Testul Spectral (Transformata Fourier Discretă) din suita NIST SP 800-22

Implementare, validare și analiză critică

WAGNER Ștefan-Daniel

OLTEAN Dan-Gabriel MATVEEV Victor-Nicolae PUFLEA Steluța Lavinia

Facultatea de Științe Aplicate, Teoria Codării și Stocării Informației

Universitatea Națională de Știință și Tehnologie POLITEHNICA București (UNSTPB)

Coordonator: conf. univ. dr. Emil Simion, `emil.simion@upb.ro`

Co-coordonator: drd. Duță Cătălin Marius

24 mai 2026

Rezumat

Lucrarea prezintă testul statistic al Transformatei Fourier Discrete (testul spectral), secțiunea 2.6 din standardul NIST SP 800-22, folosit pentru a detecta componente periodice într-o secvență de biți generată de un RNG/PRNG. Pornind de la o implementare inițială monolitică, am reproiectat testul într-o bibliotecă C++ orientată pe obiecte, capabilă să proceseze secvențe de ordinul milioanelor de biți printr-o transformată rapidă (radix-2 plus Bluestein). Am validat implementarea față de *codul de referință* NIST (nu doar față de documentație) și am rulat-o pe toate fișierele de date pe care NIST și-a testat suita (e, π, $\sqrt{2}$, $\sqrt{3}$, generator bazat pe SHA-1). În final discutăm controversele bine cunoscute ale testului: neconcordanța dintre exemplul din documentație ($N_1 = 46$) și propriul cod de referință ($N_1 = 48$), precum și distribuția de referință contestată în literatură (varianța nu a fost dedusă analitic, ci estimată numeric), pe care simulările noastre Monte-Carlo o confirmă empiric.

Cuprins

Capitolul 1
Introducere și motivație

1.1 Contextul: aleatorismul în criptografie

Securitatea aproape oricărui sistem criptografic depinde de calitatea generatoarelor de numere (pseudo)aleatoare. Cheile secrete, vectorii de inițializare, nonce-urile, sarea (salt) și parametrii efemeri din protocoalele de schimb de chei trebuie să fie impredictibili: dacă un atacator poate ghici sau reconstrui ieșirea generatorului, întreaga construcție criptografică se prăbușește, indiferent cât de robust este cifrul folosit deasupra.

Din acest motiv, generatoarele de numere aleatoare (RNG) și pseudoaleatoare (PRNG) sunt supuse unor baterii de teste statistice. Aceste teste nu pot *demonstra* că o secvență este aleatoare (proprietate imposibil de certificat dintr-un eșantion finit), dar pot respinge ipoteza de aleatorism atunci când secvența prezintă regularități statistice măsurabile. Suita NIST SP 800-22 [1] este, de peste două decenii, referința standard în acest domeniu: ea reunește 15 teste statistice, fiecare sensibil la un alt tip de abatere de la aleatorism.

1.2 Ce măsoară testul spectral

Testul Transformatei Fourier Discrete (DFT), numit și testul spectral, este testul din secțiunea 2.6 a suitei. Spre deosebire de majoritatea celorlalte teste, care lucrează în domeniul timpului (numără apariții de tipare, blocuri, treceri prin zero etc.), testul spectral mută secvența în *domeniul frecvenței*. Principiul este simplu:

- o secvență cu adevărat aleatoare are un spectru de putere aproximativ plat (zgomot alb): nicio frecvență nu domină;
- o secvență cu o componentă periodică (un tipar care se repetă la intervale regulate) concentrează energie la anumite frecvențe, ceea ce produce *vârfuri* înalte în spectru.

Testul măsoară câte vârfuri spectrale depășesc un prag corespunzător nivelului de 95% și compară acest număr cu valoarea așteptată sub ipoteza de aleatorism. Periodicitățile sunt relevante criptografic deoarece introduc predictibilitate: un PRNG cu o perioadă scurtă sau cu corelații periodice ascunse este vulnerabil.

1.3 De ce acest test, în mod special

Am ales testul spectral din două motive.

Este interesant conceptual. Leagă prelucrarea semnalelor (DFT, FFT) de criptografie și dă o imagine geometrică a „lipsei de structură”.

Este și cel mai contestat test din suită. Mai multe lucrări semnalează erori în fundamentarea sa teoretică și în implementarea de referință [3, 4, 5], iar exemplul numeric din documentația NIST este, după cum vom arăta, inconsistent cu propriul cod de referință. Tocmai de aceea se pretează unui studiu critic: nu reproducem o rețetă, ci o verificăm și arătăm unde se rupe.

1.4 Obiectivele și structura lucrării

Obiectivele concrete ale acestei lucrări sunt:

1. implementarea testului într-o bibliotecă C++ fidelă codului de referință NIST, structurată pe componente clare astfel încât să poată găzdui întreaga suită;
2. adăugarea unei transformate rapide (FFT) care permite rularea pe secvențe de ordinul milioanelor de biți, nu doar pe exemplul didactic de 100 de biți;
3. validarea implementării și rularea ei pe *toate* secvențele pe care NIST și-a testat suita;
4. analiza critică a rezultatelor și a controverselor cunoscute, susținută de simulări proprii.

Lucrarea este organizată în cinci capitole. Capitolul 2 prezintă fundamentarea matematică a testului. Capitolul 3 descrie arhitectura implementării. Capitolul 4 conține validarea și simulările. Capitolul 5 discută avantajele, limitările, îmbunătățirile posibile și controversele.

Capitolul 2

Partea matematică: descrierea și justificarea testului

2.1 Procedura testului

Intrări:

- $\varepsilon = \varepsilon_1\varepsilon_2 \ldots \varepsilon_n$ - secvența de biți testată, $\varepsilon_i \in \{0, 1\}$, produsă de generatorul evaluat;
- n - lungimea secvenței (NIST recomandă $n \geq 1000$);
- α - nivelul de semnificație (implicit 0.01).

Ieșiri:

- p - valoarea p a testului, $p \in [0, 1]$;
- decizia: *aleator* (nu se respinge H_0) dacă $p \geq \alpha$, altfel *nealeator*.

Pașii, urmați întocmai și de codul de referință, sunt:

1. **Conversie bipolară**: $x_i = 2\varepsilon_i - 1 \in \{-1, +1\}$.
2. **DFT**: $S = \mathrm{DFT}(X)$, cu $S[k] = \sum_{j=0}^{n-1} x_j\, e^{-2\pi i\, jk/n}$.
3. **Module**: $M_k = |S[k]|$ pentru primele $n/2$ componente, $k = 0, \ldots, n/2 - 1$.
4. **Prag**: $T = \sqrt{\ln(20)\, n}$ (înălțimea de 95%).
5. **Așteptat**: $N_0 = 0.95 \cdot n/2$, numărul *de vârfuri* așteptat sub T (un număr, nu o mărime comparabilă cu T).
6. **Observat**: N_1, câte dintre cele $n/2$ module sunt sub T.
7. **Statistică**: $d = (N_1 - N_0) \,/\, \sqrt{n/4 \cdot 0.95 \cdot 0.05}$.
8. **Valoarea** p: $p = \mathrm{erfc}\big(|d|/\sqrt{2}\big)$.

