Image representation and compression via sparse solutions of systems of linear equations

Alfredo Nava-Tudela
John J. Benedetto, advisor

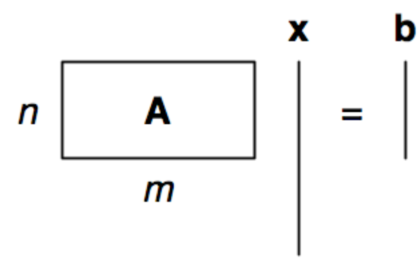
Multimedia flood

Cn u rd ths?

If you answered yes to the above question, then you have grasped what we are trying to do here, but for images. In the example above, we have compressed the sentence "can you read this?" to "cn u rd ths?," which amounts to a reduction of six characters, 33% fewer characters than in the original sentence, but without compromising its meaning.

Problem

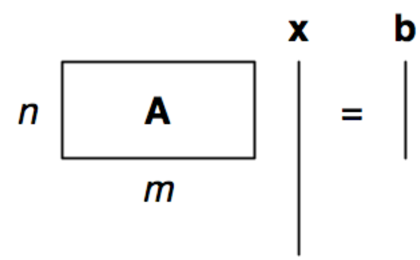
We can do something similar for images by way of the following algebraic trick. Suppose that you have the system of linear equations



where rank(**A**) = $n < m$.

Problem

It is a fact that there are an infinite number of solutions to equations of the type depicted below, provided **A** is full-rank as in our case.



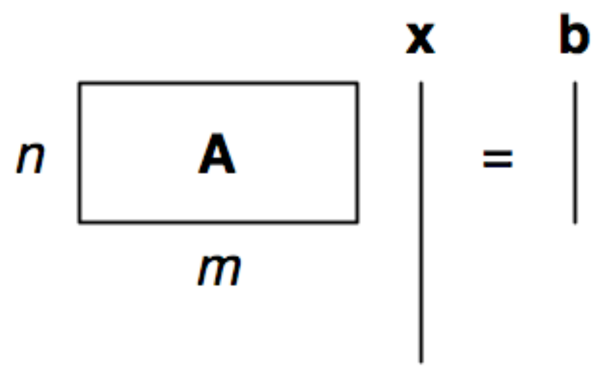
This is what we can exploit to compress an image *I*. Suppose that we can somehow convert *I* into a vector **b** and that for some ad hoc matrix **A** we can find a vector $\mathbf{x}_0$ such that the number of non-zero entries of $\mathbf{x}_0$, from now on written as $||\mathbf{x}_0||_0$, is a lot smaller than the number of non-zero entries of vector **b**, $||\mathbf{x}_0||_0 < ||\mathbf{b}||_0$ in our new notation. Then if we store or transmit $\mathbf{x}_0$ instead of **b** we would have compressed image *I*.

Results

- Fine tuning and speedup of Orthogonal Matching Pursuit (OMP)
- Efficient *QR* implementation of OMP
- Comparison with SolveOMP, a publicly available OMP solution
- DCT+Haar compression vs DCT, Haar compressions
- PSNR, SSIM, and MSSIM error estimation and bit-rate vs distortion
- 1D vs 2D bases comparison
- Quantization and distortion estimate

“Sparsity” equals compression

We can then think of signal compression in terms of our problem



If **x** is sparse, **b** is dense, store **x**!

Definition of "sparse"

- The l_0 "norm":

$$||\mathbf{x}||_0 = \# \{k : x_k \neq 0\}$$

- (P_0): $\min_{\mathbf{x}} ||\mathbf{x}||_0$ subject to $||\mathbf{A}\mathbf{x} - \mathbf{b}||_2 = 0$

- (P_0^{ε}): $\min_{\mathbf{x}} ||\mathbf{x}||_0$ subject to $||\mathbf{A}\mathbf{x} - \mathbf{b}||_2 < \varepsilon$

Observations: In practice, (P_0^{ε}) is the working definition of sparsity as it is the only one that is computationally practical. Solving (P_0^{ε}) is NP-hard [2].

Some theoretical results

Definition: The *spark* of a matrix **A** is the minimum number of linearly dependent columns of **A**. We write spark(**A**) to represent this number.

Theorem: If there is a solution **x** to **Ax** = **b**, and $||\mathbf{x}||_0 < \text{spark}(\mathbf{A}) / 2$, then **x** is the sparsest solution. That is, if $\mathbf{y} \neq \mathbf{x}$ also solves the equation, then $||\mathbf{x}||_0 < ||\mathbf{y}||_0$.

Observation: Computing spark(**A**) is combinatorial, therefore hard. Alternative?

Some theoretical results

Definition: The *mutual coherence* of a matrix **A** is the number

$$\mu(\mathbf{A}) = \max_{1\le k,j\le m,\ k\ne j} \frac{|\mathbf{a}_k^T \mathbf{a}_j|}{||\mathbf{a}_k||_2 \cdot ||\mathbf{a}_j||_2}.$$

Lemma: spark**(A)** ≥ 1+1/μ(**A)**.

Theorem: If **x** solves **Ax = b**, and $||\mathbf{x}||_0 < (1+\mu(\mathbf{A})^{-1})/2$, then **x** is the sparsest solution. That is, if $\mathbf{y} \neq \mathbf{x}$ also solves the equation, then $||\mathbf{x}||_0 < ||\mathbf{y}||_0$.

<u>Observation</u>: μ(**A**) is a lot easier and faster to compute, but 1+1/μ**(A)** far worse bound than spark**(A),** in general.

Finding sparse solutions:OMP

Orthogonal Matching Pursuit algorithm:

Task: Approximate the solution of $(P_0) : \min_{\mathbf{x}} \|\mathbf{x}\|_0$ subject to $\mathbf{Ax} = \mathbf{b}$.

Parameters: We are given the matrix $\mathbf{A}$, the vector $\mathbf{b}$, and the threshold ϵ_0.

Initialization: Initialize $k = 0$, and set

- The initial solution $\mathbf{x}^0 = \mathbf{0}$.
- The initial residual $\mathbf{r}^0 = \mathbf{b} - \mathbf{Ax}^0 = \mathbf{b}$.
- The initial solution support $\mathcal{S}^0 = Support\{\mathbf{x}^0\} = \emptyset$.

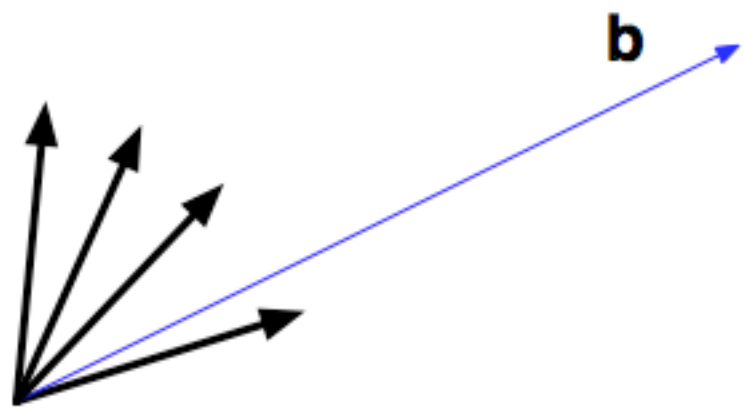
Main Iteration: Increment k by 1 and perform the following steps:

- **Sweep:** Compute the errors $\epsilon(j) = \min_{z_j} \|z_j \mathbf{a}_j - \mathbf{r}^{k-1}\|_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / \|\mathbf{a}_j\|_2^2$.
- **Update Support:** Find a minimizer j_0 of $\epsilon(j)$: $\forall j \notin \mathcal{S}^{k-1}$, $\epsilon(j_0) \leq \epsilon(j)$, and update $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$.
- **Update Provisional Solution:** Compute $\mathbf{x}^k$, the minimizer of $\|\mathbf{Ax} - \mathbf{b}\|_2^2$ subject to $Support\{\mathbf{x}\} = \mathcal{S}^k$.
- **Update Residual:** Compute $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k$.
- **Stopping Rule:** If $\|\mathbf{r}^k\|_2 < \epsilon_0$, stop. Otherwise, apply another iteration.

Output: The proposed solution is $\mathbf{x}^k$ obtained after k iterations.

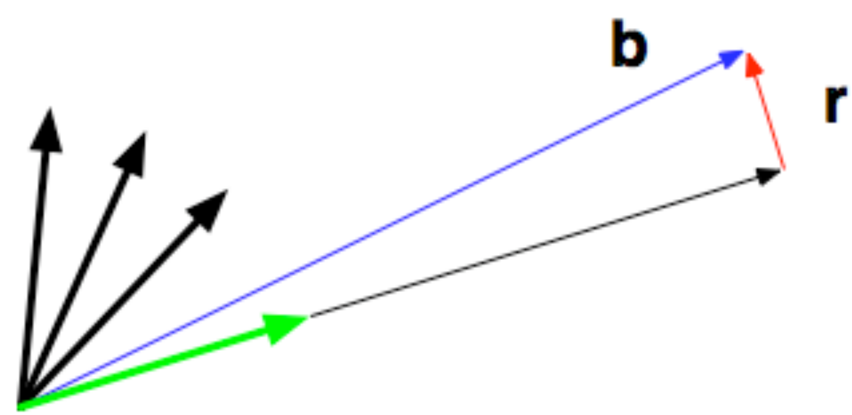
Finding sparse solutions:OMP

Orthogonal Matching Pursuit algorithm:



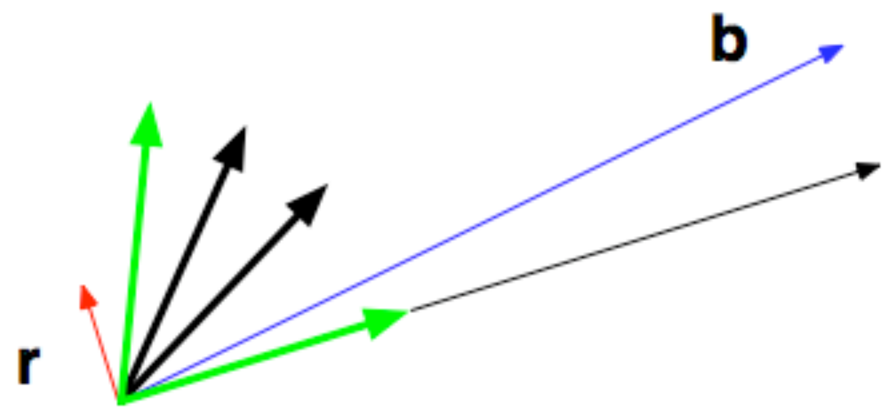
Finding sparse solutions:OMP

Orthogonal Matching Pursuit algorithm:



Finding sparse solutions:OMP

Orthogonal Matching Pursuit algorithm:



One more theoretical result

Theorem: For a system of linear equations **Ax** = **b** (**A** an *n* by *m* matrix, $n < m$, and rank(**A**) = *n*), if a solution **x** exists obeying $||\mathbf{x}||_0 < (1+\mu(\mathbf{A})^{-1})/2$, then an OMP run with threshold parameter $\varepsilon_0 = 0$ is guaranteed to find **x** exactly.

Implementation Fine Tuning

My initial OMP implementation wasn't optimized for speed. I made some improvements:

The core of the algorithm is found in the following three steps. Modifying the approach to each of them cut execution times considerably.

- **Sweep:** Compute the errors $\epsilon(j) = \min_{z_j} ||z_j \mathbf{a}_j - \mathbf{r}^{k-1}||_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / ||\mathbf{a}_j||_2^2$.
- **Update Support:** Find a minimizer j_0 of $\epsilon(j)$: $\forall j \notin \mathcal{S}^{k-1}$, $\epsilon(j_0) \leq \epsilon(j)$, and update $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$.
- **Update Provisional Solution:** Compute $\mathbf{x}^k$, the minimizer of $||\mathbf{Ax} - \mathbf{b}||_2^2$ subject to $Support\{\mathbf{x}\} = \mathcal{S}^k$.

Implementation Fine Tuning, Round 1: ompQRf

The first improvement came from computing norm($\mathbf{r}_{k-1}$) |cos(θ_j)|, where θ_j is the angle between $\mathbf{a}_j$ and $\mathbf{r}_{k-1}$. This number reflects how good an approximation to the residue z_j $\mathbf{a}_j$ is, and it is faster to compute than ε(j).

We also kept track of the best approximant during the "Sweep" so that "Update Support" is done in a more efficient way compared to what we had done in ompQR.

Finally, we sweep only on the set of columns that have not been added to the support set, resulting in further time gains on the "Sweep" step when $k > 1$.

Implementation Fine Tuning, Round 2: ompQRf2

The “Update Provisional Solution” involves an l_2 minimization that corresponds to a least squares approximation. The preferred method of choice in this case is a *QR* decomposition of the restricted system.

We implemented this part of the algorithm by taking advantage of previous *QR* steps as opposed to compute each time a brand new *QR* decomposition of the updated matrix that resulted from increasing the support set S^k.

Implementation Fine Tuning, Round 3: ompQRf3

Finally, we heed the advice of Matlab to allocate some variables for speed, this change saves time too:

Runtimes for 'experiment.m' (k = 2)

ompQR	617.802467 seconds
ompQRf	360.192118 seconds, 1.715 speedup
ompQRf2	308.379138 seconds, 1.168 speedup
ompQRf3	298.622174 seconds, 1.032 speedup

Total speedup from ompQR to ompQRf3: 2.068
(Matlab version 2010b)

Implementation and Validation

In light of the theoretical results, we can envision the following roadmap to validate an implementation of OMP.

- We have a simple theoretical criterion to guarantee both solution uniqueness and OMP convergence:

If $\mathbf{x}$ *is a solution to* $\mathbf{Ax} = \mathbf{b}$, *and* $||\mathbf{x}||_0 < (1+\mu(\mathbf{A})^{-1})/2$,
then $\mathbf{x}$ *is the unique sparsest solution to* $\mathbf{Ax} = \mathbf{b}$ *and OMP will find it.*

- Hence, given a full-rank n by m matrix $\mathbf{A}$ ($n < m$), compute $\mu(\mathbf{A})$, and find the largest integer k smaller than or equal to $(1+\mu(\mathbf{A})^{-1})/2$. That is, $k = \text{floor}((1+\mu(\mathbf{A})^{-1})/2)$.

Implementation and Validation

- Build a vector **x** with exactly *k* non-zero entries and produce a right hand side vector **b** = **Ax**. This way, you have a known sparsest solution **x** to which to compare the output of any OMP implementation.

- Pass **A**, **b**, and ε_0 to OMP to produce a solution vector $\mathbf{x}_{omp} = \mathrm{OMP}(\mathbf{A},\mathbf{b},\varepsilon_0)$.

- If OMP terminates after *k* iterations and $||\mathbf{A}\mathbf{x}_{omp} - \mathbf{b}|| < \varepsilon_0$, for all possible **x** and $\varepsilon_0 > 0$, then the OMP implementation would have been validated.

<u>Caveat</u>: The theoretical proofs assume infinite precision.

Validation Results

We ran two experiments:

1) $\mathbf{A} \in R^{100\text{x}200}$, with entries in N(0,1) i.i.d. for which $\mu(\mathbf{A}) = 0.3713$, corresponding to $k = 1 \leq \mathrm{K}$.
2) $\mathbf{A} \in R^{200\text{x}400}$, with entries in N(0,1) i.i.d. for which $\mu(\mathbf{A}) = 0.3064$, corresponding to $k = 2 \leq \mathrm{K}$.

<u>Observations</u>:

- **A** will be full-rank with probability 1.
- For full-rank matrices **A** of size n x m, the mutual coherence satisfies $\mu(\mathbf{A}) \geq \sqrt{\{(m - n)/(n \cdot (m - 1))\}}$. That is, the upper bound of $\mathrm{K} = (1 + \mu(\mathbf{A})^{-1})/2$ can be made as big as needed, provided n and m are big enough.

Validation Results

For each matrix **A**, we chose 100 vectors with *k* non-zero entries whose positions were chosen at random, and whose entries were in N(0,1).

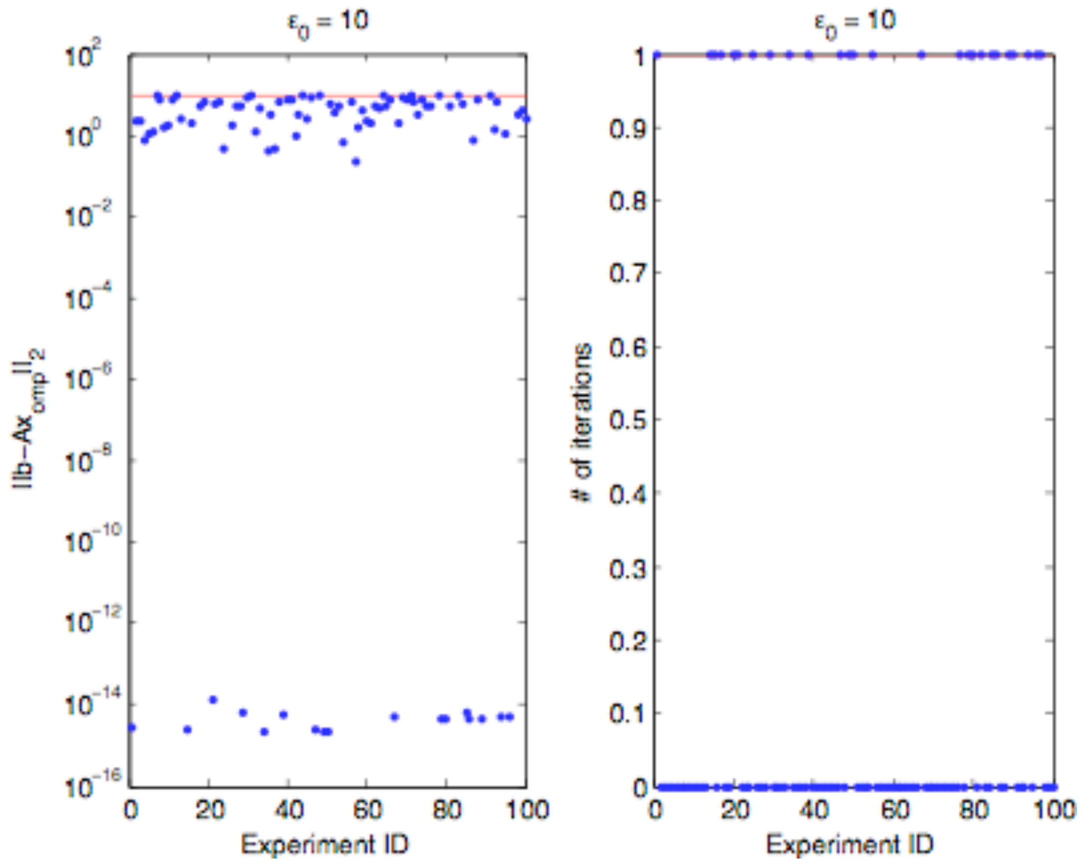
Then, for each such vector **x**, we built a corresponding right hand side vector **b** = **Ax**.

Each of these vectors would then be the unique sparsest solution to **Ax** = **b**, and OMP should be able to find them.

Finally, given $\varepsilon_0 > 0$, if our implementation of OMP were correct, it should stop after *k* steps (or less), and if $\mathbf{x}_{OMP} = OMP(\mathbf{A},\mathbf{b},\varepsilon_0)$, then $||\mathbf{b} - \mathbf{A}\mathbf{x}_{OMP}|| < \varepsilon_0$.

