Interpolation

Interpolation

- Interpolation is not a branch of mathematic but a collection of techniques useful for solving computer graphics problems

- Basically an interpolant is a way of changing one number into another.

- For example to change 2 into 4 we simply add 2 which is not particularly useful.

- The real function of an interpolant is to change one number into another in some number of equal steps

Example

- If we wish to change 2 into 4 using 10 equal steps we just add 0.2
- 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4 3.6 3.8 4.0
- We could then use these values to drive an animated parameter

- In order to repeat this interpolant for different number we require a formula to work for all given values
- We can also look at how we can control the speed of change as the above case uses linear steps and we quite often need to ease in or ease out the movement / values

Linear Interpolation

- A Linear interpolant generates equal spacing between the interpolated values for equal changes in the interpolating parameter.

- In the previous example the increment 0.2 is calculated by subtracting the first value from the second and dividing by the result by 10 (4-2)/10=0.2

- We can formally define this as follows

Linear Interpolation

- We can use linear interpolation to blend between two values using a real scalar value which ranges from 0 - 1
- The basic formula given two values **a** and **b** and the real scalar *t* we get

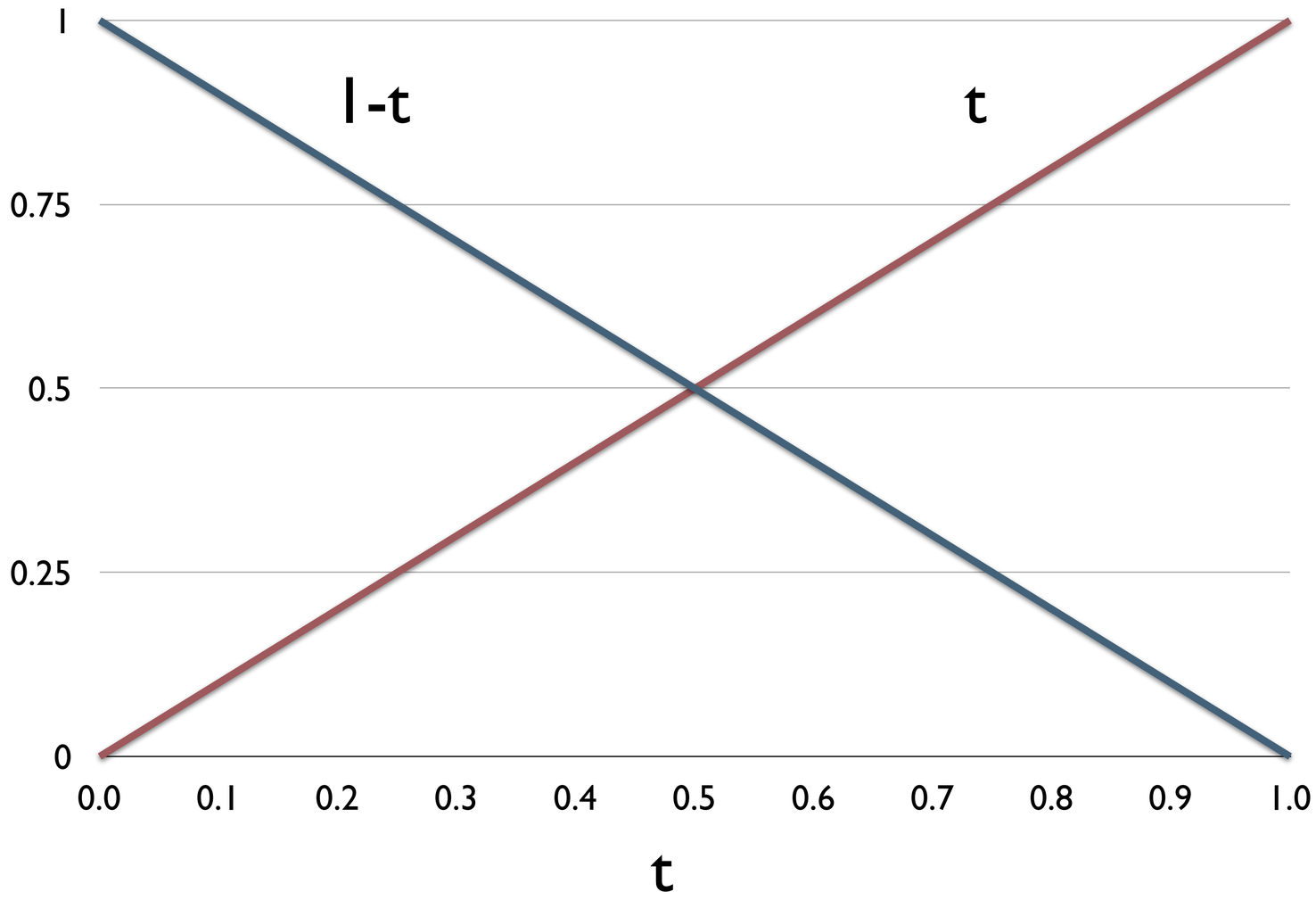
$$p = a + (b - a) * t;$$

- This can be written in code as

```
1  def Lerp( _a, _b, _t) :
2    return _a+(_b-_a)*t
```

Linear Interpolation

- The graphs of (1-t) and t over the range 0-1



Linear Interpolation

- Linear interpolation can be extended to any number of values
- For example a 3D point would use the piecewise interpolant of x y and z as shown below
- This is know as tri-linear interpolation

$$x = x_1(x_2 - x_1) * t$$

$$y = y_1(y_2 - y_1) * t$$

$$z = z_1(z_2 - z_1) * t$$

Linear Interpolation

- The terms (1-t) and t will always sum to unity over the range 0-1.
- This is important and will inform other interpolation values
- The interpolant may also be expressed in matrix form thus

$$n = \begin{bmatrix} t & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$$

Non-linear Interpolation

- A Linear interpolant ensures that equal steps in the parameter t give rise to equal steps in the interpolated values.
- However it is often required that equal steps in t give rise to unequal steps in the interpolated values.
- We can achieve this using a variety of mathematical techniques.
- For example, we could use trigonometric functions or polynomials.

