METHOD OF TOLERANCE-BASED TRAJECTORY PLANNING AND CONTROL

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention relates to navigation of a path by a moving object, and, more particularly, to motion control systems for computer-controllable machine tools.

Description of the Related Art

[0002] Automation has resulted in the development of motion controllers capable of signaling actuator devices to effect motion in linkages along a desired trajectory while performing useful work. Motion controllers permit increased speed and precision in performing a given task over manual operation. Robots and automated manufacturing equipment are examples of a few of the products that utilize motion control technology. Programming these devices is often accomplished by specifying the desired trajectory as a collection of line/arc segments, along with the desired velocity of a tool along each segment. The velocity of the tool is often kept constant along each segment or group of segments of complex trajectories because velocity optimization along each point in the trajectory would be very time consuming.

[0003] Most humans who program tool trajectory have a fundamental understanding of the trade-off between velocity and accuracy. It is well known that at higher velocities it becomes more difficult for the control system to stay on the desired trajectory. Thus, trajectory programmers must make a trade-off between the velocity and the precision of motion along the desired trajectory. These decisions are often based on the programmer's experience and result in an iterative programming process wherein the trajectory is executed and then modified to reduce the velocity in sections where an undesirable deviation from the desired trajectory is observed. Thus, programmers control the deviation from the desired trajectory, and therefore the quality of the motion, by manipulating the tool velocity along the trajectory.

[0004] Motion control systems for manufacturing equipment, often referred to as Computer Numerical Controllers (CNCs), attempt to maximize the velocity of motion control while minimizing the deviation from the desired trajectory. CNCs may be used

to control manufacturing equipment such as lathes, grinders and mills. CNCs are computing devices adapted for the real-time control of machine tools. A numerical controller receives a set of coded instructions that form a part program. Part programs are typically expressed in a standard G&M code language, or a close derivative of this language based on either the International Standards Organization (ISO) or the Electronics Industries Association (EIA) RS-274-D, using codes identified by letters such as G, M, or F. The codes define a sequence of machining operations to control motion in the manufacture of a part. The numerical controller converts the codes to a series of electrical signals which control motors attached to a machine tool effecting the motion of the tool along the programmed trajectory.

[0005] A motion controller operating a milling machine is one example of CNC. Lathes, grinders and coordinate measuring machines (CMMs) are other examples of manufacturing equipment which utilize a CNC for motion control. A three-axis CNC milling machine has a head where a tool is mounted, and a table movable relative to the tool in the X, Y plane. Motors control motion of the table in the X and Y directions and motion of tool in the Z direction, establishing an orthogonal X, Y, Z Cartesian coordinate system. Positional sensors (encoders or scales, typically) provide feedback indicating the position of the tool with respect to the coordinate system of the milling machine. The CNC reads in a part program specifying a tool path trajectory that the tool is to follow at a specified velocity or feedrate. The controller continuously compares the current tool position with the specified tool path. Using this feedback, the controller generates signals to control motors in such a way that the tool's actual trajectory matches the desired tool path or trajectory as closely as possible while the tool moves along the tool path at the desired velocity. The controller may be used in conjunction with a computer aided machining (CAM) system.

[0006] The deviation of the actual tool trajectory from the desired trajectory or tool path is referred to as "machining error." The machining error may be computed as the distance between the instantaneous tool position and the desired trajectory as specified by the tool path. CNC tolerance is defined as the amount of the permitted machining error while machining. Motion controllers are expected to maintain good or tight CNC tolerance. The machining error depends on many factors including the performance of

the motion controller and the feedrate selected for traversing the trajectory during machining. In general, higher feedrates will result in larger machining errors.

[0007] Known part programs do not explicitly address CNC tolerance issues. The machine tool operator, part programmer or machinist must set feedrates to attempt to address these issues. In fact, tolerance cannot be expressed using known CNC programming languages, such as EIA RS-274-D, nor do existing motion controllers support the notion of constraining motion so that a CNC tolerance specification is met.

[0008] A tolerance based motion control system, including a method for setting feedrates based upon tolerance, is disclosed in U.S. Patent No. 6,242,880, which is incorporated herein by reference. While this patent represents a significant step forward in the art of motion control, refinements are needed in order to improve feedrates while still operating within the tolerance limits.

SUMMARY OF THE INVENTION

[0009] The present invention applies smoothing to any trajectory of a motive body. For example, the present invention may be applied to smooth the trajectory of any motive body, such as a vehicle or projectile, regardless of whether the vehicle or projectile travels by land, sea or air. In a specific implementation, the present invention may also be applied to adjust the feedrate of a CNC machine tool based on the CNC tolerance specified by the programmer and the programmed tool path, defining the maximum allowable feedrate at each point along the programmed trajectory. This information, along with position feedback, is used to modify the feedrate along the actual trajectory of motion to limit the position deviation from the ideal programmed path so that the requested CNC tolerance can be achieved.

[0010] Tolerance Based Control (TBC) Technology introduces the notion of CNC tolerance to the programming and machining environment. It effectively allows the CNC programmer to program a part by using the maximum, or near maximum, permissible feedrates for a given cutting condition, i.e. cutter, speed, depth of cut, material conditions and so forth, and specify a desired CNC Tolerance.

[0011] The data smoothing technology in TBC control may include examining a target trajectory, for example a programmed NC tool trajectory identifying NC line segments which can be smoothed, and converting these NC line data into arcs while maintaining a given tolerance. Data smoothing may include converting point-to-point linear moves

into a sequence of mutually tangential arcs within allowed tolerances and with relatively small curvature changes. The extent of smoothing may be dependent on the specified tolerance. This technology has the benefit of achieving smoother trajectory, smoother velocity, acceleration, better feedrate control and improved surface finish.

[0012] An advantage of the present invention is that it provides improved trajectories or tool paths for the machine tool. More particularly, the tool paths are smoother, thereby permitting higher feedrates and providing smoother surfaces on the work piece.

[0013] The lookahead technology in TBC replaces non-TBC fixed buffer lookahead algorithms, reducing the possibility of data starvation, overshooting, tolerance breaching, dwell marks and poor surface finish on part products. Lookahead may ensure enough distance to accelerate or decelerate from one speed to another. When the length of a motion segment is relatively short and not enough for a large speed change, the lookahead algorithm may calculate and limit the speed of that motion segment according to some motion profile, such as an S-curve speed profile. Because lookahead is usually performed in real time and includes recursive modification of previously processed motion commands, the high efficiency of the algorithm may result in reduction of the CPU load and an improvement in machine performance. The recursive modification of motions may involve only logic and addition. Moreover, the algorithm may have the advantage of requiring fewer iterations of modifying previous commands. The algorithm may have a further advantage of improved run times which increases the system speed or throughput.

[0014] One option in a non-TBC Lookahead algorithm is to calculate a speed limit upon each instance of the motion distance being found insufficient to achieve a desired speed change. Such calculation of the speed limit may involve a large amount of computation. Repeatedly performing the calculation may be undesirable because of the processing time and resources required. The additive lookahead algorithm of the present invention may use a stop distance concept as a metric of speed limitation, which may be very computationally efficient.

[0015] Unlike the classical Proportional, Integral and Derivative (PID) approach, the TBC approach may use predictive control techniques, which replaces non-TBC point-to-point fixed cycle control with trajectory and velocity based motion control. TBC may include looking at where the cutter tool is in relation to the programmed trajectory and

adjusting accordingly. TBC continually processes machine tool position feedback along with the programmed trajectory and knowledge of a machine's capabilities to instantaneously issue a control signal that is responsive to the instantaneous machining conditions.

[0016] In the "true arc" technology employed by the present invention, tool motion is controlled along a trajectory that is modeled as an arced path without polygon approximation. That is, the tool may be commanded to follow an arced trajectory rather than a trajectory formed by a series of linear segments that approximate a curved trajectory. By use of arced trajectories and associated vectors in real time instead of segment-based trajectories, TBC control may eliminate the conversion of arc trajectories into a straight-line approximation of the arc, allowing for the direct precision machining of arc segments, the elimination of the chord error, the reduction of part program size and the elimination of the processing required to convert complex trajectories into their polynomial approximations.

[0017] The TBC motion kernel of the present invention may take the motion commands from the lookahead queue and control the servo system to track the contour trajectory. The TBC motion control is different from non-TBC motion control in that it is a closed-loop contour tracking control.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The above mentioned and other features and objects of this invention, and the manner of attaining them, will become more apparent and the invention itself will be better understood by reference to the following description of an embodiment of the invention taken in conjunction with the accompanying drawings, wherein:

[0019] Figure 1 is a flow chart of one embodiment of a tolerance based control method of the present invention.

