2D Working Memory

A Report Submitted in Partial Fulfillment
of the Requirements for SYDE 556

Dane Corneil, 20169089, 4B
Tim Gevaert, 20171060, 4B

Faculty of Engineering
Department of Systems Design Engineering

April 27, 2009.
Course Instructor: Professor C. Eliasmith

1

System Description

As this project is an extension of the dynamic working memory example presented in section 8.3 of Neural Engineering [1], much of the system description overlaps. The subpopulation under consideration in that case was the Lateral Intraparietal Area (LIP) of the neocortex of macaque monkeys. This population exhibits the behaviour of storing memories of salient stimuli, and has been studied extensively by researchers. The results of these studies indicate that multiple bumps of varying heights can be encoded by the LIP to represent multiple stimuli in the spatial field (represented by ν), as well as a non-spatial characteristic of each stimulus $f(\nu)$. Research by Colby and Goldberg[2] suggests that $f(\nu)$ represents the attentional resources given to the stimuli, while research by Andersen et al.[3] suggests that it represents intention to move to the object.

In Neural Engineering, a working memory population was built which encoded a single non-spatial parameter in response to a spatial parameter. This representation was done in function space rather than vector space, significantly reducing the neural resources required (that is, by storing a limited number of function co-efficients rather than the value of the non-spatial parameter at every point in the range). This report extends that representation using two independent dimensions, and examines the resulting error in comparison to 1D representation.

Other neural populations have been found which exhibit similar activity bumps[4]. It would seem reasonable that populations could represent non-spatial characteristics

based on two varying spatial field dimensions $f(x1, x2)$ as well. Many other independent dimensions could also be accounted for in populations (e.g. colour), leading to activation levels which vary from neuron to neuron for no apparent reason.

For the case of 1D function representation[1], the function is represented through a basis decomposition.

$$x(\nu; \mathbf{A}) = \sum_m A_m \Phi_m(\nu) \text{ for } \mathbf{A} \in \rho(\mathbf{A}), \text{ and } m \in \{1, \cdots, M\} \tag{1}$$

Where the coefficients of $\mathbf{A}$ are stored in the integrating neural population.

Note that for the case of 2D function representation, the equivalent relation is in fact,

$$x(x_1, x_2; \mathbf{A}) = \sum_m A_m \Phi_m(x_1, x_2) \text{ for } \mathbf{A} \in \rho(\mathbf{A}), \text{ and } m \in \{1, \cdots, M\} \tag{2}$$

The distribution $\rho(A)$ is not known, so suitable values for $\mathbf{A}$ to define the extents of the function space were found by projecting exemplar memories onto the basis functions $\mathbf{\Phi}$. The extents of ν and x_1, x_2 were normalized to $[-1, 1]$.

To simulate the dynamics of the system, a modified integrator was implemented.

$$\dot{x} = Ax(t) + Bu(t) \tag{3}$$

Here, the standard implementation sets $A = I$ and $B = \tau_{PSC} I$. However, the value of B was set to I, allowing the integrator to charge fully within τ_{PSC} seconds ($0.1s$ in this model). Thus, memories were presented to the integrator for only $0.1s$, preventing error from accumulating while the memory was being stored.

2

Design Specification

Neural population simulation was done using the Nengo environment, available freely online. Basis function decomposition and examination of results was performed in MatLab™.

In order to compare 1D and 2D models accurately, the neural population size was held constant at 1089 neurons. In both 1D and 2D cases, the preferred directions for these neurons were represented by Gaussians of width $\sigma = 0.02$, spaced evenly across the input space. Thus, the encoding vectors were spread much more widely in the 2D case.

Gaussians significantly below width 0.02 could not be modeled accurately in 2D without increasing the resolution (set to $dx = 0.05$ on $[-1, 1]$), which in turn proved too computationally expensive. Thus, this encoding vector width was used and held constant across the 1D and 2D case. Exemplar memories were twice this width.

Memories were represented with function space constants by projecting them onto an orthonormal function basis. This basis was found by performing Singular Value Decomposition on a set of Gaussians, thus representing a basis which spanned the possible memories. However, not all constants could be represented in the system accurately, so a given number had to be selected. The singular values of the possible 1D and 2D memories were examined to determine an adequate functional space to use. Figure 2.1 shows the first 40 singular values in 1D and 2D.

In comparing 1D and 2D, a common number of dimensions was required in order

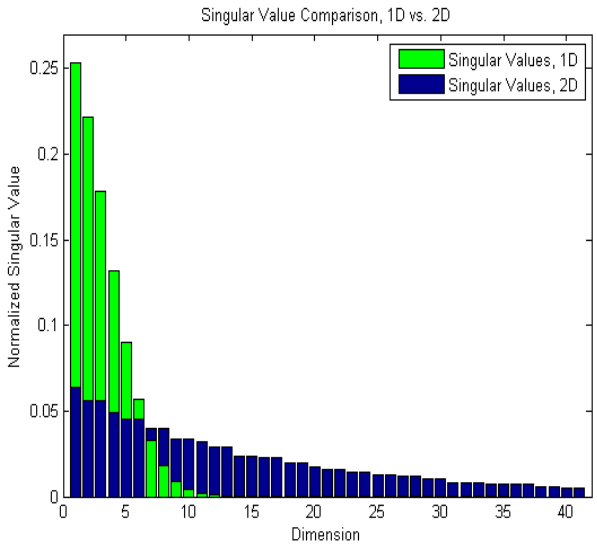

Figure 2.1: Singular Value Comparison

to avoid the effects of covering differently-sized function spaces. Based on the singular values, the first 20 dimensions were used, as they provided complete representation of the 1D memory space and adequate representation fo the 2D space. However, error was expected due to the fact that 2D memories cover a wider subspace of the functional space than 1D memories (most of the 1D constants were near zero).

Additionally, error was expected due to significant singular value representation beyond 20 dimensions in 2D. To examine this, 20 dimensional and 40 dimensional representations were compared within the 2D space.

