Quantum Information Dynamics of the Source and the Placed Source

Lindblad Dynamical Scission, von Neumann Entropy Production, and Landauer-Bounded Quantum-Classical Transduction

Verification Code Companion

Charles D. Schaper, Ph.D.
Annals of the Chrysene Formalism, Volume 1, Number 4
Article 96, November 2026

1 Scope and reproducible result

The replacement program executes **93,858 explicit checks per mode**: 93,767 exact rational/integer checks and 91 numerical comparisons. Ten negative controls are included in the total. Normal and optimized Python produce byte-identical outputs. Checks remain active under `python -O`; only the standard library is used. The count measures validation calls, not independent theorems.

These are finite matrix-unit tests, bounded registry enumerations and numerical benchmarks. They do not establish molecular realization, general semigroup convergence, the complete-positive property of an unspecified physical map, general entropy inequalities, or experimental coherence times. The manuscript gives the mathematical assumptions and analytic arguments separately.

2 Defects in the inherited artifacts

The supplied Python program fails its first coherence-rate assertion: its sink set omits the inert states $(0, \kappa)$ and $(1, \kappa)$. The source network has seven zero-outgoing-rate states. A coherence between a retained column sink and a washout sink does not decay under the bare jump generator. The old all-initial-state scission claim therefore needs the manuscript's explicit sector dephaser or a restricted initial block.

The printed one-index twirl is not trace preserving: on E_{11} it has trace zero. It differs from the double-index map used in the source proof. The old script simply assumed the vanishing of off-diagonal outputs instead of testing that displayed map. The new program checks the corrected map and supplies a counterexample to both the one-index formula and replacement-channel interpretation of an X-only twirl.

The old fixed retention value 0.998437 is not derived by that program. For the explicit illustrative competing-hazard rates here, the retained fraction is exactly

$$\left(\frac{10000}{10001}\right)^2 = \frac{100000000}{100020001} \approx 0.99980003.$$

This is a model calculation, not measured chemical purity. Several inherited thermal rounding targets were also wrong. The printed verification dossier and supplied Python file were not identical; the final listing below is the executed source.

3 Coverage

Q1: source jumps.
Twelve rank-one jumps on 15 labelled states are multiplied on every one of the 225 matrix units. Their sum agrees with the population/coherence generator formula, and every output has zero trace. The optional admitted-sector dephaser has coefficient $-1/2$ on each cross-sector unit and zero within a sector. Backwards rational recursion independently computes the illustrative retention probability.

Q2: replacement channel.
Every matrix unit of the 32-dimensional registry space is passed through dephasing after the uniform X twirl. Outputs equal $\delta_{ij}I/32$, with correct trace. Kraus completeness is checked. An X-only twirl retains the plus-state coherence and is not a replacement channel.

Q3 and Q5: agreement and protection.
The agreement permutation is tested at every site word for lengths 2 through 8, with global-flip and nearest-neighbour parity identities. Every site-space matrix unit in those lengths has unchanged registry partial trace after the collective flip, including units representing correlations. Five-bit reversal gives eight fixed words and twelve pairs. These are combinatorial classes, not independently verified molecular chirality classes. Phase noise fixes diagonal pointer states but need not preserve arbitrary quantum registry coherences.

Q4 and Q6: numerical benchmarks.
Thermal threshold inversion is checked with the nonnegative-gap clamp. Relative-entropy identities are sampled on pure, uniform and rank-two diagonal states. The illustrative independent bit-flip survival law is tested at several times and lengths; phase noise on a computational-basis pointer state instead has fidelity one. Thermodynamic work interpretations require the manuscript's endpoint Hamiltonian and bath assumptions.

At 300 K, the corrected five-contact ratios for 95% and 99% fidelity are 4.5745 and 6.2086, the splittings are 118.26 and 160.50 meV, and the corresponding coding informations are 4.5892 and 4.8956 bits. The transit illustration is 0.9754. All decimals are numerical evaluations, not exact formal certificates.

4 Negative controls and reproduction

Ten false claims are rejected: the old sink count, universal bare coherence damping, inherited yield, one-index trace preservation, X-only replacement, phase-noise quantum protection, obsolete DFS dimension, two incorrect threshold ratios and one incorrect capacity. Run `python run_verification.py`; it verifies the pinned source hash, executes both modes in fresh temporary directories and compares every output byte. Macros are generated from tested data. The script and audit records accompany the package.

