Path planning with PH G2 splines in R2

Laurent Gajny, Richard Béarée, Eric Nyiri, Olivier Gibaru

Path planning with PH G^2 splines in $\mathbb{R}^2$

Laurent Gajny*, Richard Béarée*, Éric Nyiri*, Olivier Gibaru*§
*Arts et Métiers ParisTech, LSIS - UMR CNRS 7296, 8 Boulevard Louis XIV, 59046 Lille Cedex, France
§INRIA Lille-Nord-Europe, 40 avenue Halley 59650 Villeneuve d'Ascq, France

***Abstract*—In this article, we justify the use of parametric planar Pythagorean Hodograph spline curves in path planning. The elegant properties of such splines enable us to design an efficient interpolator algorithm, more precise than the classical Taylor interpolators and faster than an interpolator based on arc length computations.**

***Index Terms*—Path planning, Pythagorean-hodograph, Splines.**

I. Introduction

Designing a good geometric support from given points is a fundamental issue in path planning for industrial machines (Machine-tool, manipulator, robot...). Traditionnally, the methods used consist in minimizing the L_p norm of the second derivative of polynomial slines which interpolate the given points, for $p \in \mathbb{N}^*$ (See for example [1] for the general method and [9] for its application). In this article, we are interested in a subclass of polynomial splines called Pythagorean Hodograph splines.

Pythagorean Hodograph curves (PH curves), introduced in 1990 in [2] are characterized by the special property that the derivative of their arc length is a polynomial (or rational) function of the curve parameter. This property enables to compute exactly useful quantities such as curvature, bending energy or arc length. For simple polynomial curve, we can not, in general, calculate these quantities and so, we apply some quadrature rules. By a good choice of the quadrature rule, we can reach a satisfying precision but it may be highly time consuming.

It is not difficult to design a geometric support which interpolates data points using PH curves. We can apply for example L_1C^1 method from [1] which gives some derivative values at each data point and reconstruct a solution as PH quintic spline by C^1 Hermite interpolation (See [3]). But C^1 continuity is not sufficient for path planning of mechanical system if trajectory following performances are of interest. Indeed, curvature discontinuities, which are not physically followed, excite the structure and can significantly deteriorate the accuracy. Thus, we use the G^2 interpolation scheme introduced by Jaklič *et al.* in [7]. By this method, we can only treat, for the moment, convex data but it uses low degree spline (cubic). The existence of solution is guaranteed and the iterative method to construct the solution converges rapidly.

Some articles (See for example [4], [5], [11]) deal with the use of PH curves in CNC. In these articles, authors only treat the case of a single PH curve and designed an interpolator based on arc length computation in order to follow the parametric geometric support. We will compare the performances in running time and accuracy of such an interpolator for both PH and classical polynomial curve cases. Then, We will extend this procedure on a PH spline.

This paper is organized as follows. Section II gives some basic information about Pythagorean Hodograph cubic curves. In section III, we present briefly the interpolation scheme that we will use in our experiment. Section IV is dedicated to the PH interpolator, its application and a comparison with other interpolators. Some conclusions will be drawn in the last section.

II. Pythagorean Hodograph cubic curves

A polynomial Pythagorean hodograph curve $r(\tau) = (x(\tau), y(\tau))$, $\tau \in [0,1]$ is defined by the property that its hodograph $r'(\tau) = (x'(\tau), y'(\tau))$ satisfies the Pythagorean condition :

$$x'^2(\tau) + y'^2(\tau) = \sigma^2(\tau), \tag{1}$$

for some polynomial $\sigma(\tau)$. This remarkable property enables to compute exactly some useful quantities like arc length, curvature or bending energy. To define curves satisfying (1), we use two polynomials, $u(\tau)$ and $v(\tau)$ such that :

$$\begin{aligned} x'(\tau) &= u^2(\tau) - v^2(\tau), \\ y'(\tau) &= 2u(\tau)v(\tau), \\ \sigma(\tau) &= u^2(\tau) + v^2(\tau). \end{aligned}$$

For PH cubics, we shall choose $u(\tau)$ and $v(\tau)$ linear functions expressed in the Bernstein basis :

$$\begin{aligned} u(\tau) &= u_0\mathbf{B}_0^1(\tau) + u_1\mathbf{B}_1^1(\tau), \\ v(\tau) &= v_0\mathbf{B}_0^1(\tau) + v_1\mathbf{B}_1^1(\tau). \end{aligned} \tag{2}$$

