5

COMPUTER PROGRAMMES FOR STOCHASTIC MUSIC ANALYSIS

Music analysis with Information Theory requires the processing of large quantities of information and a computer is an ideal medium for this task. In fact, when large compositions are comprehensively analysed a computer is indispensable and has some important benefits:

1. large quantities of information may be stored on a variety of storage mediums which may in turn be used with different systems and software packages;
2. access to, and accurate manipulation of information is greatly enhanced;
3. representation of information and processed data may be extensively varied by using, among others, numeric, descriptive, and graphic formats;
4. provided that the application programmes work properly and the information fed into the computer is correct, accurate calculations and results are ensured;
5. results are relatively free of subjective information, barring those that are incorporated into the programme; and
6. provided that the correct information is fed into the computer, experiments will be repeatable, even by different operators. In other words because of the fixed methodology that the programmes apply, the results should always be the same.

For the reasons mentioned above the author used computers and computer software extensively to collect and process information. All the programmes used for the analyses, calculations and presentation of the entropy values described in this thesis were designed and developed by the candidate. Although the initial prototypes and test programmes were written in **Pascal,** the programming languages used for the final versions of these programmes are a combination of **C** and **Visual Basic**®[1]

The programmes were written with specific criteria in mind:

1. Entry of music data should be as uncomplicated as possible. Preferably by means of a graphical interface with a pointing device—the use of a keyboard for data entry is more prone to errors.
2. The programme should be able to identify as many errors as possible. For example, excessive note values in a bar.
3. Errors in the database should be easy to rectify and by means of the same graphical interface as the entry phase.
4. The database should be able to contain the 'raw' information of the music as well as the results of the calculation so that these need not be re-entered or recalculated.
5. Results of the entropy calculations should be presented in a number of possible ways, i.e. numerical or graphical.

Although not specifically designed for entropy analysis, many software packages are currently available that could be used to do basic entropy calculations. However, although these programmes are sufficient to do the necessary calculations, the basic music information needs to be entered in numeric or alphabetic format—this is very time-consuming and prone to errors. The most useful type of programme to use for this purpose is one of the many available spreadsheet programmes, while some of the database programmes on the market also include the necessary mathematical functions. Two useful packages that were experimented with are EXCEL®, a spreadsheet programme which runs under WINDOWS®, and which has numerous functions suitable for calculating entropy values, and ACCESS®,[2] a database programme which is capable of interacting with EXCEL®, but which could also be used separately.

5.1 The music analysis programme

The programme developed for this research essentially comprises three separate programmes. Although the three programmes are capable of working independently they were designed to work to-

[1] VISUAL BASIC is a new generation of the older programming language, *Basic*. This version is predominantly 'event driven' and relies to a large extent on graphical elements. VISUAL BASIC is published by Microsoft.

[2] Both ACCESS and EXCEL are trademarks of Microsoft.

gether and to interact seamlessly. In other words each of the programmes is capable of working by itself without requiring that the other two programmes are loaded. The three programmes are linked by a number of databases which should be mutually available. The databases are:

Criteria database

This database is transparent[3] and not editable. It mainly contains all the information required to interpret the notation and in reality consists of various smaller databases.

Composition information

The identifying information of each composition is stored in this database. The fields[4] contained in this database are:

Composition identification number

Title of Composition

Composer's name

Date of composition

Composition category (defined by programmer)

Additional Information (for comments and other information)

Music Information.

All the music information for each of the compositions is stored in this database. The most important fields for the purpose of this research are (a large number of additional fields contain additional information used for other applications such as printing):

Composition identification number

Entry identification

Bar number

Pitch representation (numerical/alphabetical format: octave, pitch name, accidental)

Pitch number (allowing for octave range and accidentals)

Note value (note length expressed as a function of 128)

---

3 A transparent program function in the background and is not obvious to the operator.

4 Each field in a database represents a bit of information. A predetermined set of fields make up a record, while a group of records form the database.

Entropy Information

The results obtained by the entropy analysis routines are added to this database which has the following fields:

Composition Identification Number

Order Number

Entropy - pitch

Entropy - note duration

Entropy - pitch and note duration

Entropy - pitch ratios

Entropy - intervals

It will be noted that the three last databases have one field in common, the *Composition identification number*. This field serves as a link between the three databases for any of the compositions and allows the databases to contain the information of more than one composition at a time. To work with the information of a specific composition, its identification number is entered. Only that composition's information is then made available and database management is thus more effective.

The manner in which the three programmes interact and mutually have access to the databases described above is shown in the schematic illustration below. The dotted line connecting the three databases represents the *composition identification number*. The direction of the arrows indicates whether a programme only has access to data or whether it can also modify data in a particular database. For example, the interface for data entry has access to and can modify both the *composition information* database as well as the *music information* database but can only access the *criteria and rules* database without changing; the entropy analysis programme only has access to the *composition information* database and the *music information* database but can only modify the *entropy information* database:

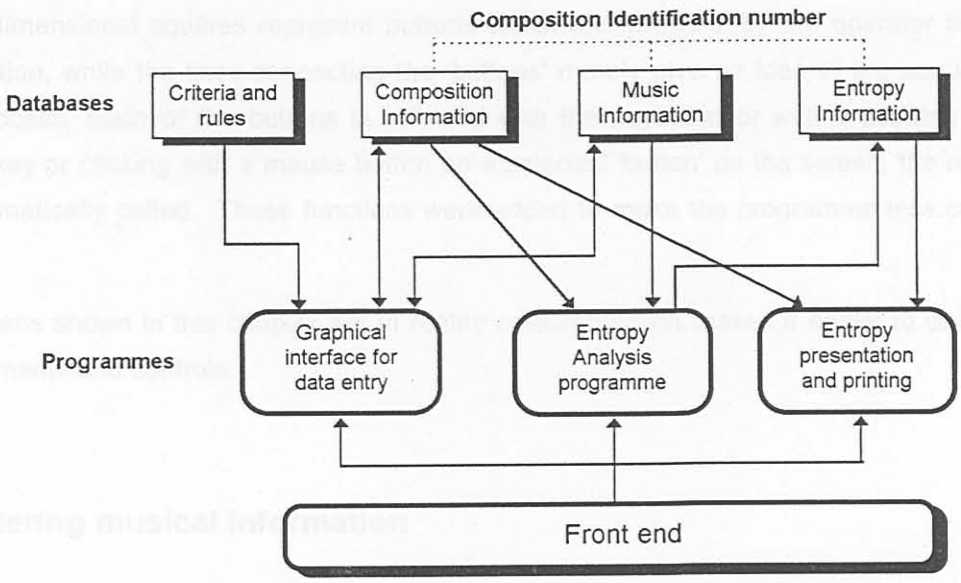
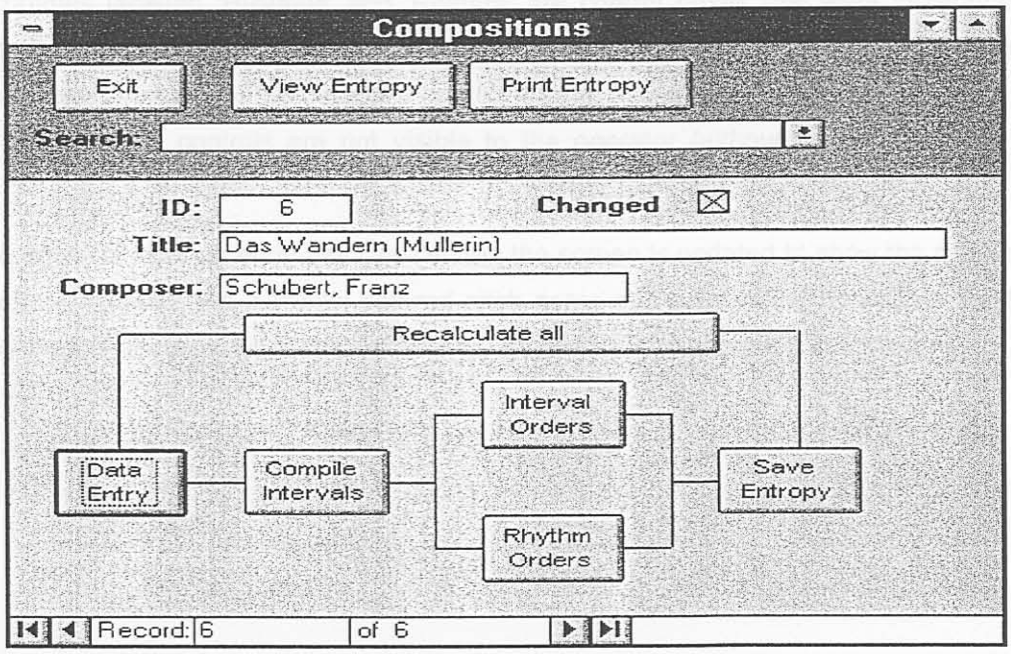

