APPLICATION OF NEURAL NETWORKS TO AUTOMATICALLY CLASSIFY ROTATIONAL PARTS INTO PART FAMILIES

by

HIREN H DESAI

B.E.(Production Engineering)., Shivaji University
Solapur, India, 1987

A THESIS

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Department of Industrial Engineering
KANSAS STATE UNIVERSITY
Manhattan, Kansas

1990

Approved by:

Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my advisor, Dr. Louis E. Grosh for his guidance and support throughout the course of this research. A special thanks to Dr. Bradley A. Kramer, for his expert and judicious guidance, unabating encouragement and enthusiasm during the course of this research.

I also wish to thank Dr. Doris L. Grosh and Dr. Prakash Krishnaswamy for consenting to be on my advisory committee. I thank my mother for her unfailing moral support.

CHAPTER 1

INTRODUCTION

1.1 Introduction

This thesis details the procedure for automatic classification of rotational parts into part families using an artificial neural network. The classification is based on geometric features and tolerances. The neural network paradigm employed belongs to a class of Adaptive Resonance Theory Models [15,16,17]. The training of the network was done on a commercially available software package [8].

1.2 Group Technology

Group Technology is a technique used in the small and medium batch manufacture of discrete components. A reduction in the setup time, throughput time, and Work in Process inventory are a few of many tangible advantages of Group Technology. Small batches and a large variety of components have prohibited the extensive use of automation. The primary prerequisite for implementing Group Technology is the grouping of parts into part families, and machines into machine cells. Many grouping techniques have been developed ranging from simple ocular techniques to complex pattern recognition techniques. The most recent work on part family groupings appear in the work of Ham et al. [44] and Grum [37]. These and other techniques will be discussed in Chapter 2.

There are several drawbacks to the existing grouping techniques. A few important drawbacks are listed below:

1. These techniques are application dependent. This means that separate algorithms have to be created for specific application domains.

2. The coding and classification of parts is the most time consuming process in implementing Group Technology. All the popular classification schemes require complicated coding procedures.

3. The number of groups into which the parts are to be classified are predetermined by the user. This requires expert knowledge of the company parts database.

4. A composite component must be created for every part family formed. The composite may not always give the best representation of a group.

5. Knowledge based classification techniques employ a time consuming process of knowledge acquisition and coding.

The above stated drawbacks provided the impetus to devise a classification scheme that would be free of the same.

1.3 Neural Networks

Neural networks are simplified models of the human brain, capable of learning, generalization, and abstraction. They are suited for achieving human like performance in fields such as speech processing, image cognition, machine

vision, autonomous navigation, and sensor processing. This research is concerned with pattern recognition. Adaptive Resonance Theory or ART2 is used as a pattern classifier in the present research. ART2 is an unsupervised learning paradigm and hence does not require a teacher to classify parts. Parts are automatically classified into part families based on geometric features and tolerances. The input to the network is a coded binary matrix of geometrical features and tolerances. The use of a simulated parallel process speeds up the task of pattern recognition considerably.

1.4 Advantages of the proposed approach

A few of the more important advantages of the proposed technique are given below:

1. The use of parallel processing (neural networks) speeds up the classification process.

2. The number of part families formed does not have to be predetermined by the user.

3. A composite or a reference component is created automatically within the system. This representative component automatically updated as learning proceeds.

4. An algorithm need not be created for every new application domain. That is, the system is application independent.

5. The system possesses the ability to learn from past

experiences.

1.5 Outline of Chapters

The next Chapter contains a detailed history and literature review of Group Technology. It also further explains the inadequacies of the existing methods of part family classification. A Literature review and the history of neural networks is covered in Chapter 3. The proposed technique and its advantages will be discussed in Chapter 4. The results, conclusion and recommendations for future work are given in Chapter 5.

Chapter 2

Group Technology

2.1 Introduction

In 1937, Sokolovski of Russia suggested that parts of similar features and configuration should be manufactured in the same way by a standardized technological process [79]. The definition of Group Technology: "Group Technology is a technique for manufacturing small to medium lot size batches of parts of similar process, of somewhat dissimilar materials, geometry, and size, which are produced in a committed small cell of machines which have been grouped together physically, specifically tooled, and scheduled as a unit" [79].

Group Technology is applicable to small and medium batch production. It endeavors to group together parts which require similar machines and manufacturing operations. The result is the formation of machine cells and part families. A large number of parts may be grouped into a smaller number of part families. A large number of machines may be grouped into a smaller number of machine cells. The application of Group Technology has many tangible and intangible benefits. The tangible benefits are quantifiable.

2.2 Advantages of Group Technology

Grouping in a manufacturing system enables the production of parts more economically than would be possible in a corresponding functional manufacturing facility. This is because of the following advantages of Group Technology over functional and job shop type manufacturing:

1) Group Technology simplifies the flow of parts and tools since the managing task is simplified at the cell level due to the reduction in the amount of information handled.

2) It reduces setup time, throughput time and work-in - process inventory. The setup time is reduced for individual parts. This is because a setup is not performed for each individual component but for all the parts under a specific group. The throughput time is reduced as a result of the reduction in the cycle time of individual parts in a part family. The work-in-process inventory is reduced because parts are manufactured on a dedicated group of machines which reduces cycle time, which in turn reduces work-in-process inventory.

3) It maximizes design and manufacturing efficiencies of parts, which are similar to those previously developed. The grouping of parts and the formation of part families avoids design duplication of new parts which are similar to the existing parts. The process plans for similar parts will in many cases be similar. This helps in solving process planning problems.

2.3 Approaches to Group Technology

There are two main approaches to Group Technology implementation. They are the classification approach and the clustering approach [79]. The difference between the classification and clustering approach is the criteria used for forming part families or machine cells. In the classification approach, parts are coded using numbers or letters or a combination of both, based on a classification system. Each number, letter or a combination represents a specific attribute such as shape and dimension, material or tolerance. In the clustering approach, clusters of parts and machines are determined based on specific attributes such as part design and routing information. The clustering techniques use a clustering algorithm to cluster parts. Most of the clustering techniques simultaneously form part families and machine cells. There are a variety of clustering and classification techniques available for implementing Group Technology.

2.3.1 The classification approach

Classification is a technique to organize specific data relating to the relevant component element(s) of a business or an institution in a logical and systematic hierarchy. It is an approach whereby like things are brought together by virtue of their similarities, and then separated by their essential differences [13].

Three rules must be adhered to strictly while applying classification to any population of parts.

1) It must be all-embracing

The classification scheme must embrace all existing parts in the database and must be able to accept necessary new items into the defined population of items. There must be scope to accommodate inclusions in the future.

2) It must be mutually exclusive

The classification must be such that all parts are mutually exclusive. There must be one and only one place for each item.

3) It must be based on permanent characteristics

The classification must be based upon visible attributes or easily confirmed permanent and unchanging characteristics of the parts.

Parts are coded using numbers or letters or a combination of both, based on a classification system [11]. All parts are classified on the basis of one or more of the following four characteristics:

1) The number and types of operations required.

2) The shape and dimensions of the parts.

3) The material of the part.

4) The tolerance requirement on the part.

Each part is assigned a ten-to thirty-digit code where each digit represents a specific attribute of the part [11]. The classification and coding schemes differ by the type of

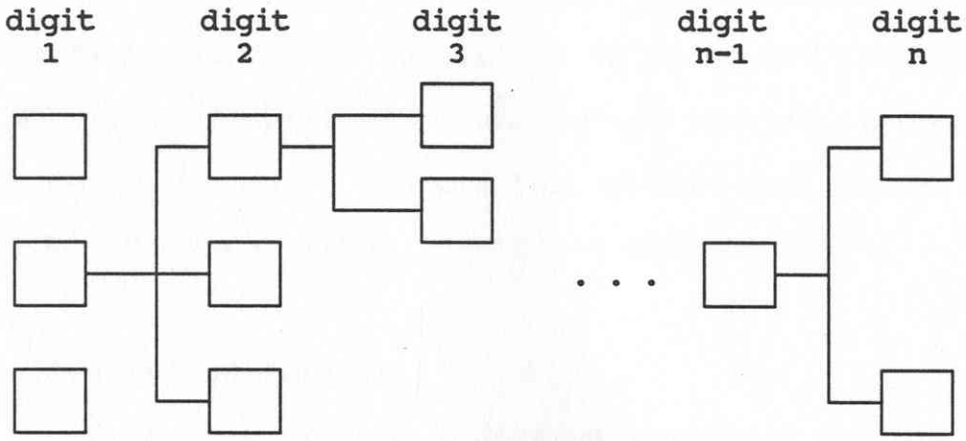


Figure 1(a). Hierarchical Code.

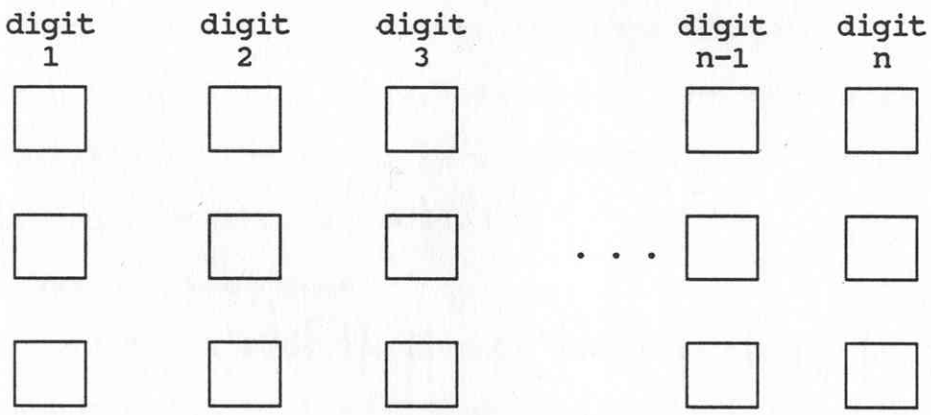


Figure 1(b). Non-Hierarchical Code.

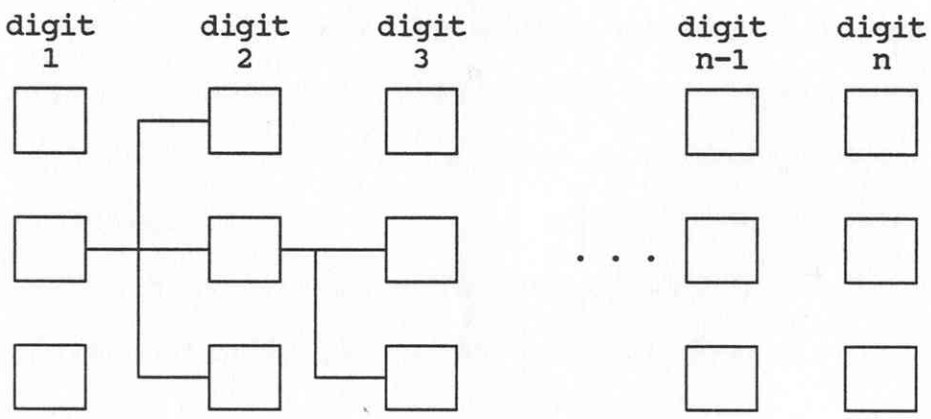


Figure 1(c). Hybrid Code.

information provided by the above mentioned characteristics. The part characteristics are coded using a variety of coding formats. Since each manufacturing system warrants a different classification criterion there are over 50 commercial classification and coding systems in operation worldwide. Some of the popular schemes are BIRSCH, DCLASS, TOYODA, MICLASS, TEKLA, OPITZ and NITMASH [11].

2.3.1.1 Coding formats

The coding format indicates the types of code and code structure that is used to code parts. There are three types of code formats. They are the hierarchical, non-hierarchical, and hybrid code. The hybrid code, as suggested by the name, is a combination of the hierarchical and non-hierarchical codes.

1) Hierarchical code.

In the hierarchical code the meaning of any digit depends upon its predecessor digits [11]. The coding pattern follows a hierarchy as shown in Figure 1(a). The presence of lines between digits indicates the existence of a hierarchical relationship among the digits. The absence of a line indicates the absence of a hierarchical relationship.

The hierarchical code (monocode) permits a large volume of information to be stored in a few digits [11].

Table 1. Clustering Approaches [27]

METHOD	COEF.	SEQ*	COM.+	LANGUAGE
I. DESIGN-ORIENTED APPROACH.				
o Multiobjective Clustering	S	Y	Y(M)	FORTRAN
Analysis	D	N	Y(S)	BASIC
o Matrix Formulation	-	-	-	-
o Dutta et al. Heuristic	D	N	Y(-)	-
II. PRODUCTION-ORIENTED APPROACH.				
A. Array-Based Techniques:				
o Rank Order Clustering	-	N	Y(-)	-
Method (ROC)	-	N	Y(-)	-
o ROC2 Algorithm	-	N	Y(-)	-
o MODROC Algorithm	S	N	Y(M)	FORTRAN
o Direct Clustering Algorithm	-	N	Y(-)	-
o Jacobs' Algorithm	-	Y	Y(S)	BASIC
o Modified Bond Energy Algorithm	S	Y	Y(S)	FORTRAN
o Occupancy Value Method	-	N	Y(-)	-
B. Hierarchical Clustering:				
o Single Linkage Method	S	N	-	-
	S,D	N	Y(-)	CLUSTAN
	S	N	-	-
	S	N	Y(S)	BASIC
	S	Y	Y(-)	FORTRAN
o Average Linkage Method	S	N	Y(-)	-
o Complete Linkage Method	S	N	-	-
	S,D	N	Y(-)	CLUSTAN
o Centroid Method	S	N	-	-
	S,D	N	Y(-)	CLUSTAN
o Median Method	S,D	N	Y(-)	CLUSTAN
	S	N	-	-
o Ward's Algorithm	S,D	N	Y(-)	CLUSTAN
o Lance & Williams Flexible	S	N	-	-
	S,D	N	Y(-)	CLUSTAN
o McQuitty's Similarity Analysis	S,D	N	Y(-)	CLUSTAN
C. Non-hierarchical Clustering:				
o Modified MacQueen's Method	D	N	Y(L)	FORTRAN
o A Divisive Procedure	-	N	Y(-)	-
D. Mathematical Programming:				
o Zero-One Integer	-	-	-	-
Programming	S	N	Y(L)	LINDO
o Dynamic Programming Based	S	Y	Y(-)	-
E. Graphic Theoretic Methods:				
o R & B Algorithm	S	N	Y(L)	FORTRAN
o De Witte's Algorithm	S	Y	-	-
o Purcheck	-	Y	-	-
o V & R Algorithm	-	N	Y(L)	FORTRAN
o C & R Algorithm	-	N	-	-
F. Heuristic & Others:				
o WUBC	-	N	Y(-)	-
o ICRMA	S	N	Y(L)	FORTRAN
o ODC	D	N	Y(-)	-
o MACE	S	N	Y(M)	FORTRAN
o Cluster Identification Algorithm	-	N	Y(L)	FORTRAN
o Cost Analysis Algorithm	-	N	Y(L)	FORTRAN
o Polyhedral Dynamics	-	N	Y(-)	-
o Mathematical Classification	S	N	Y(-)	-

* Considered operating sequence or not (Y/N) ?
\+ L: Mainframe; M: Minicomputer; S: Microcomputer.

2) Non-hierarchical code.

In the non-hierarchical code (polycode) the meaning of each digit is independent of any other digit (see Figure 1(b)). This code requires a large number of digits to store the required information, because each digit represents only a small part of the total information.

3) Hybrid code.

The hybrid coding system combines the hierarchical and non-hierarchical schemes as shown in Figure 1(c). The first digit in the figure may or may not have a relationship with the other digits as indicated by the line. In short, the digits n-1 and n may or may not be related.

2.3.2 Clustering Approach

In the clustering approach, clusters of parts and machines are determined based on design, manufacturing and routing information. Diverse clustering algorithms have been employed as an effective tool to solve the cell formation problem (refer Table 1).

There are a myriad of clustering approaches. The two most important ones are the design oriented and the production oriented approaches. The design oriented approach relies on the design information and design characteristic of parts. The production oriented approach is based on the routing information of parts. The two approaches are discussed here.

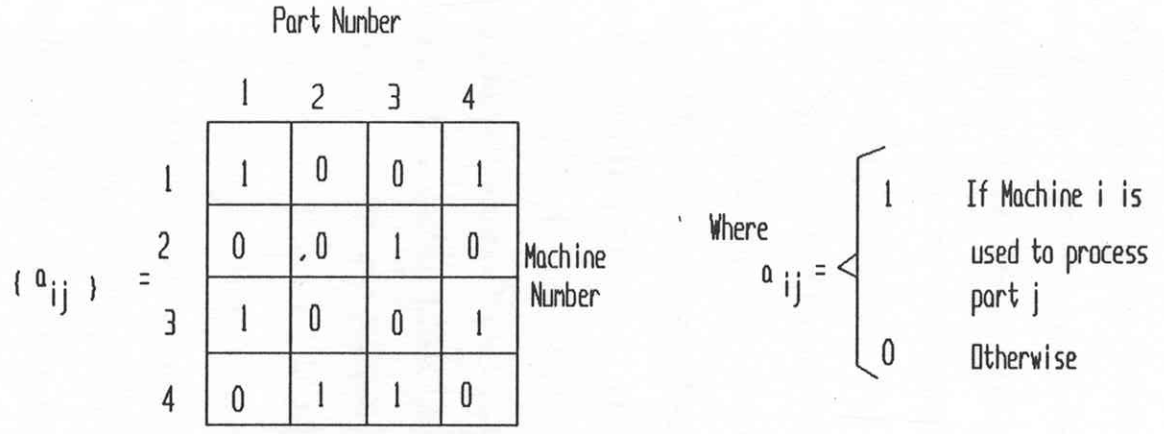


Figure 2(a), Machine Component Matrix (Initial)

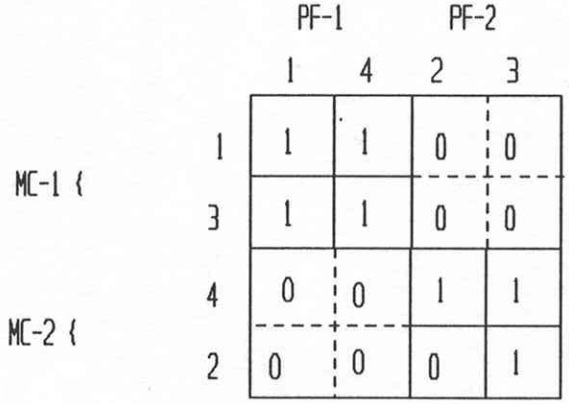


Figure 2(b), Machine Component Matrix (Resultant)

1) Design oriented approach.

The design oriented approach relies on the design features of parts to perform the necessary analyses. Grouping is done using part design information. The design information can be taken from a part blueprint or a CAD database. This approach ignores the routing and manufacturing information of parts. Techniques range from from the ocular approach to CAD driven or feature based models [14,29].

2) The production oriented approach.

The production oriented approach uses routing information to group parts into part families or machines into machine cells. This method predominantly uses a Binary Matrix called the Machine Component Matrix (MCM). The Machine Component Matrix has either a 1 or a blank in every cell. A 1 is entered in the cell C_{ij} if the part j uses machine i. The created matrix has a 1 or a blank in every cell (refer Figure 2(a)). The clustering techniques tries to rearrange the columns and rows of a machine component matrix according to an index until some diagonal blocks are formed (refer Figure 2(b)). Every cluster in the Machine Component Matrix represents a grouping in terms of part families and machine cells. The cell entries for all values of the i^{th} row and j^{th} column are $x_{ij} = 1$ or $x_{ij} = 0$ (shown as a blank entry in the matrix). A hierarchical clustering technique computes the similarity or dissimilarity between

each pair of parts or machines in order to produce a linkage or a relationship diagram for final judgment.

2.3.2.1 Clustering Criteria

Clustering is based on the calculation of a similarity or a dissimilarity index derived from binary or gray scale data. A similarity coefficient is used to measure the degree of similarity between parts or machines. The larger the coefficient the higher the degree of similarity between each pair of parts or machines. Conversely a dissimilarity coefficient measures the degree of dissimilarity between parts or machines. Most methods use the similarity coefficient.

2.4 Literature review

El-Essaway and Torrance [29], McAuley [62], Carrie [18], Rajgopalan [68,69], 1976, De Witte [28], King [45], King and Nakornchai [46], and many others have proposed different approaches for cell formation in Group Technology based on the concept of production flow analysis.
El-Essaway and Torrance [29] proposed component flow analysis for machine component cell formation in Group Technology [29]. Their paper presents a detailed analysis of components which is used for flow analysis. The components are analyzed for routing information and process plans. The flow pattern of components between machines is

used as a criterion for grouping. The drawback of the technique is that the formation of cells depends on heuristics and judgment [29]. In the same vein McAuley [62] used the technique of single linkage cluster analysis to form groups of machines having mutually high similarity coefficients. The difference was that machines are used as a criterion rather than components. The method however gives machine disjoint cells which are not compatible with the real life situation [69]. A part may need to be processed in more than one machine cell. A new approach was suggested by Carrie [18]. He applied a numerical taxonomy to Group Technology and plant layout. A numerical taxonomy is a method of analysis rather than a formula to be executed. In this approach, selection of the minimum cluster size or groups at a particular similarity level are arbitrarily decided [18].

The use of similarity coefficients has numerous drawbacks. Rajgopalan and Batra [69] proposed a graph theoretic approach for the design of cellular production systems. The selection of the threshold value for the similarity coefficient was arbitrary and required a certain amount of human judgment. De Witte [28] proposed the use of three types of similarity coefficient in production flow analysis. These coefficients showed the absolute relations and mutual interdependence among the parts. The selection of threshold values for these similarity coefficients is

arbitrary [14] and based on heuristics and human judgement. Moreover, this approach requires categorization of machines as primary, secondary and tertiary.

Algorithms developed by King [45] and King and Nakornchai [46] are computationally straightforward and do not use the concept of a similarity coefficient. On the other hand, King's algorithms consider binary positional weights (binary ranking) of the elements of the machine-component matrix. The positional weights are subject to variation if the position of matrix elements are changed. Under a non-pure diagonal block structure of cell formation, if these renumbered machines and components are serially placed, King's algorithms may not always give the minimal number of exceptional elements [62].

Thus the appraisal of the above mentioned methods indicate that the use of similarity coefficients in Group Technology have the following deficiencies.

1) They require almost arbitrary decisions.

2) They do not always give consistent results.

3) They do not always give the minimal number of exceptional elements.

An attempt to develop an algorithm which would yield the minimum number of exceptional elements was used by Waghodekar and Sahu [78]. The method is named "The Machine Component cell formation in Group Technology". This approach does not use arbitrary selection of the threshold

value for the similarity coefficient. It provides three outputs based on three different definitions of the similarity coefficient. The three different definitions of the similarity coefficient provides a cross check for consistent results. This approach is computationally straightforward and conceptually easy to understand. However it also has its drawbacks in that it cannot alone give foolproof solutions for all problems associated with machine component cell formation. This is because the use of three arbitrary similarity coefficient does not overcome the deficiencies of previous methods as mentioned earlier.

Many heuristic and nonheuristic methods are in practice for concurrent classification of parts and machines. An algorithm for the concurrent formation of part families and machine cells was advocated by M.P. Chandrashekharan and R. Rajagopalan [26]. In most of these approaches the primary input data is the machine component incidence matrix in which the columns represent the components and the rows represent machines. The Zero One Data: Ideal Seed Algorithm for Clustering (ZODIAC) elicits the best block diagonal form and computes its efficiency [26]. This enables the comparison of the results with any other solution. The drawback of this technique is that the choice of ideal seed is arbitrary.

The use of heuristics and human judgement led to the use of artificial intelligence to group parts into part

families and machines into machine cells. Andrew Kusiak [52] developed an expert system which evaluated partial solutions generated by a clustering algorithm and has an impact on its search directions. This approach takes advantage of the developments in expert systems and optimization techniques. Two basic components of the knowledge based system are the expert system and a heuristic clustering algorithm. All of the above mentioned techniques did not consider bottleneck machines and user specified constraints.

Y. Lemoine and B. Mutel [55] proposed automatic determination of production cells and part families based on a dynamic cluster algorithm. The method takes into account the capacity and the load of the machines and other user's constraints if required. The main characteristic of this method are:

1) It can analyze large data sets.

2) It takes into account past experience in the initialization procedure.

3) It defines a partition in K cells depending on constraints such as, similar machine tools and load capacity of the machine.

4) It can test the stability of the result.

5) It does not consider the user's point of view in the classification.

6) This method does not build hierarchical

classification and does not proceed by the exchange between rows and columns of the Machine Component Matrix (MCM).

The concept of design of components by variational geometry led to classification of components by variational geometry. David C. Gossard and Vincent Lin described a method to represent part families through variational geometry. It used a single representation to describe the entire family of geometries which share a generic shape. A shape model of a three dimensional component is defined with respect to a set of characteristic points. The positions of the characteristic points are fixed by a set of nonlinear algebraic equations which describe constraints imposed by dimensions. A modified form of the Newton-Raphson method is used to solve the set of constraint equations for the geometry.

The present trend in manufacturing technology indicates strong tendencies to integrate CAD/CAPP/CAM activities. It should also be based on similarity of tooling and routings of parts. These conditions require highly flexible Computer Integrated Manufacturing (CIM) processes. An important prerequisite for the realization of such an integrated system is the organization of parts in the database. It represents the basic information of parts, scheduling of operations, production and quality control of parts. A part identification number has been in use for a long period of time for efficient database management. Usually the

information content of an identification number is very limited and does not give information about the part characteristics, relevant to the design, planning and production of parts.

The part classification number contains coded information about the geometrical and technological information of parts. Grum and Pelenik [37] used the part identifier and geometric primitives to classify parts by a pattern recognition technique. The efficiency of the whole procedure is based on the development of a potential function and the coding of the part matrix.

The advantage of this approach is that it is not necessary to develop a classification system with fixed class numbers; the information for the classification is extracted by the designer directly and put into a matrix with black and white fields.

Inyong Ham et al. [44] published a working paper on an automatic classification scheme based on a supervised learning algorithm in neural networks. The network was used for design data retrieval and classification. There are several serious drawbacks of this neural network based classification technique:

1) The word "Automatic Classification" is a misnomer in Ham's work. This is because an omniscient user decides the number of groups the parts should be grouped under which is difficult in a real world situation.

2) The use of a supervised learning paradigm like a two-layer feed forward perceptron limits the number of exemplars that can be learned by the system. The network architecture has to be configured for the specific application.

3) The use of a new tool (neural networks) does not indicate its superiority over existing techniques. Ham's technique uses the concept of a potential function which is widely used by other pattern recognition techniques today. Moreover the development of a potential function is a time consuming process. It also implies that Ham's approach is not different from existing pattern recognition approaches.

4) Real time network models must do more than learn an associative map. The paper by Ham et al. is solely based on the principle of associative mapping. Real network models must do more than store distributed codes for a carefully controlled environment.

5) The architecture of popular supervised learning algorithms are often inadequate because they cannot self-organize.

6) Learning models which cannot adaptively cope with unpredictable changes in a complex input environment have an unpromising future as models of the mind and brain. They provide little hope for solving the outstanding problems in engineering which are not already handled by traditional methods of artificial intelligence.

2.5 Conclusions

In the 1970's the techniques developed for Group Technology were oriented around component flow analysis and production flow analysis. The drawback of these methods was that formation of cells depends on heuristics and personal judgement. This is useful if used in conjunction with an expert system which can deal with heuristic decision making. Further, the application of numerical taxonomy is not practicable for the classification problem because the selection of the minimum cluster size at a particular similarity level is arbitrary. This is not a analytical way to group parts or machine cells. The graph theoretic approach developed in the mid 1970's made use of the similarity coefficient. The threshold value for the similarity coefficient was again chosen arbitrarily and was based on judgement. Toward the late 1980's, algorithms were suggested for the concurrent formation of part families and machine cells. Later artificial intelligence was incorporated into this field and expert systems were used to evaluate the partial solutions obtained from clustering algorithms. Artificial intelligence techniques are based on heuristics and are suitable only for a specific application. Dynamic cluster algorithms were proposed to overcome this drawback.

The concept of modeling through variational geometry

Chapter 3

NEURAL NETWORKS

3.1 Introduction

Neural networks are an evolution of computing. At the inception of mechanized computing, programming was done using flip flop switches and plugging wires. With advances in computer hardware and software, computing advanced to machine code, assembly language and first generation FORTRAN. As time went by, fifth generation languages and tools such as LISP, Prolog, C++ and expert systems were developed. Now neural networks are increasingly being used because of their parallel processing capability. The evolution of neural network dates back to the early 19^{th} century. The commercial success of neural networks was realized only in the mid 1980's. Neural networks have been employed for tasks ranging from simple pattern recognition to complex explosive detection techniques at airline terminals and navigation of an Autonomous Land Vehicle.

3.2 Literature review

Neural network technology is a subset of Parallel Distributed Processing (PDP) Technology. Some of the earliest roots of the PDP approach can be found in the work of the neurologists, Jackson [42] and Luria [57]. Jackson was a forceful and persuasive critic of the simplistic

localizationist doctrines of the late nineteenth century neurology, and he argued convincingly for distributed, multilevel conceptions of multilevel processing systems. Luria, the Russian psychologist and neurologist, put forward the notation of the dynamic functional system. In his view, every behavioral or cognitive process resulted from the coordination of a large number of different components, each roughly localized in different regions of the brain, but all working together in dynamic interaction. A rough characterization of the kind of parallel distributed processing system of today is seen in their ideas.

Two other contributors to the deep background of PDP were Hebb [38] and Lashley [54]. Hebb introduced the concept of cell assemblies a concrete example of a limited form of distributed processing and discussed the idea of reverberation of activations within neural networks. Lashley's contribution was to insist upon the idea of distributed representation. Lashley insisted that "there are no special cells reserved for special memories".

In the 1950's, there were two major figures whose ideas have contributed to the development of their approach. One was Rosenblatt [70,71] and the other was Selfridge [75]. In his **Principles of Neurodynamics** [71], Rosenblatt articulated clearly the concept of a neurally inspired approach to computation and he developed the perceptron convergence procedure, an improvement over the Hebb rule for changing

synaptic connections. Rosenblatt's vision of the human information processing system as a dynamic, interactive, self-organizing system lies at the core of the PDP approach. Selfridge's contribution was his insistence on the importance of interactive processing, and the development of Pandemonium, an explicit computational example of a dynamic, interactive mechanism applied to computational problems in perception.

In the late 1960's and 1970's, serial processing and von Neumann computers dominated both psychology and artificial intelligence. Grossberg's mathematical analysis of the properties of neural networks led him to many insights which other researchers have only come to appreciate through extensive experience with computer simulation. He deserves credit for seeing neurally inspired mechanisms in many areas of perception and memory well before the field was ready for these kinds of ideas. Grossberg [35] was one of the first to analyze properties of competitive learning mechanisms. Anderson's [3] work differed from Grossberg's and insisted upon distributed representation, and in showing the relevance of neurally inspired models for theories of concept learning [4,5]. Anderson's work also played a crucial role in the formulation of the cascade model [63,64], a step away from serial processing down the road to PDP. Longuet-Higgins and his group at Edinburgh were also pursuing distributed memory

models during the same period. David Willshaw, a member of the Edinburgh group, provided some very elegant mathematical analysis of the properties of various distributed representation schemes [80]. His insights provided one of the sources for the idea of coarse coding. Many of the contributions of Anderson, Willshaw, and other parallel distributed processing modelers may be found in Hinton and Anderson [39]. Others who have made important contributions to learning in PDP models include Amari [2,3], Bienenstock, Cooper, and Munro [12] , Fukushima [32,33], Kohonen [48,49], and von der Malsburg [77].

Toward the middle of 1970s, the idea of parallel distributed processing began to have something of a renaissance in computational circles. The HEARSAY model of speech understanding played a prominent role in the development of neural networks. Unfortunately HEARSAY's computational architecture was too demanding for the available computational resources and so the model was not a computational success. But its parallel interactive character inspired the interactive activation model of reasoning [72,73], and the interactive model of word recognition [63,64].

The ideas represented in the interactive activation model had other precursors as well. Morton's logogen model [67], was one of the first to capture concretely the principle of interaction of different sources of

information, and Marlsen-Wilson [59] provides important empirical demonstrations of interaction between different levels of language processing. Levin's [73] Proteus model demonstrated the virtues of activation-competition mechanisms, and Glushko [73] helped display how conspiracies of partial activations could account for certain aspects of apparently rule-guided behavior. Carpenter and Grossberg [15,16,17] developed the Adaptive Resonance Theory Models (ART1, ART2) of Human Memory. Grossberg proved the models with rigorous mathematical analysis and proofs. Grossberg's work was truly the first in the area of unsupervised learning. Kohonen [51] also proposed a model of unsupervised learning. Research and application has shown that the Kohonen models are biased in favor of the winning element. Thus true classifications are not possible to model. Conversely Carpenter and Grossberg proposed the more practicable model. ART-1 accepts only binary inputs and ART-2 accepts both binary as well as gray scale. Moreover the ART models represents the human brain model very closely. The concept of functional link nets was introduced by Yoh-Han Pao [81]. His approach tries to use a flat net to store representation. One of the advantages of flat net is that both supervised and unsupervised learning can be carried out using the same net architecture. This facility avoids the need to shuffle the data in moving it from one paradigm to the next. Neural networks have a high

application potential in the manufacturing environment. Specifically, research has been done in applying this technology to process control, work flow control, and inspection. In process control they are used for real time monitoring of machining centers. The parameters monitored are the tool condition, tool temperature, etc [19,32]. The network can be trained to learn associations between situations and the appropriate feedback or conclusions. In the flow control application the network acts as a communication device between the machines and the controls. Here again the same property of association is used. Neural networks work on the principle of parallel processing. This characteristic is useful in the inspection task in defects are simultaneously detected and causes determined. Inspection can also be possible in real time by continuous monitoring. There have been no known applications of ART2 to manufacturing in the existing literature.

3.3 Neural Networks Technology

Dr. Robert Hecht-Nielsen, the inventor of one of the first neurocomputer defined a neural network as a computing system made up of a number of simple, highly interconnected processing elements, which process information by dynamic state response to external inputs [20]. In its most basic form, a serial computer is a single, central processor that can address an array of memory locations. Data and

instructions are stored in the memory locations. The processor fetches the instruction and any data required by the instruction, executes the instruction and saves any results in the specified memory location. A serial system (even a standard parallel one) is essentially sequential: everything happens in a deterministic sequence of operations. In contrast, a neural network is neither sequential nor even necessarily deterministic. It has no separate memory array for storing data. The processors that make up a neural network are not highly complex central processing elements. Instead a neural network is composed of many simple processing elements that typically do little more than take a weighted sum of all the inputs. The neural network does not execute a sequence of instructions; it responds, in parallel, to the inputs presented to it. The result is not stored in a specific memory location, but consists of the overall state of the network after it has reached some equilibrium condition. Knowledge within a network is not stored in a particular location. It is not possible to look at a memory address to retrieve the current value of any variable. Knowledge is more a function of the network's architecture or structure than the contents of a particular location.

Neural networks technology is a statistically based mapping technique. It has been mathematically proven that neural networks (of arbitrary complexity) can produce a

continuous mapping from an n-dimensional space to an m-dimensional space [21]. Examples include mapping from historical loan application to loan profitability data, from sonar signals to friend-or-foe identification, and from video input to a pass/fail on an assembly line. Since neural networks can produce these mapping without a human having to analyze the data algorithmically. They are economically appealing. Neural networks require a statistically valid representation of the solution to the problem.

3.4 Neural Network Architecture

The inspiration behind neural network architecture came from studies of the mammalian brains, particularly the cerebral cortex.

There are eight major aspects of a neural network architecture [73].

1) A set of processing units.
2) A state of activation.
3) An output function for each unit.
4) A pattern of connectivity among processing elements.
5) A propagation rule for propagating patterns of activities through the network of connectivity.
6) An activation rule for combining the inputs impinging on a unit to produce a new level of activation for the unit.
7) A learning rule whereby patterns of connectivity are

was advocated in early 1983. The creation of a composite component was also used to describe the entire family that share a common shape. The introduction of artificial intelligence techniques sparked a new way of looking at problems. Methods based on heuristics were employed. Pattern recognition was also investigated for this purpose. Grum and Pelenik [44] introduced the concept of coding features into a matrix. Ham et al. used a supervised learning paradigm to perform component data retrieval and grouping. Each of these methods has drawbacks which have inspired this research.

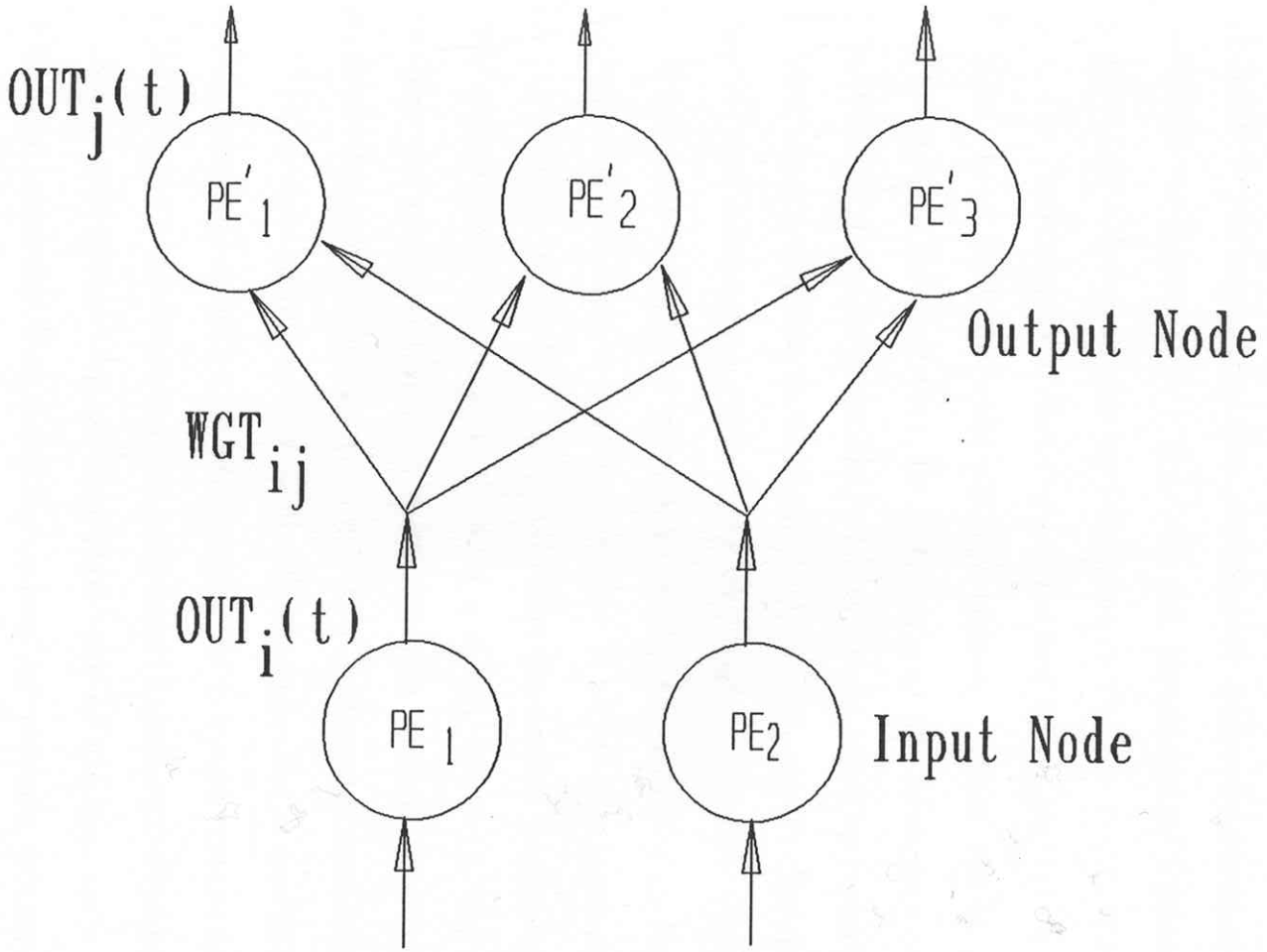


Figure 3. A Typical Neural Network Architecture

modified by experience.

8) An environment within which the system must operate.

Figure 3 illustrates the schematic representation of the network functioning. The set of processing elements are generally represented by circles.

At any instant each processing unit PE_i (i = 1,2,3....n) has an activation value. This value is denoted in Figure 3 by $ACT_i(t)$ corresponding to each processing element. This activation value is passed through a function $f(ACT_i)$ to produce an output value $OUT_i(t)$. The output value can be seen as passing through a set of unidirectional connections to other processing elements in the system. Every connection has a weight or a strength associated with it, denoted as WGT_{ij}, which determines the degree by which the first unit will effect the second unit (j represents the number of processing elements in the output layer). The higher the connection weight, the stronger the effect.

All the connection ends impinging on a particular processing element are then combined by an operator (usually addition). Therefore the combined input to a unit is $\sum_i(WGT_{ij}*OUT_i(t))$. This combined input to the unit along with its current activation value is passed through a function to determine the new activation value $ACT_j(t)$ of the output processing element PE_j (j = 1,2,3....m). These systems are viewed to be plastic in the sense that patterns of interconnections are not fixed for all time; rather, the

weights can undergo modification as a function of experience [1]. In this way the system can evolve. What a unit might represent after experience may be entirely different from what it might represent at the beginning. Therefore the system dynamically changes to perform in different ways. The elements of a typical neural network architecture are explained below.

1) A set of processing units.

Any parallel activation model begins with a set of processing units or processing elements also known as artificial neurons [73]. Specifying the set of processing elements and what they represent is the first stage of specifying a neural network model. In some models these processing elements may represent conceptual objects such as features, letters, words or concepts, or even feature detectors; in others they are simply abstract elements for which meaningful patterns can be defined. In this case it is the pattern as a whole that is the meaningful level of analysis. This can be contrasted to a one-one concept representational system in which single processing elements represent entire concepts or large entities.

The processing elements in all the layers are ordered arbitrarily. The i^t unit in the input layer is denoted by PE_i. The j_t element in the output layer is denoted by PE_j A unit's job is to simply receive input from its neighbors and to compute an output value as a function of the input it

receives, which it sends to its neighbors. The system is inherently parallel in that many processing elements can carry out their computations at the same time. Within any system to be modeled, three types of processing elements are included: input, output and hidden. Input processing elements receive inputs from sources external to the system under study. These inputs may be either sensory inputs or inputs from other parts of the processing system in which the model is embedded. The output processing elements send signals out of the system. The hidden processing elements are those inputs and outputs which are constrained to be within the system. They are not visible to the outside environment or system. These hidden units are responsible for storing features and knowledge in many supervised learning paradigms. These hidden units also represent the computational capability of the network. The next important part of the architecture is the state of activation of the network.

2) The state of activation.

In addition to the set of processing elements, the representation of the state of the system at time t is important. This is specified by a vector N, the number of processing elements in the spectrum of real numbers. Thus $ACT_N(t)$ represents the pattern of activation of the set of processing elements. Each element of the vector represents the activation for one of the processing elements at time t.

The activations of unit PE_i at time t is designated as $ACT_i(t)$. It is the pattern of activation of the set of processing elements that captures what the system is representing at any time. Different models make different assumptions about the permissible activation values. Activation values may be continuous or discrete. If they are continuous they may be bounded or unbounded. If they are discrete they usually take binary values. A digit 1 means that the unit is active, and a 0 means that the unit is inactive.

3) Output of the processing elements.

Processing elements interact by transmitting signals to their neighbors via the axon and the synapse. The strength of each signal depends on the degree of activation of the signal emitting unit. Associated with each input unit there is an output function $f(ACT_i)$ which maps the current state of activation $ACT_i(t)$ to an output signal $OUT_i(t)$. In some of the models the output level is the same as the activation level. In this case, function $f(ACT_i)$ is the identity function, $f(ACT_i(x)) = x$. More often x is some sort of a threshold function, so that a unit has no effect on the other unit unless its activation exceeds a certain value. Sometimes the function f is assumed to be a stochastic function in which the output of the unit depends in a probabilistic fashion on its activation level.

4) The pattern of connectivity.

Processing elements are connected to one another. The pattern of connectivity determines what the system knows and how it will respond to an input. In many cases each unit provides an additive contribution to the input of the other processing elements to which they are connected. In such a case the total input to the unit is simply the weighted sum of the separate inputs from each of the individual processing elements. In this case the total pattern of connectivity can be represented by merely specifying the weights for each of the connections of the system. A positive input represents an excitatory input and a negative input represents an inhibitory input. Some more complex excitation/inhibition combination rules are required.

5) The rule of propagation.

There is a need for a rule which takes the output vector and combines it with the connectivity matrices to produce a net input for each type of input. There are two types of connections in the connectivity matrix. Let TOT(i) be the net input of type i to a unit, and TOT(e) be the net input of type e to a unit. The propagation rule is generally straightforward. The net excitatory input is usually the weighted sum of the excitatory inputs to the unit PE_j from unit PE_i.

This is given by the vector product

$$TOT(e)_{ij} = \sum_i WGT(e)_{ij} * OUT_i(t)$$

where

$TOT(e)_{ij}$ = The total net excitatory input from unit PE_i to unit PE_j.

$WGT(e)_{ij}$ = The excitatory weight of the connection from unit PE_i to unit PE_j.

$OUT_i(t)$ = The output function of the unit PE_i at time t.

Similarly the net inhibitory input is the weighted sum of the inhibitory inputs to unit PE_j from unit PE_i, i.e.

$$TOT(i)_{ij} = \sum WGT(i)_{ij} * OUT_i(t)$$

where

$TOT(i)_{ij}$ = The total net inhibitory input from unit PE_i to unit PE_j.

$WGT(i)_{ij}$ = The weight of the connection from unit PE_i to unit PE_j.

$OUT_i(t)$ = The output function of the unit PE_i at time t.

6) Activation Rule.

A rule is needed whereby the net inputs of each type impinging on a particular unit are combined with one another and with the current state of the unit to produce a new state of activation. We need a function which takes the current activation of the units and the net vector TOT(e) or TOT(i) for each different type of connection and produces a new state of activation. If the function is an identity function and if all the connections are of a similar type the new state of activation can be represented by.

$$ACT_j(t+1) = \sum WGT_{ij} = TOT(N)$$

where

TOT(N) = The net combined activation of the each unit. In many real application situations the function **F** is a type of a threshold function which allows the unit to contribute only if the activation exceeds the threshold value.

7) Modifying patterns of connectivity as a function of experience

Modifying the patterns of connectivity implies changing the knowledge structure in a parallel distributed processing system. There are principally three kinds of possible modifications:

1) The development of new connections.

2) The loss of existing connections.

3) The change in strength of a connection that already exists.

Very little work has been done on 1 and 2. 1 and 2 can however be considered a special case of 3. Whenever we change the strength of a connection away from 0 to some positive value, it has the same effect as growing a new connection. Whenever we change the strength of a connection to 0, it has the same effect as losing an existing connection. There are several rules to modify connection strengths as a function of experience. Virtually all supervised learning paradigms follow the Hebbian rule or a variant of it.

8) Representation of the environment:

It is crucial in the development of any model to have a clear model of the environment in which this model is to exist. Software that defines these aspects of neural network architecture to generate a network and solve a specific problem is called netware. The next section will contrast neural network technology with conventional computing technology.

3.5 Contrasting Neural Networks with Conventional Computing

A netware programmer does not specify an algorithm to be executed by each processing element as a programmer of a more traditional machine would. Instead the programmer specifies the interconnections, transfer functions, and training laws of the network. The network programmer then applies appropriate inputs to the network and lets it react. If the netware is correctly written, the overall state of the network after it has reacted to the input will be the desired response pattern. In short, neural network programming differs fundamentally from standard programming techniques. Neural networks really are a completely different way of looking at computer systems. Neural networks do not execute programs as would a conventional computer application. They react, self organize, learn and forget. Why is there a need to build such odd systems? Over the past two years, interest in neural networks has

surged from a whisper to a roar. Why? Frequently, traditional computing and conventional Artificial Intelligence have found themselves on the rocks of computationally explosive problems and unbounded searches. They have run into the von Neumann bottleneck because many problems are naturally parallel. Neural networks are good at solving the kinds of problems people can solve easily. They are also poor at solving the kinds of problems that traditional computers do well. In general, neural networks do not do well at precise numerical computations. On the other hand this kind of computation is not a natural application for people either. Neural networks can, however, be taught to determine whether or not a visual image of the face is that of a man or a woman, or recognize a person's face, even with a different expression or hairdo. The role of neural networks is to be a partner to conventional computing systems, not a replacement for them. The neurocomputers that have been introduced have mainly been designed as co-processors working in conjunction with other sequential computing systems. They are operated by calling subroutines or procedures when a network application is encountered.

The Feigenbaum bottleneck compounds even the programmer's bottleneck not only must we wait for programs to be written, but we must also wait as Knowledge Engineers extract knowledge from domain experts. Recently researchers

focussing their efforts on neural networks have produced impressive results. Consequently, interest in the field has increased exponentially. An unusual characteristic of neural network is its interdisciplinary nature. Neural networks conferences are attended by engineers, neurophysiologists, psychologists, optical specialists, and even philosophers. Such catholic interest reflects a growing conviction in academia, government and industry that neural networks are not just another computational technique, but instead represent a major breakthrough a fundamentally different mode of computation with major advantages and wide applications.

It is easy to anthropomize neural networks beyond rational justification but not even the most ambitious advocates of neural networks are synthesizing human brain functions.

3.6 Training Methodologies

A neural network can learn either under the supervised training mode or unsupervised training mode. The particular type of training mode to be used depends on the specific application. For example, consider the task of classification of parts into part families based on geometric features and tolerances. The supervised training mode is applicable if the number of groups into which the parts are to be classified is known. Also the part database

is assumed to remain the same without any changes in the parts. In a real life situation these assumptions are not valid. This is because the fixing of the number of groups is an arbitrary choice. The parts database is bound to change in this competitive and rapidly advancing world. Thus a decision was made to use the unsupervised training mode for the proposed application.

3.6.1 Supervised Training

A set of training pairs of patterns are required in order to train a neural network with supervised training. Each training pair consists of an input pattern and a corresponding desired output pattern. During the training period an input pattern is presented and the network responds with some output pattern which may be different from the expected output pattern. The difference between the actual and the expected output pattern is called error. This error is fed to a predetermined training algorithm which modifies the network parameters to minimize the error. The training continues until the error for all training pairs is an acceptable value. The completion of the training phase signals that the network is ready to perform the desired function. In real life situations it is not possible to have a prior knowledge of the desired output for every input to the network. In such cases unsupervised training is useful.

3.6.2 Unsupervised Training

Unlike supervised training, the desired output for each training input pattern is not required. Whenever an input pattern is presented, the neural network does two things. First, it will respond with a certain output. Secondly, it will modify the network parameters so that the chance of responding to similar input patterns is reinforced. Thus the training process extracts the statistical properties of the training set and groups similar input patterns or parts into classes or families. This type of training is a biologically more plausible training mechanism than the other well known supervised training algorithms such as backpropagation.

3.7 Characteristics of neural networks.

There are four salient features of neural networks that make it an interesting technology for real world applications. They are generalization, abstraction, speed and multiprocessing.

3.7.1 Generalization

A frustrating characteristic of conventional computers is the literal, precise inputs required to produce the desired output. Neural networks can accommodate variations in their input and still produce the correct output. For example, a system trained to recognize printed letters did

so even when noise corrupted 40% of the input characters [33]. That is, the system recognized letters despite never having seen anything like them before much as humans understand incomplete and partially incorrect input. Studies show that most people can read text in which more than half the letters are obliterated [33].

The real world rarely presents information with the precision required by a computer program. Of course conventional computers have been programmed to tolerate noisy input, but the computational load often precludes using these algorithms in practical applications. Neural networks accomplish the needed generalizations by virtue of their structure rather than through elaborate programming (which tend to be application dependent). As such, neural networks provide a far more natural interface to the real world a world including human users.

3.7.2 Abstraction

Neural networks can abstract the "ideal" from a non-ideal training set. Such abstracting ability dates back to Plato's Republic and the Platonic concept of ideals. How do we determine that a given animal is a dog when every dog we have seen is different? Do we have an internal model an ideal dog to which we compare all instances?

3.7.3 Speed

One can view neural networks as associative memory,

associating input patterns with desired output patterns. When new patterns are presented to the input, associated output patterns are produced at the output. In neural networks the time required to produce outputs is independent of the number of associations stored. Thus nothing corresponds to a search time; the only time required is that associated with network stabilization a constant in most architectures. A given network storing ten million associations is just as fast as one storing ten thousand. Speed is achieved through multiprocessing.

3.7.4 Multiprocessing

Many computer architects feel that today's fastest computers operate within a factor of 10 of a single processor's theoretical speed limits. For this reason, a major effort toward multiprocessing exists. The effort is based on dividing problems into sub-problems, each of which can be assigned to separate processors. Inherently, neural networks schedule themselves, that is, each node can be viewed as a processor operating on its inputs independently of all other processors in the system. Thus while the network converges to a solution, all processors are busy, hence no expensive silicon remains idle. Furthermore processors can be added in a modular fashion to suit problem size without restructuring the system.

