Sample Rate Converter

LaGrange Interpolation

28/JAN/2004

Table of Contents

ABSTRACT

Sample rate conversion is converting a digital audio signal density to a different digital audio signal density with minimal change in the actual data. This process is performed because of bandwidth, storage, or processor limitations. Sometimes digital audio data does not utilize the bandwidth of a higher sample rate such as 44,100 Hz, so it may be stored in a lower sample rate such as 8,000 Hz. When the digital audio data is played back, the data must be converted back to the original signal, which has a sample rate of 44,100 Hz. Sample rate conversion performs this transformation without altering the actual signal that is stored.

This paper discusses the sample rate conversion, the theory of LaGrange Interpolation, and a possible method to implement a sample rate conversion based on LaGrange Interpolation in both C and assembly programming languages for the Texas Instruments TMS32055X DSP.

1. Introduction to Sample Rate Conversion

In digital systems, the digital signal consists of discrete data samples taken at periodic time intervals from a continuous analog source. These individual data samples make up the digital signal. The number of samples in a set period of time determines the accuracy of the digital signal that represents the original analog source. The more samples in a period of time allows for a digital signal to more closely resemble the original analog source. The Sample Rate is the number of samples from a signal taken during a set period of time. For example, a 32,000 Hz sample rate has 32,000 samples per second. The higher the sample rate, the digital signal is better able to represent the original continuous analog signal. However, sometimes data must be sampled at a lower frequency due to storage, processor, or bandwidth restrictions. These restrictions result in a digital signal that has a lower sampling rate than what is desired. When the desired signal sample rate is higher than the source signal sample rate, the source signal must be altered so that it is a higher sample rate. For this alteration to happen, new samples must be created to fill in the "holes" that exist in the signal with the lower sample rate. For example, a 32,000 Hz sample rate being converted to 44,100 Hz requires 12,100 new samples to be inserted. So where do we get the 12,100 other samples?

Since the analog signal is not available, it is not possible to resample the digital signal. To obtain the new samples, the original samples must be used to interpolate new samples. Interpolation is computation of a new value from the existing data. The simplest example is similar to finding the mid-point between two data points and placing a new point in between the two points that is the average of the two points. However, this method is not very accurate due to the fact that most audio signals are not linear. Therefore, other methods of interpolating new samples must be pursued. One method of interpolation that will be discussed in this paper is LaGrange Interpolation. LaGrange Interpolation is similar to the linear interpolation, but it utilizes more than just two points. The more points used, the more accurate the interpolation becomes. However, if too many points were used, the computation required could potentially become enormous.

This paper will discuss the theory of LaGrange Interpolation, tests performed on a Sample Rate Converter based on LaGrange Interpolation, and a possible implementation of the Fixed Sample Rate Converter on a Texas Instruments TMS32055x Digital Signal Processor in both C and in Assembly code.

2. Sample Rate Conversion Theory

There are various methods for obtaining new samples for a sample rate conversion. Some of the earlier models implemented digital-to-analog conversion followed by an analog-to-digital conversion. Basically, this method reconstructs the analog signal from the digital signal and then re-samples the analog signal at a new sample rate. This method for sample rate conversion was rather inefficient since the D/A converter and the A/D converters would introduce much noise. This method also increased the cost of sample rate conversion drastically since the additional D/A and A/D converters were required to perform this task.

Moving this process to a completely digital reduces the cost and the noise introduced by the various analog filters, A/D converters, and D/A converters. The cost of performing this conversion is the requirements for processor power and memory. As Digital Signal Processors become more advanced and less expensive to manufacture, the cost for processor power and memory goes down. As a result, the digital process of Sample Rate Conversion may be more practical in a Digital System.

There are many methods for digital sample rate conversion. Some methods require much more memory than others do, while other require more processor resources. Usually there is a trade off between memory and processor for each method. The method of sample rate conversion implemented usually is determined by each application. In this case, the application required less memory usage. Therefore LaGrange Interpolation was used since it did not require a large table to determine a new sample. However, new coefficients must be calculated to determine new samples. Therefore LaGrange Interpolation method is more processor intensive than other sample rate converters. Since new coefficients must be calculated each time a new sample is calculated.