Composition Identification number
Databases
Criteria and rules
Composition Information
Music Information
Entropy Information
Programmes
Graphical interface for data entry
Entropy Analysis programme
Entropy presentation and printing
Front end

Figure 5-1. Databases and primary programmes for music analysis

The box marked 'Front end' represents a short controlling routine, or user-interface which calls[5] the three main programmes. This routine comprises the main screen of the programme and allows one to choose the composition for analysis or to enter a new composition. From this main screen the operator therefore controls the other programmes. The illustration below is a copy of the main screen which also shows why it is called a 'graphical interface'—many of the elements are represented by pictures some of which appear three-dimensional.



ID: 6 Changed ☒
Title: Das Wandern (Mullerin)
Composer: Schubert, Franz
Record: 6 of 6

Figure 5-2. Main screen of the analysis programme

5 Computer terminology used when one computer programme or routine, branches to another programme or routine which then takes control of the processor.

The three-dimensional squares represent buttons which are 'pressed' by the operator to select the required action, while the lines connecting the 'buttons' merely give an idea of the sequence of the analysis process. Each of the buttons is selected with the keyboard or with a pointing device. By pressing a key or clicking with a mouse button on a selected 'button' on the screen, the required routine is automatically called. These functions were added to make the programme less cumbersome to operate.

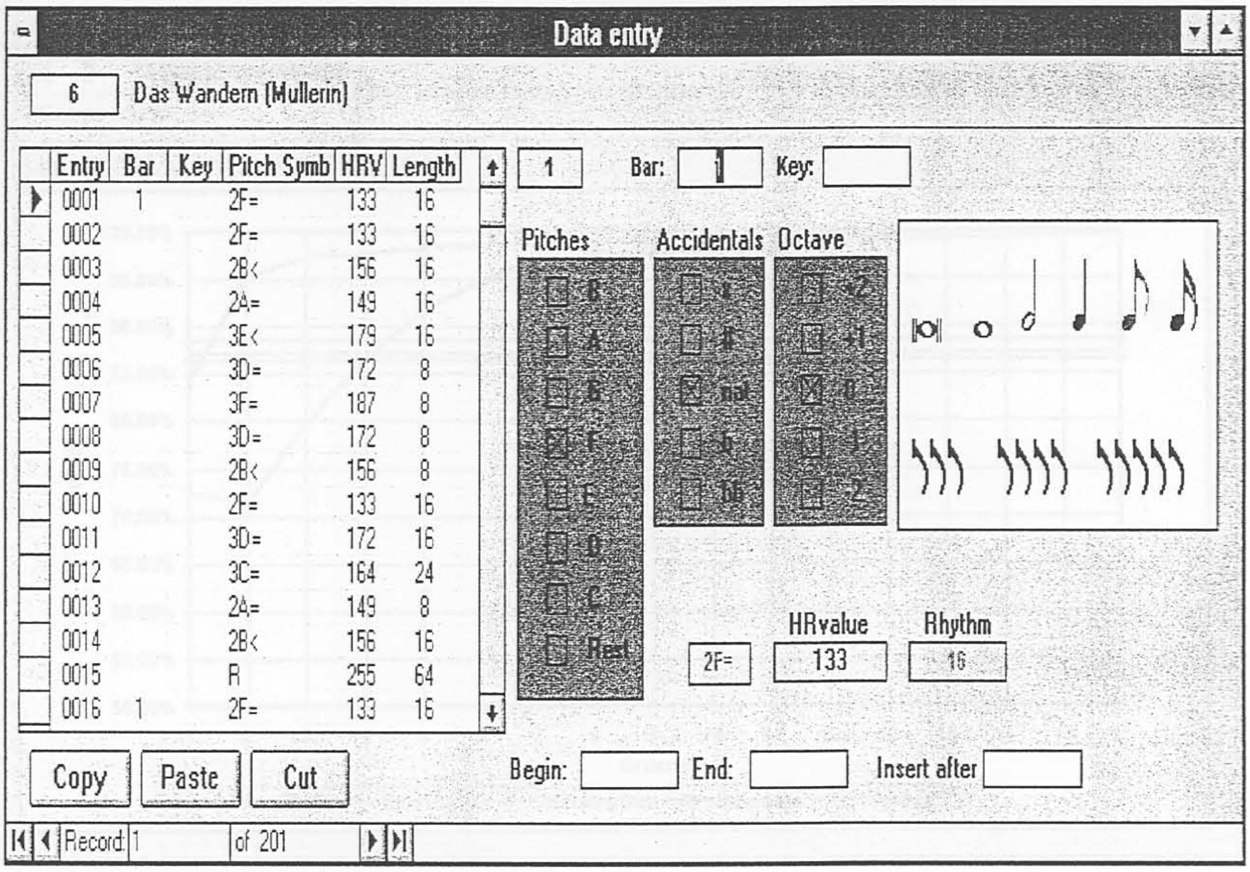
All the screens shown in this chapter are in reality coloured which makes it easier to distinguish the various elements and controls.

5.1.1 Entering musical information

Because most of the calculations are transparent and happen in the memory of the computer, only two elements of the programme, the entry of information and the presentation of calculation results are visible. The following illustration shows the screen that is used to enter the musical information into the database.

As with the main screen, a pointing device or the keyboard is used to select each of the functions on the screen, a copy of which is shown below. First a pitch is selected, then the octave number and the accidental, followed by the note value. The note value is selected by pointing at one of the values shown in the right hand box and the selected symbol then changes colour from black to red. The small rectangles labelled 'HRvalue' and 'Rhythm' are control boxes that were incorporated at the testing phase of the programme and shows the converted value for each of the pitches. The list to the left of the screen is a copy of the database in table format and is also a control. In the final version of the programme these controls are not visible to the operator (although still present in the background).

