Robotica (2003) volume 21, pp. 531–540. © 2003 Cambridge University Press
DOI: 10.1017/S0263574702004915 Printed in the United Kingdom

Predictive joint motion limiting in robotic applications

Edward Red and Brian Fielding

Department of Mechanical Engineering, Brigham Young University, 435 CTB, P.O. Box 24201, Provo, Utah 84602–4201 (USA)

(Received in Final Form: September 20, 2002)

SUMMARY

Three joint space algorithms slow the Cartesian path motion when it appears that joint motion is approaching a joint, speed, or acceleration limit. All three algorithms use quadratic curve fitting to predict where the joint motion is heading, followed by a prediction as to how much time would elapse until a limit is reached.

If a joint motion limit is encountered in the *time-to-stop* the Cartesian motion, these algorithms reduce the Cartesian speed using pulsed speed settings so that the robot or machine tool will have the necessary time to come to a complete stop. The joint space velocity and acceleration control algorithms set the override Cartesian speed to either full or some reduced speed, several times a second. This allows the joints to reach, but not exceed, their maximum velocity and accelerations limit, while remaining within the physical joint limits.

KEYWORDS: Direct control; Joint motion limiting; Robotic applications.

1. INTRODUCTION

A new architecture for controlling mechanisms, such as robots and machine tools, has been developed at Brigham Young University.[1–3] Initially directed towards *Direct Machining And Control* (DMAC), the acronym context is easily broadened to *direct mechanism applications control*, extending to all mechanisms, including robotics.

By using advances in digital networking, digital power electronics (e.g. PWM), and PC-based real-time operating systems, DMAC can directly control a mechanism from a process planning application, such as Computer-Aided Manufacturing (CAM), including robotic simulation software such as RobCad, IGRIP, or CODE.

DMAC's architectural goal is to make mechanisms look like **part printers** connected to design and process application software. This approach parallels the document printing paradigm that we use today. We simply select the **Print** option and the screen documents issue from the printer. Most of the printing complexity is hidden from the user as a device driver.

Direct control uses a single PC to integrate the process planning application, running under Windows on one processor, with the DMAC controller running under a real-time OS on a second processor. The controller's motion planning and trajectory generation subsystem, along with the servo-control loops, are configured in software on the PC.

DMAC eliminates most analog control hardware, including the servocard and A/D and D/A I/O interfaces. What remains are an integrated digital current/torque amplifier, and motor-encoder combination, interfaced to the PC through a digital high speed control network that functions dually as a digital communications interface and servo-control loop.

1.1. What this architecture means?

Direct control eliminates the need to pre-process and sometimes tessellate moves into simpler move types using standardized control languages and files, such as APT, CL, and M&G. This quasi-static control environment requires several process steps between the process application and actual mechanism control. The advantage of the contemporary process planning environment is that moves can be closely examined for motion limiting and be adjusted accordingly. The disadvantage is that the mechanism's inverse kinematics must be available to the process planning software.

DMAC replaces this quasi-static process with one that is dynamic and direct, not requiring kinematics in the CAM application. Both simple and complex moves (e.g. NURBS paths) are directly passed to the DMAC controller in their native mathematical form. But moving along complex paths without pre-processing the path can lead to motion limiting in joint space.

This paper presents several algorithms designed to dynamically adjust the motion in Cartesian space, on the controller, when joint motion approaches a physical, speed or acceleration limit.

2. CARTESIAN VERSUS JOINT SPACE

When a machine tool or robot end-effector is moved along a pre-defined Cartesian path, the mechanism joints must be coordinated to achieve this motion. The mapping between the Cartesian tool frame pose (position and orientation) and joint configuration is achieved through inverse kinematics. The mapping between Cartesian velocity and joint speeds is achieved through a familiar matrix relationship involving the Jacobian: $\dot{\boldsymbol{v}} = \boldsymbol{J}\,\dot{\boldsymbol{q}}$ where $\dot{\boldsymbol{v}}$ is the vector of Cartesian linear and angular velocity components, and $\dot{\boldsymbol{q}}$ is the vector of joint rates.J, the Jacobian, varies as a function of joint values.

Derivatives of the $\dot{\boldsymbol{v}} = \boldsymbol{J}\,\dot{\boldsymbol{q}}$ equation can be used to map Cartesian accelerations into joint accelerations, but a numerical derivative of the joint speeds is often satisfactory.

At times, a commanded move in Cartesian space may cause a joint to exceed a physical limit, speed limit, or acceleration limit. When a joint physical limit is reached, the robot physically cannot travel to the commanded target frame. And mechanism breakage or misalignment may occur.

If a joint is commanded to travel to speeds or accelerations that exceed the motor limits, the motor will saturate, and path deviation may occur.

This paper presents three algorithms to modify the Cartesian motion when it appears that joint motion is approaching a limit. All three algorithms have a similar foundation as discussed in later sections.

3. MOTION LIMITING ALGORITHMS

The algorithms in this paper do not have to understand the mathematical relationships between Cartesian motion and joint motion for a particular mechanism. The method is thus called *predictive joint motion limiting*. These algorithms are mechanism independent.

This black box approach contrasts to other approaches. Researchers typically use mathematical relationships between Cartesian and joint space for their motion limiting avoidance algorithms. These approaches require mechanism-specific models for the Jacobian, inverse Jacobian and derivatives, dynamical equations, and inverse kinematics.

Chiacchio[4] develops methods to avoid joint velocity limits using first-order inverse kinematics algorithms. This technique guarantees tracking of the desired end-effector path. The algorithms slow down the task-space trajectory when joint velocity limits are encountered. A virtual time is introduced so that the time law can be modified through a time warp, thus satisfying the velocity constraints.

Chiaverini[5] expands Chiacchio's approach to avoid joint acceleration limits. A second-order inverse kinematics algorithm ensures that both joint velocity and acceleration bounds are not violated. The acceleration is mapped using the Jacobian and other feedback correction terms.

A motor is limited in the torque it can output. The torque required to achieve a desired joint speed depends on the mechanism configuration. Zlajpah[6] develops algorithms to ensure joint speeds stay within the given bounds, while maintaining the motor torques within acceptable limits. Mechanism dynamical equations are used to map Cartesian motion to joint torques, subject to task constraints (in speed and torque).