The number of new samples created is dependent on the output sample rate desired. If the output sample rate is higher than the input sample rate, more output samples than input samples will be created. On the other hand, if the output sample rate is lower than the input sample rate, then less new output samples will be created. The resulting output samples when played back at the correct sample rate should sound close to the input samples when played back at the correct sample rate. The duration of the samples should be the same as the duration of the input samples since there should be no change in pitch in a sample rate conversion.

3. LaGrange Interpolation Algorithm

LaGrange Interpolation is a Finite Impulse Response filter that takes in equal future and past samples to generate a new sample. LaGrange Interpolation is a polynomial interpolation method that requires N+1 samples to achieve an Nth Order computation.

Example of 4th Order Interpolation

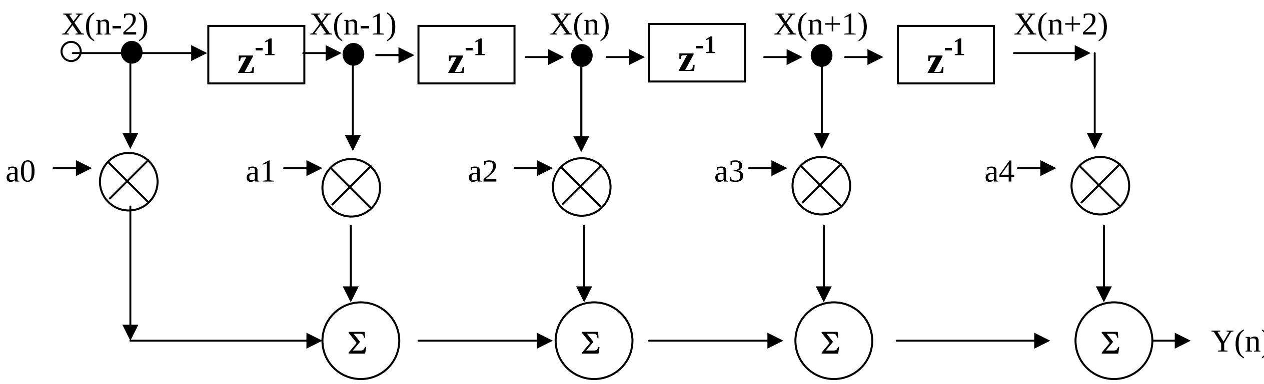

Figure 1: Block Diagram of 4th Order LaGrange Interpolation

In the LaGrange Interpolation, the new interpolated sample is created from the weighted average of the original samples. The weights of the weighted average are obtained by using calculating coefficients based on the location the new sample is going to be located in relation to the original samples. If a new sample is closer to sample A than sample B, then the coefficient multiplied with Sample A is larger than the coefficient multiplied with sample B. The resulting products of the coefficients and the samples are added together to yield a new sample.

- **Interpolation:**

Coefficients must be generated to interpolate new samples from the input samples. The distance the new sample is from the input sample must be determined to generate these unique coefficients. To determine how much to increment the distance, a ratio must be defined. This ratio is the (Input Sample Rate/ Output Sample Rate). For example a 32 kHz input sample rate conversion to 44.1 kHz would result in a ratio of 32/44.1 = .7256. This ratio determines how much of an input sample is consumed to create a new sample. For every new 44.1 kHz sample created, .7256 of the old 32 kHz sample is consumed. The ratio is essential in determining when to retrieve a new input sample, and the determination of the distance the new sample is from the input sample. Whenever a new sample is created, the ratio is added to a “Distance Counter”, the “Distance Counter” stores the decimal portion of the results and whenever the “Distance Counter” exceeds 1, then 1 is subtracted and a new input sample is added. This "Distance Counter" determines when a new input sample is added and the distance counter is also used to calculate the coefficients.

