Reproducible neural network simulations: model validation on the level of network activity data

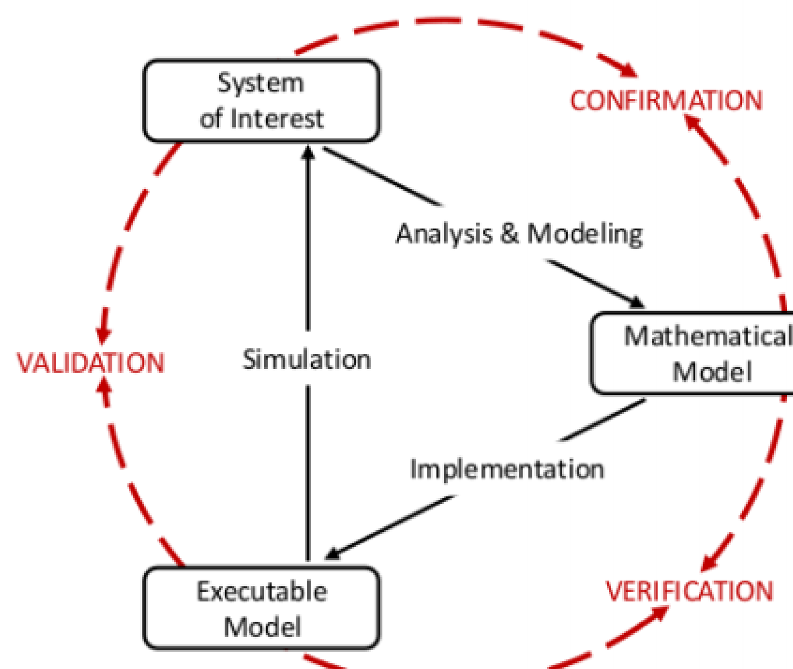
This poster summarizes [5,6].

Robin Gutzen[1], Michael von Papen[1], Guido Trensch[2], Pietro Quaglio[1], Sonja Grün[1,3], Michael Denker[1]

[1] Institute of Neuroscience and Medicine (INM-6), Institute for Advanced Simulation (IAS-6), JARA-Intitute Brain Structure-Function Relationship (INM-10), Jülich Research Centre, Germany
[2] Simulation Lab Neuroscience, Institute for Advanced Simulation, JARA, Jülich Research Centre, Jülich, Germany
[3] Theoretical Systems Neurobiology, RWTH Aachen University, Aachen, Germany

Concept

A model has the purpose to describe and predict its **system of interest**, which is a well defined entity selected for analysis. The model can be separated into two parts. The **mathematical model** is an abstract description formed by analysis and observation. The **executable model** is the implementation of the mathematical model which can perform simulations and thus generate testable predictions. [1][2]



Adapted from [1,2]

