

1 (Work in progress)

1.1 Overview

Project summary; 'Elephant-Nest-Integration', 'Modular Science', 'SIMS-LS'. Figure 1.1 shows an overview, with a detailed description of each component below.

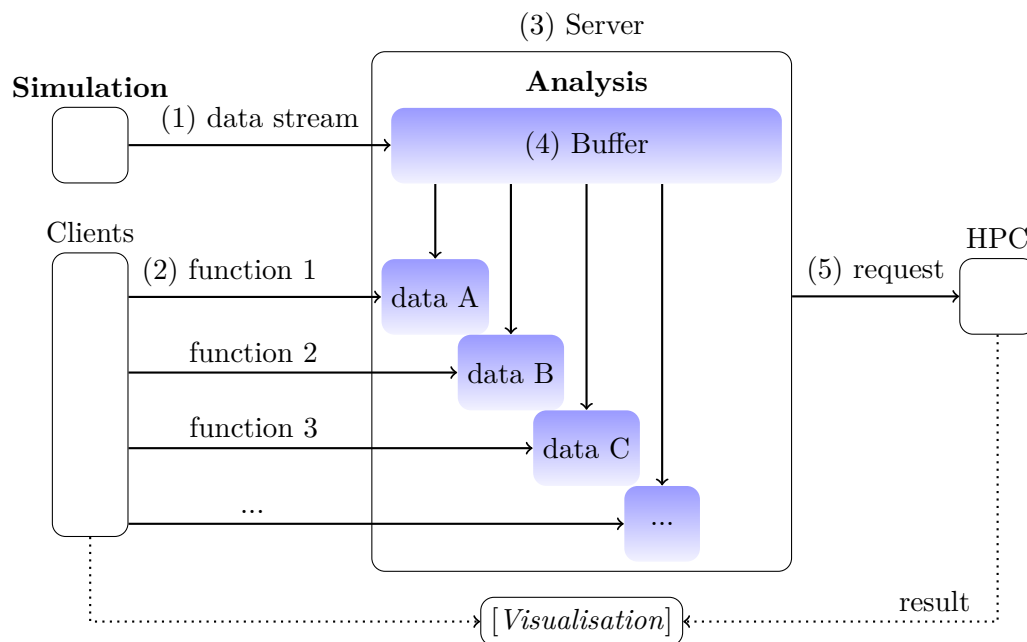


Figure 1.1: Project overview. The Server (3) provides different Elephant functions for analysis. Clients (2) can request those. Data is then taken from the simulation output stream (1), stored in a buffer (4), further processed depending on the function and forwarded to HPC resources (5).

1.2 Details

(1) Input data stream from simulation:

- Protocol -> MUSIC (?)
- Input/Output:
NEST <-> Elephant (, NEST <-> TVB, TVB <-> Elephant)
- Handle input
- Handle data loss

(2) Multiple Clients:

- Multithreading (in Python?) -> multiprocessing
- Protocol

(3) Server:

- Server architecture
- Running where? -> (for now) running on SC (e.g. Jureca)
- Logging

(4) Buffer:

- Buffer-size -> (for now) expect buffer to handle all data, buffer large enough for complete simulation output
- Extract relevant data for each Elephant analysis provided.
- Input conversion -> Simulation output to Elephant input (NEO Framework)

(5) HPC:

- Connect to SC resources -> (for now) server running directly on SC (e.g. Jureca)

2 Elephant functions

List of Elephant functions.

