験震時報（2023）
第 86 巻 ： 6

報文（REPORT）

広帯域地震計周波数特性補正のための再帰型ディジタルフィルター

Recursive Digital Filter for Frequency Response Deconvolution of Broadband Seismometers

勝間田明男 [1]

KATSUMATA Akio[1]

(Received November 21, 2022: Accepted March 14, 2023)

1 はじめに

W phase 解析 (Kanamori and Rivera, 2008)，長周期成分モニター（気象庁, 2013）などにおいては，広帯域地震計の周波数特性を再帰型ディジタルフィルターによって実時間処理の補正をしている．Kanamori and Rivera (2008) では広帯域地震計の低周波数側の周波数特性を固有周期と減衰定数をパラメーターとして補正している．Maeda et al. (2011) は，短周期地震計の記録を広帯域地震計相当に補正する再帰型フィルターを用いて，地震記録のアレイ解析を行った．小久保 (2013) は傾斜計の過渡応答の解析に用いる再帰型フィルターを提案している．Katsumata et al. (2021) では広帯域地震計の周波数特性として与えられている 2 つあるいは 1 つの極の値を直接用いて，再帰型ディジタルフィルターを構築することを提案している．以上のように地震計等の周波数特性の補正を必要とする場合がある．

IRIS (Incorporated Research Institute for Seismology) から得た広帯域地震計の極・零点情報を確認すると，1 組の共役の極によって低周波側の周波数特性を記述している広帯域地震計が大半ではあるものの，3 つの極や零点も含めて低周波側の周波数特性を表現している広帯域地震計もある．ここでは，そのような様々な広帯域地震計の周波数特性を補正するための再帰型ディジタルフィルターの理論とプログラムを示す．

2 理論

広帯域地震計の低周波側の周波数特性が，以下のようにラプラス変換の形式により表現されているとする．これは振幅が高周波側において平坦，低周波側で増加あるいは減少する特性を示している．

$$H(s)=\prod_j \frac{(s-s_{jz1})(s-s_{jz2})}{(s-s_{jp1})(s-s_{jp2})} \tag{1}$$

ここで，s_{jz1}, s_{jz2} は共役もしくは両方実数の零点，s_{jp1}, s_{jp2} は共役もしくは両方実数の極である．STS-1 のように 1 組の共役な極のみで表せられる場合には，総積は $j=1$だけになり，$s_{jz1}=0$, $s_{jz2}=0$である．極あるいは零点が一つのみの場合には，他の極・零点が 0 であるとする．この周波数特性を補正するためには，$H(s)$ の逆数の周波数特性を考える．この周波数特性を再帰型ディジタルフィルターとして実現するためには，s に式（2）の双一次変換（例えば，三谷, 1987）を代入して，$H^{-1}(s)$を$H^{-1}(z)$に変換する．

$$s=c\frac{1-z^{-1}}{1+z^{-1}} \tag{2}$$

ここで$c=2/\Delta t$，Δtはサンプリング間隔である．$H^{-1}(z)$を整理すると次のようになる．

$$H^{-1}(z)=\prod_j G_j \frac{1+a_{j1}z^{-1}+a_{j2}z^{-2}}{1+b_{j1}z^{-1}+b_{j2}z^{-2}} \tag{3}$$

$$G_j=\frac{c^2-c(s_{jp1}+s_{jp2})+s_{jp1}s_{jp2}}{c^2-c(s_{jz2}+s_{jz1})+s_{jz1}s_{jz2}}$$

[1] 富山大学都市デザイン学系，Faculty of Sustainable Design, University of Toyama

$$a_{j1} = \frac{-2c^2 + 2s_{jp1}s_{jp2}}{c^2 - c(s_{jp1} + s_{jp2}) + s_{jp1}s_{jp2}}$$

$$a_{j2} = \frac{c^2 + c(s_{jp1} + s_{jp2}) + s_{jp1}s_{jp2}}{c^2 - c(s_{jp1} + s_{jp2}) + s_{jp1}s_{jp2}}$$

$$b_{j1} = \frac{-2c^2 + 2s_{jz1}s_{jz2}}{c^2 - c(s_{jz1} + s_{jz2}) + s_{jz1}s_{jz2}}$$

$$b_{j2} = \frac{c^2 + c(s_{jz1} + s_{jz2}) + s_{jz1}s_{jz2}}{c^2 - c(s_{jz1} + s_{jz2}) + s_{jz1}s_{jz2}}$$

