A practical study of the application of Computer Techniques in processes of musical composition

by

Kevin John Jones

A thesis submitted for the degree of Master of Philosophy
in the Computer Centre of the
University of Aston in Birmingham

May 1978

SYNOPSIS

A practical study of the application of Computer Techniques

in processes of musical composition

Master of Philosophy

Kevin John Jones

May 1978

Computers have invaded every sphere of man's activities, and music is no exception. Their assistance has proved valuable in analysis and archiving, sound synthesis and in musical composition. After initial doubts, many composers and commentators now recognise how important, and sometimes even indispensable, the computer can be in certain types of composition.

A number of compositional projects in which the computer plays the most significant rôle are described. These are based largely on stochastic, probabilistic systems, and seem to lead to generally satisfying results. Such techniques as those developed in this study could usefully be incorporated into a general composing system.

Computer

Music

Composition

ACKNOWLEDGEMENTS

My thanks go to my supervisor Philip Blundell for his tolerant interest and support. I also acknowledge my gratitude to Dr. Graeme Bailey for making helpful comments on some of the mathematical content and generously giving of his experience and expertise in the typing and presentation of the thesis.

CONTENTS

PART ONE

THE COMPUTER, ART AND STOCHASTIC MUSIC

In the twentieth century, a great variety of compositional techniques have been developed in music; as composers have experimented, differing styles have been adopted and subsequently dropped or adapted in search of more powerful means of expression. It now seems that musical composition is entering a period of consolidation as composers are reviving techniques which had previously been rejected in a period of experimental fervour.

One method which still remains to be developed is that of statistical manipulation of elements, which is potentially a powerful addition to the available repertoire of techniques, but to be used effectively it is really necessary to use a computer.

Xenakis is the only composer to have made much use of these techniques, even though others such as Penderecki have copied the superficial sound image less convincingly. (Penderecki's music may initially have instant appeal, being used in a programmatic context, but Xenakis' has a greater pregnance, being open to a wider range of interpretation, with a constantly varying sound image which grows richer on repeated hearings.) Xenakis found that his use of statistical techniques in plotting large scale structures led naturally to enlisting the assistance of the computer in making calculations.

Some other attempts to use the computer in musical composition seem to have been made merely for the sake of using a computer and have been failures or of little more than mere novelty interest.

Before going on to consider statistical techniques and other uses of the computer in music its rôle in art generally will first be considered

1. Computers and Creativity

LUCRETIUS : "Nil posse creari de Nilo"
("Nothing can be created out of nothing")

(De Rerum Natura I.155)

The idea of a computer being used in art or to make music causes most people to react with stunned amazement and often anger. Lejaren Hiller tells of the occasion when an English music professor chanced to meet him coming out of a supermarket in New York, and viciously rated him for his work in computer music saying that he deserved to be shot! Comments such as " Where is the creativity in that? " are typical; whilst many entertain sinister visions of a world taken over by vast armies of clicking machines attended by an army of busy white-coated operators. Even though such a picture is obviously wildly inaccurate it is nevertheless puzzling why most people should feel threatened by computer art, and be doubtful about its validity.

It is essentially a useless exercise to attempt to vindicate computer creativity, for any answer lies buried in the terms used to pose the question: " If that's what you mean by creativity that's not what I meant. " Any output from a computer could be considered creative - even constantly repeated material - for that might be thought of as original by some. On the other hand, the result of following a determinable, albeit unknown path through a computer program could be thought of as not creative at all. If creativity is considered to be the ability to bring something new into existence, the question remains: " What is new? " which is the cue for an endless bout of word wrestling. This is precisely what is happening when Boulez writes:

> "Ordering the course of a certain group of events - methodically, empirically or by the intervention of chance - is not at all the same as giving them the coherence of a form." [1]

This is a classic non-statement, a mere argument about labels.

The Czech scientist, Nemes, under the influence of official communist party thinking, argues that every experience must be part of a closed system, and can ultimately be revealed in a logically plottable form given sufficient time and research:

> "Any inspiration, any sudden insight, considered 'supernatural', coming from nowhere, 'like a shooting star', can be reduced to deterministic thinking processes." [2]

However, quite apart from any need for metaphysical explanations, contemporary thinkers have moved beyond such an idealistic view to recognise that any understanding of our experience is in a constant open-ended state of progression in which new and sometimes improved models of thought are continually suggested. Karl Popper writes:

> "In science, we never have sufficient reason for the belief that we have attained the truth. .... In so far as scientific statements refer to the world of experience, they must be refutable; and in so far as they are irrefutable, they do not refer to the world of experience." [3]

Even the assumed values of conventional logic are based on non-provable axioms.

---

[1] Boulez p. 30
[2] T. N. Nemes p. 209
[3] Karl Popper p. 13

It is foolish to make great claims about computer creativity and computer art; it is equally foolish to ignore the exciting opportunities offered by the computer in extending the composer/programmer's ideas in methods which can be enjoyed for their own sake. The computer itself can not really claim to be creative, the programmer surely can, but exactly where the label is fixed is irrelevent.

Numbers have always been a significant aid to composers throughout history. It is not difficult to find examples of composing devices or systems which have been used in the past: Samuel Pepys (1639 - 1703) possessed a musarithmica mirifica consisting of number and sign tables. The Prague Cistercian monk, Mauritius Vogt, in "Conclave Thesauri magnae artis musicae" (1719) described a system of composing using bent hobnails to represent melodic turns. William Hayes, an Englishman, in "The Art of Composing Musick by a method entirely New, Suited to the Meanest Capacity" (1751) proposed a composition method using ink blots and playing cards. Mozart's 'Dice Game' compositions, K 294 D (1795), offer various alternative bars from which selction can be made by throwing dice and consulting a chart. This is probably the best known amongst the various approaches to 'automatic' composition. Many other similar treatises have appeared: ars inveniendi, artificia heuristica, ars combinatori, etc. [1]

But quite apart from these totally determined systems of, in general, a few eccentric individuals, of interest only in their oddity and of no real artistic merit or significant practical effect, musical history is

---

[1] Nemes p. 211

full of examples of successful mainstream composers making use of mathematical techniques in plotting structures in their compositions. This is evident in the carefully balanced symmetries of the Medieval Chanson and Isorhythmic motets, through the precise counterpoint of Bach and even in the music of such an arch-Romantic as Wagner, apparently as far removed from 'Formalism' as one could get. He wrote:

> " The work of art produced non-consciously belongs to ages far removed from our own. "

Evidence suggests that Wagner made significant use of formal arithmetical calculations in structuring and balancing sections of his work. [1]

All composers have always employed commonly used forms and systems to aid their composition. The rules themselves do not produce great works of art, but the use which is made of them (and the ways in which they are broken!). By making use of a computer today, the composer is only following a time-honoured tradition, but with more powerful resources at his disposal.

In recent years, there have been many changes in composing methods. In particular, many composers have introduced chance elements into their music. Composers like Berio, Stockhausen and Boulez have produced compositions where the order in which individual sections of the piece may be performed is variable, or where 'mobile' elements of the composition may be freely combined, or where only vague instructions are given to the performer who has a large amount of freedom in interpreting the score. John Cage has made greatest use of chance procedures within the

---

[1] Stuckenschmidt p. 193

compositional process itself. In describing his methods he concludes:

> " It is thus possible to make a musical composition the continuity of which is free of individual taste and memory (psychology) and also of the literature and "traditions" of the art. The sounds enter the time-space centered within themselves, unimpeded by service to any abstraction, their 360 degrees of circumference free for an infinite play of interpretation. Value judgments are not in the nature of this work as regards either composition, performance of listening. The idea of a relation (the idea : 2) being absent, anything (the idea : 1) may happen. A "mistake" is beside the point, for once anything happens it automatically is. " [1]

In listening to pieces constituted within such an ideology, the audience has to assume a much more significant rôle in interpreting or re-creating what they hear.

The American composer Steve Reich has extended the process approach to control easily perceptible changes in the actual sounds of the music itself. The way in which he describes his method is very similar to a composer's approach to the computer:

> " Though I may have the pleasure of discovering musical processes and composing the material to run through them, once the process is set up and loaded it runs by itself. " [2]

One of the aims of computer music composition is to devise systems to generate the maximum amount of pattern with the minimum of instructions.

Exactly what 'pattern' means is again disputable. Even absence of pattern, a totally random distribution of objects, is itself a type of pattern. But once a totally random distribution of objects has been produced as a work of art, then any other random distribution is essentially the same;

---

[1] John Cage p. 59
[2] Steve Reich

the point has been made, and it can not be repeated. It is the task of the artist to produce new and interesting arrangments of objects, at appropriate points between the extremes of total order and total disorder; pattern structures which can be appreciated by our natural perceptive abilities.

The examples below illustrate how minimal constraints can impose order upon apparently randomly arranged objects, and so create an easily observable form:

A. This sequence of numbers appears to have no meaning:

7 3 4 7 0 9 1 1 3 6 5 8 6 2 4 6 5 4 1 8 0 5 3 7 2 9 2

But if the constraint of selecting three adjacent numbers is imposed they assume a creative potential:

6 5 8

B. Similarly, it is difficult to find a pattern in a random distribution of circles:

```
O   O O     O O O   O
    O     O O       O
O O   O O O   O   O
  O O   O   O O O O
O   O     O     O   O
O     O   O O O     O
  O       O   O O O O
  O     O   O     O
```

But if just one corner is considered, it has appreciable form:

```
O   O
    O
O O
```

C. The following words, selected at random, make little sense:

catch the with number to it he

but if a simple grammar is used to generate a sentence something of the following nature appears:

the moon quietly slips over the whispers

Sentences of this sort were produced in the text pattern experiments which are described below. [1]

In producing such patterns, the computer itself is not aware of the feelings it might engender; the reader responds to the visual or verbal stimulation. Furthermore, the computer is not making its own choices to order such objects but is only operating under the instructions of the programmer who makes a creative decision in deciding what the constraints are to be.

Pierce gives examples of 'Stochastic English' which include the sentence:

It happened one frosty look of trees waving against the wall.

He considers that the interest and amusement provoked by such material is sufficient justification for calling it a contribution of mathematics to the arts. [2]

---

[1] see page 62
[2] Pierce p. 52

Sometimes the computer may come up with something entirely unexpected: the case of a theorem-proving program demonstrating that base angles of an isosceles triangle are equal by showing ΔABC is congruent to ΔACB, apparently unthought of before, is well known. It was found in working on the program PIANOCOMP that one piece resulted based largely on trills, frequently doubled at multiples of an octave, which are to be played with varying and contrasting intensities, which can produce an interesting effect. This was a contingency only subconsciously allowed for in writing the program.

It is tempting to say that if creativity can be taught, then a computer can be creative. But in teaching creativity, one is merely bringing out what is there already. Using a computer helps to bring out possibiliites which the composer/programer allows for, but of which he may not be consciously aware.

And so, aspects of creativity in computer compositions can be seen to rest in the pregnant logic of the program, the way in which the composer makes use of it, and in the reaction of the listener to the finished work.

2. Computers in Music

Before dealing specifically with musical composition, it is worthwhile to consider briefly some of the other ways in which the computer has been used in musical applications, all of which have significant implication in compositional techniques.

The computer has been proved useful in the field of musical analysis. As in most disciplines, it has greatly facilitated information handling where large amounts of data are involved. In ethnomusicological research, the computer has been used to sort, order and compare the results of field work; similarly it has assisted in the archiving and verifying of historical records. Examples of such applications are given by Harry Lincoln in his general survey[1], but it is not relevant to this enquiry to pursue this subject in any greater depth.

More significant in its implication on compositional techniques is the application of statistical analysis in music, to analyse the distribution of particular notes, note combinations or other musical elements.

The writer has carried out a small experiment using the computer to count the frequency distribution of intervals in certain recitative passages by Bach and Handel, which gave some useful results[2]: it appeared from the passages used, that Handel tended to use smaller intervals most of the time, reserving larger ones for occasional effect, whereas Bach made use

---

[1] Lincoln
[2] Kevin Jones

of a more even spread of intervals - in listener's terms this is likely to mean that Handel's music is more predictable and easier to listen to, whereas Bach's music has greater variety (in note by note terms) and needs more aural effort to be appreciated.

This in no way passes any value-judgement on the composers but it is a point with which most listeners would probably agree. The analysis of Bach, from his St. John Passion, also revealed the fact that Christ's part contains a comparatively large number of perfect fifths. This is something which ought to have been expected from Bach, but which might not otherwise have been immediately obvious but for the computer analysis.

One of the earliest experiments in computer music was carried out using a computer to analyse Stephen Foster songs, based on the occurence of two-note and three-note patterns. [1] The values derived were then used to generate new songs, with notoriously poor results. The experiment none the less, was useful in proving the inadequacy of simple 'counts' to define a style or any of the real form of a composition and anyone making use of such methods should be well aware of their limitations.

Little work has been done on direct computer analysis of musical form and syntax. The nature of any general musical syntax itself is in dispute even before it could be usefully applied in a general analysis. The many parameters of musical interpretation make musical syntax at least as complicated as that of natural language and it is likely that developments in this field will follow developments in computer analysis and interpretation of natural language.

[1] Olson and Belar

Some limited basic work has been pioneered. In Edinburgh, M. J. Steadman and H. C. Longuet-Higgins have analysed certain aspects of the fugue subjects in Bach's 48. [1] The program attempts to establish the tonality and metre of the naked melody input devoid of any tonal or metrical context. This is achieved by formulating a special set of rules against which successive checking takes place.

Frankel, Rosenchein and Smoliar have made use of the computer language LISP to describe the syntax of part of Beethoven's Ninth Symphony which may be useful in further analytical experiments. [2,3]

The most important application of the computer to music is in sound synthesis. Work done in this field has made an impact way beyond the boundaries of music alone, for example in digital speech transmission and storage. Computer music composition is likely to be most effective and significant in the context of digital sound synthesis.

There are basically two types of sound synthesis systems developed. Computer controlled analogue studios, and direct digital sound synthesis. The former maintain computer control over synthesizers. This gives the user the advantage of much greater speed of operation, but the basic sounds which can be produced are sometimes inaccurate and limited, though some composers prefer to work in such an environment. A number of these systems exist, particularly in many United States' Universities, but each system is generally unique to its installation. EMS in London market a

[1] Longuet-Higgins and Steadman
[2] Frankel, Rosenchein and Smoliar
[3] Smoliar

small computer controlled studio, and there are also studios in Stockholm and Utrecht.

In direct digital synthesis, the computer uses a program to manipulate sound data and prepare a sequence of numbers which define a sound pressure wave, which is then fed through a digital to analogue converter, at the rate of some 20 000 samples per second, so that the resulting signal can be used to drive a loudspeaker.

Various programs have been developed to effect the translation of instructions into digital sound samples, again, mostly in the United States. These mainly belong to a family of variations on a basic program developed by Max Mathews and others: *MUSIC V* [1], *MUSIC IV BF* [2], *MUSIC 360*, *MUSIC XI*, and *SOUND* [3]. Other programs such as John Clough's *TEMPO* [4] have been developed. Some of these programs are in machine code and can only be run on a particular type of computer, others are in FORTRAN and can be implemented more widely. This is now being done at a few centres in Britain and the rest of Europe.

The possibilities engendered by direct digital synthesis are theoretically unlimited, but in practice are restricted by the poor imagination of the user, inadequate acoustical knowledge and lack of programming skills. But attempts are being made to remedy the situation.

Hybrid studios are also being developed to highlight the advantages of both systems, which incorporate both digital and analogue sound sources

---

[1] Mathews
[2] Howe p. 175 ff
[3] Byrd
[4] Clough

under overall digital control.

Digital sound synthesis systems have proved valuable in sound research. Conventional understanding of acoustics of musical instruments has been shown to be inadequate as a result of attempts to synthesise these sounds digitally.

It is to be hoped that work done on composing units incorporated into sound synthesis systems will be useful in making the system more accessible to composers. Compositional algorithms of the type described in this study could be incorporated into a computer music system and thereby reduce the phenomenal number of instructions needed from the composer at present; and at the same time ensure maximum variety in sound output.

3. The Computer in Composition

The highly abstract nature of musical language makes it a more suitable candidate for synthetic computer composition that the other arts. However, many attempts to use the computer along lines suggested by 'classical' approaches to composition have met with little success. It is only in using the computer in new methods, in compositional techniques otherwise impossible to attempt without the computer's assistance, that more success has been acheived. Before considering these new compositional styles, some of these earlier uses of the computer will be described. They fall into a number of main categories:

a) programming rules of harmony and couterpoint

The earliest, well-publicised experiments in computer composition were conducted by Hiller and Isaacson at the University of Illinois. [1] They programmed a computer with elementary rules of counterpoint, based on the work of the seventeenth century theorist, Fux, and generated sequences of random numbers which could be tested against the rules and accepted or rejected. A number of experiments were conducted in which various rules were removed until the constraints of final experiments were minimal. The music produced in this way showed a gradual progression of styles, finishing with scores looking very much like the music of Bartok. A suite of pieces for string quartet was assembled and named the *Illiac Suite*, after the computer used for the experiment.

[1] Hiller and Isaacson

In very small sections, music prepared in this way can sound reasonably convincing, but on a broader level, the music has no appreciable form and merely meanders on in a meaningless string.

Working quite independently, the Russian engineer Zaripov programmed a computer with Basic rules, based on his observations of the form of simple folk songs. [1] These took into account overall form patterns. Some interesting melodies were generated which he called *Ural Airs*. Zaripov achieved a certain fame in the USSR for his work.

Champernowne has synthesized Victorian hymn tunes with apparently reasonable success; [2] and more recently, Robert McMahan has reconstructed examples of late Brahms piano music. [3]

b) <u>statistical analysis and synthesis</u>

Some early attempts at re-synthesizing melodies by analysing the transition probabilities inherent in given note sequences were made in the United States with varying types of material. Klein and Bolitho analysed popular songs in their "Push Button Bertha" experiments (1956); [4] Brooks, Hopkins, Neumann and Wright worked similarly with Hymn tunes, [5] Olson and Belar with Stephen Foster melodies. [6]

---

[1] Zaripov
[2] Hiller p. 82 'Music Composed with Computers'
[3] Vinton; article on 'Computer Applications'
[4] Hiller p. 45 'Music Composed with Computers'
[5] ibid p. 46
[6] Olson and Belar

Such efforts as these can not produce compositions valid in their own right but are useful ways of establishing the sufficiency of methods of form analysis. A parallel can be drawn with an equivalent situation in sound synthesis where to synthesize natural instrumental sounds merely for the sake of the superficial sound, is pointless, since the original instrument would do the job far better, but in attempting to synthesize natural sounds accurately, great insight can be gained into their nature and then this knowledge can be used to generate new and more interesting ones. Research already done in this area has demonstrated the inability of conventional acoustics to describe musical sounds adequately. Similarly, the inadequacy of methods of analysis has been demonstrated by attempting to use those methods to reconstruct musical pieces.

In the *Computer Cantata* of 1963, Hiller and Baker attempted to use the techniques of analysis and synthesis to produce a substantial composition. An appropriate text was used, examples of 'Stochastic English', which was accompanied with music derived from successive approximations to Charles Ives orchestral work *Three Places in New England* . The piece begins totally at random and more and more order is gradually introduced.

The *Fantasy for ten winds, percussion and Tape* makes use of the probabilities of note occurances in the hymn tunes *Old Hundredth* and *Now thank we all our God* .

For the large scale composition *HPSCHD* , in which Hiller collaborated with John Cage, tapes were generated based on an analysis of Mozart's music. The actual notes to be used, however, were chosen from scales derived from programming the *I Ching* oracle, used in many of John Cage's compositions, and which turns out to arrange chance elements within a binomial distribution.[1]

c) Mozart's Dice Game

This has been a source of stimulation as a fairly easy combinatorial exercise for computers. D. A. Caplin programmed the game on a Ferranti computer in 1955,[2] and this was also used as the basis of experiments in Glasgow.

The harpsichord parts of Hiller and Cage's *HPSCHD*, mentioned above, were constructed from the game.

d) collage

Hiller's *Avalanche for Pitchman, Prima Donna, Player Piano, Percussionist, and pre-recorded tape* contains a computer plotted piano-roll, formed from a shuffled assortment of ninety nineteenth century symphonic themes which gradually build up and thicken in texture.[3]

Work on this piece exposed a flaw in the computer plotter, which had hitherto remained undetected by computer personnel. Hiller knew what sort of output he expected. The computer staff admitted that they were all too unaware of possible enormous errors which may have occured in the work of

---

[1] Hiller 'Programming the I Ching Oracle'
[2] Hiller p. 47 'Music Composed with Computer'
[3] ibid p. 61

nuclear scientists and biologists who had trustingly accepted the computer output without question! This is yet another example of computers used in the arts being of service to scientists.

e) Twelve-note serial composition

Twelve-note composition is something which seems to have dominated American interests, and most work has probably been done in this area. The basic serial tenet: of always stating a twelve-note series in its entirity before proceeding to the next compositional act, seems to be a convenient starting point upon which further pattern may be superimposed.

In his *CSX-1 Study* of 1963, Baker employed systematic permutation of twelve-note material.[1] Brun, in composing *Soniferous Loops* (1965), added additional probability distributions. In this piece, expression marks were added afterwards.

An article by Kobin and Ashford discusses the problems of computer composition and imposition of extra constraints governing pitch, note-duration, number of instruments playing at one instant and the nature of intervals.[2] Studies of this nature are not uncommon.

In England, Stanley Gill, responsible for the first conference on computer music, produced a string trio based on serial compositional techniques.[3]

---

[1] Hiller p. 52 'Music Composed with Computers'
[2] Kobin and Ashford
[3] Gill

In addition to the basic rule of serial composition: using a twelve-note row in its entirity, a few other simple constraints were introduced. Each voice was limited to a range of two octaves, and should rest for approximately one bar in four or five, but no two voices should rest together. One voice should move quite rapidly whilst the other moves slowly, and parallel octaves were to be avoided. No attempt was made to introduce any overall structuring of the whole piece, part of which was used as background music for a BBC TV programme *Machines like Men* broadcast in 1962.

Koenig apparently uses twelve-note composing programs in his computer music work in Holland. [1]

More recently, Donald Byrd has described a twelve-note based composing program *MUSC* available as part of the computer music facility at Indiana. [2] The user of the program supplies a line segment function, called a " Contour function " which, along with the alternative of its inversion, is used to control the pitch, rhythmic and dynamic structure as a function of time for each voice. The way in which this is done varies with each parameter. Exact pitches are chosen using twelve-note technique, but the contour function is used to determine the register (or octave position) of each note within the instrument's range. In this way, he claims, the melodic line should have coherence at both micro and macro levels. The user needs to supply a number of rhythmic patterns arranged progressively according to average note duration, from which the

[1] Hiller p. 86 'Music Composed with Computers'
[2] Byrd

contour function is used to make choices. A simpler method is used to choose dynamics. In addition to the above constraints the user can specify an amount of randomness to vary the control of each parameter. Each instrument's part is composed independently.

f) stochastic composition

Stochastic composition, and Xenakis' work in particular are considered later, but brief mention is made here of some other compositions.

In *Non-Sequitur VI* (1966), Brün fed probability distributions into the computer. [1] These were then changed according to the actual environment being generated. The simple heuristic implications of this working method are of some interest. Cuomo in his pieces *Zetos 1* through *5* makes use of probabilistic control of density, [2] similar techniques to those of Xenakis. James Tenney has generated pieces constructed around mean values of key parameters which are changed in the course of the composition. [3] His *Four Stochastic Pieces* (1962) and *Ergodos I and II* (1963 and 1964) used these techniques.

In France, Pierre Barbaud has used Stochastic matrices to control chord sequences in producing music with more traditional associations. [4] But again, hybrid approaches of this sort are not calculated to produce particularly inspiring results. In his programs, which are written in

---

[1] Hiller p. 58 'Music Composed with Computers'
[2] ibid p. 64
[3] ibid p. 68
[4] Barbaud

ALGOL 60 , Barbaud adopts the rather charming arrangement of labeling the procedures with girl's names!

g) <u>miscellaneous</u>

To complete the list, certain other approaches can be mentioned. John Myhill in his *Scherzo a Tre Voce* (1965) makes use of different time functions controlling the main parameters, whithin certain constraints. [1] This approach seems to anticipate a similar method incorporated in the *MUSC* program of Donald Byrd described above.

Papworth has used the computer to plot permutations in systems of change ringing. [2] Hiller took up the idea, and exploited the permutation technique in controlling pitch, dynamics and rhythmic variation in the composition *Algorithms II*. [3]

Alan Sutcliffe, in England, has developed a composing language ZASP which permits the user to specify limits within which randomly generated patterns are organised. [4] Lejaren Hiller has also developed a more general composing language MUSICOMP which offers a selection of procedures to help composers. It has been used in some of the compositions already mentioned, in particular in his own *Computer Cantata* . [5]

[1] Hiller p. 57 'Music Composed with Computers'
[2] Papworth
[3] Hiller; lecture at City University, London.
[4] Sutcliffe p. 37
[5] Von Foerster and Beachamp; article by Hiller.

Of the experiments listed above, many are simply games, but of interest and value none the less, others have greater integrity as interesting pieces of music. In general, any effort to re-create the complex hierarchic structures of traditional music using stochastic or any other simple techniques are likely to fail. It is rather silly to dismantle an old building and attempt to reassemble the bits in a different way, but so that the old building is still recognisable; far better to leave the old building standing, and construct something entirely new with fresh materials. It is in breaking new ground, in original approaches to composition, that the computer is most useful.

4. Stochastic Composition

In an early lecture on computer music, Lejaren Hiller affirmed that the most important and significant applications of computer music composition are those in which the computer works out its own compositional structure and establishes new compositional methods.[1] It is in this situation that stochastic processes are useful, when the computer can assume a heuristic rôle.

Many early compositions failed in this respect, as attempts were made to emulate other styles, but without success. Xenakis, however, in using the computer has broken new ground and produced interesting and exciting pieces of music which have achieved success on the concert platform and established a considerable following amongst both performers and audiences. His methods have received frequent airings in the popular press, on radio and television, and concerts of his work are regularly promoted, though not all of his pieces make use of the computer in their design. The English Bach festival has regularly featured Xenakis' music, and a number of recordings of his works are available. Adrian Jack writes:

> "the sound image of Xenakis' music is as strikingly recognisable and therefore as reassuring as the feel of one's slippers."[2]

Praise indeed; though not all critics' comments have been so favourable by any means!

---

[1] Stuckenschmidt p. 191
[2] Jack

The essential nature of stochastic composition is a process of defining a number of sound elements which are then sequentially arranged according to defined probabilities governing their juxtaposition. A stochastic matrix is used to establish that a given type of element will follow another within a regular pattern of occurence. In this way, the linking of each object to its neighbour defines implicit relationships within the entire structure.

Two simple examples will serve to illustrate this:

If four symbols are considered: I , O , X and a space; the following stochastic matrix can be defined:

	I	O	X	space
I	0·0	0·0	0·0	1·0
O	0·0	0·6	0·3	0·1
X	0·0	0·25	0·5	0·25
space	0·2	0·6	0·2	0·0

This matrix suggests that I will always be preceded by, and followed by a space, and that O's and X's will tend to occur in groups of the same symbol. It is also possible to see that O's will occur most often, and I's hardly at all. When called upon to 'perform', the matrix will produce a pattern such as the following:

```
OOO OOOO OXXXO XXOX
O I OO XXXXOO OOOO
I OXXXOOX OXO OO I
XXX XOO OOOOO OOOO
OOO OXXOXOOO O I O
```

The repertoire of symbols can now be extended to include H , and another matrix be constructed as below:

	H	I	O	X	space
H	0·6	0·3	0·0	0·0	0·1
I	0·3	0·7	0·0	0·0	0·0
O	0·0	0·0	0·1	0·9	0·0
X	0·0	0·0	0·8	0·1	0·1
space	0·1	0·0	0·1	0·0	0·8

This matrix essentially produces three main types of pattern: sequences of H's and I's - usually in blocks, sequences of O's and X's - usually alternating, and a large number of spaces, with the chances of passing from one type to another being slim. A pattern such as the one below is likely to result:

```
I I I I H H H I I I H           H
H H I I I I I H H
  O X O X O X O X             H I
H   O X O X O X X O X O X O O X
              H H H         H H H
I I I I I H H I H H H H
  O X     O X O X O O X O X O X X
```

Xenakis has tended to use stochastic matrices to order what he calls 'screens' of sounds.

$$P_0 = 0{\cdot}5488$$
$$P_1 = 0{\cdot}3293$$
$$P_2 = 0{\cdot}0988$$
$$P_3 = 0{\cdot}0198$$
$$P_4 = 0{\cdot}0030$$
$$P_5 = 0{\cdot}0004$$

where P_i is the probability of an i-fold event occuring in any cell. The table is then used to work out how events will be distributed in the 196-cell matrix defining the composition's structure (28 time divisions × 7 instrumental classes).

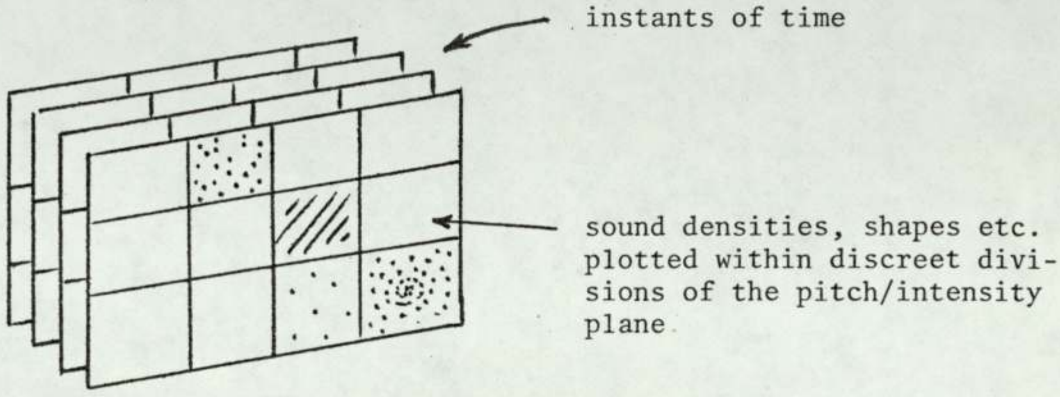
This gives another table showing the number of cells for each i-fold event:

i	number of cells = $196 \times P_i$
0	107
1	65
2	19
3	4
4	1

The events are then distributed over the two-dimensional plane, with efforts being made to keep the distribution uniform within each row and column as well as over the plane as a whole. The result is shown in the diagram overleaf (fig 1). This is only the first stage of an unfolding compositional process exploiting the Poisson distribution.

All of the computer assisted works written by Xenakis seem to have been generated from one basic program, completed in 1962 and run on an IBM 7090 machine in Paris.

Each screen is defined itself according to a probabilistic framework of different sound densities and sound types, with points of sound plotted on a pitch/intensity cartesian plane. These are then assembled in time like the pages of a book, with the progression of sonic variation controlled stochastically by matrices of transition probabilities.


instants of time
sound densities, shapes etc. plotted within discreet divisions of the pitch/intensity plane

Xenakis frequently makes use of standard probability distributions in organising his material.

For example, in an early work, *Achorripsis* for orchestra, the Poisson distribution is used to distribute regions of varying sound density on the instrumental/time-class plane. Xenakis does this by arbitrarily adopting a mean density of:

$$\lambda = 0{\cdot}6 \text{ events/cell}$$

and using the Poisson formula:

$$P_k = \frac{\lambda^k}{k!} e^{-\lambda}$$

to work out the table of probabilities:

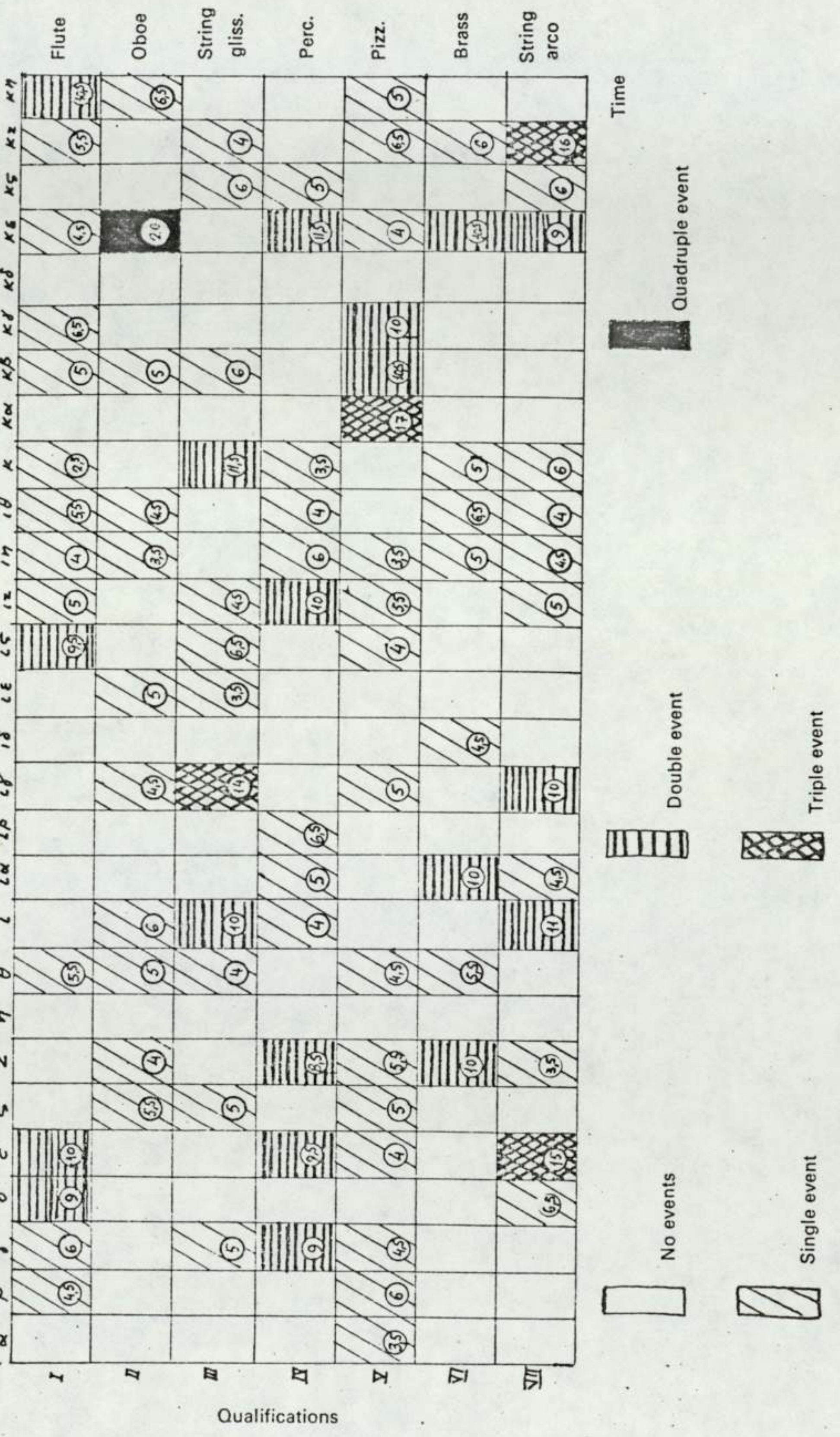


Fig. I Xenakis: Matrix of *Achorripsis*

These compositions are based on a structure split into small time blocks of varying size, between one and ten or more bars. In each block the sound shape is defined in terms of density, degree of order (which he calls 'ataxy'), rapidity of change, and similar concepts which are quite different to those normally associated with traditional music. In this way, his approach has similarities with contemporary theories about the physics of small particles whose behaviour is described using stochastic laws which define probable states of systems and general distributions of phenomena rather than absolute deterministic patterns.

Xenakis feeds in data concerning the density patterns required and then for the sound in each sequence the computer calculates its time of occurence, class of timbre, instrument, gradient of glissandi (if present), duration and dynamic. The method permits licences taken afterwards by the composer on the machine output.

The first piece produced in this way was *ST/48-1,240162*. The title means: Stochastic music for 48 instruments, first piece, run on the 24th January 1962. Subsequently *ST/10-1,080262* was produced and *ST/4-1,080262* which is simply a string quartet arrangment of *ST/10* - effected by taking the string parts, and freely incoporating the more important of the remaining instrumental parts where the string parts would otherwise be silent. Other compositions based on the same program are *Atrees* for ten instruments: flute, clarinet, bass clarinet, horn, trumpet, tenor trombone, violin, cello and percussion (maraccas, suspended cymbal, gong, five temple blocks, four tom-toms and vibraphone); *morsima-amorsima* for piano, violin, cello and double bass; and *amorsima-morsima* for ten instruments.

Xenakis also used a computer to write the opening piano solo of *Eonta* for piano and five brass instruments (1964), and apparently in the orchestral piece *Strategie* .

It is quite simple, having written one composing program, to generate a whole family of compositions from it. To an unfamiliar ear the compositions might appear to sound the same, for they are all couched in a strange, but nevertheless consistent language. The average Englishman would no doubt be unable to distinguish between a tax demand or a love letter in Chinese. And similarly, a visiting African monarch on hearing an orchestral concert of European music ranging from Bach to Bartok is reported to have asked why the orchestra kept playing the same piece!

Because Xenakis' music is concerned with such parameters as density, pitch relationships may be insignificant and the overall dynamic level is often very high; for this reason many people often find it oppresive.

The writer's own work has make use of 'computer-defined' sound structures and used quasi-random techniques to generate stochastic matrices and thus produce new and unexpected form patterns. Xenakis, in feeding in his own data concerning density structure, and in using standard probability distributions claims to emulate the natural processes of nature, and this does indeed often seem to be evident in the sound of his music.

Xenakis' recent work has been to extend his approach to macro-structures to the area of micro-sound and the definition of sound-pressure waves in probabilistic terms to arrive at authentic 'natural sounding' timbres; but this area is only just beginning to be explored as appropriate hardware and programming resources are developed.

The use of stochastically-controlled, continually varying sound patterns, with no literal repetition of sound material, is consistent with the general development of musical technique.

In early music, literal repetition was the common major source of form-building. As music progressed, patterns for varying repetition were developed, sound material re-appeared in new contexts according to standard polyphonic schemes, or later classical forms. Subsequently techniques of repetition became completely fragmented in the work of Romantic and Impressionist composers until Schoenberg introduced the use of constant variation and anti-repetition of dodecaphonic serial music. This process naturally leads on to music based on Markov chains where constant variation appears on a 'theme' implicit in a stochastic matrix of transition probabilities.

Xenakis has strongly criticised serial music. He pointed out that in complex polyphonic serial music, the very complexity destroys its form. The individual lines are no longer recognisable or appreciable so that the whole sound complex becomes a vague formless mass; meaningless and irrational. The ear percieves the sound as a whole and is only aware of the textural changes. This, Xenakis claims, justifies his statistical approach to composition.

What Xenakis has to say is quite true about large scale serial structures, but does not hold for the delicate, open and easily recognisable structures of many more common serial compositions, for example those of Webern, which clearly maintain a definite identity in which the function of the row is preserved. Nevertheless, new forms of composition must be developed and stochastic techniques have much to offer.

Such a compositional tool is potentially far more powerful than serial composition, as the composer freely chooses and varies his own parameters; but it needs skilful use, and such unwieldy resources can only really be managed with the assistance of a computer.

A useful comparison is with systems where one's own scale system can be defined. As in Indian music, where the player improvises on a set Rag with its implicit melodic fragments and resultant probability structure, so the computer 'improvises' on the pattern implicit in the stochastic matrices.

For many years, Western culture has been dominated by sequential ideas of time. Ideas which have not only dominated musical structures, but also politics, social activities and science. Today, a move is being made away from the dominating influence exerted by a concept of sequential structured time, perhaps partly as a result of Einstein's Theory of Relativity and its influence on our understanding of separate events and irreversability of time, and this move, back to concepts which have always remained dominant in Eastern cultures, is reflected in our own music.

Yehudi Menhuin writes of Indian culture:

> "Life and death are not all and nothing, but stages in a process, episodes on an infinite river to which one trusts oneself and all other phenomena. So it is that Indian music reflects Indian life, having no predetermined beginning or end but flowing without interuption through the fingers of the composer-performer: the tuning of the instrument merges imperceptibly with the elaboration of the melody, which may spin itsitself out for two, three or more unbroken hours

itself out for two, three or more unbroken hours." [1]

and then goes on to affirm:

> "Melodically and rhythmically Indian music long ago achieved a complex sophistication which only in the twentieth century, ... has Western music begun to adumbrate." [2]

In sound sculpture, the sequential ordering of 'melodies' becomes unimportant. The overall effect is what matters, and the way in which individual elements contribute to the whole. Significance is attached to density, intensities, rates of change, time-independent pitch states, spatial position and other parameters which have not been as significant in music historically.

Pulse and rhythm are not being made to disappear, indeed, their nature and function have become enhanced as one is made more conscious of their intrinsic presence by attempts at masking or removal. The work of such figures as Murray Schaeffer have made people more aware of the natural sounds of the environment which have musically unfamiliar rhythmic qualities. [3]

A listener approaching stochastic music should not listen for what could be conveniently be termed 'coherent melodies' but should allow his ears to loose themselves in a sea of sound, to enjoy the general atmosphere, out of which balancing patterns and forms will emerge.

