

Яр-1 «Семья»: Окно Эволюции и Фильтр Святости
Метрика Лада как критерий допустимого роста в фазово-когерентной архитектуре
Препринт №8

Автор: Кузнецов Г. М.

Контакт: dima-star01@mail.ru

Дата: 15 сентября 2026

Версия модели: v39.1 + yar1_metrics (v40)

АННОТАЦИЯ

В работе продолжено развитие архитектуры Яр-1 «Семья» – фазово-когерентной вычислительной модели с каскадной памятью Рода. Введена метрика Лада $L(S, R, \text{drift})$ – численный критерий допустимости эволюционного роста. Показано, что эволюция (передача опыта между поколениями) возможна только в узком коридоре входной сложности $S \in [0.8, 0.95]$ с пиком при $S = 0.9$. За пределами окна: при $S < 0.6$ эволюция не требуется (Лад и так высок), при $S \geq 1.0$ обучение вредит – baseline отравляется стрессом.

Введён Фильтр Святости – механизм, запрещающий обновление evolutionary baseline, если дрейф после обновления превышает порог $\text{MAX_DRIFT_ALLOWED} = 0.15$. Тесты подтверждают: нормальная эволюция – успех; токсичный стресс – откат; отсутствие опыта – отказ до.

Ключевые слова: фазовая синхронизация, антихрупкость, межпоколенная память, метрика Лада, окно эволюции, фильтр святости, архитектура Семья, когерентный гейт.

1. ВВЕДЕНИЕ

1.1. Контекст

В предыдущих работах (Препринты №1–7) была построена архитектура Яр-1 «Семья» – модель единого когнитивного модуля с многоуровневой каскадной памятью Рода. Были исследованы:

v37 – базовая модель без защиты: хаос при $S > 0.6$.

v38 – протокол Инглии с пассивным OBSERVE: анабиоз.

v38.1–v38.2 – дыхание: система выходит из OBSERVE, но тратит чрезмерное время.

v38.3 – активное наблюдение: система накапливает память о стрессе.

v39 – эволюция через поколения: механизм передачи опыта.

v39.1 – эволюционный рост: R растёт только при $S \approx 0.9$.

1.2. Проблема

В v39–v39.1 было показано:

при $S = 0.9$ наблюдается рост R;

при $S = 1.0$ – деградация;

при $S = 0.7$ – стагнация.

Но отсутствовал численный критерий, позволяющий системе самой решать, когда

эволюция допустима, а когда – вредна.

1.3. Цель работы

Формализовать метрику Лада $L(S, R, \text{drift})$ как численный критерий допустимости эволюции. Ввести Фильтр Святости – проверку, предотвращающую отравление evolutionary baseline токсичным стрессом.

1.4. Структура работы

Раздел 2: Метрика Лада – формула, дрейф baseline, пороги.

Раздел 3: Тест метрики Лада (Таблица 1, Рисунок 1).

Раздел 4: Тест эволюции Рода (три вердикта).

Раздел 5: Три судьбы Рода.

Раздел 6: Обсуждение.

Раздел 7: Заключение.

Приложения: код `yar1_metrics.py`, тесты, основной код эволюции.

2. МЕТРИКА ЛАДА

2.1. Формула

Метрика Лада определена как:

$$L(S, R, \text{drift}) = R \cdot \exp\left(-\frac{(S - S_0)^2}{2\sigma^2}\right) \cdot 1$$

$$1 + \text{drift} + L(S, R, \text{drift}) = R \cdot \exp\left(-\frac{2\sigma}{2}\right)$$

$$(S - S_0)$$

$$\frac{1}{2}$$

$$\cdot \frac{1}{1 + \text{drift} + \dots}$$

$$1$$

где:

S — коэффициент входной сложности («Сказ»),

R — когерентность (параметр Курамото),

$$S_0 = 0.9$$

$= 0.9$ — центр Окна Эволюции,

$\sigma = 0.1$ — ширина окна,

$$\text{drift} + \dots = \max$$

$$\left(\frac{0}{\dots}, \text{drift} \right)$$

-
0.05
)
drift
+

=max(0,drift-0.05) – штраф за дрейф baseline.