Decizie (la nivelul $\alpha = 0.01$): dacă $p < \alpha$, secvența este declarată nealeatorie; altfel, ipoteza de aleatorism nu se respinge. Echivalent, se respinge dacă d iese din intervalul de acceptare $[-z_{1-\alpha/2},\, z_{1-\alpha/2}]$ (cu $z_{0.995} = 2.576$ pentru $\alpha = 0.01$).

Justificarea fiecărui pas urmează în restul capitolului.

2.2 Ipoteza nulă și conversia bipolară

Fie $\varepsilon = \varepsilon_1\varepsilon_2 \ldots \varepsilon_n$ secvența de biți testată. Ipoteza nulă H_0 este că biții sunt independenți și identic distribuiți, fiecare cu $\Pr(\varepsilon_i = 0) = \Pr(\varepsilon_i = 1) = 1/2$ (monede corecte, independente).

Primul pas transformă biții în valori *bipolare*:

$$x_i = 2\varepsilon_i - 1 \in \{-1, +1\}. \tag{2.1}$$

Sub H_0, variabilele x_i sunt i.i.d. cu $\mathbb{E}[x_i] = 0$ și $\mathrm{Var}(x_i) = \mathbb{E}[x_i^2] = 1$. Centrarea elimină componenta medie, astfel încât spectrul reflectă structura secvenței, nu dezechilibrul dintre 0 și 1.

2.3 Transformata Fourier discretă

Se aplică transformata Fourier discretă secvenței bipolare:

$$S[k] = \sum_{j=0}^{n-1} x_j \, e^{-2\pi i \, jk/n}, \qquad k = 0, 1, \ldots, n-1. \tag{2.2}$$

Coeficientul $S[k]$ măsoară cât de mult „rezonează" secvența cu frecvența k/n. Modulele $|S[k]|$ formează spectrul de amplitudine. Deoarece x_j sunt reale, spectrul satisface simetria hermitiană

$$S[n-k] = \overline{S[k]}, \tag{2.3}$$

deci $|S[n-k]| = |S[k]|$. Informația neredundantă se află în primele $\lfloor n/2 \rfloor + 1$ componente, $S[0], S[1], \ldots, S[n/2]$, unde:

- $S[0] = \sum_j x_j$ este componenta continuă (DC), reală, egală cu diferența dintre numărul de 1 și numărul de 0;
- $S[n/2]$ (pentru n par) este componenta Nyquist, tot reală.

Testul folosește *primele* $n/2$ *module*, adică $|S[0]|, \ldots, |S[n/2-1]|$.

Observație 2.1. Alegerea exactă a setului de $n/2$ componente (cu sau fără DC, cu sau fără Nyquist) pare un detaliu, dar are consecințe numerice directe și reprezintă una dintre sursele de confuzie discutate în Capitolul 5. Codul de referință numără $|S[0]|, \ldots, |S[n/2-1]|$: *include* componenta DC și *exclude* Nyquist. Mai mult, componenta DC nu are aceeași distribuție ca celelalte (vezi mai jos), ceea ce afectează subtil statistica.

2.4 Distribuția modulelor sub H_0

Scriem $S[k] = A_k + iB_k$ cu

$$A_k = \sum_{j=0}^{n-1} x_j \cos\Big(\frac{2\pi jk}{n}\Big), \qquad B_k = -\sum_{j=0}^{n-1} x_j \sin\Big(\frac{2\pi jk}{n}\Big).$$

Pentru $0 < k < n/2$, A_k și B_k sunt sume de variabile independente mărginite. Prin teorema limită centrală, pentru n mare ele sunt aproximativ gaussiene, de medie nulă. Folosind $\sum_j \cos^2(2\pi jk/n) = n/2$ și analog pentru sinus, obținem

$$A_k \sim \mathcal{N}\Big(0, \tfrac{n}{2}\Big), \qquad B_k \sim \mathcal{N}\Big(0, \tfrac{n}{2}\Big), \tag{2.4}$$

aproximativ independente. Atunci $|S[k]|^2 = A_k^2 + B_k^2$ urmează o lege exponențială de medie n (echivalent, $|S[k]|$ urmează o lege Rayleigh):

$$\Pr\big(|S[k]|^2 \le t\big) = 1 - e^{-t/n}. \tag{2.5}$$

Componenta DC face excepție: $S[0] = \sum_j x_j \sim \mathcal{N}(0, n)$ este reală, deci $|S[0]|^2$ urmează o lege χ^2 cu un singur grad de libertate, nu exponențială. Includerea ei în numărătoare este deci o mică inconsistență de model.

2.5 Pragul de 95% și statistica de test

Căutăm pragul T astfel încât, sub H_0, 95% dintre module să fie sub T:

$$\Pr\big(|S[k]| < T\big) = 1 - e^{-T^2/n} = 0.95 \implies \frac{T^2}{n} = \ln\frac{1}{0.05} = \ln 20, \tag{2.6}$$

de unde

$$\boxed{T = \sqrt{\ln(20)\, n} = \sqrt{2.995732274\, n}\,.} \tag{2.7}$$

Aici ln este logaritmul natural; standardul NIST notează „log”, iar valoarea folosită este cea naturală, $\ln 20 = 2.9957$.

(Versiunea originală a standardului folosea pragul aproximativ $\sqrt{3n}$; valoarea exactă $\sqrt{\ln(20)\, n}$ este una dintre corecțiile propuse de Kim, Umeno și Hasegawa [3] și adoptate ulterior.)

Fie N_1 numărul de module observate sub prag (din cele $n/2$) și N_0 numărul așteptat:

$$N_0 = 0.95 \cdot \frac{n}{2}. \tag{2.8}$$

Dacă cele $n/2$ evenimente $|S[k]| < T$ ar fi independente, atunci N_1 ar urma distribuția binomială Binomial($n/2$, 0.95), de varianță $(n/2)(0.95)(0.05)$. În realitate, componentele spectrale *nu* sunt independente: constrângerea lui Parseval $\sum_k |S[k]|^2 = n\sum_j x_j^2 = n^2$ introduce o corelație care reduce varianța numărătoarei sub valoarea binomială. NIST folosește, empiric, *jumătate* din varianța binomială:

$$d = \frac{N_1 - N_0}{\sqrt{n/4 \cdot 0.95 \cdot 0.05}}. \tag{2.9}$$

Factorul $n/4$ (în loc de $n/2$) este tocmai această corecție pentru dependență. Vom vedea în Capitolul 5 că ea este *aproximativ* corectă, dar nu a fost dedusă analitic, ci estimată numeric, și că valoarea exactă rămâne subiect de dezbatere în literatură.