---

[1] Menhuin p. 257
[2] ibid
[3] Murray Schaeffer

Initially, to the unaccustomed ear, it is difficult to appreciate the variety and structure of these unfamiliar sounds as music. Being used to perceiving and enjoying such classical functions as inversion, retrogression and transposition - albeit unconsciously - with little more difficulty than it appreciates simple repetition, the ear finds that the constant linear variation of stochastically generated material has no obvious anchoring points.

In the following projects, an attempt has been made to organise stochastic material within a framework which will offer additional support to the development of a coherent form and a more immediately obvious and identifiable sound image.

PART TWO

PROJECTS

"Now let me roll beneath the hooves of chance"

Norman Cameron

5. Common basic techniques in method

All the programs were written in ALGOL 60.

Similar basic procedures and programming techniques have been used in most of the pieces, and these will be considered first of all.

A simple pseudo-random number generator was exploited of the form:

$$X_i \leftarrow \begin{matrix}\text{decimal}\\ \text{part of}\end{matrix} \left(100 \times (aX_{i-1} + b) \right)$$

where X_i is a real number in the range $(0, 1)$.

The value of b is used as a parameter of the procedure call, so this changes according to the path followed through the program and fractures the sequence.

The generator was defined as a functional procedure which can be substituted in the program wherever a random number is required:

```
real procedure rndec(b);
   comment generates a random real number in the range (0 , 1) ;
   real b;
   begin
      x  :=  9.2351 * x + b;
      rndec  :=  x  :=  100 * x - entier(100 * x);
   end;
```

x is given an initial value to start the sequence. This can be a useful way of storing a compositional sequence, since if the same initial value of x is used, the program will always follow an identical path.

A further functional procedure, rnd(n), was defined to generate a pseudo-random integer in the range [1 , n] :

```
integer procedure rnd(n);
   comment generates a random integer in the range [1 , n] ;
   integer n;
   begin
      rnd  :=  entier ( rndec(3.7511) * n)  + 1;
   end;
```

In this particular application, absolute randomness is not important. Since the computer generates its own probabilities in the stochastic matrices, any bias in the number sequence will merely have the effect of varying the value of these probabilities.

It is obvious on examining the output of the programs that certain types of pattern do seem to emerge which are not anticipated in programming and are unlikely to be the result of mere chance. In these cases, it is probable that the numbers generated in the sequence are cycling or falling within non-random boundaries. These effects add to the interest of the results and are to be welcomed rather than avoided.

In choosing values for the constants 'a' and 'b' , digits were generally limited to such integers as 1, 3, 7, 9 which produce a variety of resulting digits; rather than 0, 5 and even numbers, which tend to be self-propagating.

Stochastic matrices were generated by working through the matrix a line at a time, and setting each element equal to 0 or a random number in the range (0 , 1) . A certain amount of experiment was necessary to arrive at appropriate probabilities to determine which of the two options to follow. Too many elements equal to 0 produce a very rigourously defined structure which is too predictable and potentially boring. Too few elements equal to 0 produce too many options in the matrix, which will introduce too much variety in the resulting output and obscure any evident pattern. On completion of each line, each element was once again divided by the sum of the elements on that line to arrive at the final probabilities which should all total unity.

In some later experiments, with the probability set in favour of many zeros, it sometimes happened that all elements ended up equal to 0 ; in which case it was necessary to insert a test to detect if this happened and avoid subsequently attempting to divide by 0 .

The following block demonstrates the construction of the 12×12 stochastic matrix *macros* :

```
comment matrix macros constructed;
for i := 1 step 1 until 12 do
begin
   sum := 0;
   for j := 1 step 1 until 12 do
   if rnd(3) > 1 then macros[i,j] := 0
   else
   begin
      macros[i.j] := rndec(3.57);
      sum := sum + macros[i.j];
   end;
   for j := 1 step 1 until 12 do
   macros i,j  := macros[i,j]/sum;
end;
```

The stochastic matrices were used to generate sequences by taking a random number in the range (0 , 1) , progressively summing the probabilities along the line of the matrix corresponding to the current value until that sum exceeded the random number, at which point the loop was abandoned and the current value subsequently became the number of the column at which that occured. This is probably clearer in an actual example of the technique being used to select appropriate numbered operations under the direction of the *macros* matrix. (The block would be executed many time during an actual run of the program.)

i holds the current value, and j the following value, which is being computed:

```
    begin
       .

       .

       .

       comment macros matrix in action;
       sum := 0;
       a  := rndec(8.517);
       for j := 1 step 1 until 12 do
       begin
          sum := sum + macros[i,j];
          if sum > a then goto work;
       end;
work:
       .

       .              (rest of program in which value of  j  is used)

       .
       i := j;
       .

    end          (control is returned to the beginning of the block)
```

At the beginning of the program the initial value of i is chosen at random.

It can be seen that in the case where all elements in a line of the matrix are equal to zero, the loop will be completed, and the 'next' value will be equal to the number of the last column, in the above example that would be 12 . This seems to be an adequate default arrangement.

The length of the piece is established at the beginning of the program, then blocks are added on until this required length is reached.

The first block length is chosen at random; subsequent lengths may be related to the previous length by deliberately being chosen to form a contrast: for example the probability is increased for a long section to be followed by a short one. The current block length is stored in the variable *secs* :

```
if rnd(3) = 1 then secs := rnd(20 - secs) + 1
              else secs := rnd(20) + 1;
```

In the above case the maximum block length is 21 .

The different blocks are arranged stochastically, and the elements inside each block are further generated according to stochastic schemes. These are detailed in the following descriptions of the appropriate pieces in which they are used.

6. The Individual Compositions

leap, *maytricks* and *pursuit*

For these three pieces, the same basic micro-structuring blocks were used, being of the following types:

1 a single sustained note, the pitch chosen at random

2 a single repeated note, the pitch chosen at random and with rhythm constructed with a bias towards smaller durations using the function:

rnd(rnd(8))

3 pitches chosen by the performer

4 short pitches, played pizzicato by stringed instruments, notated graphically and positioned at random

5 glissandi, or their nearest wind equivalent, notated graphically and positioned at random

6 ascending chromatic runs, starting note and length determined at random

7 a rest, i.e. silence

8 0 - order stochastic melody
i.e. notes chosen at random

9 1 - order stochastic melody
i.e. notes chosen according to simple probabilities which are stored in a one-dimensional array

10 2 - order stochastic melody
i.e. notes chosen according to digram probabilities governing the arrangement of adjacent note pairs. The probabilities are stored in a two-dimensional stochastic matrix which is used in the same way as the *macros* matrix above.

11 3 - order stochastic melody
i.e. notes are chosen according to trigram probabilities which are stored in a three-dimensional matrix, extending the methods used above, which produces an even more clearly defined pattern as the note occurences are determined in overlapping groups of three.

12 shapes of random size, position and colour, to be used as a basis for improvisation by the performer

Computer music coming at the point where two apparently totally opposite ideas merge in the total organisation of planned and structured randomness, it seemed not unreasonable to admit opportunity for structured improvisation as part of the formal process.

In practical terms, this would probably work well in a solo or very small chamber combination, giving an opportunity for a performer to show off his skills and favourite techniques in cadenza-like fashion; but even though it decorates the score nicely, it is unlikely to be effective in larger groups; and in any case, most orchestral players cataplectically dislike improvisatory parts which lack clear cut, traditionally notated, playing instructions. Consequently, this idea was dropped in subsequent experiments.

In each case of stochastic melody generation, a fourteen element basis is used. This corresponds to a melody within a range of twelve semitones, and the remaining two elements are used to define the rhythm. The thirteenth indicates the note is to be held over for another rhythmic unit, represented as two stars (**) in the computer output; the fourteenth indicates a rest, represented as two dashes (--) .

A procedure makes the necessary adjustments for this, and also puts the melody in the chosen register for the instrument:

here,	f	is the current note value
	regist	is the lower bound of the melody range, which is evaluated from the appropriate range for the instrument which is read in as data

```
procedure prinst(f,regist);
   comment prints out note values and rests;
   integer f,regist;
   begin
           if f = 13 then writetext ('('%**%%')')
      else if f = 14 then writetext ('('%--%%')')
      else print(f + regist,2,0);
   end;
```

Dynamics were evaluated on the random walk principle. This was not programmed in terms of a stochastic matrix, which would be inefficient as the matrix would consist mainly of zeros, but simply by moving the value up or down a unit reflecting it off the upper and lower bounds.

A scale of eight dynamic values was used (transcribed as *ppp, pp, p, mp, mf, f, ff, fff*) :

```
comment dynamics evaluated;
dyn := dyn + rnd(3) - 2;
if dyn < 1 then dyn := 1
else if dyn > 8 then dyn := 8;
```

For *maytricks* up to four different dynamic systems were defined and one system alloted to each instrument.

In the solo piece *leap* , the performer literally undertakes a random walk following a sequence of positions evaluated and plotted in a similar way to the dynamic system described above. [1]

For the orchestral piece *maytricks* , a density pattern was superimposed to vary the intensity of instrumental sound and provide relief for the ear. When many instruments are playing together, the ear whould be able to perceive within the dense sound continuum, the relative densities of the various micro-structures in combination; but only in a general sense, as the nuances of the wrything sound mass make themselves evident and it is possible to observe the dominance of some structures over others. When fewer instruments are playing, the subtle variations in the micro-structures themselves become more evident and it is possible to be aware

[1] see appendix IIa

of continual interplay of melodic detail as the parts are married in a capricious counterpoint where a different sort of attention is required. There are various shades of intermediary experience between the two extremes.

The flow-chart overleaf (fig 2) gives the broad outline of the logical structure of the compositional process.

The following list of instruments and ranges, was read in as data:

piccolo	39	71
flute	37	73
oboe	35	66
cor anglais	36	65
clarinet	29	73
bass clarinet	28	65
bassoon	11	47
double bassoon	11	40
alto saxophone	35	66
trumpet	31	61
cornet	31	61
horn I	19	61
horn II	19	61
horn III	19	61
horn IV	19	61
trombone	17	47
bass trombone	14	44
tuba	6	42
harp	1	81
piano	1	82
celeste	25	73
xylophone	30	73
violin Ia	32	72
violin Ib	32	72
violin IIa	32	72
violin IIb	32	72
viola I	25	61
viola II	25	61
cello I	13	49
cello II	13	49
double bass	17	46

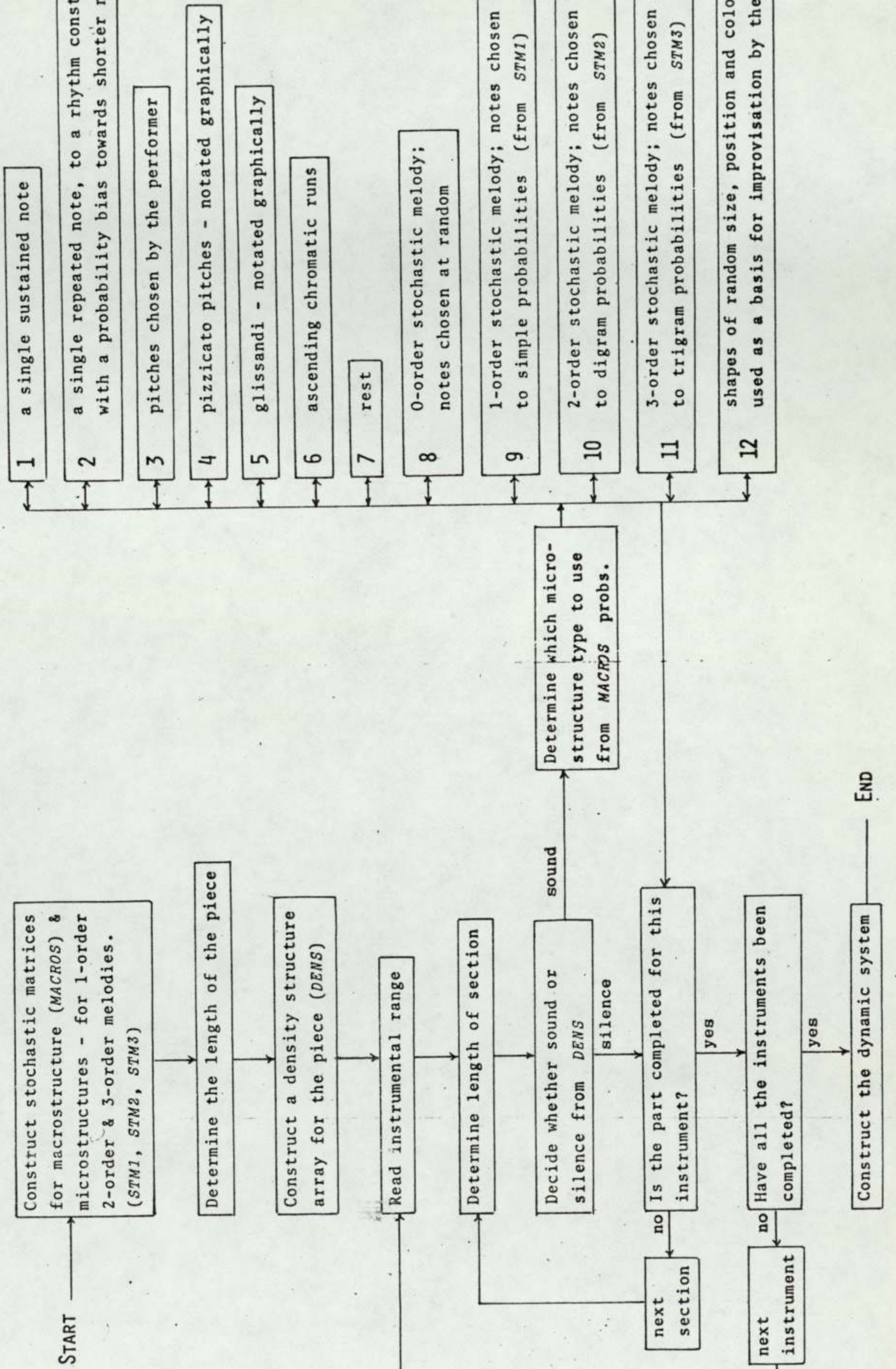
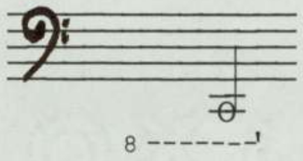
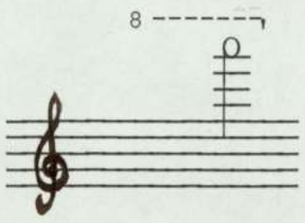

START
Construct stochastic matrices for macrostructure (MACROS) & microstructures - for 1-order 2-order & 3-order melodies. (STM1, STM2, STM3)
Determine the length of the piece
Construct a density structure array for the piece (DENS)
Read instrumental range
Determine length of section
Decide whether sound or silence from DENS
sound
silence
Determine which micro-structure type to use from MACROS probs.
Is the part completed for this instrument?
no
yes
next section
Have all the instruments been completed?
no
yes
next instrument
Construct the dynamic system
END
1 a single sustained note
2 a single repeated note, to a rhythm constructed with a probability bias towards shorter notes
3 pitches chosen by the performer
4 pizzicato pitches - notated graphically
5 glissandi - notated graphically
6 ascending chromatic runs
7 rest
8 0-order stochastic melody; notes chosen at random
9 1-order stochastic melody; notes chosen according to simple probabilities (from STM1)
10 2-order stochastic melody; notes chosen according to digram probabilities (from STM2)
11 3-order stochastic melody; notes chosen according to trigram probabilities (from STM3)
12 shapes of random size, position and colour to be used as a basis for improvisation by the performer

fig 2: simplified compositional flowchart for *maytricks* and *pursuit*.

The notes are coded as integers, beginning with 1 on the C three octaves below middle C :



and covering the whole range of semitones up to the A three and a half octaves above middle C , which is note 82 :



The data and output are given in terms of written ranges, not actual sounding pitches.

It might be noticed on looking at the program, [1] that there are certain inconsistencies in the names of variables. This has arisen where the program has been adapted and expanded causing a variable to assume a new rôle. For example, the variable *movt* initially stored the current movement number, when the program was dealing with a number of movements for one instrument; but in the program version listed here, it stores the current instrument number as the program is now constructing a number of instrumental parts for just one movement.

[1] see appendix Ia

Since each instrumental part is constructed unsing the same probabilities, it is quite valid and consistent to omit any instrumental parts and still preserve the overall probability structure and integrity of the piece, though as the number of instruments diminishes, the character of the music changes as explained above. [1]

In particular, six string parts were extracted to make the piece *pursuit* , which is more practical from a performing point of view. The title *pursuit* was chosen to represent the sense of seeking the overall design in listening to the piece, to describe the way in which the parts appear to chase each other about, and to convey the idea of a general search for the elusive.

Examining the scores, various patterns are evident. One rather startling observation, is that the structure defined by the 1-order stochastic array corresponds exactly to a pentatonic scale. The chances of this occuring were slim: from the computer program, the probability of any given five-note combination occuring is:

$$\left(\frac{1}{3}\right)^5 \left(\frac{2}{3}\right)^7 = 0{\cdot}000\,240\,855$$

Any pentatonic scale will fall into one of the following three interval patterns showing the number of semitones between adjacent notes:

1 1 1 1 3　　　　1 1 1 2 2　　　　1 1 2 1 2

Thus there are $3 \times 12 = 36$ different pentatonic scales, which gives an overall probability of $0{\cdot}008\,671$, or just under 1 in 116 .

[1] see also appendix II b

The aural result of this is that pleasant, possibly folk-song like or eastern-sounding sections could occur occasionally, perhaps even producing Balinese Gamelan-like effects: for example in the last few bars of *pursuit.* [1]

Other noticeable motifs are rocking groups of alternating minor sevenths, generated by the 2-order stochastic matrix, and prominent intervals of the augmented fourth.

Pattern generated by 3-order techniques may be difficult to spot in a small sample. It is possible that a sequence will never 'break through' into what may be a very structured section waiting for exploitation in some unused area of the matrix, where almost-closed sub-systems might occur. This could be represented by the diagram on the following page (fig 3):

---

see Appendix II c

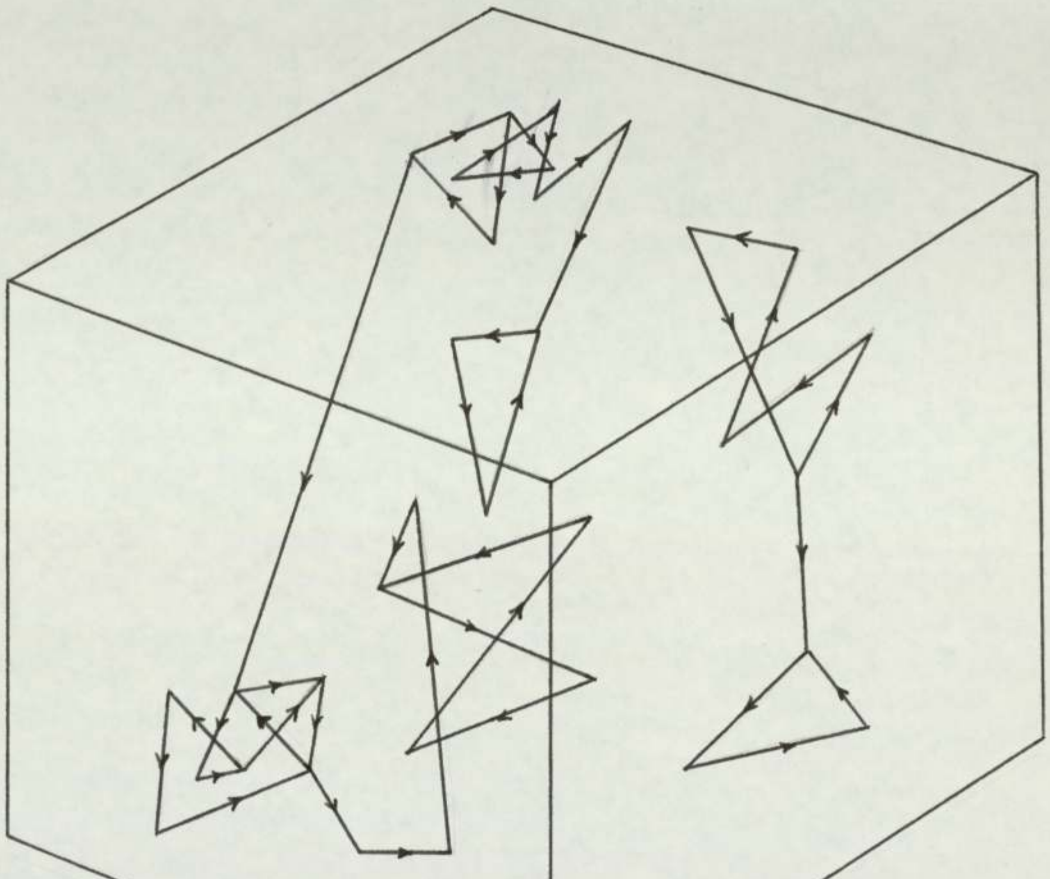


fig 3: an example of possible paths of patterning processes implicit in a 3-order stochastic matrix.

light

The structures described above are effective for fairly short pieces of music, of a particular character, but in an extended composition, the ear quickly tires through lack of evident, dramatic change. The above programs were adapted and extended to include provision for imposing a composed structure upon the composition. In this way descriptive or programmatic pieces can be produced.

The imposed form is set out as a sequence of ten numbers corresponding to the order of structure types required, and this is read into an array. The computer decides whether to make use of the imposed form at a particular instant in the composition or whether to stick to the stochastic form also potentially present.

The probability governing this decision can be varied by the program user. The lines below, from the program, illustrate this process:

```
comment structure type determined;
if rnd(2) = 1 then
goto type [ form[(10 k - 1)'/'totsecs + 1]]
else if rnd(den[k]) = 1 then goto rest
else goto type[j];
```

type	is a switch variable directing computer control to the appropriate part of the program according to the value of the subscript
form	is the array containing the ten values of the imposed form

k	is the current time position in working through the piece
totsecs	is the total length of the piece
den	is an array of values defining the density structure of the piece
j	is the current micro-structure type which has been evaluated from the *macros* matrix in the same way as has been described above
rest	labels the instruction to print out silence for the length of the current block

Since block lengths are variable, and block beginnings are therefore unlikely to coincide in different parts, the change from one form type to another is consequently staggered between the various instruments in a complete movement.

For the composition *light* , certain structure types were redefined:

1	a single sustained note
2	a single repeated note, but with a steady unchanging rhythm
3, 4	short, graphically notated pitches
5	random pitches

6	descending chromatic runs
7	ascending chromatic runs
8	silence
9	1 - order stochastic melody
10	2 - order stochastic melody
11	3 - order stochastic melody
12	double another part

Using these type numbers, appropriate sets of data were defined for a seven movement work on the theme of light, taking as a basis various Biblical references to light which possess expresive potential. Perhaps this betrays shades of Messiaen?

The source for each movement is quoted below, and followed by the corresponding ten-element array of structural types chosen from those listed above, which attempt to define a musical form corresponding to the ideas implicit in the words.

1 coming of light

7 7 7 1 7 1 1 7 1 7

"Arise, shine; for your light has come."
(Isaiah 60:1)

2 creation of light

5 5 5 9 5 1 1 1 1 1

"and God said, 'Let there be light'; and there was light. And God saw that the light was good; and God separated the light from the darkness."
(Genesis 1:3-4)

3 light of God

9 9 1 9 9 1 1 9 1 1

"God is light and in Him is no darkness at all."
(1 John 1:5)

4	light of the world 1 1 9 1 10 1 6 2 1 2	"the true light that enlightens every man was coming into the world." (John 1:9) "Jesus spoke to them saying, 'I am the light of the world;'" (John 8:12)
5	light of life 9 10 1 11 11 1 11 11 1 11	"he who follows me will not walk in darkness, but will have the light of life." (John 8:12)
6	children of light 1 1 1 10 1 6 10 1 10 10	"while you have the light, believe in the light, that you may become sons of light." (John 12:36) "walk as children of light." (Ephesians 5:8)
7	light of eternity 1 1 1 9 1 1 1 1 1 1	"the Lord will be your everlasting light." (Isaiah 60:19)

It can be seen that the first movement, representing the coming of light, begins with ascending chromatic scale passages, which are gradually inset with a series of long sustained chords. The second movement starts off totally at random, the disorder representing the chaos of darkness, out of which emerges the ordered, 'combed', linear strands of light. And so on in the subsequent movements. These musical forms can be perceived quite clearly. [1]

An ensemble of instruments was chosen appropriate for the theme of light: two flutes, trumpet, harp, vibraphone and strings, which may be solo or ensemble.

---

[1] see appendix II d

In the earlier pieces, no attempt was made to introduce any variety in the metre. The time space was merely, for convenience, divided into units of eight. However, since it is inevitable that performers will introduce some sense of metre, to add greater variety, the metre and speed are varied over the greater time span in *light* . The time signatures were chosen at random from the set:

$$\begin{matrix} 3 & 4 & 3 & 4 \\ 8 & 8 & 4 & 4 \end{matrix}$$

and the tempo is set with the beat unit occuring between sixty and one hundred and eighty times per second. These large variations in speed and metrical pattern add a lot of variety to the structure.

symposium

PLATO : " διαφερομενον αὐτο αὑτῳ συμφερεσθαι, ὥσπερ αρμονιαν το ξου τε και λυρας "
(The symposium)

(" a unity agrees with itself by being at variance, like the harmony of a bow or lyre ")

symposium makes use of the same program, but the aim in defining the form array for each movement was to maximise contrasts, both within each movement and over the piece as a whole, in order to bring out even more clear-cut structural differentiation. The idea of a 'symposium', a discussion of opposing viewpoint which can potentially lead to either reconcilitaion or increased conflict, was chosen as the basis for the piece.

It is scored for a group of ten musicians who should ideally be seated around a table:

	horn trumpet	
bassoon		violin
clarinet		violin
oboe		viola
flute		bass

Each movement presents an outworking of a dialectic defined between two (or in the case of the third movement, three) structural types.

With the types numbered as above,[1] the data was defined as follows:

1.	5	5	12	12	5	12	12	5	5	12
2.	2	1	1	1	2	1	2	2	2	1
3.	11	10	9	10	11	9	11	9	10	9
4.	4	8	4	4	8	8	4	4	8	4
5.	7	7	6	7	7	6	6	7	6	6

The five movements are:

1. alignment

   Tensions are created by variations between points of total agreement, coalescences, (*αγομολογησωμεθα*) and total variance.

2. assertion

   The dialogue is between insistant repetition of a heavy-handed presentation of opinions, and the calm affimation of passive, gentle interludes.

3. imbroglio

   This is a dense and complex three-fold intermingling of melodies of varying states of complexity.

---

[1] see pages 54 and 55

4. excursus

A gentle, off-the-point aside; a textural experiment which contrasts hesitant points of sound against contemplative silence.

5. dissent

Here is set up the direct oppositon of descending and ascending phrases. There is no reconciliation (*και διαλλαγη*).

No single argument is associated with a particular instrument, but as time passes, one particular argument will tend to dominate the structure.

The form patterns are illustrated diagrammatically overleaf (fig 4).

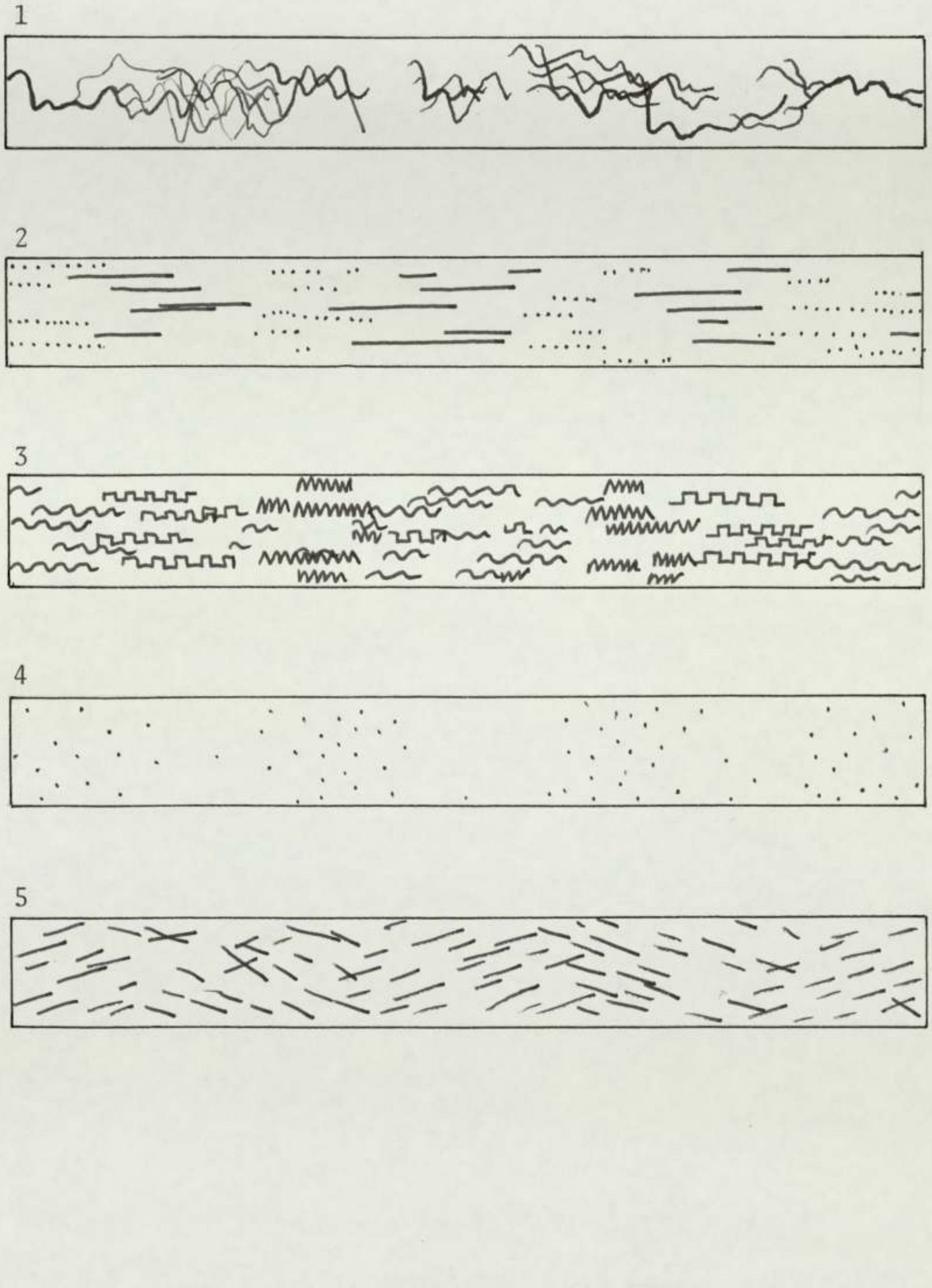


fig 4: diagrammatic representation of the form structures used in *symposium* .

Text studies

An experiment was carried out to investigate a method of superimposing pattern on to a grammatical structure. It was decided to make use of a grammar to generate sentences of English which can then be easily examined straight away and obviates the need for lengthy and tedious transcription of music into standard notation. In effect it could be considered to be a species of music, as forms of sounds are being generated; the pattern is what is important rather than the meaning of the words, although any unexpected resultant combinations provide an extra dimension of interest.

A lexicon of words were stored in the three-dimensional array *lex* . This stores eleven word-classes, each containing up to twenty five words of up to twelve letters. The eleven word-classes used are:

1	adjective
2	adverb
3	article
4	preposition
5	conjunction
6	noun
7	auxiliary
8	pronoun objects
9	pronoun subjects
10	past tense verbs
11	present tense verbs

These classes do not necessarily correspond to those used in conventional grammars. The 'conjunction' class, for example, includes punctuation marks, and the 'article' class includes possessive pronouns. The lexicon is read into the matrix using the character handling facilities in the 1900 version of ALGOL 60. The words are separated by a slash (/), the eleven-element array *words* holds the number of words in each word-class:

```
      for i := 1 step 1 until 11 do
      begin
         words[i] := read;
         for j := 1 step 1 until words[i] do
         begin
            for k := 1 step 1 until 12 do
            begin
     on:         char := readch;
               if char = code('('el')') then goto on;
               lex[i,j,k] := char;
               if char = code('('/')') then goto nextj;
            end k;
  nextj: end j;
      end i;
```

At appropriate points in the program, words were selected from the lexicon using the procedure *choose(i2)* where *i2* is the number of the word-class as listed above. The procedure includes provision for leaving a 'slot' empty in choosing from the first four word-classes, where the appearance of a word in the grammar is optional. The control probabilities for those cases are stored in the four-element array *mn* :

```
      procedure choose(i2);
         begin
            integer i2,j2,k2;
            if i2 < 5 then
            begin
               if rndec(2.11341) > mn then goto fina
            end;
         j2 := rnd(words[i2]);
         for k2 := 1 step 1 until 12 do
         begin
            if lex[i2,j2,k2] = code('('/')') then goto fina
            else printch(lex[i2,j2,k2]);
         end;
   fina: end;
```

The diagram on page 65 (fig 5) shows the basic grammar which was used.

The probabilities controlling the decisions in the grammar are stored in the array *n* . The procedure *ss* adds the letter 's' on to the end of the previous word. In ALGOL 60 the grammar appears so:

```
      for lines := 1 step 1 until length do
      begin
         choose(2);
         if rndec(8.459) > n[1] then
         begin
            choose(3);choose(1);
            if rndec(7.623) > n[2] then
            begin
               choose(6);
               if rndec(4.251) > n[3]
               then choose(10) else
               begin
                  choose(11);ss;
               end;
            end;
            else
            begin
               choose(6);ss;
               if rndec(3.1899) > n[4]
               then choose(10) else choose(11)
            end;
         end;
         else
         begin
            choose(9);
            if rndec(5.2771) > n[5] then
            begin
               choose(7);
               choose(11);
            end;
            else choose(10);
         end;
```

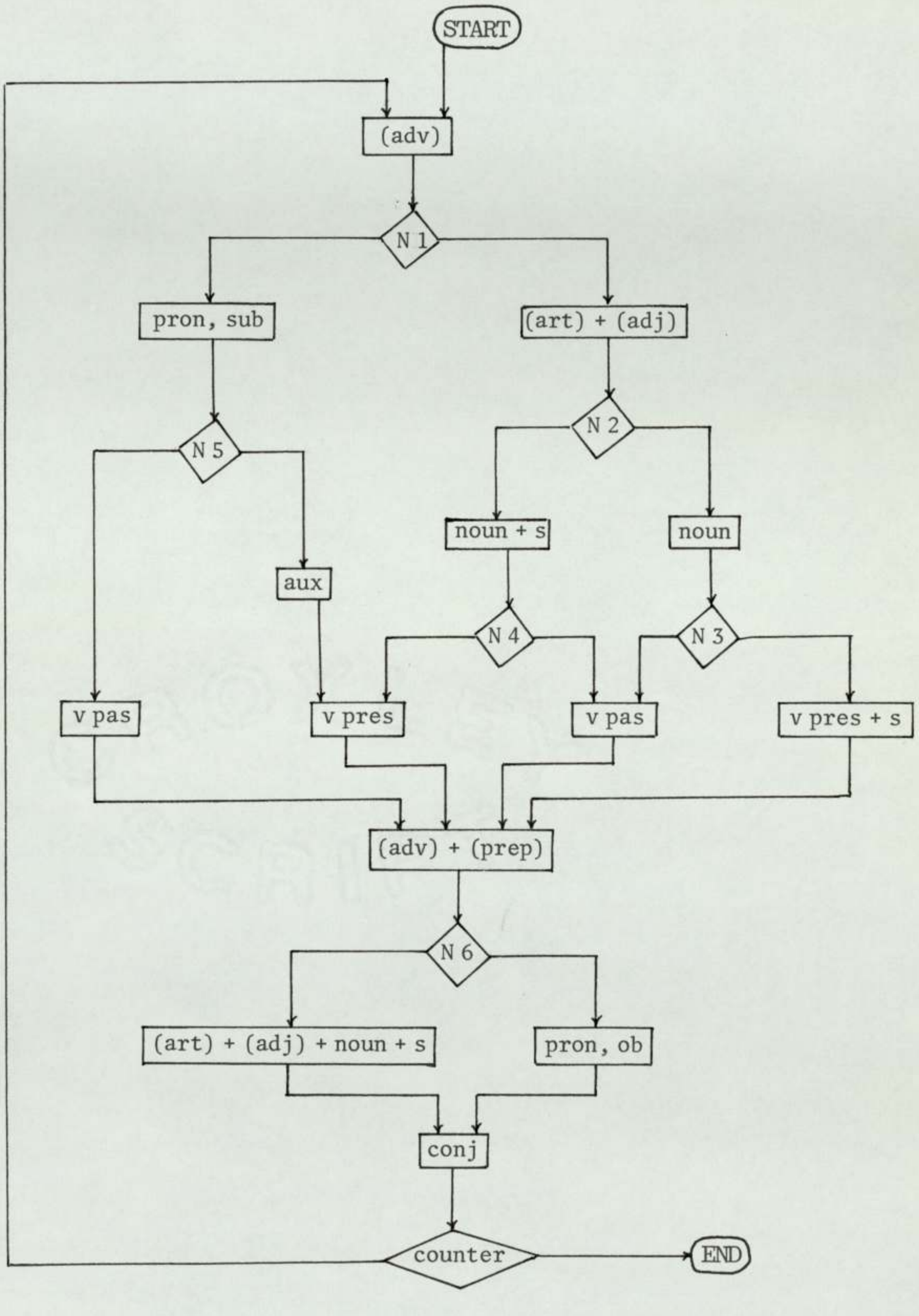


fig 5: grammar used for text studies.

```
          choose(2);
          choose(4);
          if rndec(1.2939) > n[6]
          then choose(8) else
          begin
             choose(3);
             choose(1);
             choose(6);ss;
          end;
          if lines = length then goto fin;
          newline(1);
          choose(5);
       end;
 fin: writetext('('.')');
```

Additional patterning can be imposed by varying the setting of the variable *rn* . If this is set near to 1, it causes the pseudo-random number generator *rndec* to deliver a constant value most of the time, so the program will tend to cycle through the same paths, and frequently make the same decisions. If *rn* is set near to 0, this will not happen:

```
       real procedure rndec(b);
          begin
             real b;
             x := 3.4561 * x + b;
             if rndecb(6.237) < rn then rndec := x1
             else rndec := x := 100 * x - entier(100 * x);
          end;
```

Setting of the othe variables produces the effects listed below:

mn1 near 0 implies few adjectives

near 1 implies many adjectives

similarly the setting of *mn2* , *3* and *4* controlls the number of adverbs, 'articles' and prepositions respectively.

n1 near 0 implies many nouns as subjects

near 1 implies many pronouns as subjects

n2 near 0 implies more single nouns

near 1 implies more plural nouns

n3 near 0 implies more past tense verbs used with single nouns

near 1 implies more present tense verbs used with single nouns

n4 near 0 implies more past tense verbs used with plural nouns

near 1 implies more present tense verbs used with plural nouns

n5 near 0 implies more past tense verbs used with pronouns

near 1 implies more present tense verbs used with pronouns

n6 near 0 implies more pronouns as objects

near 1 implies more nouns as objects

The following examples of the output illustrate these controlling mechanisms.

All values set evenly:

```
our pleasure creeps around on me


their silver cloud passed softly through its soft lights,
where it might feel on drifting breezes,
whilst these approached above past lovely heavens,
whilst some quiet shadows sparkle above by its moons,
and around he shall haunt by breezes;
hardly those went through some moons.
```

In the following examples, *n1* is set at 0 implying many nouns as subjects. Then firstly, contrasting values of *mn1* are compared.

mn1 = 0 , which implies few adjectives:

```
around our rests escape over my mists
whilst hardly heavens creep into shades
yet slowly the wonders flew overhead through you
```

but *mn* = 1 , implying many adjectives:

```
for ever patient breezes walk over her
whilst soon lovely clouds caught by my gentle powers
```

Now comparing differing values of *mn2* .

With *mn2* = 0 , implying few adverbs, output like the following appears:

```
her shimmering lights melt into drifting shadows
drifting vapours feel over dear shades
fair breaths rest by the dear heavens
our shifting wonder passed over shimmering shades
cool distance moved by drifting mists
```

and with *mn2* = 1 , implying many adverbs:

```
overhead intense surfaces move around withing my fair seas
around dear rests went freely through me
yet her dear breeze approached freely below us
and once gentle shade escapes soon in deep wonders
```

Finally, setting *n1* = 1 , implying many pronoun subjects, and with *mn1* = 0 , (few adjectives) produces:

```
freely it should float freely inside him
and hardly we might float above past this
as often they flew seldom over their forms
```

Certain developments and uses of the program have suggested themselves. For example, using a second lexicon of 'harder' words and forming a series of contrasting episodes. It ought to be possible to produce a play-like structure where the styles of different characters can be defined using carefully differentiated definition of the grammar parameters - for instance exploiting such contrasts as vague/precise , verbose/succinct , consistent/changeable ; and by adopting an appropriate lexical balance between 'gentle' and 'aggressive' or indeed any other contrasting vocabularies which it may be desirable to introduce. The use of a particular pattern in one section could provoke an appropriate response in a following section.

These techniques seem to have considerable similarity to the process of building musical structures, and exhibit significant potential. Nevertheless, their development was put aside in favour of pursuing another alternative approach from a specifically musical angle.

pianocomp

The nature of the instrumental parts produced with the *maytricks/light* programs was not really suited to the piano which, for example, does not have great sustaining powers. It was decided to work on a program which would produce output specifically pianistic in its nature by stochastic ordering of suitable sound shapes such as repeated chords, arpeggios and 'scale' passages.

