Appendix E. Generating smooth 1D Gaussian trajectories

This Appendix describes how to generate smooth 1D Gaussian trajectories in MATLAB (The MathWorks, Natick, USA) by summarizing the mathematical theory and code of Penny (2008). Interested readers are encouraged to consult Penny (2008) for a variety of additional background details regarding Random Field Theory and its applications. The target audience for this Appendix is anyone with (a) familiarity with linear algebra basics and MATLAB programming, and (b) a desire to generate their own random 1D trajectories for validating and/or exploring Random Field Theory predictions.

Penny W (2008) Mathematics for Brain Imaging (Course Notes), Chapter 3: "Random Field Theory". Retrieved on 12 Feb 2015 from `http://www.fil.ion.ucl.ac.uk/~wpenny/mbi/`

Smooth Gaussian trajectories can be generated using the convolution procedure described in Appendix B but in most programming languages it is considerably easier to use a single matrix multiplication to achieve the same result:

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{z} \tag{E.1}$$

where, if Q is the number of trajectory nodes:

- $\boldsymbol{y}$ is the $(Q \times 1)$ smooth 1D Gaussian trajectory
- $\boldsymbol{C}$ is the $(Q \times Q)$ convolution matrix which embodies the correlation between neighboring points in the 1D trajectory
- $\boldsymbol{z}$ is a $(Q \times 1)$ trajectory of uncorrelated Gaussian values (Fig.B3a)

In MATLAB $\boldsymbol{z}$ can be generated as follows:

```
>>  Q = 101;
>>  z = randn(Q,1);
```

The convolution matrix $\boldsymbol{C}$ requires a bit more work. First note that a Gaussian covariance function with unit power is defined as:

$$r(d) = \exp\left(-\frac{d^2}{2s^2}\right) \tag{E.2}$$

where:

- d is the distance between the current node and a reference node in the trajectory. If Q=101 then d=0.01 for adjacent nodes, and d=0.01n for two nodes which are n nodes apart.
- s is the trajectory smoothness which linearly scales with the FWHM (Appendix A) as:

$$FWHM = s(Q-1)\sqrt{4\log 2} \tag{E.3}$$

The convolution matrix $\boldsymbol{C}$ embodies this covariance (Eqn.E.2) simultaneously for all points in the trajectory. To assemble $\boldsymbol{C}$ in MATLAB first define the number of trajectory nodes and the smoothness:

```
>>  Q = 101;              % number of trajectory nodes
>>  FWHM = 10;         % smoothness (FWHM units)
>>  s = FWHM / (  (Q-1) * sqrt( 4*log(2) )  )   %smoothness (s=0.060 for FWHM=10)
```

and then build a distance matrix $\boldsymbol{D}$ as follows:

```
>>  dq = 1 / (Q -1);      % inter-node distance
>>  x = dq * (1:Q);         % trajectory position
>>  X = repmat(x, Q, 1);  % temporary holder of trajectory positions
>>  D =  X - repmat(x', 1, Q);   % matrix of distances from diagonal nodes
```

The (i, j)th element of the resulting matrix represents the distance between nodes i and j:

```
>>  D(1:5, 1:5)

ans  =

        0    0.0100    0.0200    0.0300    0.0400
   -0.0100         0    0.0100    0.0200    0.0300
   -0.0200   -0.0100         0    0.0100    0.0200
   -0.0300   -0.0200   -0.0100         0    0.0100
   -0.0400   -0.0300   -0.0200   -0.0100         0
```

Next apply Eqn.E.2 to yield the covariance matrix $\boldsymbol{A}$:

```
>>  A = exp( -0.5 * D.^2 / s^2);
```

This covariance matrix can be visualized using the 'pcolor' command (Fig.E1):

```
>>  pcolor(A)
>>  colorbar
```

