0^i is given by Equation (18) for $i = 2, \ldots, N-1$.

- `lUpperBound` $\geqslant \ell_i$ for all $i \in \{1, \dots, N-1\}$,
- `MaxAttempts` specifies the maximum number of attempts to solve Equation (16),
- the definitions of $C_i(s)$ and $S_i(s)$ given by Equations (4) and (5) must be used to avoid complex values entering the computations, and
- Newton's method exhibited far superior performance when compared with the Secant method.

Algorithm 1: Determine the sub-path parameters of Cornu spirals.

```
Input: $\theta_0^i$, $\bar{\ell}_i$, $\mathbf{x}_i$, $\mathbf{x}_{i+1}$, for $i = 2, \dots, N-1$, $\sigma_a$, $\epsilon_a$, lUpperBound and MaxAttempts
Output: $(a_i, b_i, \ell_i)$ for $i = 2, \dots, N-1$

RootVector $\leftarrow \emptyset$;
$n \leftarrow 1$;

while RootVector $= \emptyset$ and $n \leqslant$ MaxAttempts do
    $\bar{a}_i \sim \mathcal{N}(0, \sigma_a)$ such that $|\bar{a}_i| > \epsilon_a$;
    $b_i \leftarrow -0.75\, \bar{a}_i / \bar{\ell}_i$;
    Solve $(C_i(\ell_i), S_i(\ell_i)) = \mathbf{x}_{i+1} - \mathbf{x}_i$ for $(a_i, \ell_i)$ using Newton's method with $(\bar{a}_i, \bar{\ell}_i)$
    as an initial guess, such that $\bar{\ell}_i \leqslant \ell_i \leqslant$ lUpperBound;
    if A solution has been found then
        Store the solution in RootVector;
    end
    $n \leftarrow n + 1$;
end

if RootVector $\neq \emptyset$ then
    $(a_i, b_i, \ell_i)$
else
    $\emptyset$
end
```

The Interpolation algorithm

The Interpolation algorithm, which is presented in Algorithm 2, begins by returning the straight-line sub-path between the first two data points. Then for each of the remaining data points:

1. Update the initial heading.
2. If the initial heading and the heading of the straight-line sub-path are equal, then return the straight-line sub-path.
3. If the straight-line sub-path was not returned, then:
   (i) Calculate the circular sub-path.
   (ii) Call Algorithm 1 repeatedly to minimise the arc length of the Cornu-spiral sub-path.
   (iii) If a Cornu-spiral sub-path was not able to be calculated or the arc length of the circular sub-path is less than the arc length of the Cornu-spiral sub-path, then return the circular sub-path, otherwise return the Cornu-spiral sub-path.
4. Return to the first step and repeat until the interpolating path is complete.

Note that:

- the Interpolation algorithm will not return an interpolating path if there exists an $i \in \{1, \ldots, N-1\}$ such that $\mathbf{x}_{i+1} = \mathbf{x}_i$, and may not return an interpolating path if

  $$(\mathbf{x}_{i+1} - \mathbf{x}_i) \cdot (\sin \theta_0^i, -\cos \theta_0^i) = 0,$$

  which is equivalent to a reversal in direction,
- `MaxIterations` specifies the maximum number of calls to Algorithm 1,
- `TerminationFactor` > 1,
- if Algorithm 1 returns (a_i, b_i, ℓ_i) with

  $$\ell_i \leqslant \texttt{TerminationFactor} \times |\mathbf{x}_{i+1} - \mathbf{x}_i|,$$

  then the Interpolation algorithm will stop attempting to minimise ℓ_i, and
- the notation `SolutionVector`$[i]$ is the ith component of `SolutionVector`.

The performance of the Interpolation algorithm is discussed in Section 4.2.