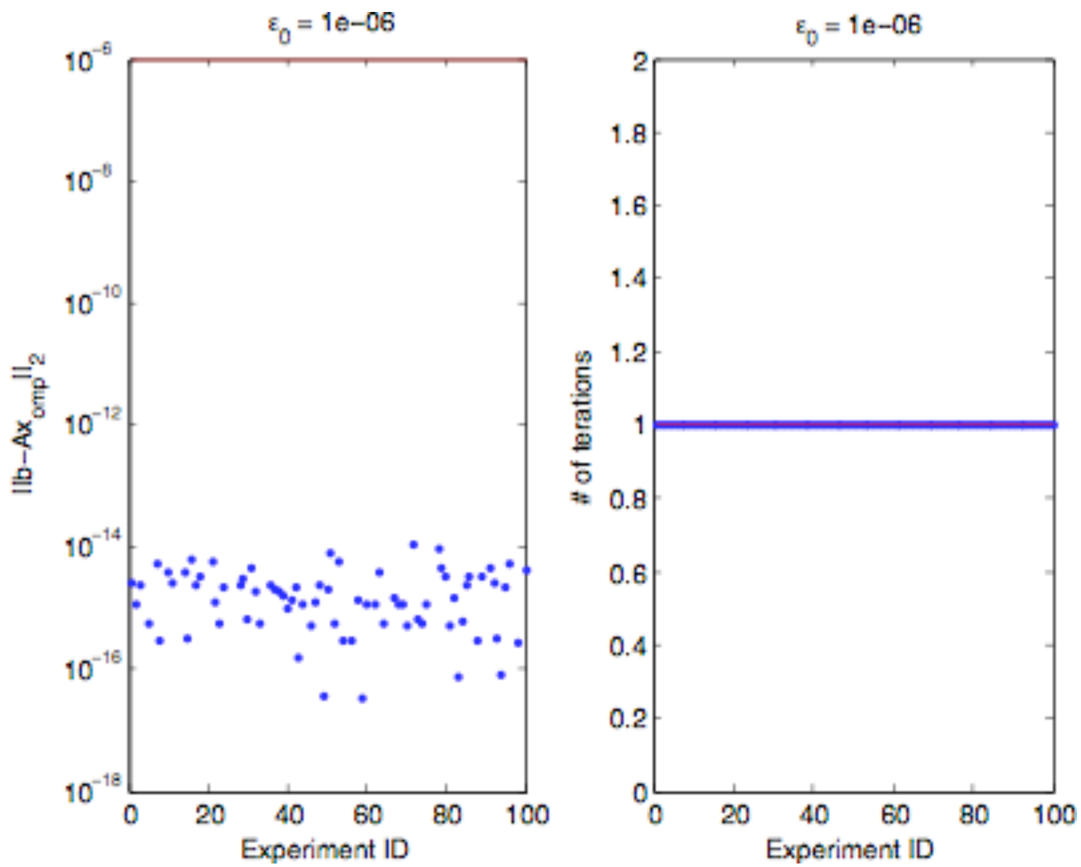
Validation Results

k = 1



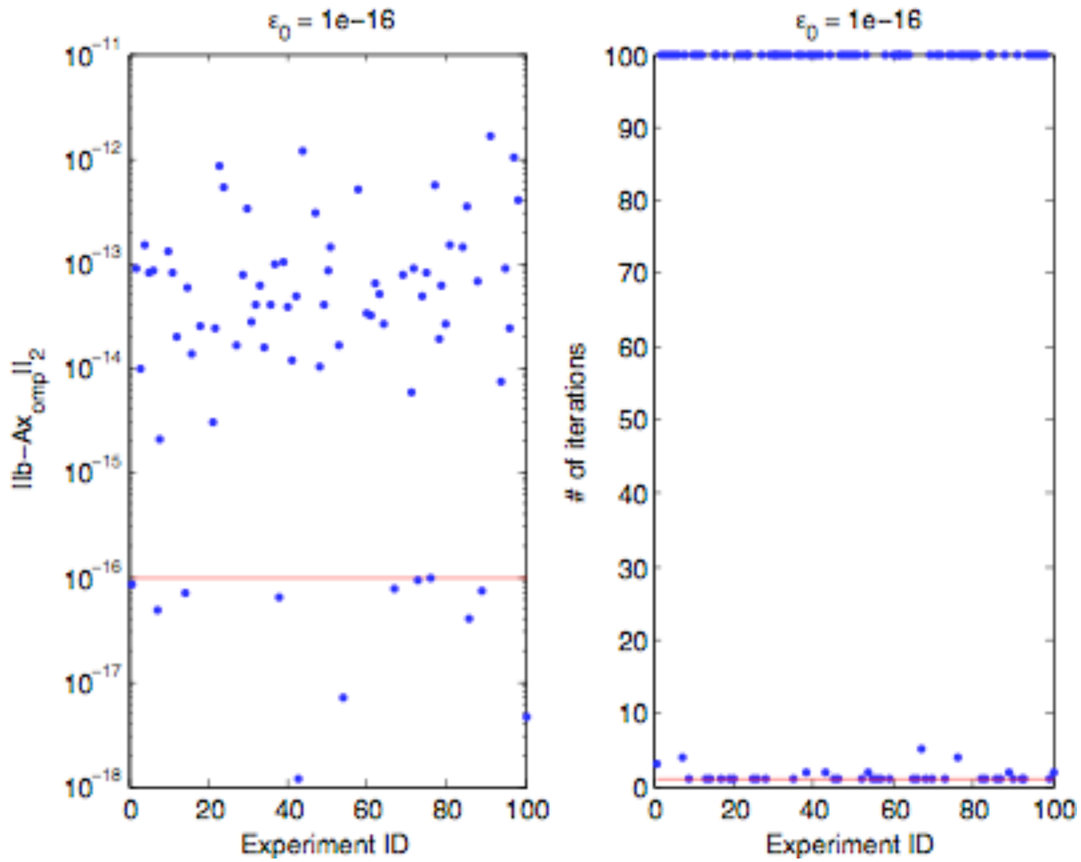
Validation Results

k = 1



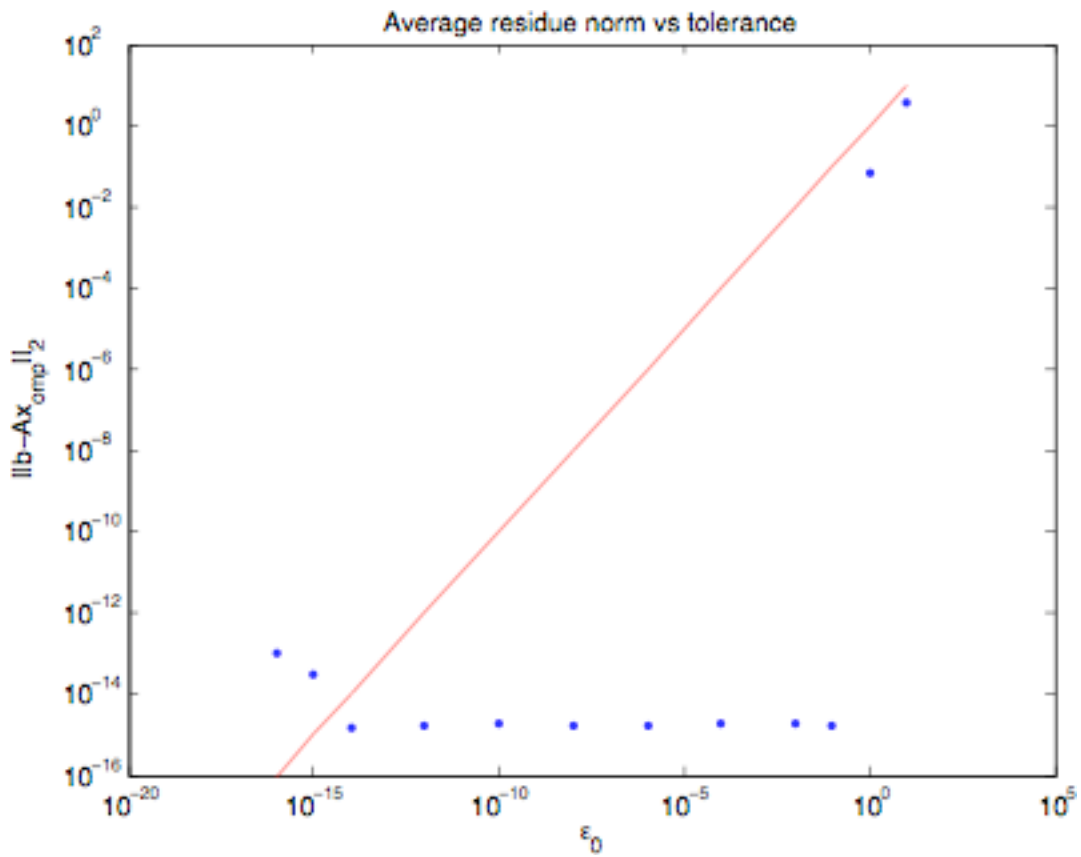
Validation Results

k = 1



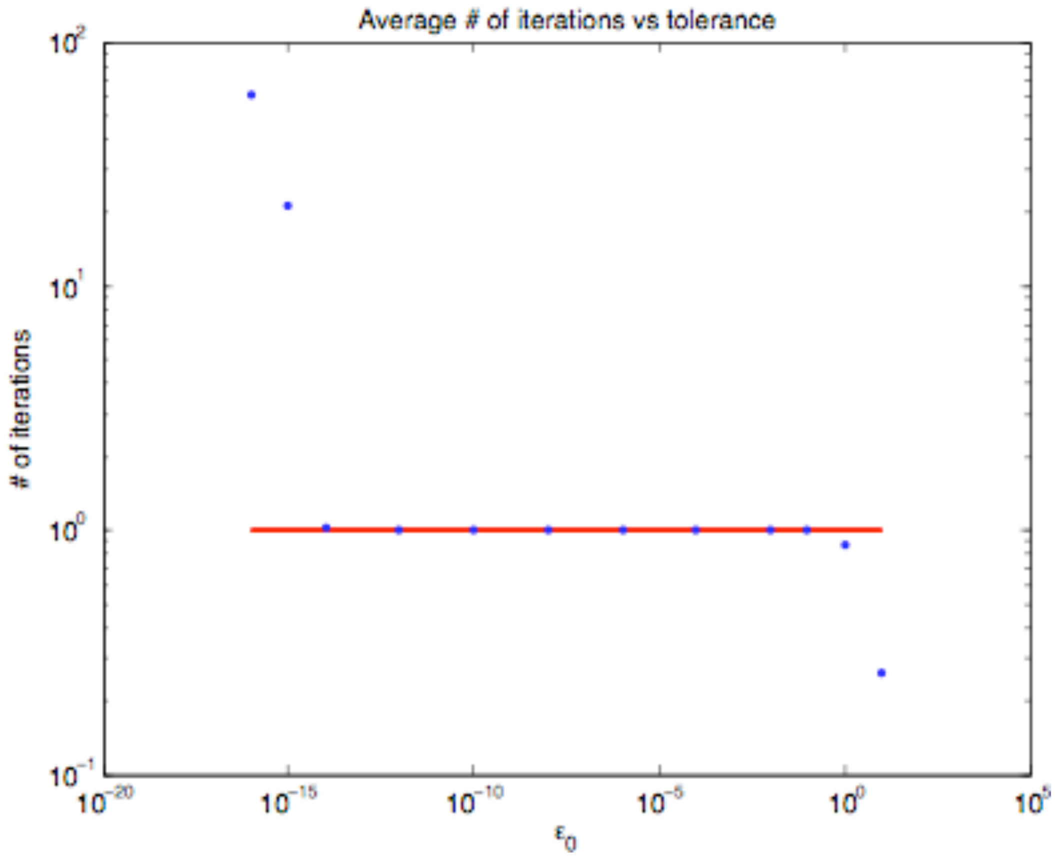
Validation Results

k = 1



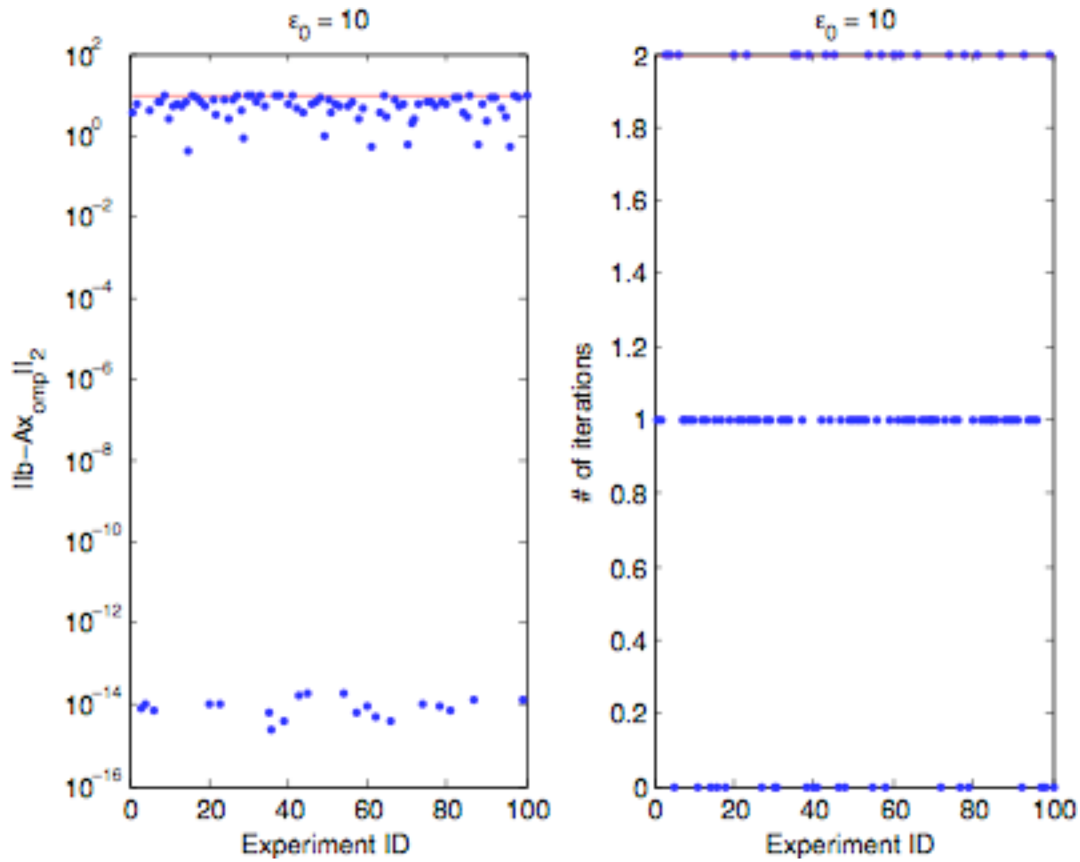
Validation Results

$k = 1$



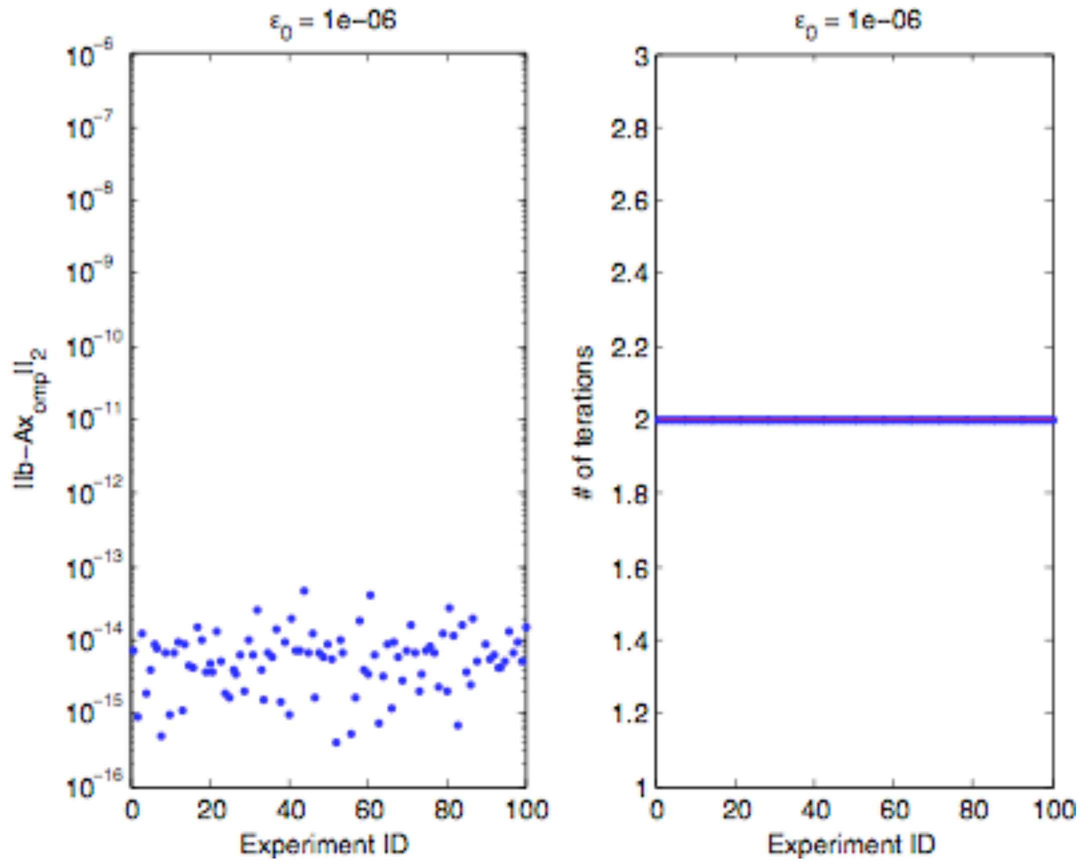
Validation Results

k = 2



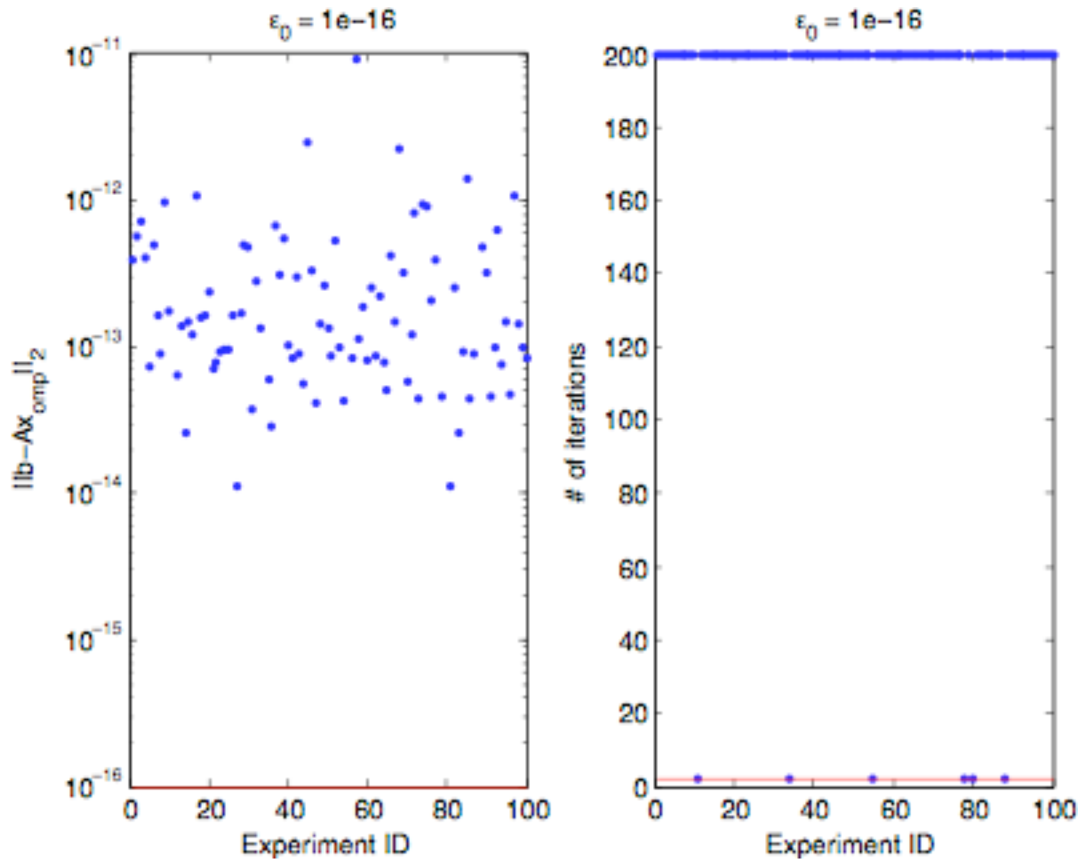
Validation Results

k = 2



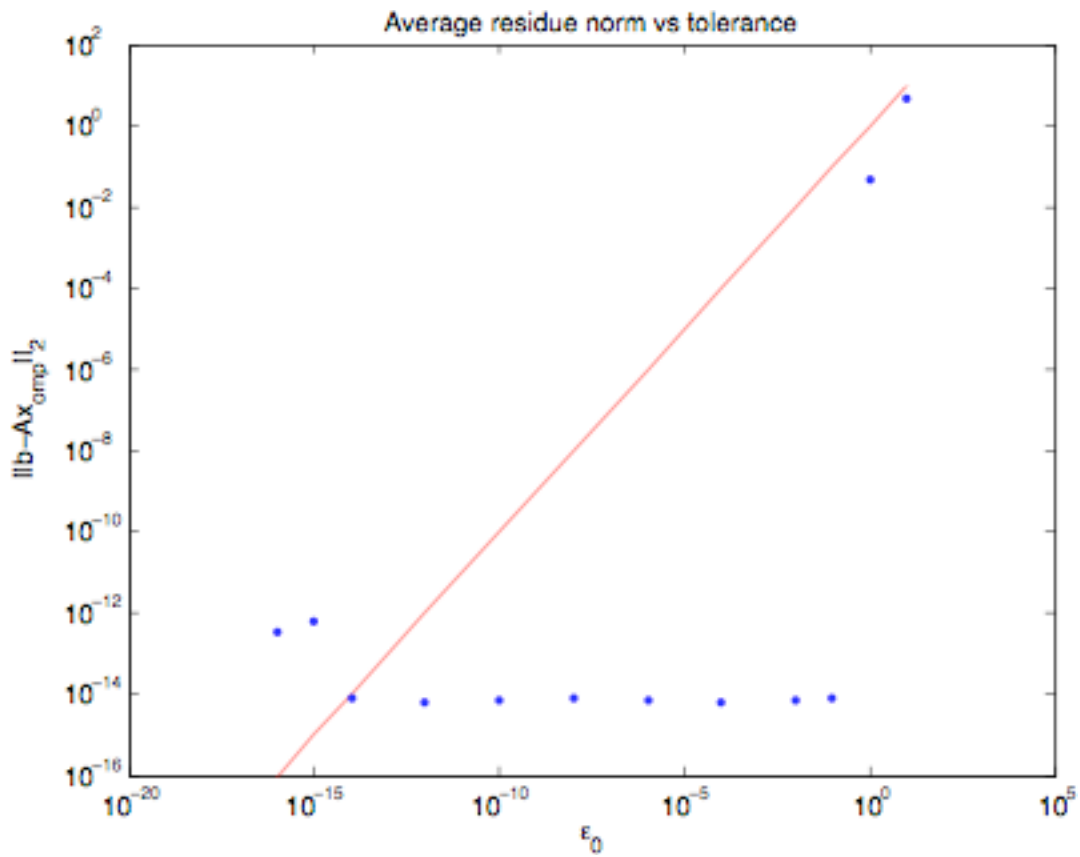
Validation Results

k = 2



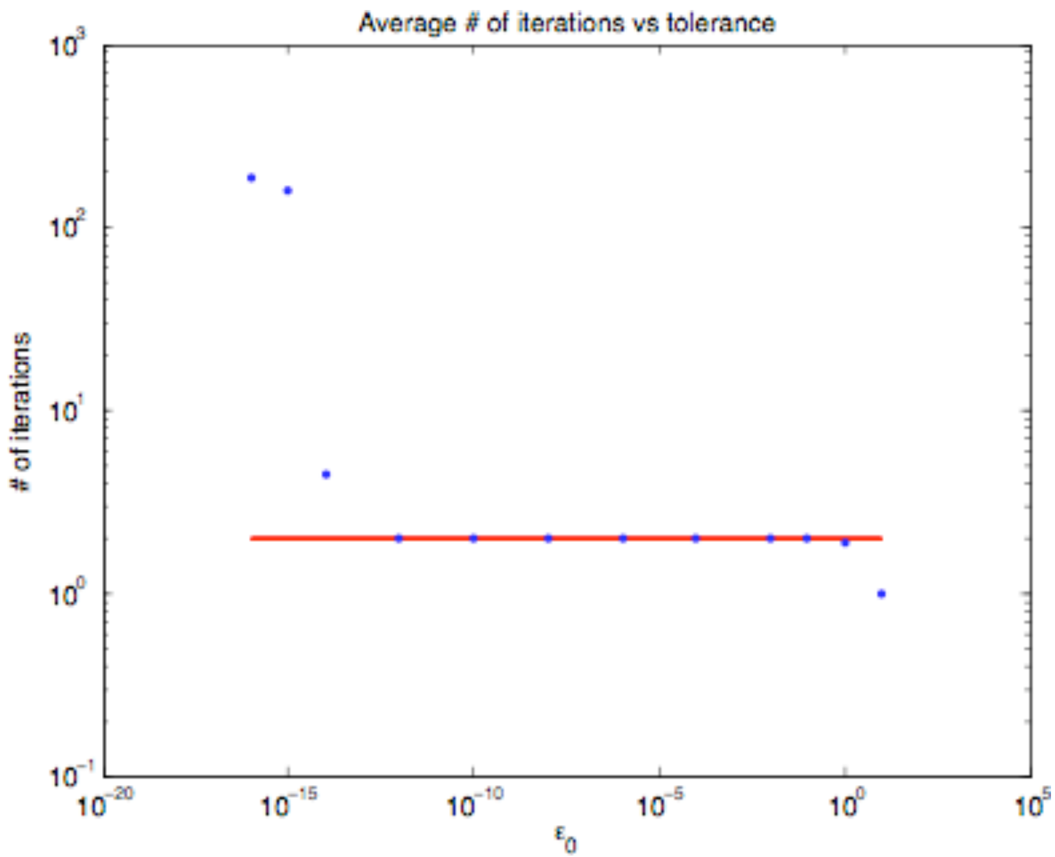
Validation Results

k = 2



Validation Results

k = 2



Reproducing Paper Results

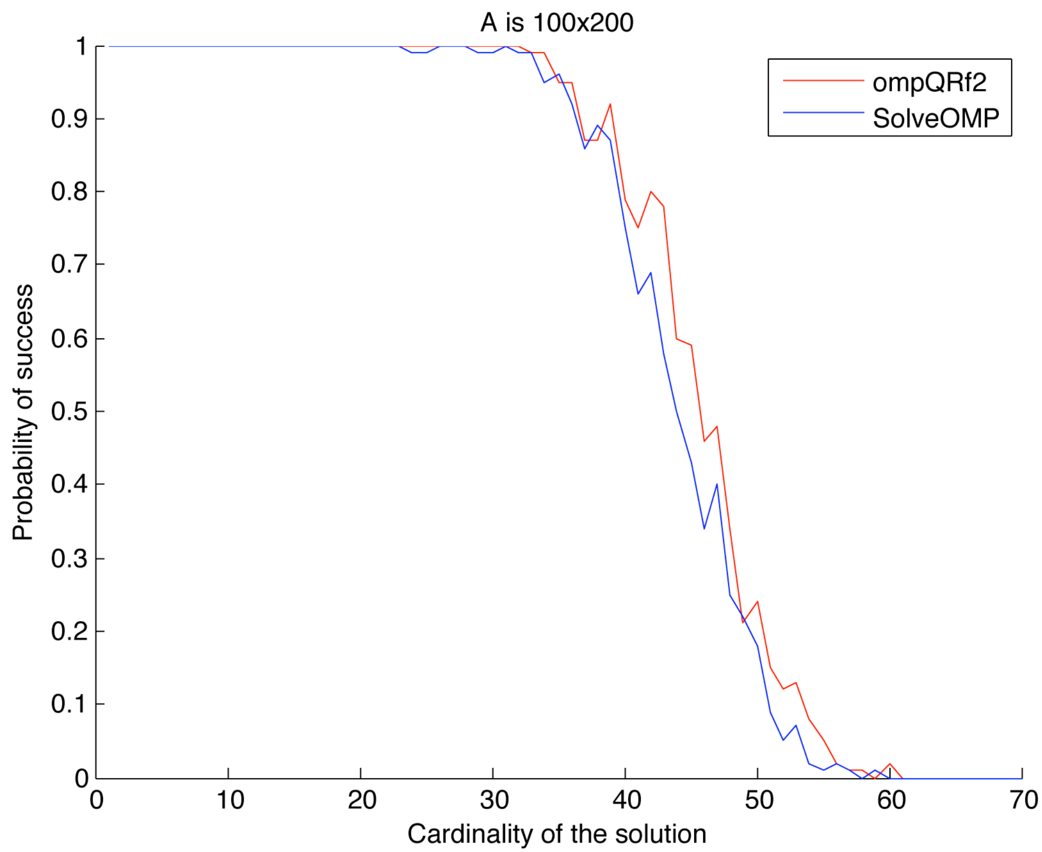
For the first portion of our testing protocol, we set to reproduce the experiment described in section (3.3.1) of [1], limited to the results obtained for OMP.

$\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is 100 x 200, each column i.i.d. N(0,1), and $\mathbf{x}$ has k non-zero entries chosen at random and i.i.d. N(0,1).

Repeat 100 times, for each k = 1 to 70, the following experiment and count the number of successes:
With $\mathbf{b}$ having been set to $\mathbf{Ax}$, does $\mathbf{x}_{omp} = omp(\mathbf{A},\mathbf{b},1e\text{-}5)$ converge to $\mathbf{x}$ within the given tolerance?