3.8 Strengths of the PDP approach

The PDP approach offers a very distinctive counter approach to conventional computing. It is an adaptive system, continually trying to configure itself so as to match the arriving data. It works automatically to adjust its own parameters so as to accommodate the input presented to it. It is a system that is flexible, yet rigid. That is, although it is always trying to mirror the arriving data, it does so by means of existing knowledge, existing configurations. It never expects to make a perfect match, but instead simply tries to get the best match possible at any time. The closer the match the more stable the system. The result is that although the system develops neither rules of classifications nor generalizations, it acts as if it had these rules. Thus, the system really mirrors experience; the regularities of its operation results from the regularities of the inputs and partially from the interpretations of the beholder. It is a system that exhibits intelligence and logic, yet has no explicit rules of intelligence or logic. The way by which these systems try to accommodate themselves to the data by minimizing the energy or maximizing harmony results in preferred states or interpretations, where the preferences reflect the particular events the system has experienced. This leads to categorization and classification of the input signals by distance from the prototypes. It is a system which

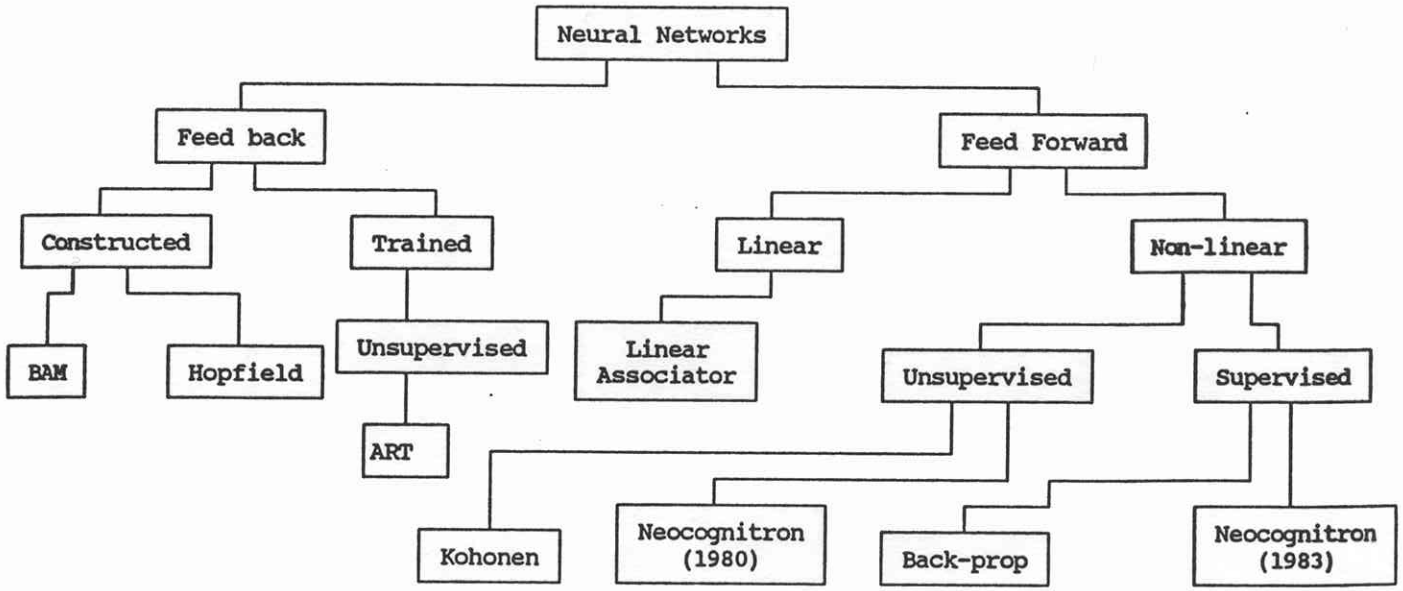


Figure 4. Classification of Neural Network Paradigms [6]

incorporates learning as a fundamental and essential aspect of its behavior. It makes no attempt to make categories or rules, yet it acts as if it were a prototype-matching or a categorization system that has explicit rules and strategies.

Neural networks can be arbitrarily categorized by topology, neuron model and training algorithm. Figure 4 shows one method of classifying neural networks. There are two main subdivisions of neural networks models:

1) Feedback
2) Feedforward

Feedback models can be constructed or trained. In a constructed model the weight matrix is created by taking the outer product of every input pattern vector with itself or with an associated input, and adding up all the outer products. After construction, a partial or inaccurate input pattern can be presented to the network, and after a time the network converges so that one of the original input patterns is the result. Hopfield and BAM are two well known constructed feedback models. The Hopfield network is a self-organizing, associative memory network. It consists of a single layer of neurons. This single layer acts both as an input as well as an output. The neurons can only take two values, -1 and +1. Hopfield networks can recognize patterns by matching new inputs with the closest previously stored pattern. These networks are used in applications

requiring some form of content addressable memory. A serious limitation of this network is the maximum number of memories which can be stored. In addition the hardware efficiency is poor. A variation of this is the Hamming network. The Bidirectional Associative Memory (BAM) network is a generalization of the Hamming network.

A trained feedback model like ART2 is much more complicated because the adjustment of the weights affects the signals as they move forward as well as backward. The ART model is a complex trained feedback paradigm. It is powerful but the number of patterns that can be stored is limited by the number of processing elements in the storage layer. No production application has been published to date. ART2 is presently considered to be a research tool.

The second division of neural networks is the feed forward category. The earliest neural network models were linear feedforward. The linear associator uses the simple delta rule. The system works very well if the maximum number of patterns to be stored is 10-20% of the number of neurons. There are two main types of training algorithms: supervised and unsupervised. Supervised learning is the most elementary form of adaptation. It requires prior knowledge of what the results should be. During training the network's output is compared to the ideal response, and any error is used to correct the network. Unsupervised learning differs in that it does not have specific

corrections made by comparison to ideal results. Supervised and unsupervised learning are methods which are mutually exclusive.

Backpropagation is useful because it provides a mathematical explanation for the dynamics of the learning process. The biggest limitation is the size of the network. A popular unsupervised feed-forward model is the Kohonen model. The basic system is a one or two dimensional array of threshold type logic units with short range lateral connections between neighboring processing elements. The neuron whose weight vector generates the largest dot-product with the input vector is the winner and is permitted to output. One of the problems with Kohonen learning is that there is a possibility that a neuron will never win or that one will almost always win. The weight vector gets stuck in isolated regions. One of the ways out of this is to give the neurons a conscience. If the neurons realize that they are winning a lot, they will step out of the competition for a while.

A special case of the feed-forward model is the Neocognitron. The original model was unsupervised, but a more recent model uses a teacher. After learning is completed, the final Neocognitron system is capable of recognizing handwritten numerals presented in any visual field location, even with considerable distortion. The major drawback of the Neocognitron is that it is highly

specialized and requires a large number of neurons and connections.

3.9 Weakness of the PDP approach

The Parallel Distributed Processing System has some weaknesses. In part, it is hard to apply it exactly to many of the difficult issues in study and research. In general the closer to perception or to motor output, the easier it is to apply. Thus there seems to be no question about its application to pattern recognition, vision, speech understanding, or categorization. The PDP models have the power to generalize. But the complimentary skill of keeping individual instances separate seems much harder. Researchers worry, and say that PDP processing elements cannot compute and analyze without variables. Aren't variables necessary? How about thought? If a problem is solved mentally a person has to postulate hypothetical situations, evaluate them and make decisions. How does a person compose music? Doesn't the person need to have mental variables, symbols that he can manipulate? This is the major deficiency of the PDP approach. Researchers argue that this problem can be solved by having several levels of the systems, each specialized in a specific domain. The PDP system is fine for perception, categorization and motor control. It is possibly the sort of system that models our automatic, subconscious reasoning. But at this stage more

research is required to handle problems of conscious, deliberate thought, planning, and problem solving.

When it comes to learning it is frequently the case that something has to watch over the operations and act as a trainer. But this trainer is different from learning mechanisms. It has to be able to evaluate the quality of performance. How does this take place? What is this second overseeing mechanism? And how did it get started? How did the trainer know what task to train, and when. And how did it acquire the knowledge of what is good performance if it was a task the person had never performed before? Even in the competitive learning mechanism where the learning can take place without an overseer, evaluatory mechanism, it is often advantageous to train the system by careful presentation of the items that are to be classified.

The PDP system is highly parallel and very fast when viewed at the level of computational operations. Conversely it is highly serial and relatively slow when viewed at the higher level of interpreting and analyzing the resulting state changes in the system. This dual virtue is similar to human cognition. People interpret the world rapidly, effortlessly. But the developments of new ideas, or evaluation of current thoughts proceeds slowly, serially and deliberately. People seem to have at least two modes of operation, one rapid, efficient, subconsciousness, the other slow, serial and conscious. The problem, however, is that

people can do multiple activities at the same time, some of them quite unrelated to one another. So researchers say that a PDP model of the entire human information processing system is going to require multiple processing elements. That is, the complete model requires that the brain consist of several independent PDP-like systems, each of which can only settle into a single state at a time.

3.10 The Future of Neural Networks

Neural networks are a rediscovered field experiencing an explosive growth in research and application interest. Algorithms and architecture proliferate. Claims and counter claims fill the literature. Despite its longevity, neural network theory and technology is rudimentary. There are more questions than answers, as technical knowledge remains narrowly disseminated. The situation resembles the Laser when it was introduced. The laser had such unique properties that many people felt it must be of immense value. Nevertheless, even to develop a small percentage of its commercial potential required nearly a decade. If this analogy is valid, some time will pass before neural networks find applications where their unique characteristics make them the clear method of choice. Meanwhile all parties researchers, commercial firms, and the press must understand the risk of promising more than can be delivered. A few industrial and military applications have been found.

Researchers and industry experts foresee an explosive growth in the use of neurocomputers in image and signal processing, recognition, and expert systems for financial, medical, and scientific uses. Industrial use would include quality control and process control. In the field of neurobiology, neurobiologists use neurocomputers to model imbalances in neuro transmitters and thus seek explanations for psychiatric disorders and the effects of psychoactive drugs. Researchers at a Pharmaceutical company are using neuro-computers to screen tertiary structures of proteins so that potentially useful structures can be found and synthesized.

Neurocomputers and conventional computers will have a continued symbiotic association. At present it is a host-resident relationship, but in the future it might result in shared processing in Database Management Systems (DMS). Patterns of information could be more easily found by neurocomputers than conventional computers. This could be of use in scientific literature databases, medical record systems, and of course numerous government databases. Another example of cooperation between the two types of computers is a true electronic secretary: a neurocomputer which recognizes speech and uses a conventional word processing program to format, spell check and so forth. At an even higher level one can envisage a master operating system which determines which kind of computer is most appropriate for the task at hand.

Chapter 4

The Proposed Approach

4.1 Introduction

This thesis uses neural networks to automatically classify rotational parts into part families. Adaptive Resonance Theory (ART2) was used as a pattern classifier. A simulation experiment was performed on seventy different rotational parts to verify the applicability of neural networks to the part family formation problem. The results were encouraging and showed promise of application in the real world domain.

4.2 Problem Statement

The task is to use neural networks (ART2) to classify parts into part families based on geometrical features and tolerances. The problem can be subdivided into three sub-statements.

1) The system must be able to classify a given set of parts into a set of part families.

2) Whenever a new part enters a system, the system must be able to classify the part as belonging to an existing part family.

3) The classification system must be able to classify novel parts into a new part family.

4.3 Proposed Technique

The proposed technique uses an unsupervised neural network paradigm (ART2) to classify rotational parts into part families. Information on geometric features and tolerances is used as a classification criterion. The proposed technique was tested with a hypothetical set of 70 parts. Refer to Appendix A for sketches of these components. The broad objective of this thesis was to prove the applicability of neural networks in general, and ART2 in particular to the classification problem. The classification procedure will be explained later in this chapter.

4.4 Introduction to ART2

The Adaptive Resonance Theory (ART2) model is used for the proposed application to automatically classify parts into part families in a Group Technology manufacturing environment. Adaptive Resonance Theory is a mathematical model of an unsupervised neural network architecture. The network architecture is unique among existing paradigms. The architecture detects and remembers statistically predictive configurations of featural elements which are derived from the input part patterns. The main advantage in using ART is the scope of learning any kind of configuration of part families or other patterns. In ART2 the learned part patterns undergo self-organization and

self-stabilization as training progresses.

Self-organization is said to have occurred when the network classifies the input part patterns into different part families automatically without external help. This process of self-organization avoids the use of a teacher to decide the size of the part family.

Self-stabilization is said to have occurred when the learned history is not washed away by the more recent learning. This is true even if the inputs are presented in any arbitrary order and in any arbitrary complexity. The search strategies are dynamically modified and updated as new part patterns are learned. The learned part patterns result in the formation of critical feature patterns for each individual part family in the network. The critical feature patterns are concepts which the network develops and cannot be physically accessed by the user. The methodology and working of ART2 are given in Appendix F.

4.5 Justification for using the ART2 Paradigm

Several neural network models offer themselves to the classification task. There were broadly two training strategies which were investigated during the course of this research. They are the supervised and unsupervised training strategies. In the supervised training mode three types of network architectures were examined. They were the back-propagation, bidirectional associative memory and

counter propagation. It was found that these network architectures do not permit any arbitrary pattern as an input. Moreover the very purpose of automatic classification is defeated if an expected output is provided. The creation of a representative part was a determining factor in deciding the number of groups in which the network would classify the parts. Several unsupervised training paradigms and network architectures were explored for the proposed application. The paradigms investigated were the Kohonen Feature Map, the Adaptive Resonance Theory - 1 and Adaptive Resonance Theory - 2. The Kohonen Feature Map was said to provide good results in pattern recognition literature [47,48,49,50,51]. The proposed classification when performed with a Kohonen feature resulted in a very large single group of parts. It was found on further study that the drawback of the Kohonen Feature Map is the bias or tendency of the network to classify all parts under the first group formed. This results in a very small number of groups to be formed for a very large number of parts. The Adaptive Resonance Theory - 1 was experimented with next. ART1 accepts strictly binary inputs. The experiment did not provide the right results. It was later found that ART1 is not yet modeled perfectly and therefore gives erratic results. The ART2 paradigm gave good results and was used as a classification tool. ART2 accepts gray scale inputs ranging from 0.0 to 1.0. The next

section will enumerate the salient features of ART2.

4.6 Properties of ART2

There are four important properties of ART2. Each property is necessary in understanding the working of the proposed application. The four subsections below will explain them.

4.6.1 Critical feature patterns

Part pattern context enters the definition so that input features which are treated as irrelevant noise when they are embedded in a given input pattern may be treated as informative signals when they are embedded in a different input pattern. The systems learning history must also enter the classification criteria. This is necessary since, portions of an input pattern which are treated as noise when they perturb the system at one stage of its self-organization may be treated as signals when they perturb the system at a different stage in its self-organization. The proposed system automatically self-scales its computational units to embody context and learning dependent definitions of signal and noise. The critical feature patterns of the parts are the computational units of the code learning process. The term critical features indicate that not all features are treated as signals by the system. The learned units are patterns of critical features. This is because

the perceptual context in which the features are embedded influences which features will be treated as signals and which will be processed as noise. Thus a part feature may be a critical feature in one pattern and an irrelevant noise element in a different pattern.

4.6.2 Self-adjusted Memory Search

In the proposed application the knowledge structure evolves due to learning. A search algorithm is needed to classify the parts into an existing or a new part family. It is impossible for a prewired search algorithm to maintain its efficiency as the knowledge structure evolves due to learning. A search order that is optimal in one knowledge domain may become extremely inefficient as the knowledge domain becomes more complex due to learning. The ART2 system is capable of parallel memory search that adaptively maintains its search order to maintain efficiency as part learning progresses. The self-adjusted search mechanism is part of the network design whereby the learning process self-stabilizes by engaging the orienting mechanism. None of these mechanisms is akin to the rules of the serial computer program. Once the ART2 architecture is developed, a little randomness in the initial values of its memory traces, rather than a carefully wired search tree, enables the search to continue until the recognition code self-stabilizes.

4.6.3 Direct Access to Learned Codes

One of the most important features of the proposed approach is the rapidity with which familiar part patterns can be recognized. The existence of many learned part patterns for alternative experiences does not necessarily interfere with rapid recognition of familiar part patterns. This type of rapid recognition is difficult to implement using models wherein trees or other serial algorithms must be searched for longer and longer periods as learned recognition codes become larger and larger. In the proposed approach, as the recognition codes of the part patterns become globally self-consistent and predictively accurate, the search mechanism is automatically disengaged. Familiar input part patterns directly access their learned code or part family no matter how large and complex the learned codes may become. The critical feature patterns act as a prototype for the entire part family. Unfamiliar part patterns which cannot stably access a learned category engage the self-adjusting search process in order to discover a new network substrate for a new part family. After this new part pattern is learned, the search process is directly disengaged and direct access ensues.

4.6.4 Attentional vigilance

As mentioned earlier the ART2 system self-organizes its recognition codes. The environment can also modulate the

learning process and thereby carry out a teaching role. This teaching role allows a system with a fixed set of representative parts or critical features to function successfully in an environment which imposes variable performance demands. In our case the environment may demand either a coarse or a fine discrimination to be made among the same set of part patterns. This environment is the cost analysis which would dictate the number of groups that could be formed. The system becomes more vigilant and forms finer categories as the value of the vigilance parameter is increased and vice versa. The ability of a vigilance parameter to alter the course of a pattern recognition illustrates a theme that is common to a variety of neural processes.

4.7 Advantages of ART2

The heart of the proposed approach was the Adaptive Resonance Theory Model (ART2) developed by Stephen Grossberg (1988). This network architecture self-stabilizes and self-organizes in response to a stream of input patterns. Self-organization is said to have occurred when the network classifies the input part patterns automatically without the help of a teacher. This unique feature of self-organization avoids the need to employ an omniscient teacher to decide the size of the part family. Self-stabilization occurs when prior learning is not washed away by more recent learning.

In the case of other architectures, learning becomes unstable due to simple changes in the input environment. Changes in the probability of inputs or deterministic sequencing of inputs can also affect the network.

The learning system in an ART2 architecture is designed to remain plastic in response to significant new input part patterns. It also simultaneously remains stable in response to previously learned part patterns. The characteristic of self-organization is central to the classification process.

The ART2 system generates recognition codes adaptively and without a teacher, in response to a series of input part patterns. As learning proceeds, the interaction between the inputs and the systems generate new steady states, or equilibrium points. The steady states are formed as the system discovers and learns the critical features of part patterns. These critical feature patterns of parts or prototypes represent invariant features in the set of all experienced inputs. The ART2 system is sensitive to novelty. It is capable of distinguishing between familiar and unfamiliar part patterns without a teacher.

Multiple interacting memory systems are needed to monitor and react adaptively to the novelty of events without an external teacher. Within ART2, interactions between two functionally complementary subsystems are used to process familiar and unfamiliar events. Familiar part patterns are processed within an attentional subsystem. The

attentional subsystem continues to develop more precise internal representation of responses to familiar part patterns. As described above the attentional subsystem is incapable of simultaneously maintaining stable representation of familiar categories and also creation of new part families for unfamiliar part patterns.

An isolated subsystem can become too rigid for creating new families for unfamiliar part patterns or conversely it might also become unstable and would ceaselessly recode the categories for familiar part patterns. The second subsystem is an orienting subsystem that resets the attentional subsystem when an unfamiliar part pattern is presented to the system. Interaction between the attentional and orienting subsystem helps to express whether a novel part pattern is familiar and well represented by an existing recognition code, or is unfamiliar and in need of a new recognition code.

4.8 Classification Procedure

The procedure consists of five steps. To begin, the parts are coded into a binary matrix for input to the network. The network parameters are then tuned and set. This involves observing the specific effect of each of the network parameters. Once the network parameters have been set, simulation runs are made. The results of the simulation are obtained in a separate output file. The

One lure of neural networks research is that the field is still in its infancy. Time will tell whether the enthusiasm is justified. As the network and their learning rules become more sophisticated, other fields will exploit the technology to improve information processing.

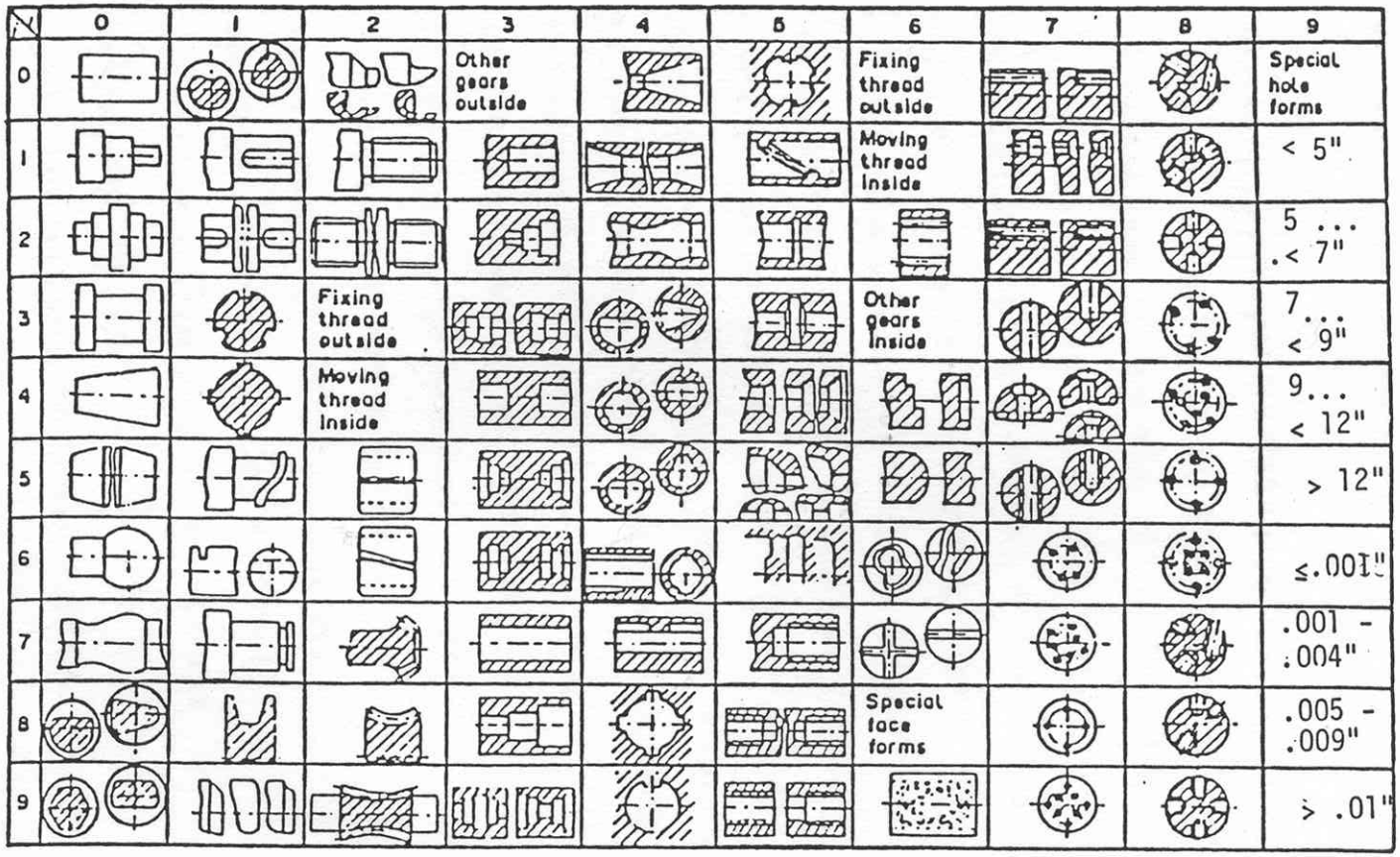


Figure 5. The Reference Matrix [37]

results are evaluated and conclusions presented to support the validity of the proposed approach. Each one of these steps is a detailed procedure which requires explanation.

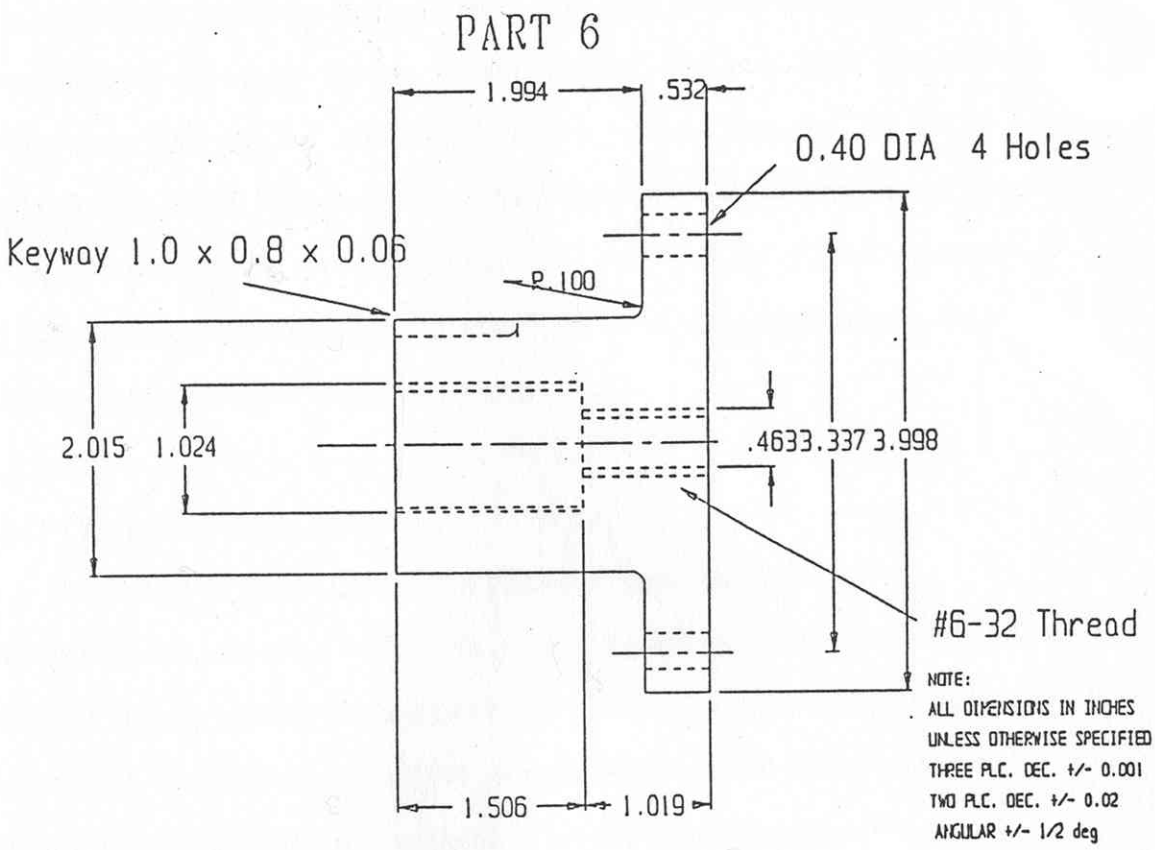
4.8.1 Coding of The Binary Matrix

The binary matrix is the input to the network. The matrix is coded manually by the user. The idea of creating the binary matrix was taken from Grum and Pelenik [37]. They introduced the concept of part representation by a unique matrix, where each cell represents a specific feature. The binary matrix was coded from a reference matrix of geometric primitives (refer to Figure 5). Every cell in the reference matrix represents a particular geometrical feature like a hole, taper or a threaded feature. The reference matrix contains information on the following geometrical primitives.

1) Basic outside geometrical features
2) Auxiliary outside geometrical features
3) Basic inside geometrical features
4) Auxiliary inside geometrical features
5) Geometrical features on flat surfaces
6) Auxiliary bores with or without threads
7) Dimension and accuracy

In the proposed approach the part information is coded into the binary matrix by using the reference matrix developed by Grum and Pelenik [37] as explained above. The coding

procedure is a manual process. The user looks at the part blue print or in this case the component sketch. The reference matrix is next observed. A cell entry of 1 is made in the binary matrix wherever a feature on the part is indicated by a specific cell in the reference matrix. For example consider part 6 (see Figure 6). The part has a step. A step feature is indicated in the reference matrix by cell C_{10}. Thus the binary matrix has cell entry of 1 in cell C_{10}. A chamfer is indicated by cell C_{91} in the reference matrix. The binary input part pattern has cell entry 1 in cell C_{91}. The specific type of step is represented by cell C_{02}. Thus the binary matrix has cell entry 1 in cell C_{02}. A step hole is given by cell C_{83} in the reference matrix. Thus the binary matrix has a cell entry 1 in cell C_{83}. Internal threading is indicated by cell C_{95} in the reference matrix. Thus the binary matrix has a cell entry 1 in cell C_{95}. The part has circumferential holes and is indicated in the reference matrix by cell C_{87}. Thus the binary matrix has a cell entry of 1. The part is less than 5" in length. This is indicated in the reference matrix by cell C_{19} in the reference as well as the binary matrix. The specific type of internal step is indicated by cell C_{55} in the reference as well as the binary matrix. The tolerance on the part is less than +- 0.001". This is indicated by the cell C_{69} in the reference and the binary matrix. Similarly the remaining part features are coded and the




```
/* INPUT PART PATTERN: 6 */
0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
1.00,1.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,1.00,0.00,0.00,0.00,1.00,0.00,0.00
0.00,1.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00
```

Figure 6. Completed Binary Matrix

binary matrix is completely coded.

Seventy binary matrices, one for each component, were created and stored in an input ASCII file. The matrix has to be created in an ASCII file in order for it to be compatible with ANSim [8]. The input sample file of six parts is as shown in Appendix C. Once the input binary matrices are created the next step is to construct and specify the network parameters.

4.8.2 Network definition

Network definition specifies and define the architecture of the network. This definition is stored and defined in a network definition file as shown in Appendix B. There are two issues to be addressed under this section. The first is that the network architecture has to be specified. The second is that the parameters which control the network function have to be tuned and fixed. An example problem will be used to demonstrate this step.

The number of processing elements in node 1 has to be of the same size as that of the input binary matrix. In the example problem the input node is a 10 x 10 array of processing elements, i.e. 100 processing elements. The number of processing elements in the output layer represents the number of part families formed by the network. In the example problem the output node is a 10 x 7 matrix, i.e. 70 processing elements.

Many parameters have to be tuned in order to train the network successfully. ART2 is sensitive to a combination of parameters such that degraded performance or instabilities can arise. Therefore tuning ART2 by progressive adjustment of these parameters controls the functioning of the network.

1) Vigilance Parameter

This parameter gives the degree of specificity in distinguishing part patterns. Higher vigilance will discriminate part patterns more specifically than with lower vigilance values. It can take values ranging from 0.1 to 0.99. It was concluded from experiments that a vigilance value below 0.90 and above 0.96 had no effect on the classification. This means that the number of classes formed remained a constant beyond the defined limits. The example problem was tested with five values of vigilance to observe its effect on the grouping. The values were 0.92, 0.93, 0.94, 0.95, 0.96.

2) The number of Learning Cycles

This gives the number of cycles during which the input part pattern will persist. This parameter can be set to any value greater than 1. It controls the time taken to train the network. In the example problem this parameter was fixed empirically by observing when a part is completely learned by the network. This is determined when the processing element in node 2 representing the part family exceeds an activation value of 1. The parameter was fixed

at 100 for the example problem.

3) Input Adjustment Gain

The Short Term Memory (STM) part pattern at the output of node 1 is used to adjust the input part pattern "BM" as it is presented at node 1 of the network. This parameter determines the degree to which the input pattern "BM" is adjusted by the STM pattern at node 1. The value ranges from 0.0 to 1.0. A parameter value of 1 adjusts the input pattern completely with the STM pattern. A value of 0 ignores the STM value. In the example problem a middle value of 0.5 was used. This value was decided as a result of several simulation runs.

4) STM Adjustment Gain

The STM pattern also is updated as the network progressively learns the part patterns. There are two factors that affect the STM updating or learning. They are the input pattern "BM" and the Long Term Memory (LTM) pattern. This parameter determines the degree to which the STM pattern should be effected by the LTM pattern. The LTM pattern is essentially the prediction of what the input pattern should be. The STM Adjustment gain range is 0.0 to 1.0. The value of 1 equally weighs the input and LTM pattern values when updating the STM pattern. A value of 1 equally weighs the input and LTM values when updating the STM pattern. A value of 0 ignores or gives a zero weight to the updating done by the LTM value. A large value makes the

network insensitive to variation in the input part pattern. This results in reducing the discriminating capability of the network. A variety of STM values were investigated and a value of 0.35 was fixed for the example problem.

5) Predicted Gain

During network training and learning the top down pattern "PVn" is compared to the input pattern "BM". The top down or recalled vector is represented by the LTM. The difference between the input pattern "BM" and learned pattern "PVn" is applied to update the resonating vector between node 1 and node 2. A value of 1 applies the total difference to the update. A setting of 0 makes the resonating vector turn off LTM recall and reset mechanism. The smaller the value of this gain parameter the slower the rate of learning of the input pattern "BM". This was concluded by conducting experiments with various values of this parameter. The value of predicted gain was fixed at 0.5.

6) LTM Adjustment Gain

This parameter determines the learning rate of the Long Term Memory (LTM) traces of the network. The learning rate for the LTM traces is typically small as compared to that of Short Term Memory (STM) traces. This parameter can take values ranging from 0.0 to 1.0. The value 0 stops the LTM learning process and a value 1.0 changes LTM traces at a rapid pace resulting in a faster rate of learning. A value

of 0.90 was first experimented with and then steadily decreased till the network gained stability. The rapid rate of change of the learning can destabilize the network by preventing a select set of LTM traces to learn the pattern. A low value of this parameter slows down the learning process to a point where the time to train the network is on the order of several days. This means that there is a specific value for this parameter which determines the optimum learning rate for the LTM. An experiment was conducted at a value of 0.20 and the network stabilized after three days. This parameter was fixed at 0.4.

7) Excitation Bias

This biasing parameter takes care of the noise component in the input vector "BM". It is the level of sensitivity to noise in the STM generated by the input. It is used for non-specific normalization of the STM vectors. It ranges from 0.0 to 1.0. The value investigated were 0.1, 0.50, 0.70, 0.001, 0.002, 0.004. The value of 0.004 was selected since the noise was very low in the data.

8) Excitation Threshold

The activation function of a processing element can be a sigmoid or any other function depending on the application. The function employed for this application was the sigmoid. The sigmoid function had an excitation threshold above which the processing elements could fire. The sigmoid may take on values ranging from 0.0001 to

0.9999. It was found that if the threshold level was low i.e. 0.01 or 0.08, the network did not learn meaningful patterns. This is because the network tried to consider every input from the part pattern that it experienced. A value of 0.5 was selected for the example problem.

4.8.3 Run Simulation

Simulations were conducted on an IBM PC Compatible, 80386, 25mhz machine. The software used was ANSim (Artificial Neural Network Simulation) from SAIC (Scientific Applications International Corporation). The ASCII input file of parts matrices was converted to ANSim compatible format. This file was then presented to the ART2 network with all the parameters discussed above. Each part was presented to the network 100 times. The network was trained at different vigilance values; 0.92, 0.93, 0.94, 0.95, 0.96. The training time for each simulation was approximately 20 hours. The simulations resulted in outputs from the networks, which gave the various part families.

4.8.4 Simulation Outputs

The output is printed to an ANSim file by the process file option once the network was trained (refer Appendix D). This option presents all the trained parts for 1-5 iterations to the already trained network. The result is an output file containing all the part families or groupings.

The output file is still in an unreadable ANSim format and has to be converted to an ASCII file to analyze parts grouping data. This procedure is carried out by using the standard ANSim Convert utility. The ASCII output file is scrutinized for the resulting group data.

4.8.5. Interpreting simulation outputs

The processing elements in the output node (Node 2) are numbered 1 to 100 starting from the left extreme and continuing down until the right bottom processing element. For example, the cell C_{18} will be indicated by the number 8 and the processing element C_{34} will be denoted by the number 24. All the processing elements in the rest of the work will be referred by this unique cell or part family identifier.

Every input part pattern file generates an output file as a result of executing the process file option. The output matrix indicates the part family for the specific part. The first matrix titled "Input Vector n" (n = 1,2...70) is the input matrix and represents the original inputs to the system. The matrix following this input matrix is titled "Output Vector n" (n = 1,2...70). It is seen that only one processing element in the output vector is greater than 0, the rest are all 0. This unique positive processing element is the part family or group number. This matrix is of interest to the user in analyzing the

groupings. It was observed that the magnitude of the activation does not indicate a specific trend within or outside the groups. This means that the magnitude did not effect the way the network classified parts into groups. Parts which activate the same part family identifier are grouped under same part family. Consider for example the output matrix of parts 58, 59 and 60 at a vigilance of 0.94 (refer to Figure 7 for the output matrix of parts). The part family identifier for parts 59 and 60 is cell C_{24}. This means that both these parts belonged to the same part family C_{24}. Part 58 has a part family identifier C_{30}.

4.9 Comparison of Proposed Approach with Grum and Pelenik [37] Approach

There are many advantages of the proposed technique over the technique proposed by Grum and Pelenik (see Figure 8). The Grum and Pelenik Approach will be explained first and then compared and contrasted with the proposed technique. The first step is the representation of the part in a form suitable for input to the system. The geometric matrix from which the parts are coded contains information as discussed in the section 4.4.1. In correspondence with this geometric matrix Grum and Pelenik described featural elements on the parts with yes (black) or no (white). As a code example refer to Figure 9. Classification in a part family is carried out on the basis of pattern recognition.

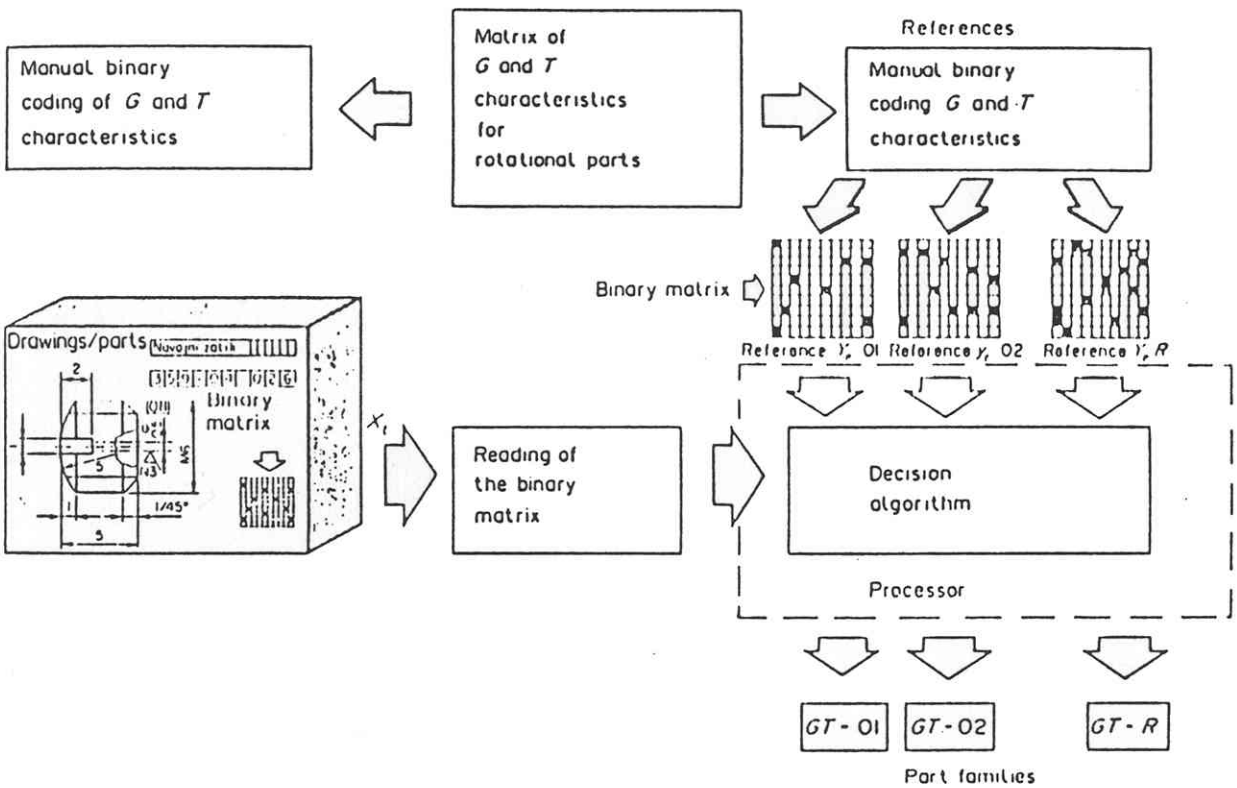
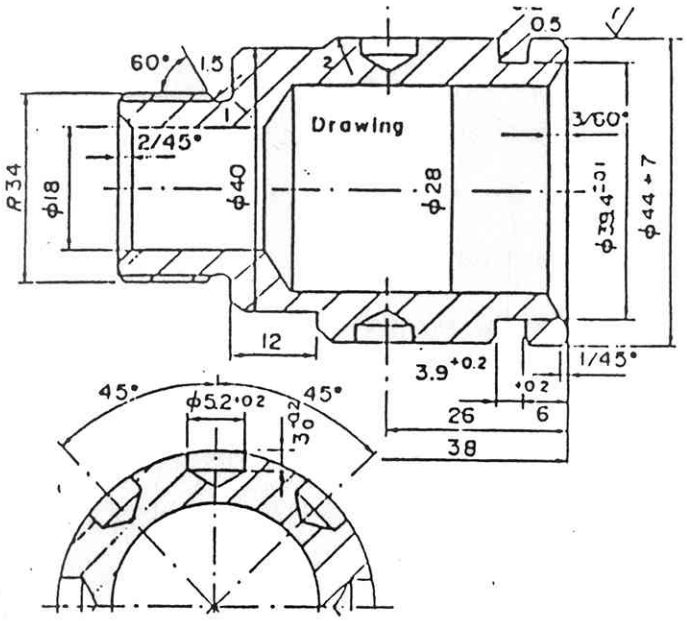


Figure 8. Grum and Pelenik Technique [37]



1	11	21	31	41	51	61	71	81	91
2	12	22	32	42	52	62	72	82	92
3	13	23	33	43	53	63	73	83	93
4	14	24	34	44	54	64	74	84	94
5	15	25	35	45	55	65	75	85	95
6	16	26	36	46	56	66	76	86	96
7	17	27	37	47	57	67	77	87	97
8	18	28	38	48	58	68	78	88	98
9	19	29	39	49	59	69	79	89	99
10	20	30	40	50	60	70	80	90	100

Figure 9. Grum and Pelenik coded example [37]

A vision system is used as a pattern recognition tool. For this purpose, a matrix of geometric characteristics of all the representatives of the part families are created and named the reference or the reference matrices. These are compared with the given parts matrix that are to be classified in a certain part family. The procedure is conducted on the basis of a potential function. This permits finding out the relative degree of similarity among the parts that are to be classified into part families.

On comparing and contrasting Grum and Pelenik [37] technique with the proposed technique, we see that:

1) The number of groups formed is not predetermined by the user in the proposed technique. Grum and Pelenik used a fixed number of representative part families which already existed or were created by an omniscient user.

2) A potential function is the heart of Grum's procedure. The potential function is the decision making aid to decide whether a part belongs to a particular part family. The proposed approach does not use the concept of a potential function. The potential function is a time consuming procedure. A canned formulae for a potential function may not necessarily satisfy all the classification criteria. The potential function is the decision making aid which decides whether a part belongs to a particular part family.

3) A decision algorithm is not necessary to classify parts by the proposed technique. A decision algorithm is the

decision aid in Grum's work. Moreover the decision algorithm has to be written for every new application domain.

4) The coarseness/fineness of the groups can be controlled by changing the vigilance parameter in the proposed technique. Grum's technique cannot accommodate changes in the size of the groups. This is because a new set of reference components will have to be defined which is a time consuming task.

5) The search time to determine the group or part family is trivial due to the parallel search scheme in neural networks. The search time in Grum's technique increases as the parts database becomes more complex. This is true of any prewired search algorithm.

6) The proposed technique can accommodate future inclusions of a set of novel components without modifying the system. In Grum's technique novel parts cannot be handled. This is because the system is rigid and any inclusion would mean reprogramming the system to accommodate the novel component.

7) The proposed system possesses learning capability and reacts as a function of experience. Grum's technique does not learn and hence the system is not updated automatically and is not intelligent.

4.10 Comparison of proposed approach with Ham [44] Approach

Ham et al. used a two layered backpropagation neural

net model in neural networks to classify parts into representative part families. The neural network is trained with all representative parts in the initially available n part families P1, P2, P3,...Pn (these representative parts are called exemplars). Each part family has a set of one or more representative parts or exemplars. When the system encounters a new part it responds by identifying the exemplar that is the closest match to the new part. The next step is to check to see if the two parts are similar to the desired extent. Ham et al. used Tanimoto's Coefficient to determine the degree of similarity. Thus the classification is then done for all other new components presented to the system. If the new part does not belong to any existing part family a new part family is formed. The input to the neural network was from a vision system. The part was represented by a vector of pixels which represents a two-dimensional view of the part's diagram.

Contrasting and comparing Ham's work with the proposed technique, we see that:

1) The proposed technique does not employ the concept of a potential function. Ham's technique uses a potential function to decide the size of the groups. This means that the neural network is not used as an intelligent decision tool, but merely as a heuristic rule-based system. The purpose of automatic classification is defeated when the potential function is developed and employed.

2) The number of parts that can be learned by the network is not limited by the network architecture in the proposed approach. In Ham's approach the network architecture needs to be modified as more parts are incorporated into the system.

3) In the proposed approach the parts are represented by a binary matrix of geometrical primitives. Ham used a two-dimensional view of the part as input to the vision system. A two-dimensional view does not represent all the features on the parts. Moreover, only symmetrical rotational parts can be classified. The proposed approach does not have these restrictions.

4) The proposed system uses an unsupervised training algorithm (ART2) to classify parts automatically. The system therefore truly models automatic classification. The use of a backpropagation (supervised learning algorithm) neural network closely models a rule-based system rather than a automatic classification system.

Chapter 5

Results Conclusions and Recommendations

5.1 Introduction

The results, conclusions and recommendations were an interesting aspect of this research. The results were encouraging. There are many important conclusions drawn as a result of this research effort. Many improvements and refinements can be implemented on the proposed system.

5.2 Results

The classification simulation was conducted for five values of the vigilance parameter. The part families are identified by a unique part family or Group Number. The cells are numbered 1 to 100 starting at the top left corner of the output matrix. Part families with a singular part in the part family are not shown in the analysis. This assumption was based on the fact that a part family should consist of at least two parts to be recognized as a part family. Moreover the analysis of part families with singular parts showed that they had some features which were not common to other part families, i.e. were unique and did not belong to any other part family. The analysis of all the outputs obtained are given in Appendix D.

Simulation runs were conducted for five values of the vigilance parameter. The vigilance values, 0.92, 0.93,

0.94, 0.95, 0.96 were used in the experiments. The concept of a vigilance parameter is contrary to the concept of a potential function where the higher the value of the potential function the more similar are the parts. The results are tabulated as Shown in Appendix E.

