

COMPUTATIONS FOR RADIO ASTRONOMY

Michael E. Valdez

TABLE OF CONTENTS

[Introduction](#)

.

CONVERSIONS:

[Convert Noise Figure to Temperature](#)

[Convert Temperature to Noise Figure](#)

[Convert Julian Calendar to Julian Day](#)

[Convert Calendar to Julian Day](#)

[Convert Julian Day to Julian Calendar](#)

[Convert Julian Day to Gregorian Calendar](#)

.

COMPUTATION OF CONSTANTS OR PARAMETERS:

[Compute Sky Background Temperature](#)

[Compute Minimum Detectable Temperature](#)

.

COMPUTATIONS FOR OPERATION:

[Compute Transit Altitude of a Source](#)

[Azimuth and Elevation of a Source every 10 minutes](#)

.

COMPUTATIONS FOR ANTENNA DESIGN:

[Compute The Beam Width of a Dish](#)

[Compute the Focal Distance of a Paraboloid](#)

[Design of an Antenna Feed Horn](#)

[Designing a Dish for Minimal Obstruction](#)

.

[REFERENCES AND READING LIST](#)

INTRODUCTION

COMPUTER PROGRAMS FOR RADIO ASTRONOMY was published by SARA several years ago. The original publication was printed and due to its popularity, it was sold out. This book contains programs written by the author plus some other programs and articles reproduced from RADIO ASTRONOMY, the Journal of the Society of Amateur Radio Astronomers. Many SARA members have submitted comments about that book and most of them have been included in this new book. The large demand for the previous book, is gratefully acknowledged by the author, and is the reason for this second book.

COMPUTATIONS FOR RADIO ASTRONOMY is not a second edition of the book mentioned above, but it constitutes a major revision of the material included in the book, and of the approach for presenting this material. The major characteristics of this book are:

- 1.- The change of name reflects the approach for presenting the material: it is understood the many amateurs to radio astronomy do not have a computer, and for them, a computer program is useless. Consequently, this book presents the computations not only as computer programs but in a form that they can be performed with a hand calculator.
- 2.- The most important difference between this book and the previous one is that each program has a full explanation. It is understood that not all amateurs to radio astronomy have a complete knowledge of all meanings, terms and constants used in this science. This explanation includes definition of terms as well as the use of the values calculated and their limitations. This is the reason for point #3. The explanation not only describes what the program does but how it does it and the meaning of every constant or parameter used. In some cases, when the explanation is either too complex or too long, only a brief description is given and the reader is referred to a text book. These references have been kept to a minimum. Wherever two or more programs exist on the same subject (example: when transformations in both directions are performed), the explanation of principles is copied with both programs, this avoids unnecessary flipping of pages.
- 3.- Only programs written by the author are included in this book, so the complete explanations can be written for all of them.
- 4.- A more common dialect of BASIC has been used, which makes easy translation to all computers. Only one statement is included per line to make the program easier to read. Any command that is not general, but that is needed, is explained. Such commands are kept to a minimum.
- 5.- Probably, the second most important difference between this book and the previous one, is the introduction of the Language C. All the programs are presented in BASIC and in C. The main advantage of C over BASIC is speed. Another advantage of C is, it is available for all computers. The programs presented in this language use the standard version of C, as defined by Kernighan and Ritchie. If the user

has another version of C, the manual generally gives the differences between that particular version and the standard Kernighan and Ritchie C.

6.- The programs are presented in a simplified version, nothing fancy has been used. The program performs its function and gives its result. If a user wishes to add fancy titles and formatted output, it can be added very easily, without any major changes.

7.- The different programs have been organized into several groups: transformations, antennas, and computations.

A companion 3 1/2 inch diskette with all the programs or part of programs in BASIC and C, is available. This diskette can be read by any IBM(c) compatible computer.

The author wishes to acknowledge and thank Mr. Jeff Lichtman, SARA President, for reading an early copy of the manuscript, and revising the final copy. His kind encouragement and criticism is highly appreciated. The author also wishes to thank Vince Caracci, Secretary of SARA, for his continued encouragement.

It is to be mentioned that the author does not derive any economical benefit from the selling of this book. The originals have been donated to the Society of Amateur Radio Astronomers Inc. and all proceedings benefit the Society.

The material in this book is copyrighted, (1990) by the Society of Amateur Radio Astronomers (SARA), Inc. P. O. Box 250462, Montgomery, AL 36125, and buying a copy of this book does not give the buyer any right but to the use of the programs. Under no circumstances should the buyer of this book reproduce any program or part of it for his own profit or for publication. Recall, that the proceeds of the sale of this book benefit SARA and such actions will damage the Society. None of the programs contained in this book should be included in any form of publication without obtaining a written permission from the author. This is the only right that the author retains when donating the originals to SARA.

Michael E. Valdez

Melbourne, Florida, July 1990

Currently in Corydon, IA

Now in Davenport, Iowa.

CONVERT NOISE FIGURE INTO TEMPERATURE

The Noise Figure is defined as

$$F = 1 + T/T_o$$

where F is the noise figure, T is the equivalent or noise temperature at the input of the receiver, and T_o is the ambient temperature, both in degree Kelvin. The ambient temperature is usually taken as 290 degree Kelvin.

The noise figure is usually expressed in db, so

$$NF = 10 \text{ Log } (F) = 10 \text{ Log}(1 + T/T_o)$$

Consequently, given the noise figure in dB, the equivalent noise temperature can be obtained from that formula as

$$\log(1 + T/T_o) = NF/10$$

$$1 + T/T_o = 10^{\text{power of } NF/10}$$

taking natural logarithms

$$\text{Ln}[1 + T/T_o] = \text{Ln}(10) * NF/10$$

now taking the exponential

$$1 + T/T_o = \text{EXP}[\text{Ln}(10) * NF / 10]$$

and

$$T = T_o * \{ \text{EXP}[\text{Ln}(10) * NF / 10] - 1 \}$$

As a BASIC program:

```
1000 PRINT:INPUT "ENTER NOISE FIGURE IN DB. ";NF
1100 IF NF=0 THEN END
1200 T=290*(EXP(NF*0.230258509)-1)
```

```
1300 PRINT "CORRESPONDING TEMPERATURE IS";T;"DEGREE KELVIN."  
1400 GOTO 1000
```

As a C program:

```
#include <osbind.h>  
#include <stdio.h>  
#include <math.h>  
  
main()  
{  
  
int i, j, m;  
char line[70];  
double tr;  
  
tr=1;  
while (tr>0)  
    {  
    printf("\n Enter noise figure in dB ");  
    for (j=0; j<70; j++) line[j]=0;  
    j=0;  
    while ((m=(Cconin() & 255)) != 13) line[j++]=m;  
    tr = atof(line);  
    tr = 290 * (exp(tr * .230258509) - 1);  
    printf(" Temperature: %6.2f degree K\n",tr);  
    }  
}
```

Notice the constant .230258509 that corresponds to $\ln(10)/10$ in both programs.

CONVERT NOISE TEMPERATURE INTO NOISE FIGURE

The Noise Figure is defined as

$$F = 1 + T/T_0$$

where F is the noise figure, T is the equivalent or noise temperature at the input of the receiver, and T₀ is the ambient temperature, both in degree Kelvin. The ambient temperature is usually taken as 290 degree Kelvin.

The noise figure is usually expressed in db, so

$$NF = 10 \text{ Log } (F) = 10 \text{ Log}(1 + T/T_0)$$

The programs for this computation is as follows:

In BASIC:

```
1000 PRINT:INPUT "INPUT TEMPERATURE IN DEGREE KELVIN";D
1100 IF D=0 THEN END
1200 NF=LOG(1+D/290)/0.230258509
1300 PRINT "CORRESPONDING NOISE FIGURE IS"NF"DB."
1400 GOTO 15900
```

In C:

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

```
main()
```

```
{
```

```
int i, j, m;
char line[70];
double nf;
```

```
nf=1;
```

```
while (nf>0)
{
printf("\n  Enter noise figure in dB ");
for (j=0; j<70; j++)  line[j]=0;
j=0;
while ((m=(Cconin() & 255)) != 13)  line[j++]=m;
nf = atof(line);
nf = log(1+nf/290)/0.230258509
printf("  Temperature: %6.2f degree K\n",nf);
}
}
```

Notice that in both programs there is an extra division by .230258509, which corresponds to $\text{LOG}(10)/10$. The reason is that both BASIC and C use only natural logarithms and since the dB are given in Logarithms of base 10, it is necessary to divide the natural logarithm by the natural logarithm of 10. The other 10 comes from the formula.