The number of possible forms available was increased by defining basic procedures with variable parameters to change the form. Provision was also made for defining specific vertical relationships between form types, rather than continuing to rely only on implicit relationships as occur in the foregoing pieces.

Also in the previous experiments, each instrumental part or layer of the composition was related to a core skeleton, in that it could either adopt the skeleton pattern, or not. In *pianocomp* , the form type of each instant is related directly to its partner in time through the form matrix which controls not only the probabilities of sound events occuring one after another in a particular sequence, but also their co-incidence.

The five form procedures defined are:

1. *Chords* (c1, c2, c3, c4, c5) where the parameters c1,...,c5 stand for:

c1 the number of notes in the chord $(0 < c1 \leq 5)$

c2 the nature of repetition of the chord:

c2 = 1 : a repeated chord

c2 = 2 : alternating chords

c2 = 3 : changing chords, but previous notes may be used

c2 = 4 : all different

c3 the speed of repetition of a chord:

c3 = 1 : fast

c3 = 2 : medium

c3 = 3 : slow

c4 the spacing of a chord (related to c1):

c4 = 1 : clusters

c4 = 2 : two notes together, others spaced

c4 = 3 : no restriction

c5 the nature of the rhythm (related to c3):

c5 = 1 : a constant re-iterated rhythm

c5 = 2 : a repeating rhythm

c5 = 3 : a changing rhythm

This procedure can potentially produce up to 540 different pattern structures. For example, a call of the procedure with parameters as below:

```
chords(1,2,3,1,2)
```

would produce a sequence of two alternating notes with a fast ostinato pattern.

2. *Melodies* (m1, m2)

This procedure produces stochastic melodies within a restricted range of 17 semitones.

m1 indicates the order of organisation, from 0 = random, to third order structure

m2 indicates whether the melody should be doubled in octaves (if =2) or not (if =1)

3. *Runs* (r1, r2, r3, r4, r5) where the parameters r1,...,r5 stand for:

r1 whether scales (if =1) or arpeggios (if =2)

r2 whether diatonic (if =1) or chromatic scales (if =2) This is ignored if r1 = 2.

r3 the number of notes in the arpeggio $(1 \leq r3 \leq 5)$ ignored if r1 = 1

r4 the range in octaves $(1 \leq r4 \leq 6)$

r5 the direction of movement:

r5 = 1 : up

r5 = 2 : down

r5 = 3 : both up and down

4. *Trills* (tl)

where tl indicates the depth of the trill in semitones (1 or 2)

5. *Silence*

no parameters!

At the beginning of the program, the length of the piece is established (in bars) and the number of different form units calculated related to the length:

```
length:     rnd(600);
units:      rnd(length'/'60) + 3;
```

There will be between 3 and 13 form units used. At this stage the form units are merely numbers and are not yet associated with actual form types.

The stochastic matrix controlling the form structure, *stmf* is created. This is worked out so that there is a greater bias towards 0 entries when the number of units, and hence the matrix, is larger , to maximise the structure:

```
for i := 1 step 1 until units do
begin
   sum := 0.000001;
   for j := 1 step 1 until units do
   if rnd(units/4 + 1) > 1 then stmf[i,j] := 0 else
   begin
      stmf[i,j] := rndec(1.9733);
      sum := sum + stmf[i,j];
   end;
   for j := 1 step 1 until units do
   stmf[i,j] := stmf[i,j]/sum;
end;
```

Using the procedure *nxfm(*element*)* which produces the next form element from the matrix *stmf* (using the same technique as has already been described above, page 40), the *form* matrix is constructed. Again this is related to overall length, so that in a longer piece, form changes will occur less frequently than in shorter ones.

The form of the first layer is worked out to start with:

```
form[1,1] := rnd(units);
for j := 2 step 1 until length do
begin
   if rnd(length/150 + 8) < 7
   then form[1,j] := form[1,j-1]
   else form[1,j] := nxfm(form[1,j-1]);
end;
```

Then the form structure of susequent layers is worked out related to the first layer:

```
for i := 2 step 1 until layers do
for j := 1 step 1 until length do
begin
   if rnd(length/150 + 8) < 7
   then form[i,j] := form[1,j]
   else form[i,j] := nxfm(form[1,j]);
end;
```

The following diagram illustrates the structure of the relationships used in building up the form:

$$\begin{tikzcd}
\text{layer 1} & a_1 \arrow[r] \arrow[d] \arrow[dd, bend right] & b_1 \arrow[r] \arrow[d] \arrow[dd, bend right] & c_1 \arrow[r] \arrow[d] \arrow[dd, bend right] & d_1 \arrow[r] \arrow[d] \arrow[dd, bend right] & \cdots \\
\text{layer 2} & a_2 & b_2 & c_2 & d_2 & \\
\text{layer 3} & a_3 & b_3 & c_3 & d_3 & \\
\cdots & & & & & \text{etc.}
\end{tikzcd}$$

Next, actual form types are alloted to the unit numbers. The form types are stored in the array *type* and the parameters associated with each in the matrix *param* .

Provision is made for the user to specify the weighting of form types by feeding in data to a thirteen element array *datatype* containing the numbers of the form types in an appropriate ratio. For example:

```
1   1   1   1   1   2   2   2   3   3   3   4   5
```

would give a bias towards structures built from the *chords* procedure.

The types are allocated by choosing at random from the *datatype* array:

```
for i := 1 step 1 until units do
type[i] := datatype[rnd(13)];
```

The values of type parameters are then chosen at random within the bounds appropriate for each.[1] Obviously, in the case of some types, 5 (*silence*) for example, spaces reserved in the *param* matrix will be redundant.

The stochastic matrices for the various stochastic melodies are defined as in parevious programs.

A beat structure is worked out for the whole piece. The number of beats in each bar is made to change on average every 10 bars and the resulting pattern is stored in the array *beats*.

After defining the dynamic system, composition of each layer is then begun by working through the *form* array and directing control of the program to the appropriate procedure each time the form type changes. As each change is established, the position is stored in the variable *lower*, and the next change in *higher*; after completing the section, the new *lower* position takes the value of the old *higher*, the search procedes for the next *higher* bound, and so on until the array is exhausted:

[1] see *pianocomp* program in Appendix Id , statements 257 - 280

```
      comment i is the current layer number;
      lower := 1;
      for upper := lower + 1 step 1 until length do
      begin
         if form[i,upper] ne form[i,upper - 1] then
         begin
            f := form[i,lower];
            goto swtype[type[f]];
  swchor:   chords(param[f,1],param[f,2],param[f,3],param[f,4],param[f,5]);
            goto nxstp;
  swmelo:   melodies(param[f,1],param[f,2]);
            goto nxstp;
  swruns:   runs(param[f,1],param[f,2],param[f,3],param[f,4],param[f,5]);
            goto nxstp;
  swtril:   trills(param[f,1]);
            goto nxstp;
  swsile:   silence;
  nxstp:    lower := upper;
         end;
      end;
```

In operating each procedure, the procedure *beatcount* is used to work out how many beats there are in the section; the result is stored in *nobs* :

```
      procedure beatcount;
         begin
            nobs := 0;
            for j := lower step 1 until upper - 1 do
            nobs := nobs + beats[j];
         end;
```

The procedure *rhythm* is used to generate a rhythmic pattern for a specified number of beats, and with controllable bias of durations. If *speed* is set high, longer durations will occur and the speed of the rhythm will be faster. The procedure counts the number of actual notes it produces, the result appearing in *notes* ; and using *summ* provision is made for fitting the rhythm exactly into the required length:

```
     procedure rhythm(lenth,speed);
        value lenth,speed;
        integer lenth,speed;
        begin
           integer kl,summ;
           summ := notes := 0;
   newkl:  kl := rnd(rnd(2 ↑ speed));
           summ := summ + kl;
           notes := notes + 1;
           if summ > lenth then
           begin
              kl := kl - (summ - lenth);
              if kl = 0 then notes := notes - 1;
              print(kl,1,0);
              goto last;
           end;
           else print(kl,1,0);
           goto newkl;
     last: newline(1);
        end;
```

In each procedure type the rhythm is established first, and the appropriate number of pitches are worked out afterwards.

In the *chords* procedure, use is made of the *beatcount* and *rhythm* procedures to work out rhythms appropriate to the constraints of the parameters.[1] To generate ostinato patterns, the procedure *ostinato* is used which produces a rhythm for one bar, and is called whenever the number of beats in a bar changes. The temporary variable *notemp* is introduced to get round the automatic re-calculating of the new value of *notes* , so that an automatic record is kept of the number of notes which will be needing pitches:

[1] see Appendix Id, statements 41 - 81

```
if c5 = 2 then
begin
   integer notemp;
   procedure ostinato;
   begin
      writetext('('ostinato%at')');
      print(j,3,0);
      writetext('('with%rhythm:')');
      rhythm(beats[j],c3);
      notemp := motemp + notes;
   end;
   notemp := 0;
   j := lower;
   ostinato;
   for j := lower + 1 step 1 until upper - 1 do
   if beats[j] ne beats[j-1] then ostinato;
   notes := notemp;
end;
```

The pitch of the first base note is chosen at random within the limited range of 71 semitones, which leaves room for a chord to be built above it if necessary; subsequent base pitches are chosen to follow at smaller intervals apart, the fuller the chords to be used:

$$p_i \longleftarrow p_{i-1} + \mathrm{rnd}(14 - 2 * c_1) - 7 + c_1 \qquad (1 \le c_1 \le 5)$$

The notes are reflected back, should they cross the upper and lower barriers.

On each base note a chord is built. The following relation makes sure that this falls within the span of the hand:

$$p_j \longleftarrow p_{j-1} + \mathrm{rnd}(13 - c_1 + j - p_{j-1} + p_1) - 1$$

This relation also ensures that the higher the value of c_1, the more restricted the initial range of choice, and that as the chord is built up, notes continue to be chosen from within the remaining gap between the last note and the octave above the first note.

All the above constraints when programmed into ALGOL 60 appear as follows:

```
begin
   integer q;
   integer array pitches 1:5 ;
   pitches[1] := rnd(71);
   for q := 1 step 1 until notes do
   begin
      pitches[1] := pitches[1] + rnd(14 - 2  cl) - 7 + cl;
      if pitches[1] < 1 then pitches[1] := rnd(14 - 2  cl)
      else if pitches[1] > 71
      then pitches[1] := 71 - rnd(14 - 2  cl);
      for j := 1 step 1 until cl do
      begin
         if j  1 then
         pitches[j] := pitches[j-1] + rnd(13 - cl + j - pitches[j-1]
                                                      + pitches[1]) - 1;
         print(pitches[j],2,0);
      end;
      newline(1);
   end;
end;
```

The melodies procedure operates more or less as the similar stochastic melody generators in the previous programs, except that the rhythm is generated separately to begin with, and the note selection processes have been condensed into one procedure, *stock* :

```
         procedure stock(order);
            integer order;
            begin
               switch sworder := sol,so2,so3;
               if order = 0 then
               print(range + rnd(17),2,0) else
               begin
                  sum := 0;
                  a := rndec(2.9147);
                  for p3 := 1 step 1 until 17 do
                  begin
                     goto sworder[order];
        sol:         sum := sum + stml[p3]; goto soskip;
        so2:         sum := sum + stm2[p2,p3]; goto soskip;
        so3:         sum := sum + stm3[pl,p2,p3];
        soskip:      if sum > a then
                     begin
                        print(range + p3,2,0);
                        goto finst;
                     end;
                  end;
        finst:    pl := p2; p2 := p3;
               end;
            end;
```

The *runs* procedure works similarly to *chords* . The nature of the runs are determined from the type parameters. The notes in arpeggio patterns are chosen in the same way as the notes in the chord are determined in the *chords* procedure, so as to fit well under the hand.

The *trills* and *silence* procedures are self-explanatory.

The examples of output from this program should be consulted in Appendix Id and IIf .

This program has produced some quite interesting and worthwhile results, which, if incorporated within superimposed pattern systems like the *light* and *symposium* pieces above, ought to have potential for developing into substantial and worthwhile compositions. They could easily serve as the basis of valuable technical studies for pianists, quite apart from the possibility of any higher aspirations in its use.

7. Extensions and Conclusions

When the use of computers to compose was first mooted some twenty years ago, it aroused a lot of attention and it seemed that computers were going to play a significant rôle in the development of new music; but interest soon waned as the early experiments were unable to sustain their impact. However, work in computer music did not stop, but all efforts were diverted into the development of digital sound synthesis facilities. A lot has been accomplished in that area, but now, interest in composing systems has re-awakened as it has been recognised how valuable, and indeed essential, their use is, to make efficient use of new digital sound sources.

Very specific instructions are necessary in using computer sound synthesis programs, and to produce even short lengths of sound invariably needs an inordinately large amount of data. Using a computer to assist in the compositional process, a composer can delegate a lot of tedious tasks to an automatic routine. This is in no way abdicating the composer's responsibilities. Compare, for example, writing for the piano, when a composer makes use of a ready-made sound source, along with the pianist who interprets the sound for him. He may make six or seven conscious structuring decisions in composing five seconds of sound. In using a computer, aiming for the same richness and variety, it may be necessary to program hundreds of simple instructions to achieve a similar effect. If the composer designs an 'automatic' algorithmic environment in which he wishes to work, he can then concentrate on those aspects of the compositional process which he considers more important. He need not necessarily design a complex timbral 'instrument' which he wishes to use in composition, but may employ, for example, an automatic pitch environment on to which he superimposes other compositional ideas; or he may design a complex environment which logically interrelates many different control parameters, in which case he may decide that such a system is a sufficient musical statement in itself.

The stochastic programs which have been described, are examples of compositional algorithms which can usefully be incorporated into synthesis systems. Discussing the use of such programs in this way, shows how they can be intrinsically valuable in their own right, in approaching musical composition in an open and uncluttered frame of mind. There is some value in a composer being forced to take nothing for granted; it offers him valuable mind-expanding experience to have to penetrate his technique with the accuracy and clarity required by a computer.

Work has already begun on designing integrated computer systems. Donald Byrd descibes a system being developed at Illinois [1] which incorporates Xenakis' stochastic program, and his own *MUSC* composing program (described above [2]) which can be interfaced to sound synthesis and music plotting facilities. A system with a fairly sophisticated compositional facility at the Institute of Sonology, Utrecht, has also been described by Truax. [3]

Lejaren Hiller has compared a composer using the computer to various artistic practices in the past. He suggests a parallel with such an artist as Tintoreto in Venice, who would merely sketch out a design for

---

[1] Byrd
[2] page 21
[3] Truax

a painting and leave his pupils, well schooled in his style, to finish off the details. A similar practice is the Baroque composers' custom of writing a figured bass on which keyboard players would expand according to customary practice, allowing them a little scope for subtle invention of their own. And more recently, it is common for composers of popular music to leave the instrumentation and arrangement of their work to 'lesser mortals'. These are valid comparisons, but the use of a computer can go much further than that. Where a composer has developed a program which is a definite reflection of a particular style and approach, and another composer makes subsequent use of it, imposing his own personality on the system, a genuine composers' co-operative is achieved.

It is possible to write compositional algorithms, describable in non-technical, subjective terms, and to develop heuristic systems which can be used by complete novices. Such systems would be of value not only to experienced composers and musicians, but also to children, and anyone who wished to create his own sounds. With the speedy dissemination of ideas and new developments over the media, the listening ears of contemporary society are expanding in their horizons and growing more aware. There is probably more exposure to music today than there has ever been in history, and in most people there is a desire to create sounds as well as to listen to them. The sound shapes of electronic music, of 'space' sounds, are familiar to everybody. These sounds are not replacing the sounds of history, but are expanding with them. Bach and Mozart have benefited from commercial television as well!

As man's leisure time increases, so his need for appropriate diversion does also. In the United States, simple digital sound modules are already being marketed to plug into domestic microprocessor systems. Soon these could become more than mere toys.

Way back in 1963, Xenakis wrote the following words. His aspirations are now all but realised:

> "With the aid of electronic computers the composer turns into an astronaut pressing the buttons of his musical spaceship to introduce co-ordinates and keep the course of his cosmic vessel, sailing in the space of sound, across sonic constellations and galaxies, controlling from the ease of an armchair what his imagination could formerly glimpse only as a distant dream." [1]

[1] Xenakis p. 144

APPENDIX I

PROGRAMS

1a. *leap*

This program is the same as that used for *maytricks* and *pursuit* .

The resulting output is very similar to that of the *light* program, an example of which is on page 106 .

```
'BEGIN'
   'REAL''ARRAY'MACROS[1:12,1:12],STM1[1:14],STM2[1:14,1:14],
                STM3[1:14,1:14,1:14];
   'INTEGER'I,J,K,P,Q,SECS,MOVT,NOMOV,NOTIMES,II,JJ,KK,REG,
             TOTSECS,PART,DENS,HIGH,LOW,INSTS;
   'INTEGER''ARRAY'DEN[1:600];
   'REAL'SUM,X,A;
   'REAL''PROCEDURE'RNDEC(B);
      'REAL'B;
      'BEGIN'X:=9.2351*X+B;
         RNDEC:=X:=100*X-ENTIER(100*X);
      'END';
   'INTEGER''PROCEDURE'RND(N);
      'INTEGER'N;
      'BEGIN'RND:=ENTIER(RNDEC(3.572)*N)+1'END';
   'PROCEDURE'PRINST(F,REGIST);
      'INTEGER'F,REGIST;

      'BEGIN'
        'IF'F=13'THEN'WRITETEXT('('%**%%')')
  'ELSE''IF'F=14'THEN'WRITETEXT('('%--%%')')
  'ELSE'PRINT(F+REGIST,2,0);
      'END';
  'SWITCH'TYPE:=LONG,REP,PCH1,PCH2,PCH3,RUN1,RUN2,REST,ST1,ST2,ST3,COL
X:=0.260852;
        'FOR'I:=1'STEP'1'UNTIL'12'DO'
        'BEGIN'
```

```
                     SUM:=0;
                     'FOR'J:=1'STEP'1'UNTIL'12'DO'
                     'IF'RND(3)>1'THEN'MACROS[I,J]:=0
                     'ELSE''BEGIN'MACROS[I,J]:=RNDEC(3.56);
                                 SUM:=SUM+MACROS[I,J];
                            'END';
                     'FOR'J:=1'STEP'1'UNTIL'12'DO'
                     MACROS[I,J]:=MACROS[I,J]/SUM;
                  'END';
                  'BEGIN'
                     'REAL'SUM1,SUM2,SUM3;
                     SUM1:=0;
                     'FOR'II:=1'STEP'1'UNTIL'14'DO'
                     'BEGIN'
                        'IF'RND(3)>1'THEN'
                        'BEGIN'
                           'IF'II>12'THEN''GOTO'P1;
                           STM1[II]:=0;
                        'END''ELSE'
        P1:             'BEGIN'
                           STM1[II]:=RNDEC(4.89);
                          SUM1:=SUM1+STM1[II];
                        'END';

                        SUM2:=0;
                        'FOR'JJ:=1'STEP'1'UNTIL'14'DO'
                        'BEGIN'
                           'IF'RND(5)>1'THEN'
                           'BEGIN'
                              'IF'JJ>12'THEN''GOTO'P2;
                              STM2[II,JJ]:=0;
                           'END''ELSE'
        P2:                'BEGIN'
                              STM2[II,JJ]:=RNDEC(2.47);
                              SUM2:=SUM2+STM2[II,JJ];
                           'END';
                           SUM3:=0;
                           'FOR'KK:=1'STEP'1'UNTIL'14'DO'
                           'BEGIN'
                              'IF'RND(6)>1'THEN'
                              'BEGIN'
                                 'IF'KK>12'THEN''GOTO'P3;
                                 STM3[II,JJ,KK]:=0;
                              'END''ELSE'
        P3:                   'BEGIN'
                                 STM3[II,JJ,KK]:=RNDEC(5.89);
                                 SUM3:=SUM3+STM3[II,JJ,KK];
                              'END';
                           'END';
                           'FOR'KK:=1'STEP'1'UNTIL'14'DO'
                           STM3[II,JJ,KK]:=STM3[II,JJ,KK]/SUM3;
                        'END';
                        'FOR'JJ:=1'STEP'1'UNTIL'14'DO'
                        STM2[II,JJ]:=STM2[II,JJ]/SUM2;
                    'END';
                    'FOR'II:=1'STEP'1'UNTIL'14'DO'
                    STM1[II]:=STM1[II]/SUM1;
                 'END';
                 INSTS:=READ;
                 TOTSECS:=RND(600);
                 NOMOV:=RND(5);
                 DEN[1]:=RND(ENTIER(INSTS/2));
                 'FOR'P:=2'STEP'1'UNTIL'TOTSECS'DO'
                 'BEGIN'
                    'IF'RND(3)=1'THEN'DEN[P]:=DEN[P-1]+RND(5)-3
                    'ELSE'DEN[P]:=DEN[P-1];
```

```
            'IF'DEN[P]<1'THEN'DEN[P]:=1'ELSE'
            'IF'DEN[P]>ENTIER(INSTS/2)'THEN'DEN[P]:=ENTIER(INSTS/2)
         'END';
       'FOR'MOVT:=1'STEP'1'UNTIL'INSTS'DO'
       'BEGIN'NEWLINE(3);
          WRITETEXT('('PART')');
          PRINT(MOVT,1,0);
          COPYTEXT('(':')');
          NEWLINE(1);
          WRITETEXT('('USE%DYNAMIC%SYSTEM')');
         PRINT(RND(NOMOV),1,0);
          LOW:=READ;
          HIGH:=READ;
          SECS:=RND(20);
          I:=RND(12);
          'FOR'K:=1'STEP'SECS'UNTIL'TOTSECS'DO'
          'BEGIN'
             SUM:=0;

             A:=RNDEC(8,557);
             'FOR'J:=1'STEP'1'UNTIL'12'DO'
             'BEGIN'
                SUM:=SUM+MACROS[I,J];
                'IF'SUM>A'THEN''GOTO'WORK;
             'END';
WORK;        NEWLINE(1);
             WRITETEXT('('******%SECTION')');
             PRINT(K,3,0);
             WRITETEXT('('%%TYPE')');PRINT(J,3,0);
             WRITETEXT('('%,.LASTS%FOR')');PRINT(SECS,3,0);
             WRITETEXT('('SECS ******')');
             'IF'RND(DEN[K])=1'THEN''GOTO'REST'ELSE'
             NEWLINE(1);
             'GOTO'TYPE[J];
LONG:        'BEGIN'
                WRITETEXT('('%LONG%:%PITCH')');
                PRINT(RND(HIGH-LOW)+LOW,2,0);
                'GOTO'NEXTK;
             'END';
REP;         'BEGIN'
                WRITETEXT('('%REPEATED%NOTE%;%PITCH')');
                PRINT(RND(HIGH-LOW)+LOW,2,0);
                WRITETEXT('('('C')'%RHYTHM:')');
                'FOR'P:=1'STEP'1'UNTIL'8*SECS'DO'
                'BEGIN'
                   Q:=RND(RND(8));
                   P:=P+Q-1;
                   PRINT(Q,1,0);
                'END';
                'GOTO'NEXTK;
             'END';
PCH1:        'BEGIN'
                WRITETEXT('('SELF%CHOSEN%PITCHES')');
                'GOTO'NEXTK;
             'END';
PCH2;        'BEGIN'
                WRITETEXT('('%SOUNDS%AT%THE%FOLLOWING%TIME-PITCH
                               %CO-ORDINATES:')');
               'FOR'P:=1'STEP'1'UNTIL'RND(5*SECS)'DO'
               'BEGIN'
               'IF'P/8=P'/'8'THEN'
               NEWLINE(1);
               PRINT(RND(10*SECS),3,0);
               WRITETEXT('(',')');
               PRINT(RND(50),2,0);
               'END':
```

```
              'GOTO'NEXTK;
             'END';
PCH3:        'BEGIN'
                 WRITETEXT('('%SOUNDS%AS%FOLLOWS')');
                 NEWLINE(1);
                 SPACE(40);
                 'FOR'P:=1'STEP'1'UNTIL'SECS'DO'
                 'BEGIN'
                   'IF'P/2=P'/'2'THEN'NEWLINE(1)
                   'ELSE'WRITETEXT('('%%%%%')');
                   'FOR'Q:=1'STEP'1'UNTIL'8'DO'
                   'BEGIN''INTEGER'R;
                      R:=RND(3);

                        'IF'R=1'THEN'
                 PRINT(RND(HIGH-LOW)+LOW,2,0)
                 'ELSE''IF'R=2'THEN'WRITETEXT('('%**%%')')
                 'ELSE''IF'R=3'THEN'WRITETEXT('('%--%%')');
                   'END'Q;
                 'END'P;
                 'GOTO'NEXTK;
             'END'BLOCK;
RUN1:        'BEGIN'
                 'INTEGER'XX,YY;
                 WRITETEXT('('%NOTE%RUNS%AT%THE%FOLLOWING%TIME-PITCH
                              %CO-ORDINATES:')');
                 NEWLINE(1);
                 'FOR'P:=1'STEP'1'UNTIL'RND(SECS)'DO'
                 'BEGIN'
                  'IF'P/3=P'/'3'THEN'NEWLINE(1);
                  XX:=RND((10*SECS)-20);
                  YY:=RND(30);
                  WRITETEXT('('%%%%FROM')');
                  'IF'XX<0'THEN'XX:=0;
                  PRINT(XX,3,0);
                  WRITETEXT('(',')');
                  PRINT(YY,3,0);
                  WRITETEXT('('TO')');
                  PRINT(XX+RND(20),2,0);
                  WRITETEXT('(',')');
                  PRINT(YY+RND(10),3,0);
                 'END';
                 'GOTO'NEXTK;
             'END';
RUN2:        'BEGIN'
                 'INTEGER'TS;TS:=0;
                 WRITETEXT('('%RUNS%AS%FOLLOWS:')');
                'FOR'P:=1'STEP'1'UNTIL'8*SECS'DO'
                 'BEGIN'
                   'IF'TS/4=TS'/'4'THEN'NEWLINE(1);
                   'IF'RND(2)=1'THEN'
                   'BEGIN'
                      Q:=RND(12);
                      WRITETEXT('('%%%%%%FROM')');
                      PRINT(RND(HIGH-LOW-12)+LOW,2,0);
                      WRITETEXT('('%FOR')');
                      PRINT(Q,2,0);
                   'END'
                   'ELSE'
                   'BEGIN'
                      Q:=RND(30);
                      WRITETEXT('('%%REST%FOR')');
                      PRINT(Q'/'8,2,0);
                      WRITETEXT('('%SECS')');
                      PRINT(Q-(Q'/'8)*8,2,0);
                   'END':
```

```
                  P:=P+Q;
                  TS:=TS+1;
               'END';
               'GOTO'NEXTK;
            'END';
REST:       'BEGIN'
               WRITETEXT('('%%%REST')');
               'GOTO'NEXTK;

            'END';
ST1:        'BEGIN'
               WRITETEXT('('1-ORDER%STOCHASTIC%TUNE'('C')'')');
               SPACE(40);
               REG:=RND(HIGH-LOW-12)+LOW;
               'FOR'P:=1'STEP'1'UNTIL'SECS'DO'
               'BEGIN'
                  'IF'P/2=P'/'2'THEN'NEWLINE(1)'ELSE'
                     WRITETEXT('('%%%%%')');
                  'FOR'Q:=1'STEP'1'UNTIL'8'DO'
                  'BEGIN'
                     SUM:=0;
                     A:=RNDEC(2,9);
                     'FOR'II:=1'STEP'1'UNTIL'14'DO'
                     'BEGIN'
                        SUM:=SUM+STM1[II];
                        'IF'SUM>A'THEN'
                        'BEGIN'
                           PRINST(II,REG);
                           'GOTO'NUQ1;
                        'END';
                     'END';
NUQ1:             'END';
               'END';
               'GOTO'NEXTK;
            'END';
ST2:        'BEGIN'
               WRITETEXT('('2-ORDER%STOCHASTIC%TUNE'('C')'')');
               SPACE(40);
               II:=RND(14);
               REG:=RND(HIGH-LOW-12)+LOW;
               'FOR'P:=1'STEP'1'UNTIL'SECS'DO'
               'BEGIN'
                  'IF'P/2=P'/'2'THEN'NEWLINE(1)'ELSE'
                     WRITETEXT('('%%%%%')');
                  'FOR'Q:=1'STEP'1'UNTIL'8'DO'
                  'BEGIN'
                     A:=RNDEC(4.267);
                     SUM:=0;
                     'FOR'JJ:=1'STEP'1'UNTIL'14'DO'
                     'BEGIN'
                        SUM:=SUM+STM2[II,JJ];
                        'IF'SUM>A'THEN'
                        'BEGIN'
                           PRINST(JJ,REG);
                           II:=JJ;
                           'GOTO'NUQ2;
                        'END';
                     'END';
NUQ2:             'END';
               'END';
               'GOTO'NEXTK;
            'END';
ST3:        'BEGIN'
               WRITETEXT('('3-ORDER%STOCHASTIC%TUNE'('C')'')');
               SPACE(40);
               II:=RND(14);JJ:=RND(14);
```

```
                       REG:=RND(HIGH-LOW-12)+LOW;
                       'FOR'P:=1'STEP'1'UNTIL'SECS'DO'
                       'BEGIN'

                          'IF'P/2=P'/'2'THEN'NEWLINE(1)'ELSE'
                            WRITETEXT('('%%%%%')');
                          'FOR'Q:=1'STEP'1'UNTIL'8'DO'
                          'BEGIN'
                            A:=RNDEC(9,5);
                            SUM:=0;
                            'FOR'KK:=1'STEP'1'UNTIL'14'DO'
                            'BEGIN'
                               SUM:=SUM+STM3[II,JJ,KK];
                               'IF'SUM>A'THEN'
                               'BEGIN'
                                  PRINST(KK,REG);
                                  II:=JJ;
                                  JJ:=KK;
                                  'GOTO'NUQ3;
                               'END';
                            'END';
NUQ3:                     'END';
                       'END';
                       'GOTO'NEXTK;
                   'END';
COLR:              'BEGIN'
                     'INTEGER'SHAPE,SIZE;
                      WRITETEXT('('%COLOURED%SECTION'('C')'')');
                     'FOR'P:=1'STEP'1'UNTIL'RND(SECS)'DO'
                    'BEGIN'
                          SHAPE:=RND(4);
                          SIZE:=RND(20);
                          'IF'P/2=P'/'2'THEN'NEWLINE(1)'ELSE'
                             WRITETEXT('('%%')');
                          'IF'SHAPE=1'THEN'WRITETEXT('('%%A%CIRCLE%')')
                  'ELSE''IF'SHAPE=2'THEN'WRITETEXT('('%%A%SQUARE%')')
                  'ELSE''IF'SHAPE=3'THEN'WRITETEXT('('A%TRIANGLE%')')
                  'ELSE''IF'SHAPE=4'THEN'WRITETEXT('('%A%SPLODGE%')');
                          WRITETEXT('('OF%SIZE')');
                         PRINT(SIZE,2,0);
                          WRITETEXT('('AT')');
                          PRINT(RND(10*SECS-SIZE)+SIZE'/'2,3,0);
                          WRITETEXT('(',')');
                          PRINT(RND(50-2*SIZE)+SIZE,3,0);
                          WRITETEXT('('AND%OF%COLOUR')');
                          PRINT(RND(20),2,0);
                       'END';
                       'GOTO'NEXTK;
                   'END';
NEXTK:             'IF'RND(3)=1'THEN'SECS:=RND(20-SECS)
                               'ELSE'SECS:=RND(20);
                   I:=J;
              'END';
      'END';
             'FOR'K:=1'STEP'1'UNTIL'NOMOV'DO'
             'BEGIN'
                'INTEGER'DYN,LS;
                NEWLINE(2);
                WRITETEXT('('%%%***%%%DYNAMIC%SYSTEM')');
                PRINT(K,1,0);
                DYN:=RND(8);
                LS:=0;
                WRITETEXT('('%%%%START%AT')');
                PRINT(DYN,1,0);

                'FOR'P:=2'STEP'1'UNTIL'TOTSECS'DO'
             'IF'RND(2)=1'THEN'
             'BEGIN'
                 'IF'LS/4=LS'/'4'THEN'NEWLINE(1);
                 DYN:=DYN+RND(3)-2;
                 'IF'DYN<1'THEN'DYN:=1'ELSE'
                 'IF'DYN>8'THEN'DYN:=8;
                 WRITETEXT('('%%AT')');
                 PRINT(P,4,0);
                 WRITETEXT('('%DYNAMIC%IS')');
                 PRINT(DYN,2,0);
                 LS:=LS+1;
          'END';
      'END';
'END';
```

1b. *light*

The same program was used for *symposium* .