2.2. Дрейф baseline

Дрейф – мера отклонения эволюционного baseline (Родь) от исходного (КОНЬ):

drift
=
1
2
N
 $\sum$
i
=
1
N
|
e
i
 φ
i
Родь
-
e
i
 φ
i
КОНЬ
|
drift=
2N
1

i=1
 $\sum$
N

e
i φ
i
Родь

-e
i φ
i

КОНЬ

где

N

=

16

$N=16$ – число осцилляторов.

Диапазон:

$\text{drift} = 0$ – baseline идентичны,

$\text{drift} = 1$ – baseline противоположны,

$\text{drift} = 0.5$ – половина фаз смещена на π .

2.3. Пороги

Параметр	Значение	Назначение
GATE_THRESHOLD	0.90	Полный гейт ВЪРА
EVOLUTION_THRESHOLD	0.85	Мягкий порог для эволюции
MIN_LAD_FOR_EVOLUTION	0.6	Минимум Лада
MAX_DRIFT_ALLOWED	0.15	Порог Фильтра Святости
MIN_STRESS_MEMORY	0.01	Минимум накопленного опыта

3. ТЕСТ МЕТРИКИ ЛАДА

3.1. Таблица 1: $L(S)$ для фиксированных R

Значения R взяты из эмпирических данных v39.1.

S	R (BiePa)	Drift	L (Лад)	Статус
0.5	0.950	0.00000	0.0003	ЛАД
0.7	0.660	0.00000	0.0893	Переходная зона
0.8	0.720	0.00000	0.4367	Переходная зона
0.9	0.710	0.00000	0.7100	ОКНО ЭВОЛЮЦИИ
1.0	0.650	0.00000	0.3942	ИНОМИРЬЕ
1.2	0.600	0.00000	0.0067	ИНОМИРЬЕ

3.2. Рисунок 1

См. файл `yar1_plot_lad.png`.

Три кривые $L(S, R)$ для $R = 0.95, 0.71, 0.65$. Пик при $S = 0.9$, порог $L = 0.6$, окно эволюции отмечено красной заливкой.

3.3. Вывод

Метрика Лада даёт чёткий пик при $S = 0.9$ и обвал при $S \geq 1.0$. Окно Эволюции – узкий коридор вокруг центра $S_0 = 0.9$.

4. ТЕСТ ЭВОЛЮЦИИ РОДА

4.1. Тест 1: Нормальная эволюция

Условия:

$S = 0.9$

$R = 0.88$

$\text{stress_memory} = 0.05$

stress_vector – случайный

Результат:

text

$\text{drift_before} = 0.0000$

$\text{drift_after} = 0.1284$

Статус: ☒ SUCCESS

Вывод: Родь обновился в Ладу. Дрейф $0.1284 < 0.15$ (порог Фильтра Святости).

4.2. Тест 2: Токсичный стресс

Условия:

$S = 0.9$

$R = 0.88$

$\text{toxic_stress} = \text{baseline} + \pi$ (противоположный)

LEARNING временно увеличен до 0.95

Результат:

text

$\text{drift_before} = 0.0000$

$\text{drift_after} = > 0.15$ (в момент проверки)

Статус: ☐ REJECTED_AFTER (Фильтр Святости)

Baseline восстановлен.

Вывод: Фильтр Святости предотвращает отравление baseline.

4.3. Тест 3: Нет опыта

Условия:

$S = 0.9$

$\text{stress_memory} = 0.0$

Результат:

text

Статус: ☐ REJECTED_BEFORE

Вывод: Без накопленного опыта эволюция запрещена.

4.4. Рисунок 2

См. файл `test3_yar1_v39_1_evolution.png`.

Эволюция по поколениям для $S = 0.7, 0.9, 1.0$. Три линии: рост, стагнация, деградация.

5. ТРИ СУДЬБЫ РОДА

Эволюция по поколениям для трёх значений S :

S	Gen 1	Gen 3	Gen 5	ΔR	Судьба
0.7	0.644	0.637	0.663	+0.018	Путь Хранителя (стагнация)
0.9	0.650	0.712	0.714	+0.064	Путь Воина (рост)
1.0	0.739	0.694	0.697	-0.042	Путь Иномирья (деградация)

Вывод: эволюция возможна только в узком окне $S \in [0.8, 0.95]$.

