**WRITING NIfTI FILES**

The aspect that makes SAA simulations stand out, however, is that the generated time-series are then written into NIfTI files. The motivation behind writing the time-series into NIfTI files, rather than keeping them in their matrix form, is that in this way they can be analysed using the exact same analysis pipeline that is used to analyse the empirical dataset. We created one 4D NIfTI file for each run using a custom function adapted from SPM. As requested by the NIfTI format, each file contained a header with metainformation (e.g. file name, data type, etc.) and an image with the actual data (the generated time-series). Importantly, the function outputs an additional 4D array with the names of the voxels arranged as they are in the NIfTI file.

A question arising is how to best arrange the voxels in the NIfTI files. A first point to keep in mind is that SPM functions have been written to manipulate images which are usually three or four dimensional. Consequently, in order to avoid unexpected downstream errors from the software, we recommend to write NIfTI files that always have a third dimension. That is, if one wanted to test an extremely simple model containing only one voxel, we would recommend to add extra voxels in each dimension - most critically the third - with no data (e.g. only NaNs). When it comes to more complex full models, different arrangement options are available. One possibility is writing separate NIfTI files for each full model. This is a straightforward option that keeps a clear one-to-one mapping between models and files. The simplicity of such solution allows experimenters to know what the content of the files is with certainty, reducing the probability of making mistakes. At the same time, it has the disadvantage of being computationally expensive. Researchers are normally interested in running simulations with multiple models. This includes modelling brain activity under different hypotheses, as well as variations of the same model by changing weights and noise parameters.

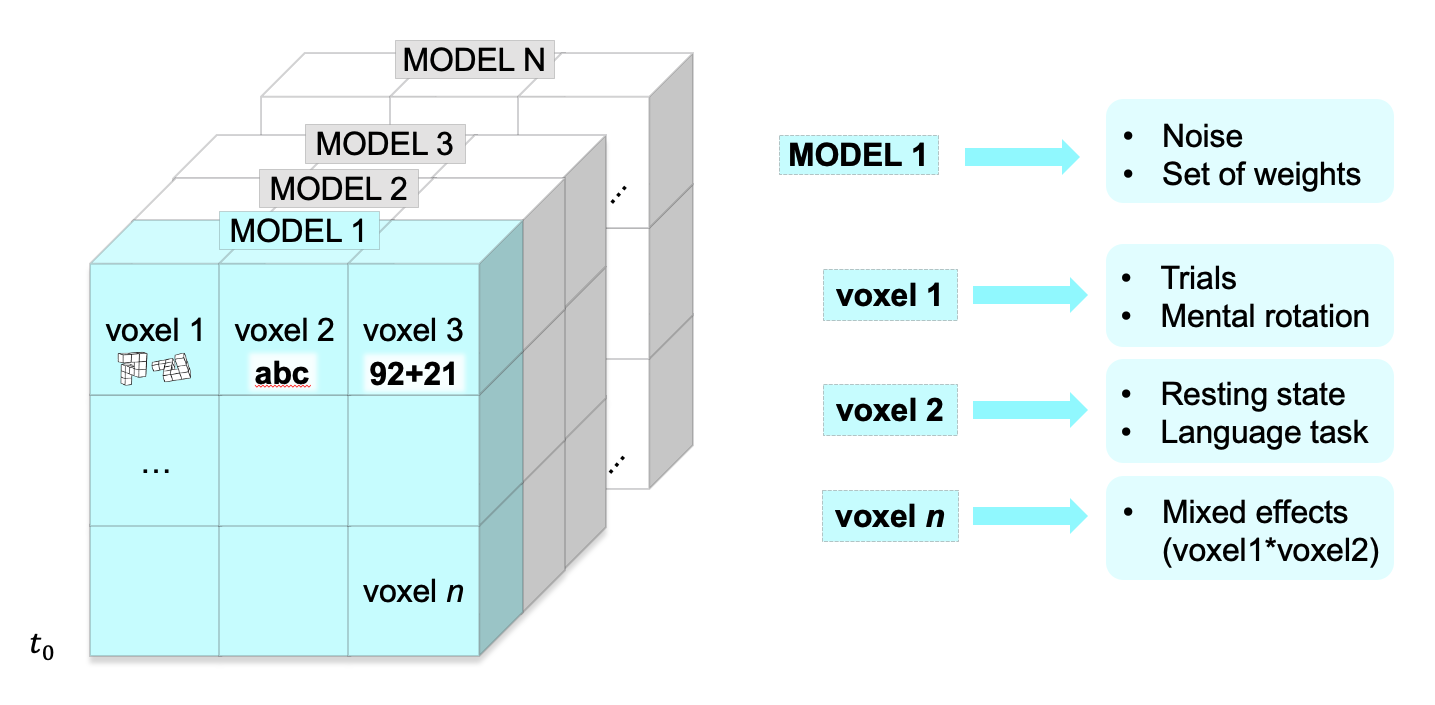
This allows to quickly and safely identify the voxels specified in the model. Since the process of writing, saving, and loading NIfTI files requires non-negligible computational resources, we suggest two complementary solutions to speed up the process. First, the temporary storage of data. In our case, this was achieved by writing the simulated dataset into a temporary directory on a local hard drive of the executing computer first and by copying it back to a permanent server-side directory once its generation was completed. Second, writing more than one full model – i.e. the total number of voxels containing the time-series generated with specified parameters (e.g. weights, noise, mixed-effects) - in each NIfTI file. This option is particularly advantageous when one wants to compare multiple versions of a model by changing its parameters and not the underlying hypothesis (e.g. investigating the effects of noise on the analysis results by running the same simulation with different noise parameters). This method allows to considerably speed up the data generation process, at the same time maintaining conceptual clarity by clustering minor variations of the same model into a unique folder structure. There are several potential ways to organise the voxel positions of each model. We stored different full models in the third dimension of the NIfTI image (Figure 4). The *n* voxels relative to each full model were arranged in two dimensional square slices with length equal to . In case a full model had fewer voxels than the ones in the slice (e.g. 8 the full model and 9 the slice), the remaining voxels (e.g. 1) would contain only *NaN*s. This is a simple way to arrange any given number of voxels in two dimensions systematically. It should be noted that, although the voxel arrangement in the two dimensions should not affect the outcome of the simulation, a clear criterion can help the researcher maintain order in the dataset and it is therefore desirable. Independently of the arrangement chosen, when writing the NIfTI files it is crucial to store information about how these models have been ordered. In order to run a separate analysis for each model, in fact, it is necessary to mask out the voxels relative to other models. Since tracing back the mapping between voxels and models *a posteriori* is hardly possible, we recommend to create multiple sets of masks together with the NIfTI files.[[1]](#footnote-1)

