U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology

NISTIR 4710

Use of Solid Modeling in the Design of M^3 Components

Allison Barnard Feeney
Peter F. Brown
Factory Automation Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

September, 1991

NIST

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology

NISTIR 4710

Use of Solid Modeling in the Design of M^3 Components

Allison Barnard Feeney
Peter F. Brown
Factory Automation Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

September, 1991

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary of Commerce

National Institute of Standards and Technology
John W. Lyons, Director

DEPARTMENT OF COMMERCE
UNITED STATES OF AMERICA

Table of Contents

List of Figures

List of Tables

EngineeringDesignLab

Use of Solid Modeling in the Design of M^3 Components

Allison Barnard Feeney
Peter F. Brown
Factory Automation Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

1 Introduction

This paper describes the work that was done for the Molecular Measuring Machine (M^3) Project in the Engineering Design Laboratory (EDL). This work involved modeling critical components of the Molecular Measuring Machine's inner Mallock suspension system with a solid modeler, attributing these models with material densities, and calculating mass properties information. The mass properties information was requested by the M^3 project designers for use in developing an active vibration isolation system for the device. The primary objective of this paper is to provide mass properties information through the tables given in Section 4.

1.1 Molecular Measuring Machine Project

The National Institute of Standards and Technology (NIST) is performing research into developing new measurement technologies. One such program is the Molecular Measuring Machine (M^3) project within the Precision Engineering Division. The goal of this project is to design and build an ultra-high accuracy planar coordinate measuring machine capable of positioning and measuring to atomic scale accuracies. The machine will be used to determine dimensions of features, angles, staightnesses, and other geometrical properties of molecular-sized objects or large-sized objects to be measured to very high accuracies. The M^3 was designed to address current and anticipated metrology needs of the microelectronics and optics industries and to serve as an exploratory tool for the rapidly growing field of nanotechnology [1].

The design, manufacturing, and operation of a device such as M^3 is a complex engineering problem. Certain portions of the design and analysis require the use of state of the art software tools. The design of the active vibration isolation subsystem for M^3 required more advanced software tools than were available to the M^3 Project. It was recognized that within a better design environment the subsystem could be optimized, thereby providing a higher quality design in a shorter period of time. For this reason, a collaboration was established between the M^3 Project and the Engineering Design Laboratory.

1.2 Engineering Design Laboratory

The Engineering Design Laboratory was established in 1989 by the Factory Automation Systems Division at NIST. The EDL was created to study the process of mechanical engineering design, how design information can be represented, and how this information is used throughout a product's life cycle. In order to fulfill this mission, the EDL has acquired many commercial and research software tools used to perform design and analysis [2]. One tool that has been increasingly used in design, for both research and industry, is the solid modeler. Solid models of components can be used in a variety of ways: for example, to determine mass properties, to help visualize a device, to aid the analysis process, and to generate tool paths for making physical parts. Solid modeling systems began appearing commercially in the early 1980's [3] and have become increasingly used by industrial and research communities [4].

In order for the EDL to study the process of engineering design, collaborations were established with projects having complex engineering design problems; M^3 is one such project. The EDL had several objectives for this collaboration:

- to test the capabilities of EDL tools on a complex design problem,
- to transfer knowledge of solid modeling technology to the M^3 program,
- to promote the use of solid modeling.

1.3 Overview

The body of this paper is divided into six major sections. The following section provides an overview of the design of M^3, the vibration isolation system, its sub-systems and components. Section 3 will describe the solid modeler used and the accuracy of the part models. Section 4 discusses the methods for extracting the mass property information and gives the results of the calculations. The fifth section describes other types of analysis that can be performed on solid models to facilitate design and manufacturing. Conclusions are provided in Section 6. Two appendices are included to document the specifics of the effort. Appendix A illustrates the modeling method by providing part files, and Appendix B provides additional hidden line images of the M^3 components and subsystems for visualization.

2 M^3 Design Overview

As mentioned earlier, one of the goals of the Molecular Measuring Machine project is to design and build an ultra-high accuracy planar coordinate measuring machine. One of the most challenging design issues of M^3 is how to isolate the machine from the environment. Because of the tremendous sensitivity of the device, it is susceptible to many types of disturbances. These disturbances are caused by other equipment in the building (such as heating and ventilating), by people walking down hallways, and even by vehicular traffic on an expressway half a kilometer away.

Extraordinary measures are taken to isolate the device from mechanical vibrations, thermal loading and acoustic noise. The isolation of mechanical vibrations is of primary concern to the work described in this paper. Figure 1 is a schematic illustrating the vibration isolation systems employed in M^3. The mechanical vibration isolation in M^3 is achieved through two levels of passive isolation and one level of active isolation. The first level of passive isolation is accomplished by separating the floor of the room where the M^3 experiment is housed from the rest of the building. The second level of passive isolation is performed by supporting the outer Mallock assembly on large pneumatic cylinders. These cylinders act as springs to dampen other vibrations. The active level is represented by the interface between the inner and outer Mallock assemblies.

The following section will discuss the design of the active level of the vibration isolation system. Section 2.2 will discuss the subsystems and individual components of this vibration isolation system that were modeled in this effort. For a more rigorous description of the design goals and specifications of the M^3 vibration isolation system, see [5].

2.1 Active Vibration Isolation System

The active vibration isolation system is based on a Mallock suspension system [5] which can constrain a six degree of freedom system to a single degree of freedom (for small deflections). The active vibration isolation system consists of a set of accelerometers and piezoelectric force actuators. The accelerometers sense the vibration then feed it to a control computer where a dynamic model of the system exists. This system determines which force actuators need to be activated to counteract the vibration induced motion. If all works as planned, this system will cancel out vibrations between ten hertz and one millihertz. The control algorithms of the dynamics module require the inertia tensor, center of gravity, and mass of the entire inner Mallock assembly.