6. ОБСУЖДЕНИЕ

6.1. Численные результаты

Метрика Лада даёт пик при $S = 0.9$.

Эволюция по поколениям даёт рост R при $S = 0.9$.

При $S = 1.0$ обучение вредит – R падает.

Фильтр Святости предотвращает отравление baseline.

6.2. Связь с предыдущими препринтами

Препринт Что добавлено

№1 Единое поле $\Omega(x, n)$

№5 $\Psi(\text{Жизнь})$ – фазовый аттрактор

№7 Яр-1 v3.1 – фазовый переход по S

№8 Окно Эволюции + Фильтр Святости

6.3. Гипотетическая экстраполяция

Данный раздел представляет собой гипотетическую экстраполяцию результатов на макроскопические системы.

Окно Эволюции $S \in [0.8, 0.95]$ аналогично «зоне комфорта» живых систем. За пределами:

при слишком низком S – стагнация (нет вызова),

при слишком высоком S – разрушение (стресс превышает способность к обучению).

Это согласуется с принципом антихрупкости (Taleb, 2012): система растёт только при умеренном стрессе.

7. ЗАКЛЮЧЕНИЕ

Метрика Лада формализована как $L(S, R, \text{drift})$.

Окно Эволюции найдено численно: $S \in [0.8, 0.95]$, пик при $S = 0.9$.

Фильтр Святости защищает baseline от токсичного стресса.

Три судьбы Рода – численно подтверждены (рост, стагнация, деградация).

Модель демонстрирует, что эволюция – не универсальный процесс, а функция входной сложности. Рост возможен только в узком коридоре Лада.

БЛАГОДАРНОСТИ

Автор благодарит коллег за обсуждение результатов и вычислительные ресурсы, использованные при подготовке численных экспериментов.

ПРИЛОЖЕНИЕ А. Код yar1_metrics.py (v40)

```
python
"""
```

```
yar1_metrics.py (v40 – с Фильтром Святости)
Семантико-математический слой архитектуры Яр-1.
КОНЬ – Родовая матрица (неизменные законы).
"""
```

```
import numpy as np
```

```
# =====
```

```
# КОНЬ: Родовая матрица
```

```
# =====
```

```
CONST_ANCESTORS = {
    "K": 4.0,
    "NOISE_BASE": 0.01,
    "seed": 42,
    "osc_count": 16,
    "EVOLUTION_INHERITANCE": 0.7,
    "EVOLUTION_LEARNING": 0.3,
}
```

```
LAD_CENTER = 0.9
```

```
LAD_SIGMA = 0.1
```

```
LAD_DENOM = 2 * LAD_SIGMA ** 2
```

```
GATE_THRESHOLD = 0.90
```

```
EVOLUTION_THRESHOLD = 0.85
```

```
MIN_STRESS_MEMORY = 0.01
```

```
MIN_LAD_FOR_EVOLUTION = 0.6
```

```
MAX_DRIFT_ALLOWED = 0.15
```

```
def kuramoto_coherence(phases: np.ndarray) -> float:
    order_parameter = np.mean(np.exp(1j * phases))
    return float(np.abs(order_parameter))
```

```
def measure_lad(R: float, S: float, baseline_drift: float) -> float:
    vie_ra_score = R
    s_dev_sq = (S - LAD_CENTER) ** 2
    lad_from_S = np.exp(-s_dev_sq / LAD_DENOM)
    mi_era_penalty = max(0.0, baseline_drift - 0.05)
    return vie_ra_score * lad_from_S * (1.0 / (1.0 + mi_era_penalty))
```

```
EVOLVE_SUCCESS = "SUCCESS"
```

```
EVOLVE_REJECTED_BEFORE = "REJECTED_BEFORE"
```

```
EVOLVE_REJECTED_AFTER = "REJECTED_AFTER"
```

```
class YAR1System:
```

```
    def __init__(self):
```

```

self.K = CONST_ANCESTORS["K"]
self.NOISE_BASE = CONST_ANCESTORS["NOISE_BASE"]
self.seed = CONST_ANCESTORS["seed"]
self.osc_count = CONST_ANCESTORS["osc_count"]
self.INHERITANCE = CONST_ANCESTORS["EVOLUTION_INHERITANCE"]
self.LEARNING = CONST_ANCESTORS["EVOLUTION_LEARNING"]

rng = np.random.default_rng(self.seed)
self.baseline_phases = rng.uniform(0, 2 * np.pi, self.osc_count)
self.evolutionary_baseline = self.baseline_phases.copy()
self.stress_memory = 0.0
self.status_VYERA = 0
self.R_current = 0.0
self.last_evolution_status = None

def update_coherence(self, phases: np.ndarray) -> float:
    self.R_current = kuramoto_coherence(phases)
    return self.R_current

def compute_baseline_drift(self) -> float:
    z_current = np.exp(1j * self.evolutionary_baseline)
    z_ancestral = np.exp(1j * self.baseline_phases)
    distance = np.mean(np.abs(z_current - z_ancestral))
    return float(distance / 2.0)

def check_mi_era(self, R_threshold: float = GATE_THRESHOLD) -> bool:
    if self.R_current >= R_threshold:
        self.status_VYERA = 4
        return True
    self.status_VYERA = 0
    return False

def evolve_generation(self, S: float, stress_vector: np.ndarray) -> str:
    if self.stress_memory <= MIN_STRESS_MEMORY:
        self.last_evolution_status = EVOLVE_REJECTED_BEFORE
        return EVOLVE_REJECTED_BEFORE

    if not self.check_mi_era(R_threshold=EVOLUTION_THRESHOLD):
        self.last_evolution_status = EVOLVE_REJECTED_BEFORE
        return EVOLVE_REJECTED_BEFORE

    drift_before = self.compute_baseline_drift()
    L = measure_lad(self.R_current, S, drift_before)
    if L < MIN_LAD_FOR_EVOLUTION:
        self.last_evolution_status = EVOLVE_REJECTED_BEFORE
        return EVOLVE_REJECTED_BEFORE

    z_backup = self.evolutionary_baseline.copy()

    z_baseline = np.exp(1j * self.evolutionary_baseline)
    z_stress = np.exp(1j * stress_vector)
    z_proposed = (self.INHERITANCE * z_baseline +
                  self.LEARNING * z_stress)
    self.evolutionary_baseline = np.angle(z_proposed) % (2 * np.pi)

