梯子型LCRフィルタ設計支援プログラムの作成

バタワース，トムソン，チェビシェフ，連立チェビシェフをサポート

西村 芳一

要求仕様のきびしいLCRフィルタを設計するとき，L,C,Rの各定数を求めるためには，非常にめんどうな計算を数多くしなければならない．そこで，LCRフィルタ設計支援ソフトウェアを紹介する．バタワース，トムソン，チェビシェフ，連立チェビシェフの各タイプのフィルタのL,C,R定数を算出する．また，各素子の定数を定めたフィルタの周波数特性を計算することもできる．ここでは，フィルタ設計時の注意点を述べながら，このソフトの使用方法を示す．プログラムはリギーForth(FifthZ80)で書かれており，CP/M-80上で動作する．ただし，筆者よりソースが公開されるので，Fifth86でコンパイルすることによりMS-DOS上でも動作可能であろう．（編集部）

はじめに

*LC*梯子(はしご)型フィルタ設計を行うためのプログラムを作成したので，ここに紹介する．バタワース，トムソン(ベッセル)，チェビシェフ，連立チェビシェフ(Cauer)の各タイプのフィルタの素子値を算出し，そのときの周波数特性を計算することができる．またその設計値を実際使用する素子値に変換した場合の周波数特性も調べることができる．

プログラムはForthで書かれており，コンパイルされて，CP/M-80のCOMファイルとして実行される(MS-DOS上で駆動させることも可能である．稿末参照)．

1 開発の経緯

▶本を読んでも，どうやって計算するのやら

電子回路の設計者にとって，フィルタは身近な存在である．しかし，実際に設計するとなると，あの難解なフィルタの教科書を開かなければならない．一般的には，フィルタの本にたいてい載っている基準フィルタの表を使い素子変換を行って安直に算出することになる．

それでうまくいけばよいが，要求されている仕様がきびしいときなど，教科書にある表に出ていない所が必要になったりする．筆者の場合も，バタワースとトムソンの中間位の特性が欲しくて，こういう特性をもつTBT(Transitional Butterworth Thomson)フィルタを理論式から計算することにした．

ところが，これを電卓片手に計算することは不可能に近い．そこで，手元にあるパソコンを用いて，フィルタを設計する汎用のソフトが市販されていないかどうか調べてみると，意外にこれがみつからない．アクティブ・フィルタのものならあるが，*LC*受動(パッシブ)素子向けはどうもない．結局，自分でソフトを作ることになった．少し古い機種だがFM-7があったので，まずF-BASICで組んでみた．

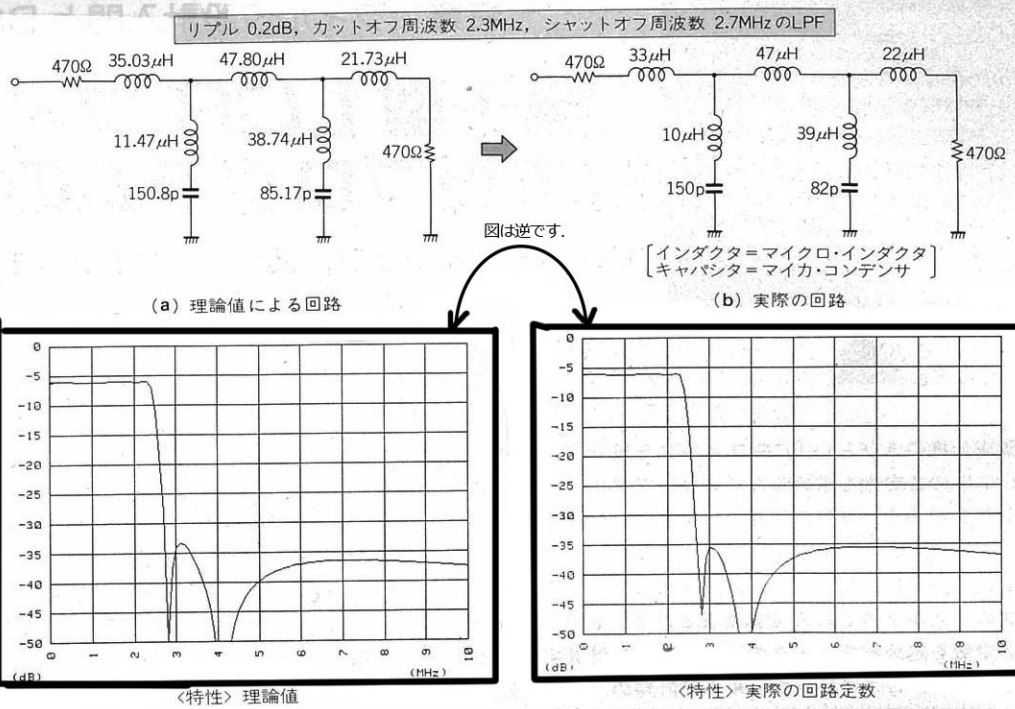
本棚に埃をかぶっていたフィルタの本を，さっそく引き出してみた．一回読んでもよくわからない．これは腰を据えて勉強しないといけないと悟り，大きな書店へ行きフィルタ関係の本を捜して，読みあさることとなった．その中で比較的，具体的な例題を載せていた本[1]を中心として，その他3冊[2][3][4]を参考にしながらようやくまとめあげた．それぞれの本は特徴があり，筆者にとってこれ一冊ですむというようなトラの巻はなかった．

▶実際の回路定数は理論値とは違う

こうやって，いちおうフィルタの設計がある程度できるようになると，それがどんな特性をしているのかが知りたくなる．これもまた，手に電卓というわけにはいかない．さらに計算した素子値は図1のように85 pFという具合で，実際に回路を実現する場合には，たぶん82 pFを使う(column A参照)．その場合の特性の変化も気になる(図1)．

これに関しては，市販のソフト(たとえばHP社の

〔図 1〕
理論値と実際の回路

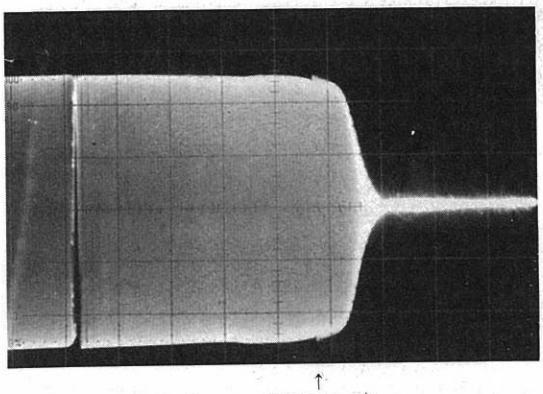


(a) 理論値による回路

(b) 実際の回路

〈特性〉理論値

〈特性〉実際の回路定数



(c) スイープ信号を入力したときの出力(横軸は周波数)

↑
2 MHz マーカ

column A

実際の LCR 数値系列

LCR の部品を購入する場合，一般には一定の数値系列のなかからその定数を選ばなければならない．数値系列は，E 6 などのように，E にその数列の公比を示す数値を付けて表す．**表 A .1** に示すように，E 3，E 6，E 12，E 24 の数値系列が，多く使われる．その数値 E*n* の意味は，その数値系列の公比が，$\sqrt[n]{10}$ であることからきている．

電解コンデンサは E 3，コイル(インダクタ)やマイラ・コンデンサなどは E 6，また抵抗やマイカ・コンデンサなどは E 12 系列が使われていることが多い．

詳しくは，日本規格協会，『JIS ハンドブック電子』などを参照されたい．

〔表 A .1〕
LCR 定数の数値系列に関する表

E数列	定数																							
E 3系	1								2.2								4.7							
E 6系	1				1.5				2.2				3.3				4.7				6.8			
E12系	1		1.2		1.5		1.8		2.2		2.7		3.3		3.9		4.7		5.6		6.8		8.2	
E24系	1	1.1	1.2	1.3	1.5	1.6	1.8	2	2.2	2.4	2.7	3	3.3	3.6	3.9	4.3	4.7	5.1	5.6	6.2	6.8	7.5	8.2	9.1

ACアナリシスなど)もあるが，自分のフィルタ設計プログラムとのインターフェースが悪い．これを結局自分で作ることとなり，最終結果をグラフィックスで表示させた．

▶ F-BASIC から Forth へ

最初，筆者の FM-7 にはフロッピ・ディスクがなかったので，テープにプログラムをセーブしておいた．その後，ディスク・ドライブを買いさっそくディスクにセーブしようとすると，Basic のユーザーズ領域が狭くなった関係上，ディスク Basic では走らなくなってしまった．コメントを削り，詰めに詰めた結果なんとか動くようになったが，新たにバグが発生し，もうどうしようもなくなってきた．

また，Basic のスピードの遅さにはとても我慢ができなくなり，F-BASIC のコンパイラを捜したが当時は実数形対応はなかった．たまたま本誌上で，リギー Forth の連載記事を見た．これは面白そうだということで，勉強も兼ねて Forth で移植してみることにした．Forth を使ったおかげで，定義した複素数演算ワード(**表 1**)を使い式を簡明に表すことができた．

F-BASIC バージョンよりもさらに機能アップし(残念ながら，標準 CP/M にインターフェースする関係上，グラフィックスの機能は省いた)完成することができた．

2 フィルタの種類とその特徴

▶振幅特性か波形特性か

フィルタというと周波数対振幅特性だけに目がいきがちになるが，それと同等に波形特性も重要である．二つの特性は相反する性格をもっており，両方とも完全に満足いく特性を得るには，急峻な遮断特性をもつフィルタと位相等価回路の組合せになってしまう．ここでは，位相等価回路を使わないである程度それらの妥協点を見つけるため，バタワース特性とトムソン特性の中間で設計できるようにした(詳細は後述の**図12** 中の m を参照)．

図 2 にバタワースおよびトムソン・フィルタの特性と混合係数 M で表されるそれらの中間特性を示す．

▶分波フィルタと最少インダクタ・フィルタ

図 3 のような，一つの信号源から複数の周波数成分に分けるフィルタを，分波フィルタという．一般的に複数の低域通過，高域通過フィルタからなっており，たとえば並列分波フィルタで要求されるフィルタの特性として，入力インピーダンスが通過帯域以外では，ハイ・インピーダンスとならなければならない．**図 4**