CONVERT JULIAN CALENDAR TO JULIAN DAY

The Julian day is a continuous number that represents the number of days since noon of the first day of the year 4176 BC. The day is considered starting at Greenwich noon. That date was chosen because it is considered that there was no astronomical observation made before that date, so, all Julian days are positive numbers.

The Julian day is used to indicate precise time because the hours, minutes, seconds, and fractions of a second are represented as fractions of a day. So, the elapsed time between two events can be very easily computed by subtracting the two event times, expressed as Julian days.

Since Julian days are very large numbers, the Julian day starting at noon January 1st, 1987 is 2,446,797, the computation of Julian days must be done in a computer that has sufficient precision. In BASIC or C, the computation should be made in double precision.

The Julian Calendar was established by Julius Caesar and fixed the year at 365 days, the leap year with 366 days, and the duration and names of the months.

The procedure for computing the Julian day from a given Julian calendar date, is relatively simple and it could be done in a single line of code. First, the integer and fractional parts of the days are extracted the date is converted into days by multiplying the months by $275/9$, the number of days in a month, the number of years by 367 and a correction is subtracted. The correction is the number of months + 9 divided by 12, plus the number of years, all divided by 4 and multiplied by 7. Finally, the number of days from the beginning of the julian day (year 4176 BC) to the beginning of the julian calendar is added. This number is 1721027 days. The fraction of days is added and half a day is also added.

The program in BASIC is then:

```

1000 PRINT:INPUT "DESIRED YEAR, MONTH, DAY";Y,M,D
1100 IF Y=0 THEN END
1200 D1=INT(D)
1300 F=D-D1-.5
1400 J=-INT(7*(INT((M+9)/12)+Y)/4)
1500 J=J+INT(275*M/9)+D1
1600 J=J+1721027+367*Y
1700 J=J+F
1800 PRINT "JULIAN DAY IS";J
1900 GOTO 1000

```

The program in C is:

```
#include <osbind.h>
#include <stdio.h>
#include <math.h>
main()
{
int i, s;
int year, month, day, hour, minu;
long y, m;
start:
printf("\n Enter Day, Month, Year, ex: 3 7 86\n Day? ");
day = (Cconin() & 255) - 48;
m = Cconin() & 255;
if ((m > 47) && (m < 58)) day = day * 10 + m - 48;
more:
printf(" Month? ");
month = (Cconin() & 255) - 48;
m = Cconin() & 255;
if ((m > 47) && (m < 58)) month = month * 10 + m - 48;
printf(" Year? 19");
year = 1900;
m = Cconin() & 255;
year = year + (m - 48) * 10;
m = Cconin() & 255;
year = year + m - 48;
printf("\n Enter Hour, Minute as 15:37 ");
hour = (Cconin() & 255) - 48;
m = Cconin() & 255;
if ((m > 47) && (m < 58)) hour = hour * 10 + m - 48;
printf(":");
minu = (Cconin() & 255) - 48;
m = Cconin() & 255;
if ((m > 47) && (m < 58)) minu = minu * 10 + m - 48;
printf("\n \n The %2d of %3s of %4d at %2d:%2d U.T. ",
day,x,year,hour,minu);
y = month + 9;
y = y / 12;
y = y + year;
y = -(7 * y)/4;
m = (275 * month) / 9;
y = 367 * year + y + m + 1721029. + day;
m = (1000000 * minu / 60 + 1000000 * hour ) / 24 - 500000;
```

```
if (m < 0)
{
m = 1000000 + m;
y = y - 1;
}
printf("is Julian day %ld.%ld \n More?",y,m);
i = Cnecin() & 255;
if ((i == 121) || (i == 89)) goto start;
}
```

CONVERT GREGORIAN CALENDAR TO JULIAN DAY

The Julian day is a continuous number that represents the number of days since noon of the first day of the year 4176 BC. The day is considered starting at Greenwich noon. That date was chosen because it is considered that there was no astronomical observation made before that date, so, all Julian days are positive numbers.

The Julian day is used to indicate time very precisely because the hours, minutes, seconds, and fractions of a second are represented as fractions of a day. So, the elapsed time between two events can be very easily computed by subtracting the two times, when they are expressed as Julian days.

Since Julian days are very large numbers, the Julian day starting at noon January 1st, 1987 is 2,446,797, the computation of Julian days must be done in a computer that has sufficient precision. In BASIC or C, the computation should be made in double precision.

The Gregorian calendar is the calendar in current use today and is a modification of the Julian calendar. The Gregorian calendar sets the year at 365 days and a leap year every four years, with 366 days, except in the century years when the century is not divisible by 4, like 1700, 1800, 1900 etc.

The program in BASIC is then:

```

1000 PRINT:INPUT "DESIRED YEAR, MONTH, DAY";Y,M,D
1100 IF Y=0 THEN END
1200 D1=INT(D)
1300 F=D-D1-.5
1400 S=SGN(M-9)
1500 A=ABS(M-9)
1600 J1=INT(Y+S*INT(A/7))
1700 J1=-INT((INT(J1/100)+1)*3/4)
1400 J=-INT(7*(INT((M+9)/12)+Y)/4)
1500 J=J+INT(275*M/9)+D1+J1
1600 J=J+1721027+367*Y+2
1700 J=J+F
1800 PRINT "JULIAN DAY IS";J
1900 GOTO 1000

```

The program in C is:

```

#include <osbind.h>
#include <stdio.h>
#include <math.h>

main()

```

```

{

int i, s;
int year, month, day, hour, minu;
long y, m;

start:
printf("\n Enter Day, Month, Year, ex: 3 7 86\n Day? ");
day = (Cconin() & 255) - 48;
m = Cconin() & 255;
if ((m > 47) && (m < 58)) day = day * 10 + m - 48;
more:
printf(" Month? ");
month = (Cconin() & 255) - 48;
m = Cconin() & 255;
if ((m > 47) && (m < 58)) month = month * 10 + m - 48;
printf(" Year? 19");
year = 1900;
m = Cconin() & 255;
year = year + (m - 48) * 10;
m = Cconin() & 255;
year = year + m - 48;

printf("\n Enter Hour, Minute as 15:37 ");
hour = (Cconin() & 255) - 48;
m = Cconin() & 255;
if ((m > 47) && (m < 58)) hour = hour * 10 + m - 48;
printf(":");
minu = (Cconin() & 255) - 48;
m = Cconin() & 255;
if ((m > 47) && (m < 58)) minu = minu * 10 + m - 48;

printf("\n \n The %2d of %3s of %4d at %2d:%2d U.T. ", day,x,year,hour,minu);

y = month + 9;
y = y / 12;
y = y + year;
y = -(7 * y)/4;
m = month - 9;
if (m < 0) m = -m;
m = m/7;
if ((month - 9.) < 0) m = - m;
m = (m + year)/100;
m = ((m+1)*3)/4;
y = y + day - m + 367 * year;
m = (275 * month) / 9;
y = y + m + 1721029.;

```

```
m = (1000000 * minu / 60 + 1000000 * hour ) / 24 - 500000;  
if (m < 0)  
{  
    m = 1000000 + m;  
    y = y - 1;  
}  
  
printf("is Julian day %ld.%ld \n More?",y,m);  
i = Cnecin() & 255;  
if ((i == 121) || (i == 89)) goto start;  
  
}
```

CONVERT JULIAN DAY INTO JULIAN CALENDAR

The Julian day is a continuous number that represents the number of days since noon of the first day of the year 4176 BC. The day is considered starting at Greenwich noon. That date was chosen because it is considered that there was no astronomical observation made before that date.

The Julian day is used to indicate time very precisely because the hours, minutes, seconds, and fractions of a second are represented as fractions of a day. So, the elapsed time between two events can be very easily computed by subtracting the two times, when they are expressed as Julian days.

Since Julian days are very large numbers, the Julian day starting at noon January 1st, 1987 is 2,446,797, the computation of Julian days must be done in a computer that has sufficient precision. In BASIC or C, the computation should be made in double precision.

The Julian Calendar was established by Julius Caesar and fixed the year at 365 days, the leap year with 366 days, and the duration and names of the months.

