RUN

Fourier Transform Package for North Star

RETURN to continue after reading each part.

INTRODUCTION

This documentation describes the use of the North Star Fourier Transform Package. It is absolutely essential that the single density DOS is used and that a working North Star Floating Point Board (FPB) is installed at locations EFF0-EFFF hex. For double density operation or the FPB at a different address, please contact National Microcomputer Associates for a special, custom version of this package.

WHY NOT USE THE 'FAST FOURIER TRANSFORM'?

Since you are interested in this package, we presume that you are familiar with what the Fourier transform is designed to do: convert a function of time to a function of frequency and vice versa. It is a common and erroneous notion in the data processing world that the so-called 'fast Fourier transform' algorithm is the last word on numerical Fourier transformation. While it certainly has its place, it has many limitations which, in our opinion, disqualifies it from consideration for most applications.

In the first place, the fast Fourier transform requires the data to be in a very special form: equally spaced time or frequency increments and a number of data points equal to an integral power of two. On many occasions, the data will instead be spaced at unequal intervals (perhaps smaller intervals where the variation is fastest) and have any number of points. Of course, interpolation can be used to force the data into the necessary form, but then one is removing oneself a further step from the original data, generally an undesirable direction to go in terms of preserving accuracy.

The next problem is that, believe it or not, the fast Fourier transform assumes that the function it is transforming is periodic, i.e. that it repeats itself indefinitely and never dies down. This can cause a lot of trouble numerically if one is trying to do anything fancy with the function and not just look at its transform.

Finally, the fast Fourier transform algorithm fixes the frequency range transformed to, once the time range of the data is given (and vice versa). In most cases, the resulting range contains too much high-frequency 'garbage' (i.e. is it really data or just round-off?) and not enough resolution in the low frequency region.

WHAT WE DO INSTEAD

A very viable alternative to the fast Fourier transform is what we like to call the speedy Fourier integral transform, or FIT, which transforms data from the time domain to the frequency domain, and the speedy inverse Fourier transform, or IFT, which goes the other way. This transformation assumes that the function is zero outside of the data range and that between any two data points, the function is a straight line. This definition of numerical transform is very close to the rigorous mathematical definition. In fact, if the function to be transformed is actually made up of straight lines, the FIT method will give the EXACT answer within the limits of computer accuracy.

The motivation behind this method is the nature of the data that are to be transformed. In most cases, it is known that the function to be transformed is zero outside of a given range. Furthermore, if the data have been digitized from a photograph using a digitization board, it is very likely that the person doing the digitization has chosen his points in such a way that the function isn't doing anything 'funny' (i.e. is a straight line) between the points he picks. This is the same criterion which would be used to make a digital plot of the function. The rule of thumb, then, is that if the digitized points make a good-looking digital plot, then they will make a good FIT.

THE METHOD IN DETAIL

The FIT method is actually conceptually quite simple. The first step is to pre-process the input data and divide it into groups of equally-spaced points. The algorithm then simply adds the contributions from each resulting trapezoid (known analytical functions) to the total at each frequency (or time). The grouping into segments of equally-spaced points is done to minimize the number of evaluations of sine and cosine which must be done. For each group, two sine and two cosine evaluations are done. Thus, the program will run faster if the number of different equally-spaced groups is kept to a minimum, although every point can have a different spacing if desired.

DATA FILE STRUCTURE

The data file structure required by FIT and IFT was very carefully chosen to allow flexible use in North Star BASIC programs. This is very important since it is usually desirable to do most of the data manipulation in BASIC (like putting in a digital filter function) except for the actual Fourier transformations. The data file format consists of a header containing identification and other information that will be useful for plotting, for instance, followed by the data proper. To be more specific:

TIME DOMAIN DATA

```
     HEADER: TITLE (string, >0 and <200 characters)
             NUMBER OF POINTS (number)
             FIRST TIME (number)
             LAST TIME (number)
             MINIMUM VALUE (number)
             MAXIMUM VALUE (number)
     DATA:   TIME (number)
             VALUE (number)
             (repeat for each point)
```

The following short BASIC program will generate a valid FIT time file:

```
10A$="Title"
20CREATE"FITFILE",1
30OPEN#0,"FITFILE"
40WRITE#0,A$,4,0,3,-1,1
50WRITE#0,0,0
60WRITE#0,1,1
70WRITE#0,2,-1
80WRITE#0,3,0
90CLOSE#0
```

