Test-Verified Gaussian Entanglement Thresholds for a Vorticity-Inspired TMST Model and Readiness for Execution on BasQ's IBM Quantum System Two (Heron Processor)

Javier Manuel Martín Alonso
(Independent Theoretical Physics Researcher)
jmma@movistar.es

February 2026

Abstract

We present a mathematically explicit and software-verified pipeline for detecting entanglement in a symmetric two-mode squeezed thermal state (TMST) model used as a controlled proxy for vorticity-driven correlations. The core physics claims are validated by a dedicated pytest suite and a deterministic toy benchmark that reproduces the expected phase boundary between separability and entanglement, including the Simon/PPT criterion and logarithmic negativity. We further document how the repository is structured to transition from purely classical verification to quantum-hardware execution using Qiskit and Qiskit Aer, with tests that automatically skip when quantum dependencies are unavailable. Finally, we outline a hardware-facing execution plan targeting the IBM Quantum System Two installation in Donostia–San Sebastián (BasQ initiative) featuring an IBM Quantum Heron processor.

1 Scope and contributions

This preprint is intentionally narrow: it is a verification and deployment-readiness note anchored to reproducible tests and a minimal analytic model. The novelty is not a new entanglement measure but a tested, end-to-end bridge from analytic Gaussian-state entanglement thresholds to a hardware-facing software interface that is ready to activate once Qiskit/Qiskit-Aer and credentials are available.

Repository verification snapshot. A representative test run executed in a Python 3.14 environment reports *64 passed*, *1 skipped*, and *1 warning* for the Gaussian test suite (with a single Qiskit-dependent file ignored in that run). A deterministic toy benchmark script prints numerical TMST diagnostics at "maximum vorticity" parameter settings and asserts a strongly entangled regime.

2 Mathematical framework (Gaussian entanglement)

We work with two bosonic modes with canonical quadratures $\hat{\mathbf{R}} = (\hat{x}_1, \hat{p}_1, \hat{x}_2, \hat{p}_2)^\mathsf{T}$ satisfying $[\hat{R}_j, \hat{R}_k] = i\Omega_{jk}$, where $\Omega = \bigoplus_{m=1}^{2} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$. For Gaussian states, the covariance matrix is

$$V_{jk} = \frac{1}{2}\langle \hat{R}_j\hat{R}_k + \hat{R}_k\hat{R}_j\rangle - \langle \hat{R}_j\rangle\langle \hat{R}_k\rangle.$$

2.1 Symmetric TMST

Definition 1 (Symmetric TMST)**.** *Let $\bar{n} \geq 0$ be a thermal occupation parameter and $r \geq 0$ a two-mode squeezing parameter. The symmetric TMST covariance matrix can be written in block form*

$$V(r, \bar{n}) = \begin{pmatrix} a\,\mathbb{I}_2 & c\,\sigma_z \\ c\,\sigma_z & a\,\mathbb{I}_2 \end{pmatrix}, \quad a = \left(\bar{n} + \frac{1}{2}\right)\cosh(2r), \quad c = \left(\bar{n} + \frac{1}{2}\right)\sinh(2r),$$

where $\sigma_z = \mathrm{diag}(1, -1)$.

2.2 Partial transpose and the Simon/PPT criterion

Partial transposition at the covariance-matrix level is implemented by flipping the sign of one momentum quadrature, e.g. $\hat{p}_2 \mapsto -\hat{p}_2$, which corresponds to $V \mapsto \tilde{V} = \Lambda V \Lambda$ with $\Lambda = \mathrm{diag}(1, 1, 1, -1)$. The symplectic eigenvalues $\tilde{\nu}_\pm$ of $\tilde{V}$ determine separability.

Theorem 1 (Simon criterion for 1×1 Gaussian states)**.** *A two-mode Gaussian state is separable if and only if the partially transposed covariance matrix $\tilde{V}$ satisfies the uncertainty relation $\tilde{V} + \frac{i}{2}\Omega \geq 0$, equivalently $\tilde{\nu}_- \geq \frac{1}{2}$.*

For the symmetric TMST defined above, one obtains the closed form

$$\tilde{\nu}_- \;=\; \left(\bar{n} + \frac{1}{2}\right) e^{-2r}.$$

Therefore the entanglement boundary is

$$\tilde{\nu}_- = \frac{1}{2} \quad \Longleftrightarrow \quad r = r_c(\bar{n}) := \frac{1}{2}\ln(2\bar{n} + 1).$$

This is the analytic threshold tested numerically across a temperature grid in the project test suite.

2.3 Logarithmic negativity

The logarithmic negativity for Gaussian states can be expressed via $\tilde{\nu}_-$ as

$$E_N \;=\; \max\left\{0, -\log_2\left(2\tilde{\nu}_-\right)\right\}.$$

In the symmetric TMST case, $E_N > 0 \iff r > r_c(\bar{n})$, and E_N grows monotonically with squeezing r for fixed $\bar{n}$, which is explicitly validated by the test suite.

3 Thermal parameterization and numerical stability

The test and toy runs parameterize thermal noise through a Bose–Einstein occupation function of the form

$$\bar{n}(T) = \frac{1}{e^{\omega/T} - 1},$$

with a fixed frequency scale ω (dimensionless units in the toy model).

Remark 1 (Overflow warning in low-temperature limit)**.** *A pytest run reports a `RuntimeWarning: overflow encountered in exp` in the implementation of $\bar{n}(T)$ for the zero-temperature limit test. This is numerically expected when ω/T is large; stable implementations typically use guarded exponentials or asymptotic forms for $\bar{n}(T)$ at low T.*

4 Software verification results (tests as reproducible evidence)

4.1 Pytest suite: what is verified

The repository contains a documented test suite structure (pytest) covering entanglement measures and related criteria. In addition, the shared pytest configuration includes (i) deterministic random seeding for reproducibility and (ii) a session-scoped fixture that checks availability of `qiskit` and `qiskit-aer` to enable graceful skipping of quantum-hardware tests when dependencies are missing.

