Zespół: Poniedziałek / 13.15 Wrocław, dn. 18.05.2014

Maciej Rudzikowski 200698  
Damian Noga 200707  
Kacperek Przezak 200616

System szeregowy z magazynem części zamiennych.

*Projekt z przedmiotu niezawodność i diagnostyka układów cyfrowych 2. Rok akademicki 2013 / 2014. Kierunek: INF.*

PROWADZĄCY:

dr inż. Jacek Jarnicki

**Spis treści:**

[1. Wstęp 3](#_Toc389669224)

[2. Model systemu : 4](#_Toc389669225)

[3. Opis symulatora: 5](#_Toc389669226)

[a) Opis: 5](#_Toc389669227)

[b) Dane wejściowe: 5](#_Toc389669228)

[c) Przedstawienie funkcji użytych w symulatorze: 6](#_Toc389669229)

[1. Funkcja: "*generujWektory.m*": 6](#_Toc389669230)

[2. Funkcja: "*symulacjaCzasu.m*": 7](#_Toc389669231)

[3. Funkcja: "*symKonserwatory.m*": 8](#_Toc389669232)

[d) Przedstawienie symulatora: 9](#_Toc389669233)

[1. Listing kodu symulatora: 9](#_Toc389669234)

[2. Opis działania: 10](#_Toc389669235)

[3. Przykład wygenerowanych czasów symulacji: 10](#_Toc389669236)

[4. Wykres histogramu dla danych które będą w dalszej części analizowane: 11](#_Toc389669237)

[4. Analiza danych w programie SAS: 12](#_Toc389669238)

[5. Literatura: 13](#_Toc389669239)

# Wstęp

Projekt miał na celu stworzenie symulatora układu n-elementowego z magazynem części zamiennych. Układ ten przewiduje wymiany zepsutych elementów przy pomocy konserwatorów. Liczba konserwatorów jest zależna od liczby elementów w układzie. Do układu przypisany jest magazyn części zapasowych, z którego pobierane są elementy do wymiany. Układ kończy swoją pracę w momencie gdy uszkodzeniu ulegnie element, dla którego nie ma już dostępnych zamienników.

Do wyznaczenia czasu życia każdego z elementów korzystamy z rozkładu Weibull’a o odpowiednich parametrach wczytanych z pliku. Czas wymiany elementu jest również generowany z powyższego rozkładu. Optymalna ilość poszczególnych elementów zapasowych znajdujących się w magazynie jest obliczana metodą przeglądu zupełnego. Przy jej generowaniu pod uwagę bierzemy dostępny budżet, ceny oraz średni czas życia każdego z elementów. W ten sposób uzyskane ilości elementów są przetwarzane w symulatorze.

# Model systemu :

System składa się z n-elementów, magazynu części zamiennych oraz k konserwatorów. Po starcie pracy systemu wszystkie elementy są sprawne. Jako pierwszy uszkodzeniu ulega element, któremu z rozkładu Weibull’a o odpowiednich parametrach przypisany został najkrótszy czas życia. Gdy dany element się zepsuje sprawdzamy czy jest dostępny w magazynie jego zamiennik, jeśli nie, to system kończy swoją pracę z powodu braku części. Jeśli znajdziemy część zamienną w magazynie, to następnie sprawdzamy, czy któryś z konserwatorów nie jest zajęty i może dokonać wymiany uszkodzonego elementu. Jeśli znajdzie się wolny konserwator, część jest pobierana z magazyny i rozpoczyna się wymiana wadliwej na sprawną, której czas jest również generowany z rozkładu Weibull’a. Jeśli nie mamy dostępu do wolnego konserwatora, uszkodzona część oczekuje, aż któryś z konserwatorów zakończy poprzednią pracę i będzie mógł zająć się jej wymianą. Warto zauważyć, że podczas gdy któraś część jest niesprawna, pozostałe elementy układu w dalszym ciągu się zużywają i mogą ulec awarii.

