Lecture 23 - Notes

Rahul Arya

April 2019

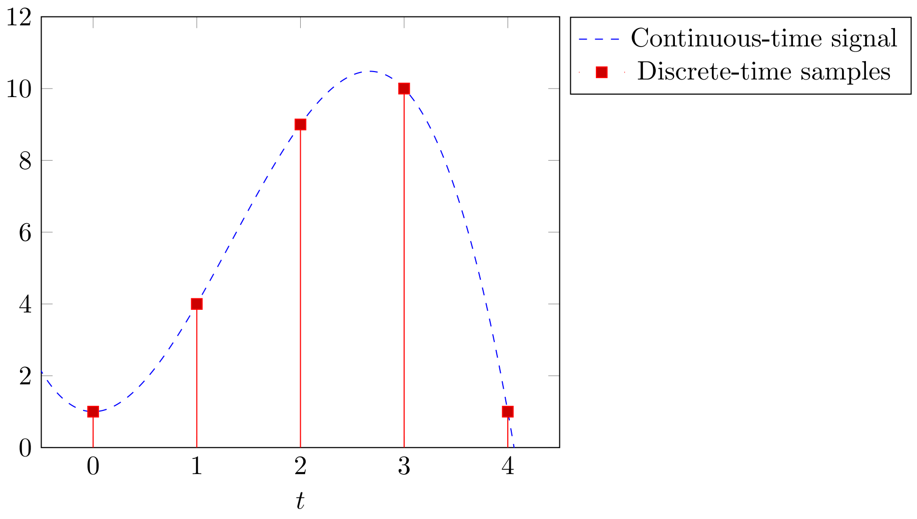
1 Overview

Over the next few lectures, we will begin to study discrete signals in the frequency domain. Of course, signals in the frequency domain are by their construction defined over continuous time. Thus, we need some way to convert a discrete-time signal into a continuous-time one, that preserves its important properties. We will address this problem, known as *interpolation*, in this lecture.

2 Properties of an Interpolation

In the above definition of interpolation, there are two main ambiguities: what exactly is meant by a "discrete-time signal", and what are its important properties that should be preserved after interpolation?

We define a discrete-time signal to be a series of measurements (known as *samples*) of some underlying continuous-time signal, as illustrated below:

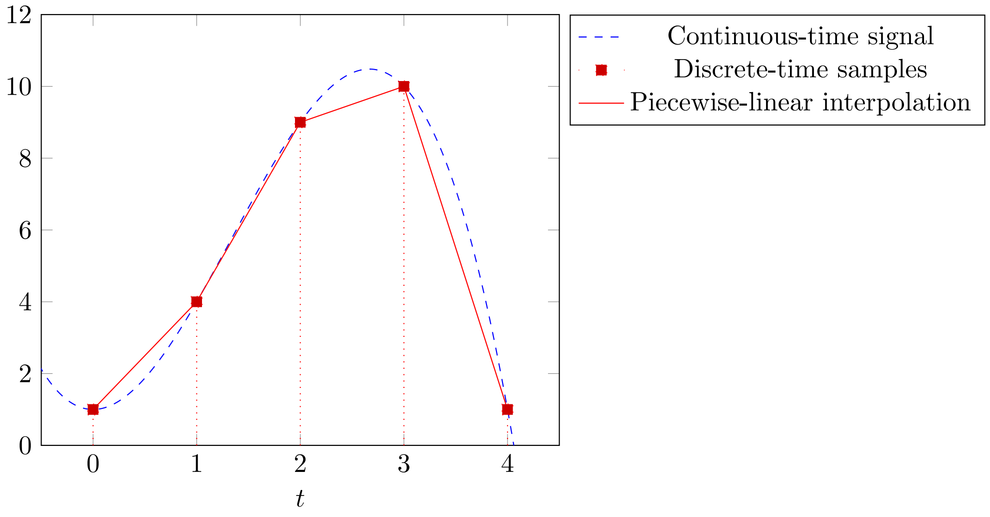


In almost all cases (including all the cases we will consider here), our samples will be made at a constant interval Δ. For instance, in the above figure, $\Delta = 1$.

Interpolation can be thought of as the task of (approximately) recovering the original continuous-time signal from our discrete samples. An immediate consequence of this is that an interpolated signal should pass through all of its sample points, since we know that the original signal certainly did. In fact, we will consider this property to be the only requirement that an interpolation must satisfy for now, though we will introduce more requirements in the future. Thus, we can view interpolation simply to be the construction of a continuous-time function that passes through all our sample points.

