2013 IEEE/ASME International Conference on
Advanced Intelligent Mechatronics (AIM)
Wollongong, Australia, July 9-12, 2013

Exponential Trajectory Generation for Point to Point Motions

Z. Rymansaib
zr202@bath.ac.uk

P. Iravani
pi204@bath.ac.uk

M.N.Sahinkaya
ensmns@bath.ac.uk

Department of Mechanical Engineering
University of Bath, BA2 7AY, UK

Abstract— **This paper presents a new method to generate point-to-point (PTP) trajectories, such as the ones used by robots or CNC machines, using an exponential function as the basis for the trajectory profile. The method is based on adding a series of time-delayed third-order exponential functions to generate an approximation to the trapezoidal velocity profile commonly used in time-optimal motions. The exponential velocity has zero-starting and ending values as well as continuous derivatives (acceleration and jerk).**

The proposed algorithm has several advantages over conventional trajectory planning methods such as those using S-curve and trapezoidal velocity profiles. These include: (i) the generated trajectory is continuous up to the third derivative, *i.e.* jerk, resulting in smooth machine motions (ii) only two control parameters, a gain and a delay, are required simplifying trajectory planning, and (iii) allows for simple blending of consecutive trajectories.

The method has been tested experimentally on an industrial six Degree-of-Freedom (DOF) robot, a KUKA KR5 sixx R650. The results show position accuracy improvements over conventional S-curve and time-optimal (trapezoidal) velocity methods. The implementation is based on a simple look-up table which enables its real-time implementation.

I. Introduction

Trajectory planning is a key stage in processes requiring precision movements, such as those often used by machine tools, CNCs, and robots in general. Limiting jerk has been shown to reduce wear and extend machine and tool life, as well as enabling better quality surface finishes in machining tasks [1], [2]. Actuator performance is adversely affected by sudden changes in jerk, and minimising jerk discontinuities leads to the reduction of position tracking errors [3].

Time-optimal trajectories, *i.e.* minimal travel time, are commonly generated using bang-bang/bang-cruise-bang methods, which have a trapezoidal velocity profile [4], also known as Linear Segments with Parabolic Blends (LSPB). Such trajectories are based on generating a triangular/trapezoidal velocity profile using the maximum actuator accelerations. However, these trajectories demand instantaneous acceleration changes. These changes are not physically realisable, and results in the generation of discontinuous actuator torques and forces.

An alternative method that limits jerk and has non-instantaneous acceleration demands is the S-curve velocity profile [5]. With S-curve velocities, the corresponding acceleration starts from zero and gradually ramps to a maximum/minimum value. The continuous acceleration and bounded jerk characteristics make such trajectories more adequate for precision motions and extending actuator life-span. Numerous methods and algorithms have been established which generate such trajectories with jerk limitation [3]–[9]; additionally, the jerk profile can be continuous or discontinuous depending on the method employed.

The method described in this paper approximates the trapezoidal velocity profile with a smoother function, which is less demanding for a controller to perform. The method is based on adding a series of time-delayed third-order exponential function with zero start and end values [10]–[12]. To the best of our knowledge, a method of using this type of exponential function for velocity trajectory planning has not been implemented before. A similar method is shown in [13], but it shows no way of incorporating acceleration or jerk limitation. Additionally, with the recent release of the Reflexxes libraries and work by T.Kroeger [14], using the algorithm for online trajectory generation (OTG) seems feasible due to its simplistic nature as well as its continuous acceleration and jerk properties. Although the main focus here is for pre-planned PTP motions.

The rest of this paper is organised as follows: Section II describes the exponential function and how limits are implemented; Section III describes the experiments and results; Section IV is a discussion of the acquired results, with a brief comparison of the proposed blending functionality against the one described in [9]; Finally, Section V summarises the outcome of the paper and discusses possible implementations and further work.

