EIS Analysis Toolkit

Version: v0.9.0 (2025-12-30)

Modular toolkit for electrochemical impedance spectroscopy (EIS) analysis with Distribution of Relaxation Times (DRT) support.

Key features:

- **Reproducibility** - Objective, data-driven parameter selection
- **Modularity** - Usable as CLI and Python library
- **Advanced optimization** - Multi-start, Differential Evolution

Supported data formats:

- **Gamry DTA** - native format (automatic metadata parsing, ZCURVE block)
- **CSV** - three columns with header: `frequency`, `Z_real`, `Z_imag`
  - Delimiter: comma, semicolon, or tab (auto-detection)
  - Decimal format: US (dot) and European (comma for semicolon-delimited)
  - Comments: lines starting with # are ignored
  - Examples: example/example_eis_data.csv

Changelog: CHANGELOG.md

Quick start

Installation

```
git clone <repository-url>
cd eis_analysis
pip install -e .
```

For alternative installation methods, see Installation options.

Basic usage

```
# Basic analysis (KK validation + DRT)
eis.py data.DTA

# With circuit fitting
eis.py data.DTA --circuit "R(100) - (R(5000) | C(1e-6))"

# Save plots to files
eis.py data.DTA --save results --format pdf
```

Run `eis.py --help` for all options.

Common workflows

Data quality check

```
# KK validation + DRT analysis (default)
eis.py data.DTA
```

Output: Kramers-Kronig residuals, mu metric, DRT spectrum with detected peaks.

Circuit fitting

```
# Fit equivalent circuit
eis.py data.DTA --circuit "R(100) - (R(5000) | C(1e-6))"

# Use Differential Evolution for global optimization (default)
eis.py data.DTA --circuit "R(100) - (R(5000) | Q(1e-6, 0.9))"

# Use multi-start for local optimization
eis.py data.DTA --circuit "..." --optimizer multistart --multistart 20
```

Automatic circuit suggestion

```
# Automatic Voigt chain fitting
eis.py data.DTA --voigt-chain

# With automatic element count optimization
eis.py data.DTA --voigt-chain --voigt-auto-M
```

Oxide layer analysis

```
# Thickness calculation from capacitance
eis.py data.DTA --circuit "R(100) - (R(5000) | C(1e-6))" --analyze-oxide

# With custom parameters
eis.py data.DTA --circuit "..." --analyze-oxide --epsilon-r 22 --area 0.5
```

Batch processing

```
# Process multiple files without interactive display
for f in *.DTA; do
    eis.py "$f" --save "${f%.DTA}" --no-show
done
```

Main features

Kramers-Kronig validation

Data quality verification using Lin-KK test. Validates causality, linearity, and stability of measured data.

```
# Default (included in standard analysis)
eis.py data.DTA

# Skip KK validation
eis.py data.DTA --no-kk

# Custom mu threshold
eis.py data.DTA --mu-threshold 0.80
```

Detailed documentation: doc/LinKK_analysis.md

DRT analysis

Distribution of Relaxation Times - model-free method for impedance data analysis. Automatic regularization parameter selection using GCV (Generalized Cross-Validation).

```
# Default DRT with auto-lambda
eis.py data.DTA

# Manual lambda selection
eis.py data.DTA --lambda 1e-3

# GMM peak detection (more robust)
eis.py data.DTA --peak-method gmm

# Robust R_inf estimation for inductive data
eis.py data.DTA --ri-fit

# Peak classification into physical processes
eis.py data.DTA --peak-method gmm --classify-terms
```

Detailed documentation: doc/GCV_IMPLEMENTATION.md, doc/GMM_PEAK_DETECTION.md

Circuit fitting

Elegant operator overloading syntax for circuit definition.

