

NIEZAWODNOŚĆ SYSTEMÓW KOMPUTEROWYCH
Laboratorium

Tytuł zadania: **Prognozowanie wskaźników niezawodności z zastosowaniem metody ekspertów**

Wykonał: **Maksymilian Sowula**
Jakub Szczur
Paweł Marek
Kierunek: **Informatyka**

Grupa: **1ID21A**

Kielce, 14.10.2025

1. Opis teoretyczny zagadnienia.

a) Wprowadzenie do niezawodności.

Niezawodność jest to właściwość obiektu technicznego określająca jego zdolność do wykonywania wymaganych funkcji w określonych warunkach i czasie. W praktyce niezawodność wyraża się przez zestaw wskaźników ilościowych, takich jak:

- Prawdopodobieństwo bezawaryjnej pracy ($R(t)$)
- Średni czas bezawaryjnej pracy (MTBF – Mean Time Between Failures)
- Intensywność uszkodzeń ($\lambda(t)$)
- Średni czas naprawy (MTTR – Mean Time To Repair)

W analizie niezawodności istotne jest nie tylko określenie bieżących wartości tych wskaźników, ale również prognozowanie ich zmian w czasie, szczególnie w sytuacji braku pełnych danych empirycznych.

b) Istota prognozowania niezawodności

Prognozowanie niezawodności polega na szacowaniu przyszłych wartości wskaźników niezawodności na podstawie:

- danych z testów i eksploatacji,
- modeli matematycznych niezawodności,
- opinii i ocen ekspertów (metoda ekspertów).

W praktyce często brak jest wystarczających danych do statystycznego modelowania niezawodności, zwłaszcza na etapie projektowania nowego urządzenia lub systemu. W takich przypadkach wykorzystuje się metody heurystyczne, a w szczególności metodę ekspertów.

c) Metoda ekspertów - istota i zastosowanie

Metoda ekspertów (ang. expert judgment method) jest jedną z najczęściej stosowanych metod heurystycznych wykorzystywanych w analizie i prognozowaniu niezawodności urządzeń technicznych, szczególnie w sytuacjach, gdy brak jest danych statystycznych z badań eksploatacyjnych lub testów trwałości. Jej istota polega na wykorzystaniu wiedzy, doświadczenia zawodowego oraz intuicji specjalistów – ekspertów, którzy na podstawie swojej praktyki dokonują oceny i prognozy wartości wskaźników niezawodnościowych.

W metodzie tej zakłada się, że doświadczeni konstruktorzy, projektanci i użytkownicy urządzeń posiadają wiedzę pozwalającą przewidzieć, jak zmienią się parametry niezawodności w nowo projektowanym systemie w porównaniu do urządzeń już istniejących i eksploatowanych. Opinie ekspertów stanowią zatem podstawę do określenia parametrów prognostycznych.

Dane prognostyczne uzyskuje się poprzez analizę urządzenia podobnego do projektowanego. W praktyce oznacza to wykorzystanie znanych wartości charakterystycznych dla systemu referencyjnego, takich jak:

- **Parametr strumienia uszkodzeń (A)** – opisującyczęstość występowania uszkodzeń, dla którego wyznacza się wartość średnią ($E^*(A) = \bar{A}$) lub kres gorny (A_g),
- **Czas poprawnej pracy (T)** między kolejnymi uszkodzeniami – dla którego wyznacza się wartość oczekiwana ($E^*(T) = \bar{t}$).

Na podstawie tych danych wyznacza się prognozowane wartości dla projektowanego urządzenia:

- **Wymaganą (dopuszczalną) wartość strumienia uszkodzeń (A_0), lub**

- **Wymaganą wartość oczekiwana czasu poprawnej pracy ($E(T_0)$).**

Metoda ekspertów pozwala więc przekształcić doświadczenie i subiektywne oceny specjalistów w mierzalne parametry matematyczne. Dzięki temu możliwe jest ilościowe oszacowanie niezawodności urządzenia już na etapie projektowania, bez konieczności długotrwałych badań eksploatacyjnych.

Zaletą tej metody jest jej elastyczność i uniwersalność, gdyż można ją stosować do różnych typów systemów technicznych, zarówno nowych, jak i modernizowanych. Ponadto umożliwia ona uwzględnienie postępu technologicznego, zmian materiałowych czy modyfikacji konstrukcyjnych, które mogą wpływać na niezawodność.

W rezultacie metoda ekspertów stanowi praktyczne narzędzie wspomagające proces decyzyjny w inżynierii niezawodności, pozwalając na określenie dopuszczalnych wartości wskaźników niezawodności i prognozowanie ich zmian w czasie, gdy brak jest twardych danych empirycznych.

Ustalanie wymagania dla parametru strumienia uszkodzeń

Parametr strumienia uszkodzeń (A) określa intensywność występowania awarii w czasie i stanowi jeden z podstawowych wskaźników niezawodności urządzeń technicznych. Dla urządzenia projektowanego ustala się wymaganą (dopuszczalną) wartość (A_0), która nie powinna przekroczyć wartości uzyskanych dla urządzeń podobnych.

Wymagana wartość (A_0) wyznaczana jest na podstawie wzoru:

$$A_0 \leq \bar{A} * \alpha_e * \alpha_k \text{ lub } A_0 \leq A_g * \alpha_e * \alpha_k$$

gdzie:

- α_e – współczynnik uwzględniający możliwe podwyższenie niezawodności elementów,
- α_k – współczynnik uwzględniający możliwe podwyższenie niezawodności konstrukcji mechanicznej urządzenia.

Współczynniki α_e i α_k określa się metodą ekspertów na podstawie ich przewidywań dotyczących wpływu modernizacji, zmian materiałowych i postępu technologicznego na

niezawodność poszczególnych elementów oraz całej konstrukcji.

Współczynnik α_k uwzględnia przewidywany wzrost niezawodności konstrukcji mechanicznej urządzenia. Jego wartość wyznacza się ze wzoru:

$$\alpha_k = \frac{1}{Km}, \quad Km = \frac{1}{J} * \sum_j Km_j$$

gdzie:

- Km_j – ocena wzrostu niezawodności konstrukcji określona przez j-tego eksperta (przyjmuje wartości od 1 do 2),
- J – liczba ankietowanych ekspertów.

Wartość ($Km = 1$) oznacza brak zmian w niezawodności, natomiast wartości większe od 1 wskazują na jej wzrost. Współczynnik α_k , jako odwrotność Km , maleje wraz ze wzrostem przewidywanej poprawy niezawodności konstrukcji.

Współczynnik α_e

Współczynnik α_e odnosi się do wzrostu niezawodności elementów elektronicznych i uwzględnia udział poszczególnych grup elementów w urządzeniu. Oblicza się go na podstawie wzorów:

$$\alpha_e = \frac{1}{J} * \sum_{g=1}^G Kg * NgN,$$

gdzie:

- Ng/N – udział elementów danej grupy w urządzeniu,
- Kg – współczynnik wzrostu niezawodności elementów w grupie (g),
- G – liczba grup elementów (np. elementy dyskretne, układy scalone, elementy elektromechaniczne, elementy mechaniczne).

Wartość ($Kg > 1$) oznacza poprawę niezawodności danej grupy, natomiast ($Kg = 1$) – brak zmian. Średnia arytmetyczna ocen ekspertów pozwala uzyskać wartość uogólnioną, wykorzystywaną w dalszych obliczeniach niezawodnościowych.

d) Prognozowanie intensywności uszkodzeń elementów

W przypadku nowych lub modernizowanych urządzeń często brak jest danych empirycznych dotyczących intensywności uszkodzeń ich elementów. Wówczas prognozuje się je metodą ekspertów, przyjmując, że nowo opracowane elementy nie będą mniej niezawodne niż te stosowane wcześniej. Prognozowana intensywność uszkodzeń elementów określana jest zależnością:

$$R = (\lambda_{di} : \lambda_{di} \leq \lambda_p)$$

gdzie:

- λ_p – intensywność uszkodzeń elementów podobnych,
- $(\alpha_i = \frac{1}{k_i})$ – współczynnik uwzględniający przewidywany wzrost niezawodności danego elementu.

Współczynnik (k_i) wyraża wielokrotność poprawy niezawodności w stosunku do dotychczasowych rozwiązań. Dla elementów o podwyższonej jakości ($k_i > 1$), a zatem ($\alpha_i < 1$), co skutkuje mniejszą prognozowaną intensywnością uszkodzeń (λ_{di}). Tym samym metoda pozwala przewidzieć parametry niezawodności nawet w przypadku elementów, które nie były jeszcze testowane w warunkach eksploatacyjnych. Metodę tę stosuje się szczególnie w sytuacjach, gdy:

- brak jest danych statystycznych o awaryjności urządzenia,
- projektowane urządzenie jest nowe lub modernizowane,
- warunki pracy mają ulec istotnym zmianom,
- istnieją trudności w przeprowadzeniu badań doświadczalnych.

W takich przypadkach metoda ekspertów stanowi jedyne praktyczne narzędzie umożliwiające racjonalne oszacowanie wymagań niezawodnościowych i pozwala już na etapie projektowania określić dopuszczalny poziom intensywności uszkodzeń.

d) Etapy prognozowania metodą ekspertów

Proces prognozowania wskaźników niezawodności z wykorzystaniem metody ekspertów przebiega w kilku uporządkowanych etapach:

Sformułowanie problemu

Na początku określa się cel prognozy, analizowane wskaźniki niezawodności (np. MTBF, R(t), λ) oraz zakres systemu objętego oceną.

Dobór ekspertów

Wybiera się grupę 5–10 specjalistów posiadających wiedzę techniczną i doświadczenie w dziedzinie dotyczącej analizowanego systemu. Eksperci powinni reprezentować różne obszary – konstrukcyjny, elektroniczny i eksploatacyjny – by zapewnić szerokie spojrzenie na problem.

Przygotowanie ankiet i kryteriów oceny

Opracowuje się zestaw pytań lub formularzy zawierających prośby o ocenę przewidywanych wartości parametrów niezawodności lub czynników wpływających na ich zmianę.

Zebranie ocen ekspertów

Każdy ekspert niezależnie określa swoje prognozy dotyczące wartości poszczególnych wskaźników (np. intensywności uszkodzeń, czasu poprawnej pracy, wzrostu niezawodności konstrukcji).

Analiza i agregacja wyników

Zebrane dane poddaje się analizie – oblicza się średnie, mediany lub wartości ważone (zależnie od kompetencji ekspertów). Dzięki temu uzyskuje się syntetyczną ocenę grupową.

Wyznaczenie wartości prognozowanych wskaźników

Na podstawie uśrednionych danych oblicza się prognozowane wartości parametrów niezawodności, takich jak (A_0), ($E(T_0)$) czy (λ_{di}).

Ocena zgodności opinii ekspertów

Sprawdza się spójność i zgodność odpowiedzi, np. za pomocą współczynnika zgodności Kendalla (W). Wysoka zgodność świadczy o wiarygodności uzyskanych wyników.

2. Algorytm działania programu

Działanie aplikacji opiera się na zbieraniu opinii mechaników i elektroników, ich walidacji oraz przeliczeniu na współczynniki korygujące wykorzystywane w obliczeniach niezawodności.

2.1 Opis ogólny algorytmu

1. **Konfiguracja programu** – użytkownik wybiera tryb obliczeń (A_0 , $E(T_0)$ lub λ_{di}), liczbę ekspertów oraz dane prognostyczne.
2. **Zbieranie danych ekspertów** – mechanicy wprowadzają współczynniki K_mj , a elektronicy określają wartości (N_g/N) i K_g dla grup elementów.
3. **Obliczenia współczynników** – program wyznacza:
 - α_k – współczynnik konstrukcyjny (mechaniczny)
 - α_e – współczynnik elementowy (elektroniczny)
 - K_e – współczynnik eksploatacyjny

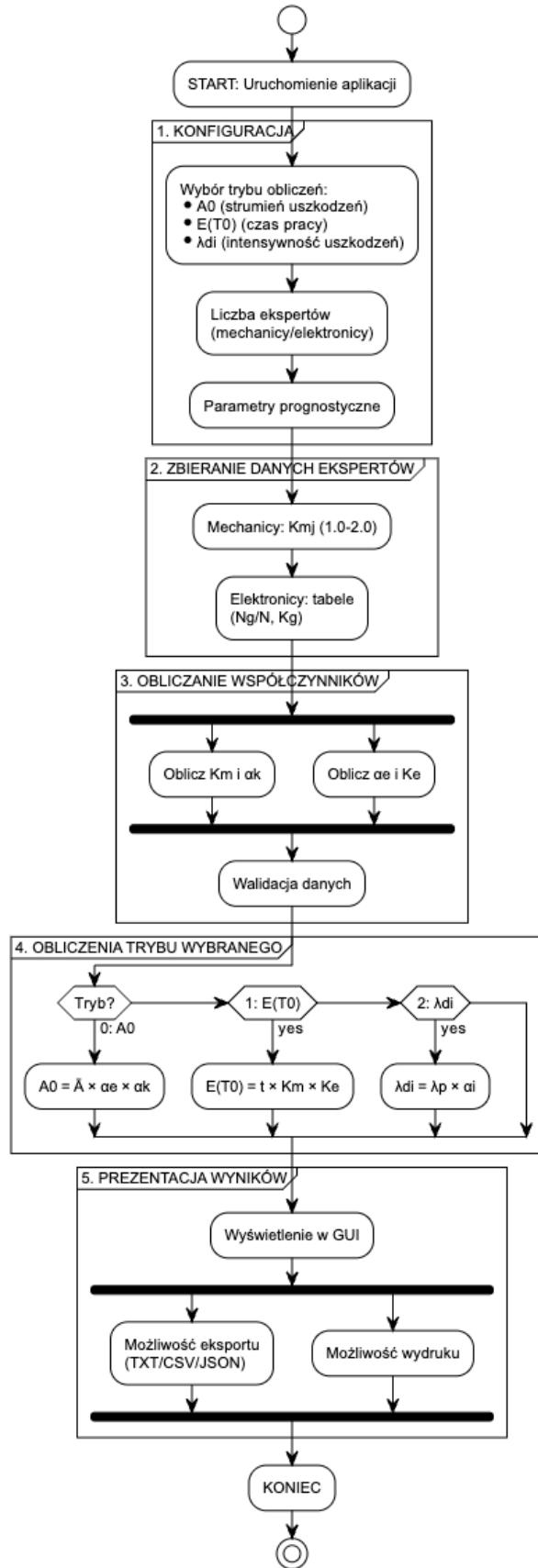
4. Obliczenia wskaźników niezawodności – w zależności od trybu:

- $A_0 = \bar{A} \times \alpha_e \times \alpha_k$
- $E(T_0) = E^*(T) \times K_m \times K_e$
- $\lambda_{di} = \lambda_p \times \alpha_i$

5. Prezentacja wyników – wartości prezentowane są w GUI, z możliwością eksportu i wydruku.

6. Walidacja i obsługa błędów – dane wejściowe są sprawdzane pod kątem poprawności zakresów i spójności sum.

2.2 Schemat blokowy algorytmu



Rysunek 2.1. Schemat blokowy algorytmu programu.

