Faculty Publications

2003-03-01

A joint trajectory generator for motion recovery

Edward Red

Wei Li

Follow this and additional works at: <https://scholarsarchive.byu.edu/facpub>

Part of the [Mechanical Engineering Commons](#)

Original Publication Citation

Robotica 21.2(Mar 23): 185-191.

BYU ScholarsArchive Citation

Red, Edward and Li, Wei, "A joint trajectory generator for motion recovery" (2003). *Faculty Publications*. 509.

<https://scholarsarchive.byu.edu/facpub/509>

This Peer-Reviewed Article is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.

A joint trajectory generator for motion recovery

Wei Li and Edward Red

Department of Mechanical Engineering, Brigham Young University, Provo, Utah 84602 (USA)

(Received in Final Form: October 11, 2002)

SUMMARY

This paper describes an adaptive S-curve used to recover a tool path upon a system crash in the Windows operating system (OS). For a mechanism such as a robot or machine tool, the joint values, being delivered as setpoints to the servo-controller, are dynamically recorded by the real-time operating system also residing on the computer. The real-time OS can control the abort and record pertinent motion data after Windows OS crash.

Upon system recovery, the recovery trajectory generator examines the setpoints intervals to determine the current *slow joint*. At every trajectory step, and for the current slow joint, the S-curve velocity profile applies the joint entry state (position, speed, acceleration, and jerk) to interpolate the motion between the setpoints in a reverse direction. The other joints are proportionally interpolated (slowed) so that they pass through each setpoint simultaneously with the slow joint, but in a reverse direction.

The trajectory algorithm is optimal since the slow joint always uses the maximum allowable jerk to change the profile speed and acceleration for each trajectory step.

KEYWORDS: Adaptive S-curve; Joint trajectory generator; Motion recovery.

INTRODUCTION

New open controllers may use a real-time operating system (OS) that resides concurrently with the Windows OS on a PC. This environment is more vulnerable to system crashes, but brings tremendous advantage to the integration of modern applications on the control platform.

The *direct machining* research at Brigham Young University uses the process plan in a Computer Aided Manufacturing (CAM) application to control directly a machine tool, without post-processing into APT, CL, or M&G files. In an instance where the Windows OS crashes, it is important to first stop the machining process gracefully, and then recover the tool to an acceptable prior configuration where the process can be restarted.

In a system crash a real-time OS such as VenturCom's RTX can continue processing until the tool can be stopped. Unfortunately, the CAM application running under Windows will lose contact with the mechanism's final joint values through the stop sequence. The tool, once stopped awaiting the Windows OS reboot, may be located relative to the part such that it cannot be safely jogged by the operator without damaging the part.

This paper presents a collision-free recovery sequence that can be applied upon primary OS failure to place the tool in a prior known configuration. It is then possible to restart the process plan from a known configuration and continue the normal process sequence. The procedure begins by recording the joint sequence of joint setpoints fed to the servo-controller in a dynamic array of fixed size, rolling over the joint values when the array fills up.

RECOVERY TRAJECTORY GENERATION

Given a block of joint values, we fit a trajectory for each individual joint through all intermediate values. There are a number of methods to fit an interpolation polynomial to pass through all given joint values.

C.C. COOK and C.Y. Ho¹ proposed cubic spline functions to represent the manipulator's tool trajectory and joint trajectories as a function of time. Lin, Chang, and Luh² used cubic polynomials to optimize joint trajectories for industrial robots. Their method is composed of two parts: (1) off-line trajectory planning; and (2) on-line trajectory tracking. Because they use a numerical algorithm to optimize the cubic polynomial joint interpolator, it is not a real-time algorithm. Therefore, the trajectory planning must be conducted off-line.

Wang³ and Patel and Thompson⁴ proposed B-Spline joint trajectory planning algorithms to blend all given joint values. The shortcoming of these approaches is that the planned trajectory approximates all given joint values and thus deviates from the desired trajectory.

