Control over non-numerical parameters in numerosity experiments

Stanislas Dehaene, Véronique Izard, Manuela Piazza

Last updated September 2005

History and disclaimer

This series of Matlab programs were first employed in Piazza et al's (2004) *Neuron* paper, where further explanations can be found:

Manuela Piazza, Veronique Izard, Philippe Pinel, Denis Le Bihan, and Stanislas Dehaene. **Tuning curves for approximate numerosity in the human intraparietal sulcus.**. *Neuron*, 44(3):547-55, October 2004. [WWW] [PDF] [Abstract]

The original idea of matching intensive and extensive parameters emerged over the years, with influences from many prior experiments, particularly in infant numerical cognition. I was particularly influenced by this beautiful paper:

Xu, F., & Spelke, E. S. (2000). Large number discrimination in 6-month-old infants. *Cognition, 74*(1), B1-B11.

Further features were added in collaboration with Chiara Turati, Francesca Simion, Marco Zorzi, and Carlo Umilta, in preparation for collaborative infant experiments, and later with Koleen McCrink during her visit in the spring 2005. The following paper is particularly relevant.

McCrink, K., & Wynn, K. (2004). Large-number addition and subtraction by 9-month-old infants. Psychol Sci, 15(11), 776-781.

In 2004-2005, Miles Shuman [mshuman@fas.harvard.edu] made further improvements in preparation for collaborative fMRI experiments with Nancy Kanwisher, but these features have not yet been integrated to our programs.

Besides the matching scheme, the programs themselves were based on prior stimulus generation code kindly provide by Andreas Nieder, and which served as a basis for his discovery of number neurons. See e.g.

Nieder, A., Freedman, D. J., & Miller, E. K. (2002). Representation of the quantity of visual items in the primate prefrontal cortex. Science, 297(5587), 1708-1711.

It should be very clear that the enclosed programs have been developed for research purposes only, without intensive debugging efforts or any particular attention being paid to internal documentation. Discussions with colleagues have convinced me that the issues of stimulus matching in the number domain can be extremely tricky. For the Piazza et al. *Neuron* paper, this issue alone generated over 15 pages of discussions with the referees! Therefore, please read and re-read the enclosed document and, in case of doubt, check and re-check the enclosed programs yourself. No further help will be provided.

If you use the software in scientific publications, please cite the above *Neuron* paper, or use the following reference:

Dehaene, S., Izard, V., & Piazza, M. (2005). Control over non-numerical parameters in numerosity experiments. *Unpublished manuscript (available on www.unicog.org)*.