The grouping analysis was done in two stages. In the first stage, parts were tabulated into their respective groups or part families. This analysis indicates that as the vigilance parameter increased from 0.92 to 0.96 the number of groups or part families increased. In the second stage the analysis was conducted for finding the basis on which parts were classified into a specific part family. There were certain parts features which were assumed to be the defining criteria for forming groups. Every grouping is analyzed for the absence and presence of these features. It is concluded that the absence or presence of a particular feature on parts within a particular part family were instrumental in defining the classification procedure. It is to be noted that for every value of the vigilance parameter the groups need not have the same part family identifier (Group Number). This is because the LTM weights are randomized at the beginning of every simulation. The results will be explained for one value of vigilance. Consider the part families formed at the vigilance value of 0.94. Please refer to Table 2. There are eight groups formed, 24, 26, 29, 30, 39, 51, 66. This is shown in the

```
/* Output vector 58 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.05,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 59 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.20, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 60 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.13, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

Figure 7. Output Matrix

Table 2. Grouping Anaysis: Vigilance Parameter:- 0.94

Group No.	PARTS
24	4,6,9,10,11,12,13,14,24,25,26,39,41,42,44,45,46, 48,50,52,53,54,59,60,61,65,66,68,70
26	7, 18,21,27,67
29	17,28
30	1,2,8,15,22,23,32,33,34,35,36,40,43,49,57,58,69
39	19,20
51	16,29,30,37,38,62,63,64
66	31,51,56

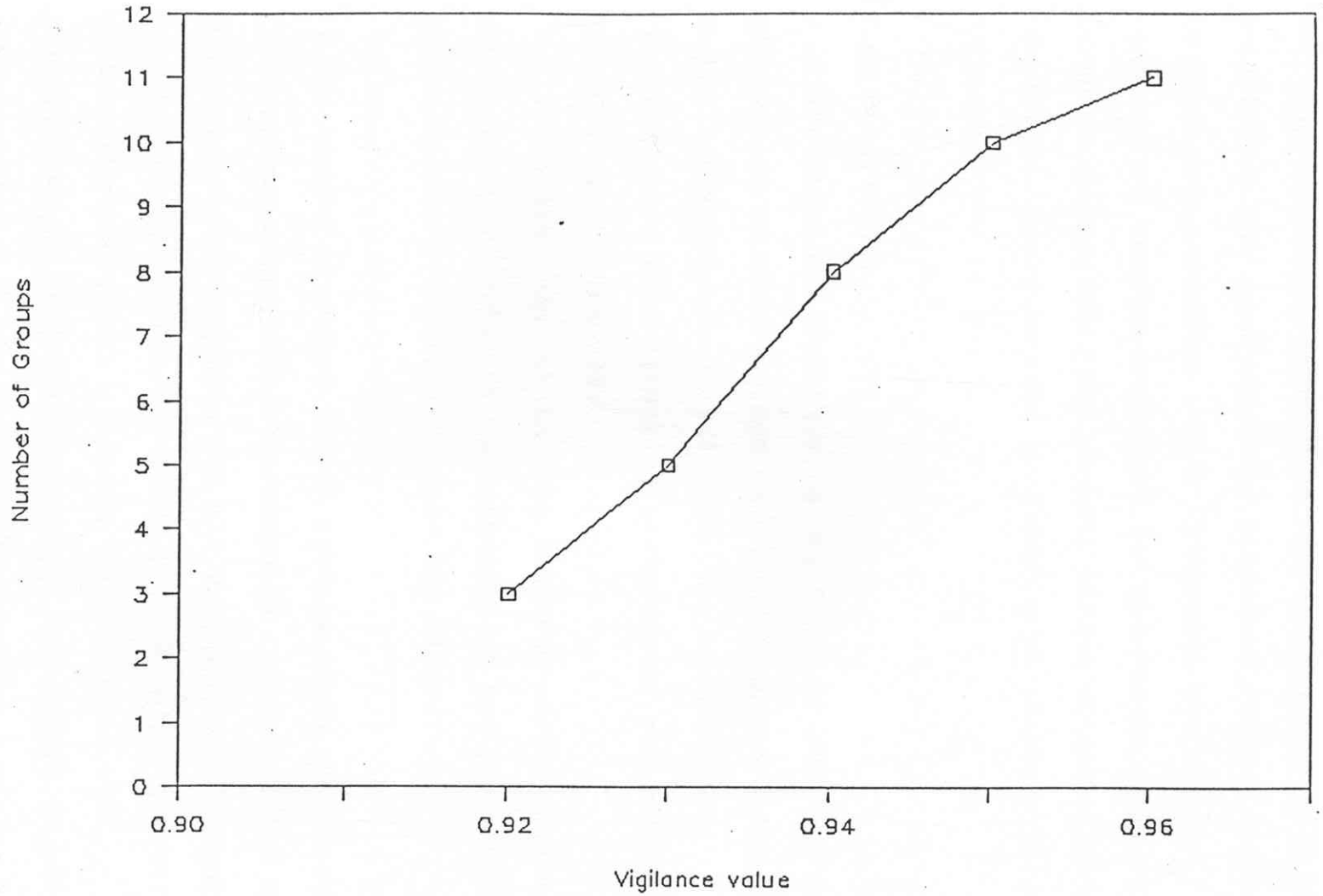
Table 3. Grouping Analysis: Vigilance Parameter:- 0.94

Group No.	24	26	29	30	39	51	66
Geometric Features							
Straight Outside	P	O	P	P	P	O	P
Taper Outside	O	O	A	P	A	O	O
Groove Outside	O	O	A	O	A	O	A
Pulley	O	A	A	O	A	O	A
Circumferential Holes	O	A	A	A	A	A	A
Keyway Outside	O	O	A	O	A	O	O
Threading Outside	O	O	A	O	A	A	O
Threading Inside	O	A	O	O	O	A	O
Cross hole	A	O	A	O	O	O	A
Straight hole	O	O	P	O	P	O	P
Taper hole	O	A	A	O	A	A	A
Blind hole	A	O	A	O	A	A	P
Step Hole	O	A	O	O	A	A	A
Hole	O	O	P	O	P	O	P

Key:-
A - Denotes absence of a feature.
P - Denotes presence of a feature.
O - Denotes either the presence or absence of a feature

first column titled Group No. The number of part families formed are 7. The number of parts in each part family are 30, 6, 3, 17, 2, 9 and 3 respectively. This was the first stage of analysis. The second stage is to analyze the groupings obtained earlier for the absence and presence of geometrical features. Please refer to Table 3 for this analysis. Consider Group No. 24 in the second column. A P in a cell indicates the presence of a geometrical feature (indicated by the row) in all parts within a group. An A in a cell indicates the absence of a geometrical feature on all parts within a group. An O indicates either the presence or absence of a feature within a group. Therefore Group No. 29 has the following features present on all parts: Straight Outside Edge, Straight Hole, Hole. It has following features absent from each part in Group No. 29 Taper Outside, Groove Outside, Pulley, Circumferential Hole, Keyway outside, Threading outside, Cross Hole, Taper Hole, Blind Hole. Group No. 29 also has the following features either present or absent: Threading inside, Step Hole. Similarly all the other part families are analyzed and evaluated for reasons of their association to a particular part family. Is is observed that no two part families have the same features absent or present. This means that the patterns of "A" and "P" are different for all part families. The conclusions of these analysis are:

1) As the vigilance parameter increased from 0.92 to 0.96


12
11
10
9
8
7
6
5
4
3
2
1
0
Number of Groups
0.90
0.92
0.94
0.96
Vigilance value

29, 30, 37, 38, 62, 63 and 64 belonged to one part family. The test was performed to test the validity of the groupings. The experiment was performed at only one value of vigilance parameter. This is because the network architecture remains unchanged for all values of vigilance parameter. Therefore the classification of novel parts into new groups should be true at every level of the vigilance parameter.

5.3 Conclusions

The proposed technique is the first of its kind in the application of ART2 (unsupervised learning) to a Group Technology application. It opens new avenues for classification procedures in manufacturing as well as non-manufacturing procedures.

There are some outstanding features of the proposed method that overcome the deficiencies of the present method. These salient features are explained below.

1) The number of groups is not predetermined by the user in the proposed approach. This actually represents and models automatic classification. In all existing methodologies the user has to decide the number of part families the parts will be grouped under. This is not a natural and justifiable way of grouping parts if the groups do not already exist. If the groups already exist then the user

has to determine the validity of the grouping. If the user has to decide the number of groups beforehand, the user will have to thoroughly study the parts database before making a decision on the number of part families to form. The proposed method allows the user to dynamically alter the number of part families by varying the vigilance parameter.

2) The classification in ART2 is based on the formation of a composite component. The composite component or critical feature pattern, is formed automatically as the system learns and evolves. The formation and the number of composite components is not determined by the user as is the existing techniques. The development of a composite component by the user casts doubt to whether it truly represents all the parts in the database.

3) Extensive and complicated programming has to be done to accomplish classification by the existing methods. The development of algorithms for every new application is a time consuming process. In the proposed method only training, tuning, and setting of learning parameters is done by the user to classify data successfully. For a very large database of parts the program development and testing time would be enormous.

4) The knowledge acquisition task in a classification performed by a conventional artificial intelligence

technique is a time consuming process. In the proposed approach there is no stage of knowledge acquisition, therefore there is no time for this process. The knowledge acquired by the system is the result of all the learning experienced by the system. The proposed method is application independent. This means that the same network architecture can be applied to any other application. The number of feature detectors have to be changed depending upon the size of the database. The learning and vigilance parameters have to be reset.

5) The ART2 neural network architecture speeds up processing time as the network is inherently parallel in nature. Moreover the simultaneous interaction of two memory mechanisms speed up the process even more.

6) The use of ART2 assures that there is a trivial amount of time for search (3 to 5 seconds). This is because parallel interactions are modeled on a sequential computing machine. On a parallel machine the search time would be nearly zero. In all conventional artificial intelligence applications to the problem of part family formation the search time is proportional to the number of components in the database. The larger the database the longer it will take to search the database. In the proposed application the search time remains constant irrespective of the number of parts in the

database. In the present time of intense global competition a product is either launched before the competitor or it faces doom. The proposed application will ensure that user has an edge in rapidly producing his/her product.

The proposed approach has some weaknessess.

1. It was observed during the analysis that some parts within a group were not intuitively obvious. As an example parts 58, 59 and 60 were classified into two groups one with part 58 and other with part 59 and 60. This classification was true for every value of the vigilance parameter. Intuitively part 58 should belong to the same group. This suggests that the reference matrix should be altered since the grouping of parts is completely outside the user control.

2. It was observed that parts which were grouped together at a higher value of vigilance were not necessarily grouped together at a lower value of vigilance. Thus the results prove to be counter intuitive.

3. Prelimnary results were encouraging but not entirely successful.

The major conclusions of this research are:

1) The ART2 is not entirely successful for the proposed approach as is observed from the above mentioned results.

2) The order in which the parts are input to the system has no effect on the classification procedure.

