# 2
Ken Stevens

# DYNAMIC ARTICULATORY MODEL OF SPEECH PRODUCTION USING COMPUTER SIMULATION

by

WILLIAM L. HENKE

B.S., University of Wisconsin
1961
S.M., Massachusetts Institute of Technology
1962

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September, 1966

ature of Author William L. Henke
Department of Electrical Engineering, September 1966

Certified by Kenneth N. Stevens
Thesis Supervisor

Accepted by ________________________________
Chairman, Departmental Committee on Graduate Students

# DYNAMIC ARTICULATORY MODEL OF SPEECH PRODUCTION USING COMPUTER SIMULATION

by

WILLIAM L. HENKE

Submitted to the Department of Electrical Engineering on September 1, 1966, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

## ABSTRACT

A dynamic articulatory model of speech production is described. From a phonemic input the model generates a description of the configuration of the articulatory mechanism in the midsagittal plane.

Positions, shapes, velocities, and other descriptive features of the modeled vocal mechanism are contained in the "state" of the model. "Operators" act as agents for modifying the state by trying to manipulate aspects of the state toward abstract "goals" which are associated with phonemes. Goals are only changed discretely in time, and in this way the desired transformation from a discrete phonemic input to a continuous articulatory output is accomplished. The operator-state bifurcation of the model allows some of the natural constraints of the real vocal mechanism to be included similarly in the model. The model exhibits coarticulation effects attributable to phonemes preceding the "current" phoneme since the state configurative position responds only slowly to the goal directed operators owing to physical and physiological limitations. Coarticulation effects attributable to following or future phonemes result from a "look ahead" procedure that may invoke goals of future phonemes when such goals do not conflict with the goals of the current or more immediate phonemes. Thus anticipatory coarticulation results from a mechanism at a higher level than the sluggish response which causes post coarticulation.

The repertoire of speech sound types in the present model includes only vowels and stops, but it is felt that the general methodologies are applicable to all speech sounds.

The model has been implemented as a simulation on a large time-shared computer system, and extensive use was made of online graphical input/output during the evolution of the model. The source of real system data for model evaluation was a cineradiographic film of a speaker, and an online retrieval and display system for X-ray tracings was included in the modeling system.

A new algorithm for calculating the acoustic transfer function of the vocal tract is included as an appendix.

Thesis Supervisor: Kenneth N. Stevens
Title: Professor of Electrical Engineering

# ACKNOWLEDGMENTS

The author expresses his gratitude to his thesis supervisor Professor Kenneth Stevens who was alternately patient and diligent as the need arose. His astute comments often served as a stimulant for the generation of new ideas. Thanks are also due to my thesis readers Professors Morris Halle and Jack Dennis. They suffered through an early draft of the manuscript and suggested some important changes in emphasis.

This work was supported in part by the U.S. Air Force Cambridge Research Laboratories, Office of Aerospace Research contract number AF19 (628)-5661. The work was performed in the Research Laboratory of Electronics at M.I.T. The computation facilities used were those of Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under the Office of Naval Research contract number Nonr-4102(01).

# TABLE OF CONTENTS

APPENDIX

# LISTS OF FIGURES AND TABLES

## Figures

## Table

# CHAPTER I

## INTRODUCTION

In this work we are endeavoring to model a natural system -- the process of human speech production -- in an effort to further our understanding of the process. Such a model may also have practical implications for the construction of speaking machines for both man-man and man-machine communications, but that is not our present primary motivation.

The activities involved in speech generation can be partially ordered into levels of a "speech chain" as follows. The highest level of the process is the origination of an idea which the conceiver wishes to communicate. The idea must then be given a grammatical structure, the study of which is in the domain of the linguist. The phonological aspect of language structure then determines the phonetic form of the signal. Phonologists have almost uniformly agreed that the signal at this level consists of a sequence of <u>discrete</u> units, and most suggest the phoneme for this "quantum" of language (cf. Halle, 1954). These discrete linguistic units must eventually be transformed into continuous control of all the articulatory structures, and these in turn execute the physical motions necessary to generate a <u>continuous</u> acoustic speech signal.

Here we are seeking to improve our understanding of the encoding of linguistic signals into acoustic signals through a study of the dynamic behavior of the vocal mechanism. The level being modeled is therefore primarily the articulatory. The input to the model is a sequence of discrete linguistic symbols, i.e., phonemes, and the output is a description of the state of the articulatory system at all instances in time. Thus we are chiefly concerned with that which may be described as a "dynamic articulatory model of speech production."

In discussing the bases of phonology Halle (1964) writes:

> In addition to viewing utterances as composed of phonemes, the phonemes themselves shall be regarded here as simultaneous actualizations of a set of attributes.

In that work Halle was espousing the so-called "distinctive feature" characterization of these sets of attributes. Our model includes a set of attributes for each phoneme, but it functions at a slightly less abstract level than that implied by the distinctive feature scheme, and these attributes are therefore of a somewhat different form. Some similarities will be evident, but no formal attempt is made to relate our model to that particular characterization of phonemes.

We have selected the articulatory domain of speech as our primary sphere of interest, and a few comments on this choice are in order since in the past the major emphasis of speech research has been in the acoustic domain.

Although an acoustic description of speech is at first glance simpler than an articulatory description because of the relatively limited number of acoustic parameters, it suffers from its apparent inability to model the dynamics of speech except by the use of many relatively ad hoc rules or stratagems which are required for the mutual accommodation between successive elements or segments of the acoustic signal. An articulatory domain model of speech production should be able to incorporate the transitional features of multisyllabic speech utterances in a much more natural, simple, and elegant manner than is possible in the acoustic domain (cf. Cooper et al, 1962).

Modeling in the articulatory domain has permitted us to place an emphasis on naturalness, i.e., model functions correspond closely to similar functions in the real system. A side result of this approach is that many of the complexities of mutual accommodation alluded to above can become more of an implicit result of "natural" constraints in the model rather than an explicit part of the model. The manner by which coarticulation is included in our model is a good example of this.

Another motivation for selecting the articulatory domain is the view that the speech production mechanism (articulatory capabilities) has been a contributing factor in the development of speech sound systems. For example, Peterson (1966, p. 7) has recently stated:

> There is considerable reason to believe that the phonological aspects of speech are primarily organized in terms of the possibilities and constraints of the motor mechanism with which speech is produced.

For reasons presented below it was felt that such a study could be significantly aided by the use of a large scale time-shared computing system with advanced graphical man-machine interaction capabilities. Thus a secondary goal of this work has been to develop computing techniques and subsystems to aid the primary investigation. Throughout the text statememts of the type "observe the model" or "compare with data" generally imply that the investigator is seated at a display console and that the observations are principally visual.

(Our model is implemented as a computer simulation written in the AED language on the MIT Project MAC time-shared computer system. It uses the ESL (Electronic Systems Lab) display console connected to the MAC system for graphical input/output.)

As a supplement to this written document a motion picture film has been produced which demonstrates both the model and the man-machine graphical communication techniques developed in support of the model. The model and the communication techniques are both spatial and dynamic in character and thus cannot be adequately demonstrated in only the written medium. Portions of the film were generated on a frame by frame basis, so that by animation they show dynamics which could not be observed "live", i.e., online in the available time-shared computer environment.

The exposition of the model in the following chapters progresses from basic general notions and hypotheses to a description of how these manifest themselves more specifically in what we refer to as the "present implementation of the model" (or just "present model"). We consider these general notions and "basic methodology" of the model to be more "permanent" than the specifics of our implementation in a limited domain. Nevertheless, although the specific methods may be more prone to change, we feel that this in no way diminishes their importance as a part of the model. We use the word "present" as a reminder of the evolutionary nature of the development of the model.

A few specific details of our implementation will help orient the reader during the initial more general discussion of the model. The present model is limited to a midsagittal plane description of the articulatory structures. Our primary data are X-ray pictures of the vocal tract, in this plane, of a human speaker. Fig 1.1 is a photograph of a computer display of one frame of our data.

The present model manipulates three movable nonrigid anatomical parts, the tongue and two lips, and one movable rigid part, the mandible. Fig. 1.2 is a display of the model during the simulation of an input phoneme sequence. The "streamers" seen on the tongue and lower lip are velocity vectors which show the velocity of those parts at that particular instant of simulation time.

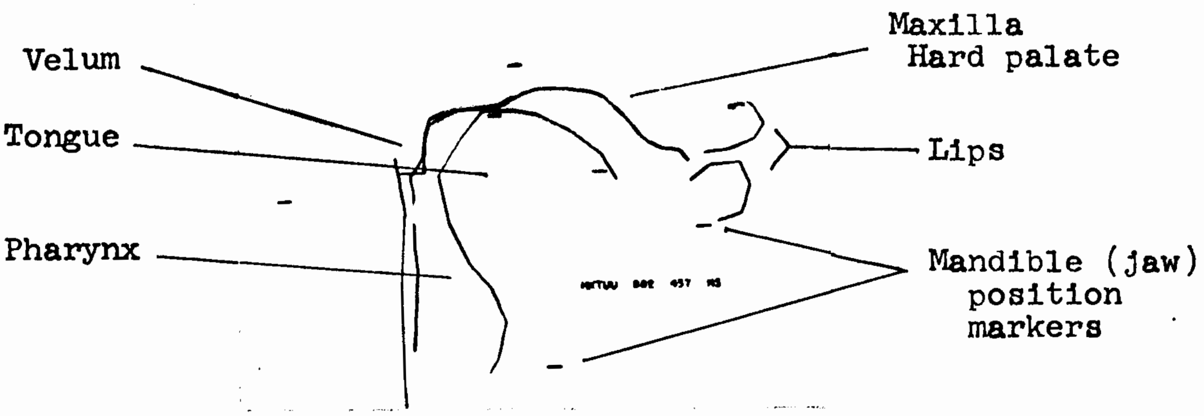


Fig 1.1 Midsagittal plane X-ray tracing. Subject producing the high back rounded vowel /u/.

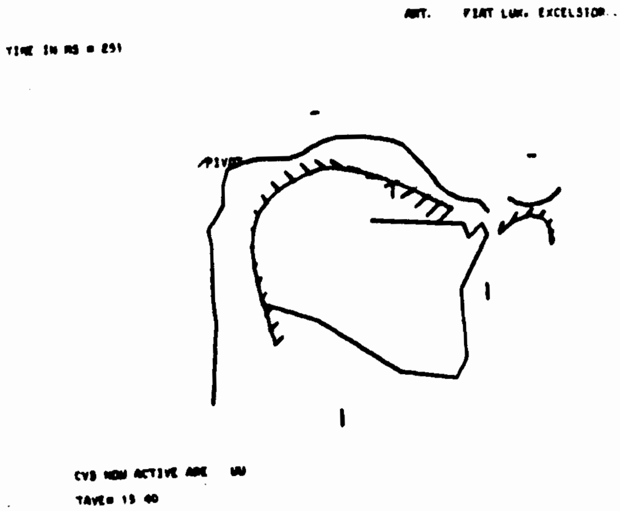


Fig 1.2 View of the dynamic articulatory model of speech production during a simulation.
Sequence and location therein shown below.

/ t u /
↑

# CHAPTER II

## DESIRABILITY OF COMPUTER SIMULATION FOR ARTICULATORY MODELS

Before discussing our particular model let us consider some of the general objectives and methodological questions which should form the basis of any attempt to formulate a model for a real system.

The purpose of developing a model of a process is to gain insight and understanding of the operation of the process.

> The continued improvement of models is, in fact, the central purpose of scholarship. To a considerable extent, the advancement of knowledge consists of the discovery of improved models. (Peterson, 1966, p. 11)

Criteria for judging the goodness of a model or theory are manifold. Peterson (1966) suggests that there are four essential requirements of an effective model of a real system, viz., (1) applicability, (2) completeness, (3) consistency, and (4) simplicity.

In endeavoring to apply such criteria it seems to us that some type of formal algorithmic statement of the model is mandatory. We shall refer to such a statement, and subsequent "solution" if necessary, as implementation of the model. Implementation of a model should be synonymous with the model itself, and it thus should not be necessary to have to consider it at all. This is a foregone conclusion in the "hard" sciences such as physics where almost by definition a theory is a well defined algorithm for calculating or explaining the behavior of the real system being modeled. We cite Newtonian mechanics and gravitation as the classical example of this. The theory predicted positions of celestial bodies, and was validated by comparison with observed data.

However, in the "softer" sciences including speech the terms model and theory appear to have a much less rigorous implication. Discussions about models are sometimes couched only in general "philosophic" terms without supporting specific and quantitative statememts. (This methodology is popularly known as "hand waving.") For this reason we have introduced the term implementation to imply a specificity often lacking in such philosophizing.

The test of any model is how well does it explain and/or describe that process which it proclaims to model. Thus it must be "implemented" so that its behavior or "outputs" for given inputs can be observed and compared with the real system. An unimplemented model cannot be so tested and thus seems quite empty of purpose.

At the articulatory level the speech production process is quite complex, both in terms of physiological and psychological "explanations" and at a more mundane level of simply describing the physical state of the articulatory structures. The descriptive complexity is due to the fact that the system cannot be characterized in terms of a relatively few parameters like the point masses and coordinates of classical mechanics or the wave functions of quantum mechanics.

Because of these both logical and descriptive complexities of the speech production process it seems to us that the only feasible way that models of the process can be implemented is by simulating them on a digital computer.

Many additional benefits accrue from a computer simulation. The requirement that actual programs be written becomes a strong and valuable disciplinarian. It forces a definition of the problem and demands the consideration in depth of many questions that could, and probably would, otherwise be glibly dismissed or not even realized to exist. In so doing a simulation may also suggest new areas of study, new ways of looking at and interpreting presently available data, and the need for new types of data.

Model simulation with sophisticated man-machine communication facilities (both hardware and software) is also a significant aid to creativity. Working with a model, in this case observing and modifying a simulated vocal mechanism on a computer driven graphical display, teaches and suggests through actual experience with the model. It is impossible to document those ideas specifically suggested by a simulation but the adage that "experience is the best teacher" is as true here as anywhere. Also many programming techniques from diverse fields in the computer applications literature have suggested relevant methods and ideas.

Model formulation is, here as in any field, an iterative process. One does something one way, observes the result, and then hopes for an inspiration to improve the method and continue the evolution of the model. The primary "results" of this work are many of the hypotheses and methodologies of our model of speech production, and they are the result of this type of iterative process.

One can no longer talk only in terms of hand waving generalities since the various subfunctions of a model must be programmed to operate in specific ways. Often we may not be particularly happy about some specific method, but we feel that some method is much preferable to no method since some method will most likely evolve into a better method whereas nothing usually only begets more of the same.

# CHAPTER III

## DESCRIPTION OF THE MODEL

### INTRODUCTION

What are some of the gross phenomena observed in speech that a model of speech production must implicitly be concerned with? Some phonemes are characterized by steady state sounds, e.g., the sounds of vowels spoken slowly reach a steady state which of course is different for each vowel. The sound patterns associated with some other phonemes, the stops for example, are more dynamic in character. Acoustic studies show that in connected speech the continuous sound cannot be segmented so that the sound of each segment is attributable to only one phoneme. Rather there are dynamic and/or contextual effects of the types known variously as coarticulation, contextual assimilation, and centralization.

In a paper concerned with the dynamic aspects of vowel (only) production, as studied using acoustic techniques, Lindblom (1963) concludes and goes on to comment:

> ... the talker's "intention" that underlies the pronunciation of the vowel is always the same, independent of contextual circumstances. A vowel target appears to represent some physiological invariance. The present data support the assumption that the control that the talker exercises over his speech organs in vowel articulation is associated with neural events that are in a one-to-one correspondence with linguistic categories. Let it be further assumed that an utterance is a sequence of such events that serve to trigger the appropriate articulatory activity. Articulators respond to control signals not in a stepwise fashion but smoothly and fairly slowly, owing

> to intrinsic physiological constraints. Since the speed of articulatory movement is thus limited, the extent to which articulators reach their target positions depends on the relative timing of the excitation signals. If the signals are far apart in time, the response may become stationary at individual targets. If, on the other hand, instructions occur in close temporal succession, the system may be responding to several signals simultaneously and the result is coarticulation.

We have quoted these general and fairly prevalent observations and speculations because they state several notions which we regard as fundamental to speech production and hence feel should be quantitatively formulated and included in a model of the process. If in the above quote "vowel" is generalized to any phoneme and "target" is accordingly generalized also, all these notions are included in our model.

Thus one hypothesis of our model is that there exists a set of higher level "invariances" which are associated with the phonemes. The input to the model is a sequence of phonemes, and these are thus viewed as references to appropriate invariances. A lower level of the model is the physical mechanism, i.e., the articulatory structures which respond to the "application" of these invariances. In accordance with observational data, the response must be strongly influenced by both the arrangement and timing of the input sequence. Thus the model must be concerned with both the form and content of these invariances, and with the "responding properties" of the physical structures.

The methodology of our dynamic articulatory model is quite different from that of other approaches to the same problem. More prevalent thinking might be summarized as follows. There exists a dictionary of a small number of complete static configurations of the vocal tract. Pseudo dynamic action is obtained by a process of blending (usually in a linear combination) various static configurations, the blending being a function of time and position along the vocal tract (cf. Ohman, 1964).

In contrast to this scheme, a basic hypothesis of our present model is that at any instant of time any particular portion (not necessarily all) of the articulatory mechanism is trying to achieve a "goal" which is part of one and only one of the invariances, i.e., there is no higher level "blending." MacNeilage and Sholes (1964, p. 231) phrase much the same distinction between these two possible schemes or hypotheses when they question whether coarticulation is a "reorganization of action" (blending), or a "combination of discrete articulatory units" (our model).

This is an example of one of the more specific initial hypotheses that one is forced to make when implementing a model. Our scheme was arrived at mainly by considerations of simplicity inasmuch as we were unable to think of any plausible scheme to accomplish "reorganization." In addition we were guided by our own purely subjective feelings of "naturalness." Experience with our model incorporating this hypothesis has not dissuaded us from continued usage of the same. Other investigators are trying to interpret electromyographic data which they feel are relevant to the same question, but no definitive conclusions have yet been reached.

Another major area of concern must be the organization and control of a timing "program". Timing may be solely determined at higher levels, or it may also be influenced by the response of the lower level structures. Prevailing theories with regard to timing are less advanced, and consequently those aspects of our model which pertain to timing are currently rather primitive and unrefined. In the present model timing is determined by information derived both from higher level inputs and by "feedback" from lower level structures.

As we stated above, the function of modeling is to raise significant questions, help suggest answers for these questions, and evaluate the answers by comparison with the real system. The model presented here both performs these functions and is the result of them. It does not purport to be complete, but is hopefully a step in the direction of improved models of speech production.

## BASIC METHODOLOGY -- OPERATOR-STATE DICHOTOMY

A salient feature of our modeling methodology is a division of the model into two intrinsically different types of entities. One is what we shall call the "state" and the other an "operator".

The state of the system (articulatory mechanism) at any time is fully contained in the values of state variables.

Several types of quantities may be described by state variables, e.g., positions, velocities, forces, masses, angles, and air pressures. (The state structure of the present implementation of the model is subdivided into the tongue, the upper lip, the lower lip, and the mandible. Here we mean to imply that the something in the model "corresponds to" the named something in the real system. In much of the following discussion such correspondence is implied, but the phrase "corresponds to" will often be omitted for brevity.)

There are really two related aspects of what we call the state. One aspect pertains to structure or form, i.e., types of items (state variables) and their interrelationships, and the other aspect pertains to values of these same items. When we speak of just "the state" we are usually referring only to the second aspect -- the composite of the values of the state variables at a given instant of time, or a "snapshot" of the system. (In this discussion "time" always means model time rather than real time.) "State trajectory" implies the information that would be contained in a sequence of snapshots in time. The state varies with time, but the state structure is time-invariant for any particular model.

Operators are algorithms, sometimes with associated data, which "operate" on and/or refer to the state variables. The only way the values of the state variables may be changed is as the result of the application of an operator.

## GOAL DIRECTED VOLITIONAL OPERATORS

The primary control mechanism of the model is a type of operator which we shall call a "volitional operator." We call them volitional since they are responsible for willful effects upon the state, and hence perform a "higher level" function than do nonvolitional types of operators. The functioning and control of volitional operators is the crux of the entire model, and is the subject of much of the remainder of this chapter. Other types of operators perform functions such as "housekeeping" and information gathering, and are of no concern here.

A volitional operator functions in a goal directed manner, i.e., it operates on the state in a manner that endeavors to cause some of the state variables to approach a "goal". Thus a volitional operator is characterized by a manner (how) and a goal (what). We consider a volitional operator to correspond to an "intention", in some sense, of the speaker.

Intentions may never be completely fulfilled, however. Consider someone trying to simultaneously grasp two objects 10 meters apart. The intention is to grasp the objects, and the hands start approaching the objects. But before the intention is completely fulfilled a constraint, the limited length of arms, dominates the action. Motion will eventually cease with the goal unattained, though the person may continue to strain in the directions of the objects. Or consider the intention to strike at something. If this intention is "canceled" before the object is actually struck the result is an abortive swing. These and similar thought experiments suggest that the responses to intentions are strongly affected by many kinds of environmental circumstances.

The degree to which a goal is ever attained in our model, as manifested in the state trajectory, is therefore conditioned by environmental circumstances. These include the state when a volitional operator is initiated, "physical" constraints and conflicting goals, and the time span associated with the operation.

Some important differences between goal attributes and the state trajectory are as follows. Due to "physical" constraints state variables can only assume physically realizable and compatible values, but such is not a requirement for the goals since they are only abstract "intentions." Also, the state trajectory traverses a continuum in both space and time, but the goals are characterized by a certain discreteness in the same dimensions. This will be developed more fully below.

More specifically, several of the dynamic and contextual effects observed in connected speech are hypothesized to be nonvolitional, and consequently they should be an implicit rather than an explicit part of the model. This aim was the primary motivation for the operator-state dichotomy of the model.

As an example, consider contextual effects on vowel production. It is hypothesized that a speaker's intention underlying his pronunciation of a vowel is invariant, i,e., independent of contextual circumstances. Vowel production is thus modeled by always applying the same operator(s) for a given vowel phoneme. However, since the state will undoubtedly differ at different onsets of the operator(s), and also since the duration of the "on" time(s) of the operator(s) may vary, the state trajectory (and thus the resultant acoustic outcome) will indeed be a function of contextual circumstances. A short (in time duration) application of a given vowel volitional operator will only have a small effect on the state. An infinite application of a vowel operator in the absence of any other volitional operator will result in the state eventually reaching the so-called target configuration.

We digress to note that we are not being particularly careful here to distinguish between our particular model and observations of and hypotheses about that which is being modeled, i.e., the real system. Since the point of the model is to embody our conceptions of the real system, there should be no essential "higher level" differences between the model and our notions concerning the real system.

As an example of a "physical" constraint, consider the production of an alveolar stop (/t,d/). In order to effect "solid" closure the model uses as a goal for the tongue tip a type of "target" of which a part is above the hard palate.

This goal is physically unrealizable, and thus the state can never completely attain it. In the process of seeking this goal the state "tongue tip" will encounter a "barrier," the hard palate, and will be restrained from further progress toward the goal. However, the goal seeking mechanism will temporarily continue to "push," in effect, against the palate. (One might prefer to think of this aspect of the alveolar stop goal as being contact and pressure against the hard palate. We consider the distinction between these two points of view to be relatively insignificant since both yield identical macroscopic "outputs," i.e., state trajectory of macroscopic components.)