In order to define regular curves, we may assume $(u_1-u_0)^2 + (v_1 - v_0)^2 \neq 0$ and $u_0v_1 - u_1v_0 \neq 0$. By integrating the hodograph, we obtain the formulation of Bézier control points of a PH cubic curve.

$$\begin{aligned} \mathbf{Q}_1 &= \mathbf{Q}_0 + \frac{1}{3}(u_0^2 - v_0^2, 2u_0v_0) \ , \\ \mathbf{Q}_2 &= \mathbf{Q}_1 + \frac{1}{3}(u_0u_1 - v_0v_1, u_0v_1 + u_1v_0) \ , \\ \mathbf{Q}_3 &= \mathbf{Q}_2 + \frac{1}{3}(u_1^2 - v_1^2, 2u_1v_1). \end{aligned} \tag{3}$$

We have a sufficient and necessary geometric condition on the control polygon, illustrated in Fig. 1, for a parametric cubic curve to be a PH cubic. The following theorem will be very useful in the spline interpolation scheme we will use in section III.

Theorem 1: Let $r(\tau)$ be a parametric planar cubic curve with Bézier control points $\mathbf{Q}_0,\ \mathbf{Q}_1,\ \mathbf{Q}_2,\ \mathbf{Q}_3$. Let $\mathbf{L}_i = \mathbf{Q}_i\mathbf{Q}_{i+1}$, $i \in \{0,1,2\}$ be the lengths of the control polygon edges and θ_1, θ_2 be the control polygon angles at the interior vertices $\mathbf{Q}_1$ and $\mathbf{Q}_2$. Then $r(\tau)$ is a PH cubic if and only if :

$$\mathbf{L}_1 = \sqrt{\mathbf{L}_0\mathbf{L}_2}, \quad \text{and,} \quad \theta_1 = \theta_2. \tag{4}$$

The proof of this theorem can be found in [6] section 18.3.

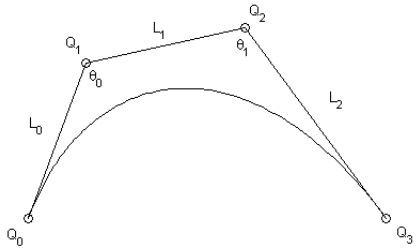

Fig. 1. A PH cubic curve and its control polygon.

III. PH G^2 SPLINE INTERPOLATION

The problem is to interpolate $P_0,\ P_1, \ldots,\ P_n$ in $\mathbb{R}^2$ by a G^2 PH cubic spline with end conditions d_0 and d_n, respectively the tangent vectors at P_0 and P_n. We will use L_1C^1 interpolation, which have good shape-preserving properties, to compute d_0 and d_n. We give the main result of the article of Jaklič *et al.* [7].

Theorem 2: Let $P_0,\ P_1, \ldots,\ P_n$ a set of convex data and $d_0,\ d_n$ the tangent vectors at P_0 and P_n. The interpolation problem of finding a G^2 PH cubic spline passing through the P_i with respect of d_0 and d_n has an admissible (without loops or cusps) solution if and only if the angles

$$\begin{aligned}
&\varphi_0 = \angle(d_1, \Delta P_1),\\
&\varphi_l = \angle(\Delta P_{l-1}, \Delta P_l), \quad l = 1,2,\ldots,n-1,\\
&\varphi_n = \angle(\Delta P_{n-1}, d_n),
\end{aligned}$$

satisfy $\varphi_i + \varphi_{i-1} < M$ with $M = 4\pi/3$ for $i = 1,2,\ldots,n-1$. If $M = K\pi$ with,

$$K = 1 + \frac{1}{\pi}\arccos\left(\frac{\sqrt{3}}{3}\right) \approx 1.304087 < 4/3,$$

then the solution is unique.