# Opis symulatora:

## Opis:

Symulator został napisany jako skrypt w programie *Matlab*. Pobierane są informacje z pliku dane.txt o ilości elementów, ilości konserwatorów, charakterystyce poszczególnych elementów dzięki którym następuje generowanie czasu życia w symulatorze. Następnie symulator poprzez wykonanie przeglądu zupełnego wyszukuje najbardziej optymalny wektor ilości poszczególnych elementów z uwzględnieniem zaplanowanego budżetu. Uruchamiamy symulację z której otrzymujemy plik wynikowy z przebiegiem symulacji (historię życia poszczególnych elementów) oraz czasem działania całego systemu. Następnie wykonujemy wiele razy w/w symulację w celu uzyskania dużej próby danych, które w dalszej części poddamy wnikliwej analizie.

W celu lepszej organizacji kodu symulator został podzielony na poszczególne funkcje takie jak:

\* *generujWektory.m*   
 \* symulacjaCzasu.m  
 \* symKonserwatory.m  
 \* symulator.m

## Dane wejściowe:

Podczas startu systemu zostaje wczytany plik  *dane.txt*  który zawiera niezbędne informacje z którymi będzie pracował symulator. Plik ten znajduje się w tym samym folderze co projekt symulatora. Zawiera on następujące informacje:

**Ilość elementów w symulacji**: n   
**Ilość konserwatorów**: k

Dla każdego z n elementów musimy zdefiniować parametry rozkładu Weibull'a w następujący sposób: "skala", "kształt".

Przykładowy plik *dane.txt* :

|  |
| --- |
| skala ksztalt  3 1 // ilość el, ilość konserwatorów  1000 10   500 2  200 20 |

1. Opis pliku dane.txt

## Przedstawienie funkcji użytych w symulatorze:

### Funkcja: "*generujWektory.m*":

Definicja funkcji:

|  |
| --- |
| function [ macierzW ] = generujWektory( kosztEl, budzet ) |

1. Nagłówek funkcji "generujWektory".

Opis:

Jak widać na powyższym listingu funkcja przyjmuje 2 argumenty takie jak: wektor kosztów poszczególnych elementów (np.: [ 10 7 ] przy 2 elementach) oraz dostępny budżet który możemy przeznaczyć na zakup części zamiennych. Funkcja następnie oblicza ile możemy zakupić najtańszych części i generuje wszystkie możliwości zakupu części. Zakładamy, że budżet musi być wykorzystany w minimum 90%. Wygenerowane dane są w postaci macierzy gdzie ilość kolumn jest równa ilości elementów a ilość wierszy jest dość duża gdyż są to wszystkie możliwe konfiguracje ilości elementów. Następnie macierz w/w wektorów będzie wykorzystywana podczas przeglądu zupełnego w celu określenia najbardziej optymalnej konfiguracji zakupu poszczególnego typu elementów.   
 Powyższe postępowanie doprowadzi nas do najlepszego wykorzystania założonego budżetu gdyż w dalszej części symulatora będziemy wyszukiwać z w/w macierzy najbardziej optymalne wydanie naszego budżetu.

Przykład wygenerowanej macierzy dla następujących danych:

**kosztyEl** = [10 15 20 25].  
 **budżet** = 100.

|  |
| --- |
| 10 0 0 0  8 1 0 0  8 0 1 0  7 2 0 0  7 0 0 1  6 1 1 0  6 1 0 1  6 0 2 0  0 1 4 0  0 1 3 1  0 0 5 0  0 0 1 3  0 0 0 4 |

1. Przykład macierzy wygenerowanej za pomocą funkcji "generujWektory".

### Funkcja: "*symulacjaCzasu.m*":

Definicja funkcji:

|  |
| --- |
| function [ srednia ] = symulacjaCzasu(magazyn) |

1. Nagłówek funkcji "symulacjaCzasu.m".

Opis:   
 Jak widać na powyższym listingu funkcja przyjmuje 1 parametr. Jest to wektor ilości elementów w magazynie (np. magazyn = [10, 4, 6] dla 2 el.). Następnie funkcja generuje czasy życia poszczególnych elementów oraz odlicza upływający czas. Gdy dojdzie do uszkodzenia elementu jest on natychmiast wymieniany oraz odejmowany od wektora reprezentującego magazyn części zamiennych. Funkcja kończy swoje działanie gdy uszkodzi się element a w magazynie nie będzie jego zamiennika. Zliczony czas jest zapisywany do wektora wynikowego. Funkcja zapisuje historię uszkodzeń elementów w pliku "*wynik.txt*".  
 Powyższą symulację przeprowadzamy 30 razy, a wynik uśredniamy dla uzyskania dokładniejszego wyniku.

