Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2335188 A1 2002/08/02

(21) 2 335 188

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2001/02/02

(41) Mise à la disp. pub./Open to Public Insp.: 2002/08/02

(51) Cl.Int.^{7}/Int.Cl.^{7} G11B 20/10

(71) Demandeur/Applicant:
BLESZYNSKI, STANISLAW, CA

(72) Inventeur/Inventor:
BLESZYNSKI, STANISLAW, CA

(54) Titre : METHODE D'INTERPOLATION DE MAXIMUM POUR CAPTEURS DE LECTEURS DE DISQUES OPTIQUES OU MAGNETIQUES

(54) Title: PEAK INTERPOLATION METHOD FOR OPTICAL OR MAGNETIC DISK DRIVE PICKUPS

(57) Abrégé/Abstract:
Presented is a digital signal processing method for accurate detection of position and amplitude of digitized analog signals coming out of optical or magnetic pickup heads The least-square parametric fit of a second-or fourth-degree polynomial curve to 4 or more data points at a time, and minimization of the sum of square residuals allow for precise determination of peak top position and amplitude for the purpose of recording quality diagnostics. The method has been proven very reliable and noise-resistant.

Peak Interpolation Method For Optical or Magnetic Disk Drive Pickups

ABSTRACT

Presented is a digital signal processing method for accurate detection of position and amplitude of digitized analog signals coming out of optical or magnetic pickup heads The least-square parametric fit of a second-or fourth-degree polynomial curve to 4 or more data points at a time, and minimization of the sum of square residuals allow for precise determination of peak top position and amplitude for the purpose of recording quality diagnostics. The method has been proven very reliable and noise-resistant.

Peak Interpolation Method For Optical or Magnetic Disk Drive Pickups

DESCRIPTION OF THE INVENTION

1. Introduction.

This invention has been designed to improve the accuracy of peak timing and amplitude detection of digitized signals in such applications as testing and diagnostics of the quality of optical & magnetic data recording, and many others. The improvement is such that the accuracy beyond digitization quantization limit is possible in both amplitude and timing. The method described in this invention has been tested and implemented in certain commercial magnetic and optical media testers, and has been proven superior to such standard analog techniques as filtering, differentiating and zero-cross detection. This technique is not limited to optical or magnetic recording signal processing, but may also be useful in other digital signal processing applications whenever determination of accurate peak position or amplitude is required.

2. Embodiment of the invention.

General Idea.

The main idea of the method is that we seek to fit a certain well defined theoretical curve of parabolic-like shape to the digitized signal waveform near the position of the supposed peak. The more the theoretical curve shape resembles the actually sampled waveform and the closer we are to the actual peak, the better the fit then becomes. The measure of how good the fit is, is calculated as the sum of square resididuals. The "residual" is defined as a difference between the theoretical curve and the actual data for all data points along the curve.

The algorithm works like this:

a) Take a buffer full of data samples (i=0,1,2,3,...N) .
b) Scan the buffer taking a sub-set of M (M>=4, and M<<N) consecutive data points centered around the index i .
c) Fit a second or higher polynomial (degree K, K>=2) or other parametrized theoretical function, to the above sub-set, using a standard least square method (see ref 1) or other technique.
d) Calculate the square-residual sum SR[i] over the current sub-set centered around index i.
e) Check if the SR has had a minimum in the previous cycle
(that is if SR[i-2]>SR[i-1]<=SR[i]) and if it is lower than certain pre-defined limit SRLIM (SR[i-1]<=SRLIM)
- if yes then assume that there is a peak near or at the the data index (i-1)

- if no then move on (i=i+1) and repeat steps b-e

The above algorithm allows one to find the most viable peaks (both positive and negative), conforming to certain criteria determined mostly by the choice of the theoretical parametrized function (eq. polynomial versus trigonometric or exponential series etc.) as well as by the value of SRLIM parameter. The smaller SRLIM, the better the fit ought to be in order for the algorithm to recognize a given peak as valid.

For every found valid peak location (i), one can now easily calculate the peak top amplitude and the accurate peak top position from the best-fit parameters of the theoretical curve at that point. Since these quantities are obtained through the least-square fit, they are generally very immune to noise, and their accuracy can easily exceed digital quantization limits in both amplitude domain or in time domain (index i).

Note that for the algorithm to work best it is important to chose the data sampling rate high enough and the width of the data sub-set window M small enough such that the typical duration of peaks (at ½ height) is equal or longer than M. This method is also somewhat sensitive to the actual shape of the waveform - which should determine the best choice for the theoretical prametric curve.

Example of an algorithmic implementation (in C language).