[0020] Figure 2 is a block diagram of one embodiment of a tolerance based control system of the present invention.

[0021] Figures 3a-c are diagrams illustrating the mapping of different tolerance values to different regions of machining space.

[0022] Figure 4a is a plot of program data points, illustrating a method of data compression according to one embodiment of the present invention.

[0023] Figure 4b is a plot of program data points, illustrating a method of data compression according to another embodiment of the present invention.

[0024] Figure 5a is a graphical representation of Level 0 data smoothing.

[0025] Figure 5b is a graphical representation of Level 1 data smoothing.

[0026] Figure 5c is a graphical representation of Level 2 data smoothing.

[0027] Figure 5d is a graphical representation of Level 3 data smoothing.

[0028] Figure 5e is a graphical representation of Level 4 data smoothing.

[0029] Figure 6 is a plot of program data points, illustrating a method of selecting data points for data smoothing according to an embodiment of the present invention.

[0030] Figure 7 is a perspective view of a plot of program data points, illustrating a method of adjusting line moves for smoothing according to an embodiment of the present invention.

[0031] Figure 8 is a plot illustrating one step of one embodiment of a double arc smoothing algorithm of the present invention.

[0032] Figure 9 is a plot illustrating another step of one embodiment of a double arc smoothing algorithm of the present invention.

[0033] Figure 10 is a plot illustrating yet another step of one embodiment of a double arc smoothing algorithm of the present invention.

[0034] Figure 11 is a plot of the tool velocity versus time, also referred to as the S-Curve.

[0035] Figure 12a is another plot of the tool velocity versus time, also referred to as an S-Curve.

[0036] Figure 12b is a diagram indicating how distances are related to the plot of Figure 32a.

[0037] Figure 13 is a diagram illustrating the concept of stop distance.

[0038] Figure 14 is yet another plot of the tool velocity versus time for both additive and non-additive lookahead algorithms.

[0039] Figure 15 is a flow chart of one embodiment of an additive lookahead algorithm of the present invention.

[0040] Figure 16 is a block diagram of one embodiment of a non-TBC motion control arrangement.

[0041] Figure 17 is a block diagram of one embodiment of a TBC motion control arrangement of the present invention.

[0042] Figure 18 is another block diagram of the TBC motion control arrangement of Figure 39.

[0043] Figure 19 is a plot illustrating the calculation of the stopper plane normal direction for line motion.

[0044] Figure 20 is a plot illustrating the calculation of the stopper plane normal direction for arc motion.

[0045] Figure 21 is a plot illustrating line motion geometry analysis.

[0046] Figure 22a is a first plot illustrating arc motion geometry analysis.

[0047] Figure 22b is a second plot illustrating arc motion geometry analysis.

[0048] Figure 23 is a flow chart illustrating operation of one embodiment of a TBC S-Curve Controller.

[0049] Figure 24 is a block diagram illustrating one embodiment of a TBC control arrangement of the present invention

[0050] Figure 25 is a block diagram illustrating one embodiment of an arrangement of the present invention for variable-gain feedback control of normal error.

[0051] Corresponding reference characters indicate corresponding parts throughout the several views. Although the drawings represent embodiments of the present invention, the drawings are not necessarily to scale and certain features may be exaggerated in order to better illustrate and explain the present invention. The exemplification set out herein illustrates an embodiment of the invention, in one form, and such exemplifications are not to be construed as limiting the scope of the invention in any manner.

DESCRIPTION OF THE PRESENT INVENTION

[0052] Referring now to the drawings and particularly to Figure 1, there is shown one embodiment of a tolerance based control method 100 of the present invention, including the performance of the major functions of a tolerance based controller of the present invention as it machines the part defined by the part program. While the controller of this disclosed embodiment relates to a particular case of the present invention, dealing with CNC machine controls, the invention may be generally applied to bodies having various motion control situations as one of skill in the art would recognize, such as robotic control, vehicular control, projectile control, and the like.

[0053] In a first step 102, the controller receives part data from the part program. The part program may include data specifying the dimensions, shape, and other physical characteristics of the part or "work piece" to be machined. In general, a trajectory preprocessing step 104 includes calculating a desired tool path and feedrates that the machine tool should follow in order to produce the part within a relatively short period of time and within dimensional tolerance constraints. As the part is machined, position feedback specifying the actual position of the machine tool is used in a predictive and random error compensation step 106 to modify servo commands to redirect the machine tool to compensate for, i.e., correct, both predictive or repeatable errors and random or non-repeatable errors in the actual machine tool path as compared to the commanded tool path. In a final step 108, the modified servo command is transmitted to the servo amplifier for use in actuating the machine tool.

[0054] The method of Figure 1 is now described in more detail with reference to the flow diagram of Figure 2. In general, steps 102 and 104 correspond to the Queue Manager Task of Figure 2, and steps 106 and 108 correspond to the Run System Task of Figure 2. The part program may provide the locations of discrete points on the surface of the part, and the Numerical Control may interpolate between these discrete points to thereby define a desired trajectory or tool path formed of contiguous lines and arcs. As used herein, the term "arc" may indicate a segment of a circle. That is, the arc may have a constant radius.

[0055] The Queue Manager Task preprocesses the part program motion data with four consecutive operations that transform it into high quality motion data before the Run System Task executes it. The four operations are tolerance understanding within Tolerance Queue 202, data compression within Compression Queue 204, data smoothing within Smoothing Queue 206, and additive lookahead within Lookahead Queue 208. The Queue Manager may perform the corresponding data processing on each queue and move the data flow through the consecutive queues in a pipeline manner.

[0056] The Queue Manager may preprocess NC motion commands such as positioning (G00), line (G01), arc (G02, G03), and an NC tolerance command E used for the tolerance based control. Use of the TBC technology is consistent with existing or legacy CNC part programs. This technology allows the operator to specify NC tolerance commands along with the existing NC commands for execution. By inserting a new NC

tolerance command within an existing G&M code part program, the CNC programmer can specify different CNC tolerance constraints.

[0057] The NC tolerance commands specify a region and a tolerance value to be applied to that region along with a command identification number. An NC tolerance command is defined as an E code:

E tol Xx1 Yy1 Zz1 Xx2 Yy2 Zz2 Iid

[0058] An E code requests that a three-dimensional rectangular region defined by the coordinates of the diagonal corners (x1, y1, z1) and (x2, y2, z2), x1<x2, y1<y2, z1<z2, hold a non-zero NC tolerance value of tol. This specific tolerance command may be identified by an integer value id.

[0059] It is possible for multiple E codes to define overlapping tolerance regions. One point may belong to a number of active NC tolerance regions.

[0060] One embodiment of tolerance regions having different values and overlapping one another is illustrated in Figures 3a-c, Figure 3a illustrating a first case, and Figure 3b illustrating a second case. For purposes of computational efficiency, a line move may be divided into at most two lines. Only one more point, point p', is generated between points p0 and p1.

[0061] In the case of an arc (Figure 3c), three equidistant points t2, t3, t4, are added between an arc starting point t1 and an end point t5. The lowest of the tolerances of the five points defines the tolerance of the arc, as indicated by the equation: tol = min {t1, t2, t3, t4, t5}.

[0062] After the tolerance understanding operation, the elements to be taken from Tolerance Queue 202 and sent to Compression Queue 204 are: positioning commands, line commands with required tolerance attached, arc commands with required tolerance attached, and the program end command.

[0063] In a data compression operation within Compression Queue 204, the program data is compressed. In many part programs, there are short line moves that result in a poor surface finish on the part. It is possible to combine some of these short lines into longer lines, which may improve the part quality. The data compression operation may determine whether short lines can be combined into longer lines, and the data compression operation may perform such data compression if it is possible to do so. In a data compression operation illustrated in Figure 4a, points P_1 and P_2 are eliminated and a

new line is formed from point P_0 to point P_3. Tolerance boundaries or constraints on the tool path are indicated by dashed lines 402, 404. Tolerance boundaries 402, 404 may be smaller by an order of magnitude than user-defined numerical control tolerances by E code.

[0064] A line command is a point-to-point move. Combining short line moves into a longer one may include determining a starting point and an end point among the target point sequence for a new line move and removing all the intermediate points. The new, longer line move can replace the old short line moves only if some conditions are met, including the tolerance condition, i.e., the distance from each of the removed points to the new line is within a given tolerance.

[0065] To do data compression, all line commands may be sequentially examined in Compression Queue 204. Four tests or criteria are used to determine whether a next point should be added to the compression queue, or whether compression should be performed on only the points that are presently in the compression queue. In the illustration of Figure 4b, assume that points p0, p1, p2 and p3 have already been found to meet the compression criteria, and that it is being determined whether point p4 should be added to the compression queue, or whether compression should be performed on only points p0, p1, p2 and p3. The four compression criteria may be as follows:

Test 1: $|p4-p0|<0.05$ inch.