Algorithm 2: The Interpolation algorithm.

```
Input: $\{\mathbf{x}_i\}_{i=1}^N$, MaxIterations and TerminationFactor
Output: $(\theta_0^i, a_i, b_i, \ell_i)$ for $i = 1, \ldots, N-1$
$\theta_0^1 \leftarrow \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right)$; $a_1 \leftarrow 0$; $b_1 \leftarrow 0$; $\ell_1 \leftarrow |\mathbf{x}_2 - \mathbf{x}_1|$;
$(\theta_0^1, a_1, b_1, \ell_1)$
$i \leftarrow 2$;
while $i \leqslant N - 1$ do
    SolutionVector $\leftarrow \emptyset$;
    StraightLineFlag $\leftarrow 0$;
    $\bar{\ell}_i \leftarrow |\mathbf{x}_{i+1} - \mathbf{x}_i|$;
    $\theta_0^i \leftarrow \theta_0^{i-1} + a_{i-1}\ell_{i-1} + b_{i-1}\ell_{i-1}^2$;
    if $(x_{i+1} - x_i)\cos\theta_0^i + (y_{i+1} - y_i)\sin\theta_0^i = \bar{\ell}_i$ then
        StraightLineFlag $\leftarrow 1$;
        $a_i \leftarrow 0$;
        $b_i \leftarrow 0$;
        $\ell_i \leftarrow \bar{\ell}_i$;
    end
    if StraightLineFlag $= 0$ then
        Calculate the circular sub-path's parameters with aCircle given by
        Equation (9) and lCircle given by Equation (11);
        $\ell_i \leftarrow \infty$;
        $j \leftarrow 1$;
        while $j \leqslant$ MaxIterations and $\ell_i >$ TerminationFactor $* \bar{\ell}_i$ do
            Call Algorithm 1 with inputs $\theta_0^i$, $\bar{\ell}_i$, $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, and store the output in
            SolutionVector;
            if SolutionVector $\neq \emptyset$ and $\ell_i >$ SolutionVector[3] then
                $a_i \leftarrow$ SolutionVector[1];
                $b_i \leftarrow$ SolutionVector[2];
                $\ell_i \leftarrow$ SolutionVector[3];
            end
            $j \leftarrow j + 1$;
        end
        if SolutionVector $= \emptyset$ or lCircle $< \ell_i$ then
            $a_i \leftarrow$ aCircle;
            $b_i \leftarrow 0$;
            $\ell_i \leftarrow$ lCircle;
        end
    end
    $(\theta_0^i, a_i, b_i, \ell_i)$
    $i \leftarrow i + 1$;
end
```

Table 1: *Algorithm 1 parameter values used to generate the figures and tables in Section 4.2.*

σ_a	ϵ_a	`lUpperBound`	`MaxAttempts`
25	10^{-12}	1	75

4.2 Performance

The Interpolation algorithm's performance is illustrated in this section, including the impact of the `MaxIterations` and `TerminationFactor` parameters on CPU time and arc length.[12]

The Algorithm 1 parameter values that were used to generate the results in this section are shown in Table 1. The smallest values of σ_a and `MaxAttempts` were chosen such that the reliability of Algorithm 1 was not compromised.

Adding Points and Hard Turns

The most significant factor that influences the CPU time of the Interpolation algorithm is the number of data points that require the interpolating path to execute a hard turn. If the number of data points increases such that no hard turns are required, then the CPU time will increase linearly with the number of points because the computations to calculate the sub-path parameters are independent. However, hard turns can substantially increase the CPU time of the algorithm.

The effect of hard turns on CPU time was ascertained by comparing the baseline case (Case 1) shown in Figure 2 with a variation to the baseline (Case 2), which is shown in Figure 3. It can be seen in Table 2 that the 95% confidence intervals[13] for the average CPU time are contained in $[0.15, 0.65]$ seconds for Case 1, for each value of `MaxIterations`. Case 2 is identical to Case 1 except that $\mathbf{x}_4$ has been altered to force hard turns from $\mathbf{x}_3$ to $\mathbf{x}_4$ and from $\mathbf{x}_4$ to $\mathbf{x}_5$. The impact of these hard turns on CPU time for Case 2 is shown in Table 2, where the 95% confidence intervals for the average CPU time are now contained in $[18, 134]$ seconds, for each value of `MaxIterations`.

[12]The Interpolation algorithm was run using Mathematica 8.0.0.0 (64-bit Kernel) on an Apple iMac running Mac OS X 10.6.5 with a 2.66 GHz Intel Core 2 Duo CPU and 4 GB of RAM.

[13]The 95% confidence intervals were generated by 100 runs of the Interpolation algorithm for each value of `MaxIterations`; a greater number of runs did not qualitatively change the confidence intervals.

