Lawrence Berkeley National Laboratory

Recent Work

Title

Rapid Detection of Spatial Pattern by Fourier Analysis

Permalink

<https://escholarship.org/uc/item/8rj5p6gn>

Journal

Computers in biology and medicine, 20(5)

Authors

Cullander, C.

Baker, J.R.

Budinger, T.F.

Publication Date

1989-11-01

Lawrence Berkeley Laboratory

UNIVERSITY OF CALIFORNIA

Submitted to Computers in Biology and Medicine

**Rapid Detection of Spatial
Pattern by Fourier Analysis**

C. Cullander, J.R. Baker, and T.F. Budinger

November 1989

For Reference

Not to be taken from this room

Donner Laboratory

**Biology &
Medicine
Division**

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

Rapid Detection of Spatial Pattern by Fourier Analysis

Christopher Cullander
Department of Pharmacy
University of California, San Francisco
San Francisco, CA 94143

John R. Baker and Thomas F. Budinger
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
and
Donner Laboratory
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, CA 94720

November 30, 1989

¹This work was supported by the Office of Energy Research, Office of Health and Environmental Research, of the U.S. Department of Energy under contract No DE-AC03-76SF00098.

Abstract

Fast Fourier transform analysis can be used to accurately and rapidly detect pattern in large two-dimensional arrangements of points, such as the locations of cells in culture or plants in a unit square. We present here a sample study of pattern in spatial random point processes. This is evidently the first time that Fourier transform-based cross-correlation techniques have been applied to the analysis of point processes of biological origin. Radial profiles of the power spectra and autocorrelation estimates revealed a nearly constant interpore distance of 0.49 ± 0.04 mm in the locations of eccrine gland pores on the surface of human skin. Additionally, gland-free areas may exist near hair follicles.

Keywords: Spatial pattern, pattern recognition, Fourier analysis, spectral analysis, eccrine gland, point processes

1 Introduction

The occurrence of patterns on a semi-microscopic scale in natural systems can be analyzed using transformation techniques. In our investigations of possible mechanisms of skin electrical activity, we discovered patterns of sweat gland distribution through fast Fourier transform (FFT) analysis. This analysis is based on spatial transforms, and is distinct from contemporary image analysis, e.g. crystallography, as well as the techniques commonly used to analyze spatial point patterns (Diggle [1]).

Let the spatial transform of a wide-sense stationary function $f(x, y)$ be $F(u, v)$, i.e.

$$f(x, y) \Leftrightarrow F(u, v),$$

where $\Leftrightarrow$ denotes Fourier transformation. By wide-sense stationary, we mean that the autocorrelation function is spatially shift invariant. Applying the Wiener-Khinchin theorem, the inverse transform of the power spectrum $|F(u, v)|^2$ is the complex autocorrelation of $f(x, y)$:

$$|F(u, v)|^2 \Leftrightarrow f(x, y) \otimes f^*(-x, -y) = f(x, y) \odot f^*(x, y),$$

where $\otimes$ is the convolution operation and $\odot$ is the correlation operation (Oppenheim and Schaffer [2]).

2 Methods

2.1 Analytical

Analysis of an array of points takes place in three stages: (1) preparation of the location data for transform analysis; (2) forward transformation to determine the power spectrum; and (3) inverse transformation to obtain the autocorrelation estimate. Location information (for example, from a pattern recognition program) is used to create a data matrix in which the (x,y) location of each point is set to one against a null background. To avoid false overlaps in the convolution, the data array is surrounded by a frame of zero samples (zero-filled), with the frame width on each side equal to half the image dimension in that direction and windowed with a circular Hanning window; i.e., a quarter-period cosine rotated about the center. Following forward transformation, the magnitude of the resultant is stored as the power spectrum (PS). The PS is then inverse transformed and the absolute value of the resultant real array is stored as the autocorrelation estimate (ACE). The data, PS and ACE matrices are usually displayed as two-dimensional digital images with pixel brightness proportional to magnitude.

Whereas the FFT has expedited frequency analysis for the past 20 years, the actual implementation on real data of array sizes of 512×512 assumes the availability of computation techniques and machinery not always accessible to biomedical researchers. Thus as part of this research program, we have developed an approach that makes large array transformation practical on microcomputers (Baker and Budinger [3]). This algorithm uses rational pairs of sine and cosine coefficients selected, as shown in the Appendix, from a Farey series of order k , where k is the maximum positive integer representable on the machine (Knuth [4]). When run on an IBM PC AT (8 MHz clock rate), the new algorithm provides 90 percent of the performance of a similar floating point algorithm executed on a VAX 11/780 equipped with a floating point processing unit. Program operation was verified using test images (impulse and square) that had predictable power spectra and autocorrelation estimates.

