Chapter 3

Image compressive sensing

One of the more intuitive applications of CS lies in spatial signals as it is easier to visualize. In this scheme, the process can be simplified either by flattening it to one dimension and processing it in its entirety, or maintaining its dimensionality and processing it by patches. The general workflow that arises from image CS is as follows:

1. Define the compression ratio m/n, where n is the signal size, and m is the desired size of the compressed signal.

2. Draw m random indices from the signal without replacement and store this as a sample sequence $\boldsymbol{\xi}$.

3. Extract the row vectors of the desired $n \times n$ sparsifying basis $\boldsymbol{\Psi}$ indexed by $\boldsymbol{\xi}$, and stack these to form the sensing matrix $\boldsymbol{\Phi}$ (i.e., $\boldsymbol{\Phi} = \boldsymbol{\Psi}_{\boldsymbol{\xi}}$)

4. With the desired reconstruction algorithm, perform the optimization (2.8) to obtain the reconstructed signal $\hat{\mathbf{x}}$.

In the case of high-definition images (whose shortest side is at least 720 pixels), it is usually more practical and yields better results if the image is processed in patches.

3.1 Test case: Sinusoidal pattern

As mentioned in Chapter 2, the most commonly used sparse representation domain for images is the Fourier domain, referred to in some fields as k-space. In this space, signals are represented as a linear superposition of a finite number of sinusoidal patterns. In Fig. 3.1a, 64×64 pixel sinusoidal patterns are generated, corresponding to sine waves traveling horizontally, vertically, and diagonally, as well as an egg tray pattern. In each case, all frequency components are 4 Hz. Figure 3.1b visualizes the compressed image when a random sample of 5% is taken from the signal. The actual compressed signal that is seen by the reconstruction algorithm is a one-dimensional sequence containing only the information from the points being sampled. Orthogonal matching pursuit (OMP) was used for reconstruction, which is a greedy algorithm that finds the combination of basis vectors which best represents the signal (similar to matching pursuit), but in addition, the residual at each iteration is recomputed using an orthogonal projection on the set of previously selected basis vectors [32]. Its objective function is

$$\arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{\Phi x}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_0 \leq \gamma \tag{3.1}$$

where γ is a hyperparameter which controls the maximum allowable number of non-zero coefficients. The Scikit-learn implementation sets this value to 10% of the number of samples by default [30]. Evaluation of the mean-squared error

(MSE) for the pure horizontal and pure vertical sine waves, as well as the egg tray pattern yields a value that is practically negligible ($\approx 10^{-31}$); the reconstruction is exact. On the other hand, the reconstructed diagonal sine wave yields an MSE of 10^{-3}—still quite small, but mild distortion can be observed at the image boundaries. This is due to the fact that the information at hand is finite, and so is the window size which, in this case, is the same size as the signal itself.

3.2 Image with multiple sinusoids

For this section, the image used is M.C. Escher's *Relativity*, an example of a more complex image but consisting of dominant sinusoidal patterns that are made apparent when you zoom in. The original image has dimensions of 1600×981 pixels, for a total of 1,569,600 pixels. Following the procedure with the previous section, this would require the construction of a $1{,}569{,}600 \times 1{,}569{,}600$ sparsifying matrix containing $\approx 2 \times 10^{12}$ entries. Assuming that the matrix would be stored as 32-bit floating point numbers, this process alone would take up ≈ 8 GB of memory, and it would be highly impractical to process similarly-sized images as a whole. The workaround is to split it into smaller, manageable patches. For this image in particular, it was first resized to 1600×976 pixels so that it could be equally divided into a grid of 16×16, each with a dimension of 100×61 pixels. After compressively sampling each patch at 40% compression ratio, reconstruction was performed using the Embedded Conic Solver (ECOS) of the Convex Optimization Python library (CVXPY), which recasts (2.8) as a convex problem and directly minimizes the ℓ_1 norm [28, 29, 33] and thus, is significantly slower compared to OMP. After stitching all patches at the end, the reconstructed image is shown

in Fig. 3.3. Selected patches with the aforementioned dominant patterns are shown with their reconstructed counterparts in Fig. 3.4, corresponding to patches dominated by horizontal sinusoids, vertical sinusoids, diagonal sinusoids, multiple sinusoids, and patches with no dominant pattern. We can observe that at this compression rate, the patches with a single apparent sinusoidal pattern (Figs. 3.4a-3.4c) are successfully recovered, with some noise present especially for the patch with a dominant diagonal pattern (similar to the previous section). The patch with multiple sinusoid patterns (Fig. 3.4d), although still recognizable, is laden with a lot of noise. Lastly, the patch with no apparent pattern (Fig. 3.4e) is barely recognizable, except for the portions where a dominant sinusoidal pattern is partially present in the frame.