After each entry, or after an entry has been edited the screen is updated to show the number of notes that have been entered, the currently entered pitch name and note value, as well as a listing of the entries. By means of cursor keys, the entries may be perused for alteration or insertion.



Entry	Bar	Key	Pitch Symb	HRV	Length
0001	1		2F=	133	16
0002			2F=	133	16
0003			2B<	156	16
0004			2A=	149	16
0005			3E<	179	16
0006			3D=	172	8
0007			3F=	187	8
0008			3D=	172	8
0009			2B<	156	8
0010			2F=	133	16
0011			3D=	172	16
0012			3C=	164	24
0013			2A=	149	8
0014			2B<	156	16
0015			R	255	64
0016	1		2F=	133	16

Figure 5-3. Data entry screen

5.1.2 Displaying or printing the calculated information

Once all the calculations have been made the analysis programme is activated. These calculations are also transparent and the only indication that the program is running is a counter which indicates the programme's progress. The entropy that has been generated by the analysis programme may be displayed or printed in a variety of formats. If printed to the screen or to the printer, a choice may be made to print out the whole calculation process or just a summary in which only the final entropy values are printed. The following illustration shows a series of entropy values in graph form as it appears on the screen (in colour) and as it will look in printed form (in grey-scale). Details of the various parts of the graph and their interpretation is discussed in the next chapter. In addition to the graph, the screen below also shows the title of the composition, its identification number and the composer of the work:

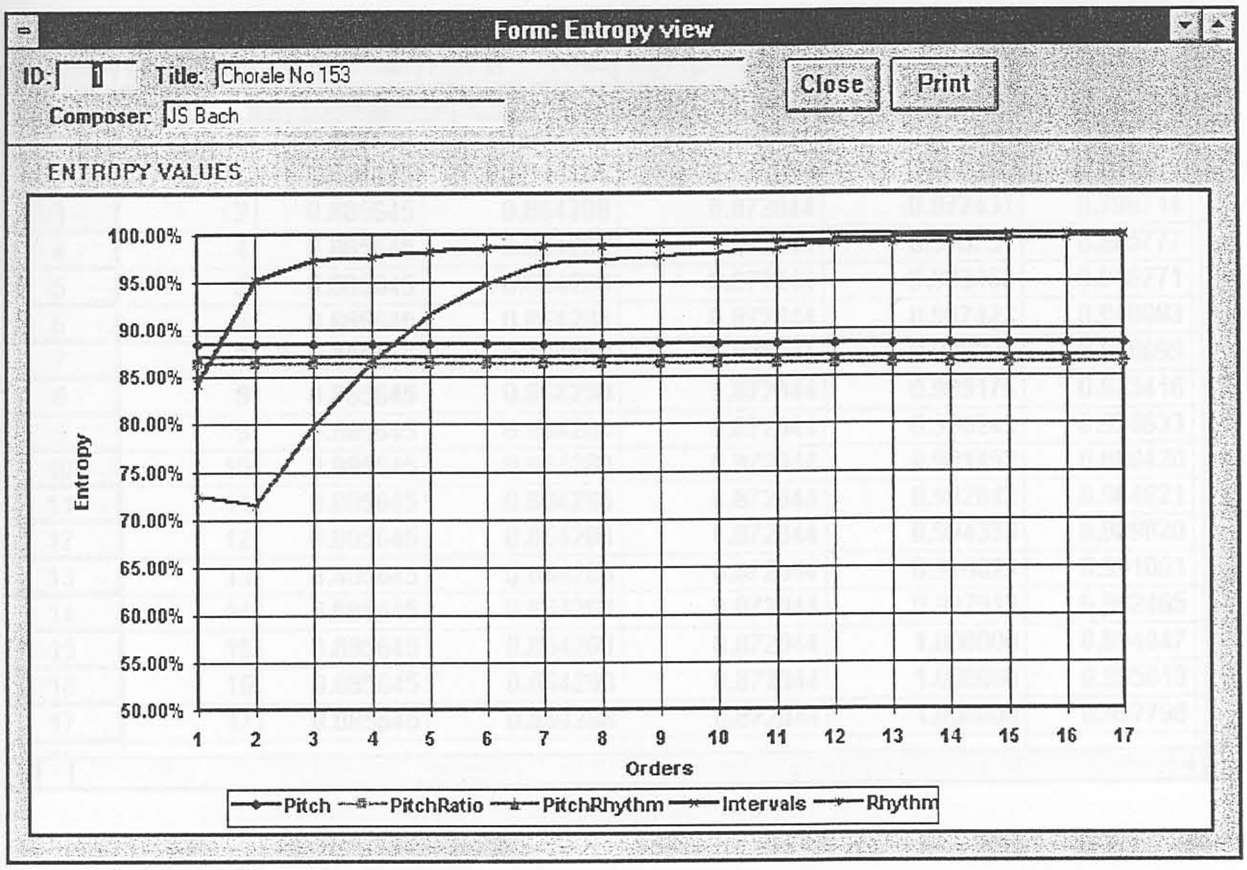
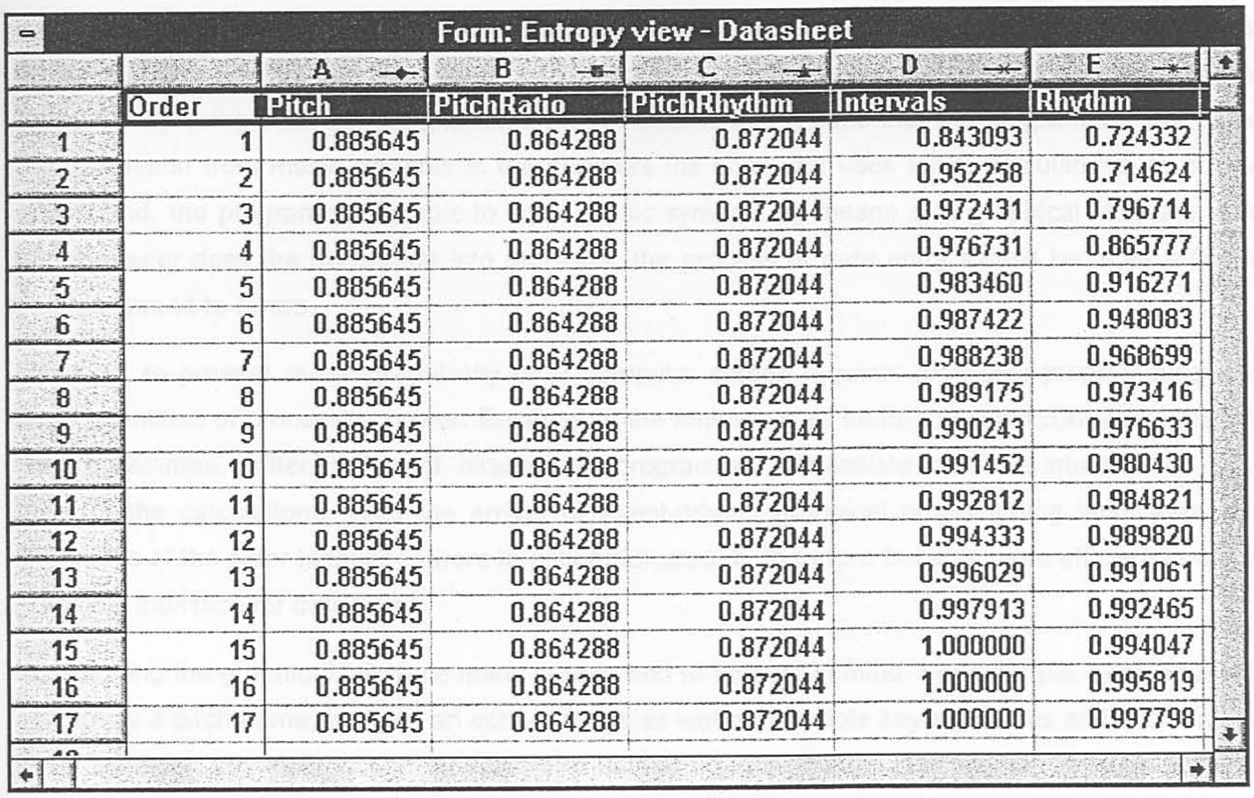


