Supplemental Material to: An Ideal Observer Analysis of Visual Working Memory

Chris R. Sims, Robert A. Jacobs and David C. Knill
Department of Brain and Cognitive Sciences & Center for Visual Science
University of Rochester, Rochester, NY

Derivation of the optimal memory channel

In this section we derive the ideal observer model of visual working memory using results from rate–distortion theory. To begin our analysis, we assume that there is an information source in the world, labeled x, which can be characterized by a Gaussian distribution with mean μ_w and variance σ_w^2. We further assume that the process of sensory encoding can be approximated by adding corrupting zero-mean Gaussian noise to the true signal. This results in a distribution over afferent sensory signals which is again Gaussian, with mean μ_w and variance $\sigma_w^2 + \sigma_s^2$, where σ_s^2 is the variance of the corrupting sensory noise.

For many visual features, psychophysical performance is more accurately modeled by assuming that sensory noise is additive with respect to the logarithm of the physical stimulus value (this is just a restatement of the Weber-Fechner law). To apply our modeling framework to such cases, one can design an experiment in which the information source is log-normal distributed (or in other words, so that the logarithm of the feature value has a Gaussian distribution). In this case, the sensory signal x_s is assumed to represent the log-transform of the physical feature, so that $p(x_s)$ still follows a Gaussian distribution with mean and variance as given above.

The task is to specify a memory channel distribution, $p(x_m \mid x_s)$, which relates the output of VWM to its input, and show that this information channel is optimal for a fixed information rate.

The memory channel describes the combined effect of two processes, a memory encoding mechanism, and a memory decoder (this can be thought of as analogous to memory encoding and retrieval in VWM). For the time being, we assume a particular (noisy) memory encoding mechanism, and pair it with a decoder that is optimal with respect to the encoder. We subsequently show that our choice for the encoding mechanism does not impact the level of performance of the resulting model, as it is shown to achieve the theoretical rate–distortion bound. That is to say, although we assume a particular parametric model of memory encoding (namely, corrupting Gaussian noise), no alternative model can achieve lower estimation error for a given capacity. If x_s is the sensory signal that is input to VSTM, then we define its memory encoding, x_e as a Gaussian distribution with mean x_s and variance σ_e^2.

The output of VWM represents an attempt to reconstruct the incoming sensory signal in face of the degradation incurred as part of the encoding process. (Note that since the af-

ferent signal is drawn from a continuous distribution, a degraded representation is a necessary consequence for any system with finite capacity). We design the decoding process to be Bayes' optimal with respect to the encoding mechanism, or in other words, to produce the maximum a posteriori (MAP) estimate of x_s given the (noisy) encoded signal x_e. From Bayes' theorem, we have

$$p(x_s \mid x_e) \propto p(x_e \mid x_s)p(x_s). \tag{1}$$

To find the MAP estimate of x_s, we take the derivative of the above expression and solve for the point where the posterior is maximum. For the case of a Gaussian signal and Gaussian encoding noise, the resulting MAP estimate is also the minimum mean-squared error (MMSE) estimator. We label this estimate x_m, which represents the best possible reconstruction of the incoming sensory signal given the encoded representation x_e. The equation for computing x_m for a particular value x_e is given by

$$x_m \mid x_e = \frac{\mu_w \sigma_e^2 + x_e \left(\sigma_w^2 + \sigma_s^2\right)}{\sigma_w^2 + \sigma_e^2 + \sigma_s^2}. \tag{2}$$

Note that this is essentially a weighted combination of the encoded signal and the mean of the distribution of the information source, where the optimal weight depends on the statistics of the source and the information channel. Since the encoding mechanism is a stochastic process, we can marginalize over x_e to derive the memory channel distribution, or the probability distribution over the output of VSTM, conditioned on a particular input sensory signal:

$$\begin{aligned} p(x_m \mid x_s) &= \int_{-\infty}^{\infty} p(x_m \mid x_e)p(x_e \mid x_s)\,\mathrm{d}x_e \\ &= \text{Normal}(\mu_m, \sigma_m^2) \\ \mu_m &= \frac{\mu_w \sigma_e^2 + x_s \left(\sigma_w^2 + \sigma_s^2\right)}{\sigma_w^2 + \sigma_e^2 + \sigma_s^2} \\ \sigma_m^2 &= \frac{\sigma_e^2 \left(\sigma_w^2 + \sigma_s^2\right)^2}{\left(\sigma_w^2 + \sigma_e^2 + \sigma_s^2\right)^2}, \end{aligned} \tag{3}$$

where in the above equation we have relied on the fact that $p(x_m \mid x_e)$ has a Dirac delta distribution,

$$p(x_m \mid x_e) = \delta\left(x_m - \frac{\mu \sigma_e^2 + x_e \left(\sigma_w^2 + \sigma_s^2\right)}{\sigma_w^2 + \sigma_e^2 + \sigma_s^2}\right). \tag{4}$$

For the channel distribution given by equation (3), we can compute the information rate R:

$$\begin{aligned} R = I(x_s, x_m) &= \iint p(x_m \mid x_s)p(x_s)\log\left(\frac{p(x_m \mid x_s)}{p(x_m)}\right)\,\mathrm{d}x_s\mathrm{d}x_m \\ &= -\frac{1}{2}\log\left(\frac{\sigma_e^2}{\sigma_w^2 + \sigma_e^2 + \sigma_s^2}\right). \end{aligned} \tag{5}$$

From this expression we see that the effective information rate of the channel is governed by the variance of the information source, sensory noise, and the encoding noise. Under the assumption that VSTM has a fixed information rate of R, we can then solve this equation for σ_e. This allows us to specify the channel directly in terms of its information rate, rather than the equivalent magnitude of encoding noise. In particular, we obtain

$$\sigma_e^2 = \frac{\sigma_w^2 + \sigma_s^2}{e^{2R} - 1}. \tag{6}$$

Note that in this supplemental document, logarithms are taken to be natural logarithms, so that the information rate R is measured in units of *nats* (the natural logarithm equivalent of bits). In the main manuscript, all quantities have been converted to units of bits.

Thus, if we wish our memory channel to achieve a given information rate R nats, we can use equation (6) to determine the equivalent level of encoding noise that achieves this rate. Substituting equation (6) into the memory channel equation (3), we arrive at the equation for the memory channel given in the main text,

$$\begin{aligned} p(x_m \,|x_s) &= \text{Normal}\left(\mu_m,\, \sigma_m^2\right) \\ \mu_m &= x_s + e^{-2R}\left(\mu_w - x_s\right), \\ \sigma_m^2 &= e^{-4R}\left(e^{2R} - 1\right)\left(\sigma_w^2 + \sigma_s^2\right). \end{aligned} \tag{7}$$

To verify that this model of VWM is optimal, we compute its information rate and distortion, and show that it achieves the theoretical bound given by rate–distortion theory. The rate for the channel is simply given by R, since the channel was designed to achieve this particular information rate. To compute the distortion associated with the channel it is necessary to pick a particular cost function on performance. In our analysis, we assume that a useful goal for the brain is to minimize the average squared error between an incoming sensory signal and its decoded representation. This objective is assumed to be useful for any of the tasks in which VWM plays an important role, as regardless of the task, the potential for optimal performance is highest when the contents of memory closely correspond to the veridical state of the world. With the choice of this cost function, we compute the channel distortion as follows:

$$\begin{aligned} D &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left(x_m - x_s\right)^2 p(x_m \mid x_s) p(x_s)\, \mathrm{d}x_m \mathrm{d}x_s \\ &= e^{-2R}\left(\sigma_w^2 + \sigma_s^2\right). \end{aligned} \tag{8}$$

Finally, by solving equation (8) for R, we obtain the theoretical rate–distortion bound for a Gaussian information source, with variance $(\sigma_w^2 + \sigma_s^2)$ and squared-error distortion measure (Berger, 1971):

$$R = \frac{1}{2}\log\left(\frac{\sigma_w^2 + \sigma_s^2}{D}\right). \tag{9}$$

Although in our derivation we made the assumption that the memory encoding process can be described by Gaussian corrupting noise, it turns out that this assumption did not impact

the optimality of the resulting memory system. The rate–distortion analysis of the memory channel demonstrates that there cannot exist any alternative memory model that achieves better performance (lower distortion) using less capacity to encode information.

Finally, the output of the memory channel represents an optimal estimate of the incoming sensory signal, subject to a fixed information rate limit. For an individual performing a specific task, the true value of the information signal, x, is generally of more importance than its noisy sensory encoding x_s. Although the true signal x is not accessible to the individual, it is possible to compute an optimal estimate of this value under the assumption that knowledge of the statistics of the sensory noise and the information source is available. As before, we assume that the agent computes the maximum a posteriori estimate of x, given x_m. By applying Bayes' theorem, we have $p(x \mid x_m) \propto p(x_m \mid x)p(x)$. Taking the derivative of this expression and solving for its maximum, we arrive at the optimal estimate, $\hat{x}$, of the true signal in the world x:

$$\hat{x} = \frac{x_m \sigma_w^2 + \mu_w \sigma_s^2}{\sigma_w^2 + \sigma_s^2}. \tag{10}$$

For the case of a Gaussian channel distribution given by equation (7), the maximum a posteriori estimate is again identical to the minimum mean square error (MMSE) estimator. While the estimate $\hat{x}$ is a deterministic function of the output of the memory channel x_m, its value conditioned on the true stimulus x is a random variable with probability density $p(\hat{x} \mid x)$:

$$\begin{aligned} p(\hat{x} \mid x) &= \text{Normal}\left(\mu_{\hat{x}}, \sigma_{\hat{x}}^2\right) \\ \mu_{\hat{x}} &= \frac{x\,\sigma_w^2 + e^{-2R}(\mu_w - x)\sigma_w^2 + \mu_w \sigma_s^2}{\sigma_w^2 + \sigma_s^2} \\ \sigma_{\hat{x}}^2 &= \frac{e^{-4R}\left(-1 + e^{2R}\right)\sigma_w^4\left(\sigma_w^2 + e^{2R}\sigma_s^2\right)}{\left(\sigma_w^2 + \sigma_s^2\right)^2}. \end{aligned} \tag{11}$$

Bias of the optimal memory channel

For the optimal memory system defined by equation (11), we can compute the predicted bias of the memory representation as a function of the true feature value that is input to memory. In particular, the estimator bias is defined as

$$\begin{aligned} \text{Estimator bias} &= \int (\hat{x} - x) p(\hat{x} \mid x) \mathrm{d}\hat{x} \\ &= -\frac{(x - \mu_w)e^{-2R}(\sigma_w^2 + e^{2R}\sigma_s^2)}{\sigma_w^2 + \sigma_s^2} \end{aligned} \tag{12}$$

From this equation, it is seen that the bias depends linearly on the difference between the feature value x and the mean over features in the environment. Further, the amount of bias depends on the information rate, R. In particular, as more items are stored concurrently in memory, the capacity allocated to each decreases, predicting that absolute bias should increase with increasing set size. The predicted effects of feature variance are less straightforward: increasing feature variance leads to a less-informative prior (suggesting smaller bias), but also

leads to noisier memory representations (predicting larger bias). Thus, depending on the specific numerical value of the information rate R, the estimator bias could either be larger or smaller for an information source with increased feature variance.

