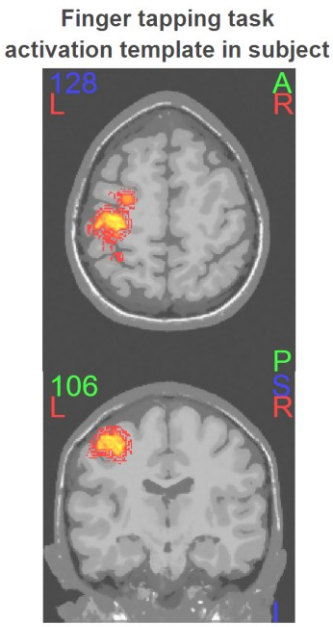
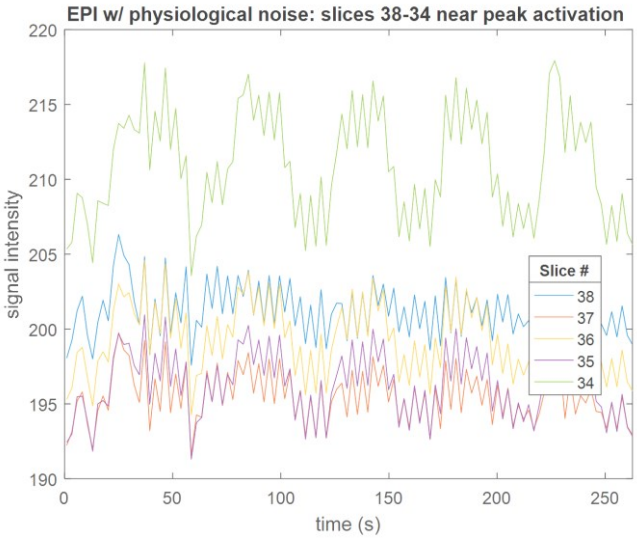
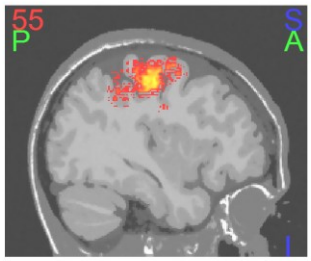
STANCE:
The Spontaneous & Task-related Activation of Neuronally Correlated Events simulator

User Guide







Jason E. Hill

Xiangyu Liu

April 6, 2017

Table of Contents

1. Introduction

The MATLAB-based functional magnetic resonance image (fMRI) simulator STANCE, the **S**pontaneous & **T**ask-related **A**ctivation of **N**euronally **C**orrelated **E**vents simulator is a Statistical Parametric Mapping (SPM) add-on toolbox with command-line functions useful for scripting. This simulator aims to generate realistic task-related and/or resting-state 4D blood oxygenation level dependent (BOLD) signals, given the experimental paradigm, fMRI scan protocol, and physiological noise parameters by using digital phantoms of twenty normal brains available from BrainWeb (http://brainweb.bic.mni.mcgill.ca/brainweb/).

See Fig. 1 for a complete flowchart illustrating the generation of a single fMRI volume. The first step is to create the fMRI study, which involves specifying the fMRI scanning protocol, loading or defining any neuronal activations being investigated, and then specifying the experimental design and noise model. The next step is to select a subject from one of the 20 normal brain phantoms available in the simulated brain database BrainWeb (Aubert-Broche *et al.* June & Nov. 2006). Each brain phantom has a T1-weighted (T1-w) anatomical volume and high resolution fuzzy membership volumes for 12 different tissue types, which are used in simulation. The third step is to allow specific factors of the session to be added, e.g. motion. Then, from the fMRI scan protocol and the brain phantom resources, the subject's anatomical "native space" is resliced to construct the "functional space," which is then used in the generation of a sequence of fMRI volumes from the EPI signal equation (Liang & Lauterbur 1999). For details about generating 4D time-series see the paper included in reports (Hill *et al.* 2017) titled "A Task-related and Resting State Realistic fMRI Simulator for fMRI Data Validation".

STANCE requires SPM8 to run; future versions will also be compatible with SPM12. SPM8 can be obtained http://www.fil.ion.ucl.ac.uk/spm/software/spm8/. It is helpful to install SPM to an easily reference directory, e.g. C:\spm\spm8. NOTE: in the development stage some functions from NIFTI tools have also been used to preprocess the phantom data, these are available from MathWorks, but any necessary for input/output operations are included along with a copy of the NIFTI tools license.

1. Download/unzip to a folder. It is best to place the STANCE folder in the parent directory of the SPM installation, e.g. for SPM8 installed in C:\spm\spm8, place the STANCE files in C:\spm\STANCE.
2. Start MATLAB.
3. Type on Command Window:

   >>addpath('C:\spm\spm8')
4. Now STANCE functions can be used on the Command Window, in MATLAB scripts or functions. See the demos, demo_3D_*.m & demo_4D_*.m in the scripts_for_demos folder.

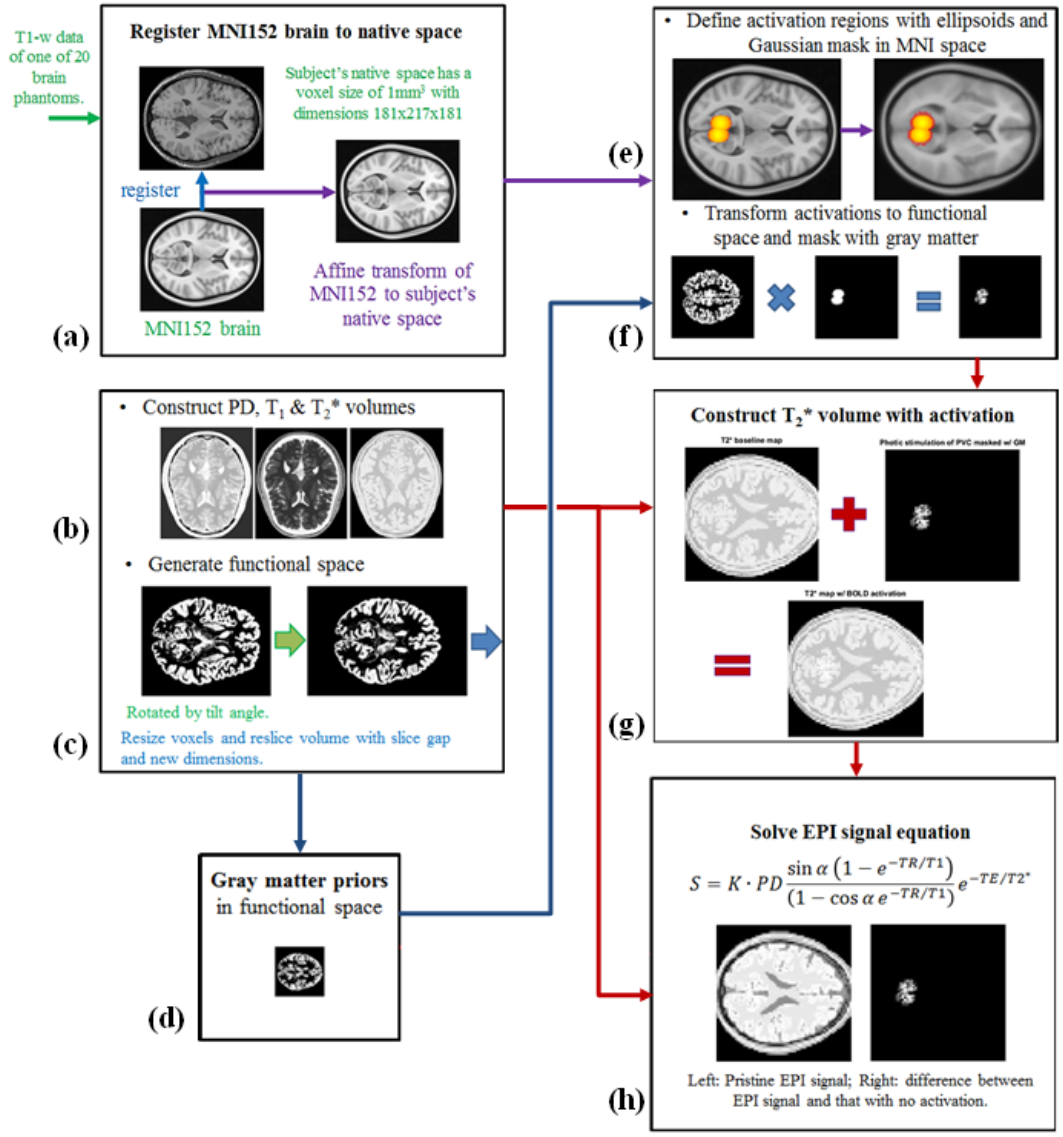


Figure 1. Flowchart outlining the generation of an fMRI volume by STANCE.

2. Functions

The STANCE simulator consists of several tool packages:

- **Input-output tools** - for reading and writing data between MATLAB and other data formats
- **Working level tools** - for setting four working levels the simulator would work on: global, study (level 1), subject (level 2) and session (level 3)
- **3D volume tools** - for registration, reslicing volumes to functional space, defining modeled neuronal activation maps and maps of MRI parameters (T1, T2* and proton density)
- **The 4D BOLD activation tools** - for simulating the BOLD time series by modelling the hemodynamic response with noise, attenuation and motion
- **Noise tools** - for adding spatially and temporally varying noise of various types
- **MRI signal tools** - for simulating MR images; the EPI sequence derived fMRI data with noise, attenuation and motion.
- **Correlation tools** – (under development) for adding correlation to time series.

These tools and scripts are organized into different folders. In addition to the tools listed above, STANCE also provides a series of demo scripts which illustrate some simulation pipelines of fMRI data. The demo scripts each showcase various STANCE function combinations.

In conclusion, a realistic fMRI simulator called STANCE that allows user-defined activations and modular improvements has been initiated, with the goal to include not only task or stimulation-related time series as directed by an experimental design convolved with a spatially varying HRF, but also to include resting state time-series driven by a potentially dynamic correlation matrix. STANCE can serve as a benchmark for various fMRI analyses via statistical error quantification from activation maps and be used as a validation tool for fMRI studies.

2.1 Global elements

When first running STANCE many items, such as STANCE.m, need to be built and initialized.

■ **STANCE_initialize_STANCE.m**

USAGE:

Call `STANCE_initialize_STANCE` with no arguments

DESCRIPTION:

Initializes STANCE.mat and other global resources, such as many of the file locations, such as the provided 20 subject brains and their file names, upon the first execution of STANCE

■ **STANCE.mat**

USAGE:

```
load('STANCE.mat')
```

DESCRIPTION:

Loads the global STANCE variables to be used in many functions (Listed below).

VARIABLES	VALUE	DESCRIPTION
SPMpath	…	Path of SPM8
STANCEroot	...	Path of STANCE simulator
M_array	4x4x20 matrix	Affine transformation matrices
filenameMNI	...	Full directory of MNI T1 image
filenameGM	...	Full directory of MNI gray matter image
MNIgmVolume	1012523.2971	Sum of gray matter priors in MNI images
Nbr_sss	[1,1,1]	Working levels of STANCE: study, subject & session
Now_sss	[1,1,1]	Current number of working level
Nbr_subject_labels	20	Number of subjects
subject_labels	4 5 6... 54	Original 20 subjects postfix labels (e.g. subject04)
male_labels	1 3 4 ...	Labels for male (index for subject labels)
female_labels	2 5 7 ...	Labels for female (index for subject labels)
subjectsBase	'.../subject'	Directory of subjects and their base: 'subject'
subjectsFuzzyPostfix	'_fuzzy.nii.gz'	Fuzzy map postfix of subjects
subjectsT1postfix	'_t1w_p0.nii.gz'	T1-w image post fix of subjects

- **STANCE_reference.m & reference.mat**

USAGE:

Run the `STANCE_reference.m` script and/or call `load('reference.mat')`.

DESCRIPTION:

Provides an optional global level reference structures or the loading of spreadsheet data for various resources of interest to the user, e.g. description of various atlas/parcellation ROIs.

2.2 File input/output tools

The STANCE simulator provides various 'load' functions to accept 3D/4D data in gzipped or un-gzipped MR data files.

- **STANCE_load_data.m**

USAGE:

```
Data = STANCE_load_data(fileName, dataDir, gzipFlag, mask);
```

DESCRIPTION:

Loads 3D/4D MRI data (with mask) from specified file in a certain data directory (optional).

INPUTS (default value):

fileName	= The file name or path location (NIFTI or ANALYZE files; can be gzipped).
dataDir ([])	= The directory of file location (optional).
gzipflag (false)	= Logical flag for keeping only archived gzipped files (lower storage cost) or allowing for unzipped files to remain (faster run times).
mask ([])	= Implicit zero mask for image datatypes without a NaNrep to be used by `spm_read_vols(v,mask)`.

OUPUT:

Data	= 3D/4D double data contained in specified file name.

- **STANCE_load_header.m**

USAGE:

```
V = STANCE_load_header(fileName, dataDir, gzipFlag);
```

DESCRIPTION:

Loads header information from the file in data directory into a SPM header structure V.

INPUTS (default value):

fileName = The file name or path location (NIFTI or ANALYZE files; can be gzipped).

dataDir ([]) = The directory of file location (optional).

gzipflag (false) = Logical flag for keeping only archived gzipped files (lower storage cost) or allowing for unzipped files to remain (faster run times).

OUPUT:

V = The SPM header structure containing info about the specified file.

- **STANCE_load_map.m**

USAGE:

```
map = STANCE_load_map(P_in, P_ref, specification, gzipFlag);
```

DESCRIPTION:

Loads a map of a spatially varying quantity for various usages in simulation with the same dimensions and origin as the reference volume.

INPUTS (default value):

P_in = The map used as activation region (filename, SPM header, or struct).

P_ref = The reference map for activation registration (filename, SPM header,).

specification ([]) = Specification parameters, e.g. the component index number, the ROI label, selection threshold for an 3D SPM, or volume number for 4D data.

gzipflag (false) = Logical flag for keeping only archived gzipped files (lower storage cost) or allowing for unzipped files to remain (faster run times).

OUPUT:

map = The 3D volume specifying a spatially varying quantity from P_in with the same dimensions and origin as P_ref.

- **STANCE_load_volume.m**

USAGE:

```
[V, Y] = STANCE_load_volume(fileName, dataDir, gzipFlag, mask);
```

DESCRIPTION:

Loads 3D/4D MRI volume header and data (with mask) from specified file in certain data directory. V is the SPM header structure, Y is 3D/4D data.

INPUTS (default value):

fileName = The file name or path location (NIFTI or ANALYZE files; can be gzipped).

dataDir ([]) = The directory of file location (optional).

gzipflag (false) = Logical flag for keeping only archived gzipped files (lower storage cost) or allowing for unzipped files to remain (faster run times).

mask ([]) = Implicit zero mask for image datatypes without a NaNrep to be used by `spm_read_vols(v,mask)`.

OUPUTS:

V = The SPM header structure containing info about the specified file.

Y = The 3D/4D image data contained in the specified file.

- **FuzzyBrain_Interpret_s04.m**

USAGE:

Run the `FuzzyBrain_Interpret_s04.m` script (after first obtaining subject04's minc format fuzzy models raw data from BrainWeb).

DESCRIPTION:

This script was used to load and preprocess the Fuzzy Model of the Anatomical Normal Brain: of subject04 for use in the simulator. This was repeated for all 20 subjects in like manner.

