

Keywords: relay protection, Java, overcurrent protection, transformer differential protection, distance protection, offline simulation

Dmytro DANYLCHENKO ¹, Vladyslav TSIUPA ¹, Oleksandr MIROSHNYK ²,
 Taras SHCHUR ^{2,3*}, Katarzyna PIOTROWSKA ⁴

¹ National Technical University “Kharkiv Polytechnic Institute”, Ukraine, dmytro.danylchenko@khp.edu.ua, vladyslav.tsiupa@gmail.com

² State Biotechnological University, Ukraine, omiroshnyk@btu.kharkiv.ua, shchurtg@gmail.com

³ University of Life Sciences in Lublin, Poland

⁴ Lublin University of Technology, Poland, k.piotrowska@pollub.pl

* Corresponding author: shchurtg@gmail.com, k.piotrowska@pollub.pl

Digitalisation of relay protection and implementation of the main protections in digital form

Abstract

This paper presents a Java-based software demonstration of overcurrent, transformer differential, and distance protection. Its contribution is an implementation-oriented treatment of three complementary protection functions, with explicit signal-processing, decision, and timing logic. The transformer differential component builds on the authors' earlier study; the present scope also includes four-stage overcurrent logic and a comparison of polygonal and circular distance characteristics. The methodology combines a scalar overcurrent example with offline processing of fault records for the other functions. Measurements of the Java implementation yielded approximately 30–70 ms operating time, 5–20 μ s processing time per sample, and a JVM memory footprint of approximately 30–80 MB. At the adopted 4 kHz sampling rate, the reported per-sample processing time corresponds to 2–8% of the 250 μ s sampling interval. These measurements support computational feasibility under the tested conditions, while qualitative timing variability remains a limitation. The results do not establish a worst-case execution bound, statistical reliability, or compliance with protection-equipment standards.

1. INTRODUCTION

Relay protection detects abnormal electrical conditions and initiates fault isolation. Electromechanical, static analogue, and numerical relays represent established implementation technologies; digitisation alone is therefore not a novel protection principle. Numerical implementations enable configurable measurement and decision functions, but their accuracy and timing must be evaluated on the complete acquisition, processing, and output chain (Bakkar et al., 2024).

Protection functions extend beyond the three considered here. They include directional and earth-fault overcurrent, line and busbar differential, under- and overvoltage, frequency, thermal, breaker-failure, and travelling-wave protection. Commercial equipment combines these functions according to the requirements of the protected asset and network (Shad et al., 2025). Overcurrent, transformer differential, and distance protection are selected here as representative functions covering current-threshold decisions, a bounded transformer zone, and impedance-based line protection. They are not an exhaustive set of universally primary protections; overcurrent may also serve as backup protection (Draganova-Zlateva & Georgiev, 2021).

Overcurrent protection compares a current magnitude with pickup settings and associated delays (Gaggero et al., 2025). Transformer differential protection compares compensated winding currents and applies restraint and inrush blocking to distinguish internal faults from through-current and energisation conditions. Distance protection compares apparent loop impedance with an operating zone (Gyori et al., 2013). This order is followed throughout Sections 2 and 3.

Existing implementations provide a demanding baseline for comparison. (Bakkar et al., 2022). Implemented distance, directional overcurrent, voltage, and frequency functions on an FPGA using parallel processing and pipelining, and evaluated accuracy, latency, and resource consumption. (Li et al., 2025)

investigated system-on-chip protection combining hardware acceleration and software processing, including experimental verification and engineering applications. Garibello-Narváez et al. (2024) examined the effects of distributed energy resources on distance-relay operation. These studies show that implementation architecture and network conditions both matter; a Java translation alone cannot establish improved protection performance.

The transformer differential algorithm follows Danylchenko et al. (2023), Digital implementation of relay protection algorithms: Transformer differential protection. That component is treated as prior work, not as a newly invented algorithm. The contribution of the present article is the broader implementation scope, the explicit treatment of overcurrent and distance decisions, and a consistent separation of record-time behaviour from runtime performance (Qawaqzeh et al., 2021). The practical gap addressed is an inspectable software description linking input records, Fourier estimates, protection characteristics, and element states. The study does not claim a new fault-detection principle or a measured advantage over industrial relays.

For deployed fault-clearing functions, timing should be treated as a hard real-time requirement: a decision after the permitted clearing deadline can fail the protection objective (Miroshnyk et al., 2023). A firm real-time classification would also make a late result unacceptable, but it does not justify occasional missed protection deadlines; a soft real-time model is unsuitable for the trip path (Allamanis et al., 2018). The allowable delay is application-specific and must be based on coordination, equipment withstandability, and system stability studies. Offline replay has no wall-clock deadline. Relevant functional evaluation frameworks include IEC 60255-151 for overcurrent protection, IEC 60255-187-1 for transformer differential protection, and IEC 60255-121 for distance protection (Barua & Cheon, 2015).

Section 2 describes the software structure and algorithms. Section 3 discusses the reported examples, timing interpretation, computational requirements, and validation limitations. Section 4 states the conclusions within that evidence boundary.

2. MATERIALS AND METHODS

Figure 1 summarises the Java classes and their responsibilities. Main connects the record reader, phase or loop elements, and the shared Trip object. DataComtrade reconstructs channels from configuration and sample files (Arefin & Khatchadourian, 2015). Furie maintains Fourier estimates; Dzt and Dz implement transformer differential and distance decisions. The overcurrent example receives a scalar current magnitude separately (Tymchuk & Miroshnyk, 2015). Listings 1–4 replace full class definitions with pseudocode that makes the decision logic explicit; they are algorithm specifications, not a claim that the revised code has been experimentally revalidated.