Derivation of the optimal decision rule for the psychophysical experiments

The previous section derived the optimal memory channel, defined as the system that achieves minimal memory distortion subject to a constraint on the total information rate of memory. In this section, we present the optimal decision rule describing how an ideal observer should (optimally) use the information available in memory to form its decision in the psychophysical experiments presented in the main manuscript. For the sake of concreteness, consider the orientation experiment where the probe stimulus is obtained by applying a clockwise or counterclockwise perturbation to the original stimulus orientation.

The optimal decision rule for maximizing the probability of being correct in the task used in the experiment is a deterministic rule that computes the decision variable

$$\begin{aligned} &\Delta_s = y_s - \hat{x}, \\ &\text{Decision} = \begin{cases} \text{Clockwise}, & \Delta_s > 0 \\ \text{Counterclockwise}, & \Delta_s < 0 \end{cases} \end{aligned} \tag{13}$$

where y_s is the noise-corrupted sensory representation of the probe orientation and $\hat{x}$ is the best estimate of the original stimulus orientation conditioned on the noisy representation in memory. The decision variable Δ_s is the difference between these two quantities. It is not, as one might expect, the difference between the best estimate of the probe orientation conditioned on the two available sources of information and the original orientation conditioned on the two available sources of information.

To derive this decision rule, we first note that the memory encoder is formally equivalent to a system in which the encoded variable is simply the true orientation plus constant variance, additive Gaussian noise (this is shown in the previous section), where the amount of noise is determined by the constraint on the information rate. Thus, we can model the memory representation of the original stimulus orientation as,

$$x_e = x + N, \tag{14}$$

where x is the true orientation and the encoded value is x_e. The variable N is Gaussian noise whose variance is due to the combined effects of sensory noise and the memory encoding noise. On any given trial, the best estimate of x (labeled $\hat{x}$) is given by

$$\hat{x} = w_m x_e + (1 - w_m)\mu_w \tag{15}$$

where μ_w is the mean of the prior distribution of orientations in the stimulus set. The weight (w_m) given to the encoded value is inversely related to the variance of the encoding noise. As memory capacity goes to 0, for example, the variance of the effective encoding noise goes to infinity and w_m approaches 0, implying that for a memory capacity of 0, the ideal observer's best estimate of the original stimulus orientation is simply the mean of the prior distribution of orientations. The crucial step in the derivation of the ideal decision rule is that we can express

the relationship between the true orientation of a stimulus and the estimate derived from the encoded memory representation in one of two ways, as

$$\hat{x} = w_m(x + N) + (1 - w_m)\mu_w, \tag{16}$$

or

$$x = \hat{x} + N', \tag{17}$$

where N and N' are Gaussian noise sources. In the second representation, (17), the variance of N' is equal to the variance of the posterior probability distribution $p(x \mid x_e)$, which is constant for all x_e. As an example, as the variance of encoding noise goes to infinity, the variance of N' reduces to the variance of the prior over x and the variance of $\hat{x}$ itself goes to 0.

The next step in the derivation of the decision rule is to note that the sensory representation of the probe stimulus is given by

$$y_s = x + \Delta + N_s, \tag{18}$$

where N_s is another Gaussian noise source representing the sensory noise, and Δ is the actual perturbation added to x on the trial. Combining (17) and (18), we obtain for the sensory signal representing the probe orientation

$$y_s = \hat{x} + \Delta + N'', \tag{19}$$

where N'' represents the sum of the two independent Gaussian noise sources, N' and N_s. The likelihood of the probe stimulus, conditioned on Δ and the optimal memory estimate is then given by

$$p(y_s \mid \Delta, \hat{x}) = p_{N''}\left(N'' = (y_s - \hat{x}) - \Delta\right). \tag{20}$$

Finally, by applying Bayes' rule, we obtain the posterior distribution over Δ, conditioned on y_s and $\hat{x}$:

$$\begin{aligned} P(\Delta \mid \hat{x}, y_s) &\propto p(y_s \mid \Delta, \hat{x}) P(\Delta) \\ &= p_{N''}\left(N'' = (y_s - \hat{x}) - \Delta\right) P(\Delta) \end{aligned} \tag{21}$$

The optimal decision rule (for maximizing probability correct) is given by selecting clockwise when $P(\Delta > 0 \mid \hat{x}, y_s) > 0.5$ and counterclockwise otherwise. Assuming that the prior on Δ is symmetric around 0 (as is true in our experiments), this condition is true when $y_s - \hat{x} > 0$. This leads to the decision rule given previously in (13). Importantly, the decision rule depends on the *difference* between the sensory encoded probe orientation and the optimal memory for the original stimulus orientation, *regardless of the actual values of the two*. As the ideal observer model bases its decisions on the difference between the sensory representation of the probe item, and its optimal memory reconstruction of the original stimulus, it achieves optimal performance for the task. Note that if the decision was instead based on comparison of the noisy encoding of the memory item, x_e, rather than its optimal reconstruction, $\hat{x}$, the decision rule would be suboptimal.

Details of the information-theoretic models

In our experiments, subjects were presented with a task that required encoding multiple items in VSTM. The main text evaluates several mechanisms for how items are selected to be stored, and how a central information-theoretic memory capacity is divided among encoded items. In this section, we provide further details on the implementation of each model.

Continuous resource model

The continuous resource model (Wilken & Ma, 2004; Bays & Husain, 2008) assumes that all stimulus items are encoded, but with precision that depends on the number of objects simultaneously stored in VWM. Our information-theoretic implementation of the continuous resource model formalizes this property in the following manner. If a subject has a capacity of R nats, and there are n stimulus items on the current trial, then R/n nats are allocated per item, so that the available capacity is evenly distributed among every item in the scene.

In this case, the derivation of the encoding and decoding process follows exactly as before, except now the variance of the encoding noise added to each stimulus item is given by

$$\sigma_e^2 = \frac{\sigma_w^2 + \sigma_s^2}{e^{2R/n} - 1}. \tag{22}$$

This is identical to equation (6), except the allocated capacity for each item is R/n nats. The rest of the derivation follows as before, leading to a distribution for the optimal estimate, conditioned on the true stimulus feature, $p(\hat{x} \mid x)$:

$$\begin{aligned} p(\hat{x} \mid x) &= \text{Normal}\left(\mu_{\hat{x}}, \sigma_{\hat{x}}^2\right) \\ \mu_{\hat{x}} &= \frac{x\,\sigma_w^2 + e^{-2R/n}(\mu_w - x)\sigma_w^2 + \mu_w \sigma_s^2}{\sigma_w^2 + \sigma_s^2} \\ \sigma_{\hat{x}}^2 &= \frac{e^{-4R/n}\left(-1 + e^{2R/n}\right)\sigma^4\left(\sigma_w^2 + e^{2R/n}\sigma_s^2\right)}{\left(\sigma_w^2 + \sigma_s^2\right)^2}. \end{aligned} \tag{23}$$

When the probe item is displayed, it is subject to the same sensory uncertainty that was applied to the original stimulus. If the true stimulus feature is given by x, and the perturbation applied is Δ, then the sensory representation of the probe item, y_s, is distributed according to a Gaussian distribution: $p(y_s \mid x) = \text{Normal}\,(x + \Delta, \sigma_s^2)$.

The subject generates a response by comparing the sensory noise corrupted probe item (y_s), to the optimal memory reconstruction of the original stimulus ($\hat{x}$, given by equation 23). Since both of these are Gaussian random variables, their difference also follows a Gaussian distribution. Conditioned on the true stimulus value x, and the perturbation Δ, the distribution governing this difference is

$$
\begin{aligned}
p(\text{difference} \mid x, \Delta) &= \text{Normal}\left(\mu_{\text{diff}}, \sigma^2_{\text{diff}}\right) \\
\mu_{\text{diff}} &= x + \Delta - \frac{x\sigma_w^2 + \mu_w(\sigma_e^2 + \sigma_s^2)}{\sigma_w^2 + \sigma_e^2 + \sigma_s^2} \\
\sigma^2_{\text{diff}} &= \sigma_s^2 + \frac{\sigma_w^4\left(\sigma_e^2 + \sigma_s^2\right)}{\left(\sigma_w^2 + \sigma_e^2 + \sigma_s^2\right)^2}
\end{aligned}
\tag{24}
$$

Whenever the probe item is perceived to be clockwise (in the orientation experiment) with respect to the memory estimate, the subject will report a positive perturbation. The probability of the subject reporting a positive perturbation is therefore $P(\text{difference} \mid x, \Delta) > 0$. This quantity is determined from the cumulative density function (CDF) for a Gaussian distribution with mean and variance given in (24). Since we also allow lapse trials, in which the response is assumed to be generated randomly, the overall probability of reporting a positive perturbation for a given stimulus value and perturbation, is

$$
P(+ \mid x, \Delta) = \lambda \cdot \frac{1}{2} + (1 - \lambda) \cdot \left[1 - \Phi\left(0 \mid \mu_{\text{diff}}, \sigma^2_{\text{diff}}\right)\right], \tag{25}
$$

where λ is the lapse rate, and $\Phi\left(\cdot \mid \mu, \sigma^2\right)$ indicates the Gaussian CDF with given mean and variance. Code for the continuous resource model is provided in Listing 2. For comparison, code for the non-information-theoretic version of the continuous resource model is provided in Listing 3.

