

***MERP- DNA Transducer Machine: A Recursive Finite-State Numerical Representation for DNA Sequence and Mutation Analysis*****Luis O Sanchez**

M.Eng. Independent Researcher, Computational Bioinformatics, USA

Citation:: Luis O Sanchez (2026) *MERP- DNA Transducer Machine: A Recursive Finite-State Numerical Representation for DNA Sequence and Mutation Analysis*. *J. of Mod Phy & Quant Neuroscience* 2(5), 1-19. WMJ/JPQN-197

Abstract

MERP- DNA is a deterministic computational method for transforming DNA sequences into recursive numerical state trajectories. The method encodes A, C, G, and T as 0, 1, 2, and 3 and updates a bounded state using a position-dependent modular recurrence. A sequence of controlled experiments evaluates baseline trajectory generation, single-mutation propagation, mutation burden, mutation position, comparison with Hamming distance, background substitution noise, repeated observations, and formal finite-state-transducer equivalence. The experiments show that a local substitution can propagate through subsequent states and that earlier mutations produce greater trajectory divergence under clean conditions. The clean benchmark also shows that MERP-DNA produces a substantially larger and nonlinear response than Hamming distance, while both methods perfectly classify the simple synthetic mutation/non-mutation benchmark. Under simulated background noise, however, MERP-DNA distances become compressed and can be dominated by noise. Repeated averaging of pairwise MERP distances does not reliably recover the mutation signal under the tested conditions. The paper therefore presents MERP-DNA as a computational bioinformatics proof-of-concept rather than a clinical or diagnostic method. An explicit position-aware deterministic finite-state transducer reproduces the original implementation exactly, providing a formal computer-science interpretation of the machine. Future work should validate the representation using real biological genomic sequences and validated variants.

***Corresponding author:** Luis O Sanchez, M.Eng. Independent Researcher, Computational Bioinformatics, USA.

Submitted: 18.09.2026**Accepted:** 22.09.2026**Published:** 08.10.2026

Keywords: Bioinformatics, DNA Numerical Representation, Recursive Computation, Finite-State Transducer, Mutation Analysis, Numerical Trajectory, Hamming Distance, Computational Genomics

Introduction

Deoxyribonucleic Acid (DNA; ADN is the Spanish acronym) is naturally represented as a sequence over a four-symbol alphabet, but computational analysis often benefits from numerical representations that permit mathematical operations, signal processing, comparison, or feature extraction. Existing work includes fixed integer mappings, multidimensional encodings, cumulative walks, and other transformations [1]. The present study follows the general computational move from symbolic DNA to numerical data but introduces a recursive mechanism in which the numerical value at each position depends on the preceding state as well as the current nucleotide and its position.

The proposed method, MERP-DNA (Modular Evolutionary Recursive Processing for DNA), is a deterministic numerical representation that processes a DNA sequence from left to right. Each nucleotide is encoded as a small integer, the current position contributes to the state update, and the resulting value is reduced modulo a fixed integer. The resulting sequence of states is treated as a numerical trajectory. A mutation can therefore be studied as a perturbation of this trajectory rather than only as a changed character in a string.

The central idea is simple and deliberately deterministic: the state at each position is recursively determined by the preceding state, the encoded nucleotide, and the position index. This construction allows a symbolic DNA sequence to be represented as a bounded sequence of numerical states while preserving the order of the original nucleotides. The resulting trajectory provides the basis for comparing sequences and examining how local sequence changes may propagate through subsequent states.

The research question is methodological: can a deterministic recursive numerical representation expose sequence changes as propagated changes in a finite-state trajectory, and what strengths and limitations appear when the representation is challenged by mutation burden, mutation position, and background noise? The answer is evaluated through progressively more demanding computational stages.

The current experiments use constructed synthetic DNA sequences. They are valid strings over the DNA alphabet, but they are not patient-derived, tumor-derived, viral, bacterial, or population genomic samples. Accordingly, the results establish computational behavior of the method rather than clinical performance or biological causality.

The paper also formalizes MERP-DNA as a position-aware deterministic finite-state transducer (FST). This is a formal description of the same machine, not a change to the MERP-DNA recurrence. The explicit transducer implementation is tested against the original implementation to establish computational equivalence.

Background and Related Work

Numerical representation of DNA is an established field. Mendizabal-Ruiz and colleagues reviewed multiple DNA numerical representations and emphasized that different mappings can have different mathematical properties and can influence downstream similarity analysis. Their review identifies integer, real, paired, Voss, tetrahedron, Z-curve, and DNA-walk representations among commonly studied approaches.

Random-walk representations are particularly relevant to the present work [2]. Bai, Liu, and Wang described DNA primary sequences through random walks and used numerical properties of the resulting processes to compare biological sequences [2]. This establishes that a DNA string can be transformed into a trajectory and analyzed quantitatively. MERP-DNA differs by using a deterministic modular recurrence with explicit dependence on the previous state and sequence position.

Hewelt and colleagues later used DNA walks and fractal analysis to compare wild-type and mutated genes, including cancer-associated examples [3]. Their work demonstrates that trajectory-like representations can reveal structural changes associated with mutations. MERP-DNA does not attempt to infer chaos or fractal

dimension; its trajectory is a bounded computational state sequence generated by a recursive rule.

Established bioinformatics tools solve different problems. Sequence aligners such as Minimap2 are designed to map nucleotide sequences efficiently, while variant-calling pipelines estimate genomic variants from sequencing evidence [4-6]. MERP-DNA in its current form is neither an aligner nor a variant caller. It is a numerical representation and comparison mechanism. Consequently, raw MERP distances should not be compared directly with the F1 scores of clinical variant callers.

The contribution of this paper is therefore narrow and methodological: a specific recursive modular DNA representation, explicit position sensitivity, mutation-induced trajectory propagation, and formal equivalence to a position-aware deterministic finite-state transducer. A systematic literature and patent review would still be required before making a stronger novelty claim.

Table 1: Positioning of MERP-DNA

Approach	Transformation	Main emphasis	Relation to MERP-DNA
Integer mapping	Nucleotide → fixed	Numerical/signal analysis	MERP adds recursive state dependence
DNA walk	Nucleotide → directional step	Trajectory/structural properties	MERP is deterministic and modular
Z-curve	Nucleotide → cumulative coordinates	Composition and similarity	MERP uses one bounded recursive state
Voss	Nucleotide → indicator	Signal processing	MERP uses a single recursive trajectory
Alignment	Sequence → aligned	Mapping and positional matching	MERP does not align sequences
Variant calling	Reads → inferred variants	Variant detection	MERP is not a variant caller
MERP-DNA	Nucleotide + position + prior state → next state	Trajectory divergence	Core method

Hamming distance is used as the principal direct baseline because it is transparent: for equal-length sequences without background noise, it equals the number of unequal positions. The benchmark therefore asks whether MERP-DNA produces a different numerical response to the same substitutions.

