

METHOD OF TOLERANCE-BASED TRAJECTORY PLANNING AND CONTROL
BACKGROUND OF THE INVENTION

1. 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.

2. 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. One of the operator's functions is to select feedrates that would result in acceptable part quality, while simultaneously achieving high metal removal rates. The selection of appropriate feedrates is based on the operator's experience, and general rules of thumb may be obtained from numerous handbooks and charts (e.g., Machinery's Handbook, 24th edition, Industrial Press Inc., New York 1992). However, the figures from such documents, while perfectly feasible when used under the correct conditions, are frequently inappropriate when applied to specific machining situations. Published figures fail to account for local machining conditions such as sudden changes in the tool path, thereby leaving optimization of the feedrate to the operator. It is difficult for an operator to select feedrate values that achieve the desired part quality while maximizing the machine tool's productivity throughout the part program.

[0008] A tolerance based motion control system, including a method for setting feedrates based upon tolerance restraints, 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 in the use of tolerance commands, E codes, and tolerance regions 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 of the present invention, the method 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 Tolerance Control may calculate feedrate limitations based on the CNC tolerance specified by the programmers and the programmed tool path, specifying the maximum, or near maximum, allowable feedrate at each point along the programmed trajectory. This information along with position feedback may be used to modify the feedrate along the actual trajectory of motion to limit position deviations from the ideally programmed path so that the requested CNC tolerance may be satisfied.

[0012] Unlike the classical Proportional, Integral and Derivative (PID) approach, which includes guessing where the cutter tool will be at various time intervals, 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, thereby eliminating the need for following error terms in the control loop. 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. Where a non-TBC PID control loop approximates the programmed trajectory by pulling the cutter tool from preplanned target to target using the following error, the TBC control pushes the cutter tool along the desired trajectory, correcting for deviations as they are encountered. TBC control may include controlling or adjusting movements of the tool based upon the current tracked position of the

tool vis-à-vis a desired tool trajectory, rather than basing future tool movements solely upon where the tool should be at a certain point in time according to a pre-planned itinerary.

[0013] In the "true arc" technology employed by the present invention, tool motion is controlled along a trajectory that is modeled as an arcuate path without polygon approximation. That is, the tool may be commanded to follow an arcuate trajectory rather than a trajectory formed by a series of linear segments that approximate a curved trajectory. By use of arcuate 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.

[0014] 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. The smoothing technology is particularly important with legacy part programs expressing complex part geometry using polygon approximations.

[0015] 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.

[0016] 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.

[0017] The Soft-motion technology is the basis of the TBC control, which is easily updated and eliminates the need for costly PID based motion control cards.

[0018] The TBC Prototype Software System may include Win32 components and Real Time Software System (RTSS) components. The term "Win32" refers to an application program interface (API) for running 32-bit Windows applications under Windows NT and Windows XP, as is commonly known, although other interfaces may be used with other underlying operating systems. Win32 Components may be interface modules for processing user requests, providing status information back to the users, updating graphics display, and converting a subset of G&M code into the required forms. Win32 components may include man-machine interface components and an interpreter of a subset of G&M code. The man-machine interface components may include display software and control panel processing.

[0019] The machine I/O software components may include drivers for the servo transducer board. The Machine I/O Servo Driver may be responsible for taking the control signals generated by the TBC control and processing them into electrical signals effecting motion. This driver may also take feedback signals from the machine tool and convert them into information used by the TBC software.

[0020] RTSS Components may be coded within the RTSS, ensuring deterministic response. The RTSS components may include TBC components, machine I/O software components, and component integration and communication components. The TBC

components may include tolerance understanding components, data compression components, data smoothing components, lookahead components, and motion kernel components. The TBC components may be functionally divided into two groups, i.e., a TBC data preprocessing group and a TBC motion control group including TBC motion kernel. The TBC data preprocessing group may include tolerance understanding components, data compression components, data smoothing components, and lookahead components. The TBC data preprocessing group may be implemented in a queue module.

[0021] The smoothing function includes three phases: prepare for smoothing, adjust for smoothing and double arc smoothing. After the three phases of smoothing operations, point-to-point moves are converted into a sequence of mutually tangential arcs within allowed tolerance and with relatively small curvature changes.

[0022] The invention comprises, in one form thereof, a method for adjusting path data for a motive body, including identifying three consecutive points of the path data. A first distance between a circular arc defined by the three points and a middle portion of a first line extending between a first of the points and a second of the points is ascertained. A second distance between the circular arc and a middle portion of a second line extending between the second point and a third of the points is ascertained. A turn angle between the first line and the second line is determined. Smoothing is performed on the second point if the first distance and/or the second distance is less than a threshold line distance, and the turn angle is greater than a threshold angle.

[0023] The invention comprises, in another form thereof, a method of adjusting path data of a motive body, including identifying four consecutive points of the path data. The four consecutive points include a first point, a second point, a third point and a fourth point. It is determined whether the second point is suitable for smoothing. If the second point is suitable for smoothing, then a first arc and a second arc are located, and the second point is moved into an area between the first arc and the second arc. The first arc is defined by the first point, the second point and the fourth point. The second arc is defined by the first point, the third point and the fourth point. A line move associated with the second point is replaced with at least one arc move associated with the second point.

[0024] The invention comprises, in yet another form thereof, a method for adjusting path data for a motive body, including identifying four consecutive points of the path data. The four consecutive points include a first point, a second point, a third point and a fourth point. A first arc defined by the first point, the second point and the fourth point is located. A second arc defined by the first point, the third point and the fourth point is located. The second point is moved into an area between the first arc and the second arc.

[0025] The invention comprises, in a further form thereof, a method for adjusting path data for a motive body, including identifying a line move extending from a first point of the path data to a second point of the path data. The line move is replaced with a first arc move and a second arc move. The first arc move begins at the first point and terminates at a junction point. The second arc move begins at the junction point and terminates at the second point.

[0026] The invention comprises, in still another form thereof, a method of processing path data for a motive body such as a machine tool, including identifying a plurality of points along a target tool path, and determining a maximum allowable stopping distance of the tool at a last of the points. A last maximum allowable speed of the tool at the last point such that the tool may come to a stop within the maximum allowable stopping distance is ascertained. An other maximum allowable speed of the tool at an other one of the points such that the tool may possibly slow down to the last maximum allowable speed upon arriving at the last point is established.

[0027] The invention comprises, in another form thereof, a method of processing path data for a machine tool, including identifying a plurality of points along a target tool path, and establishing a maximum allowable level of deviation of an actual tool path from the target tool path. A degree of curvature of the target tool path at a plurality of locations along the target tool path is ascertained. A maximum allowable tolerance speed of the tool at each of the points along the target tool path is determined, wherein the maximum allowable tolerance speed is dependent upon the maximum allowable level of deviation and the degree of curvature. A maximum allowable stopping distance of the tool at a last of the points is determined. A last maximum allowable stopping speed of the tool at the last point such that the tool may come to a stop within the maximum allowable stopping distance is ascertained. An other maximum allowable stopping speed of the tool at each of the points other than the

last point such that the tool may possibly slow down to the last maximum allowable speed upon arriving at the last point is established. A lesser of the maximum allowable tolerance speed and the maximum allowable stopping speed at each of the points is identified.

[0028] The invention comprises, in yet another form thereof, a method of processing path data for a machine tool, including identifying a plurality of points along a target tool path. Each selected two of the points being separated by a respective segment length. A maximum allowable stopping distance of the tool at a last of the points is determined. A maximum allowable speed of the tool at an other one of the points such that a required stopping distance of the tool at the other point is equal to or less than a sum of the maximum allowable stopping distance at the last point and the segment length between the other point and the last point is ascertained.

[0029] The invention comprises, in a further form thereof, a method of operating a machine tool, including identifying a target tool path. The tool is commanded to follow the target tool path from an initial position. A first actual position of the tool is sensed after the commanding step. A first target tool subpath from the first actual position is calculated to correct for predictive errors in movement of the tool.

[0030] The invention comprises, in still another form thereof, a method of operating a machine tool, including empirically determining predictive errors in movement of the tool. Actual movements of the tool are monitored. The actual movements of the tool are controlled dependent upon the empirically determined predictive errors and random errors in the actual movements of the tool.

[0031] The invention comprises, in a still further form thereof, a method of operating a machine tool having a plurality of sequential movements, including sensing an actual position of the tool after a first of the movements of the tool. A target position of the tool is calculated based on the sensing step. The target position is calculated to compensate for predictive errors and random errors in the movements of the tool. The sensing and calculating steps are repeated for other ones of the movements of the tool.

[0032] The invention comprises, in yet another form thereof, a method of operating a machine tool, including sensing an actual position of the tool. A position error between the actual position of the tool and a desired position of the tool is calculated. The position error

is compared to a specified error tolerance. A speed of the tool is adjusted dependent upon the comparing step.

[0033] The invention comprises, in another form thereof, a method of operating a machine tool, including sensing an actual position of the tool along each of a plurality of axes. A multi-dimensional position error between the actual position of the tool in each of the axes and a multi-dimensional reference trajectory is calculated. A multi-dimensional position error signal is produced dependent upon the calculating step. A speed and/or a direction of the tool are controlled dependent upon the multi-dimensional position error signal.

[0034] 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.

BRIEF DESCRIPTION OF THE DRAWINGS

[0035] 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:

[0036] Figure 1 is a flow chart of one embodiment of a tolerance based control method of the present invention.

[0037] Figure 2 is a block diagram of one embodiment of a tolerance based control system of the present invention.

[0038] Figure 3a is a diagram illustrating the mapping of different tolerance values to different regions of machining space.

[0039] Figure 3b is a diagram illustrating a first case of another embodiment of the mapping of different tolerance values to different regions of machining space.

[0040] Figure 3c is a diagram illustrating a second case of another embodiment of the mapping of different tolerance values to different regions of machining space.

[0041] Figure 3d is a diagram illustrating yet another embodiment of the mapping of different tolerance values to different regions of machining space in the case of an arc.

[0042] Figure 4a is a plot of program data points, illustrating a method of data compression according to one embodiment of the present invention.

- [0043] Figure 4b is a plot of program data points, illustrating a method of data compression according to another embodiment of the present invention.
- [0044] Figure 5a is a plot of program data points, illustrating a method of selecting data points for data smoothing according to one embodiment of the present invention.
- [0045] Figure 5b is a plot of program data points, illustrating a method of selecting data points for data smoothing according to another embodiment of the present invention.
- [0046] Figure 6a is a plot of program data points, illustrating a method of adjusting line moves for smoothing according to one embodiment of the present invention.
- [0047] Figure 6b is a perspective view of a plot of program data points, illustrating a method of adjusting line moves for smoothing according to another embodiment of the present invention.
- [0048] Figure 7 is an enlarged plot of the middle two points of Figure 6a.
- [0049] Figure 8 is a plot of the program data points of Figure 6a along with one subsequent program data point, illustrating a next step of the method of Figure 6a for adjusting line moves for smoothing.
- [0050] Figure 9 is a plot of program data points, illustrating a method of smoothing data using a double arc algorithm according to one embodiment of the present invention.
- [0051] Figure 10 is a plot of the tool velocity versus time, also referred to as the S-Curve.
- [0052] Figure 11 is a plot of a segment length extending between two consecutive points along a tool path.
- [0053] Figure 12 is a plot illustrating the deviations of an actual tool path from a desired tool path.
- [0054] Figure 13 is a plot of control loop cycle time for the servo signals in one embodiment of the present invention.
- [0055] Figure 14 is a plot of program data points, illustrating a method of predictive error correction according to one embodiment of the present invention.
- [0056] Figure 15 is a plot of program data points, illustrating a method of random error correction according to one embodiment of the present invention.
- [0057] Figure 16 is a flow chart illustrating a method of the present invention for executing a motion kernel.

[0058] Figure 17 is a block diagram of one embodiment of a machine of the present invention for machining a work piece.

[0059] Figure 18 is a flow chart of one embodiment of the method of the present invention illustrated in Figure 5.

[0060] Figure 19 is a flow chart of one embodiment of the method of the present invention illustrated in Figures 6 and 7.

[0061] Figure 20 is a flow chart of one embodiment of the method of the present invention illustrated in Figure 9.

[0062] Figure 21 is a flow chart of one embodiment of the method of the present invention illustrated in Figures 5 through 9.

[0063] Figure 22 is a flow chart of one embodiment of the method of the present invention illustrated in Figure 11.

[0064] Figure 23 is a flow chart of another embodiment of the method of the present invention illustrated in Figure 11.

[0065] Figure 24 is a flow chart of yet another embodiment of the method of the present invention illustrated in Figure 11.

[0066] Figure 25 is a flow chart of one embodiment of the method of the present invention illustrated in Figure 12.

[0067] Figure 26 is a flow chart of another embodiment of the method of the present invention illustrated in Figure 12.

[0068] Figure 27 is a flow chart of yet another embodiment of the method of the present invention illustrated in Figure 12.

[0069] Figure 28a is a graphical representation of Level 0 data smoothing.

[0070] Figure 28b is a graphical representation of Level 1 data smoothing.

[0071] Figure 28c is a graphical representation of Level 2 data smoothing.

[0072] Figure 28d is a graphical representation of Level 3 data smoothing.

[0073] Figure 28e is a graphical representation of Level 4 data smoothing.

[0074] Figure 29 is a plot illustrating one step of one embodiment of a double arc smoothing algorithm of the present invention.

[0075] Figure 30 is a plot illustrating another step of one embodiment of a double arc smoothing algorithm of the present invention.

[0076] Figure 31 is a plot illustrating yet another step of one embodiment of a double arc smoothing algorithm of the present invention.

[0077] Figure 32a is another plot of the tool velocity versus time, also referred to as an S-Curve.

[0078] Figure 32b is a diagram indicating how distances are related to the plot of Figure 32a.

[0079] Figure 33 is a diagram illustrating the concept of stop distance.

[0080] Figure 34 is yet another plot of the tool velocity versus time for both additive and non-additive lookahead algorithms.

[0081] Figure 35a is a flow chart of one embodiment of an additive lookahead algorithm of the present invention.

[0082] Figure 35b is a flow chart of one embodiment of a non-additive lookahead algorithm.

[0083] Figure 36 is a table comparing the kinematic characteristics of a non-additive lookahead algorithm and a TBC additive lookahead algorithm.

[0084] Figure 37 is a block diagram of one embodiment of a non-TBC motion control arrangement of the present invention.

[0085] Figure 38a is a plot of a reference trajectory $y=x$.

[0086] Figure 38b is a plot of the reference trajectory of Figure 38a decomposed into a time-based y-axis trajectory $y=kt$.

[0087] Figure 38c is a plot of the reference trajectory of Figure 38a decomposed into a time-based x-axis trajectory $x=kt$.

[0088] Figure 38d is a plot of actual trajectories resulting from the reference trajectories of Figures 38a-c.

[0089] Figure 39 is a block diagram of one embodiment of a TBC motion control arrangement of the present invention.

[0090] Figure 40 is another block diagram of the TBC motion control arrangement of Figure 39.

[0091] Figure 41 is a plot illustrating the calculation of the stopper plane normal direction for line motion.

[0092] Figure 42 is a plot illustrating the calculation of the stopper plane normal direction for arc motion.

[0093] Figure 43 is a plot illustrating line motion geometry analysis.

[0094] Figure 44a is a first plot illustrating arc motion geometry analysis.

[0095] Figure 44b is a second plot illustrating arc motion geometry analysis.

[0096] Figure 45 is a flow chart illustrating operation of one embodiment of a TBC S-Curve Controller.

[0097] Figure 46 is a block diagram illustrating one embodiment of a TBC feed forward control arrangement of the present invention.

[0098] Figure 47 is a block diagram illustrating one embodiment of an arrangement of the present invention for variable-gain feedback control of normal error.

[0099] 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

[00100] 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 motive 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.

[00101] 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.