2.2 Radial profile representation

In an isotropic, diffuse pattern, the distribution functions of both the PS and the ACE matrices are radially symmetric about image zero. This makes it possible to construct a 'mean radial profile' of the two-dimensional image. The values of the mean radial profile represent the frequency space intensities averaged over each successive circumference. The intensities in the image matrix are radially averaged as follows: The distance from (i,j) to the image center is rounded to the nearest integer m (i.e. there is no interpolation), the intensity at (i,j) is added to $\text{bin}(m)$, and $\text{count}(m)$ is incremented. Each such integral radius sweeps out a ring of pixels in the PS that have real radii r such that $m - 0.5 < r < m + 0.5$. The mean intensity at radius m is thus $(\text{total intensity in bin}(m))/(\text{total count}(m))$. The number of useful bins is determined by the largest spatial frequency detectable in the discrete transform $u_c = f_s/2 = 512/2 = 256$, where f_s is the Nyquist sampling frequency.

In the power spectrum, the spatial frequency is $u = n/NT$ (where n is the frequency space index, N the number of samples, and T the spatial sample interval in millimeters), thus the inter-pattern spacing $1/u$ in the original data is NT/n or $512/(\text{bin number} - 1)$. Recall that the first bin contains the DC value, thus $n = \text{bin number} - 1$. For the autocorrelation estimate, a maximum at n indicates that features separated by n in the matrix are highly correlated (Frank [5]). The DC value (which corresponds to the inner product of the data array with itself at zero displacement in the ACE) as well as large near-DC values, should be zeroed before scaling the array for display and computing the radial profile for the analysis.

XBB 889-8921

Figure 1: *Digitized eccrine gland image of human skin sample with 2006 pores.*

Figure 2: Power spectrum radial profile for human skin sample with 2006 pores. Unit pillbox zero-fill and circular Hanning window has been applied to the dataset.

Figure 3: Autocorrelation estimate radial profile for human skin sample with 2006 pores. Unit pillbox zero-fill and circular Hanning window has been applied to the dataset.

3 Results: An example of eccrine gland pattern analysis

Eccrine sweat gland pore images have been analyzed with this technique (Cullander [6]). The imaging medium was a circular paper disc, 10 mm in diameter, which had iodine adsorbed to the starch sizing. When the iodinated paper is exposed to water (e.g. sweat), a sharply defined blue dot is formed in the wetted area. Pilocarpine was iontophoresed into the test area which was then wiped dry, and the paper was pressed onto the skin by a plunger mechanism. After the gland imaging was completed, the unreacted iodine was allowed to evaporate, leaving blue dots on a cream-colored field. Digitized images of gland prints were made using a Hamamatsu C1000 video scanning system, with the camera aperture adjusted for maximum dynamic range. A calibration image of a sheet of graph paper was made at the same time.

The digitized gland images, an example is shown in Figure 1, were then analyzed as described in Section 2.1. The radial profile from the analysis of this image is shown in Figure 2. The maximum between bins 14 and 20 was associated with a pattern of small roughly circular gland-free regions or 'holes' in the field of points with a spacing of 1.48 mm to 2.11 mm; while the peak power (between bins 52 and 63) represents the modal spacing between the points themselves (0.45 mm to 0.55 mm). A similar value for the spacing was obtained from the ACE (Figure 3), and this value was very close to the minimum interpoint spacing. The secondary peaks between the two maxima are harmonics of the first maximum's component frequencies. The association of each peak with a fundamental spatial frequency gave another estimate of the range of the low frequency maximum. A similar estimate could not be made for the second maximum since the harmonics of its components were buried in the high frequency fluctuations.

The conclusion from transform analysis was that there was a minimum separation for the pore openings on the skin surface, and that the modal intergland distance was close to this minimum. This finding is probably a consequence of normal skin growth. With some exceptions (such as the formation of scars or stretch marks), increases in skin surface area are believed to take place by the diffuse addition of new tissue throughout the area rather than by the formation of patches of new skin. The number of eccrine glands does not change after birth, and as the individual grows, gland density decreases as the result of dilution; the skin of a year-old infant has eight to ten times the gland density of an adult (Montagna [7]).