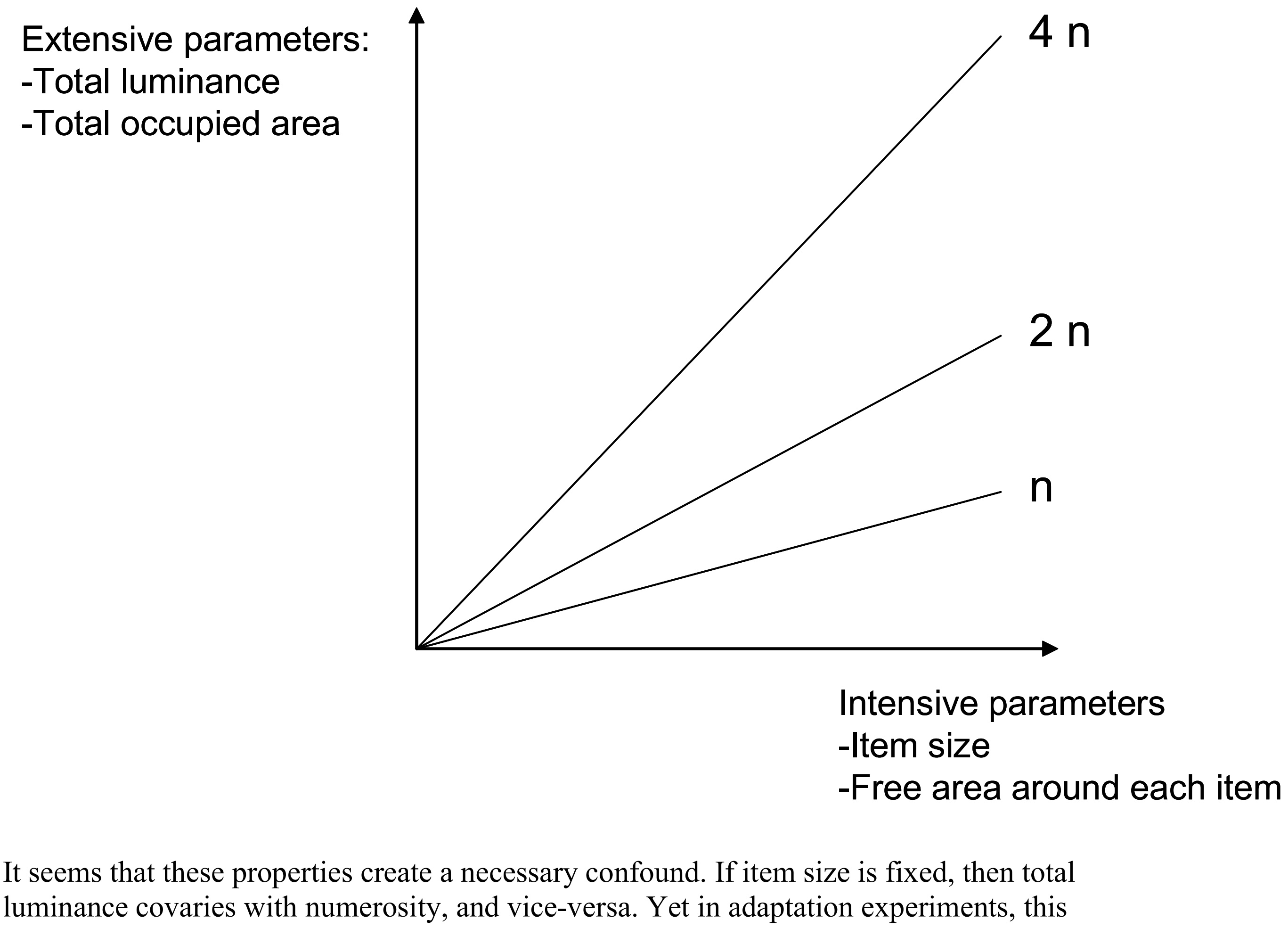
Introduction

When generating sets of objects as stimuli for numerosity experiments, the nature of the stimuli imposes obligatory correlations between intensive variables (defined at the level of each item) and extensive variable (defined at the level of the set, and which increase with numerosity).

For instance, individual item size (surface of each object) and total luminance (total surface of the objects) are linearly related once numerosity is fixed. You cannot change one without changing the other.

A similar linear relation exists between the average area occupied by each item (the average amount of "free space" around the items, or the inverse of density), and the total area occupied by the set.

These relations are depicted on the following diagram.