Ohishi[7] considers three algorithms to control torque saturation: (1) a joint space torque limiting algorithm; (2) a Cartesian motion control adjustment algorithm; and (3) an adjustment algorithm of motion reference in Cartesian space. Path tracking is smooth, with little torque saturation.

Robot redundancy is another way to avoid motion limiting, particularly for mechanisms that exhibit troubling singularities. Since this paper focuses on conventional mechanisms, only two papers are considered.

The minimum number of coordinates needed to define any pose in space is six (3 position coordinates, 3 orientation coordinates). If a robot has more than six degrees of freedom, it is redundant. However, some Cartesian tasks require fewer than six coordinates to describe a tool pose. In such cases even normal mechanisms can be redundant. Consider, for example, an axisymmetric ball endmill used in machining, or a waterjet cutting nozzle.

Chan[8] avoids joint limiting by a weighted least-norm (WLN) solution. This solution is compared to the gradient projection method (GPM), and both methods are implemented on a 7-DOF robotic manipulator. A weighting scale approaches infinity at the joint limits. The WLN scheme reduces the joint motion cost, and dampens joint motion only in the direction of the limits, as opposed to the GPM that maximizes the distance from the joint limits. Chan shows that the WLN joint trajectories are oscillation free and that joint limit avoidance is guaranteed.

Park[9] uses redundancy to reconstruct the inverse kinematics routines to ensure task motion errors do not occur. A method is proposed to overcome hardware limitations, which is called joint velocity reconstruction. The idea behind this method is that the errors induced by hardware limitations can be removed by adjusting the velocity of the joints that have no limit.

In commercial robots and machine tools, the practical approach to motion limiting is to use soft limits as a buffer zone. The mechanism is instantly stopped when any soft limit is violated. The buffer width limits the top speed allowed for the mechanism, and also reduces the practical working volume for the mechanism.

4. PREDICTIVE JOINT MOTION LIMITING

As a mechanism is moved along a Cartesian path, DMAC applies kinematics routines to return instantaneous joint values and rates at each trajectory step. The predictive routines described in this paper only need the current motion state in both Cartesian and joint space, plus several previous motion states to determine the quadratic coefficients for the predictive equations.

At each trajectory step the joint motion is examined for proximity to the motion limits. The joint motion is curve fitted into a quadratic equation by using three data points: the current value and two previous values. Using the quadratic equation, a prediction is made as to how much time would elapse before a limit is reached. These predicted times are stored and compared to the time to stop the Cartesian move, given its current motion state. Time is the invariant between the two spaces.

To understand the implementation details we must first consider the velocity profile (trajectory) generator used in DMAC. Then we consider the predictive methods applied to motion limiting.

4.1. S-curve profile generator

The velocity profile generator used in DMAC was developed by Red.[10] Applying constant jerk transitions between the constant acceleration and deceleration periods of the trajectory, the trajectory will optimally transition to the desired speed setting. This profile generator is an integral member of the predictive algorithms, because it

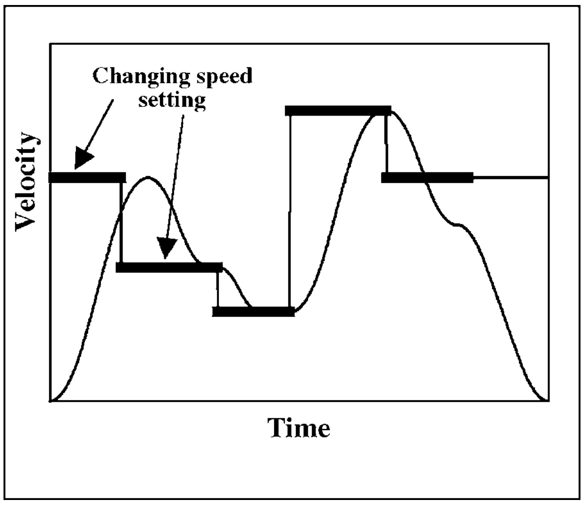

Fig. 1. S-curve adapting to speed changes.

allows efficient *time-to-stop* calculations to be made in Cartesian space.

As we will see, the predictive algorithms pulse the Cartesian speed setting when motion limits are approached. Figure 1 shows how the S-curve profile generator readily adapts to these pulses.

4.2. Time-to-stop Cartesian moves

The S-curve cannot reach the desired max rise and fall accelerations if the desired speed change is small. An intermediate acceleration must be determined, that Red describes as an *acceleration transition*. This case points out that there are an interesting set of conditions that determine the time to stop in Cartesian space. These cases depend on the entry acceleration, necessary speed changes, remaining move distance, and the jerk and acceleration allowables for the mechanism.

Tables I and II summarize the conditions and calculations as a function of the entry speed and acceleration. The Table I equations depend on calculating an intermediate speed when the entry acceleration is positive: $A_0 \geq 0$.

The intermediate speed is the minimum speed obtained when the jerk is immediately applied to reduce the current acceleration to zero – see Figure 2. It is determined by first computing the time to change the current acceleration value from A_0 to zero, then applying constant jerk transitions to

Table I. Time-to-stop (t_{total}) when $A_0 \geq 0$.

If	Then	And
$V_{\text{int}} > \frac{A_f^2}{J_m}$	$t_{fall} = \left\lvert \frac{V_{\text{int}}}{A_f} + \frac{A_f}{J_m} \right\rvert$	$t_{total} = \left\lvert \frac{V_{\text{int}}}{A_f} + \frac{A_f}{J_m} T \right\rvert + \frac{A_0}{J_m}$
$V_{\text{int}} = \frac{A_f^2}{J_m}$	$t_{fall} = \left\lvert \frac{2A_f}{J_m} \right\rvert$	$t_{total} = \left\lvert \frac{2A_f}{J_m} \right\rvert + \frac{A_0}{J_m}$
$V_{\text{int}} < \frac{A_f^2}{J_m}$	$t_{fall} = 2\sqrt{\frac{V_{\text{int}}}{J_m}}$	$t_{total} = 2\sqrt{\frac{V_{\text{int}}}{J_m}} + \frac{A_0}{J_m}$

Table II. Time-to-stop (t_{total}) when $A_0 < 0$.

First calculate: $V_1 = \frac{A_f^2}{2J_m}$, and $V_2 = V_0 + \frac{A_0^2 - A_f^2}{2J_m}$

If enter on S_FALL_CONVEX, *time-to-stop* $= t_{total}$ as follows.
If enter on S_FALL_CONCAVE, replace J_m with $-J_m$ for *time-to-stop*.