The term Machine is used because the method processes an input sequence sequentially and produces output states through deterministic transitions. The machine is computational, not physical. The finite-state-transducer formulation gives this process a precise formal interpretation.

MERP-DNA Machine and Finite State Representation

The MERP-DNA Machine processes a DNA sequence sequentially, transforming each nucleotide into a numerical state according to the recursive MERP-DNA rule. Starting from the initial state $S_0=0S_0=0$, each nucleotide produces a new state that depends on the previous state, the nucleotide encoding, and the position index. Consequently, the sequence can be represented not only as a string of nucleotides but also as a numerical trajectory through a finite state space.

For example, a DNA sequence can be represented as a sequence of transitions between computational states. Each transition corresponds to one nucleotide, while the resulting state represents the numerical output of the MERP-DNA update rule. Figure 1 illustrates this representation for the sequence ACGTACGT.

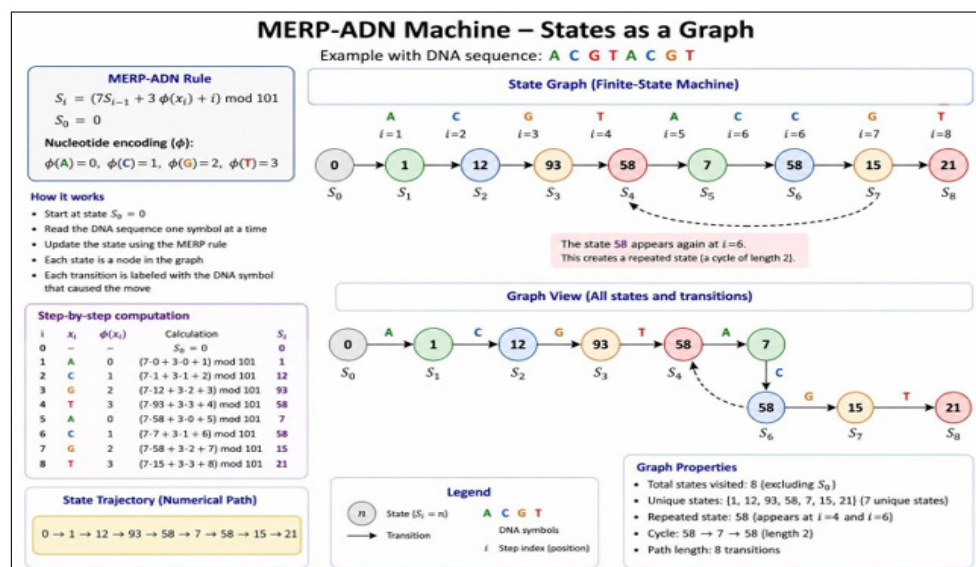


Figure 1: MERP-DNA Machine: State trajectory and finite-state representation of the DNA sequence ACGTACGT.

The example demonstrates that the MERP-DNA representation preserves the sequential structure of the input while transforming it into a numerical state trajectory. The occurrence of the state value 58 at two different positions also illustrates that the numerical trajectory may revisit previously observed state values. This behavior is a property of the finite modular state space and does not imply that the corresponding nucleotide positions are identical.

MERP-DNA Machine

MERP-DNA v1 defines a deterministic numerical representation of DNA sequences in which each nucleotide is processed sequentially. The DNA alphabet is represented by the finite set $\Sigma = \{A, C, G, T\}$, and each nucleotide is mapped to a numerical value through the encoding function ϕ . This transformation allows the symbolic DNA sequence to be processed using a mathematical recurrence while maintaining the original order of the nucleotides.

The numerical state of the machine is initialized at $S_0 = 0$. At each position i , the machine receives the nucleotide x_i and calculates a new state from three principal components: the previous state, the numerical representation of the current nucleotide, and the position of that nucleotide in the sequence. The previous state provides the recursive component of the representation, while the nucleotide encoding introduces information from the current symbol. The position term makes the transformation explicitly position-aware, meaning that the same nucleotide can contribute differently depending on where it occurs in the sequence. For MERP-DNA v1, the parameters are fixed at $a=7, b=3, c=1, d=0$, and $m=101$. Consequently, the state transition is defined by

$$S_i = (7S_{i-1} + 3\phi(x_i) + i) \bmod 101.$$

The modulo operation constrains every generated state to the finite range 00 through 100100. The resulting sequence of states,

$$S(X) = (S_1, S_2, \dots, S_n),$$

constitutes the numerical trajectory generated by the MERP-DNA Machine. This trajectory provides the basis for examining how changes in the input DNA sequence affect the resulting computational representation.

The MERP-DNA Machine: State trajectory and finite-state representation of the DNA sequence ACGTACGT. Figure 2 illustrates the nucleotide encoding, recursive state updates, resulting numerical trajectory, and position-aware finite-state representation under the MERP-DNA v1 parameters.

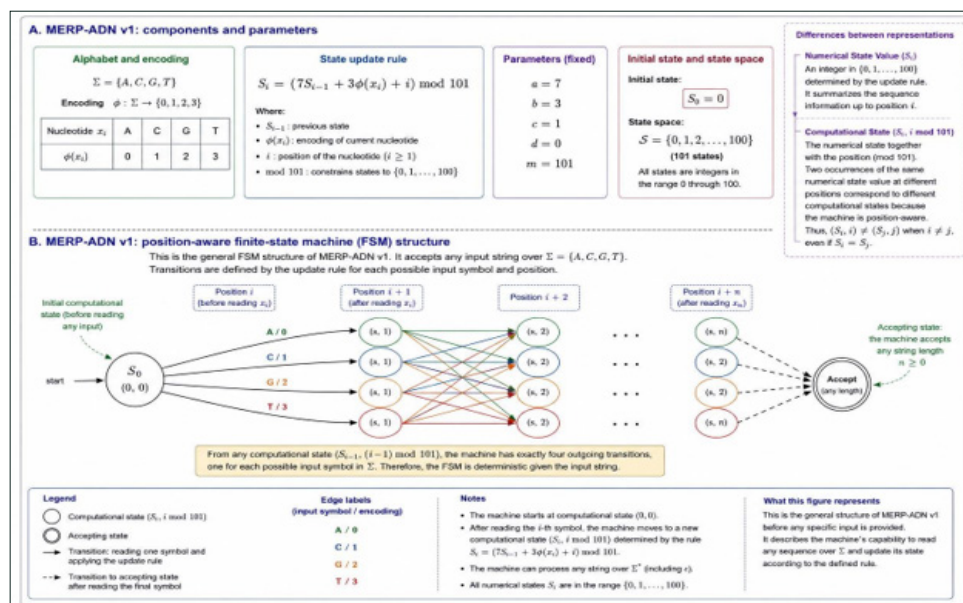


Figure 2: MERP-DNA Machine: State trajectory and finite-state representation of the DNA sequence ACGTACGT

MATLAB Code Representation:

```
a = 7;
b = 3;
c = 1;
d = 0;
m = 101;
S0 = 0;

for i = 1:length(seq)
    x = phi(seq(i));
    S(i) = mod(a*S0 + b*x + c*i + d, m);
    S0 = S(i);
end
```

Determinism and Finite-State Structure

Because the recurrence is deterministic, the same sequence, parameters, and initial state always produce the same trajectory. Because the state is reducing modulo 101, there are 101 possible numerical state values. Position dependence means that state alone does not fully determine the transition at every position. An augmented state can be represented as $(S_i, i \bmod 101)$. This produces at most $101 \times 101 = 10,201$ augmented states and motivates the finite-state interpretation used later. For ACGT, the computation is $S_1=1$, $S_2=12$, $S_3=93$, and $S_4=58$. The resulting trajectory is therefore $(1, 12, 93, 58)$. The example is deliberately hand-checkable so that the implementation can be validated before larger experiments.

