

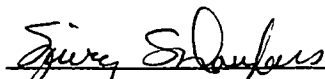
To the Graduate Council:

I am submitting herewith a thesis written by Van Buren Graves entitled "Geometric Error Compensation Software in an Open Architecture, PC-based Machine Tool Controller." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mechanical Engineering.



William R. Hamel, Major Professor

We have read this dissertation
and recommend its acceptance:



Spivey S. Douglass



Reid L. Kress

Accepted for the Council:



Associate Vice Chancellor and
Dean of The Graduate School

Geometric Error Compensation Software in an Open
Architecture, PC-based Machine Tool Controller

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Van Buren Graves
August 1999

DEDICATION

To my parents,
For providing me with love, guidance, and direction
And helping to make me who I am today.

To my children,
For generously giving up time with Dad while I was in school
And for giving me such joy and happiness.

To my wife,
For giving such gracious support, love, and sacrifice
And for being my best friend.

To my Savior,
For giving me the drive and patience to finish this race
And for giving me the desire to explore the mysteries of His
wonderful creation.

ACKNOWLEDGEMENTS

Funding for this project was provided through a Cooperative Research And Development Agreement (CRADA) between Manufacturing Data Systems, Inc. (MDSI) and the Department of Energy. I would like to thank Ray Ford, Program Manager of the Technology Partnership Program, for providing the resources for me to complete this work. Being able to use a work project as my thesis topic made it possible to complete my degree in a more timely fashion.

I also wish to express my appreciation to MDSI for providing excellent technical support during this project. The external compensation template provided by MDSI was instrumental in the development of the Geometric Error Compensation System, and I hope that the architecture developed during this project is of benefit to MDSI.

I would like to thank the members of my graduate committee, Dr. William Hamel, Dr. Spivey Douglass, and Dr. Reid Kress. Their feedback and guidance during the preparation of this document were very much appreciated. There were several times when progress slowed considerably, and Spivey kept encouraging me to keep the momentum going forward.

Appreciation also goes out to my Section Head at the Robotics and Process Systems Division, Dr. Steven Schrock. He was always supportive of my efforts, and allowed me to adjust my work schedule to attend classes when necessary. Steve, I also want to thank you for giving me the flexibility to work on this project at Y-12. I hope that many of the controls concepts and hardware I've been exposed to during these past several months will be useful in future robotics work at the Laboratory.

This project would not have begun without the efforts of Theodore Williams, who pushed through the formalities associated with the CRADA process. Ted,

thanks for your enthusiastic support, guidance, and most of all, your friendship. Maybe now we can do more fishing!

I also wish to thank Dr. Robert Hocken of the Precision Engineering Laboratory at the University of North Carolina, Charlotte and Dr. Alkan Donmez of the National Institute of Science and Technology. Their papers on this subject were very helpful references, and I appreciate the time they took to answer several questions I had.

I also wish to extend a special thanks to the staff of ORNL Publishing Services who helped turn this work into a document that I am very proud of.

Finally, I want to thank my wife, René. I remember the times I asked you to occupy the kids in the evenings while I studied or did homework, even though you had been with them the entire day. There were also several occasions when we had to change our plans because of my school responsibilities. I realize this process has taken a long time, and I appreciate the sacrifices you have made that enabled me to finish. I love you.

ABSTRACT

A project was initiated at the Oak Ridge Y-12 Plant to implement software geometric error compensation within a PC-based machine tool controller from Manufacturing Data Systems, Inc. This project may be the first in which this type of compensation system was implemented in a commercially available machine tool controller totally in software. Previous implementations typically required using an external computer and hardware to interface through the position feedback loop of the controller because direct access to the controller software was not available.

A mathematical error model of the lathe was created using homogeneous matrix transforms to relate the positions of the machine's slides to each other and to a world reference system. Equations describing the effects of the geometric errors were derived from the model. A software architecture was developed to support geometric error compensation for machine tools with up to 3 linear axes. Rotary axes were not supported in this implementation, but the developed architecture would not preclude their support in the future. Specific implementations will be dependent upon the configuration of the machine tool. The test-bed machine for this project was a 2-axis Excella 921 T-base lathe.

A laser measuring system from Automated Precision, Inc. was used to characterize the lathe's geometric errors as functions of axis position and direction of motion. Multiple data files generated by the laser system were combined into a single Error File that was read at system startup and used by the compensation system to provide real-time position adjustments to the axis servos.

A Renishaw Ballbar was used to evaluate the compensation system. Static positioning tests were conducted in an attempt to observe improved positioning accuracy with the compensation system enabled. These tests gave inconsistent

results due to the lathe's inability to position the tool repeatably. The development of the architecture and compensation template will provide a baseline platform for investigating other types of error compensation in the future.

TABLE OF CONTENTS

	Page
 CHAPTER 1	
Introduction.....	1
1.1 Machine Tool Errors.....	4
1.2 Accuracy Issues.....	5
1.3 Project Description and Scope.....	6
1.4 Test Bed Description.....	11
 CHAPTER 2	
Literature Review.....	16
 CHAPTER 3	
OpenCNC Controller Description.....	22
3.1 OpenCNC System Overview.....	22
3.1.1 Hardware.....	23
3.1.2 Software.....	25
3.2 Test Bed Retrofit.....	28
 CHAPTER 4	
Error Compensation.....	33
4.1 General Error Model.....	34
4.1.1 Homogeneous Transformation Matrices.....	34
4.1.2 Application of HTMs to Linear Axes.....	38
4.1.3 Total Error Equation for a Machine Tool.....	42
4.2 Excello Error Model.....	46
4.3 Excello Error Characterization.....	52
4.3.1 Laser Measuring System Description.....	53
4.3.2 Excello Error Data Acquisition.....	57
4.3.2.1 X-axis measurements.....	59
4.3.2.2 Z-axis measurements.....	65
 CHAPTER 5	
System Implementation.....	72
5.1 OpenCNC Overview.....	72
5.2 GECS Implementation Overview.....	73
5.2.1 Generic Approach.....	76
5.2.2 Machine-specific Approach.....	80
5.3 Excello Implementation Specifics.....	82
5.3.1 Error File Contents.....	83
5.3.2 Error Arrays.....	83

5.3.3 Units	84
5.3.4 Conversion Utility	85
5.3.5 Source Code Description	87
CHAPTER 6	
System Evaluation.....	91
6.1 Two-axis Contour Test.....	93
6.1.1 Description	93
6.1.2 Results	96
6.2 Conclusions.....	101
6.3 Future Work.....	102
REFERENCES.....	105
APPENDICES.....	108
APPENDIX A	
Excello Error Equation Development Using Mathematica.....	109
APPENDIX B	
Example GECS Error File.....	111
APPENDIX C	
Excello 5-D Laser Error Measurement Diagrams.....	114
VITA	121

LIST OF TABLES

TABLE	PAGE
4-1. HTM error term descriptions	40
4-2. Turning center HTM assignments.....	44
4-3. Excello error equation components	58
5-1. OpenCNC variable groups	74
5-2. Three-axis machine error components.....	77
5-3. Three-axis machine error array names	78
5-4. Error File parameters	83
5-5. Excello error array names	84

LIST OF FIGURES

FIGURE	PAGE
1-1. Different three-axis machine tool configurations.....	10
1-2. Front view of Excelllo 921 lathe with operator control panel	11
1-3. Excelllo 921 computer model	12
1-4. Excelllo mechanical lead-screw compensation system	13
1-5. Excelllo 921 control cabinet with GE2000 controller	15
2-1. External error compensation system block diagram.....	19
3-1. OpenCNC functional hardware diagram	24
3-2. OpenCNC functional software architecture.....	26
3-3. Excelllo control cabinet after OpenCNC retrofit.....	29
3-4. Industrial PC with control hardware	30
3-5. Operator control panel after MDSI retrofit.....	31
3-6. OpenCNC operator interface.....	32
4-1. A point in one frame relative to another frame.....	35
4-2. Error motions for a typical linear slide	38
4-3. Relationship between cutting tool and workpiece	42
4-4. HTM assignments on a two-axis turning center.....	44
4-5. Excelllo error model diagram	47
4-6. 5-D Laser Measurement System components	54
4-7. Simplified block diagram of 5-D linear measurements.....	55
4-8. Simplified block diagram of 5-D straightness/angular measurements	56
4-9. X-axis error measurement setup.....	59
4-10. X-axis lead-screw errors	61
4-11. X-axis horizontal straightness errors.....	62
4-12. X-axis yaw errors	63
4-13. Squareness error measurement.....	64
4-14. Squareness error data.....	65
4-15. Z-axis error measurement setup.....	66
4-16. Z-axis lead-screw errors.....	67
4-17. Z-axis horizontal straightness errors	68
4-18. Z-axis yaw errors.....	69

4-19.	Parallelism error measurement.....	70
4-20.	Parallelism error data	71
5-1.	Control system compensation loop flowchart.....	76
5-2.	Compensation process time-lines.....	81
5-3.	Conversion utility user interface	86
5-4.	GeoComp flowchart	88
5-5.	Compensation calculation flowchart.....	89
6-1.	Ballbar setup.....	94
6-2.	Static ballbar results with GECS disabled.....	97
6-3.	Static ballbar results with GECS enabled.....	98
6-4.	X error components in semicircular move.....	99
6-5.	Z error components in semicircular move.....	100

LIST OF ABBREVIATIONS

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASME	American Society of Mechanical Engineers
CMM	Coordinate Measuring Machine
CPU	Central Processing Unit
CRADA	Cooperative Research and Development Agreement
DEC	Digital Equipment Corporation
DOE	Department of Energy
DOE-DP	DOE Defense Programs
Encoder-IP	Encoder Industry Pack
GECS	Geometric Error Compensation System
GUI	Graphical User Interface
HTM	Homogeneous Transformations Matrix
I/O	input/output
IEC	International Electrotechnical Commission
IP	Industry Pack
LSEC	Lead Screw Error Compensation
LVDT	Linear Variable Differential Transformer
MDSI	Manufacturing Data Systems, Inc.
NC	Numerically Controlled
NIST	National Institute of Standards and technology
NWC	Nuclear Weapons Complex
OMAC	Open Modular Architecture Controls
PC	Personal Computer
PLC	Programmable Logic Controller

Chapter 1

Introduction

The U.S. Department of Energy's (DOE's) Oak Ridge Y-12 Manufacturing Plant (Y-12) is continually striving to produce higher quality parts at a reduced cost.* This is also the goal of most every manufacturing facility, both in the United States and abroad. When manufacturing nuclear weapon components, which is the primary mission of the Y-12 Plant, high accuracy is essential. In addition, consistency and repeatability from one part to the next are required to ensure proper assembly.

During the Cold War, national defense budgets were high, and millions of dollars were spent developing manufacturing and machining processes necessary to create and maintain the nation's nuclear stockpile. In today's post-Cold War environment, the manufacture of new weapons components has essentially ceased, and funding for process technology development has been shrinking for several years. As budgets decreased, many of the personnel experienced in the production of weapons components left, or their efforts were focused on non-defense projects.

* Research sponsored by U.S. Department of Energy Defense Programs, Office of Development and Technology Transfer, under contract DE-AC05-84OR21400, managed by Lockheed Martin Energy Systems, Inc.

Additionally, the size of the Y-12 manufacturing "footprint," or square footage devoted to production facilities, has decreased by more than 50% during the past decade.

Furthermore, there have been few, if any, upgrades to the machine tool controllers. Most of the plant's numerically controlled (NC) lathes, which are the workhorses of weapons manufacturing facilities, were procured in the 1960s. Because of rigid design and construction of the machines and because of high quality mechanical maintenance, the Excellos are still capable of producing high quality parts. However, the controllers on the machines, which take instructions from the part programs and convert them into axis motions, have become obsolete. The last significant NC controller upgrade occurred in the mid-1980s when the plant standardized on the General Electric® GE2000 controller. Although this controller was state-of-the-art when it was purchased, its technology is now dated, and the controller is no longer produced. Maintenance of the plant's controllers is becoming a major issue as spare part inventories are depleted and vendor support is no longer available. The cost of retrofitting these machines with current proprietary controllers is prohibitive, and lower cost alternatives are being investigated.

As computer processing power has increased, costs have decreased, and now it is possible to buy machine tool controllers that utilize personal computers (PCs) with common plug-in motion control cards that offer more functionality than did the GE2000. These controllers are often advertised as being *open-architecture*, which suggests certain hardware and software characteristics, but the interpretation of this term seems to vary between vendors. Open-architecture hardware implies that it utilizes standard, commercially available, off-the-shelf components that can be replaced as failures occur or as new technology becomes available. Open-architecture software implies that the user has some level of

access to the inner workings of the controller and can customize or enhance its operation for specific purposes.

One organization whose attentions are directed toward the development of open controllers is the Open Modular Architecture Controls (OMAC) Users' Group. The Users' Group originated from a joint effort between Chrysler, Ford, and General Motors in 1994 to create a requirements document [1] for controller vendors to better meet the needs of the automotive industry. This requirements document contained guidelines for a common set of functionality and was intended for the controller supplier and technology development communities. The document stated that to meet those needs, controllers needed to be "economical, maintainable, open, modular, and scaleable for automotive manufacturing applications." Their definition of openness was "allowing the integration of off-the-shelf hardware and software components into a controller infrastructure that supports a de facto standard environment." Modularity implied that some controller functions could be replaced or upgraded through the use of off-the-shelf components. OMAC comprises companies representing the end-user community, original equipment manufacturers, and technology providers/integrators. It was formed to provide a forum for idea exchange and initiative development through its internal working groups. One of the major objectives of these working groups is the development of specific Application Programming Interfaces (APIs) that would standardize some of the common functionalities.

Manufacturing Data Systems, Inc. (MDSI®) of Ann Arbor, Michigan is a member of OMAC and has developed a PC-based, open-architecture controller that is currently being investigated by Y-12. A characteristic that sets it apart from other PC-based controllers is that it utilizes no motion control cards; it is a software-based controller, which means that the motion control servo loop is

closed by the Intel® Pentium® central processing unit (CPU) of the PC itself. The controller software is called OpenCNC®, and it currently runs under two operating systems: Windows NT® from Microsoft® Corporation and QNX® from QNX Software Systems Ltd.

An area of particular interest to the Y-12 Plant is OpenCNC's ability to support software compensation of machine tool errors other than that currently implemented for lead-screw compensation. This compensation will be used to minimize the errors in the part surface caused by repeatable static errors in machine tool geometry.

1.1 Machine Tool Errors

Before discussing the details of this thesis, it is important to develop a set of definitions for the pertinent parameters that will be dealt with. The first one needed is accuracy. In the *Machine Tool Accuracy* report [2], Hocken defined *accuracy* relative to a metrologist as a "measure of the degree of conformance to recognized international (or national) standards." For applications related to machine tools, he defined accuracy as the "degree of conformance of the finished part to dimensional and geometric specifications." *Errors*, then, are measures of the non-conformance to either a standard or to a part specification.

Quasistatic machine tool errors are those positional errors between the tool and workpiece that change slowly in time. These types of errors can be categorized into three classes: *kinematic*, or *geometric*, errors caused by inaccuracies in the machine tool itself or in the relative motion between its components; *thermal* errors caused by temperature variations in either the machine structure or the environment it resides in; and *load-induced* errors caused by the weights of the machine tool components, over-constrained slides, and part weights [2]. This thesis will only be concerned with geometric errors.

Linear machine tool slides or carriages, like all rigid bodies, have six degrees of freedom of motion, three of which are translational, and three of which are rotational. Typically, five of these degrees of freedom are constrained. Errors of motion in the unconstrained sixth degree of freedom are typically referred to as *scale errors*, since the motion is usually measured using some kind of scale, or *lead-screw errors*, since that is usually the means of providing motion to the carriage. Undesired linear motions in directions orthogonal to the intended motion direction are referred to as *straightness errors*, usually either a *horizontal straightness error* or a *vertical straightness error*, depending on the orientation of the carriage. Rotations of the carriage are referred to as *angular errors* and are usually designated as *roll*, *pitch*, and *yaw*, analogous to the definitions used in aircraft.

1.2 Accuracy Issues

Part accuracy is another aspect of manufacturing in which improvements are needed. As companies strive for higher part quality, they require higher percentages of parts to be manufactured within specified tolerances. Different methods of improving the accuracy of machine tools have been used in the past. One such method is to actually make modifications to the motion system of machine tool itself. A CNC machine tool is composed of several subsystems:

Physical structure. Rigidity is of utmost importance, but is inherently tied to the design of the machine. The user can do little, if anything, to change this characteristic.

Motion system. It consists of the lead screws, slides, rams, etc., and can be reworked or modified to improve the system's accuracy. However, this is very labor intensive and thus expensive.

Feedback system. It comprises encoders, resolvers, and other position feedback devices as well as any other types of sensors that monitor

various machine or process conditions. These components can also be replaced, as newer or more accurate technology is available.

Control system. The control system is the "intelligence" of the machine tool and is responsible for coordinating the motion of the tool based on the desired path in the part program with information from the feedback system. In a proprietary system, the user has little or no ability to modify the controller.

A more sophisticated method of improving a machine tool's accuracy is through performing an error *characterization* of the machine. Characterizing a machine tool implies creating a quantifiable description of the repeatable errors within the motion system of the machine tool, sometimes referred to as an error map. An example would be a data table that contained the errors associated with the pitch of the lead screw of an axis. Today's modern control systems can read such a table and in real-time can modify the tool path given in a part program in order to machine the desired path. As the machine's error characteristics change, the error map could be rebuilt, which maintains the accuracy of the machine without having to repeat all the costly physical modifications.

The difficulty with this approach is that currently it is a time-consuming task to characterize a machine. In addition to the six typical error motions for a linear slide, there are errors between axes themselves, referred to as squareness errors. Thus, for a three-axis mill, there are 21 error contributors that could be characterized. The time required to measure each of these error sources is prohibitive, so usually only the major error contributors of a given machine are mapped.

1.3 Project Description and Scope

In 1990, DOE Defense Programs (DOE DP) established the Technology Transfer Initiative, which later evolved into the Technology Partnership Program

directed by the DOE DP Office of Development and Technology Transfer. The program involved the DOE Weapons Design Laboratories and Production Facilities, including the Oak Ridge Y-12 Plant. The program was established to encourage the transfer of dual-use technology from the Nuclear Weapons Complex (NWC) facilities to U.S. industry to assist the U.S. industrial partners in being more globally competitive through the infusion of the technologies developed by the NWC. During this time, Y-12 sponsored more than \$80M in technology transfer projects, which supported over 140 Cooperative Research and Development Agreements (CRADAs) and other regional business initiative programs. The CRADA is the legal vehicle through which DOE and U.S. industry partners form teams to develop these dual-use technologies.

A CRADA was created between Y-12 and MDSI to implement within the OpenCNC controller a software architecture capable of compensating for geometric errors in two- and three-axis machine tools. As a means of limiting the scope of the project, only linear axes were to be supported in the initial implementation. Since rotary axes are typically found on machining centers that have four, five, or more controlled axes, the vast majority of two- and three-axis machines would be included in the project scope.

The implemented architecture would provide a means of *software error correction* for those machines under OpenCNC control. In their report to the National Institute of Standards and Technology (NIST), the Precision Engineering Laboratory at the University of North Carolina, Charlotte gave a definition of software correction that will be used in this project and thesis [3]:

Software correction is the use of preprocess data, a machine model, and indirect sensing of process parameters relevant to that model, in order to provide data to the control system for the correction of a nominal tool position with respect to a nominal part during the

process of machining or measuring using the actuators “normally” supplied with the machine.

The definition implies that a mathematical model of the errors of the machine tool would be developed and that the magnitude of those errors would be measured and electronically stored within the controller. This information would be gathered in an “off-line” fashion, that is, with no machining operations being performed. The error data would then be utilized by the control system during machining to compensate for the machine’s errors by modification of the command signals sent to the slide actuators.

Geometric error compensation techniques seem to be much more prevalent in inspection equipment. In fact, these techniques “have been adopted on almost all commercial CMMs [Coordinate Measuring Machines] but do not appear to be in wide use on machine tools[3].” In performing research for this project, all references to geometric error compensation of machine tools detailed machines being used in a laboratory environment. This project may be the first in which geometric error compensation techniques have been implemented within a commercially available machine tool controller without having to break into the electronic interface of the machine’s position feedback loop. In fulfilling the goal of implementing geometric software error correction, several additional goals would be achieved. First, an understanding of the necessary techniques would be gained. While straightforward lead-screw compensation has been implemented on many machine tools in Y-12, full geometric error compensation has not, mainly because of the lack of access to the inner workings of the proprietary controllers. In addition, implementing this type of software architecture within OpenCNC would possibly require access to some of the lowest levels of the servo routines of the system since the compensation would occur real-time within the feedback loop of the controller. This implementation process would test the “openness” of

the controller and would give an indication of the customizations that were possible with the system. Once this type of compensation was implemented, the software interface developed could be used for other types of compensation systems.

To limit the scope of the problem, only machines with linear axes were to be supported. While the compensation system would be implemented and tested on a two-axis lathe, it was important that the solution be generic enough to be able to handle machine tools that have three or more linear axes of motion. As will be discussed later in this document, the error equations developed will be unique to a particular configuration of machine tool. These configurations are based on the geometry of the machines, as described by Dr. Hocken in his Machine Tool Metrology course notes [4]. Figure 1-1 shows four different configurations of three-axis machines. The machine nomenclature in the figure, when read left to right, describes a path from the part to the tool. Each configuration would have different error equations because of the way the slides are stacked and move relative to one another.

Even though two specific machines may be different in size and physical appearance, if both are classified as XFYZ machines, then the same set of error equations could be used for both machines. Any constant parameters in the equations that would be affected by the size of the machine, such as offsets between slides (represented as coordinate systems), would be changed in the error input files for each machine. So, in theory, if error equations were developed for the four configurations shown in Figure 1-1, and if most three-axis milling machines could be considered to be one of these configurations, then OpenCNC would be able to compensate the majority of machine tools in the field. However, this is not meant to imply that it would necessarily be trivial for the customer to utilize these equations. A good understanding of the variables involved and

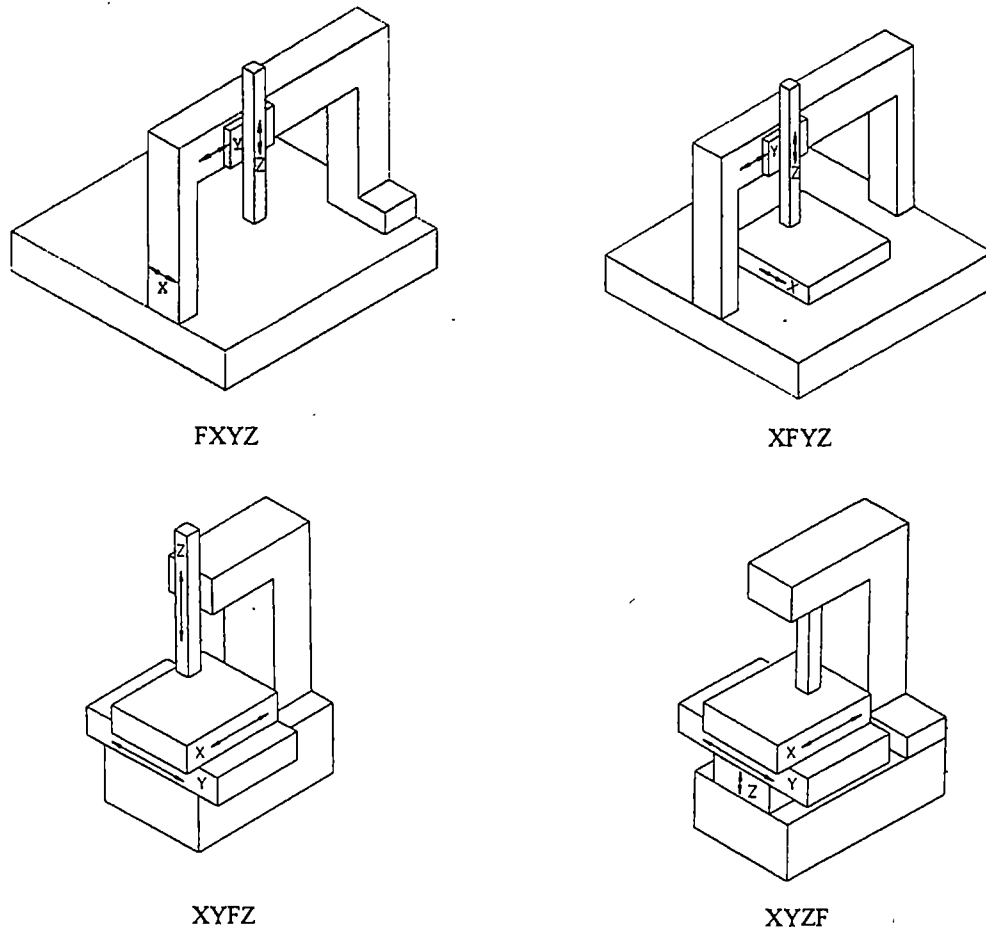


Figure 1-1. Different three-axis machine tool configurations (*used by permission*)

techniques used in deriving the equations for each configuration would have to be developed, through detailed documentation, data acquisition procedures, and training, so the end-user could correctly measure the values needed for input into the error tables.

1.4 Test Bed Description

While the scope of the project included two- and three-axis machines, the initial implementation was on a two-axis lathe. Shown in Figure 1-2, the Excello 921 is one of the machine tools most widely used by Y-12 in the production of nuclear weapons components. These lathes were designed and built specifically for Y-12.

A total of 45 Excello machines were bought between 1968 and 1970 [5]. Of these, 35 were of the 921 or 921T configurations, and 10 were of the 922 configuration. The 921T differed from the 921 in that it came equipped with a tool turret and was also capable of threading operations. The 922 configuration was built to tighter tolerances and was designed for greater accuracy than the

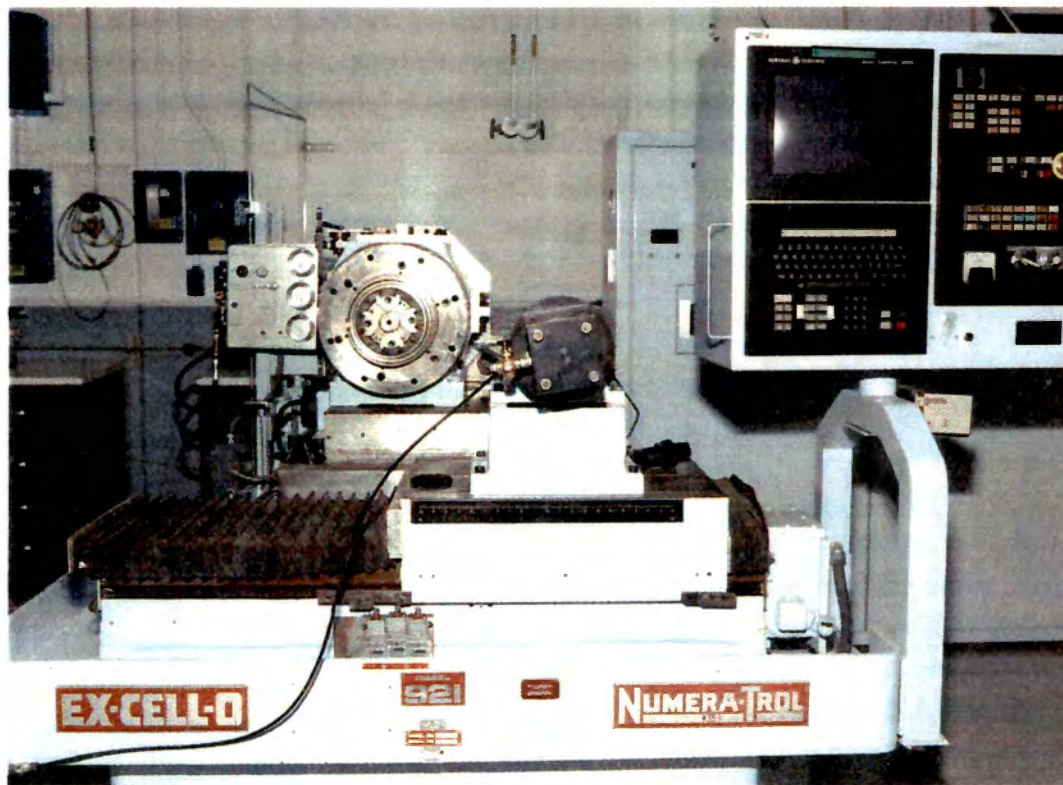


Figure 1-2. Front view of Excello 921 lathe with operator control panel

921. The acceptable surface contour tolerance for the 922 machines was ± 0.0005 inch; for the 921 machines, the tolerance was ± 0.001 inch. Bendix manufactured the original controller shipped with the Excello machines. The stroke of the 921 work slide, programmed as the Z-axis, is 12 inches, and the stroke of the tool slide, programmed as the X-axis, is 22 inches.

A computer model of the Excello 921 was developed to facilitate generation of the error model. The computer model was generated using the parametric solid modeling software package ProEngineer® and is shown in Figure 1-3. The figure highlights various components of the machine and the directions of positive motion as set in the machine controller. The tool-set sphere was used by the GE2000 controller to precisely set the internal variable containing the radius of the tool. A preprogrammed sequence of moves was generated by the system to touch the tool at various locations on the surface of the tool-set ball. When the

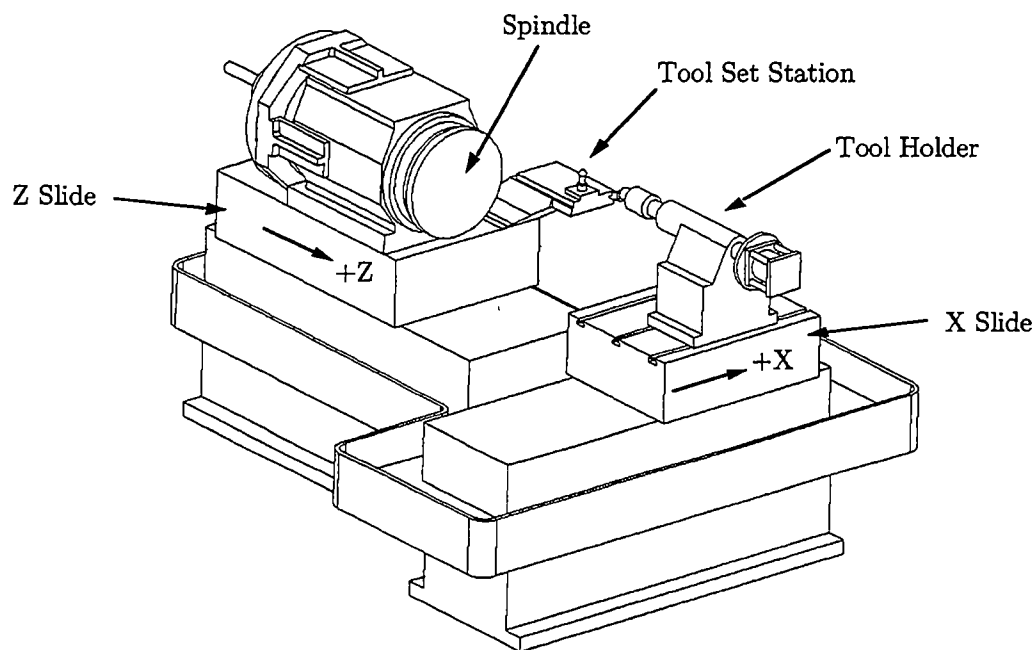


Figure 1-3. Excello 921 computer model

tool contacted the ball, motion stopped and positions of the slides were recorded. Calculations provided a very accurate location for the tip of the tool, which was used during subsequent machining operations.