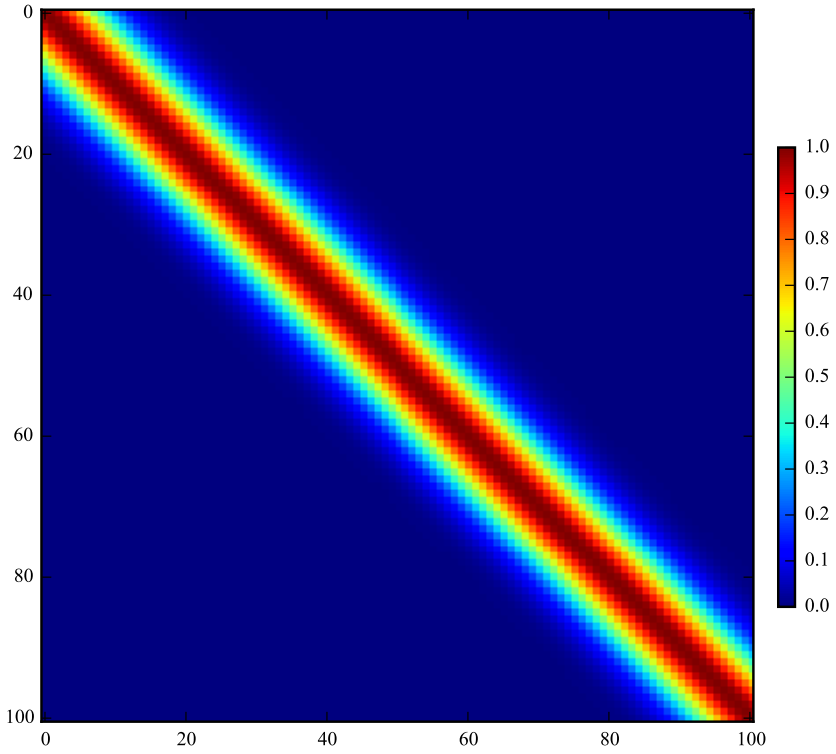

Figure E1: Visualization of the covariance matrix $\boldsymbol{A}$ for Q=101 and FWHM=10.0. The value of the (i, j)th matrix element represents the correlation values at 1D trajectory nodes i and j. When i=j the correlation is defined as 1.0, and since the data are smooth adjacent nodes' values are positively correlated. The farther apart the nodes the weaker their correlation.

Last, to generate data which have the covariance structure depicted above in Fig.E1, the convolution matrix $\boldsymbol{C}$ (Fig.E2) is assembled by first calculating the covariance matrix's eigen-solution:

```
>>  [V,U] =  eig(A);
```

where $\boldsymbol{V}$ and $\boldsymbol{U}$ are $\boldsymbol{A}$'s eigenvectors and eigenvalues, respectively, and then projecting the positive eigenvalues as follows:

```
>>  U = diag( U );
>>  U( U < 0 ) = 0;
>>  C = V * diag( sqrt( U ) ) * V' ;
```