〔表 1〕
複素数ライブラリ

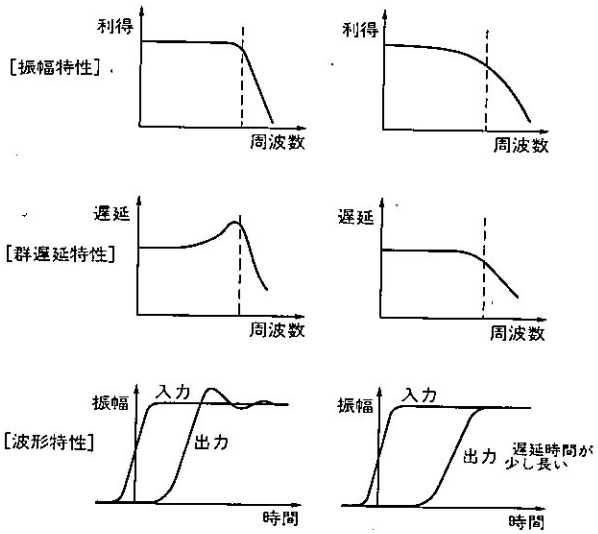
ワード名	機能	入出力パラメータ* [大文字＝複素数：16バイト] [小文字＝ワード：2バイト]
C−DUP	複素数の DUP	(C−−−C，C)
C−SWAP	複素数の SWAP	(C，D−−−D，C)
C−*	複素数の積	(C，D−−−C*D)
C−CONJ	共役複素数 ($a+jb \longrightarrow a-jb$)	(C−−−$\overline{C}$)
C−/	複素数の商	(C，D−−−C/D)
C−+	複素数の和	(C，D−−−C+D)
C−−	複素数の差	(C，D−−−C−D)
C−ROT	複素数の ROT	(C，D，E−−−D，E，C)
C−−ROT	複素数の−ROT	(C，D，E−−−E，C，D)
C−OVER	複素数の OVER	(C，D−−−C，D，C)
CINV	複素数の逆数	(C−−−1/C)
C−STO−A	複素数を汎用レジスタAにストアする	(C−−−)
C−RCL−A	汎用レジスタAから複素数をリコールする	(−−−C)
C−ABS	複素数の絶対値 ($a+jb \longrightarrow \sqrt{a^2+b^2}$)	(C−−−|C|)
ARRAYC!	先頭アドレス add の複素数の配列 n 番目にストア	(C，n，add−−−)
ARRAYC@	先頭アドレス add の複素数の配列 n 番目からフェッチ	(n，add−−−C)
R==>P	直交座標系を極座標系の複素数に変換 ($a+jb \longrightarrow re^{j\theta}$)	(C−−−C')
P==>R	極座標系を直交座標系の複素数に変換 ($re^{j\theta} \longrightarrow a+jb$)	(C−−−C')

注　入出力パラメータは，スタックの状態を示し，(入力スタック状態−−−出力スタック状態)となっている．なお，で区切られた右側がスタックのトップである．

に，入力インピーダンスで分類したフィルタの表を示す．それぞれの阻止域信号での入力側インピーダンスがHighになるのがオープン・タイプで，Low(ショートされる)になるのがショート・タイプである．並列分波フィルタは，オープン・タイプでなければならない．

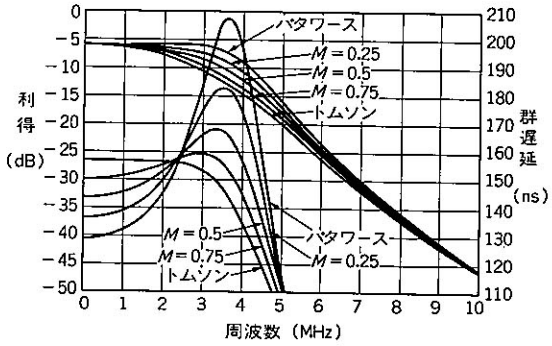
このように，同じ段数で同じ特性をもつフィルタでも2種類のタイプがある．一般的に，インピーダンス特性を気にしなくてもいい所は，なるべくインダクタ(コイル)の使用を避けたい所である．インダクタは，高いQのものが得にくく，キャパシタ(コンデンサ)ほどの品種の部品もない．また基板などに取り付ける場合，実装上の問題も多い．

〔図2〕バタワース，トムソンの特性比較



(a) バタワース

(b) トムソン



(c) パラメータMによる振幅特性，遅延特性の変化
(TBTフィルタ)
[カットオフ周波数=4MHz, n(段数)=5]

column B

振幅特性と波形特性

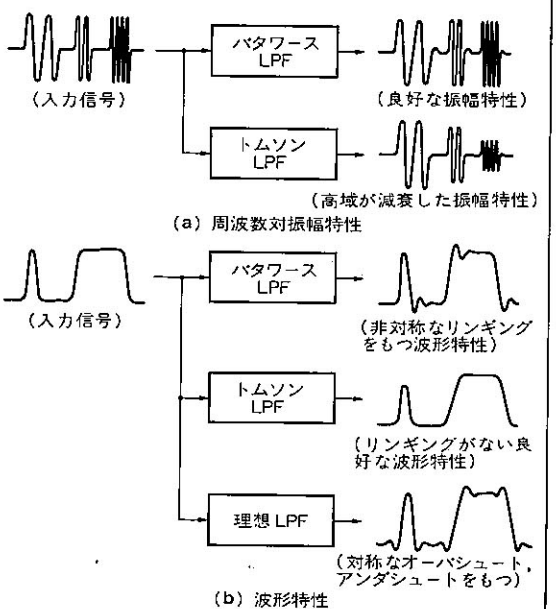
ある回路の特性を評価する指標として，周波数対振幅特性や，SIN²インパルス・ステップなどの入力信号に対する出力波形特性などがよく用いられる．図B.1にそれぞれの，良好な特性を示す代表として，バタワース，トムソンの応答を示す．

ディジタル・データ伝送などの，波形の振幅よりも時間軸でのエラーが重要な場合は，トムソン特性が向いている．また，音声処理では周波数対振幅特性の劣化のほうが目立つので，バタワースが好まれるであろう．

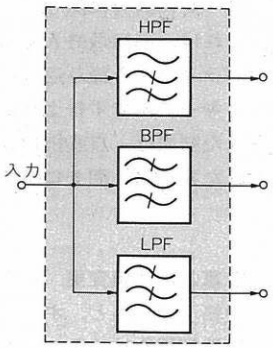
一般的に，振幅特性が急峻な遮断特性をもつほど波形特性は劣化する．すなわちチェビシェフ特性のフィルタは，あまりよくない波形特性を示す．

この相反する特性を同時に満足するためには，フィルタのあとに全域通過の位相補償フィルタを接続する．または最近のCDプレーヤに見られるように，直線位相のFIRディジタル・フィルタを用いなければならない．このような理想に近いフィルタの波形特性は，図B.1のように対称型のプリ・アンダシュート，オーバシュートを示す．

〔図B.1〕振幅特性と波形特性



〔図3〕分波フィルタ



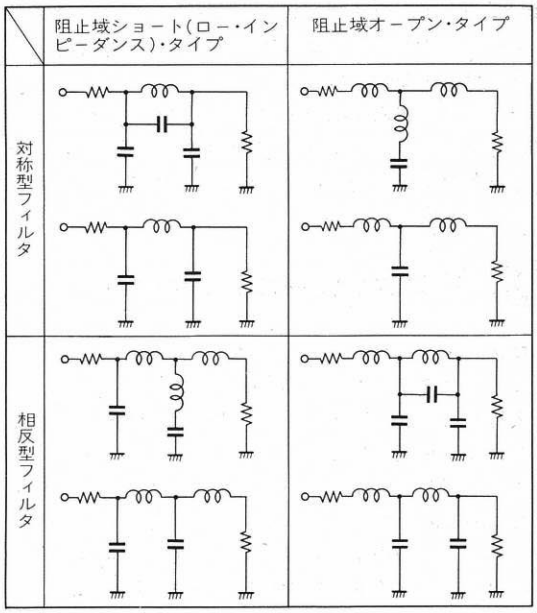
そこでこのプログラムでは，入力インピーダンス特性を2タイプの中から選べるようにした．このことは，高域通過フィルタについても，同様である(ただし，連立チェビシェフ・フィルタの高域通過フィルタでは，Lの数が多くなるショート・タイプは選べない)．

▶フィルタの各種パラメータ

このフィルタのプログラムを実行させるのに必要な各種入力パラメータを**図5**に示す．

バタワース/トムソン混合フィルタとしたため，トムソンおよびその中間型フィルタは，カットオフ(遮断)周波数の定義が不明確であるが，**図2**(c)を参考に設計してほしい(column C 参照)．

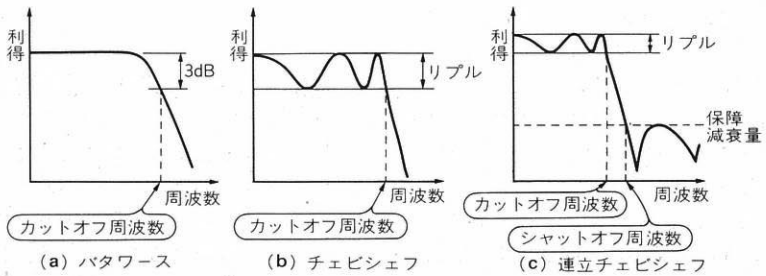
〔図4〕入力インピーダンスによる分類(LPF)



▶対称型と相反型フィルタ

フィルタは，段数nが偶数と奇数とでは，大きく特性が異なる．奇数の場合，入出力インピーダンス特性が等しく梯子型に展開するうえで問題は少ない．一方偶数次で，とくにチェビシェフ特性のフィルタの場合

〔図5〕
各種フィルタのパラメータ



column C　カットオフ(遮断)周波数

フィルタは一般的に，信号が通過する**通過域**(パス・バンド)と信号を阻止する**阻止域**(ストップ・バンド)からなる．そこで，通過域から阻止域に遷移する領域で，どこまでを通過域と見るかという問題がある．