入力をx_k，出力をy_kとすると，式（3）の係数を用いて次の式のように時間領域で処理可能である．

$$y_k = G_j(x_k + a_{j1}x_{k-1} + a_{j2}x_{k-2}) - b_{j1}y_{k-1} - b_{j2}y_{k-2}$$

このフィルターの適用例を Fig. 1 に示す．それぞれの零点・極の値を Table 1 に示す．ここでは零点・極の値は IRIS から SACPZ ではじまる名前のファイルとして提供されている地動変位に対する特性を参照している．そのため広い範囲で地動速度に対して平坦となる広帯域地震計の周波数特性は，周波数に対して傾き 1 のグラフとなっている．Fig. 1 は極が三つある場合や，零点も用いて低周波の周波数特性が表現されている例を含む．

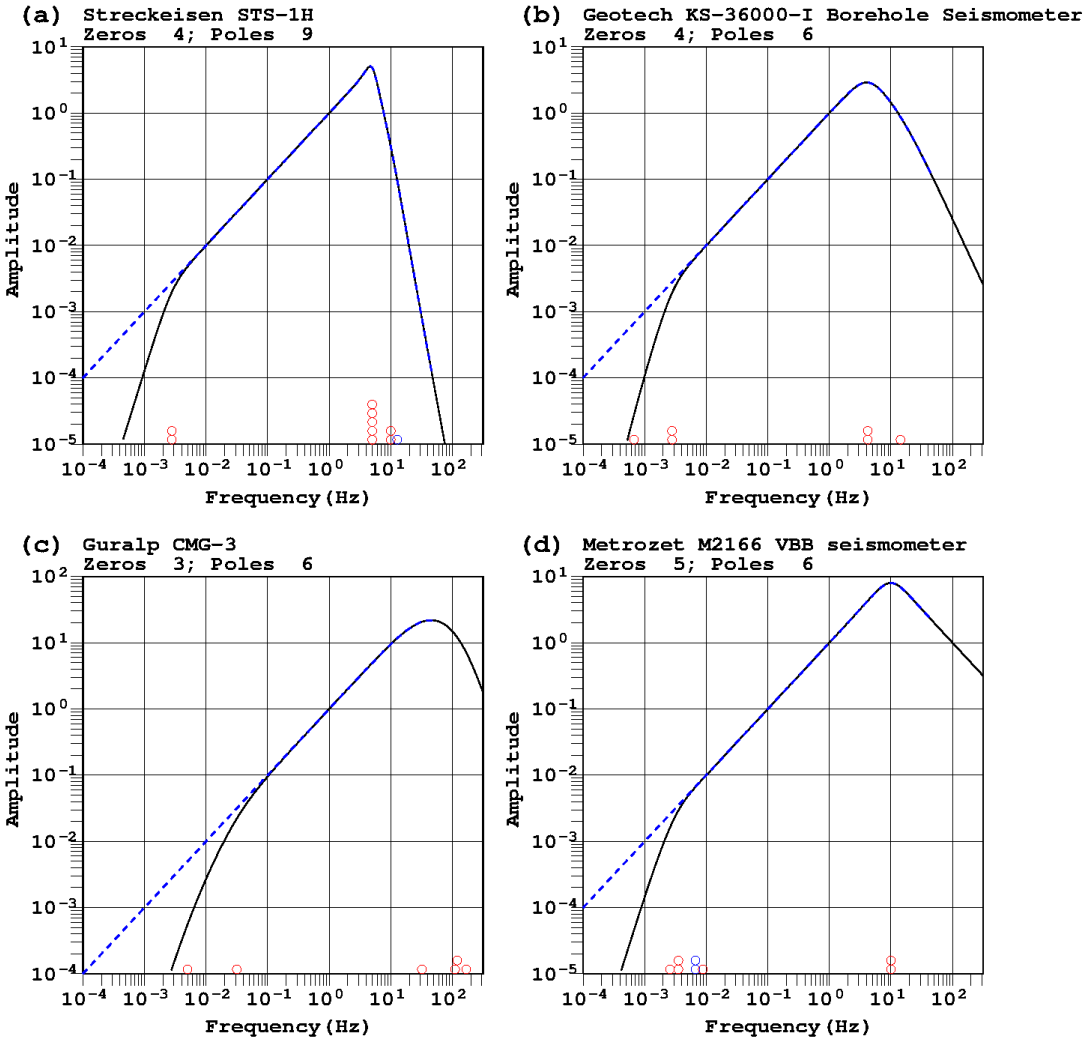

Fig. 1　周波数特性補正のためのフィルターの例．縦軸は地動変位の振幅の相対的な周波数特性を示す．黒い実線は極と零点（Table 1）から計算される地震計の周波数特性を，青い破線は地震計の周波数特性と式（3）により示される周波数特性補正フィルターの周波数特性（データのサンプリング間隔は 0.01 s を仮定）の積を表す．グラフの下に示す赤及び青の丸はそれぞれ $|s_{jp*}|/2\pi$ と $|s_{jz*}|/2\pi$ (*, 1 or 2) の値を示す．

Fig. 1 に実線により示した周波数特性は，s に $i2\pi f$ を代入して求めている．なお i は虚数単位，f は周波数である．Fig. 1 の青破線は，極・零点により与えられた周波数特性に，式 (3) で示す再帰型ディジタルフィルターの式に $z^{-1} = \exp(-i2\pi f\Delta t)$ を代入して計算される周波数特性を掛け合わせて求めている．再帰型ディジタルフィルターの周波数特性は，斎藤 (1978) によるサブルーチン RECRES にフィルター係数を代入することより求めている．なお，実用上は計器特性を補正した上に低域遮断のフィルターをかけないと発散を起こす．

Table 1　Fig. 1 に示した周波数特性の極と零点．それぞれの値は（実部，虚部）を表す．

	極	零点