Slots+averaging model

The slots+averaging model (Zhang & Luck, 2008; Cowan & Rouder, 2009), assumes that there is a limit, k, on the number of items that can be stored simultaneously in VWM. Our information-theoretic implementation of this model also assumes that there is an information limit on each slot, so that each slot can represent an item with R/k nats of precision. As a result, there is a maximum information capacity of R nats available to the system, and this capacity can only be allocated in discrete units of R/k nats. If there are fewer than k items in a scene, each object is stored in one slot. Any extra slots are randomly distributed among the items, so that there is a non-zero chance that a single item will be encoded in multiple slots. In our implementation, we assume that if an item is stored in 2 slots, it is represented using $2 \cdot (R/k)$ nats. The equation describing the probability distribution over the number of slots, j, assigned to a given item, $P(j \mid k, n)$ is given by

$$
p(j \mid k, n) = \begin{cases}
k/n, & \text{if } j = 1 \text{ and } k \leq n, \\
(1 - k/n), & \text{if } j = 0 \text{ and } k \leq n, \\
0, & \text{if } j = 0 \text{ and } k > n, \\
\binom{k-n}{j-1}\left(\frac{1}{n}\right)^{(j-1)}\left(1 - \frac{1}{n}\right)^{(k-n-j+1)}, & \text{if } j > 0 \text{ and } k > n.
\end{cases}
\tag{26}
$$

Under the slots+averaging model, the probability of reporting a positive perturbation depends on whether the probed item was one of the stimulus items that were encoded during the stimulus presentation interval. For items that were not encoded (when $j = 0$), responses

are made using the "educated guessing" model described in the main text. For a given probe stimulus, the optimal probability that a positive perturbation occurred is defined by

$$\begin{aligned} P(\Delta > 0 \mid \text{probe}) &= \frac{p(\text{probe} \mid \Delta > 0)P(\Delta > 0)}{p(\text{probe} \mid \Delta > 0)P(\Delta > 0) + p(\text{probe} \mid \Delta < 0)P(\Delta < 0)}, \\ p(\text{probe} \mid \Delta > 0) &= \sum_{\Delta > 0} p(\text{probe} \mid \Delta, \mu_w, \sigma_w^2 + \sigma_s^2)P(\Delta), \\ p(\text{probe} \mid \Delta < 0) &= \sum_{\Delta < 0} p(\text{probe} \mid \Delta, \mu_w, \sigma_w^2 + \sigma_s^2)P(\Delta), \end{aligned} \tag{27}$$

where Δ refers to the discrete perturbation levels in a given experiment. The likelihood functions $p(\text{probe} \mid \Delta, \mu_w, \sigma_w^2 + \sigma_s^2)$ are normal distributions with mean $\mu_w + \Delta$, and variance $\sigma_w^2 + \sigma_s^2$. Note that σ_w^2 is different in each variance condition of the experiment, and as a result, the probability of a positive perturbation based solely on the probe stimulus will be lower as the feature variance increases.

The "optimal guessing strategy" is to respond that a positive perturbation occurred whenever $P(\Delta > 0 \mid \text{probe}) > 0.5$. However, individual subjects in the experiment may differ in terms of their use of this strategy. It is also possible that subjects may probability match: if the probability that the probe item was perturbed positively is 0.7, a probability matching subject will respond with a positive perturbation only 70% of the time. To account for these possibilities, we model the subject's probability of reporting a positive perturbation on trials when the probe item was not encoded as

$$P(+ \mid \text{probe}) = \frac{P(\Delta > 0 \mid \text{probe})^{\psi}}{P(\Delta > 0 \mid \text{probe})^{\psi} + (1 - P(\Delta > 0 \mid \text{probe}))^{\psi}}. \tag{28}$$

The parameter ψ controls the extent to which subjects make educated guesses: when $\psi = 0$, guesses are made completely randomly; when $\psi = 1$ probability matching results, and as $\psi \to \infty$ the subject approaches optimal performance.

For items that were encoded in at least one slot ($j > 0$), the response probability is similar to the continuous resource model, except now the precision of memory representations is determined not by the number of items in the display, but rather the number of slots allocated to the probed item. If there are k total slots and j of these ($j > 0$) were used to encode the probe item, then the encoding noise is given by

$$\sigma_e^2 = \frac{\sigma_w^2 + \sigma_s^2}{e^{2j \cdot R/k} - 1}. \tag{29}$$

From this, the probability distribution over the difference between the sensed probe item and the memory representation follows the derivation given for the continuous resource model (equations 24 and 25). However, since the number of slots assigned to the probe item is an unknown quantity, it is necessary to marginalize over its value. Incorporating the possibility of lapse trials, we arrive at

$$P(+ \mid x, \Delta, k, n) = \lambda \cdot \frac{1}{2} + (1 - \lambda) \cdot \left[\sum_{j=0}^{k} P(+ \mid x, \Delta, j, k) P(j \mid k, n) \right]. \tag{30}$$

Code for the continuous resource model is provided in Listing 4. For comparison, code for the non-information-theoretic version of the slots+averaging model is provided in Listing 5.

Binomial and flexible encoding models

We also consider two more general models, the binomial encoding, and flexible encoding models. Each of these assume a total information capacity of R nats, where the capacity is evenly divided among all items stored in VWM. The models differ in terms of how many items are stored on each trial. The binomial encoding model assumes that each additional stimulus item is stored with probability θ. This leads to a distribution over the number of stored items given by a binomial distribution with success parameter θ. In our implementation, we allow this parameter to differ in different conditions of the experiment. if n items are encoded on a trial, then each receives R/n nats of capacity. Thus, on different trials, a subject may encode more items with lower precision, or fewer items with higher precision.

The flexible encoding model similarly allows for a probability distribution over the number of encoded items. However, it does not constrain this distribution to be a binomial distribution. Instead, the probabilities of encoding $0 \ldots n$ items are treated as free parameters to be estimated from the data, constrained so that the probabilities sum to 1. Our implementation separately estimates these parameters for each set size and variance condition of the experiment. As with the binomial encoding model, the total capacity of R nats is evenly distributed among each of the encoded items.

For both of these models, let n_e indicate the number of stimulus items encoded on a given trial. The amount of encoding noise corrupting each memory representation is then

$$\sigma_e^2 = \frac{\sigma_w^2 + \sigma_s^2}{e^{2R/n_e} - 1}. \tag{31}$$

As was the case for the slots+averaging model, the response depends on whether the probed item was one of the items encoded in VWM. When the probed item was not encoded (which occurs with probability $1 - n_e/n$), the response is determined by an educated guess based solely on the probe stimulus (equation 28). Including the possibility of lapse trials, the overall response probability is

$$\begin{aligned} P(+ \,|x, \Delta) = \lambda \cdot \left(\frac{1}{2}\right) + (1 - \lambda) \cdot \sum_{n_e=0}^{n} \Bigg\{ P(n_e) \cdot \Bigg[\left(1 - \frac{n_e}{n}\right) \cdot P(+ \mid \text{probe}) + \\ \frac{n_e}{n} \cdot \left(1 - \Phi(0 \mid \mu_{\text{diff}}, \sigma_{\text{diff}}^2)\right) \Bigg] \Bigg\} \end{aligned} \tag{32}$$

The term $P(n_e)$ indicates the probability that n_e items were encoded on a given trial, and is determined by the probability distribution for the particular model. In the case of the binomial encoding model, this is determined according to a binomial distribution, whereas in the flexible encoding model, $P(n_e)$ is a free parameter in the model.

Code for the binomial encoding model is provided in Listing 6, and code for the flexible encoding model is in Listing 7.

A hierarchical version of the flexible encoding model was developed in order to separately estimate memory capacity in each condition of the experiment. This hierarchical model

assumed that the capacity estimate for each subject in each condition was drawn from a probability distribution with a condition-specific mean, and these condition-specific mean capacities were distributed according to an over-arching distribution over capacity. This was necessary because the limited data available in each condition did not sufficiently constrain the posterior distributions over capacity when parameters were estimated independently for each subject and condition. In particular, for set sizes of 1 or 2 items, performance may be dominated by sensory noise rather than capacity limitations, making it difficult to unambiguously estimate capacity in the absence of a hierarchical prior. Code for the hierarchical model is provided in Listing 8.

Details of the inference procedure

A noteworthy feature of each of the information-theoretic models is the assumption that subjects have accurate knowledge of their own sensory uncertainty, as well as the distribution of features in the task. While it is at least plausible that subjects came into the experiment with implicit knowledge of the reliability of their sensory systems, it is improbable that subjects would have an accurate internal model of the variance of visual features in our particular task environment at the outset of the experiment. Because of this fact, we eliminated the first 100 trials from each session of the experiment, and estimated model parameters using the remaining data (this left 3,184 trials per subject).

Each of the models described in the main text was implemented as a Bayesian model using the JAGS software platform (available under a free software license from http://mcmc-jags.sourceforge.net/). A prior probability distribution was defined for each variable in the model (as given in Section A). Posterior distributions were obtained through a Monte Carlo Markov Chain (MCMC) sampling procedure. See (Gilks, Richardson, & Spiegelhalter, 1998) for an introduction to Monte Carlo approaches to Bayesian inference. For each model, we ran 10 parallel chains of the sampling algorithm, and collected 10,000 samples from each chain. The first 5,000 samples from each chain were discarded as "burn-in"; this is necessary to allow the sampled values for each parameter to converge to a region with high posterior probability. The remaining 50,000 samples represent approximately independent samples from the posterior distribution for each sampled parameter. No thinning was performed on the remaining samples. For the flexible allocation model in the line length experiment, a longer Monte Carlo run was performed, consisting of 25,000 posterior samples from each chain. A longer MCMC run was used for this model as diagnostic check indicated a potential lack of convergence when using a smaller sample size (details below).

Descriptive statistics were obtained by computing the posterior mean estimate for each parameter (obtained by simple averaging of the samples), as well as the 95% highest density credible intervals (HDCI). The HDCI is a Bayesian analogue to frequentist confidence intervals, and defines a region of parameter space such that there is a 95% probability that the true parameter value is contained within that region. More formally, the 95% HDCI is defined as an interval containing 95% of the mass of the posterior probability distribution, such that no point outside of the interval has higher probability density than any point within the interval. To compute credible intervals on the basis of the Monte Carlo samples, we adopted the algorithm described in (Chen & Shao, 1999). Code written in the R programming language for computing credible intervals using this procedure is available at:
http://www.indiana.edu/ kruschke/DoingBayesianDataAnalysis/Programs/HDIofMCMC.R
(code courtesy of Dr. John Kruschke).

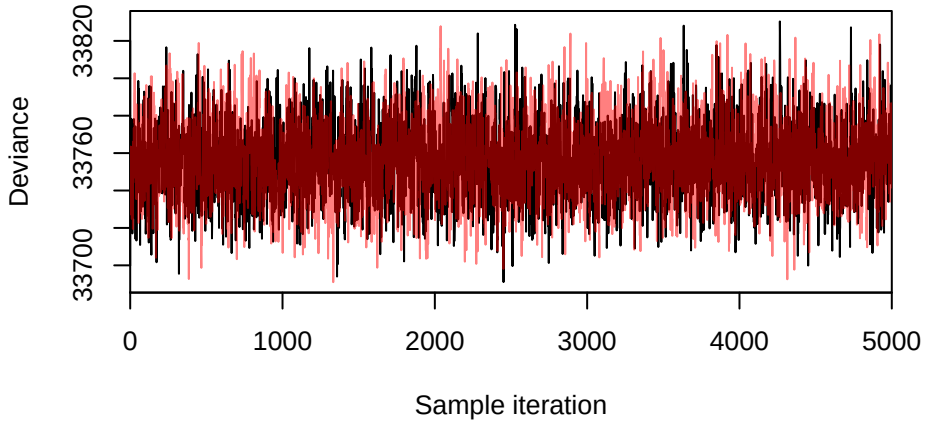

Figure 1. Plot of the deviance (data log likelihood) as a function of iteration of the MCMC sampling procedure. The samples from two chains are shown overlaid (in black and red). Data is from the binomial encoding model run on the orientation experiment.

