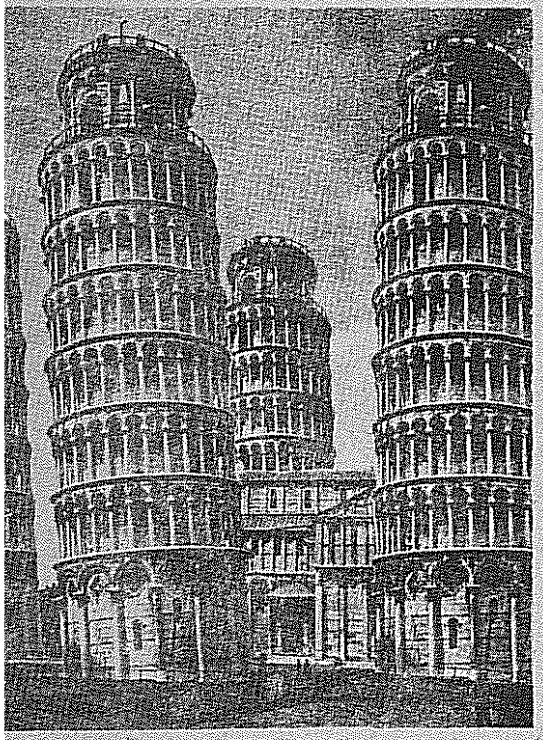
C.N.U.C.E. ISTITUTO DEL CONSIGLIO NAZIONALE DELLE RICERCHE



QUARTO COLLOQUIO DI INFORMATICA MUSICALE

1981

atti vol. I

ATTI DEL QUARTO COLLOQUIO DI INFORMATICA MUSICALE

Pisa, 1-2 Giugno 1981

CNUCE - C.N.R., Via S.Maria 36, Pisa

INTRODUZIONE

Il Quarto Colloquio di Informatica Musicale si e' svolto a Pisa nei giorni 1 e 2 Giugno 1981, organizzato con la collaborazione del Reparto Musicologia del CNUCE. I precedenti incontri furono a Pisa (CNUCE, 23-24/2/1976), a Milano (Istituto di Cibernetica, 14/12/1977), e a Padova (Universita', 2-3/4/1979).

I lavori sono stati aperti da un saluto del Prof. G.Capriz, Direttore del CNUCE, e del Maestro P.Grossi, responsabile del Reparto Musicologia.

Sono state presentate piu' di 20 relazioni, della durata di 15-30 minuti ciascuna) suddivise nelle seguenti sessioni: Composizione; Sintesi digitale; Analisi formale ed intelligenza artificiale; Sistemi musicali digitali e personal computers; Percezione uditiva/percezione visiva. Sono inoltre pervenuti 5 posters (introdotti in questo incontro per la prima volta) e 8 nastri magnetici.

Il concerto offerto la sera della prima giornata dei lavori ha compreso l'audizione di nastri di V.Asta, O.Laske, T.Rampazzi, M.Graziani, e una dimostrazione in tempo reale del sistema pisano di computer music (tau2-taumus), del Maestro P.Grossi. Altri due microsistemi di sintesi del suono ed elaborazione musicale sono stati illustrati, con dimostrazioni, durante la seconda giornata.

Ospite del Convegno e' stato Kevin Jones, (Dipartimento di Musica, City University di Londra), che ha parlato della situazione della computer music in Inghilterra e delle proprie ricerche compositive, con esempi musicali su nastro: il suo invito mira a stabilire ulteriori contatti con la attivita' inglese nel settore.

A conclusione del Colloquio e' stata annunciata la imminente nascita della Associazione di Informatica Musicale Italiana, che avra' sede a Venezia, presso la Biennale, e che avra'come scopo principale il coordinamento delle attivita' italiane nel campo delle applicazioni musicali degli elaboratori elettronici.

I lavori del Convegno sono stati seguiti da circa un centinaio di persone, provenienti da Istituti e Conservatori di oltre una dozzina di citta' italiane.

Comitato organizzatore:

Tommaso Bolognesi - CNUCE-C.N.R., Pisa .

Giovanni De Poli - Istituto di Elettronica ed Elettrotecnica, Padova.

Goffredo Haus - Istituto di Cibernetica, Milano.

Collaboratori:

Silvio Farese, Laura Melis - CNUCE-C.N.R., Pisa.

INTRODUCTION

The 4th "Colloquio di Informatica Musicale" was held at Pisa on June 1-2, 1981; it was organized with the collaboration of the Musicology Section of CNUCE, an Institute of the (Italian) National Research Council. Previous meetings were at Pisa (CNUCE, Feb. 23-24, 1976), at Milan (Cybernetics Institute, Dec 14, 1977) and at Padua (University, April 2-3, 1979).

Prof. G.Capriz, Director of CNUCE, and Maestro P.Grossi,head of the Musicology Section of CNUCE, expressed their welcome to relators and public with a short introductive talk.

Contributions included more than 20 papers (talks lasted 15 to 30 minutes), grouped into the following sessions: Composition; Digital Synthesis; Formal Analysis and Artificial Intelligence ; Digital Music Systems and Personal Computers; Aural Perception / Visual Perception. Furthermore, 5 posters (introduced for the first time in the Colloqui) and 8 tapes were received.

The concert given on Monday night included tapes by V.Asta, O.Laske, T.Rampazzi, M.Graziani, and a demonstration of the local computer music system (tau2-taumus) by P.Grossi.

Two music processing microsystems were also demonstrated.

Kevin Jones (Music Dept., City University of London) was invited to speak both about computer music in Great Britain and his own research, as a first step for further contacts between the two Countries.

It was eventually announced that the Italian Musical Informatics Association is going to be established in Venice (at the Biennale), with the purpose to promote and coordinate italian activities in the field of computer applications to music.

About one hundred people attended the meeting, coming from Institutions and Conservatoires distributed over a dozen italian cities.

The Conference Committee:

Tommaso Bolognesi	-	CNUCE-C.N.R., Pisa.
Giovanni De Poli	-	Istituto di Elettronica ed Elettrotecnica, Padova.
Goffredo Haus	-	Istituto di Cibernetica, Milano.

Collaboration:

Silvio Farese	-	CNUCE-C.N.R., Pisa
Laura Melis	-	CNUCE-C.N.R., Pisa

M E M O R I E / P A P E R S

COMPOSIZIONE MUSICALE

SINTESI DIGITALE

ANALISI MUSICALE ED INTELLIGENZA ARTIFICIALE

SISTEMI DI ELABORAZIONE MUSICALE E PERSONAL COMPUTER

PERCEZIONE UDITIVA/PERCEZIONE VISIVA

POSTER / POSTERS

ractal Sampling - T.BOLOGNESI - CNUCE, C.N.R., PISA
Gioco di velocita" - R.DOATI - CONSERVATORIO B.MARCELLO, ENEZIA
he SIN I Digital Synthesizer - P.MARRAMA, I.D.A.C. - .N.R., ROMA
athematical Models for the Synthesis of Complex Sound tructures - S.PETRARCA, ROMA
Embrio" - N.SANI - ROMA

NASTRI / TAPES

A Volte" - V.ASTA
Terpsichore" - O.LASKE
Atmen Noch" - T.RAMPAZZI
The Silent God" - M.GRAZIANI

Gioco di Velocità" - DOATI
Little Science, Little Magic" - FARNEDA
Embrio" - SANI
Cardo" - TORRESAN

Poesia, algebra lineare e musica: un'esperienza

Poetry, linear algebra and music: an experience

Vito ASTA (1)

Riassunto

La memoria descrive l'insieme di considerazioni e di operazioni chiamate in causa nella composizione e nella realizzazione di A VOLTE per nastro magnetico; l'opera è interamente basata sulle caratteristiche prosodico-fonetiche di una omonima poesia, secondo un procedimento che, partendo dall'analisi al calcolatore dei vari fonemi componenti, arriva a una serie di dati sui parametri di controllo del processo di sintesi tramite l'applicazione di un particolare omomorfismo tra spazi lineari e a varie altre decisioni logico-combinatorie.

Abstract

The paper describes the various considerations and operations required during the composition and realisation of A VOLTE for magnetic tape; the piece is entirely based on the prosodic and phonetic features of a poem, with the same title, according to a procedure which, starting from a computer analysis of the various phonemes, arrives to a series of data on the control parameters for the synthesis process through the application of a particular homomorphism between linear spaces and various other logical and combinatorial decisions.

---

(1) IRCAM, Parigi e AXIS DIGITAL, Parigi

1) Introduzione

Il punto di partenza di A VOLTE è una poesia omonima, composta dallo stesso autore alcuni anni fa, il cui testo è il seguente:

Dentro ci prende a volte un vuoto
dentro dentro dentro a volte
dentro dentro dentro

dentro ci ci prende a volte a vo
dentro a volte un vuoto
dentro ci prende Abbazia

un senso un senso un senso di a volte
mi ste ro angoscia
mi ste ro angoscia

come un coro di vecchie che pregano come un
dentro ci prende
di mistero e d'a ngoscia

coro nell'antica Abbazia
vuoto l'eco regnava
dentro vuoto

l'eco regnava
l'eco regnava
l'eco severa.

Senza entrare qui nei dettagli della poesia in se stessa, diremo che in essa possono isolarsi quattro nuclei di base, costituiti dalle frasi seguenti:

1) dentro
ci prende a volte un vuoto,

2) un senso di mistero e d'angoscia

3) come un coro di vecchie che pregano
nell'antica Abbazia;

4) l'eco regnava,
severa.

Questi quattro nuclei, che nell'insieme possono considerarsi una versione "monodica" della poesia, sono stati fatti recitare e registrati dalla voce di Maurizio Testa; il segnale ottenuto è stato quindi digitalizzato e memorizzato in un calcolatore (un PDP-10 della Digital Equipment Corporation), e con l'aiuto di un insieme di programmi di analisi e editing di segnali sonori si sono determinati, a mano, gli istanti d'inizio e fine di ciascun fonema.

L'idea di base consisteva nel voler ricavare un certo numero di dati sui parametri di controllo degli strumenti utilizzati nella composizione a partire dai dati prosodici contenuti implicitamente nel testo, di cui la dizione registrata costituisce una particolare

realizzazione, e poi altri dati a partire invece dalle caratteristiche fonetiche del testo stesso: in tal modo, tutta l'intima microstruttura della poesia (intesa come entità fisico-acustica, sintattica e semantica a un tempo) viene proiettata nella macrostruttura dell'opera (anzi la determina interamente); la composizione dunque rifletterà, ponendole in nuova e particolare luce, tutte le simmetrie, i chiasmi, i contrasti del testo, e in generale tutta quella parte d'informazione, legata alla ricchezza del linguaggio parlato, che "non passa" (se non tramite una re-interpretazione personale) a una lettura del testo scritto.

2) La trasformazione dei dati prosodici

Per questi dati si è dunque adottato il seguente procedimento:
1) per ogni fonema, si calcolano i valori di energia media (in unità arbitrarie) e di frequenza fondamentale media (in Hz); ciò è stato fatto mediante opportuni programmi di analisi del segnale, di uso generale.
2) la poesia viene considerata come un insieme ordinato di fonemi o pause (113 in tutto), a ciascuno dei quali vengono associati 3 valori prosodici (in base alle misure fatte): d (durata in msec.), e (energia in dB rispetto a un'unità arbitraria), f (frequenza in Hz). Alle frequenze delle pause e delle consonanti sorde sono stati attribuiti, per completare gli insiemi, dei valori scelti arbitrariamente. Si hanno così 3 insiemi di 113 valori prosodici.

3) si fissa un punto di riferimento (o origine) per ciascun insieme, calcolato come media dei valori dell'insieme stesso; tale origine viene sottratta a ciascun elemento, così da avere valori positivi e negativi (rispetto all'origine) per ciascun insieme.
4) si considera a questo punto uno spazio vettoriale sul campo dei

numeri reali R [Ayres, 1965]

$$P = D \times E \times F = \left\{ \begin{vmatrix} d \\ e \\ f \end{vmatrix} \right\} \sim R^3$$

isomorfo a R^3 , di cui ogni terna di valori associati ai vari fonemi può considerarsi elemento.

5) i parametri di controllo cercati verranno ricavati a partire da una trasformazione dell'insieme P delle terne ordinate di valori; si opera dunque un isomorfismo di P (cioè una trasformazione lineare di P su se stesso), che opererà su ciascun elemento dell'insieme P una traslazione delle componenti, secondo opportuni fattori, in modo da deformarle secondo un'unica legge. Tale legge è data dalla matrice non singolare $\underline{A}$ che realizza l'isomorfismo nella base data:

$$\underline{A}\ \underline{p} = \underline{p}'$$

ovvero

$$\begin{vmatrix} d_d & d_e & d_f \\ e_d & e_e & e_f \\ f_d & f_e & f_f \end{vmatrix} \begin{vmatrix} d \\ e \\ f \end{vmatrix} = \begin{vmatrix} d' \\ e' \\ f' \end{vmatrix}$$

dove con i_k si indica, per gli elementi della matrice $\underline{A}$, il coefficiente del contributo alla componente i' da parte del parametro k. A questo punto si ha un insieme di elementi di P', costituito da terne di dati completamente astratti, ogni componente delle quali non è più veramente individuabile come durata, o energia, o frequenza: ciascun valore realizza in sè una particolare sintesi dei dati di

partenza, in quanto ognuna delle tre componenti d'origine ha contribuito, in misura più o meno rilevante, alla determinazione attuale del parametro. Il metodo adottato per determinare in pratica la matrice A è esposto più avanti.

6) i valori così ottenuti vengono riportati, tramite un nuovo offset dato questa volta dal minimo valore di ciascun insieme componente, a valori tutti positivi, e infine, tramite un coefficiente di scala, a valori compresi tra 0 e 32767 (dato che i parametri accettati dagli strumenti usati sono valori positivi a 16 bits). Con questa operazione si è voluto ulteriormente denaturare i dati ottenuti dalla loro matrice originale: l'attenzione è rivolta non tanto ai valori in se stessi, presi cioè isolatamente dal contesto globale - infatti non ha senso riconoscere più in essi delle frequenze o dei tempi o delle energie -, quanto piuttosto, dato che sono normalizzati, agli scarti e ai rapporti (e dunque a tutto il gioco di relazioni che si viene a determinare) tra questo e quel valore.

7) a partire dalle sequenze di dati "astratti" di P': {d'}, {e'}, {f'}, vengono determinate alcune specifiche di base sui valori degli inviluppi di certi parametri dello strumento, secondo le modalità seguenti:

a) ad ogni fonema della poesia corrisponde un evento della composizione; cio' vale separatamente par ogni strumento in gioco e per ogni frase;

b) i valori di durata determinano la durata degli inviluppi di tutto l'evento corrispondente; tali valori vengono interpretati a partire da una base dei tempi (unità di misura) decidibile dal compositore e variabile, ma costante all'interno di una stessa frase;

c) i valori di energia determinano il valore stazionario dell'inviluppo d'ampiezza globale;

d) i valori di pitch determinano un punto di passaggio obbligatorio dell'inviluppo di frequenza.

3) La determinazione dell'isomorfismo

La matrice $\underline{A}$, che determina in pratica la trasformazione lineare per il passaggio dai dati concreti (di partenza) ai dati astratti (usati per la sintesi sonora), è stata calcolata tramite il procedimento seguente:

1) si scelgono 3 fonemi particolari come autovettori (colonna) dell'operatore lineare;

2) si scelgono i rispettivi autovalori $\lambda 1, \lambda 2, \lambda 3$;

3) a questo punto, per noti risultati di teoria delle matrici [Gantmacher, 1950; Ruberti-Isidori, 1975], la $\underline{A}$ è completamente determinata e può calcolarsi tramite la

$$\underline{A} = \underline{U} \quad \underline{L} \quad \underline{U}^{-1}$$

dove $\underline{U}$ è la matrice degli autovettori colonna, e $\underline{L}$ è la forma canonica dell'operatore:

$$\underline{L} = \begin{vmatrix} \lambda 1 & 0 & 0 \\ 0 & \lambda 2 & 0 \\ 0 & 0 & \lambda 3 \end{vmatrix}$$

Il concetto di autovettore è quindi associato, nel caso in esame, a un dato fonema che, nella trasformazione lineare, è trattato in modo da alterare le sue caratteristiche prosodiche (durata, energia, pitch) per un medesimo fattore di scala (che è l'autovalore associato). Esistono dunque nella composizione tre poli ideali, tre punti di riferimento nei quali il substrato propriamente poetico emerge in modo

iù unitario ed evidente: a questi poli tutta la composizione, nel
uo articolarsi, fa inevitabilmente costante riferimento, ora tendendo
i essi, ora da questi allontanandosi.

Passando all'attuazione pratica, si sono scelti come
utovettori i fonemi

o	di	"a volte"
e	di	"e d'angoscia"
i	di	"Abbazia"

ioè i valori

$$\begin{vmatrix} 21 \\ -4.11 \\ 8.79 \end{vmatrix} , \begin{vmatrix} 1 \\ 10.91 \\ -10.69 \end{vmatrix} , \begin{vmatrix} 1 \\ -3.67 \\ 2.67 \end{vmatrix}$$

a matrice $\underline{U}$ è ottenuta semplicemente dalla giustapposizione dei tre
ettori:

$$\underline{U} = \begin{vmatrix} 21 & 1 & 1 \\ -4.11 & 10.91 & -3.67 \\ 8.79 & -10.69 & 2.67 \end{vmatrix}$$

ili autovalori scelti sono i seguenti:

$$\lambda 1 = 2 \qquad \lambda 2 = -3 \qquad \lambda 3 = 5$$

.a matrice $\underline{A}$ che ne risulta è

$$\underline{A} = \begin{vmatrix} 2.173 & -1.624 & -1.173 \\ -6.073 & 20.036 & 22.942 \\ 5.445 & -15.402 & -18.209 \end{vmatrix}$$

e si può vedere facilmente che, quando applicata ai tre fonemi di cui sopra, la trasformazione definita dalla $\underline{A}$ realizza effettivamente un'alterazione "omogenea" delle terne associate ai fonemi:

- **o**: $\underline{A} \begin{vmatrix} 21 \\ -4.11 \\ 8.79 \end{vmatrix} = \begin{vmatrix} 42 \\ -8.22 \\ 17.58 \end{vmatrix} = 2 \begin{vmatrix} 21 \\ -4.11 \\ 8.79 \end{vmatrix}$

- e: $\underline{A} \begin{vmatrix} 1 \\ 10.91 \\ -10.69 \end{vmatrix} = \begin{vmatrix} -3 \\ -32.73 \\ 32.07 \end{vmatrix} = -3 \begin{vmatrix} 1 \\ 10.91 \\ -10.69 \end{vmatrix}$

- i: $\underline{A} \begin{vmatrix} 1 \\ -3.67 \\ 2.67 \end{vmatrix} = \begin{vmatrix} 5 \\ -18.35 \\ 13.35 \end{vmatrix} = 5 \begin{vmatrix} 1 \\ -3.67 \\ 2.67 \end{vmatrix}$

Per ogni altro fonema, invece, la trasformazione opererà in modo diverso per le tre componenti: ad esempio, per il fonema *e* di *dentro* si ha:

- e: $\underline{A} \begin{vmatrix} 71 \\ 9.3 \\ 9.7 \end{vmatrix} = \begin{vmatrix} 127.8 \\ -22.31 \\ 66.73 \end{vmatrix} = \begin{vmatrix} 1.8 \times 71 \\ -2.4 \times 9.3 \\ 6.88 \times 9.7 \end{vmatrix}$

Qui, come si vede, per effetto delle interazioni di ciascuna componente sulle altre due, il valore della durata risulta

moltiplicato per il fattore 1.8, l'energia per -2.4, e il pitch per 6.88 . Per altri fonemi, ovviamente, si avranno ancora altri coefficienti di scala, pur nell'unitarietà della procedura.

4) Gli strumenti

Prima di passare a precisare le modalità di assegnazione degli inviluppi per i vari parametri degli strumenti, è opportuno premettere la descrizione di questi ultimi, per avere un panorama completo delle grandezze in gioco e del loro significato.

La realizzazione pratica della composizione è stata fatta per mezzo del sistema di sintesi/trattamento del segnale 4C, controllato in tempo reale da un calcolatore D.E.C. PDP-11/55. Per ogni informazione sulle caratteristiche di funzionamento e sulla programmazione della 4C, si rimanda direttamente a [Moorer, et al., 1979].

Gli strumenti in gioco nella composizione sono in tutto quattro; ciascuno di essi segue, in linea di massima, una partitura che gli è propria, in polifonia con gli altri. Dato che due degli strumenti richiedono ciascuno circa l'80 % delle risorse disponibili nella 4C (a livello di unità di sintesi o trattamento), ciascuna partitura è stata realizzata separatamente (in tempo reale) e registrata, e in seguito si è provveduto al missaggio dei quattro canali risultanti. Gli strumenti sono ordinati gerarchicamente, dato che i primi due hanno un peso nettamente preponderante nel missaggio finale, gli altri intervenendo solo qui e là a completare l'ossatura del discorso (e il terzo strumento compare più spesso del quarto). Tale ordinamento è inversamente proporzionale al grado di "trasparenza" del testo originale nel segnale prodotto da ciascuno strumento, come risulterà chiaro dalla descrizione di questi ultimi.

L'elemento di base e principio ispiratore di tutto il pezzo,

come s'è già detto, è la voce umana con tutte le sue sfaccettature, più o meno nascoste - e quindi eventualmente da rimettere in luce - a livello musicale; in questo senso, anche la definizione degli strumenti non fa eccezione, riprendendo alcune tecniche di sintesi numerica della voce [Flanagan-Rabiner, 1973].

I primi due strumenti, SA ed SF, sono ciascuno una elaborazione particolare di una interconnessione per la 4C, già utilizzata in passato dall'autore per varie ricerche, realizzante un sintetizzatore di voce per formanti a connessione in parallelo. Lo strumento SA è mostrato nelle figg. 1 e 4: la descrizione che ne faremo - e ciò vale anche per gli altri strumenti - sarà necessariamente alquanto sommaria, dato che la precisazione di ogni dettaglio esulerebbe largamente dai limiti di spazio fissatici. Questo strumento consiste, in prima approssimazione, in un sintetizzatore a formanti in parallelo e un oscillatore indipendente, che può andare a modulare separatamente (secondo tre profondità di modulazione diverse) ciascuno dei tre formanti prima della loro accumulazione. I tre filtri di formante sono realizzati nella 4C con una tecnica particolare, messa a punto recentemente all'IRCAM; la loro struttura e le caratteristiche sono descritte brevemente in [Asta,1979]. I loro parametri di controllo (ce ne sono 4 per filtro, ossia 12 in tutto: coefficiente moltiplicativo d'ingresso, frequenza di risonanza, fattore di merito, coefficiente moltiplicativo di uscita) sono stabiliti globalmente dal programma di controllo in funzione di un unico parametro, letto nella partitura, che è la vocale associata all'evento.

Oltre al valore della vocale, SA è regolato in tutto da altri 8 parametri, tutti variabili tra 0 e 1 (intendendo che 1 sia il massimo valore dei parametri a 16 bits espressi in complemento a 2, cioè 32767): infatti ogni eventuale offset o fattore di scala è compreso

nello strumento.

La lista dei parametri, con una breve descrizione delle loro influenze sul risultato sonoro dello strumento, è la seguente:

- FFP frequenza di pitch dell'oscillatore che eccita i filtri di formante. 0 = 60 Hz - 1 = 118 Hz
- BP bilanciamento tra sorgente periodica e di rumore. 0 = solo rumore, 1 = solo segnale periodico
- RS ricchezza spettrale del generatore d'eccitazione dei filtri. 0 = massima attenuazione dello spettro (l'eccitazione è pressoché sinusoidale), 1 = massima ricchezza: lo spettro è all'incirca piatto per le prime 21 armoniche, quindi decresce. Ciò vale indipendentemente dalla frequenza fondamentale dell'eccitazione, dato che il filtro che controlla lo spettro è asservito al pitch
- PM1 profondità di modulazione del primo formante (la modulazione è di tipo a banda laterale doppia, portante intera). 0 = modulazione nulla; 1 = 100 % di modulazione
- PM2 idem, per il secondo formante
- PM3 idem, per il terzo formante
- RF rapporto tra frequenza modulante e frequenza portante (che è la frequenza di pitch), m/c. 0 = m/c = 0.25×10^{-4}, 1 = m/c = 3
- AA ampiezza globale dello strumento. 0 = ampiezza nulla, 1 = ampiezza massima.

Lo strumento SF, mostrato nelle figg. 2 e 4, è analogo al primo: in esso le uscite dei tre generatori di formante, controllate

da tre diversi indici di modulazione, vanno a modulare in frequenza un oscillatore finale; si ritrovano in esso praticamente gli stessi parametri che in SA, con le eccezioni che seguono:

- per FPP, i limiti sono diversi: 0 = 90 Hz, 1 = 117 Hz
- per RS, la situazione di massima ricchezza spettrale è costituita da uno spettro circa piatto per le prime 15 armoniche, decrescente in seguito
- i parametri PM1, PM2, PM3 sono sostituiti dai tre indici di modulazione I1, I2, I3: 0 = modulazione nulla, 1 = modulazione massima (indice di modulazione effettivo pari a 8,79)
- il parametro RF non esiste, in quanto il rapporto c/m è qui fisso al valore 1.5 .

Si noti, inoltre, che l'ampiezza globale dello strumento è controllata da un'interconnessione alquanto più sofisticata di quella dello strumento SA: ciò perchè, nel caso di SA, l'ampiezza del segnale è influenzata largamente dai valori del parametro RS, che contribuisce alla sua varietà; ciò non è invece per SF, dove AA è il solo responsabile delle evoluzioni del livello sonoro globale (*).

La forma d'onda usata per l'oscillatore finale è normalmente una mistura delle prime 10 armoniche, prese con fase alterna e ampiezza sempre unitaria; in alcune sezioni, per contro, si è usata una sinusoide pura.

Il terzo strumento SFn, assai più semplice dei precedenti, è

(*) questo è solo un esempio delle tante considerazioni che siamo stati condotti a fare nel decidere questo o quel dettaglio dei vari strumenti, e lo stesso dicasi ovviamente per le altre fasi del lavoro compositivo; è evidente che non è possibile darne una lista, ancorchè incompleta. Abbiamo peraltro cercato, qua e là nel testo, di inserirne qualche cenno.

costituito (figg. 3 e 4) da un oscillatore analogo a quello di SF, con frequenza portante data dal parametro FPP, modulato in frequenza da un generatore costituito dalla voce originale (convertita numericamente e inviata, in tempo reale, in uno dei registri della 4C) e da un fattore di conversione ampiezza/frequenza dato dalla media aritmetica dei tre parametri d'indice di modulazione dello strumento SF. SFn utilizza le stesse partiture di SF, tenendo però in conto solo un sottoinsieme dei parametri da queste controllati (precisamente, dunque, FPP, I1, I2, I3, AA).

Il quarto strumento SVn, infine, è semplicemente costituito dal testo originale registrato, ripetuto per tutta la durata del pezzo. Per esso, ovviamente, non c'è da prendere in conto nessuna partitura.

5) Il ruolo dei parametri fonetici e le partiture

La filosofia di base delle procedure d'assegnazione degli inviluppi dei vari parametri (fermo restando tutto quanto deciso a partire dai dati prosodici) è questa:

1) ad ogni frase del testo corrisponde una partitura completa per ciascuno strumento; si hanno in tutto, dunque, due gruppi di quattro partiture ciascuno (il primo gruppo contiene le partiture per SA, il secondo gruppo le partiture per SF e SFn).

2) in queste partiture, ad ogni fonema della frase originale corrisponde, come già detto, un evento: dunque l'assegnazione a un evento di un insieme di inviluppi per i vari parametri dello strumento può farsi, e viene fatta, in base a considerazioni sul fonema associato. La tavola 1 mostra una possibile classificazione delle consonanti della lingua italiana in funzione delle loro caratteristiche fonetiche: il raggruppamento per colonne è funzione del luogo d'articolazione (salvo per le liquide), mentre le diverse

righe distinguono il modo d'articolazione e la sonorità [Chapman, 1971].

3) per un primo gruppo di parametri, comprendente in realtà il solo AA (strumento SA), gli inviluppi sono derivati da un unico modello, estremamente semplice: un trapezio che sale al valore stazionario nel 30 % della durata totale dell'evento, vi resta per il 60 % del tempo totale, e quindi ritorna a zero (il tempo totale e il valore stazionario sono determinati dalle regole enunciate a partire dai dati prosodici).

4) per un secondo gruppo di parametri, comprendente FPP, si hanno 6 possibili modelli, tutti costituiti da un solo segmento, dipendenti da due parametri: la posizione nel segmento del valore nominale di frequenza (che può trovarsi all'inizio, al centro o alla fine del segmento) e la direzione del segmento (ascendente o discendente). In tutti i casi, la pendenza è fissa. I due parametri di controllo del modello sono assegnati - eccezionalmente - senza vincolo di alcuna regola. Il valore nominale di frequenza è anch'esso determinato dalle regole enunciate a partire dai dati prosodici.

5) per un terzo gruppo di parametri, comprendente BP e RS, vengono operate delle partizioni - diverse per ciascun parametro - sull'insieme dei fonemi; ad ogni sottoinsieme della partizione viene assegnato un (diverso) modello d'inviluppo (ogni parametro ha i suoi modelli). Per BP, la partizione è basata sulla dicotomia vocale - consonante e sul modo d'articolazione delle consonanti; si hanno otto modelli, in corrispondenza dei sottoinsiemi seguenti: vocali, plosive sonore, plosive sorde, nasali e liquide, fricative sorde, fricative sonore, affricate sorde, affricate sonore. Per RS, invece, la partizione è stata operata tenendo in conto essenzialmente il luogo d'articolazione, e si hanno cinque modelli per i sottoinsiemi: vocali e liquide laterali, liquide vibranti, labiali, alveolari, velari.

l per il quarto gruppo, comprendente il parametro "vocale" (per il ontrollo dei filtri di formante), viene assegnata ad ogni fonema una ocale secondo queste regole: se il fonema è una vocale, questa stessa unge da vocale assegnata; se si tratta di una consonante, le viene ssegnata una vocale scelta aleatoriamente, con l'esclusione della ocale che immediatamente precede o segue (secondo una segmentazione recedemente operata nel testo). Le vocali assegnabili alle onsonanti sono scelte in un ambito legermente più ampio di quello lle sette vocali italiane usuali: si sono aggiunte tre vocali entralizzate (presenti in alcuni dialetti italiani e in altre ingue), una terza "e" di apertura intermedia tra le prime due, una seudo-vocale che corrisponde a un suono sordo (tutti e tre i formanti ono estremamente bassi) e un elemento di silenzio (che viene anche ssegnato invariabilmente a tutte le pause). La vocale così eterminata viene trascritta nella partitura, e costituisce un arametro non numerico (l'unico) che viene interpretato in tempo reale al programma di controllo del sintetizzatore; per il suo inviluppo che regola qui le modalità di transizione dalla vocale precedente lla nuova) vi sono due modelli, uno per i fonemi (originali) vocalici uno per le consonanti; il primo è una transizione graduale verso la uova vocale per il 70 % della durata dell'evento, seguita da uno tato stazionario, il secondo è invece un valore fisso (cioè la vocale iene raggiunta bruscamente nelle primissime unità di tempo, e antenuta fino alla fine).

) il quinto gruppo (parametro RF) combina le procedure dei due gruppi recedenti: si considerano non i fonemi, ma le vocali ad essi ssociate, e se ne effettua una partizione, basata essenzialmente ull'altezza del secondo formante; si hanno cinque modelli, per ltrettanti sottoinsiemi

) infine per il sesto gruppo, comprendente i parametri AA (strumento

SF), PMk e Ik (k = 1,2,3), viene utilizzata una procedura "con memoria"; si considerano due attributi del fonema: il luogo d'articolazione (secondo la partizione in cinque sottoinsiemi introdotta per RS) e la vocale assegnata al fonema precedente (secondo la partizione usata per il gruppo precedente); questi due elementi vengono combinati secondo una legge di composizione che porta su un insieme ancora a cinque elementi, a ciascuno dei quali corrisponde un modello (per AA) o una terna di modelli (per PMk e Ik). La legge di composizione utilizzata è definita nella tavola 2; essa è isomorfa (a parte la diversità degli insiemi a cui è applicata) all'operazione moltiplicativa nel campo delle classi resto modulo 5 [AYRES, 1965].

Tutti i modelli sono stati ovviamente scelti tenendo conto del significato e dell'influenza dei vari parametri sul suono sintetizzato, sulla base di una pratica manuale dello strumento, effettuata in precedenza, che ha permesso di acquisire la necessaria familiarità con esso.

In base alle regole stabilite e qui enunciate, vengono dunque generate le partiture (secondo una sintassi opportuna, sulla quale però non ci soffermeremo; la tavola 3 ne mostra un esempio) con un procedimento semiautomatico, tramite un programma ad hoc che applica le regole e decide in conseguenza; il risultato è scritto in alcuni files, in formato ASCII; questi vengono poi dati in ingresso a un compilatore, che traduce la partitura in un formato interno binario; i files prodotti dal compilatore sono caricati direttamente in memoria, e interpretati in tempo reale, dal programma di controllo del sintetizzatore.

Come si vede, esiste anche qui una gerarchia tra le varie categorie strutturanti definite dalle procedure ora esposte: alcune regole, per questo o quel parametro, sono molto semplici (si traducono quindi in una struttura musicale più piana, più immediata a

cogliersi), le altre via via più complesse, sia in se stesse, sia in quanto poggiano a loro volta sui risultati di regole, di livello inferiore, già enunciate ed applicate. Ciò crea appunto, nella struttura totale, diversi gradi di evoluzione dell'opera, diversi livelli di accessibilità e di profondità del messaggio veicolato, che si schiuderanno man mano - tale almeno è la nostra speranza - con l'iterarsi dell'ascolto.

Un'ultima categoria è infine presente nella composizione, rappresentata da una regola che abbiamo voluto chiamare "del disordine": la libertà cioè, per il compositore, di turbare in alcuni punti l'equilibrio definito da questa incastellatura di regole, tramite alterazioni arbitrarie di certi parametri; questa regola è stata fatta intervenire, in pratica, soprattutto dopo la fase di generazione delle partiture, effettuando alcune correzioni manuali nei files generati. Il suo uso, comunque, è stato estremamente contenuto.

6) La partitura generale

L'ultimo passo per la composizione dell'opera consiste nello stabilire come e dove utilizzare le varie partiture, con particolare riguardo alle relazioni temporali tra le sezioni dei vari strumenti.

Il materiale concettualmente a disposizione è il seguente:

- partiture generate dalle quattro frasi per lo strumento SA, che chiameremo SA1, SA2, SA3, SA4;
- partiture generate dalle quattro frasi per lo strumento SF: SF1, SF2, SF3, SF4;
- le partiture del punto precedente applicate allo strumento SFn;
- la pseudo-partitura (fissa) del quarto strumento SVn.

Per ogni partitura (salvo l'ultima) occorre specificare la base dei tempi, ovvero l'unità di misura con la quale interpretare i valori temporali; e, per SFk (k = 1,2,3,4), la forma d'onda

dell'oscillatore finale.

Non tutto il materiale è stato utilizzato per la realizzazion dell'opera; d'altra parte, in un caso (SF1) una partitura è stat impiegata due volte, con diverse basi di tempo. In un altro cas (SA2), la base dei tempi essendo molto grande, la partitura è stat utilizzata solo in parte.

Il disegno della partitura generale dell'opera è mostrato i fig. 5; il materiale ottenuto secondo le sue direttive viene dispost su quattro piste, una per strumento, e ogni pista è generata frase pe frase. Infine, le quattro piste così realizzate vengono mixat insieme. Il mixaggio non è regolato da nessuna partitura.

7) Conclusione

A VOLTE è solo un primo tentativo (per quanto ci riguarda) d conciliare in una composizione alcune esperienze personali, nate i occasioni e in tempi diversi ma non per questo incompatibili L'esperienza compositiva di un brano basato sulla voce umana h richiesto via via, per logico sviluppo dei fatti, di affrontar problemi e scelte di vario tipo: dalle considerazioni poetiche d partenza ad altre di natura ora fonetica, ora matematica, ora tecniche di trattamento del segnale, e così via; sembra a noi - abbiamo cercato di evidenziarlo tramite l'analisi di questo cas concreto - che la conoscenza di tutte queste discipline (e di altr ancora, certamente) non potrà chè intervenire sempre più spess nell'esperienza compositiva moderna, onde il loro studio (almeno livello di conoscenze di base) appare divenir necessario per i bagaglio tecnico-culturale del musicista di oggi.

Ringraziamento

A VOLTE è stata realizzata all'IRCAM di Parigi su

sintetizzatore 4C ideato e costruito da Peppino Di Giugno. A lui va naturalmente il mio ringraziamento, per avermi fatto conoscere una macchina capace di tante meraviglie. Grazie anche a Jean-Sylvain Liénard, per avermi iniziato all'arte di divertirsi a far parlare i calcolatori (faire "causer" les ordinateurs); a Luciano Berio, che mi ha incoraggiato ad applicare questo divertimento alla musica; e poi a Lucio Lombardo-Radice, Guido Baggiani, Antonio Ruberti, Giorgio Nottoli, Monique Bétrémieux, Aldo Roveri, Francesco Galante, Nerina Latini, Marc Battier, Herbert Brün, Philippe Prévot, Dante Alighieri, Andy Moorer, Maurizio Testa, eccetera eccetera: tutti costoro, anche se non vorranno mai crederlo, hanno avuto un ruolo fondamentale per la creazione di A VOLTE.

Parigi, maggio 1981

Bibliografia

- ASTA, V., 1979, Unità elettronica per la modifica in tempo reale di suoni strumentali in segnale vocale - 7. Congresso dell'Associazione Italiana di Acustica, Siena
- AYRES, F., 1965, Modern Algebra - Mc Graw-Hill, New York
- CHAPMAN, W.H., 1971, Introduction to practical phonetics - Summer Institute of Linguistics, Merstham. Trad. it.: Introduzione alla fonetica pratica, Officina Edizioni, Roma 1972
- FLANAGAN, J.L., RABINER, L.R., 1973, Speech synthesis - Dowden, Hutchinson & Ross, Stroudsburgh, Pennsylvania
- GANTMACHER, F. R., 1950, The theory of matrices - New York
- MOORER, J.A., ABBOTT, C.W., CHAUVEAU, A., EASTTY, P.C., LAWSON, J.R., 1979, The 4C machine - Computer Music Journal, vol. 3, n. 3
- RUBERTI, A., ISIDORI, A., 1975, Teoria dei sistemi - Siderea, Roma

fig. 1 - strumento SA

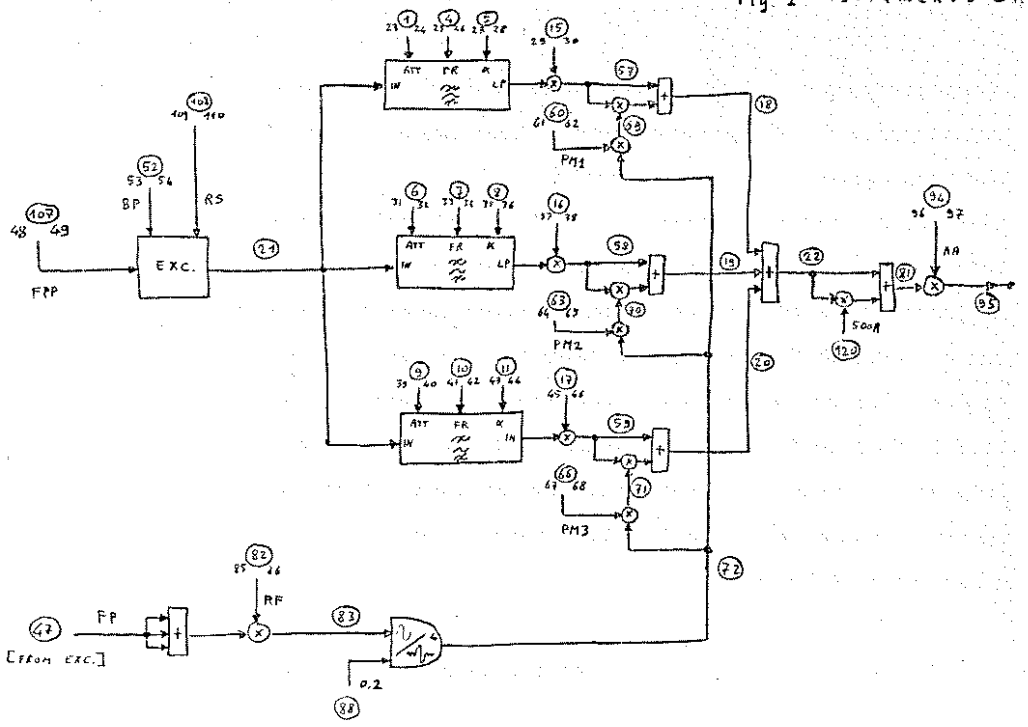
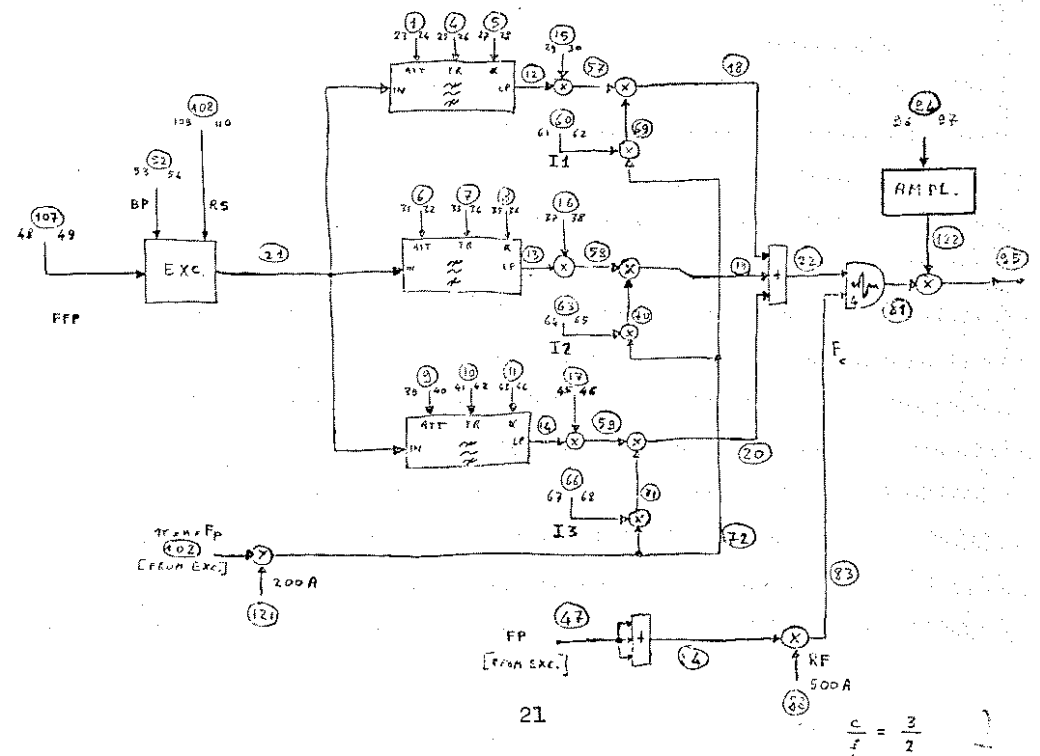


fig. 2 - strumento SF



$\frac{c}{f} = \frac{3}{2}$

fig. 3 - strumento SFn

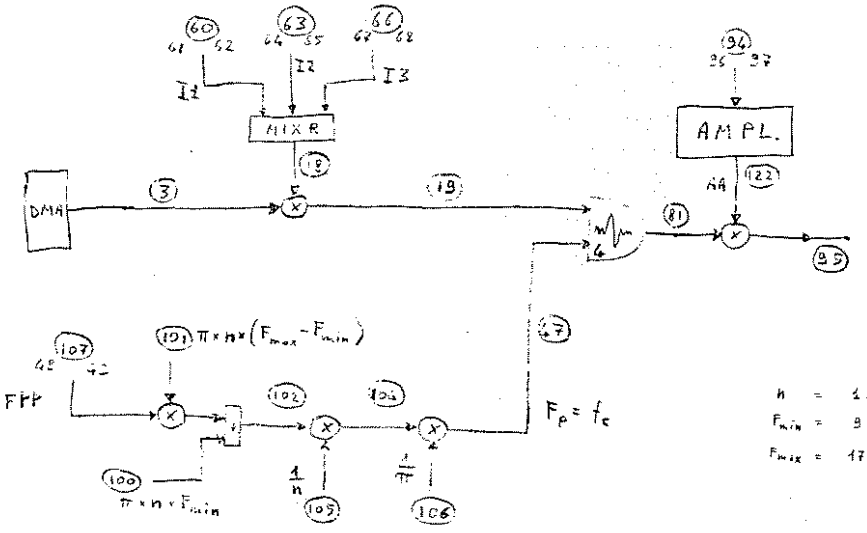
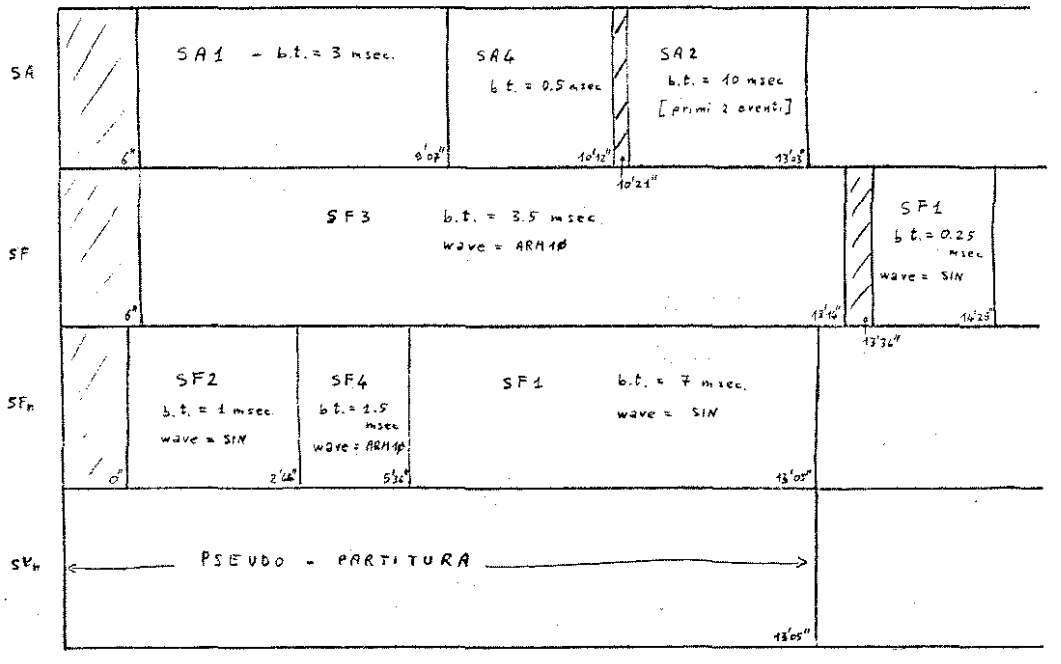


fig. 5 - partitura generale



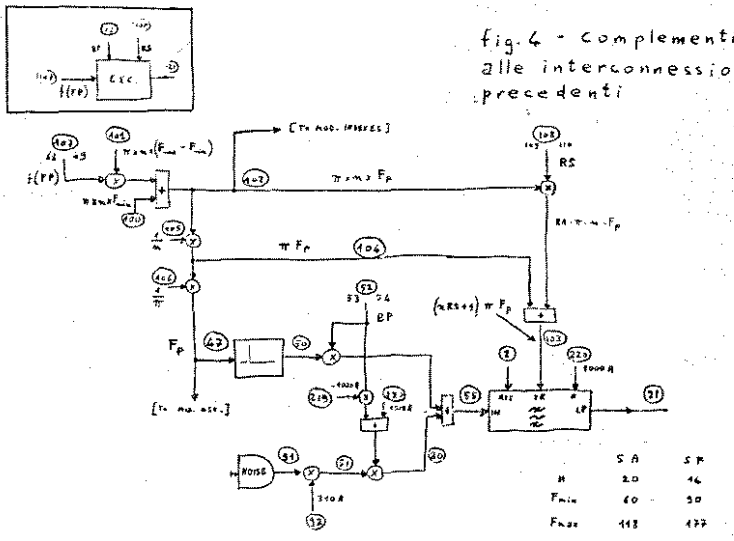


fig. 4 - complementi alle interconnessioni precedenti

Tavola 1
classificazione delle consonanti della lingua italiana

	labiali	alveolari	velari
plosive sorde	p	t	k
plosive sonore	b	d	g
nasali	m	n	ŋ
fricative sorde	f	s	ʃ
fricative sonore	v	z	ʒ
affricate sorde		ts	ʧ
affricate sonore		dz	ʤ

	vibranti	laterali alveolari	laterali palatali
liquide	r	l	ʎ

Tavola 2
legge di composizione per l'attribuzione dei modelli ai parametri del sesto gruppo

		hh Ø	a0,a3, zz 1	e1,e2, ae 2	u0,u3, o1,o2 3	i0,i3 4
liquide vibranti	Ø	Ø	Ø	Ø	Ø	Ø
labiali	1	Ø	1	2	3	4
alveolari	2	Ø	2	4	1	3
velari	3	Ø	3	1	4	2
vocali e liquide laterali	4	Ø	4	3	2	1

note - in scala verticale sono i fonemi attuali, in orizzontale le vocali assegnate al fonema precedente; ogni vocale è rappresentata da una lettera e una cifra; la cifra ha il significato seguente: Ø = vocale normale, 1 = vocale stretta, 2 = vocale larga, 3 = vocale centralizzata; ae = terza realizzazione della e (la più larga); zz = pseudo-fonema sordo; hh = elemento di silenzio

Tavola 3
esempio di partitura: SF1, parte relativa al parametro "vocale"

```
;       dentro ci prende a volte un vuoto
;       fonemi 1 - 25

  tavvoc:

        ae,  109,10851

        e1, 7418, 3179

        o2,   89, 8855

        e2,   48, 4821

        zz,   39, 3889

        o1, 7903, 3387

        a3,  103,10271

        i0, 4235, 1815

        i0,   43, 4279

        i0,   41, 4075

        ae, 6313, 2706

        e2,   87, 8662

        zz,   43, 4294

        rand1
        e1,a0
        rand2
        49
        rand3
        1
        endr, 8706

        hh,   71, 7061

        o2, 6049, 2593

        zz,  103,10210

        zz,   66, 6546

        e1, 3007, 1289

        u0, 7359, 3154

        zz,   50, 5039

        u0, 2991, 1282

        o2, 8413, 3606

        e2,   46, 4555

        o1, 4922, 2110
        FINE,-1,0,0
        END
```

A MUSICAL GAME AND ITS OPTIMAL SOLUTION

Tommaso Bolognesi

CNUCE-C.N.R.
Reparto Musicologia

Via S.Maria 36
56100 PISA

.iassunto

Viene definito un gioco che produce sequenze su un alfabeto
.i eventi sonori. I due giocatori, a turno, estendono di un even-
.o sonoro la sequenza parziale, cercando rispettivamente di massi-
.izzare e minimizzare il 'grado di armonicità' (qui definito arbi-
.rariamente) della sequenza finale.

La tecnica dell'α-β pruning é utilizzata per ridurre la di
.ensione dell'albero delle scelte e minimizzare il tempo di ricerca
.ella soluzione ottima (quella prodotta da due giocatori bravissimi).

.bstract

A game that produces music sequences over an alphabet of sound
:vents is defined. Two players, in turn, extend of one sound event
:he partial sequence, trying respectively to maximize and minimize
:he 'degree of harmonicity' (here arbitrarily defined) of the final
:equence.

The α-β pruning technique is applied to reduce the size of
:he game tree and minimize the work to find the optimal solution
:that of a match played by the best players).

0. Introduction

A frequently occurring idea in the description of a music composition process is that of a mediation between two opposite principles: order (rules) and disorder (their violation). Some rules, or patterns, shared by the composer and the listener as a common basis, must be imbedded in the composition in order to let it be a communication process; on the other hand rules must be broken once in a while (establishing perhaps new rules) in order to reduce redundancy and create elements of surprise and interest in the listener.

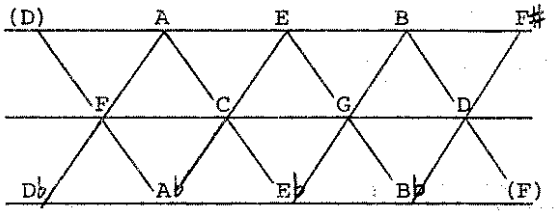
A similar idea is adopted in the game described here. Simple rules for chord transition are given, based on a geometric representation for a set of 24 fundamental chords suggested by E.Gagliardo ([1]) (this simple set was chosen as a first example, but other alphabets of sound events could be defined and used as well). These rules are never broken by the two players, but they try to apply them in the best/worst way in order to maximize/minimize the value of the final solution. The resulting model for automatic composition is elementary but has an advantage over some other computer music composition models (of equivalent simplicity): it takes into account both local (chord transitions) and global (overall degree of harmonicity) properties of the generated sequences

1. How to extend a partial solution ?

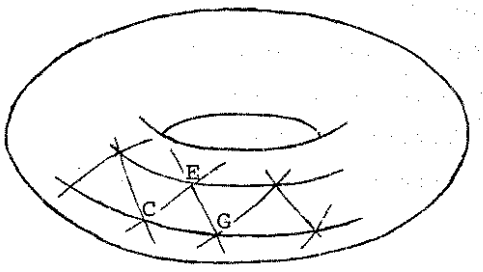
An infinite periodic graph structure (figure 1a) is obtained by letting nodes represent the 12 pitches of the chromatic scale (we do not distinguish between octaves) and edges represent only a $\pm$5th interval (like C-G), a $\pm$ major 3rd interval (like C-E) and a $\pm$ minor 3rd interval (like A-C). A finite representation of this

graph is suitably given on the thorus (figure 1b)([1]). Major chords (like C maj = $\{C, E, G\}$) are represented by triangles ($\triangle$), minor chords (like A min = A, C, E) by upside-down triangles ($\triangledown$); furthermore, the geometric disposition of chords is consistent with the aural perception.

We arbitrarily associate to each major and minor chord a reasonable ordered set of 5 neighbors, as illustrated in Figure 2. Each player will extend a partial sequence of chords by choosing a legal neighbor of the last (current) chord. Hence a 5-ary tree (game tree) is associated with the sequence of choices, where a node represents a chord, and its ordered sons represent the only possible successors.

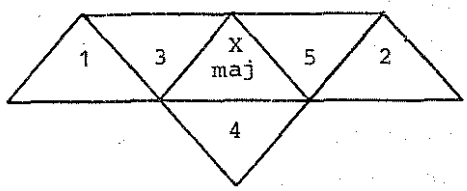


a)

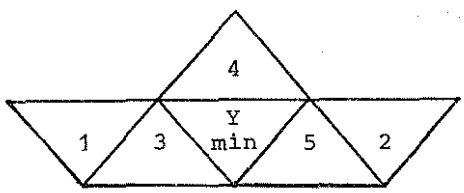


b)

Figure 1 - Major and minor chords as triangles on the thorus.



Example - Neighbors of Cmaj:
(Fmaj, Gmaj, Amin, Cmin, Emin)



Example - Neighbors of Amin:
(Dmin, Emin, Fmaj, Amaj, Cmaj)

Figure 2 - Ordered neighbors of major and minor chords.

2. How to evaluate solutions ?

Our simple set of notes contains 12 elements: C, C sharp, D...B. In order to define the harmonic content of a chord, or a sequence of chords, a note may be associated to its first 7 harmonic components : hence a note may be seen as a set of 7 notes (let us forget those slight frequency differences between harmonic components and well tempered pitches...). Example:

	note		fundam. frequency	harmonic components					
	C	=	C	C	G	C	E	G	Bflat
freq. (Hz)	100		100	200	300	400	500	600	700

The harmonic content of, say, note C may be synthetically represented by this multiset:

$$C = \{3.C,\ 2.G,\ 1.E,\ 1.Bflat\},$$

where numbers represent the 'multiplicity' of the associated note, i.e. the number of times it appears in the multiset. Similarly a chord, i.e. a set of three notes, may be represented by the union of the multisets associated to the three notes:

$$Cmaj = \{5.G,\ 4.E,\ 3.C,\ 3.D,\ 3.B,\ 1.F,\ 1.Gsharp,\ 1.Bflat\}$$

(and similarly for a minor chord).

Fundamental chords sound pleasant and 'stable' because there is a considerable overlapping between the harmonic components of the three notes.

We extend this idea to a sequence of chords: the harmonic contributions of each chord in the sequence are cumulated in an array of 12 elements, called HARMONY, which represents a sort of overall harmonic spectrum of the sequence. The degree of overlapping between harmonic components, i.e. the tendency of few harmonics to predominate in the sequence, may be measured in a simple way by the inverse of the 'gravity center' of the sorted array HARMONY:

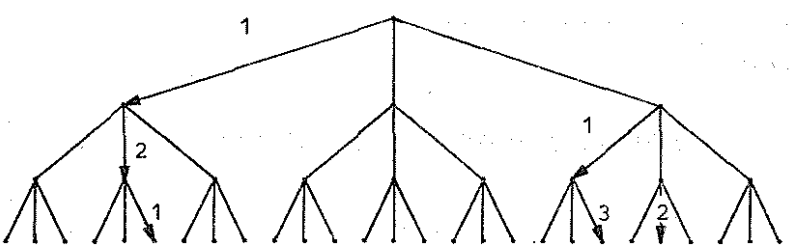
$$HARMONY(a_1) \geqslant HARMONY(a_2) \geqslant \ldots \geqslant HARMONY(a_{12})$$

$$Harmonicity = \left[\frac{\sum_{i=1}^{12} i.HARMONY(a_i)}{\sum_{i=1}^{12} HARMONY(a_i)} \right]^{-1}$$

3. Implementation and results

A PASCAL program has been written to find chord sequences of

variable length generated by optimal matches, along with their 'harmonicity' values (as a first maj/min chord, Cmaj/Amin was chosen without loss of generality). The amount of computation required to exhaustively search the game tree is reduced by pruning it with the α-β pruning technique described, for instance, in ([2]), chapter 4. Memory space to run the program is also contained because no explicit tree structure is implemented. Although α-β values are virtually assigned to the nodes of a 5-ary tree, and partial solutions (paths along the tree) have to be simultaneously kept in memory, the only data structures implemented are a one dimensional array for the α-β values and a matrix for the partial solutions. A possible configuration of the partial solutions to be kept during computation as paths on the game tree, along with its matrix representation, is given in Figure 3 (for simplicity, here the branch factor is 3).



1	-	-
2	1	
1	3	2

Figure 3 - A matrix to represent current partial solutions in a game tree.

Keeping in mind the numbering of chords from 0 to 23, the neighbors of a given major chord X or minor chord Y are obtained by adding , mod 24, the proper constants to the given chord (number):

Neighbors (1st relation)

	1st	2nd	3rd	4th	5th	
X maj (C)	X+10 (F)	X+14 (G)	X+19 (a)	X+1 (c)	X+9 (e)	(mod 24)
Y min (a)	Y+10 (d)	Y+14 (e)	Y+15 (F)	Y+23 (A)	Y+5 (C)	(mod 24)

(Major/minor chords are now represented by upper/lower case letters, for simplicity. Neighbors of C and a are shown as an example).

Comments on 'Table of results'.

- Chord sequences 1 through 10 appear to be almost 'standard' chord sequences, plain and easy to catch. The aural impression is clearly that of a single tonality (no modulation), except for f's in sequences 6 and 7, and c's in sequences 9 and 10.

Notice that solutions 1 through 5 are very similar to each other, but short sequences are not always (trivially) prefixes of longer sequences (compare 3 and 4). Furthermore sequences 6 and 7 (7 extends 6) are very different from the previous ones (yet they all start with the same chord, C). Similarly, 9 is only an extension of 8, but 10 is both a (slight) modification and an extension of 9.

The strategy of the first player (who looks for 'harmonicity') is obvious: dispersion of harmonic components is bounded by always repeating the same chord (in even positions : recall that the first chord is not chosen by a player). On the other hand we might expect a random-like behavior from the second player (who looks for 'inharmonicity', i.e. dispersion of harmonic components), but this

is not precisely the case, as some repetitions are found also in his choices (odd chords). As a result some of these sequences (example: "Cadadaea") appear to be periodic sequences (adada) with boundary effects (C ... ea), i.e. an opening and a conclusion.

- In order not to allow the repetition of the same chord every second position, a second neighborhood relation has been adopted:

	Neighbors (2nd relation)				
	1st	2nd	3rd	4th	5th
X maj	X+14	X+8	X+18	X+19	X+5
(C)	(G)	(E)	(A)	(a)	(d)
Y min	Y+14	Y+8	Y+18	Y+15	Y+9
(a)	(e)	(c)	(f)	(F)	(D)

The resulting sequences (11 through 14) are more complex than the previous ones and show tonality modulations; the new neighborhood relation helps the second player and 'harmonicity' values are sensibly decreased (see table), at the point that they are close to typical 'harmonicity' values of random chord sequences. (Yet it is easy to find simple melodic lines fitting sequence 13).

- Notice how 'harmonicity' values seem to depend exclusively on the adopted neighborhood relation.

- When α-β pruning is applied, the number of evaluated solutions (to evaluate a solution means, in this game, to compute its 'harmonicity') is smaller than the number of possible solutions (see table). Notice that the number of evaluated solutions is proportional (after a short transient) to the program execution time (sequences with execution time indicated on table were comp-

	Length of sequence	First chord	Neighborhood relation	SOLUTION	Harmonicity	Number of evaluated solutions	Number of possible solutions	Ratio	Execution time (seconds)
1	4	C	1	C a d a	.247	57	125	0.46	0.142
2	5	C	1	C a d a e	.234	229	625	0.37	0.216
3	6	C	1	C a d a e a	.239	775	3125	0.25	0.750
4	7	C	1	C a d a d a e	.234	2972	15625	0.19	3.043
5	8	C	1	C a d a d a e a	.239	10165	78125	0.13	11.091
6	9	C	1	C F B♭F B♭F C F f	.233	30892	390625	0.08	-
7	10	C	1	C F B♭F B♭F C F f F	.236	107373	1953125	0.05	-
8	8	a	1	a C G C F C G C	.240	11897	78125	0.15	-
9	9	a	1	a C G C F C G C c	.235	47872	390625	0.12	-
10	10	a	1	a C F C G C G C c C	.239	166667	1953125	0.09	-
11	8	C	2	C G E G♯ a♯ D♯ A♯ g	.189	8067	78125	0.10	-
12	10	C	2	C a f♯ a♯ g E♭ G a c♯ A	.184	144288	1953125	0.07	-
13	8	a	2	a F A E B g♯ C♯ G♯	.189	13471	78125	0.17	-
14	10	a	2	a e A b e♭ g E♭ G E B	.185	137272	1953125	0.07	-

- Table of results -

computed on a Cyber 175, while an IBM 370/168 generated the remaining ones). This means that the savings in the number of solution evaluations is immediately reflected in an equivalent savings in execution time. Notice also how this savings increases with the size of the tree (column 'ratio'). One sequence out of twenty had to be evaluated, for instance, to generate sequence 7.

4. Conclusions

Interesting algorithms for automatic composition may be designed, where two opposite tendencies determine the final result, as opposed to algorithms for the straightforward maximization of a given function.

When this idea is applied and the solution space is exhaustively searched with backtrack techniques, α-β pruning may be used to reduce the amount of computation.

The compositional technique here introduced with a very simple example, may be applied any time an alphabet of sound events is defined, along with transition rules and a way to evaluate complete results.

5. Bibliography

1 - E.Gagliardo, P.Fornasari : "Composizione di musica classica mediante elaboratore elettronico", QUADERNI DI INFORMATICA, V, 2 (1979).

2 - E.M.Reingold, J.Nievergelt, N.Deo : "Combinatorial Algorithms Theory and Practice", Prentice Hall, 1977.

RIFLESSIONI SULLA REALIZZAZIONE TECNICA DI CHEMINS V

V.Asta - IRCAM, Paris; AXIS DIGITAL, Paris.
P.Prevot - IRCAM, Paris.

CHEMINS V, di Luciano Berio, per clarinetto e sintetizzatore digitale, é stato realizzato all'IRCAM di Parigi e rappresentato in prima al Théatre d'Orsay nel maggio 1980.

In questa memoria si espongono i vari problemi tecnici che questa realizzazione ha richiesto via via di risolvere, problemi dettati in gran parte dalla portabilità (in sala da concerto) dell' apparato tecnico connesso alla composizione. La soluzione di tali problemi si é tradotta per lo piu' in una serie di accorgimenti concernenti sia l'hardware che il software, che si sono dovuti adottare per permettere un comportamento soddisfacente, dal punto di vista della sicurezza e della versatilità d'uso a un tempo, nel far svolgere ad una macchina relativamente lenta e con piccola estensione di memoria un lavoro richiedente interventi rapidi (per il tempo reale) e due generi di trattamento affatto diversi fra loro.

CHEMINS V by Luciano Berio, for clarinet and digital synthesizer has been realized at IRCAM, Paris. The first performance was given at the "Theatre d'Orsay" in May 1980.

In this paper we describe the various technical problems that this realization has gradually required to solve.

These problems have mostly been caused by transportability (to/from the concert-hall) of the technical apparatus needed for the performance.

A series of hardware and software solutions had to be adopted to have a relatively slow machine, with a small extention of memory, operate rapidly (real time) and satisfactorily from the point of view of security and versatility.

DESCRIZIONE DI PROCESSI MUSICALI PER MEZZO DI OPERATORI GEOMETRICI: UN ESEMPIO APPLICATIVO.

Stefania De Stefano - Istituto di Matematica - Università di Milano
Goffredo Haus - Istituto di Cibernetica - Università di Milano
Alberto Stiglitz - Istituto di Cibernetica - Università di Milano

Riassunto.

In questo lavoro è descritto un esempio di applicazione di operatori geometrici per la descrizione di processi musicali.

In particolare, viene utilizzato un formalismo derivato dalla nozione di omologia come linguaggio per la rappresentazione e la trasformazione di testi musicali.

Il sistema notazionale proposto costituisce un primo passo verso l'individuazione e la definizione di strumenti descrittivi sintetici particolarmente vicini ai metodi di composizione musicale assistita da elaboratore.

Abstract.

In this work we describe an example of geometric operators application for the description of musical processes.

Particularly, we use a formalism derived from the notion of homology as a language for the representation and transformation of musical texts.

The notational system which we are proposing is a first step towards the individuation and definition of synthetic descriptive tools very close to methods of musical composition aided by computer.

INTRODUZIONE.

Uno degli aspetti più rilevanti della applicazione dell'elaborazione automatica in campo musicale è senz'altro quello dei linguaggi e, più in generale, dei formalismi con cui è possibile descrivere fenomeni musicali.

Inoltre, la possibilità, introdotta con la sintesi digitale del suono, di controllare tutti i parametri acustici se, da una parte, apre spazi sonori inesplorati, dall'altra obbliga ad una specificazione completa, e quindi più onerosa, dei parametri acustici.

Diventa quindi molto opportuno pensare a forme di descrizione che consentano notazioni più sintetiche e più astratte (Bertoni et al., 1978) rispetto alla specificazione a livello di processi acustici, e più generali e complete rispetto alla notazione tradizionale delle partiture.

Nel lavoro (De Stefano/Haus, 1980) è stato proposto di applicare la nozione geometrica di omologia per la rappresentazione di processi musicali descritti in uno spazio multidimensionale.

In questo lavoro descriviamo brevemente un linguaggio che permette di rappresentare processi musicali nel piano altezze-tempi per mezzo di operatori geometrici di tipo omologico.

Il programma che accetta questo linguaggio è scritto interamente in Pascal (Grogono, 1980).

Nella prima parte vedremo le definizioni fondamentali che costituiscono la premessa alla definizione del linguaggio; nella seconda parte vedremo le caratteristiche generali del linguaggio e una descrizione sommaria dei comandi implementati; nella terza parte, infine, vedremo un esempio molto semplice di composizione musicale sviluppato con la notazione proposta in questo lavoro.

DEFINIZIONI FONDAMENTALI.

1. Un omologia piana φ è una proiettività di un piano su se stesso che possiede una retta s luogo di punti uniti: l'asse di φ.

Un'omologia possiede anche un altro punto unito S (centro di φ), che può appartenere ad s.

E' noto che due punti omologhi A e $A'=\varphi(A)$ giacciono su una retta passante per S (unita per φ).

Si osservi che il punto S può essere un punto all'infinito: ciò significa che le rette (unite) congiungenti punti omologhi sono tutte parallele tra loro. Invece il fatto che la retta s sia la retta allo infinito del piano significa che ogni retta viene mutata in una retta avente la stessa direzione (quindi l'omologia φ conserva gli angoli).

Un'omologia φ è definita noti S, s e una coppia di punti omologhi.

Ad esempio se $S\in s$ e s è la retta all'infinito, l'omologia φ è individuata quando sia nota una coppia A, A' di punti corrispondenti. Infatti per ogni punto B del piano, $\varphi(B)$ è intersezione della retta per B parallela ad AA' e della retta per A' parallela ad AB. Dunque φ è una traslazione individuata dal vettore $\vec{AA'}$.

Se $S\notin s$, l'omologia è completamente determinata da S, s e dalla caratteristica k di φ ($k=\frac{\vec{SA}}{\vec{SA'}}\cdot\frac{\vec{UA'}}{\vec{UA}}$, ove U è il punto di intersezione tra s e la retta AA').

Se S è un punto all'infinito (e $S\notin s$) l'omologia φ è dunque individuata da s, da $k=\vec{UA'}/\vec{UA}$ e dall'angolo α formato tra s e le rette congiungenti punti omologhi.

Nel seguito assumeremo sempre $\alpha=\pi/2$, cioè assumeremo che φ sia un'affinità ortogonale. Inoltre supporremo $k\neq 1$ perché per $k=1$ si ha la trasformazione identica.

Se s è la retta all'infinito (e $S\notin s$) l'omologia φ è detta omotetia ed è individuata da S e da $k=\vec{SA}/\vec{SA'}$. Si vede facilmente che - denotate con s_1 e s_2 due rette per S tra loro ortogonali - l'omotetia è prodotto dell'affinità ortogonale di asse s_1 e caratteristica $k_1=1/k$ e dell'affinità ortogonale di asse s_2 e caratteristica $k_2=1/k$.

Nel seguito faremo uso delle omotetie, sempre vedendole come trasformazioni composte: quindi ne trascuriamo la descrizione.

2. Supponiamo di introdurre nel piano un sistema di riferimento cartesiano ortogonale.

Vediamo quali sono gli effetti di traslazioni e affinità ortogonali su una configurazione originaria C.O. .

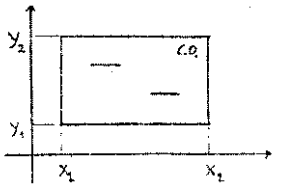


Figura 1: la scelta di questa particolare C.O. e del tipo di rappresentazione sarà giustificata dalle successive discussioni.

2.1 TRASLAZIONI

Non alterano la C.O. che viene solo trasportata parallelamente nel piano. Una traslazione è completamente determinata da un vettore di traslazione $[a,b]$.

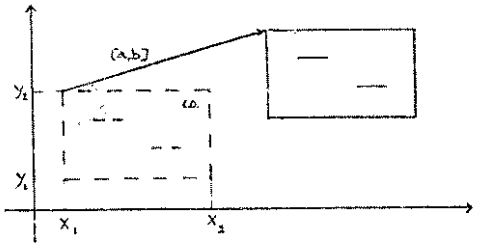


Figura 2: traslazione individuata dal vettore $[a,b]$ con $a = |x_2 - x_1| + 2$ e $b = 2$.

2.2 AFFINITA' ORTOGONALI

Il loro effetto dipende dal valore di k e dalla posizione di s. Per quanto attiene al valore di k si hanno i seguenti casi:

a) Se $k > 0$, $k \neq 1$, il corrispondente di A giace nello stesso semipiano di A rispetto ad s poiché $\overrightarrow{UA'} = k\overrightarrow{UA}$.

Un segmento AB parallelo ad s va in un segmento $A'B' = AB$ parallelo ad s.

Un segmento CD perpendicolare ad s va in un segmento C'D' che ha lunghezza pari a k volte quella di CD.

In generale una figura risulta molto deformata (non solo dilatata

o contratta come nel caso delle omotetie): ad esempio un rettangolo può esser trasformato in un parallelogrammo.

b) Se $k = -1$ si ha una simmetria ortogonale: la configurazione trasformata è speculare dell'originaria.

c) Se $k < 0$, $k \neq -1$, l'affinità ortogonale può esser vista come prodotto di una simmetria di asse s e di una affinità ortogonale di asse s e caratteristica $k' = -k$.

Quindi nel successivo studio dell'effetto delle affinità ortogonali in dipendenza dalla posizione dell'asse s, ci si potrà limitare a considerare i casi $k > 0$ e $k = -1$.

Per quanto concerne la posizione di s, si hanno le seguenti situazioni:

a) L'asse s è verticale: allora i punti trasformati hanno la stessa ordinata dei punti della configurazione originaria.

Se $k > 0$ le distanze tra le ascisse di due punti sono contratte o dilatate a seconda che sia $0 < k < 1$ oppure $k > 1$.

Se $k = -1$ si ha un ribaltamento.

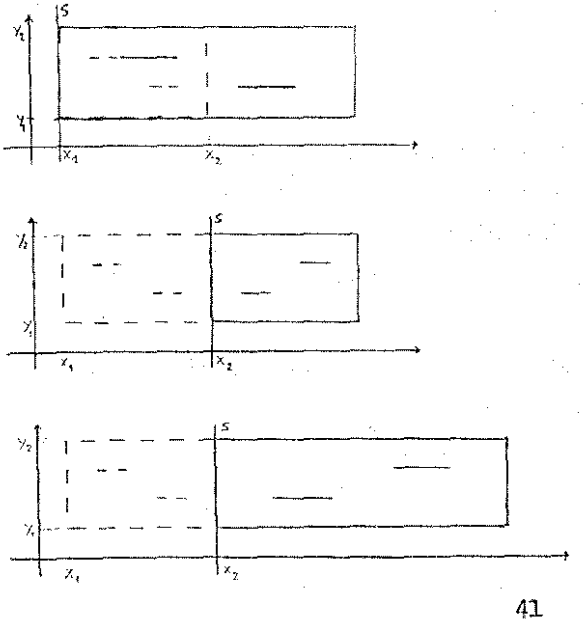


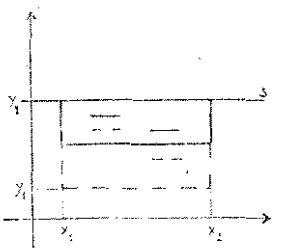
Figura 3:

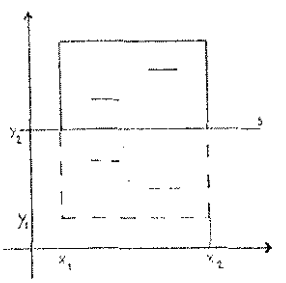
a) affinità ortogonale con asse s di equazione $x = x_1$ e caratteristica $k = 2$;

b) simmetria ortogonale ($k = -1$) con asse s di equazione $x = x_2$;

c) affinità ortogonale con asse s di equazione $x = x_2$ e caratteristica $k = -2$; si osservi che si può ottenere questa trasformazione applicando alla C.O. la trasformazione illustrata in b) e alla nuova C.O. la trasformazione a).

b) L'asse s è orizzontale: in questo caso la trasformazione agisce sulle distanze tra le ordinate dei punti della C.O., mentre le distanze tra le ascisse restano inalterate.
Gli effetti sono analoghi a quelli riscontrati nel caso a).