Przykład przeprowadzonej symulacji czasu wyżej opisaną funkcją:

|  |
| --- |
| 650.00 0 1  850.00 1 0  630.00 0 1  930.00 1 0  140.00 0 1  460.00 0 1  600.00 0 1  930.00 1 0  360.00 0 1  780.00 0 1  1070.00 1 0 |

1. Przykład użycia funkcji "symulacjaCzasu.m".

### Funkcja: "*symKonserwatory.m*":

Definicja funkcji:

|  |
| --- |
| function [ czasPracyUkladu ] = symKonserwatory( wektorCzesci )  rozkladKonserwatorow = [10 20];  .  .  . |

1. Nagłówek funkcji "symKonserwatory.m".

Opis:

Jak widać na powyższym listingu funkcja "*symKonserwatory*" przyjmuje 1 parametr. Jest nim wektor części zamiennych który reprezentuje magazyn. Wewnątrz funkcji określamy skalę oraz kształt dla rozkładu Weibull'a z którego będzie generowany czas potrzebny na wymianę uszkodzonego elementu. Funkcja działa do wyczerpania się części zamiennych. Po wykonaniu funkcji zwracany jest czas pracy układu. Funkcja odczytuje z pliku rozkłady dla poszczególnych elementów, następnie wg rozkładów generuje czas życia elementu. Wynikiem działania w/w funkcji jest również historia życia układu. Zapisane są w niej czas, oraz aktualne stany dla poszczególnych elementów.

Opis możliwych stanów:

\* 0 - element jest sprawny.

\* 1 - element jest uszkodzony podczas naprawy.

\* 2 - element czeka na naprawę, brak wolnych

Przykładowa historia wygenerowana za pomocą w/w funkcji dla 4 elementów:

|  |
| --- |
| 31.95 0 1 0 0  37.31 0 1 0 2  41.07 0 0 0 1  50.34 0 0 0 0  58.64 0 1 0 0  68.05 0 0 0 0  70.40 0 0 1 0  80.39 0 0 0 0  91.37 0 0 0 1  100.80 0 0 0 0  108.60 0 1 0 0  111.04 2 1 0 0  119.05 1 0 0 0 |

1. Przykład zwracanej historii z funkcji "symKonserwatory".

## Przedstawienie symulatora:

### Listing kodu symulatora:

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 | budzet = 100; % Budżet  n = 4; % Ilość el. Zgoda z plikiem  kosztyEl = [15 10 11 14]; % Koszt elementow    % wygenerowanie macierzy wektorow  macierzWektorow = generujWektory(kosztyEl, budzet);  tmp = size(macierzWektorow); % odczytanie ilości  iloscW = tmp(1); % ilość potencjalnych wektorów    for a=1 :20 % Przeprowadzenie symulacji 20x    clear wynik;  wynik = 0;  % przeprowadzanie symulacji  for i=1 :iloscW  wynik(i,1) = symulacjaCzasu(macierzWektorow(i,:),1);  wynik(i,2) = i;  end;    i = max(wynik(:,1)); % wybranie najdłuższego czasu  % wyszukanie wektora odpowiadającego max czas.  for k=1 :iloscW  if(wynik(k,1) == i)  break;  end;  end;  macWyniki(a,:) = macierzWektorow(wynik(k,2),:);  end;  % wyszukanie najczęściej powtarzającego się wektora  maxWektor = mode(macWyniki);    % Główna symulacja przeprowadzona 1000x  for a=1 : 1000  wynikWektor(a) = symKonserwatory(maxWektor);  end;    % Wygenerowanie histogramu  hist(wynikWektor);  % Zapis wyników do pliku  plik = fopen('czasySymulacji.txt', 'w');  for j=1 :a  fprintf(plik, '%6.2f', wynikWektor(j));  fprintf(plik, '\n');  end;  fclose(plik); |

1. Kod głównej symulacji: "symulator.m".

### Opis działania:

Powyższy listing zaczyna się od zmiennych które przyjmują następujące wartości. **Budżet** - koszt do wydania na części zamienne, **n** - ilość elementów w symulacji, **kosztyEl** - wektor który zawiera cenę poszczególnych elementów. W linii 7 wywołujemy funkcję generującą wszystkie potencjalne wektory ilości części zamiennych.   
 Następnie w liniach (11 - 35) przeprowadzamy 20 symulacji na każdym wygenerowanym wektorze danych w celu wyznaczenia najbardziej optymalnego wektora części zamiennych. Ma to na celu wykrycie optymalnego wykorzystania budżetu na części zamienne. Jak widać w liniach (17 - 21) przeprowadzana jest symulacja czasu, a następnie linie (24 - 30) odpowiadają za wyznaczenie wektora dla którego symulacja była najdłuższa.  
 W linii (35) odczytujemy znaleziony najbardziej optymalny wektor aby wykorzystać do w kolejnej symulacji.

