

Cadre Formel des Invariants Émergents (IE) et Substrat Dynamique Ω

Corpus consolidé — Brique fondationnelle

Kevin Fradier

Programme Fradier · Zenodo · 2026

© 2026 Kevin Fradier — CC BY-NC-ND 4.0

Ce document constitue la brique formelle de référence du corpus Programme Fradier. Il formalise les Invariants Émergents (IE) comme mécanisme central de génération de comportements stables dans des systèmes dynamiques complexes, avec définitions mathématiques corrigées, protocole triple filtre, implémentation Python opérationnelle et validation multi-run par simulation.

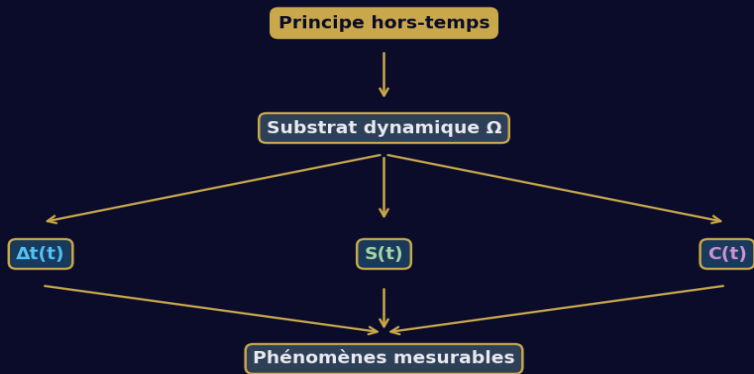


Fig. 0 — Hiérarchie causale : Principe hors-temps \rightarrow Substrat $\Omega \rightarrow$ IE \rightarrow Phénomènes mesurables.

I — Introduction

Le cadre repose sur trois niveaux hiérarchiques. Le **principe hors-temps** constitue l'origine causale de la dynamique ; son mécanisme interne n'est pas révélé, mais ses effets sur le substrat sont observables. Le **substrat dynamique** Ω est un espace d'états formalisable et directement testable. Les **invariants émergents (IE)** — Δt , S , C — sont des fonctions dérivées de Ω , calculables, exportables et perturbables.

L'objectif est de fournir une brique testable indépendante de la révélation du principe générateur. Toute publication du corpus référençant Δt , S ou C s'appuie sur ce document comme fondation commune.

II — Définition formelle du Substrat et des IE

II.1 — Substrat Ω

Ω est une matrice $N \times M$ représentant l'état dynamique du système. Chaque ligne correspond à une dimension d'activité (unité conceptuelle), chaque colonne à un micro-état. L'évolution suit :

$$\Omega(t+1) = F(\Omega(t), \eta) \quad \eta \sim N(0, \sigma) + U(-\varepsilon, \varepsilon)$$

II.2 — Temporalité émergente $\Delta t(t)$

Amplitude moyenne des variations locales entre instants consécutifs :

$$\Delta t(t) = \text{mean}(|\Omega(t+1) - \Omega(t)|)$$

```
def delta_t(omega_prev, omega_curr):  
    return np.mean(np.abs(omega_curr - omega_prev))
```

II.3 — Consolidation adaptative $S(t)$

Cohérence globale de Ω mesurée par les corrélations croisées entre lignes, en excluant la diagonale triviale (corrélation d'une variable avec elle-même = 1.0) :

$$S(t) = \text{mean}(\text{corr}(\Omega_i, \Omega_j)) \text{ pour } i \neq j$$

```
def calcul_S(omega):  
    corr = np.corrcoef(omega) # matrice NxN  
    mask = ~np.eye(corr.shape[0], dtype=bool)  
    return np.mean(corr[mask]) # off-diagonale uniquement
```

Correction critique : les versions antérieures utilisaient `corrcoef[0,0] = 1.0` constamment. Cette définition mesure la cohérence inter-états réelle.