習慣上，「電力半値(−3 dB)になる周波数」を**カットオフ周波数**と呼んでいる．通過域の端は，このカットオフ周波数までとして扱われる．

ところがフィルタを設計するうえで，かならずしもこの定義によるカットオフ周波数が使いやすいとはいえない．そこで，このプログラムではフィルタの種類によってその定義を変えている．

バタワース・フィルタのカットオフ周波数はそのまま設計のパラメータとして使えるので，**図5**のように電力半値周波数として定義している．

一方チェビシェフ・フィルタは通過域のリプルの範囲を越えて，阻止域に移るところの周波数をカットオフ周波数と定義した．これをωとし，本来の意味のカットオフ周波数をΩとすると，それらにはつぎの関係がある．

$$\Omega = \omega \cdot \cosh(1/n) \cdot \cosh^{-1}(1/e)$$

ただし，eはリプル，nは段数

連立チェビシェフ・フィルタの場合，通過域のカットオフ周波数の定義は前のチェビシェフ・フィルタと同じである．さらに阻止域でもリプルをもつので，**図5**のように，阻止域のリプルの範囲を越えるところを**シャットオフ周波数**と定義した．

〔図11〕フィルタの定数変更例

```
Ground branch modification
C( 7 ) =  243.618 pF ===> 220
C ノ ヘイレツ テイコウ  open ( Kohm ) ===>
Transfer branch modification
L( 6 ) =   38.345 マイクロ H ===> 33
L ノ チョクレツ テイコウ  short( Ohm ) ===> 0.5
Ground branch modification
C( 5 ) =  127.735 pF ===> 120
C ノ ヘイレツ テイコウ  open ( Kohm ) ===>
Transfer branch modification
L( 4 ) =   27.900 マイクロ H ===> 27
L ノ チョクレツ テイコウ  short( Ohm ) ===>
Ground branch modification
C( 3 ) =   81.395 pF ===> 82
C ノ ヘイレツ テイコウ  open ( Kohm ) ===>
Transfer branch modification
L( 2 ) =   13.221 マイクロ H ===> 10
L ノ チョクレツ テイコウ  short( Ohm ) ===>
Ground branch modification
C( 1 ) =   17.282 pF ===> 18
C ノ ヘイレツ テイコウ  open ( Kohm ) ===>
```

つぎの関係式が成り立つ．

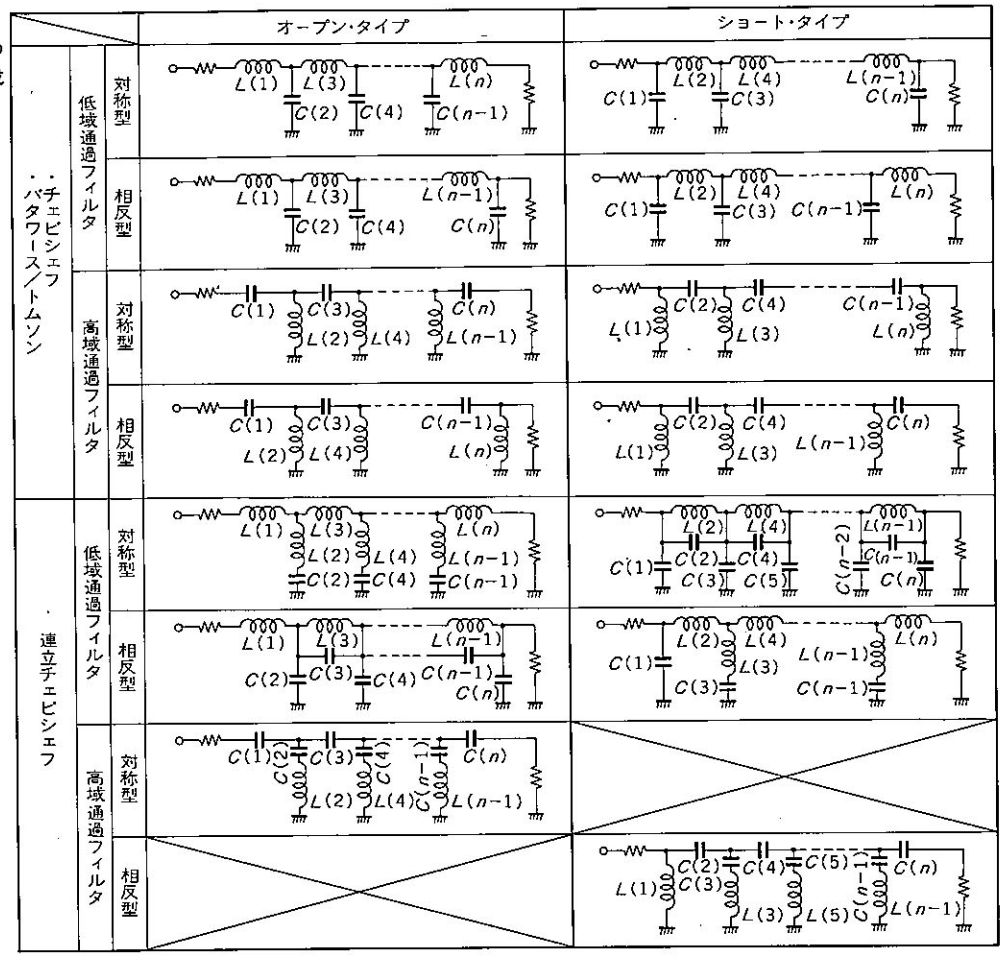
$$g(P)\cdot g(-P)=f(P)\cdot f(-P)+h(P)\cdot h(-P) \tag{1}$$

上記関係式より伝達関数，特性関数のいずれか一方が決まっていれば，因数分解により他方を求めることができる．(1)式は，両辺とも P に関して偶数次の項のみからなるため $S \triangleq P^2$ と置き換えることができる．したがって(1)式の両辺＝0の高次方程式を S に関して解くことができる．

実係数の方程式であるので，Sの根は実軸に関して対称な共役複素数となる．また $S=P^2$ であるから，S の根から P の根を求めてそれを複素平面上に示すと，虚軸に関しても対称の配置になる．そこで実数部が正か負かにより根を2グループに分けて，その根から再び二つの高次方程式を組み直す．これが，ここでいう因数分解を行ったことになる．

バタワース/トムソン，チェビシェフ，連立チェビシェフ・フィルタのそれぞれにつき，設計の手順と計算

〔図9〕
表示された素子のインデクスと構成



式を図12～14に示す．

▶リアクタンス関数から梯子型への展開

伝達関数と特性関数からリアクタンス関数が求まると，LC 梯子型に展開する．(∞以外の)減衰極をもたないバタワース/トムソン，チェビシェフ・フィルタの場合は，普通に連分数展開すれば梯子型に展開できる．

減衰極をもつ連立チェビシェフの場合は，多少複雑である．入力インピーダンス特性を $Z(P)$，その減衰極を Ω_j とし，対称型フィルタの $Z(P)$ が，

$$Z(P)=L_1 P+Z_1(P)$$

で表せるとする．$Z_1(j\Omega_1)=0$ すなわち，$Z_1(P)$ は，入力に Ω_1 を共振点にもつ直列 LC 共振回路とすると，

$$L_1=\left.\frac{Z(P)}{P}\right|_{P^2=-\Omega_1^2}$$

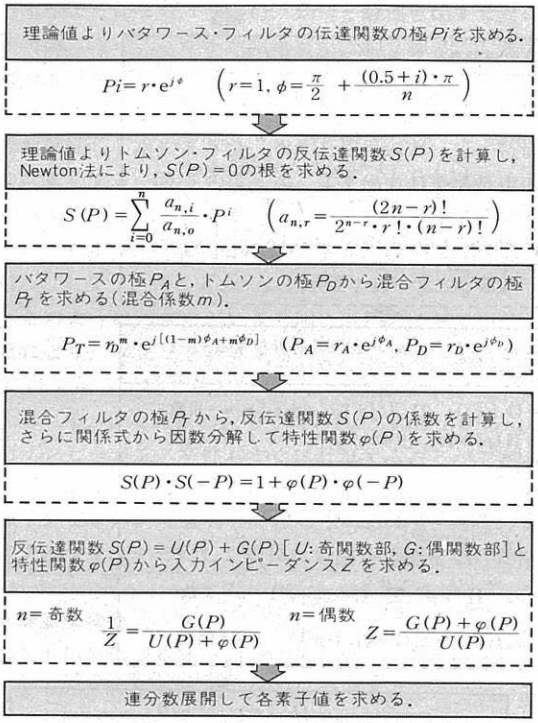
となり，L_1 は簡単に上式で求められる．また，$Z_1(P)$ は Ω_1 の直列共振回路 (L_2, C_2) をもつと仮定したので，

$$\frac{P}{L_2}=\frac{P^2+\Omega_1^2}{Z_1(P)}-\frac{P^2+\Omega_1^2}{Z_2(P)},\quad \Omega_1^2=\frac{1}{L_2\cdot C_2}$$

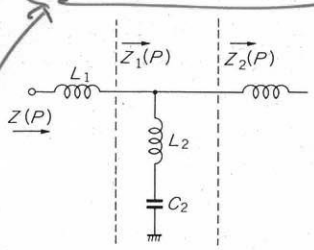
が成り立つ．$Z_1(j\Omega_1)=0$ であるから，$Z_1(P)\triangleq(P^2+\Omega_1^2)\cdot Z_3(P)$ と因数分解できる．一方 $Z_2(j\Omega_1)\neq 0$ であるから，つぎのように L_2 と C_2 が求まる．L_2 が求まったので，$Z_2(P)$ を

$$\frac{1}{L_2}=\left.\frac{1}{P\cdot Z_3(P)}\right|_{P^2=-\Omega_1^2},\quad C_2=\frac{1}{\Omega_1^2\cdot L_2}$$