- **get_file_parts.m**

USAGE:

```
[fileBase, fileExtension] = get_file_parts(fileName);
```

DESCRIPTION:

This function returns base of file name and file extension (ungzipped).

INPUTS:

fileName = The name of file (NIFTI or ANALYZE files; can be gzipped).

- **loadminc.m**

USAGE:

```
[imaVOL, scaninfo] = loadminc(filename);
```

DESCRIPTION:

This function loads MR data in the minc(.mnc) file format.

INPUTS:

fileName = The name of minc format file.

OUTPUTS:

imaVOL = The 3D/4D image data contained in the minc(.mnc) file.

scaninfo = The header info contained in the minc(.mnc) file.

SOURCE:

MATLAB library function for MIA_gui utility. University of Debrecen, PET Center/LB 2010.

- **make_fuzzy_niigz.m**

USAGE:

Run the `make_fuzzy_niigz.m` script. Used in pre-processing the data for use in STANCE.

DESCRIPTION:

This function generates a 181x217x181x12 gzipped NIFTI of the $1mm^3$ isotropic 3D fuzzy labels (0.0-1.0) collating the 12 tissue types into the 4^{th} dimension label, as from a*.mat file which was generated by FuzzyBrain_Interpret_sXX.m from the 12 McGill's BrainWeb *.mnc tissue data files (20 normal brains) . The results of this conversion are stored in the STANCE/subjects/ subfolder by phantom name.

- **make_T1_niigz.m**

USAGE:

Run the `make_T1_niigz.m` script. Used in pre-processing the data for use in STANCE.

DESCRIPTION:

This function generates a 181x217x181 gzipped NIFTI of the $1mm^3$ isotropic T1-w anatomical image, as derived from McGill's BrainWeb *_t1w_p4.mnc data files (20 normal brains). The noise was suppressed with a 3D bilateral filter to approximate a noiseless signal. The result is stored in the STANCE/subjects/ subfolder by phantom name.

- **make_T1_brain.m**

USAGE:

Run the `make_T1_brain.m` script.

DESCRIPTION:

This function generates a 181x217x181 gzipped NIFTI of the $1mm^3$ isotropic T1-w extracted brain image, as derived from the NIFTI file of the full T1-w anatomical image.

2.3 Working directory and level tools

The organizational structure of the STANCE simulator works on four levels: *global*, *study* (level 1), *subject* (level 2), and *session* (level 3). Therefore, prior to carrying out a simulation, everything specific to the simulated fMRI **study** must first be specified (fMRI protocol, activation templates, & experimental design), then the **subject** selected (brain phantom, biological sex, idiosyncrasies), and finally any scanner peculiarities to the **session** (e.g. motion) are noted. So each simulation can be specified by a unique triple integer: study number, subject number and session number. This important triple of counting numbers is referred to in variable and function names as the "sss."

- **STANCE_choose_subject.m**

USAGE:

```
[V, Y] = STANCE_choose_subject(s, mode, gzipflag);
```

DESCRIPTION:

Selects the subject brain to be used for simulation.

INPUTS (default):

s = The numerical label of the subject as used by STANCE (1-20).

mode ('fuzzy') = The mode/type of subject data to be used: 'T1' for T1-w or 'fuzzy' for fuzzy logical tissue maps.

gzipflag(false) = Logical flag for keeping only archived gzipped files (lower storage cost) or allowing for unzipped files to remain (faster run times).

OUPUTS:

V = The SPM header structure containing info about the specified file.

Y = The 3D/4D image data contained in the specified file.

- **STANCE_random_subject.m**

USAGE:

```
s = STANCE_random_subject(sex);
```

DESCRIPTION:

Selects a random subject brain to be used for simulation in STANCE.

INPUT (default):

sex ('any') = The biological sex of the phantom brain of interest (optional).

OUTPUT:

s = The numerical label of the subject as used by STANCE (1-20).

- **STANCE_new_study.m**

USAGE:

```
[Nbr_sss, Now_sss] = STANCE_new_study(new_study_number, OKflag);
```

DESCRIPTION:

Creates file paths for a new study and updates global variables in STANCE.mat. No arguments adds new study folder equal to the next available study number.

INPUTS (default value):

new_study_number = New study of interest.

OKflag (false) = If to by-pass query box about overwriting folder with the 'OK' choice or not.

OUTPUTS:

Nbr_sss = A triple of counting numbers tallying the total number of studies, subject and session folders.

Now_sss = A triple of counting numbers indicating the current active working directory according to its study, subject and session numbers.

- **STANCE_new_subject.m**

USAGE:

```
[Nbr_sss, Now_sss] = STANCE_new_subject(study_number, new_subject_number,
OKflag);
```

DESCRIPTION:

Creates file paths for a new subject in one study and updates global variables in STANCE.mat. No arguments adds new subject folder equal to the next available subject number in current active study folder.

INPUTS (default value):

study_number = The study number of interest. NOTE: can also be This_sss.

new_subject_number = New subject number of interest.

OKflag (false) = If to by-pass query box about overwriting folder with the 'OK' choice or not.

OUTPUTS:

Nbr_sss = A triple of counting numbers tallying the total number of study, subject and session folders.

Now_sss = A triple of counting numbers indicating the current active working directory according to its study, subject and session numbers.

- **STANCE_new_session.m**

USAGE:

```
[Nbr_sss, Now_sss] = STANCE_new_session(study_number, subject_number, new_session_number, OKflag);
```

DESCRIPTION:

Creates file paths for a new subject in one study and updates global variables in STANCE.mat.

INPUTS (default value):

study_number = The study number of interest. NOTE: can also be This_sss.

subject_number = The subject number of interest.

new_session_number = New session number of interest.

OKflag (false) = By-pass query box about overwriting folder with a choice of 'OK'.

OUTPUTS:

Nbr_sss = A triple of counting numbers tallying the total number of study, subject and session folders.

Now_sss = A triple of counting numbers indicating the current active working directory according to its study, subject and session numbers.

- **STANCE_how_many_studies.m**

USAGE:

```
[Nbr_studies, list] = STANCE_how_many_studies;
```

DESCRIPTION:

Counts the total number of studies operated in the simulator.

OUTPUTS:

Nbr_studies = A counting number tallying the total number of study folders.

list = A list of the numerical labels of all of the study folders.

- **STANCE_how_many_subjects.m**

USAGE:

```
Nbr_subjects = STANCE_how_many_subjects(study_number);
```

DESCRIPTION:

Counts the total number of subjects operated in one study.

INPUT:

study_number = The numerical label of the study of interest.

OUTPUTS:

Nbr_studies = A counting number tallying the total number of study folders.

- **STANCE_how_many_sessions.m**

USAGE:

```
Nbr_sessions = STANCE_how_many_sessions(study_number, subject_number);
```

DESCRIPTION:

Counts the total number of sessions operated in one subject of a study.

INPUTS (default):

study_number (`Now_sss(1)`) = The study of interest.

subject_number (`Now_sss(2)`) = The subject of interest.

OUTPUT:

Nbr_studies = A counting number tallying the total number of study folders.

- **STANCE_list_subjects.m**

USAGE:

```
subject_list = STANCE_list_subjects(study_number);
```

DESCRIPTION:

Lists the number of subjects operated in one study.

INPUT (default):

study_number (`Now_sss(1)`) = The study of interest.

OUTPUT:

subject_list = A list of the numerical labels of all of the subject folders in the study folder of interest.

- **STANCE_list_sessions.m**

USAGE:

```
session_list = STANCE_list_sessions(study_number, subject_number);
```

DESCRIPTION:

Lists the number of sessions operated in one subject of a study.

INPUTS (default):

study_number (`Now_sss(1)`) = The study of interest.

subject_number (`Now_sss(2)`) = The subject of interest.

OUTPUT:

session _list = A list of the numerical labels of all of the subject folders in the subject subfolder of the study folder of interest.

- **STANCE_change_sss.m**

USAGE:

```
Now_sss = STANCE_change_sss(session_number, subject_number,
study_number);
```

DESCRIPTION:

Switch to another study, subject and session folder from the current one.

INPUTS:

session_number (`Now_sss(1)`) = The session of interest.

subject_number (`Now_sss(2)`) = The subject of interest.

study_number (`Now_sss(3) + 1`) = The study of interest.

OUTPUT:

Now_sss = A triple of counting numbers indicating the current active working directory according to its study, subject and session numbers.

- **STANCE_genpath.m**

USAGE:

```
filepath = STANCE_genpath(This_sss, level);
```

DESCRIPTION:

Generates a file path at a specific level based upon the currently selected study, subject and session.

INPUTS (default):

This_sss (Now_sss) = A triple of counting numbers indicating the current active working directory according to its study, subject and session numbers.

level (3) = The working level of path would be generated: 1 = study, 2 = subject, 3 = session.

OUTPUT:

filepath = The full file path of the directory of interest.

- **prep_subject_files.m**

USAGE:

```
M_array = prep_subject_files;
```

DESCRIPTION:

Prepares BrainWeb subject resource files (T1/fuzzy) to for general use in STANCE by unzipping them and registering the T1-w MNI152 normalized brain to each of them.

OUTPUT:

M_array = 4 x4 affine transformation matrices describing the linear registration from the T1-w MNI152 average brain to the 20 subject brains.

2.4 Simulation of 3D volume tools

In order to, in principle, simulate any kind of fMRI data, including group studies, the T1-w MNI152 normalized brain is first registered to each subject brain. This step is necessary in order to take *any* 3D map defined in or already registered to MNI coordinates and be able to transform it onto the subject's native space. In this way any quantity defined "universally" in MNI space can be transformed to any of the subject spaces at will. Next, in order to simulate the fMRI scan itself, the subject's fuzzy membership volumes for 12 tissue types are resliced according to the fMRI scan parameters to a functional space, which is laid out with a user-specified voxel size, slice gap, inter-voxel spacing, and desired pitch tilt angle relative to the AC-PC line. These resliced tissue maps, plus, e.g. the average PD, T1 and T2* values for each tissue, are used to construct the PD, T1 and T2* baseline volumes for the subject's functional space. If one's aim is to simulate neuronal activation in a general linear model (GLM) sense, with resulting statistical parameter maps of assumed activation regions (blobs), then this is achievable by taking an activation map somehow defined in MNI152 space, transforming it to the subject's native space, reslicing to the fMRI scan's functional space, masking it with the subjects gray matter priors, and finally adding the result to the T2* baseline. Such 3D maps are manipulated and generated by the 3D simulation tools. These same tools are useful for ROI manipulation for other kind of studies.

- **STANCE_conform.m**

USAGE:

```
[V_out, Y_out] = STANCE_conform(P_in, P_ref, file_out, description, gzipFlag);
```

DESCRIPTION:

Conforms input map P_in to the dimensions with same origin and orientation of reference P_ref.

INPUTS:

P_in = Map to be conformed, as specified by either filename or SPM header.

P_ref = Reference map, as specified by either filename or SPM header.

file_out = Output directory and file name to be saved.

description ([]) = Descriptive string to be added to the header of the conformed file generated.

gzipflag (false) = Logical flag for keeping only archived gzipped files (lower storage cost) or allowing for unzipped files to remain (faster run times).

OUPUTS:

V_out = The SPM header structure containing info about the conformed file.
Y_out = The 3D/4D data contained in the conformed file.

- **STANCE_register_MNI.m**

USAGE:

```
[V_MNI_reg, Y_MNI_reg] = STANCE_register_MNI(filename, M, flags, This_sss);
```

DESCRIPTION:

Registers and transforms the 1mm^3 isotropic MNI152 T1-w brain specified in STANCE.m by filenameMNI to the 1mm^3 isotropic T1-w brain image of interest saved in filename, after setting its the voxel-to-world mapping according to M.

INPUTS (default):

filename = File name of the 1mm^3 isotropic T1-w brain image of interest to be registered to.

M = 4x4 affine transformation matrix for mapping from one brain to another, used to set the voxel-to-world mapping of the registered MNI152 brain using spm_get_space.m.

flags = The method used to realign the image flags, in the form of a structure containing various options. For details about the fields see the documentation for spm_reslice.m. Defaults: mask = 1, mean = 0, interp = 5, which = 1, wrap = [1 1 0].

This_sss (Now_sss) = A triple of counting numbers indicating the current active working directory according to its study, subject and session numbers.

OUPUTS:

V_MNI_reg = The SPM header structure containing info about the registered MNI brain volume with the same filename as the source MNI152 file except with an 'r' prefix and saved in the subject of interest subfolder for the study of interest.

Y_MNI_reg = The 3D T1-w image data of the registered MNI152 brain volume.

- **STANCE_register_activation.m**

USAGE:

```
[V_act_reg, Y_act_reg] = STANCE_register_activation(filename, task, flags,
This_sss, ADO);
```

DESCRIPTION:

Registers task.map to filename's volume and saves in the subject of interest folder as a NIFTI file named ‘r’+task.name.

INPUTS:

filename = Name of 1mm^3 isotropic T1-w brain image of interest to be registered to.

task = Structure array containing fields ‘map’ (the 3D volume describing the task-related activation region in MNI space) and ‘name’ (the name of the task/stimulus).

flags = The method used to realign the image flags, in the form of a structure containing various options. For details about the fields see the documentation for spm_reslice.m. Defaults: mask = 1, mean = 0, interp = 5, which = 1, wrap = [1 1 0].

This_sss (Now_sss) = A triple of counting numbers indicating the current active working directory according to its study, subject and session numbers.

ADO = Anatomical display option in the GUI (not included with initial release).

OUPUTS:

V_act_reg = The SPM header structure containing info about the registered task.map with the filename as [‘r’,task.name,’.nii’] saved in the subject of interest subfolder for the study of interest.

Y_act_reg = The 3D data of the registered task.map.

- **STANCE_reslice_volume.m**

USAGE:

```
[V_new, Y_new] = STANCE_reslice_volume(V_old, scan, sumThreshold);
```

DESCRIPTION:

Generates a resliced volume V_new of V_old according to the fMRI scan protocol that specifies the functional space.

INPUTS (default):

V_old = Source volume in anatomical space.

scan (default) = Scan protocol parameters used in a study.

sumThreshold (100) = If a number this quantity sets the requirement for a slice to be included in terms of the required sum of voxel intensities for the limiting slices; if an array it specifies the slice limits in the form of [sliceLimitLower, sliceLimitUpper].

OUPUTS:

V_new = The SPM header structure containing info about V_old resliced to functional space, with the same name except for including a '_resliced' suffix.

Y_new = The 3D data of V_old resliced to functional space.

- **STANCE_ reslice_tissue.m**

USAGE:

```
[V_Tissue_Labels_Resliced, Y_Tissue_Labels_Resliced] = STANCE_reslice_tissue(fileName, scan, sumThreshold, gzipFlag, showFlag);
```

DESCRIPTION:

Generates the tissue fuzzy memberships in functional space that is specified according to the fMRI scan protocol.

INPUTS (default):

fileName = Tissue fuzzy membership.

scan (default) = Scan protocol parameters used in a study.

sumThreshold (100) = If a number this quantity sets the requirement for a slice to be included in terms of the required sum of voxel intensities for the limiting slices; if an array it specifies the slice limits in the form of [sliceLimitLower, sliceLimitUpper].

gzipflag (false) = Logical flag for keeping only archived gzipped files (lower storage cost) or allowing for unzipped files to remain (faster run times).

showFlag (false) = Logical flag for displaying figures showing progress or not.

OUPUTS:

V_new = The 12 member cell containing SPM header structures containing info about fileName resliced to functional space, with the same name except for including a '_resliced' suffix.

Y_new = The 3D data of fileName resliced to functional space.