Unlike the brain, modelling was not subject to spiking fluctuations (i.e. a rate model was used rather than a spiking model). This was done because spiking-induced fluctuations made it more difficult to visualize temporal changes in the signals due to fixed points, which was the focus of the comparison between 1D and 2D. If spiking fluctuations are used, they should be corrected for by a reasonable filter.

An example of a 2D memory being stored with a spiking model is shown in the attached file spikemovie.avi (with a filter size of 0.1, in a 2029-neuron, 40-dimensional network).

The integrator network included gaussian noise of 0.1 variance. Since the encoded

memories (Gaussians of width 0.04) could be encoded accurately with far fewer than 20 vectors dimensions in 1D, many of the first 20 singular values were less than the noise variance. Thus, some high frequency noise was introduced into the representation by using 20 constants in 1D.

Finally, the evaluation points used by Nengo in order to determine the optimal decoders covered the entire 20-dimensional space. As the possible constants for these values exist in a small subspace of the hypersphere, these decoders would be inadequate. Thus, evaluation points were used which only took on values within the possible subspace. These evaluation points were generated using the values taken by all possible memories, and adding points generated randomly within the maximum range of each constant. The additional points were necessary in order to represent values produced while the integrator was charging, and after error was introduced due to fixed points. In both 1D and 2D cases, 10,000 additional points were used.

3

Implementation

3.1 Comparison of error between a 1D and 2D function space

In order to examine how the representation error changes as the number of dimensions increase, a sequence of simulations were performed, where only the dimension of the represented function space is varied. In each case, a 20 dimensional neural integrator made up of a population of 1089 neurons was used to store the first 20 coefficients of the associated one or two dimensional basis functions. Standard values for the neuron LIF characteristics were used: $\tau_{RC} = 200ms$, $\tau_{Ref} = 1ms$, $\tau_{PSC} = 100ms$, maximum firing rates of $[200Hz - 500Hz]$, and intercepts along the full range of $[-1, 1]$. In each case, each memory bump is of height of 1.

1 Bump

When holding a Gaussian of height 1 in the centre of the function space, each simulation produced similar results. Each bump reduced in height slightly, and some high frequency information was lost, causing negative bumps on either side of the memorized bump. Interestingly, the 1D representation results in a higher RMS error than the $2D$ representation (Figure 3.1). This is reasonable after observing the Figures 3.2 and 3.3, which depict the final memory after $1s$. The $1D$ representation includes

negative values on either side of the bump. The $2D$ representation shows these same cavities, however there is some surface area which remains quite flat at the value of 0, thus improving the overall average representation error of the surface.

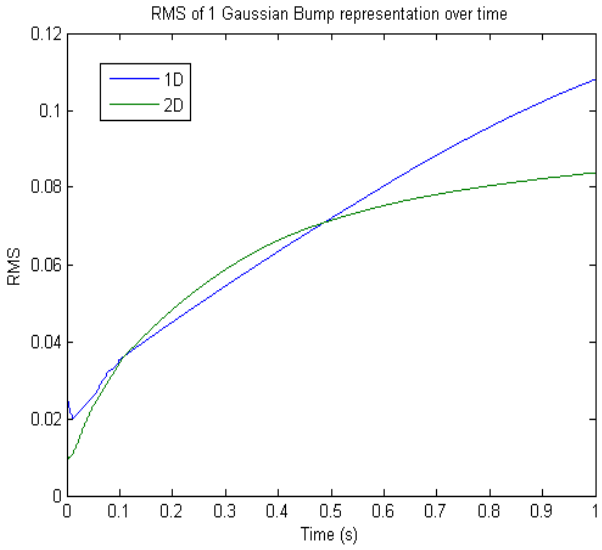

Figure 3.1: RMS of 1 Gaussian Bump representation over time

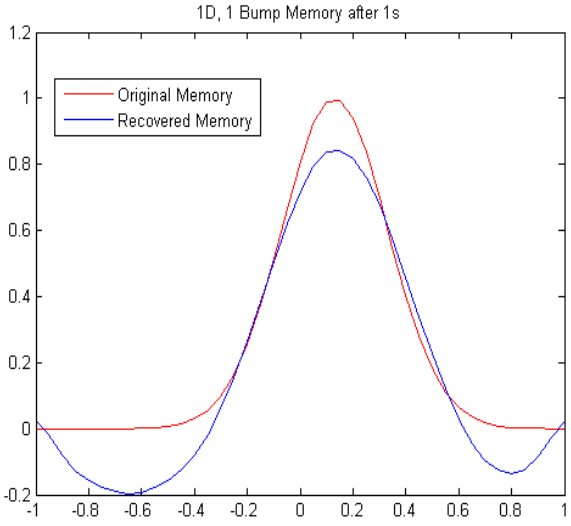

Figure 3.2: 1D, 1 Bump memory after 1s

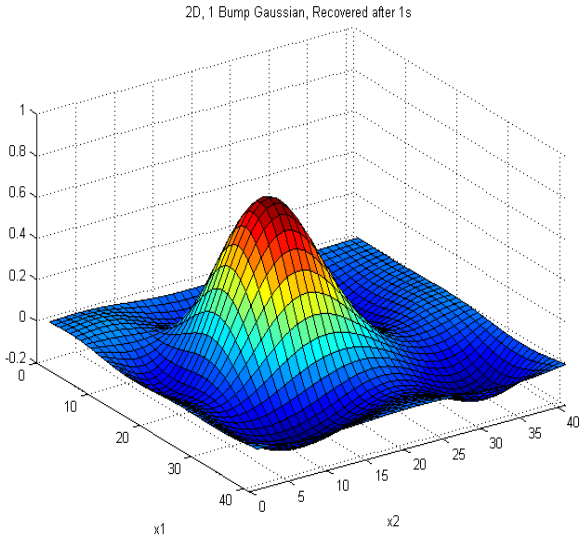

Figure 3.3: 2D, 1 Bump memory after 1s

2 Bumps

When representing two bumps, the RMS error of each simulation increased as expected (Figure 3.1). Again, the 2D representation (Figure 3.6) outperforms the 1D representation (Figure 3.5) in terms of RMS. A third representation is compared (Figure 3.7), which includes 2 bumps along the two axes of the 2 dimensional representation. In this simulation, the RMS error is roughly double that of the 2D 2 bump case.