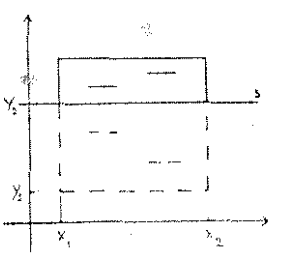
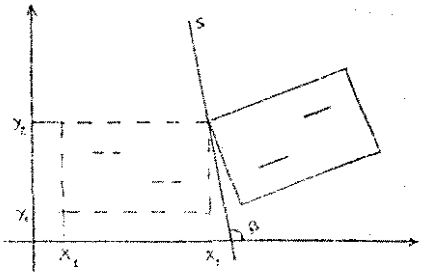


Figura 4: a) affinità ortogonale con asse s di equazione $y = y_2$ e caratteristica $k = 1/2$;

b) simmetria ortogonale con asse s di equazione $y = y_2$;

c) affinità ortogonale con asse s di equazione $y = y_2$ e caratteristica $k = -1/2$; anche in questo caso la trasformazione è il risultato dell'applicazione successiva di a) e di b).

c) L'asse s è obliquo: gli effetti sono difficilmente descrivibili a priori. In generale segmenti orizzontali sono mutati in segmenti obliqui.



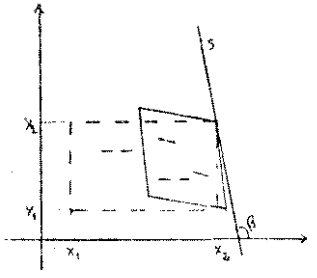


Figura 5: nelle figure a lato sono illustrate due affinità ortogonali rispetto all'asse s passante per $P = (x_2, y_2)$ e formante con l'asse x un angolo $\beta = 100°$; nel primo caso $k = -1$, nel secondo $k = 1/2$; si noti che nel secondo caso il rettangolo viene deformato in un parallelogrammo non rettangolo.

3. Finora sono state date le definizioni di alcune omologie in modo del tutto generale. Non è stato fatto cioè alcun riferimento esplicito ad eventuali applicazioni musicali; solamente la scelta degli esempi grafici è stata implicitamente condizionata dai previsti collegamenti col discorso musicale.

In questa e nelle successive sezioni tale discorso viene inserito gradualmente nel contesto più generale già delineato, in modo da chiarire progressivamente le caratteristiche e i reciproci collegamenti.

3.1. Per prima cosa è stato determinato l'insieme delle omologie che verranno utilizzate in fase applicativa.

3.1.1. In base alle esigenze specifiche sono stati formulati dei criteri per la selezione dei valori dei parametri:

- le trasformazioni devono produrre, entro un certo margine di sicurezza, dei risultati significativi anche relativamente al contesto musicale;

- occorre limitare i campi di variabilità dei parametri - ovvero limitare le possibilità di sperimentazione perché l'utilizzatore possa controllarle più facilmente - senza tuttavia ledere la generalità delle relative omologie;
- bisogna limitare per quanto possibile il numero dei controlli che dovranno essere effettuati in fase operativa sui valori dei parametri e sui risultati delle trasformazioni, per non arricchire troppo il programma con dettagli che finirebbero per appesantire, in questa prima fase, il lavoro di sperimentazione.

3.1.2. Per comprendere meglio il seguito, è necessario dare una veste musicale alla rappresentazione utilizzata negli esempi grafici (cfr. §2); quindi d'ora innanzi verranno identificati

l'asse delle ascisse e l'asse dei tempi

l'asse delle ordinate e l'asse delle altezze.

Quindi una configurazione geometrica rappresenterà una forma sonora (S.F.) e i suoi elementi rappresenteranno oggetti sonori (S.O.).

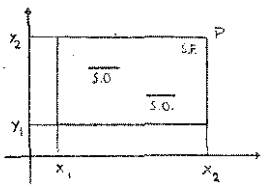


Figura 6: per motivi che si chiariranno in seguito anche le pause sono considerate oggetti sonori.

3.1.3. Vediamo in particolare le limitazioni imposte ai parametri con le relative giustificazioni.

i) Traslazioni. I parametri caratteristici sono costituiti dalle componenti del vettore $[a,b]$.

Si impongono le seguenti limitazioni:
- ad a viene sempre assegnato automaticamente un valore per difetto e non è di interesse per l'utente (cfr. §4);
- b viene supposto limitato, $|b| \leq B$, ove B può variare a seconda delle esigenze dello specifico lavoro. Nel nostro caso particolare, si è imposto $B = (|y_2 - y_1| + 2)$.

ii) Affinità ortogonali. I parametri caratteristici sono costituiti dall'asse s e dalla caratteristica k.

a) Se l'asse s è verticale (x=c), per ragioni di semplicità si considerano solamente le seguenti possibilità:

$$c = x_1 \quad e \quad k \in \{1/2,\ 2,\ 3\}$$
$$c = x_2 \quad e \quad k \in \{-3,\ -2,\ -1,\ -1/2\}.$$

b) Se l'asse s è orizzontale (y=c), per ragioni di semplicità si considerano solo le seguenti possibilità:

$$c = y_1 \qquad \text{oppure} \qquad c = y_2$$
$$e \quad k \in \{-3,\ -2,\ -1,\ -1/2,\ 1/2,\ 2,\ 3\}.$$

c) Se l'asse s è obliquo, esso viene individuato assegnando un punto $P \in s$ e l'angolo β che s forma con l'asse delle ascisse.

Ovviamente, per non ricadere nel caso a), dev'essere $\beta \neq 90°$. Si impone inoltre $\beta \neq 45°$ e $\beta \neq 135°$, poiché per tali valori di β S.O. paralleli all'asse dei tempi vengono trsformati in S.O. di durata nulla e quindi privi di significato musicale.

Per ragioni di semplicità si considerano solamente le seguenti possibilità:

$$\begin{cases} P = (x_2, y_2) \\ \beta \in \{100°, 105°, 110°, 115°, 120°, 125°\} \end{cases} \quad \text{oppure} \quad \begin{cases} P = (x_2, y_1) \\ \beta \in \{55°, 60°, 65°, 70°, 75°, 80°\} \end{cases}$$
$$k \in \{-2,\ -1,\ -1/2,\ 1/2,\ 2\}.$$

3.1.4. Si osservi per concludere che in fase operativa sarà possibile cambiare anche radicalmente i campi di variabilità dei parametri in modo molto semplice; questa possibilità conferma il fatto che le limitazioni imposte sui valori dei parametri non compromettono la generalità del lavoro.

Inoltre, in fase operativa gli apparenti problemi di sovrapposizione delle S.F. trasformate con le S.F. generatrici non sussistono: infatti le coordinate relative alle componenti di tutte le S.F. saranno espresse in termini simbolici in modo da poter operare indipendentemente su ciascuna.

IL LINGUAGGIO IMPLEMENTATO.

4. In questo paragrafo viene descritto il linguaggio implementato (con particolare riferimento ai comandi che lo caratterizzano e al modo di utilizzazione delle omologie).
E' bene premettere che lo studio è in fase avanzata solo per quanto riguarda una parte del linguaggio complessivo (ECHO), ed è tale parte che verrà descritta dettagliatamente nelle prossime sezioni. Essa riguarda la definizione del piano melodico relativo ad uno spazio sonoro (cfr. 2), ovvero la componente tempo/altezze di una partitura musicale.
A questo proposito è stato realizzato un particolare programma PASCAL: PIANMEL.
Operando sui comandi predisposti e definendo valori particolari per i dati di ingresso, un utilizzatore può costruire un testo musicale simbolico relativamente alla componente melodica, per mezzo di trasformazioni omologiche (Haus, 1979).

4.1. Nel paragrafo precedente è stato determinato un particolare insieme finito di omologie la cui descrizione è stata fatta in termini geometrici. Ma per un musicista tale descrizione è di scarso interesse, per cui è necessario spiegare musicalmente gli effetti delle trasformazioni omologiche sulle S.F. generatrici.
A questo proposito sono state predisposte delle tabelle che consentono di ricavare le sequenze di simboli da immettere nel programma, per ottenere gli effetti desiderati (ved. pagg. 14, 15, 16).
Una struttura gerarchica di opzioni consente di caratterizzare passo passo le trasformazioni, percorrendo un itinerario di scelte successive.
Le opzioni riguardano ciascuna un particolare aspetto di una trasformazione, e il loro numero e la loro definizione sono stati prestabiliti in base a criteri soggettivi di minimizzazione e di significato, in relazione alle caratteristiche proprie delle omologie.
4.1.1. Sono possibili le seguenti opzioni:
I) Tipo di moto: consente di scegliere come deve risultare la S.F. trasformata rispetto alla sua generatrice per quanto riguarda l'altezza.

Indicati con g.s.o. i S.O. della S.F. generatrice e con t.s.o. i S.O. della S.F. trasformata, si hanno tre possibilità:

Ascendente (A): i t.s.o. sono più alti dei corrispondenti g.s.o.

Discendente (D): i t.s.o. sono più bassi dei corrispondenti g.s.o.

Orizzontale (O): i t.s.o. sono alla stessa altezza dei corrispondenti g.s.o.

II) Trasformato // Originale: in caso di A o D permette di scegliere se trasformare i g.s.o. in S.O. ad essi paralleli oppure no (cioè ad esempio: note in note oppure note in glissandi). Si hanno due possibilità:

Vero (V): i t.s.o. sono paralleli ai g.s.o.

Falso (F): i t.s.o. non sono paralleli ai g.s.o.

III) Effetto globale: in caso di AV o DV permette di scegliere se gli intervalli contenuti nella S.F. trasformata devono avere lo stesso verso di quelli contenuti nella S.F. generatrice oppure se deve essere invertito. Si hanno due possibilità:

	S.F. generatrice		S.F. trasformata
Riproduzione (R):	Intervalli ascendenti (discendenti)	rimangono	Intervalli ascendenti (discendenti)
Variazione (V): (inversione)	Intervalli ascendenti (discendenti)	diventano	Intervalli discendenti (ascendenti)

L'opzione, invece, in caso di OV permette di scegliere se i t.s.o. devono essere nello stesso ordine sequenziale dei corrispondenti g.s.o. oppure se tale ordine deve essere invertito (effetto di retrogradazione). Si hanno due possibilità:

Riproduzione (R) : S.O. riprodotti nello stesso ordine rispetto alla sequenza originale.

Variazione (V) (retrogradazione) : S.O. riprodotti in ordine inverso rispetto alla sequenza originale.

IV) Effetto locale: in caso di AVR o AVV, DVR o DVV, consente di scegliere come devono essere gli intervalli contenuti nella S.F. trasformata rispetto ai corrispondenti contenuti nella S.F. generatrice. Indicata con a l'ampiezza degli intervalli nella S.F. generatrice, e con a' l'ampiezza degli intervalli corrispondenti nella S.F. trasformata, si hanno tre possibilità:

Dilatazione (D): $a' > a$

Invarianza (N): $a' = a$

Contrazione (C): $a' < a$.

L'opzione, invece, in caso di OVR o OVV, consente di scegliere come devono essere le durate dei t.s.o. rispetto alle durate dei g.s.o. Indicate con d le durate dei g.s.o. e con d' le durate dei corrispondenti t.s.o., si hanno tre possibilità:

Dilatazione (D): $d' > d$

Invarianza (N): $d' = d$

Contrazione (C): $d' < d$.

V) L'ultima opzione consiste nello scegliere i valori numerici per i parametri delle omologie; gli effetti relativi ai singoli valori sono descritti in modo completo direttamente nelle tabelle A, B e C.

4.1.2. Si è detto che le tabelle servono per costruire delle sequenze di simboli da immettere nel programma in modo da far operare le trasformazioni desiderate. Le indicazioni per costruire tali sequenze si trovano nel diagramma sintattico relativo a <lista di parametri> (ved. §5); ma per costruirle correttamente bisogna tener conto di alcune regole supplementari, riguardanti la consultazione delle tabelle A, B e C.

a) nelle liste di parametri possono comparire solo codici o valori.

b) l'ordine nel quale devono essere effettuate le scelte rispecchia l'ordine progressivo delle opzioni indicato con numerazione romana nelle tabelle. Ovvero: se si legge una tabella da sinistra a destra e si prelevano nello stesso ordine i codici e i valori relativi alle scelte effettuate, si ottiene automaticamente la <lista di parametri> corretta.

c) in corrispondenza di ogni opzione bisogna prelevare il codice relativo ad una sola delle possibilità disponibili.

d) non si devono omettere codici o valori relativi ad opzioni che si trovano nell'itinerario prescelto.

Eventuali ambiguità potranno essere risolte tenendo presente quanto segue: la struttura delle tabelle rispecchia fedelmente la struttura delle

parti di programma corrispondenti, e quindi va rispettata rigidamente onde evitare errori nel funzionamento del programma.

N.B. Nei casi AF e DF in corrispondenza delle opzioni III e IV compare una frase esplicativa, ma non compare alcun codice: questo significa che tali codici vanno omessi nel costruire la <lista di parametri> corrispondente.

Esempi di liste di parametri corrette:

(AVRD,3) (OVVD,-2) (DVVC,-0.5) (AF,+0.5,100) (DF,-1,75) (OVRN,0)

5. In questo paragrafo viene descritto il linguaggio implementato, attraverso la definizione e la spiegazione dei comandi da cui è costituito. L'architettura del linguaggio è stata progettata in base ad alcune esigenze primarie:

a) Fornire al musicista strumenti di descrizione che siano aderenti il più possibile a quelli che gli sono congeniali nella pratica compositiva, in qualsiasi ambito o linguaggio musicale egli operi.

b) Nello stesso tempo tali strumenti non devono limitare per loro natura la fruizione delle notevoli potenzialità degli elaboratori.

c) Prevedere la possibilità di ulteriori sviluppi e ampliamenti del linguaggio, come ad esempio il suo inserimento in un ambito più completo per quanto riguarda controllo e gestione di parametri musicali e acustico-musicali, o in altre parole, modularità.

d) Prevedere la possibilità da parte del musicista di costruire e quindi inserire procedure "personali" ad alto livello (PASCAL) di controllo e/o generazione dei dati musicali.

e) Essere molto leggibile, compatibilmente con le esigenze operative, per rendere agevoli il più possibile i controlli sui comandi e sulle istruzioni da parte dell'utilizzatore.

I comandi sono di due tipi: comandi di sintesi (delle strutture) e comandi operativi (sulle strutture).

Le strutture sono organizzate gerarchicamente in diversi livelli.

I		II		III		IV		V			
Tipo di moto	codice	Trasformato // Originale	codice	Effetto globale	codice	Effetto locale	codice	Parametro 1 effetto	valore	Parametro 2	valore
Ascendente	A	Vero	V	Riproduzione	R	Dilatazione	D	Raddoppiamento dell'ampiezza degli intervalli	2		
								Triplicazione dell'ampiezza degli intervalli	3		
						Invarianza	N	Trasposizione di tanti intervalli elementari quanto è valore	{1.. ..(|y2-y1|+2)}		
						Contrazione	C	Dimezzamento dell'ampiezza degli intervalli	0.5		
				Variazione	V	Dilatazione	D	Raddoppiamento dell'ampiezza degli intervalli	-2		
								Triplicazione dell'ampiezza degli intervalli	-3		
						Invarianza	N	Invarianza dell'ampiezza degli intervalli	-1		
						Contrazione	C	Dimezzamento dell'ampiezza degli intervalli	-0.5		
		Falso	F	Gli effetti sono difficilmente determinabili in modo esatto				{-2,-1,-0.5, 0.5,2}		{100,105, 110,115, 120,125}	

Tab. 4

I		II		III		IV		V	
Tipo di moto	codice	Trasformato // Originale	codice	Effetto globale	codice	Effetto locale	codice	Parametro I effetto	valore
Orizzontale	O	Vero	V	Riproduzione	R	Dilatazione	D	Raddoppiamento delle durate	2
								Triplicazione delle durate	3
						Invarianza	N	Invarianza delle durate	0
						Contrazione	C	Dimezzamento delle durate	0.5
				Variazione	V	Dilatazione	D	Raddoppiamento delle durate	-2
								Triplicazione delle durate	-3
						Invarianza	N	Invarianza delle durate	-1
						Contrazione	C	Dimezzamento delle durate	-0.5

Tabella B

I		II		III		IV		V			
Tipo di moto	codice	Trasformato // Originale	codice	Effetto globale	codice	Effetto locale	codice	Parametro 1 effetto	valore	Parametro 2	valore
Discendente	D	Vero	V	Riproduzione	R	Dilatazione	D	Raddoppiamento dell'ampiezza degli intervalli	2		
								Triplicazione dell'ampiezza degli intervalli	3		
						Invarianza	N	Trasposizione di tanti intervalli elementari quanto è valore	$\{-1 .. -(y_2-y_1+2)\}$		
						Contrazione	C	Dimezzamento dell'ampiezza degli intervalli	0.5		
				Variazione	V	Dilatazione	D	Raddoppiamento dell'ampiezza degli intervalli	-2		
								Triplicazione dell'ampiezza degli intervalli	-3		
						Invarianza	N	Invarianza dell'ampiezza degli intervalli	-1		
						Contrazione	C	Dimezzamento dell'ampiezza degli intervalli	-0.5		
		Falso	F	Gli effetti sono difficilmente determinabili in modo esatto				$\{-2,-1,-0.5, 0.5,2\}$		$\{55,60, 65,70, 75,80\}$	

Tab. C

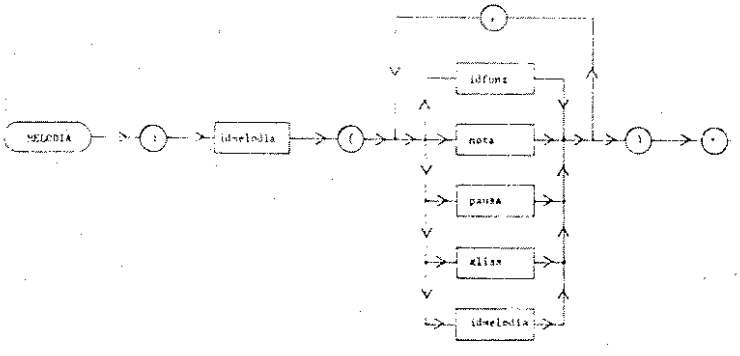
Struttura (sigla)	Livello	
Piano melodico (PIANMEL)	1	(livello più alto)
Voce (VOCE)	2	
Parte (PARTE)	3	
Unità armonica (UNARM)	4	
Accordo (ACCORDO), melodia (MELODIA)	5	
Nota (NOTA), pausa (PAUSA), glissando (GLISS), funzione (FUNZ)	6	(livello più basso)
Blocco (BLOCK)	-	(non appartiene ad alcun livello gerarchico)

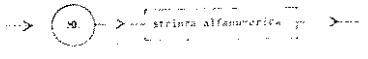
In corrispondenza di ciascuna struttura, ad esclusione di quelle del livello 6, esiste un comando di sintesi che consente al musicista di espandere la struttura desiderata (struttura musicale dal suo punto di vista; struttura di dati dal punto di vista operativo), semplicemente applicando le regole sintattiche descritte nei diagrammi sintattici.

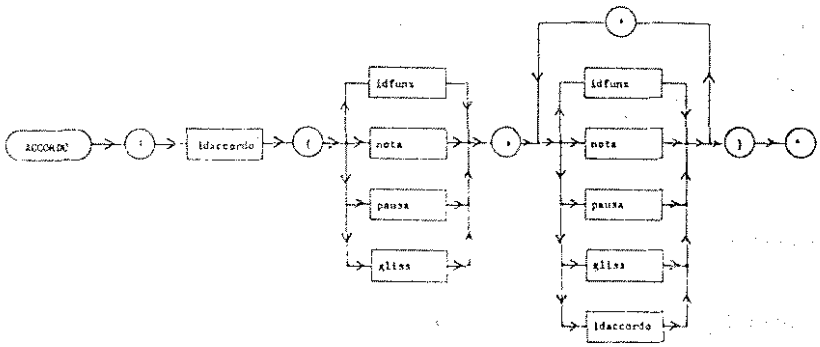
Nella sintassi di tali comandi sono compresi elementi e microstrutture → [ ] → le cui descrizioni, sempre sotto forma di diagrammi sintattici, sono contenuti in (Stiglitz, 1981).

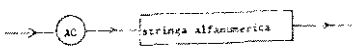
Nel tracciare i diagrammi si è tenuto conto, oltre che delle esigenze a, b, c, d, e, anche di ulteriori implicazioni indotte dalla costituzione dei singoli elementi e da esigenze di carattere operativo.

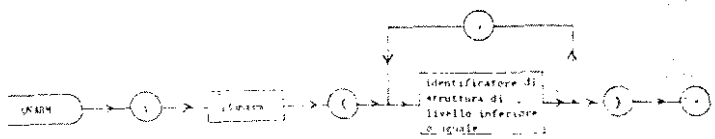
I comandi di sintesi sono: MELODIA, ACCORDO, UNARM, PARTE, VOCE, PIANMEL, BLOCK.

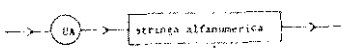


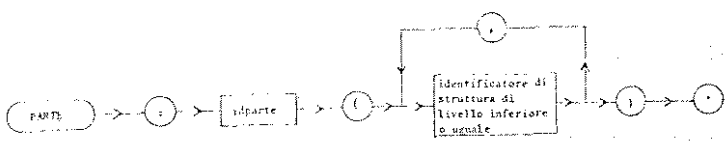


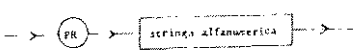

ACCORDO
:
idaccordo
(
idfunz
nota
pausa
slitta
,
)
;


AC
stringa alfanumerica

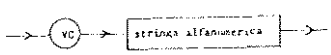

identificatore di
struttura di
livello inferiore
o uguale


UA
stringa alfanumerica

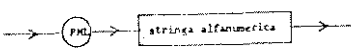

PARTE
idparte
identificatore di
struttura di
livello inferiore
o uguale

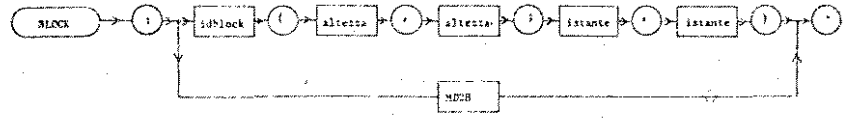

PR
stringa alfanumerica

VOCE
idvoce
identificatore di
struttura di
livello inferiore
o uguale

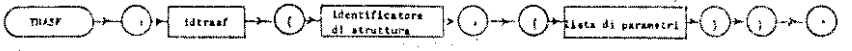

VC
stringa alfanumerica

PIANOL
idpianol
identificatore di
struttura di
livello inferiore
o uguale

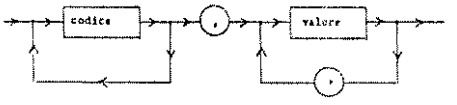

PNL
stringa alfanumerica



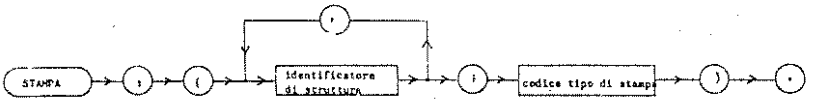
Oltre ai comandi di sintesi, come già si è visto, esistono dei comandi operativi, i quali permettono di agire su strutture già espanse.
Essi sono: TRASF, STAMPA, TEMPOR, COMPAUT.



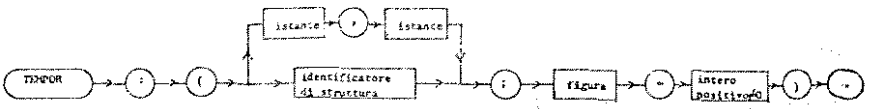
Lista di parametri:



Questo comando è strettamente legato alle trasformazioni omologiche descritte in §2 e in §3 ed è quello che permette di ricavare da strutture predefinite (S.F. generatrici) nuove strutture (S.F. trasformate) mediante istruzioni molto semplici; consente quindi di sviluppare e risolvere molteplici esigenze formali di carattere compositivo in modo organico e sintetico.
Si tenga presente che idtrasf è un identificatore di struttura dello stesso livello dell'identificatore di struttura in argomento.



Questo comando consente di visualizzare su di un'unità periferica di controllo i dati relativi ad una qualsiasi struttura espressi, a scelta de l'utente, in uno dei modi previsti (tipo di stampa).



Questo comando svolge una funzione analoga a quella di stabilire il tempo metronometrico per una struttura musicale.

Questo comando permette di cedere il controllo di una attività di gestione o generazione dei dati ad una particolare procedura realizzata da l'utilizzatore in modo opportuno.

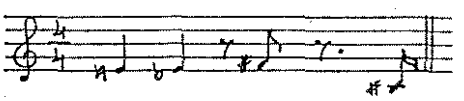
UN ESEMPIO APPLICATIVO.

6. In questo paragrafo viene descritto un esempio di utilizzazione della parte di linguaggio implementata e descritta nei paragrafi precedenti. La descrizione, per motivi di spazio e di contesto, è essenziale e pone l'accento prevalentemente sugli aspetti inerenti all'argomento dell'articolo.

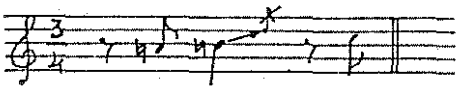
6.1 La composizione comprende quattro voci, ciascuna delle quali è costituita da due parti.

Le componenti di ciascuna parte sono generate da un tema base, mediante trasformazioni omologiche che corrispondono, come si è visto, a particolari trasformazioni delle strutture musicali (retrogradazione, inversione, trasposizione, ecc.). Si hanno così, originariamente, otto temi base, da cui traggono origine tutti gli altri elementi che costituiscono la composizione:

MELA

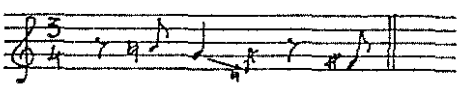


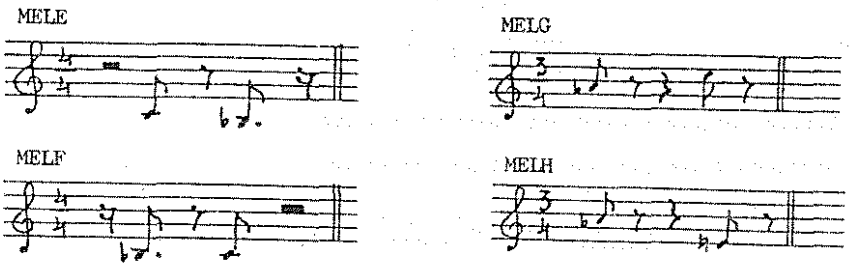
MELC



MELB

MELD





I risultati dell'elaborazione saranno riportati in una partitura musicale che consentirà di realizzare concretamente la composizione.

Lo strumento prescelto per la parte esecutiva è un sintetizzatore monofonico, la cui tastiera ha un'estensione di due ottave e mezza - (DO_i , FA_{i+2}) dove $\underline{i}$ è il numero d'ordine dell'ottava - tarata per semitoni temperati, e il cui registro è trasportabile a piacere.

Le otto parti sono registrate singolarmente e successivamente riunite in quattro piste su nastro magnetico mediante opportuno missaggio.

6.2 La scelta del modo di procedere ha introdotto alcuni vincoli (per l'utilizzazione del linguaggio) che vanno rispettati in fase di realizzazione. Viene fatto accenno ai più importanti:

- utilizzazione solo di struttture di tipo melodico e non armonico, dato che lo strumento è monofonico;

- ogni parte deve essere limitata (per quanto riguarda l'altezza) in una striscia di ampiezza due ottave e mezzo;

- gli intervalli elementari sono semitoni temperati; ne consegue che la notazione più opportuna da utilizzare sarà quella simbolico-alfabetica (sigle di note o di figure ritmiche).

Inoltre non vengono utilizzate le FUNZ come elementi base.

Le principali fasi dello sviluppo del progetto sono state le seguenti:

a) suddivisione del campo udibile musicalmente significativo in 4 strisce di ampiezza due ottave e mezzo;

b) collocazione delle 8 parti nelle strisce e conseguente trascrizione dei temi base in un sistema di riferimento cartesiano tempo/altezza;
c) descrizione dell'assetto formale delle singole parti in termini grafico-simbolici;

Siano A, B, C, P1, P2 simboli che identificano strutture formali.

Ogni struttura formale corrisponde ad una particolare sequenza di strutture melodiche ottenute da un tema base mediante trasformazioni omologiche.

Siano $\overline{A}$, $\overline{B}$, $\overline{C}$, $\overline{P1}$, $\overline{P2}$ i simboli delle strutture formali ottenute per retrogradazione rispettivamente da A, B, C, P1, P2; ricordiamo che $\overline{A}\,\overline{B} \neq \overline{AB}$, infatti $\overline{AB} = \overline{B}\,\overline{A}$; inoltre indichiamo con P1(MELA) la struttura formale P1 applicata al tema base MELA.

Utilizzando tali simboli, è possibile descrivere la struttura formale della composizione.

Tenendo conto che: P1 = AB($\overline{AB}$)C e P2 = AAB($\overline{AB}$)C$\overline{A}$, si può scrivere:

Voce1 { Parte1: P1(MELA) ; Parte2: $\overline{P1}$(MELB) }

Voce2 { Parte1: P2(MELC) ; Parte2: $\overline{P2}$(MELD) }

Voce3 { Parte1: P1(MELE) ; Parte2: $\overline{P1}$(MELF) }

Voce4 { Parte1: P2(MELG) ; Parte2: $\overline{P2}$(MELH) }

d) specificazione delle strutture formali, cioè descrizione delle sequenze di strutture melodiche che costituiscono le strutture formali;
e) formalizzazione della composizione in termini di linguaggio implementato.

Vediamo ora la descrizione della composizione-esempio ed alcune possibilità di utilizzo degli operatori definiti in §5 :

i) Descrizione della composizione:

Sintesi di MELA:

MELODIA: MELA (N(1,E;SMN),N(1,E,-(1,12,2);SMN),P(CRM),N(1,F,+(1,12,2);
CRM),P(CRM+SCR),N(1,C,+(1,12,2);SCR)).

MELA trasportata in alto di $|y_2-y_1|$ semitoni:

TRASF: MEI1 (MELA,(AVRN,($|y_2-y_1|$))).

MEI1 riprodotta con le durate dei suoi elementi raddoppiate:

TRASF: MELA1 (MEI1,(OVRD,2)).

MELA trasportata in basso di $|y_2-y_1|$ semitoni:

TRASF: MEI2 (MELA,(DVRN,-($|y_2-y_1|$))).

MEI2 riprodotta con le durate dei suoi elementi dimezzate:

TRASF: MELA2 (MEI2,(OVRC,0.5)).

MEI1 riprodotta con le durate dei suoi elementi triplicate:

TRASF: MELA3 (MEI1,(OVRD,3)).

MELA1 trasportata in basso di $|y_2-y_1|$ semitoni:

TRASF: MELA6 (MELA1,(DVRN,-($|y_2-y_1|$))).

Effetti difficilmente determinabili (si può dire però che le note vengono trasformate in glissandi):

TRASF: MELA7 (MELA6,(AF,-1,100)).

TRASF: MELA8 (MELA6,(AF,-2,100)).

TRASF: MELA9 (MEI1,(DF,-2,80)).

Sintesi di MEMA:

MELODIA: MEMA (MELA,MELA1,MELA,MELA2,MELA,MELA3,MELA,MEI2,MELA,MEI1,
MELA6,MELA7,MELA8,MELA1,MELA9).

MEMA retrogradata:

TRASF: MEMA1 (MEMA,(OVVN,-1)).

Sintesi di PRP1V1 (parte1 di voce1):

PARTE: PRP1V1 (MELA,MELA1,MELA,MELA2,MELA,MELA3, MELA,MEI2,MELA, } A

MEI1,MELA6,MELA7,MELA8,MELA1,MELA9, } B

MEMA1, } $(\overline{AB})$

MELA6). } C

Sintesi di PRP2V1 (parte2 di voce1); PRP1V1 retrogradata:

TRASF: PRP2V1 (PRP1V1,(OVVN,-1)).

Sintesi di VCUNO (voce1):

VOCE: VCUNO (PRP1V1,PRP2V1).

In modo in parte analogo si costruiscono le altre tre voci (cfr. §6.2,c), ottenendo: VCDUE, VCTRE, VCQTR. Si può quindi sintetizzare PIANMEL.

Sintesi di PMLALBA (piano melodico relativo alla composizione dell'esempio):

PIANMEL: PMLALBA (VCUNO,VCDUE,VCTRE,VCQTR).

ii) Procedura di stampa: si può scegliere ad esempio di stampare (in modo alfabetico-simbolico):

tutto PIANMEL :	STAMPA: (PMLALBA;SIMB).
oppure una sola voce:	STAMPA: (VCDUE;SIMB).
oppure più voci:	STAMPA: (VCDUE,VCQTR;SIMB).
oppure una sola parte:	STAMPA: (PRP1V4;SIMB).

e così di seguito.

iii) Se non si danno indicazioni, il tempo metronometrico assegnato a tutte le strutture sintetizzate assume un valore per difetto che corrisponde a ♩= 60. Nel caso in cui lo si desideri cambiare per tutto PIANMEL o per uno o più suoi elementi, è sufficiente utilizzare il comando TEMPOR. Ad esempio:

♩= 75 :	TEMPOR: (MPLALBA;SMN=75).
𝅗𝅥= 80 :	TEMPOR: (MPLALBA;MNM=80).
♪= 130:	TEMPOR: (VCUNO;CRM=130).
♩= 120:	TEMPOR: (MELA;SMN=120).

e così di seguito.

°° per la notazione musicale e la concezione formale della composizione-esempio vedi (Marie, 1976).

CONCLUSIONI.

La realizzazione di PIANMEL ha permesso di sperimentare parzialmente le possibilità che vengono introdotte utilizzando una notazione che fruisce della nozione di omologia ed è basata su un atteggiamento di descrizione strutturale.

Una sperimentazione più completa sarà possibile quando saranno implementate anche le procedure per la descrizione degli altri parametri sonori (essenzialmente quelli che riguardano la descrizione della timbrica) cioè quando sarà realizzato completamente ECHO (Electronic Composition by Homologies).

L'obiettivo finale di questo progetto è l'inserimento di ECHO tra gli strumenti per la descrizione del testo musicale ad alto livello (a livello strutturale) nel sistema informativo del Laboratorio di Informatica Musicale dell'Istituto di Cibernetica.

RIFERIMENTI BIBLIOGRAFICI.

1 - Bertoni/Haus/Mauri/Torelli: 'Analysis and Compacting of Musical Texts', Journal of Cybernetics, vol.8, pp. 257-272, Hemisphere Publ., 1978.

2 - De Stefano/Haus: 'A Mathematical Approach to the Representation of Musical Structures (oriented to the Transformation of Sonorous Forms)', Proceedings of 2nd International Conference on Data Bases in the Humanities and Social Sciences, 20 pp., Facultad de Informatica, Madrid, 1980.

3 - P.Grogono: Programming in Pascal (revised edition), Addison-Wesley, 1980.

4 - G.Haus: 'Trasformazione di testi musicali per mezzo di operatori', Atti del 3° Colloquio di Informatica Musicale, pp. 168-183, Università di Padova, 1979.

5 - J.E.Marie: L'homme musical, Arthaud, Paris, 1976.

6 - A.Stiglitz: 'Specifiche funzionali di PIANMEL', rapporto interno, Istituto di Cibernetica, Università di Milano, 1981.

Questo lavoro è stato realizzato nell'ambito del Communication and Programming Project tra Università degli Studi di Milano e Honeywell Information Systems Italia e con il contributo del Consiglio Nazionale delle Ricerche, Contratto N. 79.00701.02.115.9672.

INVITO ALLA COMPOSIZIONE ENNEDECAFONICA

(Toward enneadecaphonic music)

E. GAGLIARDO - M. GHISLANDI
dip. di Matematica - Università di Pavia

Riassunto: Vengono spiegate le motivazioni, le strutture e le regole fondamentali per la composizione di musica enneadecafonica, cioè nel sistema di toni le cui frequenze sono proporzionali ed alcune tra le potenze di $\sqrt[19]{2}$. Alcune battute di Beethoven con leggere modifiche risultano adatte per essere eseguite in questa nuova accordatura, e alcune nuove composizioni perderebbero senso musicale nella scala tradizionale. Si descrive uno schema matematico che è servito per programmare la composizione enneadecafonica mediante elaboratore.

Abstract : The "enneadecaphonic" system (i.e. the set of frequencies proportional to powers of $\sqrt[19]{2}$) is discussed. Beethoven's themes give rise to samples of music for this kind of scale, while new compositions could not be played with instruments tuned in the traditional way. A basic mathematical structure for programming automatic enneadecaphonic composition is described.

La musica enneadecafonica è stata introdotta nell'articolo citato nella bibliografia. In questo lavoro vengono analizzate le proprietà fondamentali utili per la composizione (automatica o umana) in questo nuovo sistema di armonie.

I principali inconvenienti della << scala "ben" temperata secondo Bach >> (brevemente indicata con << scala dodecafonica >>) segnalati dai teorici della musica sono dovuti alle 3 seguenti ragioni:

(1) La suddivisione dodecafonica dell'ottava mediante 12 frequenze in progressione geometrica con ragione $\sqrt[12]{2}$, mentre approssima soddisfacentemente gli intervalli di 5^a, non offre che una mediocre approssimazione per le 3^e minori e maggiori (con errore 8/1000 circa) e di conseguenza le rende ingiustamente meno consonanti dell'intervallo di "tono".

(2) Ancor più grave è il compromesso che identifica con la 3^a minore (che dovrebbe essere consonante) l'intervallo di 2^a eccedente (che dovrebbe essere molto dissonante).

Questo compromesso è stato definito "imbarazzante" dai teorici dell'armonia.

(3) Un analogo compromesso identifica la "quarta eccedente" con la "quinta diminuita" (che pur risolvono diversamente). Questo intervallo risulta... il più dissonante, mentre poi compare nel semplicissimo accordo di 7^a (p. es. si - fa - sol)!

Prima di studiare la scala enneadecafonica, che elimina questi inconvenienti, è opportuno ricordare che in molte ricerche musicali è stato segnalato nella storia della musica una progressiva perdita di sensibilità verso le armonie naturali alterate dalla mediore approssimazione e un progressivo avvicinamento verso successioni puramente casuali di note.

Questa tendenza è l'ovvio epilogo dei compromessi della scala tradizionale dodecafonica, ma è anche un vicolo cieco perchè oltre il puro caso non si può andare. L'unica alternativa è quindi l'introduzione di nuove scale.

L'idea di partenza della scala enneadecafonica è quella di non rinunciare a una successione di frequenze in progressione geometrica (per rendere possibili le "modulazioni" senza dover introdurre una infinità di nuove note) scegliendo però come rapporto tra due note consecutive il numero $\sqrt[19]{2}$ le cui potenze permettono una migliore approssimazione delle principali frazioni corrispondenti ad intervalli armonici senza richiedere assurdi compromessi.

Queste potenze hanno approssimativamente i seguenti valori:

do	= 1 =	(unisono)	
do #	(28/27)	intervallo cromatico	dissonante atonale
re b	14/13	2ª diminuita	
re	10/9	2ª	
re #	(15/13)	2ª eccedente	dissonante atonale
mi b	6/5	3ª minore	
mi	5/4	3ª maggiore	
mi # = fa b	9/7	4ª diminuita	atonale
fa	= 4/3 =	4ª	
fa #	18/13	4ª eccedente	
sol b	13/9	5ª diminuita	
sol	= 3/2 =	5ª	
sol #	14/9	5ª eccedente	atonale
la b	8/5	6ª minore	
la	5/3	6ª maggiore	

la♯	(26/15)	6ª eccedente	dissonante atonale
si♭	9/5	7ª minore	
si	13/7	7ª maggiore	
si♯ = do♭	(27/14)	7ª eccedente	dissonante atonale

La "scala enneadecafonica" è composta da 12 tra queste 19 note:
do re♭ re mi♭ mi fa fa♯ sol la♭ la si♭ si.
Ciononostante come si vede facilmente tutti i 19 tipi di intervalli elencati hanno qualche rappresentante.

Rimanendo 12 il numero delle note per ogni ottava, per avere uno strumento enneadecafonico è sufficiente accordare un pianoforte secondo queste nuove frequenze. Ciò è reso particolarmente facile dal fatto che le 3ᵉ minori enneadecafoniche (fa♯- la - do - mi♭ , si - re - fa - la♭ , mi - sol - si♭ - re♭) sono praticamente perfette. Si noti che le 2ᵉ eccedenti sono invece: mi♭ - fa♯ - la♭ - si - re♭ - mi e risultano, come gli intervalli cromatici, i più dissonanti; questa forte dissonanza rende ad esempio necessaria una correzione (segnalata nel lavoro citato nella bibliografia) nella 25ª battuta della sonata al chiaro di luna di Beethoven (sembra dunque evidente che Beethoven non aveva in mente la scala enneadecafonica; d'altra parte nella 27ª battuta della stessa sonata l'esecuzione enneadecafonica mette in miglior risalto le abbondanti 3ᵉ minori).

E' anche importante tener presente la differenza tra il "semitono cromatico" e la assai meno dissonante "2ª diminuita" (che con notazione tradizionale si può chiamare "semitono diatonico" e nella scala dodecafonica era identificata con il semitono cromatico.

Tra le più semplici proprietà fondamentali che differenziano la scala enneadecafonica da quella dodecafonica notiamo la comparsa di intervallo "atonali" cioè intervalli tra due note che non figurano simultaneamente in alcuna tonalità. Essi sono: Intervallo cromatico, 2^a eccedente, 4^a diminuita, 5^a eccedente, 6^a eccedente, 7^a eccedente. Particolarmente notevoli, tra questi, sono gli intervalli di 4^a diminuita e di 5^a eccedente: pur essendo atonali essi non sono molto dissonanti, e il loro uso offre notevoli possibilità armoniche e melodiche del tutto nuove, come è esemplificato nelle composizioni riportate nel lavoro citato nella bibliografia, che perdono il loro senso musicale se suonate su strumenti accordati in modo dodecafonico.

Per quanto riguarda la composizione automatica di musica enneadecafonica alfine di non ostacolare la vastità degli stili che essa permette è opportuno limitarsi a indicare i principi fondamentali che occorre rispettare, lasciando al lettore i dettagli dell'algoritmo.

Gli accordi enneadecafonici a 3 o 4 note devono soddisfare ai seguenti requisiti:

(1) L'accordo deve poter risolvere (secondo le "regole di risoluzione" di cui nel seguito verrà indicata una possibile formulazione) in una "tonalità ammessa". Sono "ammesse" le tonalità di cui esistono effettivamente le 7 note della scala (maggiore o minore naturale), e cioè:

tonalità maggiori: sol, do, fa, si♭, mi♭, la♭

tonalità minori: mi, la, re, sol, do, fa

(2) L'accordo non può contenere note troppo vicine tra loro sulla tastiera come:

(a) 2 note a distanza cromatica o di 2^a diminuita

(b) 3 note entro un intervallo di 4^a diminuita

(c) 4 note entro una 5^a

(3) Non può essere accettato un accordo che non è nè maggiore nè minore e differisce per una sola alterazione cromatica da un accordo che sarebbe maggiore o minore ma che in realtà non si può suonare con le 12 note scelte nella scala enneadecafonica. E' ad es. proibito l'accordo la - re♭ - mi che non è nè maggiore nè minore e differisce per l'intervallo cromatico do♯ -re♭ dall'accordo la - do♯ - mi che sarebbe l'accordo di "la" maggiore ma che non si può suonare con le 12 note della scala enneadecafonica.

Gli accordi ammissibili si classificano in:

(1) Accordi maggiori o minori (12 in tutto).

(2) Altri accordi tonali (ad es. fa-sol-si, la♭- do - re).

(3) Accordi atonali non dissonanti (ad es. si - mi♭ - fa che contiene l'intervallo atonale si - mi♭).

(4) Accordi dissonanti (ad es. re♭ - fa - si che contiene la dissonanza re♭ - si).

Per misurare il ruolo delle varie note sia nella armonia sia nella melodia è importante introdurre le seguenti metriche:

Metrica armonica: Il legame tra due note (di nome diverso) viene valutato secondo la seguente classifica:

(1) Legame armonico di 4^a o di 5^a

(2) Legame armonico di 3^a maggiore o minore o rivolto.

(3) Altro legame consonante.

(4) Legame dissonante.

Metrica melodica: Il legame tra una nota e un accordo di 3 note viene valutato secondo la seguente classifica:

(1) La nota appartiene all'accordo

(2) La nota è a distanza di semitono cromatico o di 2^a diminuita da una nota dell'accordo

(3) La nota non integra l'accordo in un accordo ammissibile a 4 note

(4) La nota integra l'accordo in un accordo ammissibile a 4 note.

Transizione tra 2 accordi: Il criterio che stabilisce se una transizione tra due accordi è o non è lecita deve essere del seguente tipo: Le note N del nuovo accordo, salvo al massimo p di esse (per es. p = 1) devono essere legate a note M dell'accordo precedente con un legame

$$f(N,M) + g(N,M) \geq \text{cost}$$

dove f,g sono funzioni monotone rispettivamente nella metrica armonica e in quella melodica.

Analogamente i legami tra le note della melodia e quelle degli accordi possono essere stabiliti in una grande varietà di modi, sempre in funzione delle due metriche descritte.

Alle varie formule che esprimono questi legami vengono attribuiti dei "pesi" inizialmente casuali, e l'elaboratore compone scegliendo accordi e note che massimizzano la somma dei pesi delle regole soddisfatte.

Si ha così un procedimento di composizione casuale nel quale le estrazioni di numeri a caso non determinano direttamente note o accordi ma fissano i "pesi" da dare alle varie regole.

Segue una facile selezione che porta ad individuare il livello ottimo dei vari "pesi" per ottenere il risultato

che si desidera.

Alcune musiche enneadecafoniche composte non automaticamente ma con l'aiuto delle regole ora esposte sono pubblicate nel lavoro:

BIBLIOGRAFIA

E. GAGLIARDO: Enneadecaphonic music. A new system of harmonic tones. Atti Accademia Ligure, 1980.

IL CALCOLO COMBINATORIO

COME STRUMENTO FORMALE E DI SVILUPPO NELLA COMPOSIZIONE

P. GROSSI - CNUCE, PISA
L. MICHI - FIRENZE

Riassunto

Sono stati progettati e realizzati alcuni programmi di elaborazione automatica di strutture sonore costituite da masse di densità varia e di vario sviluppo formale.

L'operatore può controllare a sua discrezione i processi di sviluppo con procedure interattive in massima parte costituite da gruppi di carattere speciali della console.

Il calcolo combinatorio, uno degli strumenti di rielaborazione e di sviluppo formale del TAUMUS, è utilizzabile tramite una subroutine del TAUMUS che possiede le seguenti caratteristiche operative:

- lo sviluppo può essere controllato dalle diverse formule quali le combinazioni, le permutazioni, le disposizioni sia semplici che con ripetizione.
- Gli elementi dei sottoinsiemi possono essere al limite 9. Tali possono definirsi i parametri singoli acustici, i suoni singoli oppure sezioni di brani.
- L'utente ha la facoltà di definire il numero dei sottoinsiemi da elaborare; di definre le voci musicali da elaborare, di chiedere l'ordinamento casuale dei sottoinsiemi.

Abstract

Some automatic elaboration programs of sound structures constituted by masses of various density and formal development have been designed and realized.
The operator can control, to his discretion, the development processes using interactive procedures formed for the most part by groups of special characters assigned at the console.
Combinatorial computation, one of the rielaboration and formal development tools of TAUMUS, is utilizable through a TAUMUS subroutine which has the following operative characteristics:

- The development can be controlled by different formulas as combinations, permutations, dispositions both simple and with repetition.
- The subsets elements can be at most 9. In this way we can define single acoustic parameters, single sounds or passages sections.
- The user can define the number of subsets and musical voices to elaborate and have the subsets randomly ordered.

P.GROSSI - CNUCE, PISA
L.MICHI - FIRENZE

IL CALCOLO COMBINATORIO COME STRUMENTO FORMALE E DI SVILUPPO NELLA COMPOSIZIONE

-Una subroutine del TAUMUS-

Funzione COMBINA (CM)

Rielabora un brano in memoria o parte di esso mediante formule del calcolo combinatorio. Dato un insieme finito di elementi si possono operare vari raggruppamenti tenendo conto delle tre eventualità di prenderli tutti senza ordine, permutazioni, di prenderli in parte con ordine, disposizioni, in parte senza ordine, combinazioni, ripetendo o no lo stesso elemento più volte e quindi tali raggruppamenti si chiamano rispettivamente semplici o con ripetizione.

La funzione COMBINA esamina tutte queste possibilità considerando un insieme I di elementi che possono essere di due tipi:

- suoni singoli o singoli parametri acustici
- sezioni del brano in esame

Il numero di elementi nel primo caso puo constare al limite di 2000 suoni, nel secondo caso e limitato a 9.
In corrispondenza si hanno due tipi di formato dell'istruzione:

FORMATO 1 - elaborazioni di parametri acustici e di suoni singoli:

COMBINA PA± CV± p1 , p2 , p3

Simboli variabili - sono racchiusi fra parentesi:

PA - sta per parametri acustici; indica una sequenza di uno o più fra i quattro simboli:

F frequenza

D durata

T timbro

V volume

CV - sta per CANALI E VOCI ed indica una sequenza di lettere (A, B, C) e numeri (1, 2, 3, 4) che si riferiscono rispettivamente ai canali e alle voci interessate dal comando.

N.B. La specifica di canale ha carattere limitativo; non scrivere alcun canale significa prenderli tutti, ma scriverne alcuni significa considerare solo quelli ed escludere i rimanenti. Le stesse considerazioni valgono, appena assegnato un canale, per le voci i cui numeri dovranno essere scritti immediatamente dopo la lettera del

canale al quale si riferiscono.

p1 - ha la funzione di deteminare l'ordine dei sottoinsiemi generati dal calcolo combinatorio. Se omesso, l'ordine dei sottoinsiemi sarà quello risultante dall'applicazione della formula del calcolo; cioè si avrà come sequenza dei raggruppamenti quella che si ottiene dal ragionamento per la generazione di tutte le eventualità. Se impiegato con un qualsiasi valore, si ottiene l'ordine dei sottoinsiemi casuale.

p2 - indica l'elemento dell'insieme a partire dal quale deve iniziare l'elaborazione. Con questo parametro e il successivo si delimita un sottoinsieme di tutti gli elementi di I che saranno esaminati dalla funzione.

p3 - indica l'altro estremo dell'intervallo prescelto all'interno dell'insieme I.

Se p2 e/o p3 sono omessi acquistano per default i valori 1 e N rispettivamente dove N è l'ultima nota del brano o l'ultimo centesimo di secondo se si utilizza la costante Z per il sezionamento temporale in centesimi di secondo del pezzo.
Se p2, p3, non vengono impiegati si considerera il numero di suoni specificato nel FORMATO 3,(N), e si elaborano solo i primi N suoni del testo.

I parametri sono separati dal carattere ","; se uno di essi è omesso si avrà la sequenza di due virgole consecutive se tutti e tre i parametri non sono specificati non si scriverà nessun carattere separatore.

Se invece di singoli suoni, si considerano parti del brano, si ha il FORMATO 2:

COMBINA Z CV±/ p1 , (p2 , p3),....,(p2 , p3)

Parametri obbligatori:

Z - è di obbligo in questo formato in quanto si considera la suddivisione dei brani in centesimi di secondo e gli intervalli così determinati sono gli elementi sui quali il programma opererà per realizzare i raggruppamenti.

Almeno una coppia:

(p2, p3) - è l'iesima coppia in esame: il primo parametro indica il centesimo di secondo a partire dal quale i suoni successivi appartengono all'elemento dato; il secondo parametro e la durata massima che può assumere un suono nell'intervallo considerato.

Parametri opzionali:

p1 - come nel formato 1 indica il modo con il quale deve essere selezionato il raggruppamento; si ricorda che se omesso l'ordine dei sottoinsiemi generati sarà quello dovuto al calcolo combinatorio.

OSS - i numeri sottoscritti 2 e 3 servono per ricollegarsi

al formato 1 indicando ciascuno il primo e l'ultimo suono, in questo caso il primo e l'ultimo centesimo di secondo della porzione di brano che vogliamo elaborare.

- le elaborazioni che prevedono più di 2000 suoni per voce vengono interrotte al raggiungimento di tale quota limite.

Se il formato 1 e 2 hanno la funzione sopra tutto di scelta degli elementi, il formato 3 che esamineremo fra poco ha lo scopo di raggruppare opportunamente i suoni secondo le scelte combinatrici. Infatti dati uno dei primi comandi, il programma invierà il messaggio ASSIGN e si porrà in attesa di informazioni riguardanti la formula, il numero degli elementi del sottoinsieme ed altri parametri di carattere limitativo. Si ha quindi il FORMATO 3 (da impiegare dopo il messaggio ASSIGN)

```
C
PR  N±,  K±/p1 , p2 , p3
D
```

Parametri obbligatori sono:

C,P,D,R, - che indicano il tipo di calcolo combinatorio scelto dall'operatore per l'elaborazione dei sottoinsiemi. Si possono impiegare così :
1) C - combinazioni semplici
2) P - permutazioni
3) D - disposizioni semplici
4) CR - combinazioni con ripetizione
5) PR - permutazioni con ripetizione
6) DR - disposizioni con ripetizione

N - è il numero degli elementi dell'insieme I sul quale si applica la funzione COMBINA

K - è il numero degli elementi di ogni insieme

OSS - Se K è uguale a N può essere omesso e in tal caso le permutazioni coincidono con le disposizioni.
K non deve essere maggiore di N.
Ambedue devono essere numeri positivi, interi, diversi da zero.

Parametri opzionali:

p1 - se impiegato, questo parametro indica il primo sottoinsieme a partire dal quale si verifica la scelta sui raggruppamenti effettuati.

p2 - indica l'ultimo dei sottoinsiemi generati al quale si deve arrestare la scelta.

p3 - è il passo con il quale si scandisce l'insieme dei raggruppamenti.

OSS - Se omessi p1 e p3 assumono come valore di default 1: cioè si considerano i sottoinsiemi a partire dal primo con passo 1: non saltando nessun raggruppamento.
- se p2 non è specificato questo parametro assume per default il numero dei sottoinsiemi previsti dalla formula e pertanto i raggruppamenti verranno

considerati fino all'ultimo;
- anche qui i parametri sono separati da virgole e valgono le stesse considerazioni dette precedentemente per il formato 1 nel caso che uno o più di questi parametri venga omesso;

OSS
- se il numero dei suoni da rielaborare è inferiore a N, l'elaborazione non ha luogo;
- se il numero dei suoni da rielaborare, definito dai parametri, è multiplo di N si avrà un numero di elaborazioni pari al rapporto fra il numero dei suoni e N. Infatti inizialmente verranno elaborati i primi N suoni poi ne rimarranno un numero maggiore o uguale a N e si può effettuare almeno un altro raggruppamento su N elementi.

Particolarità operative riguardanti i formati 1 e 2:

1) Se l'elaborazione non comprende l'intero testo in memoria, col FORMATO 1 il testo non elaborato viene conservato e unito all'elaborazione stessa; col FORMATO 2 viene invece eliminato.

2) Se l'elaborazione: FORMATO 1 concerne parte dei parametri acustici gli altri vengono impiegati per la formazione dei suoni nell'ordine in cui sono posti nell'insieme dato:
Esempio:

Dati 3 suoni e richieste le permutazioni delle frequenze e dei timbri, l'elaborato sarà così costituito (in numero d'ordine dei parametri):

frequenza	1	2	3	1	3	2	2	1	3	2	3	1	3	1	2	3	2	1
durata	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
timbri	1	2	3	1	3	2	2	1	3	2	3	1	3	1	2	3	2	1
volumi	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3

Il numero di elementi ammessi varia se come tali sono considerati i singoli parametri acustici e i singoli suoni oppure sezioni del brano le cui durate debbono essere definite in centesimi di secondo.

Nel primo caso non si avranno limiti, nel secondo il numero massimo è di 9 elementi. Si tenga presente che la capacità dell'area per voce è di 2000 suoni e che il programma a tale limite arresta l'elaborazione.
Pertanto, operando col Formato 1, si possono in effetti ottenere tutte le permutazioni di 5 elementi che constano di 720 suoni e non tutte quelle 6 suoni che constano di 4320 suoni. Si otterrà, altresì, l'elaborazione di un solo sottoinsieme se N è superiore a 1000. L'ostacolo, però, si supera con la seguente procedura permessa dal TAUMUS.

- Dati sei suoni si vogliono ottenere le permutazioni (720 pemutazioni pari a 4320 suoni).

1) - Memorizzazione in archivio dei 6 suoni:
comando SAVE SUONI

2) - Sviluppo delle prime 300 permutazioni e loro memorizzazione in archivio
- Comandi :

```
COMBINA
P6/1, 300 (elaborazione)
SAVE PERM6 (memorizzazione in archivio)
```

3) - Sviluppo di altre 300 permutazioni e loro memorizzazione in archivio
comandi:

```
LOAD SUONI (richiamo dall'archivio dell'insieme di 6 suoni)
COMBINA
P6/301, 600
SAVE PERM6A
```

4) - Sviluppo delle rimanenti 120 permutazioni.
Comandi:

```
LOAD SUONI
COMBINA
P6/601
SAVE PERM6B
```

L'esecuzione di tutte le permutazioni si ottiene facendo eseguire in sequenza i brani PERM6, PERM6A e PERM6B tramite il comando EXEC.

SOFTWARE PER COMPOSISIONI SERIALI

Vito Ozzola (Milano)

Lavoro svolto nell'ambito del Gruppo Nazionale per l'Informatica Matematica del C.N.R.

Sunto.

Si presentano tre programmi per composizioni di indole seriale creati nel rispetto dei seguenti principi generali.

Il computer viene usato come strumento atto solo ad alleviare la fatica di un grande numero di operazioni combinatorie; il programma viene ideato come finalizzato (con superiori criteri musicologici) ad ottenere effetti prestabiliti; i dati output possono essere usati sia come composizioni musicali compiute sia come semplice materiale di partenza per una ulteriore elaborazione che, risentendo del gusto e della creatività del compositore, rispetti le strutture di fondo calcolate dal computer.

I programmi di calcolo, ampiamente flessibili, corrispondono a tre diversi criteri di composizione e gli output possono anche essere usati con mutua sovrapposizione parziale o totale e anche sequenzialmente.

Abstract.

Three programs for compositions of serial character created respecting the following general principles are shown.

The computer is used as a tool capable only of relieving the fatigue of a great number of combinatorial operations; the program is conceived in order to obtain (with higher musical criteria) pre-arranged effects; output data may be used either as finished musical compositions or as simple equipment from which to start a further elaboration that, reflecting the taste and the creativity of composer, respects the basic structures processed by computer.

The computer programs, very flexibile, correspond to three different composition criteria and the output may also used with mutual partial or total superimposition and also sequentially.

1. E' ben noto che la tecnica della composizione seriale è quella che offre il rapporto probabilmente ottimale fra l'impiego di tecniche combinatorie rigorose e la resa finale in termini di udibilità, intendendo per udibilità il processo neurale del riconoscimento esplicito delle forme sonore. D'altra parte la pratica della composizione insegna che l'eccessiva conformità del processo compositivo agli schemi del serialismo della prima ora conduce abbastanza rapidamente all'isterilimento della vena inventiva. In questa nota si descrivono tre programmi in linguaggio Fortran che corrispondono ad un impiego molto libero della tecnica seriale e che cercano di realizzare un compromesso ragionevole fra il rigore combinatorio e la libertà creativa.

2. Si descrivono anzitutto gli intendimenti generali sottostanti al lavoro.

a. Il computer viene utilizzato per la preparazione di un canovaccio per il quale si può garantire a priori la buona formazione grammaticale rispetto a regole prefissate di correttezza e completezza formale. E' previsto che l'output possa essere localmente sviluppato e tradotto con un assegnato grado di libertà in una partitura definitiva. Il compositore viene in tal modo alleviato dalla fatica puramente meccanica del continuo feedback combinatorio e risulta liberato dal pericolo di non saper conciliare la coerenza formale con la spontaneità espressiva. Non è tuttavia esclusa la possibilità di una traduzione deterministica dell'output in messaggio direttamente sonoro.

b. Il programma di calcolo viene determinato in modo che si possa prevedere l'effetto complessivo della composizione in termini di audiopsicologia sperimentalmente accertati. Per esempio si potrà operare in modo da ottenere con certezza l'effetto del progressivo passaggio dal disordine all'ordine sonoro, oppure l'effetto del riempimento progressivo di tutto lo spazio compositivo con pochi elementi formali predeterminati, oppure l'effetto del "silenzio" compositivo dovuto al diradarsi degli avvenimenti sonori, e così via.

c. In conseguenza di quanto detto in (a) si ha per il compositore la possibilità di sovrapporre altri elementi formali a quelli risultanti in output.

In tutti e tre i casi studiati in questa nota si ha la possibilità di sovrapporre e anche di alternare ad ognuno dei tre output elementi formali suggeriti da un altro fra i due output rimanenti, o anche dagli altri due output presi insieme. Si ha così una struttura addittiva e moltiplicativa che, da esperimenti effettuati dall'autore di questa nota, sembra promettente in termini strettamente musicali.

I programmi qui di seguito descritti possono essere variati anche in misura notevole senza uscire dallo schema risultante da quanto detto in (a),(b),(c). La tecnica compositiva emergente dal presente schema non coincide con quella seriale tradizionale, ma le si avvicina anche e soprattutto in termini di audiopsicologia, nel senso risultante da (b).

3. Si descrive qui di seguito il primo programma di cui nel paragrafo (2). Tale programma è designato MALI1.
Il germe di MALI1 è una melodia M formata da un numero prefissato di replicazioni di serie scelte fra una serie fondamentale e le sue trasformate nel senso tradizionale (inversa, retrograda, inversa della retrograda). I parametri di M sono scelti dal compositore, compresa la eventuale omissione di qualche serie, esclusa la serie fondamentale, che deve in ogni caso essere presente. Lo sviluppo della composizione è regolato dai seguenti criteri. Dapprima tutte le voci eseguono tutte le serie ammesse o parti di esse alternando note di durate prefissate (vedi dopo) con pause, in modo che il rapporto fra la durata del suono e la durata delle pause sia un valore q diverso per ciascuna voce. E' ovvio che in tale fase ha la massima intensità la sensazione del "disordine" audiologico. Gli schemi per la successione delle durate dei suoni e delle pause sono ottenuti da schemi prefissati; i parametri che corrispondono alle varie scelte sono a disposizione del compositore. Sono inoltre previsti due vincoli strutturali locali. Anzitutto il vincolo relativo all'unisono: sono escluse automaticamente le scelte che comportano l'esecuzione contemporanea dello stesso grado della serie fondamentale e delle sue trasformate. In secondo luogo si ha un vincolo di battuta. E' fissato, cioè, un tempo elementare T (durata) e nel giro di un numero prefissa-

to di durate elementari (battuta) deve venire eseguito un numero intero di replicazioni delle serie previste o di parti di esse. La scelta fra il numero di replicazioni ammesse è scelto inizialmente dal compositore. Viene poi composto un modulo temporale (battuta) identico al primo ma con un diverso rapporto q_1 fra la durata complessiva dei suoni e quella delle pause. Naturalmente come per q_0 esiste un q_1 diverso per ogni voce. E' significativa per MALI1 la scelta $q_0 > q_1$.

La composizione prosegue, secondo successioni monotone decrescenti q_0, q_1, q_2, q_3,..... diverse per ogni voce. Ciò comporta una rarefazione dei suoni per ciascuna battuta, ed, eventualmente, un prolungamento di ciascun suono a scelta del compositore.

Anche tale prolungamento di ciascun suono può essere programmato automaticamente oppure lasciato a discrezione del compositore, che può servirsene con criteri "magici".

La composizione termina automaticamente quando tutte le voce sono ridotte, dal determinismo del programma, al silenzio.

E' ovviamente prevedibile, e le esperienze eseguite confermano, che si ha un passaggio, perfettamente avvertibile, dalla sensazione di disordine a quella di ordine audiologico.

MALI1 è particolarmente indicato, come risulta dalla sua stessa struttura, per composizioni riccamente polifoniche, tipicamente per l'orchestra. Tuttavia, con limitazioni che non ne alterano sostanzialmente la struttura, si presta anche per l'esecuzione pianistica o per pochi strumenti nello stile da camera.

4. Si descrive di seguito il programma MALI2.

Il materiale sonoro di base è ora di tipo ritmico.

Si ha ancora una serie fondamentale , con le sue derivate (come sopra). Il compositore fissa una melodia M formata da quattro replicazioni delle quattro serie ammesse. Fra queste replicazioni deve figurare ed essere privilegiata la serie fondamentale, possono mancare alcune serie derivate (o tutte). Le figure ritmiche ammesse sono (solo) quartine, cinquine e sestine. Ciò comporta una notevole ristrettezza nella scelta compositiva locale per

quanto riguarda il vincolo dell'unisono. Si può ovviare a questo inconveniente stabilendo certi canoni permissivi. Tuttavia questi sono accuratamente calibrati in base ai dati della osservazione audiopsicologica e alla professionalità compositiva.
La successione delle figure ritmiche consentite in ogni voce rispetta automaticamente il vincolo della integrità della esecuzione di ognuna delle serie impiegate, e per il resto è lasciata all'arbitrio del compositore. Questi può ad esempio decidere di sovrapporre progressivamente le voci e la loro densità creando così l'effetto del crescendo strumentale. Oppure può distribuire in modo stocasticamente uniforme le figure ritmiche fra le varie parti, creando la sensazione della ripetitività ossessiva, e così via.
A differenza di quanto avveniva in MALI1 la durata di una composizione MALI2 è fissata dal compositore, che tuttavia può e deve rispettare il vincolo della integrità delle serie.
Per evitare l'unisono in misura limitata, il programma può sostituire una nota con una pausa, riproponendo la nota omessa subito dopo, al fine di non manomettere l'integrità delle serie. In questo senso MALI2 è più rigorosamente seriale (anche nel senso della audiologia sperimentale) di MALI1.

5. Il programma MALI3 consente la composizione di musica polifonica fino ad un massimo di dodici parti, esente da unisoni e di struttura rigorosamente seriale.
Il programma riposa sulla teoria dei gruppi di sostituzioni. Consideriamo una serie a_1, a_2, a_3, ... a_{12} dei dodici suoni temperati come una sostituzione:

$$S = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} & a_{11} & a_{12} \end{pmatrix}.$$

Le successive potenze di S genereranno altrettante serie.
Le 12! serie possibili formano il gruppo totale G su dodici elementi. Il gruppo ciclico generato da una sostituzione parziale S (ossia il gruppo delle potenze tutte distinte di S) è un sottogruppo di G. Anche il gruppo libero generato da due o più sostituzioni è un sottogruppo, eventualmente improprio,

di G. La teoria dei gruppi di sostituzioni assegna condizioni molto ampie sotto le quali un elemento scelto fra i dodici gradi temperati non è unito per alcuna delle sostituzioni di un sottogruppo di G diversa dall'identità. Sottoponendo la scala cromatica a tutte le sostituzioni di un sottogruppo di G prive di elementi uniti si ottengono serie che possono essere sovrapposte liberamente (fino al massimo di dodici parti) senza pericolo di unisoni.

Entro questa struttura, molto facilmente calcolabile con MALI3, si possono liberamente scegliere e contrapporre le serie con i più svariati criteri espressivi. Si può anche decidere di costruire un sottogruppo di G e di proseguire la composizione fino alla comparsa di un unisono e di evitare questo arrestando "catastroficamente " la composizione, sostituendola con un'altra.

Con artifici compositivi che in parte derivano dalle proprietà elementari della teoria dei gruppi di sostituzioni e in parte derivano dalla professionalità del compositore è relativamente facile predisporre composizioni nelle quali la successione delle "catastrofi" è l'insieme vuoto oppure sodisfa a una precisa "regia". Come è stato detto nel paragrafo (2), è possibile comporre insieme MALI3 con gli altri due programmi precedenti.

6. In quanto precede si parla di maggiore o minore rigore seriale. E' opportuno precisare che con ciò non si intende dire che MALI1, MALI2 e MALI3 violano la prescrizione fondamentale della sintassi seriale secondo la quale tutti i gradi temperati devono essere esposti con pari evidenza e per passare da un grado all'altro si deve percorrere tutta la serie. Si intende solo dire che i programmi descritti mettono bene in evidenza la struttura seriale come un udibile. Il maggiore o minore grado di udibilità a cui si accenna è giudicato in base all'esperienza eseguita e non in base alle prescrizioni sintattiche, che sono tutte rigorosamente rispettate.

° ° ° °
° ° °

I programmi sono stati realizzati presso l'Istituto Matematico dell'Università degli Studi di Milano utilizzando l'elaboratore Digital PDP 11.

Si ringraziano i proff. Massimiliano Lunelli, Marzia Bonecchi e Antonella Tonolli per l'utile collaborazione.

Vengono, nel seguito, riportati:

- parte di un output elaborato dal programma MALI1,
- la partitura della versione per due pianoforti di tale output realizzata da Elisa Marini allieva del corso di "Nuova didattica della composizione" del prof. Azio Corghi del Conservatorio G. Verdi di Milano.

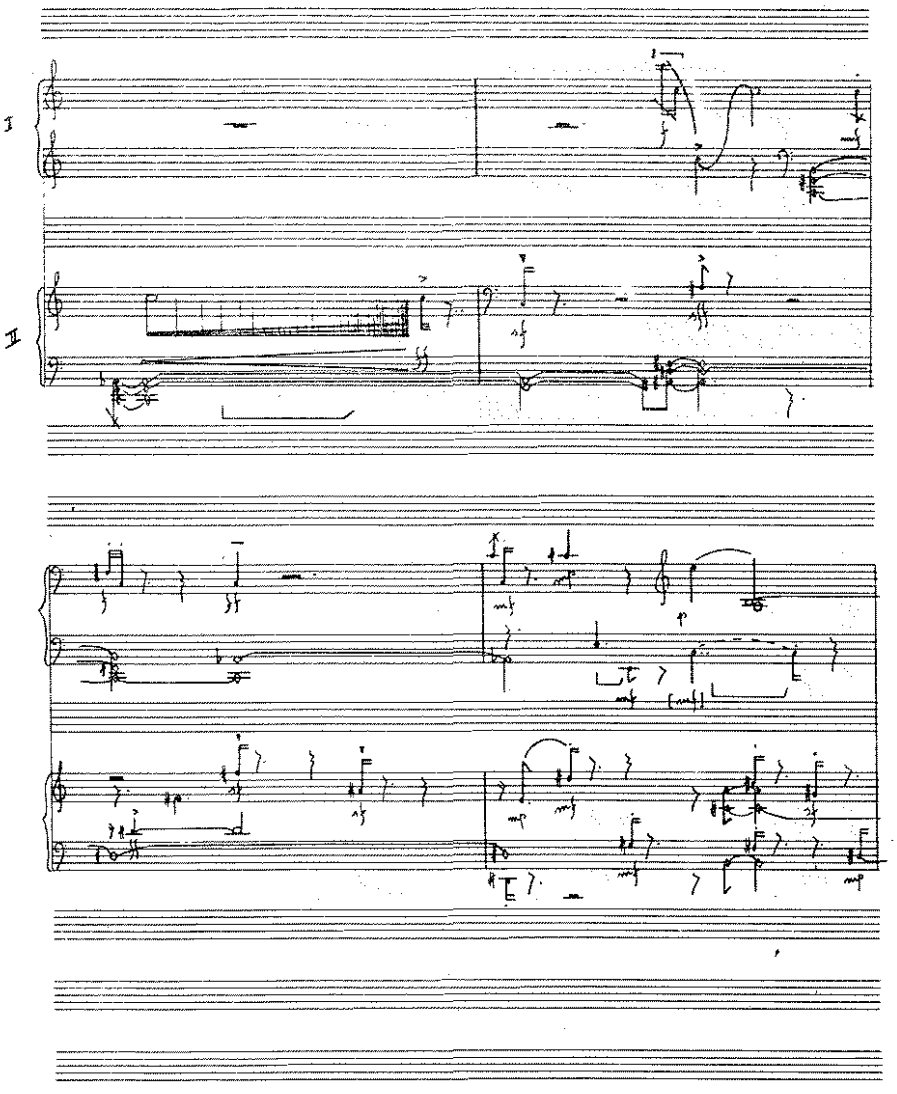
```
RUN DX1:MALI
   QUARTO CONVEGNO DI INFORMATICA MUSICALE
0 610 3 4 1 8 911 5 7 212 6 1 696
1 2 3 4

PROGMALI              QUARTO CONVEGNO DI INFORMATICA MUSICALE
   12    6    1    6   96    4
    1    2    3    4
    0    6   10    3    4    1    8    9   11    5    7    2
    0    6    2    9    8   11    4    3    1    7    5   10
    2    7    5   11    9    8    1    4    3   10    6    0
   10    5    7    1    3    4   11    8    9    2    6    0
```

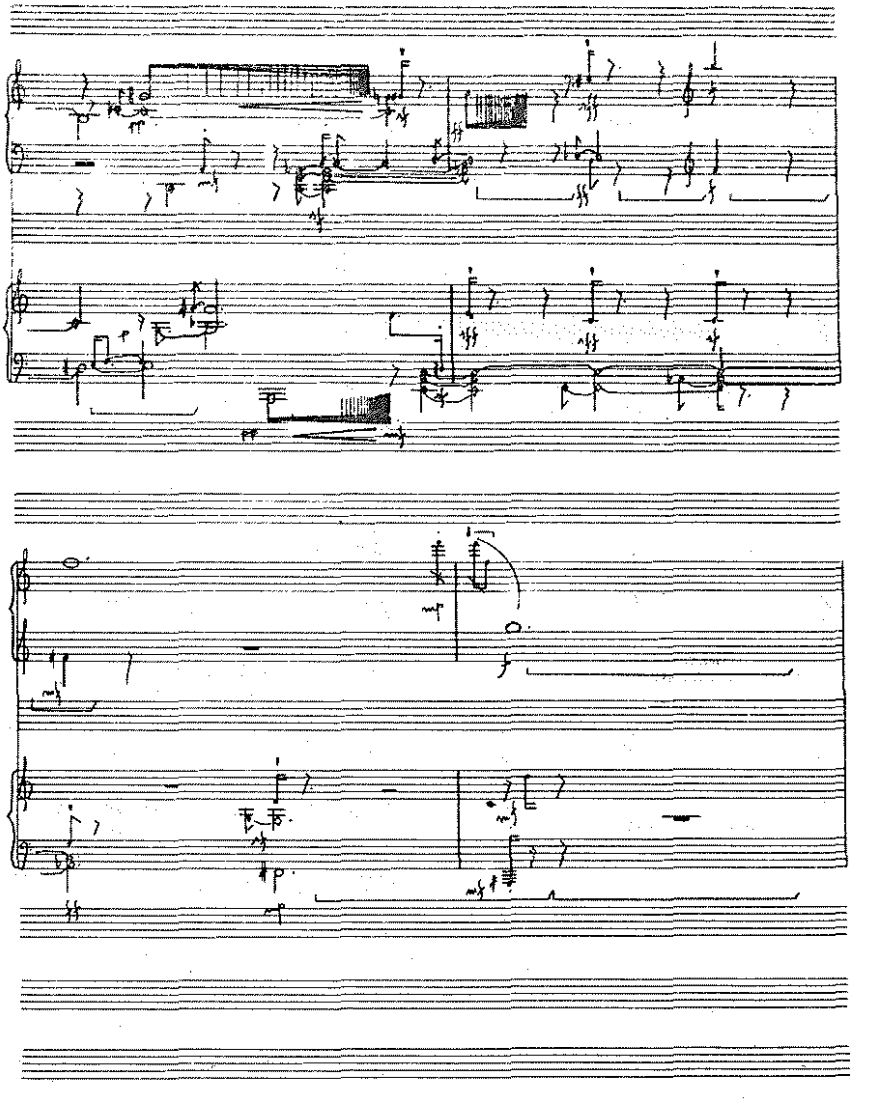
1	-	-	DO	-	F♯	L♯	+	R♯	+	MI	D♯	S♯	+	+	+	+	+	+	LA	+	SI	FA	+	SO
2	DO	-	L♯	R♯	+	+	MI	D♯	S♯	+	SI	+	+	+	FA	+	+	SO	RE	DO	-	F♯	RE	-
3	-	-	-	-	DO	+	+	+	+	-	-	-	-	F♯	L♯	+	+	R♯	MI	S♯	FA	SO	+	RE
4	-	DO	-	F♯	-	-	R♯	-	MI	-	LA	+	FA	+	SO	+	+	-	-	RE	DO	+	F♯	-
5	-	-	-	DO	-	-	-	-	-	-	F♯	+	MI	+	+	LA	+	RE	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-	-	DO	+	R♯	+	+	LA	-	DO	+	+	-	-	-	-	F♯

1	RE	+	DO	+	F♯	+	-	RE	LA	S♯	SI	+	+	MI	R♯	D♯	SO	FA	L♯	RE	+	+	+	+
2	S♯	+	SI	+	+	MI	R♯	+	D♯	-	SO	+	FA	L♯	RE	SO	-	-	-	SI	+	S♯	+	D♯
3	-	-	-	-	-	DO	F♯	+	RE	+	+	LA	+	S♯	+	+	SI	+	R♯	+	D♯	+	FA	+
4	LA	+	+	S♯	D♯	L♯	+	+	+	+	-	-	-	-	-	-	-	RE	+	SO	FA	+	LA	+
5	F♯	+	+	RE	LA	S♯	+	SI	R♯	+	D♯	+	+	+	+	-	-	SO	SI	-	-	-	-	-
6	-	-	-	-	-	-	LA	+	-	D♯	L♯	+	RE	+	-	-	-	-	-	-	-	-	SO	+

1	+	+	SO	+	FA	SI	LA	S♯	+	D♯	MI	R♯	L♯	F♯	-	-	-	-	-	DO	L♯	FA	SO	+
2	MI	+	+	+	L♯	F♯	DO	L♯	FA	+	SO	D♯	+	+	+	+	+	R♯	+	+	+	+	MI	+
3	L♯	+	+	+	RE	SO	+	LA	+	S♯	+	+	MI	L♯	F♯	DO	L♯	FA	-	-	-	-	-	-
4	+	+	+	+	S♯	-	-	-	-	-	D♯	MI	-	-	-	-	-	L♯	+	F♯	DO	L♯	+	+
5	-	-	D♯	+	MI	+	+	+	+	+	-	-	-	-	-	F♯	FA	-	SO	-	-	-	-	-
6	+	+	-	-	-	S♯	+	R♯	+	+	+	-	-	-	-	-	-	-	FA	SO	+	S♯	+	+

1	-	D♯	+	-	-	R♯	+	MI	SI	S♯	+	+	LA	+	RE	F♯	+	DO	+	+	-	-	-	-
2	SI	+	+	S♯	+	+	+	LA	RE	+	+	+	F♯	DO	+	+	+	-	-	-	-	-	-	-
3	-	SO	+	D♯	R♯	-	MI	-	-	SI	+	+	+	S♯	+	-	-	-	-	F♯	+	+	+	-
4	SO	-	-	-	-	-	-	D♯	+	+	+	+	+	+	+	+	+	S♯	+	+	LA	+	+	DO
5	-	R♯	+	+	-	-	-	-	-	MI	+	+	+	+	+	S♯	+	-	RE	+	+	+	+	F♯
6	+	+	+	-	LA	+	+	-	-	-	-	-	-	-	-	-	-	RE	-	-	DO	+	+	-


I
II



A microprogrammed oscillator bank

Authors: S.Cavaliere, I.Ortosecco, A.Piccialli, S.Verga
ra - Group for applied informatics - Physics
department - University of Naples.
P.Parascandolo - INFN Sezione di Napoli

ABSTRACT: In the following report we describe a digital oscillators bank in the acoustical range interfaced to a personal computer (or even mini). The hardware made up of fast discrete components carries on the synthesis of up to 80 oscillators on different wave shapes with a sampling rate of 32 KHz. The synthesis is carried on prevailingly with an additive tecnique, but it is possible to implement frequency modulation or non linear wave shaping.