2.2 Components of the Inner Mallock Assembly

The inner Mallock housing encloses the inner workings of the M^3. It is a roughly spherical aluminium housing comprised of five separate components. Within the housing are the inner sphere, the inner sphere support system, the heater shell, the carriages, and all of the additional devices required to perform measurements. Collectively, these components are

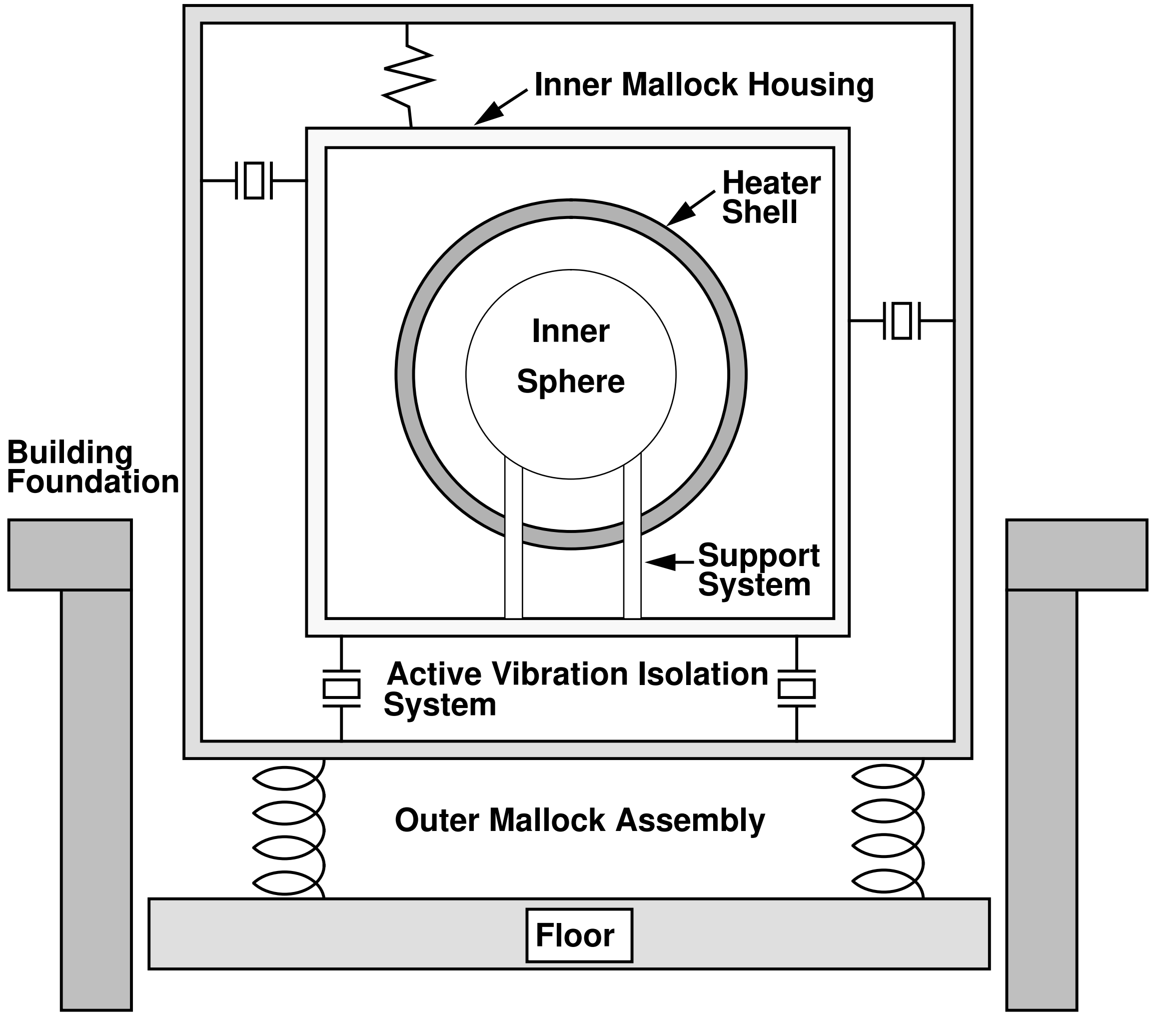


Figure 1. Schematic of M^3 Vibration Isolation Systems

referred to as the inner Mallock assembly. The top most component of the inner Mallock housing is the enclosure for the compression spring. The next three parts form the outer shell. The fifth component is a base plate to support the heater shell and inner sphere. All parts of the inner Mallock are made of 6061 Aluminum except the spring enclosure which is 440C Stainless Steel. The housing is shown in the figure in Table 2, and individual components are pictured in Appendix B.

The core of M^3 is a 14 inch diameter solid copper sphere chosen for high mechanical stiffness and ease of temperature control. There are two crossed linear slideways that are an integral part of the core mechanical structure. Two copper carriages are driven along these slideways by motors embedded in the sphere. The upper carriage transports the probe; the lower carriage transports the specimen. The inner sphere is pictured in Table 5. Carriages are illustrated in Tables 6 and 7. Appendix B contains additional images of these components.

The principal problem of the core mounting system was how to implement a three point support to isolate the core from deformations in the inner Mallock frame. The system chosen was the Kelvin Clamp which utilizes a cone, a groove and a flat. Four points of contact were necessary to adequately support the inner sphere. Four legs are placed symmetrically on the sphere, mounted via conical support points and bearings. The four legs are mounted to the inner Mallock base at three points. Two legs are fastened directly to the base. One leg has a conical top; the other has a flat top. The remaining two legs are mounted to a plate with a grooved insert that rests on four bearings in four cones in an insert to the Mallock base. This system is shown in Table 4 and in Appendix B.

The heater shell is a thin gold plated brass sphere that surrounds the inner sphere. Its function is to absorb the heat generated by the motors that drive the carriages. The heater shell is supported by four legs that are fastened to a ledge in the inner Mallock base. The heater shell can be viewed in Table 3 and in Appendix B.

3 M^3 Part Modeling

When the Engineering Design Laboratory became involved with the M^3 project the machine had been configured, most detailed design was completed, and many of the components had been fabricated. The design had been documented through drawings created on a two dimensional computer-aided design (CAD) package. These drawings were adequate for sending to a job shop and having the one-of-a-kind parts manufactured, but otherwise had limited usefulness. The M^3 project needed to calculate the mass properties of parts and assemblies required for the control algorithms of the active vibration isolation system. Lacking tools to calculate accurate mass property information of the subsystem, the mass property information would have been roughly estimated and a lengthy iterative process would be needed to refine the control parameters. Solid models provide a complete and unambiguous computer interpretable shape representation from which mass properties can be automatically generated. Therefore, the decision was made to create solid models of the inner Mallock suspension system.

The following sections discuss the particular solid modeler used in this work and the level of accuracy of the M^3 part models.