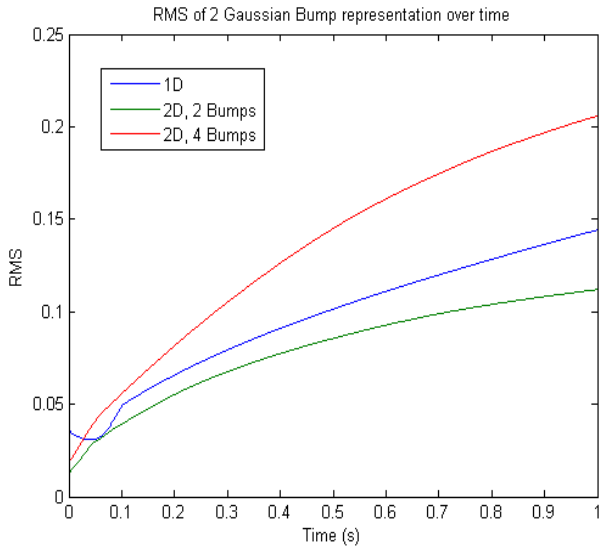

Figure 3.4: RMS of 2 Gaussian Bump representation over time

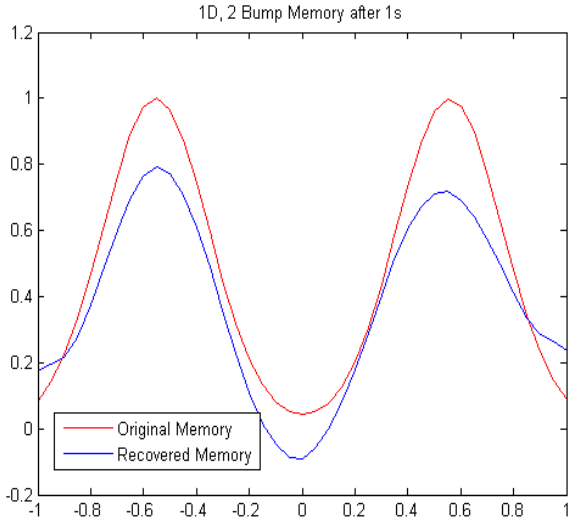

Figure 3.5: 1D, 2 Bump memory after 1s

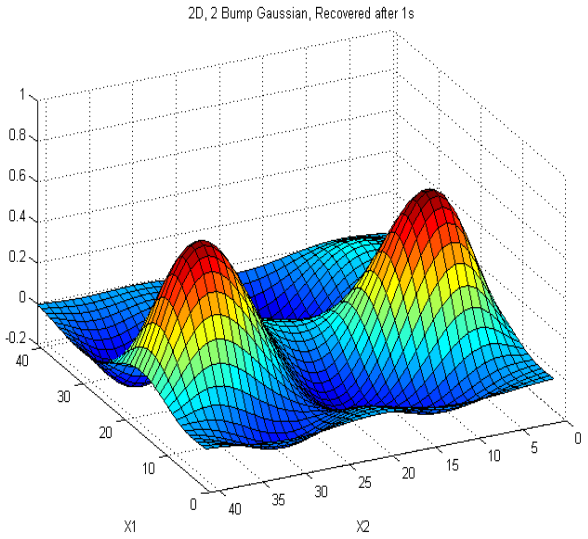

Figure 3.6: 2D, 2 Bump memory after 1s

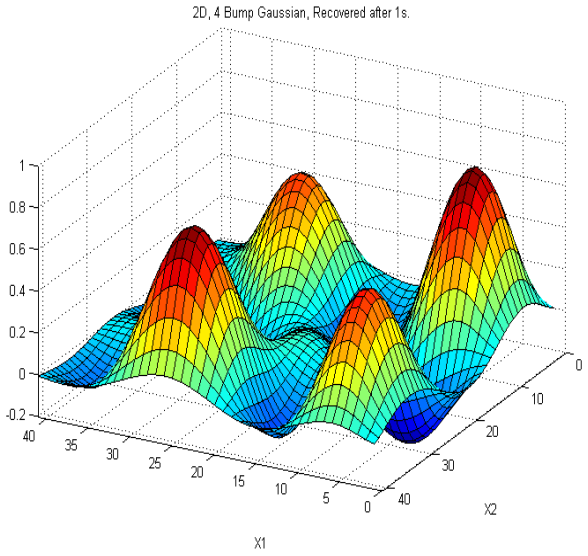

Figure 3.7: 2D, 4 Bump memory after 1s

3 Bumps

In comparing three bumps, it was not possible to place the bumps in the same plane without significant degradation, due to the limitation of 2D representation using 20 dimensions. This difficulty is examined in detail in the Effect of Dimensional Representation section. Instead, the 3 bumps in the 2D case were placed along the diagonal of the plane. Here, the error in the 2D representation is again superior to the 1D result in terms of RMS, and grows less quickly over time (Figure 3.8). By observing the Figures 3.9 and 3.10, it is clear why the 2D representation is superior to the 1D representation.