5 Complete executable source

Article 96 verification source

```
#!/usr/bin/env python3
"""Article 96 finite and numerical audit. Standard library only.
Exact checks use Fraction; floating checks are labelled numerical.
No assertion statements: checks remain active with python -O.
"""
from fractions import Fraction as F
from itertools import product
import json, math
checks = exact = numerical = negatives = 0

def check(ok, label, numeric=False):
    global checks, exact, numerical
    checks += 1
    if numeric: numerical += 1
    else: exact += 1
    if not ok: raise RuntimeError(label)

def reject(false_claim, label, numeric=False):
    global negatives
    check(not false_claim, label, numeric)
    negatives += 1

def add(A, B):
    out = dict(A)
    for k, v in B.items(): out[k] = out.get(k, F(0)) + v
    return {k:v for k,v in out.items() if v}

def mul(A, B):
    out = {}
    for (i,j), a in A.items():
        for (k,l), b in B.items():
            if j == k: out[i,l] = out.get((i,l), F(0)) + a*b
    return {k:v for k,v in out.items() if v}

def scale(A, c): return {k:v*c for k,v in A.items() if v*c}
def unit(i,j): return {(i,j):F(1)}
def trace(A): return sum(v for (i,j),v in A.items() if i == j)

# Q1. Explicit 15-state jump network, exact rates and absorption.
states = [(n,p) for n in range(5) for p in ('b','s','k')]
adm = {states.index((n,p)) for n in (2,3,4) for p in ('s','k')}
e = [F(1,1000),F(1,1000),F(1),F(1,10),F(1)]
k = [F(10),F(10),F(208,42354)*F(11,10),F(1),F(0)]
jumps = []
for n in range(5):
    b = states.index((n,'b')); s = states.index((n,'s'))
    jumps.append((s,b,e[n]))
    if n < 4: jumps.append((states.index((n+1,'b')),b,k[n]))
    if n >= 2: jumps.append((states.index((n,'k')),s,F(1)))
out = [sum(r for u,y,r in jumps if y==i) for i in range(15)]
sinks = {i for i,r in enumerate(out) if r == 0}
check((len(adm),len(sinks),len(jumps)) == (6,7,12), 'state partition')
check(min(r for r in out if r) == 1, 'positive transient exit minimum')
# Independently multiply rank-one matrices on every matrix unit.
for i,j in product(range(15), repeat=2):
    E=unit(i,j); actual={}
    for u,y,r in jumps:
        L=unit(u,y); Lt=unit(y,u); P=mul(Lt,L)
        D=add(scale(mul(mul(L,E),Lt),r),
              scale(add(mul(P,E),mul(E,P)),-r/2))
        actual=add(actual,D)
    expected=scale(E,-(out[i]+out[j])/2)
    if i==j:
```

```
        for u,y,r in jumps:
            if y==i: expected=add(expected,{(u,u):r})
    check(actual==expected,'rank-one Lindblad matrix unit')
    check(trace(actual)==0,'trace preservation')
    # Additional sector dephaser: Q E Q - {Q,E}/2.
    a,b=int(i in adm),int(j in adm)
    coeff=F(a*b)-F(a+b,2)
    check(coeff == (-F(1,2) if a!=b else F(0)), 'sector dephaser')
A=states.index((2,'k')); R=states.index((0,'s'))
check(out[A]+out[R]==0,'persistent cross-sink coherence')
# Backwards recursion for probability of eventual retained kappa state.
p=[F(0)]*5; p[4]=F(1)
for n in range(3,-1,-1):
    p[n]=(k[n]*p[n+1]+e[n]*int(n>=2))/(k[n]+e[n])
yield_retained=F(10000,10001)**2
check(p[0]==yield_retained,'exact illustrative retention')
reject(len(sinks)==5,'old sink set')
reject(out[A]+out[R]>0,'universal bare coherence decay')
reject(p[0]==F(998437,1000000),'unsupported inherited yield')

# Q2. Correct double-index dephasing-after-X-twirl, on all units.
d=32
for i,j in product(range(d), repeat=2):
    output=[F(0)]*d
    for shift in range(d):
        a,b=i^shift,j^shift
        if a==b: output[a]+=F(1,d)
    expected=[F(int(i==j),d)]*d
    check(output==expected,'replacement channel matrix unit')
    check(sum(output)==int(i==j),'replacement trace')
# Kraus K_ab=E_ab/sqrt(d): sum K* K = I, Choi=I/d.
for j in range(d):
    check(sum(F(1,d) for a in range(d))==1,'Kraus completeness')
# Actual defective one-index source formula has output rho00 I/d.
bad_trace=sum(F(1,d) for shift in range(d) if (1^shift)==shift)
reject(bad_trace==1,'one-index twirl preserves trace of E11')
# X twirl alone fixes |+><+|; it is not a replacement channel.
plus={(i,j):F(1,2) for i,j in product(range(2),repeat=2)}
xonly={(i,j):sum(plus[i^r,j^r]/2 for r in range(2)) for i,j in plus}
reject(xonly[0,1]==0,'X-only twirl erases off-diagonal plus state')

# Q3/Q5. Agreement permutation and exact registry partial trace.
def bits(w,l): return [(w>>(l-1-i))&1 for i in range(l)]