The conversion from Julian days to Julian calendar date in years, months and days, follows a simple procedure, as follows:

First, the integer and fractional parts are separated, after the Julian day has been incremented by half a day to compensate for the Julian day starting at noon. After this, the integer part is incremented by 1524 and this gives the constant B. We call C the integer number of years in B and D is the integer number of days in C. E is the difference between B and D, converted into integer number of months. Finally, the number of days is computed from the total number of days in J (B+F), minus the integer number of days in C (D), minus the number of days in the number of months since the beginning of the year (E).

Now, some readjusting is necessary and the month number is E-1, the year number is C-4716. Since E is the month number plus one, it is possible to have the 13th month, or the zero month, so we readjust these possibilities.

The program in BASIC is then:

```

1000 PRINT:INPUT "JULIAN DAY: ";J
1100 IF J=0 THEN END
1200 J=J+0.5
1300 F=J-INT(J)
1400 J=INT(J)
1500 B=J+1524
1600 C=INT((B/365.25)-.3343)
1700 D=INT(365.25*C)
1800 E=INT((B-D)/30.61)
1900 D=B-D-INT(30.61*E)+F
2000 M=E-1
2100 Y=C-4716
2200 IF E>13.5 THEN M=M-12
2300 IF M<2.5 THEN Y=Y+1
2400 PRINT "JULIAN CALENDAR DATE: ";Y;M;D
2500 GOTO 1000

```

And the program in C is:

```

#include <osbind.h>
#include <stdio.h>
#include <math.h>
main()
{
int i, a, c, d, e, year, month;
long j, f, b, day;
char line[80];
start:
printf("Enter Julian Day to Convert: ");
for (i = 0 to 80)
line[i] = 0;
i = 0;
while ((a = (Cconin() & 255)) != 13) line[i++] = a;
j = atof(line);
j = j + 0.5;
a = j;
f = j - a;
b = a + 1524;
c = b / 365.25 - 0.3345;
d = 365.25 * c;
e = (b - d) / 30.61;
a = 30.61 * e;
day = b - d - a + f;
month = e - 1;
year = c - 4716;
if (e > 13.5) month = month - 12;
if (month < 2.5) year = year + 1;
printf("\nJULIAN CALENDAR DATE: Year: %4, Month: %2, Day: %ld", year, month, day);
printf("\n\nMore? ");
i = Cnecin() & 255;
if ((i == 121) || (i == 89)) goto start;
}

```

CONVERT JULIAN DAY INTO GREGORIAN CALENDAR

The Julian day is a continuous number that represents the number of days since noon of the first day of the year 4176 BC. The day is considered starting at Greenwich noon. That date was chosen because it is considered that there was no astronomical observation made before that date.

The Julian day is used to indicate time very precisely because the hours, minutes, seconds, and fractions of a second are represented as fractions of a day. So, the elapsed time between two events can be very easily computed by subtracting the two times, when they are expressed as Julian days.

Since Julian days are very large numbers, the Julian day starting at noon January 1st, 1987 is 2,446,797, the computation of Julian days must be done in a computer that has sufficient precision. In BASIC or C, the computation should be made in double precision.

The Gregorian calendar is the calendar in current use today and is a modification of the Julian calendar. The Gregorian calendar sets the year at 365 days and a leap year every four years, with 366 days, except in the century years when the century is not divisible by 4, like 1700, 1800, 1900 etc.

The program in BASIC is:

```

1000 PRINT:INPUT "JULIAN DAY:";J
1100 IF J=0 THEN END
1200 J=J+0.5
1300 F=J-INT(J)
1400 J=INT(J)
1700 A1=INT((J/36524.25)-51.12264)
1800 A=J+1+A1-INT(A1/4)
1900 B=A+1524
2000 C=INT((B/365.25)-.3343)
2100 D=INT(365.25*C)
2200 E=INT((B-D)/30.61)
2300 D=B-D-INT(30.61*E)+F
2400 M=E-1
2500 Y=C-4716
2600 IF E>13.5 THEN M=M-12
2700 IF M<2.5 THEN Y=Y+1
2800 PRINT "DATE: ";Y;M;D
2900 GOTO 1000

```

And the program in C is:

```

#include <osbind.h>
#include <stdio.h>
#include <math.h>

```

```
main()
```

```
{
```

```

int i, a, c, d, e, year, month;
long j, f, b, day;
char line[80];

start:
printf("Enter Julian Day to Convert: ");
for (i = 0 to 80)
line[i] = 0;
i = 0;
while ((a = (Cconin() & 255)) != 13) line[i++] = a;
j = atof(line);

j = j + 0.5;
a = j;
f = j - a;
a1 = j / 36524.25 - 51.12264;
a = j + 1 + a1;
a1 = a1 / 4;
a = a - a1;
b = a + 1524;
c = b / 365.25 - 0.3345;
d = 365.25 * c;
e = (b - d) / 30.61;
a = 30.61 * e;
day = b - d - a + f;
month = e - 1;
year = c - 4716;
if (e > 13.5) month = month - 12;
if (month < 2.5) year = year + 1;

printf("\nGREGORIAN CALENDAR DATE: Year: %4, Month: %2, Day: %ld", year, month, day);
printf("\n\nMore? ");
i = Cnecin() & 255;
if ((i == 121) || (i == 89)) goto start;
}

```

COMPUTE SKY TEMPERATURE AT ANY FREQUENCY

The purpose of this program is to provide a value of the sky background temperature at a given operating frequency. The sky temperature is formed by several factors: These are non-thermal background radiation, which decreases with frequency, has a value of some 10 degree K at 500 Khz, about 1 degree K at 1.2 Ghz and is negligible above 1 Ghz; the galactic noise that also decreases with frequency and has a value of 110 degree K at 500 Khz and a value of 1 degree K at 5 Ghz; the quantum noise, that increases with frequency starting at about 20 Ghz with a value of 1 degree K and goes up with frequency to 20 degree K at 500 Ghz; and the cosmic noise that is considered constant at 3 degree K.

Naturally, this description neglects many of the influences on the sky temperature, such as the influence of azimuth, the direction with respect to the galactic plane, and the value given is an average. For the purpose of this program, the sky noise has been represented by straight lines in log-log scale, for the values given above. The approximation is good enough for estimation purposes.

The values of 1.2315, -2.63, 2.0177, -2.0414, 0.98857, and -2.765 were obtained by a regression analysis of the data available and the results fit this data with sufficient accuracy.

A listing of the BASIC program is as follows:

```
1000 INPUT "FREQUENCY IN GIGAHERTZ OR 0 TO END";F
1100 IF F=0 THEN END
1100 TN=EXP(1.2315-2.63*LOG(F))
1200 TG=EXP(2.0117-2.0414*LOG(F))
1300 TQ=EXP(0.98857*LOG(F)-2.765)
1400 TC=3
1500 TS=TN+TG+TQ+TC
1600 PRINT "SKY TEMPERATURE IS";TS;"DEGREE KELVIN"
1700 GOTO 1000
```

A listing for the C perform the same function is:

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

```
main( )
```

```
{
```

```
char line[70];
int m, j;
long f, tn, tg, tq, tc, ts;

printf("\n Enter Frequency of Operation or Zero to Exit ");

for (m = 0; m < 80; m++) line[m] = 0;
j = 0;
while ((m = cconin() & 255) != 13) line[j++] = m;
f = atof(line);
tn = exp(1.2315 - 2.63 * log(f));
tg = exp(2.0117 - 2.0414 * log(f));
tq = exp(0.98857 * log(f) - 2.765);
tc = 3;
ts = tn + tg + tq + tc;
printf("The Sky Temperature is %5.1f degree K",ts);

m = Cnecin() & 255;

}
```

PROGRAM MINIMUM DETECTABLE TEMPERATURE

This program makes a theoretical analysis of a radio astronomical station, to obtain the minimum detectable temperature of the system. This is an important figure of merit for the receiving equipment.

The results of this analysis are approximate and should only be used as a guide line, mainly to judge possible improvements and the capabilities of the station. In some cases the results are optimistic and in other are pessimistic but, in general, are as close to reality as the values used in the computations.

The minimum detectable temperature is the temperature of a source that can be detected from the wiggles of the chart recorder. This theoretical value that tells us what can be expected from a telescope.

The minimum detectable temperature does not depend on the aperture of the antenna but on the characteristics of the receiving equipment. In other words, the minimum detectable temperature is a figure of merit of the equipment by itself. The noise temperature of the front end, the length of cable between the antenna and the first amplifier, the integration time, the number of records that are averaged, are factors that influence the minimum detectable temperature.