MATLAB Code Representation:

```
seq = 'ACGT';

trajectory = merp_trajectory(seq, 7, 3, 1, 0, 101, 0);

disp(trajectory)
% Output: 1 12 93 58
```

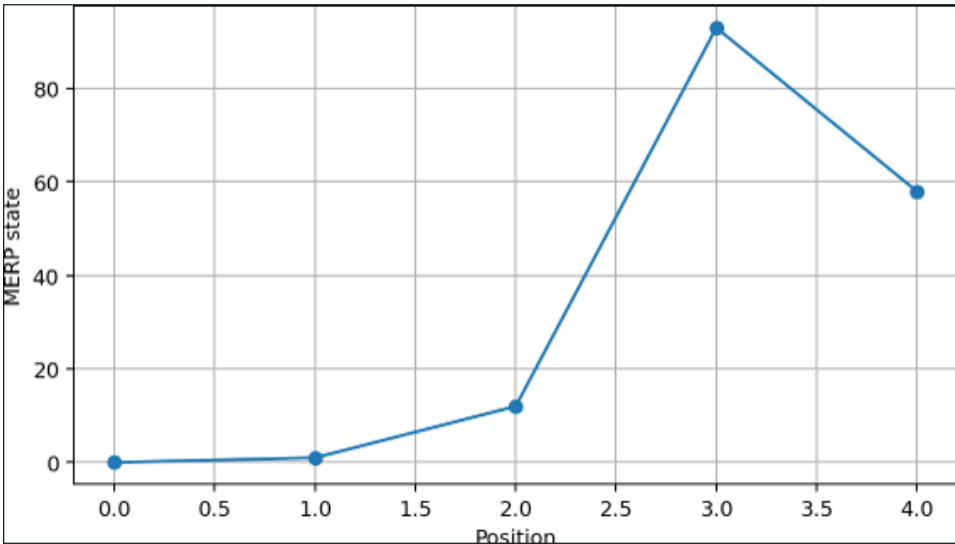


Figure 3: MERP-DNA state trajectory generated from the sequence of ACGT.

Table 2: Experimental Design

Stage	Question	Design	Output
1	Baseline correctness	ACGT	Exact trajectory
2	Mutation propagation	ACGTACGT vs ACGTTCGT	Per-position differences
3	Mutation burden	1–10 substitutions	Distance statistics
4	Mutation position	Positions 1,10,25,50,75,100;1000 trials	Distance, SD, divergent states
5	Direct benchmark	1,2,3,5,10 substitutions	MERP/Hamming comparison
6A	Noise classification	0–10% noise; threshold calibration	Accuracy/precision/recall/F1
6B	Near-neighbor robustness	Independent noise in both sequences	Mean/SD distances
6C	Position × noise	Position and noise factorial design	Distance/divergence
7	Repeated observations	1–20 repeats; 0–10% noise	F1/separation
8	Formal equivalence	Original vs explicit FST	Exact agreement

The statistical stages use randomized trials under controlled synthetic conditions. The trials establish algorithmic behavior, not biological population estimates. Ground truth is known because the mutation burden and mutation positions are deliberately planted.

The same MERP-DNA v1 parameterization is retained throughout. The experiments are therefore intended to characterize one defined machine rather than optimize parameters from one stage to the next.

Stage 1: Baseline Trajectory

Stage 1 establishes the basic operation using ACGT. The expected trajectory is (1,12,93,58), and the implementation reproduces it exactly. This stage is an implementation check rather than a statistical experiment. Its purpose is to establish that the encoding, recurrence, initial state, and modulus are correctly represented before mutation and noise are introduced. The trajectory itself has no direct biological interpretation. It is the numerical state representation produced by the machine. Later comparisons operate on these states.

MATLAB Code Representation:

```
seq = 'ACGT';

S = merp_trajectory(seq, 7, 3, 1, 0, 101, 0);

assert(isequal(S, [1 12 93 58]));
```

Stage 2: Single-Mutation Propagation

Position	Original	Mutated	Difference
1	1	1	0
2	12	12	0
3	93	93	0
4	58	58	0
5	7	16	9
6	58	20	38
7	15	52	37
8	21	78	57

The sequences of ACGTACGT and ACGTTCGT differ only at position 5. The first four states are identical, while the mutation changes the state at position 5 and continues to affect positions 6 through 8. The total absolute trajectory difference is 141. The important result is not the numerical value itself, but the propagation pattern: a local symbolic change becomes a multi-position perturbation because the next state depends on the previous state. This behavior is the central conceptual property of MERP-DNA. It distinguishes the method from a direct character-counting metric, although it also creates sensitivity to noise.

MATLAB Code Representation:

```
S1 = merp('ACGTACGT');
S2 = merp('ACGTTCGT');

difference = abs(S1 - S2);

% Output: 0 0 0 0 9 38 37 57
```

Figure 4: Shows the propagation of a single substitution.

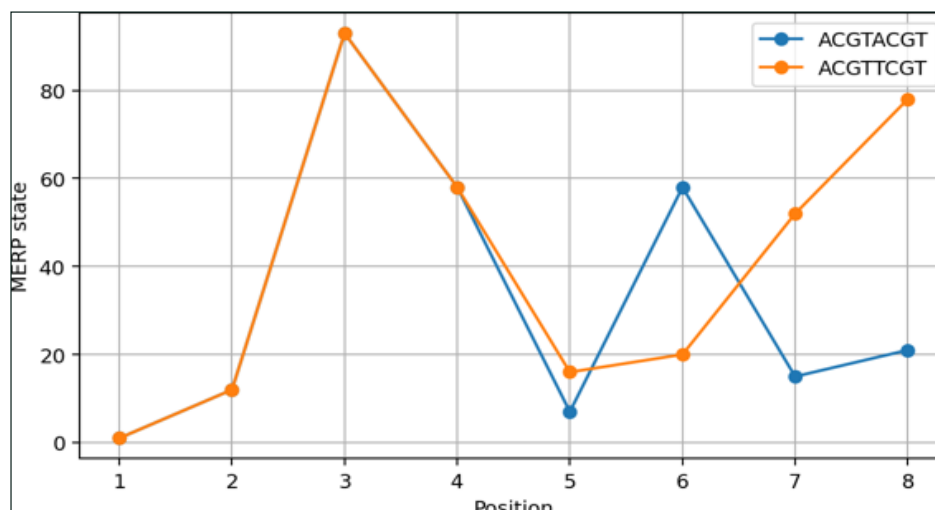


Figure 4: Original and mutated MERP-DNA trajectories; divergence begins at the mutation and propagates forward.