Figure 5-4. Entropy values in graph format

In the example above the straight horizontal lines represent the pitch, pitchratio and pitch-rhythm entropies. Since there is only one entropy value for each of these, the lines are drawn across the graph purely for the sake of clarity. The curved lines connect the respective entropy values for each of the orders of the stochastic processes of the intervals and the rhythm. The same formats mentioned above may be printed to a file on a storage medium such as a diskette or a hard disk in **ASCII** format. In this case the information may be incorporated in word-processing packages or, after suitable conversion, may be accessed by programmes such as spreadsheets from where it may be further manipulated.

The next illustration shows the same values as those of the previous graph. Note that the values in the first three columns are the same for all the order values. This ensures that a horizontal line for each specific value is drawn across the graph. Also note that the last three values (orders 15 - 17) for the Interval entropies are 100%, these are also automatically included by the programme to allow for the greater number of orders found in the Rhythm entropies:



		A	B	C	D	E
	Order	Pitch	PitchRatio	PitchRhythm	Intervals	Rhythm
1	1	0.885645	0.864288	0.872044	0.843093	0.724332
2	2	0.885645	0.864288	0.872044	0.952258	0.714624
3	3	0.885645	0.864288	0.872044	0.972431	0.796714
4	4	0.885645	0.864288	0.872044	0.976731	0.865777
5	5	0.885645	0.864288	0.872044	0.983460	0.916271
6	6	0.885645	0.864288	0.872044	0.987422	0.948083
7	7	0.885645	0.864288	0.872044	0.988238	0.968699
8	8	0.885645	0.864288	0.872044	0.989175	0.973416
9	9	0.885645	0.864288	0.872044	0.990243	0.976633
10	10	0.885645	0.864288	0.872044	0.991452	0.980430
11	11	0.885645	0.864288	0.872044	0.992812	0.984821
12	12	0.885645	0.864288	0.872044	0.994333	0.989820
13	13	0.885645	0.864288	0.872044	0.996029	0.991061
14	14	0.885645	0.864288	0.872044	0.997913	0.992465
15	15	0.885645	0.864288	0.872044	1.000000	0.994047
16	16	0.885645	0.864288	0.872044	1.000000	0.995819
17	17	0.885645	0.864288	0.872044	1.000000	0.997798

Figure 5-5. Entropy values in data-sheet view

The database programme that is used to gather the information and calculate the entropy values, is capable of exporting its information in a variety of formats to other commercially published software packages. The analysis results which appear in this dissertation were exported in a format suitable for importation to EXCEL, a spreadsheet programme. EXCEL allows further manipulation of the data for the creation of graphs and tables, and was used to incorporate the illustrations into this text (see chapter 6 and 7). The benefits of the exporting capability is that none of the data needs to be transported manually to word processors or other programmes, thereby excluding the possibility of any errors.

5.2 Aspects of computer programming for music applications

Some procedures and methods are explained here as they are common to all the computer programmes used for the analysis of this research.

To make the most effective use of the capabilities of a computer an important consideration when a score is translated into numerical information, is that a traditional music score represents sounds in a symbolic rather than in a logical and graphical manner. A single note represents various bits of information and the performer has to establish the pitch, accidental, volume and time value by interpreting it. Computer applications need to translate all this information into a workable system of numbers and the programmer needs to find a balance between what should be done manually, and what should be left to the computer. Obviously, if most of the calculations are done by the computer the likelihood of errors is reduced, while otherwise time-consuming calculations are much faster. Entering musical information by using pitch names and numbers for note length means that the programmer has to do the translation from music symbols to the numbers the computer uses for its calculations. If, on the other hand, the programmer is able to enter music symbols by means of a graphical interface, and the computer does the translation into numbers, the process of data entry should be more efficient and less prone to errors.

However, to present music graphically on a computer screen requires extensive programming and large quantities of processing power. Because of the limitations of earlier personal computers the initial programmes, written in **Pascal**, required the programmer to translate the notes into a format suitable for the calculations. With the arrival of event-driven, graphical programming languages, the limitations of the older languages were largely eradicated. It, therefore became more efficient to use a graphical interface for data entry.

In designing the graphical interface many criteria had to be kept in mind. For example, pitch does not only imply a pitch name, but also an octave range as well as possible key signatures and accidentals. For a musician it is obvious that the sharps or flats of a key signature affect all of the same notes in all octaves (unless cancelled by a natural sign). This is, however, not so obvious to a computer and a great deal of programming is required to arrive at a value that reflects the pitch accurately. A similar situation occurs when a note is altered within a bar by means of an accidental. Traditional convention has it that all subsequent occurrences of the same note should be altered with the proviso that the note does not occur in a different octave. Some editors and composers tend to ignore this convention which further complicates the process of computer translation. Each time a new note is encountered either the programmer or the computer has to allow for the implied characteristics of a pitch.

Traditional music theory and convention also imply a distinction between enharmonically equivalent notes. Most sophisticated music systems allow for thirty-five different pitch names, which accommodate the five different pitch configurations that accidentals can generate, for example, C^{bb}, C^{b}, C, $C^{\#}$, and C^{x}. As it should also be possible to apply information theory analysis to other systems, such as the dodecaphonic technique, and for that matter, also systems that may use quarter tones or smaller, the programmes had to be designed to allow for as many tuning systems as possible.

A similar situation of symbol interpretation arises concerning note duration and rhythm which, in traditional notation, is represented by specific notes and rest shapes in conjunction with a time signature and bar lines. The complexity of the interpretation increases with all additional symbols that affects the duration of a note. For example a fermata compounds the intricacies of computer interpretation of a note's duration. Note values in a score are symbolic representations of ratios of note lengths and relative accents but require extensive interpretation by a predetermined set of 'rules' incorporated in the conversion programme.

