

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
FEDERAL STATE AUTONOMOUS EDUCATIONAL
INSTITUTION OF HIGHER EDUCATION
NORTH CAUCASUS FEDERAL UNIVERSITY

Methodical Guidelines

on Laboratory Works

on the discipline

**MODERN DIGITAL SIGNAL PROCESSING / СОВРЕМЕННАЯ ЦИФРОВАЯ
ОБРАБОТКА СИГНАЛОВ**

for students of the Master's program

02.04.01 Mathematics and Computer Sciences

profile: Training Program for Electronic Information

Stavropol, 2026

CONTENT

Introduction

Laboratory Work №1: Discrete-Time Signals, Sampling Theory and Spatial Operations

Laboratory Work №2: Frequency Domain Analysis: DFT, FFT and Spectral Visualization

Laboratory Work №3: Z-Transform, LTI Systems and Stability Analysis

Laboratory Work №4: FIR and IIR Filter Design: Windowing, Bilinear Transform, Pole-Zero Placement

Laboratory Work №5: Multirate DSP, Decimation, Interpolation and Filter Banks

Laboratory Work №6: Random Processes and Parametric Spectral Estimation (AR, MA, ARMA)

Laboratory Work №7: Adaptive Filtering (LMS/RLS) and Time-Frequency Analysis (STFT, Wavelets)

Laboratory Work №8: End-to-End Signal Processing Pipeline (Project Workshop)

Guidelines for Independent Work

Evaluation Criteria and Competency Assessment

List of Recommended Literature

INTRODUCTION

1. Discipline Purpose and Position in the Educational Program

The discipline "Modern Digital Signal Processing / Современная цифровая обработка сигналов" (Index: Б1.О.10) belongs to Block 1, Mandatory Part of the Master's educational program 02.04.01 Mathematics and Computer Sciences. It is implemented in Semester 2 and carries 4 credit units (144 academic hours: 16 h lectures, 16 h laboratory sessions, 112 h independent work).

Purpose of the Laboratory Component:

To develop students' practical skills in implementing classical and Modern Digital Signal Processing / Современная цифровая обработка сигналов algorithms from first principles, ensuring deep understanding of mathematical foundations, numerical stability, spectral properties, and computational trade-offs before transitioning to high-level library abstractions or deep learning frameworks.

List of planned learning outcomes in the discipline (module) correlated with the planned outcomes of the educational program

Competency code, description	Indicator code, description	Planned learning outcomes in the discipline (module) characterizing the stages of competencies and indicators formation
UC-2 Able to conduct independent research and analyze complex technical problems using modern mathematical and computational methods	UC-2.1 Formulates research hypotheses, selects optimal algorithmic approaches, and justifies methodological choices based on signal characteristics and system constraints	Applies time-frequency representations, spectral estimation, and transform-domain models to design robust signal processing pipelines for real-world time-series and sensor data.
OPC-2 Able to design, implement, and optimize computational algorithms for processing and analyzing discrete signals	OPC-2.1 Develops efficient, numerically stable signal processing routines from first principles, ensuring reproducibility and computational efficiency	Implements core DSP algorithms from scratch using Python, NumPy, and SciPy, demonstrating understanding of numerical stability, computational complexity, and algorithmic assumptions.
OPC-3 Able to plan and execute experimental studies, validate algorithmic performance against theoretical models, and critically interpret results	OPC-3.1 Designs complete processing pipelines, performs ablation and sensitivity analysis, and documents methodological limitations and failure modes	Designs complete processing workflows (sampling → filtering → spectral analysis → feature extraction), critically evaluates parameter sensitivity and noise robustness, and justifies the integration of classical methods with hybrid AI architectures.
PC-2 Able to develop mathematical models and software systems for system identification, spectral analysis, adaptive filtering, and hybrid classical-AI signal	PC-2.1 Applies transform-domain, statistical, and optimization-based methods to solve inverse problems, estimate system parameters, and integrate classical DSP with modern machine learning	Solves system identification and spectral estimation problems, designs optimal and adaptive filters, and reconstructs signal characteristics using linear prediction, AR/MA models, and convex optimization on parameter

processing	architectures	manifolds.
------------	---------------	------------

2. Structure of Laboratory Sessions

Each laboratory session follows a 2-hour format:

- **First 30 minutes:** Brief theoretical recap and task clarification
- **Next 75 minutes:** Hands-on implementation with instructor consultation
- **Final 15 minutes:** Results demonstration and preliminary discussion

All assignments are submitted via the university LMS as Jupyter notebooks containing:

- Executable code with inline mathematical comments
- Visualizations of time-domain, frequency-domain, and pole-zero representations
- Quantitative evaluation metrics (SNR, spectral bias, convergence error, stability margin, etc.)
- Critical analysis of limitations, numerical precision effects, and failure modes

3. Technical Requirements

- Python 3.9+ with NumPy, SciPy, scikit-image, OpenCV-Python, Matplotlib
- JupyterLab environment (provided via university cloud workspace)
- Access to curated datasets: Middlebury (stereo), KITTI (visual odometry), COCO subsets (keypoints)
- Version control via Git for project submissions

LABORATORY WORK №1

Discrete-Time Signals, Sampling Theory and Spatial Operations

Objective:

Master discrete-time signal representation, sampling theorem verification, and basic time-domain operations by implementing convolution, interpolation, and geometric transformations from scratch.

Competencies Assessed: OPC-2.1, UC-2.1

Theoretical Background:

- Continuous-to-discrete sampling; Nyquist-Shannon theorem; aliasing phenomena
- Discrete-time signal types (impulse, step, exponential, sinusoidal)
- Linear convolution vs. circular convolution; overlap-add/save methods
- Interpolation techniques for signal reconstruction (zero-order, linear, sinc)

Tasks:

1. Custom Convolution Implementation

- 1.a. Implement linear and circular convolution without `scipy.signal` functions
- 1.b. Support arbitrary signal lengths, zero-padding, and boundary handling modes
- 1.c. Validate against analytical solutions on test signals (impulse response, step input)

2. Aliasing Demonstration Pipeline

- 2.a. Generate band-limited signals; subsample below and above Nyquist rate
- 2.b. Visualize spectral folding; quantify reconstruction error using ideal low-pass filtering
- 2.c. Compare aliasing effects across different sampling strategies

3. Sinc Interpolation from First Principles

- 3.a. Implement ideal band-limited interpolation using truncated sinc kernel
- 3.b. Compare with linear and nearest-neighbor interpolation in terms of SNR and frequency response
- 3.c. Document Gibbs phenomenon and propose windowing mitigation

Questions for Self-Check:

1. Why does circular convolution alias linear convolution results, and how does zero-padding prevent it?
2. How does sampling rate affect the discrete-time frequency representation of a continuous sinusoid?
3. What are the computational complexities of direct convolution vs. FFT-based overlap-add?

Report Requirements:

- Code with mathematical annotations for each algorithmic step
- Time-frequency visualizations of aliasing and interpolation artifacts
- Quantitative table: execution time, memory usage, reconstruction SNR for each operation
- Critical discussion: when would you prefer direct convolution over FFT-based methods?

Recommended Literature:

- Oppenheim and Schaffer, Ch. 2 (Discrete-Time Signals and Systems)
- Proakis and Manolakis, Sec. 2.1–2.4 (Sampling and Reconstruction)
- SciPy Documentation: `scipy.signal.convolve` (for verification only)

LABORATORY WORK №2

Frequency Domain Analysis: DFT, FFT and Spectral Visualization

Objective: Understand the relationship between time and frequency representations; implement DFT/FFT-based spectral analysis and analyze frequency response properties.