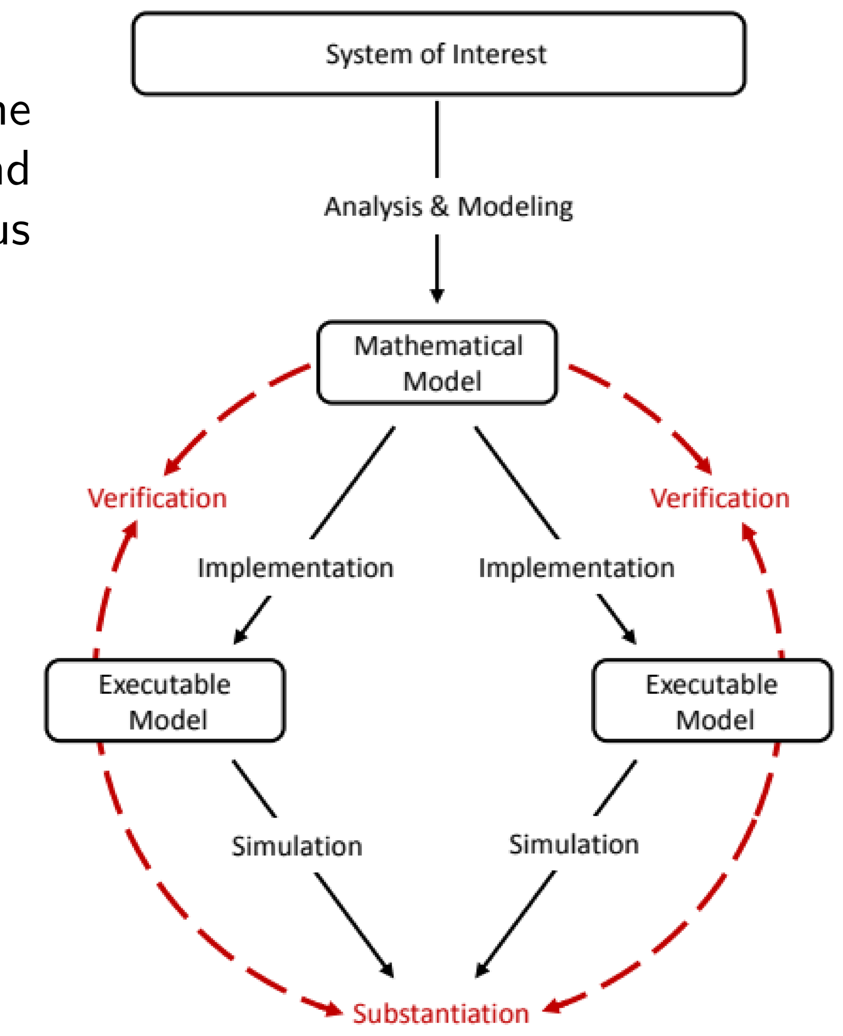
Confirmation: Assesses the plausibility of modeling choices and premises.

Verification: Ensures that the implementation is a correct representation of the mathematical model, concerning the code and the calculations.

Validation: Establishes confidence in the model by testing whether its prediction accuracy is within an acceptable agreement to the system of interest.

Substantiation: Defined here as the evaluation and quantifying the level of agreement between two executable models.

In practice the substantiation performs equivalent tests as a validation and differs only in its interpretation.



Validation approach: We validate by evaluating the simulation outcome on the **level of the network activity**, as opposed to the complementary approach of validating on a **single-cell level.**

Model & Implementations

Mathematical Model

polychronization network model: (*Izhikevich* [3])
800 exc. / 200 inh. spiking neurons
random connectivity, out-degree 100, random input,
integer delays, trained with STDP, measured without STDP

