ENGR 541

Project: Reliability of a Consecutive ***k***-out-of-***r***-from-***n*** System

Algorithm & Visualization

Zexi Liu
zexi@temple.edu

27th November 2006

1 Introduction

The purpose of this project was to simulate and visualize the reliability calculation of a consecutive ***k***-out-of-***r***-from-***n*** system with multiple failure criteria. In the model I chose, the system consists of ***n*** linearly ordered elements, and fails iff in any group of $\boldsymbol{r_1, r_2, \ldots, r_H}$ consecutive elements less than $\boldsymbol{k_1, k_2, \ldots, k_H}$ elements respectively are in working state. The algorithm for system reliability evaluation is based on an extended universal moment generating function.

Index Terms – Consecutive ***k***-out-of-***r***-from-***n*** system, multiple failure criteria, universal moment generating function

NOTATION

n	number of elements in the system
k, r	vectors representing failure criteria in the system
H	Number of failure criteria in the system
r_H	maximal element of vector ***r***
p	probability that the system element is in working state
R	system reliability

In the multi-criteria linear consecutive ***k***-out-of-***r***-from-***n*** system model, $\boldsymbol{k} = \{k_i \mid 1 \leqslant i \leqslant H\}$ & $\boldsymbol{r} = \{r_i \mid 1 \leqslant i \leqslant H\}$ are integer vectors such that $r_i \leq n$ & $k_i \leq r_i$. The system fails if at least one group of r_i consecutive elements exists in which less than k_i elements are in working condition for any $1 \leqslant i \leqslant H$.

All n ordered elements in the system are identical and mutually s-independent. The state of each system element j is characterized by a random binary variable X_j, where $X_j = 0$ corresponds to failure, and $X_j = 1$ corresponds to the working state. The probability of being in the working state is $p = \Pr\{X_j = 1\}$. The system fails if for any $1 \leqslant i \leqslant H$ at least one of the sums $\sum_{j=1}^{r_i} X_j,\ \sum_{j=2}^{r_i+1} X_j,\ \ldots,\ \sum_{j=n-r_i+1}^{n} X_j$ is less than k_i. The system reliability is, therefore, defined as follows: $R = \Pr\{\sum_{j=h}^{h+r_i-1} X_j \geq k_i \mid 1 \leq h \leq n - r_i + 1,\ 1 \leq i \leq H\}$. It can be easily seen that if there

exists an I for which $k_i = r_i$, the system becomes a series system, and its reliability is equal to p^n. Indeed, the condition $k_i = r_i$ means that any set of r_i consecutive elements should not contain failure elements.

2 Reliability Evaluation Algorithm

Note that the system considered contains exactly $\boldsymbol{n} - \boldsymbol{r} + 1$ groups of r consecutive elements, and each element can belong to no more than $\boldsymbol{r}$ such groups. To obtain the vector-u-functions corresponding to all the groups of $\boldsymbol{r}$ consecutive elements, the following procedure is introduced:

a) Define the vector-u-function $U_{1\text{-}r,\,r}(z)$ as follows

$U_{1\text{-}r,\,r}(z) = z^{y_0}$, where the vector y_0 consists $\boldsymbol{r}$ zeros.

b) Define the following shift operator Ψ over vector-u-function $U_{h,\,r}(z)$:

$$\Psi\left(U_{h,r}(z)\right) = \Psi\left(\sum_{m=1}^{2^r} Q_m z^{y_m}\right) = p\sum_{m=1}^{2^r} Q_m z^{\varphi(y_m,1)} + (1-p)\sum_{m=1}^{2^r} Q_m z^{\varphi(y_m,0)}$$, where operator φ(y, x) over an arbitrary vector y & value x shifts all the vector elements one position left: y ($\boldsymbol{j}$-1) = y ($\boldsymbol{j}$) for $1 < \boldsymbol{j} \leqslant$ r, and assigns the value x to the last element of y: y(r) = x. The first element of vector y disappears after applying the operator. The operator φ removes the state of the first element of the group, and adds the state of the next (not considered yet) element to the group preserving the order of elements belonging to the group. Therefore, applying the operator Ψ over the vector-u-function representing the state distribution of the h^{th} group, one obtains the vector-u-function representing the state distribution of the $(h+1)^{\text{th}}$ group. Using the operator Ψ in sequence as follows:

$U_{j+1\text{-}r,\,r}(z) = \Psi\,(U_{j\text{-}r,\,r}(z))$, for $\boldsymbol{j} = 1, \ldots, \boldsymbol{n}$ one obtains vector-u-functions for all of the possible groups of $\boldsymbol{r}$ consecutive elements: $U_{1,\,r}(z), \ldots, U_{n\text{-}r+1,\,r}(z)$.

c) Define the operator δ_r over vector-u-function $U_{h,\,r}(z)$:

$$\delta_r\left(U_{h,r}(z)\right) = \sum_{m=1}^{2^r} Q_m 1\left(\sum_{j=1}^{r} y_m(j) < k\right) \quad where \quad 1(x) = \begin{cases} 1, & x\,is\ true \\ 0, & x\,is\ false \end{cases}$$

d) Apply the operator δ_r to the vector-u-function $U_{h,\,r}(z)$, we can obtain the probability that the group consisting of r elements $\boldsymbol{h} + \boldsymbol{r_H} - \boldsymbol{r}, \ldots, \boldsymbol{h} + \boldsymbol{r_H} - 1$ fails. Note that if for some combination of elements states this group fails, the entire system fails independently of the states of the elements

that do not belong to this group. Therefore the terms corresponding to the group failure can be removed from the vector-u-function $U_{h,\,r}$ (z) because they should not participate in determining further state combinations which cause system failures. This consideration lies at the base of the following algorithm, which evaluates the system reliability using an enumerative technique to obtain all of the possible element state combinations leading to the system's failure.

