Problem 2 Overview C: The inverse Fast Wavelet transform
(Data reconstruction using our wavelet coefficients)

1. Target vector reconstruction using additions, subtractions, and *up-sampling*

Suppose we have a new target vector b with a word length of $L = 8$. We would like to reconstruct a target vector b as a linear combo of the Haar wavelet basis:

$$b = c_0 \overrightarrow{w_0} + c_1 \overrightarrow{w_1} + c_2 \overrightarrow{w_2} + c_3 \overrightarrow{w_3}$$
$$+ c_4 \overrightarrow{w_4} + c_5 \overrightarrow{w_5} + c_6 \overrightarrow{w_6} + c_7 \overrightarrow{w_7}$$

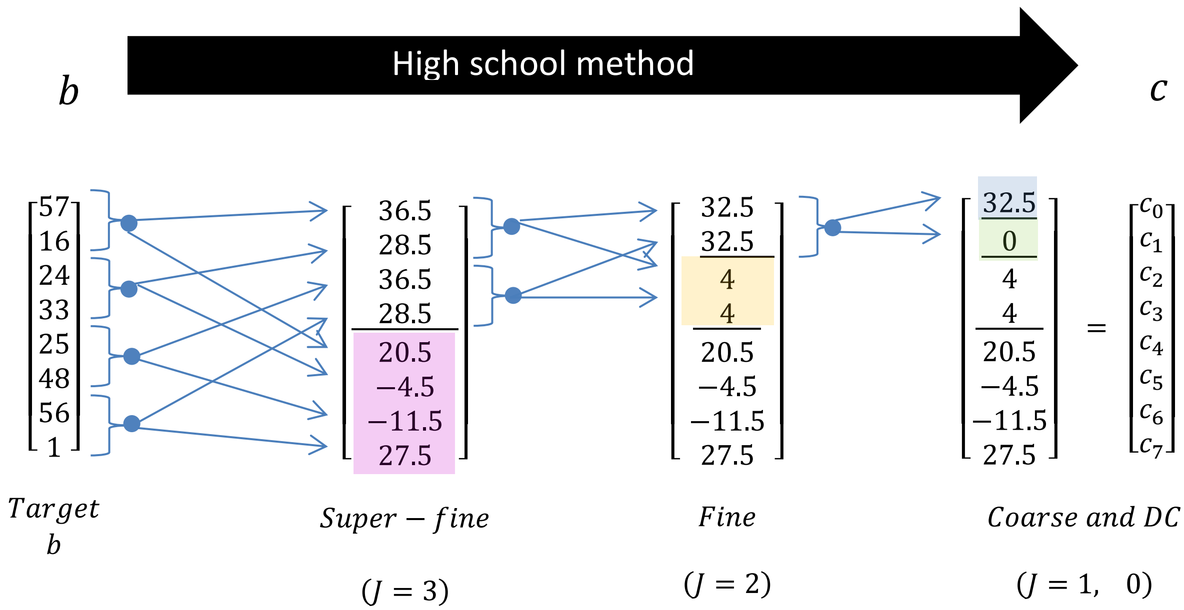
In matrix form, this equation becomes :

$$\begin{bmatrix} 57 \\ 16 \\ 24 \\ 33 \\ 25 \\ 48 \\ 56 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}$$

$$b \quad = \quad W \quad c$$

In a previous overview file, we learned that the wavelet coefficients c_0 thru c_7 can be solved by using the high-school-style, "top-bin addition / bottom-bin subtraction" method. Let's solve for them now (see Figure 1 for details)

** Note: Once we've found the super-fine ($J = 3$) and the fine ($J = 2$) wavelet coefficients, we <u>don't want to mess with it !!</u>



Where the wavelet coefficients are:

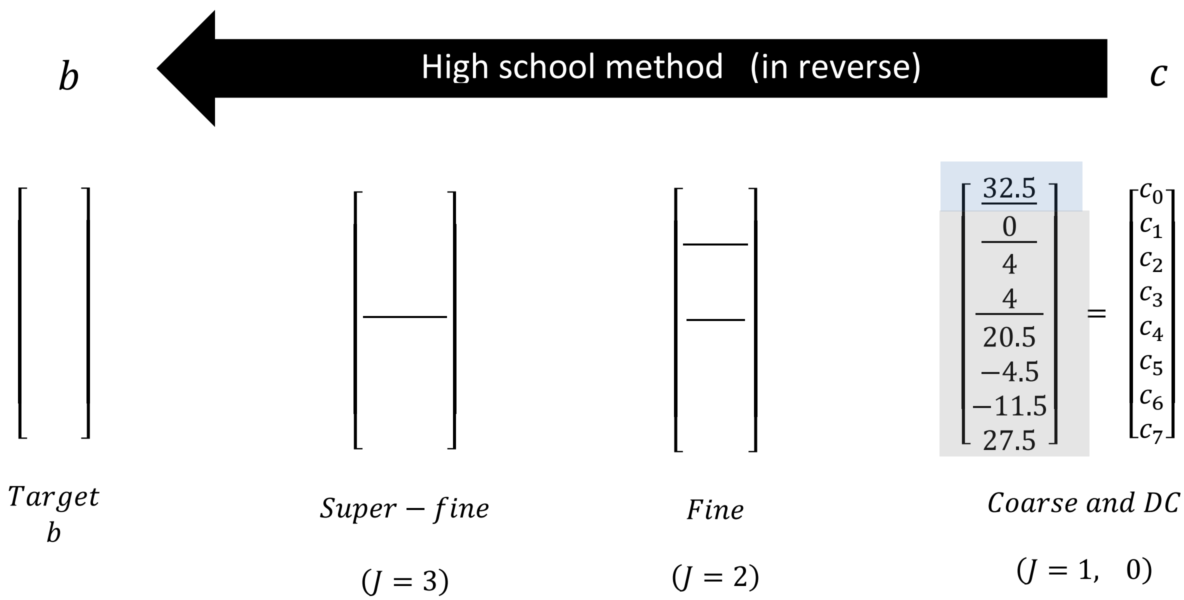
DC value $(J = 0)$:	c_0	*Fine* $(J = 2)$:	$c_2, \ c_3$
Coarse $(J = 1)$:	c_1	*Super fine* $(J = 3)$:	$c_4, \ c_5, \ c_6, \ c_7$

<u>Figure 1</u>: Using the high-school method to solve for the wavelet coefficients c_0 thru c_7

<u>Question</u>: Once you have the coefficients c_n, is there a high-school method in which we can reconstruct b in the *<u>reverse direction</u>* ???