Implementation — Implementation

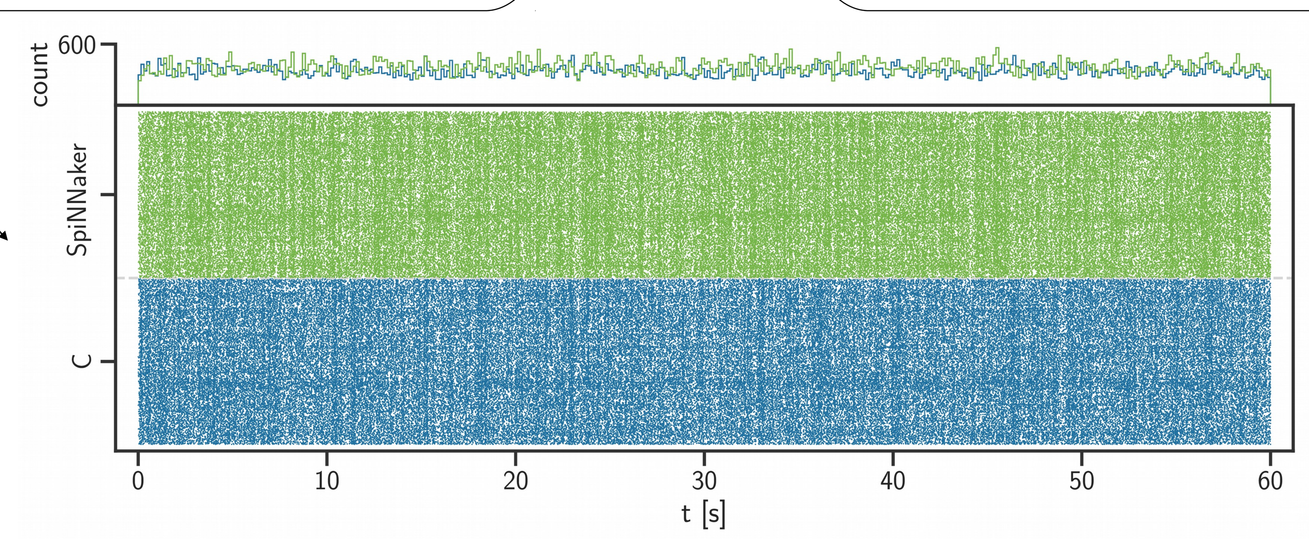
Executable Model

custom C code:
original publication [3], refactored,
adapted to make it transferable to SpiNNaker,
1ms time steps

Executable Model

SpiNNaker neuromorphic system: [4]
brain-inspired digital computer architecture,
uses fixed point representation of variables [5]
3 iterative stages of the implementation (I,II,III)

Simulation — Simulation



→ compare statistical measures:
firing rates (FR),
local coefficient of variation (LV),
correlation coefficients (CC),
inter-spike intervals (ISI),
rate correlations (RC), ...

Python Validation Framework

developed at github.com/INM-6/NetworkUnit

We developed the module **NetworkUnit** as a formalized Validation test framework based on *Elephant* [7] and *SciUnit* [8] for validation and substantiation in experiment and simulation.

```
import networkunit as newt
import sciunit
```

```
class polychonization_model(sciunit.Model, newt.capabilities.ProducesSpiketrains):
    def load():  # loads simulation outcome
        # ...

class C_model(polychronization_model):
    file_path = './simulation_data/C/'

class SpiNNaker_model(polychronization_model):
    file_path = './simulation_data/SpiNNaker/'

C = C_model()
S = SpiNNaker_model()
```

```
class rate_test(sciunit.TestM2M, newt.tests.firing_rate_test):
    score_type = newt.scores.effect_size  # equips the test with a score

FR_test = rate_test()
```

