T6A044

"Development Paper – Staggered Proof Testing Coefficients"

1 Contents

2 Revision History

Version	Date	Author	Comments
0.1	28/01/2022	RM	Draft release for public comment.

3 Introduction / Foreword

This development paper is not yet complete or fully reviewed by members of The 61508 Association. This paper has been published to elicit further comment and input on the comments within the paper from both members of the Association and any other interested party from outside the Association. Please send all comments on this paper to the Association coordinator via the email: info@61508.org.

Message from the initial author:
This document is to support *Effects of Proof Testing* in the topic area of Staggered Proof Testing.

This is where we consider the failure of a subset with the understanding that the probability of failure depends on the relative positions in the testing cycle of the items of interest. The only way I have found is to consider all the possible cases and average them.

This results in a program which is run once to generate a set of coefficients for use in the Calculation Engine.

This document represents my notes on the algorithm development use before creating Excel VBA routines to generate the coefficients required by *Staggered Proof Testing.* It is nothing more than an aide memoire.

The final result is used in *Fault Tolerant Systems* which is the document which lies behind the so-called *PFD Calculator* which is an Excel based reliability assessment spreadsheet with a reliability calculator built in. The VBA used to develop the coefficients is contained within the same Excel file.

Ray Martin

4 Executive Summary

This paper is not yet complete; therefore, an executive summary is not yet required. This paper has been published in this form to elicit comments on the content.

5 Terminology

f	General term for 'fault tolerance' – i.e. for simple redundancy, the number of failed devices a system can tolerate and still perform its function. Note: r is the general term for the number of survivors required for a system to perform its function.
F	Probability of failure (normally a function of time). Note: this has the same meaning at PFD (probability of failure on demand).
MT	Mission Time (for use with residual failures)
MTBF	Mean time before failure. MTBF = $1/\lambda$ (for constant λ)
MTTR	Mean time to restore.
PFD	Probability of failure on demand. Notes: · this has the same meaning as F (probability of failure). · This is sometimes used in the text as shorthand for PFD_{AV}.
PFD_{AV}	Time average of PFD.
PFD_D	PFD for diagnosed failures for single channel / device. $PFD_D = PFD_D^1 = \left((1-\beta_D)\lambda dd.MTTR\right)$
PFD_R	PFD for residual failures for single channel / device. $PFD_R = PFD_R^1 = \left(\frac{(1-\beta_R)\lambda drMT}{2}\right)$
PFD_U	PFD for undiagnosed failures for single channel / device. $PFD_U = PFD_U^1 = \left(\frac{(1-\beta_U)\lambda duT}{2}\right)$
PFD_D^k	PFD for diagnosed failures for k channels / devices $PFD_D^k = (PFD_D)^k$
PFD_R^k	PFD for residual failures for k channels / devices $PFD_R^k \neq (PFD_R)^k$ due to test regime
PFD_U^k	PFD for undiagnosed failures for k channels / devices $PFD_U^k \neq (PFD_U)^k$ due to replacement regime
PFD^N	PFD rolled up for all failures for N channels / devices (including common causes)
R	Probability of survival (normally a function of time).
s	Used as a suffix to represent attributes of a system. E.g. F_S is used to represent probability of system failure.
T	Proof test interval.

β	Beta factor – general term for fraction of failures which affect all channels / devices.
β_D	Beta factor specific to diagnosed failures
β_D	Beta factor specific to residual failures
β_U	Beta factor specific to undiagnosed failures
λ	General term for underlying failure rate – a function of time that represents the failure rate 'given that there is no current failure'. This paper assumes it is a constant in time. Note: this is not the same as $\dot{F}(t)$ (which is the failure rate not assuming current survival).
λ_d	General term for diagnosed failure rate – i.e. failure that is automatically revealed.
λ_u	General term for undiagnosed failure rate.
λ_{dd}	Dangerous diagnosed failure rate.
λ_{dr}	Dangerous residual failure rate – i.e. dangerous failure rate that is not automatically revealed or revealed by periodic proof test.
λ_{du}	Dangerous undiagnosed failure rate.