- **STANCE_make_activation_map.m**

USAGE:

```
activation_map = STANCE_make_activation_map(dimensions, origin, activation, method);
```

DESCRIPTION:

Generates activation region for specified dimensions.

INPUTS (default):

dimensions = Dimensions of the image matrix (can be 2D or 3D).

origin ([0, 0]/[0,0,0]) = Stereotactic origin of the space.

activation = The information structure array of the activation map to be constructed . See Section 4 below for details about its fields/

method ('linear') = Interpolation method used by MATLAB function for the activation map.

OUPUT:

activation_map = The 2D slice/3D map of spatially varying normalized activation levels (0.0-1.0).

CREDIT:

This function was initially inspired by neuRosim's implementation specifyregion() R function (Welvaert *et al.*, 2011) however our version constructs maps more efficiently (in an algorithmic sense) and offer more functionality.

- **STANCE_combine_maps.m**

USAGE:

```
combo_map = STANCE_combine_maps(operation, A, varargin);
```

DESCRIPTION:

Applies fuzzy logical operation on maps.

INPUTS:

operation = Logical operation be applied: 'NOT', 'OR', 'AND', 'XOR', or 'NAND'.

A = Base map for operation.

varargin = Other maps to be combined with A.

OUPUT:

combo_map = The normalized 2D slice/3D map generated as a combination of maps A, etc. … according to the specified fuzzy logical operation.

- **STANCE_parse_combine.m**

USAGE:

```
map = STANCE_parse_combine(task);
```

DESCRIPTION:

Combines the maps specified in the task.map cell according to the operations and options provided in the task.combine cell.

INPUT:

task = Structure array containing fields 'map' (the 3D volume describing the task-related activation region in MNI space) and 'combine' (the cell of combinations).

OUPUT:

map = The 2D slice/3D map of spatially varying normalized activation levels (0.0-1.0).

- **STANCE_display_activation_slice.m**

USAGE:

```
h = STANCE_display_activation_slice(Y_brain, Y_activation, slice, direction, origin, GUIflag);
```

DESCRIPTION:

Displays slice in dimension of activation map (hot color screen) overlaid on brain (grayscale) with coordinate and directional annotations added to the figure.

INPUTS (default):

Y_brain = 3D brain image used as the background (typically a T1-w or T2*-w image).

Y_activation = Synthetic activation map to overlay over brain.

slice (middle slice) = Slice number of activation map to be displayed.

direction (3) = Slice direction in activation map (sagittal = 1, coronal =2, axial = 3, all = []).

origin ([0 0 0]) = Origin offset between matrix and MNI space.

GUIflag (false) = Control flag for display in GUI (not included with initial release).

OUPUT:

h = The figure handle for the resulting displayed image shown.

- **STANCE_find_GM_volume.m**

USAGE:

```
GM_volume = STANCE_find_GM_volume(task, ADO);
```

DESCRIPTION:

Computes the numerical amount of gray matter volume for the activation map based on MNI tissue priors, i.e. it is simply the sum of the gray matter priors in the 3D map divided by the equivalent sum of the MNI152 brain's gray matter priors as provided by SPM.

INPUTS:

task = Structure array containing the field 'map' (the 3D map describing the task/stimulus-related activation region in MNI space).

ADO = Anatomical display option in the GUI (not included with initial release).

- **STANCE_GM_mask.m**

USAGE:

```
[activation_map, Y_GM] = STANCE_GM_mask (activation_map, actMNIgmVolume, This_sss, morphLimit);
```

DESCRIPTION:

Masks the synthetic activation map with the gray matter tissue priors of the subject, while attempting to achieve the target gray matter volume (sum of activated voxels) with the morphological operations of dilation and erosion.

INPUTS (defaults):

activation_map = 3D map of spatially varying normalized activation levels (0.0-1.0).

actMNIgmVolume = The amount of gray matter volume of the activation map in MNI space.

This_sss(Now_sss) = A triple of counting numbers indicating the current active working directory according to its study, subject and session numbers.

morphLimit (10) = Limits of number of successive morphological operations applied to reach the target gray matter volume number.

OUPUT:

activation_map = The 2D slice/3D map of spatially varying normalized activation levels (0.0-1.0).

Y_GM = The gray matter prior mask of the subject of interest in functional space.

- **STANCE_add_activation.m**

USAGE:

```
[V_act, Y_act] = STANCE_add_activation(filename_baseline, activation_map, TE, amplitude);
```

DESCRIPTION:

Adds the activation map onto the T2* baseline map of the subject to yield the peak T2* BOLD signal map for that neuronal activation Y_act for use in solving EPI signal.

INPUTS:

filename_baseline = The file name of T2* baseline map generated from tissue fuzzy memberships and resliced to functional space.

activation_map = Synthetic activation map of interest resliced and masked to the subject's functional space; normalized (0-1.0).

TE (30) = EPI pulse sequence echo time [ms].

amplitude (0.02) = Target fraction of a signal change in the resulting EPI data due to activation (can be fitted to real experimental data).

OUPUTS:

V_act = The SPM header structure containing info about the T2* with BOLD signal added file saved as filename_baseline with the suffix '_BOLD' added.

Y_act = The 3D/4D data of the T2* with spatially varying BOLD signal added map.

- **STANCE_find_affine_array.m**

USAGE:

```
M_array = STANCE_find_affine_array;
```

DESCRIPTION:

Calculates the affine transformation matrices for transforming maps in MNI space to each of the 20 brain phantom's native spaces.

OUTPUT:

M_array = 4 x4x20 array of the 20 affine transformation matrices describing the linear registration from the T1-w MNI152 average brain to the 20 subject brains.

2.5 Simulation of 4D data with time series tools

In order to simulate the fMRI 4D time series, the experimental design time-series is required for all tasks in a study. This is the idealized stimulus/response time-series for stimulus/task-evoked neuronal activation, since in practice no one is "on task" 100% of the time. A hemodynamic response

is applied to the task-evoked stimulus time-series is to yield the expected BOLD response time-series. Typically, this is accomplished by convolving the task-evoked stimulus time-series with with a hemodynamic response function (HRF). Physiological noise from simulated cardiac and respiratory sources is also simulated. The HRF and physiological noise parameters can vary spatially across the brain and subjects. It is also possible to add spatially varying lag times to the physiological noise time series. Slice timing is also considered. The generation and simulation of all of the above mentioned time series and are implemented in 4D simulation tools.

- **STANCE_design_timeseries.m**

USAGE:

```
ts = STANCE_design_timeseries(dt, event_ON_blocks, event_OFF_blocks, startFlag);
```

DESCRIPTION:

Generates an experimental design timeseries object for the specified sample time and event ON and OFF time interval blocks, given a starting preference (i.e. start with an OFF or ON block).

INPUTS (default):

dt	= The sample time of the design [s].
event_ON_blocks	= The 1D array of time intervals for events: double [s]/integer [# of dt].
event_OFF_blocks	= The 1D array of time intervals between events: double [s]/integer [# of dt].
startFlag (true)	= Options to start design with OFF = true or ON = false block.

OUTPUT:

ts	= General experimental design MATLAB timeseries object.

- **STANCE_blocked_design.m**

USAGE:

```
ts = STANCE_blocked_design(dt, start_lag, block, ISI, T_scan, startFlag);
```

DESCRIPTION:

Design a block time series with specified sample time, block duration and intervals.

INPUTS:

dt = The sample time of the design [s].

start_lag = The start time delay of each experiment: double [s]/integer [# of dt].

block = The time interval of an event due to a single condition : double [s]/integer [# of dt].

ISI = The inter-stimulus interval : double [s]/integer [# of dt].

T_scan = The duration of the experiment; type = double[s]/integer[# of dt].

startFlag (true) = Options to start design with OFF = true or ON = false block.

OUTPUT:

ts = Blocked experimental design MATLAB timeseries object.

- **STANCE_event_design.m**

USAGE:

```
ts = STANCE_event_design(dt, event_onset, event_duration, T_scan);
```

DESCRIPTION:

Generates an event-related design timeseries object for a single condition with the specified sample time, event onsets and durations.

INPUTS:

dt = The sample time of the design [s].

event_onset = A 1D array of the time onsets of events due to a single condition: double [s]/integer [# of dt].

event_duration = A 1D array of the time duration intervals of events due to a single condition: double [s]/integer [# of dt].

T_scan = The total time of the scan: double [s]/integer [# of dt].

OUTPUT:

ts = Event-related experimental design MATLAB timeseries object.

- **STANCE_apply_response_function.m**

USAGE:

```
OUT_ts = STANCE_apply_response_function(dt, ts, response_function, param, sliceOrder);
```

DESCRIPTION:

Generates a resulting time series by applying a response function to the timeseries object ts. All response functions are convolutional except for the balloon model of hemodynamic response. The cardiac and respiratory response functions can also be called with this tool.

INPUTS (default):

dt	= Sampling time [s].
ts	= Input timeseries that governs/modulates the response.
response_function ('canonical')	= Name of the response function of interest: 'canonical', 'double', 'gamma', 'triple', 'logit', 'balloon', 'cardiac', 'respiratory'
param ([])	= List/structure of parameters of the response function of interest. NOTE: spatial variation is added by replacing a scalar parameter with a structure array with fields 'max' contain the maximum value and 'map' with the name of a file that contains the normalized 3D map of the parameter resliced to the sessions functional space.
sliceOrder ('SD')	= Option for specifying the slice order of the fMRI scan.

OUTPUT:

OUT_ts	= The resulting response MATLAB timeseries object.

- **STANCE_ convolve_response_function.m**

USAGE:

```
ts_conv_Data = STANCE_convolve_response_function (dt, ts_Data,
response_function, param, sliceOrder, RF_time);
```

DESCRIPTION:

Generates a resulting timeseries object by convolving the requested response function with the time series data ts_Data. The cardiac and respiratory response functions can also be called with this tool.

INPUTS (default):

dt	= Sampling time [s].
ts_Data	= Input timeseries that governs/modulates the response.
response_function ('canonical')	= Name of the response function of interest: 'canonical', 'double', 'gamma', 'triple', 'logit', 'balloon', 'cardiac', 'respiratory'
param ([])	= List/strutcure of parameters of the response function of interest. NOTE: spatial variation is added by replacing a given scalar parameter with a

structure array with fields 'max' contain the maximum value and 'map' with the name of a file that contains the normalized 3D map of the parameter resliced to the sessions functional space.

sliceOrder ('SD') = Option for specifying the slice order of the fMRI scan.

RF_time = the duration of the response function [s].

OUTPUT:

ts_conv_Data = The resulting response time series.

- **singleGammaHRF.m**

USAGE:

```
out = singleGammaHRF(t, varargin);
```

DESCRIPTION:

Specifies a Gamma variate hemodynamic response function for the given time vector and its parameters.

INPUTS (defaults):

t = Time vector [s].

FWHM (4) = Full Width Half Maximum of the Gamma variate function followed by 'FWHM'. NOTE: this value uniquely determines theta given k.

k (4) = The scaling parameter for phase delay followed by 'k' or 'alpha'.

onset (0) = The onset delay [s] followed by 'onset'.

theta (optional) = The rate parameter that determines the time-scaling followed by 'theta'

beta (optional) = The parameter that determines the dispersion of the response followed by 'beta'. NOTE: beta = 1/theta.

alternatively

param = List/structure of parameters of the hemodynamic response function.

The structure array should contain the following fields; lists in this order (default):

- FWHM – the width of the peak (4)
- k/alpha – the delay of response relative to onset (4)
- onset – the onset delay [s] (0)

- theta – the rate parameter that determines the time-scaling
- beta – the dispersion of response, beta = 1/theta

OUTPUT:

out = The result of convolving t with the response time vector/4D data.

EQUATION:

$$h_{gamma}(t-o;k,\theta)=\frac{(t-o)^{(k-1)}\ e^{-(t-o)/\theta}}{\theta^{k}\Gamma(k)}\ .$$

where o is the onset delay and Γ is the Gamma function; originally used in (Buxton *et al.* 2004).

CREDIT:

This function was initially inspired by neuRosim's implementation gammaHRF() R function (Welvaert *et al.*, 2011) however our version offers more functionality in terms of having more flexibility with the parameters and in the FWHM calculation.

- **doubleGammaHRF.m**

USAGE:

```
out = doubleGammaHRF(t, varargin);
```

DESCRIPTION:

Specifies a double-gamma variate hemodynamic response function for the given time vector and its parameters.

INPUTS:

t = Time vector [s].

param = List/structure of parameters of the hemodynamic response function.

The structure should contain the following fields; lists in this order (default = canonical):

- alpha1 – the delay of response relative to onset (6)
- alpha2 – the delay of undershoot relative to onset (16)
- beta1 – the dispersion of the initial response (1.0)
- beta2 – the dispersion of the subsequent undershoot (1.0)
- c – the relative scale of the undershoot to the initial peak (1/6)

OUTPUT:

out = The result of convolving t with the response time vector/4D data.

EQUATION:

$$h_{double}(t) = \frac{t^{\alpha_1 - 1}\beta_1^{\alpha_1} e^{-\beta_1 t}}{\Gamma(\alpha_1)} - c\frac{t^{\alpha_2 - 1}\beta_2^{\alpha_2} e^{-\beta_2 t}}{\Gamma(\alpha_2)},$$

which was originally proposed in (Friston *et al.*, 1998) with the original default values of the canonical HRF listed in (Glover, 1999).). It is also known that the HRF varies across a subject's brain and between subjects and certain populations (Handwerker *et al.*, 2012; Wu and Marinazzo, 2017) so the more general double gamma form is useful to model such HRF variability.

CREDIT:

This function was initially inspired by neuRosim's implementation canonicalHRF() R function (Welvaert *et al.*, 2011) but our implementation is more general.

- **tripleGammaHRF.m**

USAGE:

```
out = tripleGammaHRF(t, varargin);
```

DESCRIPTION:

Specifies a triple-gamma variate hemodynamic response function for the given time vector and parameters.

INPUTS:

t = Time vector [s].

param = List/structure of parameters of the hemodynamic response function.

The structure array should contain the following fields; lists in this order (default):

- A1 – the amplitude of the 1st Gamma distribution (0.5)
- A2 – the amplitude of the 2nd Gamma distribution (6)
- A2 – the amplitude of the 3rd Gamma distribution (1)
- alpha1 – the delay of the 1st Gamma distribution peak (1.5)
- alpha2 – the delay of the 2nd Gamma distribution peak (7)
- alpha3 – the delay of the 3rd Gamma distribution peak (16)
- beta1 – the dispersion of the 1st Gamma distribution peak (0.8)
- beta2 – the dispersion of the 2nd Gamma distribution peak (1.0)
- beta3 – the dispersion of the 3rd Gamma distribution peak (1.0)

OUTPUT:

out = The result of convolving t with the response time vector/4D data.

EQUATION:

$$h_{triple}(t) = \sum_{i=1}^{3} A_i \frac{t^{\alpha_i - 1} \beta_i^{\alpha_i} e^{-\beta_i t}}{\Gamma(\alpha_i)},$$

described in (Shan *et al.*, 2014).

- **tripleLogitHRF.m**

USAGE:

```
out = tripleLogitHRF(t, varargin);
```

DESCRIPTION:

Specifies a triple-logit variate hemodynamic response function for the given time vector and parameters.

INPUTS:

t = Time vector [s].

param = List/structure of parameters of the hemodynamic response function.