The following algorithm finds the reliability of a consecutive ***k***-out-of-***r***-from-***n*** system:

1) Initialization:

   $\boldsymbol{F} = 0$; $U_{1-r_H,r_H}(z) = z^{y_0}$ (y_0 consists of $\boldsymbol{r_H}$ zeros).

2) Main loop:

   Repeat the following for $\boldsymbol{j} = 1, \ldots, \boldsymbol{n}$:

   2.1. Obtain $U_{j+1-r_H,r_H}(z) = \Psi\left(U_{j-r_H,r_H}(z)\right)$, and collect like terms in the obtained vector-u-function.

   2.2. For $\boldsymbol{i} = 1, \ldots, \boldsymbol{H}$ if $\boldsymbol{j} \geqslant \boldsymbol{r_i}$, add the value $\delta_{r_i}\left(U_{j+1-r_H,r_H}(z)\right)$ to $\boldsymbol{F}$, and remove all of the terms with $\sum_{s=r_H-r_i+1}^{r_H} y(s) < k_i$ from $U_{j+1-r_H,r_H}(z)$.

3) Obtain the system reliability as $\boldsymbol{R} = 1 - \boldsymbol{F}$. Alternatively, the system reliability can be obtained as the sum of the coefficients of the last vector-u-function $U_{n+1-r_H,r_H}(z)$

3 Matlab Calculation

In the Matlab program, the above algorithm should be changed a little bit. However, the basic idea is the same.

I) Define two initial matrix U_z and U_p as follows:

   U_z = [z_1 z_2 … z_r], $z_1 = 0$ $z_2 = 0$ … $z_r = 0$ consists of $\boldsymbol{r_H}$ zeros.

   U_p = [$1\text{-}p_1$ p_1], p_1 is the probability that the 1st element is in working state.

II) Define the following shift function ***shift ( )*** and ***shiftp ( )*** over U_z and U_p as follows:

shift (U_z) = $\begin{bmatrix} [U\_z]_{i\times j} & [0]_{i\times 1} \\ [U\_z]_{i\times j} & [1]_{i\times 1} \end{bmatrix} = \begin{bmatrix} z_2 & z_3 & \dots & z_r & 0 \\ z_2 & z_3 & \dots & z_r & 1 \end{bmatrix}$, note that the first column was deleted.

shiftp (U_p) = $\begin{bmatrix} [U\_p]_{i\times j} & [1-p_2]_{j\times 1} \\ [U\_p]_{i\times j} & [p_2]_{j\times 1} \end{bmatrix} = \begin{bmatrix} 1-p_1 & p_1 & 1-p_2 \\ 1-p_1 & p_1 & p_2 \end{bmatrix}$, 1-$p_2$ is corresponding to ***0*** in the above matrix, p_2 is corresponding to ***1*** in the above matrix.

III) Define the following removal function ***truncate (***U_z, U_p, $\boldsymbol{k_i}$, $\boldsymbol{r_i}$, $\boldsymbol{r_H}$ ***)*** as follows:

If $\sum_{r=r_H+r_i+1}^{j} U\_z_{m,r} < k_i$, then remove the $\boldsymbol{m^{th}}$ row of both matrix U_z and U_p. ***m*** = 1, … , number of rows of U_z. ***j*** is the number of columns of U_z.

IV) Repeat the main loop ***n*** times.

V) The system reliability can be obtained from the following equation:

$\text{Reliability} = \sum_{m=1}^{i} \left(\prod_{k=1}^{j} U\_p_{m,k} \right)$, where ***i***, ***j*** is the size of $U\_p_{i\times j}$

4 Simulation & Visualization

Let's take the (k_1, k_2, k_3)-out-of-(5, 7, 8)-from-9 system as an example.

1) $P = [p_1\ p_2\ p_3\ \dots\ p_9]$ are constant i.e. $p_1 = p_2 = \dots = p_9 = \text{Constant}$.

   Let Constant = 0.8

   Plot 3-D bar charts as follows.