2.6 Valoarea p și regula de decizie

Sub H_0, d este presupus aproximativ $\mathcal{N}(0,1)$, iar testul este bilateral (abateri în ambele sensuri indică nealeatorism). Valoarea p este

$$p = \operatorname{erfc}\!\Big(\frac{|d|}{\sqrt{2}}\Big) = \Pr\big(|Z| > |d|\big), \quad Z \sim \mathcal{N}(0,1). \tag{2.10}$$

Funcția erorilor și complementara ei sunt

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt, \qquad \operatorname{erfc}(x) = 1 - \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}\,dt. \tag{2.11}$$

Factorul $\sqrt{2}$ vine din normalizarea gaussiană: pentru $Z \sim \mathcal{N}(0,1)$ avem $\Pr(|Z| > a) = \operatorname{erfc}(a/\sqrt{2})$, deci $p = \operatorname{erfc}(|d|/\sqrt{2})$ este exact probabilitatea cozii bilaterale pentru valoarea observată $|d|$.

Regula de decizie, la nivelul $\alpha = 0.01$: dacă $p < \alpha$ secvența este declarată *nealeatorie*; altfel, ipoteza de aleatorism nu se respinge.

Formulare echivalentă cu valoare critică. Deoarece d este, sub H_0, aproximativ $\mathcal{N}(0,1)$, iar testul este bilateral, regula valorii p este identică cu regula clasică a valorii critice citite din tabela normală: se respinge dacă $|d| > z_{1-\alpha/2}$, adică dacă d cade *în afara* intervalului de acceptare $[\,-z_{1-\alpha/2},\, z_{1-\alpha/2}\,]$. Echivalența este exactă:

$$p = \operatorname{erfc}\bigl(|d|/\sqrt{2}\bigr) < \alpha \quad \Longleftrightarrow \quad |d| > z_{1-\alpha/2}. \tag{2.12}$$

Pentru $\alpha = 0.01$, $z_{0.995} = 2.576$, deci secvența se acceptă dacă $d \in [-2.576,\, 2.576]$; pentru $\alpha = 0.05$, intervalul ar fi $[-1.96,\, 1.96]$.

Semnul lui d arată tipul de abatere. Un d puternic negativ înseamnă $N_1 \ll N_0$, adică prea multe vârfuri *peste* prag, deci posibile periodicități. Un d puternic pozitiv înseamnă prea puține vârfuri peste prag (spectru anormal de neted), ceea ce contrazice tot comportamentul zgomotului alb.

2.7 Exemplu numeric pas cu pas

Urmărim cei opt pași pe secvența de 100 de biți din [1] (sec. 2.6.8):

$$\varepsilon = 1100\,1001\,0000\,1111\,1101\,1010\,1010\ldots$$

Pasul 1 (conversie bipolară). Fiecare bit devine $x_i = 2\varepsilon_i - 1$, deci $0 \mapsto -1$ și $1 \mapsto +1$:

$$\varepsilon : 1,1,0,0,1,0,0,1,\ldots \;\longmapsto\; x : +1,+1,-1,-1,+1,-1,-1,+1,\ldots$$

Pasul 2 (DFT). Se calculează numeric (prin FFT) cele 100 de componente complexe $S[0],\ldots,S[99]$. Componenta continuă este reală, $S[0] = \sum_i x_i = (\#1) - (\#0)$.

Pasul 3 (module). Reținem modulele primelor $n/2 = 50$ componente, $M_k = |S[k]|$ pentru $k = 0,\ldots,49$ (deci $S[0]$ inclus, $S[50]$ exclus). Primele opt module, calculate de implementare (identice cu `numpy.fft`):

k	0	1	2	3	4	5	6	7
M_k	16.000	6.449	6.958	7.095	11.865	11.481	7.014	5.337

Toate cele opt sunt sub pragul $T = 17.31$ (cele două module care îl depășesc apar la frecvențe mai înalte). Aici $|S[0]| = 16$ confirmă $S[0] = (\#1) - (\#0) = 42 - 58 = -16$.

Pasul 4 (prag). Înălțimea corespunzătoare nivelului de 95%:

$$T = \sqrt{\ln(20)\, n} = \sqrt{2.995732 \cdot 100} = \sqrt{299.5732} = 17.3082.$$

Pasul 5 (așteptat). Numărul de vârfuri așteptat sub prag:

$$N_0 = 0.95 \cdot \tfrac{n}{2} = 0.95 \cdot 50 = 47.5.$$

Pasul 6 (observat). Câte dintre cele 50 de module sunt efectiv sub prag:

$$N_1 = \#\{\, k : M_k < 17.3082 \,\} = 48.$$

Pasul 7 (statistică). Standardizăm abaterea $N_1 - N_0$:

$$d = \frac{N_1 - N_0}{\sqrt{n/4 \cdot 0.95 \cdot 0.05}} = \frac{48 - 47.5}{\sqrt{25 \cdot 0.0475}} = \frac{0.5}{\sqrt{1.1875}} = \frac{0.5}{1.08972} = 0.45883.$$

Pasul 8 (valoarea p). Probabilitatea cozii bilaterale:

$$p = \operatorname{erfc}\left(\frac{|d|}{\sqrt{2}}\right) = \operatorname{erfc}\left(\frac{0.45883}{1.41421}\right) = \operatorname{erfc}(0.32445) = 0.64636.$$

Decizie. Cum $p = 0.6464 \geq \alpha = 0.01$, secvența nu se respinge: la acest nivel se comportă ca o secvență aleatoare. Echivalent, $d = 0.459$ se află în intervalul de acceptare $[-2.576, 2.576]$, deci în afara zonei critice - aceeași concluzie. Documentația NIST raportează, pentru aceeași secvență, $N_1 = 46$ (deci $d = -1.376$, $p = 0.168669$); și această valoare cade în $[-2.576, 2.576]$, deci verdictul rămâne „aleator”, însă valoarea p diferă mult. Neconcordanța apare la pasul 6 și este analizată în Capitolul 5.

Capitolul 3
Modul de implementare

3.1 Arhitectura bibliotecii

Testul este organizat ca o bibliotecă C++ orientată pe obiecte, cu două cerințe de proiectare în prim-plan: o transformată de complexitate $O(n \log n)$ (un DFT direct $O(n^2)$ ar fi inutilizabil peste câteva mii de biți) și o structură în care fiecare test stă separat, după modelul suitei NIST de referință (care împarte fiecare test în faze `init` / `iterate` / `metrics`). Componentele sunt:

- `DftEngine` (interfață abstractă) cu două implementări, `FftEngine` (radix-2 plus Bluestein, $O(n \log n)$) și `DirectDftEngine` ($O(n^2)$, pentru verificare). Acest tipar Strategy reproduce comutatorul `LEGACY_FFT` / FFTW din codul NIST, dar la nivel de obiecte.
- `BitSequence` concentrează într-un singur loc toate sursele de intrare (șir literal, fișier text, fișier binar, stdin), conversia bipolară și împărțirea în fluxuri.
- `RandomnessTest` (interfață) și `SpectralTest` (implementarea testului 2.6). Interfața uniformă permite analizei de nivel 2 să trateze orice test în același mod.
- `Assessment` calculează, peste mai multe fluxuri, proporția de treceri și uniformitatea valorilor p, exact ca raportul `finalAnalysisReport.txt` al NIST.