3) The number of part families formed changes with a change

in the vigilance. As the vigilance increases the number of part families formed also increase. The lowering of vigilance decreases the number of part families formed.

5.4. Future Research Directions

The scope for future research in this field is immense. There are various avenues that could be explored. A few important ones are listed below.

1) The proposed technique is restricted to rotational components. Further development can be done on this application to prismatic components. The basic input form will remain the same but the binary matrix for prismatic components will have to be used instead of the binary matrix for rotational components.

2) The present classification scheme takes into consideration only the geometric feature of parts. This is not the best way to perform classification. Classification based on routing information was also attempted, and showed good results.

3) An integration of the two classification schemes i.e. based on both geometric features and routing information is proposed. Future research should be undertaken to investigate other forms of manufacturing information that can be best used to best classify parts.

4) The binary matrix can be changed to a gray scale matrix. Weights can be allocated to every cell so that the

importance of certain parts information is emphasized. This weight allocation will vary from organization to organization and can be based on heuristic judgement.

5) The input at present is a manually coded binary matrix. An attempt should be made to read and decipher parts information directly from a CAD database or a picture of a part using a vision system.

6) The development of a mathematical model to select the vigilance parameter is also a challenging area to investigate. It can be expected that the vigilance parameter can be determined as a function of cost.

The conclusion of this research is that neural networks are indeed an appropriate tool to classify rotational parts automatically into part families.

References

[1] Adam, Blum., "Bidirectional Associative Memory Systems in C++: Recent innovation makes associative memory practical for real world problems", Dr. Dobb's journal, April 1990.

[2] Amari, S.A., "A Mathematical approach to neural systems", In J. Metzler (Ed.). Systems Neuroscience pp 67- 117, New York Academia Press, 1977.

[3] Amari, S.A., "Neural Theory of Association and Concept formation", Biological Cybernetics, 26, pp 175-185, 1977.

[4] Anderson, J.A., "A theory for the recognition of items from short memorized lists", Psychological Review, 80, 417-428, 1977.

[5] Anderson, J.A., "Neural Models with cognitive implications", In D. LaBerge And S.J Samuels (Eds.), Basic properties in reading perception and comprehension, pp 213-236, Hillsdale, NJ: Earlbaum, 1977.

[6] Andrew, J., Czuchry, Jr., "A Network Instantiation environment: Dynamically creating neural nets lets you concentrate on network response characteristics", Dr. Dobb's Journal, April 1990.

[7] Anderson, J.R., The architecture of cognition, Cambridge, MA: Harvard University Press, 1983.

[8] ANSim: "Artificial Neural Network Simulation Software", Science Applications International Corporation (SAIC), 1989.

[9] Arn, E.A., Group Technology An Integrated Planning and Implementation Concept for small and Medium Batch Production, Springer-Verlag, Berlin Heidelberg New York, 1975.

[10] Auckley, D.H., Hinton, G.E., and Sejnowski, T.J., "A learning algorithm for boltzmann machine", Cognitive Science, 9, pp 147-169, 1985.

[11] Batra, J.L., and Rajgopalan, R., "Composite Component through Graphs and Fuzzy Clusters", Proceedings of the Eighteenth International Machine Tool Design and Research Conference, UK, pp 801-807, 1975.

[12] Bienstock, E.L., Cooper, L.N., and Munro, P.W., "Theory for the development of neuron activity: Orientation Specificity and binocular interaction in visual cortex", Journal of Neuroscience, 2, pp 32-48, 1978.

[13] Brankamp, K., "Objectives, Layout and Possibilities of the Opitz workpiece classification system", Proceedings of the International Seminar on Group Technology, International Center for Advanced Technical and Vocational Training, Turin, pp 127, 1970.

[14] Burbidge, J.L., The Introduction of Group technology, Heinemann, London, 1975.

[15] Carpenter, G.A., Grossberg, S., "Neural Dynamics of Category learning and recognition: Atttention, Memory Consolidation, and Amnesia, Brain, Structure, and Memory", (Davis, J., Newburgh, R. and Wegman, E., eds), AAAS Symposium Series, 1986.

[16] Carpenter, G.A., Grossberg, S., "A Massively parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", Computer Vision, Graphics, and Image Processing, Vol. 37, pp 54-115, 1987.

[17] Carpenter G.A., Grossberg, S., "ART-2: Self-Organization of Stable Category recognition Codes for Analog Input Patterns", Applied Optics, Vol. 26, No. 23, pp 4919-4930, December 1987.

[18] Carrie, A.S., "Numerical Taxonomy applied to group technology and plant layout", International Journal of Production Research, v 11, n 4, pp 399-416, 1973.

[19] Casimir, C. klimasaukas., "Neural Networks and image Processing: Finding edges only a human can see", Dr. Dobb's journal, April 1990.

[20] Caudill, Maureen., "Neural Networks Primer Part 1", AI Expert, December 1987.

[21] Caudill, Maureen., "Neural Networks Primer Part II", AI Expert, February 1988.

[22] Caudill, Maureen., "Neural Networks Primer Part III", AI Expert, June 1988.

[23] Caudill, Maureen., "Neural Networks Primer Part IV", AI Expert, August 1988.

[24] Caudill, Maureen., "Neural Networks Primer Part V", AI Expert. November 1988.

[25] Caudill, Maureen., "Neural Networks Primer part VI", AI Expert, February 1989.

[26] Chandrasekharan, M.P., and Rajgopalan, R., "An ideal seed non-hierarchical clustering algorithm for cellular manufacturing", Interantional Journal of Production Research, v 24, n 2, pp 451-464, 1986.

[27] Chao-Hsien, Chu., Pieching, Pan., "The Use of Clustering Techniques in Manufacturing Cellular Formation", Institute of Industrial Engineers, Industrial Engineering Conference Proceedings, 1988.

[28] De Witte, J., "The use of similarity coeffecient in production flow analysis", Proceedings of the 5th International Conference on Production Research(Amsterdam), August 1979.

[29] El-Essaway, I.G.K., and Torrance, J. "Component flow analysis-an effective approach to production systems design", Production Engineer, pp 165-170, May 1972.

[30] Feldman, J.A., "Connectionist models and their applications", Cognitive Science, 9, pp 1-2, 1985.

[31] Filho, E.V.G., "Computer-Aided Group Technology Part Family Formation Based on pattern Recognition Techniques", Ph.D. Thesis, Department of Industrial and Management Systems Engineering, Pennsylvania State University, 1988.

[32] Fukishima, K., Miyake, S. and Ito, T., "Neocognitron: A neural network Model for a Mechanism of Visual Pattern Recognition., IEEE Transactions on System, Man and Cybernitics, SMC-13, pp 826-834, 1983.

[33] Fukushima, K. "Neocognitron: A self organizing neural network model for a mechanism of pattern recognition unaffected by a shift in position", Biological Cybernetics, 36, pp 193-202, 1980.

[34] Ginsburg, H.P., The development of mathematical thinking, New York: Academic Press, 1983.

[35] Grossberg, S., "Adaptive pattern classification and universal recoding: Part I. Parallel development and coding of neural feature detectors", Biological Cybernetics, 23, pp 121-134, 1976.

[36] Grossberg, S., "How does the brain build a cognitive code?", Psychological Review, 87, pp 1-51, 1980.

[37] Grum, J., Logar, B., Hlebanja, G., and Pelinik, J., "Design of Database for CAD based on Group Technology", Robotics and Computer Integrated Manufacturing, Vol. 4., No. 1/2, pp 49-62, 1988.

[38] Hebb, D.O., The organization of behavior, New York: Wiley, 1949.

[39] Hinton, G.E., Anderson J.A., Parallel Models of Associative Memory, Hillsdale, NJ: Earlbaum, 1981.

[40] Hopfield, J.J., "Neural Networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences, USA, 79, pp 2554-2558, 1982.

[41] Hopfield, J.J., "Neurons with graded response have collective computational properties like those of two state neurons", Proceedings of the National Academy of Sciences, USA, 81, pp 3088-3092, 1984.

[42] Jackson, J.H., On Localization. Selected Writings (Vol 2), New York, Basic Books (1985), (Original work published, 1869.

[43] Jeannette, Lawerence., "Untangling Neural Nets: Why is one model better than the other?", Dr. Dobbs Journal, April 1990.

[44] Kamarathi, V. Sagar., Sounder, R., Kumara, T., Francis T.S.Yu., Inyong, Ham., "Neural Networks and Their Applications in Component Design Data Retreival", IMSE Working Paper, pp 89-135, College of Engineering, Department of Industrial and management Systems Engineering, Pennsylvania State University, April 1990.

[45] King, J.R., "Machine-Component grouping in production flow analysis: an approach using a rank order clustering algorithm", International Journal of Production Research, v 18, n 2, pp 213-232, 1980.

[46] King, J.R., and Nakornchai, V., "Machine-Component group formation in Group Technology: review and extension", International Journal of Production Research, v 20, n 2, pp 117-133, 1982.