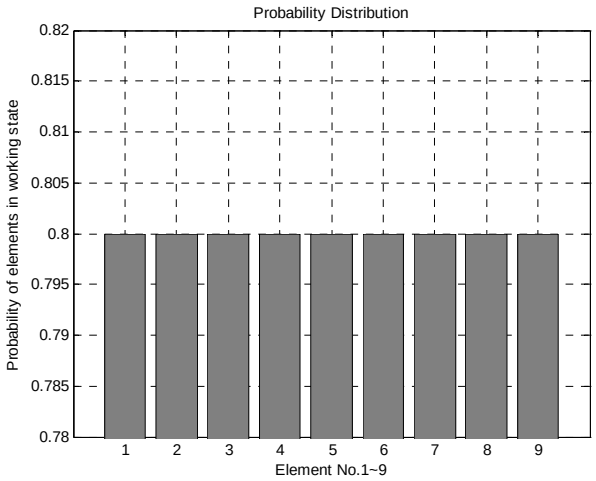

Fig. 1. Uniform Distribution of system elements

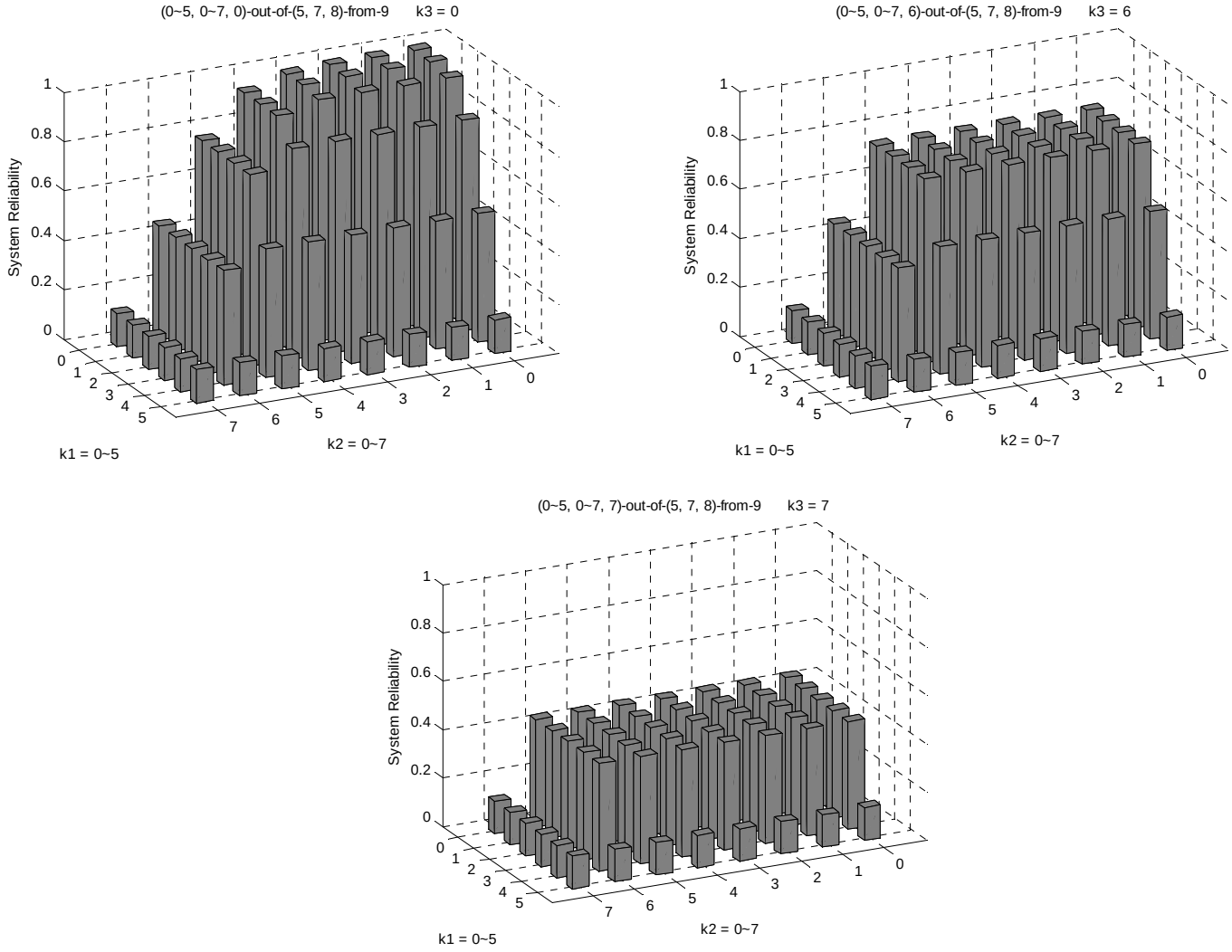

Fig. 2. Reliability of (k1, k2, k3)-out-of-(5, 7, 8)-from-9 system as function of k1, k2, k3.

2) P = [p_1 p_2 p_3 … p_n] are linearly increasing. n = 9, p_1 = 0.6, p_n = 0.9,

$$p_i = \frac{(p_n - p_1)\times(i-1)}{n-1} + p_1$$

Plot 3-D bar charts as follows.