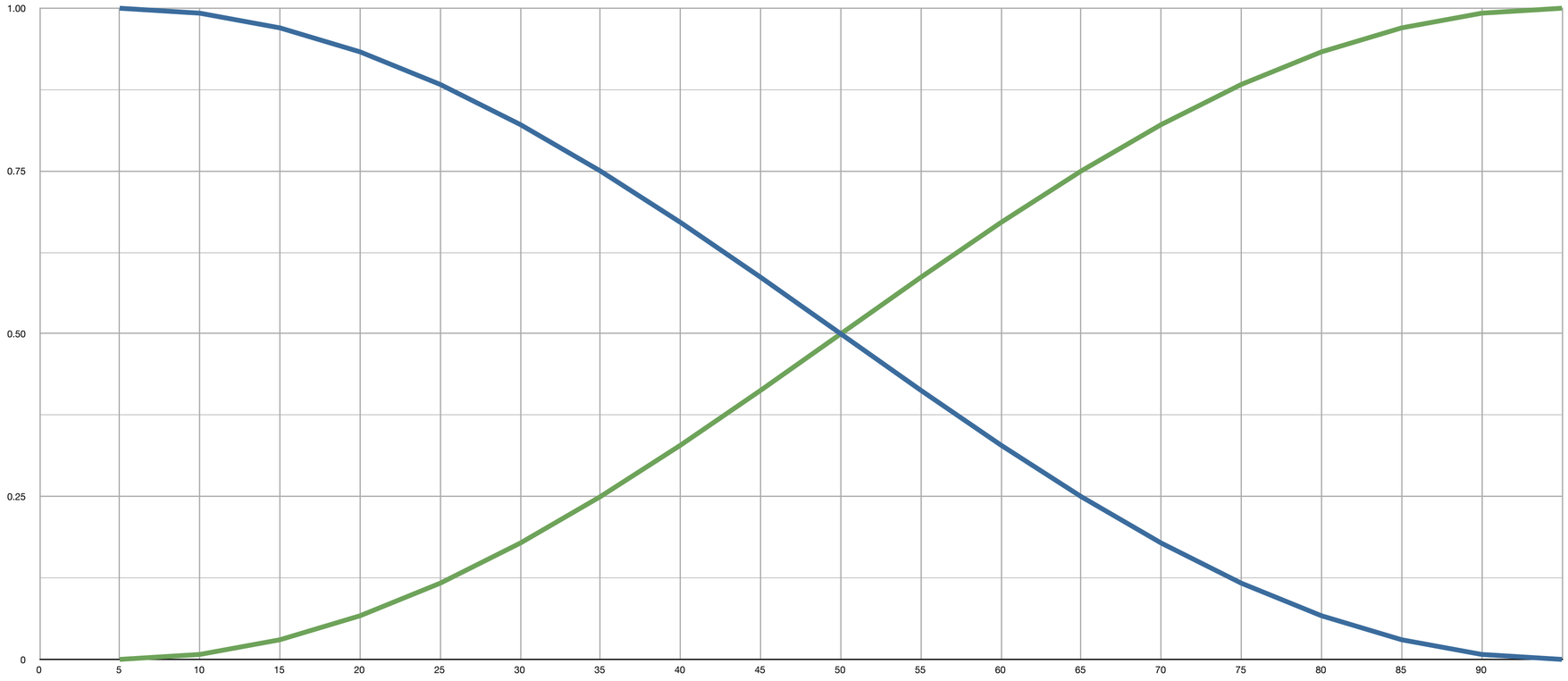
Trigonometric Interpolation

- From the basic trigonometric ratios we know that $\sin^2 \beta + \cos^2 \beta = 1$
- This satisfies one of the requirements of an interpolant that the terms must sum to 1
- If β varies between 0 and $\frac{\pi}{2}$ $\cos^2 \beta$ varies between 1 and 0 and $\sin^2 \beta$ varies between 0 and 1 which can be used to modify the interpolated values

$$n = n_1 \cos^2 t + n_2 \sin^2 t \qquad \left[0 \leq t \leq \tfrac{\pi}{2}\right]$$

Trigonometric Interpolation

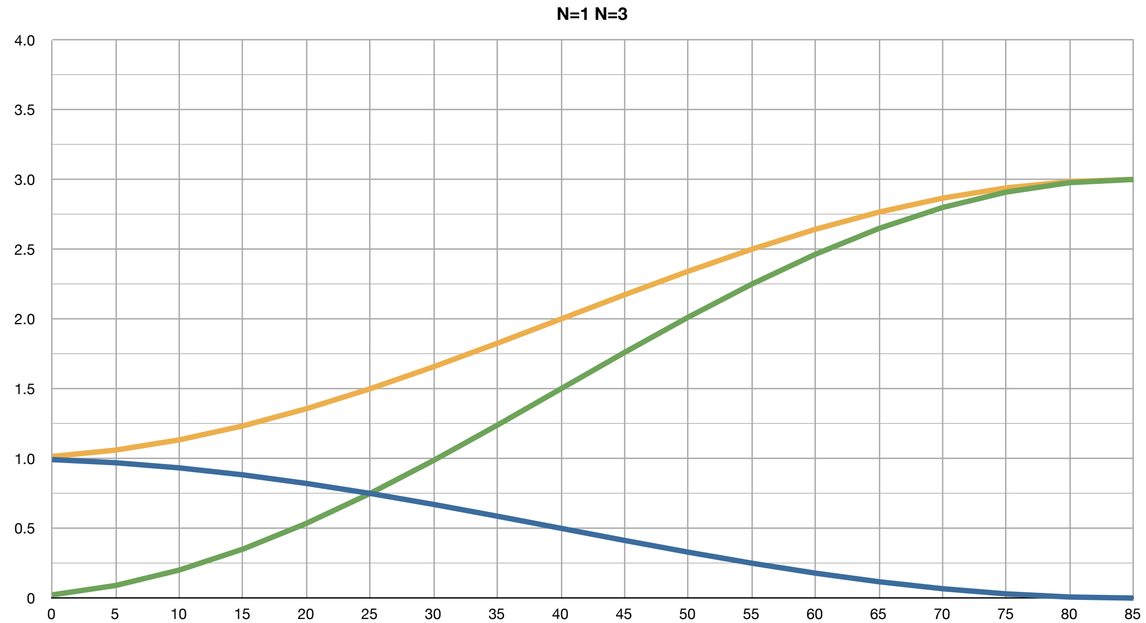
The curves for $\cos^2\beta$ and $sin^2\beta$



