Methods of Interpolation

Ansh Khandelwal

November 30, 2020

1 Abstract

This paper proposes different methods of interpolation. All the methods are coded in MATLAB and the results are recorded in the paper. The reconstructed signals are then compared with the original input signals and the mean square error is found, the error analysis on the different methods is tabulated in the paper.

2 Introduction

Interpolation is a method of fitting the data points to represent the value of a function. It has a various number of applications in engineering and science, that are used to construct new data points within the range of a discrete data set of known data points or can be used for determining a formula of the function that will pass from the given set of points (t,x).
It is a method of deriving a simple function from the given discrete data set such that the function passes through the provided data points. This helps to determine the data points in between the given data ones. This method is always needed to compute the value of a function for an intermediate value of the independent function. In short, interpolation is a process of determining the unknown values that lie in between the known data points. It is mostly used to predict the unknown values for any geographical related data points such as noise level, rainfall, elevation, and so on.

Three types of signals are which are reconstructed are:

- Polynomial: $x(t) = t^3 - 9t^2 - 0.5$
- Exponential: $x(t) = 2 * exp(\frac{1}{6}\pi t)$
- Trigonometric: $x(t) = \sin \pi t - \cos \frac{2}{3}\pi t$

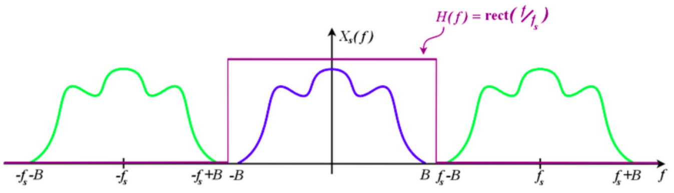
3 Methods of Interpolation

3.1 Whittaker-Shannon Interpolation

The Whittaker–Shannon interpolation or **sinc interpolation** is a method to reconstruct a continuous-time bandlimited signal from a set of equally spaced samples. Only those signals can be exactly reconstructed which:

- are band-limited. This means that the Fourier transform of the original signal, also known as the spectrum, is 0 for $|f| > B$, where B is the bandwidth.
- are sampled at the Nyquist frequency, $fs > 2B$

If it does not meet these two requirements, aliasing occurs.To recover the signal, the sampled function is simply multiplied by a low-pass filter with height equal to the sampling period T,$H_r(f)$ to isolate the original signal.
$X_r(f) = H_r(f)X_s(f)$



$$X_r(f) = H_r(f)\cdot \tag{1}$$

$X_s(f)$ where $H_r(f)$ is a LPF with gain T_s and $w_m \leq w_c \leq w_c - w_m$
Convert to the time domain. $X_s(f)$ in the time domain is the sum of a train of deltas at every period of sampling, T

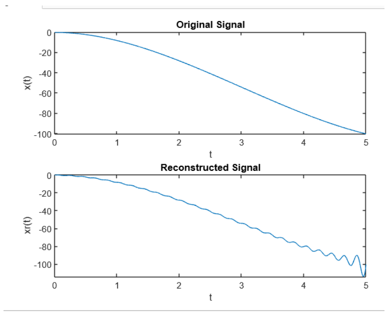
$$x_r(t) = h_r(t) * x_s(t) = T_s\frac{sin(w_c t)}{\pi t} * \sum_{k=-\infty}^{\infty} x(kT)\cdot\delta(t-kT) \tag{2}$$

When convolving a function with a shifted delta, the result is the function shifted

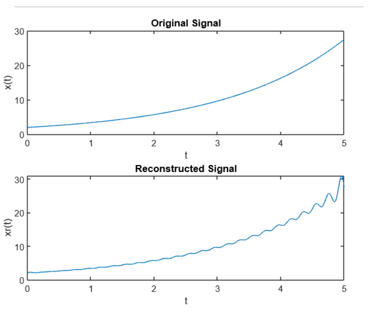
$$x_r(t) = \sum_{k=-\infty}^{\infty} x(kT)\cdot T_s\frac{sin(w_c(t-kT))}{\pi(t-kT)} \tag{3}$$

It should be noted that while this may work in the theoretical, it is impossible to make a low-pass filter with an infinite cut-off slope.

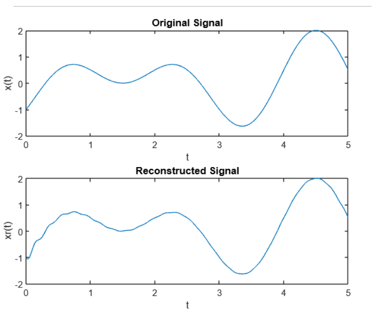
Results:



Arithmetic function



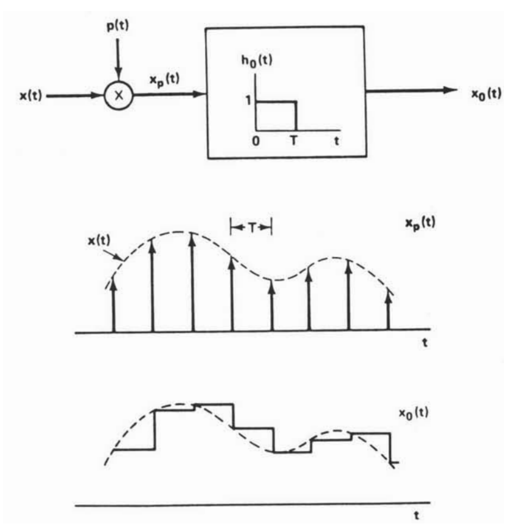
Exponential function

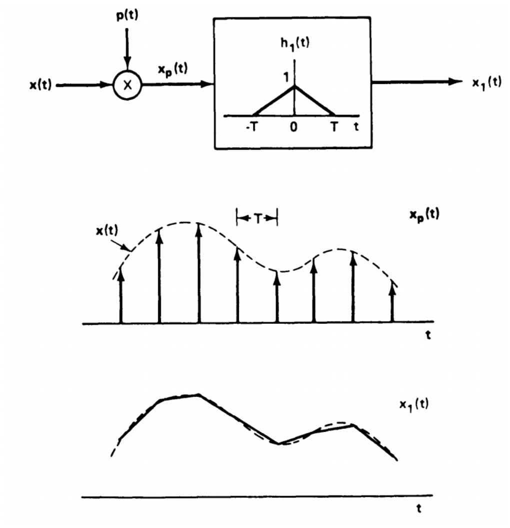


Trigonometric function

3.2 Zero-order Hold and First-order Hold Interpolation

Zero-order Hold procedure interpolates between sample points by holding each sample value until the next sampling instant. This generates a staircase-like approximation to the original signal. The zero-order hold corresponds to convolving the impulse train of samples with a rectangular pulse of duration exactly equal to the sampling period.

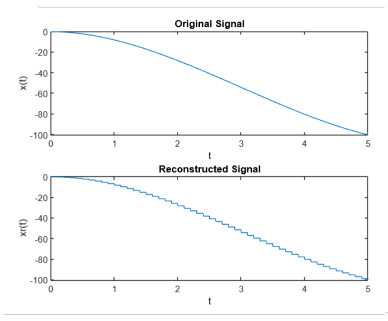




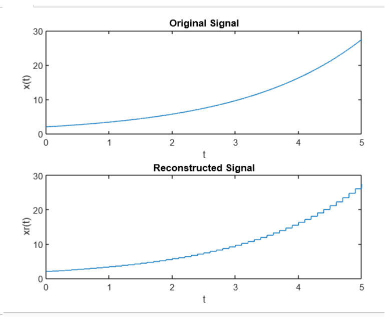
Whereas,Linear interpolation, also commonly referred to as a first-order hold, corresponds to connecting the sample points by straight line segments. The first-order hold corresponds to an impulse response for the reconstruction filter that is a triangle of duration equal to twice the sampling period. With this procedure the approximate lowpass filter has a frequency response that is the Fourier transform of a triangle.

Results:

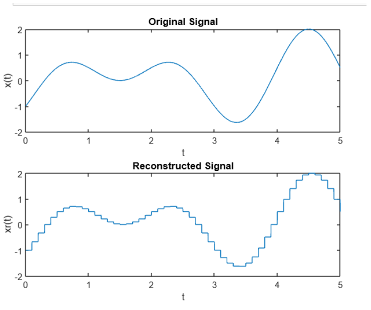
For zero-order hold:



Arithmetic function

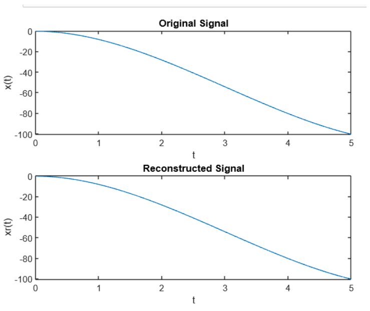


Exponential function

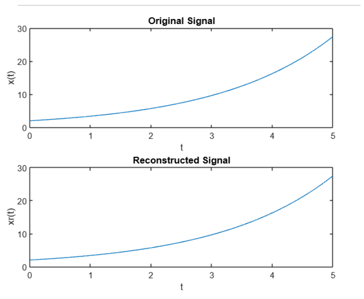


Trigonometric function

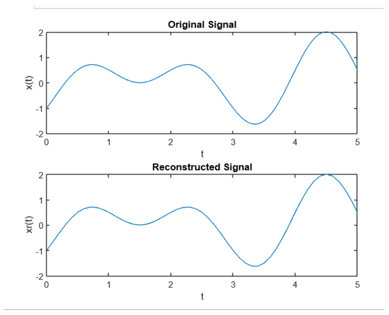
For First-order hold:



Arithmetic function



Exponential function



Trigonometric function

3.3 Polynomial Interpolation Methods

Classically interpolation was based on constructing a polynomial that passes through all sample points. Polynomial interpolation is defined as the interpolation of a given set by the polynomial of the lowest possible degree that passes through the points of the data set.