Test 2: For p3, the angle θ is between -90 degrees and 90 degrees.

Test 3: p0.tol = p1.tol = p2.tol = p3.tol = p4.tol. That is, an equal tolerance has been assigned to each point from the tolerance table.

Test 4: The perpendicular distances from p1, p2 and p3 to the line p0p4 are less than the specified tolerance.

[0066] If the above four tests are satisfied, then p4 is added to the compression queue. If the above four tests are not satisfied, then p3 is added to the smoothing queue and p1 and p2 are discarded in compression. Essentially, three segments (p0p1, p1p2 and p2p3) are combined into one segment (p0p3).

I. DATA SMOOTHING

[0067] Data smoothing operations convert NC line data into arcs while maintaining a given tolerance. The data smoothing may advantageously result in smoother velocity

and acceleration, better control of feedrates, improved surface finish, and full utilization of the TBC technology.

[0068] Figures 5a-e illustrate level 0, level 1, level 2, level 3, and level 4 smoothing, respectively: level 0, including no smoothing and producing a cut polygon; level 1, in which arcs replace lines, and arcs may not be connected to each other in most cases; level 2, in which arcs replace lines, arcs are connected to each other, and arcs are not tangential to each other in most cases; level 3, in which arcs replace lines, arcs are connected to each other, and arcs are tangential to each other; and level 4 (a smoothing level that may be provided by the present invention), in which arcs replace lines, arcs are connected to each other, arcs are tangential to each other, and arc curvatures change gradually. In level 0-3 smoothing, the output of smoothing, the final trajectory, always passes through the data points. In Figure 5d, c1 and r1 are the center and radius of the arc defined by the points p0, p1 and p2; and c2 and r2 are the center and radius of the arc defined by the points p2, p3 and p4. The curvature of an arc is defined as the inverse of the radius (1/r). Thus, in the level 3 example of Figure 5d, the relatively large radius change from r1 to r2 results in a correspondingly large curvature change between the two arcs.

[0069] Level 4 is different from the other smoothing levels in the sense that the data points are adjusted (within the specified tolerance) to achieve gradual curvature changes. This adjustment of the data points is acceptable in applications such as the multi-axis contouring (or metal cutting using CNC machines) because the adjustments are very small (typically less than 0.0005 inch) and are within a specified tolerance.

[0070] The points may be adjusted to achieve gradual curvature changes such that the trajectory approaches being elliptical. Gradual change of curvature along the trajectory may facilitate motion control.

[0071] Motion of a tool along an arc requires a force, referred to as "centripetal force", directed toward the center of the arc. In the case of CNC machines, this force may be provided by the motors. Sudden and relatively large changes in the centripetal forces (as the motion moves from one arc to the next) may result in larger overshoots, which may be detrimental to the resulting surface finish.

[0072] Figure 5e illustrates minor adjustments to the positions of points p0 to p4 as compared to their earlier positions as shown in Figure 5d. The minor adjustments may

result in the points defining a trajectory that approaches being elliptical. An elliptical trajectory may be ideal or optimal because, along an ellipse, the curvature changes continuously. However, how large an adjustment that is allowed within the scope of the invention may depend upon the tolerance. Thus, the points may not actually fall on an ellipse after the adjustments.

[0073] In one embodiment of the present invention, instead of optimizing the curvature changes, a sub-optimal solution is employed such that the trajectory includes only circular arcs as opposed to being elliptical. Defining the trajectory as a series of circular arcs may have the advantage of reducing the computational/mathematical complexity of calculating the point adjustments. Thus, even if it is possible to achieve an elliptical trajectory by adjusting the points within tolerances, the points are instead adjusted such that the trajectory is defined by a series of circular arcs in one embodiment of the present invention.

[0074] In one embodiment, the motion control is implemented in a 200 usec loop, i.e., the motion control is updated every 200 usec. Thus, there is a need for computations to be performed efficiently such that the computations may be performed every 200 usec. Due to limitations in the speed of a processor which may perform several other operations in addition to motion control, using a trajectory defined by a series of circular arcs (gradual changes in curvature) rather than an elliptical trajectory (continuous change of curvature) may be helpful.

[0075] Data smoothing may include three phases or operations, namely preparing for smoothing, adjusting for smoothing, and smoothing using a double arc algorithm. In a first operation of a smoothing method of the present invention, line data is identified for smoothing. It is not always feasible to convert NC line data into arcs while maintaining a given tolerance, as with, for example, connected lines with sharp angles. It is sometimes not desirable to smooth NC line data, for example, long line moves. Certain conditions may need to be satisfied before raw line data can be smoothed. The first operation examines the original line data identifying the line segments, which meet the conditions for smoothing.

A. Prepare for Smoothing

[0076] During the first operation, all line move target points are sequentially examined in the smoothing queue. The line segments, which can be smoothed, are determined and the target points of all these lines may be tagged as "smoothing" or "no smoothing".

[0077] In Figure 6, three points p0, p1, p2 are joined by an arc. In one embodiment, distance d1 is the shortest distance between a midpoint of line segment p0p1 and the arc p0p1p2; and a distance d2 is the shortest distance between the midpoint of line segment p1p2 and the arc p0p1p2. A middle one of the three points, i.e., point p1, may be marked for smoothing if the following three conditions or criteria are satisfied:

Test 1: Both p0p1 and p1p2 are lines.

Test 2: Max{d1,d2} < 0.0005 inch. That is, both d1 and d2 are less than some predetermined distance, such as 0.0005 inch.

Test 3: The angle $\theta > 145$ degrees, or some other predetermined angle.

If the above three conditions are not met, then point p1 may not be marked for smoothing.

B. <u>Adjust for Smoothing</u>

[0078] In a second operation, line data is adjusted for smoothing. The second operation may reduce the curvature changes on a three-dimensional curve represented by a sequence of three-dimensional points. The local curvature at a given point on such a curve is mainly determined by neighboring points. A minor change of the position of the given point or of its neighboring points may significantly change the local curvature. Thus, it is possible to modify the positions of points within the given tolerance such that the curvature changes on the three-dimensional curve represented by the modified points will be reduced. The smaller the curvature changes on the curve, the smoother the curve may be.

[0079] Figure 7 illustrates an embodiment of the present invention for adjusting line moves for smoothing. The line moves may be adjusted relative to each other to control the path curvature and to prepare for arc fitting. Two construction arcs 612, 614, which are not coplanar, are located that pass through points $P_0P_1P_3$ and $P_0P_2P_3$, respectively. A line of intersection of the planes of arcs 612, 614 is designated 616. The center of a circle defined by arc 612 is designated c2; and the center of a circle defined by arc 614 is designated c1. A projection of point p1 on the plane of arc 614 is designated point q1. That is, a line that extends through both of points p1 and q1 is perpendicular to the plane

of arc 614. Similarly, a projection of point p2 on the plane of arc 612 is designated point q2. That is, a line that extends through both of points p2 and q2 is perpendicular to the plane of arc 612. The closest point on arc 614 to point p1 is designated s1; and the closest point on arc 612 to point p2 is designated s2.

[0080] Generally, a point is adjusted if both that point and an adjacent point have been marked for smoothing. A point may be marked for smoothing if it satisfies each of the three tests or conditions given above, i.e., both p0p1 and p1p2 are lines; Max{d1,d2} < 0.0005 inch; and angle $\theta > 145$ degrees. In the example shown in Figure 7, point p1 may be adjusted if p1 has been marked for smoothing and either or both of p0 and p2 has been marked for smoothing. Point p1 may be adjusted, if at all, along an adjustment vector v1 extending from p1 to s1. Similarly, point p2 may be adjusted if p2 has been marked for smoothing and either or both of p1 and p3 has been marked for smoothing. Point p2 may be adjusted, if at all, along an adjustment vector v2 extending from p2 to s2.

[0081] After the adjustments associated with arcs 612 and 614, the process continues with subsequent adjustments associated with arcs formed by the next set of four points, i.e., points p1, p2, p3 and p4. That is, adjustments may be made based upon an arc defined by points p1, p2 and p4 and an arc defined by points p1, p3 and p4. The adjustment process may continue indefinitely with calculations based upon subsequent sets of four points along the trajectory.