Trigonometric Interpolation

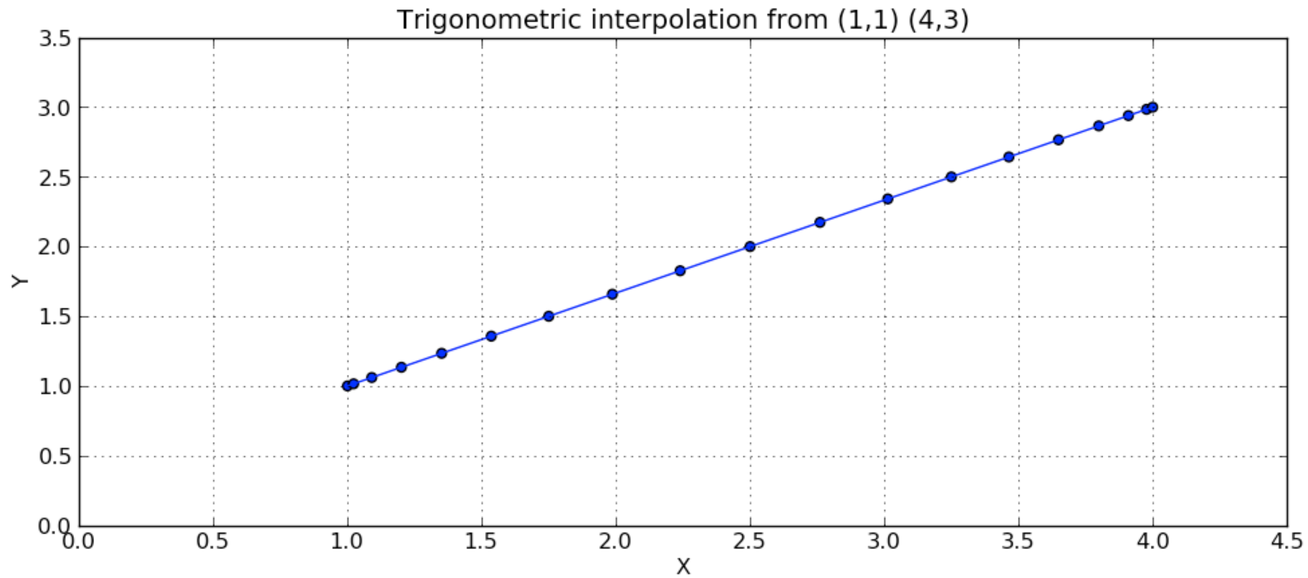
- If we make n1=1 and n2=3 we can plot the curves as shown

- If we then apply the interpolant to two 2D points we get the following



Trigonometric Interpolation

- Here we see a straight line interpolant
- However the distribution of the points is non-linear
- Equal steps in t give rise to unequal distances



Trigonometric Interpolation

- The main problem with this approach is that it is impossible to change the nature of the curve
- It is a sinusoid, and its slope is determined by the interpolated values.
- One way of gaining control over the interpolated curves is to use a polynomial as we shall see next.