## LOWEST LEVEL OF MODEL ACTION SLIGHTLY ABOVE THE LEVEL OF INDIVIDUAL MUSCLES

The basic volitional operators of our model are concerned with the activities of various anatomical structures (or portions thereof) of the vocal mechanism (tongue, lower lip, tongue tip, etc.), and their primary effect is to produce goal directed activity in that part of the state associated with the structure(s) of interest. They "operate" in the domain of movements and positions rather that at the lower level of individual muscle activity. We might rephrase this by saying that the lowest level of concern is an "intended result" of muscle activity rather than the muscle activity itself. There are several reasons for choosing this as the lowest level of the model.

We are primarily interested in studying the the speech process, and are not interested in muscle action per se. Although movements of articulatory structures must eventually result from the excitation of specific sets of muscles, it seems reasonable to hypothesize that the higher levels of the speech production chain in the central nervous system are not concerned with particular muscle sets but rather with functions of muscle action, e.g., movements and positions, since as Heffner states (1950, p 39):

> No one can at will cause a given muscle to contract; movements, not muscles, are the units of neuromuscular behavior.

We acknowledge that somewhere in the human nervous system a transformation from higher level "neural commands" to lower level to patterns of excitation for individual muscles must occur, but we suggest that speech production can be fruitfully studied via a modeling approach without reference to such microscopic muscular activity. Our model does not purport to make any statements about such microscopic activity.

For example, in the production of the phoneme /t/ the tongue tip ascends toward the hard palate. This action is probably due to muscles located near the tongue tip, but for illustrative purposes it is not impossible to conceive of this desired action resulting from muscle activity occurring elsewhere in conjunction with "mechanical" transmission to the tongue tip. Since we focus attention only on the intended result, the physiological "origin" of an action is of reduced significance for our model.

More importantly, an articulatory model of speech production founded at the musculature level would have to begin in the domain of physiology, anatomy, and the mechanics of distributed, non-linear, coupled elastic bodies with distributed sources of stress and/or strain. While this may certainly be a worthy research area it is far removed from our sphere of interest. It appears to us that it will be a long, long time before any significant degree of modeling adequacy is achieved in this area at a level macroscopic enough to be of interest to any other than physiologists. Thus since we desired a functioning model relevant to the speech process pragmatic considerations made it impossible to found the model at the musculature level.

Our basic movement mechanisims must certainly reflect in their method of operation many of the characteristics and constraints of the physical system; but as long as we do not impute to these mechanisims any microscopic physical significance and they generate adequate model action at a more macroscopic level we have provided an adequate lowest level basis for our model. One of the major aspects of the present work was thus to devise basic operator mechanisms and the associated state structure which achieve this lowest level delimiting effect and yet yield, to a first approximation, many of the macroscopic effects of the physiological system and its constraints.

Recently several laboratories have begun investigating the speech production process using electromyographic techniques. A good example of this type of approach is found in the work of the Haskins Laboratories (cf. MacNeilage and Sholes, 1964). They are trying to make inferences about the basic motor organization of speech from observed electromyographic patterns. They first attempt to deduce from these patterns of myographic activity, in conjunction with relevant physical considerations, which particular muscles were acting and what the role of each muscle was. They then progress to inferences about motor organization based upon these patterns of individual muscle activity.

At first glance it might seem that our rejection of a musculature foundation for our model would also deny the usefulness of myographic data and inferences based thereon. Such is not the case, however, since our "goals" are the intended result of observed muscle activity. Two aspects of our configurative subtargets which will be discussed later -- the region of definition and the quasi-force function -- might have correlates in certain features of myographic activity, and it would be interesting to look for these.

## DISCRETENESS AND EVENTS

The organization of a higher level timing and sequencing "program" (i.e., physiological and psychological construct) is of fundamental importance in the speech production process. One of the primary aims of our dynamic model is the conversion of a discrete type of input to a continuous type of output.

Any particular volitional operator (including a goal) "operates" continuously on the state. If the operator status were never changed by a higher level control the model state would eventually reach a steady state. In steady state all remnants of the effects of past operators would usually have been completely "forgotten" by the state.

However, at what we shall call "event" times the current operator status is modified. The general notion is familiar in other contexts to other workers. Analogous terminology that might clarify the general notion would be the onset time of "motor commands" in other modeling schemes. We do not wish to imply, however, any more specific analogy with other models.

As we have stated, the basic dynamics of our model are effected by means of goal seeking mechanisms. Only at so-called event times are some phases of the goals replaced by newer values. The important point here is that these higher level changes in the goals occur discretely in time. That is, goals are not modified continuously but only "abruptly."

## CONFIGURATIVE GOALS

The only type of goal that the present model is concerned with is a mechanical configuration, or target, of the vocal tract. (Additional types of goals might pertain to the control of phonation, stress, air pressures at constrictions, etc. A primary consideration of the model development was to devise a conceptual and computational framework which was amenable to upwards evolution, and consequently these additional types of goals have implicitly influenced the present structural form of the model.)

At any instant of time the total or overall configurative goal or target may be a composite of "subtargets" for different portions of the articulatory mechanism. Associated with each subtarget and defined over the same anatomical region is a "quasi-force function" which effects the "manner" of production by influencing the "strength" and "rapidity" of the tendency toward the target. There will be times when there is no higher level configurative goal for portions of the articulatory structures. (The lowest level mechanism in the model which accounts for some of the physical-mechanical characteristics of the articulatory structures requires certain aspects of a target for its functioning. If there is no higher level goal a neutral "target" is used for this mechanism, but such a target is not considered to be a goal.)

A set of subtargets with quasi-force functions is a form of memory, and the members of this set are one form of the invariances that phonemes will refer to. In the present model there are subsets for the tongue, the upper lip, and the lower lip.

## POSITION AND SHAPE SEEKING

The lowest level operation in our model is that which causes the configurative state to seek these targets. For purposes of mechanization the spatially distributed articulatory structures are divided into many small adjacent segments. Any segment of these structures is continually seeking both an absolute *position* and a *shape*, i.e., a position relative to adjacent segments. It is sometimes convenient to view the "position seeking" operation as global and due to extrinsic musculature and the "shape seeking" operation as local and due to intrinsic musculature, but our strictures against overly fine physical interpretation must be borne in mind. Fig. 3.1 shows hypothetical situations of position and shape seeking.

The *position* seeking operation generates for each segment a quasi-force directed toward the current goal for that segment. The magnitude of this force is a function of the distance that the begment is from its target, and the magnitude of the current quasi-force function for the segment. If the segments of structure were independent of each other they would all migrate toward their respective targets. A segment which was far from its target would at first accelerate (mass-limited region of operation), then reach a velocity limit, and finally slow and stop as it

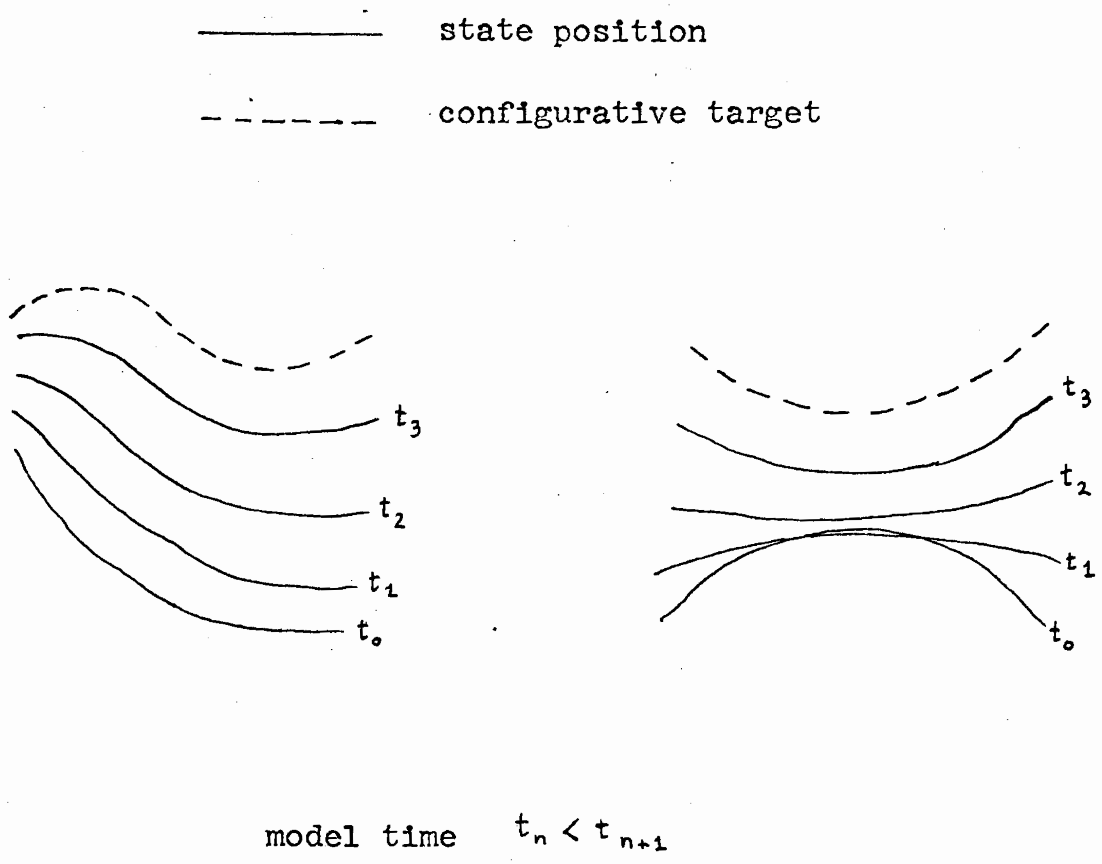


Fig. 3.1 Hypothetical state trajectories as effected by the position and shape seeking mechanism. Shape seeking functions independently of the location of the state position with respect to the configurative target. Note in the right hand example that a part of the structure temporarily regressed in the position sense since a large shape change was needed. The shape influence was effectively "stronger" than the position influence.

approached its target. Depending upon relative magnitudes of the quasi-force and the segment's quasi-mass and damping, a free segment might even overshoot its target and have to converge on it.

However, the individual segments are not free since they are attached to adjacent segments. These "physical adjacency constraints" necessitate what we call the "<u>shape seeking</u>" mechanism. We also refer to this function as the maintenance of structural integrity since without it adjacent segments would drift away from each other and this would correspond to a tissue rupture in the real system. The shape seeking operation causes the nonrigid anatomical structures of the model to try to assume a particular shape, independent of their present position.

If the real structures were passive the shape seeking mechanism could function using only time invariant physical attributes, e.g., axial and shear elastic moduli. However, due to the presence of active musculature in these structures the shape seeking mechanism must also be endowed with some higher level or volitional control, and must therefore be time varying. For example, during the production of the /k/ phoneme the central portion of the tongue (in a midsagittal section) tries to hump up wherever it may be.

Shape seeking in the model mechanism is also required since the composite positional target for a nonrigid structure may at times be discontinuous in space. This discontinuity results from the fact that the composite target for a single structure at any instant may be composed of subtargets from different "sources" for different regions of the structure. For example, the tongue tip may have a target which is of origin independent of the target for the pharynx region of the tongue, and at the point of transition between the two the targets may not be contiguous. The configurative state, however, can never be allowed to become discontinuous, and the shape seeking mechanism prevents such a catastrophic occurrence. In this aspect of its operation the shape seeking mechanism is nonvolitional and has no physical interpretation. It may be thought of as "spacially redistributing" higher level notions of control which for implementation we were forced to overly specify by referencing particular points of a structure rather than general regions.

The shape seeking mechanism (maintenance of structural integrity) also effects a type of operator precedence in space. If the quasi-forces being developed in one portion of the tongue are strong relative to those in another portion, the "pull" of the stronger forces will be partially propagated through the state structure to the regions of weaker forces. Thus, for example, while the target for the /t/ phoneme is defined only near the apex of the tongue, it is relatively strong so that the remainder of the tongue will be partially pulled up also if necessary.

The shape seeking operation is mechanized by the use of incremental targets which are set in the same way as and simultaneously with the positional targets. The mechanism operates by generating internal or intrinsic quasi-forces which are added to the extrinsic or target seeking quasi-forces.

As previously implied, there is also a "physical impenetrability" constraint mechanism. Relevant portions of the state must not be allowed to "penetrate barriers." However, the quasi-forces will "push" against the barrier. In the present model this mechanism is used between the tongue and the hard palate, and between the upper and lower lips.

Additional details of the method of position and shape seeking can be found in Chapter 5.

Although not included in the present model, the mechanism was designed so that additional forces due to air pressure could be included here. There is some indication (Perkell, 1965) that, particularly in the pharyngeal region, air pressures may affect positions of articulatory structures.

## MANDIBLE POSITION

Since the position of the mandible affects the position of the tongue and lower lip, it at first glance seems plausible to try to directly control the model's mandible from the phonemic input. The mandible position would then affect the tongue and lower lip positions. Our model, however, operates with the direction of causality just reversed from this. We view the positions of the tongue and lower lip as the entities of primary concern. The mandible action is in a sense a derivative action since it must be positioned in a way that will physically enable the primary articulators to execute their intended actions.

Certainly the mandible is not physically just pulled around by the tongue and lips, but rather is controlled by direct activation of specific muscles. But, in accordance with previous statements, this is considered to happen in the real system at a level lower than that at which our model has any physical interpretation. Thus for our model we need not consider higher level or volitional control of the mandible, but need instead devise some lower level method by which the mandible "follows", in a general sense, the primary articulators. Some of the physical constraints between the tongue, the lower lip, and the mandible should be accounted for in the model, however. For example, when the lower lip is not actively "involved" in the production of a sound (i.e., not for bilabial stops, rounded vowels, etc.) its position is primarily determined by the mandible position. (When the lower lip is involved it should have an effect upon the mandible position which in turn should affect the tongue position. These constraints have not yet been implemented, however.)

## CONFIGURATIVE MEMORY (PS's)

We turn now to a more specific description of that part of the model's "memory" which contains position and shape "target" information. For each of the individual anatomical structures of interest there is a set of configurative targets and associated quasi-force functions. Any particular target in the set may be specified for the whole or any fractional part of the structure, and the quasi-force function is defined for the same region. Thus a target pertaining only to the tip of the tongue, for example, may be specified and need not be defined for the other regions of the tongue. Since each of these low level memory abstractions contains information about the position, shape, and relative "strength" for all or part of a single articulatory structure they will subsequently be referred to by a mnemonic as PS's for Part Position, Shape, and Strength. Composites of these PS's serve as the data used by the position and shape seeking mechanism.

In this memory there are no associations with particular phonemes. Thus the model could in theory use these targets in conjunction with the basic configurative goal seeking mechanism to generate movements of structure without higher level phonological associations (which are discussed in the next section). The real system correspondence would be movement of the articulatory structures without any speech implications.

Discovering appropriate positions and strengths of these PS's for use in speech production is a part of the learning process of the model. We, as the model builder, are still in the learning loop here, and a subsystem of programs is dedicated to the online graphical input and modification of these spatial abstractions.

Fig. 3.2 shows two PS's. One PS is the "current" part and was drawn in and can be modified (including the quasi-force function) using the light pen with the tracking cross (seen near the right side of the figure) and push button commands. The data that are entered and modified are individual "flesh point" locations, as indicated by the virgules (/). Additional PS's can be displayed for comparisons, and one more is shown here. Both these PS's are for the tongue (part number PSPN = 0), but have different regions of definition.

Fig. 3.3 contains an additional PS (tongue tip) and one frame of X-ray data which can also be used for visual reference.

Fig. 3.4 is of larger scale to show additional detail.

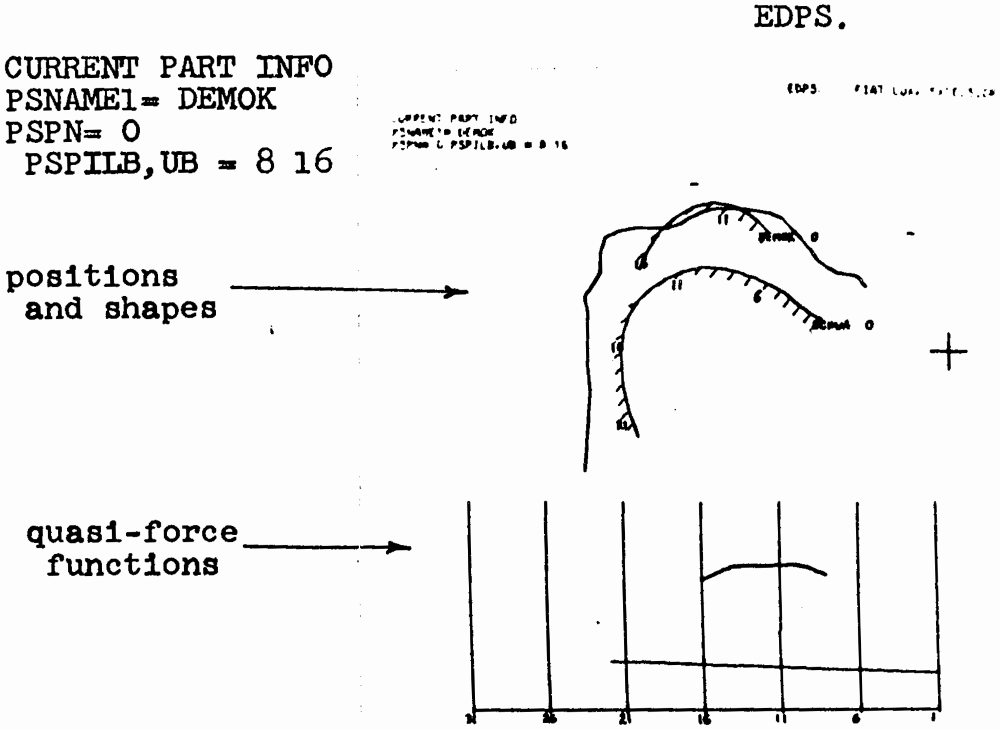


**Fig.** 3.2 Two PS's.

Note that the PS "DEMOK 0" (which might be used for a velar stop) is defined for only part of the tongue, and that its quasi-force function is much "stronger" than that of the PS "SCHWA 0" (which is used for the unstressed schwa vowel). Note also that part of the "DEMOK 0" PS position is above the hard palate, and hence unrealizable. The quasi-force function abscissa is the flesh point index, i.e., the numbers shown on the position curves.

EDPS.

```
CURRENT PART INFO
PSNAME1= DEMOK
PSPN= 0
PSPILB,UB = 8 16
```

HXKE 123 2829 MS

(X-ray frame identification)

region of definition

structural part number (0 = tongue)

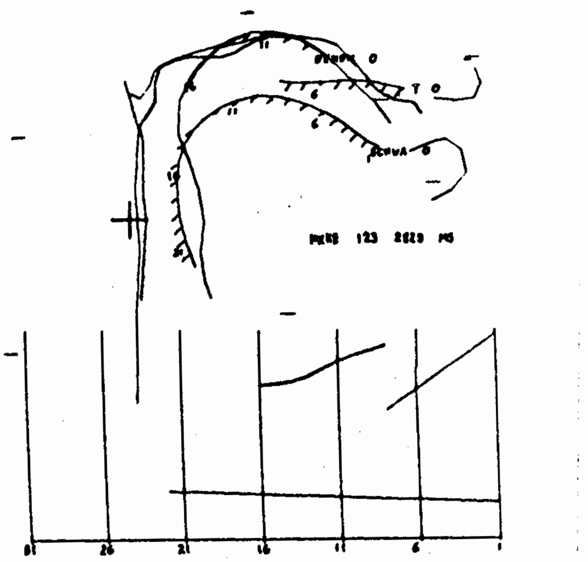
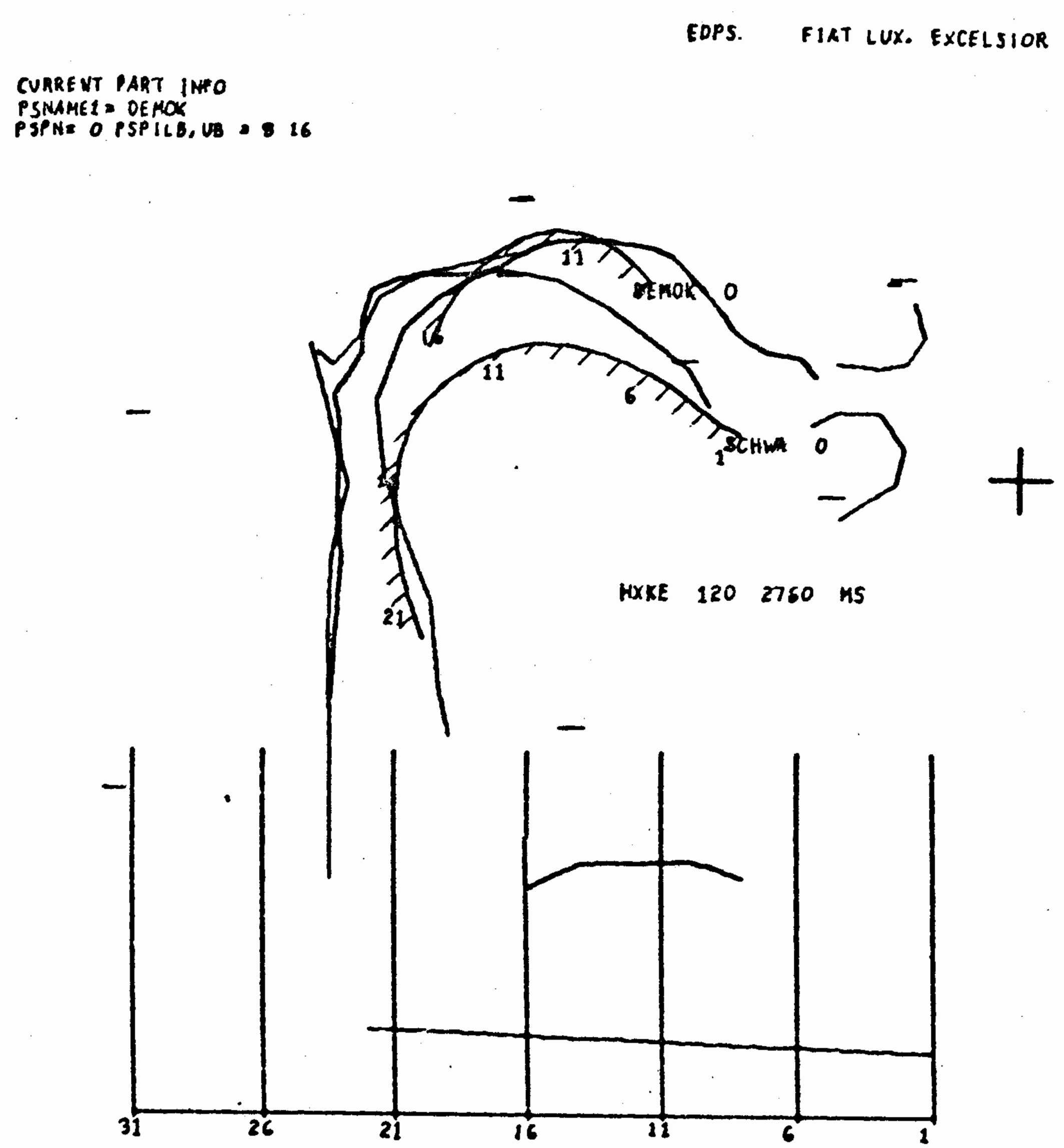


Fig. 3.3 Three PS's and an X-ray tracing.

The tracing is from the /k/ portion of the word /həˈkɛ/.



```
EDPS.    FIAT LUX. EXCELSIOR

CURRENT PART INFO
PSNAME1= DEMOK
PSPN= 0 PSPILB,UB = 8 16
```

Fig. 3.4 Two tongue PS's and an X-ray tracing.