(a)	(-0.123413E-01,0.123413E-01), (-0.123413E-01,-0.123413E-01), (-0.391757E+02,0.491234E+02), (-0.391757E+02,-0.491234E+02), (-0.282743E+02,0.136939E+02), (-0.282743E+02,-0.136939E+02), (-0.700575E+01,0.306248E+02), (-0.700575E+01,-0.306248E+02), (-0.314159E+02,0.0)	(0.0,0.0),(0.0,0.0), (0.0,0.0), (-0.797964E+02, 0.0)
(b)	(-0.184300E+02,0.189100E+02), (-0.184300E+02,-0.189100E+02), (-0.123400E-01,0.123400E-01), (-0.123400E-01,-0.123400E-01), (-0.898500E+02,0.0), (-0.421900E-02,0.0)	(0.0,0.0),(0.0,0.0), (0.0,0.0),(0.0,0.0)
(c)	(-0.314200E-01,0.0), (-0.197900E+00,0.0), (-0.201100E+03,0.0), (-0.697400E+03,0.0), (-0.754000E+03,0.0), (-0.105600E+04,0.0)	(0.0,0.0),(0.0,0.0), (0.0,0.0)
(d)	(-0.184532E-01,0.123400E-01), (-0.184532E-01,-0.123400E-01), (-0.391800E+02,0.491200E+02), (-0.391800E+02,-0.491200E+02), (-0.161798E-01,0.0), (-0.552855E-01,0.0)	(-0.419870E-01,0.0), (-0.419870E-01,0.0), (0.0,0.0),(0.0,0.0), (0.0,0.0)

3　プログラム

式（3）を c 言語のプログラムとしたものを deconv_b2v として次に示す．このプログラムでは，極・零点の絶対値を2πで除した値が 0Hz 以外の 0.1Hz よりも低周波のものを処理対象としている．Fortran プログラムからの呼び出しを想定して引数はアドレス渡しとしている．広帯域地震計の低周波側の計器特性を補正すると，地動速度について平坦な出力となる．それに数値積分を行うと地動変位に平坦な特性となる．地動変位について平坦な特性とする関数 deconv_b2d も併せて示す．各引数は以下のとおりである．