[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, *From sparse solutions of systems of equations to sparse modeling of signals and images*, SIAM Review, 51 (2009), pp. 34–81.

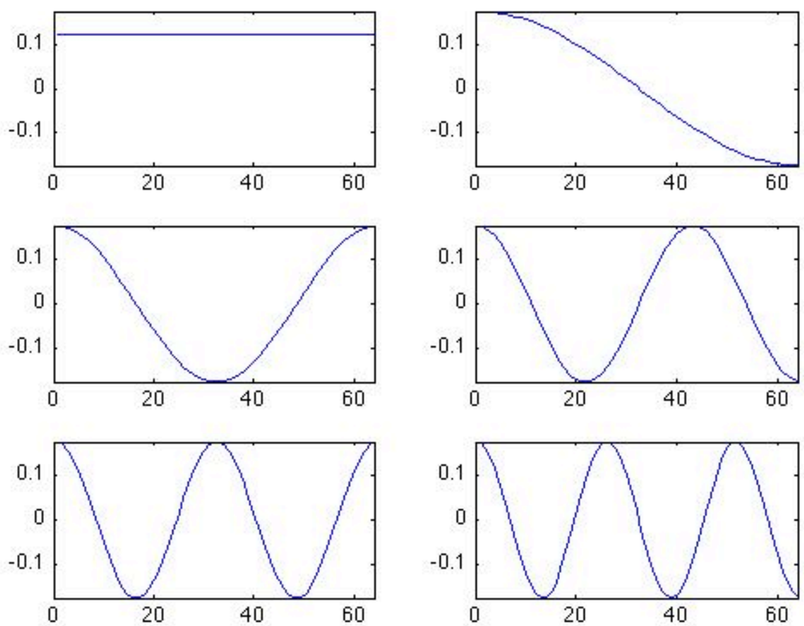
Reproducing Paper Results



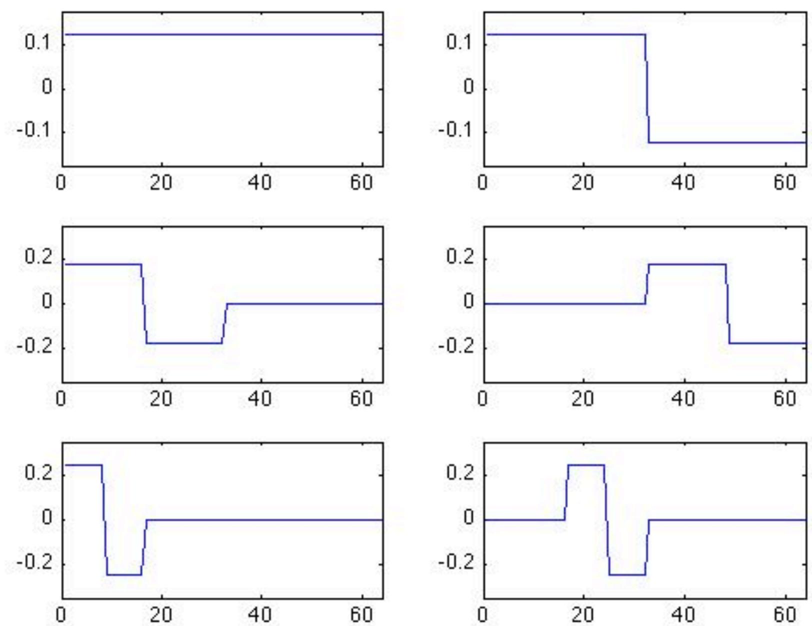
SolveOMP is SparseLab's implementation of OMP (http://sparselab.stanford.edu/)

Image Compression: setup

We need a matrix **A**, and we consider the basis of Discrete Cosine Transform waveforms, and the basis generated by the Haar wavelet. JPEG, JPEG2000 inspired.



DCT



Haar

Image Compression: setup

We are going to partition an image in smaller square sub-matrices to be linearized.

How to linearize a matrix?

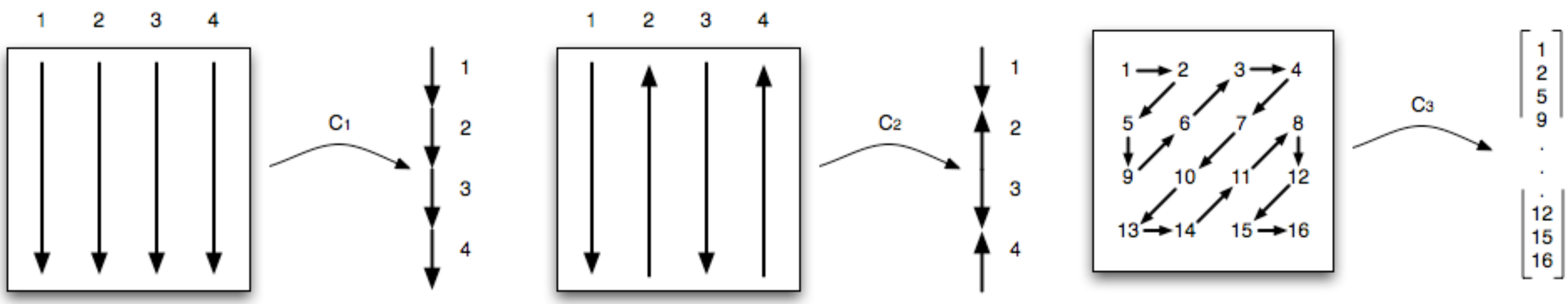
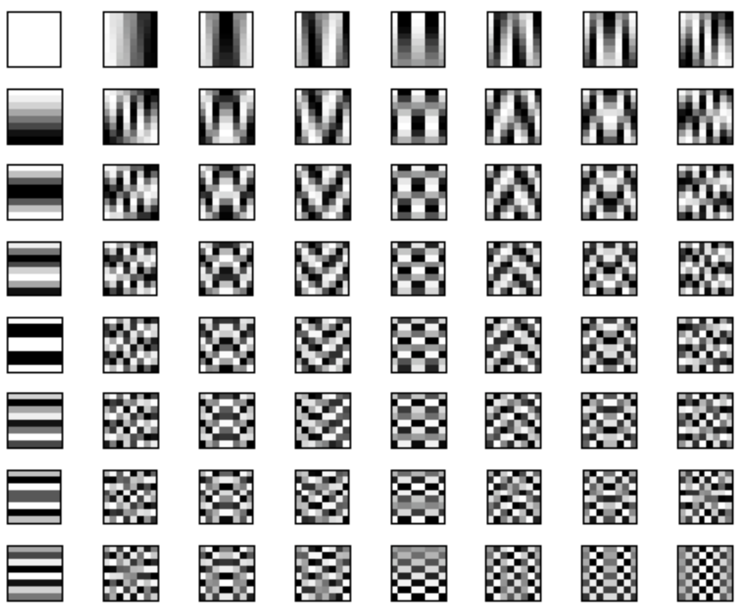
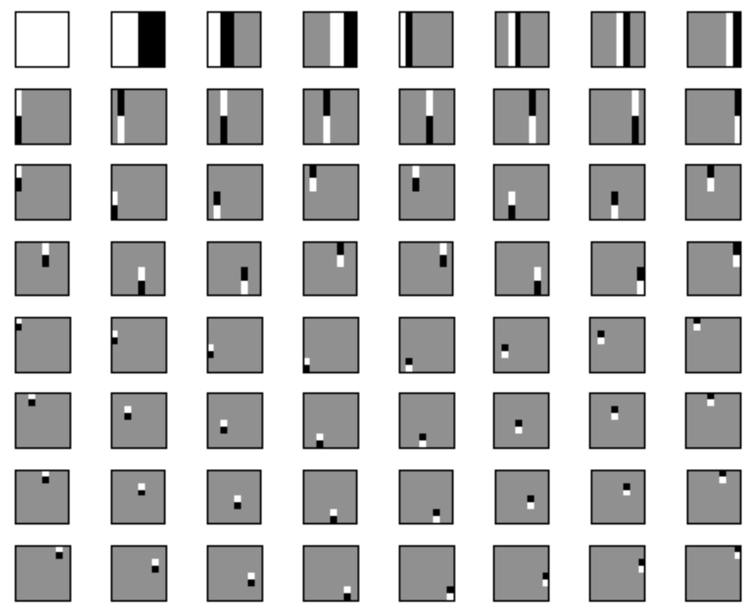


Image Compression: setup

Consider the matrix **A** = [DCT_1 $Haar_1$], where DCT_1 is the basis of 1-dimensional DCT waveforms, and $Haar_1$ is the basis of 1-dimensional Haar wavelet waveforms.



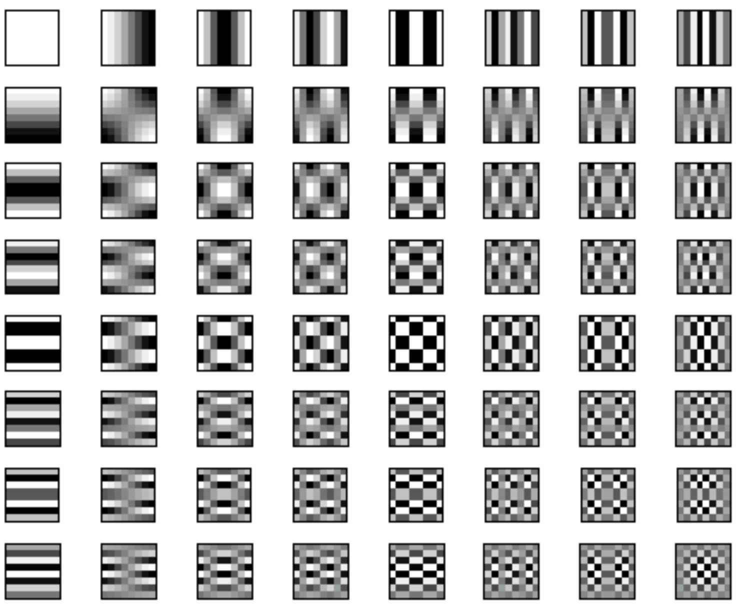
DCT_1



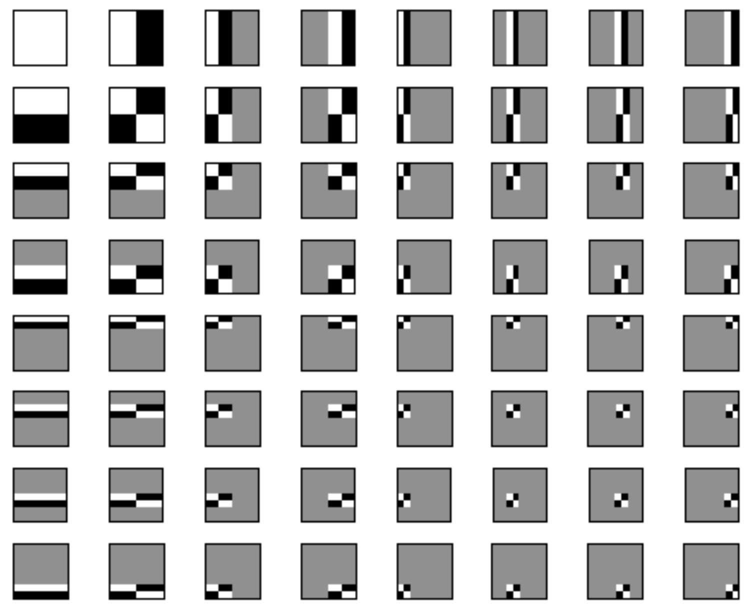
$Haar_1$

Image Compression: setup

Consider the matrix **A** = [$DCT_{2,3}$ $Haar_{2,3}$], where $DCT_{2,3}$ is the basis of 2-dimensional DCT waveforms, and $Haar_{2,3}$ is the basis of 2-dimensional Haar wavelet waveforms.