5.3 Computer representation of musical data

Most computer applications in music aim at incorporating as many elements of music as possible often seeking to provide for archaic and modern notation requirements as well. Usually the purpose of these programmes is to give an accurate graphic rendition of the score on the screen or in a printed format. Very often such programmes are also used as a tool for composition that requires it to be able to perform a composition with the help of synthesisers and other electronic equipment. This kind of application would require control over as many aspects of music as possible, including tone colours, attack, and envelope shape. There are a limited number of such programmes available e.g. MUSIC WRITER +®, SCORE®, PERSONAL COMPOSER®, and FINALE®. Research at many institutions on various applications of computers in music is being done continuously and a recent addition to the range of products is SIBELIUS®, currently one of the most powerful programmes available.

The complexity and dimensions of a software programme increases exponentially as the quantity of information demanded by it to work properly becomes greater. Especially the graphical representation of music demands intricate calculations and programming routines. Fortunately some of the facilities that the commercial programmes have, were not needed for the purpose of this research and much of the computer's resources could be used for the actual calculations. These functions—mainly MIDI compatibility—could nevertheless be added later.

Pitch and note values were the most important elements that needed careful consideration and the following sections mainly deal with the calculations of values for these elements respectively.

5.4 Pitch representation

5.4.1 Numerical pitch representation

Each of the pitches of a composition is represented by a single number. Since these pitch numbers are also used to calculate intervals between two adjacent pitches, the numbering system used for this purpose is designed in such a way that intervals are calculated by subtracting one pitch number from another. The pitch representation system developed for this research is based on a system formulated by Walter Hewlett and described by Ann K. Blombach. (ADCIS: 1989, pp. 50-58) Some minor alterations were made to make the number allocation more suitable to the requirements of the programmes. The Hewlett system allocates a specific and unique number to each letter name. As shown in Table 5-1, the value allocated to each of the pitch names was 2 greater than the Hewlett numbers. The reason for this change is that when a flat or a double flat (see Table 5-3) is added to the pitch name 'C', of which the Hewlett number is '0', the resulting value, in Hewlett numbering is -1 or -2 respectively. By increasing all the values by two, negative numbers are avoided:

Pitch Name	Pitch Number	Hewlett Number
C	2	0
D	10	8
E	18	16
F	25	23
G	33	31
A	41	39
B	49	47

Table 5-1. Hewlett pitch values

Accidentals are also given a numerical value. The third column of the table below illustrates the value of each accidental; the second column shows the symbols used for each accidental:

Accidentals	Computer symbols	Accidental values
double flat	<<	-2
flat	<	-1
natural	space	0
sharp	>	1
double sharp	>>	2

Table 5-2. Numerical values of accidentals

By adding the value of the accidental that is attached to a note, to the value of the note itself, each note obtains a unique value, or HR[6] number:

Pitch	HR	Pitch	HR	Pitch	HR	Pitch	HR	Pitch	HR
C^{bb}	0	C^{b}	1	C	2	$C^{\#}$	3	C^{x}	4
D^{bb}	8	D^{b}	9	D	10	$D^{\#}$	11	D^{x}	12
E^{bb}	16	E^{b}	17	E	18	$E^{\#}$	19	E^{x}	20
F^{bb}	23	F^{b}	24	F	25	$F^{\#}$	26	F^{x}	27
G^{bb}	31	G^{b}	32	G	33	$G^{\#}$	34	G^{x}	35
A^{bb}	39	A^{b}	40	A	41	$A^{\#}$	42	A^{x}	43
B^{bb}	47	B^{b}	48	B	49	$B^{\#}$	50	B^{x}	51

Table 5-3. Pitches and their representative HR numbers

Using these values any melody may be represented as an array of numbers, for instance the G major scale:

Pitch	G	A	B	C	D	E	F#	G
HR number	33	41	49	56	64	72	80	87

Table 5-4. HR numbers for the G major scale

The numbers in Table 5-3 are for one octave only and 54 is added or subtracted from the pitch number for higher or lower octaves respectively. For example, the G above middle C has an HR value of 33. The number of G two octaves above middle C is therefore 33 + (2 x 54) = 141. To avoid negative numbers that would be generated by octaves below middle C, the numbering system used for the

[6] HR is a mnemonic derived from 'Hewlett System of Pitch Representation'.

programmes under discussion begin with the C that lies three octaves below middle C. The HR number of middle C, for the purpose of this research, is therefore 2 + (3 x 54) = 164.

5.4.1.1 Meantone tuning and equal temperament

With traditional theoretical practice in mind, the pitch values described above are ideal as they give a different value for otherwise enharmonic intervals and pitch representation. Obviously for music that is not composed according to the theory of the old tuning systems, as for example much of twentieth century music, alternative methods could be devised. In twelve tone music there are only twelve tones, as opposed to the thirty five of the older convention, and composers are forced to make use of the archaic note representation purely because of the lack of viable alternatives and not because of the possible structural or tonal implications such a note symbol might have.

For a stylistic analysis of this nature, a single procedure in which no allowance is made for the tonal and harmonic limits, would produce erroneous results. If a true stylistic evaluation is to be achieved, it should be kept in mind that composers who were active when equal temperament prevailed, worked under totally different acoustic conditions than those who composed with the acoustical principles of equal tuning. Equal temperament allows a choice of 12 notes, each of which, because of enharmonicism, can be written in a number of different ways. Meantone[7] tuning allows for 35 different tones, only some of which may be selected depending on the tonal centre, mode and instrument. The dissimilarity in the basic theoretical principles that underlie the two tuning systems needs to be facilitated in the stylistic analysis.

A method was devised that allows analysis according to both tuning systems. For equal temperament the enharmonic pitches and intervals are simply equalised and a second array created in the database so that each melody is represented by two numerical arrays, one for meantone pitches and intervals and one for equal temperament. Pitch names for the equal temperament array use only the letter names and accidentals which result in perfect, major, and minor intervals, and in the case of fourths and fifths, also augmented intervals.

5.4.1.2 Numerical interval representation

The modified HR numbering system discussed above provides a quick and simple method to obtain numerical values for intervals between two pitches. Interval values are obtained by subtracting the HR value of the lower note from the HR value of the upper note; the perfect fifth, E^b - B^b, for example, has an interval value of 48 - 17 = 31. Negative values are made positive by adding 54. For ex-

7 The fact that there are a variety of ways in which the intervals of meantone tuning may be calculated, is of little significance here. Only the fact that the meantone differentiates between tones which are enharmonically interchangeable within the system of equal temperament are of importance.

ample, the value for the a perfect fifth, B^b - F = 25 - 48 = - 23. By adding 54 to the latter, the value of 31 (perfect fifth) is again obtained. The following table is a list of the various intervals and their number values. The order of the intervals in the table is not based on the actual musical sizes but rather on the numerical value allocated to each interval.

Interval	Value		Interval	Value
unison	0		diminished fifth	30
augmented unison	1		perfect fifth	31
diminished second	6		augmented fifth	32
minor second	7		diminished sixth	37
major second	8		minor sixth	38
augmented second	9		major sixth	39
diminished third	14		augmented sixth	40
minor third	15		diminished seventh	45
major third	16		minor seventh	46
augmented third	17		major seventh	47
diminished fourth	22		augmented seventh	48
perfect fourth	23		diminished octave	53
augmented fourth	24		octave	54

Table 5-5. Interval values derived from HR numbers

It should be stressed that the HR interval values do not reflect, in terms of semitones, the actual acoustical values of the intervals they represent. They are merely a method by which intervals are calculated in the computer programmes and have properties which expedite a variety of manipulations. One property which is rather useful is the symmetrical properties of the numbers. An interval inversion, for example, is calculated with the formula: ABS(interval value - 54)[8]. For instance the inversion of a major sixth (39) has a numerical value of 39 - 54 = -15, with an absolute value of 15 — a minor third.