For a more detailed analysis of these computations, the reader is referred to the book by John Kraus, Radio Astronomy, Chapter 3, page 43. In summary, the idea is like this: the noise output of the receiver is formed by two parts, the noise from the source and the noise introduced by the receiver itself. It is considered that the minimum source temperature that can be detected is that of a source with an equivalent temperature equal to the RMS (root mean square) value of the noise of the receiver.

Now, the RMS value of the noise of the receiver can be computed by dividing the system temperature of the receiver by the square root of the product of the predetection bandwidth of the receiver, the integration time, and the number of records averaged. An efficiency factor less than one should also be introduced, multiplying the system temperature.

The system temperature is formed by three basic factors, the sky or antenna temperature, the line temperature converted to the antenna, and the receiver temperature converted to the antenna.

To convert the receiver input temperature to the antenna, it is divided by the transmission efficiency (gain < 1) of the line. The line temperature is also converted to the input terminals by multiplying by $(1/e - 1)$, where e is the transmission efficiency of the line and the whole factor is the reciprocal of the gain of the line. Note that the gain of the line is less than one because the line produces losses.

The minimum antenna temperature is that produced by the sky at the frequency of operation. So, again, the minimum detectable temperature is a function only of the receiving equipment, and therefore, it is

necessary to consider that the equivalent temperature of a given source depends also on the antenna characteristics.

Then, to compute the minimum detectable temperature it is necessary to have the input temperature of the receiver, which is computed from the noise figure, the receiver bandwidth before the detector, the operating frequency, the length of coaxial cable between the antenna and the pre-amplifier, if any, the integration time, and the number of records averaged.

It should be mentioned that, as a figure of merit, the lowest the minimum detectable temperature, the better the equipment. The minimum detectable temperature can be reduced by reducing the system noise (or temperature), by increasing the integration time, the bandwidth of the receiver, or the number of records averaged.

The system noise or temperature can be reduced by reducing the pre-amplifier input noise, by reducing the ground objects in front of the antenna, by mounting the pre-amplifier at the antenna, eliminating the coaxial cable between the antenna and the pre-amplifier.

Then, in equation form, these computations can be expressed as:

$$T_m = T_s / X$$

where T_m is the figure of merit or minimum detectable temperature, in degree kelvin; T_s is the effective system temperature, in degree kelvin; and X is only a name of the dimensionless factor in the denominator. Then,

$$T_s = k [T_a + T_l + T_r/e]$$

where T_s is the system temperature, in degree kelvin; k is the efficiency factor mentioned above, T_a is the antenna temperature, in degree kelvin; T_l is the equivalent temperature of the line, in degree kelvin; T_r is the receiver input temperature, in degree kelvin; and e the gain of the line, which is less than one, and it does not have dimensions. If there is no line, as when the pre-amplifier is mounted directly at the antenna, e equals one, and T_l equals zero.

$$T_a = 0.7 T_k + 0.3 * 300$$

where T_a is the antenna temperature as above, T_k is the sky temperature, in degree kelvin; and the assumption is made that the antenna is seeing 70% of sky and 30% of ground at 300 degree kelvin.

$$T_l = 300 (1/e - 1)$$

where T_l is the line equivalent temperature, the 300 comes from the assumption that the actual temperature of the line is 300 degree kelvin, and e is the efficiency or gain of the line, which is less than

one. This efficiency is computed from the losses per hundred or thousand feet of line, and the actual length of the line.

$$X = \text{SQR} [Bw It n]$$

where X is only the name of the dimensionless factor in the denominator of the equation, Bw is the pre-detection bandwidth in hertz, It is the integration time, in seconds, and n is the number of records averaged.

As a BASIC program, these computations can be written as follows:

```

1000 INPUT "OPERATING FREQUENCY IN MHZ. ";F
1100 INPUT "RECEIVER BANDWIDTH IN MHZ. ";DV
1200 INPUT "RECEIVER NOISE FIGURE OR TEMP ";TR
1300 INPUT "FT OF COAXIAL CABLE, IF ANY ";LL
1400 INPUT "INTEGRATION TIME IN SEC. ";IT
1500 INPUT "NUMBER OF RECORDS AVERAGED ";NR

2000 IF F>900 THEN TA=EXP(18.788-2.66*LOG(900))
2100 IF F=<900 THEN TA=EXP(18.788-2.66*LOG(F))
2200 LE=1-LL*EXP(-0.3)/100
2300 IF TR>10 THEN 4700
2400 TR=290*(EXP(TR*LOG(10)/10)-1)
2500 TS=TA+300*(1/LE-1)+TR/LE
2600 XL=SQR(DV*1E+06*IT*NR)
2700 TM=TS/XL

3000 PRINT "MINIMUM DETECTABLE TEMPERATURE ";TM;"DEGREE K"
3100 END

```

The program in C is as follows:

```

#include <osbind.h>
#include <stdio.h>
#include <math.h>

main()

{

int i, j, m;
char line[70];
double x, aa, om, od, ad, ts, nm, ns, sm, tm, ds;

```

```

double f, xl, bw, it, rd, st, tl, tr, cl, wl;

start:
printf(" Enter operating frequency in Ghz ");
for (j=0; j<70; j++) line[j]=0;
j=0;
while ((m=(Cconin() & 255)) != 13) line[j++]=m;
f = atof(line);
if (f < 5) st = exp(3.2235 - 2.0638 * log(f));
else if (f < 20) st = exp(log(f) * log(f) / 2.75);
else st = 100 + .048 * f;
st = st / 2.;
if (st < 0.) st = 0.;

printf("\n Enter pre-amplifier noise figure or temperature ");
for (j=0; j<70; j++) line[j]=0;
j=0;
while ((m=(Cconin() & 255)) != 13) line[j++]=m;
tr = atof(line);
if (tr < 10) tr = 290 * (exp(tr * .230258509) - 1);

printf("\n Enter coaxial length between antenna ");
printf("and pre-amplifier, in ft ");
for (j=0; j<70; j++) line[j]=0;
j=0;
while ((m=(Cconin() & 255)) != 13) line[j++]=m;
cl = atof(line);
cl = cl / 100;
x = 2 + 3 * f;
cl = 1 - cl * exp(-x);
tl = 370 * (1 / cl - 1);
tr = tr / cl;

printf("\n Enter receiver bandwidth in Mhz ");
for (j=0; j<70; j++) line[j]=0;
j=0;
while ((m=(Cconin() & 255)) != 13) line[j++]=m;
bw = atof(line);

printf("\n Enter integration time in seconds ");
for (j=0; j<70; j++) line[j]=0;
j=0;
while ((m=(Cconin() & 255)) != 13) line[j++]=m;
it = atof(line);

```

```
printf("\n Enter number of records averaged ");
for (j=0; j<70; j++) line[j]=0;
j=0;
while ((m=(Cconin() & 255)) != 13) line[j++]=m;
rd = atof(line);

ts = st + tl + tr;
ts = .7 * ts + 90.;
xl = sqrt(bw * 1000000 * it * rd);
tm = ts / xl;

printf("Minimum Detectable Temperature:%6.6f degree K\n",tm);

}
```

COMPUTING THE TRANSIT ALTITUDE OF A SOURCE

A source transit is when the source crosses the meridian of the observer in its apparent motion around the Earth. The transit can be North or South of the observer depending on the observers latitude and the source declination.

The observer latitude is the distance from the point of observation to the Earth's Equator, measured along the meridian, and in degrees on the surface of the Earth. The latitude can be North or South, depending on the observer being North or South of the Equator.

The declination of a source is its distance in degrees to the celestial Equator. The declination is called North when the source is North of the Equator and South otherwise.

For the purpose of this computation, latitudes and declination that are North are considered positive, and latitudes and declination that are South are considered negative.

The altitude of a source is the angle between the horizon of the observer and the line of sight to the source. The altitude at transit or the transit altitude of a source is the altitude the source has at the moment of transit. It should be noted that since the apparent motion of the stars around the Earth follow circles around the Pole, the altitude of a source changes continuously, unless the observer is exactly at one of the Poles.

The altitude of a source at transit can be computed by subtracting the latitude of the observer from 90 degrees and adding the declination of the source. If the latitude of the observer is L degree M minutes and the source has a declination D degrees m minutes, the altitude of the source at transit is given by

$$\text{Altitude} = 90 - (L + M/60) + (D + m/60).$$

One point to notice is that when the latitude or the declination are negative, the minutes are normally expressed as positive numbers so, if the declination or the latitude is negative it is necessary to subtract the minutes divided by 60; if the latitude or declination is positive, the minutes divided by 60 are added to the longitude or declination.