3.3.1 Lagrange Interpolation

In Lagrange polynomial interpolation method, we construct a polynomial using the weighted average of slopes of all sample points with all other sample points. This Lagrange polynomial is of the order N for $N+1$ sample points.

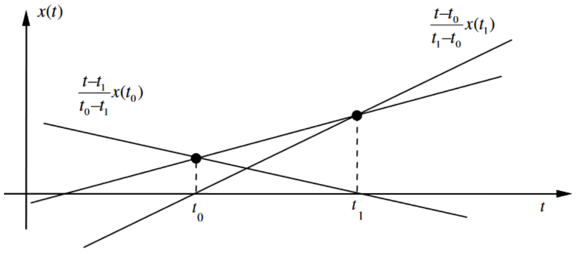
$$L_k = \frac{(t-t_0)(t-t1)(t-t_2)\dots(t-t_N)}{(t_k-t_0)(t_k-t1)(t_k-t_2)\dots(t_k-t_N)} = \prod_{i:t_k\neq t_i}^{N} \frac{t-t_k}{t_i-t_k} \tag{4}$$

L_k s are themselves polynomials of degree N, and their summation gives us:-

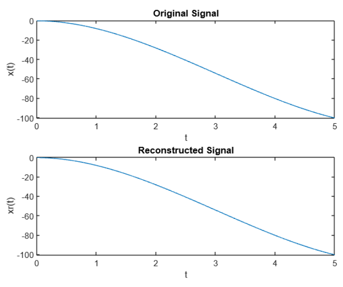
$$\hat{x}(t) = L_0 x_s(t_0) + L_1 x_s(t_1) + L_2 x_s(t_2) \dots + L_N x_s(t_N) = \sum_{k=0}^{N} x_s(k) * L_k \tag{5}$$

Here for each sample point, we take the slope of it with all other sample points weighted with y coordinate value of the sample. The sum of the product of all these components with their corresponding sample $(x_s(t_k)(L_k))$ gives us our reconstructed signal. This process can also be thought as combination of N linear lines of specified slopes. This interpolation has high computational complexity and no re usability as L_ks of a polynomial of order N cannot be re used for higher orders.

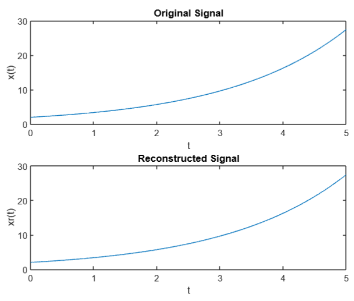
As we see here, considering first two points the intersection of lines joining (x0,t1) and (x1, t0) gives the two point for the estimated polynomial. Doing this for all points will lead us to the Lagrange interpolation.



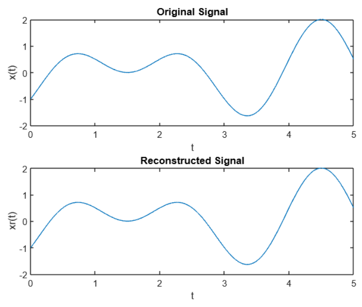
Results:



Arithmetic function



Exponential function



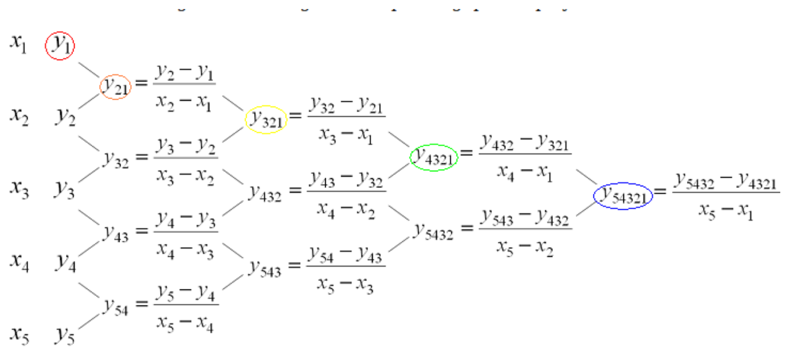
Trigonometric function

3.3.2 Newton Interpolation

This is another type of polynomial interpolation method where we try to find a polynomial that passes through the given sample points. Here we define the polynomial coefficients as:-

$$Y_0 = x(1)$$
$$Y_{2,1} = \frac{x(2) - x(1)}{t(2) - t(1)}$$
$$Y_{i,j} = \frac{Y_i - Y_j}{t(i) - t(j)} \tag{6}$$

These coefficients are multi derivatives of the points of order from 0 to N. Visual representation for the coefficient calculation for N = 5 is as :

