RTDSP Code Explanations

// Occur when updating the candidates

**Opt1:**

Optimisation 1 involves running the input frames through a low pass filter before using them to update the candidates. This removes the high frequency noise that would not be found in speech. This is implemented through the use of two arrays, P[] and P\_prev[] which hold the low pass filtered samples for the current time’s quarter frame, and the previous quarter frame, respectively. For the ith sample in the quarter frame, its filtered value is calculated using the equation

and stored in P[i], and also P\_prev[i], for use in next frame’s calculation.

**Opt2:**

Optimisation 2 is a modification to optimisation 1 that performs the same calculation in the power domain, by using the square of the magnitude of the input in the expression above, then square rooting the result to find the final value. The equivalent expression is

// Occur when updating nmb from candidates

**OptA:**

OptA was an attempt to exploit the fact that we are trying to remove everything that is not speech, by researching the human vocal range (300-3500Hz) and assuming everything outside it was noise. This was achieved by calculating which bins corresponded to those frequencies, then maximising our noise estimate for bins outside that range.

**Opt6:**

Optimisation 6 works by increasing the noise estimate for frequency bins which have a low signal to noise ratio (SNR), by multiplying the noise estimate by a coefficient calculated using the SNR. The first task was to find an expression that would calculate the SNR. This was achieved using the assumption that the input is the summation of noise and signal.

Now that we have an expression for the SNR, we have to generate a value inversely proportional to it, as we want a higher noise estimate for a lower SNR. We chose to take the negative logarithm of the SNR, which exponentially increases the noise estimate the smaller the SNR is. We chose to use a logarithm because we wanted to highly increase the noise estimate for low SNR but only barely increase it for high SNR, and a linear function would not have allowed us to achieve both of these aims. For instance, if we had maximised noise estimation for 0 SNR by using the linear function , (where FLT\_MAX is the maximum value that can be held in a float) we would get maximum noise estimate at 0 SNR, but an SNR of 0.999 would still increase the noise estimate by over 1035 times. Since the logarithmic function diverges to infinity, we capped the output of the function at FLT\_MAX to prevent overflow.

**Opt3:**

Optimisation 3 low pass filters the noise minimum buffer to remove discontinuities created when the candidates rotate.