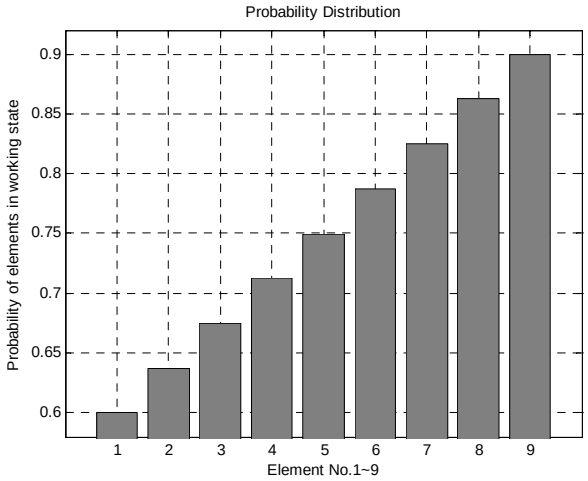

Fig. 3. Probability Distribution of system elements

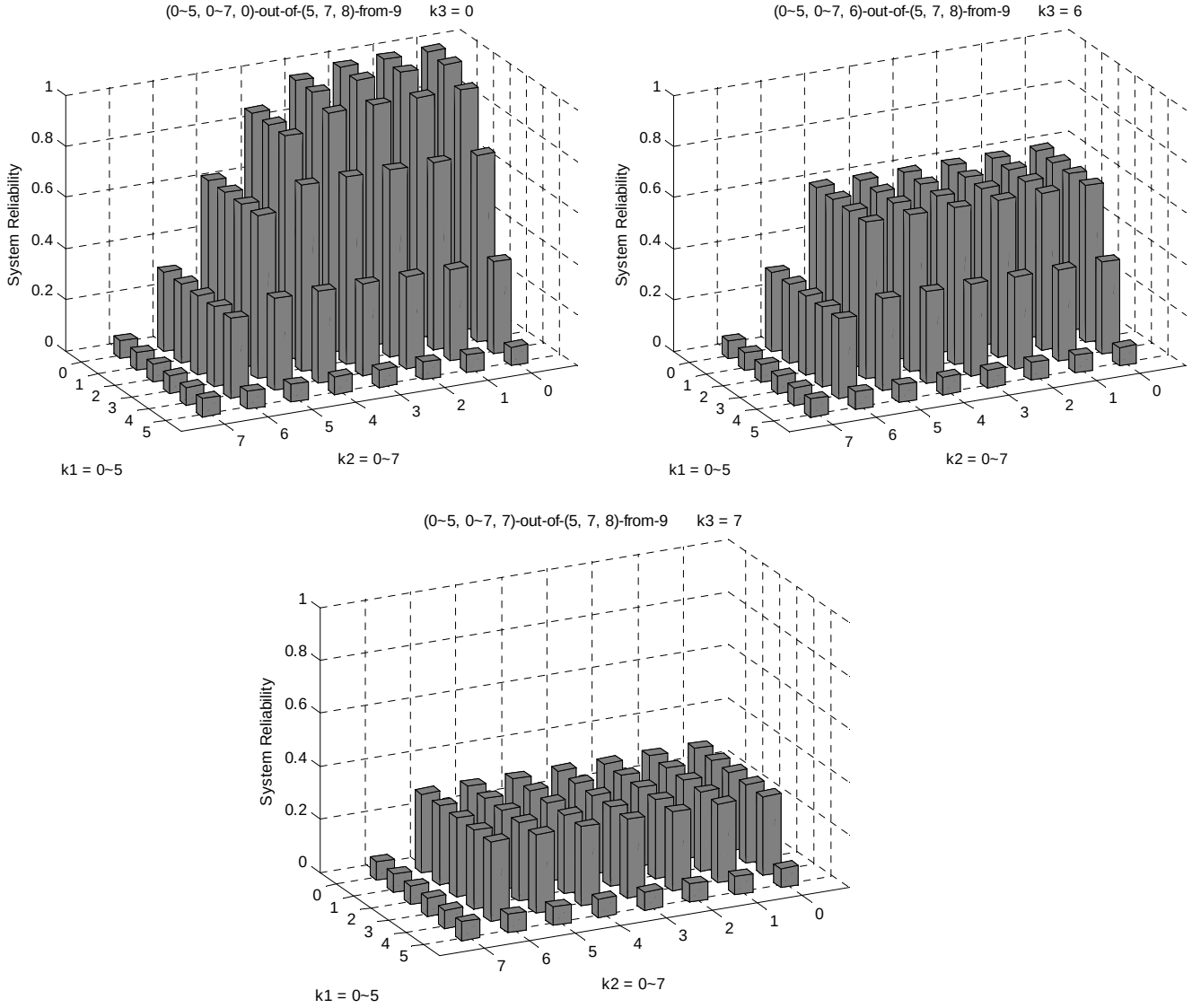

Fig. 4. Reliability of (k1, k2, k3)-out-of-(5, 7, 8)-from-9 system as function of k1, k2, k3.

3) P = [p_1 p_2 p_3 … p_n] are exponentially increasing. n = 9, i = 1, 2, … , 9, $\lambda = 0.4$

$$p_i = \lambda \times e^{-\lambda \times i}.$$

Plot 3-D bar charts as follows.