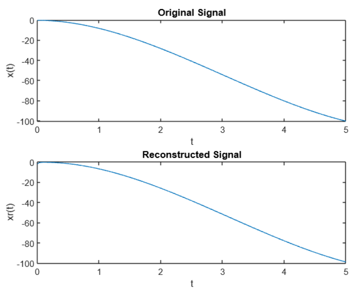

x_1 y_1
x_2 y_2
x_3 y_3
x_4 y_4
x_5 y_5
$y_{21} = \frac{y_2 - y_1}{x_2 - x_1}$
$y_{32} = \frac{y_3 - y_2}{x_3 - x_2}$
$y_{43} = \frac{y_4 - y_3}{x_4 - x_3}$
$y_{54} = \frac{y_5 - y_4}{x_5 - x_4}$
$y_{321} = \frac{y_{32} - y_{21}}{x_3 - x_1}$
$y_{432} = \frac{y_{43} - y_{32}}{x_4 - x_2}$
$y_{543} = \frac{y_{54} - y_{43}}{x_5 - x_3}$
$y_{4321} = \frac{y_{432} - y_{321}}{x_4 - x_1}$
$y_{5432} = \frac{y_{543} - y_{432}}{x_5 - x_2}$
$y_{54321} = \frac{y_{5432} - y_{4321}}{x_5 - x_1}$

The circled values can be used to form the interpolation polynomial as such:-

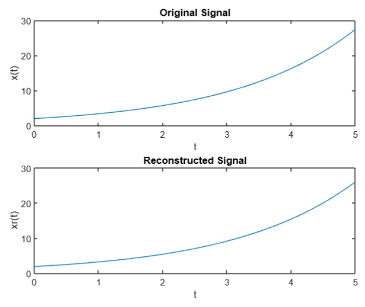
$$\hat{x}(t) = Y_0 + Y_{2,1} * (t - t(1)) + Y_{3,2,1} * (t - t(1))(t - t(2)) + \dots \tag{7}$$

In this reconstruction we take weighted average of i^{th} divided-difference of the i^{th} point. That's why it is also known as divided differences interpolation. This is a direct application of Taylor's polynomial which best fits a curve given it's values and it's rate of change and it's rate of rate of change etc. at one particular point on the curve.

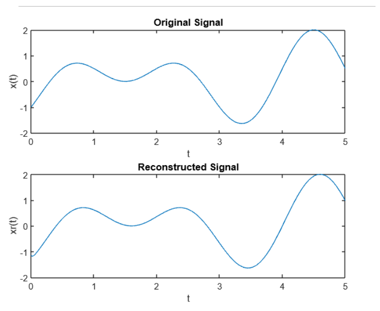
Results:



Arithmetic function



Exponential function

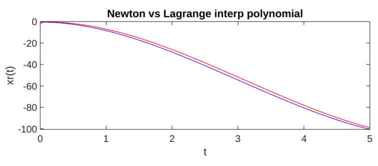


Trigonometric function

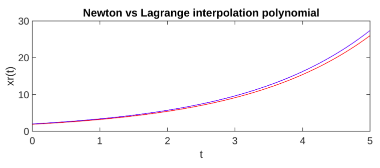
Newton Vs Lagrange

The difference between the two polynomial interpolation is that, Lagrange is sometimes said to require less work, and is sometimes recommended for problems in which it is known, in advance, from previous experience, how many terms are needed for sufficient accuracy.
The divided difference methods have the advantage that more data points can be added, for improved accuracy. The terms based on the previous data points can continue to be used. With the ordinary Lagrange formula, to do the problem with more data points would require re-doing the whole problem.
We have plotted signals reconstructed by both Lagrange and Newton methods in the same plot in order to compare the polynomials.

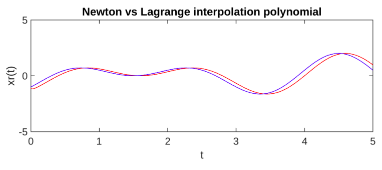
Results:



Arithmetic function



Exponential function



Trigonometric function

Red: Newton polynomial   Blue: Lagrange polynomial

Observation: Theoretically the polynomials obtained are identical but due to implementation difference in both the methods and also due to approximation difference in Matlab code we find a small deviation in both the polynomials.

3.3.3 Hermite Interpolation

This interpolation method is similar to Spline interpolations.In this method the polynomial is made to not only match the values but also the derivative values too. Say we have available values of derivatives up to k^{th} order. then Same as with spline, we assume a polynomial of degree (N+1)(K+1) to take the interpolation values. Thus we have a system of K+1 equations for all sample points as:

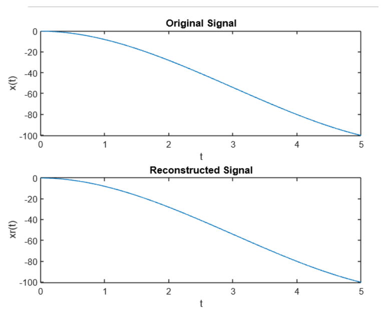
$$p(t_i) = x(t_i)$$
$$p'(t_i) = x'(t_i) \dots$$
$$p^K(t_i) = x^K(t_i)$$

Thus we will have a polynomial of the form:

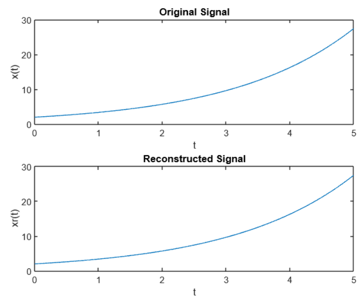
$$p(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_{M-1} t^{M-1} \tag{8}$$

Where M = (N+1)(K+1) so in total we have (N+1)(M+1) equations and we can theoretically find all coefficients using these equations. But as we observe. even for moderate values of N and M the computation becomes very large and complex. And thus this is not a practical method. Although this method gives better interpolating polynomial that also follows the derivative characteristics.