The system has the added benefit in that each number may be stored as a single ASCII[9] code in the database. To simplify the computer programming, ASCII characters were also used for data and string manipulation. This aspect is described later in this chapter. ASCII codes are a standardised series of letters, numbers and symbols, each of which has a specific number (0-256). The symbols are referred to as characters. Since each character always has the same number, it may be used as a mnemonic to represent number sequences.

5.4.2 Music rhythms

Programmers who write computer programmes for musical application have a variety of approaches to translating note values and rhythms into computer language. The most common approach is to

8 ABS means 'absolute value' and converts all numbers to a positive value.
9 ASCII is a mnemonic which stands for 'American Standard Code for Information Exchange', and ensures effective communication between different electronic media.

allocate a number to a specific note value, with the smallest note value being allocated the number 1, and each successive note value sequentially designated a higher number. Another method, and one which is often used in computer programmes, is to allocate letter symbols to the various note values. Two such systems are shown below in Table 5-6; the DARMS[10] system makes use of mnemonics while the MUSTRAN[11] system uses numerical representation. (Wittlich, Schaffer & Babb, pp. 20-23)

NOTE VALUE	DARMS	MUSTRAN
whole note	W	1
half note	H	2
quarter note	Q	4
eighth note	E	8
sixteenth note	S	16
thirty-second note	T	32
sixty-fourth note	X	64
one hundred twenty- eighth note	Y	128
dot	.	.
tie	F (prefixed)	F (before second value)
rest	R (prefixed)	R (after value)

Table 5-6. DARMS and MUSTRAN note duration codes

The DARMS system is the least complicated and very suitable if mathematical manipulation is not required. However, this approach becomes very complicated when dealing with dotted and double dotted notes, tied notes, and irregular subdivisions such as triplets. For the application of Information Theory an adapted form of the MUSTRAN system is much more suitable for the calculation of ratios, i.e. the expression of duration as a percentage of all the note values combined in the section of music being analysed. By inverting the sequence of numerical values that is allocated to each of the note values in the table above, algorithmic manipulation becomes relatively easy. The range of available values allows for 128th notes. The latter is very rare in music but allows for more accurate calculations.

The values used for the programmes of this research are shown below. Each of the note values is given a number which is directly related to the actual value of the note expressed as a multiple of 128.

NOTE VALUE	MUSTRAN
whole note	128
half note	64
quarter note	32
eighth note	16
sixteenth note	8
thirty-second note	4
sixty-fourth note	2
one hundred twenty- eighth note	1

Table 5-7. Adapted MUSTRAN note value system

[10] The DARMS code was originally called the 'Ford-Columbia' code and was developed by Stefan Bauer-Mengelberg.
[11] Jerome Wenker developed the MUSTRAN code at the Indiana University for application in ethnomusicological research.

To obtain the relative note values, each note is expressed as a ratio of the total duration of the complete composition. The latter is obtained by simply adding all the note values together. The bar division, which dictates the position of accentuated note gradations, or beats, form the subdivisions of the overall composition. A piece of music without any bar lines is regarded as comprising a single bar due to the absence of any regular metrical subdivision of the music. After a note is entered into the database, the note value is converted to a multiple of 128. This means, for instance, that a dotted quaver has a value of 24 (16 + 8), while a quaver triplet has the value, 21,33. In contrast to the pitch number, which is stored in ASCII code, the note value is stored as a number because these calculations may result in fractions as is the case of the quaver triplet. ASCII code cannot be presented by fractions.

Another approach that was considered was to express a note value in terms of real time, in other words as note value per second or per minute. However, this approach requires that the tempo indications and variations are constantly kept in mind and that note values are continuously calculated in relation to the tempo of the music. Unless, as in the case of some types of electronic music, it is expressly composed with specific real time criteria and limits in mind, using real time as a measuring unit is not an effective or expedient method.

5.5 The computer algorithms

Each of the main programmes discussed in this chapter comprises a number of different routines, some of which are not exclusive to a single programme. For the sake of clarity each of the routines are demonstrated here as if they are totally independent.

In the flow charts that follow the arrows indicate the direction of the flow of the programme and the branching is done according to basic Boolean logic. The boxes with the light borders represent the routines where branching takes place, while the rectangles with the heavier borders represent the functions that are called by the branching routines.

5.5.1 Algorithms for melody analysis

Analysis of the melodic data comprises different steps and can be done according to the following parameters controlled by the operator:

1. Pitches only
2. Pitches by ratio
3. Pitches and note values combined
4. Stochastic analysis of rhythm

5. Stochastic analysis of intervals

As the melodic information is stored in an array it is relatively easy to perform string manipulations and readily allows comparison between various sections of the array. The algorithms for each of the types of analysis listed above are described below with the flowcharts.

5.5.1.1 Pitch and rhythm analysis

Because of their similarity, the algorithms for the calculation of single pitches and note values are discussed simultaneously. In principle a shadow array[12] is created to hold the various values, be they pitches or note values. Looping through the codes of the melody, each new character is added to the shadow-array and a counter[13] for the number of different codes (C)[14] as well as a counter for that specific code (SC) is increased. If a code already exists in the shadow-array, only the counter for that specific code (SC) is increased. For every code encountered a counter for the total number of codes (TC) counted is increased as well.

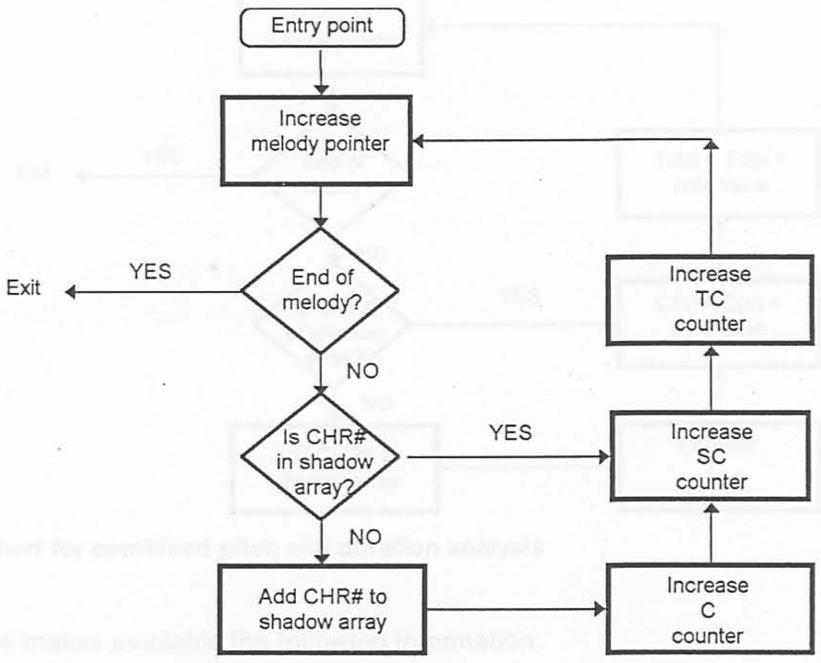


Figure 5-6. Flowchart for the analysis of pitch or note values

Once the end of a melody is reached, three different totals become available:

1. The total number of different codes encountered (C);

[12] Shadow arrays are similar to normal arrays but only stay in the computer's memory temporarily. Once it is no longer needed it is removed from memory. Usually it contains duplicate or temporary data for comparison or manipulation.