CVTAB (PHONEME TABLE)

A higher level memory in the model associates these fractional targets (PS's) with phonemes. We call this level of memory the CVTAB for "compound volitional operator table." The CVTAB contains references to particular PS's for use by the position and shape seeking mechanism, and compounds those data with additional data into descriptions of phonemes. We include here an actual CVTAB which has been shortened to contain only enough data to illustrate several important points about the model.

```
1 ALLPLX 0 .05       ALL PARTS LAX, USED IN INITIALIZATION
 2 SCHWA 0 0 .05
 2 SCHWA 1 0 .05
 2 SCHWA 2 0 .05
1 AH 0 .140      AH, TYPE 0
 2 AH 0 0 .05      TONGUE ONLY, NO RELEASE
1  EE 0 .15
 2 EE 0 0 .045     ONLY TONGUE, NO SPREADING OF LIPS YET
1 SCHWA 0 .10
 2 SCHWA 0 0 .06      RELAXED LIPS
1 UU 0 .15
 2 UU 0 0 .047
 2 UU 1 1 .043    UPPER LIP
 2 UU 2 1 .045   LOWER LIP, RELEASE
1 T 1 .049      TYPE 1, STOP
 2 TD 0 1 .04    TONGUE ONLY, RELEASE
1 P 1 .055       STOP TYPE
 2 PB 1 1 .04     ONLY INTERESTED IN THE LIPS, WITH RELEASE
 2 PB 2 1 .04    AND THE LOWER LIP
1 B 1 .05
 2 PB 1 1 .05  USES LIP ACTION COMMON WITH P
 2 PB 2 1 .05
1 K 1 .052
 2 KG 0 1 .05 CENTER OF TONGUE ONLY, WITH RELEASE, NO LIPS
```

Table 3.1  Illustrative CVTAB.
Phoneme Description Table.

We will discuss here those entries in the CVTAB whose functions can be briefly described, and "name" those entries which require more extensive discussion and return to them in later sections.

Lines prefaced by a "1" are phoneme entries in the CVTAB. The second item of these lines is the name of the phoneme (in our machine type font transliteration of a phonetic alphabet), and the third item is its "dynamic type" (see next section; type 0 = vowel, 1 = stop). The fourth item is a time duration parameter interpreted differently by each dynamic type.

Following each phoneme entry (type 1 line) and up to the next next phoneme entry are various "subcommands" to be associated with the preceding phoneme. All type 2 lines refer to a particular PS. (In a more sophisticated model subcommands of additional types might pertain to voicing, stress, etc.).

The second and third items in type 2 lines are an arbitrary name and anatomical part number, respectively, of the PS being requested. (Part numbers are: 0 = tongue, 1 = upper lip, 2 = lower lip)

The fourth item of a type 2 line specifies whether or not the action of the articulatory region affected by the line, when its phoneme (preceding type 1 line) is being executed, is characterized by a "release". This dichotomous feature is usually true for stops and false for vowels. The release specification essentially causes the stop to be released in approximately the same manner as its attack. If release is not specified the model will just progress along the input phoneme sequence with no particular emphasis on a "release" from the present phoneme. In the illustrative CVTAB it will be noted that "release" is specified for the lips of /UU/, but not for the tongue.

The fifth item of a type 2 line is a "muscle delay" time duration parameter. Configurative targets change instantaneously in time, but the quasi-forces which are generated for seeking targets do not change so abruptly. The previous quasi-forces are "faded" to new quasi-forces over a time given by this parameter (in seconds). More specific details will be found in Chapter 5. (Of course, there is further delay in the system due to purely mechanical inertial lag of response after applied force.)

An important point to note about the CVTAB is that for each phoneme only relevant structures need be explicitly operated upon. /AH/ (low back vowel), for example, only requires a reference to the tongue, whereas /UU/ (high back rounded vowel) references the tongue and the lips. /T/ references only the tongue, and if we examined the lower level configurative target specified by the only line type 2 entry for /T/ we would find that this pertains to the front of the tongue only. Thus for /T/ goals for the back of the tongue and lips are not specified. Likewise for /P/ only the lips are specified. This requirement of only having to specify items of interest should be contrasted with other types of descriptive schemes where a complete set of the same entries in a table must be stated for all phonemes.

Also compare the CVTAB entries for /B/ and /P/. These are both bilabial stops but differ in voicing, being voiced and voiceless respectively. Thus insofar as the the lips are concerned these phonemes are, to a first approximation, the same and this is reflected in the CVTAB since the same target abstractions (PS's) for the lips are specified for both. The present model does not include voicing commands, so this distinction is not displayed in the CVTAB.

This usage of a single PS by more than one phoneme is a common occurrence. The most obvious examples are the voiced-unvoiced stop pairs of /b-p/, /d-t/, and /g-k/.

## CONTROL, TIMING, AND COARTICULATION

Let us recapitulate briefly what we have already described about the model. There is a state of the articulatory mechanism, and the state variables traverse a continuum in space and time. Operators try to manipulate the state toward goals, and are the only mechanism by which the values of the state variables can be changed. The goals (abstractions, intentions) are members of a finite set, and they are changed not continuously but discretely or abruptly in time. Spatial goals and other data are associated together in the CVTAB into a kind of phonemic goal. The last major function we need to complete the present model is a timing and executive mechanism which controls the sequencing of lower level goals in accordance with the phonemic input sequence.

At any given time there is one dominant or "current" phoneme, and associated with the "dynamic type" of that phoneme is a control program which is the temporary Lord High Master. It is responsible for timing and control functions during its "reign", and its own dismissal, i.e., the passing of control to the next phoneme in the input sequence. The present model has two dynamic types -- vowel and stop -- and for each dynamic type there is a small control program (CVPROG's for CV Programs).

Before discussing the specifics of the two present CVPROG's let us consider the general philosophy and possible functioning of these control programs. They sometimes "look ahead" from the current or dominant phoneme to future phonemes so the model can begin to anticipate these future phonemes where they do not conflict with more immediate phonemes. A restatememt of this notion is that more immediate requirements or goals have precedence over future goals, but where future goals are not inhibited by more immediate goals the future goals may begin to have an influence.

There is never any type of "look back", however. As soon as a phoneme dismisses itself it is totally forgotten at higher levels. Its effects will temporarily remain, however, manifested in the low level state description. The effects will die out as the state trajectory is influenced by the more recent phonemes. Thus one might consider that the state concept has provided a temporary form of memory, but we do not see any benefits accruing from this point of view.

As we previously stated, a fundamental requirement of a model is that it provide for coarticulation effects such as those reported in spectrographic measurements by Ohman (1966). In our model pre and post, or forward and backward, coarticulation result from two distinctly different phenomena. Backward effects are due solely to noninstantaneous response of the state, i.e., physical inertia, "muscle delay", etc. On the other hand, forward effects are due entirely to a higher level look ahead or anticipation. By these methods we feel that coarticulation has been modeled both naturally and simply.

It is interesting to note that English phonology seems to have "look ahead" but not "look back" types of rules. For example, the duration of a vowel preceding a voiceless consonant is shortened relative to the same vowel before a voiced consonant.

## TIME PROGRAMS FOR STOPS AND VOWELS

The two dynamic type control programs (CVPROG's) of the present model are STPCNS for stops and VWLPRG for vowels. It is suggested that these actual program listings included in Appendix A of this report be consulted during the reading of the following text.

The stop control program (STPCNS) functions as follows: When initiated it sets all the goals associated with the particular stop and leaves all other goals intact. It then waits until contact between the structures involved with the stop is attained. When contact is attained it looks ahead or previews the next phoneme and sets goals for it except where inhibited by the current stop. That is, all the goals associated with the next phoneme are invoked except where they would overwrite a goal associated with the current phoneme. At the same time it forgets all past goals, i.e., all goals antecedent to the current stop. It then waits a short time interval, determined as discussed below, and then dismisses itself by starting the next control program.

From observations of acoustic coarticulation data for VCV utterances (cf. Ohman, 1966) we originally felt that the phoneme following a stop consonant should be "previewed" earlier than just described. This yielded, however, excessive anticipation upon comparison with our radiographic data, and the more delayed look ahead better matched these data. However, our data were all of the form /hə'CV/ where the consonant was not preceded by a stressed vowel. We suspect that the earlier look ahead would probably be needed in the case of a preceding stressed vowel. We have seen that the model is so designed that this can be readily effected if real system data so indicate.

The vowel program (VWLPRG) sets all the goals for the particular vowel and then causes all antecedent goals to be forgotten. It then waits for a time interval and then transfers control to the next phoneme. Note that if a vowel is preceded by a stop some of the vowel goals will have already been set before the vowel becomes the current phoneme.

The fixed time intervals of both these programs are normally determined by data in the CVTAB, but they may be overridden by data supplied with the higher level phoneme input list. Thus the higher level input may specify explicitly that some time interval is to be of a particular duration, but if this information is not supplied the interval duration used will default to that stored in the CVTAB.

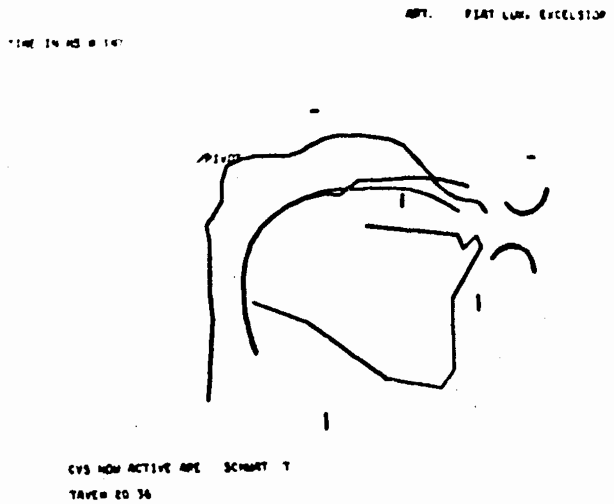
An important point to note in the stop program is that an action observed in the state was used as a trigger for higher level control. This can be viewed as a low level or proprioceptive feedback type of control. We will return to the topic of "feedback control" as viewed by psychologists in the concluding chapter.

## ILLUSTRATIVE SEQUENCE OF MODEL ACTION

In this section we include selected frames from the supplemental motion picture film which demonstrate the functioning of the model. The moving anatomical structures in the configurative state are the tongue, both lips, and the mandible. The time shown in the upper left hand corner is the simulation time. The relative time of the various frames should be noted. They are arranged chronologically but are not equally spaced.

The phoneme input sequence was /T/UU/P/AH/.

TIME IN MS = 147



CVS NOW ACTIVE ARE
SCHWA T

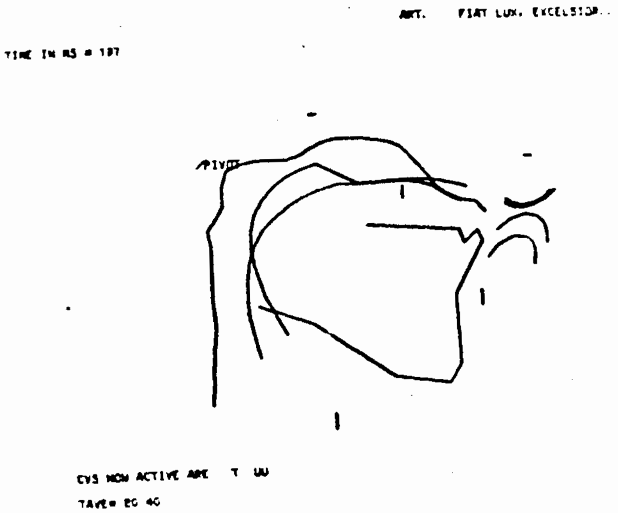
Showing state position and configurative target.

The /T/ target applies only to the front of the tongue.

TIME IN MS = 187

CVS NOW ACTIVE ARE

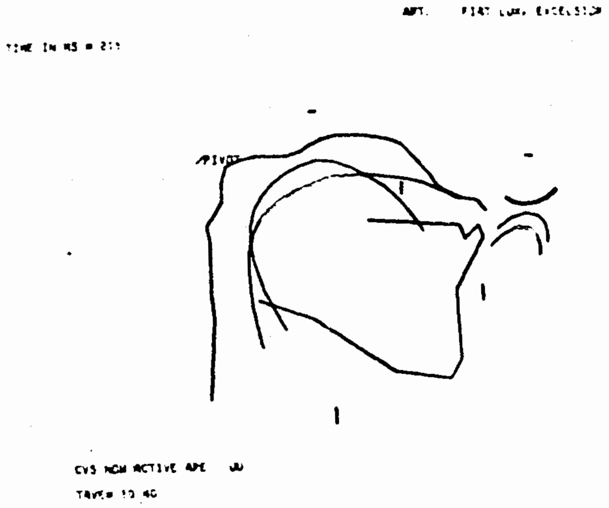
T UU



Showing state position and configurative target.

The tongue tip has hit the hard palate and is being prevented from continuing toward its target. /UU/ has been activated except where inhibited by the still active /T/.

TIME IN MS = 219



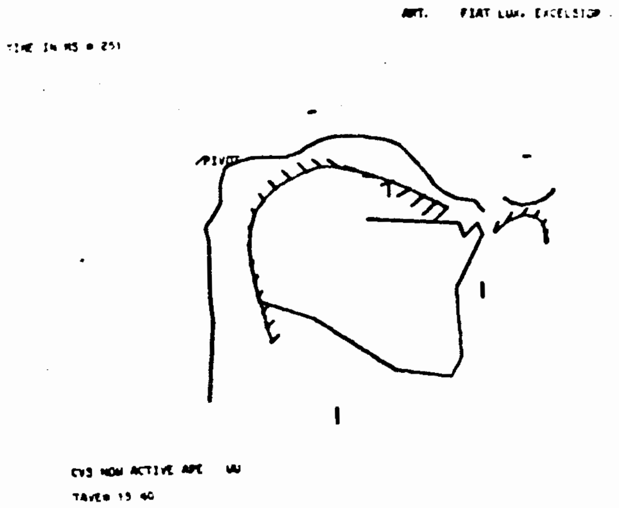
CVS NOW ACTIVE ARE

UU

Showing state position and configurative target.

Slightly later /T/ has been totally forgotten except for the feature "release" (and as "remembered" by the state). Lip rounding is progressing.

TIME IN MS = 251



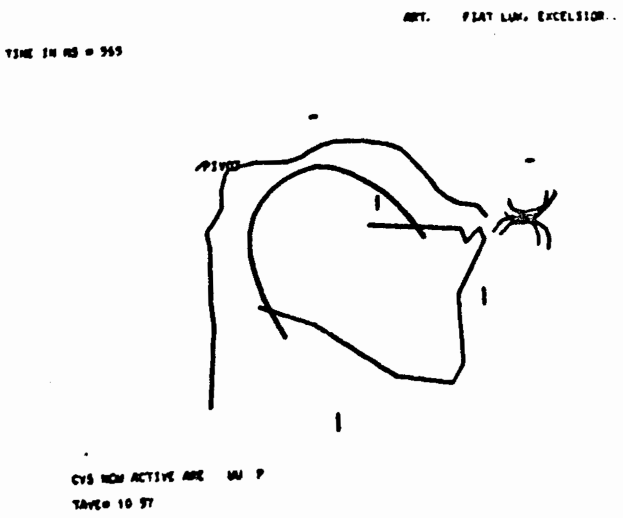
CVS NOW ACTIVE ARE

UU

Showing state position and state velocities for the nonrigid structural parts.

The "streamers" are velocity vectors for points along the structures.

TIME IN MS = 565



CVS NOW ACTIVE ARE

UU P

Showing state position and configurative target.

The /UU/, had reached a steady state. /P/ is just beginning in the vicinity of the lips.

# CHAPTER IV

# MODEL IMPLEMENTATION

## STATE DESCRIPTION

In implementing a model schemes to adequately describe the system must first be devised before one can begin to consider manipulation of the system. Just as in other fields where the invention of a notation conducive to insight about the problem is of fundamental importance, we need to be able to describe the state of the articulatory mechanism in a way that does not implicitly impose prejudices on our way of thinking about the process.

The anatomical parts which are pertinent to a quantitative specification of the configuration of the vocal organs may be subdivided into two classes. For the semirigid structures -- particularly the maxilla, the mandible, and the group of vertebrae adjacent to the posterior wall of the pharynx -- it is possible to define fixed points (so-called landmarks) in such a way that their location is fairly precisely determinable on radiographs. Thus the location of these structures with respect to each other can be succinctly specified. Tabulations of such experimental data (Perkell, 1965) show that the relative motion among these structures is sufficiently large so that it in general cannot be neglected. For the nonrigid structures of the articulatory mechanism -- particularly the tongue, lips, and velum -- the configuration must of course be described in a distributed manner along the entire region of interest.

One traditional description of configuration is by classifications such as front-back, open-close, and rounded for vowel production and the place of articulation for consonant production. These classifications have been quite popular since they do convey useful notions, but as generally used they are certainly not quantitative enough for any type of model implementation.

A popular method of description for speech synthesis procedures and articulatory measurements has been in terms of so-called parametric and/or analytic representations. Thus our initial efforts were naturally directed along these lines, and only slowly and painfully was a now quite obvious conclusion arrived at. For purposes of articulatory modeling parametric and/or analytic representations should be deemphasized.

Parametric representation is both natural and convenient for speech synthesis in the acoustic domain. Parameters such as formant frequencies, fundamental frequency and power levels seem to adequately describe the acoustic speech signal.

However, parametric representations have not enjoyed similiar success in the articulatory domain. One of the best known methods of specifying the shape of the vocal tract is the three parameter model of Stevens and House

(1955). Their scheme specifies the cross-sectional area of the vocal tract at the point of maximum constriction, the distance of this point from the glottis, and a factor representing the degree of lip opening and protrusion. It is a convenient scheme because of its quantitative nature and its extreme amount of compression into only three parameters. Other articulatory parameters that have been considered include such values as the center of mass of the tongue, angle of the tongue, tongue tip flexure with respect to the body, and parabolic parameters of lip opening.

However, these and all other articulatory parametric descriptions would seem to suffer from the fact that they are analytical or geometrical artifices which are not amenable to generalization toward a more natural or flexible model. Introspection suggests that in learning to speak one is not concerned with n-th degree mathematical curves or other n parameter descriptions, and we see no reason, therefore, why a model of the process should be so constituted. It may be "nice" that a tongue shape, for example, can be roughly described by a parabolic curve, but we feel that this is of no conceptual significance. It imposes the additional handicap of having to worry about how good is the parabolic approximation.

Another reason for this emphasis on analytic functions was no doubt due to the lack in the past of other tractable methods for conveniently describing spatial data. The availability of digital computers with graphical input/output facilities has now eliminated this reason.

A related major point in the implementation of our model is the description of the configurative state of nonrigid structures in terms of the positions of small adjacent segments of structure or "flesh points", rather than the values of intersections of structures with coordinate systems. An exactly analogous distinction is found in fluid mechanics and acoustics where the particle approach leads to the so-called "Lagrangian equations" and the fixed point in space view leads to the "Eulerian equations". Thus we focus our attention on what is happening to a particular part of structure as a function of time (Lagrangian) rather than what is happening at a particular point or along some fixed coordinate line.

This is the opposite of the more customary description, and the adoption of this method is a good example of how the specific implementation of a model forces one to reconsider many heretofore unquestioned assumptions and/or prejudices. The first implementations of the model were in terms of the more traditional description but it gradually became apparent that there were practical and conceptual difficulties with this approach, and these forced a reluctant major revision to the present method. This "flesh point" form of description makes some of the details of the realization of the computer simulation of the model more difficult, but yields a significant increase in power and elegance in view of the higher level aims and characteristics of the model.

In retrospect this revision has had a profound influence on our later thinking. Since our basic methodology is in terms of goal seeking mechanisms, the configurative articulator goals have now acquired physical significance since they represent the goals of particular flesh points instead of being statements such as "the pharynx area adjacent to the second vetebra should be 3 square cm." This refocusing of attention from coordinate values to structural descriptions was an important contribution to general model aims of naturalness and conceptual simplicity.

The present model describes the articulatory configuration in the midsagittal plane. The basic coordinate system is a two dimensional rectangular system with arbitrary origin and rotation. All part descriptions, some of which may initially be in other localized coordinate systems, can be transformed to the basic coordinate system for display and reference to other structures.

There are three nonrigid structures which are described on a point by point basis, the points being generally several millimeters apart. These nonrigid parts are the tongue and the two lips. The maxilla (hence roof of the mouth) and posterior pharynx wall comprise a single rigid structure which is fixed with respect to the basic coordinate system. Thus all positions and motions are ultimately specified with respect to the maxilla. A more advanced model would have to provide for movement of the posterior pharynx wall with respect to the maxilla, and also a controlled velum. The mandible is a rigid structure with only one degree of freedom (with respect to the maxilla), i.e., rotation about a pivot point. Though not precisely empirically true for the real system, this approximation is felt to be quite adequate for present purposes.

## SIMULATION IN TIME

The simulation of the model in time must be a continuous simulation as opposed to an event type of simulation. Continuous time is broken up into small, uniform discrete time increments, and at each step in time the state of the model is determined on the basis of the previous state of the model and all operators which are currently active. Higher level changes, i.e., a change in the status of active operators, occur at what we have previously called event times. Since these changes can only be effected between two time increments, the time increment size must be small enough so that an event can be caused at what effectively appears to be any arbitrary time. This is achieved if the intervals between event times are large compared to the basic simulation time increment.

A smaller upper bound for the time increment has been that required for the simulation of the physical mechanics (goal seeking and "natural" constraints). For the continuous part of the simulation at the physical level we encounter the problem of trying to effect what are inherently parallel and simultaneous computations using a serial machine. The best example of this problem in the present model is the shape-seeking or maintenance of structural integrity mechanism. If a strong force is applied at a particular point of a structure it should be "felt" throughout the whole structure. We speak of this as the propagation of perturbations through a structure. Since each so-called flesh point is only affected by its immediate neighbors, at each pass (i.e., each time increment) of the shape-seeking mechanism any perturbing influences at one flesh point can only propagate to the immediately adjacent points. Therefore the time increment must be small enough so that in an acceptably small amount of elapsed simulation time there will be enough passes of the shape-seeking mechanism to propagate a perturbation through the whole structure.

As dictated by these requirements, the presently used empirically determined value of the time increment is 1 millisecond. With this time increment the present model "runs" an order of magnitude or so slower than real time on a large scale computer with a 2 microsecond basic cycle time.

## RADIOGRAPHIC DATA, AND ONLINE RETRIEVAL

Our primary data source was a cineradiographic film (X-ray motion picture) of a human speaker. This film showed the articulatory configuration as seen in the midsagittal plane.

A system has been developed whereby copies of tracings of projections of films of this type are entered into the computer file system on a frame by frame basis. This is accomplished using a light pen on a CRT display and push button commands to "copy" parts of the tracing into the computer. Frame identification data are entered via the online teletype, and the frames of a "word" are stored sequentially in a file.

The programs which subsequently display these data accept as request parameters word name, frame number, etc., and reproduce the tracing on a display scope. For comparative purposes it is possible to display simultaneously more than one frame from the same file or from different files. The multiple tracings can be displayed at different intensities for distinguishability.

Our model uses these X-ray data in two primary ways. They are used as an aid when drawing and modifying the target abstractions (PS's) of the lower level memory of the model, as was shown in Chapter 3. And they are used to compare the articulatory state of the model with that of a typical example of the real system being modeled, in order to judge the adequacy of the current model and to suggest modifications for the same. Fig. 4.1 demonstrates this. It shows the state position generated by the model during a simulation run and one frame of the X-ray data.