The system of non-linear equations, directly obtained by explicitation of (4) on each segment is solved by an iterative method. To understand the algorithm, we may consider the G^1 Hermite interpolation problem with PH cubic. Let $d_0,\ P_0,\ P_1,\ d_1$ a convex G^1 Hermite data. P_0 and P_1 are two points in the plane, d_0 and d_1 the associated tangent vectors. We are looking for $\lambda_0,\ \lambda_1 > 0$ such that :

$$\begin{aligned}
\mathbf{Q}_0 &= P_0,\\
\mathbf{Q}_1 &= P_0 + \lambda_0 d_0,\\
\mathbf{Q}_2 &= P_1 - \lambda_1 d_1,\\
\mathbf{Q}_3 &= P_1,
\end{aligned} \tag{5}$$

are the control points of a PH cubic. The authors determine a unique admissible solution :

$$\lambda_0 = \Lambda_0(d_0, \Delta P_0, d_1), \quad \lambda_1 = \Lambda_1(d_0, \Delta P_0, d_1),$$

under the assumption that the angles $\varphi_0 = \angle(d_0, \Delta P_0)$ and $\varphi_1 = \angle(d_1, \Delta P_0)$ satisfy the condition :

$$\varphi_0 + \varphi_1 < \frac{4}{3}\pi.$$

The global algorithm of G^2 PH cubic spline determination can be written as follows :

Inputs : Data points $P_0, P_1, \ldots, P_n$, end conditions d_0, d_n, a tolerance ε .
Outputs : Tangent vectors at each data points and lengths of the first and the third edges of each control polygon.

1) $\lambda = (1,1,\ldots,1) \in \mathbb{R}^{2m}$, $r = 0$
2) Compute $d_l^{[0]} = \frac{P_{l+1} - P_{l-1}}{\|P_{l+1} - P_{l-1}\|}$, $l = 1,2,\ldots,m-1$.
3) Compute $\lambda_{2l} = \Lambda_0\left(d_l^{[r]}, \Delta P_l, d_{l+1}^{[r]}\right)$, $\lambda_{2l+1} = \Lambda_1\left(d_l^{[r]}, \Delta P_l, d_{l+1}^{[r]}\right)$, $l = 0,1,\ldots,m-1$.
4) Compute
$$d_l^{[r+1]} = \left(\frac{1}{\lambda_{2l}^2}\left(\Delta P_l - \lambda_{2l+1} d_{l+1}^{[r]}\right) + \frac{1}{\lambda_{2l-1}}\left(\Delta P_{l-1} - \lambda_{2l-2} d_{l-1}^{[r]}\right)\right),$$
$l = 1,2,\ldots,m-1$.
5) $D_l = \left\|d_l^{[r+1]}\right\|, \quad l = 0,1,\ldots,m.$
6) If $\|\lambda_{old} - \lambda_{new}\| < \varepsilon$ and $\|D_{old} - D_{new}\| < \varepsilon$, STOP else $r = r+1$ and go back to 3.

We have applied this algorithm to interpolate the logarithmic spiral and a circle. We note in Fig. 2 that the resulting curves preserve well the shape of the data.

IV. PH INTERPOLATOR

A. Principle

In this part, we wish that a mechanical system follows a geometric motion support with respect of a known feedrate. The geometric support is designed by interpolation of some checkpoints. We choose here a set of convex data that we interpolate by the previous procedure. Knowing the feedrate enables us, by integration, to define a position law (See

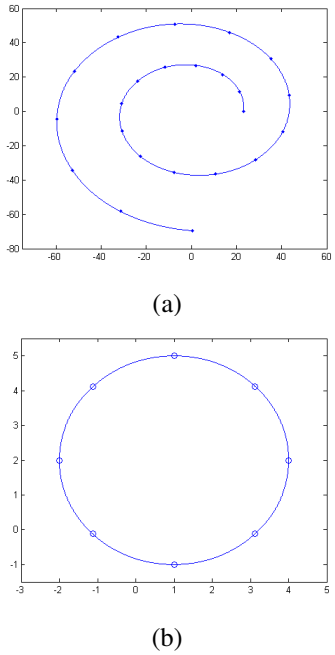

Fig. 2. Interpolation by G^2 PH cubic spline. (a) Logarithmic spiral (b) Circle.

