Convolution of Discrete-Time Signals

The *convolution sum* is :

$$y(kT) = x(kT) * y(kT) = \sum_{i=-\infty}^{\infty} x(iT)v(kT - iT)$$

If $x(kT) = 0$ and $v(kT) = 0$ for all $k<0$, then $x(iT) = 0$ for all $i<0$ and $v(kT - iT)$ for all integers $(k - i) < 0$ (or $k < i$).

Then,

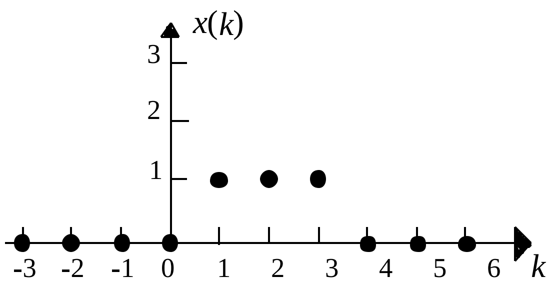
$$y(kT) = x(kT) * v(kT) = \begin{cases} 0 \quad , & k = -1, -2, \ldots \\ \sum_{i=0}^{k} x(iT)v(kT - iT), & k = 0, 1, 2, \ldots \end{cases}$$

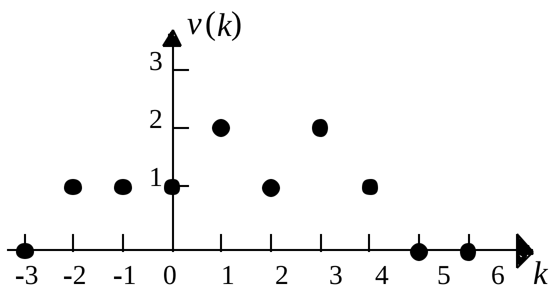
⇒ Any two signals that are 0 ∀ integers $k < 0$ can be convolved.

To compute the convolution sum,

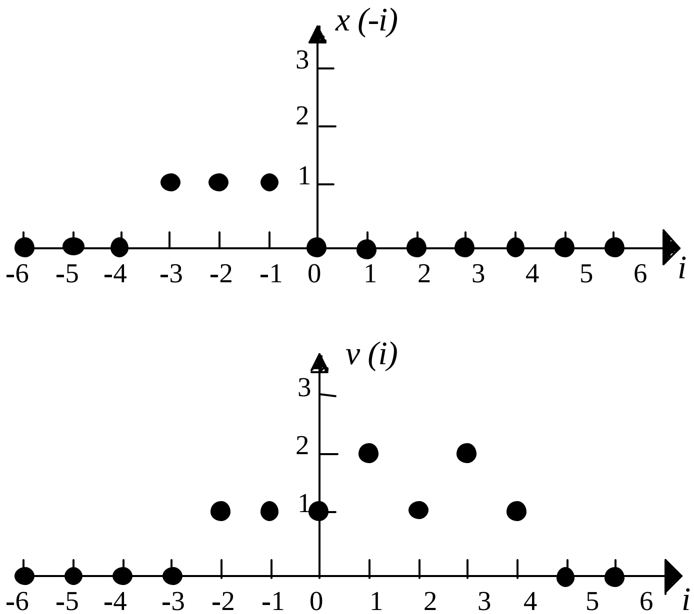
- fold $v(iT)$ to be $v(-iT)$
- shift $v(-iT)$ by kT to $v(kT - iT)$
- if $k > 0$, $v(kT - iT)$ is kT shift right of $v(-iT)$ and if $k < 0$, $v(kT - iT)$ is left shift

Convolution Example



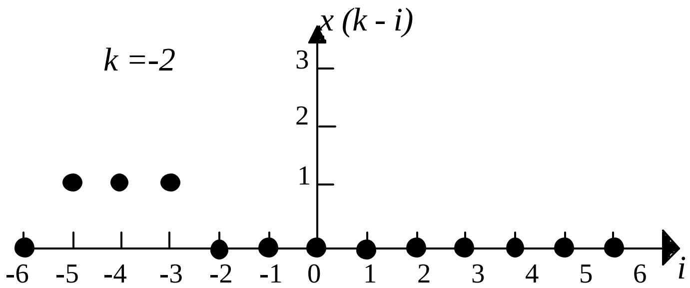


In this example, we wish to convolve the two functions, $x(kT)$ and $v(kT)$ shown on the left. To proceed, "fold" one of the functions about the vertical axis. It is best to fold the least complex function. In this case, the function $x(kT)$ is folded about the vertical axis.

