RESEARCH ARTICLE | MAY 01 1989

Digital waveform sampling rate converter FREE

Peg L. Feibig; Leslie I. Brown; Delores M. Etter

Comput. Phys. 3, 38–41 (1989)
https://doi.org/10.1063/1.168321

Digital waveform sampling rate converter

Peg L. Feibig, Leslie I. Brown, and Delores M. Etter
Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, New Mexico 87131

(Received 26 July 1988; accepted 28 November 1988)

The digital waveform sampling rate converter is an efficient, easy-to-use software system that allows a user to perform the conversion between sampling rates efficiently and accurately. The user does not need to be concerned with the technical details of the package. The user simply needs to answer the questions presented by the package, specifying certain input and output files. The software takes care of the conversion. The user can easily verify that a desired file has been selected by choosing the option to plot the signal. This package succeeds in making the mathematical process for sampling rate changes efficient and effortless.

INTRODUCTION

In digital signal processing, it is often necessary to change the sampling rate of a digital waveform. For example, in speech processing, speech parameters are often computed at low sampling rates for low bit-rate storage or transmission. When constructing synthetic speech signals from the low bit-rate representation, however, speech parameters are normally required at much higher sampling rates.[1] Hence, in this particular application, the sampling rate needs to be increased. Applications where the sampling rate needs to be decreased arise when it is desirable to store the fewest number of points possible from a large amount of signal data and still retain all of the signal information. When two signals need to be added, they must have the sampling rate. In this situation, the sampling rates of one or possibly both signals may need to be changed, perhaps by increasing one and decreasing the other until the sampling rates match.

Because the mathematical processes for sampling rate changes are tedious, a computer software system has been developed to perform the sampling rate changes with little technical expertise required on the part of the user. The sampling rate converter consists of two basic parts: (1) the user-interface routines and (2) the routines that manipulate the data. The software system makes use of preexisting data manipulation routines to perform operations while making the user interface simple and straightforward to use.[2]

The goal of the software system is to allow a user to specify a new sampling rate for a digital waveform and to perform the conversion between sampling rates, reconstructing the waveform at the new sampling rate. The interface is simple because the user is not assumed to have technical expertise in sampling rate conversions. This interface incorporates simplicity both in obtaining the necessary information from the user and in conveying the success of the conversion back to the user.

I. DISCUSSION OF THE USER INTERFACE

A. Input/output

Information needed by the system consists, basically, of two parts: (1) the information specified by the user, and (2) the data that need the conversion. The input required from the user consists of the name of the file containing the current data (assumed to be in pairs of x–y coordinates with x representing the time axis), the name of the file that will store the new data, and the new sampling rate. Since the user is not necessarily familiar with the technical aspects of the project, the option exists to enter the new sampling rate in samples per second or the new time between samples in seconds to avoid requiring the user to perform any conversion. The user can also specify a factor change, such as doubling the sampling rate or cutting the sampling time in half. The user can verify that the desired file has been selected by choosing the option to plot the signal before any conversion is performed.

The output produced by the system also consists of two parts: (1) the information presented to the user interactively, and (2) the data that are the converted signal. The converted data can be viewed separately in the form of a time domain plot, together with the original data in two time domain plots, or together with the original data in two frequency domain plots (computed by the fast Fourier transform). The FFT used is a discrete Fourier transform (DFT).[3] In addition to this graphical output, the system produces an output file that contains the numerical data that corresponds to the new sampled points of the signal.