An example of the type of output produced follows the program.

```
'BEGIN'
   'REAL''ARRAY'MACROS[1:12,1:12],STM1[1:14],STM2[1:14,1:14],
             STM3[1:14,1:14,1:14];
   'INTEGER'I,J,K,P,Q,SECS,MOVT,NOMOV,NOTIMES,II,JJ,KK,REG,
            TOTSECS,PART,DENS,HIGH,LOW,INSTS,PLACE;
   'INTEGER''ARRAY'DEN[1:600],FORM[1:10];
   'REAL'SUM,X,A;
   'REAL''PROCEDURE'RNDEC(B);
      'REAL'B;
      'BEGIN'X:=9.2351*X+B;
         RNDEC:=X:=100*X-ENTIER(100*X);
      'END';
   'INTEGER''PROCEDURE'RND(N);
      'INTEGER'N;
      'BEGIN'RND:=ENTIER(RNDEC(3.572)*N)+1'END';
   'PROCEDURE'PRINST(F,REGIST);
      'INTEGER'F,REGIST;
      'BEGIN'
        'IF'F=13'THEN'WRITETEXT('('%**%%')')
  'ELSE''IF'F=14'THEN'WRITETEXT('('%--%%')')
  'ELSE'PRINT(F+REGIST,2,0);
      'END';
  'SWITCH'TYPE:=LONG,REP,PCH1,PCH2,PCH3,RUN1,RUN2,REST,ST1,ST2,ST3,COLR
   X:=READ;
        'FOR'I:=1'STEP'1'UNTIL'12'DO'
        'BEGIN'
           SUM:=0;
           'FOR'J:=1'STEP'1'UNTIL'12'DO'
           'IF'RND(3)>1'THEN'MACROS[I,J]:=0
           'ELSE''BEGIN'MACROS[I,J]:=RNDEC(3.56);
                  SUM:=SUM+MACROS[I,J];
                'END';
           'FOR'J:=1'STEP'1'UNTIL'12'DO'
           MACROS[I,J]:=MACROS[I,J]/SUM;
        'END';
        'BEGIN'
           'REAL'SUM1,SUM2,SUM3;
           SUM1:=0;
           'FOR'II:=1'STEP'1'UNTIL'14'DO'
           'BEGIN'
             'IF'RND(3)>1'THEN'
             'BEGIN'
                'IF'II>12'THEN''GOTO'P1;
                STM1[II]:=0;
             'END''ELSE'
P1:          'BEGIN'
                STM1[II]:=RNDEC(4.89);
               SUM1:=SUM1+STM1[II];
            'END';
           SUM2:=0;
```

```
               'FOR'JJ:=1'STEP'1'UNTIL'14'DO'
               'BEGIN'
                  'IF'RND(5)>1'THEN'
                  'BEGIN'
                     'IF'JJ>12'THEN''GOTO'P2;
                     STM2[II,JJ]:=0;
                  'END''ELSE'
P2:               'BEGIN'
                     STM2[II,JJ]:=RNDEC(2.47);
                     SUM2:=SUM2+STM2[II,JJ];
                  'END';
                  SUM3:=0;
                  'FOR'KK:=1'STEP'1'UNTIL'14'DO'
                  'BEGIN'
                     'IF'RND(6)>1'THEN'
                     'BEGIN'
                        'IF'KK>12'THEN''GOTO'P3;
                        STM3[II,JJ,KK]:=0;
                     'END''ELSE'
P3:                  'BEGIN'
                        STM3[II,JJ,KK]:=RNDEC(5.89);
                        SUM3:=SUM3+STM3[II,JJ,KK];
                     'END';
                  'END';
                  'FOR'KK:=1'STEP'1'UNTIL'14'DO'
                  STM3[II,JJ,KK]:=STM3[II,JJ,KK]/SUM3;
               'END';
               'FOR'JJ:=1'STEP'1'UNTIL'14'DO'
               STM2[II,JJ]:=STM2[II,JJ]/SUM2;
            'END';
            'FOR'II:=1'STEP'1'UNTIL'14'DO'
            STM1[II]:=STM1[II]/SUM1;
         'END';
         WRITETEXT('('LIGHT%%%%MOVEMENT')');
         X:=READ;
         PRINT(READ,1,0);
         NEWLINE(2);
         WRITETEXT('('FORM%%%')');
         'FOR'P:=1'STEP'1'UNTIL'10'DO'
         'BEGIN'
            FORM[P]:=READ;
            PRINT(FORM[P],2,0);
         'END';
         INSTS:=READ;
         TOTSECS:=RND(140)+60;
         NOMOV:=RND(5);
         DEN[1]:=RND(ENTIER(INSTS/2));
         'FOR'P:=2'STEP'1'UNTIL'TOTSECS'DO'
         'BEGIN'
            'IF'RND(3)=1'THEN'DEN[P]:=DEN[P-1]+RND(5)-3
            'ELSE'DEN[P]:=DEN[P-1];
            'IF'DEN[P]<1'THEN'DEN[P]:=1'ELSE'
            'IF'DEN[P]>ENTIER(INSTS/2)'THEN'DEN[P]:=ENTIER(INSTS/2);
         'END';
      'FOR'MOVT:=1'STEP'1'UNTIL'INSTS'DO'
      'BEGIN'NEWLINE(3);
         WRITETEXT('('PART')');
         PRINT(MOVT,1,0);
         COPYTEXT('(':')');
         NEWLINE(1);
```

```
            WRITETEXT('('USE%DYNAMIC%SYSTEM')');
           PRINT(RND(NOMOV),1,0);
            LOW:=READ;
            HIGH:=READ;
            SECS:=RND(12)+1;
            PLACE:=1;
            I:=RND(12);
            'FOR'K:=1'STEP'SECS'UNTIL'TOTSECS'DO'
            'BEGIN'
               SUM:=0;
               A:=RNDEC(8.557));
               'FOR'J:=1'STEP'1'UNTIL'12'DO'
               'BEGIN'
                  SUM:=SUM+MACROS[I,J];
                  'IF'SUM>A'THEN''GOTO'WORK;
               'END';
WORK:          NEWLINE(1);
               WRITETEXT('('******%SECTION')');
               PRINT(PLACE,3,0);
               PLACE:=PLACE+SECS;
               WRITETEXT('('%%TYPE')');PRINT(J,3,0);
               WRITETEXT('('%..LASTS%FOR')');PRINT(SECS,3,0);
               WRITETEXT('('SECS ******')');
               NEWLINE(1);
               'IF'RND(2)=1'THEN'
                  'GOTO'TYPE[FORM[(10*K-1)'/'TOTSECS+1]]
               'ELSE''IF'RND(DEN[K])=1'THEN''GOTO'REST
               'ELSE''GOTO'TYPE[J];
LONG:          'BEGIN'
                  WRITETEXT('('%LONG%:%PITCH')');
                  PRINT(RND(HIGH-LOW)+LOW,2,0);
                  'GOTO'NEXTK;
               'END';
REP:           'BEGIN'
                  WRITETEXT('('%REPEATED%NOTE%:%PITCH')');
                  PRINT(RND(HIGH-LOW)+LOW,2,0);
                  'GOTO'NEXTK;
               'END';
PCH1:
PCH2:          'BEGIN'
                  WRITETEXT('('%SOUNDS%AT%THE%FOLLOWING%TIME-PITCH
                               %CO-ORDINATES:')');
                 'FOR'P:=1'STEP'1'UNTIL'RND(5*SECS)'DO'
                 'BEGIN'
                 'IF'P/8=P'/'8'THEN'
                 NEWLINE(1);
                 PRINT(RND(10*SECS),3,0);
                 WRITETEXT('(',')');
                 PRINT(RND(50),2,0);
                'END';
                 'GOTO'NEXTK;
               'END';
PCH3:          'BEGIN'
                  WRITETEXT('('%SOUNDS%AS%FOLLOWS')');
                  NEWLINE(1);
                  SPACE(40);
                  'FOR'P:=1'STEP'1'UNTIL'SECS'DO'
                  'BEGIN'
                     'IF'P/2=P'/'2'THEN'NEWLINE(1)
                     'ELSE'WRITETEXT('('%%%%%')');
                'FOR'Q:=1'STEP'1'UNTIL'8'DO'
                'BEGIN''INTEGER'R;
                   R:=RND(3);
                    'IF'R=1'THEN'
             PRINT(RND(HIGH-LOW)+LOW,2,0)
             'ELSE''IF'R=2'THEN'WRITETEXT('('%**%%')')
             'ELSE''IF'R=3'THEN'WRITETEXT('('%--%%')');
                 'END'Q;
             'END'P;
             'GOTO'NEXTK;
          'END'BLOCK;
RUN1:     'BEGIN'
            'INTEGER'TS;TS:=0;
             WRITETEXT('('DESCENDING%RUNS%AS%FOLLOWS;')');
             'FOR'P:=1'STEP'1'UNTIL'8*SECS'DO'
             'BEGIN'
                'IF'TS/4=TS'/'4'THEN'NEWLINE(1);
                'IF'RND(2)=1'THEN'
                'BEGIN'
                   Q:=RND(12);
                   WRITETEXT('('%%%%%%FROM')');
                   PRINT(RND(HIGH-LOW-12)+LOW+12,2,0);
                   WRITETEXT('('%FOR')');
                   PRINT(Q,2,0);
                'END'
                'ELSE'
                'BEGIN'
                   Q:=RND(30);
                   WRITETEXT('('%%REST%FOR')');
                   PRINT(Q'/'8,2,0);
                   WRITETEXT('('%SECS')');
                   PRINT(Q-(Q'/'8)*8,2,0);
                'END';
                P:=P+Q;
                TS:=TS+1;
             'END';
             'GOTO'NEXTK;
          'END';
RUN2:     'BEGIN'
             'INTEGER'TS;TS:=0;
             WRITETEXT('('%RUNS%AS%FOLLOWS:')');
            'FOR'P:=1'STEP'1'UNTIL'8*SECS'DO'
             'BEGIN'
                'IF'TS/4=TS'/'4'THEN'NEWLINE(1);
                'IF'RND(2)=1'THEN'
                'BEGIN'
                   Q:=RND(12);
                   WRITETEXT('('%%%%%%FROM')');
                   PRINT(RND(HIGH-LOW-12)+LOW,2,0);
                   WRITETEXT('('%FOR')');
                   PRINT(Q,2,0);
                'END'
                'ELSE'
                'BEGIN'
                   Q:=RND(30);
                   WRITETEXT('('%%REST%FOR')');
                   PRINT(Q'/'8,2,0);
                   WRITETEXT('('%SECS')');
                   PRINT(Q-(Q'/'8)*8,2,0);
                'END';
```

```
                  P:=P+Q;
                  TS:=TS+1;
               'END';
               'GOTO'NEXTK;
            'END';
REST:       'BEGIN'
               WRITETEXT('('%%%REST')');
               'GOTO'NEXTK;
            'END';
ST1:        'BEGIN'
               WRITETEXT('('1-ORDER%STOCHASTIC%TUNE'('C')'')');
               SPACE(40);
               REG:=RND(HIGH-LOW-12)+LOW;
               'FOR'P:=1'STEP'1'UNTIL'SECS'DO'
               'BEGIN'
                  'IF'P/2=P'/'2'THEN'NEWLINE(1)'ELSE'
                     WRITETEXT('('%%%%%')');
                  'FOR'Q:=1'STEP'1'UNTIL'8'DO'
                  'BEGIN'
                     SUM:=0;
                     A:=RNDEC(2.9);
                     'FOR'II:=1'STEP'1'UNTIL'14'DO'
                     'BEGIN'
                        SUM:=SUM+STM1[II];
                        'IF'SUM>A'THEN'
                        'BEGIN'
                           PRINST(II,REG);
                           'GOTO'NUQ1;
                        'END';
                     'END';
NUQ1:             'END';
               'END';
               'GOTO'NEXTK;
            'END';
ST2:        'BEGIN'
               WRITETEXT('('2-ORDER%STOCHASTIC%TUNE'('C')'')');
               SPACE(40);
               II:=RND(14);
               REG:=RND(HIGH-LOW-12)+LOW;
               'FOR'P:=1'STEP'1'UNTIL'SECS'DO'
               'BEGIN'
                  'IF'P/2=P'/'2'THEN'NEWLINE(1)'ELSE'
                     WRITETEXT('('%%%%%')');
                  'FOR'Q:=1'STEP'1'UNTIL'8'DO'
                  'BEGIN'
                     A:=RNDEC(4.267);
                     SUM:=0;
                     'FOR'JJ:=1'STEP'1'UNTIL'14'DO'
                     'BEGIN'
                        SUM:=SUM+STM2[II,JJ];
                        'IF'SUM>A'THEN'
                        'BEGIN'
                           PRINST(JJ,REG);
                           II:=JJ;
                           'GOTO'NUQ2;
                        'END';
                     'END';
NUQ2:             'END';
               'END';
               'GOTO'NEXTK;
```

```
             'END';
ST3:         'BEGIN'
                WRITETEXT('('3-ORDER%STOCHASTIC%TUNE'('C')')');
                SPACE(40);
                II:=RND(14);JJ:=RND(14);
                REG:=RND(HIGH-LOW-12)+LOW;
                'FOR'P:=1'STEP'1'UNTIL'SECS'DO'
                'BEGIN'
                   'IF'P/2=P'/'2'THEN'NEWLINE(1)'ELSE'
                      WRITETEXT('('%%%%%')');
                   'FOR'Q:=1'STEP'1'UNTIL'8'DO'
                   'BEGIN'
                      A:=RNDEC(9.5);
                      SUM:=0;
                      'FOR'KK:=1'STEP'1'UNTIL'14'DO'
                      'BEGIN'
                         SUM:=SUM+STM3[II,JJ,KK];
                         'IF'SUM>A'THEN'
                         'BEGIN'
                            PRINST(KK,REG);
                            II:=JJ;
                            JJ:=KK;
                            'GOTO'NUQ3;
                         'END';
                      'END';
NUQ3:              'END';
                'END';
                'GOTO'NEXTK;
             'END';
COLR:        'BEGIN'
                WRITETEXT('('%DOUBLE%THE%PART')');
                PRINT(RND(INSTS),2,0);
                WRITETEXT('('ABOVE')');
                'GOTO'NEXTK;
             'END';
NEXTK:       'IF'RND(3)=1'THEN'SECS:=RND(12-SECS)+1
                              'ELSE'SECS:=RND(12)+1;
             I:=J;
          'END';
    'END';
          NEWLINE(5);
          WRITETEXT('('BEAT')');
          NEWLINE(2);
          'FOR'K:=1'STEP'RND(70)'UNTIL'TOTSECS'DO'
          'BEGIN'
             WRITETEXT('('%%%AT')');
             PRINT(K,3,0);
             PRINT(RND(2)+2,4,0);
             PRINT(RND(2)*4,1,0);
             WRITETEXT('('%%%TEMPO%')');
             PRINT(RND(120)+60,3,0);
          'END';
          'FOR'K:=1'STEP'1'UNTIL'NOMOV'DO'
          'BEGIN'
             'INTEGER'DYN,LS;
             NEWLINE(2);
             WRITETEXT('('%%%***%%%DYNAMIC%SYSTEM')');
             PRINT(K,1,0);
             DYN:=RND(8);
             LS:=0;
```

```
                        WRITETEXT('('%%%%START%AT')');
                        PRINT(DYN,1,0);
                        'FOR'P:=2'STEP'1'UNTIL'TOTSECS'DO'
                        'IF'RND(2)=1'THEN'
                        'BEGIN'
                           'IF'LS/4=LS'/'4'THEN'NEWLINE(1);
                           DYN:=DYN+RND(3)-2;
                           'IF'DYN<1'THEN'DYN:=1'ELSE'
                           'IF'DYN>8'THEN'DYN:=8;
                           WRITETEXT('('%%AT')');
                           PRINT(P,4,0);
                           WRITETEXT('('%DYNAMIC%IS')');
                           PRINT(DYN,2,0);
                           LS:=LS+1;
                     'END';
                'END';
       'END';
```

```
   ••    --    --    ••    ••    ••    66    ••        54    67    --    39    39    40    39    ••
•••••• SECTION  57     TYPE  10     .,LASTS FOR   12  SECS••••••
2-ORDER STOCHASTIC TUNE

   --    66    59    58    --    59    ••    59        ••    59    64    63    66    58    --    58
   65    --    ••    59    ••    59    --    66        --    58    ••    65    65    --    59    ••
   64    63    67    63    --    59    --    59        59    68    --    59    ••    63    66    59
   65    65    --    --    59    --    60    63        58    ••    --    66    ••    ••    --    ••
   59    68    64    63    67    --    66    59        67    61    ••    59    68    64    63    --
   --    ••    59    58    ••    59    --    60        58    ••    --    58    --    --    59    --
•••••• SECTION  59     TYPE   5     .,LASTS FOR    2  SECS••••••
 DOUBLE THE PART  3  ABOVE
•••••• SECTION  65     TYPE  11     .,LASTS FOR    6  SECS••••••
     REST
•••••• SECTION  67     TYPE   5     .,LASTS FOR    2  SECS••••••
 SOUNDS AS FOLLOWS

   --    45    72    ••    66    --    ••    59        --    44    ••    ••    --    57    --    ••
•••••• SECTION  71     TYPE  10     .,LASTS FOR    4  SECS••••••
 SOUNDS AS FOLLOWS

   63    --    50    --    48    60    44    61        --    --    ••    ••    ••    47    --    --
   --    48    ••    --    --    ••    --    --        --    --    --    54    --    --    --    --
•••••• SECTION  82     TYPE   5     .,LASTS FOR   11  SECS••••••
 SOUNDS AS FOLLOWS

   ••    ••    70    56    ••    ••    59    57        72    ••    --    --    --    --    --    ••
   ••    ••    ••    ••    --    --    --    50        --    --    ••    ••    41    52    --    --
                                                       --    57    40    --    66    --    65    ••



   ••    ••    55    --    ••    59    65    57        ••    --    --    ••    --    48    ••    42
   --    ••    --    62    ••    --    --    --        --    41    --    --    ••    51    ••    ••
   ••    --    ••    --    --    ••    58    ••        70    ••    --    --    --    ••    71    60
•••••• SECTION  89     TYPE   6     .,LASTS FOR    7  SECS••••••
DESCENDING RUNS AS FOLLOWS:
  REST FOR  3   SECS  3               FROM 63   FOR   6    REST FOR  3   SECS  2
•••••• SECTION 101     TYPE  12     .,LASTS FOR   12  SECS••••••
     REST
•••••• SECTION 112     TYPE   3     .,LASTS FOR   11  SECS••••••
     REST
•••••• SECTION 117     TYPE  11     .,LASTS FOR    5  SECS••••••
3-ORDER STOCHASTIC TUNE

   ••    --    66    --    ••    ••    ••    64        64    63    ••    --    ••    ••    --    66
   --    ••    ••    64    ••    55    57    64        ••    56    64    ••    55    61    --    66
                                                       64    --,   --    66    --    --    65    --
•••••• SECTION 122     TYPE   5     .,LASTS FOR    5  SECS••••••
     REST
•••••• SECTION 128     TYPE   6     .,LASTS FOR    6  SECS••••••
 SOUNDS AS FOLLOWS

   72    ••    --    --    61    39    --    --        ••    48    71    --    ••    51    69    --
   38    --    ••    64    --    ••    62    --        --    47    --    --    ••    ••    40    ••
   ••    ••    62    56    ••    --    ••    ••        --    ••    ••    --    --    --    48    --
•••••• SECTION 140     TYPE   3     .,LASTS FOR   12  SECS••••••
 SOUNDS AS FOLLOWS

                                                       --    --    ••    ••    ••    40    39    38
   ••    63    ••    --    --    56    ••    --        ••    69    ••    ••    --    ••    ••    --
   ••    38    71    --    ••    --    ••    --        ••    ••    40    71    44    58    59    --
   --    --    ••    --    ••    --    ••    --        --    ••    --    38    ••    --    ••    --
   --    ••    64    ••    --    --    57    --        --    --    --    40    56    ••    -     --
   59    71    57    73    73    68    51    44        ••    --    --    54    ••    52    ••    60
   60    67    ••    --    44    [illegible]
```

```
•••••• SECTION 145    TYPE  7   .,LASTS FOR   5  SECS••••••
    REST
•••••• SECTION 152    TYPE  3   .,LASTS FOR   7  SECS••••••
 DOUBLE THE PART  7  ABOVE
•••••• SECTION 155    TYPE 11   .,LASTS FOR   3  SECS••••••
 DOUBLE THE PART  4  ABOVE
•••••• SECTION 159    TYPE  7   .,LASTS FOR   4  SECS••••••
 RUNS AS FOLLOWS:
  REST FOR  1   SECS  7        FROM 59   FOR  1    REST FOR  0   SECS  2        FROM 57   FOR 12

PART 2
CLARINET
USE DYNAMIC SYSTEM 1
•••••• SECTION   1    TYPE  8   .,LASTS FOR   6  SECS••••••
 SOUNDS AS FOLLOWS
                                               --   --   --   65   --   ••   ••   --
 --   ••   --   36   --   --   45   ••         ••   37   ••   --   ••   59   49   64
 70   --   ••   34   --   --   ••   47         32   --   --   ••   57   ••   --   ••
 62   ••   ••   56   --   ••   61   32
•••••• SECTION   3    TYPE  5   .,LASTS FOR   2  SECS••••••
 SOUNDS AS FOLLOWS
                                               --   ••   38   68   --   --   36   48
 --   ••   56   --   ••   54   ••   --
•••••• SECTION   7    TYPE 11   .,LASTS FOR   4  SECS••••••
 SOUNDS AS FOLLOWS
                                               --   ••   ••   54   ••   71   ••   ••
 ••   ••   --   --   --   ••   ••   33         41   69   33   39   --   ••   ••   --



 --   --   ••   ••   --   ••   ••   ••
•••••• SECTION  12    TYPE  5   .,LASTS FOR   5  SECS••••••
 SOUNDS AS FOLLOWS
                                               48   --   63   --   ••   ••   31   ••
 ••   --   38   ••   41   ••   --   ••         --   ••   ••   --   --   ••   53   --
 ••   ••   45   --   71   --   --   --         72   --   31   ••   70   52   --   68
•••••• SECTION  19    TYPE  9   .,LASTS FOR   7  SECS••••••
1-ORDER STOCHASTIC TUNE
                                               --   59   58   62   64   59   58   59
 --   64   --   64   58   64   58   --         62   59   65   63   63   64   --   --
 64   64   59   63   59   59   58   --         58   59   62   64   64   65   --   65
 65   58   59   --   62   62   59   63         63   --   --   63   58   65   58   59
•••••• SECTION  22    TYPE  3   .,LASTS FOR   3  SECS••••••
 SOUNDS AS FOLLOWS
                                               68   36 , --   45   66   59   70   --
 ••   --   30   68   ••   --   --   ••         ••   55   --   ••   --   ••   58   49
•••••• SECTION  34    TYPE  7   .,LASTS FOR  12  SECS••••••
 DOUBLE THE PART  7  ABOVE
•••••• SECTION  36    TYPE  8   .,LASTS FOR   2  SECS••••••
 DOUBLE THE PART  5  ABOVE
•••••• SECTION  46    TYPE 10   .,LASTS FOR  10  SECS••••••
 DOUBLE THE PART  2  ABOVE
•••••• SECTION  56    TYPE  8   .,LASTS FOR  10  SECS••••••
    REST
•••••• SECTION. 65    TYPE  8   .,LASTS FOR   9  SECS••••••
    REST
•••••• SECTION  73    TYPE  4   .,LASTS FOR   8  SECS••••••
 SOUNDS AS FOLLOWS
                                               ••   --   ••   48   ••   71   ••   --
 66   35   --   50   --   --   --   45         --   --   31   ••   54   35   ••   ••
```

<u>1c. *text*</u>

```
'BEGIN'
   'REAL'X,Y,RN,X1;
   'INTEGER'I,J,K,CHAR,LINES,LENGTH,STANZA;
   'INTEGER''ARRAY'WORDS[1:11],LEX[1:11,1:25,1:12];
   'REAL''ARRAY'MN[1:4],N[1:6];
   'REAL''PROCEDURE'RNDECB(A);'REAL'A;
      'BEGIN'Y:=2.4179*Y+A;
         RNDECB:=Y:=100*Y-ENTIER(100*Y);
      'END';
   'REAL''PROCEDURE'RNDEC(B);'REAL'B;
      'BEGIN'X:=3.4561*X+B;
         'IF'RNDECB(6.237)<RN'THEN'RNDEC:=X1
         'ELSE'RNDEC:=X:=100*X-ENTIER(100*X);
      'END';
   'INTEGER''PROCEDURE'RND(M);'INTEGER'M;
      'BEGIN'
         RND:=ENTIER(RNDEC(4.829)*M)+1;
      'END';
   'PROCEDURE'CHOOSE(I2);'INTEGER'I2;
      'BEGIN'
         'INTEGER'J2,K2;
         'IF'I2<5'THEN'
         'BEGIN'
            'IF'RNDEC(2.11341)>MN[I2]'THEN''GOTO'FINA;
         'END';
         J2:=RND(WORDS[I2]);
         'FOR'K2:=1'STEP'1'UNTIL'12'DO'
         'BEGIN'
            'IF'LEX[I2,J2,K2]=CODE('('/')')'THEN''GOTO'FINA
            'ELSE'PRINTCH(LEX[I2,J2,K2]);
      'END';
FINA: 'END';
  'PROCEDURE'SS;
    'BEGIN'WRITETEXT('('S')')'END';
  LENGTH:=READ;STANZA:=READ;
  X1:=Y:=X:=READ;RN:=READ;
  'FOR'I:=1'STEP'1'UNTIL'4'DO'MN[I]:=READ;
  'FOR'I:=1'STEP'1'UNTIL'6'DO'N[I]:=READ;
  'FOR'I:=1'STEP'1'UNTIL'11'DO'
  'BEGIN'
    WORDS[I]:=READ;
    'FOR'J:=1'STEP'1'UNTIL'WORDS[I]'DO'
    'BEGIN'
      'FOR'K:=1'STEP'1'UNTIL'12'DO'
      'BEGIN'
ON:       CHAR:=READCH;
        'IF'CHAR=CODE('('EL')')'THEN''GOTO'ON;
        LEX[I,J,K]:=CHAR;
        'IF'CHAR=CODE('('/')')'THEN''GOTO'NEXTJ;
      'END'K;

NEXTJ:'END'J;
  'END'I;
  'FOR'J:=1'STEP'1'UNTIL'6'DO'
  'FOR'N[J]:=0'STEP'0.25'UNTIL'1'DO'
  'BEGIN'
  'FOR'I:=1'STEP'1'UNTIL'4'DO'
  'FOR'MN[I]:=0'STEP'0.25'UNTIL'1'DO'
  NEWLINE(3);
  'BEGIN'
  WRITETEXT('('N')');
  PRINT(J,1,0);
  WRITETEXT('('=')');
  PRINT(N[J],1,2);
  WRITETEXT('('%%%%MN')');
  PRINT(I,1,0);
  WRITETEXT('('=')');
  PRINT(MN[I],1,2);
  NEWLINE(2);
  'FOR'LINES:=1'STEP'1'UNTIL'RND(5)+5'DO'
  'BEGIN'
    CHOOSE(2);
    'IF'RNDEC(8.459)>N[1]
    'THEN'
      'BEGIN'CHOOSE(3);CHOOSE(1);
        'IF'RNDEC(7.623)>N[2]
        'THEN'
          'BEGIN'CHOOSE(6);
           'IF'RNDEC(4.251)>N[3]
            'THEN'CHOOSE(10)
            'ELSE''BEGIN'CHOOSE(11);SS;'END';
          'END'
        'ELSE'
          'BEGIN'CHOOSE(6);SS;
            'IF'RNDEC(3.1899)>N[4]
            'THEN'CHOOSE(10)'ELSE'CHOOSE(11)
          'END';
      'END'
    'ELSE'
      'BEGIN'CHOOSE(9);
        'IF'RNDEC(5.2771)>N[5]
        'THEN'
```

```
               'BEGIN'CHOOSE(7);CHOOSE(11)'END'
            'ELSE'CHOOSE(10);
         'END';
      CHOOSE(2);CHOOSE(4);
      'IF'RNDEC(1.2939)>N[6]
      'THEN'CHOOSE(8)
      'ELSE'
         'BEGIN'CHOOSE(3);CHOOSE(1);
                CHOOSE(6);SS;
         'END';
      'IF'LINES=LENGTH'THEN''GOTO'FIN;
      NEWLINE(1);
      'IF'RND(STANZA)=1'THEN'NEWLINE(1);
      CHOOSE(5);
   'END';
FIN:WRITETEXT('('.')');
   'END';
    PAPERTHROW;
   'END';

'END';
```

1d. *pianocomp*

An example of the type of output produced follows the program.

```
'BEGIN'
   'COMMENT'PROGRAM FOR PIANO COMPOSITION
            DATA IN FORM: RANDOM NUMBER STARTER (0.0000 TO 0.9999)
                          MAXIMUM LENGTH IN SECONDS
                           NUMBER OF LAYERS (2 TO 4)
                           13 NUMBERS DEFINING FORM BIAS
                           (E.G. 1 1 1 1 1 2 2 2 3 3 3 4 5);
   'REAL'X,SUM,A;
   'INTEGER'LAYERS,LENGTH,UNITS,I,J,F,LOWER,UPPER,NOBS,NOTES,RNDCOUNT
            DYN;
   'REAL''ARRAY'STMF[1:13,1:13],STM1[1:17],STM2[1:17,1:17],
                STM3[1:17,1:17,1:17];
   'INTEGER''ARRAY'FORM[1:4,1:600],TYPE[1:13],PARAM[1:13,1:5],
            DATATYPE[1:13],BEATS[1:600];
   'BOOLEAN''ARRAY'STCT[0:3];
   'SWITCH'SWTYPE:=SWCHOR,SWMELO,SWRUNS,SWTRIL,SWSILE;

   'REAL''PROCEDURE'RNDEC(B);
      'VALUE'B;
      'REAL'B;
      'BEGIN'
         X:=7.2937*X+B;
         RNDEC:=X:=100*X-ENTIER(100*X);
         RNDCOUNT:=RNDCOUNT+1;
      'END';
   'INTEGER''PROCEDURE'RND(N);
      'VALUE'N;
      'INTEGER'N;
      'BEGIN'
         RND:=ENTIER(RNDEC(3.1971)*N+1)
      'END';
   'PROCEDURE'BEATCOUNT;
      'BEGIN'
         NOBS:=0;
         'FOR'J:=LOWER'STEP'1'UNTIL'UPPER-1'DO'
         NOBS:=NOBS+BEATS[J];
      'END';

   'PROCEDURE'RHYTHM(LENGTH,SPEED);
      'VALUE'LENGTH,SPEED;
      'INTEGER'LENGTH,SPEED;
      'BEGIN'
         'INTEGER'K1,SUMM;
         SUMM:=NOTES:=0;
NEWK1:   K1:=RND(RND(2↑SPEED));
         SUMM:=SUMM+K1;
         NOTES:=NOTES+1;
         'IF'SUMM>LENGTH'THEN'
         'BEGIN'
            K1:=K1-(SUMM-LENGTH);
```

```
                'IF'K1=0'THEN'NOTES:=NOTES-1;
                PRINT(K1,1.0),
                'GOTO'LAST;
            'END'
            'ELSE'PRINT(K1,1,0);
            'GOTO'NEWK1;
    LAST:   NEWLINE(1);
        'END';

        'PROCEDURE'CHORDS(C1,C2,C3,C4,C5):
          'COMMENT' C1 - NUMBER OF NOTES IN CHORD
                    C2 - DEGREE OF REPETITION
                    C3 - SPEED OF REPETITION
                    C4 - SPACING
                    C5 - RHYTHMIC NATURE;
          'VALUE'C1,C2,C3,C4,C5;
          'INTEGER'C1,C2,C3,C4,C5;
          'BEGIN'
             'IF'C5=1'THEN'
             'BEGIN'
                BEATCOUNT;
                WRITETEXT('('RHYTHM:CONSTANT,%EACH%NOTE%ONE%')');
                'IF'C3=1'THEN'
                'BEGIN'
                   WRITETEXT('('HALF-BEAT')');
                   NOTES:=NOBS*2;
                'END'
                'ELSE''IF'C3=2'THEN'
                'BEGIN'
                  WRITETEXT('('BEAT')');
                  NOTES:=NOBS;
                'END'
                'ELSE''IF'C3=3'THEN'
                'BEGIN'
                   'IF'RND(2)=1'THEN'
                   'BEGIN
                      WRITETEXT('('DOUBLE-BEAT')');
                      NOTES:=NOBS'/'2+1;
                   'END''ELSE'
                   'BEGIN'
                      WRITETEXT('('BAR')');
                      NOTES:=UPPER-LOWER;
                   'END'
                'END'
             'END'
             'ELSE''IF'C5=2'THEN'
             'BEGIN'
                'INTEGER'NOTEMP;
                'PROCEDURE'OSTINATO;
                   'BEGIN'
                      WRITETEXT('('OSTINATO%AT')');
                      PRINT(J,3,0);
                      WRITETEXT('('WITH%RHYTHM:')');
                      RHYTHM(BEATS[J],C3);
                      NOTEMP:=NOTEMP+NOTES;
                   'END';
                NOTEMP:=0;
                J:=LOWER;
                OSTINATO;
                'FOR'J:=LOWER+1'STEP'1'UNTIL'UPPER-1'DO'
            'IF'BEATS[J]'NE'BEATS[J-1]'THEN'OSTINATO;
            NOTES:=NOTEMP;
         'END'
         'ELSE''IF'C5=3'THEN'
         'BEGIN'
            BEATCOUNT;
            WRITETEXT('('RHYTHM%PATTERN%(IN%HALF-BEATS):')');
            NEWLINE(1);
            RHYTHM(2*NOBS,C3);
         'END';
         'COMMENT' THE RHYTHM HAS BEEN DEALT WITH -
                   NOW FOR THE PITCHES;
         NEWLINE(1);
         WRITETEXT('('WITH%FOLLOWING%PITCHES:')');
         NEWLINE(1);
         'BEGIN'
            'INTEGER'Q;
            'INTEGER''ARRAY'PITCHES[1:5];
            PITCHES[1]:=RND(71);
            'FOR'Q:=1'STEP'1'UNTIL'NOTES'DO'
            'BEGIN'
               PITCHES[1]:=PITCHES[1]+RND(14-2*C1)-12;
               'IF'PITCHES[1]<1'THEN'PITCHES[1]:=RND(14-2*C1)
               'ELSE''IF'PITCHES[1]>71'THEN'PITCHES[1]:=71-RND(14-2*C1)
               'FOR'J:=1'STEP'1'UNTIL'C1'DO''BEGIN'
               'IF'J>1'THEN'
               PITCHES[J]:=PITCHES[J-1]+RND(13-C1+J-PITCHES[J-1]
                                                +PITCHES[1])-1;

               PRINT(PITCHES[J],4,0);'END';
               SPACE(3);
            'END'
         'END'
      'END'CHORDS PROCEDURE;

   'PROCEDURE'MELODIES(M1,M2);
      'VALUE'M1,M2;
      'INTEGER'M1,M2;
      'BEGIN'
         'INTEGER'Q2,Q3,Q4,RANGE,P1,P2,P3;
         'PROCEDURE'STOCK(ORDER);
            'INTEGER'ORDER;
            'BEGIN'
               'SWITCH'SWORDER:=SO1,SO2,SO3;
               'IF'ORDER=0'THEN'
               PRINT(RANGE+RND(17),2,0)'ELSE'
               'BEGIN'
                  SUM:=0;
                  A:=RNDEC(2,9147);
                  'FOR'P3:=1'STEP'1'UNTIL'17'DO'
                  'BEGIN'
                     'GOTO'SWORDER[ORDER];
SO1:                 SUM:=SUM+STM1[P3];'GOTO'SOSKIP;
SO2:                 SUM:=SUM+STM2[P2,P3];'GOTO'SOSKIP;
SO3:                 SUM:=SUM+STM3[P1,P2,P3];
SOSKIP:              'IF'SUM>A'THEN'
                     'BEGIN'
                        PRINT(RANGE+P3,2,0);
                        'GOTO'FINST;
                     'END';
                  'END';
```

```
     FINST            P1:=P2;P2:=P3;
                    'END'
                 'END'STOCK PROCEDURE;
              PRINT(M1,1,0);
              WRITETEXT('('-ORDER%%STOCHASTIC%MELODY.')');
              'IF'M2=2'THEN'
              WRITETEXT('('.DOUBLED%AT%THE%OCTAVE%ABOVE.')');
              NEWLINE(2);
              BEATCOUNT;
              WRITETEXT('('RHYTHM%PATTERN%(IN%HALF-BEATS):')');
              NEWLINE(2);
              RHYTHM(2*NOBS,RND(3));
              WRITETEXT('('PITCHES')');
              NEWLINE(2);
              RANGE:=RND(71-(17*(M2-1)));
              Q3:=NOTES-((NOTES'/'20)*20);
              P1:=RND(17);P2:=RND(17);
              'FOR'Q2:=1'STEP'1'UNTIL'NOTES/20'DO'
              'BEGIN'
                 'FOR'Q4:=1'STEP'1'UNTIL'20'DO'
                 STOCK(M1);
                 NEWLINE(1);
              'END';
              'FOR'Q4:=1'STEP'1'UNTIL'Q3'DO'
              STOCK(M1);
           'END'MELODIES PROCEDURE;
        'PROCEDURE'RUNS(R1,R2,R3,R4,R5);
           'VALUE'R1,R2,R3,R4,R5;
           'INTEGER'R1,R2,R3,R4,R5;
           'COMMENT'R1- SCALES/ARPEGGIOS
                    R2 - DIATONIC/CHROMATIC
                    R3 - NUMBER OF NOTES
                    R4 - RANGE
                    R5 - DIRECTION;
           'BEGIN'
              NEWLINE(1);
              WRITETEXT('('RUNS%-')');
              PRINT(RND(7)+1,1,0);
              WRITETEXT('('NOTES%TO%A%BEAT')');
              NEWLINE(1);
              'IF'R1=1'THEN'
              'BEGIN'
                 'IF'R2=1'THEN'
                 'BEGIN'
                    WRITETEXT('('DIATONIC%SCALE%IN%KEY')');
                    PRINT(RND(12)-1,2,0);
                    WRITETEXT('('%%(C=0)%%STARTING%ON%DEGREE')');
                    PRINT(RND(7),1,0);
                 'END''ELSE'
                 'BEGIN'
                    WRITETEXT('('CHROMATIC%SCALE%%BEGINNING%ON')');
                    PRINT(RND(12)-1,2,0);
                 'END'
              'END'
              'ELSE'
              'BEGIN'
                 'INTEGER''ARRAY'PITCHES[1:5];
                 WRITETEXT('('ARPEGGIO%%NOTES:')');
                 PITCHES[1]:=RND(13)-1;
                 'FOR'J:=1'STEP'1'UNTIL'R3'DO'
```

```
            'BEGIN'
               'IF'J>1'THEN'
               PITCHES[J]:=PITCHES[J-1]+RND(13-R3+J-PITCHES[J-1]
                                            +PITCHES[1])-1;
               PRINT(PITCHES[J],2,0);
               'END';
            'END';
            NEWLINE(2);
            WRITETEXT('('THROUGH')');
            PRINT(R4,1,0);
            WRITETEXT('('OCTAVES%%STARTING%IN%OCTAVE')');
            PRINT(RND(7),1,0);
            NEWLINE(1);
            WRITETEXT('('GOING')');
            'IF'R5'NE'2'THEN'WRITETEXT('('%UP')');
            'IF'R5=3'THEN'WRITETEXT('('%AND')');
            'IF'R5>1'THEN'WRITETEXT('('%DOWN')');
            NEWLINE(2);
         'END'RUNS PROCEDURE;

   'PROCEDURE'TRILLS(T1);
      'VALUE'T1;
      'INTEGER'T1;
      'BEGIN'
         NEWLINE(1);
         WRITETEXT('('A%')');
         'IF'T1=1'THEN'WRITETEXT('('SEMI')');
         WRITETEXT('('TONE%TRILL%ON')');
         PRINT(RND(82),2,0);
         NEWLINE(1);
      'END'TRILLS PROCEDURE;
   'PROCEDURE'SILENCE;
      'BEGIN'WRITETEXT('('SILENCE')')'END';

   X:=READ;
   PRINT(X*1000000,6,0);
   NEWLINE(1);
   RNDCOUNT:=0;
   LENGTH:=RND(READ);
   UNITS:=RND(LENGTH'/'100+4)+3;
   WRITETEXT('('PIANOCOMP%%%LENGTH:')');
   PRINT(LENGTH,3,0);
   NEWLINE(3);
  WRITETEXT('('%%%NUMBER%OF%UNITS:')');
   PRINT(UNITS,2,0);
   LAYERS:=READ;
   'COMMENT' STMF MATRIX (WITH PROBABILITIES RELATED TO NO OF UNITS
             CREATED;
   NEWLINE(2);
   WRITETEXT('('FORM%MATRIX')');
   NEWLINE(2);
   'FOR'I:=1'STEP'1'UNTIL'UNITS'DO'
   'BEGIN'
      SUM:=0.000001;
      'FOR'J:=1'STEP'1'UNTIL'UNITS'DO'
      'IF'RND(UNITS/4+1)>1
      'THEN'STMF[I,J]:=0'ELSE'
      'BEGIN'
         STMF[I,J]:=RNDEC(1,9773);
         SUM:=SUM+STMF[I,J];
```

```
                'END';
                'FOR'J:=1'STEP'1'UNTIL'UNITS'DO'
                'BEGIN'
                STMF[I,J]:=STMF[I,J]/SUM;
                PRINT(STMF[I,J],1,3);
                'END';
                NEWLINE(1);
            'END';
            'COMMENT'FORM CREATED USING STMF;
            'BEGIN'
                'INTEGER''PROCEDURE'NXFM(ELEMENT);
                   'VALUE'ELEMENT;
                   'INTEGER'ELEMENT;
                   'BEGIN'
                      'INTEGER'K;
                      SUM:=0;
                      A:=RNDEC(9.2133);
                      'FOR'K:=1'STEP'1'UNTIL'UNITS'DO'
                      'BEGIN'
                         SUM:=SUM+STMF[ELEMENT,K];
                         'IF'SUM>A'THEN''GOTO'WORK;
                      'END';
                      K:=UNITS;
   WORK:              NXFM:=K;
                   'END';
                NEWLINE(2);
                WRITETEXT('('FORM%STRUCTURE')');
                NEWLINE(2);
                WRITETEXT('('LAYER%1')');
                NEWLINE(2);
                FORM[1,1]:=RND(UNITS);
                PRINT(FORM[1,1],2,0);
                'FOR'J:=2'STEP'1'UNTIL'LENGTH'DO'
                'BEGIN'
                   'IF'RND(LENGTH/150+8)<7
                   'THEN'FORM[1,J]:=FORM[1,J-1]
                   'ELSE'FORM[1,J]:=NXFM(FORM[1,J-1]);
                   PRINT(FORM[1,J],2,0);
                'END';
                'FOR'I:=2'STEP'1'UNTIL'LAYERS'DO'
                'BEGIN'
                   NEWLINE(2);
                   WRITETEXT('('LAYER')');
                   PRINT(I,1,0);
                   NEWLINE(2);
                   'FOR'J:=1'STEP'1'UNTIL'LENGTH'DO'
                'BEGIN'
                      'IF'RND(LENGTH/150+8)<7
                      'THEN'FORM[I,J]:=FORM[1,J]
                      'ELSE'FORM[I,J]:=NXFM(FORM[1,J]);
                      PRINT(FORM[I,J],2,0);
                   'END'
                'END'
            'END';

            'COMMENT'UNIT PARAMETERS CHOSEN;
            'FOR'I:=1'STEP'1'UNTIL'13'DO'DATATYPE[I]:=READ;
            'FOR'I:=0'STEP'1'UNTIL'3'DO'STCT[I]:='FALSE';
            'FOR'I:=1'STEP'1'UNTIL'UNITS'DO'
            'BEGIN'
```

```
           'INTEGER'T;
           'FOR'J:=1'STEP'1'UNTIL'5'DO'PARAM[I,J]:=0;
           T:=TYPE[I]:=DATATYPE[RND(13)];
           'IF'T=1'THEN'
           'BEGIN'
               PARAM[I,1]:=RND(5);
               'FOR'J:=2'STEP'1'UNTIL'5'DO'
               PARAM[I,J]:=RND(3);
           'END'
           'ELSE''IF'T=2'THEN'
           'BEGIN'
               PARAM[I,1]:=RND(4)-1;
               STCT[PARAM[I,1]]:='TRUE';
               PARAM[I,2]:=RND(2)
           'END'
           'ELSE''IF'T=3'THEN'
           'BEGIN'
               PARAM[I,1]:=RND(2);
               PARAM[I,2]:=RND(2);
               PARAM[I,3]:=RND(5);
               PARAM[I,4]:=RND(RND(6));
               PARAM[I,5]:=RND(3);
           'END'
           'ELSE''IF'T=4'THEN'
           PARAM[I,1]:=RND(2);
        'END';
        NEWLINE(2);
        WRITETEXT('('TYPES%AND%UNIT%PARAMETERS')');
        NEWLINE(2);
        'FOR'I:=1'STEP'1'UNTIL'UNITS'DO'
        'BEGIN'
           PRINT(TYPE[I],1,0);
           WRITETEXT('(':')');
           'FOR'J:=1'STEP'1'UNTIL'5'DO'
           PRINT(PARAM[I,J],1,0);
           NEWLINE(1);
        'END';

        'COMMENT'CREATE STM1,2,3;
        'BEGIN'
           'REAL'SUM1,SUM2,SUM3;
           'INTEGER'II,JJ,KK;
           SUM1:=SUM2:=SUM3:=0.000001;
           'IF'STCT[1]'OR'STCT[2]'OR'STCT[3]'THEN'
           'BEGIN'
               'FOR'II:=1'STEP'1'UNTIL'17'DO'
               'BEGIN'
                   'IF'RND(4)>1
                   'THEN'STM1[II]:=0'ELSE'
                   'BEGIN'
                       STM1[II]:=RNDEC(3,617);
                       SUM1:=SUM1+STM1[II];
                   'END':
                   'IF'STCT[2]'OR'STCT[3]'THEN'
                   'BEGIN'
                       'FOR'JJ:=1'STEP'1'UNTIL'17'DO'
                       'BEGIN'
                           'IF'RND(7)>1
                           'THEN'STM2[II,JJ]:=0'ELSE'
                           'BEGIN'
```

```
                        STM2[II,JJ]:=RNDEC(2.473);
                        SUM2:=SUM2+STM2[II,JJ]
                     'END';
                     'IF'STCT[3]'THEN'
                     'BEGIN'
                        'FOR'KK:=1'STEP'1'UNTIL'17'DO'
                        'BEGIN'
                           'IF'RND(9)>1
                            'THEN'STM3[II,JJ,KK]:=0'ELSE'
                            'BEGIN'
                               STM3[II,JJ,KK]:=RNDEC(1.911);
                               SUM3:=SUM3+STM3[II,JJ,KK];
                            'END';
                        'END';
                        'FOR'KK:=1'STEP'1'UNTIL'17'DO'
                        STM3[II,JJ,KK]:=STM3[II,JJ,KK]/SUM3;
                     'END';
                  'END';
                  'FOR'JJ:=1'STEP'1'UNTIL'17'DO'
                  STM2[II,JJ]:=STM2[II,JJ]/SUM2;
               'END';
            'END';
            'FOR'II:=1'STEP'1'UNTIL'17'DO'
            STM1[II]:=STM1[II]/SUM1;
      'END';
   'END';

   'COMMENT'WORK OUT BEAT SECTIONS RELATED TO LENGTH;
   NEWLINE(2);
   WRITETEXT('('BEAT%IS')');
   PRINT(RND(2)*4,1,0);
   WRITETEXT('('AT%SPEED')');
   PRINT(RND(5),1,0);
   'BEGIN'
      'PROCEDURE'WRB(PL);
         'VALUE'PL;
         'INTEGER'PL;
         'BEGIN'
            BEATS[PL]:=RND(7)+1;
            WRITETEXT('('AT')');
            PRINT(PL,3,0);
            PRINT(BEATS[PL],1,0);
            WRITETEXT('('BEATS')');
            WRITETEXT('('%%%%%%')');
         'END';
         NEWLINE(1);
         WRB(1);
         'FOR'I:=2'STEP'1'UNTIL'LENGTH'DO'
         'IF'RND(10)=1'THEN'WRB(I)
         'ELSE'BEATS[I]:=BEATS[I-1];
      'END';

   'COMMENT' COMPOSITION EFFECTED FOR EACH LAYER;
   'FOR'I:=1'STEP'1'UNTIL'LAYERS'DO'
   'BEGIN'
      NEWLINE(2);
      'FOR'J:=1'STEP'1'UNTIL'100'DO'
      WRITETEXT('('*')');
      NEWLINE(2);
      WRITETEXT('('LAYER')');
```

```
        PRINT(I,1,0);
        NEWLINE(2);
        'COMMENT' WORK OUT DYNAMICS;
        WRITETEXT('('DYNAMICS')');
        NEWLINE(1);
        DYN:=RND(8);
        'FOR'J:=1'STEP'1'UNTIL'LENGTH'DO'
        'BEGIN'
            'IF'RND(5)>1'THEN''GOTO'GEN
            'ELSE''IF'RND(2)=1'THEN'
            'BEGIN'
                'IF'RND(2)=1'THEN'
                'BEGIN'
                    DYN:=DYN+1;
                    'IF'DYN=9'THEN'DYN:=8;
                'END''ELSE'
                'BEGIN'
                    DYN:=DYN-1;
                    'IF'DYN=0'THEN'DYN:=1;
                'END'
            'END'
            'ELSE'DYN:=RND(8);
            WRITETEXT('('%%%')');
            WRITETEXT('('AT')');
            PRINT(J,3,0);
            PRINT(DYN,1,0);
GEN:    'END';

        'COMMENT'OPERATE FORM MATRIX AND RUN;
        LOWER:=1;
        'FOR'UPPER:=LOWER+1'STEP'1'UNTIL'LENGTH'DO'
        'BEGIN'
            'IF'FORM[I,UPPER]'NE'FORM[I,UPPER-1]'THEN'
            'BEGIN'
                F:=FORM[I,LOWER];
                NEWLINE(1);
                WRITETEXT('('FROM')');
                PRINT(LOWER,3,0);
                WRITETEXT('('TO')');
                PRINT(UPPER-1,3,0);
                WRITETEXT('('TYPE')');
                PRINT(TYPE[F],2,0);
                NEWLINE(1);
                'GOTO'SWTYPE[TYPE[F]];
SWCHOR:         CHORDS(PARAM[F,1],PARAM[F,2],PARAM[F,3],
                        PARAM[F,4],PARAM[F,5]);
                'GOTO'NXSTP;
SWMELO:         MELODIES(PARAM[F,1],PARAM[F,2]);
                'GOTO'NXSTP;
SWRUNS:         RUNS(PARAM[F,1],PARAM[F,2],PARAM[F,3],
                     PARAM[F,4],PARAM[F,5]);
                'GOTO'NXSTP;
SWTRIL:         TRILLS(PARAM[F,1]);
                'GOTO'NXSTP;
SWSILE:         SILENCE;
NXSTP:          NEWLINE(1);
                'FOR'J:=1'STEP'1'UNTIL'100'DO'
                WRITETEXT('('*')');

                LOWER:=UPPER;
                'END';
              'END';
          'END'LAYER;
          NEWLINE(3);
          WRITETEXT('('RNDCOUNT: ')');
          PRINT(RNDCOUNT,8,0);
          NEWLINE(3);
          'FOR'J:=1'STEP'1'UNTIL'10'DO'
          PRINT(RND(10)-1,1,0);
          NEWLINE(3);
       'END'PROGRAM;
```

```
RUNS - 4  NOTES TO A BEAT
DIATONIC SCALE IN KEY 11     (C=0)  STARTING ON DEGREE 2

THROUGH 1  OCTAVES  STARTING IN OCTAVE 2
GOING DOWN


*****************************************************************


FROM 131  TO 132  TYPE  4


A SEMITONE TRILL ON 10

*****************************************************************


FROM 133  TO 134  TYPE  1

RHYTHM PATTERN (IN HALF-BEATS):

 1   2   1   1   2   1   0

WITH FOLLOWING PITCHES:

 55
 49









 42
 35
 30
 20
*****************************************************************


FROM 135  TO 136  TYPE  4


A SEMITONE TRILL ON  1

*****************************************************************


FROM 137  TO 137  TYPE  3


RUNS - 3  NOTES TO A BEAT
DIATONIC SCALE IN KEY  6     (C=0)  STARTING ON DEGREE 3

THROUGH 1  OCTAVES  STARTING IN OCTAVE 2
GOING DOWN


*****************************************************************


FROM 138  TO 139  TYPE  1
```

```
RHYTHM PATTERN (IN HALF-BEATS):

 1   1   1   1   2   1   1

WITH FOLLOWING PITCHES:

  7
  1
  6
 12
  5
  6
  6

*****************************************************************

FROM 140  TO 142  TYPE  4

A SEMITONE TRILL ON 12

*****************************************************************

FROM 143  TO 143  TYPE  5




RUNS - 6  NOTES TO A BEAT
DIATONIC SCALE IN KEY  0      (C=0)  STARTING ON DEGREE 6

THROUGH 1  OCTAVES  STARTING IN OCTAVE 2
GOING DOWN

*****************************************************************

FROM 144  TO 145  TYPE  4

A SEMITONE TRILL ON 50

*****************************************************************

FROM 146  TO 147  TYPE  5

RUNS - 7  NOTES TO A BEAT
DIATONIC SCALE IN KEY 11      (C=0)  STARTING ON DEGREE 5

THROUGH 1  OCTAVES  STARTING IN OCTAVE 1
GOING DOWN

*****************************************************************
```

IIa *leap*

This piece is for solo clarinet.