Stage 3: Table 3: Mutation Burden

Mutations	Total distance	Mean state difference	Maximum
1	168	8.4	64
2	806	40.3	79
3	468	23.4	66
5	435	21.75	65
10	734	36.7	88

Stage 3 evaluates mutation counts of 1, 2, 3, 5, and 10. The observed total distances are 168, 806, 468, 435, and 734. The response is non-monotonic in this experiment.

This result shows that MERP trajectory distance is not a direct mutation counter. The modular recursive system can produce different levels of divergence for the same number of substitutions depending on sequence context and mutation arrangement.

The non-monotonicity is scientifically useful because it prevents an overly simple interpretation. MERP-DNA should not be described as estimating mutation count directly from total trajectory distance.

MATLAB Code Representation:

```
for k = [1 2 3 5 10]

    S_mutated = merp(mutated_sequence);

    total_distance = sum(abs(S_reference - S_mutated));

end
```

Figure 5 shows the MERP-DNA Distance versus Mutation Count.

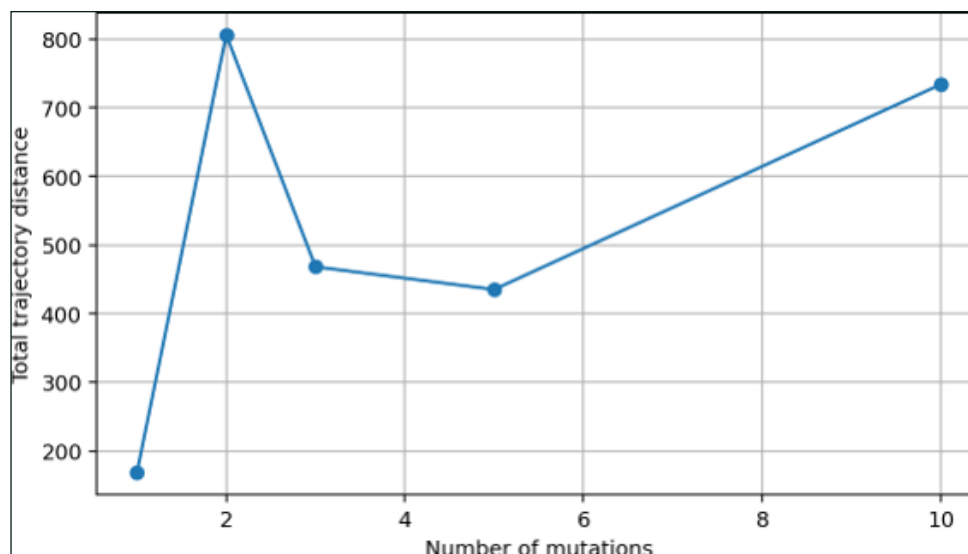


Figure 5: Total MERP-DNA trajectory distance versus mutation count in the Stage 3 experiment.

Stage 4: Table 4: Mutation Position

Position	Mean distance	SD	Divergent states
1	3397.1	189.27	100
10	3121.2	173.19	91
25	2620.9	166.4	76
50	1711.4	128.98	51
75	923.55	97.515	26
100	8.876	18.318	1

Stage 4 uses length-100 sequences and 1000 trials for each mutation position. The mean distance declines strongly as the mutation is moved toward the end of the sequence.

A mutation at position 1 has the greatest opportunity to influence later states, while a mutation at position 100 can affect only the final state. The mean divergent-state counts follow the same structural logic: 100, 91, 76, 51, 26, and 1.

The positional result demonstrates that MERP-DNA encodes location through recursive propagation. This is a distinct property of the representation, but it is not evidence that earlier biological mutations are intrinsically more important.

MATLAB Code Representation:

```
for p = mutation_positions

    mutated = reference;
    mutated(p) = alternate_base;

    D(p) = sum(abs(merp(reference) - merp(mutated)));

end
```

Figure 6 shows the positional sensitivity of MERP-DNA.

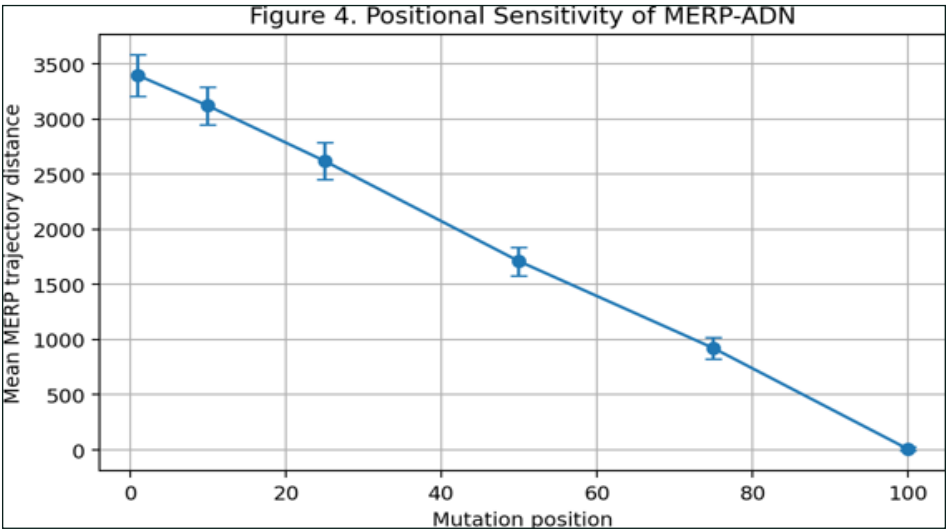


Figure 6: Mean MERP-DNA trajectory distance by mutation position; error bars show one empirical standard deviation.