4.2 Representative test run snapshot

A representative run executes:

```
py -m pytest test/ -v --ignore=test/test_injection_qiskit.py
```

and reports `collected 64 items / 1 skipped` with final status `64 passed, 1 skipped, 1 warning in 2.74s`. The skip is consistent with an environment lacking Qiskit dependencies and is handled via the fixture approach.

4.3 Toy benchmark: "maximum vorticity" regime

A standalone script is executed as:

```
py run_toy_test.py
```

and prints the following numerical diagnostics:

$$\begin{aligned} T &= 0.1, \qquad & r &= 1.8, \\ \bar{n}(T) &= 0.000045, \qquad & r_c(T) &= 0.000045, \\ \tilde{\nu}_- &= 0.013663, \qquad & E_N &= 5.193571, \end{aligned}$$

followed by ✓ `TODOS LOS ASSERTS PASADOS`. These values imply $\tilde{\nu}_- < 1/2$ and $E_N > 0$, i.e., an entanglement-dominant regime well above the analytic threshold r_c.

4.4 Summary table (from logs)

4.5 Test inventory (from logs)

Artifact	Scope / modules exercised	Result
Toy benchmark (script)	`run_toy_test.py` (standalone; not collected by pytest)	All asserts passed; TMST diagnostics printed
Gaussian suite (no Qiskit file)	`test/gaussian/test_entanglement.py` + `test/gaussian/test_tmst_injection.py`	`64 passed, 1 skipped, 1 warning`
Full suite (Qiskit deps missing)	Gaussian suite + `test/test_injection_qiskit.py` (5 tests skipped if Qiskit absent)	`64 passed, 6 skipped, 1 warning`
Entanglement-only slice	`test/gaussian/test_entanglement.py` + Qiskit injection tests (skipped)	`32 passed, 6 skipped`
TMST-threshold-only slice	`test/gaussian/test_tmst_injection.py` + Qiskit injection tests (skipped)	`32 passed, 6 skipped, 1 warning`

Table 1: Inventory of executed test blocks and outcomes reproduced from the captured console logs.

5 Readiness for IBM Quantum System Two (Donostia, BasQ)

5.1 Execution target and software stack

Project documentation states that an IBM Quantum System Two is to be installed in Donostia–San Sebastián under the BasQ–Basque Quantum initiative, and that it uses an IBM Quantum Heron processor. The same document identifies Qiskit (and related runtime components) as the Python software layer used to construct and submit quantum circuits to IBM Quantum hardware.

5.2 Test-gated transition from classical to quantum execution

The test infrastructure explicitly introduces a marker `quantum` and a `qiskit_available` fixture that returns `True` only if both `qiskit` and `qiskit-aer` import successfully, enabling automatic skipping rather than hard failures in environments without quantum tooling. In the captured runs, Qiskit-dependent tests are skipped with messages indicating `qiskit / qiskit-aer not installed`, demonstrating that the suite is already structured to "turn on" quantum execution once the dependencies are present.

5.3 Implementation plan (hardware-facing)

The intended workflow is:

1. Verify analytic Gaussian entanglement thresholds and monotonicity properties in the TMST proxy model (already covered by pytest).
2. Use injection-style tests (skipped when Qiskit is unavailable) to validate that the same entanglement witnesses and measures remain robust when realized through circuit-based simulation and (later) hardware execution.
3. Replace simulator backends with the BasQ System Two hardware backend once credentials and runtime access are configured, keeping the same test contracts (assertions) to detect regressions in physics or implementation.

6 Reproducibility checklist

- Run the full suite (current layout in logs): `py -m pytest test/ -v`.
- To display printouts inside pytest runs, use `-s` (stdout capture disabled).
- Toy benchmark: `py run_toy_test.py`.
- Quantum-facing tests: install `qiskit` and `qiskit-aer`; verify that `qiskit_available` becomes `True` and previously skipped tests execute.

Acknowledgements

his project uses the SeeMPS library for tensor-network and matrix product state (MPS) / tensor-train algorithms [4]. The quantum-inspired numerical-analysis perspective adopted in parts of this work follows the framework developed in [5]. We also acknowledge the public SeeMPS/SeeMPS2 software repositories for enabling reproducible implementations and testing.

References

[1] R. Simon, *Peres–Horodecki separability criterion for continuous variable systems*, Phys. Rev. Lett. **84**, 2726 (2000).

[2] G. Vidal and R. F. Werner, *Computable measure of entanglement*, Phys. Rev. A **65**, 032314 (2002).

[3] G. Adesso and F. Illuminati, *Entanglement in continuous-variable systems: recent advances and current perspectives*, J. Phys. A: Math. Theor. **40**, 7821 (2007).

[4] P. García-Molina, J. J. Rodríguez-Aldavero, J. Gidi, and J. J. García-Ripoll, *SeeMPS: A Python-based Matrix Product State and Tensor Train Library*, arXiv:2601.16734 [quant-ph] (2026). https://arxiv.org/abs/2601.16734

[5] J. J. García-Ripoll, *Quantum-inspired algorithms for multivariate analysis: from interpolation to partial differential equations*, Quantum **5**, 431 (2021). https://doi.org/10.22331/q-2021-04-15-431 (See also https://quantum-journal.org/papers/q-2021-04-15-431/.)

[6] J. J. García-Ripoll et al., *SeeMPS2 (software repository)*, GitHub repository. https://github.com/juanjosegarciaripoll/seemps2 (Accessed: 2026-02-20).