From this, the following information can be gleaned. First, reconstruction performs better on smaller patches, and when the patch in question contains as few frequency components as possible (such is the case with the patches with only one dominant pattern). Second, the patch with no dominant pattern—upon closer visual inspection—can be classified as being successfully recovered; however, the reconstruction noise is almost at the same level as the signal itself, which makes them indistinguishable. This can be attributed to the fact that the patches with no apparent dominant pattern are actually composed of a superposition of sinusoids residing primarily in the high-frequency region of k-space. Since the sampling points are uniformly distributed throughout the spatial domain, so are they in the frequency domain. Thus, the information in the high-frequency region is not sufficiently captured, and a higher compression ratio is required to be able to better recover these high-frequency regions. Another solution would be, as mentioned

earlier, to make the patches smaller so that lesser frequencies are captured in one patch.

3.3 Simultaneous compression & encryption

Because of the way compressively sensed images are coded with the sensing matrix, the use of CS as an encryption algorithm arises naturally. Consider the logistic map

$$x_{n+1} = rx_n(1 - x_n) \tag{3.2}$$

which is often used as an archetypal example of deterministic chaotic behavior for values of $r \in [3.57, 4]$. In this regime, the sequences produced by varying the initial parameter x_0 rapidly diverge from each other. Thus, this can become an encryption system by treating the parameters r and x_0 as an encryption key pair, and (3.2) as the hash function.

In application for images, four keys are required: one key pair for each dimension. The construction of the sensing matrix $\mathbf{\Phi}$ also differs from the general workflow, and is as follows:

1. From (3.2), generate a sequence of length $2m$ with the initial key pair r_1 and x_{01}. Discard the first n elements to avoid the transient response and store the latter n elements as a sequence $\mathbf{s}$.

2. Explicitly generate the index sequence of $\mathbf{s}$ and store it as the index sequence $\mathbf{p} = [0, 1, ..., m - 1]$.

3. Sort $\mathbf{p}$ according to ascending values of $\mathbf{s}$.

4. Generate the first sensing matrix $\mathbf{\Phi}_1$ by extracting and stacking rows of a Hadamard matrix of order N indexed by the first m elements of $\mathbf{p}$, i.e.,

$$\mathbf{\Phi}_1 = \begin{bmatrix} \mathbf{H}_{p_1} & \mathbf{H}_{p_2} & \cdots & \mathbf{H}_{p_m} \end{bmatrix}^\top \tag{3.3}$$

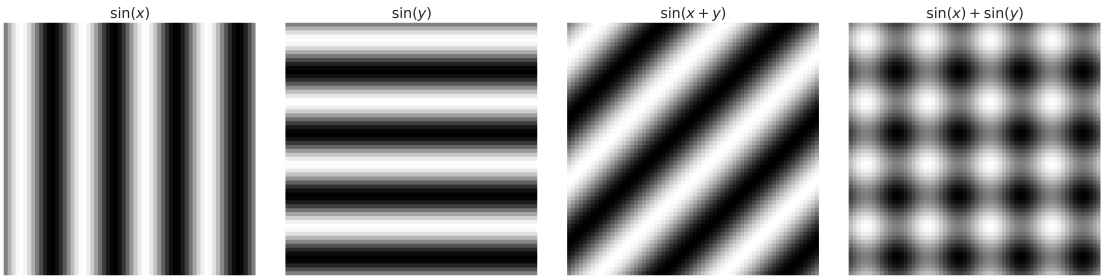
   where $\mathbf{H}_{p_i}$ denotes the p_ith row vector of $\mathbf{H}$.