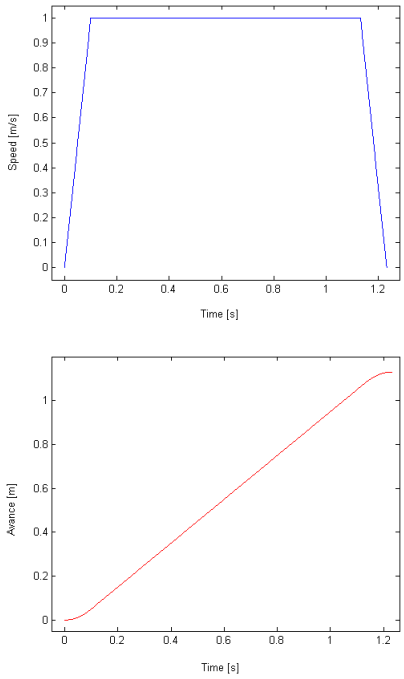

Fig. 3. A simple feedrate (top) and its associated position law (bottom).

Fig. 3) which means that we know, at time t_0, the distance the system must have run through. The main problem is to determine the geometric parameter which corresponds to the time parameter. In practice, we discretize the time interval $[0, T]$ with a constant[1] sampling period Δt. We obtain the list $0 = t_0 < t_1 < \cdots < t_m = T$. For each $i \in \{0, 1, \ldots, m\}$, we have to find the associated geometric parameter of the curve τ_i by solving the following equation :

$$l(t_i) = s(\tau_i),$$

where l and s are respectively the arc length functions of the position law and the spline.

B. Arc length computation for a PH curve

Let $r(\tau)$ be a PH cubic defined by the coefficients $u_0,\ u_1,\ v_0,\ v_1$ as in (2). Arc length at parameter $\tau \in [0, 1]$ may be expressed in the form :

$$s(\tau) = \sum_{k=0}^{3} s_k \binom{n}{k} (1-\tau)^{3-k} \tau^k, \tag{6}$$

where

$$s_0 = 0, \qquad s_k = \sum_{j=0}^{k-1} \sigma_j, \quad k = 1, \ldots, 3$$

and :

$$\sigma_0 = u_0^2 + v_0^2, \quad \sigma_1 = u_0 u_1 + v_0 v_1, \quad \sigma_2 = u_1^2 + v_1^2. \tag{7}$$

[1]We can choose a variable stepsize. It will not change the performances of the method.

We have in particular :

$$s(1) = \frac{\sigma_0 + \sigma_1 + \sigma_2}{3}. \tag{8}$$

So contrary to ordinary polynomial curve, we have a closed form for the arc length at parameter τ.

C. Algorithm

We now describe the algorithm we will use further both in case of a simple polynomial spline and a PH spline.

Inputs : Data points and associated parameters, the interpolating spline defined by some coefficients, the position law, a time list.
Output : A list of geometric parameters.

1) Compute $l_k = l(t_k),\ k = 0, 1, \ldots m$.
2) Compute $S_j,\ j = 0, 1, \ldots, n$ the arc lengths of the PH cubic spline at each ν_j.
3) $\tau_0 = \nu_1$; $r_0 = 1$.
4) Compute $r_k,\ k = 1, 2, \ldots, n$ the number of the spline piece where τ_k must be.
5) If $r_k = r_{k-1}$, solve :

$$l_k - l_{k-1} = s_{r_k}\left(\frac{\tau_k - \nu_{r_k}}{\nu_{r_k+1} - \nu_{r_k}}\right) - s_{r_k}\left(\frac{\tau_{k-1} - \nu_{r_k}}{\nu_{r_k+1} - \nu_{r_k}}\right),$$

$$\tau_{k-1} \leq \tau_k \leq \nu_{r_k+1}.$$

6) If $r_k \neq r_{k-1}$, solve :

$$l_k - S_{r_k} = s_{r_k}\left(\frac{\tau_k - \nu_{r_k}}{\nu_{r_k+1} - \nu_{r_k}}\right),$$
$$\nu_{r_k} \leq \tau_k \leq \nu_{r_k+1}.$$

Here, $s_j(.)$ is the arc length function of the j^{th} piece of the spline. The equations to solve in case of PH cubic spline are also cubics. So, we can solve them exactly by Cardano method. In practice, for a given precision $\varepsilon = 10^{-13}$, we use the Newton-Raphson algorithm which leads to the solution with machine precision in two or three iterations. For polynomial spline, using this algorithm leads to use a quadrature formula. We will see that in applications, it is more time consuming.