(Example of 1st five values for "Distance Counter" in 32kHz to 44.1kHz Sample Rate Converter)
Ratio = .7256
"Distance Counter" = 0 for 1st Output sample
"Distance Counter" = 0.7256 for 2nd Output sample
"Distance Counter" = 0.4512 for 3rd Output sample (*add new input sample*)
"Distance Counter" = 0.1768 for 4th Output sample (*add new input sample*)
"Distance Counter" = 0.9024 for 5th Output sample

- **Decimation:**

Coefficients must be generated to decimate new samples from the input samples. The ratio (Input frequency/Output frequency) in case of decimation has two parts integer part and fractional part. The integer part is used to determine the no of input samples to be consumed to generate one output sample. The fractional part is "Diastance Counter" which is used to interpolate the output samples from input samples in the same way as Interpolation does.

4. Interpolation Filter Design

The Interpolation Filter is designed in such a way that the coefficients can shape the output signal with minimal loss of Information.
Disatance Counter is the crucial factor in case of determining the coefficients.

Lagrange interpolation is a well known, classical technique for interpolation . More generically, the term *polynomial interpolation* normally refers to Lagrange interpolation. In the first-order case, it reduces to linear interpolation.

The basic equation used to generate the coefficients :-

$$h(n) = \prod_{k=0, k\neq n}^{N} \frac{(Order/2 - dist.Counter - k)}{(n-k)}$$

The above equation is the basic equation which can be used to generate the lagrange coefficients , using which we can generate the output samples from input samples . From the numerator of the above definition, we see that is an order N polynomial having zeros at all of the samples except the nth. The denominator is simply the constant which normalizes its value .

$$h(n,k) \cong \delta(n,k) = 1 \text{ , } k = n$$

$$= 0 \text{ , } k \neq n$$

In other words, the polynomial **h(n)** is the n th *basis polynomial* for constructing a polynomial interpolation of order **" k"** over the **"k+1"** sample points.

In the case of an *infinite* number of *equally spaced* samples the Lagrangian basis polynomials converge to shifts of the *sinc function*, i.e.,

h(n,k) = sinc(pi*(distance.Counter – k))

let (distance.Counter – k)=DC

h(n,k) = sinc(pi*DC)