5. The second sensing matrix can be constructed using a different key pair r_2 and x_{02}.

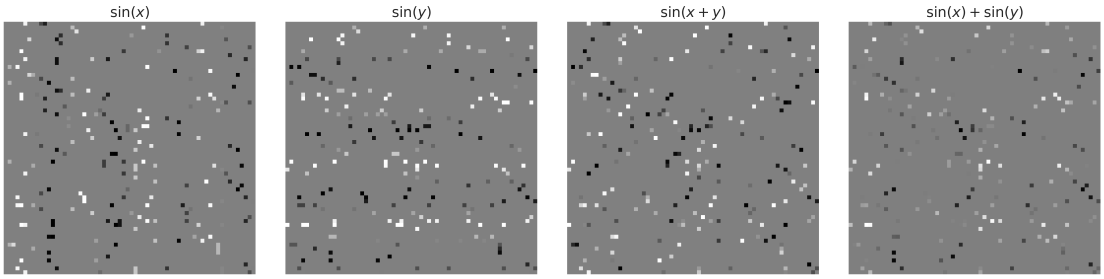
The above steps imply that the image must first be reshaped to have dimensions that are integer multiples of 4. The original image is first reshaped to 256×256 pixels, and is sparsified by transforming it to the discrete cosine transform (DCT) domain. The desired compressed dimension is set to $m = 192$, corresponding to a compression ratio $m/n = 75\%$, and the keys are set to values of $r_1 = r_2 = 3.99$, $x_{01} = 0.11$, and $x_{02} = 0.24$. Figure 3.5 shows the application of this to the Lena test image (left), its encrypted representation (middle), and the decrypted/reconstructed image (right). Visual inspection of the encrypted representation shows horizontal and vertical bands distributed throughout the representation space, and is indicative that a simple inverse Fourier transform will not recover any meaningful information. Assuming that the receiver knows the encryption scheme, recovery of the original message is successful if the same keys r_1, r_2, x_{01}, x_{02} as the encryption stage are used, which will allow the receiver to construct the exact same sensing matrices $\mathbf{\Phi}_1, \mathbf{\Phi}_2$ and perform the inverse operation on the encrypted message. In the decrypted image, encryption artifacts can be

observed, as indicated by some visible banding, but is nonetheless recognizable; evaluation of the MSE yields a value of 0.02.

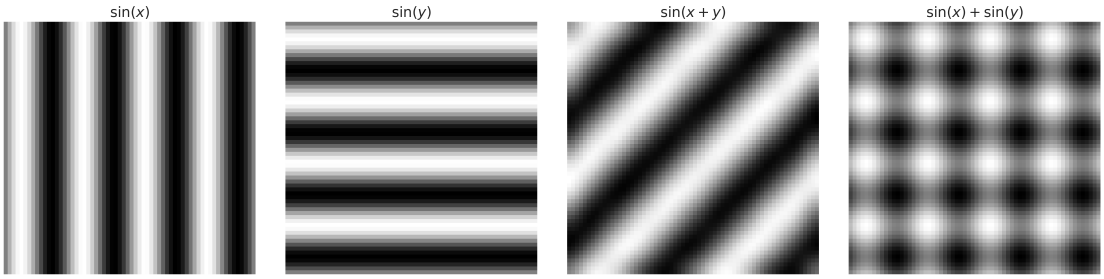
With the knowledge that the hash function (3.2) is chaotic, the encryption strength of the system can be tested by slightly perturbing the initial values. Figure 3.6 shows the decryption results when all the correct keys are used, except for x_{01}, which is perturbed by a tiny value $\approx 10^{-15}$ (third image), and similarly when the correct x_{01} is used but x_{02} is perturbed by the same amount (last image). Additionally, Fig. 3.7 shows the MSE curves for differing values of the perturbation Δx_{01} and Δx_{02}. The MSE generally oscillates at some high value for perturbations on the degree of 10^{-14}, and exhibits a sharp dip when the MSE is evaluated for the correct keys ($\Delta x_0 = 0$). This shows that brute force attacks are intractable against this kind of encryption system.



(a) Test 2D sinusoid patterns.



(b) Visualization of sinusoid patterns with compression ratio of 5%.



(c) Reconstruction from 5% of samples.

Figure 3.1: Test 64×64 pixel 2D sinusoid patterns corresponding to vertical sinusoids, horizontal sinusoids, diagonal sinusoids, and egg tray pattern. All frequency components are 4 Hz.