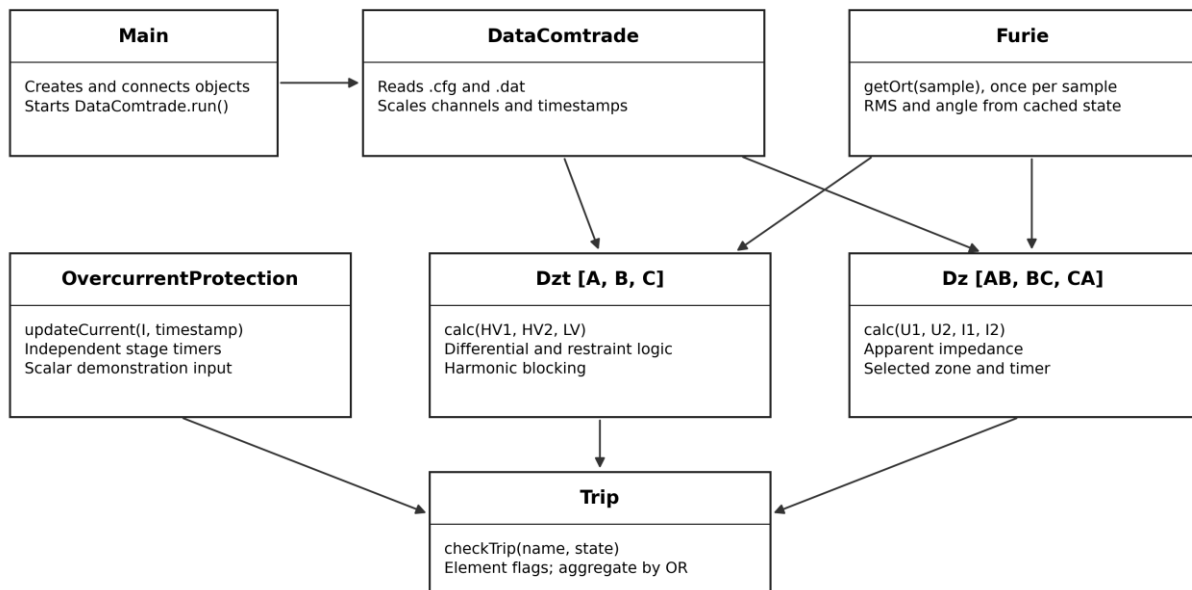


Fig. 1. Class responsibilities and data dependencies in the protection examples

For the 50 Hz examples, $N = 80$ samples per cycle corresponds to a sampling frequency of 4 kHz and a sample interval $\Delta t = 0.25$ ms. The reader must obtain timing and scaling from the COMTRADE configuration rather than infer them from line numbers. Analogue channels are reconstructed as $y = a \cdot \text{raw} + b$, with CT/VT ratios and primary/secondary units handled consistently (Allamanis et al., 2018). Files with different sampling rates require a matching filter configuration or explicit resampling. Record timestamps, not file-read speed, and advance protection timers.

2.1. Overcurrent protection

The illustrative settings are pickup currents of 10, 15, 20, and 25 A and corresponding definite delays of 1000, 2000, 3000, and 4000 ms. The input is a non-negative current magnitude, not a signed instantaneous AC sample. These settings demonstrate parameter handling and are not coordinated settings for an actual feeder. Since both pickup and delay increase, the lowest stage would operate first for a sustained current above all other pickups; this example does not demonstrate progressively faster clearance at higher fault currents (Khasawneh et al., 2021).

Each stage requires independent pickup memory and elapsed-time evaluation. Returning from the threshold loop after its first match prevents higher stages from being evaluated (Yeh et al., 2016). Blocking the sample-processing thread with Thread.Sleep also prevents a sequential input source from updating the current during the delay. Listing 1 removes both ambiguities by checking every stage on every timestamped input. Java documents sleep timing as dependent on system timers and scheduling (Shevchenko et al., 2019), it is not a mechanism for guaranteeing a trip deadline.

Listing 1. Independent definite-time overcurrent stages

```
Input: current magnitude I, monotonically increasing timestamp t
Settings: pickup[s], delay[s], for s = 0, 1, 2, 3
State: start[s] = unset; operated[s] = false
for each stage s:
    if I > pickup[s]:
        if start[s] is unset: start[s] = t
        operated[s] = (t - start[s] >= delay[s])
    else:
        start[s] = unset
        operated[s] = false
tripRequest = OR of operated[0..3]
```

The Trip output is a simulated request, not a circuit-breaker operation. A production design additionally needs defined dropout/hysteresis, invalid-input handling, output latching and reset, and coordination with backup functions (Baidoo, 2017). No measured overcurrent response-time series is reported here.

2.2. Transformer differential protection

The transformer example concerns a 110/10 kV, 63 MVA transformer. Its differential and restraint quantities are compared with the characteristic in Figure 2. The software representation follows the earlier transformer differential study by Danylchenko et al. (2023). The numerical settings used in the example are retained; their suitability for a particular transformer cannot be established without the vector group, CT data, tap range, and fault-study results (Serna-Montoya et al., 2025).

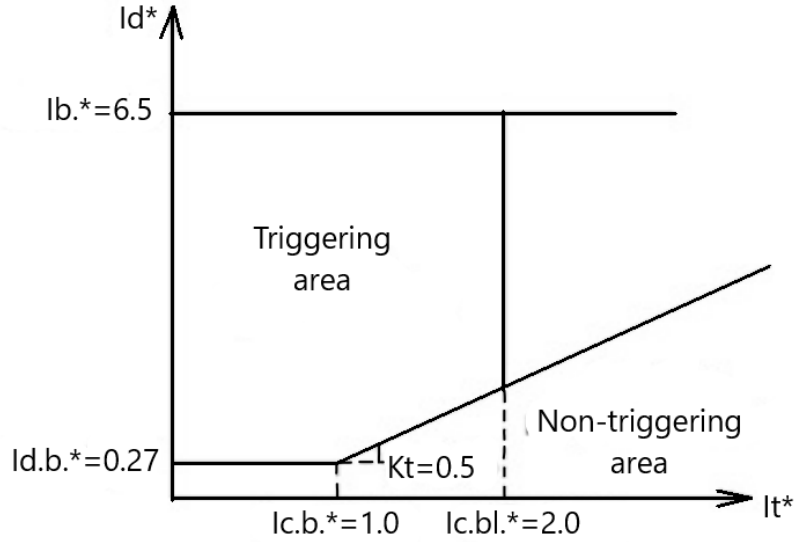


Fig. 2. Differential restraint characteristic and example settings

The Furie class extracts the fundamental and second-harmonic components using a sliding Fourier window. With h denoting harmonic order and $x[k]$ an instantaneous sample, the RMS-normalised sine and cosine components are given by Eqs. (1) and (2). The previous constructor convention $n = 2h$ means $n = 2$ for the fundamental and $n = 4$ for the second harmonic. Adjacent fundamental coefficients differ by $2\pi/80 = 4.5^\circ$. No additional sampling-time factor belongs inside this discrete phase increment.

$$F_{\sin,h} = \frac{\sqrt{2}}{N} \sum_{k=0}^{N-1} x[k] \sin(2\pi hk/N) \quad (1)$$

$$F_{\cos,h} = \frac{\sqrt{2}}{N} \sum_{k=0}^{N-1} x[k] \cos(2\pi hk/N) \quad (2)$$

Here N is the number of samples in a nominal cycle; $h = 1$ or 2 ; and $\sqrt{2}/N$ gives RMS scaling for a sinusoid at the selected frequency. These are harmonic RMS magnitudes, not the total RMS of a distorted waveform. A fixed nominal-frequency Fourier filter is affected by frequency deviation, decaying DC, and CT saturation (Dashtdar et al., 2021). No universal 75% versus 50% accuracy advantage can be inferred merely from an assumed 50% waveform loss.

Listing 2. One Fourier-state update per input sample

```
Input: sample x; harmonic h; window N = 80
State: weighted buffers S[], C[], sums sumS, sumC, index k
s = x * sin(2*pi*h*k/N); c = x * cos(2*pi*h*k/N)
sumS = sumS + s - S[k]; S[k] = s
sumC = sumC + c - C[k]; C[k] = c
Fsin = sqrt(2)/N * sumS; Fcos = sqrt(2)/N * sumC
k = (k + 1) modulo N
RMS = sqrt(Fsin*Fsin + Fcos*Fcos)
angle = atan2(-Fsin, Fcos)
```

The getOrt routine updates both components once for each sample. Magnitude and angle accessors must read the same cached state rather than advance the filter again. A separate state is maintained for each channel and harmonic. Under the convention in Listing 2, the complex phasor is $F_{\cos} - jF_{\sin}$. A full prefault window is needed before using the estimates as settled values.