[0082] The magnitude of the adjustment of a point's position along an adjustment vector may be dependent upon whether one of the two adjacent points is unmarked for smoothing. More particularly, if a marked point lies immediately after or before an unmarked point, i.e., the marked point lies between an unmarked point and another marked point, then the position of the marked point may be adjusted by half the length of the adjustment vector. That is, the marked point may be moved half way along the adjustment vector. In equation form, the position of the point may be calculated as $p = p+0.5*v$. That is, the point is moved with a coefficient or factor of 0.5. Otherwise, if a marked point lies between two other marked points, then the position of the point is adjusted twice with a coefficient or factor of approximately 0.19 in each of the two adjustments. In the first adjustment, the direction and magnitude of the adjustment vector is determined with the adjusted point being the second of the four points used in

the calculations illustrated in Figure 7. In the second adjustment, the direction and magnitude of the adjustment vector is determined with the adjusted point being the third of the four points.

[0083] To better illustrate the embodiment of Fig. 7, various adjustment scenarios are described below. The check to verify that the total adjustment is less than an adjustment limit, such as 0.0002 inch, is omitted for simplicity and ease of illustration.

Case 1:

Condition: p0 is not marked for smoothing; p1 is marked for smoothing; p2 is not marked for smoothing.

Result: Because p1 is not adjacent another marked point, p1 is not adjusted even though it is marked for smoothing .

Case 2:

Condition: p0 is not marked for smoothing; p1 and p2 are marked for smoothing; p3 is not marked for smoothing.

Result: p1 = p1+0.5*v1; p2=p2+0.5*v2.

Case 3:

Condition: p0 is not marked for smoothing; p1, p2 and p3 are marked for smoothing; p4 is not marked for smoothing.

Result: p1 = p1+0.5*v1; p2=p2+0.1877*v2+0.1877v2'; p3 = p3+0.5*v3, wherein v1 and v2 are calculated from points p0p1p2p3, and v2' and v3 are calculated from points p1p2p3p4.

Case 4:

Condition: p0 is not marked for smoothing; p1, p2, p3 and p4 are marked for smoothing.

Result: p1 = p1+0.5*v1; p2=p2+0.1877*v2+0.1877v2'; p3 = p3+0.1877*v3+0.1877v3', wherein v1 and v2 are calculated from points p0p1p2p3, v2' and v3 are calculated from points p1p2p3p4, and v3' is calculated from points p2p3p4p5.

C. Double Arc Smoothing

[0084] In a third operation, smoothing is performed using a double arc algorithm. The third operation may include converting the lines into arcs, which are mutually tangent to each other, while maintaining a given tolerance. This third operation may include sequentially checking consecutive line target points processed in the second operation in Smoothing Queue 206 and performing the conversion of lines to arcs.

[0085] The double arc algorithm causes the final smoothed trajectory to pass through all the target points of the line segments to be smoothed. Thus, the tolerance requirements will be well satisfied.

[0086] The double arc algorithm may perform two functions simultaneously in a pipeline manner. First, the algorithm may determine a tangent vector at each target point of a line segment to be smoothed. Second, the algorithm may generate double arcs to replace the line segments to be smoothed. Let $p_1, p_2, p_3, p_4, p_5, p_6, \ldots, p_n$ be consecutive line target points already processed by the second operation in smoothing queue 206. The first function may perform the following steps:

- If p_i, p_{i+1} or p_{i+2} is tagged "smoothing", calculate a desired tangent vector t_{i+1} (Figure 8) at point p_{i+1} based on the positions of points p_i, p_{i+1}, and p_{i+2}. The three points p_i, p_{i+1} and p_{i+2} define the $Arcp_ip_{i+1}p_{i+2}$ and t_{i+1} is the tangent vector to the arc at point p_{i+1}.
- If none of p_i, p_{i+1}, p_{i+2} is tagged "smoothing", do nothing
- Increase i by 1
- Repeat the above steps

[0087] The above function will create a sequence of desired tangent vectors at the target points of the line segments to be smoothed, denoted as $t_1, t_2, t_3, t_4, t_5, t_6, \ldots, t_n$. The tangent vectors may be referred to as "desired tangent vectors" because the direction changes of these vectors are gradual, which further smoothes the curvature changes in the smoothed trajectory.

[0088] The second function may perform the following steps:

- If p_i, p_{i+1} and p_{i+2} are marked for smoothing and the corresponding desired tangent vectors t_i, t_{i+1} and t_{i+2} have been calculated for p_i, p_{i+1} and p_{i+2}, then create a pair of three-dimensional arcs a_{i1} and a_{i2} such that:
  - a_{i1} starts at point p_i and is tangent to the tangent vector t_i
  - a_{i1} is tangent to a_{i2} at a point p_{i+1} between p_i and p_{i+2}
  - a_{i2} starts at point p_{i+1}, ends at point p_{i+2}, and is tangent to the tangent vector t_{i+2}.
- Check the shortest distance (d) between p_{i+1} and the closest arc. If the distance is within a given tolerance, the double arcs are valid.

- If the double arcs between p_i and p_{i+2} are invalid, then create a pair of three-dimensional arcs a_{i1} and a_{i2} between p_i and p_{i+1} such that:
  - a_{i1} starts at point p_i and is tangent to the tangent vector t_i
  - a_{i1} is tangent to a_{i2} at a point p' between p_i and p_{i+1}
  - a_{i2} starts at point p', ends at point p_{i+1}, and is tangent to the tangent vector t_{i+1}
- Repeat the above steps to generate double arcs between p_{i+1} and p_{i+2}
- Increase i by 2
- Repeat the above steps

[0089] The operation of the second function in the embodiment described above is illustrated in Figures 9 and 10. In Figure 9, p0, p1 and p2 are marked for smoothing. A pair of three-dimensional arcs a1 and a2 are created such that arc a1 starts at point p0 and is tangent to tangent vector t0; arc a1 is tangent to arc a2 at a point p' between p0 and p2; and arc a2 starts at point p', ends at point p2, and is tangent to the tangent vector t2. If a distance d between p1 and the closer arc a1 of the two arcs is within a given or predetermined tolerance, then the double arcs a1 and a2 are valid.

[0090] If, on the other hand, the double arcs a1 and a2 are invalid, i.e., the distance d is not within the given tolerance, then another pair of three-dimensional arcs a3 and a4 (Figure 10) are created such that arc a3 starts at point p0 and is tangent to tangent vector t0; arc a3 is tangent to arc a4 at a point p" between p0 and p1; and arc a4 starts at point p", ends at point p1, and is tangent to the tangent vector t1. The above steps may then be replicated in order to create yet another pair of three-dimensional arcs (not shown) between points p1 and p2. Any or all of arcs a1, a2, a3 and a4 may be circular arcs, and each of the arcs may have a different, respective radius.

[0091] With the double arc smoothing thus completed between points p0, p1 and p2, the above-described second function may be repeated for the next three points along the trajectory, i.e., points p2, p3 and p4. Further, the second function may be repeated for each subsequent set of three points along the trajectory.

[0092] The above-described double arc smoothing embodiment is directed to the case in which p0, p1 and p2 are each marked for smoothing. Assume now that, of the four points p0, p1, p2 and p3, p1, p2 and p3 are marked for smoothing. Tangential vectors t0 and t1 may be derived from the arc defined by p0, p1, p2, and tangential vector t2 may be derived from the arc defined by p1, p2, p3. The following five steps may be taken:

1. Double arcs, i.e., arc a1 from p0 to p' and arc a2 from p' to p2, which are tangential at point p' may be generated from (p0, p2, t0, t2), as shown in Figure 9.

2. The shortest distance between p1 and the closer of the two double arcs may then be checked. If the distances in the x, y and z directions are less than a predetermined distance, such as 0.0002 inch, then the double arcs a1 and a2 are valid.

3. If the double arcs a1 and a2 are invalid, then double arcs a3 and a4 are generated between points p0 and p1 and based upon tangent vectors t0, t1. The two arcs a1 and a2 may be tangential at point p".

4. There may be a tolerance check for the double arcs. A distance d in Figure 10 is the shortest distance between point p" and a line extending between points p0 and p1. If distance d is less than a predetermined distance, such as 0.0005 inch, then the double arcs a3 and a4 are valid. Otherwise, if the double arcs a3 and a4 are invalid, then define the trajectory between points p0, p1 as a line extending between points p0 and p1.

5. Repeat steps 3 and 4 to generate double arcs between p1 and p2 based upon tangent vectors t1 and t2.

II. ADDITIVE LOOKAHEAD

[0093] In an additive lookahead operation within Lookahead Queue 208, it is ensured that there is enough distance to accelerate or decelerate between all velocity changes in the program data. This has advantages in that the machine bumping and dwell marks may be reduced, and the surface finish in machining may be improved. For each programmed move, a stop distance is computed that defines the distance required to decelerate the axes to zero velocity according to S-curve acceleration. It is referred to as S-Curve because the acceleration/deceleration part of the velocity profile (velocity plotted versus time) illustrated in Figure 11 looks like an "S" instead of a straight line. The stop distance may be determined by the lower of: a) the maximum move velocity determined by the angle between the current and next moves; and b) the maximum move velocity for the path curvature determined by the machine model.