As an interesting historical note, the Excello machines were delivered with lead-screw compensation capabilities. However, the compensation was accomplished using a mechanical cam assembly, illustrated in Figure 1-4, rather than through the controller. Within the mechanical compensator, a flat cam sized to match the length of the slide travel was mounted on the slide. The assembly converted the vertical motion of the follower into gain and loss signals on the feedback transducer. To “program” the cam, it was first scribed at 1-inch intervals. A lead-screw check was then performed using a certified inch bar, and

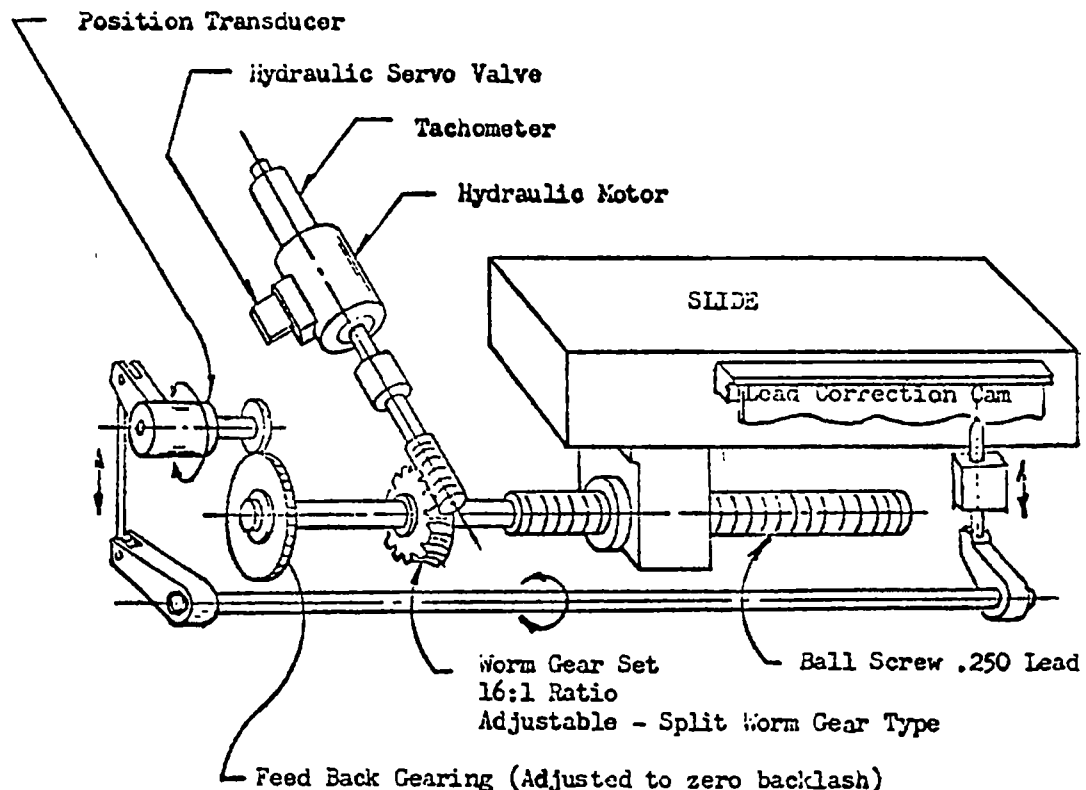


Figure 1-4. Excello mechanical lead-screw compensation system
(reprinted from the Excello 922 Instruction Manual, 1968)

the results were recorded. After multiplying the errors by the appropriate cam correction factor, points were drawn on the scribed inch markings and were connected by lines. The cam surface was then ground to the connecting lines and replaced on the slide. Results were verified and any necessary touch-up adjustment was performed. As machine controller technology advanced, the mechanical compensators were eventually replaced by software compensation techniques.

In the mid-1980s the controllers were replaced by the General Electric GE2000, which were state-of-the-art systems at the time. Like all other machine tool controllers, the GE2000 used proprietary hardware and software, thus ensuring that the users must purchase replacement or upgrade parts from the controller vendors. Large inventories of spare parts and circuit boards were procured so in-house maintenance could be performed. Figure 1-5 shows the test-bed Excella 921 control cabinet with the GE2000 controller before the MDSI retrofit took place.

During the retrofit some of the components shown in Figure 1-5 were removed since their functionality was taken over by the OpenCNC controller. The GE2000 CNC was obviously removed, along with the Digital Equipment Corporation (DEC) 11/23 computer that provided Distributed Network Control (DNC) capabilities. During the retrofit, it was decided that new servo amplifiers for the axes were needed for this machine along with a new spindle drive, which had already been removed from the cabinet. Any wiring that interfaced to these components was cut and wired to terminal blocks to allow easy connection with the MDSI hardware. While its functionality was no longer needed, the tape reader was left in the cabinet because removing it would have left an opening in the cabinet door.

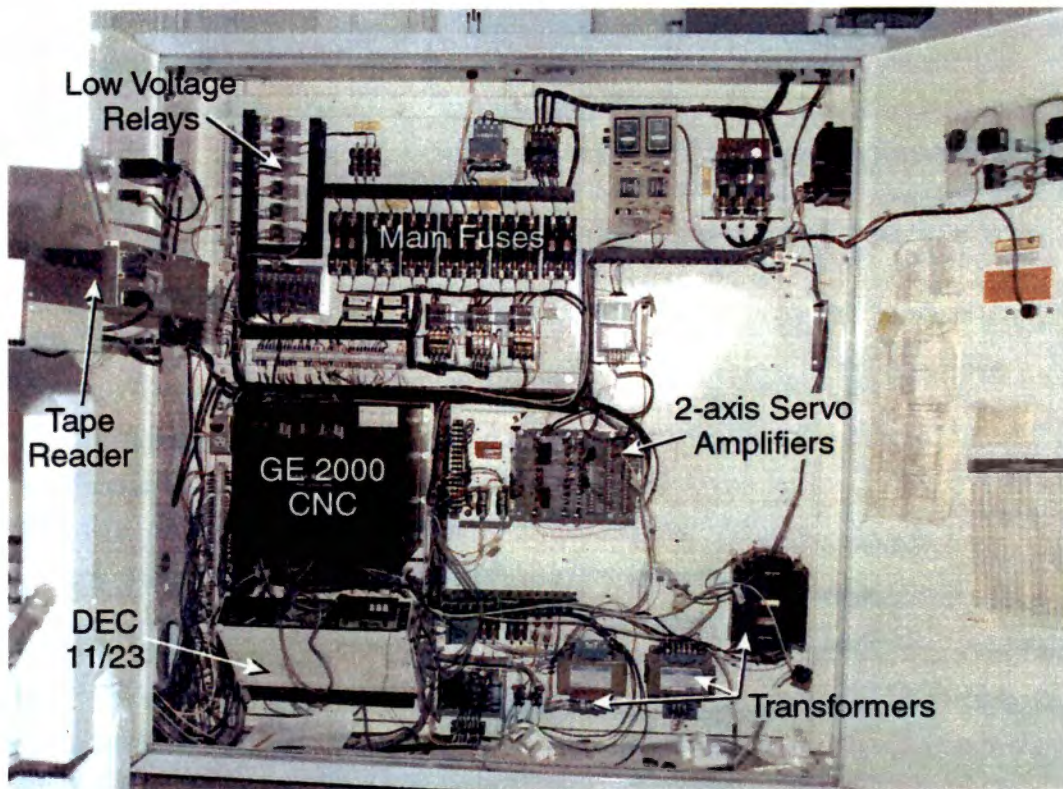


Figure 1-5. Excello 921 control cabinet with GE2000 controller

Chapter 2

Literature Review

In performing background research for this project, several references were located that were directly related to machine tool error compensation. However, much of the early compensation work was performed in facilities and laboratories operated by the Atomic Energy Commission, later called the Department of Energy, and typically was not documented in open literature.

The field of error compensation is one of several that comprise the body of knowledge referred to as *precision engineering*. While an exact definition of precision engineering is hard to develop and may be relative to the particular application, the basic concept is to develop machines and mechanisms that offer the highest performance levels available. Evans [6] gives a historical perspective on how the field of precision engineering has evolved over time and presents several examples resulting from the application of precision engineering techniques.

While the emphasis in the researched papers differed, there was a general consensus as to the technique of performing geometric error compensation on machine tools. This technique involved the application of kinematic analysis methods that are normally associated with robots.

In robotics, the location of the end-effector is typically determined using a technique called forward kinematics. It involves placing coordinate systems at each joint of the robot mechanism and using matrix transformations to describe the location and orientation of successive links (or components) relative to a nonmoving base reference system. Denavit and Hartenberg performed the early work in this area [7]. This was one of the earliest documents describing mechanisms using matrix notation and formed the basis of the DH-notation used in describing robotic motions.

One of the leading researchers in the area of machine tool errors is Dr. Robert Hocken, who formerly served at NIST and currently teaches at the Center for Precision Metrology at the University of North Carolina, Charlotte. In one of his earlier papers [8], Hocken described implementing geometric compensation on a Moore 5-Z, a three-axis bridge-configuration coordinate measurement machine. Rigid body kinematics techniques were used to analyze the machine structure. Ideal reference frames (coordinate systems) were chosen for each kinematic link in the machine. The use of "metrology bases" mounted onto some of the links served as origins for the reference frames. The metrology bases decoupled the reference frames from the imperfect machine geometry. The authors' choice of coordinate systems included the space system, the primary metrology base mounted on the column of the machine; the table system, a metrology base mounted on the moving table; and the object system, the coordinate system of the workpiece. A rotation matrix was developed between each frame, and the series of matrix multiplications gave the coordinates of a point in the object system relative to the space system. The author noted that strict adherence to sign conventions when creating the matrices can prevent problems that often occur when trying to do the same calculations using only analytic geometry.

Another paper describing machine tool error modeling was given by Soons [9]. The author described the errors associated with machine tools of arbitrary configuration using three separate models: (1) the general model, (2) the type-dependent model, and (3) the individual model. The general model used a series of transformation matrices to relate the errors in positioning the tool relative to the workpiece to errors in location of the coordinate frames attached to intermediate kinematic components of the machine. The type-dependent model attempted to quantify the accuracy of an individual machine at specific locations of its axes. It incorporated error characteristics of all the machines of identical configuration manufactured by a particular vendor. Specific error components included stiffness of the machine's components, thermal expansion of the components, and physical properties of the machine's slides, joints, and measuring systems. The individual model was obtained by utilizing piecewise polynomials to characterize the positioning errors of a given slide.

A paper by Donmez [10] presented a general methodology for modeling geometric machine tool errors using homogeneous transformation matrices (HTMs). The approach was similar to that developed using rotation matrices in earlier papers, but this was the first to incorporate the HTM techniques used by researchers in the robotics field. HTMs were assigned to the machine base and to each moving component, and HTM multiplications were used to develop expressions relating the position of the tool relative to the workpiece. A specific application of the technique on a two-axis turning center was presented.

Donmez followed that paper with another that described a general methodology for improving machine tool accuracy by compensating for geometric and thermal errors of the machine in real-time [11]. Three steps were used in developing this system. First, a general mathematical model was developed to calculate the vector error from a large number of component errors. Second, the

individual errors were decomposed into their geometric- and thermally-induced components for measurement and prediction. Third, a software system was created to compensate for the errors in real-time.

The compensation system was attached to the machine controller and injected an error signal into the position servo loop. This approach was taken because the developers did not have access to the workings of the control system's servo loop. An advantage of this approach was that the compensation calculations were performed externally to the machine tool control system. A disadvantage was that an additional computer and interfacing hardware were required. A generic functional diagram depicting this type of external interface is shown in Figure 2-1. This type of external interface was required by most all the researchers who actually implemented and tested their systems.

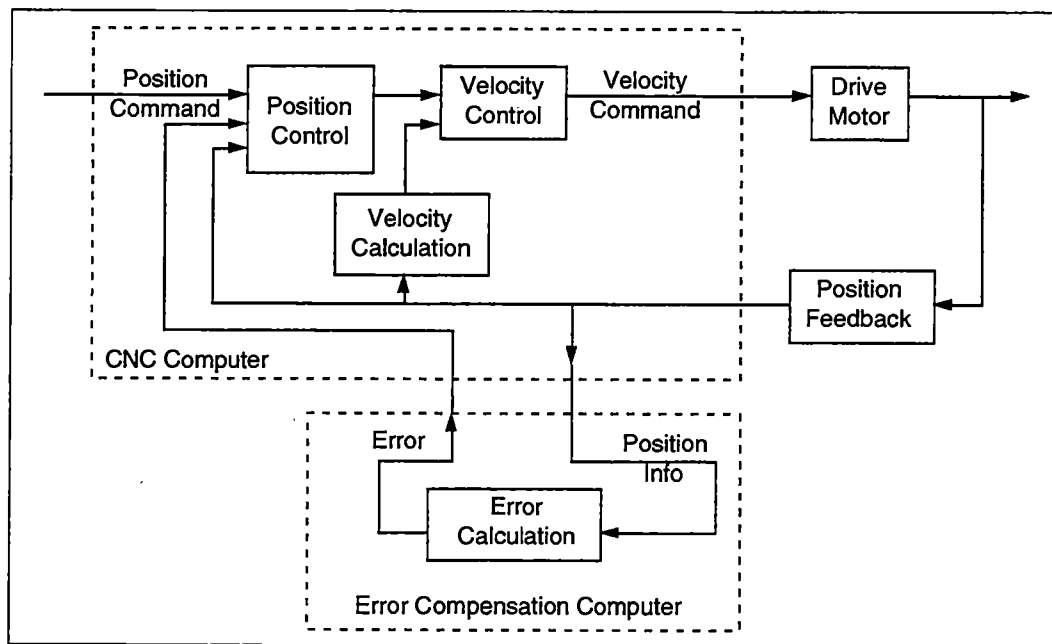


Figure 2-1. External error compensation system block diagram

Other papers reviewed shared the same approach to model machine tool errors. Zhang [12] used matrix manipulations to model the errors of a coordinate measuring machine. A 3-D error map of the machine was created by measuring along several lines in both the X and Y directions. Their mathematical model also incorporated temperature compensation. Sartori [13] discussed differences in compensating CMMs versus machine tools and noted that while machine tool error compensation must actually reposition the machine slides in real-time, in CMM operations, the compensation can be applied to the measured data, with no actual slide position modifications being needed. Eman [14] also developed matrix expressions for the geometric errors and discussed the effects of first- and second-order errors. He stated that in most machines the second-order effects could be neglected. However, for high precision machines and for some large machines, the second-order effects may attain significant levels and must be incorporated into the compensation system.

In most of the papers reviewed, the authors made several assumptions about this error compensation technique. The first was the assumption of rigid body motion. That is, each moving component of the machine was assumed to have no flexure as it traveled. Second, the principle of superposition was assumed. That is, errors due to position, direction, speed, temperature, load, etc., were measured separately and their effects were assumed to be independent of each other. Also, the errors of one slide were assumed to be independent of the positions of the other slides. Another assumption typically made was that angular error motions were of small magnitude, which implied that the mathematical order of application was unimportant. Second and higher-order error effects were also assumed to be negligible, greatly simplifying the resulting error expressions. Finally, the error data was assumed to be sufficiently smooth such that it could be described using piecewise continuous functions.

The Precision Engineering Laboratory report to NIST [3] was very thorough in its review of the evolution of the field of precision machines and contained an extensive bibliography related to the area of machine tool software correction. The paper discussed the various types of machine tool errors; namely geometric, thermal, and load-induced. It stated that three items were needed to implement software error correction, including (1) a mathematical model of the machine, (2) a methodology of measurement to supply the data for the model, and (3) a methodology for implementing the model and data into the controller. Various types of machine models were described, and some of the assumptions used in the models were given, most of which were included earlier in this section. The authors felt that compensation techniques for load-induced errors were not nearly as advanced as for geometric and thermal errors. Measurement techniques described were either parametric in nature, in that errors were measured as functions of axis position; or were derived using artifact standards, whereby certified artifacts such as spheres or grid plates were used to directly obtain the difference between nominal machine position and actual tool location.

Four correction methodologies were discussed. First mentioned was tape shimming, a technique whereby the programmed tool path is modified based on the machine model. Second was data correction after the fact, a technique useful only on measuring machines. Third was real-time correction needed for machining operations, usually implemented by severing the feedback loop of the control system and adding or subtracting pulses to the signal. Finally, mention was made of efforts to include the mathematical model in the control program itself, which the authors referred to as the "algorithmic correction method." A detailed historical summary of the published literature was also given. For anyone interested in the field of precision machines, this would be an excellent resource to review.

Chapter 3

OpenCNC Controller Description

The purpose of this chapter is to provide a brief description of the MDSI OpenCNC control system and the retrofit performed on the Excelllo 921 test-bed lathe. Detailed information regarding this retrofit is beyond the scope of this thesis; the interested reader is referred to Franks' thesis [15] documenting the retrofit of a three-axis Bridgeport® milling machine.

3.1 OpenCNC System Overview

A machine tool controller is a system of hardware and software that converts the desired motions in the form of a NC program into electrical signals that drive motors, causing movement of axis slides or spindles. In typical controllers, proprietary hardware or motion control cards are used to perform the calculations necessary to control the machine tool. These calculations are usually performed on dedicated processors, while other functions, such as instructions from the operator, utilize additional processors so that machine control functions are not interfered with. MDSI's OpenCNC system is unique in that no

proprietary hardware is utilized; all control functions are performed on a single Pentium CPU.

3.1.1 Hardware

A standard PC is typically used to control a machine tool under the OpenCNC controller. The PC is outfitted with some commercially available hardware components to provide the needed communication capabilities. One of these components is an encoder Industry Pack (Encoder-IP) that handles signals from all the machine tool's position feedback encoders. The other is a digital-to-analog converter Industry Pack (IP-DAC), which provides the analog signals needed by the slides' motor amplifiers. These two Industry Packs reside on a Technology 80® Model 5641 carrier card that has been inserted into one of the PC's available expansion slots. In addition to housing the two Industry Packs, the carrier card also has several digital input/output (I/O) pins. These pins are wired to racks of optically isolated I/O modules. These I/O modules are then wired to the appropriate relays within the control cabinet. The isolation modules protect the sensitive carrier board from the possibility of damage from current overloads in the control cabinet. A functional hardware diagram of the control system is shown in Figure 3-1. The control computer is depicted as comprising four functional components: the CPU and memory, the digital I/O signals, the Industry Packs, and the network card.

The CPU provides overall control of the other subsystems as well as directly interfacing with the operator control panel. It takes in information from the operator and displays the current status of the machine's environment. Since the typical OpenCNC operator panel consists only of a touchscreen computer monitor, all interface communications are performed using only software. The monitor communications are transmitted over a serial line to the PC.

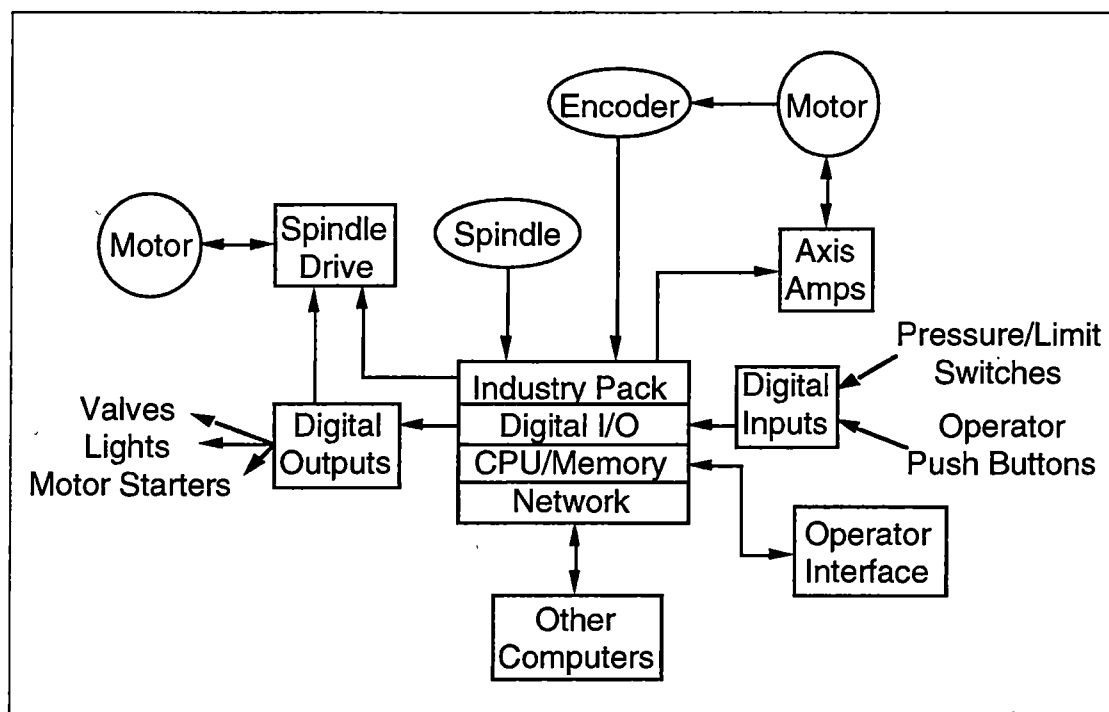


Figure 3-1. OpenCNC functional hardware diagram

The digital input lines are used to inform the OpenCNC control system of the states of various components on the machine. Examples of these inputs include axis limit switches, pressure switches, alert signals, or any operator push buttons or input devices not provided for by the touchscreen. These signals are read through the carrier card with software. The digital output lines are used to control various components on the machine tool such as solenoids, lights, and small motors. Each output line controls only one device. Again, these output lines are modified using software commands and interface to the rest of the machine tool control hardware through relays controlled by the carrier card.

The communications through the Industry Packs take priority over all other system communications because these modules control the actual motion of the slides and the spindle. The IP-DAC sends an analog signal to the amplifiers of each axis motor to provide the desired motion. Output signals from the axis

encoders are received as input by the Encoder-IP to provide position feedback information to OpenCNC.

3.1.2 Software

OpenCNC consists of several distinct executable processes that communicate with each other using shared memory. The processes are hierarchical both in CPU priority and in their relative timescales. Figure 3-2 shows a functional software architecture diagram of the OpenCNC system. The controller system is contained within the dashed box of the diagram. It is shown as comprising three distinct levels of processing. At the lowest level are the processes that interact with the environment of the machine tool hardware itself. The servo loop control software resides at this level along with the discrete logic and message-passing functions. These functions operate during each servo loop update. A user-definable parameter, the default servo loop update rate is 512 Hz. In order to provide real-time geometric error compensation, the servo loop software during each update reads the compensation values.

A requirement of any software systems that interface with others is that they have a defined protocol for information exchange. In a position compensation such as the one developed in this project, the external process must be able to use or convert the internal units of measurement used by OpenCNC. For linear distances, OpenCNC uses small units of measurement that it refers to as *distance units*. Each distance unit is equivalent to 0.00001 mm, so one inch of axis travel equates to 2,540,000 distance units. The machine's operator display shows the expected units of inches or millimeters, however.

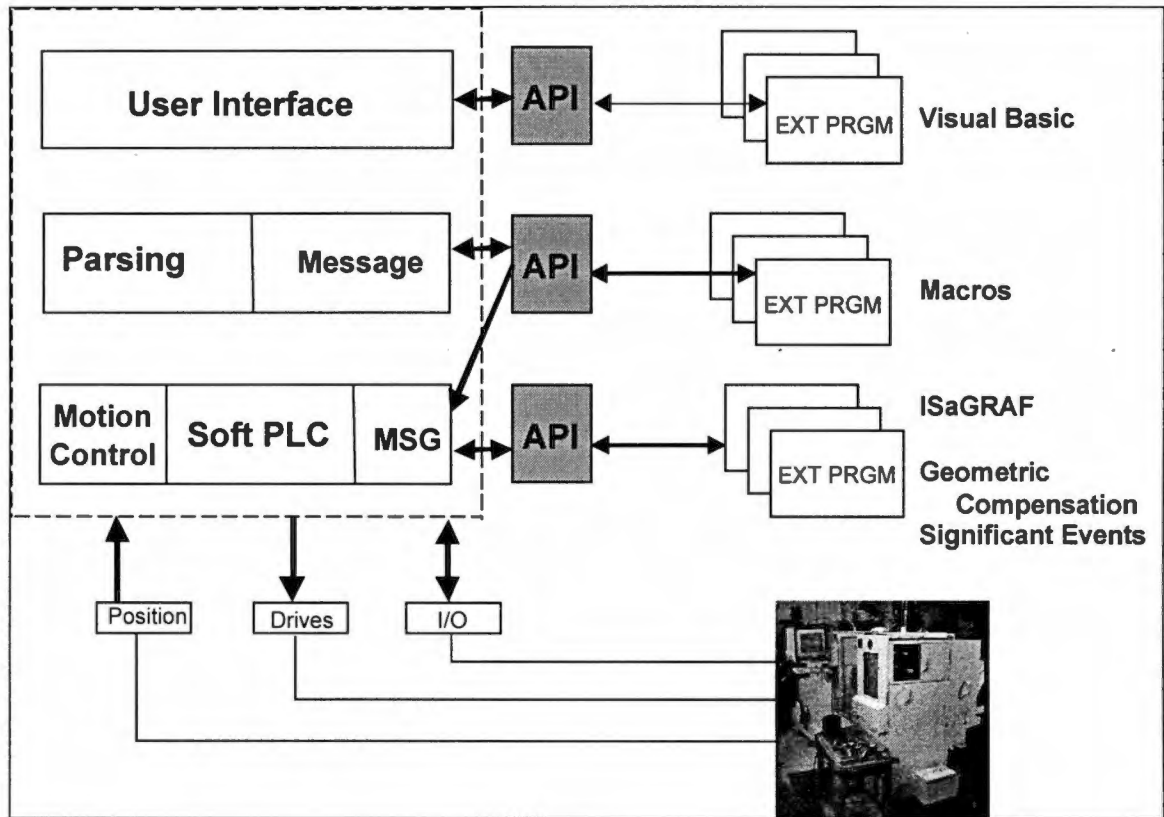


Figure 3-2. OpenCNC functional software architecture

Angular units in OpenCNC are treated differently. Since angular units are only needed for axes that are programmed in degrees, OpenCNC uses a ratio of two internal variables, *axDistParseOut* and *axDistParseIn*, to convert angles to linear units. This allows OpenCNC to treat the rotary axis as a linear axis. The conversion is carried out as

$$1 \text{ degree} = 10,000 \text{ distance units} \times \frac{\text{axDistParseOut}}{\text{axDistParseIn}}$$

and is only used for feedrate calculations. For instance, if a machine had a linear slide with a travel of 10 inches along with a rotary table, the end-user might configure OpenCNC to equate 360 degrees of rotary table motion with 10 inches of linear motion. To do so, he or she would set

axDistParseIn = 3,600,000 (360 degrees in distance units)
axDistParseOut = 254,000,000 (10 inches in distance units)

This would cause a movement of one complete revolution of the rotary table in 60 seconds if a linear move of 360 inches at a feedrate of 10 inches per minute (ipm) were commanded in a part program.

The discrete logic software is developed using a software package called ISaGRAF®. This package performs the same functions in software that were previously provided by programmable logic controllers (PLCs). It is a software development tool for industrial automated controls applications and was designed to allow inexpensive PCs to act as PLCs. It provides support for all five PLC languages specified in the International Electrotechnical Commission (IEC) Standard 1131-3. The user can customize the PLC functionality of an OpenCNC-controlled machine tool by modification of the default ISaGRAF code provided by MDSI.

The parsing and message modules are at the next layer of the software architecture. These modules interact with the NC part program and convert the part program into commands for the lower level motion routines. At the highest layer in the architecture is the user interface module. This module provides the operator control panel functionality. Because its functions are not critical operations, the user interface software runs at a lower priority than the other processes. The typical update rate for the operator interface is approximately 10–20 Hz. Because it is a safety system, the Emergency Stop (E-Stop) mechanism is wired externally to the controller and is not included as part of the user interface.

Also shown in Figure 3-2 is the OpenCNC Applications Programming Interface (API). This interface is a set of software and supporting documentation that allows users to interface external software with OpenCNC. The user

interface, for example, can be modified using Microsoft's Visual Basic® to call API functions. This API allows applications, such as the error compensation system developed in this project, access to the internal data structures of the OpenCNC system.

3.2 Test Bed Retrofit

During the fall of 1997, the GE2000 controller was removed from the Excello 921 and replaced with the hardware specified by MDSI for use with OpenCNC. Y-12 personnel performed electrical modifications, but MDSI personnel developed the ISaGRAF soft logic for this machine. As with all OpenCNC retrofits, several steps were followed in converting the lathe's control system. Using the existing Excello electrical schematic drawings, all input, output, and emergency-stop signals were identified. The machine's lead screw and encoder specifications were reviewed and recorded for later use during machine setup. Other machine upgrades not necessarily required by OpenCNC were performed. For the Excello, this included replacing the spindle and axis drive amplifiers. Components of the old GE control system no longer needed were identified, and the new layout of the control cabinet was created. The updated control cabinet is shown in Figure 3-3. The new PC and interface cards were installed, and all wiring connections were completed. At that time the soft logic modifications were made, and the system was ready for initial setup. This involved input of the machine's specific parameters such as nominal lead-screw pitch, encoder resolution, length of travel, and axis nomenclature. Additionally, the motor amplifiers were tuned to provide satisfactory performance using OpenCNC tuning software. After a period of a few weeks, the lathe was capable of running a NC program again.

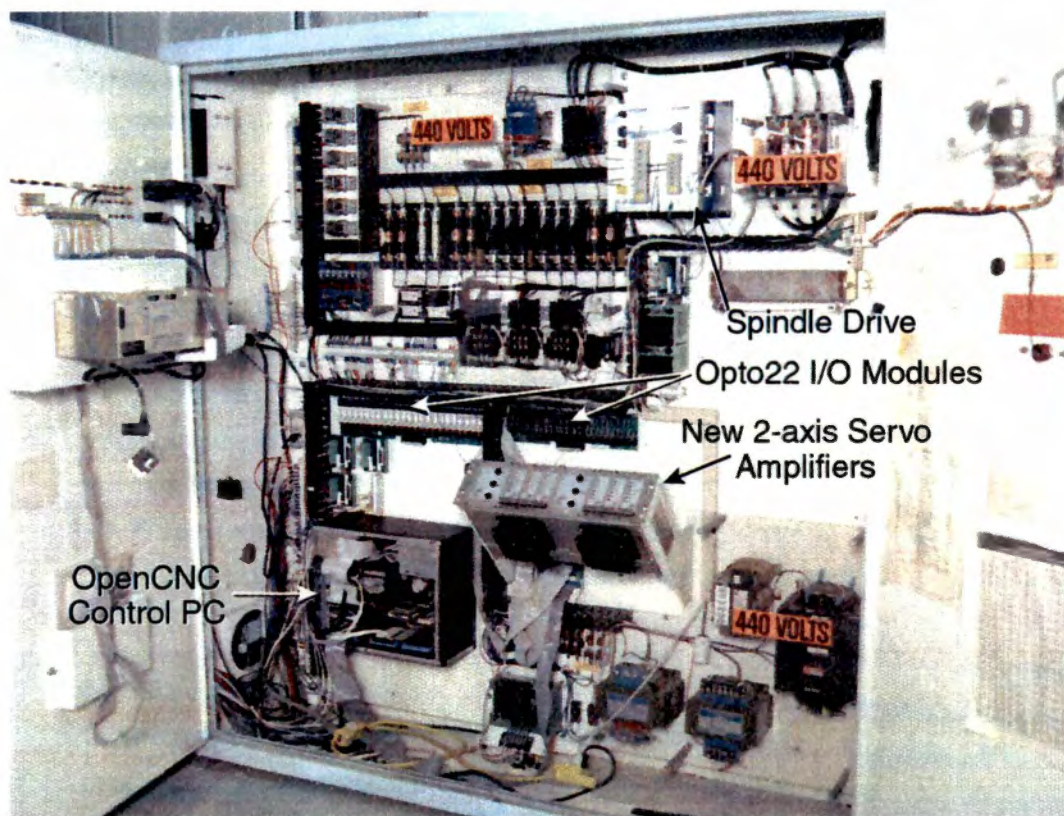


Figure 3-3. Excello control cabinet after OpenCNC retrofit

The GE2000 and its associated hardware were replaced with a single industrial PC with an interface card. Figure 3-4 shows a close-up view of the new control computer. This type of computer does not contain a typical motherboard as in most office PCs; instead, the PC resides on a single board that occupies a slot in the passive backplane of the chassis. At the time of the upgrade, the fastest available CPU was a Pentium with a clock speed of 200 MHz, which was more than adequate for running OpenCNC.