There are a number of other researchers^{5–7} who use different polynomial functions to represent joint trajectories. Their methods are either not suitable for on-line trajectory planning, or do not account for a mechanism's dynamic constraints, such as the maximum values for joint speed, acceleration, or jerk.

Red⁸ presents an adaptive optimal trajectory generator that uses dynamic S-curve velocity profiles to transition both Cartesian space and joint space. His algorithm is able to adapt to instantaneous changes in speed setting and path length.

This paper presents a motion recovery trajectory generator based on the adaptive S-curve. This algorithm is able to optimally plan a trajectory generator given a table of joints values. At each trajectory step, only one joint, the slowest joint, will dominate the motion. The remaining joints will be proportionally synchronized with the slow joint. The algorithm will continuously look ahead one step

to discover the next slow joint and make a slow joint transition, if necessary.

The trajectory is planned subject to the following constraints:

- Any joint acceleration and deceleration along the trajectory cannot exceed the max allowable value;
- Jerk is specified and limited;
- The move buffer is dynamically loaded with each new block of joint values and the algorithm is able to dynamically transition through these new joint value changes with position, speed, and acceleration continuity.

S-CURVE REVIEW

Trajectory generators have been extensively studied in the literature. There are several different types of trajectory generators that are studied and used in industry. Brady⁹ listed a few types of trajectory generators: bang-bang (also called trapezoidal profile), cosine, sine on ramp, and polynomial profiles. For more theoretical explanation of trajectory profiles, refer to references [10] and [11].

All trajectory generators share one common characteristic. There are certain boundary conditions specified for the trajectory generator. Usually, these boundary conditions are the initial and final positions, velocities, and even accelerations of the manipulator and joints. Based on the max allowable velocities, accelerations, and jerks, we need to fit a trajectory to meet all boundary conditions.

The trapezoidal profile is the simplest trajectory generator. It uses constant acceleration to move to a desired speed setting and requires the minimum traveling time. Unfortunately, as the speeds and accelerations of modern robots and NC machines continue to increase, this simple trajectory generator increases wear and stress on mechanisms, due to its inherent acceleration discontinuities at the beginning and end of the profile.

Polynomial profiles are becoming more and more popular because they provide position, velocity, acceleration, and even jerk continuity throughout the whole profile. Among all polynomial profiles, the S profile is probably the most popular. Referring to Figure 1, the S-curve is a cubic polynomial in position and a second-order polynomial in velocity. The second-order polynomial assumed for the S-curve velocity profile is

$$V(t) = b_0 + b_1 t + b_2 t^2 \quad (1)$$

Fig. 1. S-curve.

where the joint value $s(t)$ is found by integrating this equation. Taking the first and second derivative gives us the acceleration and constant jerk equations:

$$a(t) = b_1 + 2 b_2 t \quad (2)$$

$$j(t) = 2 b_2 = \text{const} \quad (3)$$

Figure 1 considers a pure S-curve (no linear transition). The rise motion ($0 \leq t \leq T$) can be divided into two periods – a concave period ($0 \leq t \leq T/2$) followed by a convex period ($T/2 \leq t \leq T$).

Considering the maximum jerk constraint j_m , we can derive the following equations for the concave and convex periods.

Concave period

$$s(t) = V_0 t + j_m t^3 / 6 \quad (4)$$

$$v(t) = V_0 + j_m t^2 / 2 \quad (5)$$

$$a(t) = j_m t \quad (6)$$

$$v(T/2) = V_h = (V_s + V_0) / 2 \quad (7)$$

$$s(T/2) = S_h = [V_0 + a_m^2 / (6 j_m)] a_m / j_m \quad (8)$$

Convex period

$$s(t) = V_h t + a_m t^2 / 2 - j_m t^3 / 6 \quad (9)$$

$$v(t) = V_h + a_m t - j_m t^2 / 2 \quad (10)$$

$$a(t) = a_m - j_m t \quad (11)$$

$$s(T/2) = [V_h + a_m^2 / (3 j_m)] a_m / j_m \quad (12)$$

For a detailed explanation of equations (4)–(12), please refer to reference [8].