3 Simple Interpolations

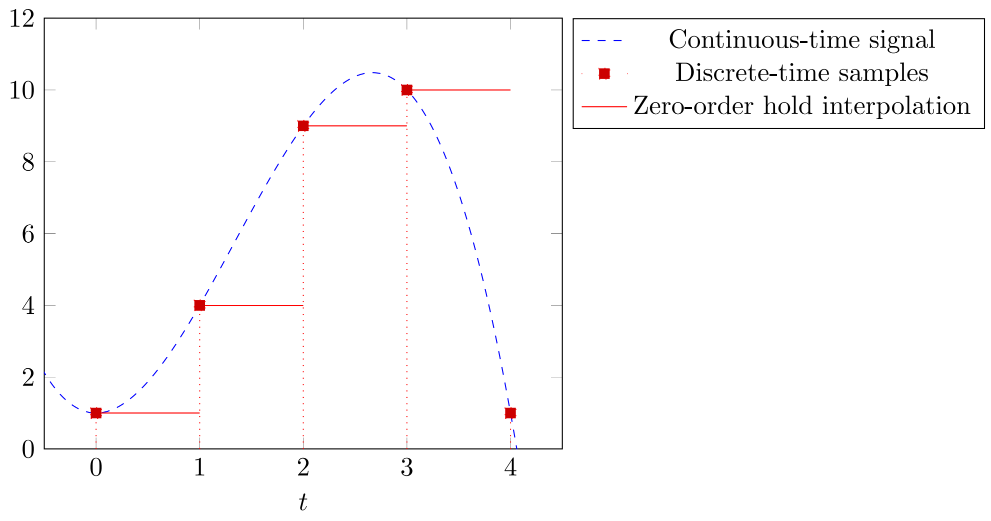
The problem of interpolation doesn't sound too hard - intuitively, it is trivial to construct a continuous-time function passing through a given set of points - just join them together with straight lines! This interpolation is known as *piecewise linear* interpolation, and is a very natural construction. For instance, in the above figure, we obtain the piecewise-linear interpolation



Our piecewise-linear interpolation seems to work fairly well at approximating the original signal, as we intuitively expected, but is less accurate near intervals where the continuous-time signal changes slope rapidly, such as near the local maximum in the above figure.

Another, even simpler, interpolation would be to just forget about trying to "interpolate" at all, and just retain the value of the previous sample point.

This technique, known as *zero-order hold* interpolation, produces the following result:



Notice that this approximation is quite a bit worse than our piecewise-linear interpolation, but still passes through all our sample points and at least somewhat retains the shape of the original signal.

4 Basis Functions

As it turns out, the previous two interpolations can both be described as examples of a broader class of interpolations, which are based on the idea of *basis functions*. Basis function interpolations have the following two additional properties, which are both fairly natural:

- The interpolation produced from a linear combination of two discrete-time signals is the same linear combination of the interpolation of the two discrete-time samples. In other words, interpolation can be thought of as a linear transformation of their input discrete-time signals onto the space of continuous functions.
- Interpolations do not depend on a reference time. That is to say, if we shift a discrete-time signal forward by k samples, its interpolation will shift forward by $k\Delta$ time.

Observe that the above two properties are clearly satisfied by both the piecewise-linear and zero-order hold interpolations we saw earlier.

Now, we define the *basis function* of a particular interpolation to be the interpolation $\phi(t)$ of the discrete signal with sample points

$$\dots, (-2\Delta, 0), (-\Delta, 0), (0, 1), (\Delta, 0), (2\Delta, 0), \dots$$

In other words, it is the interpolation of a discrete signal that is 0 everywhere except at $t = 0$, where it is 1. We claim that we can use this basis function, along with the known properties of basis function interpolations, to interpolate *any* discrete-time signal. In other words, the choice of basis function $\phi(t)$ fully determines the behavior of our interpolation.

To prove the above claim, let our arbitrary input discrete-time signal be $\vec{x}$. Let $\vec{b}_i$ be a discrete-signal that is 0 everywhere except at $t = i\Delta$, where it is 1. By the second property of basis-function interpolations, it is clear that

$$\begin{aligned} & \text{Interpolate}(\vec{b}_0) = \phi(t) \\ \implies \quad & \text{Interpolate}(\vec{b}_k) = \phi(t - k\Delta). \end{aligned}$$

It should be clear that we can choose coefficients α_i such that

$$\vec{x} = \sum_i \alpha_i \vec{b}_i,$$

treating $\vec{x}$ to be of finite length.