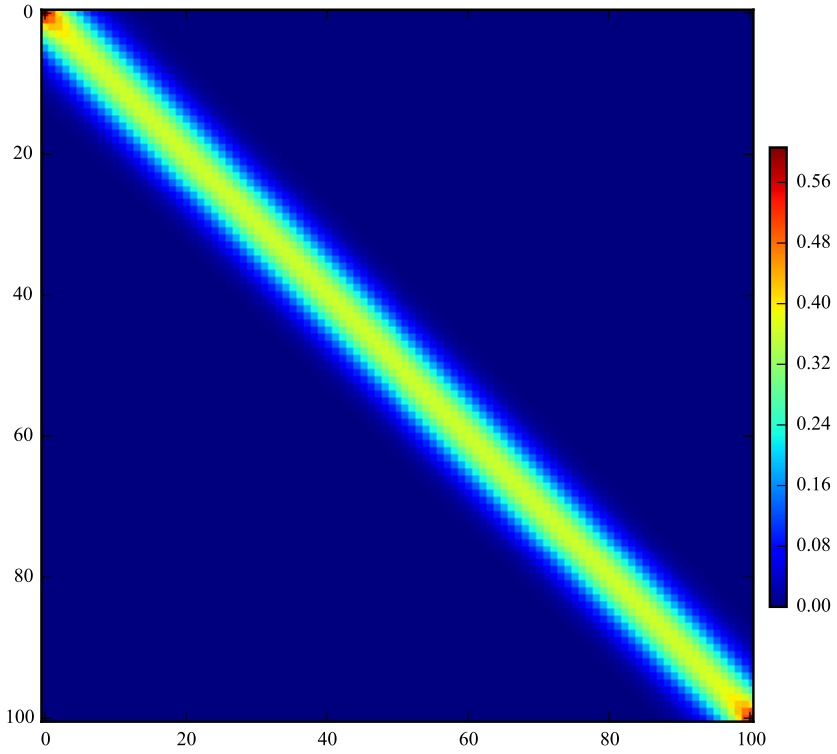

Figure E2: Visualization of the convolution matrix (Eqn.E.2) for Q=101 and FWHM=10.0.

This convolution matrix $\boldsymbol{C}$ needs to be computed only once for each smoothness value. It can then be used to generate single smooth Gaussian trajectories (Fig.E3) as follows :

```
>>  y0 = C * randn(Q,1);
>>  y1 = C * randn(Q,1);
>>  y2 = C * randn(Q,1);
>>  plot( y0 )
>>  plot( y1 )
>>  plot( y2 )
```

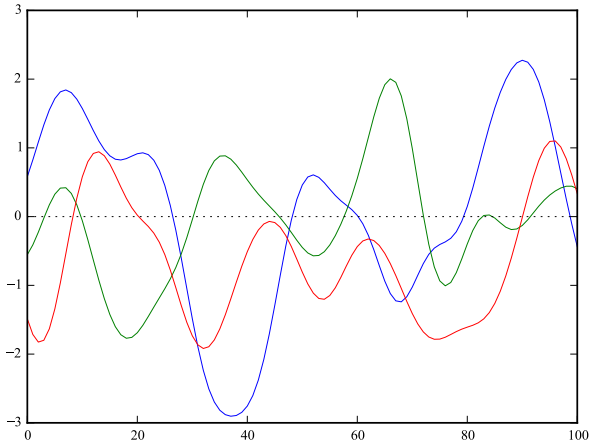

Figure E3: Three example 1D Gaussian trajectories.

Alternatively many random trajectories (Fig.E4) can be generated like this:

```
>>  J = 50;              % number of trajectories
>>  Z = randn(Q, J);     % uncorrelated Gaussian data
>>  Y = C * Z;           % (Q x J) array containing J separate random trajectories
>>  plot( Y )
```

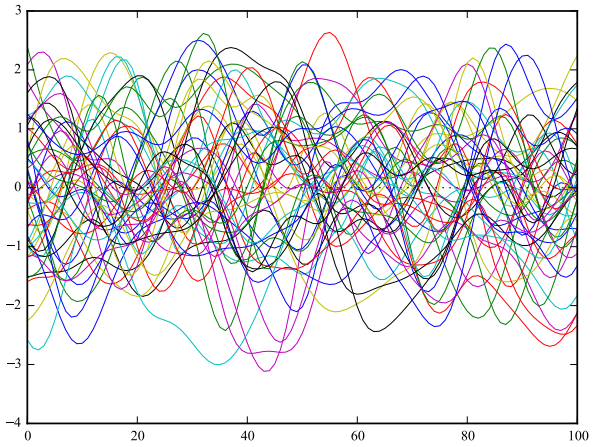

Figure E4: Fifty example 1D Gaussian trajectories.

For the analyses in the main manuscript we generated J=10 trajectories for each of two groups, computed the two-sample t trajectory using the usual definition of the two-sample t statistic, then repeated 100,000 times for each smoothness (FWHM) value. This yielded 100,000 t trajectories for each FWHM value. Last, we validated Eqns.3&4 (main manuscript) by extracting the maximum t value ($t_{\max}$) for each trajectory and then counting the number of $t_{\max}$ values which exceeded particular heights u.