B. User-interaction examples

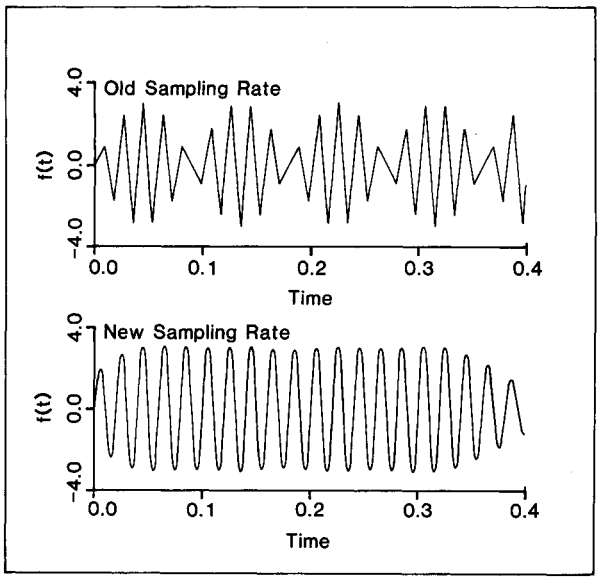
Two examples of the user interaction with the program are illustrated by the series of menus shown below.

Example 1
Initial menu:

```
Main Menu

A. Convert waveform sampling rate
B. Plot waveform
C. Add or subtract two data files
D. Information screen
E. Exit

Enter selection: A
```

Conversion menu 1:

```
Enter the name of the file to be converted:
(15 chars. max., "quit" to return to
main menu)
test1.

Do you wish to interpolate/decimate
a particular set of data
(i.e., a windowed area)(Y/N)?N

The sampling rate is 111.1111 sps

The sampling time is 0.0090 seconds

Enter the name of the output file: test1out
```



Conversion menu 2:

```
New sampling rate information

A. New sampling rate in samples
   per second

B. A factor change in the sampling rate
   (e.g., 2 to double, 0.5 to half
   sampling rate)

C. New sampling time in seconds

D. A factor change in the sampling time
   (e.g., 2 to double, 0.5 to half
   sampling time)

Enter selection: B

Enter the sampling rate change factor: 4
```

Example 2
Initial menu:

```
Main Menu

A. Convert waveform sampling rate
B. Plot waveform
C. Add or subtract two data files
D. Information screen
E. Exit

Enter selection: A
```

Output menu:

```
Output options

A. Plot waveform at new sampling rate

B. Plot two signals at the old and new
   sampling rates (time domain)

C. Plot FFT for both signals
   (frequency domain)

D. Return to main menu

E. Exit

Enter selection: B
```

Conversion menu 1:

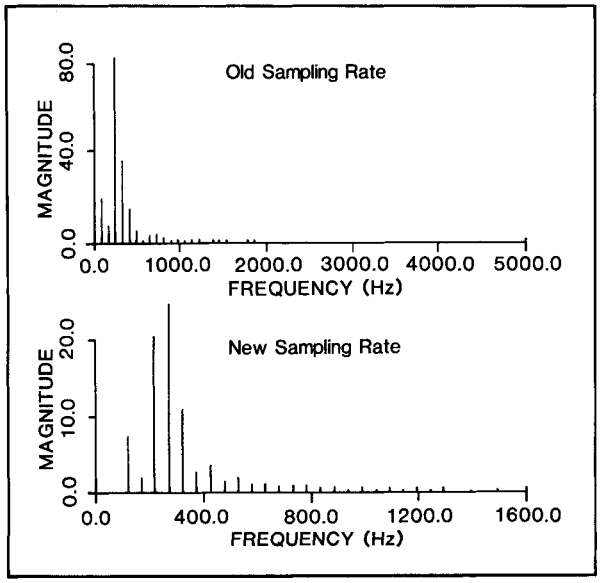
```
Enter the name of the file to be converted:
(15 chars. max., "quit" to return to
main menu)
test3.

Do you wish to interpolate/decimate
a particular set of data
(i.e., a windowed area)(Y/N)?N

The sampling rate is 10000.0000 sps

The sampling time is 0.0001 seconds

Enter the name for the output file: test3out
```