Now, we can apply the first property of basis-function interpolations, to see that

$$\begin{aligned} \text{Interpolate}(\vec{x}) &= \text{Interpolate}(\sum_i \alpha_i \vec{b}_i) \\ &= \sum_i \alpha_i \cdot \text{Interpolate}(\vec{b}_i). \end{aligned}$$

Combining this result and the result immediately preceding it, we therefore find that

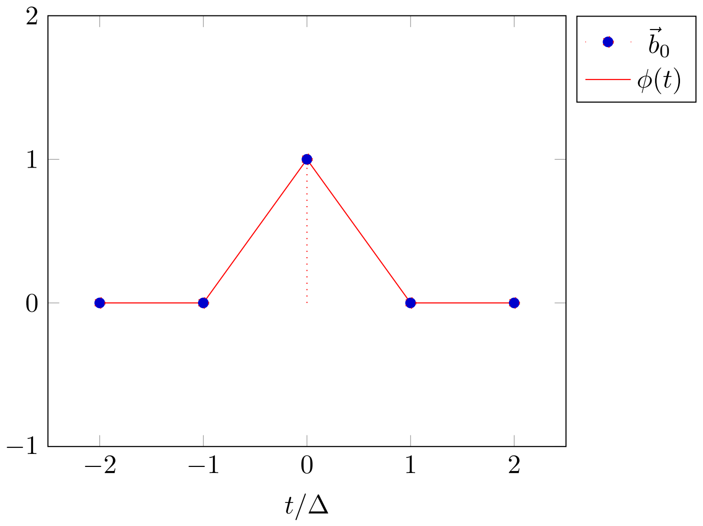
$$\text{Interpolate}(\vec{x}) = \sum_i \alpha_i \cdot \phi(t - i\Delta),$$

so we have constructed the interpolation of an arbitrary discrete-time signal from just our basis function, proving our claim.

5 Example Basis Functions

Let's apply the above general result to the particular examples of piecewise-linear and zero-order hold interpolations, in order to gain some intuition for this new way of thinking about interpolation. Recall that both these types of interpolation satisfy the two additional requirements of basis-function interpolations, so their basis functions must exist.

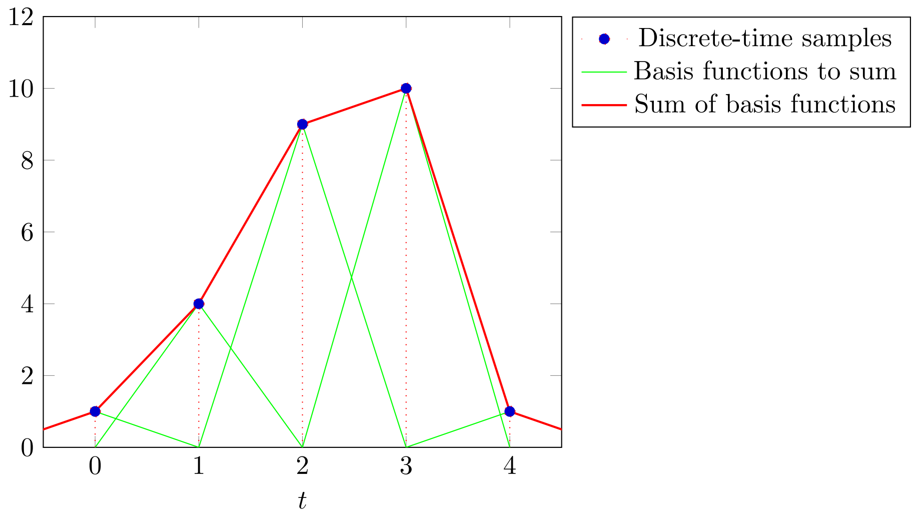
Consider piecewise-linear interpolation first. By the definition of a basis function, we need to compute the piecewise-linear interpolation of $\vec{b}_0$, which we see graphically is as follows:



Algebraically, we see that

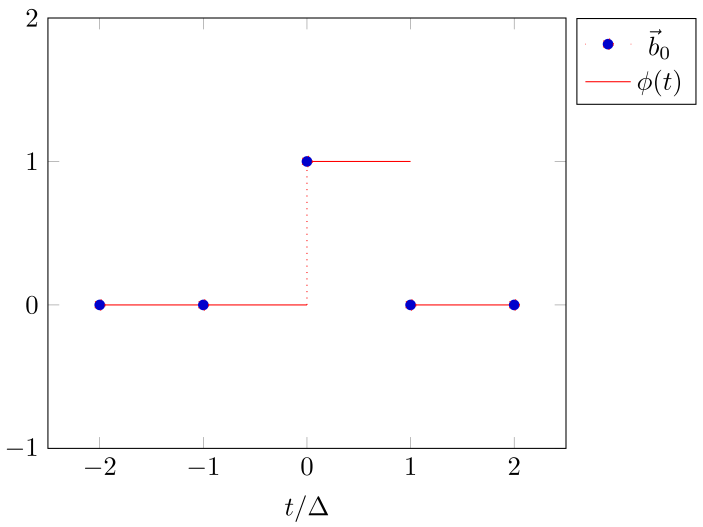
$$\phi(t) = \begin{cases} 0, & \text{for } t < -\Delta \\ 1+t, & \text{for } -\Delta \le t < 0 \\ 1-t, & \text{for } 0 \le t < \Delta \\ 0, & \text{for } t \ge \Delta \end{cases}.$$

Solving for the interpolation of a set of sample points with $\Delta = 1$ by evaluating the summation we obtained in the previous section, we see that it is equivalent to summing the following interpolations graphically



Notice that the sum of the scaled and translated basis functions yields exactly the piecewise-linear interpolation that we saw earlier, demonstrating how piecewise-linear interpolation can be thought of as summing basis functions.

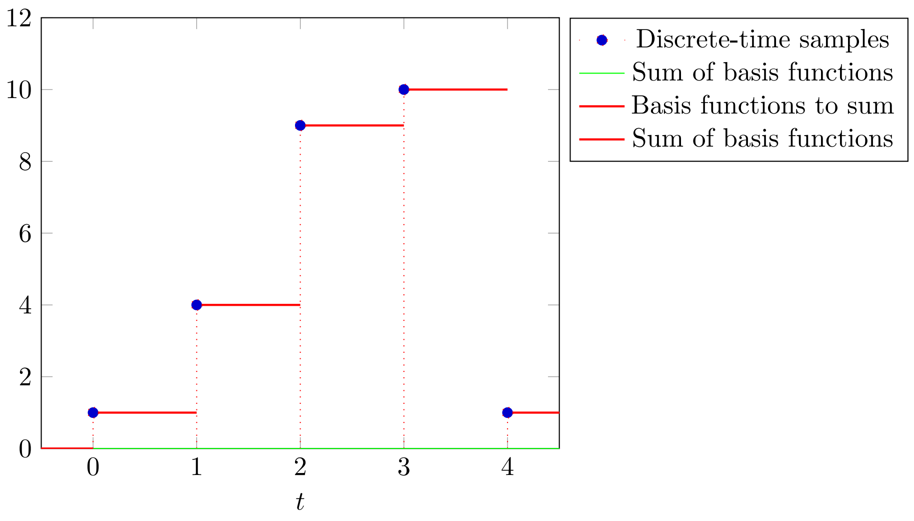
Now, let's look at zero-order hold interpolation in a similar manner. Computing the zero-order hold interpolation of $\vec{b}_0$, we graphically obtain:



Algebraically, we see that

$$\phi(t) = \begin{cases} 0, & \text{for } t < 0 \\ 1, & \text{for } 0 \le t < \Delta \\ 0, & \text{for } t \ge \Delta \end{cases}.$$

And evaluating the summation from the previous section in order to compute the zero-order hold interpolation of an arbitrary discrete-time signal with $\Delta = 1$, we see that it is equivalent to summing the following interpolations graphically:



Again, we obtain exactly the same interpolation as before, when we calculated the zero-order hold interpolation directly.

6 Sinc Interpolation

The motivation behind the general idea of basis-function interpolation is that we can now construct new interpolations easily, by picking an appropriate basis function. Recall that a basis function $\phi(t)$ was defined to be the interpolation of $\vec{b}_0$. Thus, by the first property of interpolations that we established at the beginning,

$$\phi(0) = 1 \text{ and } (\forall k \neq 0)\phi(\Delta k) = 0,$$

since an interpolation must pass through all the provided sample points. But no further restrictions are present, meaning that *any* function satisfying this condition can give rise to a new basis-function interpolation.