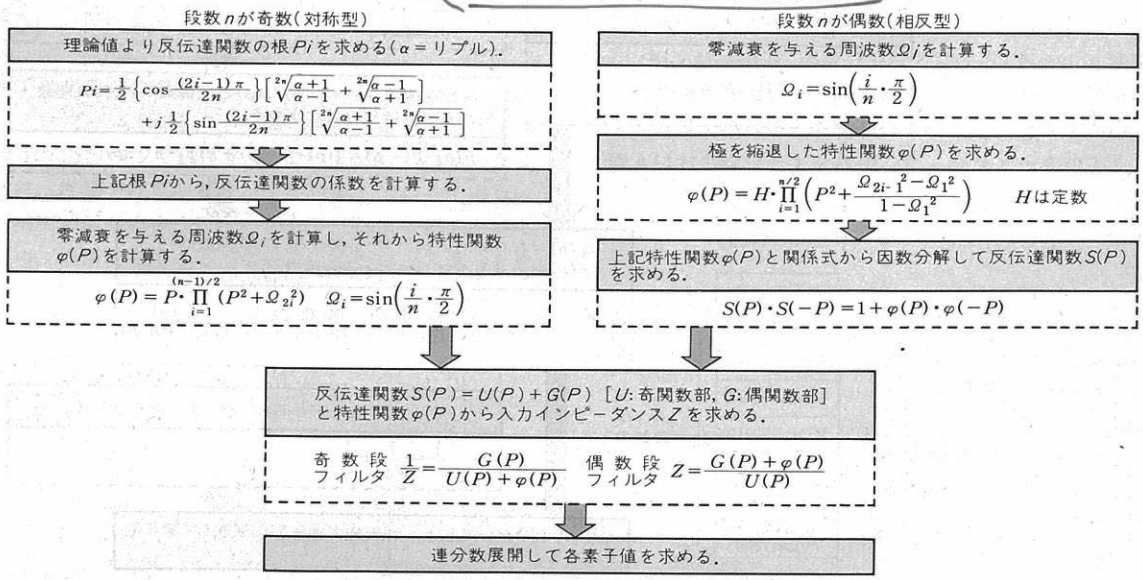
〔図12〕バタワース/トムソン混合フィルタの設計



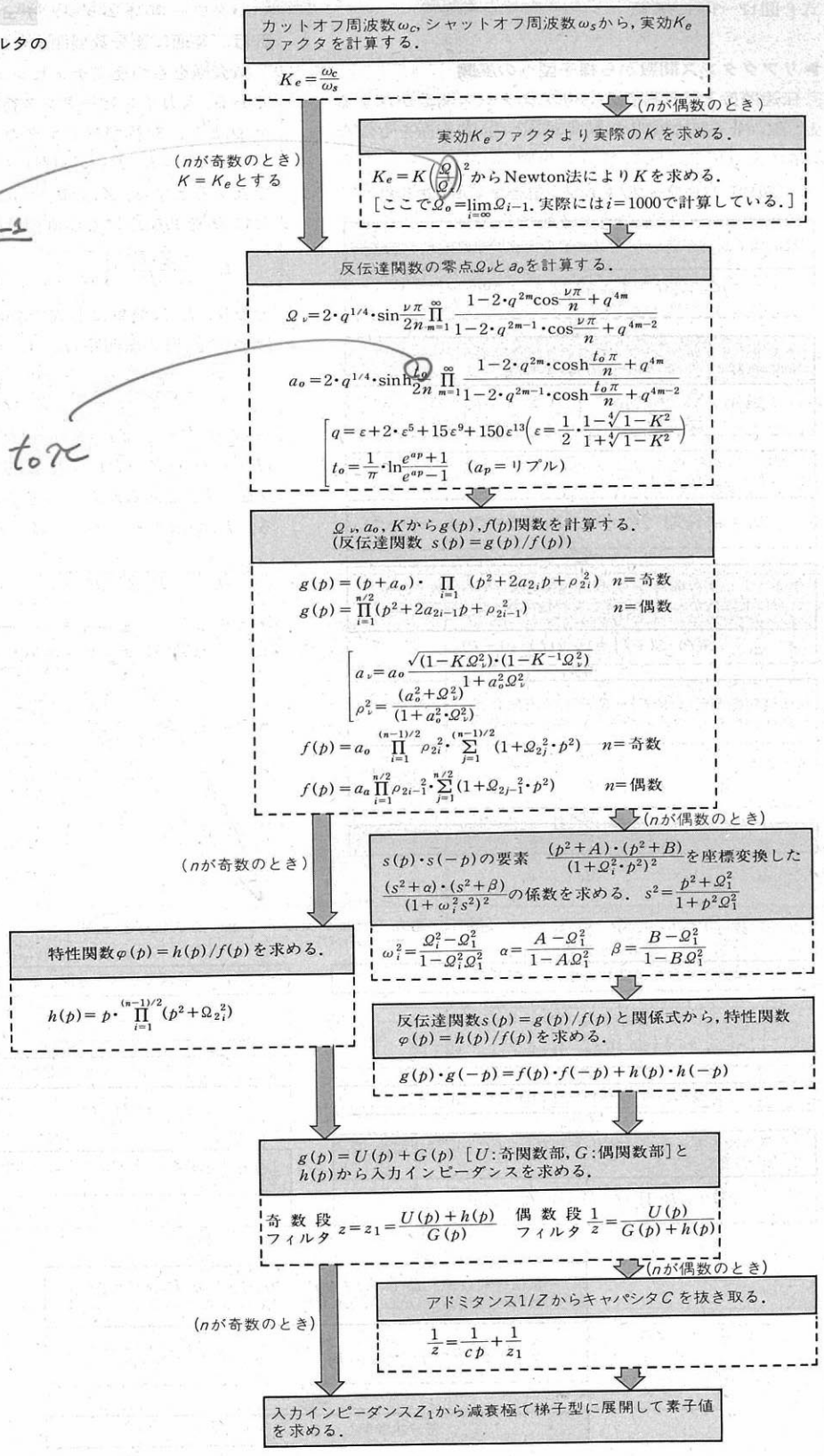
〔図13〕チェビシェフ・フィルタの設計



〔図15〕連立チェビシェフ・フィルタの梯子型展開



〔図14〕
連立チェビシェフ・フィルタの
設計



計算することができる．これを他の減衰極につき操り返すことにより，梯子型に展開できる(**図15** 参照)．

5 フィルタの周波数特性の計算法

▶出力を先に決める

フィルタの特性計算を代数式で解こうとするとたいへんなことになる．さらに，式に一般性を出すことは不可能に近い．しかし数値計算だと割り切ってしまえば先が見えてくる．

フィルタの各素子は，入力インピーダンス特性を連分数展開して求めてきた．それを逆手にとって，特性の計算をする．**図16** のように，まず出力の負荷抵抗の両端が1Vになる入力条件になっていたとする．するとただちに $I_5=1/Z_5$ と計算できる．続いて $I_4=I_5+(1/R)$ となり，I_4 と Z_4 から $V_4=Z_4\cdot I_4+1$ となる．

このようにつぎつぎに入力側にさかのぼっていくことにより，V_i と I_i を計算することができる．伝達関数 T は $T=V_o/V_i$ であるから，この場合 $T=1/V_i$ で求められる．また計算の途中で得られた V_1 と I_1 から，入力インピーダンス特性もすぐに計算することができる．

計算はすべて複素数で行っており，また各ノード間はインピーダンス Z_i として扱っているので，種々のタイプのフィルタに対して簡単に対応が可能となった．

▶群遅延特性

入力信号の周波数が ω のとき，出力信号の位相が θ とすると，群遅延は $d\theta/d\omega$(秒)である．ここでは，ω を1%変化させたときの特性を計算し，差分をとって群遅延としている．

6 プログラムの説明

▶プログラムの構成

プログラム(稿末に一部を示す)は，リギー Forth (FifthZ 80)の上で動作する．この Forth には幸いなことに，浮動小数点演算ワードが付属している．演算は倍精度が選べるようになっており，高次方程式を解くため，倍精度を用いる．これにともない，スタックを多量に消費するので256バイトの領域を確保した．

プログラムは，約30Kバイトの大きさ(オブジェクト・サイズ)である．**図17** に示すように，9000H からデータ領域が取られている．FLOAT(浮動小数点演算ライブラリ)の上に，6種類のライブラリが作られている(**図17** 参照)．

▶各ライブラリの説明

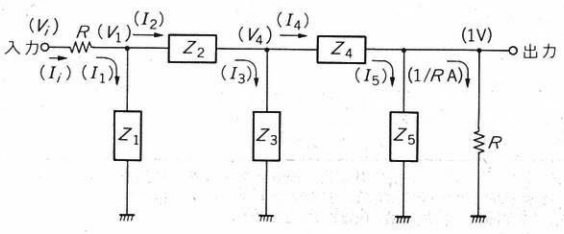
COMPLEX は，複素数演算ワードをまとめたユーティリティである．各ワードの説明は，**表1** に示した．

DAISU(代数)は，高次方程式の解，因数分解など代数演算をまとめたユーティリティである．ソース・リストにはすべて，入出力のスタック状態が書かれてある．(a, B---x)のような記述の場合，---の左側が入力スタック状態，右側が出力である．また","で区切られたパラメータは，右側がスタックのトップ方向で，小文字で書かれたものはワード長，大文字のパラメータは，浮動小数点のデータである．なお DAISU のメイン・ワードを**表2** に示す．

THOMSON は，バタワース/トムソン・フィルタを設計するライブラリである．**表3** にそのメイン・ワードを説明している．もっとも処理時間のかかる処理は高次方程式をニュートン法で解くことである．因数分解とトムソン・フィルタの極を求めるときこれを行うが，バタワース・フィルタのみのときは，それらは必要ないので処理時間は早くなる．トムソン・フィルタの極は正規化を行い，複素平面上で単位円付近に平均的にばらまく．そうすることにより，バタワース・フィルタとの混合フィルタが構成できる．

TFILTER は，連立チェビシェフ・フィルタを設計するライブラリである．相反型と対称型フィルタでは処理時間が異なる．対称型は，たんに計算すればよいが，相反型は縮退，因数分解などにより，Newton 法を多用するのでたいへん時間がかかる．したがってプログラムは，相反型と対称型とでは異なる処理が多い．**表4** におもなワードを説明した．パラメータが多くて