6 Complex redundancy with staggered proof testing

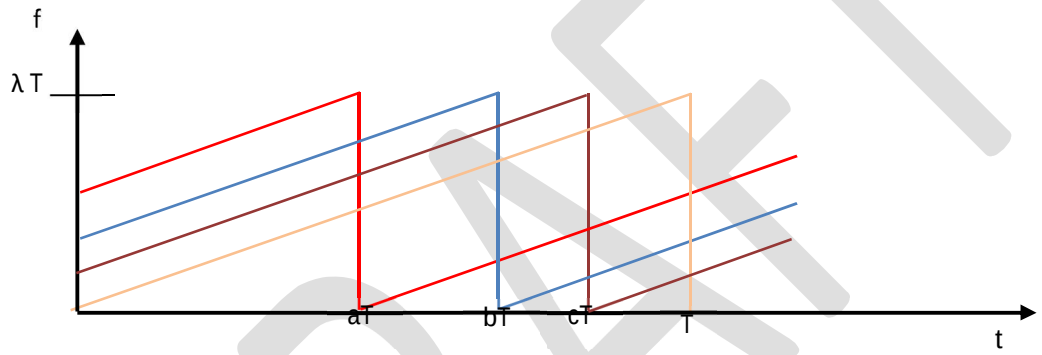
Here, we consider the case of complex redundancy where faults are undiagnosed (i.e. found only in proof testing) and where the system fails if M out of N items fail.

The purpose of this paper is to generate the algorithms that will generate a 10 by 10 table of numbers representing the test factors to be applied. The test factors are multiplying factors for M out of N failures where these factors are multiplied by $PFD_{1oo1}{}^{N}$ to find the $PFD_{AV.}$

The mathematics is dealt with in more detail in Staggered Proof Testing but the cases of 4 failures and M failures are repeated here for information.

7 Failure of 4 Items

Consider the following graph



The joint probability of failure is given by

$F(t) = \lambda t(\lambda t + (1-a)\lambda T)(\lambda t + (1-b)\lambda T)(\lambda t + (1-c)\lambda T)$ [0,aT]

$F(t) = \lambda t(\lambda t + (-a)\lambda T)(\lambda t + (1-b)\lambda T)(\lambda t + (1-c)\lambda T)$
[aT,bT]

$F(t) = \lambda t(\lambda t + (-a)\lambda T)(\lambda t + (-b)\lambda T)(\lambda t + (1-c)\lambda T)$
[bT,cT]

$F(t) = \lambda t(\lambda t + (-a)\lambda T)(\lambda t + (-b)\lambda T)(\lambda t + (-c)\lambda T)$ [cT,T]

We write this as: $F(t) = x(x+A)(x+B)(x+C)$

Where

$x = \lambda t;$

$A = (1-a)\lambda T \; [t < aT]$
$A = (-a)\lambda T \; [t \geq aT]$

B= $(1-b)\lambda T\ [t < bT]$
B= $(-b)\lambda T\ [t \geq bT]$
$C = (1-c)\lambda T\ [t < cT]$
$C = (-c)\lambda T\ [t \geq cT]$

Expanding the expression for F(t)

$$F(t) = (x^2 + Ax)(x - B)(x - C)$$

$$F(t) = (x^3 + (A + B)x^2)(x - C)$$

$$F(t) = x^4 + (A + B + C)x^3 + (AB + AC + BC)x^2 + ABCx$$

Note: The coefficients for powers of x (other than the first) are:

- the sum of all the solos, then
- the sum of all the pairs, then
- the sum of all the triples

This pattern is repeated for greater powers.

So, for 4 failures with staggered proof testing:

$$F(t) = \lambda^4 t^4 + (A + B + C)\lambda^3 t^3 + (AB + AC + BC)\lambda^2 t^2 + ABC\lambda t$$

$$F_{AV} = \frac{1}{T}\left[\lambda^4 \frac{t^5}{5} + (A + B + C)\lambda^3 \frac{t^4}{4} + (AB + AC + BC)\lambda^2 \frac{t^3}{3} + ABC\lambda \frac{t^2}{2}\right]_0^{aT}$$
$$+ \frac{1}{T}\left[\lambda^4 \frac{t^5}{5} + (A + B + C)\lambda^3 \frac{t^4}{4} + (AB + AC + BC)\lambda^2 \frac{t^3}{3} + ABC\lambda \frac{t^2}{2}\right]_{aT}^{bT}$$
$$+ \frac{1}{T}\left[\lambda^4 \frac{t^5}{5} + (A + B + C)\lambda^3 \frac{t^4}{4} + (AB + AC + BC)\lambda^2 \frac{t^3}{3} + ABC\lambda \frac{t^2}{2}\right]_{bT}^{cT}$$
$$+ \frac{1}{T}\left[\lambda^4 \frac{t^5}{5} + (A + B + C)\lambda^3 \frac{t^4}{4} + (AB + AC + BC)\lambda^2 \frac{t^3}{3} + ABC\lambda \frac{t^2}{2}\right]_{cT}^{T}$$