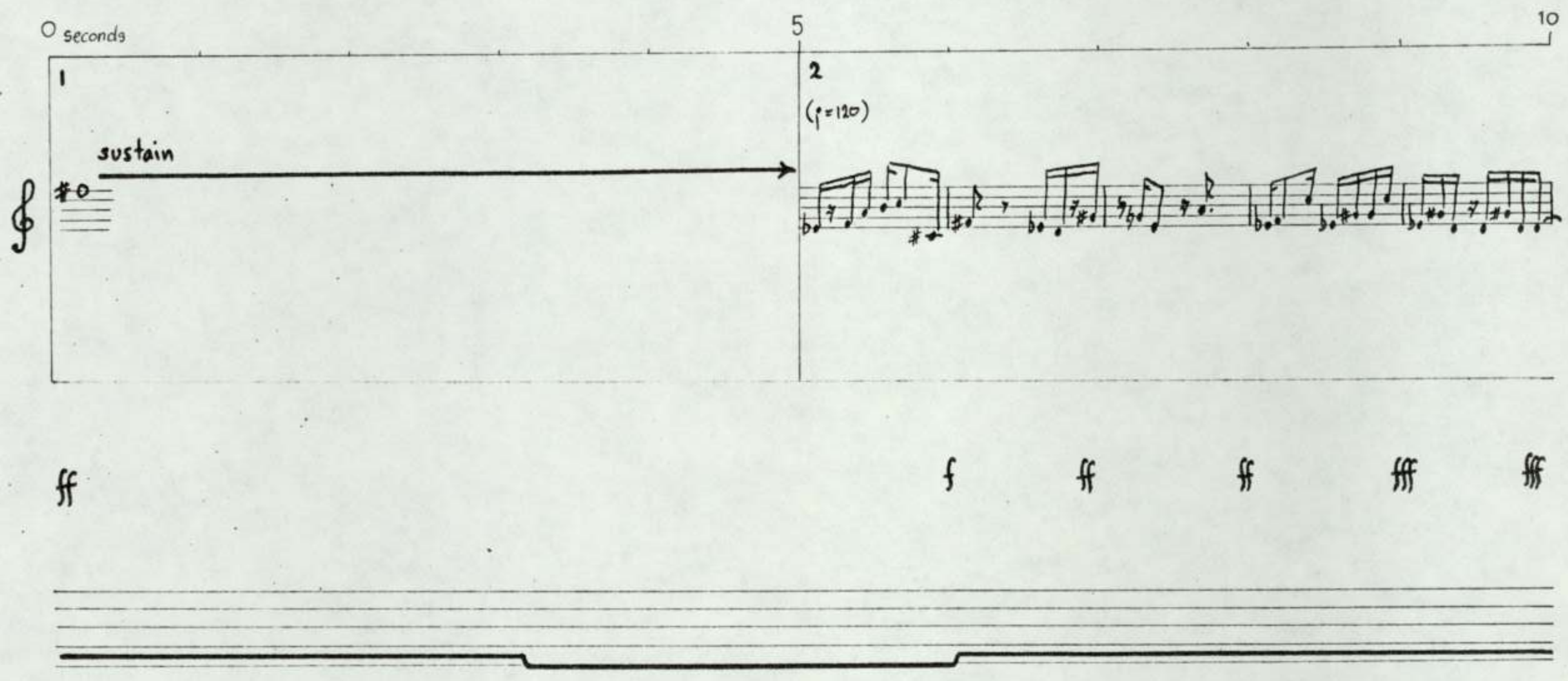
The five lines lower down each page indicate the player's movements.

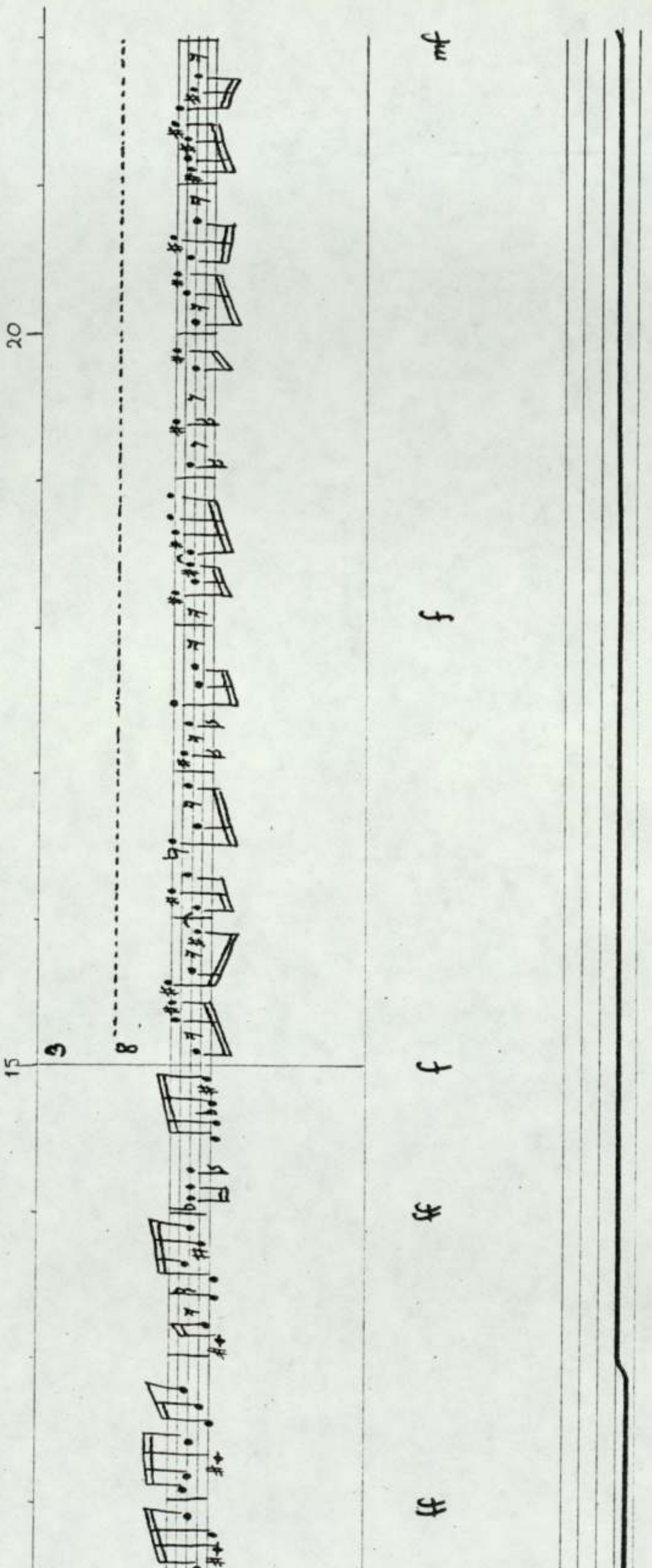
This extract is the first movement of four.

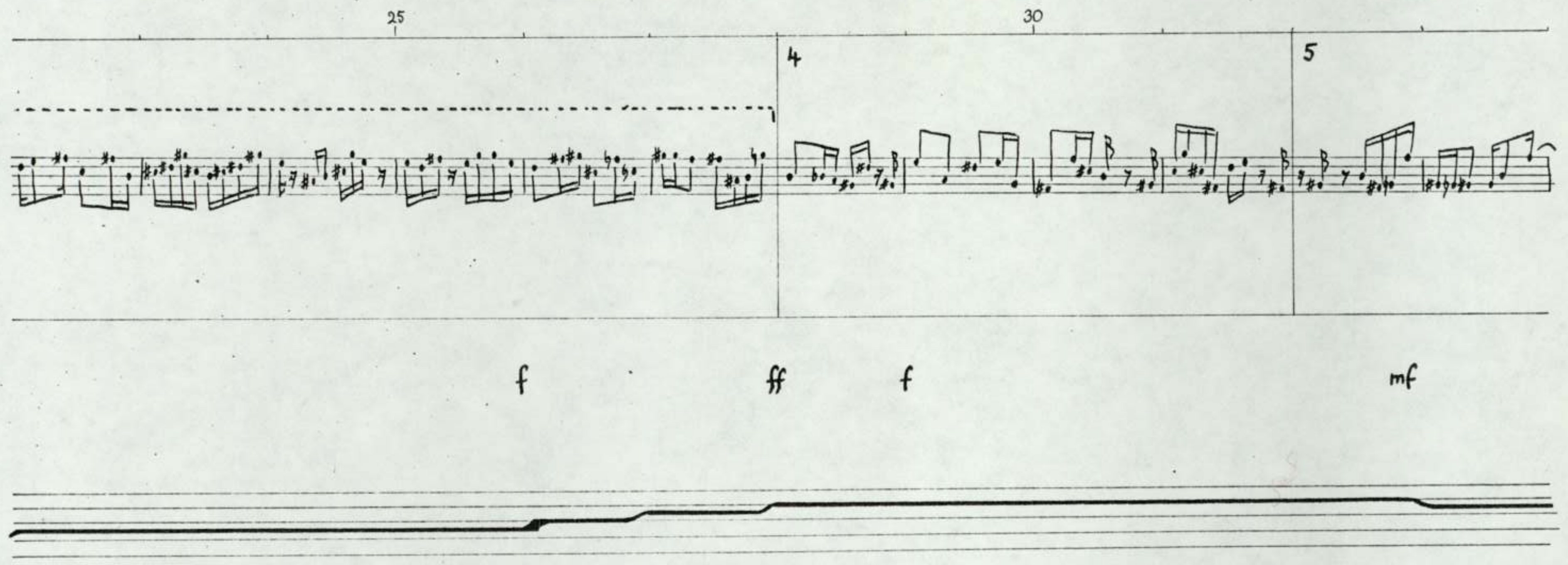
A P P E N D I X   II

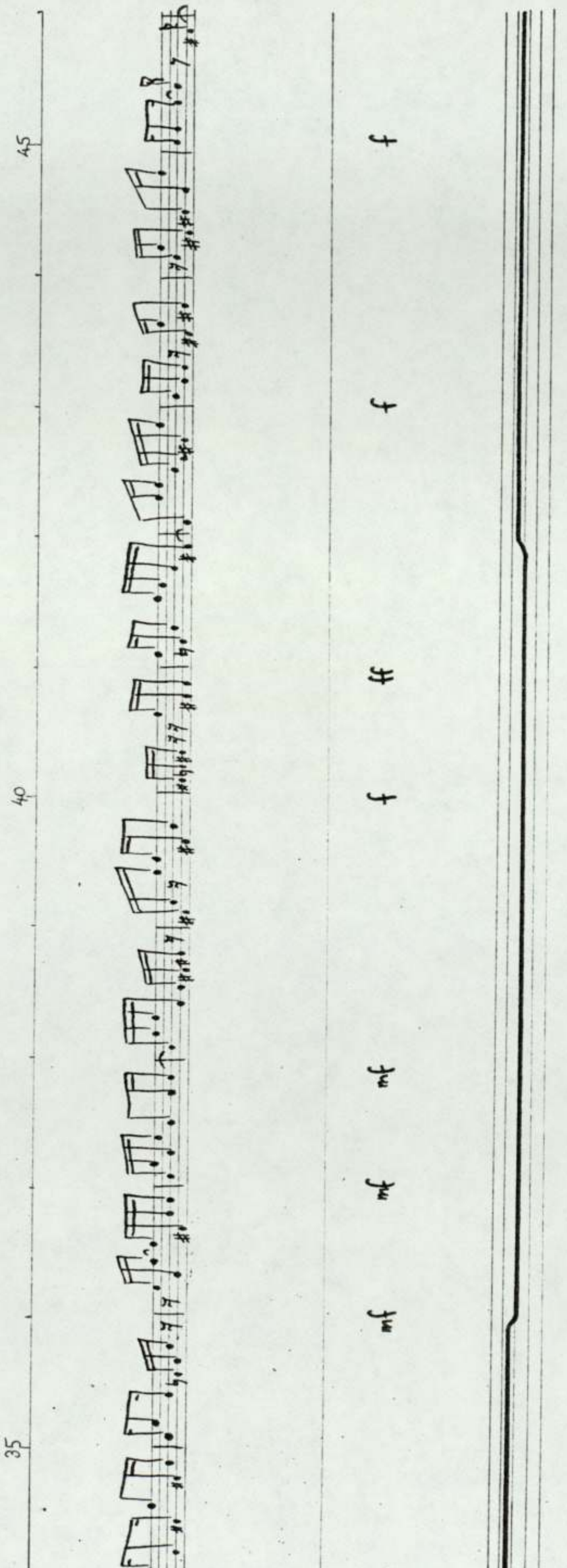
S C O R E S

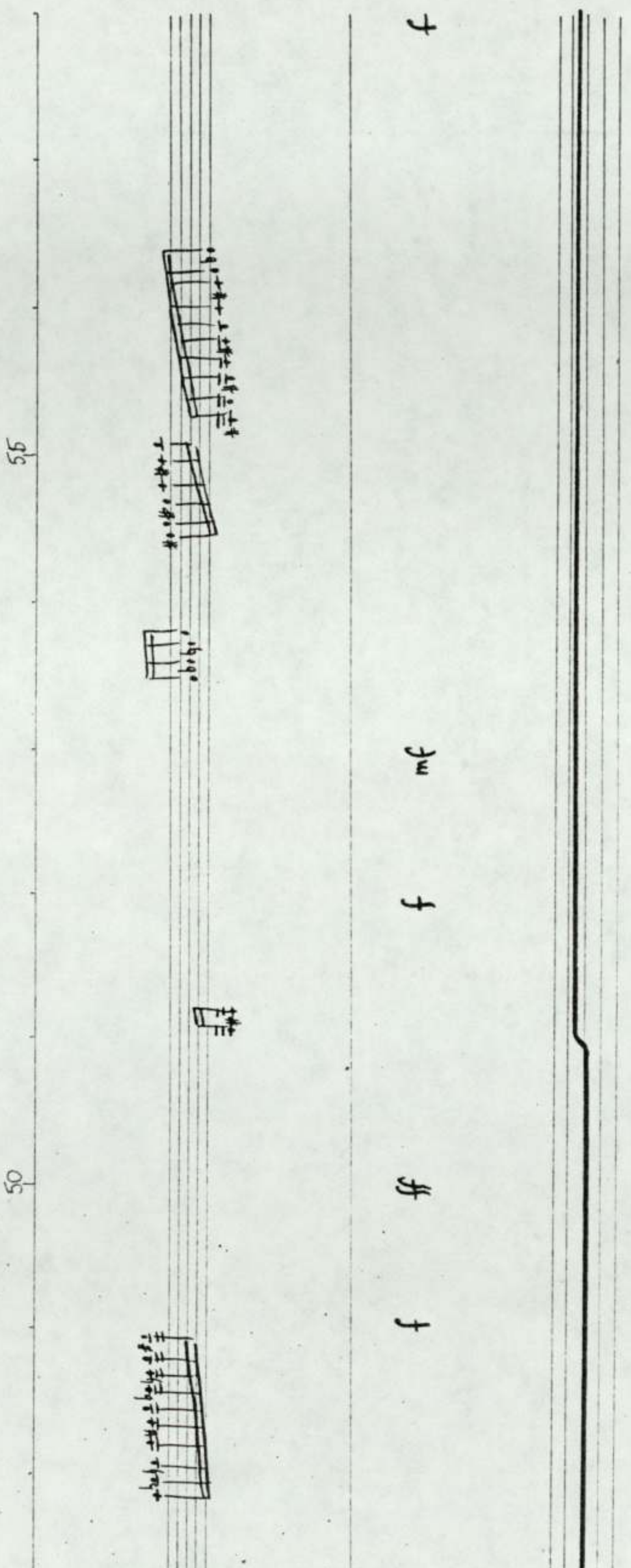
FIRST

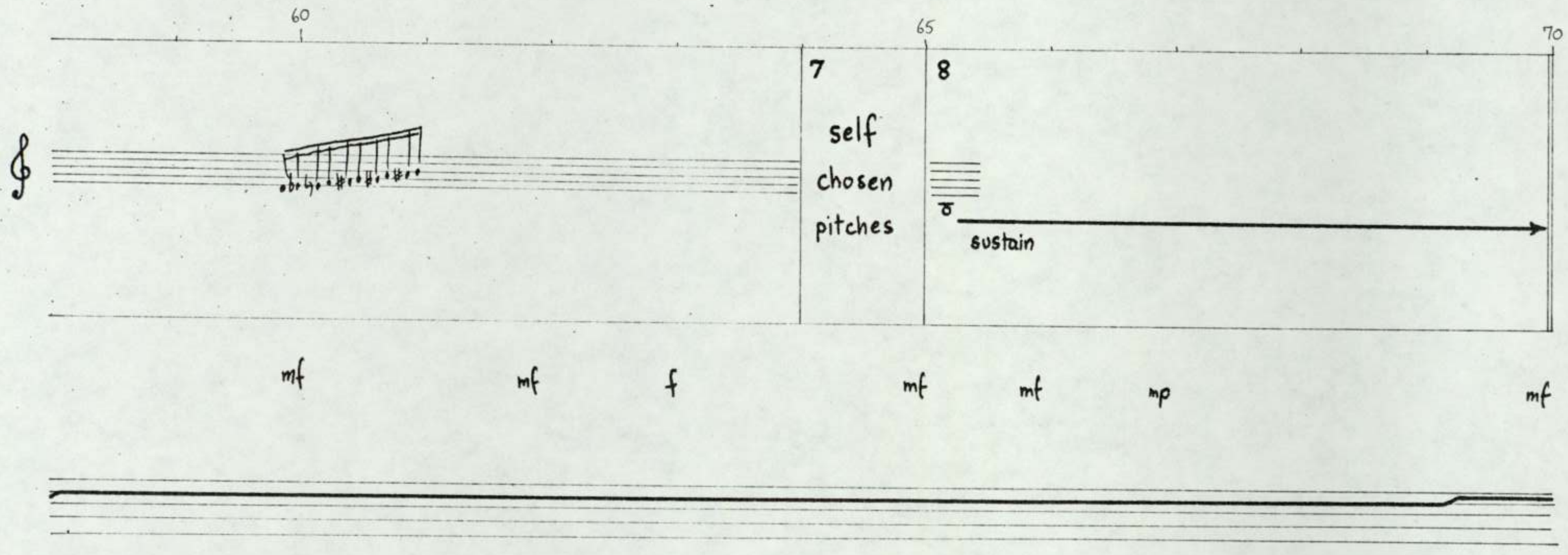



10
15
20
9
8


25
30
4
5
f
ff
f
mf


35
40
45


6
50
55
f
ff
f
mf
f

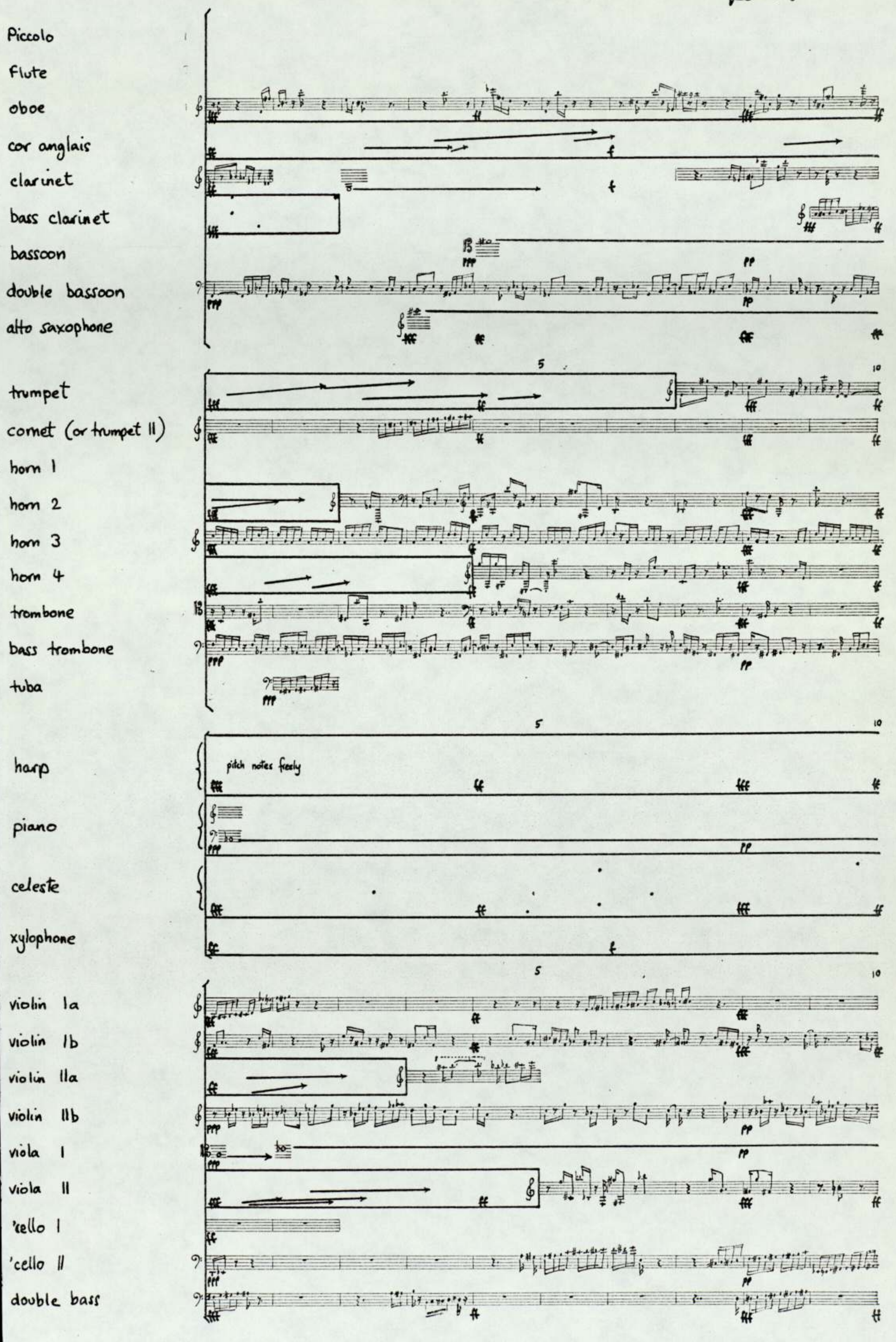

60
65
70
7
8
self
chosen
pitches
sustain
mf
mf
f
mf
mf
mp
mf

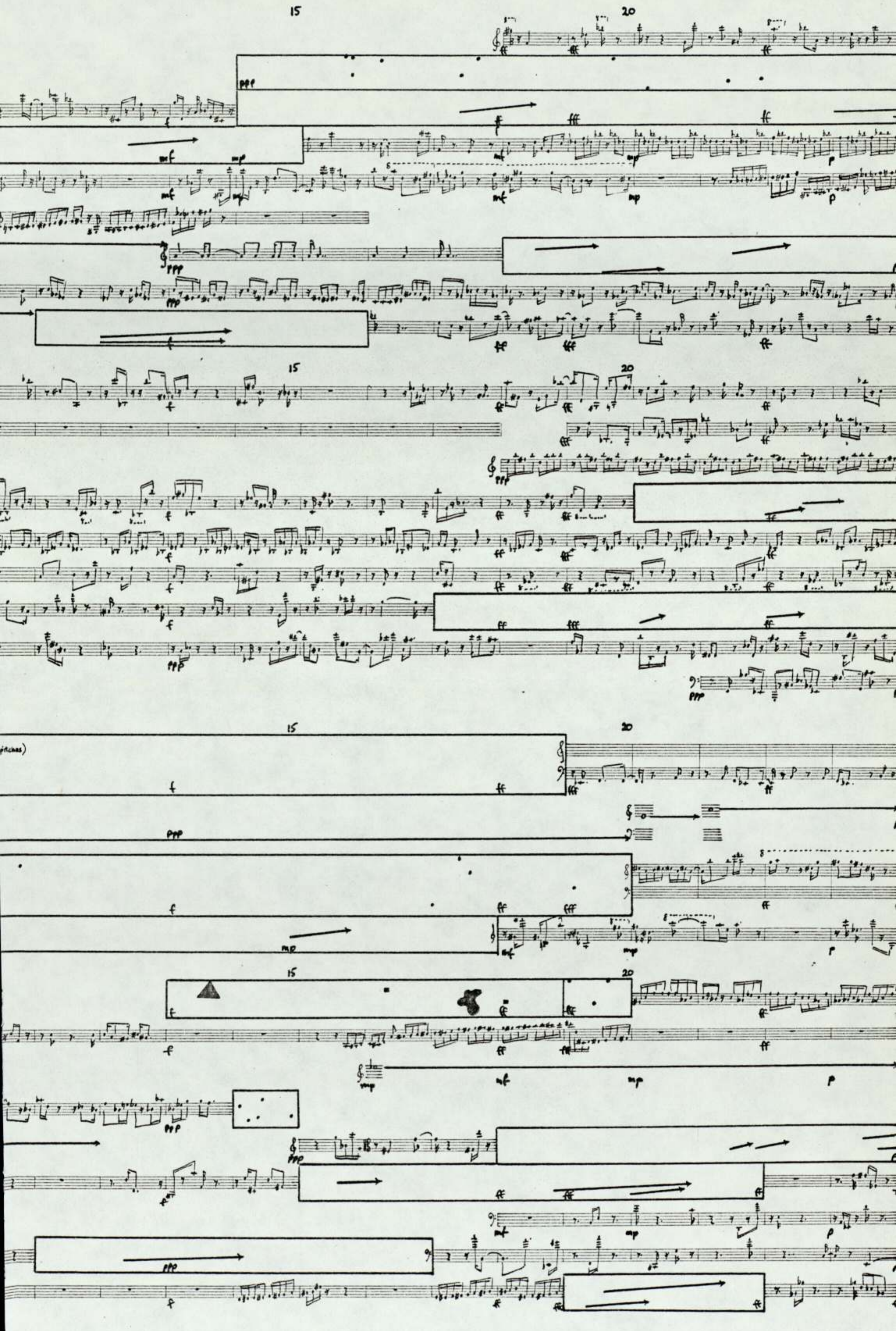
IIb *maytricks*

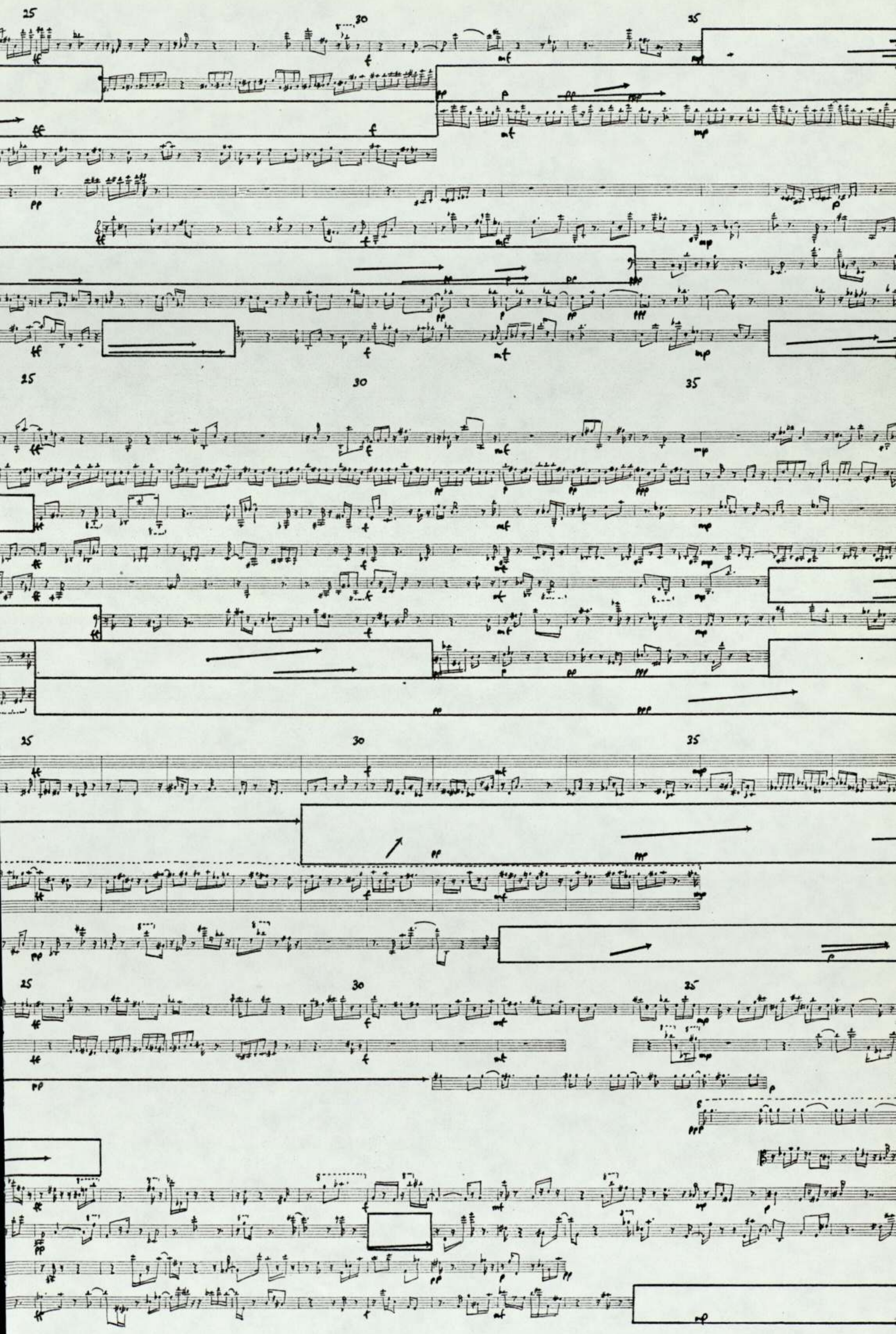
The first 66 bars of *maytricks* for orchestra.

HAT TRICKS for orchestra

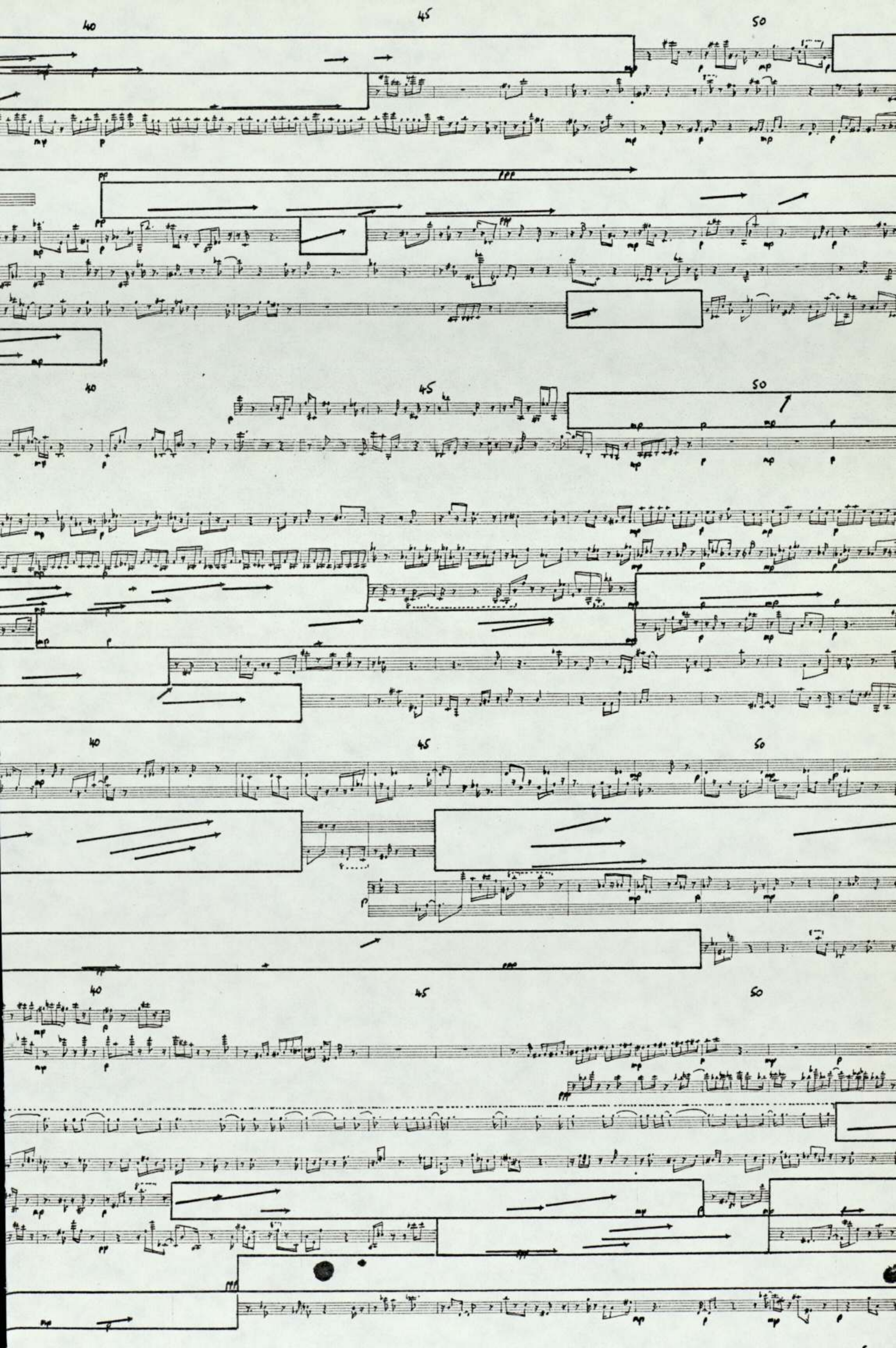
[illegible]