Stage 5: Table 5: Direct Benchmark

Mutations	MERP mean distance	Hamming mean distance	MERP F1	Hamming F1
1	1808.8	1	1	1
2	2329.7	2	1	1
3	2559.9	3	1	1
5	2831.3	5	1	1
10	3087.4	10	1	1

Under clean synthetic conditions, hamming distance equals the substitution count exactly. MERP-DNA generates much larger trajectory distances and a nonlinear response. Both methods achieved perfect accuracy, precision, recall, and F1 on clean mutation/non-mutation classification. This is a sanity check, not evidence of superiority. The task is simple and the ground truth is perfectly controlled. The MERP response increases with diminishing increments at higher mutation counts. The current bounded modular representation may therefore exhibit a saturation-like behavior, although this requires broader parameter and sequence studies before it can be generalized.

```
hamming = sum(reference ~= mutated);  
  
merp_distance = sum(abs(merp(reference) - merp(mutated)));  
  
% Classify using a threshold calibrated from negative examples
```

Figure 7 shows the Direct Benchmark: MERP-DNA versus Hamming Distance

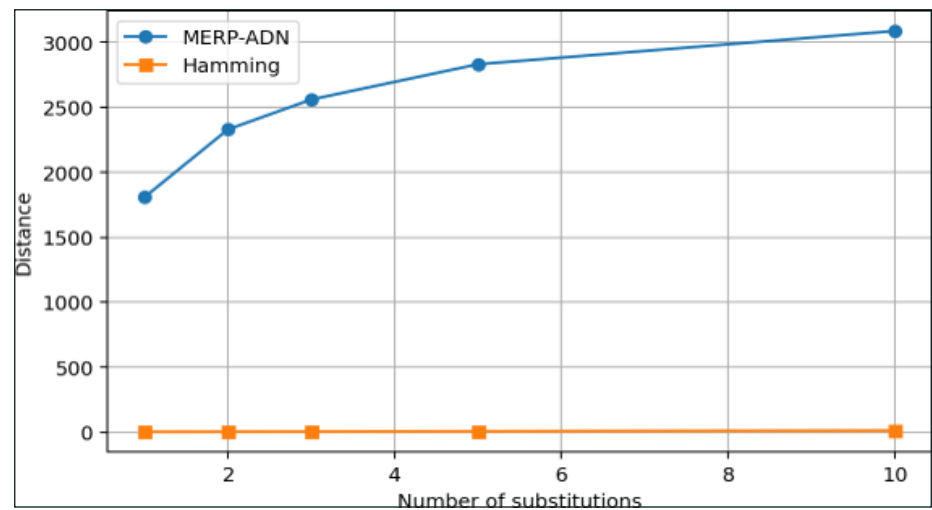


Figure 7: Clean synthetic benchmark comparing MERP-DNA trajectory distance with Hamming distance.

Stage 6A: Table 6: Noise and Near-Neighbor Classification

Noise	MERP threshold	Hamming threshold	MERP F1	Hamming F1
0%	0	0	1	1
1%	3570	4	0.01965	0.03137
2%	3551	6	0.02326	0.01972
5%	3778	10	0.01191	0.03488
10%	3704	17	0.012	0.02362

Stage 6A introduces background substitution noise independently into the compared observations. The positive class contains one planted mutation at position 50; the negative class contains no planted mutation. Thresholds are calibrated as the 99th percentile of negative calibration distances.

At every nonzero noise level, both methods show near-zero recall and F1 in this design. Accuracy remains close to 0.50 because the classifier largely predicts the negative class.

This is a robustness-boundary result. It demonstrates that a large clean mutation response does not automatically translate into reliable mutation detection when the background noise also produces substantial divergence.

MATLAB Code Representation:

```
threshold = prctile(negative_calibration_distances, 99);

prediction = test_distance > threshold;

% Evaluate accuracy, precision, recall, and F1
```

Stage 6B: Table 7: Near-Neighbor Robustness

Mutations	0%	1%	2%	5%	10%
1	1747.1	2434.8	2755.3	3065.5	3212.6
2	2289.9	2678.6	2879.1	3107.1	3240.7
3	2584.4	2814.1	2897	3116.6	3218.6
5	2847.8	2973.4	3059.2	3145.8	3239.7
10	3081.2	3120.1	3152.9	3222.3	3265.3

Stage 6B examines the raw distance distributions when both observations are independently corrupted by background substitution noise. At zero noise, MERP distances separate mutation burdens strongly. As noise increases, the means move toward a common region around the upper part of the trajectory-distance scale.

At 10% noise, the MERP means 1, 2, 3, 5, and 10 mutations are 3212.6, 3240.7, 3218.6, 3239.7, and 3265.3. The ordering is no longer monotonic.

Hamming maintains a more transparent relationship with mutation burden, although its absolute distance also becomes increasingly dominated by background discrepancies. The two metrics therefore exhibit different robustness and interpretability profiles.

MATLAB Code Representation:

```
for noise = levels

    noisyA = add_substitution_noise(A, noise);
    noisyB = add_substitution_noise(B, noise);

    dMERP = sum(abs(merp(noisyA) - merp(noisyB)));
    dHam = sum(noisyA ~= noisyB);

end
```

Figure 8 shows the Near-Neighbor MERP Distance under Noise

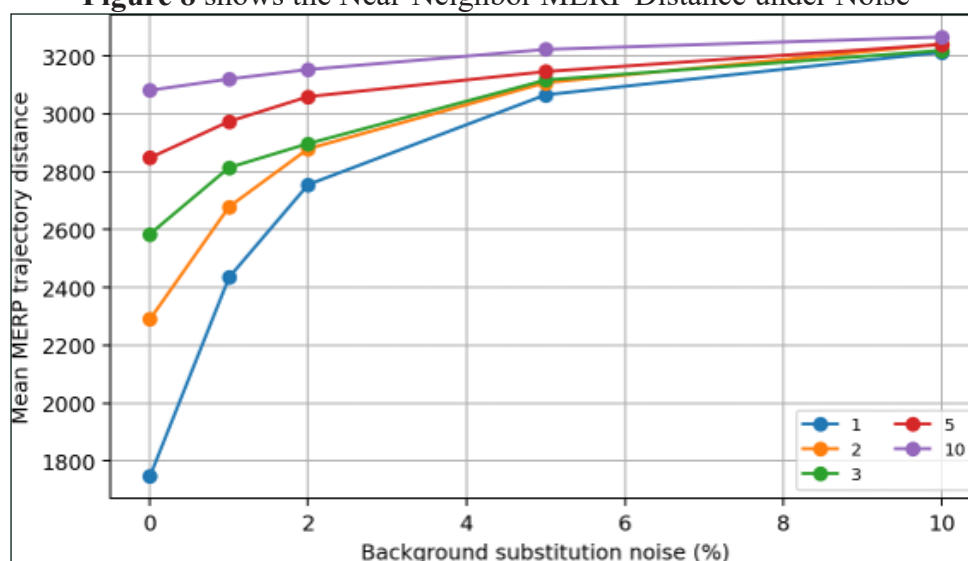


Figure 8: Mean MERP-DNA distance for near-neighbor sequences as background substitution noise increases.

Stage 6C: Table 8: Mutation Position under Noise

Position	0%	1%	2%	5%	10%
1	3381.5	3376.8	3356.2	3341	3334.9
10	3117.2	3109.3	3121.2	3173.8	3230.2
25	2625.3	2738.3	2862	3068.8	3213.5
50	1711.9	2311.4	2687.2	3054.8	3205.8
75	922.42	2117.1	2620.7	3035.9	3215.8
100	10.22	1937.6	2556.9	3061.4	3196.1

Stage 6C combines mutation position and background noise. At 0% noise, the positional ordering is strong: $1 > 10 > 25 > 50 > 75 > 100$. At 1–2% noise much of the structure remains visible, while at 5–10% the distances converge.