The structure should contain the following fields; lists in this order (default):

- A1 – the amplitude of the 1st logit function (1)
- A2 – the amplitude of the 2nd logit function (1)
- A2 – the amplitude of the 3rd logit function (1)
- T1 – the delay of the 1st logit function (4)
- T2 – the delay of the 2nd logit function (5)
- T3 – the delay of the 3rd logit function (10)
- D1 – the rate parameter of the 1st logit function (1.0)
- D2 – the rate parameter of the 2nd logit function (1.5)
- D3 – the rate parameter of the 3rd logit function (2.0)

OUTPUT:

out = The result of convolving t with the response time vector/4D data.

EQUATION:

$$h_{\mathrm{logit}}(t) = \sum_{i=1}^{3} \frac{A_i}{1 + e^{(t-T_i)/D_i}}.$$

which is described and used by (Logothetis *et al.*, 2001; Handwerker *et al.*, 2004; Lindquist *et al.*, 2007).

- **balloon.m**

USAGE:

```
bold = balloon(ts_Data, dt, T_scan,'param', param);
```

DESCRIPTION:

Simulates the balloon model (Buxton *et al.* 2004) of the hemodynamic response of capillaries in the brain.

INPUTS:

ts_Data = Input timeseries that governs/modulates the response.

dt = Sampling time [s].

T_scan = Total time of the scan [s].

param = structure array of parameters of the balloon model.

The structure array should contain the following fields (default):

- kappa – the inhibitory gain factor (2)
- tau1 – the inhibitory time constant (3)
- tauf – FWHM of the CBF impulse response (4)
- taum – the FWHM of the $CMRO_2$ impulse response (4)
- f1 – the normalized CBF response to sustained neural activation (1.5)
- deltat – the delay of the CBF relative to the CMRO2 response (1)
- n – the steady-state flow metabolism relation (3)
- E0 – the rate parameter of the 2nd logit function (0.4)
- V0 – the rate parameter of the 3rd logit function (0.03)
- a1 – the weight for deoxyHb change (3.4)
- a2 – the weight for blood volume change (1.0)
- tauMIT – the transit time through the balloon (3)
- tau – the viscoelastic time constant (20)
- alpha – the steady-state flow-volume relation (0.4)

OUTPUT:

bold = The simulated bold signal based on the balloon model.

SOURCE:

This is a MATLAB translation of neuRosim's balloon() R function (Welvaert *et al.*, 2011).

- **make_slice_timing.m**

USAGE:

```
sliceTiming = make_slice_timing(sliceOrder,N_slices,acceleration);
```

DESCRIPTION:

Generates the slice timing index vector for an MRI scan given its slice order and number of slices.

INPUTS (default):

sliceOrder = String for specifying the slice order by the prefixes and suffixes:

'S_' - Sequential ordering

'I_' - Interleaved ordering

'_A' - Ascending ordering

'_D' - Descending ordering

'__2' - start with even numbers instead of odd numbers (default)

N_slices = Number of slices to be simulated.

acceleration (1) = The multi-band acceleration factor (acquiring more than 1 slice at a time).

OUTPUT:

sliceTiming = Vector that lists slice indeces in the order that they will be acquired every TR.

- **make_impulse_timeseries.m**

USAGE:

```
ts = make_impulse_timeseries(dt,Nt,TI,TI_sigma);
```

DESCRIPTION:

Generates an impulse event time-series characterized by interval time with its standard deviation.

INPUTS:

dt = Sampling time [s].

Nt = Total number of time points.

TI = Average interval time between events [s].

TI_sigma = Standard deviation of interval time [s].

OUTPUT:

ts = Impulse MATLAB timeseries object with Gaussian power spectrum.

- **make_pulse_timeseries.m**

USAGE:

```
z_ts = make_pulse_timeseries(dt, Nt, TI, TI_sigma, power, A_sigma, SighTI,
SighTI_sigma, impulse_seed, dz_seed);
```

DESCRIPTION:

Generates a timeseries object of pulsation events characterized by interval time TI between pulses with standard deviation TI_sigma normal relaxed position is baseline z = 0. Shape of pulses are half cosines raised to the p-power Output is normalized to average amplitude.

Defaults assume a typical respiratory pulse.

INPUTS (default):

dt = Sampling time [s].

Nt = Total number of time points.

TI (4) = Average time interval [s].

TI_sigma (0.25) = Standard deviation of the time interval [s].

p (4) = Exponent power of the half cosine function that describes the chest motion.

A_sigma (0.1) = Standard deviation of the pulsation height [fraction of average height].

SighTI (400) = Average time interval between sighs [s] (to be implemented in the future).

SighTI_sigma (15) = Standard deviation of the time interval between sighs [s] (to be implemented in the future).

impulse_seed = Random number generator seed for impulse amplitude simulation.

dz_seed = Random number generator seed for pulse height simulation (reproducibility).

OUTPUT:

z_ts = Chest motion MATLAB timeseries object.

EQUATION:

A general pulsation is modeled for the i^{th} cycle as

$$d(\varphi_i) = A_i \cos^p(\pi \varphi_i),$$

where the average height of the pulse is normalized to 1, $\varphi \in$ [-0.5,+0.5] parametrizes the time/TI, relative to the i^{th} impulse and the default $p = 4$ (Lujan et al, 2003).

- **make_event_timeseries.m**

USAGE:

```
[ts,Nk] = make_event_timeseries(ms,TI);
```

DESCRIPTION:

Generates an impulse event timeseries object characterized by the modulation function time-series and interval time according to the Intergral Pulse Frequency Modulation (IPFM) model (Bayly, 1968).

INPUTS:

ms	= Modulation function timeseries object for generating events.
TI	= Average interval time between events [s].

OUTPUTS:

ts	= Impulse MATLAB timeseries object with power spectrum governed by ms.
Nk	= Number of impulses generated.

EQUATIONS:

The IPFM model (Bayly, 1968) is modulated by the function

$$m(t) = \sum_n a_n f_n(\omega_n t + \phi_n),$$

where ω_n and ϕ_n are the component frequency and phase shift for the n^{th} component, respectively, and f is a quasi-periodic function, taken to be a cosine in the traditional model. The instantaneous heart rate is then given by $(1 + m(t))/TI$, where TI is the average time interval between beats. Therefore the next event takes place whenever the integral sum of $(1 + m(t))/TI$ reaches 1.

2.6 Noise tools

Various forms of noise can be modeled and linearly added to the signal according to the accepted total image noise model of the different noise components found in the literature. STANCE is equipped with the functions to simulate spatial variation of system noise, physiological noise (from respiratory and cardiac sources), drift, autocorrelated noise, etc.

- **STANCE_make_noise_map.m**

USAGE:

```
noiseMap = STANCE_make_noise_map(fn_tissues, dilation, factor, tissues,
This_sss);
```

DESCRIPTION:

Create a noise amplitude map based on tissue priors.

INPUTS (default):

fn_tissues	= File name of the tissue priors for the target functional space.
dilation	= Morphological dilation parameters for noise map.
factor (2.0)	= Amplitude factor of noise map, quantifying how much higher the noise is in the CSF versus the GM.
tissues	= 1D array of tissue priors of interest according to their indeces.
This_sss (Now_sss)	= A triple of counting numbers indicating the current active working directory according to its study, subject and session numbers.

OUPUT:

noiseMap	= The 3D map of spatially varying noise levels with 1.0 as the minimum value.

- **STANCE_estimate_RTI.m**

USAGE:

```
[RTI_ave, RTI_sigma] = STANCE_estimate_RTI(y, TR);
```

DESCRIPTION:

Estimates the average respiratory time interval (RTI) and its standard deviation from the spectral features of a motion time series (usually in the y-direction).

INPUTS:

y	= A motion time series derived from real data taken at TR.
TR	= The repetition time [s].

OUTPUTS:

RTI_ave	= The average RTI detected impinged on the motion time series y [s].
RTI_std	= Standard deviation of the RTI detected impinged on the motion time series y [s].

- **STANCE_physio_4D.m**

USAGE:

```
[physio_4D, RP_ts, CP_ts]
 = STANCE_physio_4D(fn_tissues, Nt, scan, physio, veto_factor);
```

DESCRIPTION:

Generates a physiological noise times-series with the main sources of respiratory and cardiac noise.

INPUTS:

fn_tissues = File name of the tissue priors for the target functional space.

Nt = Total number of time points.

scan = Structure array detailing the fMRI scan protocol.

physio = Structure detailing and controlling the construction of physiological noise .

veto_factor = Number of max lambda standard deviations to suppress spiking.

OUTPUT:

physio_4D = 4D matrix of the simulated physiological noise in functional space sampled every TR according to the sliceTiming.

RP_ts = The respiratory pulse MATLAB timeseries object.

CP_ts = The cardiac pulse MATLAB timeseries object.

- **STANCE_RVT.m**

USAGE:

```
[RVT,FRC,TIV] = STANCE_RVT(dt, Nt, RTI, RTI_sigma, weight, A_sigma, tSigh,
tSigh_sigma, impulse_seed, dz_seed);
```

DESCRIPTION:

Simulates a respiratory volume timeseries object characterized by the average respiratory time and its standard deviation for a person of a specific weight. Default values from (Barrett *et al.*, 2012)

INPUTS:

dt = Sampling time [s].

Nt = Total number of time points.

RTI (4) = Average respiratory time interval [s].

RTI_sigma (0.25) = Standard deviation of the respiratory time interval [s].

weight (75) = Average weight of subject for lung volume calculation [kg].

A_sigma (0.1) = Standard deviation of the chest motion height [cm].

SighTI (400) = Average time interval between sighs [s] (to be implemented in the future).

SighTI_sigma (15) = Standard deviation of the time interval between sighs [s] (to be implemented in the future).

impulse_seed = Random number generator seed for respiratory impulse simulation .

dz_seed = Random number generator seed for chest motion simulation (reproducibility).

OUTPUTS:

RVT = Respiratory volume MATLAB timeseries object [mL].
FRC = Functional Residual Capacity (FRC), the lung volume left after a normal exhalation [mL].
TIV = Typical Inspiratory Volume (TIV), the lung volume after normal inhalation. possible [mL].

EQUATION:

From the chest motion time series $\Delta r(t) = 0.58 \cdot d(t)$, an ellipsoid model of the lungs is used to yield the respiratory lung volume:

$$RVT(t) = \frac{4\pi W}{300\,kg}[11\,cm + \Delta r(t)][11\,cm + \Delta r(t)][6\,cm + \Delta r(t)]$$

where W is the weight in kg, which yields typical lung exhalation and inhalation volumes according to established norms (Delawari & Doelman, 2010; Jones, R. L. and Nzekwu, 2006).

- **respiratoryRF.m**

USAGE:

```
out = respiratoryRF(t, varargin);
```

DESCRIPTION:

Specifies a double-gamma variate respiratory response function for given respiratory volume time-series (RVT), time vector and parameters.

INPUTS:

t = Time vector [s].

DESCRIPTION:

Specifies a double-gamma variate hemodynamic response function for the given time vector and its parameters.

INPUTS:

t = Time vector [s].

param = List/structure of parameters of the hemodynamic response function.

The structure array should contain the following fields; lists in this order:

- a1 – the delay of response relative to onset (2.1)
- a2 – the delay of undershoot relative to onset (3.54)
- b1 – the dispersion of the initial response (1.6)
- b2 – the dispersion of the subsequent undershoot (4.25)
- c1 – the relative scale of the initial peak (0.6)
- c2 – the relative scale of the undershoot to the initial peak (0.0023)

OUTPUT:

out = The result of convolving t with the response time vector/4D data.

EQUATION:

$$RRF(t) = c_1 \frac{t^{a_1-1} b_1^{a_1} e^{-b_1 t}}{\Gamma(a_1)} - c_2 \frac{t^{a_2-1} b_2^{a_2} e^{-b_2 t}}{\Gamma(a_2)}.$$

Originally proposed in (Birn *et al.* 2008), extended by (Cordes *et al.*, 2014).

- **STANCE_PWV_timeseries.m**

USAGE:

```
v_ts =
STANCE_PWV_timeseries(tC_ts,PW_width,PW_width_sigma,v_0,v_constant);
```

DESCRIPTION:

Simulates a pulse wave velocity timeseries object characterized by cardiac impulse time-series with pulse wave width and average blood flow.

INPUTS:

tC_ts = Cardiac impulse time series.

PW_width (0.5) = Width of pulse wave [s].

PW_width_sigma (0.02) = Standard deviation of pulse wave width of interest [s].

v_0 (3.0) = Average blood flow velocity over time [mm/s].

v_constant_fraction (0.1) = Constant blood flow velocity, i.e. the minimum velocity the bloodis always going as a fraction of the peak blood velocity.

OUTPUT:

v_ts = Pulse velocity MATLAB timeseries object [mm/s].

EQUATION:

The pulse wave velocity for the i^{th} heart cycle with average velocity v_0 follows

$$v(\varphi_i) = cv_0 + V_i \cos(\pi \varphi_i + \psi),$$

where the average height $V = 2\ v_0*(1\text{-}c)$, with c the v_constant_factor; $\varphi \in$ [-0.5,+0.5] parametrizes the time/TI, relative to the i^{th} beat, and ψ is a phase offset that can spatially vary (Zambanini *et al.*, 2005; Arimon, 2006). STANCE models the average blood velocity in a voxel as $v_0 = 3 + 200\rho_{BV}$ mm/s based on known velocities (Rengachary & Ellenbogen, 2005; Belliveau *et al.*, 1991), where ρ_{BV} is the blood vessel tissue prior.

- **cardiacRF.m**

USAGE:

```
out = cardiacRF(t, varargin);
```

DESCRIPTION:

Specifies a double-gamma variate cardiac response function for given heart rate, time vector and parameters.

INPUTS:

t = Time vector [s].

param = List/structure of parameters of the hemodynamic response function.

The structure array should contain the following fields; lists in this order:

- a1 – the delay of response relative to onset (2.7)
- b1 – the dispersion of the initial response (1.6)
- b2 – the dispersion of the subsequent undershoot (9.0)
- c1 – the relative scale of the initial peak (0.6)
- c2 – the relative scale of the undershoot to the initial peak (16)
- d2 – the offset of the undershoot (12)
- K – the constant of the derivative (0)

OUTPUT:

out = The result of convolving t with the response time vector/4D data.

EQUATION:

$$CRF(t) = CRF_0(t) + K\frac{d}{dt}CRF_0(t),$$

$$CRF_0(t) = c_1 t^{a_1} e^{-t/b_1} - \frac{c_2}{\sqrt{2\pi\, b_2}}\, e^{-\frac{b_2}{2}(t-d_2)^2}$$

Originally proposed in (Chang *et al.* 2009), extended by (Cordes *et al.*, 2014).

- **make_drift_timeseries.m**

USAGE:

```
ts = make_drift_timeseries(dt,Nt,period,random_phase,order,seed);
```

DESCRIPTION:

Generates a zero-mean, unit-variance drift time-series characterized by interval time, oscillatory period (Smith *et al.*, 1999) and order of polynomial.

INPUTS:

dt	= Sampling time interval [s].
Nt	= Number of time points.
period	= The characteristic period of drift oscillations [s].
random_phase	= Boolean flag to indicate whether to generate with random phases or not.
order	= The order of a random polynomial trend to add to drift model.
seed	= Random number generator seed for drift simulation (reproducibility).

OUTPUT:

ts	= Drift MATLAB timeseries object with very low frequency drift.

SOURCE:

This is a MATLAB translation of neuRosim's lowfreqdrift() R function (Welvaert *et al.*, 2011).