Radiographic techniques and the resultant films more refined the those presently available would be very useful.

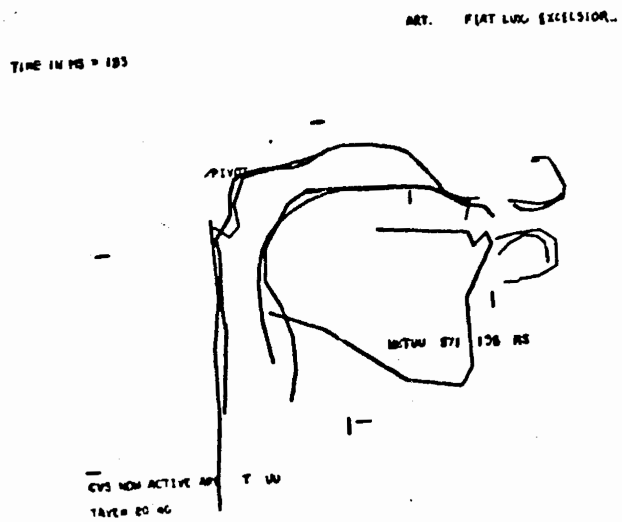


Fig. 4.1 Comparison of a model generated configuration with X-ray data.

Since the model description is in terms of individual segments of structures, it requires data such as "how much the central portion of the tongue elongates during the stop of a /k/ phoneme". In the presently available radiographs only the overall shape of most structures is barely discernible, and inferences about locations of specific segments of structures and actions such as relative elongation are at best only extremely rough guesses.

# CHAPTER V

## DETAILS OF A CENTRAL ASPECT OF THE MODEL: TARGET SEEKING FOR NONRIGID STRUCTURES

In this section we consider in more detail the algorithm of the position and shape seeking mechanism which is used for nonrigid structures. This algorithm is a part of the "low level mechanization" of the model. It does not have independent justification, but rather was somewhat empirically arrived at by the iterative process of trying a method, comparing the resultant model action with X-ray data, and then trying to think of a better method.

Most of this chapter is a description of the functioning of procedure RUNTOT in Appendix A. Since that procedure is written in language more appropriate for describing such functions it is both more concise (only 2 pages) and complete than this exposition.

Goal seeking and the maintenance of structural integrity occur continuously in time, and thus the following actions are executed at each time increment in the simulation.

All "forces" and velocities in the following discussion are two dimensional vectors since the present model is limited to the midsagittal plane.

For each "flesh point" of the structure, position seeking quasi-forces are generated as shown in Fig. 5.1.

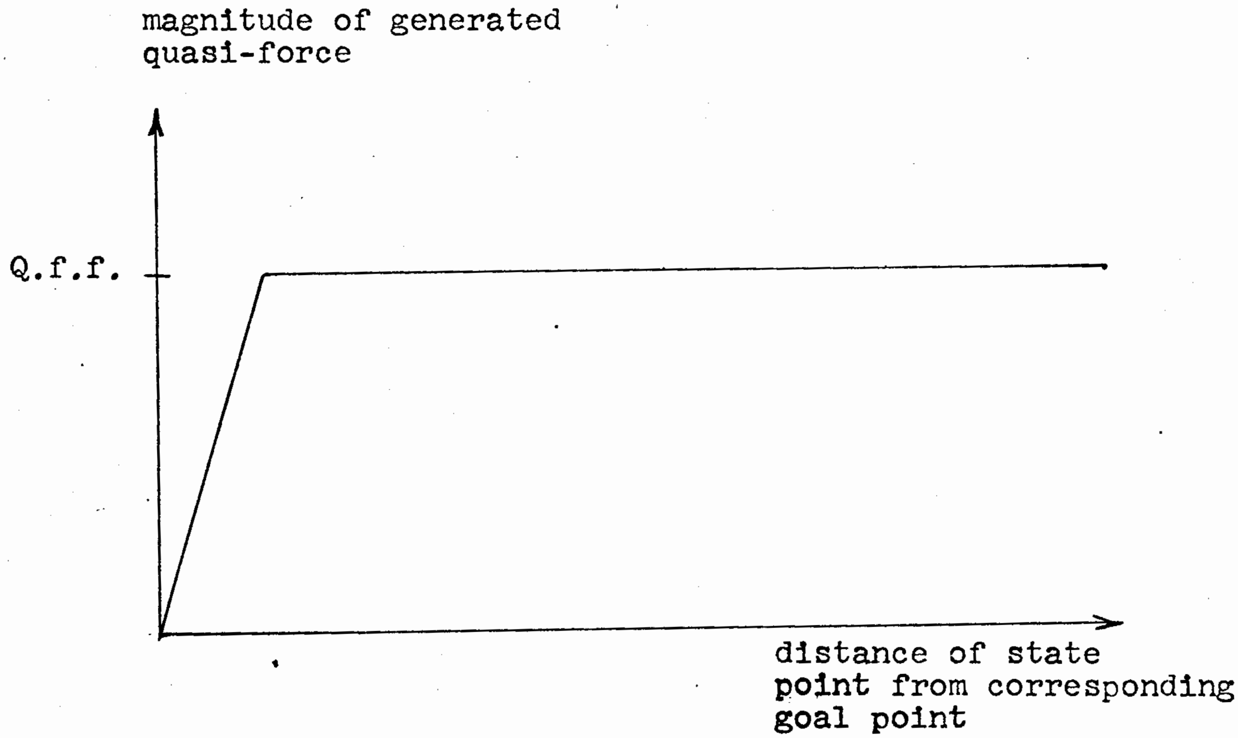


Fig. 5.1 Extrinsic quasi-force generation.

The direction of the generated quasi-force is always toward the goal point. Q.f.f. is the value of the quasi-force function at the current goal point.

They are always directed from the state point to its corresponding goal point. Except for when the state point is close to its goal point a quasi-force magnitude equal to the current quasi-force function is generated. When near the goal point the generated quasi-force decreases so that the goal point is a position of stable equilibrium for the state point.

If the goal has just been changed as the result of an event the above procedure alone would yield an abrupt change in the generated quasi-force. However, continuity is preserved in this force by what we call "muscle delay." The quasi-force effective just before the goal change is linearly faded to the new quasi-force over a period of tens of milliseconds.

To this position seeking quasi-force is added a shape seeking and maintenance of structural integrity "intrinsic" quasi-force. The <u>relative</u> position of the state point with respect to its two adjacent state points is compared to the relative position of the corresponding goal points. Tangential quasi-forces are generated which are linearly proportional to the local error in length or "stretch," and quadrature quasi-forces are generated which are linearly proportional to the error in local curvature. These shape correction forces are distributed between the state point and its immediate neighbors in such a manner that the overall sum is zero. Thus the intrinsic quasi-forces generally have no "net" external positional effect, but only affect the position of state points as relative to adjacent state points.

The state point is then "run" for one time increment by having these quasi-forces act upon a second order mechanical system with mass and damping (Eq. 5.1).

$$m \frac{d\vec{v}(t)}{dt} + D\,\vec{v}(t) = \vec{F}(t) \qquad (5.1)$$

m = point quasi-mass

D = point quasi-damping

F = total quasi-forces (extrinsic + intrinsic)

The difference equation used for the simulation follows directly from this.

$$\vec{v}(t+\mathrm{TMINC}) = \vec{v}(t) + \frac{\vec{F}(t) - D\,\vec{v}(t)}{m} * \mathrm{TMINC} \qquad (5.2)$$

TMINC is the time increment of the simulation.

It follows that if all intrinsic forces were zero (i.e., a free point) a single state point far from its goal point would thus first accelerate (mass-limited region of operation), then would reach a velocity-limited region where the velocity was determined by the magnitude of the goal quasi-force function and the damping. Eventually as the point neared its goal the operation would function in a goal-limited manner and the point would slow down and converge on the goal. These are the significant macroscopic effects. We consider the quasi-force generation method and the basic motion equation (5.1) only to be schemes to produce these macroscopic effects, and do not feel that they have any physical interpretation individually.

However, any motion so calculated that would cause the state point to "penetrate" a barrier is not allowed. A quasi-force component which is tangential to the barrier will cause the point to slide along the barrier, but a perpendicular component of a quasi-force just "pushes" against the barrier but can produce no motion. Points not directly in contact will also be affected due to the propagation of intrinsic forces from point to point.

# CHAPTER VI

## COMPUTER PROGRAMMING

We earlier stated our feeling that a formal algorithmic expression of the model is a necessary factor of any good model. For our model this expression is ultimately in the form of computer programs written in a machine independent language. We do not consider writing programs a distasteful necessary evil of implementing the model, since this process was one of the most influential factors in the evolution of the model. The actual writing of programs was the genesis of many of the ideas contained in the model. One often sits down to the writing of a program with only a vague notion of what he would like to accomplish, and then is forced to clarify and organize that notion to render it in a formal language which does not permit vagueness.

Many of the concepts of the model were suggested by recently developed computer programming techniques and languages. The division of the model into a state and operators has close parallels in the general modeling philosophy of Ross and Rodriguez (1963) where "operators" are "turned loose" on data structure ("plex" structure) to accomplish diverse functions.

### LANGUAGE USED

The early versions of the model were written in the MAD language (Michigan Algorithm Decoder, a variant of

ALGOL-58), but as the model became more complex it became increasingly difficult to implement the necessary operations in that language. The model is currently written in AED (ALGOL-60 Extended for Design) (Ross, 1964), and the plex structure processing (a generalization of list processing) feature of that language has been a sine qua non for the evolution of the model to its present status. The AED Project and associated individual personnel supplied packages of routines for "free storage" dynamic storage allocation, for use with the ESL display console, and for free-format input-output, in addition to many other utility routines.

The "event" manipulation of the model borrows ideas from SIMSCRIPT (Markowitz, 1963), a programming language designed for event based simulations. It is a tribute to the power of the AED system that the operations entailed in the essence of SIMSCRIPT, its CAUSE statement, could be implemented in AED using only a few statements by means of functions which use beads from free storage as "event notices" which are kept on a rank ordered event list.

## ANCILLARY PROGRAMS

In Appendix A are listings of the programs which provide the top level program description of the model per se. These listed programs are a small fraction of the total programming for the modeling system. A brief summary of functions of the other major subdivisions of programming which are dedicated to the model follows.

One subsystem (EDPS) was developed for the online drawing and modification of the spatial goal abstractions and their associated quasi-force functions (PS's). This system allows the user to view other PS's and also radiographic data as an aid in specifying a PS. It also includes what is essentially a two dimensional spatial low pass filter to "smooth" noisy graphical input data. The noise results from the the actual drawing process and from the lack of adequate user knowledge (radiographic data) as to "flesh point" locations. This subsystem is well demonstrated in the supplemental motion picture.

Techniques and programs have been developed for the input (XRIN) and display (XDIS) of cineradiographic data. The features of this system have been described elsewhere in this report.

CVTRAN (for CV Translation) is a program which translates the CVTAB from a form understandable by humans, an example of which was presented above, to a form understandable by the model.

A large personal library written in FAP (an assembler) and AED contains many utility routines used by the above systems. Additional routines are used from various other personal and public libraries on the MAC system. These are AEDLB1, KLULIB, TSLIB1, ASMPAK, and RDITEM.

# CHAPTER VII

## CLOSURE

In this concluding chapter we consider briefly several topics which should provide a perspective for the present model. These include suggestions for further development of the model.

Inasmuch as classical form would indicate the inclusion of a "results" section here we should state why there is none per se. The results of this work are the model and the methods of implementation which have been the subject of the preceding chapters. The dynamic action of the present model is illustrated in one sequence of frames included here and more profusely in the supplemental motion picture film. Visual observations and comparisons of this type are the key to the evolution of the model, and the gathering of a large body of numerical statistics for any one particular model would serve no purpose.

### MODEL "LEARNING"

Let us consider the various levels of learning involved in the evolution of a model of speech production. As we, the model builder, learn more about the process and develop more sophisticated hypotheses which we wish to try, the structure or form of the model will change. But for a given model the model itself must also "learn". By this we mean that the content of the memory of the given model will be expanded and modified.

For this type of "learning" we try various combinations of configurations, forces, time durations, and other parameters contained in the memory of the model and observe the behavior of the model. For the present model this consists of modifying the configurative subtargets and their associated quasi-force functions (PS's), and the entries in the CVTAB.

It might be noted that the "learning" of our present model has been by visual means (comparison with X-ray data) rather than by aural means (comparison with an acoustic speech signal), and the resultant "learned memory" is in terms of configurative goals rather than acoustic goals. For the real system it appears that the goals during learning are primarily acoustic. The learning speaker tries to mimic the speech sounds of his environment, and uses feedback through his auditory system to compare his efforts with the external sounds. In this way he learns to correlate articulatory positions and movements with phonologically meaningful sounds.

After the movements become habits, however, acoustic goals and auditory feedback are probably less significant. The lowest level memory in our model is concerned with articulatory events rather than sounds, and therefore we might conjecture that our model is a model of the speech production process after the lower level movements have been learned or have become habits, and does not pertain to the learning of these actions.

The real reasons for observing in the articulatory domain are more pragmatic. Although the articulatory to acoustic transformation is fairly straightforward, such is not the case for the inverse transformation. By observing in the articulatory rather than the acoustic domain we do not have to concern ourselves with this inverse transformation. That is, rather than having to infer back from an incorrect sound to corrections in articulation we make our observations directly at the articulatory level. Another reason is that equipment for the generation of an acoustic signal was not available.

## TIMING AND FEEDBACK CONTROL

Timing and "feedback" control are aspects of the speech act which are not presently well understood. Our present model has raised questions in these areas, but is not advanced enough to offer any definitive conclusions. Nevertheless, we feel it worthwhile to mention some of these issues which are relevant to the present implementation of the model.

For a comprehensive treatment of the literature, experimental data, and hypotheses germane to this area of speech production we refer the reader to the chapter titled "Organization of a Time Program of Syntagma" in <u>Speech: Articulation and Perception</u> by Kozhevnikov et al (1965).

A basic issue is how and at what level(s) in the system is timing control generated or derived. Do lower level events (aspects of the state trajectory in our model) serve to "trigger off" future events, or is a fixed higher level "articulatory program" issued to the lower physical levels to be executed blindly, independent of the consequences? In other words, is control "closed loop" with proprioceptive and auditory feedback, or rather is it of a preprogrammed high precision "open loop" type?

An example in our model of a "trigger" type of action is the waiting for closure during the production of a stop. The achievement of closure is used as a signal for the next articulatory "event".

Feedback for positional control is also of interest. In our present model there is a type of kinesthetic feedback inherent in the position and shape seeking mechanisms since these function by sensing and trying to minimize the difference between the state and the configurative goal.

Psychologists have been interested in the likening of neural systems to feedback control systems. Most seem to feel that there is, in some sense, feedback, but the big question is: To how high a level or how far "back" does the closed loop go? Some theories of high level closed loop control encounter difficulties due to time delays. Miller et al (1960, p.91) point out:

> ... The problem for most theories of the neural basis of skilled movements is that skilled movements run off so very rapidly that there is little time for proprioceptive feedback from one movement before another must occur. Any

> simple conception in terms of feedback, or error correction, circuits must cope with the relatively slow transmission rates that are possible over neural paths.

In a chapter entitled "Motor Systems", Ruch (1951) considers the various sources of delays and gives us an order of magnitude feeling for their duration (p. 204).

> One may pose the question: "At what rate could a single motor unit contract in voluntary contraction and yet have each contraction modulated on the basis of its previous contraction?" To 19 to 34 milliseconds for the circuit from muscle to muscle through the cerebellar and motor cortex would be added the contraction time of perhaps another 20 to 40 milliseconds, since contraction directly or indirectly stimulates the proprioceptive end organs. Taking the larger figure for circuit conduction time, discharges at slightly more than ten per second could be so modulated.

Since these high level closed loop delays thus seem to be longer than could be tolerated for effective "output informed" feedback control, Ruch later in the same work (1951, p.205) speculates about what he calls "input informed feedback circuits" in an attempt to alleviate this problem.

> The problem faced by the motor cortex in executing a voluntary eye-hand coordination, such as picking up an object, hinges on time considerations. It can be argued that, at the moment an act is launched, a time-tension pattern of muscle contraction is instituted, projecting into the future, with the object being the goal of this reaction. The end of the movement, removed in time, is implicit in the patterning of the discharge. It seems unlikely that the initial stage of the movement is ordered blindly, then the second stage, and so on, with the goal direction appearing only at the last moment.
>
> The nervous system would be handicapped in "planning movements" in this fashion because there are no known methods for storing impulses in a neuron to be discharged after a fixed delay. Significant delay can be obtained by making use of conduction time. Thus the problem of delay has apparently been solved by circular chains of neurons or reverberating circuits: the nerve impulses circulate within such circuits and give off one or more impulses per circuit. The decay characteristics give a temporal patterning of the cortical discharge.

> The cerebral-cerebellar circuit may represent not so much an error-correcting device as a part of a mechanism by which an instantaneous order can be extended forward in time. Such a circuit, though uninformed as to consequences, could, so to speak, "rough-in" a movement and thus reduce the troublesome transients involved in the correction of movement by output-informed feedbacks. Especially could such a circuit be effective in the termination of movement.

Ruch's last sentence above is interesting since such "termination of movement" occurs in our model as the state reaches a goal.

## REFINEMENT AND EXTENSIONS OF THE MODEL

Throughout the discussion of the model in preceding chapters we have been including parenthetically comments concerning the possible extensions of the model. We consider additional aspects here.

Without adding to the present state description much work needs to be done on increasing the vocabulary to more types of sounds and the sequencing thereof. This would involve extensive comparison of model action with more diverse data.

Enlarging the state description to include air flow and pressures, and voicing or phonation is certainly recommended. We consider these components of the state together since there appears to be a strong interaction between them. Because of this interaction they would no doubt exploit the possibilities inherent in the operator-state dichotomy of the model to a greater degree than is done in the present model.

For example, consider a voiceless stop. It has been suggested by K.N. Stevens that voicing may be stopped indirectly by stopping the air flow rather than directly by instructions to the glottis. That is, during a complete closure of the vocal tract supraglottal pressure will build up relative to subglottal pressure and this will "naturally" decrease the glottal air flow, thereby stopping voicing.

A similiar situation may occur during the onset of voicing. If the feature "voiced" characterizes a phoneme found in the phoneme input list a "command" to this effect will be issued which only prepares the glottis or puts it in a state of readiness, and the onset of voicing is contingent upon the attainment of suitable sub- and supraglottal air pressures and the resultant air flow.

Air pressures might also have an effect upon the configurative state, since they would create real forces to be added to the various quasi-forces already in the model. For example, it has been noted (Perkell, 1965) that a "lax" pharyngeal region will expand to permit glottal air flow during a supraglottal stop.

The attainment of a particular relative air pressure at a specific location might also be a goal in itself. Here we are thinking of some of the consonantal speech sounds.

Other additions to the model could include:

Velum action and nasal speech sound capabilities.

Extension out of the midsagittal plane.

## POSSIBILITY OF ACOUSTIC OUTPUT

Inasmuch as the perception of the acoustic signal is our ultimate concern (apologies to Tillich), we would like to comment on the possibilities of adding an acoustic output capability to the "output" end of our model. Briefly stated, the extension of the model to do this holds no major conceptual problems and is within the state of the art. It would require, however, a fair amount of competent engineering talent, time, and hardware expense.

The actual generation of the waveform could be effected by hardware with vocal tract analog electronic circuitry (cf. the late DAVO at MIT, Rosen, 1958), or by a sample data simulation in a computer (cf. Kelly and Lochbaum, 1962). For many scientific, technical, and economic reasons we suggest the software or simulation approach as opposed to the hardware approach even though the synthesis would probably require 10 to 100 times real time with current economically feasible computers. It might be remembered that the present articulatory model also "runs" considerably slower than real time.

As discussed above, a more advanced form of the model would most likely contain air pressures in the state description, and these would be the same air pressures used for the acoustic waveform generation simulation. Thus, for example, the build up and decay of air pressure associated with a stop would manifest itself both in its effects on the release (the articulatory state) and in the sound associated therewith.

Since the present articulatory description is only in the midsagittal plane a transformation from this description to the acoustically significant area function of the vocal tract would also be necessary. Data for this transformation are being accumulated at MIT (Heinz and Stevens, 1965) and elsewhere.

# APPENDIX A

## SELECTED PROGRAM SEGMENTS OF THE MODEL

In this appendix we include for the record actual programs of the present model. There are many specific details of the mechanization of the model which do not warrant additional exposition but can be found in these listings if desired. Since such details are included here mainly for completeness they can be expressed much more succinctly in an appropriate artificial algorithmic language than in natural English.

The listings are extensively annotated so that a general idea of the functioning of the programs can usually be obtained from just the comments and remarks. Additional explanatory paragraphs of standard text are also included. The first program contains the basic control loop and the machinery for the manipulating the occurrence of events. This is followed by a group of procedures which mechanize the continuous part of the simulation, i.e., physical mechanics and goal seeking. The next group is concerned with higher level event control (i.e., phoneme sequencing and goal setting), and with references to the "memory" of the model involved in these processes. Then there is the main initialization procedure. The listings conclude with input/output communication between the model simulation and the user, i.e., the model controller and observer.

Some notational conventions are as follows: All procedures whose name is prefixed by INIT are called only once at the beginning for initialization of the model. Component mnemonic prefixes of PS and CV always refer to data structure associated with these levels of memory. STB is STate Bead. PSB is PS Bead. In the AED-0 language "$," is equivalent to the ";" in ALGOL, i.e., it is the statement terminating punctuation.

For completeness we begin the listings with declarative files used as INSERTS. These declare the data structure, but for the non AED versed reader the only relevant items are the remarks pertaining to the components contained in the state.