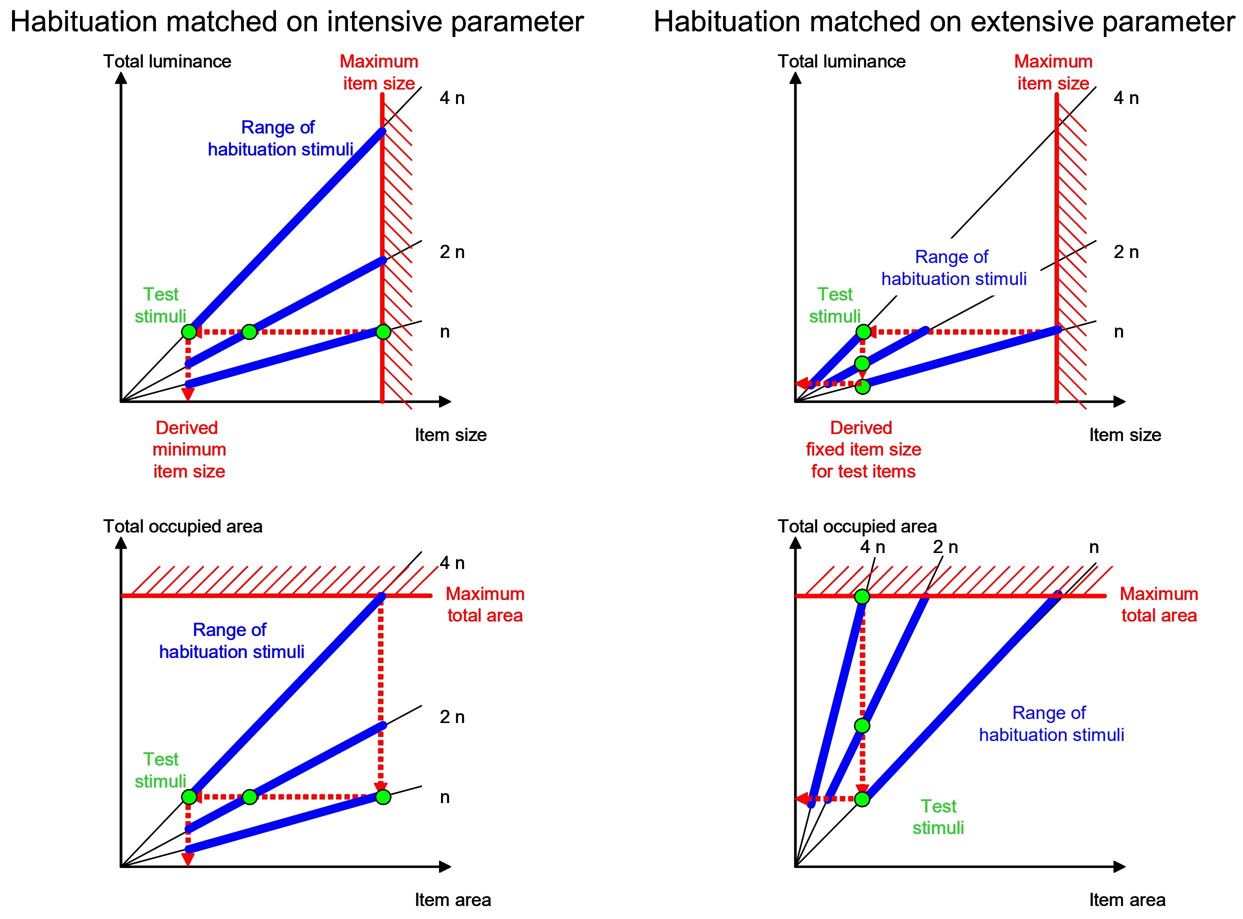
It seems that these properties create a necessary confound. If item size is fixed, then total luminance covaries with numerosity, and vice-versa. Yet in adaptation experiments, this problem can be controlled. The crucial trick is to control one parameter over the habituation, and the other over test.

Why can't the same images be used for the habituation and test sets? Because otherwise, the amount of numerical novelty would be confounded with the novelty arising from other non-numerical parameters. For instance, if individual dot area is fixed, then total stimulus area is proportional to number. If one was habituating with standard displays of 40 dots, and dishabituating with similar displays of 20, 40 or 80 dots, then the displays with 20 or 80 dots would not only deviate from the standard displays in number; they would also have a markedly different total stimulus area, which in itself might be sufficient to explain the subject's novelty response (or the neuron's recovery of firing, or the fMRI voxel's recovery from repetition suppression).

Here is how one can control for intensive and extensive parameters at once:

- For the habituation stimuli, one parameter (either the intensive or the extensive one) can be drawn from a fixed distribution, regardless of the habituation numerosity. The other parameter therefore necessarily varies with habituation numerosity
- For the test stimuli, however, this is reversed: the other parameter is fixed (without variation), and the first parameter is varied
- A further trick is that the distribution of the parameter which is statistically fixed during habituation is is chosen so that the smallest value is equal to the smallest value presented in the test, and the largest value is equal to the largest value presented in the test.
- As a result, all the individual values presented in the test have also been presented in the habituation. Thus, all of them are equally non-novel with respect to individual dot area.
- In summary: All of the test stimuli are statistically identical (and highly variable) on one parameter (say, total stimulus area), and all of the habituation stimuli are exactly identical on the other (say, individual dot area); thus none of those parameters can explain that a differential dishabituation response is obtained for specific combinations of habituation and test stimuli.
- To put this in a slightly different form: If a main effect of test number was found, it could perhaps be explained away by individual dot size (although this is unlikely because as noted above, all test dot sizes are equally non-novel). And if a main effect of habituation number was found, it could perhaps be explained away by total stimulus area. But any interaction of the two factors, as observed when a significant repetition or distance effect is found, cannot be explained by either factor alone.

The following figures illustrate how the range of parameters is selected.


Habituation matched on intensive parameter
Habituation matched on extensive parameter
Total luminance
Maximum item size
4 n
Range of habituation stimuli
2 n
Test stimuli
n
Derived minimum item size
Item size
Total luminance
Maximum item size
4 n
Range of habituation stimuli
2 n
Test stimuli
n
Derived fixed item size for test items
Item size
Total occupied area
4 n
Maximum total area
Range of habituation stimuli
2 n
Test stimuli
n
Item area
Total occupied area
4 n
2 n
n
Maximum total area
Range of habituation stimuli
Test stimuli
Item area

Programming details

In the program, the primary variables that are easy to manipulate are

- the item size (intensive)
- the total occupied area (extensive)

Once the figure size is defined (say, 400x400 voxels), and once the total number of objects to be displayed is also defined (say, max of 100 objects), this defines a pseudo-random grid where objects can be placed. This grid, in turn, defines maximum values for our two manipulated parameters.