A program in BASIC is s follows:

```
1000 INPUT "STATION LATITUDE IN DEGREE, MIN";L,M
1100 Y=L+M/60
1200 IF L<0 THEN Y=L-M/60
```

```

1300 INPUT "SOURCE DECLINATION IN DEGREE, MIN";D,M
1400 X=D+M/60
1500 IF D<0 THEN X=D-M/60
1600 A=90-Y+X
1700 Y$="SOUTH"
1800 IF A>90 THEN 2100
1900 A=180-A
2000 Y$="NORTH"
2100 A1=INT(A)
2200 A2=INT((A-A1)*60+.5)
2300 PRINT "TRANSIT ALTITUDE IS";A1;"DEGREE";A2;"MINUTES ";Y$
2400 INPUT "Signal When Ready";A$

```

Note that lines 1000, 1100, and 1200 could be replaced by statements giving directly the station latitude as degrees and minutes.

A program in C is as follows:

```

#include <osbind.h>
#include <stdio.h>
#include <math.h>

main()
{

int i;
char line[70];
double a, degr, minu, xx, yy;

printf("\n Enter latitude of the station \n Degree: ");
for (i=0; i<70; i++) line[i]=0;
line[0] = 70;
Cconrs(line);
degr = atoi(line+2);

printf("\n Minutes: ");
for (i=0; i<70; i++) line[i]=0;
line[0] = 70;
Cconrs(line);
minu = atoi(line+2);

yy = degr + minu / 60;

```

```
if (degr < 0) yy = degr - minu / 60;

start:
printf("\n Enter declination of the source \n Degree: ");
for (i=0; i<70; i++) line[i]=0;
line[0] = 70;
Cconrs(line);
degr = atoi(line+2);

printf("\n Minutes: ");
for (i=0; i<70; i++) line[i]=0;
line[0] = 70;
Cconrs(line);
minu = atoi(line+2);

xx = degr + minu / 60;
if (degr < 0) xx = degr - minu / 60;

a = 90. - yy + xx;
printf("Altitude of the Source at transit is %ld degree\n",a);
printf("\n More? ");
i = Cnecin() & 255;
if ((i == 121) || (i == 89)) goto start;

}
```

AZIMUTH AND ALTITUDE FOR A SOURCE

There are two basic ways of mounting a telescope, the equatorial and the altazimuth mounts. In the equatorial mount the main axis of the telescope is positioned parallel to the axis of rotation of the Earth and the secondary axis perpendicular to this direction. In this way, rotating the main axis of the telescope only changes the right ascension of the object in view; rotating the secondary axis changes only the declination of the object in view. Since, in their apparent motion about the Earth, the objects in a given direction have only different right ascension, to follow an object is only necessary to rotate the main axis of the telescope.

The altazimuth mount on the other hand, has the main axis of the telescope vertical and the secondary axis horizontal. Rotating the main axis of the telescope changes the azimuth of the object in view; rotating the secondary axis changes the elevation of the object in view. To follow an object in the sky it is necessary to rotate both axes continuously.

The reason why telescopes are often mounted in an altazimuth mount is because this type of mount is easier and cheaper to build. The drawback of having to change both axes to follow a source is not that great as to compensate for the difference in cost.

The purpose of this program is to provide a tool to simplify this following of a source by providing a table of values of azimuth and elevation for the given source, every ten minutes. In general, radio astronomy antennas are not too sharp and a correction every ten minutes is most of the time adequate.

The other problem of azimuth and elevation as astronomical coordinates is that the sources are at different places in consecutive days; consequently, it would be necessary to compute azimuth and elevation for every day that an observation is desired.

This problem is solved here with a change of scales; in this case, the time scale is changed. Instead of using civil, solar or sidereal time, this program uses transit time. That is to say, the times used in this program are minutes and hours before or after transit. With this simple change, only one table is needed for all objects with the same declination.

The basis for this is that all sources are in exactly the same place a given time before or after transit. For example, say one hour before transit, Cass A is always in the same place in the sky, every day, day after day. But the time at which Cass A transits, changes from day to day; so, one hour before transit means different civil times. The position in the sky of Cass A one hour before transit is always the same. The same for any other time and source.

The calculations of the azimuth and elevation of a source at a given time require the use of spherical

trigonometry and the reader is referred to a textbook in astronomy for its explanation.

The formulae are as follows:

$$\sin(\text{elev}) = \sin(\text{lati}) * \sin(\text{decl}) + \cos(\text{lati}) * \cos(\text{dcl}) * \cos(\text{time})$$

where elev, lati and decl are respectively the names given to the variables representing the elevation of the source, the latitude of the observatory and the declination of the source, all measured in radians. Time is the angle in radians corresponding to the time before (negative) or after (positive) transit.

With the value of the $\sin(\text{elev})$ it is possible to compute the $\cos(\text{elev})$ using the formula

$$\cos(\text{elev}) \text{ squared} = 1 \text{ minus } \sin(\text{elev}) \text{ squared}$$

Now, if the cosine squared is positive, the cosine is the squared root of this value and the tangent of the elevation can be computed as the ratio between the sine and the cosine. The elevation can be found by the arc tan function.

If the cosine squared is not positive, the elevation is $-\pi/2$ if the sine is negative and $\pi/2$ if the sine is positive.

Now for the azimuth, using the same nomenclature as before and using the word azim as the name of the variable representing the azimuth,

$$\cos(\text{azim}) = \cos(\text{lati}) * \sin(\text{decl}) - \sin(\text{lati}) * \cos(\text{decl}) * \cos(\text{time})$$

$$\sin(\text{azim}) = - \cos(\text{decl}) * \sin(\text{time})$$

If the cosine is zero, then the azimuth is $\pi/2$ if the sine is positive, or $-\pi/2$ if the sine is negative.

If the cosine is not zero and the sine is zero, the azimuth is zero. If the sine is not zero, the azimuth can be computed from the tangent, which is the relation between sine and cosine.

If the cosine is negative it is necessary to add π to the azimuth as found above. In any case, if the azimuth is negative, twice π is added to make it positive.

The values of the elevation and the azimuth come out in radians and they must be multiplied by 180 and divided into π , to convert them into degrees.

If desired, the fractional part of degrees, can be converted into minutes by multiplying by 60.

This program, in BASIC and then in C, produces a table of the azimuth and altitude for a given source, every ten minutes.

```

1000 PI=3.141592654
1100 INPUT "STATION LATITUDE DEGREES";L
1200 INPUT "MINUTES";M
1300 L=(L+M/60)*180/PI
1400 INPUT "Enter Source Declination Degrees";D
1500 INPUT "Minutes";M
1600 D=(D+M/60)*180/PI
1700 FOR I=0 TO 57
1800 T=(2.5*I-70)*PI/180
1900 S1=SIN(L)*SIN(D)+COS(L)*COS(D)*COS(T)
2000 C1=1-S1*S1
2100 IF C1>0 THEN 2500
2200 ELEV=PI/2
2300 IF S1<0 THEN ELEV=-ELEV
2400 GOTO 2700
2500 C1=SQRT(C1)
2600 ELEV=ATAN(S1/C1)
2700 C2=COS(L)*SIN(D)-SIN(L)*COS(D)*SIN(T)
2800 S2=-COS(D)*SIN(T)
2900 IF C2=0 THEN 3400
3000 IF S2=0 THEN AZIM=0
3100 IF S2<>0 THEN AZIM=ATAN(S2/C2)
3200 IF C2<0 AZIM=AZIM+PI
3300 GOTO 3600
3400 AZIM=PI/2
3500 IF S2<0 THEN AZIM=-AZIM
3600 IF AZIM<0 THEN AZIM=AZIM+2*PI
3700 TIM=(I-28)*10
3800 HOUR=INT(TIM/60)
3900 MIN=ABS(TIM-60*HOUR)
4000 PRINT "AT ";HOUR;" ";MIN;" ELEVATION = ";ELEV;
4100 PRINT " AZIMUTH = ";AZIM
4200 NEXT I
4300 INPUT "SIGNAL WHEN READY";A$

```

If it is desired that the program prints the table, then all the PRINT statements are changed into LPRINT; and in C,

```

#include <osbind.h>
#include <stdio.h>