If	Then
$V_2 > V_1$	$t_{total} = \frac{A_0}{J_m} - \frac{3A_f}{2J_m} - \frac{V_2}{A_f}$
$V_2 = V_1$	$t_{total} = \frac{A_0 - 2A_f}{J_m}$
$V_2 < V_1$	$t_{total} = 2\sqrt{\frac{V_0}{J_m} + \frac{A_0^2}{2J_m^2}} + \frac{A_0}{J_m}$

If enter on S_FALL_LIN, then *time-to-stop* follows.

$$t_{total} = -\frac{V_0}{A_f} - \frac{A_f}{2J_m}$$

reduce the speed and acceleration to zero. A constant jerk (J_m) transition reduces the acceleration to zero in the rise time:

$$t_{rise} = \frac{A_0}{J_m} \tag{1}$$

Substituting this time into the speed equation $V_{int} = V_0 + A_0 t - J_m t^2/2$ with $t = t_{rise}$ yields the intermediate speed:

$$V_{int} = V_0 - \frac{A_0^2}{2J_m} \tag{2}$$

Given V_{int}, Table I presents the *time-to-stop* the Cartesian move when $A_0 \geq 0$. For the special case of $A_0 = 0$, we apply the Table I conditions letting $V_{int} = V_0$.

Table II lists the conditions for $A_0 < 0$. These conditions are again functions of the motion conditions and described in Fielding's[11] thesis.

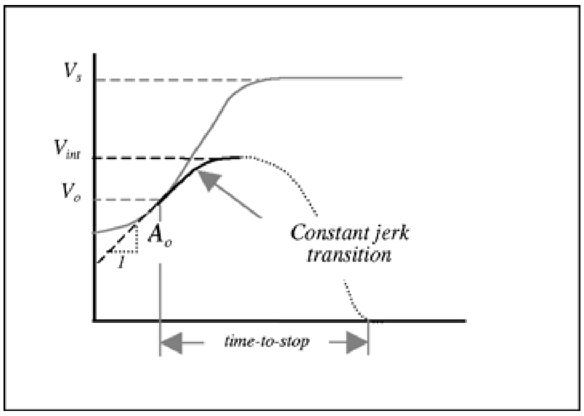

Fig. 2. Constant jerk transition to V_{int}.

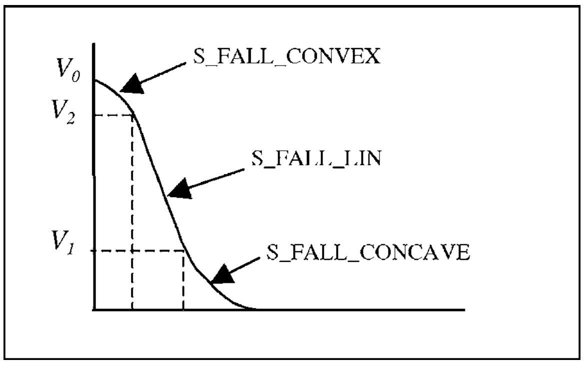

Fig. 3. S-curve fall periods.

To illustrate the Table I and II conditions, we consider the case where the entry acceleration is $A_0<0$. We assume that the remaining distance has already caused us to begin stopping the move. We determine two speeds, V_1 and V_2, to determine whether the deceleration transition requires at most three deceleration transitions, identified as S_FALL-_CONVEX, S_FALL_LIN, and S_FALL_CONCAVE. S_FALL-_CONVEX only applies if the entry conditions (V_0, A_0) place the motion state in the convex fall portion – see Figure 3. S_FALL_LIN is for entry conditions on the linear portion ($A_0=A_f$). S_FALL_CONCAVE is the final concave deceleration to zero speed and zero acceleration.

V_2 and V_1 are determined respectively by applying a constant jerk to first change A_0 to the maximum allowed deceleration, A_f, then to bring A_f back to zero acceleration at the end of the move. If $V_2>V_1$, a linear, constant deceleration period is inserted between V_2 and V_1.

If we assume that we enter on the S_FALL_CONVEX transition, the time to change from A_0 to A_f is:

$$t_{conv}=\frac{A_0-A_f}{J_m} \tag{3}$$

Using this time we get V_2:

$$S\_FALL\_CONVEX\text{: } V_2=V_0+\frac{A_0^2-A_f^2}{2J_m} \tag{4}$$

Similarly, we get V_1:

$$S\_FALL\_CONCAVE\text{: } V_1=\frac{A_f^2}{2J_m} \tag{5}$$

Now let us assume that the conditions are such that $V_2>V_1$. We insert a constant deceleration (linear) period to reduce the speed from to V_2 to V_1. The period from V_0 to V_2 is S_FALL_CONVEX. The period from V_2 to V_1 is S_FALL_LIN. The period from V_1 to zero speed is S_FALL_CONCAVE. The cumulative transition time is shown in Table II as

$$t_{total}=\frac{A_0}{J_m}-\frac{3A_f}{2J_m}-\frac{V_2}{A_f} \tag{6}$$

4.3. Quadratic time-to-limits

To predict joint motion directional properties, a curve fit is performed. Quadratic curve fitting uses invariant time to provide a localized map between motion in Cartesian space and the joint space motion response.

A quadratic curve requires three points to solve for the unknown coefficients a, b, and c:

$$y=at^2+bt+c \tag{7}$$

Joint values, speeds, and accelerations for each joint are curve fitted using the current value and two previous values, stored in three $n\times 3$ arrays, and where n is the number of joints.

We let P represent a joint vector from this array, so that $P[0]$, $P[1]$, and $P[2]$ represent the present, previous, and last data values stored for a particular joint. Applying these values we can solve for a, b, and c as:

$$c=P[0] \tag{8}$$

$$b=(-1.5\,P[0]+2\,P[1]-0.5\,P[2])/T_s \tag{9}$$

$$a=(0.5\,P[0]-P[1]+0.5\,P[2])/T_s^2, \tag{10}$$

where T_s is the trajectory step size.

The coefficients a, b, and c are used in a quadratic solution to generate, for each joint physical, speed, or acceleration limit, the *time-to-limit*(s) solution points t_{ij}, where $i=1$ to n and $j=1$, 2, or 3 for the three stored points (present plus two previous values). See Figure 4.