Supported elements:

Element	Description	Example
`R(value)`	Resistor	`R(100)`
`C(value)`	Capacitor	`C(1e-6)`
`L(value)`	Inductor	`L(1e-6)`
`Q(Q, n)`	Constant Phase Element (CPE)	`Q(1e-4, 0.8)`
`W(sigma)`	Warburg (semi-infinite)	`W(50)`
`Wo(R_W, tau)`	Warburg (bounded)	`Wo(100, 1.0)`
`K(R, tau)`	Voigt with tau parametrization	`K(1000, 1e-4)`

Values in parentheses serve as initial guesses for the nonlinear fitting algorithm. Values in quotes (e.g., `R("100")`) are treated as fixed constants and will not be fitted.

Operators:

Operator	Meaning	Example
`-`	Series connection	`R(100) - C(1e-6)`
`|`	Parallel connection	`R(1000) | C(1e-6)`

Operator precedence: `-` has HIGHER precedence than `|` (Python rules). Always use parentheses around parallel combinations: `(R|C)`.

```
# Voigt element
eis.py data.DTA --circuit "R(100) - (R(5000) | C(1e-6))"

# Randles circuit with CPE
eis.py data.DTA --circuit "R(10) - (R(100) | Q(1e-4, 0.8))"

# With fixed parameter
eis.py data.DTA --circuit 'R("0.86") - (R(2.4e9) | Q(1e-10, 0.823))'
```

Detailed documentation: doc/CIRCUIT_PARSER.md, doc/K_ELEMENT_GUIDE.md

Voigt chain (automatic circuit)

Automatic Voigt chain estimation using linear regression for initial guess, then nonlinear refinement.

```
# Basic Voigt chain
eis.py data.DTA --voigt-chain

# Automatic element count optimization
eis.py data.DTA --voigt-chain --voigt-auto-M

# Custom density (elements per decade)
eis.py data.DTA --voigt-chain --voigt-n-per-decade 5
```

Detailed documentation: doc/VOIGT_CHAIN_MATH.md

Advanced optimization

Differential Evolution (default) Global optimization for finding global minimum.

```
# Default DE
eis.py data.DTA --circuit "R(100) - (R(5000) | C(1e-6))"

# Custom parameters
eis.py data.DTA --circuit "..." --de-strategy 2 --de-popsize 20 --de-maxiter 500
```

DE strategies: 1=randtobest1bin (default), 2=best1bin, 3=rand1bin

Detailed documentation: doc/DIFFERENTIAL_EVOLUTION.md

Multi-start optimization Multiple starts from different initial points.

```
# Multi-start with 20 restarts
eis.py data.DTA --circuit "..." --optimizer multistart --multistart 20

# With larger perturbation
eis.py data.DTA --circuit "..." --optimizer multistart --multistart-scale 3.0
```

Detailed documentation: doc/MULTISTART_OPTIMIZATION.md

Oxide layer analysis

Oxide layer thickness calculation from capacitance.

```
# Automatic analysis
eis.py data.DTA --circuit "..." --analyze-oxide

# Custom permittivity and area
eis.py data.DTA --circuit "..." --analyze-oxide --epsilon-r 9 --area 0.5
```

Common permittivities: ZrO2 ~ 22, Al2O3 ~ 9, TiO2 ~ 80, SiO2 ~ 3.9

Detailed documentation: doc/OXIDE_ANALYSIS_GUIDE.md

CLI Reference

Input data and frequency range

- `input` - Input file (.DTA or .csv). Without argument, synthetic data is used for testing. Built-in test circuit: **Rs - (R0||Q0) - (R1||Q1)** with 1% Gaussian noise:
  - Rs = 10 Ω (series resistance)
  - R0 = 100 kΩ, Q0 = (1e-6 S·s^n, n=0.6)
  - R1 = 800 kΩ, Q1 = (3e-5 S·s^n, n=0.43)
- `--f-min` - Minimum frequency [Hz]. Data below this value will be cut off. Useful for removing noise at low frequencies.
- `--f-max` - Maximum frequency [Hz]. Data above this value will be cut off. Useful for removing artifacts at high frequencies.