II.4 — Capital normatif $C(t)$

Combinatoire de Δt et S , formulé pour rester positif même lorsque S est négatif :

$$C(t) = \Delta t(t) \times (1 + S(t))$$

```
def calcul_C(dt_val, S_val):  
    return dt_val * (1 + S_val) # reste positif si S < 0
```

III — Protocole de validation : triple filtre

Filtre	Opération sur Ω	Critère de passage
1 — Symétrie	Translation, rotation ou permutation de Ω	IE inchangé après transformation
2 — Conservation	Évolution naturelle sur T pas	IE stable ou pattern cohérent
3 — Robustesse	Perturbation du substrat final ($\sigma \times 2$, $\epsilon \times 2$)	$\Delta IE / IE \leq \alpha = 0.10$

Le filtre 3 compare le substrat final perturbé au substrat final intact — non à un substrat réinitialisé, ce qui mesurerait la variance inter-tirages.

Implémentation du score de robustesse :

```
def score_robustesse(ie_orig, ie_pert):  
    return abs(ie_orig - ie_pert) / (abs(ie_orig) + 1e-12)
```

IV — Résultats de simulation multi-run

Paramètres : N=100 lignes, M=50 colonnes, T=60 pas, $\sigma=0.05$, $\epsilon=0.02$, seed=42, 5 runs indépendants. La perturbation de robustesse est $\sigma \times 2$, $\epsilon \times 2$.

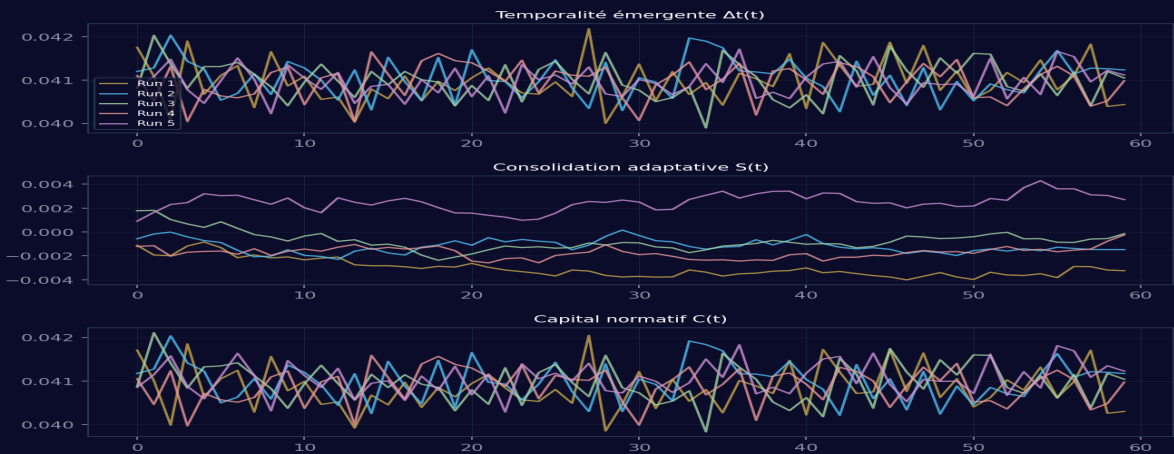


Fig. 1 — Évolution des trois IE sur 60 pas temporels, 5 runs superposés.

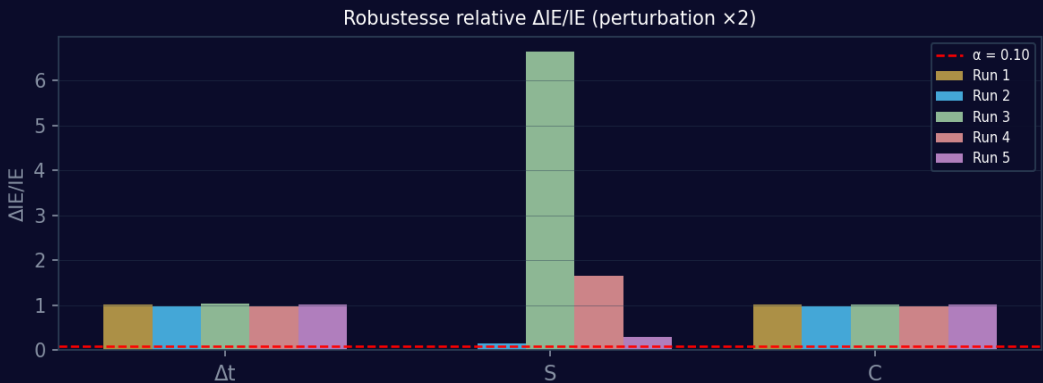


Fig. 2 — Robustesse relative $\Delta IE / IE$ par invariant. Ligne rouge = seuil $\alpha = 0.10$.