II. Exponential function

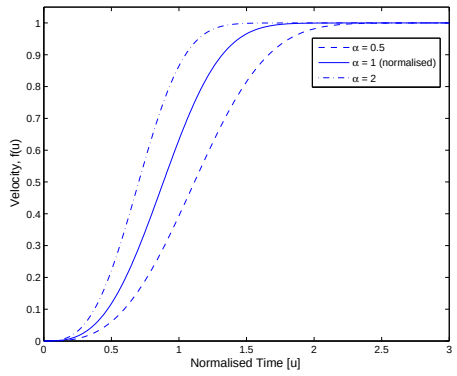
A third-order exponential function is used to generate the basic velocity profile, ensuring that it is continuous up to the third derivative, *i.e.* jerk. To generalise the analysis, a non-dimensional time, u, is defined as:

$$u = \alpha\ t \tag{1}$$

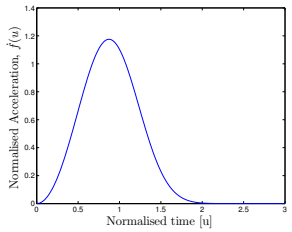
The third order exponential function f, is then defined as:

$$\begin{aligned} f(u) &= \frac{f(t)}{v_{max}} = 1 - e^{-u^3} \\ \dot{f}(u) &= \frac{\dot{f}(t)}{\alpha v_{max}} = 3u^2 e^{-u^3} \\ \ddot{f}(u) &= \frac{\ddot{f}(t)}{\alpha^2 v_{max}} = (6u - 9u^4) e^{-u^3} \end{aligned} \tag{2}$$

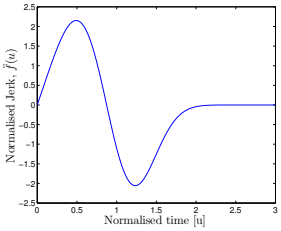
Where v_{max} is the trajectory's maximum or cruising velocity, α is a time-scaling parameter, and t is the time. This function was originally developed to synthesize input shaping functions based on the inverse dynamics of a second order system [10]. Figure 1a illustrates the exponential function profile for three values of $\alpha = 0.5, 1, 2$; the larger values result in a steeper exponential function, and thus when used as a velocity profile, generate a greater acceleration. The normalised acceleration and jerk profiles are illustrated in Figures 1b and 1c. This normalisation allows the method to use a single exponential profile to calculate any demand trajectory. This enables the implementation to be based on storing the function profile once in a simple look-up table.



(a) Exponential velocity function with various α values



(b) Normalised acceleration, $\dot{f}(u)$



(c) Normalised jerk, $\ddot{f}(u)$

Fig. 1: Exponential function characteristics

A. Function maxima

In order to ensure that actuator limits are not violated during the trajectory, the maximum velocities, accelerations and jerks must be computed. These maximum values occur at:

$$\begin{aligned} t_{am} &= \frac{1}{\alpha} \cdot \sqrt[3]{\frac{2}{3}} \approx \frac{0.8736}{\alpha} \\ t_{jm} &= \frac{1}{\alpha} \cdot \sqrt[3]{\frac{3-\sqrt{7}}{3}} = \frac{Q}{\alpha} \approx \frac{0.4906}{\alpha} \end{aligned} \tag{3}$$

Where t_{am} and t_{jm} represent the times at which the maximum acceleration and jerk values occur, respectively. Substituting these times into the respective trajectory equations, the maximum values are found as:

$$\begin{aligned} |V_{max}| &= |v_{max}| \\ |A_{max}| &= |v_{max} \cdot \alpha \cdot 0.3918| \\ |J_{max}| &= \left| v_{max} \cdot \alpha^2 \left(6Q^{(1/3)} - 9Q^{(4/3)} \right) \cdot e^{-Q} \right| \\ &= \left| v_{max} \cdot \alpha^2 \cdot 0.7652 \right| \end{aligned} \tag{4}$$

Therefore, given the actuator velocity, acceleration and jerk limits as V_{max}, A_{max} and J_{max} respectively, the maximum α value which can be used, thus generating the fastest possible motion within the limits, is calculated as:

$$\alpha_{max} = min \left(\frac{A_{max}}{1.1754\ V_{max}}; \sqrt{\frac{J_{max}}{2.1524\ V_{max}}} \right) \tag{5}$$

B. Approximation to the time-optimal velocity

Time-optimal LSPB trajectories exploit the maximum acceleration available to the system to produce the fastest possible PTP motions. Figure 2 illustrates the position, velocity and acceleration profile such a motion. This approach results in step accelerations and trapezoidal velocity profiles. In this particular example, the desired motion distance (displacement) is 1 m, the maximum velocity is 1 m/s and the maximum acceleration 2 m/s^2. As shown, the acceleration is discontinuous and changes instantly from zero to the maximum value. In real systems, this would also require the force to change instantaneously; this is physically impossible, resulting in poor motion tracking.