The sinc function

$$\mathrm{sinc}(t) \triangleq \frac{\sin(\pi t)}{\pi t}$$

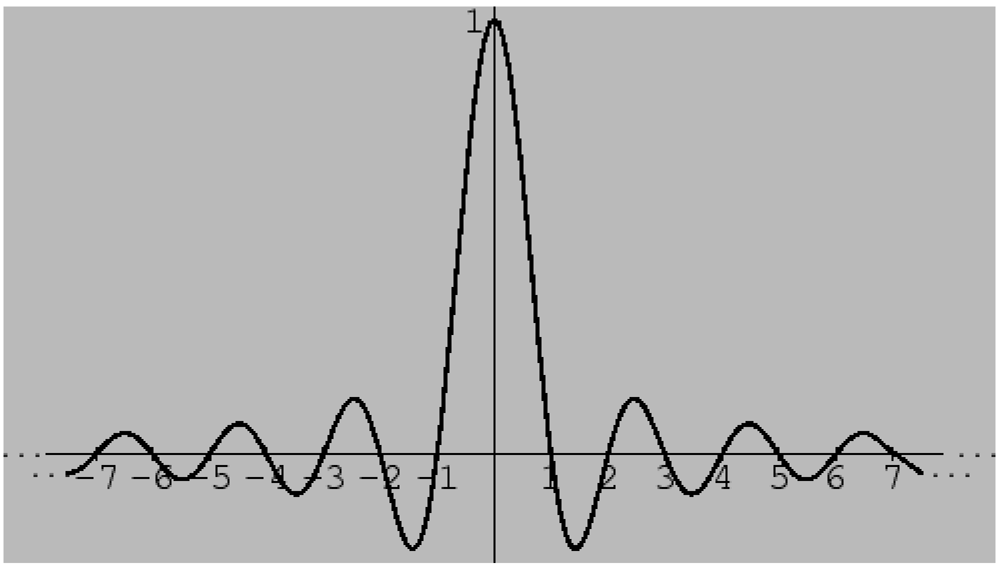

Figure 2: Sinc Function

A simple argument is based on the fact that any analytic function is determined by its zeros and its value at one point. Since **sinc(0)=1** , it must coincide with the infinite-order Lagrangian basis polynomial for the sample at **DC = 0 ,** which also has its zeros on the nonzero integers and equals 1 at **DC = 0**.

The equivalence of sinc interpolation to Lagrange interpolation was apparently first published by the mathematician Borel in 1899, and has been rediscovered many times since.

In our case we establish the equvalence using a windowed-sinc function where the window function we chose is the "Hanning Window".

Important features of Hanning window function is as follows :

Transition width (Hz)(normalized)	Passband ripple (dB)	Main lobe Relative to side lobe (dB)	Stopband Attenuation (dB)(maximum)	Window function W(n), | n | ≤ (N- 1)/2
3.1/N	0.0546	31	44	0.5 + 0.5 Cos(2πn/N)

The final equation is as follows:-

$$h(n) = \prod_{k=-N/2, k \neq n}^{N/2} \frac{Sinc(pi * (DC - k)) * (0.5 + 0.5\{Cos(2 * pi * (DC - k))\})}{N}$$

Following is the matlab equation used to generate the filter coefficients :-

Sinc function multiplied with *hanning Window .

```
N=44
Length = 8;
dl = 2*Length+1;
a=-Length: Length;
c=zeros(N,dl);

for j=1:N+1
        d = -(j-1)/N;
        for i = 1:dl
                if((d + a(i)) !=0)

        C(j,i) =32767 * Sinc(pi*(d+a(i)))*(0.5+0.5(Cos(2*pi*(d+a(i))))/(dl-1)

        Else
                C(j,i) = 32767 ;
        End
        Intc(j,i) = round(c(j,i));
   End
```

End;

Here the filter coefficients are generated for distance counter values 1/44,2/44,3/44,4/44…

We have observed that the above-specified Distance Counter values hold good result for any to any conversion within (8,11.025,12,16,22.05,24,32,44.1,48) this set.

5. AntiAliasing Filter

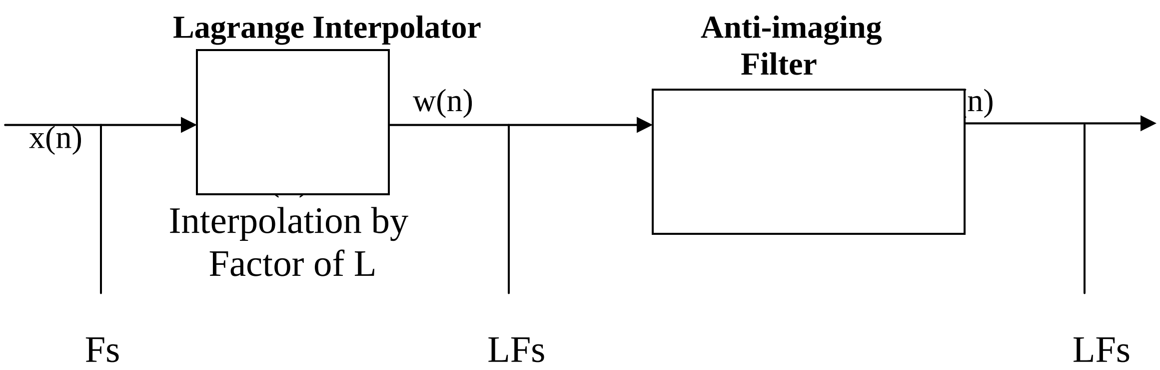

Figure 3(1)