Cubic Interpolation

- In cubic interpolation we need to develop a cubic function to do our blending

- In mathematics a cubic function is one of the form

$$f(x) = ax^3 + bx^2 + cx + d$$

- Applying this to our interpolant we get

$$v_1 = at^3 + bt^2 + ct + d$$

- or in Matrix form

$$n = [v_1 v_2] \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$$

Cubic Interpolation

- The task is to find the values of the constants associated with the polynomials v1 and v2
- The requirements are

1. The cubic function v2 must grow from 0 to 1 for $0 \leq t \leq 1$
2. The slope at point t must equal the slope at the point (1-t) this ensures the slope symmetry over the range of the function
3. The value v2 at any point t must also produce (1-v2) at (1-t) this ensures curve symmetry

Cubic Interpolation

- To satisfy the first requirement :

$$v_2 = at^3 + bt^2 + ct + d$$

- and when t=0, v2=0 and d=0

- Similarly when t=1, v2=a+b+c

Cubic Interpolation

- To satisfy the second requirement, we differentiate v2 to obtain the slope

$$\frac{dv_2}{dt} = 3at^2 + 2bt + c = 3a(1-t)^2 + 2b(1-t) + c$$

- and equating constants we discover

$$c = 0 \text{ and } 0 = 3a + 2b$$

Cubic Interpolation

- To satisfy the third requirement

$$a^3 + bt^2 = 1 - \left[a(1-t)^3 + b(1-t)^2\right]$$

- where we discover 1= a+b But 0=3a+2b therfore a=2 and b=3, therefore

$$v_2 = -2t^3 + 3t^2$$

Cubic Interpolation

- To find the the curve's mirror curve, which starts at 1 and collapses to 0 as t moves from 0 to 1 we subtract the previous equation from 1