The Technology 80 5641 card can also be seen in Figure 3-4 along with the two Industry Packs. The ribbon cable shown coming out the bottom of the PC contains the digital I/O pins and runs to the Opto-22[®] modules mounted on

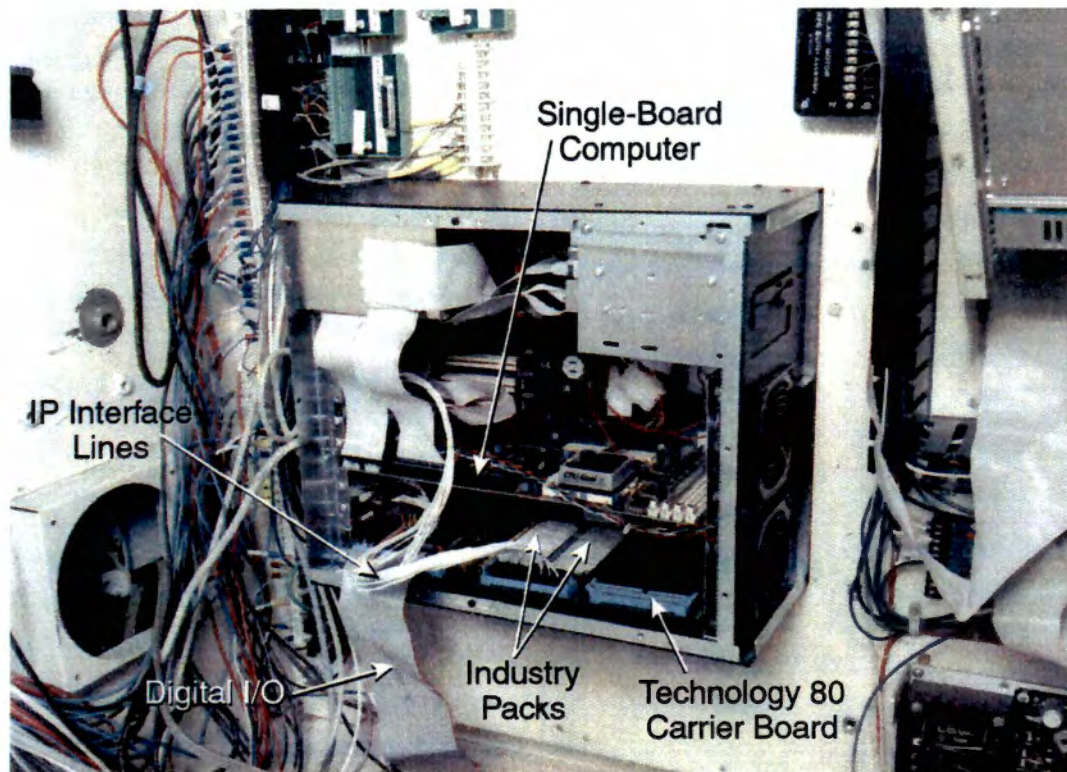


Figure 3-4. Industrial PC with control hardware

boards in the cabinet. The white wire bundles interface to the IP-DAC and Encoder-IP modules.

Figure 3-5 shows the touchscreen monitor that serves as the operator control panel. The only hardware buttons are the emergency-stop and reset buttons, and they are wired directly to the relays that control the machine's motors. Bypassing the software of the controller in this way is typical for machine tools and guarantees that motion will stop when the E-stop button is pushed. A digital input is triggered by the E-stop action, and that input notifies OpenCNC of the action.

All the functionality previously provided by hardware buttons on the GE operator interface has been replaced with the touchscreen interface utilized by

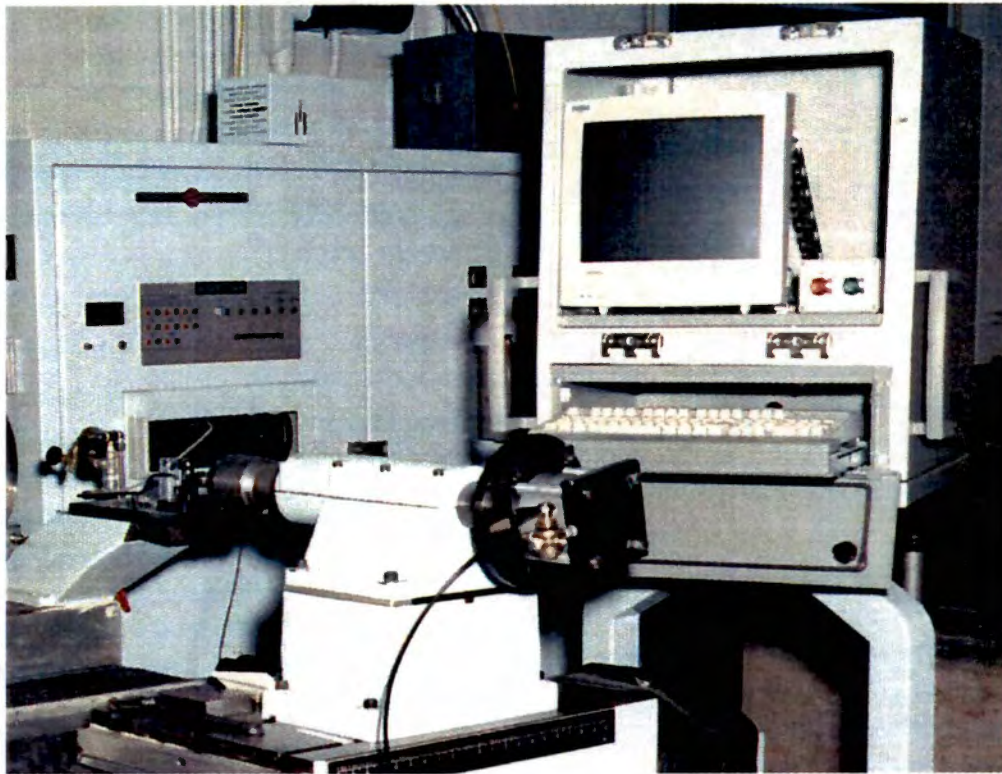


Figure 3-5. Operator control panel after MDSI retrofit

OpenCNC. An example of this interface in the Windows NT operating system is shown in Figure 3-6. It shows the control panel for jogging the axis, setting feedrates, operating the spindle, etc., in the left window. The smaller window on the right side of the screen contains the current position information, while the larger right-side window contains the instructions of a NC program. The lower window is used for operator messages.

This type of operator interface is used because it is much simpler to develop new functionality and modify the look of the interface through software than through hardware. It also provides a consistent interface from one machine type to another. That is, the operator interface for a 3-axis milling machine would not be much different from that for a 2-axis lathe. It remains to be seen whether this type of interface will become typical in the machine tool industry.

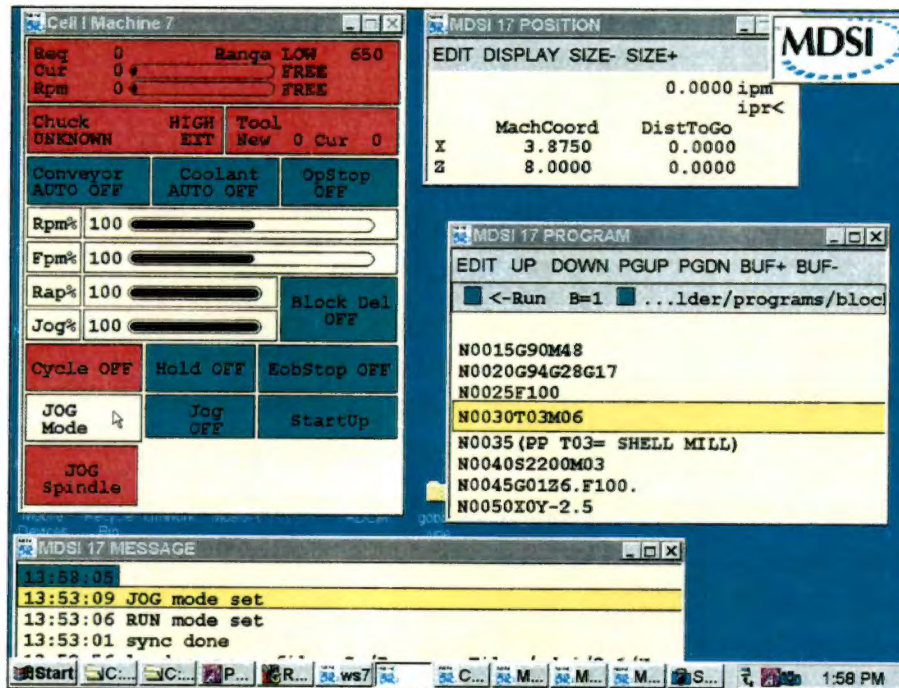


Figure 3-6. OpenCNC operator interface

Chapter 4

Error Compensation

Sartori discussed three methods of storing slide correction data in computers[13]. The first was to use an error lattice in which the error magnitudes were stored for a large number of points spread evenly throughout the work zone of the machine. Linear interpolation between the eight adjacent points was required when the tool location was not on one of the lattice points. The method was useful when "the error functions are not single variable functions, but strikes problems when it is necessary to compensate different tool (or probe) offsets." A disadvantage of this method was that it required an inordinate amount of measurements to obtain data for all the lattice points. A second method discussed was the use of error tables in which the errors were assumed to be functions of axis position only. The data in the tables were assumed to be piecewise-continuous, and linear interpolation was used between data points. This method had an advantage in that fewer measurements were required than in the lattice method. Finally, a third method discussed creating an error coefficient table. The error functions were modeled as analytical functions (e.g., Taylor polynomials, B-splines, etc.), and the coefficients of the polynomials were stored.

For this project, the error table method will be utilized. This method assumes rigid body motion, which implies that the error measurement locations of an axis are independent of the positions of the other axes. In order to develop the expressions for the compensation values that incorporate effects of all the geometric errors, it is necessary to review the mathematics of the axes' motions.

4.1 General Error Model

In its most basic form, error compensation involves calculating the amount of movement necessary to reposition one point in 3-D space (the position of the machine tool tip) onto another point in space (the cutting location on the workpiece) relative to some fixed reference coordinate system (also referred to as a reference frame). Machine tools comprise two or more stacked slides or spindles in various geometric configurations as shown previously in Figure 1-1. The relative motion between successive slides determines the position of both the workpiece and the cutting tool. This is very much similar to the situation seen in serial-linked robots, except those robots typically use axes of rotation more than translating slides. Nonetheless, the application of the techniques used by developers of robots to calculate end-effector positions are directly applicable in this situation.

4.1.1 Homogeneous Transformation Matrices

The most commonly used tool for describing the relative position between two components is the Homogeneous Transformation Matrix (HTM), which utilizes mathematical matrix notation to quantify the relationship between two coordinate systems. Figure 4-1 depicts two reference frames A and B:

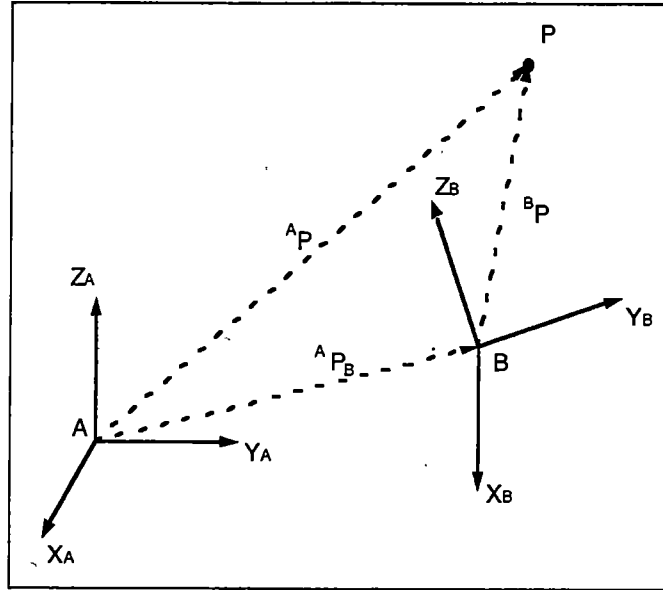


Figure 4-1. A point in one frame relative to another frame

Frame B contains a point P whose location relative to B is given by ${}^B P$. Its location relative to frame A is given by the following equation:

$${}^A P = {}^A R_B {}^B P + {}^A P_B \quad (4-1)$$

Equation (4-1) says that the location of point P relative to A equals the location of P relative to B premultiplied by a rotation matrix R plus the location of the origin of frame B relative to frame A. In Equation (4-1), R is a 3×3 rotation matrix that mathematically describes the orientation of frame B relative to frame A. In matrix format, Equation (4-1) becomes

$$\begin{bmatrix} {}^A P_x \\ {}^A P_y \\ {}^A P_z \end{bmatrix} = \begin{bmatrix} {}^A R_{Bix} & {}^A R_{Biy} & {}^A R_{Biz} \\ {}^A R_{Bjx} & {}^A R_{Bjy} & {}^A R_{Bjz} \\ {}^A R_{Bkx} & {}^A R_{Bky} & {}^A R_{Bkz} \end{bmatrix} \begin{bmatrix} {}^B P_x \\ {}^B P_y \\ {}^B P_z \end{bmatrix} + \begin{bmatrix} {}^A P_{Bx} \\ {}^A P_{By} \\ {}^A P_{Bz} \end{bmatrix} \quad (4-2)$$

When multiple frames are involved, Equation (4-2) is applied in succession starting from the last coordinate frame and ending at the first frame in the series.

A single equation in matrix format describing multiple transformations quickly becomes complicated. When working with multiple coordinate systems, the homogeneous transformation matrix provides a more compact notation for performing the calculations in Equation(4-2). Slocum [16] gives a detailed description of HTM operations, which is summarized below. The HTM is a 4×4 matrix of the following form:

$${}^A\mathbf{T}_B = \begin{bmatrix} O_{ix} & O_{iy} & O_{iz} & P_x \\ O_{jx} & O_{jy} & O_{jz} & P_y \\ O_{kx} & O_{ky} & O_{kz} & P_z \\ \hline 0 & 0 & 0 & P_s \end{bmatrix} \quad (4-3)$$

In this notation the pre-superscript (A in this case) denotes the reference frame the result should be represented in, and the post-subscript (B in this case) denotes the reference frame being transferred from. The first three columns of the HTM represent the direction cosines of the X_B , Y_B , and Z_B axes, with respect to the frame A. The fourth column represents the coordinates of the origin of the frame B with respect to frame A. P_s is a scale factor that is typically set to one in these operations. The bottom row is added to make the matrix square, enabling series multiplication of HTMs. Thus, in HTM notation Equation(4-1) becomes

$${}^A\mathbf{P} = {}^A\mathbf{T}_B {}^B\mathbf{P} \quad (4-4)$$

Consider, for example, two initially coincident reference frames A and B. If frame B is translated by an amount u along its X-axis, by an amount v along its Y-axis, and an amount w along its Z-axis, then the HTM that will transform a point in frame B into frame A is given by

$${}^A\mathbf{T}_B = \left[\begin{array}{ccc|c} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-5)$$

If frame B is rotated by an amount θ_x about its X-axis, the HTM that will transform a point in frame B into frame A is given by

$${}^A\mathbf{T}_B = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-6)$$

If frame B is rotated by an amount θ_y about its Y-axis, the HTM that will transform a point in frame B into frame A is given by

$${}^A\mathbf{T}_B = \left[\begin{array}{ccc|c} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-7)$$

If frame B is rotated by an amount θ_z about its Z-axis, the HTM that will transform a point in frame B into frame A is given by

$${}^A\mathbf{T}_B = \left[\begin{array}{ccc|c} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-8)$$

Simultaneous combinations of these motions can be represented by multiplying the HTMs in series to obtain a single HTM describing the motion. However, care must be taken when multiple rotations or large angles are involved, because the order of rotation becomes important.

The benefit of the HTM notation becomes apparent when multiple coordinate systems are involved. In the case of a machine tool, HTMs describing

the relative location between successive components can represent the structural components that make it up. In modeling the machine one can start with a base reference frame and work up to the tool tip. If a machine tool is composed of N rigid bodies connected in series, the position of the tip (N th axis) relative to the base coordinate system will be the sequential product of all the HTMs:

$${}^R\mathbf{T}_N = \prod_{m=1}^N {}^{m-1}\mathbf{T}_m = {}^0\mathbf{T}_1 {}^1\mathbf{T}_2 {}^2\mathbf{T}_3 {}^3\mathbf{T}_4 \dots \quad (4-9)$$

4.1.2 Application of HTMs to Linear Axes

Consider an ideal linear machine tool carriage, as depicted in Figure 4-2. The carriage's coordinate system is ideally parallel to the reference coordinate system, and the axis of motion is along the X -axis of the coordinate system. As mentioned previously, since the slide is a moving rigid body, it has six error components of motion. Three are rotational (ϵ_x , ϵ_y , ϵ_z , defined as rotations about the reference coordinate system axes), and three are translational (δ_x , δ_y , δ_z , defined as translations along the reference coordinate system axes).

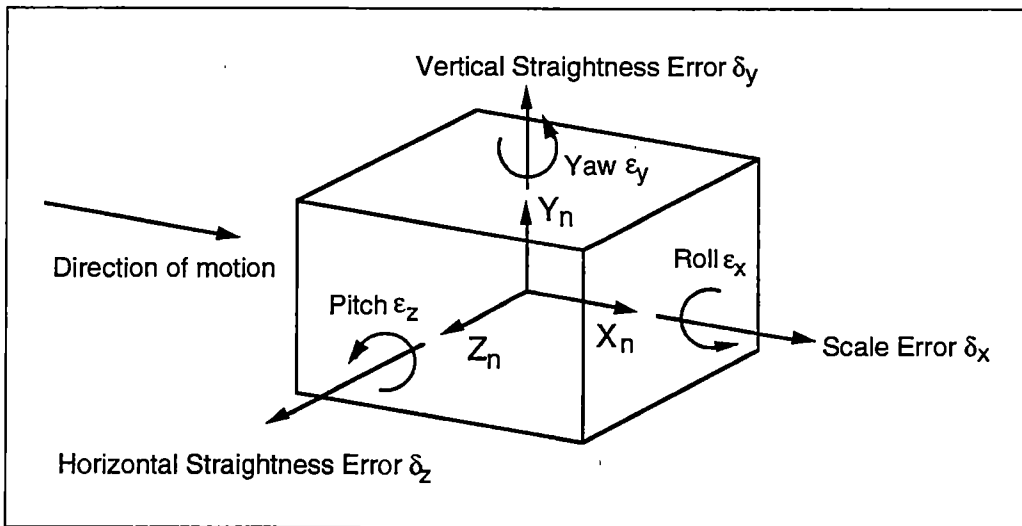


Figure 4-2. Error motions for a typical linear slide

As described in another paper by Donmez [17], to calculate the HTM that describes the error effects on the carriage, one can work with the rotational errors separately from the translation errors. The total rotational error, T_{rot} , of the error HTM can be found by multiplying Equations (4-6), (4-7), and (4-8) in series and making the substitutions ϵ_x , ϵ_y , and ϵ_z for θ_x , θ_y , and θ_z :

$$T_{rot} = \left[\begin{array}{ccc|c} \cos \epsilon_y \cos \epsilon_z & -\cos \epsilon_y \sin \epsilon_z & \sin \epsilon_y & 0 \\ \sin \epsilon_x \sin \epsilon_y \cos \epsilon_z + \cos \epsilon_x \sin \epsilon_z & -\sin \epsilon_x \sin \epsilon_y \sin \epsilon_z + \cos \epsilon_x \cos \epsilon_z & -\sin \epsilon_x \cos \epsilon_y & 0 \\ -\cos \epsilon_x \sin \epsilon_y \cos \epsilon_z + \sin \epsilon_x \sin \epsilon_z & \cos \epsilon_x \sin \epsilon_y \sin \epsilon_z + \sin \epsilon_x \cos \epsilon_z & \cos \epsilon_x \cos \epsilon_y & 0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-10)$$

Since small angular rotations (on the order of minutes of a degree at most) are being assumed, the approximations $\cos \epsilon \approx 1$ and $\sin \epsilon \approx \epsilon$ can be used. Additionally, since small quantities are being multiplied together, second- and higher-order terms can be safely neglected. With these simplifying assumptions, Equation (4-10) reduces to

$$T_{rot} = \left[\begin{array}{ccc|c} 1 & -\epsilon_z & \epsilon_y & 0 \\ \epsilon_z & 1 & -\epsilon_x & 0 \\ -\epsilon_y & \epsilon_x & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-11)$$

The total translational error, T_{trans} , of the error HTM is obtained from Equation (4-5) with error term substitutions δ_x , δ_y , and δ_z for u , v , and w .

$$T_{trans} = \left[\begin{array}{ccc|c} 1 & 0 & 0 & \delta_x \\ 0 & 1 & 0 & \delta_y \\ 0 & 0 & 1 & \delta_z \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-12)$$

Thus, the resultant HTM describing the position error of the linear carriage is

$$\mathbf{T}_{\text{err}} = \mathbf{T}_{\text{rot}} \mathbf{T}_{\text{trans}} = \left[\begin{array}{ccc|c} 1 & -\epsilon_z & \epsilon_y & \delta_x \\ \epsilon_z & 1 & -\epsilon_x & \delta_y \\ -\epsilon_y & \epsilon_x & 1 & \delta_z \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-13)$$

When multiple axes are involved it will be convenient to show the error terms as functions of the position of a particular axis. Since in this example the direction of motion is X, the error HTM becomes

$$\mathbf{T}_{\text{err}} = \left[\begin{array}{ccc|c} 1 & -\epsilon_z(x) & \epsilon_y(x) & \delta_x(x) \\ \epsilon_z(x) & 1 & -\epsilon_x(x) & \delta_y(x) \\ -\epsilon_y(x) & \epsilon_x(x) & 1 & \delta_z(x) \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-14)$$

At this point it would be beneficial to look in detail at each term of the HTM in Equation(4-14) to clarify the meaning of the error symbols as they relate to Figure 4-2. This information is summarized in Table 4-1.

Table 4-1. HTM error term descriptions

Term	Description
$\epsilon_x(x)$	Rotation about X-axis (roll) as a function of X position
$\epsilon_y(x)$	Rotation about Y-axis (yaw) as a function of X position
$\epsilon_z(x)$	Rotation about Z-axis (pitch) as a function of X position
$\delta_x(x)$	Deviation from ideal position in direction of motion (scale error) as a function of X position
$\delta_y(x)$	Deviation from ideal position in Y direction as a function of X position. Comes from two sources, vertical straightness error and squareness error
$\delta_z(x)$	Deviation from ideal position in Z direction as a function of X position. Comes from two sources, horizontal straightness error and squareness error

As shown in Table 4-1, the rotational errors have straightforward descriptions, whereas the translational errors are more complicated. Linear errors in directions perpendicular to the axis of motion must take into account any squareness errors between the two designated directions. So for the two orthogonal directions the total linear errors become

$$\begin{aligned}\delta_y &= \delta_y' + \alpha_{xy}X \\ \delta_z &= \delta_z' + \alpha_{xz}X\end{aligned}\tag{4-15}$$

where

δ_y'	vertical straightness error
δ_z'	horizontal straightness error
α_{xy}	squareness error of the Y-axis relative to the X-axis
α_{xz}	squareness error of the Z-axis relative to the X-axis

Each term in Equation (4-14) represents a piecewise-continuous function that quantifies the magnitude of that particular error at incremental locations along the X-axis. These functions would be stored as arrays of data within the control system. In addition, the errors of an axis may be dependent upon the direction of travel, so each axis might utilize two sets of error tables. To use these tables, the software would have to obtain both the axis position and direction of motion for each axis and retrieve the appropriate value from up to six different tables plus calculate deviations due to errors in squareness. Any tables not containing values would default to zero, so the user would still be able to use this new compensation system for lead-screw compensation only. In the case of this example, the values for lead-screw compensation would be stored in the $\delta_x(x)$ table.

As previously mentioned, compensating rotary axes of motion is not within the scope of this project. However, the same technique described above would be

used to develop the HTM for a rotary axis; the difference would be that in a rotary axis the assumption of small angular motions would not be valid. This would cause the subsequent rotation matrix expressions to become more complicated, but it is certainly an option for future implementations if needed.

4.1.3 Total Error Equation for a Machine Tool

As detailed by Slocum [16], an element of a machine tool (slide, tool, workpiece, base, turret, etc.) of a machine tool can be represented by a HTM. The HTM gives the position and orientation of that element relative to another element of the machine that has its own HTM assigned. Starting from the HTM of the machine base, one can create a series of HTMs that traverse the elements of the machine tool to either the cutting tool or to the workpiece. This gives the location of both the workpiece and the tool in the same base reference system, as shown in Figure 4-3.

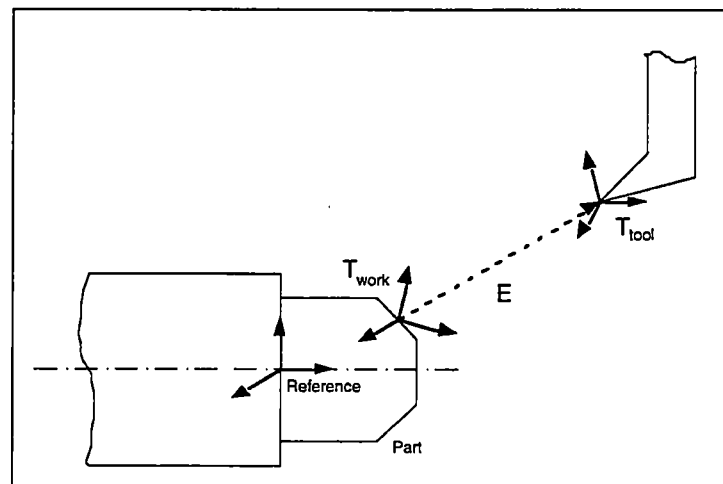


Figure 4-3. Relationship between cutting tool and workpiece

In the ideal case, the HTM of the workpiece relative to some reference frame R , ${}^R T_{\text{work}}$, and the HTM of the tool, ${}^R T_{\text{tool}}$, would coincide. However, in the actual cutting case, due to the inaccuracies of the system, the two HTMs are not identical. To make ${}^R T_{\text{tool}}$ coincide with ${}^R T_{\text{work}}$ one can use another HTM, referred to as E_{rel} , which quantifies the relative error between the tool and workpiece, as shown by the following formula:

$${}^R T_{\text{tool}} = {}^R T_{\text{work}} E_{\text{rel}} \quad (4-16)$$

The homogeneous transformation matrix E_{rel} , like all HTMs, is of the form of Equation (4-3), and is given by

$$E_{\text{rel}} = {}^R T_{\text{work}}^{-1} {}^R T_{\text{tool}} \quad (4-17)$$

where ${}^R T_{\text{work}}^{-1}$ represents the inverse of the homogeneous transformation matrix.

The relative error HTM E_{rel} represents the position and orientation of the tool relative to the workpiece. From another perspective, E_{rel} quantifies the changes in position and orientation required to cause ${}^R T_{\text{tool}}$ to coincide with ${}^R T_{\text{work}}$. Because the contact between the tool and the workpiece is assumed to be point contact, the orientation of the tool, represented by the rotation matrix in E_{rel} , is of no concern. The position vector P in E_{rel} represents the translations relative to the tool's coordinate frame that must be made to the tool so that it is properly located on the workpiece.

An example of the expressions that result from this type of analysis is given by the two-axis turning center presented in Donmez' paper [10]. The turning center structure consists of a spindle, a carriage, a cross slide, a tool turret, and a tool. HTMs were assigned to each of these elements in addition to the machine base reference system and to the workpiece as shown in Figure 4-4.

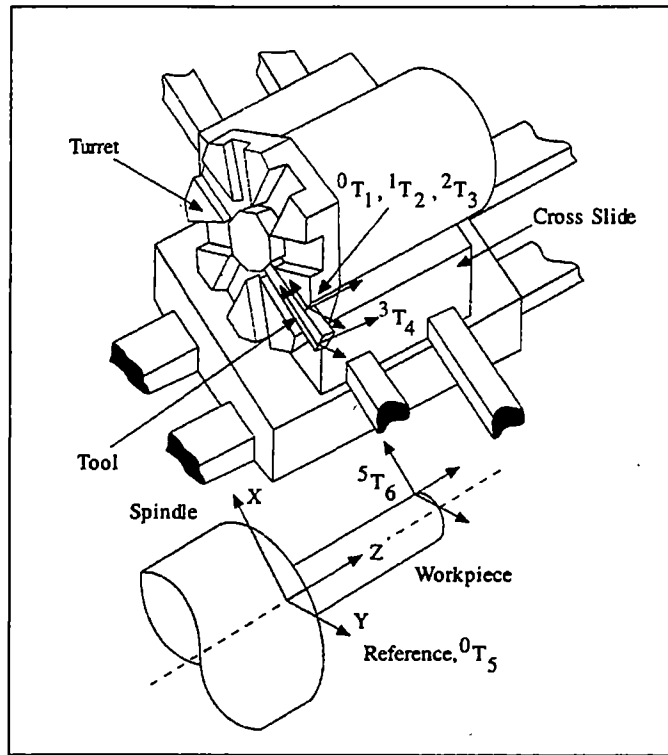


Figure 4-4. HTM assignments on a two-axis turning center (*used by permission*)

Table 4-2 shows how the HTMs were assigned to the turning center elements. The base reference system was chosen to be located at the base of the spindle.

Table 4-2. Turning center HTM assignments

HTM	Description
${}^0T_1$	Carriage frame with respect to reference frame
${}^1T_2$	Cross slide frame with respect to carriage
${}^2T_3$	Tool turret frame with respect to cross slide
${}^3T_4$	Cutting tool frame with respect to tool turret
${}^0T_5$	Spindle frame with respect to reference frame
${}^5T_6$	Workpiece frame with respect to spindle

In developing the error equations, the HTMs describing the position and orientation of the tool and the workpiece were calculated as

$$\begin{aligned} \mathbf{T}_{\text{tool}} &= {}^0\mathbf{T}_1 {}^1\mathbf{T}_2 {}^2\mathbf{T}_3 {}^3\mathbf{T}_4 = {}^0\mathbf{T}_4 \\ \mathbf{T}_{\text{work}} &= {}^0\mathbf{T}_5 {}^5\mathbf{T}_6 = {}^0\mathbf{T}_6 \end{aligned} \quad (4-18)$$

Each transformation matrix in Equation(4-18) was developed using the techniques described earlier for a linear axis. When multiplying the matrices, second order effects were neglected, which greatly simplified the expressions. The final expressions for the error in each direction were given as

$$\begin{aligned} P_{E_x} = & -x + x(w) - X_1 - X_4 - \delta_x(c) - \delta_x(s) - \delta_x(t) - \delta_x(w) - \delta_x(x) - \delta_x(z) \\ & + \varepsilon_y(s) * z(w) - Z_4 [\varepsilon_y(x) + \varepsilon_y(z) + \varepsilon_y(t)] \end{aligned} \quad (4-19)$$

$$\begin{aligned} P_{E_y} = & -\delta_y(c) - \delta_y(s) - \delta_y(t) - \delta_y(w) - \delta_y(x) - \delta_y(z) - \varepsilon_x(s) * z(w) - x * \varepsilon_z(z) \\ & - X_4 [\varepsilon_z(x) + \varepsilon_z(z) + \varepsilon_z(t)] + Z_4 [\varepsilon_x(x) + \varepsilon_x(z) + \varepsilon_x(t)] \end{aligned} \quad (4-20)$$

$$\begin{aligned} P_{E_z} = & -z + z(w) - Z_4 - \delta_z(c) + \delta_z(s) - \delta_z(t) + \delta_z(w) - \delta_z(x) - \delta_z(z) + x * \varepsilon_y(z) \\ & - \varepsilon_y(s) * x(w) + X_4 [\varepsilon_y(x) + \varepsilon_y(z) + \varepsilon_y(t)] \end{aligned} \quad (4-21)$$

In Equations (4-19) through (4-21), x and z are the axis positions; $x(w)$ and $z(w)$ are the cutting location coordinates on the workpiece (functions of the part program); X_1 , X_4 , Z_1 , Z_4 are constants representing the linear offsets between the HTMs; and the ε and δ terms represent the various rotational and linear error terms described in Equation(4-14). Since this example was for a two-axis turning center there could be no Y-axis motion, so errors in that direction could not be compensated.

As seen in Equations (4-19) through (4-21), the error expressions developed are quite complex and would be difficult to derive without the methodology presented in this section. Several points about this development are worth noting:

1. The expressions take into account all geometric errors of the machine tool, including scale error; horizontal and vertical straightness errors; and roll, pitch, and yaw errors.
2. The derivation is dependent upon the placement and orientation of the HTMs. By changing location and orientation of the HTMs, the expressions would change, but the final numeric values would not, assuming that the world reference system remained constant.
3. A new error model must be developed for each specific machine type. That's not to say each specific machine, however. If two different milling machines had equivalent axis configurations, then the same error model would apply. Any geometric constants used in the models would have different values, however.

4.2 Excello Error Model

Before describing the development of the error model used in compensating the test-bed lathe, some general comments are in order. The location and orientation of the coordinate systems used in the model are somewhat arbitrary and are chosen by the developer for convenience. It is usually beneficial to use coordinate systems that are oriented to match the coordinate system used in programming the machine tool, since this makes process of converting from model space to machine space and vice versa easier. Another consideration in selecting the locations of the coordinate systems is the ease of obtaining error data in that coordinate system.

The machine model used in the development of the error equations is shown in Figure 4-5, which depicts a plan view of the lathe. All the coordinate systems shown are at spindle center height.