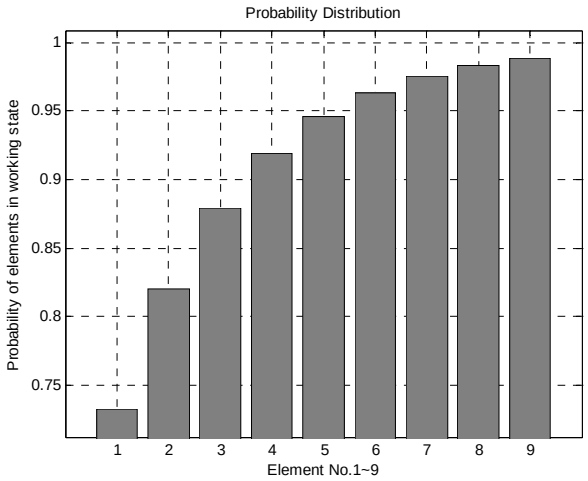

Fig. 5. Exponential Distribution

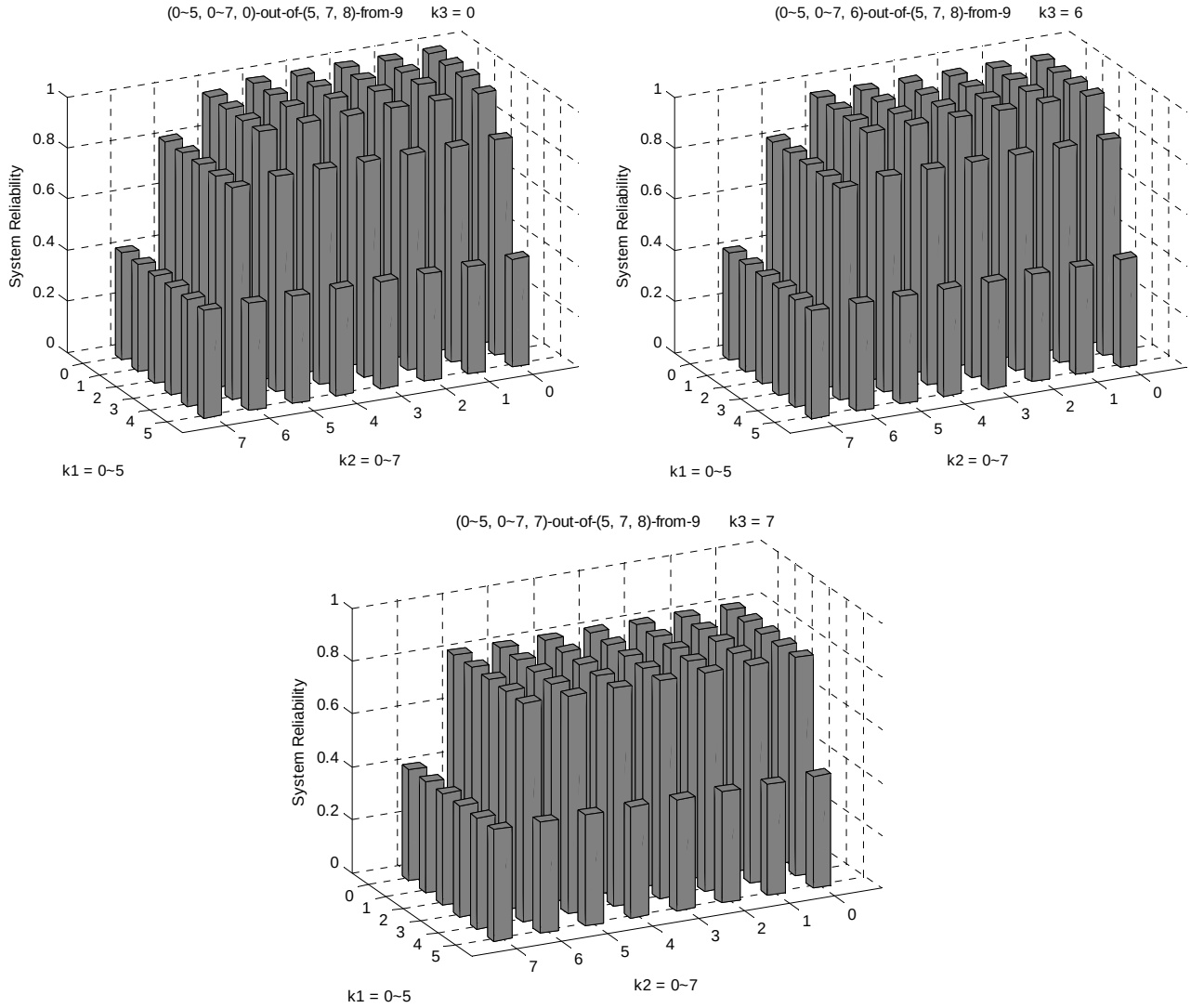

Fig. 6. Reliability of (k1, k2, k3)-out-of-(5, 7, 8)-from-9 system as function of k1, k2, k3.

4) P = [p_1 p_2 p_3 … p_n] are Gaussian distribution. n = 9, i = 1, 2, … , 9, μ = (n+1)/2, σ = 2.5

$$p_i = 3.5 \times \frac{1}{\sqrt{2 \times \pi \times \sigma}} e^{\frac{-(i-\mu)^2}{2 \times \sigma^2}} .$$

Plot 3-D bar charts as follows.