[47] Kohonen, T., Self-Organization and Associative Memory, Springer Verlag, New York, 1984.

[48] Kohonen, T., "An adaptive associative memory principle", IEEE Transactions, c23, pp 444-445, 1974.

[49] Kohonen, T., Associative memory: A system theoretical approach, New York: Springer Verlag, 1977.

[50] Kohonen, T., "Clustering, taxonomy, and topological maps of patterns", In M.Lang (Eds), Proceedings of the Sixth International Conference on Pattern Recognition, 1982, pp 114-125, Silver Spring, MD: IEEE Computer Society Press.

[51] Kohonen, T., Self organization and associative memory, Berlin, Springer-Verlag, 1984.

[52] Kusiak, Andrew., and Heragu, S.S., "Group Technology State-of-the-Art", Computers in Industry, v 9, n 2, pp 83-91, Oct 1987.

[53] Kusiak, Andrew., "A generalized group technology concept", International Journal of Production Research, v 25, n 4, p 561-569, 1987.

[54] Lashley, K.S., "In search of the engram", Society of Experimental Biology Symposium no. 4: Psychological mechanisms in animal behavior, pp 478-505, London: Cambridge University Press, 1950.

[55] Lemoine, Y., amd Mutel, B., "Automatic Recognition of Production Cells and Part Families", Advances in CAD/CAM, Proceedings of the 5th International IFIP/IFAC conferance on Programming Research and Operation Logistics in Advanced Manufacturing Technology, PROLAMAT 82, T.M.R. Ellis and O.I. Semenkov (editors), North Holland Publishing Company, IFIP 1983.

[56] Lindsay, P.H., Norman, D.A., Human information processing: An introduction to Psychology, New York: Academic Press, 1972.

[57] Luria, A.R. Higher Cortical functions in Man, New York: basic Books, 1966.

[58] Luria, A.R., The working brain. London: Penguin, 1982.

[59] Marslen-Wilson, W.D., Welsh, A., "Processing interactions and lexical access during word recognition in continuous speech", Cognitive Psychology, 10, pp 29-63, 1978.

[60] McCarthy, J., "Comments. In Mechanization of thought processes", Proceedings of the Symposium held at the National Physical laboratory, November 1958, Vol. I, pp 464-466, London, Her Majesty's Stationary Office.

[62] McAuley, J., "Machine grouping for effecient production", Production Engineer, pp 51-53, 1972.

[63] McClelland, J.L., "On the time relations of mental processes: An estimation of systems of processes in cascade", Psychological Review, 86, pp 287-330, 1979.

[64] McClelland, J.L., "Retrieving general and specific information from stored knowledge of specifics", Proceedings of the Third Annual Meeting of the Cognitive Science Society, 170-172, 1981.

[65] Minsky, Marvin., "Neural nets and the brain-model problem", Unpublished Doctoral Dissertation, Princeton University, 1974.

[66] Minsky, Marvin., And Paret, S., Perceptrons, Cambridge MA", MIT Press 1969.

[67] Morton, J., "Interaction of information in word recognition", Psychological Review, 76, 165-178, 1969.

[68] Rajgopalan, R., "Design of Cellular production System-A graph theoritic approach", International Journal of Production Research, 16, pp 577-580, 1976.

[69] Rajgopalan, R., and Batra, J.L., "Design of cellular production systems: A graph-theoritic approach", International Jounal of Production Research, v 13, n 6, pp 567-579, 1975.

[70] Rosenblatt, F., "Two theorems of statistical separatibility in the perceptron. In the Mechanization of thought process", Proceedings of a symposium held at the National Physical Laboratory, November 1959, Vol I, pp 421-456, London, HM Stationery Office.

[71] Rosenblatt, F. Principles of Neurodynamics, New York, Spartan 1962.

[72] Rumelhart, D.E., "Schemata: The building blocks of cognition", In R. Spiro, B. Bruce, and W. Brewer (Eds). Theoretical Issues in reading comprehension, pp 33-58, Hillsdale, NJ: Earlbaum, 1980.

[73] Rumelhart, David E., McClelland, James., and the PDP research group., "Parallel Distributed Processing Explorations in the Microstructure of Cognition, Vol 1 and 2", Institute for Cognitive Science, University of California, San Diego. The MIT Press, Cambridge, Mass, London, England, 1988.

[74] Sejnowski, T.J., "Skeleton filters in the brain", In G.E Hinton and J.A. Anderson (Eds.), Parallel models of associative memory, pp 49-82, Hillsdal, NJ: Earlbaum, 1981.

[75] Selfridge, O.G., "Pattern recognition in modern computers", Proceedings of the Western joint computer conference, 1955.

[76] Simpson, P.K., "Associative Memory Systems", Proceedings of the International Joint Conference on Neural Networks, January 1990.

[77] Von der Malsberg, C., "Self organizing of orientation sensitive cells in the striate cortex", Kybernitik, 14, pp 85-100, 1973.

[78] Waghodekar, P.H., and Sahu, S., "Machine-component cell formation in group technology: MACE", International Journal of Production Research, v 22, n 6, pp 37-948, 1984.

[79] William, F Hyde., "Improving Productivity by classification, coding, and data base standardization, The Key to Maximizing CAD/CAM and Group Technology", Manufacturing Engineering and Material Processing. Marcel Dekksr, Inc. New York and Basel 1981.

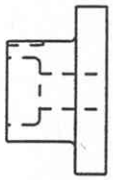
[80] Willshaw, D.J., "Holography, associative memory, and inductive generalization", In G.E. Hinton and J.A. Anderson (Eds), Parallel models of associative memory , pp 83-104, Hillsdale NJ:Earlbaum, 1982.

[81] Yoh-Han Pao., "Functional Link Nets: Removing hidden layers", AI Expert, April 1989.

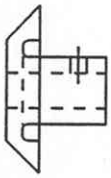
APPENDIX A

COMPONENT PART SKETCHES

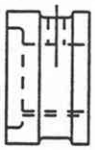
PART 1



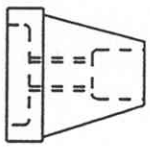
PART 2



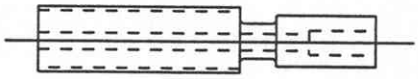
PART 3



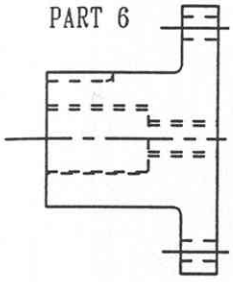
PART 4



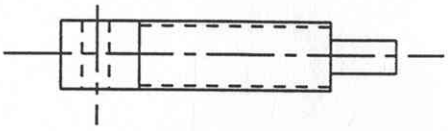
PART 5



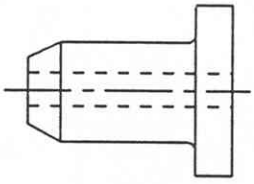
PART 6



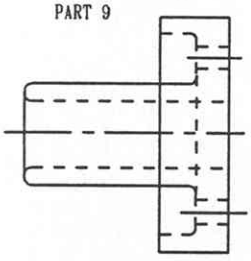
PART 7



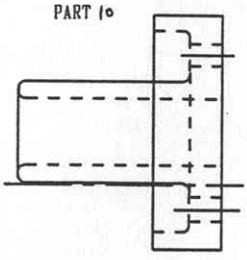
PART 8



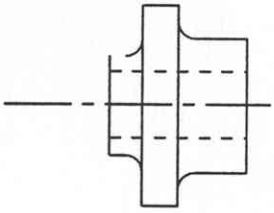
PART 9



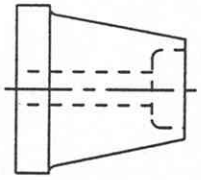
PART 10



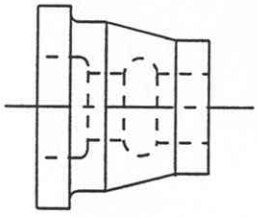
PART 11



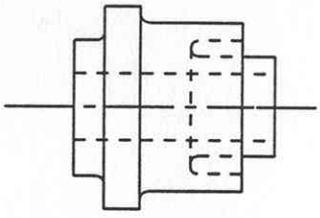
PART 12



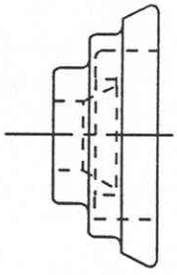
PART 13



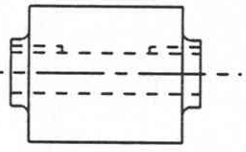
PART 14



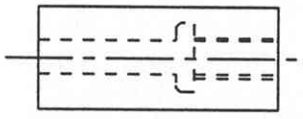
PART 15



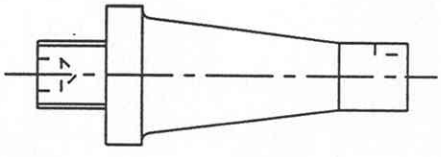
PART 16



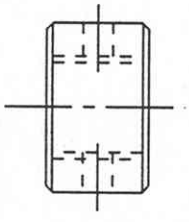
PART 17



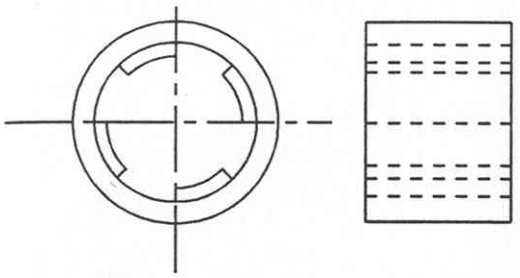
PART 18



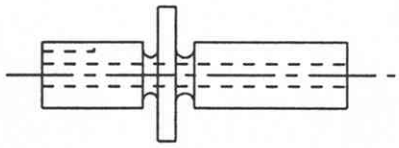
PART 19



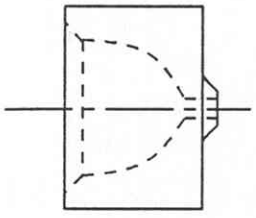
PART 20



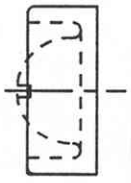
PART 21



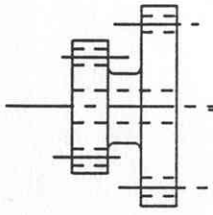
PART 22



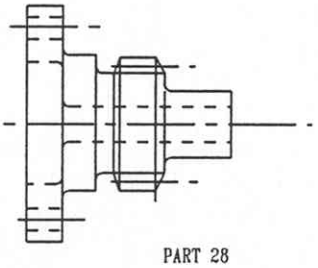
PART 23



PART 24



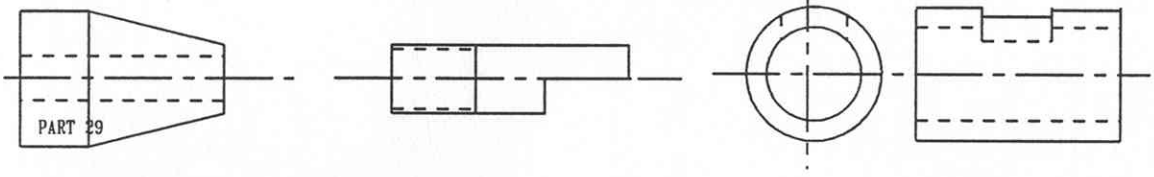
PART 25



PART 26

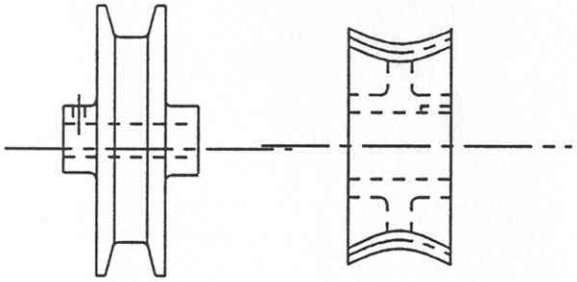
PART 27

PART 28

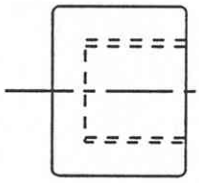


PART 29

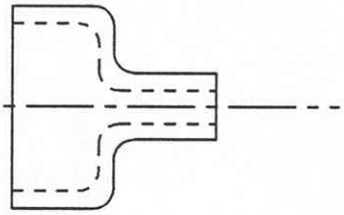
PART 30



PART 31



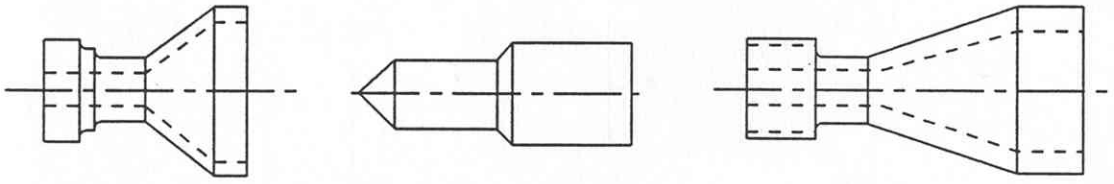
PART 32



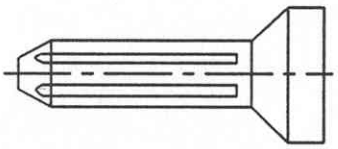
PART 33

PART 34

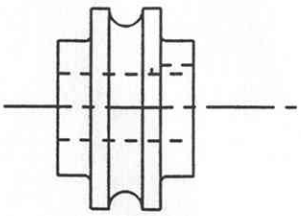
PART 35



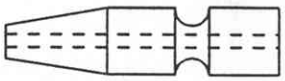
PART 36



PART 37



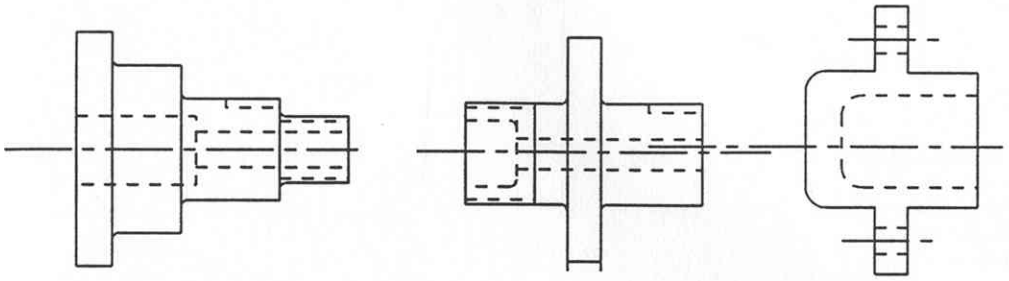
PART 38



PART 39

PART 40

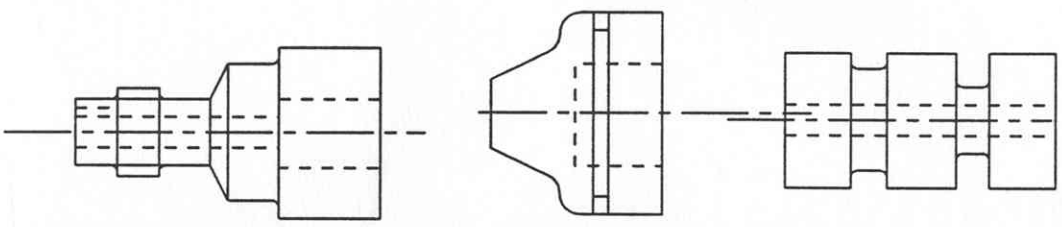
PART 41



PART 42

PART 43

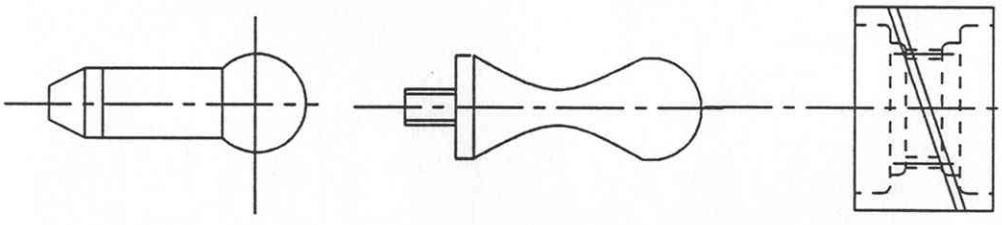
PART 44



PART 45

PART 46

PART 47

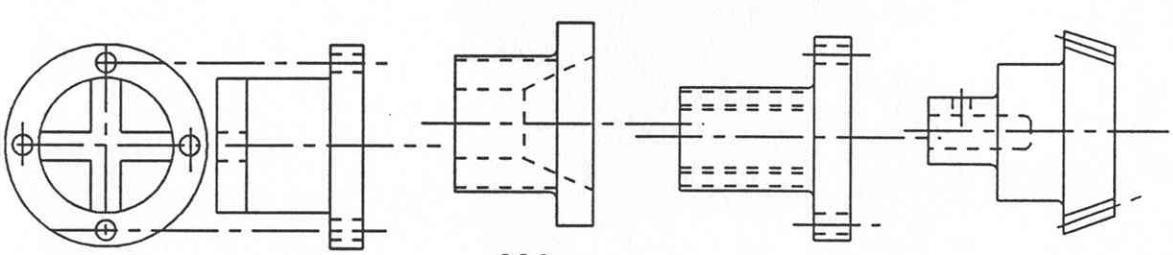


PART 48

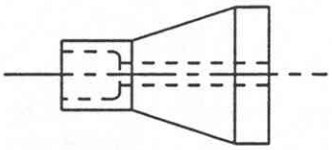
PART 49

PART 50

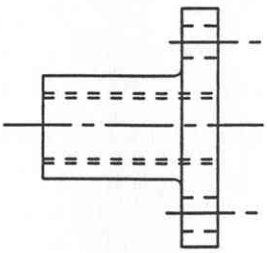
PART 51



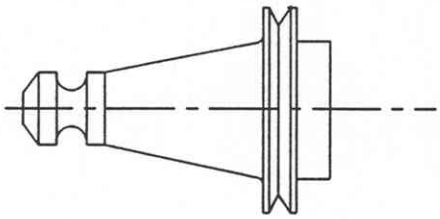
PART 52



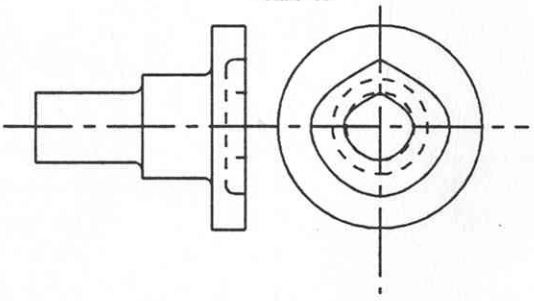
PART 53



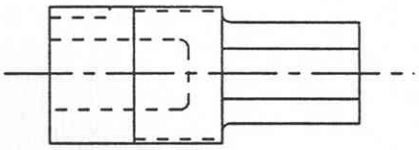
PART 54



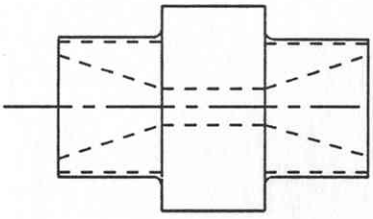
PART 55



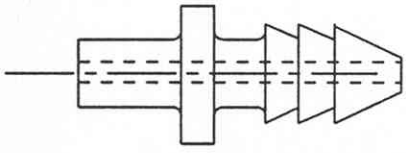
PART 56



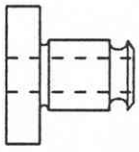
PART 57



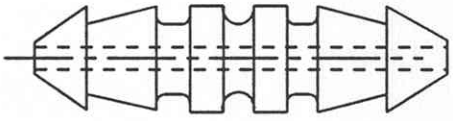
PART 58



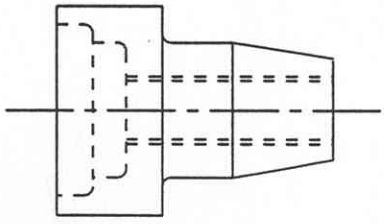
PART 59



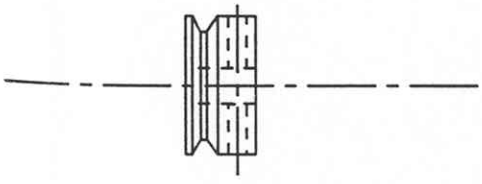
PART 60



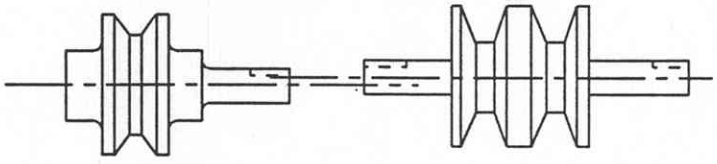
PART 61



PART 62

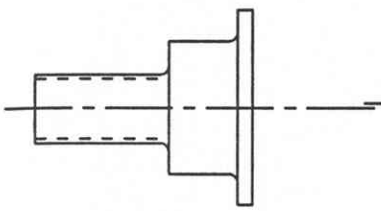


PART 63

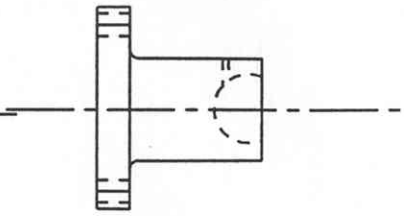


PART 64

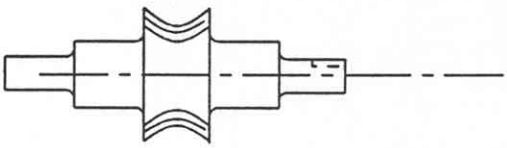
PART 65



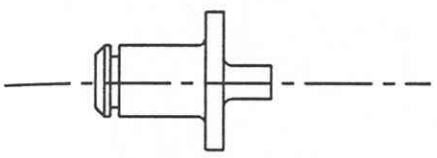
PART 66



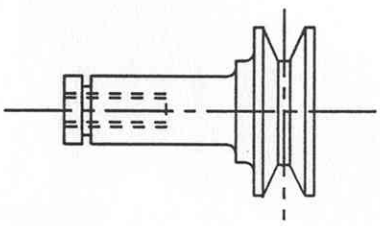
PART 67



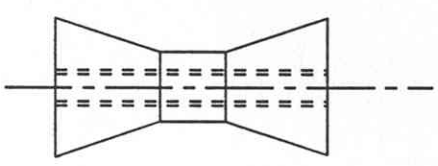
PART 68



PART 69



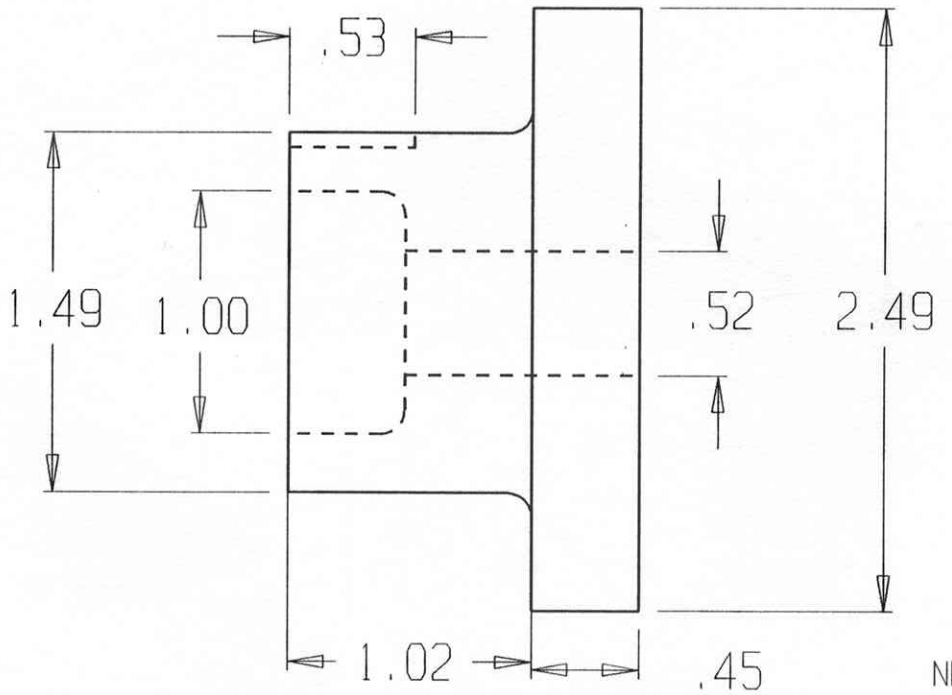
PART 70



APPENDIX B

DETAILED COMPONENT PART SKETCHES

PART 1



NOTE:

ALL DIMENSIONS IN INCHES

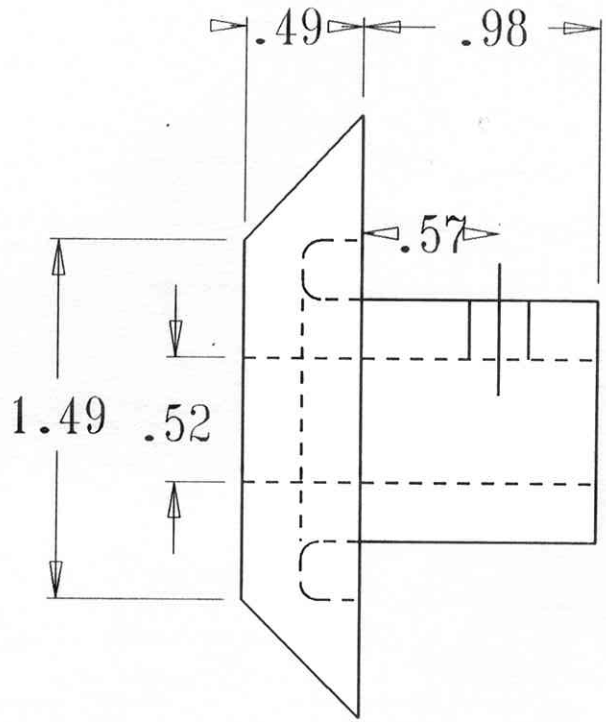
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

ANGULAR +/- 1/2 deg

PART 2



NOTE:

ALL DIMENSIONS IN INCHES

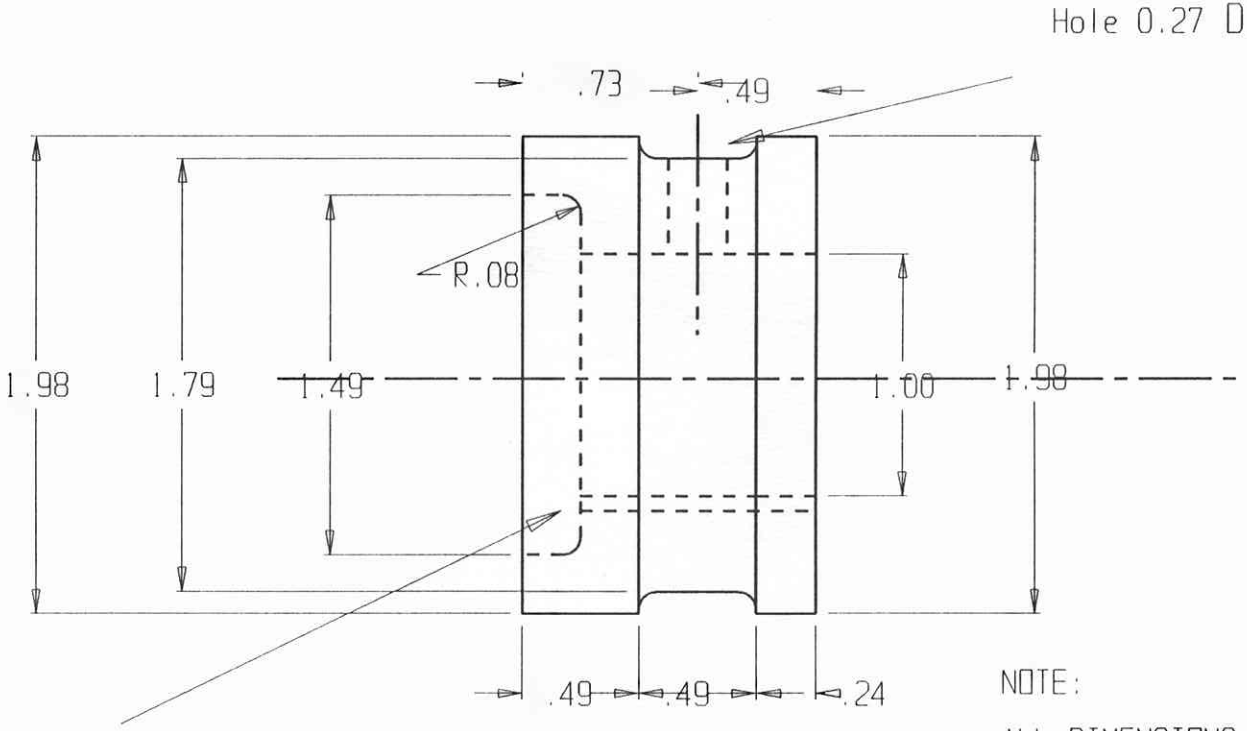
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

ANGULAR +/- 1/2 deg

PART 3



NOTE:

ALL DIMENSIONS IN INCHES

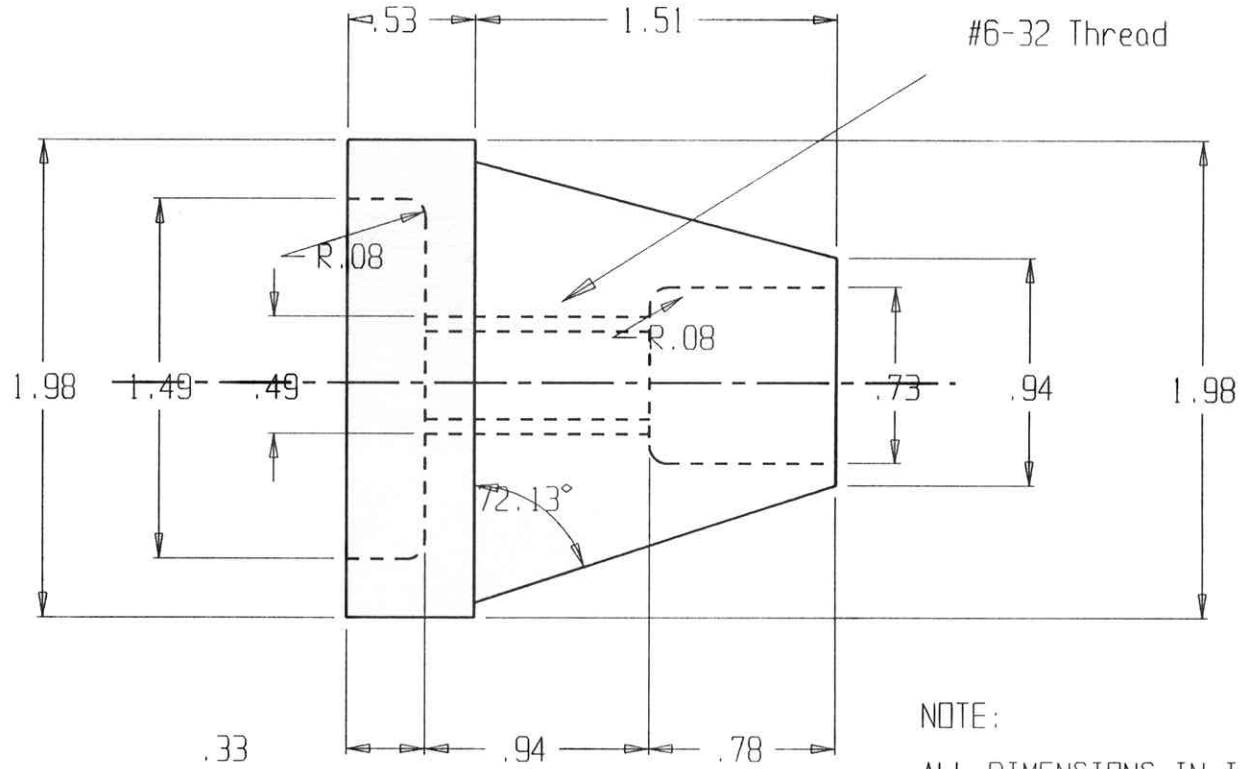
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

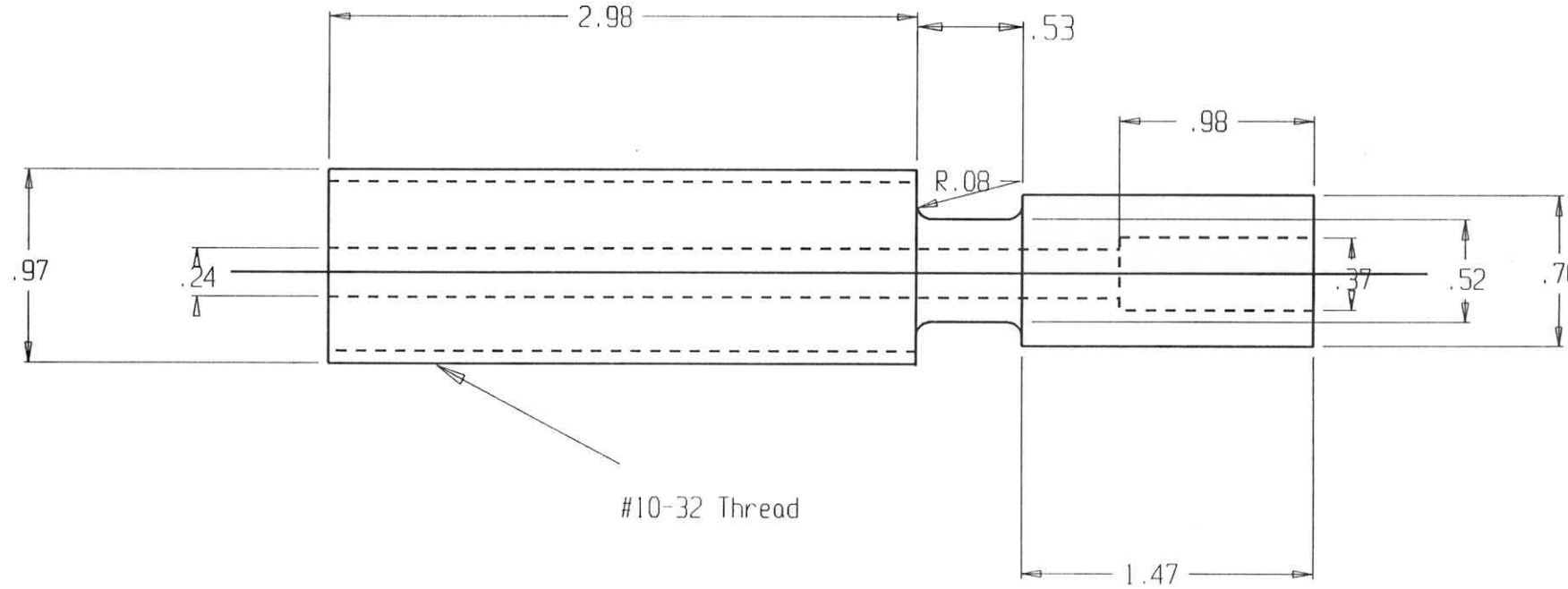
ANGULAR +/- 1/2 deg

PART 4



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 5



NOTE:

ALL DIMENSIONS IN INCHES

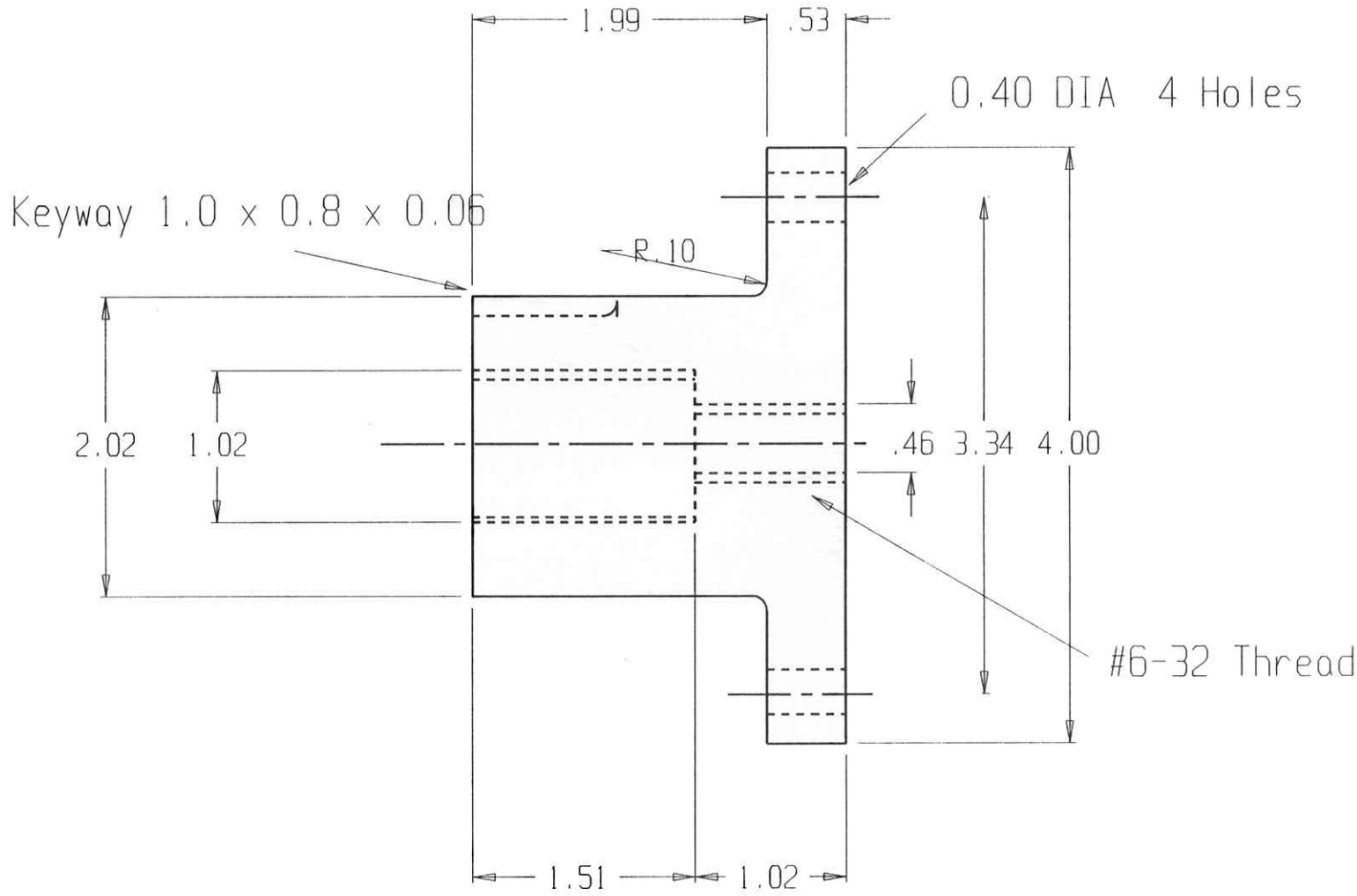
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

