MP1: A COMPUTER PROGRAM FOR MUSIC COMPOSITION

MP1: A COMPUTER PROGRAM FOR MUSIC COMPOSITION

MP1 is an algorithm translated into computer language which generates music. It supplants part of the composer's routine work by trying to simulate as closely as possible some of the operations which, presumably, take place in his mind. Although compared to such intricate mental processes, MP1 presently performs only a few simpler tasks, it is a starting point and a framework to which more complicated procedures can be added. MP1 is designed to be used for very different musical situations and by people with various degrees of musical training.

MP1 allows a wide range of choices concerning the style of the composition itself as well as the taking into account of patterns (melodic, rhythmic motives) which can either be fed in as a pre-existent material, or can be invented at execution time. MP1's output consists of a string of musical symbols which can be easily translated into a score for any combination of voices, instruments and electronic sources.

The actual composer is the user and he has to have a precise idea of the kind of music he wants. But the computer will perform part of the routine work and it is conceivable that a person with a limited musical background but with creative gifts, and a professional can attain comparable results when using MP1. However, they will both obtain only the kind of music defined by the available features of the algorithm. Obviously, it is of interest to add more sophisticated devices to MP1's present options: the existent frame allows almost anything desired by a composer to be incorporated if someone is willing to contribute the necessary formalizing and programing effort.

MP1 is based on the following assumptions:[1]

*All sounds belong to the same vector space. Implication: a composition can be reduced to a tensor.

*Equivalence relations modulo m can be defined on every element of the vector space's basis.

*Any continuous succession of sounds or of values for the same sound parameter can be described either as a Markov chain, as a stochastic distribution or as a random occurence.

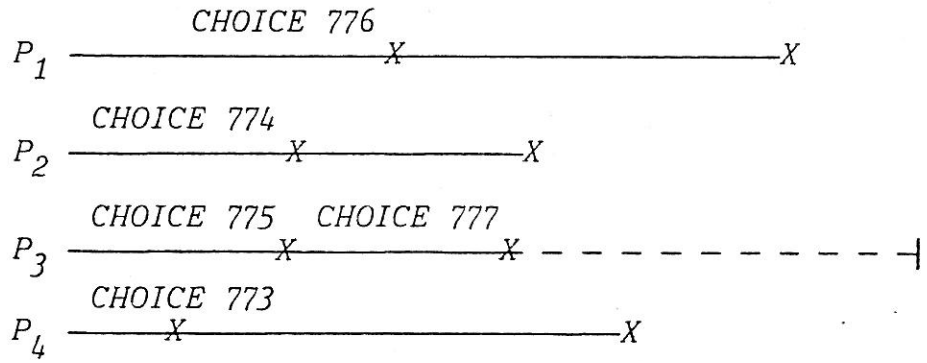
*To compose means to make decisions, to choose from a range of possibilities taking into account either a set of rules or a chance operation or both.

[1] Detailed understanding of the passages marked with a vertical line is not an absolute necessity for the user.

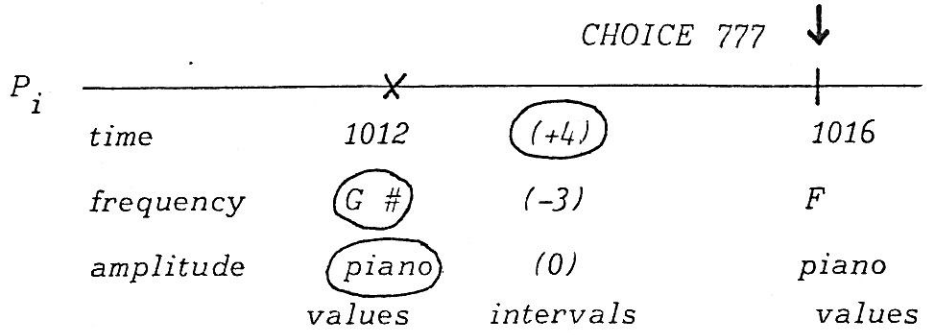
What MP1 mainly does is to choose particular values for every sound in the piece. The operation is executed successively in time (from left to right in the score), at all PARTS (= sound sources) simultaneously, in the very order they will be performed. To each sound corresponds a CHOICE or a complete execution of the program's main loop. It includes selections of values for each of the sound parameters (duration, frequency, amplitude, timbre, envelope) which are taken into account.