〔図16〕梯子型フィルタの設計



〔図17〕メモリ・マップとライブラリ

スタックが重くなる所は，内部に配列を取っている．しかし種々のワードで同じ配列を使っているときがあるのでプログラム変更の際は注意が必要である．

TBFILTER は，チェビシェフ・フィルタを設計するライブラリである．これも相反型のほうが処理時間がかかる．プログラムは，バタワース/トムソン・フィルタの設計と同様の処理が多いので，THOMSON ライブラリの多くのワードを使っている．それにより，短いステップ数ですんだ．**表 5** に，おもなワードをまとめた．

最後に FILTER は，前に説明した処理ワードとユーザ・インターフェースするユーティリティである．また，周波数分析や設計した素子値の変更を行うワードも含んでいる．メイン・ワードは filter-design で，COM ファイル実行時，まずこのワードにジャンプする．**表 6** におもなワードを掲げる．ワード名に prompt とあるワードは，すべてキーボードからパラメータを取り込む機能をもつ．また print とあるワードはすべて CRT 上に結果を表示する機能をもつ．

▶移植性

アセンブラで組んであるワードは，rpick と frpick の二つしかないので，他の CPU に移植するのは，簡単だと思う．機会があれば，8087 付きのマイコンで移植したいものだ．ただし，リギー Forth は，標準 Forth と異なる所も多いので，標準 Forth を用いて移植する

〔表 2〕
"DAISU"のワード

ワ ー ド 名	機 能 説 明	入出力パラメータ（スタック）
インスウ_ブンカイ	P^{2i} のみの項からなる高次多項式を因数分解する． $S(P)\cdot S(-P) \rightarrow S(P)$	(係数のアドレス，段数，R_{in}/R_{out} －－－ 分解後の係数アドレス，段数)
H2-1_ノ_ケイサン	反伝達関数$S(P)$などから$S(P)\cdot S(-P)-1$の係数を求める．	(係数のアドレス，ダミー，段数 －－－ 係数のアドレス，ダミー，段数，R_{in}/R_{out})
タコウシキ_テンカイ	方程式の根から，その根をもつような高次多項式の係数を求める．	(根のアドレス，段数，flag(t＝正規化) －－－ 係数アドレス，根アドレス，段数)
ダイスウ_ノ_コン	高次方程式の根を Newton 法により求める．	(係数のアドレス，段数 －－－ 係数アドレス，根アドレス，段数)

注 1 （入力スタック状態 －－－ 出力スタック状態）
注 2 R_{in}/R_{out}：入出力インピーダンスの比．ここでは 1 となる．
注 3 係数のアドレス：係数のデータを格納した配列のアドレス
注 4 段数：フィルタの段数

〔表 3〕
"THOMSON"のワード

ワ ー ド 名	機 能 説 明	入出力パラメータ（スタック）
thomson-filter	あたえられたパラメータからバタワース/トムソン・フィルタの素子値を計算する．	(n，結合係数，Z_{in}，F_c，*type* －－－ 素子のアドレス，*type*，n)
get-real-component-of-filter	基準フィルタの係数から，ハイパス/ローパス等々の実際のパラメータに合うように素子変換を行う．	(素子のアドレス，Z_{in}，F_c，*type* －－－ 素子のアドレス，n)
get-filter-component	入力インピーダンス関数から，連分数展開して基準フィルタの係数を求める．	(奇関数のアドレス，偶関数アドレス，n －－－ 素子のアドレス，n)
impedance-function	反伝達関数および特性関数から入力インピーダンス関数を求める．	(反伝達関数，特性関数，n －－－ 奇関数，偶関数，n)
mfdmfa-filter	フィルタの段数，結合係数Mから反伝達関数および特性関数を計算する． (バターワース/トムソン・フィルタ)	(n，結合係数 －－－ 反伝達関数，特性関数，n)
トクセイ_ホウテイシキ_ヲ_モトメル	反伝達関数から関係式により因数分解して特性関数を求める．	(反伝達関数，n －－－ 反伝達関数，特性関数，n)
hyblid-filter	バターワース，トムソン各フィルタの極を結合係数Mで結んで，混合フィルタの極を作る．	(*MFA* の極，*MFD*，n，結合係数 －－－ 混合フィルタ，n)

注 n＝フィルタの段数，Z_{in}＝入力インピーダンス，F_c＝カットオフ周波数，*type*＝フィルタの種類を示すパラメータ
MFA の極＝バタワース・フィルタの極の複素平面での角度，*MFD*＝トムソン・フィルタの極のアドレス
反伝達関数＝反伝達関数の係数のアドレス，特性関数＝特性関数の係数のアドレス

〔表4〕
"TFILTER"のワード

ワード名	機能説明	入出力パラメータ（スタック）
real-チェビシェフ	入力パラメータから，実際の連立チェビシェフ・フィルタの素子値を計算する．	(リプル，n，Z_{in}，F_s，F_c，*type* −−− 素子のアドレス，*type*，n)
チェビシェフ	リプル，Kファクタ，段数から基準連立チェビシェフ・フィルタの素子値をスタック上に求める．	(リプル，n，K −−− 素子値1，，，，素子値i，，)
トクセイカンスウ_ヲ_モトメル	リプル，Kファクタ，段数から連立チェビシェフ・フィルタの伝達関数および特性関数を求める．(対称形フィルタ)	(リプル，n，K −−− 特性関数，伝達関数，n)
chara-function	相反型フィルタのときの連立チェビシェフ・フィルタの伝達関数および特性関数を求める．	(リプル，n，K −−− 特性関数，伝達関数，n)
_get-k-from-ke	相反型フィルタのとき，実効Kファクタ(Ke)から，縮退時のKファクタをNewton法を使って求める．	(Ke，n　−−−　K)
t-imp-function	求められた伝達関数，特性関数から，入力インピーダンス関数を求める．	(特性関数，伝達関数，n −−− 偶関数部，奇関数部，n)
h-function	反伝達関数を求める際計算した極周波数から特性関数を求める．(対称型フィルタ)	(反伝達関数，n −−− 特性関数，反伝達関数，n)
tfil-keisu	計算したパラメータから連立チェビシェフ・フィルタの伝達関数の係数を求める．	(a_1およびρ_1^2，，a_iおよびρ_i^2，，n −−− 伝達関数，n)

※a_iとρ_i^2については図12(c)参照のこと．

〔表5〕
"TBFILTER"のワード

ワード名	機能説明	入出力パラメータ（スタック）
tw-filter	入力パラメータから，チェビシェフ・フィルタの実際の素子値を得る．	(リプル，n，Z_{in}，F_c，*type* −−− 素子のアドレス，*type*，n)
チェビシェフ_ワグナ	リプルとフィルタの段数から，基準チェビシェフ低域通過フィルタの反伝達関数の係数を求める．	(リプル，n −−− 係数のアドレス，n)
_even-トクセイ_カンスウ	相反型フィルタの極を縮退したときの特性関数を求める．	(Ω_v，，，Ω_1，n　−−−　係数のアドレス)
odd-トクセイ_カンスウ	対称型フィルタの極から特性関数を求める．	(Ω_v，，，Ω_1，n　−−−　係数のアドレス)
_トクセイ_カンスウ_セイキカ	反伝達関数と関係式を満足するように，特性関数を正規化する．	(反伝達関数，特性関数，n −−− 反伝達関数，特性関数，n)

〔表6〕
"FILTER"のワード

ワード名	機能説明	入出力パラメータ（スタック）
filter-design	このプログラムのメイン・ワード．ユーザからパラメータを得て，それぞれの処理を行う．	パラメータなし（　−−−　）
_f-responce-mode	設計またはそれを変更したフィルタの周波数特性を計算する．その結果を画面上に表示する．	パラメータなし（　−−−　）
modify-parameters	設計したフィルタの定数を，会話形式で実際の素子値に変更してゆく．	パラメータなし（　−−−　）
_thomson-design	会話形式でユーザから設計パラメータを得，バタワース/トムソン・フィルタを設計して，その結果を表示．	(n，結合係数，Z_{in}　−−−　） 出力パラメータなし
_チェビシェフ-design	会話形式でユーザから設計パラメータを得，チェビシェフ・フィルタを設計して，その結果を表示．	(リプル，n，Z_{in}　−−−　）
_レンリツ-design	会話形式でユーザから設計パラメータを得，連立チェビシェフ・フィルタを設計して，その結果を表示．	(リプル，n，Z_{in}　−−−　）
_response	設計されたフィルタの周波数特性を計算し，群遅延，振幅特性，入力インピーダンスを計算する．	(　−−−　*Delay*，[dB]，Z_0，Z_r) 入力パラメータとしての周波数がFREQの中にストアされている．

[dB]：振幅特性　　Z_0，Z_r：入力インピーダンス$Z_r e^{jZ0}$

場合は注意が必要である．

CRTのインターフェースは，CRとLFの二つのコントロール・コードのみを使いエスケープ・シーケンスは用いていない．しかし，カタカナを使っているので，ASCIIのターミナルを使っている場合は，それを変更しなければならない．

7 問題点

(1) 周波数特性のグラフィックス表示がない．

移植性を犠牲にすれば，いまのパソコンにはほとんどグラフィックスの機能が付いているので，簡単に変更できると思う．

(2) 倍精度演算の精度があまりよくない．

指数部が少ないためもあろうが，F-BASICの倍精度演算と比較して，ケタ落ちが多い．Newton法の収束条件に苦労した．

(3) 処理速度がまだ遅い．

倍精度演算を，コプロセッサなしで行っている限界だろう．

(4) プログラムの構造化がすっきりしていない．

Basicのプログラムから進化させたためか，そのなごりがある．また，筆者自身の計画性のなさからか少しづつ思い付くままに作ったためもある．ワードの名前とパラメータを，もっと整理すべきだった．

おわりに

世の中は16ビットのMS-DOS全盛であるが，どっこいCP/M-80はまだまだ使われている．機械組込みソフト開発には，現役で活躍している所も多いと思う．そのようなどこにでもある手軽な環境の中で，このプログラムは使うことができる．