[00102] 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. The quality of the data in the user part program may be directly linked to the quality of the cutting results on a machine tool. Data of poor quality often results in a part surface that is not smooth.

[00103] 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. The data processing includes the tolerance understanding operation on the tolerance queue, the data compressing operation on the compression queue, the smoothing operation on the smoothing queue, and the lookahead operation on the lookahead queue.

[00104] The queue module 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.

[00105] 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 lid

[00106] 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.

[00107] It is possible for multiple E codes to define overlapping tolerance regions. One point may belong to a number of active NC tolerance regions. In this case, the NC tolerance value at the point is defined as the tolerance value specified in the latest E code whose region covers this point.

[00108] In a tolerance mapping operation within tolerance queue 202, tolerance values are assigned to each line or arc within the tool path. The tolerance mapping operation processes variable tolerance regions that are defined in the part program and assigns the required tolerances to each motion move. Tolerance regions having different values may overlap one another, as shown in Figure 3a. Three areas 302, 304, 306 have respective tolerances specified as 0.001 inch, 0.0005 inch, and 0.0003 inch. As a move such as from P_0 to P_1 crosses multiple tolerance regions, the move is split into multiple moves, i.e., from P_0 to P_1' , from P_1' to P_1'' , and from P_1'' to P_1 , with each move having a different tolerance value. The tolerance command E codes in a part program may specify regions and tolerance values to be applied to those regions in the part. The motion commands in the part program may fall

into different tolerance regions. Tolerance understanding may identify the correct tolerance value required for each motion command according to the programmed E codes.

[00109] In processing a tolerance command, if the command is a new tolerance specification, a space may be created for a new item in a tolerance table. In the tolerance table, tolerance values are mapped to respective tolerance regions. Then, the new item may be filled in with the tolerance specification extracted from the command. Finally, the new item may be inserted at the beginning of the tolerance table. After being processed, a tolerance command may be discarded, because the information from the command has been stored in the tolerance table.

[00110] In processing a line command or an arc command, the required tolerance value for the command may be calculated and attached to the command. The E codes may specify different tolerance regions, and each line or arc of the NC program may belong to one or more of these tolerance regions. The algorithm for determining the required tolerance for each line or arc may include checking the tolerance of some sample points on the line or arc. If a line or arc lies in one tolerance region, the sample points will have the same tolerance, which will be the required tolerance for this line or arc. On the other hand, if a line or arc lies across more than one tolerance region, the sample points will have different tolerances. In this case, the line will be divided into segments based on the sample points, with each line segment having a unique required tolerance. By the same token, an arc may be segmented by different tolerances, however, the arc will not be divided. The tightest tolerance value will be chosen as the required tolerance for this arc instead.

[00111] A routine may be used to determine the required tolerance for a given point. It performs the following operations: 1) take the position of the given point as a key value; 2) starting from the beginning, search the tolerance table for the E code whose tolerance region covers the key value (i.e., the given point is located within the tolerance region); and 3) assign the tolerance value of the E code first found as the required tolerance for the given point.

[00112] If a point belongs to several tolerance regions specified by different tolerance commands, the latest tolerance command is in control. A new tolerance command is always

inserted at the beginning of the tolerance table. Therefore, in the above operation, the first found E code is the latest E code which specifies the required tolerance for the given point.

[00113] Another routine may be used to divide a line into segments according to multiple tolerance specifications. Yet another routine may be used to determine the tightest tolerance among the multiple tolerance specifications on an arc.

[00114] Another embodiment of tolerance regions having different values and overlapping one another is illustrated in Figures 3b-c, Figure 3b illustrating a first case, and Figure 3c illustrating a second case. In this embodiment, 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.

[00115] In the case of an arc (Figure 3d), 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 $A1.tol = \min \{t1, t2, t3, t4, t5\}$.

[00116] 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.

DATA COMPRESSION

[00117] 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 P₁ and P₂ are eliminated and a new line is formed from point P₀ to point P₃. 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.

[00118] 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.

[00119] To do data compression, all line commands may be sequentially examined in compression queue 204. The compression operation may process point-to-point moves in the following steps: 1) examine a sequence of point-to-point moves with the consecutive target points $p_0, p_1, p_2, p_3, \dots, p_{n-1}, p_n$; and 2) remove the points between p_0 and p_n in the sequence, i.e., $p_1, p_2, p_3, \dots, p_{n-1}$, if the following conditions are all met:

- For all i ($1 \leq i < n$), the distance from p_i to the straight line connecting p_0 and p_n is less than a given tolerance
- For all i ($1 \leq i < n$), the perpendicular projection point of p_i on the line connecting p_0 and p_n is located between p_0 and p_n .
- For all i ($0 \leq i \leq n$), the line command with p_i as the end position have the same tolerance value
- The distance from p_0 to p_n is bounded by a predetermined length

[00120] Thus, the line move from p_0 to p_n replaces n shorter line moves. After the compression operation, the elements in compression queue 204 may be more compact since some consecutive short lines have merged. All the elements in compression queue 204, including positioning commands, compressed line commands, arc commands, and the program end command, are sent to smoothing queue 206.

[00121] In another data compression embodiment, 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 p_0, p_1, p_2 and p_3 have already been found to meet the compression criteria, and that it is being determined whether point p_4 should be added to the compression queue, or whether compression should be performed on only points p_0, p_1, p_2 and p_3 . The four compression criteria may be as follows:

Test 1: $|p_4 - p_0| < 0.05$ inch.

Test 2: For p_3 , the angle θ is between -90 degrees and 90 degrees.

Test 3: $p_0.tol = p_1.tol = p_2.tol = p_3.tol = p_4.tol$. That is, an equal tolerance has been assigned to each point from the tolerance table.

Test 4: The perpendicular distances from p_1 , p_2 and p_3 to the line p_0p_4 are less than the specified tolerance.

[00122] If the above four tests are satisfied, then p_4 is added to the compression queue. If the above four tests are not satisfied, then p_3 is added to the smoothing queue and p_1 and p_2 are discarded in compression. Essentially, three segments (p_0p_1 , p_1p_2 and p_2p_3) are combined into one segment (p_0p_3).

DATA SMOOTHING

[00123] 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 true arc technology.

[00124] The following different smoothing levels may be provided: 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. The preparatory smoothing operation, the corrective smoothing operation, and the double arc smoothing may be performed on the smoothing queue simultaneously and in a pipeline manner.

[00125] Figures 28a-e illustrate level 0, level 1, level 2, level 3, and level 4 smoothing, respectively. In level 0-3 smoothing, the output of smoothing, the final trajectory, always passes through the data points. In Figure 28d, c_1 and r_1 are the center and radius of the arc defined by the points p_0 , p_1 and p_2 ; and c_2 and r_2 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 28d, the relatively large radius change from r1 to r2 results in a correspondingly large curvature change between the two arcs.

[00126] 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.

[00127] The points may be adjusted, both in direction and magnitude, to achieve gradual curvature changes such that the trajectory approaches being elliptical. Gradual change of curvature along the trajectory may facilitate motion control. Gradual change of curvature may be characterized by the curvature changing, or at least possibly changing, after each point along the trajectory. As shown in Figure 28e, the radius of curvature becomes incrementally smaller along the trajectory from p0 to p4. That is, the radii become smaller with each point along the trajectory. In Figure 28d, in contrast, the radius of curvature drops in one large step as the trajectory passes through p2. However, it may also be possible within level 4 smoothing for the curvature to be the same on either side of a point. As indicated by the overlapping points in Figure 28e, the points may be adjusted or moved such that the trajectory approaches an elliptical path. However, the points may be adjusted only within a specified tolerance.

[00128] 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.

[00129] Figure 28e illustrates minor adjustments to the positions of points p0 to p4 as compared to their earlier positions as shown in Figure 28d. 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, or much of 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.

[00130] 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.

[00131] 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 micro seconds. 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 in updating the motion control with a particular frequency, such as every 200 usec.

[00132] 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.

Prepare for Smoothing

[00133] 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".

[00134] Let p_1 , p_2 , and p_3 be three consecutive target points in the smoothing queue. The point p_2 may be tagged as “smoothing” if the following conditions are all met:

- The motion from p_1 to p_2 is a line move
- The motion from p_2 to p_3 is a line move
- The length from p_1 to p_2 is bounded by a constant
- The length from p_2 to p_3 is bounded by a constant
- The perpendicular distance from p_1 to line p_2p_3 is bounded by a constant
- The perpendicular distance from p_3 to line p_1p_2 is bounded by a constant
- The angle between line p_1p_2 and line p_2p_3 is bounded by a constant

[00135] Otherwise, the point p_2 may be tagged as “no smoothing”. The line elements in smoothing queue 206 processed by the first operation may be all tagged. In one embodiment, the length from p_1 to p_2 and the length from p_2 to p_3 are both bounded by a constant of approximately between 1 cm and 2 cm; the perpendicular distance from p_1 to line p_2p_3 is bounded by a constant of approximately between 20% and 40% of the length from p_2 to p_3 ; the perpendicular distance from p_3 to line p_1p_2 is bounded by a constant of approximately between 20% and 40% of the length from p_1 to p_2 ; and the angle between line p_1p_2 and p_2p_3 is bounded by a constant approximately between 135° and 165° .

[00136] In a data smoothing operation within smoothing queue 206, line data is converted into arcs that are connected and tangent to one another and that have gradually changing curvature or radius. This provides smoother motion velocity and acceleration that results in a better part surface finish. The process of data smoothing includes a preparing for smoothing step 210 (Figure 2), an adjusting for smoothing step 212, and a double arc smoothing step 214. Smoothing preparation step 210 includes identifying data points for smoothing. As illustrated in Figure 5a, point P_1 may be suitable for smoothing, and thus may be marked for smoothing, if the length of the moves P_0 to P_1 and P_1 to P_2 , and an angle 502 between these moves, are below specified limits. A maximum distance that point P_0 can be disposed from point P_1 while keeping the length of the move P_0 to P_1 below specified limits is indicated by dashed line 504. Similarly, a maximum distance that point P_2 can be disposed from point P_1 while keeping the length of the move P_1 to P_2 below specified limits is indicated by dashed line 506. Although the distance from P_1 indicated by dashed line 504

is smaller than the distance from P_1 indicated by dashed line 506, these distances may be equal. In one embodiment, point P_1 may be suitable for smoothing if the length of the moves P_0 to P_1 and P_1 to P_2 , i.e., the first line distance between points P_0 and P_1 and the second line distance between points P_1 to P_2 , are both below a limit or threshold line distance of approximately between one centimeter and two centimeters, and turn angle 502 is above a limit or threshold angle approximately between 135° and 165° .

[00137] In another embodiment, a further requirement for marking point P_1 for smoothing is that the perpendicular distance from point P_0 to line P_1P_2 , and the perpendicular distance from point P_2 to line P_0P_1 , are below specified limits or threshold perpendicular distances. In one embodiment, point P_1 may be marked for smoothing if the perpendicular distance d_0 from point P_0 to line P_1P_2 is less than approximately between 20% and 40% of the length of line P_1P_2 , and if the perpendicular distance d_2 from point P_2 to line P_0P_1 is less than approximately between 20% and 40% of the length of line P_0P_1 .

[00138] In yet another embodiment, which is illustrated in Figure 5b, three points p_0 , p_1 , p_2 are joined by an arc. A distance d_1 is the shortest distance between a middle portion of line segment p_0p_1 and the arc $p_0p_1p_2$; and a distance d_2 is the shortest distance between the middle portion of line segment p_1p_2 and the arc $p_0p_1p_2$. In one embodiment, distance d_1 is the shortest distance between a midpoint of line segment p_0p_1 and the arc $p_0p_1p_2$; and a distance d_2 is the shortest distance between the midpoint of line segment p_1p_2 and the arc $p_0p_1p_2$. A middle one of the three points, i.e., point p_1 , may be marked for smoothing if the following three conditions or criteria are satisfied:

Test 1: Both p_0p_1 and p_1p_2 are lines.

Test 2: $\text{Max}\{d_1, d_2\} < 0.0005$ inch. That is, both d_1 and d_2 are less than some predetermined distance approximately between 0.0002 and 0.001 inch, such as 0.0005 inch.

Test 3: The angle $\theta > 145$ degrees, or some other predetermined angle approximately between 130 degrees and 160 degrees.

If the three above conditions are not met, then point p_1 may not be marked for smoothing.

[00139] Figure 18 illustrates a method 1800 of the present invention as described above with reference to Figure 5b. In a first step 1802, three consecutive points of a data path are identified. For example, points p_0 , p_1 and p_2 may be identified. In step 1804, a first line

distance between a first of the points and a second of the points and/or a second line distance between the second point and a third of the points are ascertained. More particularly, a first line distance 518 between points p_0 and p_1 , and a second line distance 520 between points p_1 and p_2 is ascertained. In a next step 1806, a turn angle between a first line extending between the first point and the second point and a second line extending between the second point and the third point is determined. In the example of Figure 5b, a turn angle θ between the line extending between points p_0 and p_1 and the line extending between points p_1 and p_2 is determined. As indicated in steps 1808, 1810 and 1812, smoothing is performed on the second point if the first line distance and/or the second line distance is less than a threshold line distance, and the turn angle is greater than a threshold turn angle. That is, smoothing may be performed on point p_1 if first line distance 518 and/or second line distance 520 is less than a threshold line distance, and turn angle θ is greater than a threshold turn angle. Else, another three consecutive data points of the data path may be identified, as indicated in step 1802. For example, consecutive points p_1 , p_2 and a next point p_3 (not shown) may be identified.

Adjust for Smoothing

[00140] 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.

[00141] The second operation may include sequentially checking a group of four consecutive line target points already processed by the first operation in smoothing queue 206 and adjusting the positions of middle points to modify the local curvature. Let p_1 , p_2 , p_3 and p_4 be four consecutive line target points processed by the first operation in smoothing queue 206. Assume that points p_2 and p_3 are both tagged "smoothing", then these two

middle points will be slightly moved. The algorithm for adjusting point positions as illustrated in Figure 6 may include the following steps: 1) create an arc ARC_{134} passing through points p_1 , p_3 and p_4 ; 2) create an arc ARC_{124} passing through points p_1 , p_2 and p_4 ; 3) move point p_2 towards arc ARC_{134} and keep the moving distance within the given tolerance; and 4) move point p_3 towards arc ARC_{124} and keep the moving distance within the given tolerance.

[00142] The positional adjustment proceeds continuously with groups of four consecutive points.

[00143] After one group is adjusted, the successive group that consists of the last three points in the previous group and a new target point of the next line element will be processed. The target points of line elements in smoothing queue 206 processed by the second operation are moved (within the given tolerance range) so that the curvatures at these points change more smoothly, i.e., less drastically, than they did before the second operation.

[00144] Step 212 includes 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. In the illustration in Figure 6a, two construction arcs 602, 604 are located that pass through points $P_0P_1P_3$ and $P_0P_2P_3$, respectively. Points P_1 and P_2 may both be moved into an area between construction arcs 602, 604. Points P_1 and P_2 are moved closer to one another such that point P_2 is moved approximately midway between construction arcs 602, 604, and point P_1 is moved a lesser distance toward construction arc 604. That is, point P_1 may not be moved so far as to be positioned midway between construction arcs 602, 604. Rather, point P_1 may be moved less than half of the distance to the opposite construction arc 604. More particularly, point P_1 may be moved a distance equal to a factor between 0 and 0.5 multiplied by a spacing S_1 (Figure 7) between the point P_1 and the opposite construction arc 604. The factor may be less than 0.45. In the embodiment illustrated in Figure 6a, the factor is approximately between 0.18 and 0.20, which has been established mathematically as providing favorable results.