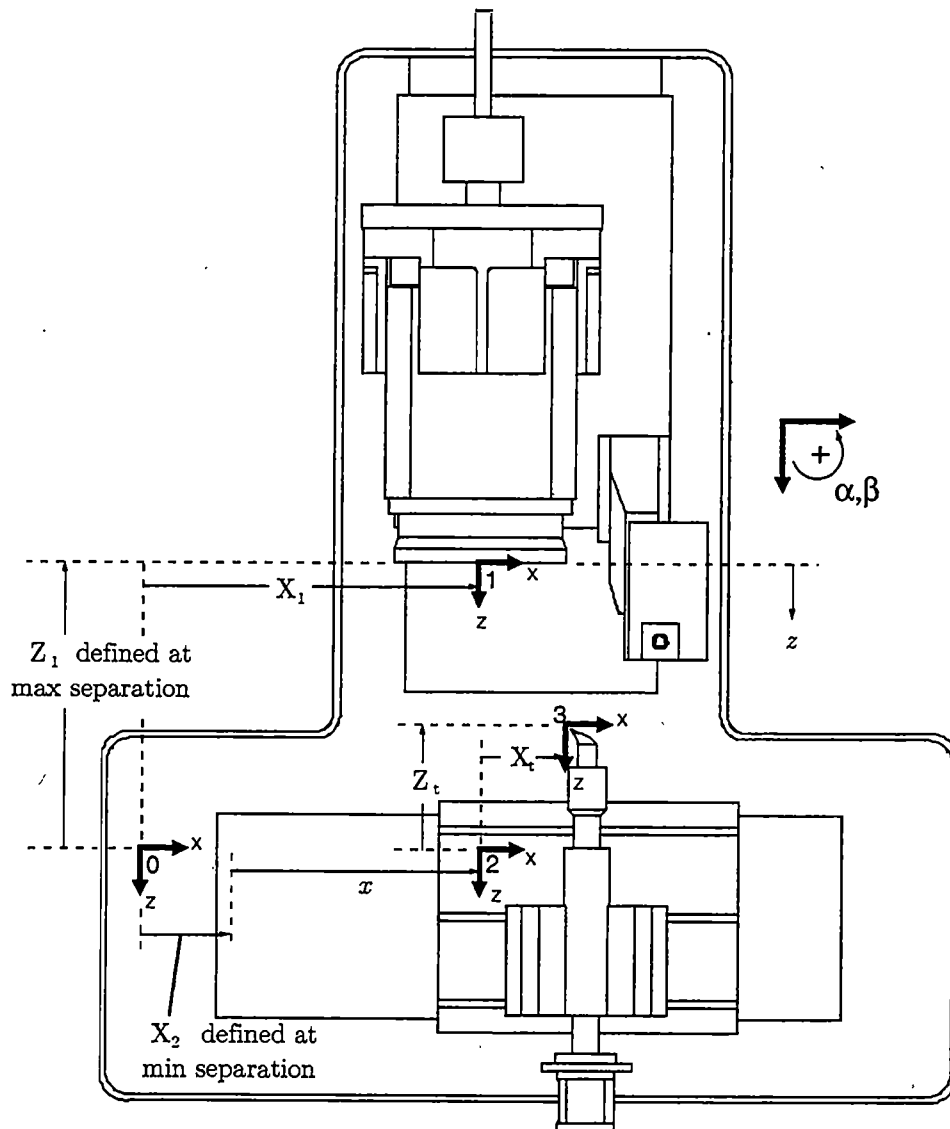


Figure 4-5. Excello error model diagram

Figure 4-5 shows the location of four HTMs or coordinate systems used in the error model derivation. The HTMs are aligned so that the positive direction of the coordinate axes corresponds to positive displacement of the machine slides. In this orientation the Y-axis of each HTM is vertically upward. This direction matches the direction of positive vertical straightness error given by the laser

interferometric measuring system used to characterize the machine tool's geometric errors. The direction of positive rotation is also given in Figure 4-5, which will become important for compensation of angular errors. In Figure 4-5 and in the error equation development, uppercase Roman letters designate constant terms while lowercase Roman letters designate position variables. Greek letters designate geometric errors.

HTM0 is the base reference system used in the model, and its location does not change. Its location corresponds to the end of the laser head. The laser is mounted on an aluminum bracket attached to the machine structure. HTM 1 is attached to the spindle face and moves with the Z-slide. Z_1 is defined as the Z distance between HTM0 and HTM1 when the Z-slide is positioned at its most negative limit switch. Relative to HTM0, Z_1 is a negative quantity. The position variable z is defined from the Z_1 location. X_1 is the constant X offset between HTM0 and HTM1. HTM2 is attached to the table of the X-slide and moves with it. The X-axes of HTM0 and HTM2 align, so there is no Z offset between them. X_2 is defined as the positive X offset between HTM0 and HTM2 when the X-slide is positioned at its most negative limit switch. The position variable x is defined from the X_2 location.

HTM3 is defined at the tip of the cutting tool with constant offsets from HTM2 of X_t (a positive quantity as shown in Figure 4-5) and Z_t (a negative quantity as shown). These offsets would be different for different cutting tools. The tool-set station is used to precisely determine these offsets, and would be stored in the controller for access by the compensation system. If multiple tool set operations were performed during machining of a particular part, the effects of tool wear could be compensated. However, since the tool-set station functionality was not incorporated into OpenCNC at the time of this development, these offsets will be stored in the error data file.

The error equations for a given machine tool are created using Equation (4-17). Before applying this equation to the Excella lathe, the HTMs for ${}^R T_{\text{work}}$ and ${}^R T_{\text{tool}}$ must be developed. To do that, formulas for the individual error motions of the X and Z-slides must be created, as described in Section 4.1.2. Similar in form to Equation (4-14), the motion of the Z-axis (HTM1) relative to the reference coordinate system (HTM0) is given by

$${}^0 T_1 = \left[\begin{array}{ccc|c} 1 & 0 & \epsilon_y(z) & X_1 + \delta_x(z) + \beta \cdot z \\ 0 & 1 & 0 & 0 \\ -\epsilon_y(z) & 0 & 1 & Z_1 + z + \delta_z(z) \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-22)$$

where

$\epsilon_y(z)$	yaw error of Z-slide as function of z position
X_1, Z_1	constant offsets between HTM0 and HTM1
z	position of Z-slide
$\delta_x(z)$	straightness error in X direction as function of z position
$\delta_z(z)$	displacement error in Z direction as function of z position
β	parallelism error between Z-slide and machine spindle

In Equation (4-22) the only rotational error terms are those that correspond to rotations about the Y-axis. Rotations about the X- or Z-axes cause errors in the vertical direction, which cannot be compensated by a two-axis machine tool so these terms are left out to simplify the derivation. Similarly, no term is given for translation errors in the Y direction, which physically corresponds to vertical straightness errors in this case. The expression giving the X location of HTM1 relative to HTM0 comes from the straightness and parallelism errors of the slide. Algebraic signs for these terms are assigned by determining the effect of an error in the positive direction. For instance, using the direction of positive rotational errors in Figure 4-5, when β is positive, measured from the positive Z-axis, an

error in the positive X direction is generated. The Z location of HTM1 relative to HTM0 comes from the nominal position and error in the direction of motion.

In a similar fashion, the HTM describing the position of HTM2 with respect to HTM0 is given by

$${}^0T_2 = \left[\begin{array}{ccc|c} 1 & 0 & \varepsilon_y(x) & X_2 + x + \delta_x(x) \\ 0 & 1 & 0 & 0 \\ -\varepsilon_y(x) & 0 & 1 & \delta_z(x) - \alpha \cdot x \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-23)$$

where

$\varepsilon_y(x)$	yaw error of X-slide as function of x position
X_2	constant offset between HTM0 and HTM2
x	position of X-slide relative to X_2
$\delta_x(x)$	displacement error in X direction as function of x position
$\delta_z(x)$	straightness error in Z direction as function of x position
α	perpendicularity error between X-slide and machine spindle

The rotational errors are again determined solely from the yaw error of the X-slide. The nominal position and the error in the direction of motion give the position of HTM2 relative to HTM0 in the X direction. In the Z direction, the position of HTM2 is given by the straightness and perpendicularity errors of the slide. Positive perpendicularity error, as measured from the positive X-axis, causes a displacement in the negative Z direction, so the sign on that term is negative.

Finally, the homogeneous transformation matrix describing the location of HTM3 relative to HTM2 is given by

$${}^2T_3 = \left[\begin{array}{ccc|c} 1 & 0 & 0 & X_t \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & Z_t \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (4-24)$$

where X_t and Z_t are constant offsets locating the tool tip with respect to HTM2. Since the location and orientation of HTM3 relative to HTM2 remains fixed, no error terms are involved in this expression. It is possible to include error terms for tool wear, incorrect insertion of the tool into the holder, etc., but in this derivation these terms are assumed to be negligible.

Now that the motions of the individual HTMs have been defined, expressions can be calculated for the tool and work HTMs:

$$\begin{aligned} {}^R\mathbf{T}_{\text{work}} &= {}^0\mathbf{T}_1 \\ {}^R\mathbf{T}_{\text{tool}} &= {}^0\mathbf{T}_3 = {}^0\mathbf{T}_2 {}^2\mathbf{T}_3 \end{aligned} \quad (4-25)$$

The complete error expression for the Excello lathe can now be calculated using Equation (4-17). The final HTM is given by

$$\mathbf{E}_{\text{rel}} = {}^R\mathbf{T}_{\text{work}}^{-1} {}^R\mathbf{T}_{\text{tool}} = {}^0\mathbf{T}_1^{-1} {}^0\mathbf{T}_3 \quad (4-26)$$

The specific calculations used in Equation (4-26) were performed using the algebraic software package Mathematica[®] because of the complexity of the matrix calculations involved. The matrix multiplications were carried out with the assumption that second- and higher-order error terms were negligible. Details are given in Appendix A. As in the lathe example described in Section 4.1.3, the only quantities of interest in the $\mathbf{E}_{\text{rel}}$ HTM resulting from Equation (4-26) are the translational terms since they describe the linear compensations required to bring the tool tip onto the workpiece. The final error equations are calculated as

$$P_x = x - X_1 + X_2 + X_t + \delta_x(x) - \delta_x(z) + Z_t \cdot \varepsilon_y(x) + (z + Z_1 - Z_t) \cdot \varepsilon_y(z) - \beta \cdot z \quad (4-27)$$

$$P_z = -z - Z_1 + Z_t + \delta_z(x) - \delta_z(z) - X_t \cdot \varepsilon_y(x) + (x - X_1 + X_2 + X_t) \cdot \varepsilon_y(z) - \alpha \cdot x \quad (4-28)$$

Even for this simple two-axis machine, the error model is complicated. For machines where the highest levels of accuracy are needed, some of the simplifying

assumptions may not be justified, and the resulting expressions would grow increasingly more complex.

Equations (4-27) and (4-28) contain expressions describing the position of the tool tip relative to the workpiece and include the positioning errors of all moving components between them. The equations comprise several terms, but these terms can be grouped as

$$P_x = \text{Nominal X Position} + \text{X Offsets} + \Delta P_x \quad (4-29)$$

$$P_z = \text{Nominal Z Position} + \text{Z Offsets} + \Delta P_z \quad (4-30)$$

When implementing a software compensation system, the terms of interest are those pertaining to the error motions of the slides, which are ΔP_x and ΔP_z in Equations (4-29) and (4-30). With this in mind, the equations used in the actual computer code are given as

$$\Delta P_x = \delta_x(x) - \delta_x(z) + Z_t \cdot \epsilon_y(x) + (z + Z_1 - Z_t) \cdot \epsilon_y(z) - \beta \cdot z \quad (4-31)$$

$$\Delta P_z = -\delta_z(z) + \delta_z(x) - X_t \cdot \epsilon_y(x) + (x - X_1 + X_2 + X_t) \cdot \epsilon_y(z) - \alpha \cdot x \quad (4-32)$$

4.3 Excello Error Characterization

As described in Chapter 2, software error compensation requires a mathematical model of the machine tool and a method of measuring the error quantities represented in the model. Measurement of the motion errors must also be a consideration in the design of the system architecture. While the quantities of interest in the error model equations have specific geometric meanings, they are independent of the method used to measure them. Thus, the metrologist should not be restricted to using any specific tool or method for characterizing his or her machine tool.

A distinction must be made between the machine characterization phase and the machine compensation phase. Machine characterization first determines that the machine has repeatable errors and then quantifies the magnitude of those errors. Repeatable errors are of the utmost importance in software error compensation because the machine tool errors are measured a priori, and since mathematical equations are used to calculate the compensation values, the same set of inputs will always produce an identical set of outputs and cannot compensate for random variations.

Several methods are available for measuring the motion errors of machine tool axes. In the past, the prevalent method for machine characterization was the use of certified artifacts such as step gauges. These artifacts have precisely spaced lugs that can be used as probing locations with a precise probing instrument such as an electronic indicator. A NC program would be developed to position the indicator tip on each lug so the deviation could be recorded. Today, laser interferometer systems have become the measurement tool of choice. They provide accuracies on the order of nanometers, and can be set up for automated data acquisition. By using other optics, laser systems can be set up to measure straightness and angular errors as well. The disadvantage of these systems is the time required to set up the laser and optics for a particular measurement. To characterize all the errors associated with one axis usually takes several days of data acquisition.

4.3.1 Laser Measuring System Description

At Y-12 a new laser system has been procured that significantly reduces the time required for error measurement. Automated Precision, Inc., in Gaithersburg, Maryland produces the 5-D Laser Measuring System, shown in Figure 4-6. By combining several sets of optics into one sensor head, the 5-D system allows the

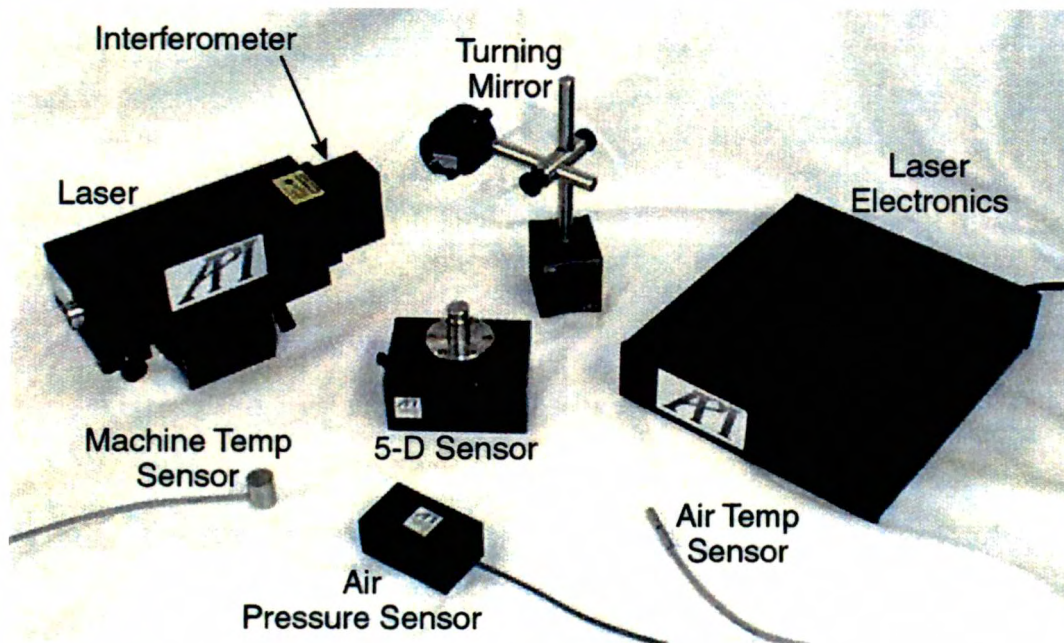


Figure 4-6. 5-D Laser Measurement System components

acquisition of five error measurements (scale, horizontal straightness, vertical straightness, pitch, and yaw) with a single setup of the laser. It is also capable of measuring squareness between axes. An available add-on module allows horizontal roll measurements to be made, creating a 6-D system, but that is currently not part of the Y-12 system. This 5-D laser system will be used to provide the error data for the test-bed lathe.

As described in the 5-D User's Guide [18], the laser used in the 5-D laser system is a Helium-Neon laser operated at a frequency around 473 THz, which produces a wavelength of 632.9 nm. As with all laser interferometer systems, absolute linear distance is not directly measured, only displacement from an initial position. Figure 4-7 depicts a simplified diagram of the technique used to measure linear displacements. The beam from the stationary laser head is split into two components by a polarizing beam splitter within a laser interferometer mounted to the end of the laser head. One component, the reference beam, is

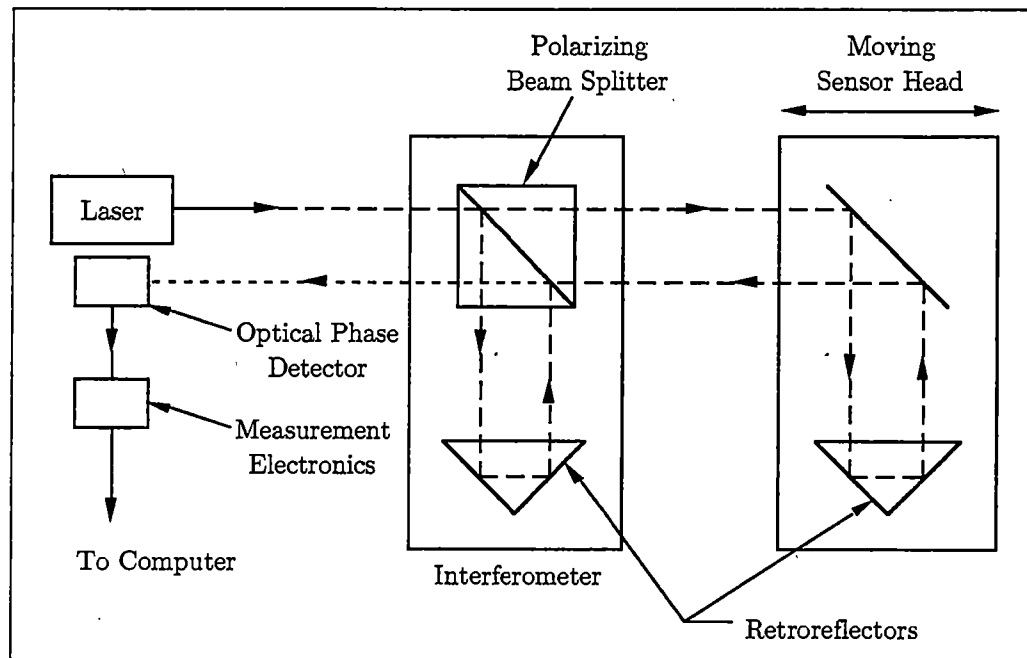


Figure 4-7. Simplified block diagram of 5-D linear measurements

sent to a retroreflector and returns to the beam splitter to await recombination with the second component. The second component is the measurement beam, and it travels to the 5-D sensor head that moves with the machine axis. The sensor redirects the beam to another splitter and retroreflector and returns the beam back to the interferometer through another opening. The two beam components are then recombined and directed to the optical phase detector. The recombination produces fringes that are used by the phase detector to determine phase difference between the two signals. The distance traveled is determined by counting these fringes. One fringe is equivalent to a distance equal to one-half of a wavelength.

With regard to linear measurements, the 5-D system is functionally equivalent to traditional laser interferometer systems. Differences are noticed when performing other types of measurements. Traditional systems use separate optics for straightness and angular measurements, whereas the 5-D system

incorporates additional sensors in its sensor head. Figure 4-8 shows a simplified diagram of these types of measurements using the 5-D laser measuring system.

The interferometer described earlier for linear displacement measurements is not used for straightness, pitch, or yaw measurements, so it was left out of Figure 4-8 to simplify the diagram. Upon entering the 5-D sensor head, the incoming beam is split into three parts. One part is sent to a retroreflector for linear measurements, as described previously; one is sent to a lateral motion sensor

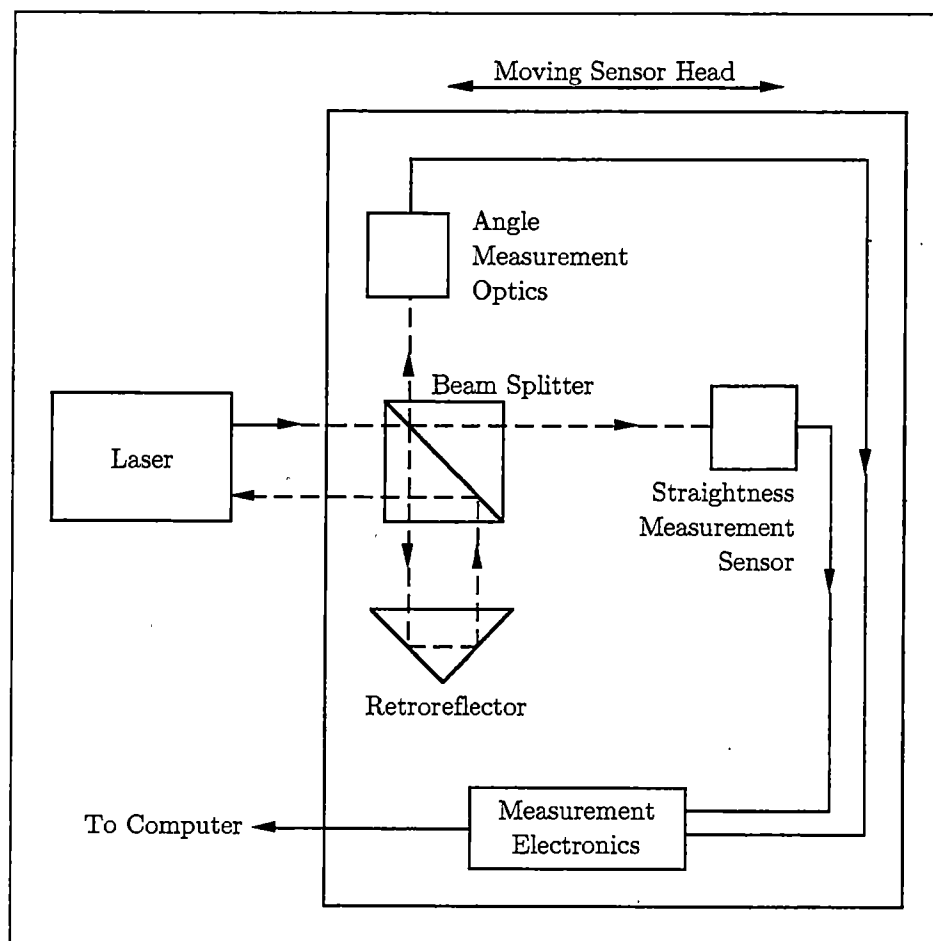


Figure 4-8. Simplified block diagram of 5-D straightness/angular measurements

for straightness measurements; and one is sent to an angular measurement optic for pitch and yaw measurements. Assuming axis motion in the X direction, the lateral motion sensor detects motion in the Y and Z directions. If the X motion were perfectly straight, the beam would remain centered on the sensor. Any movement in the Y or Z directions causes the beam to move away from center. This causes a change in the signal output from the sensor, which is converted to a distance by the control computer. A similar technique is used for angular deviations. Any rotational motion between the laser and the sensor causes changes in the signal output from the angular measurement sensor, which is converted into angular displacement readings.

Using traditional laser systems, straightness measurements would be performed by changing the optics that reflect the measurement beam back to the interferometer. A straightness mirror would replace the linear retroreflector and would be configured to provide horizontal straightness measurements. For vertical straightness measurements, the mirror would have to be reconfigured. Angular measurements for pitch and yaw would require setup of their specialized optics as well. Configuring each of these tests takes several minutes at a minimum and possibly a few hours. The 5-D laser system provides the same data as the traditional systems, but is able to obtain that data with only one setup of the optics instead of five. Considering the savings realized from the reduced time required by the plant's machine maintenance personnel, one can see the advantages of using the 5-D system for characterizing machine tool errors.

4.3.2 Excella Error Data Acquisition

In Section 4.2, Equations (4-31) and (4-32) were developed that described the cumulative position errors of each axis due to its geometric errors. Each term in the equations contained error information that must be measured. Various

Table 4-3. Excello error equation components

Component	Description	Variable type	Measurement used
$\delta_x(x)$	X-axis lead screw	Array	5-D laser
$\delta_z(x)$	X-axis straightness	Array	5-D laser
$\epsilon_y(x)$	X-axis yaw	Array	5-D laser
α	Squareness	Value	Electronic indicator
$\delta_z(z)$	Z-axis lead screw	Array	5-D laser
$\delta_x(z)$	Z-axis straightness	Array	5-D laser
$\epsilon_y(z)$	Z-axis yaw	Array	5-D laser
β	Parallelism	Value	Electronic indicator

methods can be used to obtain these measurements. Table 4-3 lists the needed variables and the method of measurement to be used.

When making measurements of this type, one must take into consideration the effects of Abbe errors. These errors are present whenever linear measurements are not made with the measuring instrument directly in line with the displacement being measured. A common example used to illustrate this effect is measuring the diameter of a cylinder using both a micrometer and a dial caliper. With the dial calipers, the jaws holding the cylinder are offset from the measurement scale, and inaccurate readings occur because of the flexing of the jaws. The micrometer, on the other hand, has its measuring scale inline with the holding clamps, which provide a direct linear measurement.

When measuring lead-screw positioning errors, one would ideally like to measure along the screw itself because Abbe errors are introduced as the measurement location moves away from the screw. For the Excello characterization all linear measurements were made at tool height. While this does not eliminate the Abbe error, the data contain the same amount of error as

the tool, so the measurement does not introduce any additional error relative to the tool.

4.3.2.1 X-axis measurements

The measurements dealing with the X-axis will be described first. The six error parameters measured with the 5-D laser were obtained using two setups of the equipment. Figure 4-9 shows the setup used on the Excello's X-axis. Although most machining operations occur with the X-slide toward the positive end of travel, motion errors throughout the entire range of travel were measured. The two-axis mechanism holding the laser head was not part of the 5-D system but was purchased separately to expedite alignment of the laser since vertical adjustments were not possible with a two-axis lathe.

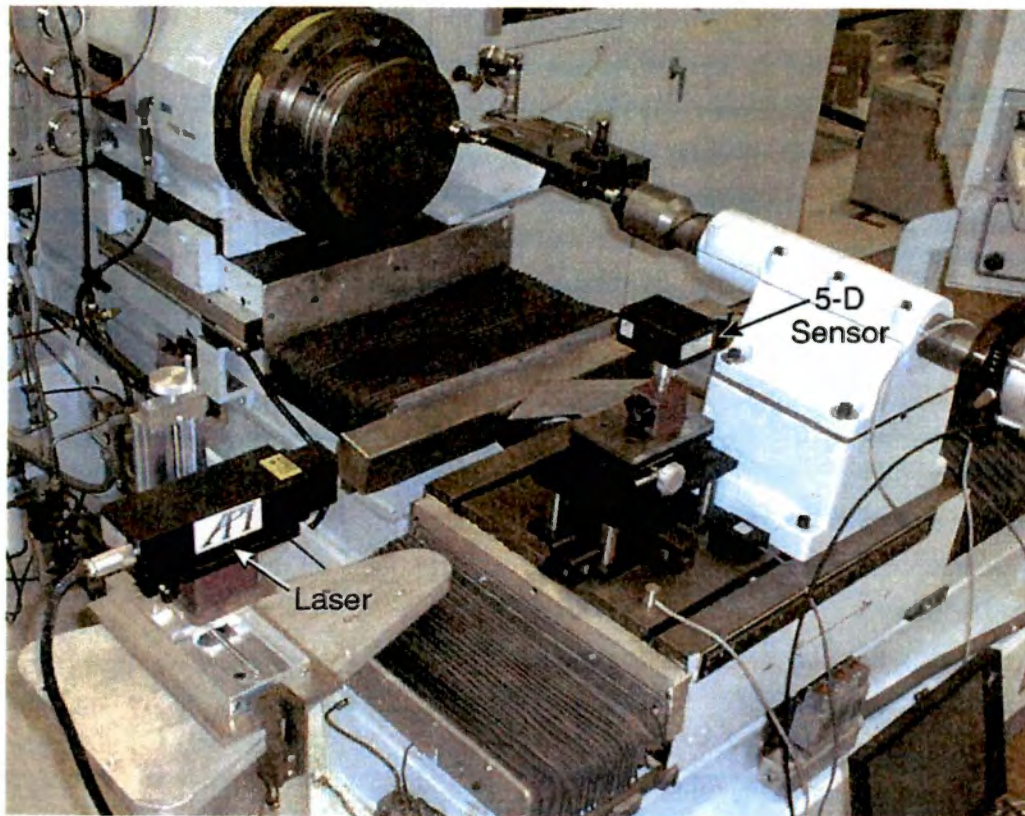


Figure 4-9. X-axis error measurement setup

The spacing between error measurements was 1 inch, which corresponds to a recommended interval [4] of no more than 5% of the total axis travel of 22 inches. A NC program was written that incrementally moved the slide to each measurement location and dwelled for 10 seconds to allow the laser sensor readings to settle. After aligning the laser equipment and warming up the X-slide, a total of three bi-directional runs were conducted, starting at the negative end of the X-slide. These results were averaged to obtain the final error characteristics. Complete error measurement results showing all data as graphed by the 5-D system are shown in Appendix C.

Figure 4-10 shows the average lead-screw positioning errors from the three bi-directional runs. The forward direction is the initial movement direction of the 5-D sensor relative to the laser. For the X-axis measurements, motion started at $X = -21$ inches and moved toward $X = 0$. Near the positive end of travel, the forward and reverse readings tracked each other very well; in the middle and near the negative end of travel, they did not track as well. The difference between corresponding readings in the forward and reverse directions is referred to as *reversal error*. The 5-D system reported the maximum reversal error for this data as 0.123 mils, which is not particularly good for this machine. Expected reversal for these lathes would normally be on the order of 50 millionths of an inch. This particular machine had not been used in many months and was in need of some maintenance operations. Unfortunately, the budget allocated to this project did not have sufficient resources to fund that work.

The maximum average positioning error is computed by taking the difference between the highest and lowest average readings. From Figure 4-10 it is seen that the maximum average error was nearly 1 mil, which again would be unexpected for machines with linear scale position feedback. Also notice that the errors have

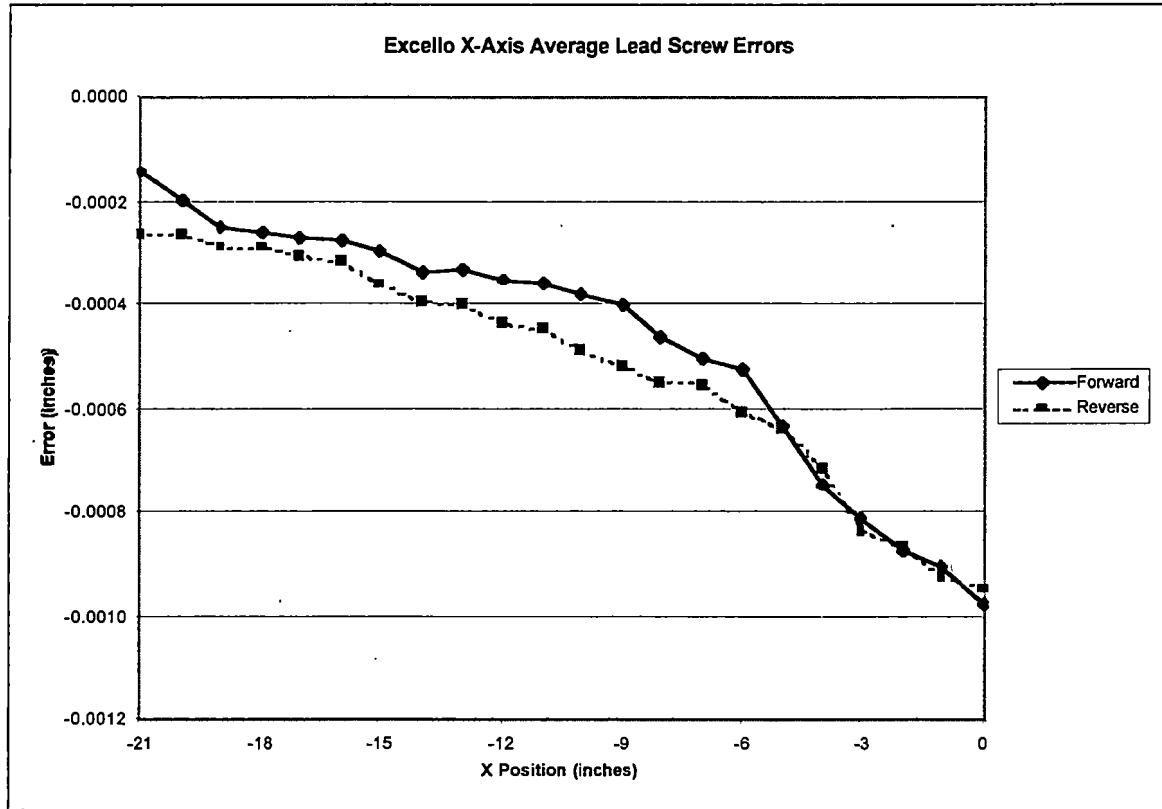


Figure 4-10. X-axis lead-screw errors

negative values, which indicates that the slide was always short of its commanded position. Positioning errors of the X-slide are represented in the error equation for the X-slide as $\delta_x(x)$ and are corrected with compensating movements of the X-slide.