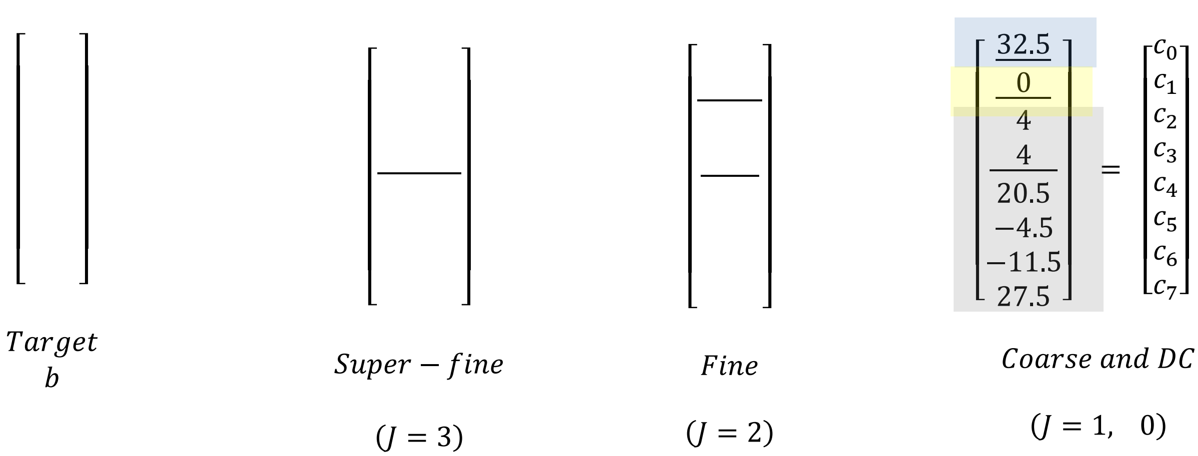
1a) Reverse-engineer our target vector using additions / subtractions

Let's begin by re-examining the butterfly diagram in Figure 1. If we wanted to reverse the calculations, all we have to do is to…… you guessed it: Reverse the averages and ½ differences level-by-level !! We will now try to recreate Figure 1, but this time, we'll start with the wavelet coefficients c_n.

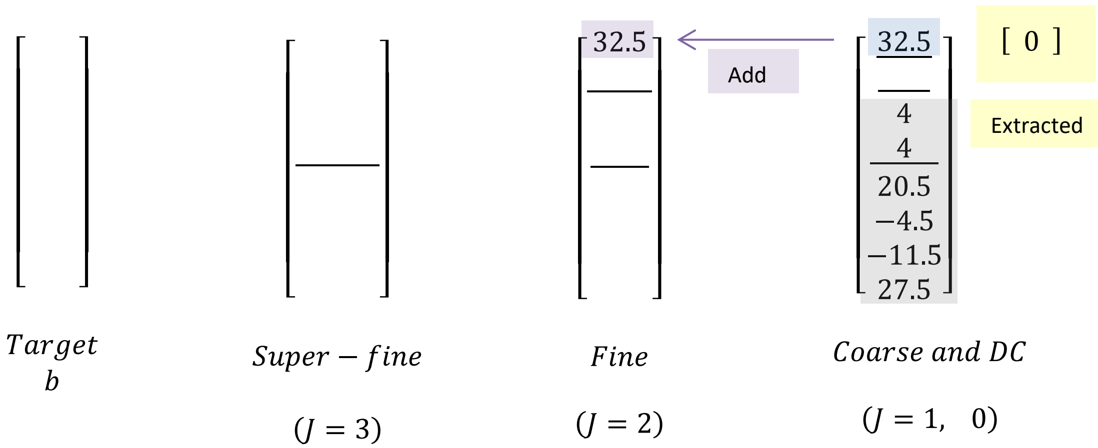


For starters, we see that we have a *single isolated top bunk* with a value of 32.5 (shaded in blue). We also see that the rest of the bunks below are shaded in gray.

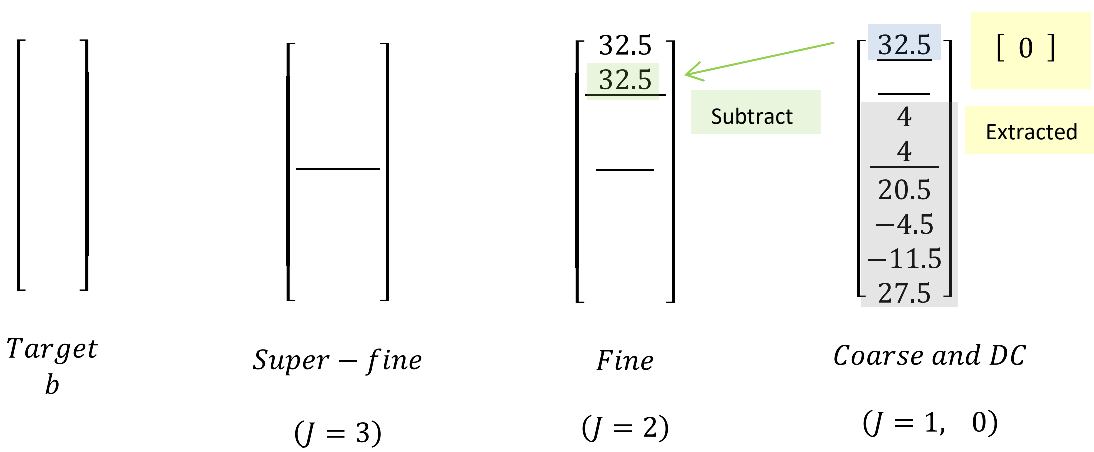
Step #1a: If the top-most bunk (shaded in blue) was a *single-membered bunk*, we will first "extract" a *single-membered bunk* from the top of the gray stack and color it yellow.



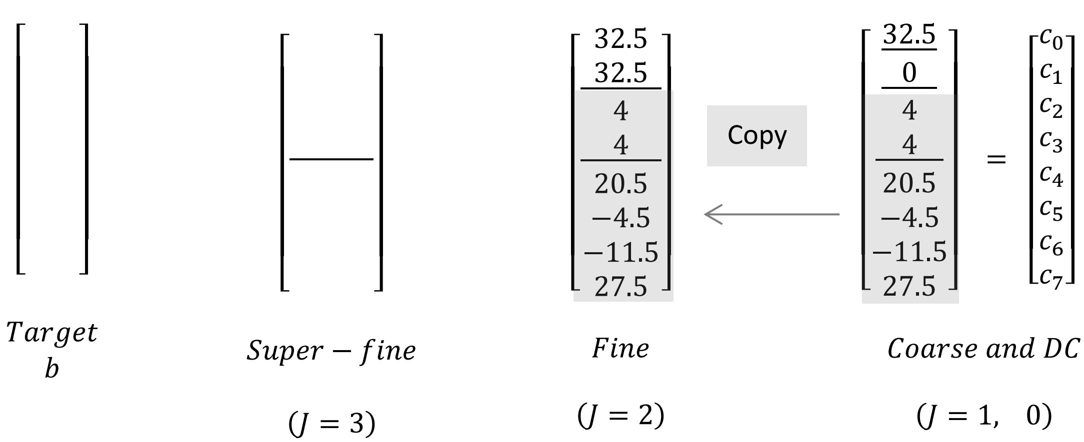
__*Step #1b*__: Now, we add the blue + yellow values and insert the answer in the top bunk of our new “fine ($J = 2$)” stack. This effectively reverses the forward average sum operations we did earlier !



