Proceedings of The Institute of Acoustics

THE SYNTHESIS OF SOUND USING WALSH FUNCTIONS

JOHNATHAN AUCKLAND AND MICHAEL GREENHOUGH

Physics Department, University College, Cardiff

INTRODUCTION

Since any sound is characterised by a unique waveform facilities for generating waveforms are very important in the field of electronic music. Using a digital computer any mathematical function can be generated, piecewise if necessary, and converted to an analogue voltage signal and thence to sound. However, the correspondence between this function and the resulting sound, although unambiguous, is often far from obvious. Moreover, the use of a method of sound synthesis of such generality is often unjustifiable because a high proportion of the resulting sounds are almost indistinguishable and of very limited musical value.

In practice so-called additive methods of synthesis are often preferred. By means of hardware or software a number of sinusoidal components with chosen amplitudes and phases are summed to produce the desired signal. The practical restriction on the number of sinusoids available limits the number of sounds that may be produced. However, since the ear acts as a Fourier analyser and many natural and musical sounds have approximately discrete frequency spectra, this synthesis method is both a fairly intuitive and a useful one.

The provision of a large number of pure sinusoids can be quite demanding on hardware or software and so the use of alternative functions suggests itself. In principle, any set of functions which is complete and orthogonal can be used as the basis of a synthesis method. The advantage of choosing Walsh functions (1) is that they are rectangular, two-state functions (see Fig.1) and thus readily represented and generated by digital means.

The derivation of the Walsh spectrum corresponding to a desired signal waveform is analogous to Fourier analysis. In both cases determining the appropriate amplitude for each constituent function involves integrating the product of that function and the function being analysed. In the Fourier analysis of arbitrary functions numerical integrations, involving time-consuming complex arithmetic, may be necessary. The corresponding Walsh analysis is much simpler because the Walsh functions are, in effect, either zero or unity and so may be placed outside the integral sign. Thus only the function to be analysed need be integrable for an exact determination of the Walsh function amplitudes, and even a numerical integration reduces to a simple summation. Moreover, there exists a Fast Walsh Transform which is analagous to the Fast Fourier Transform and reduces calculation and memory requirements in a similarly dramatic way.

Thus Walsh analysis and synthesis seems an attractive alternative to the Fourier approach and indeed several methods for generating the Walsh functions and using them for musical sound generation have been proposed (2,3). We now present a flexible, software approach to Walsh function generation and use. A remaining problem is that each Walsh function contains a harmonic series of

Fourier-type components and so predicting the results of a Walsh synthesis is more difficult and less intuitive than the more common Fourier synthesis. This problem is considered in the final section.

SYSTEM HARDWARE

The Walsh-function generator system hardware involves a standard Nascom II microcomputer (based on a Z80 CPU chip) with 64k of RAM, a VDU and cassette storage. Two additional pieces of hardware were specially designed and built for the project.

(i) A light-pen facility (4) for inputting data to control the amplitude envelopes of the functions.

(ii) A voice card for producing the sound signals. This contains its own Z80 CPU, 4k of on-board RAM and 16 digital-to-analogue converters and thus has considerable flexibility and autonomy. Controlling programs and data are transferred to it from the host computer by direct memory access.

SYSTEM SOFTWARE

In order to ensure efficient data exchange between various parts of the system and satisfactory speed of operation the software is written entirely in the Z80 assembly language.

Data for the first 31 Walsh functions (see Fig.1) are stored as bit patterns in an area of RAM. Up to 16 of these can be selected at a time to form the constituents of the output signal. By using the light-pen or keyboard input the user can define a bank of up to 96 envelope shapes which can then be applied to these functions. The envelopes may be assigned arbitrarily and independently to each function or, alternatively, a few selected shapes may be applied to sub-groups of the functions.

Convenient control of the system is facilitated by user-friendly, menu-driven software. This allows the user to define, store, recall and screen-edit the envelopes, which are structured in 16 blocks of 6.

SOUND SYNTHESIS

The sound spectra of conventional instruments and other natural sounds are strongly dynamic, the amplitudes of the harmonics showing complex variations particularly during the starting transient. Thus interesting sounds can be synthesised by varying the amplitudes of a large number of sine components in a similar way but this requires a lot of control data. Besides, as Chowning has shown, the fact that there are complex variations is much more important than the precise form these variations take. Thus his Frequency-Modulation technique of sound synthesis (5), which is simple and economic, involves somewhat arbitrary harmonic amplitude variations but produces rich and interesting timbres.

THE SYNTHESIS OF SOUND USING WALSH FUNCTIONS

This F.M. technique does allow very useful control of certain gross spectral features such as the presence or absence of odd and even harmonics and the dependence of bandwidth on intensity. For example, brass-like tones may be simulated by allowing all harmonics and contriving an increase in the high-frequency spectral content with increasing intensity. Similarly, allowing the odd harmonics to dominate and decreasing the intensity of the higher harmonics as the attack proceeds tends to produce woodwind-like timbres.