V. Numerical experiments

A. Travelling a single PH curve

Let us give a PH curve and a feedrate as shown in Fig. 4. This feedrate has been generated with the limited jerk strategy described in [10]. We will apply the algorithm presented in section IV-C for this curve using the Pythagorean property (1) and in a second experiment, seeing it as a classical polynomial curve and using a quadrature rule. In both case, we choose a sampling period of 10^{-3}s.

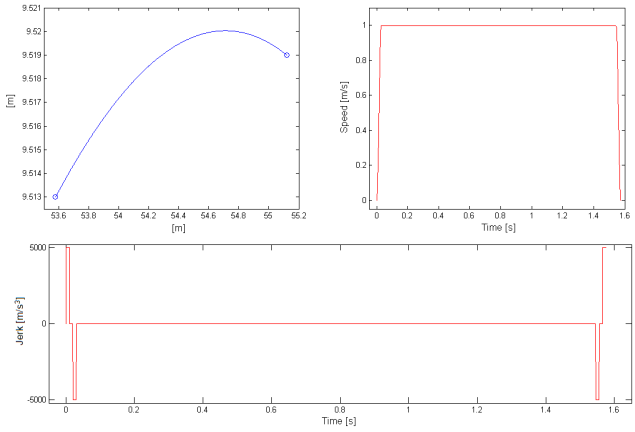

Fig. 4. A single PH curve to travel (Top, left). The desired feedrate (Top, right) and jerk (Bottom).

An error analysis for the PH case is shown in Fig. 5. Final position error is about 10^{-13}m and speed error globally oscillates between 10^{-11} and 10^{-13} $\mathrm{m.s}^{-1}$. It occurs very small oscillations on the feedrate.

We now consider that we do not have a PH curve but only a general polynomial one. We obtain similar accuracy as shown in Fig. 6. The major difference is the running time. For this example, treating about 1500 points of discretization takes near to 85 seconds in the polynomial case and only 6 in the PH case.

In order to prove this observation on running time, we launch several experiments with different sampling periods. The results are summarized in Table I and Fig. 7. We notice a quasi-linearity property with approximated constant 0.0038

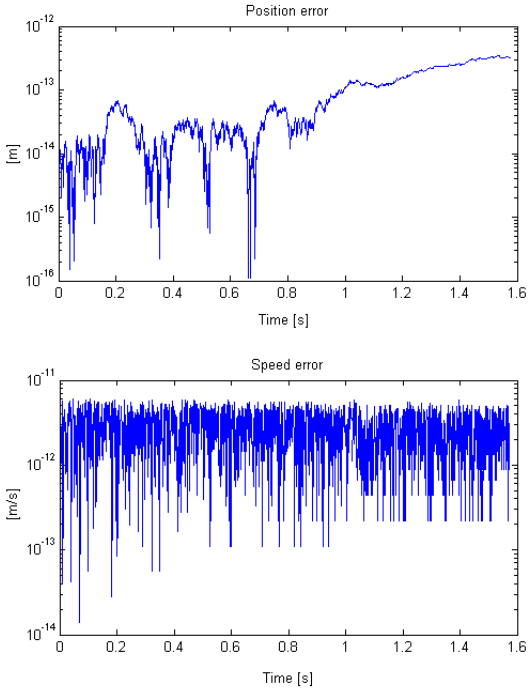

Fig. 5. Semilogarithmic errors for position and speed in the single PH case.

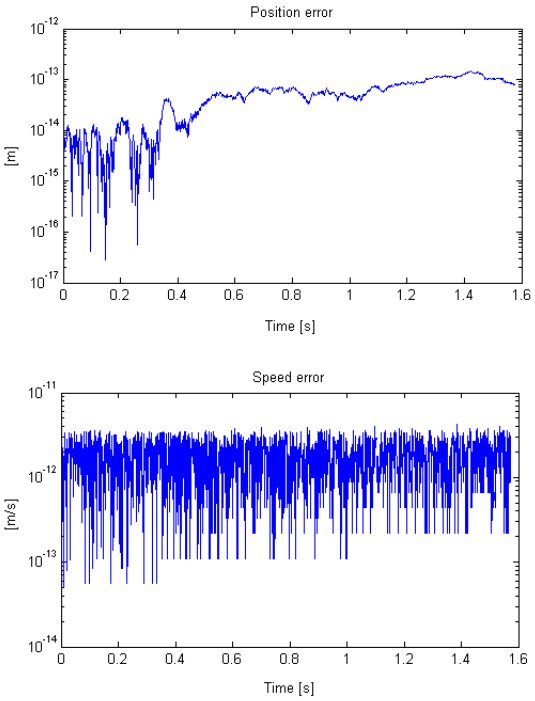

Fig. 6. Semilogarithmic errors for position and speed in the single polynomial case.