- **make_ARMA_timeseries.m**

USAGE:

```
ts = make_ARMA_timeseries(dt,Nt,sigma,p,q,constant,pLags);
```

DESCRIPTION:

Generates an autoregressive moving average model of time correlations to a time-series arising from the repetitive sampling time (Purdon & Weisskoff, 1998).

INPUTS:

dt	= Sampling time [s].
Nt	= Number of time points.
sigma	= Standard deviation of the signal.
p	= Cell vector of non-seasonal autoregressive coefficients corresponding to a stable polynomial (the AR coefficients).
q	= Cell vector of non-seasonal moving average coefficients corresponding to an invertible polynomial (the MA coefficients).
constant	= Scalar constant in the linear time series.
pLags	= Vector of positive integer lags associated with the AR coefficients.

OUTPUT:

ts	= ARMA MATLAB timeseries object with autocorrelation.

SOURCE:

This is a MATLAB translation of neuRosim's temporalnoise() R function (Welvaert *et al.*, 2011).

2.7 Simulation of MRI data tools

These tools specifically simulate the MR signals themselves as based on the signal equations, slice timing, etc. Also, physiological noise, motion and system noise are added linearly to the pristine signal *in that order*.

- **STANCE_EPI_signal.m**

USAGE:

```
[V_signal_out,Y_signal_out] =
STANCE_EPI_signal(fn_tissues,T2star,scan,noiseMap,physio4D,motion,ap
proximation,gzipFlag,seed);
```

DESCRIPTION:

Computes the EPI signal sequences from the fMRI scan protocol, pulse sequence, as well as the PD, T1, and BOLD T2* parameter maps in functional space. This function also linearly adds physiological noise, motion and Rician system noise (Gudbjartsson *et al.* 1995) to the pristine signal *in that order*.

INPUTS:

fn_tissues = Filename of tissue priors for target space.

T2star = T2* baseline volume in functional space.

scan = Scan protocol parameters in an experiment.

noiseMap = 3D noise map on tissue priors.

physio4D = Simulated 4D physiological noise times series for the experiment.

motion = Motion parameters for each scan.

approximation = Flag for enabling computation approximation (less accurate but a bit faster).

gzipflag (false) = Logical flag for keeping only archived gzipped files (lower storage cost) or allowing for unzipped files to remain (faster run times).

seed = Random number generator seed for thermal noise simulation (reproducibility).

OUPUTS:

V_signal_out = The SPM header structure containing info about the output signal file.

Y_signal_out = The 3D/4D image data contained in the output signal file.

- **STANCE_make_parameter_map.m**

USAGE:

```
[V_Parameter_Map, Y_Parameter_Map] = STANCE_make_parameter_map(filename, parameter, gzipFlag);
```

DESCRIPTION:

Generates T1, T2, T2*... parameter maps in functional space from the tissue fuzzy membership provided. NOTE: in principle this could be used to expand to multimodal data simulation.

INPUTS:

filename = File name of simulated tissue fuzzy membership.

parameter = Objective medical image parameter map of interest: 'T1', 'T2', or 'T2*' relaxation times, 'PD' (proton density), 'delta' (chemical shift), 'PETatt' (PET attenuation), 'PETact' (PET activity), 'SPECTatt' (SPECT attenuation), or 'SPECTact' (SPECT activity).

gzipflag (false) = Logical flag for keeping only archived gzipped files (lower storage cost) or allowing for unzipped files to remain (faster run times).

OUPUTS:

V_ Parameter_Map = The SPM header structure containing info about the specified file.
Y_ Parameter_Map = The 3D/4D image data contained in the specified file.

- **make_parameter_maps.m**

USAGE:

Run M-script.

DESCRIPTION:

Generates optimized parameter maps derived from McGill's BrainWeb data for 20 normal brain phantoms; saves as gzipped NIFTI files. Used in pre-processing.

2.8 Simulation of correlation tools

STANCE aims also at simulating correlated time series according to a target correlation matrix. Unto this goal a few preliminary tools have been developed.

- **STANCE_make_uncorrelated_noise.m**

USAGE:

```
X = STANCE_make_uncorrelated_noise(nT,nR);
```

DESCRIPTION:

Generates totally uncorrelated normalized white noise X. This forces random time series to be completely uncorrelated.

INPUTS:

nT = Number of time points.

nR = Number of regions/time series.

OUTPUT:

X = An nT x nR array of totally uncorrelated normalized white noise.

- **STANCE_make_correlated_noise.m**

USAGE:

```
Z = STANCE_make_correlated_noise(nT,CM,PSD_in,dt_in,dt_out);
```

DESCRIPTION:

Generates correlated normalizes noise Z with correlation matrix CM and/or PSD (optional).

INPUTS:

nT = Number of time points.

CM = The target square (nR x nR) correlation matrix.

PSD_in ([]) = The target power spectral density (PSD). NOTE: PSD = (abs((fft(X))).^2)/nT;

dt_in ([]) = The sample time of the input data used to calculate PSD_in.

dt_out = The sample time of the output data as specified by the user.

OUTPUT:

Z = An nT x nR array of correlated normalized white noise with correlation ~= CM.

- **find_sliding_correlation.m**

USAGE:

```
rho_wins = find_sliding_correlation(ts, CMref, win, shift);
```

DESCRIPTION:

Computes the Pearson's correlation coefficients for windows of size win moved across timeseries ts by shift; as compared with the reference correlation matrix CMref.

INPUTS:

ts = Input timeseries object .

CMref = The reference square correlation matrix.

win = The times series window.

shift = The window as successively shifted by this amount.

OUTPUT:

rho_wins = Pearson correlation coefficients, rho, as measured across the windowed time series.

- **Pearsons_correlation_coefficient.m**

USAGE:

```
[rho, pvalue] = Pearsons_correlation_coefficient(A, B);
```

DESCRIPTION:

Computes the Pearson's correlation coefficient of the vectorized upper triangles of input matrices A and B.

INPUTS:

A = Input square matrix.

B = Input square matrix,the same size as A.

OUTPUT:

rho = The Pearson's correlation coefficient of A with B.

pvalue = The p-value for testing the NULL hypothesis of no correlation.

- **plot_CM.m**

USAGE:

```
handle = plot_CM(CM, titleString, Ncolors, zerocolor);
```

DESCRIPTION:

Plots a correlation matrix CM with color options.

INPUTS:

CM = Input square correlation matrix.

titleString([]) = The title string for the plot.

Ncolors (256) = The number of colors channels to use in colorizing the correlation matrix.

zerocolor ([1; 1; 1]) = The RGB color used to label the value of zero correlation (white).

OUTPUT:

handle = The figure handle for the correlation matrix plot.

- **make_CM_colormap.m**

USAGE:

```
map = make_CM_colormap (Ncolors, zerocolor);
```

DESCRIPTION:

Generates correlation matrix color map that transitions from the most positive correlation (+1) dark blue -> blue -> cyan -> white -> yellow -> red -> dark red, the most negative valued correlation (-1).

INPUTS:

Ncolors (256) = The number of colors channels to use in colorizing the correlation matrix.

zerocolor ([1; 1; 1]) = The RGB color used to label the value of zero correlation (white).

OUTPUT:

map = The color map for use in colorizing a correlation matrix plot.

2.9 Auxiliary tools

There are some auxiliary tools used in STANCE, that perform sometimes necessary miscellaneous tasks including dos command file removal, FWHM of the Gamma distribution calculation, find the center of mass, etc. These are often obtained from elsewhere.

- **bfilter3.m**

USAGE:

```
B = bfilter3(A,w,sigma);
```

DESCRIPTION:

Three dimensional bilateral filtering.

INPUTS (default):

A = 3D normalized grayscale/color image of interest to be filtered.

w (5) = The half-size of the Gaussian bilateral filter window.

sigma ([3, 0.1]) = Standard deviations of the bilateral filter.

sigma(1) – the spatial-domain standard deviation.

sigma(2) – the intensity-domain standard deviation.

OUTPUT:

B = Bilateral filtered 3D grayscale/color image the same size as A.

SOURCE:

Adapted for use in 3D Volumes by Kevin Matlock, kevin dot matlock at ttu dot edu.

Original 2D Filter By: Douglas R. Lanman, Brown University, September 2006; dlanman at brown dot edu, http://mesh.brown.edu/dlanman.

- **cmd_rmdir.m**

USAGE:

```
[status, message] = cmd_rmdir(folderspec);
```

DESCRIPTION:

Removes a directory and its contents, handling any PC file access issues by issuing a DOS command.

INPUT:

folderspec = The folder to remove (along with any contents).

OUTPUTS:

status = If command is successfully executed = 0, non-zero otherwise.

message = Explanatory message of the result.

- **gamFWHM.m**

USAGE:

```
delta_x = gamFWHM(k);
```

DESCRIPTION:

Calculates the FWHM of gamma distribution with assumption that theta is one.

INPUT:

k = The scaling parameter for phase delay of the gamma distribution (alpha).

OUTPUT:

delta_x = The (FWHM/theta) of the gamma distribution.

EQUATION:

$$FWHM_{gamma}(k,\theta) = \theta(k-1)\left[W_0\left(\frac{-2^{\frac{1}{1-k}}}{e}\right) - W_{-1}\left(\frac{-2^{\frac{1}{1-k}}}{e}\right)\right],$$

where the $W(\cdot)$'s are Lambert W functions (AKA product-log's) and one must require that $k > 1$.

- **reslice.m**

USAGE:

```
reslice(PI,PO,dim,mat,hld);
```

DESCRIPTION:

Slices the 3D data array to 2D slices in certain dimension with given registration.

INPUTS:

PI = Input 3D data of interest, file name or SPM header struct.

PO = Output filename or SPM header structure to be saved to.

dim = 1x3 matrix of image dimensions.

mat = 4x4 affine transformation matrix mapping from voxels to [mm] world coordinates for output image.

hld = Interpolation method argument used by spm_slice_vol.m.

SOURCE: JohnsGems.html 1.42 John Ashburner 05/02/02.

- **center_of_mass.m**

USAGE:

```
CenterOfMass = center_of_mass(data);
```

DESCRIPTION:

Finds the center of mass coordinates of a 3D data array.

INPUT:

data = Array of real 3D data.

OUPUT:

CenterOfMass = The center of mass 3x1 coordinate triple.

- **enforce_positive_definiteness.m**

USAGE:

```
Mout = enforce_positive_definiteness(M);
```

DESCRIPTION:

Force the square matrix M to be positive definite.

INPUT:

M = A square matrix.

OUTPUT:

Mout = A matrix calculated from M via Cholesky decomposition that is positive definite.

- **estimate_PDF.m**

USAGE:

```
[pdf,bins] = estimate_PDF(data,bin_size,bin_range);
```

DESCRIPTION:

Estimates the probability density function (PDF) of input data over given size of bin over the first data dimension.

INPUTS (default):

data = Input data with first dimension being temporal .

bin_size (1) = The width size of bins to use.

bin_range ([min() max())] = Specifies the range of bins to use; determines the number of bins.

OUTPUT:

pdf = A discrete estimate of the PDF of the input data over the bins.

bins = bin areas, i.e. the bin height times the bin_size;

- **multivariate_kurtosis.m**

USAGE:

```
b = multivariate_kurtosis(data);
```

DESCRIPTION:

Computes the multivariate kurtosis of data with the temporal dimension as the first dimension.

INPUT:

data = Time series data with the temporal dimension as the first dimension.

OUTPUT:

b = The multivariate kurtosis calculated from temporal data.

- **tal2min.m**

USAGE:

```
outpoints = mni2tal();
```

DESCRIPTION:

Convert Talairach inpoints (a 3xN array of coordinates) to MNI outpoints (best guess) .

OUTPUT:

outpoints = 3xN array of coordinates that have been transformed from Talairach coordinates to MNI coordinates.

SOURCE:

Matthew Brett 2/2/01, matthew dot brett at mrc-cbu dot cam dot acdot uk; modified 02/2003, by Darren Weber, darren dot weber at radiology dot ucsf dot edu.

- **mni2tal_matrix.m**

USAGE:

```
M2T = mni2tal_matrix();
```

DESCRIPTION:

Returns a structure array containing rotation matrices used by mni2tal() and tal2mni() functions that convert between Talairach coordinates and MNI coordinates (best guess) .

OUTPUT:

M2T = Structure array of three 4x4 conversion matrices to approximately transform between Talairach coordinates and MNI coordinates.

M2T.rotn = spm_matrix([0 0 0 0.05]);

M2T. upz = spm_matrix([0 0 0 0 0 0 0.99 0.97 0.92]);

M2T. downz = spm_matrix([0 0 0 0 0 0 0.99 0.97 0.84]);

SOURCE:

Matthew Brett 2/2/01, matthew dot brett at mrc-cbu dot cam dot acdot uk; modified 02/2003, by Darren Weber, darren dot weber at radiology dot ucsf dot edu.

■ **colorspace3d.m**

USAGE:

```
A = colorspace3d(I, rgb2lab);
```

DESCRIPTION:

Converts a 3-dimensional RGB image from RGB to LAB color space.

INPUTS:

I = Input 3D color image to be converted.

rgb2lab = Options for converting RGB image to LAB (true) color space or inversely (false).

OUTPUT:

A = Converted 3D color image the same size as I.

3 Key structure arrays used by STANCE functions

STANCE simulator functions often require much information about various phenomena. Several structure arrays are used to facilitate this passing of information.

❖ **The task structure**

The task structure organizes important information about a task or stimulus of interest. Note the name of the structure itself is flexible, but it must have some of the following fields (defaults):

name = a string naming the overall task/stimulus.

activation(n) = a structure detailing the 4^{th} component activation map.

combine{1} = {1^{st} fuzzy logic operator string, scope option string}.

…

combine{m} = {m^{th} fuzzy logic operator string, scope option string}.

amplitude (0.02) = the amplitude of peak activation as a fraction of peak signal.

map = the activation map, normalized to values between 0 and 1.

GMvolume = the sum of gray matter priors masked with activation map.

exp_design = (optional) associated experimental design timeseries object.

Fuzzy logical operator choices: 'NOT', 'OR', 'AND', 'XOR', or 'NAND'; the directive: 'mask'.

Combine scope choices: with logical: ‘all, flip’; with mask: ‘L’, ‘R’ – for left/right masking.

❖ **The activation structure**

The activation structure controls how a 2D/3D activation map is constructed; see demo_3D_define_act.m for examples of usage. It contains the following fields with the default values in parentheses and alternate contexts listed and color coded

region ('') = a string naming the region being constructed.

= a file name.

= a string labelling the ROI.

origin ([0, 0]/ [0,0,0]) = Stereotactic origin of the resulting activation map.

center ([0, 0]/ [0,0,0]) = characteristic coordinate of the activation, i.e. the center of a shape.

volume ([]) = [mm^3] the scalar volume (the amount of space occupied ~ sum of voxels), negative volume is used as flag so that its magnitude becomes a multiplicative factor to the default construction of proportion. If empty, then computed from the proportion.

= the component/ROI integer label for atlas; sometimes the sign of the integer is used as a flag between related atlases (e.g. (+)cortical/(-)subcortical).

shape ('circle'/ 'sphere') = a string name of the shape.

= a cell array with: shape{1} = a string name of the shape.

shape{2} = superpower (used as exponent for super* shapes).

if superpower itself is a cell array, then

n = abs(superpower{1}); r = superpower{2}, where signs are flags.

= 'data' - to indicate the loading of the activation map from a data file.

= 'atlas'/'parcellation' - to indicate the loading of the activation map from an atlas.

= 'mask' – define activation map from a list of coordinates.

proportion ([1,1]/ [1,1,1]) = a scalar isotropically specifying a characteristic length or aspect ratio.