A composition contains a set of simultaneous, parallel PARTS, P_1, P_2, P_3,...,P_m, where 1, 2, 3,...m are ordinal numbers. Usually, the PARTS are numbered in the same order they appear in the score, from top, down: a flute, if present, will, probably, be P_1, while a contrabass is very likely to be P_m. However, the user is free to arrange the instruments in the score as he pleases.

Every CHOICE has an ordinal number which is the ordinal number of the sound, counting all sounds and silences already attributed to all PARTS, from the beginning of the piece.



MP1 takes into account in its computations both values and intervals (difference between values). With every CHOICE a value is assigned to each sound parameter. This is the actual sound characteristic as far as frequency, amplitude, timbre or envelope are concerned (G#, piano, for example). But for the time dimension, the assigned value defines the duration of the previous sound, the moment when the previous sound or silence stops and the current one starts. For this reason, the output printed at the end of a CHOICE relates to the values of the previous sound in the same PART and to the duration (time interval) defined by both the preceding time-value and the current one:



(circled values and intervals are printed following CHOICE 777)

Before beginning a new CHOICE, MP1 compares the last assigned values for the time parameter, at all PARTS, and finds out whose string of durations (last time-value) is smaller. Next CHOICE will be at the respective PART.

The CHOICE itself is a conversion of quantitative measurements (obtained through comparisons with both a desired situation and the previous decisions) into qualitative judgments. A formula of the type:

$$e^{-r(\varphi_i - \varphi_p)^2}$$

is used. Here r is a coefficient, φ_p is a value the composer prefers for a particular parameter and whose variation in time can be controlled through FUN (see page 9) and φ_i is a candidate value which is compared to φ_p. Because $e^{-0}=1$ and the value of e^{-n} decreases when n increases, the smaller the difference $\varphi_i - \varphi_p$, the more probable the assignment of φ_i.

When the probabilistic option is in effect, any decision is influenced by:

- *a preferred value for each parameter (see above)*
- *limits within which CHOICES can be made*
- *registers*
- *possible range (durations, pitches, dynamics, etc.) of the sound-sources*
- *degree of coherence between simultaneous sounds in different PARTS and between sequential sounds in the same PART*
- *degree of diversity allowed (how many discrete choices are available within the chosen limits)*

MP1 has a MEMORY where the last n CHOICES for each PART are stored. This is helpful for any option (including probabilistic) because any candidate value is compared to the last ones assigned to all PARTS. At the same time; it could become an elementary serial restriction.

After choosing a new time-value, MP1 decides if the duration starting there will be either a sound or a silence; a DENSITY factor is controlled by the user. The decision itself is made either by ascribing (through the previously mentioned mechanisms) a probability of occurrence to every possible value, or by directly picking-up the one with the biggest probability.

Another feature of MP1 is the optional use of PATTERNS. A PATTERN is defined as a succession of elements (at one or more sound parameters) which retain the same recognizable configuration throughout the piece. They may differ for each sound parameter or they may be interconnected in a sort of melodic sequence. At the present stage of MP1, the user feeds in the PATTERNS but it is conceivable that they also can be created at execution time. Although MP1 now considers only PATTERNS of intervals, this feature is extendable to value-PATTERNS, too.

When the PATTERN option is in effect, the following are available:

- The analysis of a given PATTERN

- The reproduction of a given PATTERN with variable degrees of fidelity. This influences the probability of occurrence of the various candidate values. If 100% accuracy is desired, all the other options are short-circuited

- The analysis of the previous CHOICES, stored in MEMORY, is used in order to see if any fragment of any PATTERN has been accidentally reproduced

- If a sequence of values belongs to more than one PATTERN, one of them is favored, usually the one with a better chance to be reproduced at more parameters

All data controlling any of MP1's features discussed above is subject to variance in time (can take various values at different moments of the piece). A way of controlling these variances is through a set of functions similar to the elementary waves available in an electronic studio and which are already familiar to most composers. Sine, square or exponential functions taken separately or combined can satisfactorily approximate almost any wave shape (evolution in time) a user may need.