[0094] The additive lookahead function may include searching in Lookahead Queue 208 for velocity and distance information and determining whether there is enough

distance for the machine to accelerate/decelerate properly from one motion command to the next.

[0095] There is a mathematical conversion between the stop distance and the velocity. The proper velocity may be calculated from the stop distance and the S-Curve, as discussed in more detail below.

[0096] The additive lookahead algorithm may require knowledge of the maximum velocities allowed for different part geometries. A tolerance/arc radius/velocity table may be built for this purpose. The table may provide the maximum allowable velocities for arcs of different radii or curvatures with different tolerance requirements.

A. **Stop distance and stopper**

[0097] The additive lookahead algorithm introduces the concepts of stop distance and stopper. The stop distance of a motion command with nominal feedrate v_1 (Figure 12a) and end point speed v_0 may be defined as the distance traveled from a target point (Figure 12b) to the point at which a full stop can be achieved according to the S-curve speed profile, as illustrated by the shaded area in Figure 12a. Stopper, which is a virtual target point for a motion command, may be selected such that if the system reduces speed from nominal feedrate v_1 and comes to a full stop at the stopper according to the S-curve, it passes the actual target point at the desired end point speed v_0. The stop distance may be defined as the distance between the target point and the stopper.

[0098] Once the nominal feedrate v_1 of a motion command is determined, there exists a one-to-one mapping between the stop distance and the end point speed v_0, assuming that the S-curve speed profile is fixed, i.e., the maximum acceleration and jerk are fixed. It may be assumed herein, unless otherwise stated, that the S-curve profile is fixed. Therefore, the stop distance has a direct correspondence to the end point speed; and if the stop distance is given, the end point speed may be accordingly uniquely determined, and vice versa. The additive lookahead algorithm may use the stop distance as a metric of end point speed, which allows the algorithm to be computationally efficient, as discussed below.

[0099] The stopper may be used in an S-curve controller to design a motion profile. The S-curve controller may aim at the stopper instead of at an actual target point. The S-curve controller may generate a speed profile in which the controlled tool passes the

actual target point at the same point in time at which the tool reaches the desired slowdown speed.

B. Speed limitations of motion commands

[00100] In the additive lookahead algorithm, each motion command may have four speed limitations: specified feedrate, machine allowable speed, path velocity limit and joint velocity limit. The specified feedrate may be defined as the desired feedrate value given by the upper-level motion planning. The machine allowable speed is a machine limitation on the speed, and depends on the hardware of the machine and laws of physics. The path velocity limit is a limit for arc motion. The path velocity limit may depend on the arc radius and tolerance specification. The larger the arc radius and tolerance are, the larger the maximum path velocity limit is. The joint velocity limit may be defined as a speed limit at the joint of two consecutive motion commands. The joint velocity limit may depend on the angle between the two motions and the tolerance specification. The larger the tolerance and the smaller the angle are, the larger the joint speed limit value is.

[00101] In the additive lookahead algorithm, the nominal feedrate is the lowest value among the specified feedrate, the machine allowable speed and the path velocity limit. For example, if the specified feedrate is higher than the machine allowable speed and the path velocity limit, the lookahead algorithm would force the feedrate to slow down to the smaller value of the machine allowable speed and the path velocity limit.

[00102] The slowdown feedrate at the end point of a motion command may depend on the joint velocity limit as well as the nominal feedrate of that motion command and the next motion command. Because the nominal feedrate is the smallest one among the specified feedrate, the machine allowable speed and the path velocity limit, slowdown may occur during a current motion segment and speedup may occur in a next motion segment, such that the feedrate for the current motion segment may be less than the nominal value. The additive lookahead algorithm may use the nominal and slowdown feedrates to calculate the stop distance.

C. Valid stop distance condition

[00103] The lookahead algorithm may ensure that the distance between the tool and some location ahead of the tool is sufficient to allow the speed of the tool to be changed to a desired level by the time the tool arrives at the location. This condition may be

referred to as a "valid stop distance condition" with regard to the additive lookahead algorithm and may be defined mathematically by equation (1):

(1) $ds_i \le l_{i+1} + ds_{i+1}$

where *ds* and *l* represent the stop distance and the length of a motion segment, respectively, and subscripts i and i+1 represent the index of motion commands. In the diagram of Figure 13, motion command cmd_i has an end point at p_i, stopper point at stopper_i, and stop distance ds_i. If the stop distance of cmd_i does not exceed the sum of the length and stop distance of cmd_(i+1), or if the stopper_i is in front of the stopper_(i+1), then cmd_i satisfies the valid stop distance condition.

[00104] The valid stop distance condition can also be explained in another way as illustrated by equation (2):

(2) $ds_i - ds_{i+1} \le l_{i+1}$

Because the stop distance ds_i and ds_(i+1) correspond to the feedrate at the end point p_i of motion cmd_i and the feedrate at the end point p_(i+1) of motion cmd_(i+1) respectively, equation (2) indicates that the segment length l_(i+1) is long enough to accommodate a speed slowdown from the feedrate at the point p_i to the feedrate at the point p_(i+1).

C. **Additive Lookahead vs. Non-additive Lookahead**

[00105] A difference between a TBC additive lookahead algorithm and a non-additive lookahead algorithm is that the additive lookahead algorithm generates an S-curve targeting the stopper and uses only part of the S-curve so that the speed is reduced from v_1 to v_0 at the target point. The non-additive lookahead generate an S-curve targeting the actual target point and the speed is reduced from v_1 to v_0 at the target point.

[00106] Figure 14 shows a comparison between the additive and non-additive S-curve speed profile. The speed is reduced from v_1 to v_0 at the target point in both profiles. However, in the additive S-curve, the slowdown occurs between time t_{tbc1} and time t_{tbc0}, while in the non-additive S-curve, the slowdown occurs between time t_{wm1} and time t_{wm0}. Thus, the non-additive S-curve requires a longer time and a longer distance than the additive S-curve in order to achieve the same reduction in speed from v_1 to v_0.

[00107] In the additive lookahead process, the continuous trajectory changes may affect the previously-processed commands. Each time a new command is added to the lookahead queue and a stop distance is calculated for the new command, all the

commands in the queue which have already been processed may be reviewed to see if the previously-calculated stop distances for these commands need to be updated or not.

[00108] When a new command cmd_i is received and processed, it has a path length l_i, and a stop distance ds_i. The additive lookahead algorithm may perform the following steps (Figure 15):

- Check command cmd_{i-1} to see if it meets the "valid stop distance condition", i.e., if ds_(i-1) is equal to or less than the sum of l_i and ds_i.
- If command cmd_{i-1} does not meet the "valid stop distance condition", i.e., ds_(i-1) is greater than the sum of l_i and ds_i, then replace the stop distance ds_(i-1) for command cmd_{i-1} with the sum of l_i and ds_i.
- If command cmd_{i-1} meets the "valid stop distance condition", i.e., ds_(i-1) is equal to or less than the sum of l_i and ds_i, then the stop distance ds_(i-1) for command

  cmd_{i-1} may remain unchanged, and the update operation is complete.
- If (i-1) is equal to 0, the update operation is complete.
- i-- (decrement the value of i) and repeat the above steps.

[00109] When the additive lookahead function is sequentially reviewing backward the commands previously processed in the Lookahead Queue 208, checking the "valid stop distance condition," and updating the stop distance of the commands, the updating operation may stop after the first command that meets the "valid stop distance condition" is found. Because this command meets the "valid stop distance condition," the stop distance of this command may remain unchanged. Further, because this command meets the "valid stop distance condition," all the commands previous to it met, and still meet, the "valid stop distance condition," and none of the previous commands may need to change.

[00110] In the additive lookahead (Figure 15), only the third step, i.e., the step of calculating the stop distance ds_i, may require a lot of computation, and may be calculated as shown in equation (3). This calculation may be needed only once for every motion command, and there may be no need to repeat it.

$$(3)\qquad ds=\begin{cases}\dfrac{1}{6}J\left(\dfrac{2v_0}{J}\right)^{\frac{3}{2}} & v_0<\dfrac{1}{2}v_1\\ v_1\sqrt{\dfrac{v_1}{J}}+\dfrac{1}{6}J\left(\dfrac{2(v_1-v_0)}{J}\right)^{\frac{3}{2}}-\left(\dfrac{2(v_1-v_0)}{J}\right)^{\frac{1}{2}}v_1 & v_0\geq\dfrac{1}{2}v_1\end{cases}$$

[00111] In general, the additive lookahead algorithm may be more computationally efficient than the non-additive lookahead algorithm due to the additive lookahead algorithm having only one heavy load step which is outside the recursive loop and is only performed once. The additive lookahead algorithm also needs less distance to reduce a same amount of speed, as shown in Figure 14, which means fewer previous motion commands need to be modified.