The divergent-state measure shows the same effect. Under 0% noise, positions 1 and 100 produce 100 and 1 divergent states, respectively. Under 10% noise, all positions produce approximately 94–99 divergent states. Thus, background noise does not merely reduce the planted signal. It creates a recursive trajectory response of its own, causing the system to appear globally divergent and reducing positional contrast.

MATLAB Code Representation:

```
for p = mutation_positions
    for noise = noise_levels
        D(p, noise) = sum(abs(merp(noisy_ref) - merp(noisy_mut)));
    end
end
```

Figure 9 shows the mutation position response under background noise.

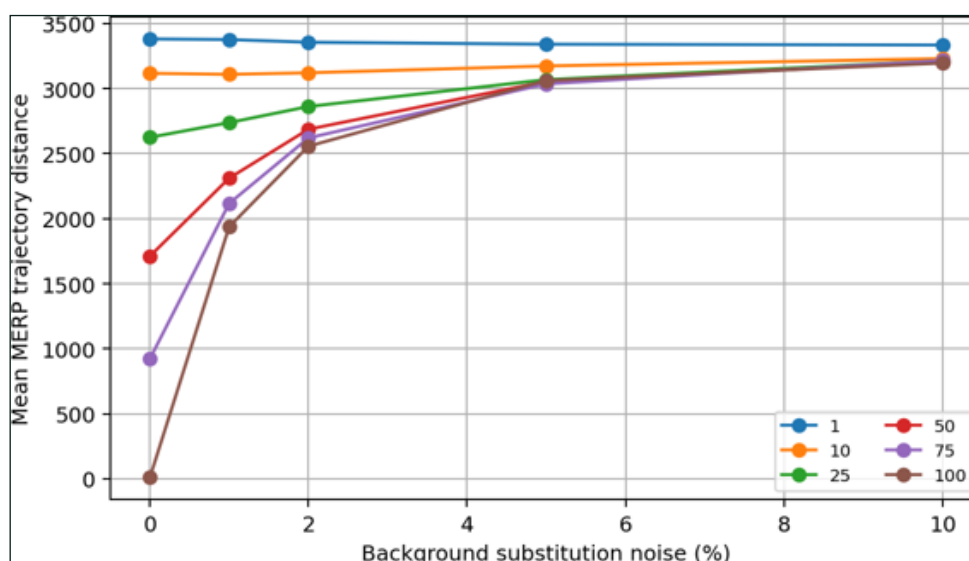


Figure 9: MERP-DNA mutation-position response as background substitution noise increases.

Stage 7: Table 9: Repeated Observations

Noise	1	3	5	10	20
0	1	1	1	1	1
1	0.0079	0.0428	0.0236	0.0943	0.1264
2	0.004	0.0195	0.0236	0.0309	0.0349
5	0.0079	0.0197	0.0456	0.0233	0.004
10	0.0158	0.004	0.0197	0.0349	0.0421

Stage 7 asks whether repeated noisy observations can recover a single mutation signal. The experiment uses one mutation at position 50, repeated observations of 1, 3, 5, 10, and 20, and noise levels of 0%, 1%, 2%, 5%, and 10%. The aggregation averages pairwise distances; it does not average nucleotide calls or state trajectories

before computing distance. MERP F1 remains very low at nonzero noise. At 1% noise it rises from 0.0079 at one observation to 0.1264 at twenty observations, but this remains poor classification. At 2%, 5%, and 10% the improvement is inconsistent. The result is a limitation of this particular aggregation strategy, not proof that no repeated-observation method could ever help MERP-DNA. Alternative aggregation schemes should be evaluated separately.

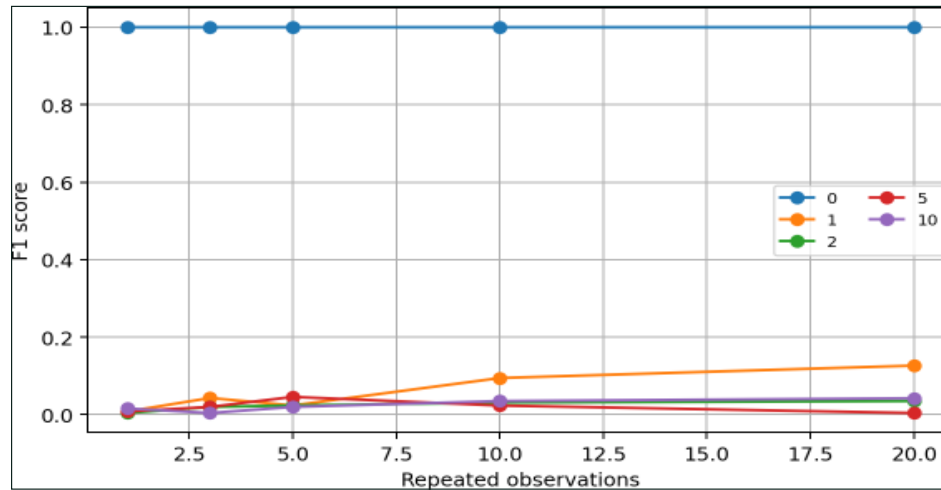


Figure 10: MERP-DNA F1 score versus repeated observations across simulated noise levels.

Table 10: Hamming Comparison and Signal Separation

Noise	Hamming F1 @1	@3	@5	@10	@20
0	1	1	1	1	1
1	0.0275	0.1673	0.2163	0.4773	0.8661
2	0.0237	0.0875	0.1165	0.3776	0.5718
5	0.0119	0.0728	0.0467	0.172	0.1025
10	0.0236	0.0197	0.0617	0.1491	0.1986

Hamming benefits more strongly from repeated observations at low noise. At 1% noise, Hamming F1 reaches 0.8661 with 20 observations; at 2% it reaches 0.5718. This result demonstrates that the repeated-observation design favors the direct symbol-count baseline more strongly than the current MERP aggregation. MERP positive-minus-negative separation remains positive and substantial at 1% noise, but it shrinks markedly at 2% and can approach or cross zero at 5–10%. This shows that the mutation signal becomes entangled with the noise-induced trajectory response. The repeated-observation stage is therefore an important negative result. It prevents the paper from claiming that simple replication automatically solves MERP-DNA noise sensitivity.