Owing to the hardware techniques we used, first of all micro programming, the bank appears as a modular piece of hardware which can be used in next realizations for a wide range of digital signal processing such as IIR - FIR - FFT and others.

About the precision of the processing, the system allows the synthesis of pitches ranging from Ø to 16 KHz with a precision of 30 mHz.

Each voice has a dynamic range of 72 db (which can easily reach 96 db using a 16 x 16 bits multiplier chip).

The oscillators can oscillate with arbitrary wave shapes, written in a RAM by the computer, made of 4 K bytes samples of 12 bits, or, selecting the appropriate made of operation, made of 2 k or 1 K (therefore you can have up to 4 different wave shapes in order to synthetize a wide variety of timbres)

The system works well for real time sound synthesis and tries to define an efficient and modular architecture for digital signal processing in the range of small systems eventually personal ones.

Un banco microprogrammato di oscillatori.

S.Cavaliere, I.Ortosecco, A.Piccialli, S.Vergara.
Gruppo di informatica applicata - Istituto di Fisica Sperimentale - Università di Napoli.
P.Parascandolo - INFN - Sezione di Napoli

SOMMARIO

Nella comunicazione viene descritto un banco di oscillatori digitali nel range acustico interfacciato ad un calcolatore del tipo personal o anche mini. Il banco, realizzato con componenti discreti veloci, consente di sintetizzare 80 oscillatori su diverse forme d'onda e con una frequenza di campionamento di 32 KHz. La tecnica di sintesi é prevalentemente quella additiva ma é possibile anche la tecnica della modulazione di frequenza e della distorsione non lineare. Per le tecniche hardware utilizzate, in particolare modo la microprogrammazione, il banco si presenta come un dispositivo hardware modulare che può essere utilizzato in versioni future per altri campi della elaborazione digitale dei segnali come IIR, FIR, FFT ed altri.

Per quanto riguarda la precisione della elaborazione nel sistema realizzato vengono sintetizzate frequenze da Ø a 16 KHz con la precisione di 30 mHz. La singola voce ha una dinamica di 72 db (che può essere portata a 96 usando un moltiplicatore a 16 bit). Gli oscillatori sono realizzati su forme d'onda arbitrarie, scritte in RAM dall'esterno, da 4 k x 12 (quando occorre un rumore basso - 60.4 db -) o a scelta da 2 K e dal 1 K (quindi fino a 4 forme d'onda

diverse). Il sistema risulta abbastanza efficiente per realizzare la sintesi in tempo reale e vuole essere un tentativo di definire un'architettura efficiente e modu lare per il digital signal processing nel campo dei pic coli sistemi eventualmente di tipo personale.

INTRODUZIONE

Un aspetto piuttosto evidente della elettronica digi tale in questi anni è che essa sta conquistando alle te cniche digitali campi sempre più ampi del trattamento di segnali analogici, in particolare nel range acustico (14) La precisione e la stabilità della elaborazione, la pro-programmabilità della elaborazione o dei parametri di que sta, la possibilità di multiplexare più segnali su un solo dispositivo, sono aspetti molto attrattivi rispetto alle prestazioni usuali degli apparati analogici. Negli ultimi anni infatti, in vari centri di ricerca ed ormai anche in ambiente industriale sono cresciute le realizzio-ni in questo campo (vedi 1, 2, 4, 18). In più, recente-mente gli sviluppi della microelettronica si sono centrati molto sulla integrazione su chip di dispositivi di com-plessità e di concezione analoga a quella dei microproces-sori, orientati al trattamento digitale di segnali analo-gici per applicazioni di tipo acustico o per servomecca-nismi ed altro (vedi 15, 16, 17).

E' facile previsione quella che la conquista da parte della microelettronica di svariati campi della elaborazione analogica sarà un fenomeno sempre più rilevante con un

impatto paragonabile a quello che ha avuto dal 1971 in poi lo sviluppo dei microprocessori per l'elaborazione digitale.

Questo anche se le prestazioni di questi dispositivi attualmente consentono solo alcuni tipi di elaborazione con precisione ancora limitata e non si prestano comunque alla sintesi di segnali.

In questo quadro risulta quindi importante realizzare strutture hardware con componenti MSI/LSI che consentono elaborazioni con una precisione molto spinta, programmabilità della elaborazione e modularità, ad alta velocità e funzionanti in tempo reale: approfondire quindi la ricerca per definire le architetture hardware più adatte alla sintesi, che riescono a realizzare elaborazioni sofisticate ed alta qualità del suono; aspetti quest'ultimi che consentono evidentemente grossi avanzamenti nell'informatica musicale.

Architettura del sistema

Un sistema generale per l'elaborazione digitale di segnali può avere la struttura di Fig.1.

La separazione della memoria di lavoro del calcolatore veloce (memoria dati) dalla memoria delle istruzioni (memoria di programma) consente di realizzare permanentemente una sovrapposizione delle fasi di "fetch" e di "execute" con conseguente aumento della velocità di calcolo anche grazie ad una struttura di pipeline che sincronizza il flusso delle informazioni alla massima velocità compatibile con la tecnologia TTL (circa 16 MHz). La velocità di calcolo viene inoltre incrementata dal parallelismo delle istruzioni logico aritmetiche e computazionali realizzate interamente in hardware da una ALU, da un Moltiplicatore veloce, da un registro di ritardo per

le convoluzioni e da una memoria che calcola funzioni di trasferimento programmabili (vedi 19 - 4) Le elaborazioni vengono fatte a seconda delle varie unità con una precisione di 20-24 bit e 16 bit, precisione che consente elaborazioni molto sofisticate (vedi 4, 7, 8). Un sistema di conversione analogico digitale e digitale analogico (ADC/DAC) viene visto dal sistema come un blocco di locazioni di memoria indirizzabili in lettura (ADC) o in scrittura (DAC).

Del sistema fa parte infine un interfaccia parallela asincrona con cui vengono aggiornati dinamicamente i parametri della elaborazione. Le memorie del sistema appaiono multiplexate in uno spazio ridotto, nella memoria del μP per consentire trasferimenti più veloci in scrittura (e in lettura per il debug automatico del sistema). Per le memorie più capaci il trasferimento può essere fatto in DMA. Il microprocessore sarà a 8 bit o meglio a 16 bit per incrementare la velocità del flusso di informazioni nelle due direzioni (ed in futuro a 32 bit).

La struttura descritta, con opportuni microprogrammi può compiere ad alta velocità, con precisione molto spinta svariati tipi di elaborazione di segnali in un range di frequenze che va anche molto altre il range audio.

Di questa struttura generale é stata realizzata una versione dedicata (orientata in particolare alla sintesi di segnali acustici) e sono allo studio versioni dedicate a compiti di filtraggio digitale (FIR-IIR), convoluzioni, calcolo di polinomi, risoluzioni di equazioni differenziali nel campo discreto.

La struttura realizzata, é mostrata in uno schema a blocchi in fig.2. Il flusso di dati nel sistema é stato

reso meno flessibile di quello della struttura generale in funzione del tipo particolare di blocchi computazionali da realizzare cioé oscillatori.

Questo a vantaggio di notevoli semplificazioni nella struttura hardware. La struttura hardware ha un certo numero di registri di pipeline che servono insieme da separatori di bus e, abilitati e clockati da microprogramma, a gestire la direzione del flusso di dati, sincronizzando il funzionamento delle varie unità di calcolo ognuna alla massima velocità.

Descrizione del sistema: il microprogramma

Il microprogramma fornisce indirizzi alle memorie, segnali di controllo delle abilitazioni delle varie unità sui bus, segnali di clock per il trasferimento dei dati, segnali di scrittura alle memorie e garantisce l'accesso dall'esterno (microprocessore) o il trasferimento di dati all'esterno (convertitori). Per garantire un tempo di campionamento costante esso non ha istruzioni di salto. E' realizzato con un contatore ed una PROM veloce (che può essere una RAM) e consiste di un ciclo elementare di calcolo di un singolo oscillatore che viene ripetuto 80 volte nel tempo di spazzolamento di tutta la memoria, ogni volta su indirizzi diversi forniti in parte dalla stessa PROM secondo lo schema di figura 3.

In fig.4 é indicata la temporizzazione del microciclo elementare (il clock é a 60 ns) e le relative operazioni compiute.

L'unità realizzata, in cui le interconnessioni sono fisse (nella versione attuale con struttura "dedicata") é

indicata in fig.5 struttura(a) essa viene multiplexata su 80 diversi set di dati consentendo quindi la sintesi di 80 oscillatori.Connettendo inoltre l'uscita di un blocco con un terzo ingresso dell'oscillatore si può realizzare la sintesi per modulazione di frequenza (fig.5 struttura (b))

Descrizione del sistema: l'hardware

Il sistema realizzato su un'unica piastra wirewrap é costituito da integrati TTL prevalentemente del tipo Schttky. (103 integrati MSI/LSI ed un multiplicatore TRW)

La memoria dati, in cui in ogni microciclo viene fatta una operazione di scrittura o di lettura a 60 ns, é una memoria veloce con tempi di accesso di 30-35 ns. L'unità aritmetica é realizzata con sommatori veloci e fa una operazione in un microciclo. Il moltiplicatore veloce é un moltiplicatore integrato a 12 bit con un tempo di moltiplicazione di 120 ns per operazioni pipeline (vedi 1]). La memoria funzione é una RAM da 4k x 12 e può essere usata a scelta come unica memoria da 4k o divisa in pagine da 2k o da 1k per sintetizzare oscillatori su forme d'onda diverse. Un accumulatore finale accumula i campioni degli oscillatori su 4 canali digitali distinti a 16 bit.

I convertitori finali sono a 12 bit (dinamica di 72 db) ma possono essere sostituiti da convertitori a 16 bit (dinamica di 96 db).

Interfaccia con il microprocessore

I parametri dei segnali sintetizzati,frequenze ampiezze

e forma d'onda vengono controllati con una interfaccia con un personal computer APPLE II realizzato su microprocessore 6502. Le memorie del sistema ed alcuni registri di controllo appaiono tutte multiplexate in uno spazio di memoria da 1k del microprocessore.

Il trasferimento "memory mapped" é asincrono: nel microprogramma sono previste delle finestre per l'eventuale accesso dall'esterno. Questo trasferimento é bidirezionale per consentire anche la lettura dalle memorie e da punti di test predisposti in anticipo. In tal modo risulta facilitato il compito di un debug anche dinamico del sistema, grazie ad appositi programmi di test.

Conclusioni

Il sistema realizzato é stato messo a punto per quanto riguarda la parte hardware. Il software di gestione é ancora da realizzare, almeno in versione definitiva.

I risultati sono comunque soddisfacenti; sembra cioé possibile definire una struttura hardware che, ridotta rispetto alla complessità ed ai costi di grossi sistemi di elaborazione in tempo reale, consente comunque prestazioni non degradate per quanto riguarda la precisione della elaborazione e la sua programmabilità: una struttura hardware che funzioni in tempo reale che sia modulare e passibile di sviluppi per implementare altri tipi di elaborazione digitale.

BIBLIOGRAFIA

Sistemi discreti "da laboratorio"

1) Alles H.G. A Portable Digital Sound Synthesis System C.M.J. Vol. 1 N.4 p. 5-6 Aprile 1977
2) Alles H.G. Di Giugno P. - A one-card 64 - Channel Digital Sintetizer C.M.J. Vol. 1 N.4 p. 7-9 Aprile 1977
3) Alles H.G. - A modular approach to building large digital synthesis Systems C.M.J. Vol. 4 N.4 p. 10-13 Aprile 1977
4) Asta V. Chauveau A. G.Di Giugno, J.Kott - Il sistema sintesi digitale in tempo reale 4 X - AUTOMAZIONE E STRUMENTAZIONE N.2 Febbraio 1980 p. 119-133
5) Tierney J. Rader C.M., Gold B - A digital frequency synthetizer IEEE Trans. Audio Elecroacust Vol AU-19 pp.48-56 Mar.1971
6) Moorer J.A. - Signal Processing aspects of computer music A survey C.M.J. Vol. 1 N.1- February 1977 p. 4-37
7) Snell J. - Design of a digital oscillator which will generate up to 256 low distortion sine waves in real time C.M.J. Vol. 1 N.2 - April 1977 pp.4-25
8) Moore F.R. - Table look up noise for Sinusoidal Digital oscillators C.M.J. Vol. 1 N.2 April 1977 pp.26-29
9) Bolognesi T., Casini A., Castellini G. - Sintesi dei suoni ed elaborazione di strutture musicali - Informatica 1980 pp.60-65 (v. anche Atti del III Convegno di Informatica musicale-Milano 1980)
10) Alles H.G. - Music synthesis unsing real time digital techniques - Proceedings of the IEEE Vol.68 N.4 April 1980

CHIPS per il digital signal processing

11) TRW Product note: LSI multipliers H.J. series 1978
TRW Product note - Multipliers accumulators 1979
TRW Product note - Digital correlator 1978
12) Geist D.J. - MOS Processors picks up speed with bipolar multipliers Electronics July 7 -1977 pp.113-115
13) Schirm L IV - Packing a signal processor onto a single digital board-Electronics December 20, 1979 pp. 109-115
14) Posa J.G. - Signal processors spur changes Electronics February 14 - 1980 pp. 100-101
15) Hoff M.E. Townsend M. - Single-chip n. MOS microcomputer processes signals in real time Electronics March 1979 pp. 109-110
16) Blasco R.W. - V-MOS chip joins microprocessor to handle signalsim real time Electronics August 30 - 1979 pp.131-138
17) Capece R.P. - Digital n-MOS chip handles analog signals at record 200 KHz - Electronics December 4, 1980

Sistemi commerciali

18) Strawn John - Report from the 1980 Audio Engineering Society convention in Los Angeles C.M.J. 1980 (panoramica dei sistemi disponibili) Vedi anche anche C.M.J. Products of interest: anni 1979-80-81

Per gli aspetti di digital signal processing

19) Rabiner L. Gold B. - Theory and application of digital signal processing 1979 - Prentice Hall International cap. 9 - cap 11

20) Robinson E.A., Silvia M.T. - Digital Signal Processing and Time series Analysis. San Francisco, CA - Holden-Day 1978.

21) Oppenhein A.V., Shafer R.W. - Digital Signal Processing Englewood Cliffs. N.J. Prentice_Hall, 1975.

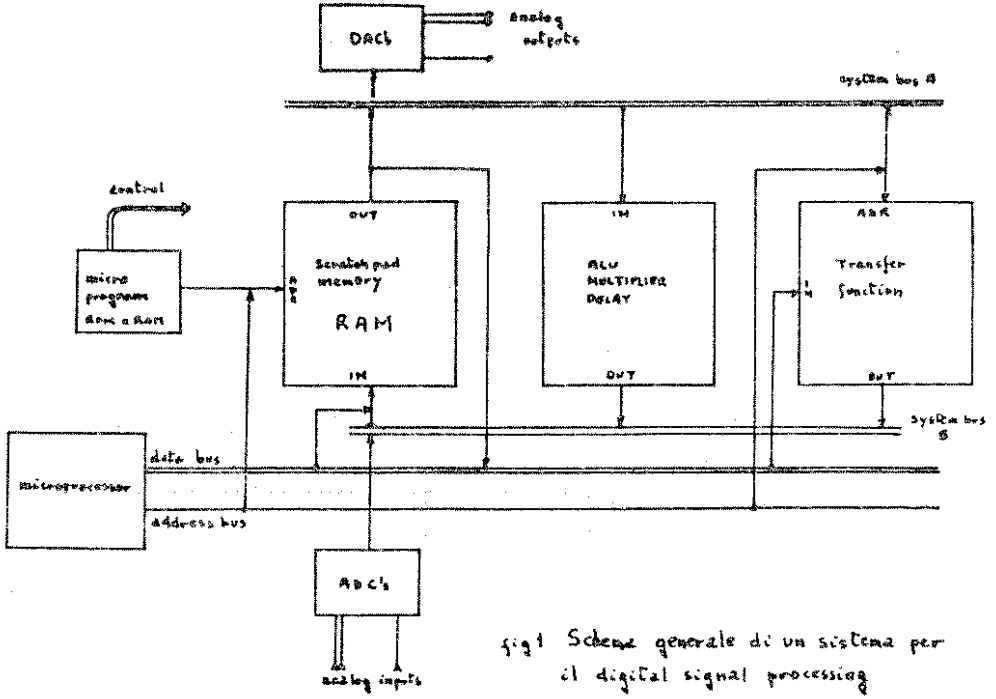


fig 1 Schema generale di un sistema per il digital signal processing

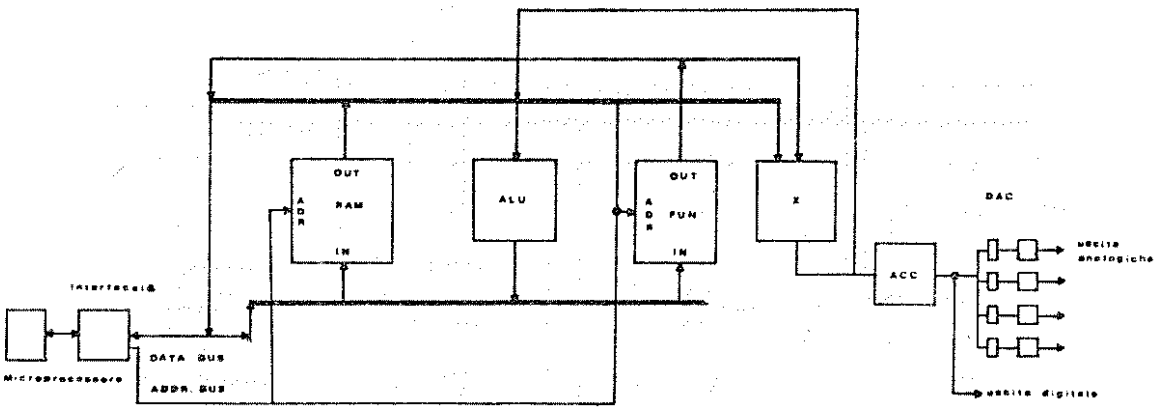


FIG. 2

IL BANCO DI OSCILLATORI

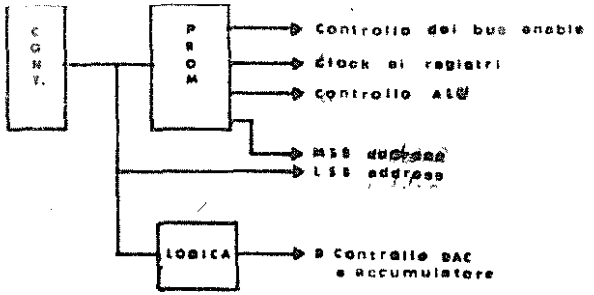


FIG. 3

LOGICA DI MICROPROGRAMMA

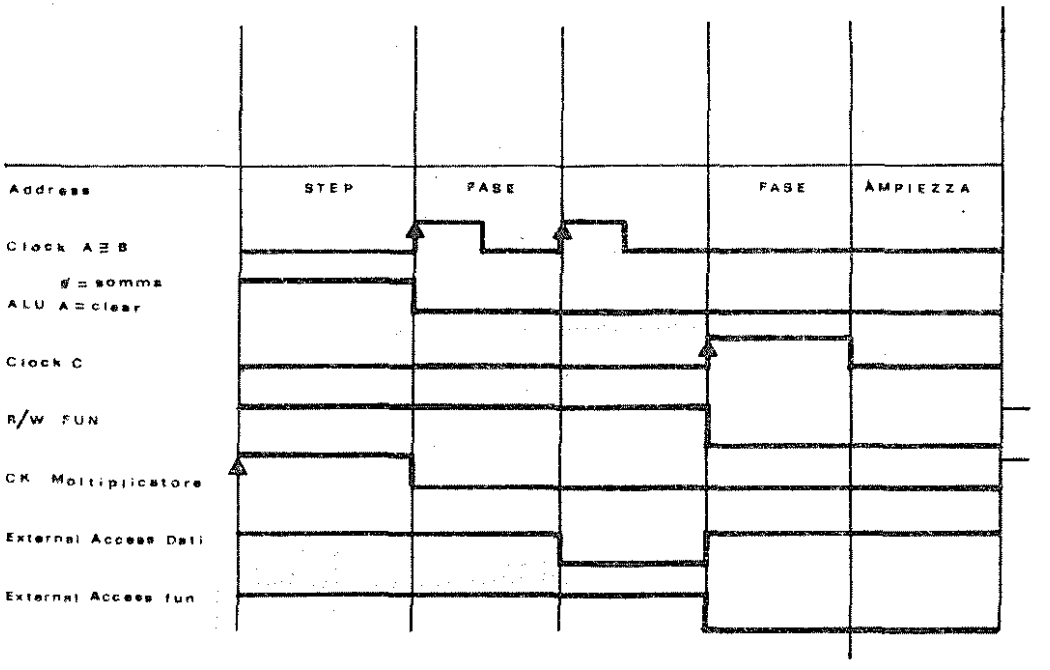
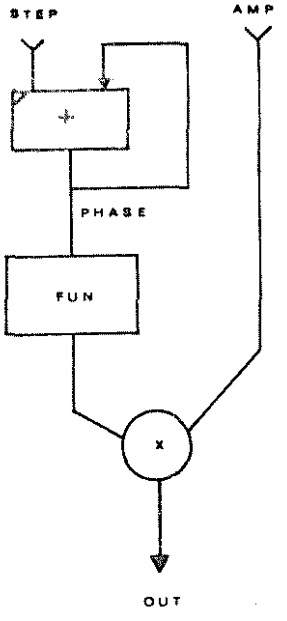
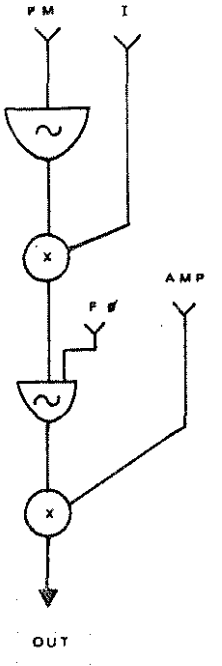


FIG 4

MICROPROGRAMMA



(a)



(b)

FIG. 5

SINTESI DI SUONI MEDIANTE FUNZIONE DISTORCENTE CON POLI COMPLESSI CONIUGATI

Giovanni DE POLI

C.S.C. - Istituto di Elettrotecnica e Elettronica
Università di Padova

Riassunto

Viene presentata una tecnica di sintesi non lineare mediante una trasformazione istantanea, descritta da una funzione con due poli complessi coniugati. Sono analizzati gli spettri che possono essere ottenuti al variare dei parametri ed è illustrato il procedimento di calcolo. Ne vengono infine discussi i vantaggi e i problemi nelle applicazioni musicali.

Abstract

A nonlinear synthesis technique using a memoryless transformation described by a function with two complex conjugate poles is presented.

The spectra obtained by varying the different parameter values are analyzed and the computing procedure is illustrated. It is shown how this technique can be advantageously applied to musical synthesis.

Con l'evoluzione dell'elettronica e dei calcolatori, si va sempre più diffondendo la produzione del suono per via numerica. Viene chiamata tecnica di sintesi il procedimento che consente di generare un suono ossia, nel caso di segnali numerici, la formula matematica che consente di calcolare il valore dei campioni che rappresentano il suono. Questa espressione matematica in genere dipende da vari parametri di ingresso che vengono specificati per ottenere uno fra tutti i suoni producibili da quell'espressione. Questi parametri possono essere costanti o variabili nel tempo.

Le tecniche di sintesi possono essere divise in tecniche di generazione e tecniche di trasformazione. Si parla di generazione quando i parametri di ingresso, che descrivono un suono sono costanti o lentamente (con banda <20 Hz) variabili nel tempo mentre si parla di trasformazione se in ingresso si hanno funzioni audio. In genere in quest'ultimo caso le funzioni di ingresso sono prodotte a loro volta da qualche semplice tecnica di generazione ed è nell'operazione di trasformazione che il suono acquista quelle caratteristiche acustiche che lo rendono musicalmente interessante.

Dato un sistema che trasforma un segnale di ingresso in uno di uscita in base ad un insieme di regole o procedure prefissate, si definisce come trasformazione il modello matematico che descrive il comportamento esterno del sistema. La trasformazione consiste in due classi di segnali X e Y, che rappresentano rispettivamente la classe dei segnali in ingresso e la classe dei segnali in uscita e di un funzionale Φ: $x \to y$ in cui ad ogni segnale $x \in X$ corrisponde uno ed un solo segnale $y=\Phi[x] \in Y$. In tale modello le regole di elaborazione sono condensate nel funzionale Φ.

Una trasformazione si dice lineare se vale il principio di sovrapposizione:

$$\Phi[a_1x_1+a_2x_2] = a_1\Phi[x_1] + a_2\Phi[x_2]$$

per ogni a_1, a_2 R costanti e x_1, x_2 X.

Una trasformazione si dice invariante se la $x(t) \to y(t)$ implica la $x(t-t_o) \to y(t-t_o)$ per ogni traslazione t_o applicabile al segnale di ingresso.

Una trasformazione lineare invariante viene chiamata filtro e da luogo alla cosidetta tecnica di sintesi sottrattiva.

Da alcuni anni vanno sempre più diffondendosi nelle applicazioni musicali le trasformazioni non lineari.

Fra le tecniche non lineari particolare importanza per la semplicità di realizzazione, assumono le trasformazioni invarianti istantanee (senza me

moria) ad un ingresso. In esse se x(t) è il segnale di ingresso e y(t) il segnale di uscita, vale la relazione

$$y(t) = f[x(t)]$$

essendo f una funzione ad un solo valore. Si ha cioè che il valore del segnale di uscita in un certo istante dipende solo dal valore del segnale in ingresso in quell'istante. Si può immaginare che il segnale x(t) venga distorto dalla funzione f detta talvolta per questa ragione funzione distorcente. Variando l'ampiezza del segnale in ingresso, varia sostanzialmente lo spettro del segnale in uscita; risulta così agevole ottenere spettri complessi e dinamici.

Recentemente è stato studiato da Le Braun(1979) ed Arfib (1979) il caso in cui la funzione distorcente sia esprimibile con un polinomio e l'ingresso sia un coseno. E' stato poi studiato da Reinhard (1979) il caso in cui l'ingresso sia costituito da una combinazione lineare di due coseni di frequenza diversa e la funzione distorcente sempre un polinomio.

Sebbene non presentata in questa forma anche la tecnica di sintesi sommatoria discreta proposta da Moorer (1976) può essere pensata come una trasformazione di questo tipo in cui la funzione distorcente ha un polo reale. Formule simili sono state proposte da Lehmann e Brown (1976).

In questo lavoro ci si propone di studiare il caso in cui la funzione distorcente sia una funzione razionale e in particolare presenti due poli complessi coniugati al denominatore.

POLI COMPLESSI CONIUGATI: SPETTRO STATICO

La funzione che esprime la trasformazione sia

$$f(x) = \frac{1}{ax^2+bx-c} = \frac{1}{a(x-p)(x-\hat{p})}$$

in cui sono stati evidenziati i poli complessi coniugati. Essi sono dati da

$$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

in cui il discriminante $D=b^2-4ac$ è minore di zero. Si ha quindi

$$p = \frac{-b}{2a} + i\,\frac{\sqrt{-D}}{2a}$$

$$\hat{p} = \frac{-b}{2a} - i\,\frac{\sqrt{-D}}{2a}$$

Essendo i poli complessi coniugati il denominatore non diventa mai nullo e quindi la funzione f(x) è definita su tutto l'asse reale.

Il segnale di ingresso sia di tipo coseno espresso come $x(t)=\cos\omega t=\cos 2\pi f_o t=\cos\frac{2\pi}{T_o}t$. Essendo la trasformazione istantanea, il segnale di uscita $y(t)=f[x(t)]=f(\cos\omega t)$ risulta periodico almeno con lo stesso periodo del segnale di ingresso. Infatti da

$$\cos\omega t = \cos\omega(t+T_o) \qquad \text{si ha che}$$

$$y(t) = f(\cos\omega t) = f[\cos\omega(t+T_o)] = y(t+T_o)$$

Si noti che in casi particolari il periodo del segnale in uscita può essere un sottomultiplo di T_o, risultando cioè

$$y(t) = y(t+\frac{T_o}{k}) \qquad \text{con } k \text{ intero maggiore od uguale ad 1.}$$

Per determinare lo spettro del segnale di uscita si può quindi ricorrere alla serie di Fourier:

$$y(t) = \sum_{-\infty}^{+\infty} {}_k S_k\, e^{ik\omega t}$$

dove i coefficienti di Fourier S_k sono dati dalla

$$S_k = \frac{1}{T_o}\int_O^{T_o} y(t)\, e^{-ik\omega t}\, dt$$

Nel nostro caso

$$y(t) = f(\cos\omega t) = \frac{1}{a\cos^2\omega t + b\cos\omega t + c} = \frac{1}{a(\cos\omega t-p)(\cos\omega t-\hat{p})}$$

Per cui posto $\omega t = \xi$

$$S_k = \frac{1}{2\pi}\int_O^{2\pi} \frac{e^{-ik\xi}}{a(\cos\xi-p)(\cos\xi-\hat{p})}\, d\xi$$

Questo integrale può facilmente essere ridotto all'integrale su un contorno chiuso di una funzione analitica ad una variabile. A questo scopo si

osservi che essendo y(t) reale, per i coefficienti di Fourier vale la relazione $S_{-k}=\hat{S}_k$. Da cui

$$S_{-k} = \frac{1}{2\pi a}\int_0^{2\pi} \frac{e^{ik\xi}}{(\cos\xi-p)(\cos\xi-\hat{p})}\,d\xi = \frac{1}{2\pi a(p-\hat{p})}\left[\int_0^{2\pi}\frac{e^{ik\xi}}{\cos\xi-p}\,d\xi - \right.$$

$$\left. - \int_0^{2\pi} \frac{e^{ik\xi}}{\cos\xi-\hat{p}}\,d\xi\right]$$

Introducendo al posto della variabile reale ξ la variabile complessa $z=e^{i\xi}$. Al variare di ξ nell'intervallo $(0,2\pi)$ la variabile complessa z descrive la circonferenza di raggio unitario. Inoltre vale

$$d\xi=\frac{dz}{iz} \qquad \cos\xi = \frac{1}{2}\,(e^{i\xi}+e^{-i\xi}) = \frac{1}{2}\,(z+\frac{1}{z}) \qquad \operatorname{sen}\xi = \frac{1}{2i}\,(z-\frac{1}{z})$$

Sostituendo si ottiene

$$S_{-k} = \frac{1}{i\pi a(p-\hat{p})}\left[\oint\frac{z^k}{z^2-2pz+1}\,dz - \oint\frac{z^k}{z^2-2\hat{p}z+1}\,dz\right]$$

Per calcolare questi integrali si può ricorrere al teorema dei residui che afferma che se f(z) è una funzione analitica ovunque in un dominio chiuso, eccetto un numero finito di punti singolari (poli), vale

$$f(z)dz = 2\pi i \sum \operatorname{Res}\{f(z)\}$$

essendo la somma estesa a tutti i punti singolari all'interno del contorno di integrazione.

Il residuo di una funzione analitica in un polo p di ordine m è dato da

$$\operatorname{Res}\{f(z)\} = \lim_{z\to p}\frac{1}{(m-1)!}\,\frac{d^{m-1}}{d\cdot z^{m-1}}\,\{(z-p)^m\cdot f(z)\}$$

Nel caso di polo semplice risulta

$$\operatorname{Res}\{f(z)\} = \lim_{z\to p}\{(z-p)\cdot f(z)\}$$

Nel nostro caso il contorno di integrazione è il cerchio di raggio unitario percorso in senso positivo. Ponendo il denominatore uguale a zero si

ha un'equazione reciproca, in cui cioè le radici sono l'una reciproca all'altra. Ne risulta che uno solo dei due poli è interno al cerchio di raggio unitario, non potendo essere entrambi di modulo unitario, essendo i poli della funzione distorcente complessi.

Dall'equazione

$$z^2 - 2pz + 1 = 0$$

si ottiene

$$z_{1,2} = p \pm \sqrt{p^2-1}$$

Chiamata d la radice in modulo di 1, polo in cui calcolare il residuo, si ha che

$$p = \frac{d+d^{-1}}{2}$$

$$\hat{p} = \frac{\hat{d}+\hat{d}^{-1}}{2}$$

$$S_{-k} = \frac{1}{i\pi a(p-\hat{p})}\left[\oint \frac{z^k}{(z-d)(z-d^{-1})}\,dz - \oint \frac{z^k}{(z-\hat{d})(z-\hat{d}^{-1})}\,dz\right] =$$

$$= \frac{2\pi i}{i\pi a(p-\hat{p})}\left[\frac{d^k}{d-d^{-1}} - \frac{\hat{d}^k}{\hat{d}-\hat{d}^{-1}}\right]$$

Posto $d = \rho\, e^{i\theta}$, $g = \frac{2}{d-d^{-1}} = \frac{2}{\cos\theta(\rho-\rho^{-1})+i\,\mathrm{sen}\theta(\rho+\rho^{-1})} = \gamma\, e^{i\delta}$

quindi

$$|\gamma|^2 = \frac{4}{\rho^2+\rho^{-2}-2\cos 2\theta} \qquad tg\delta = tg\theta\,\frac{\rho+\rho^{-1}}{\rho^{-1}-\rho}$$

$$S_{-k} = \frac{1}{a(p-\hat{p})}\left(\rho^k e^{ik\theta}\gamma e^{i\delta} - \rho^k e^{-ik\theta}\gamma e^{-i\delta}\right) = \frac{2i}{a(p-\hat{p})}\,\gamma\,\rho^k\,\mathrm{sen}(k\theta+\delta)$$

Si osservi innanzitutto che i coefficienti sono reali, essendo il segnale una funzione pari e quindi scomponibile in somma di coseni.

Risulta quindi che $S_{-k} = S_k$ e l'espressione trovata vale sia per i coefficienti positivi che negativi.

fig. 1 (cont.)

Fig. 1 (cont.)

Nella figura 1 sono riportati alcuni spettri per vari valori di ρ e θ. Come si può vedere dalla formula e dalle figure lo spettro è composto dal prodotto di un termine sinusoidale per un esponenziale decrescente, al crescere dell'indice dell'armonica (fig. 2). Come prima considerazione si può osservare che lo spettro non è rigorosamente limitato, in quanto l'esponenziale non giunge mai a zero. In pratica si possono ignorare le componenti sotto una certa soglia. L'esponenziale determina che l'ampiezza massima delle varie armoniche decrescono con rapporto ρ; per cui se si considera di poter ignorare le armoniche con ampiezza massima inferiore ad un centesimo dell'ampiezza massima dell'esponenziale si ottiene che il massimo numero M di armoniche significative è ricavabile da

$$\rho^M \simeq \frac{1}{100} \qquad \text{e risulta} \qquad M = \frac{-2}{\lg_{10}\rho}$$

$$\rho = \left(\frac{1}{100}\right)^{1/M}$$

oppure se ρ è espresso in decibel (ρ_{dB})

$$M = \frac{-40}{\rho_{dB}} \qquad \rho_{dB} = \frac{-40}{M}$$

Ad esempio se $\rho = \frac{1}{2}$ $M = 6.64$. In tabella 1 sono riportati i valori di ρ in funzione della larghezza di banda. La periodicità della parte seno dipende invece da θ per cui si vengono ad avere massimi, in valore assoluto, distanti fra loro $\frac{\pi}{\theta}$ armoniche. Ad esempio se $\theta = \frac{\pi}{6}$ si avranno dei formanti distanti fra loro 6 armoniche. Si ha in definitiva uno spettro a formanti ad ampiezza decrescente la cui distanza è règolata da θ e con banda dipendente da ρ.

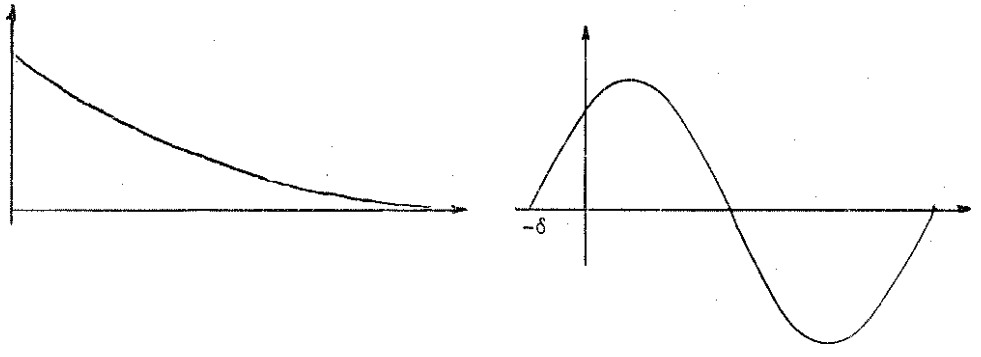


fig. 2

Tabella 1

M	ρ	ρ_{db}	$\frac{\rho+\rho^{-1}}{2}$	$\frac{\rho-\rho^{-1}}{2}$
1	0.010	-40.000	50.005	-49.995
2	0.100	-20.000	5.050	-4.950
3	0.215	-13.333	2.429	-2.213
4	0.316	-10.000	1.739	-1.423
5	0.398	-8.000	1.455	-1.057
6	0.464	-6.667	1.309	-0.845
7	0.518	-5.714	1.224	-0.706
8	0.562	-5.000	1.170	-0.608
9	0.599	-4.444	1.134	-0.534
10	0.631	-4.000	1.108	-0.477
12	0.681	-3.333	1.075	-0.393
14	0.720	-2.857	1.055	-0.335
16	0.750	-2.500	1.042	-0.292
18	0.774	-2.222	1.033	-0.259
20	0.794	-2.000	1.027	-0.232
25	0.832	-1.600	1.017	-0.185
30	0.858	-1.333	1.012	-0.154
35	0.877	-1.143	1.009	-0.132
40	0.891	-1.000	1.007	-0.115
45	0.903	-0.889	1.005	-0.103
50	0.912	-0.800	1.004	-0.092
55	0.920	-0.727	1.004	-0.084
60	0.926	-0.667	1.003	-0.077

Per quanto riguarda δ si osservi che per $\rho \to 0$ tg $\delta \simeq$ tgθ e per ρ crescente tg δ è sempre maggiore di tg θ e tende a $+\infty$: l'argomento δ tende quindi a $\frac{\pi}{2}$ e quindi la posizione del primo massimo nello spettro tende a zero.

E' possibile esprimere le varie grandezze in funzione di ρ e θ. Risulta

$$p = \cos\theta\left(\frac{\rho+\rho^{-1}}{2}\right) + i\,\mathrm{sen}\theta\left(\frac{\rho-\rho^{-1}}{2}\right)$$

$$b = -a(p+\hat{p}) = -a\cos\theta(\rho+\rho^{-1})$$

$$c = ap\cdot\hat{p} = a|p|^2 = \frac{a}{4}(\rho^2+\rho^{-2}+2\cos 2\theta)$$

$$p-\hat{p} = i\,\mathrm{sen}\theta(\rho-\rho^{-1})$$

oppure posto $q = \ln\rho = 0.11513 \cdot \rho_{dB}$ e $w = q+i\theta$

$$\rho = e^{q} \qquad d = e^{q+i\theta} = e^{w}$$

$$\cosh q = \frac{\rho^{+1}+\rho^{-1}}{2} \qquad \mathrm{senh}\, q = \frac{\rho^{+1}-\rho^{-1}}{2}$$

$$p = \cos\theta \cosh q + i \operatorname{sen}\theta \operatorname{senh} q = \cosh(q+i\theta) = \cosh w$$

$$b = -2a \cos\theta \cosh q$$

$$c = \frac{a}{2} (\cosh 2q + \cos 2\theta)$$

$$p-\hat{p} = 2 \operatorname{sen}\theta \operatorname{senh} q$$

$$g = \gamma e^{i\delta} = \frac{1}{\operatorname{senh} w}$$

L'ampiezza massima della funzione distorcente si ha in corrispondenza del minimo della parabola. L'ascissa è $x_m = \frac{-b}{2a}$ e sostituendo nella funzione si ha

$$y_m = \frac{4a}{4ac-b^2} = \frac{4a}{-D} = \frac{4}{-a(p-\hat{p})^2}$$

Si osservi che x_m è la parte reale dei poli della funzione distorcente. Si può sostituire y_m nei coefficienti di Fourier, ottenendo

$$s_k = \sqrt{\frac{y_m}{a}} \cdot \gamma \cdot \rho^k \operatorname{sen}(k\theta+\delta)$$

L'ampiezza massima del segnale è effettivamente y_m se

$$|x_m| \leq 1 \quad \rightarrow \quad |b| \leq 2 \cdot |a|$$

Altrimenti l'ampiezza massima effettiva in valore assoluto y_M diventa

$$y_M = \frac{1}{|a|-|b|+|c|}$$

$$y_M = \frac{1}{a-|b|+c} \qquad \text{per a positivo}$$

Si possono fare alcune considerazioni sul segno di a,b,c. Inanzitutto a e c hanno lo stesso segno; se sono positivi, il segnale è positivo, se sono negativi, è negativo. Il cambio di segno di b equivale a cambiare di segno il segnale di ingresso e quindi ad uno sfasamento di π; nello spettro le righe dispari cambiano segno.

Spettri dinamici

Se l'ampiezza del coseno in ingresso non è unitaria, ma varia con un indice I, allora risulta $x(t)=I\cos\omega t$

$$y(t) = \frac{1}{aI^2\cos^2\omega t + bI\cos\omega t + c} = \frac{1}{a(I\cos\omega t - p)(I\cos\omega t - \hat{p})} =$$

$$= \frac{1}{aI^2(\cos\omega t - \frac{p}{I})(\cos\omega t - \frac{\hat{p}}{I})}$$

E' come se si fosse cambiata la funzione distorcente con

$$a' = aI^2 \qquad b' = bI \qquad c' = c$$

$$p' = \frac{p}{I} \qquad \hat{p}' = \frac{\hat{p}}{I}$$

Si noti però che y'_m risulta uguale ad y_m. Infatti essendo $x' = Ix$, si fa solo un cambiamento di scala delle ascisse. L'effetto della variazione è quello di un cambiamento del modulo dei poli, inversamente proporzionale all'indice I. Nel piano complesso i poli si muovono secondo raggi con centro l'origine degli assi, avvicinandosi sempre più all'origine al crescere di I. In figura 3 sono riportati l'andamento delle varie armoniche per I da zero ad uno ed $a = 1 \quad b = 1.064 \quad c = 0.382$ corrispondenti a $\rho = 0.7 \quad \theta = \pi/3$.

Esaminiamo ora la corrispondenza che esiste tra ρ e θ e i poli p. Si ricordi che vale la relazione

$$p = \frac{d+d^{-1}}{2}$$

Ciò rappresenta una trasformazione conforme attuata dalla funzione complessa $y|z| = \frac{1}{2}(z+z^{-1})$.

Vediamo come si trasforma un reticolo di coordinate polari nel piano D, cioè come si trasformano le circonferenze a ρ costante e il fascio di rette con argomento θ. Posto

$$p = r + is = \cos\theta\ (\frac{\rho+\rho^{-1}}{2}) + i\ \mathrm{sen}\theta\ (\frac{\rho-\rho^{-1}}{2})$$

Considerando ρ costante ed eliminando θ, si ottiene

fig. 3

UNIVERSITÀ DI PADOVA - CENTRO DI CALCOLO

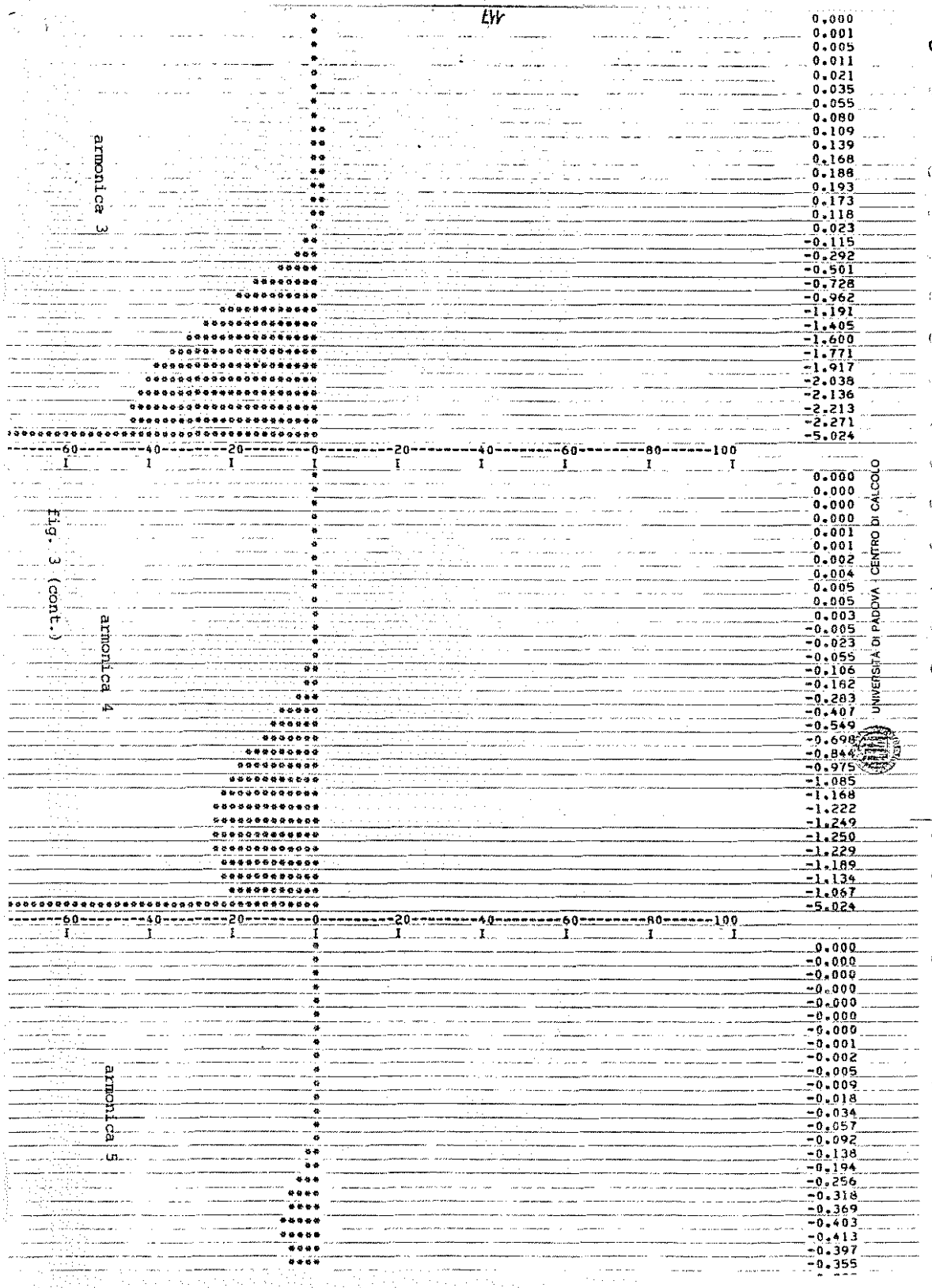

armonica 3
0.000
0.001
0.005
0.011
0.021
0.035
0.055
0.080
0.109
0.139
0.168
0.188
0.193
0.173
0.118
0.023
-0.115
-0.292
-0.501
-0.728
-0.962
-1.191
-1.405
-1.600
-1.771
-1.917
-2.038
-2.136
-2.213
-2.271
-5.024
-60 -40 -20 0 20 40 60 80 100
armonica 4
0.000
0.000
0.000
0.000
0.001
0.001
0.002
0.004
0.005
0.005
0.003
-0.005
-0.023
-0.055
-0.106
-0.182
-0.283
-0.407
-0.549
-0.698
-0.844
-0.975
-1.085
-1.168
-1.222
-1.249
-1.250
-1.229
-1.189
-1.134
-1.067
-5.024
-60 -40 -20 0 20 40 60 80 100
armonica 5
0.000
-0.000
-0.000
-0.000
-0.000
-0.000
-0.000
-0.001
-0.002
-0.005
-0.009
-0.018
-0.034
-0.057
-0.092
-0.138
-0.194
-0.256
-0.318
-0.369
-0.403
-0.413
-0.397
-0.355
UNIVERSITÀ DI PADOVA CENTRO DI CALCOLO

Fig. 3 (cont.)

$$\frac{r^2}{\left(\frac{\rho+\rho^{-1}}{2}\right)^2} + \frac{s^2}{\left(\frac{\rho-\rho^{-1}}{2}\right)^2} = 1$$

equazione di un'ellisse con semiassi $A=\frac{\rho+\rho^{-1}}{2}$ e $B=\left|\frac{\rho-\rho^{-1}}{2}\right|$ e fuochi dati da $C=1$.

Per $\rho=1$ l'ellisse degenera nel segmento $(-1,1)$; al decrescere di ρ le ellissi si allargano su tutto il piano fino a sembrare sempre più delle circonferenze. Tutte le ellissi infatti sono omofocali, hanno infatti i fuochi nei punti ± 1.

Vediamo ora come si modificano le rette passanti per l'origine con argomento θ. Eliminando la variabile ρ si ha

$$\frac{r^2}{\cos^2\theta} - \frac{s^2}{\operatorname{sen}^2\theta} = 1$$

si ottiene cioè una famiglia di iperboli di semiassi $A=|\cos\theta|$ $B=|\operatorname{sen}\theta|$ e fuochi dati da $C=1$. Le iperboli sono cioè omofocali con le ellissi sopra considerate ed hanno per asintoti le rette $y=x\ \mathrm{tg}\theta$ $y=-x\mathrm{tg}\theta$ passanti per l'origine. Per $\theta=0$ l'iperbole degenera nelle semirette dell'asse reale $(-\infty,-1)$ e $(1,+\infty)$; per $\theta=\frac{\pi}{2}$ degenera nell'asse immaginario. Si osservi inoltre che in ogni punto l'ellisse e l'iperbole sono perpendicolari. I due piani sono rappresentati in fig. 4.

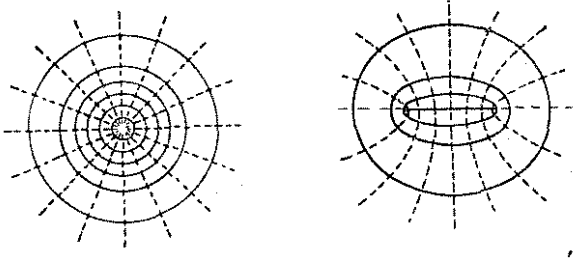


fig. 4

Va precisato che, essendo nel nostro caso $0 < \rho < 1$, risulta che $\frac{\rho-\rho^{-1}}{2}$ è minore di zero. Ciò significa che per θ compreso tra 0 e $\pi/2$, l'argomento di p varia da 0 a $-\pi/2$, mentre l'argomento di $\hat{p}$ varia da 0 a $\pi/2$. Per cui il punto $\hat{d}$ ed il punto $\hat{p}$ possono essere rappresentati nello stesso quadrante e per $\rho \to 0$ hanno lo stesso argomento.

Seguendo quindi nel piano P lo spostamento dei poli lungo raggi che partono dal centro, si possono determinare i valori ρ e θ via via corrispondenti e quindi le caratteristiche dello spettro. In particolare si osserva che al crescere di I, il polo p tende all'origine; per cui $\rho \to 1$ e $\theta \to \pm\pi/2$. Si ha la proprietà che il numero di armoniche significative cresce al crescere dell'indice.

La normalizzazione di ampiezza può essere fatta o sul valore efficace o sul valore massimo. Il calcolo del valore efficace porta a formule piuttosto complicate, che qui non verranno date. Più semplice è quella sul valore massimo. Supposto a positivo si osserva che per $I \geq \frac{|b|}{2a}$ l'ampiezza massima vale y_m e non dipende da I, mentre per I inferiore essa vale $\frac{1}{aI^2-|b|I+c}$; per cui la funzione di normalizzazione G(I) per $I < \frac{|b|}{2a}$ è una parabola e vale

$$G(I) = y_m(aI^2 - |b|I + c),$$

per $I \geq \frac{|b|}{2a}$ $G(I) = 1$.

In fig. 5 e 6 sono riportati la funzione di normalizzazione e l'andamento delle armoniche del segnale di fig. 3 normalizzato.

Numeratore

Se il segnale y(t) prima trovato viene moltiplicato per un segnale $s_1(t)$ corrisponde ad aggiungere all'espressione di y(t) un numeratore variabile nel tempo. Lo spettro risultante è dato dalla convoluzione degli spettri dei due segnali moltiplicati. Particolare interesse presenta il caso in cui $s_1(t)$ sia un termine sinusoidale con lo stesso periodo T_o di x(t). Se $s_1(t) = \cos(\omega t+\varepsilon) =$
$= \cos\left(\frac{2\pi}{T_o} t + \varepsilon\right)$

$$y_1(t) = \frac{\cos(\omega t+\varepsilon)}{a\cos^2\omega t + b\,\omega t + c}$$

Anche $y_1(t)$ è periodico con lo stesso periodo. I coefficienti di Fourier si ricavano con il procedimento sopra esposto.

Posto quindi $z = e^{i\omega t}$ risulta

0.000	I	3.565
0.050	I	3.336
0.100	I	2.875
0.150	I	2.457
0.200	I	2.108
0.250	I	1.803
0.300	I	1.541
0.350	I	1.333
0.400	I	1.175
0.450	I	1.068
0.500	I	1.010
0.550	I	1.000
0.600	I	1.000
0.650	I	1.000
0.700	I	1.000
0.750	I	1.000
0.800	I	1.000
0.850	I	1.000
0.900	I	1.000
0.950	I	1.000
1.000	I	1.000

fig. 5

UNIVERSITÀ DI PADOVA - CENTRO DI CALCOLO
armonica 2
UNIVERSITÀ DI PADOVA - CENTRO DI CALCOLO
armonica 1
armonica 0

fig. 6

fig. 6 (cont.)

$$S_{-k} = \frac{1}{2i \cdot \pi a(p-\hat{p})} \left[\int \frac{(z\, e^{i\varepsilon} + z^{-1}\, e^{-i\varepsilon}) z^k}{z^2 - 2pz + i}\, dz - \int \frac{(z\, e^{i\varepsilon} + z^{-1}\, e^{-i\varepsilon}) z^k}{z^2 - 2\hat{p}z + 1}\, dz \right]$$

Chiamata ancora d la radice in modulo minore di uno del primo denominatore risulta

$$S_{-k} = \frac{1}{a(p-\hat{p})} \cdot \left[\frac{d\, e^{i\varepsilon} + d^{-1}\, e^{-i\varepsilon}}{d - d^{-1}}\, d^k - \frac{\hat{d}\, e^{i\varepsilon} + \hat{d}^{-1}\, e^{-i\varepsilon}}{\hat{d} - \hat{d}^{-1}}\, \hat{d}^k \right]$$

Per k=0 bisogna aggiungere, per ciascun integrale, il residuo nello zero. Tali residui sono uguali e quindi si eliminano; l'espressione data vale quindi anche per k=0.

Dopo alcuni passaggi si ottiene

$$S_{-k} = \frac{1}{a(p-\hat{p})} \cdot \cos\varepsilon \left(\frac{d+d^{-1}}{d-d^{-1}}\, d^k - \frac{\hat{d}+\hat{d}^{-1}}{\hat{d}-\hat{d}^{-1}}\, \hat{d}^k \right) + i\, \mathrm{sen}\varepsilon\, (d^k - \hat{d}^k)$$

posto $\frac{d+d^{-1}}{d-d^{-1}} = \gamma_1\, e^{i\delta_1}$ e ricordando che $S_k = \hat{S}_{-k}$ risulta

$$S_{+k} = \frac{2i}{a(p-\hat{p})} \left(\cos\varepsilon\, \gamma_1\, \rho^k\, \mathrm{sen}(k\theta + \delta_1) - i\, \mathrm{sen}\varepsilon\, \rho^k\, \mathrm{sen}k\theta \right)$$

In particolare per un numeratore di tipo coseno, $\varepsilon = 0$ e S_k risulta reale

$$S_k = \frac{2i}{a(p-\hat{p})}\, \gamma_1\, \rho^k\, \mathrm{sen}(k\theta + \delta_1)$$

per un numeratore di tipo seno, $\varepsilon = \pi/2$

$$S_k = -\frac{2i}{a(p-\hat{p})}\, i\, \rho^k\, \mathrm{sen}k\theta$$

immaginario. In questo caso il valore medio S_o del segnale è nullo.

Al variare di ε, S_k descrive un'ellisse con semiassi proporzionali a $\gamma_1\, \mathrm{sen}(k\theta + \delta_1)$ e $\mathrm{sen}\, k\theta$. Il modulo varia quindi con continuità dall'ampiezza del coefficiente corrispondente ad un ingresso coseno a quella corrispondente ad un ingresso seno.

UNIVERSITÀ DI PADOVA - CENTRO DI CALCOLO

Fig. 7

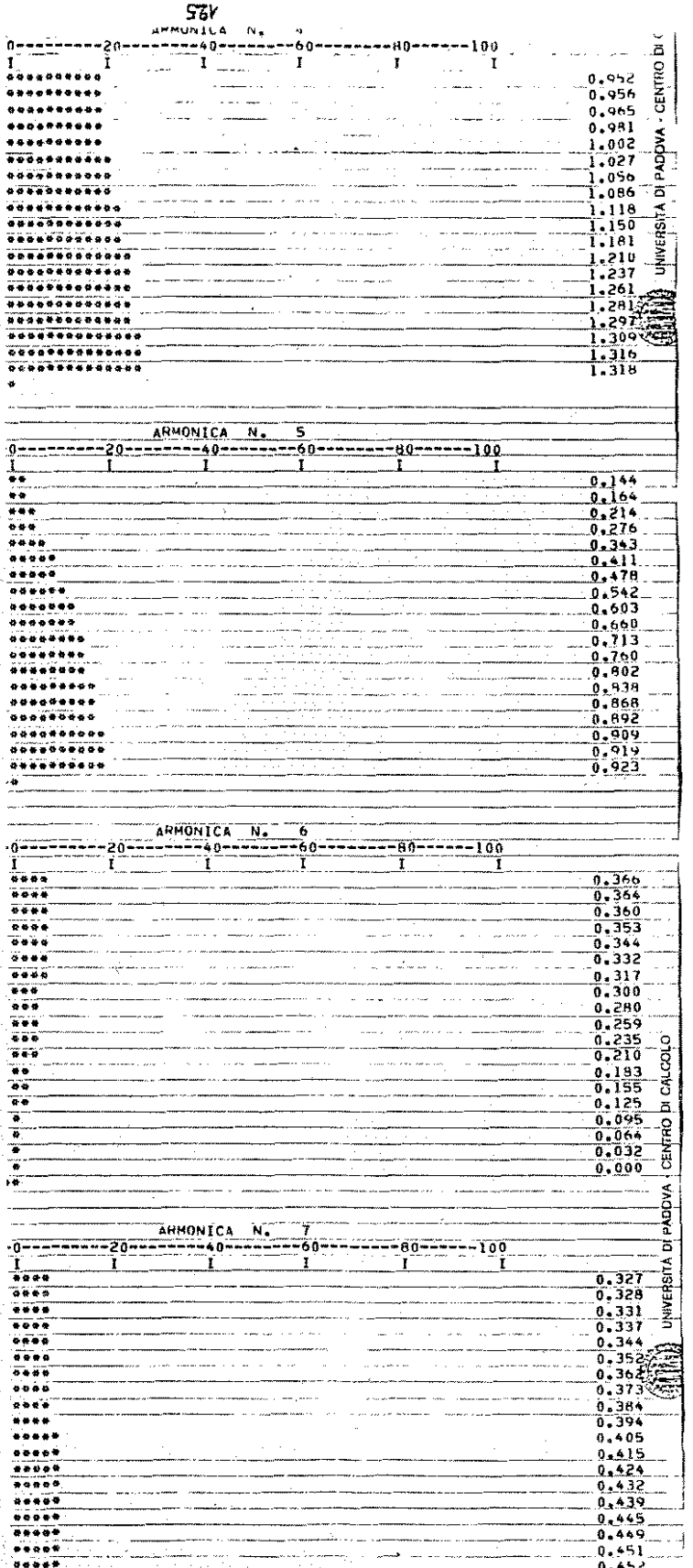

ARMONICA N. 4
0----------20----------40----------60----------80----------100
0.952
0.956
0.965
0.981
1.002
1.027
1.056
1.086
1.118
1.150
1.181
1.210
1.237
1.261
1.281
1.297
1.309
1.316
1.318
ARMONICA N. 5
0----------20----------40----------60----------80----------100
0.144
0.164
0.214
0.276
0.343
0.411
0.478
0.542
0.603
0.660
0.713
0.760
0.802
0.838
0.868
0.892
0.909
0.919
0.923
ARMONICA N. 6
0----------20----------40----------60----------80----------100
0.366
0.364
0.360
0.353
0.344
0.332
0.317
0.300
0.280
0.259
0.235
0.210
0.183
0.155
0.125
0.095
0.064
0.032
0.000
ARMONICA N. 7
0----------20----------40----------60----------80----------100
0.327
0.328
0.331
0.337
0.344
0.352
0.362
0.373
0.384
0.394
0.405
0.415
0.424
0.432
0.439
0.445
0.449
0.451
0.452
UNIVERSITA DI PADOVA - CENTRO DI CALCOLO
UNIVERSITA DI PADOVA - CENTRO DI CALCOLO

fig. 7 (cont.)

In fig. 7 sono riportati gli andamenti dei moduli delle armoniche per ε che varia da θ a $\pi/2$. Come si può vedere alcuni sono crescenti, altri decrescenti, rendendo quindi le variazioni dello spettro acusticamente interessanti.

Ciò suggerisce una applicazione. Se la frequenza del numeratore è leggermente differente da quella del denominatore, ε cresce linearmente, S_k ruota sull'ellisse e le armoniche avranno ampiezza pulsante. Ad esempio se la differenza di frequenza $\Delta F = 1$ Hz, risulta $\varepsilon = 2\pi\Delta F.t$. Per cui S_k fa una rotazione al secondo e l'ampiezza di un'armonica passa dal massimo al minimo in un quarto di secondo; nel quarto di secondo successivo passa dal minimo al massimo e così via.

Se il numeratore ha frequenza multipla si può calcolare lo spettro con lo stesso procedimento.

Si può osservare che se il segnale a numeratore ha la stessa frequenza del denominatore ed è un segnale pari, esso può essere espresso come un polinomio in coseno. Ad esempio

$$y_2(t) = \frac{\cos 4\,\omega t}{a\cos^2\omega t + b\cos\omega t + c} = \frac{8\cos^4\omega t - 8\cos^2\omega t + 1}{a\cos^2\omega t + b\cos\omega t + c}$$

chiamata

$$g_2(x) = \frac{8x^4 - 8x + 1}{ax^2 + bx + c} \qquad \text{e} \qquad x = \cos\omega t$$

Risulta

$$y_2(t) = g_2(\cos\omega t)$$

Il segnale risultante è ottenibile direttamente con un'unica funzione distorcente in cui il polinomio al numeratore viene determinato con le trasformazioni usuali della distorsione non lineare polinomiale.

Se il segnale a numeratore ha la stessa frequenza del denominatore ma è una funzione dispari, esso può essere scomposto nel prodotto di un seno per una funzione pari. Risulta cioè

$$y_3(t) = \text{sen}\omega t \cdot g_3(\cos\omega t)$$

Ad esempio se $y_3(t) = \frac{\text{sen } 4\omega t}{a \cos^2 \omega t + b \cos \omega t + c}$ $g_3(x) = \frac{8x^3 - 4x}{a x^2 + b x + c}$

Una scelta opportuna del numeratore rende lo spettro rigorosamente limitato in banda. Moorer (1976) dà alcuni esempi di scelta per il caso di polo semplice. Si può osservare che ciò corrisponde al fatto che il numeratore è dividibile per il denominatore.
Ad esempio se $f(x) = \frac{1}{x^2 + 2x + 1.5}$ e si ha come numeratore $s_1(t) = \cos 4\omega t + 17$ risulta

$$g_1(x) = \frac{8x^4 - 8x^2 + 18}{x^2 + 2x + 1.5} = 8x^2 - 16x + 12$$

$$y_1(t) = s_1(t) \cdot f(\cos \omega t) = g_1(\cos \omega t) = 4 \cos^2 \omega t - 16 \cos \omega t + 8$$

In generale se il numeratore è di tipo sinusoidale con frequenza F_N non multipla di quella del denominatore F_O, allora si avrà il tipico spettro prodotto dalla sintesi moltiplicativa con parziali a frequenza $F_N \pm k F_O$ con k intero. Se il numeratore ha frequenza F_N, ma è composto da più armoniche, lo spettro risultante ha parziali a frequenza $jF_N \pm k F_O$. A seconda della scelta della frequenza del numeratore, si possono produrre spettri armonici e inarmonici in analogia con le altre tecniche di sintesi non lineari come la modulazione di frequenza (Chowning 1973).

IMPLEMENTAZIONE

Essendo la trasformazione, applicata al segnale, istantanea, essa può essere calcolata e tabulata una volta per tutte per x variante nell'intervallo $(-I_{MAX}, I_{MAX})$. Al momento della sintesi basterà usare un modulo di ricerca nella tabella, che in corrispondenza all'ampiezza istantanea in ingresso (ascissa della tabella), trova il corrispondente valore di uscita . In fig. 8 è illustrata con un diagramma a blocchi tipo Music V la tecnica di sintesi; si è messo in evidenza che l'ampiezza del segnale di ingresso varia nel tempo. Questa è l'implementazione più semplice; il segnale in uscita può poi essere normalizzato moltiplicandolo per la funzione di normalizzazione G(I) dipendente dall'indice e può poi essere moltiplicato per un segnale costituente il numeratore. Infine il segnale risultante può essere moltiplicato per un inviluppo. Per evitare un generatore di segnale,in alcuni casi l'inviluppo può essere combinato con la funzione di normalizzazione; si può cioè modificare la forma dell'inviluppo in modo da tenere conto della normalizzazione

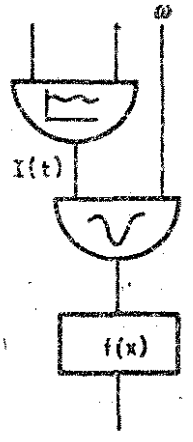


fig. 8

voluta.

Si osservi che il segnale generato con questa tecnica spesso ha una componente continua che dipende dall'indice e dal numeratore. Si può annullare questo effetto sostituendo alla funzione una funzione dell'indice di ampiezza pari al termine S_0. Questo procedimento non risulta conveniente in quanto si devono aggiungere ulteriori moduli. Si può attenuare questo effetto sottraendo alla funzione tabulata una costante direttamente al momento della generazione. Il valore di questa costante può essere scelto in varie maniere; esso può essere pari al valore medio di S_0 per quel segnale, che risulta però di difficile determinazione, oppure può essere uguale a 1/c in modo che S_0 sia nullo per I=0. Forse la scelta migliore è porre la costante uguale a $Y_m/2$ in modo che l'ampiezza massima e l'ampiezza minima siano circa uguali in modulo, consentendo così di sfruttare al meglio la dinamica offerta dalla rappresentazione numerica del segnale.

Normalmente le funzioni tabulate vengono normalizzate in ampiezza rispetto ad un certo valore massimo; esse sono cioè definite a meno di una costante moltiplicativa. Nella tabulazione di una funzione con due poli complessi coniugati, equivale a poter specificare solo due parametri indipendenti.

La funzione risulta quindi determinata specificando in un punto ρ e θ, oppure le coordinate di p, oppure b e c considerando a=1. In quest'ultimo caso si deve ricordare che deve essere $4ac-b^2>0$, da cui $c>\frac{b^2}{4}$.

Questa tecnica può essere implementata anche in maniera diversa e cioè eseguendo la divisione campione per campione. In questo caso la funzione viene scritta come

$$f(x) = \frac{1}{(ax+b)x+c}$$

oppure $f(x) = \frac{1}{(x+b)\cdot x+c}$ se $a = 1$

in cui per ogni x vengono effettuate due addizioni, una moltiplicazione ed una divisione. Questa implementazione risulta conveniente quando si vuole variare b e c in maniera diversa da quella derivante dalla variazione di ampiezza di x. Ad esempio se a e c restano costanti e varia solo b, si ha che il polo p descrive una circonferenza, essendo b/2a la parte reale del polo e c/a il quadrato del modulo. Analizzando nel piano p le intersezioni di questa circonferenza con il reticolo di ellissi ed iperboli a ρ e θ costanti si può determinare l'andamento dello spettro.

CONCLUSIONE

E' stata presentata una tecnica di sintesi non lineare mediante una trasformazione istantanea, descritta da una funzione con 2 poli complessi coniugati. Sono stati analizzati gli spettri che possono essere ottenuti al variare dei vari parametri e fornito il procedimento di calcolo. Sono inoltre state date delle indicazioni ed un criterio grafico per la scelta dei valori più appropriati per varie applicazioni.
Va qui evidenziato che la tecnica proposta possiede proprietà simili a quelle delle tecniche non lineari più diffuse, in particolare il fatto che la banda del segnale cresce al crescere di un certo indice e la possibilità di ottenere spettri armonici e inarmonici a seconda della scelta del rapporto fra due frequenze (in questo caso fra frequenza del numeratore e del denominatore). Si ha quindi la possibilità di ottenere tutti i suoni del tipo di quelli ottenibili con queste tecniche. Un'ulteriore proprietà significativa è che si possono ottenere in maniera semplice spettri con formanti ben caratterizzati, e questo risulta particolarmente utile nella produzione di molti suoni musicali.

Le formule che legano i vari parametri possono sembrare piuttosto complicate. Si spera che un'intensiva sperimentazione di questa tecnica da parte dei musicisti ed un esame dei grafici prodotti dai programmi di analisi scritti, consenta loro di raggiungere quella padronanza che la renda effettivamente un mezzo di espressione artistica.

Riferimenti

ARFIB D.: "Digital Synthesis of Complex Spectra by Means of Multiplication of Nonlinear Distorted Sine Waves", J. Audio Eng. Soc., vol. 27, n. 10, p. 757-768, oct. 1979.