III. MOTION KERNEL

[00112] The motion kernel module may perform the tolerance based motion control and may control the motion associated with program lines and arcs. The motion kernel provides predictive and true arc motion, wherein a line or an arc is the basic trajectory unit. The control may push the axes along the programmed trajectory, correcting for deviations as they are encountered.

[00113] The predictive TBC motion control may be based on a different mechanism that recognizes line/arc data (no pre-computed interpolation points) and directly pushes the axes along the line/arc trajectory. At every control cycle (which may have a period of 200 μsec, for example), the control may read the position feedback, may compare the current position with the commanded trajectory (a line or an arc), and may determine a point on the trajectory that is the point closest to the actual current position. This point may be defined as the ideal current position. The difference between the actual current position and the ideal current position may be defined as the instantaneous random position error or tolerance error. The instantaneous random position error, also referred to simply as the "position error," may be thought of as the deviation of the actual current position from the line/arc trajectory.

[00114] The TBC tolerance error is the vector difference between the actual current position and the ideal current position which may be dynamically calculated in real time. In a TBC control system, the ideal current position may be calculated every control cycle, and the tolerance error may be directly corrected by TBC. Moreover, in a TBC system, the tolerance errors for different axes may be handled coordinately.

[00115] The TBC predictive motion control, at every control cycle, may perform the following steps:

- Calculate the three-dimensional tangential velocity at the ideal current position of the trajectory according to S-Curve
- Calculate the instantaneous tangential and centripetal acceleration forces and their predictive compensate vectors
- Calculate the instantaneous random position error
- Determine a three-dimensional random corrective velocity
- Combine the tangential velocity, the predictive compensate vectors and the random corrective velocity together and issue the resultant velocity to push the axes along the trajectory

[00116] The TBC motion kernel of the present invention may take the motion commands from the Lookahead Queue 208 and control the servo system to track the contour trajectory. The TBC motion control is different from non-TBC motion control in that it is a closed-loop contour tracking control. A comparison between non-TBC motion control and TBC motion control is provided below.

A. **Non-TBC Motion Control Structure**

[00117] In order to track a three-dimensional contour trajectory, a conventional control system may decompose the three-dimensional contour into X/Y/Z axes and interpolate the X/Y/Z motion with respect to time. Each axis may have its own time based reference trajectory and may use feedback and feedforward control to track its own motion trajectory independently. Ideally, if each axis tracks its reference trajectory perfectly, the resulting three-dimensional contour would also be perfectly followed. Due to the existence of disturbances, shown in Figure 16, and modeling errors, perfect tracking is normally not achievable and there may be tracking errors between the actual axis motions and reference trajectories. There may be an attempt to minimize the tracking error in each axis without any teamwork or coordination among the three axes. The trajectory of each axis may be tracked without knowing or caring what the state of the trajectory in other axes is. For example, even if the trajectory in one axis meets a large resisting force and lags behind its reference trajectory, there may be an attempt to track the preset trajectories of the other two axes. The result is that even though the trajectory

may be tracked perfectly for two of the three axes, the actual contour that is achieved may still deviate from the reference contour significantly.

[00118] The motion control system illustrated in Figure 16 may be time deterministic. That is, the x/y/z axis motion trajectories may be time based and the entire system's runtime may be accurately determined beforehand. F_x, F_y and F_z may indicate feedforward controllers for each of the three axes; and C_x, C_y and C_z may indicate feedback controllers for each of the three axes.

B. TBC Motion Control Structure

[00119] TBC motion control does not involve decomposing a three-dimensional contour into three independent time-based axis trajectories. Rather, TBC motion control may compare the actual contour position with a reference contour and determine a control signal to minimize the contour error. One embodiment of TBC motion control structure is shown in Figure 17, where all signals are vectors. In contrast, in the non-TBC motion control structure shown in Figure 16, signals are scalars.

[00120] The TBC motion control structure of Figure 17 may employ contour-based feedback plus feedforward control. The contour tracking may be closed-loop, and true arc motion may be possible. That is, the tool may be commanded to follow an arced trajectory rather than a trajectory formed by a series of linear segments that approximate an arced trajectory. TBC motion control may not be time deterministic, and may be capable of on-the-fly feedrate adjustments.

[00121] At every control cycle, such as every 200 usec, the controller may read the actual contour position feedback, compare the actual position with the reference contour, and find out the closest point on the reference contour to the actual position. This point is defined as the ideal current position (ICP). The controller may include three functional blocks: feedforward, feedback and stick friction/backlash compensation as illustrated in Figure 18 and described below.

[00122] At every control cycle, the controller may calculate the desired motion based on the ICP (including acceleration and velocity in both normal and tangential directions), and may include a feedforward controller to convert the desired motion into a velocity control signal. A runtime feedrate adjuster may also be added to adjust the feedrate (velocity) for tolerance control purposes.

[00123] The controller may also calculate the tracking errors in both normal and tangential directions at every cycle. The error in the normal direction is the distance from the actual contour position to the ICP, i.e., the tolerance error or contour error. The error in the tangential direction is defined as the amount of path length that the ICP lags behind. Because the normal error may be important to the system performance, the TBC controller may include two different feedback controllers for the normal and tangential errors and may convert the tracking errors into a velocity control signal. The stick friction and backlash compensation may further improve the accuracy of the machine tool.

[00124] As compared with the non-TBC motion control, an advantage of the TBC motion control is the closed-loop structure. The closed-loop contour tracking control may coordinate the three axes and minimize the contour error. For example, if the x-axis has a large resisting force and lags behind its reference trajectory, the other two axes could slow down to match the x-axis motion so that the actual contour follows the reference contour. The TBC motion kernel may also enable true arc motion because the centripetal acceleration may be considered in the control.

[00125] Because the desired motion at each control cycle is based on the contour, the ICP, and the current system condition, such as the tolerance error, the system may no longer be time deterministic. On-the-fly feedrate adjustment may be possible. When the contour/tolerance error is below the specified value, the system feedrate may be increased. Conversely, when the contour/tolerance error exceeds the specified value, the feedrate may be reduced such that the error is also reduced. In other words, the system may maximize the feedrate while maintaining the contour/tolerance error within a specified range for optimal system runtime or throughput.

C. TBC Geometry Analysis

1. Motion Command Pre-Analysis

[00126] Before executing any motion command, the TBC motion kernel may load the command and analyze the geometry. The stopper plane normal (SPN) direction and stopper position may be calculated in the pre-analysis and may be used through the entire executing period until this motion command is finished. For a line motion, the SPN may be calculated as indicated by equation (6) and shown in Figure 19, which illustrates the line motion geometry pre-analysis:

(6) $$SPN_i = \frac{P_{i-1} - P_i}{\|P_{i-1} - P_i\|}$$

[00127] For an arc motion, the SPN may be calculated as indicated by equation (7) and shown in Figure 20, which illustrates arc motion geometry pre-analysis, where APN stands for the unit direction vector of the arc plane normal direction, and C stands for the center of the arc.

(7) $$SPN_i = -APN \times \frac{P_{i-1} - C_i}{\|P_{i-1} - C_i\|}$$

[00128] The stopper position Psi may be calculated as indicated by equation (8) for both line and arc motions.

(8) $$Ps_i = P_i - ds_i \cdot SPN_i$$

2. Real-time Geometry Analysis

[00129] At each control cycle, the TBC motion kernel may read the actual contour (tool) position feedback Ptl and compare the actual position with the reference contour to find the ideal current position (ICP), contour/tolerance error (e_n) and the distance to the stopper (distance_to_go, d).

[00130] For a line motion, the geometry is shown in Figure 21, which illustrates line motion geometry analysis, and may be calculated as indicated by equations (9) and (10).

(9) $$d = < Ptl - Psi, \quad SPN >$$

(10) $$e_n = (Ptl - Psi) - d \cdot SPN$$

wherein $<A,B>$ is the inner product of vector A and vector B. Thus, if A = [xa, ya, za], and B = [xb, yb, zb], then $<A,B>$ = xa*xb + ya*yb + za*zb. *Ptl* is the tool position, *Psi* is the stopper position of the i motion segment, and *SPN* is the unit direction vector of stopper plane normal (SPN) direction. Equation (9) may be interpreted as the distance d being equal to the length of *Ptl-Psi* in the SPN direction.

[00131] For an arc motion, because the actual contour or tool position may not be on the arc plane, the tool position may be first projected onto the arc plane, as may be calculated as indicated in equation (11) and as shown in Figures 22a-b, which are illustrative of arc motion geometry analysis.