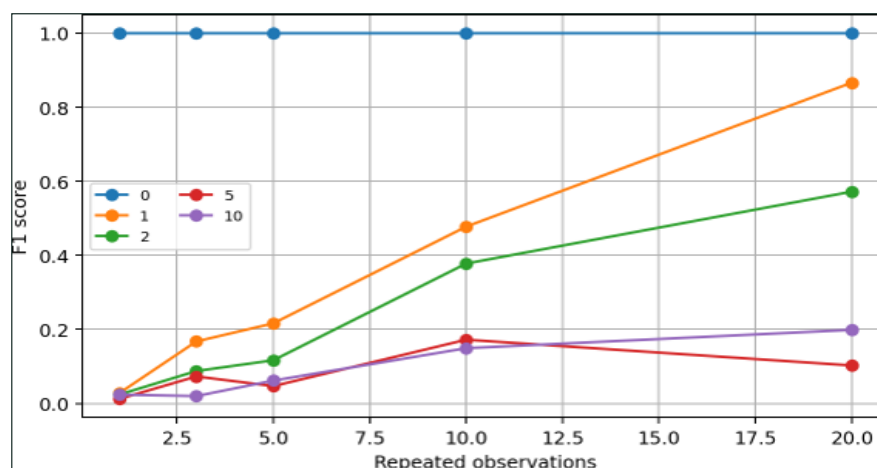


Figure 11: Hamming F1 score versus repeated observations across simulated noise levels.

Figure 12 Shows the MERP Signal Separation under Repeated Observation

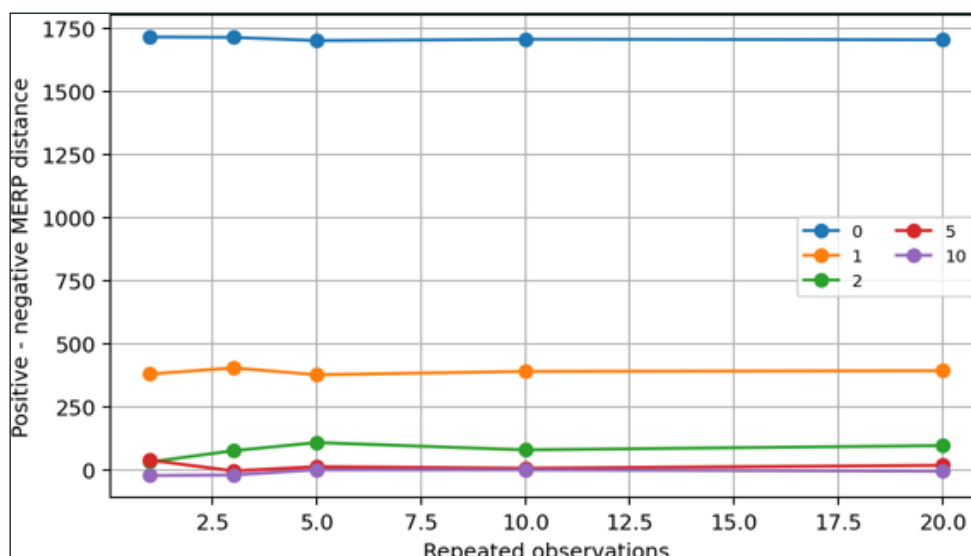


Figure 12: MERP-DNA signal separation, defined as mean positive distance minus mean negative distance.

Stage 8: Finite-State Transducer Equivalence

Stage 8 formalizes MERP-DNA as a position-aware deterministic finite-state transducer (FST). The FST consumes one nucleotide at a time, encodes the symbol, computes the next state with the same recurrence, and emits the newly computed state. The formalization is an exact computational description of the existing machine.

$$\delta(S_{i-1}, x_i, i) = (7S_{i-1} + 3\phi(x_i) + i) \bmod 101$$

MATLAB Code Representation:

```
function trajectory = merp_fst(seq, a, b, c, d, m, S0, phi)

    trajectory = zeros(1, length(seq));
    current_state = S0;

    for i = 1:length(seq)

        encoded_symbol = phi(seq(i));

        next_state = mod( ...
            a * current_state + ...
            b * encoded_symbol + ...
            c * i + d, m);

        trajectory(i) = next_state;
        current_state = next_state;

    end

end
```

For ACGT, both implementations return (1,12,93,58). For ACGTACGT and ACGTTCGT, the explicit FST reproduces the same trajectories and the same difference vector (0,0,0,0,9,38,37,57).

Table 11: Equivalence Tests

Sequence	Result
A	PASS
C	PASS
G	PASS
T	PASS
ACGT	PASS
AAAA	PASS
CCCC	PASS
GGGG	PASS
TTTT	PASS
ACGTACGT	PASS
AGCTAGCT	PASS
TGCATGCA	PASS

All listed equivalence tests passed. The explicit FST formulation therefore reproduces MERP-DNA v1 exactly for the tested sequences.

The transition table for ACGT is 0→1 on A, 1→12 on C, 12→93 on G, and 93→58 on T. Each transition emits the newly computed state. This provides a compact machine-level description of the same recurrence.

The FST perspective does not change MERP-DNA. It gives the method a formal representation within automata theory and clarifies why position must be represented as part of the augmented machine state.

Table 12: Position Sequences

Position	Input	Previous	New
1	A	0	1
2	C	1	12
3	G	12	93
4	T	93	58

Discussion

Stages 1–8 define a coherent computational profile. MERP-DNA is deterministic, bounded, recursive, position-aware, and formally expressible as a finite-state transducer. A local substitution changes the trajectory at the mutation position and can propagate through later states. Mutation-position sensitivity is one of the clearest findings. Earlier mutations have more subsequent transitions through which their perturbation can propagate. This property could be useful for sequence analyses where the location of a change matters, but it should not be interpreted as a measure of biological importance.

The comparison with Hamming clarifies the trade-off. Hamming is simple and interpretable. MERP provides a richer trajectory response and positional sensitivity, but its raw distance is less directly interpretable and more vulnerable to recursive accumulation of noise.

Noise experiments are the main limitations. Independent background substitutions can generate large MERP trajectory differences even without a planted mutation. Therefore, the present raw distance should not be treated as a robust mutation detector in noisy sequences.

Repeated averaging of pairwise MERP distances does not solve the problem. The next generation of experiments

should investigate whether consensus sequences, state-wise robust summaries, or other aggregation strategies can improve robustness, while keeping such methods clearly separate from the current v1 results.

Scientific Contribution and Appropriate Claim

The most defensible contribution of the present work is methodological. MERP-DNA provides a specific recursive numerical representation in which nucleotide substitutions can become propagated perturbations of a finite-state trajectory. At each position, the machine effectively asks three questions: What DNA letter am I reading? Where am I in the sequence? And what state did I have immediately before? The answers determine the next numerical state through the MERP-DNA recurrence. In this way, the machine combines the current nucleotide, its position, and the preceding computational state to generate an ordered numerical representation of the DNA sequence.

The recursive structure is important because the state generated at one position becomes part of the calculation at the next position. Consequently, a change to a nucleotide does not necessarily remain confined to a single numerical state; it can propagate through subsequent states of the trajectory. The resulting trajectory therefore provides a computational representation in which local sequence changes can produce measurable changes in the broader numerical path.