def agr(w,l):
    b=bits(w,l); z=b[0]
    for i in range(l-1): z=2*z+(b[i]^b[i+1])
    return z
for l in range(2,9):
    D=1<<l; reg=D//2
    check(sorted(agr(w,l) for w in range(D))==list(range(D)), 'agreement bijection')
    for w in range(D):
        z=agr(w,l)
        check(agr(w^(D-1),l)==(z^reg),'collective frame flip')
        for r in range(l-1):
            check((bits(w,l)[r]^bits(w,l)[r+1])==bits(z,l)[r+1], 'ZZ registry factor')
    # Matrix-unit partial trace after global flip, including correlated inputs.
    for w,v in product(range(D),repeat=2):
        z,t=agr(w,l),agr(v,l)
        before={(z%reg,t%reg):1} if z//reg==t//reg else {}
        z2,t2=agr(w^(D-1),l),agr(v^(D-1),l)
        after={(z2%reg,t2%reg):1} if z2//reg==t2//reg else {}
        check(before==after,'DFS partial trace matrix unit')
# Reversal orbits in five registry bits.
rev=lambda x:int(format(x,'05b')[::-1],2)
check(sum(rev(x)==x for x in range(32))==8,'palindromic words')
check(sum(rev(x)!=x for x in range(32))//2==12,'reversal pairs')
# Z noise fixes basis states but damps off-diagonal registry units.
for i,j in product(range(32),repeat=2):
```

```
    phase=(-1)**((i&1)+(j&1))
    if i==j: check(phase==1,'phase-noise pointer fixed point')
reject((-1)**1==1,'registry quantum coherence survives Z noise')
reject(32==1024,'obsolete DFS dimension')

# Q4/Q6. Numerically evaluated thermodynamic and bit-flip benchmarks.
kBT=8.617333262e-5*300*1000
land1=kBT*math.log(2); land5=5*land1
Ec=1.602176634e-19/(2e-19)*1000

def threshold(delta,n):
    return max(0.0,-math.log(math.expm1(-math.log1p(-delta)/n)))
def entropy(q): return -sum(x*math.log(x) for x in q if x)
def h2(x): return -x*math.log2(x)-(1-x)*math.log2(1-x)
bench={}
for delta in (0.05,0.01):
    r=threshold(delta,5); eps=1/(1+math.exp(r)); fid=(1-eps)**5
    cap=5*(1-h2(eps)); bench[delta]=(r,r*kBT,cap)
    check(abs(fid-(1-delta))<1e-13,'thermal threshold inversion',True)
    check(0<cap<5,'coding information range',True)
for n in range(1,9):
    for delta in (0.01,0.1,0.5,0.99):
        r=threshold(delta,n)
        check((1+math.exp(-r))**(-n)>=1-delta-1e-13,'nonnegative gap threshold',True)
    for t in (0.0,0.001,0.01,0.1,1.0,10.0):
        q=(1+math.exp(-t))/2
        check(0<=q**n<=1,'bit-flip survival bounds',True)
for q in ([1]+[0]*31,[F(1,32)]*32,[F(1,2)]*2+[0]*30):
    D=sum(float(x)*math.log(float(x)*32) for x in q if x)
    check(abs(D-(math.log(32)-entropy(q)))<1e-12,'relative entropy identity samples',
        True)
transit=((1+math.exp(-0.01))/2)**5
check(abs(transit-0.97537087)<1e-7,'transit benchmark',True)
reject(abs(bench[0.05][0]-4.5744)<5e-5,'inherited 95 percent ratio', numeric=True)
reject(abs(bench[0.01][0]-6.2076)<5e-5,'inherited 99 percent ratio', numeric=True)
reject(abs(bench[0.05][2]-4.5965)<5e-5,'inherited coding capacity', numeric=True)
values={'valSourceStates':15,'valAdmittedStates':6,'valRejectedStates':9,
'valEtaTotal':f'{float(yield_retained):.8f}','valGammaCohMin':'0.5000',
'valRegDimSix':32,'valSiteDimSix':64,'valMaxVnEntropyBits':5,
'valMaxVnEntropyNats':f'{5*math.log(2):.4f}','valHolevoEndo':0,
'valRelEntPureBits':5,'valRelEntMesoBits':5,'valRelEntChiralBits':4,
'valKbTmeV':f'{kBT:.2f}','valLandauerOneBitMeV':f'{land1:.2f}',
'valLandauerFiveBitsMeV':f'{land5:.2f}','valChargingEnergyMeV':f'{Ec:.2f}',
'valDfsCommutatorZeros':'4,096','valTransitFidelitySix':f'{transit:.4f}'}
for delta,name in [(0.05,'NinetyFive'),(0.01,'NinetyNine')]:
 r,mev,cap=bench[delta]
 values['valStarkRatio'+name]=f'{r:.4f}'
 values['valStarkMeV'+name]=f'{mev:.2f}'
 values['valHolevo'+name]=f'{cap:.4f}'
report={'checks':checks,'exact_checks':exact,'numerical_checks':numerical,
'negative_controls':negatives,'source_sinks':len(sinks),
'illustrative_retention':str(yield_retained),'values':values,
'scope':'Finite matrix-unit identities and bounded bit enumerations; numerical
    thermodynamic examples. Not molecular realization or general analytic proof.'}
with open('verification_report.json','w') as f: json.dump(report,f,indent=2);f.write('\
    n')
with open('verification_values_a96.tex','w') as f:
 for name,value in values.items():f.write('\\newcommand{\\'+name+'}{'+str(value)+'}\n')
print(json.dumps(report,sort_keys=True))
```