(11) $$P'tl = (Ptl - Ci) - < Ptl - Ci, \quad APN > \cdot APN$$

Equation (11) may be interpreted as P'tl being equal to the unit arc plane normal direction vector APN being multiplied by a difference between vector (Ptl-Ci) and the

vector <Ptl-Ci, APN>.

[00132] The vector P'tl is the projection of (Ptl-Ci) on the arc plane. Once the vector P'tl is found on the arc plane, then ICP, the unit tangential and radial direction vectors dt and dr, and contour/tolerance error e_n (also referred to as the normal error) may be determined as indicated in equations (12), (13), (14) and (15).

(12) $$dr = \frac{P'tl}{\|P'tl\|}$$

(13) $$ICP = Ci + R \cdot dr$$

(14) $$dt = dr \times APN$$

(15) $$e_n = Ptl - ICP$$

[00133] In order to calculate the distance_to_go, i.e., the path length from ICP to the stopper, the arc length between ICP and target point Pi may first be calculated. Any arc that is more than 180 degrees may be cut into two arcs. The distance_to_go may be calculated by use of the following three equations (16), (17) and (18). The meanings of the variables of equations (16), (17) and (18) may be discerned with reference to Figures 22a-b.

(16) $$l = \frac{1}{2}\|Pi - ICP\|$$

(17) $$a = \sin^{-1}\left(\frac{l}{R}\right)$$

(18) $$d = 2aR + dsi$$

D. TBC S-Curve Controller

[00134] S-Curve acceleration provides a non-linear acceleration method that softens and smoothes the transition between the at-the-speed phase and the acceleration/deceleration phase. It is referred to as "S-Curve" because the acceleration/deceleration part of the velocity profile is S-shaped instead of linear.

[00135] The S Curve velocity profile usually has seven phases: increasing acceleration; constant acceleration; decreasing acceleration; at speed; increasing deceleration; constant deceleration; and decreasing deceleration.

[00136] Non-TBC motion control systems may use the S-Curve to generate reference trajectories, i.e., scheduled position, velocity, acceleration and so on. The S-Curve is time-based and implemented outside the control loop (real-time portion). At each

control cycle, based on the distance to go, the TBC S-Curve controller may calculate the desired velocity and acceleration along the reference contour trajectory. The TBC S-Curve controller may be distance-based and implemented as a real time process. The velocity may be calculated from the instantaneous path distance at every control cycle. This mechanism may reduce the velocity error accumulation, and may also support some real time requests such as on-the-fly feedrate adjustment, feed override, pause, resume and single step.

[00137] A flow chart of one embodiment of a method of operation 4500 of a TBC S-Curve controller is shown in Figure 23. In a first step 4502, the flag is checked to see if the current system status is "accelerate" or "decelerate." If the status is decelerate, operation proceeds to step 4512. On the other hand, if the current status is "accelerate", the stop_distance_required is calculated in step 4504. The stop_distance_required may be defined as the minimum distance that the tool would travel according to the S-curve before the system could bring the tool to a stop. The stop distance may be a function of current speed and acceleration. In step 4506, if the given distance_to_go "d" is greater than the stop_distance_required "d0," which means that it is safe to continue to move forward, then operation proceeds to step 4512 to calculate the desired acceleration and velocity. Otherwise, if the distance_to_go "d" is less than the stop_distance_required "d0," which means the system must brake immediately, then brake information is calculated and the flag is set to decelerate in step 4508. Because the distance_to_go "d" is not enough for the system to brake according to the original S-Curve (with preset maximum jerk and acceleration), maximum jerk may be modified in step 4510 so that the system does not overshoot. Two distances, d1 and d3, which may be used to detect the S-Curve phase, may also be updated in step 4510. Step 4512 may be considered the core of the S-Curve controller. According to the current system status and the critical distances calculated beforehand, the S-Curve phase, desired velocity and acceleration may be determined in step 4512. Because the S-Curve controller runs in real time, care may be taken to ensure that the controller does not run out of data. When the total moving distance of the motion commands at the queue becomes too short, the system may need to slowdown to prevent data starvation. In step 4516, on-the-fly feedrate adjustment and user override is applied.

E. TBC Feed Forward Control

1. Feed Forward Signals and Gains

[00138] Feed forward may be based on the ideal current position (ICP). The calculation of the three feed forward parts, i.e., normal acceleration FFan, tangential velocity FFvt, and tangential acceleration FFat, is shown in Figure 24, which illustrates velocity and acceleration feedforward. In Figure 24, "d" represents the distance_to_go obtained from the geometry analysis; and dt and dr represent unit tangential and radial/normal direction vectors, respectively.

[00139] S-Curve controller 4604 may receive the distance and calculate the desired velocity v and acceleration a (both v and a are scalars) along the trajectory.

[00140] The acceleration a on line 4606 may be multiplied by the unit tangential direction vector dt 4608 to produce the tangential acceleration vector at 4610. The feed forward gain, Kaff, may convert the tangential acceleration at into a control signal FFat, which is acceptable by the servo system.

[00141] The Modified Laplace block 4612 may estimate the current velocity, compare the current velocity with last velocity command, and add a predictive compensation into the S-curve velocity command v to reduce the tangential tracking error. The compensated velocity command multiplies the unit tangential direction vector dt 4608 and becomes the tangential velocity vector vt 4614. Because the servo system takes the velocity control commands, no further conversion may be needed for the velocity vector. A feed forward gain 1 at 4616 results in a signal FFvt and may enable all channels to be at the same level.

[00142] The centripetal acceleration an 4618 may be calculated by the circular motion physics, i.e., $a_n = v^2/R$. The centripetal acceleration may be directed toward the arc center and opposite of the radial/normal direction. The same feed forward gain Kaff as applied to the tangential acceleration may be applied to convert the normal acceleration into a suitable control signal FFan.

2. Runtime Feedrate Adjuster (μ)

[00143] The runtime or "on-the-fly" feedrate adjuster is a feature of TBC motion control. The TBC motion kernel may not have any predetermined reference trajectories, and desired motion may be determined inside the control loop at every control cycle based on the current system condition. Thus, the runtime feedrate adjuster may be possible.

[00144] The runtime feedrate adjuster may compare the position/tolerance error to the specified error tolerance. If the position error is bigger than, or too close to, the specified error tolerance, the runtime feedrate adjuster may slow down the feedrate to improve tracking accuracy. On the other hand, if the error is smaller than, and not too close to, the specified tolerance, the runtime feedrate adjuster may increase the feedrate to improve the system throughput. Generally, the speed of the tool may be increased if the position error is below the specified error tolerance, and the speed of the tool may be decreased if the position error is above the specified error tolerance. It is possible, in one embodiment, to not increase the feedrate even if the contour error is very small, but to retain the ability to either increase or reduce the feedrate as appropriate.

[00145] The runtime feedrate adjuster may be a constrained optimization mechanism. That is, under the constraint of the contour/tolerance error within a specified range, the runtime feedrate adjuster may maximize the feedrate to achieve optimal system throughput.

F. TBC Feedback Control

1. Normal Error Feedback Control with Variable Gain (τ)

[00146] The normal error e_n, or contour/tolerance error, may be defined as the shortest distance from the current tool position to the reference contour, and TBC motion control may reduce this error. Because this is a three-dimensional error, the coordinated motion of the three axes together may be needed to correct it. The error may also be five or six-dimensional when additional axes, in the form of rotational axes, are included in the trajectories and in the calculations. Thus, where three dimensions or three axes are referred to in the discussion below, it is to be understood that the discussion is equally applicable to multi-dimensional systems including more than three dimensions and more than three axes.

[00147] As indicated above, a difference between the TBC and the non-TBC control is that the non-TBC control typically decomposes the three-dimensional reference contour trajectory into three independent axis motions, and controls the three axes without any knowledge of the contour/tolerance error. The TBC motion control, in contrast, may directly address the three-dimensional contour/tolerance error. The actual position of the tool may be sensed along each of three axes. A three-dimensional position error between the actual position of the tool in each of the three axes and a desired position of the tool

in each of the three axes may then be calculated. A three-dimensional position error signal dependent upon the calculated three-dimensional position error may be produced. The speed and/or direction of the tool may be controlled dependent upon the three-dimensional position error signal. As described herein, and illustrated herein with various examples, three-dimensional position errors may be calculated by use of vector mathematics.

[00148] As is clear from the above description, TBC control may enable motion along one axis to affect motion control along the other two axes in real time, i.e., there may be intercoupling between the motion control of the different axes.