npoles　極の数（入力）．
poler[]　各極の実部（入力）．
polei[]　各極の虚部（入力）．
nzeros　零点の数（入力）．
zeror[]　各零点の実部（入力）．
zeroi[]　各零点の虚部（入力）．
dt_in　サンプリング間隔 (s)　（入力）．
m_filt　フィルターの段数（出力）．
gn　フィルターの倍率係数（出力）．式（3）の $\prod_j G_j$ にあたる．
hfilt[]　フィルター係数（出力）．hfilt[4(j-1)+0～3] はそれぞれ式（3）の a_{j1}，a_{j2}，b_{j1}，b_{j2} にあたる．
ierror　エラー出力（出力）．

```
#include <stdio.h>
#include <math.h>
//================================================
//   deconvolution based on poles and zeros
//        broadband seismogram response -> velocity
//           (velocity flat)
//================================================
void deconv_b2v_(
    int *npoles,
    double poler[],
    double polei[],
    int *nzeros,
    double zeror[],
    double zeroi[],
    double *dt_in,
    int    *m_filt,
    double *gn,
    double hfilt[],
    int    *ierror)
{
   double dt;
   int i,j,idx_lp_conj[50],idx_lp_real[50],idx_lz_conj[50],
       idx_lz_real[50];
   int f_checked[50];
   double omega0_pole[50],omega0_zero[50],v_product_p,v_sum_p,
          v_product_z,v_sum_z;
   double freq_w,gn_p,gn_z;
   double pi = M_PI;
   int n_lp_real, n_lp_conj,n_lz_real, n_lz_conj;
   int nr_pole_conj, nr_pole_real, nr_zero_conj, nr_zero_real;
   int iset_conj;

   *ierror = 0;
   *m_filt = 0;
   *gn = 1.0;

   if(*npoles > 50 || *nzeros > 50){
      printf("deconv_b2v_ npoles(%d) > 50 or nzeros(%d) > 50)¥n",
             *npoles , *nzeros);
      *ierror = 1;
   }

   dt = *dt_in;
//--------- search low freq. poles ---------------------------
   n_lp_real = 0;
   n_lp_conj = 0;
   for(i=0;i< *npoles;i++) f_checked[i] = 0;
   for(i=0;i< *npoles;i++){
      if(f_checked[i] == 0){
         omega0_pole[i] = sqrt(poler[i] * poler[i]
                             + polei[i] * polei[i]);
         freq_w = omega0_pole[i] / (2.0 * pi);
         if(freq_w < 0.1){
         //--------- real pole ?
```

```
            if(fabs(polei[i]) < fabs(poler[i]) * 0.01){
               idx_lp_real[n_lp_real] = i;
               n_lp_real++;
               f_checked[i] = 1;
            }
            else{
          //---------- find conjugate pole
                iset_conj = 0;
                if(i < *npoles - 1){
                   for(j=i+1;j < *npoles ; j++){
                      if(fabs(poler[i]-poler[j]) < fabs(poler[i])
                                              * 0.01 &&
                         fabs(polei[i]+polei[j]) < fabs(polei[i])
                                              * 0.01){
                         idx_lp_conj[n_lp_conj] = i;
                         n_lp_conj++;
                         f_checked[i] = 1;
                         idx_lp_conj[n_lp_conj] = j;
                         n_lp_conj++;
                         f_checked[j] = 1;
                         iset_conj = 1;
                         break;
                      }
                   }
                }
                if(iset_conj  == 0){
                   printf(
                       "deconv_pole_b2v_3   error,couldn't   found
paired pole for %.3e %.3e¥n",
                         poler[i],polei[i]);
                   *ierror = 3;
                   return;
                }
            }
          }
          else{         // if(freq_w < 0.1)
             f_checked[i] = 1;
          }  // if(freq_w < 0.1)
       }  // if(f_checked[i] == 0)
    }     // for(i=0;i< *npoles;i++)
    if( n_lp_real == 0 && n_lp_conj == 0){
       *ierror = 2;
       printf("deconv_pole_b2v_3 N of low freq. pole is zero¥n");
       return;
    }
  //--------- search low freq. zeros --------------------------
    n_lz_real = 0;
    n_lz_conj = 0;
    for(i=0;i< *nzeros;i++) f_checked[i] = 0;
    for(i=0;i< *nzeros;i++){
       if(f_checked[i] == 0){
          omega0_zero[i] = sqrt(zeror[i] * zeror[i]
                              + zeroi[i] * zeroi[i]);
          freq_w = omega0_zero[i] / (2.0 * pi);
          if(freq_w < 0.1 && freq_w > 0.0001){
          //--------- real zero ?
             if(fabs(zeroi[i]) < fabs(zeror[i]) * 0.01){
                idx_lz_real[n_lz_real] = i;
                n_lz_real++;
                f_checked[i] = 1;
             }
             else{
          //----------- find conjugate zero
                iset_conj = 0;
                if(i < *nzeros -1){
                   for(j=i+1;j < *nzeros ; j++){
                      if(fabs(zeror[i] - zeror[j])
                            < fabs(zeror[i]) * 0.01 &&
                         fabs(zeroi[i] + zeroi[j])
                            < fabs(zeroi[i]) * 0.01){
                         idx_lz_conj[n_lz_conj] = i;
                         n_lz_conj++;
                         f_checked[i] = 1;
                         idx_lz_conj[n_lz_conj] = j;
                         n_lz_conj++;
                         f_checked[j] = 1;