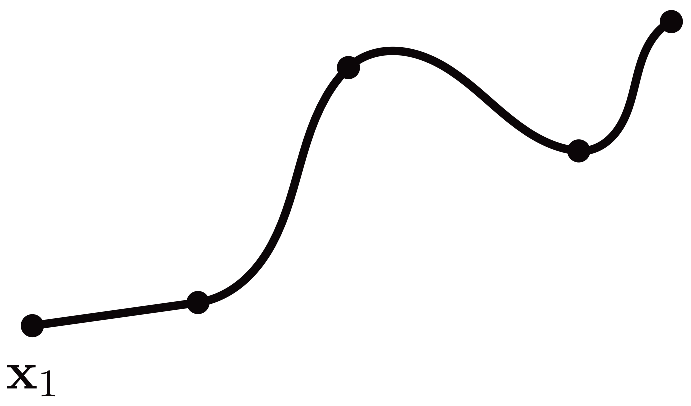


Figure 2: *Case 1, the baseline case, which is given by* $\mathbf{x}_1 = (-0.0022, 0.22)$*,* $\mathbf{x}_2 = (0.14, 0.24)$*,* $\mathbf{x}_3 = (0.27, 0.44)$*,* $\mathbf{x}_4 = (0.47, 0.37)$ *and* $\mathbf{x}_5 = (0.55, 0.48)$*.*

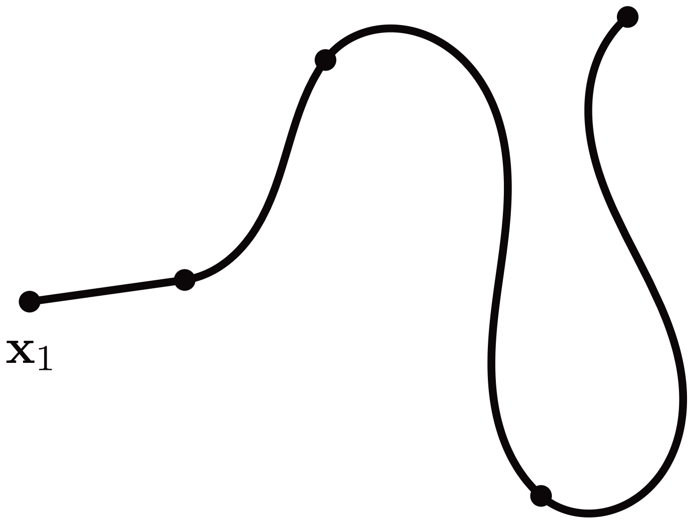


Figure 3: *Case 2. This case was chosen to demonstrate the effect of hard turns on the CPU time of the Interpolation algorithm; the points are identical to Case 1 with the exception of* $\mathbf{x}_4 = (0.47, 0.037)$*.*

Table 2: *95% confidence intervals for the average CPU time and total arc length for different values of* `MaxIterations` *with* `TerminationFactor` $= 1.2$*. Case 1 is shown in Figure 2 and Case 2 is shown in Figure 3.*

Case	`MaxIterations`	CPU Time (sec)	ℓ
	3	$[0.19, 0.48]$	$[0.78, 0.79]$
Case 1	6	$[0.15, 0.47]$	$[0.78, 0.79]$
	18	$[0.23, 0.65]$	$[0.78, 0.79]$
	3	$[18, 24]$	$[1.5, 1.6]$
Case 2	6	$[44, 51]$	$[1.5, 1.6]$
	18	$[118, 134]$	$[1.5, 1.6]$

The impact of hard turns on total arc length was investigated by considering three data points of the form shown in Figure 4; the parameter $\phi \in [0, \pi]$ was incremented by 0.01π. The Interpolation algorithm was run once for each value of ϕ with `TerminationFactor` $=$ 1.2 and `MaxIterations` $= 3$. Between $\mathbf{x}_2$ and $\mathbf{x}_3$ the Interpolation algorithm returned Cornu-spiral sub-paths for $\phi \in [0.01\pi, 0.78\pi]$, where the total arc length increased from 0.50002 to 0.59841. Whereas for $\phi \in [0.79\pi, 0.99\pi]$ the Interpolation algorithm returned circular sub-paths between $\mathbf{x}_2$ and $\mathbf{x}_3$, where the total arc length increased from 1.26 to 25.0. Therefore, if Cornu-spiral sub-paths cannot be computed due to the existence of points that require the interpolating path to execute a hard turn, then the total arc length may increase rapidly. Adding intermediate points to decrease the angle of the turn may reduce this effect.