The minimum-real-positive time is checked against the *time-to-stop*. If any joint physical/speed/acceleration *time-to-limits* exceeds the *time-to-stop*, the Cartesian speed setting is pulsed to a value less than the desired speed.

In its most primitive form, the algorithm pulses the Cartesian speed to zero, and then resets it to the desired (or reduced proximity) speed once the *time-to-stop* is less than the intersection time for all joints. In some adaptations we reset the Cartesian speed to a fraction of the desired speed based on proximity to the joint boundaries – see the section **Five-Axis Machine Tool Example**.

4.4. Joint limits example

In the first example we consider a simple two link mechanism with revolute joints. The commanded Cartesian path is a straight line for the Link 2 tip, as shown in Figure 5 by the *start* and *end* points. The tip speed is referred to as the TCF (tool control frame) speed. Speed and acceleration limiting is not considered in this example.

A physical joint limit is imposed on Link 1 at 90°. Without Cartesian predictive speed limiting, Figure 5 shows

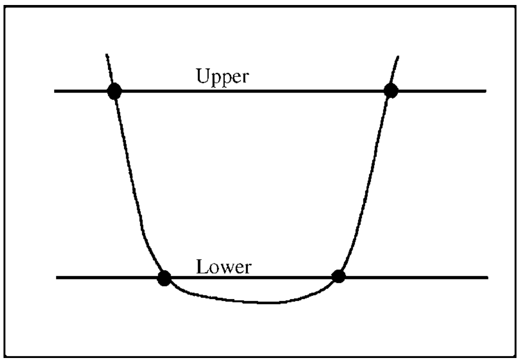

Fig. 4. Quadratic intersection points (shown for physical limits).

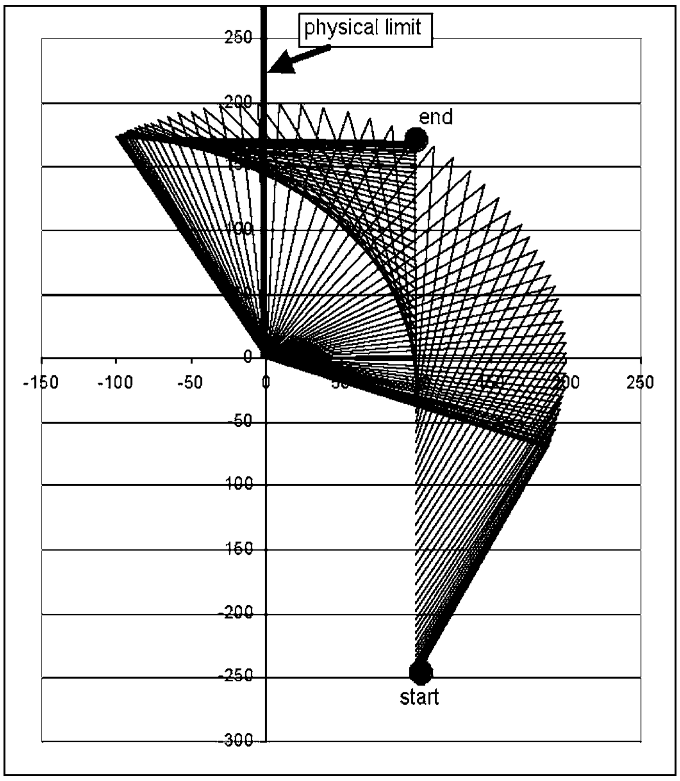

Fig. 5. No predictive limiting.

that Link 1 violates this physical limit. Link 1 speed is about 360° per second when its physical limit is violated.

Figure 6 shows the motion of the robot with the predictive algorithm applied. The mechanism is not allowed to travel where commanded; thus, it stops.

Figure 7 shows the TCF set speed (800 mm/s), and the actual Cartesian path speed as the set speed is reduced in proximity to the joint limit. The S-curve smoothly slows the TCF path speed to zero.

4.5. *Speed limiting*

Fielding[11] notes that predictive speed limiting can be treated more moderately than physical limits. He multiplies a scaling factor to the *time-to-limits* value so that the Cartesian speed is only pulsed down when very near a joint speed limit. Scaling factors of 1.5 to 2 generate smooth speed reductions in Cartesian space when a joint approaches its speed limit.

4.6. *Acceleration limiting*

Joint accelerations are inherently more noisy and oscillatory than speed, which is correspondingly more oscillatory than position. This occurs because the S transitions adjust the acceleration many times to move through the path distance while maintaining the path speed. Figure 8 shows the joint acceleration response for the two link mechanism when velocity predictive control is applied to avoid joint speed limiting. With such response characteristics the *time-to-limits* predictions for acceleration limiting will be inaccurate.

The primary difference in the predictive acceleration algorithm is that an exponentially weighted moving average is used to filter out rapid fluctuations in the joint accelerations. This results in smooth, filtered data so that curve fitting can reasonably predict the joint acceleration response.

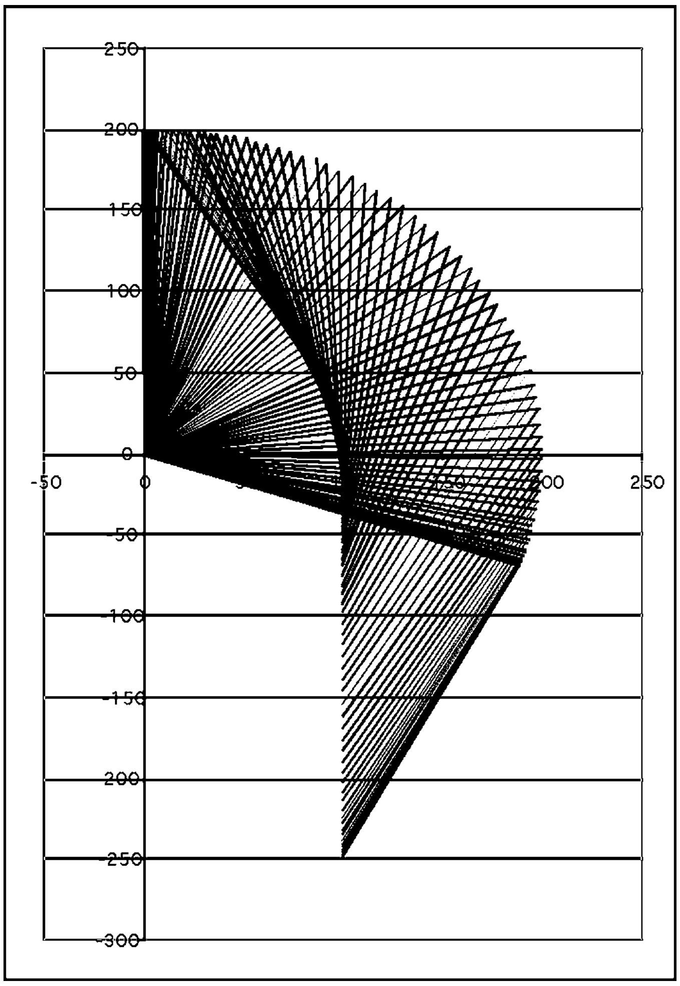

Fig. 6. Straight line, with predictive control.