Prior distributions were chosen for each parameter to cover a broad range of values that were deemed plausible for the task. Parameters constrained to fall within the interval (0, 1) were given beta distribution priors; parameters constrained only to be positive reals were given gamma distribution priors. Whenever the same parameter appeared in multiple models, it was given the same prior distribution in all models. In particular, the prior over the total memory capacity was set as Gamma(shape = 1, rate = 0.3). This results in a probability distribution over the positive reals with 95% of the probability mass containing the interval (0, 14.4) bits. For all models and experiments, the resulting posterior over memory capacity fell well within this interval (in the range of 3-5 bits, depending on the model). We also examined a model using a uniform prior over capacity in the range (0, 20) bits, and found no meaningful difference in the resulting posterior estimates. Similar criteria were used to construct the priors over each of the model parameters. For the slots+averaging models, we examined two different priors over the item limit. The prior for models reported in the main text assumed a discrete uniform prior over item limits in the range of 1 through 8 items. We also examined a prior which assigned 75% of the probability mass to item limits of 3–5 items, distributing the rest of the mass uniformly among other item limits between 1 and 8 items. This prior was tested as previous research has shown estimated item limits to fall within this range (Cowan, 2001; Zhang & Luck, 2008; Cowan & Rouder, 2009; Anderson, Vogel, & Awh, 2011). The choice of prior had minimal impact on the estimated item limits for each subject, and did not alter the results of the model comparisons.

By running multiple sampling chains for each model we were able to verify that the MCMC sampling procedure was converging to regions of high posterior probability. Figure 1 shows the deviance (the log-likelihood of the data) across iterations of the MCMC sampling procedure. The sampled deviance from two chains are shown overlaid. The fact that the sam-

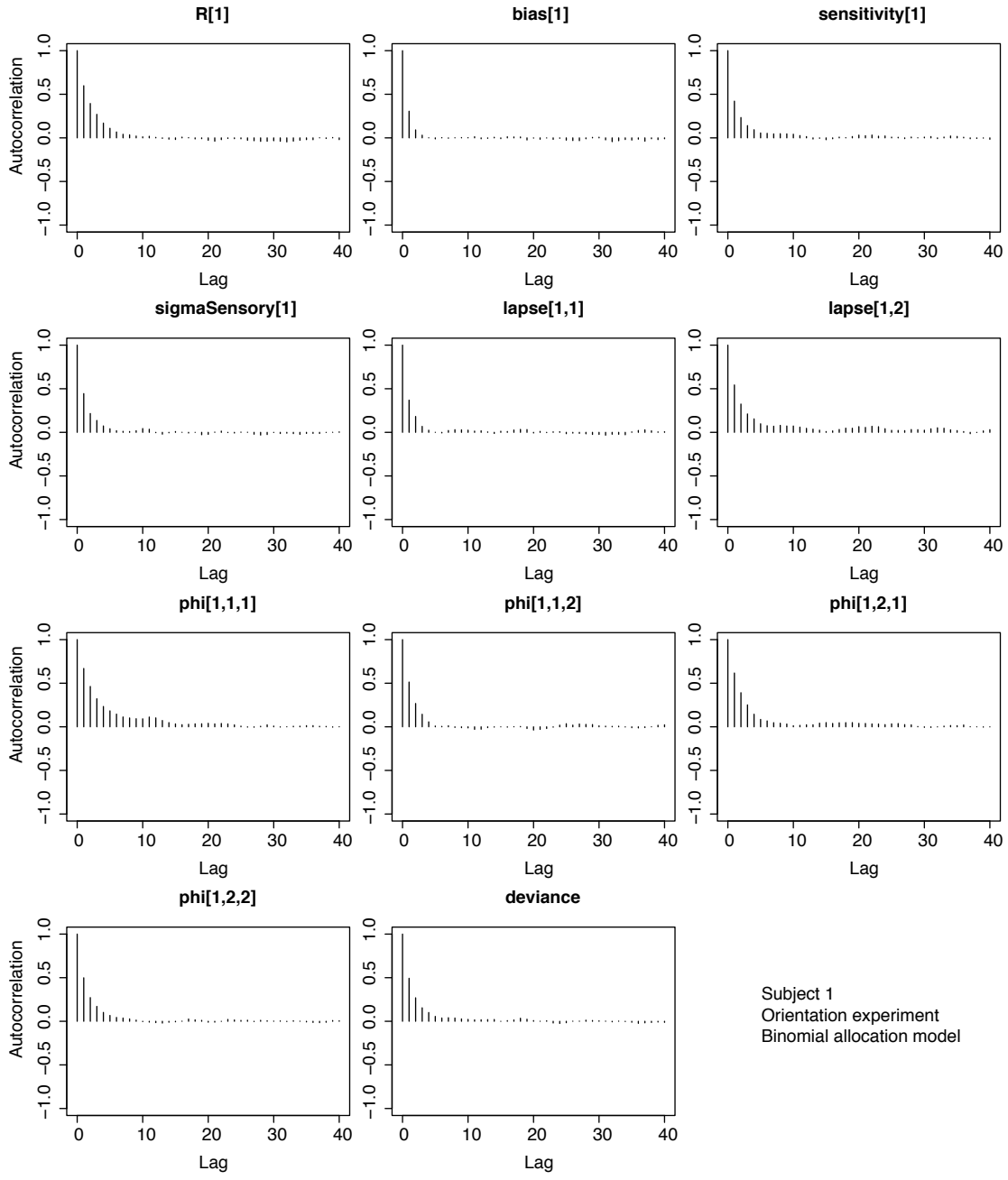

Figure 2. Plot of the autocorrelation function for each parameter of the binomial encoding model for a single subject in the orientation experiment.

ples appear stationary (their mean value is not increasing or decreasing across iterations), and do not exhibit high autocorrelation are both indicators that the MCMC procedure has converged and is sampling efficiently from the posterior distribution. Figure 2 plots the autocorrelation function for all model parameters of the binomial encoding model. Data are shown for a single subject in the orientation experiment. Refer to Listing 6 for the definition of each variable. The results indicate the absence of high autocorrelations among samples.

Figure 3 shows a bivariate correlation matrix plot among four different variables in the binomial encoding model (data is shown for one subject in the orientation experiment). Each plot marker corresponds to a sample from the posterior distribution over parameters obtained via the MCMC sampling procedure. Since the binomial and flexible encoding models include numerous parameters, it is important to verify that the parameters are identifiable, or sufficiently constrained from the available data. The four parameters shown are key parameters from the model: the overall capacity, the magnitude of sensory noise, the probability matching exponent, and the encoding probability in the largest set size, low variance condition. Although correlations are visible among different parameters, it is not the case that any of the parameters are highly correlated with other model parameters. Further, the analyses and plots reported in the main text report 95% highest density credible intervals around means, as a result, comparisons among models and conditions take the inherent uncertainty in parameters into account. Correlations were generally higher for the flexible encoding model, since this model included significantly more parameters than the other models (as it estimated the probability of encoding $0 \ldots n$ items separately in each set size and variance condition of the experiment).

As a further check on the performance of the MCMC sampling procedure, we computed the potential scale reduction factor (PSRF, Gelman & Rubin, 1992) for the 10 parallel chains from each model. This statistic compares the variance within each chain to the estimated variance from all chains. If the estimated overall variance is significantly greater than the variance observed in single chains, it suggests that the chains have not all converged and additional samples should be collected. We computed the multivariate version of this statistic (Brooks & Gelman, 1998) for each model and experiment. Values much above 1 ($> \sim 1.2$) indicate lack of convergence. Table 1 reports the PSRF statistic for each model. The flexible encoding model in the line length experiment showed a PSRF above 1.2 when using the same sampling procedure as the remaining models. The results reported in Table 1 are therefore based on a larger sample size for this model (using 25,000 posterior samples per chain), and indicate no evidence for lack of convergence.

References

Anderson, D. E., Vogel, E. K., & Awh, E. (2011). Precision in visual working memory reaches a stable plateau when individual item limits are exceeded. *The Journal of Neuroscience*, *31*(3), 1128–1138.

Bays, P. M., & Husain, M. (2008). Dynamic shifts of limited working memory resources in human vision. *Science*, *321*(5890), 851.

Berger, T. (1971). *Rate distortion theory: A mathematical basis for data compression*. Englewood Cliffs, NJ: Prentice-Hall.

Brooks, S. P., & Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, *7*(4), 434–455.

Chen, M. H., & Shao, Q. M. (1999). Monte Carlo estimation of Bayesian credible and hpd intervals. *Journal of Computational and Graphical Statistics*, *8*, 69–92.