                         iset_conj = 1;
                         break;
                      }
                   }
                }
                if(iset_conj  == 0){
                   printf(
                       "deconv_pole_b2v_3    error,couldn't    found
paired zero for %.3e %.3e¥n",
                             zeror[i],zeroi[i]);
                   *ierror = 3;
                   return;
                }
             }
          }
          else{
             f_checked[i] = 1;
          }  // if(freq_w < 0.1)
       }  // if(f_checked[i] == 0)
    }     // for(i=0;i< *npoles;i++)
    printf("deconv_b2v_  n_lp_conj=%d  n_lp_real=%d  n_lz_conj=%d
n_lz_real=%d¥n",
            n_lp_conj,n_lp_real,n_lz_conj,n_lz_real);
  //-------------------------------------------------------
  //        construct filter
  //-------------------------------------------------------
    nr_pole_conj = n_lp_conj;   // remaining conjugate poles
    nr_pole_real = n_lp_real;   // remaining real poles
    nr_zero_conj = n_lz_conj;   // remaining conjugate zeros
    nr_zero_real = n_lz_real;   // remaining real zeros
    while( nr_pole_conj > 0 || nr_pole_real > 0 ||
           nr_zero_conj > 0 || nr_zero_real > 0 ){

       v_product_p = 0.0;
       v_sum_p = 0.0;
       v_product_z = 0.0;
       v_sum_z = 0.0;
  //----- conjugate poles ---------------------------------
       if(nr_pole_conj >= 2){
          v_product_p =
            (dt * omega0_pole[idx_lp_conj[nr_pole_conj-2]] / 2.0)
          * (dt * omega0_pole[idx_lp_conj[nr_pole_conj-2]] / 2.0);
          v_sum_p =
            (dt * poler[idx_lp_conj[nr_pole_conj-2]] / 2.0) * 2.0;
          nr_pole_conj -= 2;
       }
  //----- real poles --------------------------------------
       else if(nr_pole_real >= 2){
          v_product_p =
              (dt * poler[idx_lp_real[nr_pole_real - 2]] / 2.0)
            * (dt * poler[idx_lp_real[nr_pole_real - 1]] / 2.0);
          v_sum_p =
              (dt * poler[idx_lp_real[nr_pole_real - 2]] / 2.0)
            + (dt * poler[idx_lp_real[nr_pole_real - 1]] / 2.0);
          nr_pole_real -= 2;
       }
       else if(nr_pole_real >= 1){
          v_product_p = 0.0;
          v_sum_p =
              (dt * poler[idx_lp_real[nr_pole_real - 1]] / 2.0);
          nr_pole_real -= 1;
       }
  //----- conjugate zeros ---------------------------------
       if(nr_zero_conj >= 2){
          v_product_z =
            (dt * omega0_zero[idx_lz_conj[nr_zero_conj-2]] / 2.0)
          * (dt * omega0_zero[idx_lz_conj[nr_zero_conj-2]] / 2.0);
          v_sum_z =
            (dt * zeror[idx_lz_conj[nr_zero_conj-2]] / 2.0) * 2.0;
          nr_zero_conj -= 2;
       }
  //----- real zeros --------------------------------------
       else if(nr_zero_real >= 2){
          v_product_z =
            (dt * zeror[idx_lz_real[nr_zero_real-2]] / 2.0)
          * (dt * zeror[idx_lz_real[nr_zero_real-1]] / 2.0);
          v_sum_z =
              (dt * zeror[idx_lz_real[nr_zero_real-2]] / 2.0)
            + (dt * zeror[idx_lz_real[nr_zero_real-1]] / 2.0);
          nr_zero_real -= 2;
       }
       else if(nr_zero_real >= 1){
          v_product_z = 0.0;
          v_sum_z = (dt * zeror[idx_lz_real[nr_zero_real - 1]] /
2.0);
          nr_zero_real -= 1;
       }

     //-------------------- deconv filter
       gn_p = 1.0 - v_sum_p + v_product_p;
```

```
        gn_z = 1.0 - v_sum_z + v_product_z;
        *gn = *gn * gn_p / gn_z;
        hfilt[(*m_filt) * 4 + 0] =(-2.0+2.0 * v_product_p)/gn_p;
        hfilt[(*m_filt) * 4 + 1] =(1.0+v_sum_p + v_product_p)/gn_p;
        hfilt[(*m_filt) * 4 + 2] =(-2.0+2.0 * v_product_z)/gn_z;
        hfilt[(*m_filt) * 4 + 3] =(1.0+v_sum_z + v_product_z)/gn_z;
        (*m_filt) ++;
     }

     return;
  }
  int integ_filt_(
      double *dt,
      double *gn,
      double hfilt[])
  {

  //-------------------- integ_filt filter
     *gn = *dt / 2.0;
     hfilt[0] = 1.0;
     hfilt[1] = 0.0;
     hfilt[2] = -1.0;
     hfilt[3] = 0.0;

     return 0;
  }
  //=====================================================
  //   deconvolution based on poles and zeros
  //        broadband seismogram response -> displacement
  //           (displacement flat)
  //=====================================================
  void deconv_b2d_(
      int *npoles,
      double poler[],
      double polei[],
      int *nzeros,
      double zeror[],
      double zeroi[],
      double *dt_in,
      int    *m_filt,
      double *gn,
      double hfilt[],
      int    *ierror)
  {
     double gn_tmp;

     deconv_b2v_(npoles,poler,polei,nzeros, zeror, zeroi,dt_in,
                m_filt,gn, hfilt,ierror);
     integ_filt_(dt_in,&gn_tmp,&hfilt[*m_filt * 4]);
     *gn *= gn_tmp;
     (*m_filt)++;
}
```

