Christopher Tsai
February 1, 2008
EE 398 – Image Compression

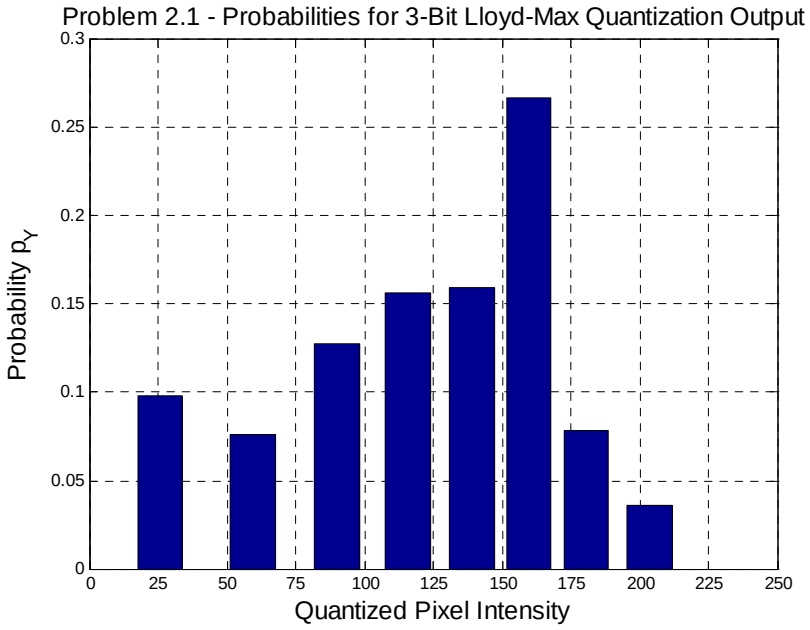
Problem Set III – Quantization

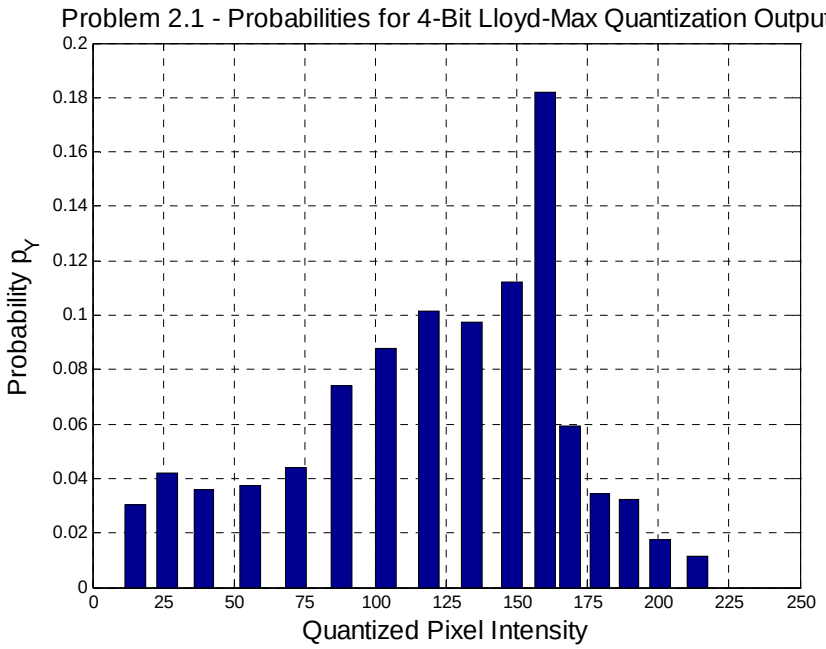
Problem #2.1 – Lloyd-Max Quantizer

To train both the Lloyd-Max Quantizer and our Entropy-Constrained Quantizer, we employ the following training set of images, sampling every four lines and every four columns:

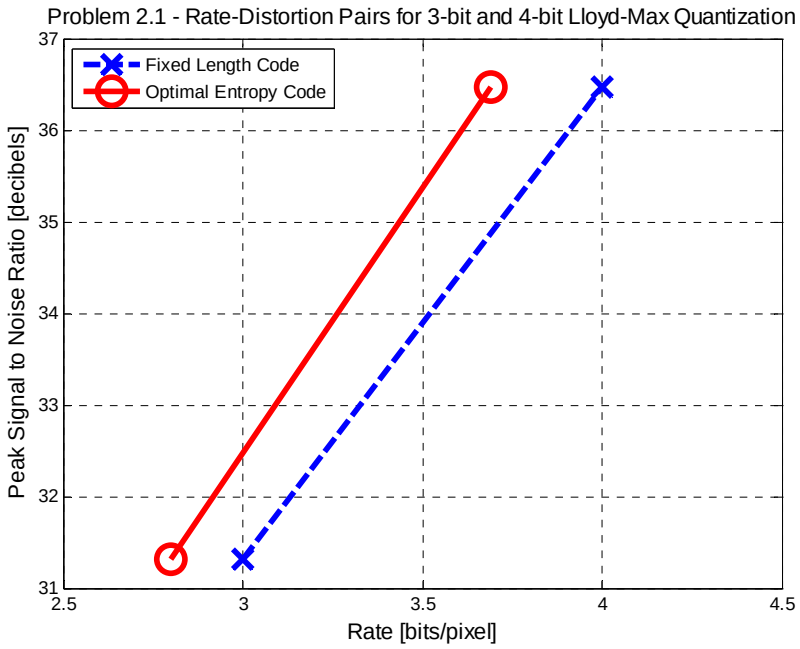


We set our Lagrange multiplier to zero and our termination tolerance to $\epsilon = 0.001$. As recommended, we replace the codeword in an empty cell with a new codeword, generated by splitting the most populated cell. Implementing the algorithm with a full cost-minimum search, we obtain the following optimal codebook probabilities for the 3-bit and 4-bit quantizer:

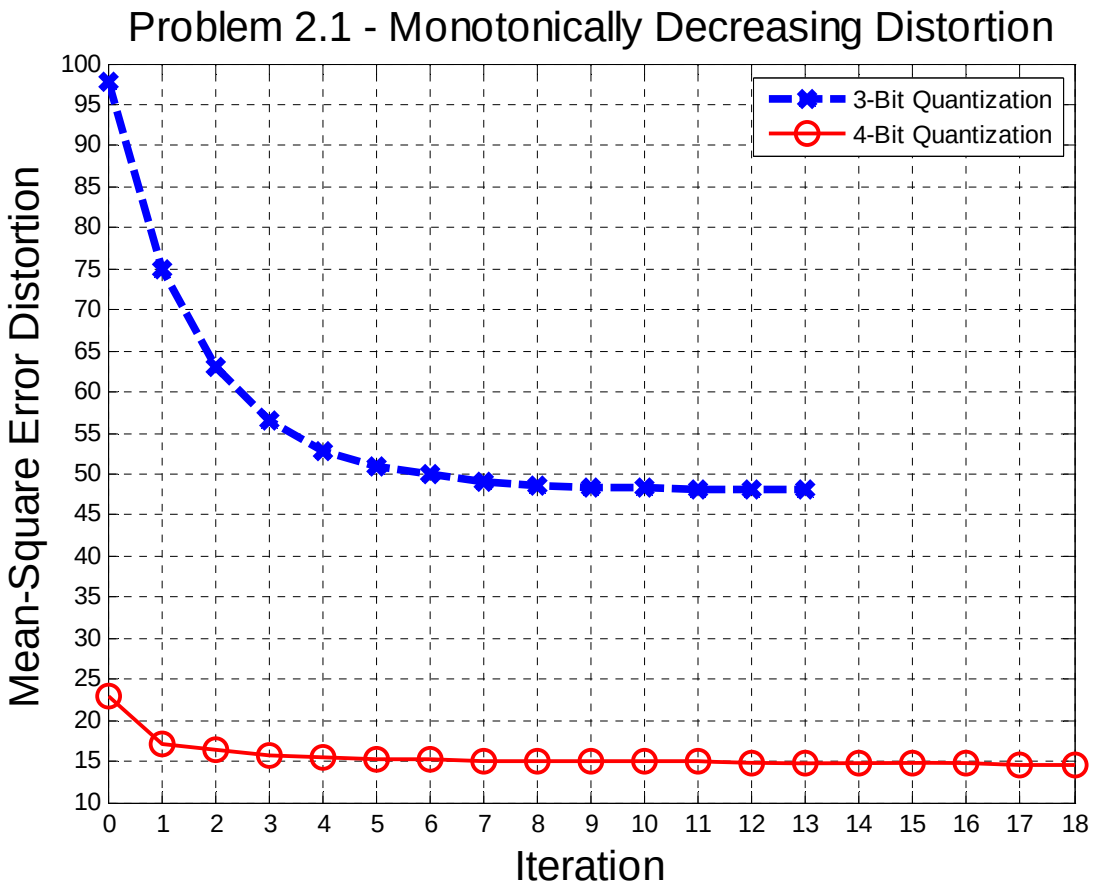




We first notice that the 4-bit quantization contains – unsurprisingly – more levels and therefore finer approximation of the true probability density. Furthermore, both mass functions ascertain the relative paucity of image pixel intensities beyond 225. Hence, the images contain few bright whites, as our normalized histograms reveal. These probability mass functions form the basis of our fixed length code, which yields the following rate-distortion pairs:



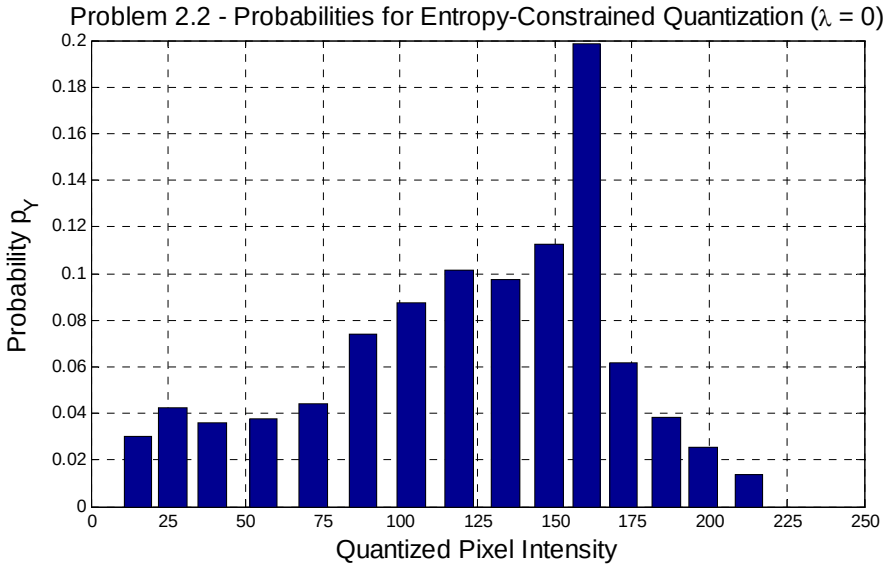
From the 3-bit and 4-bit rate-distortion curves, we can measure the slope to quantify the amount by which we can decrease distortion (elevating the PSNR) by increasing rate (fine-tuning quantization). The Lloyd-Max Quantizer fixed-length code offers a rate-distortion curve slope of **5.157163** dB/bit, while the theoretically optimal entropy code exchanges **5.810281** dB of PSNR for each additional bit of quantization. Meanwhile, we can track the algorithm's convergence by measuring the distortion at each step and watching its monotonic decrease:

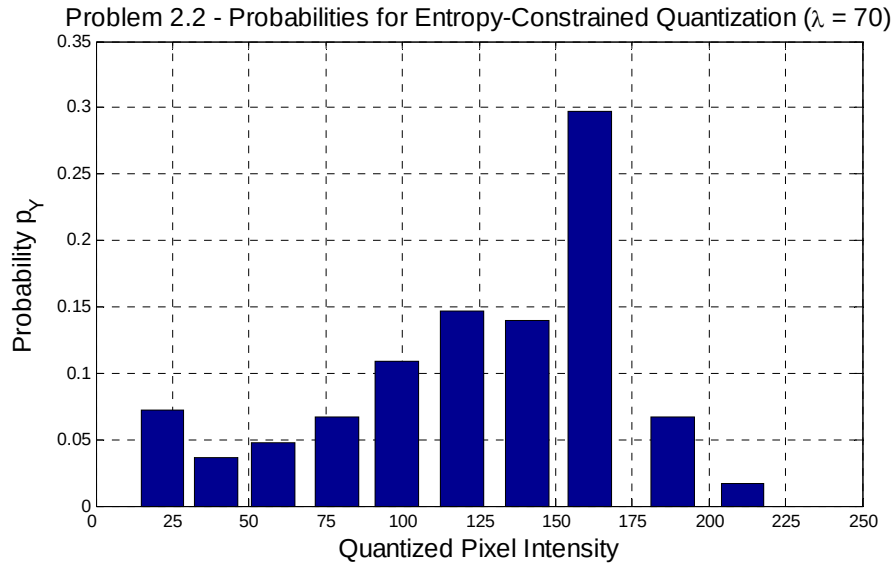


The 3-bit quantization algorithm, containing fewer levels and hence a simpler codebook, converges in only **13 iterations**, whereas the 4-bit quantization algorithm starts with a lower distortion thanks to its improved resolution but also requires more steps to converge based on its self-comparison termination metric; the 4-bit quantization requires **18 iterations** to converge.

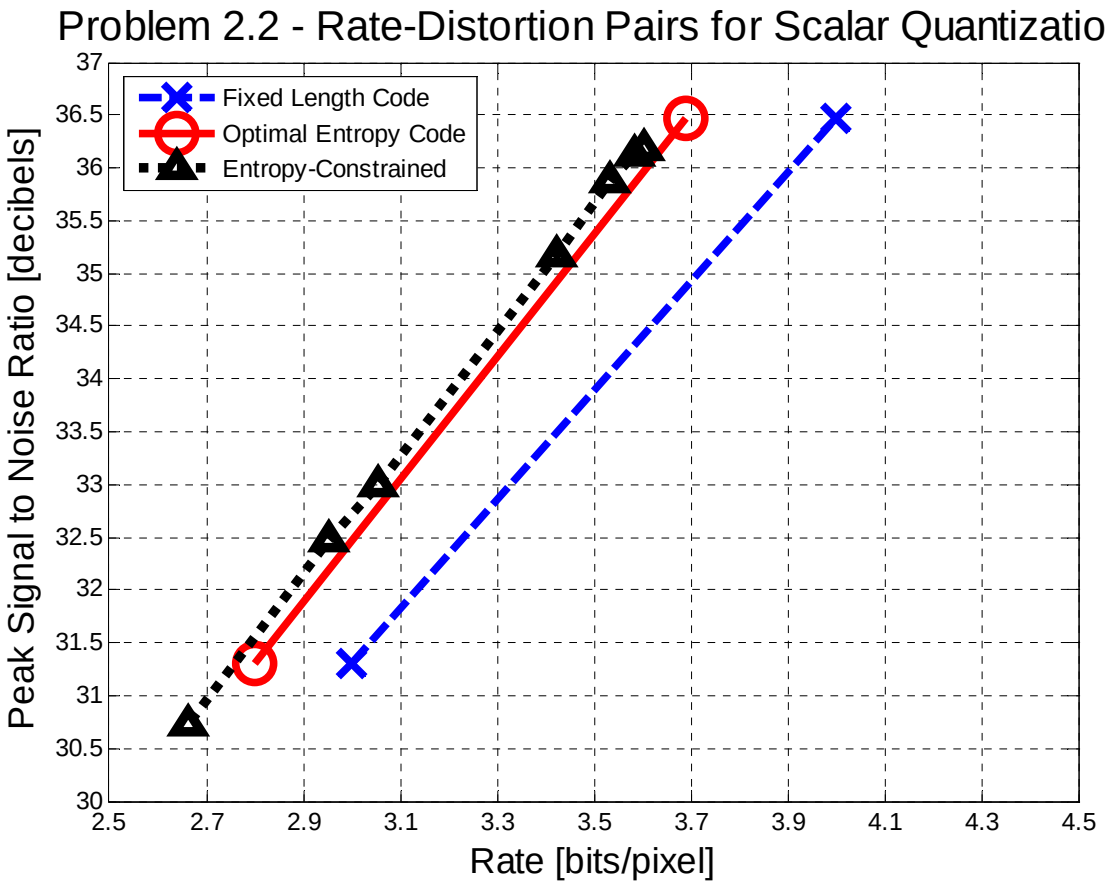
Problem #2.2 – Entropy-Constrained Scalar Quantization