File .COMDA ALGOL

```
... .COMDA DECLARES STATE PART DATA STRUCTURE AND COMMON //
INTEGER COMPONENT PREV,NEXT $,
PACK 77777C,0,ADDRESS COMPONENTS NEXT $,
PACK 77777C18,18,DECREMENT COMPONENTS PREV $,
PREV $=$ NEXT $=$ 0 $,
INTEGER COMPONENT STPI,TRGPHM $,
BOOLEAN COMPONENT VFORMGRB $,
REAL COMPONENT SPOSX,SPOSY,SVX,SVY,TRGX,TRGY,TRGI,TRGJ,
 TMLSTSETG,FPVFORX,FPVFORY,VOPTON.TI,VFORMG,
 VFORX,VFORY,FORX,FORY,COSX,SINX $,
... STATE POSITION AND VELOCITY $,
SPOSX $=$ 1 $,
SPOSY $=$ 2 $,
STPI $=$ 3  ... ST POSITION INDEX $,
SVX $=$ 4 $,
SVY $=$ 5 $,
... LOCAL PHM NAME, POSITION, SHAPE ATTRIBUTES $,
TRGPHM $=$ 6 $,
TRGX $=$ 7 $,
TRGY $=$ 8 $,
TRGI $=$ 9 $,
TRGJ $=$ 10 $,
... MUSCLE DELAY STATE ATTRIBUTES $,
TMLSTSETG $=$ 11  ... TIME LAST SET GOAL $,
```

```
 FPVFORX $=$ 12  ... VFOR WHEN LAST CHANGED GOAL $,
 FPVFORY $=$ 13 $,
 ... MUSCLE DELAY OP ATTRIBUTE $,
 VOPTON.TI $=$ 14  ... VOP TURN ON TIME INTERVAL $,
 ... MUSCLE MANNER PHM NAME AND MAX MAGNITUDE $,
 VFORMGRB $=$ 15  ... RELEASE BIT $,
 VFORMG $=$ 16 $,
 ... INTERNAL QUANTITIES $,
 VFORX $=$ 17  ... VOLITIONAL FORCES $,
 VFORY $=$ 18 $,
 FORX $=$ 19  ... TOTAL FORCES $,
 FORY $=$ 20 $,
 COSX $=$ 21  ... OF STATE SEG LEAVING PI $,
 SINX $=$ 22 $,
SYNONYMS 23 = STBDSZ $,
 ... PHMLIST STRUCTURE. - PREV AND NEXT IN 0, A 2-WAY RING . $,
 INTEGER COMPONENT PHMNAME,PHMCVHD $,
 REAL COMPONENT PHMTMD1,PHMTMD2 $,
 PHMNAME $=$ 1 $,
 PHMCVHD $=$ 2 $,
 PHMTMD1 $=$ 3 $,
 PHMTMD2 $=$ 4 $,
SYNONYMS 5 = PHMBDSZ $,
 ... COMMON * * * * * $,
 REAL TIME,TMINC,DAMP,MASS,
  FDISBP,INTRGI,INTRGJ,
  JPOSOD,JTAU $,
 INTEGER EVENTLIST,PHMLIST $,
 INTEGER ARRAY FIRSTSTB(2) $,
 COMMON TIME,TMINC,DAMP,MASS,FDISBP,INTRGI,INTRGJ,
  EVENTLIST,PHMLIST,FIRSTSTB,
  JPOSOD,JTAU $,
 PROCEDURE WFLX,MWFLX  ... FOR CONVENIENCE $,


File  .CVCOM ALGOL

 ... CVHEAD //
INTEGER COMPONENT CVNAME1,CVTYPE,CVFIRST $,
REAL COMPONENT CVTMD1,CVTMD2 $,
CVNAME1 $=$ 1  $,
CVTYPE $=$ 2 $,
CVTMD1 $=$ 3 $,
CVTMD2 $=$ 4 $,
CVFIRST $=$ 5 $,
 ... CV PART BEAD $,
INTEGER COMPONENT CVPNAME1,CVPSPN,CVPSHD $,
REAL COMPONENT CVPOPTON.TI $,
BOOLEAN COMPONENT CVPREL $,
CVPNAME1 $=$ 1 $,
CVPSPN $=$ 2 $,
CVPREL $=$ 3 $,
CVPOPTON.TI $=$ 4 $,
```

```
CVPSHD $=$ 5 $,
```

File .PSCOM ALGOL

```
INTEGER COMPONENT PREV,NEXT $,
PACK 77777C18,18,DECREMENT COMPONENTS PREV $,
PACK 77777C,0,ADDRESS COMPONENTS NEXT $,
PREV $=$ NEXT $=$ 0 $,
INTEGER COMPONENT PSNAME1,PSPN,PSPILB,PSPIUB,PSFIRST $,
INTEGER COMPONENT PDIS.R,FDIS.R  ... NEEDED ONLY IN EDPS VERSION $,
PSNAME1 $=$ 1 $,
PSPN $=$ 2 $,
PSPILB $=$ 3 $,
PSPIUB $=$ 4 $,
PSFIRST $=$ 5 $,
PDIS.R $=$ 6 $,
FDIS.R $=$ 7 $,
SYNONYMS 6 = PSHDSZ $,
SYNONYMS 8 = EPSHDSZ $,
REAL COMPONENT PSX,PSY,PSF $,
INTEGER COMPONENT H,V,FORFN $,
PSX $=$ H $=$ 1 $,
PSY $=$ V $=$ 2 $,
PSF $=$ FORFN $=$ 3 $,
SYNONYMS 4 = PSBDSZ $,
SYNONYMS 7 = EPSBDSZ $,
 ... COMPONENTS 4,5,6  DECLARED AND USED IN EDSMOO $,
```

TIMER is the (MAIN) program. It first declares and initializes the eventlist, then makes a call on INITAL to initialize the rest of the system. It then enters the basic execution loop which is executed once every time increment for the continuous part of the simulation. This loop also continuously checks the eventlist to see if an "event" is due.

```
         BEGIN
COMMENT - - - TIMER, I.E., MAIN CONTROL. NOTE SIMILARITY
 WITH SIMSCRIPT 'CAUSE' STATEMENT. ALL REQUESTS FOR ACTION
 ARE EFFECTED THRU 'CAUSIT'. CAUSIT CREATS AN EVENT NOTICE
 AND PUTS IT ON AN EVENT LIST WHICH IS RANKED ON 'WHEN.TO'
 (TIME WHEN TO DO ACTION) IN ASCENDING ORDER. EVENTLIST IS A
 2-WAY COMPLETE RING FOR EASE IN HANDLING INSERTS ANYWHERE.
 WHEN TIME GETS TO 'WHEN.TO' OF AN EVENT NOTICE IT IS
 EFFECTED VIA 'WHERE.AT' AND THEN THE EVENT NOTICE IS
 DELETED. 'CANCLIT' CANCELS AN EVENT NOTICE PUT IN BY
 'CAUSIT'. REQUESTS FOR IMMEDIATE ACTION FROM ANYWHERE IN
 THE SYSTEM ARE SIMPLY EFFECTED BY A 'CAUSIT' WITH 'WHEN.TO'
 LEQ PRESENT TIME. $,
         INTEGER COMPONENT WHERE.AT ... LOC OF LABEL OR
                    PROC. $,
         REAL COMPONENT WHEN.TO ... TIME WHEN TO EXECUTE
                    DOIT(WHERE.AT) $,
         WHEN.TO $=$ 1 $,
         WHERE.AT $=$ 2 $,
         SYNONYMS
            3 = ELISTBDSZ $,
COMMENT - - - - MASTER TIME AND EVENT CONTROL - - - - $,
COMMENT - - - - RUNTOT AND STOPSHOW COULD BE SCHEDULED VIA
 EVENT NOTICES ON THE EVENT LIST. INSTEAD THEY ARE TREATED
 AS SPECIAL CASES SINCE RUNTOT IS SO CONSISTENT, AND
 STOPSHOW MAY GET CALLED BY FAULT CONDITIONS. $,
COMMENT - - - - AT ANY INSTANT IN TIME THE FIRST STOPSHOW
 OCCURS BEFORE EVENT EXECUTIONS, SUCCEEDING ONES AFTER. THUS
 IMMEDIATE EVENTS CAUSED BY STOPSHOW INPUT WILL BE EXECUTED
 AND SEEN IN NEXT SHOW $,
         .INSERT .COMDA $,
         INTEGER PROCEDURE CAUSIT ... DEFINED HERE $,
         PROCEDURE EVTRTN,CANCLIT ... DEFINED HERE $,
         INTEGER PROCEDURE FREZ,LNKBD,UNLNKBD $,
         PROCEDURE FRET,DOIT,SETBRK,STRTNCV $,
         REAL PROCEDURE STOPSHOW $,

COMMENT - - (MAIN) ENTRY, SO DO INITIALIZATION. $,
         PROCEDURE INITAL ... INITIALIZE MANY THINGS $,
```

```
          INITAL() $,
          EVENTLIST = FREZ(1) ... INIT. EVENTLIST HERE $,
          NEXT(EVENTLIST) = EVENTLIST $,
          PREV(EVENTLIST) = EVENTLIST $,
          CAUSIT(LOC INIT2,0.) ... REQUEST AN IMMEDIATE
                     ACTION, FUDGING TO GET STARTED $,
         GOTO MAINBRK $,
INIT2 $   STRTNCV() ... TO GET STARTED. WILL FAULT TO
                     STOPSHOW FOR INPUT. THEN STRTNCV() WILL
                    TRANSFER TO NEW CVPROG, AND EVTRTN()
                    BACK TO NORMAL FROM THERE $,
          INTEGER FIRST $,
          REAL NEXTSHOWTM $,
MAINBRK $ SETBRK(MAINBRK) $,
          GOTO SHOWBRK $,


 COMMENT - - BEGIN BASIC EXECUTION LOOP WHICH IS SCANNED
  EVERY TIME INCREMENT. FIRST CHECK IF A STOP AND DISPLAY IS
  WANTED, THEN SEE IF ANY EVENTS ARE DUE. $,
SHOWCK $  IF NEXTSHOWTM GRT TIME
          THEN GOTO NXTECK ... NO SHOW YET $,
SHOWBRK $ NEXTSHOWTM = STOPSHOW() $,
 COMMENT - - EXECUTE ALL EVENTS DUE NOW $,
NXTECK $  IF (FIRST = NEXT(EVENTLIST)) EQL EVENTLIST
          THEN GOTO NOEVTS ... LIST EMPTY $,
          IF WHEN.TO(FIRST) LEQ TIME ... DO IT NOW //
          THEN BEGIN
              DOIT(WHERE.AT(FIRST)) ... GO TO IT $,
              DEFINE PROCEDURE EVTRTN TOBE
                    ... EVENT RETURN //
                   GOTO EVTRTNLB $,
EVTRTNLB $    FRET(ELISTBDSZ,UNLNKBD(FIRST,PREV,NEXT))
                    ... FLUSH EVENT NOTICE $,
              GOTO NXTECK $,
              END $,
NOEVTS $           ... NO MORE EVENTS FOR NOW $,
          IF NEXTSHOWTM LEQ TIME
          THEN GOTO SHOWCK $,
 COMMENT - - NOW RUN THE SIMULATION AHEAD FOR ONE TIME
  INCREMENT $,
          PROCEDURE RUNTOT,RUNJ,BARRIE $,
          TIME = TIME+TMINC ... UPDATE TIME $,
          RUNTOT(FIRSTSTB(0)) ... ADVANCE THE TONGUE $,
          RUNTOT(FIRSTSTB(1)) ... UPPER LIP $,
          RUNTOT(FIRSTSTB(2)) ... LOWER LIP $,
          RUNJ()    ... AND THE JAW (MANDIBLE) $,
          INTEGER PROCEDURE PALATE $,
          BARRIE(FIRSTSTB(0),PALATE()) ... CHECK BARRIER
                     PENETRATION, TONGUE AND PALATE $,
          BARRIE(FIRSTSTB(2),FIRSTSTB(1)) ... AND THE 2 LIPS
                     $,
          GOTO SHOWCK $,
```

```
 COMMENT - - END OF BASIC EXECUTION LOOP $,

 COMMENT - - DEFINE PROCEDURES CAUSIT AND CANCLIT FOR
  MANIPULATING THE EVENTLIST $,

          DEFINE INTEGER PROCEDURE CAUSIT(LOCWHERE,TIMEWHEN)
           WHERE INTEGER LOCWHERE $,
          REAL TIMEWHEN TOBE
              BEGIN
              INTEGER NWEVTNOTE, ... NEW EVENT NOTICE //
               ESCAN ... EVENT SCAN $,
              CAUSIT = NWEVTNOTE = FREZ(ELISTBDSZ) $,
              WHEN.TO(NWEVTNOTE) = TIMEWHEN $,
              WHERE.AT(NWEVTNOTE) = LOCWHERE $, ... //
                   ... INSERT EVENT NOTICE INTO EVENT LIST
                     $,
              IF (ESCAN = NEXT(EVENTLIST)) EQL EVENTLIST
                    ... LIST IS EMPTY //
              THEN GOTO PUTON $,
              FOR ESCAN = ESCAN,NEXT(ESCAN) WHILE ESCAN NEQ
               EVENTLIST ... CHECKS FOR END //
              DO  IF WHEN.TO(NWEVTNOTE) LES WHEN.TO(ESCAN)
                  THEN GOTO PUTON $,
PUTON $       LNKBD(NWEVTNOTE,PREV(ESCAN),PREV,NEXT)
                    ... PUT ON JUST AHEAD OF ESCAN, NEEDS
                     RING FOR END SITUATIONS $,
              END   ... OF CAUSIT() $,

          DEFINE PROCEDURE CANCLIT(EVTNOTE) WHERE INTEGER
           EVTNOTE TOBE
              BEGIN
             INTEGER ESCAN $,
              ESCAN = EVENTLIST $,
             FOR ESCAN = NEXT(ESCAN) WHILE ESCAN NEQ
               EVENTLIST
              DO  IF ESCAN EQL EVTNOTE
                  THEN BEGIN
                      FRET(ELISTBDSZ,UNLNKBD(ESCAN,PREV,NEXT)
                       ) $,
                      GOTO RETURN $,
                      END $,
              END   ... OF CANCLIT() $,

         END FINI
```

The next group of programs are executed once each time increment by the basic execution cycle for the continuous part of the simulation. They comprise the lowest level of the model -- mechanical constraints and goal seeking.

Procedure RUNTOT advances the state of a nonrigid structural part for one time increment. It is used for the tongue, the upper lip, and the lower lip. It generates extrinsic or positional quasi-forces which include the effect of "muscle delay". To these are added shape (internal, integrity) quasi-forces. The state is then advanced on the basis of the resultant forces acting upon a second order system with mass and damping.

```
  BEGIN
   .INSERT .COMDA $,
COMMENT - - - - - - - - - PROCEDURE RUNTOT ACCEPTS EXTRINSIC
 FORCES, ADDS ITS OWN INTERNAL INTEGRITY FORCES, AND
 ADVANCES THE STATE FOR ONE TIME INCREMENT. DOES NOT WORRY
 ABOUT IMPENETRABILITY. BARRIE CALLED FROM ELSEWHERE $,
   REAL PROCEDURE SQRT $,
   REAL SPERX,SPERY,SPERM, ... //
    FORFN,REFX,REFY,REFMAG $,


   DEFINE PROCEDURE RUNTOT(FRSTSTB) WHERE INTEGER FRSTSTB
    TOBE
       BEGIN
       INTEGER STB,NSTB,PSTB $,
COMMENT - - - CALCULATE VFORX,Y FROM TRGX,Y , SPOSX,Y , AND
 VFORMG. SV NOT CONSIDERED HERE $,
       FOR STB = FRSTSTB,NEXT(STB) WHILE STB NEQ 0
       DO  BEGIN   ... FOR ALL PTS //
            SPERX = TRGX(STB)-SPOSX(STB) $,
           SPERY = TRGY(STB)-SPOSY(STB) $,
          SPERM = SQRT(SPERX*SPERX+SPERY*SPERY) $,
COMMENT - - - CALC. FORCE FROM FROM SPERM AND VFORMG(STB).
 THIS VERSION FORCE IS LINEAR W.R.T. SPERM UP TO BREAK POINT
 DISTANCE FDISBP, AND FLAT BEYOND THAT. $,
           FORFN =
           IF SPERM GRT FDISBP
           THEN VFORMG(STB)
           ELSE(SPERM/FDISBP)*VFORMG(STB) $,
```

```
           VFORX(STB) = FORFN*(SPERX/SPERM) $,
           VFORY(STB) = FORFN*(SPERY/SPERM) $,
           END $,
COMMENT - - - - - MUSCULAR FORCE TIME LAG - - - - - $,
       REAL TI.SINCE.SET,PERCENTNEW $,
       FOR STB = FRSTSTB,NEXT(STB) WHILE STB NEQ 0
       DO  IF (TI.SINCE.SET = TIME-TMLSTSETG(STB)) LES
        VOPTON.TI(STB)
           THEN BEGIN ... MUSCULATURE LETHARGY STIIL
                   EFFECTIVE //
                PERCENTNEW = TI.SINCE.SET/VOPTON.TI(STB)
                   ... PERCENT NEW MUSCLE FORCE,HERE LINEAR
                    $,
               VFORX(STB) = VFORX(STB)*PERCENTNEW ...
                   DIMINISH NEW //
                +FPVFORX(STB)*(1.-PERCENTNEW) ... REMEMBER
                    OLD $,
               VFORY(STB) = VFORY(STB)*PERCENTNEW+FPVFORY(
               STB)*(1.-PERCENTNEW) $,
               END $,
COMMENT - - USE VFORX,Y TO RESET FORX,Y. NOW NEED THE TWO
 SETS $,
       FOR STB = FRSTSTB,NEXT(STB) WHILE STB NEQ 0
       DO  BEGIN
           FORX(STB) = VFORX(STB) $,
           FORY(STB) = VFORY(STB) $,
           END $,
COMMENT - MUSH ADDS SHAPE FORCES TO VOLOP FORCES. SHAPE
 ERROR IS DETERMINED BEFORE TRIAL MUSH, I.E., NO TWO.PASS AS
 EARLIER. NOW ADD IN DISTRIBUTED FORCES RATHER THAN FINDING
 A CUMULATIVE NEIGHBORHOOD ERROR. $, ... //
                  ... SIN + COS WRT X AXIS OF ST SEG
                      LEAVING PI $,
      FOR STB = FRSTSTB,NEXT(STB) WHILE NEXT(STB) NEQ 0
       DO  BEGIN   ... ALL EXCEPT LST PT //
            REFX = SPOSX(NEXT(STB))-SPOSX(STB) $,
          REFY = SPOSY(NEXT(STB))-SPOSY(STB) $,
           REFMAG = SQRT(REFX*REFX+REFY*REFY) $,
           COSX(STB) = REFX/REFMAG $,
           SINX(STB) = REFY/REFMAG $,
           END $,
COMMENT - START BIG LOOP ADDING FORCES AT ALL POINTS
 (INCLUDING END POINTS) DUE TO ERRORS AT ALL POINTS EXCEPT
 END POINTS $,
       REAL NXTDX,NXTDY,SPI,SPJ,SH.ERRI,SH.ERRJ,FORI,FORJ $,
       STB = NEXT(FRSTSTB) ... START W 2ND BEAD $,
MUSHLP1 $ NSTB = NEXT(STB) $,
       PSTB = PREV(STB) $,
       NXTDX = SPOSX(NSTB)-SPOSX(STB) $,
       NXTDY = SPOSY(NSTB)-SPOSY(STB) $,
       SPI = NXTDX*COSX(PSTB)+NXTDY*SINX(PSTB) $,
       SPJ = -NXTDX*SINX(PSTB)+NXTDY*COSX(PSTB) $,
       SH.ERRI = SPI-TRGI(STB) ... NOT PER UNIT HERE $,
```

```
       SH.ERRJ = (SPJ-TRGJ(STB))/TRGI(STB) ... RADIANS $,
COMMENT - - TRULY LINEAR FORCE FCTN HERE, COMPARE WITH
 LINEAR-BREAKPOINTABOVE, CONSIDER OTHER FCTN FORMS IN BOTH
 PLACES. $,
       FORI = INTRGI*SH.ERRI $,
       FORJ = INTRGJ*SH.ERRJ $,
COMMENT - - - - - - - NEW METHOD, 12 / 28 / 65, ONLY ADJACENT
 PTS $,
       REAL SAFORX,SAFORY,SLFORX,SLFORY $,
       SLFORX = FORI*COSX(STB) ... SHAPE LENGTH FORCE $,
       SLFORY = FORI*SINX(STB) $,
       SAFORX = -FORJ*SINX(STB) ... ANGLE FORCE $,
       SAFORY = FORJ*COSX(STB) $,
      FORX(PSTB) = FORX(PSTB)-0.5*SAFORX ... ADD FORCES
                   SUCH THAT SUM IS ZERO $,
       FORY(PSTB) = FORY(PSTB)-0.5*SAFORY $,
       FORX(STB) = FORX(STB)+SAFORX+SLFORX $,
       FORY(STB) = FORY(STB)+SAFORY+SLFORY $,
       FORX(NSTB) = FORX(NSTB)-0.5*SAFORX-SLFORX $,
       FORY(NSTB) = FORY(NSTB)-0.5*SAFORY-SLFORY $,
       IF NEXT(STB = NSTB) NEQ 0 ... SKIP LAST BEAD //
       THEN GOTO MUSHLP1 $,
COMMENT - - - FINALE - - TAKES FORCES FROM FORX,Y AND
 UPDATES SVX,Y USING DAMPING AND MASSES. SECOND ORDER
 SYSTEM. THEN UPDATES SPOSX,Y $,
       FOR STB = FRSTSTB,NEXT(STB) WHILE STB NEQ 0 ... ALL
                    BDS //
       DO  BEGIN
           SVX(STB) = SVX(STB)+((FORX(STB)-DAMP*SVX(STB))/
            MASS)*TMINC $,
          SVY(STB) = SVY(STB)+((FORY(STB)-DAMP*SVY(STB))/
            MASS)*TMINC $,
           SPOSX(STB) = SPOSX(STB)+SVX(STB)*TMINC $,
           SPOSY(STB) = SPOSY(STB)+SVY(STB)*TMINC $,
           END $,
       END          ... OF RUNTOT() $,

   END FINI
```

Procedure BARRIE mechanizes physical impenetrability for the simulation. It is used for the tongue and the hard palate, and the upper lip and the lower lip. It allows the two parts in question to slide along each other when in contact but not to penetrate. It also "listens" for contact between the two parts of interest, and, if requested to do so, makes this fact known when it occurs.

```
    BEGIN
 COMMENT - - - - BARRIER PENETRATION PREVENTATION - - - $,
    .INSERT .COMDA $,
    INTEGER IFTOUCHGO ... LOC OF WHO WANTS TO KNOW WHEN ANY
                     BARRIER IS HIT, SET BY P. SETCONTACT $,
    PRESET IFTOUCHGO = 0 $,

    DEFINE PROCEDURE BARRIE(FRSTSTB,FRSTBB) ... //
     WHERE INTEGER FRSTSTB,FRSTBB ... FIRST BEAD OF STATE,
                     BARRIER. BARRIER MAY BE A STATE PART //
     TOBE
        BEGIN
        INTEGER STB, ... STATE POSITION //
         LB,B,NB    ... LAST,PRESENT,NEXT BARRIER PNT $,
        REAL COMPONENT BARRX,BARRY ... X AND Y BARRIER DATA
                     $,
        BARRX $=$ 1 ... MUST BE SAME AS BOTH SPOSX,Y AND
                     STORED BARRIERS $,
        BARRY $=$ 2 $,
        BOOLEAN NOWTOUCH $,
 COMMENT - - - - - CHECKS EACH STATE POINT BY FINDING THE 2
  NEAREST BARRIER POINTS, ASSUMES MONOTONIC TRAVEL ALONG BOTH
  $,
        REAL SQDISL,SQDISP,SQDISN, ... SQUARED DISTANCE
                     BETWEEN LAST,PRESENT, AND NEXT BARRIER
                     PNT AND PRESENT STATE PNT //
        BTOSX,BTOSY, ... BARRIER TO STATE POSITION //
         NBTOSX,NBTOSY, ... NEXT B TO STATE POSITION //
         BVECX,BVECY, ... ROTATED BARRIER SEGMENT //
         DOTPROD,   ... DOT PRODUCT //
         SCALEF,VTANGT $,
        B = FRSTBB  ... GET BARRIER START $,
        NOWTOUCH = FALSE $,
        STB = FRSTSTB $,
        B = NEXT(LB = B) $,
        NB = NEXT(B) $, ... LB,B,NB START AT FIRST 3 BAR PNTS
        //
NXTSP $ SQDISL = (BTOSX = SPOSX(STB)-BARRX(LB))*BTOSX
                    ... //
         +(BTOSY = SPOSY(STB)-BARRY(LB))*BTOSY $,
        SQDISP = (BTOSX = SPOSX(STB)-BARRX(B))*BTOSX
                    ... //
         +(BTOSY = SPOSY(STB)-BARRY(B))*BTOSY $,
        IF NB EQL 0 ... NO MORE BARRIER, USE LAST SEG OF IT
            //
        THEN GOTO BARSET $,
BARADV $ SQDISN = (NBTOSX = SPOSX(STB)-BARRX(NB))*NBTOSX+(
         NBTOSY = SPOSY(STB)-BARRY(NB))*NBTOSY $,
        IF SQDISN LES SQDISL ... ADVANCE ALONG BARRIER //
        THEN BEGIN
            SQDISL = SQDISP $,
            SQDISP = SQDISN $,
            BTOSX = NBTOSX $,
```

```
            BTOSY = NBTOSY $,
            B = NEXT(LB = B) $,
           IF (NB = NEXT(B)) EQL 0
            THEN GOTO BARSET ... RAN OUT OF BARRIER, USE LAST
                     SEGMENT FOREVERMORE //
            ELSE GOTO BARADV ... NEXT BARRIER SEGMENT $,
            END $,
BARSET $ BVECX = -BARRY(B)+BARRY(LB) ... BVEC IS BARRIER
                     SEGMENT ROTATED 90 DEGS CCW $,
        BVECY = BARRX(B)-BARRX(LB) $,
        DOTPROD = BTOSX*BVECX+BTOSY*BVECY $,
        IF DOTPROD LES 0. ... HAVE PENETRATED BARRIER. BRING
                     BACK ALONG A LINE PERPENDICULAR TO
                    BARRIER SEGMENT. //
        THEN BEGIN
            NOWTOUCH = TRUE ... FOR STATUS SENSING $,
            SCALEF = DOTPROD/(BVECX*BVECX+BVECY*BVECY) $,
            SPOSX(STB) = SPOSX(STB)-SCALEF*BVECX $,
            SPOSY(STB) = SPOSY(STB)-SCALEF*BVECY $, ... //
                   ... SETS PERPENDICULAR COMPONENT OF
                     VELOCITY = 0. ALLOWS TANGENTIAL
                   COMPONENT TO REMAIN. NOT PERFECTLY
                     CONSISTENT WITH STATF POSITION
                     MODIFICATION , BUT SO WHAT $,
            SVX(STB) = COSX(STB)*(VTANGT = SVX(STB)*COSX(STB)
             +SVY(STB)*SINX(STB)) $,
            SVY(STB) = SINX(STB)*VTANGT $,
            END      ... OF PENETRATION FIXUP $,
        IF (STB = NEXT(STB)) NEQ 0
        THEN GOTO NXTSP $,
 COMMENT - - - SEE IF JUST HIT A BARRIER ON THIS PASS $,
        IF NOWTOUCH AND IFTOUCHGO NEQ 0 ... SOMEONE WANTS TO
                     KNOW //
        THEN BEGIN
            PROCEDURE CAUSIT $,
            CAUSIT(IFTOUCHGO,TIME) ... EFFECTS A DOIT TO
                     IFTOUCHGO NOW $,
            IFTOUCHGO = 0 ... TURN OFF UNTIL NEXT CALL TO
                     SETCONTACT $,
            END $,
        END          ... OF BARRIE $,


    DEFINE PROCEDURE SETCONTACT(WHERE.TO) WHERE INTEGER
     WHERE.TO TOBE
        IFTOUCHGO = WHERE.TO $,

    END FINI
```

Procedure INITMD, Initializes the mandible.