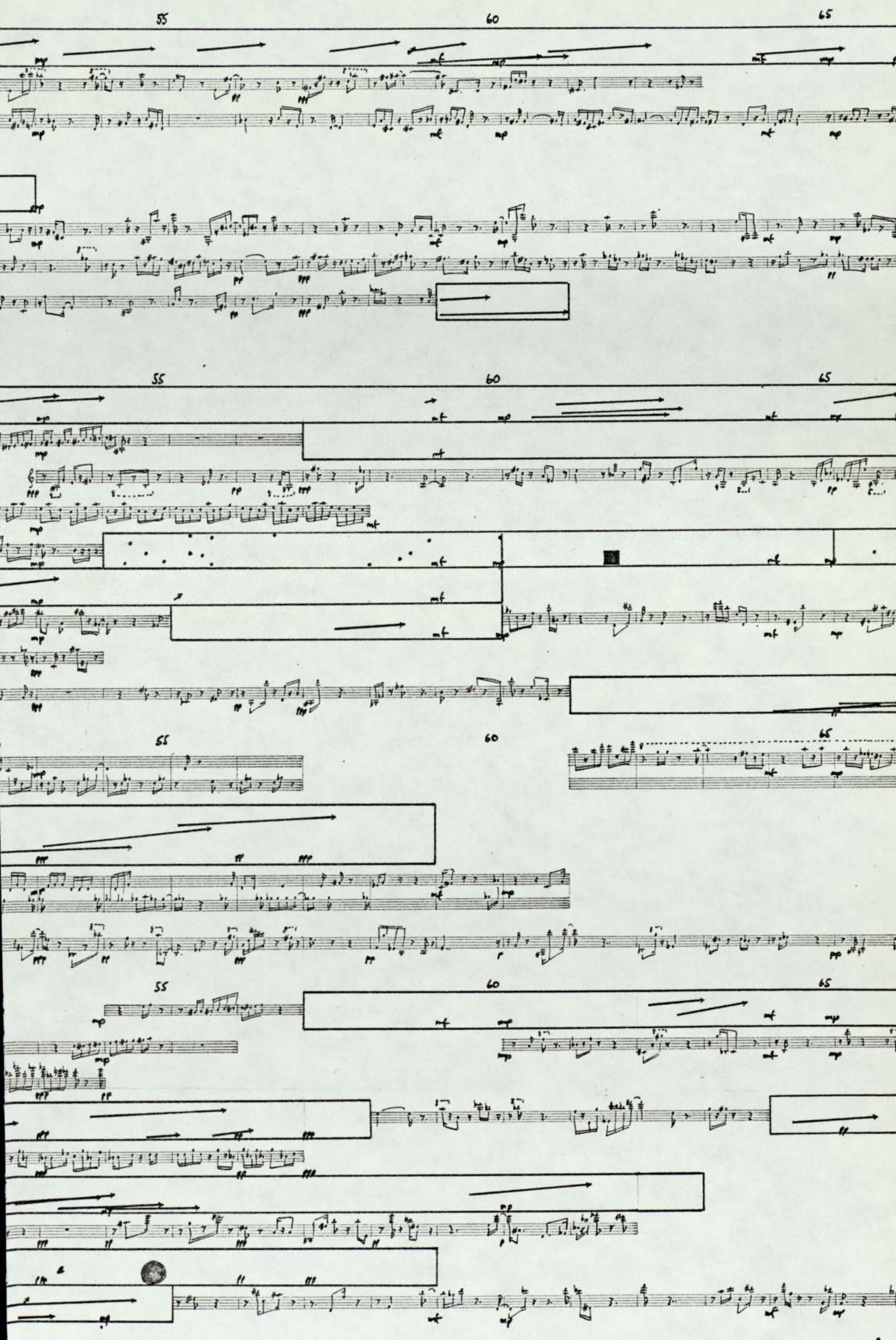

15
20
ppp
mf
mp
ff
fff
f
p
15
20
pitches)
15
20



25
30
35
ff
f
mf
mp
pp
p
ppp



40
45
50
mp
p
pp
ppp
40
45
50
40
45
50
40
45
50

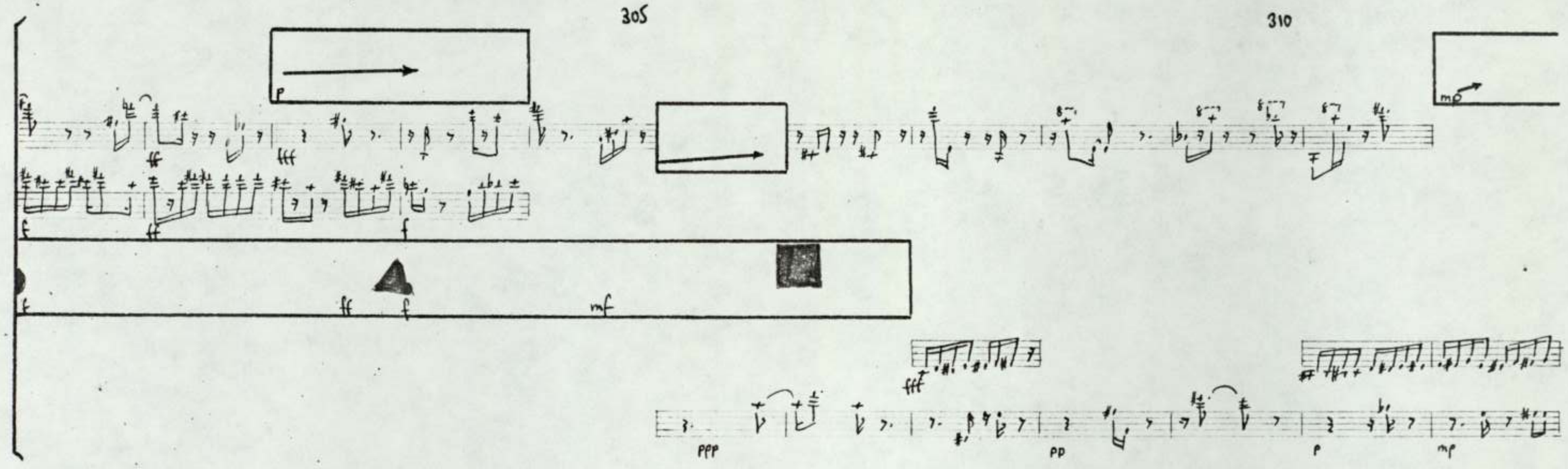


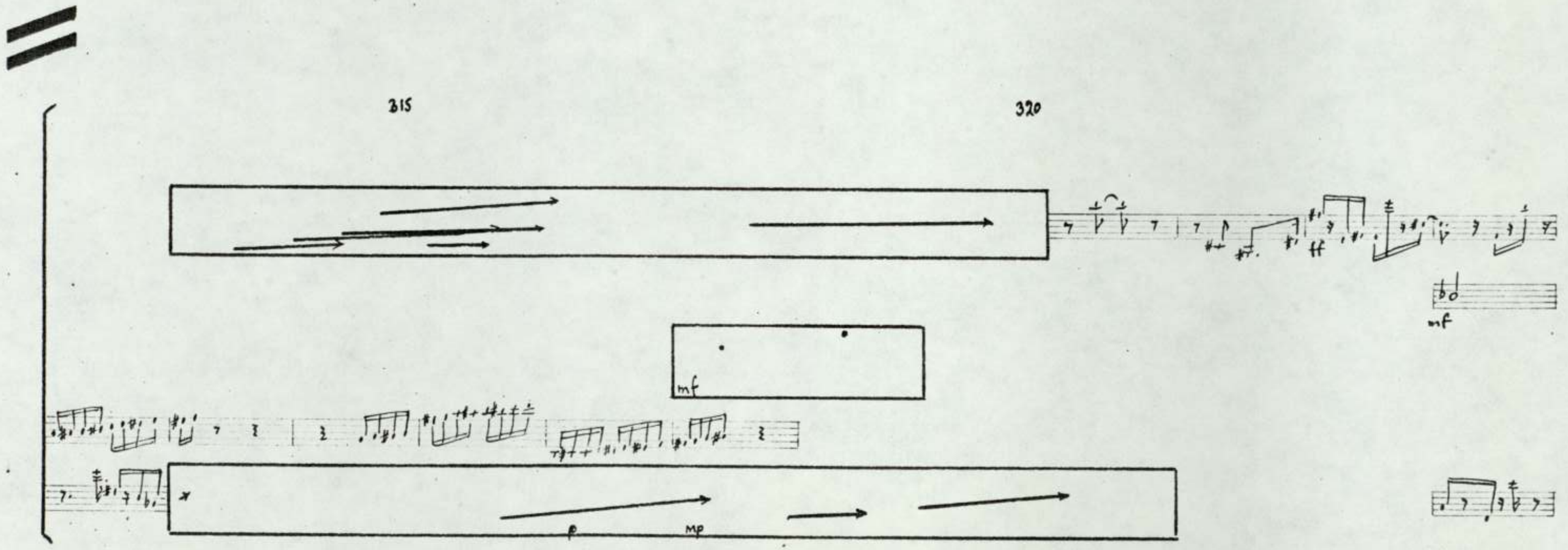
55
60
65
mp
mf
pp
ppp
ff
55
60
65
55
60
65
55
60
65

IIc *pursuit*

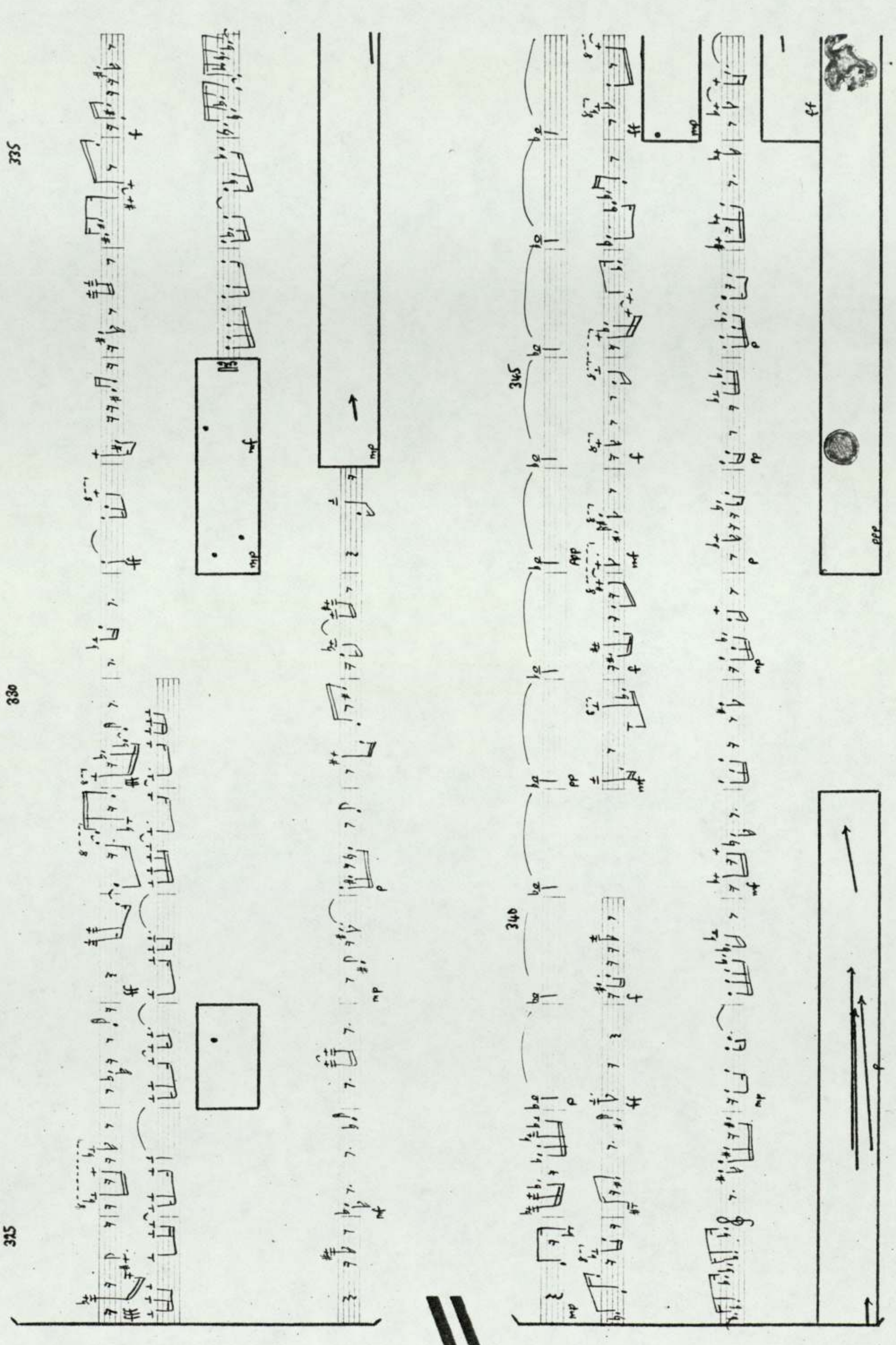
Scored for 3 violins, viola, 'cello and double bass.

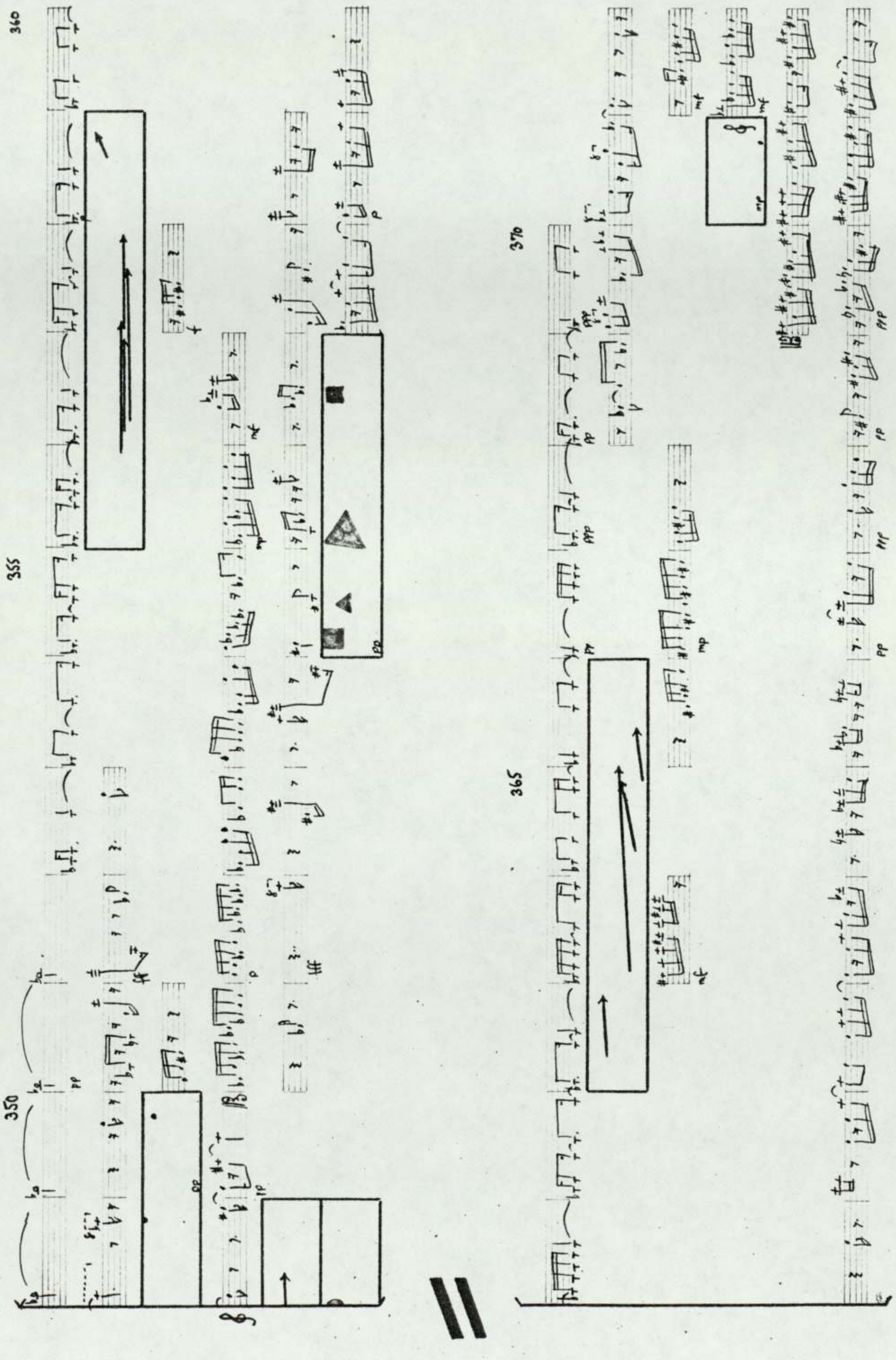
This extract is from the end of the piece, bars 301 - 381.

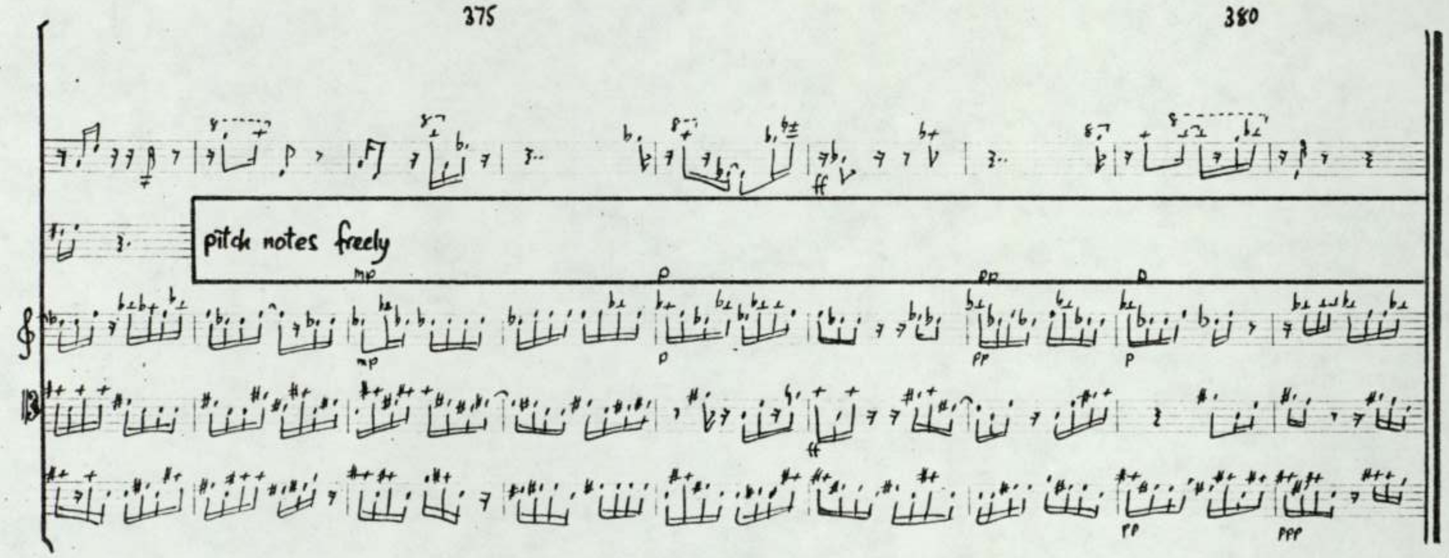

305
310
p
fff
ff
f
mf
mp
ppp
pp


315
320
mf
p
mp

C1




350
355
360
365
370



10 mins.

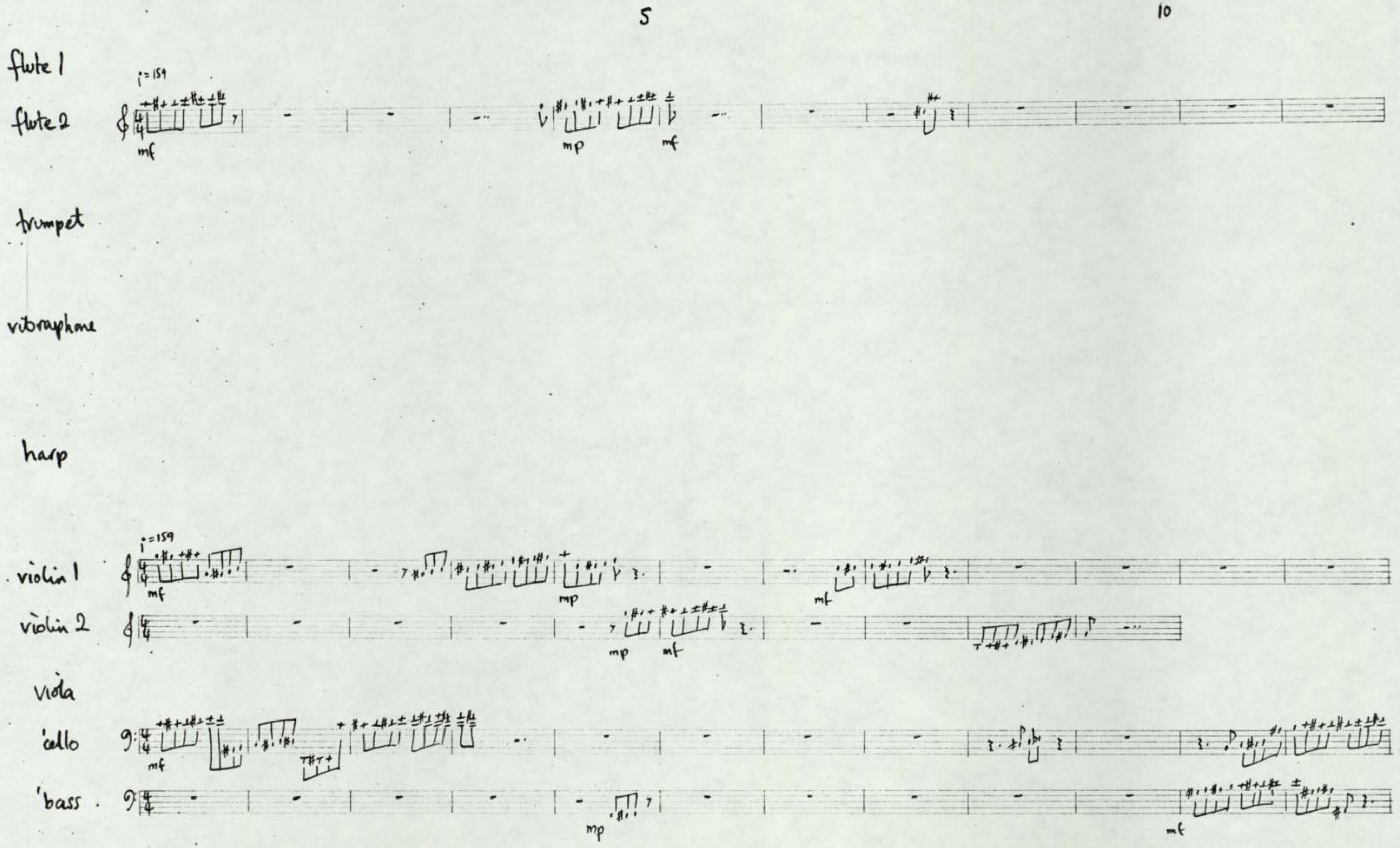
IId *light*

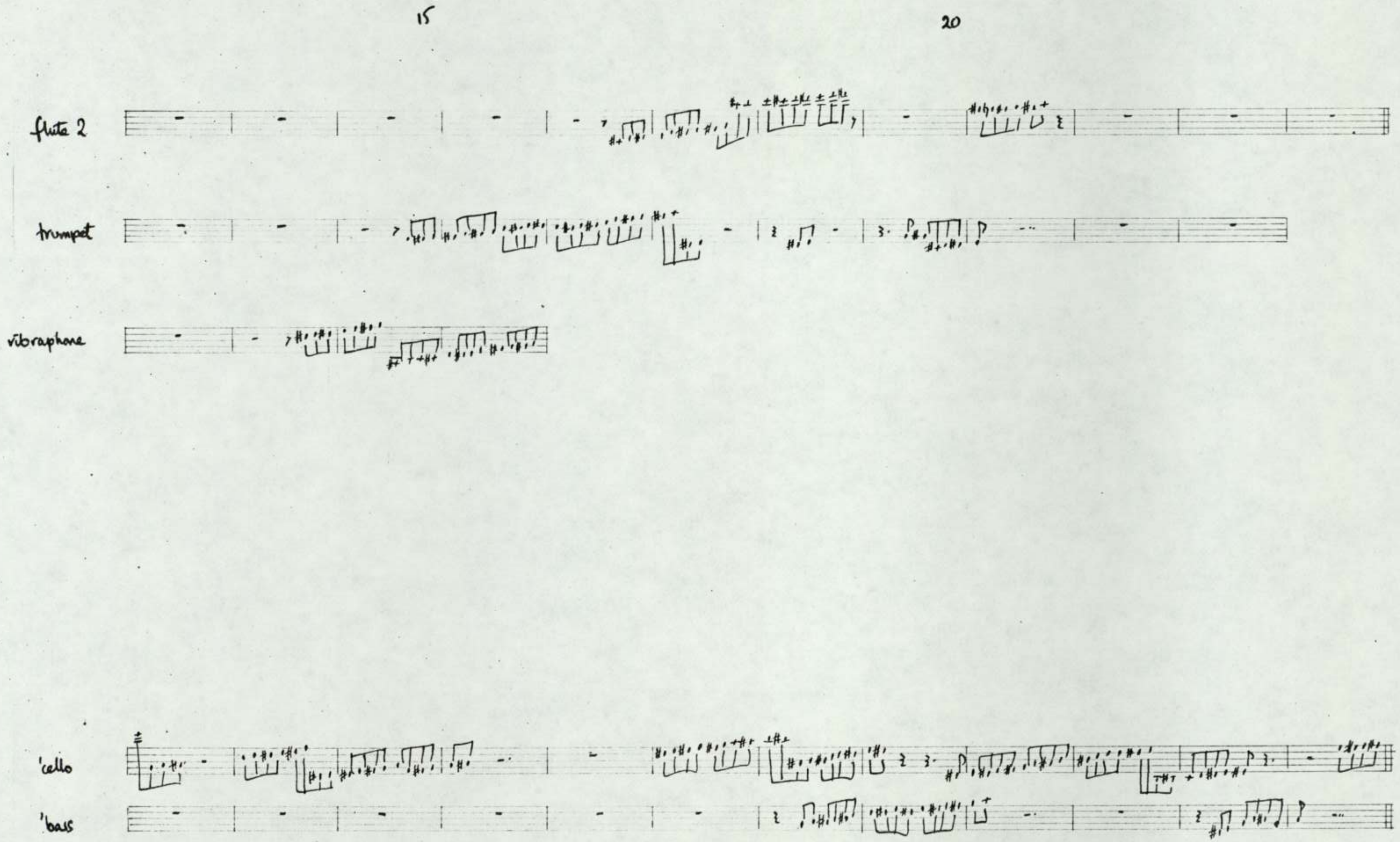
Scored for two flutes, trumpet, vibraphone, harp, two violins, viola, 'cello and double bass.

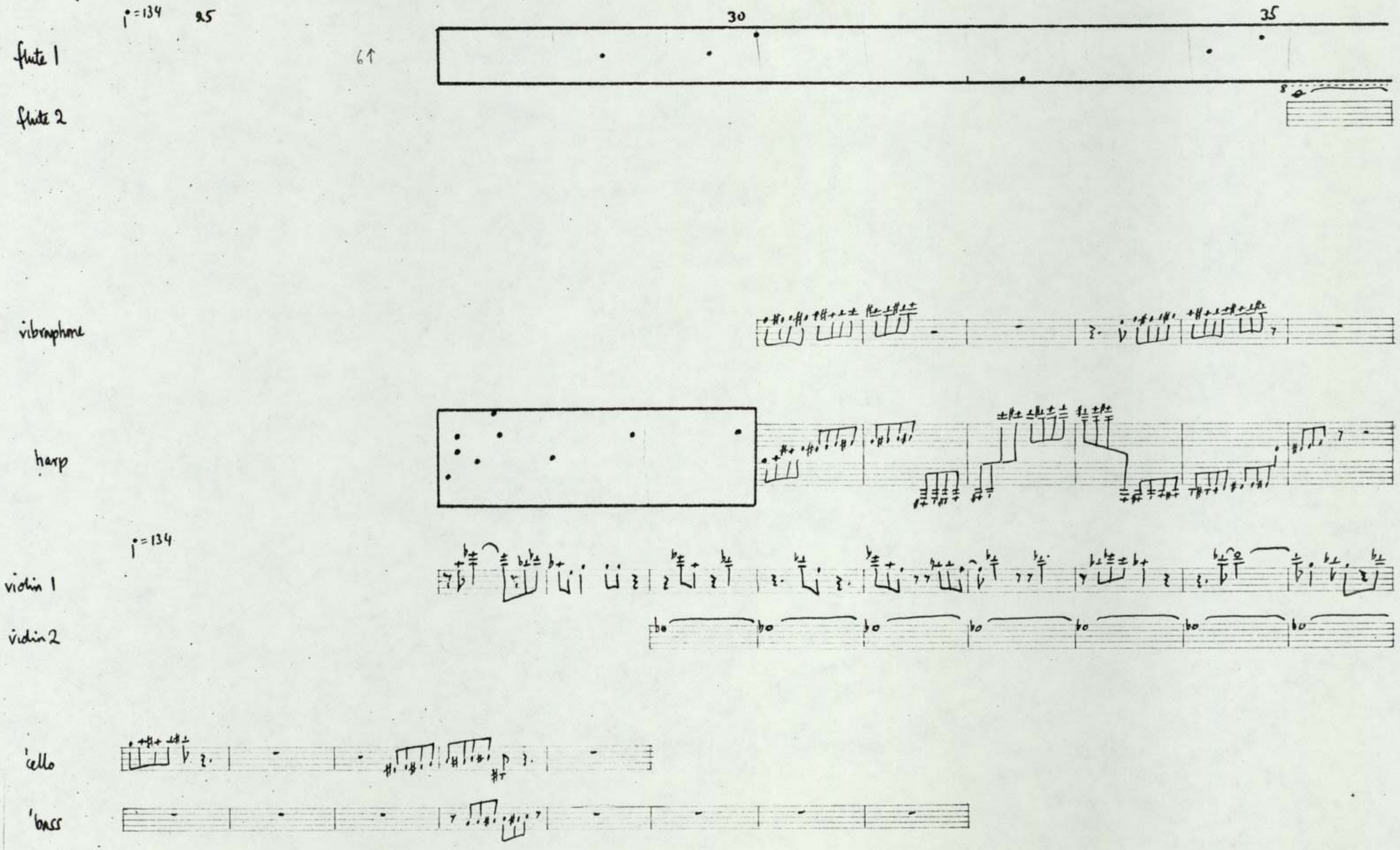
This extract is the first movement of seven: *Coming of light*.

light I: coming of light

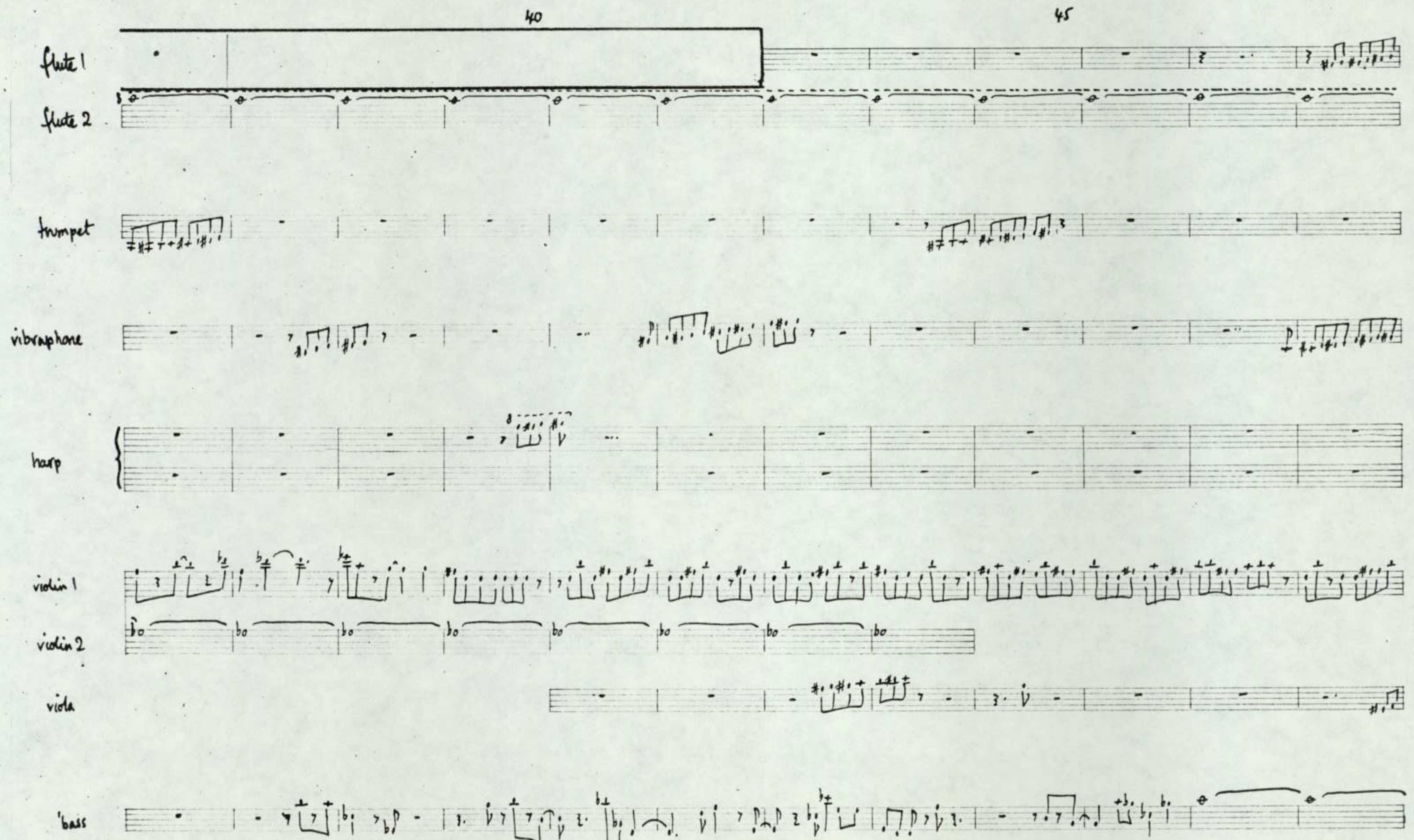
'Arise, shine; for your light has come' (Is 60:1)

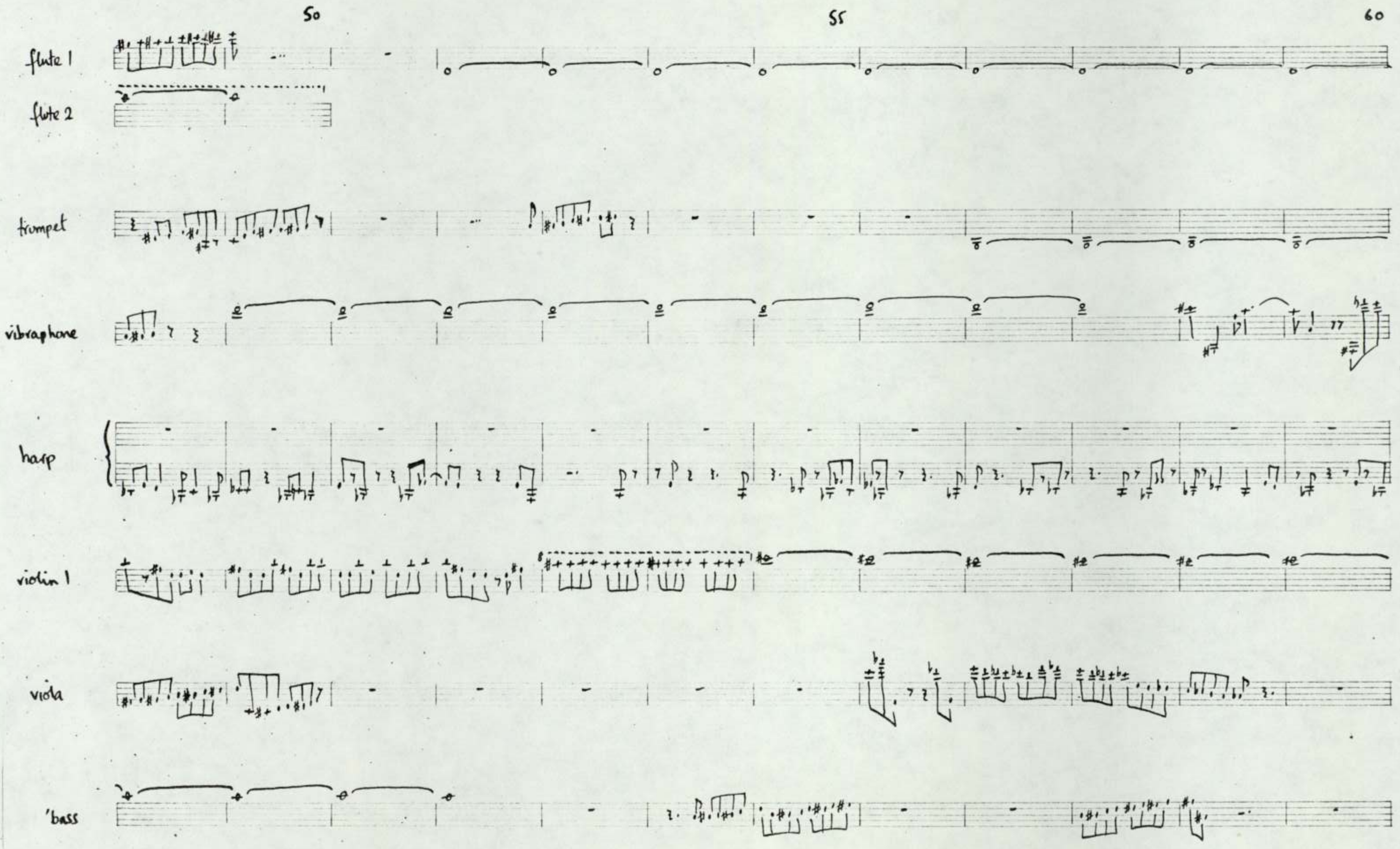


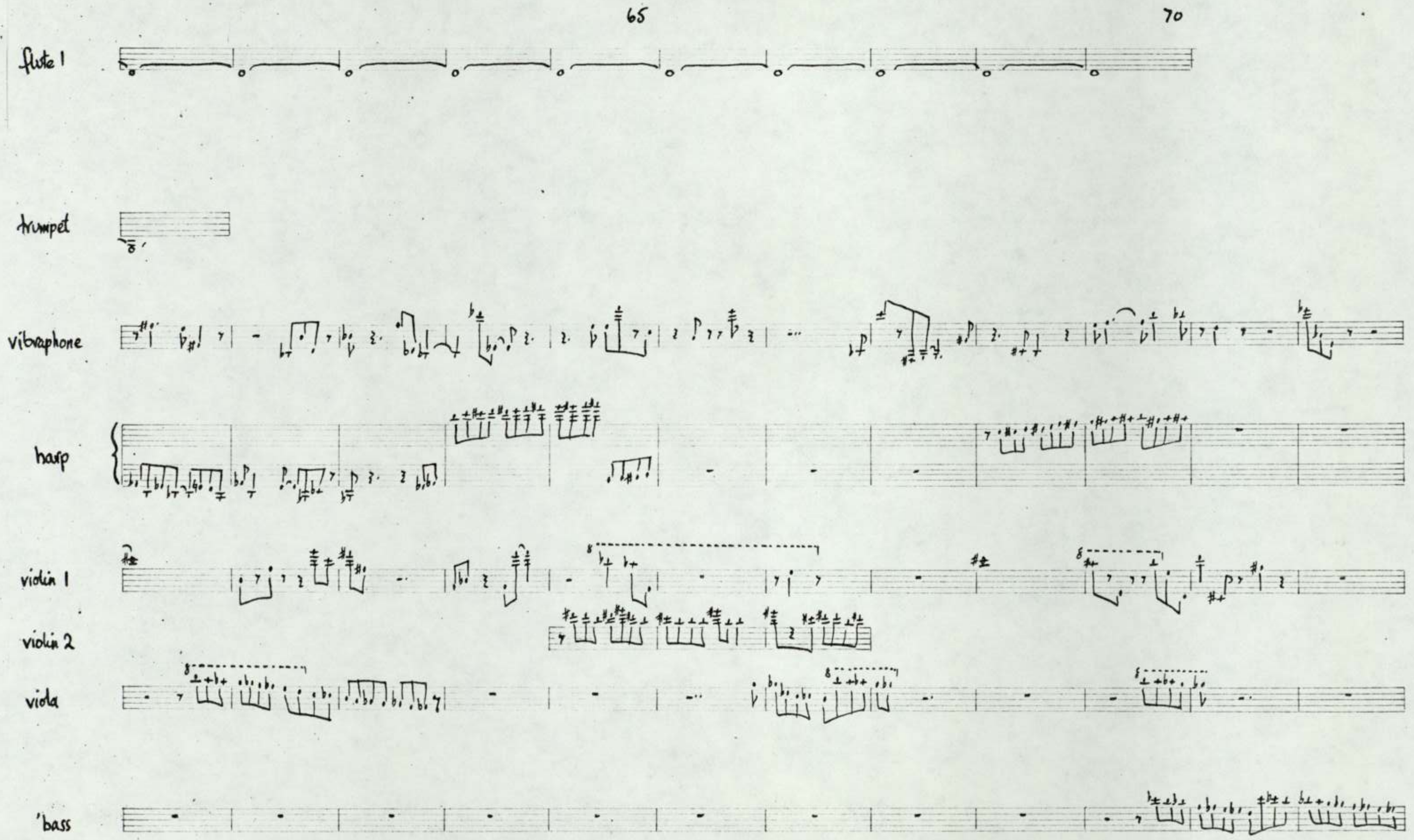

15
20
flute 2
trumpet
vibraphone
'cello
'bass

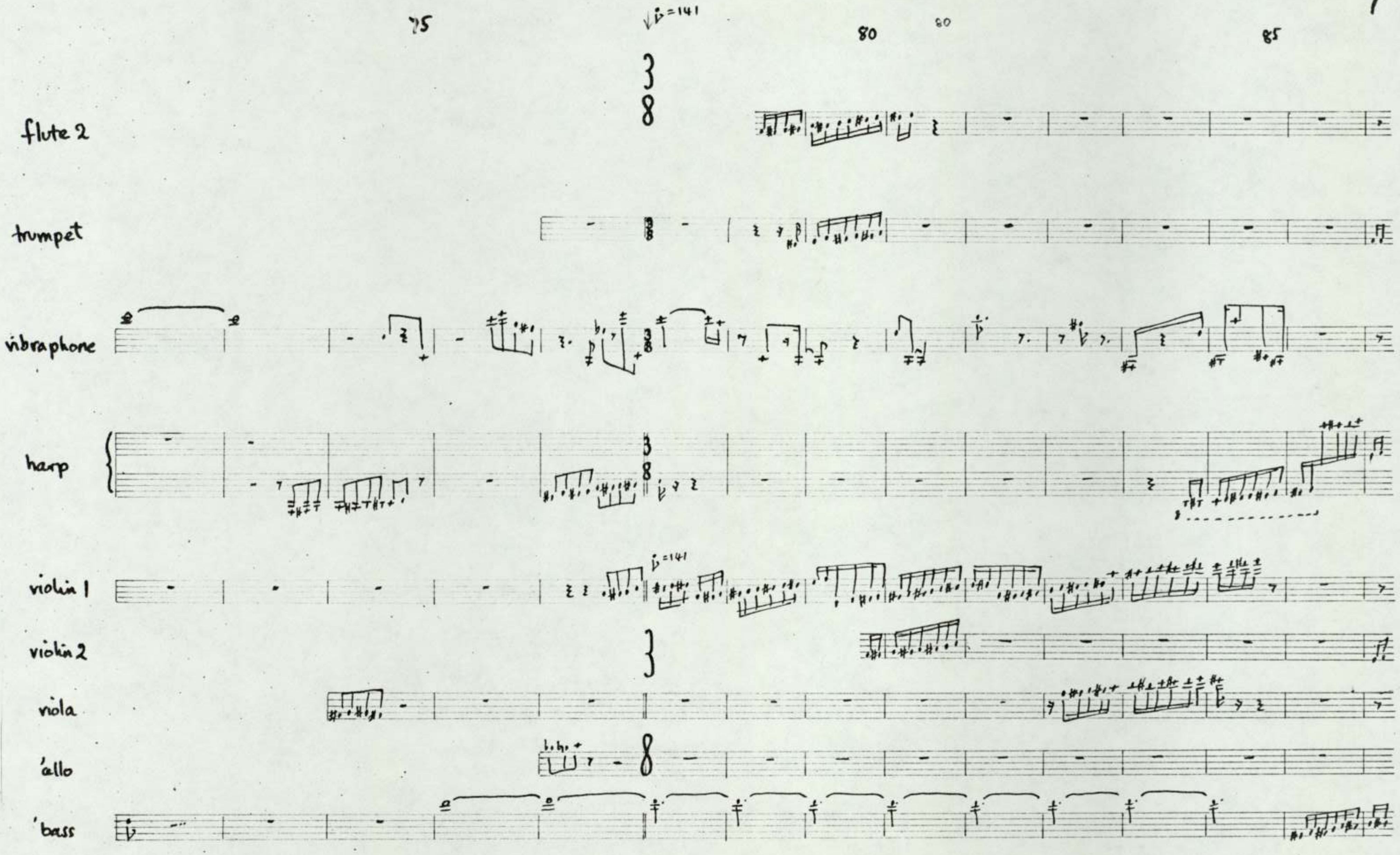


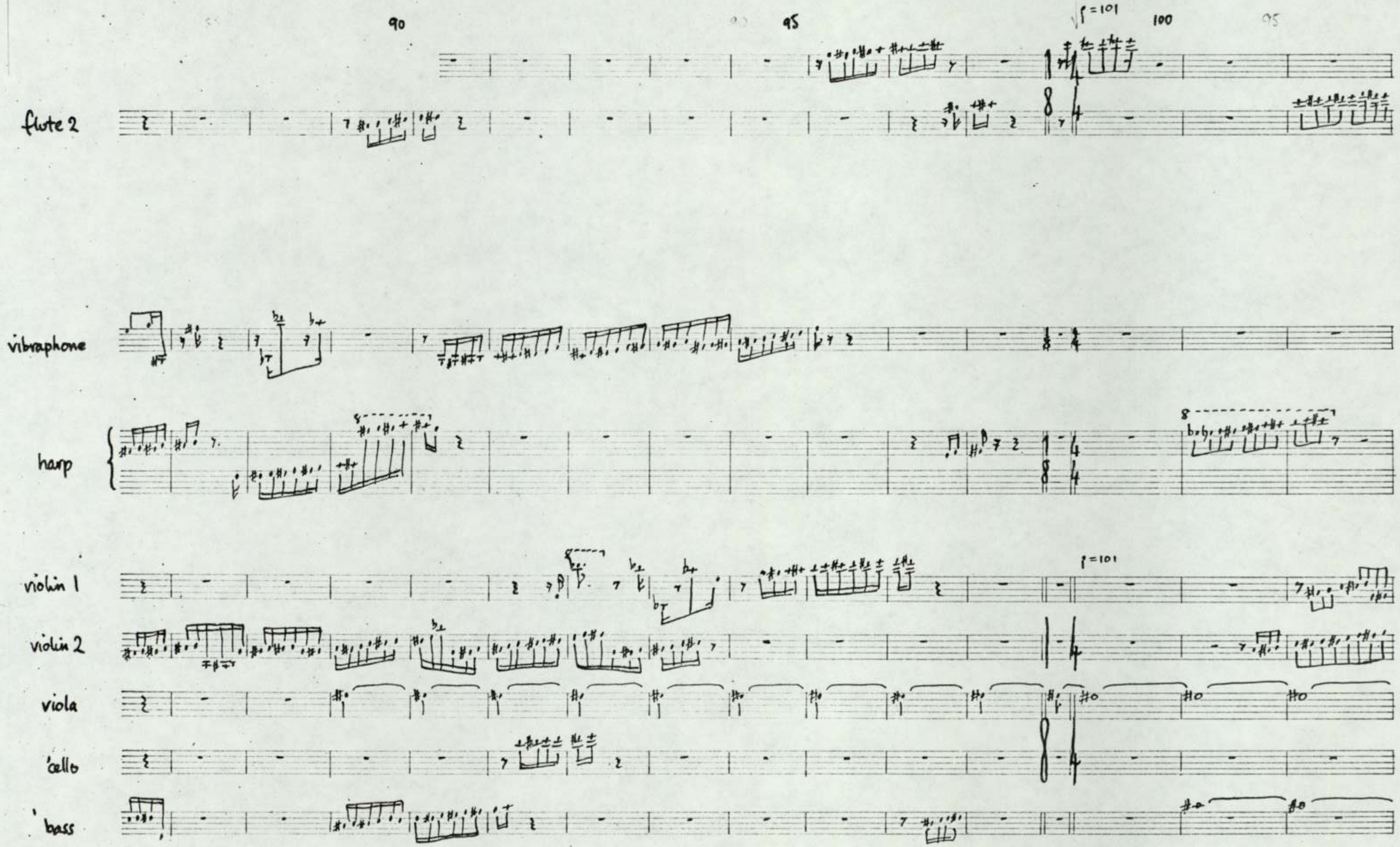
♩=134
25
30
35
flute 1
6↑
flute 2
8
vibraphone
harp
♩=134
violin 1
violin 2
'cello
'bass