3. Kod programu

3.1 Architektura projektu

Program został zbudowany w architekturze MVC (Model-View-Controller) z wykorzystaniem PyQt5 do interfejsu graficznego.

3.1.1 Struktura katalogów

ReliabilityExpertMethod/

```
|  
|   └── src/          # Kod źródłowy  
|       ├── main.py      # Punkt wejścia aplikacji  
|       ├── calculations/    # Moduły obliczeniowe (Model)  
|           ├── alpha_coefficients.py  
|           ├── expert_weights.py  
|           └── reliability_indicators.py  
|       ├── gui/         # Interfejs graficzny (View)  
|           ├── main_window.py  
|           ├── config_tab.py  
|           ├── experts_tab.py  
|           ├── results_tab.py  
|           ├── calculator.py    # Kontroler obliczeń  
|           ├── visualization.py  
|           ├── export/        # Eksport wyników  
|               ├── export_manager.py  
|               ├── txt_exporter.py  
|               ├── csv_exporter.py  
|               └── json_exporter.py  
|           ├── print/        # Drukowanie  
|               └── print_manager.py  
|           └── widgets/      # Komponenty GUI  
|               ├── export_button_panel.py  
|               └── results_text_widget.py
```

```
|   └── utils/          # Narzędzia pomocnicze
|       ├── data_generator.py
|       ├── data_validation.py
|       └── export_manager.py
|
|   └── tests/          # Testy jednostkowe
|       ├── test_calculations.py
|       ├── test_examples.py
|       └── test_validation.py
|
|   └── docs/           # Dokumentacja
|   └── build/          # Pliki budowania
|   └── requirements.txt      # Zależności
└── README.md        # Dokumentacja główna
```

3.2 Moduły obliczeniowe (Model)

3.2.1 `calculations/alpha_coefficients.py`

Moduł odpowiedzialny za obliczanie współczynników α (konstrukcyjnego i elementowego).

```
def calculate_alpha_k(kmj_values):
    """
    Oblicza współczynnik konstrukcyjny  $\alpha_k$  na podstawie ocen mechaników.

    Args:
        kmj_values (list): Lista wartości  $K_{mj}$  od mechaników (zakres: 1.0-2.0)

    Returns:
        tuple: ( $K_m$ ,  $\alpha_k$ ) - średnia mechaniczna i współczynnik konstrukcyjny

    Raises:
        ValueError: Gdy lista  $K_{mj}$  jest pusta

    Przykład:
        >>> calculate_alpha_k([1.2, 1.5, 1.3])
        (1.333, 0.750)
    """
    if not kmj_values:
        raise ValueError("Lista  $K_{mj}$  nie może być pusta")

    J = len(kmj_values)
    Km = sum(kmj_values) / J # Średnia arytmetyczna
    alpha_k = 1.0 / Km # Odwrotność średniej

    return Km, alpha_k

def calculate_alpha_e(expert_data):
    """
    Oblicza współczynnik elementowy  $\alpha_e$  na podstawie danych elektroników.

    Args:
        expert_data (list): Lista list krotek (ng_n, kg) dla każdego eksperta
            ng_n - udział grupy elementów (0-1)
            kg - współczynnik jakości grupy (1-10)
```

```

Returns:
    tuple: (alpha_ej_list, alpha_e) - lista współczynników dla ekspertów
        i średni współczynnik elementowy

Raises:
    ValueError: Gdy dane są puste lub Kg ≤ 0

Przykład:
>>> expert_data = [[(0.5, 10), (0.3, 3), (0.15, 4), (0.05, 2)]]
>>> calculate_alpha_e(expert_data)
([0.2125], 0.2125)
"""
if not expert_data:
    raise ValueError("Dane ekspertów nie mogą być puste")

J = len(expert_data)
alpha_ej_list = []

# Dla każdego eksperta
for expert_groups in expert_data:
    alpha_ej = 0.0
    # Dla każdej grupy elementów
    for ng_n, kg in expert_groups:
        if kg <= 0:
            raise ValueError(f"Kg musi być większe od 0, otrzymano: {kg}")
        alpha_ej += (ng_n / kg) # Suma (Ng/N) / Kg
    alpha_ej_list.append(alpha_ej)

# Średnia ze wszystkich ekspertów
alpha_e = sum(alpha_ej_list) / J

return alpha_ej_list, alpha_e

def calculate_alpha_i(ki):
"""
Oblicza współczynnik αi dla intensywności uszkodzeń.

Args:
    ki (float): Współczynnik Ki (> 0)

Returns:
    float: Współczynnik αi = 1/Ki

Raises:
    ValueError: Gdy ki ≤ 0
"""
if ki <= 0:
    raise ValueError("ki musi być większe od 0")

return 1.0 / ki

```

Listing 3.1. Kod obliczania współczynników α .

3.2.2 `calculations/expert_weights.py`

Moduł do obliczania współczynników eksploatacyjnych i wag modułów.

```
def calculate_Ke(expert_data):
    """
    Oblicza współczynnik eksploatacyjny Ke.

    Args:
        expert_data (list): Lista list krotek (ng_n, kg) dla każdego eksperta

    Returns:
        tuple: (ke_j_list, ke) - lista współczynników dla ekspertów
               i średni współczynnik Ke

    Wzór:
        Kej = Σ[(Ng/N) × Kg] dla wszystkich grup
        Ke = średnia(Kej) dla wszystkich ekspertów
    """
    if not expert_data:
        raise ValueError("Dane ekspertów nie mogą być puste")

    J = len(expert_data)
    ke_j_list = []

    for expert_groups in expert_data:
        ke_j = 0.0
        for ng_n, kg_j in expert_groups:
            ke_j += (ng_n * kg_j) # Suma (Ng/N) × Kg
        ke_j_list.append(ke_j)

    ke = sum(ke_j_list) / J # Średnia

    return ke_j_list, ke

def calculate_module_weights(expert_damage_counts, num_modules):
    """
    Oblicza wagi modułów na podstawie ankiet ekspertów (metoda ankietowa).

    Args:
        expert_damage_counts (list): Macierz [ekspert][moduł] z liczbą
                                     uszkodzeń
        num_modules (int): Liczba modułów
    Returns:
        list: Lista wag dla każdego modułu (suma = num_modules)

    Wzór:
        mj = średnia liczba uszkodzeń modułu j ze wszystkich ankiet
        waga_j = (mj × N) / Σ(mj)
    """

```

```

if not expert_damage_counts:
    raise ValueError("Dane ankiet nie mogą być puste")

K = len(expert_damage_counts) # Liczba ekspertów

N = num_modules

mj_list = []
# Dla każdego modułu oblicz średnią liczbę uszkodzeń
for j in range(N):
    mjk_sum = sum(expert_damage_counts[k][j] for k in range(K))
    mj = mjk_sum / K
    mj_list.append(mj)

m = sum(mj_list) # Suma wszystkich średnich

if m == 0:
    raise ValueError("Suma uszkodzeń nie może być zero")

# Normalizacja wag
module_weights = [(mj * N) / m for mj in mj_list]

return module_weights

def normalize_weights(weights):
    """
    Normalizuje wagi tak, aby suma wynosiła 1.0.
    Args:
        weights (list lub dict): Wagi do normalizacji
    Returns:
        list lub dict: Znormalizowane wagi
    """
    if isinstance(weights, dict):
        total = sum(weights.values())
        if total == 0:
            raise ValueError("Suma wag nie może być zero")
        return {key: val / total for key, val in weights.items()}
    else:
        total = sum(weights)
        if total == 0:
            raise ValueError("Suma wag nie może być zero")
        return [w / total for w in weights]

```

Listing 3.2. Kod obliczania współczynników eksploatacyjnych i wag modułów.

3.2.3 `calculations/reliability_indicators.py`

Moduł obliczający końcowe wskaźniki niezawodności.

```
def calculate_A0(A_mean, alpha_e, alpha_k):
    """
    Oblicza średni strumień uszkodzeń A0.

    Args:
        A_mean (float): Średni strumień uszkodzeń urządzenia wzorcowego [h-1]
        alpha_e (float): Współczynnik elementowy
        alpha_k (float): Współczynnik konstrukcyjny

    Returns:
        float: Górna granica A0 ≤  $\bar{A} \times \alpha_e \times \alpha_k$ 

    Przykład:
        >>> calculate_A0(1.1e-5, 0.46, 0.83)
        4.207e-06
    """
    return A_mean * alpha_e * alpha_k

def calculate_E_T0(E_T, K_m, K_e):
    """
    Oblicza oczekiwany czas poprawnej pracy E(T0).

    Args:
        E_T (float): Oczekiwany czas pracy urządzenia wzorcowego [h]
        K_m (float): Średni współczynnik mechaniczny
        K_e (float): Współczynnik eksplotacyjny

    Returns:
        float: Dolna granica E(T0) ≥  $E^*(T) \times K_m \times K_e$ 

    Przykład:
        >>> calculate_E_T0(900, 1.2, 3.673)
        3967.56
    """
    return E_T * K_m * K_e
def calculate_lambda_di(lambda_p, alpha_i):
    """
    Oblicza intensywność uszkodzeń nowych elementów λdi.

    Args:
        lambda_p (float): Intensywność uszkodzeń elementu porównywalnego [h-1]
        alpha_i (float): Współczynnik korygujący

    Returns:
        float: λdi = λp × αi
```

```
Przykład:  
>>> calculate_lambda_di(352.8e-8, 0.17)  
5.998e-08  
***  
return lambda_p * alpha_i  
  
def compute_reliability_indicators(A_mean, E_T, lambda_p,  
                                   alpha_e, alpha_k, K_m, K_e, alpha_i):  
    ***  
    Oblicza wszystkie wskaźniki niezawodności jednocześnie.  
  
Args:  
    A_mean, E_T, lambda_p: Dane progностyczne  
    alpha_e, alpha_k, K_m, K_e, alpha_i: Współczynniki  
  
Returns:  
    tuple: (A0, E_T0, lambda_di)  
***  
A0 = calculate_A0(A_mean, alpha_e, alpha_k)  
E_T0 = calculate_E_T0(E_T, K_m, K_e)  
lambda_di = calculate_lambda_di(lambda_p, alpha_i)  
  
return A0, E_T0, lambda_di
```

Listing 3.3. Kod obliczania końcowych wskaźników niezawodności.

3.3 Kontroler (Calculator)

3.3.1 `gui/calculator.py`

Główny kontroler zarządzający przepływem obliczeń.

```
class ReliabilityCalculator:  
    """  
        Kontroler główny zarządzający obliczeniami niezawodności.  
        Koordynuje wywołania modułów obliczeniowych i formuluje wyniki.  
    """  
  
    def __init__(self):  
        pass  
  
    def calculate(self, config, kmj_values, expert_data):  
        """  
            Główna metoda obliczeniowa.  
            Args:  
                config (dict): Konfiguracja z ConfigurationTab  
                kmj_values (list): Wartości Kmj od mechaników  
                expert_data (list): Dane od elektroników  
            Returns:  
                tuple: (results_text, results_data)  
                    - results_text: Sformatowany tekst wyników  
                    - results_data: Słownik z danymi wyników  
            Workflow:  
                1. Oblicz współczynniki podstawowe ( $\alpha_k$ ,  $\alpha_e$ ,  $K_e$ )  
                2. Wybierz tryb obliczeń i oblicz wskaźnik docelowy  
                3. Sformatuj wyniki tekstowe  
                4. Przygotuj dane strukturalne do eksportu  
        """  
        # Krok 1: Obliczenia podstawowe  
        km, alpha_k = calculate_alpha_k(kmj_values)  
        alpha_ej_list, alpha_e = calculate_alpha_e(expert_data)  
        ke_j_list, ke = calculate_Ke(expert_data)  
        # Krok 2: Budowanie tekstu wyników  
        results_text = self._build_results_text(  
            km, alpha_k, alpha_e, alpha_ej_list, ke, ke_j_list  
        )  
        # Krok 3: Obliczenia specyficzne dla trybu  
        mode = config['mode']  
        mode_results, mode_data = self._calculate_mode_specific(  
            mode, config, km, ke, alpha_e, alpha_k  
        )  
        results_text += mode_results
```

```

results_text += "\n" + "=" * 60 + "\n"

# Krok 4: Przygotowanie danych strukturalnych
results_data = {
    'config': config,
    'kmj_values': kmj_values,
    'expert_data': expert_data,
    'km': km,
    'alpha_k': alpha_k,
    'alpha_e': alpha_e,
    'alpha_ej_list': alpha_ej_list,
    'ke': ke,
    'ke_j_list': ke_j_list,
    'mode': mode,
    'mode_specific': mode_data
}

return results_text, results_data

def _build_results_text(self, km, alpha_k, alpha_e,
                       alpha_ej_list, ke, ke_j_list):
"""
Formatuje wyniki jako tekst do wyświetlenia.
"""

results = "=" * 60 + "\n"
results += "WYNIKI OBLICZEŃ\n"
results += "=" * 60 + "\n\n"

# Sekcja 1: Współczynniki mechaniczne
results += f"1. WSPÓŁCZYNNIKI MECHANICZNE\n"
results += f"    Km = {km:.4f}\n"
results += f"    αk = {alpha_k:.4f}\n\n"

# Sekcja 2: Współczynniki elektroniczne
results += f"2. WSPÓŁCZYNNIKI ELEKTRONICZNE\n"
results += f"    αe = {alpha_e:.4f}\n"
results += f"    Współczynniki ekspertów:\n"
for i, alpha_ej in enumerate(alpha_ej_list, 1):
    results += f"        αe{i} = {alpha_ej:.4f}\n\n"

# Sekcja 3: Współczynniki eksploatacyjne
results += f"\n3. WSPÓŁCZYNNIKI EKSPOLOATACYJNE\n"
results += f"    Ke = {ke:.4f}\n"
results += f"    Współczynniki ekspertów:\n"
for i, ke_j in enumerate(ke_j_list, 1):
    results += f"        Ke{i} = {ke_j:.4f}\n\n"

return results

def _calculate_mode_specific(self, mode, config, km, ke, alpha_e,
                           alpha_k):