Procedure RUNJ, mandible state advancement, also relocates the lower lip target if necessary.

Procedure MANDPL, used to produce a display of the mandible when desired.

```
    BEGIN
 COMMENT - - - MANDIBLE AND LOWER LIP PROCEDURES INITMD,
  RUNJ, MANDPL. $,
    .INSERT .COMDA $,
    INTEGER FIRSTLLPSB ... SCHWA LOWER LIP PSB $,
    REAL PROCEDURE SQRT $,
    REAL TX,TY $,
    REAL JTRGOD,    ... JAW TARG OPEN DIS //
     LFTX,LFTY,     ... LOWER FRONT TEETH //
     JPIVOTX,JPIVOTY, ... JAW PIVOT POINT //
     RLXLFTX,RLXLFTY, ... RELAXED LFT POSITION //
     RDLFTD,RDLFTX,RDLFTY, ... RADIUS VECTOR //
     J.THETA,COSTHETA ... JAW ANGLE W.R.T. RELAXED $,
    REAL ARRAY JAWARX,JAWARY, ... JAW OUTLINE //
     PERKJX,PERKJY  ... PERKELL'S LANDMARKS $,
    PRESET
        BEGIN
        JPIVOTX = -5.5 $,
        JPIVOTY = 5.2 $,
        RLXLFTX = 5.0 $,
        RLXLFTY = 2.8 $,
        JAWARX = -4.2,-.7,-.5,0.,.2,-.8,-.6,-1.,-3.1,-6.3,-
         8.4 $,
        JAWARY = .1,0.,-.5,0.,-.4,-2.4,-5.,-5.7,-5.5,-3.6,-3.
         $,
        PERKJX = -5.4,.2 $,
        PERKJY = -6.9,-2.2 $,
        END $,

    DEFINE PROCEDURE INITMD TOBE
        BEGIN
        .INSERT .PSCOM $,
        INTEGER PROCEDURE LOCPS $,
 COMMENT - SETUP SOME GLOBAL VARIABLES WHICH ARE CONSTANT FOR
  AN ENTIRE RUN. $,
        RDLFTD = SQRT((RDLFTX = RLXLFTX-JPIVOTX)*RDLFTX+(
         RDLFTY = RLXLFTY-JPIVOTY)*RDLFTY) $,
        FIRSTLLPSB = PSFIRST(LOCPS(.BCD. /SCHWA/,2,NSHW2)) $,
        JPOSOD = .7 ... AND A STATE INITAL VALUE $,
        GOTO RETURN $,
NSHW2 $ WFLX(.BCI. /INITMD ERROR/) $,
        END         ... OF INITMD() $,

    DEFINE PROCEDURE RUNJ TOBE
                    ... RUN JAW EACH TMINC //
```

```
        BEGIN
COMMENT - - - CALC. A MEASURE OF TONGUE POSITION AND USE IT
 TO DETERMINE JAW TARGET. $,
        INTEGER NTPS,STB,PSB $,
        NTPS = 9 $,
       TX = 0. $,
        TY = 0. $,
        FOR STB = FIRSTSTB(0),NEXT(STB) WHILE STPI(STB) LEQ
         NTPS       ... FRONT OF TONGUE //
        DO  BEGIN
            TX = TX+SPOSX(STB) $,
            TY = TY+SPOSY(STB) $,
            END $,
        TX = TX/NTPS $,
        TY = TY/NTPS $,
COMMENT - - - CALC. CURRENT JTRGOD. NOT A TRUE 'TARGET' IN
 THE TONGUE SENSE OF THE WORD. $,
        JTRGOD =
        IF TY GRT 4.2
        THEN 0.
        ELSE(4.2-TY)*.65 $,
COMMENT - - - RUN JAW. $,
        REAL DWNBP,JDIFF $,
       PRESET DWNBP = .2 $,
        IF (JDIFF = JTRGOD-JPOSOD) LES DWNBP
        THEN JPOSOD = JPOSOD+JDIFF*(TMINC/JTAU)
        ELSE        ... FORCE DOWN //
             JPOSOD = JTRGOD-DWNBP $,
COMMENT - - RELOCATE LOWER LIP TARGET IN ACCORDANCE WITH JAW
 STATE POSITION IFF NOTHING SIGNIFICANT IS ON THERE. I.E.,
 PHM = 1 $,
        IF TRGPHM(STB = FIRSTSTB(2)) NEQ 1
        THEN GOTO RUNJOUT ... SKIP THE FOLLOWING RELOCATION
                     $,
        .INSERT .PSCOM $,
        REAL LL.THETA,LCOSTHETA,RX,RY $,
        LL.THETA = -.8*JPOSOD/RDLFTD ... .8 SINCE L LIP RIDES
                     UP W.R.T. MANDIBLE AS MANDIBLE IS
                     LOWERED $,
        LCOSTHETA = 1.-LL.THETA*LL.THETA/2. $,
        PSB = FIRSTLLPSB $,
        FOR STB = FIRSTSTB(2),NEXT(STB) WHILE STB NEQ 0
        DO  BEGIN
            RX = PSX(PSB)-JPIVOTX $,
            RY = PSY(PSB)-JPIVOTY $,
            TRGX(STB) = JPIVOTX+RX*LCOSTHETA-RY*LL.THETA $,
            TRGY(STB) = JPIVOTY+RX*LL.THETA+RY*LCOSTHETA $,
            PSB = NEXT(PSB) $, ... SHAPE, I.E., TRGI,J LEFT
                     UNCHANGED $,
            END $,
RUNJOUT $
        END         ... OF RUNJ() $,
```

```
   DEFINE INTEGER PROCEDURE MANDPL ... MANDIBLE PLOT //
    TOBE
       BEGIN
       .INSERT .KLDP $,
       INTEGER PROCEDURE XTOH,YTOV $,
       INTEGER H,V,NH,NV,I $,
COMMENT - - - CALC. PRESENT VALUE OF GLOBAL VARIABLES $,
       J.THETA = -JPOSOD/RDLFTD ... SIGN FOR STANDARD ANGLE
                    DIRECTION CONVENTION $,
       COSTHETA = 1.-J.THETA*J.THETA/2. ... SMALL ANGLE
                    APPROX $,
       LFTX = JPIVOTX+(RDLFTX*COSTHETA-RDLFTY*J.THETA) $,
       LFTY = JPIVOTY+(RDLFTX*J.THETA+RDLFTY*COSTHETA) $,
       DEFINE PROCEDURE JAWPLACE(X1,Y1,H,V) WHERE REAL X1,Y1
        $,         ... INPUT IN JAW FRAME //
        INTEGER H,V ... OUTPUT IN H,V //
       TOBE
           BEGIN
           REAL X2,Y2 $,
           X2 = X1*COSTHETA-J.THETA*Y1 ... ROTATE ABOUT
                    LOCAL ORG $,
           Y2 = X1*J.THETA+Y1*COSTHETA $,
          H = XTOH(LFTX+X2) ... TRANSLATE AND CONVERT $,
           V = YTOV(LFTY+Y2) $,
           END      ... OF JAWPLACE $,
       DOBJS() $,
       DOBJAW(STD.STC) $,
       DOBJAW(MASEPOI(XTOH(JPIVOTX),YTOV(JPIVOTY)))
                  ... JAW PIVOT LOC. $,
       QTODOB(.BCQ. '/PIVOT') $,
       JAWPLACE(JAWARX,JAWARY,H,V) ... START JAW SHAPE $,
       DOBJAW(MASEPOI(H,V)) $,
       FOR I = 1 STEP 1 UNTIL 10
      DO  BEGIN
          JAWPLACE(JAWARX(I),JAWARY(I),NH,NV) $,
           DOBJAW(MALIGEC(NH-H,NV-V)) $,
           H = NH $,
          V = NV $,
           END $,
       FOR I = 0,1 ... LANDMARKS //
       DO  BEGIN
           JAWPLACE(PERKJX(I),PERKJY(I),H,V) $,
           DOBJAW(MASEPOI(H,V)) $,
           DOBJAW(MALIGEC(0,-25)) $,
           END $,
       MANDPL = DOBJPT() $,
       END         ... OF MANDPL() $,

   END FINI
```

The following group of programs pertains to the higher level "events" and are executed only when the status of the goals needs to be changed. A new goal may be set, an older goal may be replaced by a later one, or a goal may be forgotten. It should be remembered that there is always one dynamic phoneme type program (CVPROG) which is in command at any one time. First there are programs used by all the CVPROG's, and they are

PRVUNPHM, preview next phoneme.

STRTNCV, start next phoneme (CV).

FINIPCV, finish past phoneme.

```
    BEGIN
    PROCEDURE INITPHM,STRTNCV,NXTCV,FINIPCV,PHMDIS ...
                     DEFINED HERE $,
    INTEGER PROCEDURE PRVUNPHM ... DEFINED HERE $,
    .INSERT .COMDA $,
    .INSERT .CVCOM $,
    INTEGER PROCEDURE LOCCV,LNKBD,UNLNKBD,FREZ $,
    PROCEDURE FRET,DOIT $,
 COMMENT - - - - CVTYPES - - - - - - - - //
   0. VOWEL //
   1. STOP CONSONANT //
   $,
 COMMENT - - - CV PROG STARTING LOCS, FOR TRANSFERS VIA
  STRTNCV(). $,
    PROCEDURE VWLPRG,STPCNS $,
    INTEGER ARRAY LCVPROG $,
    PRESET LCVPROG = VWLPRG,STPCNS $,
   DEFINE PROCEDURE FORCTV TOBE
                    ... FORCE INTO T.V. //
         BEGIN
        VWLPRG() $,
        STPCNS() $,
        END $,
 COMMENT - - ARRAY CKCVSEQU CHECKS FOR CV SEQUENCE
  ACCEPTABILITY. //
   ARRAY INDEX = CURRENT CV TYPE //
   BIT-NUMBERS = ALLOWABLE FOLLOWING CV TYPES. $,
    INTEGER ARRAY CKCVSEQU $,
    PRESET
        BEGIN
        CKCVSEQU(0) = 600000C18 ... VOWEL - - VOWEL, STOP $,
```

```
        CKCVSEQU(1) = 400000C18 ... STOP - - VOWEL $,
        END $,
    SYNONYMS
        1 = CVTYPEMAX $,
    INTEGER CURPHM,NXTPHM ... GLOBAL PHM PTRS $,

    DEFINE PROCEDURE INITPHM ... INITIALIZE PHMLIST //
     TOBE
        BEGIN
        PHMLIST = FREZ(1) $,
        NEXT(PHMLIST) = PHMLIST $,
        PREV(PHMLIST) = PHMLIST $,
        CURPHM = LNKBD(FREZ(PHMBDSZ),PREV(PHMLIST),PREV,NEXT
         )          ... NEED A NON NIL CURPHM TO GET STARTED
                    $,
        PHMCVHD(CURPHM) = LOCCV(.BCD. /ALLPLX/) $,
        END $,

   DEFINE INTEGER PROCEDURE PRVUNPHM ... PREVIEW NEXT PHM
                     (CV) //
     TOBE
        BEGIN
        NXTCV()     ... COMMON PART OF PRVUNPHM AND STRTNCV
                    $,
        PRVUNPHM = NXTPHM ... NORMAL RETURN $,
       END $,

    DEFINE PROCEDURE STRTNCV ... START NEXT CV //
     TOBE
        BEGIN
        NXTCV() $,
        CURPHM = NXTPHM ... ADVANCES CURPHM $,
        DOIT(LCVPROG(CVTYPE(PHMCVHD(CURPHM))),CURPHM)
                    ... NEVER RETURNS FROM DOIT. CVPROG MUST
                     EVTRTN(). SUPPLY CURPHM ARG TO NEW
                     CVPROG $,
        END $,

    DEFINE PROCEDURE NXTCV ... PROCEDURE COMMON TO STRTNCV
                     AND PRVUNPHM //
     TOBE
        BEGIN
        INTEGER NXTCVTYPE $,
        INTEGER PROCEDURE STOPSHOW $,
        BOOLEAN PROCEDURE BITTES $,
TRYNXTPHM $
        IF (NXTPHM = NEXT(CURPHM)) EQL PHMLIST ... NO MORE
            //
        THEN BEGIN
            WFLX(
             .BCI. /PHMLIST END REACHED, FAULT TO STOPSHOW/)
             $,
            WFLX(.BCI. /PLEASE ADD PHMS/) $,
```

```
            STOPSHOW() $,
            GOTO TRYNXTPHM $,
            END $,
        PHMCVHD(NXTPHM) = LOCCV(PHMNAME(NXTPHM),NOPHMCV)
                   ... IS IT KNOWN $,
        IF PHMTMD1(NXTPHM) EQL 0. ... NOT INPUTTED, SO
                    DEFAULT TO CV //
        THEN PHMTMD1(NXTPHM) = CVTMD1(PHMCVHD(NXTPHM)) $,
        IF PHMTMD2(NXTPHM) EQL 0.
        THEN PHMTMD2(NXTPHM) = CVTMD2(PHMCVHD(NXTPHM)) $,
        IF (NXTCVTYPE = CVTYPE(PHMCVHD(NXTPHM))) GRT
         CVTYPEMAX
        THEN BEGIN
            MWFLX(PHMNAME(NXTPHM),.BCI. / CVTYPE UNKNOWN./)
             $,
           GOTO BADPHM $,
            END $,
        IF NOT BITTES(CKCVSEQU(CVTYPE(PHMCVHD(CURPHM))),
         NXTCVTYPE) ... CHECK FOR AN ACCEPTABLE SEQUENCE.
                     ALL CVPROGS ASSUME IT IS OK //
        THEN BEGIN
            MWFLX(PHMNAME(CURPHM),PHMNAME(NXTPHM),
            .BCI. / - AN UNLEARNED TYPE SEQUENCE/) $,
            GOTO BADPHM $,
            END $,
        PHMDIS()    ... DISPLAY NEW ACTIVE CV LIST $,
        GOTO RETURN $,
NOPHMCV $ MWFLX(.BCI. /HAVE NOT LEARNED /,PHMNAME(NXTPHM))
         $,
BADPHM $ MWFLX(PHMNAME(NXTPHM),
         .BCI. / EXPUNGED FROM PHMLIST/) $,
        FRET(PHMBDSZ,UNLNKBD(NXTPHM,PREV,NEXT)) ... DO
                     EXPUNGING $,
        GOTO TRYNXTPHM $,
        END         ... OF NXTCV $,

    DEFINE PROCEDURE FINIPCV ... FINISH ALL PAST CV'S. MUST
                    BE CALLED ONCE BY EACH CVPROG. USUALLY
                     JUST DOES DISPLAY AND PHMLIST
                     BOOKKEEPING, BUT SOMETIMES MAY ALSO
                     FORGET REMNANTS OF PAST CV'S AS WELL.
    //
     TOBE
        BEGIN
        PROCEDURE SETPGS $,
        INTEGER PRVPHM $,
        IF PREV(PRVPHM = PREV(CURPHM)) NEQ PHMLIST ... //
             OR PRVPHM EQL PHMLIST
        THEN WFLX(.BCI. /FINIPCV ERROR 1/) $,
        FRET(PHMBDSZ,UNLNKBD(PRVPHM,PREV,NEXT)) ... FLUSH
                     PRVPHM $,
        SETPGS(1,LOCCV(.BCD. /ALLPLX/),CURPHM,NXTPHM)
                    ... ALL PARTS LAX CVOLOP $, ... N.B. PHM
```

```
                  = 1, AND INHIBITORY PHMS $,
    PHMDIS()    ... SHOW NEW REDUCED ACTIVE CV LIST $,
    END         ... OF FINIPCV() $,

DEFINE PROCEDURE PHMDIS ... DISPLAY ACTIVE PHM STATUS //
 TOBE
    BEGIN
    PROCEDURE SWORG,RMV,PRTA,ASMBCD $,
    INTEGER PROCEDURE PLOT,SWFINI $,
    INTEGER PHMDISP,DIS.R,PHM $,
    PRESET PHMDISP = DIS.R $,
   RMV(PHMDISP) $,
    SWORG(-400,-200) $,
    PRTA(0,.C. /CVS NOW ACTIVE ARE /) $,
    PHM = PHMLIST $,
    FOR PHM = NEXT(PHM) WHILE PHMCVHD(PHM) NEQ 0 AND PHM
     NEQ PHMLIST
    DO ASMBCD(0,PHMNAME(PHM)) $,
    PLOT(SWFINI(),PHMDISP) $,
    END $,
END FINI
```

The specific CVPROGS, one for each dynamic phoneme type.

VWLPRG, vowel program.

STPCNS, stop consonant.

```
    BEGIN
    PROCEDURE STPCNS,VWLPRG ... CVPROGS DEFINED HERE $,
    PROCEDURE EVTRTN,STRTNCV,FINIPCV,SETPGS,SETCONTACT $,
    INTEGER PROCEDURE CAUSIT,PRVUNPHM $,
    .INSERT .COMDA $,

    DEFINE PROCEDURE STPCNS(THSPHM) ... STOP .ONSONANT,
                      INITIAL ENTRY FROM STRTNCV //
     WHERE INTEGER THSPHM TOBE
        BEGIN
        INTEGER CURPHM,NXTPHM $,
        CURPHM = THSPHM $,
       SETPGS(CURPHM,PHMCVHD(CURPHM)) ... SET UP PARTS OF
                     INTEREST, LEAVE REST AS BEFORE $,
        SETCONTACT(LOC STPCNS2) ... WAIT FOR CONTACT, SETS A
                      CAUSIT TO STPCNS2 $,
        EVTRTN() $,
STPCNS2 $            ... START NEXT VOWEL WHERE THIS CONSONANT
                      NOT ACTIVE $,
        NXTPHM = PRVUNPHM() ... PREVIEW NEXT PHM $,
        SETPGS(NXTPHM,PHMCVHD(NXTPHM),CURPHM) ... SET NEXT
                      EXCEPT WHERE THIS ONE $,
        FINIPCV()    ... FORGET ALL EXCEPT THIS AND NEXT $,
        CAUSIT(LOC STPCNS3,TIME+PHMTMD1(CURPHM)) $,
        EVTRTN() $,
STPCNS3 $            ... BEGIN RELEASE, I.E., FORGET ALL AND
                      ON TO NEXT EXCEPT FOR MANNER OF RELEASE
                      $,
        STRTNCV()    ... GOTO NEXT CVPROG VIA STRTNCV $,
        END          ... OF STPCNS() $,


    DEFINE PROCEDURE VWLPRG(CURPHM) ... VOWEL CVPROG //
     WHERE INTEGER CURPHM TOBE
        BEGIN
        SETPGS(CURPHM,PHMCVHD(CURPHM)) $,
        FINIPCV()    ... FORGET ALL OTHER REMNANTS $,
        CAUSIT(LOC VWLPRG2,TIME+PHMTMD1(CURPHM)) $,
        EVTRTN() $,
VWLPRG2 $            ... ENOUGH OF THIS VOWEL $,
        STRTNCV() $,
        END          ... OF VWLPRG() $,

    END FINI
```