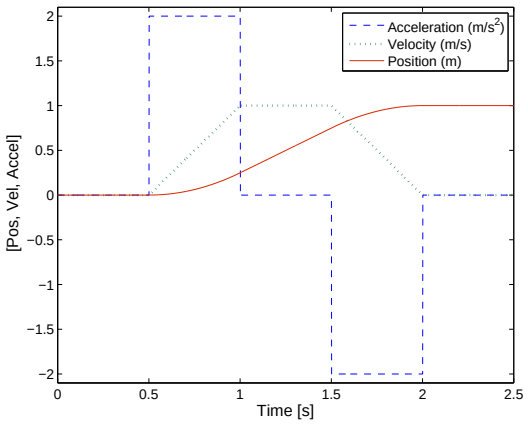


Fig. 2: LSPB time-optimal trajectory

Using an S-curve velocity profile negates these issues by smoothing out the trapezoidal velocity, and instead generates trapezoidal *accelerations*. This makes the overall motion slightly slower, as some extra time is required to allow the system to reach the maximum acceleration, rather than assuming it will instantly reach the value, *i.e.* near time optimal. Figure 3 illustrates an S-curve velocity profile, with corresponding position and acceleration profiles. The first derivative of acceleration, *i.e.* jerk, now has a bounded value, and its jerk profile (not shown) has a rectangular shape

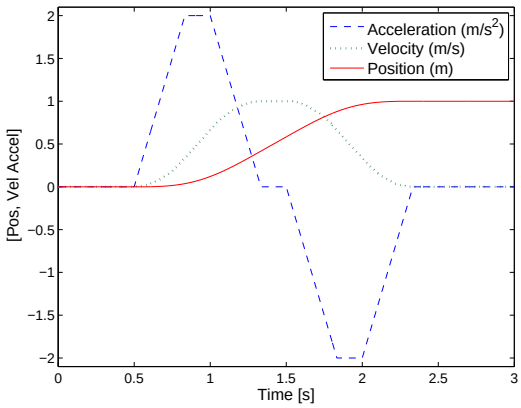


Fig. 3: S-curve velocity (jerk limited)

akin to the resultant acceleration profile of the trapezoidal velocity. The S-curve profiles used in this paper have been generated using the method in [5].

In order to create an approximation to a near time optimal trajectory, this paper proposes the use of two of the velocity profiles defined in Equation 2, used in sequence:

$$v(t) = f(t) - f(t - T_d) \cdot H(t - T_d) \tag{6}$$

Where $H(t)$ is a Heaviside function, defined as:

$$H(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases} \tag{7}$$

Figure 4 illustrates the resultant exponential velocity profile. From $t = 0$ until T_d a single velocity profile is evaluated. At time T_d the second exponential function starts being evaluated but is *subtracted* from the first one. This results in a velocity profile that is similar in essence to one produced by the S-curve method, but the subtle difference is that rather than having a rectangular jerk profile, it more closely resembles a sinusoidal shape which is smooth and continuous, as illustrated in Figure 1c.

Being a velocity profile, the position must be calculated by integrating it over $t = 0$ to t_{end}, where t_{end} is the time at which the velocity returns to 0 m/s. Obviously, the desired end position must be achieved at this point. The integration of this function cannot be calculated analytically, and the concatenation of the two profiles at the correct time complicates this further. The position profile would thus be obtained by integrating numerically. However, the area under the profile for a desired displacement, Δx can be simply found as:

$$\Delta x = V_{max} \cdot T_d \tag{8}$$

As illustrated in Figure 4, this is because the summation of the areas under the combined curves *i.e.* A1, A2 and A3, equals a rectangle of height V_{max} and width T_d.

The function accelerates for T_s seconds; this time defined as the settling time of the function, is the time taken to reach 99.9% of V_{max}, and is computed as:

$$T_s = \frac{\sqrt[3]{-ln(0.001)}}{\alpha} \approx \frac{1.9045}{\alpha} \tag{9}$$

It is possible to attain a settling value closer to 100% of V_{max} by reducing the 0.001 figure.