CHOWNING J.M.: "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation", J. Audio Eng. Soc., vol. 21, n. 7, p. 526-534, sept. 1973.

LE BRUN M.: "Digital Waveshaping Synthesis", J. Audio Eng. Soc., vol. 27, n. 4, p. 250-265, apr. 1979.

LEHMAN R., BROWN F.: "Synthèse rapide des sons musicaux", Revue d'Acoustique n. 38, p. 211-215, oct. 1976.

MOORER J.A.: "The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulas", J. Audio Eng. Soc., vol. 24, p. 717-727, nov. 1976.

REINHARD P., ROPELATO F., VECELLI A.: "Distorsione non lineare tramite polinomi", relazione d'esame, Istituto di Elettrotecnica e di Elettronica, Università di Padova, 1979.

TRUAX B.: "Organizational Techniques for C:M Ratios in Frequency Modulation", Comp. Music Journ., vol. 1, n. 4, p. 39-45, nov. 1977.

Controllo in tempo reale di un sistema di trattamento/sintesi del suono

Real time control of a sound preocessing system.

Philippe PREVOT (1)

Riassunto

Il sistema qui presentato è stato concepito per pilotare la macchina 4C di Peppino di Giugno.

Fortemente orientato sull'interazione in tempo reale, e essenzialmente in constanta evoluzione, questo sistema già utilizzato in concerto e in numerose applicazioni, vuole essere un progresso verso uno strumento di tempo reale, e non soltanto un sintetizzatore di suoni. Puo essere conversazionale o eseguire un "programma musicale" e non soltanto una sequela di note. Permette un asincronismo totale dei parametri di uno strumento, tramite un grande insieme di "azioni" programmati, commandati da differenti tipi di eventi, e ha una filosofia specifica del tempo. Nessuna assegnazione a priori dei controlli gestuali.

Abstract

The system presented here has been designed and written to drive Peppino di Giugno's 4C machine.

Strongly oriented towards the real time control of the sound processor, and essentially in endless evolution, this system already used in several experiments and performances, intends to be a step towards a real time instrument, and not only a sound synthesizer. It can be conversational or execute a "musical program", and not only a sequence of notes. It allows a complete asynchronism between the parameters of an instrument, by a large set of programmed "actions", triggered by various types of events, and has an original philosophy of time. It does not imply any pre-assignation of the gestual controls.

---

(1) IRCAM, Parigi
31 rue Saint Merri
75004 Paris France

THE ARTS SYSTEM

INTRODUCTION :

This article presents the first "general purpose" software system designed to drive Peppino di Giugno's 4C machine (note 1). This system, called .ARTS. was first written (it was a SYN4B [Rolnick 78]) to drive a previous machine of di Giugno's. After the birth of the 4C , I could not keep from being attracted by the new lady, and transported the 4B software to the 4C machine, thus creating the present .ARTS. This article is not the user's manual for .ARTS. It is not an exhaustive description of how to use the system but a discussion about the fundamental choices.

.ARTS. is intended to be a "general purpose" system for the 4C machine. In fact, it is not. Just because today's mini or micro computers do not offer enough memory to build systems able to satisfy such a versatile sound processor as the 4C. At least, the aims of .ARTS. are two :

1) Give the musician the closest control as possible of the sound processor without asking him to speak octal or to deal with interrupt vectors. Indeed the 4C is a very versatile machine, and a software system must keep, as far as possible, from hiding this quality to the user.

2) Add a "software dimension" to the sound processor, so building a complete "sound processing system". It is absolutely essential to understand that a "real time sound processing system" is not only a "board of 160 integrated circuits ...", even with a genious organization, but the association of such a hardware with a host computer, a set of gestual control gadgets, and , not primarily but equally, a language and a software system. It is the whole set of all these elements which define á system and differenciate it from another one, even if the basic hardware is the same.

PROGRAM DESIGN CONSIDERATIONS

Very stongly oriented towards the real time sound control, .ARTS. which has already been used in public performances and in various applications, is designed on the basis of a few strong ideas :

- It is a conversational system

- It allows to run a "musical program" and not only a sequence of notes. This means that one can program the changes in instrumentation or in gestual environment, and consider these changes as being part of the score.

- It permits a complete asynchronism between the different parameters, and a high degree of reentrance, when using the lists. Each parameter of an instrument can change for an independent reason thus allowing a real time asynchronism, at the opposite of a traditional score in which all the parameters have their value pre-established and synchronized.

- It takes care not to mix up the internal and external times of the events. In simple terms, the duration of an envelope may be fixed by the envelope itself, or by any external event which shortens or enlarges it.

- It keeps from doing any pre-assignation to the gestual controls, thus leaving the user free to set his own environment.

- It puts its heart and soul into leaving as much free memory space as possible to the user, especially by a double level language, and by the ability for the user to generate the system with only the code and data space that he needs.

- It is divided into two sub-systems, of which the biggest is "synthesizer-independent". The language itself also contains a large "synthesizer-independent" part.

THE ENVIRONMENT OF .ARTS. :

.ARTS. is built around the following hardware elements :

- a 4C digital sound processor,
- a DEC LSI/11-03 host computer,
- 2 RK05 hard disks, of 2.5 Mega-bytes each, in a PLESSEY configuration,
- a DMA (direct memory access) on 16 bits,
- 1 ADC (analog to digital converter) 16 bits, designed by Tim Orr,
- 1 to 16 DACs (digital to analog converters) 16 bits, designed at IRCAM by Didier Roncin,
- a sophisticated controller for gestual inputs, based on a Motorola 68000, using 12 bit A to D converters, designed at IRCAM by Martin de Loye.
- an envelope follower, connected to the ADC and digitally to the DMA to control it.

This makes a very nice net, except one point : the host computer. An LSI-03 is fairly slow for this application, the floating point processor is unusable in real time, and a memory space of 28K 16 bit word is too little for such an application. In fact, the software system .ARTS. is able to run on any PDP-11 with at least 16K words.

The 4C sound processor is now a well known machine. It has been described in [Moorer J. 79]. One will find a brief resumé at the beginning of [Abbott C. 81].

HOW TO PLAY .ARTS. :

Instructions and data for .ARTS. are expressed in the source .ARTS. language. The source program may come from either the teletype or some file(s), previously written with your favorite editor, or made up through .ARTS. itself.

When you are in front of a complex sound processing system, there are a few ways to proceed, but not many.

1. If you don't know anything about the language, you ask a local expert to give you some typical and existing files as examples, and you discover the world through these files.

2. If you know about the language, but you don't know about what to do as sound processing, or how to do it, you just go for a walk around the sounds, trying successive things and receiving the response of the system.

3. If you are a wizzard, if you know what to do and how to do it, you store all of it in some file(s), and play it for pleasure. May be, you would still change things time to time.

4. If you play for a performance, with programs and scores definitely and lovely written, you want them to be processed in a fast, efficient and secure way.

What ever your way of working is, .ARTS. is good for you.

Before seeing why and how, let us give a few details on the instructions format : comments may be inserted, lower and upper case are equivallent (note 8), keywords are scanned up to 4 letters, the following ones being useful only for clearness, private symbol length is fixed at system generation time and must be a multiple of 3.

Now, let us take successively the four situations listed above, and see how .ARTS. can help you.

1. You don't know anything about the language. You are given some files as examples. When being given files to read, .ARTS. can receive switches :

P means "print the source file, in the same time than processing it".

S means "step by step" i.e. wait for a <CR> before executing next line from the file, and allow insertions from the teletype.

Q means "query" i.e. same as S but ask for confirmation for every line from the file before processing it.

... and several other switches ...

So you may be given a file called TEST1 that you call in this way : TEST1/S . Every time you hit <CR> with an empty line, you get the next line from TEST1 . If ever you type a non-empty line, it will be processed by .ARTS. The file TEST1 may look like this :

```
LOAD harm17 4K1                 ; loading a wave table called harm17 in the
                                ;    first 4K word table of the sound processor.
o1: OSC harm17 FA4#             ; an oscillator called o1 reading that wave table
                                ;      with a frequency corresponding to some note.
LISTEN o1  2 3                  ; listen to o1 on channels 2 and 3 .
o2: OSC harm17 HZ369.950        ; an other oscillator called o2 reading the
                                ;    same table at a very close frequency.
LISTEN o1  1 4                  ; listen to o2 on channels 1 and 4 .
LOAD wouw7 4K1                  ; replace the first wave by an other one.
```

Reading the first two instructions will not give you any apparent result, but when you hit <CR> and get the first LISTEN instruction, you hear the oscillator o1 . Then, when reading the next LISTEN , you hear the other oscillator. Finally, when reading the LOAD instruction, you hear the change of wave form. Between each line, you had time to read the comments, understand the effect of the instruction, and also insert some other instruction, like the MAP instruction which gives you the current content of the sound processor.

2. You know about the language, but you are looking for the sounds you want and how to make them. You may work this way :

Either you already have some files doing approximately what you want to do now, then you use them with the switches S or Q , and modify them in a conversational way from the teletype, or you really start from the very beginning, by typing everything from the teletype. Everything you type or you take from existing files, provided that it is error free, may be saved into an output file for wich you give a name. Once your output file is saved, you probably have to clean it up, since it must be full of trials and errors.

Here is an example. You want to study beats between different instruments. Since it is simple, you don't look for a file doing a similar thing, but you type the instructions and data in conversational mode :

```
f1: HZ123              ; f1 will be a real time variable containing 123 Hz
o1: OSC 4K1 f1         ; First define an oscillator and use f1 as frequency
LISTEN o1 1            ;   then listen to it on channel 1 . For the moment,
                       ;   you don't know what the table 4K1 contains.
LOAD arm23 4K1         ; now you know: it contains arm23 ... but you don't
LOAD arm237 4K1        ;   like it . Oh, how lovely this one is !
f1: HZ109              ;   even better at a lower frequency : you redefine f1
f2: HZ110              ; let us define an other oscillator on a different wave
o2: OSC 4K2 f2         ;   table with a slightely different frequency
LOAD arm347 4K2        ; load some nice wave form for that oscillator
LISTEN o2 2            ;   and listen to it on a separate channel
; at this point , o1 and o2 play at close frequencies, on two different wave
;   forms, respectively on channels 1 and 2. You want slower beats :
f2: HZ109.951          ; here it is
f1: HZ218              ; or emphasize the beats on some other harmonic
; and so on ... At some point, you want to save every thing :
CLOSE beat01           ; write and close a file containing all of that, with
                       ; the name  beat01
```

The file BEAT01 is now available for further processing. One can now cancel the useless lines, or modify the comments about the results.

3. Now you are a wizzard and are determined about what to do and how to do it.

You would probably use your beloved text editor, or re-arrange existing files, or even use some pre-processor. In a way or another, you would get some file(s) defining your instrumentation, data, environment definition, etc...

Assuming you built files called

E101 E102 EF304 containing envelope definition for instance.

FR1Z8 FR2Z8 containing some data (e.g. frequency lists)

INS203 VIOL04 containing different instrumentations.

you now build your complete instrument and score by associating the proper files together. The following arrangements will give :

```
-------------------------------------------------------------------------
E101,FR1Z8,INS203              ; same data (envelopes, frequencies, ...)
E102,FR1Z8,VIOL04              ;       on two different instruments

EF304,FR2Z8,INS203             ; same instrument
E102,FR1Z8,INS203              ;               using different data
-------------------------------------------------------------------------
```

PLEASE, DON'T UNDERSTAND THAT SPECIFIC FILES MUST HOLD SPECIFIC DATA !

If you prefer to deal with a huge mass in only one file, or various things in a long string of files, you may !

4. Finally, your programs and scores are ready and fine. There is no need any longer to change anything in the conversational mode. More, you would like to get a fast processing and a larger amount of free space in memory for scores and data.

.ARTS. is a two level language. The upper level, not very high, is the SOURCE level. Programs written at this level are scanned and translated into the lower level language, or "intermediate" language. This language is a binary, unreadable and barbare language, but it has the advantage of allowing a very fast processing. Programs translated in intermediate language are interpreted. The "scanner", which unhappily takes half the amount of space of the whole system, does not need any longer to be in memory.

Finally you save space and time.

HOW TO PROGRAM IN .ARTS. :

Now you know how to get access to .ARTS. , through files or via the teletype, in conversational or batch mode, at source level or in binary language. The following sections will describe the different aspects of .ARTS. as seen by the user.

It is essential to understand that .ARTS. is not made of sub-languages and that one does not need to separate the instructions in various files containing specific things. However, and only in order to simplify this article, I shall present the language in separated regions :

- Variables and data types in .ARTS.
- Defining lists or scores
- Defining envelopes
- Setting a patch in the sound processor
- Linking the data in the host computer to the sound processor

VARIABLES AND DATA TYPES IN .ARTS. :

Variables in .ARTS. may be any of the following :

Internal variables

System variables

Lists

Envelopes

Data in .ARTS. may be of the following types :

Constant data

Gestual inputs

Internal sample data

Lookup tables

Origin : Data may come either from the host computer, or from the sound processor, or from gestual gadgets. All data are considered after the analog to digital conversion.

Destination : Most of the data are sent to the sound processor, but some data may be given to the host computer to control it, or sent to the gestual instrumentation or to the teletype to display the state of the system.

Length : The 4C machine is a 24 bit word processor. Thus data are generally 24 bits large, except multipliers inputs, table lookup outputs, and time values, which are 16 bits large. When creating a data of which the destination is in the sound processor, .ARTS. can hardly know whether the actual destination is a 16 bit or a 24 bit word. At least, it would take more time to detect the actual data size than the time saved by not loading the 8 extra bits. Thus, the data are all 24 bits large, even when the 8 right most bits are unsignificant.

Constant data may be expressed with or without a unit, thus specifying a conversion scale. Units and scales are listed below :

UNIT	KEYWORD	SCALE	AMBITUS	COMMENTS
octal	↑	none	0 to 077777.377	15 bits . 8 bits signed value
thousands	without unit	linear	0 to 1000.xxx	3 decimal figures signed value
Hertz	HZ	linear	0 to 8000.xxx	3 decimal figures signed value
degrees	DEG	linear	0 to 360.xxx	3 decimal figures signed value
déciBels	DB	logarithmic stepping by 6 in linear segments	6 to 90	signed value
milli-sec.	MS	linear	1 to 65535	unsigned value
beats per minute	MET (metronom)	hyperbolic	60000 to 1	unsigned value

Gestual inputs are a more tricky affair. The way the MC68000 based controller works is the following: it receives, in DMA mode, data coming from an open end string of "boxes", in which are plugged about any kind of gadgets

responding to electrical constraints, and some generous limitation on tl total data volume on each box. This means that the user has to tell whi gadget (type, length and number) he plays with. This is told in a very simp language that I shall not present in this article. In the same time, the declarations are used to generate .ARTS. so that .ARTS. and tl controller think of having the same gestual environment.

Thus we assume that .ARTS. know the type, bit length and number inputs linked to the system. They have names, given by the user. Ea separate data may receive a separate name, or they can be grouped and th accessed by a generic name, indexed by the number within the group. F instance, assuming that we have 16 identical potentiometers, we can name th separately POTA POTB POTC ... POTP

or give them the generic name POT and call them

POT(1) POT(2) .... POT(16)

As an other example, assuming that we have a two dimension joy stick. We c separately name the horizontal and vertical dimensions

HSTICK and VSTICK

or call them as two elements of the same kind

STICK(1) STICK(2)

Now, how do the host computer see these inputs ? In other word which type and length do the data have ? Good question .

First about their type. They are of two fundamentaly differe types. The host computer sees the value corresponding to the prese position of the gadget. The length of this value is generally 1 bit or bits. The other aspect the host computer sees of the same input is interrupt when the input moves. In some cases, you would like to trigger event by any displacement of a potentiometer, without taking its value in account. Note that the same potentiometer can be read, in the same time, as value by some other process. The difference between the two types is made .ARTS. itself. The context is sufficient to determine wether you want value, in a patch or list definition, or you want an interrupt, in a comma instruction.

Then about their length. Since the sound processor expects 24 bit data, and the inputs do not deliver so large data, you have to specify some scaling algorithm. This may be done in two ways. The first way is to specify scaling factors A additive factor M multiplicative factor S arithmetic binary shift factor Assuming X is the real input value, the Y scaled value is :

$$Y = ((X * 2^{S}) * M) + A$$

The second way is to specify the lowest and highest values you want to reach. The input data is then scaled according to its length, in order to fit into these boundaries.

Example : POT(2) /HZ 25,,8/

means "shift the input by 8 to the left, then add 25 Hz to it"

where as POT(2) [HZ 25,HZ 1025]

means "scale the data from this pot in order to get values ranging from 25 Hz to 1025 Hz". A wizzard will see that both scalings are equivallent and will tell you to choose the first way which is a bit faster (note 5).

The gestual data may be catenated in order to get up to 24 bits large data. If you attempt to build a larger data, it results in a fatal error. If you specify scaling factors and catenation in such a way that it may result in an overflow for high values of the non scaled data, you are warned, at scanning time, and the scaled data are truncated (high order bits lost) at playing time.

Example : POT(2,3)

will give a 24 bit datum where the first pot occupy the left most 12 bits, and the second one occupy the 12 right most bits.

POT(6,7) /,,1/

which means "catenate the potentiometers 6 and 7, then multiply the value by 2 (shift 1 to the left)", will result in a warning, since the scaled data will overflow if the left pot goes further than the half of its range.

Internal and system variables receive data of any type.

Internal variables are the sound processor internal register: Their number is about 223. Some of them are assigned fixed values by .ARTS. like 0 , 1 (i.e. 2**23 -1) , -1 (i.e. -2**23) , epsilon (i.e. 1) , etc.. They can be given a symbolic name, and then referenced either by their nam or by their physical number prefixed by $. They are 24 bit words, holdi fixed point data. Here is a very simple example :

```
f1: POT(2,3)          ; f1 becomes a 4C register, containing the real ti
                      ;    value of two potentiometers catenated. Nobody kno
                      ;    its physical number, which depends on what has be
                      ;    setup previously, except if one types    MAP

f2: $123              ; f2 becomes the physical 4C register number 123
                      ;    which we assume we have some data.

$124:  MUL f1 f2      ; and here is the real time product of the pots by t
                      ;    real time contents of the register 123. The resu
                      ;    goes into the physical register number 124 .
```

Such internal variables either send their content to some unit in t 4C itself, through the patch that is currently setup, or are read by the ho computer to be displaid on the teletype, or for any other reason. Th receive data either from the 4C itself through the patch, or from the ho computer, or from the gestual inputs (note 2).

System variables are contained in a system table, of which t length is fixed at generation time. The initial contents of this tab represent the frequencies of an equally tempered scale, but the instructi SET allows the user to modify its content, so that it may finally contain a kind of data, without restriction on their physical significance. There a several ways of accessing this table. One is to use note names (lat denominations) with octave number and flat or sharp sign. Another way is use the keyword SYSTAB(n) where <n> is the number of the element. Y another way is to use the piano keyboard linked to the system via the gestu inputs controller. The variable thus created takes its value only at t

real time when it is loaded into the sound processor, so that if, between two uses, the contents of the table are modified, the actual data in the processor change (note 4).

Lists and envelopes are variables which generate data of any type. They are described in the following sections.

DEFINING LISTS :

A list is an ordered set of data or variables (with some limitations on variables, note 3). They receive a name and are used to input data into the internal variables, that is into the patch.

Lists may be of different types, according to their use. Lists intended to input data to the internal variables are "values lists". Lists intended to switch a table lookup unit onto different tables are "tables lists". Lists intended to switch an envelope generator unit onto different envelopes are "envelopes lists". Since it is impossible to mix the different uses, the scanner has to check the homogeneity of the list. Thus, the user must specify the type for the list currently defined (values list is the default type). Two other list types are not yet implemented : channel numbers lists and internal variables (patches) lists.

1. The most current type is the values list . The elements may be constant data, internal sample data, gestual input data, system variables. As said above, constant data may be expressed in different units. One can specify, at the beginning of the list, a default unit, which is applied only to those elements without specific unit. Also the list can accept scaling factors which are applied to all the constant data elements, when the list is scanned . The values in memory are then the scaled values. Each non constant data can accept specific scaling factors which are applied, in real time, to the value.

Example : LIST zoubeth HZ
123.21 234 345 D04# 456.654 567.765
678 POT(2)/HZ678,,8/ f1 f2 234.32 ?

This a list of 11 elements (note 6), in which some are expressed in Hertz (default unit), the 4th element - DO4# - is a system variable, the 8th element - POT(2) - scaled in real time according to the given scaling factors, the 9th and 10th elements - f1 and f2 - are internal variables previously defined, and the ? terminates the list.

2. Envelopes lists contain names of previously defined envelopes. They must be used by, and only by envelope generator units.

Example : Assuming e1 e2 e3 e4 e5 eg1 eg56 are defined envelopes,

```
LIST asdfg ENV                  ; ASDFG is an envelope type list
e1 e1 e1 e2 e3 e4 eg1 e5        ;       containing a sequence of
eg56 e5 eg1 e2 e2 e2 e1 ?       ;       different envelopes.
```

3. Lookup tables lists contain either previously defined names of lookup tables, or physical number of lookup tables. Lookup table names are defined when loading the tables. The name of the table is generally that of the file containing the table data.

```
LOAD sinus 4K4          ; loads the physical table number 4 of the 4K size
                        ;       from a file called sinus and the table is
                        ;       now called sinus

LOAD harm17 4K2         ; loads a table called harm17

LIST rtyu TABLE                 ; rtyu is a list of lookup tables
4K1 2K3 sinus 4K1 1K7 harm17 1K15 ?
```

DEFINING ENVELOPES :

The envelopes are defined as break points in the cartesian coordinates. The abscissa are expressed in unit increments, the time value of the unit being defined separately. The ordinates are expressed in an abstract scale whose maximum is 1000, with 3 decimal figures, in gestual inputs data. Segments may have a fixed time, i.e. their duration is fixed at

scanning time, and cannot be changed in real time. They may have a duration indexed on some real time data. In this case, these segments are defined with a * after the first ordinate of the segment. The actual value of the maximum abstract value, i.e. 100, is specified in the definition of the envelope. The envelope is reduced, at scanning time, according to this factor. The default specification is the maximum positive value of the 4C, that is

77777.377 in octal
90 deciBels
8000 Hertz
1000 in thousandth

The time value for one abscissa unit during the segments preceding the first "stared" segment, or between two non consecutive "stared" segment, and after the last "stared" segment, is expressed in milli-seconds, on the second line of the definition. There may be non "stared" segment, or there may be only "stared" segments. The time value of one abscissa unit during the "stared" segments is the current value of an "expansion factor". This factor is either specifically defined for each envelope generator, or defined at once for all the generators which do not have a specific factor. This kind of definition is versatile enough. One can easily change portions of an envelope, either in shape or in duration, without changing the complete definition. It is generally not important for micro dynamic of envelopes, but may be very useful in "large dynamic" envelopes.

Example :

```
ENV wouw DB 87          ; maximum value of this envelope is 87 dB
12 134 14 22            ; times in ms of the abscissa unit during
                        ; the successive "non stared" parts
0                       ; initial ordinate
10      600             ; up to 600 in 10 units (10*12 = 120 ms)
5       550
5       1000    *       ; segment between 1000 and 670 will have
10      670             ;   a duration depending on the real time
5       700             ;   value of the time expansion for the
10      750             ;   generator using this envelope.
5       650     *
10      1000    *
80      505             ; segment between 505 and 701 has a
40      701     *       ;   duration of 40 units * 14 ms = 560 ms
10      812
1       0       ?       ; end of the definition
```

SETTING A PATCH IN THE 4C :

This is of course an important chapter. It will not contain a complete description, since a user's manual exists. And before presenting .ARTS. in this topic, let me emphasize a few basic ideas.

Some digital sound processors may have dedicated and built in processing units, as frequency modulation oscillators, filters, reverberators, and so on. The 4C machine is far more versatile than that : it "only" offers atomic arithmetic and logic units. These are a "table lookup", a "3-input add", a "multiply and add", a "add and compare result to reference, then multiply and add", finally a "decrement and interrupt if negative". Did you see any "oscillator" or "ramp generator" in there ?

THERE IS NONE . Please erase everything you read about "the 4C oscillators".

It seems that you would not play music with that ! And yet, a control memory, that the user should not care about, and a data memory, allow all these atomic units to be patched together into molecules . This data memory is the set of "internal variables" seen in the chapter about "data and variables".

Now, what the software approach gives you, is the access to some "software built in" molecules , among which are the atoms themselves. I bet you will find some "oscillator" or "ramp generator" stereotypes, of course. An important choice in .ARTS. is to propose "built in molecules" which are not very big (note 7). As a matter of fact, it's no use to start in the "big moles race", since it has no end. There will allways be "some other user" expecting some "not yet built in molecule". Instead of giving a poor man a fish every day, it is better to give him a fish net. .ARTS. gives the user some molecules, and the "MACRO ability" to generate "private molecules" as big as the user likes. For instance there is no built in filter, but the user will easily find some file containing a filter macro definition, which fits with his needs.

The general syntax and the way to use the software built in molecules are :

[<output>:] <molecule keyword> <list of inputs>

The <list of inputs> structure depends on the molecule. In the case of a

lecule where a table is to be looked up, the first field is the table ecification. The next fields, in all cases, are arranged in such a way that e parameters which are most likely to be omitted, are in the last position, that if they are actually omitted, one does not need to specify an empty eld. The inputs may be any of the following :

Constant data
Gestual inputs
Internal variables
System variables
Envelopes (in the envelope field of a ramp generator)
Lists of elements fitting with the input nature (no lookup table for a one word input, nor channel number for an arithmetic input)

MACRO PATCH GENERATION :

When you want to reproduce a patch which is too complex to be just peated, it is possible to call it as a MACRO definition. Variables in the tch can be separated in two groups:

Local variables which do not need to be known outside this impementation the patch. These variables are defined as "local variables".
Global variables which are either defined or refered to from outside this plementation of the patch. They are called "global variables".

e way to use a MACRO generation is a little similar to Fortran function lls. You create a file in which a GLOBAL instruction specifies which mbols are global. The other symbols defined in the file are considered as cals, and are cancelled from the symbol table at the end of file ocessing. At the next call, these local symbols do not yet or any longer ist, so that there is no conflict. At every call, you specify the proper mbols or values to give to the global symbols, for this call. In this way, u setup as many different patches as you call the macro-definition.

Here is a list of the main built in molecules, in the present state of .ART

OPCODE	ARGUMENT	FUNCTION
OSC	<table spec.><input freq.1>[<input freq.2>][<phase>] then <phase output><-<phase input>[+<input 1>][+<input 2 <phase output> and <phase input> are the same 4C regist	Oscillator
MODIF	<table spec.><phase input>[<input 1>][<input 2>] table lookup then <phase output><-<phase input>[+<input 1>][+<input 2 <phase output> and <phase input> are two different 4C registe	Wave shaper
ADD	<input 1><input 2>[<input 3>]	Three input adder
RANDOM	nil	Noise generator
RETARD	<input>	One sample delay
MUL	<mul-input 1>[<mul-input 2>][<add-input>]	Multiply then add
MIXER	<input list>	n-entry mixer with implicit factor 1/n
PONDER	<list of pairs of <input><factor>>	n-entry mixer with explicit factors
GEN	@<envelope spec.>[<scaler input>][<offset input>]	Ramp generat with scaling and offs
INF	<input 1><input 2>[<offset in>]	Min (<in 1><in 2>) then add <offset>
SUP	<input 1><input 2>[<offset in>]	Max (<in 1><in 2>) then add <offset>
COMPAR	<input 1><input 2><creterium input>[<offset in>] INF if <creterium> positive, SUP if <creterium> negati	
LISTEN	<output spec.><channel spec.>	send <output> to <channel
AFFECT	<output list>	send the successive <output to channels 1 2 3 ..

DEFINING AND PLAYING SCORES

DISPATCHING THE DATA TO THE SOUND PROCESSO

Setting a patch in the sound processor is not only the way defining the different real time algorithms, but also to link the exter data and variables to the sound processor. When a patch definition links 4C register to a variable, that is to a list or an envelope or a sys variable, it is not yet a datum. We then need a way to pick up, at s specific moment, the present value of the variable and give it to register. This is done by means of the real time commands. Before using th

ommands, one has to setup their context, that is the relationship between he actions to do, the objects on which doing them, and the reason why or the eans to do them. This is done in the following general syntax :

<action><list of means><list of objects>

henever a <means> among the list occurs, the <action> is done on all the <objects> of the list. There are a large number of <actions> built in .ARTS. Unhappily, I did not yet succeed in writing .ARTS. so that the user an build his own new <actions> . The <means> can be any of these :

Teletype key (prefixed by !)
Gestual input data of the "interrupt" type
Pulser, giving programmed "ticks"
End of a list
End of an envelope

Objects> are among the following :

Lists
Envelopes
Pulsers
Messages
Internal variables

lere is a list of the main presently existing <actions> :

1.) Handling an envelope ;

ORCE Initiates an envelope. If the envelope is presently running, it is first forced to the first segment of the last non scaled part or to the last segment, then starts the envelope again.

NIT Same as FORCE but only if the envelope is already finished.

EPEAT Flag owned by an envelope process. When on, the envelope starts again as soon as it is finished. The effect of this command is to liken the internal time of the envelope to the external events time (works as a flip-flop).

TEP Starts a single segment, and stops at the end of the segment. Useful for large dynamic envelopes or to debug an envelope.

2.) Proceeding in a list :

PROCEED Increments the pointer to the next element in the list. If the e of the list is reached, points to the element before the firs which is a zero for values list, a null envelope for envelo lists, equal to the first element in a table list.

CYCLE As PROCEED but points to the first element when reaching the end.

BACK As CYCLE but backwards.

STRIDE Increments the pointer by N elements, the value of N being fix when generating the system. Useful to have a "coarse and fin exploration of a list, especially if the source input is edit with N elements per line.

FPOT As CYCLE but when reaching the end, stops on the last element.

BPOT As FPOT but backwards.

BEGIN (BURP) Points back to the first element (the element before the first

MARK Memorises the current position of a pointer.

RESUME Points back or forth onto the last MARKed element.

INDEX Feeds the <object> with the value of the system variable indexed the order number in the <list of means> of the one just activated

3.) Controlling the time generators : (called "PULSERs")

SCHEDULE Reverses the state of the time generator. If working, it stops. If stopped, it starts (works as a flip-flop).

START (STOP) Starts (stops) a time generator.

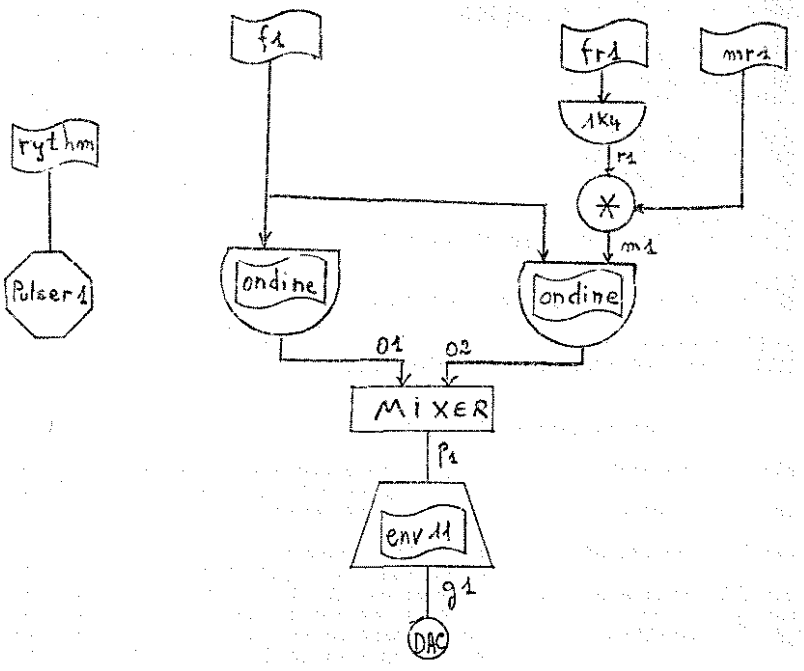
4.) Miscelaneous :

PRINT either prints a message, or displays the current pointer of process to a list.

DISPLAY displays the dynamic contents of a 4C register in a "machine lik octal format, with the 16 left and 8 right most bits separatel Very useful for debugging purpose (works as a flip-flop).

EXECUTE Runs the specified, user written routine. This is a mean to provi user written <actions> , but not wonderful since the routines ha to be written in assembler, and the interface with .ARTS. is t wizzard's domain.

```
Example :                ; first defining some data :

LIST f1 HZ                              ; list called  f1  in which the
DO RE MI FA SOL FA MI RE MI& ?          ;       default unit is Hertz.

LIST env11 ENV                          ; list of envelopes, called  env11
e1 e2 e1 e2 e3                          ;      envelopes are assumed to have
 e4 e1 e2 e5 ?                          ;      been previously defined

LIST ondine WAVE                        ; list of wave forms called  ondine
2K1 2K3 2K1 2K3 4K2                     ;     waves are assumed to have been
2K1 2K1 2K2 4K2 ?                       ;     previously loaded

LIST fr1 HZ                             ; list of values called  fr1 in hertz
1.10 100 1.00 70 0.01 80 90 950 0.01 ?  ;

LIST mr1                                ; list called  mr1  in thousands
0.1 0.5 0.1 0.7 8.5 0.1 0.2 0.1 10.5 ?

LIST rythm MS                           ; list called  rythm  in milliseconds
2200 1500 1100 1200 3000 1100 800 1000 ?


        ; then setting up the patch :

r1: OSC 1k4 @fr1                ; this oscillator reads the table 1K4,
                                ; previously loaded with a random sequence.
m1: MUL r1 @mr1                 ; this scales down the output of the table

o1: OSC @ondine @f1     ; oscillator reading the table currently pointed to
                        ; among the list  ondine at a frequency from the  f1
o2: OSC @ondine @f1 m1      ; same as  o1  but with a frequency offset of  m1
p1: MIX o1 o2               ; sums the two signals  o1  and  o2  divided by 2
g1: GEN @env11 p1       ; ramp generator using the envelope currently pointed
                        ; to in the list  env11 and multiplying the signal  p1

LISTEN g1               ; send  g1  to all the channels

PULSER 1 @rythm         ; defines a "time generator" numbered 1 onto the list

        ; finally setting up the real time commands environment :

SCHEDULE !x PULSER1     ; start the PULSER 1 when the key  x  is hit if it is
                        ; stopped, stop it if it is running
STOP  f1 PULSER1        ; stop it at the end of the list  f1

PROCEED PULSER1 @f1 @fr1 @mr1 @env11 @ondine          ; every time the PULSER
                        ; gives its "tick", go to the next element in the
                        ; lists of frequency, envelope, wave, etc..

FORCE PULSER @env11     ; also start the current envelope even if already on

REPEAT !y @env11        ; if y is hit a first time, and PULSER 1 is stopped,
                        ; restart the envelope as soon as it is finished.
REPEAT f1 @env11        ; and stop the repetition at the end of the list  f1

PROCEED @env11   @f1 @fr1 @mr1 @env11 @ondine     ; do the same as the PULSER
                        ; every time the envelope pinted to in the list  env11
                        ; is finished.

PROCEED !f @f1          ; same action than the PULSER but independently
FORCE !f @env11         ; for each parameter of the sound and "by hand"
PROCEED !r @fr1         ; using the teletype as a gestual gadget !
PROCEED !m @mr1
PROCEED !w @ondine
PROCEED !e @env11
```



Explanations : When the key x is hit a first time, the PULSER starts. since it is defined on the list ruthm , it takes its successive values from that list, and delivers a "tick" at the specified intervals of time. At every tick, it starts the envelope generator, first forcing it to decay if it is still running. In the same time, it makes all the lists pointing to the next element. In this case, all the parameters are synchronous. On the opposite, if one uses the teletype commands (f m r w e) all the parameters are asynchronous and independent. They can go backwards, or be marked and jump, if one uses other "actions". It is important to notice the difference between using the PULSER and the command REPEAT. When the PULSER runs, the time between two notes is decided by the PULSER through its list ruthm . When the PULSER is stopped and the REPEAT function is on, this time is the duration of the current envelope in the list env11 . This is the main difference between the internal and the external time.

DEBUGGING PATCHES :

When you setup a connection in the sound processor, you do not allways get what you were expecting. .ARTS. gives you a few ways to debug the patch.

First, you will probably need to know, what the complete connection in the sound processor looks like, after several source inputs from files or from teletype or by macro generation. The MAP instruction will give you the complete connection in source language. It is a "disassembling" of the sound processor control memory. This map gives you more than what you entered. It also gives you the physical number of the registers and cycles used to setup the connection. This is not only useful to debug the patches, but also to optimize them. One can see the free "hidden" resources, like adders inside a MUL-ADD, or multipliers inside a logic unit.

Further, .ARTS. being a conversational system, one can allways modify the connection without modifying the source files. Thus one can send to any DAC such or such internal datum, either to listen to it or to send it to an oscilloscope. This has still the inconvenient to give an analog value. Then the instruction DISPLAY allows to know the dynamic digital value of any internal datum. For instance, since the multiplication has 16 bits input, if one of the inputs has only a variation among its 8 right most bits, the result will not depend on that input. The DISPLAY command will give an immediate diagnostic.

CONCLUSIONS AND FORESEEING :

As said at the beginning, this article is not a user's manual.

Although .ARTS. offers some nice and original features, as the conversational mode, the asynchronism between the different parameters of a score, through a set of independently controled lists, the real time control of data and variables, the file manipulation, a double level language, we are still very far from a highly musical software.

Such a software should be strictly respectful to the sound processors, which are more and more versatile. The more complex the sound processor is, the more difficult it is to design a system, both respectful to that processor and doing more than just reading notes.

It should also eliminate as much as possible the alpha-numeric access to the system, although such an access is as versatile as boring. This assumes a very well designed input device, with which the input language can become a gestual instead of a lexical language.

Such a software should include a general powerfull language as C or PASCAL. This assumes a very large memory space.

Such a system should offer a smart graphic output. Of course, this is nice for wave forms, spectra or envelopes, and already exists in several systems. What a future system should offer is a logic graphic output, disassembling the patches in the universal language of the drawings. As a matter of fact, among a large number of musical softwares, either as software processors or as handling hardware processors, and as long as we consider "unit generator" structured processors (unlike MUSIC 10 for instance), although the basic resources are allways about the same, the syntax is allways different, while the drawings are very easily understood by any body, as well in the syntactic as in the semantic point of vew.

At IRCAM, we are now developing, beside P. di Giugno's powerful 4X, a smaller but integrated system, based on the fundamental element of the 4X , the 4U, a "generalized 4C", with a large memory processor, optional peripherals, as disks, graphic tablet and console, and mainly a very well designed gestual input console, with its own control processor.

NOTES :

te 1 : A more recent and very interesting system has been designed and written at IRCAM by Curtis Abbott [Abbott C. 81], for the same sound processor, but in a slightely different hardware environment.

te 2 : The transfer from the gestual inputs to the sound processor is presently made via the host computer. A next version will allow a direct access from the gestual input controller to the sound processor.

te 3 : It is not difficult, when not in real time and without memory limitations, to handle lists in which the elements themselves may be lists. In real time, with drastic memory limitations and without real time overlay ...

te 4 : The interest of such a system table is not very clear to me, and a new version will probably suppress it, and replace it by a more powerfull indexed access to the lists which can be of any length.

te 5 : With a basic sampling rate of 16 K Hertz in the 4C, 2**10 is very near 1 Hertz. If a 12 bit data is shifted by 8 bits to the left, it occupies 20 bits, that is 10 bits to the left of 1 Hertz, thus reaching a maximum value of 1000 Hertz. Added to an offset of 25 Hertz, the scaled value will range from 25 Hertz to 1025 Hertz, stepping by 0.25 Hertz. The reason why the first scaling (explicit scaling factors) is faster than the second (boundaries) is that, when computing the scaling factors for the second scaling, .ARTS. will evaluate a multiplicative factor, which will probably be very near a power of 2, while the second scaling will only use a shift operation.

te 6 : The maximum number of elements in a list is defined at the system generation time, a typical value being 400, the maximum maximorum being more than 32000. In the course of this article, the examples are very short, only to be more clear.

te 7 : Some built in molecules may be very big because they are recursive (e.g. MIXER or PONDER). Indeed, the size of a molecule must not be measured in terms of sound processing resources,but in terms of how much code is necessary to scan and interpret the patch parameters.

te 8 : In the course of this article, keywords are written in upper case, and private symbols in lower case, this only to make things more clear. <CR> means "carriage return".

REFERENCES :

3OTT C. 1981. "The 4CED program" in Computer Music Journal V.1 : 13-33

)RER J. et al. 1979. "The 4C machine" in Computer Music Journal III.3:16-24

.NICK N. 1978. "A Composer's notes on the development and implementation of software for a digital synthesizer" in Computer Music Journal II.2:13-22

Distorsione Non Lineare della somma di due cosinusoidi : analisi dello spettro tramite matrici

NON LINEAR DISTORTION OF THE SUM OF TWO COSINE WAVES ANALYSIS OF THE SPECTRUM USING MATRICES

P. REINHARD

CSC - Università di Padova, Facoltà di Ingegneria

ABSTRACT

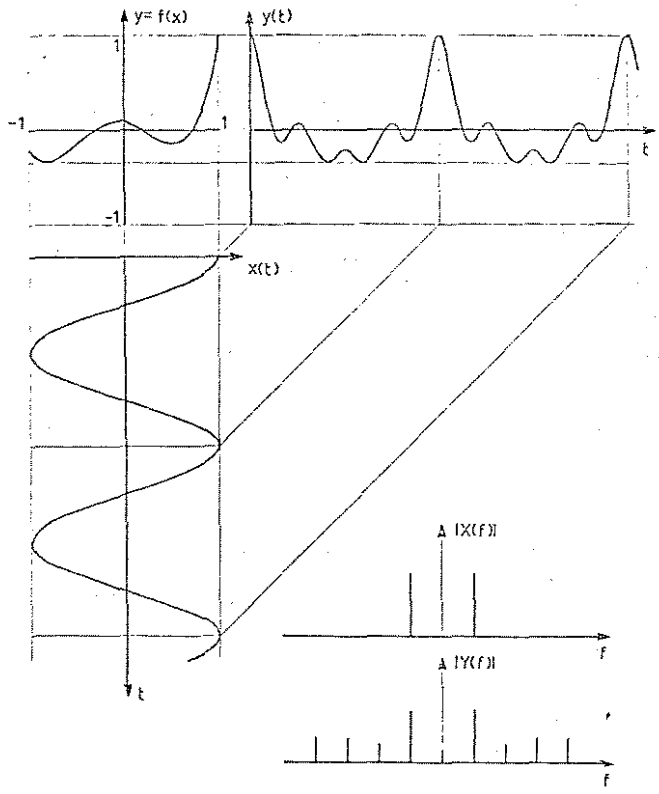
Questo scritto propone una completa descrizione del distorsione di una cosinusoide tramite polinomi usando u notazione matriciale molto compatta. Inoltre estende risultati e le notazioni al caso della distorsione del somma di due coseni di differenti frequenze.

This paper, in addition to a complete description of t distortion of a cosine wave with polynomial function proposes, using a compact matrix notation, an extension these results to the analysis of the complex spectr resulting from the distortion of the sum of two cosines different frequencies.

INTRODUZIONE

Già molti articoli apparsi su riviste specializzate hanno
:attato la sintesi per distorsione non lineare, sottolineando
ı un lato la grande generalità di questa tecnica, ed esponen-
)ne dall'altro particolarizzazioni sia teoriche che pratiche.

Ricordiamo brevemente di cosa si tratta: nel caso più sem-
:ice si produce un segnale cosinusoidale $x(t)=\cos(2\pi Ft)$, lo si
)ltiplica per un "indice di distorsione" (o "di modulazione")
e lo si usa come argomento per una funzione non lineare con
.nua e istantanea f(.) che restituisce un nuovo segnale y(t),
t)=f(Ix(t)), periodico dello stesso periodo del segnale di
ırtenza, ma di "forma" diversa (di qui il nome inglese di que
:a operazione: waveshaping, cioè "formatura d'onda").



ESEMPIO DI DISTORSIONE NON LINEARE

Assieme all'alterazione della forma, la distorsione comp
ta naturalmente un arricchimento nello spettro del segnale,
pendente staticamente dalla natura della funzione distorcen
e dinamicamente dall'andamento dell'indice che inviluppa il
gnale da distorcere. Tradotto in termini acustici, que
significa in sostanza che la distorsione agisce sul timbro
suoni, "generando" armoniche in modo controllabile mediante
indice di distorsione.

Nella pratica, i modi in cui è realizzata la distorsi
non lineare sono due: con dispositivi analogici, oppure in
calcolatore digitale. In quest'ultimo caso tutti i segn
sono quantizzati sia nel tempo che nelle ampiezze, ed anche
funzione distorcente è quasi sempre nient'altro che una tab
zione memorizzata e "consultata" dal programma di generazio
dei campioni sonori (tecniche di table look-up).

Tra le tecniche digitali di sintesi, le tecniche di dis
sione sono molto popolari e competitive rispetto ad altri m
di, come la sintesi additiva o il filtraggio, soprattutto
l'efficienza accompagnata dalla possibilità di controllare
modo soddisfacente, e agendo su pochissimi parametri, la din
ca dello spettro dei suoni prodotti.

In questo scritto si parlerà di distorsione con funzi
polinomiali, un tipo di distorsione la cui caratteristica p
cipale è la semplicità matematica.

In particolare, agevolati dalla concisione di formali
che questa semplicità consente, affronteremo quantitativame
il problema dell'analisi e sintesi di suoni multifonici pro
ti distorcendo con un polinomio una somma di due cosinuso
di frequenze diverse.

Come è noto, i suoni multifonici - che per altro non s
una novità della Computer Music, ma possono essere prodotti

che con strumenti tradizionali suonati in modo non ortodosso - sono suoni in generale non-armonici il cui spettro ha tuttavia una struttura che è in un certo senso una generalizzazione di quella degli spettri armonici. Più esplicitamente, un suono armonico ha una frequenza fondamentale F e componenti spettrali a frequenze nF, con $n=0,\pm1,\pm2,\ldots$, mentre un suono multifonico ha due frequenze fondamentali F_1 ed F_2 , e presenta righe spettrali alle frequenze $mF_1+ nF_2$, per le varie combinazioni di $m,n=0,\pm1,\pm2,\ldots$.

Nella notazione adottata, quella del calcolo matriciale, sarà interessante vedere come questa intuitiva "duplice armonicità" dei suoni multifonici si sposi proprio ad un aumento di dimensione (dai vettori alle matrici) nel calcolo e nella rappresentazione degli spettri.

2 POSIZIONE DEI PROBLEMI

La funzione distorcente f sia d'ora in poi un polinomio di grado N e coefficienti $d_0,d_1,\ldots,d_N$

$$f(x) = \sum_{n=0}^{N} d_n x^n \qquad (2.1)$$

Per dare progressività all'esposizione tratteremo due coppie di problemi di complessità crescente, ma con forti simmetrie:

1)distorsione di una sola cosinusoide

Posto $x=\cos(2\pi Ft)$, si dimostra che il "coseno distorto" $y(t)=f(Ix(t))$ (che è un segnale periodico e pari perchè funzione "tempo-invariante", o perlomeno considerabile tale, e istantanea di un segnale periodico pari), ammette uno sviluppo (unico) come somma di un numero finito, e uguale ad N, di coseni. Sono allora lecite le equazioni con le quali formalizziamo i seguenti sottoproblemi, differenti fra loro solo per l'assenza o presenza dell'indice di distorsione.

1.1)senza indice di distorsione: nella Eq.

$$f(x)=\sum_{n=0}^{N} d_n(\cos(2\pi Ft))^n = \sum_{n=0}^{N} h_n\cos(2\pi nFt) \qquad (2.2$$

determinare gli h_n, cioè le ampiezze delle armoniche del suono prodotto, assegnati i coefficienti del polinomio (problema di analisi della distorsione) e, dualmente, determinare i d_n, dati gli h_n (problema della sintesi identificazione del polinomio distorcente)

1.2)con indice di distorsione: nelle Eq.

$$f(Ix)=\sum_{n=0}^{N} d_n(I\cos(2\pi Ft))^n = \sum_{n=0}^{N} h_n(I)\cos(2\pi nFt) \qquad (2.3$$

$$f(Ix)=\sum_{n=0}^{N} h_n\cos(2\pi nFt) = \sum_{n=0}^{N} d_n(I)(I\cos(2\pi Ft))^n \qquad (2.3$$

determinare rispettivamente gli $h_n(I)$, assegnati i d_n il valore dell'indice I (analisi) e, viceversa, i $d_n(I)$ dati gli h_n e I (sintesi).

)distorsione della somma di due cosinusoidi

Poniamo $x_1=\cos(2\pi F_1 t)$ e $x_2=\cos(2\pi F_2 t)$. Anche qui sono giustificabili gli sviluppi in somme di coseni scritti.

2.1) senza indice di distorsione: nella Eq.

$$f(x_1+x_2)=\sum_{n=0}^{N} d_n (x_1+x_2)^n = \sum_{m,n=0}^{N} H_{m,n} \frac{1}{2}\left[\cos(2\pi(mF_1-nF_2)t)+\cos(2\pi(mF_1+nF_2)t)\right] \qquad (2.4)$$

calcolare, in analogia al caso 1.1, le ampiezze delle righe spettrali $H_{m,n}$ dai coefficienti d_n, e viceversa (in questa seconda coppia di problemi, la sintesi non è più perfettamente duale all'analisi, perchè non è possibile assegnare completamente ad arbitrio gli spettri)

2.2) con indici di distorsione: in analogia al caso 1.2, considerare la presenza di due indici I_1 e I_2 in $f(I_1 x_1+I_2 x_2)$, e quindi risolvere un'equazione del tipo della (2.4), una volta per calcolare $H_{m,n}(I_1,I_2)$ da d_n, I_1 e I_2, ed un'altra per calcolare (se possibile) $d_n(I_1,I_2)$ da $H_{m,n},I_1,I_2$.

Come si può vedere dall'aspetto delle formule, la soluzione dei problemi richiede sostanzialmente la manipolazione di espressioni trigonometriche e di sommatorie.

Tutto ciò che si cercherà di fare nel seguito è alleggerire, e rendere il più possibile "indolori" tali manipolazioni. L'uso diretto di formule trigonometriche per i passaggi da potenze di coseni a coseni di angoli multipli, e viceversa, sarà evitato ricorrendo ai polinomi di Chebyshev, mentre le sommatorie saranno scritte come "forme" di opportuni vettori.

I prossimi due paragrafi contengono appunto i preliminari alla attuazione di questa strategia.

3 I POLINOMI di CHEBYSHEV

I polinomi di Chebyshev sono una famiglia di funzioni la cui proprietà più interessante per i nostri scopi è quella di essere tali che, se hanno per argomento una cosinusoide, danno come valore un'altra cosinusoide di angolo multiplo.

Definizione: l' n-esimo (n=0,1,2,...) polinomio di Chebyshev (di prima specie) si definisce come

$$T_n(x) = \cos(n.\arccos(x)) \qquad (3.1$$

Alcune proprietà

1) $$T_n(\cos\vartheta) = \cos(n\vartheta) \qquad ,\forall\vartheta \qquad (3.2)$$

che è la proprietà preannunciata

2) $$T_n(x) = \frac{(x + \sqrt{x^2-1}\,)^n + (x - \sqrt{x^2-1}\,)^n}{2} = \sum_{m=0}^{N} B_{m,n}x^m \qquad (3.3$$

mostra in modo esplicito la natura polinomiale di $T_n(x)$ (un espressione dei coefficienti $B_{m,n}$ sarà data nel paragrafo s guente): $T_n(x)$ è un polinomio di grado n

3) $$T_{n+1}(x) = 2x.T_n(x) - T_{n-1}(x) \quad , n=1,2,... \qquad (3.4$$

è la formula di ricorrenza per i polinomi di Chebyshev: c i "valori iniziali"

$T_0(x) = \cos(0) = 1$

$T_1(x) = \cos(\arccos x) = x$

è sufficiente, e comunemente usata, per costruire in sequen za i polinomi di Chebyshev

Es:

$T_2(x) = 2x.x-1 = 2x^2 - 1$

$T_3(x) = 2x(2x^2-1)-x = 4x^3 - 3x$

$T_4(x) = 8x^4 - 8x^2 + 1$

$T_5(x) = 16x^5 - 20x^3 + 5x$

I VETTORI p E T , LE MATRICI A E B

Data la generica variabile x, indichiamo con $\mathbf{p}(x)$ il ettore delle potenze di x"

$$(x)=\begin{bmatrix} x^0 \\ x^1 \\ x^2 \\ \vdots \\ x^N \end{bmatrix} \qquad (4.1)$$

ta: d'ora in poi, quando parleremo di vettori e di matrici , tenderemo sempre che gli indici varino tra 0 ed N, cioè che vettori appartengano ad R^{N+1}, e le matrici a $R^{(N+1)x(N+1)}$.

Allora, dato un qualsiasi polinomio di grado non superiore N, $g(x)= \sum_{n=0}^{N} c_n x^n$, ordinandone in un vettore **c** i coefficien , potremo scrivere

$$x) = \mathbf{c}^T \mathbf{p}(x) \quad , \qquad \mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} \qquad (4.2)$$

Ogni polinomio può essere scritto in questo modo, ed in rticolare i polinomi di Chebyshev. Anzi, poichè come le tenze essi sono una famiglia di funzioni, è anche qui più op rtuno cercare un'espressione complessiva piuttosto che espres oni individuali.

Definiamo allora il vettore $\mathbf{T}(x)$ come il "vettore dei po nomi di Chebyshev in x"

$$(x)=\begin{bmatrix} T_0(x) \\ T_1(x) \\ T_2(x) \\ \vdots \\ T_N(x) \end{bmatrix} \qquad (4.3)$$

Raggruppiamo poi come colonne di una matrice B i "vettori dei coefficienti" dei polinomi di Chebyshev che compongono il vettore T (la matrice B abbia cioè nel posto m,n il coefficiente della m-esima potenza nell'n-esimo polinomio di Chebyshev: evidentemente è una matrice quadrata).

In questo modo, per il singolo polinomio di Cheb. si può scrivere:

$$T_n(x) = B^T_{.,n}\, p(x) \qquad (4.4)$$

(ove $B_{.,n}$ indica la n-esima colonna di B),
e per l'intero vettore $T(x)$:

$$T(x) = B^T\, p(x) \qquad (4.5)$$

Supponendo che B sia (come è) invertibile, e chiamando A la sua inversa, potremo scrivere anche:

$$p(x) = A^T\, T(x)\ , \qquad A = B^{-1} \qquad (4.6)$$

Queste uguaglianze sono molto importanti e saranno estesamente usate nel seguito perchè in esse è compendiato il passaggio da potenze di coseni a coseni di angoli multipli (nella 4.5) e viceversa (nella 4.6).

Affinchè siano utilizzabili praticamente, descriviamo ora matrici A e B.

Le formule che danno le espressioni "assolute" degli elenti di queste matrici sono abbastanza complicate, ma possono sere vantaggiosamente sostituite da semplici relazioni ricorve tra gli elementi stessi.

Prima di riportare le formule ed una tabulazione (per N=9), cciamo alcune osservazioni:

A e B sono triangolari superiori, cioè hanno tutti gli elementi al di sotto della diagonale principale nulli
sono matrici "a scacchiera", cioè con nulli tutti gli elementi per i quali la somma (o la differenza) degli indici è dispari
la somma degli elementi di ciascuna colonna è 1
nella costruzione delle matrici con le formule ricorsive, è conveniente procedere per righe nella matrice B, e per diagonali nella matrice A

MATRICE A

$$\begin{array}{cccccccccc}
1 & 0 & \frac{1}{2} & 0 & \frac{3}{8} & 0 & \frac{10}{32} & 0 & \frac{35}{128} & 0 \\
0 & 1 & 0 & \frac{3}{4} & 0 & \frac{10}{16} & 0 & \frac{35}{64} & 0 & \frac{126}{256} \\
0 & 0 & \frac{1}{2} & 0 & \frac{4}{8} & 0 & \frac{15}{32} & 0 & \frac{56}{128} & 0 \\
0 & 0 & 0 & \frac{1}{4} & 0 & \frac{5}{16} & 0 & \frac{21}{64} & 0 & \frac{84}{256} \\
0 & 0 & 0 & 0 & \frac{1}{8} & 0 & \frac{6}{32} & 0 & \frac{28}{128} & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{16} & 0 & \frac{7}{64} & 0 & \frac{36}{256} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{32} & 0 & \frac{8}{128} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{64} & 0 & \frac{9}{256} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{128} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{256} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}$$

Espressione per il generico elemento:

$$A_{m,n} = \begin{cases} \frac{1}{2^n}\binom{n}{n/2} & \text{per } m=0,\ n \text{ pari} \\ \frac{1}{2^{n-1}}\binom{n}{(n-m)/2} & 1 \le m \le n,\ (n-m) \text{ pari} \\ 0 & \text{altrove} \end{cases}$$

Formula ricorsiva:

$$A_{m,n} = \begin{cases} \frac{1}{2}A_{m+1,n-1} & \text{per } m=0,\ n \ge 1 \\ A_{m-1,n-1} + \frac{1}{2}A_{m+1,n-1} & \text{per } m=1,\ n \ge 1 \\ \frac{1}{2}\left(A_{m-1,n-1} + A_{m+1,n-1}\right) & \text{per } m \ge 2,\ n \ge 1 \end{cases}$$

Valori iniziali:

$$A_{m,0} = \begin{cases} 1 & \text{per } m=0 \\ 0 & \text{altrove} \end{cases}$$

MATRICE B

1	0	-1	0	1	0	-1	0	1	0	...
0	1	0	-3	0	5	0	-7	0	9	...
0	0	2	0	-8	0	18	0	-32	0	...
0	0	0	4	0	-20	0	56	0	-120	...
0	0	0	0	8	0	-48	0	160	0	...
0	0	0	0	0	16	0	-112	0	432	...
0	0	0	0	0	0	32	0	-256	0	...
0	0	0	0	0	0	0	64	0	-576	...
0	0	0	0	0	0	0	0	128	0	...
0	0	0	0	0	0	0	0	0	256	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Espressione per il generico elemento:

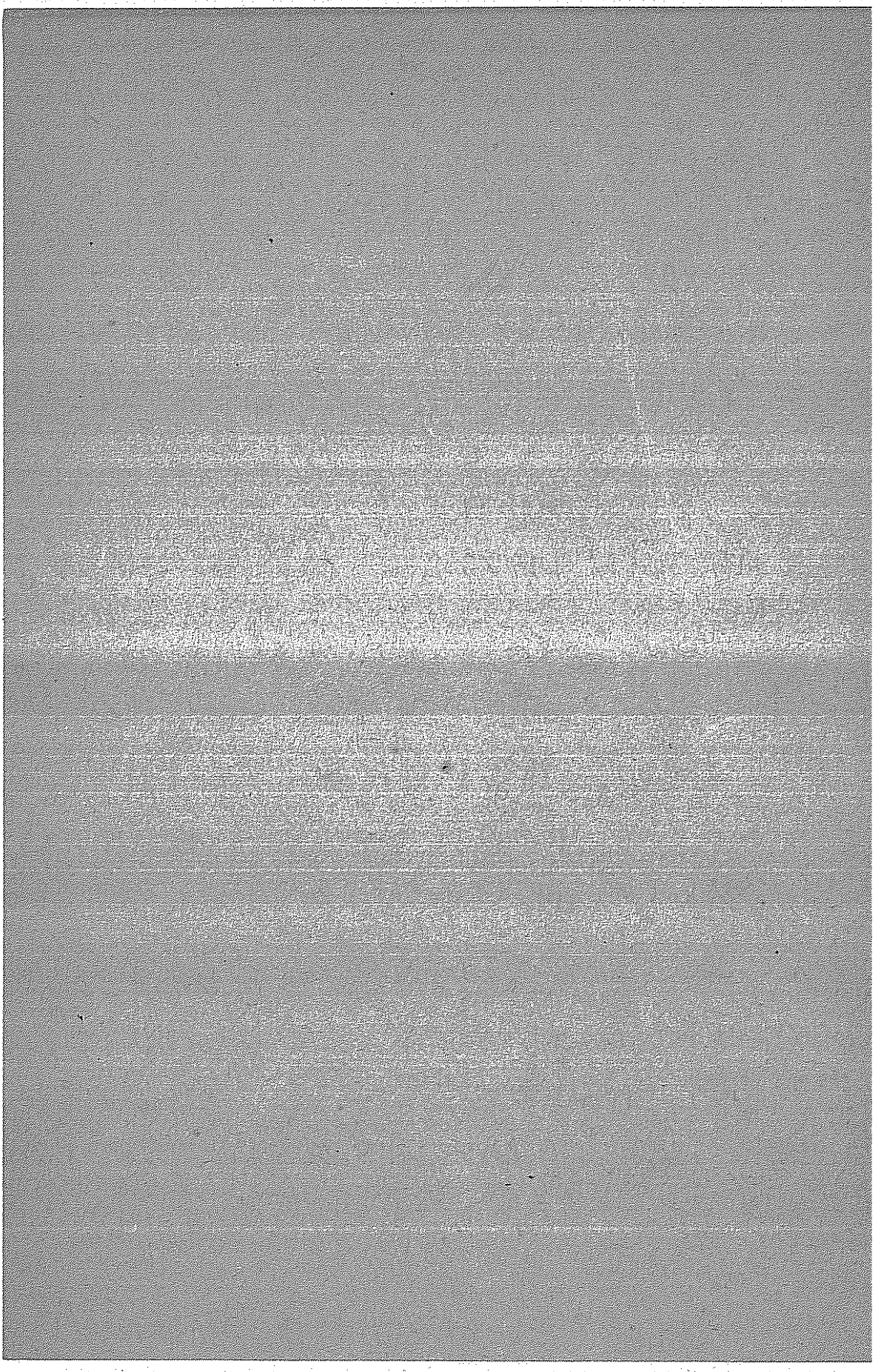
$$B_{m,n} = \begin{cases} (-1)^{\frac{n-m}{2}} \sum_{\substack{k=n-m \\ k \text{ pari}}}^{n} \binom{n}{k} \binom{k/2}{\frac{k-(n-m)}{2}} & \text{per } m \le n,\ (n-m) \text{ pari} \\ 0 & \text{altrove} \end{cases}$$

Formula ricorsiva:

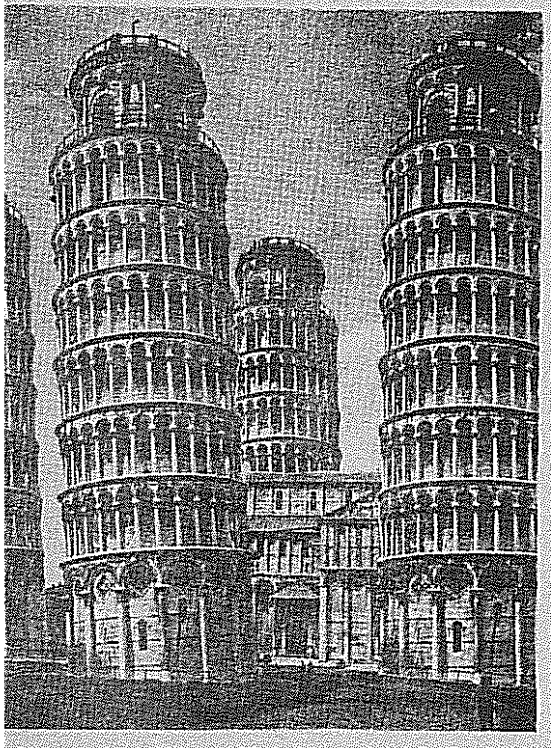
$$B_{m,n} = 2\,B_{m-1,n-1} - B_{m,n-2} \qquad \text{per } m \ge 1,\ n \ge 2$$

Valori iniziali:

$$B_{0,n} = \cos\left(\frac{\pi n}{2}\right), \qquad B_{m,0} = \begin{cases} 1 & \text{per } m=0 \\ 0 & \text{altrove} \end{cases}, \qquad B_{m,1} = \begin{cases} 1 & \text{per } m=1 \\ 0 & \text{altrove} \end{cases}$$



CNUCE ISTITUTO DEL CONSIGLIO NAZIONALE DELLE RICERCHE



QUARTO COLLOQUIO DI INFORMATICA MUSICALE

1981

atti vol. II

5 DISTORSIONE DI UNA COSINUSOIDE

Abbiamo già visto che, se chiamiamo d il vettore dei coefficienti del polinomio f, è possibile scrivere:

$$f(x) = d^T p(x) \quad , \qquad d = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \tag{5.1}$$

Fatto questo richiamo, iniziamo con il caso della distorsione senza indice di modulazione.

Nella notazione "algebrica" che ci siamo procurati con i polinomi di Chebyshev, con la scrittura compatta del polinomio distorcente, e "dimenticando" che $x=\cos(2\pi Ft)$, il problema dell'analisi della distorsione assume la forma seguente:
fissato d , trovare il vettore h tale che

$$f(x) = d^T p(x) = h^T T(x) \qquad \forall x \tag{5.2}$$

La soluzione è banale e si consegue semplicemente sostituendo $p(x)=A^T T(x)$

$$f(x) = d^T p(x) = d^T A^T T(x) = (Ad)^T T(x) \tag{5.3}$$

h è unico, e dunque

$$h = Ad \tag{5.4}$$

Il problema della sintesi è duale:

fissato h, trovare d tale che

$$f(x) = h^T T(x) = d^T p(x) \qquad \forall x \qquad (5.5)$$

anche qui la soluzione è immediata con la sostituzione in f(x) di $T(x)=B^T p(x)$

$$f(x) = h^T T(x) = h^T B^T p(x) = (Bh)^T p(x) \qquad (5.6)$$

e dunque

$$d=Bh \qquad (5.7)$$

Riassumendo, la trasformazione lineare h=Ad permette di conoscere il contenuto armonico di un coseno distorto con un polinomio assegnato, mentre la d=Bh permette di determinare quali coefficienti un polinomio deve avere per produrre, distorcendo un coseno, un suono con contenuto armonico prefissato.

Alcune osservazioni che è opportuno fare sono le seguenti:

- Il segnale distorto è rigorosamente limitato in banda, cioé non ha componenti spettrali oltre la frequenza NF, dove N è il grado del polinomio (è quindi agevole "tenere a bada" il fold-over), e viceversa, per ottenere un suono con N armoniche si deve usare un polinomio di grado N (infatti A e B sono quadrate e triangolari superiori)
- nelle relazioni h=Ad e d=Bh, data la struttura a scacchiera delle matrici A e B, solo le componenti di h e di d aventi

lo stesso tipo di parità sono reciprocamente vincolate

- lo sviluppo armonico più elegante per il segnale distorto è sì una somma di coseni, ma bilatera:

$$f(\cos(2\pi Ft)) = \sum_{n=-N}^{N} k_n \cos(2\pi nFt) \qquad (5.8)$$

ove i k_n sono pari rispetto ad n:

$$k_{-n} = k_n \qquad (5.9)$$

E' allora immediato passare dagli h che sappiamo calcolare ai k:

$$k_n = \begin{cases} h_n & n=0 \\ \frac{1}{2} h_{|n|} & n \neq 0 \end{cases} \qquad (5.10)$$

Passiamo ora al caso della <u>distorsione con indice di modulazione</u>. Dobbiamo studiare

$$?(Ix) = d^T p(Ix) \qquad (5.11)$$

love p(Ix), concordemente alla definizione data in precedenza, è il vettore che ha nella n-esima posizione la n-esima potenza li Ix.

Se, data la generica variabile I, definiamo $G(I)$ come la "<u>matrice diagonale delle potenze di I</u>"

$$G(I) = \text{diag}(p(I)) = \begin{bmatrix} I^0 & 0 & \cdots & 0 \\ 0 & I^1 & & \\ & & I^2 & \\ \vdots & & & \ddots \\ 0 & & & I^N \end{bmatrix} \qquad (5.12)$$

Nota: $G(I)=G(I)^T$ e $G(I)^{-1}=G(I^{-1})$ (5.13)

allora è immediato vedere che

$$p(Ix) = G(I)\, p(x) \quad , \text{ e} \quad p(x) = G(I^{-1})\, p(Ix),\ I\neq 0 \qquad (5.14$$

(dato che premoltiplicare una qualsiasi matrice o vettore co una matrice diagonale equivale a moltiplicarne ordinatamente l righe per gli elementi sulla diagonale)

Ora i problemi possono anche qui essere risolti con poch sostituzioni.

Analisi: fissato il vettore dei coefficienti d e l'indice I (o -più spesso- avendolo come parametro), determinare H(I)in m do che

$$f(Ix) = d^T p(Ix) = h(I)^T T(x) \qquad (5.15$$

sostituiamo

$$f(Ix) = d^T p(Ix) = d^T G(I) p(x) = d^T G(I) A^T T(x) = (AG(I)d)^T T(x) \qquad (5.16$$

e quindi, se $I\neq 0$

$$h(I) = A\, G(I)\, d \qquad (5.17$$

Sintesi: fissate le ampiezze delle armoniche h e l'indice I determinare d(I) in modo che

$$f(Ix) = h^T T(x) = d(I)^T p(Ix) \qquad (5.18$$

sostituiamo

$$f(Ix)=h^T T(x)=h^T B^T p(x)=h^T B^T G(I^{-1})p(Ix)=(G(I^{-1})Bh)p(Ix) \qquad (5.19)$$

e quindi, se $I\neq 0$

$$d(I) = G(I^{-1})Bh \qquad (5.20)$$

Mentre in quest'ultima espressione per d(I) la dipendenza da I è molto semplice, in quella per h(I) conviene effettuare qualche riarrangiamento ed esplicitarla per componenti.

La n-esima componente di h(I) è data dalla:

$$h_n(I) = A_{n,.}\, G(I)\, d = d^T\, G(I)\, A_{n,.}^T \qquad (5.21)$$

dove $A_{n,.}$ indica la n-esima riga di A.
Sfruttando il fatto che nella moltiplicazione di una matrice diagonale e di un vettore si possono scambiare gli elementi sulla diagonale della matrice con gli elementi del vettore, e ricordando che G(I)=diag(p(I)), otteniamo:

$$h_n(I) = d^T G(I) A_{n,.}^T = d^T\, \mathrm{diag}(A_{n,.}^T)\, p(I) =$$

$$= (\mathrm{diag}(A_{n,.}^T)\, d)^T\, p(I) \qquad (5.22)$$

da cui si vede che $h_n(I)$ è un polinomio in I i cui coefficienti sono contenuti nel vettore $(\mathrm{diag}(A_{n,.}^T)\, d)$, cioè sono i coefficienti del polinomio distorcente, "pesati" con la n-esima riga della matrice A.

Ricordando ancora la struttura triangolare e a scacchiera di A, si può affermare che $h_n(I)$ contiene solo le potenze con la stessa parità di n, e tra queste solo quelle comprese tra la n-esima e la N-esima.

Dato che in pratica I è vincolato tra -1 ed 1 (perchè f(.) viene di norma tabulata e memorizzata per i valori compresi fra questi estremi, nè è possibile "riciclare" per periodicità la tabella come nella tecnica della FM), ciò significa che se esso decresce in modulo, le armoniche superiori decadono più rapidamente di quelle di ordine più basso.

6 DISTORSIONE DELLA SOMMA DI DUE COSENI

Come già nella distorsione di una sola cosinusoide, il primo passo consiste anche qui nel procurarsi una scrittura compatta per i polinomi, questa volta nella somma di due variabili.

Proposizione: Un polinomio di coefficienti $d_0, d_1, \ldots, d_N$ nella somma di due variabili $(x+y)$, può essere espresso come una "forma bilineare" nei vettori $p(x)$ e $p(y)$:

$$f(x+y) = \sum_{n=0}^{N} d_n (x+y)^n = p(x)^T D\, p(y) \qquad (6.1)$$

dove D è una matrice $(N+1)\times(N+1)$ di componenti:

$$D_{m,n} = \begin{cases} d_{m+n} \binom{m+n}{m} & \text{per } 0 \le (m+n) \le N \\ 0 & \text{per } N < (m+n) \le 2n \end{cases} \qquad (6.2)$$

cioè

$$D = \begin{bmatrix} d_0 & d_1 & d_2 & d_3 & \cdots & d_N \\ d_1 & 2d_2 & 3d_3 & & & 0 \\ d_2 & 3d_3 & & & & \\ d_3 & & & & & \\ \vdots & & & & & \\ d_N & 0 & & & & 0 \end{bmatrix}$$

verifica: si basa sullo sviluppo delle potenze del binomio, e riarrangiamenti di sommatorie

$$f(x+y) = \sum_{k=0}^{N} d_k \sum_{m=0}^{k} \binom{k}{m} x^m y^{k-m} = \sum_{k=0}^{N} \sum_{m=0}^{N} d_k\, \mu(m,k) \binom{k}{m} x^m y^{k-m} =$$

(abbiamo introdotto l'indicatore $\mu(m,k) = \begin{cases} 1 \text{ se } m \le k \\ 0 \text{ se } m > k \end{cases}$)

$$= \sum_{m=0}^{N} \sum_{k=0}^{N} d_k \, \mu(m,k) \binom{k}{m} x^m y^{k-m} = \sum_{m=0}^{N} x^m \sum_{k=m}^{N} d_k \binom{k}{m} y^{k-m} =$$

(facciamo il cambiamento di indice n=k-m)

$$= \sum_{m=0}^{N} x^m \sum_{n=0}^{N} d_{m+n} \binom{m+n}{m} y^n = \sum_{m=0}^{N} x^m \sum_{n=0}^{N} D_{m,n} y^n = p(x)^T \, D \, p(y) \ .$$

Fatta questa premessa, i problemi vengono affrontati in modo formalmente analogo al caso del paragrafo precedente. Cambierà naturalmente la interpretazione dei risultati.

Caso senza indici: sia $x_1=\cos(2\pi F_1 t)$ e $x_2=\cos(2\pi F_2 t)$

analisi: fissato d, e dunque D, trovare una matrice H tale che

$$f(x_1+x_2) = p(x_1)^T \, D \, p(x_2) = T(x_1)^T \, H \, T(x_2) \quad , \forall x_1, x_2 \qquad (6.3)$$

facciamo le consuete sostituzioni

$$f(x_1+x_2) = (A^T T(x_1))^T \, D \, A^T \, T(x_2) = T(x_1)^T \, ADA^T \, T(x_2) \qquad (6.4)$$

essendo H unica

$$H = ADA^T \qquad (6.5)$$

sintesi: fissata una matrice H, trovare una matrice D (dalla quale risalire, se possibile, ad un vettore d) tale che:

$$f(x_1+x_2) = T(x_1)^T \, H \, T(x_2) = p(x_1)^T \, D \, p(x_2) \quad , \forall x_1, x_2 \qquad (6.6)$$

sostituiamo

$$f(x_1+x_2) = (B^T p(x_1))^T \, H \, B^T \, p(x_2) = p(x_1)^T \, BHB^T \, p(x_2) \qquad (6.7)$$

e quindi

$$D = BHB^T \qquad (6.8)$$

Vediamo ora di interpretare il significato della matrice H. Essa porta nella posizione m,n il coefficiente di $T_m(x_1)\cdot T_n(x_2)$, ovvero, ricordando che $x_1=\cos(2\pi F_1 t)$ e $x_2=\cos(2\pi F_2 t)$, il coefficiente di

$$\cos(2\pi F_1 t)\cdot\cos(2\pi F_2 t)=\frac{1}{2}\left[\cos(2\pi(mF_1-nF_2)t)+\cos(2\pi(mF_1+nF_2)t)\right]$$

Se F_1 ed F_2 sono tali che il loro rapporto è irrazionale, o razionale ma soddisfa certi vincoli dipendenti dal grado del polinomio distorcente, siamo certi che la distorsione non produce più di una componente spettrale ad ogni frequenza $mF_1\pm nF_2$. In questo caso, $H_{m,n}$ se uno degli indici è nullo, oppure $\frac{1}{2}H_{m,n}$ se entrambi gli indici sono diversi da zero, dà la ampiezza delle componenti spettrali a frequenza $(mF_1 \pm nF_2)$.