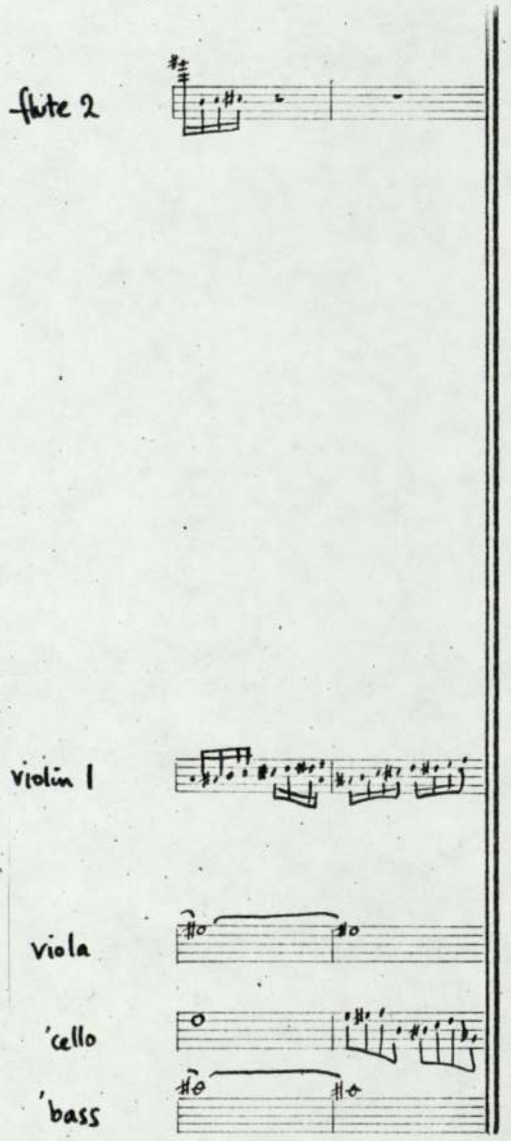

40
45
flute 1
flute 2
trumpet
vibraphone
harp
violin 1
violin 2
viola
bass


50
55
60
flute 1
flute 2
trumpet
vibraphone
harp
violin 1
viola
'bass


65
70
flute 1
trumpet
vibraphone
harp
violin 1
violin 2
viola
'bass


75
♪=141
80
80
85
3/8
flute 2
trumpet
vibraphone
harp
violin 1
violin 2
viola
'cello
'bass


90
95
♩=101
100
flute 2
vibraphone
harp
violin 1
♩=101
violin 2
viola
'cello
bass


flute 2
violin I
viola
'cello
'bass

IIe *symposium*

Scored for flute, oboe, B♭ clarinet, bassoon, horn, B♭ trumpet, violin, viola, 'cello and double bass.

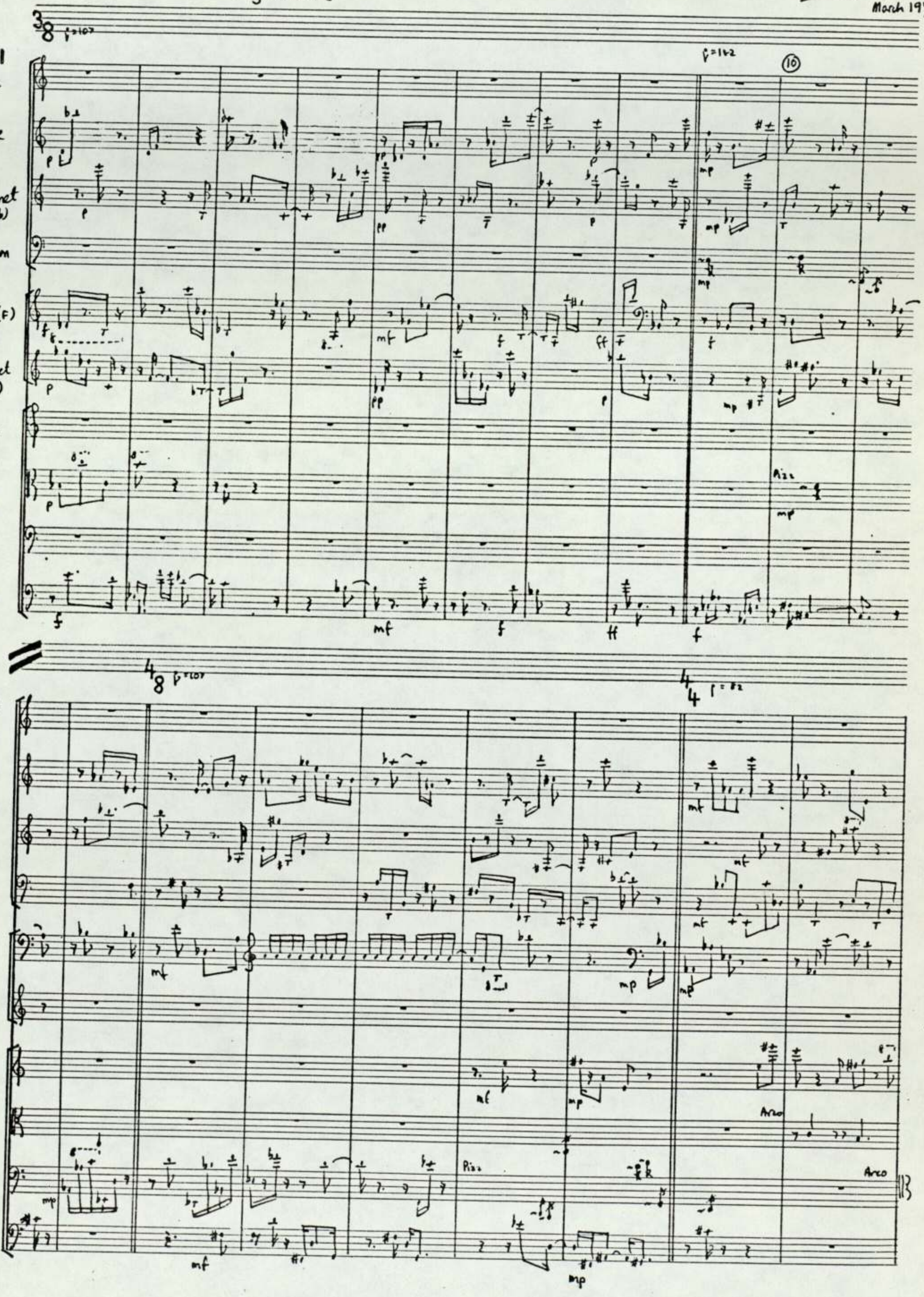
This extract is the first movement of five: *alignments*.

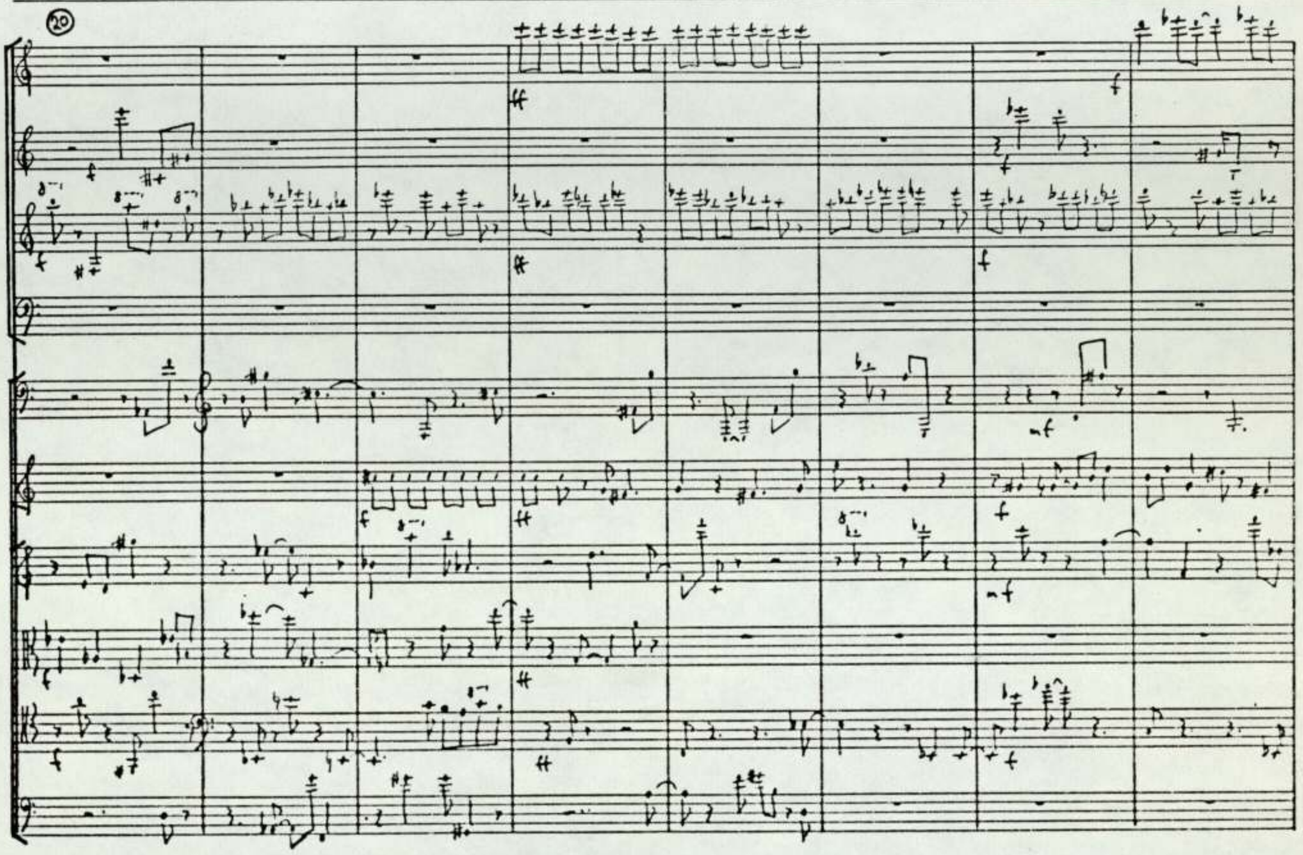
Symposium I for Chamber Ensemble

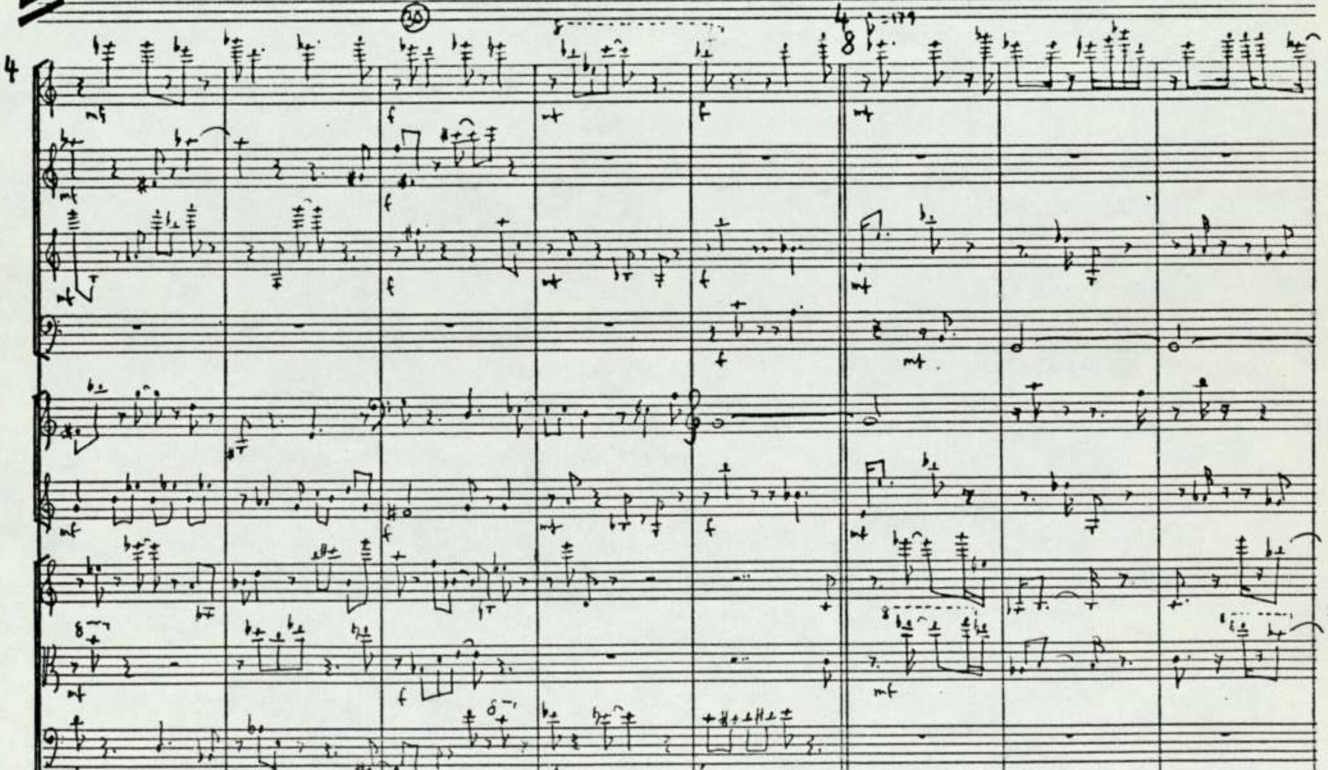
[illegible]

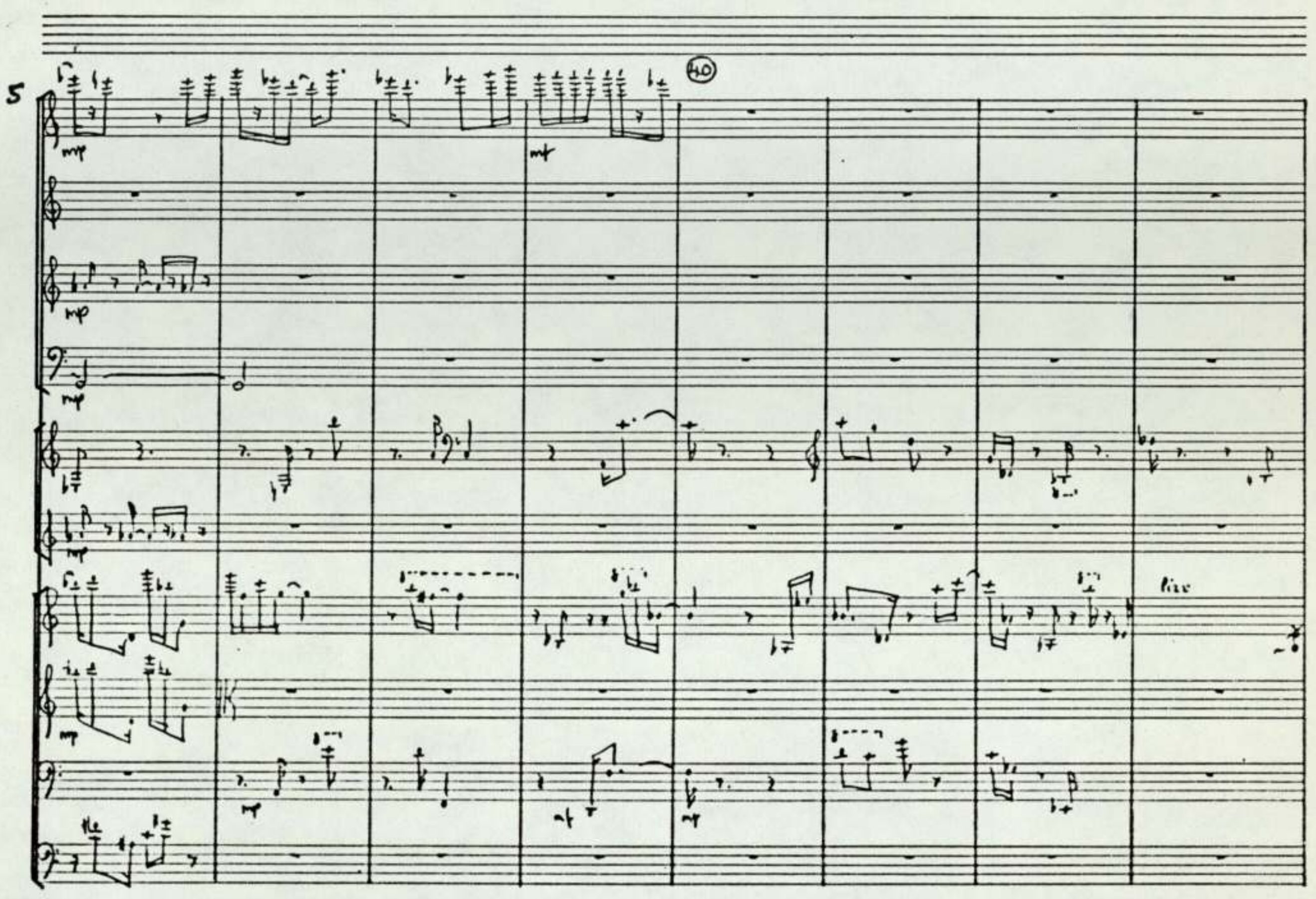
March 1977

1 : alignments

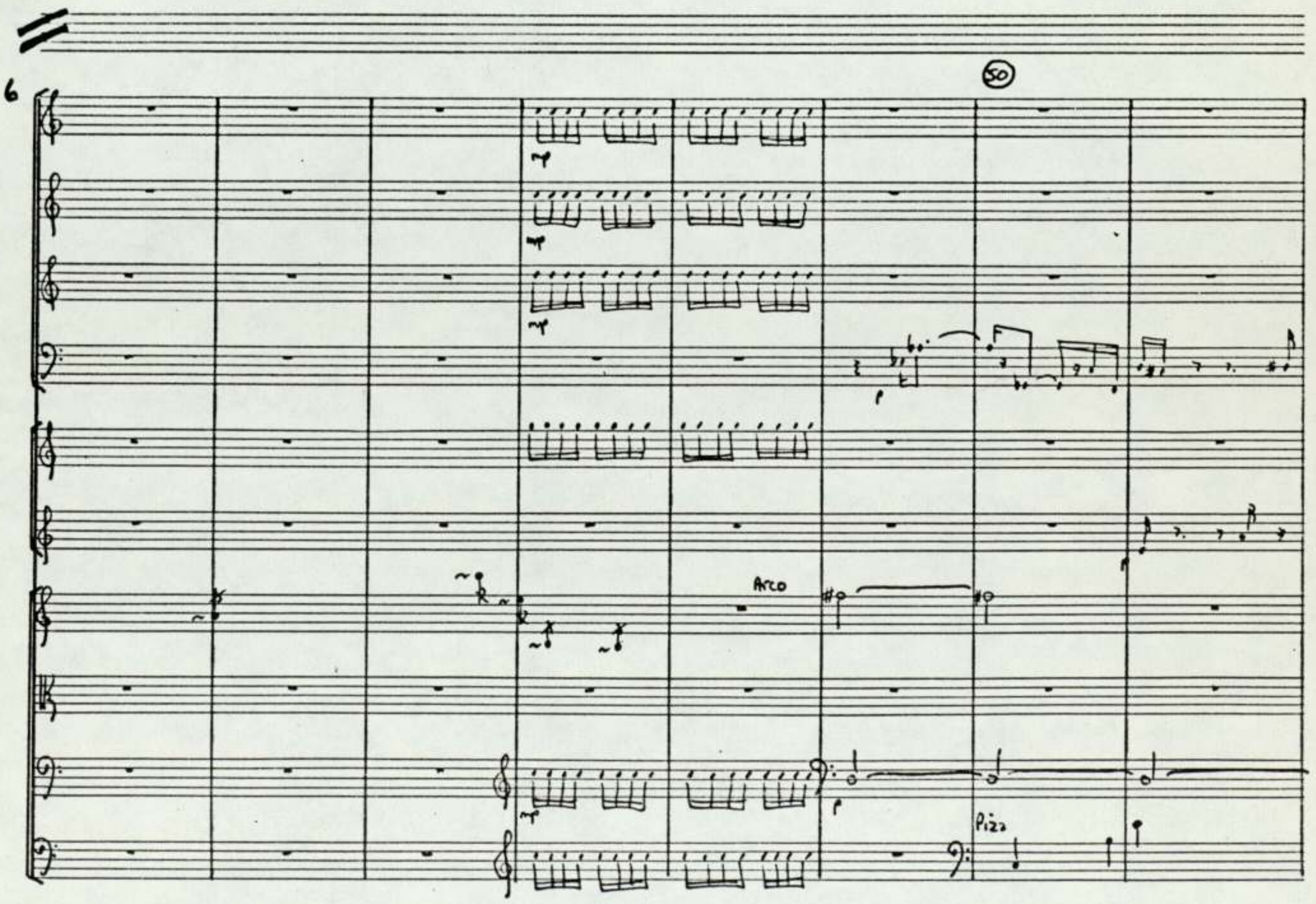


3


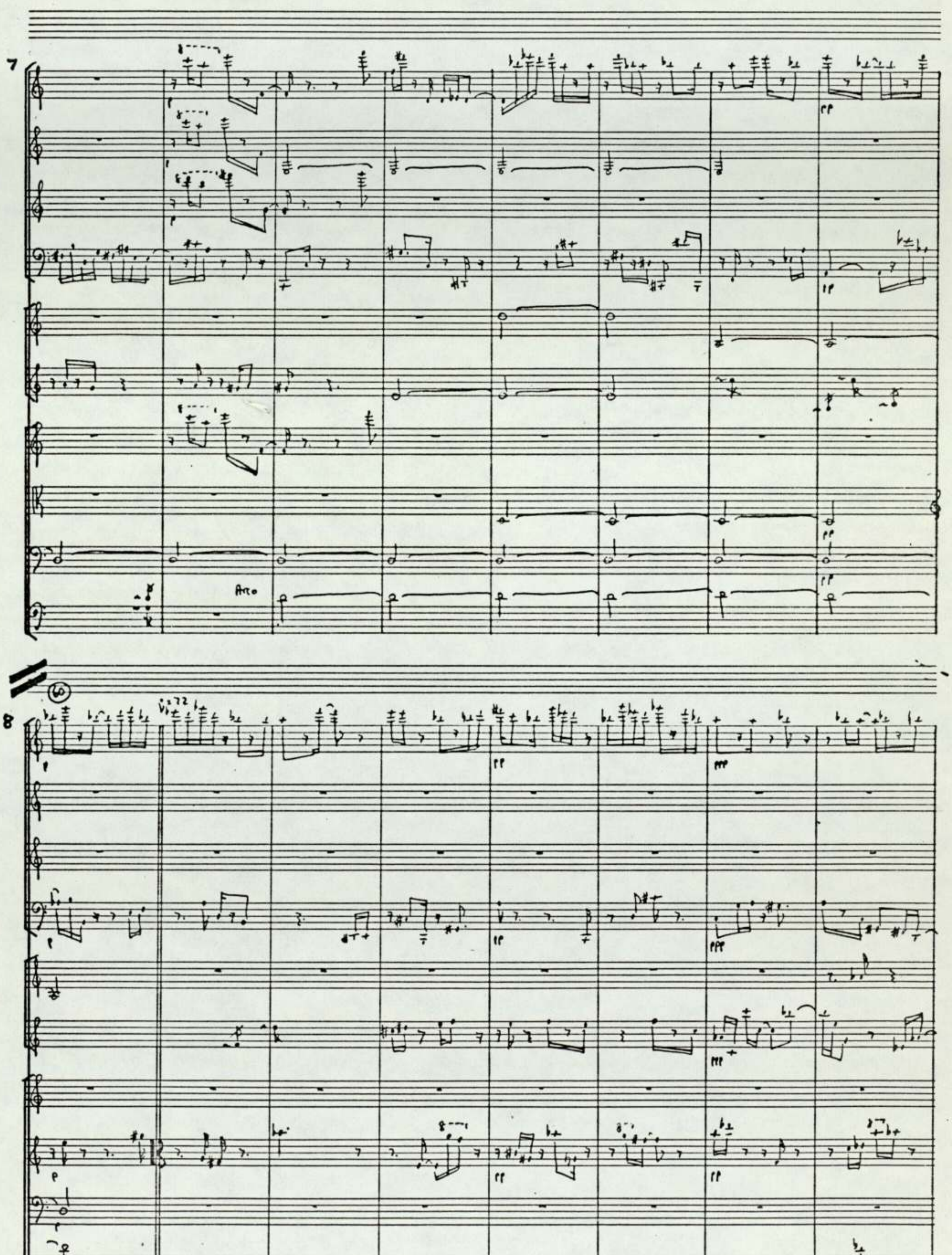
4


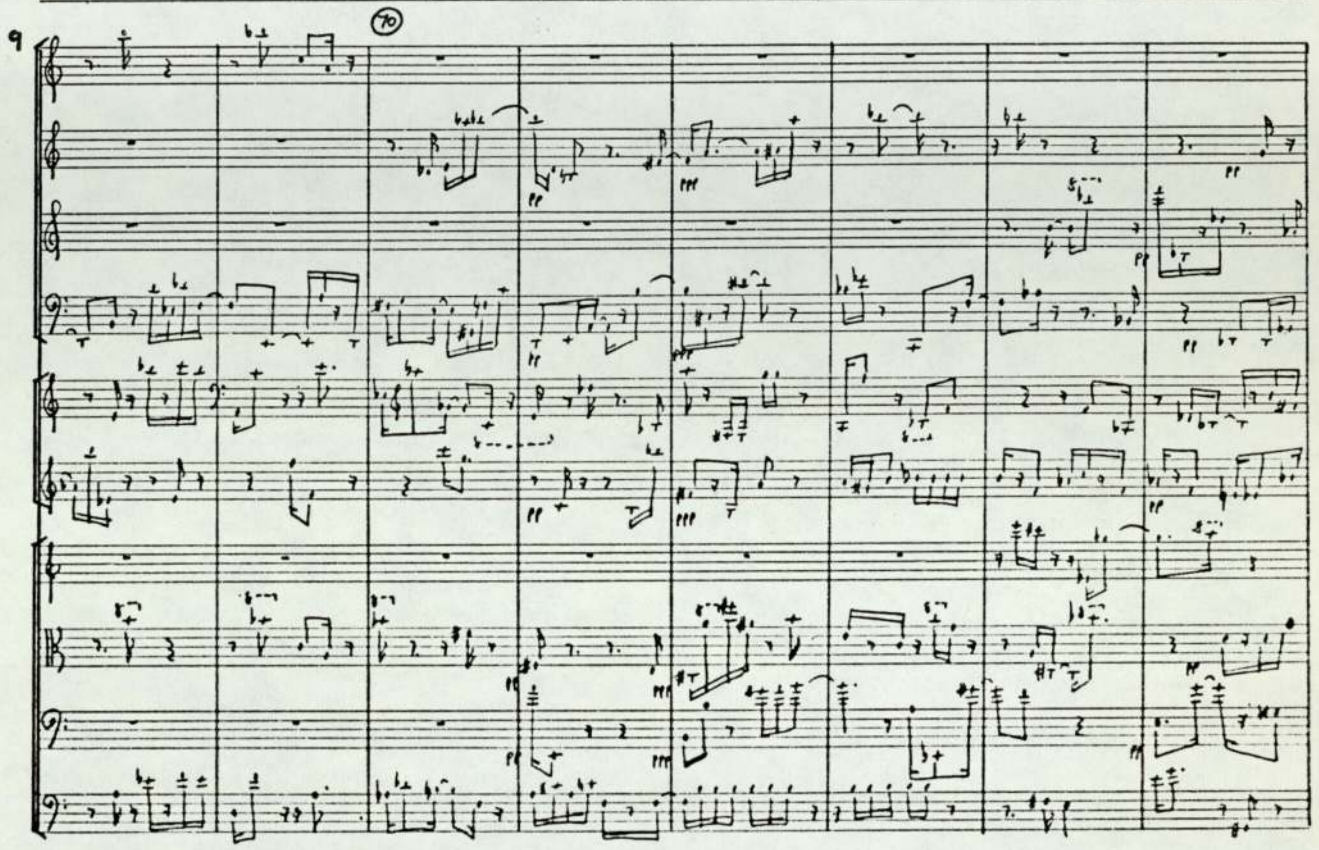
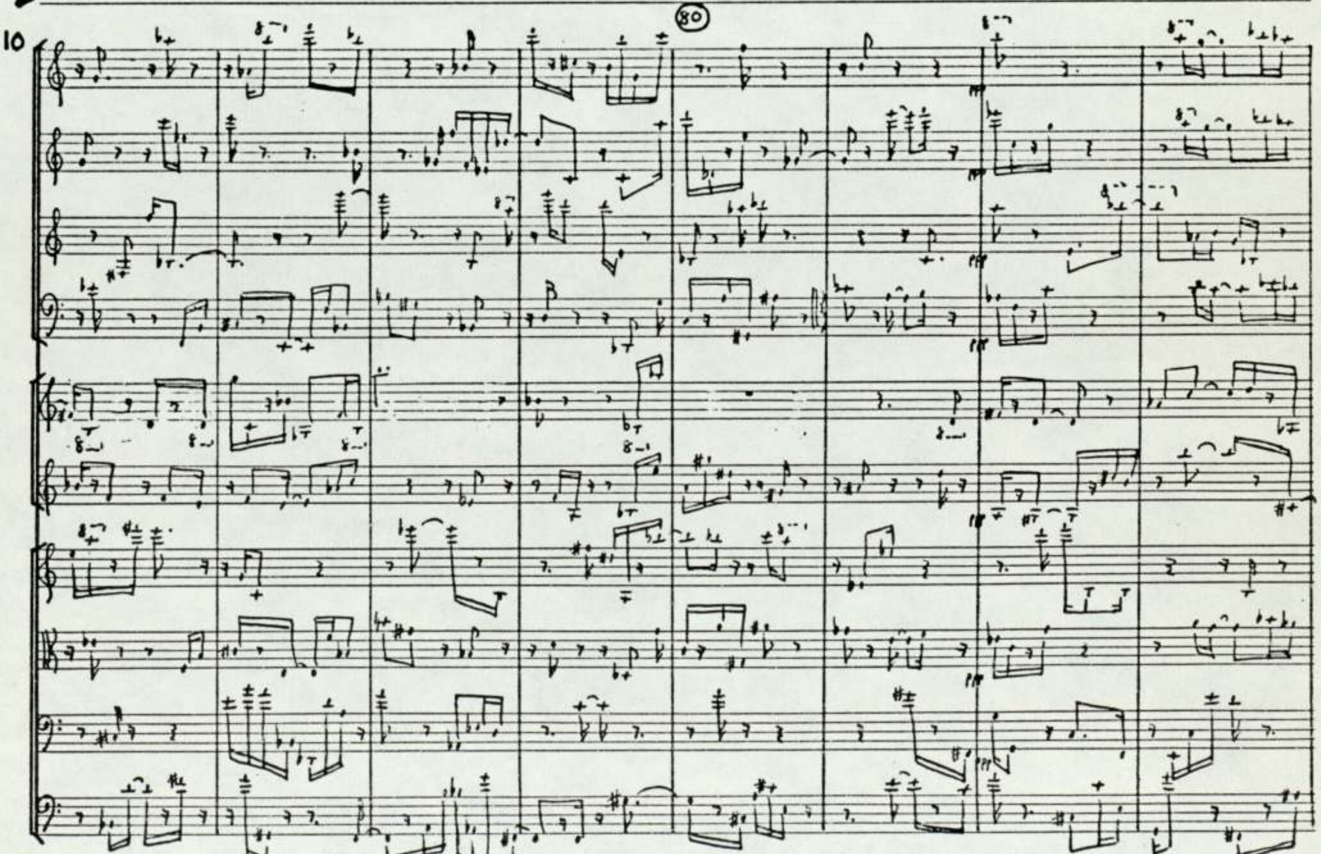