DCT_2



$Haar_2$

Image Compression: images

We selected 5 natural images to test the compression properties of **A**, and compare to compression via DCT or Haar alone, i.e. **B** = [DCT], or **C** = [Haar]

Barbara

Boat

Elaine

Peppers

Stream

Normalized bit-rate

To study the tradeoff between error and compression, we need to introduce a measure of how many bits it takes to store our image. If our image I is composed of M sub-images I_j, and each can be represented by $\mathbf{x}_j$, where $j = 1,..., M$. Then the normalized bit-rate is

$$\text{nbr}(I,\mathbf{A},\varepsilon) = \sum ||\mathbf{x}_j||_0/(n_1\ n_2),$$

where each sub-image I_j is of size $n_1\ n_2$.

Image Compression: Barbara

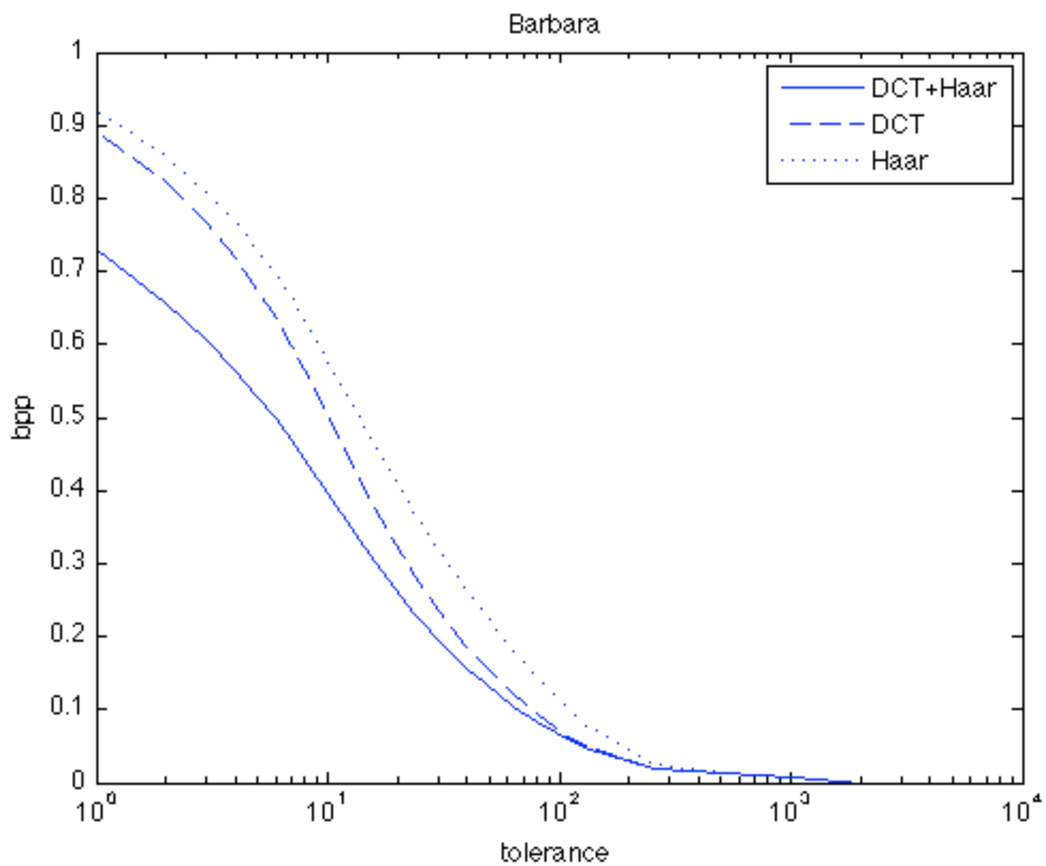


Image Compression: Boat

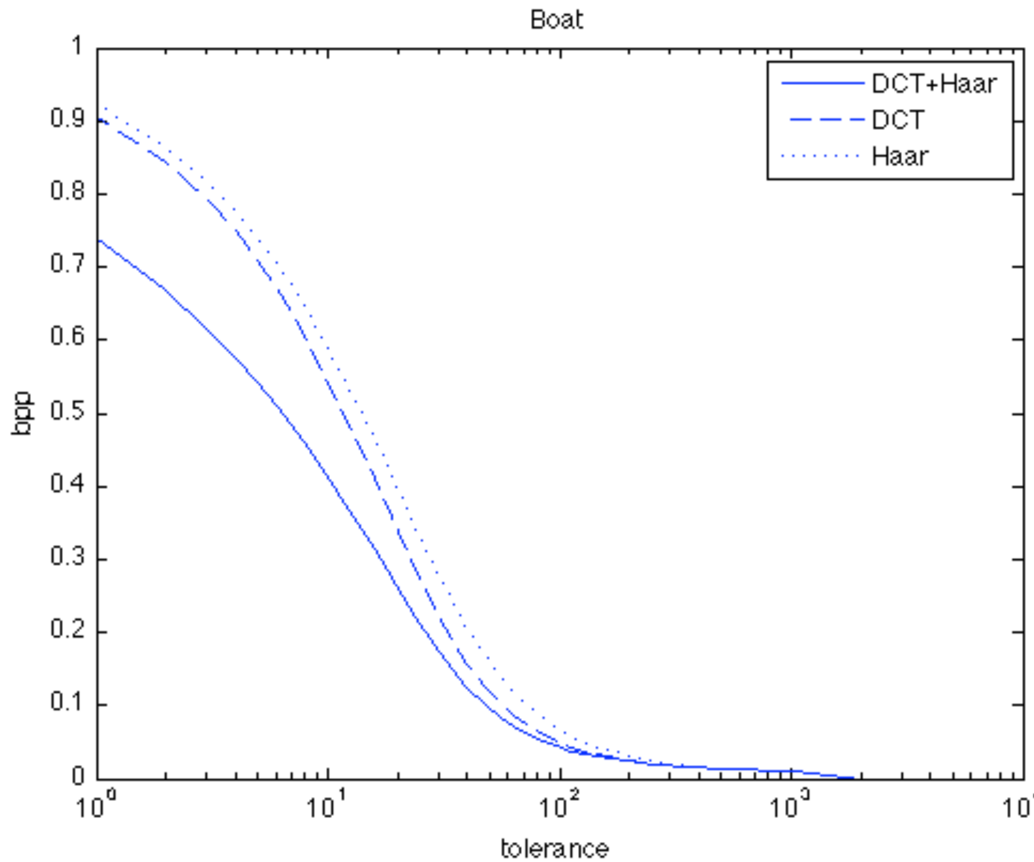


Image Compression: Elaine

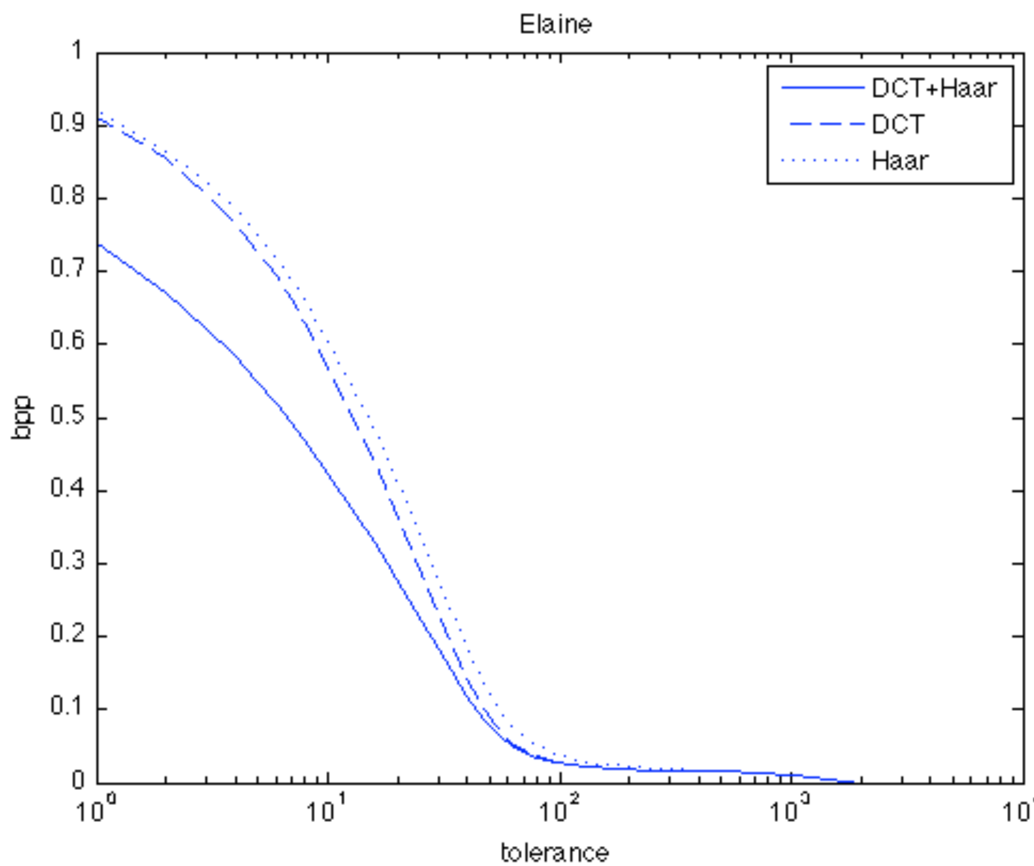


Image Compression: Peppers

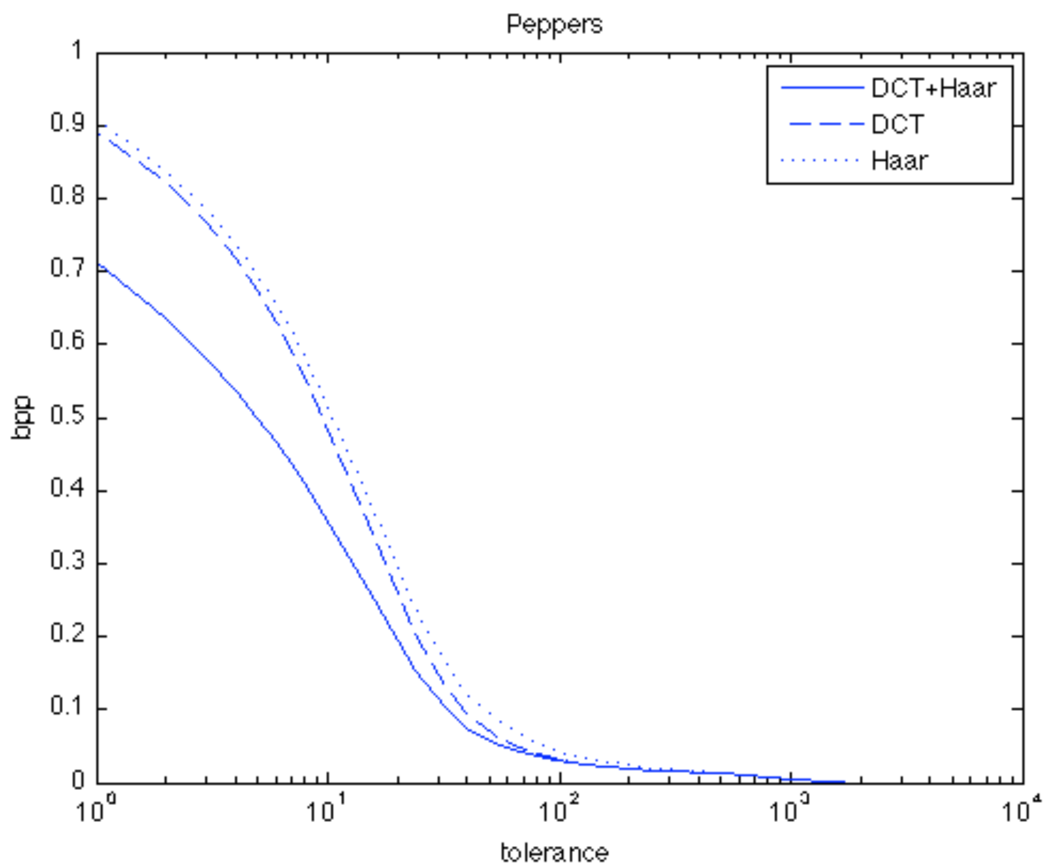
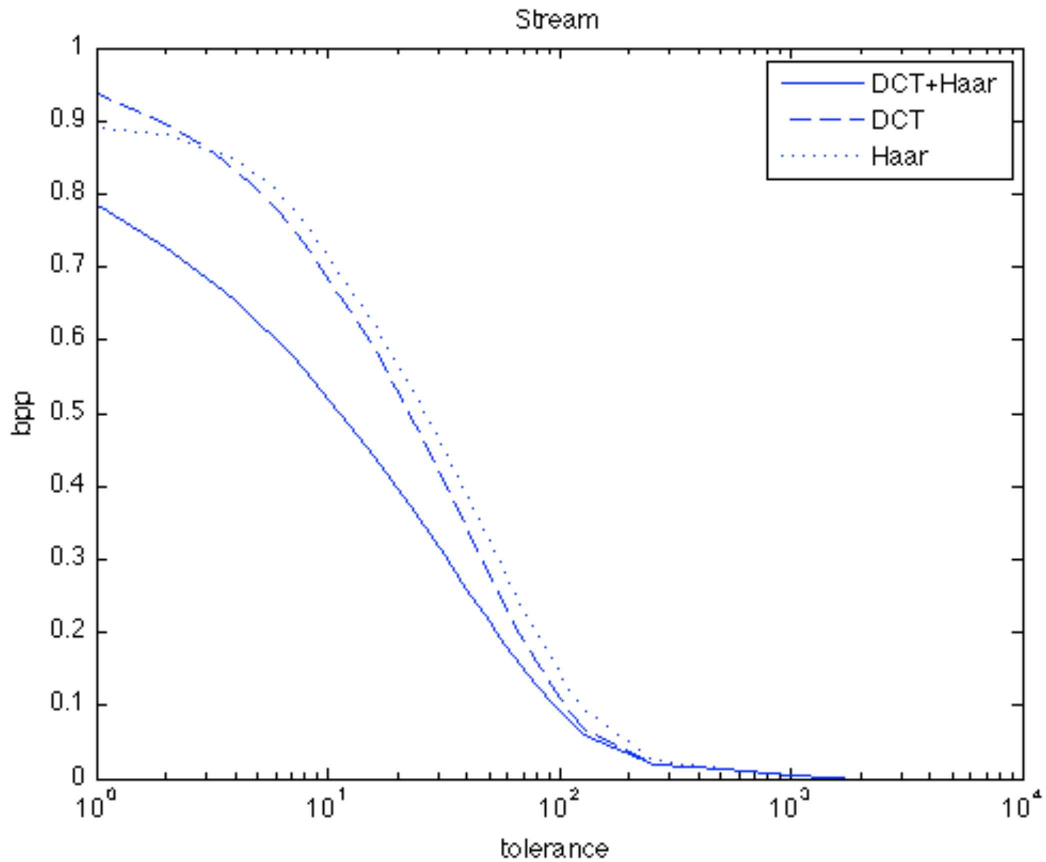


Image Compression: Stream



Error Estimation

Peak Signal-to-Noise Ratio (PSNR):

PSNR = 20 $\log_{10}(MAX_{\mathbf{X}} / \sqrt{MSE})$, (units in dB)

with $MAX_{\mathbf{X}} = 255$, and MSE = $\Sigma_{i,j} (\mathbf{X}(i,j) - \mathbf{Y}(i,j))^2 / nm.$

Structural Similarity (SSIM), and Mean Structural Similarity(MSSIM) indices:

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\,\mu_x\,\mu_y + C_1)\,(2\,\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)\,(\sigma_x^2 + \sigma_y^2 + C_2)}$$

$$\mathrm{MSSIM}(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{j=1}^{M} \mathrm{SSIM}(\mathbf{x}_j, \mathbf{y}_j)$$

Error Estimation: PSNR

Ideal error distribution. Consider an LxL image that has been linearized to a vector **b** of length L^2. Assume that the OMP approximation within ε has distributed the error evenly, that is, if $\mathbf{y} = \mathbf{A}\mathbf{x}_{omp}$

$$\| \mathbf{A}\mathbf{x}_{omp} - \mathbf{b} \|_2 < \varepsilon \Leftrightarrow \| \mathbf{y} - \mathbf{b} \|_2^2 < \varepsilon^2$$

$$\Leftrightarrow \sum_{j = 1,\ldots,L^2} (\mathbf{y}_j - \mathbf{b}_j)^2 < \varepsilon^2$$

$$\Leftrightarrow L^2 c^2 < \varepsilon^2$$

$$\Leftrightarrow c < \varepsilon/L$$

That is, if we want to be within c units from each pixel, we have to choose a tolerance ε such that c is equal to ε/L.