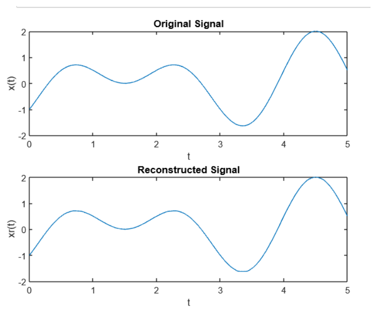
Results:



Arithmetic function



Exponential function



Trigonometric function

3.4 Spline Interpolation

These are set of interpolation methods where the function is interpolated based on a special function called spline function. These are preferred over polynomial as the results it yields are same even when using lower degree polynomials.

3.4.1 Cubic Spline Interpolation

This is a type of Spline interpolation method where we try to fit a cubic spline polynomial between the given time interval. Usually to reduce error, the given signal is divided into multiple smaller time intervals. And we try to find a cubic spline polynomial to interpolate that specific interval. The cubic spline polynomial that we assume is of the form:

$$p(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0 \tag{9}$$

Lets assume that we are finding the cubic spline polynomial for the time interval $y \leq t \leq z$ We define a local variable $\tau = t - y$ for simplification and from now on we will be talking in terms of $p(\tau)$. These 4 a_is characterize the cubic polynomial and we need system of 4 equations to solve for the coefficients a_is.First coefficient can be obtained easily using

$$p(0) = x(y) = a_0 \tag{10}$$

a_2 and a_3 can be determined if we know the 2^{nd} derivative of $p(t)$ at $t = 0$ and z. and on substituting in 8 we get a_1.

$$a_2 = p''_y/2 \qquad a_3 = \left(p''_{z-y} - p''_y\right)/6(z-y) \qquad a_1 = \tfrac{x(z-y)-x(0)}{z-y} - (z-y)\tfrac{p''_{z-y}+2p''_y}{6}$$

substituting these in p we have:-

$$p(\tau) = \tfrac{p''_{z-y}-p''_y}{6(z-y)}\tau^3 + \tfrac{p''_y}{2}\tau^2 + \tfrac{x(z-y)-x(0)}{z-y} - (z-y)\tfrac{p''_{z-y}+2p''_y}{6}\tau + x(0)$$

Now to find the second derivative values at the two points, we need two more equations. We can get these equations by realising that the first derivative of the polynomial should be continuous at the end points (0, z-y). Thus we can equate their first derivatives on the shared point.

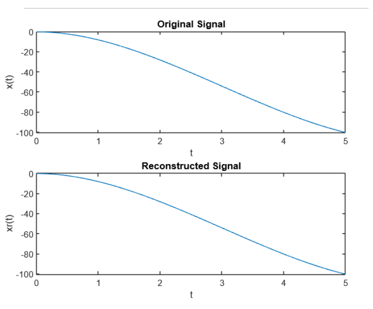
$$p'(0) = \tfrac{-(z-y)}{6}(p''_{z-y} + 2p''_y) + \tfrac{1}{z-y}(x(z-y) - x(y))$$
$$p'(z-y) = \tfrac{(z-y)}{6}(2p''_{z-y} + p''_y) + \tfrac{1}{z-y}(x(z-y) - x(y))$$

similarly for preceding time interval $q \leq t \leq y$

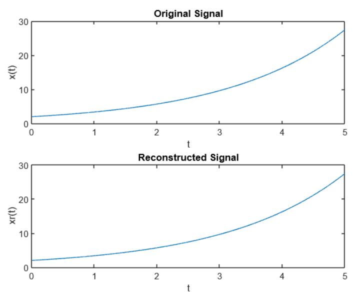
$$p'(y-q) = \tfrac{(y-q)}{6}(2p''_{y-q} + p''_q) + \tfrac{1}{y-q}(x(y-q) - x(q))$$

If we equate $p'(y-q)$ and $p'(0)$ of the intervals [q,y] and [y,z] respectively, and repeat this for all samples we will have N-1 equations and N+1 second derivative variable we need to find. We can approximate the second derivatives of the signal at the two end points according the samples and substitute or we can take them as zero. After this approximation we will have N-1 equations and N-1 variables to find.