Procedure SETPGS sets part goals for the nonrigid structures. Note the optional inhibitory phonemes for calls on SETPGS.

```
    BEGIN
 COMMENT - - - PROCEDURE SETPGS - SET PART GOALS - SETS NEW
  GOALS FOR ALL PARTS OF A CV EXCEPT WHERE TRGPHM IS A STILL
  ACTIVE PHM. I.E., THESE STILL ACTIVE PHMS ACT AS INHIBITORY
  PHMS FOR THE CALL TO SETPGS. $,
    .INSERT .COMDA $,
    PROCEDURE SETPGS ... DEFINED HERE $,
        BEGIN
       .INSERT .PSCOM $,
       .INSERT .CVCOM $,
        INTEGER CVHD,CVPB,PSHD,PSB,STB,PI,N $,
        INTEGER PROCEDURE ISARGP $,

        DEFINE PROCEDURE SETPGS(PHM,CVHD) ...
                     INHIBITPHM1*,INHIBITPHM2*, .. ) //
         WHERE INTEGER PHM,CVHD ... CVHD = PHMCVHD(PHM)
                     EXCEPT WHEN CALLED BY FINICV(), THEN PHM
                     = 0 //
         TOBE
            BEGIN
           PROCEDURE SETPG1 ... SET 1 POINT $,
            FOR CVPB = CVFIRST(CVHD),NEXT(CVPB) WHILE CVPB
             NEQ 0  ... SCAN PARTS OF INTEREST //
            DO  BEGIN
                PSHD = CVPSHD(CVPB) $,
                PSB = PSFIRST(PSHD) $,
                PI = PSPILB(PSHD) $,
                FOR STB = FIRSTSTB(CVPSPN(CVPB)),NEXT(STB)
                 WHILE STB NEQ 0 ... SCAN STATE OF ONE PART
                    //
                DO  IF STPI(STB) EQL PI
                    THEN BEGIN
                       N = 2 ... SETUP TO SCAN FOR OPTIONAL
                     INHIBITORY PHMS $,
INHIBTST $                IF TRGPHM(STB) EQL ISARGP(RETURN,(N =
                           N+1),OKTOSET)
                        THEN GOTO NXTPT ... AN INHIBIT, DON'T
                     OVERWRITE //
                        ELSE GOTO INHIBTST ... CONTINUE
                     INHIBIT SCAN $,
OKTOSET $                 SETPG1() ... SO A NEW GOAL FOR THIS
                     POINT $,
NXTPT $                   IF (PI = PI+1) LEQ PSPIUB(PSHD)
                        THEN PSB = NEXT(PSB) ... STEP ALONG
                     DATA //
                        ELSE GOTO CONT1 ... SKIP REMAINDER OF
                     STATE SCAN FOR THIS PART $,
```

```
                        END $,
CONT1 $         END ... OF PART SCAN $,

            DEFINE PROCEDURE SETPG1 ... SET ONE GOAL POINT //
             TOBE
                BEGIN
                REAL PROCEDURE SQRT $,
                REAL REFX,REFY,REFMAG,RCOS,RSIN,NXTDX,NXTDY
                $,
                TRGPHM(STB) = PHM ... RECORD PHM NOW ON AT
                     THIS LOCALE $,
                TRGX(STB) = PSX(PSB) $,
                TRGY(STB) = PSY(PSB) $,
                TMLSTSETG(STB) = TIME ... FOR MUSCLE LETHARGY
                     $,
               FPVFORX(STB) = VFORX(STB) ... I.E., REMEMBER
                     LAST FORCE AND FADE IT OUT $,
                FPVFORY(STB) = VFORY(STB) ... WHILE TURNING
                    ON THE NEW $,
                IF STPI(STB) NEQ PSPILB(PSHD)
                THEN BEGIN ... SET INCREMENTAL, BUT NOT AT
                     PILB //
                     REFX = PSX(PSB)-PSX(PREV(PSB)) $,
                    REFY = PSY(PSB)-PSY(PREV(PSB)) $,
                    REFMAG = SQRT(REFX*REFX+REFY*REFY) $,
                    RCOS = REFX/REFMAG $,
                    RSIN = REFY/REFMAG $,
                    NXTDX = PSX(NEXT(PSB))-PSX(PSB) $,
                    NXTDY = PSY(NEXT(PSB))-PSY(PSB) $,
                    TRGI(STB) = NXTDX*RCOS+NXTDY*RSIN $,
                    TRGJ(STB) = -NXTDX*RSIN+NXTDY*RCOS $,
                    END $,
 COMMENT - - - SET MANNER, REMEMBERS PREVIOUS RELEASE MODE
  FOR NEXT REPERENCE, THEN FORGETS IT. THUS CAN FORCE
  FORGETTING BY CALLING TWICE. $,
               IF CVPREL(CVPB) ... SET RELEASE BIT AND USE
                     NEW VFORMG //
                THEN BEGIN
                    VFORMGRB(STB) = TRUE $,
                   GOTO DOVFORMG $,
                    END
                ELSE ... NEW ONE NOT RELEASE //
                     IF VFORMGRB(STB) ... LAST ONE WAS
                     RELEASE //
                    THEN BEGIN
                        VFORMGRB(STB) = FALSE $,
                        IF PSF(PSB) GRT VFORMG(STB) ... USE
                     NEW ONLY IF STRONGER THAN LAST //
                        THEN GOTO DOVFORMG
                        ELSE GOTO OUT1 $,
                        END $,
DOVFORMG $         ... ALSO FALL TO HERE IF NOT NEWRELEASE
                     AND NOT OLDRELEASE $,
```

```
                VFORMG(STB) = PSF(PSB) $,
                VOPTON.TI(STB) = CVPOPTON.TI(CVPB) $,
OUT1 $          END ... OF SETPG1() $,

           END     ... OF SETPGS() $,

       END         ... OF .BBCOM BLOCK ISOLATION $,
    END FINI
```

LOCCV handles the CVTAB level of memory.

```
    BEGIN
    .INSERT .CVCOM $,
    .INSERT .PSCOM  ... ONLY USED FOR THE NEXT,PREV $,
    INTEGER CVHD,FIRSTCVHD,CVPB $,
    PROCEDURE WFLX,MWFLX,ISARGD $,
    DEFINE PROCEDURE INITCV ... READ CVTAB FROM DISK, BUILD
                     STRUCTURE //
     TOBE
        BEGIN
        PROCEDURE FSTATE,OPEN,RDWAIT,CLOSE ... FILE SYSTEM $,
        INTEGER PROCEDURE FREZ $,
       INTEGER P,PP,CVBDSZ1,CVBDSZ2 $,
        INTEGER ARRAY CVTABSZ(0) $,
        INTEGER COMPONENT WI $,
        WI $=$ 0 $,
        FENCE .BCDN. /777777777777/ $,
        CVTAB .BCD. / CVTAB/ $,
        NAME2 .BCD. /BINARY/ $,
        FSTATE(CVTAB,NAME2,CVTABSZ TO 1) $,
        P = FREZ(CVTABSZ) $,
        OPEN(.BCD. /R/,CVTAB,NAME2,-0,-0) $,
        RDWAIT(CVTAB,NAME2,0,WI(P)TO CVTABSZ+0,-0,-0) $,
        CLOSE(CVTAB,NAME2) $,
       CVBDSZ1 = WI(P) $,
        CVBDSZ2 = WI(P+1) $,
        P = P+2 $,
        CVHD = FIRSTCVHD = P $,
NXTHD $ IF WI(P) NEQ 1
        THEN GOTO DONECK $,
        NEXT(CVHD) = P ... LINK PRECEDING CVHD TO NEW $,
        CVHD = P $,
       NEXT(CVHD) = 0 ... IN CASE LAST ONE $,
       P = P+CVBDSZ1 $,
        IF WI(P) NEQ 2
        THEN GOTO ERROUT ... REQUIRE AT LEAST 1 CVSUB $,
        CVFIRST(CVHD) = P $,
       NEXT(P) = 0 $,
        PP = P $,
L2 $    P = P+CVBDSZ2 $,
        IF WI(P) NEQ 2
        THEN GOTO NXTHD $,
```

```
        NEXT(PP) = P ... ANOTHER CVSUB $,
        NEXT(PP = P) = 0 ... IN CASE LAST $,
        GOTO L2 $,
DONECK $
        IF WI(P) EQL FENCE
        THEN GOTO RETURN $,
ERROUT $ WFLX(.BCI. /INITCV ERROR/) $,
        END            ... OF INITCV() $,

    DEFINE INTEGER PROCEDURE LOCCV(NAME) ... ERRET*) //
     WHERE INTEGER NAME TOBE
        BEGIN
        INTEGER PROCEDURE LOCPS ... GETS A PSHD $,
       FOR CVHD = FIRSTCVHD,NEXT(CVHD) WHILE CVHD NEQ 0
        DO  IF CVNAME1(CVHD) EQL NAME
            THEN GOTO FOUNDCV $,
        MWFLX(NAME,.BCI. / CV UNKNOWN/) $,
        ISARGD(RETURN,2,RETURN) $,
FOUNDCV $
        IF CVPSHD(CVFIRST(CVHD)) EQL 0 ... FIRST USE, MUST
                     FIND PS'S //
        THEN FOR CVPB = CVFIRST(CVHD),NEXT(CVPB) WHILE CVPB
         NEQ 0
            DO CVPSHD(CVPB) = LOCPS(CVPNAME1(CVPB),CVPSPN(
             CVPB),NOPSLBL) $,
        LOCCV = CVHD $,
        GOTO RETURN $,
NOPSLBL $ MWFLX(.BCI. /PART UNKNOWN FOR CV /,NAME) $,
        ISARGD(RETURN,2,RETURN) $,
        END            ... OF LOCCV() $,

    END FINI
```

LOCPS and PSREAD handle the PS level of memory.

```
    BEGIN
    INTEGER PROCEDURE LOCPS,PSREAD ... DEFINED HERE $,
    .INSERT .PSCOM $,
    INTEGER PROCEDURE LNKBD,LNKFBD,NUMTOD,FREZ,RJUST $,
    PROCEDURE WFLX,MWFLX,OPEN,RDWAIT,CLOSE $,

    DEFINE INTEGER PROCEDURE LOCPS(NAME1,PSPRTN,NLOCPS)
                    ... LOC OF A PSHD //
     WHERE INTEGER NAME1,PSPRTN $,
    LABEL NLOCPS TOBE
        BEGIN
        INTEGER PSHD,PSHDLIST,START $,
       PRESET START = 0 $,
        PRESET PSHDLIST = START $,
        PSHD = PSHDLIST $,
        FOR PSHD = NEXT(PSHD) WHILE PSHD NEQ 0 ... LOOK IN
```

```
                     CORE, MORE THAN ONE CV MAY USE THE SAME
                     PS //
        DO  IF PSNAME1(PSHD) EQL NAME1 AND PSPN(PSHD) EQL
         PSPRTN
            THEN BEGIN
                LOCPS = PSHD $,
                GOTO RETURN $,
                END $,
        PSHD = PSREAD(NAME1,PSPRTN,NOPSLB) ... LOOK ON THE
                     DISC $,
        LNKBD(PSHD,PSHDLIST,PREV,NEXT) ... FOUND, PUT ON LIST
                     $,
        LOCPS = PSHD $,
        GOTO RETURN $,
NOPSLB $ MWFLX(NAME1,NUMTOD(PSPRTN),.BCI. /PS NOT FOUND/) $,
        GOTO NLOCPS $,
        END         ... OF LOCPS() $,

    DEFINE INTEGER PROCEDURE PSREAD(PSNM1,PSPRTN,NOFIND)
                    ... PS READ AND BUILD //
     WHERE INTEGER PSNM1,PSPRTN $,
    LABEL NOFIND TOBE
       BEGIN
        INTEGER PSB,PSHD,I,J,BUFSZ,ERRCOD $,
        INTEGER HDCPYLB,HDCPYUB,PIVAL,NAME1,NAME2 $,
        PRESET BUFSZ = 140 $,
        INTEGER ARRAY BUF(140) $,
        INTEGER COMPONENT WI $,
        REAL COMPONENT WR $,
        WR $=$ WI $=$ 0 $,
        NAME1 = RJUST(PSNM1) $,
        NAME2 = .BCD. /    P0/-PSPRTN $,
        DEFINE PROCEDURE ERREXIT(BCI.STR) WHERE INTEGER
         BCI.STR TOBE
            BEGIN
            MWFLX(.BCI. /PSREAD ERR, FILE /,NAME1,NAME2,
             BCI.STR) $,
            GOTO NOFIND $,
            END      ... OF ERREXIT() $,
        OPEN(.BCD. /R/,NAME1,NAME2,-0,-0,OPNERR,ERRCOD) $,
        GOTO READ1 $,
OPNERR $
        IF ERRCOD EQL 12 ... NO FIND, NO ERRMSG //
        THEN GOTO NOFIND $,
        ERREXIT(.BCI. /OPEN ERR CODE NEQ 12/) $,
READ1 $ RDWAIT(NAME1,NAME2,0,BUF TO BUFSZ,REOF,-0) $,
        ERREXIT(.BCI. / LENGTH GRT BUFSZ/) $,
REOF $  CLOSE(NAME1,NAME2) ... DESIRED EXIT FROM RDWAIT $,
 COMMENT - - - - - BUILD STRUCTURE AND FILL WITH INFO $,
        PSHD = FREZ(PSHDSZ) $,
        HDCPYLB = BUF(1) $,
        HDCPYUB = BUF(I = 2) $,
        FOR J = HDCPYLB STEP 1 UNTIL HDCPYUB
```

```
          DO WI(PSHD+J) = BUF(I = I+1) ... CPY PSHD INFO $,
          PSB = LNKFBD(FREZ(PSBDSZ),PSHD,PSFIRST,PREV,NEXT)
                      ... FIRST DATA BD $,
          PIVAL = PSPILB(PSHD) $,
          I = LOC BUF+I ... NEEDED FOR INTEGER TO REAL FAKING
                       BELOW $,
          GOTO PSR1 $,
PSR2 $    PSB = LNKBD(FREZ(PSBDSZ),PSB,PREV,NEXT) $,
PSR1 $    PSX(PSB) = WR(I = I+1) $,
          PSY(PSB) = WR(I = I+1) $,
          PSF(PSB) = WR(I = I+1) $,
          IF (PIVAL = PIVAL+1) LEQ PSPIUB(PSHD)+1
          THEN GOTO PSR2 $,
          PSREAD = PSHD $,
          END         ... OF PSREAD() $,

      END FINI
```

Remaining programs perform miscellaneous housekeeping and control functions.

Procedure INITAL initializes the model, builds the state structure, turns on the display console, etc.

```
      BEGIN
      PROCEDURE INITAL ... INITIALIZES ALMOST ALL, DEFINED
                       HERE. $,
      .INSERT .COMDA $,
      .INSERT .KLDP $,
      INTEGER PROCEDURE FREZ,LNKBD,COMARG,OCTTOI $,
      INTEGER STPRT,STB,I $,
      INTEGER ARRAY STPRTSZ $,
      PRESET STPRTSZ(0) = 22,6,7 ... NO. PTS IN EACH STATE PART
                       $,

      DEFINE PROCEDURE INITAL TOBE
          BEGIN
          PROCEDURE INITCV,INITPHM,INITMD $,
          INITCV()    ... READ AND LINK CVTAB, USED (ALLPLX) IN
                      STATE INITIALIZATION $,
          INITPHM()   ... START PHMLIST $,
          INITMD()    ... GO SEE THE MANDIBLE $,
          TIME = 0. $,
    COMMENT - - - BUILD THE STATE STRUCTURE. $,
         MWFLX(.BCI. /LOC FIRSTSTB(0)=/,OCTTOI(LOC FIRSTSTB))
                      ... FOR DEBUGGING $,
          FOR STPRT = 0 STEP 1 UNTIL 2
          DO  BEGIN
```

```
           FIRSTSTB(STPRT) = STB = FREZ(STBDSZ) $,
           STPI(STB) = 1 $,
           FOR I = 2 STEP 1 UNTIL STPRTSZ(STPRT)
           DO  BEGIN
               STB = LNKBD(FREZ(STBDSZ),STB,PREV,NEXT) $,
               STPI(STB) = I $,
               END $,
           END $,
COMMENT - - - INITIALIZE THE STATE $,
       INTEGER PROCEDURE LOCCV $,
       PROCEDURE SETPGS $,
       SETPGS(1,LOCCV(.BCD. /ALLPLX/)) ... SET ALL GOALS $,
       FOR STPRT = 0 STEP 1 UNTIL 2 ... AND PUT STATE AT
                    THEM //
       DO  FOR STB = FIRSTSTB(STPRT),NEXT(STB) WHILE STB NEQ
        0  DO  BEGIN
               SPOSX(STB) = TRGX(STB) $,
               SPOSY(STB) = TRGY(STB) $,
               SVX(STB) = SVY(STB) = 0. $,
               END $,
       WFLX(.BCI. /SET DIGI/) $,
COMMENT - - - - ****** START KLUDGE ***** - - - - - - $,
       SGNON(
       IF COMARG(1) EQL .BCD. /     2/
       THEN 2
       ELSE 1,0) $,
       SATBUF(0) $,
       DOBJS() $,
       DOBJAW(STD.STC) ... TURNS OFF PEN INK SEE $,
       DOBJAW(MASEPOI(150,500,0)) $,
       QTODOB(.BCQ. /ART.    FIAT LUX, EXCELSIOR../) $,
       PLOT(DOBJPT()) $,
      INTEGER PROCEDURE BARRP1 ... PLOT PALATE $,
       PLOT(BARRP1()) $,
       END         ... OF INITAL() $,
   END FINI
```

PRSORQ presets certain model parameters, and contains a compiled symble table which allows these parameters to be changed during a simulation run.

```
    BEGIN               ... ORQCL MIGHT JUST BE A PREFACE OF
                         RUNTOT //
 COMMENT - - - ORQCL PRESETS SOME PARAMETERS AND CONTAINS THE
  PSEUDO SYMTABLE FOR INSPECTION OF THESE AND OTHERS VIA CALL
  TO ORQ (OPEN REQUEST) $,
    .INSERT .COMDA $,
    PRESET
        BEGIN
```

```
        FDISBP = .15 ... CM, VOLOP FORCE BRK PT $,
        INTRGI = 2000. ... FORCE / LENGTH,INTEGRITY FORCE
                     FACTOR $,
        INTRGJ = 6500. ... FORCE / RADIAN $,
        MASS = .05 $,
        JTAU = .070 ... JAW TIME CONSTANT $,
        DAMP = 10. $,
        TMINC = .001 $,
        END $,

    DEFINE PROCEDURE ORQCL TOBE
        BEGIN
        PROCEDURE ORQ $,
        ORQ(OUTLB,  ... ALWAYS RETURNS TO FIRST ARG //
        .BCD. /FDISBP/,FDISBP, ... //
         .BCD. /INTRGI/,INTRGI, ... //
         .BCD. /INTRGJ/,INTRGJ, ... //
        .BCD. /  MASS/,MASS, ... //
         .BCD. /  DAMP/,DAMP, ... //
         .BCD. /  TIME/,TIME, ... SIMULATION TIME //
         .BCD. /  JTAU/,JTAU, ... //
         .BCD. /JPOSOD/,JPOSOD, ... JAW STATE POSITION //
         .BCD. / TMINC/,TMINC) $,
OUTLB $ GOTO RETURN $,
        END $,
    END FINI
```

Procedure STOPSHOW is called when it is desired to stop the simulation to view the current state or to communicate control or input information to the model.

```
     BEGIN
     .INSERT .COMDA $,
    .INSERT .KLDP $,

    DEFINE REAL PROCEDURE STOPSHOW ... STOPS THE
                     SIMULATION, SHOWS THE PRESENT STATE,
                     AND RETURNS THE TIME WHEN IT WISHES TO
                    BE CALLED AGAIN. //
      TOBE
         BEGIN
         INTEGER ARRAY CMDAR(14) $,
         INTEGER NWORDS,I,PHM,TEMP $,
         PROCEDURE ORQCL,GDCMD,PICTUR,FRET $,
         INTEGER PROCEDURE NUMTOI,RCMD,FREZ,LJUST,LNKBD,
          UNLNKBD $,
         PICTUR()   ... WHAT HAVE WE NOW $,
         PROCEDURE CAMERA ... IN CASE WE ARE SHOOTING $,
         INTEGER CAMFRAMES ... FRAMES PER CALL TO PICTUR $,
```

```
         PRESET CAMFRAMES = 0 $,
         IF CAMFRAMES NEQ 0 ... WE ARE FILMING //
         THEN CAMERA(CAMFRAMES) ... THESE FRAMES NOW $,
         MWFLX(NUMTOD(I = TIME*100.)) ... FOR THE WRITTEN
                     RECORD $,
MASTER $ NWORDS = RCMD(CMDAR,14,.BCI. /ART./) $,
         IF NWORDS EQL 0 ... JUST RETURN IF NO CMDS //
         THEN GOTO LEAVE $,
         GDCMD(CMDAR,NFMST, ... MASTER TTY CMDS //
          .BCD. /     */,LEAVE, ... LEAVE STOPSHOW //
          .BCD. /    PF/,PF, ... STRT FRESH PH LST //
          .BCD. /    PA/,PA, ... ADD TO PH LIST //
          .BCD. /  PPRT/,PPRT, ... PRINT PH LIST //
          .BCD. /     X/,XDIS., ... DISPLAY XRIN FRAME //
          .BCD. /CAMERA/,CAMERA., ... SET CAMFRAMES //
          .BCD. /   ORQ/,ORQUS) ... OPEN REQUEST //
          $,
NFMST $  MWFLX(CMDAR,.BCI. / FOREIGN TO MASTER/) $,
         GOTO MASTER $,
ORQUS $  ORQCL()    ... LOOK AT AND / OR ADJUST PARAMETERS
                     $,
         GOTO MASTER $,
CAMERA. $ CAMFRAMES = DTONUM(CMDAR(1)) $,
         GOTO MASTER $,
XDIS. $  PROCEDURE XDIS $,
         XDIS(CMDAR(1)) ... CMDAR(1) IS XDIS'S CMD(0) $,
         GOTO MASTER $,
PF $                ... FRESH PHMLIST. EXPUNGE ALL PHMS THAT
                    HAVE NOT YET BEEN SEEN $,
         PHM = PHMLIST $,
         FOR PHM = NEXT(PHM) WHILE PHM NEQ PHMLIST
         DO  IF PHMCVHD(PHM) EQL 0 ... NOT SEEN YET //
             THEN BEGIN
                 PHM = PREV(PHM) ... SINCE PHM BD TOBE
                    FRETTED $,
                 FRET(PHMBDSZ,UNLNKBD(NEXT(PHM),PREV,NEXT))
                 $,
                 END $,
PA $               ... ADD PHMS TO END OF PHLIST $,
         DEFINE INTEGER PROCEDURE NXTCMDW TOBE
             BEGIN
            IF (I = I+1) EQL NWORDS
            THEN GOTO MASTER $,
             IF CMDAR(I) EQL .BCD. '     /'
            THEN GOTO NXTINPHM $,
            IF CMDAR(I) EQL .BCD. /     */
             THEN GOTO LEAVE $,
             NXTCMDW = CMDAR(I) $,
             END $,
         I = 0 $,
NXTINPHM $ TEMP = LJUST(NXTCMDW()) $,
         PHM = LNKBD(FREZ(PHMBDSZ),PREV(PHMLIST),PREV,NEXT)
          $,
```

```
         PHMNAME(PHM) = TEMP $,
        PHMTMD1(PHM) = DTONUM(NXTCMDW())/1000. $,
         PHMTMD2(PHM) = DTONUM(NXTCMDW())/1000. $,
         TEMP = NXTCMDW() ... ERROR IF EVER RETURNS $,
         WFLX(.BCI. /IMPROPER INPUT FORMAT/) $,
         GOTO MASTER $,
PPRT $   WFLX(.BCI. /PRESENT PHMLIST IS/) $,
         PHM = PHMLIST $,
         FOR PHM = NEXT(PHM) WHILE PHM NEQ PHMLIST
         DO MWFLX(PHMNAME(PHM),NUMTOD(I = 1000.*PHMTMD1(PHM)
          )) $,
         GOTO MASTER $,
LEAVE $             ... ALL DONE FOR THIS SHOW, RETURN NEXT
                    TIME $,
         STOPSHOW = TIME+REDIGI(7,9)/1000. $,
         GOTO RETURN $,
         END        ... OF STOPSHOW() $,


     DEFINE PROCEDURE PICTUR TOBE
 COMMENT - - - - - - - - - - - - - - - - - - PICTURE PREPARES PICT
  PARTS ACCORDING TO SETTING OF TOGA, NEEDS .INSERT .COMDA
  $,     BEGIN
         PROCEDURE UPDIS,RMVDIS ... THESE IN TURN CALL
                     B-CORE $,
         BOOLEAN PROCEDURE BITTES $,
         PROCEDURE BITON,BITOFF $,
         INTEGER PCPT,N,I,PRTN $,
         INTEGER PICWNT, ... WANTED PARTES, BITS //
          PICON,    ... NOW IN DISPLAY //
          PICFIX,   ... FIXED PARTS //
          PICMX     ... MAX NO OF PARTS $,
         PRESET PICON = 0 $,
         PRESET PICFIX = 000000200000C $,
         PRESET PICMX = 19 ... 0 THRU 19 $,
         SWITCH PICSW = SPOSP,SPOSP,SPOSP,TRGP,TRGP,TRGP,
          VELP,VELP,VELP,VFORP,VFORP,VFORP,TFORP,TFORP,TFORP
          ,FORMGP,FORMGP,FORMGP, ... 0 TO 17 //
          MANDPLP,AGRIDPL ... 18 AND 19 $,
         INTEGER PROCEDURE TOGA $,
         PICWNT = TOGA() $,
         N = -1 $,
NXTP $   IF (N = N+1) GRT PICMX
         THEN GOTO RETURN $,
         IF BITTES(PICWNT,N) ... WANT A DISPLAY //
         THEN BEGIN
             IF BITTES(PICON,N) AND BITTES(PICFIX,N)
             THEN GOTO NXTP ... ALREADY ON AND NO CHANGE //
            ELSE BEGIN
                 PRTN = N-(N/3)*3 ....IN CASE NEEDED $,
                GOTO PICSW(N+1) $,
                 END
             END
```

```
         ELSE IF BITTES(PICON,N) ... WANT OUT //
             THEN BEGIN
                 RMVDIS(N) $,
                  BITOFF(PICON,N) $,
                 END $,
         GOTO NXTP $,
PUTIN $  UPDIS(PCPT,N) $,
         BITON(PICON,N) $,
         GOTO NXTP $,
         INTEGER PROCEDURE ALINEP,VECP ... PLOT PREPS $,
SPOSP $  PCPT = ALINEP(SPOSX,SPOSY,FIRSTSTB(PRTN)) ...
                      POSITION $,
        IF PRTN EQL 0 ... SHOW TIME WITH TONGUE POSITION //
         THEN BEGIN
             DOBJAW(MASEPOI(-500,460,0)) $,
             QTODOB(.BCQ. /TIME IN MS = /) $,
             QTODOB(NUMTOQ(I = 1000.*TIME)) $,
            END $,
         GOTO PUTIN $,
TRGP $   PCPT = ALINEP(TRGX,TRGY,FIRSTSTB(PRTN)) ... TARGET
                      $,
         GOTO PUTIN $,
VELP $   INTEGER PROCEDURE VRTOHV ... VELOCITY $,
         PCPT = VECP(SPOSX,SPOSY,SVX,SVY,FIRSTSTB(PRTN),
          VRTOHV) $,
         GOTO PUTIN $,
VFORP $  INTEGER PROCEDURE FRTOHV ... VOLIT FORCES $,
         PCPT = VECP(SPOSX,SPOSY,VFORX,VFORY,FIRSTSTB(PRTN),
          FRTOHV) $,
         GOTO PUTIN $,
TFORP $  PCPT = VECP(SPOSX,SPOSY,FORX,FORY,FIRSTSTB(PRTN),
          FRTOHV)   ... TOTAL F'S $,
         GOTO PUTIN $,
FORMGP $            ... MAG OF FORCE OF COMPOS TARG //
          PCPT = VECP(TRGX,TRGY,VFORMG,VFORMG,FIRSTSTB(PRTN)
         ,FRTOHV) $,
         GOTO PUTIN $,
MANDPLP $ INTEGER PROCEDURE MANDPL ... MANDIBLE $,
         PCPT = MANDPL() $,
         GOTO PUTIN $,
AGRIDPL $ INTEGER PROCEDURE AGRIDP ... ARTICULATORY GRID $,
         PCPT = AGRIDP() $,
         GOTO PUTIN $,
         END        ... OF PICTUR() $,

     END FINI
```