[00145] Figure 7 is an enlarged view that illustrates with more precision the direction and distance of the movement of points P_1 and P_2 in one embodiment. A line 702 intersects

point P_2 and is tangent to arc 604 at point P_2 . A spacing S_2 is defined between point P_2 and arc 602 in a direction perpendicular to line 702. Point P_2 may be moved a distance equal to a factor between 0.45 and 0.55 multiplied by spacing S_2 in a direction perpendicular to line 702. In the embodiment illustrated in Figures 6a and 7, the factor is approximately 0.50. Similarly, a line 704 intersects point P_1 and is tangent to arc 602 at point P_1 . A spacing S_1 is defined between point P_1 and arc 604 in a direction perpendicular to line 704. Point P_1 may be moved a distance of approximately 0.19 multiplied times spacing S_1 in a direction perpendicular to line 704.

[00146] The process may then be repeated for each successive move in the programmed path. Figure 8 illustrates a repetition of the process after a next point P_4 has been identified. Two construction arcs 802, 804 are defined that pass through points $P_1P_2P_4$ and $P_1P_3P_4$, respectively. Points P_2 and P_3 are then moved by factors of 0.19 and 0.50, respectively, in a manner substantially similar to the movements of points P_1 and P_2 in Figures 6a and 7.

[00147] Figure 6b illustrates another 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 c_2 ; and the center of a circle defined by arc 614 is designated c_1 . A projection of point p_1 on the plane of arc 614 is designated point q_1 . That is, a line that extends through both of points p_1 and q_1 is perpendicular to the plane of arc 614. Similarly, a projection of point p_2 on the plane of arc 612 is designated point q_2 . That is, a line that extends through both of points p_2 and q_2 is perpendicular to the plane of arc 612. The closest point on arc 614 to point p_1 is designated s_1 ; and the closest point on arc 612 to point p_2 is designated s_2 .

[00148] 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 p_0p_1 and p_1p_2 are lines; $\text{Max}\{d_1, d_2\} < 0.0005$ inch; and angle $\theta > 143.2$ degrees. In the example shown in Figure 6b, point p_1 may be adjusted if p_1 has been marked for smoothing and either or both of p_0 and p_2 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.

[00149] 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. As with the embodiment of Figures 6a and 7, the adjustment process may continue indefinitely with calculations based upon subsequent sets of four points along the trajectory.

[00150] 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 $p = p + 0.5 \cdot v$. 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 6b. 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.

[00151] Adjustment factors of 0.19 and 0.5 are discussed above in conjunction with the embodiment of Figure 6b. However, as with the embodiment of Figures 6a and 7, these factors may lie within respective ranges of 0.18 to 0.20 and 0.45 to 0.55.

[00152] The total magnitude of the adjustment of a point may be limited by a constant value. In one embodiment, the total distance of adjustment is limited to 0.0002 inch. If the

calculated total adjustment is greater than 0.0002 inch, then the actual implemented adjustment is set to 0.0002 inch in the direction of the calculated adjustment.

[00153] To better illustrate the embodiment of Fig. 6b, 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: $p1 = p1 + 0.5 * v1$; $p2 = p2 + 0.1877 * v2 + 0.1877 v2'$; $p3 = p3 + 0.5 * v3$, wherein v1 and v2 are calculated from points p0p1p2p3, and points 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: $p1 = p1 + 0.5 * v1$; $p2 = p2 + 0.1877 * v2 + 0.1877 v2'$; $p3 = p3 + 0.1877 * v3 + 0.1877 v3'$, wherein v1 and v2 are calculated from points p0p1p2p3, v2' and v3 are calculated from points p1p2p3p4, and v3' is calculated from points p2p3p4p5.

[00154] Figure 19 illustrates a method 1900 of the present invention as described above with reference to Figure 6b which represents a trajectory path of a motive body. In a first step 1902, four consecutive points of a data path, including a first point, a second point, a third point and a fourth point, are identified. For example, points p_0 , p_1 , p_2 and p_3 may be identified. In step 1904, a first arc defined by the first point, the second point and the fourth point is located. In the illustration of Figure 6b, a first arc 612 defined by points p_0 , p_1 and p_3 is identified. In step 1906, a second arc defined by the first point, the third point and the fourth point is located. In Figure 6b, a second arc 614 defined by points p_0 , p_2 and p_3 is identified. In a final step 1908, the second point is moved into an area between the first arc and the second arc. For example, point p_1 may be moved along adjustment vector v_1 into an area between arcs 612 and 614.

[00155] Step 214 (Figure 2) includes smoothing data using a double arc algorithm. Line moves may be converted into two arc moves that are tangent to each other and that maintain a given path tolerance. As illustrated in Figure 9, a line move associated with points P_0 and P_1 may be replaced with two arc moves associated with points P_0 and P_1 . More particularly, a line move 902 from point P_0 to point P_1 may be converted into Arc₁ from P_0 to arc junction point P' and Arc₂ from arc junction point P' to P_1 . Arc₁ and Arc₂ may be connected to each other and may be tangent to each other at point P' . Arc₁ may be tangent to a preceding move 904 which terminates at point P_0 . Preceding move 904 may be a line move or an arc move. Further, Arc₂ may be tangent to a succeeding or next move 906 which begins at point P_1 . Succeeding move 906 may be a line move or an arc move, and may be defined after the adjusted position of the next point P_2 (not shown in Figure 9) is determined. Smooth transitions between all the program moves are thus enabled. In one embodiment, a radius 908 of Arc₁ is different from a radius 910 of Arc₂. Radii 908 and 910 are shown only partially in Figure 9. Data smoothing essentially smoothes out abrupt turns or discontinuities in the tool path to provide a smoother tool motion.

[00156] Figure 20 illustrates a method 2000 of the present invention as described above with reference to Figure 9. In a first step 2002, a line move extending from a first point of the data path to a second point of the data path is identified. For example, line move 902 extending from point P_0 to point P_1 may be identified. In a second step 2004, the line move

is replaced with a first arc move and a second arc move, the first arc move beginning at the first point and terminating at a junction point, the second arc move beginning at the junction point and terminating at the second point. In the illustration of Figure 9, line move 902 is replaced with arc moves Arc_1 and Arc_2 , Arc_1 beginning at point P_0 and terminating at junction point P' , and Arc_2 beginning at junction point P' and terminating at second point P_1 .

Double Arc Smoothing

[00157] 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.

[00158] The double arc algorithm is developed for the third operation. This 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.

[00159] 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 $p_1, p_2, p_3, p_4, p_5, p_6, \dots, p_n$ 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+1} is tagged “smoothing”, calculate a desired tangent vector t_i (Figure 29) at point p_i based on the positions of points p_{i-1}, p_i , and p_{i+1} . The three points p_i, p_{i-1} and p_{i+1} define the $Arc_{p_{i-1}p_i p_{i+1}}$ and t_i is the tangent vector to the arc at point p_i .
- If none of p_i, p_{i-1}, p_{i+1} is tagged “smoothing”, do nothing
- Increase i by 1
- Repeat the above steps

[00160] The above function will create a sequence of desired tangent vectors at the target points of the line segments to be smoothed, denoted as $t_1, t_2, t_3, t_4, t_5, t_6, \dots, t_n$. 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.

The second function may perform the following steps:

- Check points p_i and p_{i+1} to see if the corresponding desired tangent vectors t_i and t_{i+1} have been calculated for p_i and p_{i+1} or not
- If the tangent vectors t_i and t_{i+1} are both available, 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 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}
- If any of the tangent vectors t_i and t_{i+1} is not available, do nothing
- Increase i by 1
- Repeat the above steps

[00161] In another embodiment, 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' between p_i and p_{i+2}
 - a_{i2} starts at point p' , 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

[00162] The operation of the second function in the embodiment described above is illustrated in Figures 30 and 31. In Figure 30, p_0 , p_1 and p_2 are marked for smoothing. A pair of three-dimensional arcs a_1 and a_2 are created such that arc a_1 starts at point p_0 and is tangent to tangent vector t_0 ; arc a_1 is tangent to arc a_2 at a point p' between p_0 and p_2 ; and arc a_2 starts at point p' , ends at point p_2 , and is tangent to the tangent vector t_2 . If a distance d between p_1 and the closer arc a_1 of the two arcs is within a given or predetermined tolerance, then the double arcs a_1 and a_2 are valid.

[00163] If, on the other hand, the double arcs a_1 and a_2 are invalid, i.e., the distance d is not within the given tolerance, then another pair of three-dimensional arcs a_3 and a_4 (Figure 31) are created such that arc a_3 starts at point p_0 and is tangent to tangent vector t_0 ; arc a_3 is tangent to arc a_4 at a point p'' between p_0 and p_1 ; and arc a_4 starts at point p'' , ends at point p_1 , and is tangent to the tangent vector t_1 . The above steps may then be replicated in order to create yet another pair of three-dimensional arcs (not shown) between points p_1 and p_2 . Any or all of arcs a_1 , a_2 , a_3 and a_4 may be circular arcs, and each of the arcs may have a different, respective radius.

[00164] With the double arc smoothing thus completed between points p_0 , p_1 and p_2 , the above-described second function may be repeated for the next three points along the trajectory, i.e., points p_2 , p_3 and p_4 . Further, the second function may be repeated for each subsequent set of three points along the trajectory.

[00165] The above-described double arc smoothing embodiment is directed to the case in which p_0 , p_1 and p_2 are each marked for smoothing. Assume now that, of the four points p_0 , p_1 , p_2 and p_3 , p_1 , p_2 and p_3 are marked for smoothing. Tangential vectors t_0 and t_1 may be derived from the arc defined by p_0 , p_1 , p_2 , and tangential vector t_2 may be derived from the arc defined by p_1 , p_2 , p_3 . The following five steps may be taken:

1. Double arcs, i.e., arc a_1 from p_0 to p' and arc a_2 from p' to p_2 , which are tangential at point p' may be generated from (p_0, p_2, t_0, t_2) , as shown in Figure 30.
2. The shortest distance between p_1 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 31 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.

[00166] The double arc algorithm replaces a line segment with a pair of three-dimensional arcs. Adjacent arcs created by the double arc algorithm may be tangent to each other, and all arcs may have reduced curvature changes. The double arc algorithm may be especially suitable for use in conjunction with TBC, which would yield several advantages. First, for most three-dimensional surfaces and curves, the curvature constantly changes. Mutually tangent shorter arcs provide better and smoother approximation than longer arcs. Second, for five or six axes machining, short three-dimensional XYZ arcs can coordinate with point-to-point moves in ABC axis space. This is a significant advantage since it is difficult to extend the non-TBC long arc three-dimensional smoothing to five or six axes machining. Third, all the three-dimensional arc commands will be directly executed by TBC using true arc technology, which controls motion along the true arc trajectory without polygon approximation. After smoothing, the elements in smoothing queue 206, including positioning commands, line commands, arc commands, and the program end command, may be sent to lookahead queue 208.

[00167] Figure 21 illustrates a method 2100 of the present invention as described above with reference to Figures 5b, 6b, 30 and 31. In a first step 2102, four consecutive points of

the path data, including a first point, a second point, a third point and a fourth point, are identified. In the embodiment described above with reference to Figure 6b, four consecutive points p0, p1, p2 and p3 are identified. In a second step 2104, it is determined whether the second point is suitable for smoothing. For example, it may be determined, according to the method described above with reference to Figure 5b, whether point p1 is suitable for smoothing. If not, operation may return to step 2102 wherein another set of four consecutive points is identified. For example, four consecutive points p1, p2, p3 and p4 may be identified. If the second point is suitable for smoothing, then a first arc defined by the first point, second point, and fourth point is located (step 2106). In the embodiment of Figure 6b, arc 612 is identified. In a next step 2108, a second arc defined by the first point, third point, and fourth point is located. In the embodiment of Figure 6b, arc 614 is identified. Next, in step 2110, the second point is moved into an area between the first arc and the second arc. For example, as indicated in Figure 6b, point p1 may be moved along adjustment vector v1 into an area between arcs 612 and 614. Lastly, in step 2112, a line move associated with the second point may be replaced with at least one arc move associated with the second point. For example, a line move leading into point p1, or a line move originating from point p1, may be replaced with an arc move, as illustrated in Figures 30 and 31.

ADDITIVE LOOKAHEAD

[00168] 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. 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 10 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.

[00169] The Lookahead operation may ensure enough distance to accelerate or decelerate between all velocity changes. This has advantages in that the machine bumping and dwell marks may be reduced, and the surface finish in machining may be improved.

[00170] Three functions may be planned for the Lookahead operation, namely, third order smoothing, pre-lookahead, and additive lookahead. 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. If there is not enough distance, then the information may be stored and used in the motion kernel to reduce the velocity so that the machine may accelerate/decelerate properly.

[00171] 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.

[00172] 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.

[00173] The additive lookahead function may check each newly coming motion command at the head of lookahead queue 208 and calculate the stop distance for the previous command in the following steps:

- Compare the previous command with the command preceding the previous command to get the angle or curvature between these two commands, and use this angle to determine the maximum joint-point velocity which is the allowable maximum velocity at the joint point of these two consecutive motion commands.
- Look up the tolerance/arc radius/velocity table. According to the motion trajectory and the required tolerance, determine the maximum allowable trajectory-velocity for the previous command's motion trajectory.
- Compute the stop distance for the previous command by taking the smaller of the maximum joint-point velocity and the maximum trajectory-velocity, and convert this smaller velocity to the stop distance according to the S-curve velocity profile.

- Attach the calculated stop distance to the previous command.

[00174] The lookahead is a dynamic process. The continuous trajectory changes may affect the previously processed commands. Each time a new command is added to lookahead queue 208 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 determine whether the previously calculated stop distances for these commands need to be updated or not. After a new move is processed and placed in a Lookahead queue 208, all previous moves in queue 208 may be reviewed and may have their velocities adjusted downward if required. This final preprocessing operation is analogous to placing speed limits signs along a road before a curve so that the driver can safely navigate through the curve.

[00175] The lookahead process includes a first step 216 of determining joint/path feedrate changes, and a second step 218 of looking ahead for the stop distance. In step 216, feedrate changes along the path from point P_{i-1} to point P_i are determined. In step 218, a segment length L_i may be determined between points P_{i-1} and P_i , as illustrated in Figure 11. Stop distances may also be determined for each of points P_{i-1} and P_i . The stop distance at point P_{i-1} should not be greater than the sum of L_i and the stop distance at point P_i . Else, it may not be possible to comply with velocity constraints beyond point P_i . It is this sum to which the term “additive” in “additive lookahead” refers. If it is found that the stop distance at point P_{i-1} is greater than the sum of L_i and the stop distance at point P_i , then the velocity at point P_{i-1} may be reduced such that the stop distance at point P_{i-1} is equal to the sum of L_i and the stop distance at point P_i . In equation form,

if $(\text{Stop distance}_i + \Sigma L_i) < \text{Stop distance}_{i-1}$, then

set $(\text{Stop distance}_i + \Sigma L_i) = \text{Stop distance}_{i-1}$.

[00176] In another form, let $\text{cmd}_1, \text{cmd}_2, \text{cmd}_3, \text{cmd}_4, \text{cmd}_5, \text{cmd}_6, \dots, \text{cmd}_j$ be consecutive commands in lookahead queue 208 which have been processed. For command cmd_j , $0 < j < i+1$, let its path length be len_j , and the stop distance be stopdis_j . A “valid stop distance condition” is defined as follows: If the stop distance stopdis_j for the command cmd_j is equal to or less than the sum of the path length len_{j+1} and the stop distance stopdis_{j+1} for the command cmd_{j+1} , then the command cmd_j meets the “valid stop distance condition”.