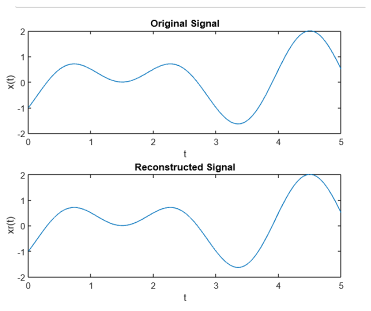
Results:



Arithmetic function



Exponential function

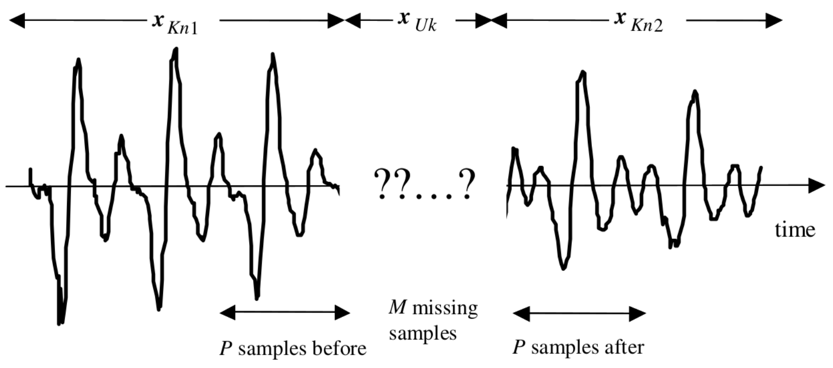


Trigonometric function

3.5 Model-Based Interpolation

Model-based interpolation, prediction, and approximation are contingent on the choice of model: since multiple alternative models typically can reasonably be entertained for each of these tasks, and the results are correspondingly varied, this often is a considerable source of uncertainty.

Problem statement: Lets consider a N-length signal given with M lost samples. Lets denote the whole N-length signal vector be x, unknown sample vector as x_{Uk} of length M and known samples vector as X_{Kn} of length $N - M$. Now we need to make a model so that it can predict the lost samples based on the known samples available.



Here in this figure on each side of the missing samples, P samples are used to interpolate the segment.
Also here $x_{Kn} = [x_{Kn1}\ x_{Kn2}]^T$, x can be written as an equation of rearranged matrices as:

$$x = Kx_{Kn} + Ux_{Uk} \tag{11}$$

Both K and U matrix are rearrangement matrices of size $NX(N-M)$ which together form the x vector as a column vector as given in the equation above. Here K matrix is implemented as a matrix with 0 values at all its elements except it has 1s at the known samples position. Similarly, U matrix has 1s only at the missing sample element position.

3.5.1 Maximum A Posteriori Interpolation

In this method a posterior pdf of unknown samples is estimated. The N-signal using is modelled using a gaussian vector process with zero mean. So the pdf of x is:

$$f_X(x) = \frac{1}{(2\pi)^{N/2}|\Sigma_{xx}|^{1/2}} exp(\frac{-1}{2} x^T \Sigma_{xx}^{-1} x) \tag{12}$$

here Σ_{xx} is the covariance matrix of x.
The posterior pdf of x_{Uk} using the known samples, can be expressed using Bayes' rule. This pdf basically finds the probability of x_{Uk} vector given the known samples x_{Kn} (conditional probability)

$$f_X(x_{Uk}|x_{Kn}) = \frac{f_X(x)}{f_X(x_{Kn})} \tag{13}$$

Now for the above equation $f_x(x_{kn})$ is constant as these are the known samples, henceforth to get best possible estimate of the unknown samples we have to maximise $f_X(x)$. Hence we get:

$$\hat{x}_{Uk}^{MAP} = arg_{x_{Uk}} max f_x(Kx_{Kn} + Ux_{Uk}) \tag{14}$$

In order to model a covariance matrix we started with the main diagonal. The main diagonal has the constant value which is equal to the variance of the known samples.

$$var = variance(x_{Kn}) \tag{15}$$

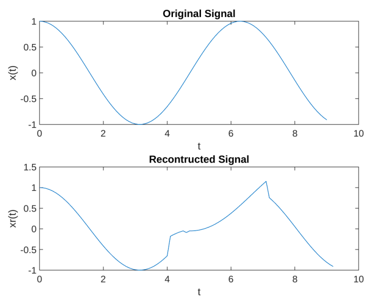
So the main diagonal has the value equal to v. Now the subsequent diagonals of the matrix are filled with values which are just smaller (c) than the diagonal before it. As the value of the signal at time t would be more related to samples closer to it than the samples further away, and that is why the covariance decreases as we move away from the diagonal. This is the model used to create the covariance matrix as described above:

$$\Sigma_{xx} = \begin{bmatrix} v & v-c & v-2c & ... & v-(N-1)c \\ v-c & v & v-c & ... & v-(N-2)c \\ v-2c & v-c & v & ... & v-(N-3)c \\ . & & & & \\ . & & & & \\ . & & & & \\ v-(N-1)c & v-(N-2)c & v-(N-3)c & ... & v \end{bmatrix} \tag{16}$$