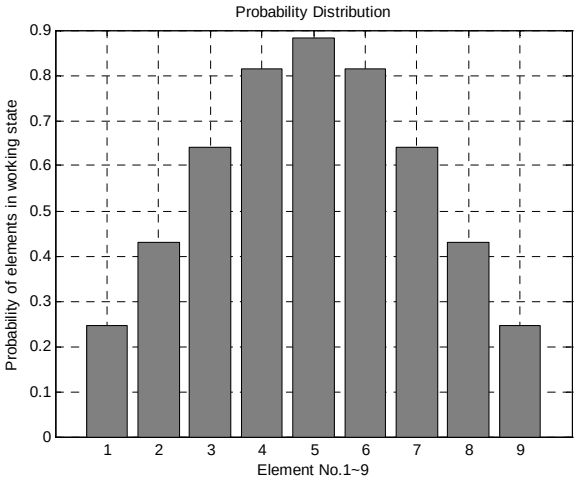

Fig. 7. Gaussian Distribution

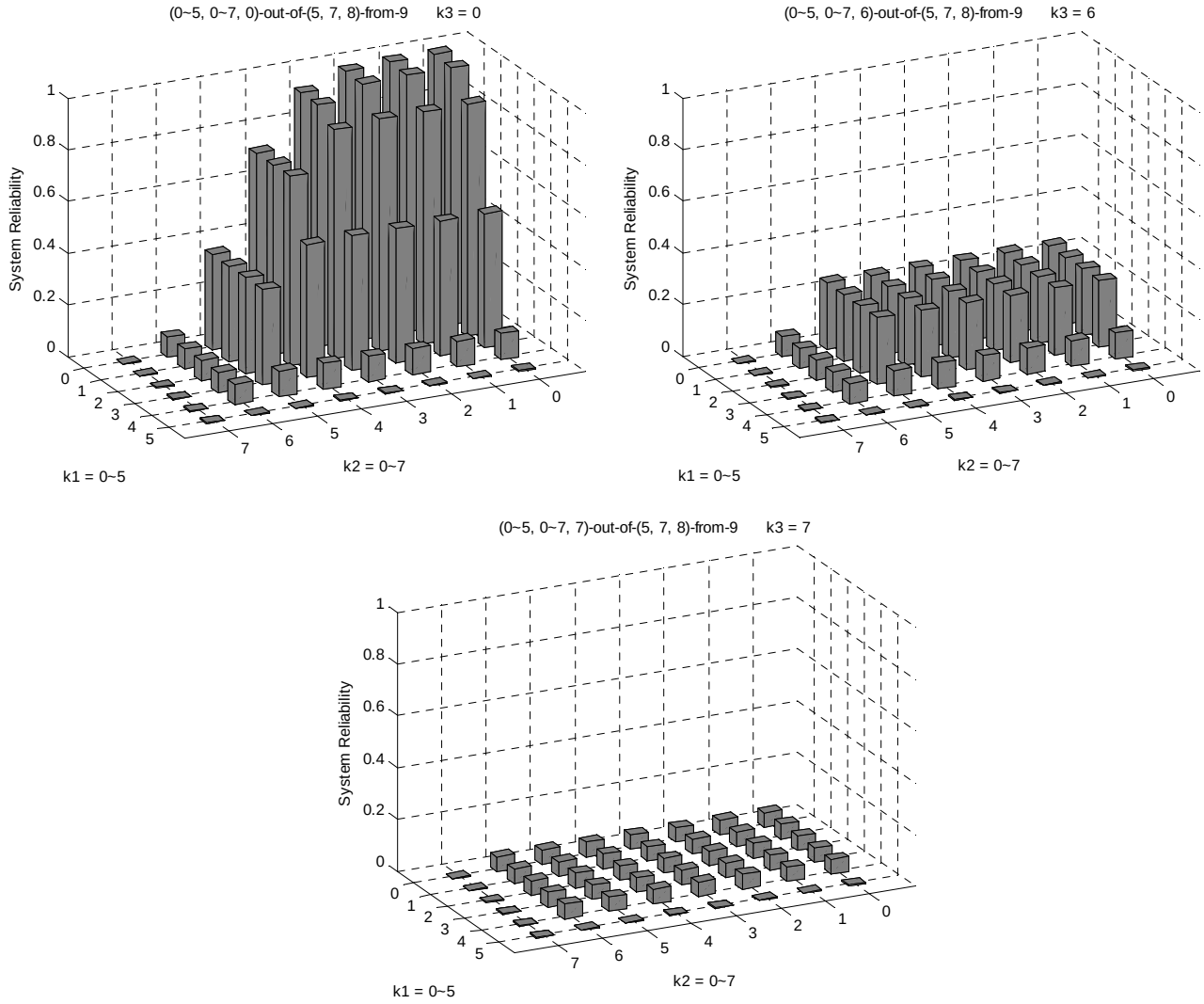

Fig. 8. Reliability of (k1, k2, k3)-out-of-(5, 7, 8)-from-9 system as function of k1, k2, k3.

5) P = [p_1 p_2 p_3 … p_n] are not assigned any value.