Function solve_lsqfit listed below may be used to scan almost entire data buffer (y[i] , i=2,3,...,N-2). If at any point (i) the the function returns 1 - it means that a valid peak was found . One should also test the residual sum of squares returned by the sixth argument to make sure it lower than the limit SRLIM. Peak parameters are returned by the pointer arguments 3-5, such as: peak top amplitude level = *a0, peak polarity is determined by *a2 (negative = maximum, positive=minimum), and the precise timing of the peak extremum is t = i*T - d*T (where T=sampling period). Example of a call:

```
float  a0, a2, d, r=-1.0;
int  y[4] = {1,3,4,2};
if( solve_lsqfit(y, 2, &a0, &a2, &d, &r) )
      printf("Extremum found in y[0..3]  at i=2");
```

```
//*************************************************************
// ca_solve_lsqfit - Solves four-point least-square quadratic fit.
//    Calculates coefficients a0, a2, and d of quadratic polynomial:
//        y(i+k) = f(k) = a0 + a2*(k+d)^2    k=-2,-1,0,1.
// Input:  y    - ptr to array of digitized waveform.
//         i    - array subscript at which an extremum condition is
//                being tested.  Note:  y[i-2],y[i-1],
//                y[i] and y[i+1] must all be valid data
//         *res_sum_sqrs - less than zero value enables
//                calculation of the residual sum of squares, after
//                fitting.  Value >0.0 disables calculations of
//                residual sum of squares.
```

```
// Output: *a0 - Calculated polynomial coefficient. If valid extremum
//              is found (returns 1)then *a0 is equal to the true
//              interpolated peak amplitude.
//         *a2 - Calculated polynomial coefficient. If valid extremum
//              is found (returns 1)then the sign of *a2 indicates
//              whether this is a maximum (*a2 < 0.0 ) or a minimum
//              ( *a2 > 0.0 )
//         *d  - offset parameter. If valid extremum is
//              found (returns 1) then *d is equal to the negative
//              offset between the index i and the true interpolated
//              position of the peak extremum  ( 0 <= *d <= 1 )
//              For example: if *d==0.0  then extremum is at i,
//                           if *d==1.0 then extremum is at i-1,
//         *res_sum_sqrs - residual sum of squares calculated
//              (if enabled) after the least square fitting.
//               If disabled - then no change.
// Returned value:
//         0   - No extremum found between data position (i-1) and i .
//         1   - Extremum was found between, or at (i-1) and i .
//*************************************************************
int solve_lsqfit(int *y, int i, float *a0, float *a2, float *d,
                 float *res_sum_sqrs)
  {
   unsigned char  calc_resid_sumsqrs;
   const float    det = 80.0;
   long   p3, p2, p1;
   int    u0, u1, u3, u2;
   long   a, b, c;
   float  p, q;
   calc_resid_sumsqrs = (*res_sum_sqrs < 0.0);
   u0 = y[i-2];
   u1 = y[i-1];
   u2 = y[i  ];
   u3 = y[i+1];
   p3 =  4L*(long)u0 + (long)u1             + (long)u3 ;
   p2 = -2L*(long)u0 - (long)u1             + (long)u3;
   p1 =     (long)u0 + (long)u1 + (long)u2 + (long)u3 ;
   a  =  20L*(p3     + p2    - p1);
   b  =  20L*p3 + 36L*p2 - 12L*p1;
   c  = -20L*p3 - 12L*p2 + 44L*p1;
   if(a==0L)          // Special case (1)
     {                // the best fit is a straight line
      *a0 = (u1 + u2)/2.0;
      *a2 = 0.0;
      *d  = 0.5;
      if(fabs(b)<1e-5)    // the slope is horizontal --> all equal
      {
       if(calc_resid_sumsqrs)
          *res_sum_sqrs =
          calc_resid_sum(y,i,(float)a/det,(float)b/det,(float)c/det);
       return( 1 );
      }
      else                // the slope is inclined
      {
       return( 0 );
      }
   }
```

```
        p = (-0.5*(float)b/(float)a);
        q = ((float)c - (float)a*p*p)/det;
        if( p<-1.0 || 0.0<p ) // Special case (2)
          {                   // extremum is not within (i-1) and i
           *a0            = (a<0L ? max(u1,u2) : min(u1,u2));
           *a2            =  (float)a/det;
           *d             =  (p>0.0 ? 0.0 : 1.0);
           if(calc_resid_sumsqrs)  *res_sum_sqrs = 1e30;
           return( 0 );
          }
        // Normal return - extremum is found between (i-1) and i
        *a0            =  q;                          // peak value
        *a2            =  (float)a/det;               // coeff. of curvature
        *d             = -p;                          // position of the peak
        if(calc_resid_sumsqrs)   // calculate residual sum of squares
           *res_sum_sqrs =
              calc_resid_sum(y,i,(float)a/det,(float)b/det,(float)c/det);
        return( 1 );
       }

      //*********************************************************
      // calc_resid_sum - calculate residual sum of squares
      //    of polynomial f(k)=a*k^2+b*k+c  to data points
      //    y[k],  k = i-2, i-1, i, i+1
      //*********************************************************
      static float  calc_resid_sum(int *y, int i, float a, float b, float c)
        {
        float   d,e,f,g;
        d = (float)y[i-2] - 4.0*a + 2.0*b - c;
        e = (float)y[i-1] -     a +     b - c;
        f = (float)y[i  ]                 - c;
        g = (float)y[i+1] -     a -     b - c;
        return( d*d + e*e + f*f + g*g );
        }
```

Reference.

1. "Numerical Recipes in C", W.H.Press, W.T.Vetterling, S.A.Teukolsky, B.R.Flannery, 2-nd ed., Cambridge Univ. Press, 1992 (p.671)

Peak Interpolation Method For Optical or Magnetic Disk Drive Pickups

CLAIMS

1. Specific to this invention is the fitting of a parametric theoretical peak-resembling curve to the digitized signal waveform.

2. Specific to this invention is the usage of the least-square technique for fitting a theoretical curve to the digitized signal.

3. Specific to this invention is the peak searching method by performing a repeated least-square fitting of a parametric theoretical curve to the consecutive intervals of the digitized signal and testing for the minimum value of the residual sum of squares.

4. Specific to this invention is the extraction of the accurate peak parameters such as amplitude and position out of the parametric theoretical curve fitted to the digitized signal.