There is a secondary pattern in most of the distributions that consists of small gland-free areas. The pattern probably has an anatomic basis. In cases where the hairs in a photograph of the test area could be counted, the total number of hairs was close to the

number of holes in the gland location data. The spatial distribution of the hairs was also similar to that of the gland free regions.

The assumption of wide-sense stationarity of the point process was verified by repeatedly performing the analysis on regions of varying size and position. In each case, the power spectrum was found to be approximately equal. In general, these point patterns do not exhibit stochastic fractal morphology because the log-power spectrum is not described sufficiently by a linearly decreasing function.

4 Summary

Radial profiling provides a concise graphical representation of the two-dimensional power spectrum and autocorrelation estimates of a wide-sense stationary, isotropic point process, e.g. the eccrine sweat gland locations. From radial profiles, the frequency content and modal spacing of points in the two-dimensional field is easily inferred. Profiles of the power spectra and autocorrelation estimates of eccrine gland distribution found evidence of pattern, in that the pores tend to be about the same distance apart (0.49 ± 0.04 mm) and may also be absent in the immediate neighborhood of hair follicles.

This is evidently the first time that Fourier techniques have been used to make a spectral analysis of biologically related spatial point patterns. In general, these point patterns do not exhibit stochastic fractal morphology. Fourier analysis can provide information about the two-dimensional spatial organization of points accurately and more rapidly than can conventional statistical methods, but is perceived as requiring mainframe computing power. However, the rational arithmetic microcomputer FFT algorithm described has performance characteristics that are comparable to that obtained by minicomputers using floating point arithmetic.

5 Acknowledgements

The authors would like to thank R H Huesman, M S Roos, D C Uber, and S T S Wong for helpful discussions. This work was supported by Public Health Service Fellowship No 5 T32 GM07379-03, by the U.S. Department of Energy under contract No DE-AC03-76SF00098, by the National Institute of Health, National Heart, Lung, and Blood Institute under grants No HL07367 and HL25840, by Cray Research Inc., and by IBM Inc.

References

- [1] Diggle PJ. *Statistical Analysis of Spatial Point Patterns*. Academic Press, New York, NY, 1983.
- [2] Oppenheim AV and RW Schafer. *Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1975.
- [3] Baker JR and TF Budinger. *A new FFT Algorithm for Microcomputers using Rational Arithmetic*. Technical Report LBL-26502, LBL, 1988.
- [4] Knuth DE. *The Art of Computer Programming*. Volume 2, Addison Wesley, Reading, MA, second edition, 1981.
- [5] Frank J. The role of correlation techniques in computer image processing. In Hawkes PW, editor, *Computer Processing of Electron Microscope Images*, chapter 5, pages 187–194, Springer-Verlag, Berlin and Heidelberg, 1980.
- [6] Cullander C. *Local Human Skin Potential - Spatial Distribution, Magnitude and Origin*. Technical Report LBL-23450, LBL, 1987. (Ph.D. Thesis).
- [7] Montagna W. *The Structure and Function of the Skin*. Academic Press, New York, NY, second edition, 1974.

6 Appendix

The basic computation block of a fast Fourier transform (FFT) is commonly called a *butterfly*. Figure 4 is a schematic of a butterfly computation for a radix-2 decimation in time FFT algorithm. The following equations describe the computations implied by the butterfly.

$$\begin{aligned} x_{m+1}(p) &= x_m(p) + W_N^r x_m(q) \\ &= x_m(p) + [\cos(\frac{2\pi r}{N}) + i \sin(\frac{2\pi r}{N})] x_m(q) \end{aligned}$$

$$\begin{aligned} x_{m+1}(q) &= x_m(p) - W_N^r x_m(q) \\ &= x_m(p) - [\cos(\frac{2\pi r}{N}) + i \sin(\frac{2\pi r}{N})] x_m(q) \end{aligned}$$

Since the inputs $x_m(p)$ and $x_m(q)$ are complex, each butterfly operation requires four real multiplications and six real additions. On most microcomputers, floating point arithmetic computations are considerably slower than performing similar operations on integers even if special hardware accelerators are available. There has been considerable effort in implementing FFT algorithms using only integer arithmetic to utilize the performance of integer computational units. However, integer FFT algorithms have two sources of error. The first is truncation due to finite word length and the second is quantization of the coefficients used to approximate sine and cosine (Oppenheim and Schaffer [2]).