Circuit fitting

- `--circuit`, `-c` - Equivalent circuit for fitting. Syntax: `-` = series, `|` = parallel. Example: `"R(100) - (R(5000) | C(1e-6))"`. Supported elements: R, C, L, Q, W, Wo, K.
- `--weighting` (default: sqrt) - Weighting type for fitting: `uniform` (all points equal), `sqrt` (square root of 1/|Z|), `proportional` (1/|Z|), `square` (1/|Z|^2).
- `--no-fit` - Skip circuit fitting.

Optimizer selection

- `--optimizer` (default: de) - Optimizer type: `de` (Differential Evolution - global), `multistart` (multiple local fits), or `single` (one local fit).

Differential Evolution (global optimization)

- `--de-strategy` (default: 1) - DE strategy: 1=randtobest1bin (balanced, default), 2=best1bin (fast convergence), 3=rand1bin (more exploration).
- `--de-popsize` (default: 15) - Population size as multiple of parameter count. Higher = better exploration but slower.
- `--de-maxiter` (default: 1000) - Maximum number of generations. Increase if optimization doesn't converge.
- `--de-tol` (default: 0.01) - Convergence tolerance (relative fitness change).
- `--de-workers` (default: 1) - Number of parallel workers. -1 = all CPU cores.

Multi-start optimization

- `--multistart N` - Number of restarts for multi-start optimization. Each restart starts from perturbed initial values. Default: 16 when `--optimizer multistart` is used.
- `--multistart-scale` (default: 2.0) - Perturbation size in sigma units (standard deviation from covariance matrix). Higher = larger parameter space exploration.

DRT analysis

- `--lambda`, `-l` (default: auto GCV) - Regularization parameter for DRT. Without this parameter, automatic selection using GCV (Generalized Cross-Validation) and L-curve method is used. Higher values = smoother DRT, lower = more detail but also noise.
- `--n-tau`, `-n` (default: 100) - Number of points on the tau time constant axis. Higher values give finer DRT resolution but increase computational cost.
- `--normalize-rpol` - Normalize gamma(tau) by polarization resistance so that integral = 1. Useful for comparing samples with different R_pol.
- `--peak-method` (default: scipy) - Peak detection method in DRT: `scipy` (fast, scipy.signal.find_peaks) or `gmm` (robust, Gaussian Mixture Model - requires scikit-learn).
- `--ri-fit` - Robust R_inf estimation using R+L+K model fit on high-frequency data. Suitable for data with inductive loop.
- `--classify-terms` - Classification of DRT peaks into physical processes (requires `--peak-method gmm`).
- `--no-drt` - Skip DRT analysis. Useful if you only want circuit fitting.

Kramers-Kronig validation

- `--no-kk` - Skip Kramers-Kronig validation. KK test verifies causality, linearity, and stability of data.
- `--mu-threshold` (default: 0.85) - Threshold value of mu metric for Lin-KK test. Values below threshold indicate problematic data.

- `--ocv` - Display OCV (Open Circuit Voltage) curve if available in data.

Voigt chain (automatic circuit)

- `--voigt-chain` - Use automatic Voigt chain fitting. Linear regression for R_i and tau_i estimation, then nonlinear refinement.
- `--voigt-n-per-decade` (default: 3) - Number of time constants per decade for Voigt chain. Higher = finer coverage but more parameters.
- `--voigt-extend-decades` (default: 0.0) - Extend tau range by N decades beyond data limits. Useful if you expect processes outside measured range.
- `--voigt-prune-threshold` (default: 0.01) - Threshold for removing small R_i (as fraction of R_pol). Elements with R_i < threshold * R_pol are removed.
- `--voigt-allow-negative` - Allow negative R_i values (Lin-KK style). Otherwise negative elements are removed.
- `--voigt-no-inductance` - Do not include series inductance L in model.
- `--voigt-fit-type` (default: complex) - Fit type: `complex` (default, real+imag), `real` (real part only), `imag` (imaginary part only).
- `--voigt-auto-M` - Automatically optimize number of M elements using mu metric.
- `--voigt-mu-threshold` (default: 0.85) - Mu threshold value for `--voigt-auto-M`.
- `--voigt-max-M` (default: 50) - Maximum number of M elements for `--voigt-auto-M`.
- `--no-voigt-info` - Do not display detailed Voigt chain fitting info.