今後の課題は，位相特性を調べるためにステップ応答を計算できるようにすることである．パルス伝送回路などに使う場合，これがあると設計が非常に楽になる．また数値演算ルーチンを見直して，もう一歩スピードアップを図りたい．

このプログラムは，長期にわたって少しずつ作っていったので，統一性を欠き見にくいソース・リストになってしまった．また勉強不足のため，計算のアルゴリズムにおかしな所があるかもしれない．少しでも，お役に立てば幸いである．

参考文献

1) 矢崎，武部，『伝送回路網およびフィルタ』，電子通信学会，1980
2) 八槇ほか，『波回路』，日刊工業新聞社
3) 柳沢，神林，『フィルタの理論と設計』，産報，1974
4) 柳沢ほか訳，『アナログフィルタの設計』，産業報知センター，1985

にしむら・よしかず

〔リスト〕梯子型フィルタ設計支援プログラム(ソース・リストの一部)①

```
¥----- ﾀﾞｲｽｳ ﾈｳﾃｲｼｷ ﾉ ｶｲ ｦ ﾓﾄﾒﾙ ---------------------------------
¥       Entry   Top --- number of stage
¥               2nd --- coefi array index address
¥
¥       Output  Top --- number of stage
¥               2nd --- answer index address
¥               3rd --- coefi array index address
¥-------------------------------------------------------

: ﾀﾞｲｽｳ_ﾉ_ｺﾝ
        dup epsiron dup NBUFFE !        ¥ save stage number
        _ﾀｲｺｳｼﾞ_ｦ_+_ﾆ_ｽﾙ                ¥ ( a(0), n --- a(n), n )
        _ｹｲｽｳ_ｦ_ﾋｲｷｶﾑ_ｽﾙ               ¥ ( a(0), n --- a(0), n )
        KAIBUF swap                     ¥ ( a(0), n --- a(0), x(0), n)
        _ｼｮｷﾁ_ﾉ_ｾｯﾃｲ                   ¥ ( a(0), x(0), n --- a(0), x(0), n )
        _ｼﾞｽｳ_ｶﾞ_ｷｽｳ_ﾎｦ_ｼﾗﾍﾞﾙ          ¥ (  --- a(0), x(0), n, falg )
        dup
        if                              ¥ ﾓｼ ｷｽｳｼﾞ ﾉ ﾄｷ
                w-f                     ¥ (  --- a(0), x(0), n, Ux )
                newton_method1          ¥ (  --- a(0), x(0), n, last Ux )
                _ｼﾞｽｳ_ｦ_1ｼﾞ_ｹﾞﾝ        ¥ (  --- a(0), x(0), n-1 )
        else
                drop
        then
        _ｶｲ_ﾉ_ｼｮｷﾁ                     ¥ (  --- a(0), x(0), n )
        dup 2 /
        do
                i 1+                    ¥ (  --- a(0), x(0), n, k )
                -rot d>r swap r>d       ¥ (  --- k, a(0), x(0), n )
                4 pick 3 pick arrayc@   ¥ (  --- k, a(0), x(0), n, kai )
                EPS f@ UTILT f!
                begin
                        _ﾌｸｿｽｳ_ﾉ_ｼｮｷﾁ_ｦ_ﾓﾄﾒﾙ    ¥ k,a(0),x(0),n,kai,Y',¥
                        _ﾀﾞｲﾆｭｳﾁ_ｦ_ﾓﾄﾒﾙ          ¥ k,a(0),x(0),n,kai,Z',Z
                        _newton_ﾎｳ_ﾆ_ﾖﾘ_ｱﾗﾀｼｲ_ｶｲ ¥ k,a(0),x(0),n,kai,delta
                        _a12!                   ¥ test accuracy
                until                           ¥ (  --- k, a(0), x(0), n, kai )
                _?arrow
                _ｴﾗﾚﾀ_ｶｲ_ｦ_store        ¥ (  --- k, a(0), x(0), n, kai )
                _ｼﾞｽｳ_ｦ_2ｼﾞ_ｹﾀﾞ         ¥ (  --- k, a(0), x(0), n )
                d>r swap drop r>d       ¥ (  --- a(0), x(0), n )
        loop
        _ｺﾝ_ｦ_ﾓﾄ_ﾆ_ﾓﾄﾞｽ ;

¥----- ﾀｺｳｼｷ ﾃﾝｶｲ ------------------------------------------
¥       Entry   Top --- flag ( true = normalize )
¥               2nd --- n stage
¥               3rd --- array address of root
¥               4th --- array address of coefi
¥
¥       Output  Top --- n stage
¥               2nd --- array address of root
¥               3rd --- array address of coefi
¥-------------------------------------------------------
¥       X(1) ﾆ ｱﾙ ｺﾝ ｶﾗ, ﾀｺｳｼｷ ﾃﾝｶｲ ｼﾃ,A(1) ｦ ﾓﾄﾒﾙ
¥         a(n)*x^n+...+a(1)*x+a(0)
¥-------------------------------------------------------

:R ﾀｺｳｼｷ_ﾃﾝｶｲ                   ¥ ( a(0),x(0),n,flag --- a(0),x(0),n )
        over 2 =
        if                              ¥ ﾓｼ 2ｼﾞ ﾉ ﾄｷ
                _n_ｶﾞ_2ｼﾞ_ﾉ_ｶﾞｲｲ        ¥ ( a(0),x(0),n,f --- a(0),x(0),n )
        else
                _ﾀｺｳｼｷ_ｹｲｽｳ_ｼｮｷｶ       ¥ ( a(0),x(0),n,f --- a(0),x(0),n,f,n )
                _ｺﾝﾖﾘ_ｹｲｽｳ_ｦ_ﾓﾄﾒﾙ  ¥ ( a(0),x(0),n,f,n --- ...,x(0),n,a(0),f,n )
                ｹｲｽｳ_ﾃﾝｶｲ           ¥ ( ....,x(0),n,a(0),f,n --- x(0),n,a(0),f )
                0=
                if                              ¥ if Q12=0
                        -rot                    ¥ ( --- a(0),x(0),n )
                        _normalize-ｹｲｽｳ         ¥ ( --- a(0),x(0),n )
                else
                        -rot                    ¥ ( --- a(0),x(0),n )
                then
        then ;
```