in PH case and 0.0546 in polynomial case. So, for the given precision ε, it takes approximatively 14.4 more time to run the interpolator with polynomial splines than PH splines. In polynomial case, running time will be too high when we will have to treat an entire spline and not just a single part. That seems to be the reason that such an interpolator is not used and

that interpolators based on Taylor expansions, less accurate but less time consuming, are prefered.

Sampling period	Number of points	CPU Time, PH	CPU Time, polynomial
10^{-1}s	30	0.1100 s	1.5620 s
8.10^{-2}s	34	0.1410 s	1.7960 s
6.10^{-2}s	40	0.1560 s	2.1250 s
4.10^{-2}s	53	0.2040 s	2.8120 s
2.10^{-2}s	91	0.3430 s	4.8600 s
10^{-2}s	169	0.6250 s	9.1100 s
8.10^{-3}s	209	0.7810 s	11.2650 s
6.10^{-3}s	274	1.0000 s	14.6880 s
4.10^{-3}s	403	1.5320 s	21.7040 s
2.10^{-3}s	794	3.0310 s	42.6720 s
10^{-3}s	1576	6.1090 s	84.6720 s
8.10^{-4}s	1966	7.2030 s	105.7340 s
6.10^{-4}s	2619	9.8590 s	141.0320 s
4.10^{-4}s	3920	13.9530 s	210.8900 s
2.10^{-4}s	7828	29.2030 s	422.6090 s
10^{-4}s	15643	59.3120 s	853.7970 s

TABLE I
CPU TIMES OF THE INTERPOLATOR IN SINGLE PH AND SINGLE POLYNOMIAL CASES FOR SOME SAMPLING PERIODS.

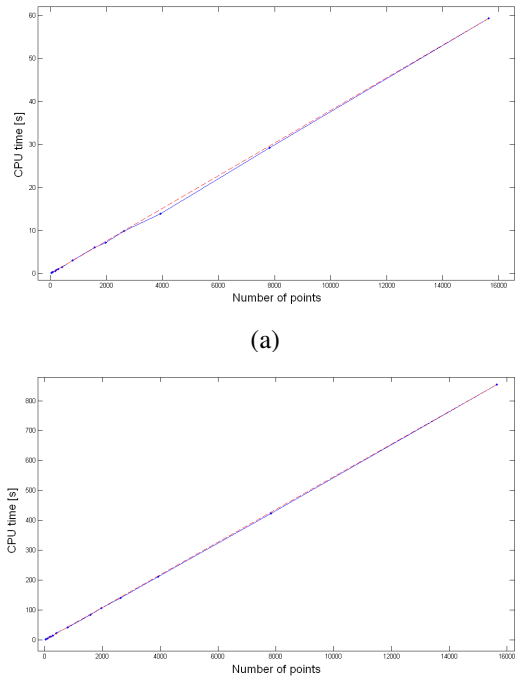

Fig. 7. CPU time in function of the number of discretization points, (a) in the PH case, (b) in the polynomial case.

B. Travelling a PH spline

We are now interested in following a real PH G^2 spline with a desired feedrate as shown in Fig. 8. Again, this feedrate has been generated with the strategy described in [10].