Where

$A = (1-a)\lambda T\ [t < aT]$
$A = (-a)\lambda T\ [t \geq aT]$

B= $(1-b)\lambda T$ $[t < bT]$
B= $(-b)\lambda T$ $[t \geq bT]$

$C = (1-c)\lambda T$ $[t < cT]$
$C = (-c)\lambda T$ $[t \geq cT]$

We could simplify this whole expression by replacing as follows:

$A' = A\lambda T$

$B' = B\lambda T$

$C' = C\lambda T$

Then, for 4 failures with staggered proof testing:

$$F(t) = \lambda^4 t^4 + (A+B+C)\lambda^3 t^3 + (AB+AC+BC)\lambda^2 t^2 + ABC\lambda t$$

$$F_{AV} = \lambda^4 T^4 \left[\frac{x^5}{5} + (A'+B'+C')\frac{x^4}{4} + (A'B'+A'C+B'C')\frac{x^3}{3} + A'B'C'\frac{x^2}{2}\right]_0^a$$

$$+\lambda^4 T^4 \left[\frac{x^5}{5} + (A'+B'+C')\frac{x^4}{4} + (A'B'+A'C+B'C')\frac{x^3}{3} + A'B'C'\frac{x^2}{2}\right]_a^b$$

$$+\lambda^4 T^4 \left[\frac{x^5}{5} + (A'+B'+C')\frac{x^4}{4} + (A'B'+A'C+B'C')\frac{x^3}{3} + A'B'C'\frac{x^2}{2}\right]_b^c$$

$$+\lambda^4 T^4 \left[\frac{x^5}{5} + (A'+B'+C')\frac{x^4}{4} + (A'B'+A'C+B'C')\frac{x^3}{3} + A'B'C'\frac{x^2}{2}\right]_c^1$$

Where

$A' = (1-a)$ $[x < a]$
$A' = (-a)$ $[x \geq a]$
$B' = (1-b)$ $[x < b]$
$B' = (-b)$ $[x \geq b]$
$C' = (1-c)$ $[x < c]$
$C' = (-c)$ $[x \geq c]$

8 Failure of N Items

It is possible to expand and find the general case from the above.

$$F_{AV} = \lambda^N T^N \left[\frac{x^{N+1}}{N+1} + (Sum\ of\ solos)\frac{x^N}{N} + (Sum\ of\ pairs)\frac{x^{N-1}}{N-1} + (Sum\ of\ triples)\ \frac{x^{N-2}}{N-2} + \ldots.\right]_0^a$$

$$+\lambda^N T^N \left[\frac{x^{N+1}}{N+1} + (Sum\ of\ solos)\frac{x^N}{N} + (Sum\ of\ pairs)\frac{x^{N-1}}{N-1} + (Sum\ of\ triples)\ \frac{x^{N-2}}{N-2} + \ldots.\right]_a^b$$

$$+ \ldots$$

$$+\lambda^N T^N \left[\frac{x^{N+1}}{N+1} + (Sum\ of\ solos)\frac{x^N}{N} + (Sum\ of\ pairs)\frac{x^{N-1}}{N-1} + (Sum\ of\ triples)\ \frac{x^{N-2}}{N-2} + \ldots.\right]_{..}^1$$

Where

$A' = (1-a)\ [x < a]$
$A' = (-a)\ [x \geq a]$
$B' = (1-b)\ [x < b]$
$B' = (-b)\ [x \geq b]$
$C' = (1-c)\ [x < c]$
$C' = (-c)\ [x \geq c]$

$$a < b < c < d \ldots \ldots$$

9 Finding the products of singles, pairs, triples etc in a set

To be able to evaluate the above, we need to be able to find the sum of the products of pairs, triples, quads etc.