〔リスト〕梯子型フィルタ設計支援プログラム(ソース・リストの一部)②

```
¥----- H^2-1 ヲ ケイサン スル----------------------------------------
¥       Entry   Top --- n
¥               2nd --- x(0)
¥               3rd --- a(0)
¥
¥       Output  Top --- Phai (Rin/Rout)
¥               2nd --- n
¥               3rd --- x(0)
¥               4th --- a(0)
¥-------------------------------------------------------------
¥       Y=a(n)x^n+,,,,+a(1)x+a(0)
¥       Y¥Y-Phai ノ ケイスウ ヲ x(i) ニ set
¥-------------------------------------------------------------

:R H2-1_ノ_ケイサン
        _intxn                          ¥ ( a(0),x(0),n --- a(0),x(0),n )
        >r i 1+
        do
                j
                do
                        over i swap k over  ¥ ( --- a(0),x(0),j,a(0),i,a(0) )
                        arrayf@ f>r arrayf@
                        r>f f*                  ¥ A(i)*B(j)
                        i 2 mod 0<>     ¥ ( --- a(0),x(0),A*B,int(J/2) )
                        if
                                fnegate
                        then
                        i k + 6 pick d>r ij arrayf@
                        f+ r>d arrayf!
                loop
        loop
        i epsiron i 2* 1+
        do
                i over arrayf@ fabs EPS f@ f<
                if
                        i over d>r fzero r>d arrayf!
                then
        loop
        r> over f@ f>r          ¥ ( --- a(0),x(0),n )
        fzero 6 pick f! r>f ;

¥----- インスウ ブンカイ ----------------------------------------
¥       Entry   Top --- Phai ( Rin/Rout )
¥               2nd --- n
¥               3rd --- a(0),[x(0)] coefficient
¥
¥       Output  Top --- n
¥               2nd --- a(0) coefficient
¥-------------------------------------------------------------
¥       a(0) ニ Y¥Y - phi ノ ケイスウ ガ ハイッテ イル
¥       コン ノ ウチ 2 3 ショウゲン ノミ ヲ トリダシ インスウ ブンカイ シ a(0) ニ ケイスウ set
¥-------------------------------------------------------------

:R インスウ_ブンカイ            ¥ ( a(0),n --- a(0),n )
        _test-an=0                      ¥ a(0) ノ ナカデ 0 ノ モノ ヲ カゾエル
                                        ¥ ( a(0),n --- a(0),n,Xx )
        ddup - 1- 0=                    ¥ ( a(0),n,Xx --- a(0),n,Xx,flag )
        if                              ¥ If n-1=Xx
                _xn1-proc               ¥ ケイスウ ガ X^n ダケノ トキ
                                        ¥ ( a(0),n,Xx --- a(0),n )
        else
                _set-an=1               ¥ x^n ノ ケイスウ ガ 1 ニ ナル ヨウニ スル
                                        ¥ ( --- a(0),n,Xx,X )
                6 pick 2 - 6 pick =     ¥ ( --- a(0),n,Xx,X,flag )
                if                      ¥ If n-2=Xx
                        _xn2-proc       ¥ X^n ト X^[n-1] ダケ ノ トキ
                                        ¥ ( a(0),n,Xx,X --- a(0),n )
                else
                        _x^2==>x        ¥ x^2==w ト シテ ケイスウ ヲ セッテイ
                                        ¥ ( a(0),n,Xx,X --- a(0),n,Xx,X )
                        f>r d>r ij - 1- ¥ ( a(0),n,Xx,X --- a(0),n-Xx-1 )
                        ダイスウ_ノ_コン        ¥ ( a(0),n-Xx-1 --- a(0),x(0),n-Xx-1 )
                        drop r>d r>f    ¥ ( --- a(0),x(0),n,Xx,X )
                        _x^2root=>xroot ¥ ( --- a(0),x(0),n,Xx,X )
                        f>r d>r ij - 1- ¥ ( --- a(0),x(0),n-Xx-1 )
                        1               ¥ ( --- a(0),x(0),n-Xx-1,1 )
                        タコウシキ_テンカイ      ¥ ( --- a(0),x(0),n-Xx-1 )
                        drop swap r>d r>f ¥ ( --- x(0),a(0),n,Xx,X )
                        _real-coefiq    ¥ ( --- a(0),n )
                then
        then ;

¥----- MFD-MFA filter ノ トクセイ ホウテイシキ ト ハンテンタツ カンスウ -------------
¥       Entry   Top --- M ( float ) ケイコウ ケイスウ
¥               2nd --- n ジスウ
¥
¥       Output  Top --- n ジスウ
¥               2nd --- a(0) charactor function
¥               3rd --- b(0) inversed transfer function
¥-------------------------------------------------------------

:R mfdmfa-filter
        f>r mfd-filter                  ¥ ( n --- a(0),n )
        d>r j mfa-filter                ¥ ( a(0),n --- ,,,,,,n )
        drop r>d r>f hyblid-filter      ¥ ( ,,,,2*n --- a(0),n )
        トクセイ_ホウテイシキ_ヲ_モトメル ;

¥----- Input impedance parameter ---------------------------------
¥       Entry   Top --- n ジスウ
¥               2nd --- a(0) charactor function
¥               3rd --- b(0) inverse transfer function
¥
¥       Output  Top --- n ジスウ
¥               2nd --- a(0) ゲンカンスウ
¥               3rd --- b(0) キカンスウ
¥-------------------------------------------------------------

:R impedance-function
        >r i 1+
        do
                i over arrayf@          ¥ ( --- b(0),a(0),A(i) )
                i 7 pick arrayf@        ¥ ( --- b(0),a(0),A(i),B(i) )
                f+                      ¥ ( --- b(0),a(0),A(i)+B(i) )
                i 2 mod 0=              ¥ ( --- b(0),a(0),X,even flag )
                if
                        i 6 pick arrayf!
                        fzero i 7 pick arrayf!
                else
                        i 7 pick arrayf!
                        fzero i 6 pick arrayf!
                then
        loop
        r> ;

¥----- Get real component of filter -----------------------------
¥       Entry   Top --- type    0=low-l 1=low-h 2=hihg-l 3=hihg-h
¥               2nd --- Cut off frequency ( Float )
¥               3rd --- input impedance ( Float )
¥               4th --- n ジスウ
¥               5th --- array address of coefi
¥
¥       Output  Top --- n ジスウ
¥               2nd --- array address of filter component
¥-------------------------------------------------------------

:R get-real-component-of-filter
        >r f 2 fpai f* f* sto-a         ¥ ( --- a(0),n,Z )
        f>r ddup r>d rot r>d rot        ¥ ( --- a(0),n,a(0),Z,n )
        dup 2 mod 0<> swap r>           ¥ ( --- a(0),n,a(0),Z,odd,n,flag )
        select
                _thomson-low-limp
                _thomson-low-himp
                _thomson-high-limp
                _thomson-high-himp
        endselect
        execute
        5 ndrop ;

¥----- main program for thomson ----------------------------------
¥       Entry   Top --- Flag ( true=lowpass, false=highpass )
¥               2nd --- Fc ( Cut off frequency )
¥               3rd --- Z  ( Input impedance )
¥               4th --- M ( float )
¥               5th --- n ( filter ノ ダンスウ )
¥
¥       Output  Top --- n
¥               2nd --- f
¥               3rd --- a(0)
¥-------------------------------------------------------------

:R thomson-filter
        >r f>r f>r mfdmfa-filter        ¥ ( --- b(0),a(0),n )
        impedance-function              ¥ ( --- b(0),a(0),n )
        get-filter-component            ¥ ( --- b(0),n )
        r>f r>f i get-real-component-of-filter r> swap ;

¥----- normalized component for cauer filter --------------------
¥       Entry   Top --- K ( float )
¥               2nd --- n ( ダンスウ )
¥               3rd --- Alpha ( float )
¥
¥       Output  Top --- ,,,,, ( component of filter )
¥-------------------------------------------------------------

:R チェビシェフ
        5 pick 2 /mod drop              ¥ ( --- Al,n,K,odd-flag )
        if
                トクセイカンスウ-ヲ-モトメル    ¥ ( --- h(0),a(0),n )
        else
                5 pick _get-k-from-ke   ¥ ( --- Al,n,Ke )
                chara-function          ¥ ( --- h(0),a(0),n )
        then
        t-impe-function                 ¥ ( --- x(0),y(0),n )
        even-first                      ¥ ( --- x(0),y(0),n )
        dup 1- 2 / >r                   ¥ ( --- x(0),y(0),n ) [j]
        do                              ¥ ( --- x(0),y(0) )
                1do                     ¥ If do start from 0 then set 1
                d>r k 5 rpick r>d       ¥ ( --- i,j,x(0),y(0) )
                get-filter-comp         ¥ ( --- Comp,x(0),y(0) )
                d>r k 4 rpick r>d       ¥ ( --- Comp,i,n,x(0),y(0) )
                1-node                  ¥ ( --- Comp,i,n,x(0),y(0) )
                1+i
                f>r 1+i fi d>r drop 9 rpick r>d ¥ ( --- Comp,i+1,j,x(0),y(0) )
```

```
        2nd-comp                ¥ ( --- Comp1,Comp2,x(0),y(0),Comp2 )
        get-capacitor           ¥ ( --- C1,C2,CC,x(0),y(0),Comp2 )
        j i new-keisu2          ¥ ( --- Comp1,Comp2,x(0),y(0),Comp2 )
        _6ndrop r>f i 1+ j =    ¥ ( --- Comp1,Comp2,i,n,x(0),y(0),flag )
        if
                get-real-root   ¥ ( --- Comp1,Comp,,,,)
                leave
        else
                ﾃﾝｶｲ3           ¥ ( --- B,,,B,i,n,x(0),y(0) )
                store-keisu
        then
    loop
    rdrop ;

¥----- Main program for real component of cauer filter ---------
¥       Entry   Top --- flag    0=low-himp,1=low-limp
¥                               2=high-himp
¥               2nd --- Fc      cut off frequency
¥               3rd --- Fa      shut off frequency
¥               4th --- Z       input frequency
¥               5th --- n       ( node )
¥               6th --- A       ( Ripple dB )
¥
¥       Output  Top --- n       ( node )
¥               2nd --- f       0,1,2
¥               2nd --- a(0)    index
¥-------------------------------------------------------------

:R real-ﾌｨﾙﾀ
    >r i 2 <>
    if                          ¥ If ne highpass filter
        fswap
    then
    get-k-factor                ¥ ( --- A,n,Z,K,Fcc )
    f>r fswap f>r               ¥ ( --- A,n,K ) [f,Fcc,Z]
    5 pick >r ﾌｨﾙﾀ i             ¥ ( --- ,,,,,n )
    1+ 2 /mod 1- 3 * 1+ +
    do
        j i - bkeisu arrayf!
    loop
    r> r>f r>f f2* fpai f* sto-a r> ¥ ( --- n,Z,f )
    6 pick d>r i                ¥ ( --- n,Z,f )
    case
        0 _low-himp
        1 _low-limp
        2 _high-himp
        nocase _low-himp
    endcase
    execute
    fdrop r>d ;

¥----- ﾌｨﾙﾀ ﾌｧﾝｸｼｮﾝ ﾌｨﾙﾀ ﾉ ｿｼ ｹｲｽｳ ｦ ｾｯﾄ --------------------
¥       Entry   Top --- flag ( true=lowpass )
¥                           ( false = highpass )
¥               2nd --- Fc ( Cutoff frequency )
¥               3rd --- Z ( Input impedance )
¥               4th --- n ( node )
¥               5th --- Ripple ( dB )
¥
¥       Output  Top --- n
¥               2nd --- flag
¥               3rd --- a(0)
¥-------------------------------------------------------------

:R tw-filter
    >r f>r f>r >r dB> r>        ¥ ( --- Ripple,n )
    ﾌｨﾙﾀ ｼｭﾄｰﾜｰﾄﾞ               ¥ ( --- a(0),n )
    >r i _omega-ﾃｰﾌﾞﾙｾｯﾃｲﾝ fdup RATIO f!     ¥ ( --- a(0),Omegan,,,Omega1 )
    i 2 /mod swap 0=            ¥ ( --- a(0),Omen,,Ome1,n/2,even)
    if                          ¥ If n is even number
        _even-ﾄｳｶｾｲｰｶﾝｽｳ        ¥ ( --- a(0),b(0) )
        swap (keisu + 16) i h2-1_ﾉ_ﾃｲｷﾞﾝ ¥ ( --- b(0),a(0),keisu,n,Phi )
        6 pick f! _even-t^2     ¥ ( --- b(0),a(0),keisu,n )
        fzero keisu f! fzero i keisu arrayf!
        drop2nd keisu swap 1+ ｲﾝｽｳ_ﾌﾞﾝｶｲ
        drop drop2nd 8 + swap   ¥ ( --- b(0),a(0) )
    else
        odd-ﾄｳｶｾｲｰｶﾝｽｳ          ¥ ( --- b(0),a(0) )
    then
    r> _ﾄｳｶｾｲ_ｶﾝｽｳ_ｾｲｷｶ        ¥ ( --- b(0),a(0),n )
                                ¥ a(0)=inverse, b(0)=charactor func
    impedance-function
    get-filter-component
    r>f r>f i
    get-real-component-of-filter r> swap ;

¥-------------------------------------------------------------
¥       This is the aria for saving filter parameters
TYPE            dsw     1       ¥ Filter type
HIGHLOW         dsw     1       ¥ Lowpass/Highpass
IMPTYPE         dsw     1       ¥ Impedance type
NODE            dsw     1       ¥ node
CUTOFF          dsw     4       ¥ Cutoff frequncy
SHUTOFF         dsw     4       ¥ Shut off freqency
IMPEDUNCE       dsw     4       ¥ Input impedance
RIPPLE          dsw     4       ¥ Ripple (dB)
MFACTOR         dsw     4       ¥ M factor for thomson filter
INITFLAG        dsw     1       ¥ Initial flag
FREQ            dsw     4       ¥ Plot frequency buffer

¥----- Print thomson filter -------------------------------------
¥       Entry   Top --- flag 1=lowpass 0=highpass
¥               2nd --- Cutoff frequency
¥               3rd --- Input impedance
¥               4th --- M factor
¥               5th --- n ( node )
¥
¥       Output  Top --- n
¥               2nd --- a(0)
¥-------------------------------------------------------------

: print-thomson
        14 pick real-type-nor
        thomson-filter swap             ¥ ( --- a(0),n,flag )
        store-normal-filter             ¥ ( --- a(0),n,flag )
        select
                _print-low-n
                _print-low-n-himp
                _print-high-n
                _print-high-n-himp
        endselect
        execute ;

¥----- Print normal ﾌｨﾙﾀ filter ---------------------------
¥       Entry   Top --- flag 1=lowpass 0=highpass
¥               2nd --- Cutoff frequency
¥               3rd --- Input impedance
¥               4th --- n ( node )
¥               5th --- Ripple ( dB )
¥
¥       Output  Top --- n
¥               2nd --- a(0)
¥-------------------------------------------------------------

: print-tw
        10 pick real-type-nor
        tw-filter swap                  ¥ ( --- a(0),n,flag )
        store-normal-filter             ¥ ( --- a(0),n,flag )
        select
                _print-low-n
                _print-low-n-himp
                _print-high-n
                _print-high-n-himp
        endselect
        execute ;

¥----- Print ﾌｨﾙﾀ filter component --------------------------
¥       Entry   Top --- flag ( 0=low-h,1=low-l,2=high-h )
¥               2nd --- Cut off frequency
¥               3rd --- Supressed frequency
¥               4th --- Input impeduce
¥               5th --- n ( node )
¥               6th --- Ripple ( dB )
¥
¥       Output  Top --- n
¥               2nd --- bkeisu ( parameter top address )
¥-------------------------------------------------------------

: print-ﾌｨﾙﾀ
        _real-imp-type              ¥ ( --- no change )
        real-ﾌｨﾙﾀ swap               ¥ ( --- a(0),n,flag )
        store-che-filter            ¥ ( --- a(0),n,flag )
        select
                _print-low-h
                _print-low-l
                _print-low-l
        endselect
        execute ;

¥----- Calculate filter responce for one sampled data ----------
¥       ( --- Phase,Argument,Input-impedance )
¥-------------------------------------------------------------

:R freq-responce                    ¥ ( --- Phase,Arg,Zin )
        fzero fone fzero IMPEDUNCE f@ finv 0    ¥ ( --- Vi,Ii,index )
        begin
                LCR-para-imp        ¥ ( --- Vi,Ii,Zp,index )
                >r c-over c>r c-* c-+   ¥ ( --- Vo )
                r>c r> cwswap c>r   ¥ ( --- Vo,index )
                LCR-ser-imp >r      ¥ ( --- Vo,Zs)
                c-over c-swap c-/ r> r>c ¥ ( --- Vo,I',index,II )
                wcswap >r c-+ r>    ¥ ( --- Vo,I',index )
                _test-end-para      ¥ ( --- Vo,Ii,index,flag )
        until
        drop c-over c-over c-/ c>r  ¥ ( --- Vo,Io )
        fzero IMPEDUNCE f@ c-* c-+  ¥ ( --- Vr )
        cinv r==>p r>c r==>p ;
¥----- Get parameter for each type of filter -------------
¥       Entry   Top --- empty
¥
¥       Output  Top --- n ( node )
¥               2nd --- a(0)
¥-------------------------------------------------------------
```

```
: _runnig       crlf crlf s'>>> ﾀﾀﾞｲﾏ ｹｲｻﾝﾁｭｰ ' sprint ;

:R _thomson-design
        _prompt-cutoff                  ¥ ( --- n,M,Imp,Cut )
        _prompt-highlow                 ¥ ( --- n,M,Impe,Cutoff,flag )
        2* _prompt-imp-type +           ¥ ( --- n,M,Impe,Cutoff,type )
        _runnig
        print-thomson ;

:R _ﾁｴﾋﾞｼｪﾌ-design
        _prompt-cutoff                  ¥ ( --- Ripple,n,Impe,Cutoff )
        _prompt-highlow                 ¥ ( --- Ripple,n,Impe,Cutoff,flag )
        2* _prompt-imp-type +           ¥ ( --- n,M,Impe,Cutoff,type )
        _runnig
        print-tw ;

:R _ﾚﾝｿﾘｰ-design
        _prompt-highlow >r
        _prompt-shutoff                 ¥ ( --- Ripl,n,Impe,Shut )
        _prompt-cutoff c-dup f< r>      ¥ ( --- Ripl,n,Impe,Shut,Cut,fl,flag )
        if                              ¥ If highpass filter
                if                      ¥ If shutoff < cutoff
                        _runnig
                        2 print-ﾁｴﾋﾞｼｪﾌ
                else
                        s'Error Shut < Cut !' sprint crlf
                        8 ndrop _ﾚﾝｿﾘｰ-design
                then
        else
                if
                        s'Error Shut > Cut !' sprint crlf
                        8 ndrop _ﾚﾝｿﾘｰ-design
                else
                        _prompt-imp-type
                        _runnig
                        print-ﾁｴﾋﾞｼｪﾌ
                then
        then ;

¥----- Calculate respoce included group delay ------------------
¥       ( --- Delay,Gain,Phase,Impedance )
¥-------------------------------------------------------------

:R _responce                            ¥ ( --- Delay,dB,Phase,Ohm)
        _tra-omega sto-a
        freq-responce c-swap            ¥ ( --- Pr,Ar,P,A )
        rcl-a f 1.01 f* sto-a
        freq-responce 12 ndrop          ¥ ( --- Pr,Ar,P,A,P' )
        fswap f>r fswap f-              ¥ ( --- Pr,Ar,P'-P )
        rcl-a _tra-omega f- f/ c>r      ¥ ( --- Pr )
        fpai f/ fdup fone f>            ¥ ( --- Pr',f )
        if
                f 2 f-
        then
        f 180 f*
        r>c r>f >dB ;

¥-----Calculate frequency responce of filter ------------------
¥       ( --- )
¥-------------------------------------------------------------

:R _f-responce-mode                     ¥ ( --- )
        INTFLAG @ 0<>
        if
                s' Filter ﾉ ｼﾞｮｰｽｳ ｶﾞ ﾐﾃｲ ﾃﾞｽ !!' sprint crlf crlf
        else
                begin
                        _prompt-sample          ¥ ( --- Freq )
                        f0<>
                        if
                                _responce       ¥ ( --- Pr,Rr,Time,A )
                                _print-responce
                                crlf
                        repeat
                then ;

¥----- Prompt modifing parameters ------------------------------
¥       ( --- )
¥-------------------------------------------------------------

:R modify-parameters                    ¥ ( --- )
        INTFLAG @ 0<>
        if
                s' Filter ﾉ ｼﾞｮｰｽｳ ｶﾞ ﾐﾃｲ ﾃﾞｽ !!' sprint crlf crlf
                RET;
        then
        0 NODE @ d>r
        begin
                _fechimp 1+i
                j mod-para-imp r>d drop swap d>r
                i 1- _compo! i _compo! 1+i
                _fechimp 1+i
                j mod-serial-imp r>d drop swap d>r
                i 1- _compo! i _compo! 1+i
                i _test-end-para drop2nd
        until
        r>d ddrop ;

¥----- Main program --------------------------------------------
¥       ( --- )
¥-------------------------------------------------------------

:R filter-design                                ¥ ( --- )
        savesp                                  ¥ Save parameter stack pointer
        _int-parameter
        _print-opening                          ¥ ( --- )
        begin
                _prompt-type >r
                i 5 <>
                if                              ¥ If ne reboot
                        i 2 >                   ¥ If ne filter design
                        if
                                r> 3 -
                                select
                                        _f-responce-mode
                                        modify-parameters
                                endselect
                                execute
                        else
                                r> _filter-parameter    ¥ ( --- R,n,Imp,typ )
                                0 INTFLAG !             ¥ Intflag reset
                                select
                                        _thomson-design
                                        _ﾁｴﾋﾞｼｪﾌ-design
                                        _ﾚﾝｿﾘｰ-design
                                endselect
                                execute crlf crlf
```