There are several modes of filtering out noise, but the one implemented[12] is called *exponentially weighted moving averages*. It is similar to a simple moving average filter, except that it gives a higher weighting to the most recent points. More data points will cause a greater filtering effect, but increase the prediction lag.

This filter was chosen because of the simple mathematical computations involved: one subtraction, one addition, and two multiplications at each time step:

$$\bar{x}_k = \alpha \bar{x}_{k-1} + (1-\alpha)x_k \tag{11}$$

$\bar{x}_k$ is the current exponentially weighted moving average; $\bar{x}_{k-1}$ is the average on the previous step; and α, the filtering factor, is a function of the total number of data points, m:

$$\alpha = \frac{m}{m+1} \tag{12}$$

Equation (11) is converted to a standard difference equation to determine a digital transfer function by the equation:

$$y_m = \alpha \cdot y_{m-1} + (1-\alpha)x_m \tag{13}$$

Converting this transfer function into a continuous time domain transfer function yields a simple first-order transfer function:

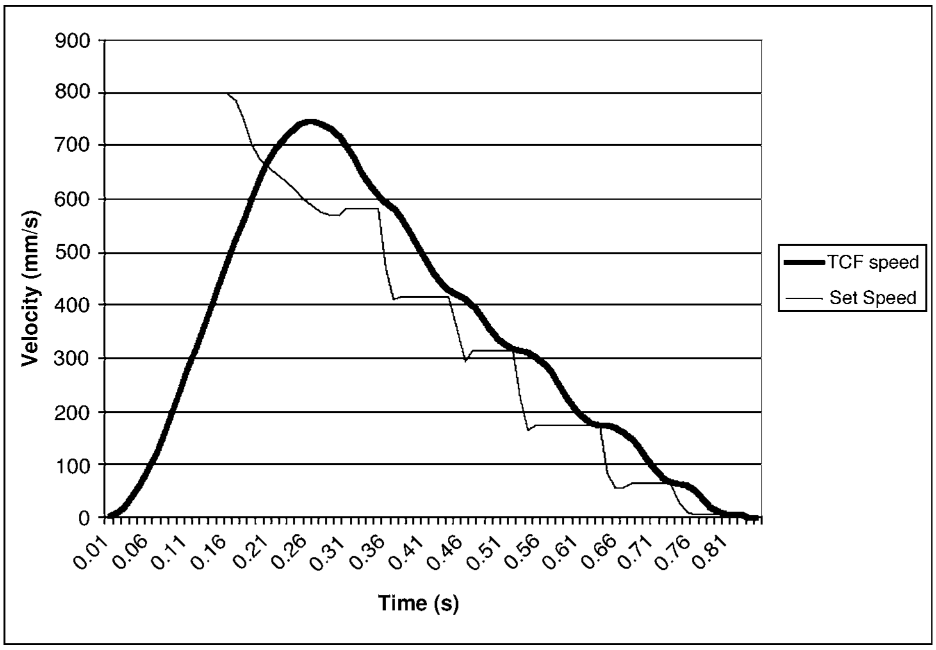

Fig. 7. Tip speed slowing by predictive pulsing.

$$H(s)=\frac{1}{\tau s+1} \tag{14}$$

where τ, the time constant, is a function of the number of data points used in the filter and the trajectory step size T_s:

$$\tau=m\,T_s \tag{15}$$

The magnitude of (14) is:

$$Mag(H(s))=\frac{1}{\sqrt{(\tau\omega)^2+1}} \tag{16}$$

At the cutoff frequency (ω_c), the magnitude is 70.7% of the input and has a phase lag of 45. Anything beyond this frequency begins to be filtered out. The higher the frequency, the less effect it will have on output.

The question that is raised is what should the cutoff frequency be? Fortunately, the S-curve acceleration response time when changing from zero to full acceleration can be estimated from equation (1). A reasonable S-curve acceleration frequency is the inverse.

Using a conservative 95% magnitude for the transfer function magnitude, we solve for τ from:

$$\tau=\frac{A_r}{J_m}\sqrt{\frac{1-0.95^2}{0.95^2}} \tag{17}$$

giving

$$m=\frac{A_r}{T_s\,J_m}\sqrt{\frac{1-0.95^2}{0.95^2}} \tag{18}$$

The number of data points used in the exponentially weighted moving average filter is then a function of the

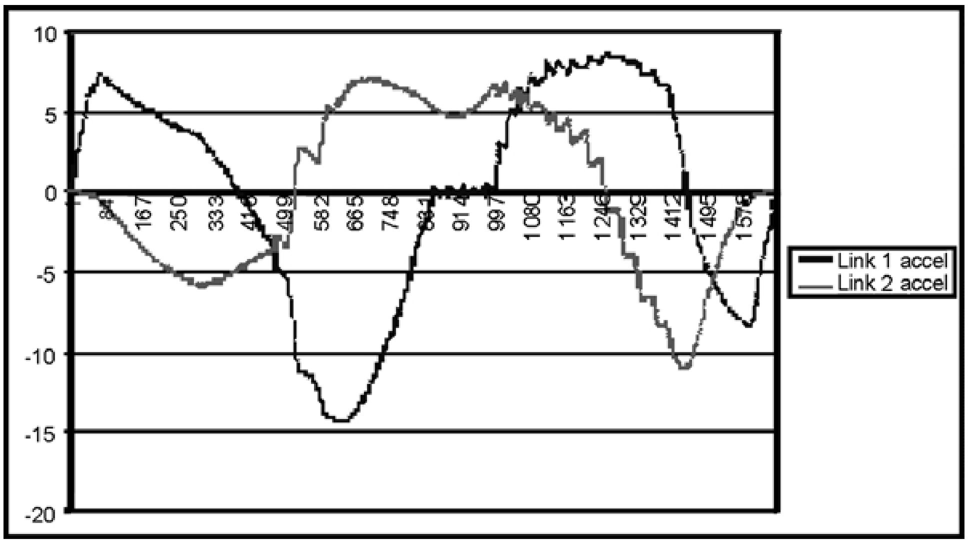

Fig. 8. Joint acceleration oscillations.

maximum rise acceleration, maximum jerk, and trajectory step size. Using the filter for the Figure 8 gives the response shown in Figure 9.