TRAJECTORY PLANNING

Given a set of joint values, it is the function of the trajectory-planning algorithm to determine what paths to take and the trajectory time to pass through all given joint values.

One trajectory-planning approach is to plan an optimal trajectory for each individual joint without considering other joints. Obviously this approach is simple, but cannot guarantee that all joints will be able to pass each intermediate position simultaneously and come to a full stop at their end position at the same time. Some joints will reach their final values before others; these faster joints will remain idle at their final position until every other joint reaches its final destination. The path would deviate from the original Cartesian path.

The desired motion recovery trajectory algorithm requires that all joints pass each intermediate position simultaneously and come to a full stop at their destination at the same time. By doing that, it is guaranteed that the tool tip of the manipulator will be able to follow exactly its

original Cartesian path, before control failure, and remain collision-free.

The desired trajectory-planning approach uses the joint-interpolated method. Given a set of joints values, the trajectory-planning algorithm will fit a smooth curve $s_i(t)$ through all given intermediate points. The motion recovery trajectory requires that all joint trajectory profiles be coordinated so that they will pass each of their intermediate setpoints simultaneously and come to a full stop at their destination at the same time.

The adaptive S-curve trajectory generator will be used to interpolate the motion. Our algorithm only plans an adaptive S-curve for the slow joint. The remaining joints will be slowed proportional to the slow joint motion.

To start our algorithm, we must determine the initial slow joint. In our algorithm we arbitrarily assume joint 1 is the slow joint. This begins our method to determine the initial slow joint. Assume that the entry velocity and acceleration of joint 1 is v_0 and a_0 . We specify that the initial jerk value for joint 1 is the maximum allowable jerk j_m . Based on the known entry velocity, acceleration, and jerk values for joint 1, we can derive the following equations to calculate entry velocity, acceleration, and jerk for the remaining joints.

$$ratio_{j1} = \frac{s_{j2} - s_{j1}}{s_{12} - s_{11}} \quad (13)$$

$$V_j(0) = ratio_{j1} * v_0 \quad (14)$$

$$a_j(0) = ratio_{j1} * a_0 \quad (15)$$

$$j_j(0) = ratio_{j1} * j_m \quad (16)$$

where s_{j1} , s_{j2} , s_{11} , s_{12} are the given first and second joint values for joint j and joint 1.

For the j th joint, the entry joint velocity, acceleration, and jerk are subject to the following constraints:

$$\|v_j(0)\| < v_{jmax} \quad (17)$$

$$\|a_j(0)\| < a_{jmax} \quad (18)$$

$$\|j_j(0)\| < j_{jmax} \quad (19)$$

If one or more of the equations in (17)–(19) is violated, we replace joint 1 with joint j as the slow joint and load the corresponding joint j entry velocity and acceleration values into v_0 and a_0 . We repeat this procedure until we have examined all the independent joints. At this point, one of the joints will be chosen as the initial slow joint.

Our motion recovery trajectory algorithm uses the same trajectory rate that was applied when the system failure occurred. Suppose at any moment t , the j th joint is the slow joint. We can find its instantaneous position, velocity, acceleration and jerk from its S-curve trajectory. Suppose their values are $s_j(t)$, $s'_j(t)$, $s''_j(t)$, and $s'''_j(t)$. The j th joint value is an interpolated value that falls within the joint list: $s_{j1}, s_{j2}, \dots, s_{jn}$. At the moment t , $s_j(t)$ must fall within two joint values in the list. Suppose $s_j(t) \in [s_{jk}, s_{jk+1}]$. The following interpolation equations calculate the instanta-

neous position, velocity, acceleration, and jerk for the other joints noted here by i :

$$ratio_{ji} = \frac{s_{ik+1} - s_{ik}}{s_{jk+1} - s_{jk}} \quad (20)$$

$$s_i(t) = s_{ik} + ratio_{ji} [s_j(t) - s_{jk}] \quad (21)$$

$$s'_i(t) = ratio_{ji} s'_j(t) \quad (22)$$

$$s''_i(t) = ratio_{ji} s''_j(t) \quad (23)$$

$$s'''_i(t) = ratio_{ji} s'''_j(t) \quad (24)$$

For the i th joint, the instantaneous joint velocity, acceleration, and jerk are subject to the following constraints:

$$\|s'_i(t)\| < s'_{imax} \quad (25)$$

$$\|s''_i(t)\| < s''_{imax} \quad (26)$$

$$\|s'''_i(t)\| < s'''_{imax} \quad (27)$$

If at any moment, the i th joint velocity, acceleration, or jerk exceed their maximums, then one or more of the equations in (25)–(27) is no longer valid. At this moment, the i th joint becomes the slow joint and we have to slow the motion to not violate its constraints. When transitioning to a new slow joint, the trajectory algorithm plans a new adaptive S-curve for the new joint. The entry state can be determined by the last motion state proportionally determined from (21)–(24).

Suppose the new slow joint entry velocity and acceleration are V_0 and a_0 . We can then plan a new adaptive S-curve for the new slow joint based on the values of V_0 , a_0 , the remaining joint interpolation distance, and the transitional state.

Figure 2 shows the adaptive S-curve profile for the new slow joint, where V_0 and a_0 are the entry velocity and acceleration of joint i ; V_s is the set speed and a_m is the maximum allowable acceleration. Because a slow joint transition can happen in any profile period, Table I lists all possible transitional profiles necessary to reach the set speed. Depending on the transitional type we apply a different jerk sign for the adaptive S-curve. Sometimes the jerk is zero.

Fig. 2. Adaptive S-curve profile.

Table I. Transitional profile states.

Transitional State	State Number	Jerk
S_RISE_CONCAVE	1	j_m
S_RISE_LINEAR	2	0
S_RISE_CONVEX	3	$-j_m$
CONSTANT_SPEED	4	0
S_FALL_CONVEX	5	$-j_m$
S_FALL_LINEAR	6	0
S_FALL_CONCAVE	7	j_m
PARTIAL_CONCAVE_RISE	8	j_m
PARTIAL_CONVEX_RISE	9	$-j_m$
PARTIAL_CONVEX_FALL	10	$-j_m$
PARTIAL_CONCAVE_FALL	11	j_m

The adaptive S-curve is revised for the transitional entry values V_0 and a_0 , starting from the position polynomial equation:

$$s_i(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 \quad (28)$$

By knowing the entry position, velocity, and acceleration, we solve for the unknown coefficients in equation (28):

$$b_0 = s_i(0) \quad (29)$$

$$b_1 = V_0 \quad (30)$$

$$b_2 = a_0/2 \quad (31)$$

The time t is always adjusted to zero when beginning a new slow joint transition.

By knowing the transitional state, we select the right value for jerk from Table I, giving b_3 as:

$$b_3 = j/6 \quad (32)$$

where $j=0$, $+j_m$, or $-j_m$.

Rather than solve a very difficult closed-form solution for a feasible upper speed limit, the remaining distance and other motion conditions are used to limit the speed increase adaptively. The concept is to make a max jerk step to increase the entry speed towards the set speed. By knowing the remaining distance and other motion conditions, we only increase the speed if we can meet the terminal conditions. At any one moment, at least one of the joints uses its max jerk to change the speed. So this approach is incrementally optimal.

For a detailed explanation of the transitional states listed in Table I, refer to reference 8.

DEMONSTRATION

To illustrate the algorithm, we consider two examples. The first example is for a simple two revolute joint robot. Table II lists a sequence of joint values stored in the recovery array.

The velocity, acceleration and jerk constraints for this two revolute joint robot are listed in Table III.

The robot links begin at rest at the starting point, and come to full stop at the end point. This gives the initial and final boundary conditions for position and velocity.

Table II. Interpolation points for joint 1 & 2.