- The maximum item size is defined so that, when two objects occupy neighboring positions, they don't touch each other
- The maximum total occupied area is defined by using the entire set of locations in the grid

Other values of those parameters can thus be defined relative to those maximum (expressed as percentage of the maximum).

Thus, two Matlab programs have been defined:

Generate_set.m

Arguments:

- numerosity = numerosity of the set to be generated
- shape = a standardized vector defining the shape
- windowsize = size of figure in pixels (the figure will be square)
- rmax = maximum radius of each item (set to 0 if you want automatic setting of this value (see below for discussion))
- totaloccupiedarea (expressed as percentage of max, from 0 to 1 – use 1 by default)
- itemsize (expressed as percentage of max, from 0 to 1 – use 1 by default)

This function will generate one set of objects on the screen.
The shape must be defined in the following way by the calling program:

```
 shape = cell(1,2);
 %%CIRCLE
 t = (0:2*pi/200:2*pi);
 shape{1} = sin(t);
 shape{2} = cos(t);
%% TRIANGLE (of same surface as circle)
 lambda = 2*pi/(3*sqrt(3));
 shape{1} = lambda*[-sqrt(3)/2,0,sqrt(3)/2];
 shape{2} = lambda*[-0.5,1,-0.5];
%% SQUARE
 shape{1} = [ -1 -1 1 1 ];
 shape{2} = [ -1 1 1 -1 ];
```

The figure can be saved with the following code:

```
 outfile = 'example.bmp';
%%%%% take a snapshot
 f = getframe(gcf);
 [img, map] = frame2im(f);
 imwrite(img, strcat(outdir,outfile));
```

Generate_habdevstim.m

Arguments:

- habnum: a vector of habituation numerosities (e.g. 1:5)
- testnum: a vector of test numerosities (e.g. 1:5)
- nhabtrials: number of habituation trials generated for each numerosity
- ntesttrials: number of test trials generated for each numerosity
- habshape: 1,2, or 3 for circle, triangle or square
- devshape: 1,2, or 3 for circle, triangle or square
- matchitemsizeonhab : 0 or 1 to have item size matched on habituation trials
- matchtoaonhab: 0 or 1 to have total occupied area matched on habituation trials
- outdir: output directory for the images

This function generates a whole set of images for a habituation/dishabituation type of experiment

The program will generate, within the directory “outdir”, one subdirectory for each habituation value that must be generated. In this directory, there will habituation images (starting with h), and deviant images (starting with the letter d if it is simply a numerosity test stimulus without shape change, or t if there is also a shape change). The nice thing about this naming convention is that the very same eprime program can be used with different directories to create habituation experiments with different numerosities.

Special note on the use of rmax variable

If you are only generating isolated numerical displays using generate_set, setting rmax=0 will automatically select the largest possible item size given the constraints of the display. This is ideal for picture legibility given the number of pixels available.

If, however, you want to generate multiple numerical displays, matched according to the above scheme, then it is essential to adopt a single value of rmax common to all displays (so that total luminance is adequately controlled). Generate_habdevstim.m does this correctly. Note that, unfortunately, the above controlling scheme implies that sometimes the items are quite small and hard to discriminate on screen. There is no simple solution to this. In many cases, only changing the range of tested numerosities will solve the problem (what determines the size, ultimately, is the larger numerosity).

In generate_habdevstim, a trick has been used to address this problem. For the habituation stimuli, instead of generating random values independently along the dimensions of itemsize and total occupied area, a single random number is used (i.e. the two dimensions are correlated). This has the advantage of allowing a greater size of items, because larger item sizes are used only when the set has a larger total occupied area, and hence the dots are well spread apart. One problem, however, is that not all combinations of itemsize and total occupied area are presented during habituation (only a “line” is presented in this stimulus space, rather the full “square” of random possibilities).

This scheme implies, in particular, that the particular combination of item size and total occupied area which occurs on test trials may never be presented in the habituation set. This is not a problem if the null hypothesis is that subjects attend only to individual values of these non-numerical parameters, not their combinations. Furthermore, the latter can be totally avoided if habituation stimuli are matched either on item size, or on total occupied area, but not both. For this reason, I recommend to use generate_habdevstim with 0,1 or 1,0 combinations of the parameters matchitemsizeonhab and matchtoaonhab.