Competencies Assessed: OPC-2.1, UC-2.1, OPC-3.1

Theoretical Background:

- Discrete Fourier Transform: definition, symmetry, periodicity, shift theorem
- Fast Fourier Transform: Cooley-Tukey decimation-in-time algorithm, butterfly diagram
- Frequency domain filtering: ideal, windowed low/high/band-pass filters
- Spectral leakage, zero-padding, and windowing techniques

Tasks:

1. Manual FFT Implementation

- 1.a. Compute DFT using direct summation; implement radix-2 FFT from scratch
- 1.b. Verify linearity, conjugate symmetry, and Parseval's theorem
- 1.c. Visualize magnitude/phase spectra with proper frequency axis scaling

2. Frequency-Domain Filtering Pipeline

- 2.a. Design windowed low-pass filter using Hamming and Kaiser windows
- 2.b. Apply filter in frequency domain; reconstruct signal via inverse FFT
- 2.c. Compare with time-domain convolution: visual quality, computational cost, edge preservation

3. Spectral Resolution Experiment

- 3.a. Analyze closely spaced sinusoids under different record lengths and window functions
- 3.b. Quantify frequency resolution using Rayleigh criterion; discuss trade-offs

Questions for Self-Check:

1. Why does zero-padding improve visual spectral smoothness but not actual frequency resolution?
2. How does the choice of window function affect main-lobe width and side-lobe attenuation?
3. What does phase information encode that makes it critical for signal reconstruction?

Report Requirements:

- Mathematical derivation of FFT decomposition steps
- Spectral visualizations with annotated frequency axes
- Quantitative comparison: SNR, runtime for spatial vs. frequency filtering
- Analysis of resolution experiment with perceptual examples

LABORATORY WORK №3

Z-Transform, LTI Systems and Stability Analysis

Objective:

Master Z-domain representation of discrete-time systems, analyze stability and causality, and implement system simulation.

Competencies Assessed: UC-2.1, PC-2.1

Theoretical Background:

- Z-transform: definition, region of convergence (ROC), properties
- System function $H(z)$: poles, zeros, ROC-stability relationship
- Difference equations: direct-form I/II, cascade, parallel realizations
- Stability analysis: Jury test, pole magnitude constraints, BIBO stability

Tasks:

1. Z-Transform Computation and ROC Visualization

- 1.a. Compute Z-transform of causal/anti-causal sequences analytically and numerically
- 1.b. Plot pole-zero diagrams; shade ROC regions on complex plane
- 1.c. Verify convolution theorem in Z-domain

2. System Realization and Stability Check

- 2.a. Implement direct-form II transposed structure for given transfer function
- 2.b. Test stability using pole magnitude and impulse response decay
- 2.c. Compare numerical precision effects in floating-point vs. fixed-point simulation

3. Inverse Z-Transform via Residues and Long Division

- 3.a. Implement partial fraction expansion for rational $H(z)$
- 3.b. Compute inverse transform analytically; verify against numerical simulation
- 3.c. Analyze transient vs. steady-state response components

Questions for Self-Check:

1. Why does the ROC determine causality and stability simultaneously?
2. How does coefficient quantization shift pole locations and affect stability?
3. What are the advantages of cascade/parallel forms over direct-form structures?

Report Requirements:

- Pole-zero plots with stability boundary annotations
- Impulse/step response visualizations with transient/steady-state separation

- Quantitative table: stability margin, numerical error for different realizations
- Critical discussion: when would you prefer cascade form for implementation?

Recommended Literature:

- Oppenheim and Schaffer, Ch. 3 (The Z-Transform)
- Proakis and Manolakis, Ch. 3 (Z-Transform and Analysis of LTI Systems)
- Lyons, Ch. 6 (IIR Filters)

LABORATORY WORK №4

FIR and IIR Filter Design: Windowing, Bilinear Transform, Pole-Zero Placement

Objective:

Develop practical skills in designing finite and infinite impulse response filters using classical and optimization-based methods.