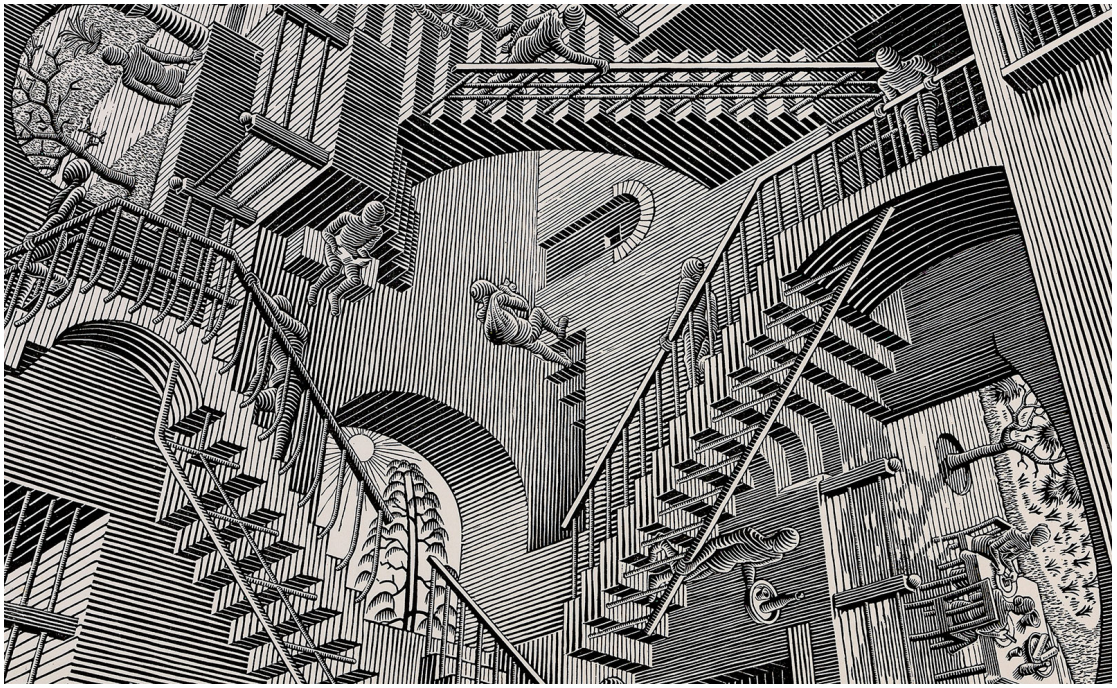


Figure 3.2: *Relativity* by M.C. Escher, a complex image consisting of various sinusoidal patterns.