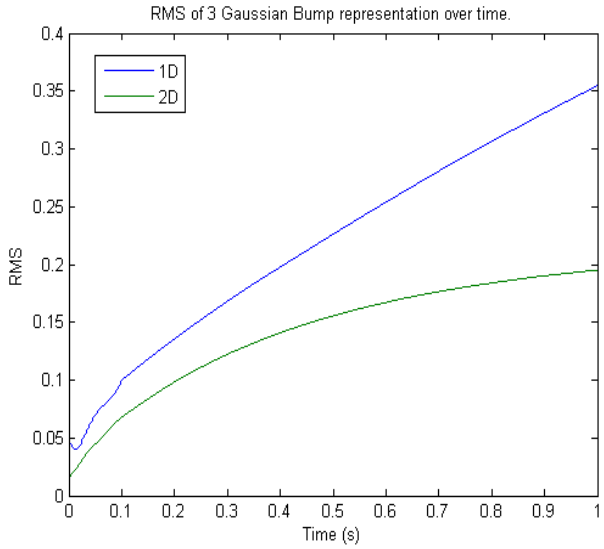

Figure 3.8: RMS of 3 Gaussian Bump representation over time

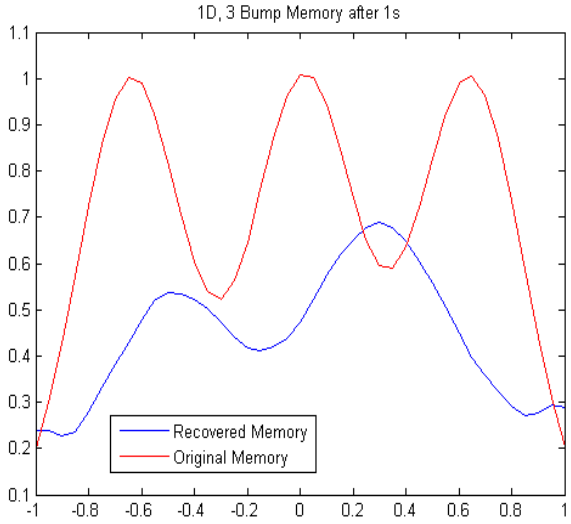

Figure 3.9: 1D, 3 Bump memory after 1s

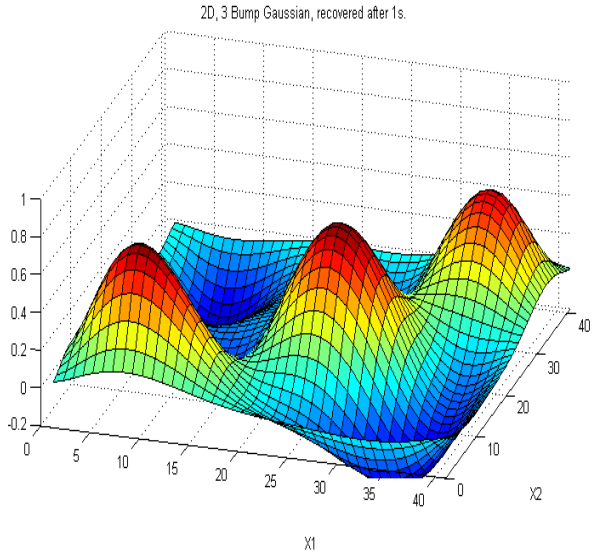

Figure 3.10: 2D, 3 Bump memory after 1s

The 1D case exhibits a large amount of forgetting, as the middle peak is completely missing from the representation after $1s$. As there is a larger function space over which the bumps can be placed in 2D, the 3 bumps do not interact as much, and hence can be stored distinctly. This effect is examined in greater detail at the end of the Stability Analysis section.

3.2 Effect of Dimensional Representation

As shown by the singular value distribution, the range of possible memories in 2D is not fully represented by 20 dimensions. In particular, densely populated memories show significant representation error. In order to examine this effect, the number of vector dimensions represented was increased to 40, while holding the neuron population size constant at 1089.

A memory distribution was used which was particularly difficult to represent with 20 dimensions: nine bumps distributed evenly throughout the graph. The original memory is shown below, as well as the projection of the memories on the first 20 and 40 basis dimensions.