Competencies Assessed: OPC-2.1, OPC-3.1, PC-2.1

Theoretical Background:

- FIR design: windowing, frequency sampling, Parks-McClellan (Remez) algorithm
- IIR design: impulse invariance, bilinear transformation, pole-zero placement
- Filter specifications: passband/stopband ripple, transition width, group delay
- Phase linearity: minimum-phase, linear-phase, all-pass decomposition

Tasks:

1. Windowed FIR Design Pipeline

- 1.a. Implement ideal low/high-pass filter derivation via inverse DTFT
- 1.b. Apply Hamming, Blackman, Kaiser windows; quantify passband ripple and stopband attenuation
- 1.c. Compare linear phase vs. minimum phase implementations

2. Bilinear IIR Design

- 2.a. Design analog Butterworth/Chebyshev prototype; apply bilinear transform with pre-warping
- 2.b. Verify digital frequency response matches analog specifications
- 2.c. Analyze phase distortion and group delay variation

3. Remez Exchange Algorithm Implementation

- 3.a. Implement simplified Parks-McClellan for equiripple FIR design
- 3.b. Compare with windowed design in terms of order efficiency and transition sharpness
- 3.c. Document computational cost and convergence behavior

Questions for Self-Check:

1. Why does the bilinear transform map the entire $j\omega$ -axis to the unit circle, avoiding aliasing?
2. How does Kaiser window parameter β control the trade-off between main-lobe width and side-lobe level?
3. What are the stability implications of pole-zero cancellation in IIR design?

Report Requirements:

- Filter specification tables with measured vs. target parameters
- Magnitude/phase/group delay plots with specification boundaries
- Parameter sweep plots showing order vs. transition width
- Discussion of implementation constraints: fixed-point effects, quantization noise

Recommended Literature:

- Oppenheim and Schaffer, Ch. 7 (Filter Design Techniques)
- Mitra, Ch. 4–5 (FIR and IIR Filter Design)
- `scipy.signal` documentation: `remez`, `bilinear` (for verification)

LABORATORY WORK №5

Multirate DSP, Decimation, Interpolation and Filter Banks

Objective:

Master sample rate conversion, polyphase structures, and filter bank theory.

Competencies Assessed: OPC-2.1, UC-2.1

Theoretical Background:

- Decimation and interpolation: spectral effects, anti-aliasing/reconstruction filtering
- Rational rate conversion: cascade of upsampling/downsampling
- Polyphase decomposition: computational efficiency, filter bank realization
- Quadrature Mirror Filters (QMF) and perfect reconstruction conditions

Tasks:

1. Decimation and Interpolation Implementation

- 1.a. Implement integer decimation/interpolation with proper filtering
- 1.b. Visualize spectral folding and image components; quantify distortion
- 1.c. Compare naive vs. polyphase implementations in terms of multiply-add count

2. Rational Rate Conversion Pipeline

- 2.a. Implement L/M conversion using upsampling, filtering, and downsampling
- 2.b. Design optimal anti-aliasing filter using Kaiser window
- 2.c. Validate output spectrum against theoretical expectations

3. Two-Channel Filter Bank

- 3.a. Implement analysis/synthesis bank using polyphase decomposition
- 3.b. Test perfect reconstruction on test signals; measure reconstruction error
- 3.c. Analyze aliasing cancellation mechanism

Questions for Self-Check:

1. Why must decimation be preceded by low-pass filtering to prevent aliasing?
2. How does polyphase decomposition reduce computational complexity by a factor of M?
3. What conditions must filter coefficients satisfy for perfect reconstruction in a QMF bank?