Alcune osservazioni che è utile fare sono le seguenti:

H, come D, è "specularmente triangolare superiore", il che significa che la riga spettrale a massima frequenza si trova alla frequenza $\max_{m+n=N}(mF_1+nF_2)= N\cdot\max(F_1,F_2)$; inoltre, questo significa che gli spettri "multifonici" ottenuti in questo modo sono diversi, e più poveri in componenti a frequenze elevate, di quelli che si producono moltiplicando due segnali con spettri armonici (in questo caso la matrice H in generale sarebbe piena)

H, come D, è simmetrica, ovvero le componenti spettrali a frequenze $mF_1 \pm nF_2$ e $nF_1 \pm mF_2$ hanno la stessa ampiezza

- benchè i segnali con spettro di questo tipo non siano in generale periodici, e quindi non ammettano sviluppo in serie di Fourier, è senz'altro anche qui proponibile e preferibile uno sviluppo in coseni di tipo bilaterale

$$f(\cos(2\pi F_1 t)+\cos(2\pi F_2 t)) = \sum_{m,n=-N}^{N} K_{m,n} \cos(2\pi(mF_1+nF_2)) \qquad (6.9)$$

ove i $K_{m,n}$ sono pari sia rispetto a m che a n:

$$K_{-m,-n}=K_{-m,n}=K_{m,-n}=K_{m,n} \qquad (6.10)$$

è immediato passare dagli H ai K:

$$K_{m,n}=\begin{cases} H_{m,n} & m\cdot n=0 \\ \frac{1}{4}H_{|m|,|n|} & m\cdot n\neq 0 \end{cases} \qquad (6.11)$$

il passaggio dall' $\frac{1}{2}$ della parità monodimensionale (a due metà) all' $\frac{1}{4}$ della parità bidimensionale (a quattro quadranti) è giustificato anche intuitivamente, e rende conto della apparente anomalia della matrice H.

Passiamo ora al caso in cui sono presenti gli indici.

analisi: fissato d, e dunque D, e gli indici I_1 ed I_2, trovare la matrice $H(I_1,I_2)$ tale che

$$f(I_1x_1+I_2x_2) = p(I_1x_1)^T D\, p(I_2x_2) = T(x_1)^T H(I_1,I_2)\, T(x_2) \qquad (6.12$$

con il consueto procedimento si arriva al risultato

$$H(I_1,I_2) = AG(I_1)DG(I_2)\, A^T \qquad (6.13)$$

sintesi: fissata H e I_1, I_2, trovare una $D(I_1,I_2)$ tale che

$$f(I_1x_1+I_2x_2) = T(x_1)^T\ H\ T(x_2) = p(I_1x_1)^T D(I_1,I_2)p(I_2x_2) \quad (6.14)$$

con le solite sostituzioni si giunge alla

$$D(I_1,I_2) = G(I_1^{-1})\,BHB^T\,G(I_2^{-1}) \quad (6.15)$$

Analogamente a quanto fatto nel paragrafo precedente, esplicitiamo $H(I_1,I_2)$ per componenti in modo da rendere più evidente possibile la dipendenza dagli indici

$$H_{m,n}(I_1,I_2) = A_{m,.}\ G(I_1)\ D\ G(I_2)\ A_{.,n} \quad (6.16)$$

usando la solita intercambiabilità nella moltiplicazione di vettori e matrici diagonali, otteniamo

$$H_{m,n}(I_1,I_2) = p(I_1)^T\ \ diag(A_{m,.}^T)\ D\ diag(A_{.,n})\ \ p(I_2) \quad (6.17)$$

espressione che significa che $H_{m,n}(I_1,I_2)$ è data pur essa da una "forma bilineare" delle potenze di I_1 e delle potenze di I_2 (e non tuttavia da un polinomio in (I_1+I_2), poichè in generale $diag(A_{m,.}^T)\ D\ diag(A_{.,n})$ non conserva la struttura di D).

Anche qui gli indici devono in pratica sottostare a certi vincoli (ad es. $|I_1| + |I_2| \leq 1$ oppure 2) e valgono considerazioni, circa il decadimento ed il comportamento delle componenti spettrali, analoghe (ma "bidimensionali") a quelle fatte nel precedente paragrafo.

7 CONCLUSIONI

Si è trattata la distorsione non lineare tramite polinomi.

E' stata presentata una descrizione matriciale delle relazioni intercorrenti fra i coefficienti del polinomio distorcente, gli indici di distorsione ed i coefficienti spettrali, sia nel caso standard della distorsione di una sola cosinusoide, che nel caso della distorsione della somma di due cosinusoidi.

Pensiamo che i risultati esposti, per compattezza e simmetria, oltre a consentire una buona economia concettuale, possano essere un valido punto di partenza per la stesura di programmi applicativi, nonchè per ulteriori studi.

BIBLIOGRAFIA

1. Schaefer R.A., "Electronic Musical Tone Production by Nonlinear Waveshaping", J. Audio Eng. Soc., vol.18, n.4, pp.413-417 (Aug. 1970)

2. Suen C.Y. "Derivation of Harmonic Equations in Nonlinear Circuits", J. Audio Eng. Soc., vol.18, n.6, pp.675-676 (Dec. 1970)

3. Arfib D. "Digital Synthesis of Complex Spectra by Means of Multiplication of Nonlinear Distorted Sine Waves", J. Audio Eng. Soc.,vol.27, pp.757-768, (Oct. 1979)

4. Le Brun M. "Digital Waveshaping Synthesis", J. Audio Eng. Soc., vol.27,pp.250-265 (Nov. 1979)

5. Reinhard P.,Ropelato F.,Vecelli A., "Distorsione Non Lineare tramite Polinomi", Rel. d'Esame,Istituto di Elettrotecnica ed Elettronica, Univ. di Padova, 1979

Ennio SIMIONI (C.S.C. - Università di Padova)

ANALISI DEL FILTRO FB1 DEL MUSIC V

Il linguaggio di programmazione MUSIC V provede l'uso di filtri digitali per il trattamento dei segnali musicali. Tra questi viene usato il filtro denominato FB1 come filtro passa-banda.
Sono state analizzate le caratteristiche filtranti di FB1 facendo riferimento ai parametri consigliati dal MUSIC V e i risultati ottenuti confrontati con le specifiche fornite.
Sono state altresì ricavate le relazioni effettivamente esistenti tra parametri forniti e la struttura del filtro, evidenziando le eventuali incongruenze e/o incompatibilità. Si sono suggerite infine alcune parziali modifiche alla struttura del sistema atte a migliorarne le prestazioni, evidenziando gli effetti più immediati di tali modifiche.

Many digital filters are involved in the treatment of musical signals with the MUSIC-V programming language. Among these filters, the one called FB1 is used as a band pass filter.
Referring to the parameters recommended by the MUSIC-V, the filtering characteristics of FB1 have been analysed and the results compared with the given specifications.
The actually existing relations between the given parameters and the filter structure have been obtained, emphasiging the eventually existing incompatibilities and inconsistencies.
Finally some partial modifications to the system's structure have been suggested with the intention to improve the performances, and the most immediate results of these modifications have been emphasised.

Ennio Simioni

Analisi del filtro "FB1"

Il linguaggio di programmazione MUSIC-V prevede l'uso di filtri per il trattamento dei segnali musicali.
Ci occupiamo qui del filtro "FB1" che è indicato come filtro passa - banda.
L'istruzione MUSIC-V che richiama il filtro FB1 (analoga a quelle degli altri filtri, tranne lievi differenze nei parametri supplementari) è del tipo :

FB1 I1 O I2 P(I) P(J)

Si fa riferimento , qui , al sistema MUSIC-V implementato presso l' IRCAM di Parigi, nella versione 1979, da cui è derivata la versione in uso presso il Centro Sonologia Computazionale dell'Università di Padova.
Nell'istruzione MUSIC-V vista i vari campi hanno il seguente significato :

1) FB1 - nome simbolico che identifica il tipo di filtro che viene richiamato;
2) I1 - campo contenente l'ingresso del filtro;
3) O - campo che conterrà il segnale d'uscita;
4) I2 - coefficiente che determina la posizione di zeri e poli della H(z) e perciò le caratteristiche filtranti finali;
5) P(I) - campo ausiliario di memoria;
6) P(J) - campo ausiliario di memoria.

La relazione Ingresso/Uscita per l' "FB1" è :

(1) $Y(I)=X(I)-I2/2*(X(I)-X(I-1)+2*Y(I-1))$;

da questa, usando le trasformate Z, si può ricavare la H(z) del filtro :

(2) $H(z**-1) = ((1-I2/2)+I2/2*z**-1)/(1+I2*z**-1)$

(3) $H(z) = (1-I2/2)*(z+I2/(2-I2))/(z+I2)$

I valori di polo e zero si ricavano immediatamente dalla (3) :

(4) zero -------- $Z=-I2/(2-I2)$

(5) polo -------- $P=-I2$

Il valore da assegnare a I2 viene calcolato in base alla larghezza della banda passante che si vuole ottenere, viene posto infatti :

(6) $I2 = R_h - R_l$

dove - R_h - è la frequenza di taglio (-3 db) superiore , normalizzata

R_l - è la frequenza di taglio (-3 db) inferiore , normalizzata.

Facciamo ora due esempi:(*) posto R_h=2ØØØ Hz ------ Ø.2Ø

R_l=18ØØ Hz ------ Ø.18

oppure R_h=4ØØØ Hz ------ Ø.4Ø

R_l=38ØØ Hz ------ Ø.38

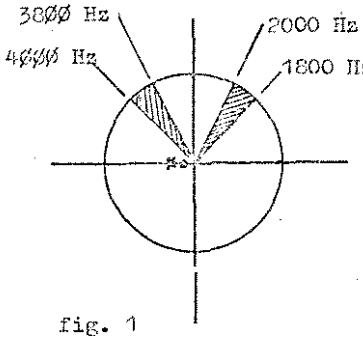


fig. 1

si ottiene in entrambi i casi (v. fig. 1) :

Zero --- Z = -Ø.Ø1

Polo --- Z = -Ø.Ø2

Si noti che polo e zero sono ambedue negativi e praticamente coincidenti, il che significa effetti filtranti sul segnale d'ingresso minimi; e inoltre che le frequenze di taglio non influenzano la posizione del polo e dello zero.

Per la stabilità del filtro si ha la condizione su I2 :

(7) $|I2| < 1$

di qui la necessità di normalizzazione di R_h e R_l.

La posizione del polo e dello zero in funzione di I2 sono date in fig. 2.

Per I2 - positivo polo e zero sono vicini e negativi, hanno influenza alle alte frequenze (maggiori di $\pi/2$);

- negativo hanno influenza per frequenze minori di $\pi/2$, corrispondenti a frequenze inferiori a $f_c/4$, sono tra loro più distanziati e quindi la H(z) è meno piatta che nel caso precedente.

(*) In questo e in tutti gli esempi numerici che seguiranno, si assumerà per f_c, frequenza di normalizzazione o di campionamento, un valore di 1Ø kHz.

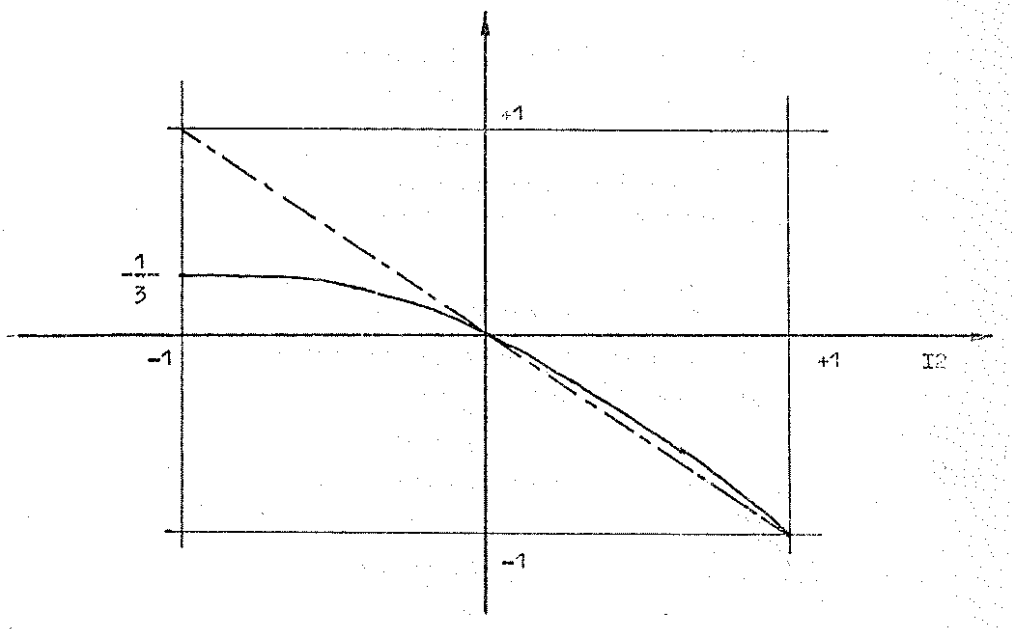


fig. 2 -- Posizione del polo (—·—·—) e dello zero (———) in funzione di I2.

Si noti come il polo e lo zero sono indipendenti dalle frequenze di taglio e sensibili solo alla larghezza di banda (e anche in questo caso gli spostamenti sono minimi).
Come si vede dalla fig. 2, l'influenza zero-polo (leggi "interferenza) è quasi totale per I2 positivo, ci sono differenze minime tra le loro posizioni, mentre assumono valori abbastanza diversi uno dall'altro (il che equivale ad una forma più decisa per la H(z), ossia non troppo piatta) per I2 negativo, valore, questo, che è incompatibile con la definizione data per I2.
Conclusioni : (1) - La disposizione del polo e dello zero e soprattutto il loro numero, (1 solo polo), indicano chiaramente che il filtro non può essere un passa-banda.

(2) - Supponendo di accettarlo così com'è, la scelta del valore da dare a I2 è la peggiore che per questo filtro si possa fare; il minimo per evitare un "altipiano" da Ø a 5ØØØ Hertz (banda utilizzabile) sarebbe di scegliere :

$$I2 = -(R_h - R_l)$$

rimane in ogni caso "ignota" la relazione che lega I2, e quindi il polo, alle frequenze di taglio.

(3) - Tale relazione infatti non esiste , il filtro ha la stessa disposizione zero-polo qualunque sia la f_o centrale desiderata; si ottiene una H(z) identica per un filtro a 5ØØ Hertz e uno a 4ØØØ Hertz qualora le due bande passanti siano uguali.

(4) - L'algoritmo è perciò del tutto inutilizzabile per gli scopi desiderati!

La migliore utilizzazione di FB1, lasciandolo così com'è, è quella di passa-basso; infatti, come si è già visto, la maggiore distanza tra zero e polo (ossia una H(z) più "marcata") si raggiunge per I2 negativo, quindi per polo e zero positivi.
Un suo uso come passa-alto è, più che sconsigliabile, inutile; si otterrebbe praticamente un passa-tutto.
Per vedere l'andamento della H(z) si può calcolarne il modulo :

(8) $$H(e^{j\omega}) = H(z)\Big|_{z=e^{j}}$$

(9) $$\left|H(e^{j\omega})\right| = K\sqrt{(1+x*\cos(\omega))/(1+y*\cos(\omega))}$$

(10) $$\left|H(e^{j\omega})\right|' = \frac{K}{2}\,\frac{\sin(\omega)*(y-x)}{(1+y*\cos(\omega))^{3/2}*(1+x*\cos(\omega))^{1/2}}$$

con :

$$x = 2*I2*(2-I2)/(I2+(2-I2)^2)$$

$$y = 2*I2/(I2^2+1)$$

$$K = (1-I2/2)*\sqrt{2*(I2^4-2*I2^3+3*I2^2-2*I2+2)/(2-I2)^2}$$

La derivata (eq. (1Ø)) si annulla in $\omega = Ø$ e in $\omega = \pi$, ossia agli estremi della banda utile ed è positiva o negativa (il che implica una H(z) crescente o decrescente) a seconda del valore assunto dal termine (y-x) che è funzione di I2.
Imponiamo la "crescita" di H(z) , fatto questo che corrisponde ad avere un passa-alto, cioè I2 Ø.
Dalla definizione di y e di x si ottiene :

(11) $y - x > \emptyset$

(11.a) $$2 * \frac{I2}{I2^2+1} + 2 * \frac{I2*(I2-2)}{I2+(2-I2)^2} > \emptyset$$

(11.b) $$\frac{I2^2+I2*(4-2*I2+I2^2)+(I2^3+I2)*(I2-2)}{(I2^2+1)*(I2+(2-I2)^2)} > \emptyset$$

il denominatore è sempre positivo qualunque sia il valore di I2; può essere trascurato.

(11.c) $I2^4-I2^3+2*I2 > \emptyset$

(11.d) $I2*(I2^3-I2^2+2) > \emptyset$

Il termine tra parentesi della (11.d) è sempre positivo qualunque sia il valore di I2, per cui si può dire subito che la derivata segue il segno di I2; questo corrisponde a dire che la derivata è positiva , la H(z) crescente, ossia si ha un passa-alto, per I2 positivo , cioè per un polo negativo; viceversa per I2 negativo, vedi fig. 3.
In ogni caso la risposta in frequenza è una funzione monotona qualunque sia il valore di I2, il che conferma l'impossibilità di ottenere un filtro passa-banda dalla relazione Imput/output fornita.

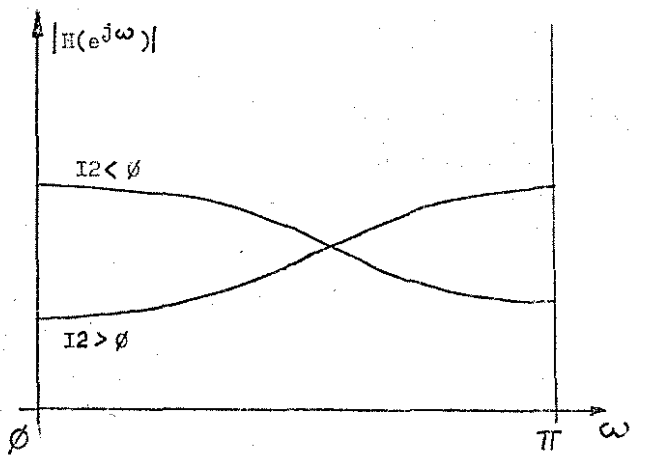


Fig. 3
Andamento qualitativo della risposta in frequenza per due valori di I2.

La posizione relativa polo-zero è :
per $I2 < \emptyset$:

$$\text{zero} \quad --- \quad \frac{-I2}{2-I2}$$

$$\text{polo} \quad --- \quad -I2$$

il rapporto Polo/Zero è :

$$(12) \qquad RP\% = \frac{-I2}{\frac{-I2}{2-I2}} = 2-I2 \geqslant 1 \qquad \forall\ I2$$

$$2 \leqslant RP\% \leqslant 3$$

questo significa che il modulo del polo è sempre maggiore del modulo dello zero : il polo ha più influenza alle basse frequenze; man mano che si sale in frequenza (v. fig. 4) si ha che il rapporto tra i vettori che congiungono polo e zero al generico punto $e^{j\omega}$ che percorre la circonferenza, si fa sempre più prossimo ad 1, fino a raggiungere il minimo per $\omega = \pi$; è un passa-basso.

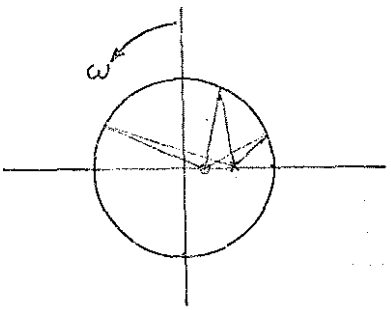


Fig. 4

Per $I2 > \emptyset$:

$$\text{zero} \quad --- \quad \frac{-I2}{2-I2}$$

$$\text{polo} \quad --- \quad -I2$$

il rapporto Polo/Zero è :

$$(13) \qquad RP\% = \frac{-I2}{\frac{-I2}{2-I2}} = 2-I2 \geqslant 1 \qquad \forall\ I2$$

$$1 \leqslant RP\% \leqslant 2$$

si vede (fig. 5) come, pur rimanendo sempre il modulo del polo maggiore del modulo dello zero, la distanza è ora minore che nel caso precedente; l'interferenza reciproca è ora più forte e

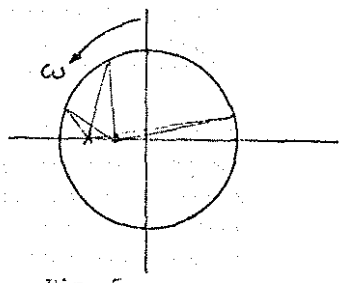


Fig. 5

questo porta a variazioni minori nella forma della H(z).
Valgono le stesse considerazioni di prima solo con andamento opposto della risposta in funzione della frequenza.
Il guadagno del filtro, visto che si tratta di un passa-alto o di un passa-basso, può venir calcolato imponendo opportuni valori a Z, Z = 1 per il passa-basso, Z = -1 per il passa-alto, e ovviamente rispettando i valori permessi a I2, legati, come si è visto, al problema della stabilità del filtro.

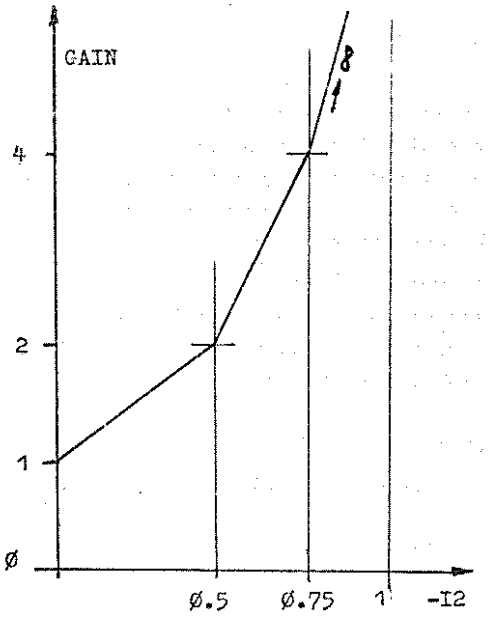


Fig. 6 Andamento del guadagno in Z=1, in funzione di I2.

Per I2 positivo se ottiene un passa-alto, si avrà allora :

(14) $GAIN = H(z)\Big|_{z=-1} = 1$

costante, indipendente dal valore assunto da I2.
Per I2 negativo si ottiene un passa-basso, si avrà allora :

(15) $GAIN = H(z)\Big|_{z=1} = \frac{1}{1+I2}$

Di qui, vedi fig. 6, si ha anche che non si può spingere troppo I2 verso -1 per ottenere poco meno che un discreto passa-basso, in quanto, a meno di attenuazioni sul segnale d'ingresso e/o sulla H(z) stessa, si avrebbe un'amplificazione teoricamente infinita, in ogni caso già molto elevata per I2 < -Ø.5. Il "molto elevata" va inteso nel senso che da un filtro non ci si aspetta un'amplificazione, ma piuttosto un'attenuazione , anche in banda passante, rispetto ai valori dell'ingresso.
Questo è vero più per filtri analogici, che bene o male assorbono sempre un pò di potenza dal segnale, che per quelli digitali, per i quali è sufficiente inserire in serie alla realizzazione della H(z) un opportuno moltiplicatore per avere all'uscita il segnale, oltre che filtrato, riscalato (amplificato o attenuato, a seconda delle esigenze)

Per rendere FB1 un passa-banda, come dovrebbe essere, bisogna aggiungere almeno un polo, in posizione opportuna; questo si può fare introducendo nella relazione I/O un ulteriore elemento di ritardo, diciamo Y(I-2), con un opportuno coefficiente I3.
La nuova relazione I/O è perciò :

(16) $$Y(I)=X(I)-I2/2*(X(I)-X(I-1)+2*Y(I-1))-I3*Y(I-2)$$

La corrispondente H(z) è ora : (posto B=I2 e C=I3)

(17) $$H(z^{-1}) = \frac{1-B/2+B/2*z^{-1}}{1+B*z^{-1}+C*z^{-2}}$$

(18) $$H(z) = (1-\frac{B}{2})* \frac{z*(z+B/(2-B))}{z^2+B*Z+C}$$

Come si vede è stato aggiunto anche uno zero in Z = Ø; la sua influenza sul modulo è però nulla : il suo contributo è pari ad 1 su tutto il campo di frequenze.
I poli sono, ora :

(19) $$P_{1,2} = \frac{-B \pm \sqrt{B^2-4*C}}{2}$$

Due poli reali, distinti o meno, servono poco alla realizzazione di un passa-banda; ci servono invece due poli complessi coniugati e questo si ottiene se :

(2Ø) $$B^2-4*C < Ø \quad ====> \quad C > \frac{B^2}{4}$$

Si ottiene in questo caso :

(21) $$P_{1,2} = -\frac{B}{2} \pm j \frac{\sqrt{4*C - B^2}}{2}$$

dove : $-\frac{B}{2}$ è la parte reale dei poli

$\frac{4*C - B^2}{2}$ è la parte immaginaria

Si noti come C abbia il controllo della parte immaginaria dei poli.

Calcoliamo ora il modulo di questi :

$$(22) \qquad MOD = \frac{I2^2}{4} + C - \frac{I2^2}{4} = C$$

$$(23) \qquad |P_{1,2}| = \sqrt{MOD} = \sqrt{C}$$

questo implica ovviamente che C dovrà essere positivo.
Le condizioni per la stabilità sono ora :

$$(24.a) \qquad \emptyset < C < 1$$

$$(24.b) \qquad |I2| < 2$$

Per I2 negativo si ha che i poli giacciono (vedi fig. 7) sul semicerchio destro, il che corrisponde a frequenze f minori di $f_c/4$ (nel nostro caso minori di 25ØØ Hz) e quindi si avranno frequenze di centro-banda f_o in tale intervallo; per I2 positivo la situazione si ribalta e si avranno f_o maggiorá di 25ØØ Hz (caso che corrisponde nel "vecchio" FBM alla distinzione tra passa-basso e passa-alto).

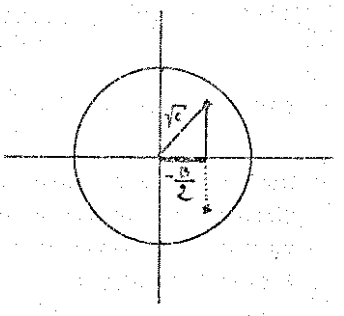


Fig. 7

Per quel che riguarda la forma della risposta del filtro si può dire che essa sarà tanto più selettiva quanto più è alto il valore del modulo dei poli, ossia C, infatti tanto più sarà difficile per gli zeri(uno solo in verità, quello in Z = Ø ha modulo unitario a tutte le frequenze : è come se non ci fosse agli effetti del modulo) neutralizzare l'influenza dei poli, dato che la distanza polo-cerchio rimane piccola per un più ampio tratto di frequenze.
Per la frequenza centale del filtro si ha che :

θ, angolo di fase dei poli, ossia angolo (e quindi frequenza) al quale l'infuenza dei poli stessi è massima, si può ottenere come :

$$(25) \qquad \theta = \text{Arcos}\left(\frac{\text{Parte Reale}}{\text{Modulo}}\right) = \text{Arcos}\left(\frac{-I2}{2\sqrt{C}}\right)$$

oppure come :

$$(26) \qquad \theta = \text{Arctang}\left(\frac{\text{Parte Immaginaria}}{\text{Modulo}}\right)$$

cioè :

(26.a) $$\Theta = \text{Arctang} \left(\frac{-2\sqrt{0 - I2^2/4}}{I2} \right)$$

La frequenza corrispondente è facilmente ricavabile da Θ come :

(27) $$f_o = f_c \times \Theta / 2\pi$$

Per il guadagno statico si ha :

(27) $$GAIN = H(z)\Big|_{z=1} = \frac{1}{1+I2+I3}$$

I valori del guadagno per vari valori di I2 e di I3 sono riportati in figura 8 e tabulati in Tabella 1.
Si può vedere come il guadagno per una $f_o = \pi/2$, ossia $f_c/4$ che corrispondono a 2500 Hertz, vale 1 per I3=0 cioè modulo nullo e cala al crescere del modulo stesso.
Poli con Θ minore di $\pi/2$, corrispondeni a I2 negativo, danno guadagni minori di 1, il che coincide con ciò che ci si aspettava essendo l'influenza di questi maggiore a $\omega = 0$, cioè a Z = 1, che a $\omega > \pi/2$; si vede infatti come per I2 positivo, cioè Θ maggiore di $\pi/2$ il guadagno cala al crescere del modulo.
Un grafico, analogo a quello visto prima, per $\omega = \pi$, Z = -1, mostra che :

(28) $$GAIN = H(z)\Big|_{z=-1} = \frac{1-I2}{1-I2+I3}$$

I valori del guadagno per vari valori di I2 e di I3 sono riportati in figura 9 e tabulati in Tabella 2.
La dispersione dei valori del guadagno è molto minore che nel caso precedente a causa della maggiore vicinanza dello zero ai poli; non sono stati presi in considerazione valori di I2 maggiori di 1 e minori di -1.
Vediamo infatti cosa succede in $\omega = 0$, Z = 1 con I2 < -1.
Dalla (27) si ottiene, posto :

(29) $$X = 1+I2$$
$$-1 < X < 0$$

(30) $$GAIN = 1/(X+I3)$$

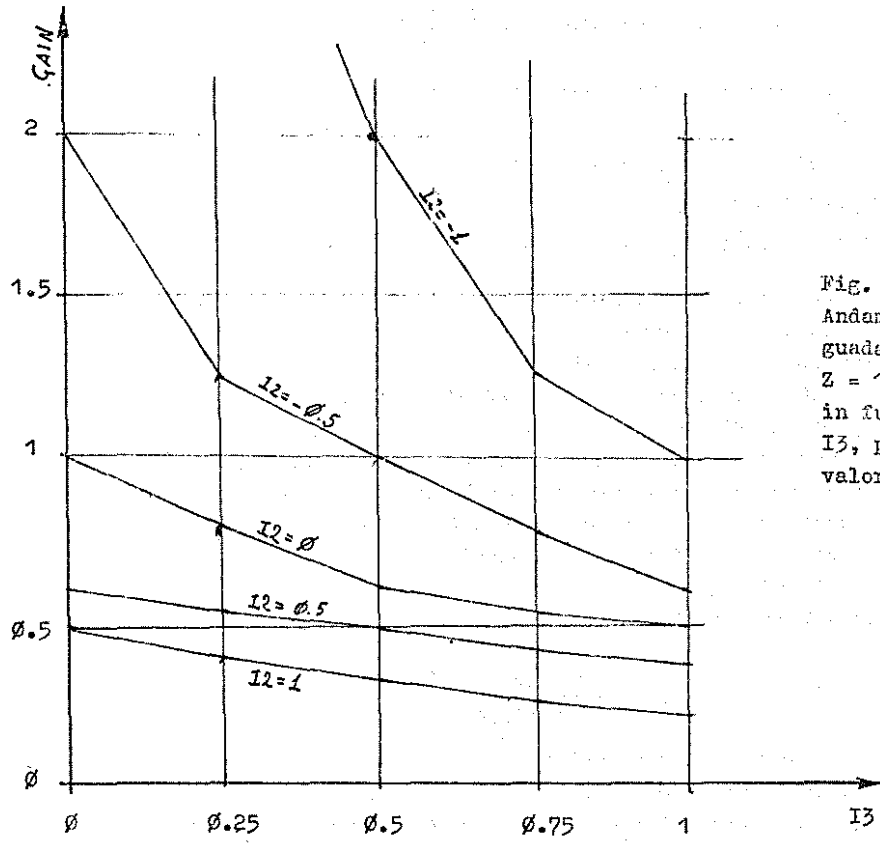


Fig. 8
Andamento del guadagno in $Z = 1, \omega = \emptyset$, in funzione di I3, per vari valori di I2.

I3 \ I2	-1	-Ø.5	Ø	Ø.5	1
Ø	→ ∞	2	1	Ø.66	Ø.5
Ø.25	4	1.3	Ø.8	Ø.57	Ø.44
Ø.5	2	1	Ø.6	Ø.5	Ø.4
Ø.75	1.3	Ø.8	Ø.57	Ø.44	Ø.36
1	1	Ø.66	Ø.5	Ø.4	Ø.33

Tabella 1.

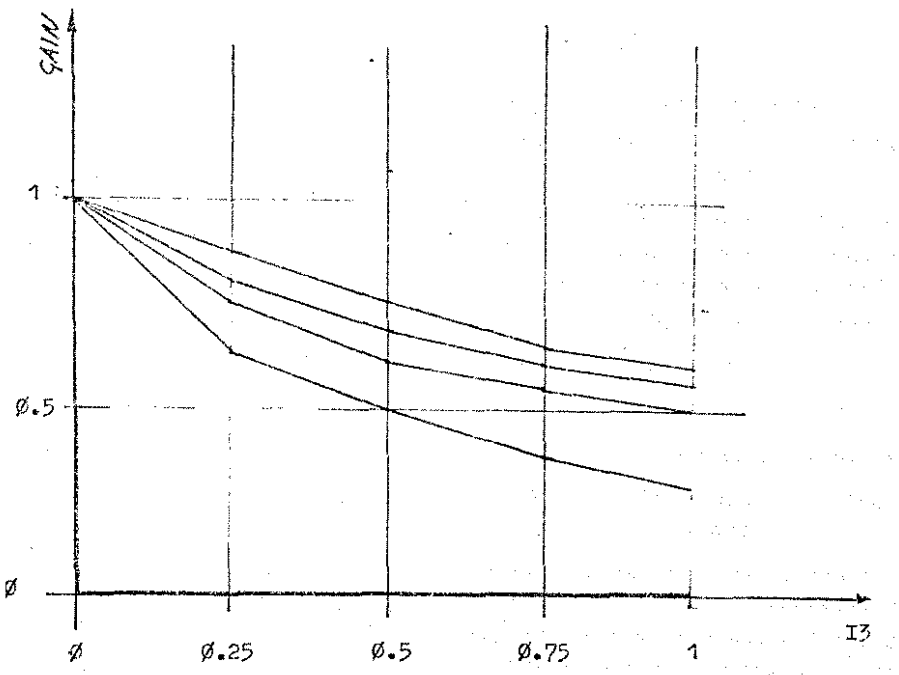


Fig. 9 Andamento del guadagno in $Z = -1, \omega = \pi$, in funzione di I3, per vari valori di I2.

I3 \ I2	-1	-Ø.5	Ø	Ø.5	1
Ø	1	1	1	1	Ø
Ø.25	Ø.88	Ø.85	Ø.8Ø	Ø.66	Ø
Ø.5	Ø.8	Ø.75	Ø.66	Ø.5	Ø
Ø.75	Ø.72	Ø.66	Ø.57	Ø.4Ø	Ø
1	Ø.66	Ø.6Ø	Ø.5	Ø.33	Ø

Tabella 2

da questo si ha che :

(31) $\lim_{I3 \to -X} GAIN = \infty$

Per I2 > 1 , invece, non si hanno problemi.
Vediamo il caso analogo in $\omega = \pi$, Z = -1 con I2 > 1.
Dalla (28) si ottiene , posto :

(32) $X = 1 - I2$

$-1 < X < \emptyset$

(33) $GAIN = X / (X + I3)$

da questa si ha che :

(34) $\lim_{I3 \to -X} GAIN = \infty$

Si ricordi che si parla di moduli, perciò di quantità positive per definizione.
In conclusione,per evitare che GAIN tenda a infinito, si ha che bisogna scartare i valori di I2 < -1 in $\omega = \emptyset$ e i valori di I2 > 1 per l'analogo in $\omega = \pi$.
Il valore che il guadagno assume per I2 = 1 in $\omega = \pi$, è comprensibile se si pensa che lo zero giace esso stesso sul cerchio di raggio unitario :

(35) $Zero\Big|_{I2=1} = -\frac{1}{2 + 1} = -1$

e perciò qualunque sia il modulo dei poli si avrà sempre uno "$\emptyset$" al numeratore.
Pertanto si ha che I2 dovrà essere minore di 1, in modulo, come nel vecchio filtro; questo porta però a limitazioni anche per i poli che pure dipendono da I2.
Abbiamo infatti che i poli, con I2 minore di 1,in modulo, possono variare, come parte reale, nella zona - I2/2 + I2/2, ossia -1/2 + 1/2, valori questi che racchiudono la zona compresa tra

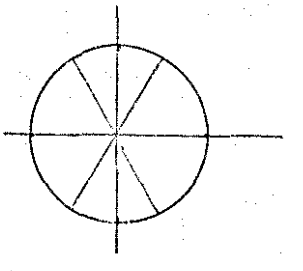


Fig. 1$\emptyset$

60° ($\pi/3$) e 120° ($2/3\ \pi$) (non avvicinarsi troppo ai limiti !);
in termini di frequenza : tra $f_c/6$ e $f_c/3$, nel nostro caso tra
i 16ØØ Hertz e i 32ØØ Hertz (vedi fig. 1Ø).
Per evitare tale limitazione in frequenza sarebbe opportuno eliminare
l'interdipendenza zero-poli; questo si può fare in due modi :

1 -- eliminazione totale del termine X(I-1) dalla relazione Input/Output del filtro;

2 -- moltiplicare tale termine per un suo-coefficiente I4 e non per I2/2 , come nella relazione originale/

La soluzione (1) è indubbiamente la più semplice, ma ha l'inconveniente di lasciare ai soli poli il controllo della "forma" dell'H(z).
La soluzione (2) permette, a parità di f_o, e quindi di poli, di modificare la H(z) lavorando sulla posizione dello zero, vedi fig. 11, (come al solito lo zero in Z = Ø è come se non ci fosse : il suo unico contributo è l'introduzione di una componente lineare nella fase).
Supponendo di dare alla relazione I/O la forma :

(36) $Y(I) = X(I)-I4*X(I-1)-I2*Y(I-1)-I3*Y(I-2)$

si ottiene una H(z) del tipo :

(37) $$H(z) = \frac{Z*(Z - I4)}{Z^2+I2*Z+I3}$$

A questo punto, però, del filtro originale rimane praticamente ben poco : stiamo riproponendo, infatti, qualcosa di simile al filtro FLT.

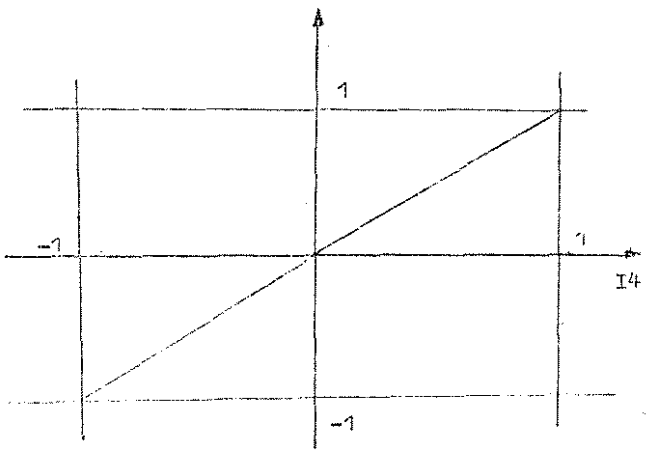


Fig. 11
Andamento dello zero in funzione di I4.

BIBLIOGRAFI/

J.L. RICHER

"MUSIC-V : Manuel De Réference"
IRCAM, Paris 1979

A.V. OPPENHEIM e R.W. SHAFER
"Digital Signal Processing"
Englewood Cliffs, Prentice-Hall 1975

E. SIMIONI
"Un Programma Per L ' Analisi Di Reti Digitali"
Tesi di Laurea, Padova 1981

A P P E N D I C E

A titolo di esempio viene di seguito riportata l'analisi del filtro FB1 eseguita al calcolatore utilizzando il linguaggio D.N.A. (Digital Network Analysis).
Tale linguaggio, sviluppato dall'autore presso l'Università di Padova, si rivolge in modo specifico all'analisi di strutture digitali quali, appunto, la rete interessata.
L'appendice consiste di due gruppi di figure rappresentanti rispettivamente :

Fig. 1 -- Testo del programma DNA

" " 2 -- Risposta in frequenza del filtro

" " 3 -- Poli e zeri della H(z) del filtro con rappresentazione grafica

per due scelte del parametro I2; la prima (Fig. _.A) fatta seguendo le specifiche IRCAM, la seconda (Fig. _.B) effettuata dopo l'analisi del filtro ossia tenendo conto delle caratteristiche del filtro.

```
[illegible]   [illegible] VERSION 1                TIME = 19.26.03    DATE = 05 MAY 81     PAGE  2

STMT      SOURCE LISTING

  1    INITIAL**************************************************************************

  2        SAMPLE FREQUENCY               10000
  3        ST[illegible] INPUT            1
  4        STA[illegible] OUTPUT          3
  5        [illegible]                    .FM1(A.B.C)
  6             [illegible]               1.3.A
  7             [illegible]               2.3.B
  8             [illegible]               2.1.C
  9             DELAY                     1.2

 10    [illegible]************************************************************************

                                            PROVA SECONDO I R C A M
 11        .FM1                             1.0.95.0.05.-0.1
           *****[illegible]                   1.    3. 9.50000E-01
           *****[illegible]                   2.    3. 5.00000E-02
           *****[illegible]                   2.    1.-1.00000E-01
           *****DELAY                         1.    2. 1.00000E+00

 12    [illegible]***********************************************************************

 13        [illegible]
 14                 [illegible]           3
 15                 [illegible]           3
 16                 *A*VALUE
 17        [illegible]                    1.1000.0.
                                          1.2500.0.
                                          1.4000.0
 18                 [illegible]           3
 19                 [illegible]           3
 20                 [illegible]           3
 21        [illegible]                    0.5000
 22                 [illegible]           MODULE.DA.PHASE.06
 23                 [illegible]           REAL
 24                 [illegible] 7 [illegible]
```

Fig. 1.A

Fig. 2.A

RICHIESTA # 3 FREQUENCY

ZERI DELLA FUNZIONE DI TRASFERIMENTO

	PARTE REALE	PARTE IMMAGINARIA	MODULO	FASE	FREQUENZA
1° ZERO	-5.263156E-02	+ 0.000000E+00	5.263156E-02	1.000000E+00	5.000000E+03

Fig. 3.A

RICHIESTA # 3

POLI DELLA FUNZIONE DI TRASFERIMENTO

	PARTE REALE	PARTE IMMAGINARIA	MODULO	FASE	FREQUENZA
1° POLO	-4.999999E-02	+ 0.000000E+00	4.999999E-02	1.000000E+00	5.000000

Fig. 3.A (continuazione)

Fig. 3.A (continuazione)

STMT SOURCE LISTING

```
  1   INITIAL*****************************************************************
  2         SAMPLE FREQUENCY                1000
  3         NUMBER OF INPUT                 1
  4         STANDARD OUTPUT                 4
  5         SELECT PARAMETERS               .F@1(A+H+C)
  6               TRANSF                    1+3+A
  7               TRANSFH                   2+3+H
  8               TRANSFX                   2+1+C
  9               DELAY                     1+2
 10   SET*********************************************************************
 11                                         PROVA DOPO L'ANALISI
 11         .F@1                            1+1.35+-0.35+0.7
            +++++TRANSFA                       1.    3.  1.35000E+00
            +++++TRANSFH                       2.    3. -3.50000E-01
            +++++TRANSFX                       2.    1.  7.00000E-01
            +++++DELAY                         1.    2.  1.00000E+00
 12   ANALYSIS****************************************************************
 13         IMPULSE
 14               PLOT                      3
 15               SHAPE                     3
 16               ARGUMENT
 17         SIN@SQUARE                      1+1000+0+
                                            1+2500+0+
                                            1+4000+0
 18               PLOT                      3
 19               SHAPE                     3
 20               DOUBLE                    3
 21         FREQUENCY                       0+5000
 22               SHAPE                     MODULE+DB+PHASE+DG
 23               DOUBLE                    REAL
 24               POLE ZERO
```

Fig. 1.13

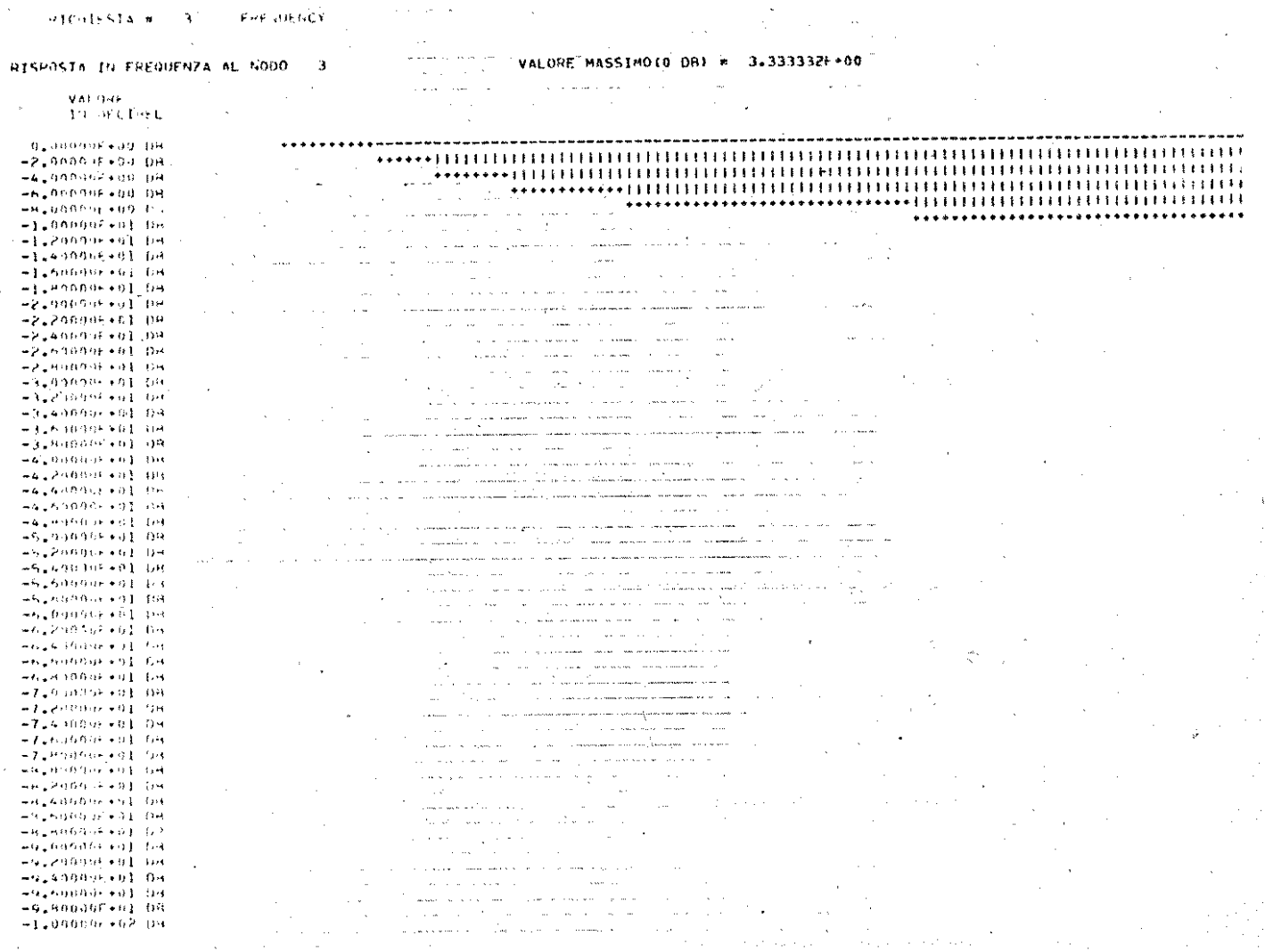

richiesta n 3 FREQUENCY
RISPOSTA IN FREQUENZA AL NODO 3
VALORE MASSIMO(0 DB) = 3.333332E+00
VALORE IN DECIBEL

Fig. 2.B

		DOUBLE	FALSE	FALSE
1	2.542594E-01 + 0.000000E+00	2.592593E-01	0.000000E+00	0.000000E+00

Fig. 3.B

Fig. 3.3 (continuazione)

Fig. 3.

Progetto di grammatica generativa di melodie e di armonie

M. BARONI - UNIVERSITA' DI BOLOGNA
R. BRUNETTI, C. JACOBONI - UNIVERSITA' DI MODENA

Riassunto

Allo studio della grammatica della melodia presentato dal gruppo di Modena e Bologna nei precedenti Colloqui d'Informatica Musicale si è affiancato recentemente un progetto di analisi riguardante l'armonizzazione di una melodia data.

Nella memoria verrà assegnata particolare enfasi alla descrizione di questa seconda parte del progetto, che ha ancora come oggetto di studio i Corali a quattro voci di Bach.

Il progetto di armonizzazione avviene tramite un albero generativo che fissa prima le funzioni armoniche di maggiore importanza strutturale e successivamente quelle di passaggio via via meno importanti. Completata questa fase si genera, adattando i procedimenti della grammatica della melodia, la linea del basso e successivamente quella delle altre parti. Man mano che le altre voci vengono generate si identificano sempre più precisamente le armonie da utilizzare all'interno delle funzioni armoniche prefissate.

Verranno infine presentati esempi di armonizzazione insieme ad esempi di produzione melodica che illustrano i risultati ultimi della grammatica della melodia la quale è ora in grado di generare un intero corale controllando le macro e le microstrutture.

Abstract

A new project on the harmonization of a given melody has been recently started by the group operating at Bologna and Modena, as a branch of the studies on melodic grammars presented at previous Colloqui di Informatica Musicale. This new aspect of the research, still dealing with Bach's four voice Chorales, is here emphasised.
The harmonization process is based on a generative tree which first produces the harmonic functions of highest structural priority and then those at lower levels. In the successive steps the bass melodic line and other voices are generated, by properly applying the melodic grammars. As voices are generated, harmonies to be used within the already fixed harmonic functions are precisely determined. Exemples of melodic generation and harmonization are presented to illustrate the most recent results obtained by the melodic grammar, which can now generate a whole chorale and control its micro and macrostructures.

PROGETTO DI GRAMMATICA GENERATIVA DI MELODIE E DI ARMONIE

M.BARONI,
Istituto di Studi Musicali e Teatrali,
Università di Bologna;

R.BRUNETTI, C.JACOBONI,
Centro di Calcolo Elettronico,
Università di Modena.

Questa comunicazione riporta gli sviluppi di una ricerca riguardante l' elaborazione di una grammatica generativa di melodie e di armonie.
Alcuni risultati di tale ricerca sono stati presentati in precedenza [1] assieme ad una riflessione riguardante la collocazione di tale tipo di attività nel quadro degli studi musicologici e linguistici degli ultimi decenni.

Prima di passare alla presentazione dei contenuti della ricerca e dei risultati ottenuti, riteniamo opportuno riassumere e ribadire alcune premesse di carattere metodologico che consentono di definire con precisione le caratteristiche fondamentali della nostra attività e, nello stesso tempo, di inquadrarla nel vasto ambito dell' informatica musicale.

La nostra ricerca é stata ideata ed impostata sulla base di un fondamentale presupposto di carattere semiotico, ossia sull' idea che la musica possa essere concepita come uno strumento di comunicazione, come un " linguaggio ", per il quale siano identificabili convenzioni grammaticali e stilistiche ben precise. Questo presupposto é frutto, da un lato di una lunga tradizione di analisi musicale, dall' altro di una relativamente recente elaborazione teorica, soprattutto in relazione al complesso problema dei rapporti con la linguistica moderna [2]. La semiotica musicale, tuttavia, non ha ancora raggiunto un assetto scientifico maturo ed é tuttora alla ricerca di propri fondamenti teorici e di proprie metodologie.

Finalizzate a questo scopo sono, perciò, numerose indagini a carattere di sperimentazione, tra le quali si colloca la nostra attività che ha scelto come proprio un campo di indagine parallelo a quello della linguistica generativa. Il nostro obiettivo tuttavia non é quello di verificare l' applicabilità del modello teorico di Chomsky al linguaggio musicale, quanto piuttosto quello di sperimentare in generale la possibilità di descrivere il linguaggio musicale in termini di un numero di regole generative che noi chiameremo " grammatica ". Esse pur non costituendo una grammatica nel senso formale del termine, possono essere utili per gettar luce sulle strutture e sulle caratteristiche del linguaggio musicale che, tutto sommato, risultano a tutt'oggi abbastanza poco note.

Per realizzare il nostro obiettivo abbiamo scelto come oggetto di analisi un corpus limitato e omogeneo, costituito dai Corali a quattro voci di J.S.Bach, che presenta caratteristiche di particolare generalità e consente di affrontare prospettive di studio molto vaste circa l' uso dell' armonia tonale, l' influenza dell' armonia sull' invenzione melodica e i problemi relativi alla funzione sintattica della melodia quand' essa nasce come voce autonoma o quando invece presenta rapporti di dipendenza dalle altre voci.

Lo scopo che ci prefiggiamo é individuare un insieme di regole per la strutturazione di un Corale. In questo contesto l' elaboratore elettronico viene utilizzato come strumento di verifica della completezza e non contraddittorietà del sistema di regole ipotizzato.
I prodotti musicali generati automaticamente vengono sottoposti alla " competenza musicale " dell' ascoltatore analista il quale, mediante il confronto con il corpus originale, li riconoscerà come sufficientemente corretti o meno. Quest' ultimo giudizio é un test della validità dell' insieme di regole che costituiscono la grammatica.

Esaurita questa breve introduzione di carattere generale, passiamo ad analizzare più in dettaglio le caratteristiche del progetto di grammatica generativa introdotto in precedenza. Esso si articola in due parti: la prima riguarda

lo studio di una grammatica generativa delle melodie di soprano dei Corali; la seconda affronta il problema dell' armonizzazione di una melodia data, l' analisi della condotta della parte del basso e delle voci intermedie.
I due programmi di studio sono stati sviluppati separatamente e soltanto adesso che la ricerca ha raggiunto un livello abbastanza soddisfacente é stato possibile tentare di riunirne e coordinarne i risultati.

Del primo programma, relativo all' elaborazione di una grammatica generativa delle melodie di soprano, si sono già dati resoconti in precedenti convegni[1, 3]. Per questo ci limiteremo qui a richiamarne brevemente le linee essenziali.

Nella sua versione attuale la grammatica non solo é in grado di descrivere la struttura di una singola frase, ma anche di individuare gli elementi essenziali che fanno dell' intera melodia un' entità organica, che si sviluppa conservando un senso di unità complessiva.

Per illustrare le caratteristiche principali della grammatica é opportuno analizzare la sua applicazione diretta alla generazione.
Il procedimento inizia con la definizione di uno o più nuclei primitivi. Da un nucleo primitivo (Fig. 1), costituito da un intervallo che fissa il rapporto di altezza tra la prima nota in tempo forte e il tempo forte conclusivo della frase si sviluppa una " scala nucleare ", tramite il collegamento tra le due note con gradi congiunti intermedi. Ad essa può venir applicata una certa serie di regole di trasformazione, opportunamente codificate, ottenendo così un insieme di " frasi primitive ". Da queste ultime vengono generate, tramite una serie di operazioni, anch' esse codificate (come, per esempio, il taglio delle prime o delle ultime note), delle " frasi modificate ", che in sostanza costituiscono delle varianti con caratteristiche comuni, spesso facilmente riconoscibili, della stessa frase. Queste frasi devono poi subire degli adattamenti che consentono di ottenere il coordinamento con le altre frasi del Corale (sistemazione su date finali, modulazioni ecc.) e possono essere infine completate con l' ag-

giunta di eventuali fioriture.

Il grosso passo avanti rispetto alle linee di ricerca esplorate in passato (4) consiste nell' aver concepito una grammatica che non solo si preoccupa di definire le regole " minute " e le relazioni reciproche tra le note delle singole frasi, ma é in grado di riconoscere e controllare un insieme di vincoli e parentele che regolano la struttura globale della composizione e che l' ascoltatore recepisce riconoscendo nella melodia, come abbiamo detto, un senso di unitarietà. E' immediato comprendere che queste parentele tra frasi superficiali distinte derivano dall' aver avuto in comune qualche fase del processo generativo descritto in precedenza (nucleo, scala nucleare ecc.) : due frasi ottenute da nuclei diversi sono totalmente prive di vincoli di parentela, mentre due frasi che hanno avuto lo stesso albero generativo sono gemelle, cioé totalmente identiche.

Il sistema delle relazioni parentali che lega le frasi, prima ancora della loro completa generazione, é stato denominato " albero delle analogie formali ".

In Fig.2-a é mostrato un esempio di albero delle analogie formali per una melodia generata automaticamente che é riportata in Fig.2-b. In Fig.3 sono presentati altri esempi di melodie generate automaticamente.

Veniamo ora alla descrizione delle caratteristiche del secondo progetto, riguardante il problema della formulazione di un insieme di regole per l' armonizzazione di una melodia di Corale.

E' necessario dire subito che questo secondo settore di ricerca é molto più " giovane " dell' altro, precedentemente illustrato; tuttavia l' esperienza passata e i risultati acquisiti nell' ambito dello studio della melodia di soprano hanno permesso di ottenere già risultati di un certo interesse. Per motivi di semplicità si é ristretto in questa fase iniziale il corpus da analizzare alle singole frasi di Corali in tonalità maggiore non modulanti.

Partendo dall' esame delle successioni armoniche del campione abbiamo cercato anzitutto una conferma delle tradizionali teorie della funzione armonica.

Questo ha permesso di classificare gli accordi utilizzati in tre categorie:

area funzionale di TONICA	I, VI	(accordi di primo livello)
area funzionale di DOMINANTE	V, V^7, VII	(accordi di secondo livello)
area funzionale di SOTTODOMINANTE	II, II^7, IV	(accordi di terzo livello)

Gli accordi del primo livello hanno funzioni di stabilità: occupano i punti cardine della sequenza (inizio, fine e punti di riposo intermedi); gli accordi del secondo livello hanno la funzione di condurre all' area di tonica (le settime si trovano in cadenza*); gli accordi del terzo livello hanno la funzione di condurre all' area di tonica o a quella di dominante (le settime si trovano in cadenza).

Questa classificazione non esaurisce tutte le possibilità offerte dal campione: altri accordi (come quelli costruiti sul terzo grado o altre settime) pure presenti, anche se più raramente, risultano possedere un ruolo più superficiale e obbediscono a regole di transizione armonica meno rigide. Essi sono stati distinti col termine di "accordi di transizione" e assegnati ad un quarto livello.

La regolarità di questo comportamento ci ha consentito di concepire una elementare grammatica generativa dell' armonia del campione basata sullo stesso principio generale che regola la grammatica della melodia (e che in parte si ispira anche a principi di tipo shenkeriano) ; essa ipotizza l' esistenza di una struttura armonica superficiale da cui é possibile isolare livelli via via sempre più profondi e più fondamentali fino ad arrivare ad un denominatore comune che é l' accordo di tonica, che risulta il nucleo armonico fondamentale poiché contiene in sé il "germe" su cui viene sviluppata la melodia, cioé il suo carattere tonale.

Una volta accettata questa strutturazione a livelli si può, nella fase di analisi, costruire un possibile albero armonico, come mostrato in Fig.4 su esempi presi dal corpus di Bach. Esaminando questi alberi si vede che a partire dal livello fondamentale, in cui si fissano gli accordi di tonica della struttura, si passa al secondo livello in cui vengono

sono eventualmente inseriti gli accordi dell' area di dominante e, successivamente, al terzo livello che corrisponde alla sistemazione degli accordi dell' area di sottodominante. Al quarto livello, che é il più superficiale, vengono eventualmente inseriti gli accordi di settima e/o i raddoppiamenti degli accordi dei livelli precedenti.

La grammatica generativa allora, una volta assegnata la melodia del soprano, é in grado di costruire un possibile albero armonico, assegnando alle note corrispondenti ai tempi principali della battuta (quattro per una battuta di 4/4) la possibilità di portare un accordo. In particolare, per ognuno dei primi tre livelli dell' albero armonico, in successione, si selezionano alcune note della melodia stessa che dovranno portare accordi con quella data funzione armonica (Fig.5).

In questo modo ad un certo numero di note della melodia sono associati dei possibili accordi che svolgono le funzioni armoniche principali. A questo punto però ad alcune note della melodia che si trovano sui tempi principali possono non essere ancora associati accordi possibili. In questo caso si inseriscono i possibili accordi di transizione.

Si é trovato conveniente, tuttavia, lasciare la possibilità di inserire in queste posizioni non solo accordi di settima e raddoppiamenti, ma qualsiasi accordo compatibile con una certa tabella delle transizioni armoniche possibili. Questa tabella traduce un insieme di regole, piuttosto complesse, dedotte sia dalla tradizione della scuola tonale sia, soprattutto, dall' esame del campione. Naturalmente durante la scelta di queste armonie viene sempre fatta attenzione alla necessità di preparare e risolvere le settime.

Un discorso a parte va fatto per la cadenza: dall' esame del campione appare evidente che anche la cadenza armonica, come quella melodica, é soggetta a regole molto rigide (alternanza delle successioni dominante-tonica e tonica-dominante negli ultimi due tempi di cadenza); inoltre la cadenza armonica é fortemente legata al tipo di cadenza melodica del soprano ed é quindi immediatamente identificabile quando sia assegnata la melodia. Di conseguenza, in fase generativa gli ultimi due tempi della cadenza armonica vengono fissati subi-

to tramite l' esame della cadenza melodica, compatibilmente con le regole indicate sopra.

La grammatica finora esaminata produce, come risultato finale, una certa serie di armonie per ogni nota principale della melodia (Fig.5); rimane il problema di scegliere una delle possibili successioni che possono potenzialmente derivare dallo stesso albero armonico.
Questo problema naturalmente é collegato alla determinazione delle altre tre parti del Corale e, in particolare, a quella del basso, dotata di maggiori legami con la struttura armonica. L' analisi del campione mostra infatti che la natura melodica del basso é diversa da quella del soprano: mentre la melodia del soprano risulta chiaramente concepibile come un itinerario articolato tra una nota iniziale e una finale, quella del basso mostra di possedere funzioni armoniche che interferiscono in modo a volte molto evidente sulla sua condotta.
L' interferenza tra la logica delle successioni armoniche e la logica della condotta del basso ha aspetti molto complessi. In particolare é necessario chiarire se la parte di basso é concepibile come una successione di note, subordinata alla struttura melodico-armonica e regolata da relazioni soltanto locali tra le singole note o, piuttosto, se essa mantiene effettivamente la connotazione di una melodia, con una struttura autonoma e con delle connessioni non soltanto locali.

Un' attenta analisi di queste questioni sul corpus dei Corali di Bach ci ha condotto ad una soluzione che consente di mantenere la complessa rete di interconnessioni tra successioni armoniche e basso, senza subordinare i problemi generali di sviluppo dell' armonia e della melodia del basso a problemi locali di compatibilità, assumendo che il basso rimanga sostanzialmente una parte cantabile, cioé mostri seppur in forma più complessa e articolata, strutture generali analoghe a quelle della melodia del soprano.
Questa é un' ipotesi di lavoro costruita a partire dall' analisi, la cui validità può essere valutata sulla base dei risultati finali.

A partire da questi presupposti ci é sembrato allora opportuno concepire per la melodia di basso un insieme di regole strutturali dello stesso tipo di quelle adottate per il soprano. Tuttavia, a differenza di quanto accade per quest' ultimo, il prodotto della grammatica per il basso deve essere sottoposto, ad ogni stadio, a controlli di compatibilità con il contesto musicale già esistente, che consiste di una struttura melodica e dell' insieme delle possibili armonie previste dall' albero armonico. Nel caso in cui questa compatibilità si realizzi, le scelte effettuate per il basso comportano via via una selezione degli accordi teoricamente previsti dall' albero per la determinazione delle successioni armoniche.

Come nel caso della melodia, il primo stadio del procedimento generativo consiste nella definizione di un nucleo. Una volta scelta la cadenza del basso tra le formule cadenzali compatibili con la cadenza armonica, la prima nota di cadenza viene presa come seconda nota del nucleo della melodia del basso, mentre la prima nota del nucleo viene scelta a caso e assegnata, come basso corrispondente, alla prima nota in tempo forte del soprano. In questa prima fase del progetto abbiamo previsto la generazione di bassi che procedono per quarti.

Definito il nucleo, si fissa la scala nucleare associando le corrispondenti note del basso a note di soprano comprese tra la prima nota in tempo forte e la prima di cadenza. Il basso viene poi completato con una serie di trasformazioni scelte a caso tra quattro possibili, analoghe a quelle definite per il soprano[1]:

(a) ripetizione
(b) volta
(c) appoggiatura
(d) figura di salto

Alcuni risultati di questo insieme di regole sono mostrati in Fig.6, assieme alle successioni armoniche ottenute per selezione, come discusso in

precedenza partendo da melodie di soprano delle prime frasi dei Corali.

Fissato in questo modo il basso, risulta abbastanza semplice costruire le voci intermedie, a motivo dei vincoli imposti dalle due parti di soprano e di basso (assenza di quinte e ottave parallele, limiti di estensione ecc.) e delle possibilità di sviluppo dell' armonia associate all' albero armonico, diminuite notevolmente dalla scelta del basso.

Per lo sviluppo di queste parti non é ancora stata definita una procedura generativa. Per esse si ripresenta lo stesso problema discusso a proposito del basso e solo da un' attenta analisi sarà possibile decidere se é opportuno applicare una grammatica di tipo trasformazionale, oppure se la funzione di " riempimento " delle parti intermedie effettivamente priva le successioni di note di relazioni di tipo globale.

In Fig.7 sono mostrati i risultati ottenuti per l' intera frase di un Corale, completato, per quanto riguarda le parti intermedie, tramite una scelta casuale delle possibilità offerte dalle selezioni armoniche.

A questo punto é possibile riunire i due progetti e finalmente confrontare da un punto di vista " grammaticale " i risultati ottenuti dall' insieme di regole generative per melodia, armonia, basso e voci intermedie con i Corali di Bach del campione (Fig.8).

I risultati, seppur preliminari, ci sembrano abbastanza positivi e confermano l' idea che una grammatica di tipo trasformazionale, opportunamente concepita sulla base degli stessi principi di carattere generale, possa essere in grado di generare sequenze sia melodiche che armoniche dello stesso tipo di quelle del campione analizzato.

Certo il progetto relativo alla struttura armonica e alla struttura delle parti intermedie necessita di ulteriori raffinamenti: per esempio per quanto riguarda il basso, é necessario analizzare con più attenzione le relazioni che intercorrono tra la cadenza e la restante parte della melodia. Anche lo studio delle parti intermedie é ancora in fase iniziale.

Risolti questi problemi altri orizzonti sempre più vasti si aprono: basti citare il problema dell' armonia estesa all' intero Corale, includendo la possibilità di modulazioni, oppure il problema dell' analisi delle analogie che intercorrono tra le varie frasi della melodia del basso, qualora si estenda l' analisi all' intero Corale.

Dai primi e modesti risultati che abbiamo brevemente presentato, tuttavia, emergono alcune significative conferme delle ipotesi di base dell' intera ricerca e della validità delle metodologie che abbiamo scelto, e questo ci sostiene nella speranza di poter affrontare con successo anche i problemi più complessi di cui abbiamo dato cenno.

BIBLIOGRAFIA

1) M.Baroni : Studi su una grammatica musicale , in Atti del Convegno "Musica ed Elaboratore elettronico", Milano (1980).

2) J.J.Nattiez : Fondaments d' une sémiologie de la musique , Union Générale d' Editions, Paris (1973).

3) M.Baroni, C.Jacoboni : Analysis and generation of Bach' s choral melodies , in Acts de 1^{e}Congress International de Semiotique Musicale, Belgrado (1973) ;
M.Baroni, C.Jacoboni : Analogie formali in frasi diverse di una composizione monodica , in Atti del 3^{o} Colloquio di Informatica Musicale, Padova (1979).

4) M.Baroni, C.Jacoboni : Verso una grammatica della melodia , Quadrivium XVII, 2 (1976).

* Negli studi precedenti sulla melodia si é trovato conveniente definire "cadenza" l' insieme delle note collocate sugli ultimi tre quarti della melodia stessa. Tale definizione é stata provvisoriamente utilizzata anche nella restante parte del progetto, sebbene in certi casi risulti evidente la necessità di una diversa, più funzionale definizione di cadenza armonica.

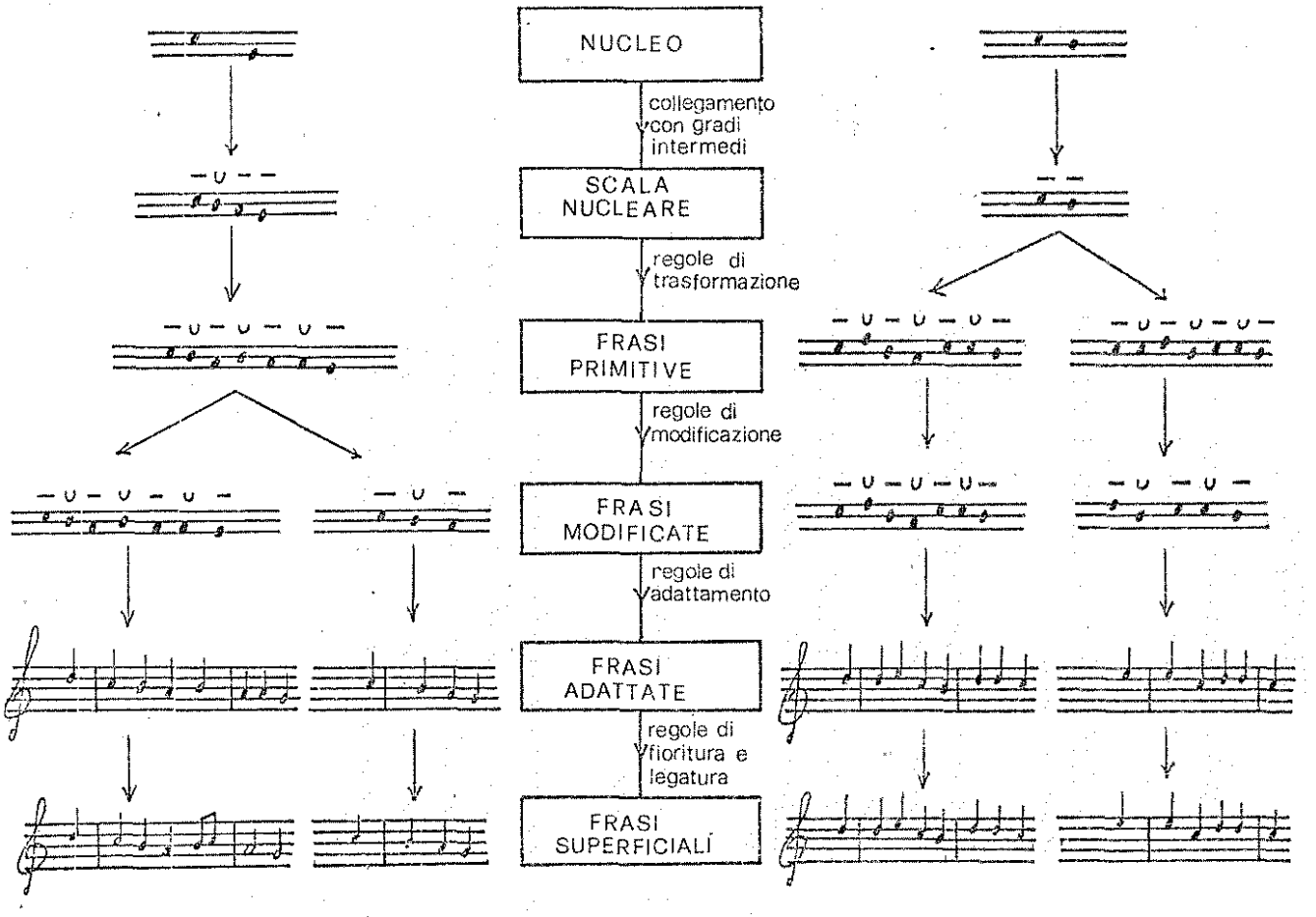
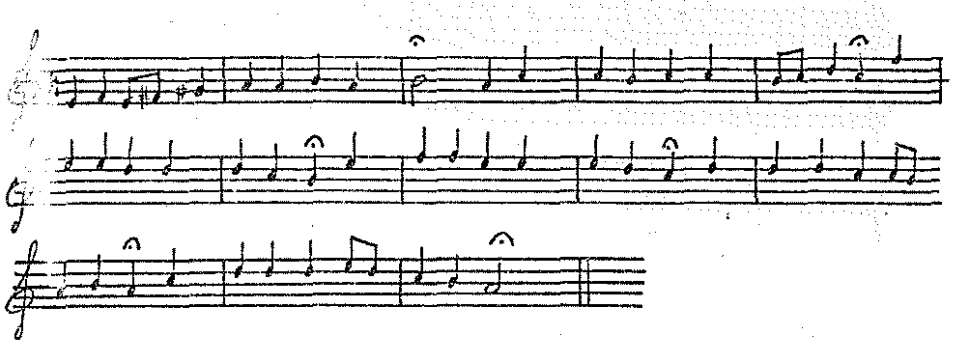
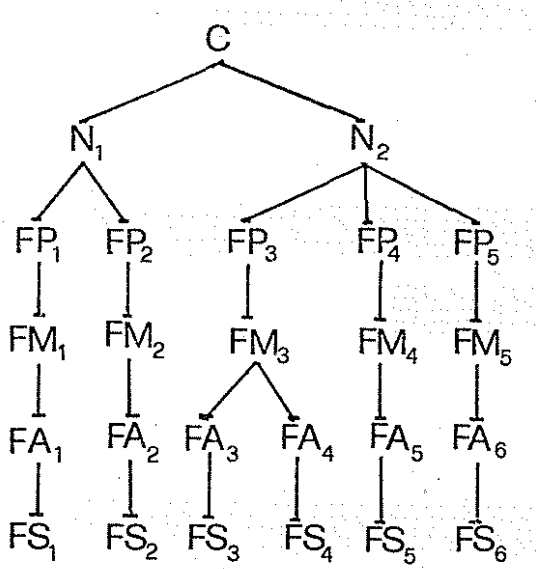


Fig.1: Procedimento generativo di melodie di frasi di soprano. A sinistra é riportato un esempio di applicazione a frasi tratte dal corpus dei Corali bachiani e a destra a frasi generate automaticamente. I simboli – e ∪ stanno a indicare la posizione metrica rispettivamente forte e debole della nota.



(a)



(b)

Fig.2: Melodia generata automaticamente (a) e relativo albero delle analogie formali (b).