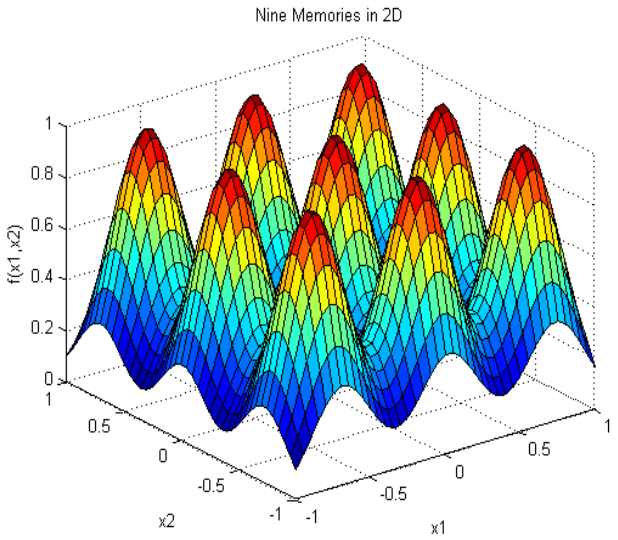

Figure 3.11: Original Memory

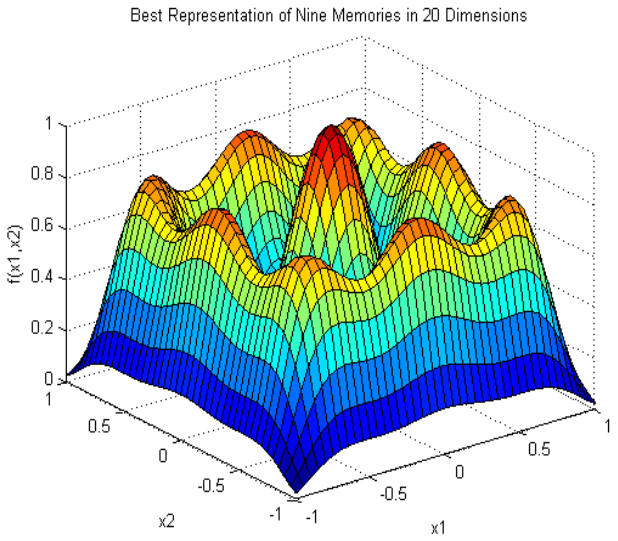

Figure 3.12: Memory Projected on 20 Dimensions

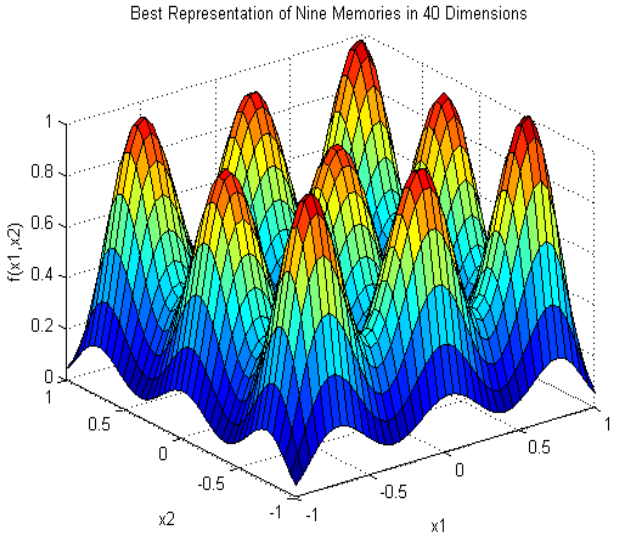

Figure 3.13: Memory Projected on 40 Dimensions

The higher frequency gaps between the memories are not possible in the 20 dimension representation; thus, the memories will already have begun to blend by the time when they are stored in the integrator.

Figures 3.14 and 3.15 show the results of storing the nine memories in the integrator for one second, using 20 and 40 dimensions respectively.

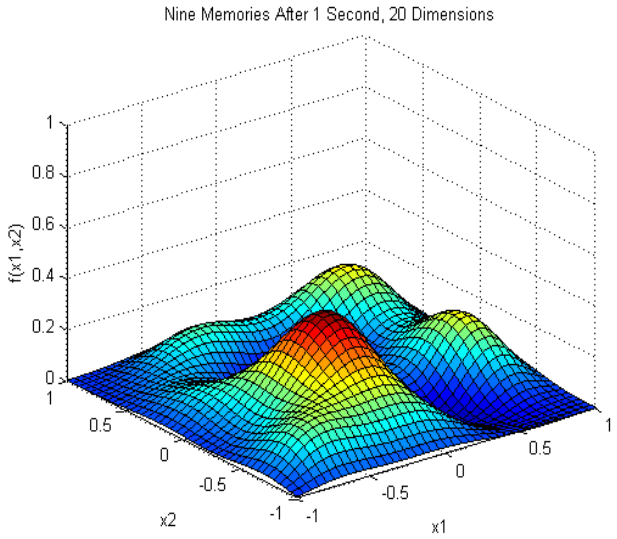

Figure 3.14: Memory After 1s, 20 Dimensions

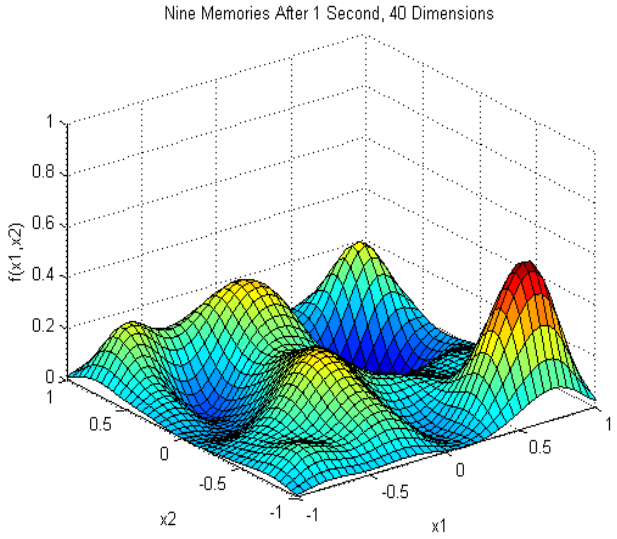

Figure 3.15: Memory After 1s, 40 Dimensions

Although 40 dimensions represent the function space better, the same number of neurons do not represent the increased vector space as well as with 20 dimensions. This results in the system fixed points being farther away from the accurate values of the constants, so that the values are not stored as accurately and attenuate more as time passes. This can be shown in the videos all920dim.avi and all940dim.avi.

Figure 3.16 compares the RMS error between 20 and 40 dimensional representations over time. The dotted lines show the error compared to the best possible representation of the memories given the number of dimensions used; the solid lines show the error compared to the original memory. The graph shows that the 20 dimensional representation performs very well against the memory projected on 20 dimensions, but poorly against the original. The error values for the 40 dimensional case are much closer, because the 40 dimensional projection is a more accurate representation of the original.

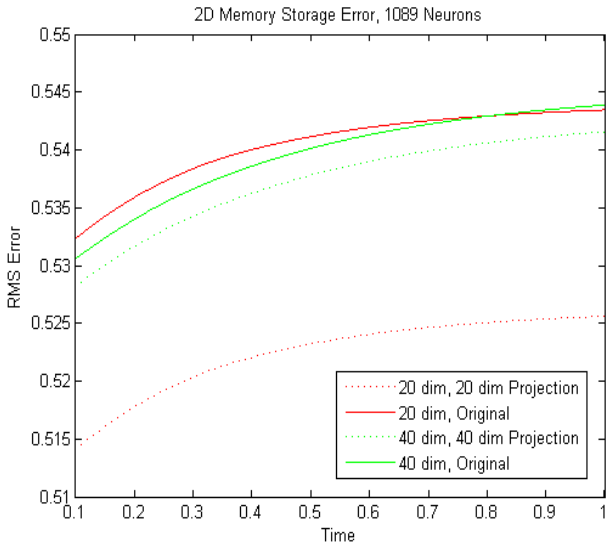

Figure 3.16: Error Comparison, 20 vs. 40 Dimensions

Immediately after storage, the 40 dimensional projection performs better; however, because the values attenuate more quickly, it is surpassed by the 20 dimensional projection after about 0.8 seconds. However, this takes into account error that may not be important for accurate recall (e.g. greater dips below the 0 plane between the

memory bumps). Figures 3.14 and 3.15 appear to show that the varied surface using 40 dimensions represents the stored memories more accurately after 1 second.

3.3 Effect of Population Size

In order to evaluate the effect of population size, the number of neurons was approximately doubled to 2209. The results after one second are shown in Figure 3.17, and an error comparison to a 40 dimensional representation using 1089 neurons is shown in Figure 3.18.