For the illustrated phase-difference compensation on the HV side, the three compensated phase samples are distinct cyclic combinations:

$$i_{A,HV,c} = \frac{i_{a,HV} - i_{b,HV}}{\sqrt{3}} \quad (3)$$

$$i_{B,HV,c} = \frac{i_{b,HV} - i_{c,HV}}{\sqrt{3}} \quad (4)$$

$$i_{C,HV,c} = \frac{i_{c,HV} - i_{a,HV}}{\sqrt{3}} \quad (5)$$

The compensation direction, polarity, and phase mapping must match the actual transformer vector group and CT connections. Equations (3)–(5) describe the mapping used by the illustrated calc inputs, not a universal compensation rule (Gomes et al., 2015). For each corresponding phase, the compensated HV and LV samples are normalised as follows:

$$i_{vn} = \frac{i_{HV,c}}{I_{bHV}} \quad (6)$$

$$i_{nn} = \frac{i_{LV}}{I_{bLV}} \quad (7)$$

The base currents $I_{bHV} = 3.163$ A and $I_{bLV} = 3.464$ A are the example secondary-current bases. With CT polarities directed into the protected zone, the differential sample is $ivn + inn$. I_d is its fundamental RMS magnitude; $rmsVN$ and $rmsNN$ are the two winding fundamental RMS magnitudes. $I1 = \max(rmsVN, rmsNN)$ and $I2 = \min(rmsVN, rmsNN)$. Their phase-angle difference α is wrapped into $[0^\circ, 360^\circ)$. The phase-dependent restraint in the example is:

$$I_t = \sqrt{I_1 I_2 \cos(180^\circ - \alpha)}, \quad 90^\circ < \alpha < 270^\circ \quad (8)$$

$$I_t = 0, \text{ otherwise} \quad (9)$$

The quantity in Eq. (8) is already calculated from RMS phasors; it must not be passed through another fundamental-frequency RMS filter as though it were an instantaneous sinusoid. A small negative radicand caused by rounding at a boundary can be clipped to zero (Gururajapathy et al., 2017). This restraint law is specific to the example and requires validation through fault CT saturation.

Second-harmonic blocking is based on the differential-signal harmonic ratio:

$$K_{bl} = \frac{I_{d,2}}{I_{d,1}}, \quad I_{d,1} > \varepsilon \quad (10)$$

Here, $I_{d,1} = I_d$, $I_{d,2}$ is the second-harmonic RMS magnitude, and ε is a positive numerical guard selected for the signal scale. The ratio is not evaluated when $I_{d,1} \leq \varepsilon$. The example blocking threshold is $K_{bl,ust} = 0.2$. Harmonic content can also occur during internal faults, so blocking is not a general guarantee of discrimination between all internal and external faults (Shevchenko et al., 2019).

The low-current pickup is $I_{d0} = 0.27$ per unit up to $I_{t0} = 1.0$ per unit. The sloping part shown in Figure 2 is expressed as a pickup threshold, not as the measured differential current:

$$I_{pickup} = I_{d1} + k_t I_t = -0.23 + 0.5 I_t \quad (11)$$

The characteristic is continuous at $I_t = 1.0$. Figure 2 also marks $I_{tbl} = 2.0$ and the high-set level $I_{ots} = 6.5$ per unit. The supplied implementation does not unambiguously define the low-set branch at and above I_{tbl} or the priority of the high-set branch relative to harmonic blocking (Kommey et al., 2022). Accordingly, those policies must be specified before implementation validation; they cannot be inferred from a partially formed conditional statement. Listing 3 exposes them as explicit configuration decisions rather than as silently invented operating rules.

Listing 3. Transformer differential decision structure

```
Update compensated channel and harmonic estimates once per sample
Compute Id, It and guarded harmonic ratio Kbl
pickup = configured characteristic(It), including It >= Itbl
lowSetAllowed = (Id > pickup) AND (Kbl < 0.2)
Update low-set pickup timestamp; reset it if lowSetAllowed is false
lowSetTrip = lowSetAllowed AND (elapsed time >= 30 ms)
highSetTrip = configured high-set rule(Id, Kbl, threshold = 6.5)
tripRequest = lowSetTrip OR highSetTrip
Trip.checkTrip(phaseName, tripRequest)
```

DataComtrade supplies timestamped HV and LV phase samples to three Dzt elements. Main creates these elements and a shared Trip object. Each element owns its timer and writes only its own flag. The 30 ms value is an intentional delay in the low-set example; it is not a measurement of Java execution time.

2.3. Distance protection

Distance protection estimates the apparent impedance of phase-to-phase loops and compares each estimate with a selected operating zone. RS denotes the distance-relay measuring element (impedance relay); the AB, BC, and CA loops correspond to the original phase-A, phase-B, and phase-C element labels. For loop p, $U_p = U_1 - U_2$ and $I_p = I_1 - I_2$ are the corresponding complex voltage and current differences. The notation uses the common complex convention of Listing 2:

$$Z_p = \frac{U_p}{I_p} = R_p + jX_p \quad (12)$$

$$R_p = \frac{U_{px}I_{px} + U_{py}I_{py}}{I_{px}^2 + I_{py}^2} \quad (13)$$

$$X_p = \frac{U_{py}I_{px} - U_{px}I_{py}}{I_{px}^2 + I_{py}^2} \quad (14)$$

Subscripts x and y denote real and imaginary phasor components. A denominator guard is required when the loop current is near zero; an invalid impedance estimate must not be interpreted as an in-zone fault. The sign of X must remain consistent between the Fourier output, Eqs. (12)–(14), and the plotted characteristic. Phase-to-phase loops do not constitute a complete earth-fault distance scheme; residual-current compensation and the appropriate ground loops would be needed (Franklin et al., 2013).

For a polygonal characteristic in the $Z_p = R + jX$ plane, a non-vertical boundary is $X = K_m R + b_m$, where $K_m = (X_{m+1} - X_m)/(R_{m+1} - R_m)$ and $b_m = X_m - K_m R_m$. The four inequalities are combined by AND, with their directions fixed by vertex ordering. A general half-plane form $a_m R + b_m X + c_m \leq 0$ also handles vertical boundaries. The polygon and circular operating regions are illustrated in Figure 5. These are impedance regions, not resistance-only intervals.

For the circular characteristic, $R_0 = (Z_{set}/2)\cos \theta$, $X_0 = (Z_{set}/2)\sin \theta$, and operation requires $(R - R_0)^2 + (X - X_0)^2 \leq (Z_{set}/2)^2$. The example uses $Z_{set} = 76.92 \Omega$ and $\theta = 82^\circ$. Other supplied polygon settings are $R_{lst} = 11.14 \Omega$, $R_{dop} = 38.46 \Omega$, $X_{lst} = 76.11 \Omega$, and boundary angles of 15° and 115° . A reproducible implementation must retain the full ordered vertex list or signed boundary coefficients; the figure alone is insufficient to reconstruct every boundary precisely.

Listing 4. Distance element with one selected operating characteristic

```

Input: timestamped phase voltages and currents for one loop
Update all Fourier states once; form  $U_p$  and  $I_p$ 
if squared current magnitude  $\leq$  configured minimum:
    clear pickup timestamp; set this element flag false; return
Compute  $R$  and  $X$  from Eqs. (13) and (14)
if mode == POLYGON:
    inZone = all configured half-plane inequalities are satisfied
else if mode == CIRCLE:
     $\text{inZone} = (R - R_0)^2 + (X - X_0)^2 \leq (Z_{\text{set}}/2)^2$ 
else:
    reject invalid configuration
if inZone:
    if pickup timestamp is unset: set it to current timestamp
    tripRequest = elapsed time  $\geq$  10 ms
else:
    clear pickup timestamp; tripRequest = false
Trip.checkTrip(loopName, tripRequest)

```

The original OR combination of polygon and circle would form the union of two zones, not a comparison of independently selected characteristics. Listing 4 makes the mode explicit. The three-loop flags are aggregated by OR only after each loop has evaluated its own selected zone and timer. Output latching and breaker logic remain outside this replay model (Barua & Cheon 2014).