- Statistics:
 - ISI - Inter Spike Intervals
 - Mean Firing Rate
 - CV - Coefficient of Variation (also: LV,CV2)
 - Instantaneous Firing Rate
 - Time Histogram
 - Complexity PDF
- ASSET - Analysis of Sequence of Synchronous Events in massively parallel spikeTrains
- SPADE - spatio-temporal Spike PAttern Detection and Evaluation
- (*UE - Unitary Events*)

NOTE:

- *ASSET:*
 - line 376/377: signal.view(pq.Quantities)[elements_to_keep]*
 - line 631: mask[~(mask >= -np.inf)] = False*
- *Statistics, IFR:*
 - line 754: sskernel['optw'] with 'bootstrap = TRUE'...not needed?*

2.1 Required modules and imports

- Neo Framework
 - imported within elephant
- Numpy
 - imported within elephant
- Quantities
 - » *import quantities as pq*
- Elephant
 - » *import elephant.spike_train_generation as stg*
 - » *import elephant.asset as asset*
 - » *import elephant.statistics as stat*
 - » *import elephant.spade as spade*

2.2 Required input parameters

- Global parameters
 - List of <neo.SpikeTrain> objects
 - > list of neurons
 - Firing rate
 - > Quantities datatype
 - » $rate = 15 * pq.Hz$
 - Length of the Signal/SpikeTrain
 - > Quantities datatype
 - » $T = 1 * pq.s$
 - Binsize or sampling period
 - > Quantities datatype
 - $binsize = 5 * pq.ms$
- Additional parameters per module:
 - Statistics:
 - * Kernel for Instantaneous Firing Rate
 - ASSET:
 - * neo.SpikeTrain objects need additional t_stop attribute if signal length > 1s (standard is 1s).
 - * Build the intersection matrix 'imat' and probability matrix 'pmat':
 - Number of surrogates for the bootstrapping method
 - Window size for spike train dithering
 - * Joint probability matrix 'jmat', using a suitable filter:
 - Filter shape (length, width)
 - Number of largest neighbors
 - * Create from 'pmat' and 'jmat' a masked version of 'imat':
 - alpha1
 - alpha2
 - * Cluster significant elements of 'imat' into diagonal structures:
 - epsilon
 - minimum size
 - stretch
 - SPADE:
 - * Window length
 - * Some optional parameters

NOTE: convert input (numpy array) from simulation to Neo.SpikeTrain and get the needed global parameters

2.3 Profiling

2.3.1 Statistics

Runtime < Signal length, except for IFR

Number of Neurons: 1000

Length of Signal: 1000ms

| % | : | name | : | (count) | : | total time spend |
|----------------------------|---|----------------------|---|---------|---|-------------------|
| ===== | | | | | | |
| 0 | : | start | : | 1 | : | 0.0 |
| 4 | : | DATA | : | 1000 | : | 0.877937078476 |
| 0 | : | ISIs | : | 1000 | : | 0.0740790367126 |
| 2 | : | MFR | : | 1000 | : | 0.33157491684 |
| 0 | : | IFR_start | : | 1000 | : | 0.00199174880981 |
| 3 | : | IFR_sskernel_density | : | 1000 | : | 0.682367086411 |
| 25 | : | IFR_sskernel_optw | : | 1000 | : | 5.00267696381 |
| 14 | : | IFR_rescale | : | 1000 | : | 2.86129784584 |
| 11 | : | IFR_slice | : | 1000 | : | 2.28599452972 |
| 14 | : | IFR_fft | : | 1000 | : | 2.75114107132 |
| 17 | : | IFR_rest | : | 1000 | : | 3.52695941925 |
| 0 | : | IFR_end | : | 1000 | : | 0.00867342948914 |
| 0 | : | CV | : | 1000 | : | 0.06112408638 |
| 4 | : | HIST | : | 1 | : | 0.858650922775 |
| 0 | : | FANO | : | 1 | : | 0.000443935394287 |
| 4 | : | cPDF | : | 1 | : | 0.858818054199 |
| 0 | : | end | : | 1 | : | 0.00200796127319 |
| Total time : 20.1857380867 | | | | | | |

Summary:

Once per neuron:

- DATA: Data will be taken from simulation. No runtime needed for SpikeTrain creation...
- MFR (mean firing rate): 2% of total Statistics runtime.
Neurons: $O(n)$
Signal length: $O(1)$
- IFR (instantaneous firing rate): 85% of total Statistics runtime.
Neurons: $O(n)$
Signal length: $O(n)$
- CV,ISIs: almost no runtime.
Neurons: $O(n)$
Signal length: $O(1)$