Furthermore, notice (though we will not prove it explicitly) that an interpolation will be non-differentiable when its basis function is itself non-differentiable. For instance, observe that the basis function for piecewise-linear interpolation had three non-differentiable points (at $t = -1$, $t = 0$, and $t = 1$), so it produces interpolations that are themselves non-differentiable at some points.

Similarly, an interpolation will be discontinuous only if its basis function is itself discontinuous. Since the piecewise-linear basis function was continuous, the piecewise-linear interpolations of arbitrary discrete-time signals were all continuous as well. However, since the zero-order hold basis function was discontinuous at $t = 0$ and $t = 1$, it gave rise to discontinuous interpolations as well.

More generally, most properties that are preserved under linear transformations (such as differentiability, continuity, boundedness, or continuity) are true of a basis-function interpolation exactly when they are true of the basis function itself. It is often the case that we want a method of producing a smooth interpolation from a discrete-signal - for instance, smoothness turns out to be useful when performing frequency analysis.

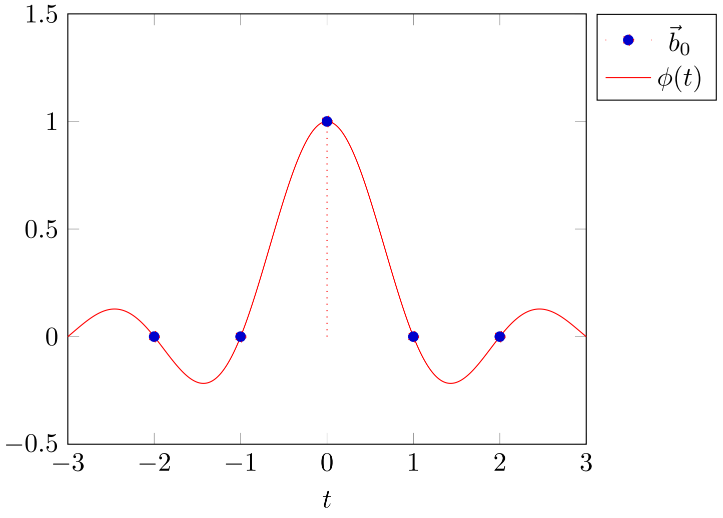
But to produce one, we need to have a smooth basis function - since neither of the two functions we currently have are differentiable, they are clearly not suitable candidates. Instead, we will propose the *sinc* basis function, defined as

$$\phi(t) = \frac{\sin(\pi t/\Delta)}{\pi t/\Delta}$$

for all $t \neq 0$, with the discontinuity at $t = 0$ removed by defining $\phi(0) = 1$[1]. Observe that for any integer $k \neq 0$,

$$\phi(\Delta k) = \frac{\sin(\pi(k\Delta)/\Delta)}{\pi(k\Delta)/\Delta} = \frac{\sin(k\pi)}{k\pi} = 0,$$

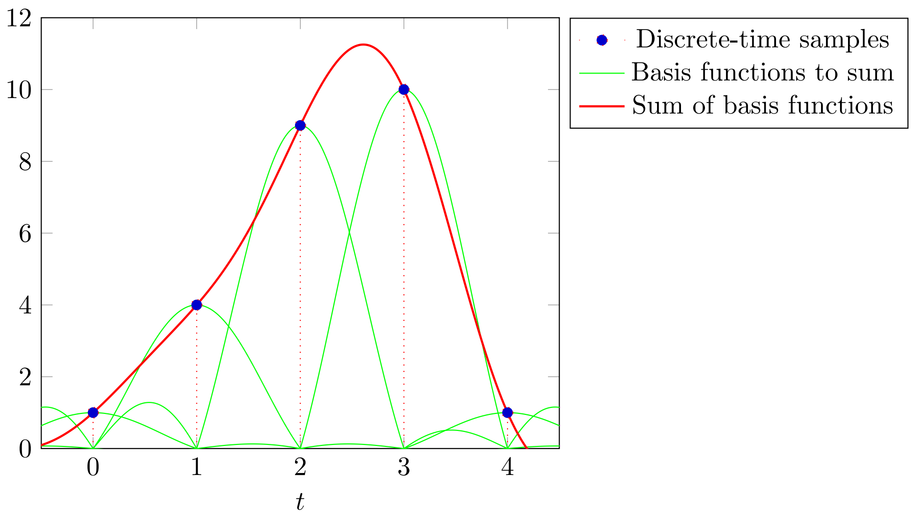
so $\phi(t)$ satisfies the requirements of a basis function. Plotting this function with $\Delta = 1$, we obtain



Although we will not prove it explicitly, it is not hard to show that $\phi(t)$ is also smooth, as we had hoped.