Two modalities of beginning a piece are provided. One of them considers the beginning of the piece a privileged moment and feeds in a preconceived combination of sounds, like for any traditional western-music composition. The second modality considers the music as a continuous process which has begun before the first bar of the piece (the starting point is not more important than any other moment of the piece).

The printed output may have between three and five lines. The leftmost string of characters in the first line reads "CHOICE" and is followed by the ordinal number of the CHOICE. In the middle of the line is indicated the PART (which sound source the CHOICE refers to). The third and last string on the first line contains the numerical values for all sound parameters already assigned to the respective PART before the current CHOICE.

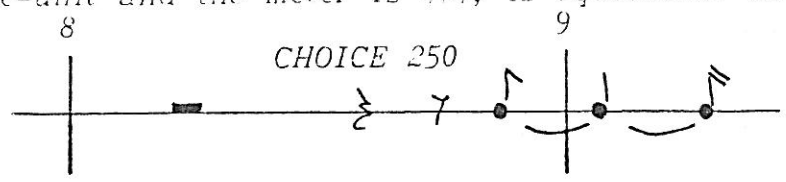
If the sound refered to was preceded by a silence, a special line is printed (line #2 in the figure). It indicates how many of the smallest time-units are between the last bar line and the beginning of the sound.

The next two lines display the main information about the sound. First string in the second of these two lines includes integers and/or fractions whose sum equals the duration of the sound. They are spelled in the way the duration should be written with regard to the meter and to the bar lines, although they all relate to only one sound. When a duration starts in one measure and ends in the next one after crossing the bar line, the first of these two lines of output (line #3 in the figure) will print: BAR followed by an ordinal number placed in such a way as to show which part of the duration will fall in one measure and which in the other. For instance:

```
(1)   CHOICE          250              PART          4 ...
(2)   BEGINS     14 UNITS AFTER BAR      8           ...
(3)          BAR          9                          ... etc.
(4)                  1/8    1/4    1/16             ...
```

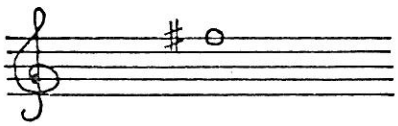
when 𝅘𝅥𝅯 *is the smallest time-unit and the meter is 4/4, is equivalent to:*

(The fourth sound source in the score).



For the next sound parameter (frequency) line #3 in the above figure will read: 8VA. Underneath, line #4 in the figure will appear the ordinal number of the octave and the name of the note, in english letter notation. The character # is used for sharps and > for flats; if the ordinal number of the octave is 0, no digit will be printed under 8VA. The way octaves are numbered is up to the user. However, a reasonable approach will be to consider the lowest sound ever to be used in the piece as the first sound in the lowest octave numbered as octave #0). If the lowest sound of the piece is:

then

8VA *will indicate:* 

1 F #

On the same printed line, the information which follows refers to the amplitude of the sound (pp, p, ff, etc.). If there is no change in amplitude between two adjacent sounds, nothing is printed.

The second line from the bottom reads: PATTERN followed by one string of digits for every sound parameter taken into account. There are as many digits in a string as the maximum possible number of PATTERNS for that sound parameter in the whole piece. A digit shows what place the current value occupies in the PATTERN. For instance,

(5) PATTERN 0 1 3 0 0 5 0 3 6 0 1 1 3 6 7

indicates that there are three sound parameters taken into account (three strings), the maximum number of PATTERNS in the piece is five (five digits in a string) and that the current value of the first parameter can be found as the first one in the second PATTERN and the third one in the third PATTERN, the current value of the second parameter can be found on the fifth place in the first PATTERN for this sound parameter, etc. This line of printed output is optional. More frequent is the indication:

(line 5 or 6) CONTINUATION 7

which informs the user that the current values of <u>all</u> sound parameters

belong to PATTERNS with the same ordinal number (7, in the example) and have the same place in all of them (probably, because they coincide in a melodic sequence).

MP1 is a cluster of interconnected and interchangeable subroutines, each of them accomplishing a well defined task. Here is a brief description of the principal subprograms (features) of MP1. There are, also, a number of service subroutines dealing with rather elementary operations (like producing a set of random numbers, finding the largest value in a set, writing equivalence relations modulo m, etc.), which will not be discussed.