Peste bibliotecă stau două executabile generice - `nist_test` (o singură secvență) și `nist_assess` (analiză pe multe fluxuri) - plus suita de teste unitare `dft_tests`. Logica comună (citirea biților, selectarea testului printr-un registru și formatarea raportului) este partajată, fără duplicare.

Arhitectura este, de fapt, gândită pentru întreaga suită SP 800-22, nu doar pentru testul spectral. Fiecare test implementează interfața `RandomnessTest`, întorcând un `TestReport` uniform (valoarea p plus statisticile proprii), și se înregistrează printr-o singură linie într-un registru de teste; restul - rularea generică, analiza de nivel 2 (`Assessment`), API-ul și paginile web - funcționează neschimbat peste oricare test. Pe această structură sunt acum implementate și validate, față de codul de referință și de exemplele NIST, *toate cele 15 teste* ale suitei. În plus, biblioteca este publicată ca aplicație web (Next.js) cu un API JSON ale cărui rute lansează direct binarul C++, accesibilă la `prng-nist-tests.student-dev.ro`. Lucrarea de față rămâne însă concentrată pe testul spectral.

3.2 Nucleul testului

Pașii 1–8 din Capitolul 2 se regăsesc unu-la-unu în metoda `SpectralTest::analyze`:

```
1 SpectralResult SpectralTest::analyze(const BitSequence &seq) const
2 {
3     const std::size_t n = seq.size();
4     // Pasii 1-2: bitii devin +/-1, apoi DFT.
5     const std::vector<Complex> spectrum = engine_->transform(seq.toBipolar());
```

```
6
7       // Pasul 3: primele n/2 module (DC inclus, Nyquist exclus).
8       const std::size_t half = n / 2;
9       // Pasul 4: pragul de 95%.
10      const double threshold = std::sqrt(std::log(20.0) * (double)n);
11
12      // Pasul 6: cate varfuri cad sub prag.
13      long n1 = 0;
14      for (std::size_t i = 0; i < half; i++)
15          if (std::abs(spectrum[i]) < threshold)
16              n1++;
17
18      SpectralResult r;
19      r.n = n; r.threshold = threshold;
20      r.n0 = 0.95 * (double)n / 2.0; // Pasul 5
21      r.n1 = n1;
22      const double denom = std::sqrt((double)n / 4.0 * 0.95 * 0.05);
23      r.d = ((double)n1 - r.n0) / denom; // Pasul 7
24      r.pValue = std::erfc(std::fabs(r.d) / std::sqrt(2.0)); // Pasul 8
25      r.passed = r.pValue >= alpha_;
26      return r;
27  }
```

Listing 3.1: Nucleul testului spectral (simplificat din src/spectral_test.cpp)

Rezultatul este exact cel produs de aritmetica corectă, identic cu al codului de referință NIST (vezi Capitolul 4).

3.3 Transformata rapidă pentru lungimi arbitrare

DFT-ul direct are complexitate $O(n^2)$: pentru un milion de biți ar însemna aproximativ 10^{12} operații pe secvență, deci minute sau ore. Fișierele de date NIST au însă $\approx 10^6$ biți, iar 10^6 nu este o putere a lui 2, deci un FFT radix-2 clasic nu se aplică direct.

Soluția este **algoritmul lui Bluestein** (chirp-z) [10], care exprimă un DFT de lungime arbitrară n printr-o *convoluție*, calculabilă cu un FFT radix-2 de lungime m, cea mai mică putere a lui 2 cu $m \geq 2n-1$. Folosind identitatea $jk = \frac{1}{2}\big(j^2 + k^2 - (k-j)^2\big)$,

$$S[k] = e^{-i\pi k^2/n} \sum_{j=0}^{n-1} \Big(x_j \, e^{-i\pi j^2/n} \Big) \, e^{i\pi (k-j)^2/n}, \tag{3.1}$$

adică o convoluție liniară între semnalul modulat $a_j = x_j e^{-i\pi j^2/n}$ și nucleul simetric $b_\ell = e^{i\pi \ell^2/n}$. Complexitatea devine $O(n \log n)$. Argumentul j^2 este redus modulo $2n$ înainte de scalare, pentru a păstra precizia unghiului chiar și pentru n de ordinul milioanelor.

`FftEngine` alege automat calea rapidă: dacă n este putere a lui 2, aplică direct FFT radix-2; altfel comută pe Bluestein. În practică, o secvență de 10^6 biți este procesată în 0.43 secunde (vezi Capitolul 4).

3.4 Intrare/ieșire și analiza de nivel 2

`BitSequence` acceptă atât fișiere ASCII (în care se păstrează doar caracterele `'0'` și `'1'`, ignorând spații și linii noi, cum sunt `data.e`, `data.pi`) cât și fișiere binare (în care fiecare

octet este despachetat în 8 biți, MSB întâi, cum este `data.sha1`). Detecția formatului este automată.

Pentru a reproduce analiza globală a NIST, `Assessment` împarte secvența în fluxuri consecutive, rulează testul pe fiecare și calculează:

- **proporția** de fluxuri trecute și intervalul de acceptare $\hat{p} \pm 3\sqrt{\hat{p}\,\alpha/s}$, cu $\hat{p} = 1 - \alpha$ și s numărul de fluxuri;
- **uniformitatea** valorilor p, printr-un test χ^2 pe un histogram cu 10 intervale, transformat în valoare p prin funcția gamma incompletă regularizată $Q(a, x)$ (portată din Cephes, ca în NIST).

3.5 Compilare și validare

Proiectul se compilează cu CMake și Ninja:

```
cmake -S . -B build -G Ninja -DCMAKE_BUILD_TYPE=Release
cmake --build build
```

Listing 3.2: Compilare

Suita `dft_tests` (doctest) verifică, printre altele, două lucruri: (i) că `FftEngine` și `DirectDftEngine` coincid până la 10^{-9} pe intrări aleatoare de diverse lungimi (inclusiv ne-puteri ale lui 2), și (ii) că testul reproduce exact rezultatul codului de referință NIST pe exemplul din secțiunea 2.6.8. Pentru aceasta din urmă am extras și compilat FFT-ul original (`__ogg_fdrfftf` din `dfft.c`) și am rulat logica originală `DiscreteFourierTransform`, obținând valorile față de care ne validăm.