4.7. Acceleration predictive algorithm

Fielding[11] notes that predictive acceleration limiting can be treated more moderately than physical limits. Like the speed algorithm, he multiplies a scaling factor to the time-to-limits value so that the Cartesian speed is only pulsed down when near a joint acceleration limit. Tests showed that a scaling factor of 3 generates smooth speed reductions in Cartesian space when a joint approaches its acceleration limit.

4.8. Predictive acceleration example

In this example we choose a maximum rise/fall acceleration of 3000 mm/s^2, jerk of 40,000 mm/s^3, and trajectory step size of 1 millisecond. The number of filtering data points is calculated as approximately 25.

Figure 10 shows the two link mechanism tracing a circular path at a speed of 800 mm/s. Both speed and acceleration predictive limiting is being applied with the joint speed and acceleration limits being set at 4 rad/s and 40 rad/s^2.

Figures 11 and 12 show the joint speed and acceleration responses when the speed is pulsed near the limits. Note that both speed and acceleration values remain within their limits.

5. SINGULARITIES

Robots and 5-axis machine tools may have singular configurations where a nominal Cartesian speed results in unexpected large joint rates. The singularity issue has not been directly addressed by these algorithms. Nevertheless, these algorithms will attempt to slow down the Cartesian velocity when a singularity is encountered, but no guarantee can be made that the joint rates will be kept within their limits.

Y. Nakamura[13] presents a method for inverse kinematics with singularity robustness. He also presents a "manipulability" function (originally derived by T. Yoshikawa),[14] which equals zero at a singular point and becomes larger the further distance away from the singular point. These equations might provide a foundation for implementing a singularity robust algorithm into these motion limiting algorithms.

6. FIVE-AXIS MACHINE TOOL EXAMPLE

We now consider a practical application of the predictive limiting algorithms on a 5-axis (X-Y-Z-A-C) machine tool. Table 3 lists the physical properties for the mechanism. Units are in mm (X-Y-Z) and degrees (A-C), with rates in per s, per s^2, or per s^3.

The predictive algorithms were simplified in the case of speed and/or acceleration limiting, only pulsing the Cartesian speed lower when in close proximity to these limits. The joint physical limits algorithm was implemented as

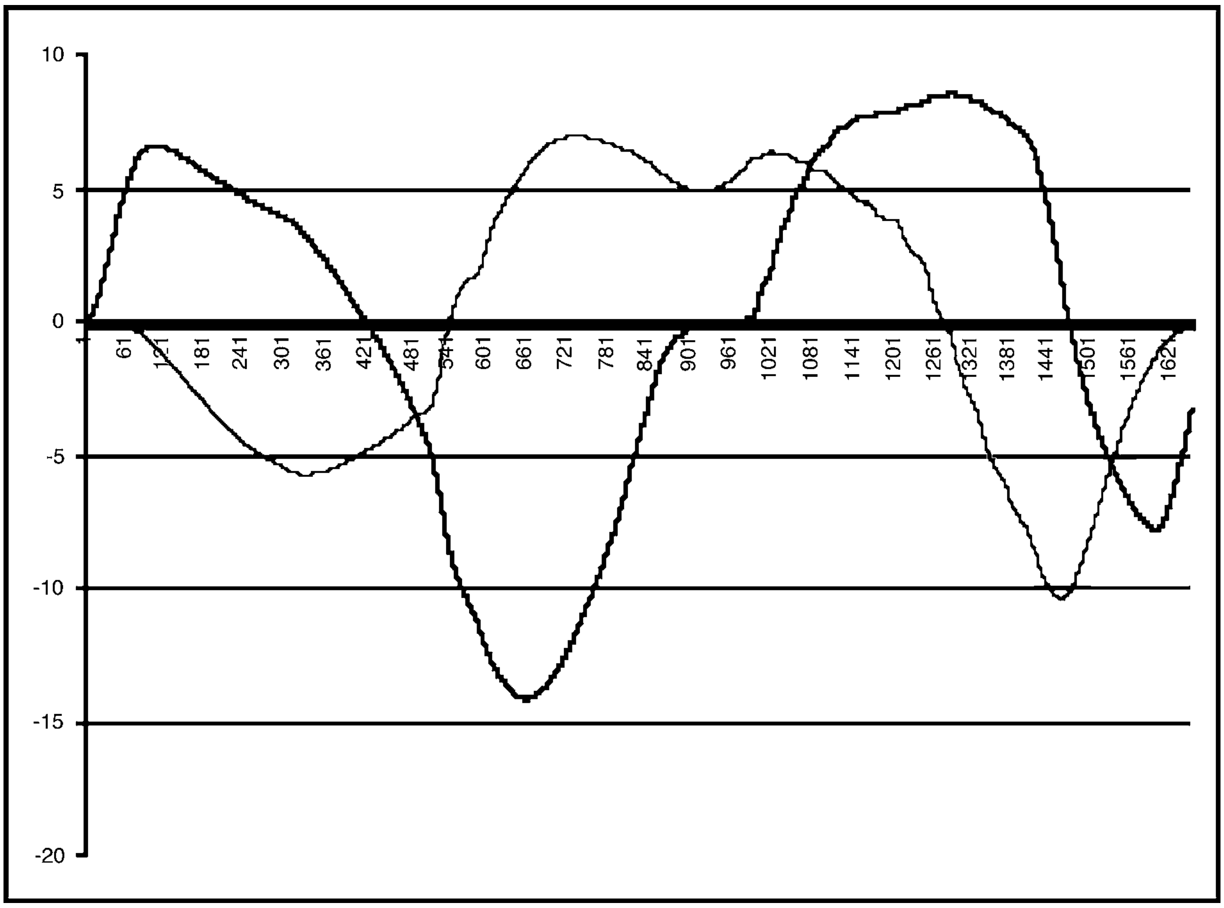

Fig. 9. Acceleration response with filter.