The horizontal straightness errors shown in Figure 4-11 quantifies the amount of movement that the X-slide makes in the Z-direction as it traverses the X-axis. The 5-D sensor was oriented such that positive straightness values corresponded to positive displacements as defined by the Excello error model. Unlike the positioning error diagram, the straightness error diagram has both

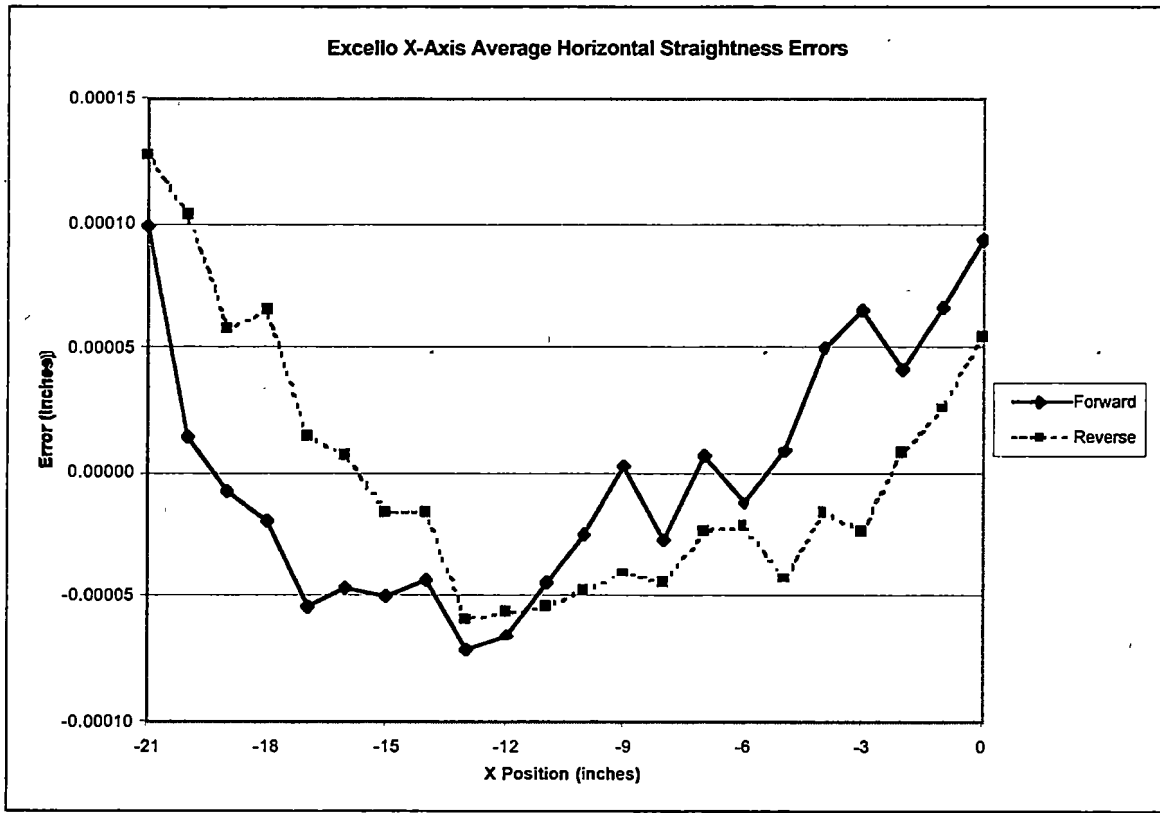


Figure 4-11. X-axis horizontal straightness errors

negative and positive values. The maximum average error, again computed by taking the difference between the error limits, was calculated as 0.2 mils. While this is an error of the X-slide, represented as $\delta_z(x)$ in the error equations, it is compensated for by the Z-slide.

The yaw errors of the X-slide, depicted in Figure 4-12, show how the slide twists as it moves along the X-axis. Positive angular errors correspond to positive rotations as defined in the Excello error model. The values ranged from +3 to -3 arc-seconds, and some difference between forward and reverse readings was noticeable. While these values are very small angular errors, it should be noted that angular errors are amplified by sometimes considerable moment arms to create relatively significant linear errors. This yaw error is represented in both

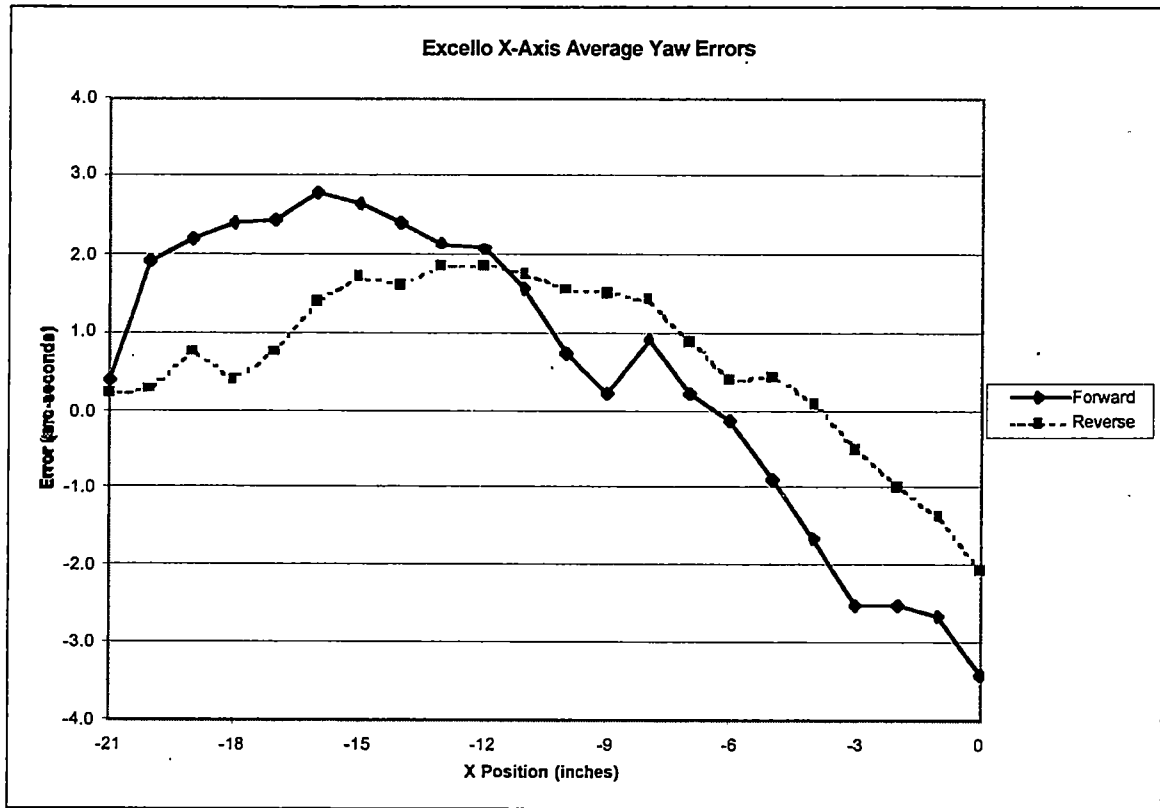


Figure 4-12. X-axis yaw errors

the X and Z error equations as $\epsilon_y(x)$, and both axes must compensate for these errors.

The other X-axis error measurement made was to determine the squareness error between the X-axis and the lathe's spindle. This measurement was not made easily using the 5-D system, so a more traditional method was used. As described in the American Society of Mechanical Engineers (ASME) B5.57 Lathe Evaluation Standard [19], the method involves measuring deviations across a flat surface mounted to the spindle of the machine. The deviations are measured at various locations of the X-axis; the spindle is then rotated 180°, and the measurements are repeated. This process removes any flatness errors of the measured surface. The two sets of readings are plotted, and the squareness error

is calculated by averaging the slopes of the best-fit lines between the readings. The equipment used for this measurement is shown in Figure 4-13.

Special care was taken in the setup of the electronic indicator with regards to the sign convention. The direction of a positive deviation was made to correspond to the direction of a positive squareness error as shown in the error model diagram, Figure 4-5. The results of the measurements are shown in Figure 4-14.

The best fit line through the data taken at a spindle rotation of 0° (called the Forward data in Figure 4-14) had a slope of 0.054 mils/inch, while the best fit line through the data taken at a spindle rotation of 180° (called the Reverse data)

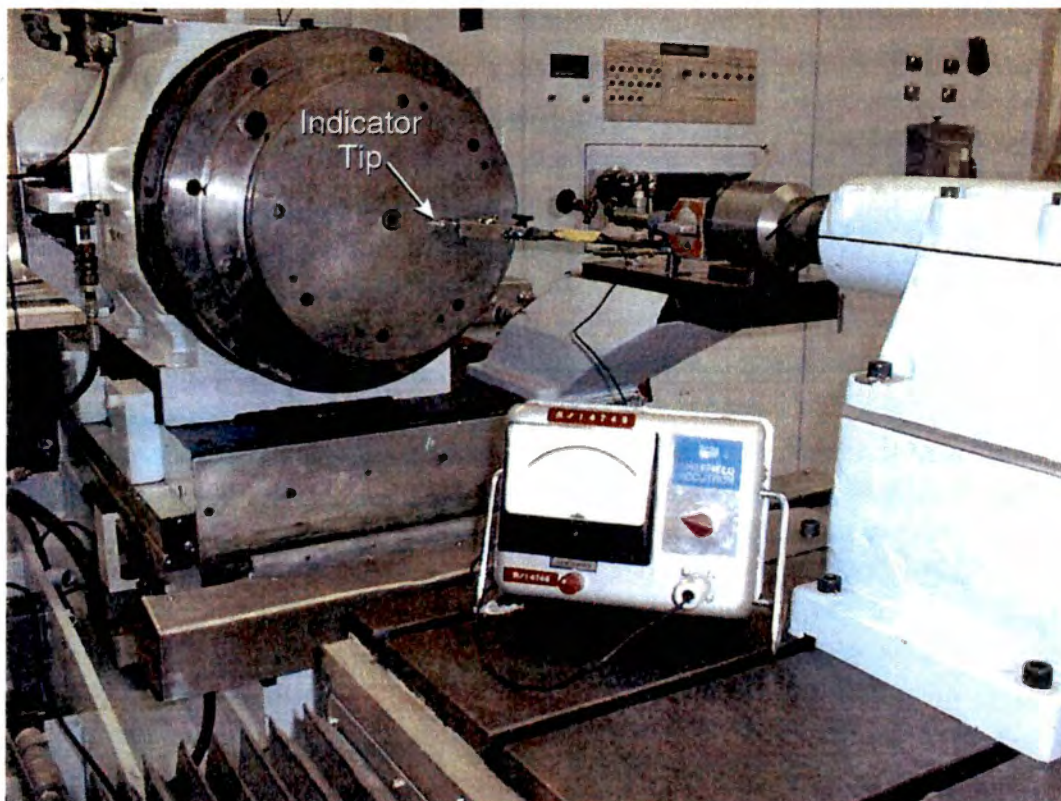


Figure 4-13. Squareness error measurement

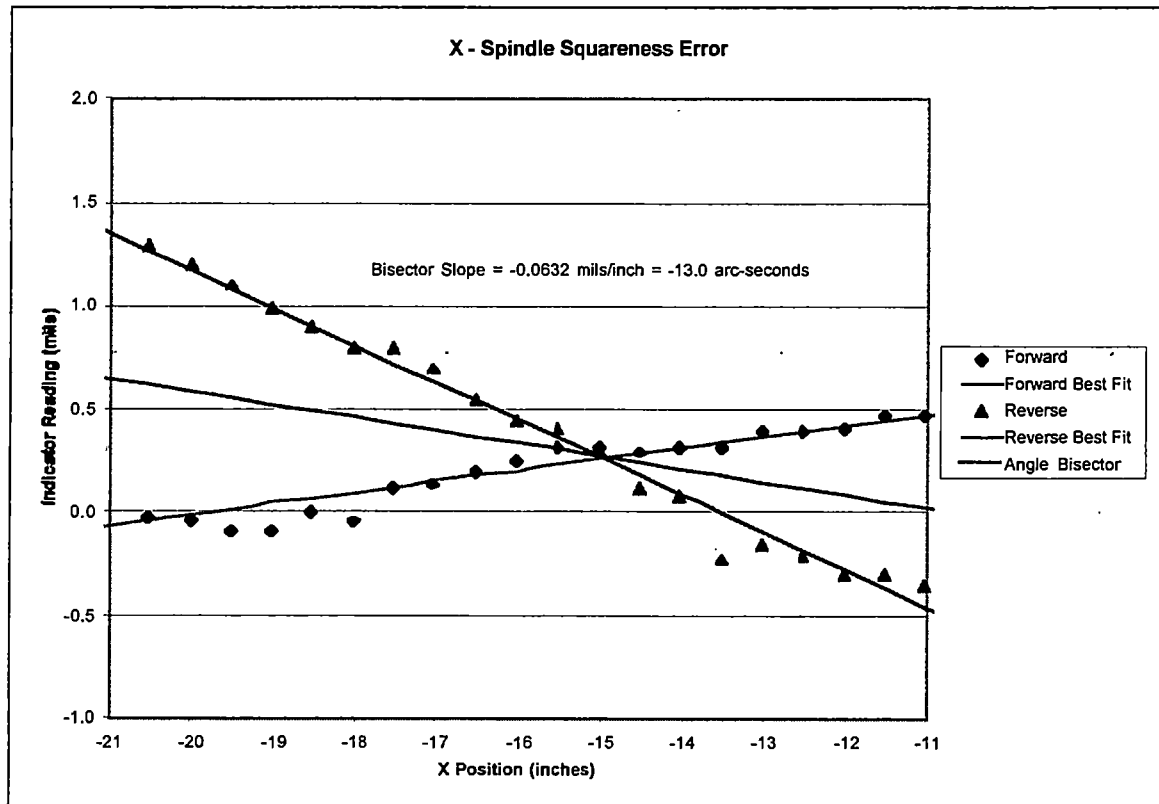


Figure 4-14. Squareness error data

had a slope of -0.180 mils/inch. The average slope was calculated to be -0.063 mils/inch. This corresponded to a squareness error of -13.0 arc-seconds.

4.3.2.2 Z-axis measurements

The measurements of the Z-axis were similar to those for X except in the setup of the laser equipment. As shown in Figure 4-15, the position of the laser was not changed; an additional turning mirror was used to redirect the beam. The magnetic base holding the 5-D sensor was purchased expressly to allow easier alignment of the laser beam with the sensor head.

The interval between measurement locations for the Z-axis was set to 0.5 inch, which was approximately 5% of the total axis travel. The same dwell

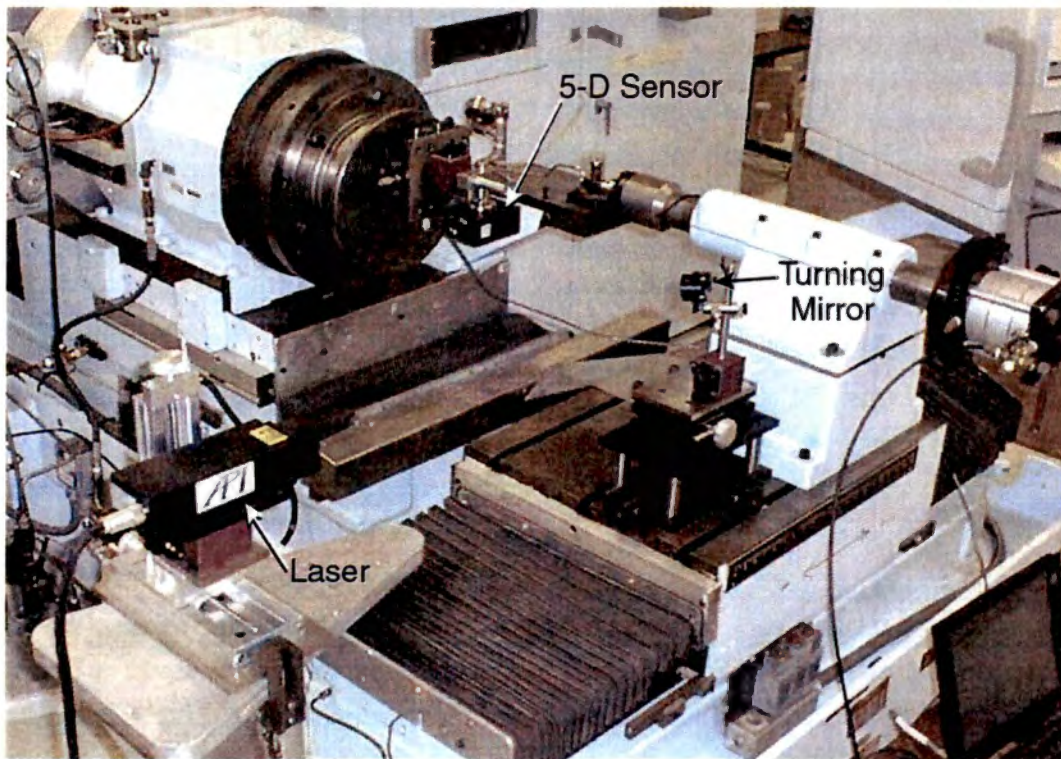


Figure 4-15. Z-axis error measurement setup

time (10 sec) between measurements and total number of runs (3) matched those used for the X-axis. The error measurements started at the positive end of the Z-slide. As was the case for the X-axis data, complete graphs from the 5-D system are included in Appendix C.

The positioning errors of the Z-slide are shown in Figure 4-16. Data acquisition for this axis started at $Z = 11$ inches and moved in the “forward” direction toward $Z = 0$, so the forward direction in the figure is read from right to left. This could also be deduced by noticing that the forward reading on the right side of the graph is closest to zero; when beginning acquisition of 5-D data, the linear reading is zeroed at the starting location.

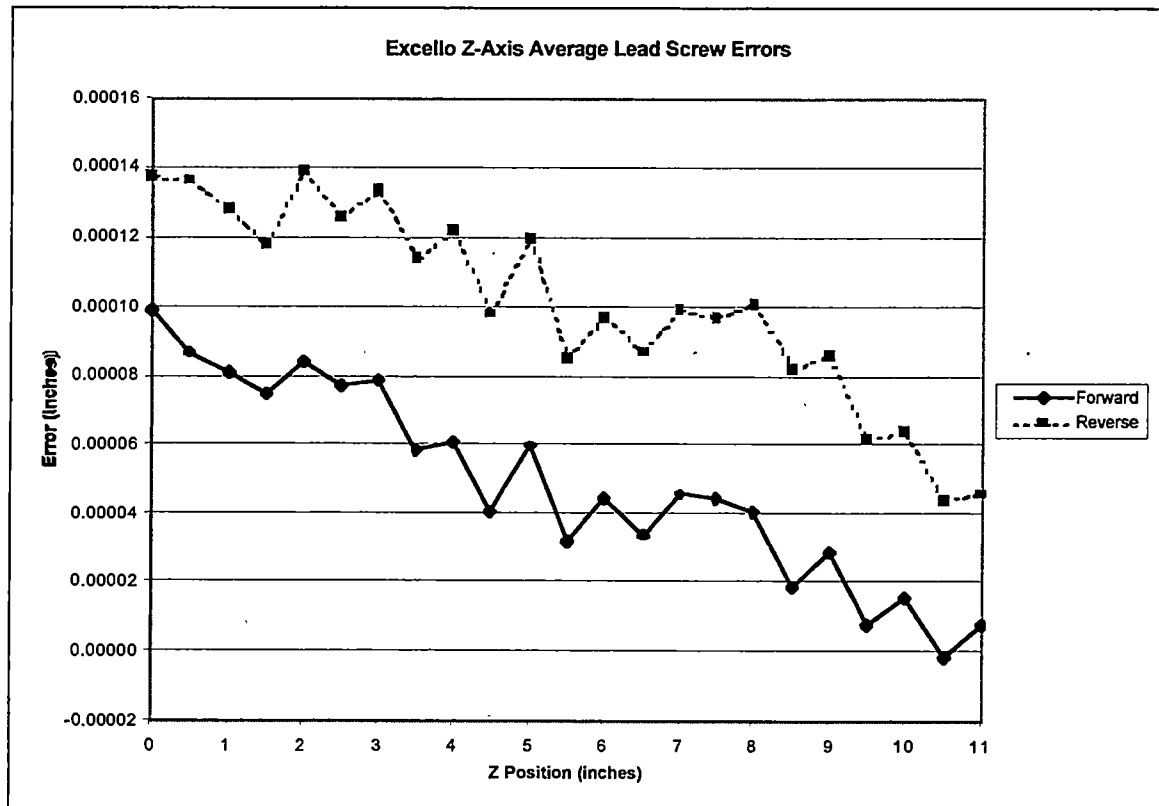


Figure 4-16. Z-axis lead-screw errors

The maximum average error was calculated to be 140 millionths of an inch. Maximum reversal error was -64 millionths, which was half that of the X-slide, even though Figure 4-16 seems to show a larger gap between the forward and reverse data. The scales of Figures 4-10 and 4-16 are different, which accounts for the gap size. The Z-slide is actually more accurate than the X-slide on this machine. As was the case with the X-slide, positioning errors of the Z-slide [$\delta_z(z)$ in the error equations] are corrected only by the Z-axis.

Figure 4-17 shows the average horizontal straightness errors of the Z-slide. The graph shows that the straightness data do not follow a smooth curve as is usually seen with linear positioning data. It should be noticed that the magnitude of the fluctuations is on the order of 30 millionths of an inch, so the fluctuations

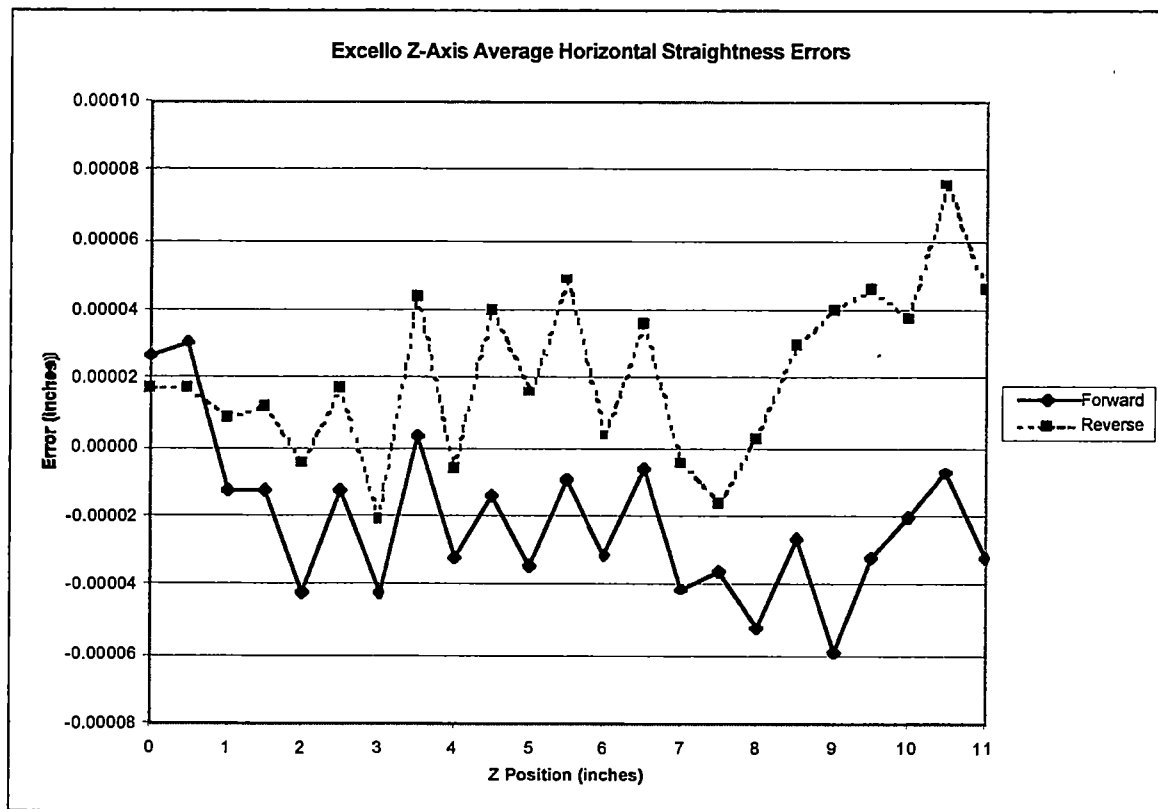


Figure 4-17. Z-axis horizontal straightness errors

may be an artifact of the small, random motions of the slide. Of somewhat more concern is the fact that the forward and reverse data do not track each other well on the positive end of the slide. This may be due to some mechanical binding of the slide or perhaps indicates a nonrigid setup of the 5-D sensor. Positive straightness errors in Figure 4-17 are in the +X direction. These errors require compensation by the X-slide and are represented as $\delta_x(z)$ in the error equations.

The yaw errors of the Z-slide are shown in Figure 4-18. Positive angular errors correspond to positive rotations as defined in the Excello error model. The values ranged from +0.2 to -2.0 arc-seconds, and the forward and reverse readings tracked each other very well. As was the case for the X-slide yaw errors, multiplying angular errors by significant moment arms can create significant

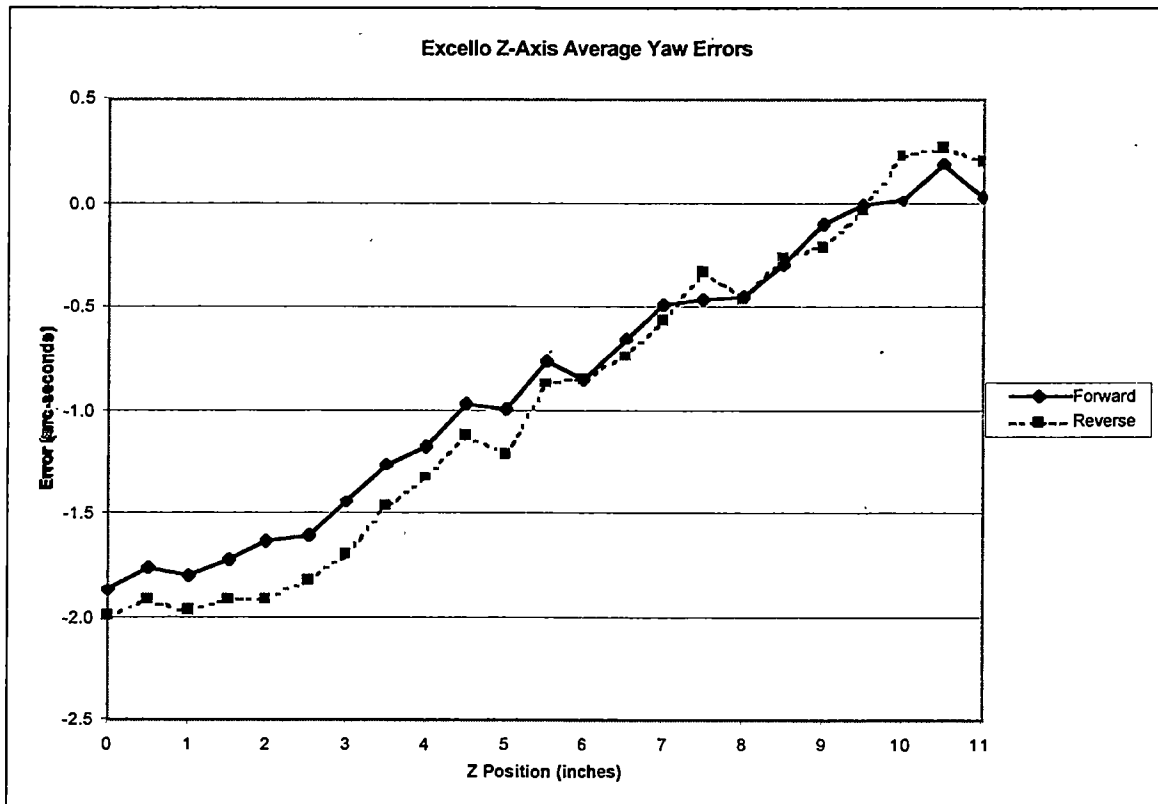


Figure 4-18. Z-axis yaw errors

linear errors. This yaw error is represented in both the X and Z error equations as $\epsilon_z(x)$, and both axes must compensate for these errors.

The final error measurement needed in the error equations was the parallelism error between the Z-slide and the spindle. As was the case with the squareness measurement, an electronic indicator was used to make the measurements. Instead of measuring across a flat face, however, for parallelism measurements, a cylinder square was used. The equipment setup is shown in Figure 4-19.

The cylinder square was centered on the faceplate, and the indicator was positioned on one end of the cylinder on the +X side. The Z-slide was then incrementally moved and deviation data recorded. The spindle was then rotated

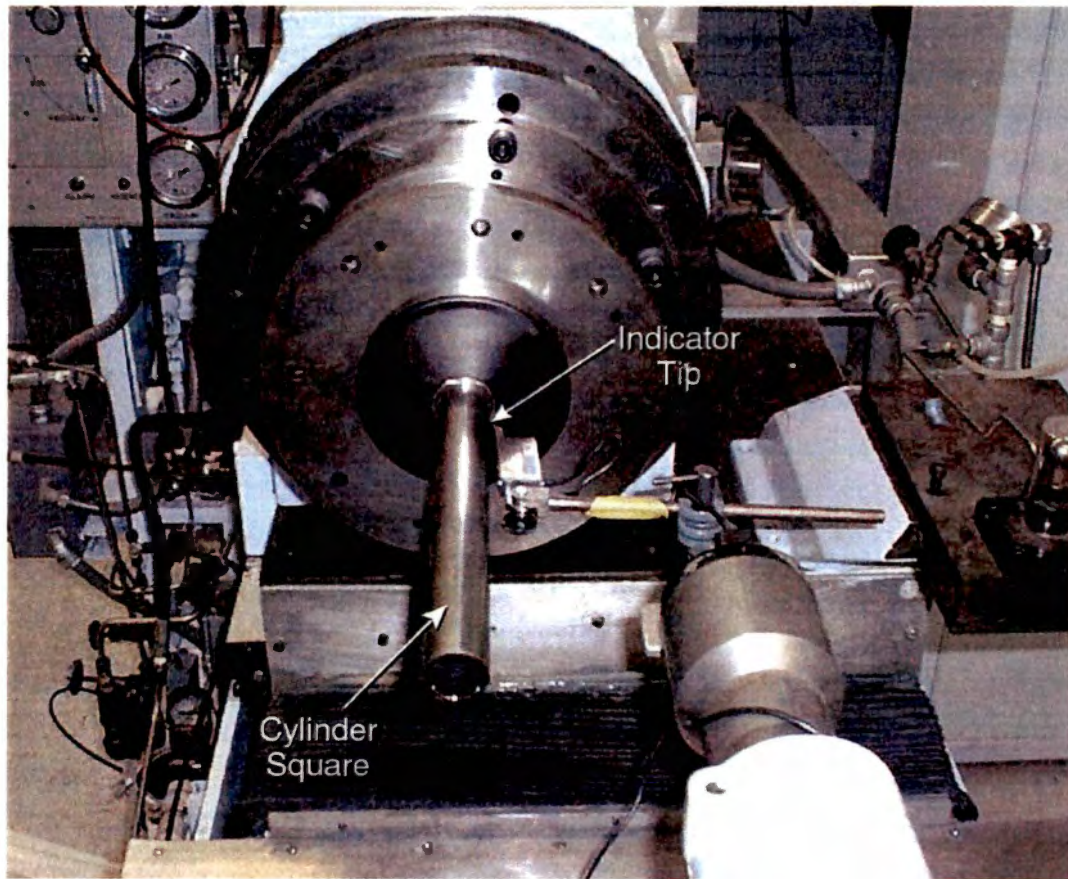


Figure 4-19. Parallelism error measurement

180°, the Z-slide was returned to its starting location, and the indicator was rebracketed so that it could again start on the same point on the cylinder, which was now on the -X side, without having to move the X-slide. The Z-slide movements were repeated and deviation data recorded. The results are shown in Figure 4-20.

The best fit line through the data taken at a spindle rotation of 0° (called the Forward data in Figure 4-20) had a slope of -0.138 mils/inch, while the best fit line through the data taken at a spindle rotation of 180° (called the Reverse data) had a slope of 0.225 mils/inch. The average slope was calculated to be 0.043 mils/inch, which corresponded to a parallelism error of 8.9 arc-seconds.