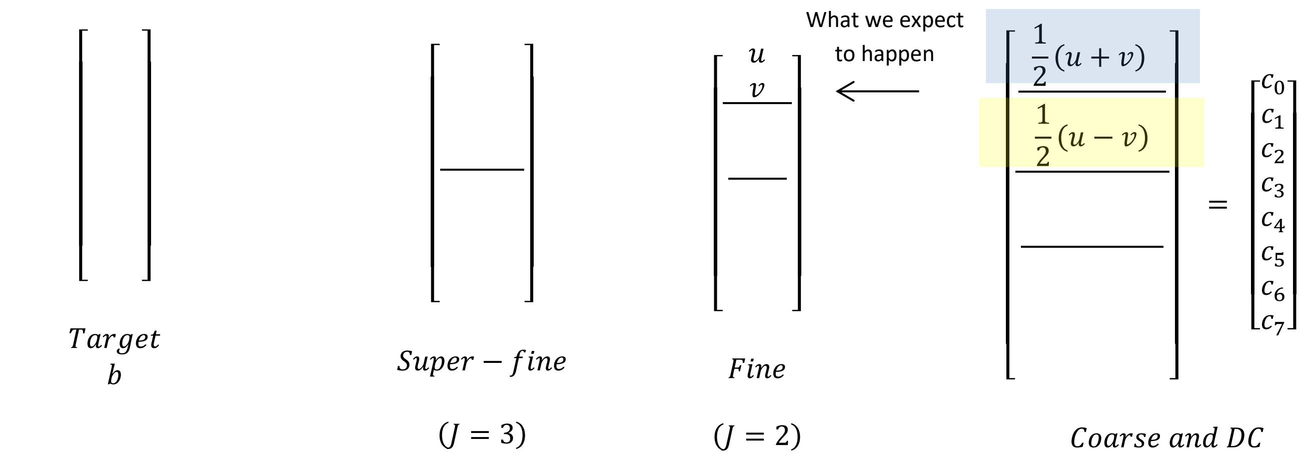
__*Step #1c*__: Then, we subtract the blue + yellow values and insert the answer in the top bunk of our new “fine ($J = 2$)” stack. This effectively reverses the forward ½ difference operation.



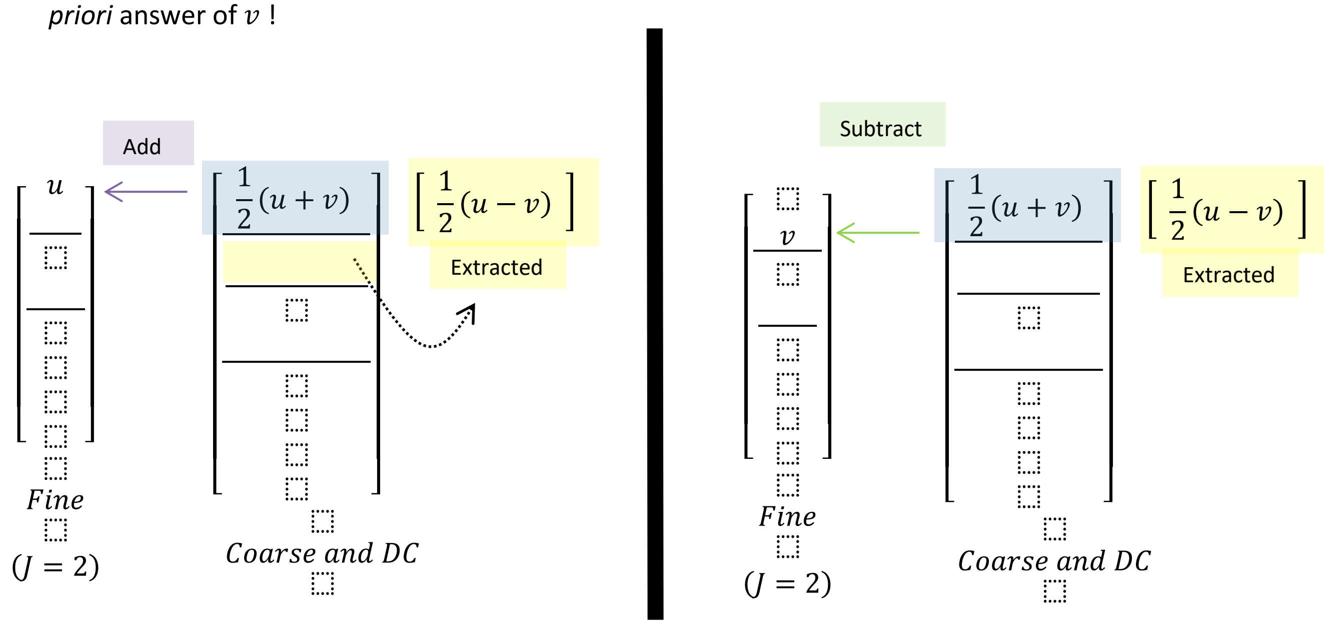
__*Step #1d*__: Finally, we copy all left-over gray bunk values over to the lower “basement” of the new “fine ($J = 2$)” stacks. These represent the hard-earned wavelet coefficients that we didn’t want to mess with when were calculating the c’s in the forward direction.



__The reason why it works__: To see what really happened in Steps 1b and 1c, we will reexamine the math by substituting the matrix values with dummy variables . For instance, you know *a priori* that the values within the "fine ($J = 2$)" and the "coarse & DC ($J = 1 \; and \; 0$)" stacks must look like this:



Hence, if we add the blue and yellow values together and insert the answer into the top bunk of the "fine ($J = 2$)" stack, it will match the *a priori* answer of u. Similarly, if we subtract the blue and yellow values and place the answer in the 2nd slot in the top bunk of "fine ($J = 2$)" stack, it will match the *a* *priori* answer of v !