$$v_1 = 2t^3 - 3t^2 + 1$$

- Therefore the polynomials are

$$v_1 = 2t^3 - 3t^2 + 1$$

$$v_2 = -2t^3 + 3t^2$$

Cubic Interpolation

- These can be used as interpolants as follows

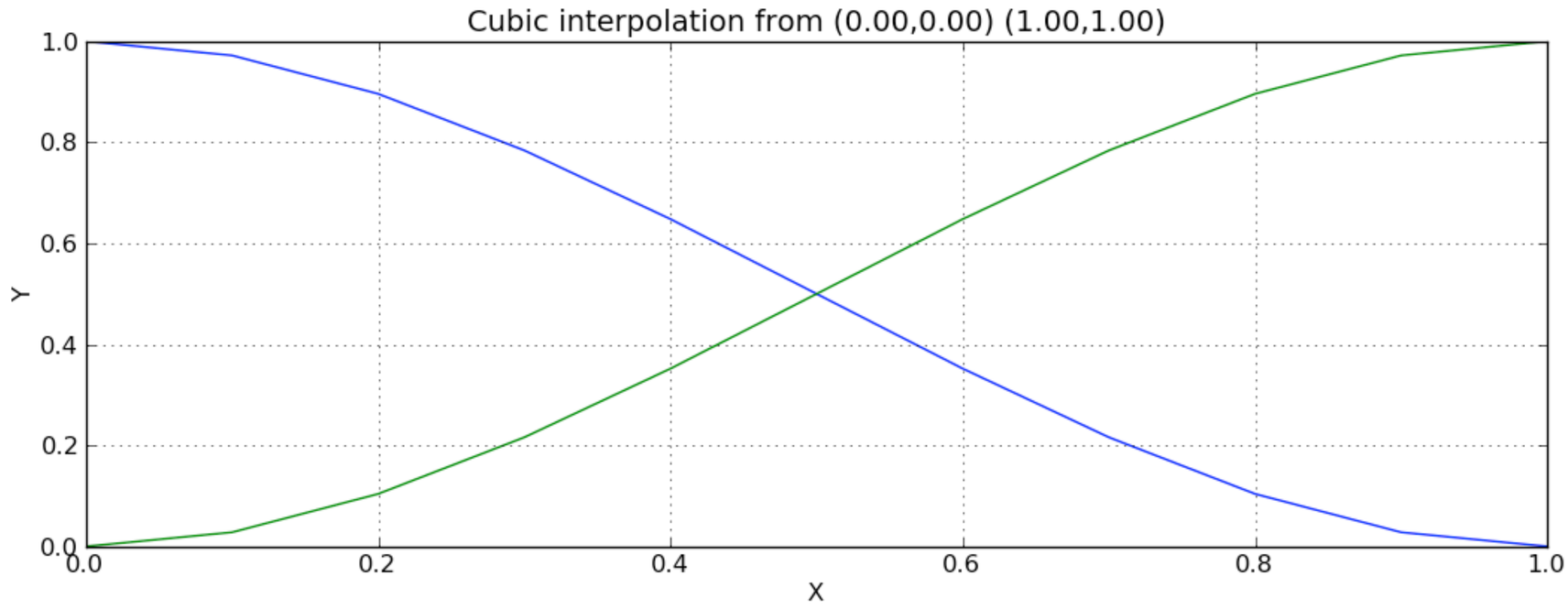
$$n = v_1 n_1 + v_2 n_2$$

$$n = \begin{bmatrix} 2t^3 - 3t^2 + 1 & -2t^3 + 3t^2 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$$