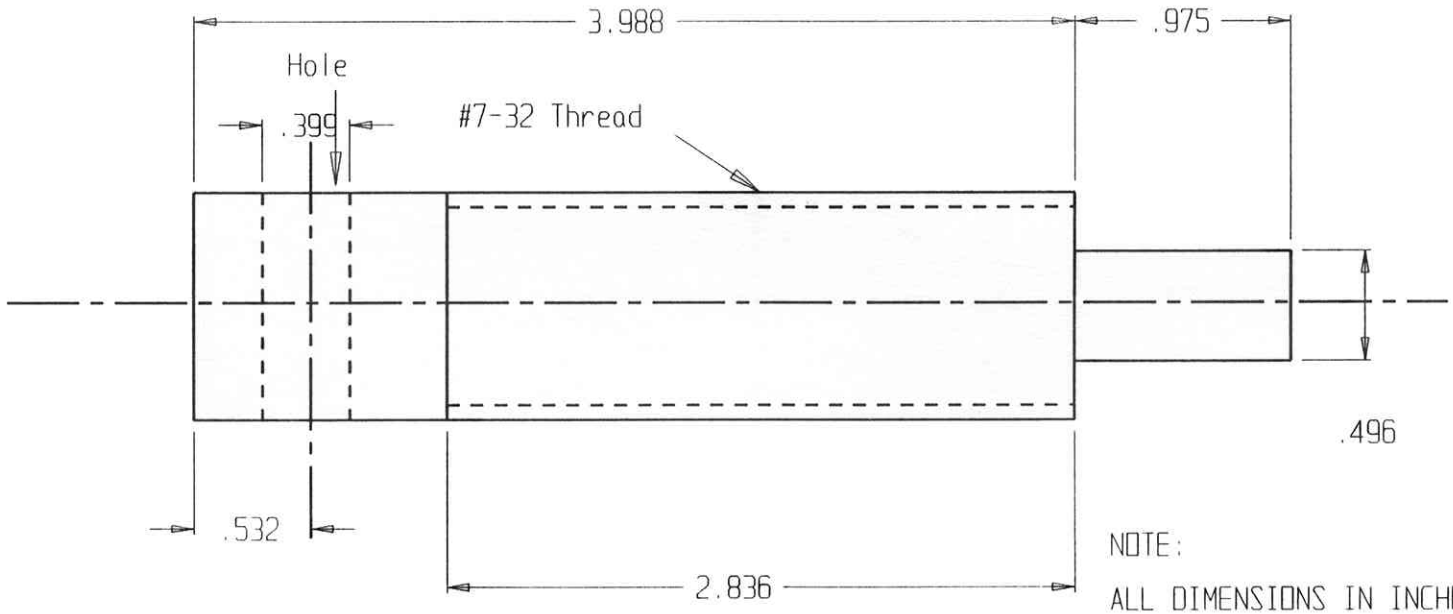
TWO PLC. DEC. +/- 0.02

ANGULAR +/- 1/2 deg

PART 6

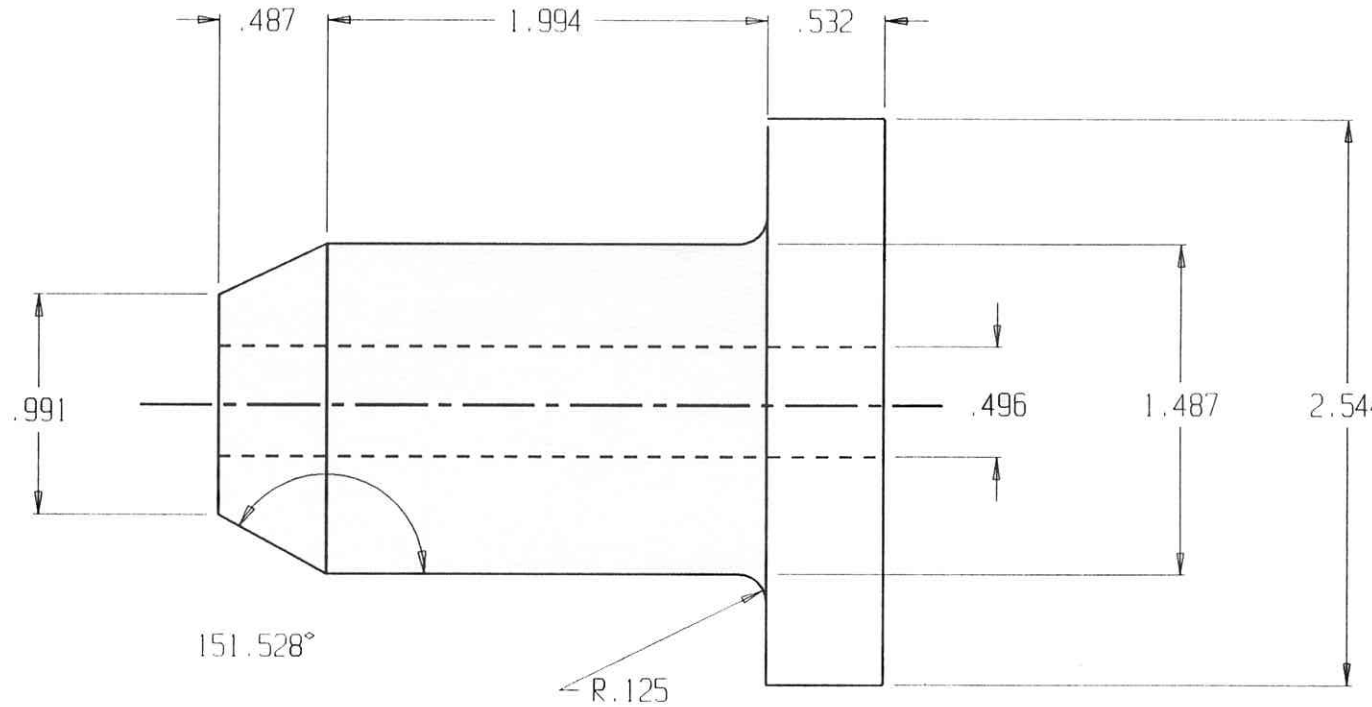

1.99
.53
0.40 DIA 4 Holes
Keyway 1.0 x 0.8 x 0.06
R.10
2.02
1.02
.46
3.34
4.00
#6-32 Thread
1.51
1.02

PART 7



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 8



NOTE:

ALL DIMENSIONS IN INCHES

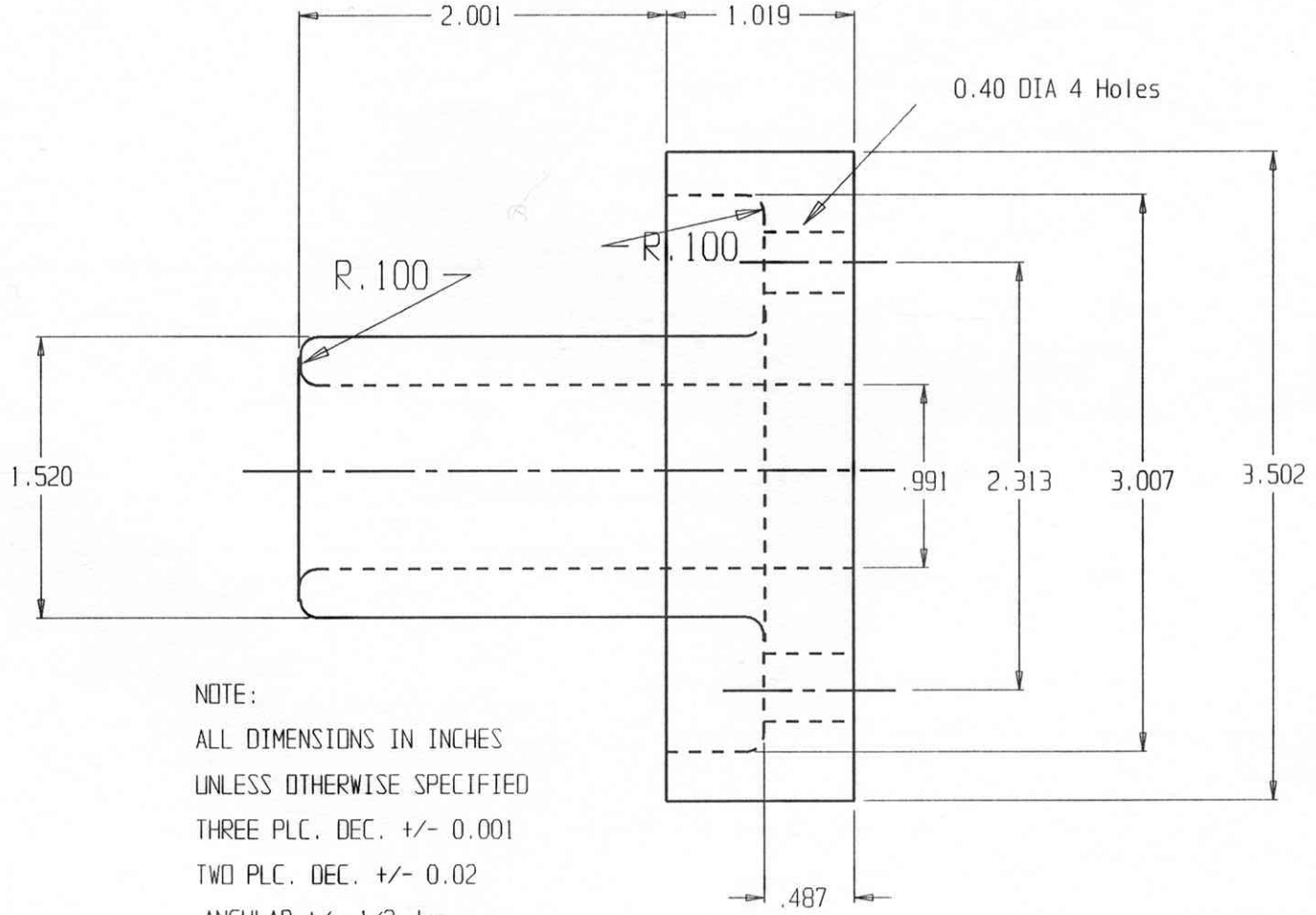
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

ANGULAR +/- 1/2 deg

PART 9



NOTE:

ALL DIMENSIONS IN INCHES

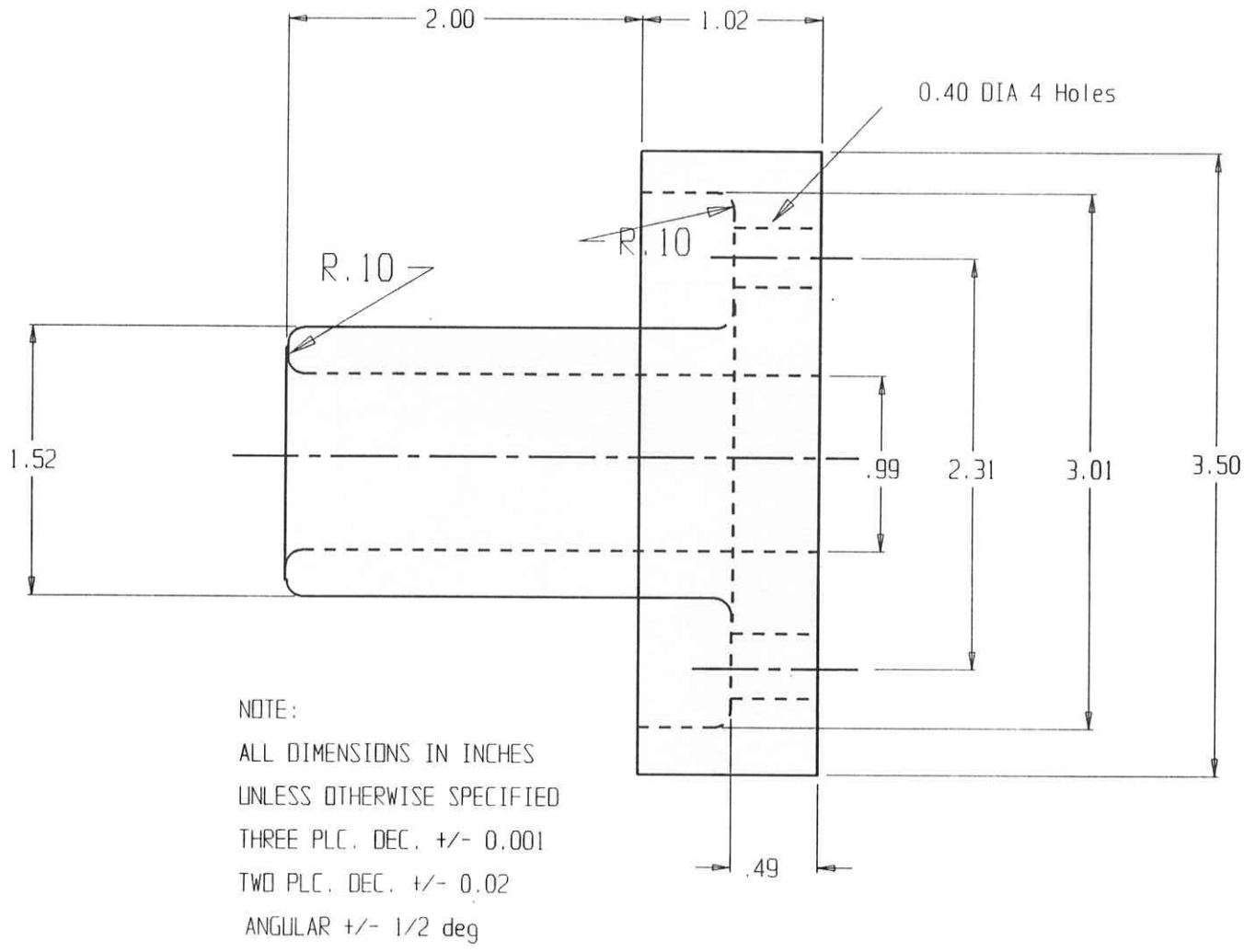
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

ANGULAR +/- 1/2 deg

PART 10



NOTE:

ALL DIMENSIONS IN INCHES

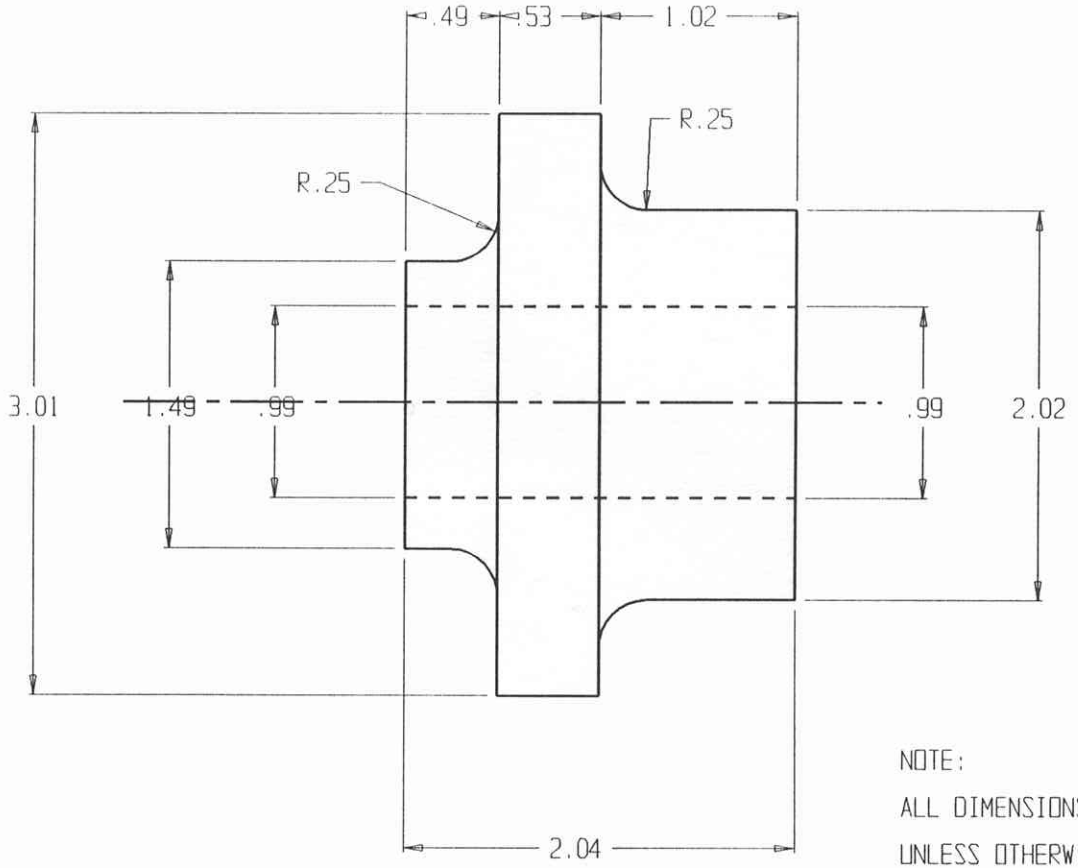
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

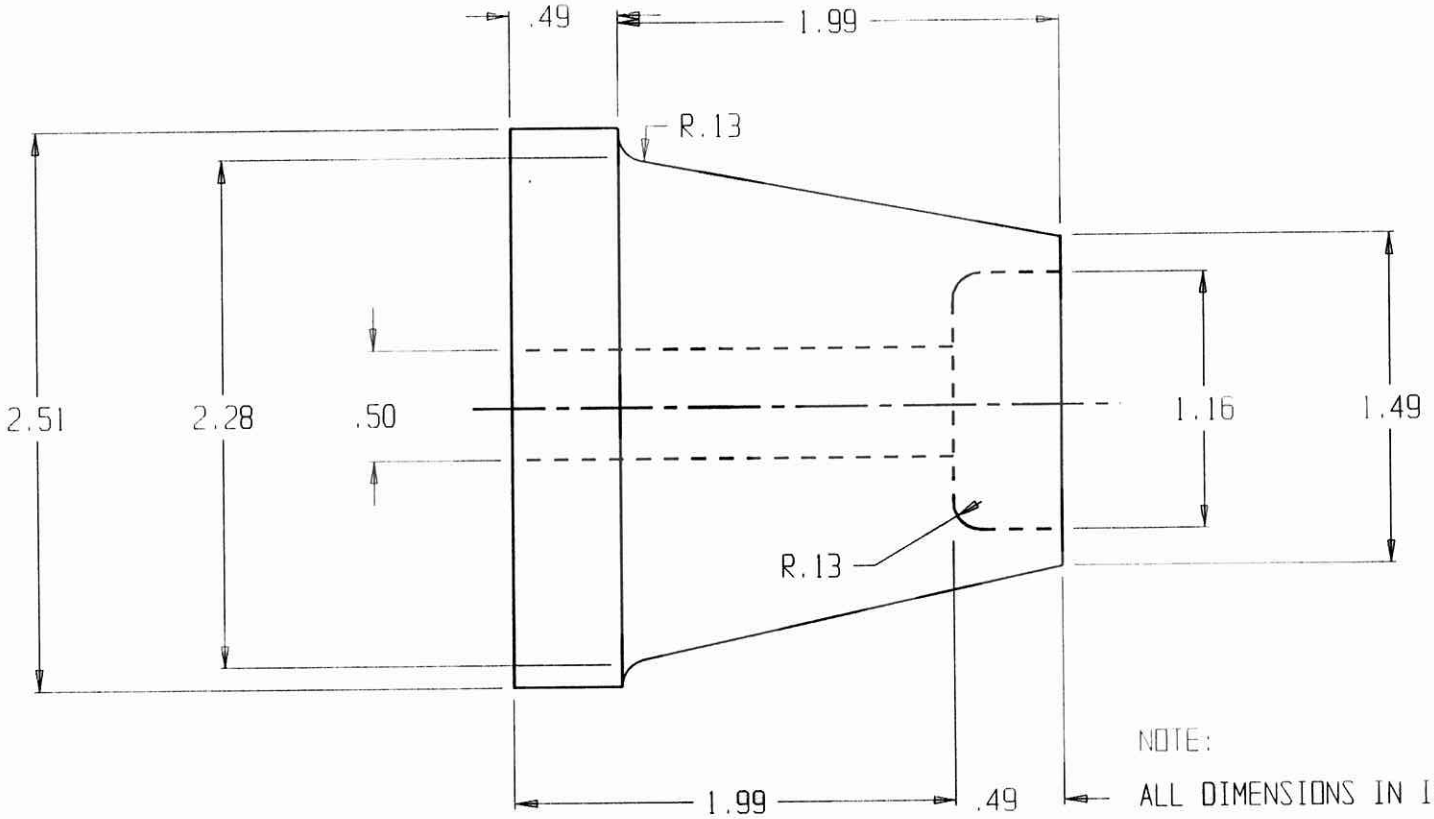
ANGULAR +/- 1/2 deg

PART 11



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 12



NOTE:

ALL DIMENSIONS IN INCHES

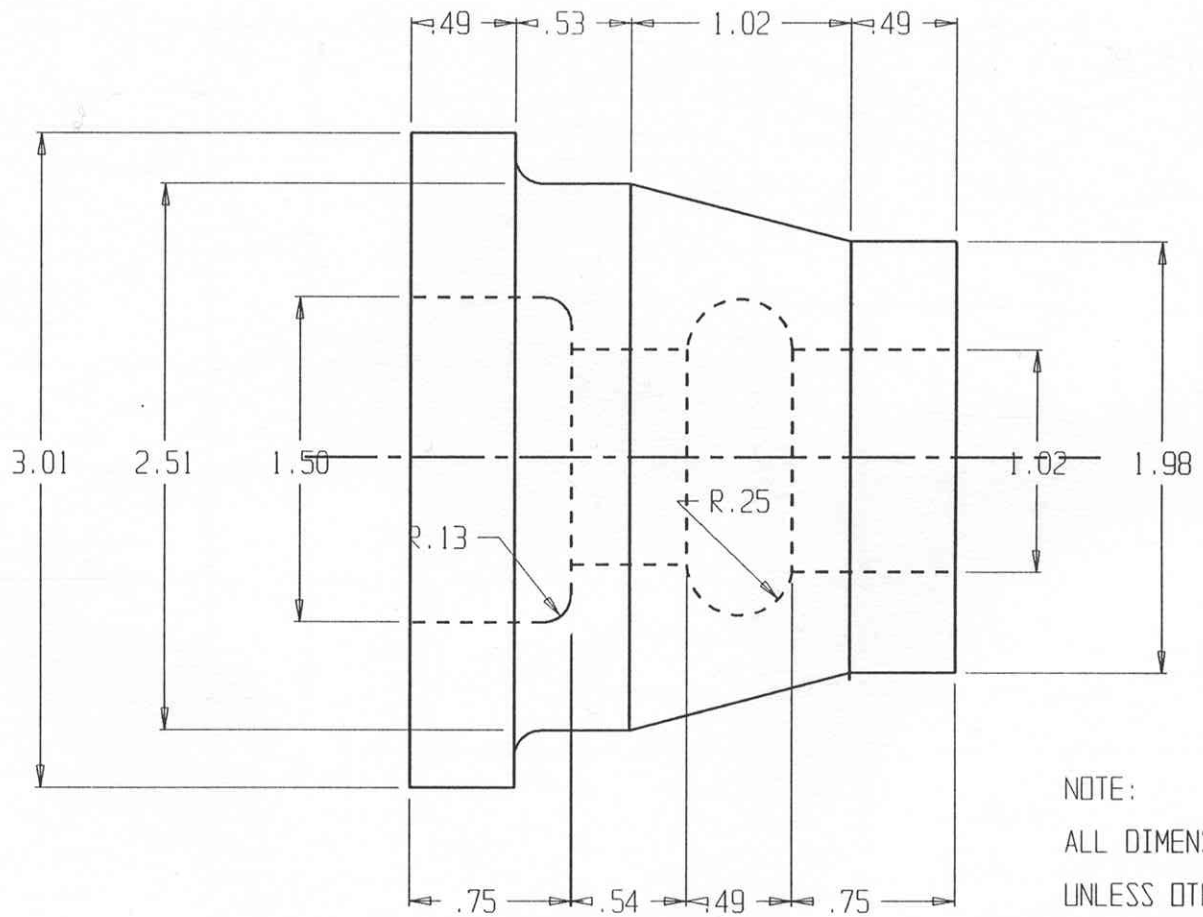
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

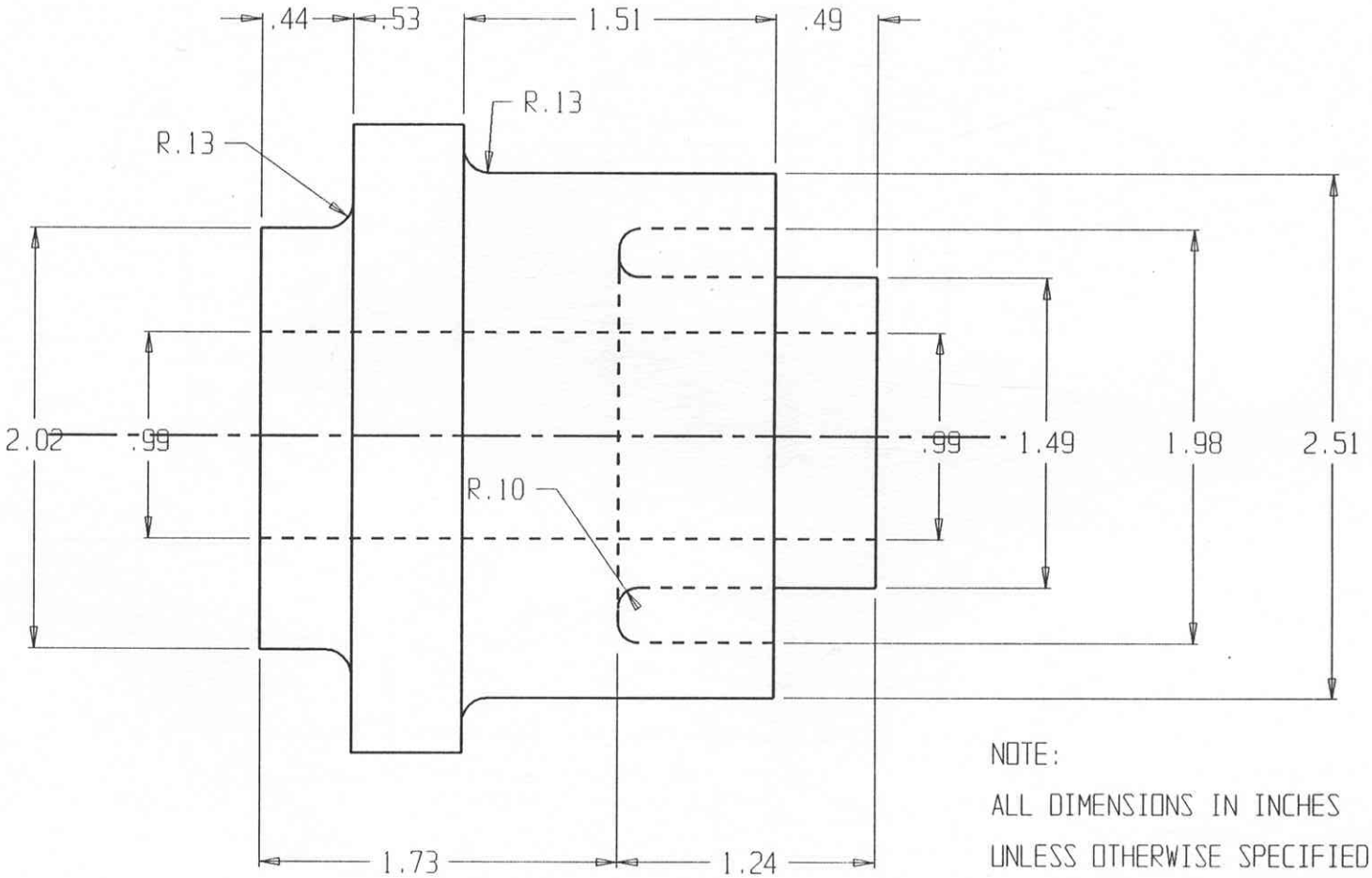
ANGULAR +/- 1/2 deg

PART 13



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 14



NOTE:

ALL DIMENSIONS IN INCHES

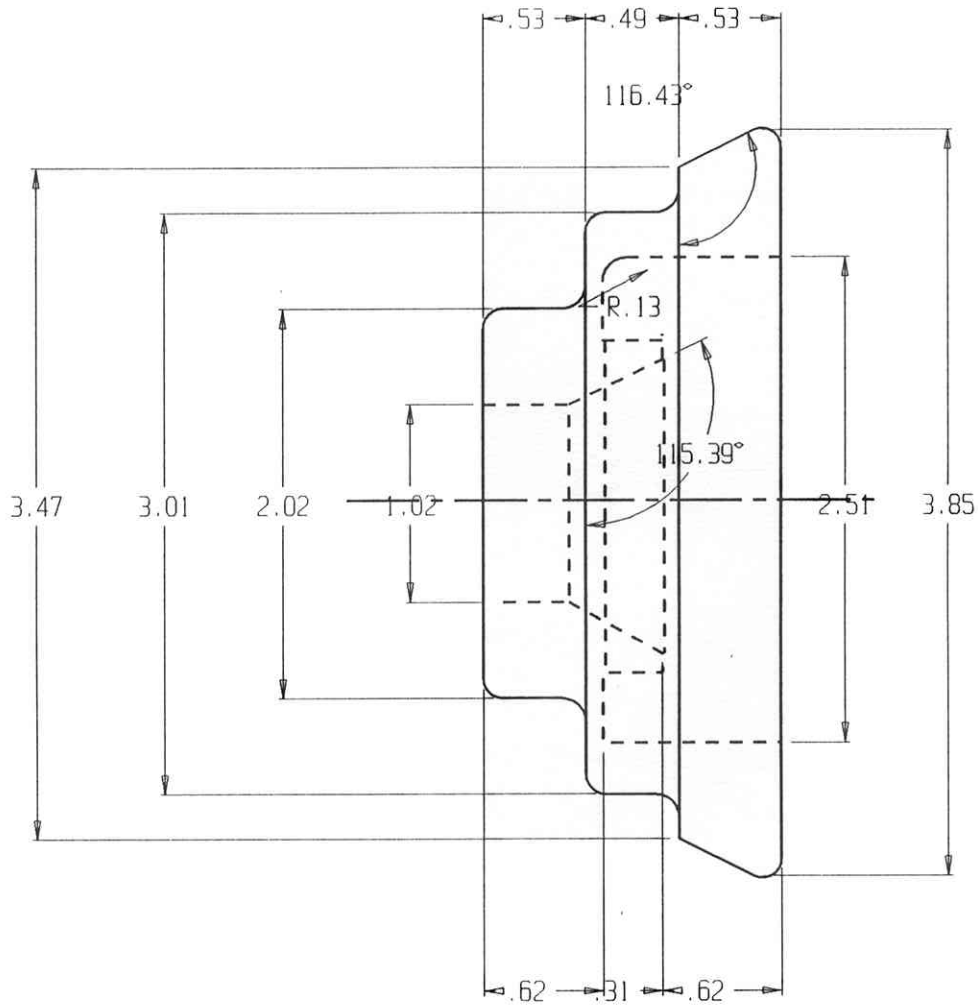
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

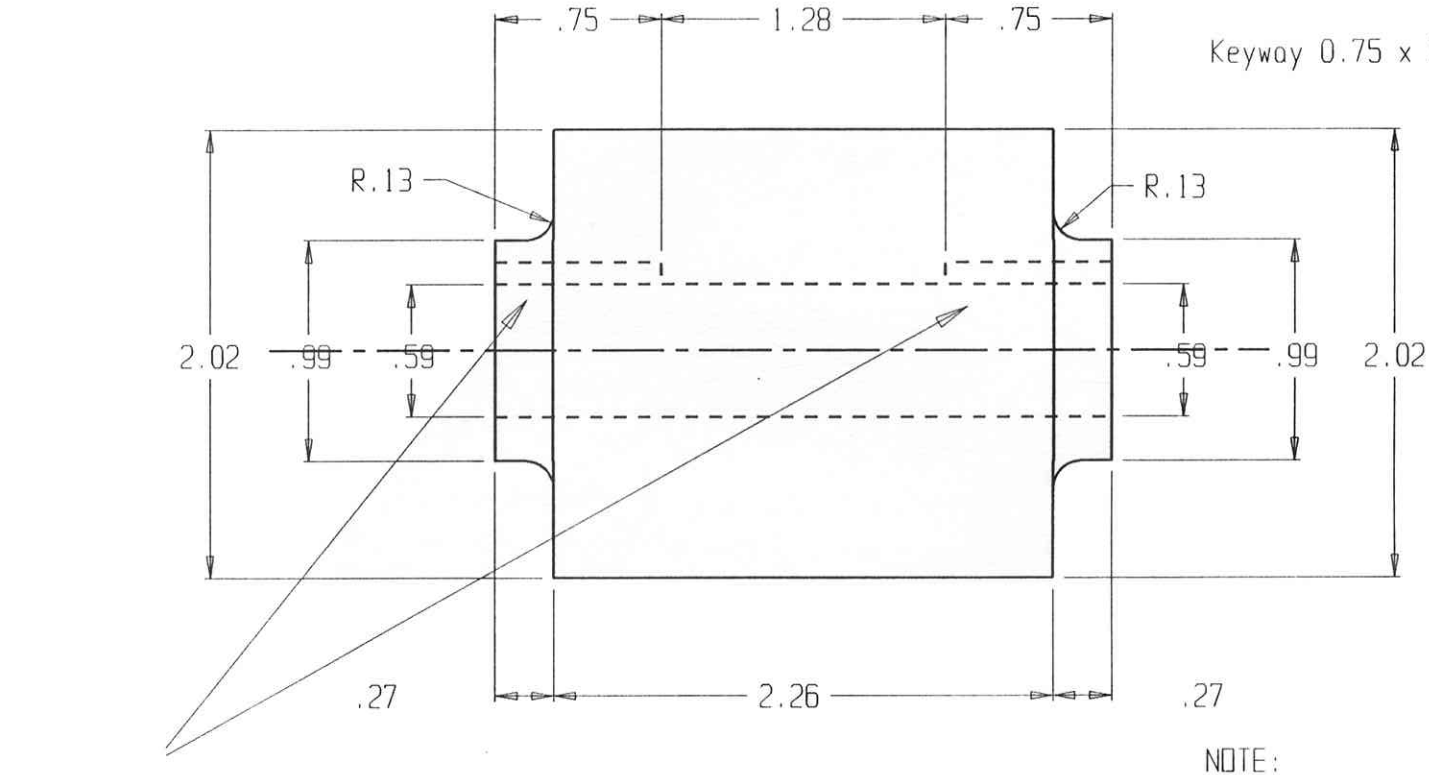
ANGULAR +/- 1/2 deg

PART 15



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 16



NOTE:

ALL DIMENSIONS IN INCHES

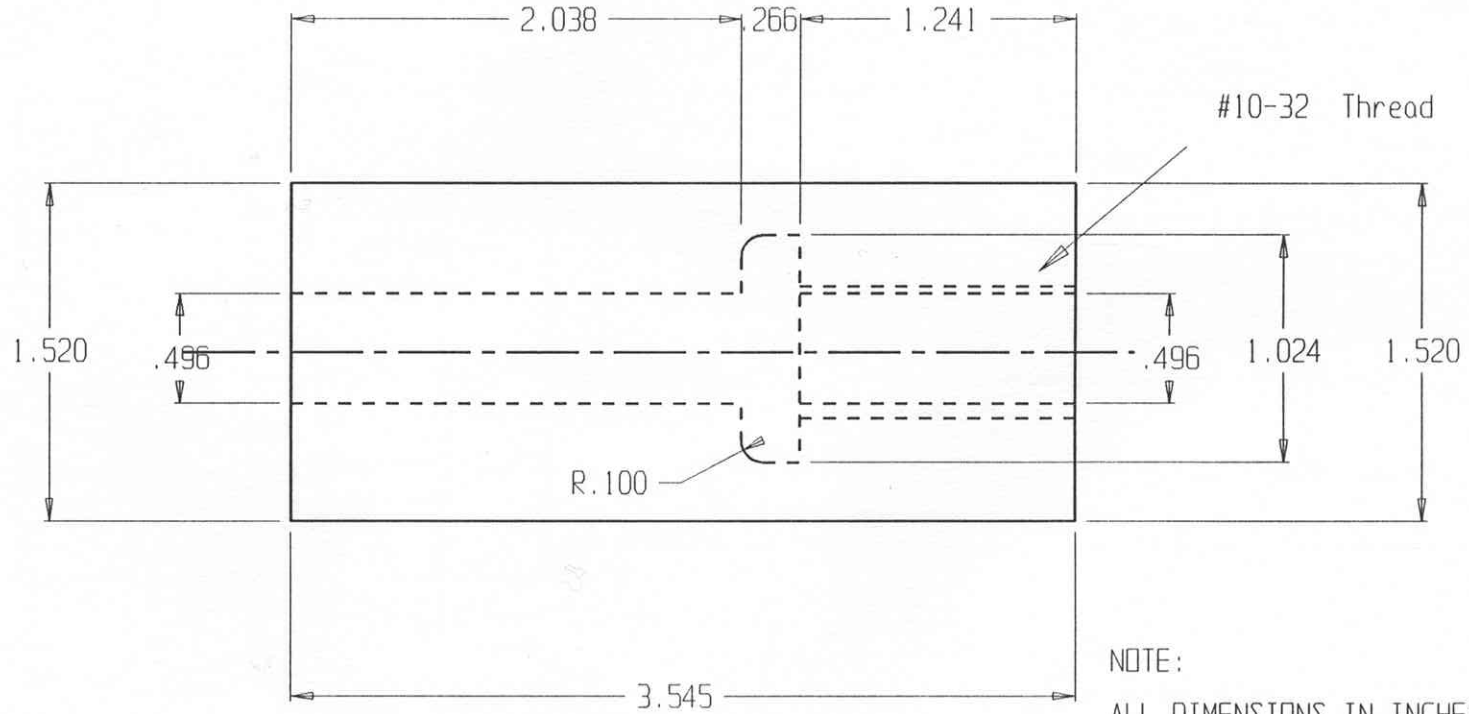
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

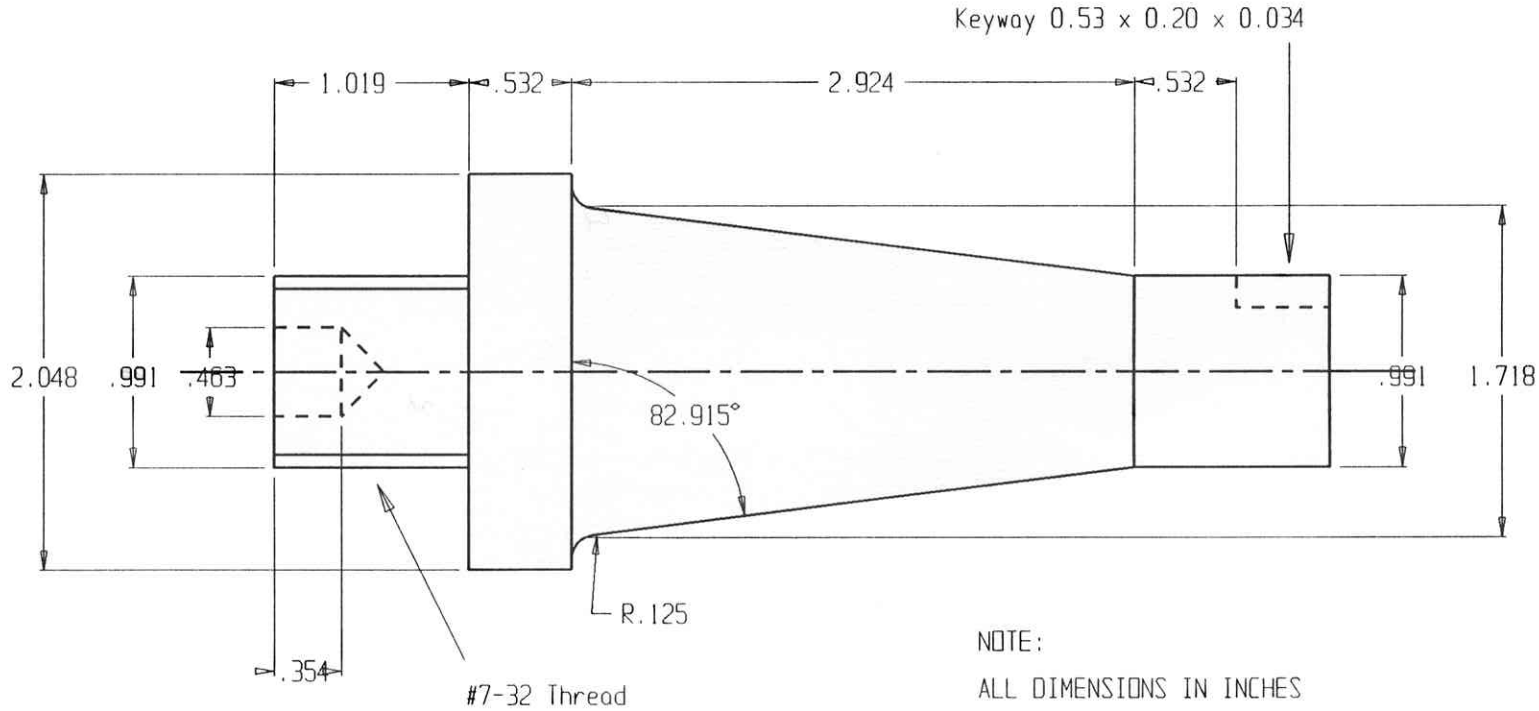
ANGULAR +/- 1/2 deg

PART 17



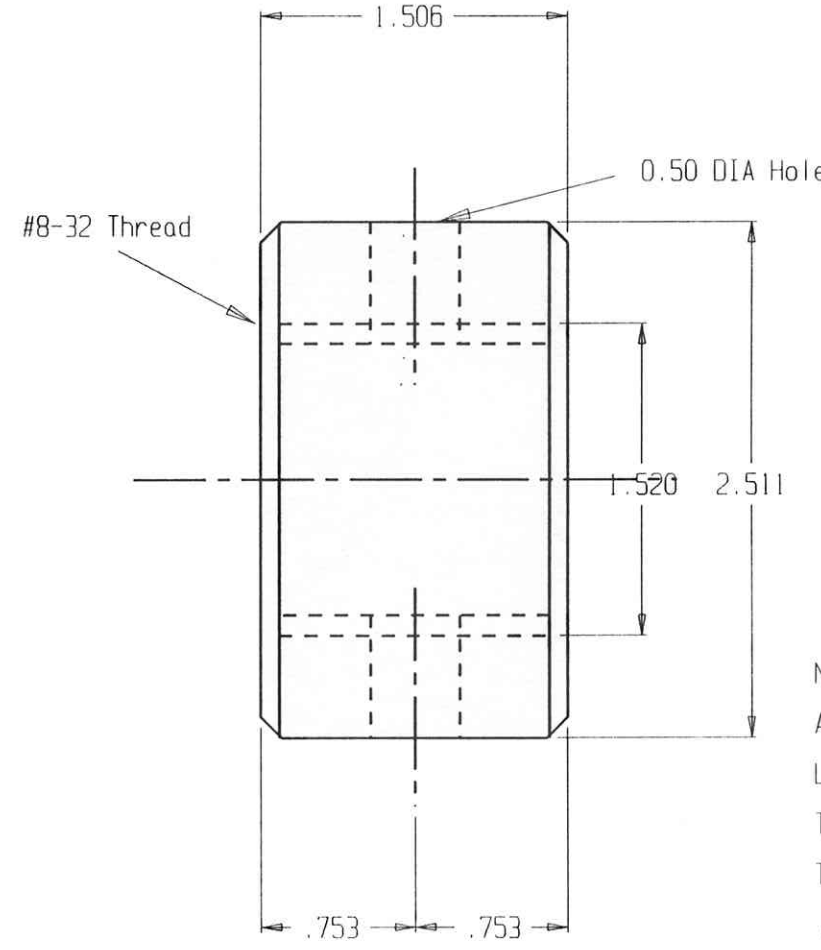
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 18



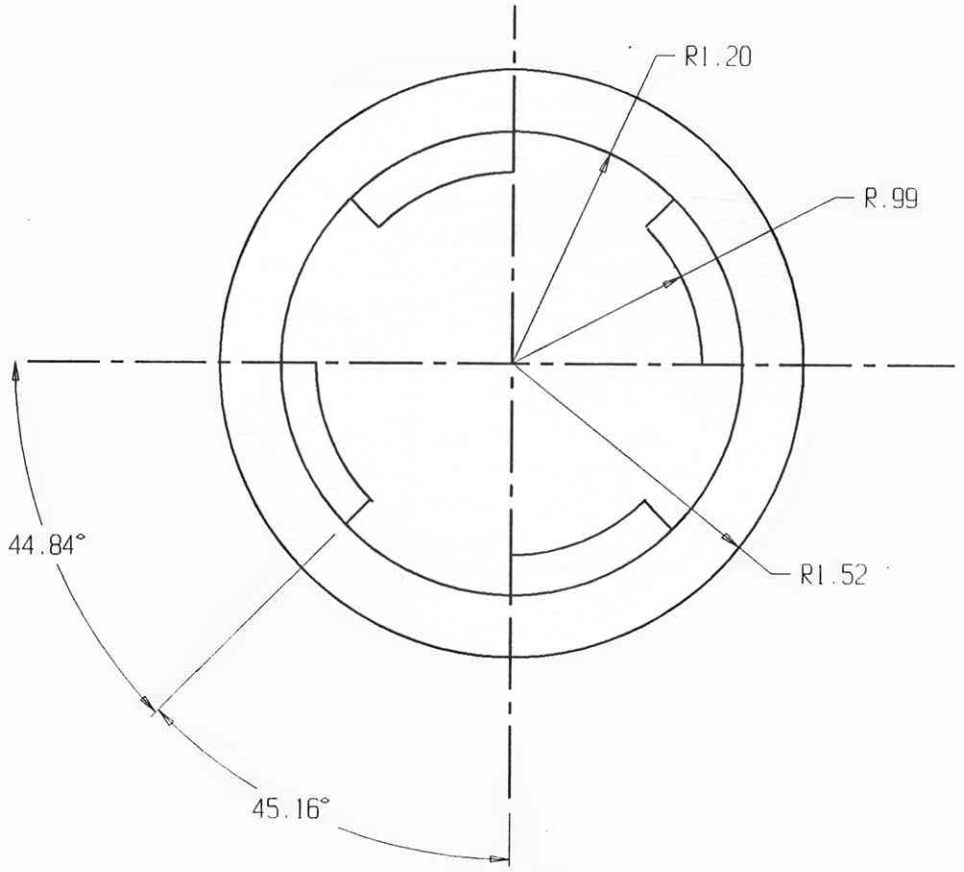
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

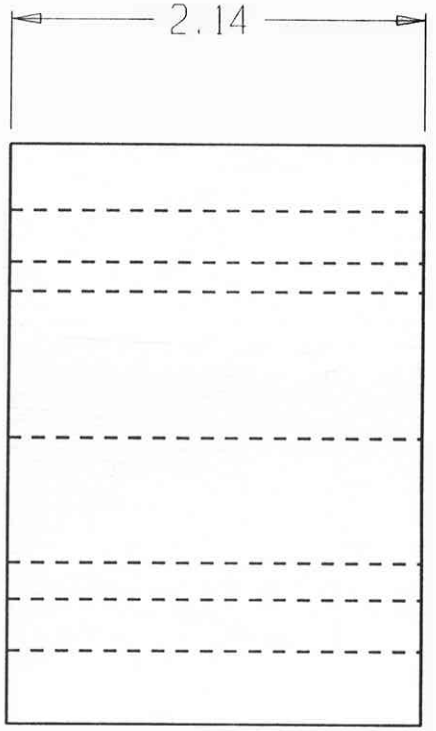
PART 19



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 20





NOTE:

ALL DIMENSIONS IN INCHES

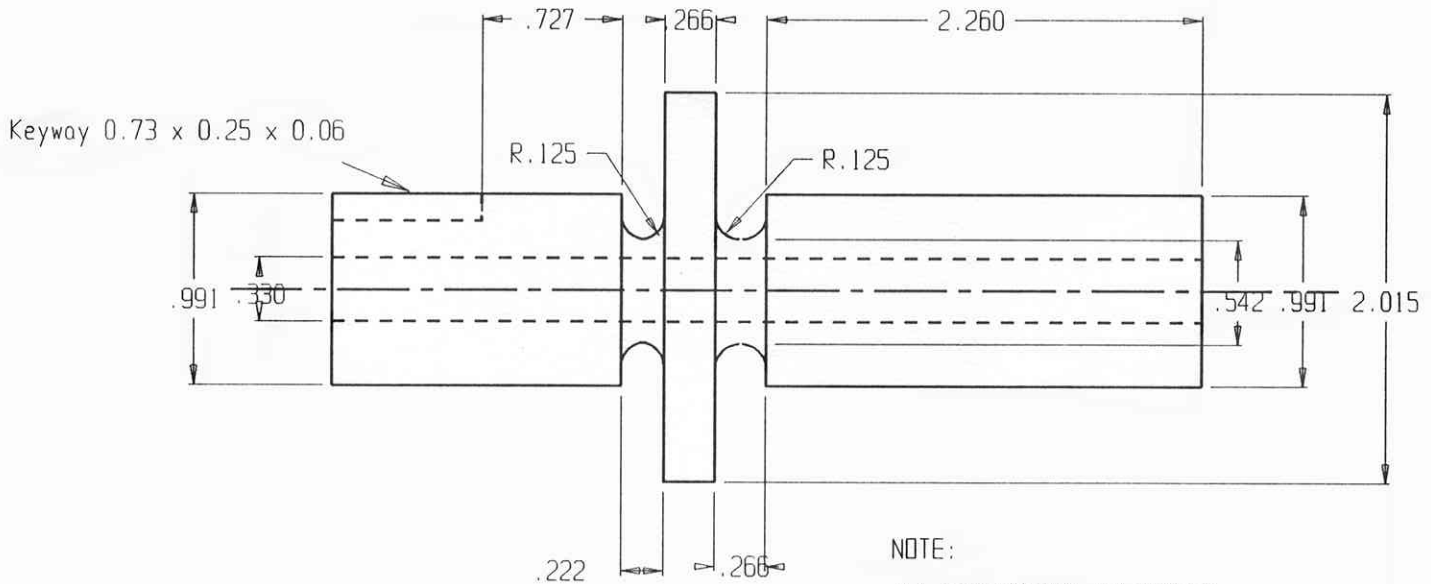
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

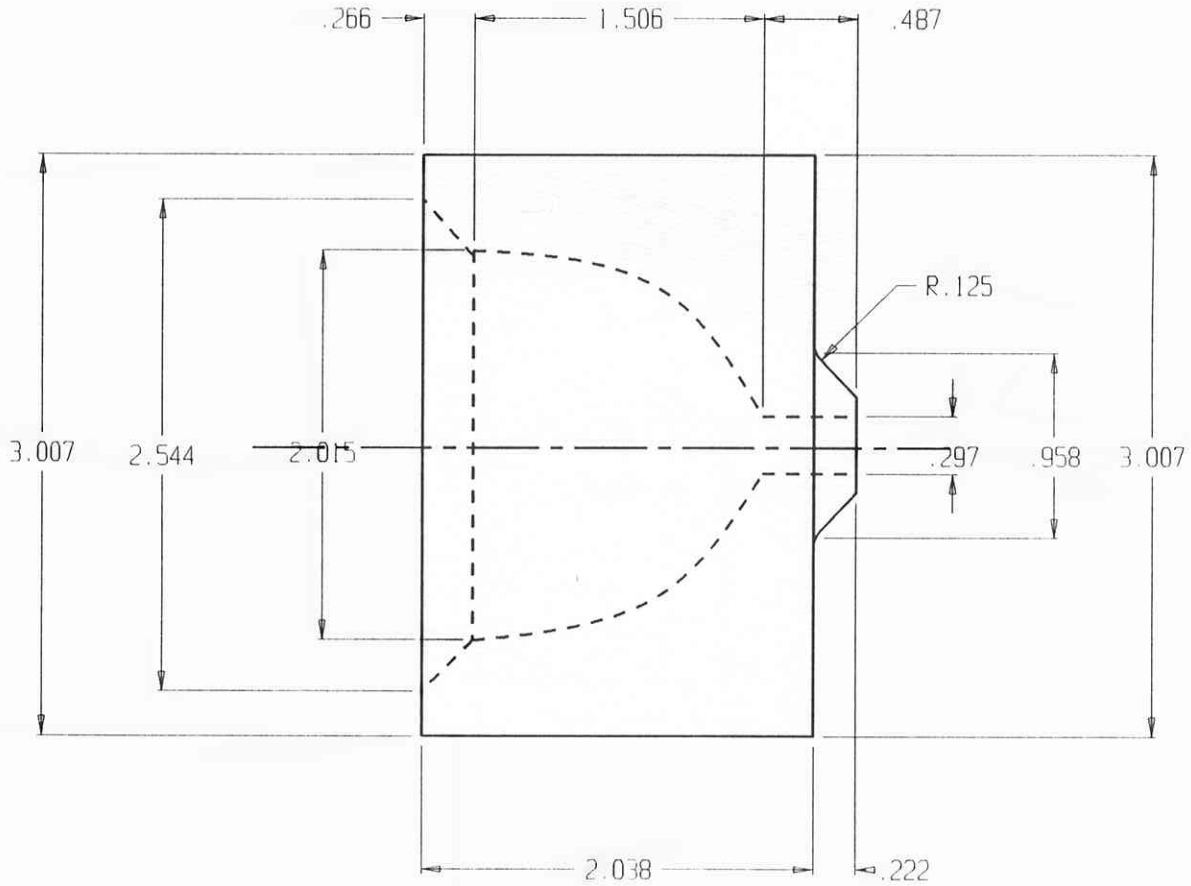
ANGULAR +/- 1/2 deg

PART 21



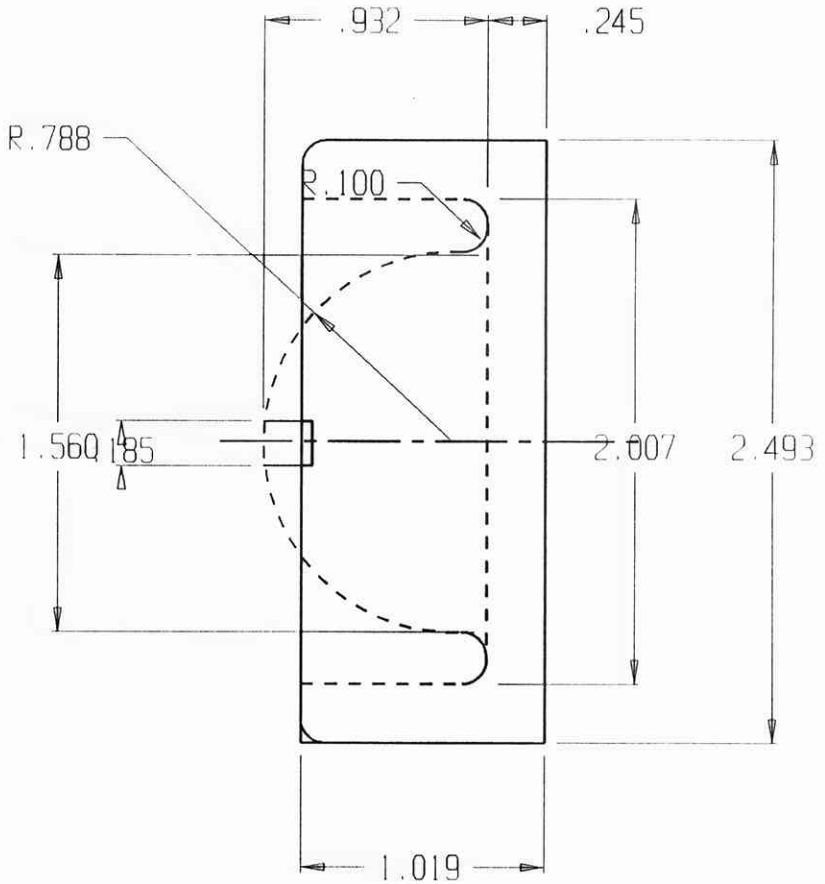
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 22



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 23



NOTE:

ALL DIMENSIONS IN INCHES

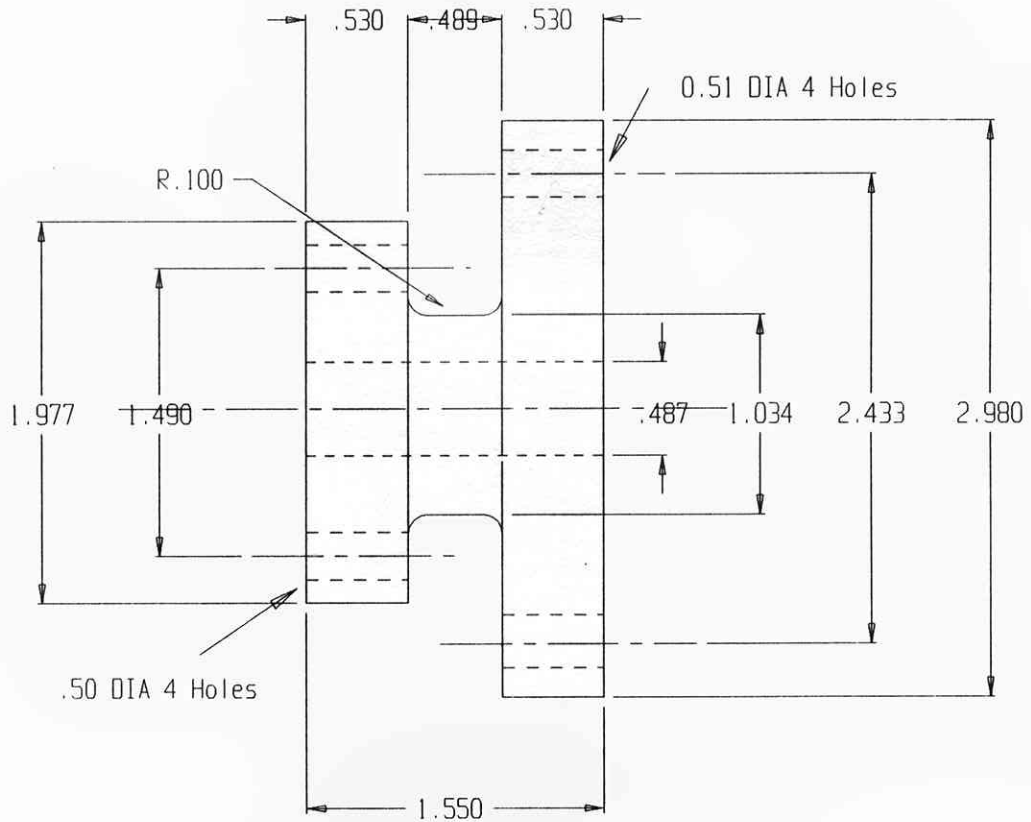
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

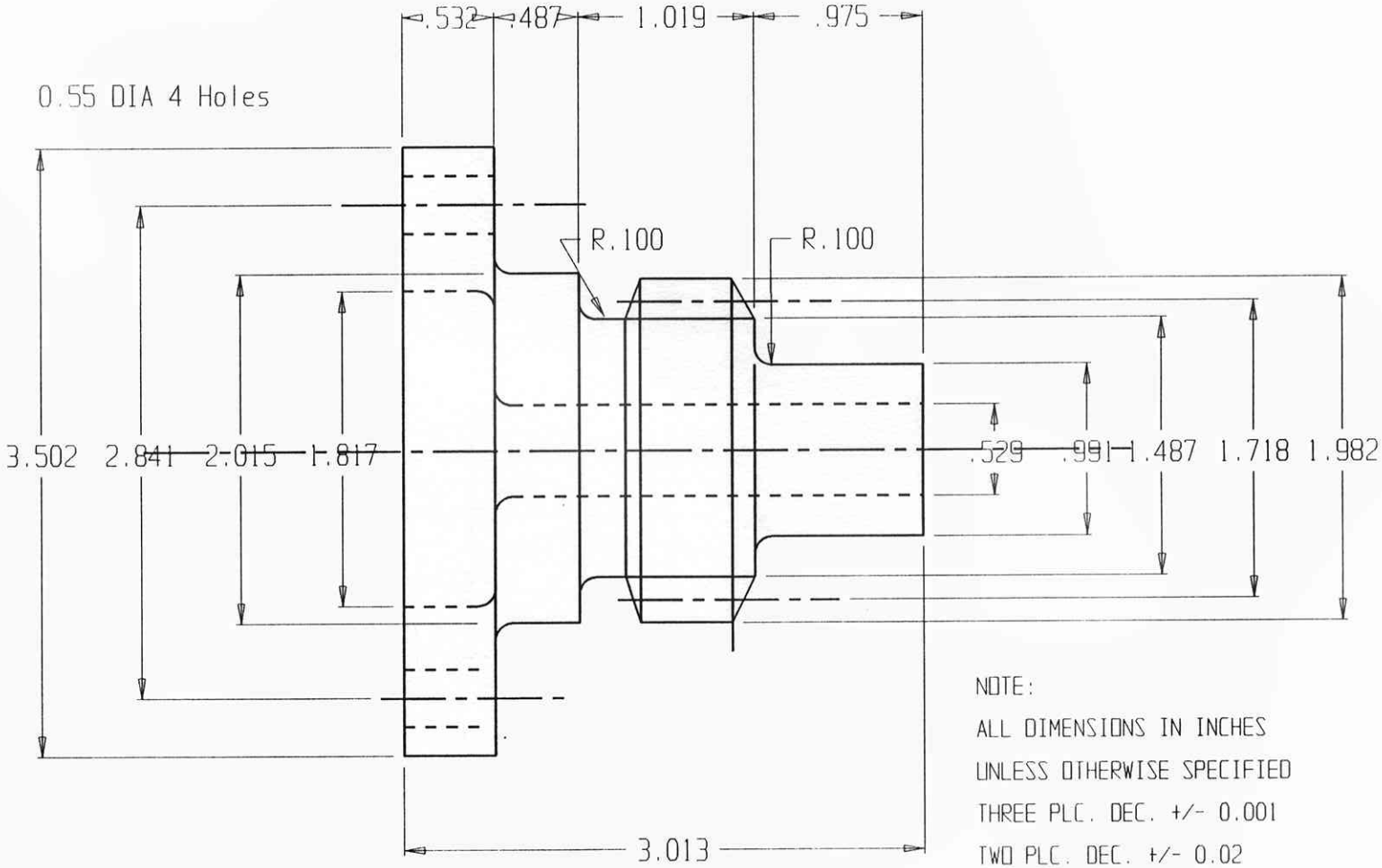
ANGULAR +/- 1/2 deg

PART 24



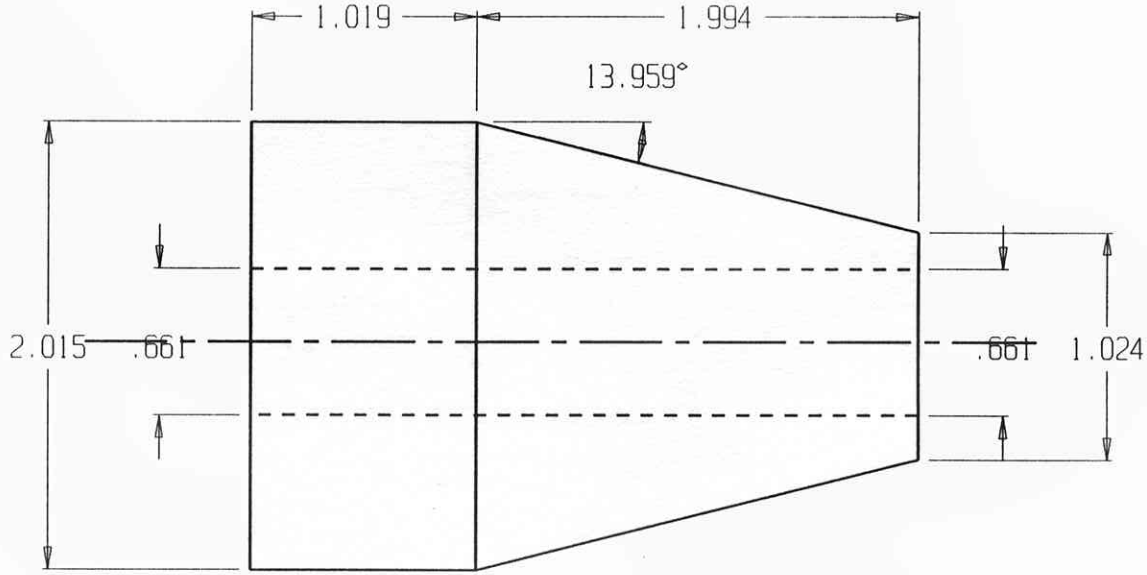
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 25



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 26



NOTE:

ALL DIMENSIONS IN INCHES

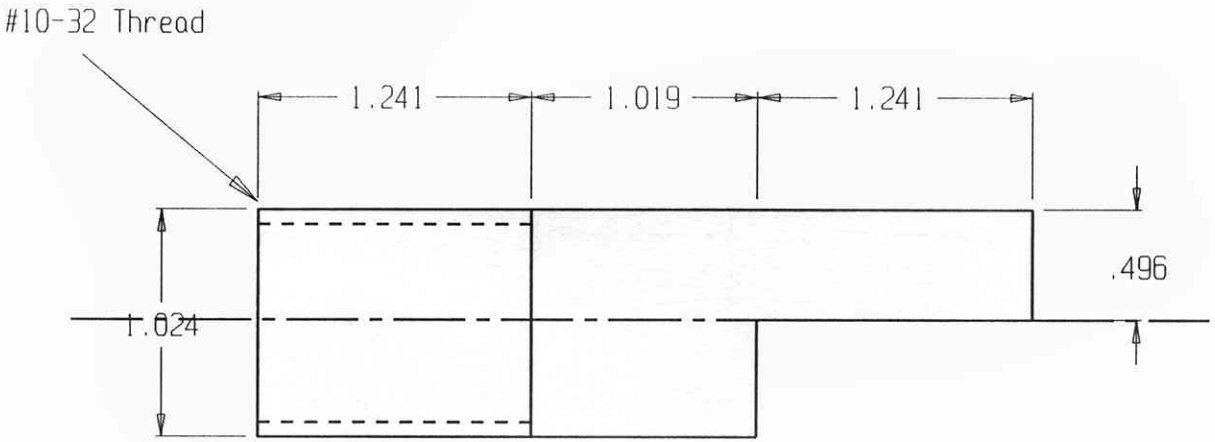
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

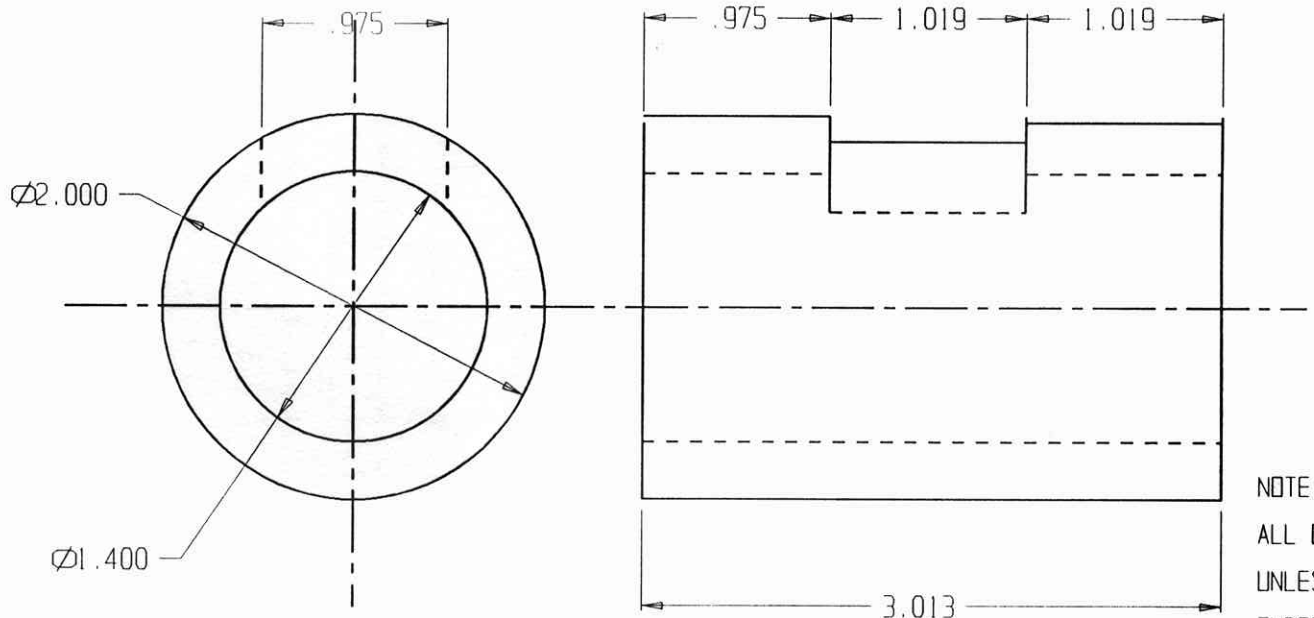
ANGULAR +/- 1/2 deg

PART 27



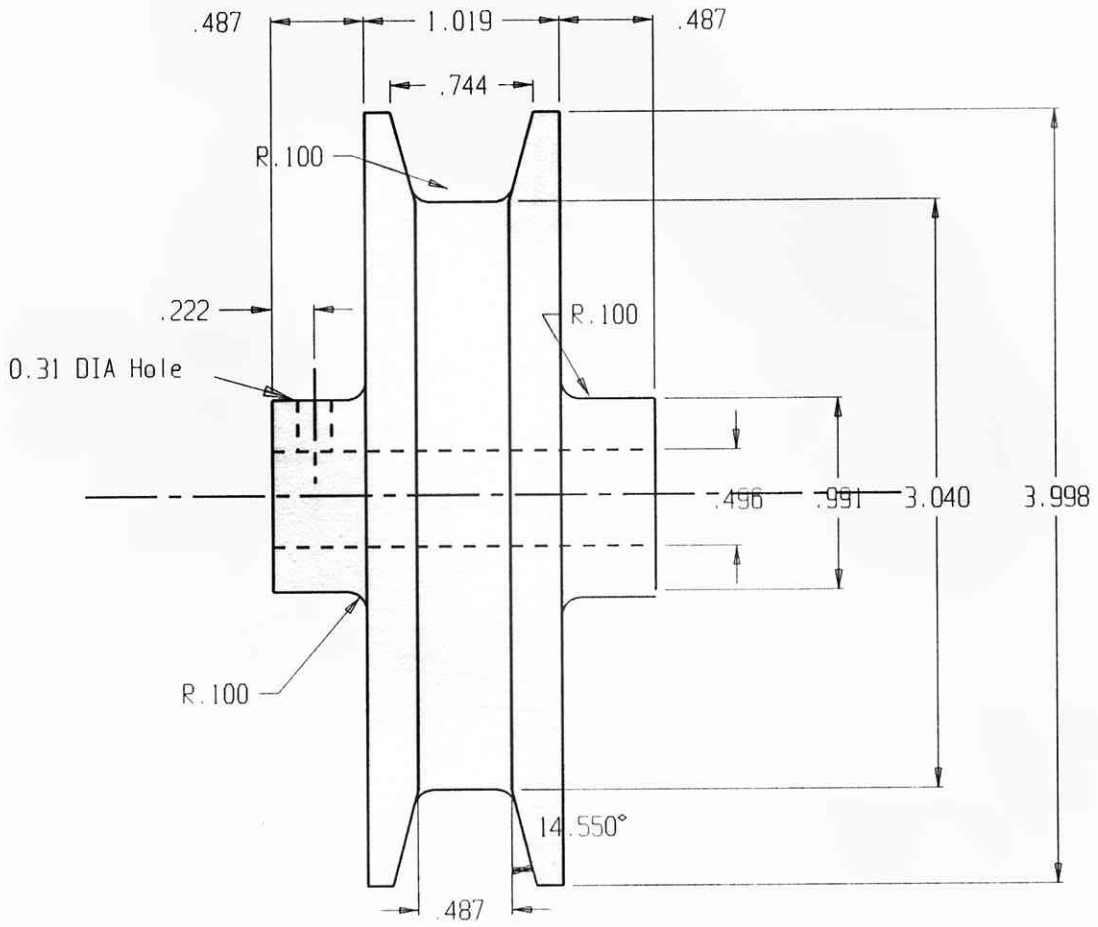
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 28



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 29



NOTE:

ALL DIMENSIONS IN INCHES

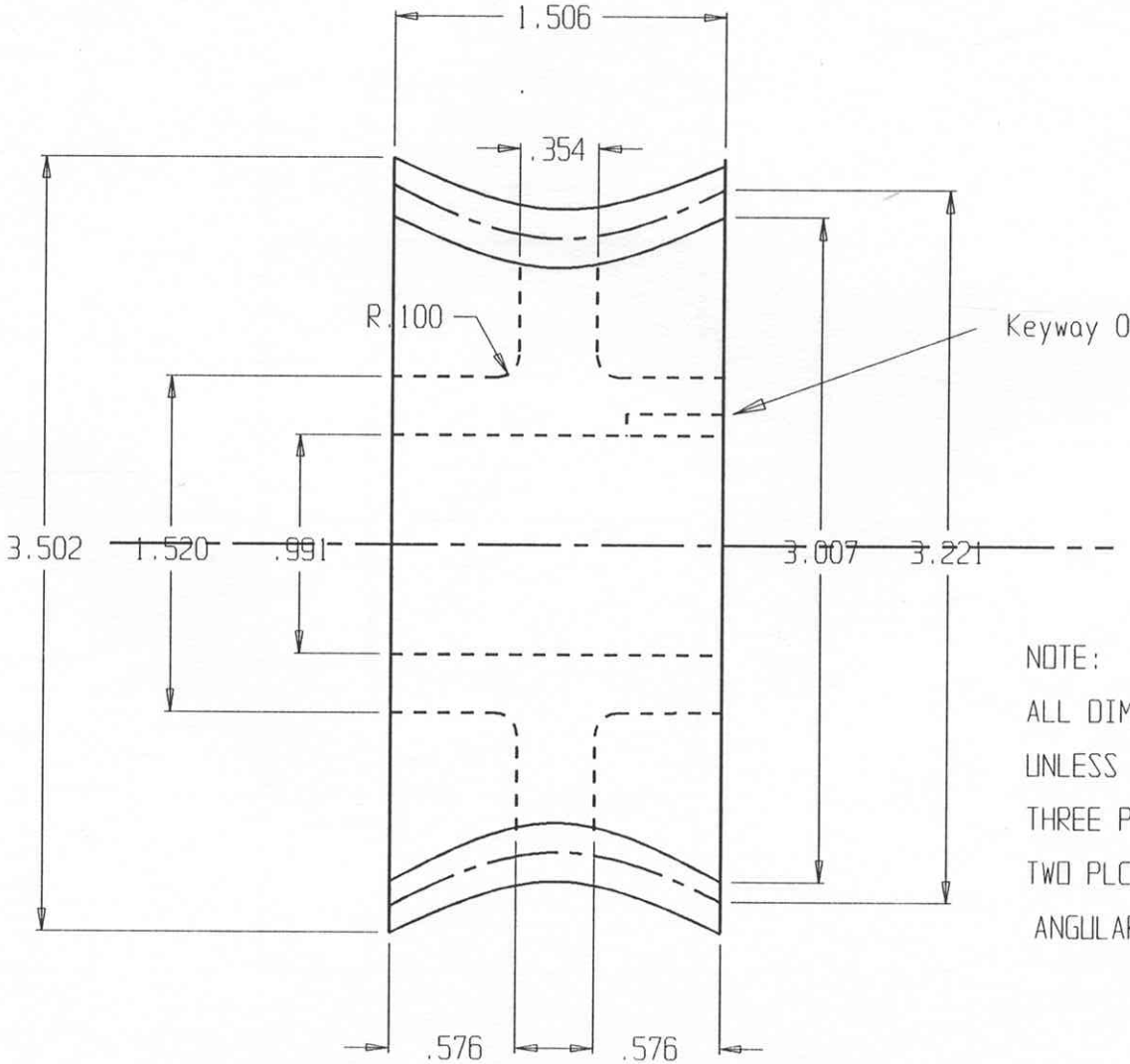
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

ANGULAR +/- 1/2 deg

PART 30



NOTE:

ALL DIMENSIONS IN INCHES

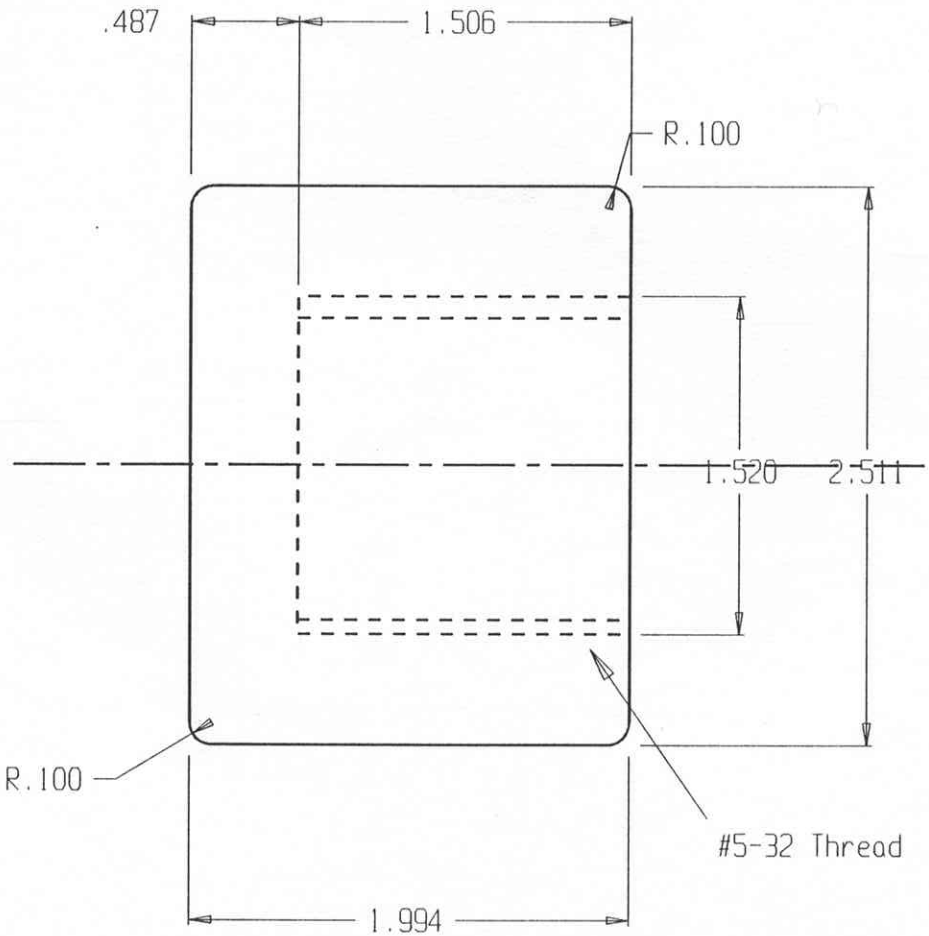
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

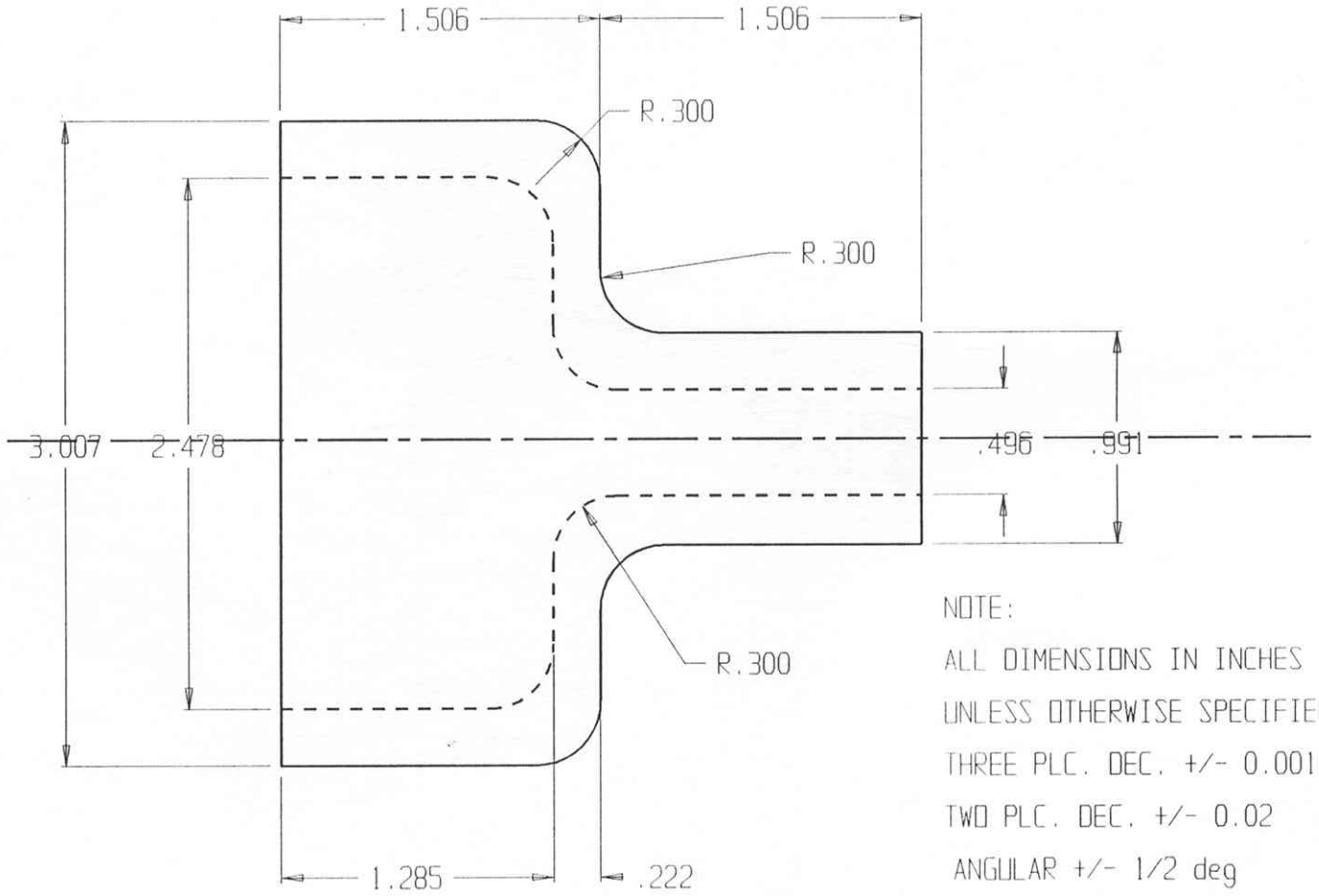
ANGULAR +/- 1/2 deg

PART 31



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 32



NOTE:

ALL DIMENSIONS IN INCHES

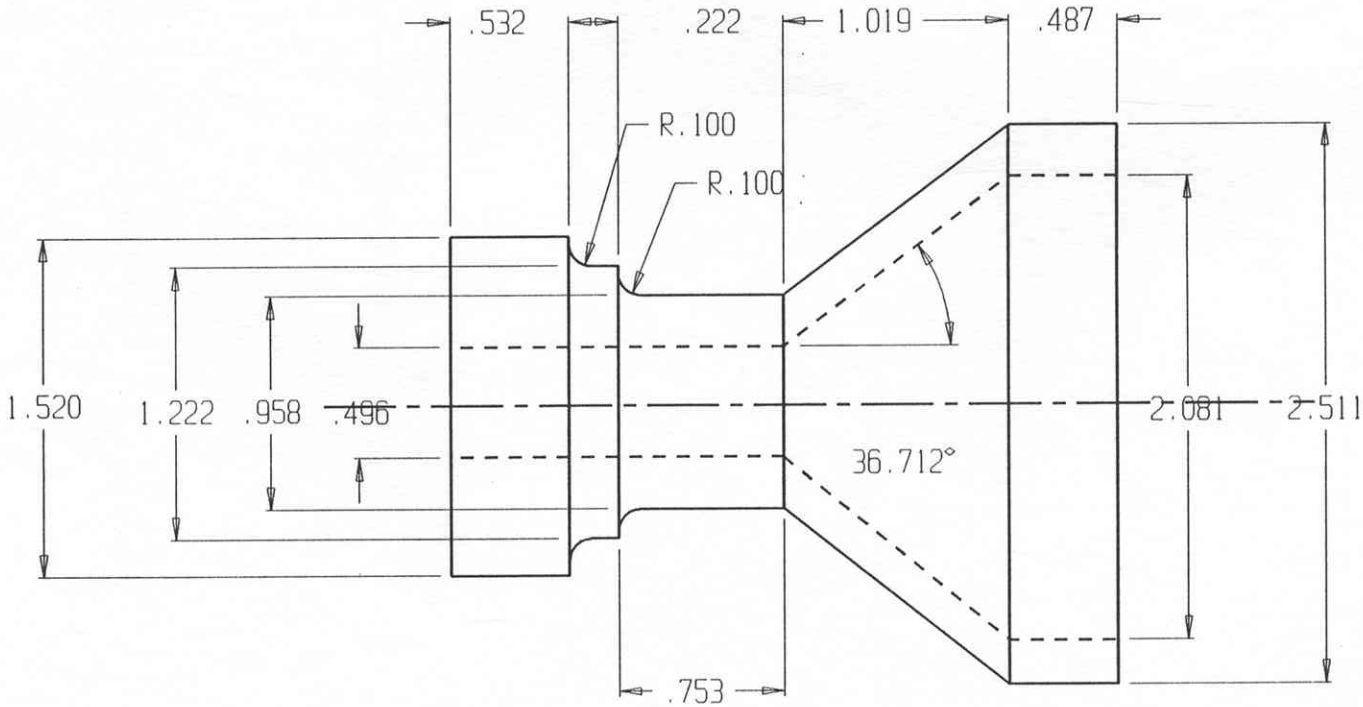
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

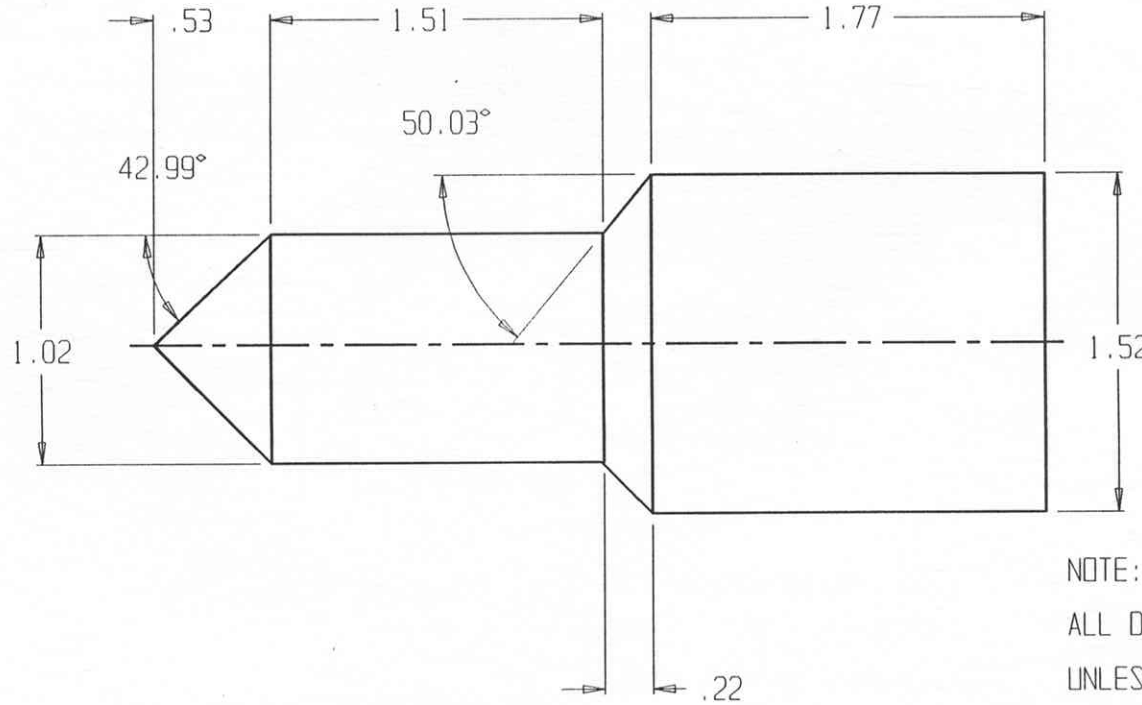
ANGULAR +/- 1/2 deg

PART 33



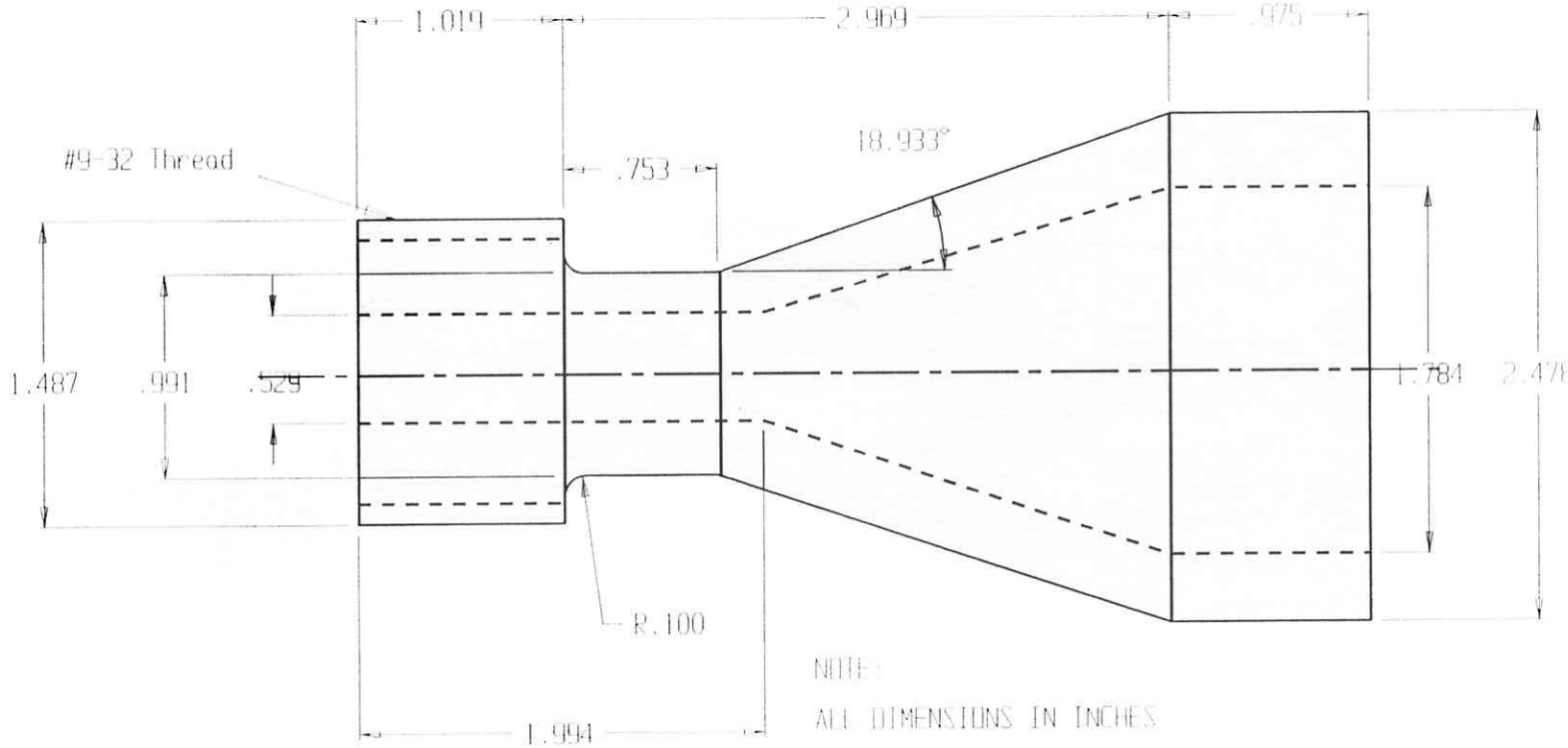
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 34



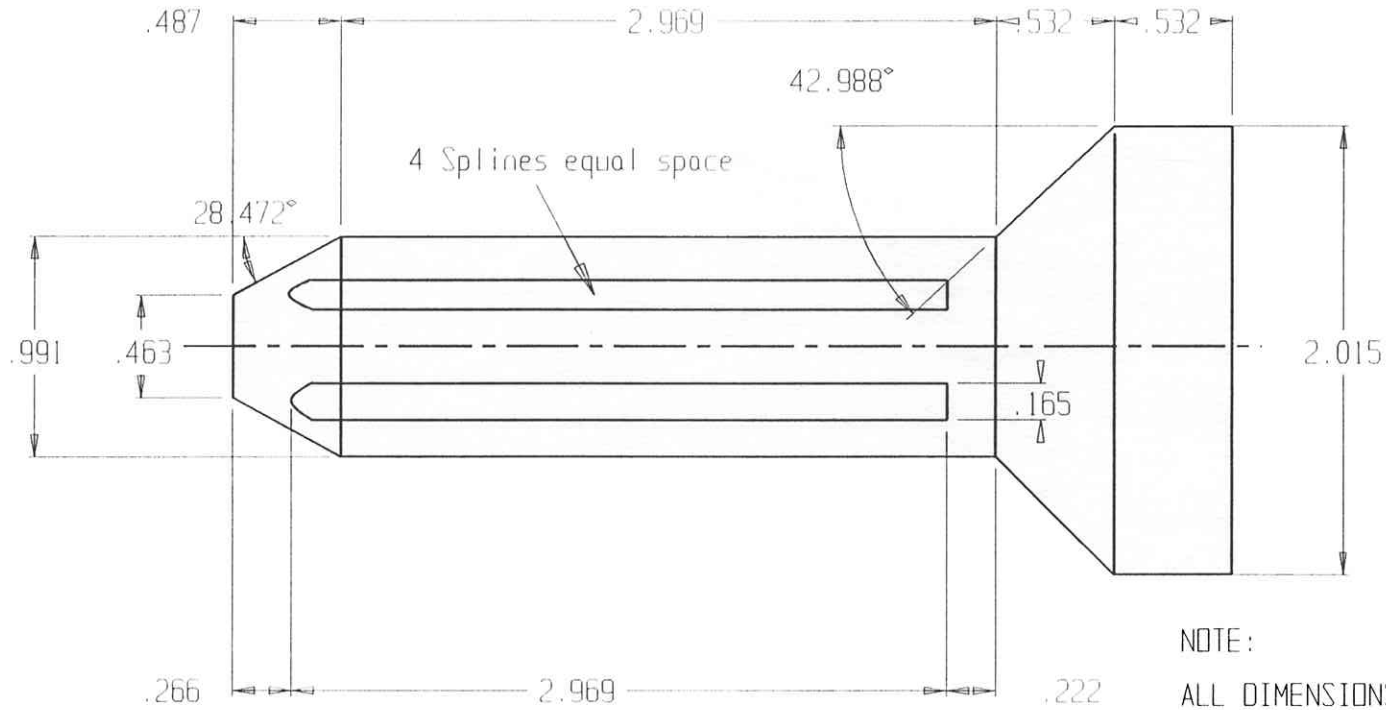
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 35



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 36



NOTE:

ALL DIMENSIONS IN INCHES

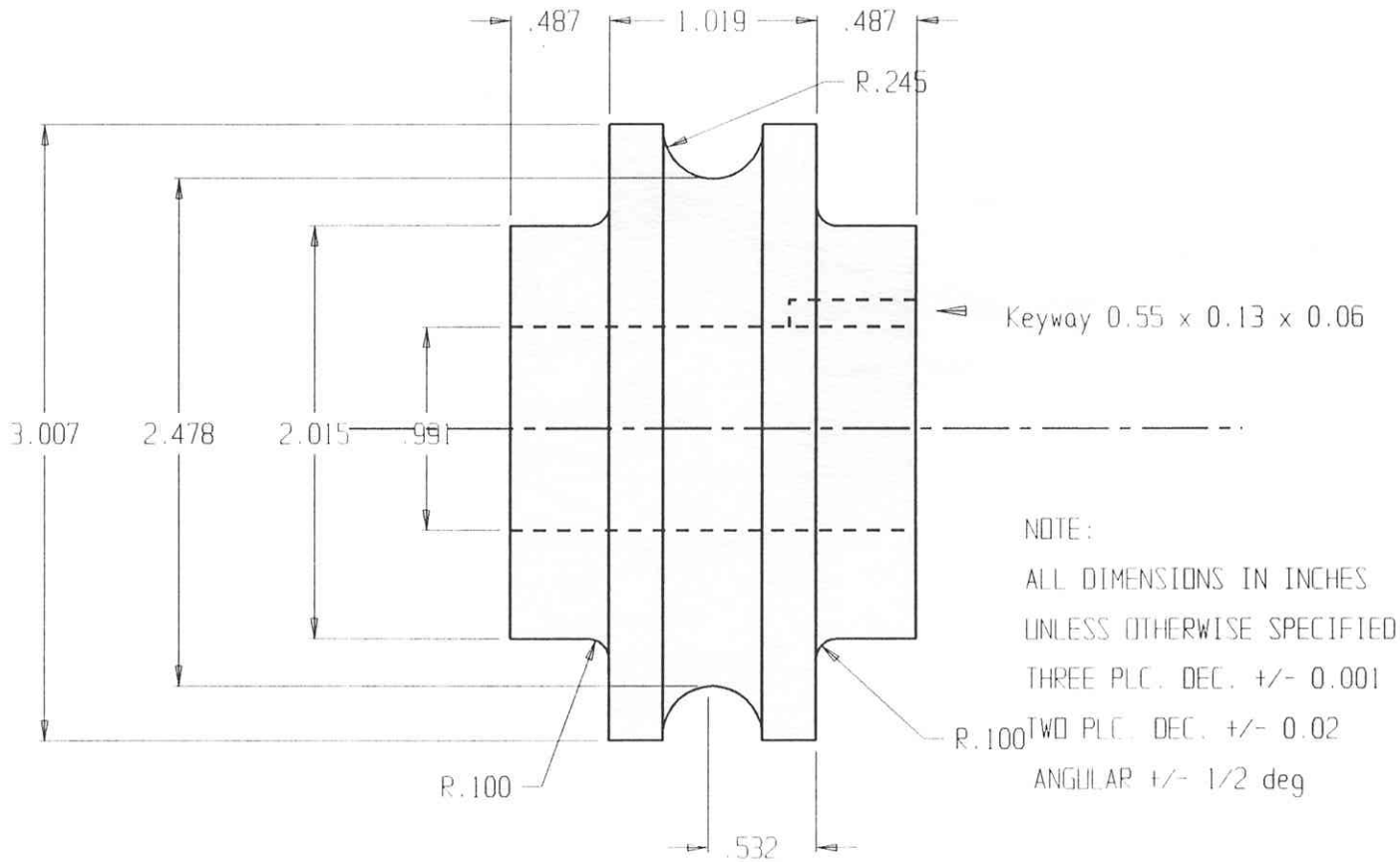
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

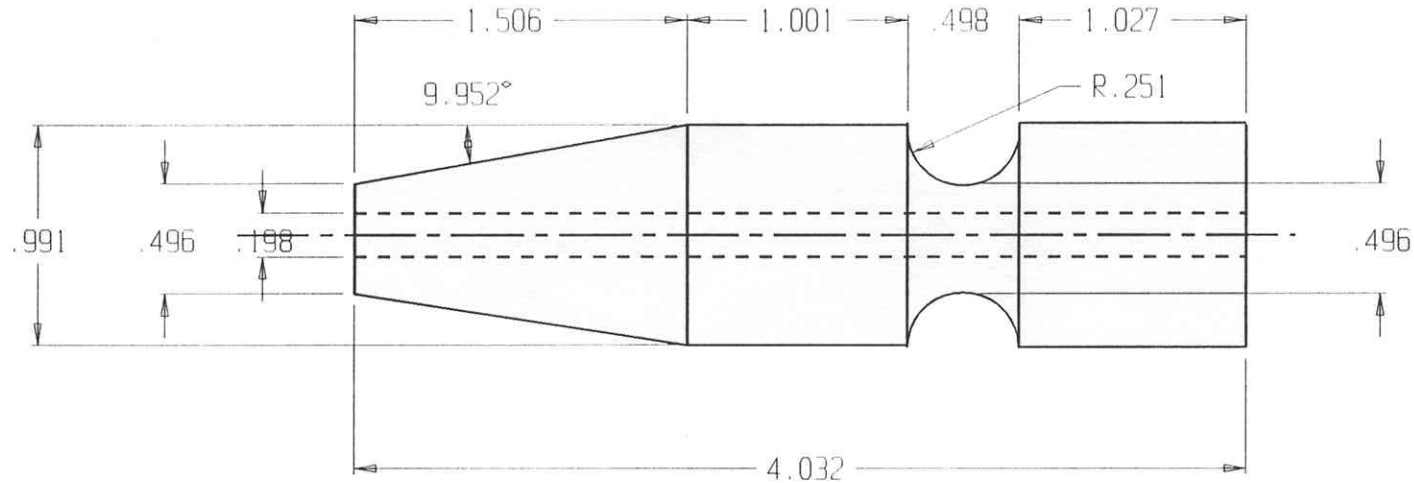
TWO PLC. DEC. +/- 0.02

ANGULAR +/- 1/2 deg

PART 37


.487
1.019
.487
R.245
Keyway 0.55 x 0.13 x 0.06
3.007
2.478
2.015
.991
R.100
R.100
.532
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 38



NOTE:

ALL DIMENSIONS IN INCHES

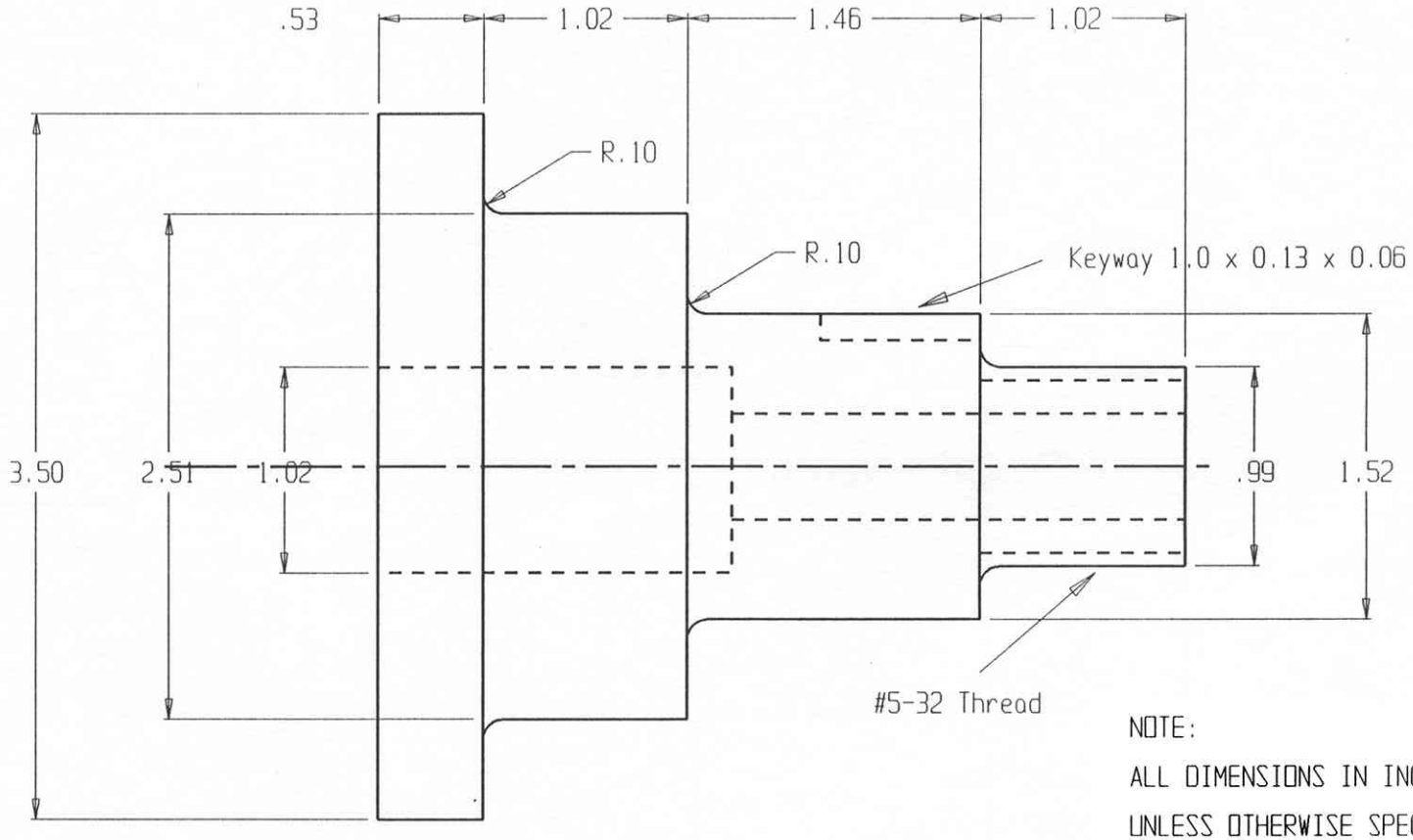
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

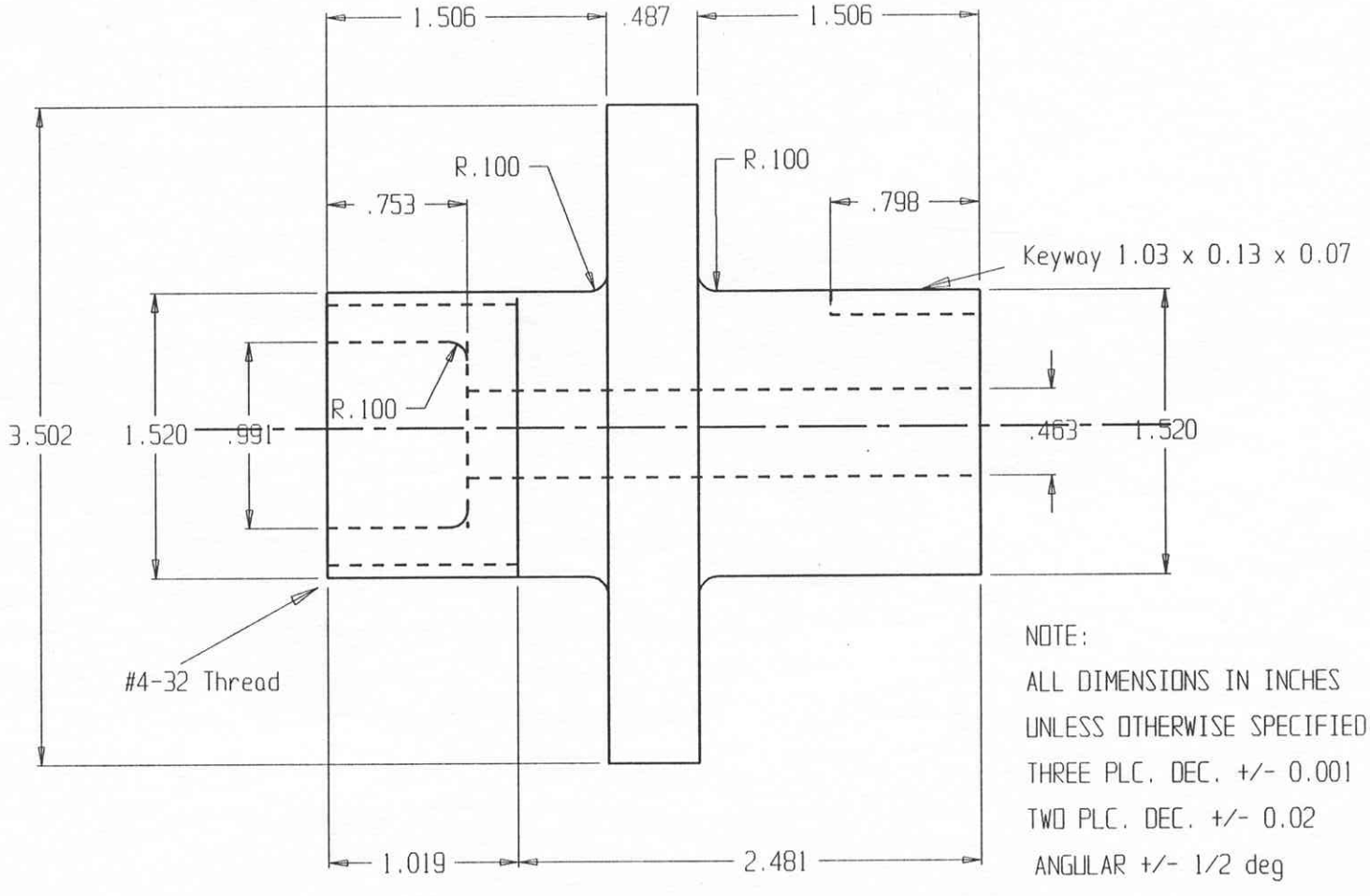
ANGULAR +/- 1/2 deg

PART 39



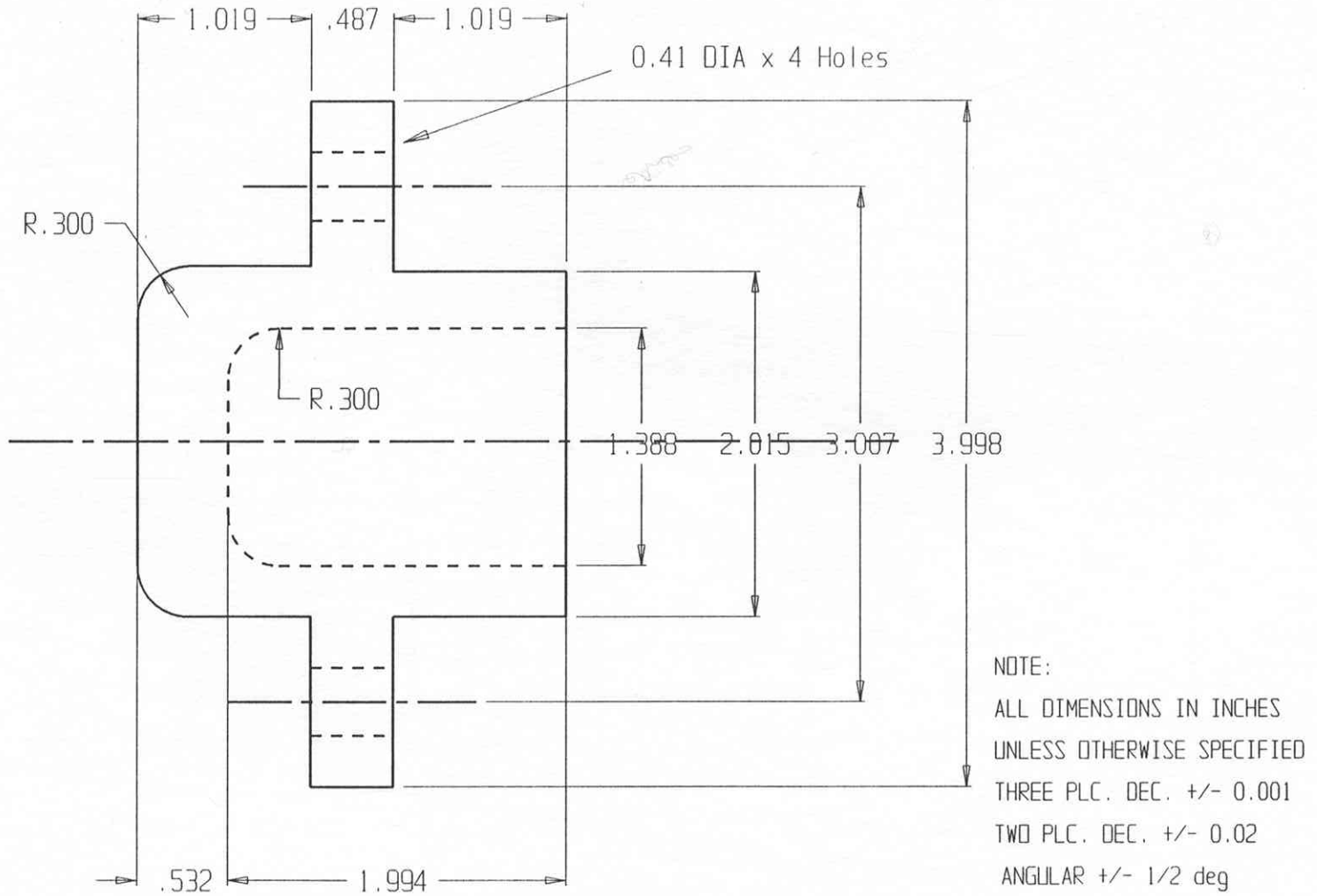
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 40



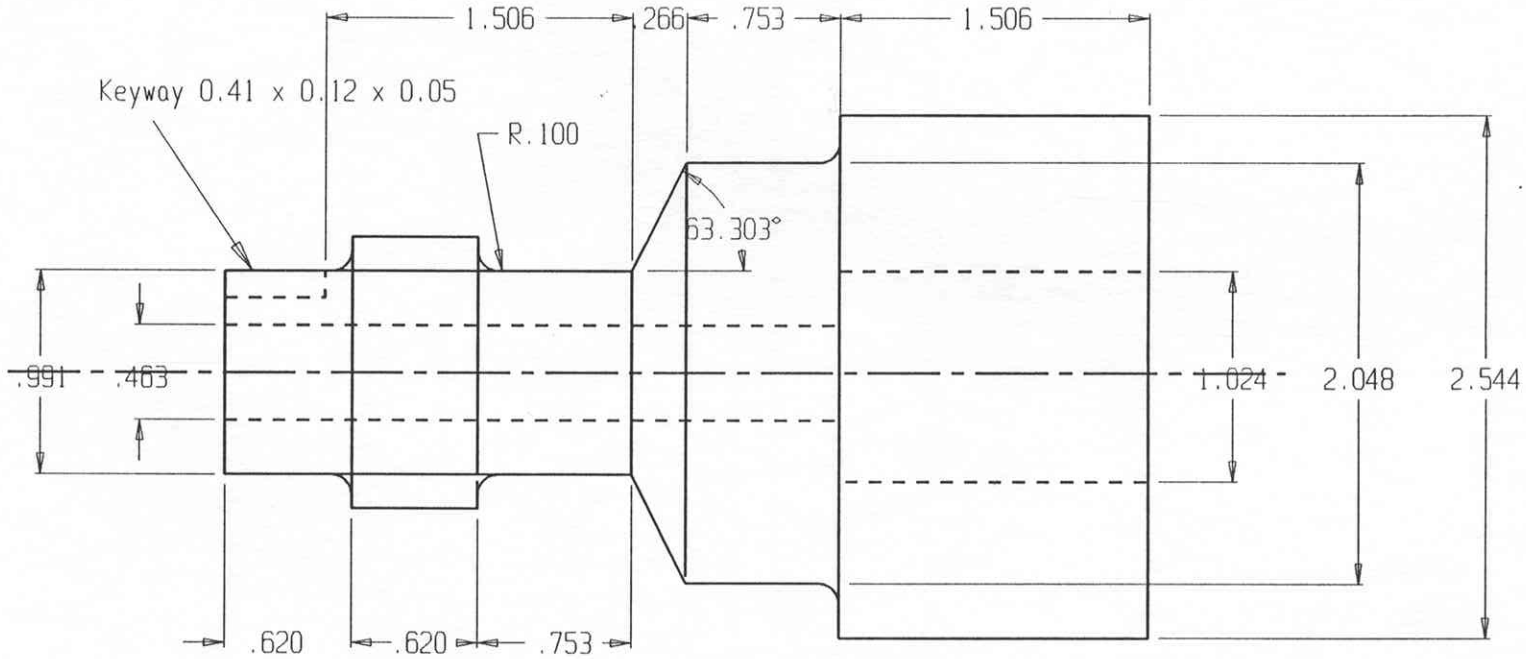
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 41



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 42



NOTE:

ALL DIMENSIONS IN INCHES

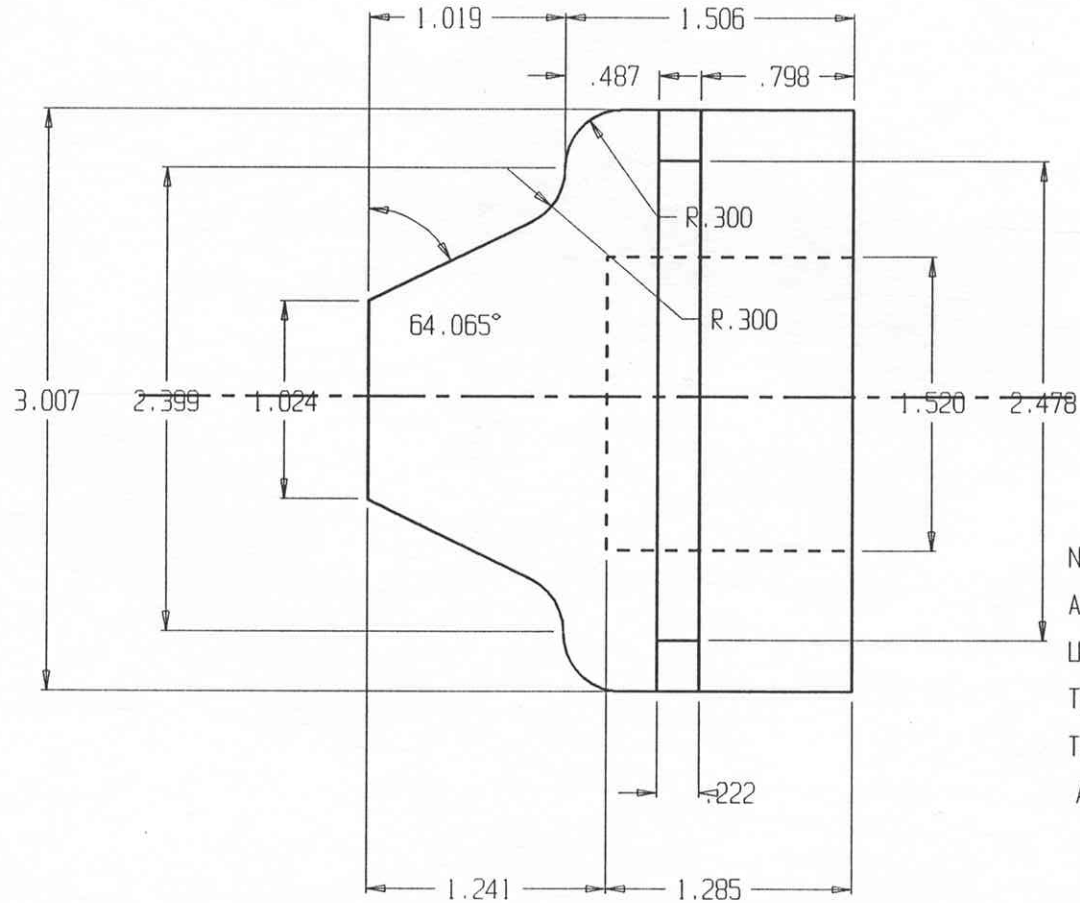
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

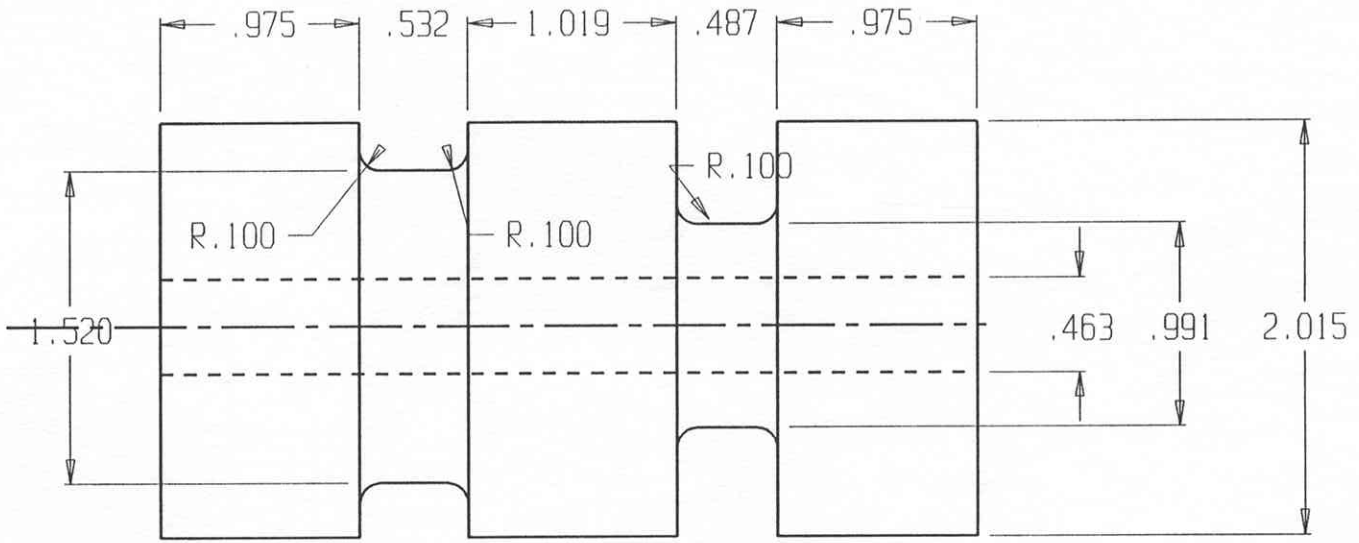
ANGULAR +/- 1/2 deg

PART 43



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 44



NOTE:

ALL DIMENSIONS IN INCHES

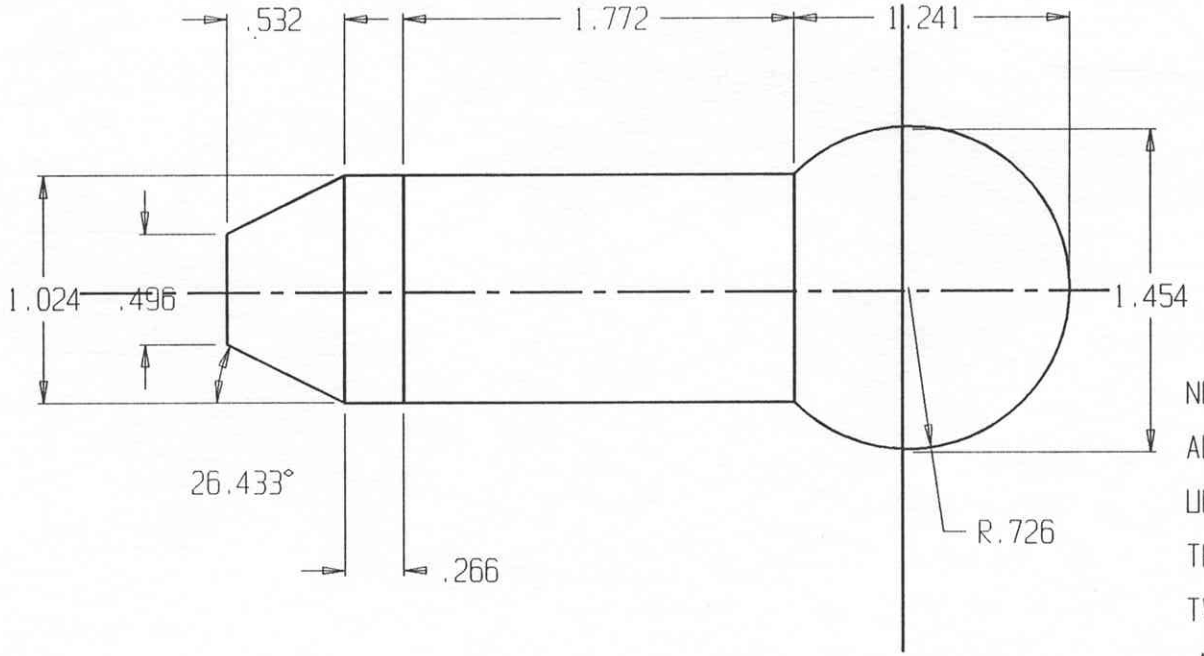
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

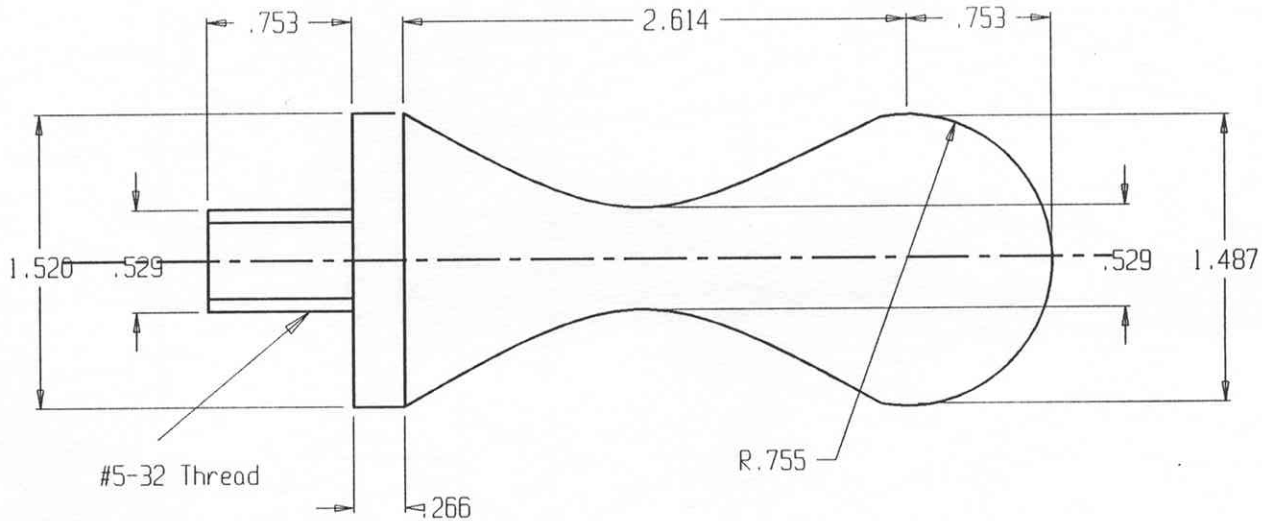
ANGULAR +/- 1/2 deg

PART 45



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 46



NOTE:

ALL DIMENSIONS IN INCHES

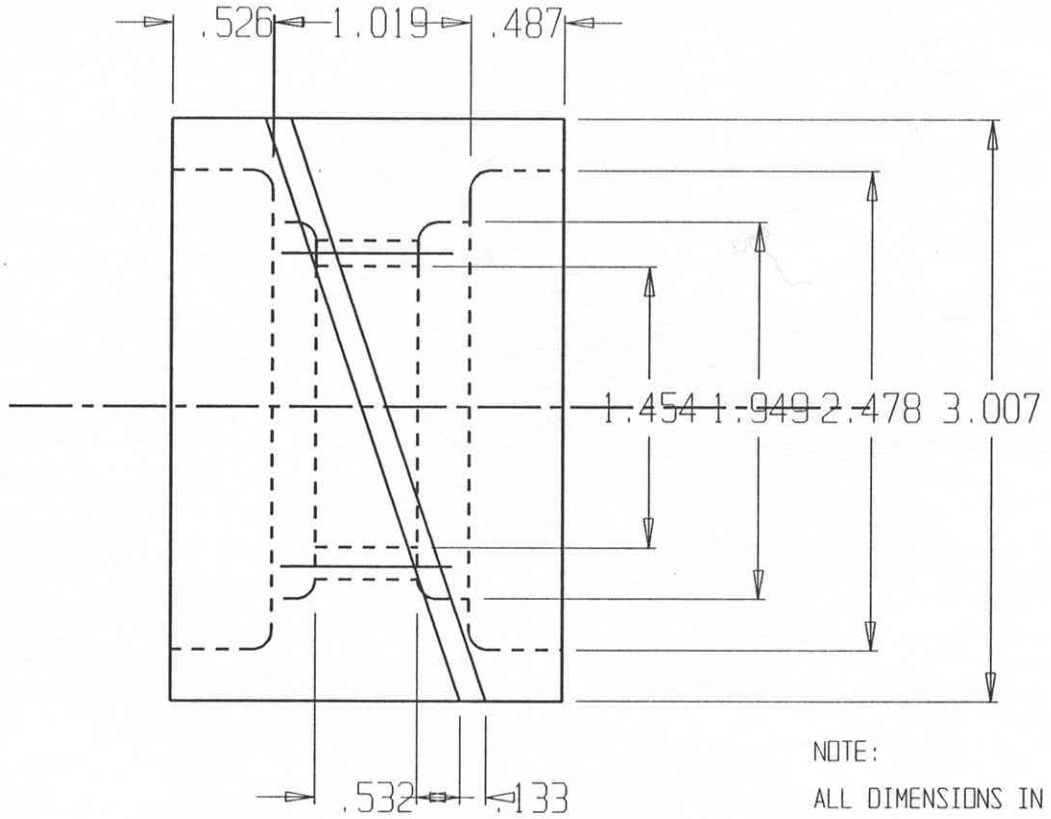
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

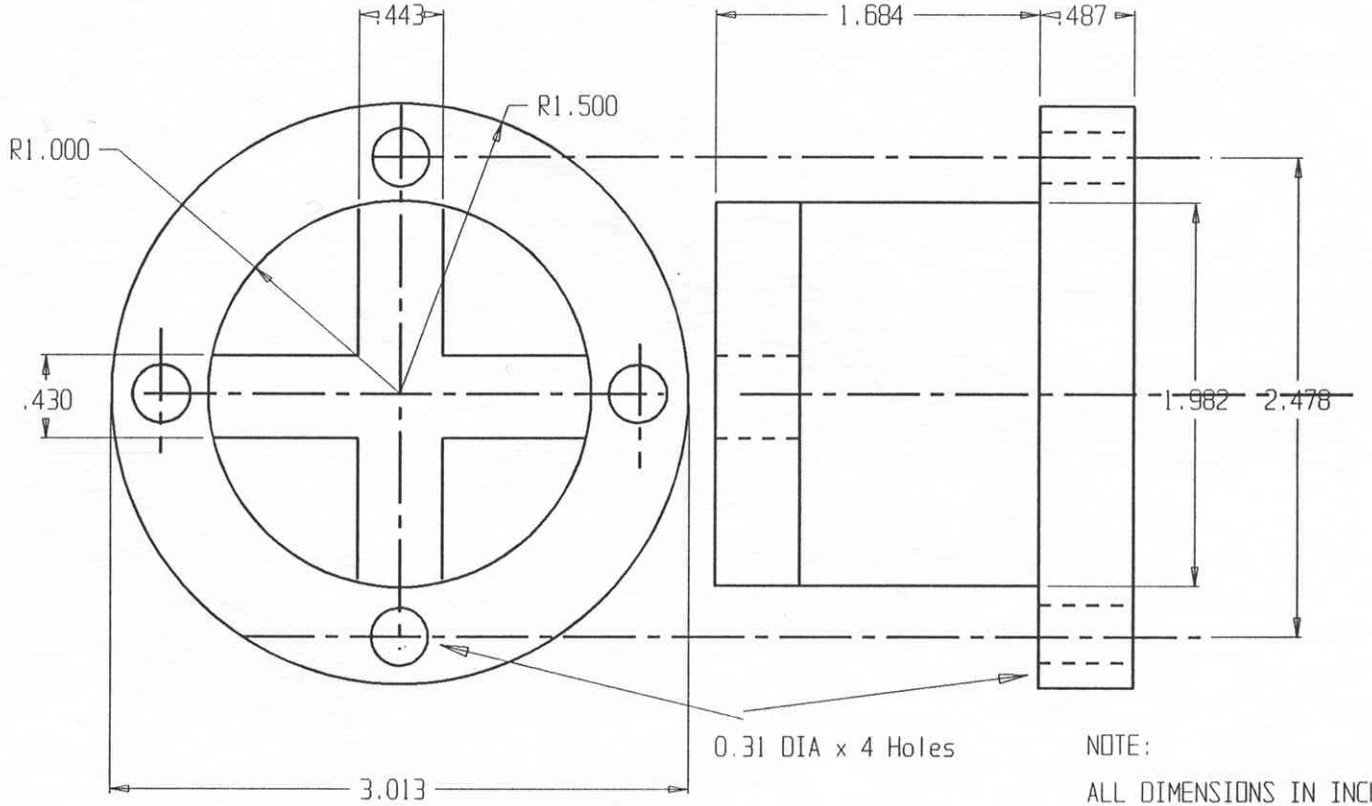
ANGULAR +/- 1/2 deg

PART 47



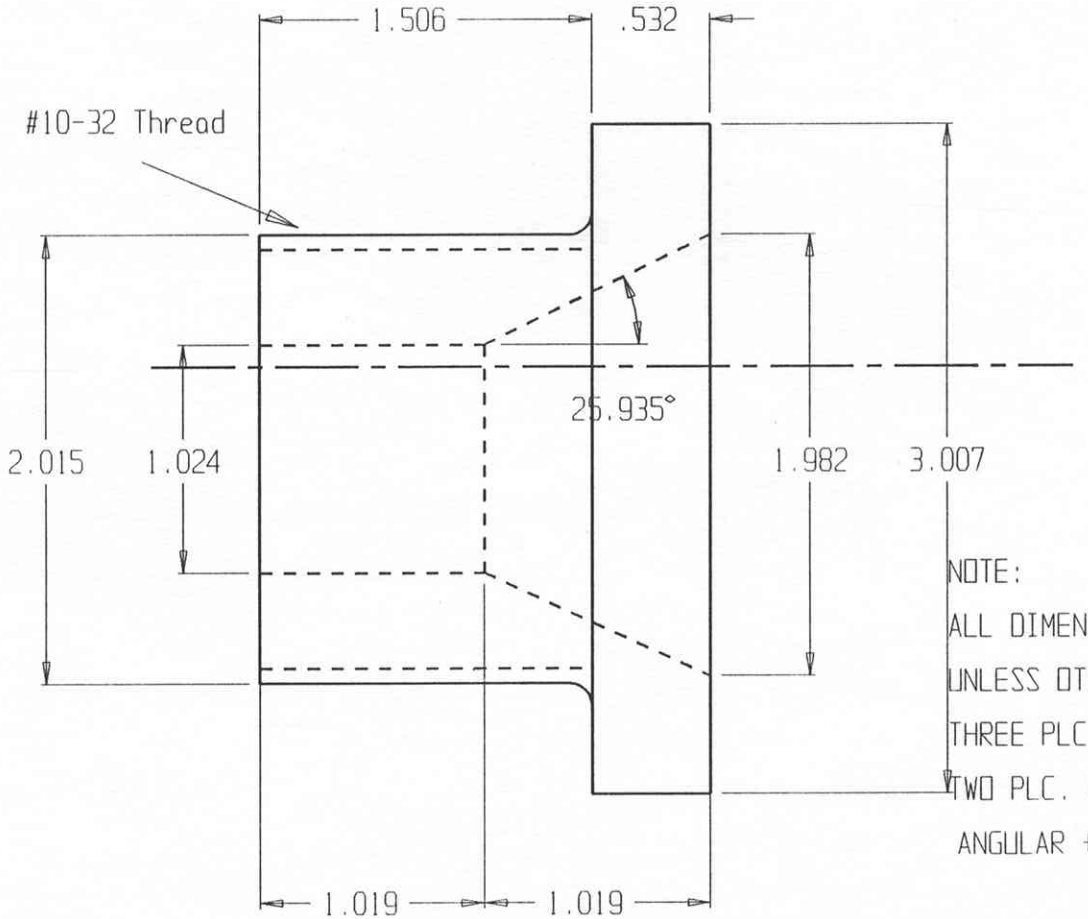
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 48



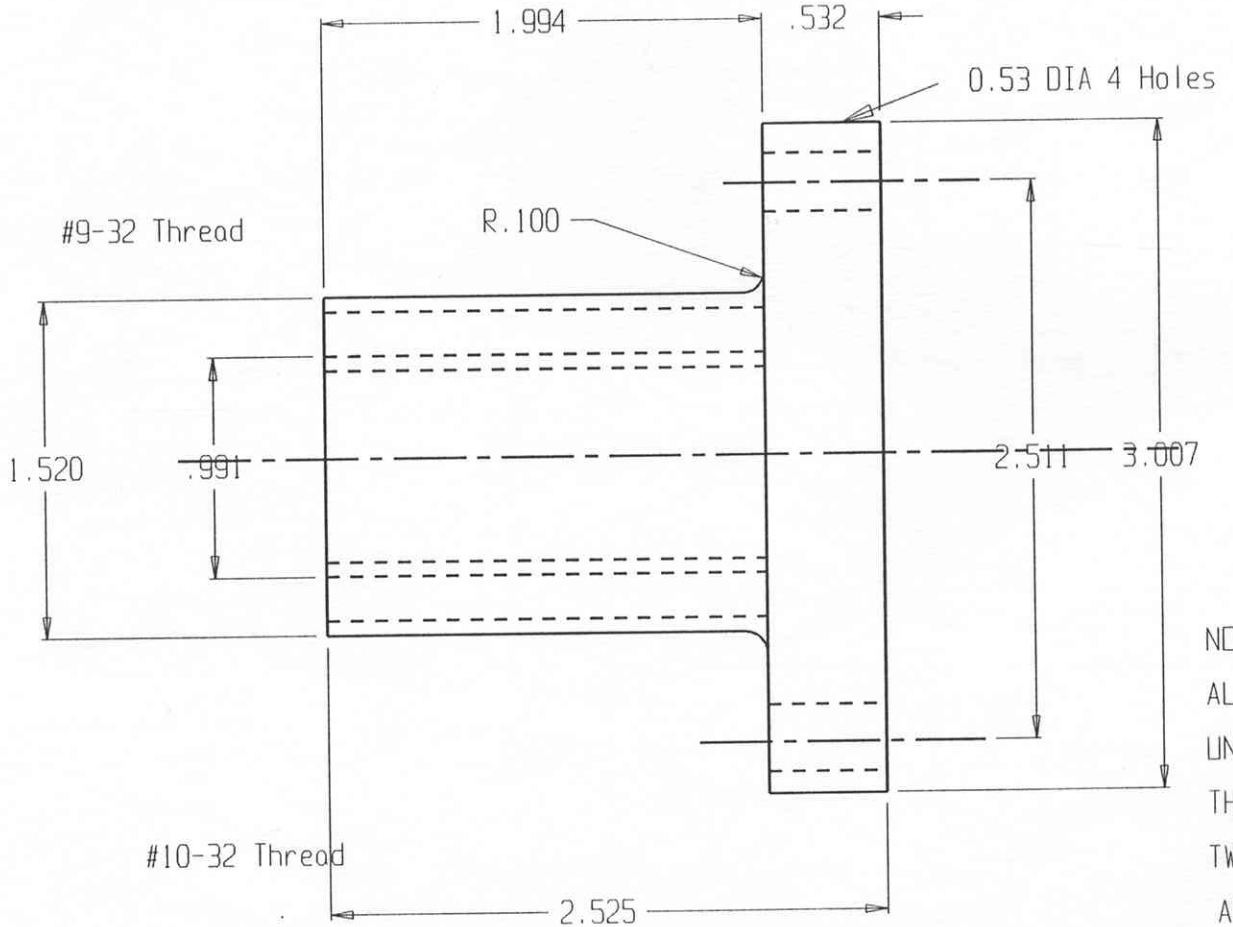
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 49



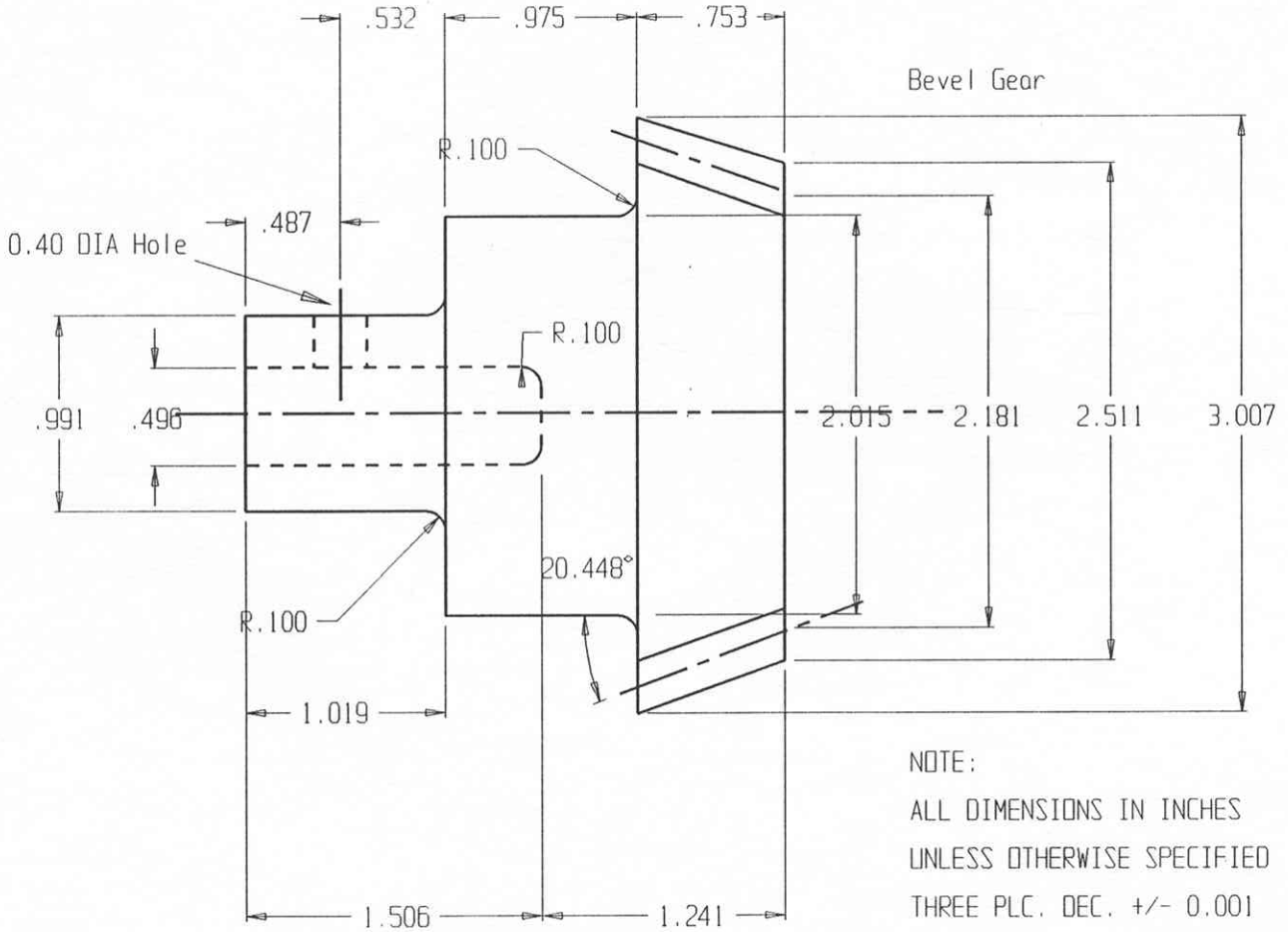
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 50



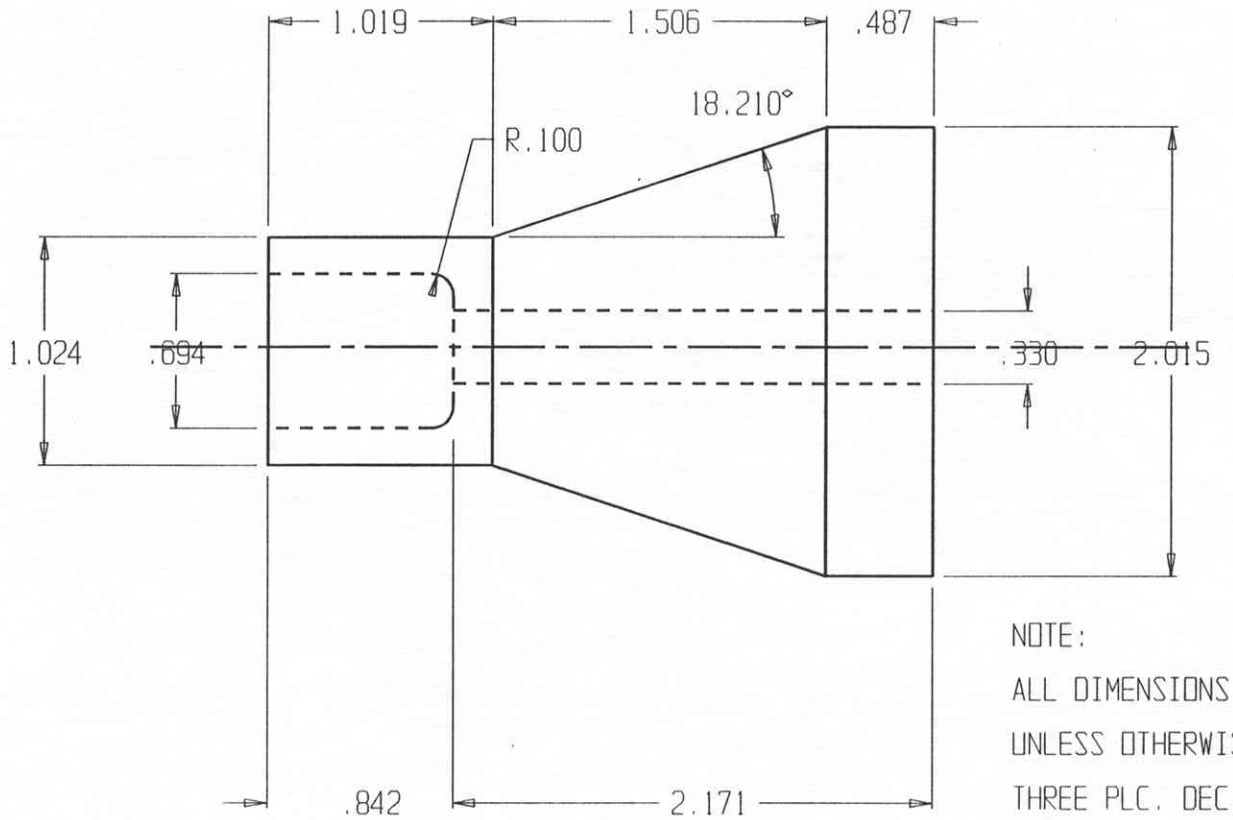
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 51



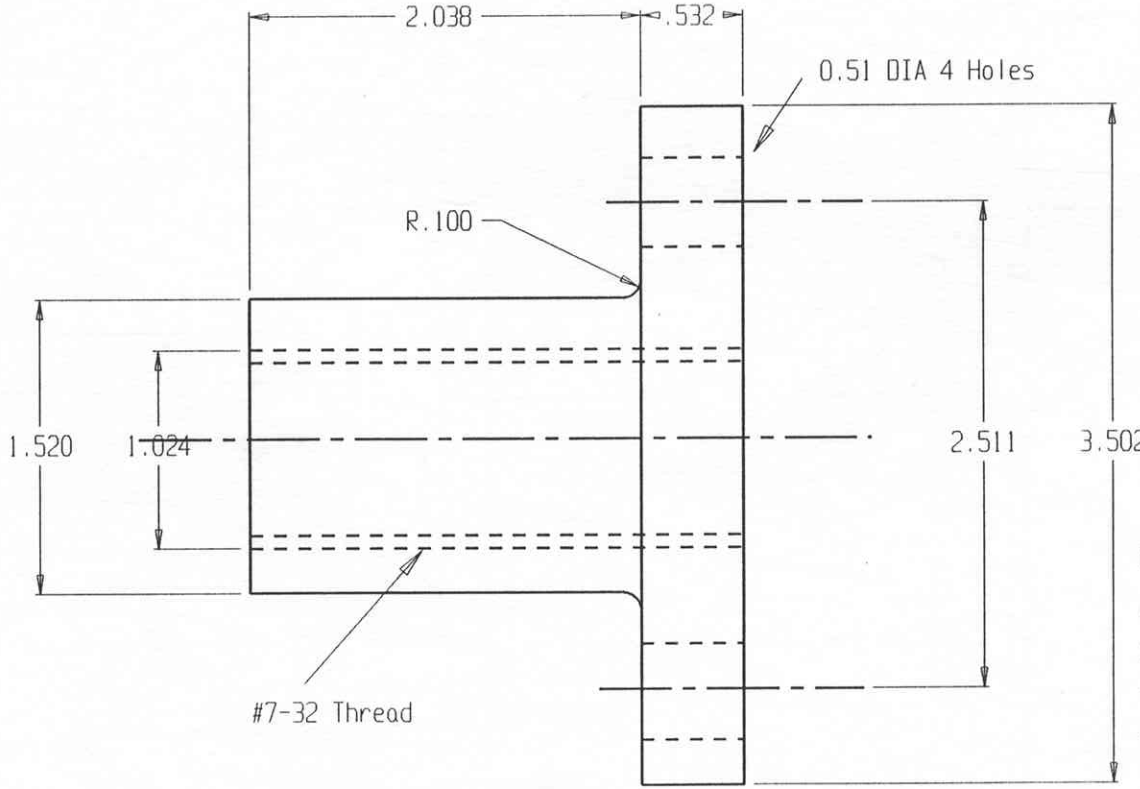
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 52



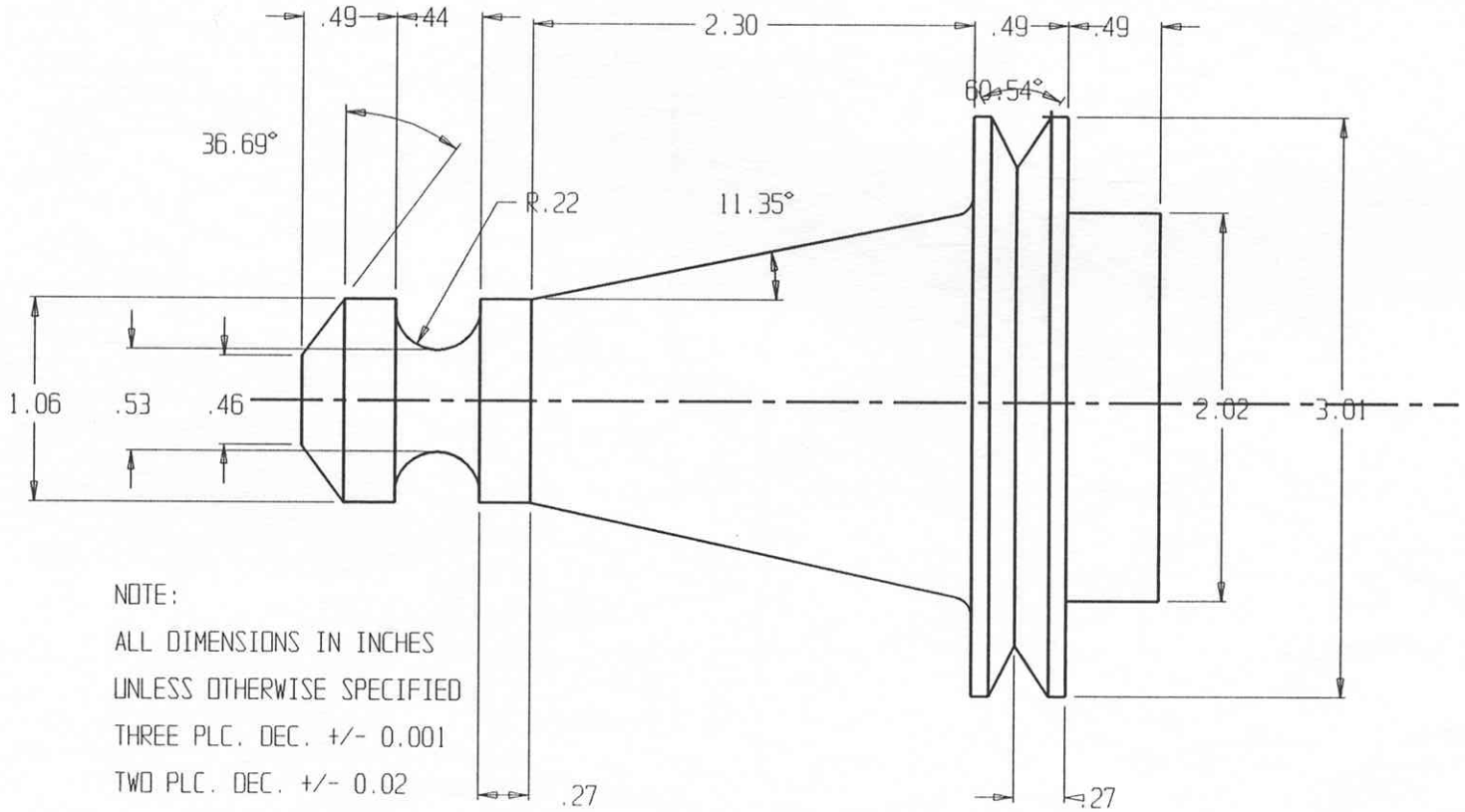
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 53



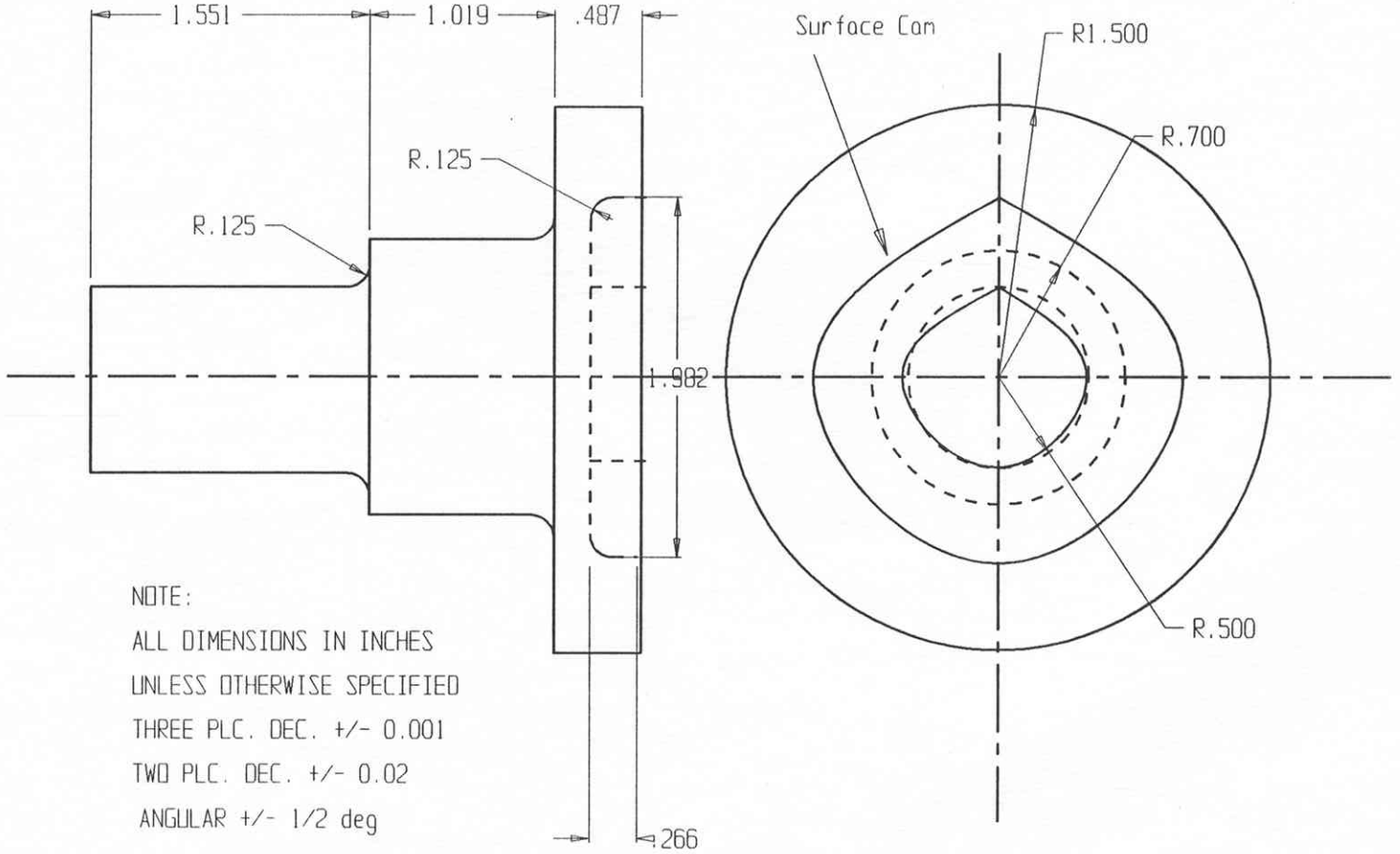
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 54



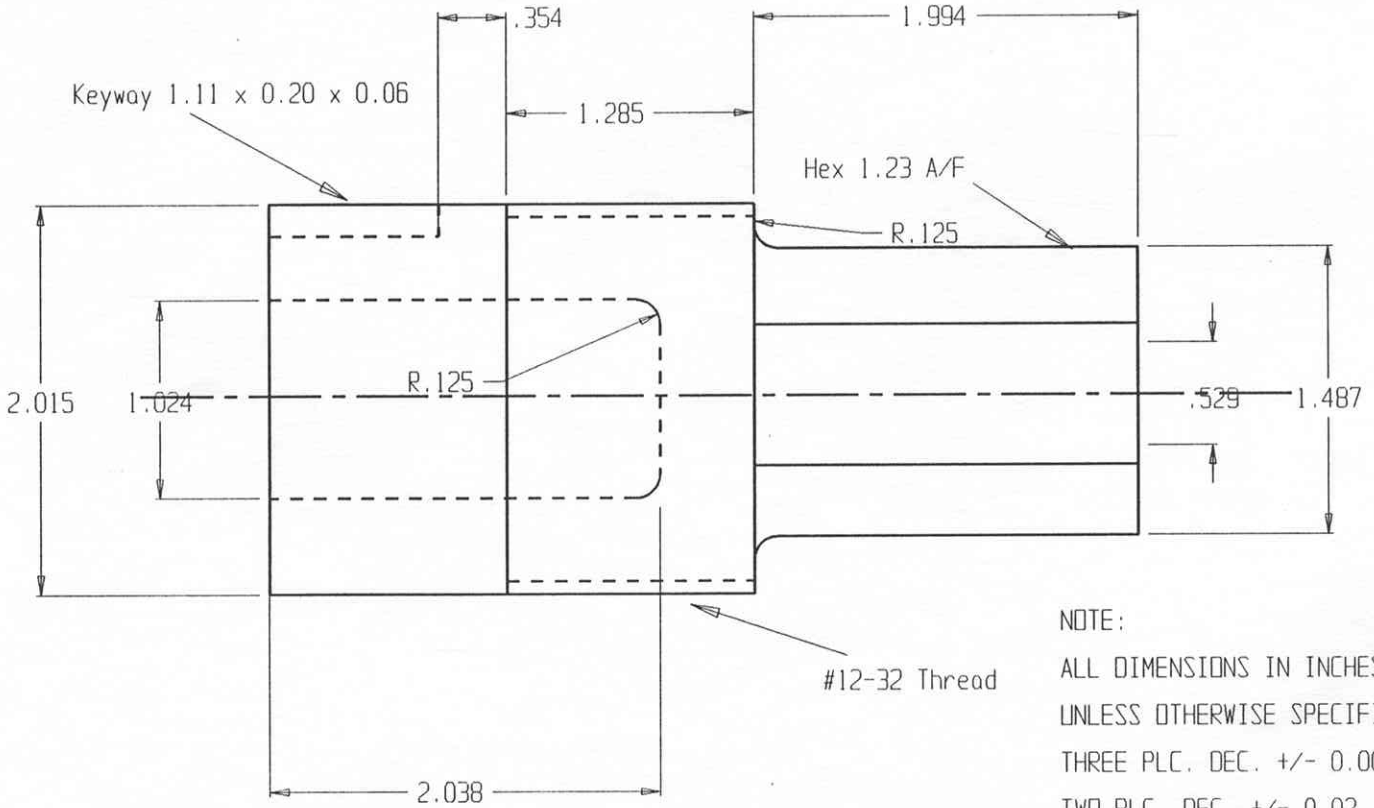
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 55



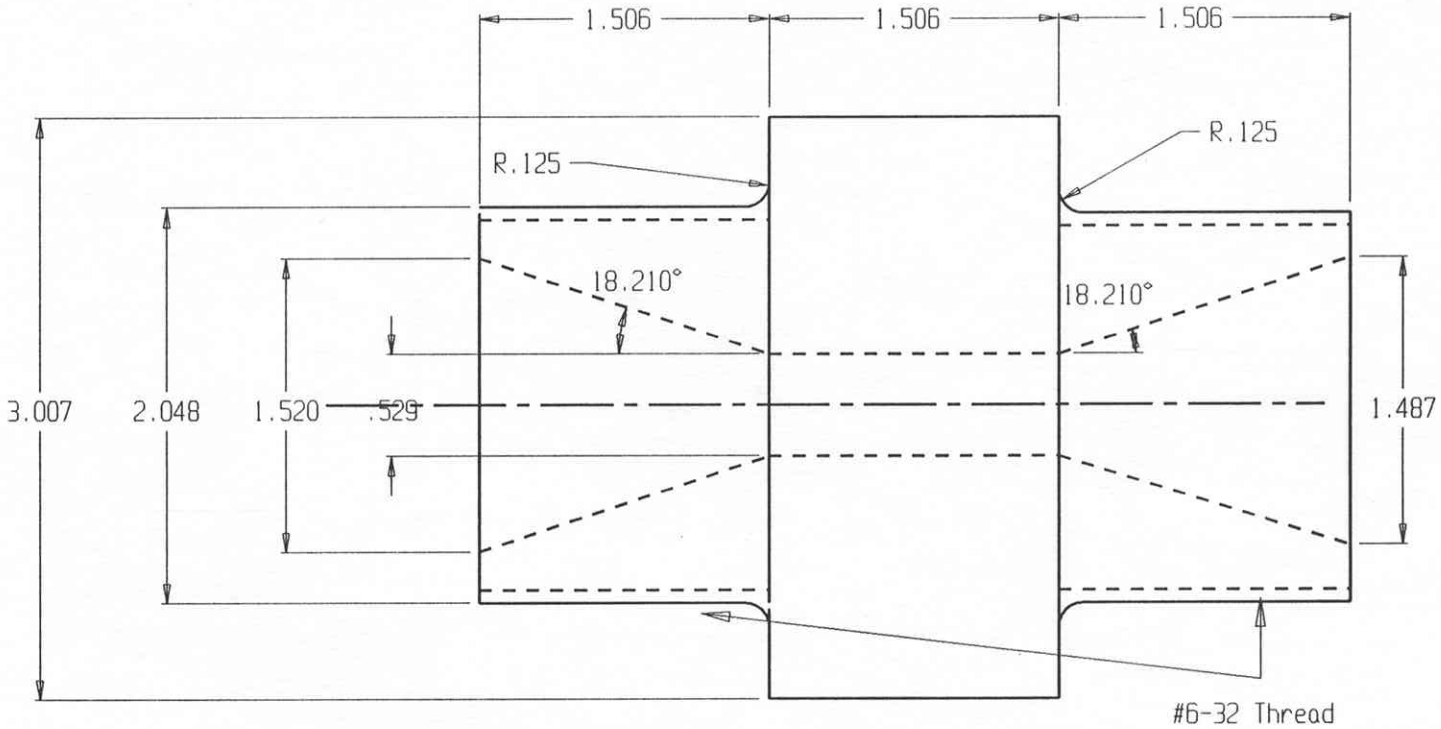
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 56



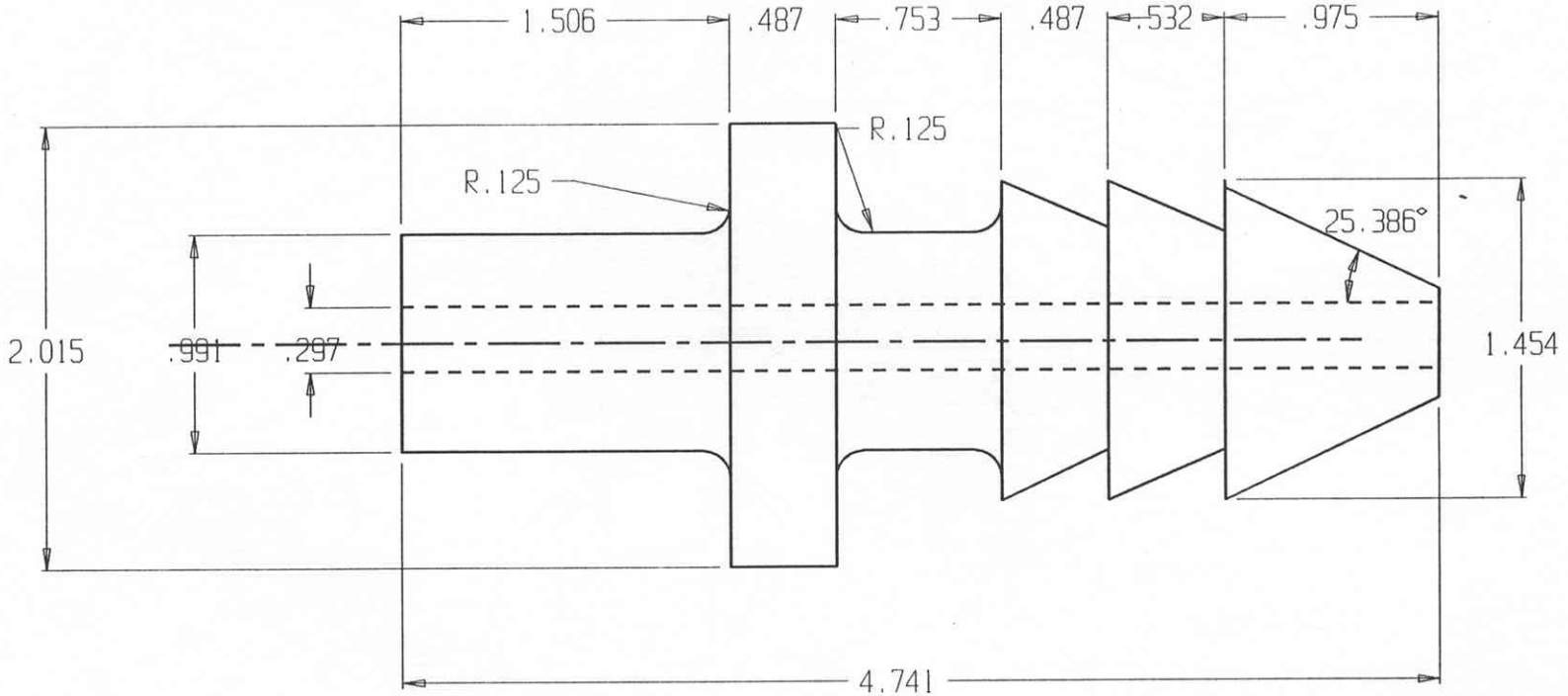
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 57



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 58



NOTE:

ALL DIMENSIONS IN INCHES

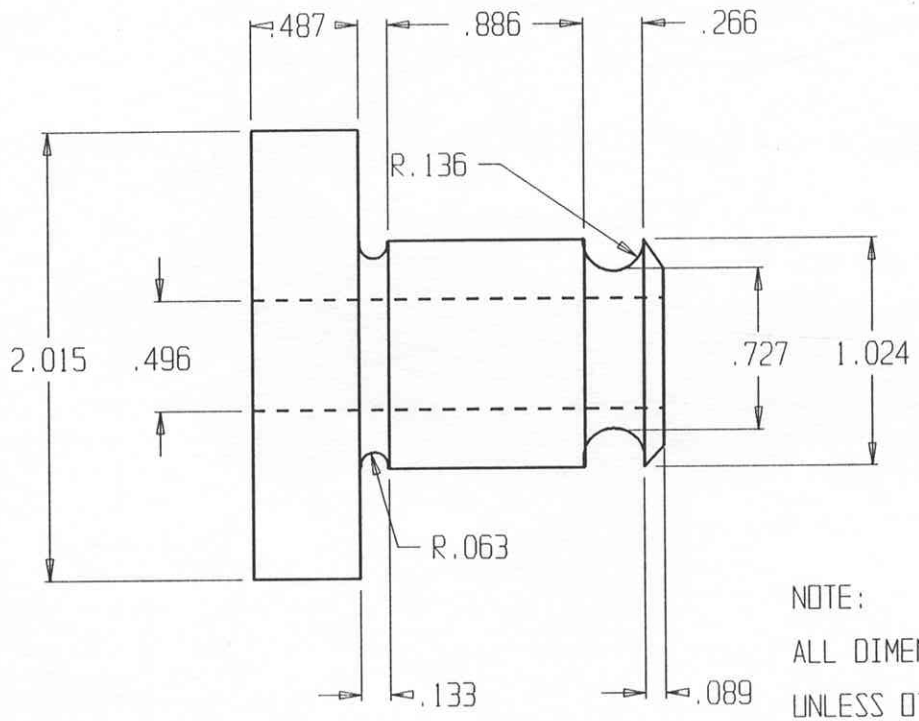
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

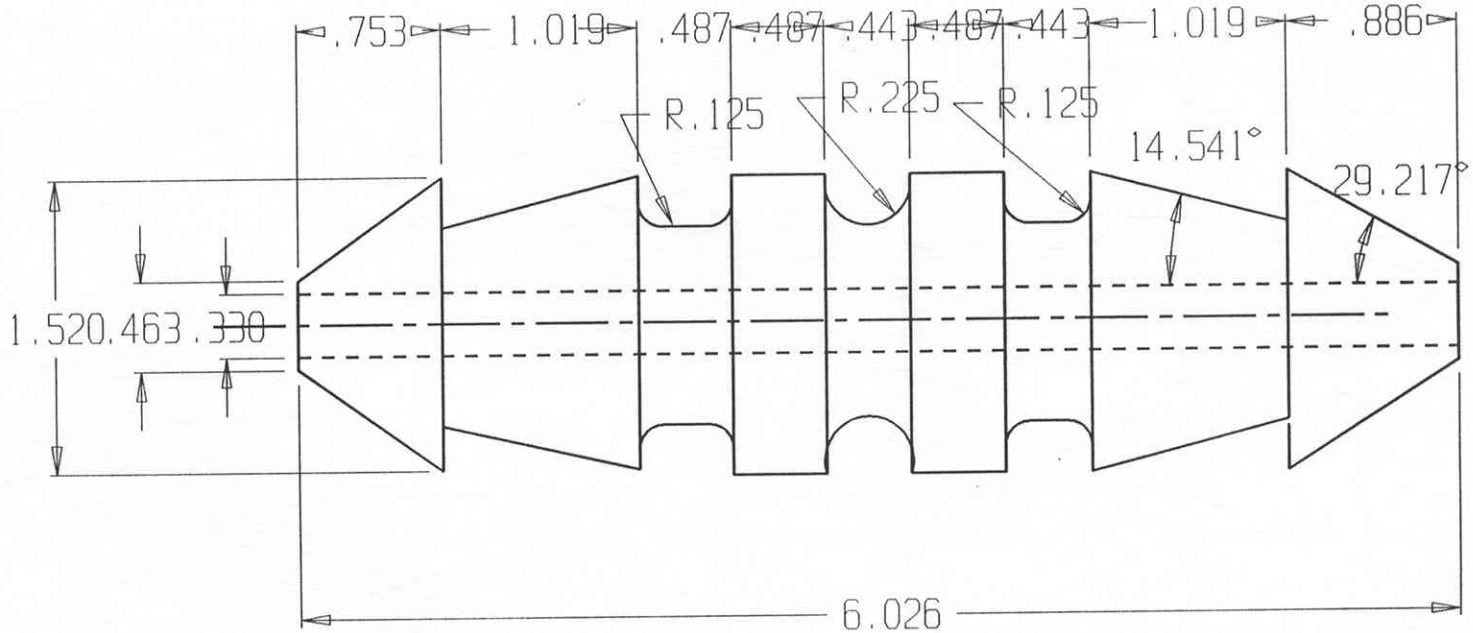
ANGULAR +/- 1/2 deg

PART 59



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 60



NOTE:

ALL DIMENSIONS IN INCHES

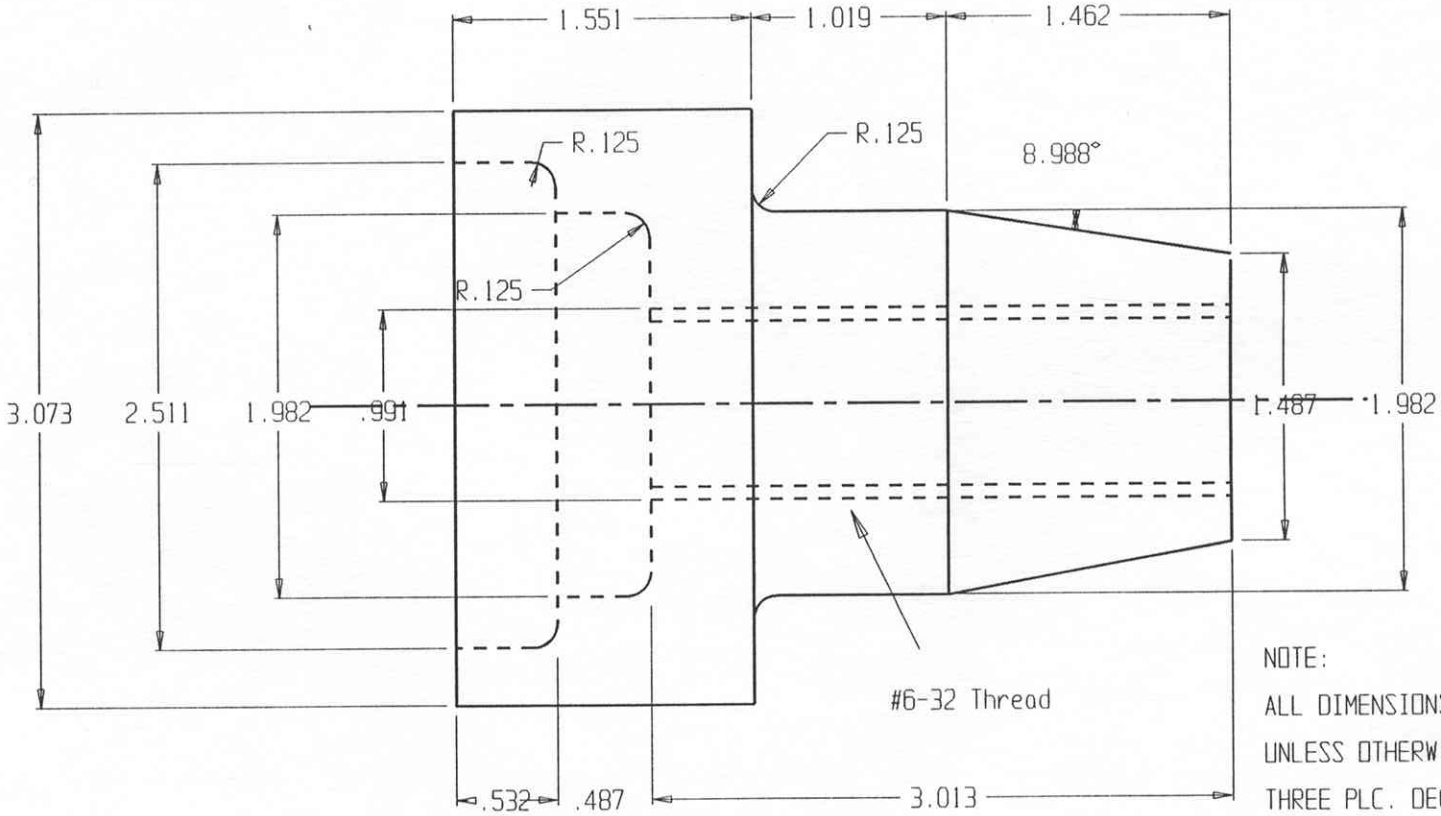
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

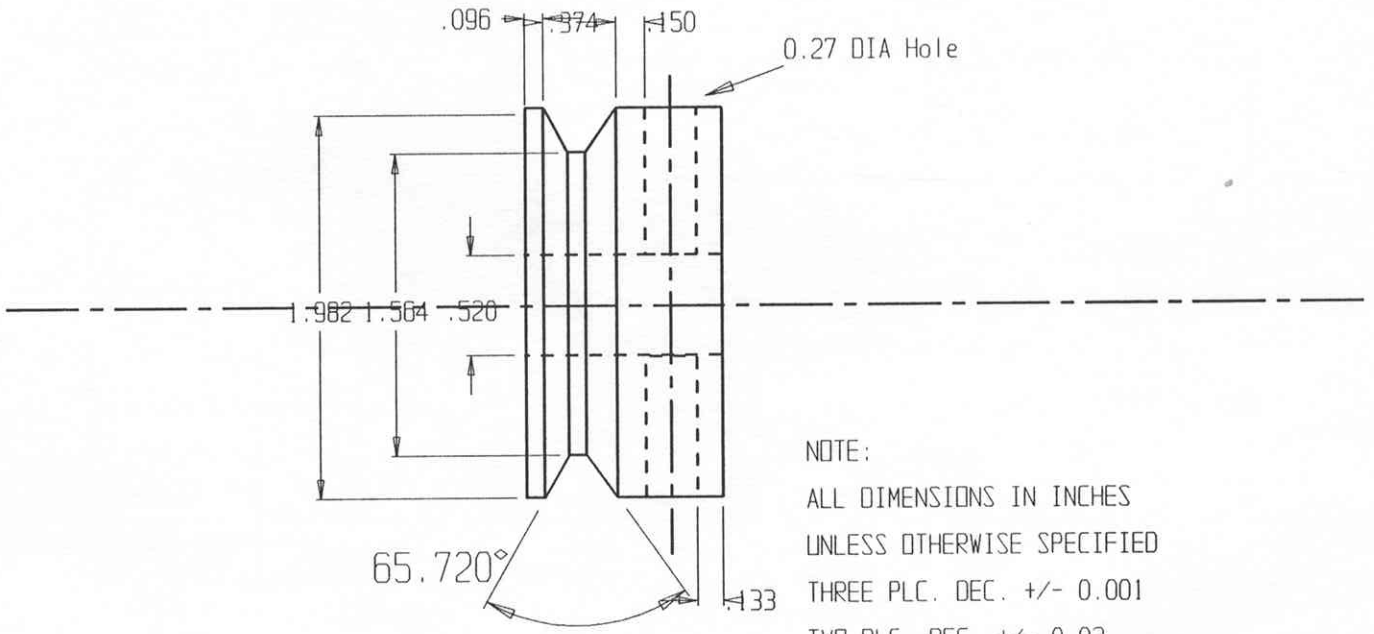
ANGULAR +/- 1/2 deg

PART 61



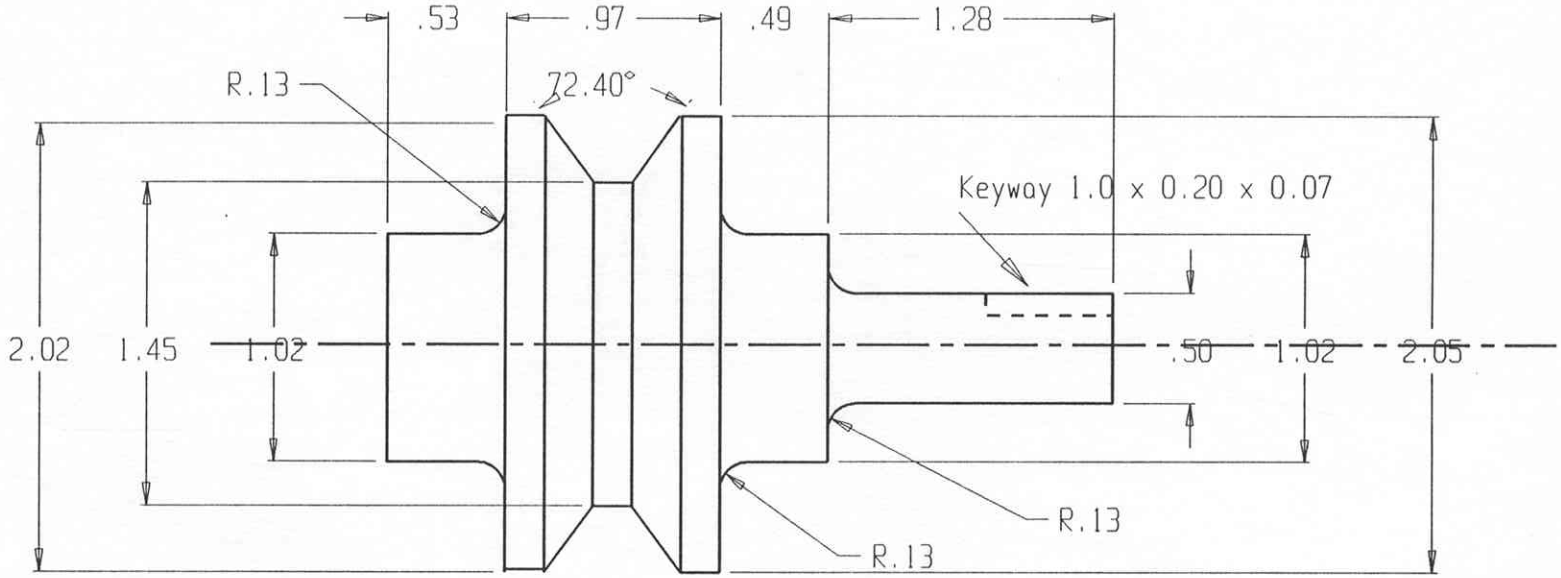
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 62



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 63



NOTE:

ALL DIMENSIONS IN INCHES

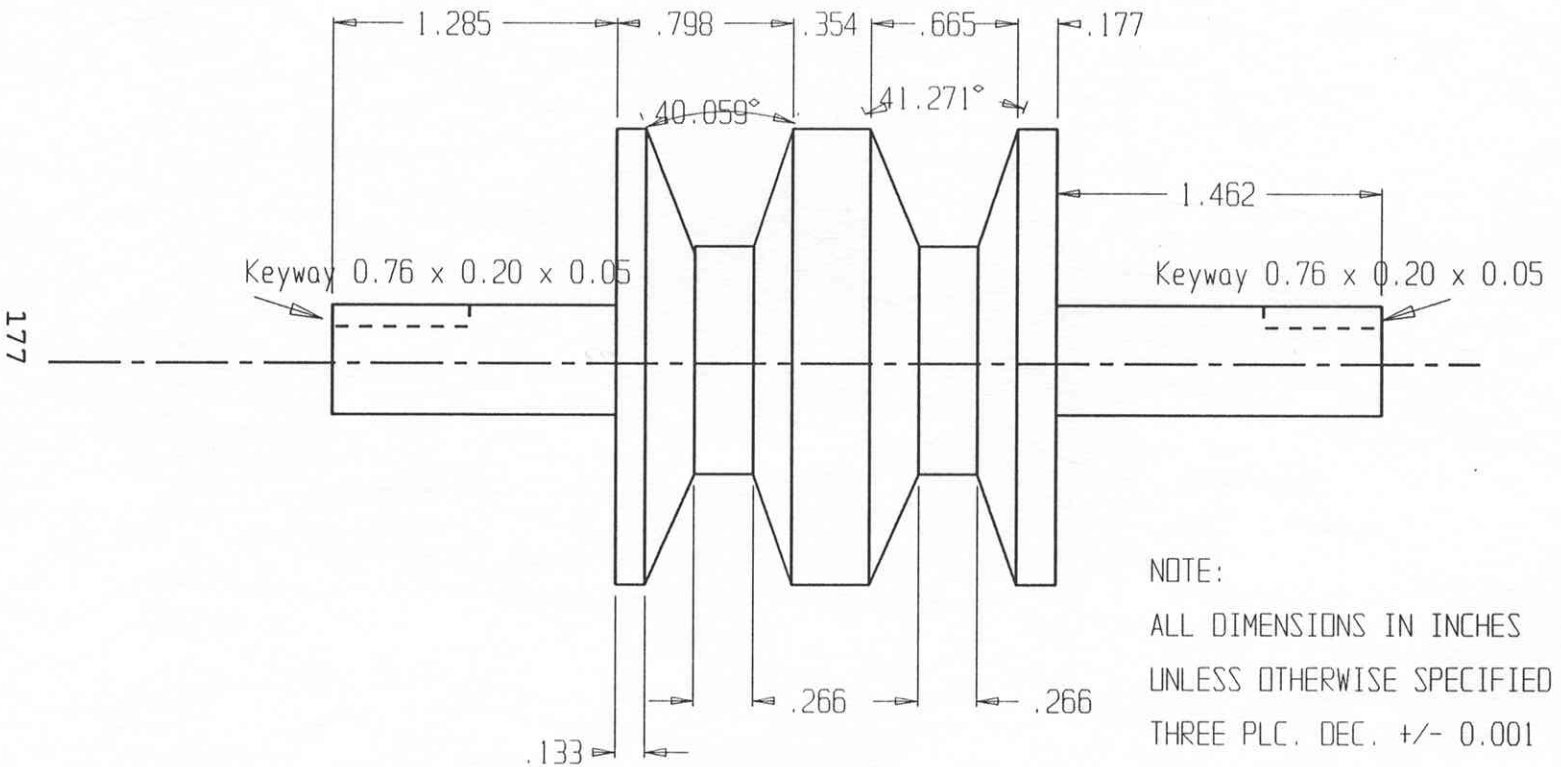
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

ANGULAR +/- 1/2 deg

PART 64



NOTE:

ALL DIMENSIONS IN INCHES

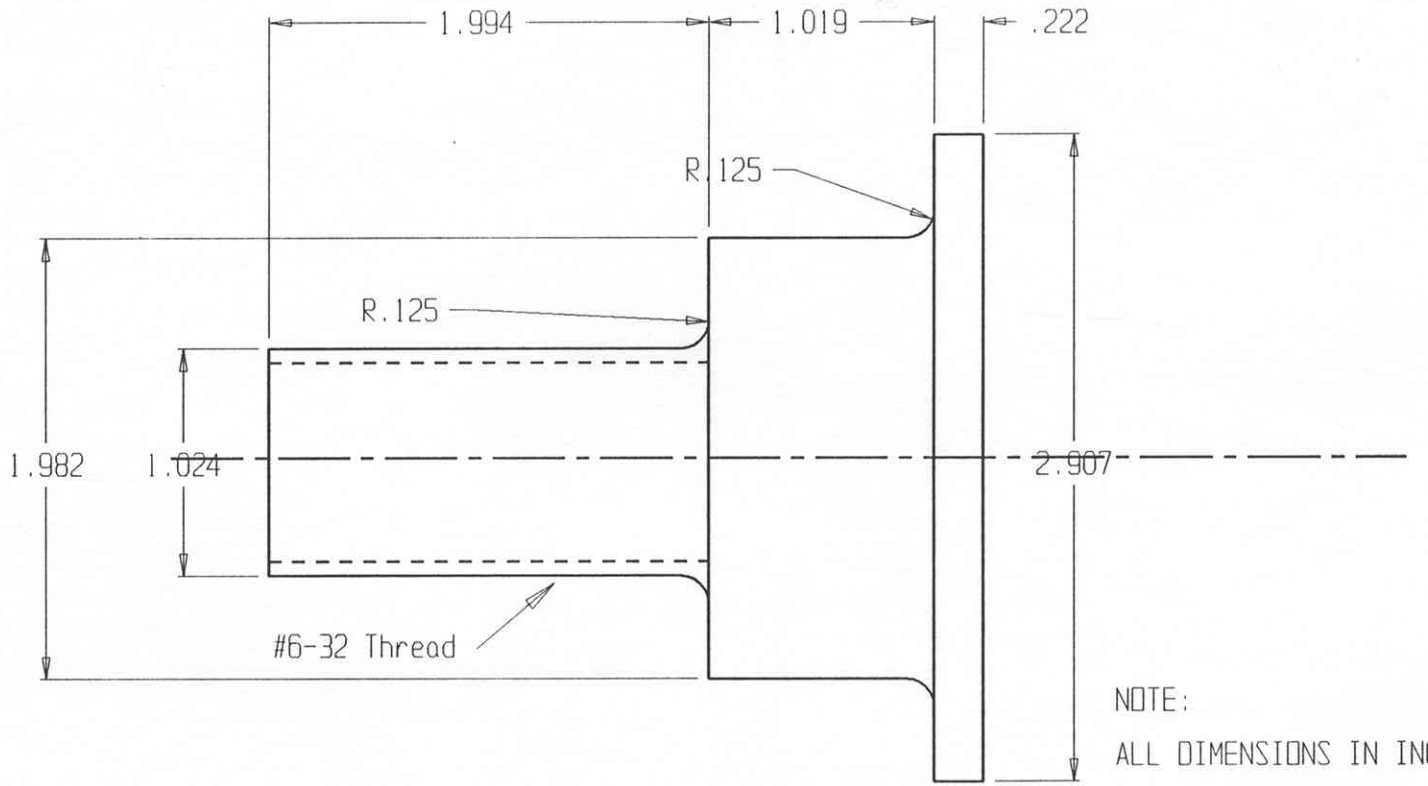
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

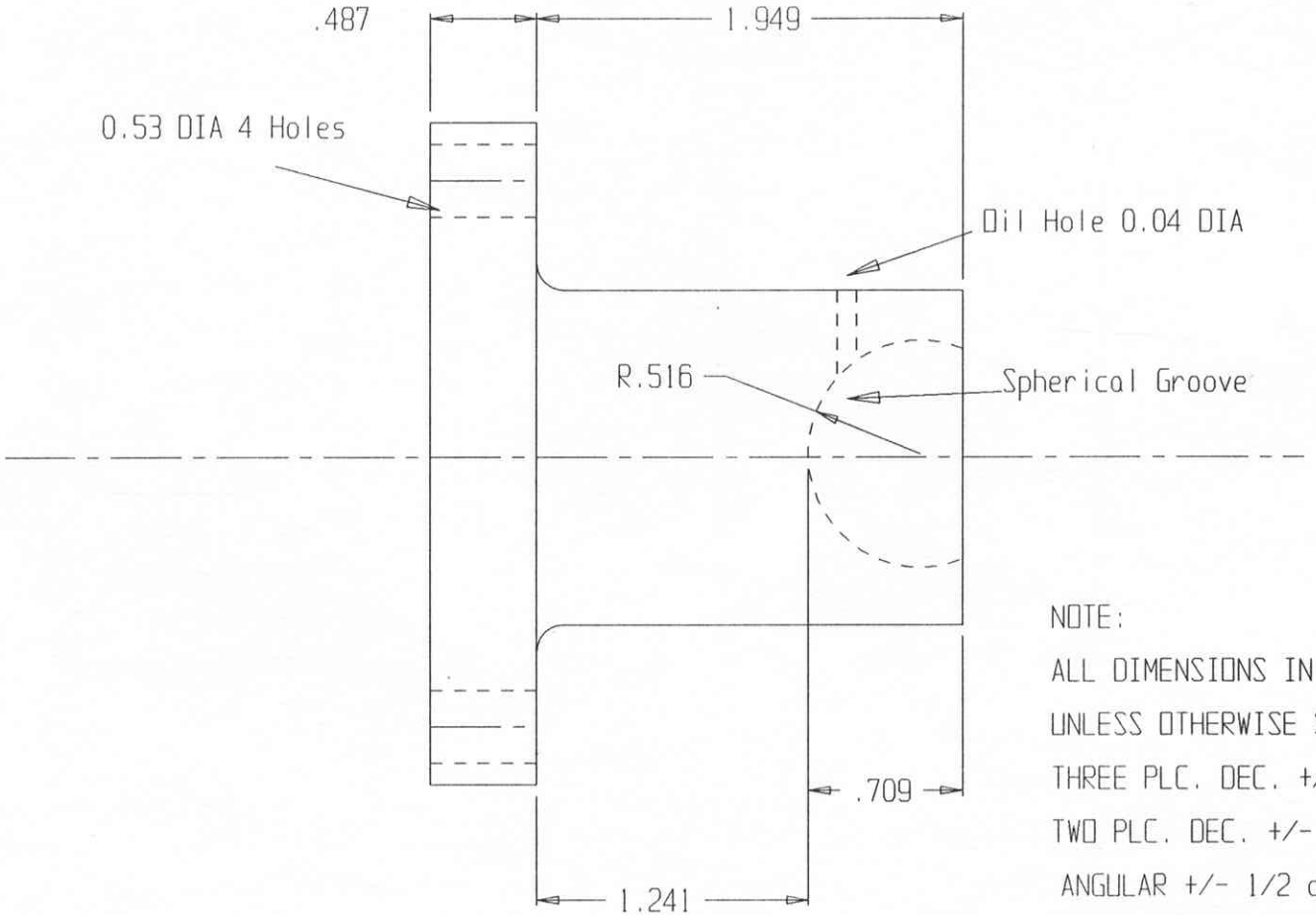
ANGULAR +/- 1/2 deg

PART 65



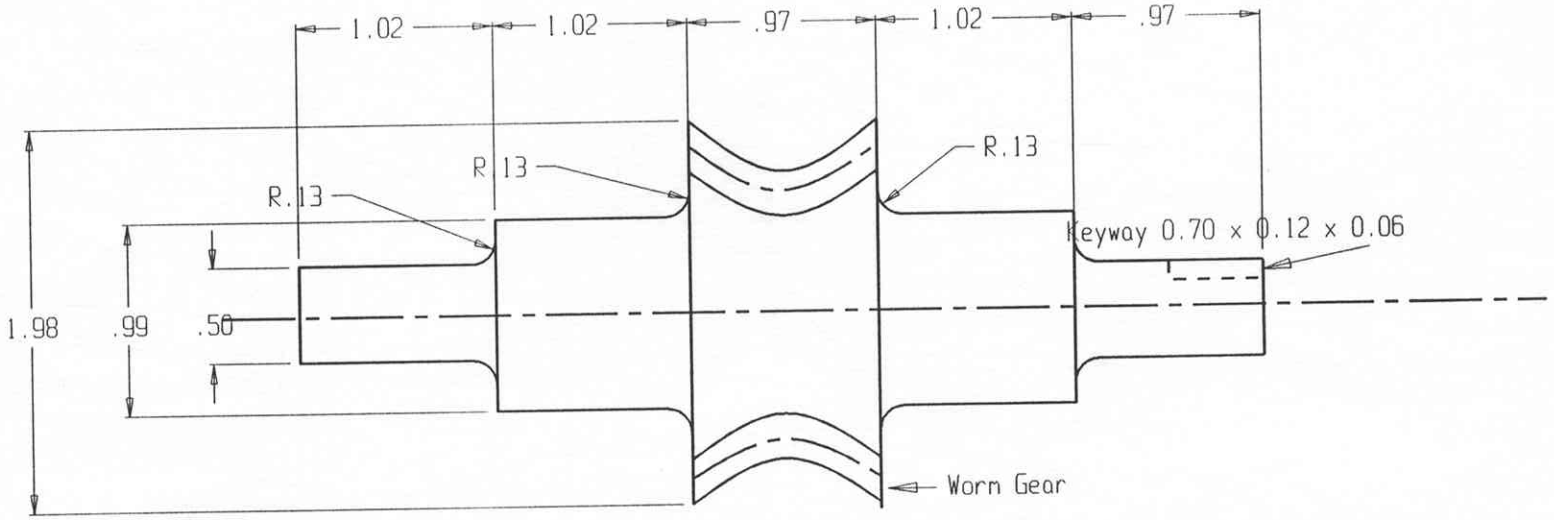
NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg .

PART 66



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 67



NOTE:

ALL DIMENSIONS IN INCHES

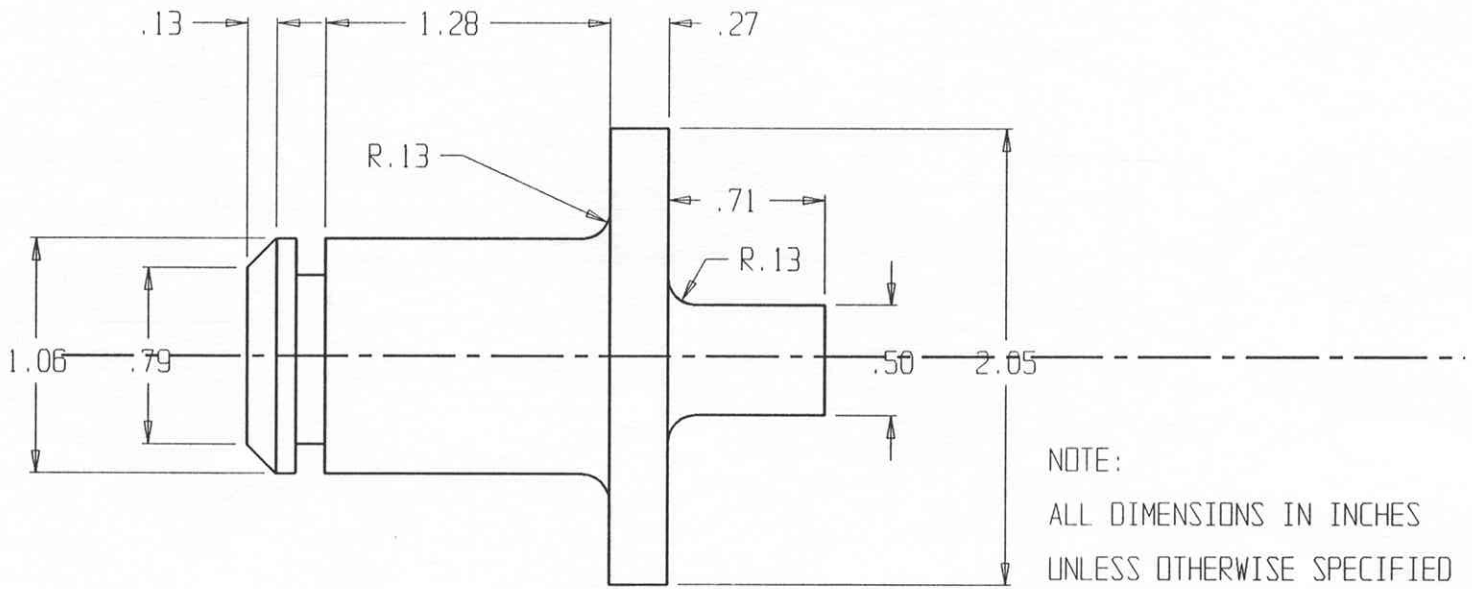
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

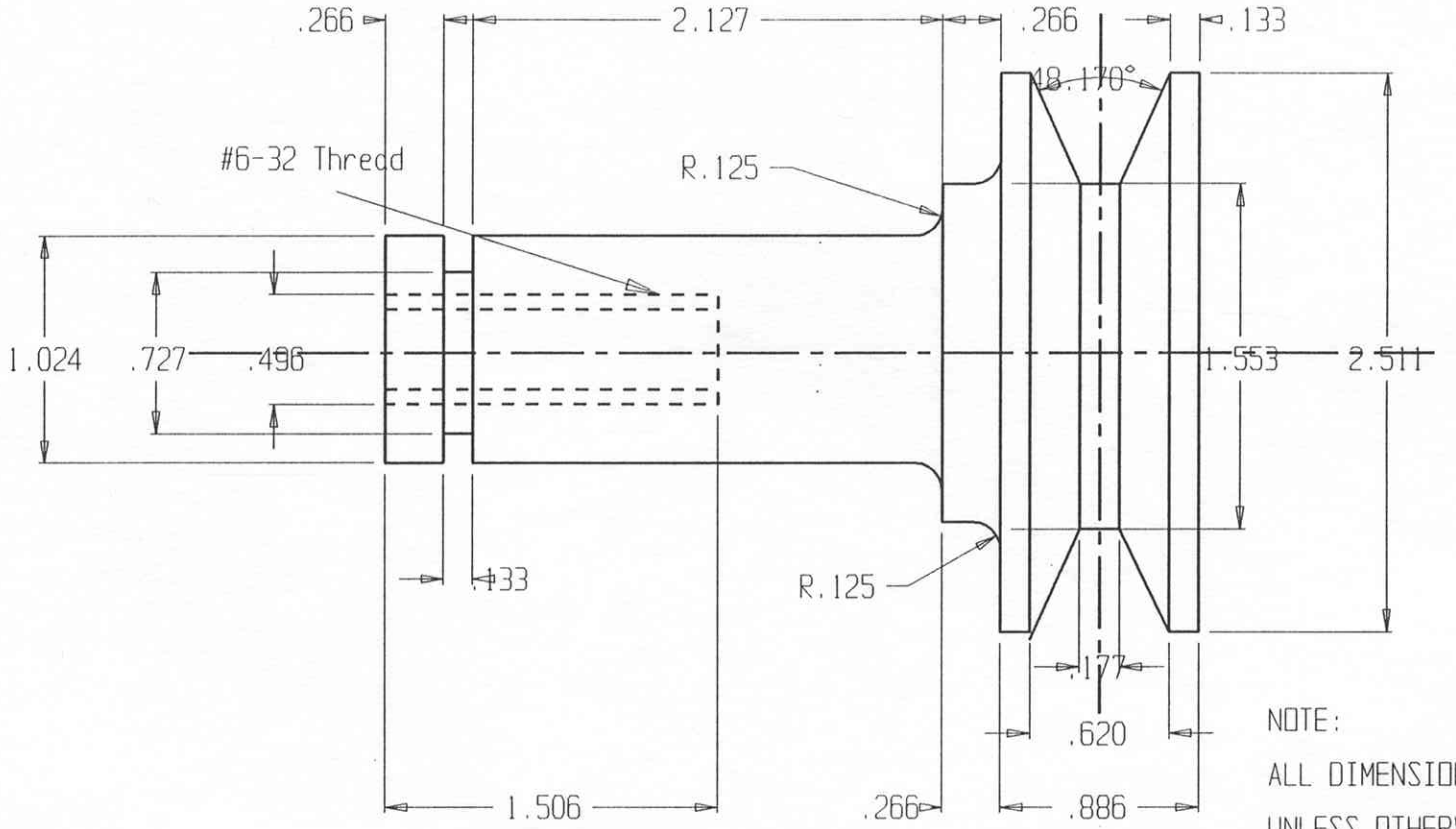
ANGULAR +/- 1/2 deg

PART 68



NOTE:
ALL DIMENSIONS IN INCHES
UNLESS OTHERWISE SPECIFIED
THREE PLC. DEC. +/- 0.001
TWO PLC. DEC. +/- 0.02
ANGULAR +/- 1/2 deg

PART 69



NOTE:

ALL DIMENSIONS IN INCHES

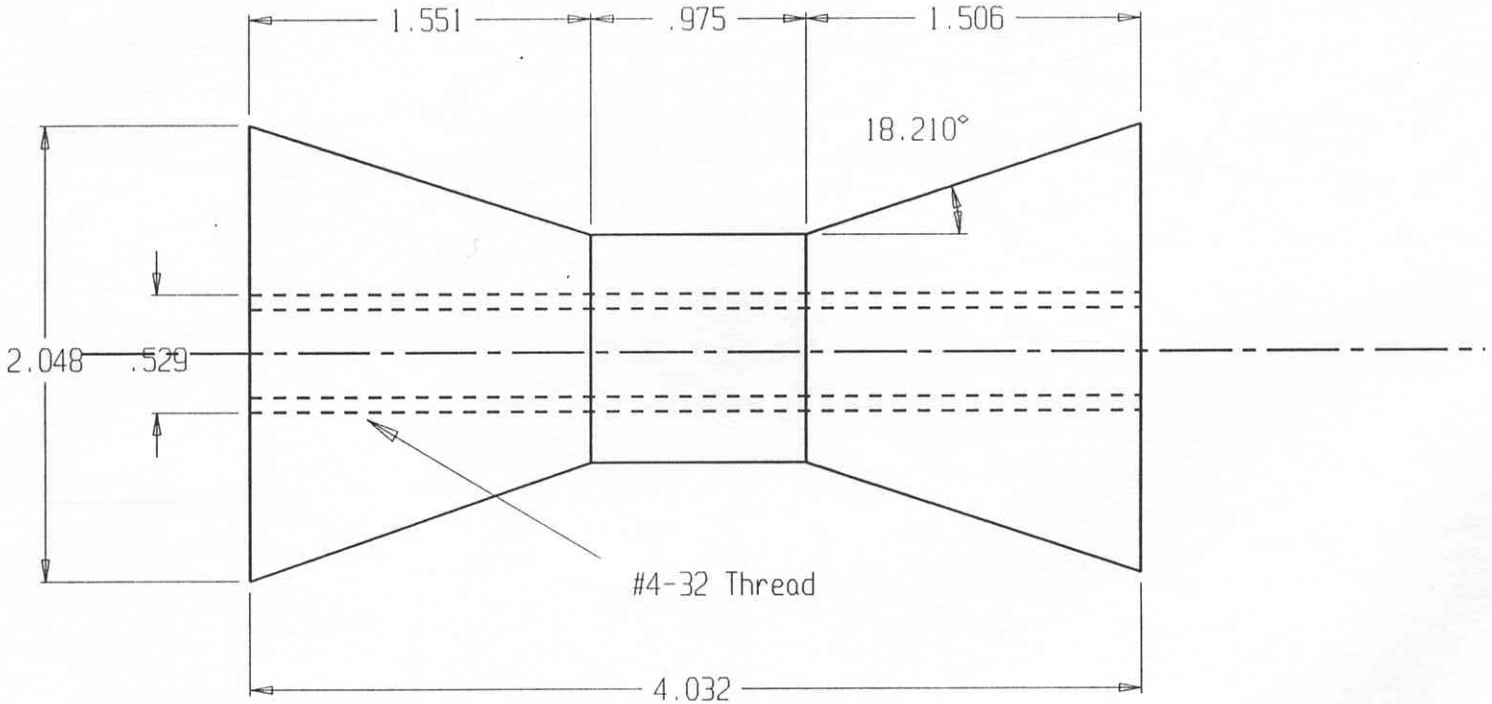
UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

ANGULAR +/- 1/2 deg

PART 70



NOTE:

ALL DIMENSIONS IN INCHES

UNLESS OTHERWISE SPECIFIED

THREE PLC. DEC. +/- 0.001

TWO PLC. DEC. +/- 0.02

ANGULAR +/- 1/2 deg

APPENDIX C

SAMPLE NETWORK DEFINITION FILE

Sample Network Definition File.

Adaptive Resonance 2 network. Vigilance = 0.94

Input gain = 0.70000

STM gain = 0.20000

Predictive gain = 0.70000

Learn rate = 0.40000

Excitation bias = 0.00400

Excitation threshold = 0.20000

Vigilance = 0.94000

run cycles = 10

inputs = 100 (10, 10)

outputs = 70 (10, 7)

Bottom-up weights:

0.05159	0.02075	0.03763	0.05626	0.07993	0.05817	0.05917	0.08992
0.01052	0.08789	0.01723	0.07120	0.05603	0.04071	0.08853	0.07488
0.07471	0.03767	0.01276	0.07113	0.06131	0.05276	0.08828	0.04542
0.03247	0.01489	0.04210	0.08269	0.06638	0.06440	0.04931	0.03605

Top-down weights:

0.00000	0.00000	0.00000	0.00000	0.00000	0.01030	0.00000	0.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.30172	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00445
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

APPENDIX D

SAMPLE INPUT FILE

Sample Input File