The distance records were generated from a transmission-line model in PSCAD and exported as COMTRADE .cfg and .dat files. DataComtrade reconstructs phase voltages and currents and calls calc for the AB, BC, and CA Dz elements. Furie and Trip have the responsibilities shown in Figure 1; duplicating their class definitions is unnecessary.

3. RESULTS AND DISCUSSION

The following figures retain the reported replay examples. They are illustrative records, not a statistical test campaign. The clarified pseudocode and formula corrections in Section 2 require regression replay against the executable source and original COMTRADE data before the figures can be treated as validation of the revised algorithms. In particular, the restraint calculation, high-current policies, Fourier sign convention, and distance-zone selection affect operating decisions.

3.1. Transformer differential protection results

Figures 3 and 4 show the reported phase-A differential-protection states for internal and external faults, as well as transformer energisation. These examples are discussed in the context of the transformer differential implementation developed in Danylchenko et al. (2023); they are not presented as an independent new protection principle.

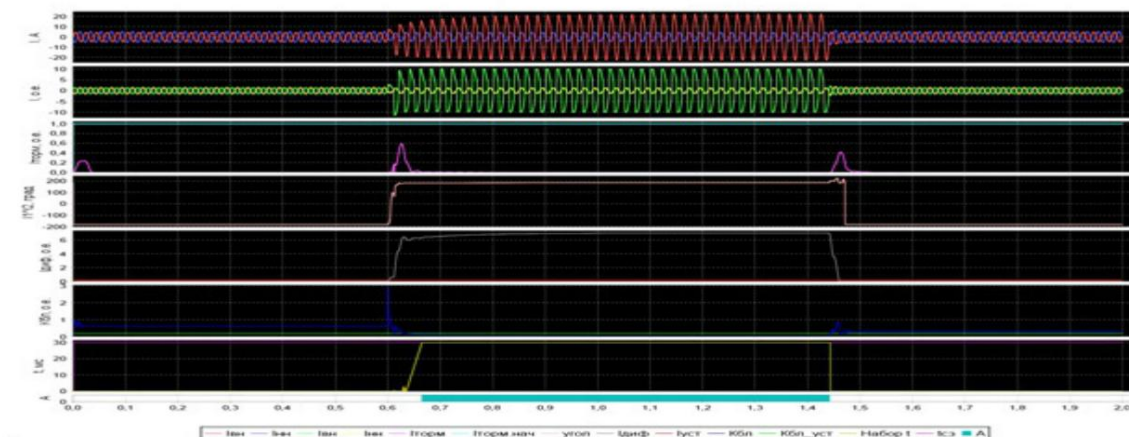
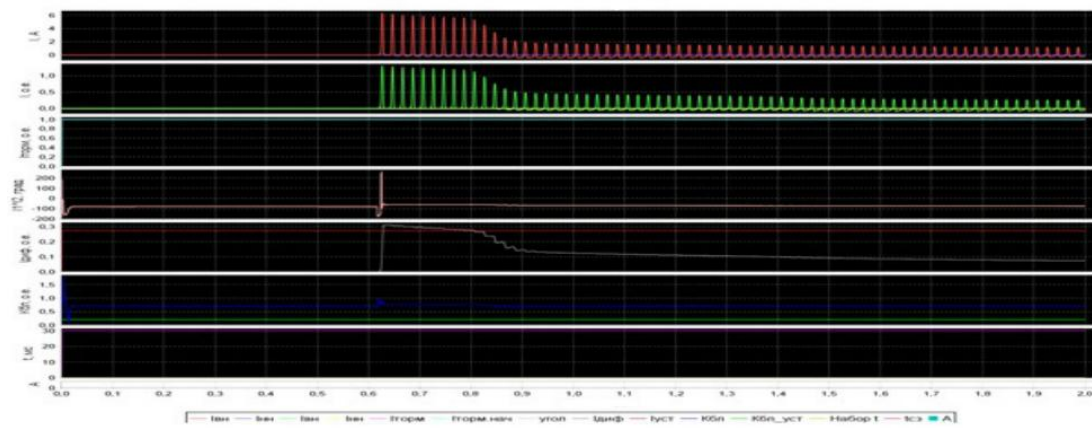
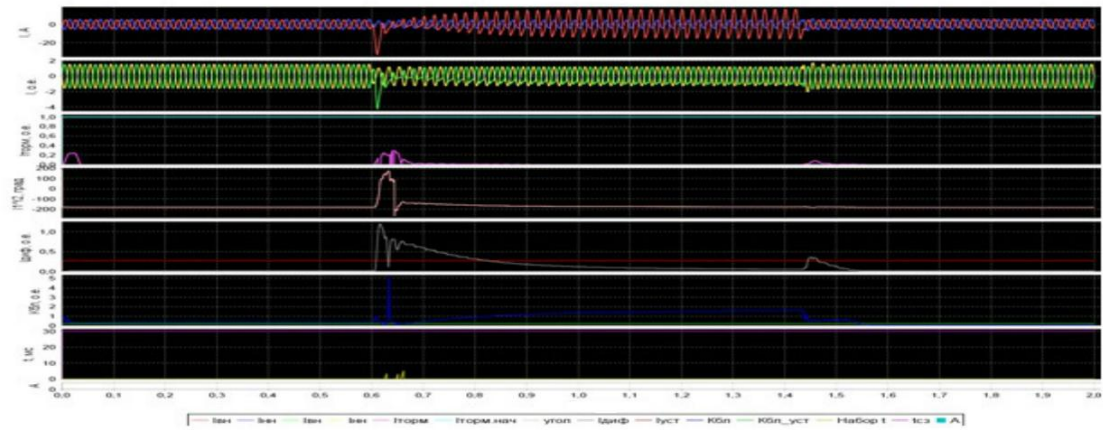


Fig. 3(a). Transformer differential protection for phase A under an internal fault



The internal-fault example shows a trip assertion following fault inception. A precise fault-to-trip interval cannot be established from this raster figure alone; it requires the numerical fault-inception and trip timestamps. The previously stated 0.066 s value is therefore not used as a verified response-time result. The external-fault and energisation examples show that the trip output is blocked. Second-harmonic blocking is consistent with those particular traces, but does not establish security for every external fault or dependability for every internal fault. The figures provide no distribution of timing error or false-trip probability.

3.2. Distance protection results

Figure 5 illustrates the impedance vectors and the polygonal and circular characteristics. The selected characteristic must be evaluated separately for each loop. Figures 6–9 show the reported loop conditions in two fault cases.