```

```

#include <math.h>
main()
{
int h, i, s, n;
char c, line[70];
int degr, minu;
long m, count;
double pi, lati, decl, angl, s1, c1, s2, c2, aux, azi, elev;
try:
pi = atof("3.14159265");
printf("\n Enter Station Latitude \n Degree: ");
for (i=0; i<70; i++) line[i]=0;
line[0] = 70;
Cconrs(line);
degr = atoi(line+2);
printf("\n Minutes: ");
for (i=0; i<70; i++) line[i]=0;
line[0] = 70;
Cconrs(line);
minu = atoi(line+2);
lati = degr + minu / 60;
lati = lati * pi / 180.;
start:
printf("\n Enter declination of the source \n Degree: ");
for (i=0; i<70; i++) line[i]=0;
line[0] = 70;
Cconrs(line);
degr = atoi(line+2);
printf("\n Minutes: ");
for (i=0; i<70; i++) line[i]=0;
line[0] = 70;
Cconrs(line);
minu = atoi(line+2);
printf("\n Do you want to print?");
count = 0;
m = Cnecin() & 255;
if ((m == 121) || (m == 89)) count = 1;
decl = degr;
decl = decl + minu / 60.;
decl = decl * pi / 180.;
for (i=0; i<57; i++)
{
    angl = (2.5 * i - 70.) * pi / 180.;

```

```

s1 = sin(lati)*sin(decl) + cos(lati)*cos(decl)*cos(angl);
c1 = 1. - s1 * s1;
if (c1 > 0)
{
c1 = sqrt(c1);
elev = atan(s1/c1);
}
else
{
if (s1 < 0) elev = - pi/2.;
else elev = pi/2.;
}
c2 = cos(lati)*sin(decl) - sin(lati)*cos(decl)*cos(angl);
s2 = - cos(decl)*sin(angl);
if (c2 == 0)
{
if (s2 < 0) azi = - pi/2.;
else azi = pi/2.;
}
else
{
if (s2 == 0) azi = 0;
else azi = atan(s2/c2);
if (c2 < 0) azi = azi + pi;
}
if (azi < 0) azi = azi + 2. * pi;
azi = azi * 180. / pi;
elev = elev * 180. / pi + 0.05;
s = i * 10 - 280;
h = s / 60;
s = s - 60 * h;
if (h != 0) s = abs(s);
if (count != 0)
{
sprintf(line,"%15d%3d%20.1f%20.1f\n",h,s,azi,elev);
n = 0;
while((c=line[n++]) != '\0') Cprnout(c);
Cprnout(13);
}
else printf("%15d%3d%20.1f%20.1f\n",h,s,azi,elev);
}
if (count != 0) Cprnout(12);
printf("\n More? ");

```

```
m = Cnecin() & 255;  
if ((m == 121) || (m == 89)) goto start;  
}
```

PROGRAM FOR COMPUTING THE BEAM OF A DISH ANTENNA

The parabolic dish is probably the most important type of antenna in radio astronomy, due to its narrow beam angle. Another important factor is the effective aperture of the antenna, which can only be estimated.

All the parameters of the antenna depend on factors which are not easy to calculate or measure, such as the diffraction produced by the feed and its supports, the diffraction at the edge of the dish, the reflection from surfaces near or in the field of view of the dish, etc. Since these factors cannot be considered in a calculation, the results of any calculation will be only approximate. But this is no reason for not calculating a value; after all, all calculations are only approximations.

The calculations performed by this program will give a fairly good idea of the expected performance from a dish antenna, they are as follows:

The effective antenna aperture is a factor that directly influences the amount of energy absorbed by the antenna; consequently, an increase in effective aperture increases the amount of noise received from a source. This increase is directly proportional, a doubling of the area, means double the power. In this way, this program can help you decide if it is beneficial to increase the size of the dish, or to compare the dollar value of commercially available dishes vs. building your own.

The antenna gain and directivity are related. The idea of gain is in comparison with a theoretical isotropic antenna. An isotropic antenna is defined as an antenna that receives exactly the same from all directions. A given dish will receive more from certain directions and little from others. The relation between the power received from the preferred direction and with an isotropic antenna is called the gain, and it is usually expressed in decibels.

When we look at the power received from all directions, there is one special direction from where the power is a maximum; we say the antenna is pointed in that direction. Remember, that the antenna always receives from all directions, even from the back by diffraction at the edges. If we look at the power received from a certain other direction, as we go away from the place where the antenna is pointed, there is a certain direction where the power has gone down to half the power in the preferred direction. This is the 3 dB point, or half power point. Naturally, there are many of these points, forming a circle around the point where the antenna is pointed. There are two ways to consider this circle; one, is by taking the relation between the area of the circle on a sphere surrounding the antenna, to the total area of the sphere. This is what is called the solid angle and it is measured in radians squared or degree squared. The other is to measure the plane angle between two points in a diameter of this circle; this is the half power beam angle, and it is measured in degrees.

One point to consider is that if the dish is not perfectly parabolic, the half power points will not form a circle but an ellipse and the half power beam angle will be different in different orientations. Notice that this is not the same as the polarization of the wave, which is influenced by the antenna feeding the dish.

The computation of the antenna aperture is simple, pi times the radius squared is the area of the circle of the opening of the antenna and since the diameter is given in feet, the area comes out in squared feet and must be converted to squared meters; thus the constant $cc = 0.02325625$ which is 0.305 (meters in a foot) squared and divided by 4, because we use the diameter instead of the radius. The factor efficiency is to account for the difference between the actual area of the antenna and the effective area. We consider that 10% of the area is lost because of the feed, the supports, etc. Notice that in the C program, all the constants have been combined into one. Then

$$\text{Effective Area} = \text{efficiency} * \pi * \text{diameter squared} / 4$$

if the diameter is in feet, multiply by 0.305 squared. The efficiency of a dish antenna is on the order of 0.55 .

The half power solid angle of a dish antenna is given by the square of the free air wave length at the operating frequency divided by the effective area of the antenna. The area must be entered in squared meters and the wave length in meters. The angle comes out in radians squared and it can be converted to degrees. Then, for the frequency in megahertz,

$$\text{wave length} = 300 / \text{frequency}$$

$$\text{solid angle} = \text{wave length squared} / \text{effective antenna area}$$

The half power beam angle can be computed by taking the square root of the solid angle. In the program presented below, this value is multiplied by $180/\pi$, to convert it into degrees.

The directivity of the antenna is the relation between 4π , the number of solid radians in the sphere, and the half power solid angle computed above. Then as follows:

$$\text{Directivity} = 4\pi / \text{solid angle}$$

The gain of the antenna is generally converted to dB. This value is not given by the program.

The program in BASIC is then:

```
1000 CC=.02325625
1100 PI=3.141592624
1200 RD=180/PI
```

```

1300 INPUT "ANTENNA DIAMETER IN FT. ";D
1400 INPUT "OPERATING FREQUENCY IN MHZ. ";F
1500 A=.55*PI*D*D*CC
1600 F=300/F
1700 OM=F*F/A
1800 DI=4*PI/OM
1900 OD=SQR(OM)*RD

2000 PRINT
2100 XX=INT(A*100+.5)/100
2200 PRINT "ANTENNA APERTURE ";XX;"SQ. MT."
2300 XX=INT(OM*100+.5)/10
2400 PRINT "SOLID BEAM ANGLE ";XX;"RAD SQ."
2500 XX=INT(OD+.5)
2600 PRINT "EQUIVALENT BEAM ANGLE ";XX;"DEGREES"
2700 XX=INT(DI+.5)
2800 PRINT "ANTENNA DIRECTIVITY ";XX
2900 PRINT
3000 PRINT "Signal When Ready"

```

And the program in C is as follows:

```

#include <osbind.h>
#include <stdio.h>
#include <math.h>

main()
{

int j, m;
char line[70];
double f, aa, om, od, ad;

start:
printf(" Enter operating frequency in Ghz ");
for (j=0; j<70; j++) line[j]=0;
j=0;
while ((m=(Cconin() & 255)) != 13) line[j++]=m;
f = atof(line);
wl = .3 / f;
printf("\n Enter diameter of the antenna in ft ");
for (j=0; j<70; j++) line[j]=0;
j=0;

```

```
while ((m=(Cconin() & 255)) != 13) line[j++]=m;
ad = atof(line);
aa = ad * ad * 0.040183915;
om = wl * wl / aa;
ad = 12.5663705 / om;
om = om * 3282.806406;
od = sqrt(om);
printf(" Antenna aperture:%30.2f sq. mt.\n",aa);
printf(" Solid beam angle:%30.2f deg. sq.\n",om);
printf(" Half power beam angle:%25.2f degrees\n",od);
printf(" Antenna directivity:%27.2f \n",ad);
printf("\n\n More? ");
i = Cnecin() & 255;
if ((i == 121) || (i == 89)) goto start;
}
```

CALCULATING THE FOCAL LENGTH OF A DISH

Most of the antennas used in radio astronomy are of the parabolic reflector type. In reality, the form of the antenna is a solid figure that is called a paraboloid. When a section through its center is taken from a paraboloid, the curve that results is a parabola. In any case, the standard equation for a parabola is as follows:

$$y^2 = 4 f x$$

where f is the focal length of the parabola, x is the depth, which is the distance from the bottom of the parabola along the axis, and y is the distance from the parabola to the axis.