Now, we can try using this basis function to construct interpolations. Our aim will be to produce interpolations that smoothly move between our sample points. For an arbitrary set of sample points with $\Delta = 1$, we obtain

[1]It is straightforward to see that this value will indeed remove the discontinuity.



Observe that the interpolation passes through all the sample points (as we showed it must) much more naturally than did either of our previous two interpolations.

7 Polynomial Interpolation

Recall from EE16A that we could construct a polynomial of minimum degree that passed through a given set of points, if one existed. It is a natural question to ask whether this method can be adapted into another way of constructing a smooth interpolation of a discrete-time signal (since all polynomials are clearly smooth). Imagine that we had n sample points, from $t = 0$ to $t = (n-1)\Delta$, and wished to construct a polynomial of degree d passing through all these n sample points. Let such a polynomial be

$$p(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_d t^d,$$

and let our sample points be

$$(0, y_0), (\Delta, y_1), \ldots, ((n-1)\Delta, y_{n-1}).$$

For convenience, let $x_i = \Delta i$ for integer $0 \leq i < n$. Since $p(x_i) = y_i$ for all our sample points, we obtain the system of linear equations

$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_d x_0^d &= y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_d x_1^d &= y_1 \\ &\vdots \\ a_0 + a_1 x_{n-1} + a_2 x_{n-1}^2 + \ldots + a_d x_{n-1}^d &= y_{n-1}, \end{aligned}$$

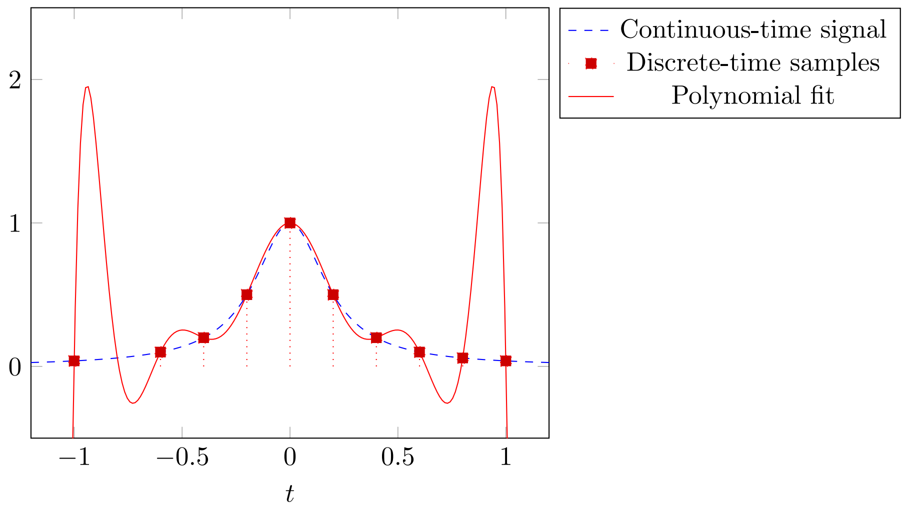
which can be rewritten in matrix form as

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^d \\ 1 & x_1 & x_1^2 & \dots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^d \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}.$$

Clearly, for a unique solution to always exist, the matrix on the left-hand-side must be square, so $n = d + 1$. In that case, that matrix is known as a *Vandermonde matrix.* An important property of Vandermonde matrices is that, assuming all the x_i are distinct, they will always be invertible, so we can always solve for the a_i, as we expected.

The question remains - what does such a polynomial look like, and how does it compare to our earlier methods of interpolation (specifically, sinc interpolation). Unfortunately, while polynomial interpolation can work in some cases, it turns out that in general it performs poorly, particularly near the edges of the samples, where the interpolated polynomial starts to rise and fall rapidly to pass through all the sample points. This behavior is known as *Runge's phenomenon*, though we will not study it in great detail for this course.

Nonetheless, we can see an example of this in the following polynomial interpolation of a discrete-time signal with $\Delta = 0.2$:



The polynomial fit, though it passes through the points, is extremely poor near the edges, oscillating violently. As a consequence, we will typically use sinc interpolation when we require a smooth interpolation, unless there is a good reason to do otherwise.