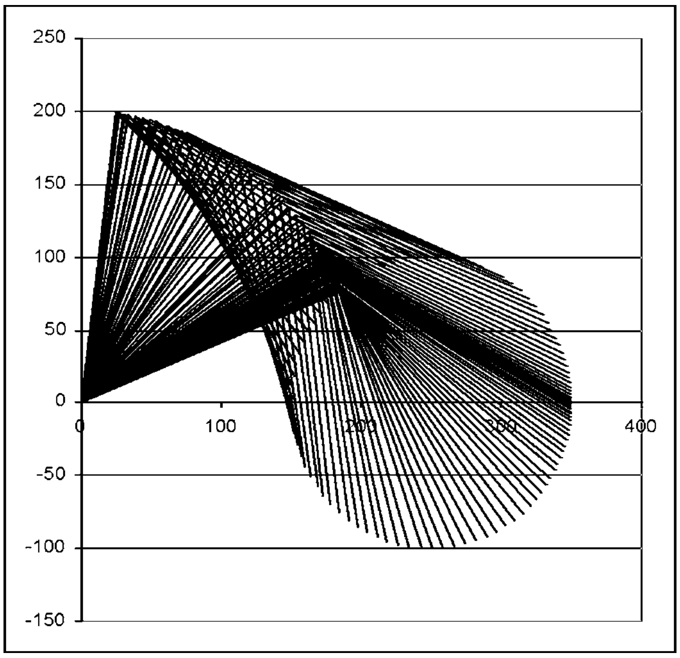

Fig. 10. Following a circular path.

previously described, except that the pulsed feedrate magnitude was linearly reduced in proximity to the joint limit boundaries.

A safety buffer zone (soft limits) of 6.6 mm was implemented for the X, Y, and Z axes. In an abort situation occurred by violating a soft limit, a trapezoidal deceleration would limit the maximum possible Cartesian speed (feedrate) as determined by:

$$V_{max} = \sqrt{2A_f L} \tag{19}$$

Using the values $A_f = 3000\ \text{mm/s}^2$ and $L = 6.6\ \text{mm}$, we calculate V_{max} to be about 200 mm/s, a normal maximum feedrate for a mechanism with this working volume.

We apply a sequence of linear moves with a speed setting of 300 mm/s and purposely violate the joint limits, also watching the joint speed and acceleration rates for possible limiting conditions. It is not important to detail the move sequence; rather observe the joint performances in Figures 13 and 14.

Figure 13 shows the X-Y-Z joint speed response for the planned sequence of moves, in the case where soft limits are active. Note that this 5-axis machine tool would probably be sold with a feedrate capability advertised as 200 mm/s. But the predictive limiting algorithms allow the machine to maintain speeds near 280 mm/s when not in close proximity to its physical limits. This is a performance boost for such mechanisms.

Figure 14 shows the joint rate performance when soft limits are not active. The predictive algorithms will slow the Cartesian motion to near zero speed when approaching the physical limits, regardless of the commanded speed. This may or may not be acceptable, but does err on the side of caution. Again, the moves can run at higher speeds when away from the physical limits.

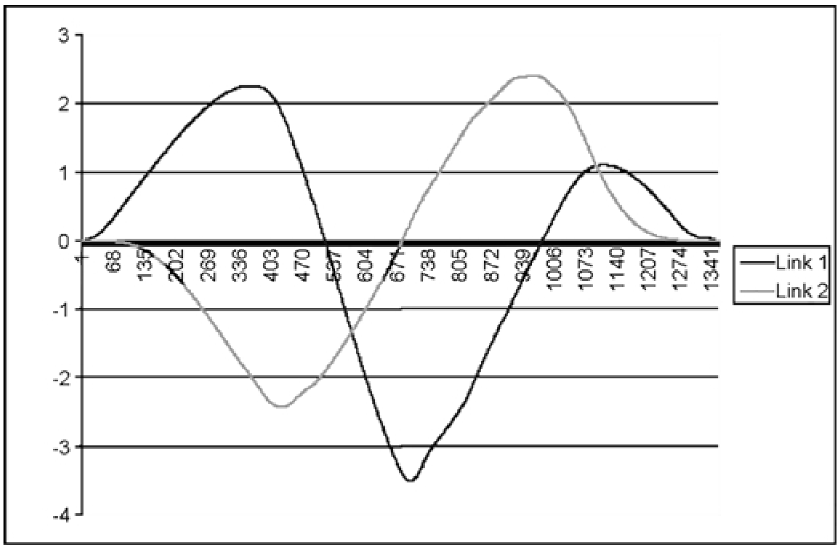

Fig. 11. Joint speeds with limits at 4 rad/s.

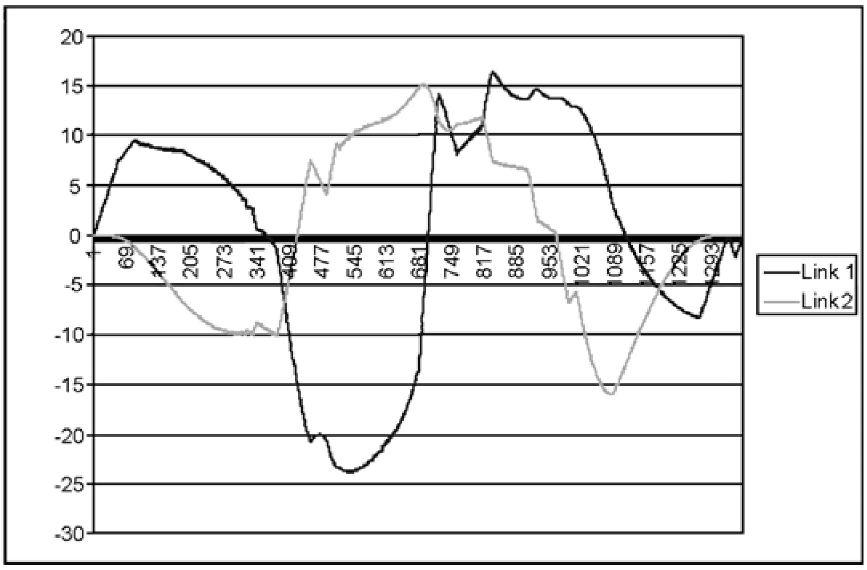

Fig. 12. Acceleration response with speed & accel limiting (accel limit at 40 rad/s^2).