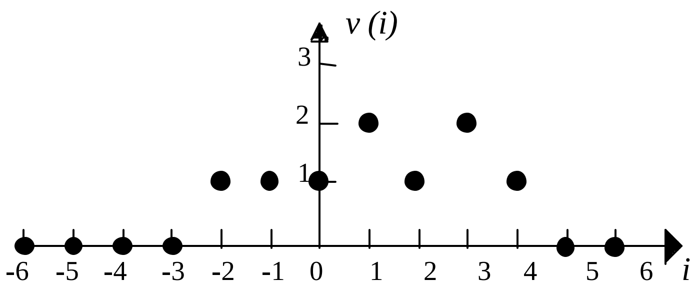


This is the folded version of the function $x(kT)$ or $x(-iT)$. Now we "slide" the function $x(-iT)$ over $v(iT)$ and see where the "over laps" occur. At overlapping points, we multiply the value of $x(-iT)$ and $v(iT)$. We then sum up the results of the multiplication. Notice that we have folded $x(kT)$ instead of $v(kT)$. It does not matter which one we fold.

Also note that in order to get a good result, we must shift the function $x(-iT)$ by an amount $kT = -2$ so that the all of $x(-iT)$ eventually slides over $v(iT)$.

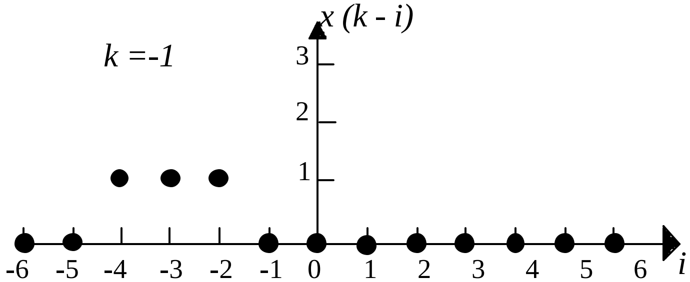


At this point, there is no overlap. We now create a table to help us tabulate the convolution.

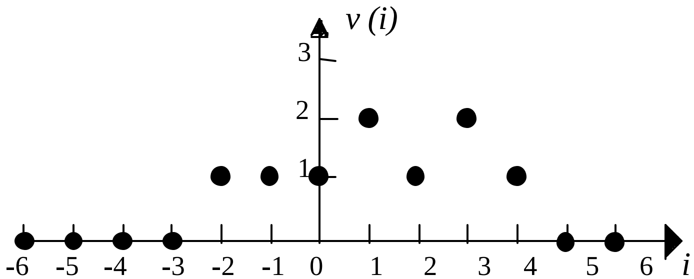


i *k* shift	-3	-2	-1	0	1	2	3	4	5	6	Sum = *y*(*kT*)
	v(*iT*)										
-2	0	0	0	0	0	0	0	0	0	0	0
-1											
0											
1											
2											
3											
4											
5											
6											
7											

8											



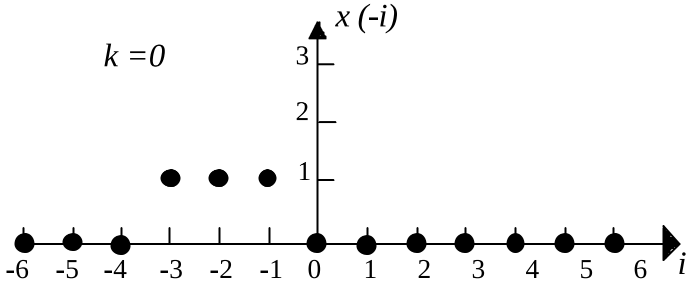
At this point, there is overlap at $i = -2$.