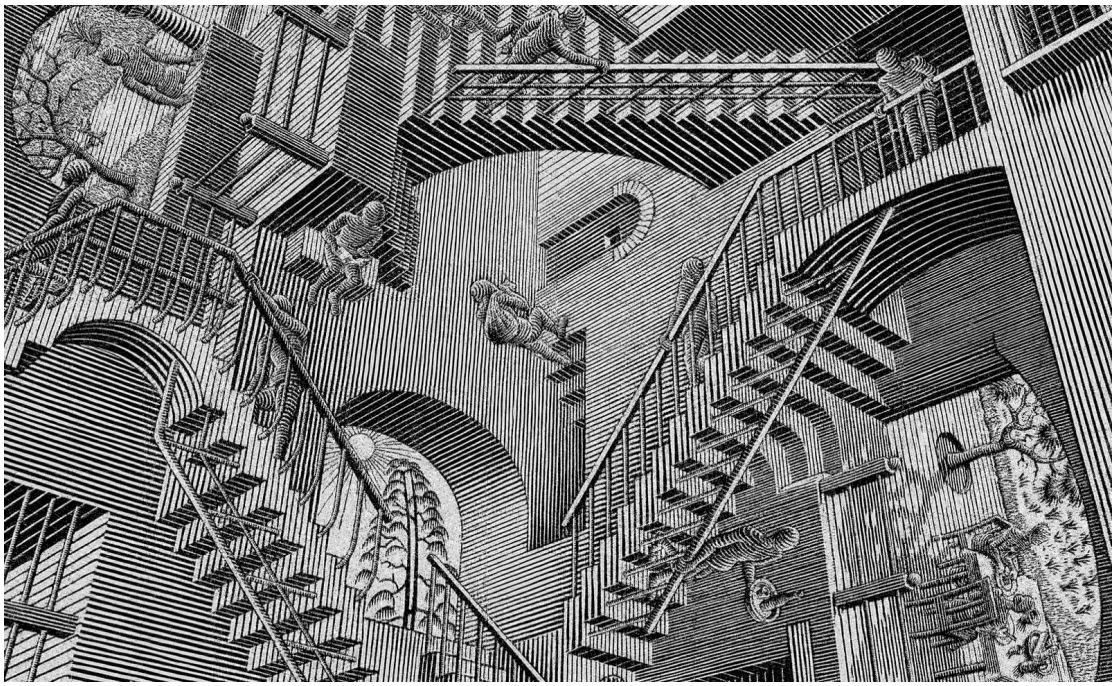
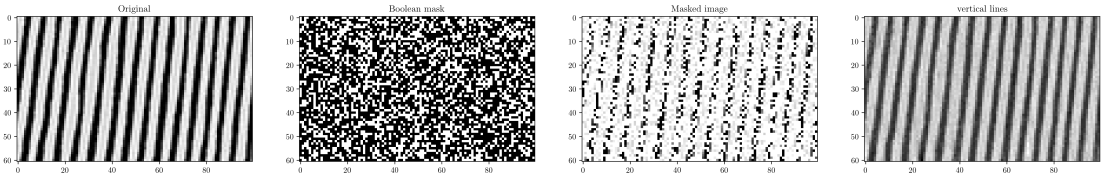
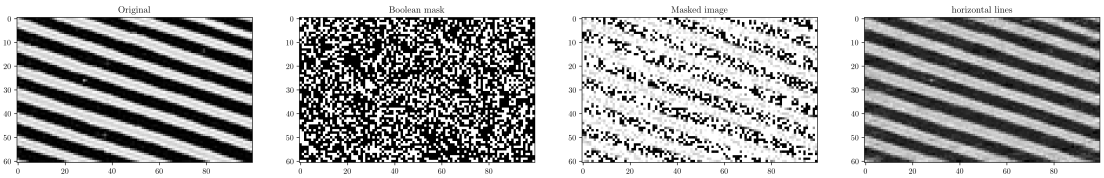


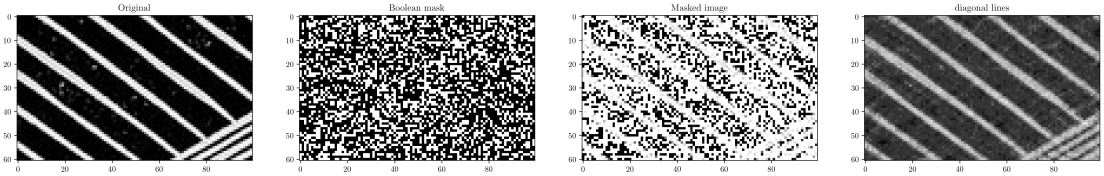
Figure 3.3: Reconstructed *Relativity* from 50% of samples from each patch.



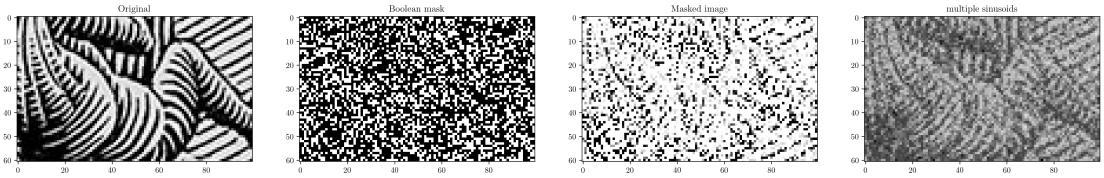
(a) Dominant vertical



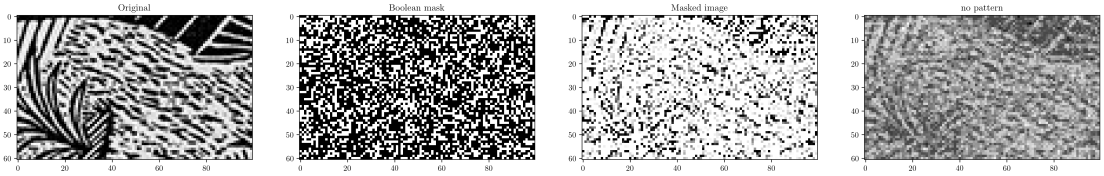
(b) Dominant horizontal



(c) Dominant diagonal



(d) Multiple dominant sinusoids



(e) No dominant pattern

Figure 3.4: Extracted and reconstructed patches from *Relativity* using 40% of samples.