[00177] When a new command cmd_{i+1} is received and processed, it has a path length len_{i+1} , and a stop distance stopdis_{i+1} . Let $j = i$. The additive lookahead algorithm may perform the following steps:

- Check command cmd_j to determine whether it meets the “valid stop distance condition”, i.e., whether stopdis_j is equal to or less than the sum of len_{j+1} and stopdis_{j+1}
- If command cmd_j does not meet the “valid stop distance condition”, i.e., stopdis_j is greater than the sum of len_{j+1} and stopdis_{j+1} , then the stop distance for command cmd_j will be replaced by the sum of len_{j+1} and stopdis_{j+1}
- If command cmd_j meets the “valid stop distance condition”, i.e., stopdis_j is equal to or less than the sum of len_{j+1} and stopdis_{j+1} , then stop distance stopdis_j for command cmd_j may remain unchanged, and the update operation is completed
- If j is equal to 0, the update operation is completed
- Set $j = j - 1$, i.e., decrease j by 1
- repeat the above steps

[00178] When the additive lookahead function is, in a backward direction, sequentially reviewing the commands previously processed in 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 the command meets the “valid stop distance condition”, the stop distance of the command may remain unchanged. Thus, all the commands previous to the command were met and are still meeting the “valid stop distance condition”. That is, nothing changed and nothing needs to change. This is an advantageous property of the additive lookahead algorithm, which provides great efficiency for lookahead operation. Instead of looking back and updating the whole lookahead queue 208, only a few elements in the queue need to be updated, on average.

Stop distance and stopper

[00179] The additive lookahead algorithm introduces the concepts of stop distance and stopper. The stop distance of a motion command with nominal feedrate v_l (Figure 32a) and

end point speed v_0 may be defined as the distance traveled from a target point (Figure 32b) 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 32a. 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_f 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.

[00180] Once the nominal feedrate v_f 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.

[00181] 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.

Speed limitations of motion commands

[00182] 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. The maximum allowable speed 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.

[00183] 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.

[00184] 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.

Valid stop distance condition

[00185] 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):

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

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 33, 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.

[00186] The valid stop distance condition can also be explained in another way as illustrated by equation (2):

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

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} .

Additive Lookahead vs. Non-additive Lookahead

[00187] 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_I 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_I to v_0 at the target point.

[00188] Fig. 34 shows a comparison between the additive and non-additive S-curve speed profile. The speed is reduced from v_I to v_0 at the target point in both profiles. However, in the additive S-curve, the slowdown occurs between time t_{tbcI} and time t_{tbc0} , while in the non-additive S-curve, the slowdown occurs between time t_{wmI} 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_I to v_0 .

[00189] Additive lookahead may be a dynamic 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.

[00190] When a new command cmd_i is received and processed, it has a path length l_i , and a stop distance d_i . The additive lookahead algorithm may perform the following steps:

- 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 d_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 d_i , then replace the stop distance $ds_{(i-1)}$ for command cmd_{i-1} with the sum of l_i and d_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 d_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.

[00191] When the additive lookahead function is sequentially reviewing backward the commands previously processed in the lookahead queue, 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.

[00192] Figures 35a-b provide a comparison of the additive lookahead and the non-additive lookahead algorithms, respectively. In the additive lookahead (Figure 35a), 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.

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

[00193] In the non-additive lookahead algorithm (Figure 35b), both the third and fifth steps, i.e., the steps of calculating slowdown distance and the step of calculating a speed reduction for command i-1, may require a lot of computation, and may be calculated as shown in equations (4) and (5), respectively. The “distance to slowdown”, equation (4), is similar to the “calculate stop distance” equation (3).

$$(4) \quad d_{stop} = \begin{cases} v_0 T_1 + \frac{v_0 v_d}{a_{max}} + \frac{v_d T_1}{2} + \frac{v_d^2}{2a_{max}} & v_d > a_{max} T_1 \quad (\text{general case}) \\ (2v_0 + v_d) T_1 & v_d = a_{max} T_1 \quad (\text{critical point}) \\ (2v_0 + v_d) \sqrt{\frac{v_d T_1}{a_{max}}} & v_d < a_{max} T_1 \quad (\text{degenerate case}) \end{cases}$$

[00194] The “reduce speed for command i-1”, equation (5) calculates maximum allowable speed change to v_0 in the given distance, which may involve large computations such as in solving cubic equations.

$$(5) \quad v_d = \begin{cases} \frac{\sqrt{(2v_0 - a_{max} T_1)^2 + 8a_{max} d} - (2v_0 + a_{max} T_1)}{2} & d > a_{max} T_1^2 + 2v_0 T_1 \quad (\text{general case}) \\ a_{max} T_1 & d = a_{max} T_1^2 + 2v_0 T_1 \quad (\text{critical point}) \\ \text{positive real solution of} & \\ v_d^3 + 4v_0 v_d^2 + 4v_0^2 v_d - \frac{a_{max} d^3}{T_1} = 0 & d < a_{max} T_1^2 + 2v_0 T_1 \quad (\text{degenerate case}) \end{cases}$$

[00195] Furthermore, the two steps of the non-additive algorithm represented by equations (4) and (5) may need to be performed recursively until there is enough distance to slow down. Because the non-additive lookahead algorithm needs a longer distance to slow down

a same amount of speed, as shown in Figure 34, the loop has to be repeated more times than the additive lookahead algorithm until enough distance is satisfied.

[00196] In general, the additive lookahead may be more computationally efficient than the non-additive lookahead due to the additive lookahead having only one heavy load step compared with two heavy load steps in the non-additive lookahead. Another reason the additive lookahead may be more computationally efficient than the non-additive lookahead is that the additive lookahead's heavy load step is outside the recursive loop and is only performed once. In contrast, the non-additive lookahead performs the two heavy load steps recursively until enough distance is valid. The additive lookahead also needs less distance to reduce a same amount of speed, as shown in Figure 34, which means fewer previous motion commands need to be modified.

Kinematics Comparison of Additive and Non-Additive Lookahead Algorithms

[00197] Assume that two consecutive motion commands $\text{cmd}_{(i-1)}$ and cmd_i have nominal feedrates $v_{(i-1)}$ and v_i , respectively. At the joint of the two motions, the feedrate has to slow down to v_0 . Figure 36 compares the kinematical characteristics of the non-additive lookahead algorithm and the TBC additive lookahead algorithm.

[00198] Lookahead operations ensure enough distance to accelerate or decelerate between all speed changes and may be applied to all real time motion control and trajectory planning, such as CNC machine tools, robotics, autonomous vehicles and so on. Because lookahead is usually performed in real time and may modify previously processed motion commands recursively, the efficiency of the algorithm may be of importance to the reduction of the CPU load and the improvement of machine performance.

[00199] Non-additive lookahead algorithms may calculate the distance needed to accelerate/decelerate from one speed to another according to some speed profiles, such as the S-curve speed profile. When the distance of a motion segment is short and is not great enough to accommodate a large speed change, lookahead algorithms may calculate and limit the speed of that motion. The calculation of the speed limit involves a large amount of computation, and repeatedly performing the calculation is not desirable because it takes a lot of processing time and resources.

[00200] The additive lookahead algorithm may use a stop distance concept as a metric of speed limitation. Updating the speed limit in the additive lookahead is only one addition compared with the large amount of computation involving solving cubic equations in non-additive lookahead algorithms. Overall, the computational efficiency, less iteration of modifying previous motions, and increased system throughput provide the additive lookahead algorithm of the present invention with several advantages over non-additive lookahead algorithms.

[00201] Figure 22 illustrates a method 2200 of the present invention as described above with reference to Figure 11. In a first step 2202, a plurality of points along a target tool path are identified. In the example of Figure 11, points P_{i-2} , P_{i-1} and P_i along a target tool path are identified. Next, in step 2204, a maximum allowable stopping distance of the tool at a last of the points is determined. In Figure 11, segment length L_{i+1} may represent a maximum allowable stopping distance at last point P_i . Point P_{i+1} may not have advanced far enough down the data pipeline for it to be identified, and thus its characteristics may not be known. That is, the location of point P_{i+1} and whether it is a corner point or a point at the end of the target tool path may not be known. However, a minimum allowable distance L_{i+1} between points P_i and P_{i+1} is known. Because point P_{i+1} may be a corner point or a point at the end of the target tool path, it may be necessary for the tool to come to a stop, or nearly come to a stop, at point P_{i+1} . Thus, in order to ensure that it is possible for the tool to come to a stop upon reaching point P_{i+1} , the maximum allowable stopping distance of the tool at last point P_i is determined to be equal to the minimum allowable distance L_{i+1} between points P_i and P_{i+1} . In a next step 2206, a last maximum allowable speed of the tool at the last point such that the tool may come to a stop within the maximum allowable stopping distance is ascertained. In one embodiment, the S-curve of Figure 10 may be used to relate the maximum allowable stopping distance L_{i+1} to the maximum speed at last point P_i at which it may be possible for the tool to come to a stop upon reaching point P_{i+1} , i.e., the last maximum allowable speed. In a final step 2208, an other maximum allowable speed of the tool at an other one of the points such that the tool may possibly slow down to the last maximum allowable speed upon arriving at the last point is established. In the example of Figure 11, established is an other maximum allowable speed at an other point P_{i-1} such that

the tool may possibly slow down to the last maximum allowable speed at point P_i which was ascertained in step 2206. In one embodiment, the S-curve of Figure 10 may be used to relate the distance L_i to the maximum speed at other point P_{i-1} at which it may be possible for the tool to come slow down to the last maximum allowable speed upon reaching point P_i .

[00202] Taking the example above further, the maximum allowable speed at other point P_{i-1} may be compared to a maximum allowable tolerance speed at other point P_{i-1} , i.e., the maximum speed at other point P_{i-1} at which it can be ensured that the actual tool path will remain within tolerance given the curvature of the target tool path at other point P_{i-1} . The lesser of the maximum allowable speed at other point P_{i-1} and the maximum allowable tolerance speed at other point P_{i-1} may be identified, and the tool may be moved at this lesser speed at other point P_{i-1} .

[00203] Taking the above example yet further, a preceding maximum allowable speed of the tool at a point P_{i-2} preceding other point P_{i-1} such that the tool may possibly slow down to the lesser speed upon arriving at other point P_{i-1} may be established. A maximum allowable tolerance speed of the tool at preceding point P_{i-2} may be determined dependent upon the maximum allowable level of deviation and/or the curvature of the target tool path at preceding point P_{i-2} . A lesser of the maximum allowable tolerance speed of the tool at the preceding point and the preceding maximum allowable speed at the preceding point may be identified.

[00204] Figure 23 illustrates another method 2300 of the present invention as described above with reference to Figure 11. In a first step 2302, a plurality of points along a target tool path are identified. In the example of Figure 11, points P_{i-2} , P_{i-1} and P_i along a target tool path are identified. Next, in step 2304, a maximum allowable level of deviation of an actual tool path from the target tool path is established. For example, depending upon the requirements of the work piece, 0.1 cm might be established as the maximum allowable level of deviation of the actual tool path from the target tool path. In step 2306, a curvature of the target tool path is ascertained at a plurality of locations along the target tool path. For example, an angle 1907 (Figure 11) between the line move from point P_{i-2} to point P_{i-1} and the line move from point P_{i-1} to point P_i may be ascertained at point P_{i-1} . In step 2308, a maximum allowable tolerance speed of the tool at each of the points along the target tool

path is determined. The maximum allowable tolerance speed is dependent upon the maximum allowable level of deviation and the curvature. For example, given the angle 2307, a maximum allowable tolerance speed at point P_{i-1} may be determined such that the actual tool path does not deviate from the target tool path by more than 0.1 cm. Next, in step 2310, a maximum allowable stopping distance of the tool at a last of the points is determined. In Figure 11, segment length L_{i+1} may represent a maximum allowable stopping distance at last point P_i . Point P_{i+1} may not have advanced far enough down the data pipeline for it to be identified, and thus its characteristics may not be known. That is, the location of point P_{i+1} and whether it is a corner point or a point at the end of the target tool path may not be known. However, a minimum allowable distance L_{i+1} between points P_i and P_{i+1} is known. Because point P_{i+1} may be a corner point or a point at the end of the target tool path, it may be necessary for the tool to come to a stop, or nearly come to a stop, at point P_{i+1} . Thus, in order to ensure that it is possible for the tool to come to a stop upon reaching point P_{i+1} , the maximum allowable stopping distance of the tool at last point P_i is determined to be equal to the minimum allowable distance L_{i+1} between points P_i and P_{i+1} . In a next step 2312, a last maximum allowable stopping speed of the tool at the last point such that the tool may come to a stop within the maximum allowable stopping distance is ascertained. In one embodiment, the S-curve of Figure 10 may be used to relate the maximum allowable stopping distance L_{i+1} to the maximum speed at last point P_i at which it may be possible for the tool to come to a stop upon reaching point P_{i+1} , i.e., the last maximum allowable speed. In step 2314, an other maximum allowable stopping speed of the tool at each of the points other than the last point such that the tool may possibly slow down to the last maximum allowable speed upon arriving at the last point is established. In the example of Figure 11, established is an other maximum allowable stopping speed at an other point P_{i-1} such that the tool may possibly slow down to the last maximum allowable speed at point P_i which was ascertained in step 2312. In one embodiment, the S-curve of Figure 10 may be used to relate the distance L_i to the maximum speed at other point P_{i-1} at which it may be possible for the tool to come slow down to the last maximum allowable speed upon reaching point P_i . In a final step 2316, a lesser of the maximum allowable tolerance speed and the maximum allowable stopping speed is identified at each of the

points. For example, the lesser of the maximum allowable tolerance speed determined in step 2308 and the maximum allowable stopping speed established in step 2314 may be identified at each of points P_{i-2} , P_{i-1} and P_i .

[00205] Figure 24 illustrates yet another method 2400 of the present invention as described above with reference to Figure 11. In a first step 2402, a plurality of points along a target tool path are identified. In the example of Figure 11, points P_{i-2} , P_{i-1} and P_i along a target tool path are identified. Next, in step 2404, a maximum allowable stopping distance of the tool at a last of the points is determined. In Figure 11, segment length L_{i+1} may represent a maximum allowable stopping distance at last point P_i . Point P_{i+1} may not have advanced far enough down the data pipeline for it to be identified, and thus its characteristics may not be known. That is, the location of point P_{i+1} and whether it is a corner point or a point at the end of the target tool path may not be known. However, a minimum allowable distance L_{i+1} between points P_i and P_{i+1} is known. Because point P_{i+1} may be a corner point or a point at the end of the target tool path, it may be necessary for the tool to come to a stop, or nearly come to a stop, at point P_{i+1} . Thus, in order to ensure that it is possible for the tool to come to a stop upon reaching point P_{i+1} , the maximum allowable stopping distance of the tool at last point P_i is determined to be equal to the minimum allowable distance L_{i+1} between points P_i and P_{i+1} . In a next step 2406, a maximum allowable speed of the tool at an other one of the points such that a required stopping distance of the tool at the other point is equal to or less than a sum of the maximum allowable stopping distance at the last point and the segment length between the other point and the last point. In the embodiment of Figure 11, a maximum allowable speed of the tool at point P_{i-1} may be ascertained such that a required stopping distance of the tool at point P_{i-1} is equal to or less than a sum of the maximum allowable stopping distance at last point P_i and segment length L_i between other point P_{i-1} and last point P_i . In one embodiment, the S-curve of Figure 10 may be used to relate the required stopping distance of the tool at point P_{i-1} to the maximum allowable speed at point P_{i-1} .

[00206] Taking the above example further, a preceding maximum allowable speed of the tool at a point P_{i-2} preceding other point P_{i-1} such that a required stopping distance of the tool at the preceding point is equal to or less than a sum of the maximum allowable stopping

distance at last point P_i and segment length $L_i + L_{i-1}$ between preceding point P_{i-1} and last point P_i . A maximum allowable tolerance speed of the tool at preceding point P_{i-2} may be determined dependent upon the maximum allowable level of deviation and/or the curvature of the target tool path at preceding point P_{i-2} . A lesser of the maximum allowable tolerance speed of the tool at the preceding point and the preceding maximum allowable speed at the preceding point may be identified.