Image Compression: PSNR

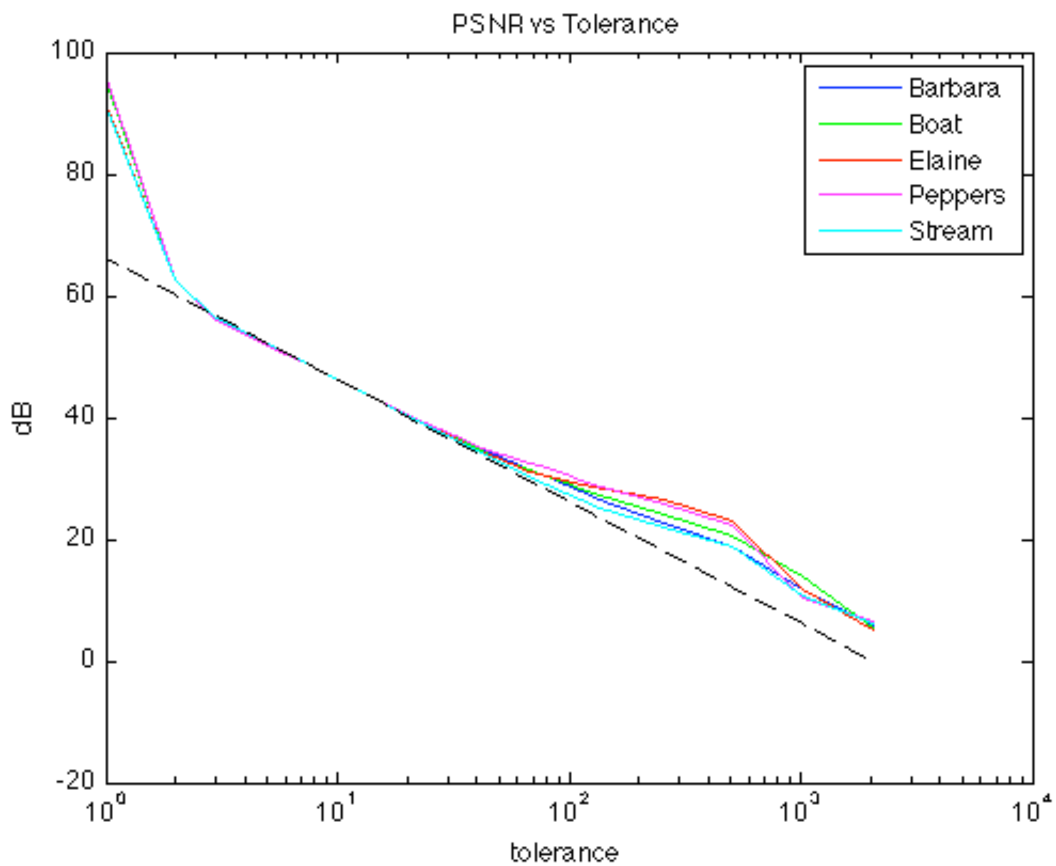


Image Compression: PSNR

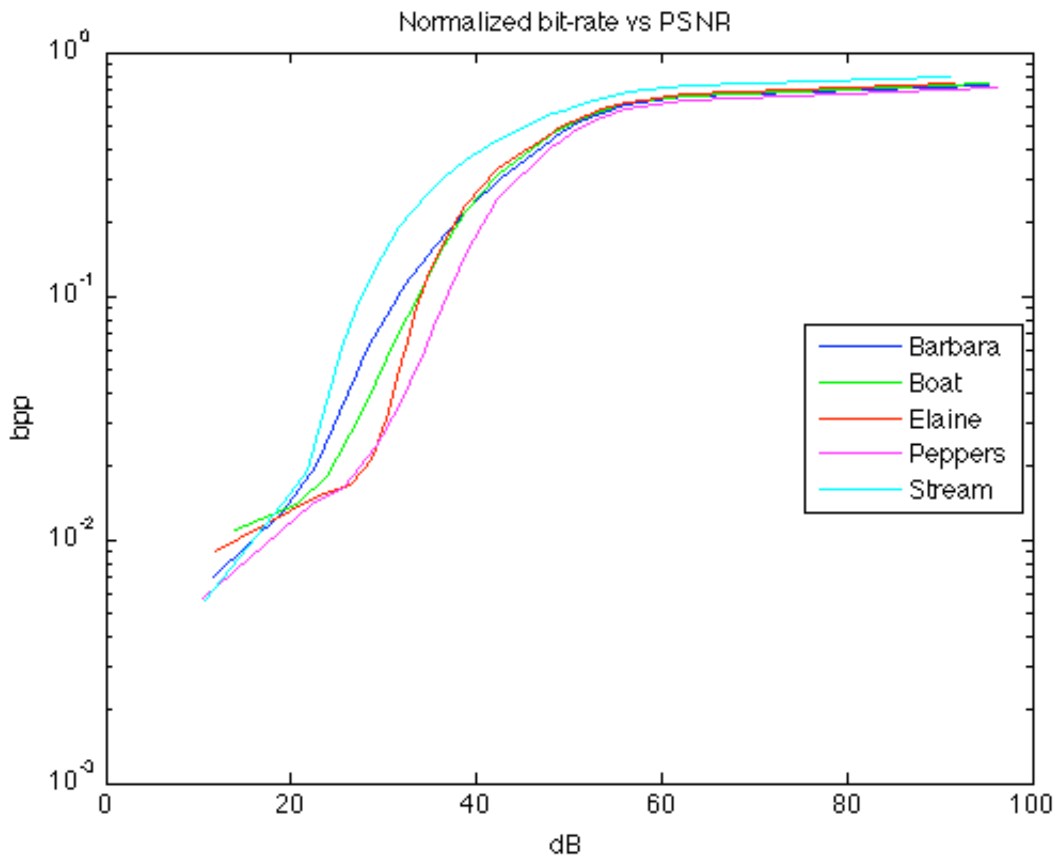


Image Compression: MSSIM

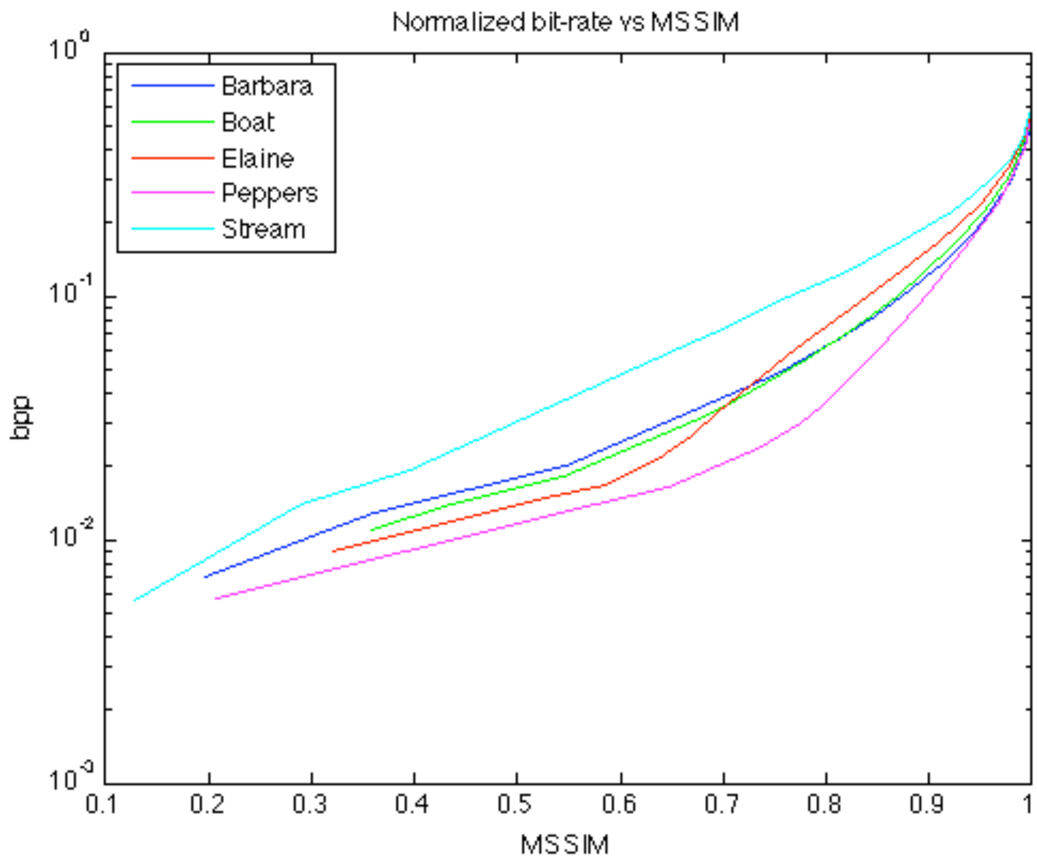
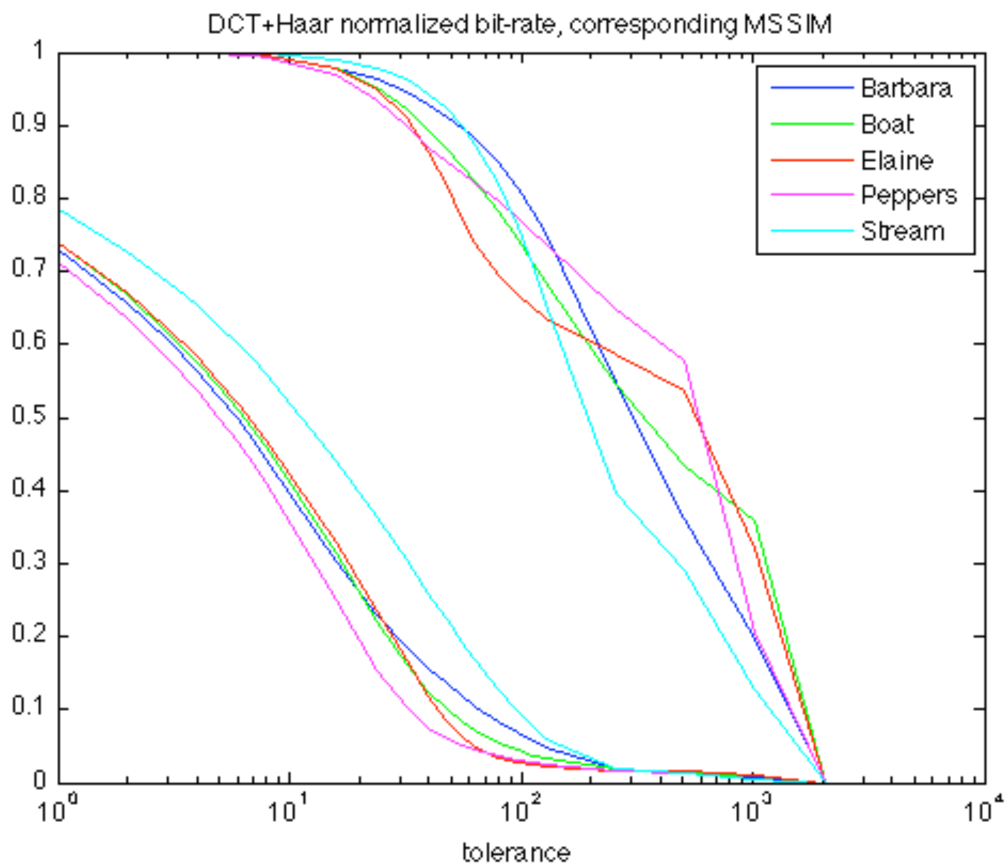
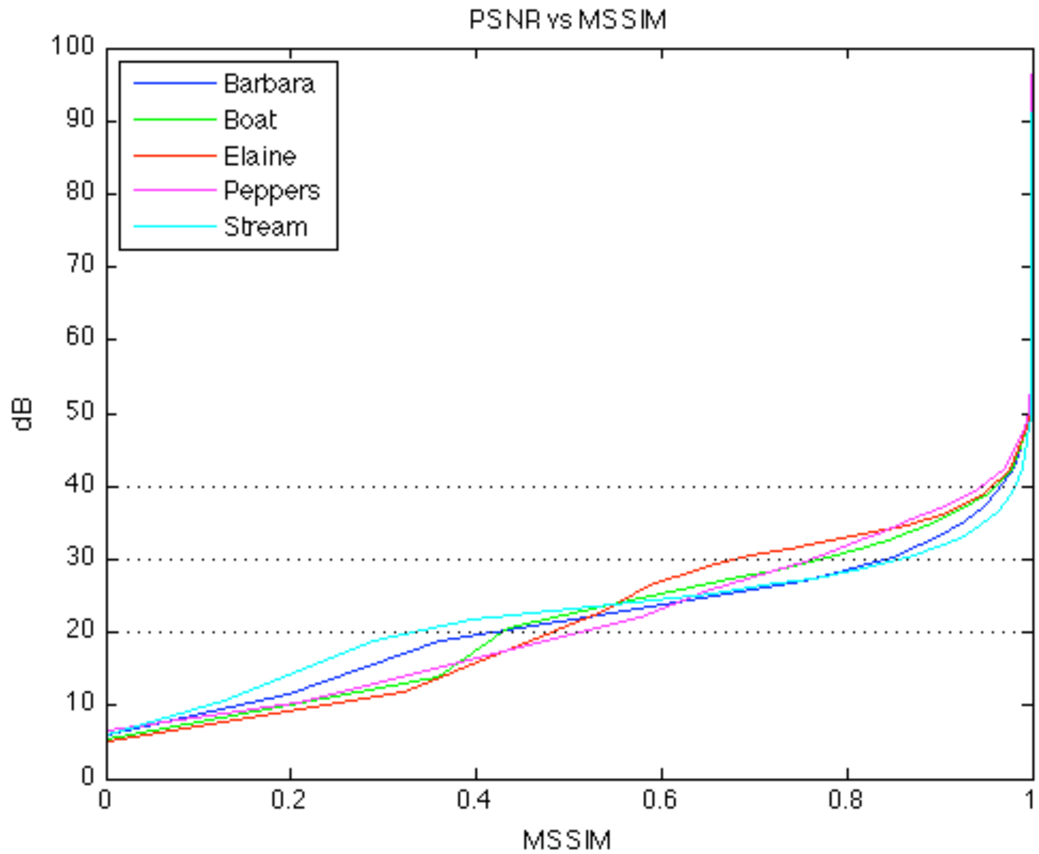