We now consider whether the set of Walsh functions may be used to produce such rich timbres in an economical way. The Fourier components of the first 31 Walsh Functions are given in Table 1. These are calculated from the standard equations

$$Wal(m,t) = \tfrac{1}{2}a_0 + \sum_{n=1}^{\infty}\left(a_n \cos 2\pi \nu_n t + b_n \sin 2\pi \nu_n t\right)$$

$$a_n = \frac{2}{T}\int_{(T)} Wal(m,t)\cos 2\pi \nu_n t \, dt \quad \text{etc.}$$

For the nth harmonic of Wal(m,t), this can be reduced to

$$a_{nm} = \frac{1}{\pi n}\sum_{t=-16}^{15}\left[\sin 2\pi n\left(\frac{t+1}{32}\right) - \sin 2\pi n \frac{t}{32}\right] Wal(m,t) \quad (m < 32) \quad \text{etc.}$$

It can be seen from Fig.1 that Wal(m,t) is an odd or even function depending simply on whether m is odd or even. So for the odd functions any cosine transform is zero and any sine transform is nonzero. The reverse is true for the even functions. Thus the harmonics of odd Walsh functions have sine phase and those of even ones, cosine phase, as indicated. A study of Table 1. reveals several features which may be of use when considering how best to combine Walsh functions for musical purposes.

(i) Pairs of functions are readily found (eg. m = 1 and m = 5) which have the same harmonics but at different amplitudes and often with opposite phases.

(ii) No function contains all the Fourier harmonics although many contain all the odd ones.

(iii) The functions with $m = 2^{k-1}$, where k=1,2,3 ..., form a complementary group containing between them all the harmonics. Many similar groups are to be found.

(iv) As m increases so does the number of the harmonic with the maximum amplitude in the spectrum of Wal(m,t).

From (i) we see that if, for example, Wal(2,t) is given a ramp-shaped attack of length T and Wal(6,t) an identical attack delayed by T/2, the following will occur. Harmonics 3,5,11,13,19,21,27 and 29 of Wal(2,t) will increase smoothly in amplitude with a phase of either 0° or 180°. Then at time T/2 the same harmonics of Wal(6,t) will enter, but with opposite phases. Since each of these is greater in magnitude than its counterpart in Wal(2,t), the resultant amplitude of each of these harmonics will pass through zero, at different times, and then grow with the opposite phase until the steady state is reached after a total time of 3T/2. Other pairs of Walsh Functions with similar complementary spectra may be added with one function under each envelope to create a potentially interesting attack to the sound.

From (ii) and (iii) it is clear that we can synthesise sounds which contain only odd, only even, or all of the harmonics. From (iv) it follows that we can easily move the maximum in the spectral envelope from low to high, or vice versa, during the evolution of a sound. This is accomplished by choosing envelope shapes in such a way that, say, lower-order Walsh functions give way to higher ones as time progresses.

We hope to present examples of sound synthesis at the meeting.

BIBLIOGRAPHY

1. J.L. WALSH 1923 Ann. Journ. Math. 55, 5-24.
A closed set of orthogonal functions.

2. B.A. HUTCHINS, JR. 1973 J. Audio Eng. Soc., 21, 640-645.
Experimental electronic music devices employing Walsh functions.

3. M. ROZENBERG 1979 Computer Music Journal 3 (1), 42-47.
Microprocessor-controlled sound processing using Walsh functions.

4. J. AUCKLAND 1983 80-BUS News, April-May, 4-10.
A light pen for the Nascom 2.

5. J.M. CHOWNING 1973 J. Audio Eng. Soc., 21, 526-534.
The synthesis of complex audio spectra by means of frequency modulation.

ACKNOWLEDGEMENTS

We are grateful to Professor C A Taylor for the provision of research facilities.