$$n = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 \\ -3 & 3 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$$

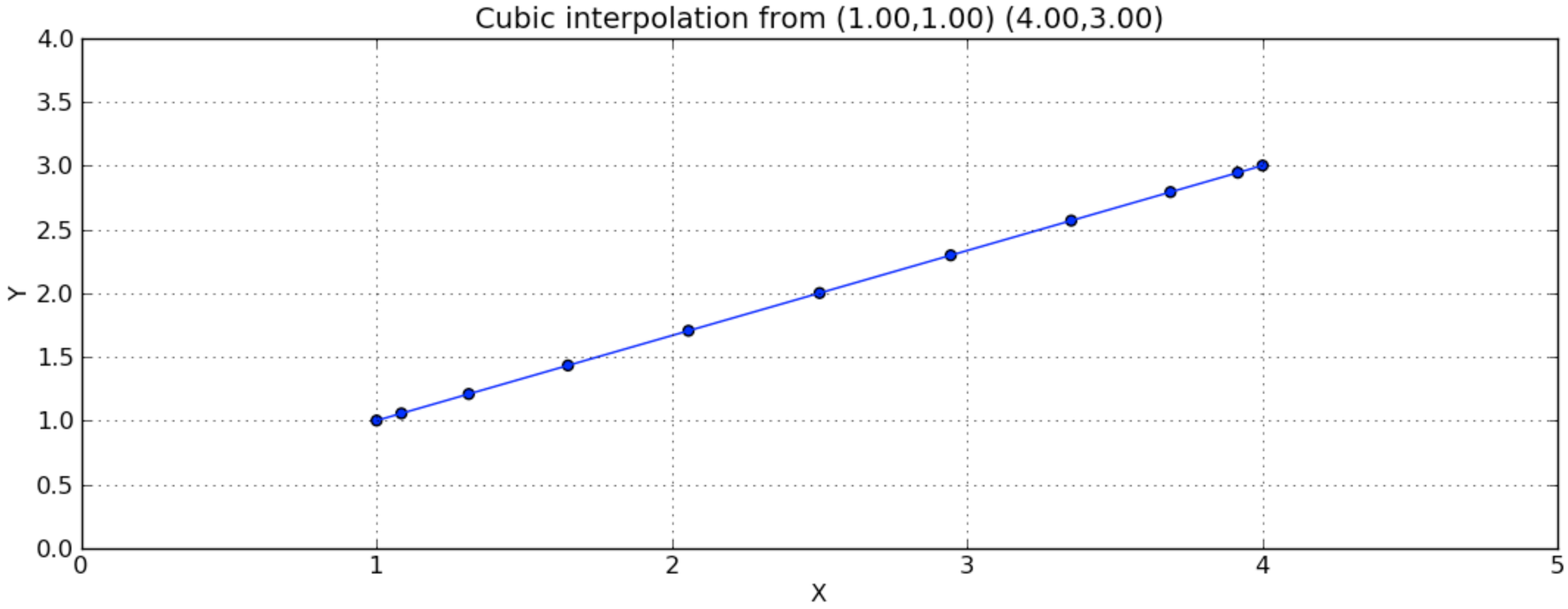
Cubic Interpolation

- Plotting v1 and v2 we get



Cubic Interpolation

- applying to two points $n = v_1 n_1 + v_2 n_2$



References

- Vince, John (2010). Mathematics for Computer Graphics. 2nd. ed. London: Springr Verlag.