

How Do State Variables Work?

Gerwin Schalk, 06/25/03

Overview

In BCI2000, system parameters are the variables that are constant throughout a defined period of operation. They are exposed to the investigator through the operator interface, and cannot be changed during operation.

In contrast, system states are the variables that are stored with each sample in the data file. Thus, they could change with every sample. In the current implementation of BCI2000, state variables are updated with every sample block. Thus, the size of a sample block determines the maximum temporal resolution of changes in these variables.

Details

As described in the BCI2000 paper¹, states are cycled through the system and stored in the data file. Each module can passively monitor or actively change any state variable. Specifically, all current states (packaged in a data structure called “state vector”) are passed from the Source module to the Signal Processing module on successful acquisition of a block of signal samples. Signal Processing then converts this block of raw signals into a vector of control signals, and passes these, and the updated state vector, on to the User Application module. This module uses these control signals to provide feedback to the user, and encodes the task-relevant variables in the state vector. The updated state vector is then communicated back to the Source module, which stores it together with the next block of signal samples. This in practice unavoidable delay from a change of an event to its update in the data file has implications as described below.

State variables as markers of events of user feedback:

States might be used to encode the status of user feedback. For example, a particular state might encode whether or not a stimulus was presented to the user. Since the delay between successful acquisition of a block of data and feedback update is very small (and for practical purposes negligible), it is appropriate to store state variables with the next block of signal samples, since these were the signals the user produced in reaction to the changed feedback.

State variables as markers of events associated with a data block

In contrast, state variables might also be used to mark events that are associated with a particular block of data. For example, signal processing might employ a procedure that detects artifacts in a given block of signal samples, and that marks an artifact by setting a corresponding state variable. Since the state variable will be stored with the next, as opposed to the current, block of data, offline analyses have to account for this time shift.

¹ Schalk et al., “BCI2000: A General-Purpose Brain-Computer Interface (BCI) System,” IEEE TRANS BIOMED ENG, 2003.

Further Considerations

The purpose of state variables is to store the information that is necessary to reconstruct the session offline. To a certain extent, online events are deterministic since they are defined by a computer program, and the starting configuration is defined by system parameters. Thus, the purist might argue that they are, in most cases, not necessary. However, for practical purposes it proves to be most convenient because the state of the online system does not have to be reconstructed offline, but can rather be determined by evaluating the appropriate state variables.

As mentioned, state variables are intended to reflect events during online operation. They are not intended to provide a mechanism for “global system variables” that are used to transmit and evaluate information during online operation. While this sometimes cannot be completely avoided (this is why we even provide a method for reading state variables), we strongly suggest to avoid such a mechanism as much as possible. Using state variables to communicate information across module boundaries makes these modules interdependent and thus makes module interchange more difficult.