K = [1 2 7], R = [5 7 8], n = 9 ($K_3 \neq R_3$)

P = [p1, p2, p3, p4, p5, p6, p7, p8, p9]

System Reliability
=(1-p1)*p2*p3*p4*p5*p6*p7*p8*(1-p9)+p1*p2*p3*p4*p5*p6*p7*p8*(1-p9)+p1*p2*p3*p4*p5*p6*p7*(1-p8)*p9+p1*p2*p3*p4*p5*p6*(1-p7)*p8*p9+p1*p2*p3*p4*p5*(1-p6)*p7*p8*p9+p1*p2*p3*p4*(1-p5)*p6*p7*p8*p9+p1*p2*p3*(1-p4)*p5*p6*p7*p8*p9+p1*p2*(1-p3)*p4*p5*p6*p7*p8*p9+p1*(1-p2)*p3*p4*p5*p6*p7*p8*p9+(1-p1)*p2*p3*p4*p5*p6*p7*p8*p9+p1*p2*p3*p4*p5*p6*p7*p8*p9

$$U\_p = \begin{bmatrix} 1-p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & 1-p_9 \\ p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & 1-p_9 \\ p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & 1-p_8 & p_9 \\ p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & 1-p_7 & p_8 & p_9 \\ p_1 & p_2 & p_3 & p_4 & p_5 & 1-p_6 & p_7 & p_8 & p_9 \\ p_1 & p_2 & p_3 & p_4 & 1-p_5 & p_6 & p_7 & p_8 & p_9 \\ p_1 & p_2 & p_3 & 1-p_4 & p_5 & p_6 & p_7 & p_8 & p_9 \\ p_1 & p_2 & 1-p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 \\ p_1 & 1-p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 \\ 1-p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 \\ p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 \end{bmatrix}$$

K = [1 2 8], R = [5 7 8], n = 9 ($K_3 = R_3$, series system)

P = [p1, p2, p3, p4, p5, p6, p7, p8, p9]

System Reliability =p1*p2*p3*p4*p5*p6*p7*p8*p9

$$U\_p = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 \end{bmatrix}$$

5 Conclusion

The presented algorithm allows one to estimate the influence of the reliability of individual elements on the reliability of the entire system (the element reliability importance). Though the reliability evaluation algorithm was provided, the usage of method is limited by its computational complexity. Obviously, this evaluation method is in fact an enumerative technique. As a result, if the number of system elements is large or the criteria r_1 is large, it will take extremely long period of time and RAM space for Matlab to calculate the system reliability. Simply speaking, if r_1 is larger than 20 (of course n will be also larger than 20), the program can not get any results in a bearable long period of time. For instance, a consecutive (1, 2, 3)-out-of-(5, 7, 8)-from-9 system with p = 0.8, Matlab use 0.343 seconds to get the result. Table 1 shows that when n is increasing, the time is getting larger and larger.

n	**Time** (sec)	**Size of U_p** (row × column)	**System Reliability**
9	0.343	410×9	99.736 %
10	0.625	784×10	99.663 %
11	1.125	1501×11	99.591 %
12	2.078	2875×12	99.519 %
13	4.703	5506×13	99.448 %
14	14.641	10537×14	99.376 %
15	55.953	20148×15	99.305 %
16	114.687	38491×16	99.234 %
17	202.500	73556×17	99.162 %

6 Bibliography

[1] Levitin, G. "Consecutive k-out-of-r-from-n system with multiple failure criteria," IEEE TRANSACTIONS ON RELIABILITY, vol. 53, pp. 394-400, 2004.

[2] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, Fourth edition, 2002.

[3] H. Stark, J. W. Woods. *Probability and Random Processes with Applications to Signal Processing.* Prentice Hall, third edition, 2002.