We have developed a novel algorithm that uses only integer arithmetic and is insensitive to the effects of coefficient quantization. It uses rational pairs chosen from a Farey series (Knuth [4]) to approximate the sine and cosine values used to perform rotations. These values are precomputed and stored in a lookup table. For signed 16 bit computations, the maximum coefficient quantization error is $2^{-24.1}$ using this method compared to 2^{-16} using signed 16 bit fixed point coefficients. The algorithm also reduces truncation effects by using two sets of sine and cosine lookup tables. When truncation would result from a calculation during the m^{th} step, a flag is set to cause a renormalization during the $(m+1)^{\text{st}}$ step of the algorithm. The renormalization is performed by using a sine and cosine table that have been scaled by one-half. This procedure uses the property that if inputs to a butterfly have modulus 1, the outputs have at most modulus 2. Fixed point methods would lose one bit of precision from this step but the rational pair approximation actually gains one bit of precision.

A register transfer level pseudo-coding of the algorithm is given in section 6.1. On the IBM PC AT, our integer algorithm to compute the butterfly requires approximately

775 clock cycles; whereas, on the same machine, a floating point algorithm using the 80287 requires approximately 1585 clock cycles. Both routines were written in macro assembler. Since the 80287 clock period is 50% longer than the 80286 clock period, the integer algorithm is 3.1 times faster and has speed comparable to a floating point algorithm on a VAX 11/780 with a floating point unit. When implementing the algorithm with rational coefficients, we made the following assumptions.

- Input to the butterfly has modulus $2^{15} - 1$ or modulus $2^{16} - 2$.
- The sine and cosine table selected is unscaled for inputs with modulus $2^{15} - 1$ and is scaled by one-half for inputs with modulus $2^{16} - 2$.
- There is hardware support for
 - signed 16 bit addition and subtraction with carry,
 - multiplication of signed 16 bit operands to form a signed 32 bit product, and
 - division of a signed 32 bit dividend by a signed 16 bit divisor to form a signed 16 bit quotient and a signed 16 bit remainder. Overflow results if the quotient would be larger than 16 bits.

The sine and cosine rational pairs for the FFT algorithm are found by choosing the rational number n_i/d_i generated from

$$\begin{aligned}
 n_0 &= 0 \\
 d_0 &= 1 \\
 n_1 &= 1 \\
 d_1 &= k \\
 n_{i+2} &= \lfloor (d_i + k)/d_{i+1} \rfloor n_{i+1} - n_i \\
 d_{i+2} &= \lfloor (d_i + k)/d_{i+1} \rfloor d_{i+1} - d_i
 \end{aligned}$$

that is closest to the desired sine or cosine values; where k is equal to $2^{15} - 1$. A C language program to compute the coefficients is in section 6.2.

Figure 4: Schematic of radix-2 decimation in time FFT butterfly.

6.1 Rational arithmetic FFT butterfly algorithm

```

/*
** Header:      @(#) fftfar.alg 1.6 89/12/03 19:02:21 baker penguin UCB/LBL
**
** Name:        fftfar
**
** Purpose:
**
**      Radix-2 decimation in time FFT butterfly using rational
**      sine and cosine lookup values from a Farey series.
**
** Input parameters:
**
**      p.real, p.imag      - real and imaginary components of p
**      q.real, q.imag      - real and imaginary components of q
**
**      sin.numer, sin.denom - numerator and denominator of
**                          sine(2Pi r/N)
**      cos.numer, cos.denom - numerator and denominator of
**                          cosine(2Pi r/N)
**
**      norm.in             - input is not normalized
**
** Output parameters:
**
**      p.real, p.imag      - rotated real and imaginary
**                          components of p
**      q.real, q.imag      - rotated real and imaginary
**                          components of q
**
**      norm.out            - output is not normalized
**
*/

/*
** Definitions.

