number of points	2	3	4	5	6	7	8	9	10
$p_{n,\max}$	1	1.25	1.667	1.799	1.989	2.083	2.203	2.275	2.362

Table 2: Maximum overshoot fraction, $p_{n,\max}$, given the number of points in the polynomial interpolation (to 4 digits rounded up).

2.3 Weight Initialization

Weight initialization is a huge concern for neural networks (Yam and Chow (2000)). In this paper, the weights within a link are initialized such that $F(x)$ (Equation) is a line across the link, the equation is given as

$$\omega_i = \left(\frac{x_i - r_{\min}}{r_{\max} - r_{\min}} \right) \omega_{\mathrm{a}} + \omega_{\mathrm{a}} \,, \tag{3}$$

where w_a is chosen with a uniform random distribution in the range $[-1, 1]$.

2.4 Choosing Ranges $r_{\mathbf{min}}$ and $r_{\mathbf{max}}$

Choosing proper ranges for the links is key to getting good solutions. The weights used in the Lagrange polynomial interpolation do not mark the limits of the polynomial value except in the linear case. Figure 8 shows 5 Lagrange polynomials with maximum overshoot for weights within the desired range - note that only in the linear case is the range limited by the values of the weights. Instead, if the weights are in the range $[-\omega_{\max}, \omega_{\max}]$ then a given choice of weights will produce a maximum overshoot $p_{n,\max}\omega_{\max}$ where values of $p_{n,\max}$ are given in Table 2. For a given input, the maximum possible output would be $p_{n,\max}\omega_{\max}$ which could then be the input to the next link. The input range for each link should be set to $[-p_{n,\max}\omega_{\max}, p_{n,\max}\omega_{\max}]$ to account for these overshoots. One potential consequence of this choice of range is input value decay. Consider a deep network with only one unit per layer and one link between units. Each layer is initialized using Equation () with $w_a = 1$ and input range $[-p_{n,\max}, p_{n,\max}]$ Suppose the input value is x_{in} then as the input value passes through each layer it will be compressed if $r_{max} = p_{n,\max} > \omega_{max}$ by the ratio ω_{max}/r_{max}. After passing through n layers, the output signal will be $(\omega_{max}/r_{max})^n \, x_{in}$. As a consequence, the output from each layer is pushed towards the value 0 which means fewer and fewer of the network sub links are used. Fortunately, two things can occur to help the situation: (1) if a discontinuity is at the origin rapid learning can still occur since if a signal is either side of 0 it will adjust disconnected weights; (2) as the network is trained, more and more of the sub links are used. It would seem randomly initializing the weights could alleviate this problem, however, we've found that symmetric initialization as in Equation () works better. In this paper we use $r_{max} = -r_{min} = p_{n,\max}$ where $p_{n,\max}$ is provided in Table 2 which gives the maximum overshoot for the polynomials when the weights are constrained to be between 1 and -1.