Capitolul 4
Simulări și rezultate

4.1 Validarea implementării

Înainte de a interpreta orice rezultat, am verificat corectitudinea numerică. Suita de teste `dft_tests` confirmă că motorul rapid `FftEngine` coincide cu DFT-ul direct până la eroare de ordinul 10^{-13} pentru lungimi de la $n = 2$ până la $n = 1000$, inclusiv lungimi care nu sunt puteri ale lui 2 (de exemplu $n = 100$ sau $n = 257$, care activează calea Bluestein). Așadar transformata rapidă nu introduce erori semnificative.

Pe exemplul de validare din NIST SP 800-22, secțiunea 2.6.8 (secvența de 100 de biți din dezvoltarea binară a lui π), implementarea noastră produce:

$$N_1 = 48, \quad N_0 = 47.5, \quad d = 0.458831, \quad p = 0.646355,$$

valori *identice* cu cele produse de codul de referință NIST original, pe care l-am compilat separat. Documentația NIST raportează în schimb $N_1 = 46$ și $p = 0.168669$; această neconcordanță este analizată în Capitolul 5.

4.2 Comportamentul spectral: aleator versus periodic

Figura 4.1 ilustrează principiul testului. La stânga, o secvență aleatoare de 1024 biți are un spectru relativ plat, cu aproximativ 5% dintre vârfuri peste pragul T, exact cum prezice H_0. La dreapta, o secvență cu perioada 8 concentrează energie la armonicele frecvenței fundamentale: apar câteva vârfuri mult peste prag, iar restul spectrului este aproape nul.

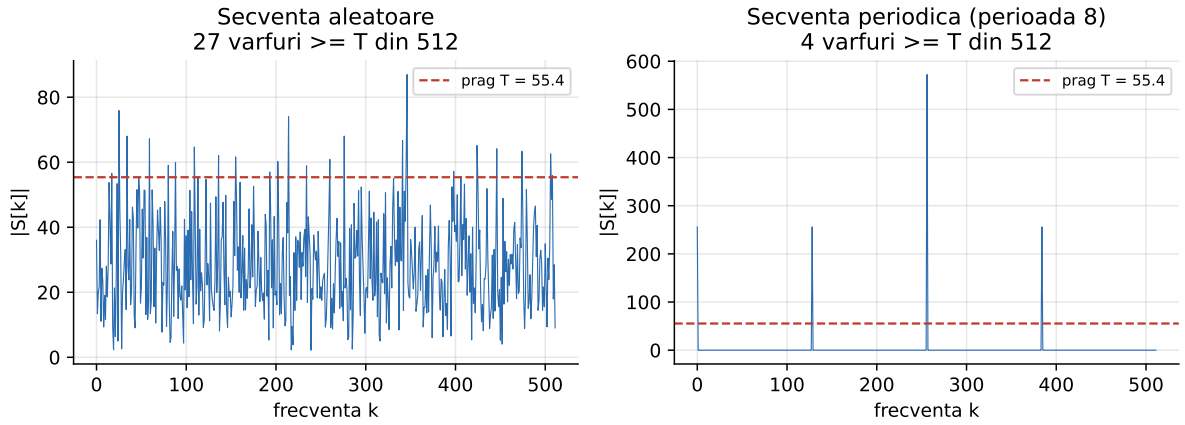

Figura 4.1: Spectrul de amplitudine $|S[k]|$ pentru o secvență aleatoare (stânga) și una periodică (dreapta). Linia roșie este pragul de 95%.

Pe *edge cases* - secvențe cu structură evidentă - implementarea răspunde corect: tiparul `0001` ($n = 10000$) sau o secvență numai de 1 dau $p \approx 0$. Un caz aparte este perioada

2 (`0101...`): vârful ei spectral cade fix pe componenta Nyquist, singura exclusă din numărătoare. Secvența este totuși respinsă, dar indirect - toate componentele numărate rămân aproape nule, deci N_1 atinge maximul $n/2$, iar d devine puternic pozitiv.

4.3 Rularea pe fișierele de date NIST

Am rulat testul pe toate secvențele pe care NIST și-a validat suita: dezvoltările binare ale constantelor e, π, $\sqrt{2}$, $\sqrt{3}$ și un generator criptografic bazat pe SHA-1. Fiecare fișier (aproximativ 10^6 biți) a fost împărțit în 100 fluxuri de 10000 de biți; pentru fiecare flux am calculat o valoare p, apoi proporția de treceri și uniformitatea, ca în Tabelul 4.1.

Tabela 4.1: Rezultatele testului spectral pe fișierele de date NIST (100 fluxuri $\times$ 10000 biți, $\alpha = 0.01$).

Fișier	Fluxuri	Proporție treceri	Uniformitate p	Verdict
`data.e`	100	99/100	0.000233	PASS
`data.pi`	100	100/100	0.162606	PASS
`data.sqrt2`	100	100/100	0.048716	PASS
`data.sqrt3`	100	100/100	0.003996	PASS
`data.sha1`	100	100/100	0.455937	PASS

Toate cele cinci surse trec testul. Se remarcă însă că uniformitatea pentru `data.e` este la limită (0.000233, abia peste pragul 0.0001): valorile p ale fluxurilor lui e sunt mai puțin uniform distribuite decât cele ale celorlalte constante. Figura 4.2 compară histogramele pentru π (uniform) și e (ușor neuniform).

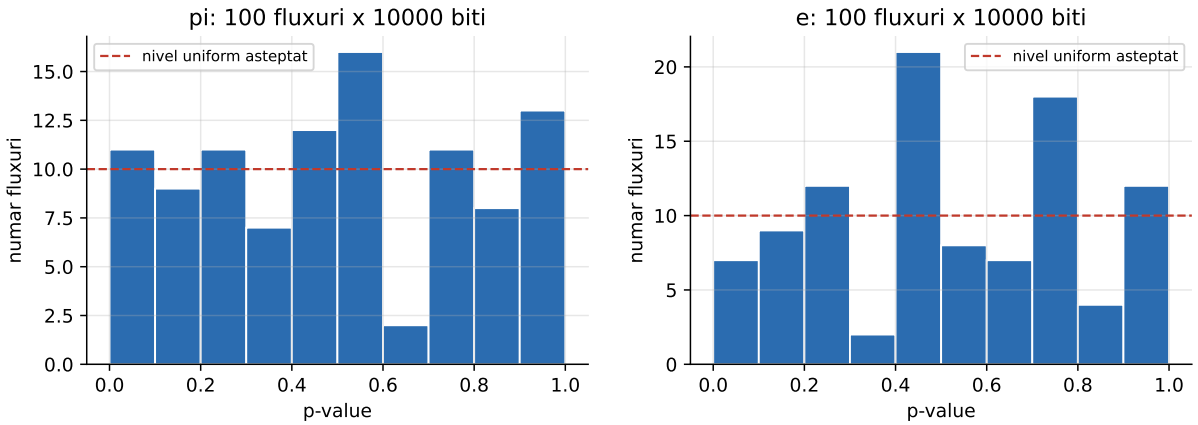

Figura 4.2: Histograma valorilor p pe cele 100 de fluxuri, pentru π (stânga) și e (dreapta). Linia roșie este nivelul uniform așteptat.