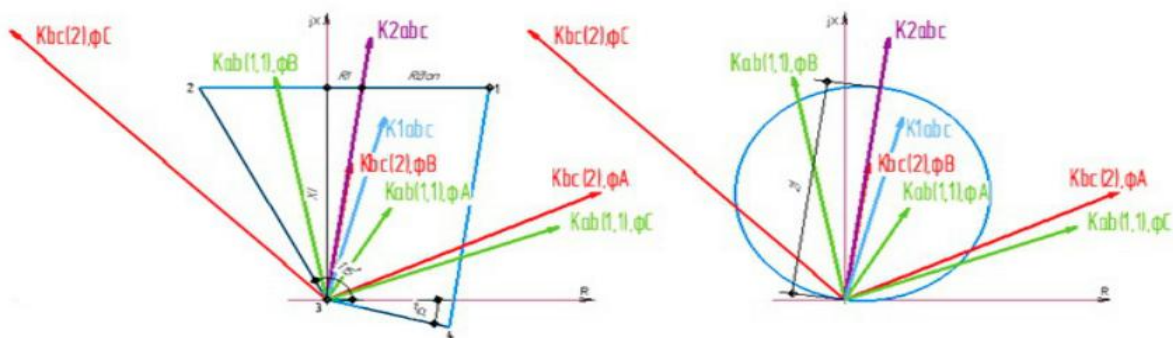


Fig. 5. Impedance vectors and polygonal and circular distance-relay operating regions

Case 1 is a two-phase B–C fault, denoted KBC(2), at 150 km from the relay. The reported loop impedances are $Z_{AB} = 83.12 + j38.84 \Omega$, $Z_{BC} = 7.25 + j49.21 \Omega$, and $Z_{CA} = -95.65 + j96.70 \Omega$. The BC loop is the faulted loop; the other loop measurements should not be read as direct distances to this fault.

The fault begins at $t = 0.5$ s. The reported BC reactance changes from 171.1 Ω to 49.21 Ω . Its estimate satisfies the four polygon inequalities, and the circle criterion in the respective comparisons, and the example applies a 10 ms intentional delay. Fault clearance after a further 0.5 s corresponds to $t = 1.0$ s. The 10 ms setting does not include an independently measured acquisition, filter-transient, scheduling, or output delay.

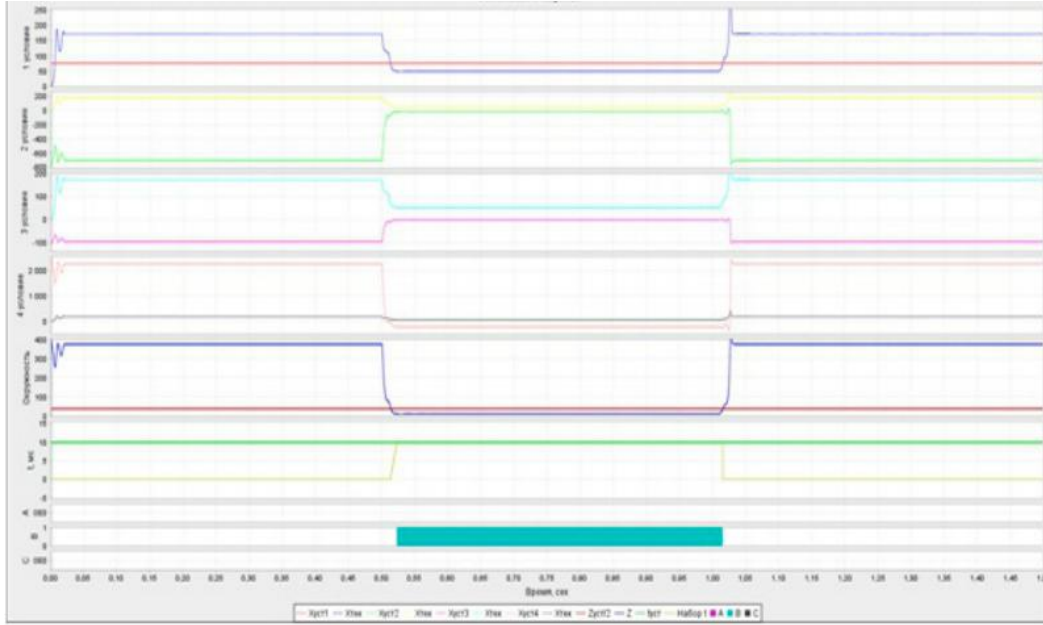


Fig. 6. Distance element states for the BC and AB loops during KBC(2)

For the AB loop, polygon condition 4 is not satisfied, and the circle criterion is not satisfied (Figure 6). For the CA loop, polygon conditions 1 and 2 are not satisfied, and the circle criterion is not satisfied (Figure 7). These loop decisions agree with the reported selectivity of the illustrated B–C fault example.



Fig. 7. Distance element state for the CA loop during KBC(2)

Case 2 is the A–B fault case denoted KAB(1,1), at 100 km with a reported fault-path resistance of 15 Ω . The loop impedances are $Z_{AB} = 19.30 + j33.00 \Omega$, $Z_{BC} = -15.75 + j80.26 \Omega$, and $Z_{CA} = 70.44 + j26.19 \Omega$. The specific grounding and fault-path arrangement associated with this notation must be retained in the model description for reproducibility.

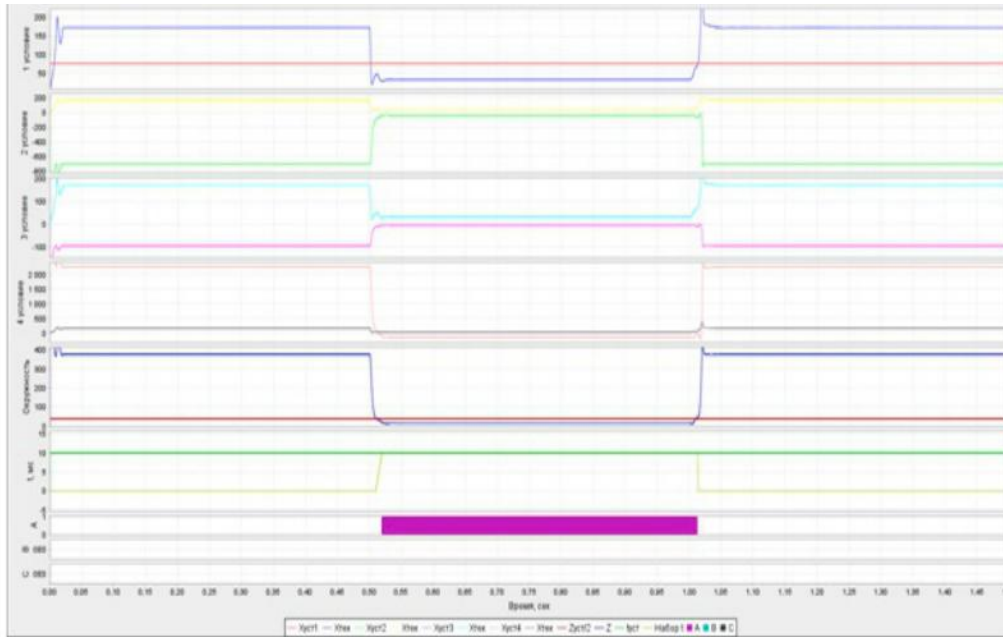


Fig. 8(a). Distance element state for the AB loop during KAB(1,1)