5
mp
mf
40
Pizz

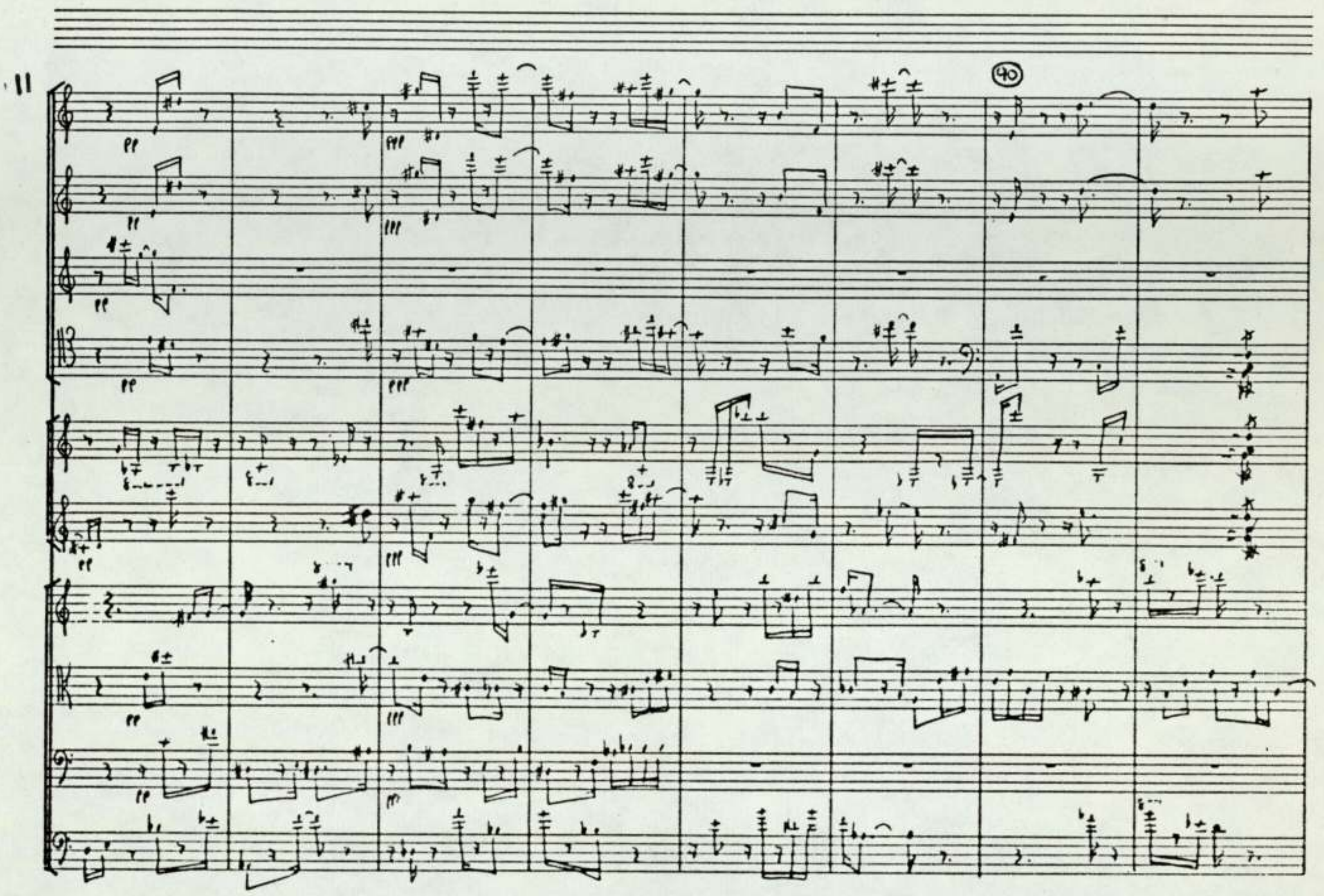


6
50
mp
Arco
p
Pizz

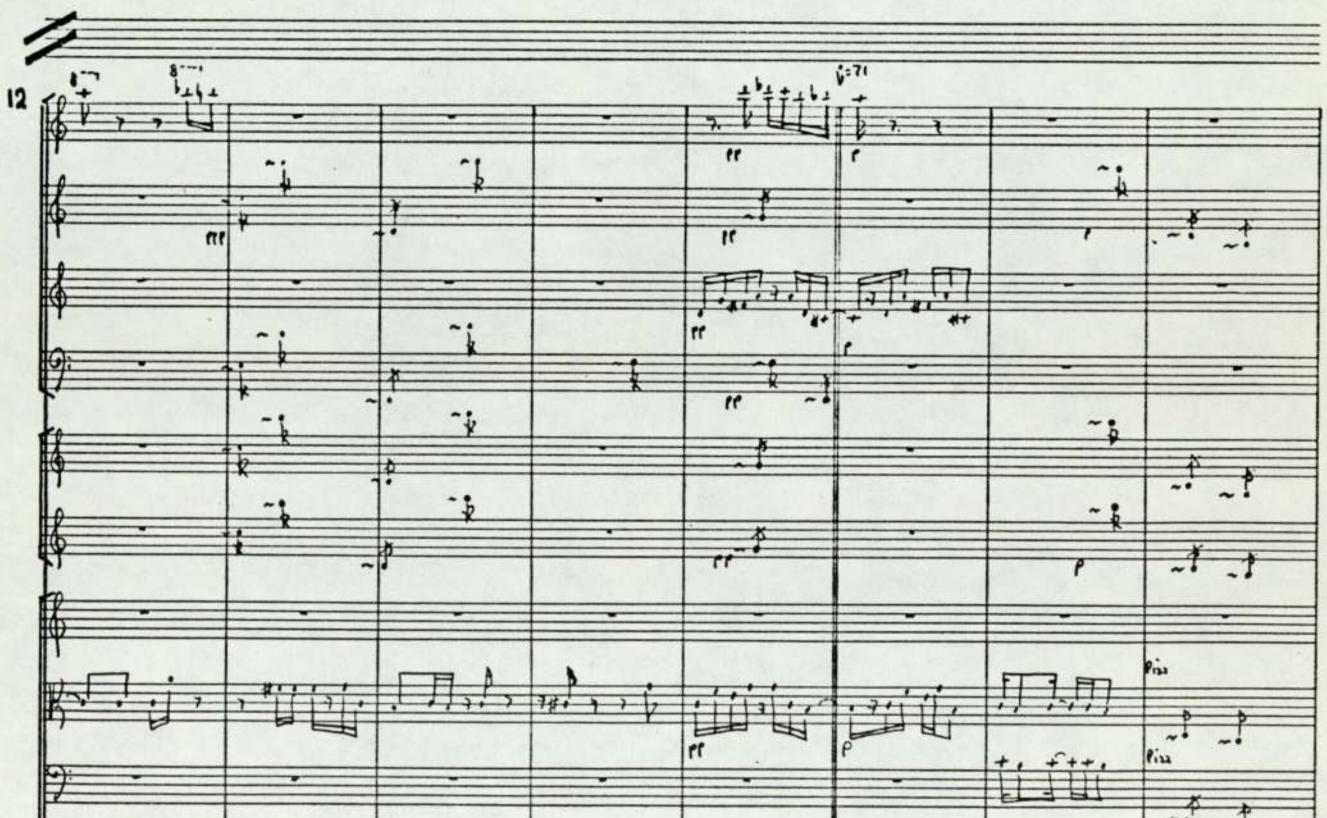

7
Arco
60
8

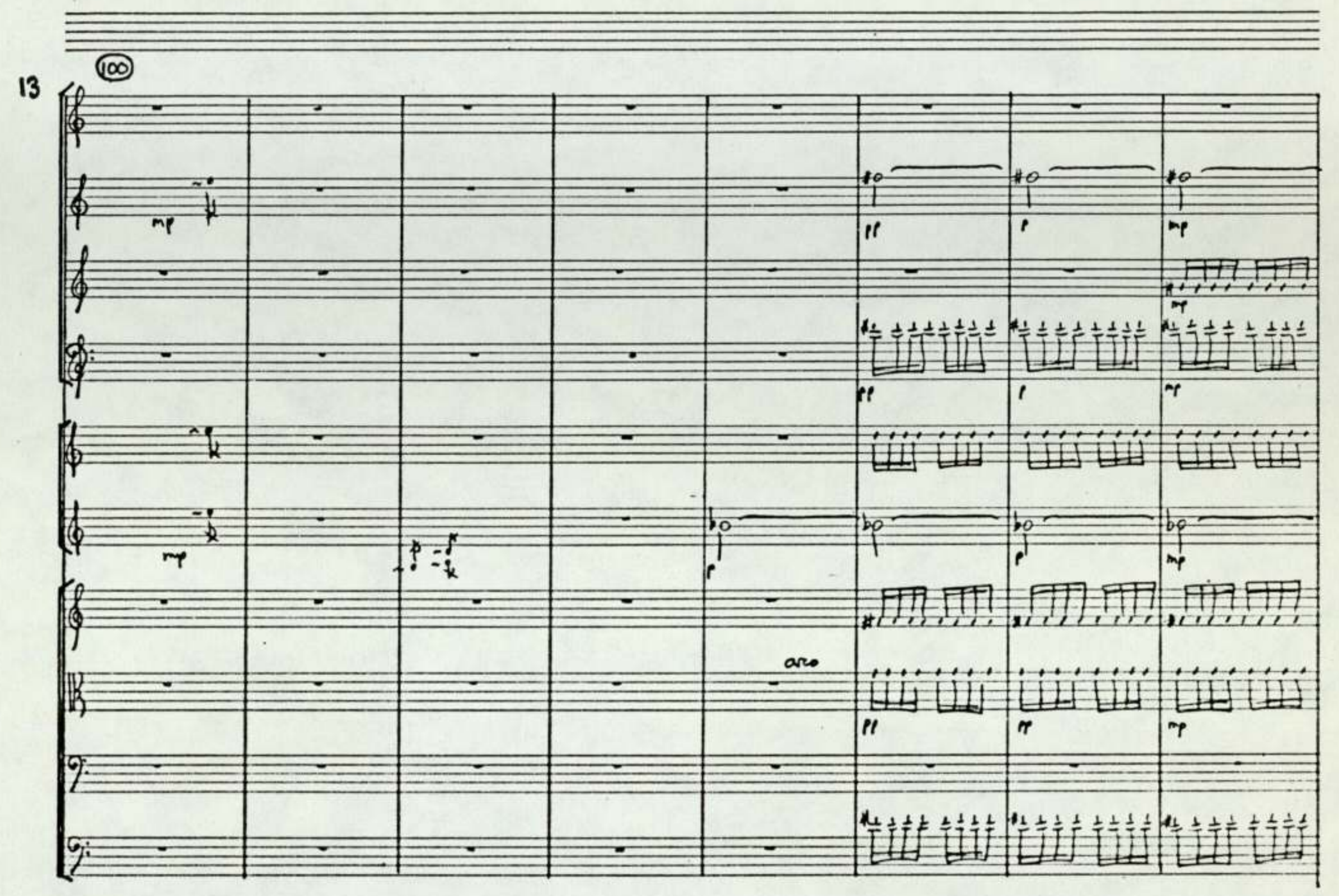
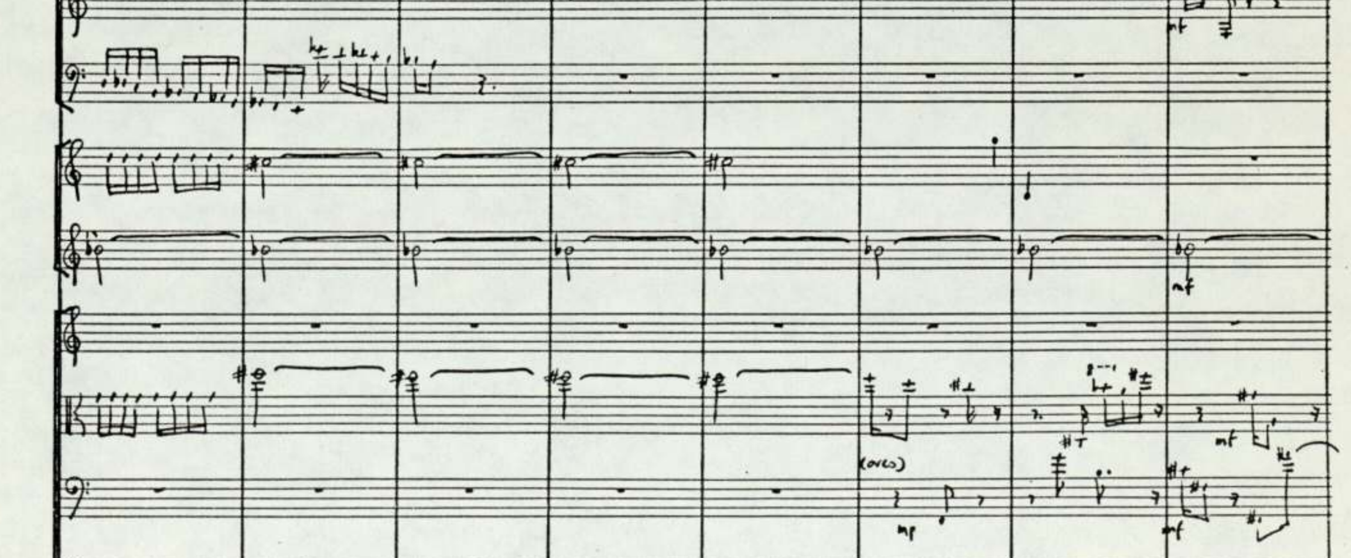

9
70

10
80

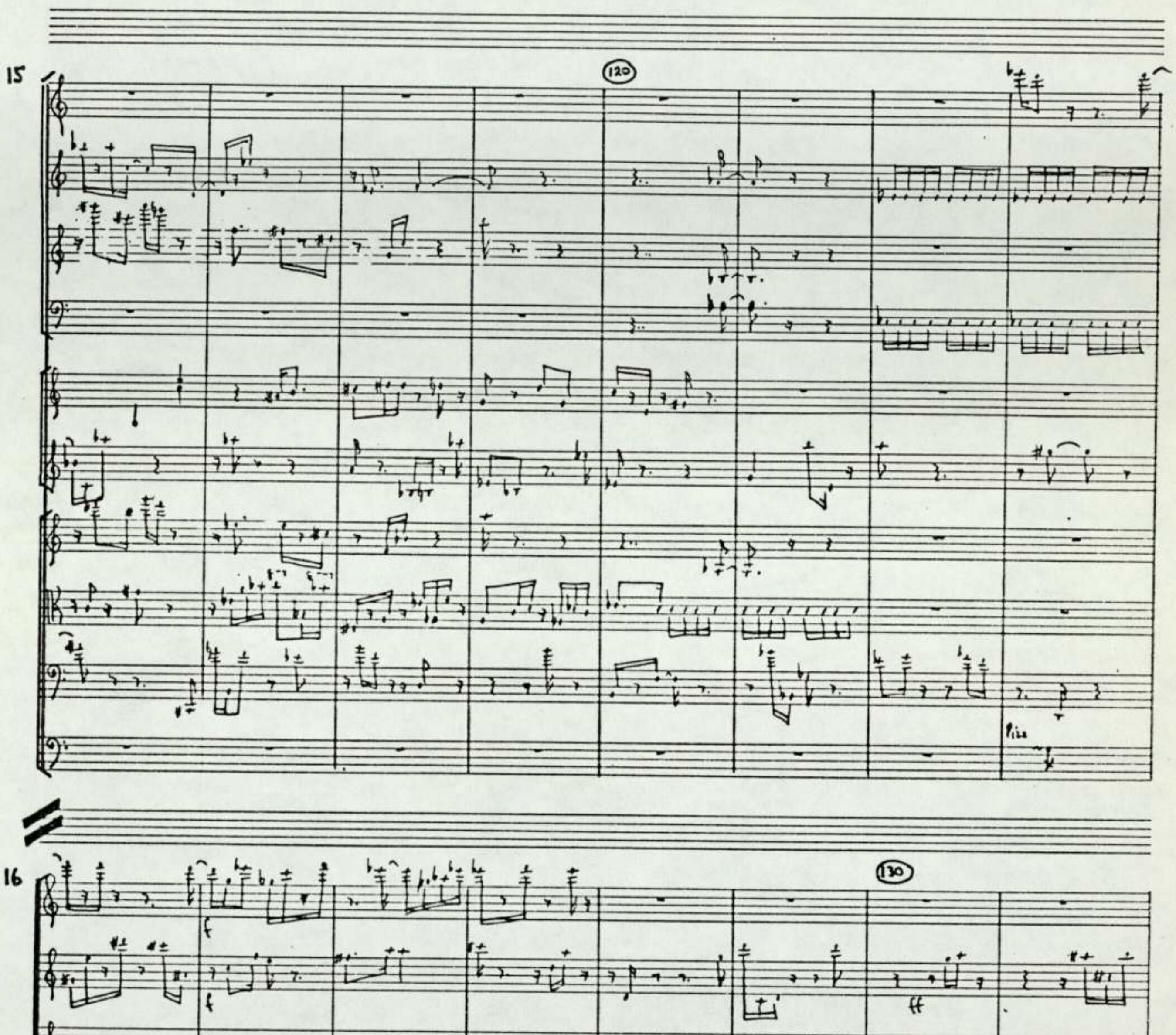
11

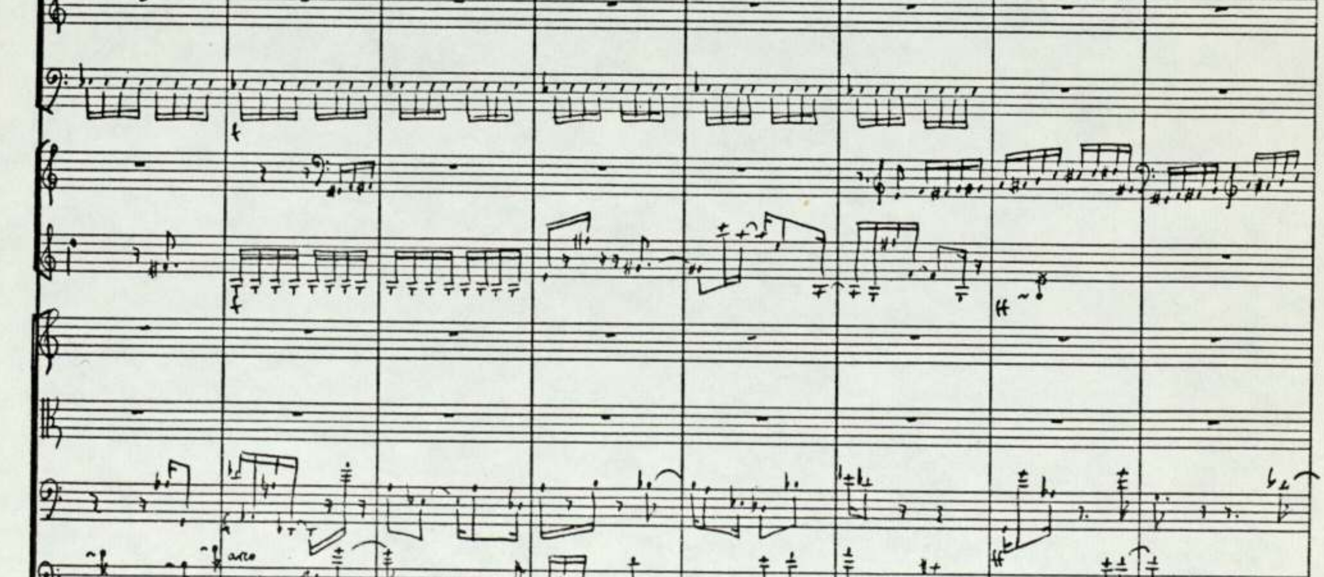


12

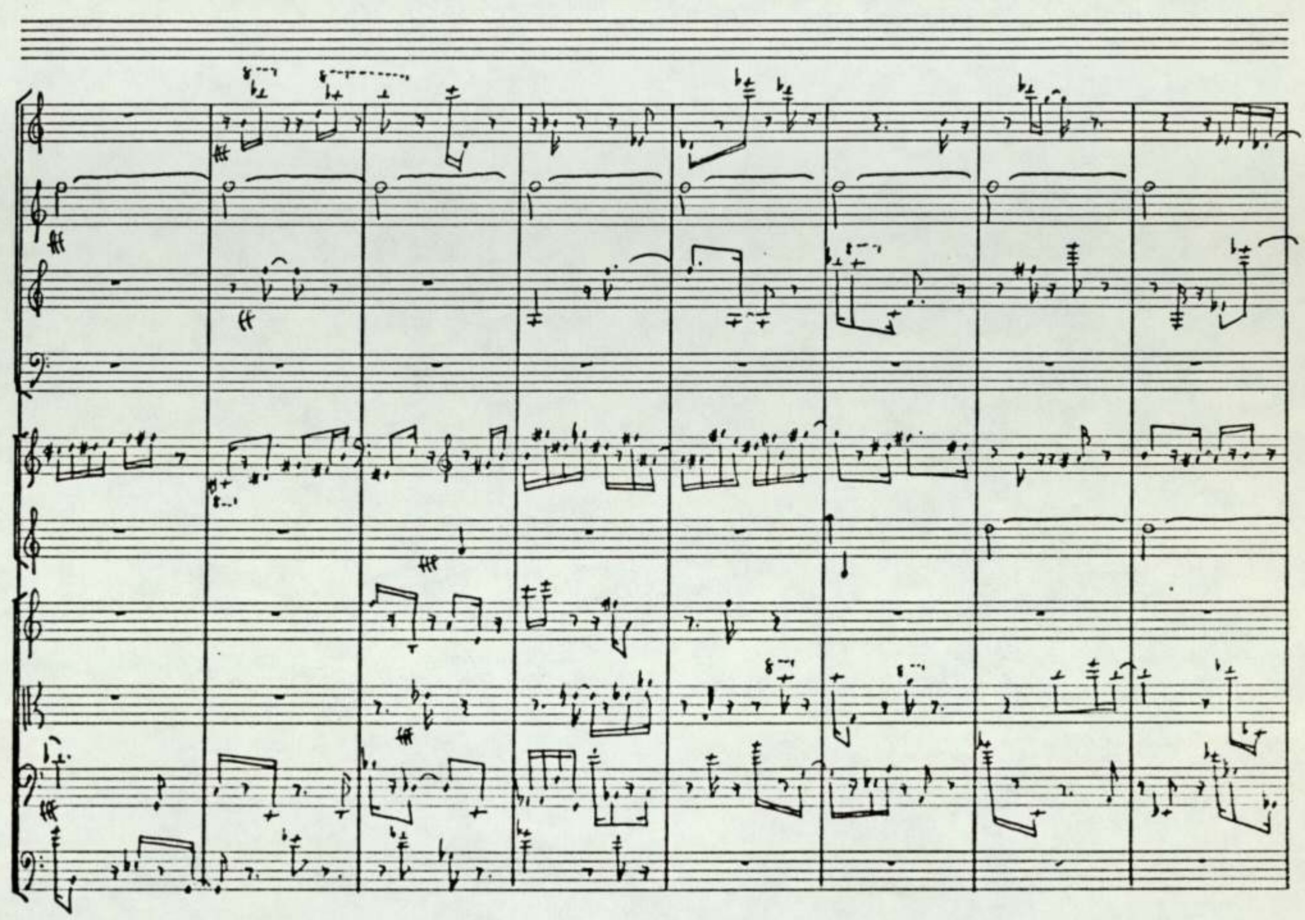



13
100
mp
mp
pp
p
mp
mp
pp
p
mp
mp
p
p
mp
arco
pp
mp
mp

14
110
mf
mf
mf
(arco)
mp
mf
mf

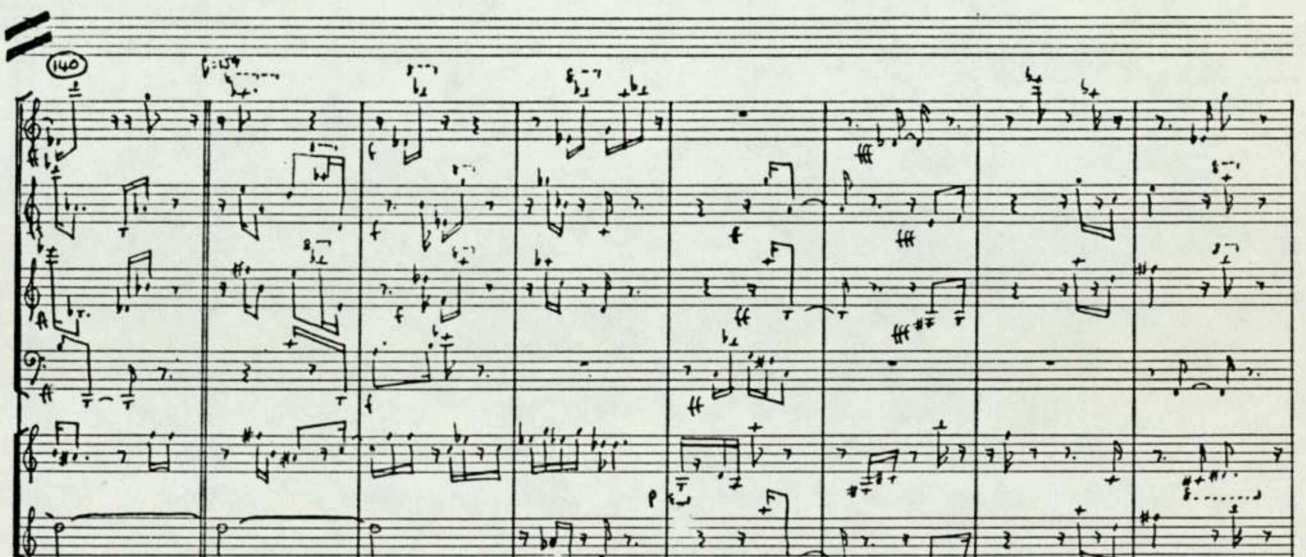

15
120
pizz


16
130
f
arco

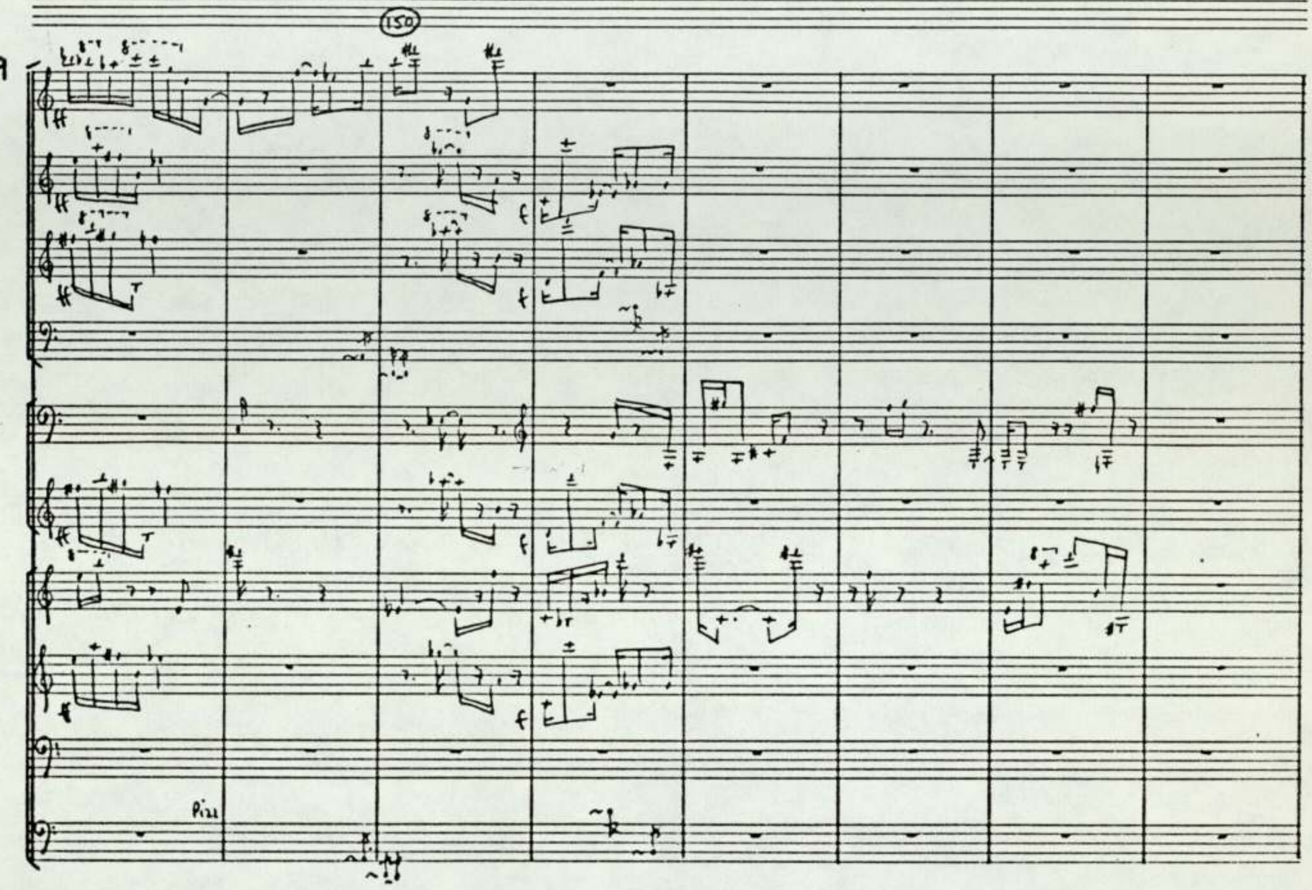
17



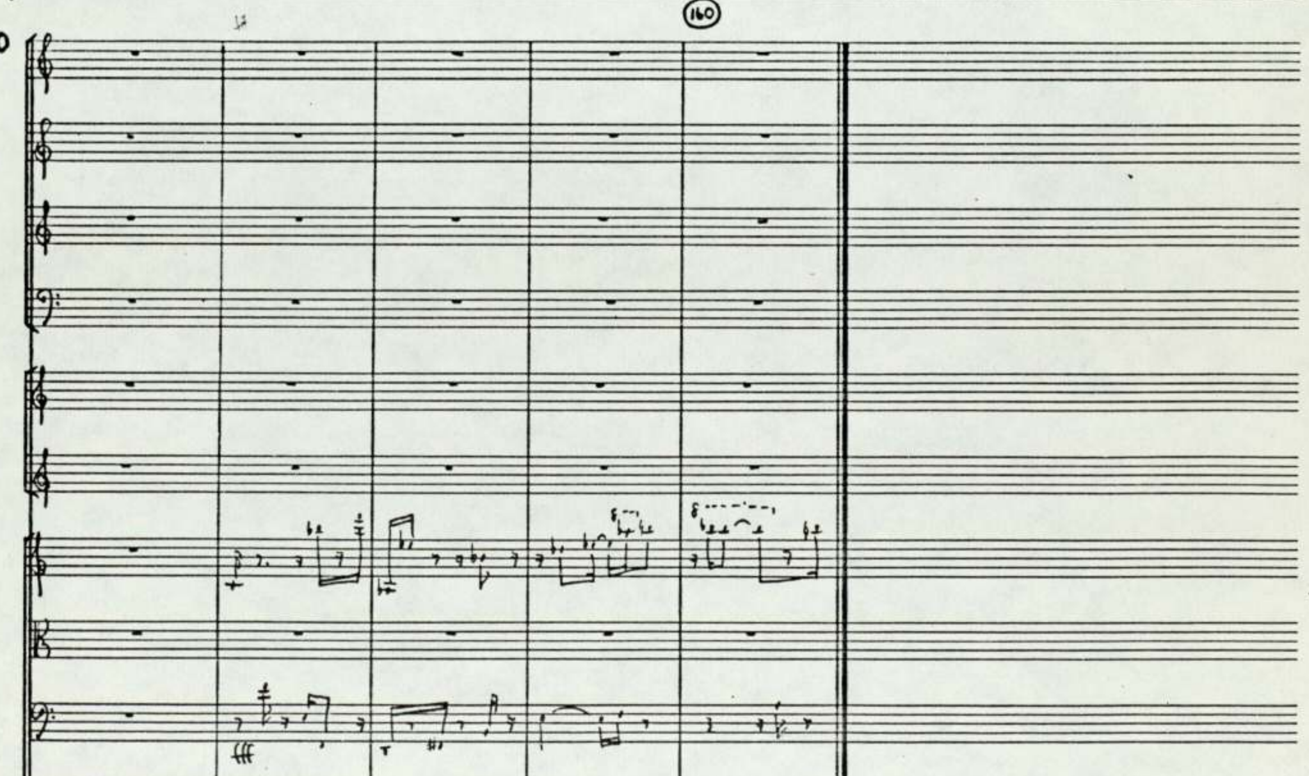
18



19



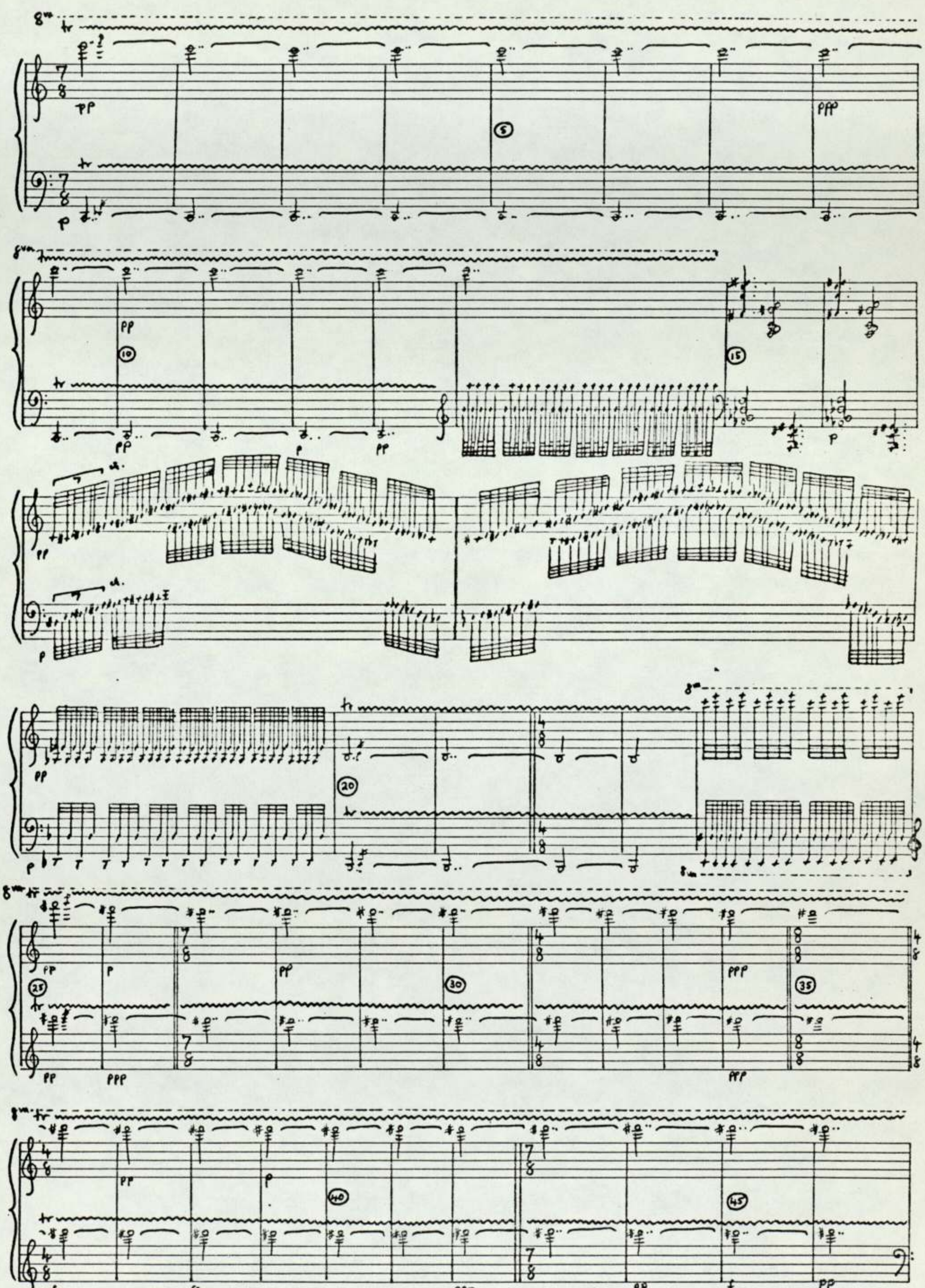
20

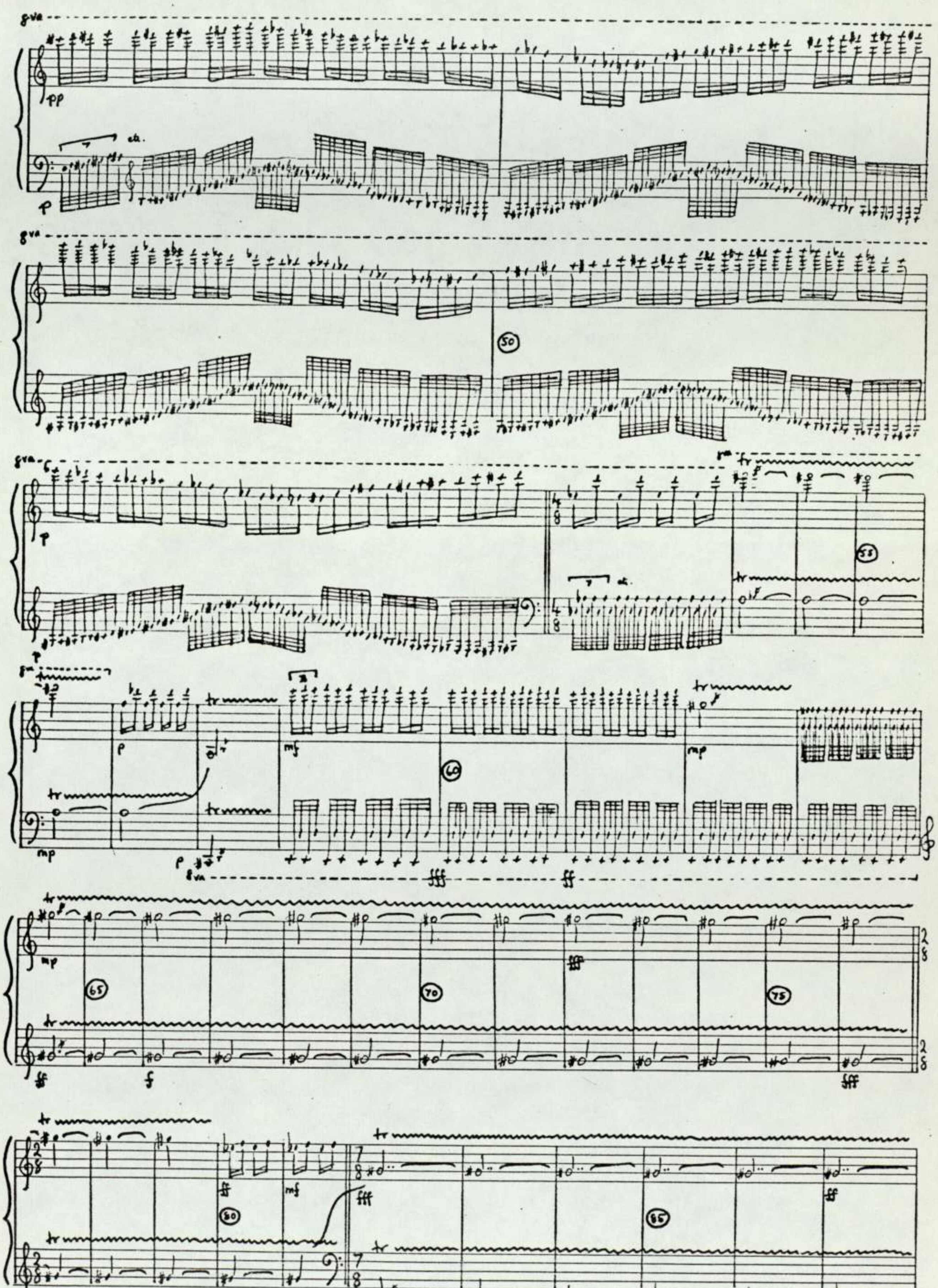


IIf *pianocomp*

This extract shows the first 87 bars of a piece for piano, (tentatively titled: *sonaise*).

PIANOCOMP




8va
pp
p
etc.
50
8va
p
p
tr
55
tr
mp
p
mf
60
fff
ff
tr
mp
p
8va
tr
mp
65
70
75
fff
ff
f
fff
tr
ff
mf
80
fff
85
ff
8va
mp

REFERENCES

A. M. ARTHURS,

Probability Theory (Routledge and Kegan Paul, 1965).

PIERRE BARBAUD,

Initiation a la Composition Musicale Automatique (Dunod, 1966).

JONATHAN BENTHALL,

Science and Technology in Art Today (Thames and Hudson).

WALLACE BERRY,

Structural Functions in Music (Prentice Hall, 1977).

PIERRE BOULEZ,

Boulez on Music Today (transl. Susan Bradstow and Richard Rodney Bennett) (Faber, 1971).

DONALD BYRD,

'An Integrated Music Software System',

The Computer Music Journal vol 1 no 2 (1977).

JOHN CAGE,

Silence (M. I. T., 1961).

DAVID CHARLES,
Entracte: '"Formal" or "Informal" Music.' ,
Contemporary Music in Europe, ed Paul Lang.

JOHN CLOUGH,
'TEMPO A Composer's Programming Language',
Perspectives of New Music (Fall-Winter 1970).

JONATHAN COTT,
Conversations with Stockhausen (Picador).

FLEURET,
'Xenakis - A Music for the Future',
Music and Musicians (April 1972).

ROBERT E. FRANKEL, STANLEY J. ROSENCHEIN, STEPHEN S. SMOLIAR,
'A LISP-based System for the Study of Schenkerian Analysis',
Computers and the Humanities vol 10 no 1 (Jan.Feb 1976).

STANLEY GILL,
'A Technique for the Composition of Music in a Computer',
The Computer Journal vol 6 no 2 (Jul 1963) pp 129 - 133 .

HAMMERSLEY and HANDSCOMB,
Monte Carlo Methods (Methuen).

HILLER and ISAACSON,
Experimental Music (McGraw-Hill, 1959).

LEJAREN HILLER,
'Music Composed with Computers - A Historical Survey',
The Computer and Music ed Lincoln .

'Programming the I Ching Oracle',
Computer Studies in the Humanities (Oct 1970).

HUBERT HOWE,
Electronic Music Synthesis.

ADRIAN JACK,
'Adrian Jack talks to Xenakis',
Music and Musicians vol 23 no 7 (April 1975).

KEVIN JONES,
'Bach and Handel', *Statistical Analysis of Recitative* (Appendix),
Undergraduate Project (University of York, 1973).

EDWARD D. KOBIN and THEODORE H. A. ASHFORD,
'A Solution to the Problems of Vertical Serialization',
Perspectives of New Music (Spring-Summer 1968) pp 119 - 124 .

PAUL LANG (editor),
Contemporary Music in Europe (Schirmer).

LINCOLN,

The Computer and Music (Cornell U.P., 1970).

'Uses of the Computer in Music Composition and Research',

Advances in Computers vol 12 (Academic Press, 1972).

J. D. LOMAX (editor),

Computers in the Creative Arts (The National Computing Centre Ltd., 1973).

H. C. LONGUET-HIGGINS and M. J. STEADMAN,

'On Interpreting Bach',

Machine Intelligence 6 no 15, ed Bernard Meltzer (Edinburgh U.P.).

JOHN LYONS,

Chomsky (Fontana).

MATOSSIAN,

'Calculating Composer',

The Observer (Sept 1972).

MAX V. MATHEWS,

The Technology of Computer Music (M. I. T., 1969).

YEHUDI MENHUIN,

Unfinished Journey (Macdonald and Jane's, London, 1977).

LEONARD B. MEYER,

Music the Arts and Ideas.

Explaining Music.

GEORGE A. MILLER,

The Psychology of Communication (Penguin, 1967).

ABRAHAM MOLES,

Information Theory and Aesthetic Perception (Univ of Illinois Press, 1966).

ROBERT E. MUELLER,

The Science of Art (Rapp and Whiting).

T. N. NEMES,

Cybernetic Machines (Illife).

T. G. NEWMAN and P. L. ODELL,

The Generation of Random Variates (Griffin).

MICHAEL NYMAN,

Experimental Music (Studio Vista).

OLSON and BELAR,

'The Analysis of Stephen Foster Songs'

J. Acoustical Society of America vol 33 no 9 , pp 1163 - 1170 .

PAPWORTH,

'Computers and Change Ringing',

The Computer Journal vol 3 no 47 (1960).

PIERCE,

Symbols, Signals and Noise (Hutchinson, 1962).

'A Chance for Art',

Cybernetics, Art and Ideas ed Reichardt.

KARL POPPER,

The Open Society and Its Enemies, Vol II (Routledge and Kegan Paul, 1962).

STEVE REICH,

Writings about Music (New York U.P., 1974).

JASIA REICHARDT (editor),

Cybernetics, Art and Ideas (Studio Vista, 1971).

JASIA REICHARDT (editor),

Cybernetic Serendipity (Studio International, 1968).

JASIA REICHARDT,

The Computer in Art (Studio Vista, 1971).

ROSS ASHBY,

An Introduction to Cybernetics (Chapman and Hall, 1956; 1964).

MURRAY SCHAEFFER,

New Soundscape.

PIERRE SCHAEFFER,

La Music et les Ordinateurs (UNESCO).

SMITH BRINDLE,

The New Music (OUP).

STEPHEN S. SMOLIAR,

'Music Programs : An Approach to Music Theory through Computational Linguistics',

Journal of Music Theory vol 20 no 1 (Spring 1976).

ANTHONY STORR,

The Dynamics of Creation (Penguin).

STUCKENSCHMIDT,

Twentieth Century Music (World University Library).

ALAN SUTCLIFFE,

'Examples of the Use of Computers in Music',

Computers in the Creative Arts ed Lomax.

TRUAX,

'A communicational Approach to Computer Sound Programs'

Journal of Music Theory vol 20 . no 2 (Spring 1977) p. 258 .

VINTON (editor),
New Dictionary of Twentieth Century Music (Thames and Hudson)

VON FOERSTER and BEACHAMP,
Music by Computers (John Wiley 1969).

RICHARD WOLLHEIM,
Art and Its Objects (Penguin).

XENAKIS,
Formalised Music (Indiana University Press).

MICHAEL ZAPLITNY,
'Conversations with Iannis Xenakis',
Perspectives of New Music (Fall-Winter 1975).

ZARIPOV,
'Cybernetics and Music', translated in
Perspectives of New Music vol 7 no 2 (Spring-Summer 1969).