Figure 4 Layer-wise storage of computational models in the third dimension of an example NIFTI file. Note that all models need to have the same number of voxels.

**FUNCTION SETS**

Piecewise-defined functions

We first used piecewise linear functions to provide a simple model to describe the activity of a single voxel throughout a block. That is, we defined the amplitude of the signal before convolution with the HRF at every time point of the block. We used two sets of piecewise-defined functions, one for the persistent activation hypothesis and one for the active suppression hypothesis. In this case, we did not simulate the no-aftereffect hypothesis.

The *persistent activation* hypothesis was modelled using the function *ffadeout* defined over three sub-domains [ *tcue*, *ttrial*), [*ttrial*, *trest*), and [*trest*, *tend*], which represent respectively the time interval (in seconds) over the cue phase [0, 2), the time interval over the trial phase [2, 54), and the time interval over the resting phase. In the time interval from the beginning of the block until the trial onset, the modelled signal rises linearly. Then it remains constant until the trial offset, when it decreases linearly until it reaches 0 at the end of the trial:

|  |  |  |
| --- | --- | --- |
|  |  | ( 1 ) |

Where *a, b,* and *c* are arbitrary coefficients that model respectively the activity in the cue, trial, and resting state phases.

The *active inhibition hypothesis* was modelled using the function *finhibition* defined over four sub-domains [ *tcue*, *ttrial*), [*ttrial*, *trest1*), [*trest1 , trest2*), and [*trest2*, *toff*], which represent respectively the time interval over the cue phase [0, 2), the time interval over the trial phase [2, 54), the time interval at the beginning of the resting phase where an eventual active inhibition is to be expected, and the time interval over the resting state, once the activity has returned to baseline after the suppression. In this case too, the modelled signal first rises linearly and then remain constant until the trial offset. Then, it decreases linearly until it reaches its minimum after 2 s (one TR), to then recover to baseline logarithmically. Formally, it was defined as

|  |  |  |
| --- | --- | --- |
|  |  | ( 2 ) |

Boxcar functions

We used boxcar functions to provide a simple model that would describe how brain activity may vary depending on the experimental condition, including both the block phase (cue, trial, or resting state) and the task (language, working memory, arithmetic task). In this case, we modelled only the no-aftereffect hypothesis. We used three boxcar functions, *fcue ftrial frest*, defined over the time interval *t* [*tcue*, *tend*] where *tcue* = 0s and *tend* = 90s, and where [ *tcue*, *ttrial*) is the time interval over the cue phase, [*ttrial*, *trest*) the time interval over the trial phase, and [*trest*, *tend*] the time interval over the resting phase:

|  |  |  |
| --- | --- | --- |
|  |  | ( 3 ) |
|  |  | ( 4 ) |
|  |  | ( 5 ) |

Where *kcue* , *ktrial , krest* describe the amplitude of the signal in each block phase. Each function was used to determine the activation of at least one voxel. The stimulations using the boxcar function sets, thus, assumed that there would be at least one voxel responding to each and only one of the three block phases, in contrast to the piecewise-defined function sets that modelled the activity of voxels always throughout all three phases.

Stick functions

We used stick functions to refine our simulations by accounting for stimulus onset at every trial within the block, a feature of the experimental design that is lost when brain activity is modelled using piecewise-defined or boxcar functions.