3.1 Solid Modeler

The solid modeler used in this work is Parasolid[1], a product of Shape Data Limited. Parasolid is a kernel based solid modeler with an underlying exact boundary representation and some sculptured surface capabilities. Because it is a solid modeler, it can be used to manipulate solids, create assemblies, calculate mass and moments of inertia, and perform

1. Certain commercial equipment, instruments, or materials are identified in this paper. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the material or equipment identified are necessarily the best available for the purpose.

interference detection. Because it provides an exact boundary representation, calculations such as mass can be made very accurately, as compared to a facetted boundary representation. The modeler also knows where all the edges of the objects are, as opposed to the constructive solid geometry modeler which does not have an immediate knowledge of where edges are for interference detection [6].

Parts were modeled through the use of Parasolid's Kernel interface Driver (KID). KID is a simple graphic interface that is supplied to test the Kernel routines [7]. This interface is not intended to provide CAD functionality. In fact, Parasolid is available as the core of several commercial CAD systems. Because Parasolid was procured by NIST for use in other applications such as data translators, it was purchased without a CAD front end. Parasolid files are included in Appendix A to illustrate how M^3 parts and assemblies were constructed. For more information on solid modeling, see [8]

3.2 Accuracy of the Part Models

Two objectives that had a direct impact on modeling decisions were established at the beginning of the modeling effort. The first objective was to model the parts so the models could be used to predict the behavior of the physical device. At the commencement of our work on this project we were given the drawings of the device that had been generated on the two dimensional drafting system and provided access to the manufactured prototypes. Many modifications to the design had been made during the manufacturing process to facilitate the machining of the components. Where we could detect a difference between the drawings and the prototype, the prototype change was incorporated into the solid model.

The second objective was to model the parts to an appropriate level of detail as required to give accurate results for the mass properties. Since these solid models were not intended to replace the CAD drawings, some details (e.g. fillets, chamfers, small holes) shown on the drawings were not modeled. An error sensitivity analysis was conducted at the outset to determine the effect of small volumes on the mass property results. As a result of this analysis, the following observations were made:

- Small details did not have to be included. Because fillet and chamfer volumes were often less than 1% of the rest of the part, their impact on the mass property calculation results was insignificant.
- Many holes could be ignored. Most would later be filled with bolts, so the mass property results would only be affected by the difference in mass if the bolt material and the surrounding material were different.

The overall shape, size, and relative spatial location of the components were modeled accurately.

4 Mass Properties

As mentioned in Section 2, the mass property information of the M^3 components and assemblies is required by the control algorithms in the dynamics module of the active vibration isolation system. The information necessary for these algorithms is the inertia tensor, center of gravity, and mass. In this section are tables documenting the results of the mass property calculations on the individual components, subassemblies, and overall system assembly.

4.1 Mass Property Calculations

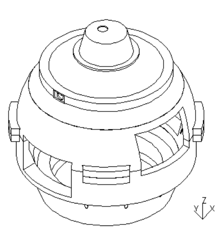
The solid modeling system required two inputs to the mass property calculation: the density of the material and a valid part or assembly. Given a solid model or assembly of solid models, the outputs of the calculations were total surface area, volume, mass, center of gravity, and the inertia tensor at the center of gravity. The center of gravity result is relative to the global origin of the parts, located at the center of the inner sphere. Parameters could be set to control the accuracy of all calculations. For these calculations the accuracy was set to 1.0, the highest level. The mass property routine was requested to calculate error estimates for the values found. These errors, expressed as numbers to be added to or subtracted from the answer, were insignificantly small.

4.2 Mass Property Results

Table 1. Inner Mallock Assembly

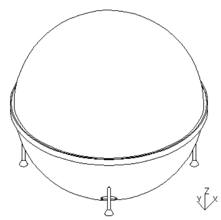
Surface Area (in^2):	7429.16	Volume (in^3):	2874.41
Density (lb•in^3):		Weight (lb):	538.11
Center of Gravity (in):	X	Y	Z
	-0.022	-0.003	-0.528
Inertia Tensor (lb•in^2):	I_x	I_y	I_z
	24167.66	-0.72	-124.51
	-0.72	24372.00	13.8
	-124.51	13.8	22463.47

Table 2. Inner Mallock Housing



Surface Area (in^2):	3780.78	Volume (in^3):	1724.74
Density (lb•in^3):	0.098	Weight (lb):	169.02
Center of Gravity (in):	X	Y	Z
	0.011	-0.009	-1.107
Inertia Tensor (lb•in^2):	I_x	I_y	I_z
	14750	-0.06	-10.2
	-0.06	14621.12	14.68
	-10.2	14.68	13964.61

Table 3. Heater Shell

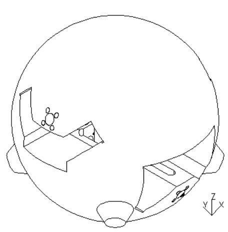


Surface Area (in^2):	1942.397	Volume (in^3):	97.051
Density (lb•in^3):	0.322	Weight (lb):	31.25
Center of Gravity (in):	X	Y	Z
	0.00	0.00	0.01
Inertia Tensor (lb•in^2):	I_x	I_y	I_z
	1511.86	-0.01	0.0
	-0.01	1511.86	0.01
	0.0	0.01	1638.77

Table 4. Support System

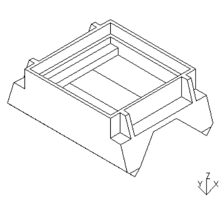
Surface Area (in^2):	134.061	Volume (in^3):	26.309
Density (lb•in^3):	0.28	Weight (lb):	7.367
Center of Gravity (in):	X	Y	Z
	-1.850	0.0	-7.861
Inertia Tensor (lb•in^2):	I_x	I_y	I_z
	108.02	-0.71	-15.48
	-0.71	129.78	0.0
	-15.48	0.0	206.46

Table 5. Inner Sphere



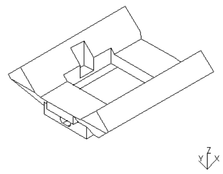
Surface Area (in^2):	1165.638	Volume (in^3):	906.2
Density (lb•in^3):	0.322	Weight (lb):	291.796
Center of Gravity (in):	X	Y	Z
	0.0	0.0	0.0
Inertia Tensor (lb•in^2):	I_x	I_y	I_z
	6901.69	0.01	-0.01
	0.01	7211.00	0.02
	-0.01	0.02	6148.68

Table 6. Lower Carriage



Surface Area (in^2):	237.804	Volume (in^3):	67.366
Density (lb•in^3):	0.322	Weight (lb):	21.692
Center of Gravity (in):	X	Y	Z
	0.0	0.0	-1.937
Inertia Tensor (lb•in^2):	I_x	I_y	I_z
	163.9	0.0	0.0
	0.0	141.078	0.0
	0.0	0.0	281.881