9.1.1 Set of 3

Assume we have a vector of 3 elements: x_1, x_2 and x_3

The sum of the singles is: $x_1 + x_2 + x_3$

The sum of pairs is: $x_1x_2 + x_1x_3 + x_2x_3$

The sum of triples is: $x_1x_2x_3$

9.1.2 Set of 4

The sum of the singles is: $x_1 + x_2 + x_3 + x_4$

The sum of pair products is: $x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$

The sum of triple products is: $x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4$

The sum of quad products is: $x_1x_2x_3x_4$

The sum of the singles is: $x_1 + x_2 + x_3 + x_4$

The sum of pairs is: $x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4$

This is x_1 times (sum of singles in range x_2 to x_4) + x_2 times (sum of singles in range x_3 to x_4) + x_3 times (sum of singles in range x_4 to x_4)

The sum of triples is: $x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4$

This is x_1 times (sum of pairs in range x_2 to x_4) + x_2 times (sum of pairs in range x_3 to x_4)

The sum of quads is: $x_1x_2x_3x_4$

This is x_1 times (sum of triples in range x_2 to x_4)

9.1.3 **Set of 5**

The sum of the singles is: $x_1 + x_2 + x_3 + x_4 + x_5$

The sum of pair products is:

$x_1x_2 + x_1x_3 + x_1x_4 + x_1x_5 + x_2x_3 + x_2x_4 + x_2x_5 + x_3x_4 + x_3x_5 + x_4x_5$

This is x_1 times (sum of singles in range x_2 to x_5) + x_2 times (sum of singles in range x_3 to x_5) + x_3 times (sum of singles in range x_4 to x_5) + x_4 times (sum of singles in range x_5 to x_5)

The sum of triple products is:

$x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_3x_4 + x_1x_3x_5 + x_1x_4x_5 + x_2x_3x_4 + x_2x_3x_5 + x_2x_4x_5 + x_3x_4x_5$

This is x_1 times (sum of pairs in range x_2 to x_5) + x_2 times (sum of pairs in range x_3 to x_5) + x_3 times (sum of pairs in range x_4 to x_5)

The sum of quad products is:

$x_1x_2x_3x_4 + x_1x_2x_3x_5 + x_1x_2x_4x_5 + x_1x_3x_4x_5 + x_2x_3x_4x_5$

This is x_1 times (sum of triples in range x_2 to x_5) + x_2 times (sum of triples in range x_3 to x_5)

The sum of the quints is: $x_1x_2x_3x_4x_5$

This is x_1 times (sum of quads in range x_2 to x_5)

Note, the above in blue text can all be found algorithmically.

The sum of the singles is: $x_1 + x_2 + x_3 + x_4 + x_5$

The sum of pair products is: $x_1(x_2 + x_3 + x_4 + x_5)$

$+x_2(x_3 + x_4 + x_5)$

$+x_3(x_4 + x_5)$

$+x_4(x_5)$

The sum of triple products is: $x_1x_2(x_3 + x_4 + x_5)$

$+x_1x_3(x_4 + x_5)$

$+x_1x_4(x_5)$

$+x_2x_3(x_4 + x_5)$

$+x_2x_4(x_5)$

$+x_3x_4(x_5)$

The sum of quad products is: $x_1x_2x_3(x_4 + x_5)$

$+x_1x_2x_4(x_5)$

$+x_2x_3x_4(x_5)$

The sum of quint products is: $x_1x_2x_3x_4(x_5)$

10 Finding sums of products by programming

10.1.1 **The sum of pairs (selections of 2)**

To find sums of pairs (selections of 2) in range x_K to x_M

For example: x_4 to x_7

K=4, M=7, N=2

The sum is:

$x_4.x_5 + x_4.x_6 + x_4.x_7$
$+ x_5.x_6 + x_5.x_7$
$+ x_6.x_7$

Written another way, the sum is:

$x_4(x_5+x_6+x_7) + x_5(x_6+x_7) + x_6(x_7)$

i.e.:

x_4 times (sum of singles in range x_5 to x_7)

+ x_5 times (sum of singles in range x_6 to x_7)

+ x_6 times (sum of singles in range x_7 to x_7)

i.e.

x_K times (sum of singles in range x_{K+1} to x_M)

+ x_{K+1} times (sum of singles in range x_{K+2} to x_M)

+.....