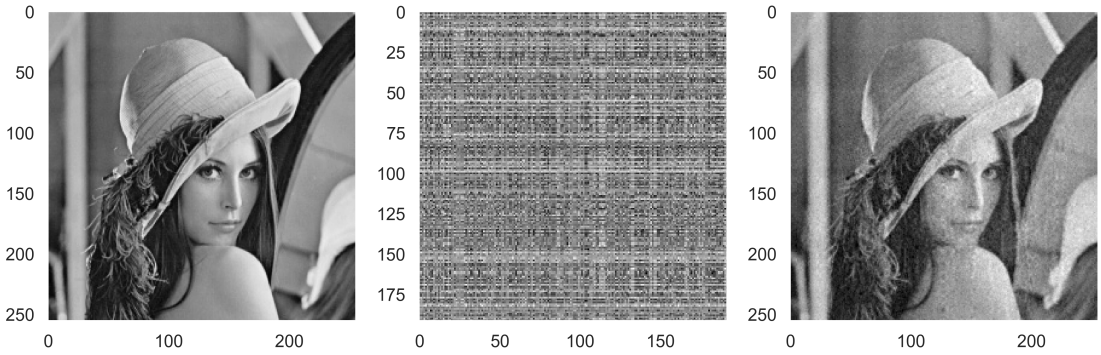


Figure 3.5: Simultaneous compression and encryption achieved with compressive sensing: original image (left), encrypted image (middle), and decrypted/reconstructed image (right).

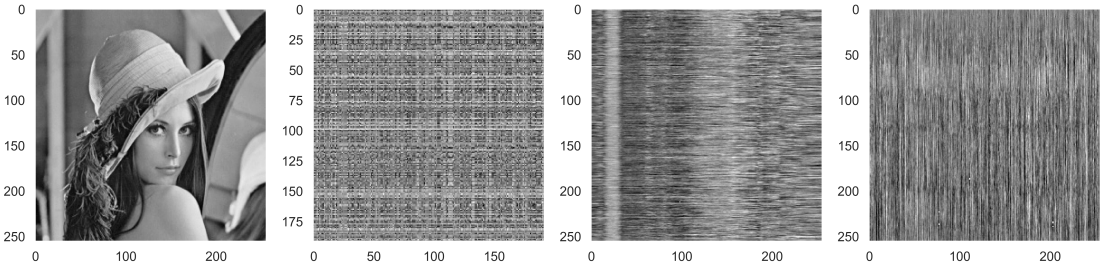
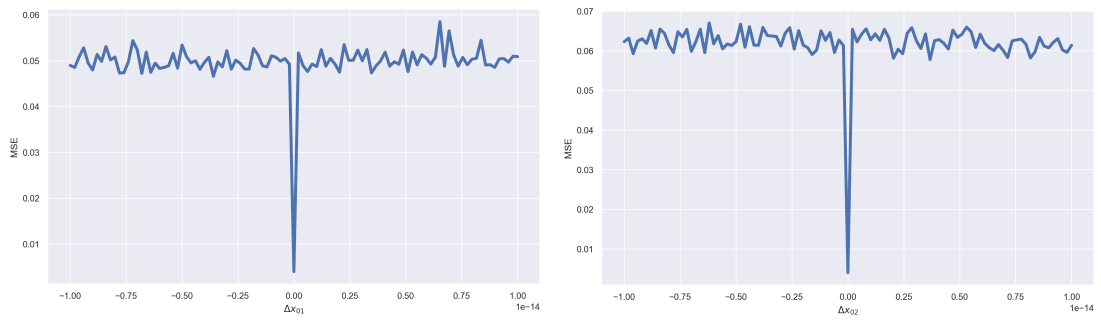


Figure 3.6: Test image Lena (first) with the encrypted representation (second), the decryption result when the correct keys are used but x_{01} is perturbed by a value of 10^{-15} (third), and the decryption result when the correct keys are used but x_{02} is perturbed by a value of 10^{-15}.



(a) Perturbation of Δx_{01}

(b) Perturbation of Δx_{02}

Figure 3.7: MSE curves resulting from evaluation of reconstruction error for tiny perturbations in the initial values Δx_{01} and Δx_{02}.