Linie (37 - 41) odpowiadają za przeprowadzenie pełnej symulacji z uwzględnieniem konserwatorów. Czasy zapisywane są do wektora wynikowego "*wynikWektor*" w zapisania ich do pliku aby można było przeprowadzić dalszą analizę w innym programie. Symulacja jest przeprowadzana 1000x aby była dokładniejsza. Następnie w linii (43) generujemy histogram z przeprowadzonej symulacji.

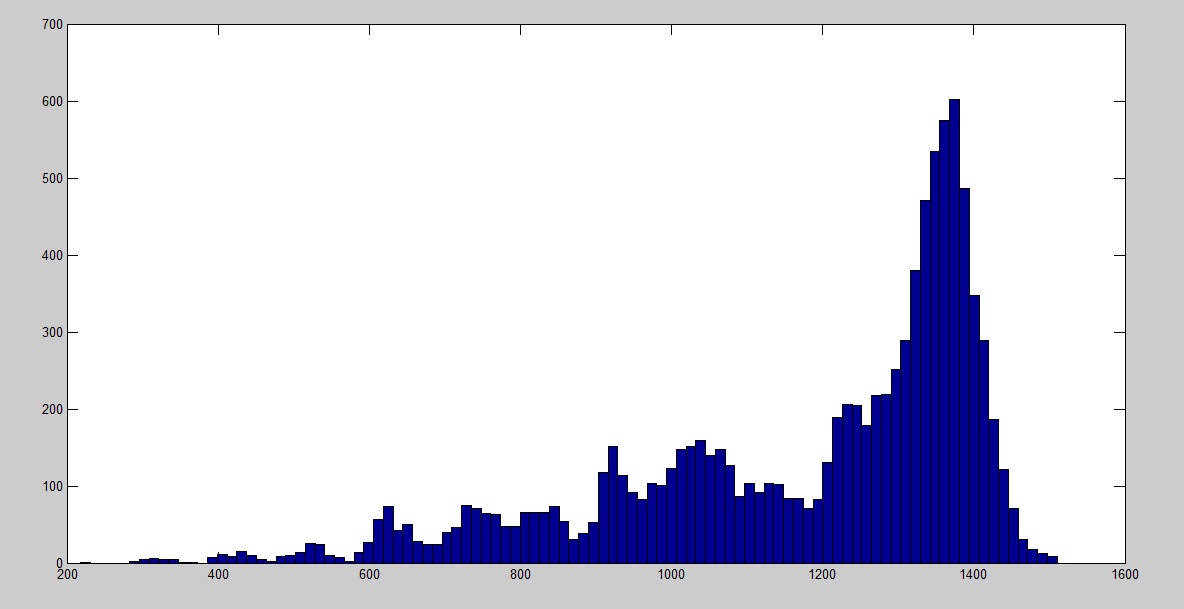
Zapis wygenerowanych danych odbywa się w liniach (47 - 54) do pliku o nazwie "*czasySymulacji.txt*".

### Przykład wygenerowanych czasów symulacji:

|  |
| --- |
| 1188.52  1319.08  776.15  1410.71  1213.79  1123.60  1259.00  1418.87  1328.93  928.36  910.32  1414.59  1316.01  799.27  1222.88  752.05  1002.58  919.45  969.43  1342.00  840.63  1221.04  1020.47 |

1. Przykład danych z pliku "czasySymulacji.txt".

### Wykres histogramu dla danych które będą w dalszej części analizowane:



1. Wykres histogramu wygenerowany w programie Matlab.

# Analiza danych w programie SAS:

# Literatura:

1. www.pl.wikipedia.org

2. www.pl.wikibooks.org

3. www.bogdro.ciki.me/linux/index.php

4. http://cs.fit.edu/~mmahoney/cse3101/float.html