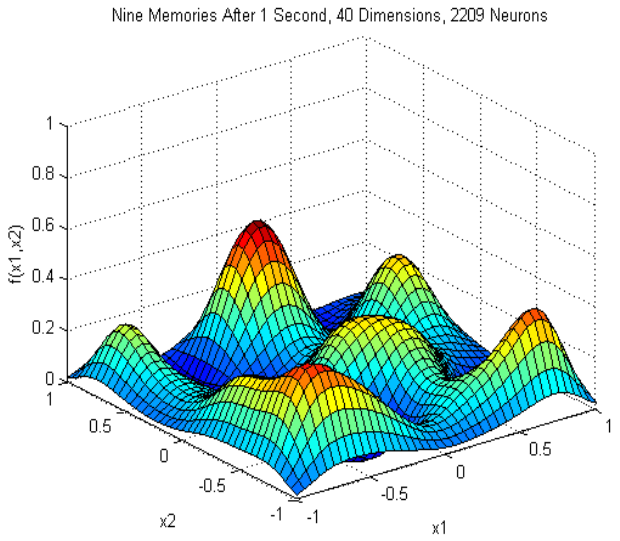

Figure 3.17: Memory After 1s, 2209 Neurons

Qualitatively, more memories are stored at comparable heights using 2209 neurons. As well, the RMS error is shifted down. However, the error does not attenuate any more slowly.

The storage of the memories using 2209 neurons is shown in the video all940dim2000.avi.

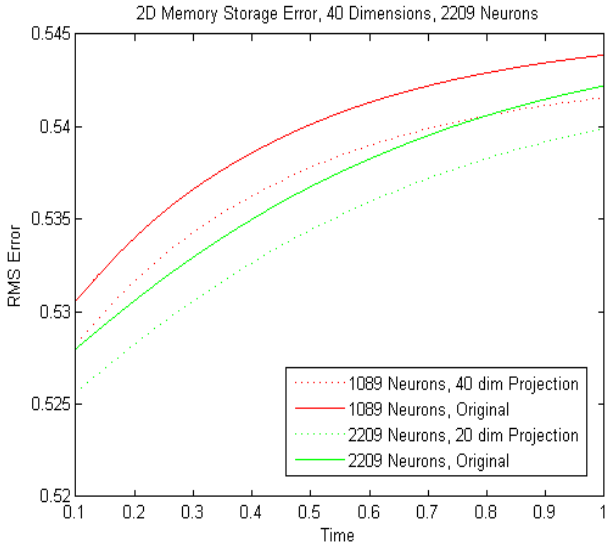

Figure 3.18: Error Comparison, 1089 vs. 2209 Neurons

3.4 Stability Analysis

Two particular types of stability decay are evident when encoding double Gaussian bumps in the 1D model. Bumps which are close together and of similar height tend to blend into a single bump over time, with a peak halfway between the two. When bumps are close together and of dissimilar height, the smaller bump tends to be absorbed by the larger one, while the centre of the larger bump does not change (that is, the system forgets the smaller bump). Examples of each are shown in Figures 3.19 and 3.20.

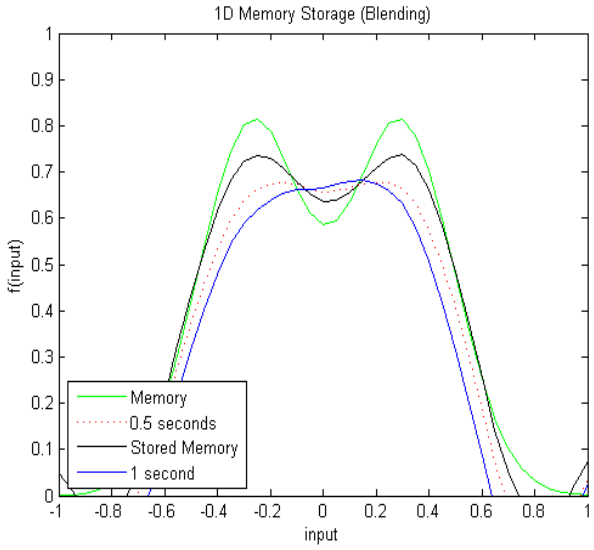

Figure 3.19: Blending in 1D

In both cases, the combined two-bump shape causes the first, and lowest frequency, dimensional constant to move outside of the range represented by the single-Gaussian encoding vectors. This saturates the system, causing the first constant to attenuate over time and resulting in a single bump. This effect is illustrated by Figure 3.21.