[00207] The Queue Manager Task may be performed by a general queue manager in a queue module that manages all the operations performed on tolerance queue 202, compression queue 204, smoothing queue 206 and lookahead queue 208 in a pipeline manner. The operations may include tolerance understanding on the elements in tolerance queue 202, compression on the elements in compression queue 204, preparatory, corrective, and double-arc smoothing on the elements in smoothing queue 206, third-order smoothing (if necessary), pre-lookahead (if necessary), and lookahead on the elements in lookahead queue 208.

[00208] The queue manager may call the related routines to manage the queue pipeline by performing the following operations: 1) stuff tolerance queue 202 with NC commands received from the NC interpreter; 2) process the NC commands in tolerance queue 202: if it is an E code, then get the tolerance information from this command and store the specified tolerance region and the related tolerance value; else, if it is a non-TBC motion command, calculate the required tolerance for this command according to the available tolerance information, and attach the required tolerance value to this command; 3) take NC commands from tolerance queue 202 and add them to compression queue 204; 4) merge the very short line moves to a longer one, i.e., replace the very short line elements by a longer line element in compression queue 204; 5) take NC commands from compression queue 204 and add them to smoothing queue 206; 6) examine all the line elements in smoothing queue 206 to identify those elements to be smoothed, and make local adjustments for those elements; 7) convert those line elements into arc elements; 8) take NC commands, mostly arc elements, from smoothing queue 206 and add them to lookahead queue 208; 9) check each newly added command and look back to the previously processed commands in lookahead queue 208; 10) determine a proper velocity for the new command, and modify

the velocities for all the previous commands if necessary; and 11) take NC commands from lookahead queue 208 and send them to the motion control kernel for execution.

[00209] All the above operations on the queues may be performed concurrently. The data process in queues may continue whenever queues are not full. The Queue Manager may call a first routine to perform all the queue operations and move the NC commands through the queue pipeline until all of the queues are full or the NC program is entirely loaded. The first routine may call a second routine to perform the tolerance, compression and smoothing operations, and may call a third routine to perform the lookahead operation.

[00210] 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 may replace the non-TBC point-to-point control with the 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.

[00211] A non-TBC CNC controller may interpolate a programmed trajectory into sequential target points with required velocities, and may load them into a PID motion control card for execution. The non-TBC CNC controller may support time-based interpolation, i.e., breaking up the line/arc data into position points and velocity points of fixed (or variable) time intervals. Then, the motion control card may perform PID loops to minimize the following error, i.e., the lag between the instantaneous position and the interpolated target point. Intuitively, imagine an elastic string with one end attached to a motion axis, the other end stepping from target point to target point. The length of the string represents the following error. The motion control card may pull the axis to reduce the length of the string. This motion control is a "pulling" mechanism, wherein line/arc data are interpolated into position points and velocity points of time intervals before execution. Individual axes may move independently under the motion control. Further, the trajectory interpolator may be an open loop, incapable of controlling the following errors. Lastly, the target points may be pre-computed, with little adjustment being made for real-time conditions.

[00212] 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 μ sec, 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.

[00213] The TBC tolerance error is different from the non-TBC following error. More particularly, the following error is the difference between the actual current position and the pre-computed target point on one axis. 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 non-TBC controller, the pre-computed target points may be interpolated based on a time interval that is usually on the order of a millisecond. In a TBC control system, the ideal current position may be calculated every control cycle. In a non-TBC control system, the motion control card, rather than the host computer, may deal with the following error. In a TBC system, the tolerance error may be directly corrected by TBC. In a non-TBC control system, the following errors of different axes may be handled independently. In a TBC system, the tolerance errors for different axes may be handled coordinately by TBC.

[00214] 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 velocity and the random corrective vector together and issue the resultant velocity to push the axes along the trajectory

[00215] The Run System Task may issue servo commands to the machine tool in order to follow the tool path and associated velocities determined in the preceding Queue Manager Task. A command designating a portion of the tool path that the machine tool is to follow may be issued once per millisecond, for example. The portion of the tool path may include a one or more line segments and/or one or more arcs. In an example shown in Figure 12, a command specifies a line segment extending from (1,4,0) to (5,1,6) in x-y-z space. A servo intended to redirect the tool path to compensate for any detected errors, i.e., deviations in the actual tool path from the corrected tool path, may be issued more frequently than a command. In the embodiment illustrated in Figure 12, a servo is issued every 200 microseconds, i.e., corresponding with each of the points A-F. Figure 13 illustrates the timing and duration of each servo over the course of a one millisecond command. That is, Figure 13 illustrates the control loop cycle time of the servos. Each servo may have a duration of less than 35 microseconds. In one embodiment, each servo has a duration of approximately between 10 and 15 microseconds. On the rising edge of the servo, the current position or some other parameter of the machine tool may be determined. Over the duration of the servo, calculations using the determined parameter may be performed. On the falling edge of the servo, an output signal to redirect the machine tool relative to the work piece may be issued. A time period of 200 microseconds is used repeatedly herein as an example of a duration of a servo. Of course, however, the cycle time of a servo may be less than or greater than 200 microseconds within the scope of the present invention.

[00216] After the first 200 microseconds of the command, the tool may be positioned at point B in Figure 12. An encoder or some other type of position sensing device may sense the actual position of the tool at point B. Upon sensing the tool position at point B, a new target point T to which the tool should be redirected by the servo along a tool subpath 1202 may be calculated or otherwise identified. The calculation of target tool subpath 1202 may be dependent upon a deviation of the actual position of the tool from the target tool path. An example of such a deviation 1204 is shown in Figure 12. The tool may continue to move

past the first actual position B during the calculation of tool subpath 1202. The Run System Task of Figure 2 includes a servo 220 having a compute position step 222, an S-Curve step 224, a Predictive Error Compensation step 226, and a Random Error Compensation step 228. The tool may have veered off to point B instead of following the linear tool path from (1, 4, 0) to (5, 1, 6) due to random errors as well as predictive errors that are systematic and repeatable. The predictive errors may be identified via empirical analysis of historic tool position data. Based upon the predictive error, it may be determined that redirecting the tool to point T is most likely to result in the tool actually moving near the desired linear tool path in a smooth motion from point B.

[00217] S-Curve step 224 includes calculating desired acceleration levels along target tool subpath 1202. More particularly, the acceleration needed to complete the servo from point B to point T, i.e., complete target tool subpath 1202, within a desired period of time and at a desired final velocity may be determined. In actuality, the tool may arrive at point C due to random errors and unidentified predictive errors. After additional servos, the tool arrives at points D, E and F. Point F may be reached at the end of the one millisecond command. More particularly, subsequent target tool subpaths may be calculated at the completion of each servo in a manner substantially similar to the calculation of target tool subpath 1202 as described above. The target tool path from (1, 4, 0) to (5, 1, 6) and the target tool subpath 1202 are shown in Figure 12 as being linear. However, it is to be understood that any target tool path or any target tool subpath may be arcuate.

[00218] The second function performed by the Tolerance Controller is the Predictive Error Compensation (step 106 of Figure 1 and step 226 of Figure 2), which may be part of a "motion kernel." The motion kernel replaces a non-TBC point-to-point control technique with predictive and true arc trajectory motion, wherein lines and arcs are the basic trajectory unit. The predictive control pushes the axes along the programmed trajectory and corrects for deviations as they are encountered.

[00219] In one embodiment, at every 200 μ sec control cycle, the Tolerance Control may read the position feedback, compare the current position with the commanded trajectory (a line or arc), and find a point on the trajectory that is the closest to the current actual position. This point on the trajectory may be defined and referred to as the current ideal position. The

difference between the current actual position and current ideal position as dynamically calculated in real time is defined as the position error, which may be dynamically calculated in real time.

[00220] Next, the predictive control may analyze the motion trajectory and the machine condition to compensate for repeatable motion errors. The correction for these errors may be predicted by using a parameterized mathematical model of the machine that has been configured by measuring actual dynamic machine responses. Predictive motion control may analyze the motion trajectory and the machine condition to make appropriate compensations to certain types of motion errors, which are repeatable and can be measured by experiments. The position error may then be combined with the predictive corrections to compute the new velocity command for each axis. The predictive correction that may be done at each control cycle is illustrated in Figure 14.

[00221] The predictive error compensation may include Centripetal Force Compensation for Arc Motion, Tangential Acceleration Compensation, "Stick" Friction Compensation, Leadscrew and Backlash Compensation, Spindle Softness Compensation, and System Delay Compensation, which are each described in turn below.

[00222] The predictive control may precisely machine an arc move without linearizing the arc into many line segments. Under Centripetal Force Compensation for Arc Motion, using the laws of physics, the predictive control may instantaneously and in real time calculate the centripetal force needed to pull or hold the axes on the arc at each ideal current position according to the current velocity and the arc trajectory curvature. The centripetal force may be converted to a predictive compensate vector, which may be instantaneously added to the velocity issued to the motion axes.

[00223] A centripetal force vector, based on the current velocity and the arc curvature, is added to a predictive compensation vector that adjusts the velocity command issued to the motion axes. This correction may enable the machine to cut highly accurate arcs. The predictive motion allows the direct precision machining of arc segments without the conversion of arc trajectories into a straight-line approximation.

[00224] Issuing a proportional velocity to multiple axes may not guarantee that a three-dimensional move is linear, i.e., in a straight line. The motion axes involved may not go

proportionally as required, because the acceleration/deceleration for different axes depend on their dynamics, which changes from axis to axis. Rather than the required straight-line move, the move may be curved due to differences in axis acceleration/deceleration caused by differing axis dynamics such as belt compliance or servo amplifier gain. Under Tangential Acceleration Compensation, the predictive control may analyze the acceleration for each individual axis and calculate the appropriate compensation forces for the axes. These compensation forces may be converted to the predictive compensation vector to generate the required axis accelerations. This correction causes the axes to be coordinated and to accurately track the programmed trajectory. These compensation forces may be converted to the predictive compensate vector to generate the required axis accelerations.

[00225] The stick friction or static friction of a machine tool has considerable influence on the motion control. An axis starting from rest requires an extra kick of energy to overcome the stick friction and begin moving. Under Stick Friction Compensation, to correct and compensate for the stick friction effect, the predictive motion uses a stick friction compensator that may trigger when an axis changes its motion direction, i.e., crosses a zero velocity point. The compensation profile is a notch shape, the height and width of the notch is a function of the velocity and the local curvature of the trajectory. The stick friction is different from machine to machine, and from axis to axis. Thus, the compensation is a function of the velocity and the local curvature of the trajectory, and is different from machine to machine and from axis to axis. This compensation may eliminate problems such as flat spots on an arc at the axis quadrants. The compensator should be built for each individual axis.

[00226] Leadscrew and Backlash Compensation may correct and compensate for the mechanical inaccuracy and backlash of the machine axis ball screws. The function of getting position feedback is responsible for the leadscrew and backlash compensation. The position values used by the motion kernel are all leadscrew and backlash compensated according to the leadscrew mapping tables. Each time the axes position feedback is read, it may be adjusted by the leadscrew and backlash compensation values stored in leadscrew map tables. This compensation may correct for mechanical axis positioning errors and for “lost motion” during an axis direction change.

[00227] For some machines, the mechanical frame of the Z-axis or the spindle may not be rigid enough, causing unwanted vibration during machining. Spindle Softness Compensation may modify the motion parameters for only the Z-axis to reduce this unwanted vibration. The parameters that are modified may include the Z-axis velocity, acceleration and jerk, wherein jerk is defined as the change in acceleration per unit time. As a result of Spindle Softness Compensation, the part surface finish may be improved.

[00228] The machine system response to a motion command is typically delayed for mechanical and electrical reasons. For example, there may be a mechanical delay between axis motor motion and actual axis motion because of the mechanical transmission. There also may be an electrical delay between the input of a velocity command to the servo amplifier and actual motion on the motor. The predictive control may compensate for this delay under System Delay Compensation by giving an anticipated adjustment before it is actually needed. This may provide the necessary correction to the motion axes at the time it is needed on the programmed trajectory.

[00229] Via the above-described compensation schemes, the Tolerance Control may compensate for predictive, recurring errors. The predictive control may push the axes along the programmed trajectory and the compensation from the machine model may compensate for and correct repeatable machine errors. Both of these adjustments may greatly reduce the position error between the current position and the line/arc trajectory.

[00230] In addition to compensating for predictive error, the Tolerance Control may correct random error, as illustrated in Figure 15. Any remaining position error that is not corrected by the predictive control as illustrated in Figure 14 may be further reduced by the random error correction mechanisms. By utilizing the predictive motion, which directly pushes the axes along the programmed trajectory and the pre-measured/computed compensation, which corrects the repeatable machining errors, the TBC greatly reduces the instantaneous position error (the deviation of the actual current position from the line/arc trajectory). Random error correction is analogous to driving on a road and making small adjustments to the steering wheel in order to compensate for variations in the surface of the road and to thereby stay exactly in the center of the road.

[00231] The position deviation resulting from the random error occurred in real-time is further reduced by using random error correction mechanisms, which may include Proportional Control, Integral Control, Derivative Control, S-Curve Acceleration and Velocity Control, and Dynamic Velocity Adjustments. Each of these random error correction mechanisms is described in turn below.

[00232] Under Proportional Control, the position error may be multiplied by a proportional gain parameter to arrive at the random corrective velocity vector. The resulting random corrective velocity vector may be subtracted from the Predictive Correction velocity, i.e., from the commanded velocity, to thereby reduce the error.

[00233] Under Integral Control, the position error may be integrated using a modified Laplace transform algorithm. A correction value, which is proportional to the above-integrated result, may be computed and added to the Predictive Correction velocity (commanded velocity). The modified Laplace transform algorithm may eliminate position error that persists over a period of time. A direct integral control may cause undesirable oscillation. A lower frequency modified Laplace transformation may be used to integrate the position errors with much less oscillation.

[00234] Under Derivative Control, a derivative value of the position error may be added to the Predictive Correction velocity to dampen the Random Error correction. This dampening may improve the stability of the Random Error correction. An artificial viscous friction may be added to increase the damping coefficient of the system so that the system is steadier. This is equivalent to a proportional plus derivative control.

[00235] Under S-Curve Acceleration and Velocity Control, changes in velocity that are defined in the part program or are adjusted by the Additive Lookahead algorithm may be controlled using S-curve acceleration. The S-curve acceleration may be computed at each 200 μ sec control cycle or servo cycle. Thus, very smooth control of the axes motion and, ultimately, the axes position may be provided. S-Curve acceleration provides a non-linear acceleration method that softens and smoothes the transition between the at-velocity phase and the acceleration/deceleration phase.

[00236] The S-Curve velocity profile typically has seven phases, including increasing acceleration; constant acceleration; decreasing acceleration; at speed (constant velocity); increasing deceleration; constant deceleration; and decreasing deceleration.

[00237] Non-TBC CNC control uses a time-based scheduling mechanism. Thus, the S-Curve may be implemented as a function of time. The function output is the scheduled position and velocity. The TBC uses the true arc technology. At each control cycle, the ideal position on the trajectory may be calculated instead of the scheduled position. Thus, only the velocity along the trajectory needs to be calculated from the S-Curve. The S-Curve in the TBC system may be implemented as a real-time process.

[00238] The velocity may be calculated from the instantaneous path distance at every control cycle. This mechanism may eliminate the velocity error accumulation and may support some real-time requests such as feed override, pause, resume and single step.

[00239] The S-Curve in TBC may have the typical seven phases of increasing acceleration; constant acceleration; decreasing acceleration; at speed (constant velocity); increasing deceleration; constant deceleration; and decreasing deceleration. The increasing acceleration phase may include a constant initial jerk J_0 . Acceleration may be linearly increased to the maximum acceleration with the initial jerk J_0 . Velocity may increase as a quadratic function of time.

[00240] In the constant acceleration phase, there may be no jerk, i.e., jerk may equal 0. Acceleration may remain at the maximum acceleration. Velocity may increase as a linear function of time.

[00241] In the decreasing acceleration phase, jerk may equal $-J$, wherein J is the maximum jerk. Acceleration may linearly decrease from the maximum acceleration to 0 with the negative maximum jerk $-J$. Velocity may increase as a quadratic function of time.