```

```

"""
Wywołuje odpowiednie obliczenia w zależności od trybu.

Tryby:
    0 - Obliczanie A0
    1 - Obliczanie E(T0)
    2 - Obliczanie λdi
"""

if mode == 0:
    return self._calculate_a0(config, alpha_e, alpha_k)
elif mode == 1:
    return self._calculate_e_t0(config, km, ke)
elif mode == 2:
    return self._calculate_lambda_di(config, alpha_e)
else:
    return "", {}

def _calculate_a0(self, config, alpha_e, alpha_k):
    """Oblicza A0 (tryb 0)"""
    a_mean_str = config['a_mean']
    if a_mean_str:
        a_mean = float(a_mean_str)
        a0 = calculate_A0(a_mean, alpha_e, alpha_k)
        text = (f"\n4. ŚREDNI STRUMIEŃ USZKODZEŃ\n"
                f"    Ā = {a_mean:.6e}\n"
                f"    A0 ≤ {a0:.6e}\n")
        data = {'a_mean': a_mean, 'a0': a0}
        return text, data
    return f"\n4. BRAK DANYCH dla Ā\n", {}

def _calculate_e_t0(self, config, km, ke):
    """Oblicza E(T0) (tryb 1)"""
    t_str = config['t_expected']
    if t_str:
        t_expected = float(t_str)
        e_t0 = calculate_E_T0(t_expected, km, ke)
        text = (f"\n4. OCZEKIWANY CZAS PRACY\n"
                f"    E*(T) = {t_expected:.2f} h\n"
                f"    E(T0) ≥ {e_t0:.2f} h\n")
        data = {'t_expected': t_expected, 'e_t0': e_t0}
        return text, data
    return f"\n4. BRAK DANYCH dla t\n", {}

def _calculate_lambda_di(self, config, alpha_e):
    """Oblicza λdi (tryb 2)"""
    lambda_p_str = config['lambda_p']
    if lambda_p_str:
        lambda_p = float(lambda_p_str)
        # Przyjęcie Ki jako odwrotności αe
        ki_avg = 1.0 / alpha_e if alpha_e > 0 else 1.0
        alpha_i = 1.0 / ki_avg
        lambda_di = lambda_p * alpha_i

```

```

text = (f"\n4. INTENSYWNOŚĆ USZKODZEŃ NOWYCH ELEMENTÓW\n"
        f"    λp = {lambda_p:.6e}\n"
        f"    αi = {alpha_i:.4f}\n"
        f"    λdi = {lambda_di:.6e}\n")
data = {
    'lambda_p': lambda_p,
    'alpha_i': alpha_i,
    'lambda_di': lambda_di
}
return text, data
return f"\n4. BRAK DANYCH dla λp\n", {}

```

Listing 3.4. Kontroler zarządzający przepływem obliczeń.

3.4 Interfejs graficzny (View)

3.4.1 `gui/main_window.py`

Główne okno aplikacji zarządzające zakładkami.

```

from PyQt5.QtWidgets import QApplication, QMainWindow, QVBoxLayout, QWidget,
QLabel, QMessageBox, QTabWidget
from PyQt5.QtCore import Qt
import sys
from gui.config_tab import ConfigurationTab
from gui.experts_tab import ExpertsTab
from gui.results_tab import ResultsTab
from gui.calculator import ReliabilityCalculator

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Prognozowanie Niezawodności - Metoda Ekspertów")
        self.setGeometry(100, 100, 1200, 800)

        self.calculator = ReliabilityCalculator()

        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)

```

```

        main_layout = QVBoxLayout()
        self.central_widget.setLayout(main_layout)
        header_label = QLabel("System Prognozowania Niezawodności - Metoda
Ekspertów")
        header_label.setStyleSheet("font-size: 18px; font-weight: bold;
padding: 10px;")
        header_label.setAlignment(Qt.AlignCenter)
        main_layout.addWidget(header_label)
        self.tabs = QTabWidget()
        main_layout.addWidget(self.tabs)
        self.config_tab = ConfigurationTab()
        self.tabs.addTab(self.config_tab, "Konfiguracja")
        self.experts_tab = ExpertsTab()
        self.tabs.addTab(self.experts_tab, "Dane Ekspertów")
        self.results_tab = ResultsTab()
        self.tabs.addTab(self.results_tab, "Wyniki")
        self.config_tab.form_generated.connect(self.on_form_generated)
        self.experts_tab.calculation_requested.connect(self.on_calculate_relia
bility)
        self.statusBar().showMessage("Gotowy do pracy")

    def on_form_generated(self):
        config = self.config_tab.get_configuration()
        self.experts_tab.generate_forms(
            config['j_mech'],
            config['j_elec'],
            config['n_groups']
        )
        self.statusBar().showMessage("Formularze wygenerowane pomyślnie")

    def on_calculate_reliability(self):
        try:
            config = self.config_tab.get_configuration()
            kmj_values, expert_data = self.experts_tab.get_expert_data()
            results_text, results_data = self.calculator.calculate(config,
kmj_values, expert_data)
            self.results_tab.display_results(results_text, results_data)
            self.tabs.setCurrentIndex(2)
            self.statusBar().showMessage("Obliczenia zakończone pomyślnie")
            QMessageBox.information(self, "Sukces", "Obliczenia wykonane
pomyślnie!")

```

```

        except Exception as e:
            QMessageBox.critical(self, "Błąd", f"Wystąpił błąd podczas
obliczeń:\n{str(e)}")
            self.statusBar().showMessage("Błąd obliczeń")

    def run(self):
        self.show()

def main():
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()

```

Listing 3.5. Głównie okno aplikacji.

3.4.2 `gui/config_tab.py`

Zakładka konfiguracji z wyborem parametrów.

```

class ConfigurationTab(QWidget):
    """
    Zakładka konfiguracji programu.
    Umożliwia wybór trybu obliczeń, liczby ekspertów i danych progностycznych.
    """

    form_generated = pyqtSignal()

    def __init__(self):
        super().__init__()
        self.data_generator = DataGenerator()
        self.init_ui()

    def _create_mode_group(self, layout):
        """Tworzy grupę wyboru trybu obliczeń"""
        mode_group = QGroupBox("Wybierz tryb obliczeń")
        mode_layout = QVBoxLayout()
        mode_group.setLayout(mode_layout)

```

```

self.mode_combo = QComboBox()
self.mode_combo.addItems([
    "Obliczanie A0 (średni strumień uszkodzeń)",
    "Obliczanie E(T0) (oczekiwany czas pracy)",
    "Obliczanie λdi (intensywność uszkodzeń nowych elementów)"
])
mode_layout.addWidget(QLabel("Tryb:"))
mode_layout.addWidget(self.mode_combo)
layout.addWidget(mode_group)

def get_configuration(self):
    """
    Zwraca słownik z konfiguracją.

    Returns:
        dict: {
            'mode': int (0/1/2),
            'j_mech': int, 'j_elec': int,
            'n_modules': int,
            'n_groups': int,
            'a_mean': str,
            't_expected': str,
            'lambda_p': str
        }
    """
    return {
        'mode': self.mode_combo.currentIndex(),
        'j_mech': self.mech_experts_spin.value(),
        'j_elec': self.elec_experts_spin.value(),
        'n_modules': self.modules_spin.value(),
        'n_groups': self.groups_spin.value(),
        'a_mean': self.a_mean_input.text(),
        'a_upper': self.a_upper_input.text(),
        't_expected': self.t_expected_input.text(),
        'lambda_p': self.lambda_p_input.text()
    }
}

```

Listing 3.6. Zakładka konfiguracji z wyborem parametrów.

3.5 Narzędzia pomocnicze (Utils)

3.5.1 `utils/data_validation.py`

```
def validate_positive_number(value):
    """
    Waliduje czy wartość jest liczbą dodatnią.
    Args:
        value: Wartość do walidacji (str lub liczba)
    Returns:
        float: Zwalidowana liczba
    Raises:
        ValueError: Gdy wartość nie jest dodatnią liczbą
    """
    try:
        number = float(value)
        if number <= 0:
            raise ValueError("Wartość musi być liczbą dodatnią.")
        return number
    except ValueError as e:
        raise ValueError(f"Nieprawidłowe dane wejściowe: {e}")

def validate_percentage(value):
    """
    Waliduje czy wartość jest procentem (0-100).
    Args: value: Wartość do walidacji

    Returns:
        float: Zwalidowana wartość procentowa

    Raises:
        ValueError: Gdy wartość nie jest w zakresie 0-100
    """
    try:
        percentage = float(value)
        if percentage < 0 or percentage > 100:
            raise ValueError("Procent musi być w zakresie 0-100.")
        return percentage
    except ValueError as e:
        raise ValueError(f"Nieprawidłowe dane wejściowe: {e}")
```

Listing 3.7. Walidacja parametrów wejściowych.

3.5.2 `utils/data_generator.py`

Generator losowych danych testowych.

```
import random

class DataGenerator:
    """Generates random data for reliability calculations"""

    def __init__(self):
        self.random = random.Random()

    def set_seed(self, seed=None):
        """Set random seed for reproducible results"""
        self.random.seed(seed)

    def generate_kmj_values(self, count):
        """Generate random Kmj values"""
        return [self._generate_kmj() for _ in range(count)]

    def _generate_kmj(self):
        """Generate single Kmj value (0.5 - 1.5)"""
        return round(self.random.uniform(0.5, 1.5), 4)

    def generate_expert_data(self, numExperts, numGroups):
        """Generate random expert data"""
        return [self._generate_expert_groups(numGroups)
               for _ in range(numExperts) ]

    def _generate_expert_groups(self, numGroups):
        """Generate groups for a single expert"""
        return [self._generate_group_values()
               for _ in range(numGroups) ]

    def _generate_group_values(self):
        """Generate (ng_n, kg) values for a group"""
        ng_n = round(self.random.uniform(0.8, 1.5), 4)
        kg = round(self.random.uniform(0.7, 1.2), 4)
        return (ng_n, kg)
```

```

def generate_a_mean(self):
    """Generate random average failure stream (scientific notation)"""
    exponent = self.random.randint(-6, -3)
    mantissa = self.random.uniform(1.0, 9.9)
    return f"{mantissa:.2f}e{exponent}"

def generate_t_expected(self):
    """Generate random expected operation time (hours)"""
    return round(self.random.uniform(500, 10000), 2)

def generate_lambda_p(self):
    """Generate random failure intensity (scientific notation)"""
    exponent = self.random.randint(-7, -4)
    mantissa = self.random.uniform(1.0, 9.9)
    return f"{mantissa:.2f}e{exponent}"

def generate_config_value(self, mode):
    """Generate appropriate value based on calculation mode"""
    mode_generators = {
        0: self.generate_a_mean,
        1: self.generate_t_expected,
        2: self.generate_lambda_p
    }
    generator = mode_generators.get(mode)
    return generator() if generator else None

```

Listing 3.8. Generator losowych danych testowych.

3.6 System eksportu

3.6.1 `gui/export/txt_exporter.py`

```
class TxtExporter:
    """Ekporter wyników do formatu TXT"""

    def export(self, results_text, results_data):
        """
        Tworzy sformatowany tekst do zapisania.
        Args:
            results_text (str): Tekst wyników
            results_data (dict): Dane strukturalne
        Returns:
            str: Pełny tekst do zapisu
        """
        timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        content = f"""
{'=' * 70}
    RAPORT OBLICZEŃ NIEZAWODNOŚCI - METODA EKSPERTÓW
{'=' * 70}
Data wygenerowania: {timestamp}
{results_text}
{'=' * 70}
DANE WEJŚCIOWE:
{'=' * 70}
Konfiguracja:
    - Tryb obliczeń: {self._get_mode_name(results_data['mode'])}
    - Liczba mechaników: {len(results_data['kmj_values'])}
    - Liczba elektroników: {len(results_data['expert_data'])}
Wartości Kmj: {results_data['kmj_values']}
{'=' * 70}
Koniec raportu
{'=' * 70}
"""

        return content

    def _get_mode_name(self, mode):
        """Zwraca nazwę trybu"""
        modes = {
            0: "Obliczanie A0",
            1: "Obliczanie E(T0)",
            2: "Obliczanie λdi"
        }
        return modes.get(mode, "Nieznany")
```

Listing 3.9. Kod eksportu danych do pliku.

3.7 Punkt wejścia aplikacji

3.7.1 `main.py`

```
import sys
from PyQt5.QtWidgets import QApplication
from gui.main_window import MainWindow

def main():
    """
    Punkt wejścia aplikacji.
    Workflow:
        1. Utworzenie instancji QApplication
        2. Utworzenie głównego okna
        3. Wyświetlenie okna
        4. Uruchomienie pętli zdarzeń Qt
    """
    app = QApplication(sys.argv)

    # Konfiguracja stylu aplikacji
    app.setStyle('Fusion')

    # Utworzenie i wyświetlenie głównego okna
    window = MainWindow()
    window.show()

    # Uruchomienie pętli zdarzeń
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```

Listing 3.10. Punkt wejścia aplikacji.

4. Opis i wyniki testowania aplikacji

Aplikacja została przetestowana w czterech etapach:

1. **Testy jednostkowe** – sprawdzające poprawność funkcji obliczeniowych (calculate_alpha_k(), calculate_alpha_e(), calculate_A0() itp.).
2. **Testy integracyjne** – weryfikujące poprawną współpracę modułów programu (konfiguracja, formularze, obliczenia, GUI).
3. **Testy walidacji** – badające reakcję programu na błędne dane wejściowe.
4. **Testy manualne** GUI i zgodności wyników z literaturą.

4.1. Wyniki testów jednostkowych oraz integracyjnych

W celu weryfikacji poprawności działania aplikacji ReliabilityExpert przeprowadzono zestaw automatycznych testów jednostkowych oraz integracyjnych z wykorzystaniem frameworka pytest.

Testy miały na celu potwierdzenie zgodności wyników obliczeń z dokumentacją teoretyczną, poprawności walidacji danych wejściowych oraz spójności działania pomiędzy poszczególnymi modułami programu.

Zestaw testów podzielono na trzy główne grupy:

1. **Testy obliczeniowe (calculation tests)** – sprawdzające poprawność implementacji podstawowych funkcji matematycznych programu, takich jak:
 - obliczanie współczynnika niezawodności A0,
 - wyznaczanie oczekiwanej czasu bezawaryjnej pracy E(T0),

- obliczanie współczynników eksperckich α_k , α_e oraz wskaźnika uszkodzeń λdi .

Wszystkie testy zakończyły się wynikiem pozytywnym (5/5), co potwierdza poprawność algorytmów i zgodność wyników z literaturą.

2. Testy przykładowe (example tests) – obejmujące porównanie wyników programu z danymi wzorcowymi pochodząymi z dokumentacji teoretycznej.

Dla każdego przykładu (m.in. α_k , α_e , A_0 , $E(T_0)$ oraz wag modułów) uzyskano wartości identyczne lub z błędem mniejszym niż 0.001%. Wszystkie testy zakończyły się wynikiem pozytywnym (5/5).

3. Testy walidacji danych (validation tests) – badające odporność programu na niepoprawne dane wejściowe, w tym wartości ujemne, zerowe, przekraczające dopuszczalne granice oraz błędne typy (np. tekst zamiast liczby).

Przeprowadzono 9 testów, które potwierdziły prawidłowe działanie mechanizmów walidacyjnych – w każdym przypadku program poprawnie zgłaszał wyjątek lub komunikat błędu. Wynik: 9/9 testów zakończonych pozytywnie.