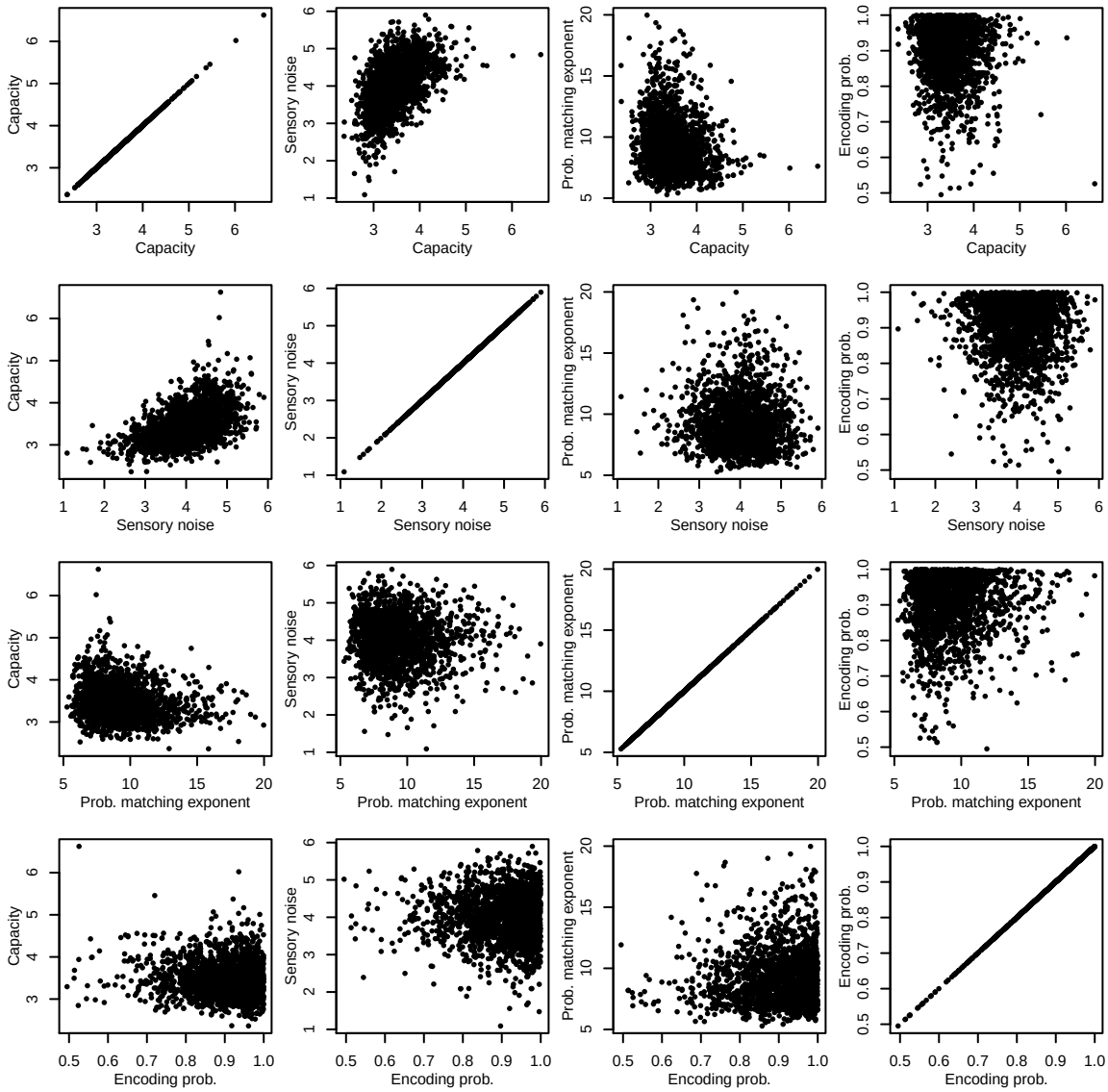

Figure 3. Plot of bivariate correlations among four main parameters of the binomial encoding model. Data is from one representative subject in the orientation experiment.

Table 1
Multivariate potential scale reduction factor (PSRF) computed for each model and in each experiment.

Experiment and model	Multivariate PSRF
Orientation	
Information-theoretic continuous resource	1.01
Non-information-theoretic continuous resource	1.13
Information-theoretic slots+averaging	1.01
Non-information-theoretic slots+averaging	1.15
Binomial encoding	1.03
Flexible encoding	1.14
Line length	
Information-theoretic continuous resource	1.01
Non-information-theoretic continuous resource	1.09
Information-theoretic slots+averaging	1.06
Non-information-theoretic slots+averaging	1.05
Binomial encoding	1.03
Flexible encoding	1.19

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, *24*(01), 87–114.

Cowan, N., & Rouder, J. N. (2009). Comment on "Dynamic shifts of limited working memory resources in human vision". *Science*, *323*(13), 877c.

Gelman, A., & Rubin, D. B. (1992). *Statistical Science*, *7*(4), 457–511.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1998). *Markov chain Monte Carlo in practice*. London: Chapman & Hall.

Wilken, P., & Ma, W. J. (2004). A detection theory account of change detection. *Journal of Vision*, *4*, 1120–1135.

Zhang, W., & Luck, S. J. (2008). Discrete fixed-resolution representations in visual working memory. *Nature*, *453*(7192), 233-5.

Appendix
Code listings

This section contains code listings for the VWM models as well as the psychometric function fit to empirical data. All code is written in the JAGS model description language. Code for the following models is provided:

1. Gaussian + random response psychometric function
2. Information-theoretic continuous resource model
3. Non-information-theoretic continuous resource model
4. Information-theoretic slots+averaging model
5. Non-information-theoretic slots+averaging model
6. Information-theoretic binomial encoding model
7. Information-theoretic flexible encoding model
8. Hierarchical (multiple capacity) flexible encoding model

Listing 1: JAGS model code for the Gaussian psychometric function.

```
#*********************************************************************
# Data and inputs:
#
# delta[s,v,n,t] - An indicator variable that refers to the
#   perturbation level for each subject on each trial of each
#   condition
#
# deltas[j] - A vector of the perturbation levels used
#
# respondClockwise[s,v,n,t] - A binary variable indicating whether
#   subject s reported a clockwise (positive) perturbation on trial
#   t of variance condition v and set size condition n.

model {

  # Population-level parameters ***************************************
  for(v in 1:nVariances) { # For each variance condition
    for(n in 1:nSetSizes) { # For each set size condition

      # Lapse rate ****************************************************
      # Each subject's lapse rate is drawn from a population-level
      # beta distribution with parameters (alpha, beta).
      # Priors parameterized in terms of the mean and concentration of
      # the beta distribution (concentration = alpha + beta).
      lapseMuCond[v,n] ~ dbeta(1, 1);     # Mean parameter
      lapseKCond[v,n] ~ dgamma(1, 0.001); # Concentration parameter

      # Convert to alpha & beta
      lapseAlphaCond[v,n] <- lapseMuCond[v,n]*(2 + lapseKCond[v,n]);
      lapseBetaCond[v,n] <- (1 - lapseMuCond[v,n])*
        (2 + lapseKCond[v,n]);

      # Gaussian CDF bias *********************************************
      # The population-level bias parameter of the Gaussian CDF.
      # The bias is given a Gaussian distribution with parameters
      # (mean, precision), parameterized in terms of mean
      # and standard deviation.
      biasMeanCond[v,n] ~ dnorm(0, 0.1);
      biasSDCond[v,n] ~ dgamma(1, 1);

      # Convert to precision (1 / variance)
      biasPrecCond[v,n] <- 1 / (biasSDCond[v,n]^2);
```

```
      # Gaussian CDF width ***********************************************
      # The width of the Gaussian CDF, defined as the stimulus distance
      # between the 25% and 75% quantile
      widthMeanCond[v,n] ~ dgamma(1, 0.05);
      widthSDCond[v,n] ~ dgamma(1, 0.1);

      # Convert to shape & rate parameters
      widthAlphaCond[v,n] <- widthMeanCond[v,n]^2 / widthSDCond[v,n]^2;
      widthBetaCond[v,n] <- widthMeanCond[v,n] / widthSDCond[v,n]^2;

    }
  }

  # Subject-specific parameters ***************************************
  for(s in 1:nSubjects) { # For each subject
    for(v in 1:nVariances) { # For each variance condition
      for(n in 1:nSetSizes) { # For each set size condition

        # Sample each parameter from the population-level distribution

        lapse[s,v,n] ~ dbeta(lapseAlphaCond[v,n], lapseBetaCond[v,n]);

        bias[s,v,n] ~ dnorm(biasMeanCond[v,n], biasPrecCond[v,n]);

        width[s,v,n] ~ dgamma(widthAlphaCond[v,n], widthBetaCond[v,n]);
        tau[s,v,n] <- (z / width[s,v,n])^2; # Convert to precision
        # z is a constant relating psychometric width and standard
        # deviation: z = qnorm(1 - 0.25, mean = 0, sd = 1) -
        #      qnorm(0.25, mean = 0, sd = 1);
      }
    }
  }

  # Response probability **********************************************
  # Can pre-compute the response probability for each condition
  for(s in 1:nSubjects) { # For each subject
    for(v in 1:nVariances) { # For each variance condition
      for(n in 1:nSetSizes) { # For each set size
        for(d in 1:nDeltas) { # For each perturbation level

          # theta indicates the probability of reporting a positive
          # perturbation.
          theta[s,v,n,d] <- lapse[s,v,n]*(1/2) +
            (1 - lapse[s,v,n])*
            pnorm(deltas[d], bias[s,v,n], tau[s,v,n]);
        }
      }
    }
  }

  # Observed data *****************************************************
  # Each response is drawn from a Bernoulli distribution with success
  # parameter theta, computed above.
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      for(n in 1:nSetSizes) {
        for(t in 1:trials[s,v,n]) {
          respondClockwise[s,v,n,t] ~ dbern(
            theta[s,v,n,delta[s,v,n,t]]);
        }
      }
    }
  }
}
```

Listing 2: JAGS model code for the information-theoretic continuous resource model.

```
#*********************************************************************
# Data and inputs:
#
# x[s,v,n,t] - The feature value of the probed item (before the
#   perturbation is applied) for subject s, variance condition v,
#   set size n, trial t
#
# delta[s,v,n,t] - An indicator variable that refers to the
#   perturbation level for each subject on each trial of each
#   condition
#
# deltas[j] - A vector of the perturbation levels used
#
# respondClockwise[s,v,n,t] - A binary variable indicating whether
#   subject s reported a clockwise (positive) perturbation on trial
#   t of variance condition v and set size condition n.

model {

  # Response bias ******************************************************
  for(s in 1:nSubjects) {
    bias[s] ~ dnorm(0, 0.1);
  }

  # Lapse rate *********************************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      lapse[s,v] ~ dbeta(1, 10);
    }
  }

  # Sensory noise ******************************************************
  for(s in 1:nSubjects) {
    sigmaSensory[s] ~ dgamma(1, 0.1);
  }

  # Capacity (in nats) *************************************************
  for(s in 1:nSubjects) {
    R[s] ~ dgamma(1, 0.3);
  }

  # Encoding noise *****************************************************
  # Can pre-compute the encoding noise as a function of stimulus
  # conditions.
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      for(n in 1:nSetSizes) {

        # Encoding noise
        sigmaEnc[s,v,n] <- sqrt(sigmas[v]^2 + sigmaSensory[s]^2) /
          sqrt(-1 + exp(2*R[s]/setSizes[n]));

        # Mean of the Gaussian distribution for encoded items
        meanConstant[s,v,n] <- sigmas[v]^2 /
          (sigmaEnc[s,v,n]^2 + sigmas[v]^2 + sigmaSensory[s]^2);

        # Variance of the Gaussian distribution for encoded items
        variance[s,v,n] <- sigmaSensory[s]^2 +
          (sigmas[v]^4*(sigmaEnc[s,v,n]^2 + sigmaSensory[s]^2))/
            (sigmaEnc[s,v,n]^2 + sigmas[v]^2 + sigmaSensory[s]^2)^2;

        precision[s,v,n] <- 1 / variance[s,v,n];
      }
```

```
    }
  }

  # Observed data ******************************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      for(n in 1:nSetSizes) {
        for(t in 1:trials[s,v,n]) {

          # Response probability for a non-lapse response *************
          mean[s,v,n,t] <- deltas[delta[s,v,n,t]] +
            x[s,v,n,t]*(1 - meanConstant[s,v,n]);

          theta[s,v,n,t] <- (1 - pnorm(0, bias[s] + mean[s,v,n,t],
            precision[s,v,n]));

          # Response probability, including lapse rate ****************
          thetaMarginal[s,v,n,t] <- lapse[s,v]*(1/2) +
            (1 - lapse[s,v])*theta[s,v,n,t];

          # Observed response *****************************************
          respondClockwise[s,v,n,t] ~ dbern(thetaMarginal[s,v,n,t]);
        }
      }
    }
  }
}
```