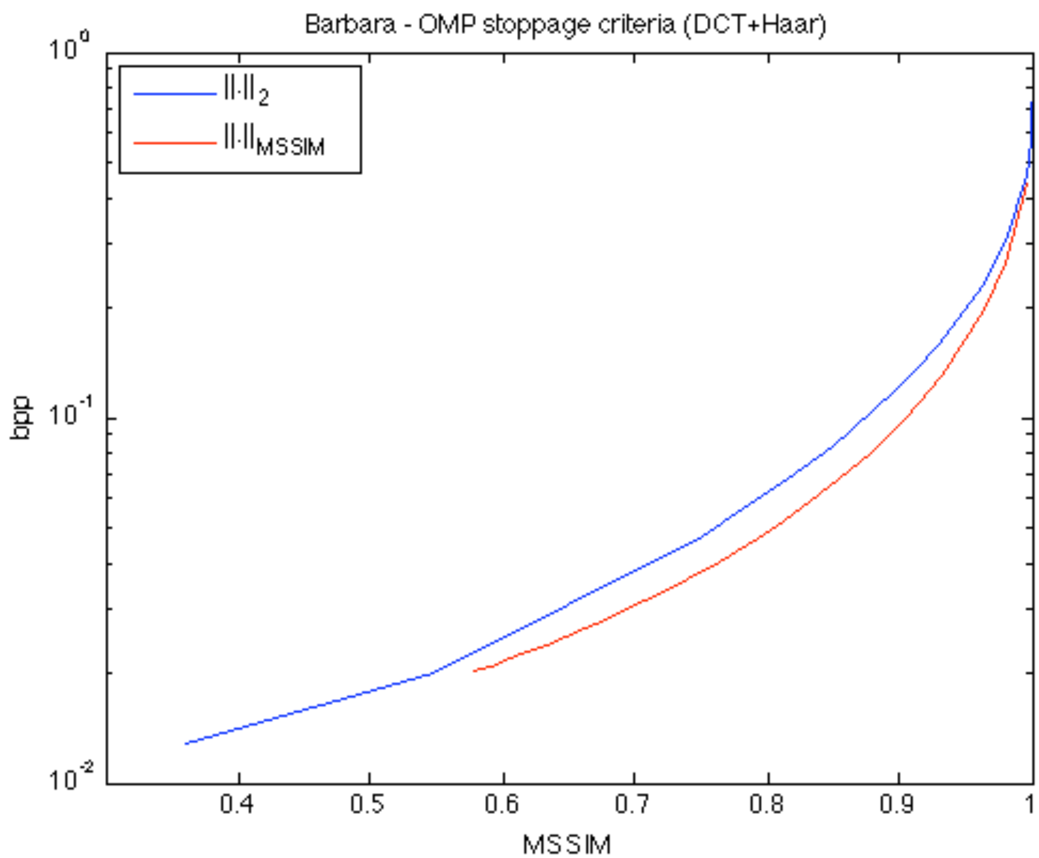
Image Compression: MSSIM



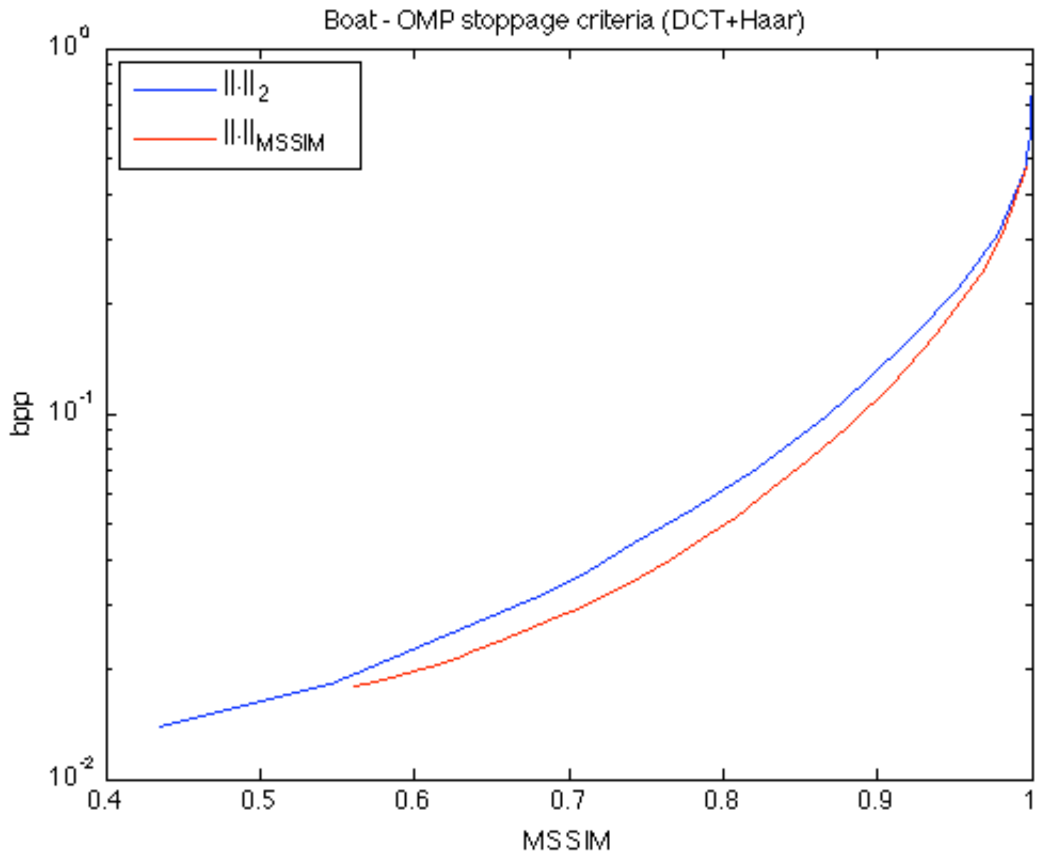
Error Comparison



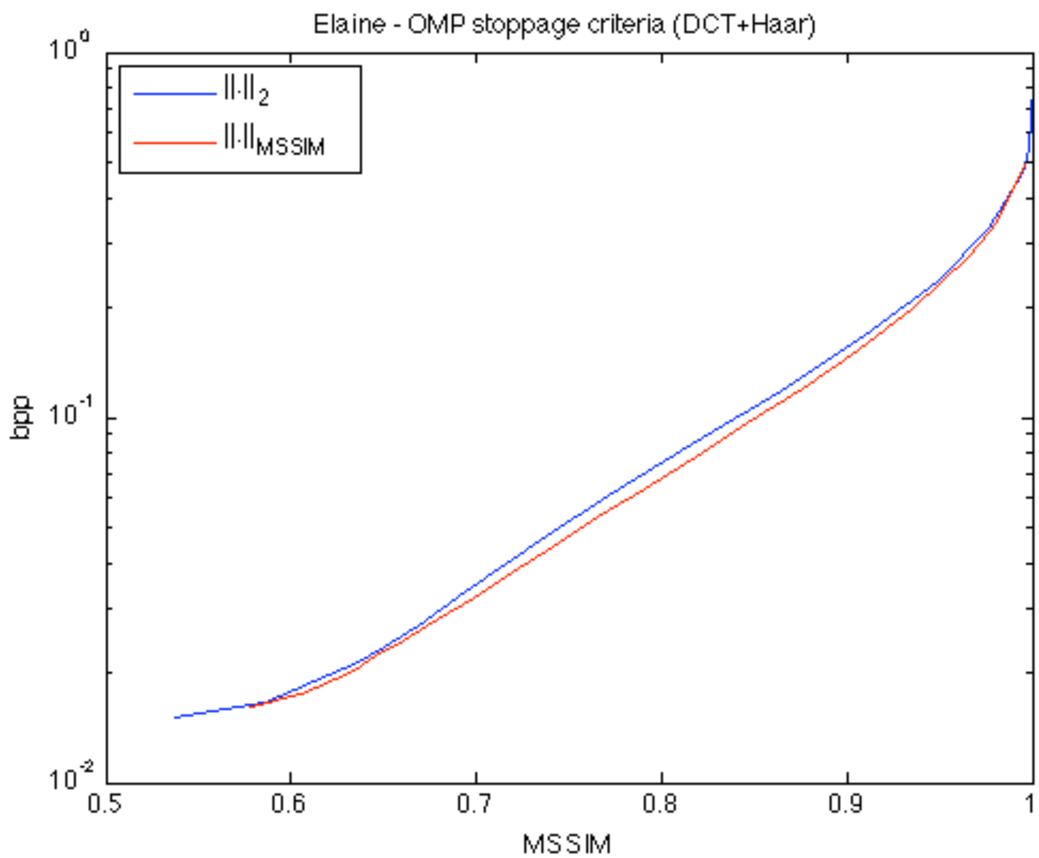
Error Comparison: Barbara



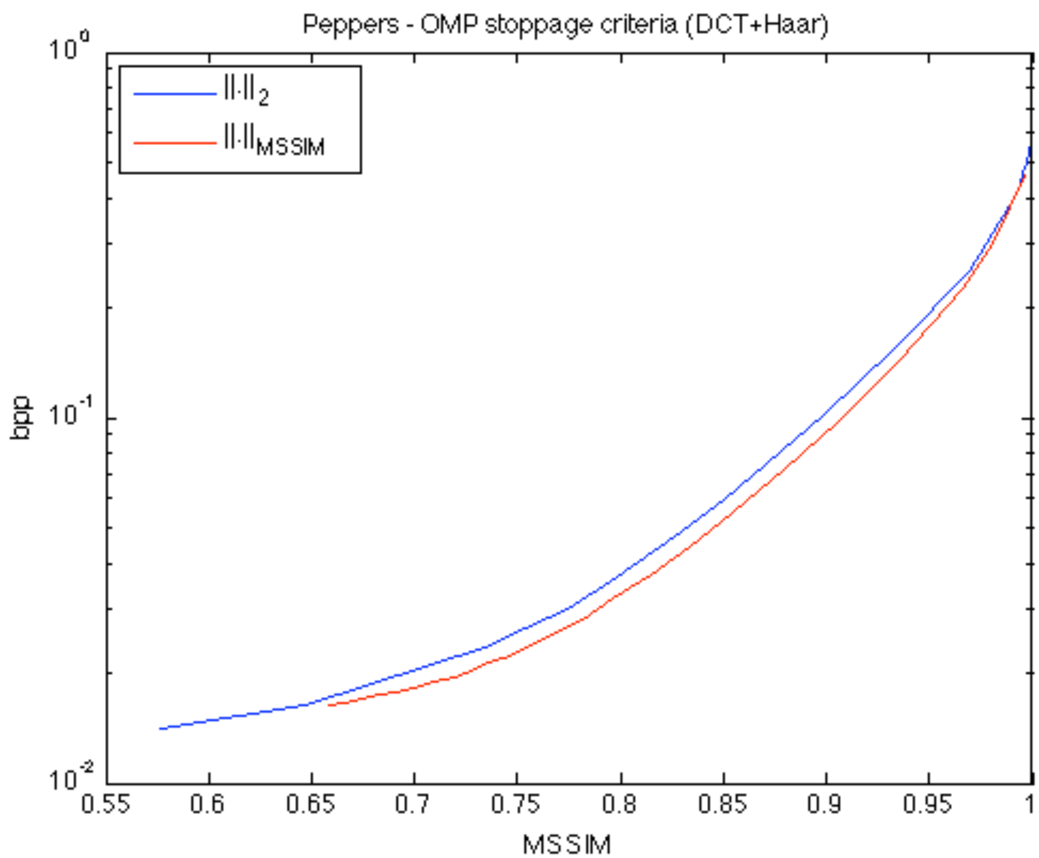
Error Comparison: Boat



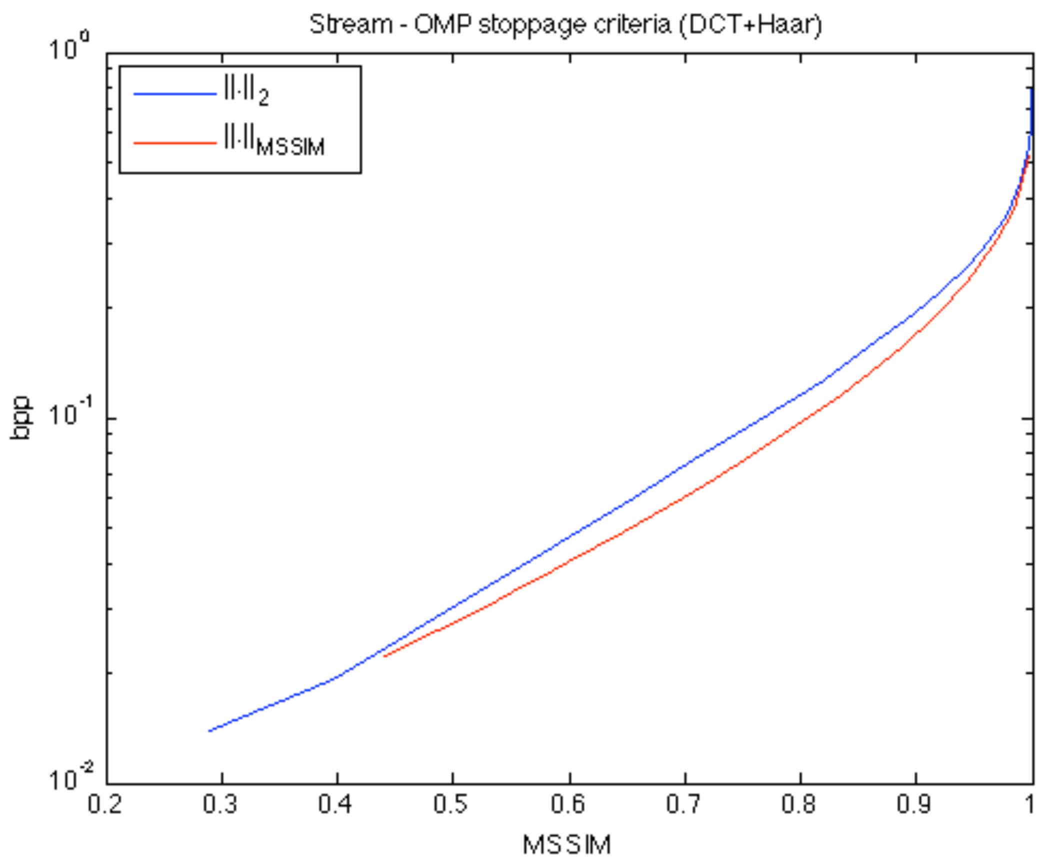
Error Comparison: Elaine



Error Comparison: Peppers

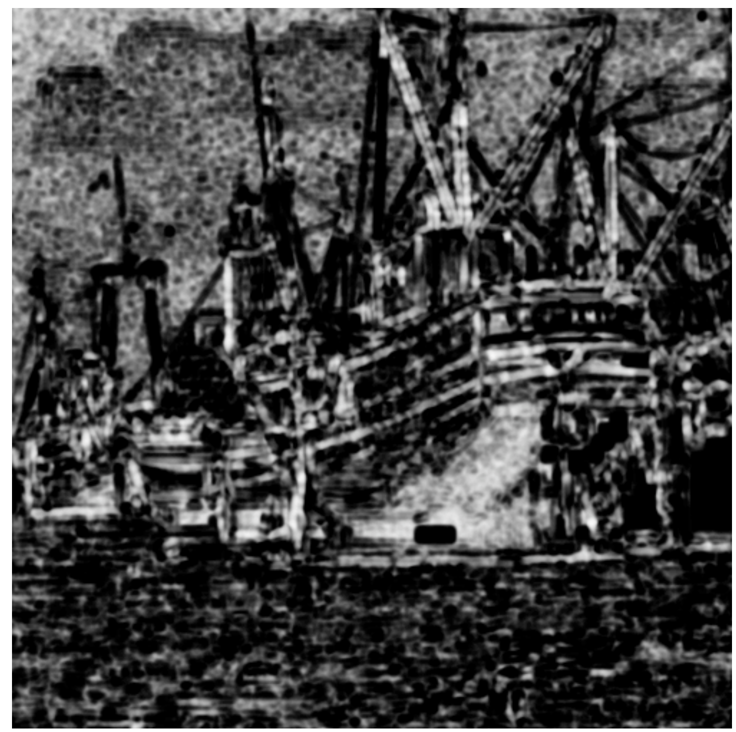


Error Comparison: Stream



Visual overview: Boat

$\varepsilon = 200$, $c = 25$
PSNR = 25.2711 dB
MSSIM = 0.6006
n-bit-rate = 0.0217 bpp
Termination: $||.||_2$



Visual overview: Boat

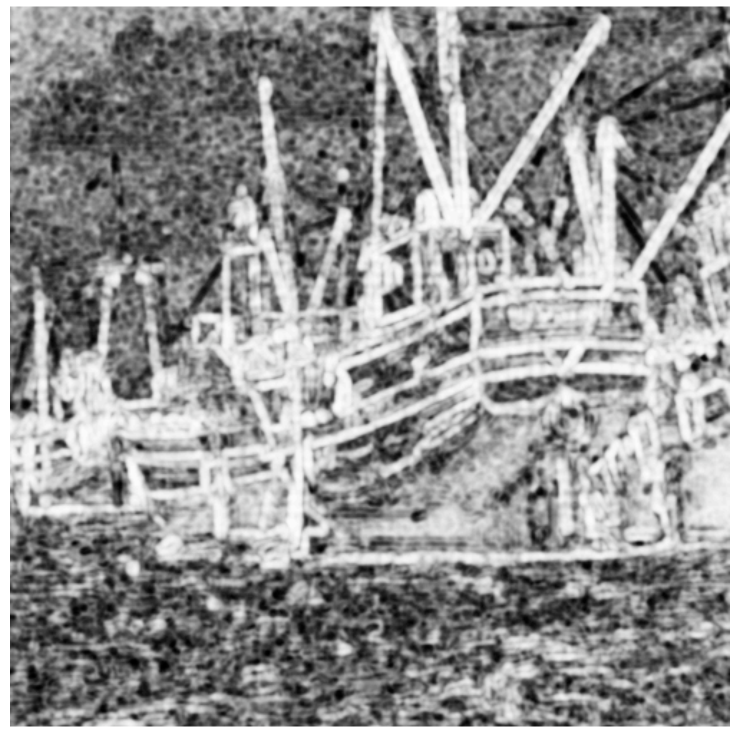
$\varepsilon = 64$, $c = 8$

PSNR = 31.7332 dB

MSSIM = 0.8222

n-bit-rate = 0.0710 bpp

Termination: $||.||_2$



Visual overview: Boat

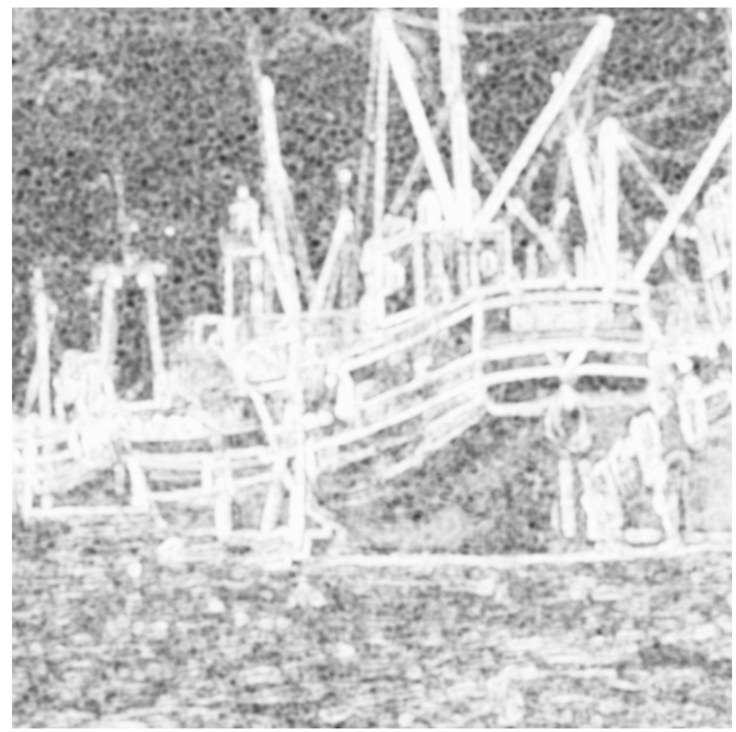
$\varepsilon = 32$, $c = 4$

PSNR = 36.6020 dB

MSSIM = 0.9214

n-bit-rate = 0.1608 bpp

Termination: $||.||_2$



Visual overview: Boat

$\delta = 0.92$
PSNR = 34.1405 dB
MSSIM = 0.9355
n-bit-rate = 0.1595 bpp
Termination: $||.||_{ssim}$

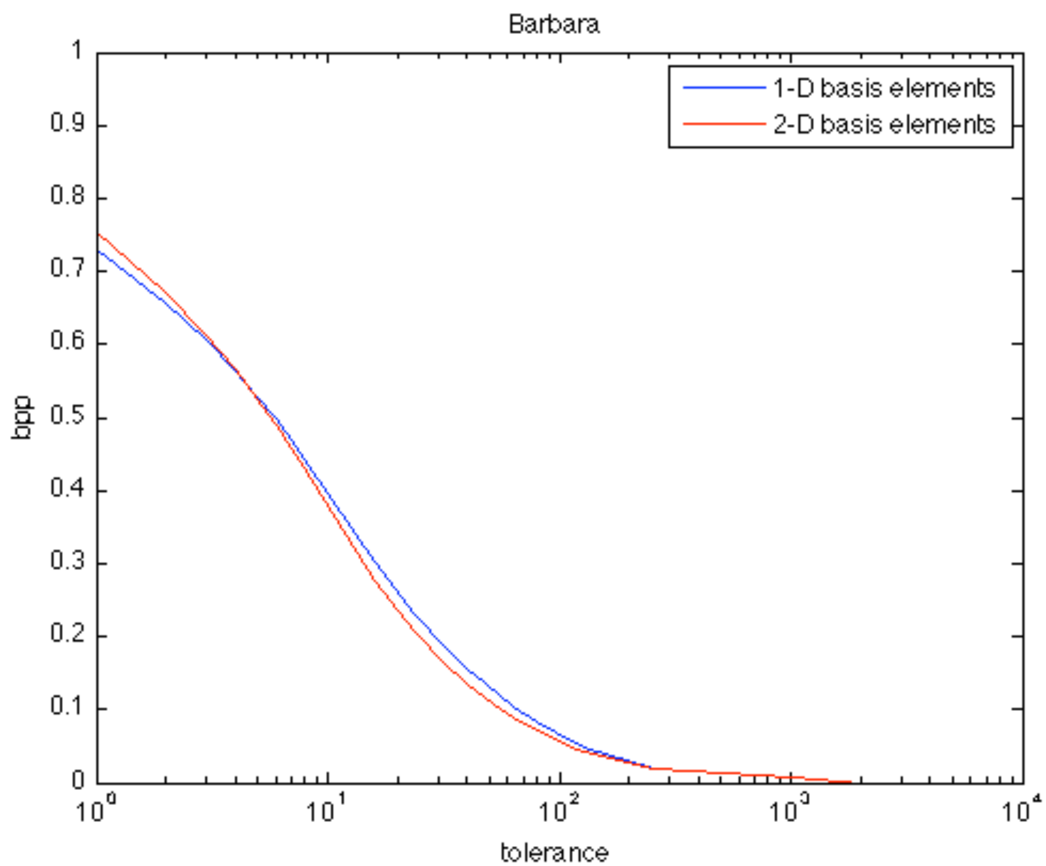
Visual overview: Barbara

$\varepsilon = 32$, $c = 4$
PSNR = 36.9952 dB
MSSIM = 0.9447
n-bit-rate = 0.1863 bpp
Termination: $||.||_2$

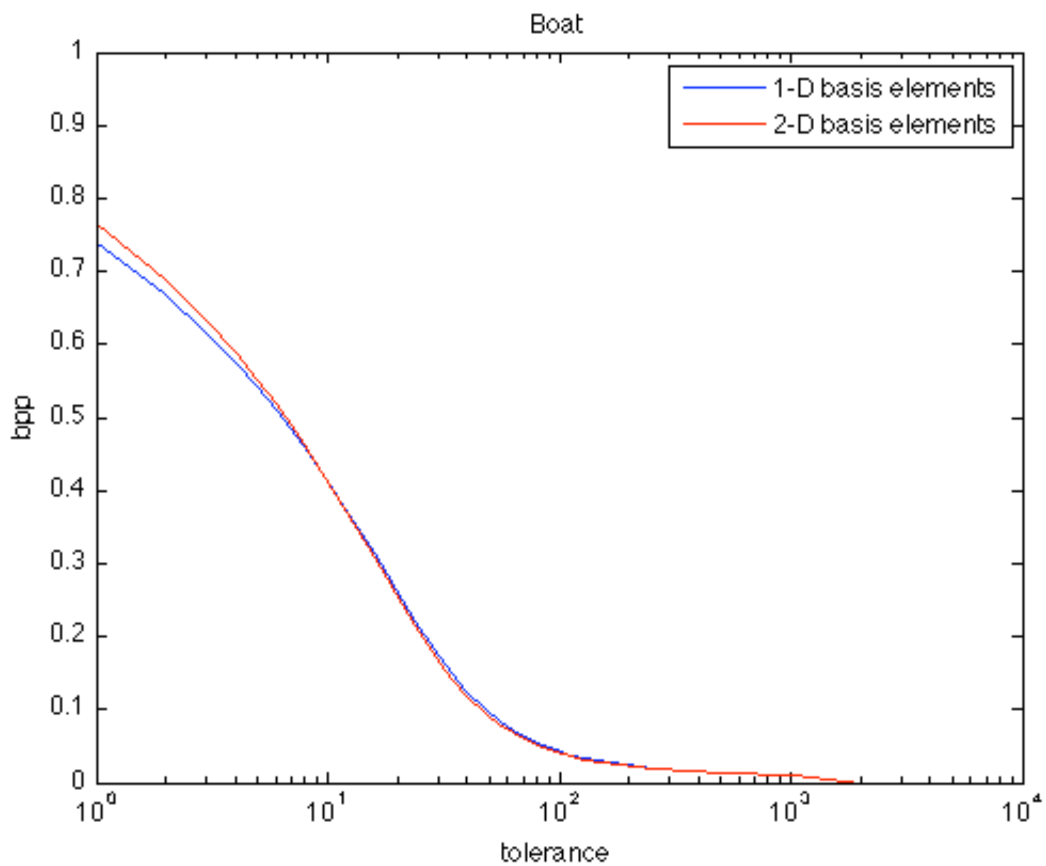
Visual overview: Barbara

$\delta = 0.94$
PSNR = 32.1482 dB
MSSIM = 0.9466
n-bit-rate = 0.1539 bpp
Termination: $||.||_{ssim}$