Once for all neurons:

- HIST: $\sim O(n)$
- FANO: almost no runtime.
- cPDF: $\sim O(n)$

2.3.2 ASSET

Runtime >> Signal length, $\sim O(n)$

Number of Neurons: 5

Length of Signal: 1000ms

| % | : | name | : | (count) | : | total time spend |
|-------|---|----------------------|---|---------|---|-------------------|
| ===== | | | | | | |
| 0 | : | start | : | 1 | : | 0.0 |
| 0 | : | generate data | : | 1 | : | 0.005774974823 |
| 0 | : | intersec mat | : | 1 | : | 0.0325310230255 |
| 0 | : | prob mat mc | : | 1 | : | 0.363873958588 |
| 0 | : | jmat_pmat_neighb | : | 1 | : | 0.0803229808807 |
| 0 | : | jmat_jsf_uni_init | : | 1 | : | 0.00139403343201 |
| 16 | : | jmat_jsf_uni_diff | : | 1395360 | : | 30.1089406013 |
| 7 | : | jmat_jsf_uni_reshape | : | 31878 | : | 12.6044211388 |
| 7 | : | jmat_jsf_uni_sum | : | 31878 | : | 14.2393944263 |
| 56 | : | jmat_jsf_uni_log | : | 31878 | : | 105.956119537 |
| 8 | : | jmat_jsf_uni_exp | : | 31878 | : | 14.9054570198 |
| 7 | : | jmat_jsf_uni_step | : | 1395360 | : | 12.4127993584 |
| 0 | : | jmat_jsf_uni | : | 1 | : | 5.96046447754e-06 |
| 0 | : | jmat | : | 1 | : | 0.000458002090454 |
| 0 | : | mask | : | 1 | : | 0.000148057937622 |
| 0 | : | cmat | : | 1 | : | 0.0209028720856 |
| 0 | : | extract_sse | : | 1 | : | 6.31809234619e-05 |
| 0 | : | end | : | 1 | : | 5.79357147217e-05 |

Total time : 190.732665062

Summary:

Once for all neurons:

- 100% of runtime for Joint Probability Matrix (jmat)
- Almost no runtime for:
 - Intersection Matrix (imat)
 - Probability Matrix, MonteCarlo (pmat)
 - Mask (from pmat and jmat)
 - Cluster Matrix
 - Extraxt SSE (synchronous spike events)

2.3.3 SPADE

Few Neurons: Runtime < Signal length, $\sim O(n)$

Increasing Neurons: Runtime > Signal length, $\sim O(n^2)$

Increasing Firing Rate: Runtime > Signal length (>90% from fast_fca, $\sim O(n^2)$)

| | Runtime for: 10 Neurons | | Runtime for: 10 s |
|-----|--------------------------------|----|--------------------------------|
| [s] | data mining / p-value spectrum | #N | data mining / p-value spectrum |
| 1 | 0.03 / 0.4 | 10 | 0.09 / 1.1 |
| 10 | 0.09 / 1.1 | 20 | 0.18 / 3.1 |
| 20 | 0.17 / 2.1 | 30 | 0.3 / 12 |
| 30 | 0.24 / 3.8 | 40 | 0.48 / 150 |

```
% :      name      : (count) : total time spend
=====
0 :      import      :      1      : 0.0
0 :      data-gen      :      1      : 0.0178639888763
0 :      data mining    :      1      : 0.000244140625
2 : pvalue: surr_init    :     10      : 0.110061883926
6 : pvalue: con_mining_binning :     11      : 0.257157325745
29 : pvalue: con_mining_context :     11      : 1.34415078163
0 : pvalue: con_mining_max_spike :     11      : 0.00502347946167
0 : pvalue: con_mining_max_occ :     11      : 0.00521540641785
62 : pvalue: con_mining_fast_fca :     11      : 2.81482839584
0 : pvalue: step      :     10      : 0.00246357917786
0 :      spade end      :      1      : 0.000770807266235
Total time : 4.55782604218
```

TODO: Unitary Events?