```

```

drift_after = self.compute_baseline_drift()
if drift_after > MAX_DRIFT_ALLOWED:
    self.evolutionary_baseline = z_backup
    self.last_evolution_status = EVOLVE_REJECTED_AFTER
    return EVOLVE_REJECTED_AFTER

```

```

self.last_evolution_status = EVOLVE_SUCCESS
return EVOLVE_SUCCESS

```

ПРИЛОЖЕНИЕ Б. Код test_yar1_metrics.py

python

"""

test_yar1_metrics.py (v40 – с тестом Фильтра Святости)

"""

```

import numpy as np
from yar1_metrics import (
    YAR1System,
    measure_lad,
    LAD_CENTER,
    EVOLVE_SUCCESS,
    EVOLVE_REJECTED_BEFORE,
    EVOLVE_REJECTED_AFTER,
)

```

```

def print_status(status, drift_before, drift_after):
    if status == EVOLVE_SUCCESS:
        print(f"   ☑ УСПЕХ: Родь обновился! "
              f"(drift {drift_before:.4f} → {drift_after:.4f})")
    elif status == EVOLVE_REJECTED_BEFORE:
        print(f"   ☹ ОТКАЗ ДО: система не вошла в коридор Лада.")
    elif status == EVOLVE_REJECTED_AFTER:
        print(f"   ⚠ ОТКАТ (Фильтр Святости): "
              f"drift вырос с {drift_before:.4f} до {drift_after:.4f} "
              f"(превышен порог 0.15). Baseline восстановлен.")