If you don't like this special trick, you can replace "randomreal" by "rand" in the code. To avoid collisions between objects, you'll also have to reduce rmax on line 62, to something like (1/sqrt(maxnum))/2.5.

Rectangular stimuli and control over contour length

There are yet other ways of controlling stimuli, which can be better in some situations. For instance in baby experiments McCrink and Wynn (2005) suggest that it is best to match *both* total area and total contour length over the pair of stimuli proposed in a surprise experiment. This is because babies seem to be very sensitive to both of these variables, and failing to control for them might generate a major bias in their looking times to the two alternative stimuli.

Koleen McCrink has found that the two variables can be controlled simultaneously if one use displays made of rectangle stimuli. This is because, for a fixed area, perimeter can be changed at will (above some minimal value) by changing the aspect ratio of the rectangle.

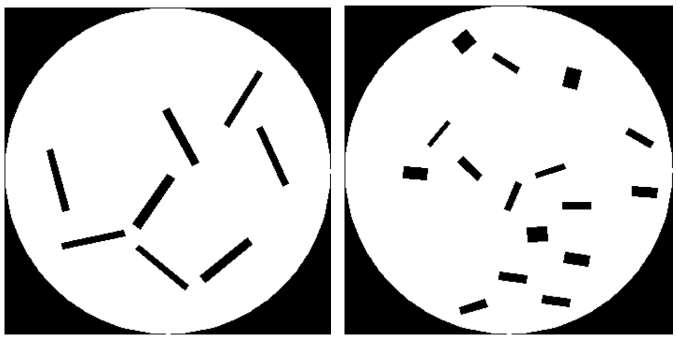
I have modified generate_set.m so that it can generate stimuli matched on total area and total contour length. The new program is called generate_rectangle.m. It requires specifying not just the current numerosity to be generated, but also the maximum numerosity used in the experiment, and to which all stimuli are matched in terms of both area and contour length.

The function outputs the actual measured area and contour length so that one can verify that they are identical across a bunch of stimuli;

No attempt is made to control that the rectangles do not overlap. This would need more programming, but meanwhile you can just select a sufficiently small itemsize so that overlap is generally prevented.

One problem with this way of generating stimuli is that the average shape of the items (i.e. their aspect ratio) is confounded with numerosity. At an extreme, if all shapes were identical, then the largest numerosity would be made solely of squares, while the smallest numerosity would have only elongated rectangles. To mitigate this problem, I have introduced some inter-item variability in the area of the individual items (and therefore their shape). This is controlled by the parameter sizevariability.

Example of matched stimuli generated by
generate_rectangles(8,350,0,1,0.3,0.4,16) and generate_rectangles(16,350,0,1,0.3,0.4,16);
(parameters explained below)



Note that already in this example the change in shape between the two displays is quite striking. This problem becomes much worse as the numerosities to be matched differ more. Here the ratio is 2:1 (16:8). With a higher ratio (e.g. 32:8) the rectangles become impossibly skinny and cross each other for the smaller number. Thus, it is not possible to use this method to make well-matched stimuli across a large range of the logarithmic number line.

generate_rectangles(numerosity,windowsize,rmax,totaloccupiedarea,itemsize,sizevariability, maxdesired)
% This function will generate one set of objects in figure 101.

```
% Arguments:
% -      numerosity = numerosity
% -      windowsize = size of figure in pixels (the figure will be square)
% -      rmax = maximum radius of each item (set to 0 if you want automatic
% setting of this value (see below))
% -      totaloccupiedarea (expressed as
% percentage of max, from 0 to 1)
% -      itemsize (expressed as percentage of max, from 0 to 1)
% - sizevariability (expressed as percentage from 0 to 1) = amount of
% variation in area between the different shapes;
% - maxdesired = max desired numerosity in the entire experiment (to which
% the total contour length will be matched)
%
% Typical example:
%
% generate_rectangles(8,350,0,1,0.3,0.4,16);
% and generate_rectangles(16,350,0,1,0.3,0.4,16);
% creates matched sets of 8 and 16 rectangles in a 350-voxel window.
% The objects are distributed over the entire window (1=100% of windowsize),
% and are at 30% of the maximum allowable size (this parameter chosen to
% avoid too much frequent between items)
% They exhibit a + or - 40% variation in area.
```