+ x_{M-1} times (sum of singles in range x_M to x_M)

To generate this in a program:

```
Sum = 0
For i = K to M-1
 For j = K+1 to M
  Product = xi.xj
  Sum = Sum + Product
 Next j
Next i
```

10.1.2 **The sum of triples (selections of 3)**

To find sums of triples (selections of 3) in range x_K to x_M

For example: x_5 to x_8

K=5, M=9, N=3

Sum is

x_5 times($x_6.x_7 + x_6.x_8 + x_6.x_9 + x_7.x_8 + x_7.x_9 + x_8.x_9$) + x_6 times($x_7.x_8 + x_7.x_9 + x_8.x_9$) + x_7 times($x_8.x_9$)

i.e.:

x_5 times (sum of pairs in range x_6 to x_9)

+ x_6 times (sum of pairs in range x_7 to x_9)

+ x_7 times (sum of pairs in range x_8 to x_9)

10.1.3 **The sum of selections of N**

To generate this algorithmically:

```
For i1 = K To M + 1 - N
      product(1) = x(i1)
      If N > 1 Then
         For i2 = i1 + 1 To M + 2 - N
            product(2) = product(1) * x(i2)
            If N > 2 Then
               For i3 = i2 + 1 To M + 3 - N
                  product(3) = product(2) * x(i3)
                  sum = sum + product(3)
               Next i3
            Else
               sum = sum + product(2)
            End If
         Next i2
      Else
         sum = sum + product(1)
      End If
   Next i1
```

Where the nesting would be continued to the maximum level required.

11 Finding subsets of M from N

In the following, we are finding all the possible subsets of M integers in the range 1 to N.

What seems an easy enough task when N = 3 and M = 2, soon becomes very difficult when these values are higher. A system is required.

11.1 Subsets of 4 in the range 1 to 7

Below is a systematic progression for 4 integers in the range 1 to 7.

1	2	3	4
1	2	3	5
1	2	3	6
1	2	3	7
1	2	4	5
1	2	4	6
1	2	4	7
1	2	5	6
1	2	5	7
1	2	6	7
1	3	4	5
1	3	4	6
1	3	4	7
1	3	5	6
1	3	5	7
1	3	6	7
1	4	5	6
1	4	5	7
1	4	6	7
1	5	6	7
2	3	4	5
2	3	4	6
2	3	4	7
2	3	5	6
2	3	5	7
2	3	6	7

2	4	5	6
2	4	5	7
2	4	6	7
2	5	6	7
3	4	5	6
3	4	5	7
3	4	6	7
3	5	6	7
4	5	6	7

We started with 1234 and then indexed the 4th column until it reached the limit 7.

We then indexed the 3rd column (3 to 4) and started by resetting the 4th column to one higher.

We repeated the indexing of the 3rd column until it could go no higher, i.e. the 4th column started at 7.

We then indexed the second column (2 to 3) and started by resetting the 3rd and 4th columns in ascending values.

We repeated the above process until indexing the second column meant that it could go no higher because all columns to the right were packed and so on.

11.2 Subsets of M in the range 1 to N

It is possible to generalise the above process

Set the first column to 1 and then reset all columns to the right.

Note that resetting all columns to the right means make them ascend in turn from the current column's contents.

Increment the Nth column until the contents is equal to M

Index the (N-1)th column and 'reset' column to right

Index from the Nth column until the contents is equal to M.

Index the (N-1)th column again and repeat the above process until the contents is equal to M-1.

Then start indexing the (N-2)th column and repeat.

On the next page is a routine which generates these combinations.

```
For N = 1 To M
   count = 0
   K = N - 1
   For i1 = 1 To M - K
      S(1) = i1
      If N > 1 Then
         For i2 = i1 + 1 To M - K + 1
            S(2) = i2
            If N > 2 Then
               For i3 = i2 + 1 To M - K + 2
                  S(3) = i3
                  If N > 3 Then
                     For i4 = i3 + 1 To M - K + 3
                        S(4) = i4
                        If N > 4 Then
                        Else
                           count = count + 1
                           Write (S)
                        End If
                     Next i4
                  Else
                     count = count + 1
                     Write (S))
                  End If
               Next i3
            Else
               count = count + 1
               Write (S)
            End If
```

```
        Next i2
      Else
        count = count + 1
        Write (S)
      End If
    Next i1
  Next N
```

12 Conclusion

This paper is not yet complete; therefore, a conclusion is not yet required. This paper has been published in this form to elicit comments on the content.

13 Existing and Emerging Standards

IEC 61508 Edition 2.
IEC 61511 Edition 2.

14 61508 Association Recommended Practices

This document sets out to describe current best practices in *Reliability and Availability* for functional safety systems, but does not seek to prescribe specific measures, since these will depend on the application, and any existing constraints of the installation.

This paper is not yet completed, it has been published solely to elicit comment and encourage input to further develop the technical content.

DISCLAIMER: Whilst every effort has been made to ensure the accuracy of the information contained in this document neither "The 61508 Association" nor its members will assume any liability for any use made thereof.

*** END OF DOCUMENT ***