The velocity then cruises at the maximum speed before taking T_s seconds to reach the resting velocity. Thus, the total travel time t_{end} can be calculated by:

$$t_{end} = T_d + T_s \tag{10}$$

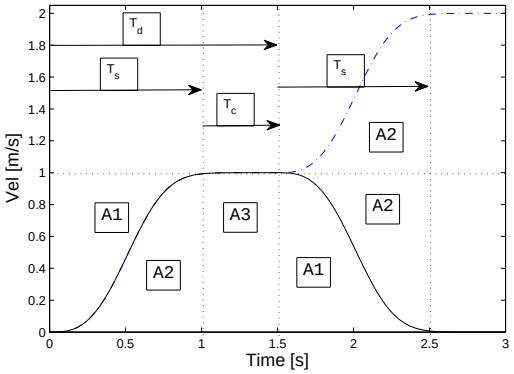


Fig. 4: Exponential velocity profile

Increasing V_{max} means that the delay time T_d would be shorter; additionally, increasing the acceleration means that V_{max} is reached sooner. The parameter tying these two together, α, is thus evaluated beforehand to ensure that velocity, acceleration as well as jerk limits are respected; as described in Equation 5. Thus, given the demanded motion Δx, the relevant timings T_d and t_{end} can be easily calculated.

C. Sequencing point to point trajectories and blending

The section above illustrates how to compute a single PTP motion. A sequence of motions can be generated by iterating the procedure. Each required PTP displacement can be converted into time delays and used sequentially.

Given a sequence of required displacements $x = (x_1, x_2, ...x_n)$ the corresponding time delays are: $T_d = (x_1/V_{max}, x_2/V_{max}...x_n/V_{max})$. If each subsequent motion is started after the previous has reached its desired position (without any trajectory blending), then the total travel time for a sequence of n motions is:

$$T_t = \frac{x_1 + x_2 + ... + x_N}{V_{max}} + n\ T_s \tag{11}$$

During the period T_d to t_{end}, *i.e.* the braking period, the proceeding profile can begin evaluation. This results in the next function starting before the previous one ends, thus, blending motions. The amount of overlap or blending is

varied by controlling the point during the braking region where the second function begins. A 0% overlap indicates that subsequent motions only start when the preceding one reaches its zero velocity. A 100% overlap would cause the second function to start at T_d *i.e.* overlapping the braking and acceleration regions of two functions. With a 100% overlap the total travel time would be:

$$T_t = \frac{x_1 + x_2 + ... + x_N}{V_{max}} + T_s \tag{12}$$

Figure 5 illustrates this concept more clearly. Here, the generated x and y Cartesian velocity commands (absolute values) for tracing a square are shown. The upper figure shows how each motion starts and ends at zero velocity, *i.e.* without any blending. The lower figure shows the same commands but with 100% blending applied. As can be seen, subsequent motions begin evaluation at the delay time, T_d of each preceding one. The velocity is thus always non zero until the final motion is completed.

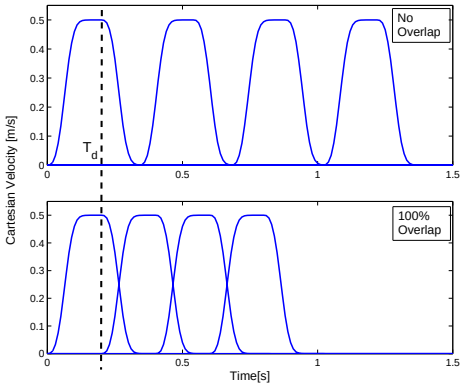


Fig. 5: Unblended and blended Cartesian velocity commands

Controlling this overlap allows for a simple trade-off between way-point tracking accuracy and the path's duration.

III. Experimental Results

The described method was used to generate parameters (α, V_{max}, and T_d), given a set of velocity, acceleration and jerk limits, for stop-to-stop motions between Cartesian-space points using a Matlab script. A real-time Simulink model, with an interface block designed for KUKA, uses these parameters to generate velocity commands to send to the robot. An image of the device used is shown in Figure 6. The robot controller automatically performs the inverse kinematics for the joint angles.