Listing 3: JAGS model code for the non-information-theoretic continuous resource model.

```
#*********************************************************************
# Data and inputs:
#
# x[s,v,n,t] - The feature value of the probed item (before the
#   perturbation is applied) for subject s, variance condition v,
#   set size n, trial t
#
# delta[s,v,n,t] - An indicator variable that refers to the
#   perturbation level for each subject on each trial of each
#   condition
#
# deltas[j] - A vector of the perturbation levels used
#
# respondClockwise[s,v,n,t] - A binary variable indicating whether
#   subject s reported a clockwise (positive) perturbation on trial
#   t of variance condition v and set size condition n.

model {

  # Response bias ******************************************************
  for(s in 1:nSubjects) {
    bias[s] ~ dnorm(0, 0.1);
  }

  # Lapse rate *********************************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      lapse[s,v] ~ dbeta(1, 10);
    }
  }
```

```
  # Sensory noise ******************************************************
  for(s in 1:nSubjects) {
    sigmaSensory[s] ~ dgamma(1, 0.1);
  }

  # Single item encoding noise *****************************************
  # The encoding noise for a single item; the encoding noise for
  # multiple items is a power-law function of the set size
  for(s in 1:nSubjects) {
    sigmaEncSingleItem[s] ~ dgamma(1, 0.1);
  }

  # Power law exponent *************************************************
  # Exponent of the power-law relation betwee encoding noise and
  # set size.
  for(s in 1:nSubjects) {
    z[s] ~ dgamma(1, 1);
  }

  # Encoding noise *****************************************************
  # Can pre-compute the encoding noise as a function of the set size
  for(s in 1:nSubjects) {
    for(n in 1:nSetSizes) {
      sigmaEnc[s,n] <- sqrt(sigmaSensory[s]^2 +
        ((setSizes[n]^z[s])*sigmaEncSingleItem[s] +
        sigmaSensory[s]^2)^2);
      precision[s,n] <- 1 / sigmaEnc[s,n]^2;
    }
  }

  # Observed data ******************************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      for(n in 1:nSetSizes) {
        for(t in 1:trials[s,v,n]) {

          # Response probability for a non-lapse response *************
          theta[s,v,n,t] <- (1 - pnorm(0, bias[s] +
            deltas[delta[s,v,n,t]], precision[s,n]));

          # Response probability, including lapse rate ****************
          thetaMarginal[s,v,n,t] <- lapse[s,v]*(1/2) +
            (1 - lapse[s,v])*theta[s,v,n,t];

          # Observed response *****************************************
          respondClockwise[s,v,n,t] ~ dbern(thetaMarginal[s,v,n,t]);
        }
      }
    }
  }
}
```

Listing 4: JAGS model code for the information-theoretic slots+averaging model.

```
#*********************************************************************
# Data and inputs:
#
# x[s,v,n,t] - The feature value of the probed item (before the
#   perturbation is applied) for subject s, variance condition v,
```

```
#   set size n, trial t
#
# delta[s,v,n,t] - An indicator variable that refers to the
#   perturbation level for each subject, and on each trial of each
#   condition
#
# deltas[j] - A vector of the perturbation levels
#
# kPrior - A vector defining the prior probability of item limits in
#   the range of 1 - 8 items. In current implementation, a uniform
#   prior is used.
#
# pi[j,k,n] - A lookup table that stores the probability of j
#   slots being assigned to the probe item, as a function
#   of slot limit k and the set size n. The equation for this
#   is given in the main text.
#
# probe[s,v,n,t] - The probe stimulus (x + \Delta)
#   for subject s in variance condition v, set size condition n
#   and trial t.
#
# respondClockwise[s,v,n,t] - A binary variable indicating whether
#   subject s reported a clockwise (positive) perturbation on trial
#   t of variance condition v and set size condition n.

model {

  # Response bias ******************************************************
  for(s in 1:nSubjects) {
    bias[s] ~ dnorm(0, 0.1);
  }

  # Lapse rate *********************************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      lapse[s,v] ~ dbeta(1, 10);
    }
  }

  # Item limit *********************************************************
  for(s in 1:nSubjects) {
    k[s] ~ dcat(kPrior);
  }

  # Probability matching exponent **************************************
  for(s in 1:nSubjects) {
    sensitivity[s] ~ dgamma(1, 0.3);
  }

  # Sensory noise ******************************************************
  for(s in 1:nSubjects) {
    sigmaSensory[s] ~ dgamma(1, 0.1);
  }

  # Capacity (in nats) *************************************************
  for(s in 1:nSubjects) {
    R[s] ~ dgamma(1, 0.3);
  }

  # Response distribution **********************************************
  # Mean and variance of the response distribution governing the
  # probability of responding with a positive perturbation, as a
  # function of the variance and number of slots allocated to the probe
  # item.
```

```
for(s in 1:nSubjects) {
  for(v in 1:nVariances) {
    for(na in 1:8) {

      sigmaEnc[s,v,na] <- sqrt(sigmas[v]^2 + sigmaSensory[s]^2) /
        sqrt(-1 + exp(2*(na*R[s]/k[s])));

      meanConstant[s,v,na] <- sigmas[v]^2 /
        (sigmaEnc[s,v,na]^2 + sigmas[v]^2 + sigmaSensory[s]^2);

      variance[s,v,na] <- sigmaSensory[s]^2 +
        (sigmas[v]^4*(sigmaEnc[s,v,na]^2 + sigmaSensory[s]^2))/
          (sigmaEnc[s,v,na]^2 + sigmas[v]^2 + sigmaSensory[s]^2)^2;

      precision[s,v,na] <- 1 / variance[s,v,na];
    }
  }
}

# Observed data ********************************************************
for(s in 1:nSubjects) {
  for(v in 1:nVariances) {

    # Precision of the sensory encoding of the probe stimulus
    likPrec[s,v] <- 1 / (sigmas[v]^2 + sigmaSensory[s]^2);

    for(n in 1:nSetSizes) {
      for(t in 1:trials[s,v,n]) {

        #*************************************************************
        # Response probability when the probe item was not encoded

        likClockwise[s,v,n,t] <- (1/4)*
          (dnorm(probe[s,v,n,t], bias[s] + abs(deltas[1]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] + abs(deltas[2]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] + abs(deltas[3]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] + abs(deltas[4]),
            likPrec[s,v]));

        likCounterClockwise[s,v,n,t] <- (1/4)*
          (dnorm(probe[s,v,n,t], bias[s] - abs(deltas[1]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] - abs(deltas[2]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] - abs(deltas[3]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] - abs(deltas[4]),
            likPrec[s,v]));

        probClockwise[s,v,n,t] <- likClockwise[s,v,n,t] /
          (likClockwise[s,v,n,t] + likCounterClockwise[s,v,n,t]);

        thetaGuess[s,v,n,t] <-
          (probClockwise[s,v,n,t]^sensitivity[s]) /
          (probClockwise[s,v,n,t]^sensitivity[s] +
          (1 - probClockwise[s,v,n,t])^sensitivity[s]);

        # Response probability when zero slots were assigned
        # to the probe item
        theta[s,v,n,t,1] <- thetaGuess[s,v,n,t];
```

```
          #*************************************************************
          # Response probability for each possible number of items
          # slots assigned to the probe item, na, assuming na > 0
          for(na in 1:8) {
            mean[s,v,n,t,na] <- deltas[delta[s,v,n,t]] +
              x[s,v,n,t]*(1 - meanConstant[s,v,na]);
            theta[s,v,n,t,na+1] <- (1 - pnorm(0, bias[s] +
               mean[s,v,n,t,na], precision[s,v,na]));
          }

          # Marginalize over the number of slots assigned,
          # and the probability of a lapse trial.
          thetaMarginal[s,v,n,t] <- lapse[s,v]*(1/2) +
            (1 - lapse[s,v])*sum(theta[s,v,n,t,1:9]*
            pi[1:9, k[s], setSizes[n]]);

          # Observed response
          respondClockwise[s,v,n,t] ~ dbern(thetaMarginal[s,v,n,t]);
        }
      }
    }
  }
}
```