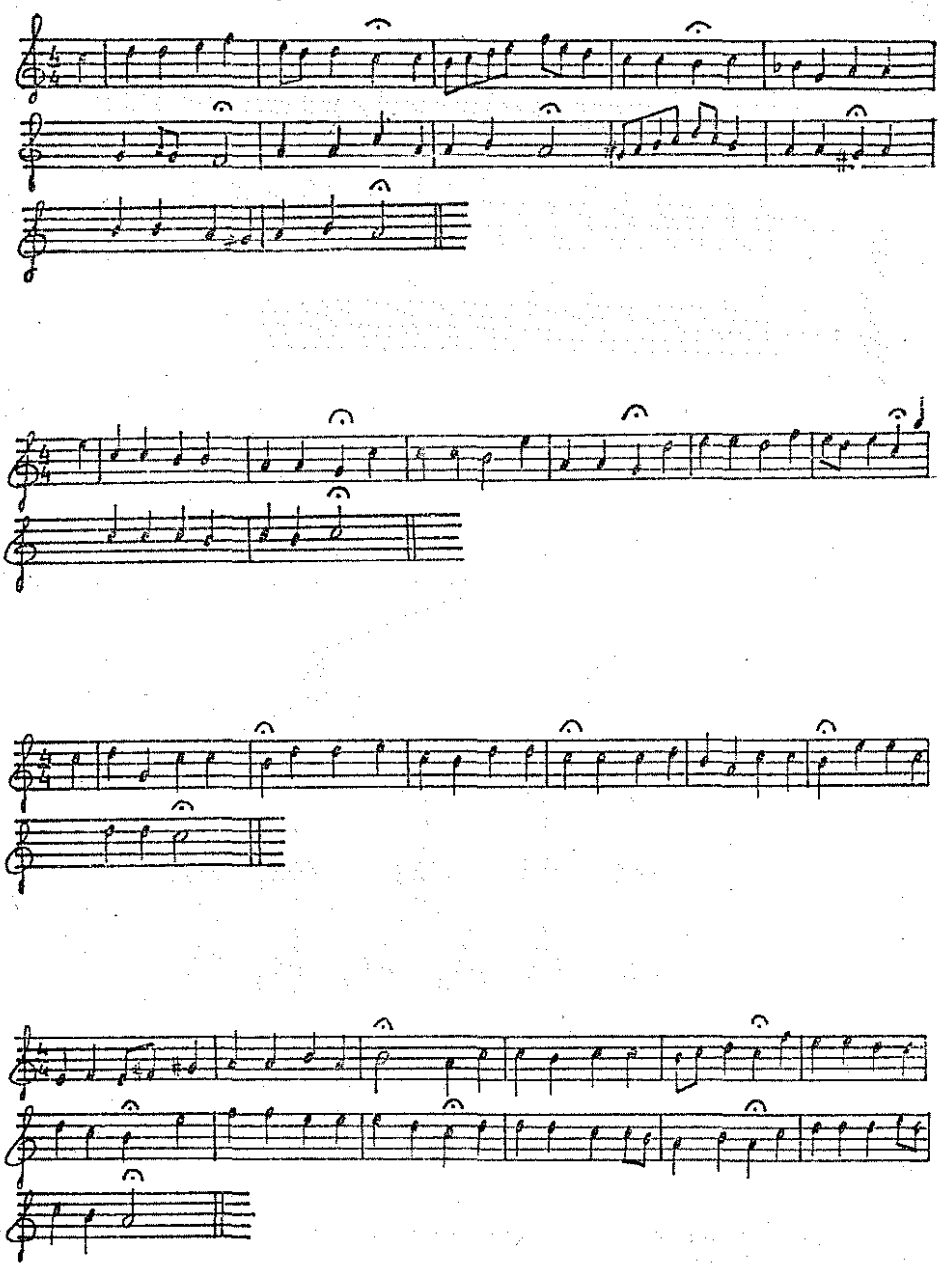


Fig.3: Esempi di melodie generate automaticamente.

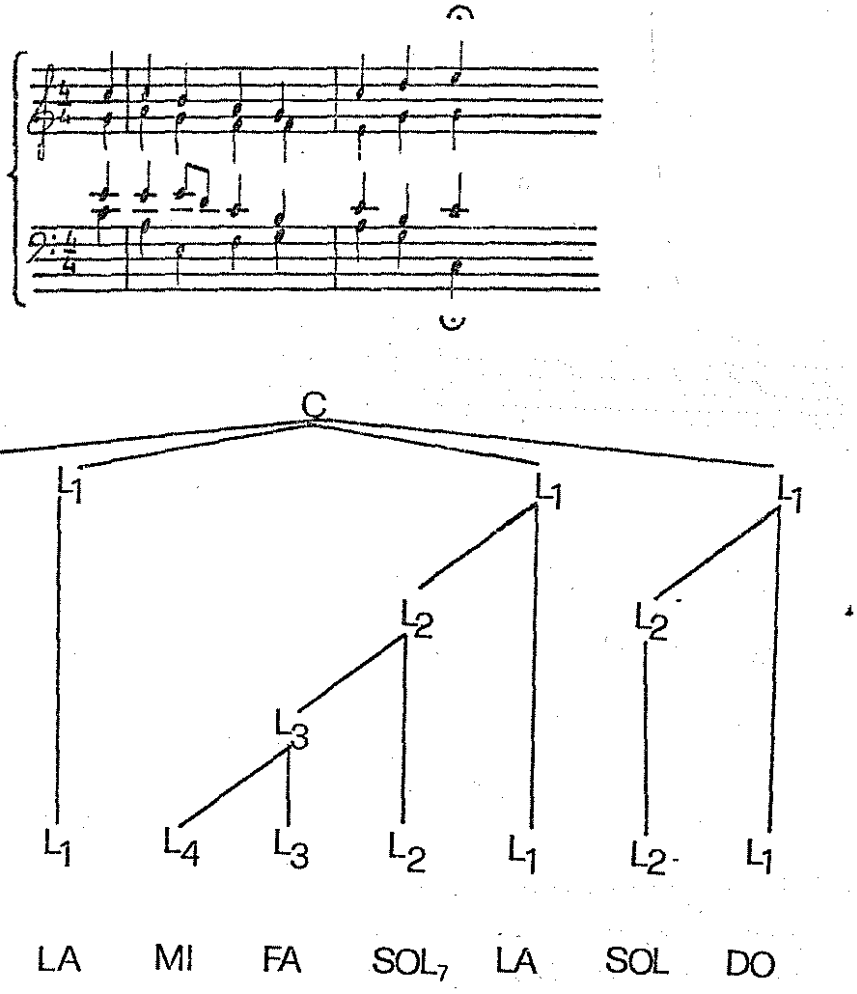


Fig.4: Albero della struttura armonica di un Corale del campione bachiano; il Corale (trasportato in DO maggiore) é riportato nella parte superiore della figura. L_1, L_2, L_3, L_4 indicano i successivi livelli funzionali discussi nel testo.

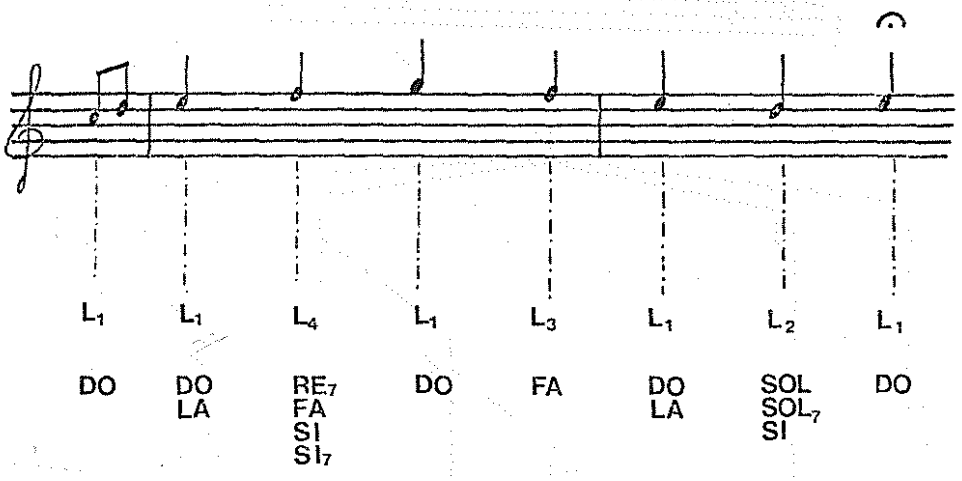


Fig.5: Risultato della assegnazione automatica dei livelli armonici alle note di una melodia del campione. Sotto ad ogni livello sono indicati i possibili accordi compatibili con i livelli assegnati, con le note della melodia e con il complesso delle regole armoniche che costituiscono il modello teorico assunto.

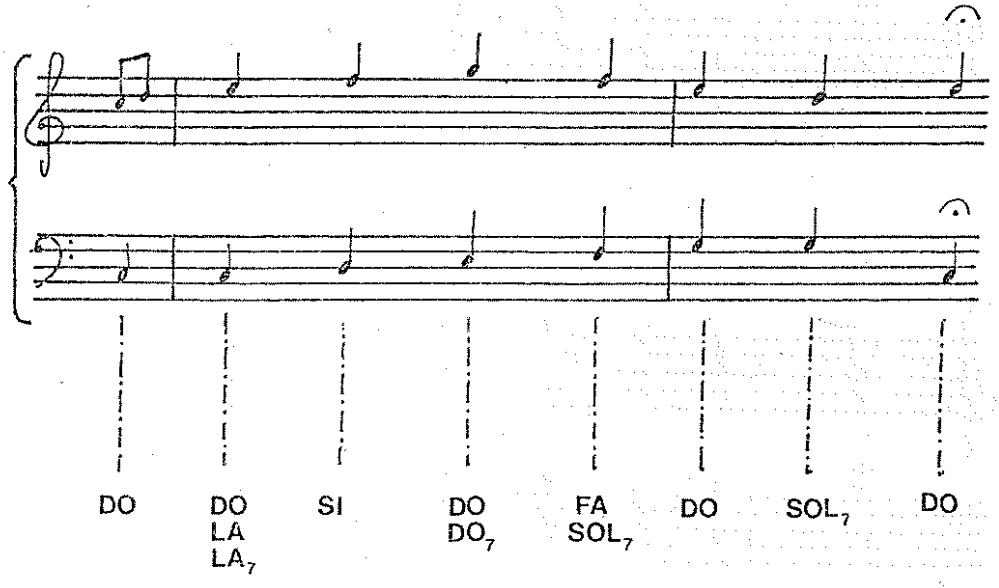


Fig.6: Risultato della assegnazione automatica dei livelli armonici e della melodia del basso per una melodia del campione. Gli accordi riportati sotto ad ogni nota principale del soprano risultano compatibili con i livelli assegnati e il complesso delle regole armoniche, nonché con le note della melodia e del basso. Questo esempio di generazione é indipendente da quello in Fig.5.

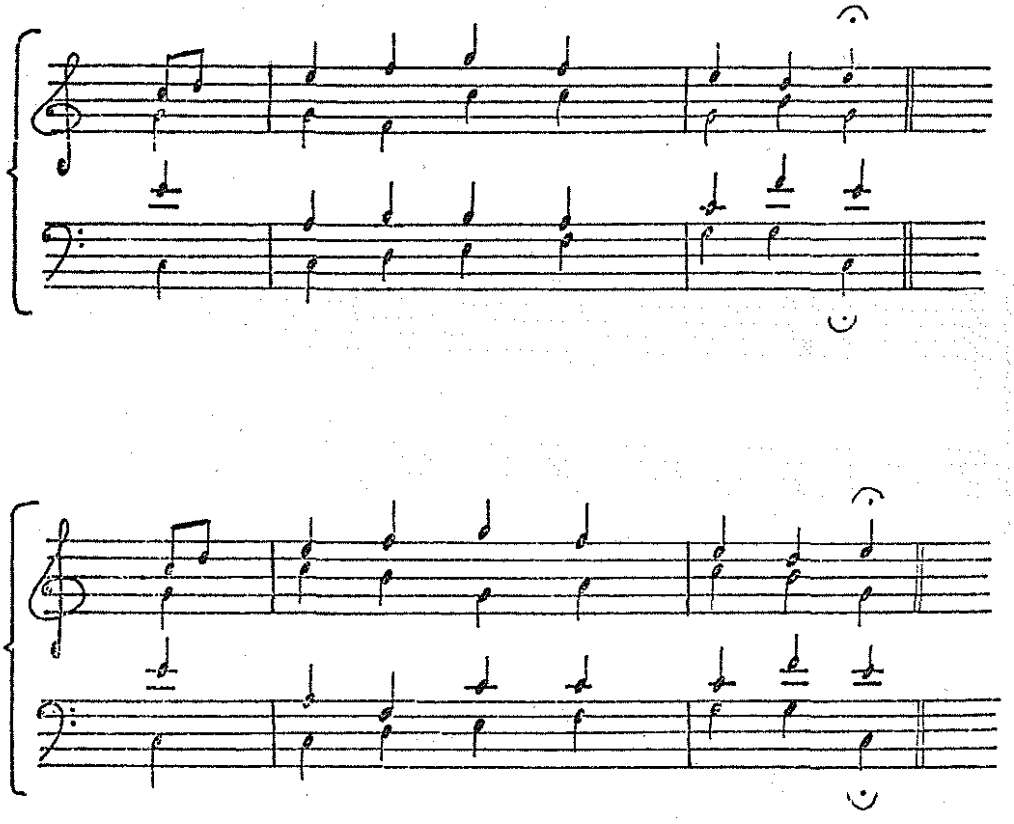


Fig.7: Risultato della assegnazione automatica delle parti intermedie per una melodia del campione (ved.testo). I due esempi sono stati ottenuti a partire dalla generazione del basso riportato in Fig.6.

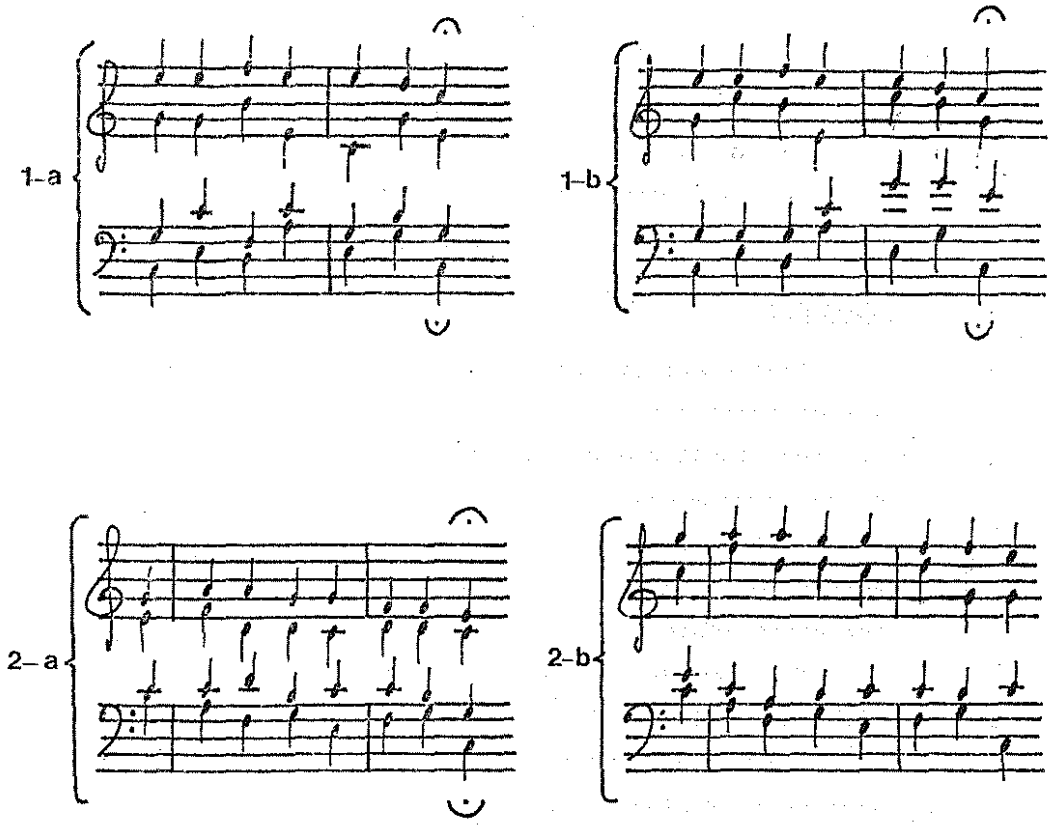


Fig.8: Due diversi risultati per la assegnazione automatica della melodia del basso e delle parti intermedie per ciascuna di due melodie generate automaticamente sulla base delle regole della grammatica generativa per la melodia (ved. testo).

A SYSTEM FOR THE GENERATION OF FOUR VOICE CHORALE STYLE COUNTERPOINT USING ARTIFICIAL INTELLIGENCE TECHNIQUES

Alberto Maria Segre

Department of Electrical Engineering
University of Illinois
Urbana, Illinois 61801, U.S.A.

and

Istituto di Cibernetica
Università degli Studi di Milano
Via Viotti 5
20133 Milano, Italy

ABSTRACT

Given a set of pieces in four part chorale style, the system creates a network of assertions through the analysis of these input pieces that can then be used in the creation of new counterpoint. The generation of new pieces uses a state space representation technique which reduces the generation of counterpoint to a series of searches through the state space.

The system consists of a set of programs written in LISP on a UNIVAC 1100. The LISP interpreter actually used is the MACCLISP interpreter written at the University of Wisconsin (Madison, Wisconsin, U.S.A.).

RIASSUNTO

L'applicazione della tecnica di rappresentazione degli spazi al problema della generazione di testi musicali viene fatta da una serie di programmi realizzati in LISP. I programmi hanno il compito di analizzare i testi musicali dati come esempi da emulare e poi di generare nuovi testi usando le informazioni ricavate durante la fase di analisi.

Support for this project came in the form of a Fulbright grant in Music Composition.

INTRODUCTION

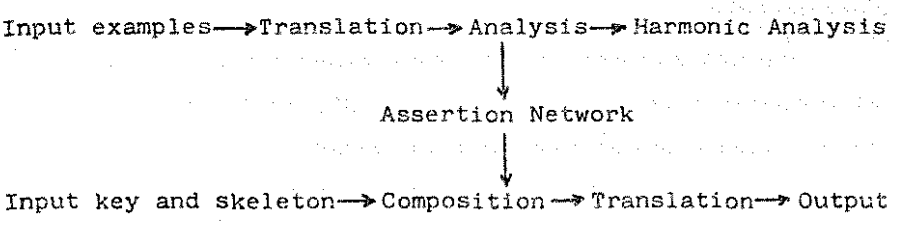
The system described in this paper consists of a set of computer programs, written in LISP, that generate four voice chorale style counterpoint based on a set of assertions created by an analagous set of programs that analyze a pool of input pieces. The input pieces used in this case are from the "371 vierstimmige Choralgesänge" of Johann Sebastian Bach (1685-1750), although any similar pieces would do as well.

The main idea behind a project of this sort is to create a new piece, in this case in a highly structured form such as the four part voice leading of Bach's era, without explicitly setting the parameters for the generation of the new piece. In other words, the parameters for the new composition are actually set by another set of programs that "learn" the rules for this type of counterpoint by examining a set of examples. This is exactly the same manner that a student composer learns from studying the works of other composers. The composition routine must then set a rather loose framework for the application of the knowledge gathered in the analysis phase.

The above mentioned works by Bach were chosen because they tipify a highly structured style, thus simplifying the evaluation of the quality of the computer produced compositions. The general approach should be valid for almost any musical style period, provided the proper modifications are made to overcome the limitations imposed by certain assumptions that had to be made during the development of the system.

OVERVIEW AND RATIONALE

The general scheme of this analysis-generation system is as shown below:



Note that the "center" of the system is the assertion network. The assertion network functions as a bank of information containing all of the details regarding the construction of the input pieces. These input pieces serve essentially as a learning base for the composition routine.

According to the above diagram, it is possible to divide the task of the analysis-composition programs into the following sub-tasks:

1) translation of the input pieces into some form that can be used to construct the assertion network.
2) analysis of the translated input (including the construction of the assertion network). The harmonic analysis is essentially a by-product of this sub-task.
3) composition of new pieces based on the input key and skeleton provided by the user.
4) translation of the newly generated counterpoint into some easily readable form.

Throughout the development of this system, a conscious effort was made to inject the minimum possible author bias with respect to the musical contents. There are however a certain number of assumptions that had to be made which, by influencing the analysis and generation procedures, might have some bearing on the musical contents of the generated pieces.

1) It is assumed that the input pieces all have a certain harmonic rhythm, and that particular harmonic rhythm is taken as the pulse of the piece. For example, a piece in six eight time might have a pulse of two or three beats per measure depending on that piece in particular.
2) It is also assumed that the input pieces all have four and only four voices (this number is rather arbitrary but was chosen to comply with the Baroque norm) and furthermore that the lowest or bass voice was the most important from the harmonic standpoint. This could be changed to give the predominant role to any voice, but again a choice was made on the basis of the Baroque (or Thorough-Bass period) norm.
3) It is further assumed that the pieces are composed of phrases and that each phrase has certain important structural characteristics. Furthermore it is assumed that the first and last chords of each phrase are the most important from the structural point of view.
4) It is also assumed that the cadential gesture at the end of each phrase is an important harmonic element.
5) It is assumed that the system operates in a strict major-minor tonality where all major keys are essentially identical, as are all minor keys.
6) Finally, it is assumed that the generation of new counterpoint will take place in a harmony preferred rather than a melody preferred manner. In other words, the chord progression will be determined first and then the individual voices will be filled in. It should be relativly simple to modify the system in order to switch these two modes of operation, keeping in mind that the proper modifications must be made to the analysis program in order to retain more melody based harmonic information rather than the currently retained harmony based melodic information.

THE INPUT LANGUAGE

The pieces that are to be used as the basis for the assertion network created by the analysis program are coded in a LISP type input language. The following is a brief overview of the basic format of this input coding.

The basis of the input coding is the LISP list. An entire piece of music (in this case a four voice chorale) is therefore represented by a construction consisting of nested lists. The general form is:

(<title> <keysignature> <voice 1> ... <voice n>)

where in this case 'n' will always be four. Each element of the above construct is in turn represented by a list, an atomic symbol, or a character string.

<title> := character string, ex. "JESU MEINE FREUDE"

<keysignature> := list, (<flats> <sharps>)

<flats> := list, (<notename> ...)

<sharps> ex. ((B) NIL) for F major
(NIL NIL) for C major
(NIL (F C G)) for F sharp minor

<voice n> := list, (<beat 1> ... <beat m>)

<beat m> := list, (<subbeat 1> ...) OR
(<notename/modifier 1> ...) OR
a mixed version of the above two.

<subbeat m> := has the same form as <beat n>

<notename> := one of nine possible atomic symbols
the seven notenames C D E F G A B
the two special symbols
* := retain previous note
R := rest

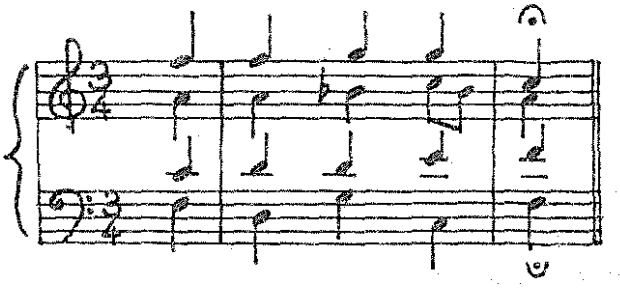
<modifiers> := + or # to indicate a sharp (carries thru measure).

- to indicate a flat (also carries)

$ to indicate a cadence point

N to indicate natural (also carries)

An example of this input coding is shown below:



becomes:

```
( "EXAMPLE" (NIL NIL)
      ((F)(F G G)(C$))
      ((A)(A B- (C B))(A$))
      ((C)(D * E)(F$))
      ((F)(D G C)(F$))  )
```

CODED REPRESENTATION

The internal representation of the input piece is based on a note numbering system that spans from zero to eleven. After the key of the piece has been determined, the piece is "transposed" such that the tonic is considered the "zeroeth" degree. The other numbers run up the chromatic scale. Therefore, if the piece is in C major:

C = 0; C+ = D- = 1; D = 2; D+ = E- = 3; E = 4; F = 5;
F+ = G- = 6; G = 7; G+ = A- = 8; A = 9; A+ = B- = 10;
B = 11.

If the key had been E minor, then E would have been set to zero, and so on up the scale.

The representation of chords is based on the specification of the chord type (i.e. quality) and the root of the chord. The root of the chord is indicated using the same note numbering system described above. The chord type is determined with a matching routine by the analysis program. Possible chord types are:

MAJOR; MINOR; DIM; AUG indicate major, minor, augmented and diminished triads.

DOM; MAJMAJ; MINMIN; HDIM; FDIM indicate dominant, major-major, minor-minor, half-diminished and fully-diminished seventh chords.

Therefore, a chord that consists of (5 9 0) would yield (MAJOR 5), while (4 7 11) gives (MINOR 4). Here are some more examples:

(DOM 7) = a dominant seventh chord on the dominant.

(MAJMAJ 2) = a major triad with a major seventh on the super tonic.

(AUG 11) = an augmented triad on the leading tone.

There are two special cases in relation to chord representation. Should the chord contain an insufficient amount of information to allow a unique identification (specifically if there are only two distinct degrees due to voice doublings or rests) then the chord will be represented with (UNREC LENGTH x y) where x and y are the degrees that are specified. This representation shall be treated just like any other chord since it shall be unique for any set of two degrees x and y.

The other special case is when the degrees included in the chord fail to match any expected chord type. In this case, the chord is identified with either an

(UNREC SEVENTH w x y z) or an (UNREC TRIAD x y z) depending on whether the unrecognized "cluster" had three or four distinct notes. As before, these are treated just like an "identifiable" chord since these are also unique identifiers for given degrees w, x, y and z.

THE ANALYSIS PROGRAM

The task of the analysis program is to take the coded input pieces, construct the chords from the individual voices, put the chords through a matching routine that gives each distinct chord a unique identifier, and then construct the assertion network.

This task is relatively straight forward, the only real difficulty is presented in chord construction when there is more than one note in a particular voice, such as in the case of a passing seventh. In this case, the matching routine returns a composite chord identifier. For example, given a major triad on the dominant with a passing seventh in one of the voices, the matching routine will return the composite identifier ((MAJOR 7)(DOM 7)) which is not the same as either of the two individual chords. The parentheses are nested in the same manner that they were nested in the moving voice. If more than one voice moves simultaneously, then the identifier becomes more complex. It is assumed that this is not the norm, and should there be too many of these complex identifiers it would be wise to check the harmonic rhythm of the piece.

The assertion network per se is nothing more than a construction of LISP properties. Therefore, the analysis program constructs the network through a series of LISP PUT and GET commands. The exact form of the assertion network is described below.

THE ASSERTION NETWORK

The analysis program has the task of extracting

assertions for the assertion network from the pool of input pieces. The extracted assertions must retain all of the characteristics of the input pieces, and therefore must function at a general as well as a detailed level.

Throughout this discussion notation will be used that is consistent with the implementation. This notation is basically of the form:

(<item> . <occurance index>) OR (<item> . <cost>)

(this is a LISP cons-ed pair). The occurance index (or cost) is simply an indication of the number of times the analysis routine added that particular assertion to the assertion network. Therefore, an item that occured more frequently will have a higher occurance number or cost. The term "cost" is used in order to comply with the more common search terminology and should not be taken to imply that a lower cost is somehow "better".

As mentioned before, the assertions function at several different levels, or more particularly; the structural level (dealing with the overall characteristics of the piece), the phrase level (dealing with structural characteristics of the phrase), the chordal level (dealing with the characteristics of the chord progression), and the note level (dealing with the particulars of the voice leading).

The Structural Level:

Consists of LISP properties keyed to the key of the piece as transposed to the zeroeth degree (i.e. the tonic is always zero). The properties are BEGINS and ENDS and each consist of a list of the first (or last) chords of the input pieces as indexed by the key. For example, a piece beginning and ending on (MAJOR 0) in the key of (MAJOR 7) would yield the following assertion network:

```
(MAJOR 0)  BEGINS  ((MAJOR 0) . 1)
(MAJOR 0)  ENDS    ((MAJOR 0) . 1)
```

Note that the key name has also been transposed to the zeroeth degree. In the same manner, a piece in (MINOR 6) beginning on (MINOR 6) and ending on (MAJOR 6) (a raised or "piccardy" third on the final cadence) would add the following assertions to the assertion network:

```
(MINOR 0)  BEGINS  ((MINOR 0) . 1)
(MINOR 0)  ENDS    ((MAJOR 0) . 1)
```

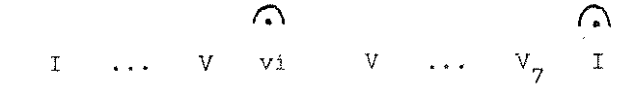
Suppose for the moment that the next piece analyzed was in the key of (MINOR 8) and that the piece began and ended on (MINOR 8). Now the status of the assertion network would be:

```
(MINOR 0)  BEGINS  ((MINOR 0) . 2)
(MINOR 0)  ENDS    ((MAJOR 0) . 1)((MINOR 0) . 1)
```

Therefore, not only does the number of assertions contained in the network continue to grow as the number of pieces analyzed increase, but the occurance indexes of each assertion also increase as the analysis program continues to add the same "fact" to the network.

Th Phrase Level:

Consists of the properties STARTS, CADENCES and PRECADENCES as keyed by the preceeding chord. For example, a piece of two phrases in length with the following harmonic structure:



that is:

```
(MAJOR 0) ... (MAJOR 7) (MINOR 9) (CADENCE)
(MAJOR 7) ... (DOM 7) (MAJOR 0) (CADENCE)
```

would yield the following assertion network:

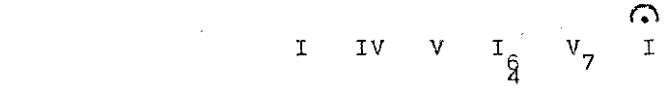
```
(MAJOR 0)  BEGINS       ((MAJOR 0) . 1)
(MAJOR 0)  ENDS         ((MAJOR 0) . 1)

(MAJOR 0)  CADENCES     ((MINOR 9) . 1)
(MAJOR 7)  PRECADENCES  ((MINOR 9) . 1)
(MINOR 9)  STARTS       ((MAJOR 7) . 1)
(MAJOR 7)  CADENCES     ((MAJOR 0) . 1)
(DOM 7)    PRECADENCES  ((MAJOR 0) . 1)
```

Note that each assertion is keyed to the structural chord that preceeds it; in other words, the CADENCES property lists the cadence chords of a phrase as keyed to the starting chord of that phrase, the PRECADENCES property lists the cadence chords of a phrase as keyed to the precadence chord, and the STARTS property lists the first chord of a phrase as keyed to the cadence chord of the preceeding phrase.

Note that the precadential chord is defined as the last distinct chord preceeding the cadence chord, and not just as the last chord preceeding the cadence.

The Chord Level:

Consists of the LEADS and the BASS properties. The LEADS property lists a chord as keyed to the chord immediatly preceeding it. For example, the following chord progression:



would yield the following additions to the network:

```
(MAJOR 0)  LEADS  ((MAJOR 5) . 1)((DOM 7) . 1)
(MAJOR 5)  LEADS  ((MAJOR 7) . 1)
(MAJOR 7)  LEADS  ((MAJOR 0) . 1)
(DOM 7)    LEADS  ((MAJOR 0) . 1)
```

Note that repeated sequences of chords would increase the related occurance indexes of the components of the progression in question.

The BASS property lists the notes appearing in the bass (i.e. lowest) voice of each chord as keyed to the chord name. Thus a dominant seventh type chord on the seventh degree in root position would add this assertion to the network:

```
(DOM 7)    BASS   (7 . 1)
```

while in first inversion it would give:

```
(DOM 7)    BASS   (11 . 1)
```

and so on.

The Note Level:

Consists of the properties LEADS and UNDERLIES. The LEADS property functions in analagous manner to the LEADS property at the chord level, except in this case it lists individual notes in the bass line as indexed by the note preceeding the note in question.

The UNDERLIES property lists the movement in the upper voices as indexed by the motion in the bass voice. The other voices are treated independently. For example, the following bass line:

```
5   7   0
```

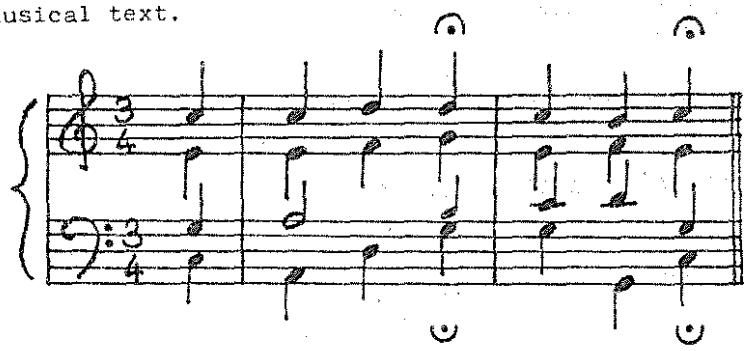
under the following upper line (which particular line is not that important):

```
2   2   7
```

would give:

```
(5)     LEADS     (7 . 1)
(7)     LEADS     (0 . 1)

(5 7)   UNDERLIES  ((2 2) . 1)
(7 0)   UNDERLIES  ((2 7) . 1)
```

The following is an example of the assertions added to the assertion network at all levels for a particular musical text.



Input Coding:

```
("INPUT EXAMPLE" (NIL NIL)
  ((C)(C D D$)(C B C$))
  ((E)(E F G$)(E F E$))
  ((G)(A * B$)(C D G$))
  ((C)(A D G$)(G G C$))  )
```

Translated:

```
soprano = ((0)(0 2 2$)(0 11 0))
alto    = ((4)(4 5 7$)(4 5 4$))
tenor   = ((7)(9 9 11$)(0 2 7$))
bass    = ((0)(9 2 7$)(7 7 0$))
```

Harmonic Analysis:

(MAJOR 7)

```
((MAJOR 0) (MINOR 9) (MINOR 2) (CADENCE)
 (MAJOR 0) (DOM 7) (MAJOR 0) (CADENCE))
```

```
Structural Level:

   (MAJOR 0)  BEGINS  ((MAJOR 0) . 1)
   (MAJOR 0)  ENDS    ((MAJOR 0) . 1)

Phrase Level:

   (MAJOR 0)  CADENCES     ((MAJOR 7) . 1)((MAJOR 0) . 1)
   (MINOR 2)  PRECADENCES  ((MAJOR 7) . 1)
   (MAJOR 7)  STARTS       ((MAJOR 0) . 1)
   (DOM 7)    PRECADENCES  ((MAJOR 0) . 1)

Chord Level:

   (MAJOR 0)  LEADS  ((MINOR 9) . 1)((DOM 7) . 1)
   (MINOR 9)  LEADS  ((MINOR 2) . 1)
   (MINOR 2)  LEADS  ((MAJOR 7) . 1)
   (DOM 7)    LEADS  ((MAJOR 0) . 1)

   (MAJOR 0)  BASS   (0 . 2)(7 . 1)
   (MINOR 9)  BASS   (9 . 1)
   (MINOR 2)  BASS   (2 . 1)
   (MAJOR 7)  BASS   (7 . 1)
   (DOM 7)    BASS   (7 . 1)

Note Level:

   (0)    LEADS  (9 . 1)($ . 1)
   (9)    LEADS  (2 . 1)
   (2)    LEADS  (7 . 1)
   (7)    LEADS  (0 . 1)($ . 1)(7 . 1)

   (0 9)  UNDERLIES  ((0 0) . 1)((4 4) . 1)((7 9) . 1)
   (9 2)  UNDERLIES  ((0 2) . 1)((4 5) . 1)((9 9) . 1)
   (2 7)  UNDERLIES  ((9 11) . 1)((5 7) . 1)((2 2) . 1)
   (7 $)  UNDERLIES  ((11 $) . 1)((7 $) . 1)((2 $) . 1)
   (7 7)  UNDERLIES  ((0 2) . 1)((4 5) . 1)((0 11) . 1)
   (7 0)  UNDERLIES  ((2 7) . 1)((5 4) . 1)((11 0) . 1)
   (0 $)  UNDERLIES  ((7 $) . 1)((4 $) . 1)((0 $) . 1)
```

COMPOSITION

The composition routine is based upon four separate applications of a search aimed at finding all possible solutions present in the search space.

The user must provide the composition program with the key and rhythmic skeleton of the new piece. In other words

for a four phrase piece in the key of F minor with each phrase of length eight (notated 8.8.8.8 in the chorale books), the user must supply the composition routine with (MINOR 5) and (8 8 8 8) upon prompting ("WHAT KEY?" and "WHAT SKELETON?", respectively).

After the skeleton and key have been provided, the composition routine begins a series of searches in the assertion network to come up with all of the possible solutions. Exactly which of the solutions will be chosen as the "best" solution can be determined in one of two ways:

1) the user can choose to pick his or her own "best" solutions, thus in effect creating a computer aided composition.
2) the user can allow the computer to pick the "best" solution based on the costs of the various components of a particular solution as operated upon by the detach functions. The four individual detach functions (one operating at each level) determine the interrelationships between the costs of the solutions at a particular level. The "best" solution is then that solution with the maximum cost. The default detach.functions are set to the LISP "PLUS" function, but the user can supply part or all of the detach functions.

The four searches take place on the four different levels. The first search finds all possible structures for a given key and skeleton. Each solution is then listed along with the cost for that solution (the sum of the costs of the individual assertions used in that solution). A solution with a higher cost indicates that the assertions present in that solution are relatively more frequent than the assertions in another solution with a lower cost. The search routine used is a breadth-first algorithm, since the depth of the search is a fixed value.

The second search takes place at the phrase level and supplies all of the possible chord progressions (as well as their respective costs). The third search provides all possible bass lines for a given chord progression (as well as the costs), while the fourth search provides all possible voice leadings for a given chord progression and bass line (again, with the proper costs).

The following is an example of the first search (i.e. at the top level, where the general structure of the piece is determined). Note that the component costs are summed to give the cost of each particular solution. This final cost is the cost that the detach functions operate on, along with the final costs from the other three searches.

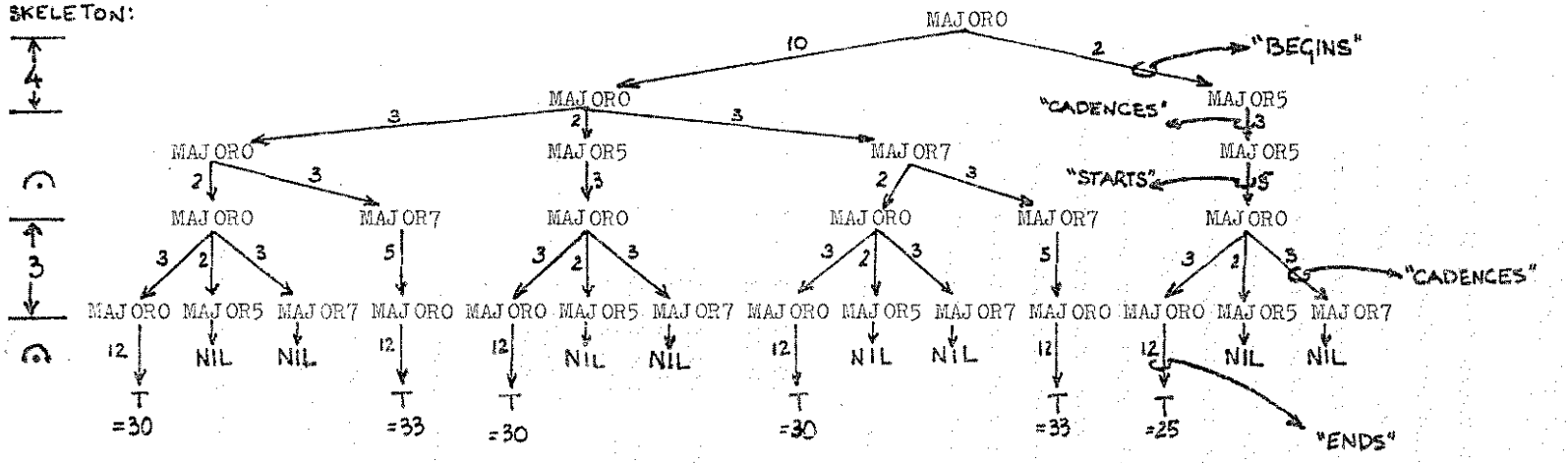
Given this assertion network:

```
(MAJOR 0)  BEGINS    ((MAJOR 0) . 10)((MAJOR 5) . 2)
(MAJOR 0)  ENDS      ((MAJOR 0) . 12)

(MAJOR 0)  CADENCES  ((MAJOR 0) . 3)((MAJOR 5) . 2)
                     ((MAJOR 7) . 3)
(MAJOR 7)  CADENCES  ((MAJOR 0) . 5)
(MAJOR 5)  CADENCES  ((MAJOR 5) . 3)

(MAJOR 7)  STARTS    ((MAJOR 7) . 3)((MAJOR 0) . 2)
(MAJOR 5)  STARTS    ((MAJOR 0) . 5)
```

and given that the user supplied:

```
WHAT KEY?:         (MAJOR 0)
WHAT SKELETON?:    (4 3)
```

the following search would take place:


SKELETON:
4
3
2
1
MAJ ORO
10
2
"BEGINS"
MAJ ORO
"CADENCES"
MAJ OR5
MAJ ORO
MAJ OR5
MAJ OR7
"STARTS"
MAJ OR5
MAJ ORO
MAJ OR7
MAJ ORO
MAJ OR7
MAJ ORO
"CADENCES"
MAJ ORO MAJ OR5 MAJ OR7 MAJ ORO MAJ ORO MAJ OR5 MAJ OR7 MAJ ORO MAJ OR5 MAJ OR7 MAJ ORO MAJ ORO MAJ OR5 MAJ OR7
NIL NIL NIL NIL NIL NIL NIL NIL
T T T T
=30 =33 =30 =30 =33 =25
"ENDS"

Thus the search procedure yields six possible solutions that are consistent with the information in the assertion network:

```
(MAJOR 0)   (MAJOR 0);  (MAJOR 7)   (MAJOR 0); cost = 33
(MAJOR 0)   (MAJOR 0);  (MAJOR 0)   (MAJOR 0);      = 30
(MAJOR 0)   (MAJOR 5);  (MAJOR 0)   (MAJOR 0);      = 30
(MAJOR 0)   (MAJOR 7);  (MAJOR 0)   (MAJOR 0);      = 30
(MAJOR 0)   (MAJOR 7);  (MAJOR 7)   (MAJOR 0);      = 33
(MAJOR 5)   (MAJOR 5);  (MAJOR 0)   (MAJOR 0);      = 25
```

The solutions that have higher costs are not necessarily any better than the solutions with lower costs, but they do have relatively more frequent components. The default detach functions would add these costs together with the costs of the other levels in order to determine the best solution (with the highest final cost). However, should the user wish to choose the least frequent solution components or emphasize a particular level, then he or she is free to supply his or her own detach function(s).

Following this search (at the first or top level) the composition program supplies the expanded phrases (i.e. with the chord progressions fleshed out) with a second search. It should be noticed that all of the solutions from the first level are expanded, that is the detach functions are applied at the end of all four searches, not as the composition routine goes along. This is necessary, since what may appear to be the best solution at one level may prove not to be so at the next level, or may even be completely unexpandable.

The second search uses the phrase level property LEADS, and also checks for the proper cadence formula at the end of each expanded phrase with the PRECADENCES property. For example, given that the following is a part of the assertion network:

```
(MAJOR 0)  LEADS  ((MAJOR 7) . 8)((DOM 7) . 4)
                  ((MINOR 6) . 3)
(MAJOR 7)  LEADS  ((MAJOR 0) . 8)
(DOM 7)    LEADS  ((MAJOR 0) . 3)((MAJOR 9) . 1)
(MINOR 6)  LEADS  ((MAJOR 0) . 1)((MINOR 2) . 2)

(DOM 7)    PRECADENCES  ((MAJOR 0) . 5)
```

and the following results from the first search (given that the key is (MAJOR 0) and that the skeleton is (3)):

```
((((MAJOR 0)(MAJOR 0)) . 10)
  ((MAJOR 5)(MAJOR 0)) . 13))
```

the result from this second search would be:

```
(((((MAJOR 0)(DOM 7)(MAJOR 0)) . 12) . 10)(NIL . 13))
```

Note that while the second solution from the first level had a slightly higher cost, and thus might appear to have been better that the first solution, it proved to be unexpandable later. The cost of the second level solution (=12) comes from the sum of the component properties, namely the application of two LEADS and one PRECADENCES assertions.

Had the assertion:

```
(MAJOR 7)  PRECADENCES ((MAJOR 0) . 3)
```

also been included in the assertion network then the solution would have been:

```
((((MAJOR 0)(DOM 7)(MAJOR 0)) . 12)
 (((MAJOR 0)(MAJOR 7)(MAJOR 0)) . 19) . 10)(NIL . 13))
```

The solution that will be considered "best" depends upon the interrelationships between the costs at various levels as determined by the detach functions. In the default

situation, where all of the detach functions are set to the LISP "PLUS" function, the second solution would be chosen since 10 + 19 = 39 is greater than 10 + 12 = 22, but this may not be the case if the user chooses to provide a different set of detach functions. In either case, no choice would take place until all four levels had been expanded, since what may be the "best" solution at one stage may not be expandable in succeeding stages.

The third search will provide all possible bass lines for each phrase in the solution set from the second search. This search uses the BASS and LEADS (as applied to bass notes) properties.

Given a chord progression, it is easy to determine which possible bass notes can be used for the first chord by looking at the BASS property of that first chord. Once the first chord has a bass note(s) (it is possible for each chord progression to have more than one solution starting on the same or different bass note) then the search proceeds by examining both the LEADS property of the preceeding bass note as well as the BASS property of the approaching chord.

The fourth search then generates the upper voices by applying the UNDERLIES property to all possible bass lines and chord progressions from the preceeding searches. The search insures that no illegal motion occurs in the upper voices (as measured against the bass) since such motion could not appear in the UNDERLIES list. In addition, the fourth search checks to make sure that the upper voices conform to the chord progression.

It is at this point that the detach functions come into play making a choice as to which solution is the "best" solution.

PERSPECTIVES

The scope of the system is still relatively small. It should be possible to expand the system to allow more flexibility in the style of the learning input, by increasing the number of voices allowed and allowing both harmony and melody preferred generation of new pieces.

It might also be possible to create some sort of hybrid system that reconciles both harmony based and melody based generation simultaneously. Another possible addition to the system would allow differentiation of register (as of now, register information is not even coded in the input language), or perhaps it might be feasable to increase the interdependance of the voices at generation time.

One interesting by product of this kind of project is that by inspecting the assertion networks after the seperate analysis of works by two different composers the differences in the assertion networks should directly reflect the style differences of these two composers. This may seem trivial, but may prove useful in some musicology related study.

One supposes that should this system ever be perfected and given all of Beethoven's works it should be able to provide the user with the tenth symphony, just as Beethoven would have written it. However, it is also said that given enough monkees and enough typewriters the researcher will eventually get Shakespeare's "Hamlet" in return. For now, this author will be satisfied by generating good, solid four voice counterpoint!

ACKNOWLEDGEMENTS

The author would like to thank the following people who helped at some point during the project:

Goffredo Haus for his guidance throughout;
Alberto Bertoni for his help with the theoretical aspects of fuzzy probability;
Gianni Degli Antoni and his Istituto di Cibernetica for their support; and
Piercarlo Grandi for his help with an (at times) reluctant UNIVAC.

BIBLIOGRAPHY

Bach, J. S.; "371 vierstimmage Choralegesänge," (Edition Breitkopf nr.10), Wiesbaden, GDR.

Forte, A.; "Tonal Harmony in Concept and Practice," Second Edition, (Holt, Rinehart and Winston), N.Y, N.Y., U.S.A. (1974).

Kling, R.; "Fuzzy-Planner: Reasoning with Inexact Concepts in a Procedural Problem Solving Language," JOURNAL OF CYBERNETICS,4 p.105-122, (1974).

Lefaivre, R.; "The Representation of Fuzzy Knowledge," JOURNAL OF CYBERNETICS,4 p.57-66, (1974).

Nilsson, N.; "Problem Solving Methods in Artificial Intelligence," (McGraw-Hill), N.Y., N.Y., U.S.A., (1971).

Norman, E.; "Lisp Primer," Academic Computing Center, The University of Wisconsin - Madison, Madison, WI., U.S.A.

Norman, E. and Lefaivre, R.; "Lisp Reference Manual," Academic Computing Center, The University of Wisconsin - Madison, Madison, WI., U.S.A.

Salzer, F. and Scachter, C.; "Counterpoint in Composition," (McGraw-Hill), N.Y., N.Y., U.S.A. (1969).

Schoenberg, A.; "Manuale di Armonia," Il Saggiatore Editore, Milano, Italy (1963).

Smoliar, S.W.; "A Parallel Processing Model of Musical Structures," Massachusettes Institute of Technology, U.S.A. (1971).

Winston, P.H.; "Artificial Intelligence," (Addison Wesley) Reading, MA., U.S.A. (1977).

Zadeh, L.; "Fuzzy Sets," INFORMATION AND CONTROL,8 p.338-353 (1965).

Zadeh, L.; "Fuzzy Algorithims," INFORMATION AND CONTROL,12 p.94-102 (1968).

M.Bartolini - G.Berardo C.R.A.U.S. Bologna/Messina

MUSICA E PERSONAL COMPUTER

L'avvento del Personal computer ha reso possibile la computerizzazione musicale a basso costo e con linguaggi facili. Alcuni esempi vengono presentati adoperando l'Apple II plus e il Music Synthesizer Alf che permette la scrittura(sul video) e la composizione con due potenziometri esterni.
Con detto sistema si realizza musica prevalentemente melodica, mentre invece, tramite tastiera, adoperando un fondamentale programma "Chroma", è possibile generare suoni e musica elettronica, anche randomizzata.

MUSIC AND THE PERSONAL COMPUTER

The invention of the Personal computer has made possible the computerization of music at a low cost and with a simple language. Some examples are presented using the Apple II plus and the Music Synthesizer Alf which permit the writing (on video) and the composition with two external potentiometers.
With this system mainly melodious music is obtained, while in fact, by means of the keyboard, using a "Chroma" basic program, it is possible to create electronic sounds and music, even randomized.

M.Bartolini,G.Berardo

MUSICA E PERSONAL COMPUTER

Non è nostro compito sintetizzare la storia,non recentissima, dell'uso del calcolatore elettronico nel campo musicale anche perchè l'avvento del Personal computer sta modificando le varie ricerche.
Annotiamo,invece, le varie possibilità di computerizzazione con l'uso del Music Syntnesizer Alf dell'Apple II:
1 - Scrittura (sul video,anche a colori) e composizione adoperando due potenziometri all'esterno del calcolatore(cfr."The Apple Music Synthesizer Owner's Manual" - Alf Products Inc. 1448 Estes Denver,CO 8215, Usa);
1.1 Modifiche dell'originale programma Lens adoperando l'Integer Basic,come illustriamo nel listing(tav.1)
a) Il nostro intervento é avvenuto sostanzialmente dalla linea 0 alla 1500, mutando l'esecuzione dei tempi,che compongono il programma, determinatamente(abbreviando i tempi di esecuzione di ogni pezzo) e casualmente(randomizzando le tre variabili N,S, T della linea 1100).
Ne consegue una differenziazione "determinata" e "casuale" nell'ascolto della composizione originale;
b) Il programma è registrato su floppy disc.Per l'esecuzione occorre un fondamentale programma "CHROMA" (BAGHS, Piazza Costituzione 8/3,Palazzo Affari,Bologna),un amplificatore esterno ed un altoparlante,collegati al computer.
2.1 Originale programma "B & B",ottenuto con la tastiera del computer,opportunamente adoperando la tastiera(l'Integer basic è il linguaggio fondamentale) ed il succitato programma "Chroma", che va dalla linea 32767 della tavola 1 di 1.1.
Il programma verrà eseguito come dimostrazione.Non trascriviamo il programma che presenta le stesse modalità di registrazione e di ascolto delle quali in 1.1 b).
Di ambedue le composizioni abbiamo visualizzato,con numeri,i tempi al fine di individuare meglio l'algoritmo.

CONCLUSIONI

E' sufficiente fare notare che il suddetto Personal computer permette, a basso costo, la realizzazione di ciò che era alquanto complesso e dispendioso non molti anni fa.
Ma non possiamo sottovalutare l'utilità didattica di un tale modo di operare sia per il programmatore , come per quanti si oc-

cupano della manipolazione dei suoni o della musica computeriz_
zata in genere.

Bologna Aprile 91-M.Bartolini,G.Berardo

del "Centro ricerche sulle attività umane su_
periori"(CRAUS) di Messina-Bologna

BOLOGNA,Oberdan 15,40126

```
>LIST LENS B%B
*** SYNTAX ERR
>LIST
    0 DIM A$(112)
    1 CALL -936: REM  INTEGER "HOME"
    5 PRINT "  L E N S"
    6 PRINT : PRINT "VARIANTI RANDOM DI BARTOLINI & BERARDO"
   10 SLOT=4
   11 J=0
   15 B=0:N0=0:N1=0:N2=0:R0=0:R1=0:R2=0
   16 A=B-S-N0-N1-N2-R0-R1-R2
   20 GOSUB 32767:CHANNEL=PITCH+1:OFFSET=PITCH+3:CHROMA=PITCH+7:VOL1=VOL0+
      1:VOL2=VOL0+2
   25 GOSUB 1000
   26 VTAB (J*2+4)
   29 PRINT N,S,T
   30 POKE PITCH,N: POKE CHANNEL,0: CALL CHROMA: POKE CHANNEL,1: CALL CHROMA:
       POKE CHANNEL,2: CALL CHROMA
   40 FOR A=50 TO 255: POKE VOL0,A: POKE VOL1,A: POKE VOL2,A: FOR B=0 TO 14
      : NEXT B,A
   50 N0=N*256:N1=N0:N2=N1:R0=(10240-N0)/S:R1=(19968-N1)/S:R2=(24576-N2)/S
   60 FOR A=1 TO S:N0=N0+R0:N1=N1+R1:N2=N2+R2: POKE CHANNEL,0: POKE PITCH,
      N0/256: POKE OFFSET,N0 MOD 256: CALL CHROMA
   70 POKE CHANNEL,1: POKE PITCH,N1/256: POKE OFFSET,N1 MOD 256: CALL CHROMA:
       POKE CHANNEL,2: POKE PITCH,N2/256: POKE OFFSET,N2 MOD 256: CALL CHROMA
   80 FOR B=0 TO (S+5-A)/50: NEXT B,A
   90 A$="<225466036506:036546103656618Z>4066<2254660365:660365663<265663820666
      6620666642<526622:5066>165<56:125<568105063"
  100 FOR B=1 TO 9*T: NEXT B: FOR A=1 TO 112 STEP 7: FOR C=0 TO 2: POKE CHANNEL
      ,C
  110 POKE PITCH, ASC(A$(A+C+C))+16* ASC(A$(A+C+C+1))-2992: CALL CHROMA: NEXT
      C
  120 FOR B=1 TO ( ASC(A$(A+6))-176)*T-30: NEXT B,A
  130 FOR A=255 TO 50 STEP -1: POKE VOL0,A: POKE VOL1,A: POKE VOL2,A: FOR
      B=0 TO 25: NEXT B,A
  140 GOTO 15
 1000 IF J<>0 THEN GOTO 1030
 1010 N=72:S=512:T=150: GOTO 1500
 1030 IF J<>1 THEN GOTO 1060
 1040 N=7:S=512:T=150: GOTO 1500
 1060 IF J<>2 THEN GOTO 1090
 1070 N=72:S=51:T=150: GOTO 1500
 1090 IF J<>3 THEN GOTO 1100
 1093 N=72:S=512:T=15: GOTO 1500
 1100 N=7+ RND (72):S=50+ RND (512):T=15+ RND (150)
 1500 J=J+1: RETURN
32767 REM   CHROMA   ALF PRODUCTS INC.
65531 PITCH= PEEK (76)+176:VOL0= PEEK (77)-2+PITCH/256:PITCH=PITCH MOD 256
      : IF VOL0<128 THEN PITCH=VOL0*256+PITCH
65532 IF VOL0>127 THEN PITCH=(VOL0-128)*256+PITCH-14304-18464: POKE PITCH+
      62, PEEK (76): POKE PITCH+66, PEEK (77)-2
65533 POKE PITCH+2,16*SLOT: POKE PITCH+3,0
65534 VOL0=16*SLOT-16256: POKE VOL0,0: POKE VOL0+1,0: POKE VOL0+2,0: POKE
      VOL0+3,3: POKE VOL0+7,54: POKE VOL0+7,118: POKE VOL0+7,182: RETURN
65535 REM , _ % HIMEM: HIMEM:617514387824O)kukn OR ha#[ZTKNhH SGN C!= LEN(
      45821d-q((40821k=36368 LOAD 34327;4778543348 PLOT 35413,410356518638228
       INPUT HhPx1 INPUT 34384 END 1544 INPUT , TAB , END , INPUT 616321681
       TAB 5128025940 END 36945 _ f INPUT JP' 11 INPUT SGN ,H LOMEM: INPUT
      209014928440274268164105621681514S6>=59662>=hPw REM  REM  REM  REM
65535 REM , HLIN 45480 INPUT 3489751796 SGN CON ; END ,J HIMEM:z HLIN HIMEM:
      58882 END 37301 INPUT H% END 4104421681p,5947434387 TAB 41553 RUN 1717
```

```
LOAD MUSIC                                                    TAV.II
LIST

000  POKE 770,173: POKE 771,48: POKE 772,192: POKE 773,136: POKE 774,208:
     POKE 775,4: POKE 776,198: POKE 777,1: POKE 778,240
010  POKE 779,8: POKE 780,202: POKE 781,208: POKE 782,246: POKE 783,166: POKE
    784,0: POKE 785,76: POKE 786,2: POKE 787,3: POKE 788,96
020  POKE 1,0
030  GET A$: POKE 0,( ASC (A$) - 32) * 4: CALL 770: GOTO 1030

LOAD BER(CHROMA)
LIST
    5 CALL -936
   10 SLOT=4
   20 GOSUB 32767:PART=PITCH+1:OFFSET=PITCH+3:CHROMA=PITCH+7:WIDTH=PITCH+4

   21 FOR J=1 TO 3
   22 A=0:B=255
   25 FOR I=1 TO 255:A=A+1:B=B-1:C= RND (255)
   30 POKE PART,0: POKE PITCH,A: CALL CHROMA: POKE VOL0,255: POKE WIDTH, RND
      (100)
   40 POKE PART,1: POKE PITCH,B: CALL CHROMA: POKE VOL0+1,255
   50 POKE PART,2: POKE PITCH,C: CALL CHROMA: POKE VOL0+2,255
   52 VTAB (1): PRINT "VARIAZIONI CHROMA DI BERARDO & BARTOLINI": PRINT : PRINT
      : PRINT
   54 PRINT A,B,C
   55 NEXT I: NEXT J
   59 GOTO 10
   60 END
32767 REM    CHROMA    ALF PRODUCTS INC.
55531 PITCH= PEEK (76)+176:VOL0= PEEK (77)-2+PITCH/256:PITCH=PITCH MOD 256
      : IF VOL0<128 THEN PITCH=VOL0*256+PITCH
55532 IF VOL0>127 THEN PITCH=(VOL0-128)*256+PITCH-14304-18464: POKE PITCH+
      62, PEEK (76): POKE PITCH+66, PEEK (77)-2
55533 POKE PITCH+2,16*SLOT: POKE PITCH+3,0
55534 VOL0=16*SLOT-16256: POKE VOL0,0: POKE VOL0+1,0: POKE VOL0+2,0: POKE
      VOL0+3,3: POKE VOL0+7,54: POKE VOL0+7,118: POKE VOL0+7,182: RETURN
55535 REM h
10240;36888 USER (8240)kukn OR ha#[ZTKNhH SGN C!= LEN(45821d-q((40821k=36368
       LOAD 34327;4778543348 PLOT 35413,410356518638228 INPUT HhPx1 INPUT
      34384 END 1544 INPUT , TAB , END , INPUT 616321681, TAB 5128025940 END
      36945 _ f INPUT JP' 11 INPUT SGN ,H LOMEM: INPUT 20901492844027426816
      4105621681514S6>=59662>=hPw REM  REM  REM  REM
55535 REM , HLIN 45480 INPUT 3489751796 SGN CON ; END ,J HIMEM:z HLIN HIMEM:
      58882 END 37301 INPUT H% END 4104421681p,5947434387 TAB 41553 RUN 1717
```

IL SISTEMA INFORMATIVO DEL LABORATORIO DI INFORMATICA MUSICALE DELLO ISTITUTO DI CIBERNETICA DI MILANO.

Goffredo Haus - Istituto di Cibernetica - Università di Milano

Riassunto.

In questo lavoro viene descritto il sistema informativo del Laboratorio di Informatica Musicale dell'Istituto di Cibernetica di Milano.

E' basato sull'uso del DMX-1000 Digital Signal Processor ed è costituito da moduli software corrispondenti alle diverse funzionalità ed ai diversi livelli di rappresentazione dell'informazione musicale.

Il sistema è orientato alla elaborazione interattiva di testi musicali (analisi, trasformazione, sintesi) con la possibilità di eseguire testi musicali in tempo reale per mezzo del DMX-1000.

I linguaggi in cui è stato implementato il sistema sono il Pascal (per la elaborazione a livello simbolico) ed il COBOL (per la gestione delle librerie, la microprogrammazione del DMX-1000 e la comunicazione dati).

Abstract.

In this paper we describe the computing system of the Laboratory of Musical Informatics of the Institute of Cybernetics of Milan.

It has DMX-1000 as digital signal processor and a set of software modules corresponding to functionalities and levels of representation of musical data.

The system is oriented to interactive musical texts computing and allows to exucute musical texts by means of DMX-1000 in real time.

Symbolic computing has been implemented in Pascal and libraries handlers, DMX-1000 microprogramming, data communications in COBOL.

INTRODUZIONE.

Nell'ambito dell'attività di ricerca di informatica musicale, lo Istituto di Cibernetica ha costituito un Laboratorio per la sperimentazione delle metodologie per la descrizione formale e la trasformazione dei testi musicali e per la sperimentazione di tecniche per la sintesi digitale del suono.

I criteri che hanno ispirato le scelte della strumentazione e dell'architettura del sistema di elaborazione sono stati orientati alla ricerca di soluzioni flessibili e di interesse generale che, allo stesso tempo, permettessero di effettuare la verifica acustica dei procedimenti definiti, sviluppati e sperimentati (a livello di descrizione simbolica) nel corso delle precedenti ricerche.

In particolare, la descrizione strutturale funzionale (detta 'ad operatori') (Bertoni et al., 1978) e la trasformazione (Haus, 1979) dei testi musicali saranno oggetto di verifiche e miglioramenti al fine di costituire un sistema efficiente per la manipolazione di testi musicali a diversi livelli di astrazione.

Inoltre sarà dedicata una particolare attenzione a sistemi di elaborazione musicale ad intelligenza distribuita, o, in altre parole, a sistemi multiprocessore con distribuzione periferica di processori dedicati.

IL LABORATORIO.

Il Laboratorio ha come strumento a cui fa riferimento tutta l'attività il DMX-1000 Digital Signal Processor della Digital Music Systems.

Con questo apparecchio è possibile implementare qualsiasi tecnica per la sintesi digitale del suono: è infatti completamente microprogrammabile ed esegue molto velocemente (200 nanosecondi per ogni microistruzione) i microprogrammi producendo segnali analogici in uscita dai convertitori digitali/analogici di cui è dotato.

E' perciò necessario controllare il DMX-1000 con un elaboratore general purpose; è stato scelto a tal fine un'architettura hardware più complicata di quanto sia strettamente necessario (vedi Figura 1).

Un elaboratore Honeywell I.S.I. Livello 62/40 comunica con un sistema a microprocessore SDK 80 che a sua volta comunica con il DMX-1000.

La scelta di interporre l'SDK 80 ha due motivazioni principali: è così possibile demandare all'SDK 80 alcune funzioni, tra cui quella di temporizzare i processi musicali in corso di esecuzione, e inoltre il Livello 62/40 non consente di emettere dati nel formato a 8 bit richiesto dall'interfaccia seriale del DMX-1000.

L'SDK 80 quindi riceve i dati dal Livello 62/40, li prepara per il DMX-1000, li organizza con una struttura a lista (permettendo così una gestione dinamica della sua memoria) e li invia al DMX-1000 al momento opportuno.

Il collegamento tra Livello 62/40 e SDK 80 è seriale mentre quello tra SDK 80 e DMX-1000 è parallelo; il DMX-1000 ha la possibilità di generare interrupts sull'SDK 80.

Il primo obiettivo dell'attività del Laboratorio sarà di verificare le possibilità di prestazioni di questo sistema sviluppando un package di software orientato sia ad applicazioni in tempo differito, sia ad applicazioni di editing in tempo reale, sia ancora ad applicazioni di performance musicale in tempo reale.

Più in generale, gli obiettivi complessivi dell'attività del Laboratorio riguarderanno i seguenti temi, come già accennato in (Haus, 1980b):

a) sviluppo di procedure software per l'elaborazione di testi musicali a partire da una forma di descrizione simbolica gerarchica e compatta; tali procedure permetteranno anche l'elaborazione interattiva in tempo reale;

b) studio di architetture di sistemi per l'elaborazione di processi musicali basati sull'uso di microprocessori (in particolare sistemi multiprocessore);

c) progettazione, sviluppo e sperimentazione di prototipi di dispositivi per la sintesi digitale del suono, con particolare attenzione alla qualità delle prestazioni ed al contenimento dei costi;

d) conversione di packages di software musicale tra diversi elaboratori.

Per il conseguimento di questi obiettivi ci si avvarrà anche di altre apparecchiature non dedicate all'attività del Laboratorio; ad esempio, collegamenti a reti di calcolatori, uso di altri elaboratori, etc.

Quando sarà completata la fase di sviluppo del software di base del sistema basato sul DMX-1000, saranno effettuati presso il Laboratorio sia seminari informativi che di addestramento all'uso delle apparecchiature ed in generale dei sistemi in dotazione al Laboratorio stesso.

Analizziamo quindi brevemente quali sono le caratteristiche più peculiari del sistema in via di sviluppo descrivendo dapprima il DMX-1000 e le modalità di microprogrammazione del DMX-1000 e nel paragrafo successivo l'architettura del software del sistema ed i comandi già implementati.

IL DMX-1000: ARCHITETTURA GENERALE E MICROPROGRAMMAZIONE.

Il DMX-1000 è un minicomputer ultraveloce a 16 bit appositamente progettato per l'elaborazione dei segnali audio; è collegabile ad un altro elaboratore che assume funzioni di 'master' e da cui viene visto come un terminale intelligente; può essere collegato come periferica a molti tra i più comuni elaboratori e microelaboratori (è interfacciabile con S-100, RS232C, UNIBUS, porte parallele programmabili).

Il 'master' controlla il DMX-1000 mediante microprogrammazione; fornisce un microprogramma e un insieme di dati al DMX-1000 a cui fa seguire una sequenza di updates; il DMX-1000 esegue ripetutamente il microprogramma 'attuale' in memoria e produce un campione ad ogni esecuzione per la conversione digitale/analogica.

L'alta velocità del DMX-1000 è in parte dovuta al parallelismo delle operazioni controllate durante l'esecuzione dei microprogrammi.

Tutte le operazioni sono compiute con una precisione di 16 bit; in particolare, le moltiplicazioni danno un risultato di 32 bit.

L'architettura del DMX-1000 è illustrata in Figura 2 (Wallraff, 1980a).

La CONTROL MEMORY è la memoria che contiene il microprogramma che determina l'operatività del DMX-1000; il microprogramma è costituito da una sequenza di microistruzioni (da un minimo di 10 ad un massimo di 256) di 32 bit ciascuna; queste vengono eseguite in sequenza poichè non esistono microistruzioni di salto; il tempo di esecuzione è uguale per ogni tipo di microistruzione ed è di 200 nanosecondi.

L'unità aritmetico-logica (ALU) esegue operazioni logiche ed aritmetiche; contiene 17 registri a 16 bit utilizzabili durante l'esecuzione di microprogrammi; l'ALU esegue una operazione ogni ciclo-macchina, cioè ogni 200 nsec.

La DATA MEMORY è una memoria di 4096 parole di 16 bit destinata a contenere i dati (che possono essere parametri, come la frequenza di un

oscillatore, o variabili di stato, come la fase attuale di un oscillatore, o tabelle che descrivono forme d'onda).

Il MULTIPLIER esegue moltiplicazioni con segno di due numeri di 16 bit complementati a due; il risultato (32 bit) viene caricato nei due registri d'uscita L e M.

Il convertitore digitale/analogico (DAC) genera una tensione proporzionale al numero caricato nell'apposito registro; il tempo d'azione del DAC è di 50 cicli-macchina per campione; per questo è opportuno, a tutto vantaggio della precisione, non superare una velocità di campionamento di 25 KHz.

I filtri hanno una frequenza di cutoff (modificabile) di 9,6 KHz.

Il DMX-1000 del Laboratorio è dotato di due DAC.

L'attività del DMX-1000 è diretta dall'elaboratore 'master' mediante i seguenti cinque comandi:

'0' : ferma l'esecuzione del microprogramma permettendo l'aggiornamento delle memorie;

'1' : determina l'esecuzione del microprogramma per una volta;

'2' : ordina l'esecuzione ripetuta del microprogramma; in questa condizione permette un solo update per ogni esecuzione;

'3' : carica o aggiorna una locazione di DATA MEMORY;

'4' : carica o aggiorna una locazione di CONTROL MEMORY.

La struttura generale di un microprogramma è data in Figura 3 (Wallraff, 1980b). Prima di eseguire la prima microistruzione, il microprogramma ha tre cicli speciali: L1 e L2 (durante i quali il 'master' può effettuare un update nella CONTROL MEMORY o nella DATA MEMORY) e PRE (durante il quale la prima microistruzione che deve essere eseguita viene portata nel MPR (microinstruction pipeline register)).

L'esecuzione vera e propria del microprogramma inizia perciò nel quarto ciclo-macchina; non esistendo microistruzioni di salto, l'esecuzione del microprogramma continua fino a che viene incontrata la prima microi-

struzione HALT presente nella CONTROL MEMORY; il ciclo-macchina successivo sarà L1, seguito da L2, etc.

La velocità di campionamento viene determinata dividendo i 5 MHz del clock per l'indirizzo della prima microistruzione HALT più 4; la minima velocità di campionamento, cioè quando l'HALT è all'indirizzo 255, è di 19,3 KHz.

Il formato delle microistruzioni è mostrato in Figura 4; i codici operativi delle microistruzioni hanno le seguenti funzioni:

A, B: selezionano i registri, uno per campo, che l'ALU usa come operandi durante le operazioni;

ALOAD: informa l'ALU su cosa fare internamente con il risultato dell'operazione;

AOP: informa l'ALU sul tipo di operazione da eseguire e su quali operandi;

X, Y, S, W: sono campi ad un bit che corrispondono ai registri periferici pipeline di input;

OUTSEL: seleziona i registri (fino a sette) che devono essere caricati con il contenuto dell'Y-BUS;

CN: è il bit di riporto dell'ALU;

CND: specifica le condizioni sotto le quali devono essere eseguite le microistruzioni;

INSEL: seleziona i registri sul D-BUS, il cui contenuto viene usato come operando per l'istruzione corrente.

Il linguaggio microassembler implementato per il DMX-1000 prevede 15 tipi di microistruzione, di cui sei ad un operando e nove a due operandi:

ADD : addizione di due operandi in complemento a due;

ADI : addizione di due operandi in complemento a due e incremento di 1;

AND : addizione logica;

BIC : addizione logica (il secondo operando complementato a uno);

CLR : dà sempre 0 come risultato;

COM : complementazione dell'operando;

DEC : sottrae 1 dall'operando;

INC : somma 1 all'operando;

MOV : il risultato dell'operazione è l'operando stesso;

NEG : negazione; complementazione a 2 dell'operando;

OR : 'or' logico;

SBD : sottrazione tra due operandi in complemento a 2 e decremento di 1;

SUB : sottrazione tra due operandi in complemento a 2;

XNOR: 'nor' esclusivo;

XOR : 'or' esclusivo.

Per meglio comprendere cosa si intende per microprogrammazione e come sia possibile realizzare la sintesi digitale del suono mediante periferiche microprogrammate, vediamo un microprogramma esemplare che implementa digitalmente l'oscillatore.

Il microprogramma è riportato in Figura 5; l'oscillatore è realizzato con il metodo classico della forma d'onda fissa (detto anche 'stored function') in cui si memorizza un ciclo della forma d'onda che si vuole generare mediante l'oscillatore e lo si campiona con una frequenza proporzionale alla frequenza del segnale desiderato.

Le microistruzioni dalla (0) alla (5) costituiscono la sezione di inizializzazione; le microistruzioni dalla (6) alla (15) costituiscono l'oscillatore vero e proprio e possono essere replicate per implementare più oscillatori contemporaneamente; la microistruzione (16) è l'ultima prima dell'HALT che riporta nello stato L1 e ha l'effetto di muovere l'uscita di tutti gli oscillatori presenti (in questo caso uno) al convertitore digitale/analogico.

Gli indirizzi indicati nel microprogramma sono espressi in ottale, per cui $400_8 = 256_{10}$, $1000_8 = 512_{10}$, e così via; nel registro M viene impostato un valore compreso tra $-40C_8$ e $+400_8$ a cui viene sommato 1000_8 così che

oscilla tra 400_8 e 1400_8; il valore corrente viene quindi usato come indice della tabella che contiene la forma d'onda.

Ogni volta che si incrementa SI si incrementa la frequenza prodotta in uscita di 0,29 Hz per volta; questo fattore determina la risoluzione in frequenza dell'oscillatore; se per esempio vogliamo produrre una frequenza di 440 Hz dobbiamo usare il valore di SI che produce la frequenza più vicina a 440 Hz, cioè 1494; questo darà in uscita una frequenza di 440,089 Hz.

La risoluzione di 0,29 Hz in genere è accettabile, ma a frequenze basse diventa circa un quarto di semitono (intorno ai 20 Hz); per avere una maggiore risoluzione è possibile allora aumentare la precisione usando dati di 32 bit per la fase e per SI.

Il microprogramma rimane circa lo stesso; basta aggiungere 6 microistruzioni nel corpo del microprogramma, e più precisamente sostituire le microistruzioni dalla (6) alla (8) con le microistruzioni ed i dati indicati sempre in Figura 5 in basso.

Con questa 'ultraprecisione' si migliora la risoluzione di un fattore 2^{16} cioè si raggiunge una risoluzione di 0,000004 Hz.

IL SOFTWARE DEL SISTEMA E I COMANDI IMPLEMENTATI.

Un sistema generale per l'elaborazione musicale può essere organizzato come è illustrato in Figura 6.

E' un sistema in cui è prevista la possibilità di rappresentare e manipolare l'informazione musicale a diversi livelli di astrazione, come era già stato previsto in (Haus, 1976) e in (Vidolin, 1980); sono infatti previsti i seguenti quattro livelli:

a) LIVELLO STRUTTURALE: in cui il testo musicale è costituito da simboli che rappresentano oggetti sonori e da operatori definiti su sequenze di tali simboli;

b) LIVELLO SIMBOLICO: in cui il testo musicale è costituito da sequenze di simboli che rappresentano oggetti sonori;

c) LIVELLO FISICO: in cui gli oggetti sonori sono descritti come processi acustici ovvero il testo musicale viene descritto come processo fisico;

d) LIVELLO OPERATIVO: in cui i processi acustici sono espressi mediante istruzioni di linguaggi di programmazione capaci di realizzarli.