Getting ready for the next step: Now, we will focus on the transition from the "fine ($J = 2$)" stack to the "super-fine ($J = 3$)" stack. We notice that the top-most bunk (shaded in blue) is now a *2-membered bunk*. Let's colored that blue, and color the rest in gray.

$$
\underset{\substack{Target \\ b}}{\begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}}
\qquad
\underset{\substack{Super-fine \\ \\ (J=3)}}{\left[\begin{array}{c} \\ \\ \hline \\ \\ \end{array}\right]}
\qquad
\underset{\substack{Fine \\ \\ (J=2)}}{\left[\begin{array}{c} 32.5 \\ 32.5 \\ \hline 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{array}\right]}
\qquad
\underset{\substack{Coarse\ and\ DC \\ \\ (J=1,\quad 0)}}{\left[\begin{array}{c} 32.5 \\ \hline 0 \\ \hline 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{array}\right]}
=
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}
$$

Step #2a: To begin, If the top-most bunk (shaded in blue) was a *2-membered bunk*, we will "extract" a *2-membered bunk* from the top of the gray stack and color it yellow.

$$
\underset{\substack{Target \\ b}}{\begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}}
\qquad
\underset{\substack{Super-fine \\ \\ (J=3)}}{\left[\begin{array}{c} \\ \\ \hline \\ \\ \end{array}\right]}
\qquad
\underset{\substack{Fine \\ \\ (J=2)}}{\left[\begin{array}{c} 32.5 \\ 32.5 \\ \hline 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{array}\right]}
\qquad
\underset{\substack{Coarse\ and\ DC \\ \\ (J=1,\quad 0)}}{\left[\begin{array}{c} 32.5 \\ \hline 0 \\ \hline 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{array}\right]}
=
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}
$$

Step #2b: Now, we add the blue + yellow values and insert the answer in the 1st and 3rd elements within the top bunk of our new "super-fine ($J = 3$)" stack. Note that when we do these additions, we have to skip every other slot when we're storing the sums !! In engineering, this is called ***up-sampling***.

$$
\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}
\qquad
\begin{bmatrix} 36.5 \\ \\ 36.5 \\ \\ \hline \\ \\ \\ \\ \end{bmatrix}
\xleftarrow{\text{Add}}
\begin{bmatrix} 32.5 \\ 32.5 \\ \hline \\ \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix}
\begin{bmatrix} 4 \\ 4 \end{bmatrix}
\qquad
\begin{bmatrix} 32.5 \\ \hline 0 \\ \hline 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix}
=
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}
$$

Extracted

$$
\begin{array}{cccc}
\begin{array}{c} Target \\ b \end{array} & \begin{array}{c} Super-fine \\ (J = 3) \end{array} & \begin{array}{c} Fine \\ (J = 2) \end{array} & \begin{array}{c} Coarse\ and\ DC \\ (J = 1, \quad 0) \end{array}
\end{array}
$$

Step #2c: Then, we subtract the blue + yellow values and insert the answer in the 2nd and 4th elements within the top bunk of our new "super-fine ($J = 3$)" stack.

$$
\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}
\qquad
\begin{bmatrix} 36.5 \\ 28.5 \\ 36.5 \\ 28.5 \\ \hline \\ \\ \\ \\ \end{bmatrix}
\xleftarrow{\text{Subtract}}
\begin{bmatrix} 32.5 \\ 32.5 \\ \hline \\ \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix}
\begin{bmatrix} 4 \\ 4 \end{bmatrix}
\qquad
\begin{bmatrix} 32.5 \\ \hline 0 \\ \hline 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix}
=
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}
$$

Extracted

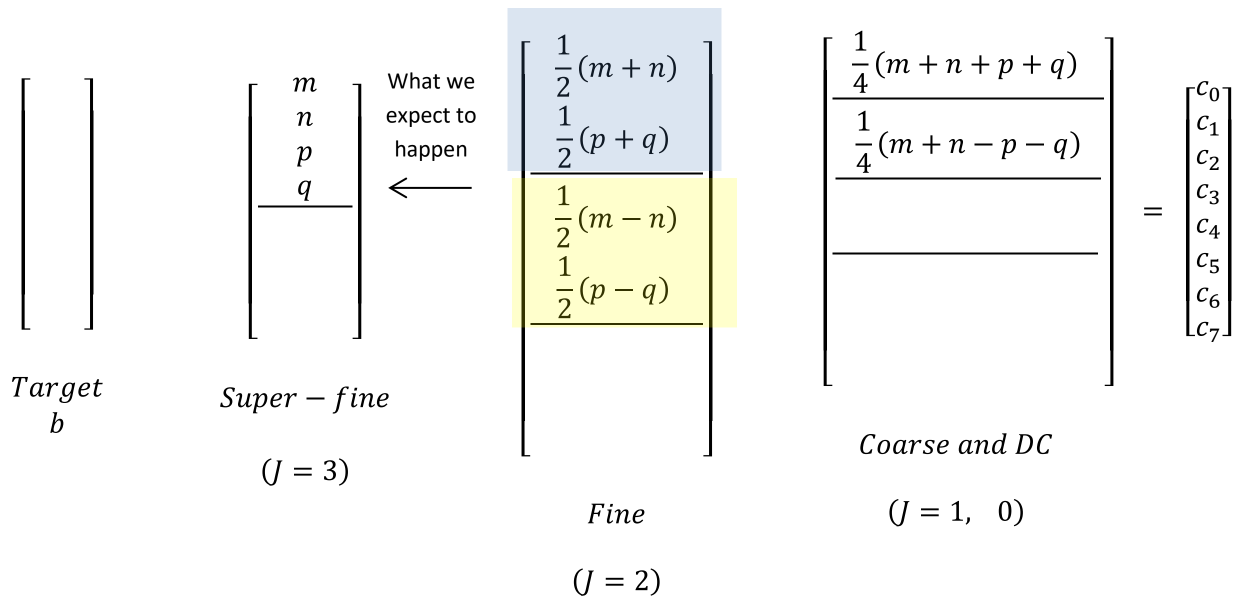
$$
\begin{array}{cccc}
\begin{array}{c} Target \\ b \end{array} & \begin{array}{c} Super-fine \\ (J = 3) \end{array} & \begin{array}{c} Fine \\ (J = 2) \end{array} & \begin{array}{c} Coarse\ and\ DC \\ (J = 1, \quad 0) \end{array}
\end{array}
$$

Step #2d: Finally, we copy all left-over gray bunk values over to the lower "basement" of the new "fine ($J = 3$)" stack.