本プログラムをPC-9801で動かす方法

本プログラムをPC-9801で実行させるにはつぎの方法がある.

(1) Fifth86を用いて再コンパイルする方法

リギーFifthシリーズは,共通文法で統一されており,アセンブラの部分(rpickとfrpickの2ワード)などを変更すれば8086での実行ファイルを作ることができると思う(コンソール出力のDOSインターフェース部も注意する必要がある)が,少しめんどうかもしれない.

(2) PC-9801上のZ80カードを用いる方法

Z80またはHD64180カードが数社から出ているので,それを用いてCP/M-80のCOMファイルを実行させる.

(3) Z80シミュレーション・ソフトを用いる方法

CP/M-86およびMS-DOS用に,これも数社から出ている.筆者の場合は,メガソフトのEM/3+を用いて実行させている.

本ソフトを頒布します!!

ここで示した「梯子型フィルタ設計支援プログラム」と,つぎの記事で示す「トランジェント・アナライザ」の二つのソフトを1枚のディスケットに収めて頒布します.いずれも,ソース・リスト(FifthZ80)とオブジェクト・ファイルを含みます.頒布メディアはつぎのものに限らせていただきます.

① 標準CP/M-80　8インチ 1S
② FM-7, PC-8801用　5インチ 2D
③ SMC-777用　3.5インチ 1DD

▶申し込み方法

費用(ディスケット代を含む)4,000円をを下記銀行口座にお振り込みください.なお振込み用紙の通信欄に必ず住所・氏名と希望のメディアをご記入ください.

口座番号:三菱銀行駒込支店　普通預金 No. 4694934
口座名:CQ出版㈱　C&E出版部