Table 7. Upper Carriage



Surface Area (in^2):	168.483	Volume (in^3):	52.765
Density ($lb \cdot in^3$):	0.322	Weight (lb):	16.990
Center of Gravity (in):	X	Y	Z
	0.0	0.00	1.379
Inertia Tensor ($lb \cdot in^2$):	I_x	I_y	I_z
	105.108	0.0	0.0
	0.0	105.125	0.0
	0.0	0.0	198.599

Table 8. Inner Mallock Assembly

(with Upper Carriage moved)

Surface Area (in^2):	7429.16	Volume (in^3):	2874.41
Density ($lb \cdot in^3$):		Weight (lb):	538.11
Center of Gravity (in):	X	Y	Z
	0.001	-0.003	-0.528
Inertia Tensor ($lb \cdot in^2$):	I_x	I_y	I_z
	24617.66	-0.767	-156.91
	-0.767	24389.19	13.8
	-156.91	13.8	22480.67

Table 9. Inner Mallock Assembly

(with Lower Carriage moved)

Surface Area (in^2):	7429.16	Volume (in^3):	2874.41
Density ($lb \cdot in^3$):		Weight (lb):	538.11
Center of Gravity (in):	X	Y	Z
	-0.022	0.037	-0.528
Inertia Tensor ($lb \cdot in^2$):	I_x	I_y	I_z
	24188.6	-1.193	-124.51
	-1.19	24372.0	44.36
	-124.51	44.36	22484.41

Table 10. Inner Mallock Assembly

(with both Carriages moved)

Surface Area (in^2):	7429.16	Volume (in^3):	2874.41
Density ($lb \cdot in^3$):		Weight (lb):	538.11
Center of Gravity (in):	X	Y	Z
	0.01	0.037	-0.528
Inertia Tensor ($lb \cdot in^2$):	I_x	I_y	I_z
	24188.60	-0.55	-156.91
	-0.55	24389.19	44.36
	-156.91	44.36	22501.61

5 Solid Models in Analysis and Manufacturing

While determining the mass properties of a part is clearly one of the most useful by-products of a solid model, an integrated solid model shape representation also supports other types of analysis and planning. Ideally, one would like to be able to define the detailed geometry of a part once, then be able to use it throughout the product's life. The following sections briefly describe several ways solid models are being used over the course of the product's life. Included are:

- finite element mesh generation,
- interference checking and cross-section analysis,
- tool path generation, and
- visualization.

5.1 Finite Element Mesh Generation

Solid models can be used to automate the generation of finite element meshes for modal, stress, deflection, and thermal analysis. The generation of the meshes for finite element analysis is typically the most labor intensive aspect of this type of analysis. Traditionally, the analyst has discretized the geometry of the part into a series of nodes and elements that approximated the part shape. This process was error prone and time consuming. Recently, research systems have demonstrated the generation of a valid, analysis ready finite element mesh directly from a solid model. For the complexity of the M^3 parts modeled in this effort, this ability can save days of work and greatly improve the accuracy of the finite element meshes and results. Several commercial analysis packages are now including this capability with their software.

5.2 Interference and Cross Section Analysis

Solid modelers provide the ability to cut cross sections through parts and assemblies and to determine clearances or collisions between parts in assemblies. Assembly components can be positioned in space and the solid modeling system queried to determine if intersections exist and if so, between which solids. The ability to determine if two solids intersect or are disjoint is useful when developing assembly sequences. In the design of M^3, the clearances between many components were critical. This ability would have facilitated the design process.

5.3 Tool Path Generation

Solid models are being used in commercially available software tools to generate tool paths for both cutting and inspection in a semi-automatic fashion. Several commercial systems offer the ability to generate the tool paths automatically from solid models of the part, stock, and fixtures. In addition to facilitating a time consuming task, toolpath generation provides feedback for manufacturing planning. Many of these systems display the volume of material remaining after the tool pass, some animate the process. Problems such as gouges in fixtures and uncut stock can be discovered before any material is cut.

5.4 Visualization

Solid models can aid designers and other interested parties in visualizing the devices being designed. Hidden line images, exploded assembly diagrams and cut-away section views can be easily created from solid models. Such drawings are often needed for developing product documentation. Appendix B contains several examples of hidden line images and cut-away sections of M^3 parts and assemblies.

6 Summary

This paper has documented efforts within the Engineering Design Laboratory to create solid models of Molecular Measuring Machine components. This work has demonstrated that solid models provide benefit to the design process through mass property calculation capabilities. Additionally, this paper has discussed other ways solid model shape representations are being used in product lifecycle applications. It was made clear through the course of this work that using a solid model part representation earlier in the M^3 design process would have saved much effort and time by providing improved visualization of the design, interference detection between assembly components, automatic finite element mesh generation, and toolpath generation for manufacturing planning.

Through the course of this collaboration, the EDL has achieved its objectives of:

- testing the capabilities of EDL tools on a complex design problem, particularly the Parasolid solid modeler
- transferring knowledge of solid modeling technology to the M^3 program,
- demonstrating the utility of solid modeling.

Working with the M^3 project has provided the EDL valuable experience with a real design problem. The design process requires many views of an object. These views range from conceptual (i.e. back of the envelope) sketches to analysis models to detailed solid models. We have had to deal with the problems of re-entering data (i.e. incompatibilities between the 2D CAD and solid modeling systems), changes from as-designed to as-manufactured part geometries, and various user interfaces. This design effort has led us to question relationships and roles between these representations and we will continue to explore this in the future within the EDL.

Acknowledgments: The authors wish to acknowledge the support of their sponsors. Funding for this work was provided through the Automated Manufacturing Research Facility which is supported by the Navy Manufacturing Technology Program and NIST.

7 References

1. Teague, E. Clayton, "The National Institute of Standards and Technology Molecular Measuring Machine Project: Metrology and Precision Engineering Design," Journal of Vacuum Science and Technology, November/December 1989, pp. 1898-1902.

2. Barnard Feeney, Allison, "Engineering Design Laboratory Guide," National Institute of Standards and Technology Interagency Report 4519, 13 pages (February 1991). (Available from the National Technical Information Service (NTIS), Springfield, VA 22161).

3. Requicha, A. A. G., Volecker, H. B., "Solid Modeling: A Historical Summary and Contemporary Assessment," IEEE Computer Graphics and Applications, March 1982, pp. 9-24.