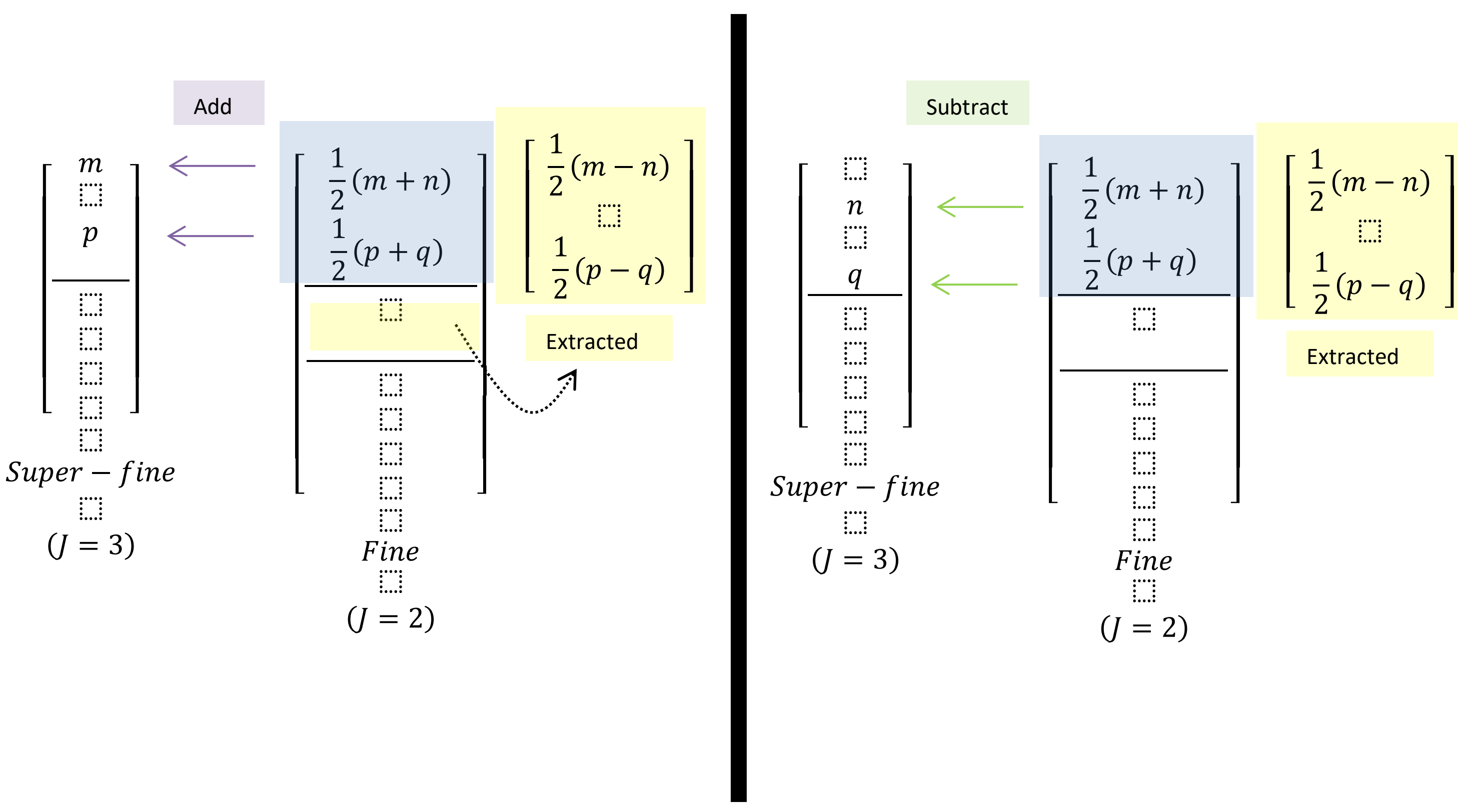
$$
\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}
\qquad
\begin{bmatrix} 36.5 \\ 28.5 \\ 36.5 \\ 28.5 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix}
\xleftarrow{\text{Copy}}
\begin{bmatrix} 32.5 \\ 32.5 \\ \hline 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix}
\qquad
\begin{bmatrix} 32.5 \\ \hline 0 \\ \hline 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix}
=
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}
$$

$$
\begin{array}{cccc}
\begin{array}{c} Target \\ b \end{array} & \begin{array}{c} Super-fine \\ (J = 3) \end{array} & \begin{array}{c} Fine \\ (J = 2) \end{array} & \begin{array}{c} Coarse\ and\ DC \\ (J = 1, \quad 0) \end{array}
\end{array}
$$

__The reason why it works__: To see what really happened in Steps 2b and 2c, we will once again reexamine the math using dummy variables . . For instance, you know *a priori* that the values within the "super-fine ($J = 3$)" and the "fine ($J = 2$)" stacks must look like this:



Hence, if we add or subtract the blue and yellow values together and place the answers in the correct slots within the "super-fine ($J = 3$)" stack, you will get back the expected answers of m, n, p, and q. Again, notice that when we are storing the addition results, we must ***__up-sample__*** the top bunk of the "super-fine ($J = 3$)" stack by skipping every other slot.



Getting ready for the final step: Now, we will focus on the transition from the "super-fine $(J = 3)$" stack to the target vector$b$. We notice that the top-most bunk (shaded in blue) is now a *4-membered bunk*. Let's colored that blue, and color the rest in gray.

$$\underset{\substack{Target\\ b}}{\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}} \qquad \underset{\substack{Super-fine\\ \\ (J=3)}}{\left[\begin{array}{c} 36.5\\ 28.5\\ 36.5\\ 28.5\\ \hline 20.5\\ -4.5\\ -11.5\\ 27.5 \end{array}\right]} \qquad \underset{\substack{Fine\\ \\ (J=2)}}{\left[\begin{array}{c} 32.5\\ 32.5\\ \hline 4\\ 4\\ \hline 20.5\\ -4.5\\ -11.5\\ 27.5 \end{array}\right]} \qquad \underset{\substack{Coarse\ and\ DC\\ \\ (J=1,\quad 0)}}{\left[\begin{array}{c} 32.5\\ \hline 0\\ \hline 4\\ 4\\ \hline 20.5\\ -4.5\\ -11.5\\ 27.5 \end{array}\right]} = \begin{bmatrix} c_0\\ c_1\\ c_2\\ c_3\\ c_4\\ c_5\\ c_6\\ c_7 \end{bmatrix}$$

Step #3a: To begin, If the top-most bunk (shaded in blue) was a *4-membered bunk*, we will "extract" a *4-membered bunk* from the top of the gray stack and color it yellow.

$$\underset{\substack{Target\\ b}}{\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}} \qquad \underset{\substack{Super-fine\\ \\ (J=3)}}{\left[\begin{array}{c} 36.5\\ 28.5\\ 36.5\\ 28.5\\ \hline 20.5\\ -4.5\\ -11.5\\ 27.5 \end{array}\right]} \qquad \underset{\substack{Fine\\ \\ (J=2)}}{\left[\begin{array}{c} 32.5\\ 32.5\\ \hline 4\\ 4\\ \hline 20.5\\ -4.5\\ -11.5\\ 27.5 \end{array}\right]} \qquad \underset{\substack{Coarse\ and\ DC\\ \\ (J=1,\quad 0)}}{\left[\begin{array}{c} 32.5\\ \hline 0\\ \hline 4\\ 4\\ \hline 20.5\\ -4.5\\ -11.5\\ 27.5 \end{array}\right]} = \begin{bmatrix} c_0\\ c_1\\ c_2\\ c_3\\ c_4\\ c_5\\ c_6\\ c_7 \end{bmatrix}$$

Step #3b: Now, we add the blue + yellow values and insert the answer in the 1st, 3rd, 5th, and 7th elements in our b-vector stack. Once again, ***up-sampling*** of the empty target vector is required when we're storing the sums.