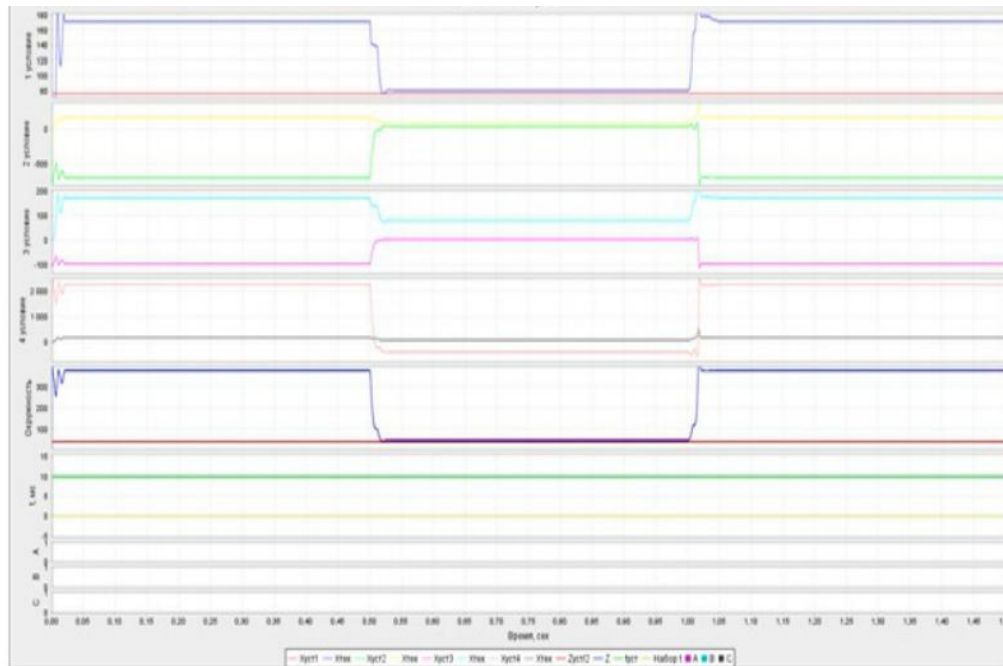


Fig. 8(b). Distance element state for the BC loop during KAB(1,1)

Impedance plane trajectory plot (R-X plot) for the CA phase loop during a two-phase A-B fault with fault-path resistance, illustrating non-operating impedance locus positioning outside zone limits.

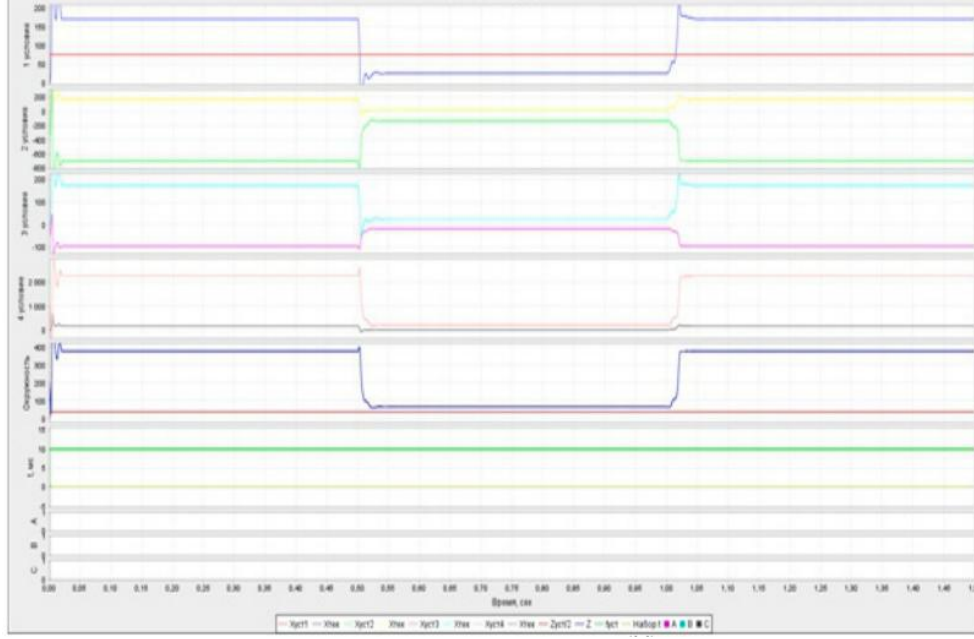


Fig. 9. Distance element state for the CA loop during KAB(1,1)

For the BC loop in Case 2, polygon condition 1 is not satisfied; for the CA loop, condition 4 is not satisfied. Neither loop satisfies the circle criterion (Figs. 8 and 9). These are scenario-specific observations. The two examples do not establish accuracy over all fault locations, fault resistances, source impedances, or DER operating conditions; the latter are known to affect distance-relay behaviour (Garibello-Narváez et al., 2024).

3.3. Timing requirements and available evidence

Table 1 relates the protection requirements to the replay results and measured Java performance. The measurements are summarised in Table 2 and reported for the implementation as a whole, as the supplied ranges are not separated by protection function. The cited IEC documents provide functional evaluation frameworks, but the complete test procedures are not reported here. Intentional pickup delay, per-sample processing time, and end-to-end operating time are distinct quantities and must not be interchanged.

Tab. 1. Protection requirements and evidence from the Java examples

Function	Evaluation requirement	Reported evidence	Remaining verification
Overcurrent	Pickup/dropout and time-delay accuracy; coordination. IEC 60255-151.	Four illustrative pickups: 10–25 A; delays: 1–4 s. No measured timing series.	Independent-stage regression, coordinated settings, timing error and worst-case execution time.
Transformer differential	Internal-fault dependability; external-fault and inrush security; dynamic accuracy. IEC 60255-187-1.	Internal-fault trip and blocking examples; 30 ms low-set delay setting. Fault-to-trip time remains unverified.	Fault-inception timing, CT saturation, vector group, high-restraint/high-set policies, repeated tests.
Distance	Impedance accuracy, loop selection, zone boundaries and operating time. IEC 60255-121.	Two reported fault cases; 10 ms intentional delay; polygon/circle comparisons.	Full zone coordinates, sign and mode regression, boundary sweeps, total fault-to-trip timing.
All functions	Bounded end-to-end delay and correct operation under intended deployment conditions.	Measured Java ranges: ≈ 30 –70 ms operating time; 5–20 μ s per sample; ≈ 30 –80 MB JVM footprint. See Table 2.	Test configuration and timing endpoints; numerical jitter; hardware-in-the-loop and concurrent-event validation.

At 4 kHz, the 80-sample window spans 20 ms. This is an observation interval, not an invariant additional trip delay: threshold crossing depends on the prefault state, waveform, and sliding-window transient. Timestamp-based timers quantise an otherwise fixed pickup interval by less than one sample interval (0.25 ms) when evaluated at the first sample that occurs at or after the deadline. This bound concerns timer discretisation only; it excludes uncertainty in fault inception, filter response, and wall-clock scheduling.

3.4. Measured performance and implementation comparison

Measurements of the Java implementation yielded an operating time of approximately 30–70 ms, a per-sample processing time of 5–20 μ s, and a JVM memory footprint of approximately 30–80 MB (Table 2). Timing variability was characterised qualitatively as moderate, on a millisecond scale, in association with JVM and garbage-collection behaviour. The timing and memory ranges are experimental observations; the determinism and development-effort entries are qualitative assessments. They are not statistical confidence intervals or guaranteed worst-case bounds.