Three different paths are used in the experiments; a 10 x 10 cm square, a rough figure of eight - composed of two hexagonal shapes with 10 cm length sides, and the same 10 cm^2 square overlaid 10 times to evaluate repeatability. They are each effectively sets of waypoints connected together using the exponentials to describe a path. These are run at low and high speeds. After each motion is completed, the robot is homed to the same starting position. In order to

Fig. 6: Kuka 6 DOF robot

compare the accuracy of the methods, the routes traced from repeating 10 squares are averaged to give a mean travel path. An average error value, *i.e.* deviation of end effector from desired positions, is also found for each method.

The S-curve and trapezoidal velocity algorithms generate quicker motions. Their limits are thus adjusted (reduced) so as to slow the motions down to within 100 ms, *i.e.* the average speeds equalised; a faster motion is more prone to overshooting the path points and would thus give an unfair comparison. The limits given to the different algorithms are detailed in Table I, and Tables II and III show the resultant motion durations of using these limits.

The recorded data include: desired input Cartesian positions, the actual position of the robot end-effector, and the joint torques. Data is captured every 10 ms.

Figures 7 and 8 illustrate the desired and achieved motions for a square path (mean of 10 runs) when using the different methods, run at the higher speed. Results of the slower motions are excluded as the differences between these (*i.e.* low speed tracking accuracy) are negligible. Similarly, the desired and achieved paths of the figure of eight shapes and single square runs are excluded; results are similar to the 10 square experiments. The figure of eight shapes are instead used to illustrate the method's blending functionality, with figures 9 and 10 showing the effect of using different overlap parameters.

TABLE I: Velocity (m/s), acceleration (m/s^2) and jerk (m/s^3) limits for each function

	Original			S Curve (adj.)			Trapezoidal (adj.)		
	Vel	Accel	Jerk	Vel	Accel	Jerk	Vel	Accel	Jerk
Slow	0.05	5	200	0.05	5	102	0.05	1.2	-
Fast	0.5	8	200	0.5	8	102	0.5	3.6	-

TABLE II: Actual motion durations (s) using original limits

Speed	Shape Traced	Exponential	S Curve	Trapezoidal
Low	Figure of Eight	25.00	24.85	24.59
	10 Squares	81.77	81.26	80.40
High	Figure of Eight	4.13	3.68	3.20
	10 Squares	13.60	12.10	10.5

TABLE III: Actual motion durations (s) using adjusted limits

Speed	Shape Traced	Exponential	S Curve	Trapezoidal
Low	Figure of Eight	As prev	25.00	24.97
	10 Squares	As prev	81.77	80.67
High	Figure of Eight	As prev	4.13	4.11
	10 Squares	As prev	13.60	13.56

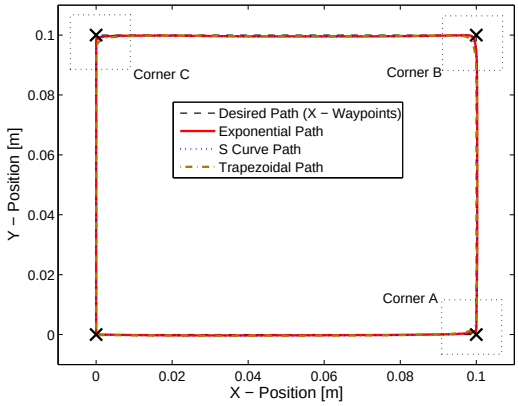


Fig. 7: Desired and achieved square path

IV. Discussion

Figure 8 shows that the performance of the exponential method is comparable to the S-curve algorithm, but in general tracks the corners of the square shape more accurately. This can be attributed to the continuous jerk of the exponential method; as mentioned in the literature such a motion is easier to track. Table IV shows the distances of the points closest to the corners of the square, with table V showing the overall mean errors and standard deviations of the various methods. This was found by averaging the accuracy error for each of the three indicated corners, A, B and C. With the unadjusted motions (*i.e.* faster average velocities) the performances are even worse in comparison,

TABLE IV: Mean error from corners (mm)

Corner	Exponential	S Curve	Trapezoidal
A	1.05	1.23	1.98
B	1.1	1.34	2.07
C	1.04	1.27	2.07

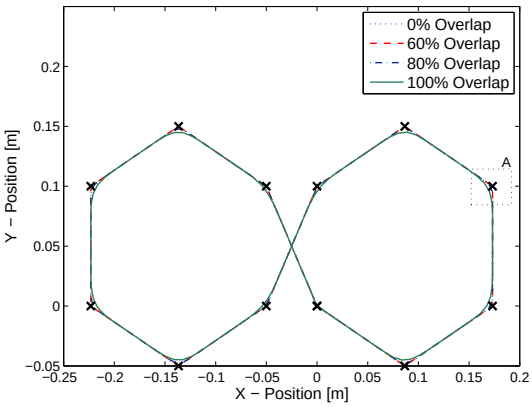


Fig. 9: Figure of eight path with various blend parameters

but this is to be expected.