```
(venv) maksymiliansowula@Maks-3 ReliabilityExpertMethod % cd /Users/maksymiliansowula/Documents/Studia/2semestrngr/Niezawodność Systemów Komputerowych/Laboratorium\ -\ Tuszyński/ReliabilityExpertMethod && source venv/bin/activate && PYTHONPATH=.. pytests tests/ -v
=====
===== test session starts =====
platform darwin -- Python 3.9.6, pytest-8.4.2, pluggy-1.6.0 -- /Users/maksymiliansowula/Documents/Studia/2semestrngr/Niezawodność Systemów Komputerowych/Laboratorium - Tuszyński/ReliabilityExpertMethod/venv/bin/python3
cachedir: .pytest_cache
rootdir: /Users/maksymiliansowula/Documents/Studia/2semestrngr/Niezawodność Systemów Komputerowych/Laboratorium - Tuszyński/ReliabilityExpertMethod
collected 19 items

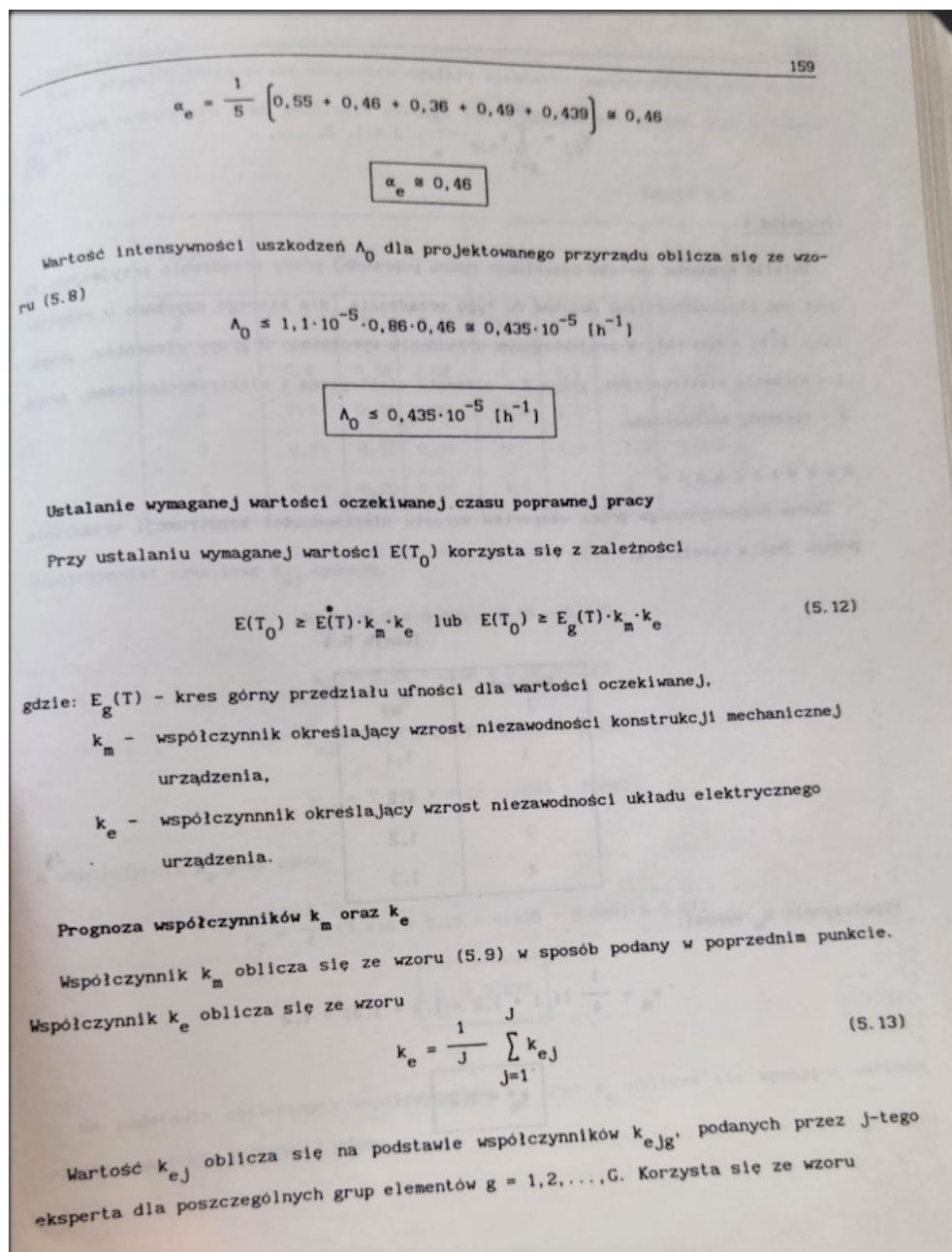
tests/test_calculations.py::TestCalculations::test_calculate_A0 PASSED [ 5%]
tests/test_calculations.py::TestCalculations::test_calculate_E_T0 PASSED [ 10%]
tests/test_calculations.py::TestCalculations::test_calculate_alpha_e PASSED [ 15%]
tests/test_calculations.py::TestCalculations::test_calculate_alpha_k PASSED [ 21%]
tests/test_calculations.py::TestCalculations::test_calculate_lambda_di PASSED [ 26%]
tests/test_examples.py::test_example_alpha_k PASSED [ 31%]
tests/test_examples.py::test_example_alpha_e PASSED [ 36%]
tests/test_examples.py::test_example_A0 PASSED [ 42%]
tests/test_examples.py::test_example_E_T0 PASSED [ 47%]
tests/test_examples.py::test_example_module_weights PASSED [ 52%]
tests/test_validation.py::test_validate_positive_number_valid PASSED [ 57%]
tests/test_validation.py::test_validate_positive_number_string PASSED [ 63%]
tests/test_validation.py::test_validate_positive_number_zero PASSED [ 68%]
tests/test_validation.py::test_validate_positive_number_negative PASSED [ 73%]
tests/test_validation.py::test_validate_percentage_valid PASSED [ 78%]
tests/test_validation.py::test_validate_percentage_zero PASSED [ 84%]
tests/test_validation.py::test_validate_percentage_hundred PASSED [ 89%]
tests/test_validation.py::test_validate_percentage_over_hundred PASSED [ 94%]
tests/test_validation.py::test_validate_percentage_negative PASSED [100%]