For 80 samples per period at 50 Hz, the sampling interval is 250 μ s. Dividing the measured 5–20 μ s processing range by this interval gives 2–8%, leaving a nominal arithmetic margin within one sampling interval for the measured processing unit. This ratio is not a measure of total CPU utilisation; its interpretation depends on whether the sample represents a single-channel value or a complete multi-channel frame, and it excludes any work outside the timed region. The observed processing range alone therefore does not establish schedulability under concurrent faults, I/O load, or garbage-collection pauses.

Tab. 2. Measured Java performance and comparison ranges for alternative implementations

Metric	Java This work	Analogue relay	C++ on MCU	FPGA
Operating time	$\approx$ 30–70 ms	$\approx$ 10–30 ms	$\approx$ 5–20 ms	<1–5 ms*
Processing time per sample	5–20 μ s	Not applicable	$\approx$ 1–5 μ s	<1 μ s
Timing variability/jitter	Moderate; ms scale (JVM / GC)	Very low	Low	Very low (hardware)
Memory/implementation resources	$\approx$ 30–80 MB (JVM)	Not applicable	KB–MB range	On-chip logic
Reconfigurability/development effort	High/low	Low / not specified	Medium / not specified	Low/high

Notes to Table 2. The Java column reports the authors’ measurements. The other columns reproduce the comparison ranges supplied alongside those measurements; without identified devices, configurations, and matched test conditions, they provide contextual ranges rather than a controlled benchmark. “<1–5 ms” is retained from the supplied FPGA range and denotes implementation-dependent values extending from below 1 ms to approximately 5 ms. Memory in bytes and FPGA logic resources are different quantities and are not directly comparable. Reconfigurability and development effort are qualitative, workflow-dependent assessments.

The measured 30–70 ms operating range lies within the 20–100 ms line-protection reference band used in the comparison. That band is an illustrative application context, not a universal IEC acceptance limit: adequacy must be assessed against the deadline for the particular protection zone. The 30–70 ms range must not be attributed to all three functions, especially in the overcurrent example, which includes intentional delays of 1–4 s. Its timing endpoints and included acquisition, filtering, output, and intentional-delay stages must be specified before it can be interpreted as a total fault-clearing time.

The comparison ranges suggest that the Java implementation trades a larger runtime footprint and less deterministic scheduling for ease of software modification. They do not demonstrate a numerical speedup or a controlled performance ranking. FPGA and SoC protection studies illustrate alternative hardware processing architectures (Li et al., 2025), but the results reported in the literature are not the source of the numerical ranges in Table 2. The present measurements do not quantify precision, false-trip rates, or missed-fault rates.

For a fixed channel count, a sliding Fourier update uses constant work per sample and $O(N)$ buffer storage per channel and harmonic. With C channels, H harmonics, and S timed stages, arithmetic work scales approximately as $O(CH + S)$ per sample and buffer storage as $O(CHN + S)$. These structural estimates describe scaling; the measured 30–80 MB footprint also includes runtime overhead and should not be interpreted as filter-buffer storage alone.

To make the performance measurements reproducible, the test report needs the processor and memory configuration, operating system, JVM version and options, executable revision, channel count, timed code region, warm-up procedure, and number of repetitions. The memory metric must distinguish heap use, allocated heap, and resident process memory. CPU time and elapsed time should be identified separately, and numerical jitter statistics and deadline misses should supplement the current qualitative assessment. The supplied ranges support a feasibility assessment but do not bound worst-case behaviour.

3.5. Error sources and simultaneous faults

Measurement uncertainty arises from CT/VT ratio and phase errors, CT saturation, sample quantisation and synchronisation, frequency drift, harmonics, decaying DC, and finite-precision arithmetic. Distance estimation becomes ill-conditioned as the loop current approaches zero; small errors can also change the decision near a zone boundary. Differential protection is sensitive to imperfect vector-group and ratio compensation. Numerical guards prevent undefined operations but do not quantify physical measurement error. Error margins require comparison against known reference phasors and controlled sweeps around pickup and zone boundaries.

The reported cases concern individual simulated fault scenarios. Multiple faulted phases are not equivalent to independent simultaneous faults on different protected assets. With a fixed set of sampled channels and all elements evaluated on every sample, additional asserted fault flags do not necessarily increase the core arithmetic workload. More channels or assets, event recording, communications, and competing threads can increase processing time and memory pressure. Independent timers and element flags are necessary to prevent one event from resetting or blocking another; sequential replay does not demonstrate adequate response under concurrent live load.

Further validation should include simultaneous and evolving faults, external faults with CT saturation followed by internal faults, energisation with an internal fault, and several active distance loops. Each case should measure correct element selection, false trips, missed trips, fault-to-trip delay, jitter, and deadline misses. The workload and pass criteria must be fixed before comparing implementations.

3.6. Validation scope and reproducibility

The measured Java processing time and memory footprint complement the offline functional examples, but hardware-in-the-loop validation remains necessary to evaluate the acquisition and trip interfaces together. A suitable experiment would apply identical fault waveforms to the Java target and a reference relay, record synchronised fault-inception and output timestamps, and include channel scaling, filtering, output contacts or messages, and imposed processing load. Such testing is not reported here. The current evidence supports software feasibility under the measured conditions, not deployment readiness or a quantified reliability improvement.

Reproducibility requires the complete executable source, a fixed revision, build, and runtime configuration, PSCAD model parameters, COMTRADE records, and full protection settings. The current evidence does not include a verified public repository or a machine-readable dataset sufficient to reproduce the figures and benchmark the corrected algorithms. The low-resolution waveform exports also limit the extraction of precise timing and amplitudes; numerical traces and high-resolution or vector exports are needed for definitive graphical verification.

4. CONCLUSIONS

Overcurrent, transformer differential, and distance protection have been described as Java-based software examples that explicitly separate input processing, measurement, decision logic, and trip-state handling. The overcurrent example requires independent timers and coordinated settings to demonstrate selective multistage operation. The transformer differential examples illustrate internal-fault operation and blocking in selected external-fault and energisation cases, building on the earlier transformer study. The distance examples illustrate loop selectivity and the effect of polygonal and circular operating regions.

The Java measurements yielded approximately 30–70 ms operating time, 5–20 μ s processing time per sample, and a JVM memory footprint of approximately 30–80 MB. The per-sample processing range occupies 2–8% of the nominal 250 μ s sampling interval, supporting computational feasibility for the measured

workload. Moderate timing variability nevertheless prevents a hard real-time guarantee from being inferred from these ranges. Device-specific comparison tests, complete measurement metadata, regression replay of the clarified algorithms, and hardware-in-the-loop evaluation are still required to establish deadline compliance, precision, and reliability under simultaneous faults.

Conflicts of Interest

The authors declare no conflict of interest.