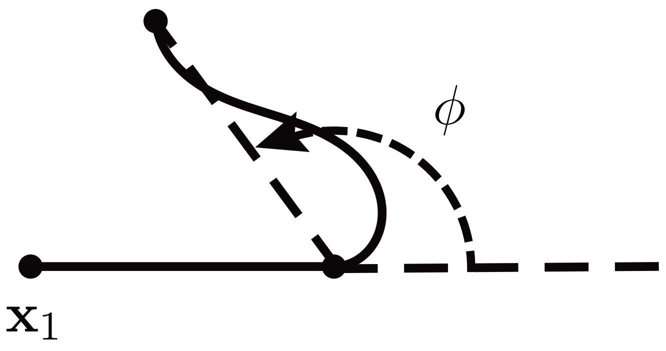


Figure 4: *Definition diagram showing the points used to demonstrate the effect of hard turns on total arc length for different values of* ϕ*:* $\mathbf{x}_1 = (0, 0)$*,* $\mathbf{x}_2 = (0.25, 0)$ *and* $\mathbf{x}_3 = (0.25, 0) + 0.25\,(\cos\phi, \sin\phi)$ *for* $\phi \in [0, \pi]$*.*

Varying `MaxIterations` and `TerminationFactor`

Increasing `MaxIterations` will linearly increase the CPU time of the Interpolation algorithm, if `TerminationFactor` does not terminate the algorithm. If `TerminationFactor` is active, then increasing `MaxIterations` may have little effect on CPU time.

Table 2 shows that increasing `MaxIterations` had a negligible effect on reducing the total arc lengths for Cases 1 and 2. This is not always the case, which was observed when the Secant method was used in Algorithm 1. Furthermore, configurations of points may exist where increasing `MaxIterations` reduces the total arc length. Hence, for completeness, `MaxIterations` was retained in the Interpolation algorithm.[14]

The linear dependence of CPU time on `MaxIterations`, and the considerable reduction in CPU time that may be achieved by increasing `TerminationFactor`, are exhibited in Table 3 where the analysis of Case 2 was repeated with `TerminationFactor` $= 2$. Compared with Table 2, where `TerminationFactor` $= 1.2$, the 95% confidence intervals for the average CPU time have decreased approximately linearly for each value of `MaxIterations`. For example, with `MaxIterations` $= 3$, $\frac{1}{3}[18, 24] = [6.0, 8.0]$. This is sufficiently close to the corresponding interval reported in Table 3 to suggest that increasing `TerminationFactor` caused the Interpolation algorithm to terminate, predominantly, after a single call to Algorithm 1.[15] These substantial reductions in CPU times have been attained without increasing the total arc lengths.

Table 3: 95% *confidence intervals for the average CPU time and total arc length for Case 2, which is shown in Figure 3, for different values of* `MaxIterations` *with* `TerminationFactor` $= 2$.

`MaxIterations`	CPU Time (sec)	ℓ
3	$[7.0, 11]$	$[1.5, 1.6]$
6	$[5.4, 8.6]$	$[1.5, 1.6]$
18	$[8.2, 12]$	$[1.5, 1.6]$

5 Conclusion

A simple algorithm has been developed to compute a two-dimensional path that interpolates between ordered data points. The interpolating path consists of Cornu-spiral, straight-line and circular sub-paths (refer to Section 2). The Interpolation algorithm (Algorithm 2 in Section 4.1) determines the sub-path parameters such that the interpolating path has constant speed, continuous heading and bounded curvature, while endeavouring to sequentially minimise the sub-path arc lengths (refer to Section 3).

[14]Setting `MaxIterations` $= 1$ will prevent the Interpolation algorithm from attempting to further minimise the sub-path arc lengths returned by Algorithm 1.

[15]To disable `TerminationFactor`, set `TerminationFactor` $= 1$.