```
TRAIN 10,10,1,1

/* INPUT PART PATTERN: 1 */
0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
1.00,1.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,1.00,0.00,0.00,1.00,0.00,0.00,0.00
0.00,1.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00

/* OUTPUT PART PATTERN: 1 */
0.00

/* INPUT PART PATTERN: 2 */
0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
1.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00
0.00,0.00,0.00,1.00,1.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,1.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00

/* OUTPUT PART PATTERN: 2 */
0.00

/* INPUT PART PATTERN: 3 */
0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,1.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00

/* OUTPUT PART PATTERN: 3 */
0.00
```

```
/* INPUT PART PATTERN: 4 */
0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,1.00,0.00,1.00,0.00,1.00,0.00,0.00,0.00,0.00

/* OUTPUT PART PATTERN: 4 */
0.00

/* INPUT PART PATTERN: 5 */
0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00
1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00

/* OUTPUT PART PATTERN: 5 */
0.00

/* INPUT PART PATTERN: 6 */
0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
1.00,1.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00
0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
0.00,0.00,0.00,1.00,0.00,0.00,0.00,1.00,0.00,0.00
0.00,1.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00

/* OUTPUT PART PATTERN: 6 */
0.00
```

APPENDIX E

SAMPLE OUTPUT FILE

Sample Output File: Vigilance = 0.94

```
TRAIN 10, 10, 10, 7
/* Input vector 1 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 1 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.93,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 2 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 2 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.91,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 3 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 3 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.96, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 4 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 4 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.08, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 5 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 5 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.21, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 6 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 1.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 6 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 7 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 7 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 2.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 8 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 8 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.08,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 9 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 9 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.20, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 10 : */
0.00, 0.00, 1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 1.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 10 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.07, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 11 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 11 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.06, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 12 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 12 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.13, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 13 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 13 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.04, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,


/* Input vector 14 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 14 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.22, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 15 : */
0.00, 0.00, 1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 15 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.19,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 16 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 16 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
2.26, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 17 : */
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 17 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.01, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 18 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 18 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 2.07, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 19 : */
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 19 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.16, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 20 : */
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 20 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.10, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 21 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 21 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 2.06, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 22 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 22 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.26,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 23 : */
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 23 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.91,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 24 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 24 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.20, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 25 : */
0.00, 0.00, 1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 1.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 25 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.28, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 26 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 26 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.32, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 27 : */
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 27 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 2.21, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 28 : */
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 28 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.24, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 29 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 29 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
2.29, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 30 : */
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 30 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
2.13, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 31 : */
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 31 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.54, 0.00, 0.00, 0.00, 0.00,

/* Input vector 32 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 32 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.08,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 33 : */
0.00, 0.00, 1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 33 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.14,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 34 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 34 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.18,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 35 : */
0.00, 0.00, 1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 35 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.20,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 36 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 36 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.98,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 37 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 37 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
2.22, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 38 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 38 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
2.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 39 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 39 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.01, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 40 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 40 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.93,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 41 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 41 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.02, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 42 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 42 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.97, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 43 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 43 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.93,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 44 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 44 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.16, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 45 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 45 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.13, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 46 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 46 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.04, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 47 : */
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,


/* Output vector 47 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.94, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Input vector 48 : */
0.00, 0.00, 1.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 48 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.04, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 49 : */
0.00, 0.00, 1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 49 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 2.14,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,


/* Input vector 50 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 50 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.20, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
/* Input vector 51 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00,
1.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 51 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.40, 0.00, 0.00, 0.00, 0.00,

/* Input vector 52 : */
0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 1.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00,

/* Output vector 52 : */
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 2.14, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

APPENDIX F

GROUPING ANALYSIS

Table 4. Grouping Analysis: Vigilance Parameter:- 0.92

Group No.	PARTS
2	17,19,20,21,28,29,30,31
15	5,6,7,18,27,68,69
39	1,2,3,4,8,9,10,11,12,13,14,15,16,22,23,24,25,26 32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47, 48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63, 64,65,66,70

Table 5. Grouping Analysis: Vigilance Parameter:- 0.92

Group No.	2	15	39
Geometric Features			
Straight Outside	P	O	P
Taper Outside	O	O	O
Groove Outside	O	O	O
Pulley	O	O	O
Circumferential Holes	A	O	O
Keyway Outside	A	O	O
Threading Outside	A	O	O
Threading Inside	A	O	O
Cross hole	O	O	O
Straight hole	P	A	O
Taper hole	A	A	O
Blind hole	O	O	O
Step Hole	O	O	O
Hole	P	O	O

Key:-
A - Denotes absence of a feature.
P - Denotes presence of a feature.
O - Denotes presence or absence of a feature

Table 6. Grouping Anaysis: Vigilance Parameter:- 0.93

Group No.	PARTS
8	7,8,18,21,22,27,56,57,58,59,60
26	1,2,3,4,5,6,39,40,41,42,43,44,45,46,47,48,49,50, 52,53
38	17,20,28
50	9,10,11,12,13,14,15,16,23,24,25,26,32,33,34,35, 36,37,38,54,61,62,63,64,65,66,68,69,70
62	19,29,30,31

Table 7. Grouping Analysis: Vigilance Parameter:- 0.93

Group No.	8	26	38	50	62
Geometric Features					
Straight Outside	P	P	P	P	O
Taper Outside	O	O	A	O	O
Groove Outside	O	O	A	O	A
Pulley	A	A	A	P	O
Circumferential Holes	A	O	A	O	A
Keyway Outside	O	O	A	O	A
Threading Outside	O	O	A	O	A
Threading Inside	A	O	O	O	O
Cross hole	O	O	O	O	O
Straight hole	O	O	P	P	P
Taper hole	A	O	A	O	A
Blind hole	O	O	A	O	A
Step Hole	A	O	O	O	A
Hole	O	O	P	O	P

Key:-
A - Denotes absence of a feature.
P - Denotes presence of a feature.
O - Denotes the presence or absence of a feature

Table 8. Grouping Anaysis: Vigilance Parameter:- 0.94

Group No.	PARTS
24	4,6,9,10,11,12,13,14,24,25,26,39,41,42,44,45,46, 48,50,52,53,54,59,60,61,65,66,68,70
26	7,8,18,21,27,67
29	17,28
30	1,2,8,15,22,23,32,33,34,35,36,40,43,49,57,58,69
39	19,20
51	16,29,30,37,38,62,63,64
66	31,51,56

Table 9. Grouping Analysis: Vigilance Parameter:- 0.94

Group No.	24	26	29	30	39	51	66
Geometric Features							
Straight Outside	P	O	P	P	P	O	P
Taper Outside	O	O	A	P	A	O	O
Groove Outside	O	O	A	O	A	O	A
Pulley	O	A	A	O	A	O	A
Circumferential Holes	O	A	A	A	A	A	A
Keyway Outside	O	O	A	O	A	O	O
Threading Outside	O	O	A	O	A	A	O
Threading Inside	O	A	O	O	O	A	O
Cross hole	A	O	A	O	O	O	A
Straight hole	O	O	P	O	P	O	P
Taper hole	O	A	A	O	A	A	A
Blind hole	A	O	A	O	A	A	P
Step Hole	O	A	O	O	A	A	A
Hole	O	O	P	O	P	O	P

Key:-
A - Denotes absence of a feature.
P - Denotes presence of a feature.
O - Denotes either the presence or absence of a feature

Table 10. Grouping Anaysis: Vigilance Parameter:- 0.95

Group No.	PARTS
8	36,44,45,46,58,65,68,69,70
11	4,32,33,34,35,42,43,61
20	2,8,9,10,11,12,13,14,15,24,25,26,38,50,52,53,54, 59,60
23	19,28
35	23,48
41	7,18,22
55	1,3,6,39,40,49,55
60	27,67
64	29,30,37,62,63,64
65	16,41,51

Table 11. Grouping Analysis: Vigilance Parameter:- 0.95

Group No.	8	11	20	23	35	41	55	64	65
Geometric Features									
Straight Outside	P	P	P	P	P	P	P	P	P
Taper Outside	P	O	O	A	A	O	A	P	O
Groove Outside	P	A	O	A	A	A	A	O	A
Pulley	O	A	A	A	A	A	A	P	A
Circumferential Holes	A	A	O	A	O	A	O	A	O
Keyway Outside	A	O	A	A	A	O	O	O	A
Threading Outside	O	A	O	A	A	O	O	A	O
Threading Inside	O	O	O	A	A	A	O	A	A
Cross hole	A	A	O	O	A	O	O	O	A
Straight hole	O	P	P	P	A	O	O	O	P
Taper hole	A	O	O	A	A	A	O	O	O
Blind hole	O	A	A	A	A	O	A	O	O
Step Hole	A	P	O	A	A	A	O	O	O
Hole	O	P	P	P	A	O	O	O	P

Key:-
A - Denotes absence of a feature.
P - Denotes presence of a feature.
O - Denotes either the presence or absence of a feature

Table 12. Grouping Anaysis: Vigilance Parameter:- 0.96

Group No.	PARTS
9	23,48
12	8,9,10,11,12,14,19,24,25,28,50
15	4,13,15,32,33,34,35,45,46,49
20	26,44,59,60,65,68
40	29,37,62,63,64,70
44	18,22
49	36,43,54,69
52	1,3,55
53	16,30
65	6,39,40,42
68	27,67

Table 13. Grouping Analysis: Vigilance Parameter:- 0.96

Group No.	9	12	15	20	40	44	49	52	53	65	68
Geometric Features											
Straight Outside	P	P	P	P	P	P	P	P	P	P	P
Taper Outside	A	O	P	O	P	O	P	A	A	O	O
Groove Outside	A	O	A	O	A	A	O	A	P	A	A
Pulley	A	A	A	A	P	A	O	A	A	A	A
Circumferential Holes	O	O	A	A	A	A	A	A	A	O	A
Keyway Outside	A	A	A	A	O	A	A	O	A	P	O
Threading Outside	A	A	O	O	A	A	O	A	A	O	O
Threading Inside	A	O	O	A	O	O	O	A	A	O	A
Cross hole	A	A	A	A	O	A	A	O	P	A	A
Straight hole	A	P	P	O	O	O	O	O	P	P	A
Taper hole	A	A	O	A	A	O	O	A	A	A	A
Blind hole	A	A	A	A	A	O	O	A	A	A	A
Step Hole	A	O	O	A	A	O	O	O	A	P	A
Hole	A	P	O	O	O	P	O	O	P	P	A

Key:-
A - Denotes absence of a feature.
P - Denotes presence of a feature.
O - Denotes either the absence or preence of a feature

Adaptive Resonance Theory

Introduction

This model was used for the proposed application to automatically classify parts into part families in a Group Technology manufacturing environment. Adaptive Resonance Theory is a mathematical model of a unsupervised neural network architecture. The network architecture is unique among all the existing paradigms. The architecture detects and remembers statistically predictive configuration of featural elements which are derived from all the input patterns. The main advantage in using ART is the scope of learning any kind of configuration of part families or other patterns. In ART2 the learned part patterns undergo self-organization and self-stabilization as training progresses.