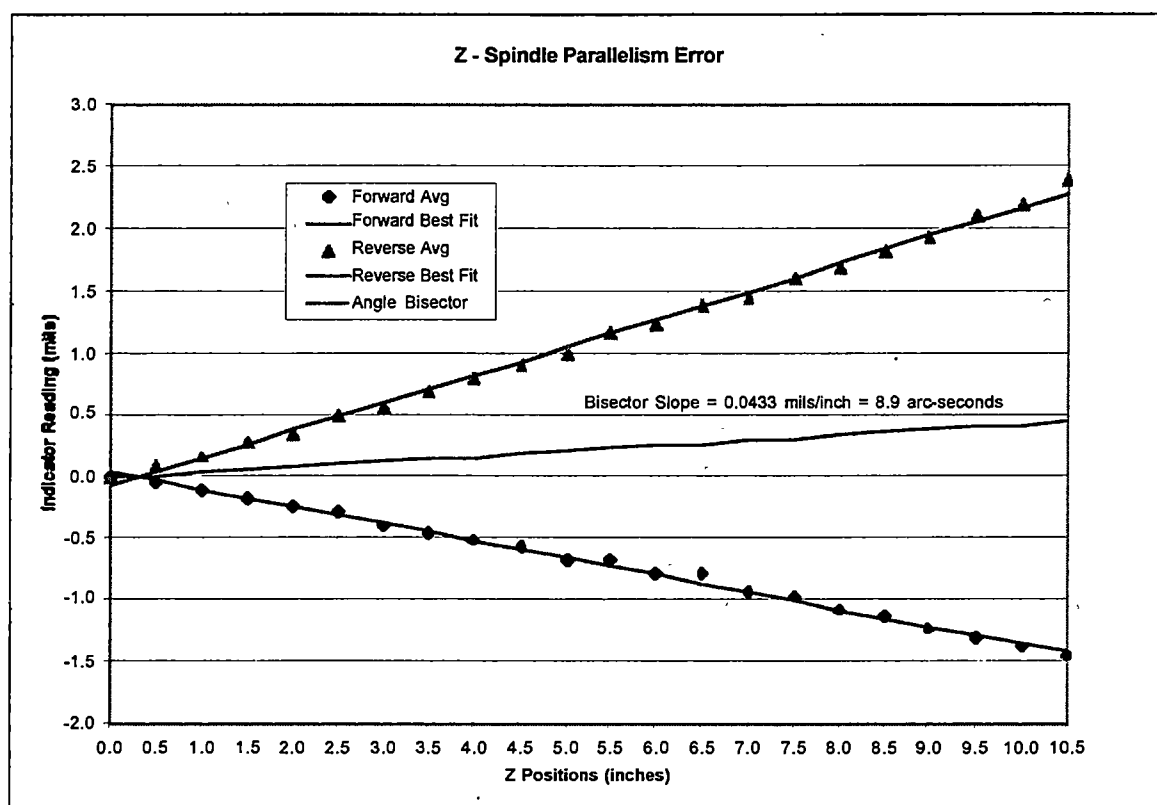


Figure 4-20. Parallelism error data

At this point, all needed error measurements had been obtained. The data was then ready to be converted into the proper format for the Geometric Error Compensation System (GECS) Error File, which will be described in Chapter 5. The various error measurements were acquired over a period of several days because some of the measurement instrumentation and equipment was not readily available. Had all the equipment been available at the same time, and all NC programs prepared beforehand, the entire suite of tests could have been performed in a single day.

Chapter 5

System Implementation

The purpose of this chapter is to provide details on how the GECS was implemented within OpenCNC. A generic implementation directed toward three-axis machines will be discussed, and then the specific details of the Excella implementation will be described. For the Excella, it will be assumed that machine error measurements were made with a 5-D laser system.

5.1 OpenCNC Overview

In designing the architecture of the GECS, an attempt was made to follow the architecture and conventions of the current OpenCNC Lead-Screw Error Compensation (LSEC) System. Compensation data for the LSEC System is stored in the parameter file read by OpenCNC during system start-up, which is called the Tune File. Its format is a series of variable names, some of which contain array subscripts, followed by a value. Variables and data are applied to different axes of the machine tool, which are differentiated within OpenCNC by axis numbers. Axis numbers start at 0 and increase by 1 for each additional axis. The machine spindle is typically assigned the highest axis number. For the

Excello lathe, axis number 0 was assigned to the X-axis, axis number 1 to the Z-axis, and axis number 2 to the spindle.

OpenCNC variable naming conventions include beginning all variable names with specific prefixes that indicate to which group a specific variable belongs [20]. The groups are shown in Table 5-1.

OpenCNC stores LSEC data in two arrays, `axLSCompDirPos` and `axLSCompDirNeg`. These are two-dimensional arrays that are specific to the direction of motion. The first dimension subscript references the compensation data points, and the second references the axis number. Other arrays needed to implement the LSEC System are `axLSCompCount`, which contains the number of measured axis positions for each axis; `axLSCompSpacing`, which contains the spacing increment between measured axis positions; and `axLSCompPosMin`, which contains the starting locations of the measured axis positions. By convention, the starting location of the measurements is always at the most negative axis location. Each of these arrays uses the subscript to designate the machine axis the data applies to. For example, `axLSCompSpacing[0]` contains the distance between measurement points for axis 0, and `axLSCompSpacing[1]` contains the distance for axis 1.

5.2 GECS Implementation Overview

In characterizing the machine errors, each axis would be measured independently using the 5-D laser system, producing one set of data files for each measured axis. The data files, which are in man-readable American Standard Code for Information Interchange (ASCII) format, would each contain bi-directional error data. The 5-D system could also be used to measure the three squareness values, although other methods may be used and may be more efficient. Thus, 18 of the 21 error components associated with Equation (4-14)

Table 5-1. OpenCNC variable groups

Group	Description
def	Definition variables—define general system parameters and user preferences
ax	Axis variables—contain information specific to each axis. Variables in this category are arrays containing one entry for each axis on the machine
enc	Encoder variables—contain information that is read out of the encoder's hardware registers. Variables in this category are arrays containing one entry for each encoder on the machine
dac	DAC digital-to-analog variables—describe information that is written to the DAC's. Variables in this category are arrays containing one entry for each DAC on the machine
sp	Spindle variables—define characteristics of the machine's spindle
js	Job stream variables—a collection of variables that OpenCNC requires for each job stream. Variables in this category are arrays containing one entry for each job stream
blk	Part program block information—describes information about the current part program. Variables in this category are arrays containing one entry for each job stream that executes a part program
m	Miscellaneous variables such as feedhold and failhold status, override status, and the status of several modal conditions, etc.
h	Hardware variables—required for specific hardware components supported by OpenCNC

would be obtained; only roll would not be obtainable with the 5-D system. In the case of the Excello lathe, roll error is in a *nonsensitive* direction, meaning it would not affect the diameter of a turned part, so it would be of lesser concern. For machines where roll must be considered, other instruments are available for measurement, or a 6-D laser system could be used.

Once the error data was obtained, it would be converted into a format usable by OpenCNC and stored in a data file to be read upon start-up by the system. Since the error data would not change during operations, it would only need to be read once, and system start-up would be a logical time. The error data file, which will be referred to as the Error File, would be kept separate from other OpenCNC start-up files since its use would be optional, and it would isolate the OpenCNC compensation system code from the rest of the system.

Once the Error File data is read into OpenCNC, compensation should behave just as OpenCNC's LSEC is implemented. This relationship is shown in Figure 5-1. As the axes are moved under program control, the current axis positions and directions of motion are obtained and used to determine the corresponding locations within the appropriate error arrays. The values from the arrays are used in the error equations to calculate the amounts to move each of the axes, and the modified position command is sent to the motor amplifiers. These steps must occur during each system interrupt. As seen in Equations (4-31) and (4-32), there are many mathematical operations involved, even for the relatively simple Excello equations. Any implementation should incorporate some study of the CPU time needed to complete the calculations to determine the minimum interrupt interval that could be supported. A discussion of the CPU time required for the Excello implementation is discussed in Section 5.3.5.

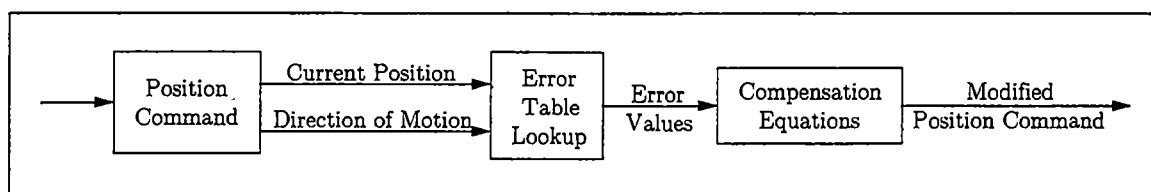


Figure 5-1. Control system compensation loop flowchart

5.2.1 Generic Approach

In its initial design, the GECS architecture was tightly integrated into the OpenCNC source code. The error data would have been read into memory sometime during OpenCNC start-up, probably after reading the Tune File. The actual compensation code would have been inserted directly into the position loop of the controller, which would have required modification of some of the lowest level, most efficiently written, OpenCNC code. These modifications would have been made by MDSI personnel since access to this code by the typical end-user would not have been permitted by MDSI for obvious safety reasons. The compensation values would have been calculated during each system interrupt. The time interval between interrupts is user-configurable through the Tune File. There would have been a large number of computations required to calculate the compensation values because (1) several error table lookups would be required to obtain all the necessary component error quantities and (2) the correction equations themselves would require several CPU cycles to perform. It is entirely possible that not all of these computations would have been performed within the given timeframe.

Additionally, it was assumed that GECS support for all machine types would have been provided within a single OpenCNC executable. During machine start-up, the OpenCNC Error File would indicate the type of machine tool being

controlled, and the appropriate set of error equations would be loaded into memory. It was envisioned that the source code of the GECS would eventually contain several different sets of error equations corresponding to the particular set of supported machine tools. A variable in the Error File would be used to set pointers to the needed equations. This would prevent the need to check the machine type during every system interrupt with if-then-else-endif type constructs. This approach also implied that the error equations for any new machine would have to be compiled into the OpenCNC source code. A method to allow the user to incorporate new equations without having access to the source code would have been useful.

A generic error table naming convention was another major implementation issue. Table 5-2 shows all the possible error components for a three-axis machine. Computer array names would be needed for all the linear and angular components since they would be stored in tables. Under LSEC, a single 2-dimensional array held the error data for all axes for positive direction motions, and another array held the data for negative direction motions. Using the same single array approach for GECS would have required multiple offsets into the array, which would have caused unnecessary confusion.

Table 5-2. Three-axis machine error components

Linear error			Angular error			Squareness
$\delta_x(x)$	$\delta_y(x)$	$\delta_z(x)$	$\epsilon_x(x)$	$\epsilon_y(x)$	$\epsilon_z(x)$	α_{xy}
$\delta_x(y)$	$\delta_y(y)$	$\delta_z(y)$	$\epsilon_x(y)$	$\epsilon_y(y)$	$\epsilon_z(y)$	α_{xz}
$\delta_x(z)$	$\delta_y(z)$	$\delta_z(z)$	$\epsilon_x(z)$	$\epsilon_y(z)$	$\epsilon_z(z)$	α_{yz}

It was important in the software design that the names of the arrays provide enough information so that a person familiar with the GECS could easily understand the error component in question by the name. While the Greek

letters in Table 5-2 are very useful in algebraic equations, they aren't well suited to being used as array names. Incorporating axis letters would make descriptive names, but OpenCNC typically uses axis numbers to distinguish axes in order to allow the user to use letters other than XYZ. Therefore, it was decided that array names should be of the following form:

ax <type> <error axis> <motion axis> Dir <dir>

where

type	Lin, Str, or Ang
error axis	axis number corresponding to direction (axis) of error
motion axis	axis number corresponding to direction (axis) of motion
dir	Pos or Neg

This convention would transform Table 5-2 error components into the Error File array names found in Table 5-3.

Table 5-3. Three-axis machine error array names

Linear Error Arrays		
axLin00DirPos	AxStr10DirPos	axStr20DirPos
axLin01DirNeg	AxStr10DirNeg	axStr20DirNeg
axStr01DirPos	AxLin11DirPos	axStr21DirPos
axStr01DirNeg	AxLin11DirNeg	axStr21DirNeg
axStr02DirPos	AxStr12DirPos	axLin22DirPos
axStr02DirNeg	AxStr12DirNeg	axLin22DirNeg
Angular Error Arrays		
axAng00DirPos	AxAng10DirPos	axAng20DirPos
axAng00DirNeg	AxAng10DirNeg	axAng20DirNeg
axAng01DirPos	AxAng11DirPos	axAng21DirPos
axAng01DirNeg	AxAng11DirNeg	axAng21DirNeg
axAng02DirPos	AxAng12DirPos	axAng22DirPos
axAng02DirNeg	AxAng12DirNeg	axAng22DirNeg

In most configurations, axis0 would map to X, axis1 would map to Y, and axis2 would map to Z. Thus, axLin10DirPos would contain those linear errors in the Y direction (axis1) for positive motions of the X-axis (axis0). Similarly, axAng21DirNeg would contain those angular errors around the Z direction (axis 2) for negative motions of the Y-axis (axis1). In the case of the Excello lathe, axis0 maps to X, but axis1 maps to Z, so if axis letters were used instead of axis numbers, the array names would not match the actual data.

In the linear error arrays, those with names containing *Lin* would contain lead-screw error data. The *Str* arrays would contain horizontal or vertical straightness error data; direction would be dependent on the error model of the machine. Angular error arrays with matching axis numbers would contain roll data (axis of rotation matches axis of motion). The other arrays would contain either pitch or yaw data. Squareness values would be stored in variables names axSqu01, axSqu02, and axSqu12.

A point worth reiterating is that the use of any or all of the error arrays would be optional. It would be up to the metrology engineer to decide which errors were significant enough to require compensation for the particular machine tool in question. This would become especially critical if the time required to obtain a given set of error data were prohibitive, such as when a 5-D system is not available. Defaults for all the arrays would be zero, so any array not specified in the Error File would not cause invalid results from the error equations; only some terms in the error equations would drop out. However, this would imply that the returned values were not as accurate as if the complete equation were utilized.

5.2.2 Machine-specific Approach

Upon discussions with MDSI personnel, it turned out that a simpler solution would be to design the compensation system as a separate process from the main OpenCNC executable. The two processes would each have their own CPU resources and would communicate through shared memory. In this design, the priority assigned to the compensation process could be different from that of the motion process, thus ensuring that the motion process always had adequate CPU resources. The compensation process would have its own interrupt rate, again independent from the motion process. The default interrupt rate for the OpenCNC motion process is 512 interrupts per second, which equates to an interrupt approximately every 0.002 seconds. The initial interrupt interval for the compensation process will be 0.010 seconds, equating to a rate of 100 interrupts per second. But this rate could easily be changed within the GECS source code. Further study of the amount of calculation time for the error equations and overall system effects would be needed before a maximum compensation interrupt rate could be established. The fact that the current position of the axes would be updated faster than the compensation system could utilize them is another area of study. As shown by the timelines in Figure 5-2, the same compensation values would be in effect over multiple motion interrupts. The approximate number of motion interrupts is calculated as I_c/I_m , where I_c is the time interval between compensation system interrupts, and I_m is the time interval between motion system interrupts. The actual change in axis location during this time would, of course, be dependent upon the velocity of the axis.

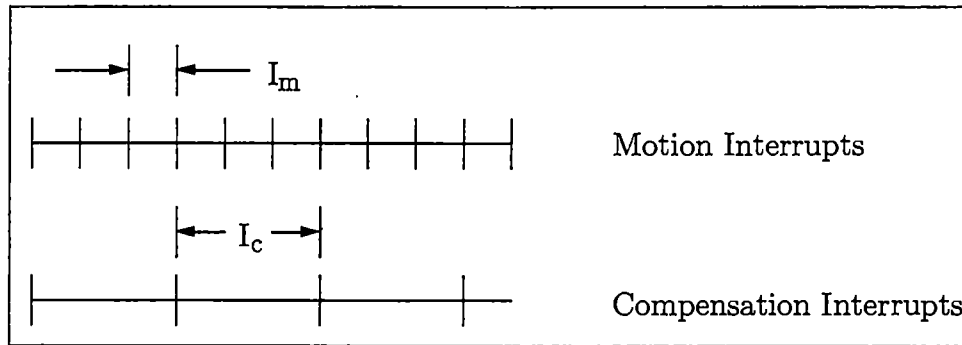


Figure 5-2. Compensation process time-lines

As a numeric example, assuming the interrupt intervals mentioned previously ($I_c = 0.010$, $I_m = 0.002$), the axis slide would have been moving for five motion interrupts before new compensation values were calculated. Assuming an axis velocity of 20 ipm, this equates to a distance traveled of

$$\left(\frac{20 \text{ inch}}{60 \text{ sec}} \right) \left(\frac{I_c}{I_m} \right) (I_m \text{ sec}) = 0.33 \text{ ips}(5)(0.002 \text{ sec}) = 0.0033 \text{ in}$$

before new compensation values would be obtained.

The same effect would be noticed for the remaining machine axes that were in motion. For a three-axis machine, the tool compensation values would be calculated for a position that was a few mils from the actual location. The actual error in the compensation values would depend upon the error equations, but it is assumed that the error map of the machine would change slowly for small motions.

To facilitate the incorporation of the GECS functionality into OpenCNC, MDSI provided an interface and a template for an external compensation system. This general-purpose template will allow any type of the compensation system to be implemented within OpenCNC. Thermal compensation, tool-wear compensation, and geometric compensation are some examples that could be

supported with this template. The process is started by the OpenCNC start-up script and is assigned its own priority. The compensation process and the main executable exchange information through shared memory, which provides the compensation system access to any needed information within the controller. The existing template changes the value in a register that informs OpenCNC that external compensation is enabled and displays a window that contains the current compensation values. The compensation system performs some initialization functions, during which the Error File is read and loaded into memory. The remainder of the time the compensation process spends in a loop, reading the current axes' positions, calculating the compensation values, and changing the current commanded positions. The source code for this external process is available to the user and can be modified without intervention from MDSI. This gives the user the ability to interface external sensors and other systems with a commercial controller in an easy fashion. The ability to manipulate the controller in this way will be extremely beneficial to those users doing development tasks, as well as providing a supported means of improving the accuracy of a given machine tool.

5.3 Excello Implementation Specifics

In this section, more specific information regarding the actual implementation of the GECS on the Excello test-bed machine is given. Topics include the contents of the Error File, the internal array naming convention chosen, and the internal units of measurements used for the error data. A conversion utility developed to convert the 5-D data files into an Error File is also described. Finally, the section ends with an overview of the GECS source code.

5.3.1 Error File Contents

The Excello Error File was constructed to contain all the necessary parameters needed by the GECS to compute the error values given by the error equations. Since the GECS was designed to use machine-specific implementations, all the code needed to read the Error File and load the internal memory arrays was written specifically for the Excello machine. However, care was taken to write robust code in order to provide a good template for other implementations. Along with the error tables, the Excello Error File contains the information found in Table 5-4. An example Excello Error File is included as Appendix B.

Table 5-4. Error File parameters

Error File parameter	Description
axCompCount[n]	Number of locations on an axis where error measurements were taken. Array variable corresponds to axis number
axCompSpacing[n]	Spacing between successive measurement locations in millimeters. Array variable corresponds to axis number
axCompPosMin[n]	Starting (most negative) location for first error measurement in millimeters. Array variable corresponds to axis number
axSquXZ	Squareness error between X-axis and spindle expressed in arc-seconds
axParZS	Parallelism error between Z-axis and spindle expressed in arc-seconds

5.3.2 Error Arrays

In Section 5.2.1 a generic array naming convention was described, and proposed array names were given in Table 5-3. One purpose of the naming convention was to aid the developer in placing the correct error data in the

proper array. Another was to develop an approach that would be consistent for different machine types and extendable to machines with more than three axes.

In the Excello GECS implementation, all of the arrays listed in Table 5-3 were not needed since the lathe is only a 2-axis machine. The error tables required are given by the machine's error equations, which were given as Equations (4-31) and (4-32). These are shown in Table 5-5. Of the linear error arrays, only scale and horizontal straightness were applicable to the Excello, since vertical straightness errors could not be compensated and were intentionally left out of the error equation development. The only angular error tables needed were the yaw tables, again because pitch and roll errors could not be compensated.

Table 5-5. Excello error array names

Linear arrays		Angular arrays
axLin00DirPos	axLin11DirPos	axAng20DirPos
axLin00DirNeg	axLin11DirNeg	axAng20DirNeg
axStr10DirPos	axStr01DirPos	axAng21DirPos
axStr10DirNeg	axStr01DirNeg	axAng21DirNeg

5.3.3 Units

Another issue in the design of the GECS architecture related to the units used for the error tables written in the Error File and the error arrays used in the GECS software. As described in Section 3.1.2, OpenCNC uses its own units of measurement. So before OpenCNC could use any error values they would have to be converted to the OpenCNC units of measurement. Since these units are unique to OpenCNC, they are not readily interpreted by humans and thus would not be the best choice for the Error File data. It was felt that the information stored should be in units typically used to quantify machine tool errors.

Therefore, in the Error File linear errors were stored in millimeters, and angular errors were stored in arc-seconds.

Based on discussions with MDSI personnel, it was felt that the GECS implementation would be more straightforward if floating point arithmetic were used within the compensation system. The complication of using integer arithmetic was seen as a hindrance in the development and debugging of the system. Once the code has been written and is running correctly, an effort could be made to determine if efficiency improvements provided by integer arithmetic were needed.

Since the OpenCNC linear unit of measurement is 0.00001 millimeter, using millimeters for GECS computations would allow easy conversion to distance units. So all GECS linear error arrays store values in millimeters. Angular data storage was more complicated. Anytime an angular error motion (designated by ϵ) is specified in an error equation of the form of Equation (4-31), it is multiplied by some linear parameter, which produces a linear error component. For the multiplication to produce the desired quantity, the angle must be specified in radians. So as the angular data in the Error File is read into system memory during GECS initialization, the values are converted to radians.

5.3.4 Conversion Utility

The 5-D Laser Measuring System produces five error tables in three separate files for each measured axis. Each table contains data for each measurement location for both the forward and reverse directions. Because of the potential for errors in translating this large quantity of error data, a conversion utility was created to automatically translate the 5-D data into the Error File. This utility was created using Microsoft's Visual C++[®] and was designed to run under the Windows 95[®] operating system. The dialog-based graphical user interface (GUI)

is shown in Figure 5-3. It was intended that the converter be run on the laptop computer that controlled the 5-D system; the newly created Error File could then be transferred to the OpenCNC controller computer, irregardless of whether OpenCNC was being run under Windows NT or QNX.

The conversion utility was written specifically for converting Excella data files and therefore only supports two axes. As was the case in the LSEC System, the GECS assumes the data in the error tables begins with measurements at the most negative end of the slide and continues in the positive direction. The Error File follows this convention as well. In contrast, the 5-D data is written based on

Figure 5-3. Conversion utility user interface

forward and reverse motions of the sensor relative to the laser. As shown in Figure 5-3, the laser data conversion utility uses information provided by the user regarding the direction of forward motion and the most negative measurement location. It can then place the error data in the appropriate array, either *DirPos* or *DirNeg*. It should also be noted from Figure 5-3 that the user manually inputs the Parallelism and Squareness values along with the manually measured constants that correspond to the Excella error model. The name of the resulting Error File is also specified.

5.3.5 Source Code Description

The GECS is implemented as three separate source code files written in the C programming language. These files were created by MDSI using the OpenCNC API and were provided as the previously mentioned external compensation template. The first file, *geoComp.c*, provides the interface between GECS and OpenCNC and controls the overall compensation process. A flowchart depicting its execution is shown in Figure 5-4.

GeoComp sets up the shared memory for getting information in and out of OpenCNC, initializes the compensation calculation routine, and sets the GECS process timer so execution occurs at regular intervals. The default interval was set to 10 ms. It then enters an infinite loop in which it obtains axis position and direction information, converts the positions to millimeters, and passes the information to the compensation calculation subroutine *geoWork*. Once it receives the compensation values, it converts them to OpenCNC's linear distance units and passes them to OpenCNC, which positions the axes accordingly. Few modifications to this function from its initial state was received from MDSI.

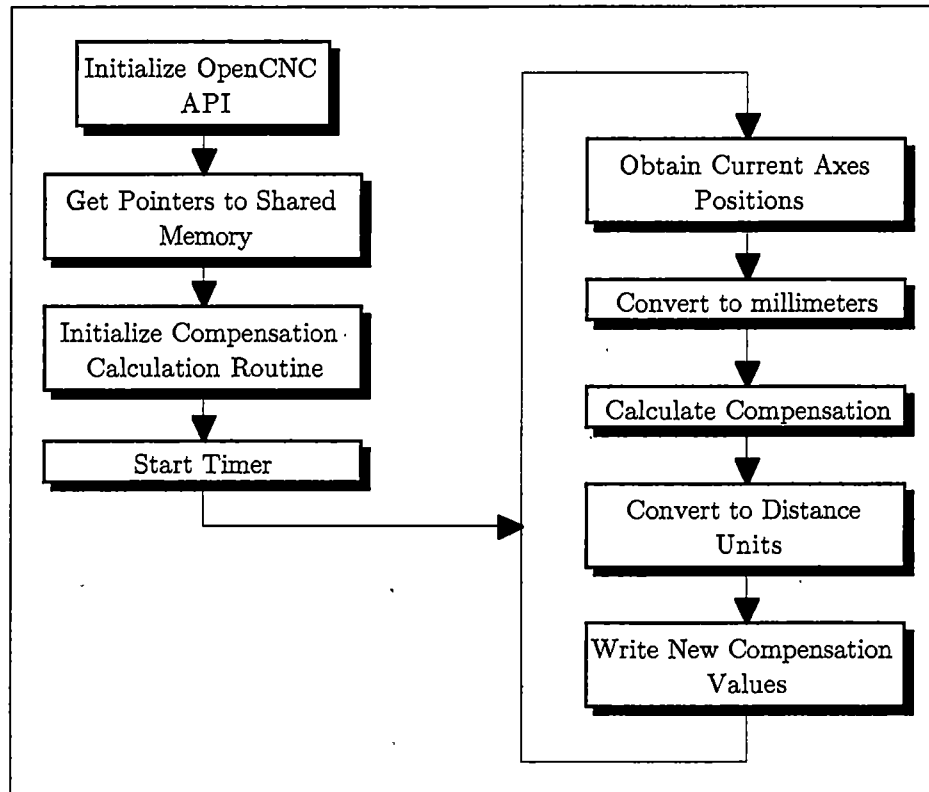


Figure 5-4. GeoComp flowchart

As its name implies, the second source file, `geoWork.c`, performs the actual work of calculating the axis compensations. As received from MDSI this file contained placeholders for two subroutines, one for GECS initialization and one for calculating compensation values. Both of these were developed and implemented during this project. The source code for these functions is included as Appendix D. The initialization subroutine contains the code that parses the Error File and loads the error arrays. The second subroutine, called `calcComp`, contains the code that implements the Excello error equations, given as Equations (4-31) and (4-32). A flowchart of this subroutine is shown in Figure 5-5. Data provided to `calcComp` include the positions of each axis in millimeters and the signs of their velocities (directions of motion).

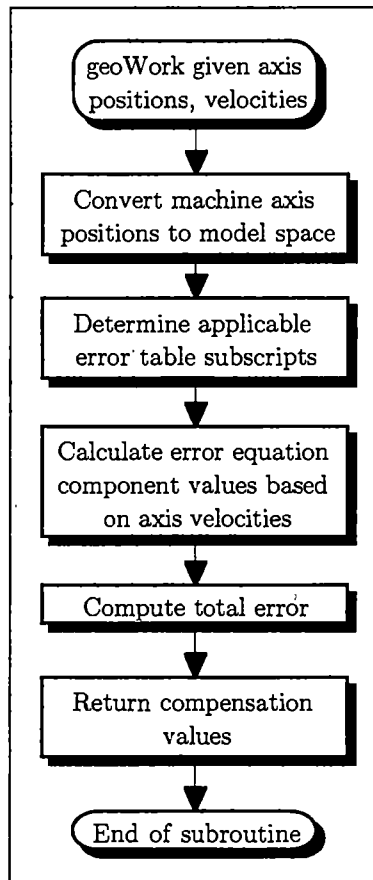


Figure 5-5. Compensation calculation flowchart

The first step in calcComp is to convert the given machine positions into model space coordinates as depicted in Figure 4-5. Once the position data is converted to model space, the appropriate error table subscripts can be determined. Next, the error equation components are calculated, using data from the appropriate *DirPos* or *DirNeg* arrays, depending on the slides' directions of motion. Linear interpolation is used to calculate values not falling directly on one of the measured locations. The error terms are then combined using the error equations to give the tool's total errors. The needed compensations are in the directions opposite the errors, so the signs of the errors are changed before passing the results back to geoComp.

Chapter 5. System Implementation

A test was conducted to determine the amount of CPU time required to perform the compensation calculations in geoWork. A software routine was written that called geoWork 100,000 times and calculated the average amount of elapsed CPU cycles for each call. This routine was run on the Excella controller while the machine was under OpenCNC control to simulate more realistic operating conditions. The average number of clock cycles required for a compensation calculation was 70, which equated to a clock time of 70 μ sec. This amount of time is three orders of magnitude less than the default compensation interrupt interval of 10 ms, indicating that using floating point arithmetic is adequate for the GECS implementation on the Excella lathe.

The third file comprising the GECS is called winGeo.c, and its purpose is to provide the operator with feedback regarding the amount of compensation currently being applied. It also provides the ability to turn geometric compensation off and on.

Chapter 6

System Evaluation

Once the error model was developed, the error equations implemented within OpenCNC, the error measurements completed, and the data integrated into the GECS, it was time to evaluate the system to see if the machine could position its axes more accurately than before and to attempt to quantify that improvement. Given the vast amount of measured data involved, the potential for sign errors was very high, and since the computed compensation was the combination of several error components, quickly identifying the erroneous data would have been nearly impossible.

The preferred test for this kind of system evaluation would be an actual machining test. A test part would have been designed that incorporated physical features that isolated some combinations of the geometric errors of the machine tool. Cylindrical surfaces would magnify errors in Z-axis straightness and parallelism, flat faces would magnify errors in squareness and X-axis straightness, and contours would isolate positioning errors in the axes, for example. A NC program would be written and tested to machine the part, and appropriate tooling and fixturing would be prepared along with part material. Two identical parts would be machined, one with error compensation disabled and one with

error compensation enabled. The two parts would then be inspected using equipment suitable for axi-symmetric parts, preferably a Coordinate Measurement Machine, which would have required another NC program to be developed and tested. Unfortunately, project funding was not sufficient to support these activities. Instead, a positioning test was needed that did not require any machining operations but that would adequately quantify the effects of the GECS.

When considering what type of positioning test to employ, several possibilities were explored. The first was a simple repeat of the positioning tests used to characterize the machine's positioning errors. It would have been simple to implement given that the laser equipment was already set up and the NC programs already created. The disadvantage of this type of test was that it would not show the compensations of one axis due to the errors of the other. For instance, the error equations for the X-axis contain components due to errors of the Z-slide such as straightness. The effect of these compensations would not be seen during a laser check of the X-axis because none of the laser equipment would be mounted on the Z-slide.

A second candidate was to use the laser system to perform a diagonal positioning test on the machine. In this test, the axes would be positioned at their limits, and the 5-D sensor would be mounted to the spindle faceplate as in the Z-axis tests, but oriented towards the tool holder. A spherical mirror would be mounted in the tool holder so that the laser beam was directed toward the sensor. A NC program would be created that positioned the axes so that the sensor and mirror vectored toward each other, stopping at specific intervals for position measurements. This test is used many times to assess the overall positioning accuracy of a machine tool. However, its use as a diagnostic tool is limited because it cannot be used to determine what geometric error component

is contributing the most to the position error. This test would be used for GECS evaluation only if other alternatives were not available.

The final test considered was a contour positioning test. In this type of test, some type of instrument is usually mounted onto the tool holder, and the machine slides are programmed to follow a given path, sometimes following the contour of a part whose geometry is well known. By comparing the commanded position to the known actual geometry, the position error could be determined. This type of test was chosen for the GECS evaluation and will be described in the next section.

6.1 Two-axis Contour Test

The ASME B5.57 Lathe Evaluation Standard provides procedures for performance evaluation of NC lathes and turning centers. It contains equipment specifications and testing procedures to facilitate performance comparisons between machines. One section of the standard is devoted to contouring performance using circular tests. Circular contours are used because they "provide one of the best checks for contouring performance in that as a machine is traversing with multiple axes along a circular trajectory each axis goes through sinusoidal acceleration, velocity, and position changes [19]." Various means are described for implementing these circular contour tests, including ballbars, grid plates, and precision disks.