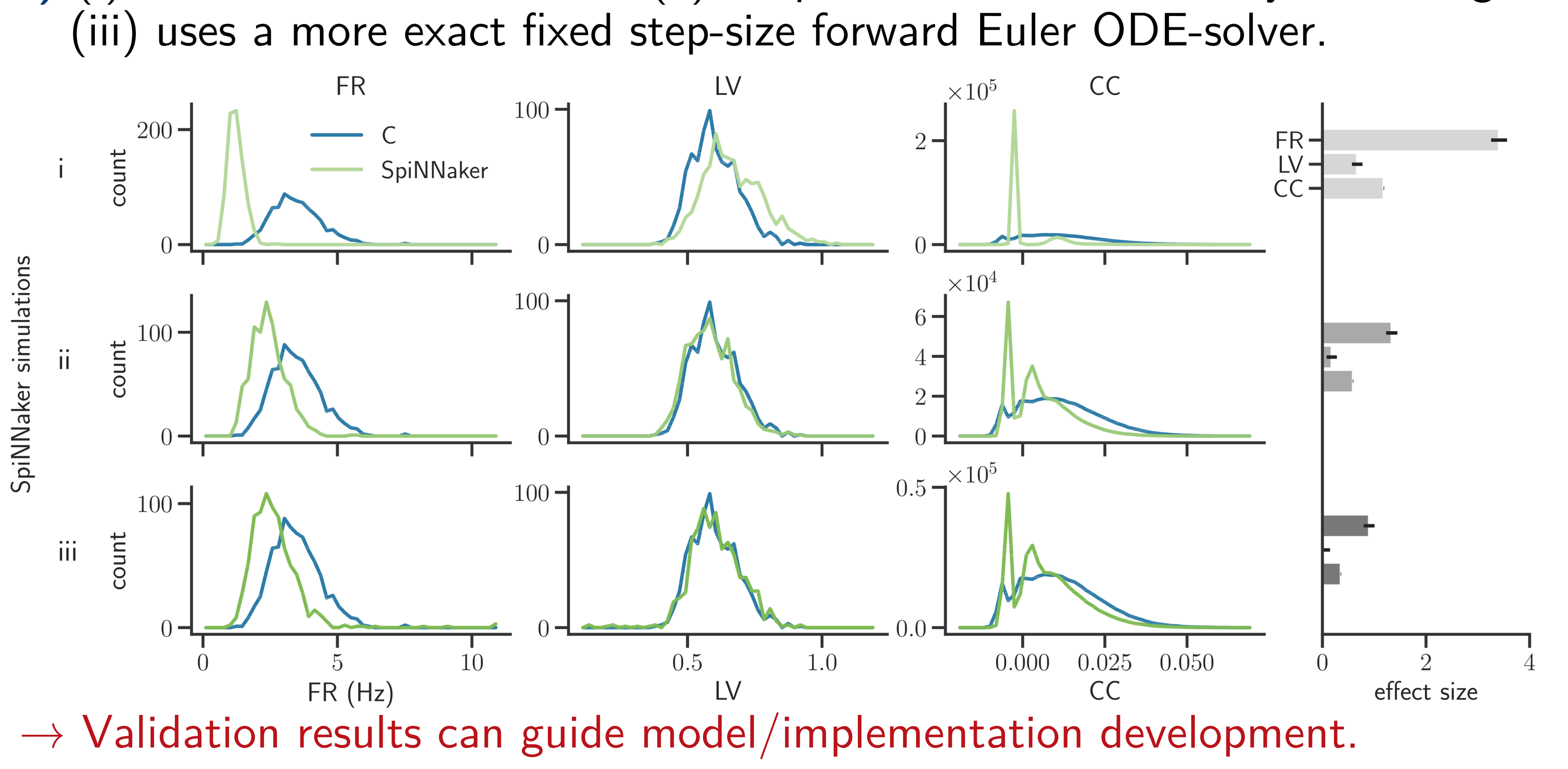
```
FR_test.judge([C,S])  # performs the validation test
```

```
Effect Size
        datasize: 800    800
        Effect Size = 0.394      CI = (0.295, 0.493)
```

Substantiation by Iterative Validation Test Cycles (I, II, III)

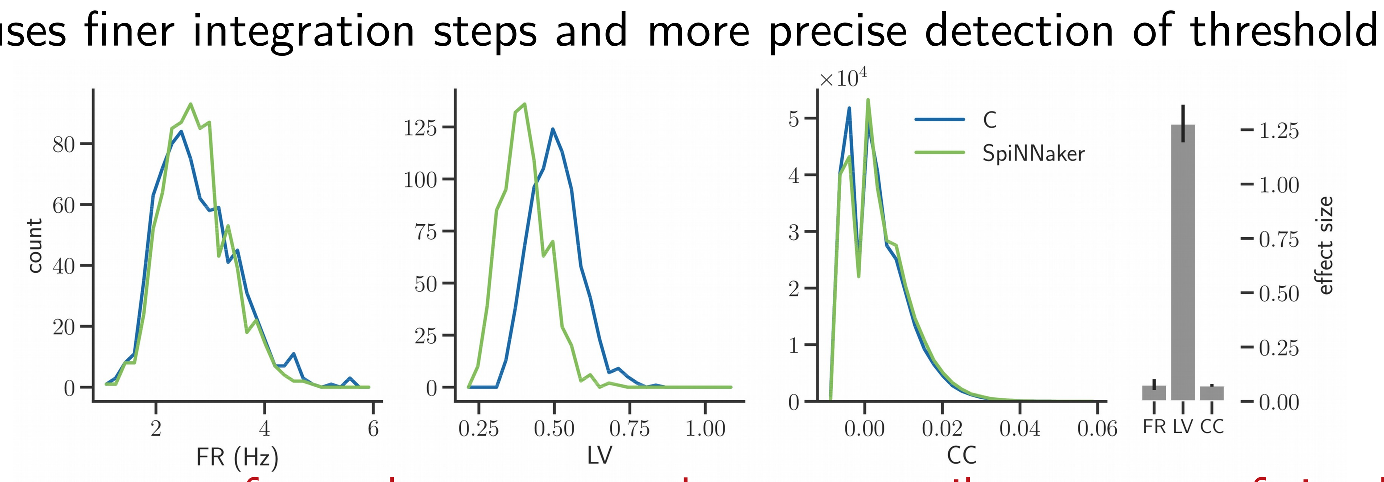
data, simulation code, and analysis code published at web.gin.g-node.org/INM-6/network_validation