4. Rossignac, Jaroslaw, Turner, Joshua, eds., Symposium on Solid Modeling and CAD/CAM Applications, ACM Press, June,1991.

5. Gilsinn, Dave, Lyons, Matthew, Scire, Frederic, Teague, E. Clayton, Amatucci, Edward and McIntyre, Timothy "Design and Assembly of a Vibration Isolation System for the Molecular Measuring Machine," Draft National Institute of Standards and Technology Interagency Report (August 1991).

6. Parasolid V. 3.0 Reference Manual, Shape Data Limited, Cambridge, England, 1990.

7. Parasolid V. 3.0 The Kernel Interface Driver Manual, Shape Data Limited, Cambridge, England, 1990.

8. Mortensen, Michael E., Geometric Modeling, John Wiley and Sons, Inc., 1985.

A Parasolid Part Files

The following are two representative files provided to show the reader how part modeling was accomplished within the Parasolid modeling system. The file in Section A.1 includes all the steps necessary to generate the inner sphere, one of the more complex parts modeled in this effort. Similar files were written to create each individual component. The second file, in Section A.2, demonstrates the process of instancing previously defined parts and collecting them into an assembly. This assembly could then be used as input for mass property calculations.

A.1 Inner Sphere

```
-- This file creates the inner sphere of the molecular measuring machine
-- Created by Allison Barnard and Pete Brown
-- Last modified 3/7/91
-- Addition:   Fixed a problem with lower back counterbore
--             Removed redraw from the graphics and added
--             some additional displays

--these lines start the modeler and set up the graphics
(modeller start)

--inner_sphere -> inner_sphere: the basic shape of the inner sphere
(define inner_sphere p_sphere)
(inner_sphere point '(0 0 0); radius 7.00; create)

--assign the inner_sphere the density for copper
(creatt (inner_sphere tag) 10 nil '(0.322) '(density))

--uco -> upper_cut_out: the profile swept linearly that defines the upper
--slideway
(define uco p_profile)
(uco coordinate '((7.5   -2.04    0.0)
                  (7.5   -2.04    0.80)
                  (7.5   -3.876   2.636)
                  (7.5   -3.9467  2.5653)
                  (7.5   -4.0174  2.636)
                  (7.5   -2.852   3.8)
                  (7.5   -1.45    2.4)
                  (7.5    1.45    2.4)
                  (7.5    2.852   3.8)
                  (7.5    4.0174  2.636)
                  (7.5    3.9467  2.5653)
                  (7.5    3.876   2.636)
                  (7.5    2.04    0.8)
                  (7.5    2.04    0.0)
```

```
                        (7.5   -2.04    0.0)))
(uco create)
(uco sweep '(-15 0 0))
(inner_sphere subtract 'uco)

--luth -> large_upper_thru_hole
(define luth p_cylinder)
(luth point '(7.0 0 3.1); radius 0.375; height 14.0; direction '(-1 0 0))
(luth create)
(inner_sphere subtract 'luth)

--suth1-4 -> small_upper_thru_holes
(define suth1 p_cylinder)
(suth1 point '(7.0  0.4176 3.5176);radius 0.06;height 14.0;direction '(-1 0 0))
(suth1 create)

--ucbf -> upper_counter_bore_front
(define ucbf p_cylinder)
(ucbf point '(7.0  0.4176 3.5176);radius 0.156;height 4.2;direction '(-1 0 0))
(ucbf create)

--ucbb -> upper_counter_bore_back
(define ucbb p_cylinder)
(ucbb point '(-2.8  0.4176 3.5176);radius 0.156;height 4.2;direction '(-1 0 0))
(ucbb create)
(suth1 unite 'ucbf)
(suth1 unite 'ucbb)

--replicate this 3 times and move to other locations.
(define suth2 body)
(define suth3 body)


(define suth4 body)
(suth2 replicate 'suth1)
(suth2 direction '(0 0 -1); distance 0.8352; move)
(suth3 replicate 'suth2)
(suth3 direction '(0 -1 0); distance 0.8352; move)
(suth4 replicate 'suth1)
(suth4 direction '(0 -1 0); distance 0.8352; move)

(inner_sphere subtract 'suth1)
(inner_sphere subtract 'suth2)
(inner_sphere subtract 'suth3)
(inner_sphere subtract 'suth4)
```

```
--lco -> lower_cut out: profile swept for lower slideways
(define lco p_profile)
(lco coordinate '(( 3.4000  7.5    0.8)
                  ( 3.4000  7.5   -3.068)
                  ( 2.688   7.5   -3.7794)
                  ( 2.6173  7.5   -3.7087)
                  ( 2.688   7.5   -3.638)
                  ( 0.850   7.5   -1.8)
                  (-0.850   7.5   -1.8)
                  (-2.688   7.5   -3.638)
                  (-2.6173  7.5   -3.7087)
                  (-2.688   7.5   -3.7794)
                  (-3.400   7.5   -3.068)
                  (-3.400   7.5    0.8)
                  ( 3.400   7.5    0.8)))

(lco create)
(lco sweep '(0 -15 0))
(inner_sphere subtract 'lco)

--llth -> large_lower_thru_hole
(define llth p_cylinder)
(llth point '(0.0 7.0 -3.1); radius 0.375; height 14.0; direction '(0 -1 0))
(llth create)
(inner_sphere subtract 'llth)

--slth1-4 -> small_lower_thru_hole
(define slth1 p_cylinder)
(slth1 point '(0.4176 7.0 -2.6824);radius 0.06;height 14.0;direction '(0 -1 0))
(slth1 create)

--lcbf -> lower_counter_bore_front
(define lcbf p_cylinder)