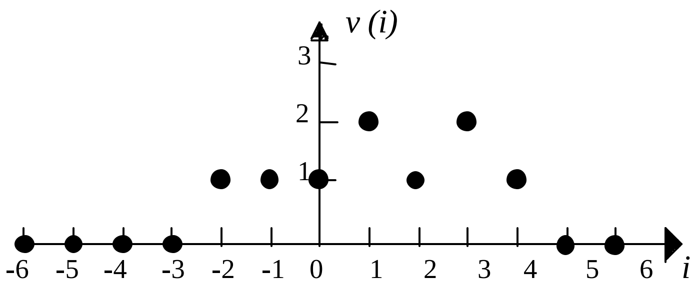
v(iT)

i *k* shift	-3	-2	-1	0	1	2	3	4	5	6	Sum = *y(kT)*
-2	0	0	0	0	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0	0	0	0	1
0											
1											
2											
3											
4											
5											
6											

7											
8											



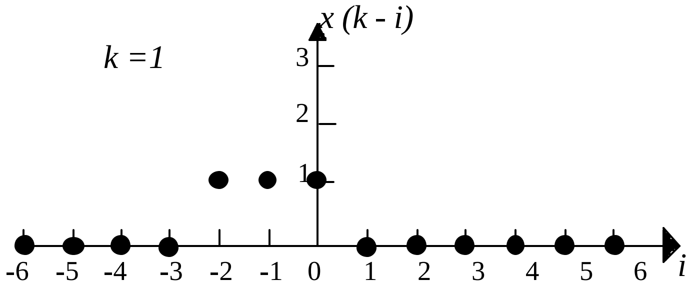
At this point, there is overlap at *i* = -2 and -1.



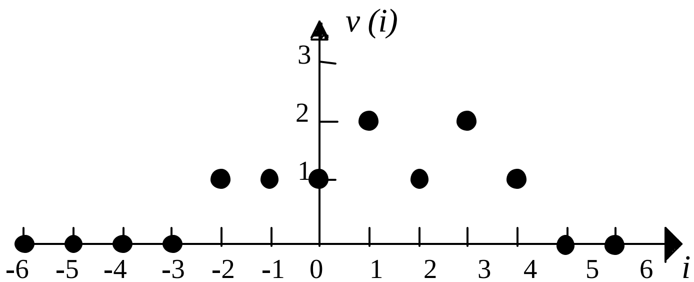
v(iT)

i *k* shift	-3	-2	-1	0	1	2	3	4	5	6	Sum = *y(kT)*
-2	0	0	0	0	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	0	2
1											
2											
3											
4											
5											

6											
7											
8											



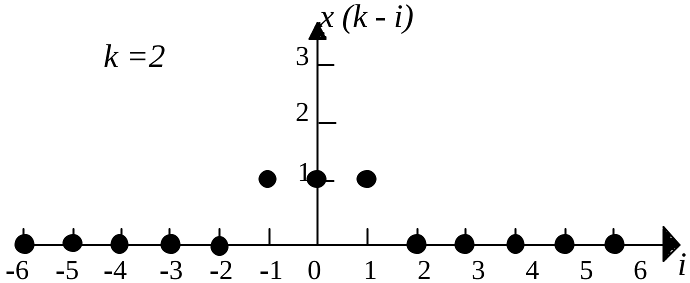
At this point, there are overlaps at *i* = -2, -1, and 0.