Step	Joint 1 (deg)	Joint 2 (deg)
1	-0.000018	0.000037
2	-0.018209	0.037477
3	-0.147414	0.298961
4	-0.494594	0.96371
5	-1.128572	2.040649
6	-2.166889	3.447231
7	-3.876482	4.892391
8	-6.533327	5.279459
9	-9.28923	3.65331
10	-11.625085	0.972056
11	-13.909417	-1.862173
12	-16.432699	-4.266361
13	-19.28596	-5.364723
14	-21.702635	-4.519405
15	-23.182045	-3.026524
16	-24.086106	-1.697279
17	-24.630057	-0.731742
18	-24.907297	-0.189215
19	-24.958384	-0.085431

To start our motion recovery trajectory generator algorithm, we initially determine the slow joint to plan our trajectory. Based on the procedure described in equations (13)–(19), joint 2 is chosen as the initial slow joint. After 0.275 seconds, joint 1 becomes the slow joint. At this moment, we transition the slow joint from joint 2 to joint 1 and plan a new adaptive S-curve trajectory for joint 1. At time 1.60 seconds, the slow joint transition takes place as joint 2 becomes the slow joint again. Based on the entry position, velocity, acceleration conditions and the remaining distance, we plan a new adaptive S-curve for joint 2.

Figure 3 shows the trajectories for joints 1 and 2. As we can see from this figure, the two joints start their motion at the same time and come to a full stop simultaneously. The position, velocity, and acceleration of two joints are continuous throughout the whole trajectory. The velocity and acceleration of both joints remain within the robot's velocity and acceleration constraints.

The second example is a 5-Axis mill that is used in DMAC (Direct Machine And Control) research. The mill has X, Y, Z translational joints which displace the tool, and A and C revolute joints placed on the base table.

Table IV lists a sequence of joint values through which all the joints must be interpolated. The velocity, acceleration and jerk constraints are listed in Table V.

Joint 1 is chosen as the initial slow joint based on the procedure outlined in equations (13)–(19). At time $T=0.43$ seconds, joint 3 becomes the slow joint based on joint dynamic constraints equations (25)–(27). At this instant, joint 3 has a velocity of 29.214059 mm/s and acceleration of

Table III. Two joint robot motion constraints.

Joint constraints	Joint 1	Joint 2
Velocity (deg/s)	400	500
Acceleration (deg/s ²)	1000	1000
Jerk (deg/s ³)	2000	2000

Fig. 3. Trajectories for joints 1 & 2.

– 270.308220 mm/s². Based on this entry velocity, acceleration and the remaining distance of 9.296675 mm, we plan a new adaptive S-profile for joint 3 starting at time 0.43 seconds and proportionally plan the motion for the other four joints.

Table IV. Interpolation points for joints X, Y, Z, A, and C.

X (mm)	Y (mm)	Z (mm)	A (rad)	C (rad)
0	0	0	–0.7	0
0.029	0.029	0	–0.7	0
0.236	0.236	0	–0.7	0
0.783	0.783	0	–0.7	0
1.644	1.644	0	–0.7	0
2.647	2.647	0	–0.7	0
3.628	3.628	0	–0.7	0
4.415	4.415	0	–0.7	0
4.853	4.853	0	–0.7	0
4.989	4.989	0	–0.7	0
5.002	5.001	–0.002	–0.7	0
5.072	5.061	–0.065	–0.701	–0.001
5.371	5.315	–0.31	–0.704	–0.007
6.115	5.928	–0.756	–0.712	–0.018
7.484	7.012	–1.113	–0.725	–0.038
9.343	8.436	–0.793	–0.744	–0.063
11.172	9.835	0.616	–0.765	–0.09
12.321	10.808	2.954	–0.787	–0.118
12.457	11.152	5.552	–0.808	–0.142
11.799	10.972	7.678	–0.827	–0.163
10.932	10.588	8.975	–0.84	–0.177
10.346	10.3	9.561	–0.848	–0.184
10.096	10.171	9.765	–0.851	–0.187
10.049	10.147	9.801	–0.851	–0.188

Table V. Motion constraints for 5-axis mill.