=====
===== 19 passed in 0.01s =====
(venv) maksymiliansowula@Maks-3 ReliabilityExpertMethod %
```

Rysunek 4.1. Wyniki testów jednostkowych oraz integracyjnych programu.

4.2. Wyniki testów obliczeń programu

Testy obliczeń programu zostały wykonane poprzez sprawdzenie wyników zgodnie z przykładami zawartymi w [1] punkcie literatury. Rysunek 4.2. przedstawia wynik obliczenia wartości intensywności uszkodzeń dla przykładu, którego obliczenia za pomocą utworzonego programu ukazano na rysunkach 4.3, 4.4 oraz 4.5.



Rysunek 4.2. Wynik wartości intensywności uszkodzeń dla przykładu obliczanego w programie [1].

Poniższy rysunek (4.3) przedstawia wprowadzenie średniej wartości uszkodzeń oraz kresu górnego wraz z wybraniem trybu obliczania wartości intensywności uszkodzeń dla woltomierza cyfrowego.

Prognozowanie Niezawodności - Metoda Ekspertów

System Prognozowania Niezawodności - Metoda Ekspertów

Konfiguracja Dane Ekspertów Wyniki

Wybierz tryb obliczeń

Tryb:
Obliczanie A0 (średni strumień uszkodzeń)

Liczba ekspertów

Mechanicy (J_mech): 3
Elektronicy (J_elec): 5

Parametry systemu

Liczba modułów (N): 15
Liczba grup elementów: 4

Dane prognostyczne (z podobnego urządzenia)

Ā (średni strumień): 0.000011
Ag (górnny kres): 0.000011
t (oczekiwany czas [h]): np. 900
 λ_p (intensywność podobnego): np. 352.8e-8

Losuj parametry prognostyczne Generuj formularze dla ekspertów

Obliczenia zakończone pomyślnie

Rysunek 4.3. Wprowadzenie niezbędnych parametrów startowych dla obliczenia wartości intensywności uszkodzeń dla projektu woltomierza cyfrowego.

Poniższy rysunek (4.4) przedstawia uzupełnione formularze ekspertów – mechaników i elektroników zgodnie z danymi, które były zawarte w przykładzie w literaturze.

Prognozowanie Niezawodności - Metoda Ekspertów

System Prognozowania Niezawodności - Metoda Ekspertów

Konfiguracja Dane Ekspertów Wyniki

Mechanicy - Współczynniki Km_j (od 1 do 2)

Km1:	1,30	^	v
Km2:	1,00	^	v
Km3:	1,20	^	v

Elektronicy - Dane dla grup elementów

Ekspert elektroniczny 1:

Ng/N (udział grupy)	0.15	Kg (współczynnik 1-10)	1	
1	0.05	1	5	10

Ekspert elektroniczny 2:

Ng/N (udział grupy)	0.48	Kg (współczynnik 1-10)	1.5	
1	0.45	5	10	15

Ekspert elektroniczny 3:

Ng/N (udział grupy)	0.10	Kg (współczynnik 1-10)	1	
1	0.0	1	5	10

Ekspert elektroniczny 4:

Ng/N (udział grupy)	0.01	Kg (współczynnik 1-10)	1	
1	0.04	1	5	10

Ekspert elektroniczny 5:

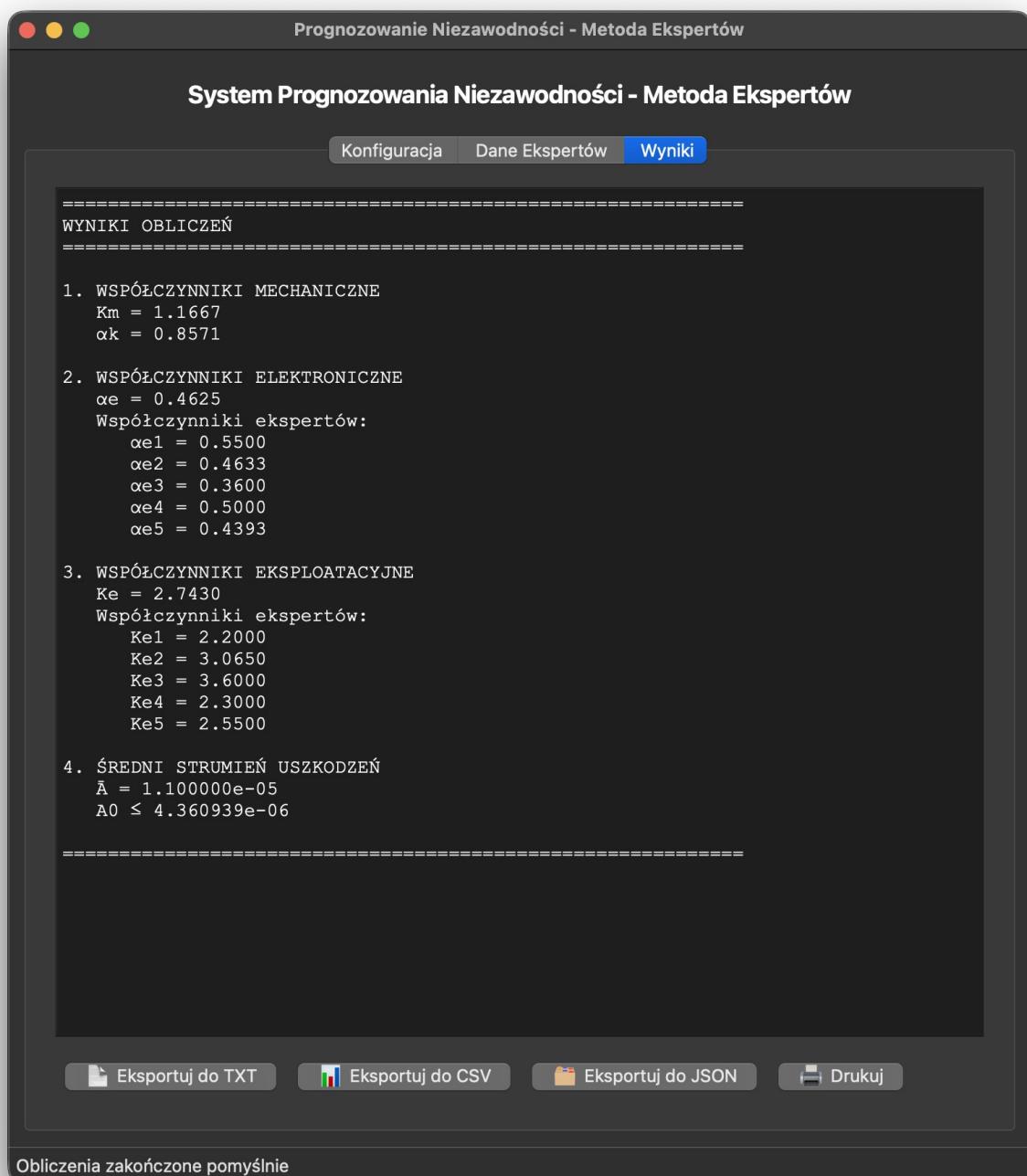
Ng/N (udział grupy)	0.01	Kg (współczynnik 1-10)	1	
1	0.01	1	5	10

Generuj losowe dane
Oblicz wskaźniki niezawodności

Obliczenia zakończone pomyślnie

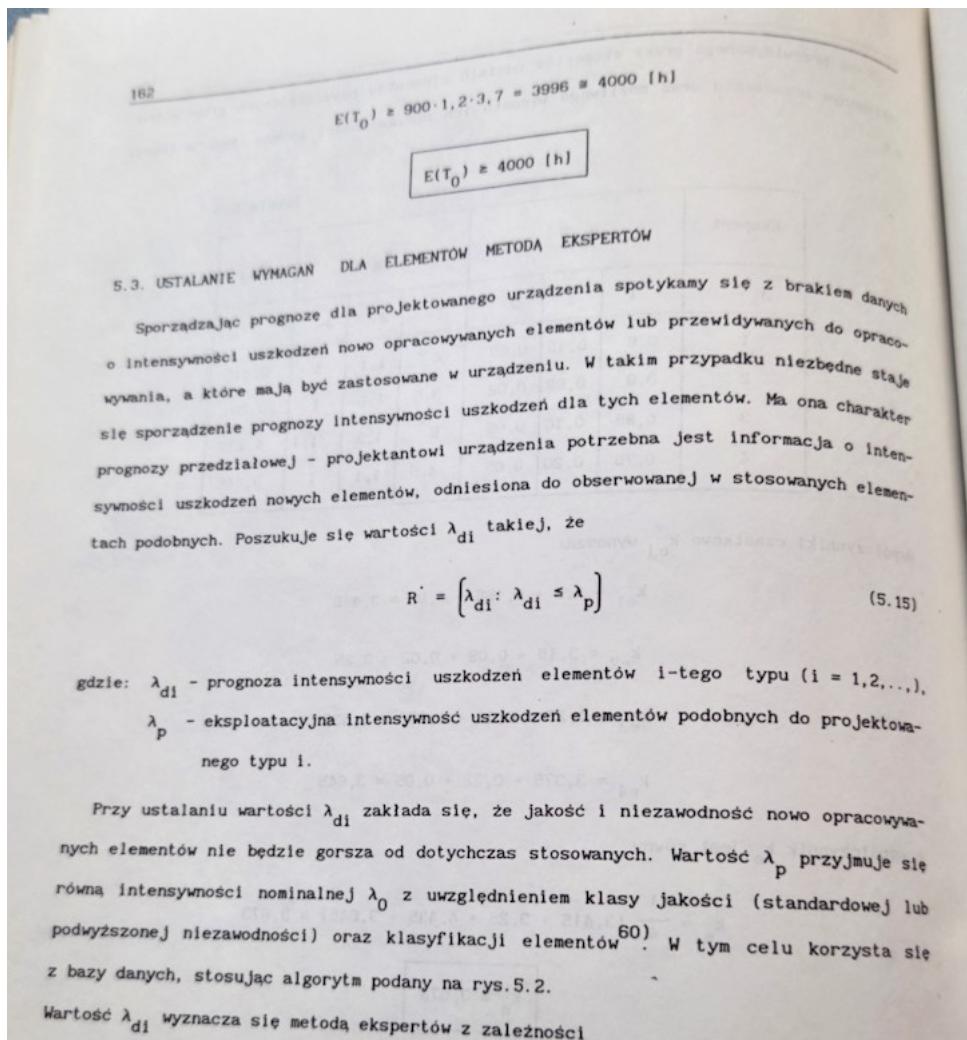
Rysunek 4.4. Uzupełnione formularze ekspertów dla wyliczania współczynnika intensywności uszkodzeń dla woltomierza cyfrowego.

Poniższy rysunek (4.5) przedstawia uzyskane wyniki obliczone przez program na podstawie wprowadzonych danych. Ostateczny wynik ma zgodną wartość z wartością ukazaną na rysunku 4.2.



Rysunek 4.5. Uzyskane wyniki dla wprowadzonych danych w programie.

Drugim przeprowadzonym testem poprawności działania programu było wyznaczenie oczekiwanyego czasu pracy dla urządzenia, które niezawodnościowo jest podobne, dla którego uzyskano w eksploatacji współczynnik oczekiwanygo czasu pracy na poziomie 900 [h]. Rysunek 4.6 przedstawia ostateczny wynik uzyskany z literatury, zaś rysunki 4.7, 4.8 oraz 4.9 wprowadzenie danych do programu oraz wynik obliczeń uzyskany z programu.



Rysunek 4.6. Wynik wartości oczekiwanej czasu pracy urządzenia dla przykładu obliczanego w programie [1].

Poniższy rysunek przedstawia wprowadzoną ilość elektroników, mechaników (ekspertów), liczbę grup elementów oraz oczekiwany czas pracy podobnego urządzenia zgodnie z prezentowanym przykładem w literaturze.

Prognozowanie Niezawodności - Metoda Ekspertów

System Prognozowania Niezawodności - Metoda Ekspertów

Konfiguracja Dane Ekspertów Wyniki

Wybierz tryb obliczeń

Tryb:

Obliczanie E(T0) (oczekiwany czas pracy)

Liczba ekspertów

Mechanicy (J_mech): 4

Elektronicy (J_elec): 4

Parametry systemu

Liczba modułów (N): 15

Liczba grup elementów: 3

Dane prognostyczne (z podobnego urządzenia)

\bar{A} (średni strumień): np. 1.1e-5

Ag (górnny kres): np. 1.5e-5

t (oczekiwany czas [h]): 900

λ_p (intensywność podobnego): np. 352.8e-8

Losuj parametry prognostyczne Generuj formularze dla ekspertów

Formularze wygenerowane pomyślnie

Rysunek 4.7. Wprowadzenie niezbędnych parametrów startowych dla obliczenia wartości oczekiwanej czasu pracy dla pewnego urządzenia z przykładu.

Poniższy rysunek ukazuje uzupełnione wartości tabelek dla mechaników i elektroników (ekspertów), zgodnie z przykładem z literatury.

Prognozowanie Niezawodności - Metoda Ekspertów

System Prognozowania Niezawodności - Metoda Ekspertów

Konfiguracja Dane Ekspertów Wyniki

Mechanicy - Współczynniki Km_j (od 1 do 2)

Km1:	1,10	↑ ↓
Km2:	1,20	↑ ↓
Km3:	1,20	↑ ↓
Km4:	1,30	↑ ↓

Elektronicy - Dane dla grup elementów

Ekspert elektroniczny 1:

	Ng/N (udział grupy)	Kg (współczynnik 1-10)	
1	0.8	4	<div style="width: 40%;"></div>
2	0.15	1.1	<div style="width: 15%;"></div>

Ekspert elektroniczny 2:

	Ng/N (udział grupy)	Kg (współczynnik 1-10)	
1	0.9	3.5	<div style="width: 35%;"></div>
2	0.08	1	<div style="width: 8%;"></div>

Ekspert elektroniczny 3:

	Ng/N (udział grupy)	Kg (współczynnik 1-10)	
2	0.10	1.3	<div style="width: 10%;"></div>
3	0.05	1.1	<div style="width: 5%;"></div>

Ekspert elektroniczny 4:

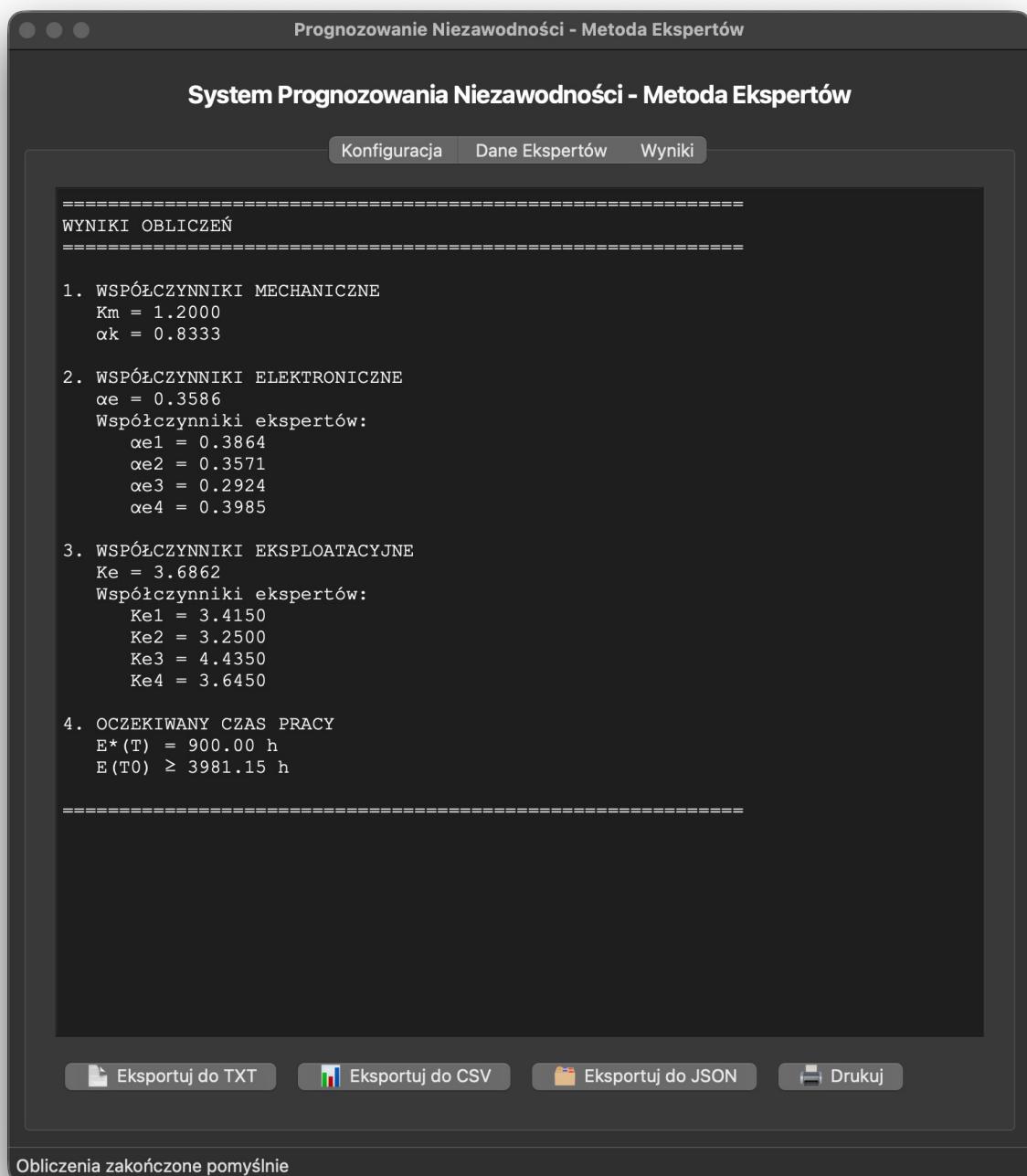
	Ng/N (udział grupy)	Kg (współczynnik 1-10)	
1	0.75	4.5	<div style="width: 75%;"></div>
2	0.20	1.1	<div style="width: 20%;"></div>

 Generuj losowe dane Oblicz wskaźniki niezawodności

Formularze wygenerowane pomyślnie

Rysunek 4.8. Uzupełnione formularze ekspertów dla obliczenia wartości oczekiwanej czasu pracy dla pewnego urządzenia z przykładu.

Na poniższym rysunku zaprezentowano wyniki obliczeń uzyskane z pomocą programu, których oczekiwany czas pracy urządzenia jest zgodny z wynikiem obliczeń z przykładu z literatury.



Rysunek 4.9. Uzyskane wyniki dla wprowadzonych danych w programie.

4.3. Wyniki testów generowania danych do plików

Ostatnimi testami dokonanymi w programie było generowanie danych wejściowych i wyjściowych do plików oraz ich wydruk – testy przeprowadzono dla formatów txt, json oraz csv. Wyniki przedstawiono na rysunkach 4.10, 4.11, 4.12 oraz 4.13.

```
reliability_results_20251020_092435.txt
=====
DANE WEJŚCIOWE
=====
KONFIGURACJA:
Tryb: Oczekiwany czas pracy (E(T0))
Liczba ekspertów: 4
Liczba grup: 5
Oczekiwany czas (E*(T)): 8005.87 h

DANE MECHANICZNE (Kmj):
Kmj1 = 1.2900
Kmj2 = 1.0000
Kmj3 = 1.0000
Kmj4 = 1.1500

DANE EKSPERTÓW (Grupy):
Ekspert 1:
Grupa 1: ng,n = 1.2498, kg = 1.1490
Grupa 2: ng,n = 1.1932, kg = 0.7811
Grupa 3: ng,n = 1.2912, kg = 0.7368
Grupa 4: ng,n = 1.2001, kg = 0.9991
Grupa 5: ng,n = 1.4683, kg = 0.9428

Ekspert 2:
Grupa 1: ng,n = 1.4505, kg = 0.9259
Grupa 2: ng,n = 0.8509, kg = 1.0481
Grupa 3: ng,n = 1.1480, kg = 0.9302
Grupa 4: ng,n = 1.0554, kg = 0.7781
```

Rysunek 4.10. Wynik zapisu danych do pliku txt.