6.1.1 Description

A Renishaw® ballbar was used for this evaluation. The ballbar consists of a high accuracy displacement sensor housed inside a telescopic bar. The sensor contains two coils and a moving core and works similarly to Linear Variable

Differential Transformers (LVDTs). As the bar changes length, the core moves inside the coils, changing their inductance. This inductance is measured electronically and converted to displacement readings. The transducer has a precision sphere on one end and a kinematic mount on the other. This mount houses a strong magnet that is used to hold a second ball in place. These two balls allow the transducer to rotate in any attitude. In the case of the Excello, one of the balls was mounted on the spindle faceplate and one on the tool holder. A picture of the test setup is shown in Figure 6-1.

According to the ballbar user's guide [21], the transducer has a nominal length of 150 mm between ball centers and a stroke of ± 2.5 mm. Extension bars

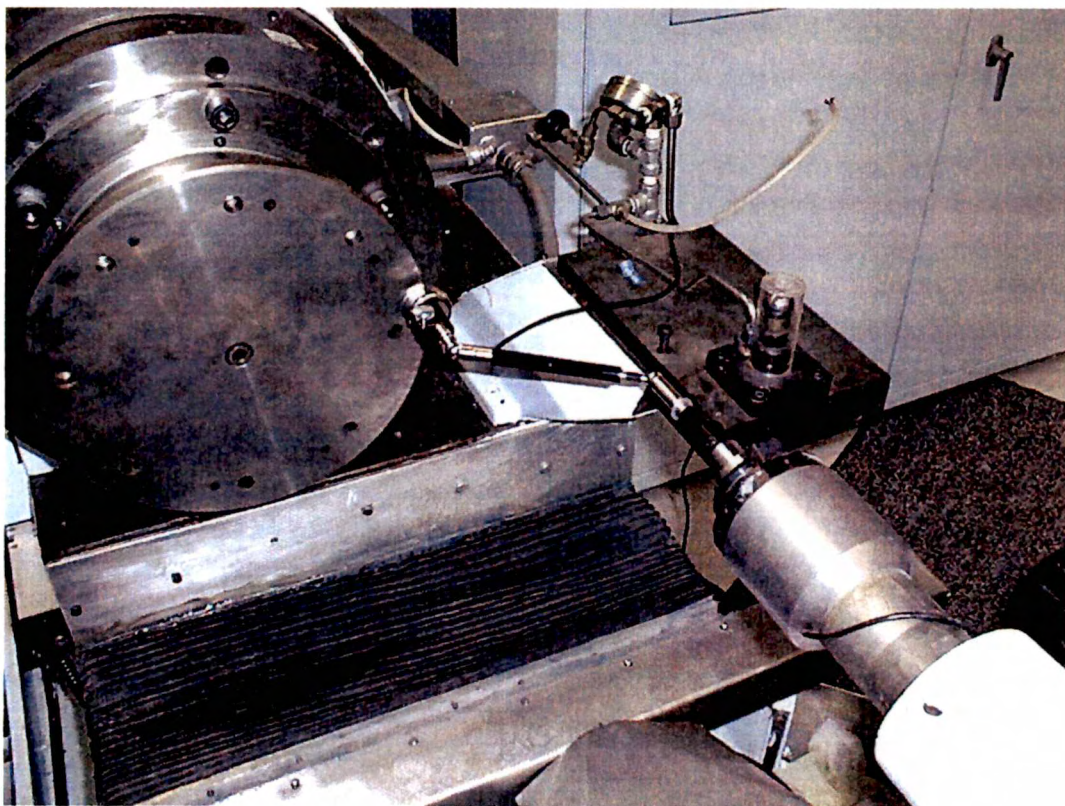


Figure 6-1. Ballbar setup

are supplied with the system to allow for longer nominal lengths. The actual measurement range of the transducer is ± 0.8 -mm minimum. Its measuring resolution is 0.1 micron, and its accuracy is advertised as ± 1.0 micron. The transducer is capable of sampling data at a frequency of 300 Hz.

The ballbar can be used for both static and dynamic tests. Static tests provide information on the geometric accuracy and repeatability of a machine. In this type of test, point locations around a circle are programmed into a toolpath. The machine is commanded to move to each location and pause for the ballbar reading to be recorded. When all data has been collected, the system displays positioning information. This type of test has advantages in that effects of feedrates, following error, and other properties of the control system are not observed. Dynamic tests providing information on the contouring ability of a machine, including backlash, axis squareness and straightness, servo behavior, vibration, etc. In this type of test, a circular toolpath is sent to the machine controller, and ballbar length information is obtained continuously throughout the motion. The ballbar software provides a plot of the readings, along with circularity information. In this project, it was desired to see if the static positioning capability of the machine could be improved, so that was the test that was conducted.

The B5.57 standard indicates that a full, 360° trace in the XZ-plane is preferred on machines that have the capacity to do so, although 180° traces are also described. The shortest length of the ballbar available was 150mm. If a 360° trace were desired, a 300-mm diameter circle would be swept by the bar, which required a range of motion that the Z-slide could not meet. Such a sweep would also have required some special fixturing to be used to hold the stationary cup away from the spindle faceplate. It was possible, however, to perform a 180° trace by placing the stationary cup as shown in Figure 6-1. During testing, as the

machine's axes moved, the ballbar rotated in the XZ-plane. If perfect motion had occurred, there would have been no change in the length of the ballbar. As the motion progressed, the ballbar measured the positioning errors of the slides and provided a diagram of the deviations.

For this test, a simple program was created to move the slides in a 180° arc, stopping at five locations around the arc. To remove any affects of the machine's reversal errors while conducting static positioning tests, the recommended NC program includes intermediate stopping locations before each measurement location. The stopping locations are positioned so that all measured locations are approached from the same direction. The program would be run with the GECS functionality turned off and on, and the results of the two runs would be compared. The feedrate between measurement locations was set at 300 mm/pm in the NC program.

6.1.2 Results

In performing the evaluation of the GECS, no repeatable data was taken that conclusively showed positioning improvements being provided by the GECS. Both static and dynamic tests were conducted, at different feedrates, with GECS enabled and disabled. Sometimes the tests showed improved circularity data, and sometimes they did not. The results of a static ballbar test as provided by the ballbar software is shown in Figure 6-2. Circularity is the quantitative parameter returned by the system. It computes this value by simply taking the difference between the maximum and minimum deviations from the least-squares best-fit arc through the measurement data.

In this test, three clockwise runs were performed in which ballbar readings were taken at five locations. The plot shows the position of each reading relative to the best-fit circle. The circularity computed for this case was 14.3 microns

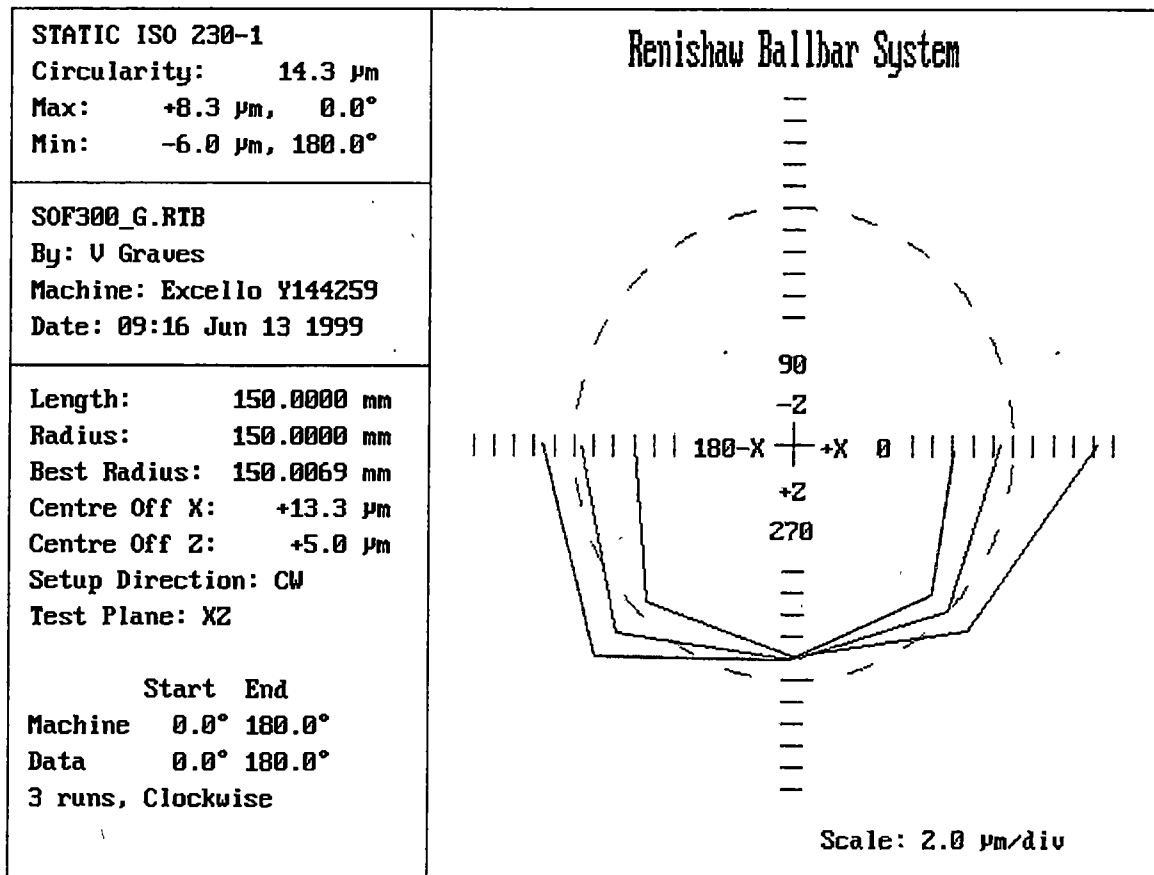


Figure 6-2. Static ballbar results with GECS disabled

(0.56 mils), which gives an indication of the positioning accuracy of the machine. Of some concern is the trend shown in Figure 6-2. The three curves were taken sequentially, and it appears that each curve is rotating around the point on the Z-axis. It is suspected that additional readings would continue this trend, which would have caused the circularity value to increase. The worst repeatability as reported by the ballbar system for this test was 14.0 microns (0.55 mils), and it occurred at the point on the +X side of the arc.

An identical test was run with the GECS enabled, and the results are shown in Figure 6-3. While the actual circularity value did show an 8% improvement, these results were not always observed. The worst repeatability of this test was

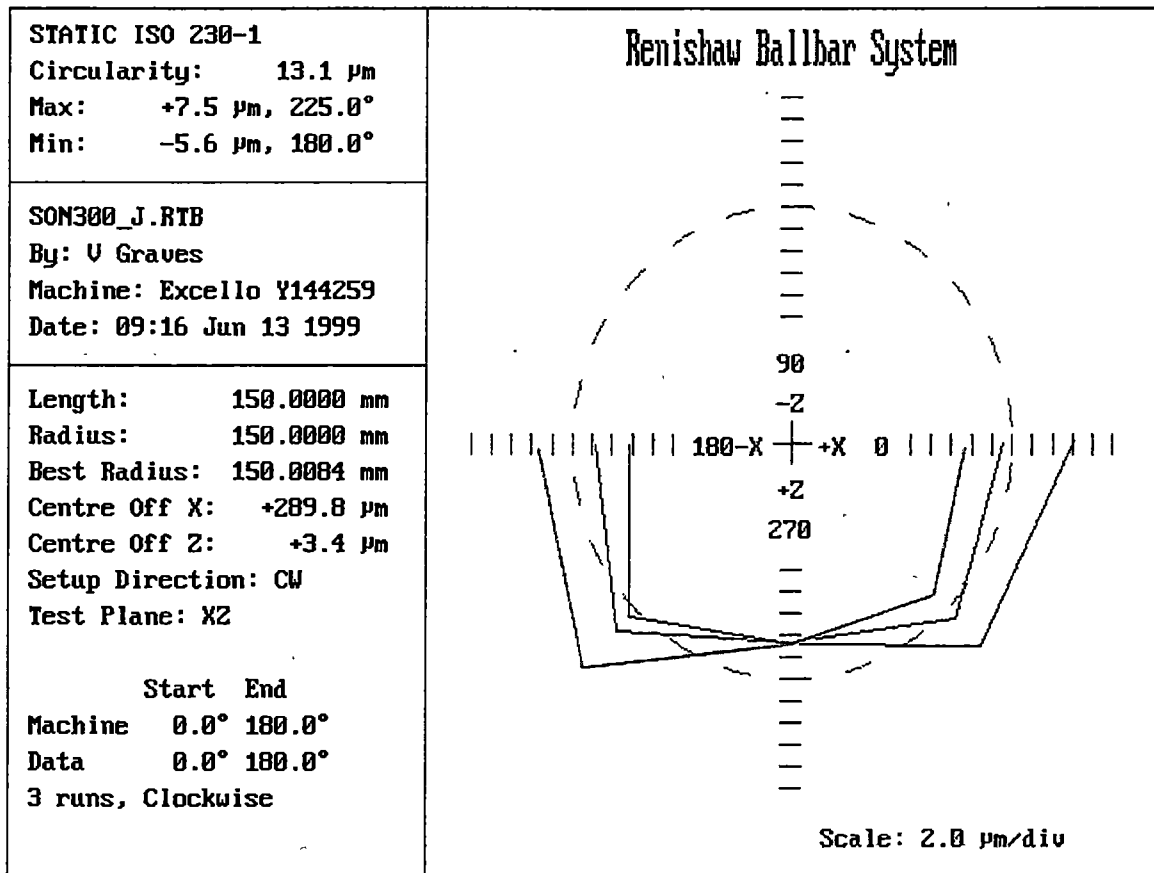


Figure 6-3. Static ballbar results with GECS enabled

10.8 microns (0.43 mils), also an improvement from the earlier test. Since the GECS is providing compensation values based on axis positions and velocities, there is no logical reason to believe that the repeatability of the machine should improve, only the accuracy. So the repeatability “improvement” should be treated with skepticism. Also of note was the fact that the same type of data rotation trend was also observed in this test, which would indicate that additional runs would have produced worse readings.

While investigating the possible causes of the inconsistent results, an analysis of the component values that made up the X and Z compensations was performed. The actual axis positions and directions of travel for a semicircular

ballbar sweep were passed to the compensation calculation function in a simulated movement. The same Error File was used for both the simulated motion and the actual machine tool motion. Each term of the error equations (4-31) and (4-32) was computed and stored for every 5° of angular motion around the arc. The results for the X-axis are shown in Figure 6-4.

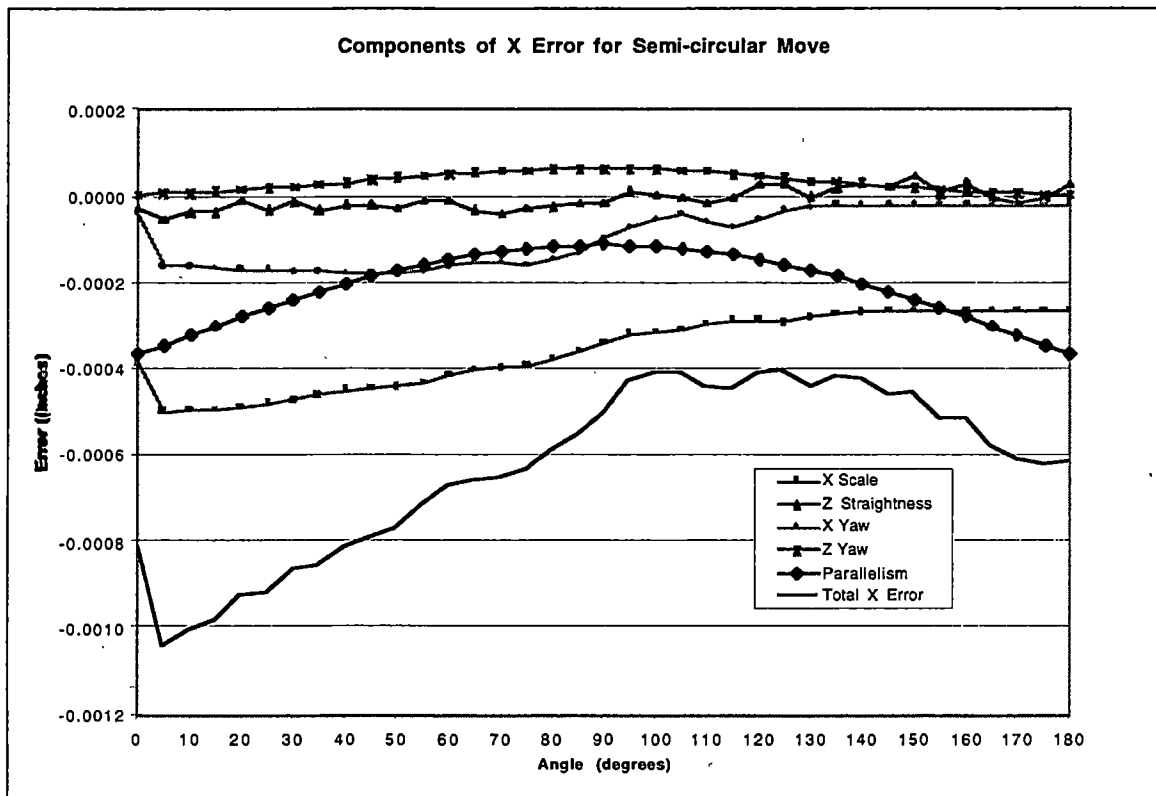


Figure 6-4. X error components in semicircular move

The figure shows the actual error component values, with the appropriate algebraic sign as assigned in Equation (4-31), at each angular location along the arc. The Total X Error is the algebraic sum of all the components and represents the magnitude of the axis compensation that would be applied. In the case of the X-axis, the errors that contribute most to the total error are the scale error and

the parallelism error. The least significant contributor is the straightness of the Z-axis. Figure 6-5 shows the graph for the Z-axis error components.

From Figure 6-5, it is apparent that the squareness error makes the most significant contribution to the total error, and subsequent compensation, of the Z-axis. This is due to the amplification of the small angular error by the long moment arm provided by the range of X-axis motion, which is the distance of the X-slide from the most negative end of the axis. If there were any errors in the measurement and calculation of the XZ-squareness value, the effects were obviously severe.

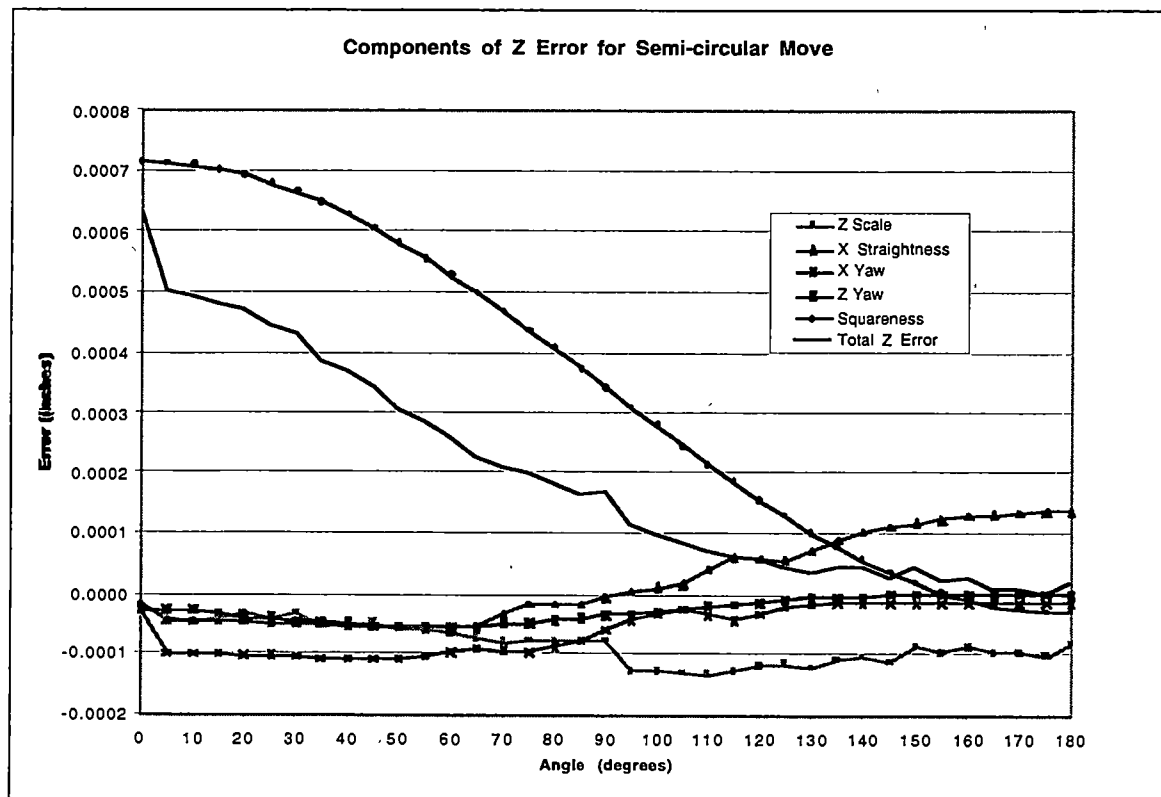


Figure 6-5. Z error components in semicircular move

Because of the significant error contributions made by the squareness and parallelism measurements, some final ballbar tests were conducted in which these components were removed from the error equations in the GECS. Again, results were inconsistent when comparing circularity measurements with the GECS enabled and disabled. Unfortunately, time constraints prevented further investigation.

6.2 Conclusions

It was hoped that a definite improvement in positioning accuracy would have been observed when the GECS was enabled. While improvements were observed, the results were not consistently repeatable. The technique applied in this project relies on having a machine tool which has repeatable errors. Based on the results of the ballbar test the lathe's positioning repeatability was larger than the small compensations that were being applied. The test-bed Excella lathe had not been used for part manufacture in several months prior to the MDSI controller retrofit and was in need of mechanical maintenance work. If the machine was in optimum operating condition, perhaps the results would have been different.

With regard to its other goals, the project was very much a success. The GECS architecture was developed and can serve as the basis for future implementations on other types of machines. While a lathe served as an adequate test-bed, a three-axis milling machine would receive the greatest benefit from this type of compensation system given that it could compensate for more types of errors than a two-axis lathe. The techniques needed to develop the error equations were learned during this project, which was the first in which in-house personnel implemented geometric error correction. The knowledge gained during this process will be extremely beneficial in future work and discussions related to precision machining operations. Finally, by interfacing an external application

with OpenCNC, the openness of the PC-based controller was verified. Very few, if any, other commercial controllers provide a mechanism for actually changing the commanded position of a machine's slides. It is felt that other types of compensations could easily be implemented using this capability.

6.3 Future Work

This initial implementation of the GECS showed that it is feasible to implement geometric error compensation software in a commercially available machine tool controller without having to interface through the machine position feedback electronics. The fact that it was implemented within a commercially available controller only adds to its value and vastly increases the number of machine tools to which it may be applied. Based on this initial effort, the potential exists for future work in several different areas.

Implementing the GECS on machines with differing geometric configurations is a natural extension of this work. While other types of lathe implementations would be useful, three-axis milling machines would probably be the most needed implementation, based on their widespread usage within the general manufacturing industry. Of the various machine configurations shown in Figure 1-1, the majority of three-axis mills found in use are XYFZ machines. New GECS implementations would require the same steps used in the Excello implementation: (1) an error model would have to be developed to determine which error components needed to be measured; (2) an Error File format would have to be developed; (3) software would have to be written to read the Error File and apply the data according to the error equations; (4) machine error characterization measurements would have to be made to populate the Error File; and (5) the system would have to be tested. The most difficult part of a

new implementation would be the development of the error model itself; much of the remaining work could be done by modification of the Excella source code.

Of great benefit to the Y-12 Plant would be the incorporation of the GECS on its remaining Excella lathes. The error model and measurement techniques have been defined, and the compensation software has been developed, so any Excella implementation would require only that each machine be characterized. Unfortunately, only the GECS test-bed lathe uses the OpenCNC controller. The remaining Excella machines still utilize the GE2000 controllers, which do not directly support geometric compensation techniques. They would require separate computers running the GECS code that interfaced through the electronics of the position feedback system. There is an effort within Y-12 to survey the commercial machine tool controller market for possible replacements for the GE controllers, and the MDSI controller is being considered. It may be beneficial to require any new controller to have the capability of supporting these types of compensations if geometric error compensation is not directly supported.

By having machine tool controllers capable of supporting geometric as well as other types of error compensations, a new field of further research would be opened in the area of rapid machine characterization. Acquiring a machine's error data to fill the software error tables becomes more important as controllers are able to actually use the data to improve a machine's performance. As tools such as the 5-D Laser Measuring System continue to decrease the time and expense required to characterize a machine tool, the ability to monitor the operating condition of the machine dramatically improves. It would be possible to incorporate this type of characterization data into a preventive maintenance program, and by trending the data one could foresee its usefulness as a predictive maintenance tool.

The area that would seem to have the most potential for future work is the development of other types of compensation systems. The external compensation template provided by MDSI is not limited only to geometric error compensation; it was developed as a general-purpose interface for external applications of any type. For the GECS, it passed axis position and direction information to the compensation calculation subroutine and received in return the compensation values for each axis. Any other data within the OpenCNC controller relating to the operating state of the machine would be available to an external application. In addition, the compensation application could be written to interface to sensors and hardware that were totally external to OpenCNC. Several possible compensation systems could be implemented with this type of interface. One example would be affixing thermocouples in strategic locations of the machine frame, monitoring the temperature of the machine, and using neural network techniques to compensate for the thermal growth of the system. Another would be to interface some type of vision system that periodically inspected the tool tip for wear and provided the necessary tool path compensations. As a third example, accelerometers could be installed on the spindle to monitor for conditions that might lead to part inaccuracies or tool breakage, such as chattering. Having this kind of access to the inner workings of a machine tool controller, combined with the ability to interface with any kind of external application, provides a previously unheard-of mechanism to explore almost any aspect of precision machining.

REFERENCES

-
- [1] OMAC Users Group, "Requirements of Open, Modular Architecture Controllers for Applications in the Automotive Industry," Version 1.1, Dec. 13, 1994.
 - [2] R. Hocken, "Technology of Machine Tools, Vol. 5: Machine Tool Accuracy," Report of the Machine Tool Task Force, UCRL-52960-5, 1980.
 - [3] "Software Correction of Precision Machines," A Report from Precision Engineering Laboratory UNC Charlotte to National Institute of Standards and Technology in fulfillment of Contract No. NIST-60NANB2D1214, Jan. 16, 1995.
 - [4] R. Hocken, J. Miller, "Machine Tool Metrology," University of North Carolina, Charlotte, Spring Semester 1998 course notes.
 - [5] Conversation with Harold A. Fell, Y-12 Development Division.
 - [6] C. Evans, *Precision Engineering: an Evolutionary View*, Cranfield Press, Bedford, United Kingdom, 1989.
 - [7] J. Denavit, R.S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," in *Journal of Applied Mechanics*, June 1955, pp. 215-221.
 - [8] R. Hocken et al, "Three Dimensional Metrology," in *Annals of the CIRP*, Vol. 26/2, pp. 403-408, 1977.
 - [9] J.A. Soons, F.C. Theuws, and P.H. Schellekens, "Modeling the errors of multi-axis machines: a general methodology," in *Precision Engineering*, Vol. 7, pp. 5-19, 1992.
 - [10] M.A. Donmez, C.R. Liu, M.M. Barash, "A Generalized Mathematical Model for Machine Tool Errors," presented at The Winter Annual Meeting of the American Society of Mechanical Engineers, Anaheim, California, December 7-12, 1986.
 - [11] M.A. Donmez, D.S. Blomquist, R.J. Hocken, C.R. Liu, M.M. Barash, "A General Methodology for Machine Tool Accuracy Enhancement by Error Compensation," in *Precision Engineering*, Vol. 10, pp. 187-196, 1986.
 - [12] G. Zhang, R. Veale, T. Charlton, B. Borchardt, and R. Hocken, "Error Compensation of Coordinate Measuring Machines," in *Annals of the CIRP*, Vol. 34/1, pp. 445-448, 1985.

-
- [13] S. Sartori and G. Zhang, "Geometric Error Measurement and Compensation of Machines," in *Annals of the CIRP*, Vol. 44/2, pp. 599–609, 1995.
 - [14] K.F. Eman, B.T. Wu, and M.F. DeVries, "A Generalized Geometric Error Model for Multi-Axis Machines," in *Annals of the CIRP*, Vol. 36/1, pp. 253–256, 1987.
 - [15] J.R. Franks, "Open Architecture CNC for a 3 Axis Milling Machine," University of Tennessee, Knoxville, Master's Thesis, May 1998.
 - [16] A.H. Slocum, *Precision Machine Design*, Society of Manufacturing Engineers, Dearborn, Michigan, 1992.
 - [17] M.A. Donmez, C.R. Liu, M.M. Barash, "A Generalized Mathematical Model for Machine Tool Errors," in PED Vol. 23, ASME, pp. 231–243.
 - [18] "User Manual for the 5/6-D Laser Measuring System," Automated Precision, Inc., 7901-C Cessna Avenue, Gaithersburg, Maryland 20879.
 - [19] "ASME B5.57M Methods for Performance Evaluation of Computer Numerically Controlled Lathes and Turning Centers," Version 9.0, ASME B5/TC52 Committee, May 1997.
 - [20] "OpenCNC Integrator Manuals," Manufacturing Data Systems, Inc., 220 East Huron, Suite 600, Ann Arbor, Michigan 48104.
 - [21] "Renishaw Ballbar User Guide," Renishaw, New Mills, Wotton-under-Edge, Gloucestershire GL 12 8JR.

APPENDICES

APPENDIX A

EXCELLO ERROR EQUATION DEVELOPMENT

USING MATHEMATICA

■ Excello Error Equation Development

Input all HTMs. The double subscript notation is more convenient for Mathematica.

$$\text{In[1]:= } T01 = \begin{pmatrix} 1 & 0 & \varepsilon_{yz} & x_1 + \delta_{xz} + \beta z \\ 0 & 1 & 0 & 0 \\ -\varepsilon_{yz} & 0 & 1 & z + Z_1 + \delta_{zz} \\ 0 & 0 & 0 & 1 \end{pmatrix}; T02 = \begin{pmatrix} 1 & 0 & \varepsilon_{yx} & x + X_2 + \delta_{xx} \\ 0 & 1 & 0 & 0 \\ -\varepsilon_{yx} & 0 & 1 & Z_2 + \delta_{zx} - \alpha x \\ 0 & 0 & 0 & 1 \end{pmatrix}; T23 = \begin{pmatrix} 1 & 0 & 0 & x_t \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

Calculate Ttool (T03). The matrix multiplication expansion automatically removes all 2nd- and higher-order error terms. Break the HTM into its rotation matrix and its position vector for later HTM inverse calculations. Twork requires no calculation; it's defined as T01.

```
In[2]:= tmp03 = Expand[T02 . T23 /. \varepsilon_j_ -> a \varepsilon_j /. \delta_k_ -> a \delta_k /. \alpha -> a \alpha /. \beta -> a \beta];
Ttool = tmp03 /. a^(i_ /; i > 1) -> 0 /. a -> 1;
Rtool = Ttool[[{1, 2, 3}, {1, 2, 3}]];
Ptool = Ttool[[{1, 2, 3}, 4]];
MatrixForm[Ttool]
```

Out[2]/MatrixForm=

$$\begin{pmatrix} 1 & 0 & \varepsilon_{yx} & x + X_2 + X_t + \delta_{xx} + Z_t \varepsilon_{yx} \\ 0 & 1 & 0 & 0 \\ -\varepsilon_{yx} & 0 & 1 & -x \alpha + Z_2 + Z_t + \delta_{zx} - X_t \varepsilon_{yx} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
In[3]:= Twork = T01;
Rwork = Twork[[{1, 2, 3}, {1, 2, 3}]];
Pwork = Twork[[{1, 2, 3}, 4]];
MatrixForm[Twork]
```

Out[3]/MatrixForm=

$$\begin{pmatrix} 1 & 0 & \varepsilon_{yz} & z \beta + X_1 + \delta_{xz} \\ 0 & 1 & 0 & 0 \\ -\varepsilon_{yz} & 0 & 1 & z + Z_1 + \delta_{zz} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Calculate HTM inverse of Twork.