Listing 5: JAGS model code for the non-information-theoretic slots+averaging model.

```
#*********************************************************************
# Data and inputs:
#
# x[s,v,n,t] - The feature value of the probed item (before the
#   perturbation is applied) for subject s, variance condition v,
#   set size n, trial t
#
# delta[s,v,n,t] - An indicator variable that refers to the
#   perturbation level for each subject, and on each trial of each
#   condition
#
# deltas[j] - A vector of the perturbation levels
#
# kPrior - A vector defining the prior probability of item limits in
#   the range of 1 - 8 items. In current implementation, a uniform
#   prior is used.
#
# pi[j,k,n] - A lookup table that stores the probability of j
#   slots being assigned to the probe item, as a function
#   of slot limit k and the set size n
#
# probe[s,v,n,t] - The probe stimulus (x + \Delta)
#   for subject s in variance condition v, set size condition n
#   and trial t.
#
# respondClockwise[s,v,n,t] - A binary variable indicating whether
#   subject s reported a clockwise (positive) perturbation on trial
#   t of variance condition v and set size condition n.

model {

  # Response bias ******************************************************
  for(s in 1:nSubjects) {
    bias[s] ~ dnorm(0, 0.1);
```

```
}

# Lapse rate ***********************************************************
for(s in 1:nSubjects) {
  for(v in 1:nVariances) {
    lapse[s,v] ~ dbeta(1, 10);
  }
}

# Encoding probabilities ***********************************************
for(s in 1:nSubjects) {
  k[s] ~ dcat(kPrior);
}

# Probability matching exponent ****************************************
for(s in 1:nSubjects) {
  sensitivity[s] ~ dgamma(1, 0.3);
}

# Sensory noise ********************************************************
for(s in 1:nSubjects) {
  sigmaSensory[s] ~ dgamma(1, 0.1);
}

# Encoding noise *******************************************************
for(s in 1:nSubjects) {
  sigmaEncoding[s] ~ dgamma(1, 0.1);
}

# Response distribution ************************************************
# Mean and variance of the response distribution governing the
# probability of responding with a positive perturbation, as a
# function of the number of slots allocated to the probe item.
for(s in 1:nSubjects) {
  for(na in 1:8) {

      sigmaEnc[s,na] <- sqrt(sigmaSensory[s]^2 +
        sigmaEncoding[s]^2)/sqrt(na);

      variance[s,na] <- sigmaSensory[s]^2 + sigmaEnc[s,na]^2;

      precision[s,na] <- 1 / variance[s,na];
   }
}

# Observed data ********************************************************
for(s in 1:nSubjects) {
  for(v in 1:nVariances) {

    # Precision of the sensory encoding of the probe stimulus
    likPrec[s,v] <- 1 / (sigmas[v]^2 + sigmaSensory[s]^2);

    for(n in 1:nSetSizes) {
      for(t in 1:trials[s,v,n]) {

        #*******************************************************
        # Response probability when the probe item was not encoded

        likClockwise[s,v,n,t] <- (1/4)*
          (dnorm(obs[s,v,n,t], bias[s] + abs(deltas[1]),
            likPrec[s,v]) +
          dnorm(obs[s,v,n,t], bias[s] + abs(deltas[2]),
            likPrec[s,v]) +
          dnorm(obs[s,v,n,t], bias[s] + abs(deltas[3]),
```

```
              likPrec[s,v]) +
            dnorm(obs[s,v,n,t], bias[s] + abs(deltas[4]),
              likPrec[s,v]));

          likCounterClockwise[s,v,n,t] <- (1/4)*
          (dnorm(obs[s,v,n,t], bias[s] - abs(deltas[1]),
            likPrec[s,v]) +
           dnorm(obs[s,v,n,t], bias[s] - abs(deltas[2]),
            likPrec[s,v]) +
           dnorm(obs[s,v,n,t], bias[s] - abs(deltas[3]),
            likPrec[s,v]) +
           dnorm(obs[s,v,n,t], bias[s] - abs(deltas[4]),
            likPrec[s,v]));

          probClockwise[s,v,n,t] <- likClockwise[s,v,n,t] /
            (likClockwise[s,v,n,t] + likCounterClockwise[s,v,n,t]);

          thetaGuess[s,v,n,t] <-
            (probClockwise[s,v,n,t]^sensitivity[s]) /
            (probClockwise[s,v,n,t]^sensitivity[s] +
            (1 - probClockwise[s,v,n,t])^sensitivity[s]);

          # Response probability when zero slots were assigned
          # to the probe item
          theta[s,v,n,t,1] <- thetaGuess[s,v,n,t];

          #************************************************************
          # Response probability for each possible number of items
          # slots assigned to the probe item, na, assuming na > 0
          for(na in 1:8) {
            theta[s,v,n,t,na+1] <- (1 - pnorm(0, bias[s] +
              deltas[delta[s,v,n,t]], precision[s,na]));
          }

          thetaMarginal[s,v,n,t] <- lapse[s,v]*(1/2) +
            (1 - lapse[s,v])*sum(theta[s,v,n,t,1:9]*
            pi[1:9, k[s], setSizes[n]]);

          # Observed response
          respondClockwise[s,v,n,t] ~ dbern(thetaMarginal[s,v,n,t]);
        }
      }
    }
  }
}
```

Listing 6: JAGS model code for the information-theoretic binomial encoding model.

```
#*********************************************************************
# Data and inputs:
#
# x[s,v,n,t] - The feature value of the probed item (before the
#   perturbation is applied) for subject s, variance condition v,
#   set size n, trial t
#
# delta[s,v,n,t] - An indicator variable that refers to the
#   perturbation level for each subject, and on each trial of each
#   condition
#
# deltas[j] - A vector of the perturbation levels
```

```
#
# probe[s,v,n,t] - Stores the probe stimulus (x + \Delta)
#   for subject s in variance condition v, set size condition n
#   and trial t.
#
# respondClockwise[s,v,n,t] - A binary variable indicating whether
#   subject s reported a clockwise (positive) perturbation on trial
#   t of variance condition v and set size condition n.

model {

  # Response bias ******************************************************
  for(s in 1:nSubjects) {
    bias[s] ~ dnorm(0, 0.1);
  }

  # Lapse rate *********************************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      lapse[s,v] ~ dbeta(1, 10);
    }
  }

  # Encoding probability ***********************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      for(n in 1:nSetSizes) {
        phi[s,v,n] ~ dbeta(1, 1);

        for(ne in 0:setSizes[n]) {
          # Compute the probability of encoding 0 ... n items under the
          # given binomial distribution.
          pi[s,v,n,ne+1] <- dbin(ne, phi[s,v,n], setSizes[n]);
        }
      }
    }
  }

  # Probability matching exponent **************************************
  for(s in 1:nSubjects) {
    sensitivity[s] ~ dgamma(1, 0.5);
  }

  # Sensory noise ******************************************************
  for(s in 1:nSubjects) {
    sigmaSensory[s] ~ dgamma(1, 0.1);
  }

  # Capacity (in nats) *************************************************
  for(s in 1:nSubjects) {
    R[s] ~ dgamma(1, 0.3);
  }

  # Encoding noise *****************************************************
  # Can pre-compute the encoding noise as a function of the number
  # of items encode (1 ... 8) and trial conditions.
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      for(ne in 1:8) {

        sigmaEnc[s,v,ne] <- sqrt(sigmas[v]^2 + sigmaSensory[s]^2) /
          sqrt(-1 + exp(2*R[s]/ne));

        meanConstant[s,v,ne] <- sigmas[v]^2 /
```

```
        (sigmaEnc[s,v,ne]^2 + sigmas[v]^2 + sigmaSensory[s]^2);

      variance[s,v,ne] <- sigmaSensory[s]^2 +
        (sigmas[v]^4*(sigmaEnc[s,v,ne]^2 + sigmaSensory[s]^2))/
          (sigmaEnc[s,v,ne]^2 + sigmas[v]^2 + sigmaSensory[s]^2)^2;
      precision[s,v,ne] <- 1 / variance[s,v,ne];
    }
  }
}

# Observed data ******************************************************
for(s in 1:nSubjects) {
  for(v in 1:nVariances) {

    # Precision of the sensory encoding of the probe stimulus
    likPrec[s,v] <- 1 / (sigmas[v]^2 + sigmaSensory[s]^2);

    for(n in 1:nSetSizes) {
      for(t in 1:trials[s,v,n]) {

        #************************************************************
        # Compute the response probability for trials where the
        # probe item was not encoded
        likClockwise[s,v,n,t] <- (1/4)*
          (dnorm(probe[s,v,n,t], bias[s] +  abs(deltas[1]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] +  abs(deltas[2]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] +  abs(deltas[3]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] + abs(deltas[4]),
            likPrec[s,v]));

        likCounterClockwise[s,v,n,t] <- (1/4)*
          (dnorm(probe[s,v,n,t], bias[s] - abs(deltas[1]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] - abs(deltas[2]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] - abs(deltas[3]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] - abs(deltas[4]),
            likPrec[s,v]));

        probClockwise[s,v,n,t] <- likClockwise[s,v,n,t] /
          (likClockwise[s,v,n,t] + likCounterClockwise[s,v,n,t]);

        thetaGuess[s,v,n,t] <- (probClockwise[s,v,n,t]^
          sensitivity[s]) /
          (probClockwise[s,v,n,t]^sensitivity[s] +
          (1 - probClockwise[s,v,n,t])^sensitivity[s]);

        theta[s,v,n,t,1] <- thetaGuess[s,v,n,t];

        #************************************************************
        # Response probability for each possible number of items
        # encoded, ne, assuming ne > 0
        for(ne in 1:setSizes[n]) {

          mean[s,v,n,t,ne] <- deltas[delta[s,v,n,t]] +
            x[s,v,n,t]*(1 - meanConstant[s,v,ne]);

          theta[s,v,n,t,ne + 1] <- (1 - ne / setSizes[n])*
            (thetaGuess[s,v,n,t]) +
            (ne / setSizes[n])*
```

```
            (1 - pnorm(0, bias[s] + mean[s,v,n,t,ne],
              precision[s,v,ne]));
        }

        # Marginalize over the number of items encoded
        thetaMarginal[s,v,n,t] <- lapse[s,v]*(1/2) +
          (1 - lapse[s,v])*sum(theta[s,v,n,t,1:(setSizes[n] + 1)]*
            pi[s,v,n,1:(setSizes[n] + 1)]);

        # Observed response
        respondClockwise[s,v,n,t] ~ dbern(thetaMarginal[s,v,n,t]);
      }
    }
  }
}
```