```

```

*/
/*
** Input and output to butterfly.
*/
signed integer      p.real<31:0>, p.imag<31:0>
signed integer      q.real<31:0>, q.imag<31:0>

/*
** Rational sine and cosine values.
*/
signed integer      sin.number<15:0>, sin.denom<15:0>
signed integer      cos.number<15:0>, cos.denom<15:0>

/*
** Normalization flag.
*/
boolean             norm.in<0>, norm.out<0>

/*
** Temporary storage.
*/
signed integer      temp1<31:0>, temp2<31:0>
signed integer      tempa<15:0>, tempb<15:0>
signed integer      temp.real<31:0>, temp.imag<31:0>

/*
** tempa := q.real * cos
*/
/*
** Multiplication by cosine numerator to form two word product.
** Renormalization is handled by using a cosine scaled by one-half.
*/
temp1<31:0> := q.real<15:0> * cos.number<15:0>
temp2<31:0> := q.real<31:16> * cos.number<15:0>
temp1<31:16> := temp1<31:16> + temp2<15:0>

```

```

/*
** Compute quotient and remainder with cosine denominator as divisor.
*/
temp2<15:0> := temp1<31:0> DIV cos.denom<15:0>
temp2<31:16> := temp1<31:0> MOD cos.denom<15:0>

/*
** Compute and add round up. It can be +1, 0, or -1.
*/
temp1<31:0> := 2 * temp2<31:16>
temp1<15:0> := temp1<31:0> DIV cos.denom<15:0>
tempa<15:0> := temp2<15:0> + temp1<15:0>

/*
** tempb := q.imag * sin
*/
temp1<31:0> := q.imag<15:0> * sin.numer<15:0>
temp2<31:0> := q.imag<31:16> * sin.numer<15:0>
temp1<31:16> := temp1<31:16> + temp2<15:0>

temp2<15:0> := temp1<31:0> DIV sin.denom<15:0>
temp2<31:16> := temp1<31:0> MOD sin.denom<15:0>

temp1<31:0> := 2 * temp2<31:16>
temp1<15:0> := temp1<31:0> DIV sin.denom<15:0>
tempb<15:0> := temp2<15:0> + temp1<15:0>

/*
** temp.real := tempa + tempb
*/
temp.real<15:0> := tempa<15:0> + tempb<15:0>

/*
** tempa := q.imag * cos

```

```

*/
    temp1<31:0> := q.imag<15:0> * cos.numer<15:0>
    temp2<31:0> := q.imag<31:16> * cos.numer<15:0>
    temp1<31:16> := temp1<31:16> + temp2<15:0>

    temp2<15:0> := temp1<31:0> DIV cos.denom<15:0>
    temp2<31:16> := temp1<31:0> MOD cos.denom<15:0>

    temp1<31:0> := 2 * temp2<31:16>
    temp1<15:0> := temp1<31:0> DIV cos.denom<15:0>
    tempa<15:0> := temp2<15:0> + temp1<15:0>

/*
** tempb := q.real * sin
*/
    temp1<31:0> := q.real<15:0> * sin.numer<15:0>
    temp2<31:0> := q.real<31:16> * sin.numer<15:0>
    temp1<31:16> := temp1<31:16> + temp2<15:0>

    temp2<15:0> := temp1<31:0> DIV sin.denom<15:0>
    temp2<31:16> := temp1<31:0> MOD sin.denom<15:0>

    temp1<31:0> := 2 * temp2<31:16>
    temp1<15:0> := temp1<31:0> DIV sin.denom<15:0>
    tempb<15:0> := temp2<15:0> + temp1<15:0>

/*
** temp.imag := tempa - tempb
*/
    temp.real<15:0> := tempa<15:0> - tempb<15:0>

/*
** If necessary, renormalize p
*/
    if (norm.in<0>) then

```

```

    temp1<31:0> := p.real<31:0>
    temp1<16>   := temp1<31>
    p.real<15:0> := temp1<31:0> DIV 2
    p.real<31:16> := 0
    temp1<31:0> := p.imag<31:0>
    temp1<16>   := temp1<31>
    p.imag<15:0> := temp1<31:0> DIV 2
    p.imag<31:16> := 0
endif

/*
** q.real := p.real - temp.real
*/
q.real<15:0> := p.real<15:0> - temp.real<15:0>
/*
** Set flag for renormalize next cycle.
*/
if (OVERFLOW) then
    if (BORROW) then
        q.real<31:16> := 1
    else
        q.real<31:16> := -1
    endif
    norm.out<0> := 1
else
    q.real<31:16> := 0
endif

/*
** q.imag := p.imag - temp.imag
*/
q.imag<15:0> := p.imag<15:0> - temp.imag<15:0>
if (OVERFLOW) then
    if (BORROW) then
        q.imag<31:16> := 1
    
```

```
        else
            q.imag<31:16> := -1
        endif
        norm.out<0> := 1
    else
        q.imag<31:16> := 0
    endif

/*
** p.real := p.real + temp.real
*/
p.real<15:0> := p.real<15:0> + temp.real<15:0>
if (OVERFLOW) then
    if (CARRY) then
        p.real<31:16> := -1
    else
        p.real<31:16> := 1
    endif
    norm.out<0> := 1
else
    p.real<31:16> := 0
endif

/*
** p.imag := p.imag + temp.imag
*/
p.imag<15:0> := p.imag<15:0> + temp.imag<15:0>
p.imag<31:16> := 0
if (OVERFLOW) then
    if (CARRY) then
        p.imag<31:16> := -1
    else
        p.imag<31:16> := 1
    endif
    norm.out<0> := 1
```

```
else
    p.imag<31:16> := 0
endif
```