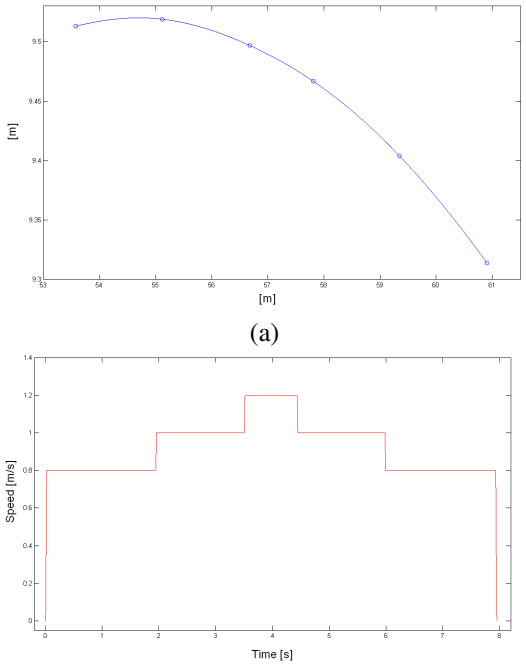

Fig. 8. (a) PH cubic spline to travel. (b) The desired feedrate.

Once again, we present error graphs in Fig. 9. We clearly distinguish the pieces of the spline in these graphs. When we change of spline piece, the previous error is corrected by starting on the new piece by its extremity and not by the last computed point. This produces limited jumps for the speed error which are not important.

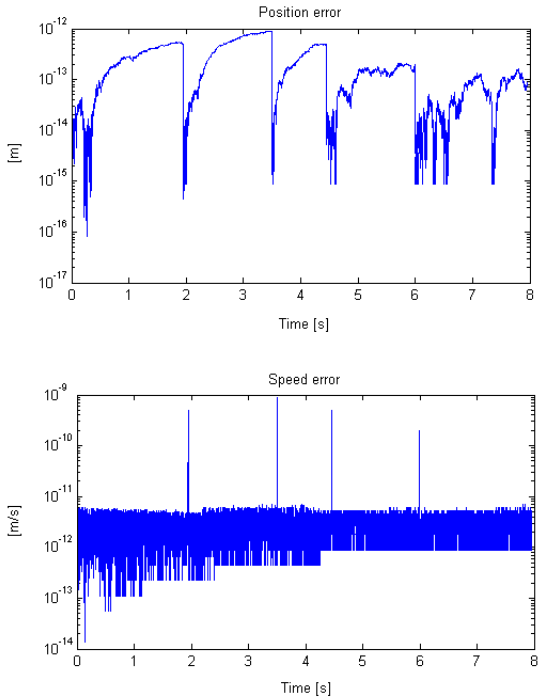

Fig. 9. Semilogarithmic errors for position and speed in the PH spline case.

We compare the efficiency of the algorithm with an interpolator based on a first order Taylor expansion. The parameters τ_i, $i = 0, 1, \ldots, m$ (with $m = 7959$ in our example) are given by the approximation :

$$\tau_{i+1} = \tau_i + \frac{V_i}{\left|\left|\frac{\mathrm{d}\Gamma(\tau)}{\mathrm{d}\tau}\right|\right|_{\tau=\tau_i}} T_s, \tag{9}$$

where V_i is the desired speed at time t_i, Γ the spline obtained by the $L_1^2C^2$ method (see [1]) and T_s the sampling period. With this interpolator, we evidence in Fig. 10 an average position error of 0.3 mm and a final position error of 5 microns. Error graphs presented in Fig. 11 show the accuracy default of the method.

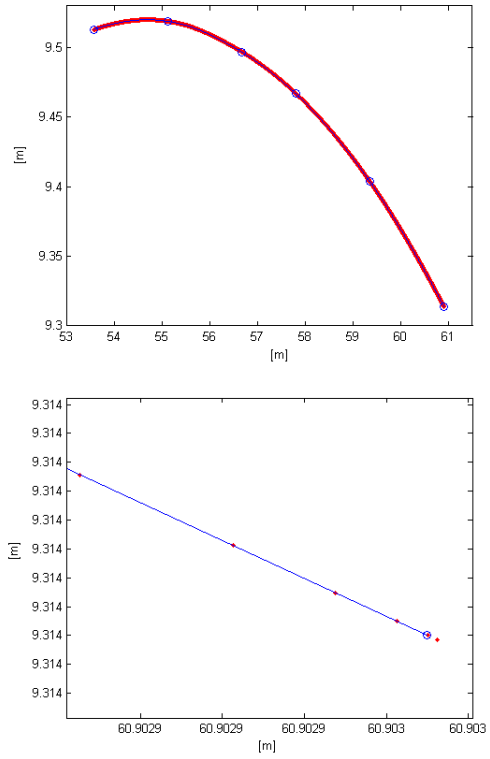

Fig. 10. Global result and final error position for the first order Taylor interpolator.