[13] Counter usually begin with a value of 0 and are increased each time a specific action occurs.

[14] These are mnemonics used as variables in the programme. Variables represent registers which can hold a variety of changing data.

2. The number of times each code occurs (SC); and
3. The grand total number of codes in the melody array (TC).

Another algorithm uses these results to calculate the various entropy and redundancy values. Towards the end of this chapter the algorithms for entropy calculations are discussed.

5.5.1.2 Pitches combined with duration

The flowchart for the routine to calculate the values of pitches based on their duration values is very similar to the one that merely calculates the frequencies of pitches. It differs in that duration is used as the measuring unit in multiples of 128th notes. Each time a note code is encountered, the note value associated with it is added to the counter:

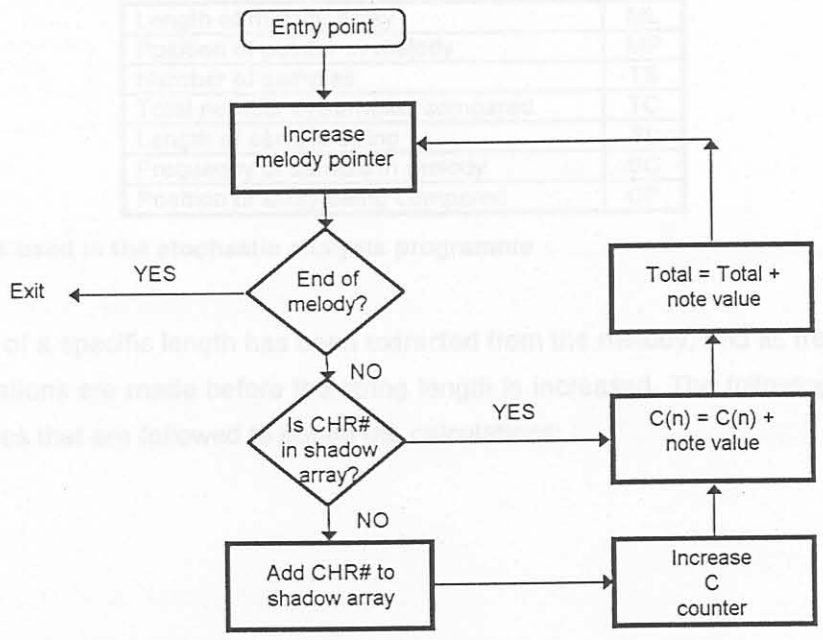


Figure 5-7. Flowchart for combined pitch and duration analysis

The above routine makes available the following information:

1. The total number of pitch codes in values of 128ths (Total);
2. the occurrence of the different pitches, also in values of 128ths (TC[n]);
3. the total number of different codes encountered (C).

5.6 Stochastic analysis of intervals and rhythm

The programme routines for the stochastic analysis of melodies are somewhat more complex as the frequency of increasingly larger portions of the melody have to be counted and compared. In order to achieve this a two-dimensional interval array, containing the interval values for both equal temperament as well as meantone tuning, is created. As with the pitches the intervals are converted to ASCII codes, which make comparisons easier to manage. The second dimension of the array contains the direction of the interval and is indicated by a mnemonic; '>' for a descending interval, and '<' for an ascending interval.

In essence the programme consists of various nested loops. In the principal loop the string length for comparison is continuously lengthened until it has the same length as the melody array. A second loop, which is nested inside the first loop, runs through the melody array, first compiling the sample and then counting the frequency of its occurrence by stepping through the melody from beginning to the end. Other loops in the program are dependent on certain conditions being met.

The variables[15] used in the programme and flowchart are shown in the following table:

Length of melody array	ML
Position of pointer in melody	MP
Number of samples	TS
Total number of samples compared	TC
Length of sample string	SL
Frequency of sample in melody	SC
Position of array being compared	CP

Table 5-8. Variables used in the stochastic analysis programme

Each time a string of a specific length has been extracted from the melody, and its frequency counted the entropy calculations are made before the string length is increased. The following flowchart illustrates the sequences that are followed to obtain the calculations:

[15] Variables are labels given to pieces of information. A counter is a variable which is incremented or decremented every time a condition is met. Fixed variables cannot be changed and usually serve as a mnemonic for a specific numerical value.