4　まとめ

低周波側の特性が複数の極及び零点によって表されている広帯域地震計の計器特性補正のための再帰型ディジタルフィルターの理論と，そのプログラムを提示した．

謝辞

IRIS (Incorporated Research Institute for Seismology) のサイトから得た極・零点情報を使用した．査読者である気象研究所火山研究部の吉田康宏氏及び編集委員の青木重樹氏からは，当稿の改善に非常に有益な意見を頂いた．

文献

気象庁 (2013): 過小評価判定手法及び想定最大マグニチュードについて，第 9 回津波予測技術に関する勉強会資料 1, https://www.data.jma.go.jp/svd/eqev/data/study-panel/tsunami/benkyokai9/shiryou1.pdf,（参照 2022-11-17）.

小久保一哉 (2013): 火山の短周期成分を含む地殻変動モデルに対する傾斜計の応答, 験震時報, **77**, 1-14.

斎藤 正徳 (1978): 漸化式ディジタルフィルターの自動設計, 物理探鉱, **31**, 240-263.

三谷 政昭 (1987): ディジタルフィルタデザイン, 昭晃堂, 223pp.

Kanamori, H. and L. Rivera (2008): Source inversion of W phase: speeding up seismic tsunami warning, Geophy. J. Int., **178**, 222-238, doi:10.1111/j.1365-246X.2008.03887.XCorpus.

Katsumata, A., M. Tanaka, T. Nishimiya (2021): Rapid estimation of tsunami earthquake magnitudes at local distance, Earth Planets Space, **73**, 72, doi:10.1186/s40623-021-01391-7.

Maeda, T., K. Obara, T. Furumura, and T. Saito (2011): Interference of long-period seismic wavefield observed by dense Hi-net array in Japan. J. Geophys. Res., **116**, B10303,doi:10.1029/2011JB008464.

（編集担当　青木重樹）