When considering antennas, it is more convenient to measure the diameter and not the radius (y is a radius) and the equation becomes

$$d^2 = 16 f x$$

where d now is the diameter of the antenna at a point that has a distance x from the bottom of the antenna.

From here, it is easy to get the expression for the focal length as

$$\text{focal length} = d^2 / 16 x$$

Then, measure the diameter of the dish at its mouth, using a rigid stick, and the distance from the stick to the bottom of the paraboloid. The formula above produces the focal length in the same unit used for the measurements. It is important to note that both measurements should be made in the same unit.

THE DESIGN OF FEED HORNS

Radio astronomers work with very minute energies. To understand this, consider that the total energy received by the Earth from all the radio sources in the sky is less than the energy of a snow flake when it strikes the ground. On the other hand, your neighborhood TV transmitter might radiate one thousand kilowatts from its tower a few thousand feet away from your antenna.

Amateur radio astronomers work generally, under very limited conditions. Most amateurs work with antennas six feet in diameter. Very few are fortunate enough to work with antennas that are 12 or more feet in diameter. Consequently, amateur radio astronomers must try to get their equipment working at maximum efficiency. One point in favor of amateur radio astronomers is that they have time and abilities at their disposal. By using this time and these abilities they can achieve very high efficiencies for their equipment and thus, obtain interesting results from their observations.

The antenna is a place where care concerning details such as good design and construction will prove beneficial. The first point to consider is that for an antenna to work properly the surface should be close enough to a true parabola. Now, the question is what is close enough. One good rule, that is also used by optical astronomers, is that if the surface does not depart more than $1/10$ of a wave length from the true parabola, it is good enough. At 600 Mhz, this means that the surface should not depart from a true parabola more than 5 cm (2 inch) for the total of errors at any point. At 1.2 Ghz this value is one inch. This is not too difficult to achieve for a small antenna.

The second point about the antenna is that a parabolic antenna is only a reflector; it does not convert any energy. Thus, a parabolic reflector requires some element, mounted at its focus, to convert the energy concentrated by the parabola. This element receives the name of feed. This name comes from communications, where the feed is used to feed power into the antenna, to be transmitted in the desired direction. The feed also works during reception because an antenna is a bi-directional device.

The important point about the feed has to do with directivity. It is easy to see this from the point of view of the feed, that the parabola covers only a small portion of the total space that the feed can see. Ideally, the feed should be designed in such a way as to receive energy mostly from that direction and very little from other directions.

It seems important to analyze this point further so it is clear. Imagine a parabolic reflector and a dipole installed some how at the focal point. A dipole receives energy from all directions but in this case, since the parabola concentrates energy coming from the direction of its axis, the dipole will receive some more energy from this direction than from the others, but the difference cannot be too large.

Imagine now that we put a plate one quarter of a wave length behind the dipole, as to reflect back some of the energy that otherwise will be lost. At the same time, such a plate will block energy coming from that side of the dipole. In this case the gain will increase a little because the plate reflects like a mirror, it

does not concentrate and because the plate itself has to be quite large to work properly, one wave length in side. At 600 Mhz this means that the plate must be 0.5 meters in side, or 0.25 square meters in area. Since the area of a 6 foot dish is only about 2 square meters, the plate covers a substantial portion of the dish.

So we see that a dipole and a plate is a very inefficient way to convert the energy concentrated by the parabola and this is so because this combination does not have enough directivity. Another bad point that is many times overlooked, the plate produces a very large diffraction, sending to the parabola and the dipole energy coming from directions that can be as far as ninety degrees away from the direction of the axis of the parabola. Thus, a broadcast station or communications transmitter located totally outside the view of the parabola can create strong interference to the desired signal because of this diffraction.

The solution is to feed the parabola with another type of collector. The most common type of collector used to feed a parabola is the wave guide or horn type of feed. A wave guide is a device, like a tube with walls that are quite smooth and sufficiently thick, that permits the propagation of electromagnetic energy with very small losses. The wave guide works by reflecting the energy at the walls in a way that the energy propagates in a zigzag path. This means that the wave length inside the wave guide is longer than the wave length outside the wave guide.

When the energy traveling inside the wave guide gets to the mouth of the horn, it finds a discontinuity. It propagates into the free space following a path that depends on the relation between the wave length and the diameter of the horn. Here is where cylindrical wave guides are easier to analyze and this paper is limited to circular feed horns without any taper; that is, to cylindrical horns. Another advantage of the cylindrical feed horn for the amateur, is that it is easier to build with good characteristics.

The pattern of the wave outside the horn can be predicted and is quite directive. The angle from which a feed horn will receive energy depends on the relation between diameter of the horn and wave length. The angle from which the horn must receive energy depends on the parabola. Recall that the parabola will concentrate energy into its focus, so the energy will come from an angle which is the angle at which the feed horn sees the surface of the dish. This angle depends on the diameter of the dish and how far the horn is from the dish, the focal length. In other words, the angle from which the energy will come to the horn depends on the focal ratio of the parabola.

The second point to consider when designing a feed horn is what was said about the wave traveling through the wave guide by bouncing from the walls as it travels. For this to take place, the wave length of the energy inside the wave guide has to be small compared with the diameter of the guide. As the frequency is reduced, the wave length increases and there is a frequency below which the guide does not work as a wave guide any more. This is called the cut off frequency. A feed horn working at a frequency below the cut off is only a dipole enclosed in a box. This being quite an inefficient collector of the energy concentrated by the parabola.

FEED HORN DIAMETER COMPUTATION

The relation between the diameter of a feed horn and the operating frequency, produces a path for the energy coming into the horn. The desired path or angle depends on the focal ratio of the parabola and the diameter of the horn.

The half power angle in radians of a cylindrical horn is given by:

$$\text{HP angle} = 1.02 \lambda / d$$

where λ is the wave length of the signal and d the diameter of the horn, both in the same units. The angle at which the parabola is seen by the feed can be computer considering the triangle formed by the axis of the parabola, a line from the focus to the edge of the parabola an the perpendicular from this point to the axis. Then, calling D the diameter of the parabola and f its focal length, the sine of half the half power angle is

$$\sin(\text{HP}/2) = D / 2f$$

Then, the computation starts with the dimensions of the parabola and obtains the half power angle HP; then, with this value and the wave length at the frequency of operation, the optimal diameter of the cylindrical horn can be computed.

This is for the horn illuminating the parabola in such a way that the side lobes are -18 dB below the maximum. If the diameter of the horn is larger than this value, the energy is concentrated towards the center of the parabola and the spill over is less but the whole area of the parabola is not used. If the diameter is smaller than this value, the spill over increases but the area of the parabola is used more. The value computed above is a compromise.

FEED HORN CUT OFF FREQUENCY

The cut off frequency of a cylindrical wave guide is defined as that frequency below which the wave guide does not work as a guide any more. This frequency depends on the diameter of the horn or guide.

$$\text{Cut Off Frequency in Mhz} = 6900 / D \text{ in inches.}$$

HORN WAVE LENGTH CALCULATION

The wave length of a signal inside a wave guide is longer than the wave length for the same signal in free air.

The equation relating the wave length in free air, the cut off frequency of the wave guide and the wave length inside the wave guide, all in the same units, is as follows:

$$l_g = l_o / \sqrt{1 - (l_o / l_c)^2}$$

LOCATION OF THE PROBE

At this point, the only other factor to consider is, where to put the probe (dipole or monopole) to catch the energy and convert it into electrical energy. The probe should be located one quarter of a wave length inside the wave guide, from the closed end of the feed horn. This position is critical.

If the probe is a pin (monopole) through the wall of the guide, it should be exactly one quarter of the wave length, so it is good to make it adjustable. The impedance of the monopole can be adjusted by a movable element (washer) that is soldered in place after the impedance have been matched. The probe can also be a dipole mounted one quarter of a wave length from the back end of the guide and this dipole should be half wave length long.