MMAIN is the central coordinating system. It reads in all needed information except for DATA related to PATTERNS or to PRECONceived choices. It prints out the general information preceding the actual output like number of PARTS, sound parameters, length of the PATTERNS, how many PATTERNS are considered, length of the MEMORY, various coefficients, RANGES, and so on. Generally, they are self explanatory and useful mainly to the programer. In addition, the whole deck of DATA cards is printed, allowing a double check of the information the machine receives. No information related to PATTERns and to PRECONceived choices is printed by MMAIN. As choices are needed, it calls in the subroutines to be used, according to the options selected.

PATTER analyses the PATTERNS. It reads in the available PATTERNS and prints them out, and sets up a four dimensional matrix

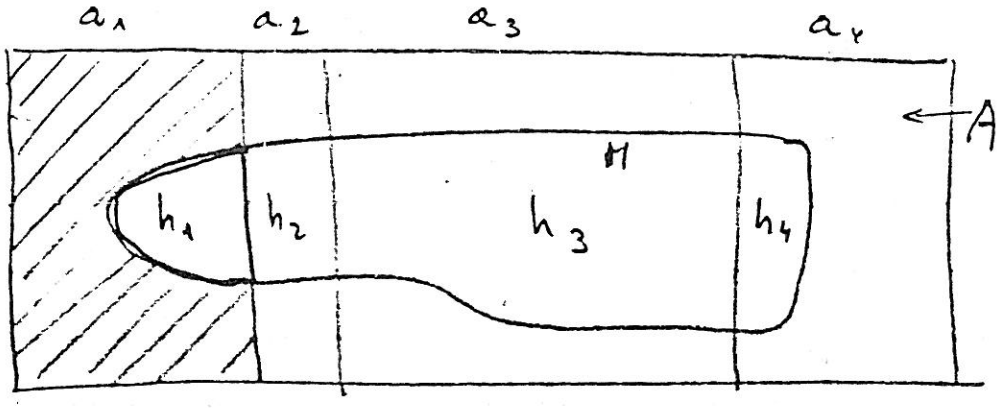
$$M(i,j,k,m)$$

where i is the number of sound parameters taken into account, j is the total number of PATTERNS, k is the degree of fidelity desired and m is the ordinal number of the value in the PATTERN (its place in the PATTERN).

START is an alternate to PRECON. It regards the piece as a process begun before the first note in the score. It assignes values to all parameters for the last n sounds or silences, at all PARTS, preceding the actual beginning of the piece, where n equals the length of the MEMORY. This is a chance operation and an arbitrary one; if the user needs a smoother transition to the real piece, the first CHOICE should be less than 0. This will allow time for the process of incorporating all influences that may come from options and choices of values, RANGES, PATTERNS, etc. decided by the user.

PRECON is an alternate to START. In accordance with western traditional composition, it reads in a set of preconceived values describing the first sounds of the piece; it also prints them out.

LIST enumerates all possible elements to choose a final value from, establishing a list of possible choices. A preferred value (see on page 3) for each member of the equivalence classes modulo m and a variety factor (the cardinal number of the list or how long the list can be) are given for both absolute values (moments in time, frequencies, amplitudes, etc.) and intervals (durations, frequency intervals, variation of amplitude, etc.). The list is trimmed through TRIM and SIEVES so as to contain only elements within the range of the sound sources involved and/or part of the modes or scales in effect. The list contains all values obtained both by using the preferred value plus the diversity factor and by adding intervals to the last one assigned.



$$p_{\bar{H}}(a_i) = \frac{p(a_i \cap \bar{H})}{p(\bar{H})} = \frac{p(a_i) - p(a_i \cap H)}{1 - p(H)} = \frac{p(a_i) - \frac{h_i}{A}}{1 - \frac{H}{A}} = \frac{MA() - \frac{Mx}{C(N)}}{1 - \frac{I8}{C(N)}}$$

$$= \frac{p \text{ from matrix} - \#\text{ of elem. of the } a_i \text{ type already chosen} / \text{Card. of the pattern}}{1 - \#\text{ of elem of any type already chosen} / \text{Card. of the pattern}}$$

TRIM rejects any value which is outside the possible range or which has already been included on the list through a different procedure.