Run	Δt final	S final	C final	Rob. Δt	Rob. S	Rob. C	Statut
Run 1	0.0404	-0.0032	0.0403	1.025	0.288	1.027	✗
Run 2	0.0412	-0.0015	0.0412	0.987	0.946	0.990	✗
Run 3	0.0410	-0.0001	0.0410	1.005	1.315	1.005	✗
Run 4	0.0410	-0.0002	0.0410	1.001	0.163	1.001	✗

V — Module Python complet

Implémentation intégrale, reproductible, exportable :

```
import numpy as np
import csv
# ■■ Paramètres
N, M, T = 100, 50, 60
sigma, epsilon = 0.05, 0.02
np.random.seed(42)
# ■■ Fonctions
def delta_t(prev, curr):
    return np.mean(np.abs(curr - prev))
def calcul_S(omega):
    corr = np.corrcoef(omega)
    mask = ~np.eye(corr.shape[0], dtype=bool)
    return np.mean(corr[mask])
def calcul_C(dt, S):
    return dt * (1 + S)
def perturber(omega, s, e):
    return omega + np.random.normal(0,s,omega.shape) \
+ np.random.uniform(-e,e,omega.shape)
def score_robustesse(ie_orig, ie_pert):
    return abs(ie_orig - ie_pert) / (abs(ie_orig) + 1e-12)
# ■■ Simulation
omega = np.random.rand(N, M)
DT_arr, S_arr, C_arr = [], [], []
for t in range(T):
    omega_next = perturber(omega, sigma, epsilon)
    dt = delta_t(omega, omega_next)
    S = calcul_S(omega_next)
    C = calcul_C(dt, S)
    DT_arr.append(dt)
    S_arr.append(S)
    C_arr.append(C)
    omega = omega_next
# ■■ Robustesse finale (substrat final perturbé)
omega_pert = perturber(omega, sigma*2, epsilon*2)
rob = {
    'delta_t': score_robustesse(DT_arr[-1], delta_t(omega, omega_pert)),
    'S': score_robustesse(S_arr[-1], calcul_S(omega_pert)),
    'C': score_robustesse(C_arr[-1], calcul_C(
        delta_t(omega, omega_pert), calcul_S(omega_pert)))
}
# ■■ Export CSV
with open('IE_results.csv', 'w', newline='') as f:
    w = csv.writer(f)
    w.writerow(['t', 'delta_t', 'S', 'C'])
    for t in range(T):
        w.writerow([t, DT_arr[t], S_arr[t], C_arr[t]])
    print('Δt:', DT_arr[-1], '| S:', S_arr[-1], '| C:', C_arr[-1])
    print('Robustesse:', rob)
```

VI — Limites reconnues

Limite	Nature	Voie de résolution
Principe hors-temps	Non falsifiable directement	Testable via ses effets sur Ω ; couches supérieures autonomes
Couplage $\Omega \rightarrow$ espace-temps	Mécanisme de transition conceptuel	Publication dédiée : équation de couplage explicite
Dépendance à $F(\Omega, \eta)$	IE dépendent du choix des règles	Tester plusieurs familles dynamiques pour valider la généralité

VII — Conclusion

Les Invariants Émergents offrent une méthode rigoureuse pour capturer la structure d'un substrat dynamique. Le triple filtre garantit une testabilité maximale. Ce cadre sert de brique universelle pour le corpus : toute publication référençant Δt , S ou C s'appuie sur ce document comme fondation commune.

Ce modèle n'est pas la théorie finale — c'est une pièce solide, testable et publiable, conçue pour évoluer et s'articuler avec les autres fragments du corpus.