4.4 Performanță

Transformata rapidă face testul practic chiar și la scară mare. Tabelul 4.2 rezumă timpii măsurați (g++ 14.2, optimizare `-O2`, o secvență, respectiv întreaga analiză pe un fișier).

Tabela 4.2: Timpi de rulare.

Sarcină	Timp
Un FFT pe 10^6 biți (Bluestein)	0.43 s
500 fluxuri $\times$ 10^4 biți (toate cele 5 fișiere)	1.7 s

4.5 Simulare Monte-Carlo sub H_0

Pentru a studia comportamentul testului pe un generator ideal, am generat 10^6 secvențe aleatoare de $n = 4096$ biți și am calculat distribuția statisticilor. Secvențele provin dintr-un generator de calitate (PCG64, prin `numpy.random.default_rng`), nu din `rand()` din C; astfel distribuția empirică reflectă testul în sine, nu eventuale defecte ale generatorului. Figura 4.3 arată distribuția empirică a lui N_1 și a valorilor p.

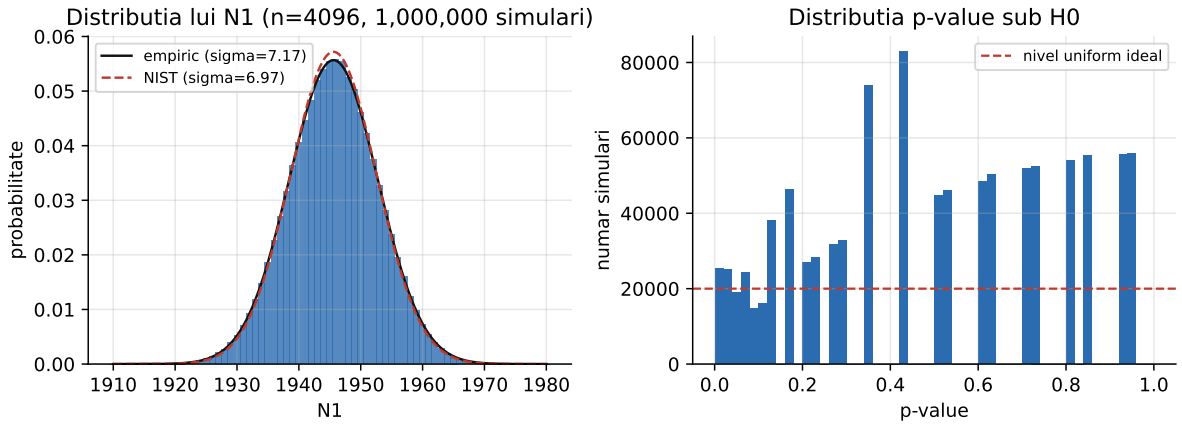

Figura 4.3: Stânga: distribuția lui N_1 peste 10^6 simulări (bare), cu curbele gaussiene teoretice folosind σ_{emp} și σ_{NIST}. Dreapta: distribuția valorilor p sub H_0.

Deviația standard empirică a lui N_1 este $\sigma_{\text{emp}} \approx 7.17$, în timp ce deviația standard presupusă de NIST (numitorul din ecuația (2.9)) este $\sigma_{\text{NIST}} = \sqrt{n/4 \cdot 0.95 \cdot 0.05} \approx 6.97$. Raportul este ≈ 1.03, deci varianța empirică a statisticii d este $\text{Var}(d) \approx 1.056$ în loc de 1, iar d este ușor mai dispersat decât $\mathcal{N}(0, 1)$. Rata empirică de respingere sub H_0 urcă astfel la aproximativ 0.012, peste valoarea nominală 0.01: testul respinge generatoare *bune* ceva mai des decât ar trebui.

Abaterea, deși reală, este mică: factorul $n/4$ ales de NIST prinde cea mai mare parte a corecției pentru dependența spectrală. Valoarea măsurată $\text{Var}(d) \approx 1.056$ se potrivește cu predicția $4/3.8 \approx 1.05$ ce rezultă din constanta corectată $c = 3.8$ propusă de Pareschi, Rovatti și Setti [5] - tocmai controversa teoretică din capitolul următor.

4.6 Deviația standard empirică pe fiecare generator

Simularea Monte-Carlo de mai sus folosește un singur generator (PCG64). Este firesc să ne întrebăm dacă ușoara umflare a varianței ($\text{Var}(d) \approx 1.056$) se vede și pe celelalte surse. Am repetat măsurarea pe cele cinci fișiere de date NIST: fiecare fișier ($\approx 10^6$ biți) a fost

împărțit în blocuri de $n = 4096$ biți, am calculat N_1 pe fiecare bloc, iar σ_{emp} este deviația standard a acestor valori. O comparăm cu $\sigma_{\text{NIST}} = \sqrt{n/4 \cdot 0.95 \cdot 0.05} \approx 6.97$.

Tabela 4.3: Deviația standard empirică a lui N_1 pe fiecare generator ($n = 4096$, $\sigma_{\text{NIST}} = 6.97$). Ultimul rând este simularea Monte-Carlo de referință (PCG64, 10^6 secvențe), mult mai precisă.

Generator	Blocuri	$\overline{N_1}$	σ_{emp}	$\sigma_{\text{emp}}/\sigma_{\text{NIST}}$	Var(d)
`data.e`	245	1946.0	6.95	0.997	0.994
`data.pi`	245	1945.5	6.77	0.971	0.943
`data.sqrt2`	245	1946.2	6.79	0.973	0.947
`data.sqrt3`	245	1946.0	7.56	1.084	1.176
`data.sha1`	244	1945.5	6.74	0.967	0.934
PCG64 (Monte-Carlo)	10^6	1945.6	7.17	1.028	1.056

Cele cinci surse reale se grupează strâns în jurul lui σ_{NIST} (rapoarte între 0.97 și 1.08, medie 0.998); niciuna nu prezintă o varianță anormală, deci sub statistica spectrală toate se comportă cum prezice H_0. Trebuie însă subliniat că un singur fișier oferă doar ≈ 245 de blocuri, adică o incertitudine de eșantionare de circa $1/\sqrt{2 \cdot 245} \approx 4.5\%$ pe σ_{emp}. La acest nivel de zgomot, umflarea de $\approx 5\%$ a variației nu poate fi rezolvată dintr-un singur fișier: intervalul [0.97, 1.08] acoperă deopotrivă valoarea 1.00 și valoarea 1.03 măsurată precis de simulare. Cu alte cuvinte, abaterea este atât de mică încât doar cele 10^6 de secvențe ale simulării Monte-Carlo o pot distinge clar de zero; datele reale sunt compatibile cu ea, fără să o contrazică.