The performance of the Interpolation algorithm was illustrated in Section 4.2, where it was shown that the most significant factor to influence the CPU time of the algorithm and arc length of the interpolating path is the number of points that require the interpolating path to execute a hard turn. This is because Algorithm 1 (in Section 4.1) is more likely to fail to return Cornu-spiral sub-paths for hard turns, and this consumes CPU time. Furthermore, if Algorithm 1 fails to return Cornu-spiral sub-paths, then the Interpolation algorithm will return circular sub-paths, which typically have much greater arc lengths compared with Cornu-spiral sub-paths. However if no hard turns are required, then the Interpolation algorithm quickly returns an interpolating path with minimal arc length (in a heuristic sense).

Although this note emerged from a study of path planning for military aircraft, the Interpolation algorithm is generic and potentially has a broad range of applications. The utility of the Interpolation algorithm will be determined by the context in which the data is generated and the intended use of the interpolating path.

Acknowledgements

This work was stimulated by discussions with Josef Zuk. During the preparation of this note I also received invaluable feedback from Russell Connell, Simon Goss and Josef Zuk; this work would not have come to fruition without their input.

References

Abramowitz, M. & Stegun, I. A. (1972) *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, Dover, New York.

Atkinson, K. E. (1989) *An introduction to numerical analysis*, 2nd edn, Wiley, New York.

Dai, R. & Cochran Jr., J. E. (2010) Path planning and state estimation for unmanned aerial vehicles in hostile environments, *Journal of Guidance, Control, and Dynamics* **33**(2), 595–601.

Davis, T. G. (1999) Total least-squares spiral curve fitting, *Journal of Surveying Engineering* **125**(4), 159–176.

Kelly, A. & Nagy, B. (2003) Reactive nonholonomic trajectory generation via parametric optimal control, *The International Journal of Robotics Research* **22**, 583–601.

Meek, D. S. & Walton, D. J. (1989) The use of cornu spirals in drawing planar curves of controlled curvature, *Journal of Computational and Applied Mathematics* **25**, 69–78.

Meek, D. S. & Walton, D. J. (1992) Clothoid spline transition spirals, *Mathematics of Computation* **59**(199), 117–133.

Stoer, J. (1982) Curve fitting with clothoidal splines, *Journal of Research of the National Bureau of Standards* **87**(4), 317–346.

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. CAVEAT/PRIVACY MARKING

2. TITLE Constant Speed Interpolating Paths	3. SECURITY CLASSIFICATION Document (U) Title (U) Abstract (U)
4. AUTHOR Jason R. Looker	5. CORPORATE AUTHOR Defence Science and Technology Organisation 506 Lorimer St, Fishermans Bend, Victoria 3207, Australia

6a. DSTO NUMBER DSTO–TN–0989	6b. AR NUMBER AR 014–939	6c. TYPE OF REPORT Technical Note	7. DOCUMENT DATE March, 2011

8. FILE NUMBER 2011/1024184/1	9. TASK NUMBER DS 07/245	10. TASK SPONSOR CAOD	11. No. OF PAGES 16	12. No. OF REFS 8

13. URL OF ELECTRONIC VERSION `http://www.dsto.defence.gov.au/publications/scientific.php`	14. RELEASE AUTHORITY Chief, Air Operations Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for Public Release

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS

No Limitations

18. DSTO RESEARCH LIBRARY THESAURUS

algorithms, approximation, interpolation, numerical algorithms, trajectories

19. ABSTRACT

Constant speed paths have been used as interpolating functions for highway and railway design, motion planning for robots, and path planning for military aircraft. A simple algorithm is developed in this note to compute a two-dimensional path that interpolates between ordered data points, consisting of Cornu-spiral, straight-line and circular sub-paths. The interpolation algorithm determines the sub-path parameters such that the interpolating path has constant speed, continuous heading and bounded curvature, while endeavouring to sequentially minimise the sub-path arc lengths.

The most significant factor to influence the performance of the interpolation algorithm is the number of data points that require the interpolating path to execute a hard turn. If no hard turns are required, then the interpolation algorithm quickly returns an interpolating path with minimal arc length (in a heuristic sense). However, hard turns can substantially increase both the CPU time of the algorithm and arc length of the path.