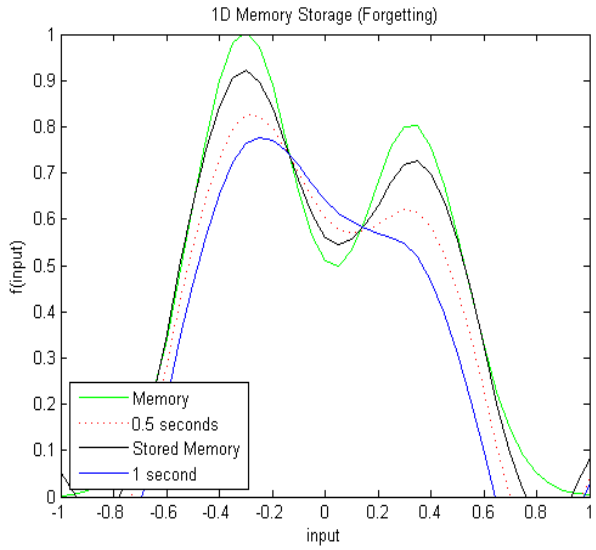

Figure 3.20: Forgetting in 1D

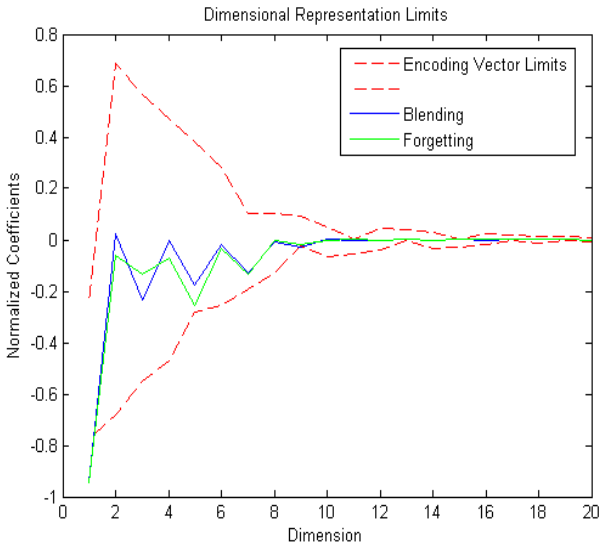

Figure 3.21: Constant Representation Limits

The Gaussian bumps used above were placed in the 2D space. The bump centres were placed at the same distance apart as in the 1D case in dimension $x1$, and at zero distance apart in dimension $x2$ (that is, on the same plane). Thus, the 1D and 2D results could be compared graphically by taking the projection on $x1$ in the 2D case.

In both the blending and forgetting cases, 20 dimensions were inadequate to represent the memories well in 2D space, exacerbating the respective errors before the memory had been stored. Thus, the memories were also stored in 40 dimensional function space, holding the population size constant. Figures 3.22 and 3.23 show the results for the forgetting scenario projected on input $x1$. The 40 dimensional representation maintained a more accurate representation of the primary memory, although forgetting is still evident in both cases.

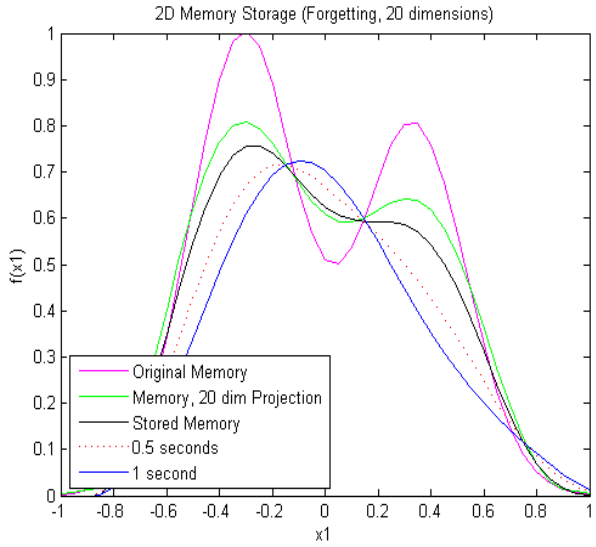

Figure 3.22: Forgetting in 2D, 20 Dimensions

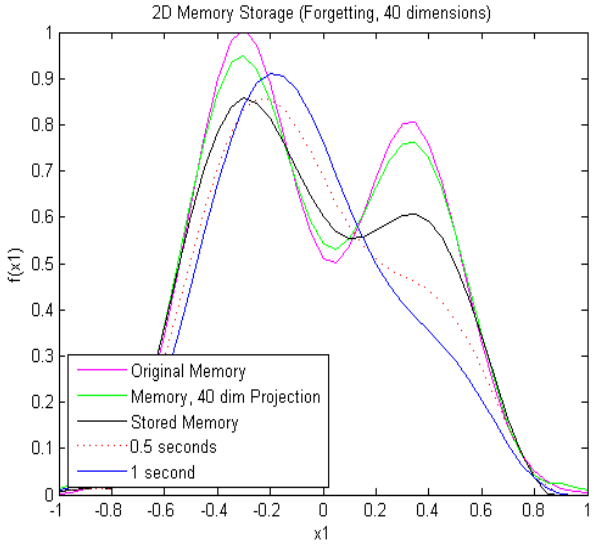

Figure 3.23: Forgetting in 2D, 40 Dimensions

Figures 3.24 and 3.25 show the results for blending. Interestingly, the final blended memory in both cases is at or above the heights of the original memory. This is potentially due to error in higher frequency dimensions, which vary significantly more in 2D than in 1D (as shown by the singular values). In the 40 dimensional case in particular, the original memories are stored with greater accuracy, but they lead to an emergent 'false memory' over time that is much greater in height than the original bumps.

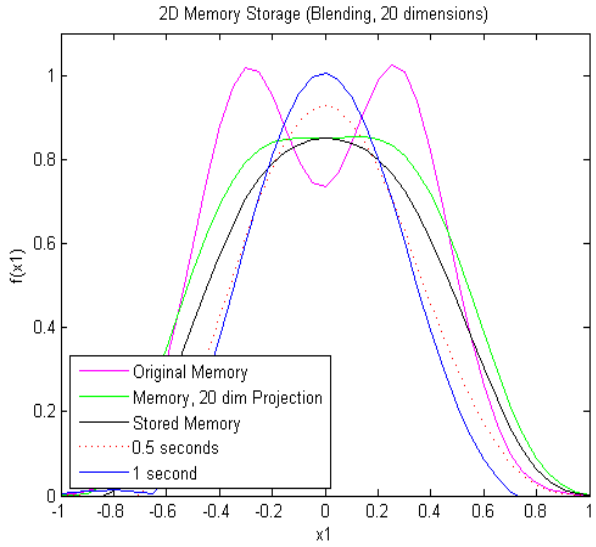

Figure 3.24: Blending in 2D, 20 Dimensions

In these cases, comparing RMS values is not particularly meaningful; for instance, a single large bump between two original memories might have a lower RMS error than an attenuated version of the original memories, although the latter is a better representation of the original function.