6.2 Rational sine and cosine table generation

```
/*
** Header:  @(#) farsincos.c 1.1 89/11/28 14:52:53 baker penguin UCB/LBL
**
** Name:    farsincos
**
** Purpose:
**          Compute sine and cosine table of length n from a farey sequence
**          of length m.
**
**/
```

```
static char      SccsId[] = "@(#) farsincos.c  1.1 89/11/28 14:52:53
                           baker penguin UCB/LBL";
```

```
#include <stdio.h>
#include <errno.h>
#include <math.h>
```

```
main (argc, argv)
```

```
    int          argc;
    char          *argv[];
```

```
{
    int          i, m, n;

    double       s, c, angle;
    double       ts, tc;
    double       res, tres;
    double       scale;

    int          numer, denom;
    int          tnumer, tdenom;
```



```

FILE                                *sin_file, *cos_file;

printf("Table size: ");
scanf("%d", &n);

printf("Maximum integer: ");
scanf("%d", &m);

printf("Scale factor: ");
scanf("%le", &scale);

sin_file = fopen("sin.dat", "w");
cos_file = fopen("cos.dat", "w");

for (i = 0; i < n-1; i++)
{
    angle = 2.0 * M_PI * i / n;
    s      = scale * sin(angle);
    res    = 1.0;
    for (tdenom = 1; tdenom <= m; tdenom++)
    {
        tnumer = ROUND(s * tdenom);
        ts      = (double) tnumer / (double) tdenom;
        tres    = fabs(s - ts);
        if (tres < res)
        {
            numer = tnumer;
            denom = tdenom;
            res    = tres;
        }
    }
    fprintf(sin_file, "\t\t\t{%10d, %10d},\n", numer, denom);
}

angle = 2.0 * M_PI * (n - 1) / n;

```

```
s      = scale * sin(angle);
res    = 1.0;
for (tdenom = 1; tdenom <= m; tdenom++)
{
    tnumer = ROUND(s * tdenom);
    ts      = (double) tnumer / (double) tdenom;
    tres    = fabs(s - ts);
    if (tres < res)
    {
        numer = tnumer;
        denom = tdenom;
        res    = tres;
    }
}

fprintf(sin_file, "\t\t\t{%10d, %10d}\n", numer, denom);
fclose(sin_file);

for (i = 0; i < n-1; i++)
{
    angle = 2.0 * M_PI * i / n;
    c      = scale * cos(angle);
    res    = 1.0;
    for (tdenom = 1; tdenom <= m; tdenom++)
    {
        tnumer = ROUND(c * tdenom);
        tc      = (double) tnumer / (double) tdenom;
        tres    = fabs(c - tc);
        if (tres < res)
        {
            numer = tnumer;
            denom = tdenom;
            res    = tres;
        }
    }
}

fprintf(cos_file, "\t\t\t{%10d, %10d}\n", numer, denom);
```

```
}
angle = 2.0 * M_PI * (n - 1) / n;
c      = scale * cos(angle);
res     = 1.0;
for (tdenom = 1; tdenom <= m; tdenom++)
{
    tnumer = ROUND(c * tdenom);
    tc      = (double) tnumer / (double) tdenom;
    tres     = fabs(c - tc);
    if (tres < res)
    {
        numer = tnumer;
        denom = tdenom;
        res    = tres;
    }
}
fprintf(cos_file, "\t\t\t{%10d, %10d}\n", numer, denom);
fclose(cos_file);

exit(0);
}
```

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
1 CYCLOTRON ROAD
BERKELEY, CALIFORNIA 94720