Capitolul 5
Avantaje, limitări, îmbunătățiri și controverse

5.1 Avantaje

- **Unic în suită.** Este singurul test care lucrează în domeniul frecvenței. El poate detecta periodicități globale, răspândite pe toată secvența, pe care testele locale din domeniul timpului (frecvență, blocuri, treceri) le pot rata.
- **Intuiție clară.** Noțiunea de „vârf spectral peste prag" oferă o imagine geometrică directă a abaterii de la zgomotul alb.
- **Practic la scară mare.** Cu o transformată $O(n \log n)$, testul rulează pe milioane de biți în fracțiuni de secundă, deci poate fi aplicat pe eșantioane realiste, nu doar pe exemple didactice.

5.2 Controversa 1: documentația își contrazice propriul cod

Cea mai cunoscută problemă privește chiar exemplul de validare. Documentația NIST (secțiunea 2.6.8) afirmă, pentru secvența de 100 de biți, că $N_1 = 46$ și $p = 0.168669$. Însă o transformată Fourier *corectă* produce $N_1 = 48$ și $p = 0.646355$. Nu este vorba de o eroare a implementării noastre: am compilat și rulat codul de referință NIST original (FFT-ul `__ogg_fdrfftf` din `dfft.c`) și acesta produce tot $N_1 = 48$. Așadar *documentația contrazice propriul cod de referință*. Verificarea independentă cu `numpy` confirmă valoarea 48. Mai mult, nicio convenție rezonabilă de selecție a componentelor (cu sau fără DC, cu sau fără Nyquist) nu produce 46: ele dau între 47 și 49, deci 46 nu este o interpretare alternativă validă, ci o eroare de calcul. Că setările testului DFT din SP 800-22 sunt eronate a fost semnalat independent încă din 2004 de Kim, Umeno și Hasegawa [3], care au corectat atât pragul ($\sqrt{3n} \to \sqrt{\ln(20)\, n}$), cât și varianța de normalizare.

Reproducerea valorii documentate ar necesita un prag sensibil mai mic decât cel specificat. Figura 5.1 arată de ce: două vârfuri de la limită, $|S[23]| = 16.81$ și $|S[40]| = 17.20$, se află sub pragul corect $T = 17.31$ (deci sunt numărate, $N_1 = 48$), dar ar fi excluse de un prag cu aproximativ 5% mai mic (≈ 16.48), caz în care $N_1 = 46$. Numărătoarea este, așadar, foarte sensibilă la valoarea exactă a pragului, iar valoarea documentată 46 nu corespunde pragului specificat. Această fragilitate, împreună cu problema variației (controversa următoare), arată că parametrii testului (pragul și normalizarea) ar trebui corectați pe baze teoretice, exact în spiritul corecțiilor obiective propuse de autorii care au studiat testul [3, 4, 5, 6].

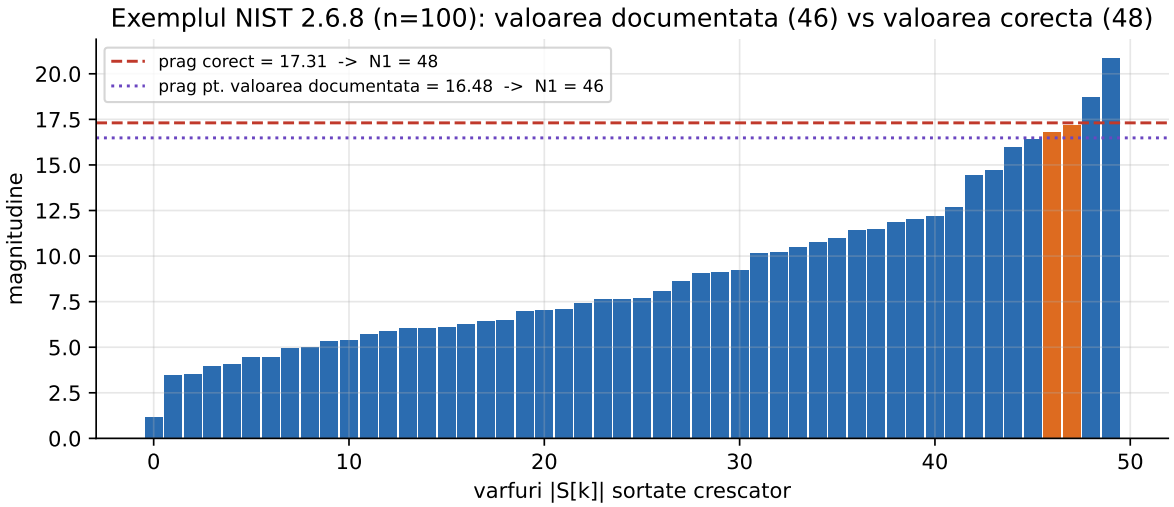

Figura 5.1: Cele 50 de module ale exemplului NIST, sortate. Pragul corect (17.31) dă $N_1 = 48$; pentru a obține valoarea documentată 46 ar fi necesar un prag cu aproximativ 5% mai mic (≈ 16.48), care ar exclude cele două vârfuri portocalii. Numărătoarea este sensibilă la valoarea exactă a pragului.

Implementarea noastră este fidelă aritmeticii corecte și codului de referință ($N_1 = 48$); discrepanța față de valoarea documentată și sensibilitatea de mai sus arată că testul are nevoie de o corecție principială a parametrilor, în linia celei propuse de autorii care l-au studiat.

5.3 Controversa 2: distribuția de referință nu a fost dedusă, ci estimată

Critica cea mai profundă, formulată de Kim, Umeno și Hasegawa [3] (2004) și de Hamano [4] (2005), este că *distribuția de referință a testului nu a fost dedusă matematic*, ci estimată numeric, rulând un generator pseudoaleator presupus „bun”. Aceasta este o formă de raționament circular: testul care ar trebui să certifice un generator a fost calibrat presupunând că un alt generator este deja perfect. Drept consecință, atât pragul, cât și varianța au fost inițial aproximate, nu demonstrate.

Kim et al. au corectat mai multe astfel de aproximări, printre care pragul, care a trecut de la valoarea originală $\sqrt{3n}$ la valoarea exactă $\sqrt{2.995732274\,n} = \sqrt{\ln(20)\,n}$ folosită azi. Această corecție a fost adoptată în standard, dar varianța a rămas neschimbată, ceea ce ne aduce la controversa următoare.