```
In[4]:= invWork = IdentityMatrix[4]; invWork[[{1, 2, 3}, {1, 2, 3}]] = Transpose[Rwork];
invWork[[{1, 2, 3}, 4]] =
Expand[-Transpose[Rwork] . Pwork /. \varepsilon_j_ -> a \varepsilon_j /. \delta_k_ -> a \delta_k /. \alpha -> a \alpha /. \beta -> a \beta];
invWork[[{1, 2, 3}, 4]] = invWork[[{1, 2, 3}, 4]] /. a^(i_ /; i > 1) -> 0 /. a -> 1;
MatrixForm[invWork]
```

Out[4]/MatrixForm=

$$\begin{pmatrix} 1 & 0 & -\varepsilon_{yz} & -z \beta - X_1 - \delta_{xz} + z \varepsilon_{yz} + Z_1 \varepsilon_{yz} \\ 0 & 1 & 0 & 0 \\ \varepsilon_{yz} & 0 & 1 & -z - Z_1 - \delta_{zz} - X_1 \varepsilon_{yz} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Calculate Erel, which gives total error equations. Pe is the position vector of Erel, which is all we are interested in.

```
In[5]:= tmperr = Expand[invWork . Ttool /. \varepsilon_j_ -> a \varepsilon_j /. \delta_k_ -> a \delta_k /. \alpha -> a \alpha /. \beta -> a \beta];
err = tmperr /. a^(i_ /; i > 1) -> 0 /. a -> 1; Pe = err[[{1, 2, 3}, 4]]
```

```
Out[5]:= {x - z \beta - X_1 + X_2 + X_t + \delta_{xx} - \delta_{xz} + Z_t \varepsilon_{yx} + z \varepsilon_{yz} + Z_1 \varepsilon_{yz} - Z_2 \varepsilon_{yz} - Z_t \varepsilon_{yz}, 0,
-z - x \alpha - Z_1 + Z_2 + Z_t + \delta_{zx} - \delta_{zz} - X_t \varepsilon_{yx} + x \varepsilon_{yz} - X_1 \varepsilon_{yz} + X_2 \varepsilon_{yz} + X_t \varepsilon_{yz}}
```

Collect terms for easier viewing.

```
In[6]:= MatrixForm[Collect[Collect[Collect[Collect[Pe, \delta_], \varepsilon_], \beta], \alpha]]
```

Out[6]/MatrixForm=

$$\begin{pmatrix} x - z \beta - X_1 + X_2 + X_t + \delta_{xx} - \delta_{xz} + Z_t \varepsilon_{yx} + (z + Z_1 - Z_2 - Z_t) \varepsilon_{yz} \\ 0 \\ -z - x \alpha - Z_1 + Z_2 + Z_t + \delta_{zx} - \delta_{zz} - X_t \varepsilon_{yx} + (x - X_1 + X_2 + X_t) \varepsilon_{yz} \end{pmatrix}$$

APPENDIX B

EXAMPLE GECS ERROR FILE

Appendix B

```
# Excello Error Data File
# Creation Date: 06/12/99 13:43:30

1      # Version Number
22     # axCompCount[ 0]
23     # axCompCount[ 1]
25.400 # axCompSpacing[ 0] (mm)
12.700 # axCompSpacing[ 1] (mm)
-533.400 # axCompPosMin[ 0] (mm)
0.000   # axCompPosMin[ 1] (mm)

-13.0   # axSqu01 (arc-sec)
8.9     # axPar1S (arc-sec)

723.900 # X1 (mm)
-756.920 # Z1 (mm)
406.400 # X2 (mm)
297.180 # Xt (mm)
-487.680 # Zt (mm)

# X Axis 5-D Information (Windows Data File)
# Software Version: 1.0.1
# Instrument(S/N): 3051
# File Name: C:\Program Files\API\API 5-6D Laser\Data Files\XLO X5.pos
# TestDate: 06/12/1999
# TestTime: 10:18:39
# MachID: Y144259 MachName: Excello Operator: van CNCProg: xabs3run10secnp
# FeedRate: 20 ipm
# AxesLoc: (Z) (Y)
# TestAxis: (X)

# X Axis Error Data (Axis 0)
#
# Lin00DirPos Lin00DirNeg Str10DirPos Str10DirNeg Ang20DirPos Ang20DirNeg
# mm mm mm mm arc-sec arc-sec
-0.003632 -0.006706 0.002540 0.003226 1.9 0.2
-0.005029 -0.006731 0.000381 0.002616 2.2 0.2
-0.006325 -0.007341 -0.000178 0.001473 2.4 0.3
-0.006579 -0.007264 -0.000483 0.001651 2.5 0.8
-0.006883 -0.007849 -0.001372 0.000381 2.8 0.4
-0.006985 -0.008103 -0.001168 0.000178 2.6 0.8
-0.007493 -0.009144 -0.001245 -0.000381 2.4 1.4
-0.008560 -0.009957 -0.001092 -0.000406 2.2 1.7
-0.008509 -0.010135 -0.001803 -0.001499 2.1 1.6
-0.008865 -0.011074 -0.001651 -0.001422 1.6 1.9
-0.009068 -0.011354 -0.001118 -0.001372 0.7 1.9
-0.009601 -0.012395 -0.000610 -0.001194 0.2 1.8
-0.010084 -0.013208 0.000076 -0.001016 0.9 1.6
-0.011684 -0.013970 -0.000660 -0.001118 0.2 1.5
-0.012725 -0.014148 0.000178 -0.000584 -0.1 1.4
-0.013310 -0.015392 -0.000305 -0.000533 -0.9 0.9
-0.016078 -0.016205 0.000229 -0.001067 -1.6 0.4
-0.018923 -0.018110 0.001270 -0.000381 -2.5 0.4
-0.020523 -0.021133 0.001651 -0.000584 -2.5 0.1
-0.022073 -0.021996 0.001067 0.000203 -2.7 -0.5
-0.022911 -0.023444 0.001702 0.000660 -3.4 -1.0
-0.024765 -0.024054 0.002388 0.001397 -2.1 -1.4
```

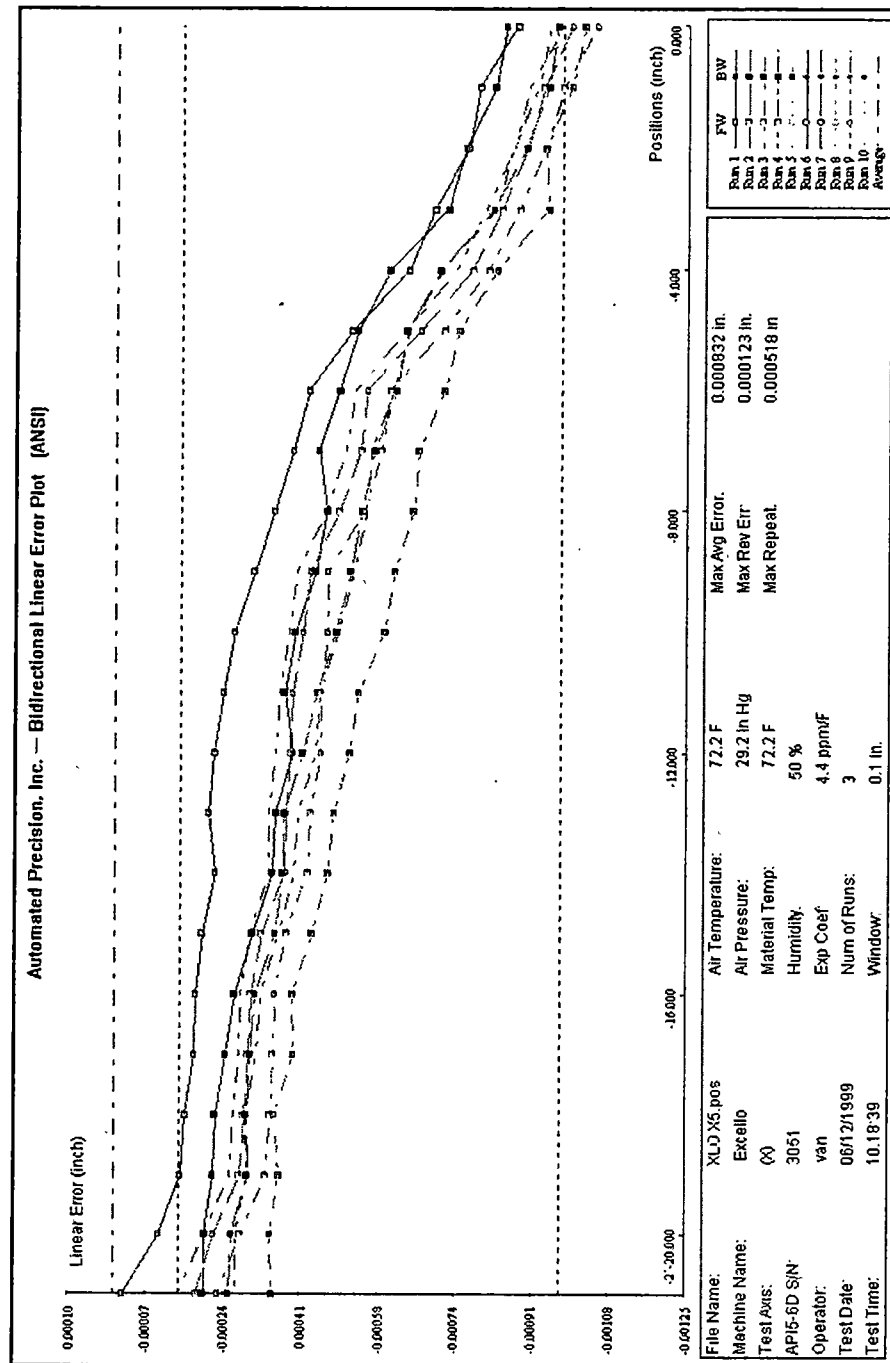
Appendix B

```
# Z Axis 5-D Information (Windows Data File)
# Software Version: 1.0.1
# Instrument(S/N): 3051
# File Name: C:\Program Files\API\API 5-6D Laser\Data Files\XLO Z11.pos
# TestDate: 06/12/1999
# TestTime: 11:36:46
# MachID: Y144259 MachName: Excelllo Operator: van CNCProg: zabs.53run10sec
# FeedRate: 20 ipm
# AxesLoc: (X) -9.3607 (Y)
# TestAxis: (Z)

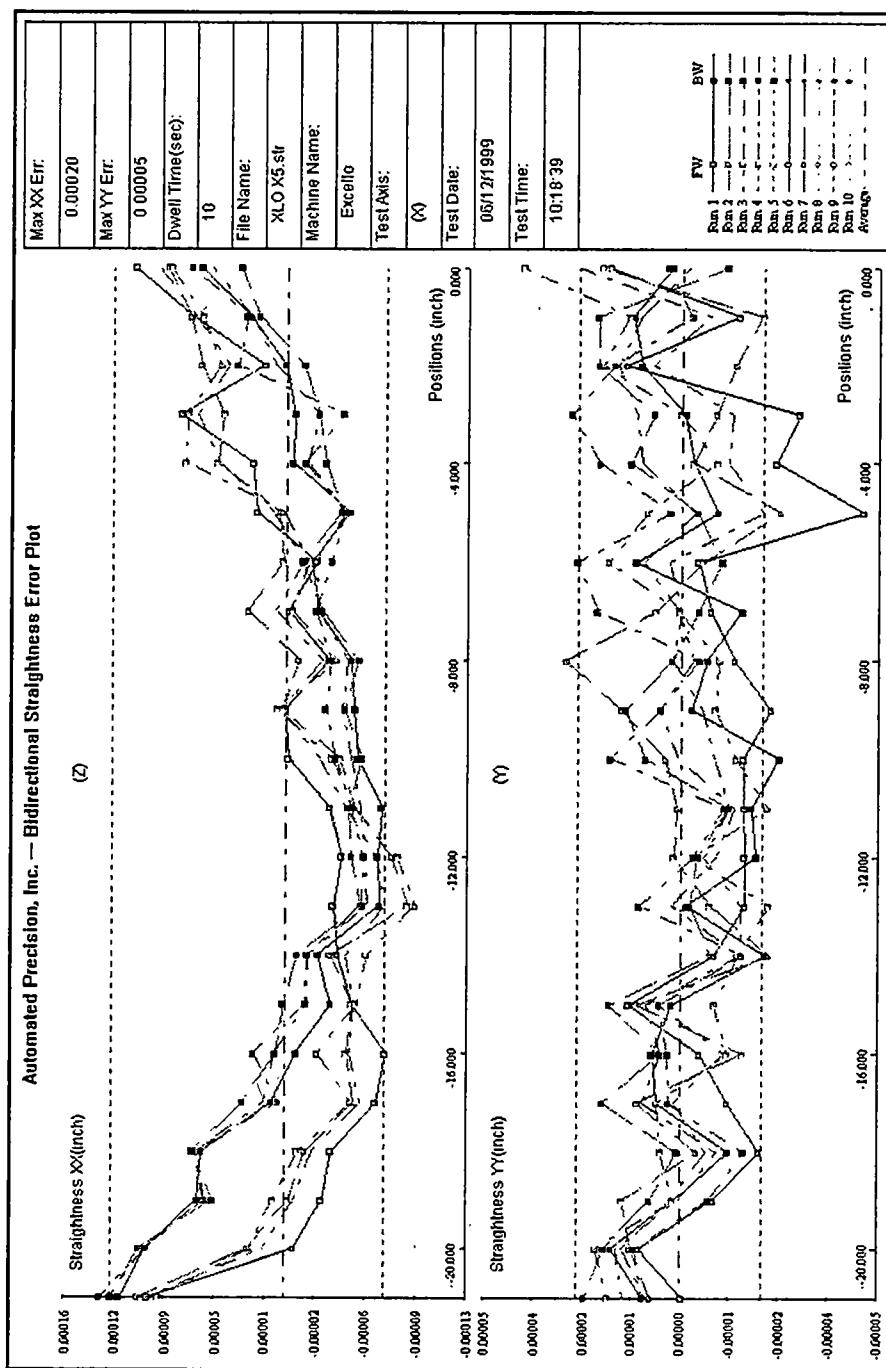
# Z Axis Error Data (Axis 1)-
#
# Lin11DirPos Lin11DirNeg Str01DirPos Str01DirNeg Ang21DirPos Ang21DirNeg
# mm mm mm mm arc-sec arc-sec
0.003480 0.002515 -0.001092 -0.001016 -3.7 -3.8
0.003454 0.002210 -0.000686 -0.000406 -3.9 -3.6
0.003251 0.002057 -0.000762 0.000203 -3.8 -3.5
0.002997 0.001905 -0.000178 0.000305 -3.8 -3.5
0.003531 0.002134 -0.000025 0.000838 -3.8 -3.5
0.003200 0.001956 -0.000178 0.000737 -3.8 -3.4
0.003378 0.002007 0.000076 0.001473 -3.4 -3.4
0.002896 0.001499 -0.000305 0.000305 -3.5 -3.2
0.003099 0.001549 0.000127 0.001118 -3.4 -3.0
0.002489 0.001041 -0.001270 0.000483 -3.4 -3.1
0.003048 0.001524 -0.000229 0.001041 -3.3 -3.0
0.002159 0.000813 -0.001600 0.000432 -3.4 -2.7
0.002464 0.001143 -0.000457 0.001600 -3.2 -2.6
0.002210 0.000864 -0.001016 0.000940 -3.2 -2.6
0.002515 0.001168 -0.000610 0.001702 -3.2 -2.4
0.002464 0.001143 -0.000965 0.000914 -3.2 -2.3
0.002565 0.001041 -0.000076 0.002388 -3.0 -2.3
0.002083 0.000483 -0.001041 0.000610 -3.1 -2.2
0.002184 0.000737 -0.001295 0.001067 -3.1 -2.3
0.001575 0.000203 -0.001575 0.000737 -2.8 -2.1
0.001626 0.000406 -0.001118 0.000610 -2.6 -1.8
0.001118 -0.000025 -0.001397 -0.000025 -2.9 -1.8
0.001168 0.000203 -0.001245 0.000864 -2.9 -1.5
```

APPENDIX C

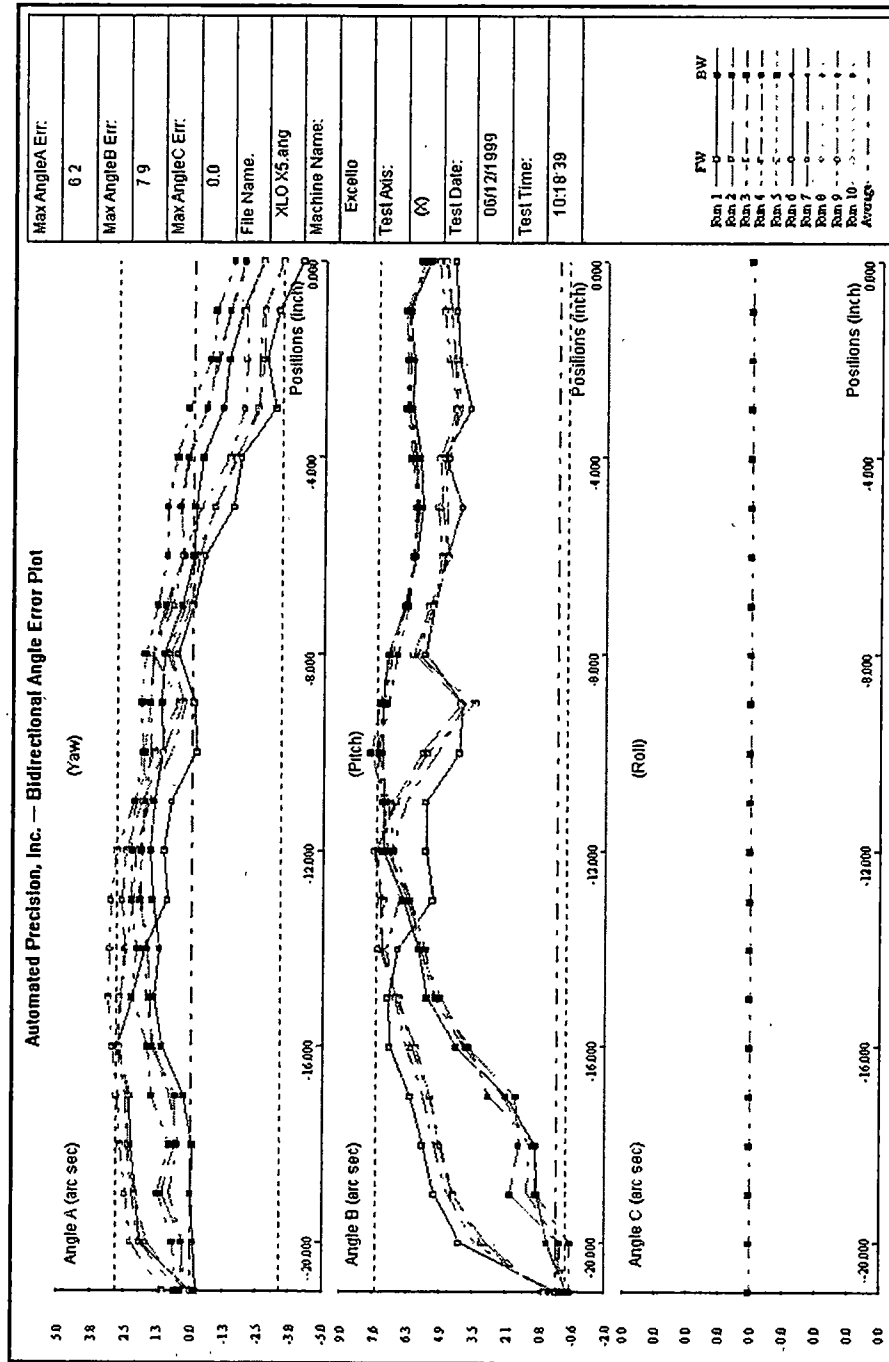
**EXCELLO 5-D LASER ERROR
MEASUREMENT DIAGRAMS**



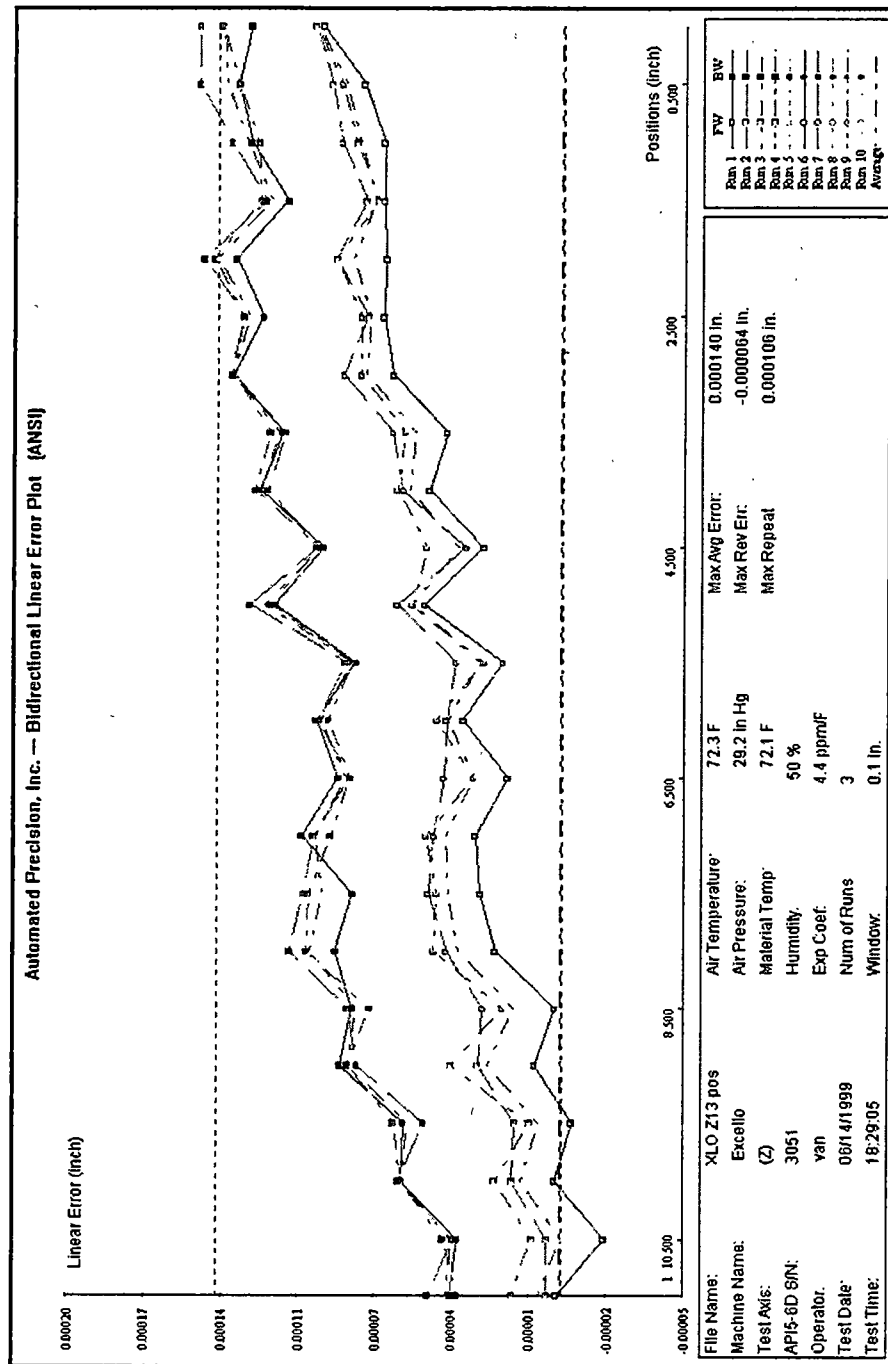
X-AXIS LEAD SCREW ERRORS



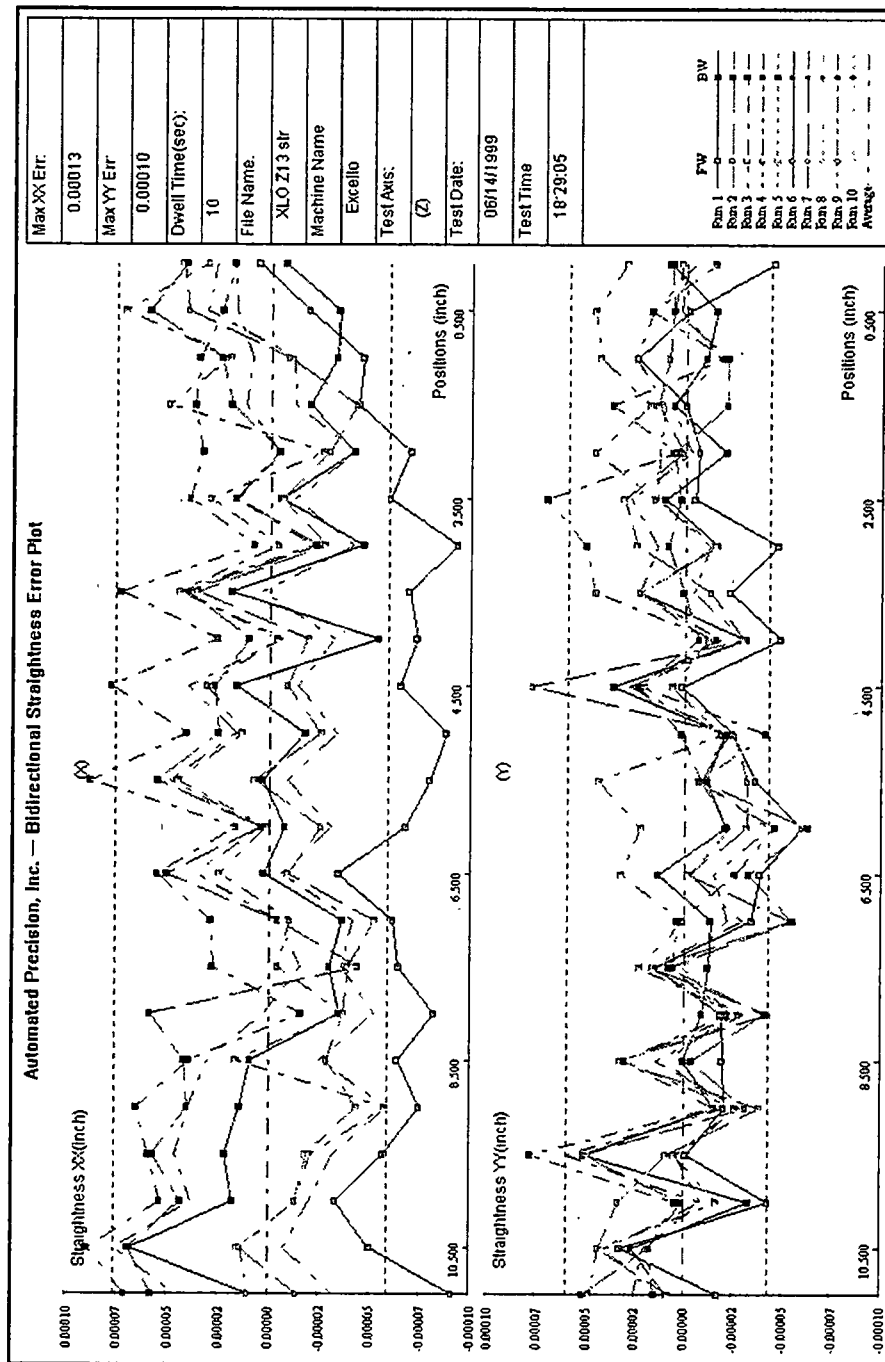
X-AXIS STRAIGHTNESS ERRORS



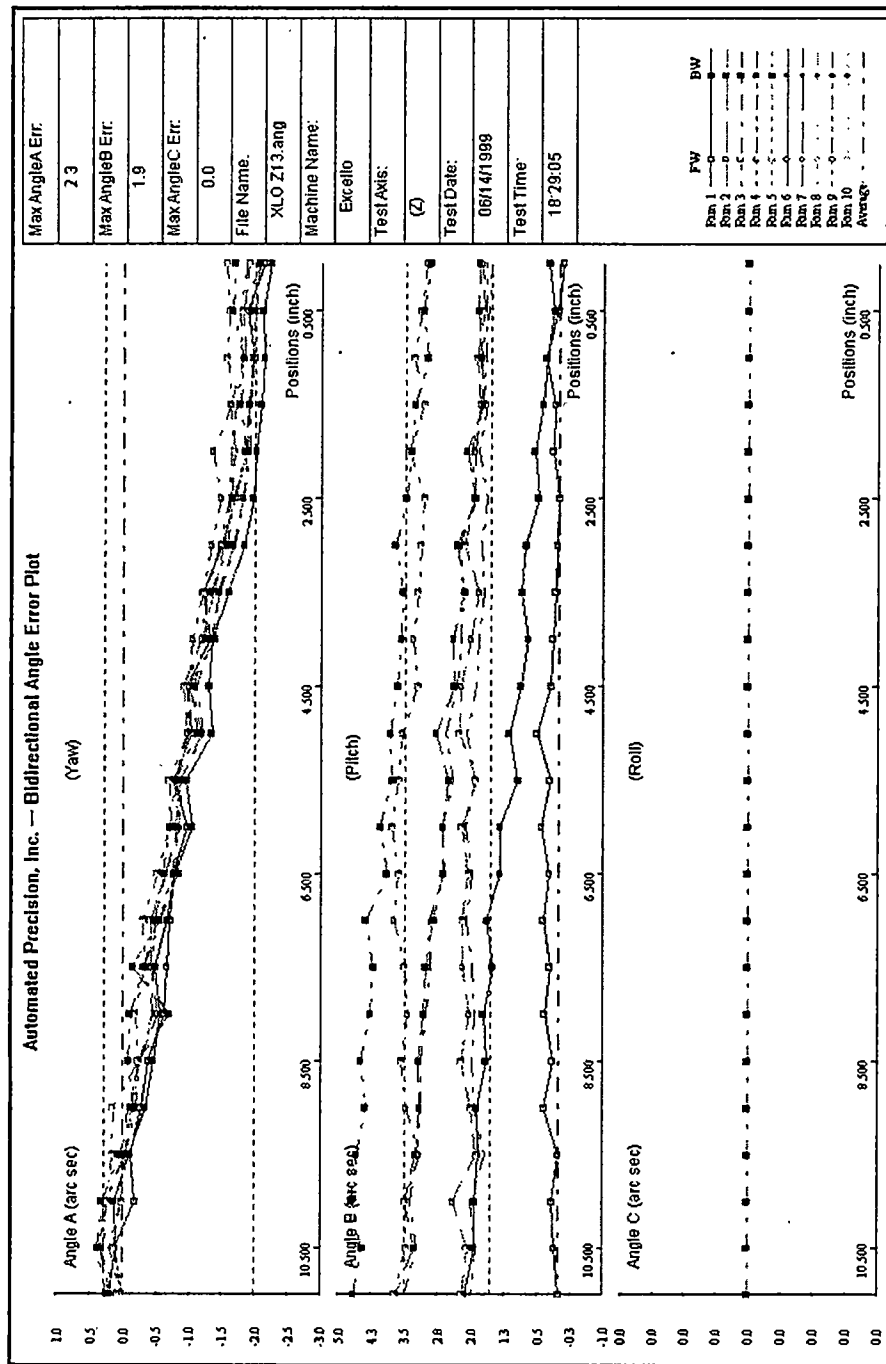
X-AXIS ANGULAR ERRORS



Z-AXIS LEAD SCREW ERRORS



Z-AXIS STRAIGHTNESS ERRORS



Z-AXIS ANGULAR ERROR

VITA

Van Graves was born in Knoxville, Tennessee on March 18, 1964. He attended public schools in Knox and Blount counties and graduated from Heritage High School in June 1981. He entered the University of Tennessee, Knoxville, in September of 1981 and majored in Mechanical Engineering. During his sophomore and junior years of study he participated in the university's Cooperative Education Program and worked at the Department of Energy's Y-12 Plant in Oak Ridge, TN. Co-op work periods in the plant's Maintenance and Computer Integrated Manufacturing divisions were completed. He received the Bachelor of Science degree in Mechanical Engineering in June of 1986. In August 1986, he began working as an associate engineer in the Development Division at Y-12. During that period he primarily worked on developing manufacturing software applications that integrated with the plant's CAD systems. In 1993, he transferred to the Robotics and Process Systems Division of the Oak Ridge National Laboratory where he participated in automated ammunition handling projects for the U.S. Army. It was during that time that he began working part-time toward a Master's Degree in Mechanical Engineering from the University of Tennessee. His thesis project was funded and performed at Y-12, and the Master's Degree was awarded in August 1999.

He is currently employed at the Oak Ridge National Laboratory and is the Principal Investigator for a R&D project in support of the Spallation Neutron Source (SNS), a billion-dollar research facility planned for construction at the Laboratory. Remote handling and maintenance tests using manipulator systems will be conducted on a prototype of the SNS Target Cell. In addition, thermal and hydraulic studies will be performed on the liquid mercury flow loop.