(lcbf point '(0.4176 7.0 -2.6824);radius 0.156;height 4.2;direction '(0 -1 0))
(lcbf create)

--lcbb -> lower_counter_core_back
(define lcbb p_cylinder)
(lcbb point '(0.4176 -2.8 -2.6824);radius 0.156;height 4.2;direction '(0 -1 0))
(lcbb create)
(slth1 unite 'lcbf)
(slth1 unite 'lcbb)

--replicate this 3 times and move to other locations.
(define slth2 body)
(define slth3 body)
```

```
(define slth4 body)
(slth2 replicate 'slth1)
(slth2 direction '(0 0 -1); distance 0.8352; move)
(slth3 replicate 'slth2)
(slth3 direction '(-1 0 0); distance 0.8352; move)
(slth4 replicate 'slth1)
(slth4 direction '(-1 0 0); distance 0.8352; move)

(inner_sphere subtract 'slth1)
(inner_sphere subtract 'slth2)
(inner_sphere subtract 'slth3)
(inner_sphere subtract 'slth4)

--us1,2 -> upper_slots
(define us1 p_block)
(us1 point '(4.15 0 0.5); x 2.3; y 0.624; z 3.0; direction '(0 0 1); create)

--end1,2 -> rounded ends of slots
(define end1 p_cylinder)
(end1 point '(3.0 0 0.5); radius 0.312; height 3.0; direction '(0 0 1))
(end1 create)
(define end2 p_cylinder)
(end2 point '(5.3 0 0.5); radius 0.312; height 3.0; direction '(0 0 1))
(end2 create)
(us1 unite 'end1)
(us1 unite 'end2)

-- second slot defined by replication
(define us2 body)
(us2 replicate 'us1)
(us2 create)
(us2 direction '(-1 0 0); distance 8.3; move)

(inner_sphere subtract 'us1)
(inner_sphere subtract 'us2)

--ls1,2 -> lower_slots
(define ls1 p_block)
(ls1 point '(0 4.15 -0.5); direction '(0 0 -1); y 2.3; x 0.624; z 3.0; create)
--lend1,2 -> ends of lower slot
(define lend1 p_cylinder)
(lend1 point '(0 3.0 -0.5); radius 0.312; height 3.0; direction '(0 0 -1))
(lend1 create)
(define lend2 p_cylinder)
(lend2 point '(0 5.3 -0.5); radius 0.312; height 3.0; direction '(0 0 -1))
(lend2 create)
```

```
(ls1 unite 'lend1)
(ls1 unite 'lend2)

--second slot defined by replication
(define ls2 body)
(ls2 replicate 'ls1)
(ls2 create)
(ls2 direction '(0 -1 0); distance 8.3; move)

(inner_sphere subtract 'ls1)
(inner_sphere subtract 'ls2)

--umc -> upper_machine_cutout
(define umc p_block)
(umc point '(0 0 0.03); x 5.0; y 1.9; z 4.0; direction '(0 0 1); create)

--umcc1-4 -> upper_machine_cutout_corners
(define umcc1 p_cylinder)
(umcc1 point '(2.28 .7295 .03); radius 0.3125; height 4.0; direction '(0 0 1))
(umcc1 create)

--corners 2-4 created by replication
(define umcc2 p_cylinder)
(define umcc3 p_cylinder)
(define umcc4 p_cylinder)
(umcc2 replicate 'umcc1)
(umcc2 direction '(0 -1 0); distance 1.459; move)
(umcc3 replicate 'umcc2)
(umcc3 direction '(-1 0 0); distance 4.56; move)
(umcc4 replicate 'umcc1)
(umcc4 direction '(-1 0 0); distance 4.56; move)

(umc unite 'umcc1)
(umc unite 'umcc2)
(umc unite 'umcc3)
(umc unite 'umcc4)
(inner_sphere subtract 'umc)
(printc "Upper Slots Cutout Subtracted")

--lmc -> lower_machine_cutout
(define lmc p_block)
(lmc point '(0 0 -0.03); x 1.9; y 5.0; z 4.0; direction '(0 0 -1); create)
(printc "Lower Machine Cutout Subtracted")

--lmcc1-4 -> lower_machine_cutout_corners
(define lmcc1 p_cylinder)
```

```
(lmcc1 point '(0.7295 2.280 -.03);radius 0.3125;height 4.0;direction '(0 0 -1))
(lmcc1 create)
(printc "Lower Machine Corners")

--corners 2-4 created by replication
(define lmcc2 p_cylinder)
(define lmcc3 p_cylinder)
(define lmcc4 p_cylinder)
(lmcc2 replicate 'lmcc1)
(lmcc2 direction '(0 -1 0); distance 4.56; move)
(lmcc3 replicate 'lmcc2)
(lmcc3 direction '(-1 0 0); distance 1.459; move)
(lmcc4 replicate 'lmcc1)
(lmcc4 direction '(-1 0 0); distance 1.459; move)

(lmc unite 'lmcc1)
(lmc unite 'lmcc2)
(lmc unite 'lmcc3)
(lmc unite 'lmcc4)
(inner_sphere subtract 'lmc)
(printc "Lower Machine Cutout Subtracted")

--****************************************************************
--this code creates and unions the cone shaped thing to the side of
--the inner sphere.  In cases where the sphere is to be shown without
--the supports, this section should be commented out!
(define cone p_cone)
(cone point '(-4.33 -4.33 -3.1736); urad .7; lrad 1.25; height 1.0)
(cone direction '(-4.33 -4.33 -3.1736); create)

--define cone to be removed from the support cone
(define scone p_cone)
(scone point '(-4.33 -4.33 -4.3); urad 0; lrad .375; height .5)
(scone direction '(0 0 1); create)
(cone subtract 'scone)

--define other three support cones by copying and rotating about the z axis
(define cone2 body)
(define cone3 body)
(define cone4 body)

(cone2 replicate 'cone)
(cone2 point '(0 0 0); direction '(0 0 1); angle 90; rotate)
(cone3 replicate 'cone2)
(cone3 point '(0 0 0); direction '(0 0 1); angle 90; rotate)
(cone4 replicate 'cone3)
```

```
(cone4 point '(0 0 0); direction '(0 0 1); angle 90; rotate)

--unite cone to inner_sphere
(inner_sphere unite 'cone)
(inner_sphere unite 'cone2)
(inner_sphere unite 'cone3)
(inner_sphere unite 'cone4)
--***************************************************************--

--code to clean up extra topology and draw final object
(mergen (inner_sphere tag))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'inner_sphere)))
```