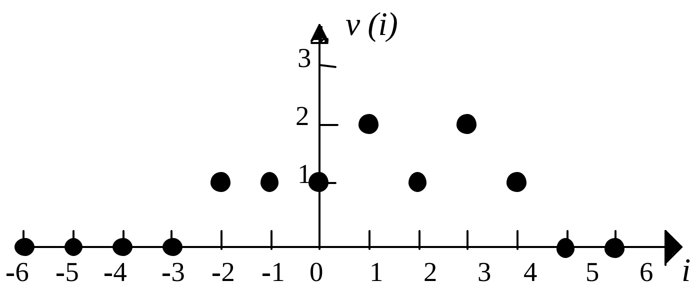
i / *k* shift	*v(iT)* -3	-2	-1	0	1	2	3	4	5	6	Sum = *y(kT)*
-2	0	0	0	0	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	0	2
1	0	1	1	1	0	0	0	0	0	0	3
2											
3											
4											

5											
6											
7											
8											

k =2



At this point, there are overlaps at *i* = -1, 0 and 1.

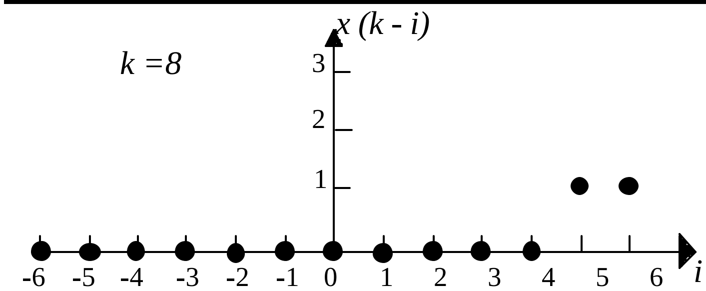


v(iT)

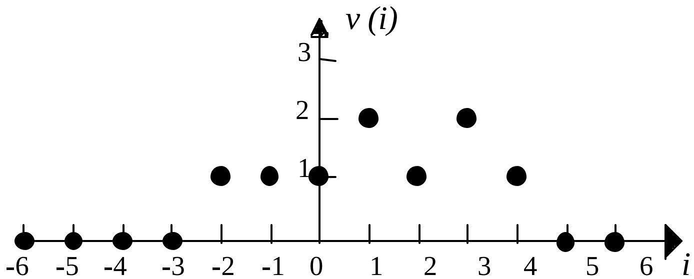
i *k* shift	-3	-2	-1	0	1	2	3	4	5	6	Sum = *y(kT)*
-2	0	0	0	0	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	0	2
1	0	1	1	1	0	0	0	0	0	0	3
2	0	0	1	1	2	0	0	0	0	0	4
3											

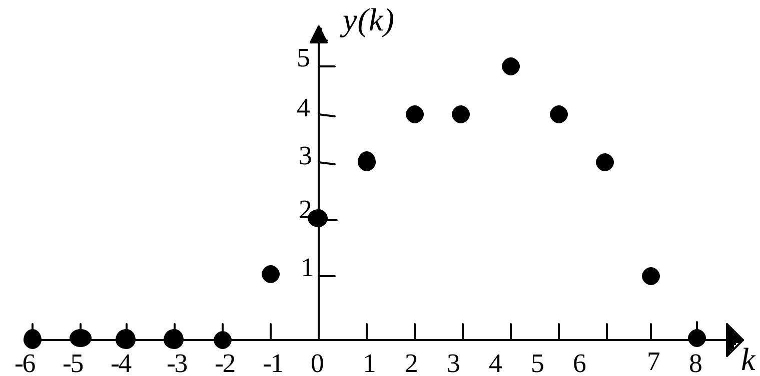
4											
5											
6											
7											
8											

k =8



This is the point where the folded $x(iT)$ no longer overlaps $v(iT)$. The result is shown below with the table of results.





v(iT)

i / *k* shift	-3	-2	-1	0	1	2	3	4	5	6	Sum = *y(kT)*
-2	0	0	0	0	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	0	2
1	0	1	1	1	0	0	0	0	0	0	3
2	0	0	1	1	2	0	0	0	0	0	4
3	0	0	0	1	2	1	0	0	0	0	4

4	0	0	0	0	2	1	2	0	0	0	5
5	0	0	0	0	0	1	2	1	0	0	4
6	0	0	0	0	0	0	2	1	0	0	3
7	0	0	0	0	0	0	0	1	0	0	1
8	0	0	0	0	0	0	0	0	0	0	0