# APPENDIX B

## ACOUSTIC TRANSFER FUNCTION CALCULATION FOR THE VOCAL TRACT

In this appendix we present a new algorithm for the calculation of the transfer function (frequency domain) of the vocal tract. It appears to us that this algorithm is significantly more general than those previously reported in the literature. This work is independent of the model which is the subject of the rest of this report, but is included here for the record since it is relevant to the general study of speech production.

As a frequency domain solution the algorithm yields quasi-static spectral characteristics of the acoustic speech waveform. This should be distinguished from a time domain sample data simulation which, using the same input data, would produce a (sampled) acoustical waveform which could be listened to after appropriate digital-to-analog conversion and time buffering.

The acoustic speech wave can be regarded as the output of vocal tract filter systems which are excited by one or more sound sources. Thus in modeling the acoustics of speech production source characteristics are combined with the transmission properties of the vocal tract to yield the output: a spectrum in the frequency domain or a waveform in the time domain. For a more concise format of the results it is often convenient to represent an output spectrum by its equivalent natural frequencies (poles and zeros) which are identified with the well known formant structure of speech.

Using this approach many acoustic aspects of speech have been enunciated by Fant (1960), and he presents numerous characteristics of such a source-filter acoustic theory of speech production.

A major requirement of this theory is the calculation of the pertinent transfer functions of the vocal tract. For commonly used approximations (discussed below) the acoustically significant factor in the determination of transfer functions is an area function which gives the cross sectional area of the vocal tract as a function of position along the tract. The vocal tract configuration is time varying but the frequencies of interest are high enough compared to the time rate of change of the configuration so that a quasi-static approximation can be used with negligible inaccuracy. This allows the calculation of transfer functions based on fixed configurations.

Several methods have been devised for calculating transfer functions from area functions and other requisite parameters (e.g., Fant, 1960; Heinz, 1962). One general approach is to represent the vocal tract by a sequence of analogous lumped electrical elements and to solve the resultant electrical network. Another method uses Webster's horn equation to find the imaginary parts of the natural frequencies.

The algorithm presented here is more general than the previous methods since many of the restrictions to specialized and simplified cases have been removed. It is felt to be more elegant due to its innate simplicity, which also increases the speed of calculation. Before presenting the details of the algorithm we will discuss its general characteristics, and in so doing contrast it to other methods.

The algorithm considers only plane wave propagation in the vocal tract and assumes that all higher modes do not exist or are insignificant. This is a universally used assumption and its justfication, which becomes borderline at the highest audible frequencies, is based upon a comparison of wavelength with vocal tract cross dimensions. At 5000 Hz the wavelength of sound in air is approximately 7 cm, which is beginning to be comparable to the dimensions of interest. At lower frequencies, of course, the validity of the approximation increases.

This approximation allows the vocal tract to be treated as a transmission line. The necessary arguments for the analogy are given by Morse (1948). For the purposes of this method only the complex propagation constant (phase and attenuation) and the characteristic impedance are needed. The analogy can be carried further into analogous L's, C's, R's, and G's, and these are the elements used in some other solutions.

The basic methodology can best be described as a hybrid "lumped" and "distributed" technique based upon the transmission line analogy. The variable area vocal tract is represented by a series of contiguous sections, each of fixed area. The exactness of this representation is of course dependent upon the length quantization interval size, and for most purposes an interval length of several millimeters is adequate. Within each section wave propagation is treated on a distributed basis using the complex propagation constant, as opposed to using lumped elements to approximate each section. Junctions between individual sections, at source and termination points, and at branch points are treated on a lumped basis using (1) continuity of pressure and (2) conservation of volume velocity.

The solution method is quite general. It allows for arbitrary source characteristics and terminal impedances (both real and imaginary part) wherever such lumped characteristics are desired. Sources can be inserted anywhere and transfer functions calculated from any point to any other point. Multiple sources are treated by superposition of standing wave patterns due to individual sources (linearity being assumed throughout, acoustically valid except for very high intensity sound).

Arbitrary topological forms are amenable to solution by the algorithm. The form of the vocal tract will usually be that of a single tube or that of a "Y" (pharynx, oral, and nasal cavities). Cases where the acoustic path branches and remerges symmetrically (perhaps useful for the liquids) are easily treated. The most general topological forms consisting of closed loops, i.e., asymmetrical branching and remerging, are soluble. However these situations require the simultaneous solution of a set of complex equations. Except for this last less prevalent case the method is computationally clean and fast. It does not require the inversion of matrices, nor the iterative multiplication of complex matrices, nor the evaluation of transcendental functions other than real sine and cosine.

The standing wave pattern of volume velocity and/or sound pressure for any frequency is available.

The transfer function (magnitude and phase) is calculated directly as a function of frequency. Natural frequencies, if desired, are determined from the phase spectrum. Other methods first solve for natural frequencies and then calculate the transfer function. This two step technique requires the use of a so-called higher pole correction factor to account for the fact that a distributed system has an infinite number of natural frequencies, of which only the first lower few are calculated. This correction can be made with with reasonable accuracy although not exactly.

Another feature different from most other solutions is that damping is an integral part of the solution, rather than a perturbing term added later. Energy loss along the tract is set by the attenuation constant (db/cm). Other contributions to damping, e.g., the real parts of radiation impedances, are included in their respective impedances. Since the attenuation constant can be a function of both frequency and position, the influence of cross sectional area, shape factor, and wall composition can be included.

## ALGORITHM

The modus operandi of the algorithm consists of assuming arbitrary pressures or volume velocities at all passive terminations, propagating these through the acoustical structure, and then scaling the assumed quantities to satisfy continuity of pressure and conservation of volume velocity at all points.

At any point in the structure a quantity is represented by a sum of forward and backward components, and each of these components is a phasor (a complex quantity).

$$I(z) = I_+(z) + I_-(z)$$

In the following equations the analogous electrical notation will be used because of its familiarity. I is volume velocity (current) and V is pressure (voltage). Unless otherwise stated all appearances of these two symbols are complex. The subscripts of + and - identify direction with respect to a forward direction which is arbitrarily defined for each branch.

Within each section wave components are propagated in accordance with the complex propagation constant $\gamma$.

$$\gamma = \alpha(\omega, z) + j\beta(\omega, z)$$

$$I_+(z+\Delta z) = I_+(z)\, e^{-\gamma \Delta z}$$

$$I_-(z+\Delta z) = I_-(z)\, e^{+\gamma \Delta z}$$

This is computed by a rotation and attenuation (general term including amplification). The direction of rotation and effect of attenuation is opposite on the two components, and the polarity of this process is set by a comparison of the "direction of travel" with the previously defined "forward" direction. The rotation is effected by a two dimensional rotational transformation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \qquad \begin{aligned} \theta &= \beta \cdot \Delta z \\ &= \frac{2\pi f}{c} L \end{aligned}$$

where the angle ($\theta$) is a function of the section length ($\Delta z = L$), the frequency ($f$), and the speed of sound ($c$). The attenuation is effected by a multiplication of the components by factors of $e^{-\alpha L}$ and $e^{+\alpha L}$.

The procedure at a junction between two adjacent segments is as follows:

conservation of volume velocity $\quad I_{1+} + I_{1-} = I_{2+} + I_{2-}$

continuity of sound pressure $\quad Z_1(I_{1+} - I_{1-}) = Z_2(I_{2+} - I_{2-})$

The Z's are the characteristic impedances of the two adjacent sections. These characteristic impedances are inversely proportional to the cross sectional area of the sections (for small loss), and are taken to be $Z = \rho c/A$. Solving for the new components in terms of the old components, we have

$$I_{2+} = \frac{1}{2}\left\{ I_{1+}\left(1+\frac{Z_1}{Z_2}\right) + I_{1-}\left(1-\frac{Z_1}{Z_2}\right)\right\}$$

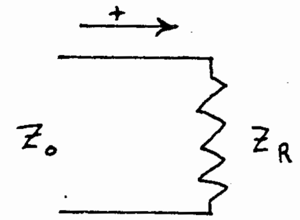
$$I_{2-} = \frac{1}{2}\left\{ I_{1+}\left(1-\frac{Z_1}{Z_2}\right) + I_{1-}\left(1+\frac{Z_1}{Z_2}\right)\right\}$$

All passive terminations of the structure are lumped radiation impedances. These may vary from a "short circuit" (open tube with no radiational loading, physically unrealizable) to an "open circuit" (completely closed tube end). At all these locations an arbitrary volume velocity or pressure is assumed.

$I_{as}$ = assumed volume velocity

$Z_R$ = radiation impedance

$Z_0$ = characteristic impedance of adjacent tube section



Equating "line" quantities to assumed quantities in the radiation impedance

$$I_+ + I_- = I_{as}$$

$$Z_0(I_+ - I_-) = Z_R I_{as}$$

yields upon solution the "outgoing" line quantities.

$$I_+ = \frac{1}{2}\left(1 + \frac{Z_R}{Z_o}\right) I_{as}$$

$$I_- = \frac{1}{2}\left(1 - \frac{Z_R}{Z_o}\right) I_{as}$$

In the case where $Z_R = \infty$ (complete closure) a value of V is assumed and similar equations result. If the forward direction is opposite to that shown here several of the signs change. In general $Z_R$ will be a function of frequency and area. The currently used version is programmed to allow switching of these "loads", and the loads presently extant are (1) a short circuit and (2) a piston in an infinite baffle (Morse, 1948).

At a junction where three branches come together a scaling factor (complex) for one of the two incoming branches is determined so that both incoming branches have equal pressure. Volume velocities are then added and a variation of the normal junction procedure is applied to determine the outgoing components. The algebraic manipulation is straightforward and will not be presented here.

Other situations of interest are sources at terminations, intermediate sources, symmetrical and asymmetrical branching and remerging. In all cases the procedure used to derive the final equations is the same. Equations containing one or more scaling factors are written for pressure and volume velocity. These equations are solved for the scaling factor(s), the necessary adjustments made, and travel along the structure continued until the solution is finished.

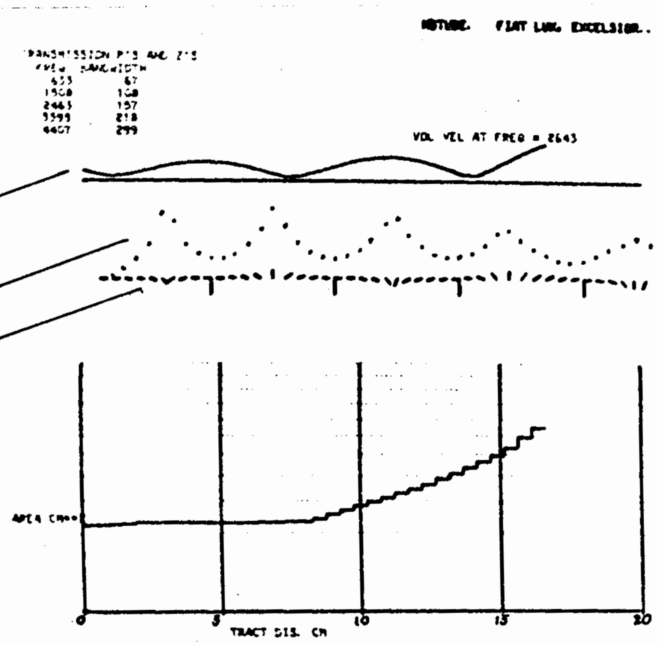
The currently programmed version of algorithm is for a single tube with the source at one end. Area functions are entered and modified graphically with the light pen. Other parameters such as tube length, losses, and loads are changed via the teletype. Fig. B.1 shows some but not all of the items which may be displayed.

pole (formant)
frequencies and bandwidths

TRANSMISSION P'S AND Z'S

| FREQ | BANDWIDTH |
|---|---|
| 633 | 67 |
| 1508 | 108 |
| 2463 | 157 |
| 3399 | 218 |
| 4407 | 299 |

HSTUBE.



volume velocity
at any frequency

magnitude (in db)
phase
spectra

area function

AREA CM**2
(square cm)

Fig. B.1 Typical display of acoustic analysis algorithm.

# BIBLIOGRAPHY

Cooper, F.S., Liberman, A.M., Lisker, L. and Gaitenby, J. (1962) Speech synthesis by rules. Proc. Speech Communication Seminar, Stockholm.

Fant, G. (1960) Acoustic Theory of Speech Production, Mouton: 's-Gravenhage.

Halle, M. (1954) The strategy of phonemics. Word, 10, 197-209.

Halle, M. (1964) On the bases of phonology. In J.A. Fodor and J.J. Katz (eds.), The Structure of Language, Prentice-Hall: Englewood Cliffs, New Jersey, Pp. 324-333.

Heffner, R-M. S. (1950) General Phonetics, Univ. of Wisconsin.

Heinz, J. (1962) Reduction of speech spectra to descriptions in terms of vocal-tract area functions. ScD. Thesis, Mass. Inst. of Tech.

Heinz, J.M. and Stevens, K.N. (1965) On the relation between lateral cineradiographs, area functions, and acoustic spectra of speech. Proc. Fifth International Cong. on Acoustics, Liege.

Kelly, J.L. and Lochbaum, C. (1962) Speech synthesis. Proc. Speech Communications Seminar, Stockholm.

Kozhevnikov, V.A. and Chistovich, L.A. (eds.) (1965) Speech: Articulation and Perception, Moscow-Leningrad, (Translation - US Dept. of Com., Joint Publications Research Service: 30,543).

Lindblom, B. (1963) Spectrographic study of vowel reduction. J. Acoust. Soc. Am., 35, 1773-1781.

MacNeilage, P.F. and Sholes, G.N. (1964) An electromyographic study of the tongue during vowel production. J. Speech and Hearing Res., 7, 209-232.

Markowitz, H.M., Hausner, B. and Karr, H.W. (1963) SIMSCRIPT: A Simulation Programming Language, Rand Corp. Prentice-Hall.

Miller, G.A., Galanter, E. and Pribram, K.H. (1960) _Plans and the Structure of Behavior_, Holt: New York.

Morse, P.M. (1948) _Vibration and Sound_, McGraw-Hill: New York.

Ohman, S. (1964) Numerical model for coarticulation, using a computer simulated vocal tract (abstract). _J. Acoust. Soc. Am._, _36_, 1038.

Ohman, S. (1966) Coarticulation in VCV utterances. _J. Acoust. Soc. Am._, _39_, 151-168.

Perkell, J.S. (1965) Cineradiographic studies of speech: Implications of a detailed analysis of certain articulatory movements. _Proc. Fifth International Cong. on Acoustics_, Liege.

Peterson, G.E. and Shoup, J.E. (1966) A physiological theory of phonetics. _J. Speech and Hearing Res._, _9_, 5-67.

Rosen, G. (1958) A dynamic analog speech synthesizer. _J. Acoust. Soc. Am._, _34_, 201-209.

Ross, D.T. and Rodriguez, J. (1963) Theoretical foundations for a computer-aided design system. _Proc. Spring Joint Computer Conference_, Spartan Books.

Ross, D.T. (1964) AED-0 Programming Manual, MIT Electronic Systems Lab, Computer Aided Design Project, Cambridge, Mass. (plus subsequent documentation)

Ruch, T.C. (1951) Motor Systems. In S.S. Stevens, ed., _Handbook of Experimental Psychology_, Wiley: New York.

Stevens, K.N. and House, A. (1955) Development of a quantitative description of vowel articulation. _J. Acoust. Soc. Am._, _27_, 484-493.

# Biographical Note

William L. Henke was born on May 2, 1938 in Janesville, Wisconsin, and graduated from Janesville High School in 1956.

He received a B.S. degree in electrical engineering from the University of Wisconsin in 1961, and a S.M. degree from the Massachusetts Institute of Technology in 1962, also in electrical engineering. During his graduate studies at MIT he was a National Science Foundation Fellow (1961-1963), a Teaching Assistant (1963-1965), and an Instructor (1965-1966) in the Department of Electrical Engineering. He received a junior staff teaching award in 1965.

He has worked for the Autonetics Division of North American Avai'tion, Bell Telephone Laboratories, and others in the electronics and food processing industries.

Henke

```
; TBFDA.HLP - Help file for Tube Frequency Domain Analyzer TBFDA

        KEYBOARD COMMANDS
X ; Calculate a new transfer function using the current tract
    configuration, and display results.
    Also shows current settings of most parameters on the terminal.
HELP  ; Offer help by typing this file on the user's terminal.

     Tract Modification Commands
Two different types of radiation load at the mouth can be specified.
Typing one of the following commands will put that load in effect for
all future calculations, until changed by typing the other command.
        LOADS   ; Use short circuit load.  Loadtype = 1 on output
        LOADP   ; Use piston load [default].  Loadtype = 2
NSEG n   ; Set number of segments.  Default is 34.  Maximum is 50
PSOURCE i ; Causes the "excitation" to be a series pressure source
   inserted to the right of the i'th section.  If i<=0 (default state)
   the excitation is a volume velocity source at the golttis.

   For the following indexed commands, a positive starting index (i)
   causes the following values to be taken as areas for consecutive
   segments i,i+1,...
   A negative starting index causes the values to be taken
   in pairs to set ranges, i.e., A -i n1 area1 n2 area2 ...
   means that startinf with segment i, n1 segments will have area1,
   the next n2 segments will have area2, etc.  Up to 50 area may be
   specified; all values specified will be changed, but only as many
   segments as are called for by NSEG will be used.
   If more than 50 segments are specified, the SLEN array will be
   overwritten with areas, giving strange results.  If this happens,
   respecify SLEN and then start over specifying areas.
A i A[i] A[i+1] ... ; Area input, sequence terminated by EOL. Default is
   all areas = 1.0
SLEN i L[i] L[i+1] ... ; Segment length, in cm. Default is all
   lengths = 0.5 cm.
ATF i AT[i] AT[i+1] ... ; Attenuation factor, in per cm.  Default is all
   factors = .006 dB/cm

     Calculation Display Commands
PRINT   ; Print out current freqs and area (i.e., append to file TBFDA.DAT).
TYPEX   ; Type out on terminal current transfer function
The following commands are not currently working.  If issued, they
will cause TBFDA to crash:
 PLOTX n ; NZ n -> Graphical plot of each new transfer function.
 PLOTVV Freq  ; Plot the magnitude of the volume velocity at given Freq.
 PRINTF n  ; NZ n -> Printer listing of each formant calculation, + areas.

    File Handling Commands
WRAF  ; Write an area file.  TBFDA will solicit name of file
RDAF  ; Read an area file.  TBFDA will solicit name of file
@FileSpec  ; Read commands from "indirect command" file FileSpec

NOTES:  To enter program, type:  run tbfda    at system level.
        To leave program, type ^C.
        Use all capital letters in commands to TBFDA.
        X will compute a transfer function and find formants from the
current area file, using current values of the various options.  If
you type X immediately after entering TBFDA you will get the transfer
function of the default area function, which is a uniform tube with
parameters given as defaults above.  PLOTVV will similarly operate
```

on the current tract configuration, if the plotting option is working. These options can all be changed according to the above commands.

If you want plots (and if the plotting option is working), issuing the appropriate commands to PLOTX or PRINTF before issuing the X command will set the proper flags so that a call to X will plot or print as well as doing X's usual tasks.

TYPEX and PRINT will put results from the most recent transfer function calculation (i.e. the X command) on the terminal/file TBFDA.DAT respectively.

A typical sequence of commands would thus be:

1) Set up tract 1 using any or all of NSEG, SLEN, A, ATF, PSOURCE, and LOADS or LOADP.
2) Set up plotting commands PLOTX and/or PRINTF, if the plotting option is working.
3) Calculate transfer function of tract1 using X and/or PLOTVV.
4) Get results with TYPEX or PRINT. Save areas with WRAF.
5) Modify to tract 2. Any values not changed will remain as set for tract 1.

The algorithm for TBFDA is described in Appendix B of Bill Henke's Sc.D. thesis, M.I.T., 1966, available in the Speech Group Library.

To see this file again, and especially the beginning, type:

```
        ty tbfda.hlp
```

if you are at system level (prompted with @), or:

```
        HELP
```

if you are in TBFDA (prompted with TBFDA>>), and at either level type ^S (hold down control key and type S) to stop the file from scrolling. ^Q will restart the scrolling.

```
TBFDA

 A 34 0.1

 X

 PRINT

 A 34 1.0

 ...

 END

SPOOL TBFDA.DAT
```