[00242] In the at speed phase, jerk and acceleration both equal 0. Velocity is equal to the final constant velocity.

[00243] In the increasing deceleration phase, jerk may equal $-J$, wherein J is the maximum jerk. Deceleration may linearly increase from 0 to the maximum deceleration with the maximum jerk J . Velocity may decrease from the final velocity as a quadratic function of time.

[00244] In the constant deceleration phase, there may be no jerk, i.e., jerk may equal 0. Deceleration may remain at the maximum deceleration. Velocity may decrease as a linear function of time.

[00245] In the decreasing deceleration phase, jerk may equal the maximum jerk J. Deceleration may decrease linearly from the maximum deceleration to 0 with jerk being equal to the maximum jerk J. Velocity may decrease as a quadratic function of time.

[00246] During the lookahead processing, the additive lookahead algorithm discussed above may attach a stop distance to each motion command. The stop distance may be used in the S-Curve implementation to guarantee that there is enough distance in which to accelerate/decelerate properly between velocities.

[00247] In one embodiment, S-Curve computation involves the use of several variables, such as distance_to_go to represent the trajectory length from the current position to the target position of a command. Variable stop_distance_left represents the sum of distance_to_go and the value of the stop_distance attached to the command. The values of distance_to_go and stop_distance_left may vary with the instantaneous position of the machine. Variable stop_distance_required represents the moving distance required to completely stop the machine according to the velocity, deceleration and the S-Curve for the command. The value of stop_distance_required may vary with the velocity.

[00248] When a motion command is being executed, at every control cycle, the variables stop_distance_left and stop_distance_required for this command may be calculated and compared. Whenever the value of stop_distance_left is less than or equal to the value of stop_distance_required (not enough distance is left), a velocity/distance conversion algorithm may be used to calculate the velocity from the stop_distance_left, as described in more detail below. When the value of stop_distance_left is greater than the value of stop_distance_required (enough distance is left), the velocity in each of the seven phases may be calculated as described below.

[00249] In Phase 1, acceleration is increased linearly according to the initial jerk. Acceleration may be calculated according to the equation:

$$\text{acceleration}_{i+1} = \text{acceleration}_i + J_0 \times \text{cycle_time}$$

wherein J_0 is the initial jerk and cycle_time is the time interval of each control cycle, which

may be approximately 200 μ sec. Velocity is increased according to the acceleration value. Velocity may be calculated according to the equation:

$$\text{velocity}_{i+1} = \text{velocity}_i + \text{acceleration} \times \text{cycle_time}.$$

When acceleration reaches the maximum_acceleration, it may be truncated as follows:

If (acceleration $\geq$ maximum_acceleration), then set:

$$\text{acceleration} = \text{maximum_acceleration};$$

and enter Phase 2.

[00250] In Phase 2, acceleration equals maximum_acceleration. Velocity may be calculated according to the equation:

$$\text{velocity}_{i+1} = \text{velocity}_i + \text{maximum_acceleration} \times \text{cycle_time}.$$

The difference between the final_velocity and current velocity is expressed by the equation:

$$\text{final_velocity} - \text{velocity} = 0.5 \times J \times t^2,$$

wherein J is the maximum jerk, and t is the time required to reach the final velocity. Since the acceleration equals the product of jerk and time, i.e., $J \times t$, the equation above may be rearranged as:

$$2 \times J \times (\text{final_velocity} - \text{velocity}) = \text{acceleration}^2$$

or

$$\text{acceleration} = \sqrt{2J(\text{final_velocity} - \text{velocity})}.$$

If acceleration is less than the maximum_acceleration, then Phase 3 (decreasing acceleration) may be entered.

[00251] In Phase 3, velocity may be calculated by the equations:

$$\text{acceleration} = \sqrt{2J(\text{final_velocity} - \text{velocity})}$$

$$\text{velocity}_{i+1} = \text{velocity}_i + \text{acceleration} \times \text{cycle_time}.$$

If velocity reaches final_velocity, then Phase 4 (steady state speed) may be entered.

[00252] In Phase 4, velocity equals the maximum_velocity, and acceleration equals 0.

[00253] In each of Phase 5 (increasing deceleration), Phase 6 (constant deceleration), and Phase 7 (decreasing deceleration), the velocity may be calculated using a velocity/distance conversion algorithm.

[00254] At the control cycle when the value of stop_distance_left is less than or equal to the value of stop_distance_required (the condition automatically satisfies in Phases 5, 6 and 7), the velocity/distance conversion algorithm may be used to calculate the velocity from the stop_distance_left. The key features of one embodiment of this algorithm are described below.

[00255] Since S-Curve is a function that defines the relation between velocity and time, it can be represented by the velocity equation:

$$\text{velocity} = S_Curve(t).$$

Let t_1 be the current time, and let t_2 be the time when the motion completely stops. The time required for the motion to completely stop is then given by $t_2 - t_1$. The algorithm may cause the value of stop_distance_required to be equal to the value of stop_distance_left by selecting a proper velocity.

The moving distance of the machine during $t_2 - t_1$ may be defined as the stop_distance_required at time t_1 . The stop_distance_required at time t_1 may be calculated by taking the integral of the velocity equation with respect to the time t as per the equation:

$$\text{stop_distance_required} = \int_{t_1}^{t_2} S_Curve(t) dt.$$

The above equation defines the relation between stop_distance and time. For simplicity, the equation may be rewritten as:

$$\text{stop_distance_required} = \text{Stop_Dis}(t).$$

Therefore,

$$t = \text{Stop_Dis}^{-1}(\text{stop_distance_required})$$

wherein Stop_Dis^{-1} is the inverse function of Stop_Dis.

Substituting for t in the velocity equation yields:

$$\text{velocity} = S_curve(\text{Stop_Dis}^{-1}(\text{stop_distance_required})).$$

In order to make the value of stop_distance_required equal to the value of stop_distance_left, the velocity may be set according to the equation:

$$\text{velocity} = S_curve(\text{Stop_Dis}^{-1}(\text{stop_distance_left})).$$

[00256] In a machine tool, different axes may have different maximum velocities. The maximum velocity of the Z axis is usually lower than the maximum velocity of the X axis or of the Y axis. Assume maxv_x, maxv_y and maxv_z are the maximum velocities for axes

X, Y and Z, respectively, and dmv is a vector in three-dimensional space. If the machine tool moves along the direction of dmv , then the maximum allowable vector velocity along the direction of dmv is a function of the vector dmv :

$$\text{maximum_allowable_velocity} = f(dmv).$$

The function is a mapping from a three-dimensional unit vector to a real number. A conservative way to determine function f is to define it as a constant:

$$f(dmv) = \text{Min}\{ \text{maxv_x}, \text{maxv_y}, \text{maxv_z} \}$$

[00257] If the machine tool starts from the origin, along the direction of an arbitrary direction vector dmv , and with a given vector velocity $f(dmv)$, and moves for one unit of time, then the machine tool will arrive at a certain end point. The end points resulting from all possible moving directions will form a three-dimensional surface, which is referred to as the velocity envelope. The shape of the velocity envelope depends on the function f . In this case, the distance from an end point to the origin is a constant $\text{Min}\{ \text{maxv_x}, \text{maxv_y}, \text{maxv_z} \}$, and thus the velocity envelope is a sphere.

[00258] An aggressive strategy is to move the machine tool along direction dmv with the highest possible vector velocity, provided the velocities of axes X, Y, and Z are bounded by maxv_x , maxv_y and maxv_z , respectively. In this case, the velocity envelope is a three-dimensional cubic box, with length $2 \times \text{maxv_x}$, width $2 \times \text{maxv_y}$, and height $2 \times \text{maxv_z}$.

[00259] The velocity envelope can be used to define the velocity coordinate type of axes.

There are four velocity coordination types, in order of decreasing aggressiveness:

AXIS_VELOCITY_CUBIC_TYPE, AXIS_VELOCITY_CYLINDER_TYPE, AXIS_VELOCITY_ELLIPSE_TYPE, and AXIS_VELOCITY_SPHERE_TYPE. The TBC may use AXIS_VELOCITY_ELLIPSE_TYPE. The velocity envelope of

AXIS_VELOCITY_ELLIPSE_TYPE is a three-dimensional ellipse, with X axis length equal to $2 \times \text{maxv_x}$, Y axis length equal to $2 \times \text{maxv_y}$ and Z axis length equal to $2 \times \text{maxv_z}$.

[00260] The TBC supports the dynamic velocity adjustment. Under Dynamic Velocity Adjustments, the Tolerance Control may monitor the position error and compare the error with the specified tolerance. A safety level, defined as a percentage of the required tolerance, may be used to control the position error. If the position error is beyond the safety

level, the axes velocity may automatically be decreased in order to reduce the position error. If the position error is below the safety level, the axes velocity may automatically be increased to the original velocity. In this way, the machine may be allowed to run as fast as possible, or nearly so, while still maintaining the desired tolerance. The dynamic velocity adjustment may be based on a Laplace transformation.

[00261] 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. A comparison between non-TBC motion control and TBC motion control is provided below.

Non-TBC Motion Control Structure

[00262] In order to track a three-dimensional contour trajectory, non-TBC motion 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 37, 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 though two of the three axes, the actual contour that is achieved may still deviate from the reference contour significantly.

[00263] The non-TBC motion control system illustrated in Figure 37 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. The contour tracking may be open loop in non-TBC motion control. Motion along each of the three axes may be independent without knowledge of, or dependence upon, motion along any other axis. In other words, there may be no teamwork among the axes. There may be no arc motion even with circular interpolation in non-TBC motion control. Different dynamic responses in the axes may negatively affect tracking accuracy and surface quality.

[00264] In non-TBC motion control the contour tracking may be open loop, and there may be no guarantee that the actual contour achieved is close enough to the desired reference contour. An example of uncoordinated motion is given in Figures 38a-d, where a reference trajectory $y=x$ (Figure 38a) is decomposed into time based axis trajectories: $y=kt$ (Figure 38b) and $x=kt$ (Figure 38c). The non-TBC motion controller may control motion in the x and y axes to track their trajectories independently. Two different cases are represented in Figures 38b-d. Case 1 is characterized by small tracking errors in both x and y axes; however the contouring error as shown in Figure 38d is relatively large. Case 2 is characterized by larger tracking errors in both x and y axes, but there is no contouring error at all shown in Figure 38d. Because the non-TBC motion control reduces the tracking error in each axis individually without any coordination among axes, the non-TBC motion system may consider case 1 to be more desirable than case 2.

[00265] When a contour is decomposed into $x/y/z$ axes and becomes three time-based axis trajectories, important kinematics information, e.g., centripetal acceleration information, may be lost. Without the centripetal acceleration information, a factor that distinguishes arc motion from line motions, true arc motion may not be possible. Moreover, because the effects of centripetal force may not be compensated for, the actual tool position may be biased in a radially-outward direction due to the centripetal force.

[00266] In non-TBC motion control, the dynamic response of each axis may affect the contouring performance. If the dynamics are not matched with each other, e.g., some axis responds faster than others or some axis responds slower than others, the actual contour may distort or deviate from the desired contour.

TBC Motion Control Structure

[00267] 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 Fig. 39, where all signals are vectors and represented by thick lines. In contrast, in the non-TBC motion control structure shown in Fig. 37, signals are scalars and are represented by thin lines.

[00268] The TBC motion control structure of Figure 39 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 arcuate trajectory rather than a trajectory formed by a series of linear segments that approximate a curved trajectory. TBC motion control may not be time deterministic, and may be capable of on-the-fly feedrate adjustments.

[00269] At every control cycle, such as every 200 microseconds, 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 40 and described below.

[00270] 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.

[00271] 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.

[00272] 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.

[00273] 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. However, 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.

[00274] The motion kernel may receive motion commands from lookahead queue 208. The motion commands may go through tolerance understanding processing, data compressing, data smoothing, and lookahead processing before being executed by the motion kernel. Each command may carry the required tolerance value for TBC control and the stop distance for S-Curve computation. The motion kernel may be executed every 200 μ sec in RTSS environment wherein a deterministic response is ensured.

[00275] Figure 16 illustrates one embodiment of a method 1600 of the present invention for executing a motion kernel. In a first step 1602, an element is taken from lookahead queue 208. The element may be the motion command to be executed next. Next, in step 1604, a routine may be used to initialize and calculate some variables and parameters which are needed for executing the command just taken in step 1602. In step 1606, a routine that is the main part of the motion kernel is executed. The routine of step 1606 may perform the

predictive motion, true arc move, tolerance control and S-Curve computation. In step 1608, a routine may determine whether the execution of the current motion command has been completed. If so, operation is returned to step 1602 to take another motion command from lookahead queue 208. If not, operation is returned to step 1606 to continue performance of the predictive motion, true arc move, tolerance control and S-Curve computation.

[00276] Step 1606 may include twenty substeps as described below, and referred to in Figures 39 and 40, in order to convert a motion command into a servo amplifier digital-to-analog converter (DAC) command, or a servo amplifier command for use with pulse width modulation and digital controls and drives. These twenty substeps may be performed within servo 220. In Step 1, the current tool position is obtained, and the position may be leadscrew and backlash compensated.

[00277] Position geometry analysis is performed in Step 2. More particularly, four computations may be made: the ideal current position on the arc or line trajectory; the current tangential moving direction; the positional deviation from the ideal position; and the tolerance error.

[00278] Step 3 includes performing lag distance delay compensation. The lag distance required to compensate time delay in the system may be computed, and this lag distance may be added to the stop distance which is used to calculate the velocity by the S-Curve.

[00279] S-Curve velocity control is performed in Step 4. The tool velocity may be calculated as a function of stop distance according to the S Curve. The output is relative speed, having a value between 0 and 1.

[00280] Modified Laplace transformation and tangential predictive control is performed in Step 5. The current velocity may be estimated by the modified Laplace transformation. Tangential predictive control may be added.

[00281] Step 6 includes velocity control for axes. Velocity components may be computed for each axis. The computation may be based on the velocity calculated at step 4 and the tangential moving direction computed at step 2.

[00282] Acceleration is estimated in Step 7. The centripetal force as well as the compensation vector for the centripetal compensation may be estimated. The order, i.e., sequence, of Steps 5 through 7 is arbitrary and may be changed in the program code.

[00283] Step 8 includes centripetal predictive compensation in which the centripetal compensation vector may be added to velocity. Step 9 includes statistical analysis, and may be used for research.

[00284] In Step 10, the current moving direction is computed. The current moving direction may be needed for looking up the leadscrew mapping tables to implement the leadscrew/backlash position compensation. Step 11 includes viscous friction estimation in which the error velocity is estimated.

[00285] Step 12 includes random error correction and viscous friction correction. The random error correction vector may be calculated, and the viscous friction may be added thereto.

[00286] A run time velocity adjustment is performed in Step 13. It is determined whether the position error is within the safety tolerance level. If so, the velocity slow down function may be disabled. If not, the velocity slow down function may be enabled, which may decrease the velocity accordingly.

[00287] Step 14 includes a variable gain calculation in which the constant gain, the first order gain, the second order gain, and the combined gain may be calculated. In Step 15, machine vibrations may be analyzed, compensated for, and/or eliminated.

[00288] Random error correction may be performed in Step 16. The random error correction vector may be multiplied by the combined gain. The random error correction vector may be truncated and/or applied to velocity.

[00289] In Step 17, the velocity error may be reduced by using a modified Laplace transformation. Step 18 includes motor saturation protection. The DAC voltage change may be detected at every cycle. The DAC voltage value may be truncated if it is too high.

[00290] Velocity coordination may be performed in Step 19. Machine movement for axes may be coordinated with non-uniform maximum velocities.

[00291] In Step 20, a velocity command, ranging from -10 to +10 volts, may be transmitted in order to control velocity.