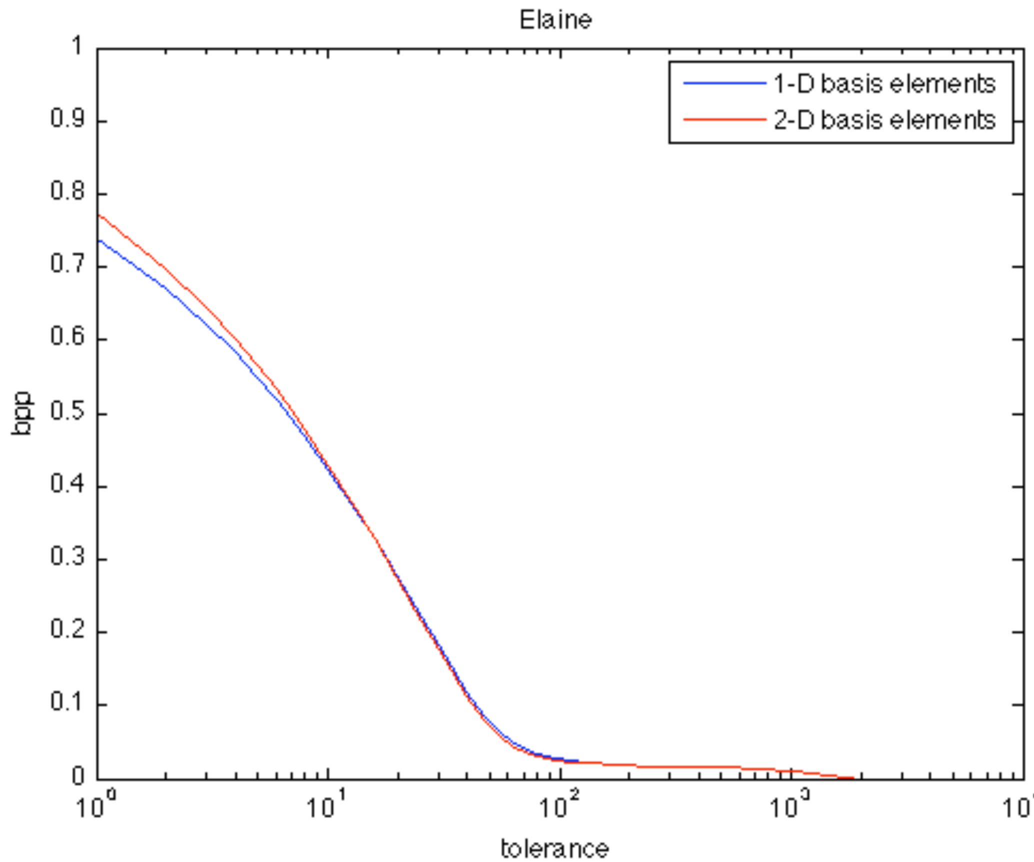
1D vs 2D basis comparison



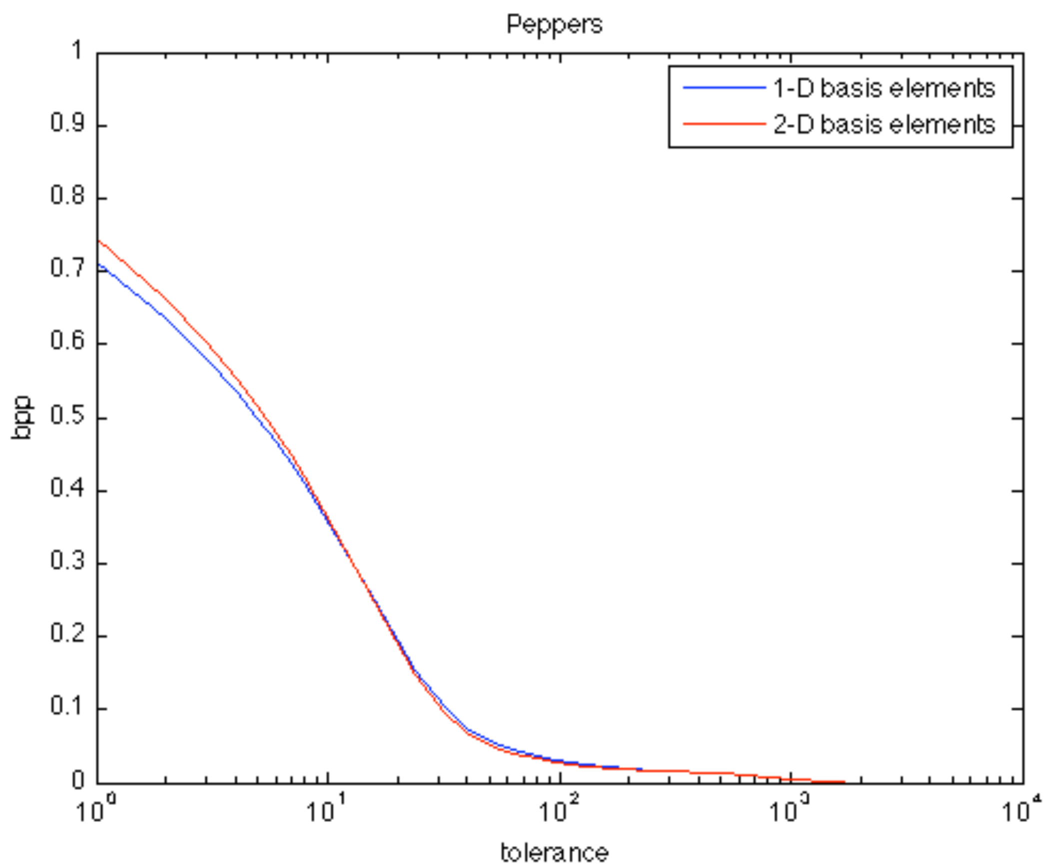
1D vs 2D basis comparison



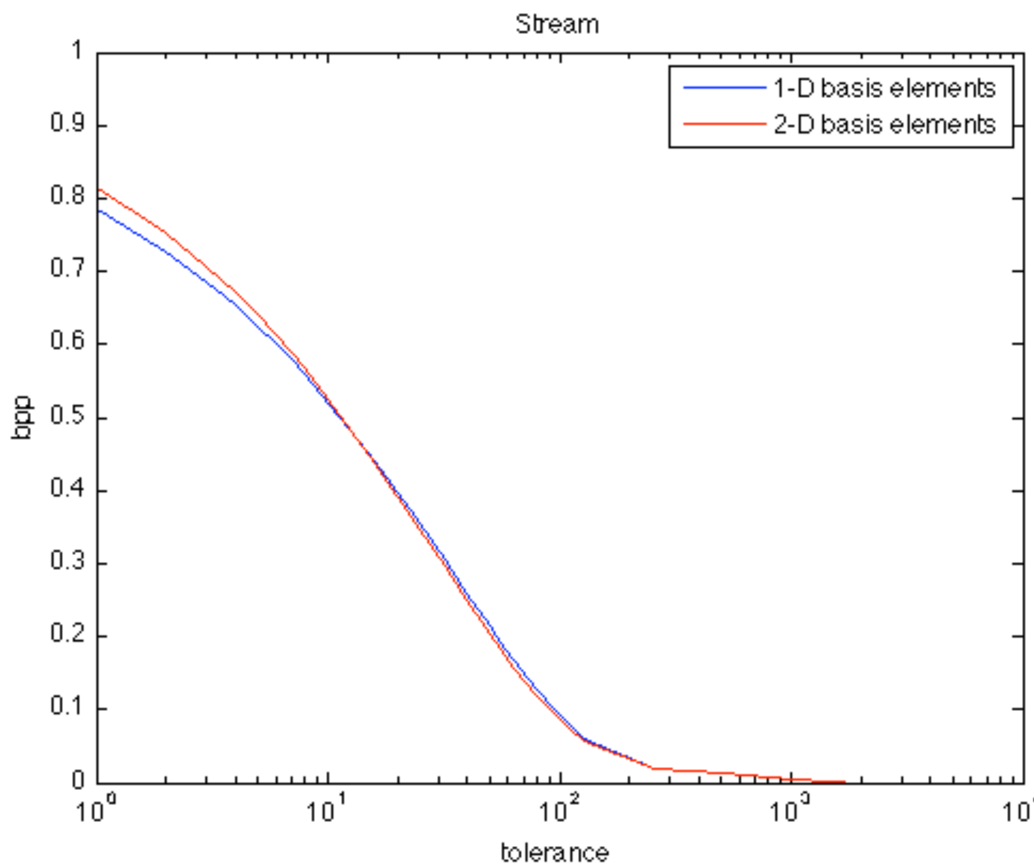
1D vs 2D basis comparison



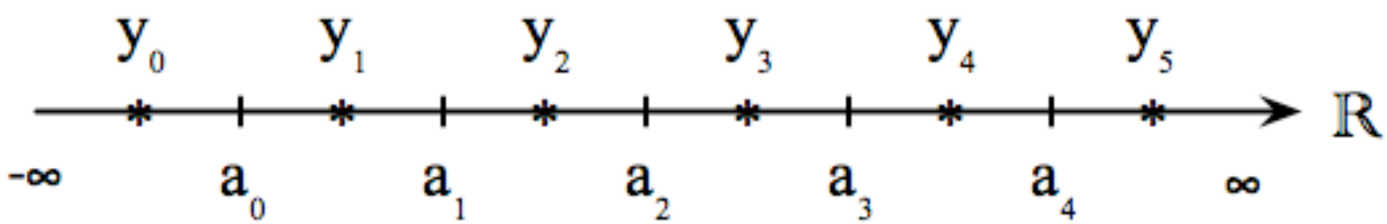
1D vs 2D basis comparison



1D vs 2D basis comparison



Quantization



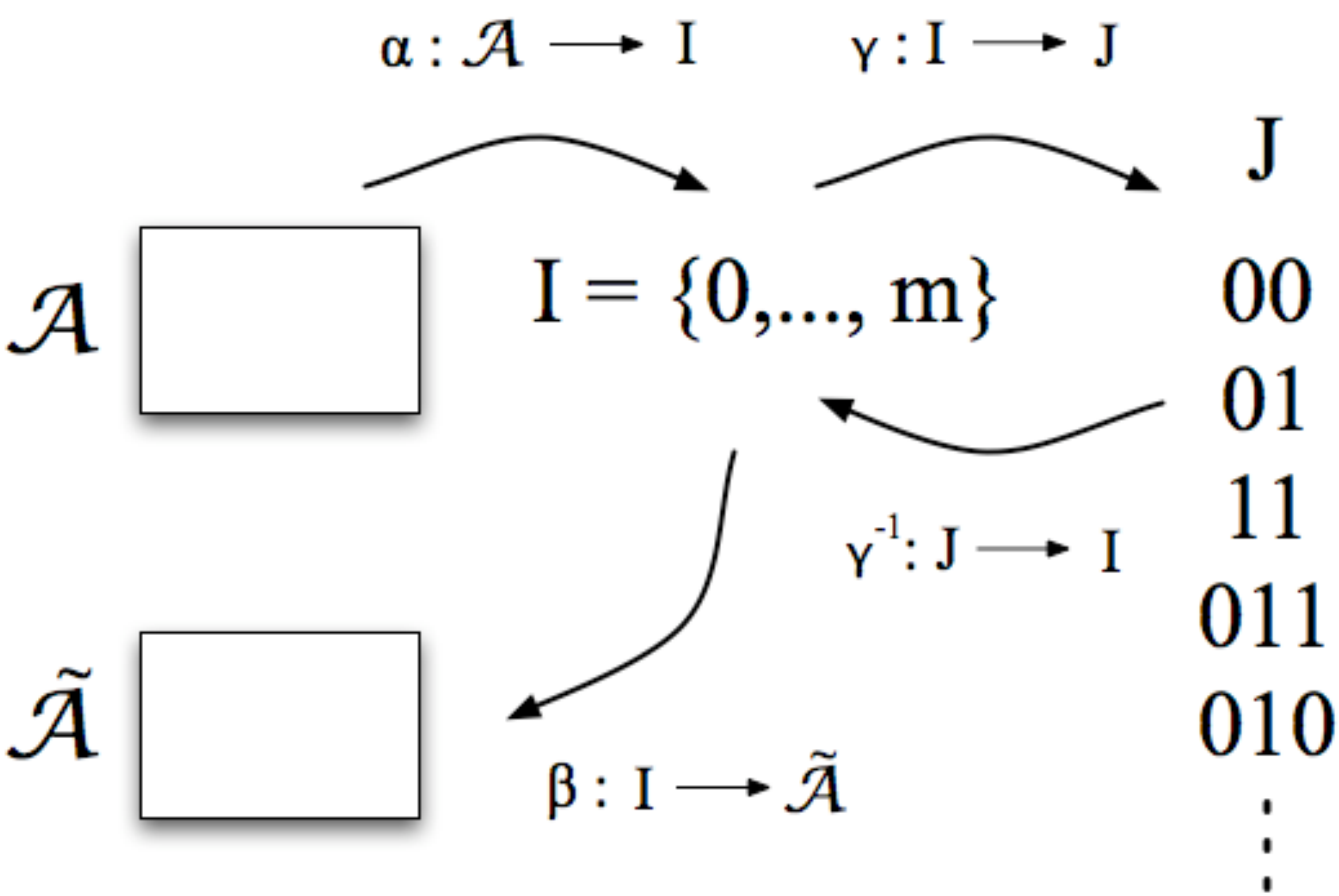
$$S_0 = (-\infty, a_0] \quad S_i = (a_{i-1}, a_i] \quad S_4 = (a_4, \infty)$$

$$i = 1, 2, 3$$

$$\mathcal{S} = \{S_i \subset \mathbb{R} : i \in I\} \qquad \mathcal{C} = \{y_i \in \mathbb{R} : i \in I\}$$

$$q(x) = \sum_{i \in I} y_i \mathbb{1}_{S_i}(x)$$

Quantization



Quantization

Instantaneous rate

$$r(\mathbf{x}) = \tfrac{1}{k} l(\gamma(\alpha(\mathbf{x})))$$

Distortion measure

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$$

$$\begin{aligned} R(\alpha, \gamma) = E[r(X)] &= \frac{1}{k} E[l(\gamma(\alpha(X)))] \\ &= \frac{1}{k} \sum_i l(\gamma(i)) \int_{S_i} f(\mathbf{x})\, d\mathbf{x} \end{aligned}$$

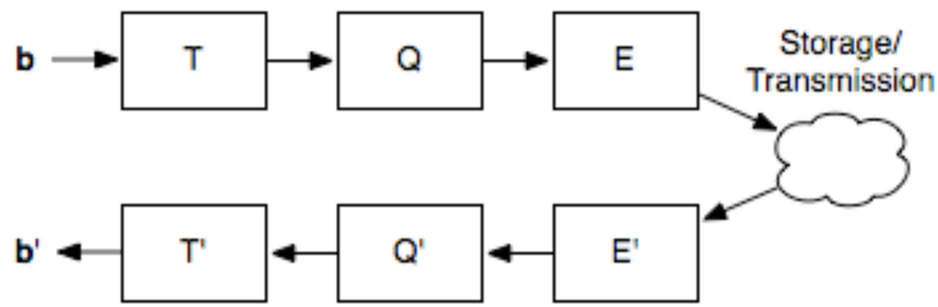
Normalized average rate

$$\begin{aligned} D(\alpha, \beta) &= \frac{1}{k} E[d(X, \beta(\alpha(X)))] \\ &= \frac{1}{k} \sum_i \int_{S_i} d(\mathbf{x}, \mathbf{y}_i) f(\mathbf{x})\, d\mathbf{x} \end{aligned}$$

Normalized average distortion

Quantization

Transform encoding



$\mathbf{x}_0 = T\mathbf{b} = \text{OMP}(a\mathbf{A}, \mathbf{b}, \varepsilon_0)$ $\alpha = QT$ $d(\beta(\alpha(\mathbf{b})),\mathbf{b}) = ?$

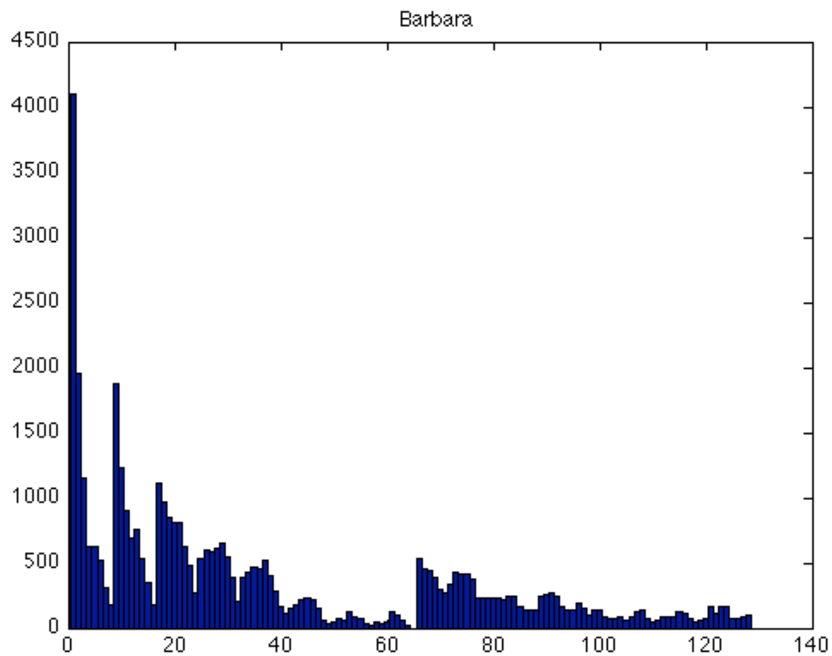
$T' = a\mathbf{A}$ $\beta = T'Q'$

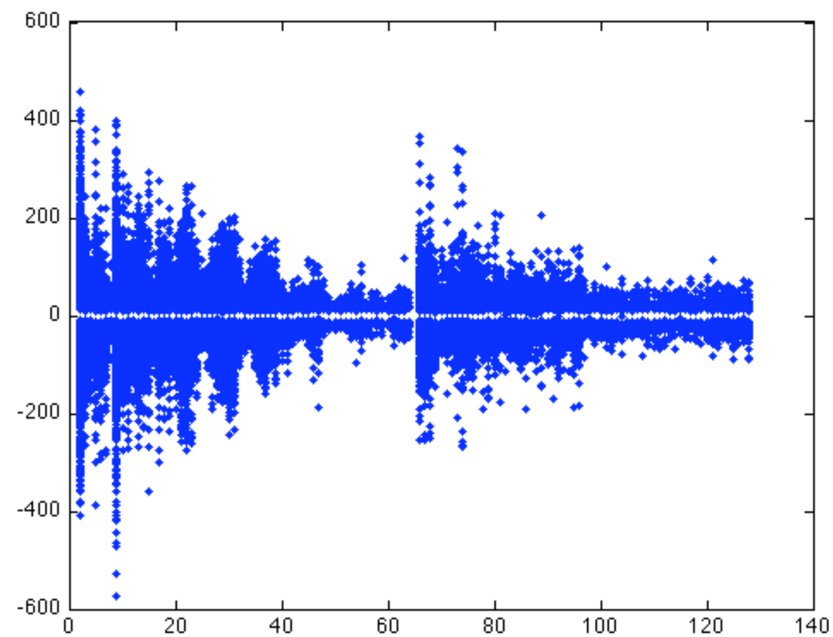
$\underline{\mathbf{b}} = a\mathbf{A}\, \underline{\mathbf{x}}_0 = T'\, Q'Q\, \mathbf{x}_0$

Quantization

$$
\begin{aligned}
d(\beta(\alpha(\mathbf{b})), \mathbf{b}) &= \|T'Q'QT\mathbf{b} - \mathbf{b}\|_2 \\
&= \|T'Q'QT\mathbf{b} - T'T\mathbf{b} + T'T\mathbf{b} - \mathbf{b}\|_2 \\
&= \|a\mathbf{A}\tilde{\mathbf{x}}_0 - a\mathbf{A}\mathbf{x}_0 + a\mathbf{A}\mathbf{x}_0 - \mathbf{b}\|_2 \\
&\leq \|a\mathbf{A}\tilde{\mathbf{x}}_0 - a\mathbf{A}\mathbf{x}_0\|_2 + \|a\mathbf{A}\mathbf{x}_0 - \mathbf{b}\|_2 \\
&= ac\|\delta\|_\infty \|\mathbf{x}_0\|_0 + \epsilon_0, \quad \delta = \tilde{\mathbf{x}}_0 - \mathbf{x}_0
\end{aligned}
$$