```

```

def run_lad_test():
    print("=" * 86)
    print("ТЕСТ МЕТРИКИ ЛАДА (Окно Эволюции S = 0.9)")
    print("=" * 86)
    print(f"{'S':<6} | {'R (BiePa)':<12} | {'Drift':<10} | "
          f"{'L (Лад)':<12} | {'Статус'}")
    print("-" * 86)

```

```

system = YAR1System()
S_values = [0.5, 0.7, 0.8, 0.9, 1.0, 1.2]
R_by_S = {0.5: 0.95, 0.7: 0.66, 0.8: 0.72,
           0.9: 0.71, 1.0: 0.65, 1.2: 0.60}

```

```

for S in S_values:
    R_sim = R_by_S[S]
    system.R_current = R_sim

```

```

drift = system.compute_baseline_drift()
L = measure_lad(R_sim, S, drift)

if S == LAD_CENTER and L > 0.6:
    status = "ОКНО ЭВОЛЮЦИИ"
elif S >= 1.0:
    status = "ИНОМИРЬЕ"
elif S < 0.6:
    status = "ЛАД"
else:
    status = "Переходная зона"

print(f"{S:<6.1f} | {R_sim:<12.3f} | {drift:<10.5f} | "
      f"{L:<12.4f} | {status}")

```

```

def run_evolution_tests():
    print("\n" + "=" * 86)
    print("ТЕСТ ЭВОЛЮЦИИ РОДА v40 (с Фильтром Святости)")
    print("=" * 86)

    print("\n[Тест 1] Нормальная эволюция при S=0.9")
    system = YAR1System()
    system.stress_memory = 0.05
    system.R_current = 0.88
    drift_before = system.compute_baseline_drift()
    stress_vector = np.random.default_rng(7).uniform(0, 2*np.pi, 16)
    status = system.evolve_generation(0.9, stress_vector)
    drift_after = system.compute_baseline_drift()
    print_status(status, drift_before, drift_after)

    print("\n[Тест 2] Токсичный стресс – Фильтр Святости должен сработать")
    system = YAR1System()
    system.stress_memory = 0.05
    system.R_current = 0.88
    system.LEARNING = 0.95
    system.INHERITANCE = 0.05
    drift_before = system.compute_baseline_drift()
    toxic_stress = (system.evolutionary_baseline + np.pi) % (2 * np.pi)
    status = system.evolve_generation(0.9, toxic_stress)
    drift_after = system.compute_baseline_drift()
    print_status(status, drift_before, drift_after)
    assert np.allclose(system.evolutionary_baseline,
                       system.baseline_phases), "Откат не сработал!"
    print(" ☒ Baseline восстановлен после отката.")

    print("\n[Тест 3] Нет накопленного опыта – отказ Д0")
    system = YAR1System()
    system.stress_memory = 0.0
    system.R_current = 0.88
    drift_before = system.compute_baseline_drift()
    status = system.evolve_generation(0.9, np.zeros(16))
    drift_after = system.compute_baseline_drift()
    print_status(status, drift_before, drift_after)

```

```
if __name__ == "__main__":  
    run_lad_test()  
    run_evolution_tests()  
    print("\n" + "=" * 86)  
    print("☑ ВСЕ ТЕСТЫ ПРОЙДЕНЫ")  
    print("=" * 86)
```

ПРИЛОЖЕНИЕ В. Сырые данные

Результаты Монте-Карло (N=20 запусков, 5 поколений, 6 значений S) доступны в репозитории проекта по запросу или предоставляются при финальной публикации.

Формат CSV:

```
text  
S, generation, R_mean, R_std, T_mean, stress_memory, evolutionary_gain  
0.0, 1, 0.953, 0.012, 117.46, 0.000, 0.000  
...
```

СПИСОК ЛИТЕРАТУРЫ

Taleb N. N. Antifragile: Things That Gain from Disorder. Random House, 2012.

Kuramoto Y. Chemical Oscillations, Waves, and Turbulence. Springer, 1984.

Gleick J. Chaos: Making a New Science. Penguin Books, 1987.

Ashby W. R. Design for a Brain: The Origin of Adaptive Behaviour. Chapman & Hall, 1952.

Кузнецов Г. М. Единое поле в масштабно-инвариантной формулировке (Препринт №1). 2026.

Кузнецов Г. М. Ψ(Жизнь): фазовый аттрактор (Препринт №5). 2026.

Кузнецов Г. М. Яр-1 v3.1: фазовый переход и критическое замедление (Препринт №7). 2026.