We built several sets of stick function-based simulations, which shared one characteristic with the boxcar-based simulations: they aimed at capturing differences in activation depending on task-phase and consisted of three sub-models - one that described activation in the cue phase (cue model), one in the trial phase (trial model), one in the resting state phase (resting model). These models can roughly thought of as brain regions that are more active preferably in one of the three block phases.

The combination between different types of cue, trial, and resting state models, depended on the underlying hypothesis. No after-effect models shared the same cue and trial models, but varied in the way the resting state models were defined. Persistent activation and active inhibition models shared the same cue and resting state models, but varied in the way the trial models were defined. That is, we imagined that the voxels or brain regions most active during the trials would also be the one that show either an persistent activation or an active inhibition of the signal. On the contrary, voxels or brain regions only engaged in the resting state phase would not be affected.

*Cue models* all consisted of a single stick at the beginning of the cue phase, i.e. at the beginning of the block, and where therefore set to 0 in any other time point of the block.

*Trial models* consisted of 8 sticks of the same height, except for one family set in which the first stick was 50% higher than the other 7. Sticks were placed at the beginning of every trial, which were thus equally spaced (6 seconds apart) throughout the trial phase. Trial models, however, differed in the way they were defined over the resting phase, depending on the hypothesis of interest.

In a first class of no-aftereffect models, which was conceived as a control, the signal amplitude was set to 0 for the resting state phase. A second class of models was used to model the *persistent activation* hypothesis. We modelled the slow signal fadeout by introducing a constant decrease of the sticks from the end of the last trial (*trest1*) to the end of the block. We sampled 5 out of 6 datapoints at equal intervals of 6 s from a function of the kind *F(t) = k/t*, where k is a parameter representing the speed of the persistent activation. Each datapoint was used to define the height of the “stick”. To describe the return to baseline, we set the last datapoint to 0, as the continuous function F(x) approaches it only asymptotically. A third class of models was used to model the *active inhibition* hypothesis. To model the suppression of the signal, we used two different approaches, both relying on negative stick functions. A first sub-model (*long inhibition*) was computed using sticks that approach zero increasingly. In this case, we sampled 5 out of 6 datapoints at equal intervals of 6 s from a function of the kind *I(t) = - k/t*, where k is a parameter representing the strength of the inhibition. Each datapoint was used to define the height of the “stick”. Like in the previous class of models, in order to describe the return to baseline, we set the last datapoint to 0, as the continuous function *I* (x) approaches it only asymptotically. A second sub-model (*short inhibition*) consisted of a single stick placed at the end of the trial phase. The height of the stick in the resting state phase was the additive inverse of the height of the sticks in the trial phase.

*Resting models* were defined only in the resting phase and set to 0 in the cue and trial phases. However, they were computed in three different ways. In a first version, the resting model was not defined using a stick function, but using a boxcar function. This model assumed a constant activation throughout the resting phase. The peak amplitude of the boxcar function was scaled to match the signal energy of the stick function in the trial phase, such as

|  |  |  |
| --- | --- | --- |
|  |  | ( 6 ) |

Where is the peak amplitude of the stick function, is the peak amplitude of the boxcar function, is the length of the time bin in seconds, and is the length of the time interval between sticks. In a second version of the resting model, we used 6 sticks with the same frequency as the trial model (one at every 6s interval). This is a straightforward way of building the model, but it has at least one major limitation: the resting phase has no regularly occurring trial onsets. Therefore, we ran a third version of the resting model, where we used 4 sticks placed at random locations withing the resting state phase. The time points for each stick were determined randomly, but in a way that made sticks less likely to occur later on in the resting phase. The random position was determined drawing the target time points from a cumulative probability distribution that was obtained from a probability density function of the kind *f(x) = 1/x***.** This version of the resting model was chosen over the other two because it relies on fewer assumptions about the activity of a voxel, modelling spontaneous neural activations with participant- and run-dependent frequency.

|  |  |  |  |
| --- | --- | --- | --- |
| Function set | Hypothesis | Model before convolution | Model after convolution |
| Boxcar | none |  |  |
| Piecewise-defined | fade-out |  |  |
| Piecewise-defined | inhibition |  |  |
| Stick function | fade-out |  |  |
| Stick function/boxcar | no-aftereffect |  |  |
| Stick function  (all rest sticks) | no-aftereffect |  |  |
| Stick function  (one rest stick) | no-aftereffect |  |  |
| Stick function  (higher trial stick) | no-aftereffect |  |  |
| Stick function  (random rest stick) | no-aftereffect |  |  |
| Stick function | inhibition (short) |  |  |
| Stick function | inhibition |  |  |

Figure 3. Models used in the simulations, before and after convolution with the HRF. In each table cell, three models are depicted: cue models (left), trial models (centre), resting state models (right).

1. For example, masks may allow to mask out voxels that are most responsive to a particular task or block phase. [↑](#footnote-ref-1)