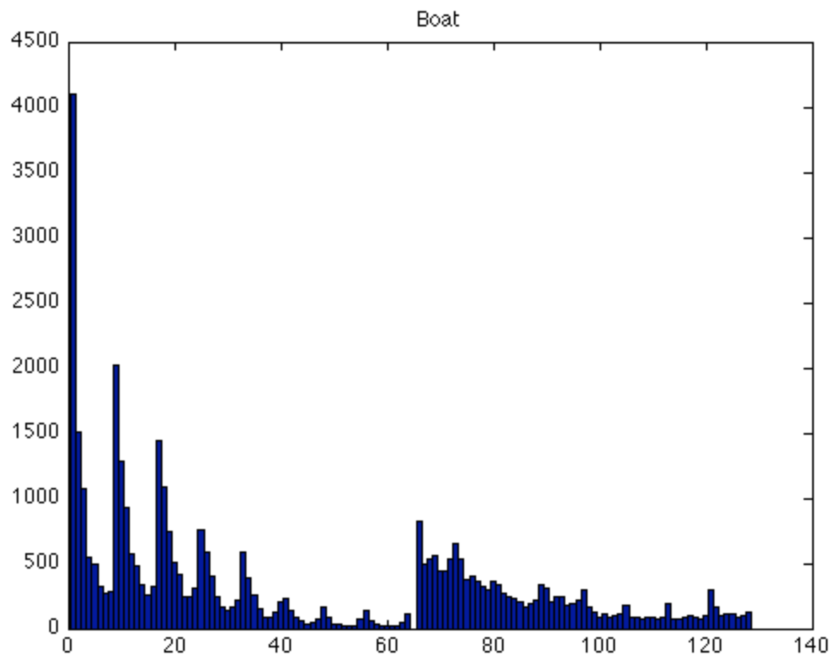
Histogram and coefficient distribution

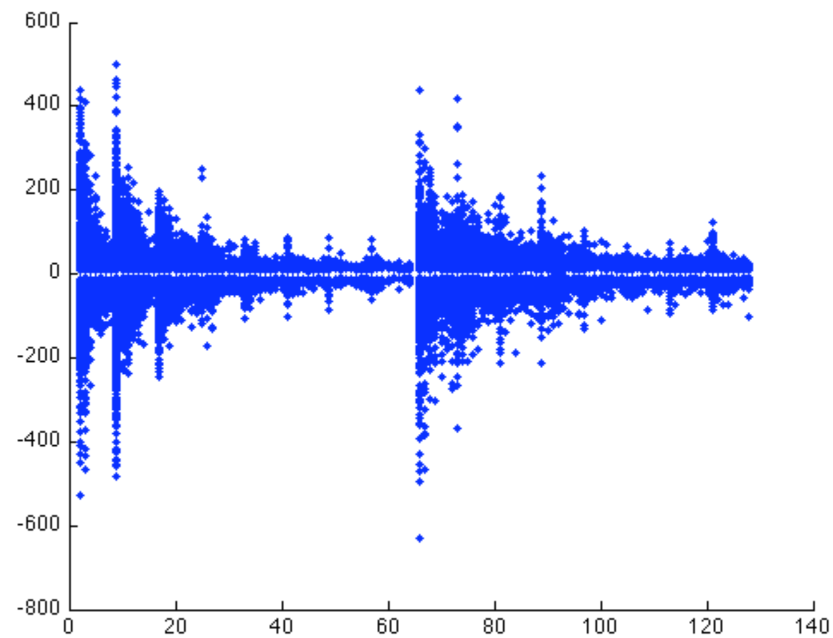




2D basis elements with $c = 1/8$

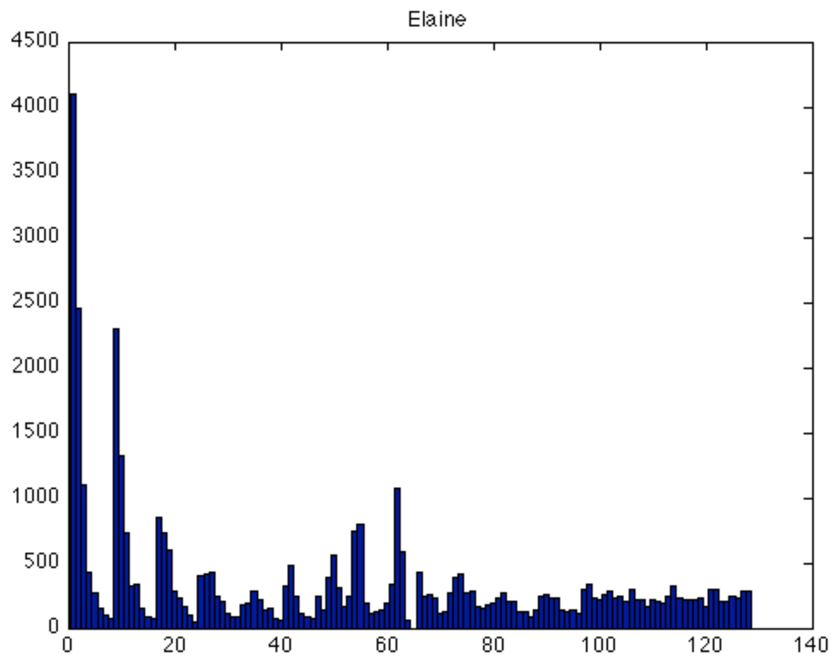
Histogram and coefficient distribution

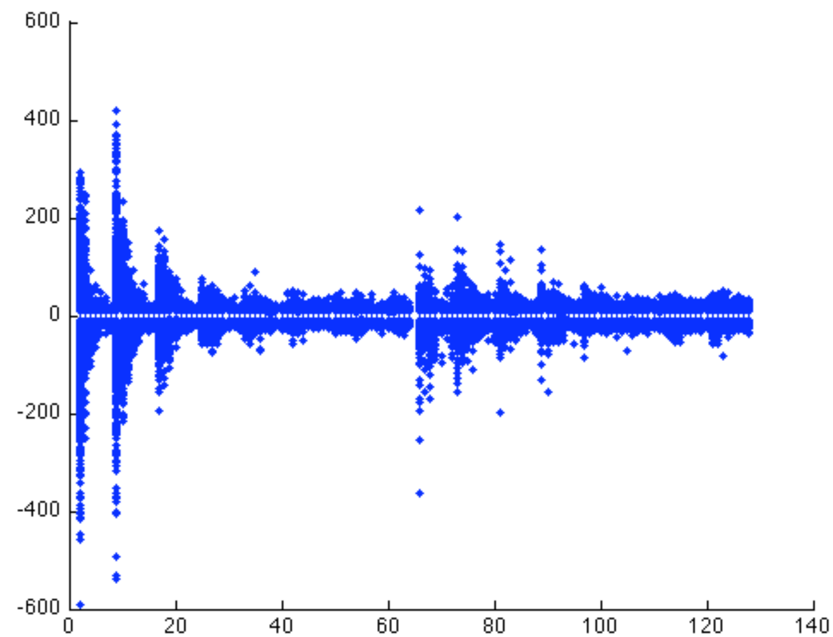




2D basis elements with *c* = 1/8

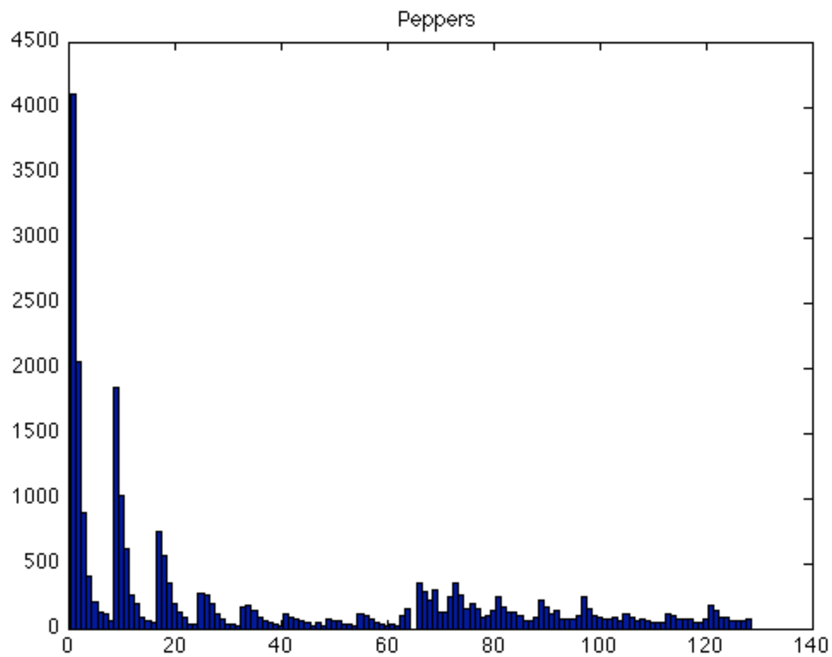
Histogram and coefficient distribution

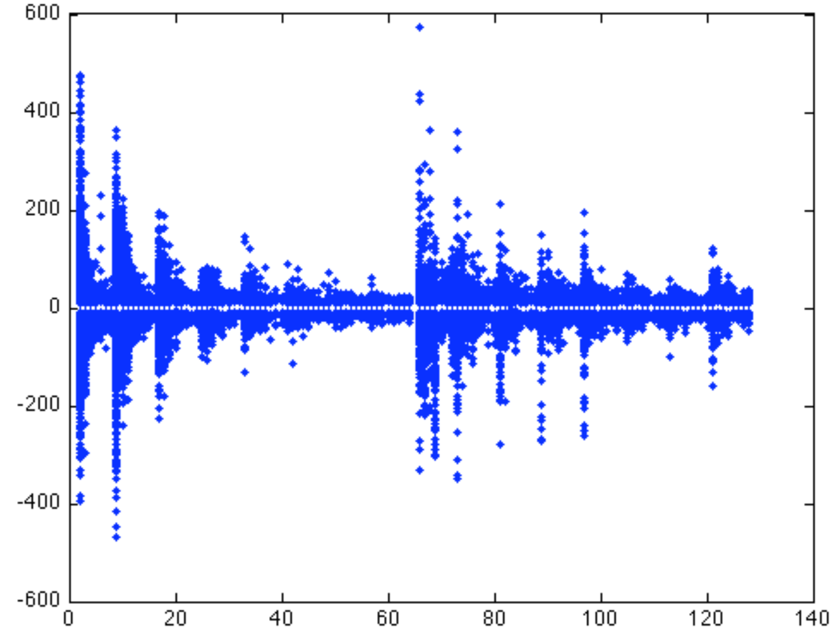




2D basis elements with *c* = 1/8

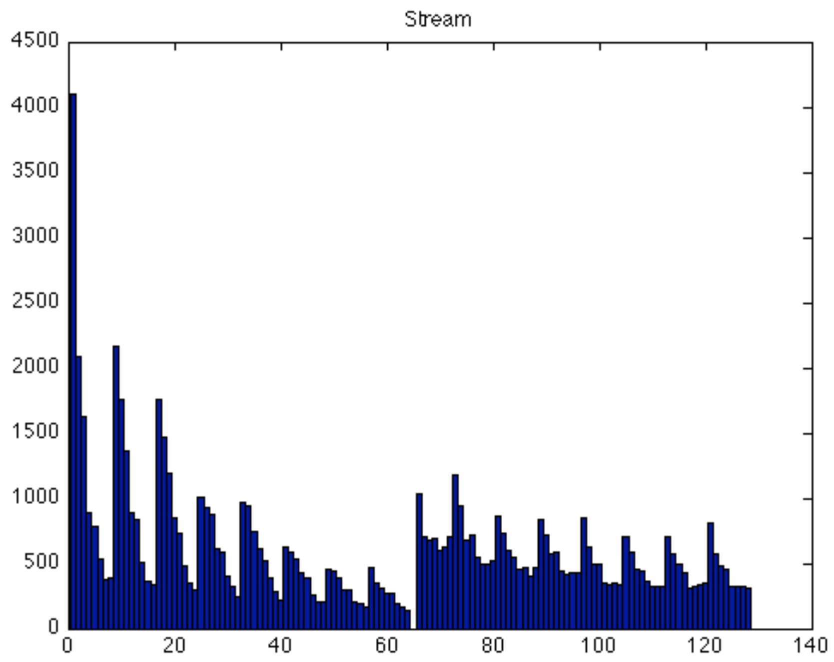
Histogram and coefficient distribution

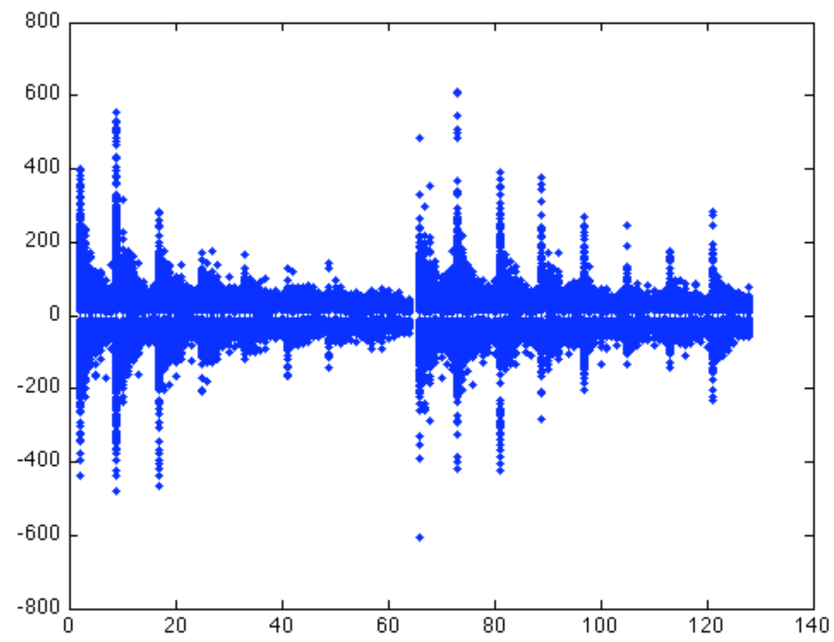




2D basis elements with c = 1/8

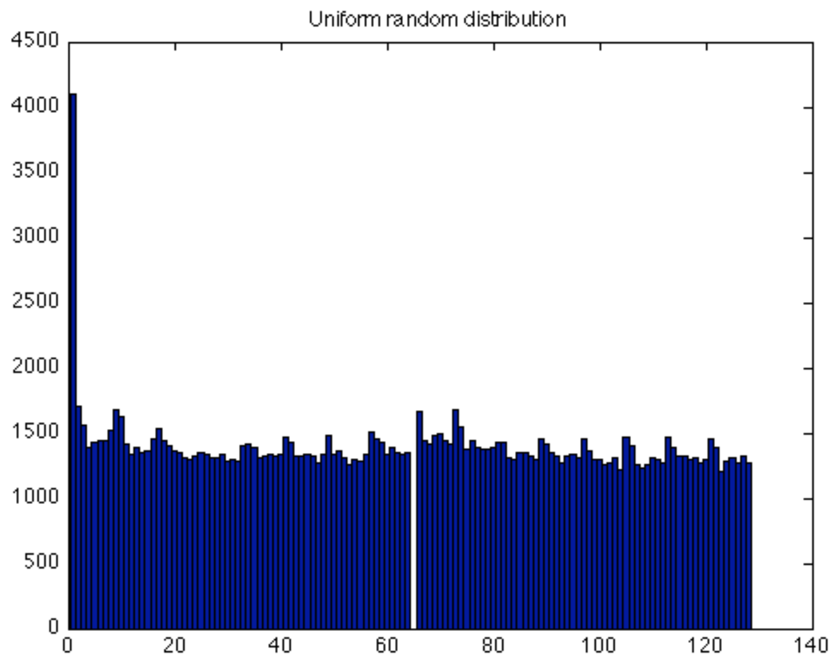
Histogram and coefficient distribution

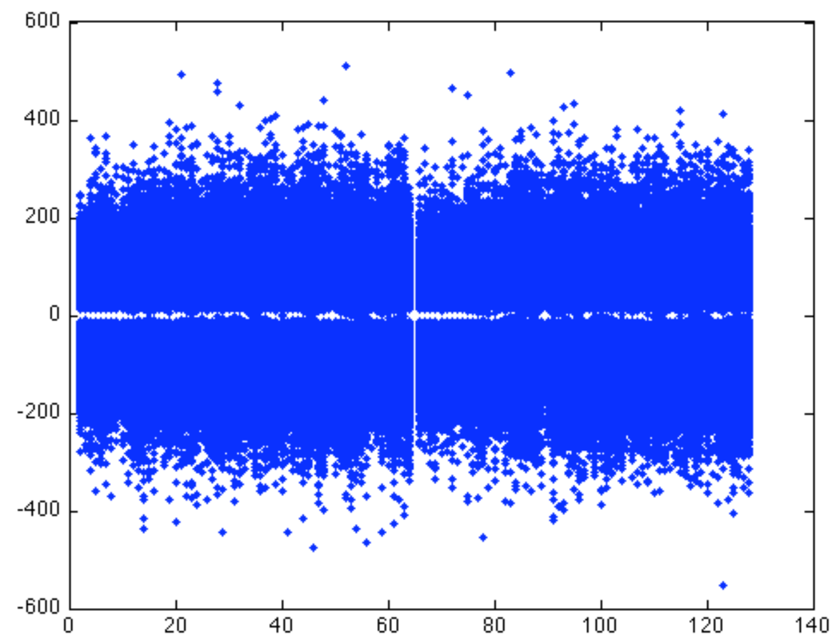




2D basis elements with $c = 1/8$

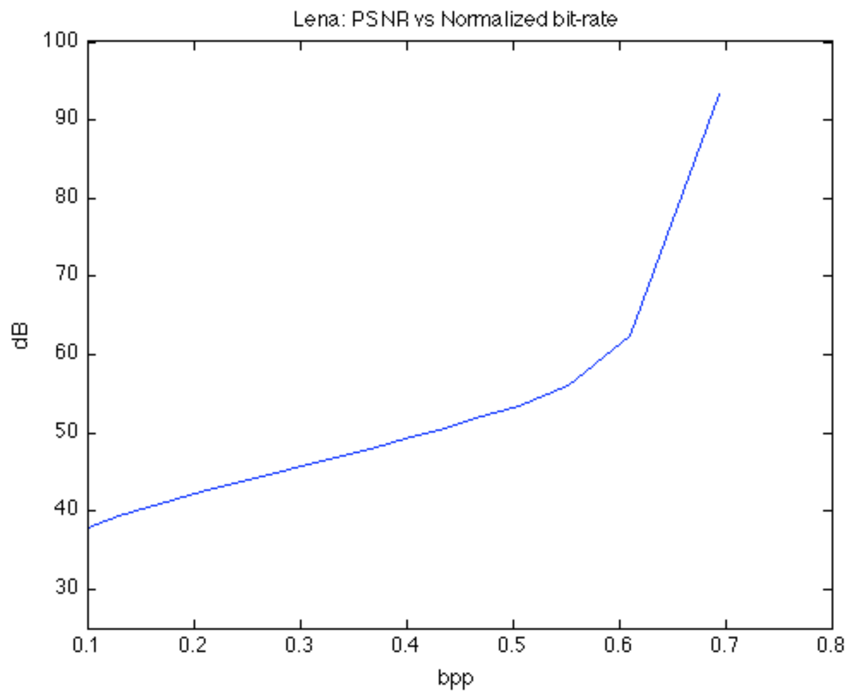
Histogram and coefficient distribution

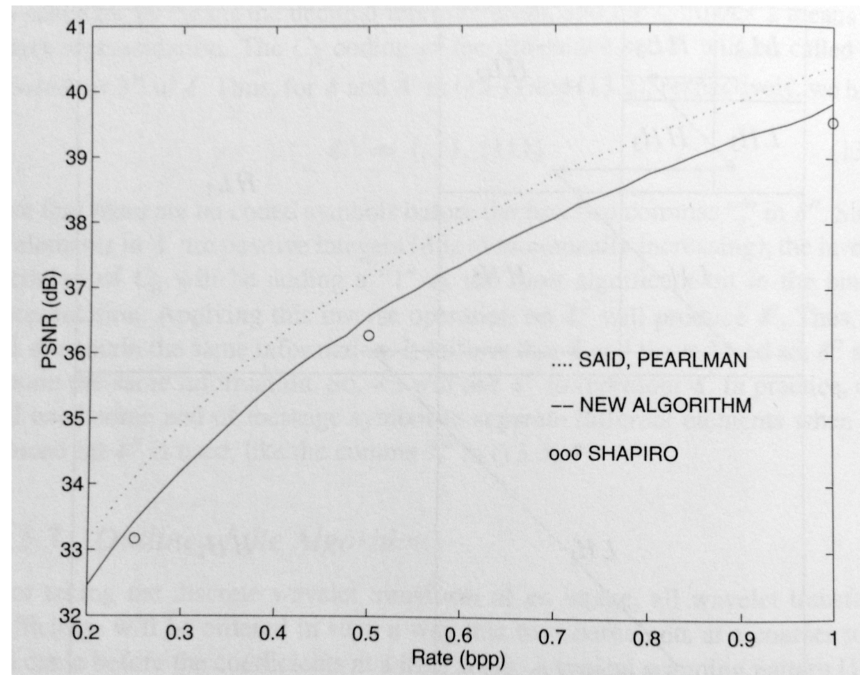




2D basis elements with *c* = 1/8

PSNR vs bit-rate





Comparison between our work and published results

Future Work

- Fast algorithm for DCT+Haar?

- γ functions for quantizer in the DCT+Haar setting

- Complete theory for Gabor systems and other matrices

Thank you!

Some References

[1] A. M. Bruckstein, D. L. Donoho, and M. Elad, *From sparse solutions of systems of equations to sparse modeling of signals and images*, SIAM Review, 51 (2009), pp. 34–81.

[2] B. K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM Journal on Computing, 24 (1995), pp. 227-234.

[3] G. W. Stewart, **Introduction to Matrix Computations**, Academic Press, 1973.

[4] T. Strohmer and R. W. Heath, *Grassmanian frames with applications to coding and communication*, Appl. Comput. Harmon. Anal., 14 (2004), pp. 257-275.

[5] D. S. Taubman and M. W. Mercellin, **JPEG 2000: Image Compression Fundamentals, Standards and Practice**, Kluwer Academic Publishers, 2001.

[6] G. K. Wallace, *The JPEG still picture compression standard*, Communications of the ACM, 34 (1991), pp. 30-44.

[7] S. Mallat, **A Wavelet Tour of Signal Processing**, Academic Press, 1998.

[8] Z. Wang, A.C. Bovik, H.R. Sheikh and E.P. Simoncelli, *Image quality assessment: from error visibility to structural similarity*, IEEE Transactions on Image Processing , vol.13, no.4 pp. 600- 612, April 2004.
https://ece.uwaterloo.ca/~z70wang/research/ssim/index.html