The FST equivalence experiment strengthens this formal description by demonstrating that the MERP-DNA Machine can be represented exactly as a deterministic, position-aware finite-state transducer while preserving the trajectories produced by the original recurrence. This establishes a formal connection between the numerical formulation and a finite-state computational model without changing the underlying MERP-DNA algorithm.

The paper does not claim that MERP-DNA replaces DNA sequencing, sequence alignment, variant calling, or established clinical genomics workflows. These technologies and computational methods address different problems and require different forms of validation. The contribution presented here is instead the definition and experimental characterization of a recursive numerical representation of DNA sequences.

The novelty claim is therefore intentionally narrow. Numerical DNA representations and DNA-walk approaches are established areas of research. The potential novelty of MERP-DNA lies in the particular combination of recursive modular state updating, explicit position dependence, trajectory divergence, and formal FST equivalence. These elements define the computational behavior investigated in this work without requiring a broader claim that numerical DNA representation itself is new.

The next major validation step is the application of MERP-DNA to real biological data. Public genomic sequences containing validated variants can be used to determine whether the computational behavior observed in the controlled experiments persists in biologically structured sequences and whether the resulting trajectory information provides useful information for a clearly defined bioinformatics task.

Limitations

Synthetic sequences are the first limitation. They provide exact control over mutations and noise but do not reproduce the full complexity of biological genomes or sequencing platforms.

The current distance definition is another limitation. The sum of absolute state differences is transparent but not necessarily optimal. Alternative trajectory descriptors may have different sensitivity and robustness.

The noise model is simplified. Background errors are represented as substitutions applied independently. Real sequencing errors can be context-dependent and can include insertions and deletions.

The results are parameter specific. The reported experiments use $a=7$, $b=3$, $c=1$, $d=0$, $m=101$, and $S_0=0$. Other parameterizations may behave differently. The current paper intentionally freezes v_1 rather than optimizing it.

Finally, no biological outcome validation is presented. A large trajectory difference does not imply pathogenicity, functional impact, clinical relevance, or evolutionary significance. Such claims require independent biological evidence.

Conclusions and Future Work

The MERP-DNA Transducer Machine provides a deterministic recursive numerical representation of DNA sequences. Rather than treating a DNA sequence only as a string of nucleotide symbols, the machine processes each nucleotide sequentially and transforms the sequence into a bounded numerical trajectory. At each position, the computational process incorporates the current nucleotide, its position within the sequence, and the state generated at the preceding position. This recursive mechanism allows a local change in the input sequence to produce a propagated change in subsequent numerical states [7-12].

The experimental results provide evidence for several identifiable properties of the MERP-DNA representation. Under controlled synthetic conditions, nucleotide substitutions produced measurable changes in the resulting trajectories, and mutations introduced at earlier positions generally produced larger trajectory divergence because they had more opportunities to propagate through subsequent states. Increasing mutation burden also produced a nonlinear trajectory-distance response. These observations demonstrate that MERP-DNA does not simply count nucleotide differences; it transforms sequence differences into changes in a computational state trajectory.

The comparison with Hamming distance provides an important reference point. Hamming distance directly measures the number of positions at which two sequences differ, whereas MERP-DNA measures the accumulated difference between their generated numerical trajectories. In the controlled clean benchmark, both approaches achieved perfect classification of the simple synthetic mutation/non-mutation task. Therefore, the present results do not establish that MERP-DNA is superior to Hamming distance. Instead, they demonstrate that the two representations measure sequence differences in fundamentally different ways.

The noise experiments further establish an important limitation of the current formulation. Background substitution noise can generate substantial MERP-DNA trajectory divergence, causing the distances associated with different mutation conditions or positions to become increasingly compressed as noise increases. Repeated observations using the tested aggregation strategy did not reliably recover the mutation signal under higher noise levels. These findings are scientifically relevant because they define a current robustness boundary of the method rather than being treated as unsuccessful experiments.

The formal finite-state transducer formulation provides a complementary theoretical interpretation of the machine. The equivalence experiments demonstrated that the explicit position-aware deterministic FST reproduces the trajectories generated by the original MERP-DNA recurrence. Thus, the FST formulation does not introduce a different algorithm; it provides a formal computational description of the same recursive numerical process.

The overall contribution of this work is therefore methodological and computational. MERP-DNA should be understood as a proposed representation of DNA sequences, rather than as a replacement for sequencing technologies, sequence-alignment methods, variant callers, or established clinical genomics workflows. Its potential value lies in providing a different numerical representation in which nucleotide changes can be examined as perturbations of a recursive, position-aware trajectory.

The most important next step is validation using real biological sequences and experimentally or publicly validated variants. Such data will allow future studies to determine whether the trajectory behaviors observed

in controlled synthetic sequences persist in biologically structured genomic contexts. Further investigation should also examine realistic sequencing-error models, broader sequence contexts, insertions and deletions, alternative biological datasets, and clearly defined bioinformatics tasks in which the MERP-DNA representation can be evaluated using task-appropriate performance measures.

Ultimately, the present work establishes the computational behavior and formal structure of the MERP-DNA Transducer Machine. It demonstrates that DNA can be processed through a recursive, position-aware finite-state mechanism to generate a bounded numerical trajectory, and that sequence changes can produce measurable trajectory perturbations. Whether these properties provide practical biological or bioinformatics, value remains an empirical question for future validation on real genomic data.

References

1. Mendizabal Ruiz G, Román Godínez I, Torres Ramos S, Salido Ruiz RA, et al. (2017) On DNA numerical representations for genomic similarity computation. *PLOS ONE* 12: e0173288.
2. Bai FL, Liu YZ, Wang TM (2007) A representation of DNA primary sequences by random walk. *Mathematical Biosciences* 209: 282-291.
3. Hewelt B, Li H, Jolly MK, Kulkarni P, Mambetsariev I, et al. (2019) The DNA walk and its demonstration of deterministic chaos—relevance to genomic alterations in lung cancer. *Bioinformatics* 35: 2738-2748.
4. Li H (2018) Minimap2: Pairwise alignment for nucleotide sequences. *Bioinformatics* 34: 3094-3100.
5. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, et al. (2010) The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research* 20: 1297-1303.
6. Poplin R, Chang PC, Alexander D, Schwartz S, Colthurst T, et al. (2018) A universal SNP and small-indel variant caller using deep neural networks. *Nature Biotechnology* 36: 983–987.
7. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. *Nature Methods* 9: 357-359.
8. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, et al. (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 25: 2078-2079.
9. Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48: 443-453.
10. Durbin R, Eddy SR, Krogh A, Mitchison G (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.
11. Waterman MS (1995) *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman & Hall.
12. McCarthy DJ, Humburg P, Kanapin A, Rivas MA, Gaulton K, et al. (2014) Choice of assembly and variant calling pipeline influences human genome variant calls. *Genome Biology* 15: 445