Add

$$\begin{bmatrix} 57 \\ \\ 24 \\ \\ 25 \\ \\ 56 \\ \\ \end{bmatrix} \quad \begin{bmatrix} 36.5 \\ 28.5 \\ 36.5 \\ 28.5 \\ \hline \\ \\ \\ \\ \end{bmatrix} \begin{bmatrix} 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix} \quad \begin{bmatrix} 32.5 \\ 32.5 \\ \hline 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix} \quad \begin{bmatrix} \underline{32.5} \\ \underline{0} \\ 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}$$

Extracted

$Target$ b | $Super-fine$ $(J = 3)$ | $Fine$ $(J = 2)$ | $Coarse\ and\ DC$ $(J = 1,\ \ 0)$

Step #3c: Then, we subtract the blue + yellow values and insert the answer in the 2nd, 4th, 6th, and 8th elements in our b-vector stack.

Subtract

$$\begin{bmatrix} 57 \\ 16 \\ 24 \\ 33 \\ 25 \\ 48 \\ 56 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 36.5 \\ 28.5 \\ 36.5 \\ 28.5 \\ \hline \\ \\ \\ \\ \end{bmatrix} \begin{bmatrix} 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix} \quad \begin{bmatrix} 32.5 \\ 32.5 \\ \hline 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix} \quad \begin{bmatrix} \underline{32.5} \\ \underline{0} \\ 4 \\ 4 \\ \hline 20.5 \\ -4.5 \\ -11.5 \\ 27.5 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}$$

Extracted

$Target$ b | $Super-fine$ $(J = 3)$ | $Fine$ $(J = 2)$ | $Coarse\ and\ DC$ $(J = 1,\ \ 0)$

Yay !!!! We have successfully reconstructed our original target vector b =)

Now, we are ready to talk about *filtered reconstruction* of a target data set

2) Data filtering: Identify the "useless" wavelet coefficients in your problem

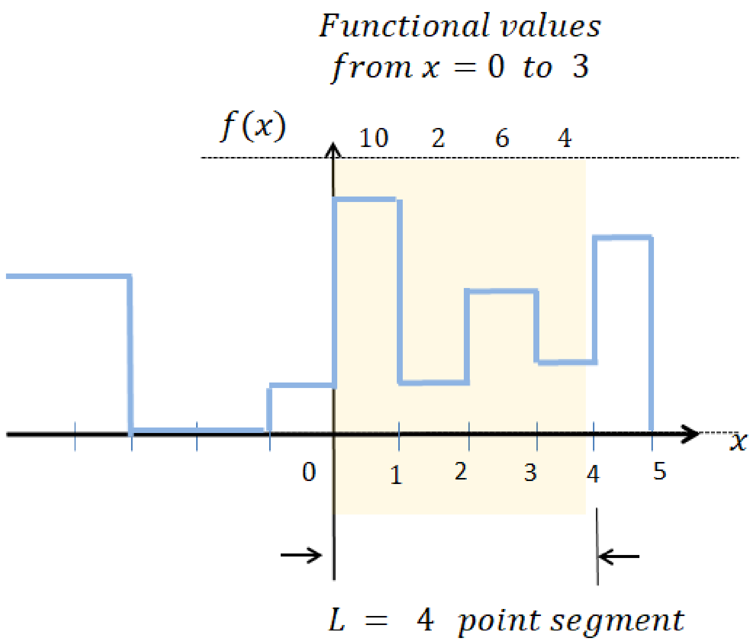


Figure 2: Our original wavelet problem, with a data word length of $L = 4$.

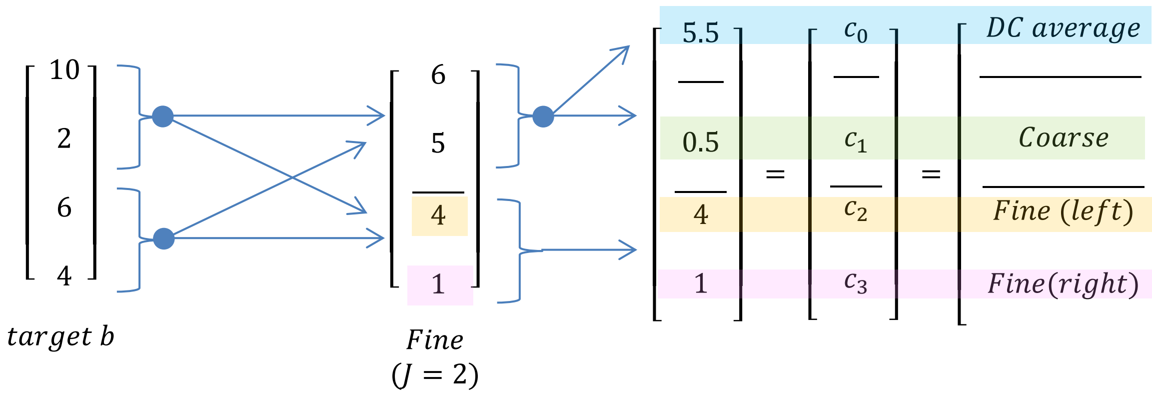
Recall our original, easy-peasty 4-point Haar matrix problem: We would like to reconstruct a target vector b as a linear combo of the Haar wavelet basis:

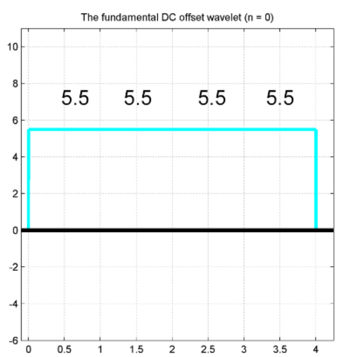
$$b = c_0 \overrightarrow{w_0} + c_1 \overrightarrow{w_1} + c_2 \overrightarrow{w_2} + c_3 \overrightarrow{w_3}$$

In matrix form, this equation becomes our familiar friend:

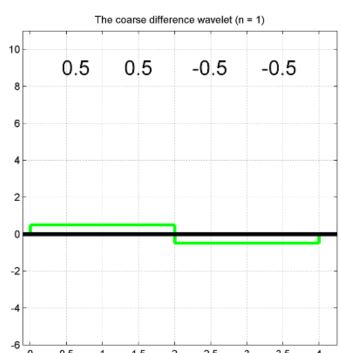
$$\underset{b}{\begin{bmatrix} 10 \\ 2 \\ 6 \\ 4 \end{bmatrix}} = \underset{W}{\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix}} \cdot \underset{c}{\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}}$$

In a previous overview file, we learned that the wavelet coefficients c_0 thru c_3 can be solved by using the high-school-style, “top-bunk addition / bottom-bunk subtraction” method:

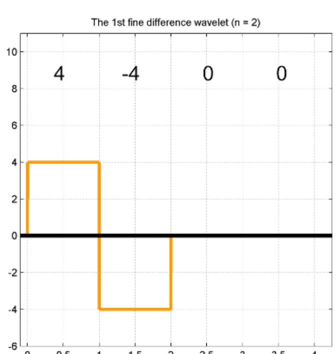




$c_0\overrightarrow{w_0}$ $=$ *Contribution of the DC average component towards our target data vector b*



$c_1\overrightarrow{w_1}$ $=$ *Contribution of the coarse* $(J = 1)$ *component towards our target data vector b*



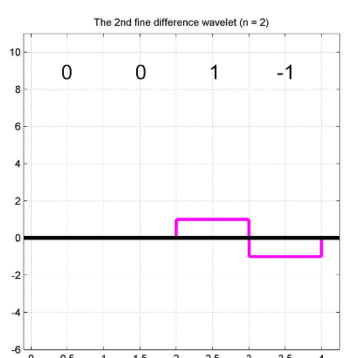


Figure 3: The $c_1\overrightarrow{w_1}$ and $c_3\overrightarrow{w_3}$ components contribute relatively little in the overall wavelet reconstruction of our target data vector b

$c_2\overrightarrow{w_2}$ $+$ $c_3\overrightarrow{w_3}$ $=$ *Contributions of the fine* $(J = 2)$ *components toward our target data vector b*

From Figure 1, we can immediately see that the components associated with c_1 and c_3 are relatively useless in the overall reconstruction of our target vector b because the data values of $c_1\overrightarrow{w_1}$ and $c_3\overrightarrow{w_3}$ are so small. This suggests:

1) Maybe we can *delete* the data associated with c_1 and c_3 by setting both wavelet coefficients to zero

2) Then, we'll just try to reconstruct the target vector b with only contributions from c_0 and c_2.

3) The hope is that our simplified reconstruction from (2) will be good enough…. and at the same time, maybe we can also save a lot of computer memory !!! =)

2b) Removing unwanted wavelet coefficients c_n using a "filter" matrix D

To remove any unwanted wavelet coefficients from your analysis, all you have to do is to "shrink" those coefficients to zero ! In matrix language, this means you can remove c_1 and c_3 using a diagonal matrix Λ, where you place the zeros in the appropriate positions along the diagonal:

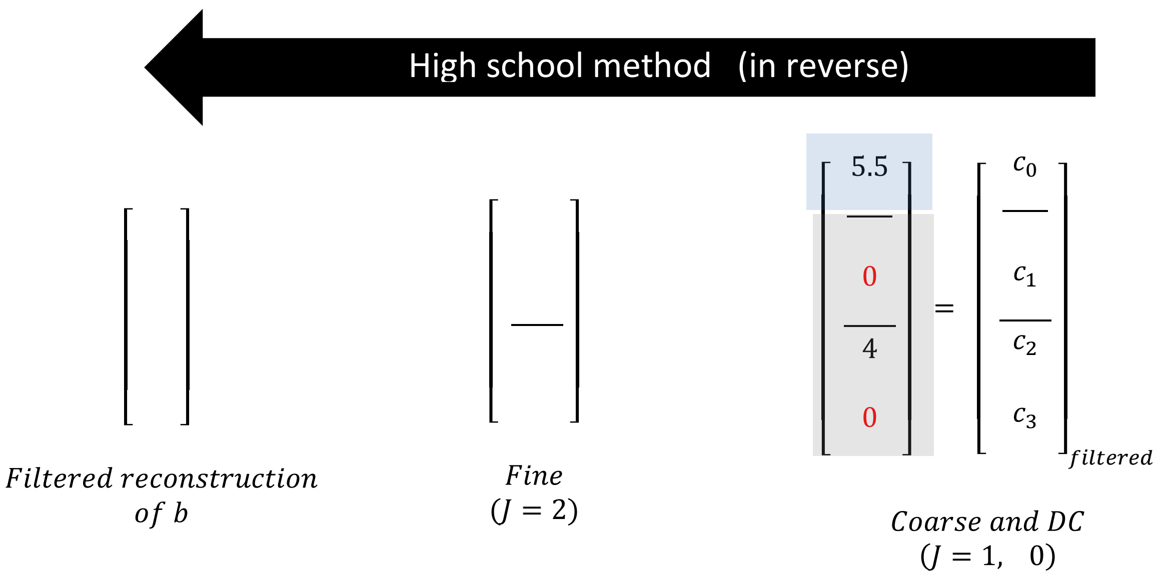
$$c_{filtered} = \Lambda\ c$$

$$= \begin{bmatrix} 1 & & & \\ & 0 & & \\ & & 1 & \\ & & & 0 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

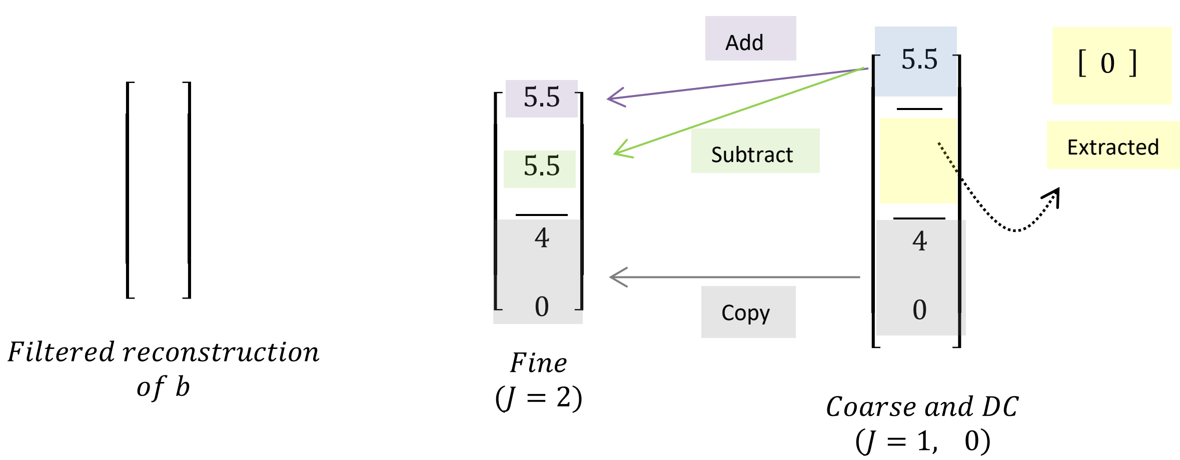
$$c_{filtered} = \begin{bmatrix} c_0 \\ 0 \\ c_2 \\ 0 \end{bmatrix} = \textit{Easy as pie !!!} \quad =)$$

2c) “Filtered” reconstruction of target data b (using reverse high-school math)

Using our filtered wavelet coefficients $c_{filtered}$, we can use the high-school math method to reconstruct our target data b. Let’s draw out our bins and identify the top-most bunk, and note that our post-filtered $c_1 = c_3 = 0$.



