The purpose of this note is to collect and explain certain mathematical aspects of Recent Average Credit (RAC). In short, RAC is BOINC's attempt at measuring a user's time-averaged computational power. More precisely, RAC is an exponentially weighted moving average over computational power, thus granting more weight to recently earned credits versus old credits. The details of its calculation are as follows.

Generally speaking, an exponential moving average S_i of a fixed-interval time series X_i is a moving average which assigns exponentially less weight to data points in the past. More precisely,

$$S_i = (1-\alpha)^i X_0 + \alpha \sum_{j=1}^{i} (1-\alpha)^{j-1} X_{i-j+1} \tag{1}$$

where $0 < \alpha < 1$ is a constant weighting factor. A large α, corresponding to smaller $1-\alpha$, assigns smaller weight to data points in the past. S_i can also be computed recursively for $i > 1$:

$$S_i = \alpha X_i + (1-\alpha) S_{i-1}. \tag{2}$$

with $S_0 \equiv X_0$. This is sometimes called a low-pass filter, and has a smoothing effect on the time series X_i, which may be noisy. This is why this kind of calculation makes sense for BOINC: we want to filter out short-time fluctuations in user output in favor of a smoothly varying average, since the number of workunits completed in a given (fixed) time interval can fluctuate significantly for any number of reasons.

One difficulty with this approach is that generally BOINC updates its statistics at uneven time intervals $\ldots < t_{i-2} < t_{i-1} < t_i$, and hence a time-dependent weight α must be defined. BOINC has found it convenient to use the exponential weighting factor:

$$\alpha_i = 1 - \mathrm{e}^{-(t_i - t_{i-1}) \cdot \ln(2)/th} \tag{3}$$

where t is given in days. The BOINC source code actually works equivalently in terms of the weight function

$$w(t_i - t_{i-1}) \equiv \mathrm{e}^{-(t_i - t_{i-1}) \cdot \ln(2)/th}. \tag{4}$$

The quantity th is called the halving parameter, since

$$w(t_i - t_{i-1}) = \left(\mathrm{e}^{-\ln(2)}\right)^{(t_i - t_{i-1})/th} = \left(\frac{1}{2}\right)^{(t_i - t_{i-1})/th} \tag{5}$$

using elementary properties of logarithms. BOINC has found it convenient to set th equal to 7 days. Note that when $t_i - t_{i-1}$ is large, α_i is close to 1, thus indeed granting smaller weight to older computations. In fact, if RAC is updated weekly, credit from 1 week ago is granted half the weight, credit from 2 weeks ago is granted one-quarter the weight, etc. Roughly speaking, credit from more than 2 months ago will not contribute significantly.

Let's put all of this together. Suppose the computational power (credits per day) at time t_i is R_i. Having defined $RAC(t_i)$ in the above as the weighted moving average of the previous R_i, we have:

$$RAC(t_i) = \left[1 - w(t_i - t_{i-1})\right] R_i + w(t_i - t_{i-1}) \cdot RAC(t_{i-1}). \tag{6}$$

Of particular interest is the *long time limit* of this equation. If we crunch at a constant rate R for several months, what value will RAC level out to? If we assume RAC is updated at constant time intervals $t_i - t_{i-1} = \Delta t$, then it is not hard to find at least *one* time-invariant solution to the above equation: $RAC = R$. To be careful, one should prove that there are no other solutions. This turns out to be in fact the case, as one can show using equation (1), but I won't write out the steps here.

Thus, if one crunches at a constant rate R *credits per day*, and RAC is updated in constant time intervals, then RAC levels out to R. If $t_i - t_{i-1}$ is not constant but depends on i, there will probably be some variations, although I haven't tried to quantify these.