In particolare, l'informazione a livello strutturale comprenderà operatori derivanti dalla prassi della musica tradizionale (trasposizione, retrogradazione, inversione speculare, etc.) e operatori più vicini alle composizioni musicali sintetiche (siano esse analogiche o digitali) come introdotto in (De Stefano/Haus, 1980) ed esemplificato in (De Stefano et al., 1981).

In corrispondenza a questi quattro livelli l'informazione può essere archiviata (cioè memorizzata in modo permanente) sotto quattro diverse forme descrittive in diverse opportune librerie.

Le 'uscite' del sistema sono:

a) ACUSTICHE: per mezzo del digital signal processor;

b) GRAFICHE: per mezzo di una stampante elettrostatica ad alta risoluzione per la stampa di partiture (con la possibilità di diverse convenzioni notazionali).

In altri termini, il sistema è in grado di generare sia processi musicali che testi musicali; il sistema in via di sviluppo è appunto di questo tipo; vediamo quindi sinteticamente le parti di programmazione già sviluppate.

La struttura principale del MONITOR che gestisce tutte le attività del sistema è descritta in dettaglio nel documento di specifiche funzionali (Casazza et al., 1980); la stesura di tutti i documenti di specificazione è stata effettuata seguendo la metodologia per lo sviluppo del software definita in (Haus, 1980a).

Il MONITOR è costituito da moduli software corrispondenti a comandi; esssendo altamente modulare è espandibile ed è infatti in continuo accrescimento.

I comandi attualmente implementati sono i seguenti:

- NEW: carica microprogrammi, dati, testi da schede sulle opportune librerie;
- LIST: stampa microprogrammi, dati, testi (completi o in parte);
- DELETE: cancella microprogrammi, dati, testi dalle opportune librerie;
- LOAD: traduce l'assembler mnemonico in codice oggetto dei microprogrammi ed invia l'oggetto con i dati associati al DMX-1000;
- EXECUTE: abilita il DMX-1000 all'esecuzione e provvede all'invio degli updates in base alle informazioni contenute nei testi musicali;
- DEBUG: permette di eseguire il microprogramma caricato sul DMX-1000 per una sola volta e di accedere ai registri del DMX-1000;
- EDIT: più che un comando è un 'ambiente' di lavoro in cui si possono effettuare operazioni di editing in tempo reale dei microprogrammi di libreria per mezzo di inserzioni, cancellazioni, sostituzioni; un comando di SAVE permette di salvare i microprogrammi modificati.

I comandi in ambiente EDIT sono i seguenti:

- INSERT: inserzione di nuove microistruzioni o nuovi dati;
- CANCEL: cancellazione di microistruzioni o dati;
- SUB: sostituzione di microistruzioni o dati;
- SAVE: memorizza il microprogramma o i dati modificati (con opzione REPLACE);

- PRINT: stampa del microprogramma o dei dati (completa o parziale) allo stato attuale (ovvero con tutti gli aggiornamenti effettuati durante la sessione di EDIT);
- UPDATE: invio del pacchetto di aggiornamenti effettuati durante la sessione sul microprogramma e/o sui dati al SDK 80;
- START: invio al SDK 80 dei comandi per l'avvio del DMX-1000 (ovvero per lo svuotamento della coda formata nella memoria del SDK 80 con la sequenza di UPDATE compresa tra il precedente comando di START e l'attuale).

Lo sviluppo del software proseguirà con l'implementazione dei moduli del sistema per la descrizione dell'informazione ai tre livelli (fisico, simbolico e strutturale); questi moduli a loro volta si articoleranno in sottomoduli devoluti all'accettazione dei testi (e quindi al colloquio musicista-sistema), alla traduzione a livelli meno astratti e in particolare alla traduzione dei testi in sequenze di updates verso il DMX-1000.

Inoltre, verrà implementato un comando, PERFORM, che permetterà di eseguire mediante il DMX-1000 testi musicali residenti in libreria e di poterli modificare in tempo reale (cioè durante l'esecuzione).

PROSPETTIVE.

Il completamento del sistema software è previsto avvenga entro un anno; allo sviluppo dei programmi sarà affiancata la sperimentazione di tecniche di sintesi digitale del suono mediante microprogrammazione, volta soprattutto alla definizione di microprogrammi standard, parametrici, che consentano l'adeguamento ad una standardizzazione della descrizione a livello fisico del testo musicale, e di conseguenza anche ai livelli di descrizione più astratti.

Di particolare interesse è l'aspetto che riguarda lo sviluppo della grafica musicale; per questo si è scelto di utilizzare una stampante elettrostatica Versatec V-80 a 200 punti di risoluzione per pollice, sotto il controllo di un package software per la generazione di stampe di alta qualità (Knuth, 1978); l'utilizzo di questo programma richiede lo sviluppo della 'fonte' musicale (cioè delle tabelle che descrivono i segni musicali come insiemi di punti).

Inoltre, in parallelo a questo sistema già in fase realizzativa avanzata, sarà sviluppato un analogo sistema, di più modeste dimensioni, con una diversa organizzazione delle librerie e con un livello di performance più elevato, basato sempre sull'uso del DMX-1000 controllato questa volta da un elaboratore Onyx C 8002 con sistema operativo UNIX.

RINGRAZIAMENTI.

Desidero ringraziare per i fattivi contributi allo sviluppo del progetto: F. Bisoni, M. Casazza, M. Malcangi, R. Martini, G. G. Triulzi.

Questo lavoro è stato realizzato nell'ambito del Communication and Programming Project tra Università degli Studi di Milano e Honeywell Information Systems Italia e con il contributo del Consiglio Nazionale delle Ricerche, Contratto N° 79.00701.02.115.9672.

RIFERIMENTI BIBLIOGRAFICI.

Bertoni/Haus/Mauri/Torelli: "A Mathematical Model for Analysing and Structuring Musical Texts", Interface, 7: 31-43, 1978, Swets & Zeitlinger B.V.

Casazza/Haus/Malcangi/Martini: "Monitor del DMX-1000 Signal Processing Computer: specifiche funzionali del sistema L62/SDK 80/DMX-1000", Rapporto Interno, Istituto di Cibernetica, 1980, Milano.

De Stefano/Haus: "A Mathematical Approach to the Representation of Musical Structures (oriented to the Transformation of Sonorous Forms)", Proceedings of the 2nd International Conference on Data Bases in the Humanities and Social Sciences, Facultad de Informatica, 1980, Madrid.

G. Haus: "Descrizione formale di processi musicali", Tesi di Laurea in Fisica, Università degli Studi di Milano, 1976, Milano.

G.Haus: "Trasformazione di testi musicali per mezzo di operatori", Atti del 3° Colloquio di Informatica Musicale, Università di Padova, 1979, Padova.

G.Haus: "Metodi di definizione di specifiche funzionali di procedure software con uso di reti di Petri" - Rapporto N° 24, Progetto Finalizzato Informatica, Sottoprogetto P1, Obiettivo Metod, CNR, Istituto di Cibernetica, 1980, Milano.

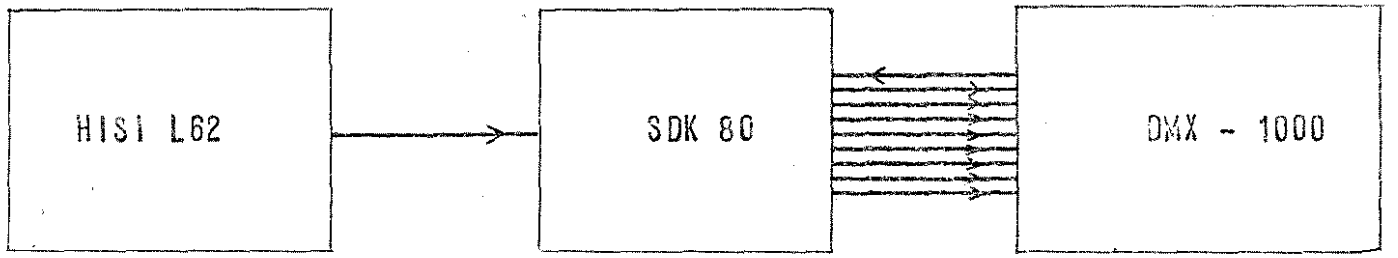
G.Haus: "Computer Music at the Institute of Cyberntics of the University of Milan", Proceedings of the 5th International Computer Music Conference, Queens College, 1980, New York.

D.Knuth: "Tau-Epsilon-Chi: a System for Technical Text", Artificial Intelligence Laboratory, 1978, Stanford University.

A.Vidolin: "Interazione tra i livelli di rappresentazione dell'informazione musicale nella composizione mediante elaboratore", Automazione e Strumentazione, ANIPLA, Vol. 27, N° 2, 1980, Milano.

D.Wallraff: "DMX-1000 Hardware Manual", Digital Music Systems, 1980, Boston.

D.Wallraff: "DMX-1000 Programming Manual", Digital Music Systems, 1980, Boston.



Architettura hardware del sistema di elaborazione.

Figura 1

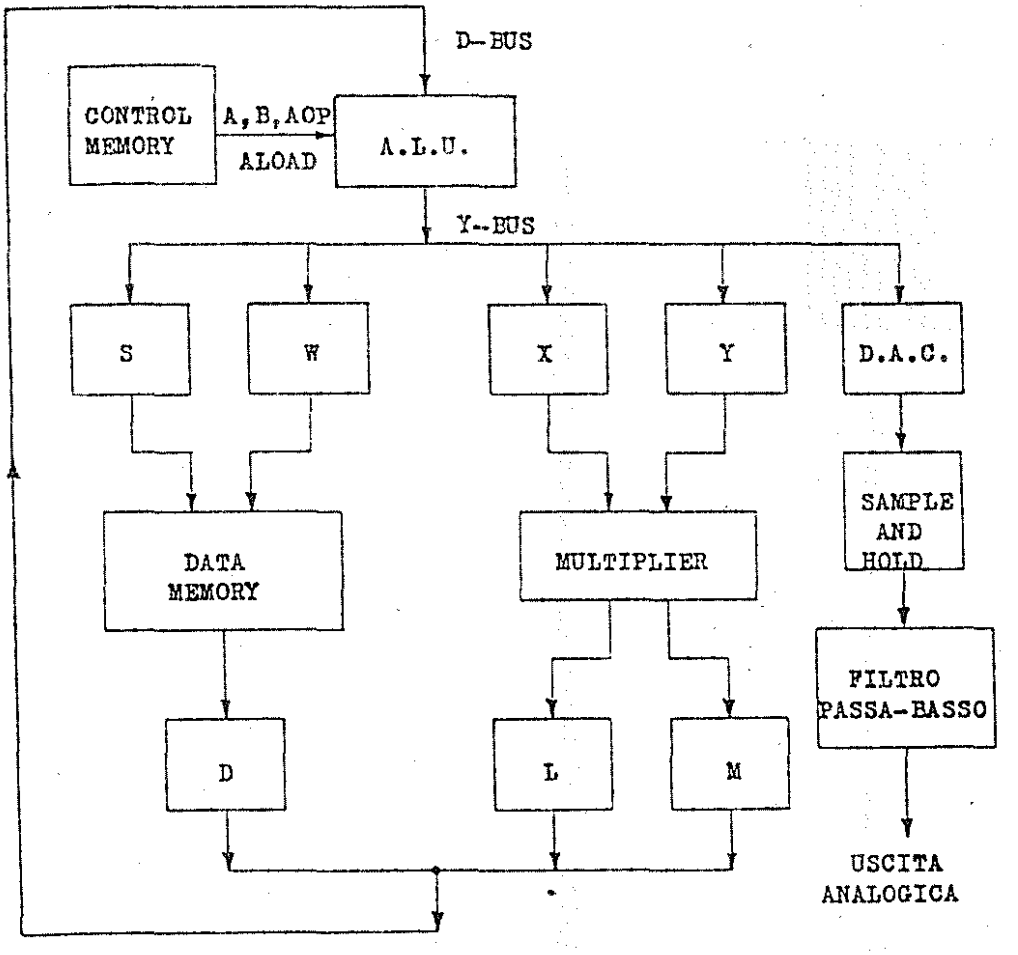


Figura 2: architettura DMX - 1000

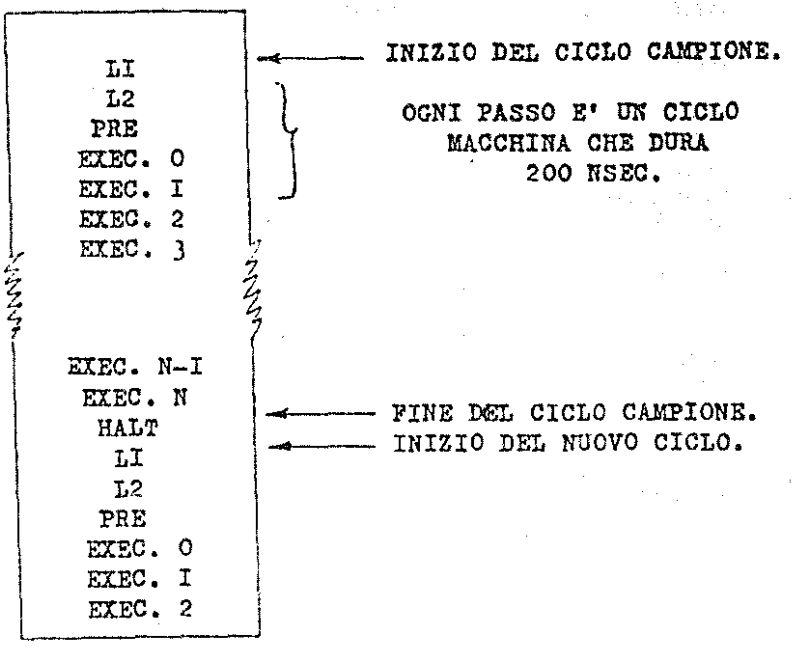


Figura 3: struttura generale di un microprogramma

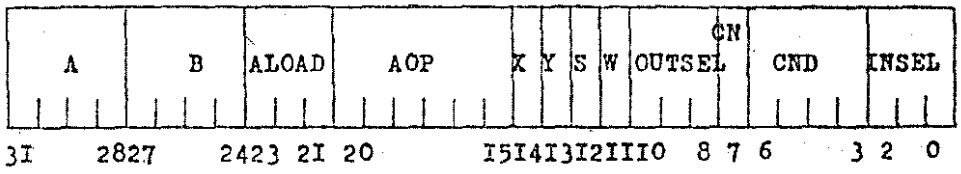


Figura 4: formato di una microistruzione

```
(0)  CLR      ,X0,B,,S          ; inizializza il puntatore dati
(1)  CLR      ,XF,B             ; pulisce l'accumulatore di output
(2)  NXT
(3)  MOVD     ,X1,B,D           ; X1=400  da  |0|
(4)  NXT
(5)  MOVD     ,X2,B,D           ; X2=1000  da  |1|
(6)  NXT
(7)  MOVD     ,X3,B,D           ; X3=SI  da  |2|
(8)  ADDDA    X3,X3,,D,WY       ; somma alla fase da |3|
(9)  MOVA     X1,,,,X           ; moltiplica per il fattore di scala
(10) NXT
(11) ADDDA    X2,,,M,S          ; somma all'indirizzo del centro
                                  della tabella eusa il risultato
                                  come indirizzo per table lookup
(12) MOVD     ,,,D,Y            ; ampiezza da |4|
(13) MOVD     ,,,D,X            ; campione dalla tabella
(14) NXT
(15) ADDDA    XF,XF,B,M         ; somma il campione 'scalato' alla
                                  uscita totale
(16) MOVA     XF,,,,DAC
```

oscillatore digitale

Dati:

```
|0|  400                    fattore di scala (1/64.)
|1|  1000                   indirizzo del centro della tabella
|2|  SI                     sample increment
|3|  ....                   fase
|4|  ....                   ampiezza
|400| - |1377|  ....        ciclo della funzione da generare
|1400| come |400|
```

```
(6)  CLR     ,X4,B          ; X4 viene usato per la parte più
                              significativa della fase
(7)  NXT
(8)  MOVD    ,X3,B,D        ; parte meno significativa di SI da |2|
(9)  ADDDA   X3,X3,B,D,W    ; somma nella p.m.s. della fase da |3|
                              e lo riscrive nella DATA MEMORY
(10) INCB    ,X4,B,,,IFCS   ; somma nella p.p.s. della fase
(11) NXT
(12) NXT
(13) ADDDA   X4,X4,B,D      ; somma nella p.p.s. di SI da |4|
(14) ADDDA   X4,X4,B,D,WY   ; somma nella p.p.s. della fase da
                              |5| e lo riscrive in |5|
```

modifiche doppia precisione

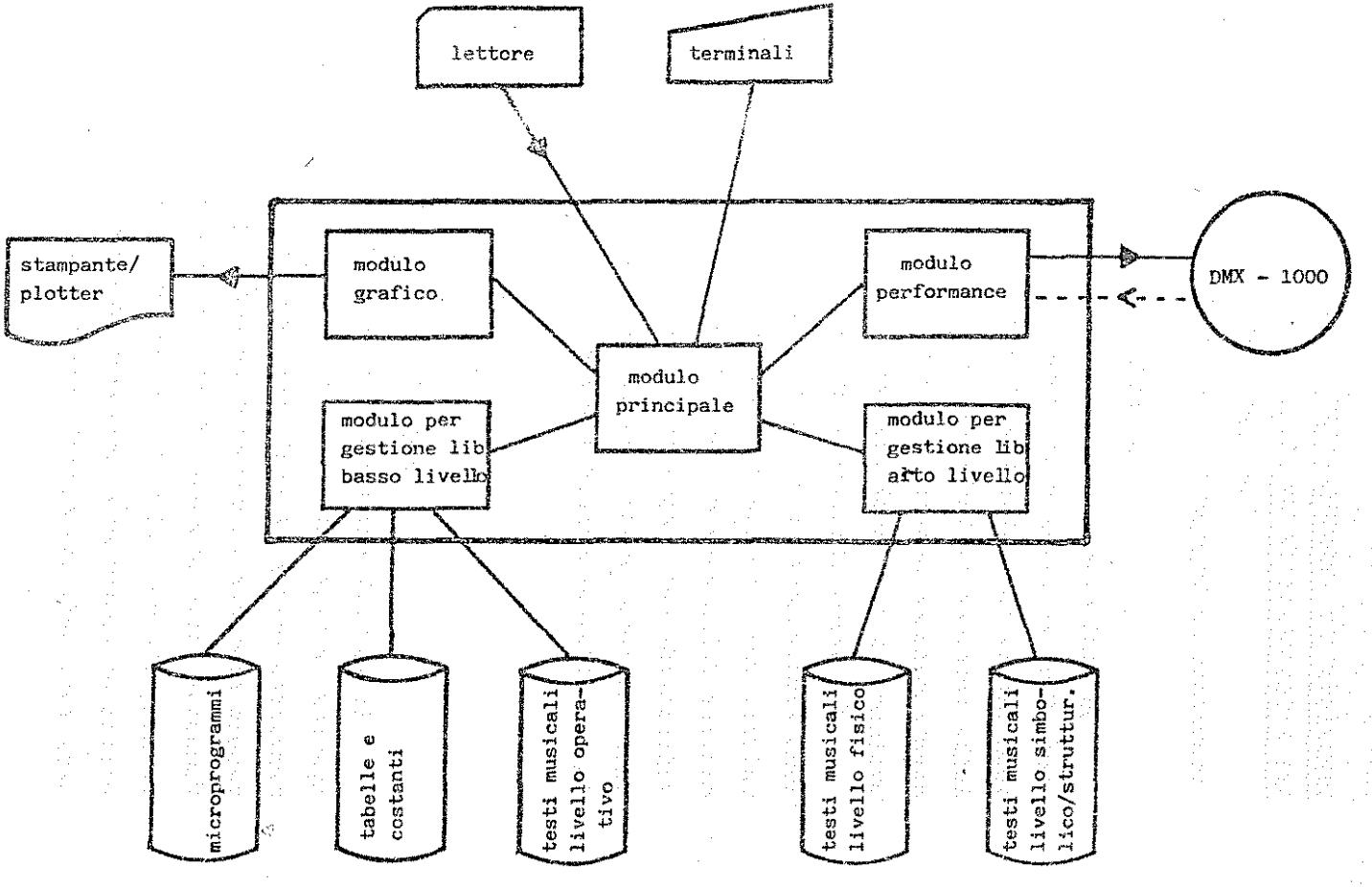
e i nuovi dati saranno:

```
|0|  come la posizione |0| precedente
|1|  come la posizione |1| precedente
|2|  parte meno significativa di SI
|3|  parte meno significativa della fase
|4|  parte più significativa di SI
|5|  parte più significativa della fase.
|6|  come la posizione |4| precedente
|400| - |1400|  come le posizioni |400| - |1400| precedenti
```

Figura 5

ARCHITETTURA DEL SISTEMA INFORMATIVO

Figura 6



ESPERIENZE DI PROGETTAZIONE E SVILUPPO DI SISTEMI EFFICIENTI PER L'ELABORAZIONE MUSICALE

Mº Mario Malcangi

laureando in Ingegneria Elettronica al Politecnico di Milano

SOMMARIO

La limitatezza delle risorse e le stringenti richieste di esecuzione in tempo reale rendono la progettazione di un sistema per la "Computer Music" particolarmente complessa.Molte delle tecniche sviluppatesi nell'ambito dell'informatica generale possono essere oggi potenti mezzi per la ricerca e lo sviluppo in tali applicazioni.

Qui verrà illustrata una serie di esperienze che riguardano praticamente i più significativi passi della progettazione.Essi sono: scelta del linguaggio di programmazione, definizione della base dei dati in memoria centrale e di massa, definizione del linguaggio di comunicazione uomo-macchina,problemi di tempo reale durante la comunicazione uomo-macchina e durante l'esecuzione,distribuzione dell'informazione.

ABSTACT

It is well known that the development of Computer Music systems is made quite complex by the heavy demand on computer resources mainly if real time performance is considered. Many recent advances in computer science may help in developing Computer Music systems. In this work several experiences concerning the most significant design steps are reported,such as: choise of programming language,man-machine language definition, data base definition,real time and multiprocessing problems.

1.- La scelta della macchina HARDWARE/SOFTWARE

Per quanto non ci si trovi molto spesso nelle condizioni di scegliere liberamente la macchina su cui sviluppare il sistema, è importante che questo, oltre le tipiche proprietà(un buon rapporto velocità/capacità di memorizzazione) deve avere le seguenti caratteristiche:

a) basso costo

b) elevata diffusione

c) caratteristiche costruttive ed operative standardizzate

A mio avviso queste sono proprità della macchina base che potenzialmente possono offrire all'intero sistema caratteristiche di alta diffondibilità e trasportabilità.

La diffondibilità a basso costo è ovviamente importante per sistemi di questo tipo in quanto è sempre più prossimo il tempo che la computer music uscirà dai laboratori di ricerca ed entrerà in ambienti sociali sempre più ampi quali per esempio la scuola.

La standardizzazione assicura inoltre, sè il sistema è stato progettato con criteri di trasportabilità, la possibilità di usare macchine base diverse da quelle su cui è stato sviluppato il sistema stesso.

2.- Scelta del linguaggio di programmazione

La scelta del linguaggio di programmazione risulta meno condizionata della precedente in quanto qualsiasi calcolatore è in grado di offrire oggi i più importanti linguaggi di programmazione. L'importanza di tale scelta stà comunque nel fatto che ogni linguaggio è in grado di lavorare in modo efficiente in determinate applicazioni e meno efficiente in altre.
Un sistema per l'elaborazione musicale prevede applicazioni molto diverse fra loro come elaborazione di stringhe ed elaborazione numerica, oppure gestione di archivi e comunicazione uomo-macchina. Chiaramente un linguaggio tipo Assembler è potenzialmente in grado di risolvere tutti questi problemi, ma risulterebbe poco maneggevole e troppo legato alla macchina base, la qual cosa disattenderebbe le ipotesi di standardizzazione del sistema. Se si deve comunque optare per un unico linguaggio allora conviene scegliere un linguaggio di tipo "general purpouse" suppurtato dalla maggior parte dei calcolatori.
Se invece si è nelle condizioni di poter utilizzare più linguaggi allora ci si pone nelle condizioni di risolvere il problema nel modo più efficiente possibile. Infatti in questo caso si può utilizzare il linguaggio adatto a risolvere lo specifico problema. Nella progettazione del Musicomp(°) sono stati utilizzati ben tre linguaggi: il Pascal, il Fortran, il Macro (assembler del PDP 11/34). Il Pascal è stato scelto in quanto è un linguaggio "general purpouse" in grado di risolvere cioè efficentemente molti problemi relativi al trattamento della informazione, alla sua memorizzazione (gestione dinamica della memoria), alla definizione dei tipi dei dati, alla manipolazione dei dati (operatori e funzioni speciali), ecc; il Fortran è stato

(°)-Tesi di Laurea in Ingegneria Elettronica del Politecnico di Milano, relatore Prof. F. Maffioli, sul tema: "Un sistema per la composizione e la sintesi sonora in tempo reale su calcolatore elettronico".

scelto per l'elaborazione numerica dei segnali; l'Assembler per realizzare tutte quelle operazioni che normalmente non si riescono ad avere con i linguaggi ad alto livello per esempio accesso random ai records dei files sequenziali.
Il main program è stato implementato nel linguaggio a più alto livello, cioè il Pascal. Il linguaggio a più alto livello è dunque quello che deve avere le più ampie caratteristiche operative. Così, con la scelta multi-linguaggio ci assicuriamo la possibilità di poter utilizzare nell'ambito del sistema progettato parti di software sviluppate altrove da altri.

3.- Definizione della base dei dati.

La definizione della base dei dati è uno dei momenti fondamentali nella progettazione dei sistemi per applicazioni musicali. Una base dei dati opportunamente definita può assicurare particolari vantaggi che conferiscono efficenza al sistema. In particolare assicura indipendenza della definizione dei dati dalla particolare applicazione che se ne desidera fare. In tal modo il sistema risulta immune da variazioni od ulteriori sviluppi che esso potrebbe subire.

La definizione della base dei dati parte dall'analisi dettagliata dei dati su cui deve operare,nel nostro caso il suono e le sue molteplici articolazioni che rappresentano in ultima analisi la "Composizione".

Per formulare quanto verrà detto,chiamiamo"Evento" l'entità singola qual'è il suono e "Macroeventi" gli elementi molteplicemente composti di eventi. Secondo tale definizione è un Evento per esempio il singolo suono che si otterrebbe premendo un tasto di pianoforte. E' invece un Macroevento quello che si otterrebbe premendo più tasti.

La logica relazione che lega Eventi e Macroeventi è la seguente:

$$\text{Macroevento} \supseteq \text{Evento} \qquad \text{(a)}$$

A questa relazione se ne può imporre arbitrariamente un'altra che ci permetterà di identificare univocamente l'Evento e quindi per la (a) anche il Macroevento:

L'Evento è una quadrupla del tipo:

Evento = $E(T(t),F(t),A(t),D)$

$T(t)$: timbro;

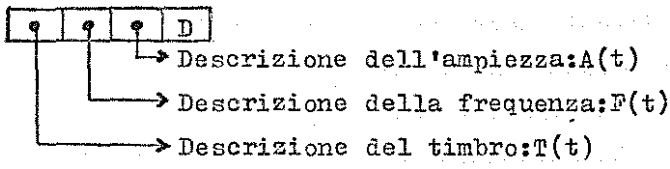
$F(t)$: frequenza;

$A(t)$: ampiezza;

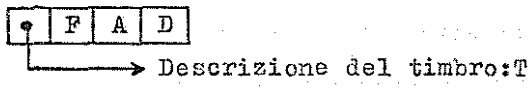
D : durata.

Da tale definizione se ne deduce che l'Evento è un'entità dinamica in quanto esiste in esso una dipendenza funzionale dal tempo. Ciò significa che T(t),F(t),A(t) non sono rappresentabili con un semplice numero come invece può essere fatto per D.

L'Evento può dunque essere così rappresentato:



Per semplificare la trattazione trascuriamo la dipendenza temporale in quanto così non perdiamo in generalità ottenendo la seguente struttura semplificata:



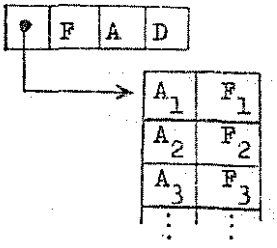
Indipendentemente dalla dipendenza temporale il timbro resta comunque un parametro strutturato così definibile:

$$\text{Timbro} = T(A_i, F_i) \quad i=1..n$$

Il timbro è dunque un'insieme di coppie che descrivono le caratteristiche di ampiezza e fase del suo spettro. Esso ha dunque una struttura del tipo:

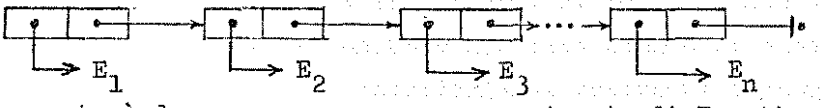
A_1	F_1
A_2	F_2
A_3	F_3
⋮	⋮

La struttura dell'Evento si completa come segue:



Il Macroevento è stato definito come un insieme qualsiasi di Eventi. Anche in questo caso possiamo considerare in prima approssimazione il Macroevento come un insieme temporale di Eventi ordinati come una sequenza.

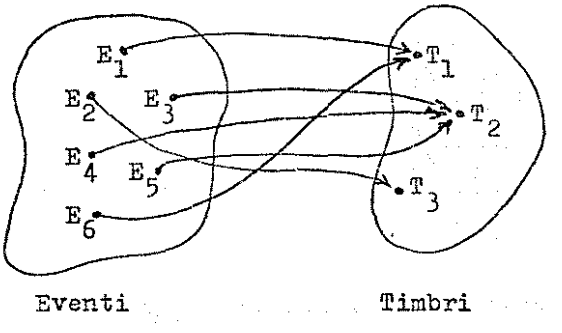
La struttura informativa che bene rappresenta un simile Macroevento è la seguente:



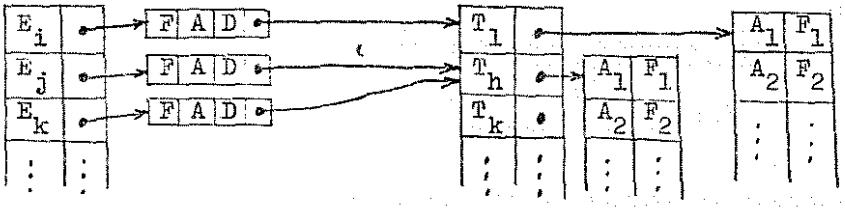
Il Macroevento è dunque una sequenza concatenata di Eventi.

Secondo passo della definizione della base dei dati è l'analisi delle relazioni esistenti fra i tre tipi di dati prima definiti:Macroeventi,Eventi e Timbri.

Va subito notato che la relazione esistente fra Eventi e Timbri è unidirezionale in quanto un Evento ha come attributo un Timbro,ma il Timbro non ha come attributo un evento. Inoltre possono esistere più Eventi caratterizzati dallo stesso Timbro, per esempio uno strumento che suona tre suoni diversi. La relazione Eventi-Timbri è dunque la seguente:

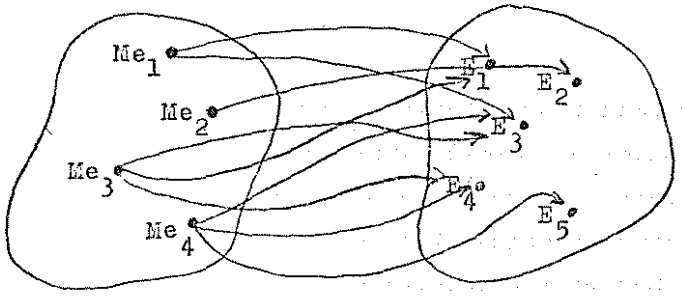


Se rappresentassimo la struttura informativa Evento-Timbro come un tutt'uno otterremmo una struttura ridondante. Per esempio le coppie E_1T_1, E_6T_1 se fossero così rappresentate T_1 verrebbe memorizzato due volte, mentre ciò non accadrebbe se T_1 fosse memorizzato una sola volta ed E_1 ed E_6 potessero referenziarlo tramite un puntatore alla sua locazione di memoria.
E' chiaro allora che i due insimi di dati devono essere organizzati come archivi in cui ogni elemento vi è memorizzato una sola volta ottenendo così:



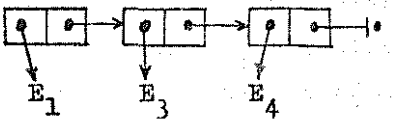
Archivio Eventi Archivio Timbri

Analizziamo ora la relazione esistente fra Macroeventi ed Eventi. Anche in questo caso la relazione è unidirezionale e come per i Timbri un Evento può essere referenziato da più Macroeventi. In più un Macroevento può referenziare molti Eventi. In definitiva abbiamo una relazione del tipo:

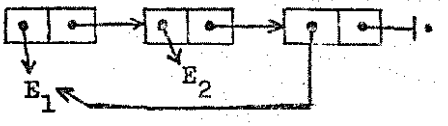


Macroeventi Eventi

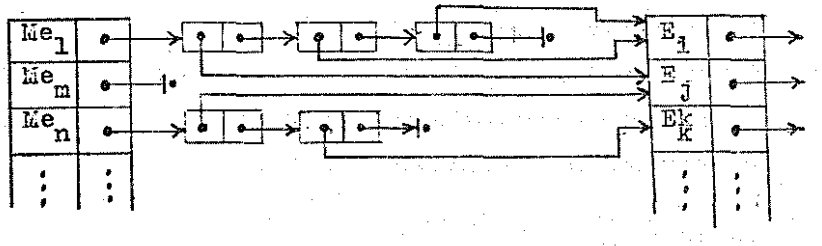
Nella rappresentazione dei Macroeventi il problema è il medesimo: evitare le ridondanze di informazione. Anche in questo caso si può adottare una struttura del tipo a lista i cui elementi non sono altro che puntatori all'archivio Eventi:



Questo per esempio potrebbe essere la rappresentazione del Macroevento Me_3. In questo modo un Macroevento può referenziare più volte lo stesso Evento senza che questo venga memorizzato più volte:



Come vedremo in seguito è utile che anche i Macroeventi vengano organizzati in un unico archivio in modo che anche questi possano comodamente essere referenziati da quel particolare Macroevento chiamato Composizione:



Questo procedimento ascendente di analisi (in quanto si risale dall'informazione di livello inferiore a quella di livello superiore) può essere applicato fino a raggiungere l'entità più complessa che si prevede venga utilizzata nel nostro sistema. La struttura informativa così sintetizzata è un'ottima implementazione della base dei dati da porre in memoria di massa in quanto è minima per quanto riguarda l'occupazione di memoria e come vedremo in seguito è anche efficente dal punto di vista dell'ac-

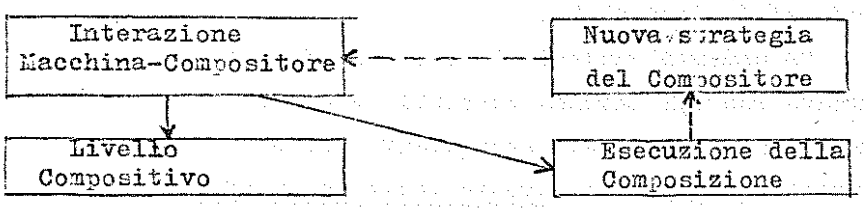
cesso all'informazione.

In memoria centrale la base dei dati è la stessa con la differenza che qui saranno presenti solo quelle informazioni strettamente necessarie all'avanzamento del processo in esecuzione corrente nel sistema. Questo permette un'efficente utilizzo della memoria centrale in modo dinamico in quanto l'occupazione di memoria aumenta o diminuische in funzioni delle reali richieste del processo corrente.

4.- Definizione del linguaggio di comunicazione uomo-macchina.

Com'è noto tutte le volte che dobbiamo "comunicare" con una certa macchina è necessario disporre di un oggetto che ci renda facile tale comunicazione. Questo oggetto è il "linguaggio". Il linguaggio può essere a più o meno alto livello a seconda che risulti più o meno vicino all'uomo o alla macchina. Nel nostro caso la macchina è il calcolatore che viene programmato per contenere certi oggetti per esempio Eventi e Macroeventi. La definizione del linguaggio di comunicazione in questo caso prevede la ricerca di una forma tale da equilibrare in modo efficente le due esigenze: la macchina non deve essere impegnata in un trattamento del linguaggio troppo elevato e l'uomo non deve cimentarsi con linguaggi troppo complessi. La prima scelta da farsi è quella fra linguaggio interpretativo e linguaggio compilativo. Il primo è certamente preferibile per sistemi per applicazioni musicali anche se la macchina in tal modo potrebbe funzionare in modo meno efficente. Il vantaggio comunque è quello di dare una parvenza di colloquio con la macchina, la qual cosa risulta certamente più gradita all'utilizzatore.
Il calo di efficenza nella macchina che comunque si avrebbe seguendo tale scelta viene ampiamente colmata dalla possibilità di retroazione. In pratica con un linguaggio interpretativo è possibile variare la strategia operativa in funzione dei ri-

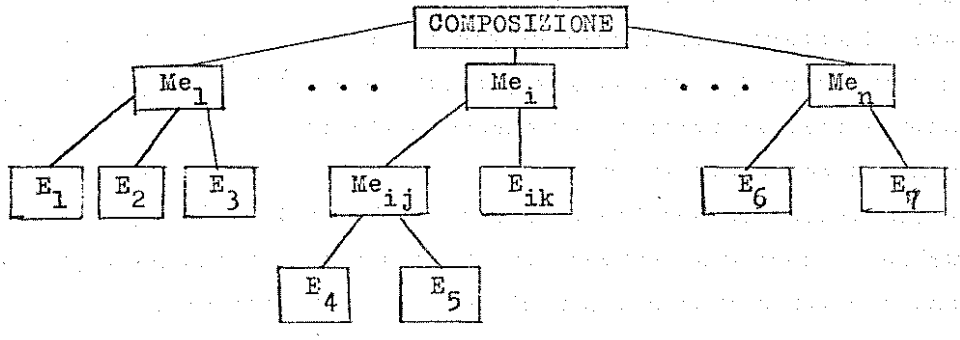
sultati che passo passo si ottengono.



Nelle applicazioni di tipo musicale la comunicazione con la macchina avviene a due livelli: prima di tutto l'operatore deve sceglire le funzioni o procedure di lavoro che il sistema mette a disposizione, quindi operare in questa procedura comunicando tramite un secondo tipo di linguaggio. Quest'ultimo è più critico in quanto è quello che deve permettere di tradurre l'idea musicale in istruzioni per la macchina.

Il problema è dunque quello di sceglire una grammatica che generi un linguaggio con cui l'utente riesce ad esprimere facilmente le sue idee compositive e che sia facilmente implementabile sulla macchina.

Qui verranno mostrate le scelte che nell'implementazione della procedura compositiva del Musicomp sono state fatte al fine di soddisfare nel modo migliore queste esigenze contrastanti. La composizione è un insieme di Macroeventi i quali a loro volta sono insiemi di Eventi, dunque una struttura del tipo:



In questa struttura,dall'alto in basso,si possono mettere in evidenza tre componenti fondamentali:

- radice: è l'elemento da cui si parte;
- nodi : sono elementi intermedi a cui si arriva in un unico modo e si riparte in più modi;
- foglie: sono elementi terminali;

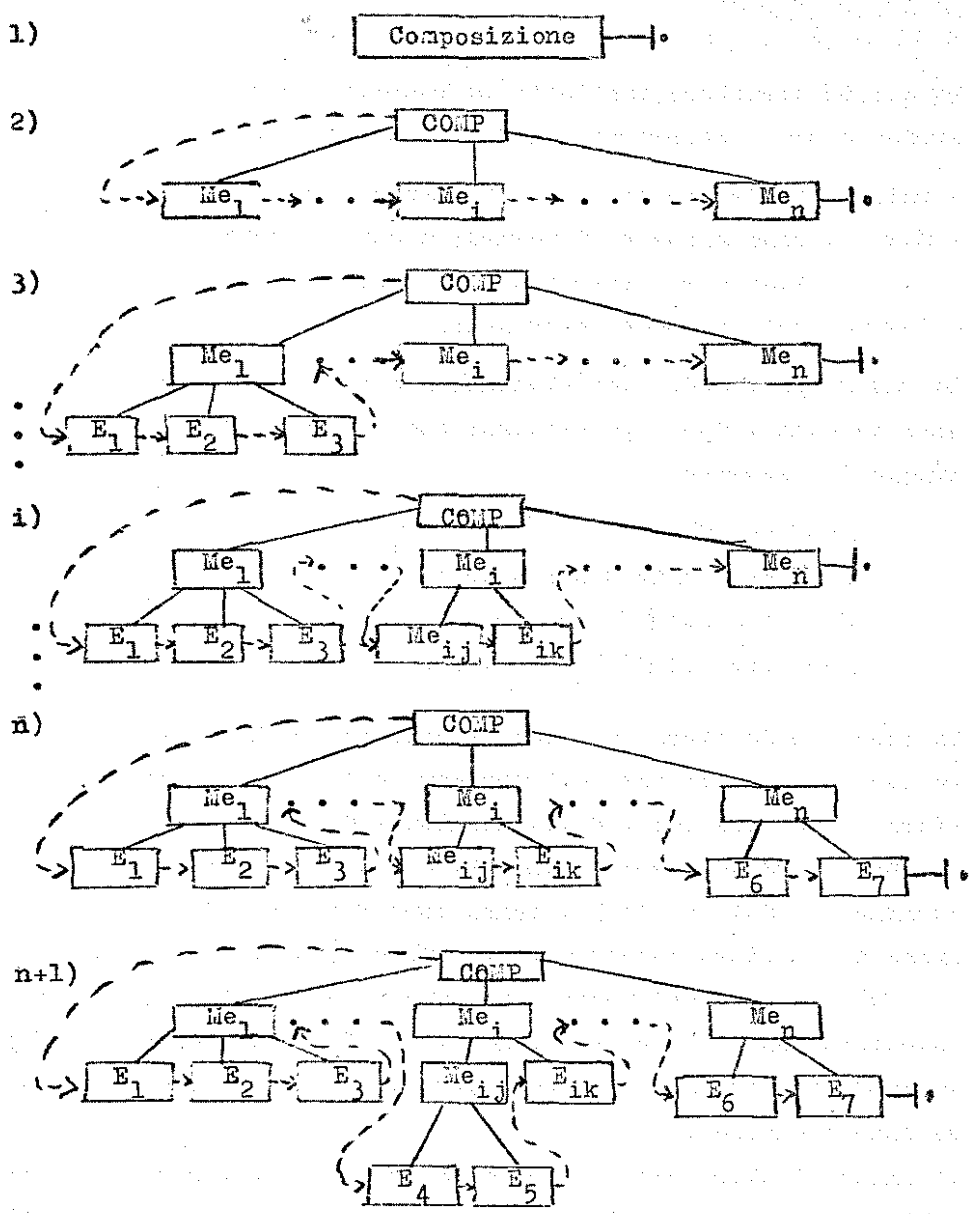
La radice ed i nodi sono elementi non terminali <nt>,le foglie sono terminali <t> . La sintassi che allora è stata proposta è dunque la seguente:

<nt> ::= <s>
<s> ::= (<A> <op> <A>)
<A> ::= <t> | <nt>
<op> ::= . | ∪ | ∩ | . . .

In questo modo siamo in grado di controllare la costruzione della composizione in modo estremamente efficente in quemto controlliamo l'evolvere della della struttura della composizione stessa. Va ricordato comunque che gli operatori hanno come operandi stringhe di <t> e <nt> e danno come risultato della loro azione una stringa concatenata di <t> ed <nt> .

Un operatore elementare in questo caso è l'operatore di concatenazione '.' che,applicato a due <t> o due <nt> o un <t> ed un <nt> o stringhe di questi,produce come risultato la loro concatenazione.
E' chiaro inoltre che ,disporre di operatori molto potenti, risulta estremamente utile per una rapida espansione di nodi<nt> quando tali operatori,applicati a piccole stringhe,generano grandi stringhe. Questo,per esempio potrebbere essere il caso dell'operatore "prodotto cartesiano".

La struttura di dati su cui si opera in tali termini è l'albero. La composizione quindi,istante per istante,è la frontiera di tale albero da sinistra a destra:


1)
Composizione
2)
COMP
Me1
Mei
Men
3)
COMP
Me1
Mei
Men
E1
E2
E3
i)
COMP
Me1
Mei
Men
E1
E2
E3
Meij
Eik
n)
COMP
Me1
Mei
Men
E1
E2
E3
Meij
Eik
E6
E7
n+1)
COMP
Me1
Mei
Men
E1
E2
E3
Meij
Eik
E6
E7
E4
E5

Come è noto, tutte le volte che dobbiamo "comunicare" con una certa macchina è necessario disporre di un oggetto che ci renda facile tale comunicazione : questo oggetto è il linguaggio. Il linguaggio può inoltre essere a più o meno alto livello a seconda che sia più vicino all'uomo o alla macchina. nel nostro caso la macchina è il calcolatore che è stato programmato per contenere certi oggetti che abbiamo chiamato Eventi e Macroeventi. Sulla base della grammatica proposta se ne può dedurre un linguaggio ad assegnamenti con regole sintattiche tipo:

1 : <assegnamento> ::= <variabile> := <espressione> ;
2 : <espressione> ::= (<termine><operatore><termine>)
3 : <variabile> ::= <identificatore di variabile>
4 : <termine> ::= <variabile> <espressione>
5 : <operatore> ::= .| ∪ | ∩ | . . .

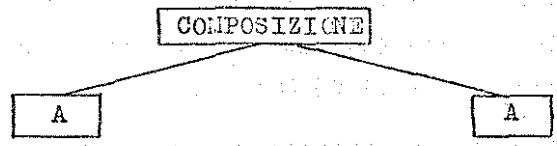
Si tratta in pratica di un linguaggio costituito di frasi di asse assegnamento. Con tale linguaggio siamo in grado di costruire in modo discendente la composizione desiderata. Per esempio la struttura di composizione, precedentemente illustrata, può essere descritta tramite la seguente serie di assegnamenti (non unica):

1) si assegna il nome di identificazione alla composizione:
   Composizione:=COMP;
2) $COMP := Me_1 \cdot \ldots \cdot Me_i \cdot \ldots \cdot Me_n$
3) $Me_1 := E_1 \cdot E_2 \cdot E_3$;
⋮
i) $Me_i := Me_{ij} \cdot E_{ik}$;
⋮
n) $Me_n := E_6 \cdot E_7$;
n+1) $Me_{ij} := E_4 \cdot E_5$;

Si potrebbe osservare che il tutto si poteva fare anche con un unico assegnamento in cui a COMP si assegnava direttamente la stringa $E_1E_2E_3E_4E_5E_6E_7$. A tale prososito si possono ricordare due aspetti importanti della composizione:

a) in generale la struttura della composizione è alquanto complessa ed articolata;
b) ci sono molto spesso, ripetizioni o altre forme strutturali tipo specularità, molteplicità, ecc.

Con un semplice esempio illustriamo ora il punto b) per meglio chiarire la potenza del metodo. Consideriamo il seguente caso:



La composizione è composta di due parti uguali A. Supponiamo che A sia composta a sua volta di cento Me. Non seguendo il metodo discendente dovremmo eseguire il seguente assegnamento:

COMPOSIZIONE:= $Me_1 \ldots Me_{100} \cdot Me_1 \ldots Me_{100}$;

Con il metodo discendente facciamo invece i due seguenti assegnamenti:

COMPOSIZIONE:= A·A;
A:= $Me_1 \ldots Me_{100}$;

Così abbiamo ridotto praticamente a metà il lavoro!
Questo è solo uno dei possibili vantaggi; mentre per esempio un altro potrebbe essere quello di una visione corrente della struttura della composizione stessa da parte del compositore nel momento stesso in cui la costruisce.

5.- Problemi di tempo reale durante la comunicazione uomo-macchina e durante l'esecuzione.

I problemi di tempo reale sono quelli che limitano maggiormente le applicazioni musicali sugli elaboratori. E' quindi su queste problematiche che bisogna concentrarsi per ottenere i migliori risultati dalla macchina base utilizzata. Questi problemi possono essere divisi in due categorie fondamentali: quelli che riguardano la comunicazione uomo-macchina e quelli che riguardano l'esecuzione di una sequenza musicale. La prima categoria non pone stringenti problemi in quanto i tempi di riflessione dell'uomo sono nettamente inferiori a quelli tipici della macchina. In tal caso conviene fare in modo che la macchina faccia più elaborazioni possibili, cioè lavori il più possibile nei tempi di colloquio con l'uomo: il ritardo nella risposta in questo caso non viene sentito. In queste fasi conviene fare quelle eleborazioni pesanti tipo aggiornamenti di archivi, ordinamento di archivi, pre-elaborazione dei dati relativi alla sintesi del suono, riduzione della sintesi del suono mediante algoritmi di compattamento. Soprattutto questi ultimi due punti saranno quelli che in particalar modo favoriranno l'esecuzione in tempo reale, cioè la seconda categoria di problematiche.

Questa seconda categoria pone problemi molto stringenti. Per rendercene conto basta fare il seguente calcolo: per una esecuzione in tempo reale che copra l'intero campo di udibilità (16 ÷ 16000 Hz) è necessario nell'ipotesi di segnale campionato, di emettere, teoricamente, 1 campione ogni:

$$\frac{1}{2 \times 16000} = \frac{1}{32000} = 31{,}25 \ \mu s$$

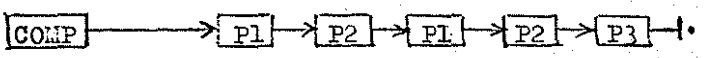
In realtà bisognerà scendere a valori inferiori di tempo in quanto non si lavora mai in condizioni ideali.
Questi tempi di trasmissione impegnano seriamente anche le macchine più veloci,non tanto per la velocità stessa di trasmissione,ma per il fatto che questo è l'intervallo di tempo che la macchina ha a disposizione per preparare un campione da mandare in uscita. In un simile intervallo di tempo non si è in grado di far eseguire un gran che di istruzioni alla macchina;
Ormai ci si è convinti che non conviene più tenere in memoria tutte le sequenze campionate da mandare al sistema di conversione e trasduzione in quanto ciò comporterebbe occupazioni di memoria inaccettabili,ma piuttosto conviene generarli "run-time". E questo dunque il motivo per cui bisogna fare più preelaborazioni possibili "off-line",cioè quando i problemi di tempo reale non sono troppo stringenti.

Per ottenere questi risultati nella progettazione del Musicomp ci si è rifatti ai principi su cui si basa la progettazione dei linguaggi di programmazione. Questi si dividono in due categorie fondamentali: interpretativi e compilativi.

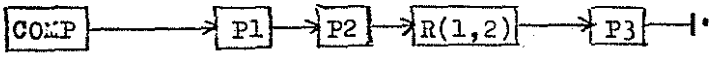
Il linguaggio interpretativo tipo LISP prende una stringga che rappresenta l'istruzione corrente data dall'operatore e la esamina durante l'esecuzione stessa del programma generando run-time il codice maccina per eseguirla ed i risultati di tale istruzione allo stesso tempo.
Il linguaggio compilativo produce invece compile-time il codice macchina che poi run-time verrà eseguito.
La differenza sta nel fatto che il primo impiega più tempo a dare gli stessi risultati,però nel primo caso con l'esecuzione run-time si ottiene un minor ingombro di memoria.
Nella progettazione del Musicomp si è tenuto conto di questi vantaggi come segue:

Durante la comunicazione uomo-macchina per la definizione della composizione si è fatto in modo che nella sua descrizione in memoria non fosse contenuto nulla che durante l'esecuzione dovesse essere oggetto di interpretazione. Per tale motivo è stata adottata una rappresentazione della composizione del tipo:

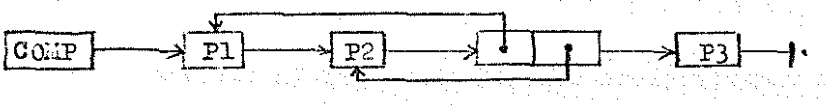
Ogni componente Pi è un insieme di puntatori che rappresentano la mappa dei percorsi che il programma di esecuzione deve seguire in memoria per ottenere tutti i dati richiesti dal programma di sintesi sonora. Ciò in pratica equivale ad aver "compilato" il "programma" rappresentato dalla composizione.ed il "codice macchina" è nel nostro caso un indirizzo di memoria.
Al fine di ottimizzare ulteriormente l'occupazione di memoria si è anche ricorso ad algoritmi di compattamento del tipo di quelli proposto da Haus ed altri di descrizione dei testi in termini di operatori.
In questo caso se la composizione consiste di:



ove P1 ÷ P2 è ripetuto 2 volte si può scrivere:



ove R(1,2) è l'operatore di ripetizione che utilizziamo qui per ridurre l'occupazione di memoria.
Poichè durante l'esecuzione questi operatori andrebbero interpretati,cosa che nel nostro caso non è desiderabile,la composizione viene compilata come segue:

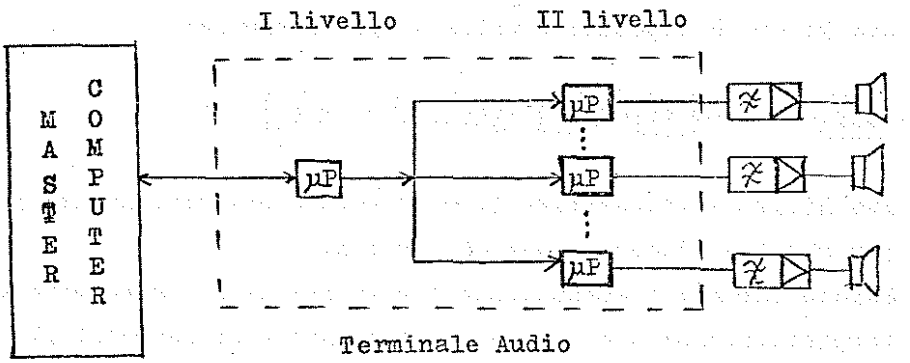


In tal modo siamo ritornati ad avere un "codice macchina" che ha inoltre il vantaggio di essere compattato. Abbiamo cioè ottenuto contemporaneamente due risultati: ridotta occupazione di memoria ed esecuzione veloce.

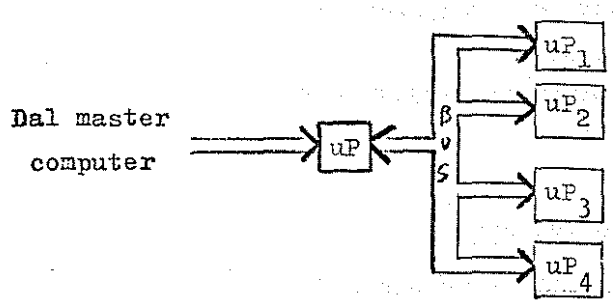
6.- Distribuzione dell'elaborazione.

I problemi di tempo reale trovano ottima risoluzione quando si ricorre alla distribuzione dell'elaborazione. Questo risulta abbastanza logico in quanto, se si adotta per sempio una macchina multi-processor si arriva ad ottenere tempi di elaborazione che, grosso modo, sono 1/n volte quello simple-processor, ove n è il numero di processori che possono funzionare in parallelo.

In generale comunque non conviene adottare una macchina multiprocessor in conseguenza dei suoi elevati costi, ma piuttosto conviene utilizzare più macchine simple-processor che lavorino in parallelo. Questa scelta favorisce anche l'ipotesi di standardizzazione del sistema. In particolare risulta abbastanza conveniente adottare la scelta del sistema composto da una macchina Master ed uno o più terminali audio:



Come si può vedere il terminale audio può,a sua volta,essere multi-processor,cioè un insieme a due livelli d.. μP simple-processor.
Una soluzione di questo tipo porta ad avere un sistema multi-processor composto da molti simple-processor di tipo standard. Tutto questo è in grado di soddisfare efficentemente l'ipotesi di elaborazione distribuita.
I vantaggi che se ne ottengono sono notevoli ed in particolare trovano facile soluzione i problemi di tempo reale.
Nella progettazione del terminale audio del sistema Musicomp è stata adottata la seguente architettura:



Il uP svolge il ruolo di concetratore,cioè riceve dati dal Master e li invia su richiesta ai uP_i.Esso in realtà realizza una memoria tampone che permette di ovviare a punte di elaborazione particolarmente elevate.I uP_i si occupano esclusivamente dell'elaborazione del segnale e realizzano ognuno un canale audio.

7.- Bibliografia.

- Scelta del linguaggio di programmazione.

Abbot C. - MACHINE TONGUES I/II/III - Computer Music Journal, Vol II,n°1,2,3 July,September,December-1978.

Elson M. - CONCEPTS OF PROGRAMMING LANGUAGES - Chicago: SRA,inc. 1973.

McCracken D.D. - GUIDA ALLA PROGRAMMAZIONE IN FORTRAN IV - Roma: Bizzarri,1972.

Pratt T. - PROGRAMMING LANGUAGES: DESIGN AND IMPLEMENTATIONS - Prentice Hall,1975.

Roads C. - MACHINE TONGUE IV - Computer Music Journal,Vol III, n° 1,March 1979.

Wirt N. - PASCAL USER MANUAL AND REPORT - New York: Springer Jensen K. Verlag,1975.

- Definizione della base dei dati.

Abbot C. - MACHINE TONGUE III - Computer Music Journal,Vol II, n° 3,December 1978.

Baecker R. - THE USE OF HYERARCHY AND INSTANCE IN A DATA STRUCTU-
Buxton W. RE FOR COMPUTER MUSIC - Computer Music Journal,
Leslie M. Vol 2,n° 4,December 1978.
Reeves W.

Bracchi G. - SISTEMI PER LA GESTIONE DI BASI DI DATI,Milano:
Martella G. ISEDI,1979.
Pelagatti G.

- Definizione del linguaggio di comunicazione uomo-macchina.

Aho A.V. - THE THEORY OF PARSING,TRNSLATING AND COMPILING
Ullman J.D. Englewood Cliffs: Prentice-Hill,Vol.1,1972.

Buxton W. - DESIGN ISSUES IN THE FUNDATION OF A COMPUTER-BASED TOOL FOR MUSIC COMPOSITION- Computer System Research Group,University of Toronto,Tecnical Report CSRG-97, October,1978

Crespi Reghizzi S. - LINGUAGGI FORMALI E COMPILATORI - Milano:
Della Vigna P.L.
Ghezzi C. ISEDI,1978.

Holtzman S.R. - AN AUTOMATED DIGITAL SOUND SYNTESIS INSTRUMENT - Computer Music Journal,Vol III,n° 2,June 1979.

Roads C. - GRAMMAR AS REPRESENTATION FOR MUSIC - Computer Music Journal,Vol III,n° 1,March 1979.

Salomaa A. - FORMAL LANGUAGES - New York: Academic Press,1975.

Smoliar - MUSIC PROGRAMS: AN APPROACH TO MUSIC THEORY THROUGH COMPUTATIONAL LINGUISTICS - Journal of Music Theory Vol 20,Spring.

Winnograd T. - LINGUISTICS AND THE COMPUTER ANALYSIS OF TONAL HARMONY - Journal of Music Theory,Vol 12,n° 1, 1968.

- Problemi di tempo reale.

Gries D. - COMPILER CONSTRUCTION FOR DIGITAL COMPUTERS - Traduzione italiana a cura della Franco Angeli Ed.,1977.

- Distribuzione dell'informazione.

Anderson G.A. - COMPUTER INTERCONNECTION STRUCTURES: TAXONOMY,
Jensen E.D. CHARACTERISTICS AND EXAMPLES - Computing Surveyes Vol 7,n° 4,December 1975.

Enslow P.H. Jr. - MULTIPROCESSOR ORGANIZATION: A SURVEY - Computing Survey,Vol 9,n° 1,March 1977.

Una realizzazione del linguaggio PRIMULA

L. TARABELLA - CNUCE, PISA

Riassunto

Viene presentata un'implementazione del linguaggio PRIMULA (PRogramming Interactive MUsic LAnguage) sul Personal Computer CBM 3032 Commodore..

Il linguaggio si basa sulla considerazione che tutti i parametri musicali possono essere espressi con sequenze numeriche e sovrapposizioni di sequenze numeriche (melodie, ritmi, dinamiche, timbri, etc...).

PRIMULA è infatti un linguaggio interattivo ed interpretativo ad alto livello con il quale è possibile definire variabili multidimensionali, funzioni, operazioni complesse (aritmetiche, logiche, interpolazioni, manipolazioni di vario genere) e controllare il flusso delle operazioni con strutture di controllo (if-then-else-, while, etc...).

Il linguaggio prevede un I/O di tipo memory-mapped così che le strutture numeriche elaborate prendono significato al momento della processazione da parte di periferiche opportune.

In questa fase dimostrativa, comunque, il CBM 3032 viene utilizzato anche come generatore sonoro.

Abstract

PRIMULA stands for PRogramming Interactive Music LAnguage; it runs on PET CBM 3032 of Commodore.

The basic idea is that all musical parameters can be expressed as numerical sequences and their overlappings (melodies, rithms, timbres, harmonies, etc...).

PRIMULA is a high level interactive language which allows to define multidimentional array variables, complex operations over them (in the sense of APL), functions, and to control the flow of operation by the use of if-then-else, while, etc. structured programming tools (in the sense of Pascal).
PRIMULA communicates with the out-side world with memory-mapped I/O technique: a specified memory location (or a buffer) corresponds to a particular musical parameter.
In this way, processed numerical arrays take meaning at the time of sound processing, so that both implementation affort and learning process are reduced.

PRIMULA

Una proposta di linguaggio di programmazione musicale

Da una sommaria analisi sull'uso degli elaboratori elettronici in vari campi di applicazione risulta che esistono principalmente due modi di utilizzo: con programmi gia' predisposti ad eseguire determinate funzioni ed in programmazione.

Il primo modo si espleta di solito con un sistema conversazionale, dove l'elaboratore si dichiara predisposto ad eseguire un certo numero di funzioni di base. L'utente non ha che da scegliere una delle funzioni proposte, ed essa sara' immediatamente attivata.

Di solito il tutto viene realizzato mediante la definizione di 'liste di comandi' (o menu') in modo tale che ad ogni comando della lista corrisponde una ben precisa operazione.

Oppure, se l'operazione richiesta ha bisogno di ulteriori specificazioni, il sistema presenta una sottolista di opzioni;questo modo di procedere a sottoliste ramificate implica alcune cose importanti:

-a) il realizzatore del sistema deve avere classificato in qualche modo le operazioni possibili per renderne facile l'utilizzo ed arrivare alla funzione voluta;

-b) le funzioni che il sistema e' in grado di svolgere sono tutte e sole quelle previste dal realizzatore;

-c) non e' detto che fra tutte le funzioni presenti esista quella voluta dall'utilizzatore.

Applicazioni di questo tipo ben si adattano laddove vengano richieste operazioni ripetitive come nella gestione di archivi, magazzini e contabilizzazioni.

L'utente di tali applicazioni non ha bisogno di alcuna preparazione tecnica, e con un minimo di pratica e' in grado di compiere tutte le operazioni del caso.

Esiste una variante al sistema ramificato delle liste e sottoliste di comandi e cioe' quella di associare ad un comando una serie di parametri per arrivare in maniera piu' flessibile alla funzione voluta.

Ma questo e' infatti il sistema appena descritto con la sola differenza che in questo modo si presuppone che l'utente conosca anticipatamente e nel dettaglio le varie possibilita' operative.

Il secondo modo e' quello della programmazione che da' la possibilita' di affrontare i problemi piu' disparati nei piu' disparati modi possibili.

Io ritengo che questa sia la migliore soluzione per affrontare l'argomento musicale con gli elaboratori elettronici.

Infatti l'argomento creativo in se' ha bisogno di uno 'strumento' di per se' creativo.

Tantopiu' che la musica, sia dal punto di vista della codifica grafica, sia dal punto di vista degli elementi che la caratterizzano, possiede molte analogie con i linguaggi di programmazione, al punto di riconoscerle la natura algoritmica.

Vediamo quindi in base a quali considerazioni viene fatta questa affermazione.

Si dice parimenti "eseguire" una brano di musica ed "eseguire" un programma.

Si presuppone allora l'esistenza di:

-a) un 'oggetto' di senso compiuto

-b) un mezzo di codifica

-c) un esecutore.

Nel caso della musica l'oggetto di senso compiuto e' il brano musicale come concezione spirituale prima ed intellettiva poi;

Il mezzo di codifica e', nella tradizione, lo spartito o gli spartiti musicali che rappresentano il veicolo informativo tra il compositore e lo strumentista; l'esecutore e' poi una persona tecnicamente abile ad attivare fisicamente strumenti musicali e di attivarli in base alle precise istruzioni contenute negli spartiti.

E' ovvio che quest'ultima definizione e' restrittiva rispetto a quello che succede in realta'.

Nel caso dell'informatica l'oggetto di senso compiuto e' la procedura concepita; ad esempio un calcolo di ingegneria civile, un analisi lessicografica di un testo, un indagine statistica etc.

Il mezzo di codifica e' un programma redatto in forma di diagramma di flusso e nel linguaggio di programmazione piu' orientato al problema (FORTRAN e Pascal per argomenti tecnico/scientifici, COBOL ed altri per argomenti amministrativi/gestionali etc.).

L'esecutore e' un calcolatore opportunamente equipaggiato in software ed hardware.

Vale la pena di accennare al fatto che brani musicali e programmi di elaborazione dati vedono la loro esistenza drammaticamente legata al momento della loro esecuzione. Questa considerazione di carattere filosofico viene appena accennata e serve solo per ribadire l'affermazione fatta.

Approfondiamo invece l'analisi del raffronto sul punto b), mezzo di codifica.

Un brano musicale e' di solito suddiviso in sezioni o movimenti ed ogni movimento e' costituito da parti ripetibili piu' volte come i ritornelli, rimandi tipo 'da-segno..a-segno..' , 'poi-coda' etc.

Ogni parte e' costituita da frasi che modulano sui vari gradi della tonalita' ed ogni frase e' costituita da una successione di suoni esprimibili con coppie di valori, altezza-durata, piu' informazioni globali sulle modalita' di trattamento delle stesse.

Se proviamo ad esprimere gli stessi concetti nel linguaggio dei 'linguaggi di programmazione', ci si accorge che utilizzando termini quali 'main program' , subroutines, strutture di controllo tipo 'loop' 'while' 'if-then-else' 'goto' e strutture dati tipo 'array' si ottiene lo stesso risultato.

Di qui all'idea di un LINGUAGGIO DI PROGRAMMAZIONE MUSICALE il passo e' breve.

Il linguaggio che io propongo si chiama PRIMULA che significa PRogramming Interactive MUsic LAnguage ed ha le seguenti caratteristiche:

- tratta quantita' numeriche intere tra -32768 e +32767 (16 bits)

- le variabili sono considerate come arrays multidimensionali (una variabile adimensionale nel senso classico e' un vettore di un elemento)

- esistono gli operatori aritmetici logici e di assegnamento, e operatori ad alto livello di intervento sulle variabili: composizione riordino e selezione.

- esistono inoltre operatori per generazioni di sequenze numeriche ordinate e casuali.

- operazioni tra costanti e variabili sono componibili in istruzioni di assegnamento con le regole di precedenza nel senso classico.

- insiemi di istruzioni possono essere sequenzializzate con le strutture di controllo 'while' 'if-then-else' e 'times'.

- esiste inoltre la possibilita' di definire funzioni etichettate, come particolari sequenze di istruzioni.

- le variabili sono in 'COMMON' per tutte le funzioni.

- la comunicazione con l'esterno viene assicurata mediante l'uso di opportune variabili denominate 'canali' per cui assegnare il risultato di una operazione o il valore di una variabile ad un canale, corrisponde ad eseguire una trasmissione dati verso l'esterno.

PRIMULA e' un linguaggio interattivo e puo' essere usato per ottenere risultati immediati (direct), o per definire ed attivare insiemi di funzioni (programming)

PRIMULA genera strutture dati utilizzabili da qualsiasi apparato esterno di generazione sonora, previa definizione di una opportuna interfaccia. Il linguaggio e' infatti orientato alla definizione di strutture dati il cui significato non e' stabilito a priori, ma viene loro dato al momento della esecuzione sonora.

PRIMULA e' un linguaggio che secondo la classificazione di Chomsky si colloca in quelli di tipo 2 o 'liberi dal contesto'.

Vediamone la definizione formale.

Albero sintattico del Primulà

```
[ 1]            <s >  ::= <blank> <oggetto> <blank>
[ 2]         <oggetto>  ::= |<funzione>| <corpo.funz>
[ 3]       <funzione>   ::= <nome.funz>
[ 4]      <nome.funz>   ::= <lett.maius><ric.let.maius><blank>
[ 5]     <lett.maius>   ::= |A| B| C| D| E| F| G .........| T| U| V| Z| X| Y| J| K| W
[ 6] <ric.let.maius>    ::= |<nome.funz>| ▒
[ 7]     <corpo.funz>   ::= <elem.funz><ric.el.funz><blank>
[ 8]      <elem.funz>   ::= |<nome.funz>| <control>| <assegna>
[ 9]        <assegna>   ::= <var>    <espr.arit>
[10]            <var>   ::= |<variabile>| <canale>
[11]      <esp.aritm>   ::= <fattore><operazione>
[12]        <fattore>   ::= |<variabile>| <costante>| <esp.in.parent>
[13]      <variabile>   ::= <lettera><ric.lettera><blank>
[14]       <costante>   ::= <cifra><ric.cifra><blank>
[15]     <operazione>   ::= |<op>| ▒
[16]    <ric.el.funz>   ::= |<aus.el.funz>| ▒
[17] <esp.in.parent>    ::= (<blank><esp.aritm>)<blank>
[18]         <lettera>  ::= |a| b| c| d| e| f| g .... t| u| v| z| x| y| j| k| w
[19]    <ric.lettera>   ::= |<variabile>| <blank>| ▒
[20]          <cifra>   ::= |1| 2| 3| 4| 5| 6| 7| 8| 9| 0
[21]      <ric.cifra>   ::= |<costante>| <blank>| ▒
[22]         <canale>   ::= $ <espr.aritm>
[23]        <control>   ::= [ <corpo.contr> ]
[24]           <cond>   ::= <relazione><ric.relaz><blank>
[25]      <relazione>   ::= <esp.aritm><op.rel><blank><esp.aritm>
[26]       <ric.relaz>  ::= |<aus.ric.rel>| ▒
[27]    <aus.ric.rel>   ::= <op.logico><blank><relazione>
[28]         <op.rel>   ::= |<| =| >
[29]      <op.logico>   ::= |&| ||  -
[30]         <blank>    ::= |<spazio>| ▒
[31]    <corpo.contr>   ::= |<if>| <while>| <times>
[32]            <op>    ::= <operatore><blank><esp.aritm>
[33]      <operatore>   ::= |+| -| *| /| ▒| ┤| ┐| ┘| ┴| @| ||  /| %| ?| ▗| ▾
[34]    <aus.el.funz>   ::= ;<blank><corpo.funz>
[35]         <spazio>   ::= «<ric.spazio>
[36]      <ric.spazio>  ::= |<spazio>| ▒
[37]            <if>    ::= if <cond>
[38]         <while>    ::= while <cond>
[39]         <times>    ::= times <espr.arit>
[40]       <nomefun>    ::= <lett.maius><ric.let.maius>
```

Vediamo in breve gli operatori:

+ - * / operatori aritmetici

← operatore di assegnamento

= uguale

> maggiore

< minore

; : delimitatori

! genera una sequenza ordinata di valori tra limiti specificati

? genera un valore casuale tra due valori specificati

% genera una sequenza di valori interpolati tra due coppie di valori specificati

↓ estrae una sottosequenza a partire dall'indice piu'basso

↑ estrae una sottosequenza a partire dall'indice piu' alto

⍪ compone due arrays di eguali dimensioni aggiungendo una dimensione

, compone due arrays di dimensioni uguali a meno di una (concatena)

[indicizza un elemento di un array

REALIZZAZIONE

Nella sua fase prototipale il linguaggio Primula e' stato realizzato intorno al sistema PET Commodore 3032.

Il software e' costituito da 5 moduli principali: l'editore, l'analizzatore sintattico, il compilatore, l'interprete e l'esecutore sonoro.

L'hardware e' costituito da una console di sistema (video e tastiera), memoria centrale di lavoro da 32K bytes, memoria periferica a mini-floppy-disks, ed un convertitore digitale analogico (DAC) a 8 bits per la generazione sonora.

Il software

L'editore.

L'editore e' 'cursor oriented' nel senso che le 1000 (40x25) posizioni dello schermo sono raggiungibili mediante lo spostamento del cursore e naturalmente dall'avanzamento progressivo per tutti gli altri tasti di carattere.

Inoltre all'interno di ogni riga si possono eseguire operazioni di inserimento e cancellazione caratteri.

Per il momento il video costituisce una pagina insuperabile nei limiti delle 25 righe e delle 40 colonne; e' prevista comunque l'introduzione della funzione di 'scroll' che permettera' l'edizione di funzioni di lunghezza maggiore.

L'editore entra in funzione al momento dell'attivazione del Primula; 'esce' quando vengono battuti i tasti SHIFT/RETURN; viene riattivato dopo un'analisi sintattica con esito positivo e/o la fase di interpretazione.

L'analizzatore sintattico.

Le istruzioni e le funzioni che vengono preparate con l'editore devono seguire la sintassi del linguaggio vista in precedenza.

Durante l'analisi viene eseguita la traduzione del testo in una forma intermedia (compilazione), che rende piu' efficiente la fase di interpretazione.

Un errore di sintassi rilevato durante l'analisi provoca l'arresto dell'analisi stessa e della compilazione ed il ritorno all'editore con la segnalazione del punto d'errore per mezzo dell'inversione di campo (bianco su nero) del carattere.

Il compilatore.

Il compilatore genera un codice intermedio, una sorta di P-code del Pascal che sarebbe il caso di chiamare Pri-code, che semplifica e rende piu' veloce il funzionamento dell'interprete: i dati numerici vengono tradotti in forma binaria; le istruzioni vengono messe in forma inversa polacca.

Il Pri-code e' in sostanza una forma 'tokenizzata' del testo sorgente; in piu' salti condizionati e non condizionati generati in seguito alla presenza nel testo di strutture di controllo 'while' 'if-then-else' e 'times', vengono messi in forma relativa cosi' da permettere la rilocazione in memoria.

Il compilatore esegue collateralmente anche operazioni di trattamento di files: quando una funzione e' stata editata correttamente il codice generato viene catalogato in memoria centrale e il testo sorgente viene catalogato sulla memoria permanente a floppy-disks; codice generato e testo sorgente, se gia' esistevano, vanno a sostituire le rispettive precedenti versioni.

L'interprete.

L'interprete e', per cosi' dire, il cuore esecutivo del Primula: viene attivato non appena sia stata data una istruzione in forma diretta. Qualora l'istruzione sia il richiamo ad una funzione precedentemente definita e compilata, l'operazione assume il significato di attivazione di un processo piu' complesso e cioe' di un programma.

L'interprete esegue operazioni aritmetiche e logiche; alloca dinamicamente in memoria risulati intermedi e permanenti ottimizzando l'uso della memoria stessa.