Report Requirements:

- Spectral visualization of decimation/interpolation stages
- Computational complexity comparison: direct vs. polyphase
- Reconstruction error analysis for filter banks under different input classes
- Critical discussion: trade-offs in sample rate conversion for real-time systems

Recommended Literature:

- Vaidyanathan, Ch. 1–3 (Multirate Systems)
- Oppenheim and Schaffer, Ch. 10 (Multirate Signal Processing)
- Lyons, Ch. 10 (Decimation and Interpolation)

LABORATORY WORK №6

Random Processes and Parametric Spectral Estimation (AR, MA, ARMA)

Objective:

Implement statistical signal processing algorithms, analyze stationarity, and estimate power spectral density using parametric methods.

Competencies Assessed: PC-2.1, OPC-3.1

Theoretical Background:

- Random signals: autocorrelation, PSD, ergodicity, stationarity
- Non-parametric estimation: periodogram, Bartlett, Welch methods
- Parametric estimation: AR, MA, ARMA models; Yule-Walker, Burg algorithms
- Bias-variance trade-offs in spectral estimation

Tasks:

1. Welch PSD Estimation

- 1.a. Implement segment averaging with Hann window and 50% overlap
- 1.b. Compare with raw periodogram; quantify variance reduction
- 1.c. Validate against known theoretical PSD of AR(1) process

2. Burg AR Parameter Estimation

- 2.a. Implement forward-backward linear prediction error minimization
- 2.b. Estimate AR coefficients; compute reflection coefficients
- 2.c. Derive PSD from AR parameters; compare with true spectrum

3. Model Order Selection

- 3.a. Implement AIC, BIC, and MDL criteria for AR order selection
- 3.b. Analyze overfitting vs. underfitting effects on spectral peaks
- 3.c. Document computational complexity and convergence behavior

Questions for Self-Check:

1. Why does the periodogram exhibit high variance, and how does segment averaging reduce it?
2. How does Burg's method ensure stability of AR parameter estimates?
3. What is the theoretical justification for BIC penalizing model complexity more heavily than AIC?

Report Requirements:

- PSD plots with confidence intervals and theoretical benchmarks
- Reflection coefficient decay visualization for AR processes
- Model order selection tables with AIC/BIC/MDL values

- Critical discussion: when to prefer parametric vs. non-parametric methods

Recommended Literature:

- Proakis and Manolakis, Ch. 12 (Power Spectrum Estimation)
- Haykin, Ch. 2–3 (Random Processes)
- `scipy.signal` documentation: `welch`, `lpc` (for verification)

LABORATORY WORK №7

Adaptive Filtering (LMS/RLS) and Time-Frequency Analysis (STFT, Wavelets)

Objective:

Implement adaptive algorithms and time-frequency representations for non-stationary signal analysis.

Competencies Assessed: UC-2.1, OPC-3.1

Theoretical Background:

- LMS and NLMS algorithms: gradient descent, step-size stability, convergence
- RLS algorithm: recursive least squares, forgetting factor, numerical stability
- STFT: windowed Fourier transform, time-frequency resolution trade-off
- Discrete Wavelet Transform: filter banks, multiresolution analysis, thresholding

Tasks:

1. LMS Adaptive Filter Implementation

- 1.a. Implement system identification using LMS with unknown FIR channel
- 1.b. Analyze convergence speed vs. step-size; measure steady-state misadjustment
- 1.c. Compare with NLMS normalization effects

2. RLS Implementation

- 2.a. Implement recursive inverse correlation matrix update
- 2.b. Compare convergence speed with LMS under identical conditions
- 2.c. Analyze numerical stability with and without regularization

3. STFT and DWT Time-Frequency Analysis

- 3.a. Compute spectrogram using variable window lengths; analyze resolution trade-offs
- 3.b. Implement Haar/DB2 DWT; visualize approximation/detail coefficients
- 3.c. Compare energy compaction for piecewise-smooth vs. oscillatory signals

Questions for Self-Check:

1. Why does the LMS step-size bound depend on the maximum eigenvalue of the input correlation matrix?
2. How does the forgetting factor in RLS affect tracking speed and noise immunity?
3. What is the uncertainty principle limitation in STFT, and how do wavelets overcome it?