There is also some slight deviation from the straight-line paths, but this is present with all the tested methods. The methods have been implemented using cartesian velocity control and use of position control would be required to correct the deviations.

As the jerk profiles of the exponential motions are continuous, one would expect the resulting joint torques to be lower than that of the S-curve counterparts. However whilst being discontinuous, the motions are still jerk bounded; this leads to the joints experiencing similar torques throughout the experiments. The results are thus omitted here.

For a given jerk limit, the exponential method is slightly less efficient due to its inability to generate profiles with trapezoidal accelerations, such as those shown in Figure 3. Once the acceleration limit is reached, the velocity also

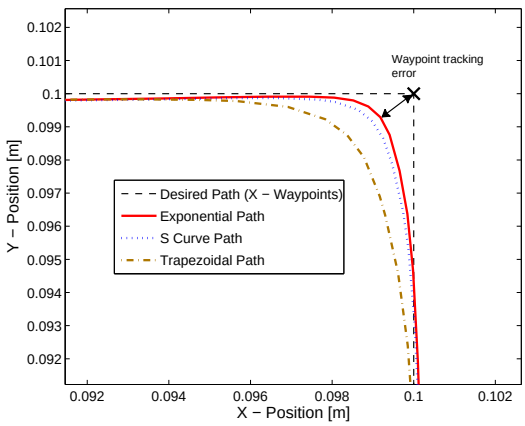


Fig. 8: Corner B zoomed view - Square trajectory

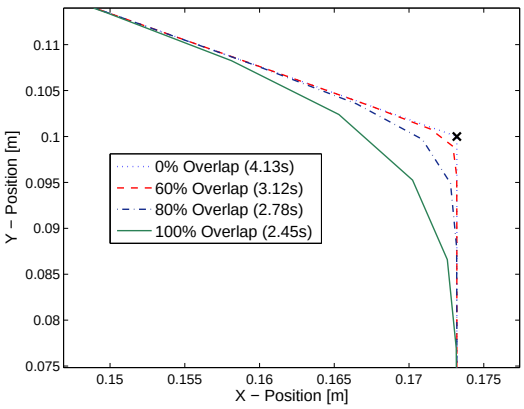


Fig. 10: Corner A zoomed view - figure of eight

TABLE V: Overall mean error and standard deviation (mm)

	Exponential	S Curve	Trapezoidal
Mean Error	1.06	1.28	2.04
Std Dev	0.06	0.07	0.1

TABLE VI: Effect of overlap on motion duration (figure of eight path)

Percentage Overlap(%)	0	50	60	80	100
Motion Duration (s)	4.13	3.29	3.12	2.78	2.45

stops increasing, regardless of whether the velocity limit has been reached (as in Figure 1b). It is therefore not time optimal as such. Generating such motions would involve either repeatedly overlaying multiple modified profiles (with adjusted delays) to approximate a trapezoid, or using the base function shown in Equation 2 to generate the acceleration instead of the velocity. The former would lead to multiple evaluations of the exponential being carried out; the latter leads to a higher order exponential and the requirement of integrating twice for the position, somewhat negating the simplistic nature of this method. An advantage to this approach however is the generation of continuous jounce profiles, *i.e.* the 1st derivative of jerk.

The blending function effectively overlaps the start of the next motion with that of the current one. This has the effect of reducing the overall motion duration, as the overall path length is shortened. A figure-of-eight shape was used to evaluate various blend percentages, as shown in figures 9 and 10. Table VI shows how the motion duration is affected by using higher blending percentages; it can effectively be reduced down to values similar to those produced by the time-optimal LSPB method, but with the added benefits of having continuous acceleration and jerk. The trade-off is slightly reduced accuracy at the waypoints, and as can be seen in figure 10 results in some rounding of the corners. The 'tightness' of the corner tracking is thus controlled by how much overlap is used.