Conclusion

In this article, we have justified the use of Pythagorean hodograph splines in path planning. The associated PH interpolator based on exact arc length calculation shows similar accuracy than a similar one for simple polynomial splines but it is almost fifteen times faster. Moreover, it is much more precise than a Taylor interpolator. Thus, PH splines seems to be a good solution for real time and accuracy issues.

Future work will concentrate on application of this work for real path and on designing a G^2 PH interpolation scheme for non convex data.

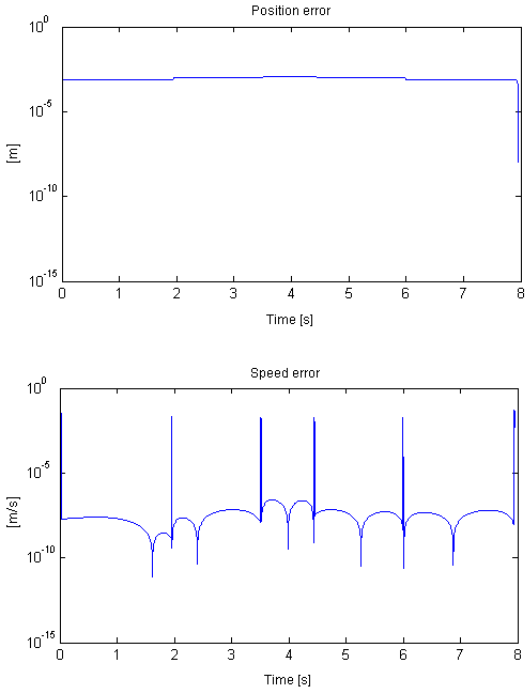

Fig. 11. Semilogarithmic errors for position and speed with the first order Taylor interpolator.

References

[1] Philippe Auquiert, Olivier Gibaru, Éric Nyiri, *Fast $L_1^kC^k$ polynomial spline interpolation algorithm with shape-preserving properties*. Computer Aided Geometric Design 28 (2011) 65 - 74.

[2] Rida T. Farouki, T. Sakkalis, *Pythagorean Hodographs*. IBM Journal of Research and Development 34 (1990) 736 - 752.

[3] Rida T. Farouki, Andrew Neff, *Hermite interpolation by Pythagorean hodograph quintics*. Mathematics of computation 64, n212 (1995) 73 - 83.

[4] Rida T. Farouki, Sagar Shah, *Real time CNC interpolators for Pythagorean-hodograph curves*. Computer Aided Geometric Design 13 (1996) 583 - 600.

[5] Rida T. Farouki, Yi-Feng Tsai, *Exact Taylor series coefficient for variable-feedrate CNC curve interpolators.* Computer Aided Geometric Design 33 (2001) 155 - 165.

[6] Rida T. Farouki, *Pythagorean-hodograph curves - Algebra and geometry inseparable.* Springer-Verlag (2008).

[7] Gašper Jaklič, Jernej Kozak, Marjeta Krajnc, Vito Vitrih & Emil Žagar, *On interpolation by planar cubic G^2 Pythagorean-Hodograph spline curves*. Mathematics of computation 79, n69 (2010) 305 - 326.

[8] D.S. Meek, D.J. Walton, *G^2 curve design with a pair of Pythagorean Hodograph quintic spiral segments.* Computer Aided Geometric Design 24 (2007) 267 - 285.

[9] Adel Olabi, Richard Béarée, Éric Nyiri, Olivier Gibaru *Enhanced Trajectory Planning For Machining With Industrial Six-Axis Robots*.IEEE International Conference on Industrial Technology, ICIT (2010).

[10] Adel Olabi, Richard Béarée, Olivier Gibaru, Mohamed Damak, *Feedrate planning for machining with industrial six-axis robots.* Control Engineering Practice, Vol. 18 Issue 5 (2010) 471 - 482.

[11] Yi-Feng Tsai, Rida T. Farouki, Bryan Feldman, *Performance analysis of CNC interpolators for time-dependent feedrates along PH curves*. Computer Aided Geometric Design 18 (2001) 245 - 265.