= a 1D array specifying the aspect ratio proportions or characteristic lengths.

= threshold to apply to loaded data.

= threshold indicator for atlas.

rotation (0/ = a scalar specifying the rotation about an axis perpendicular to a plane.

[0,0,0]) = a 1D array specifying the rotations about all axes.

falloff (0) = a scalar isotropically specifying the rate of intensity falloff from the center.

= a 1D array specifying the rate of intensity falloff from the center of shape.

minimum (0) = the minimum intensity threshold (0.0-1.0) cutoff for the resulting map.

map = the resulting 2D/3D map.

Standard shape geometric options with 2D/3D geometric correspondences and alternates shown

2D shape options:	'circle',	'square',	'rectangle',	'ellipse',	'diamond'
	↕	↕	↕	↕	↕
3D shape options:	'sphere',	'cube',	'box',	'ellipsoid',	'prism'
alternative:	'ball'		'cuboid'	'spheroid'	

Super geometric shape options (may require superpower exponent):

2D shape options:	'superellipse',	'supercircle',	'squircle'	
	↕	↕		
3D shape options:	'superellisoid'	'superball',	'superegg',	'astroid'

❖ **The scan structure**

The fMRI scan protocol for every simulation is specified with this structure. The fields (default) are

voxel.size ([3 3 3]) = Voxel size physical dimensions [mm].

voxel.matrix ([64 64 NaN]) = Size of the resulting matrix in functional space (NaN indicated that number of Z slices is to be autodetected based on the gray matter priors).

voxel.spacing ([0 0 0.2*scan.voxel.size(3)]) = The slice gap spacing; assume only 20% Z-gap spacing.

tiltAngle (15) = Tilt angle relative to the AC-PC line [degrees].

TR (2000) = Repetition time [ms].

TE (30) = Echo time [ms].

ES (0.51) = Echo spacing [ms].

FA (90) = Flip angle [degrees].

BW (2232) = Bandwidth [Hz/Px].

order ('SD') = Slice timing order; SD = sequential descending order.

B0 (3) = Field strength of the main magnet [T].

KM0 (2225) = Value of ($K \cdot \mathrm{PD}$) in the EPI signal equation; default value is fit to data with a maximum observed intensity of 909 at 3T and FA = 90 degrees .

noise_method ('sigma') = Method of specifying noise – as variation in terms of, σ, 'sigma' or as 'percent' of peak signal.

noise (0) = Quantifies the thermal noise in the grey matter by noise_method.

attenuation (0) = Attenuation factor λ modelling exponential falloff $\sim\exp(-\lambda r)$, where r is measured from the surface of the head.

acceleration (1) = The multi-band acceleration factor; imaging multiple slices every TR.

The string for specifying the slice order (scan.order) is constructed by the prefixes and suffixes:

'S_' - Sequential ordering

'I_' - Interleaved ordering

'_A' - Ascending ordering

'_D' - Descending ordering

'__2' - start with even numbers instead of odd numbers (default)

The value scan.KM0 is the effective value of the constant term ($K \cdot \mathrm{PD}$) of the EPI signal equation:

$$S = K \cdot PD \frac{\sin\alpha\left(1 - e^{-TR/T1}\right)}{\left(1 - \cos\alpha\, e^{-TR/T1}\right)} e^{-TE/T2^*}$$

This constant varies based on B_0, the pulse sequence and even from scanner to scanner, so it is best fit directly from raw data.

❖ **The physio structure**

Characterizing and controlling the simulation of the six main sources of physiological noise (Nunes, 2014; Wu and Marinazzo, 2017) requires much information. The physio structure organizes these parameters into the following fields and subfields (default)

weight (75.0) = Weight [kg].

lambdas = 1D array of lambda values for major tissue types according to

([0.009, 0.006, 0.02, 0.05]) the tSNR model in the order [GM WM CSF BV].

Rsquared = R^2 of component [RP RR CP CR BP InterCRP] versus tissue type.

([[0.046, 0.022, 0.042, 0.013, 0.00325, 0.022]; … % cerebrospinal fluid (CSF) row.

[0.04, 0.021, 0.03, 0.012, 0.0025, 0.017];… % gray matter (GM) row.

[0.042, 0.020, 0.032, 0.011, 0.00275, 0.020];... % white matter (WM) row.

[0.037, 0.017, 0.079, 0.009, 0.00225, 0.040];... % brain stem (BS) row

[0.033, 0.02, 0.25, 0.01, 0.0025, 0.02]]) % blood vessel (BV) Row

respiratory.TI (4.0) = Average respiratory time interval [s].

respiratory.sigma (0.25) = Standard deviation of the respiratory time interval [s].

respiratory.A_z (1.0) = Average chest motion height [cm].

respiratory.A_z_sigma (0.005) = Standard deviation of chest motion height [cm].

respiratory.sigh.flag (false) = Flag to indicate whether or not to simulate spontaneous sighing.

respiratory.sigh.TI (400.0) = Average spontaneous sighing time interval [s].

respiratory.sigh.sigma (15) = Standard deviation of the spontaneous sighing time interval [s].

respiratory.seed.impulse ([]) = Random number generator seed for respiratory impulse simulation.

respiratory.seed.dz ([]) = Random number generator seed for chest motion simulation.

respiratory.RRF.param ([]) = Structure/list of the subject's respiratory response (RR) parameters.

respiratory.lags ([]) = Spatial map of time-lags for the respiratory pulse (RP) [# of dt].

cardiac.TI (1.05) = Heart beat (HB) time interval [s].

cardiac.IPFM.freqs = Frequencies for Integral Pulse Frequency Modulation Model

([0.02,0.1,NaN]) (IPFM) where NaN indicates to use RP(t) instead of a sinusoid [Hz].

cardiac.IPFM.sigmas = Standard deviations of rates in terms of fractional values of

([0.2,0.2,NaN]) (1/f) for IPFM [s].

cardiac.IPFM.amplitudes = The sinusoid amplitudes for the IPFM VLF, LF and HF components

([1.0, 1.0, (2/3)]) (Mitov, 2001).

cardiac.IPFM.seeds = Random number generator seeds for the IPFM (reproducibility).

([[[],[]], [[],[]], NaN])

cardiac.CRF.param ([])	= Structure/list of the subject's cardiac response (CR) parameters
cardiac.PWV.width (0.5)	= Width of the pulse wave (PW) [fraction of TI].
cardiac.PWV.width_sigma (0.02)	=Standard deviation of the width of the pulse wave [fraction of TI].
cardiac.PWV.v0 (3.0)	= Average velocity of the pulse wave in the capillaries [mm/2].
cardiac.PWV.v_constant_fraction (0.1)	= Fraction of the total velocity that is always flowing (minimum continuous blood flow).
cardiac.PWV.seed ([])	= Random number generator seed for the PWV simulator.
cardiac.lags ([])	= Spatial map of time-lags [# of dt].
max_lag_time (5)	= Maximum lag time expected [s].

4 Scripts of demos

The STANCE simulator package contains a library of demo scripts, which are located in the '.../scripts_for_demos' directory. These demos illustrate some pipelines for simulating fMRI data, defining activation maps and generating time series, for a given experiment. In the case of 3D activation map simulations, the demos model some activation maps and simulate the BOLD signal of given task-related and resting-state experiments based on some results well studied the literature. In the case of 4D time series simulations, the demos involve specifying the experiment design, defining the activation maps, various noise and time series emulation, and finally the calculation of the expected EPI signal is done taking account of the slice timing and motion. All of the scripts demos showcase the use of the various functions within the STANCE simulator toolbox described in Section 2. They are scripts (can run without any arguments) and are self-initialized, automatically locating the STANCE and SPM file directories and loading global variables. Other details of the demos are described in the sections below:

4.1 Demos of 3D tools

- **demo_3D_define_act.m**

This demo gives a simple pipeline and usage for defining and modeling the activation maps of interest. The activation regions can be specified by the user with different shapes (with specific volumes, rotations and coordinates), by loading regions of interest (ROIs) from an atlas/parcellation, or by loading from a data file. The general procedure of this demo follows these steps:

1. The target space dimensions and origin location for 2D slices or 3D volumes should be defined.
2. The activation structure array with all necessary fields needs to be fully specified, including its name, geometric shape, volume (area), center location, the degree of Gaussian smoothing, interpolation method and aspect ratio. Some arguments have default values and do not need to always be explicitly given.
3. In the case that one needs more than a single shape to define a region, the activation map can also be constructed by combining different regions via fuzzy logical operators (OR, AND, etc.).

4. One can explicitly define the coordinates of every voxel in the activation template to create a custom mask by specifying shape = 'mask' and providing a list of voxel coordinates as an array to center.
5. With the arguments set up above, it is easy to use function 'STANCE_make_activation_map' to generate a synthetic activation region.

Sometimes it is useful to use parceled ROIs already cataloged in a brain atlas. Currently STANCE has five built-in atlas resources, the Automated Anatomical Labels atlas (aal.nii), and the Brodmann atlas (brodmann.nii), the Harvard-Oxford cortical and subcortical atlases (each with 0%, 25%, and 50% thresholding versions), the Craddock parcellations (with 200 or 400 ROIs), and a 25% threshold Brainnetome atlas (BrainnetomeAtlas_BNA_MPM_thr25_1.25mm.nii). Examples of loading brain atlas ROIs (e.g. the mask for the Left Precuneaus region from aal.nii) are provided in this demo. In principle, any data file or labeled atlas could be accessed as long as the filepath is provided, however, it may need to be conformed to the same dimensions and voxel size as our T1-w anatomical volumes.

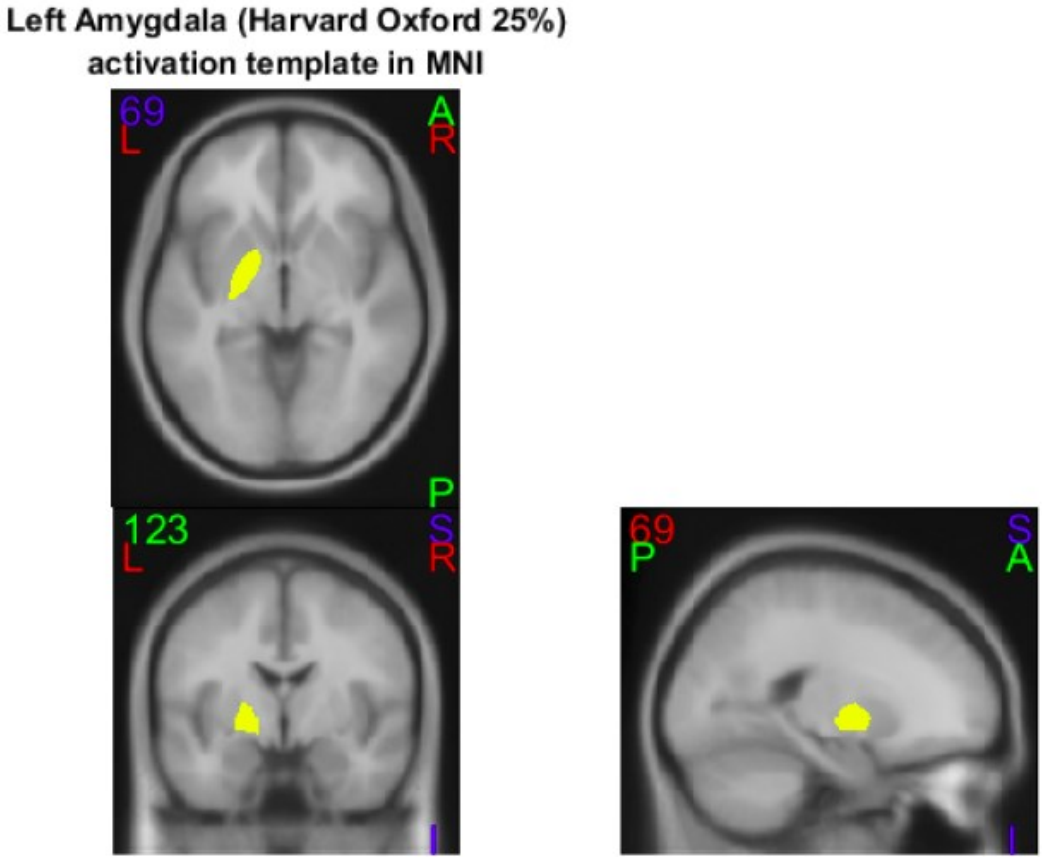


Figure 2. The left amygdala ROI loaded from the Harvard-Oxford atlas.

- **demo_3D_ex1.m**

This demo gives a simple pipeline and usage for defining and modeling the activation map of a simple finger opposition task, where a right-handed subject is asked to touch each digit to their thumb in a progressive and repetitive fashion. The neuronal activation template is modeled as a sphere with fading, having a volume of 45.9 voxels, centered at Talaraich coordinates [-33.8, -18.2, 51.8], according to the mean observed values. The general procedure of this demo is described below:

1. **Load** the MNI T1-w brain image and the subject brain of interest, **and** then **register** the MNI brain to the subject's T1-w brain.
2. **Define the activation template** of a specific task or stimulus by modelling its location, shape and volume the same as the activation blob observed and described in a published fMRI study. *Alternatively*, this demo illustrates how to load statistical parameter maps from the source files finger tapping_pFgA_z_FDR_0.01.nii.gz & finger tapping_pAgF_z_FDR_0.01.nii.gz, provided by www.neurosynth.org ,which are derived from ~60 studies, each related to a finger activity tasks.
3. **Define the fMRI scan protocol** and model the subject's 3D baseline T2* map in functional space from the subject's tissue priors with the given scan protocol.
4. **Define the subject's activation map with a cortical constraint** by first transforming the activation region in MNI space onto the subject's functional space and then masking it with the subject's gray matter priors.
5. **Construct the T2* BOLD map** by adding the subject specific activation map to the T2* baseline according to a specified amplitude of activation and the TE of the fMRI scan protocol. This serves as the 3D activation-influenced map used in generating the BOLD-related signal.
6. **Solve for the fMRI signal** from an Echo Planar Imaging-Gradient-Recalled Echo (EPI-GRE) MRI pulse sequence, with or without noise, motion, or attenuation.

Figure 3. Some results from the pipeline in demo_3D_ex1.m.

- **demo_3D_ex2.m**

This demo illustrates how stimulus activation can be modeled from the literature with a similar procedure as outlined in demo_3D_ex1.m. The first published fMRI experiment (Ogawa *et al.*, 1990) involved the photic stimulation of the visual cortex, and has been modeled and illustrated in this demo. The photic stimulation template is defined by two ellipsoids both with volume 6,000 mm^3 and aspect ratio 3:8:4, and each with the observed centers [9,-84, 10] & [-10,-80, 10] (each adjusted by ~25% of reported values in a subject to match MNI space); both with rotations about the X-axis of 30° and with a 0.005 Gaussian falloff with a minimum value of 0.2 relative to a maximum normalized value of ~1.0. This activation map is estimated from a cross-section of the activation having an area $\cong 300\ \text{mm}^2 \cong (18\ \text{mm})^2$ which then yields the volume of $6000 \cong (18\ \text{mm})^3$.