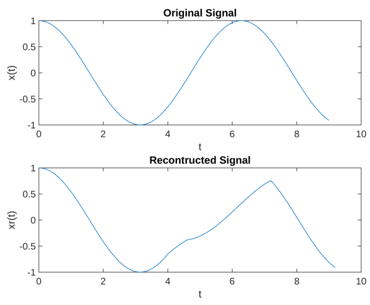
Finally, the MAP signal estimate is obtained by finding derivative of log- likelihood function $lnf_X(x|x_{Kn})$ with respect to x_{Uk} and the final equation is:

$$x_{Uk} = -(U^T\Sigma_{xx}^{-1}U)^{-1}U^T\Sigma_{xx}^{-1}Kx_{Kn} \tag{17}$$

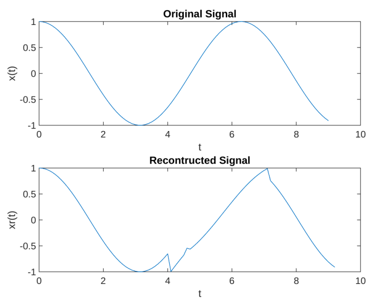
Results:



for c = 0.009



for c = 0.01

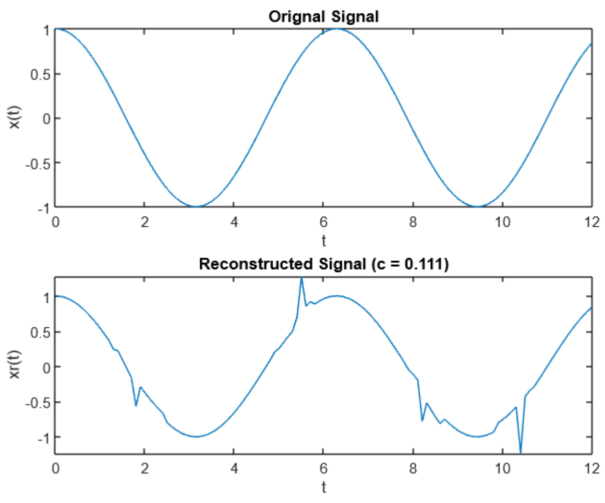


for c = 0.0111

Problem statement: Lets consider a N-length signal given with lost samples at arbitrary time intervals. Try reconstructing the signal using the MAP method.

We can use the exact code but we only need to change the K and U matrices in order to reconstruct using the similar method for each individual missing set. We also change the covariance matrix accordingly, as here there are lesser unknown samples but at various irregular time intervals.

Result:



3.5.2 Interpolation Based on a Short-Term Prediction Model

This model estimates the missing samples x_{Uk} using the previous P samples which will be inputted by the user. It uses a specific autoregressive (AR) model for prediction. An autoregressive (AR) signal $x(m)$ is described as follows:

$$x(m) = \sum_{k=1}^{p} a_k x(m-k) + e(m) \tag{18}$$

Here a_k are model coefficients and $e(m)$ is a zero mean excitation signal (error vector). By rearranging term in matrix form we get:

$$e(u_{Kn}, a) = x - Xa \tag{19}$$

Here X matrix is just a rearrangement of terms to satisfy the equation in vector form and a is the column vector having all the AR model coefficients.
To find error vector from this equation we have two unknown vectors a and x_{Uk} which we try to find by finding an inner vector product. The minimum mean square error is given by

$$e^T e(x_{Uk,a}) = x^T x + a^T X^T X a - 2a^T X^T x \tag{20}$$

Differentiating this equation results in a set of nonlinear equations of cubic order whose solution is non-trivial. In order to estimate the AR model coefficients a we take an assumption to consider all x_{Uk} in the X matrix as 0 as the system needs to be linearized and the estimate of AR coefficients should be made only based on the known samples. An estimate and maximise algorithm is used and we get

$$\hat{a} = (X_{Kn}^T X_{Kn})^{-1} (X_{Kn}^T x_{Kn}) \tag{21}$$

Now as the a vector is known it is substituted in the (18) equation. The equation is rearranged to get x_{Uk} and x_{Kn} separated and hence

$$e = A_1 x_{Uk} + A_2 x Kn \tag{22}$$

where A_1 and A_2 are rearranged matrices derived from the (18) equation. Now the least square error is found which is given by

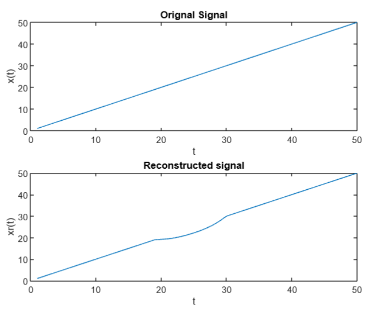
$$e^T e = (A_1 x_{Uk} + A_2 x_{Kn})^T (A_1 x_{Uk} + A_2 x_{Kn}) \tag{23}$$

This error is minimised with respect to the unknown samples and finally the unknown samples are predicted as

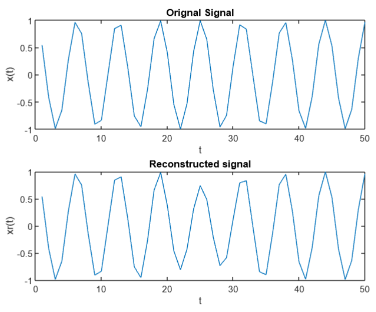
$$\hat{x}_{Uk}^{LSAR} = -(A_1^T A_1)^{-1}(A_1^T A_2)x_{Kn} \tag{24}$$

where $\hat{x}_{Uk}^{LSAR}$ is the least square error estimate of the unknown samples.