Appendix A

Matlab Code

```
clear;
clc;

c_begin = clock;                % get system time

K = [1 2 8];                    % vectors representing failure criteria in the system
R = [5 7 8];                    % vectors representing failure criteria in the system
n = 9;                          % number of elements in the system;
type = 0;
P = define (n, type, 0.8);      % define p
MODE = 0;

switch MODE
    case 0
        [RELIABILITY U_p] = Reliability(P, K, R, n);
        if type > 0
            Ans = sprintf('System Reliability = %.3f %%', RELIABILITY * 100);
            disp(Ans);
        else
            disp(U_p);
            disp('System Reliability = ');
            disp(RELIABILITY);
        end
    case 1
        if type > 0
            Re = zeros(R(2), R(1));
        end
        for i = 1 : R(1) + 1
            for j = 1 : R(2) + 1
                K(1) = i - 1;
                K(2) = j - 1;
                Re(j, i) = Reliability(P, K, R, n);
            end
        end

        disp('Reliability Matrix = ');
        disp(Re);
        figure(2);
        bar3(Re, 0.4);
        colormap([0.5 0.5 0.5]);
        axis([0 R(1)+2 0 R(2)+2 0 1]);
        view(64, 20);

        XTickL = [0 : R(1)]';
        YTickL = [0 : R(2)]';

        set(gca, 'XTickLabel', XTickL);
        set(gca, 'YTickLabel', YTickL);
        Xlabel = sprintf('k1 = 0~%d', K(1));
        xlabel(Xlabel);
        Ylabel = sprintf('k2 = 0~%d', K(2));
        ylabel(Ylabel);
        zlabel('System Reliability');
```

```
        Title = sprintf('(0~%d, 0~%d, %d)-out-of-(%d, %d, %d)-from-%d          k3 = %d', K(1), K(2), K(3), R(1),
R(2), R(3), n, K(3));
        title(Title);
end

c_end = clock;
t = c_end(5) * 60 - c_begin(5) * 60 + c_end(6) - c_begin(6);
Time = sprintf('It took %.4f seconds to calculate.', t);
disp(Time);

function [RELIABILITY U]= Reliability(p, K, R, n)

    H = size(R, 2);
    U_z = zeros(1, R(H));
    U_p = [1 - p(1); p(1)];

    L1 = 1;
    while L1 <= n
        U_z = shift(U_z);
        if L1 > 1
            U_p = shiftp(U_p, p(L1));
        end
        for L2 = 1 : size(R, 2)
            if L1 >= R(L2)
                [U_z, U_p] = truncate(U_z, U_p, K(L2), R(L2), R(H));
            end
        end
        L1 = L1 + 1;
    end

    U = U_p;
    number_of_rows = size(U_p, 1);
    RELIABILITY = 0;
    for i = 1 : number_of_rows
        RELIABILITY = RELIABILITY + prod(U_p(i, :));
    end

function U = shift(U_z)

    number_of_rows = size(U_z,1);
    number_of_columns = size(U_z,2);

    k = number_of_rows;
    r = number_of_columns;

    U_z(:,1) = [ ];
    U_z(:,r) = zeros;

    for m = 1:k
        U_z(k+m,:) = de2bi(bi2de(U_z(m,:),'left-msb')+1,r,'left-msb');
    end

    U = U_z;

function U = shiftp(U_p, p)

    U_p = [U_p;U_p];
```

```
        number_of_rows = size(U_p,1);
        number_of_columns = size(U_p,2);

        k = number_of_rows;
        r = number_of_columns;

        U_p(1:(k/2),r+1) = 1-p;
        U_p((k/2+1):k,r+1) = p;

        U = U_p;

function [U1,U2] = truncate(U_z,U_p,ki,ri,rh)

        r = rh-ri+1;

        number_of_rows = size(U_z,1);
        number_of_columns = size(U_z,2);

        i = number_of_rows;
        j = number_of_columns;

        m = 1;
        ks = i;
        while m <= ks
                if sum(U_z(m,r:j)) < ki
                     U_z(m,:) = [ ];
                       U_p(m,:) = [ ];
                       ks = size(U_z,1);
                else
                       m = m+1;
                end
        end

        U1 = U_z;
        U2 = U_p;

function P = define (n, type, prob)
        if type == 0                                    % Symbol
                temp = 'p1';
                syms p1;
                p = [p1];
                for x = 1 : n
                       if x <10
                              temp(2) = num2str(x);
                              p(x) = sym(temp);
                       else
                              t = num2str(x);
                              temp(2) = t(1);
                              temp(3) = t(2);
                              p(x) = sym(temp);
                       end
                end
                disp('p = ');
                disp(p);
        elseif type == 1                                % Gaussian N
                sigma = 2.5;
                miu = (n+1) / 2;
                x = 1 : n;
```

```
        p = 3.5 * 1 / sqrt(2 * pi * sigma) * exp((-(x - miu).^2) / (2 * sigma ^ 2));
        figure(1);
        bar(x, p); grid on;
        colormap([0.5 0.5 0.5]);
        %axis([0 n + 1 min(p) - 0.02 max(p) + 0.02]);
        Xlabel = sprintf('Element No.1~%d', n);
        xlabel(Xlabel);
        ylabel('Probability of elements in working state');
        title('Probability Distribution');
elseif type == 2                        % Exponential
        lanmuda = 0.4;
        x = 1 : n;
        q = lanmuda * exp(-lanmuda.*x);
        p = 1- q;
        figure(1);
        bar(x, p); grid on;
        colormap([0.5 0.5 0.5]);
        axis([0 n + 1 min(p) - 0.02 max(p) + 0.02]);
        Xlabel = sprintf('Element No.1~%d', n);
        xlabel(Xlabel);
        ylabel('Probability of elements in working state');
        title('Probability Distribution');
elseif type == 3                        % Linear
        x = 1 : n;
        a = 0.6;
        b = 0.9;
        p = ((b - a) / (n - 1)) * (x - 1) + a;
        figure(1);
        bar(x, p); grid on;
        colormap([0.5 0.5 0.5]);
        axis([0 n + 1 min(p) - 0.02 max(p) + 0.02]);
        Xlabel = sprintf('Element No.1~%d', n);
        xlabel(Xlabel);
        ylabel('Probability of elements in working state');
        title('Probability Distribution');
elseif type == 4                        % Constant
        for x = 1 : n
                p(x) = prob;
        end
        figure(1);
        x = 1 : n;
        bar(x, p); grid on;
        colormap([0.5 0.5 0.5]);
        axis([0 n + 1 min(p) - 0.02 max(p) + 0.02]);
        Xlabel = sprintf('Element No.1~%d', n);
        xlabel(Xlabel);
        ylabel('Probability of elements in working state');
        title('Probability Distribution');
end

P = p;
```