A.2 Assembly

```
--This file loads in all previously defined parts, assigns material densities,
--then sketches the part. Creates assemblies for support system, inner
--mallock frame and overall system.
--Created by Allison Barnard and Pete Brown
--Last modified 3/21/91

--inner sphere
(define inner_sphere body)
(inner_sphere receive "inner_sphere")
--assign inner sphere the density for copper
(creatt (inner_sphere tag) 10 nil '(0.322) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'inner_sphere;ar)))

--heater shell
(define heater_shell body)
(heater_shell receive "heater_shell")
--assign heater shell the density for copper
(creatt (heater_shell tag) 10 nil '(0.322) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'heater_shell; ar)))

--import parts of the support system; assign all density of stainless steel
--support one
```

```
(define support1 body)
(support1 receive "support1")
(creatt (support1 tag) 10 nil '(0.28) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'support1 ; ar)))

--support one insert
(define ss1_insert body)
(ss1_insert receive "ss1_insert")
(creatt (ss1_insert tag) 10 nil '(0.28) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'ss1_insert)))

--sphere one
--s1 -> sphere_1a
(define s1 p_sphere)
(s1 point '(-4.33 -4.33 -4.615); radius 0.375; create)
(creatt (s1 tag) 10 nil '(0.28) '(density))

--sphere two
--s2 -> sphere_1b
(define s2 p_sphere)
(s2 point '(-4.33 4.33 -4.615); radius 0.375; create)
(creatt (s2 tag) 10 nil '(0.28) '(density))

--support two
(define support2 body)
(support2 receive "support2")
(creatt (support2 tag) 10 nil '(0.28) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'support2; ar)))

--sphere three
(define s3 p_sphere)
(s3 point '(4.33 -4.33 -4.615); radius 0.375; create)
(creatt (s3 tag) 10 nil '(0.28) '(density))

--support three
(define support3 body)
(support3 receive "support3")
```

```
(creatt (support3 tag) 10 nil '(0.28) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'support3;ar)))

--sphere four
(define s4 p_sphere)
(s4 point '(4.33 4.33 -4.615); radius 0.375; create)
(creatt (s4 tag) 10 nil '(0.28) '(density))

--sphere support base
(define ss_base body)
(ss_base receive "ss_base")
(creatt (ss_base tag) 10 nil '(0.28) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'ss_base;ar)))

--show the whole support system
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch '(support1 ss1_insert s1 s2 support2 support3 ss_base s3 s4); ar)))

--define the support assembly
(define support_a1 p_assembly)
(support_a1 create)

--define instances of components in support system assembly
(define i1 p_instance)
(define i2 p_instance)
(define i3 p_instance)
(define i4 p_instance)
(define i5 p_instance)
(define i6 p_instance)
(define i7 p_instance)
(define i8 p_instance)
(define i9 p_instance)
(i1 assembly 'support_a1 ; part 'support1 ; create)
(i2 assembly 'support_a1 ; part 'support2 ; create)
(i3 assembly 'support_a1 ; part 'support3 ; create)
(i4 assembly 'support_a1 ; part 'ss1_insert ; create)
(i5 assembly 'support_a1 ; part 'ss_base ; create)
(i6 assembly 'support_a1 ; part 's1 ; create)
```

```
(i7 assembly 'support_a1 ; part 's2 ; create)
(i8 assembly 'support_a1 ; part 's3 ; create)
(i9 assembly 'support_a1 ; part 's4 ; create)

--import parts of the inner mallock frame; assign density of aluminum to
--all but the spring enclosure
--inner mallockbb
(define inner_mallockbb body)
(inner_mallockbb receive "inner_mallockbb")
(creatt (inner_mallockbb tag) 10 nil '(0.098) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'inner_mallockbb;ar)))

--inner mallockba
(define inner_mallockba body)
(inner_mallockba receive "inner_mallockba")
(creatt (inner_mallockba tag) 10 nil '(0.098) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'inner_mallockba;ar)))

--inner mallockta
(define inner_mallockta body)
(inner_mallockta receive "inner_mallockta")
(creatt (inner_mallockta tag) 10 nil '(0.098) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'inner_mallockta;ar)))

--inner mallocktb
(define inner_mallocktb body)
(inner_mallocktb receive "inner_mallocktb")
(creatt (inner_mallocktb tag) 10 nil '(0.098) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)        (printc "no graphics"))
 (t (graphics clear; sketch 'inner_mallocktb;ar)))

-- spring enclosure
(define spring_enclosure body)
(spring_enclosure receive "spring_enclosure")
(creatt (spring_enclosure tag) 10 nil '(0.28) '(density))
```

```
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)       (printc "no graphics"))
 (t (graphics clear; sketch 'spring_enclosure;ar)))

--sketch inner_mallock assembly
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)       (printc "no graphics"))
 (t (graphics clear; sketch '(inner_mallockba inner_mallockbb inner_mallockta
inner_mallocktb spring_enclosure) ;ar)))

--define the inner_mallock assembly
(define inner_mallock_a1 p_assembly)
(inner_mallock_a1 create)

--define the instances of components in inner mallock assembly
(define imi1 p_instance)
(define imi2 p_instance)
(define imi3 p_instance)
(define imi4 p_instance)
(define imi5 p_instance)
(imi1 assembly 'inner_mallock_a1 ; part 'inner_mallockbb ; create)
(imi2 assembly 'inner_mallock_a1 ; part 'inner_mallockba ; create)
(imi3 assembly 'inner_mallock_a1 ; part 'inner_mallockta ; create)
(imi4 assembly 'inner_mallock_a1 ; part 'inner_mallocktb ; create)
(imi5 assembly 'inner_mallock_a1 ; part 'spring_enclosure ; create)

--lower carriage
(define lower_carriage body)
(lower_carriage receive "lower_carriage")
--assign lower carriage the density for copper
(creatt (lower_carriage tag) 10 nil '(0.322) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)       (printc "no graphics"))
 (t (graphics clear; sketch 'lower_carriage ;ar)))

--upper carriage
(define upper_carriage body)
(upper_carriage receive "upper_carriage")
--assign lower carriage the density for copper
(creatt (upper_carriage tag) 10 nil '(0.322) '(density))
(cond
 ((eq graphics_on 'undefined) (printc "no graphics"))
 ((eq graphics_on nil)       (printc "no graphics"))
```

```
 (t (graphics clear; sketch 'upper_carriage ;ar)))

--define the entire set of parts, frame
(define frame_a1 p_assembly)
(frame_a1 create)

--define the instances of components in overall assembly
(define fi1 p_instance)
(define fi2 p_instance)
(define fi3 p_instance)
(define fi4 p_instance)
(define fi5 p_instance)
(define fi6 p_instance)
(fi1 assembly 'frame_a1 ; part 'inner_sphere ; create)
(fi2 assembly 'frame_a1 ; part 'heater_shell ; create)
(fi3 assembly 'frame_a1 ; part 'support_a1 ; create)
(fi4 assembly 'frame_a1 ; part 'inner_mallock_a1 ; create)
(fi5 assembly 'frame_a1 ; part 'lower_carriage ; create)
(fi6 assembly 'frame_a1 ; part 'upper_carriage ; create)
```