Finally, the total length of the cylinder should be at least three quarters of the wave length computed above.

SENSITIVITY TO FREQUENCY

One of the advantages of the parabola is that it works in a large frequency range. The lower limit is given by that frequency at which the feeder obstructs most of the area of the dish. The upper frequency is given by that frequency at which the irregularities are larger than the tolerance (1/10 of the wave length). For its part, the feed horn works at any frequency above the cut off frequency. So, it is possible to design a parabola and feed horn that work over a wide range of frequencies.

There are two factors that limit this frequency response; they are: the length of the probe and the distance from the probe to the back of the horn. These two dimensions should be equal to one quarter wave length. The frequency response of the probe could be extended some what by making the probe with a piece of copper tube, thus reducing its Q.

In any case, such an arrangement does not give too wide a frequency coverage and different probes could be inserted at the appropriate distance to have wide coverage. A better solution is to design different feed horns for the various bands of interest because in this way the obstruction caused by the feed will be minimal for each case.

COMPUTATIONS

This type of computation is better done with a hand calculator but it can also be done with a computer in the calculator mode. The only case where it will be justifiable to develop a program for these computations would be if a large number of trials are planned in the design of an antenna.

COMPUTATIONS TO DESIGN A PARABOLOID ANTENNA

One of the two following situations will be understood to be designing a paraboloid antenna: one, when determining the dimensions for the construction of a paraboloid antenna; two, when the choice to purchase an antenna and/or any other choice is available (as with satellite dishes) and it is necessary to select one of them.

There are several characteristics of an antenna that require analysis before building or buying one, the main ones are: the diameter, the focal length, the steerability, space available, and the precision of the surface.

With respect to the diameter of an antenna, it is easy to realize from the computations presented in other pages, that the larger the better. These computations consider that the diameter of the antenna is an economical consideration and that nobody will be satisfied with a small antenna if he can afford a larger one.

The highest frequency at which the antenna will operate is a function of the surface precision or accuracy. When building the antenna, a high precision is obtained by careful construction, a strong structure that will not deform when moved, and using materials of the proper quality which will last as long as necessary. None of these parameters is amenable to computation.

The problem of the mounting of the antenna is also a matter of cost and affordability. Although it is true that with ingenuity and hard work it is possible to develop a nice mount for the antenna, in general, the quality of the mount reflects directly the amount of money invested in it. Again, it will be considered here that very few persons will be satisfied with a type of mount if they can afford something better and, consequently, it is not necessary to make any computations.

So, the only factor that remains is the focal length of the dish. In both cases, when the antenna is built or bought, this is a complete variable that affects the cost very little and leaves wide flexibility for choosing the focal length of a dish. The analysis that follows, presents a procedure for determining the best focal length for a dish, taking as the only criteria that the dish will have maximum efficiency.

To obtain maximum efficiency it is necessary:

- 1) That there is a minimum area of obstructions in front of the antenna, so most of the area of the dish becomes effective area of the antenna;
- 2) That there is a minimum length of edges in front of the antenna, so that there is minimum diffraction;

3) Finally, the horn feeding the antenna should illuminate the dish as well as possible with a minimum of spill over, so the pick up element in the feeder will receive most of the energy from the direction where the antenna is pointing and little from other directions.

To visualize the problem of obstructions consider that the dish is pointing to the Sun and you look at the surface of the dish. The area illuminated by the Sun is the area that is really working, all the areas with shadows will be lost. Naturally, this assumes that the antenna is uniformly good, without irregularities.

The idea is then to reduce the areas that produce shadows on the dish. The main shadows are the feeder and the supports for the feeder. The shadow produced by the feeder depends on its diameter, actually, it is proportional to the square of the diameter, this is a good point to be considered. The support of the feeder has to be strong so the feeder does not move as the antenna is moved and must always be pointing to the right place. A strong support will also avoid vibrations produced by the wind.

The supports can be of the spider type with three or four legs supporting the horn from the edges of the dish, or the goose neck type that supports the horn from the center of the dish. The goose neck has substantially less area of shadow on the dish, because most of it is in the shadow of the feed, but it is directly in front of the horn, producing diffraction. The goose neck is also more difficult to build. A variation of the goose neck that has interesting characteristics, is the plastic pipe, transparent to microwaves, that supports the horn from the center of the dish and goes straight to the horn. It produces no shadow and little diffraction. It might need some lateral support at the horn end, unless it is solidly supported with a flange or similar device at the dish end.

In any case, all the elements that produce a shadow on the dish are dependent on the construction characteristics, with the exception of the feed horn, and no design is necessary. One final consideration, round supports produce much less diffraction than square or rectangular ones.

So, again, the only element that is left for computation is the feed horn diameter. It happens that the feed horn diameter and the focal length of the paraboloid are related and the procedure to be presented is then an analysis that selects a focal length that produces the minimum diameter for the feed horn. This satisfies the conditions of the problem.

The analysis starts with the feed horn diameter which is desired to make a minimum. The minimum value for the diameter is naturally zero but this is not practical. A small diameter feed horn has a high cut off frequency, and the horn might not work at the desired frequency of operation. So the first step in the calculation is the selection of the lowest frequency of operation that is desired from the antenna.

The cut off frequency of the horn must be lower than this frequency for the horn to work at all. It is considered that the cut off frequency is made equal to 80% of the lowest frequency of operation.

By the way, the highest frequency of operation of a paraboloid antenna depends of the accuracy of the surface and not on the horn. If the dish is not solid, the accuracy of the surface is affected very much by

the holes.

The cut off wave length of a horn, is equal to 1.71 times the diameter of the horn, when both are in the same units; in other words, the diameter of the horn can be found by dividing the wave length of the cut off frequency into 1.71.

With the diameter d of the horn and the wave length λ at the frequency of operation, the illumination angle can be found from

$$\text{HP angle} = 1.02 \lambda / d$$

The focal ratio of the paraboloid can be found from the triangle of rays in the parabola. If the diameter of the parabola is D and the focal length is f , the sine of half the half power angle is given by

$$\sin(\text{HP}/2) = D / 2 f$$

and, rearranging,

$$f = D / 2 \sin(\text{HP}/2)$$

With the focal length of the paraboloid it is possible to use the standard equation of the parabola (d squared equals 16 times depth times focal length) to compute the diameter of the dish at different depths from the bottom of the dish.

An example will be developed in detail, to set the concepts. Our goal is build or buy an antenna that will work above 1 Ghz.

The wave length at 1 Ghz is practically 12 inches. The cut off wave length will be $12/0.8$ or 15 inches. A horn with this cut off frequency has a diameter of $15/1.71$ or 8.8 inches. So, the half power angle of the horn is $\text{HP} = 1.02 * 12 / 8.8 = 1.39$ radians. The focal length of a 12 feet parabola can now be computed as $f = 12 * 12 / 2 * \sin(\text{HP}/2)$ of 112 inches.

Then, under these conditions, a focal length of 112 inches will produce the maximum efficiency by having the smallest diameter of feed horn that produces complete illumination of the dish.

REFERENCES AND READING LIST

- 1989 Compendium from Radio Astronomy, SARA, 1990
- Amateur Radio Astronomers Circuit Cookbook, J. M. Lichtman, 1982
- Amateur Radio Astronomers Handbook, R. M. Sickels, 1988
- Antennas, John D. Kraus, Mc Graw Hill Book Co. 1950
- Antenna Engineering, Walter LeRoy Weeks, Mc Graw Hill Books, 1968
- Antenna Theory and Design, Warren L Stutzman, Wiley
- The ARRL Antenna Book, American Radio Relay League, 1984
- The Big Ear, John D. Kraus, Cygnus-Quasar Books, 1966
- The C Programming Language, Brian W Kernigham and Dennis M Ritchie, Prentice Hall, 1978
- Electromagnetics, John D. Kraus, Mc Graw Hill Book Co. 1984
- Handbook of Antenna Design, A W Rudge, 1982
- Microwave Radio Astronomy, Jeffrey M Lichtman, 1984
- Our Cosmic Universe, John D. Kraus, Cygnus-Quasar Books, 1980
- Radio Astronomy, John D. Kraus, Cygnus-Quasar Books, 1986
- Solar Amateur Radio Astronomy, Jeffrey M Lichtman, 1983
- Training Manual, R. W. Paterson, SARA, 1982