```
reliability_results_20251020_092447.json
{
    "timestamp": "2025-10-20T09:24:47.429949",
    "input_data": {
        "config": {
            "mode": 1,
            "n_mech": 4,
            "n_elect": 4,
            "n_modules": 10,
            "n_groups": 5,
            "a_mean": "",
            "a_upper": "",
            "t_expected": "8005.87",
            "lambda_p": ""
        },
        "kmj_values": [
            1.29,
            1.0,
            1.0,
            1.15
        ],
        "expert_data": [
            {
                "group": 1,
                "n_g,n": 1.2498,
                "kg": 1.149
            },
            {
                "group": 2,
                "n_g,n": 1.1932,
                "kg": 0.7811
            },
            {
                "group": 3,
                "n_g,n": 1.2912,
                "kg": 0.7368
            },
            {
                "group": 4,
                "n_g,n": 1.2001,
                "kg": 0.9991
            },
            {
                "group": 5,
                "n_g,n": 1.4683,
                "kg": 0.9428
            }
        ],
        "group": 1,
        "n_g,n": 1.4505,
        "kg": 0.9259
    }
}
```

Rysunek 4.11. Wynik zapisu danych do pliku json.

reliability_results_20251020_092442

Dane tabeli zostały zainportowane i można je skorygować.

DANE WEJŚCIOWE I WYNIKI OBLCZEŃ

Tryb obliczeń	Oczekiwany czas pracy (E(T0))
Liczba ekspertów	4
Liczba grup	5
Oczekiwany czas (E*(T))	8005.87 h

DANE MECHANICZNE - Kmj

Kmj1	1.2900
Kmj2	1.0000
Kmj3	1.0000
Kmj4	1.1500

DANE EKSPERTÓW (Grupy)

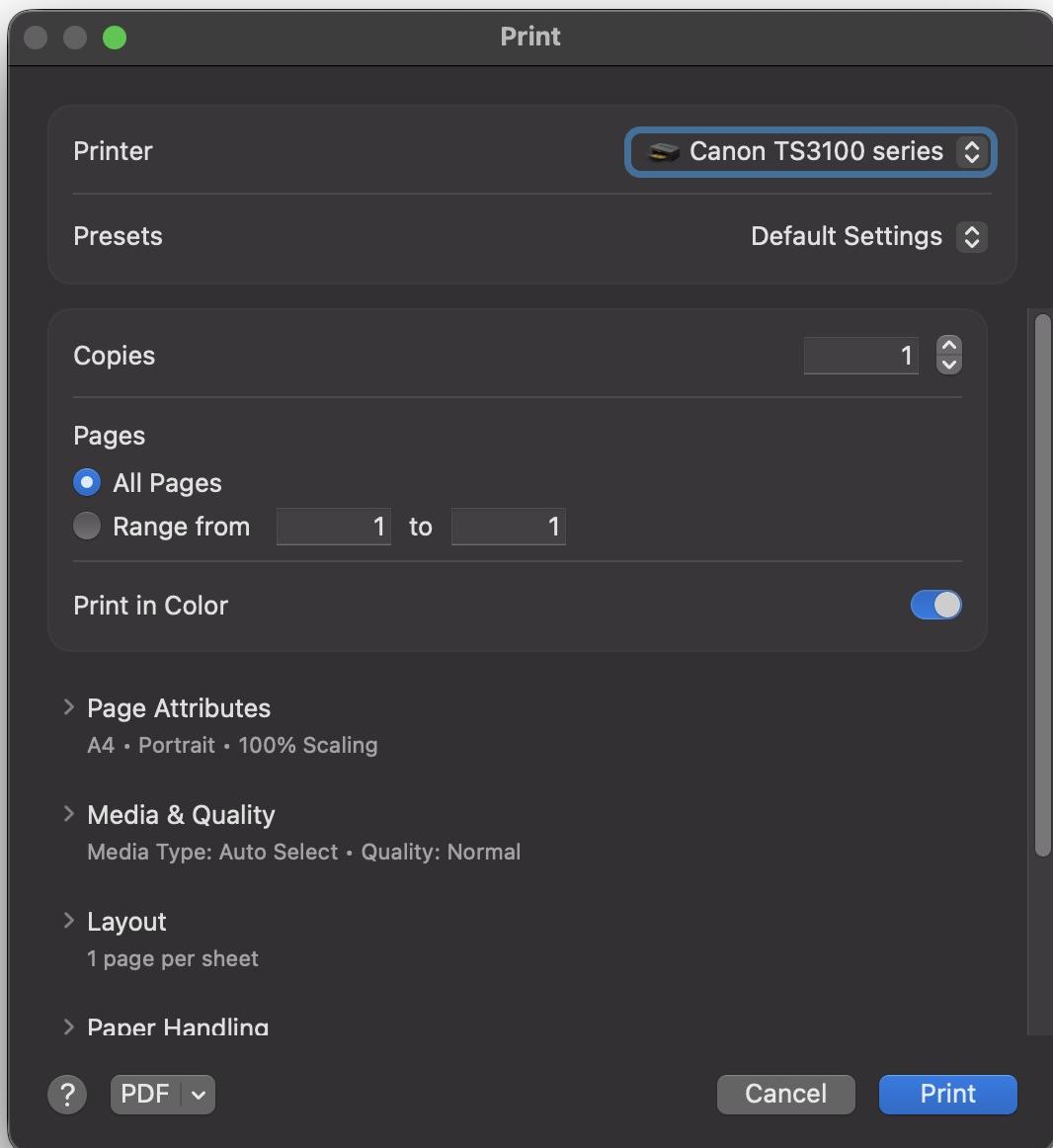
Ekspert 1

Grupa	ng_n	kg
1	1.2498	1.1490
2	1.1932	0.7811
3	1.2912	0.7368
4	1.2001	0.9991
5	1.4683	0.9428

Ekspert 2

Grupa	ng_n	kg
1	1.4505	0.9259

Rysunek 4.12. Wynik zapisu danych do pliku csv.



Rysunek 4.13. Menu kontekstowe systemu do wydrukowania wyników osiągnięte po kliknięciu przycisku „Drukuj” w programie.

5. Instrukcja obsługi programu

5.1. Konfiguracja środowiska uruchomieniowego

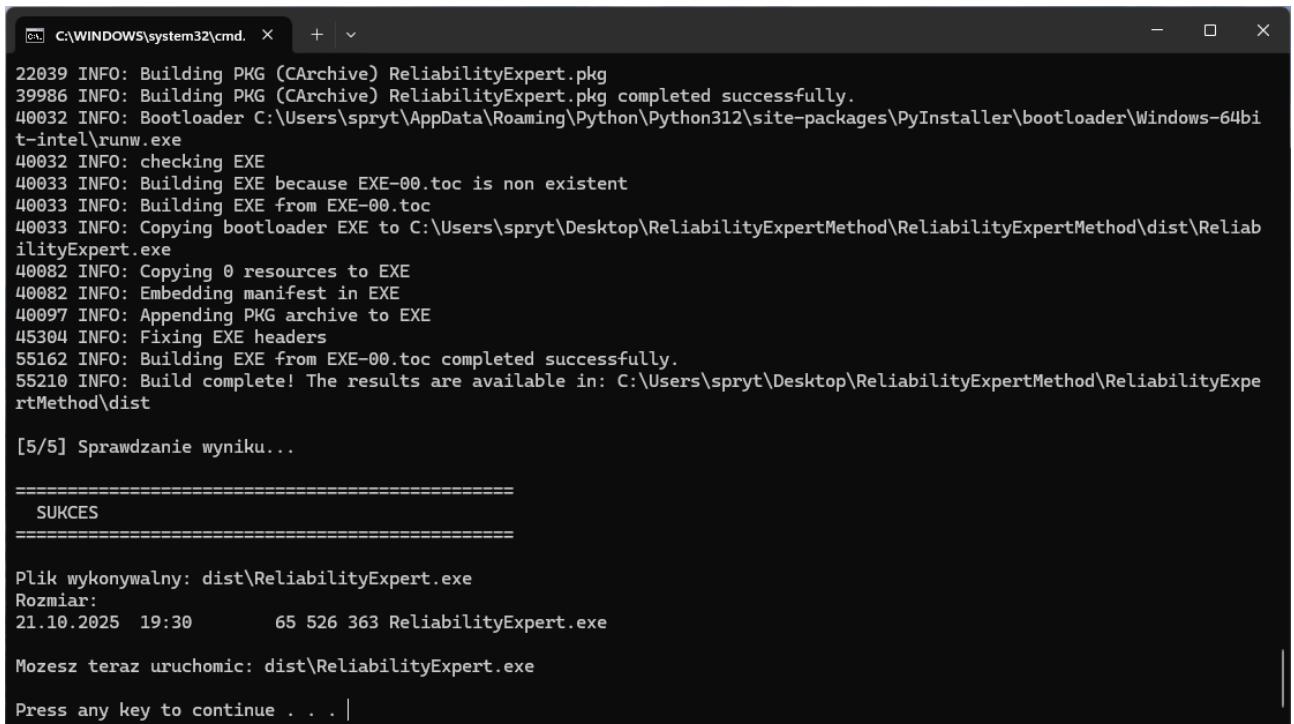
Aby uruchomić aplikację, należy w pierwszej kolejności pobrać paczkę .zip zawierającą cały projekt. Po jej pobraniu należy rozpakować archiwum w dowolnym miejscu na dysku lokalnym, na przykład na Pulpicie lub w folderze „Dokumenty”. W rozpakowanym katalogu powinny znajdować się wszystkie pliki źródłowe niezbędne do działania programu, takie jak main.py, requirements.txt oraz katalogi z modułami i danymi.

Kolejnym krokiem jest instalacja środowiska **Python** w wersji **3.10 lub nowszej**. W tym celu należy przejść na oficjalną stronę projektu Python pod adresem <https://www.python.org/downloads/>, a następnie pobrać instalator odpowiedni dla używanego systemu operacyjnego (Windows, macOS lub Linux). Podczas instalacji należy zaznaczyć opcję „**Add Python to PATH**”, co umożliwia uruchamianie poleceń Pythona z poziomu wiersza poleceń. Po zakończeniu instalacji można zweryfikować poprawność konfiguracji, wpisując w terminalu polecenie python --version (lub python3 --version w systemach Unix), które powinno zwrócić numer zainstalowanej wersji.

Po poprawnym zainstalowaniu python paczka zawiera dwa skrypty do zbudowania aplikacji na różne systemy operacyjne:

- build_windows.bat na system Windows,
- build_macos_linux.sh na system macOS/Linux

Należy uruchomić jeden ze skryptów w zależności od posiadanego systemu operacyjnego i odczekać do końca instalacji. Pomyślną instalację dla systemu Windows ukazano na rysunku 5.1, a dla systemów macOS/Linux na rysunku 5.2.



```
C:\WINDOWS\system32\cmd. + v
22039 INFO: Building PKG (CArchive) ReliabilityExpert.pkg
39986 INFO: Building PKG (CArchive) ReliabilityExpert.pkg completed successfully.
40032 INFO: Bootloader C:\Users\spryt\AppData\Roaming\Python\Python312\site-packages\PyInstaller\bootloader\Windows-64bit-intel\runw.exe
40032 INFO: checking EXE
40033 INFO: Building EXE because EXE-00.toc is non existent
40033 INFO: Building EXE from EXE-00.toc
40033 INFO: Copying bootloader EXE to C:\Users\spryt\Desktop\ReliabilityExpertMethod\ReliabilityExpertMethod\dist\ReliabilityExpert.exe
40082 INFO: Copying 0 resources to EXE
40082 INFO: Embedding manifest in EXE
40097 INFO: Appending PKG archive to EXE
45304 INFO: Fixing EXE headers
55162 INFO: Building EXE from EXE-00.toc completed successfully.
55210 INFO: Build complete! The results are available in: C:\Users\spryt\Desktop\ReliabilityExpertMethod\ReliabilityExpertMethod\dist

[5/5] Sprawdzanie wyniku...

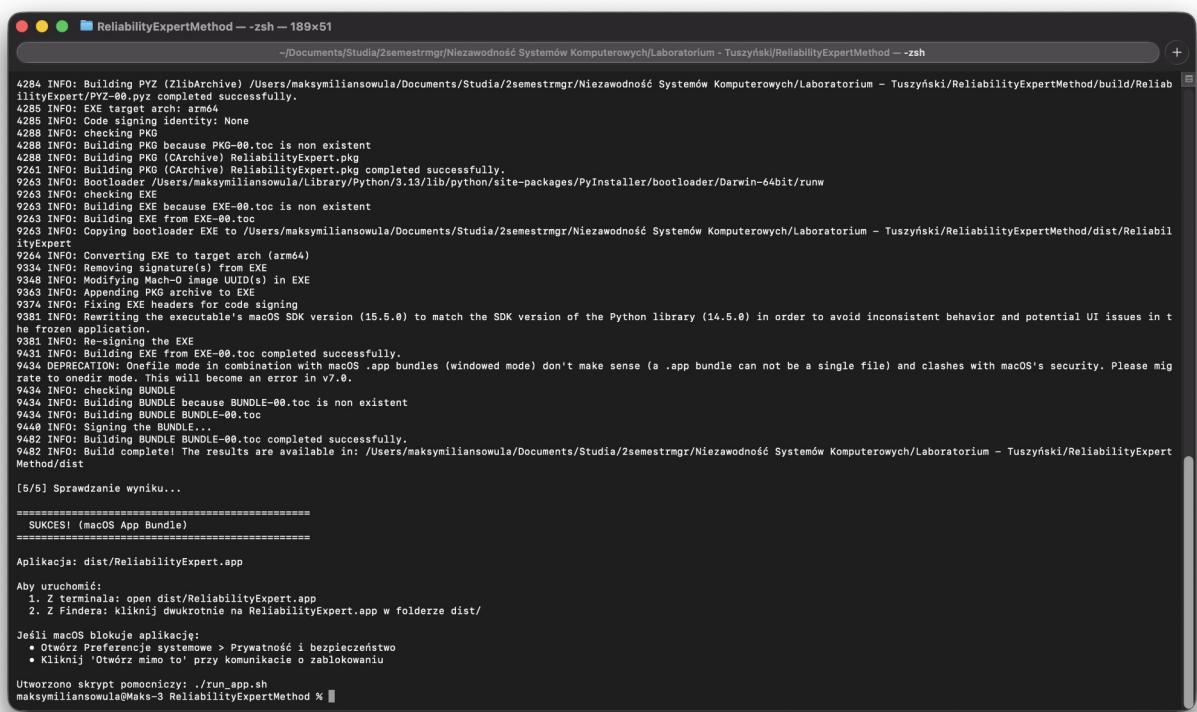
=====
SUKCES
=====

Plik wykonywalny: dist\ReliabilityExpert.exe
Rozmiar:
21.10.2025 19:30      65 526 363 ReliabilityExpert.exe

Mozesz teraz uruchomic: dist\ReliabilityExpert.exe

Press any key to continue . . . |
```

Rysunek 5.1. Wynik pomyślnej instalacji aplikacji na system Windows.