Self-organization is said to have occurred when the network classifies the input part patterns into different part families automatically without external help. This process of self-organization avoids the use of a teacher to decide the size of the part family.

Self-stabilization is said to have occurred when the learned history is not washed away by the more recent learning. This is true even if the inputs are presented in any arbitrary order and in any arbitrary complexity. The search strategies are dynamically modified and updated as

new part patterns are learned. The learned part patterns result in the formation of critical feature patterns for each individual part family in the network. The critical feature patterns are concepts which the network develops and cannot be accessed by the user.

ART2 Methodology and Working

Within ART multiple interacting memory mechanisms are employed to monitor and adaptively react to the novelty of part patterns or process familiar and unfamiliar part patterns. Familiar events are processed by an attentional subsystem. This system establishes more precise internal representation of familiar part patterns. The attentional subsystem by itself is unable to simultaneously maintain stable representation of familiar part patterns and to create new part families for novel or unfamiliar part patterns. The attentional subsystem comprises the whole system excluding the orienting mechanism.

The second subsystem is an orienting subsystem that resets the attentional subsystem when a unfamiliar part pattern is encountered by the network. The orienting subsystem is capable of identifying whether a new part pattern is familiar and well represented by an existing part family or is unfamiliar and in need of a new part family.

The Attentional subsystem consists of the following components.

1) Node 1

2) Node 2

3) Attentional Gain Control

4) Attentional Priming

Node 1 and Node 2 consists of processing elements or "feature detectors". Node 1 and node 2 are interconnected fully from every processing element in node 1 to each element in node 2. These interconnections are named the Long Term Memory (LTM) traces. There also exists two types of memory in an ART2 architecture, as in human beings. The Short Term Memory (STM) is the temporary memory which maps the input pattern to the nodes. The Long Term Memory (LTM) is the memory in the system resulting from past learning. The orienting subsystem has no further sub-classification.

The Attentional Priming mechanism primes or sensitizes the input node to receive a bottom-up input pattern. This bottom-up input is the input pattern to the network. The bottom-up input part pattern is compared with the expected part pattern created by the network. The attentional gain control mechanism is a test system which enables the network to distinguish between top-down and bottom-up part patterns.

ART2 Working

The working of ART2 can be explained by following a step by step procedure on the flow chart and schematic given in the following pages. The flow chart is arranged in an hierarchy of two levels. The first level (see Figure 10(a) and Figure 10(b)) outlines the working of ART2. This level explains the overall working of ART2.

The second level explodes and explains each step in level 1 in a systematic and logical fashion. In this section each step will be explained in detail and a reference will be made to the flowchart for better understanding. Now a description of all the steps under level 2 will be given.

Step 1:

The input to the network is a file of parts coded as binary matrices. Refer to Figure 11(a) and 11(b). Let a typical binary matrices be named BM. The binary matrix impinges on the processing elements of node 1 which is part of an attentional subsystem as described above. The input pattern BM as it impinges on the node 1 is passed through an output function of each unit. The propagation and activation rule are responsible for achieving this output. The output pattern obtained as a result is an activation pattern across node 1. Let the pattern of activation be termed PAn as shown in the Figure 10(a). The pattern

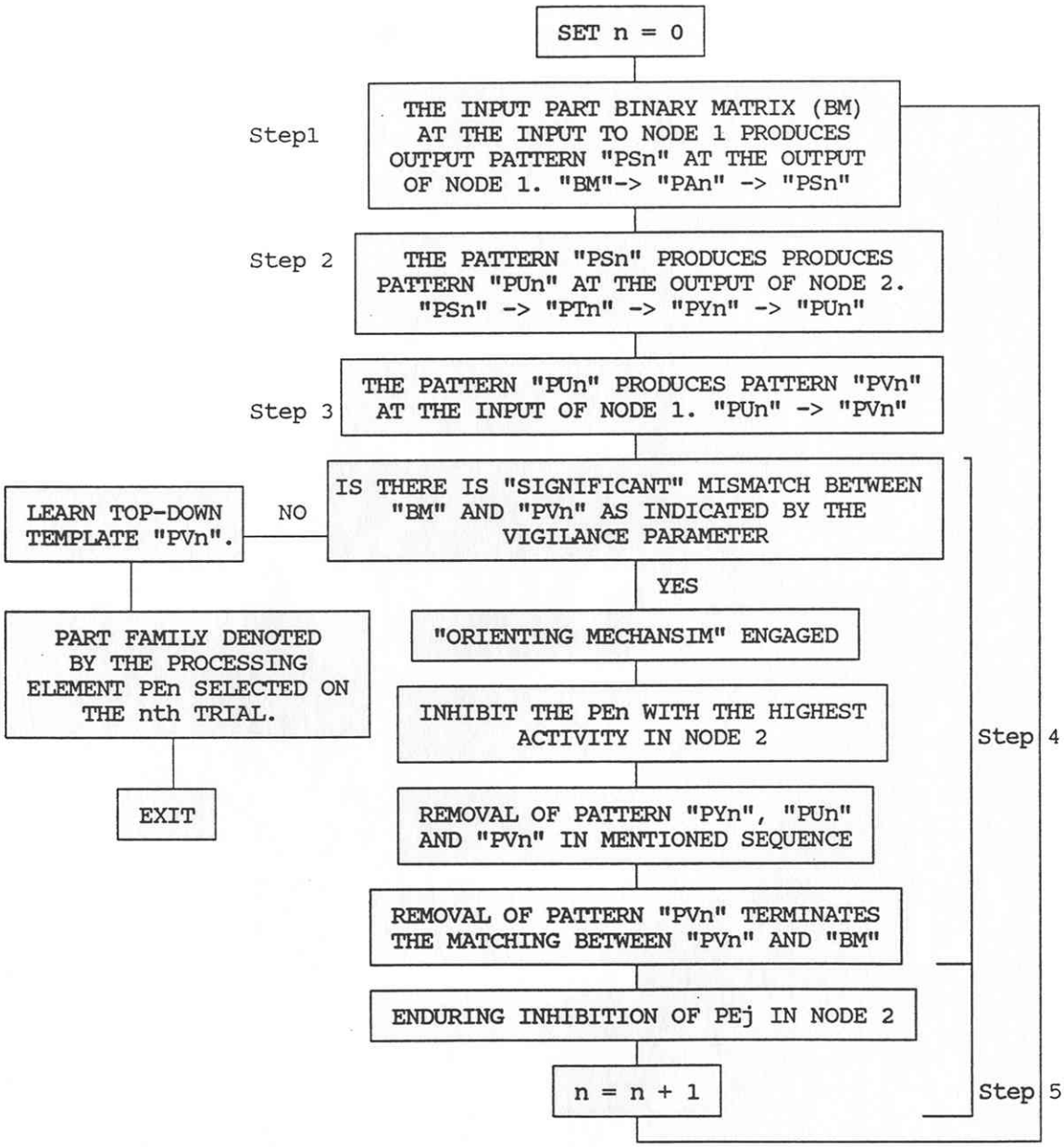


Figure 11(a). ART2 - METHODOLOGY, Level 1

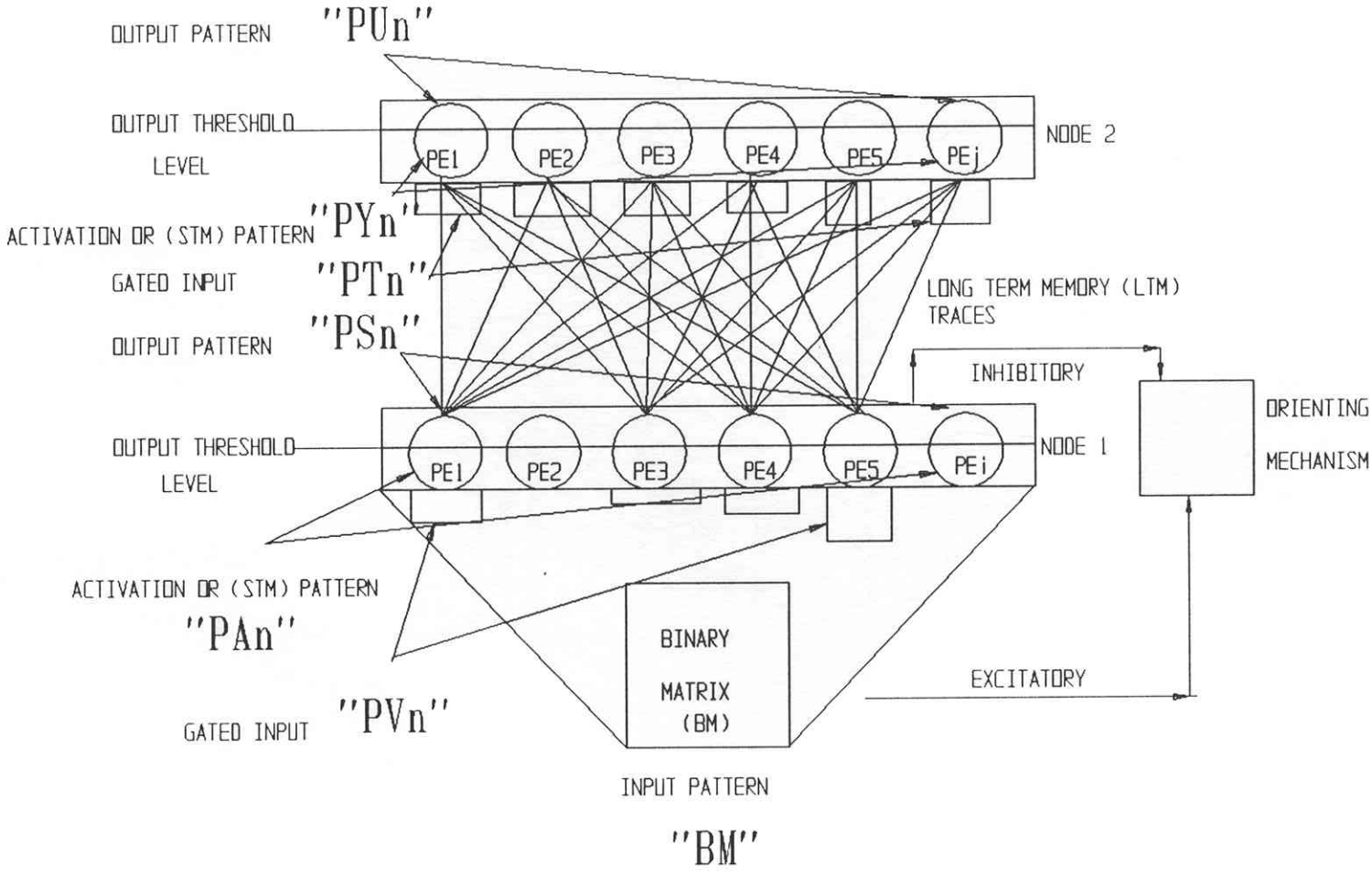


Figure 11(b), ART2 - METHODOLOGY, Level 1

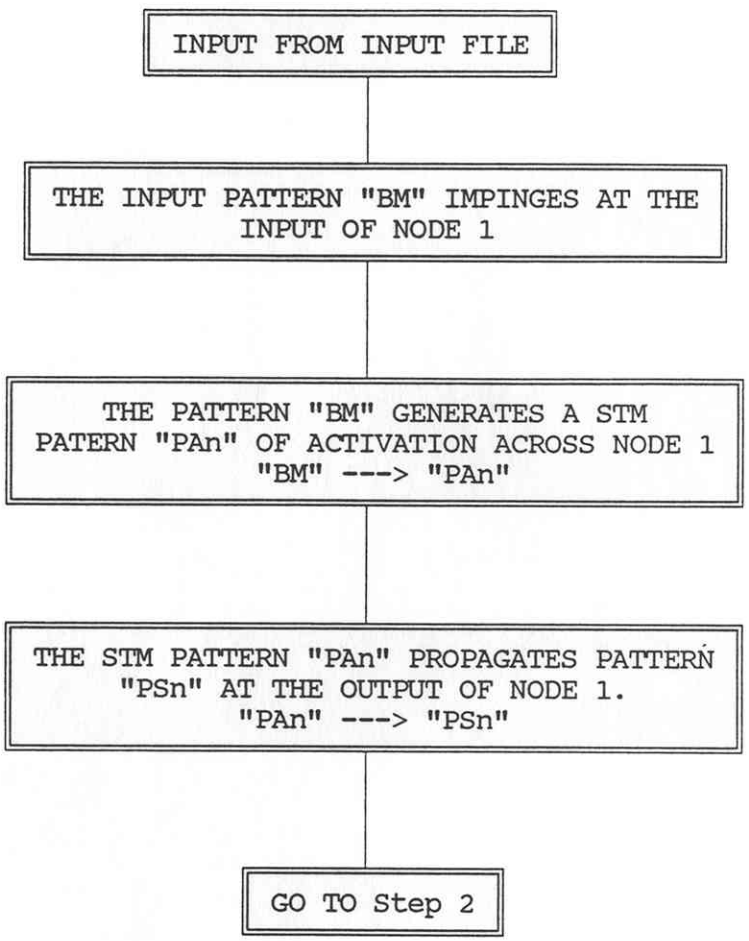


Figure 12(a). ART2 - METHODOLOGY, Level 2, Step 1

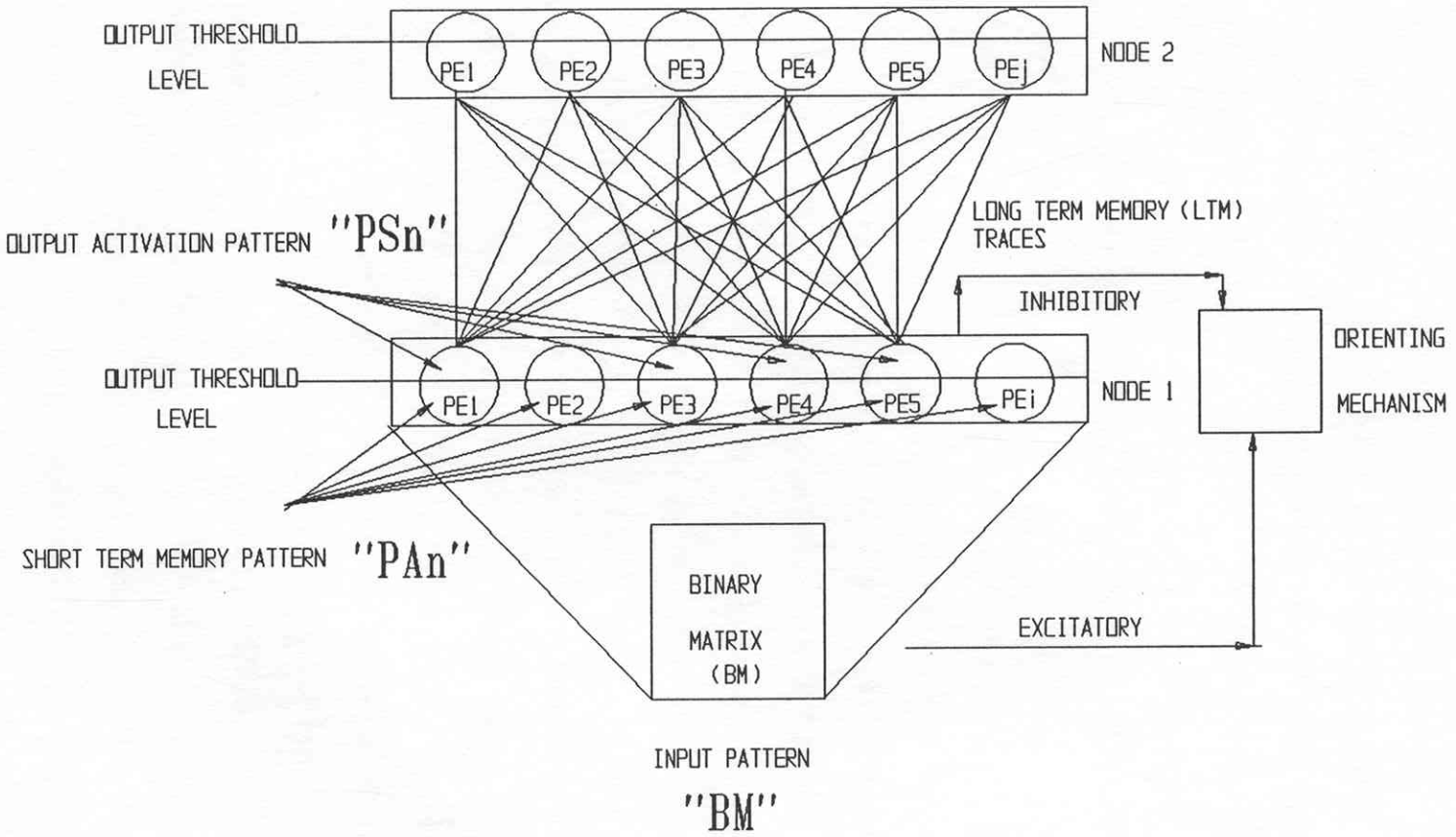


Figure 12(b), ART2 - METHODOLOGY, Level 2, Step 1

obtained is a STM pattern. Every input node has a pattern of activation. There is a threshold defined by the propagation rule for every processing element which does not allow patterns below a certain activation to propagate. The propagation function used is a sigmoid activation function. These patterns propagate to node 2. Thus as a result the pattern PAn results in a selective pattern PSn across the input node.

Step 2:

The pattern PSn traveling to node 2 is multiplied by the connection weight or the LTM trace. Refer to Figure 12(a) and 12(b). This multiplied signal reaches node 2. This signal is added at all the processing elements in node 2. The result of adding all the signals at node 2 generates the pattern PTn as the input of node 2. The transformation of pattern PSn to PTn is called an Adaptive Filter. The pattern PTn is now the input to node 2. At this node an interesting phenomenon takes place. The processing element which has the highest activation becomes more and more active. Also the elements next to it with lesser activations become more and more inhibited and as a result become less positive. This results in only one positive element in the output layer. This process is technically termed "Contrast Enhancement". The activation pattern obtained at the output of node 2 is termed PYn. The choosing of the element having the largest input is a

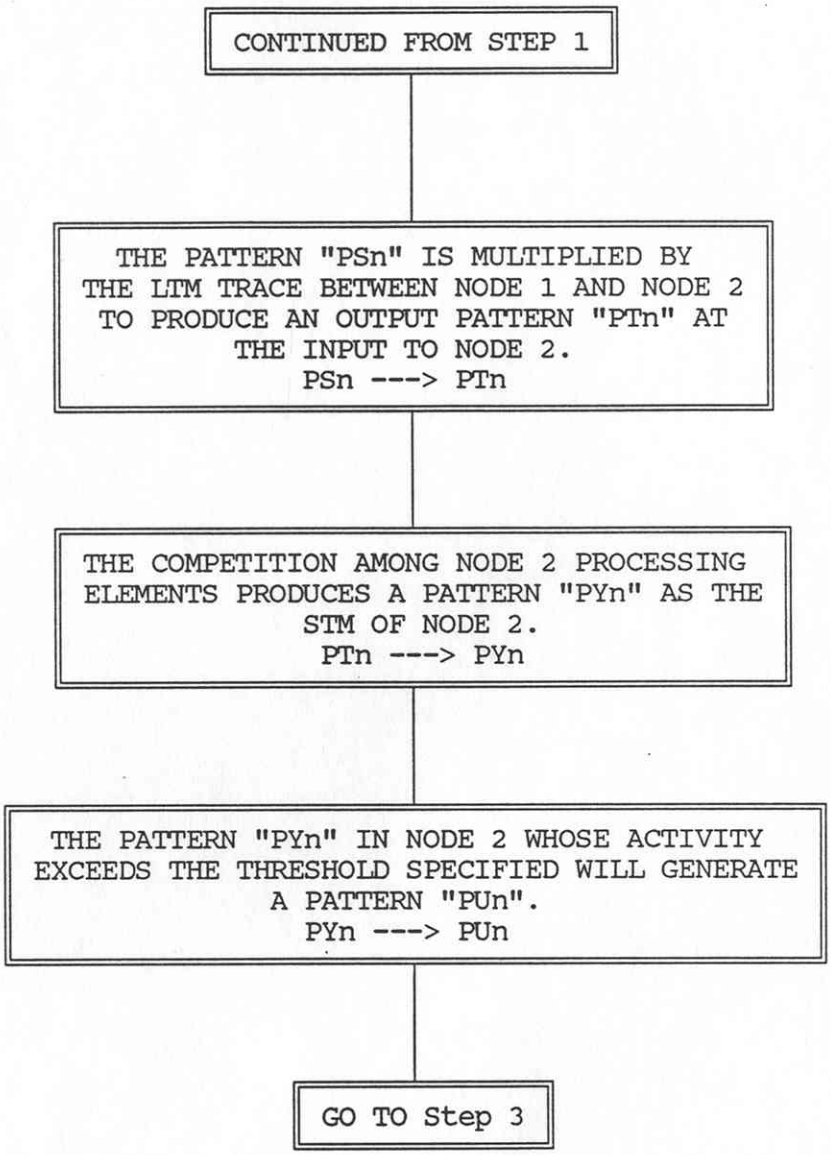


Figure 13(a). ART2 - METHODOLOGY, Level 2, Step 2

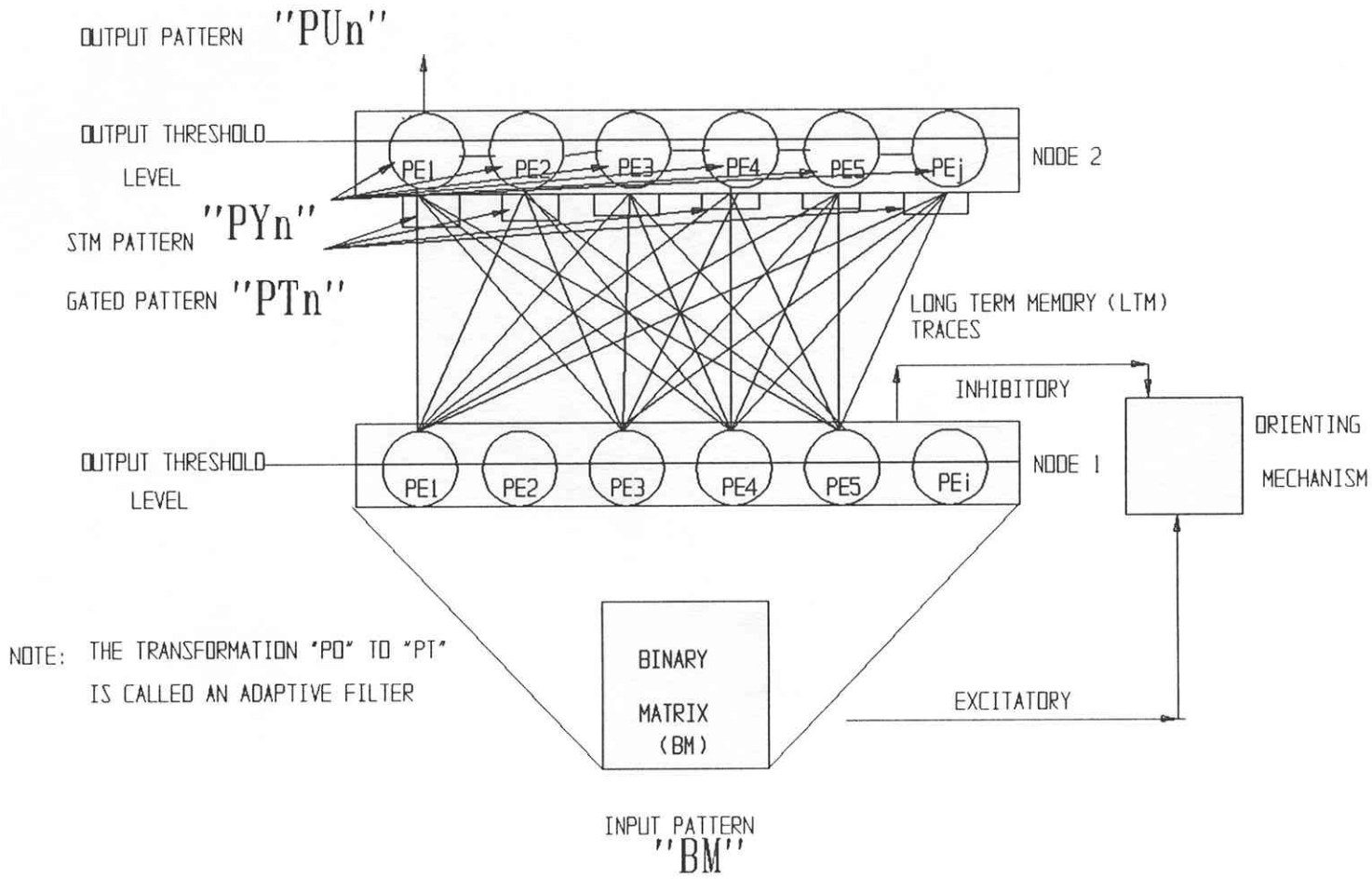


Figure 13(b), ART2 - METHODOLOGY, Level 2, Step 2

special case of the class of Adaptive Resonance Theory models. The part pattern PYn now stored in the short term memory (STM) of node 2. The STM contrast enhanced pattern PYn in node 2 propagates a pattern down to the node 1. This top-down pattern is termed PUn. Activities above the threshold generate a pattern PUn.

Step 3:

The pattern PUn is multiplied by the connections between node 1 and node 2. Refer to Figure 13(a) and 13(b). These are also termed the LTM traces. The multiplied signals are now at node 1. The multiplied and summed input signal node 1 generates a pattern PVn at the input of node 1. The pattern PVn is also an adaptive filter. The pattern PVn is known as the top-down template or "Learned Expectation" because this pattern is obtained as a result of the bottom-up original part pattern. The node 1 is now acted upon by two patterns. One pattern is BM which is the original input to the network and which resulted in the STM pattern PAn at node 1, and the pattern PVn. The second pattern PVn is as a result of the pattern PYn at node 2. The two patterns BM and PVn now generate an entirely different pattern at node 1. Thus initially the input pattern resulted in the pattern PAn of STM across node 1. Now the combined effect of the two patterns causes a pattern PAn^*. This pattern PAn^* is different from pattern PAn. In a conceptual sense node 1 does a matching job. It tries to

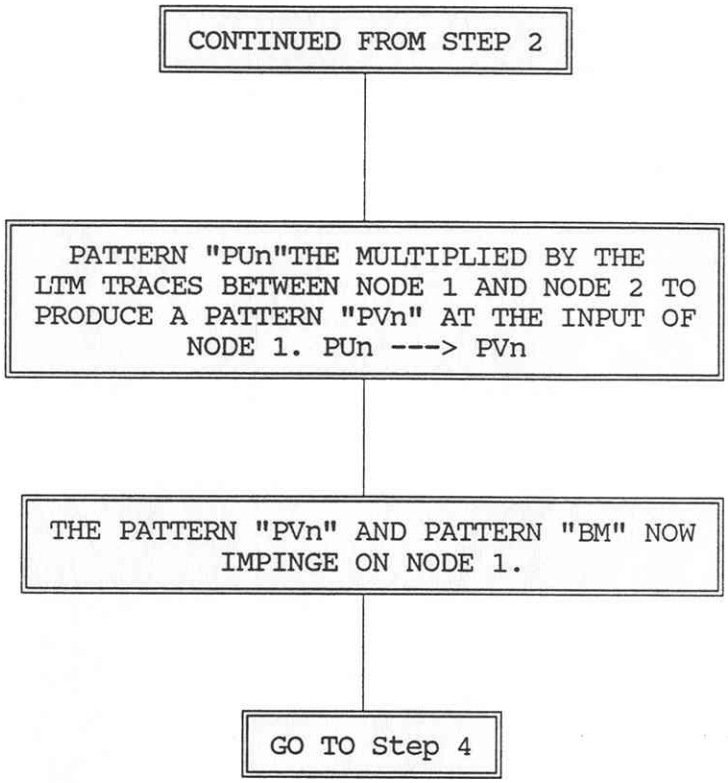


Figure 14(a). ART2 - METHODOLOGY, Level 2, Step 3

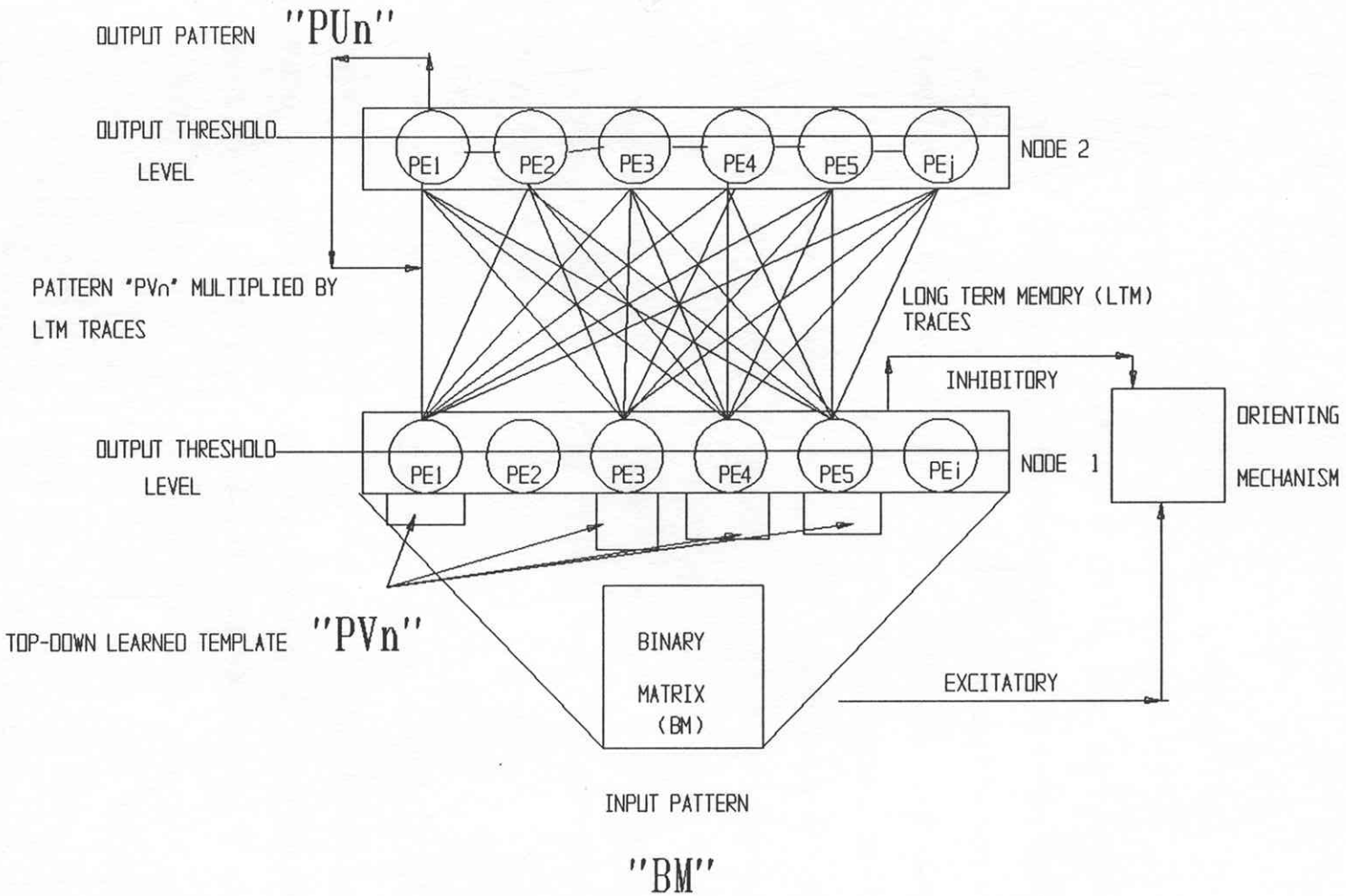


Figure 14(b), ART2 - METHODOLOGY, Level 2, Step 3

match and compare PVn and BM. The amount of match or mismatch determines the future course of learning. The transformation of the pattern from PAn to PVn takes place at high speed. Conversely the LTM traces and the adaptive filters change their values very slowly. The second memory system now comes in use in deciding the future course of action of the system. This system is the orienting subsystem. The input pattern BM also excites the orienting mechanism. The pattern PAn of activation inhibits the same. If the patterns BM and PVn do not match then this inhibits the pattern PAn from node 1. The inhibition of the STM activity across node 1 causes the orienting mechanism to fire because of less inhibition. The degree of inhibition is indicated by the difference between the input pattern BM and the pattern PVn. The higher the mismatch the higher is the inhibition.

Step 4:

The inhibition of the STM activity causes the orienting system to become active and trigger a signal to node 2. Refer to Figure 14(a) and Figure 14(b). The signal is inhibitory in nature and selective and inhibits only the active nodes. In the special case the node with a positive output. This causes node 2 activity to be reset. This inhibition is long lasting to ensure that the same processing element is not selected again for the reinstated pattern. The inhibition of the STM activity leads to the

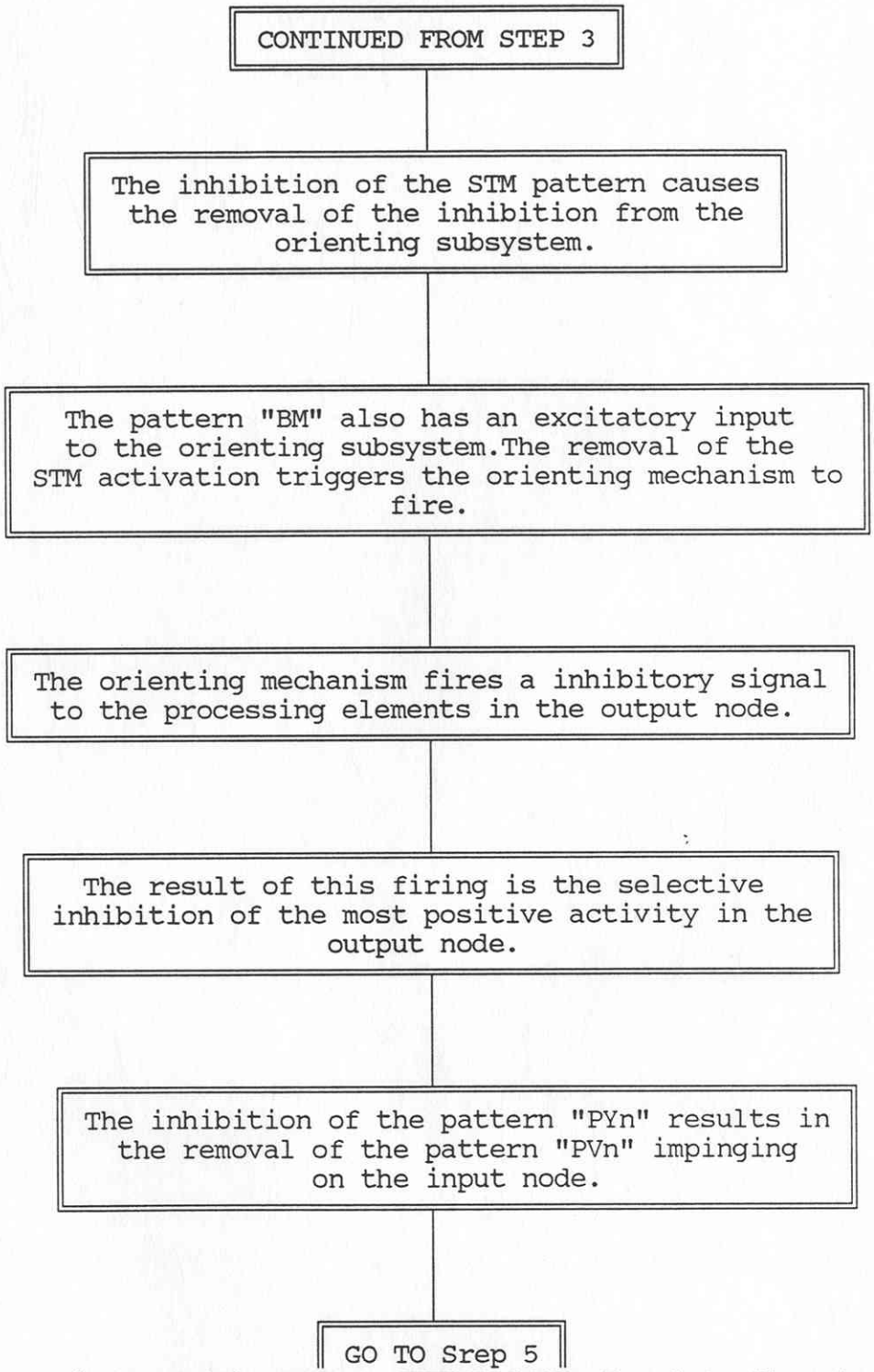


Figure 15(a). ART2 - METHODOLOGY, Level 2, Step 4

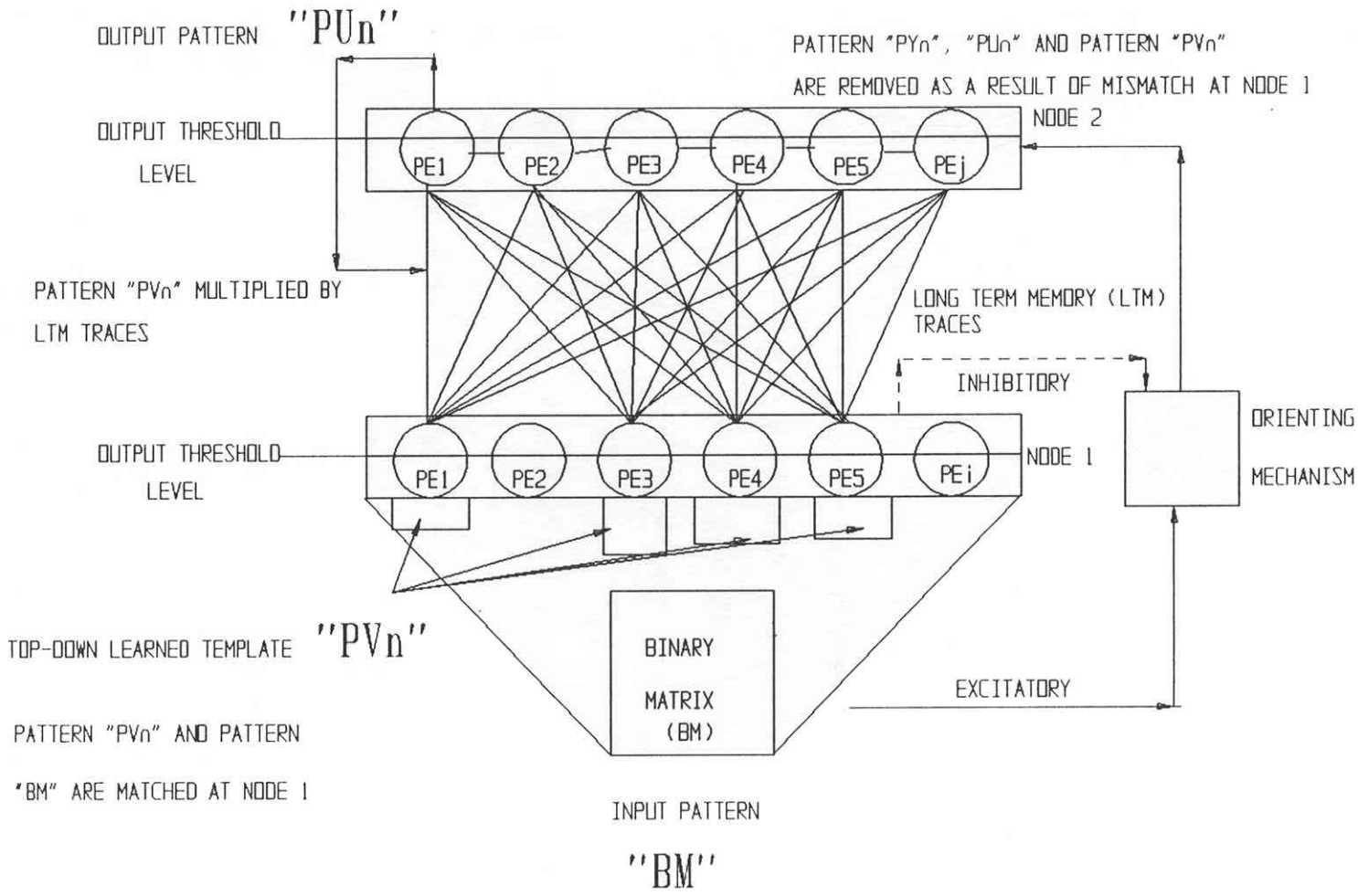


Figure 15(b), ART2 - METHODOLOGY, Level 2, Step 4

inhibition of the STM pattern PYn across node 2. This removes the template PVn across node 1.

Step 5:

This results in the stopping of the mismatch between PVn and BM. Refer to Figures 15(a) and 15(b). The inhibition of node 2 is long lasting. Now again the new pattern PAn* causes the creation of a new pattern PAn* across node 2. This pattern as described earlier causes the creation of the pattern PVn* across node 1. There is again a pattern matching as explained earlier. Node 2 which was aroused earlier cannot be aroused now due to the enduring inhibition. If there is a mismatch again at node 1 then the orienting subsystem is engaged again.

Vigilance Level Tunes Categorical Coarseness: Disconfirming feedback

First we need to define the vigilance parameter ϱ. Let |POS| denote the number of input pathways which receive positive inputs when the pattern BM is presented. Assume that each such input pathway sends an excitatory signal of fixed size E to the orienting mechanism O whenever BM is presented, so that the total excitatory input to O is E|POS|. Assume that each input node whose activity becomes positive due to BM generates inhibitory signal of fixed size I to the orienting mechanism O. Let |NEG| denote the number of active pathways from node 1 to the orienting mechanism

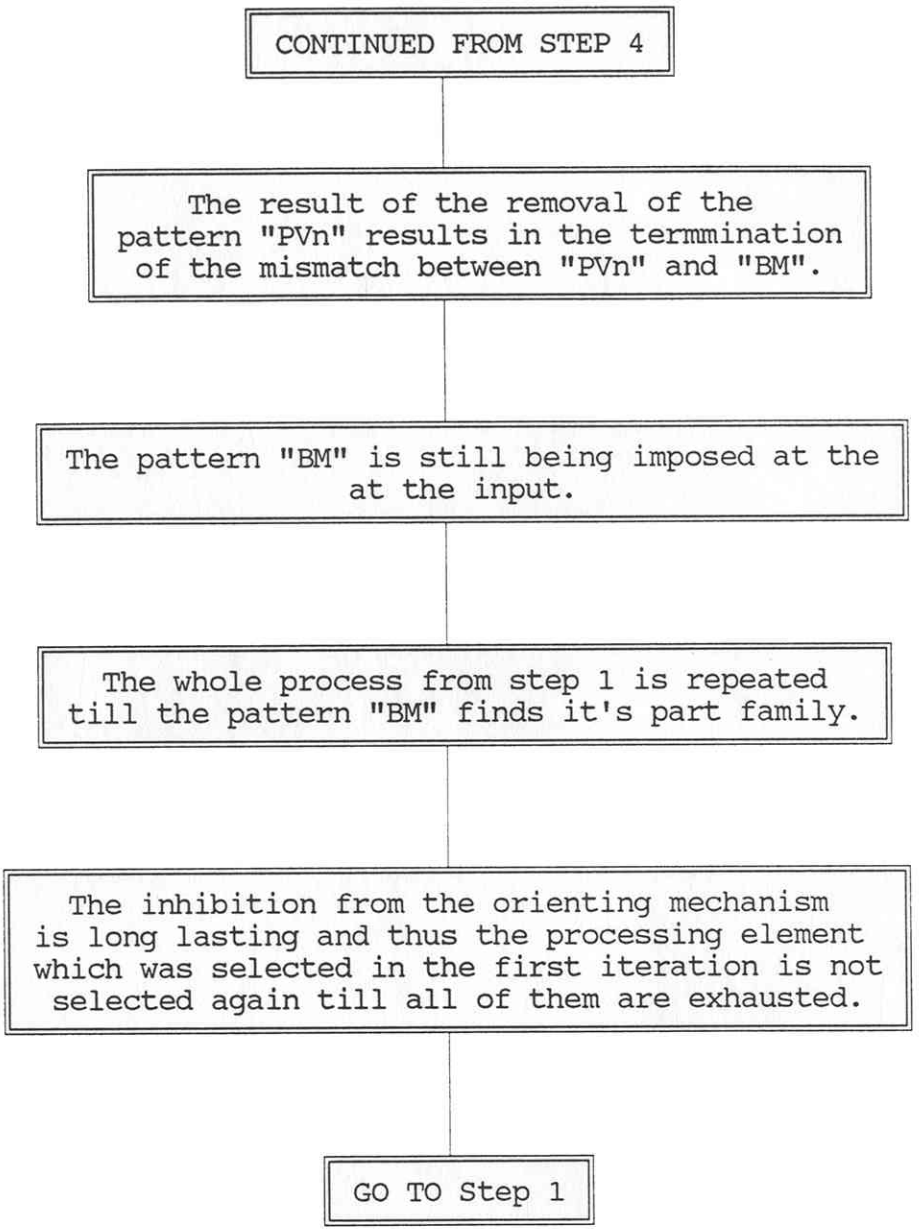


Figure 16(a). ART2 - METHODOOGY, Level 2, Step 5

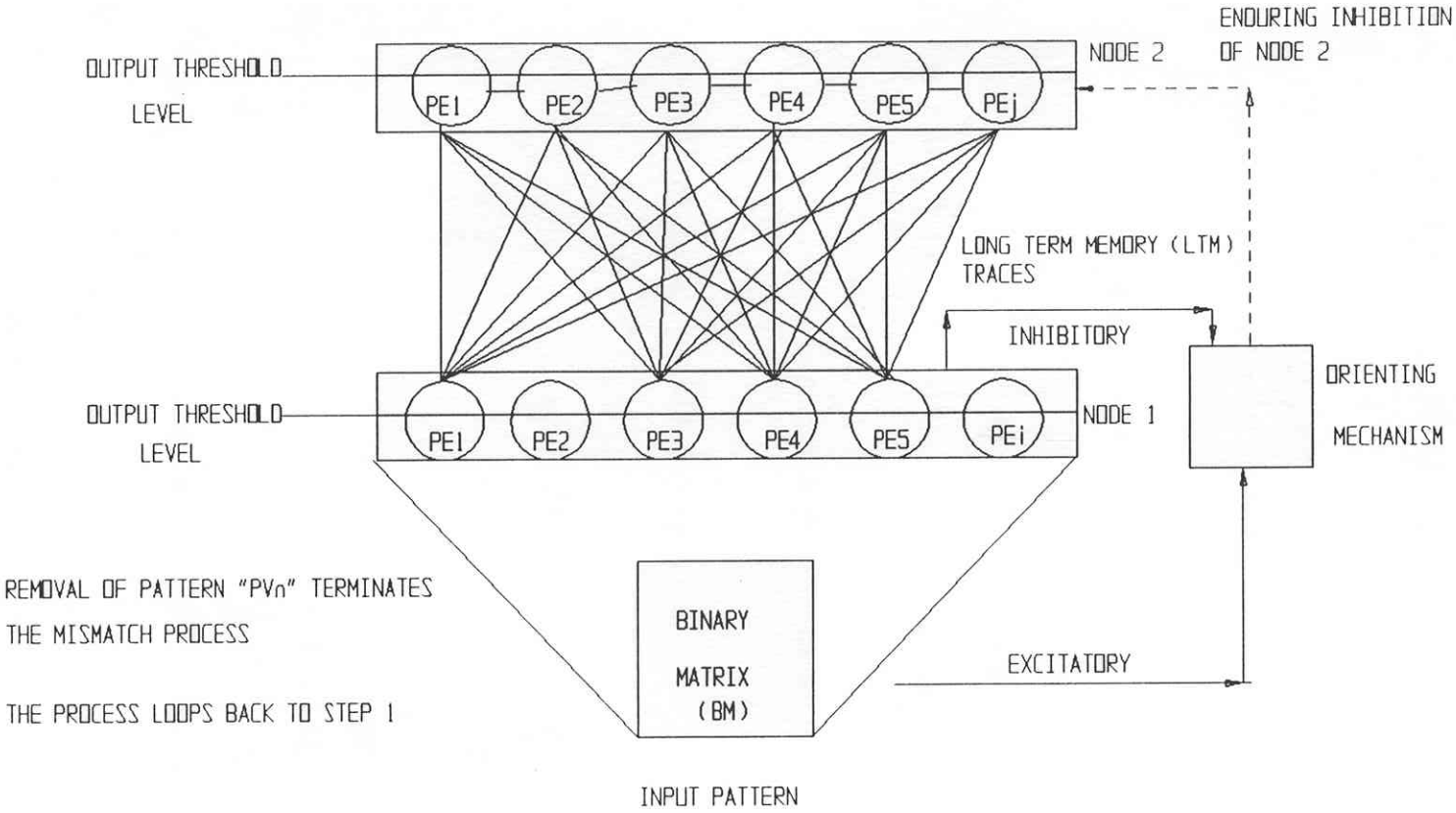


Figure 16(b), ART2 - METHODOLOGY, Level 2, Step 5

that are activated by the STM pattern PAn across node 1. The total inhibitory signal from node 1 to the orienting mechanism is I|NEG|.

When E|POS| ≥ I|NEG|, the orienting subsystem receives a net excitatory signal and generates a non-specific reset signal to node 2.

The quantity $\varrho = E \div I$

is called the vigilance parameter of the orienting subsystem. The STM reset is triggered when

$$\varrho \geq |POS| \div |NEG|$$

The STM reset is prevented when

$$\varrho \leq |POS| \div |NEG|$$

In short, the proportion |POS| ÷ |NEG| of the input pattern BM which is matched by the top-down template to generate PX must exceed ϱ in order to prevent STM reset at node 2.

While node 2 is inactive, |POS| = |NEG|. Activation of the orienting subsystem is always forbidden in this case to prevent an input I from resetting its correct node 2 code. This constraint is achieved if $\varrho \leq 1$.

that is $\varrho \leq I$.

In summary, a bad mismatch at node 1 causes a large

collapse of the activity of node 2, which leads to the activation of the orienting mechanism.

Advantages of ART2 over supervised learning models

In this section an attempt is made to compare ART2 to the popular supervised learning paradigms.

1) Stability.

Supervised learning paradigms become unstable as the input environment becomes complex. This means that the network never converges to a optimum value, instead it fluctuates around some arbitrary unpredictable value. An omniscient teacher has to decide if the network has learnt enough in response to an arbitrary input environment. Conversely ART2 paradigm learns without a teacher and accepts any arbitrary inputs.

2) Exemplar or Prototypes:

Within a supervised learning paradigm an expected or a template pattern is imposed on every trial by an external teacher. The errors are computed by comparing each component of the expected output with the actual output. This deficit deviods this paradigm the important characteristic of creating critical feature patterns.

APPLICATION OF NEURAL NETWORKS TO AUTOMATICALLY CLASSIFY ROTATIONAL PARTS INTO PART FAMILIES

by

HIREN H DESAI

B.E.(Production Engineering)., Shivaji University
Solapur, India, 1987

AN ABSTRACT OF A THESIS

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Department of Industrial Engineering
KANSAS STATE UNIVERSITY
Manhattan, Kansas

1990

ABSTRACT

This thesis details the procedure for automatic classification of rotational parts into part families using an artificial neural network. The classification is based on geometric features and tolerances. The neural network paradigm employed belongs to a class of Adaptive Resonance Theory Models. The training of the network was done on a commercially available software package.

The major conclusions that can be drawn from this research are: 1) The ART2 paradigm in neural networks is capable of automatically grouping parts into part families. 2) The number of groups increased with an increase in vigilance of the system. The number of groups also decreased with a decrease in vigilance. 3) The order in which the parts are input to the system has no effect on the system performance.

Thus the overall conclusion of this thesis is that neural networks are indeed an appropriate tool to automatically group rotational parts into part families.