Tapping the same training set of *boats*, *harbour*, and *peppers*, we train our entropy-constrained scalar quantizer without discarding empty bins; because removing a codeword does not waste any bits in the entropy-constrained algorithm, we eradicate empty cells. We again set $\epsilon = 0.01$ as our termination tolerance criterion, but we now vary the Lagrange multiplier λ over the set {0, 5, 10, 30, 62, 70, 90}. Beginning out iterative process with a 4-bit uniform quantizer, we obtain the following probability mass functions for the $\lambda = 0$ and the $\lambda = 70$ cases:



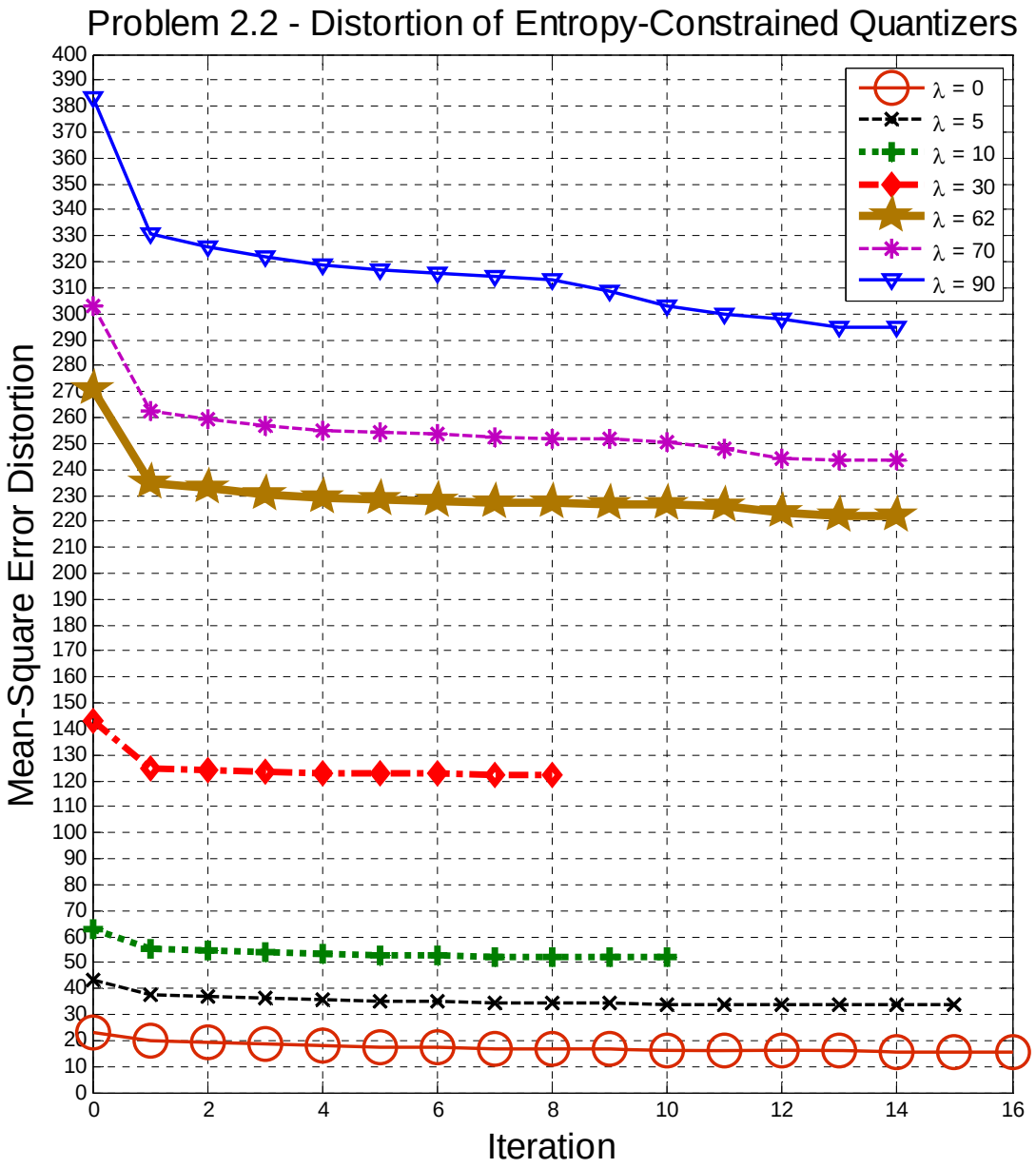


We observe that the higher Lagrange multiplier actually yields a coarser quantization, leading to fewer levels and hence poorer resolution. Let us compare the rate-distortion curves with those of our Lloyd-Max Quantizer:



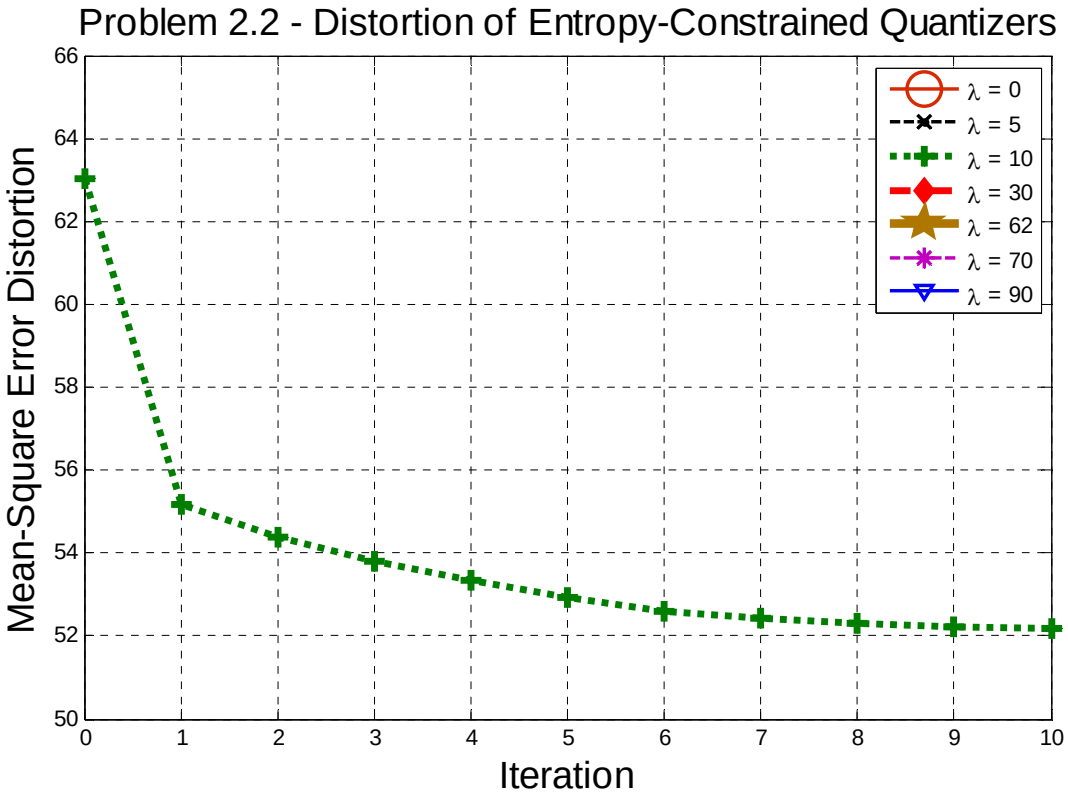
The graph reveals that the entropy-constrained scalar quantization algorithm performs, as its name suggests, extremely well, narrowly outperforming even the optimal entropy code computed under the Lloyd-Max Quantization algorithm. However, for all intents and purposes, the entropy-constrained code, under the assumption of a theoretically optimal entropy code, meets the optimal entropy limit. The rate-distortion curve sports a **slope of approximately 5.719414 dB/bit**.

Finally, we consider the convergence of the distortion as we vary the Lagrange multiplier. As one might expect, the Lagrange multiplier controls how quickly we converge to the optimal codebook, although excessively large values of the multiplier result in overshoot:



For large values of $\lambda > 50$, the distortion remains extremely high, even after convergence. This makes sense; the higher we charge for increasing the rate *R*, the more content we become in settling for distortion. In other words, the optimal distortions are so high for large Lagrange multipliers because we harden the cost of increased rate, resulting in lower bit rates, coarser quantization, fewer levels, and, ultimately, higher distortion. We can focus on the curves for lower λ, noting that $\lambda = 30$ offers the fastest convergence:

In particular, it appears that the λ = 10 algorithm offers the best balance of rapid convergence and low distortion:



Lagrange Multiplier λ	**Iterations to Convergence**
0	16
5	15
10	10
30	8
62	14
70	14
90	14