REFERENCES

- Allamanis, M., Barr, E. T., Bird, C., Devanbu, P., Marron, M., & Sutton, C. (2018). Mining semantic loop idioms. *IEEE Transactions on Software Engineering*, 44(7), 651–668. <https://doi.org/10.1109/TSE.2018.2832048>
- Arefin, M., & Khatchadourian, R. (2015). Porting the NetBeans Java 8 enhanced for loop lambda expression refactoring to Eclipse. In *Companion proceedings of the 2015 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity – SPLASH Companion* (pp. 58–59). ACM. <https://doi.org/10.1145/2814189.2817277>
- Baidoo, E. (2017). Fireworks algorithm for unconstrained function optimization problems. *Applied Computer Science*, 13(1), 61–74. <https://doi.org/10.23743/acs-2017-06>
- Bakkar, M., Bogarra, S., Córcoles, F., Aboelhassan, A., Wang, S., & Iglesias, J. (2024). Three-level secure smart protection for ring grid-connected distributed generation. *Energies*, 17(13), Article 3188. <https://doi.org/10.3390/en17133188>
- Bakkar, M., Bogarra, S., Córcoles, F., Iglesias, J., & Al Hanaineh, W. (2022). Multi-layer smart fault protection for secure smart grids. *IEEE Transactions on Smart Grid*, 14(3), 3125–3135. <https://doi.org/10.1109/TSG.2022.3229848>
- Barua, A., & Cheon, Y. (2014). Finding specifications of while statements using patterns. In K. Elleithy & T. Sobh (Eds.), *New trends in networking, computing, E-learning, systems sciences, and engineering* (pp. 581–588). Springer. https://doi.org/10.1007/978-3-319-06764-3_75
- Barua, A., & Cheon, Y. (2015). *A systematic derivation of loop specifications using patterns* (Technical Report No. 15-90). Department of Computer Science, University of Texas at El Paso.
- Danylchenko, D., Tsiupa, V., & Mylko, V. (2023). Digital implementation of relay protection algorithms: Transformer differential protection. In *2023 IEEE 4th KhPI Week on Advanced Technology (KhPIWeek)* (pp. 1–6). IEEE. <https://doi.org/10.1109/KhPIWeek61412.2023.10312912>
- Dashtdar, M., Rubanenko, O., Danylchenko, D., Hosseinimoghadam, S. M. S., Sharma, N. K., & Baiai, M. (2021). Protection of DC microgrids based on differential protection method by fuzzy systems. In *2021 IEEE 2nd KhPI Week on Advanced Technology (KhPIWeek)* (pp. 22–27). IEEE. <https://doi.org/10.1109/KhPIWeek53812.2021.9570079>
- Draganova-Zlateva, I., & Georgiev, D. N. (2021). Digital substations. In *2021 13th Electrical Engineering Faculty Conference (BulEF)* (pp. 1–6). IEEE. <https://doi.org/10.1109/BulEF53491.2021.9690844>
- Franklin, L., Gyori, A., Lahoda, J., & Dig, D. (2013). LambdaFicator: From imperative to functional programming through automated refactoring. In *2013 35th International Conference on Software Engineering (ICSE)* (pp. 1287–1290). IEEE. <https://doi.org/10.1109/ICSE.2013.6606699>
- Gaggero, G. B., Girdinio, P., & Marchese, M. (2025). Cybersecurity issues in electrical protection relays: A systematic review. *Energies*, 18(14), Article 3796. <https://doi.org/10.3390/en18143796>
- Garibello-Narváez, D. R., Gómez-Luna, E., & Vasquez, J. C. (2024). Performance evaluation of distance relay operation in distribution systems with integrated distributed energy resources. *Energies*, 17(18), Article 4735. <https://doi.org/10.3390/en17184735>
- Gomes, J., & Velho, L. (2015). *From Fourier analysis to wavelets*. Springer.
- Gururajapathy, S., Mokhlis, H., & Illias, H. (2017). Fault location and detection techniques in power distribution systems with distributed generation: A review. *Renewable and Sustainable Energy Reviews*, 74, 949–958. <https://doi.org/10.1016/j.rser.2017.03.021>
- Gyori, A., Franklin, L., Dig, D., & Lahoda, J. (2013). Crossing the gap from imperative to functional programming through refactoring. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)* (pp. 543–553). ACM. <https://doi.org/10.1145/2491411.2491461>
- Khasawneh, A., Qawaqzeh, M., Kuchanskyy, V., Rubanenko, O., Miroshnyk, O., Shchur, T., & Drechny, M. (2021). Optimal determination method of the transposition steps of an extra-high voltage power transmission line. *Energies*, 14(20), Article 6791. <https://doi.org/10.3390/en14206791>
- Kommey, B., Addo, E., Tamakloe, E., Tchao, E., & Nunoo-Mensah, H. (2022). A six-port measurement device for high power microwave vector network analysis. *Applied Computer Science*, 18(3), 105–129. <https://doi.org/10.35784/acs-2022-24>
- Li, X., Yao, J., Chen, W., Zhou, W., Zhou, Z., Wang, H., Jiang, Z., Dai, W., & Wang, Z. (2025). Grid-forming converter fault control strategy and its impact on relay protection: Challenges and adaptability analysis. *Energies*, 18(11), Article 2933. <https://doi.org/10.3390/en18112933>
- Miroshnyk, O., Moroz, O., Shchur, T., Chepizhnyi, A., Qawaqzeh, M., & Kocira, S. (2023). Investigation of Smart Grid operation modes with electrical energy storage system. *Energies*, 16(6), Article 2638. <https://doi.org/10.3390/en16062638>
- Qawaqzeh, M. Z., Miroshnyk, O., Shchur, T., Kasner, R., Idzikowski, A., Kruszelnicka, W., Tomporowski, A., Bałdowska-Witos, P., Flizikowski, J., Zawada, M., & Doerffer, K. (2021). Research of emergency modes of wind power plants using computer simulation. *Energies*, 14(16), Article 4780. <https://doi.org/10.3390/en14164780>

- Serna-Montoya, L. F., Saldarriaga-Zuluaga, S. D., López-Lezama, J. M., & Muñoz-Galeano, N. (2025). Optimal microgrid protection coordination for directional overcurrent relays through mixed-integer linear optimization. *Energies*, 18(8), Article 2035. <https://doi.org/10.3390/en18082035>
- Shad, H., Amini Khanavandi, H., Abrisham Foroushan Asl, S., Aranizadeh, A., Vahidi, B., & Mirmozaffari, M. (2025). Optimal protection coordination for grid-connected and islanded microgrids assisted by the Crow Search Algorithm: Application of dual-setting overcurrent relays and fault current limiters. *Energies*, 18(7), Article 1601. <https://doi.org/10.3390/en18071601>
- Shevchenko, S., Danylchenko, D., Kulagin, D., Fedosha, D., & Nitsenko, V. (2019a). Using a phase-differential busbar protection for switchgears of power system facilities. *Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu*, (4), 63–67. <https://doi.org/10.29202/nvngu/2019-4/10>
- Shevchenko, S., Danylchenko, D., Mykhailyshyn, R., & Diahovchenko, I. (2019b). Rogowsky coil applications for power measurement under non-sinusoidal field conditions. *Energetika*, 65(1), 14–20. <https://doi.org/10.6001/energetika.v65i1.3972>
- Tymchuk, S., & Miroshnyk, O. (2015). Assess electricity quality by means of fuzzy generalized index. *Eastern-European Journal of Enterprise Technologies*, 3(4), 26–31. <https://doi.org/10.15587/1729-4061.2015.42484>
- Yeh, C.-C., Chen, C.-S., Ku, T.-T., Lin, C.-H., Hsu, C.-T., Chang, Y.-R., & Lee, Y.-D. (2016). Design of special protection system for an offshore island with high-PV penetration. *IEEE Transactions on Industry Applications*, 53(2), 947–953. <https://doi.org/10.1109/TIA.2016.2627518>