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 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-crui se-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 lifespan. 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:

\[
u = \alpha t
\]

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

\[
f(u) = \frac{f(t)}{v_{max}} = 1 - e^{-u^3}
\]

\[
\dot{f}(u) = \frac{\dot{f}(t)}{\alpha v_{max}} = 3u^2e^{-u^3}
\]

\[
\ddot{f}(u) = \frac{\ddot{f}(t)}{\alpha^2 v_{max}} = (6u - 9u^4)e^{-u^3}
\]

978-1-4673-5320-5/13/$31.00 ©2013 IEEE
Where \( v_{max} \) is the trajectory’s maximum or cruising velocity, \( \alpha \) 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.

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

\[ V_{max} = |v_{max}| \]
\[ A_{max} = |v_{max} \cdot \alpha \cdot 0.3918| \]
\[ J_{max} = \left| v_{max} \cdot \alpha^2 \left(6Q^{(1/3)} - 9Q^{(4/3)} \right) \cdot e^{-Q} \right| = |v_{max} \cdot \alpha^2 \cdot 0.7652| \]

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

\[ \alpha_{max} = \min \left( \frac{A_{max}}{1.1754 \ V_{max}} ; \sqrt{\frac{J_{max}}{2.1524 \ V_{max}}} \right) \]

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.

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.
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 \( V_{\text{max}} \), and is computed as:

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

It is possible to attain a settling value closer to 100\% of \( V_{\text{max}} \) 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 \( t_{\text{end}} \) can be calculated by:

\[
t_{\text{end}} = T_d + T_s \tag{10}
\]

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 \( t_{\text{end}} \), where \( t_{\text{end}} \) 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, \( \Delta x \) can be simply found as:

\[
\Delta x = V_{\text{max}} \cdot T_d \tag{8}
\]

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 \( V_{\text{max}} \) and width \( T_d \).
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:

$$T_t = \frac{x_1 + x_2 + \ldots + x_N}{V_{\text{max}}} + T_s$$  \hspace{1cm} (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.

![Figure 5: Unblended and blended Cartesian velocity commands](image)

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 ($\alpha$, $V_{\text{max}}$, 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 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.

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.

![Fig. 6: Kuka 6 DOF robot](image)

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

<table>
<thead>
<tr>
<th>Speed</th>
<th>Original Vel</th>
<th>Original Accel</th>
<th>Original Jerk</th>
<th>S Curve (adj.) Vel</th>
<th>S Curve (adj.) Accel</th>
<th>S Curve (adj.) Jerk</th>
<th>Trapezoidal (adj.) Vel</th>
<th>Trapezoidal (adj.) Accel</th>
<th>Trapezoidal (adj.) Jerk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow</td>
<td>0.05</td>
<td>5</td>
<td>102</td>
<td>0.05</td>
<td>5</td>
<td>102</td>
<td>0.05</td>
<td>1.2</td>
<td>-</td>
</tr>
<tr>
<td>Fast</td>
<td>0.5</td>
<td>8</td>
<td>200</td>
<td>0.5</td>
<td>8</td>
<td>102</td>
<td>0.5</td>
<td>3.6</td>
<td>-</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Speed</th>
<th>Shape Traced</th>
<th>Exponential Vel</th>
<th>Exponential Accel</th>
<th>Exponential Jerk</th>
<th>S Curve Vel</th>
<th>S Curve Accel</th>
<th>S Curve Jerk</th>
<th>Trapezoidal Vel</th>
<th>Trapezoidal Accel</th>
<th>Trapezoidal Jerk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Figure of Eight</td>
<td>25.00</td>
<td>24.85</td>
<td>24.59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 Squares</td>
<td>81.77</td>
<td>81.26</td>
<td>80.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>Figure of Eight</td>
<td>4.13</td>
<td>3.68</td>
<td>3.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 Squares</td>
<td>13.60</td>
<td>12.10</td>
<td>10.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE III: Actual motion durations (s) using adjusted limits

<table>
<thead>
<tr>
<th>Speed</th>
<th>Shape Traced</th>
<th>Exponential</th>
<th>S Curve</th>
<th>Trapezoidal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Figure of Eight 10 Squares</td>
<td>As prev 25.00</td>
<td>24.97</td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>Figure of Eight 10 Squares</td>
<td>As prev 4.13</td>
<td>4.11</td>
<td></td>
</tr>
</tbody>
</table>

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, 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

TABLE IV: Mean error from corners (mm)

<table>
<thead>
<tr>
<th>Corner</th>
<th>Exponential</th>
<th>S Curve</th>
<th>Trapezoidal</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.05</td>
<td>1.23</td>
<td>1.98</td>
</tr>
<tr>
<td>B</td>
<td>1.1</td>
<td>1.34</td>
<td>2.07</td>
</tr>
<tr>
<td>C</td>
<td>1.04</td>
<td>1.27</td>
<td>2.07</td>
</tr>
</tbody>
</table>

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

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

Fig. 10: Corner A zoomed view - figure of eight
TABLE V: Overall mean error and standard deviation (mm)

<table>
<thead>
<tr>
<th></th>
<th>Exponential</th>
<th>S Curve</th>
<th>Trapezoidal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Error</td>
<td>1.06</td>
<td>1.28</td>
<td>2.04</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.06</td>
<td>0.07</td>
<td>0.1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Percentage Overlap(%)</th>
<th>0</th>
<th>50</th>
<th>60</th>
<th>80</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion Duration (s)</td>
<td>4.13</td>
<td>3.29</td>
<td>3.12</td>
<td>2.78</td>
<td>2.45</td>
</tr>
</tbody>
</table>

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


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.