Conversion menu 2:

```
New sampling rate information

A. New sampling rate in samples
   per second

B. A factor change in the sampling rate
   (e.g., 2 to double, 0.5 to half
   sampling rate)

C. New sampling time in seconds

D. A factor change in the sampling time
   (e.g., 2 to double, 0.5 to half
   sampling time)

Enter selection: C

Enter the sampling time in seconds:
0.0003
```

Output menu:

```
Output options

A. Plot waveform at new sampling rate

B. Plot two signals at the old and new
   sampling rates (time domain)

C. Plot FFT for both signals
   (frequency domain)

D. Return to main menu

E. Exit

Enter selection: C
```



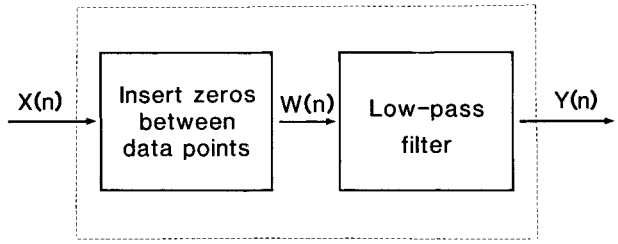


FIG. 1. Block diagram of the interpolation process.

II. SIGNAL PROCESSING TECHNIQUES

The sampling rate conversions performed by this system use three processes: (1) interpolation to increase the sampling rate, (2) decimation to decrease the sampling rate, and (3) filtering to avoid aliasing and to keep the signal within the frequency range of the signal at the new sampling rate. Aliasing is a phenomena that occurs if a signal is sampled too slowly. According to Shannon's sampling theorem, a signal must be sampled at twice the highest frequency component in the signal to avoid aliasing. Aliasing is undesirable because erroneous results are obtained; higher frequency components appear as lower frequency components.

For integer changes in the sampling rate, such as multiplying the sampling rate by 3 or dividing the sampling rate by 5, the appropriate combination of interpolation and filtering, or decimation and filtering, is used. Otherwise, for noninteger changes in the sampling rate, such as increasing the sampling rate to $\frac{3}{2}$ or decreasing the sampling rate to $\frac{2}{5}$, a combination of interpolation, decimation, and filtering is necessary.

For increased sampling rates, aliasing is not a problem and, therefore, the signal points can be processed using interpolation. The interpolation algorithm uses a zero-fill technique which requires a low-pass filter to perform the interpolation between data points.[2] Figure 1 illustrates this process.

For decreased sampling rates, aliasing can be a problem. The signal points must be low-pass filtered first to ensure that the frequencies contained in the signal are no higher than the frequency which introduces aliasing at the new sampling rate. The filtered points are then processed using decimation. Thus, for integer decreases, a sequence of low-pass filtering combined with decimation converts the sampling rate.[2] Figure 2 illustrates this process.

When the desired change in the sampling rate is by a noninteger amount, the process is more complicated. If the total frequency change desired is L/M (where L and M

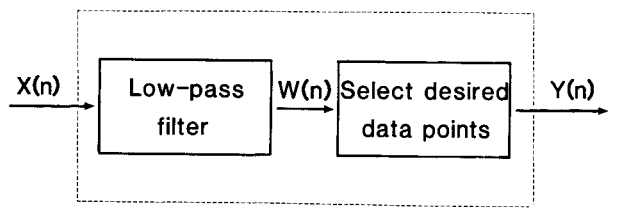


FIG. 2. Block diagram of the decimation process.

are integers), the sampling rate is first increased by a factor of L using the interpolation/filtering scheme, and then the sampling rate of the interpolated data is decreased by a factor of M using the filtering/decimation scheme. Thus, for noninteger changes all three processes (interpolation, filtering, and decimation) are incorporated for the sampling rate conversion. All of these steps are handled by the program and are transparent to the user.

III. SPECIAL DESIGN CONSIDERATIONS

For both interpolation and decimation low-pass filtering is required. Because of its characteristics of sharp rolloff with relatively low order, an infinite impulse response (IIR) Butterworth filter was used. Use of a Butterworth filter minimizes distortion of the input signal. An IIR filter rather than a FIR (finite impulse response) filter was chosen after carefully weighing the advantages and disadvantages of both types of filters.