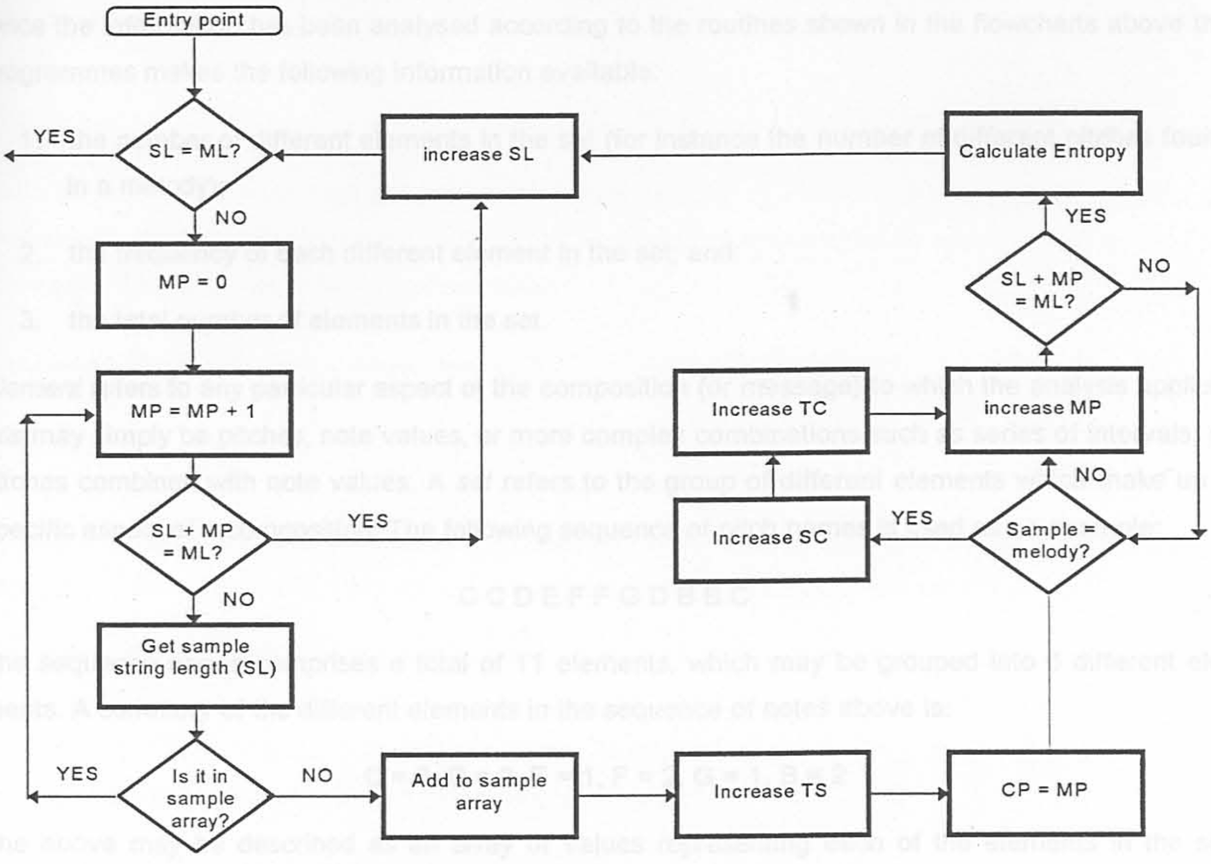


Figure 5-8. Flowchart for stochastic melody analysis

The flowchart for the stochastic analysis of rhythm is virtually the same, except that it only needs a one-dimensional array because there is no need to allow for any interval differentiation.

5.6.1 Calculation of entropy values

Once all the necessary data has been collected in the database it may be analysed using some basic algorithmic calculations. The computer used for the analysis in this research, an IBM compatible personal computer, is fitted with a mathematical co-processor and has a precision of seven decimal digits and a dynamic range of 10-38 to 10+38. The co-processor increases the speed of calculations and is more accurate but is not a prerequisite.

One problem that was encountered was that some calculations result in extremely small fractions, smaller than seven decimal places, operations on these tend to result in an error of < 0.0000001. Because results of analyses are only calculated to the nearest sixth decimal place, this error has no significant effect on the results. However, it causes summations which are placed within a continuous loop, and which can only be exited when two values are the same, to continue ad infinitum. The only way to overcome this was to round off any number to the nearest sixth decimal place.

Once the information has been analysed according to the routines shown in the flowcharts above the programmes makes the following information available:

1. the number of different elements in the set (for instance the number of different pitches found in a melody);
2. the frequency of each different element in the set; and
3. the total number of elements in the set.

Element refers to any particular aspect of the composition (or message) to which the analysis applies; this may simply be pitches, note values, or more complex combinations such as series of intervals, or pitches combined with note values. A *set* refers to the group of different elements which make up a specific aspect of a composition. The following sequence of pitch names is used as an example:

C C D E F F G D B B C

The sequence above comprises a total of 11 elements, which may be grouped into 6 different elements. A summery of the different elements in the sequence of notes above is:

C = 3, D = 2, E = 1, F = 2, G = 1, B = 2

The above may be described as an *array* of values representing each of the elements in the sequence. An array is thus a summary of the different elements in a set. In computer language the shown example would be contained in a two-dimensional array — dimensioned as: *Array(6, 2)* — to contain:

1. the number of different elements (6 in this case) and
2. the quantity of each specific element in the array.

A table illustrates the structure of the array more clearly:

	Counter:	1	2	3	4	5	6
1	Pitch:	C	D	E	F	G	B
2	Quantity:	3	2	1	2	1	2

Table 5-9. Illustrative values in an element array

If the array, represented by **A** contains the frequency of each element of a set of **n** elements, the total number of elements would be expressed as:

Equation 5-1. Total elements in an array

$$\begin{aligned} \textit{total elements} &= A_1 + A_{2+} \dots A_n \\ &= \sum A_n \end{aligned}$$

With the value of the total set as well as the values of each of the elements now available, the relative frequencies of each element may be calculated by dividing the value of each element by the total set value. This is shown in the equation below:

Equation 5-2. Element proportions of total elements

$$pA_n = \frac{A_n}{\sum A_n}$$

A simple calculation at this point checks for any calculation error by adding all the relative frequencies together. If the calculations were done correctly, the summation should add up to 1.

The next step is to calculate the entropy values of each of the different elements in the set, **A**, and summate them to obtain the actual entropy value, **H**, of the set as a whole. In order to obtain the information contents of the set in bits, the binary logarithms of the relative frequencies are weighted (multiplied) by the relative frequencies, **p**, and summated:

Equation 5-3. Calculation of the entropy of a set of values

$$H(pA_1, pA_2, \dots, pA_n) = -(pA_1 \log pA_1 + pA_2 \log pA_2 + \dots pA_n \log pA_n)$$

or

$$H = \sum_{i=1}^{n} pA_1 \log_2 pA_1$$

To achieve the information (entropy) value of the set, a simple loop is set up that operates on the relative frequencies value of each element and adds it to a variable called **entropy_total**[16]. The flowchart below illustrates the steps the programme runs through to achieve this. As PASCAL has no direct function to do conversion from logarithms on base 10 to binary logarithms, it was necessary to include an algorithm in the programme to do this conversion. This algorithm is also shown in the following flowchart. The variable **EV** is used to hold the entropy value:

16 onger mnemonics for variables consisting of more than one word are connected with underscores.

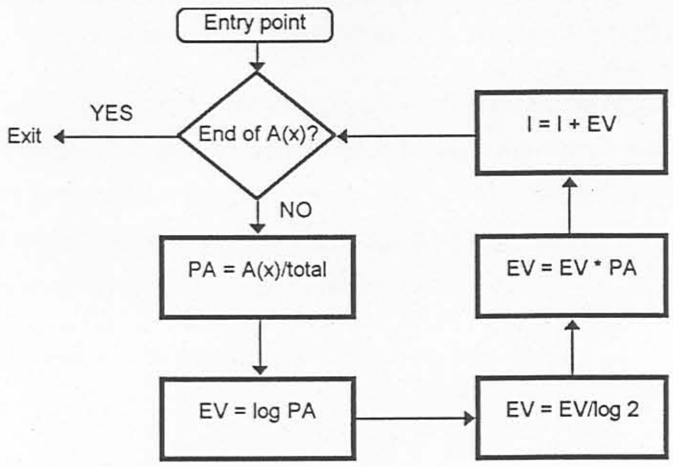


Figure 5-9. Flowchart for the calculation of entropy

Maximum entropy is somewhat simpler to calculate as it requires no loop and the operation is only done on the number of elements in the set, where **N** represents the total number of elements in the array being investigated:

Equation 5-4. Calculation of Maximum Entropy

$$H_{\max} = \frac{bin\ N}{bin\ 2}$$

With the entropy value and maximum entropy values in two separate variables a third variable is used to calculate the relative entropy as a percentage value:

Equation 5-5. Calculation of relative Entropy

$$H_{rel} = \frac{H}{H_{\max}} 100$$

A fourth variable is used to hold the redundancy value:

Equation 5-6. Calculation of relatively Redundancy

$$R_{rel} = 1 - H_{rel}$$

5.7 Conclusion

In the development of the analysis programmes a structured approach was adopted, in which sections of each of the three main programmes were divided into smaller routines. Each phase was thoroughly tested to assess the effectiveness and accurateness of each routine and where possible error traps were included to further ensure accuracy of the results.