FREQUENCY DOMAIN DATA

```
     HEADER: TITLE (string, >0 and <200 characters)
             NUMBER OF POINTS (number)
             LOW FREQUENCY (number)
             HIGH FREQUENCY (number)
             0 (number)
             MAXIMUM MAGNITUDE (number)
     DATA:   FREQUENCY (number)
             REAL VALUE (number)
             IMAGINARY VALUE (number)
             (repeat for each point)
```

The following short BASIC program will generate a valid IFT frequency file:

```
10A$="Title"
20CREATE"IFTFILE",1
30OPEN#0,"IFTFILE"
40WRITE#0,A$,4,0,3,0,1
50WRITE#0,0,0,0
60WRITE#0,1,.5,.5
70WRITE#0,2,.8,.6
80WRITE#0,3,0,.9
90CLOSE#0
```

MISCELLANEOUS DETAILS

This section of the documentation will print miscellaneous information about the transforms No particular logical order will be used.

The data file formats used above are preserved through the transformations. Thus, a file which had just been created by the FIT transformation could immediately be transformed by IFT. The FIT program automatically adds the string " FIT" to the title, while IFT adds the string " IFT". This is useful for keeping track of what has been done to the data. Note that the original data file is not usually removed or overwritten by any of these programs.

The programs hold in memory all data needed to calculate the new function. The amount of memory needed depends on the size of the file being transformed (the output file can be any size). To calculate the amount of memory used, add the size of the DOS plus the size of the transform program (4K) plus the size of the file to be transformed in bytes, plus 512 bytes to be safe. If the file to be transformed has many different step sizes, add a maximum extra 50% of the file size in bytes. When this is the case (expected to be a rare condition), the programs will print the message "Expanding the data area" many times during the pre-processing. A 24K memory area starting at 2000 hex should be plenty for most applications.

HOW TO RUN THE PROGRAMS

There are four transform programs, two of them for FIT (time to frequency) and two for IFT (frequency to time). The fundamental programs are FIT and IFT (oddly enough!). To run either, simply load the disk containing the programs and use the GO command from DOS. The programs will prompt for all needed information. Note that the usual file name convention with the unit number following a comma is NOT used. Instead, the unit number will be separately requested. All information needed will be input after a prompt.

An important feature of the programs is that the destination file will be created with the proper size if it doesn't already exist. If it does exist and is too small, a message will be printed and a new file name will be requested. If the remaining space on the disk is too small, a message will be printed and the program will wait for a keypress to give the user time to put in a new (initialized) diskette.

At any time during the running of the program, the user can type control-C to interrupt. The program will print the current frequency point being calculated (FIT) or time point (IFT) and wait for another keypress. If another control-C is pressed, the DOS is reentered and all calculations are lost (the new file is invalid). Otherwise, the program goes back to where it was and resumes. Every now and then (time depending on how many points are being transformed), the disk drive will turn on as another buffer of output is stored.

DISK-TO-DISK TRANSFORMS

In many cases, a whole disk of data files needs to be transformed. In this case, two disk drives are used, one for the input data and one for the output. The diskette with the input data should have no other files on it and the diskette to hold the output data should be initially blank. The programs to run are FITDISK (time to frequency) and IFTDISK (frequency to time). The new files will all have the same number of points and the same range. The new file names will be the same as the old except with an extra "F" (FITDISK) or "T" (IFTDISK) if there is room.

These programs are convenient to use overnight or at other times when the computer will be unattended. The control-C interrupt still works the same way. If the diskette becomes full, the program will stop and wait (indefinitely) for a new diskette.

CONCLUSION

The programs above will take about 2 minutes on an 8080 machine (2 MHz) to transform 100 points to 100 points. The exact time depends on the number of groups of equally-spaced points and other factors. The majority of the processor time is spent waiting for the FPB to finish a calculation. The time also varies roughly as the product of the time and frequency point counts.

Please contact us if you would like to obtain the source code or have custom modifications made.

If any other questions remain, please write

National Microcomputer Associates
P.O.Box 5245
Albay, New York 12205

READY