Oxide layer analysis

- `--analyze-oxide` - Perform oxide layer analysis - thickness calculation from capacitance.
- `--epsilon-r` (default: 22.0) - Relative permittivity of oxide. Default 22 for ZrO2. Other oxides: Al2O3 ~ 9, TiO2 ~ 80, SiO2 ~ 3.9.
- `--area` (default: 1.0) - Electrode area [cm^2]. Required for correct thickness calculation.

Jacobian

- `--numeric-jacobian` - Use numeric Jacobian instead of analytic. Analytic Jacobian is faster and more accurate but not available for all elements. Use this option for custom/non-standard elements.

Output and visualization

- `--save`, `-s` - Save plots with this prefix. Example: `--save results` creates `results_nyquist.png`, etc.
- `--format`, `-f` (default: png) - Format of saved plots: `png` (raster), `pdf/svg/eps` (vector for publications).
- `--no-show` - Do not display plots interactively. Useful for batch processing with `--save`.
- `-v`, `--verbose` - Show debug messages on stderr (prefix `[DEBUG]`).
- `-q`, `--quiet` - Quiet mode - hide INFO messages, show only warnings and errors.

Logging levels:

- INFO (default): stdout, no prefix
- WARNING: stdout, prefix `!`
- ERROR: stderr, prefix `!!`
- DEBUG (`-v`): stderr, prefix `[DEBUG]`

Installation options

Option 1: pip install (recommended)

```
git clone <repository-url>
cd eis_analysis
pip install -e .    # Editable install
# or
pip install .       # Standard install
```

Option 2: Dependencies only

```
pip install -r requirements.txt
python3 eis.py --help
```

Option 3: System packages (Debian/Ubuntu)

```
sudo apt install python3-numpy python3-scipy python3-matplotlib
```

Optional dependencies

```
# GMM peak detection
pip install scikit-learn

# Development tools
pip install -e ".[dev]"
```

Windows

```
git clone <repository-url>
cd eis_analysis
pip install -e .
python eis.py data.DTA    # Use 'python' instead of 'python3'
```

Python API

Complete Python API: doc/PYTHON_API.md

Documentation

Document	Description
doc/PYTHON_API.md	Complete Python API reference
doc/CIRCUIT_PARSER.md	Circuit parser syntax
doc/K_ELEMENT_GUIDE.md	K element guide
doc/LinKK_analysis.md	Kramers-Kronig validation
doc/GCV_IMPLEMENTATION.md	GCV technical documentation
doc/GMM_PEAK_DETECTION.md	GMM peak detection
doc/DRT_METHOD_ANALYSIS.md	DRT method analysis
doc/VOIGT_CHAIN_MATH.md	Voigt chain mathematics
doc/DIFFERENTIAL_EVOLUTION.md	Differential Evolution
doc/MULTISTART_OPTIMIZATION.md	Multi-start optimization
doc/NONLINEAR_FIT_ANALYSIS.md	Nonlinear optimization overview
doc/OXIDE_ANALYSIS_GUIDE.md	Oxide layer analysis
doc/CODE_ANALYSIS_REPORT.md	Code structure analysis

References

- Orazem, M.E., Tribollet, B.: Electrochemical Impedance Spectroscopy (2008)
- Schonleber, M. et al.: "A Method for Improving the Robustness of linear Kramers-Kronig Validity Tests", Electrochimica Acta 131 (2014)
- Wahba, G.: "A comparison of GCV and GML", Annals of Statistics 13 (1985)
- Saccoccio, M. et al.: "Optimal regularization in DRT", Electrochimica Acta 147 (2014)

This code was developed with the assistance of Claude Code.