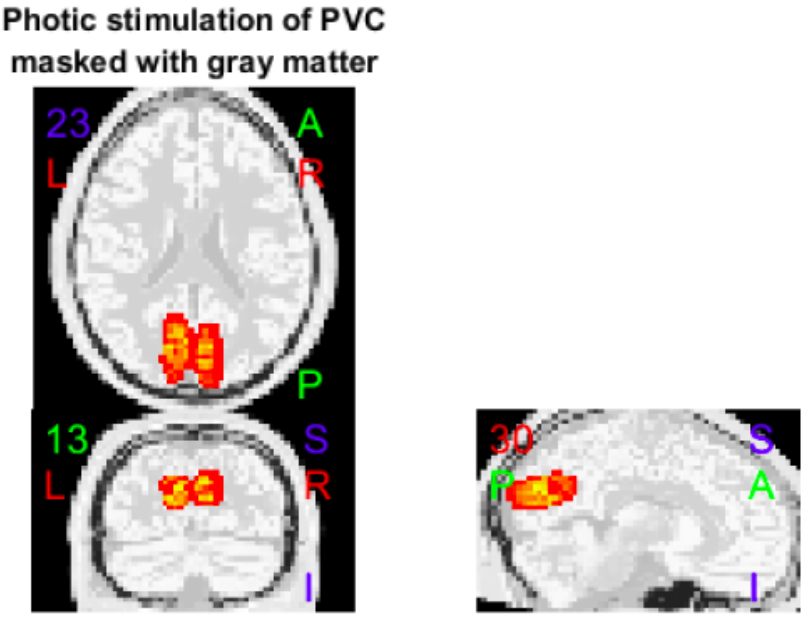


Figure 4. The resulting modeled activation map for photic stimulation after transforming to the subject's functional space and applying a gray matter mask to enforce cortical activation.

- **demo_3D_ex3.m**

The Gustatory (tasting) task modeled in this demo gives a more complex example of a task-related activation pattern. The 8 activation blobs found from the statistical analysis of a group study of the gustatory task are relatively small with irregular shapes and are modeled by the closest apparent shapes at the observed locations with the reported volumes. Standard fading is added and activation regions are combined via a fuzzy logical OR operator with the 'STANCE_combine_maps' function.

The resulting fMRI volume is then simulated following a similar procedure as that described in 'demo_3D_ex1.m'.

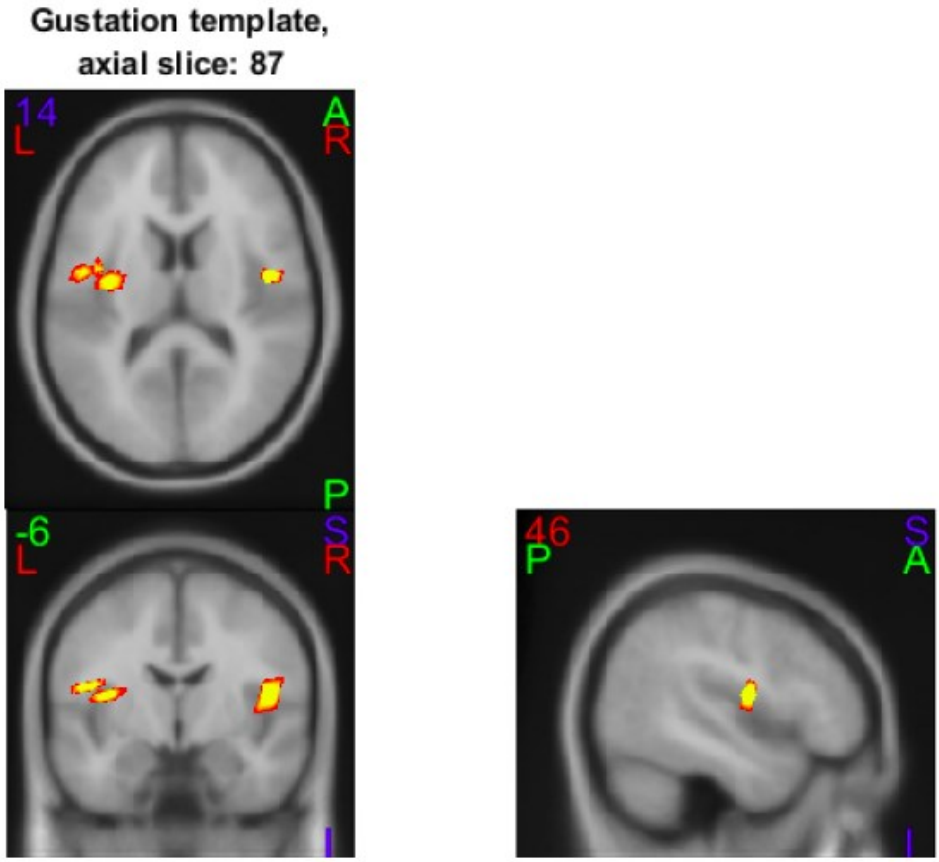


Figure 5. The resulting activation template for the gustatory task in MNI space as modeled from descriptions in the literature.

- **demo_3D_RSNs_models.m**

This demo models the complicated shapes of 3D baseline signals for some important and well-studied Resting State Networks (RSNs) detailed in the landmark paper "A baseline for the multivariate comparison of resting-state networks" by (Allen *et al.*, 2011) following a similar procedure as that outlined above. RSN ICs of the Basal Ganglia Network, Auditory Network and Default Mode Networks have been modeled and compared with corresponding ICs determined from a group study. It should be noted that the first two RSNs where initially modeled by two high school interns, demonstrating STANCE's usefulness as an educational aid.

4.2 Demos of 4D tools

- **demo_4D_physio.m**

This demo attempts to realistically model respiratory- and cardiac-controlled physiological signals in a normal heathy awake subject lying in a supine position. In detail, six main sources of noise (Nunes, 2014; Wu and Marinazzo, 2017) are emulated:

I. The respiratory pulse (RP), which follows the respiratory lung volume time-series, RVT(t).

II. The cardiac pulse (CP), which follows the blood flow pulse wave velocity time-series, PWV(t), that is modulated by the heartbeat, HB(t).

III. The interaction between cardiac and respiratory phases (InterCRP), which approximately follows the product of RP and CP time-series.

IV. The respiratory volume (RV) response, which comes from convolving the respiratory response function (RRF) with RVT(t).

V. Heart rate (HR) response, which comes from convolving the cardiac response function (CRF) with the HRV(heart rate variability).

VI. Arterial blood pressure ABP(t), described by Mayer waves with a frequency ~0.1 Hz inducing a fluctuation about ~25% of that caused by the HR (Katura *et al.*, 2006).

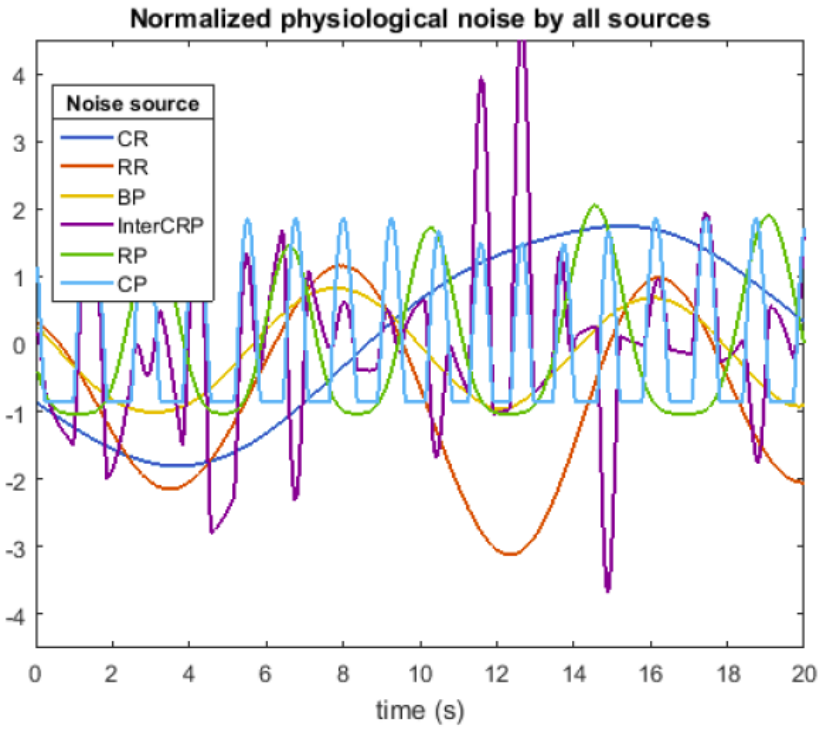


Figure 6. The results of the simulation of the six main source of physiological noise.

These are then combined following the tSNR model (Krüger *et al.*, 2001), where their relative contributions to the overall standard deviation in the signal due physiological noise varies by tissue type. The total physiological noise is parameterized by λ and affects the tSNR by the relation

$$\mathrm{tSNR} = \frac{\mathrm{tSNR}_0}{\sqrt{1+\lambda^2 \mathrm{tSNR}_0^2}}, \text{ where } \mathrm{tSNR}_0 = \langle S(t) \rangle / \sigma_0.$$

The six time-series are combined according to observed contributions to the signal standard deviation (Wu and Marinazzo, 2015) , $\sigma_{\mathrm{P},m} = \lambda_m \langle S(t) \rangle$, and added to the signal following the tSNR equation above, according to the tissue type (Krüger and Glover, 2001).

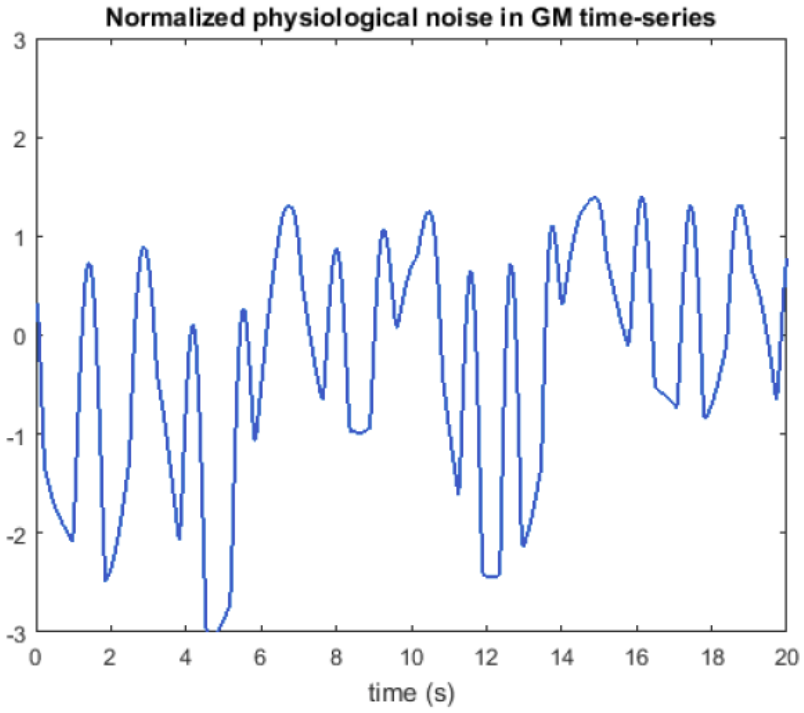


Figure 7. The results of simulating physiological noise in the gray matter (GM).

- **demo_4D_expdesign.m**

This demo simulates the full time series of a finger tapping fMRI experiment. The first half part illustrates the pipeline to define and model the fMRI signal of the task in a frame with peak activation, like demo_3D_ex1, etc. The second half part mainly focus on designing the 4D time series, estimating BOLD response, adding various kinds of noise and finally solving the signal equation for fMRI data obtained from an EPI sequence while taking into account slice timing.

1. Follow the steps illustrated in demo 'demo_3D_ex1.m' and emulate the 3D BOLD signal of the finger tapping task, this time from data sources.
2. Specify the experimental time-series which is then convolved with the canonical BOLD hemodynamic response function (HRF) to yield the ideal expected BOLD response, which is used to generate the “pristine” EPI signal.
3. Add system and/or physiological noise to the 4D EPI signal.

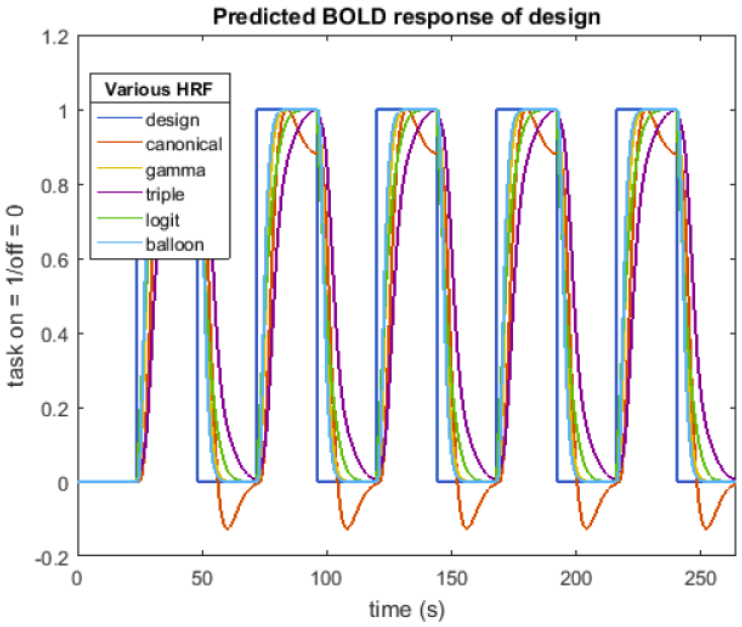


Figure 8. The BOLD response as modeled by the default values of the various HRF options.

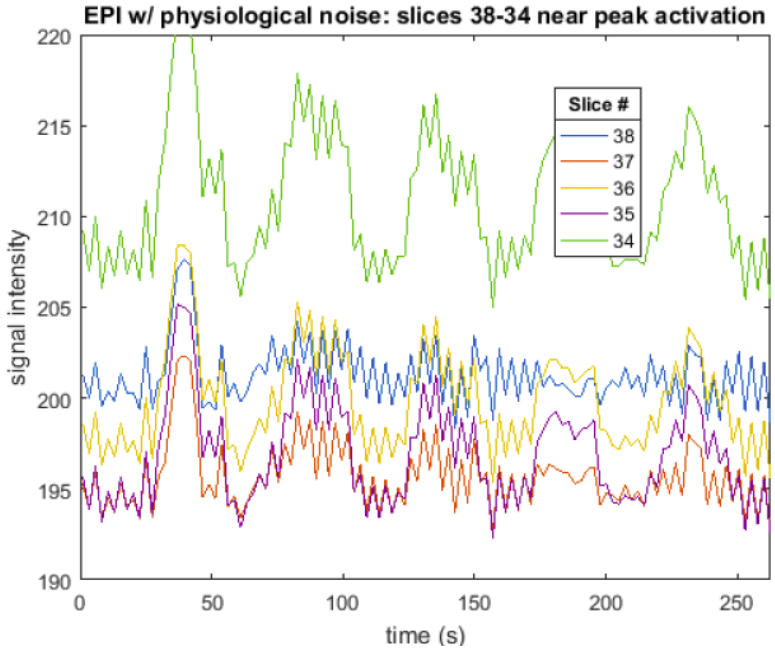


Figure 9. The EPI signal with in selected voxels with physiological noise added to the task activation.

- **demo_4D_simblock.m**

This demo emulates the fMRI data from a block design experiment of alternating right and left finger tapping tasks with similar procedure described in 'demo_4D_expdesign.m' while following the webpage: 'http://www.mccauslandcenter.sc.edu/crnl/sw/tutorial/html/blockspm.html', which is used to test the performance of STANCE simulator. This demo contains several advanced simulation approaches, including:

  - Having more than one task-related activation condition and experimental design time series.
  - Adding spatial variability to the hemodynamic response function parameters.
  - Automatically determining the respiratory time interval average and standard deviation from motion time-series derived from the real data.
  - Adding spatially varying lag times to the physiological noise (proof-of-principle).
  - Simulating motion, derived from the real data, which affects the time series not only as rigid body motion, but also impinges spin history effects on the data.