5.4 Controversa 3: varianța de normalizare

A treia problemă privește numitorul din statistica d (ecuația (2.9)). NIST folosește varianța $n/4 \cdot 0.95 \cdot 0.05$, adică jumătate din varianța binomială naivă. Hamano [4] a arătat, plecând de la distribuția spectrului, că dependența indusă de teorema lui Parseval reduce varianța reală la aproximativ jumătate din cea binomială, ceea ce justifică *de ce* factorul $n/4$ este aproape corect. Totuși, „aproape” nu înseamnă „exact”:

- Pareschi, Rovatti și Setti [5] (2012) au arătat, prin analiză și experimente, că o variantă $0.95 \cdot 0.05 \cdot n/c$ cu $c \approx 3.8$ (în loc de $c = 4$) se potrivește mai bine cu distribuția reală, iar rata efectivă de respingere diferă de cea nominală;
- lucrări ulterioare au derivat varianța corectă pornind explicit de la teorema lui Parseval [6], arătând că statistica d nu urmează exact $\mathcal{N}(0, 1)$; pe lângă variantă, persistă și o deplasare a mediei (asimetria valorilor statisticii) [8], iar analizele de erori ale testelor de nivel 2 leagă astfel de abateri de rate de fals-pozitiv mai mari la eșantioane foarte mari [9].

Simularea noastră Monte-Carlo (Capitolul 4) reproduce acest efect: varianța empirică a statisticii d este ≈ 1.056, în loc de 1, în acord cantitativ cu valoarea $4/3.8 \approx 1.05$ implicată de constanta $c = 3.8$ a lui Pareschi. Rata de respingere sub H_0 urcă astfel de la 0.01 la aproximativ 0.012. Pentru un singur flux efectul este mic, dar la analiza agregată pe multe fluxuri (proporție și uniformitate) el se acumulează și poate duce la respingerea unor generatoare bune.

5.5 Alte limitări și capcane cunoscute

- **Bug istoric în codul de referință.** Versiunea originală calcula $m[i + 1]$ până la $i = n/2 - 1$, accesând $X[n]$, adică un element *dincolo* de capătul vectorului (citire în afara limitelor). Eroarea nu afecta numărătoarea și a fost corectată în fork-urile întreținute [2], dar ilustrează fragilitatea codului.
- **Alegerea componentelor.** Includerea componentei DC (care are distribuție χ^2_1, nu exponențială) și excluderea celei Nyquist este o convenție subtilă. Multe reimplementări independente inversează alegerea (exclud DC, includ Nyquist) și obțin un N_1 diferit, de unde confuzia frecventă în jurul „valorii corecte”.
- **Componentele DC și Nyquist.** Un *edge case* este perioada exact 2: vârful ei cade pe componenta Nyquist, exclusă din numărătoare. Secvența este totuși respinsă, dar indirect (toate componentele numărate fiind nule, N_1 atinge maximul), nu prin observarea directă a vârfului.
- **Testele de nivel 2 sunt ele însele slabe.** Testele de proporție și de uniformitate folosite pentru a agrega valorile p au putere statistică redusă; au fost propuse variante mai fiabile [8].
- **Putere de detecție limitată.** În comparații cu suite moderne, testul spectral este considerat printre cele mai slabe din SP 800-22, motiv pentru care au fost propuse teste spectrale alternative (folosind, de exemplu, transformate Fourier corect normalizate, transformata sinus discretă sau abordări de tip QFT) [7].

5.6 Îmbunătățiri posibile

- Folosirea variantei asimptotice corecte (Hamano [4], Pareschi [5]) sau calibrarea prin simulare, astfel încât rata de respingere să coincidă cu cea nominală.
- Înlocuirea motorului propriu cu o bibliotecă FFT optimizată (FFTW) pentru secvențe foarte mari, păstrând interfața `DftEngine`.
- Includerea componentei Nyquist și tratarea separată a componentei DC, pentru ca vârfurile de la capete să fie observate direct și pentru a elimina inconsistența de model (cu recalibrarea corespunzătoare a lui N_0).

- Corectarea documentației oficiale: faptul că exemplul a rămas greșit prin mai multe revizii (până în Rev 1a, 2010), deși corecțiile erau publicate din 2004, este în sine o controversă de mentenanță a standardului.

5.7 Concluzie

Testul spectral are o calitate reală - este singurul din suită care privește secvența în domeniul frecvenței - dar trebuie folosit cu rezerve. Distribuția sa de referință nu a fost dedusă analitic, ci estimată numeric; exemplul din documentație nu corespunde codului; varianța de normalizare este doar aproximativ corectă. În practică ar trebui folosit alături de alte teste, nu izolat, iar verdictul interpretat ținând cont de abaterea sistematică măsurată mai sus. Implementarea de aici, validată față de codul de referință, oferă o bază curată pentru îmbunătățirile propuse.

Bibliografie

[1] L. E. Bassham et al., *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, NIST Special Publication 800-22 Revision 1a, 2010. DOI: 10.6028/NIST.SP.800-22r1a. `https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software`

[2] L. C. Noll, T. Gilgan, R. Paccagnella, *sts: NIST Statistical Test Suite (versiune refactorizata)*, depozit GitHub, `https://github.com/arcetri/sts` (clonat 2026).

[3] S.-J. Kim, K. Umeno, A. Hasegawa, *Corrections of the NIST Statistical Test Suite for Randomness*, arXiv:nlin/0401040; IACR Cryptology ePrint Archive, Report 2004/018, 2004. `https://arxiv.org/abs/nlin/0401040`

[4] K. Hamano, *The Distribution of the Spectrum for the Discrete Fourier Transform Test Included in SP800-22*, IEICE Transactions on Fundamentals, vol. E88-A, nr. 1, pp. 67–73, 2005. DOI: 10.1093/ietfec/e88-a.1.67.

[5] F. Pareschi, R. Rovatti, G. Setti, *On Statistical Tests for Randomness Included in the NIST SP800-22 Test Suite and Based on the Binomial Distribution*, IEEE Transactions on Information Forensics and Security, vol. 7, nr. 2, pp. 491–505, 2012. DOI: 10.1109/TIFS.2012.2185227.

[6] A. Iwasaki, *Deriving the Variance of the Discrete Fourier Transform Test Using Parseval's Theorem*, arXiv:1806.10357, 2018 (IEEE Transactions on Information Theory, vol. 66, nr. 2, pp. 1164–1170, 2020). `https://arxiv.org/abs/1806.10357`

[7] A. Iwasaki, K. Umeno, *A new randomness test solving problems of Discrete Fourier Transform Test*, arXiv:1708.08218, 2017. `https://arxiv.org/abs/1708.08218`

[8] S. Zhu, Y. Ma, J. Lin, J. Zhuang, J. Jing, *More Powerful and Reliable Second-level Statistical Randomness Tests for NIST SP 800-22*, ASIACRYPT 2016, LNCS 10031, Springer; IACR Cryptology ePrint Archive 2016/863. `https://eprint.iacr.org/2016/863`

[9] D. Chen, H. Chen, L. Fan, K. Luo, *Error Analysis of NIST SP 800-22 Test Suite*, IEEE Transactions on Information Forensics and Security, vol. 18, pp. 3745–3759, 2023. DOI: 10.1109/TIFS.2023.3287391.

[10] L. I. Bluestein, *A linear filtering approach to the computation of discrete Fourier transform*, IEEE Transactions on Audio and Electroacoustics, vol. 18, nr. 4, pp. 451–455, 1970. DOI: 10.1109/TAU.1970.1162132.