[00149] As shown in Figure 25, which illustrates the variable-gain feedback control of normal error, the TBC motion control may use a variable-gain feedback control to address the normal error. The error may be calculated in the real time geometry analysis 4002 (Figure 18). A special feedback controller Cn may provide system stability and reduce the normal error.

[00150] The variable gain τ 4702 may be a function of both tangential velocity and normal acceleration. Loosely speaking, the faster the system moves, the larger the gain that is needed to correct the normal error. The larger the gain, the more quickly the normal error may be corrected.

2. Tangential Error Feedback Control

[00151] At each control cycle, the TBC may generate a tangential velocity command. The integration of the tangential velocity commands may produce a position on the reference contour, which may be considered a current target point on the reference contour. The tangential error may be defined as the path length from the ideal current position (ICP) to the current target point. A feedback controller Ct 4004 (Figure 18) may reduce the tangential error.

3. Different Dynamic Responses of the Two Feedback Controls

[00152] The TBC feedback control may include two parts: tangential error feedback control and normal error feedback control. Instead of treating the two errors the same or equally, two feedback controllers may be provided for the errors, with each feedback controller addressing a respective one of the two errors. A first reason for the different treatment of the two errors is that the normal error may be more important than the tangential error in terms of tracking accuracy and surface finish quality, and thus more

weight may be placed on the normal error. A second reason for the different treatment of the two errors is that the servo system may have limited power, and it may be desirable to use the power to do more important things first. Thus, two different feedback controllers may be used for treating normal and tangential errors. The normal error loop may have higher bandwidth or faster dynamic response. Loosely speaking, more servo power may be used to correct normal error than tangential error, and the normal error may be reduced faster than the tangential error.

G. TBC Stick Friction and Backlash Compensation

[00153] The stick friction may have a considerable influence on the motion control. To correct the stick friction effect, the TBC motion kernel may use a stick friction compensator. This compensator may be triggered when an axis changes its motion direction, i.e., across a zero velocity point. The compensation profile may have a notch shape, with the height and width of the notch being a function of the velocity and the local curvature of the trajectory.

[00154] Leadscrews may be used in machine tools as a mechanical transmission to convert rotary motion into linear motion. The inaccuracy of the ball screw and backlash may need to be compensated. The position values used by the motion kernel may be all leadscrew and backlash compensated according to the leadscrew mapping tables.

[00155] The compensation block may determine the current moving direction, which may be needed for looking up the leadscrew mapping table to implement leadscrew/backlash position compensation. The compensation block may also compensate for the stick friction.

H. TBC Control Signal Protection

[00156] The TBC motion control signal may be the sum of the feed forward control signal, the feedback control signal and the stick friction compensation signal. Before sending the control signal to the servo system through digital-to-analog converters (DACs), protection procedure may be performed to avoid overloading the machine. The protection procedure may include two steps, including a first step of limiting the control signal changing rate. The servo system may receive the velocity command, and the control signal change rate may correspond to acceleration/deceleration. Although the S-Curve controller may take the acceleration/deceleration into account, it is still possible that the overall control signal may change too fast due to other reasons, such as tracking

error feedback, disturbances, stick friction compensation, and so on. In order to avoid exceeding the servo system capacity and for smooth motion, the control signal change rate may be limited. That is, if the difference between the current and previous control commands exceeds a threshold, the current command may be limited so that the difference is saturated at the threshold.

[00157] A second step of the protection procedure may be to limit the control signal. Both the DAC and the servo system may be capable of receiving only limited control signal input up to a maximum amount. If the current control command exceeds this threshold, then steps may be taken to saturate the current control command to the maximum control input.

[00158] One embodiment of a method of trajectory motion control of the present invention is shown in pages 1-16 of the TBC Flowcharts and pages 1-5 of the TBC Kernel Flowcharts of the attached Appendix I.

[00159] While this invention has been described as having an exemplary design, the present invention may be further modified within the spirit and scope of this disclosure. This application is therefore intended to cover any variations, uses, or adaptations of the invention using its general principles.

What is claimed is:

1. A method of computing velocity limitations for a plurality of motion commands provided to a motion controller to cause an object to move along a trajectory, including the steps of: providing a look-ahead queue including a plurality of motion commands, the plurality of motion commands including a first motion command; calculating a first stop distance for the first motion command, the first stop distance being a distance between a first target point and a first virtual point beyond the first target point; adding a second motion command to the plurality of motion commands in the look-ahead queue; calculating a second stop distance for the second motion command, the second stop distance being a distance between a second target point and a second virtual point beyond the second target point; using a computing device to modify a velocity limitation for the first motion command by reducing the first stop distance based on the second stop distance; and providing the first motion command to the motion controller.

2. The method of claim 1, wherein the step of modifying a velocity limitation, includes the steps of: calculating a sum of a length of a motion segment associated with the second motion command and the second stop distance; and comparing the first stop distance to the sum.

3. The method of claim 2, wherein the first stop distance is reduced if the first stop distance exceeds the sum.

4. The method of claim 3, wherein the first stop distance is reduced to equal the sum.

5. The method of claim 1, further comprising the steps of: adding a third motion command to the look-ahead queue; calculating a third stop distance associated with the third motion command; and determining if the second stop distance associated with the second motion command should be reduced based on the third stop distance, thereby modifying a velocity limitation for the second motion command.

6. A method of computing velocity limitations of motion segments having corresponding motion commands used by a motion controller for movement of a body along a trajectory, the motion segments including a first point having a first velocity and a target point having a target velocity, including the step of using a computing device to determine a virtual point beyond the target point having an end velocity, the virtual point being determined such that the trajectory includes the first point with the first velocity, the target point with the target velocity, and the virtual point with the end velocity, wherein the target velocity is greater than the end velocity and the first velocity is greater than the target velocity.

7. The method of claim 6, wherein the end velocity is zero.

8. The method of claim 6, wherein the trajectory is an S-curve velocity profile.

9. The method of claim 6, wherein the body passes through a plurality of positions

between the first point and the target point and the trajectory has a unique speed value for each position of the body from the first point to the target point.

10. The method of claim 6, further including the step of determining a first distance between the target point and the virtual point.

11. The method of claim 6, wherein the motion segments further include a third point having a third velocity.

12. The method of claim 11, further including the steps of determining a second virtual point having a second end velocity, the second virtual point being determined such that the trajectory includes the target point with the target velocity, the third point with the third velocity, and the second virtual point with the second end velocity.

13. The method of claim 12, further including the steps of determining a first distance between the target point and the virtual point, determining a second distance between the third point and the second virtual point, and determining whether the first distance is sufficient to permit a change from the first velocity at the first point to the target velocity at the target point.

14. The method of claim 13, wherein the first distance is determined to be sufficient to permit the change from the first velocity at the first point to the target velocity at the target point if the first distance is less than or equal to the sum of the second distance and a length from the target point to the third point.

15. The method of claim 13, wherein if the first distance is greater than the sum of the second distance and a length from the target point to the third point, the first distance is determined to be insufficient to permit the change from the first velocity at the first point to the target velocity at the target point.

16. The method of claim 15, wherein if the first distance is greater than the sum of the second distance and a length from the target point to the third point, the first distance is reduced to a distance sufficient to permit the change from the first velocity at the first point to the target velocity at the target point.

17. The method of claim 16, wherein if the first distance is greater than the sum of the second distance and a length from the target point to the third point, the first distance is reduced to be equal to the sum.

18. The method of claim 6, wherein the body is a cutting head of a CNC machine.

19. The method of claim 6, further including the step of: providing motion commands to the motion controller for controlling a current velocity of the body along the trajectory, the providing step including the steps of comparing at least two of a specified velocity; a machine allowable velocity; and a path velocity limit; and setting the current velocity to a value not exceeding the lowest of the at least two of the specified velocity; the machine

allowable velocity; and the path velocity limit.

20. The method of claim 19, wherein the current velocity at the target point does not exceed a joint velocity limit.

21. A method of computing velocity limitations for movement of a body along a contour having a plurality of segments, including the steps of sequentially storing in a queue a stop distance and a length associated with each segment, the stop distance being a distance between a target point and a virtual point beyond the target point; and if a current stop distance associated with a current segment is greater than a sum of a subsequent stop distance and a subsequent length associated with a subsequent segment, then using a computing device to set the current stop distance equal to the sum and to verify that a prior stop distance associated with a prior segment is less than or equal to the current stop distance plus a current length associated with the current segment.

22. The method of claim 21, further including the step of recursively checking whether each stop distance is less than or equal to a subsequent sum of a stop distance and a length associated with a subsequent segment and reducing any stop distance in the queue that is greater than the subsequent sum; and stopping the recursively checking step upon encountering a stop distance that is less than or equal to the sum of the stop distance and the length associated with the subsequent segment.