Another consideration in the design of the sampling rate converter is the quantity of data in the input file. The signal processing algorithms normally require the data to be in an array and, because FORTRAN 77 does not allow dynamic data allocation, array sizes must be declared at compile time. Although it is relatively easy to modify FORTRAN routines to increase the sizes of arrays, this requires recompilation of the entire program. A designer cannot assume that a system has unlimited memory available. To increase the ability of the package to accept data files of any size, the data are processed in sections. The data are separated into blocks of 1024 points; this is a reasonable number to work with and it is easily increased, if necessary. A power of two was chosen, in part, to accommodate the fast Fourier transform routine. Using sections of the data does not cause problems with the IIR filter, which requires feedback or reuse of the signal points, because the routine to perform the filtering saves the previous values required for the overlap computations.

IV. SPECIAL ALGORITHMS

When a user wishes to change the sampling rate of a signal by a noninteger factor, the sampling rate converter converts the factor into a rational number, L/M, then interpolates by a factor of L and decimates by a factor of M. It is not always a direct step to determine L and M, so a special algorithm is used that utilizes Farey's numbers.

The algorithm uses an iterative, or "cut and try," approach to find an approximation to the number R which is a Farey fraction. Farey fractions are used because one of their properties is that, given any real number R, there is always a "nearby" Farey fraction L/M for which the error is bounded to a reasonable value.[4] The definition of Farey fractions and their known properties allows straightforward implementation of the algorithm, and produces accurate results.

When using an IIR filter, the phase shift introduced by the filter needs to be taken into account. Since IIR filters introduce a nonlinear phase, corrective measures must be taken so that no phase shift is introduced into the signal. In order to achieve zero phase response, a technique of forward and then reverse filtering is used.[5] The signal is first passed through the filter as it occurs in real time (forward filtering). The output signal is then reversed in time and becomes the next input signal. This reversed signal is then passed through the filter (reverse filtering). The output from this pass is then reversed in time to yield the final result, which has zero phase shift.

V. SOFTWARE/HARDWARE ENVIRONMENT

The digital waveform sampling rate converter is implemented on a Digital Equipment Corporation (DEC) VAX 8650 computer using DEC VT-240 terminals that have graphics capabilities. The operating system for the VAX 8650 is DEC's VMS version 4.5, and the sampling rate converter runs as a single process under this operating system.

The system software is written in FORTRAN 77 using only standard features of the language to ensure portability. The graphics routines are written with Dissplа Graphics version 9.0. Dissplа is a library of FORTRAN subroutines for presenting information in a graphical form and is used to implement the plotting routines. The version of Dissplа used is a special version designed for the VMS operating system. The system was designed in a modular manner. Modularity makes the system easy to understand, simplifies implementation and testing, and facilitates modifications.

VI. SUMMARY

The digital waveform sampling rate converter is a software system that allows a user to easily perform sampling rate conversions. The package is designed so that both the technical and the nontechnical user can perform powerful signal processing techniques. The software package utilizes user-interactive menus that make the mathematical process of the actual conversion transparent to the user.

ACKNOWLEDGMENTS

This work was funded in part by Sandia National Laboratories and initially developed as a part of a class project.

1. R. W. Schafer and L. R. Rabiner, Proc. IEEE **61**, 692 (June 1973).
2. S. D. Stearns and R. A. David, *Signal Processing Algorithms* (Prentice-Hall, Englewood Cliffs, NJ, 1988).
3. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1975).
4. M. R. Schroeder, *Number Theory in Science and Communication* (Springer, New York, 1986).
5. D. M. Etter and R. H. Johnston, "A Versatile Software Package for IIR Digital Filters," Sandia National Laboratories Technical Report (1982).