THE SYNTHESIS OF SOUND USING WALSH FUNCTIONS

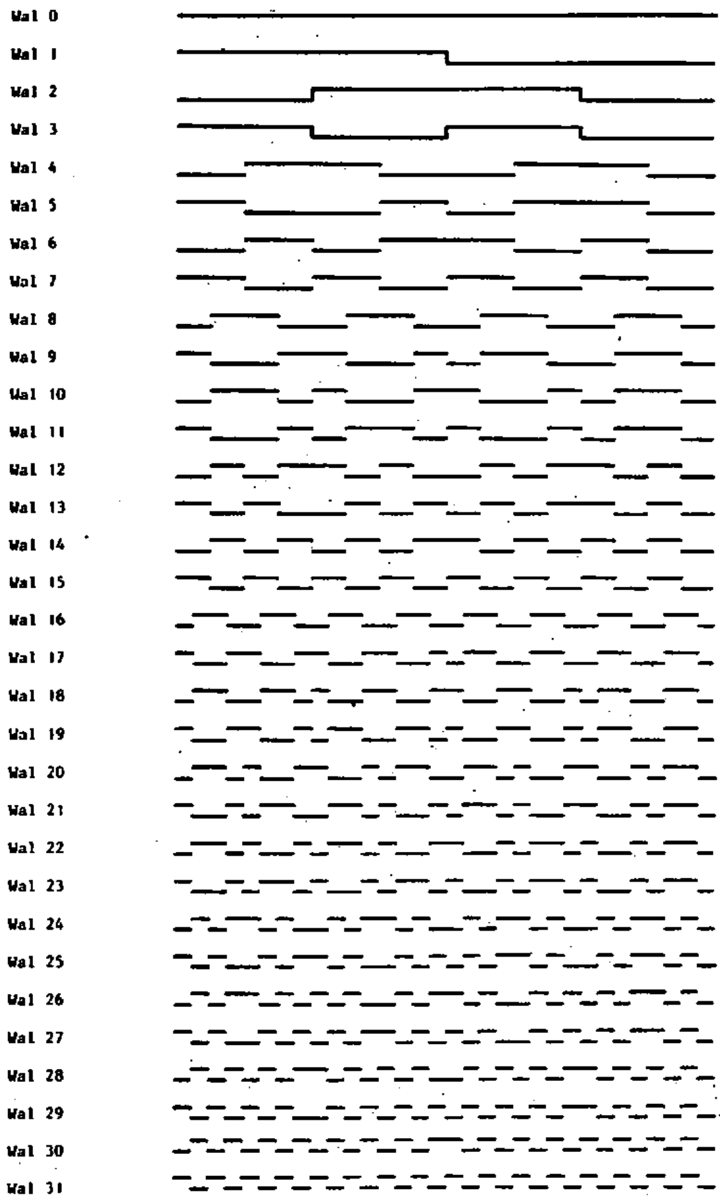

Figure 1. The first 32 Walsh Functions in Sequency Order.

THE SYNTHESIS OF SOUND USING WALSH FUNCTIONS

Walsh Function Number	1000 x Amplitude of Fourier Component															
	harmonic number															
	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
1	637	212	127	091	071	058	050	042	037	034	030	028	025	023	022	020
2	-637	212	-127	091	-071	058	-050	042	-037	034	-030	028	-025	023	-022	020
5	-264	512	307	-037	-029	140	118	-018	-016	081	073	-011	-011	056	053	-009
6	-264	-512	307	037	-029	-140	118	018	-016	-081	073	011	-011	-056	053	009
9	-052	-342	460	189	147	209	-079	-003	-003	054	110	058	053	085	-036	-002
10	052	-342	-460	189	-147	209	079	-003	003	-054	-110	058	-053	085	036	-002
13	-126	142	-191	451	356	-087	033	-008	-007	022	-045	139	128	-035	014	-004
14	-126	-142	-191	-457	356	087	033	008	-007	-022	-045	-139	128	035	014	004
17	-012	-043	-101	-375	433	162	107	086	076	073	085	170	-105	-189	-004	-000
18	012	-043	101	-375	-433	162	-107	086	-076	073	-085	170	105	-189	004	000
21	005	-104	-246	155	-179	391	260	-036	-031	178	205	-070	044	-046	-011	000
22	005	104	-246	-155	-179	-391	260	036	-031	-178	205	070	044	046	-011	-000
25	-026	-155	164	031	-036	-261	390	178	157	267	-137	-014	008	030	-016	-001
26	026	-155	-164	031	036	-261	-390	178	-157	267	137	-014	-008	030	016	-001
29	-063	064	-068	074	-086	108	-161	431	380	-110	057	-034	021	-013	007	-002
30	-063	-064	-068	-074	-086	-108	-161	-431	380	110	057	034	021	013	007	002

Walsh Function Number	harmonic number														
	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
3	636		212		127		091		071		058		050		042
4	-637		212		-127		091		-071		058		-050		042
7		637				212				127				091	
8		-637				212				-127				091	
11	-264		512		307		-037		-029		140		118		-018
12	-264		-512		307		037		-029		-140		118		018
15				636								212			
16				-636								212			
19	-052		-342		460		189		147		209		-079		-003
20	052		-342		-460		189		-147		209		079		-003
23		-264				512				307				-037	
24		-264				-512				307				037	
27	-127		142		-191		457		356		-087		033		-008
28	127		-142		-191		-457		356		087		033		008
31								637							

Table 1. Fourier Analysis of Walsh Functions from 1 to 31
(Functions have sine phase if underlined, cosine phase if not)