Finally, a comparison is made between the time-series of voxels in both the real data and the simulated data for voxels at peak activation for both conditions, also contrasted with a control voxel in the gray matter with no discernable activation correlated with the tasks.

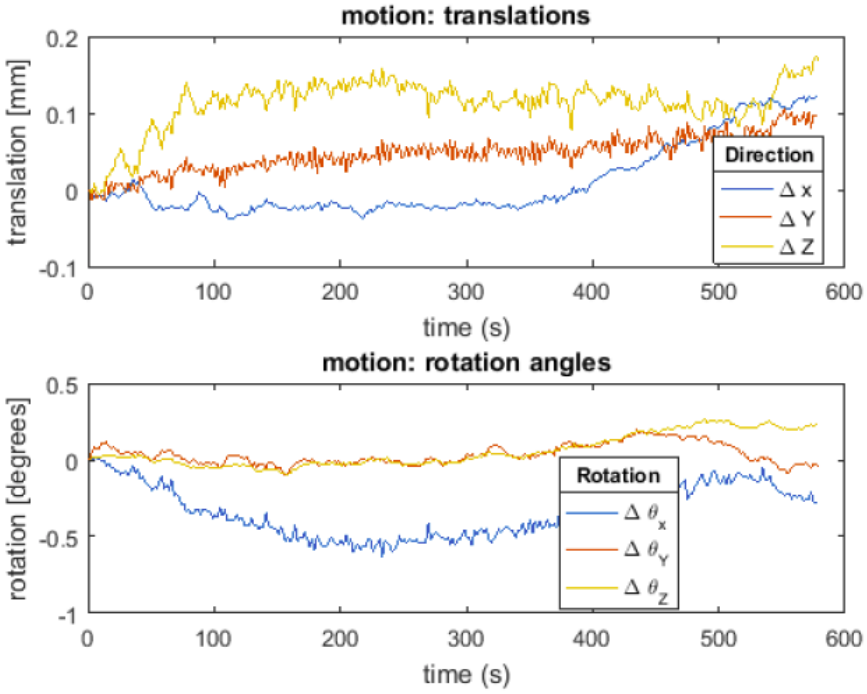


Figure 10. The motion time-series of the real data. Observe how apparently respiration is impinged on the y translations and the experimental design is impinged on the z translations.

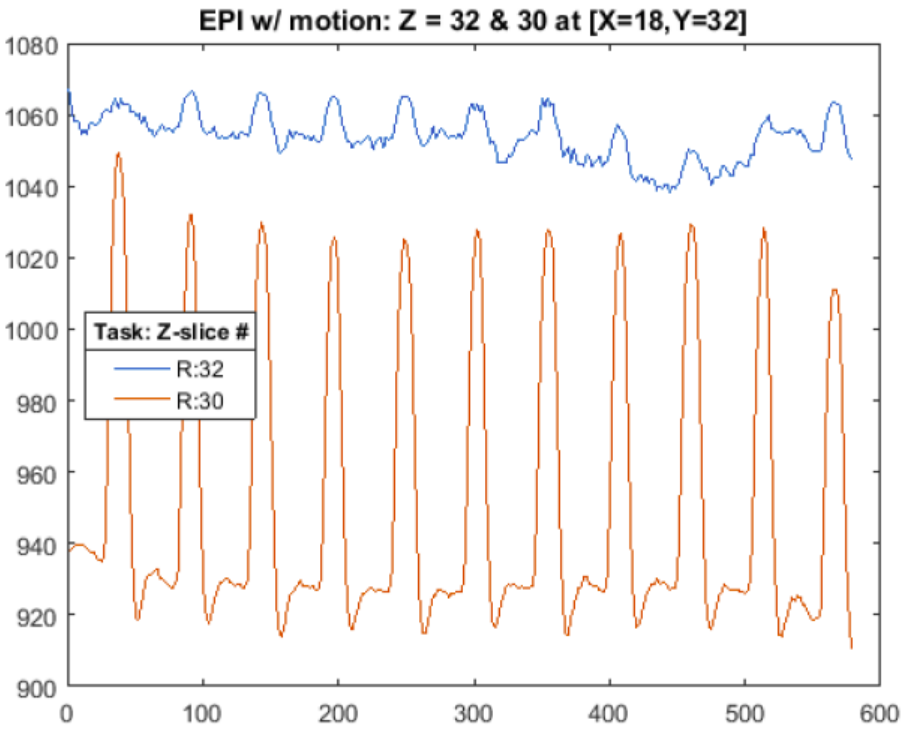


Figure 11. Motion added to the simulated data. This plot compares the peak activation voxel (Z=30) with a nearby voxel in the CSF that has no activation time-series (Z=32). The apparent correlation with the experimental design in the Z=32 voxel comes *completely* from motion! This illustrates how important it is to have motion time-series from real data to correctly take into account the affect of task-related motion.

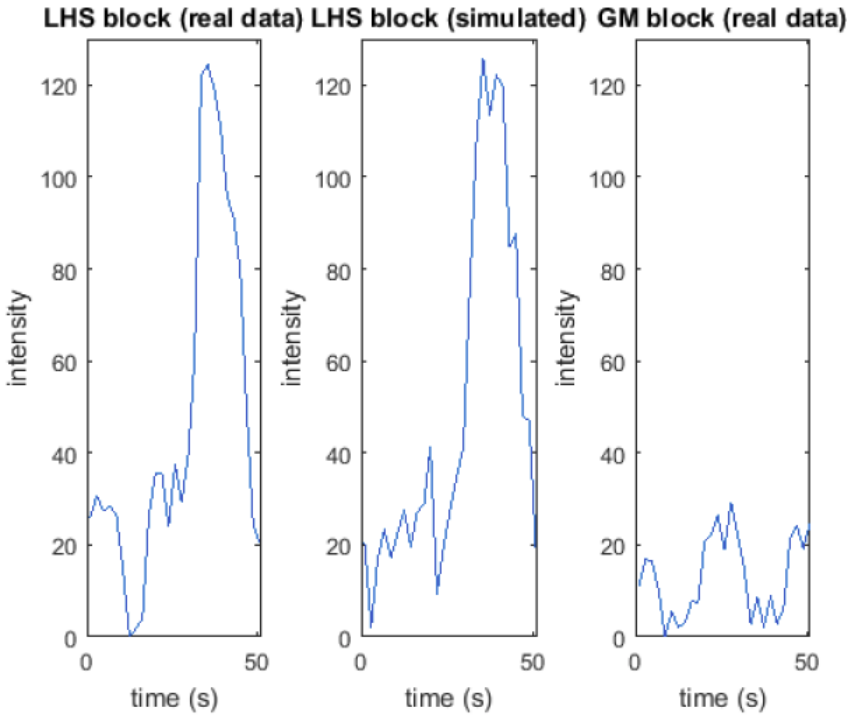


Figure 12. Comparison of real and simulated data at the voxel with peak activation. The time-series have been averaged over the 11 blocks of the experimental design. Rightmost is a control GM voxel.

5 References

Allen, E. A., Erhardt, E. B., Damaraju, E., Gruner, W., Segall, J. M., Silva, R. F., Havlicek, M., Rachakonda, S., Fries, J., Kalyanam, R., Michael, A. M., Caprihan, A., Turner, J. A., Eichele, T., Adelsheim, S., Bryan, A.D., Bustillo, J., Clark, V. P., Feldstein-Ewing, S.W., Filbey, F., Ford, C. C., Hutchison, K., Jung, R. E., Kiehl, K. A., Kodituwakku, P., Komesu, Y. M., Mayer, A. R., Pearlson, G. D., Phillips, J. P., Sadek, J. R., Stevens, M., Teuscher, U., Thoma, R. J., Calhoun, V. D., "A baseline for the multivariate comparison of resting-state networks," Front Syst Neurosci. (Feb. 2011).

Arimon, J. A., "Numerical modelling of pulse wave propagation in the cardiovascular system: development, validation and clinical applications," Ph.D. Thesis, The University of London (2006).

Aubert-Broche, B., Evans, A. C. and Collins, L., "A new improved version of realistic brain phantoms," NeuroImage **32**, 138–145 (June 2006).

Aubert-Broche, B., Griffin, M., Pike, G. B., Evans, A. C. and Collins, D. L. "Twenty New Digital Brain Phantoms for Creation of Validation Image Data Bases," IEEE Trans. on Med. Imaging **25**(11), 1410–1416 (Nov. 2006).

Barrett, K. E., Barman, S. M. Boitano, S. and Brooks H., [Ganong's Review of Medical Physiology] (24th ed.), McGraw-Hill, New York, 619 (2012).

Bayly, E. J., "Spectral analysis of pulse frequency modulation in the nervous systems," IEEE Trans. Biomed. Eng., BME **15**(1) 257–265, (1968).

Belliveau, J. W., Kennedy, D. N., McKinstry, R. C., Buchbinder, B. R., Weisskeopf, R. M., Cohen, M. S., Vevea, J. M., Brady, T. J. and Rosen, B. R. "Functional Mapping of the Human Visual Cortex by Magnetic Resonance Imaging," Science **254**(5032), 716-9 (1991).

Birn, M. B., Smith, M. A., Jones, T. B., and Bandettini, P. A., "The respiration response function: The temporal dynamics of fMRI signal fluctuations related to changes in respiration," NeuroImage **40**(2), 644–654 (2008).

Buxton R, Uludağ K, Dubowitz D, Liu T., "Modeling the Hemodynamic Response to Brain Activation." NeuroImage, **23**, S220-S233 (2004).

Chang, C., Cunningham, J. P., and Glover, G. H., "Influence of heart rate on the BOLD signal: The cardiac response function", NeuroImage **44**, 857–869 (2009).

Cordes, D., Nandy, R R., Schafer, S., and Wage, T. D., "Characterization and Reduction of Cardiac- and Respiratory-Induced Noise as a Function of the Sampling Rate (TR) in fMRI" NeuroImage **89**, 314-30 (2014).

Delawari, A. and Doelman, R., "Simulation of an Artificial Respiratory System: Choosing a New Actuator for Implementation in a Lung Simulator," B.Sc. Thesis, The University of Delft (2010).

Friston K., Fletcher P., Josephs O., Holmes A., Rugg M., and Turner, R., "Event-Related fMRI:Characterizing Differential Responses." NeuroImage, **7**, 30-40 (1998).

Glover, G., "Deconvolution of Impulse Response in Event-Related BOLD fMRI." NeuroImage, **9**, 416-429 (1999).

Gudbjartsson, H. and Patz, S., "The Rician Distribution of Noisy MRI Data," Magn. Reson. Med. **34**(6), 910–914 (1995).

Handwerker, D. A., Ollinger, J. M., D'Esposito, M., "Variation of BOLD hemodynamic responses across subjects and brain regions and their effects on statistical analyses," NeuroImage **21**: 1639–1651 (2004).

Handwerker, D. A., Gonzalez-Castillo, J., D'Esposito, M., and Bandettini, P. A., "The continuing challenge of understanding and modeling hemodynamic variation in fMRI" NeuroImage **62**(2), 1017-1023 (2012).

Hill, J. E., Liu, X., Nutter, B., and Mitra, S., "A Task-related and Resting State Realistic fMRI Simulator for fMRI Data Validation," Proc. SPIE **10133**, Medical Imaging 2017: Image Processing, 101332N (February 24, 2017).

Jones, R. L. and Nzekwu, M. M., "The effects of body mass index on lung volumes," Chest. **30**(3), 827–33 (2006).

Katura, T., Tanaka, N., Obata, A., Sato, H., and Makia. A., "Quantitative evaluation of interrelations between spontaneous low frequency oscillations in cerebral hemodynamics and systemic cardiovascular dynamics," NeuroImage **31**(4), 1592-1600 (2006).

Krüger, G. and Glover, G. H., "Physiological Noise in Oxygenation-Sensitive Magnetic Resonance Imaging," Magn. Reson. Med. **46**, 631–637 (2001).

Liang, Z-P., and Lauterbur, P. C., [Principles of Magnetic Resonance Imaging: A Signal Processing Perspective], Wiley-IEEE Press, New York, Eq. (9.24), 299 (1999).

Lindquist, M.A., and Wager, T. D., "Validity and power in hemodynamic response modeling: a comparison study and a new approach," Hum Brain Mapp **28**:764–784 (2007).

Logothetis, N. K., Pauls, J., Augath, M., Trinath, T., and Oeltermann, A., "Neurophysiological investigation of the basis of the fMRI signal," Nature **412**: 150–157 (2001).

Lujan, A. E., Balter, J. M. and Ten Haken, R. K., "A method for incorporating organ motion due to breathing into 3D dose calculations in the liver: sensitivity to variations in motion," Med. Phys. **30**(1), 2643–2649 (2003).

Mitov, I. P., "Spectral analysis of heart rate variability using the integral pulse frequency modulation model" Med. Biol. Eng. Comput. **39**(1), 1-7 (2001).

Nunes, S., "Characterization of physiological noise in resting-state fMRI data at 7T", Thesis, Técnico Lisboa (2014).

Ogawa, S., Lee, T. M., Kay, A. R. and Tank, D.W., "Brain magnetic resonance imaging with contrast dependent on blood oxygenation," Proc. Natl. Acad. Sci. USA **87**(24), 9868-9872 (1990).

Rengachary, S. S. and Ellenbogen, R. G., (Eds.), [Principles of Neurosurgery], Elsevier Mosby, Edinburgh (2005).

Shan, Z. Y., Wright, M. J., Thompson, P. M., McMahon, K. L., Blokland, G. G., de Zubicaray, G. I., Martin, N., Vinkhuyzen, A. A. E. and Reutens, D. C. "Modeling of the hemodynamic responses in block design fMRI studies," J. of Cerebral Blood Flow & Metabolism **34**, 316–324 (2014).

Smith, A., Lewis, B., Ruttimann, U., Ye, F., Sinnwell, T., Yang, Y., Duyn, J., and Frank, J., "Investigation of Low Frequency Drift in fMRI Signal." NeuroImage, **9**, 526-533 (1999).

Purdon P, and Weisskoff, R., "Effect of Temporal Autocorrelation Due to Physiological Noise and Stimulus Paradigm on Voxel-Level False-Positive Rates in fMRI." Human Brain Mapping, **6**, 239-249 (1998).

Welvaert, M., Durnez, J., Moerkerke, B., Verdoolaege, G., and Rosseel, Y., "neuRosim: An R package for generating fMRI data," J. Stat. Software **044**(i10) online (2011).

Wu, G-R., and Marinazzo, D., "Hemodynamic response function in resting brain: disambiguating neural events and autonomic effects" bioRχiv (beta): the preprint server for biology, doi: https://doi.org/10.1101/028514 (2015).

Wu, G-R., and Marinazzo, D., "Sensitivity of the resting state hemodynamic response function estimation to autonomic nervous system fluctuations," Phil. Trans. R. Soc. A., in press (2017).

Zambanini, A., Cunningham, S. L., Parker, K. H., Khir, A. W., Thom, S. A. and Hughes, M. A. D., "Wave-energy patterns in carotid, brachial, and radial arteries: a noninvasive approach using wave-intensity analysis" Am. J. Physio. - Heart & Circulatory Physio. **289**(1), H270-H276 (2005).