Report Requirements:

- Learning curve plots for LMS/RLS under different step-sizes/forgetting factors
- Time-frequency spectrograms with resolution annotations
- Wavelet coefficient energy distribution plots

- Critical discussion: algorithm selection for real-time vs. offline applications

Recommended Literature:

- Haykin, Ch. 5–6 (Adaptive Filters)
- Oppenheim and Schaffer, Ch. 11 (Time-Frequency Analysis)
- Mallat, Ch. 7–8 (Wavelets)

LABORATORY WORK №8

End-to-End Signal Processing Pipeline (Project Workshop)

Objective:

Integrate classical DSP modules into a complete, reproducible pipeline for a real-world task.

Competencies Assessed: OPC-3.1, PC-2.1

Theoretical Background:

- Pipeline architecture design: data flow, error propagation, modular abstraction
- Hybrid classical/deep approaches: DSP priors for neural networks, learned feature extraction with classical verification
- Reproducibility practices: version control, environment specification, automated testing
- Performance evaluation: benchmark datasets, ablation studies, failure mode analysis

Tasks:

Select one project track and implement a complete pipeline:

1. Track A: Real-Time Noise Cancellation

Adaptive filtering (LMS/RLS) → step-size optimization → convergence analysis → validation on synthetic/real interference

2. Track B: Spectral Feature Extraction Pipeline

Multiresolution analysis → parametric modeling → dimensionality reduction → classification-ready feature vectors

3. Track C: Hybrid Signal Enhancement

Classical denoising (wavelet/Wiener) → neural residual learning → objective quality assessment → computational bottleneck analysis

All tracks require: modular code design, configuration management, automated metric computation, and comprehensive documentation.

Questions for Self-Check:

1. How does error accumulation in multi-stage pipelines affect final feature quality?
2. When should classical DSP constraints be preferred over end-to-end learning, and vice versa?
3. What are the minimum requirements for a reproducible DSP experiment in a collaborative environment?

Report Requirements:

- Pipeline architecture diagram with data flow and dependency mapping
- Quantitative results on benchmark signals (SNR improvement, spectral accuracy, convergence time)
- Ablation study: impact of removing/modifying individual modules
- Failure mode analysis: non-stationary inputs, numerical overflow, computational bottlenecks
- Discussion of hybrid extension potential: where classical methods outperform or complement deep learning

Recommended Literature:

- Proakis and Manolakis, Full pipeline chapters
- Haykin, Adaptive Filter Theory (advanced applications)
- Course textbooks (Oppenheim and Schaffer, Lyons) for foundational references
- Selected arXiv papers on hybrid classical/neural pipelines (to be provided via LMS)

List of references recommended for use on this topic

Essential References:

1. Oppenheim, A. V., and Schafer, R. W. Discrete-Time Signal Processing. 3rd Edition. Pearson, 2010.
2. Proakis, J. G., and Manolakis, D. G. Digital Signal Processing: Principles, Algorithms, and Applications. 4th Edition. Pearson, 2007.
3. Lyons, R. G. Understanding Digital Signal Processing. 3rd Edition. Pearson, 2010.

Supplementary References:

1. Mitra, S. K. Digital Signal Processing: A Computer-Based Approach. 4th Edition. McGraw-Hill, 2011.
2. Vaidyanathan, P. P. Multirate Systems and Filter Banks. Prentice Hall, 1993.
3. Haykin, S. Adaptive Filter Theory. 5th Edition. Pearson, 2013.

Online Resources:

- SciPy Signal Documentation: <https://docs.scipy.org/doc/scipy/reference/signal.html>
- NumPy FFT Documentation: <https://numpy.org/doc/stable/reference/routines.fft.html>
- arXiv eess.SP section: <https://arxiv.org/list/eess.SP/recent>
- DSPGuide.com: <http://www.dspguide.com>