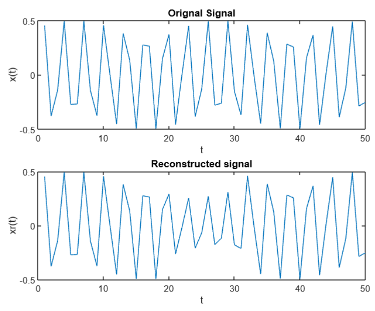
Results:



x(t) = t, M = 12



x(t) = cos(t), M = 12

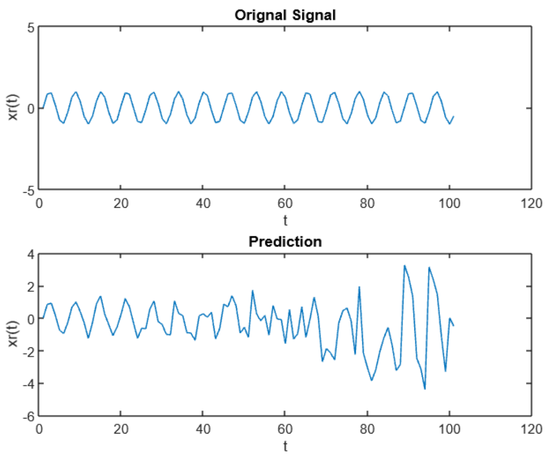


x(t) = cos(t)*sin(t), M = 12

In the above reconstructed signals, the original signals had the lost samples in the range [20,31].

Problem Statement: 10 samples of a sine signal is given as an input, predict the signal for next 90 samples and show the output.

Here we initially inputted the signal and predicted the next sample of it using the short term prediction based on last 10 samples, then shifted the signal to predict the next sample. The process is repeated till we predict 100 samples.
This is how the AR model is used in estimating the stock price fluctuations and weather forecast.



Avg mean error = 1.0066

4 Error Analysis

In this section, we try and analyse the quality of interpolation that different methods give for the three different types of signals that we have chosen. We are going to observe the Maximum Absolute error for each of the interpolating signals. Which is given by:

$$Err_{MAE} = max_t(|x(t) - \hat{x}(t)|) \tag{25}$$

For Trigonometric Signal:

Interpolation Method	Maximum Absolute Error
Spline	$1.2410 * 10^{-4}$
Lagrange	$1.2 * 10^{-3}$
Newton	$2.2 * 10^{-3}$
Hermite	$1.45 * 10^{-2}$
Sinc	$1.69 * 10^{-2}$
Linear	$1.76 * 10^{-2}$
Zero Hold	$4.95 * 10^{-1}$

For Exponential Signal:

Interpolation Method	Maximum Absolute Error
Spline	$5.5175 * 10^{-6}$
Hermite	$1.6549 * 10^{-4}$
Newton	$7.732 * 10^{-3}$
Lagrange	$8.4 * 10^{-3}$
Linear	$9.2 * 10^{-3}$
Sinc	$3.97 * 10^{-1}$
Zero Hold	1.3842

For Polynomial Signal Signal:

Interpolation Method	Maximum Absolute Error
Spline	$5.6843 * 10^{-14}$
Newton	$4.08 * 10^{-3}$
Hermite	$6.7 * 10^{-3}$
Linear	$2.21 * 10^{-2}$
Lagrange	$6.54 * 10^{-2}$
Sinc	1.040
Zero Hold	2.6720

4.1 Observations

NOTE: Some Interpolation methods give higher distortion at the ends (due to not being able to have infinite summation or data points) we find the error after excluding some portion of the ends.

- Spline consistently for all three types of signals gives the least error.
- Zero hold gives the maximum error for all signals.
- Newton and Lagrange Interpolations give very similar quality of interpolations, though If we increase K for Hermite interpolation, it would be better but with Large number of computations.
- Sinc Interpolation also gives considerable error, but if we were to provide it with samples from $-\infty$ to ∞, It would act as ideal interpolation and give zero error.
- Lagrange and linear give average error compared to other interpolation methods. While Lagrange gives better results for more fluctuating signals like trigonometric which linear performs worse at.

5 References

- https://www.projectrhea.org/rhea/index.php/HW3_Signal_Reconstruction_Interpolation
- https://www.sci-hub.do/10.1109/CDC.1994.411070
- https://www.gisresources.com/types-interpolation-methods_3/
- https://byjus.com/maths/interpolation/#:~:text=They%20are%3A,the%20most%20adjacent%20data%20point.
- https://www.math-linux.com/mathematics/interpolation/article/newton-s-interpola
- http://faculty.washington.edu/finlayso/ebook/interp/spline.htm
- https://en.wikipedia.org/wiki/Hermite_interpolation