[00292] The table below illustrates which of the twenty steps are performed in converting three types of motion commands into a servo amplifier DAC command. A "Rapid Line" motion command initiates a straight line move that may be executed at the maximum

feedrate for each axis in the move. A Rapid Line command always stops at the end of a move, while a Line command may not. A Rapid Line command may be used to move the tool to a given point as fast as possible.

Step	Step Name	Rapid Line	Line	Arc
1	Get Current Position	Yes	Yes	Yes
2	Position Geometry Analysis	Distance to go based on end position. Set err=0.	Distance to go based on stopper position. Calc err.	Distance to go based on stopper position. Calc err.
3	Lag Distance Delay Compensation	No	Yes	Yes
4	S-Curve Controller	Uncoordinated Exponential Deceleration	Coordinated Modified S-Curve Deceleration	Coordinated Modified S-Curve Deceleration
5	Modified Laplace velocity transformation	No	Yes 5% cap	Yes 5% cap
6	Component Velocity Control	No	Yes	Yes
7	Acceleration Estimation	No	Yes	Yes
8	Centripetal Predictive Compensation	No	Yes	Yes
8.5	Integral Compensation	No	Yes	Yes
9	Error Statistical Analysis	No	Yes	Yes
10	Backlash Compensation	No	Yes	Yes
11	Viscous Friction Estimation	No	Yes	Yes
12	Random Error Correction (Method #1)	Yes	No	No
13	Run Time Velocity Adjustment	No	Yes	Yes
14	Variable Gain Calculator	No	Yes	Yes
15	Vibration Elimination	No	Yes	Yes
16	Random Error Correction (Method #2)	No	Yes	Yes
17	Speed Regulator	No	No	No

18	Motor Saturation Protection	Yes	Yes	Yes
19	Velocity Coordination	Yes	Yes	Yes
20	Set Dac Value	Yes	Yes	Yes

[00293] The TBC motion kernel may be deployed on a wide variety of CNC machine platform types. Like other CNC controllers, parameters may be adjusted to optimize the control's performance for a particular machining platform or environment. Adjustment of these parameters is referred to as "tuning".

[00294] The present invention encompasses five categories of tuning parameters, including S-Curve control parameters, smoothing/compression control parameters, motor speed adjustment parameters, dynamic control gains, and a speed/radius tolerance relation lookup table. The S-Curve control parameters are adjustable within a reasonable range in order to achieve maximum allowable speed, maximum allowable acceleration, maximum allowable deceleration, maximum allowable start jerk, and/or maximum allowable end jerk. Smoothing/Compression control parameters are also adjustable within a reasonable range in order to achieve first order smoothing length limit (segments longer than value are not smoothed), first order smoothing angle limit (angles larger than value are not smoothed), second order smoothing length limit, and/or data compression Off/On control. Motor speed adjustment parameters may be manually adjustable to achieve M - multiplier (i.e., at X percent of maximum voltage, X percent of maximum speed is observed), and/or B - constant drifting adjuster for each axis (electronic zero compensation). Dynamic control gains may be preset for a machine class, and may be manually modifiable as needed. Dynamic control gains may include time/response gain parameters. Excessively high values of the time/response gain parameters may cause oscillations, and low values may cause slow/soft motion response. Lastly, the speed/radius tolerance relation lookup table may be preset for each class of machines, and may be manually adjusted as necessary. The above-described tuning parameters may be listed together with other system configuration parameters in a configuration file which may be read and interpreted during the system configuration time.

TBC Geometry Analysis

Motion Command Pre-Analysis

[00295] 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 41, which illustrates the line motion geometry pre-analysis:

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

[00296] For an arc motion, the SPN may be calculated as indicated by equation (7) and shown in Figure 42, 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.

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

[00297] The stopper position P_{si} may be calculated as indicated by equation (8) for both line and arc motions.

$$(8) \quad P_{si} = P_i - ds_i \cdot SPN_i$$

Real-time Geometry Analysis

[00298] At each control cycle, the TBC motion kernel may read the actual contour (tool) position feedback P_{tl} 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_go d (to the stopper).

[00299] For a line motion, the geometry is shown in Figure 43, which illustrates line motion geometry analysis, and may be calculated as indicated by equations (9) and (10).

$$(9) \quad d = \langle P_{tl} - P_{si}, SPN \rangle$$

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

wherein $\langle a, b \rangle$ is the inner production of vector a and vector b . Thus, if $a = [x_a, y_a, z_a]$, and $b = [x_b, y_b, z_b]$, then $\langle a, b \rangle = x_a \cdot x_b + y_a \cdot y_b + z_a \cdot z_b$. 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.

[00300] 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 44a-b, which are illustrative of arc motion geometry analysis.

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

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 $\langle Ptl - Ci, APN \rangle$.

[00301] 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).

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

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

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

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

[00302] 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 44a-b.

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

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

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

TBC S-Curve Controller

[00303] 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.

[00304] The S Curve velocity profile usually has seven phases: increasing acceleration; constant acceleration; decreasing acceleration; at speed; increasing deceleration; constant deceleration; and decreasing deceleration.

[00305] 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). The TBC S-Curve controller may be different from the Non-TBC S-Curve controller. 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.

[00306] A flow chart of one embodiment of a method of operation 4500 of a TBC S-Curve controller is shown in Figure 45. 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 great 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. In step 4518, an advanced nonlinear jerk-acceleration-velocity (JAV) filter is applied to remove the noise in the velocity signal.

TBC Feed Forward Control

Feed Forward Signals and Gains

[00307] Feed forward may be based on the ideal current position (ICP). The calculation of the three feed forward parts, i.e., normal acceleration FF_{an} , tangential velocity FF_{vt} , and tangential acceleration FF_{at} , is shown in Figure 46, which illustrates velocity and acceleration feedforward. In Figure 46, "d" represents the distance_to_go obtained from the geometry analysis; and dt and dr represent unit tangential and radial/normal direction vectors, respectively.

[00308] In a lag distance compensation block 4602, the lag distance required to compensate time delay in the system may be computed and added to the distance_to_go to arrive at a compensated distance. S-Curve controller 4604 may receive the compensated distance and calculate the desired velocity v and acceleration a (both v and a are scalars) along the trajectory.

[00309] 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, K_{aff} , may convert the tangential acceleration a into a control signal $FFat$, which is acceptable by the servo system.

[00310] 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.

[00311] The centripetal acceleration a_n 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 K_{aff} as applied to the tangential acceleration may be applied to convert the normal acceleration into a suitable control signal $FFan$.

Runtime Feedrate Adjuster (μ)

[00312] The runtime or "on-the-fly" feedrate adjuster is a unique feature of TBC motion control, and is not available in other control systems. The non-TBC, time-based control strategy's reference contour trajectory is predetermined outside the control loop, and no modification can be done inside the control loop. In contrast, 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.

[00313] 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 desired line/arc trajectory. 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.

[00314] 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.

TBC Feedback Control

Normal Error Feedback Control with Variable Gain (τ)

[00315] 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.

[00316] A fundamental 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 the 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.

[00317] 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. Non-TBC motion control may not include compensating for inter-axes effects, such as centripetal force, as TBC motion control may. Because of the lack of compensation for centripetal force, a tool controlled by a non-TBC method may systematically travel radially outward of a desired arcuate trajectory.

[00318] As shown in Fig. 47, 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 40). A special feedback controller Cn may provide system stability and reduce the normal error.

[00319] 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.

Tangential Error Feedback Control

[00320] 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 40) may reduce the tangential error.

Different Dynamic Responses of the Two Feedback Controls

[00321] 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.

TBC Stick Friction and Backlash Compensation

[00322] 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.

[00323] 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.

[00324] 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.

TBC Control Signal Protection

[00325] 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.

[00326] 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.

[00327] Figure 17 is a block diagram of one embodiment of a machine 1700 of the present invention for machining a work piece 1702. Machine 1700 includes a controller in the form of a controller card 1704, an actuator in the form of servo drives 1706x, 1706y, 1706z, 1706s, a cutting tool 1708, and position sensors 1710x, 1710y, 1710z and 1710s, which may be in the form of encoders.

[00328] The machine tool retains and carries cutting tool 1708 while the cutting tool rotates and engages the work piece, as indicated by the dashed line between the work piece and the cutting tool in Figure 17. The machine tool provides a mounting approach and

linear/rotational drive approach for the cutting tool. Cutting tool 1708 is the material remover in machine 1700, and may be mounted in the rotating spindle mechanism (not shown) which may be attached to the z-axis of the machine tool. The geometric orientation of the cutting tool with respect to the work piece may be fixed due to its mounting position on the z-axis.

[00329] As is typical of most motion control cards, controller card 1704 includes connectors 1712, 1714, 1716 and 1717 for physically connecting controller card 1704 to servo drives 1706 and position sensors 1710 in order to facilitate communications. Servo drives 1706 may move work piece 1702 in the x, y and z axes and rotate cutting tool 1708 about the z-axis. Four connectors 1712, 1714, 1716, 1717 are depicted on controller card 1704, one for each controlled axis of motion and one for the rotation of cutting tool 1708, but other configurations are also possible. The communication signals exchanged between controller card 1704 and servo drives 1706 must often be converted into the appropriate format. For example, some actuator motors use an analog voltage input signal between -10 volts and +10 volts to determine at what rate the motors should turn.

[00330] Most digital computers, like the Intel Pentium Based PC, do not support analog voltages, but utilize groups of bits (ones and zeros) represented by discrete voltage levels (+5 volts and 0 volts) to represent values. Thus, if a PC based motion control card were to command a motor to spin in the clockwise direction at its maximum speed (+10 volts in our example) the collection of bits representing this request would have to be converted to the +10 volt signal understood by the motor. This conversion of signals for communication purposes is standard in the industry and often performed by the input/output, I/O, module of the controller card. In Figure 17, one I/O module (1718, 1720 and 1722) is depicted for each controlled axis.

[00331] Some control cards incorporate a CPU 1724 and on-board memory 1726 for the execution of programs and control algorithms. The PMAC offered by Delta Tau and the DMC-1700 by Galil Motion Control, Inc. of Mountain View, Calif. is typical of cards incorporating a CPU and memory. The servo transducer 5-axis PCB Assembly (part no. 415-0622-001), offered by Hurco Companies, Inc. of Indianapolis, Ind., may be used as the control card in the exemplary embodiment. This Hurco servo transducer does not have an

onboard CPU and memory but utilizes the CPU and memory of the computer into which they are plugged for these services. The difference between these two configurations is primarily one of cost and is of little consequence to a programmer implementing the tolerance based control technology of this invention. The integration of these components and the method by which control algorithms are stored and executed by the CPU is well understood in the art. The details of this process have been omitted for the sake of brevity. To simplify the explanation of the invention, Figure 17 shows CPU 1724 and memory 1726 physically integrated on motion control card 1704.

[00332] In the exemplary embodiment, a multi-tasking operating system (OS), like Windows XP offered by the Microsoft Corporation of Redmond Washington, may be employed for running the algorithms implementing this invention. A real-time OS, like RTX offered by Ready Systems of Sunnyvale, Calif. could also be used improving the run time performance of the algorithms as well. The decision of what operating system to use and whether or not to employ multi-tasking paradigm depends on the application at hand and is a decision routinely made by those skilled in the art of computer system programming and design. Contained in memory 1726 is a data area for storage of values relevant to the task at hand (1728, motion command and permitted feedrates; 1730, NC tolerance commands; 1732, feedrate limitation mappings; 1734, other data), as is commonly employed by people skilled in the art of programming.

[00333] Position sensors 1710x, 1710y, 1710z sense the positions of work piece 1702 in the x, y and z directions, respectively, as indicated by the dashed lines between the position sensors and the work piece in Figure 17. More particularly, in one embodiment, position sensors 1710x, 1710y, 1710z sense the positions of three respective ballscrew mechanisms (not shown) that drive movable positioning slides (not shown) of the machine tool. The work piece is attached to one of the positioning slides, which, by the nature of the cascaded mounting approach of all of the positioning slides, permits the work piece to be moved in two directions, i.e., the x and y directions. Thus, the position of the work piece in the x-y plane may be deduced from knowledge of where the work piece is attached to the x and y positioning slides and knowledge of the positions of the ballscrew mechanisms. There is generally a fixed geometric relationship between the work piece, the ballscrew mechanisms

and the positioning slides. Given that the rotating spindle mechanism is attached to the z-axis positioning slide, position sensor 1710z may sense the position of the z-axis (third direction) ballscrew mechanism, which enables the linear position of the cutting tool along the z-axis to be deduced. Position sensor 1710s senses the rotational position of cutting tool 1708, as indicated by the dashed line between position sensor 1710s and the cutting tool in Figure 17. Position sensors 1710x, 1710y, 1710z, 1710s may transmit associated position signals to CPU 1724 via connectors 1712, 1714, 1716, 1717. Thus, position sensors 1710 may provide the Position Feedback depicted in Figure 1.

[00334] Figure 25 illustrates a method 2500 of the present invention as described above with reference to Figure 12. In a first step 2502, a target tool path is identified. In the example shown in Figure 12, a target tool path from (1,4,0) to (5,1,6) is identified. In step 2504, the tool is commanded to follow the target tool path from an initial position. For example, the tool may be commanded to follow the target tool path from initial position (1,4,0) to (5,1,6). Next, in step 2506, an actual position of the tool is sensed after the commanding step. In the example of Figure 12, an actual position of the tool at point B is sensed. In a final step 2508, a target tool subpath from the actual position is calculated to correct for predictive errors in movement of the tool. For example, a target tool subpath 1202 from the actual position at point B is calculated to correct for predictive errors in movement of the tool. The predictive errors that are corrected may occur anywhere along the path between points A and F, for example. Further, the predictive errors may be empirically determined long before the target tool path is identified.

[00335] Figure 26 illustrates another method 2600 of the present invention as described above with reference to Figure 12. In a first step 2602, predictive errors in movement of the tool are empirically determined. In the example shown in Figure 12, predictive errors in the movement of the tool may have been empirically determined through testing before the machine tool has arrived at (1,4,0) or before the machine tool has even gotten within proximity of the work piece. The empirical determination may occur in a laboratory, in a manufacturing plant, or in a combination of both types of environments, for example. The determination of the predictive errors may be dependent upon ambient temperature and/or humidity, for example. In step 2604, actual movements of the tool are monitored. For

example, position sensors 1710 may sense the positions of machine tool 1708 and transmit associated position signals to CPU 1724 as described above. In a final step 2606, the actual movement of the tool are controlled dependent upon the empirically determined predictive errors and random errors in the actual movements of the tool. For example, the tool may be directed along target tool subpaths, such as subpath 1202, by servo amplifier 1706.

[00336] Figure 27 illustrates yet another method 2700 of the present invention as described above with reference to Figure 12. In a first step 2702, an actual position of the tool after a first of a plurality of sequential movements is sensed. For example, position sensors 1710 may sense the position of machine tool 1708 after a movement of tool 1708. Next, in step 2704, a target position of the tool is calculated based on the sensing step. The target position is calculated to compensate for predictive errors and random errors in the movements of the tool. For example, CPU 1724 may calculate target position T based upon position sensors 1710 sensing the position of machine tool 1708 at point B. Target position T may be calculated to compensate for predictive errors and random errors in the movements of tool 1708. Target position T may be calculated to maximize the probability of point C, i.e., the actual position after the servo, being on the target tool path between points (1,4,0) and (5,1,6) after the effects of the predictive errors during the servo. In a final step 2706, the sensing and calculating steps are repeated for other movements of the tool. For example, the position of the tool at each of points C, D, E and F may be sensed and a new target position may be calculated after and based upon each of these sensing steps.

[00337] 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.