I) (i) uses an ESR ODE-solver, (ii) adapts Izhikevich's neural dynamics algorithm, (iii) uses a more exact fixed step-size forward Euler ODE-solver.



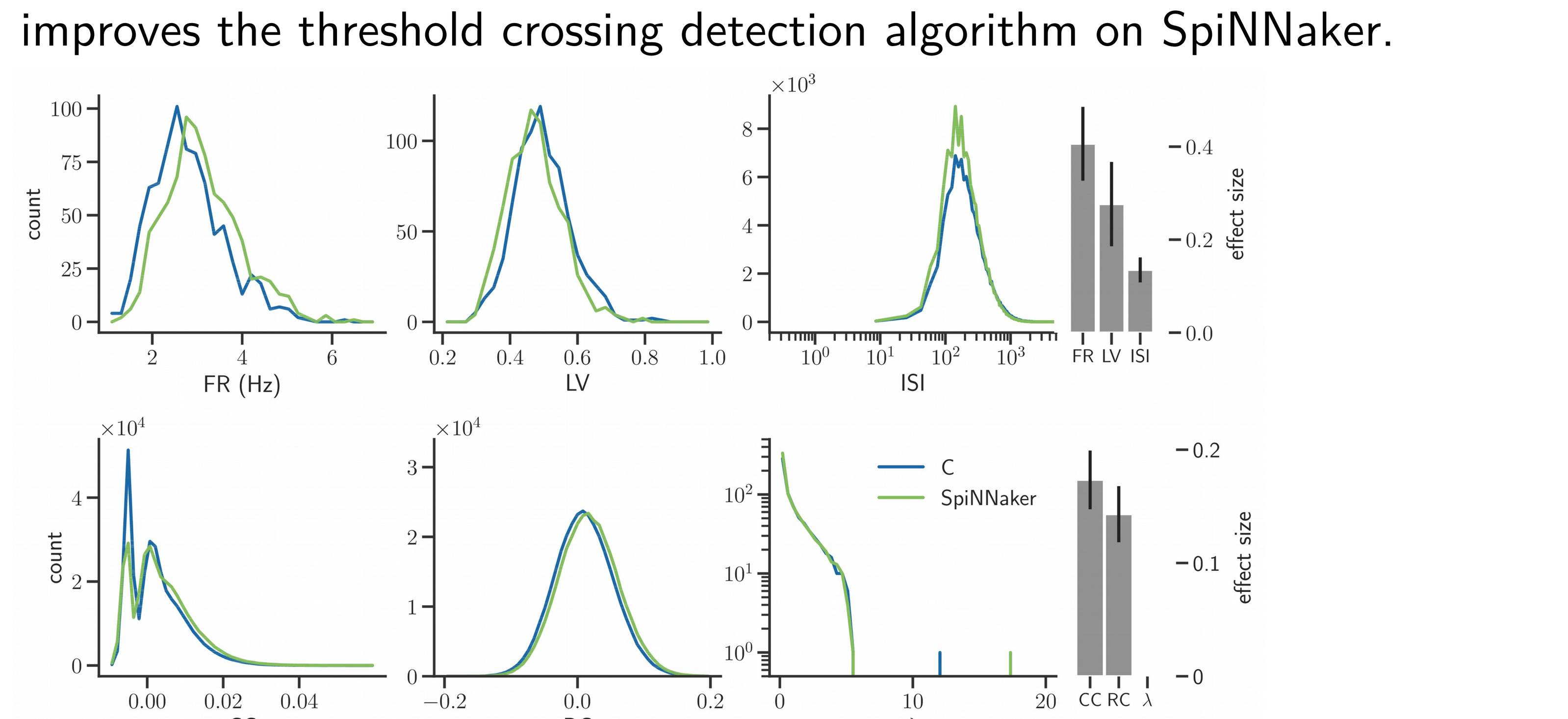
→ Validation results can guide model/implementation development.