FUTURE compares every element from the list to a desired situation (preferred value, preferred register or distance to the limits of the given range). It makes use of the exponential formula on page 3.

PASPRE compares every element from the list both to the values previously assigned to the same PART (and stored in MEMORY), to the simultaneous values assigned to the other PARTS and to a desired diversity factor. It also makes use of the exponential expression on page 3.

PATTY looks if an element from the list matches any element of any PATTERN and, if it does, it modifies the probability of occurrence of the element according to a variant of BAYES' Theorem:

$$p_s = \frac{p_k}{\sum_1^j p_k}$$

where

$$p_k = \frac{p_{(a_i)} - \frac{a_i(H)}{\Sigma a_i}}{1 - \frac{\Sigma h}{\Sigma a_i}}$$

and $\sum_1^j p_k$ *is the sum of all* p_k. *Also,* $p_{(a_i)}$ *is the probability of occurrence read from the* $M_{(i,j,k,m)}$ *matrix,* $a_i(H) = a_i \cap H$ *shows how many elements of the* a_i *type precede the current one in the PATTERN,* Σa_i *is the total number of* a_i *elements in the PATTERN (the cardinal number of the PATTERN),* Σh *shows how many elements of any type precede the current one in the PATTERN (its place). A few more corrections are applied as to increase the effectiveness of the expression and to make it more flexible in various situations: they distinguish if the reproduction of the PATTERN has started before the current choice or if this is the first element to be matched to an element of the PATTERN. The user can modify these coefficients through FORMP1. When 100% accuracy in reproducing the PATTERNS is desired, PATTY gives control to UPDATE, short-circuiting other operations and subroutines in the sequence.*

CHOOSE picks-up a value from those available on the list according to its probability of occurrence and a set of random numbers. This is actually when a CHOICE is made.

DENSIT determines if the duration assigned at the current CHOICE is a sound or a silence. Both a set of random numbers, the analysis of whether sounds or silences were last assigned in the other PARTS and a desired density are taken into account, in a feed-back type of mechanism.

UPDATE is called when a maximum fidelity in reproducing a PATTERN is needed. But, it calls itself (entry NORMAL), TRIM and, if the proposed value is rejected, command is returned to the normal sequence of subroutines which provide a lesser degree of fidelity. Also, it permutes the last chosen values in MEMORY so that the length of the MEMORY remains constant: the last CHOICE is always stored in the first location and at every CHOICE a value is "forgotten:"

Values stored:
Locations in MEMORY

a	b	c	d
I	II	III	IV

Value assigned at the current CHOICE: x

becomes:

x	a	b	c
I	II	III	IV

d was "forgotten"

The same thing happens to the indication that a duration is represented by a sound or by a silence.

LOOKIN determines if the last assigned value is part of any PATTERN and, if yes, of how many PATTERNS and what places it occupies in the PATTERNS.

WHICH intervenes after a CHOICE is completely finished. If the sound is part of more than one PATTERN and the degree of fidelity is bigger than a certain limit, it chooses for all parameters one PATTERN the sound will be considered to be part of. This will be helpful for PATTY, LOOKIN or UPDATE at the next CHOICE. Favored are PATTERNS which coincide at more parameters and in which the current value has a smaller ordinal number (is placed closer to the beginning of the PATTERN).

FUN provides the functions and constants governing the structure of the piece and their variations in time.

TYPLOO controls the lines of output as described on pages 5 to 7.

AUXIL is an auxiliary program to MP1 which prepares the MMAIN program. It reads in part of the DATA and copies the appropriate declaration and dimension statements directly on the file where MMAIN is stored.

MP1 is a collection of quite abstract operations based on a number of very general musical facts. All numeric or concrete information, from coefficients to symbols for the printed output, are fed in through DATA cards punched according to FORMP1, a form to be filled in by the user. For this reason, all DATA can very easily be modified. New subroutines can be added to MP1 at any time, performing different operations without disturbing the basic frame, in the same way a human being can increase through training the number of musical procedures he employs. Also, it is easy to use MP1 for non-musical situations if its fundamental operations are considered to relate to other domains.