Esecutore sonoro.

Come detto precedentemente il Primula vien proposto in versione di prototipo con le limitazioni del caso.

Nella sua versione definitiva, al sistema ospite, PET 3032 o altri, verra' delegato solo di eseguire programmi e di conseguenza generare insiemi di dati da trasmettere ad apparati esterni atti alla generazione sonora.

Nella versione attuale il PET 3032 agisce anche da generatore sonoro, vale a dire che il sistema 'simula' l'apparato esterno in partizioni di tempo.

Prima di descrivere questo modo di operare sono necessarie alcune premesse:

- i canali di ingresso/uscita coincidono con zone di memoria interna prefissate e diverse da quelle riservate alle funzioni ed ai dati;

- il metodo di sintesi utilizzato e' la modulazione di frequenza di Chowning.

- i canali a disposizione, che come 'si ricordera' vengono individuati con un $ seguito da un numero, sono i seguenti:

```
$0   video
$1   tastiera
$2   frequenze          (buffer di 255 valori)
$3   durate             (buffer di 255 valori)
$4   inviluppo volume (buffer di 255 valori
$5   invil.indice.mod (buffer di 255 valori)
$6   rapporto f/m
$8   disco
```

- spero che non sia difficile immaginarsi l'interprete come un algoritmo riproducente un vero e proprio 'ciclo macchina' per cui ad ogni codice che incontra esegue le operazioni corrispondenti.

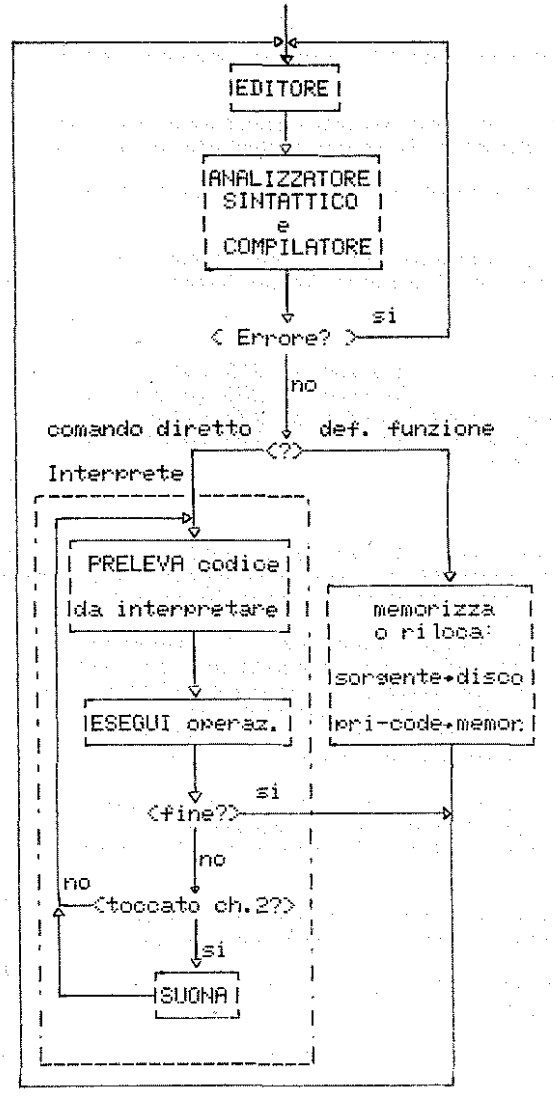
Orbene, ad ogni ciclo macchina, l'interprete si chiede se nel precedente ha eseguito una assegnazione sul canale 2, e cioe' quello delle frequenze; se questo e' avvenuto allora viene interrotto il normale ciclo di interpretazione ed il sistema 'diventa' un esecutore sonoro attraverso l'attivazione di una routine di generazione di segnale digitale.

Durante l'esecuzione, i parametri della modulazione di frequenza vengono prelevati dalle zone di memoria in cui avremo avuto cura di caricare opportuni valori, attraverso l'assegnazione ai rispettivi canali.

In particolare ad ogni frequenza viene associata la durata corrispondente alla stessa posizione dei buffer (o canali) 3 e 4.

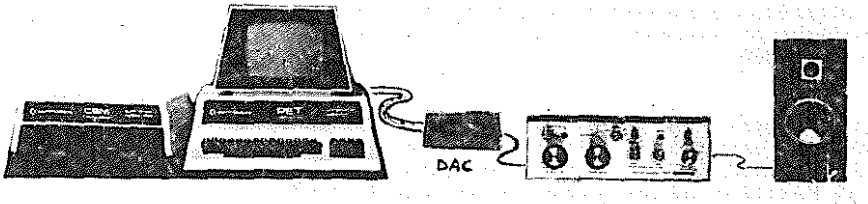
Terminata l'esecuzione sonora, viene ripristinato il normale ciclo di interpretazione.

Si confronti di seguito il diagramma a blocchi riassuntivo.


EDITORE
ANALIZZATORE SINTATTICO e COMPILATORE
< Errore? >
si
no
comando diretto
<?>
def. funzione
Interprete
PRELEVA codice da interpretare
ESEGUI operaz.
<fine?>
si
no
no
<toccato ch.2?>
si
SUONA
memorizza o riloca
sorgente→disco
pri-code→memor

L'hardware

Il sistema e' attualmente quello mostrato in figura:



Il PET 3032 ed il Dual Drive Floppy Disck 3040 vengono utilizzati normalmente, e poco c'e' da dire in proposito.

Dedichiamo invece una particolare attenzione alla parte di conversione digitale-analogica.

Il circuito, costruito appositamente, consta di un DAC ad 8 bits in chip e di un filtro passa-basso di 'smoothing'; il DAC ha una nonlinearita' dello 0.1% su tutto il range operativo ed un settling-time di 1µs.

Fisicamente si connette al porto parallelo a 8 bits disponibile sulla parte posteriore del PET 3032.

Il sistema audio che deriva dalla combinazione soft/hard per la generazione sonora ha le seguenti caratteristiche:

range di frequenza: 1hz - 5Khz
risoluzione di fre: <1hz
metodo di sintesi : FM ad una sola modulante
inviluppo volume : 255 valori
range 0-max : 128 steps
inviluppo ind.mod.: 255 valori
range 0-max : 128 steps
dinamica : 54 db

Prospettive: la polifonia

Le soluzioni piu' appetibili individuate tra le molte prese in esame per un sistema polifonico controllato dal linguaggio PRIMULA sono tre:

-1) Il linguaggio e' provvisto di un alto numero di canali di uscita raggruppati in modo che ciascun gruppo rappresenti i parametri di una voce o strumento.

Una interfaccia hardware opportuna possiede un corrispondente numero di buffers di memoria e prepara i dati per un apparato di sintesi tipo DMX-1000.

-2) Come sopra, con la differenza che l'interfaccia ed all'apparato di sintesi vengono sostituiti da un apparato modulare, ciascun modulo costituito da: un u-processore, buffers corrispondente ai canali relativi ad una voce, programma di sintesi e un DAC.

La miscelazione audio viene fatta in maniera analogica.

-3) Forse la piu' affascinante: il sistema ospite del linguaggio prepara le funzioni (edit e compilazione) e alimenta con la parte compilata moduli esterni, ciascun modulo costituito da: una sezione a u-processore che supporta l'interprete Primula in ROM e una zona di memoria RAM per programmi e dati, ed una sezione a u-processore per la sintesi.

Per queste due soluzioni non e' difficile pensare, con opportuno uso di DACs, alla realizzazione di effetti spaziali stereofonici e quadrifonici.
a)

Dr. Leonello Tarabella

Divisione Musicologica del
CNUCE/CNR - via S.Maria 36
56100 - PISA - t.050/45245

'Un'applicazione musicale della teoria gestaltica sulla percezione di strutture temporali'-Roberto Doati Rossi, Conservatorio B.Marcello,Venezia.

I principi descrittivi formulati dalla scuola psicologica della Gestalt sono utilizzabili per la composizione di opere musicali.Gli stessi fondatori di tale scuola hanno fornito numerosi esempi musicali,e più in generale sonori,per dimostrare le loro teorie.
Dopo una breve esposizione di queste teorie,si analizzano alcune delle esperienze che hanno portato ad una composizione,realizzata mediante computer,la cui struttura è definita e controllata da alcune leggi della Gestalt.

'A musical application of the Gestalt theory on the perception of temporal structures'

Abstract.
The descriptive principles which have been formulated by the Gestalt school seem utilizable for music composition. Several musical examples(and acoustic in general)have been provided by the same founders of this school to prove their theories.
After a brief exposition of these theories,some of the experiences which have led to a composition,realized by means of a computer,whose structure is defined and controlled by some Gestalt laws are analyzed.

Da parecchi anni ormai,l'utilizzo di una disciplina quale la psicoacustica si è dimostrato valido per lo studio della percezione dei suoni.
Anche per la conoscenza dei principi che in qualche modo regolano la percezione delle forme musicali,può risultare efficace un'indagine di tipo psicologico.
Quella della Gestalt è la scuola psicologica che più di ogni altra si è occupata del problema della percezione delle forme,non solo visive,ma anche uditive.Numerosi esempi musicali e più in generale sonori,sono infatti stati formulati dai fondatori di tale scuola per dimostrare le loro teorie.
Nel 1890 von Ehrenfels nel suo articolo sulla psicologia delle qualità delle forme,pose due criteri fondamentali:
1.le strutture(sia spaziali che temporali)anche se scomponibili in elementi,non sono riducibili ad essi,e 2.possono essere trasposte senza subire alcun cambiamento percettivo.
Ciò conduce alla seguente domanda:se una melodia non è determinata dai singoli suoni che la compongono,quali sono le ragioni per cui viene percepita come un'unità organica?
Per risolvere tale problema i promotori della Gestalttheorie (Koffka,Köhler,Wertheimer)respinsero definitivamente l'idea di sensazione cui era rimasto legato von Ehrenfels,e tramite esperimenti sulle condizioni delle forme e sulle leggi delle loro trasformazioni,giunsero a formulare i principi da cui dovrebbe dipendere la percezione delle forme.
Alcuni dei più noti sono:

1. legge della vicinanza:tra elementi vicini esiste un legame più forte rispetto a quelli lontani.
2. legge della somiglianza:in presenza di una moltitudine di elementi diversi,si organizzano in gruppi gli elementi somiglianti.
3. legge della buona continuazione:elementi che si susseguon

in una data direzione,tendono ad essere unificati.
A questi fatti va aggiunto quello della discriminazione della figura da uno sfondo:gli elementi unificati dalle leggi precedenti, costituendo un contorno,portano alla formazione di una figura(definita,più vicina,interessante,significativa,più piccola)su uno sfondo(amorfo,indefinito,più distante,neutro,più grande).
(Occorre ricordare che questi principi di organizzazione sono dimostrati sia nel sistema percettivo visivo,sia in quelli uditivo e tattile.)
I concetti esposti dai gestaltici sono stati impiegati in maniera vantaggiosa,sia per continuare l'indagine sull'ordinamento di stimoli sonori,sia per lo studio in campi quali l'analisi musicale,la formazione di immagini uditive,la percezione di singoli parametri musicali(ritmo,melodia)°.
Una composizione ed alcuni esempi sonori sono,per il momento, i risultati di una ricerca intrapresa dall'autore sulla possibile applicazione musicale delle teorie esposteprecedentemente.
La legge della buona forma,per esempio,favorisce la scelta di forme semplici,regolari,simmetriche,e tali sono le strutture polifoniche da me generate graficamente in uno spazio bidimensionale con coordinate tempo(x)-frequenza(y).
Nell'esempio n.1 si è giunti alla percezione della struttura come 'intero',rendendone pregnanti,e quindi facilmente memorizzabili(anche la memoria ha la sua parte nell'organizzazione percettiva)i contorni.La legge di coesione adottata per questo scopo è quella di buona continuazione;l'esempio comincia infatti con una rapida successione di note(di frequenze

°Per gli interessati a questi argomenti,è disponibile una esauriente bibliografia.

diverse)che dà l'impressione di un moto ascendente.
Il parametro principale di tale fenomeno è l'entry delay delle varie voci;i suoi valori devono essere compresi fra .05 e .3 sec.;valori minori di .05 rendono le note simultanee,valori maggiori di .3 non hanno la forza coesiva sufficiente per evitare che i singoli suoni vengano percepiti come entità separate;questo accade all'interno della struttura considerata,quando il valore di entry delay,progressivamente incrementato,raggiunge .7 sec.La sensazione di chiusura è successivamente creata dal retrogrado della situazione iniziale.
L'esempio n.2 è esattamente l'opposto del precedente.In esso i primi valori di entry delay sono maggiori di .3 sec.,ed una diminuzione nel tempo di tali valori,rende evidenti uditivamente le 'linee' centrali.
Nella composizione cui si accennava prima il grado di complessità è maggiore a causa dell'uso polifonico di sette tipi di forme ritmiche.
Quella della pregnanza(o buona forma)pare sia la legge su cui sono basati i principi dell'unificazione formale;nel caso specifico,da essa dipende la riconoscibilità di una struttura trasposta in un diverso range frequenziale(quattro le ripetizioni di ogni modello nel corso del pezzo)con rapporti intervallari diversi;la stessa legge crea l'alternanza percettiva delle strutture fra figura e sfondo.
Inoltre,poste in opposizione la legge della vicinanza(ampi rapporti frequenziali)e quella di buona continuazione(contorni pregnanti),si è notato il prevalere di quest'ultima sulla prima.
(Per maggiori dettagli si veda la presentazione del nastro: 'Gioco di velocità'.)

Carmelo Genovese C.R.A.U.S. Bologna/Messina

SUONO - COLORE

SOUND - COLOUR

Considerando il fondamentale substrato biologico della percezione visiva e acustica, è possibile ipotizzare una correlazione suono - colore. Idonee ricerche stabiliranno i meccanismi correlativi.
La metodologia adoperata si estrinseca con un trasduttore che genera ritmi, che pilotano luci a colori, e con Personal computer che genera colori e suoni.
Si è così in grado di misurare sui veri soggetti esaminati le reazioni agli stimoli sincroni, determinati dal suono - colore.
I risultati permetteranno un più razionale impiego del suono - colore nel teatro, nella cinematografia, nelle varie correlazioni suono - colore.

Considering the fundamental biological substratum of visual and acoustic perception, it is possible to hypothesize a correlation sound-colour. Suitable research will establish the correlation mechanisms.
The methodology employed is carried out with a transducer which generates rhythms, which pilot coloured lights, and with a Personal computer which generates colours and sounds.
It is therefore possible to measure in the various subjects the reactions to synchronous stimuli, determined by sound - colour. The results will permit a more rational use of sound - colour in the theatre, in the cinema, and in the various correlations sound-colour.

Carmelo Genovese

SUONO - COLORE

1.1 Il fondamento biologico dei suoni e dei colori

Gli organi dell'udito percepiscono il suono secondo leggi fisiche, così come quelli della vista percepiscono i colori con leggi particolari. Il substrato di queste percezioni è biologico ed è riferibile alla frequenza del suono, espressa in herz e alla lunghezza d'onda dei colofi.

Su queste basi è ipotizzabile un'interazione suono - colore anche se non facile ne è la dimostrazione.

2.1 Metodologia di ricerca

Coinvolgendo le scelte estetiche di suono - colore si può esaminare il più complesso fenomeno psicofisico.

Fino a non molti anni fa una tale ricerca era quasi impossibile, sia perchè si riteneva la sfera estetica di dominio filosofico, sia per la difficoltà di disporre trasduttori che permettessero di misurare,in tempo reale, le reazioni a determinati stimoli.

Il vertiginoso sviluppo dell'elettronica e dell'informatica hanno reso possibile l'approccio e lo sviluppo delle ricerche in questione.

2.2 I trasduttori

Abbiamo ideato e realizzato - in collaborazione con G.Berardo- una apparecchiatura(l'adoperiamo nel nostro corso d'estetica sperimentale presso l'Accademia di Belle Arti di Firenze) che permette di studiare le reazioni sincrone a contemporanei stimoli acustici - visivi.

Nell'Agosto del 1980 illustrammo tale apparecchiatura al Convegno internazionale d'Estetica sperimentale a Montreal, in questa sede ne presentiamo il funzionamento:

a) determinati ritmi musicali pilotano luci colorate;
b) sistematicamente si mutano gli accoppiamenti ritmi - colori, al fine d'osservare quali sono le scelte dei soggetti;
c) Variando il ritmo variano le combinazioni dei colori, la luminosità etc..
d) E' in studio un'interfaciamento con il Computer.

2.3 IL PERSONAL COMPUTER

L'avvento del Personal computer ci ha consigliato di sviluppare programmi sia per la ricerca su vari soggetti, sia per la simulazione di determinati meccanismi. Ed ecco alcuni risultati

ottenuti in collaborazione di M.Bartolini:
A) Modificando un programma di D.Cecconi(Bit 80 N°13 pag.105) è possibile la realizzazione di disegni a colori con la randomizzazione di tutte le variabili, correlata ai suoni.
La tav.1 mostra il programma e gli asterischi segnano le linee delle nostre modifiche con altre linee(285 - 560 -600-1040).
Inizializzando il programma (questa avvertenza è indispensabile per il funzionamento) con altro programma (Music), l'introduzione della linea 1030 POKE 1,(X*10) : Call 770 è determinante per la produzione dei suoni sincronizzati.
Oltre a questa sincronizzazione con i colori(sei differenti) stiamo studiando la possibile modifica che permetterà la durata del suono in funzione della lunghezza dei vari tratti che costituiscono la struttura dei disegni.
B) Un nostro programma originale è illustrato nella tavola 2. Disponiamo così dell'automazione di una sequenza di prove atte a permettere la scelta(in tempo reale) dei possibili sincronismi tra 15 colori differenti e 15 note.Il programma è così attuato:

a) Inizializzazione del programma dalla linea \0 alla 90;
b) Input del numero delle risposte (scelte) dei colori,in riferimento alle note sincrone(linee 100 / 150);
c) Generazione e selezione casuale dei 15 colori(1000/1090)
f) Generazione dei suoni(2010).Come in A è indispensabile la inizializzazione con il programma Music(Tav.1);
g) Test per l'automazione delle singole scelte(da 3010 a 3110;
h) Visualizzazione dei risultati delle prove,oltre la linea 4000.

CONSIDERAZIONI FINALI

Il potere disporre di un metodo,che permette l'automazione di prove su molteplici soggetti,lascia bene sperare per il chiarimento di complessi meccanismi che regolano l'interazione suono - colore.
Accenniamo e insistiamo sulla portata pratica che potranno dare tali ricerche, specie per le applicazioni che potranno interessare il teatro, cinema,tv etc..

Bologna Aprile 1981 prof.Carmelo Genovese
del "Centro ricerche sulle attività umane superiori"(CRAUS) Messina - Bologna
via Oberdan 15, 40126 BOLOGNA

```
]LOAD TOND - BEEP                                                    TA, I
]LIST

 10  DIM C(150),X1(150),Y1(150),X2(150),Y2(150)
 20 TA = 1: HGR2 :MX = 279:MY = 191
 30  HCOLOR= 0
 40  HPLOT X1(TA),Y1(TA) TO X2(TA),Y2(TA): HCOLOR= X
 50  IF C1 = 0 THEN C(TA) = 1 +  INT ( RND (1) * 6): HCOLOR= C(TA):C1 = 5 *
     (1 +  INT ( RND (1) * 10)):X = C(TA)
 60  IF C2 = 0 THEN DA =  INT ( RND (1) * 5):DB =  INT ( RND (1) * 5):DC =
     INT ( RND (1) * 5):DD =  INT ( RND (1) * 5):C2 = 5 * (1 +  INT ( RND
     (1) * 10))
 70 PA = X1 + DA
 80  IF PA >  = 0 AND PA <  = MX THEN 100
 90 PA = X1:DA =  - DA
100 X1 = PA
110 PB = X2 + DB
120  IF PB >  = 0 AND PB <  = MX THEN 140
130 PB = X2:DB =  - DB
140 X2 = PB
150 PC = Y1 + DC
160  IF PC >  = 0 AND PC <  = MY THEN 180
170 PC = Y1:DC =  - DC
180 Y1 = PC
190 PD = Y2 + DD
200  IF PD >  = 0 AND PD <  = MY THEN 220
210 PD = Y2:DD =  - DD
220 Y2 = PD
250  HPLOT X1,Y1 TO X2,Y2
260 X1(TA) = X1:X2(TA) = X2:Y1(TA) = Y1:Y2(TA) = Y2
270 TA = TA + 1:C1 = C1 - 1:C2 = C2 - 2
280  IF TA > 150 THEN TA = 1
*285  GOSUB 1030
290  GOTO 30
500  REM
530  FOR BEEP = 1 TO 2
550 S =  PEEK ( - 16336)
555 SOUND =  PEEK (S) -  PEEK (S) +  PEEK (S) -  PEEK (S) +  PEEK (S) -  PEEK
    (S)
*560  NEXT
*600  RETURN
1000  POKE 770,173: POKE 771,48: POKE 772,192: POKE 773,136: POKE 774,208:
      POKE 775,4: POKE 776,198: POKE 777,1: POKE 778,240
1010  POKE 779,8: POKE 780,202: POKE 781,208: POKE 782,246: POKE 783,166: POKE
    784,0: POKE 785,76: POKE 786,2: POKE 787,3: POKE 788,96
1020  POKE 1,0
*1030  POKE 1,(X * 10): CALL 770
*1040  RETURN
1050  GOTO 1030
11300  GOSUB 11700
11700  REM  MUSIC
11710  READ I,J: IF J = 0 THEN  RETURN

]
```

```
]LOAD TEST COLORE SUONO                                          TAV. 2
]LIST

)  DIM SC(15): DIM SS(15)
5  HOME : HTAB 8
.0  PRINT "** TEST COLORE SUONO **"
:5  PRINT : HTAB 8
:0  PRINT "** C R A U S    (BO)   **"
:0  PRINT : PRINT : PRINT
'0  POKE  - 16384,100: REM  PULIZIA TASTIERA
.00  REM  **INIZIO PROGRAMMA**
02  INPUT "N' RISPOSTE  (20-100) ? ";NO
04  VTAB 9: HTAB 1
05  INPUT "UNICA NOTA (SI,NO) ? ";A$: PRINT
06  IF  LEFT$ (A$,1) = "S" OR  LEFT$ (A$,1) = "N" THEN  GOTO 110
07  GOTO 104: REM  HAI SBAGLIATO
10  IF  LEFT$ (A$,1) = "N" THEN  GOTO 150
30  INPUT "N' RIFERIMENTO NOTA (0-50) ? ";MO:MO = MO + 2: IF MO > 50 THEN
    MO = 50
50 N9 = 0
70  REM   PULIZIA TOTALIZZATORI DI SCELTA COLORE SC(C)
80  FOR C = 0 TO 15:SC(C) = 0: NEXT C
000  REM ** GEN. CAS. COLORE **
010  GR : HOME : REM  GEN.BARRA MULTICOLORE
020  FOR I = 0 TO 31
030  COLOR= I / 2
040  VLIN 0,10 AT (I + 3)
050  NEXT I
060 C =  INT (16 *  RND (1)): COLOR= C: REM  GEN CAS COLORE
070  FOR L = 15 TO 39
080  HLIN 1,39 AT L
090  NEXT
000  REM **GENERAZIONE SUONO**
005  IF  LEFT$ (A$,1) = "S" THEN  GOTO 2100
010  POKE 0,(C * 10): CALL 770
050  GOTO 3000
100  POKE 0,(MO * 5): CALL 770
000  REM **TEST DI SCELTA**
010 T =  PEEK ( - 16384): REM  TASTO PREMUTO?
020  IF T < 128 THEN  GOTO 1060: REM  NO!VAI NUOVO COLORE
025  REM  SI! TASTO PREMUTO
030  POKE  - 16368,100: REM  RESETTA TASTO
050  REM   AUMENTA UNO DEI 16 TOTALIZZATORI
060 SC(C) = SC(C) + 1
100  REM   VERIFICA SE RAGGIUNTO NUMERO PROVE
105 N9 = N9 + 1
110  IF N9 <  > NO THEN  GOTO 1060: REM   VISUALIZZA I RISULTATI
000  REM  **VISUALIZZAZIONE**
050  TEXT : HOME
100  PRINT "N'RISPOSTE          : "N9
150  PRINT
180  IF  LEFT$ (A$,1) = "S" THEN  GOTO 4200
190  PRINT : GOTO 4250
200  PRINT "N'RIFERIMENTO NOTE: "MO
250  PRINT : HTAB 8
300  PRINT "DISTRIBUZIONE RISULTATI"
350  PRINT
```

```
4400  FOR C = 0 TO 15
4500  REM    **TOTALIZZAZIONE RIS.**
4550 SS(C) = SS(C) + SC(C)
4600  IF C < 10 THEN Q$ = " ": GOTO 4620
4620  PRINT Q$C"  "SC(C)
4630 Q$ = ""
4650  NEXT C
5000  REM    **VISUAL. TOTALIZZ.**
5100  HTAB 1: VTAB 24: PRINT "VUOI VEDERE LA DISTRIBUZIONE FINALE ? ";: GET
     B$
5200  IF  LEFT$ (B$,1) <  > "S" THEN  GOTO 10000: REM      NON HAI DETTO S
     I !  RICOMINCIO
5250  HOME : HTAB 9: VTAB 2: PRINT "DISTRIBUZIONE TOTALE": PRINT
5270 AA$ = "  A NOTA FISSA  N° "
5280  IF A$ = "S" THEN  PRINT "        "AA$MO: GOTO 5300
5290  PRINT
5300  PRINT : FOR C = 0 TO 15: REM  HAI DETTO SI
5400  IF C < 10 THEN Q$ = "  "
5450  IF C < 100 AND C > 9 THEN Q$ = " "
5500  HTAB 15: PRINT Q$C"  "SS(C):Q$ = ""
5600  NEXT C
5700  FOR P = 1 TO 9000: NEXT P: GOTO 150
10000  IF  LEFT$ (B$,1) <  > "N" THEN  GOTO 5100: REM  NON HAI DETTO SI
10100  GOTO 150: REM  HAI DETTO NO

]
```

INFORMATICA, MUSICA, TEATRO

Progetto di teatro cibernetico.

Loreto Papadia, collab. del C.S.C. di Padova
Sergio Cavaliere, Ist. di Fisica Sperimentale di Napoli

Riassunto

Viene presentato e discusso un lavoro in programma al Festival dei Due Mondi di Spoleto '8I.
Si tratta della realizzazione di un sistema di controlli e feedback in cui tutti gli elementi della messa in scena sono regolati da una logica di interrelazioni strutturate sui tempi della musica prodotta presso il C.S.C. dell'Università di Padova.
Vengono evidenziati gli aspetti teorici e tecnici connessi alla creazione di un tale sistema computerizzato, ed esaminate alcune implicazioni estetiche connesse al suo impiego.
Il sistema interattivo è costituito da un calcolatore della classe mini/personal computer che controlla un complesso di apparecchiature appositamente progettate. Questo calcolatore aggiorna sotto programma i parametri da inviare al mondo esterno, legando spesso eventi o sequenze di eventi all'elaborazione di segnali ottici e acustici prelevati dalla scena teatrale, in primo luogo da movimenti della danza e scenici.

Abstract

A theatrical work included in the program of the Festival of Two Worlds, Spoleto '8I is presented and discussed, involving the realization of a system of controls and feedback in which all elements of the mise-en-scène are regulated by a logic of interrelations structured on the music produced at the C.S.C. of the Univ. of Padua.
The theoretical and technical aspects of creating such a computerized system are shown and the aesthetic implications of its employment are examined.
The interactive system is constructed from a calculator of the mini/personal computer class, which controls a complex of devices purposely designed.
This computer updates, by program, the parameters to be sent to the external world: stage motors, laser beams, synthesis of sound effects, reproduction system, lighting, etc., often connecting events or sequences of events with the elaboration of optical and acoustic signals drawn from the action and movement on the stage.

PREMESSA

Le applicazioni dell'informatica alla ricerca musicale sono caratterizzate essenzialmente da due tendenze principali per quello che riguarda la quantità e la qualità dei lavori svolti.
La prima tendenza è rivolta verso la parte più astratta della musica, la formalizzazione dei processi compositivi; la seconda resta spesso su un piano esclusivamente scientifico e consiste nelle ricerche di analisi e di sintesi.
Per quanto affascinanti possano essere per chi li conduce, tali livelli di ricerca non sempre lo sono anche per chi ne fruisce i prodotti finali.
Non si discute qui la fondamentale validità di queste ricerche, però si vuole sottolineare il grosso dislivello che esiste tra queste e gli studi sulla comunicazione musicale, meno aristocratici ma forse più importanti.
Alla comunicazione musicale sono legati una notevole quantità di problemi, quali la fruizione, la partecipazione, i sistemi di diffusione ed esecuzione in tempo reale, la relazione tra i suoni sintetici e quelli naturali, gli ambienti di ascolto, ecc.
Bisogna considerare il fatto che lo studio e l'esperienza pratica di questi problemi possono suggerire nuove direzioni di ricerca o anche soluzioni tecniche ed estetiche a questioni correnti.
I lavori svolti in questo senso non sono molti, per lo più sono stati eseguiti da operatori e artisti di altre discipline culturali e consistono di solito in piccole performances. Questi esperimenti anche se mantengono tutti il grosso difetto della limitatezza dell'esperienza e del formalismo, forniscono però un utile materiale di studio per applicazioni più approfondite e comprensive quali le azioni teatrali.

TEATRO CIBERNETICO E MUSICA

Un primo esperimento teatrale in cui vengono fatte delle applicazioni informatiche a larga scala è in programmazione al Festival dei Due Mondi di Spoleto '81, si intitola "Libertà a Brema, la regia è di Maurizio Di Mattia e Anna Brasi, la realizzazione del Teatro Stabile dell'Aquila. Agli autori della presente memoria è stata affidata la composizione delle musiche generate dall'elaboratore e l'allestimento cosiddetto cibernetico.a cui ha collaborato per alcuni programmi software M.Missikov.

Inizialmente l'uso dell'informatica doveva essere limitato alla produzione della musica, ma poi, in seguito ad una serie di implicazioni a catena tra i vari elementi della rappresentazione, si è deciso di informatizzare tutta l'azione scenica.

L'informatica permette di creare dei complessi sistemi di controllo, automazione e feedback che applicati al teatro provocano un'estensione delle sue capacità comunicative. Lo studio del rapporto tra la comunicazione e questi sistemi, di qualsiasi tipo essi siano appartiene alla Cibernetica,quindi una forma di teatro che vada in questa direzione è un esperienza cibernetica.

Una ricerca teatrale così impostata, oltre che all'apertura di possibilità nuove, conduce anche allo stravolgimento dei tradizionali schemi delle rappresentazioni. Diventa tecnicamente possibile costruire un corpo unico di interazioni in cui si fondono tra di loro la dimensione sonora con quella visiva e spaziale.

In questo progetto si è tentato di estendere il più possibile l'applicazione dei concetti cibernetici,Inizialmente tutti gli aspetti della scena sono stati considerati su di un piano uniforme; tuttavia la musica è la parte che si arricchisce di più con l'uso di tecniche elettroniche.

Innanzitutto la parte musicale, nell'ambito di una struttura di di interazioni spaziali e temporali, viene ad assumere il ruolo di elemento connettivo e unificatore di tutta l'azione scenica. Poi arriva a comprendere nella sua forma anche le voci degli attori prelevate da microfoni ed elaborate in tempo reale dal complesso di controlli che verrà descritto più avanti.

STRUTTURA GENERALE DELLA RAPPRESENTAZIONE

Con l'assorbimento delle voci da parte della musica si ha anche un ridimensionamento della parte recitativa. Il risultato è un organismo formato da elementi da percepire senza un significato discorsivo, il tempo di svolgimento dell'azione non è direzionale ma accumulativo. Il fatto che non esista un inizio, un culmine ed una fine consente al sistema interattivo dei feedback e dei controlli, di svolgere le sue funzioni di omeostasi in modo ripetitivo e perciò più facilmente formalizzabile.

Gli elementi coinvolti dai controlli del sistema sono: la danza, i movimenti delle scene, le voci dirette degli attori e quelle registrate su memorie digitali e su nastro magnetico, le luci, uno speciale strumento per il mixaggio automatico di bande magnetiche registrate, la diffusione sonora.

L'attività di tutti questi elementi costituisce una struttura generale dinamica che può essere rappresentata da un insieme di piani paralleli sui quali sono inserite le funzioni delle diverse interazioni.

Tutto l'andamento dello spettacolo segue una logica a funzioni circolari prestabilite da un programma di regia che rimanendo sempre costante, permette alle singole parti di compiere delle piccole variazioni in tempo reale.

DESCRIZIONE DEL SISTEMA DI INTERAZIONI (Fig.1)

La gestione e il controllo di tutto l'apparato è affidato a due personal computer che nei momenti di maggior impegno si dividono

il lavoro da svolgere.
La loro funzione essenziale è quella di seguire le azioni dei singoli elementi della scena tramite una rete di sensori e di una telecamera e rispondere tramite feedback ai messaggi ricevuti. I movimenti della danza interagiscono sulla musica che è organizzata su linee separate e registrate su otto canali. I movimenti del corpo possono in questo modo modificare in modo rilevante l'ascolto dell'insieme di queste linee rendendo così realizzabile l'idea di intervenire mediante movimenti lo spazio sonoro, di fare in qualche modo musica con il corpo.
Questo aspetto è particolarmente interessante per l'enorme potenzialità di sviluppo che contiene in sé. Infatti a parte il fatto della danza qui viene realizzata la possibilità di interagire con una musica già registrata su di un nastro, oggi il principale mezzo di diffusione ed esecuzione musicale insieme al disco. Il suggerimento che viene proposto è insomma l'ascolto interattivo, in alternativa alla fruizione passiva della musica. Tornando alla danza essa ha anche la possibilità di modificare in frequenza dei suoni prodotti in tempo reale.
Per quanto riguarda la voce diretta, essa viene elaborata mediante l'uso di un vocoder in cui si invia, come segnale portante il suono prodotto da semplice sintetizzatore digitale i cui parametri di frequenza e ampiezza vengono stabiliti dai gesti degli attori. Il risultato è una recitazione distorta e più musicale.
Agli attori e ai danzatori è ancora lasciato il compito di attivare il sistema per gli spostamenti delle scene. Questo fatto è molto importante perchè contribuisce a rendere attraverso il movimento l'idea della disarticolazione delle strutture fisse(il concetto chiave dello spettacolo), insieme alla musica e a tutto il resto.

LA MUSICA

La musica è stata composta seguendo lo stesso tipo di strutturazione dell'intera messa in scena, in modo da potersi inserire nel suo organismo e guidarne la fluidità per tutta la sua durata.

Sono stati creati alcuni modelli base, che sviluppandosi in modo variabile nel tempo creano delle linee bidimensionali parallele. Questo parallelismo genera una forma, che pur essendo costituita da più suoni simultanei non ha niente a che vedere con la tridimensionalità polifonica e tantomeno politonale perchè non vengono usate tonalità specifiche.

Facendo riferimento più sugli aspetti ritmici che melodici, nella creazione di queste linee, si è tentato di sostituire alla struttura tridimensionale specifica una simultaneità di suoni diversa, la quale più che esprimersi in termini di tridimensionalità andasse alla ricerca di una più ricca e percepibile bidimensionalità.

Tutto il materiale musicale e vocale (alcuni personaggi dello spettacolo esisistono solo come voci sintetiche) è stato ottenuto presso il Centro di Sonologia Computazionale dell'Università di Padova.

Alla produzione di questo materiale hanno collaborato: S. Farneda e T. Patella per la musica, G. Tisato per la sintesi della voce mediante predizione lineare.

I modelli per le linee musicali sopra accennate sono stati sviluppati mediante l'uso di alcune PLF e CONVT in MUSIC5 disponibili presso il C.S.C. di Padova. Le aggiunte, gli effetti e ricerche sui suoni da usare sono stati ottenuti tramite il MUSIC360 e il sistema interattivo per la sintesi ICMS.

ALCUNI ASPETTI REALIZZATIVI

A) Aspetti generali

L'organizzazione del sistema di controlli e di attuatori è mostrata in figura 2 per quanto riguarda le connessioni hardware. Le unità d'ingresso e di uscita sono poi connesse tra di loro da programma, quindi in un modo flessibile. Le informazioni dal mondo esterno sono prelevate ed eventualmente preelaborate a velocità sufficienti a realizzare il funzionamento in tempo reale. E' il caso soprattutto della telecamera in cui il segnale video viene preelaborato da hardware veloce dedicato, che calcola tutti i parametri necessari, mentre il microprocessore deve soltanto leggere sui registri di uscita questi parametri per poterli utilizzare.

Analogamente per le unità di uscita è stata prevista sufficiente logica specializzata da consentire al personal di alleggerirsi i compiti che comportano dispendio di tempo; è il caso soprattutto del banco di motori passo passo a cui il personal deve soltanto fornire velocità e posizione finale. Il successivo compito di fornire i segnali necessari a controllare l'avanzamento è affidato a logica discreta esterna. Analogamente per il sistema di selezione e inviluppo e di mixaggio di otto canali audio preregistrati: il personal deve soltanto fornire dei parametri mentre il compito di realizzare inviluppi anche molto lenti è svolto da un microprocessore dedicato a questo compito.

L'hardware dunque è organizzato in modo tale da realizzare complessivamente una macchina che reagisce in tempo reale ad una vasta gamma di eventi esterni elaborandone, da programma, il contenuto di informazione e restituendolo al mondo esterno.

L'uso intensivo di controlli manuali acquisiti tramite ADC ed elaborati da programma consente inoltre un funzionamento interattivo di tutto l'insieme.

Le connessioni tra le unità d'ingresso e di uscita vengono fatte tutte da programma e sono quindi definibili di volta in volta per gli scopi individuati dalla regia, definizione che può anche cambiare dinamicamente e in modo sperimentale: questo dà una grossa flessibilità al sistema.
Le unità d'ingresso e di uscita sono comunque concepite in senso generale secondo le seguenti linee. I motori passo passo sono previsti per realizzare movimenti controllati di scena (pannelli, carrelli ed altro) e movimenti delle luci. L'idea di fondo è di definire dinamicamente la scena realizzandone variazioni coordinate anche lente che, sottolineando o producendo eventi scenici, rendano l'idea di una macchina teatrale in movimento. Oltre che da programma, in base alle esigenze di regia, queste modificazioni possono essere controllate da eventi esterni tramite le informazioni sul movimento degli attori o danzatori prelevate da sensori ottici e meccanici (o anche da telecamera). Il sistema basato su telecamera raccoglie e preelabora informazioni provenienti soprattutto dalla danza fornendo misure di parametri quali la posizione dei danzatori, il loro numero, la velocità, ecc.
Queste informazioni vengono tramutate da programma in eventi sonori (inviluppi, entrate e mixaggio di voci già prodotte digitalmente e preregistrate o anche controllo, specie in frequenza ed ampiezza, di semplici fasce sonore prodotte in tempo reale).
Il banco di controlli manuali infine, necessari a realizzare un sistema interattivo, regola parametri definibili da programma e è quindi di uso piuttosto libero.

B) Il banco di motori.

Il banco è costituito da un massimo di trentadue motori passo p so a controllo digitale a quattro fasi di varia potenza e con l risoluzione di 0,8° (vedi figura 3). Il controllo realizzato in

logica discreta può fornire trentadue velocità diverse in ambedue i sensi di rotazione ed una definizione della posizione incrementale a I6 bit - con un massimo quindi di 65536 step corrispondenti a circa 3277 giri.-. E' inoltre previsto un controllo di moto continuo per usi particolari ed un sensore meccanico che permette di raggiungere posizioni assolute prefissate.
Ognuno dei motori viene visto nello spazio di memoria dell'Apple come un gruppo di tre byte in cui va scritto il valore di velocità e di posizione incrementale e da cui va letto sia una flag che indica il raggiungimento della posizione richiesta, sia un bit che segnala l'intercettazione di un sensore meccanico sullo asse del motore (quindi una posizione assoluta).

C) I sensori ottici.

I sensori ottici sono costituiti da coppie di trasmettitori (ad infrarossi o in luce visibile) e di ricevitori (fotodarlington ad alta sensibilità) montati otticamente in modo da garantire alta immunità alla luce ambientale. Essi possono, se abilitati da programma, fornire un interrupt che darà corso ad una routine di attuazione di comandi predisposti. Lo stato dei sensori istante per istante ed il vettore di interrupt vengono visti come locazioni di memoria dall'Apple.

D) Il sistema di elaborazione delle immagini.

E' costituito da una telecamera (Bell & Howell) il cui segnale video viene digitalizzato tra due soli livelli (bianco-nero) per creare un'immagine completamente contrastata.
L'immagine viene inoltre quantizzata su 256 punti per ogni riga che corrispondono, in un campo visivo di 4 metri, ad una risoluzione elevata di I.56 cm. Su questa immagine viene fatta in tempo reale una elaborazione (con componenti discreti) che permette di apprezzare parametri come area della figura ripresa (a I6 bit),

posizione e velocità del baricentro (a 8bit), contorno dell'immagine (riga più in alto , più in basso,prima riga a destra, prima a sinistra e tutte a 8 bit), momento di inerzia della figura (dispersione delle masse - rapporto contorno/superficie a 8 bit), area di 4 o 8 settori verticali (a 8 bit).
Molte informazioni vengono integrate nel tempo o nello spazio per renderle più significative. Da tutte le informazioni possono essere ricavate le relative grandezze incrementali.
Tutti i parametri calcolati appaiono al micro processore come locazioni di memoria in lettura.

E) Il sistema generale di inviluppi e di mixer

Trattandosi di un compito ripetitivo con alto dispendio di tempo, è stato previsto un sistema a microprocessore che realizza permanentemente I6 rampe (normalmente lineari ma anche ad esempio esponenziali) tra un valore iniziale ed uno finale programmabili (a 8 bit). Raggiunto il valore finale, la rampa si arresta, mentre raggiunto lo zero disconnette mediante un interruttore a JFET(National LF13331) il canale.
La velocità di incremento (step della rampa) è definita a 16 bit più il bit del segno, consentendo così inviluppi tra un valore minimo 10msec ed uno massimo 65536 volte maggiore.
I campioni così calcolati vengono inviati ad un banco di 16 DAC multipyng (National DAC 3032) controllati da 8 bit.
I parametri dell'inviluppo: iniziale, finale e step sono scritti nella memoria del microprocessore del personal computer in DMA in modo asincrono. La sezione di missaggio e di selezione, realizzata con interruttori a JFET ed operazionali è anch'essa interamente programmabile. L'intero sistema permette di selezionare ed attaccare segnali inviluppando con velocità e legge programmabili 8/16 canali scegliendo anche, dinamicamente, il canale audio su cui smistarli (per consentire effetti di spazializzazione e movimento del suono).

BIBLIOGRAFIA:

1. M.J.Apter, Cybernetics and Art, Leonardo 2, 257(1969).
2. H.W.Franke,Acybernetic Approach to Aesthetics,Leonardo 10,(1977).
3. S.Ceccato, La Mente Vista da un Cibernetico, ERI (1972).
4. T. Shank, Theatre in Real Time, Studio forma ed.(1980).
5. J.Quiniou,J.M.Philippe, ICervelli Artificiali,Sansoni(Firenze1972).
6. Tisato,Vidolin,De Poli, Manuale Operativo del Sistema Musica, C.S.C. di Padova.
7. R.Goldberg, Performance:the Art of Notation, Studio Inter.
8. M.Nyman, Hearing/Seeing, Studio International.

SISTEMA DELLE INTERAZIONI

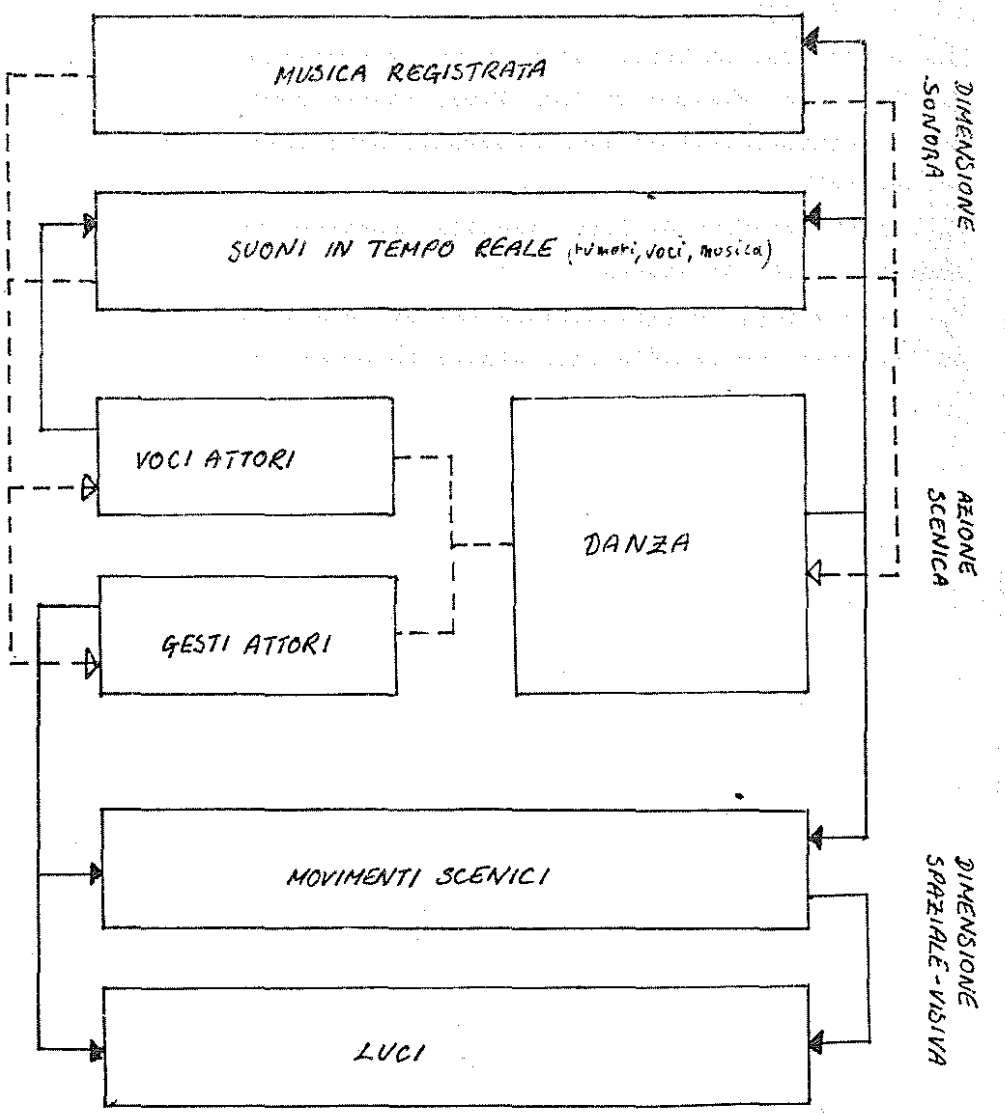


FIG. 1

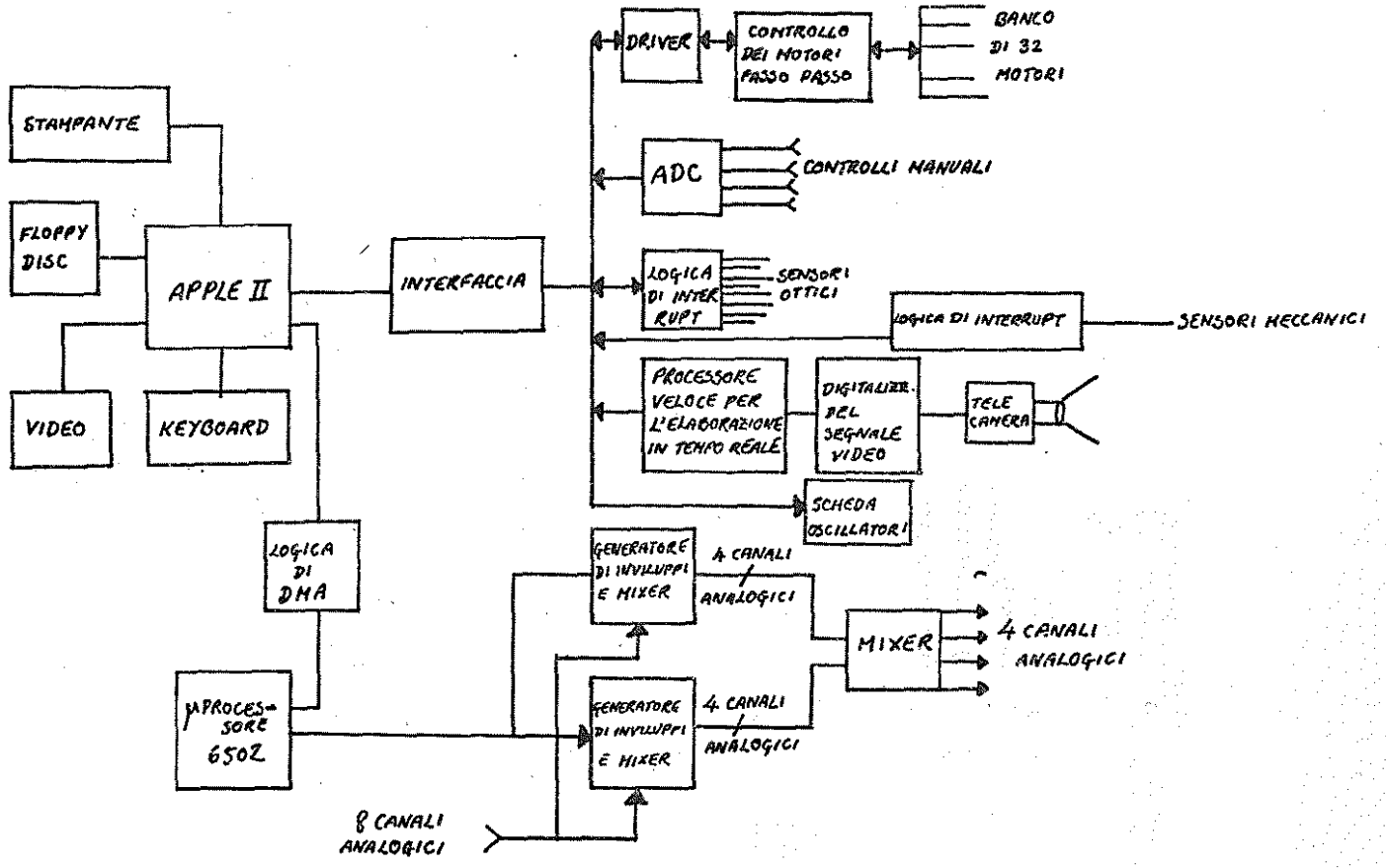

STAMPANTE
FLOPPY DISC
APPLE II
INTERFACCIA
VIDEO
KEYBOARD
LOGICA DI DMA
PROCESSORE 6502
DRIVER
CONTROLLO DEI MOTORI PASSO PASSO
BANCO DI 32 MOTORI
ADC
CONTROLLI MANUALI
LOGICA DI INTERRUPT
SENSORI OTTICI
LOGICA DI INTERRUPT
SENSORI MECCANICI
PROCESSORE VELOCE PER L'ELABORAZIONE IN TEMPO REALE
DIGITALIZZ. DEL SEGNALE VIDEO
TELE CAMERA
SCHEDA OSCILLATORI
GENERATORE DI INVILUPPI E MIXER
4 CANALI ANALOGICI
GENERATORE DI INVILUPPI E MIXER
4 CANALI ANALOGICI
MIXER
4 CANALI ANALOGICI
8 CANALI ANALOGICI

FIG. 2

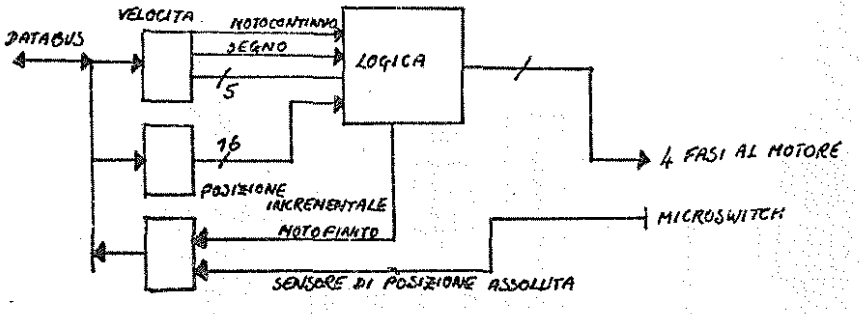


FIG. 3
IL BANCO DI MOTORI

STOCHASTIC MUSIC : COMPOSITIONAL TECHNIQUES AND APPLICATIONS

KEVIN JONES
Music Department
The City University
London EC1.

This paper briefly outlines a progression of musical applications of stochastic techniques ranging from simple probability distributions to generative grammars. Since many of the basic structures will already be familiar, these will not be explained in any detail. The references may be consulted for more extended descriptions. In using stochastic techniques I am not, in general, attempting to model traditional compositional or acoustic structures. Rather, I am aiming firstly to employ the computer to ease the compositional load by taking decisions within defined boundaries when appropriate, secondly to define and examine the musical properties of interesting abstract structures, and thirdly to surprise myself with unanticipated sounds. If I am satisfied with an interesting abstract or mathematical structure, I can be fairly sure that the resulting sound will be interesting, although, its precise nature may be unpredictable. I have used abstract stochastic structures not only to organise groups of "notes", but also to order sounds of very short duration to create interesting vibrant textures.

Constrained Choice

The simplest type of stochastic system is where a simple random choice made between specified upper and lower limits delineating, for example, pitch range, note duration, note density, intensity and timbral indeces. In Stockholm, Michael Clinton has developed the IMPAC program for stochastic generation which by controlling a bank of digital oscillators will deliver a constant stream of "notes" Their parameter boundaries may be varied either in real-time using input devices such as joy stick, bit-pad and Keyboard, or by a set of pre-defined functions which are similar to, but more general than, Barry Truax's POD programs. The IMPAC system was used to realise my composition ICE BREAKER.

Probability Distribution

Next comes the possibility of weighted choice defined by a probability distribution. In my orchestral piece FIRELAKE (one movement from a larger work) a particular type of random decaying function was used. If the function RND (N) delivers an integer in the range 1 to N, then the function RND (RND(N)) will deliver an integer in the same range, but with probabiliti weighted towards the integer 1. Thus N will occur rarely, and 1 very often. This function was used to make compositional decisions in FIRELAKE by being applied over a number of different event spaces. Figure 1 shows the overall conceptual plan of the movement from which base the function was used to determine all pitches, duration of woodwind sections, brass patterns etc.

Markov Chain

The next stage is to use Markov chains where the probability of any event's occurance depends upon the event preceeding it. This model was used in the compositon for voices TEXT YEARS where pre-composed samples of text were analysed by computer, using a program written in ALGOL, to draw up a matrix of transition probabilities. This matrix was then used to generate substantial chunks of text each having a consistent probabilit structure and which will consequently generate a consistent sound texture when read freely in chorus (see Figure 2).
I have used Markov chains extensively in some earlier instrume al works, notably SYMPOSIUM and MAYTRICKS (see Jones, 1978), and have considered the musical implications of various types of event classification and analysis of Markov chains (see Jones, 1980a and 1981).

Random Walk

One special case of a Markov chain is the random walk, where a process can move from any event only to an adjacent event, a model of Brownian motion. The random walk principle was used in the computer tape composition LAITRAPARTIAL.

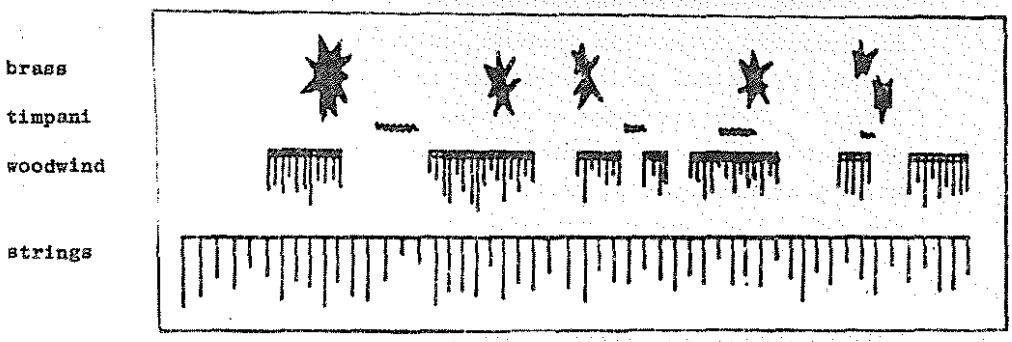


Figure 1

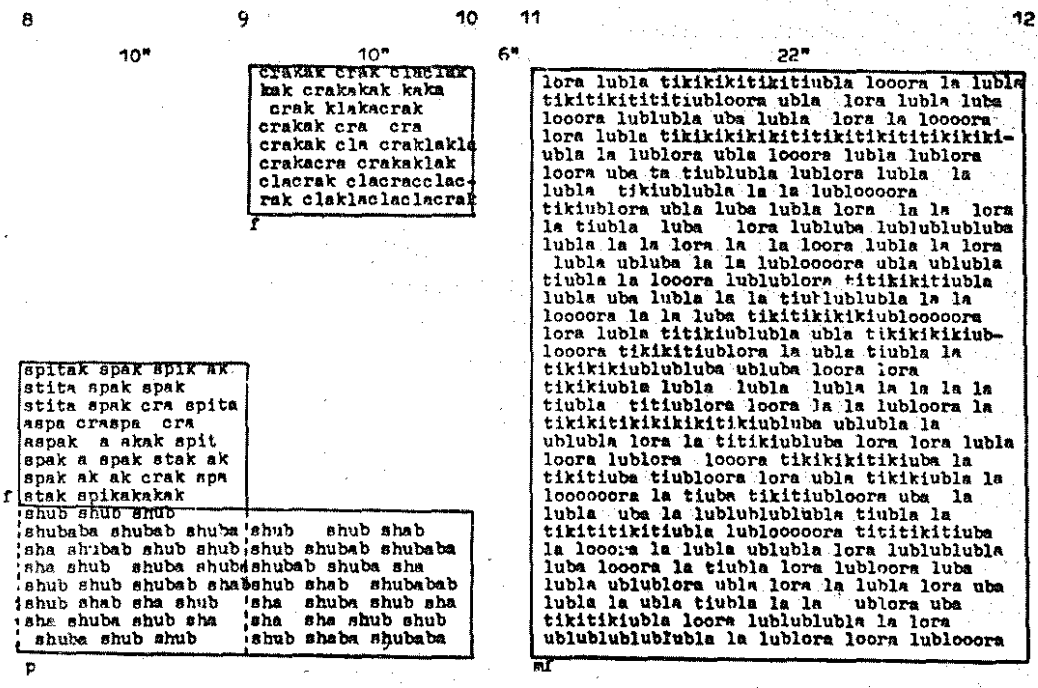


Figure 2

Up to twenty sound particles were varied every twentieth of a second by moving through adjacent harmonics and intensity values on the random walk principle (see figure 3 (a)). This produces a curious shimmering effect. Sometimes a bias was introduced towards upward movement into higher harmonics (figure 3 (b)) or downward movement so that the particles end up together on a pulsating fundamental (figure 3 (c)).

Three-Dimensional Markov Chains

Another possibility is to use a three-dimensional Markov Chain model where an event s occurance depends on the two preceeding events. This was used to generate rhythms in the compositior SKIRTRIKS (title from Scherzo/Tricks).

When used with the two simple duration values, long and short, it means that rhythmic cells of three or four notes are continually repeated, with occasional stutters which introduces a more natural, human-like quality to the result.

String grammar

Markov chains are in fact a special case of the more general grammars used in formal linguistics. Formal grammars as a means of expressing compositonal ideas in computer music have been suggested by Roads (1979) and Holtzman (1979). I have done some experiments using simple grammars, not simply to order a set of terminal events, but to allow a syntax generated by recursive production rules to define events by dividing up a continuous event space. This is shown in its simplest form by applying the following production rules to generate rhythms.

$$A \rightarrow AA \quad (1)$$
$$A \rightarrow a \quad (2)$$

When rule (1) is applied, a line segment representing time is split in half leaving two smaller segments to be further operated upon.

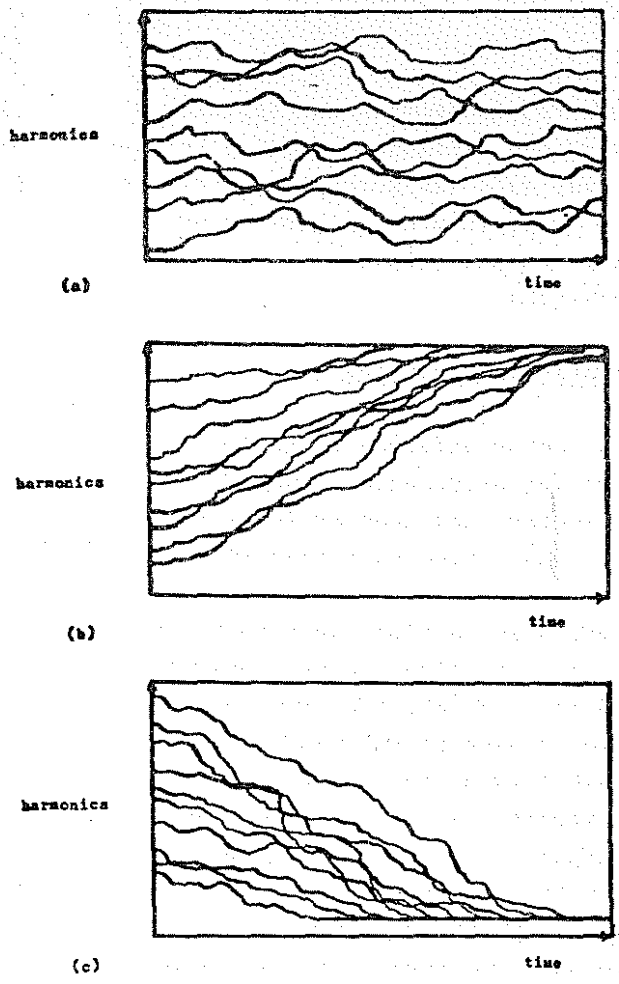


Figure 3

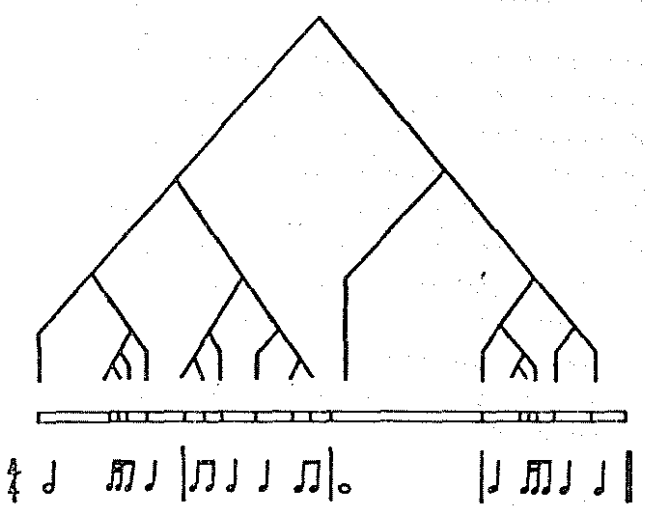


Figure 4

When rule (2) is applied the splitting operation for that segment ceases. Figure 4 shows a line segment divided up in this way with its generating syntax tree and resulting rhythmic interpretation. Increasing the probability of rule (1) being applied will generate faster rhythms with shorter note durations.

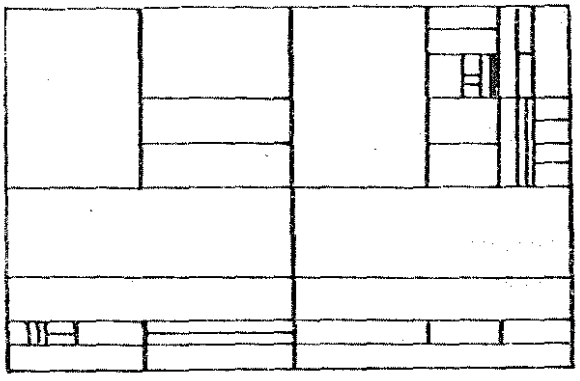
Two-dimensional splitting

The idea can be extended to divided up the plane by having two splitting productions; one for a vertical division of a plane segment and another for a horizontal division. In this way, just three rules can produce a pattern such as that in figure 5 (a). By applying divisions along the horizontal azis to time, as before, and divisons along the vertical azis to pitch, this transcribes into the form of figure 5 (b). Notice that this has an interesting harmonic structure, generated automatically by the grammar. Increasing the probability of horizontal divisions will introduce a bias towards parallel activity in long held chords; increasing the probability of vertical divisions will introduce a bias towards sequential activity. This grammar has also been written as a recursive ALGOL procedure to automatically generate data for sound synthesis programs.

Space Grammar

I have used the term "space grammar" to describe such a generative structure. Its application does not stop at two dimensions but may be used to divide up blocks in three-dimensions or indeed in any number of dimensions that may be required to define musical control parameters. (See Jones, 1980). To make the important generative implications clear, the example grammar has deliberately been kept simple, but sophisticated extensions are possible. A wider repertoire of variables and terminals may be used to define more precise structural contrasts, and divisions need not be made "Nel Mezzo" for example.

(a)



(b)

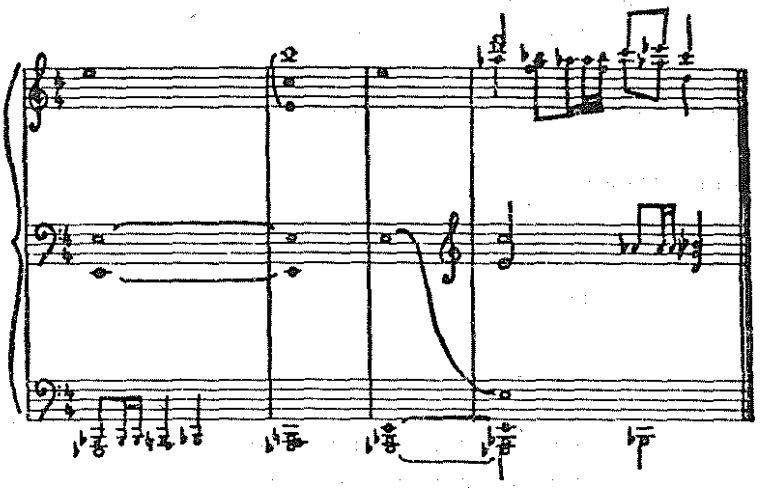


Figure 5

REFERENCES

GONZALES, R.C. & THOMASON M.G. (1978). "Syntactic Pattern Recognition, An Introduction" Addison-Wesley.

HINTON, Michael (1979) "Interactive Music Performance And Composition" EMS, Stockholm.

HOLTZMAN, Steven (1979). "Generative Grammar Definitional Language", Department of Computer Science Edinburgh University.

JONES Kevin (1978) "A Practical Study of the Application of Computer Techniques in Processes of Musical Composition " M. Phil, Thesis, University of Aston in Birmingham.

JONES Kevin (1980a) "Computer Assisted Applica tion of Stochastic Structuring Techniques in Musical Composition and Control of Digital Sound Synthesis Systems "Ph.D. Thesis, The City University, London.

JONES Kevin (1980b) "A Space Grammar for the Stochastic Generation of Multi-Dimensional Structures". Proceedings of the 1980 International Computer Music Conference, Queens College of the City University, New York.

JONES Kevin (1981) "Compositional Applications of Stochasti Processes" Computer Music Journal 5 (2), (appearing soon).

OADS, Curtis (1979). "Grammars as Representations for Music" Computer Music Journal 3 (1).

RUAX, Barry. (1973) "The Computer Composition-Sound Synthesis Programs POD4, POD5 and POD6." Institute of Sonology, Utrecht State University.

COMPUTER MUSIC IN GREAT BRITAIN

KEVIN JONES
Music Department
The City University
London EC1.

The outlook for computer music in Britain has not been as good as in the United States and many parts of Europe. It has been a struggle, with very limited hardware, for a few enthusiasts which has lacked any substantial financial support. Even now, most serious computer music compositio by British composers has been done in other countries, and many of those working in the field have left Britain for more favourable environments. With one or two exceptions most interest in computer music has been developed in university music departments.

EMS, LONDON

Britain got off to a leading start with the pioneering wor of Peter Zinovieff at his EMS Putney studios in London. During the sixties he set up what was probably the worlds first sophisticated computer-controlled electronic music studio. As early as 1968, he presented a concert of compu music, composed and performed in real time, at the Queen Elizabeth Hall in London. Alan Sutcliffe of the Computer Arts Society and the composer Harrison Birtwhistle have worked on projects in the studio in Putney, which is described in Douglas (1973).

In the mid-seventies the studio moved to Oxford where the possibility of an affiliation with Oxford University's mu department was mooted. This did not materialise, the stu encountered financial difficulties and had to be broken u Much of the old studio currently lies buried in a corner the National Theatre in London; unused and probably by nc unusable.

"Computer-synthi" manufactured by EMS was installed at
asgow University, but with no software available the system
: only used for analogue generation.

anwhile, Oxford University has installed an Australian
'airlight" digital synthesizer.

UTHAMPTON

e first attempt to instal a general purpose sound synthesis
ogram on a large computer was pioneered by Stanley Haynes
Southampton in 1972 when MUSIC 5 was set up on the
iversity's ICL 1907. In 1973 Eric Graebner added MUSIC
F. In 1977 Stanley Haynes moved to the City University
d Southampton acquired a new computer, an ICL 2970.
anging the music programs for the new machine proved
possible. However, MUSIC 4BF has been set up on the CDC 7600
the University of London Computing Centre (ULCC) and can
used via a land line. For conversion tapes have to be
nt back to Southampton by train ! A poor D/A conversion
stem operating at 10K means that the system has been used
r little more than occasional experiment.

nathan Harvey was associated with the project at Southampton
its early stages. He has since been appointed professor
music at Sussex University, Brighton. He has used computer
sic systems in the United States and recently completed a
w work, "Mortuos Plango, Vivos Voco", at IRCAM in Paris,
ich makes use of computer modification of digitised pre-
corded sounds.

TY

anley Haynes came to the City University, London in 1977 and
t up MUSIC 5 on the ICL 1905E. Between 1978 and 1980 it
s operational, being used for teaching and research.

Again, certain unidentified system bugs made the system unsuitable for serious composition even though it was operating with a sampling rate of 20K. A special D/A conversion and buffer unit were built by the computer engineers for off-line conversion. In 1979, Stanley Haynes left to work at IRCAM whilst in 1980 the computer centre acquired a new Honeywell computer. This made the music system inoperable; not least of the problems being the incompatibility of the 9 - track tape standard on the new machine, with our 7 - track conversion system. Although possible ways to circumvent the difficulties have been devised, it has been decided to concentrate on other approaches. A copy of City's MUSIC 5 was set up on an identical machine at Birmingha University's Physics Dept., who still come down from time to time to use our tape conversion facility.

At City our resources have been channelled into micro-computer applications. An APPLE II has been in use for 2 years. Currently it supports three MC16 ALF synthesizer boards giving a total of nine channels of square wave synthesis. These are easily programmed in BASIC which allows possibilities for sophisticated control, for example in producing rich amplitude modulation effects, and in writing automatic composing prograr We have also recently acquired the Mountain Hardware music system which has 16 digital oscillators with programmable wav forms. This should prove useful in teaching, and for simple acoustic experiments, but the poor sound quality resulting fr 8 - bit resolution precludes an serious compositional use.

Don Finlay,

a member of the Electrical Engineering department, has been developing a music system based on the NASCOM micro-processor which can be used to control an analogue synthesizer, or for direct synthesis (8 - bit). An organ Keyboard is currently being interfaced to this system. A research student Richard Attree , is investigating computer models of the compositiona process.

My own research is described in the accompanying paper. This has concentrated on composing algorithms and structured pattern generation rather than on digital sound synthesis along traditional lines.

The department has just taken delivery of a Fairlight digital synthsizer, but this will probably be used more as a studio instrument than for serious computer music research.

EDINBURGH

From 1978 - 1980 Steven Holtzman was engaged in setting up a sophisticated computer music system at Edinburgh University. He initially worked in the department of Artificial Intelligence and then in the Computer Science department, which possesses some of the best resources in Europe. He set up a non-standard, noise-based synthesis system on a PDP15 (a project related to similar experiments in Utrecht) and has developed a general prupose composing language, GGDL (Generative Grammar Definitional Language). David Hamilton has realised music for a radio play using the system. Regrettably, when the research period came to an end, no further support was forthcoming. The project is no longer operational, and Holtzman is currently working for the Apple computer company in the United States. GGDL has, however, been set up at IRCAM.

CARDIFF

An EMS synthi 100 with the possibility of computer control from a PDP8 has been available in the Physics Department of Cardiff University since 1972. Experimental work and composition has been carried out by Marcus West and Michael Greenhaugh.

DURHAM

A project has been underway for some time to introduce computer sound synthsis into the electronic music studio at Durham University by Dr. Peter Manning. Barry Vercoe's MUSIC XI is now set up and should be available for composition and research very soon.

KEELE

A two year project to introduce computer sound synthesis in the music department at Keele University was abandoned through technical and organisational difficultes. An Apple II with ALF synthesizers has been acquired and is being used for teaching. Tim Souster who has composed computer music in the United States and was initially involved in the project, now has his own analogue studio in London.

SURREY

Surrey University music department has had a long-standing commitment to music and technology. An Apple II Computer with Mountain Hardware boards has been installed by Robin Maconie.

OTHER PROJECTS

An old computer at the University of East Anglia music studio remains unused. Digital sound synthesis projects are underway in a number of electrical engineering departments. Those with musical interests include Huddersfield Polytechnic and Central London Polytechnic. Lawrence Casserley of the Royal College of Music's electronic music studio is developing a project based on a Research Machine's 380 Z microcomputer.

REFERENCES

DOUGLAS, Alan (1973). "Electronic Music Production. "Pitman Publishing, London.

HAYNES, Stanley (1980). "The Musician-Machine Interface in Digital Sound Synthesis". Computer Music Journal 4 (4).

HOLTZMAN, Steven (1979). "An Automated Digital Sound Synthesis Instrument". Computer Music Journal 3 (2).

(1979) "GGDL, Generative Grammar Definitional Language" Dept. of Computer Sciece, Univ. of Edinburgh.

(1980) "Generative Grammars and the Computer-Aided Composition of Music", Doctoral Thesis, University of Edinburgh, Dept. of Computer Science.

(1981) "Using Generative Grammars for Music Composition" Computer Music Journal 5 (1)

JONES, Kevin, (1978) "A practical study of the application of Computer techniques in processes of musical composition." M. Phil Thesis, University of Aston, Birmingham.

(1980) "Computer assisted application of Stochastic structuring techniques in musical composition and control of digital sound synthesis systems." Ph.D. Thesis,City University, London.

MANNING, Peter (1977) "Electronic Music and the computer" Ph.D. Thesis, University of Durham.

see also "Computer Music In Britain", Proceedings of the national conference at Edinburgh University, Dept. of Computer Science, 1981, Steven Holtzman(Editor),

"Computer Music 1976/77 : a directory of current work". William Buxton (Editor), Canadian Commission for UNESCO.