Listing 7: JAGS model code for the information-theoretic flexible encoding model.

```
#*********************************************************************
# Data and inputs:
#
# x[s,v,n,t] - The feature value of the probed item (before the
#   perturbation is applied) for subject s, variance condition v,
#   set size n, trial t
#
# delta[s,v,n,t] - An indicator variable that refers to the
#   perturbation level for each subject, and on each trial of each
#   condition
#
# deltas[j] - A vector of the perturbation levels
#
# probe[s,v,n,t] - Stores the probe stimulus (x + \Delta)
#   for subject s in variance condition v, set size condition n
#   and trial t.
#
# respondClockwise[s,v,n,t] - A binary variable indicating whether
#   subject s reported a clockwise (positive) perturbation on trial
#   t of variance condition v and set size condition n.

model {

  # Response bias ******************************************************
  for(s in 1:nSubjects) {
    bias[s] ~ dnorm(0, 0.1);
  }

  # Lapse rate *********************************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      lapse[s,v] ~ dbeta(1, 10);
    }
  }

  # Encoding probabilities *********************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      for(n in 1:nSetSizes) {
        pi[s,v,n,1:9] ~ ddirch(alpha[n,1:9]);
      }
```

```
  }
}

# Probability matching exponent ************************************
for(s in 1:nSubjects) {
  sensitivity[s] ~ dgamma(1, 0.3);
}

# Sensory noise ****************************************************
for(s in 1:nSubjects) {
  sigmaSensory[s] ~ dgamma(1, 0.1);
}

# Capacity (in nats) ***********************************************
for(s in 1:nSubjects) {
  R[s] ~ dgamma(1, 0.3);
}

# Response distribution ********************************************
# Mean and variance of the response distribution governing the
# probability of responding with a positive perturbation, as a
# function of the variance and number of items encoded. Derivation is
# provided in the supplementary material.
for(s in 1:nSubjects) {
  for(v in 1:nVariances) {
    for(ne in 1:8) { # ne = Number encoded

      sigmaEnc[s,v,ne] <- sqrt(sigmas[v]^2 + sigmaSensory[s]^2) /
        sqrt(-1 + exp(2*R[s]/ne));

      meanConstant[s,v,ne] <- sigmas[v]^2 /
        (sigmaEnc[s,v,ne]^2 + sigmas[v]^2 + sigmaSensory[s]^2);

      variance[s,v,ne] <- sigmaSensory[s]^2 +
        (sigmas[v]^4*(sigmaEnc[s,v,ne]^2 + sigmaSensory[s]^2))/
          (sigmaEnc[s,v,ne]^2 + sigmas[v]^2 + sigmaSensory[s]^2)^2;

      precision[s,v,ne] <- 1 / variance[s,v,ne];
    }
  }
}

# Observed data ****************************************************
for(s in 1:nSubjects) {
  for(v in 1:nVariances) {

    # Precision of the sensory encoding of the probe stimulus
    likPrec[s,v] <- 1 / (sigmas[v]^2 + sigmaSensory[s]^2);

    for(n in 1:nSetSizes) {
      for(t in 1:trials[s,v,n]) {

        #************************************************************
        # Response probability when the probe item was not encoded

        likClockwise[s,v,n,t] <- (1/4)*
          (dnorm(probe[s,v,n,t], bias[s] + abs(deltas[1]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] + abs(deltas[2]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] + abs(deltas[3]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] + abs(deltas[4]),
            likPrec[s,v]));
```

```
          likCounterClockwise[s,v,n,t] <- (1/4)*
            (dnorm(probe[s,v,n,t], bias[s] - abs(deltas[1]),
              likPrec[s,v]) +
            dnorm(probe[s,v,n,t], bias[s] - abs(deltas[2]),
              likPrec[s,v]) +
            dnorm(probe[s,v,n,t], bias[s] - abs(deltas[3]),
              likPrec[s,v]) +
            dnorm(probe[s,v,n,t], bias[s] - abs(deltas[4]),
              likPrec[s,v]));

          probClockwise[s,v,n,t] <- likClockwise[s,v,n,t] /
            (likClockwise[s,v,n,t] + likCounterClockwise[s,v,n,t]);

          thetaGuess[s,v,n,t] <-
            (probClockwise[s,v,n,t]^sensitivity[s]) /
            (probClockwise[s,v,n,t]^sensitivity[s] +
            (1 - probClockwise[s,v,n,t])^sensitivity[s]);

          theta[s,v,n,t,1] <- thetaGuess[s,v,n,t];

          #************************************************************
          # Response probability for each possible number of items
          # encoded, ne, assuming ne > 0
          for(ne in 1:setSizes[n]) {

            mean[s,v,n,t,ne] <- deltas[delta[s,v,n,t]] +
              x[s,v,n,t]*(1 - meanConstant[s,v,ne]);

            theta[s,v,n,t,ne + 1] <- (1 - ne / setSizes[n])*
              (thetaGuess[s,v,n,t]) +
              (ne / setSizes[n])*
                (1 - pnorm(0, bias[s] + mean[s,v,n,t,ne],
                precision[s,v,ne]));
          }

          # Marginalize over the number of items encoded,
          # and the probability of a lapse trial.
          thetaMarginal[s,v,n,t] <- lapse[s,v]*(1/2) +
            (1 - lapse[s,v])*sum(theta[s,v,n,t,1:(setSizes[n] + 1)]*
            pi[s,v,n,1:(setSizes[n] + 1)]);

          # Observed response
          respondClockwise[s,v,n,t] ~ dbern(thetaMarginal[s,v,n,t]);
        }
      }
    }
  }
}
```

Listing 8: JAGS model code for the hierarchical, multiple capacity flexible encoding model.

```
#*********************************************************************
# Data and inputs:
#
# x[s,v,n,t] - The feature value of the probed item (before the
#   perturbation is applied) for subject s, variance condition v,
#   set size n, trial t
#
# delta[s,v,n,t] - An indicator variable that refers to the
```

```
#   perturbation level for each subject, and on each trial of each
#   condition
#
# deltas[j] - A vector of the perturbation levels
#
# probe[s,v,n,t] - Stores the probe stimulus (x + \Delta)
#   for subject s in variance condition v, set size condition n
#   and trial t.
#
# respondClockwise[s,v,n,t] - A binary variable indicating whether
#   subject s reported a clockwise (positive) perturbation on trial
#   t of variance condition v and set size condition n.

model {

  # Response bias ******************************************************
  for(s in 1:nSubjects) {
    bias[s] ~ dnorm(0, 0.1);
  }

  # Lapse rate *********************************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      lapse[s,v] ~ dbeta(1, 10);
    }
  }

  # Encoding probabilities *********************************************
  for(s in 1:nSubjects) {
    for(v in 1:nVariances) {
      for(n in 1:nSetSizes) {
        pi[s,v,n,1:9] ~ ddirch(alpha[n,1:9]);
      }
    }
  }

  # Probability matching exponent **************************************
  for(s in 1:nSubjects) {
    sensitivity[s] ~ dgamma(1, 0.3);
  }

  # Sensory noise ******************************************************
  for(s in 1:nSubjects) {
    sigmaSensory[s] ~ dgamma(1, 0.1);
  }

  # Capacity (in nats) *************************************************
  # Mean capacity across all conditions
  RMean ~ dgamma(1, 0.3);

  # SD of mean capacity across conditions
  RSD ~ dgamma(1,1);

  # Convert to shape and rate
  RAlpha <- RMean^2 / RSD^2;
  RBeta <- RMean / RSD^2;

  # SD of capacity within each condition
  # (Assume homogeneity of variance)
  RSDCond ~ dgamma(1, 1);

  # Condition-specific distribution over capacity
  for(v in 1:nVariances) {
    for(n in 1:nSetSizes) {
```

```
    RMeanCond[v,n] ~ dgamma(RAlpha, RBeta);

    # Convert to shape and rate
    RAlphaCond[v,n] <- RMeanCond[v,n]^2 / RSDCond^2;
    RBetaCond[v,n] <- RMeanCond[v,n] / RSDCond^2;
  }
}

# Each subject's capacity sampled from the
# condition-specific distribution
for(s in 1:nSubjects) {
  for(v in 1:nVariances) {
    for(n in 1:nSetSizes) {
      R[s,v,n] ~ dgamma(RAlphaCond[v,n], RBetaCond[v,n]);
    }
  }
}

# Response distribution ********************************************
# Mean and variance of the response distribution governing the
# probability of responding with a positive perturbation, as a
# function of the variance and number of items encoded. Derivation is
# provided in the supplementary material.
for(s in 1:nSubjects) {
  for(v in 1:nVariances) {
    for(ne in 1:8) { # ne = Number items encoded

      sigmaEnc[s,v,ne] <- sqrt(sigmas[v]^2 + sigmaSensory[s]^2) /
        sqrt(-1 + exp(2*R[s]/ne));

      meanConstant[s,v,ne] <- sigmas[v]^2 /
        (sigmaEnc[s,v,ne]^2 + sigmas[v]^2 + sigmaSensory[s]^2);

      variance[s,v,ne] <- sigmaSensory[s]^2 +
        (sigmas[v]^4*(sigmaEnc[s,v,ne]^2 + sigmaSensory[s]^2))/
          (sigmaEnc[s,v,ne]^2 + sigmas[v]^2 + sigmaSensory[s]^2)^2;

      precision[s,v,ne] <- 1 / variance[s,v,ne];
    }
  }
}

# Observed data ****************************************************
for(s in 1:nSubjects) {
  for(v in 1:nVariances) {

    # Precision of the sensory encoding of the probe stimulus
    likPrec[s,v] <- 1 / (sigmas[v]^2 + sigmaSensory[s]^2);

    for(n in 1:nSetSizes) {
      for(t in 1:trials[s,v,n]) {

        #************************************************************
        # Response probability when the probe item was not encoded

        likClockwise[s,v,n,t] <- (1/4)*
          (dnorm(probe[s,v,n,t], bias[s] + abs(deltas[1]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] + abs(deltas[2]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] + abs(deltas[3]),
            likPrec[s,v]) +
          dnorm(probe[s,v,n,t], bias[s] + abs(deltas[4]),
            likPrec[s,v]));
```

```
          likCounterClockwise[s,v,n,t] <- (1/4)*
            (dnorm(probe[s,v,n,t], bias[s] - abs(deltas[1]),
              likPrec[s,v]) +
            dnorm(probe[s,v,n,t], bias[s] - abs(deltas[2]),
              likPrec[s,v]) +
            dnorm(probe[s,v,n,t], bias[s] - abs(deltas[3]),
              likPrec[s,v]) +
            dnorm(probe[s,v,n,t], bias[s] - abs(deltas[4]),
              likPrec[s,v]));

          probClockwise[s,v,n,t] <- likClockwise[s,v,n,t] /
            (likClockwise[s,v,n,t] + likCounterClockwise[s,v,n,t]);

          thetaGuess[s,v,n,t] <-
            (probClockwise[s,v,n,t]^sensitivity[s]) /
            (probClockwise[s,v,n,t]^sensitivity[s] +
            (1 - probClockwise[s,v,n,t])^sensitivity[s]);

          theta[s,v,n,t,1] <- thetaGuess[s,v,n,t];

          #************************************************************
          # Response probability for each possible number of items
          # encoded, ne, assuming ne > 0
          for(ne in 1:setSizes[n]) {

            mean[s,v,n,t,ne] <- deltas[delta[s,v,n,t]] +
              x[s,v,n,t]*(1 - meanConstant[s,v,ne]);

            theta[s,v,n,t,ne + 1] <- (1 - ne / setSizes[n])*
              (thetaGuess[s,v,n,t]) +
              (ne / setSizes[n])*
                (1 - pnorm(0, bias[s] + mean[s,v,n,t,ne],
                 precision[s,v,ne]));
          }

          # Marginalize over the number of items encoded,
          # and the probability of a lapse trial.
          thetaMarginal[s,v,n,t] <- lapse[s,v]*(1/2) +
            (1 - lapse[s,v])*sum(theta[s,v,n,t,1:(setSizes[n] + 1)]*
            pi[s,v,n,1:(setSizes[n] + 1)]);

          # Observed response
          respondClockwise[s,v,n,t] ~ dbern(thetaMarginal[s,v,n,t]);
        }
      }
    }
  }
}
```