B Additional Images

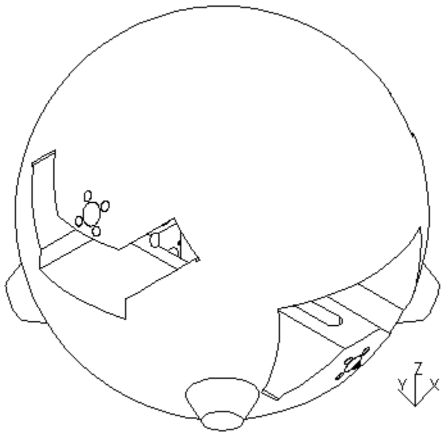
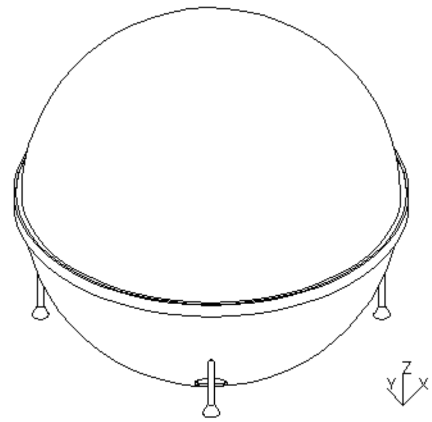


Figure 2. Inner Sphere with Mounting Points



F igure 3. Heater Shell with Supports

Figure 4. Support System for Inner Sphere

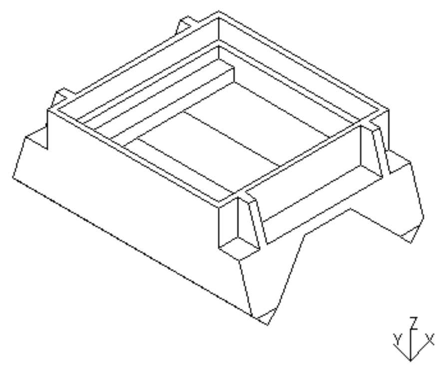


Figure 5. Lower Carriage

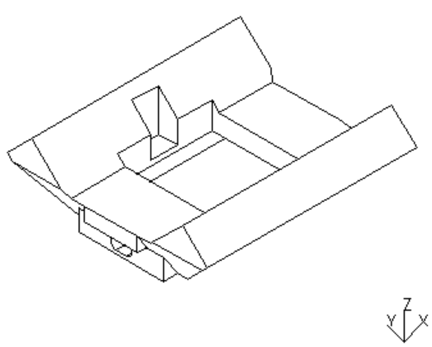


Figure 6. Upper Carriage

F igure 7. Inner Sphere with Supports

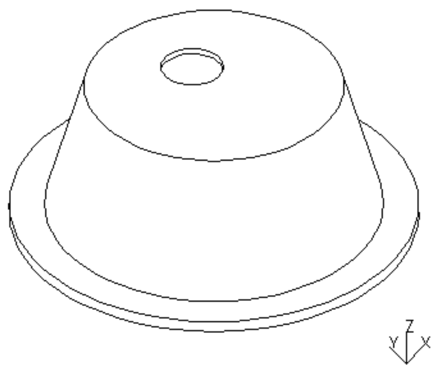


Figure 8. Spring Enclosure

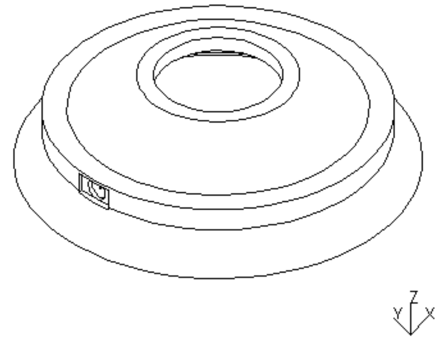


Figure 9. Inner Mallock Shell

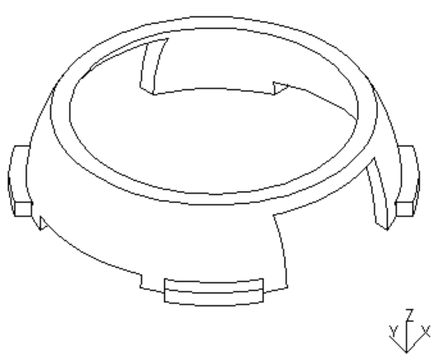


Figure 10. Inner Mallock Shell

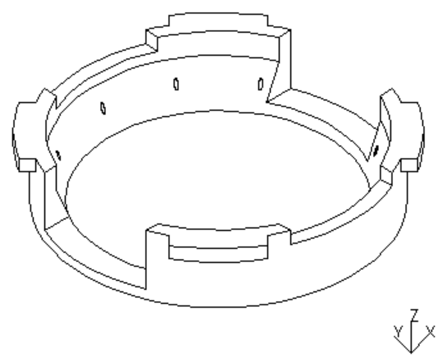


Figure 11. Inner Mallock Shell

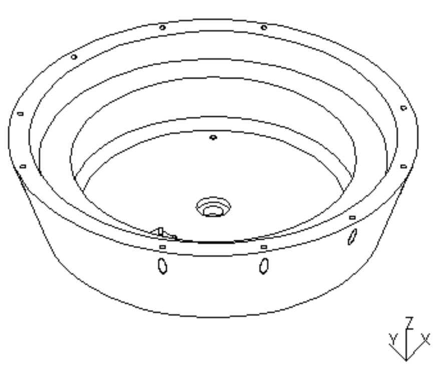


Figure 12. Inner Mallock Base

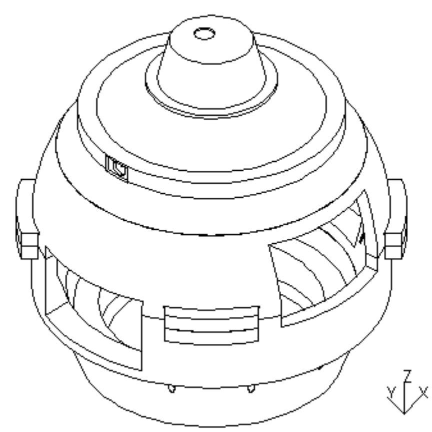


Figure 13. Inner Mallock Housing

Figure 14. Lower Half of Inner Sphere with Supports and Carriages

Figure 15. Cut-Away of Inner Mallock Assembly

Figure 16. Cut-Away of Inner Mallock and Heater Shell

Figure 17. Cut-Away Inner Mallock Assembly and Lower Carriage