```
~/Documents/Studia/2semestrng/Niezawodnoś Systemów Komputerowych/Laboratorium - Tuszyński/ReliabilityExpertMethod --zsh - 189x51
~/Documents/Studia/2semestrng/Niezawodnoś Systemów Komputerowych/Laboratorium - Tuszyński/ReliabilityExpertMethod/build/ReliabilityExpert/PYZ-00.pyz completed successfully.
4285 INFO: EXE target arch: arm64
4285 INFO: Code signing identity: None
4285 INFO: Copying PKG because PKG-00.toc is non existent
4288 INFO: Building PKG (CArchive) ReliabilityExpert.pkg
9261 INFO: Building PKG (CArchive) ReliabilityExpert.pkg completed successfully.
9263 INFO: Bootloader /Users/maksymiliansowula/Library/Python/3.13/lib/python/site-packages/PyInstaller/bootloader/Darwin-64bit/runw
9263 INFO: checking EXE
9263 INFO: Building EXE because EXE-00.toc is non existent
9263 INFO: Copying bootloader EXE to /Users/maksymiliansowula/Documents/Studia/2semestrng/Niezawodnoś Systemów Komputerowych/Laboratorium - Tuszyński/ReliabilityExpertMethod/dist/ReliabilityExpert
9264 INFO: Converting EXE to target arch (arm64)
9334 INFO: Removing signature(s) from EXE
9348 INFO: Modifying Mach-O image UUID(s) in EXE
9363 INFO: Appending PKG archive to EXE
9374 INFO: Fixing EXE headers for code signing
9381 INFO: Rewriting the executable's macOS SDK version (15.5.0) to match the SDK version of the Python library (14.5.0) in order to avoid inconsistent behavior and potential UI issues in the final application
9381 INFO: Re-signing the EXE
9334 INFO: Building EXE from EXE-00.toc completed successfully.
9334 DEPRECATION: Onefile mode in combination with macOS .app bundles (windowed mode) don't make sense (a .app bundle can not be a single file) and clashes with macOS's security. Please migrate to onefile mode. This will become an error in v7.0.
9434 INFO: checking BUNDLE
9434 INFO: Building BUNDLE because BUNDLE-00.toc is non existent
9434 INFO: Copying BUNDLE BUNDLE-00.toc
9440 INFO: Signing the BUNDLE
9442 INFO: Building BUNDLE BUNDLE-00.toc completed successfully.
9442 INFO: Build complete! The results are available in: /Users/maksymiliansowula/Documents/Studia/2semestrng/Niezawodnoś Systemów Komputerowych/Laboratorium - Tuszyński/ReliabilityExpertMethod/dist

[5/5] Sprawdzanie wyniku...

=====
SUKCES! (macOS App Bundle)
=====

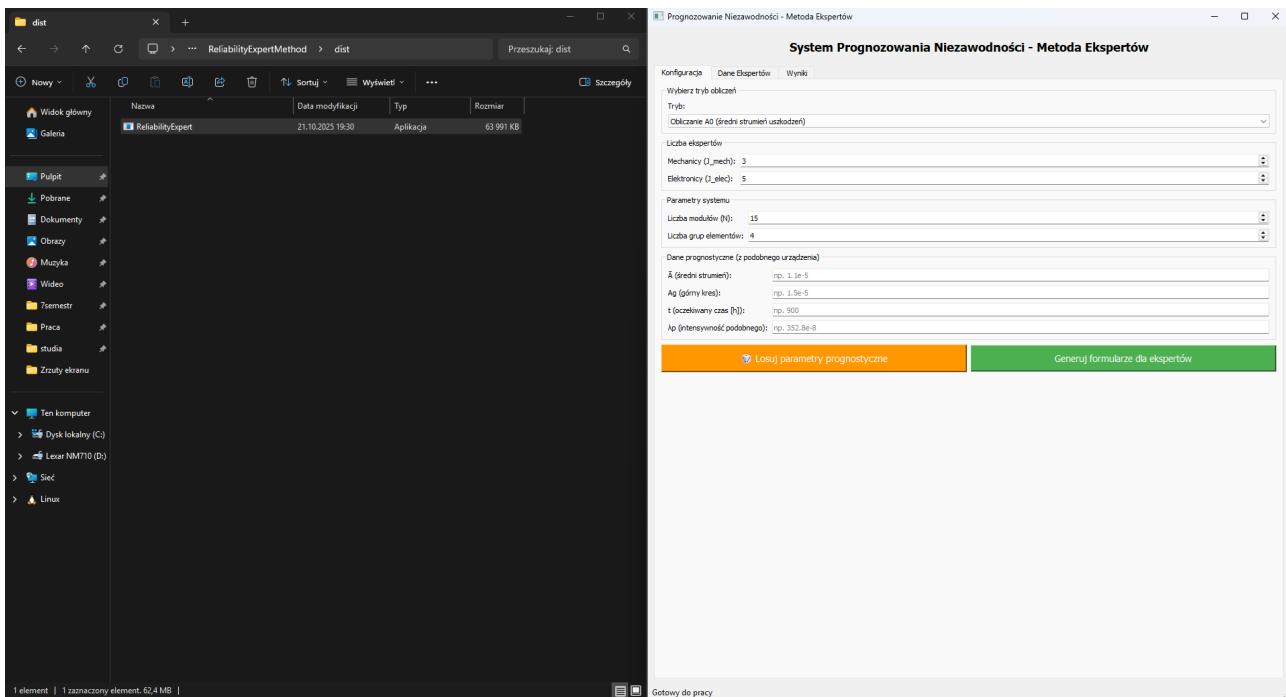
Aplikacja: dist/ReliabilityExpert.app
Aby uruchomić:
1. Z terminala: open dist/ReliabilityExpert.app
2. Z Findera: Kliknij dwukrotnie na ReliabilityExpert.app w folderze dist/

Jeśli macOS blokuje aplikację:
> Otwórz Preferencje systemowe > Prywatność i bezpieczeństwo
> Kliknij 'Otwórz mimo to' przy komunikacie o zablokowaniu