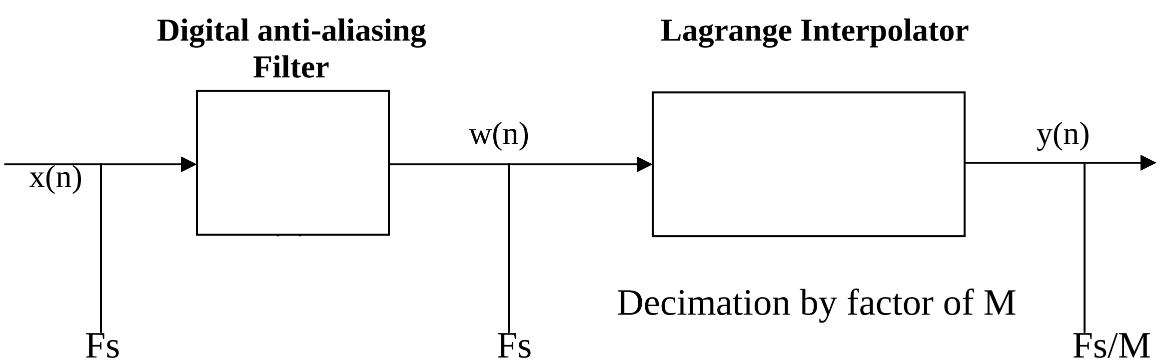

Figure 3 (2)

First diagram describes the Interpolation Process. It consist of a sample rate expander (which is the lagrange Interpolator here) and a Lowpass Filter. The lagrange interpolator increase the signal sampling frequency From “Fs” to “LFs”. This signal is then low pass filtered to remove image frequencies created by the rate increase (Lagrange Interpolator) .

Second diagram describes Decimation Process. It consist of a Digital anti-aliasing filter , h(k) and a sample rate compressor. The rate compressor reduces the sample rate from Fs to Fs/M . To prevent aliasing at the lower rate the digital lowpass filter is used to band limit the input signal to less than Fs/2M .

In both the cases we choose 2nd order elliptic lowpass Filter. The elliptic filter exhibits equiripple behaviour in both the passband and the stopband . The elliptic characteristics provides the most efficient filters interms of amplitude response. It yields the smallest filter order for a given set of specification and should be the method of first choice in IIR filter design except where the phase response is of concern.

In our implementation we use 2nd order elliptic lowpass Filter.

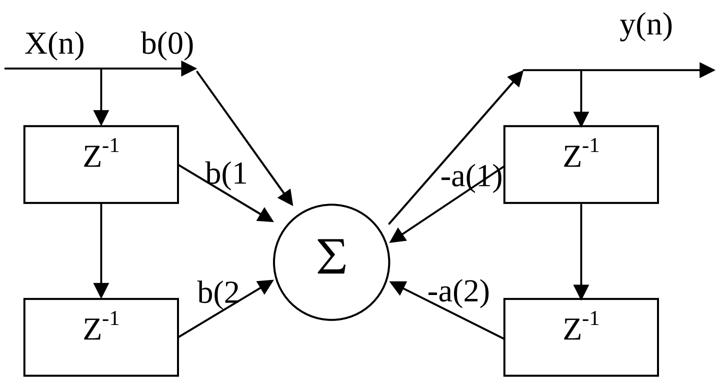

Figure 4: Block Diagram of 2nd Order Elliptic Low Pass Filter

Here the digital filter we have used is the 2nd Order lowpass digital elliptic filter with Rp=0.495 decibels of ripple in the passband and stopband Rs=50 decibels down.
The cut-off frequency Wn must be 0.0 < Wn <1.0, with 1.0 corresponding to half the sample rate.

For example suppose the signal sample rate converted from 44.1 KHz to 8KHz, then Wn = 8/44.1.

Appendix A – Document change history

Ver.No.	Editor/Author	Date dd-mmm-yy	Changes made
0.1	Arnab & Rocky Lin	14th Jan 2003	Initial Draft
0.2	Srividya M. S.	28th Jan 2004	Updated the template