Joint constraints	X	Y	Z	A	C
Velocity (rad/s or mm/s)	200	200	200	10	10
Acceleration (rad/s ² or mm/s ²)	1000	1000	1000	100	100
Jerk (rad/s ³ or mm/s ³)	2000	2000	2000	300	300

At time $T=0.89$ seconds, the transition of slow joint takes place and joint 1 becomes the slow joint again. Based on the same method as above, we plan a new adaptive S-profile for joint 1 and proportionally synchronize the rest of joints with joint 1.

Figures 4 and 5 show the trajectories for joints X-Y-Z, and A-C, respectively. As we can see joints 1 and 2 start moving while joints 3, 4, and 5 are initially at rest. Joint 3 starts moving at $T=0.27$ seconds and joint 4 and 5 start moving at $T=0.28$ seconds. These results agree with the joint sequence in Table IV. All five joints stop simultaneously at the end position.

The velocity and acceleration of all joints are limited by the robot joints constraints. The position, velocity, and acceleration profiles are continuous throughout the whole trajectory.

The trajectory generator is run at a constant rate of 1,000Hz on both examples. Because this trajectory rate is very high, the spaces between any two interpolated joint values are very small. This guarantees the accuracy of the recovered Cartesian tool path.

Fig. 4. Trajectories for joints X, Y and Z.

Fig. 5. Trajectories for joints A and C.

CONCLUSIONS

Using an adaptive S-curve velocity profile, a new trajectory algorithm recovers the position, velocity, and acceleration joint sequence before a system crash. This is critical in salvaging parts that might normally be damaged when jogging the tool to a safe position after system restart.

To prove the feasibility of this motion recovery trajectory algorithm, two examples were considered. The first example demonstrates recovery for a two revolute joint robot while the second example applies the recovery methods to a more complex 5-axis mill. Both simulation and physical experiments using the DMAC open-controller demonstrated the

feasibility and usefulness of the recovery trajectory algorithms.

Experiments show that the tool could be recovered to follow its original Cartesian path before controller failure, using the same trajectory rate that originally generated the stored setpoints list (1KHz or higher).

References

1. C.C. Cook and C.Y. Ho, "The Application of Spline Functions to Trajectory Generation for Computer-Controlled Manipulators", **In: Computing Techniques for Robotics** (Chapman and Hall, 1st edition, 1985) pp. 102–110.
2. C.S. Lin, P.R. Chang and J.Y.S. Luh, "Formulation and Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots", *IEEE Trans. Automatic Control* **AC-28**(12), 1066–1073 (1983).
3. K.S. Wang, "B-Splines Joint Trajectory Planning", *Computers in Industry* **10**, 113–122 (1988).
4. R.V. Patel and S.E. Thompson, "Formulation of Joint Trajectories for Industrial Robots Using B-Splines", *IEEE Transactions on Industrial Electronics* **34**, 192–199 (1987).
5. A.A. Goldenberg and D.L. Lawrence, "End Effector Path Generation", *Journal of Dynamic Systems, Measurement, and Control* **108**, 158–162 (June, 1986).
6. C.S. Lin and P.R. Chang, "Joint Trajectory Generation of Mechanical Manipulators Using Spline Functions", *Automach International Conference* (Nov. 16–18, 1982) **MS 82-503**.
7. S. Chand and K.L. Doty, "On-Line Polynomial Trajectories for Robot Manipulators", *Int. J. Robotics Research* **4**, 38–48 (1985).
8. E. Red, "A dynamic optimal trajectory generator for Cartesian Path following", *Robotica* **18**, Part 5, 451–458 (2000).
9. M. Brady, *Trajectory Planning. Robot Motion Planning and Control* (The MIT Press, 3rd edition, 1983).
10. J.J. Craig, *Introduction to Robotics: Mechanics and Control* (Addison-Wesley, 2nd edition, 1989).
11. M. Brady, J.M. Hollerbach, T.L. Johnson and M.T. Mason, *Robot Motion* (The MIT Press, 3rd edition, 1983).