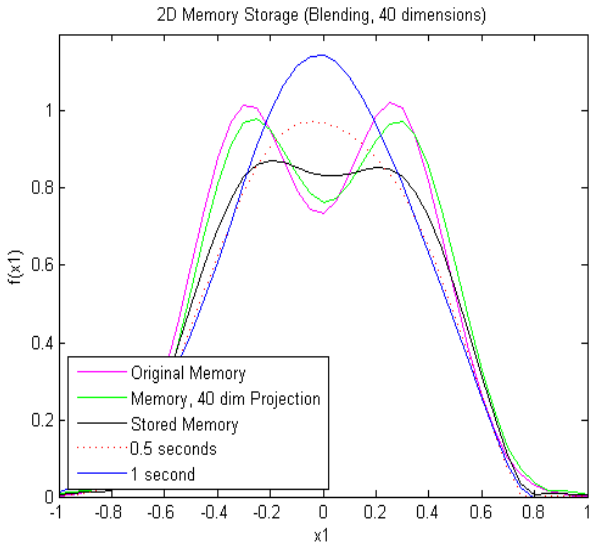

Figure 3.25: Blending in 2D, 40 Dimensions

Finally, a significant advantage of two spatial dimensions is that memories can be spaced farther across the space due to differences in the second dimension, preventing blending and forgetting errors. This was illustrated earlier when storing 3 bumps diagonally. To examine this, the two 'forgetting' bumps were spaced at the same distance along variable $x2$ as along $x1$, so that their centres were separated by $\sqrt{2}$ times the original distance. The value after one second is shown below, as well as the projection on dimension $x1$ (using 20 functional dimensions). Due to the greater distance between the centres, both memories can now be accurately recalled. Figures 3.26 and 3.27 show the stored memories after 1 second in 2 dimensions, and the storage over time projected on x1, respectively.

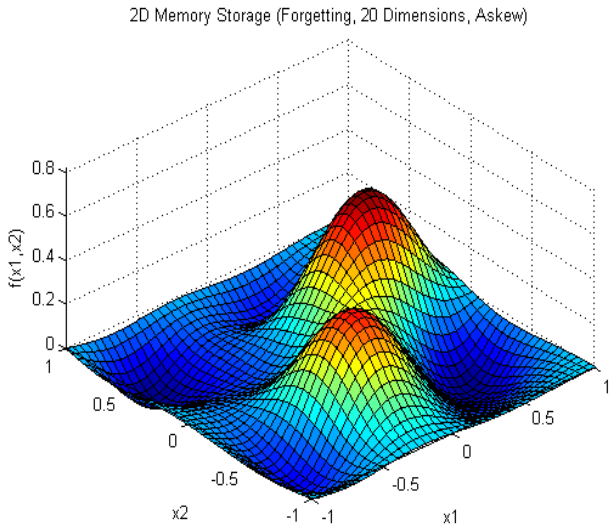

Figure 3.26: Forgetting in 2D, Askew Memories

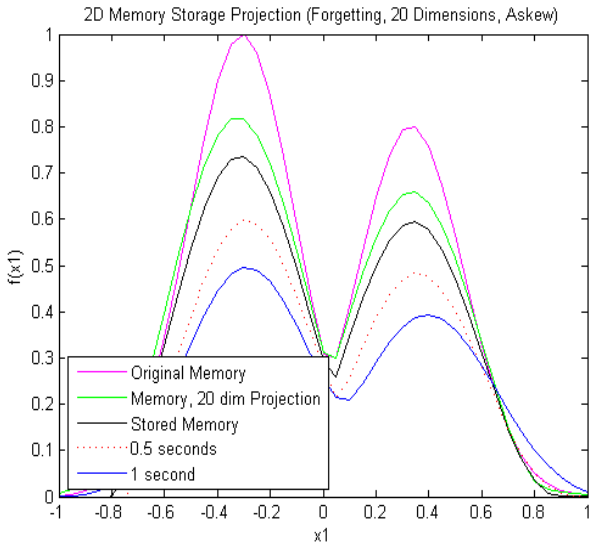

Figure 3.27: Forgetting in 2D, Askew Memories, Projection on x1

4

Discussion

Comparing 1 bump and well-spaced 2 bump memories in the 1D in 2D case revealed similar RMS error. This suggests that, for simple functions which can be well-represented in 2D with the first 20 dimensions and a low number of neurons, 1D and 2D error does not differ significantly.

In fact, the 2D case slightly outperformed the 1D case. This is potentially due to the error introduced by modeling unnecessary, higher dimensions in 1D. Future work would compare the results of freely varying this parameter between the 1D and 2D conditions.

Memories which were densely populated in the 2D plane, or spaced closely together, could not be well-represented in the 2D plane and attenuated quickly. This problem was addressed in several ways. By placing the memories diagonally in the 2D plane, so that they differed equally in both input dimensions, memories could be stored which would otherwise blend or forget. These 'askew' memories outperformed their 1D equivalents, illustrating how the same number of neurons can store some memories more accurately in 2D by utilizing their differences along both input dimensions.

As well, the number of dimensions was increased in 2D from 20 to 40. Without increasing the number of neurons, the 40-dimensional representation could remember more densely populated bumps. However, the error increased more rapidly during storage as the fixed points could less adequately represent the space. The error

was again improved by doubling the number of neurons, although the rate at which values approach fixed points did not improve significantly. It is possible that this could be addressed by increasing the number of evaluation points to represent the 40-dimensional space more fully.

As mentioned, future work should address the effect of freely varying the number of functional dimensions represented between the 1D and 2D case, in order to represent the space as accurately as possible. As well, the effect of varying the number of evaluation points within the N-dimensional subspace should be considered. It is possible that better evaluation points could be chosen by examining how constants vary in relation to each other, rather than choosing coordinates by randomly varying values within the possible range.

Finally, with greater computational resources, future work could focus on storing narrower Gaussian bumps in 1D and 2D, as well as increasing the population size.

References

[1] C. Eliasmith and C. H. Anderson, *Neural Engineering*. Massachusetts Institue of Technology, 2003.

[2] C. Colby and M. Goldberg, "Space and attention in parietal cortex," *Annual review of Neuroscience*, vol. 22, no. 1, pp. 319–349, 1999.

[3] R. Andersen, L. Snyder, D. Bradley, and J. Xing, "Multimodal representation of space in the posterior parietal cortex and its use in planning movements," *Annual Review of Neuroscience*, vol. 20, no. 1, pp. 303–330, 1997.

[4] K. Zhang, "Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory," *Journal of Neuroscience*, vol. 16, no. 6, pp. 2112–2126, 1996.