Hierbij een beetje een beschrijving van de verbeterde versie. Deze kan bijna het gehele spectrum restaureren. Verder zaten in mijn oude code nog wat onnodige shifts van de subbanden, maar dit kwam doordat de subband 0 begint met een halve subband.

In de buurt van de rand van een subband, krijgen we bijvoorbeeld de volgende gewichten [0.4, 0.6, 0.001] voor een FFT bin binnen een subband, met dezelfde FFT bin van de naastgelegen subbanden. Zonder een al te grote fout te maken kunnen eenvoudig het laatste gewicht weg laten. Dit verhaal beschrijft alleen deze situatie. Voor het geval de eerste waarde wegvalt, kan een soortgelijke redenatie worden gehouden als nu volgt.

De gemeten respons bestaat dus uit de convolutie van de echte respons. Dit is in feite waar mijn vorige verhaal eindigde, maar in plaats van een oneindge convolutie, nu met slechts de [0.6, 0.4] vector.

Om deze convolutie ongedaan te maken, kunnen we de kern van de convolutie eerst schrijven als een matrixvermenigvuldiging.

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 0.4 & 0.6 & 0 & 0 \\ 0 & 0.4 & 0.6 & 0 \\ 0 & 0 & 0.4 & 0.6 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \tag{1}$$

Hierbij is d de power van het gedecimeerde spectrum en s is het werkelijke spectrum. De filter matrix zelf zal ik F gaan noemen. Omdat de convolutie meer input data heeft dan output data, is de matrix niet vierkant.

In de originele code heb ik domweg de eerste kolom van de matrix afgekapt en s_0 verwijderd. Hierdoor kan s eenvoudig worden opgelost. Als de coefficienten flink verschillen benadert $F_{1,1}$ de waarde nul, dus het aliasing is klein. Echter zodra de filter coefficienten dichtbij elkaar komen (de rand van de subband) heeft dit gevolgen.

Een andere oplossing om de matrix vierkant te maken is om er een geschat gedecimeerd datapunt $\hat{d_0}$ aan toe te voegen. Het feit dat d_0 op zijn beurt ook weer ge-aliased zou moeten zijn met de waarde van s_{-1} maakt met de hieronder besproken methode niet uit.

$$\begin{bmatrix} \hat{d_0} \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 0.6 & 0 & 0 & 0 \\ 0.4 & 0.6 & 0 & 0 \\ 0 & 0.4 & 0.6 & 0 \\ 0 & 0 & 0.4 & 0.6 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \tag{2}$$

$$d = F \cdot s \tag{3}$$

Nu kunnen we s oplossen

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1.67 & 0 & 0 & 0 \\ -1.11 & 1.67 & 0 & 0 \\ 0.74 & -1.11 & 1.67 & 0 \\ -0.49 & 0.74 & -1.11 & 1.67 \end{bmatrix} \begin{bmatrix} \hat{d}_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} \tag{4}$$

$$s = F^{-1} \cdot d \tag{5}$$

Verder ga ik ervan uit, dat het gedecimeerde spectrum van $d_{1,\cdots n}$ wordt gemeten, dus dat we daarom ook niet geinteresseerd zijn in de waarde van s_0. We kunnen dus vereenvoudigen tot

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} -1.11 \\ 0.74 \\ -0.49 \end{bmatrix} \hat{d}_0 + \begin{bmatrix} 1.67 & 0 & 0 \\ -1.11 & 1.67 & 0 \\ 0.74 & -1.11 & 1.67 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \tag{6}$$

Als we de eerste kolom van F^{-1} behalve de eerste waarde $(F^{-1})_0$ noemen en F^{-1} zonder de eerste kolom en eerste rij $(F^{-1})_{\text{rest}}$ noemen, dan

$$s_{1,\cdots,n} = (F^{-1})_0 \hat{d}_0 + (F^{-1})_{\text{rest}} d_{1,\cdots,n} \tag{7}$$

De tweede term $(F^{-1})_{\text{rest}} d_{1,\cdots,n}$ is eigenlijk wat ik in mijn oorspronkelijke code deed. Me moeten deze oplossing dus nog corrigeren met de term $(F^{-1})_0 \hat{d}_0$.

De waarde van $\hat{d}_0$ kunnen we afschatten door gebruik te maken van het feit dat over het algemeen het spectrum van een FFT bin lijkt op dat van de naastgelegen bin(s). Met behulp van de kleinste kwadraten methode vinden we

$$\hat{d}_0 = \frac{(F^{-1})_0^T \left[\hat{s}_{1,\cdots,n} - (F^{-1})_{\text{rest}} d_{1,\cdots,n}\right]}{(F_0^{-1})^T (F^{-1})_0} \tag{8}$$

Waarbij $\hat{s}$ dan het geschatte spectrum bevat.

De implementatie van dit algorithme kan worden uitgevoerd door de matrix F op te bouwen en te inverteren, letterlijk zoals hierboven beschreven. Dit is wat ik in mijn oude code deed. We kunnen de implementatie nog iets anders aanpakken. Dit is dus puur de implementatie. Wiskundig blijft alles gelijk.

Wat opvalt aan de inverse matrix F^{-1} is dat afgezien van een verschuiving, alle rijen gelijk zijn. Voor de eerste kolom kunnen we uitrekenen dat

$$(F^{-1})_{i,1} = \frac{1}{F_{1,1}} \left(\frac{-F_{2,1}}{F_{1,1}}\right)^{(i-1)} = \frac{1}{-F_{2,1}} \left(\frac{-F_{2,1}}{F_{1,1}}\right)^{i} \tag{9}$$

De laatste rij is gelijk aan de eerste kolom in omgekeerde volgorde. Verder kunnen we de verschuiving van de rijen gebruiken om de matrixvermenigvuldiging als een convolutie te schrijven.

Een 3-terms convolutie is meer dan genoeg, afgezien bij exact de grens van een subband. Hier zijn alle termen nodig. Ik doe nog gewoon een convolutie, maar aangezien de termen in dit laatste geval alternerend 2 en -2 zijn, zijn efficientere oplossingen mogelijk.

Tot slot is de correctie met $\hat{d}_0$ alleen echt nodig in de buurt van de rand van een subband. In de code is dit afgevangen met een if statement. Het zou echter geen kwaad kunnen om dit over de gehele subband uit te voeren. Aangezien een 3-terms convolutie al goed genoeg is, zou je zelfs (als je genoeg subbanden hebt) deze hele excercitie ook alleen hoeven toe te passen op de grens van de subband. De eerste en de laatste twee tot drie subbanden zullen dan iets verslechterd zijn.