Since the top blue bunk is a *single-membered bunk*, we will extract out a yellow *single-membered bunk* from the top of the gray stack. Then, we’ll fill in the “fine ($J = 2$)” stack via a combination of addition, subtraction, and copying operations !



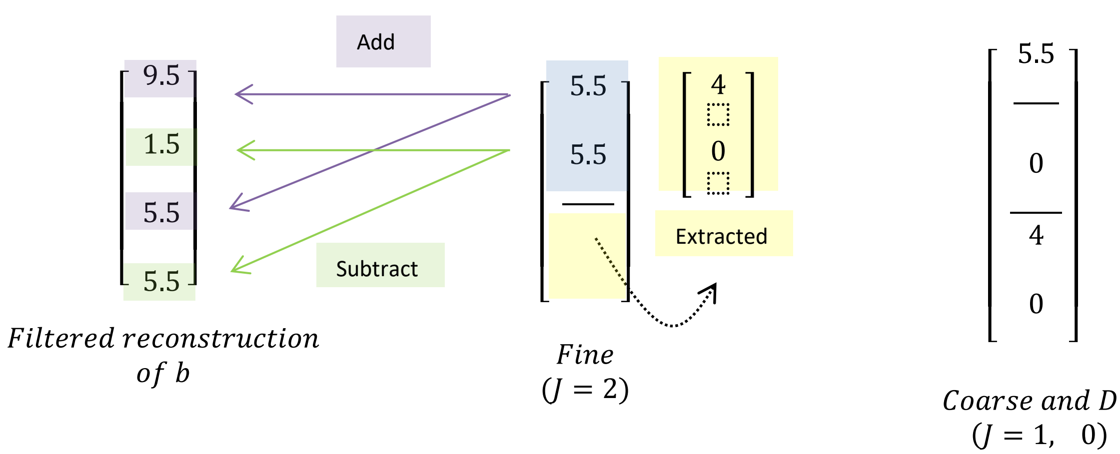
Then, we can repeat the process, where:

1) We identify the top blue bunk in the “fine $(J = 2)$” stack, and note how many members it contains
2) We “extract” out the necessary yellow bunk from the top of the gray stack
3) Fill in the filtered reconstruction of b by perform addition, subtraction, and copy operations (if any)

$$\begin{bmatrix} \ \\ \ \\ \ \\ \ \end{bmatrix} \qquad \begin{bmatrix} 5.5 \\ 5.5 \\ \hline 4 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 5.5 \\ \hline 0 \\ \hline 4 \\ 0 \end{bmatrix}$$

Filtered reconstruction of b | *Fine* $(J = 2)$ | *Coarse and DC* $(J = 1, \quad 0)$

Note that we have nothing to “copy” at this step…. and as always, when we’re first storing the purple addition results, we need to *<u>up-sample</u>* the reconstruction stack first !!



Our filtered reconstruction of target data b:

$$b_{recon} = \begin{bmatrix} 9.5 \\ 1.5 \\ 5.5 \\ 5.5 \end{bmatrix} = c_0 \overrightarrow{w_0} + \cancel{c_1 \overrightarrow{w_1}} + c_2 \overrightarrow{w_2} + \cancel{c_3 \overrightarrow{w_3}}$$

2 components removed from our reconstruction

2d) Compare the original vs. filtered data

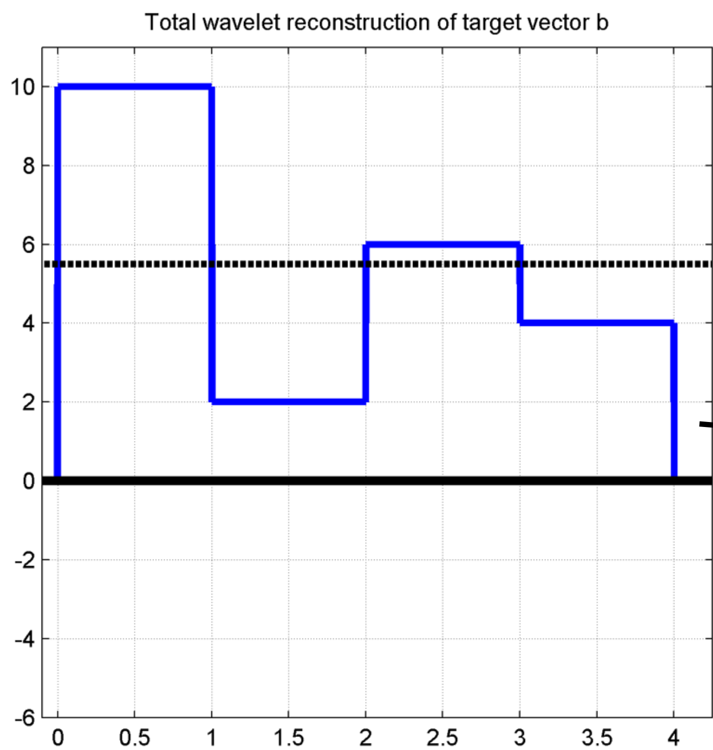


Figure 4

Original target data b

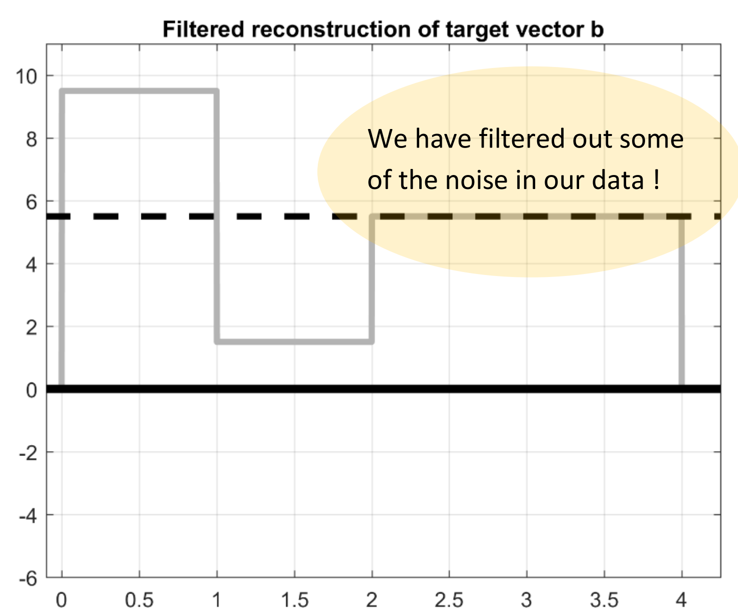


Figure 5

Filtered reconstruction of target data b by removing c_1 and c_3 components