II) uses finer integration steps and more precise detection of threshold crossing.

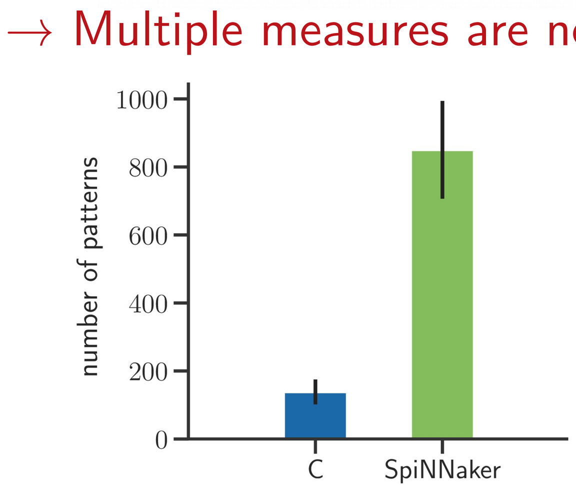


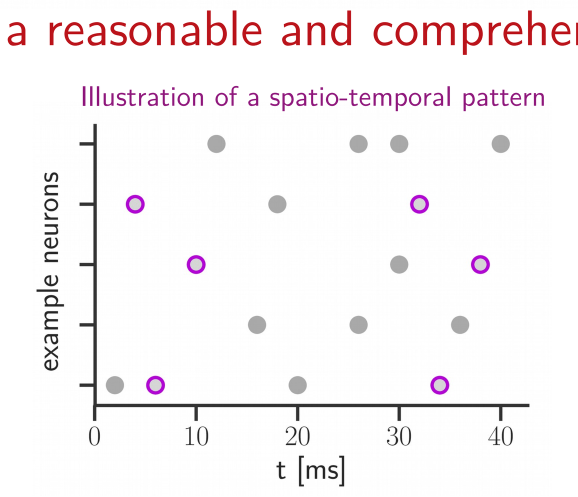
→ Agreement of complex measures does not entail agreement of simpler measures.

III) improves the threshold crossing detection algorithm on SpiNNaker.



→ Multiple measures are needed for a reasonable and comprehensive validation.





→ The evaluation of the level of agreement depends on the intended application. Here, despite good agreement of other measures, complex measures such as the pattern density (detected with SPADE [9]) is not yet consistent.

References

[1] Schlesinger, S. (1979). *Terminology for model credibility.* Simulation 32, 103–104
[2] Thacker, B. et al. (2004) *Concepts of model verification and validation.* Tech. rep., Los Alamos National Lab
[3] Izhikevich, E. M. (2006) *Polychronization: Computation with spikes,* Neural Computation 18, 245–282
[4] Furber, S. et al. (2013) *Overview of the SpiNNaker system architecture.* IEEE Transactions on Computers 62
[5] Trensch G. et al. (2018) *Rigorous neural network simulations: model cross-validation for boosting the correctness of simulation results,* under review in Frontiers of Neuroinformatics
[6] Gutzen R. et al (2018) *Reproducible neural network simulations: model validation on the level of network activity data,* under review in Frontiers of Neuroinformatics
[7] Electrophysiology Analysis Toolkit (python-elephant.org)
[8] SciUnit (scidash.github.io)
[9] Quaglio, P. et al. (2017) *Detection and evaluation of spatio-temporal spike patterns in massively parallel spike train data with SPADE,* Frontiers in Computational Neuroscience 11, 41

Acknowledgments:
This project has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under Specific Grant Agreements No. 720270 (Human Brain Project SGA1) and 785907 (Human Brain Project SGA2)