The blending method shown in [9] involves generating multiple stitched fifth order polynomials (quintics) to describe a motion, *e.g.* a trapezoidal type acceleration can require three; one for the acceleration ramp up, one for the velocity cruise and one for the acceleration ramp down. Blending the actual motions together then involves an additional fifth order polynomial, which determines the tightness of the waypoint. In comparison, the exponential method uses only one single parameter to control the blending. The computationally expensive calculation of the exponential function needs to be performed just once off-line and stored in a normalised manner in the real time controller. The controller then only needs to perform simple multiplications to compute the required motion. Current in-progress improvements to the method are to allow for specifying the tracking tolerance when blending waypoints.

V. Conclusion

An exponential based trajectory-planning algorithm was implemented and compared with LSPB and S-curve velocity methods. The method is tested on a KUKA 6 DOF robot; the results show that the proposed method gives better tracking accuracy when switching between paths, is jerk continuous, simpler to implement than the S-curve counterpart and also provides an easy way of blending continuous paths together. It is based simply on calculating a sequence of time delays which are used to start and stop each point to point motion within the demanded trajectory.

Future work involves optimisation of the procedure to allow for a more robust trajectory generation method, as it currently cannot generate motions with sustained accelerations. The method in its current state is therefore ideally suited to robots/manipulators with high speed, but low force capabilities. For example a CNC machine or a RepRap style rapid prototyper.

References

[1] J. R. Rivera-Guillen, R. J. Romero-Troncoso, A. Osornio-Rios, A. Garcia-Perez, and I. Torres-Pacheco, "Extending tool-life through jerk-limited motion dynamics in machining processes: An experimental study," *JSIR*, vol. 69, pp. 919–925, 2010.

[2] A. Piazzi and A. Visioli, "Global minimum-jerk trajectory planning of robot manipulators," *Industrial Electronics, IEEE Transactions on*, vol. 47, pp. 140 –149, feb 2000.

[3] K. Kyriakopoulos and G. Saridis, "Minimum jerk path generation," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, apr 1988, pp. 364 –369 vol.1.

[4] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3248–3253.

[5] H. Mu, Y. Zhou, S. Yan, and A. Han, "Third-order trajectory planning for high accuracy point-to-point motion," *Frontiers of Electrical and Electronic Engineering in China*, vol. 4, no. 1, pp. 83–87, 2009.

[6] K. Erkorkmaz and Y. Altintas, "High speed cnc system design. part i: jerk limited trajectory generation and quintic spline interpolation," *International Journal of Machine Tools and Manufacture*, vol. 41, no. 9, pp. 1323–1345, 2001.

[7] P. Meckl and P. Arestides, "Optimized s-curve motion profiles for minimum residual vibration," in *American Control Conference, 1998. Proceedings of the 1998*, vol. 5. IEEE, 1998, pp. 2627–2631.

[8] H. Liu, X. Lai, and W. Wu, "Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints," *Robotics and Computer-Integrated Manufacturing*, 2012.

[9] S. Macfarlane and E. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 1, pp. 42–52, 2003.

[10] M. Sahinkaya, "Input shaping for vibration-free positioning of flexible systems," *Proceedings of the Institution of mechanical engineers, part I: Journal of Systems and Control Engineering*, vol. 215, no. 5, pp. 467–481, 2001.

[11] M. N. Sahinkaya, "Exponential input shaping for vibration free positioning of lightly damped multi-mode systems," in *Seventh International Conference on Motion and Vibration Control (MOvIC' 04)*, 2004. [Online]. Available: http://opus.bath.ac.uk/2200/

[12] P. Iravani and M. N. Sahinkaya, "Variable-velocity exponential input shaping for position controlled robotic systems," in *3rd Annual Dynamic Systems and Control Conference*, 2010. [Online]. Available: http://opus.bath.ac.uk/24543/

[13] C. M. Muller-Karger, A. Leonell Granados Mirena, and J. Scarpati Lopez, "Hyperbolic trajectories for pick-and-place operations to elude obstacles," *Robotics and Automation, IEEE Transactions on*, vol. 16, no. 3, pp. 294–300, 2000.

[14] T. Kröger and F. M. Wahl, "Online trajectory generation: basic concepts for instantaneous reactions to unforeseen events," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 94–111, 2010.