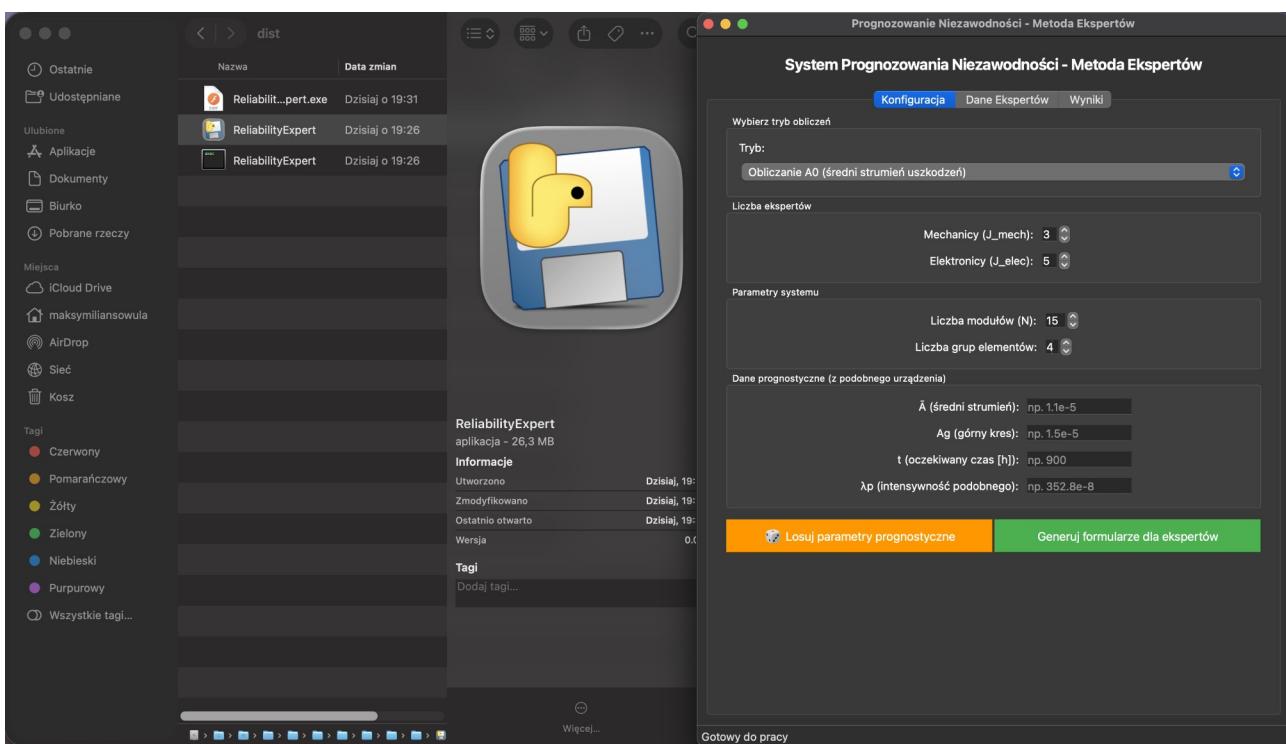
Utworzono skrypt pomocniczy: ./run_app.sh
maksymiliansowula@Maks-3 ReliabilityExpertMethod %
```

Rysunek 5.2. Wynik pomyślnej instalacji aplikacji na systemy macOS/Linux.

Następnie należy uruchomić aplikację z utworzonego po instalacji folderu dist. Wynik uruchomienia w systemie Windows przedstawiono na rysunku 5.3, a wynik uruchomienia w systemie macOS na rysunku 5.4.



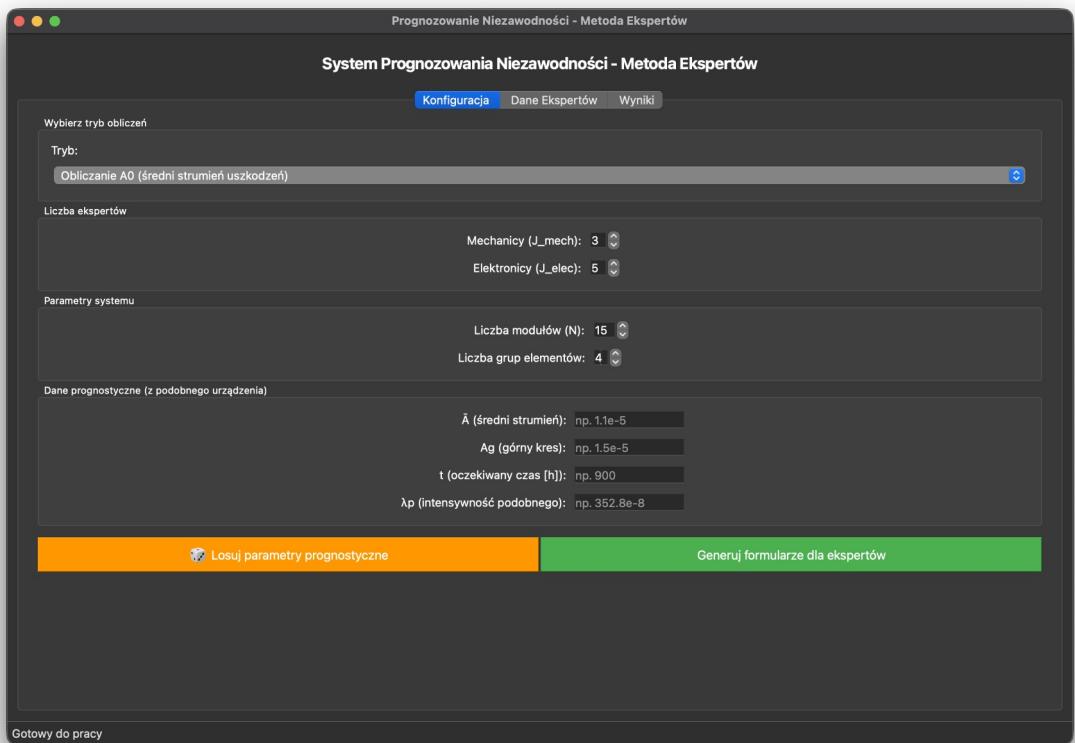
Rysunek 5.3. Wynik uruchomienia aplikacji w systemie Windows.



Rysunek 5.4. Wynik uruchomienia aplikacji w systemie macOS.

5.2. Generowanie formularzy ekspertów

Aby wygenerować formularze ekspertów na początku należy wybrać jeden z trzech trybów obliczeń ukazanych na rysunku 5.5.

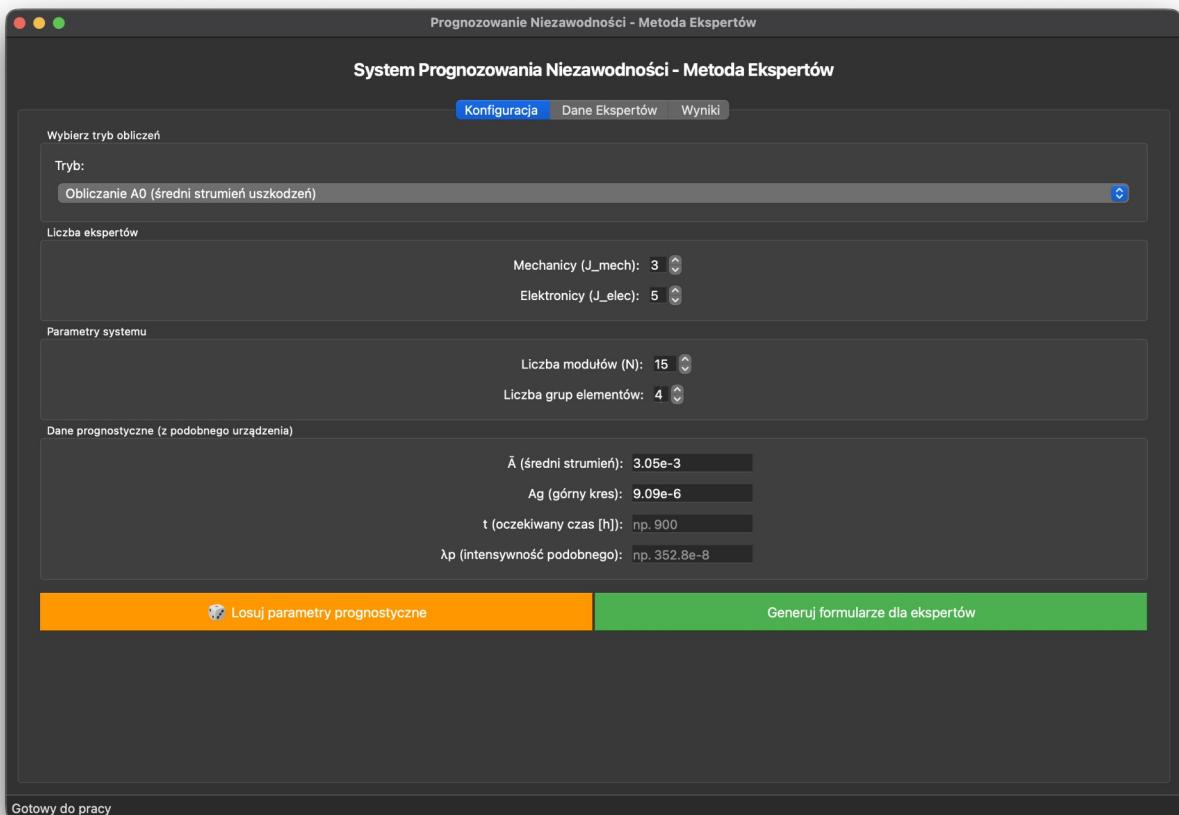


Rysunek 5.5. Wybór jednego z trzech trybów obliczeń.

Następnie należy uzupełnić niezbędne dane do wykonania obliczeń. Można to wykonać w sposób tradycyjny lub poprzez kliknięcie „Losuj parametry prognostyczne”. W tabeli 5.1 przedstawiono wymagane dane dotyczące każdego z trybu obliczeń, a na rysunku 5.6 przedstawiono losowo wygenerowane dane dla obliczenia średniego strumienia uszkodzeń urządzenia.

Tabela 5.1. Wymagane dane do konkretnych trybów obliczeń.

Tryb obliczeń	Wymagane dane
Średni strumień uszkodzeń (A0)	Mechanicy, Elektronicy, Liczba modułów, Liczba grup elementów, średni strumień, górny kres.
Oczekiwany czas pracy (E(T0))	Mechanicy, Elektronicy, Liczba modułów, Liczba grup elementów, t (oczekiwany czas [h])
Intensywność uszkodzeń nowych elementów (λ_{di})	Mechanicy, Elektronicy, Liczba modułów, Liczba grup elementów, λ_p (intensywność podobnego)



Rysunek 5.6. Losowo wygenerowane dane dla trybu obliczania średniego strumienia uszkodzeń.

5.3. Wypełnianie formularzy ekspertów danymi

Kolejnym krokiem jest wypełnienie formularzy ekspertów danymi. W tym celu w zakładce „Konfiguracja” po wypełnieniu niezbędnych danych należy kliknąć przycisk „Generuj formularze dla ekspertów”. Program w lewym dolnym rogu powinien wyświetlić komunikat o treści „Formularze wygenerowane pomyślnie”, co oznacza, że można przejść do zakładki „Dane Ekspertów”. Rysunek 5.7. przedstawia wynik generacji formularzy.

Prognozowanie Niezawodności - Metoda Ekspertów

System Prognozowania Niezawodności - Metoda Ekspertów

Konfiguracja Dane Ekspertów Wyniki

Wybierz tryb obliczeń

Tryb:
Obliczanie A0 (średni strumień uszkodzeń)

Liczba ekspertów

Mechanicy (J_mech): 3
Elektronicy (J_elec): 5

Parametry systemu

Liczba modułów (N): 15
Liczba grup elementów: 4

Dane progностyczne (z podobnego urządzenia)

\bar{A} (średni strumień): 3.05e-3
Ag (górnny kres): 9.09e-6
t (oczekiwany czas [h]): np. 900
 λ_p (intensywność podobnego): np. 352.8e-8

Losuj parametry progностyczne Generuj formularze dla ekspertów

Formularze wygenerowane pomyślnie

Rysunek 5.7. Wynik generacji formularzy.

Kolejnym krokiem jest uzupełnienie formularzy ekspertów. W tym celu należy kliknąć zakładkę „Dane Ekspertów”. Ukaże się nam okno do uzupełnienia współczynników mechaników oraz elektroników. Można uzupełnić je ręcznie lub wylosować wartości za pomocą przycisku „Generuj losowe dane”. Rysunek 5.8. przedstawia wynik wygenerowania losowo danych.

Prognozowanie Niezawodności - Metoda Ekspertów

System Prognozowania Niezawodności - Metoda Ekspertów

Konfiguracja Dane Ekspertów Wyniki

Mechanicy - Współczynniki Km_j (od 1 do 2)

Km1:	1,20
Km2:	1,37
Km3:	1,41

Elektronicy - Dane dla grup elementów

Ekspert elektroniczny 1:

Ng/N (udział grupy)	Kg (współczynnik 1-10)
1 1.2715	1.1368
2 1.0779	1.1672

Ekspert elektroniczny 2:

Ng/N (udział grupy)	Kg (współczynnik 1-10)
1 1.2011	0.9714
2 0.9145	0.8383

Ekspert elektroniczny 3:

Ng/N (udział grupy)	Kg (współczynnik 1-10)
1 1.0866	1.1645
2 1.4268	0.8606

Ekspert elektroniczny 4:

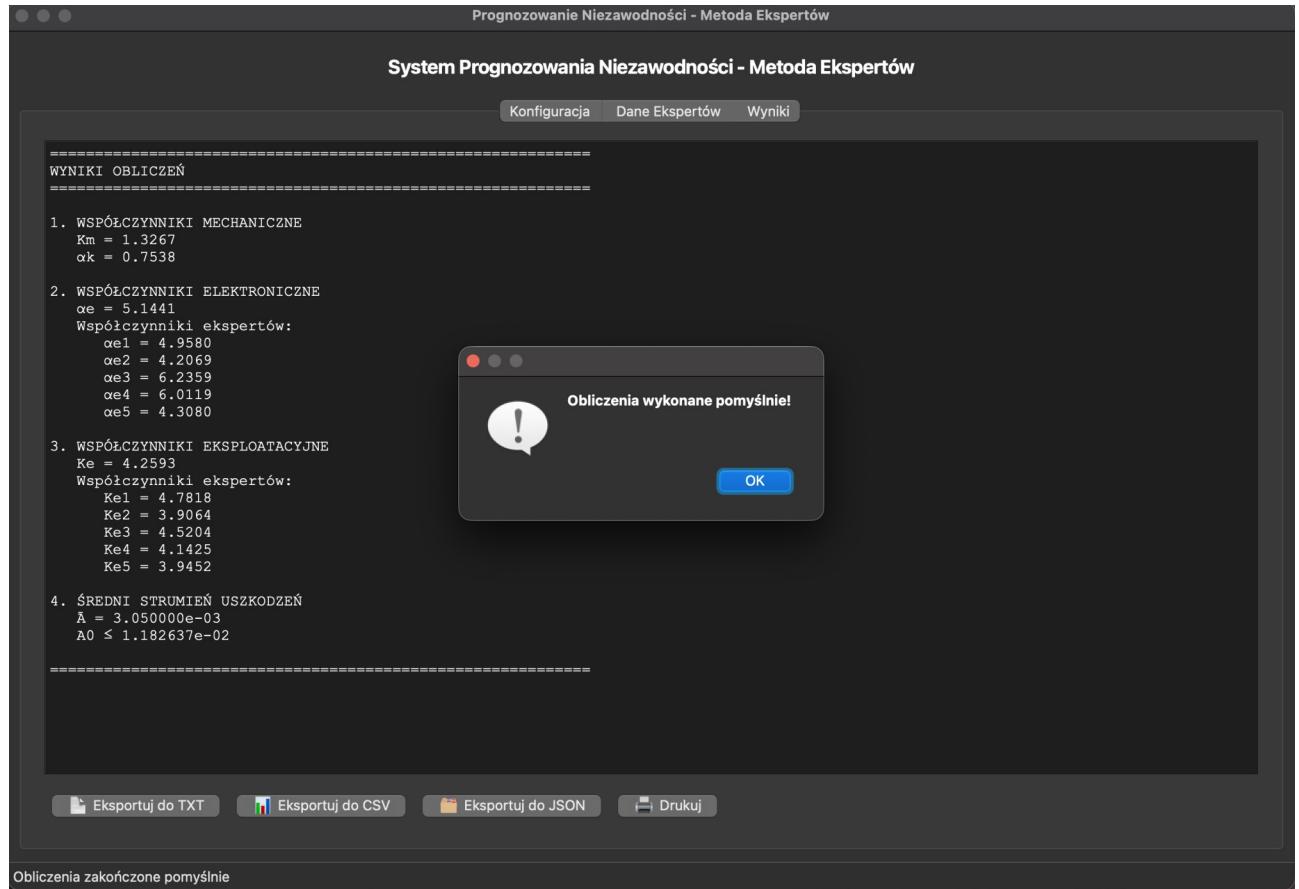
Ng/N (udział grupy)	Kg (współczynnik 1-10)
1 1.2699	1.1329
2 1.2995	0.7049

Generuj losowe dane Oblicz wskaźniki niezawodności

Formularze wygenerowane pomyślnie

Rysunek 5.8. Wygenerowanie losowo danych formularzy ekspertów.
Ostatnim krokiem jest kliknięcie przycisku „Oblicz wskaźniki niezawodności”, który

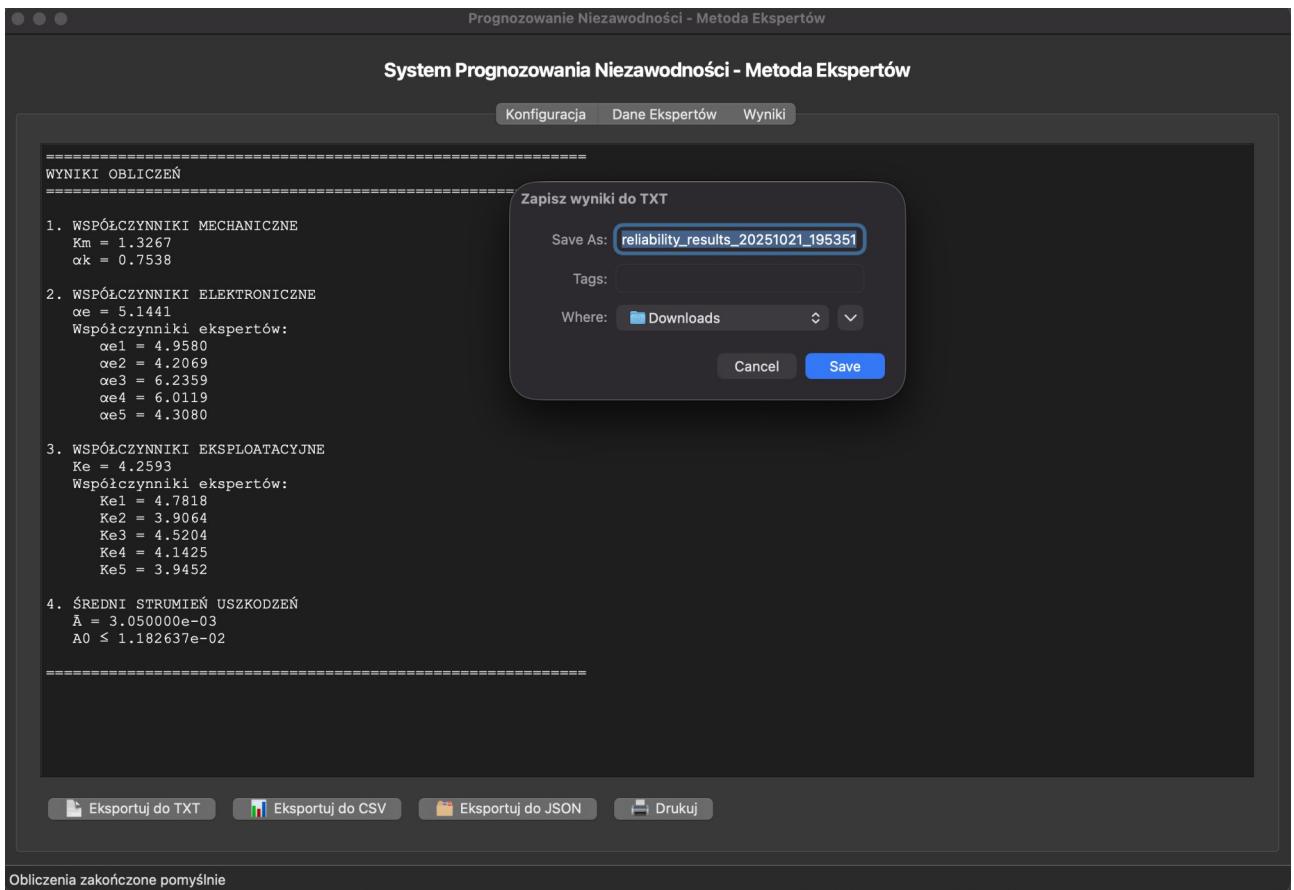
spowoduje automatyczne przekierowanie do zakładki „Wyniki”, wyświetlenie wyników oraz komunikatu o pomyślnym dokonaniu obliczeń. Efekt kliknięcia przycisku przedstawiono na rysunku 5.9.



Rysunek 5.9. Wynik kliknięcia przycisku „Oblicz wskaźniki niezawodności”.

5.4. Eksport do plików oraz druk

Ostatnim etapem działania aplikacji jest możliwość eksportu wyników obliczeń do trzech formatów plików – TXT, CSV oraz JSON, a także ich wydrukowania bezpośrednio z poziomu programu. Aby skorzystać z tych funkcji, należy najpierw przeprowadzić obliczenia wskaźników niezawodności zgodnie z opisem przedstawionym w poprzednich punktach. Następnie, w zakładce „Wyniki”, użytkownik może wybrać jedną z dostępnych opcji: „Eksportuj do TXT”, „Eksportuj do CSV”, „Eksportuj do JSON” lub „Drukuj”, aby zapisać dane w wybranym formacie lub przygotować je do wydruku. Na rysunku 5.10 zaprezentowano przykładowe eksportowanie wyników do pliku w formacie TXT.



Rysunek 5.10. Wynik kliknięcia przycisku „Eksportuj do TXT”.

LITERATURA

1. Niezawodność wyrobów. Zarządzanie i prognozowanie. Skrypt PŚk nr 255, Kielce 1994.