[00338] 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 adjusting path data for a motive body, said method comprising the steps of:
 - identifying (2102) four consecutive points of the path data including a first point, a second point, a third point and a fourth point;
 - determining (2104) whether the second point is suitable for smoothing;
 - if the second point is suitable for smoothing, then:
 - locating (2106) a first arc defined by the first point, the second point and the fourth point;
 - locating (2108) a second arc defined by the first point, the third point and the fourth point;
 - moving (2110) the second point into an area between the first arc and the second arc; and
 - replacing (2112) a line move associated with the second point with at least one arc move associated with the second point.
2. The method of claim 1 wherein the at least one arc move is tangent to both a preceding move and a succeeding move.
3. The method of claim 1 wherein the at least one arc move comprises two arc moves, the two arc moves being tangent to one another.
4. The method of claim 1 comprising the further step of moving the third point into the area between the first arc and the second arc.
5. The method of claim 1 wherein the line move and the at least one arc move are also associated with the third point.

6. The method of claim 1 comprising the further steps of:
identifying a fifth consecutive point succeeding the fourth point;
determining whether the third point is suitable for smoothing;
if the third point is suitable for smoothing, then:
 locating a third arc defined by the second point, the third point and the fifth point;
 locating a fourth arc defined by the second point, the fourth point and the fifth point;
 moving the third point into an area between the third arc and the fourth arc;
and
 replacing a line move associated with the third point with at least one arc move associated with the third point.

7. A method of adjusting path data for a motive body, said method comprising the steps of:
identifying (1802) three consecutive points of the path data;
ascertaining (1804) at least one of:
 a first distance between a circular arc defined by the three points and a middle portion of a first line extending between a first of the points and a second of the points; and
 a second distance between the circular arc and a middle portion of a second line extending between the second point and a third of the points;
determining a turn angle between the first line and the second line; and
performing smoothing on the second point if:
 at least one of the first distance and the second distance is less than a threshold distance; and
 the turn angle is greater than a threshold angle.

8. The method of claim 7 wherein smoothing is performed on the second point if a first move from the first point to the second point comprises a first line move and a second move from the second point to the third point comprises a second line move.

9. The method of claim 8 wherein the step of performing smoothing includes replacing at least one of the first line move and the second line move with at least one arc move.

10. The method of claim 7 wherein the threshold line distance is approximately between 0.0002 inch and 0.001 inch.

11. The method of claim 7 wherein the threshold angle is approximately between 130 degrees and 160 degrees.

12. The method of claim 7 wherein the first distance is between the circular arc and a midpoint of the first line, and the second distance is between the circular arc and a midpoint of the second line.

13. The method of claim 12 wherein smoothing is performed on the second point if each of the first distance and the second distance is less than the threshold distance.

14. A method of adjusting path data for a motive body, said method comprising the steps of:

identifying (1902) four consecutive points of the path data including a first point, a second point, a third point and a fourth point;

locating (1904) a first arc defined by the first point, the second point and the fourth point;

locating (1906) a second arc defined by the first point, the third point and the fourth point; and

moving (1908) the second point into an area between the first arc and the second arc.

15. The method of claim 14 wherein the second point is moved in a direction toward a closest point on the second arc.

16. The method of claim 14 comprising the further step of moving the third point into the area between the first arc and the second arc.

17. The method of claim 16 comprising the further step of determining whether the first point, the second point and the third point are suitable for smoothing.

18. The method of claim 17 wherein:

if each of the first point, the second point and the third point is suitable for smoothing, moving the second point less than 45% of a distance between the second point and the second arc;

if the second point is suitable for smoothing, one of the first point and the third point is suitable for smoothing, and an other of the first point and the third point is not suitable for smoothing, moving the second point approximately between 45% and 55% of a distance between the second point and the second arc; and

if the second point is suitable for smoothing, and neither of the first point and the third point is suitable for smoothing, leaving the second point unmoved.

19. The method of claim 18 comprising the further steps of:
identifying a fifth consecutive point succeeding the fourth point;
locating a third arc defined by the second point, the third point and the fifth point;
locating a fourth arc defined by the second point, the fourth point and the fifth point;
and
moving the third point and the fourth point into an area between the third arc and the fourth arc.

20. The method of claim 16 wherein the second point is moved in a direction toward a closest point on the second arc, and the third point is moved in a direction toward a closest point on the third arc.

21. A method of adjusting path data for a motive body, said method comprising the steps of:

identifying (2002) a line move extending from a first point of the path data to a second point of the path data; and

replacing (2004) the line move with a first arc move and a second arc move, the first arc move beginning at the first point and terminating at a junction point, the second arc move beginning at the junction point and terminating at the second point.

22. The method of claim 21 wherein the first arc move is tangent to a preceding move at the first point.

23. The method of claim 21 wherein the first arc move is tangent to the second arc move at the junction point.

24. The method of claim 21 wherein the second arc move is tangent to a succeeding move at the second point.

25. The method of claim 21 wherein the first arc move has a first radius and the second arc move has a second radius different from the first radius.

26. A method of processing path data for a motive body, said method comprising the steps of:

identifying (2202) a plurality of points along a target path;

determining (2204) a maximum allowable stopping distance of the motive body at a last of the points;

ascertaining (2206) a last maximum allowable speed of the motive body at the last point such that the motive body may come to a stop within the maximum allowable stopping distance; and

establishing (2208) an other maximum allowable speed of the motive body at an other one of the points such that the motive body may possibly slow down to the last maximum allowable speed upon arriving at the last point.

27. The method of claim 26 comprising the further steps of:

establishing a maximum allowable level of deviation of an actual path from the target path;

determining a maximum allowable tolerance speed of the motive body at the other point, the maximum allowable tolerance speed being dependent upon the maximum allowable level of deviation; and

identifying a lesser speed equal to a lesser of the maximum allowable tolerance speed and the other maximum allowable speed at the other point.

28. The method of claim 27 comprising the further step of establishing a preceding maximum allowable speed of the motive body at a point preceding the other point such that the motive body may possibly slow down to the lesser speed upon arriving at the other point.

29. The method of claim 28 comprising the further steps of:

determining a maximum allowable tolerance speed of the motive body at the preceding point dependent upon the maximum allowable level of deviation; and

identifying a lesser of the maximum allowable tolerance speed of the motive body at the preceding point and the preceding maximum allowable speed at the preceding point.

30. The method of claim 25 wherein the maximum allowable stopping distance at the last point corresponds to a segment length between the last point and a subsequent point, the subsequent point being adjacent to and downstream of the last point.

31. The method of claim 30 wherein the subsequent point may comprise one of an end of the target path and a corner point.

32. The method of claim 26 wherein the step of ascertaining a last maximum allowable speed comprises using an S-curve to relate the maximum allowable stopping distance to the last maximum allowable speed.

33. A method of processing path data for a motive body, said method comprising the steps of:

identifying (2302) a plurality of points along a target path;

establishing (2304) a maximum allowable level of deviation of an actual path from the target path;

ascertaining (2306) a curvature of the target path at a plurality of locations along the target path;

determining (2308) a maximum allowable tolerance speed of the motive body at each of the points along the target path, the maximum allowable tolerance speed being dependent upon the maximum allowable level of deviation and the curvature;

determining (2310) a maximum allowable stopping distance of the motive body at a last of the points;

ascertaining (2312) a last maximum allowable stopping speed of the motive body at the last point such that the motive body may come to a stop within the maximum allowable stopping distance;

establishing (2314) an other maximum allowable stopping speed of the motive body at each of the points other than the last point such that the may possibly slow down to the last maximum allowable speed upon arriving at the last point; and

identifying (2316) a lesser of the maximum allowable tolerance speed and the maximum allowable stopping speed at each of the points.

34. The method of claim 33 wherein the maximum allowable stopping distance at the last point corresponds to a segment length between the last point and a subsequent point, the subsequent point being adjacent to and downstream of the last point.

35. The method of claim 34 wherein the subsequent point may comprise one of an end of the target path and a corner point.

36. The method of claim 33 wherein the step of ascertaining a last maximum allowable stopping speed comprises using an S-curve to relate the maximum allowable stopping distance to the last maximum allowable stopping speed.

37. The method of claim 33 wherein the step of establishing an other maximum allowable stopping speed comprises using an S-curve to relate segment lengths between the other points and the last point to the other maximum allowable stopping speed.

38. The method of claim 33 wherein the step of establishing an other maximum allowable stopping speed comprises ensuring that a respective stop distance at each of the other points is one of less than and equal to a sum of a respective segment length between the other point and the maximum allowable stopping distance at the last of the points.

39. The method of claim 33 comprising the further step of moving the motive body at the lesser of the maximum allowable tolerance speed and the maximum allowable stopping speed at each of the points.

40. A method of processing path data for a motive body, said method comprising the steps of:

identifying (2402) a plurality of points along a target path, each selected two of the points being separated by a respective segment length;

determining (2404) a maximum allowable stopping distance of the motive body at a last of the points; and

ascertaining (2406) a maximum allowable speed of the motive body at an other one of the points such that a required stopping distance of the motive body at the other point is one of equal to and less than a sum of the maximum allowable stopping distance at the last point and the segment length between the other point and the last point.

41. The method of claim 40 comprising the further steps of:

establishing a maximum allowable level of deviation of an actual path from the target path;

determining a maximum allowable tolerance speed of the motive body at the other point, the maximum allowable tolerance speed being dependent upon the maximum allowable level of deviation; and

identifying a lesser speed equal to a lesser of the maximum allowable tolerance speed and the maximum allowable speed at the other point.

42. The method of claim 41 comprising the further step of establishing a preceding maximum allowable speed of the motive body at a point preceding the other point such that a required stopping distance of the motive body at the preceding point is one of equal to and less than a sum of the maximum allowable stopping distance at the last point and the segment length between the preceding point and the last point.

43. The method of claim 42 comprising the further steps of:

determining a maximum allowable tolerance speed of the motive body at the preceding point dependent upon the maximum allowable level of deviation; and

identifying a lesser of the maximum allowable tolerance speed of the motive body at the preceding point and the preceding maximum allowable speed at the preceding point.

44. The method of claim 41 comprising the further step of ascertaining a curvature of the target path at a plurality of locations along the target path, wherein the maximum allowable tolerance speed of the motive body at the other point is dependent upon the curvature.

45. The method of claim 40 wherein the maximum allowable stopping distance at the last point corresponds to a segment length between the last point and a subsequent point, the subsequent point being adjacent to and downstream of the last point.

46. The method of claim 45 wherein the subsequent point may comprise one of an end of the target path and a corner point.

47. The method of claim 40 wherein the step of ascertaining a maximum allowable speed at the other point comprises using an S-curve to relate the required stopping distance at the other point to the maximum allowable speed at the other point.

48. A method of operating a motive body, said method comprising the steps of:

- identifying (2502) a target path;
- commanding (2504) the motive body to follow the target path from an initial position;
- sensing (2506) a first actual position of the motive body after the commanding step;
- and
- calculating (2508) a first target subpath from the first actual position to correct for predictive errors in movement of the motive body.

49. The method of claim 48 comprising the further step of empirically determining the predictive errors in movement of the motive body, the empirically determining step occurring before the identifying step.

50. The method of claim 48 wherein the target subpath is calculated to compensate for random error in movement of the motive body from the initial position to the actual position.

51. The method of claim 48 wherein the calculating step is dependent upon a deviation of the first actual position of the motive body from the target path.

52. The method of claim 48 comprising the further steps of:

- directing the motive body to follow the first target subpath;
- sensing a second actual position of the motive body after the directing step; and
- calculating a second target subpath to correct for errors in movement of the motive body.

53. The method of claim 48 comprising the further step of determining acceleration needed to complete the first target subpath within a desired period of time and at a desired final velocity.

54. The method of claim 48 comprising the further step of calculating desired acceleration levels along the first target subpath.

55. The method of claim 48 wherein the calculating step includes at least one of Centripetal Force Compensation for Arc Motion, Tangential Acceleration

Compensation, Stick Friction Compensation, Leadscrew and Backlash Compensation, Spindle Softness Compensation, and System Delay Compensation.

56. The method of claim 48 wherein the motive body continues to move past the first actual position during the calculating step.

57. A method of operating a motive body, said method comprising the steps of:

empirically (2602) determining predictive errors in movement of the motive body;
monitoring (2604) actual movements of the motive body; and
controlling (2606) the actual movements of the motive body dependent upon:
the empirically determined predictive errors; and
random errors in the actual movements of the motive body.

58. The method of claim 57 comprising the further step of commanding the tool to follow a target path from an initial position, the monitoring step including sensing a first actual position of the motive body after the commanding step, the controlling step including calculating a first target subpath from the first actual position to correct for the predictive errors and the random errors in the movement of the motive body.

59. The method of claim 58 wherein the calculating of the first target subpath is dependent upon a deviation of the first actual position of the motive body from the target path.

60. The method of claim 58 comprising the further steps of:
directing the motive body to follow the first target subpath;
sensing a second actual position of the motive body after the directing step; and
calculating a second target subpath to correct for errors in movement of the motive body.

61. The method of claim 58 comprising the further step of determining acceleration needed to complete the first target subpath within a desired period of time and at a desired final velocity.

62. The method of claim 57 wherein the empirically determining step precedes the monitoring step and the controlling step.

63. The method of claim 57 wherein the controlling step includes compensating for the predictive errors and the random errors.

64. The method of claim 57 comprising the further step of calculating desired acceleration levels of the motive body.

65. The method of claim 57 wherein the controlling step includes at least one of Centripetal Force Compensation for Arc Motion, Tangential Acceleration Compensation, Stick Friction Compensation, Leadscrew and Backlash Compensation, Spindle Softness Compensation, and System Delay Compensation.

66. A method of operating a motive body having a plurality of sequential movements, said method comprising the steps of:

sensing (2702) an actual position of the motive body after a first of the movements of the motive body;

calculating (2704) a target position of the motive body based on the sensing step, the target position being calculated to compensate for predictive errors and random errors in the movements of the motive body; and

repeating (2706) the sensing and calculating steps for other ones of the movements of the motive body.

67. The method of claim 66 wherein the predictive errors are empirically determined.

68. The method of claim 66 wherein the predictive errors are empirically determined before the sensing and calculating steps.

69. The method of claim 66 wherein the target position is calculated to compensate for predictive errors in both the first movement and the other movements of the motive body.

70. The method of claim 66 comprising the further step of determining acceleration needed to reach the target position within a desired period of time and at a desired final velocity.

71. The method of claim 66 wherein the calculating step includes at least one of Centripetal Force Compensation for Arc Motion, Tangential Acceleration

Compensation, Stick Friction Compensation, Leadscrew and Backlash Compensation, Spindle Softness Compensation, and System Delay Compensation.

72. The method of claim 66 wherein the motive body continues to move past the actual position during the calculating step.

73. A method of operating a motive body, said method comprising the steps of:

- sensing (2702) an actual position of the motive body;
- calculating (2704) a position error between the actual position of the motive body and a desired position of the motive body;
- comparing (2602, 2604) the position error to a specified error tolerance; and
- adjusting (2606) a speed of the motive body dependent upon the comparing step.

74. The method of claim 73 wherein the adjusting step includes increasing the speed of the motive body if the position error is below the specified error tolerance, and decreasing the speed of the motive body if the position error is above the specified error tolerance.

75. A method of operating a motive body, said method comprising the steps of:

- sensing (2702) an actual position of the motive body along each of a plurality of axes;
- calculating (2704) a multi-dimensional position error between the actual position of the motive body in each of the axes and a multi-dimensional reference trajectory;
- producing (2602, 2604) a multi-dimensional position error signal dependent upon the calculating step; and
- controlling (2606) at least one of a speed and a direction of the motive body dependent upon the multi-dimensional position error signal.

76. The method of claim 75 wherein the multi-dimensional position error is calculated as a vector.

77. The method of claim 75 wherein said controlling step includes using a tangential error feedback controller and a normal error feedback controller.

78. The method of claim 77 wherein said controlling step includes assigning more weight to an output of said normal error feedback controller than to an output of said tangential error feedback controller.

79. The method of claim 75 wherein the axes include an x-axis, a y-axis and a z-axis of a machine tool.