Table III. X-Y-Z-A-C motion example parameters.

Axis	Lower Limit	Upper Limit	Speed Limit	Accel Limit	Jerk
X	− 150	150	300	3000	30000
Y	− 150	150	300	3000	30000
Z	− 100	100	300	3000	30000
A	− 145	145	10	100	3000
C	Infinite		10	100	3000

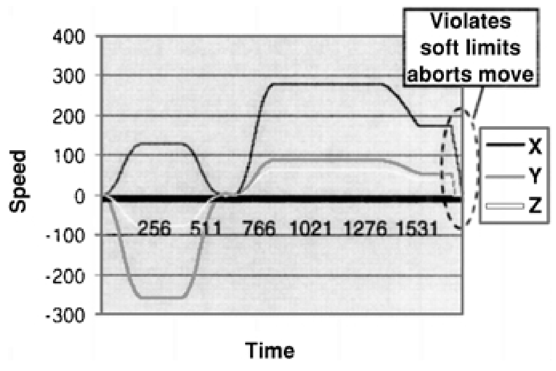

Fig. 13. X-Y-Z axis speeds for commanded feedrate of 300 mm/s and 7 mm buffer applied to limits.

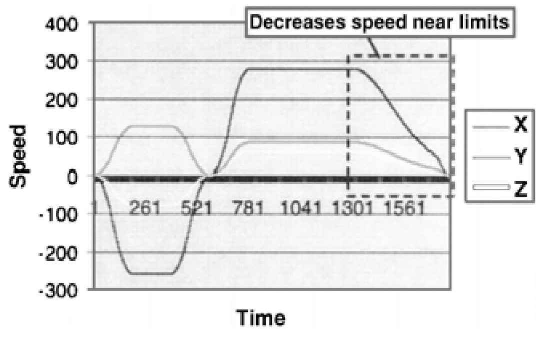

Fig. 14. X-Y-Z axis speeds for commanded feedrate of 300 mm/s and no safety buffer.

CONCLUSION

Time, the invariant between Cartesian and joint space, is used to predict *time-to-limits* in joint space, so that it does not exceed the *time-to-stop* in Cartesian space. The predictive motion limiting algorithms prove quite effective for protecting mechanisms from violating their physical and rate limits. More importantly, they are simple and need not understand the kinematics and dynamical equations of motion for a mechanism.

On a few occasions, the mechanism could not respond quick enough to keep the joint motion within the limits. This, however, only occurred when settings were unrealistic. For example, when the maximum jerk was set extremely low, the S-curve trajectory generator exhibits a slow response. In such unusual applications the algorithm could not always keep the joint speeds within their limits at all times.

Mechanism singularities could pose a problem for the algorithms, but practical applications will design processes to avoid singularities.

The predictive algorithm could be applied in a number of ways. It could be used in a dry run to verify that joint limits are not violated. Then it could be used during actual processes with only speed and acceleration limiting checked. Or the algorithm could always be on to ensure that motion limits are never violated.

References

1. C.P. Bassett, C.G. Jensen, J. Bosley, Y. Luo, W.E. Red and M. Evans, "Direct Machining Architectures Using CAD-CAM Generative Methods", *Proceedings of the IASTED International Conference Control and Applications* (May 24–27, 2000) pp. 287–294.
2. C.P. Bassett, et. al., "Direct Machining: A New Paradigm for Machining Data Transfer", *ASME 5th Design for Manufacture Conference*, Baltimore, Maryland (Sept. 10–13, 2000) paper # DFM-1498.
3. W.E. Red, M. Evans, C.G. Jensen, J. Bosley and Y. Luo, "Motion Planning and Trajectory Control of a Direct Machining Application", *Proceedings of the IASTED International Conference Control and Applications* (May 24–27, 2000) pp. 484–489.

4. P. Chiacchio and S. Chiaverini, "Coping With Joint Velocity Limits In First-Order Inverse Kinematics Algorithms: Analysis And Real-Time Implementation", *Robotica* **13**, Part 5, 515–519 (1995).
5. S. Chiaverini and G. Fusco, "A New Inverse Kinematics Algorithm With Path Tracking Capability Under Velocity and Acceleration Constraints", *Proceedings of the 38th Conference on Decision and Control* (May, 1999) pp. 2064–2069.
6. L. Zlajpah, "On Time Optimal Path Control of Manipulators with Bounded Joint Velocities and Torques", *International Conference on Robotics and Automation* (April, 1996) pp. 1572–1577.
7. K. Ohishi and T. Someno, "Robust Robot Manipulator Control with Autonomous Consideration Algorithm for Torque Saturation", *Advanced Robotics* **12** Nos 7 & 8, 755–769 (1999).
8. T. Chan and R. Dubey, "A Weighted Least-Norm Solution Based Scheme for Avoiding Joint Limits for Redundant Joint Manipulators", *IEEE Transactions on Robotics and Automation* **11**, No. 2, 286–293 (April, 1995).
9. J. Park, W. Chung and Y. Youm, "Reconstruction of the Inverse Kinematic Solution Subject To Joint Kinematic Limits Using Kinematic Redundancy", *Advanced Robotics* **11**, No. 4, 377–395 (1997).
10. E. Red, "A Dynamic Optimal Trajectory Generator for Cartesian Path Following", *Robotica* **18**, Part 5, 451–458 (2000).
11. B. Fielding, "Predictive Joint Motion Limiting in Robotic Applications," *M.S. Thesis* (Brigham Young University, April, 2002).
12. Newcastle University's Chemical and Process Engineering Web Server, "Dealing with Measurement Noise", © M.T. Tham (1996–1998). See Web URL: http://lorien.ncl.ac.uk/ming/filter/fiewma.htm.
13. Y. Nakamura and H. Hanafusa, "Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control", *Journal of Dynamic Systems, Measurement, and Control* **108**, No. 3, 163–171 (Sept. 1986).
14. T. Yoshikawa, "Manipulatability of Robotic Mechanisms", *2nd International Symposium of Robotics Research*, Tokyo, Japan (1984) pp. 91–98.