Algoritmo Probabilístico

Software RMES

Centro de Desarrollo de Gestión Empresarial
1 Oriente 1007 – Viña del Mar – Chile
Fono: (56) (32) 2688987 – Fax: (56) (32) 2684079
empresa@cgessa.com

René González Leiva
Ingeniero de Desarrollo
Septiembre 2012

Pamela Cuevas González
Ingeniero Analista
Septiembre 2012

Resumen

El presente documento tiene como objetivo documentar el algoritmo probabilístico implementado en RMES para la obtención de indicadores probabilísticos de confiabilidad. Este documento requiere que el lector tenga conocimientos básicos en el uso y conceptos asociados al software RMES y la Ingeniería en Confiabilidad. Además, se necesita que el lector este familiarizado con el documento (Gonzalez, 2012).

Contenido

Ilustraciones

Tablas


CGS

Introducción

El documento (Gonzalez, 2012) introduce los términos principales que utilizará este documento. Por ende, se dará por hecho de que el lector comprende los siguientes conceptos:

1. **Nodo**
2. **Bloque**
3. **Configuración**
4. **Tupla**

Entendiendo estas definiciones, es posible saber como afecta el no funcionamiento de los nodos hijos sobre los nodos padres, idea recursiva que comienza desde los bloques hacia los nodos padres llegando hasta el nodo raíz, o sea, la planta o flota completa.

Gracias el esquema RBD y el uso de los conceptos de nodo y tupla es posible obtener KPIs históricos a todo nivel de la planta y de cualquiera de sus componentes.

Existen muchas configuraciones lógico-funcionales en las que se puede encasillar a una configuración (en serie, paralelo, fraccionamiento, entre otras). Éstas rigen el comportamiento de cómo deben ser tratadas las tuplas que llegan desde los nodos hijos hacia los nodos padres.

Para efectos en la explicación de algunos temas del documento, se adelantará que existen al menos 4 tipos de fallas:

1. MP (Mantención Preventiva Programada)
2. MC (Mantención Correctiva No Programada)
3. DO (Detención Operacional Programada)
4. DONP (Detención Operacional No Programada)

Las MC de dividen en subtipos y existen muchos mas tipos de fallas, pero por ahora no nos interesan y nos bastará solo con estas cuatro.

El algoritmo probabilístico utiliza el concepto de recorrido Post-Orden, explicado en (Gonzalez, 2012). Se recomienda comprenderlo antes de continuar con este documento.

1. Análisis básico automático

Cada vez que en RMES se agregan, modifican o eliminan tuplas de algún nodo, automáticamente se disparan varios procesos.

Uno de ellos es el **Filtro de Intersección**, el cual procesa las tuplas modificadas de forma que éstas finalmente queden ordenadas y no traslapadas. El documento (Gonzalez, 2012) explica en detalle el algoritmo. Luego de ello, siempre tendremos tuplas consistentes (que cumplen las dos condiciones anteriormente mencionadas), las cuales son las que finalmente serán asignadas al nodo.

A continuación, se gatilla un proceso que permite generar sobre el nodo indicadores de confiabilidad básicos como son:

- Cantidad de tuplas de tipo MC ($\#_{MC}$)
- Cantidad de tuplas de tipo MP ($\#_{MP}$)
- Cantidad de tuplas de tipo DO ($\#_{DO}$)
- Tiempo Total de tuplas de tipo MC (t_{MC})
- Tiempo Total de tuplas de tipo MP (t_{MP})
- Tiempo Total de tuplas de tipo DO (t_{DO})
- Costo de Reparación Promedio

Con ellos, se pueden derivar los siguientes:

- $MTTR = \frac{t_{MC}}{\#_{MC}}$
- $MTTI = \frac{t_{MC}+t_{MP}}{\#_{MC}+\#_{MP}}$

Además, se estima una distribución de TBF (Tiempo entre fallas) extraída desde las tuplas. **Esto solo se realiza si el nodo es un bloque**. Con esta distribución de TBF, es posible obtener la curva de confiabilidad y el MTBF del bloque.

La siguiente sección explicará el algoritmo de estimación de parámetros para el TBF.

2. Estimación de parámetros para TBF

2.1. Ajuste con censura a la derecha

Ya hemos adelantado que al menos existen 4 tipos de fallas en RMES:

1. MP (Mantención Preventiva Programada)
2. MC (Mantención Correctiva No Programada)
3. DO (Detención Operacional Programada)
4. DONP (Detención Operacional No Programada)

La estimación de parámetros sobre la variable TBF de un conjunto de tuplas estudia el buen funcionamiento entre tuplas de tipo MC y MP. Las tuplas de tipo DO, DONP u otras no reflejan un comportamiento propio del componente dado que se refieren a no funcionamientos debido a causas externas al bloque, por ende, congelan el horómetro del componente hasta encontrar una tupla de tipo MC o MP.

En detalle, las tuplas de tipo MC proporcionan información directa del comportamiento del equipo. La primera aproximación que se puede hacer para realizar una estimación del comportamiento del bloque es solo considerar los tiempos entre fallas (como horómetro) entre tuplas de tipo MC, lo cual no es una mala aproximación pero podría ser mejor. El algoritmo denominado **Ajuste de Curva con Información Censurada a la Derecha** consiste en incorporar a la estimación la información de las tuplas MP. La Ilustración 1 y la Tabla 1 presentan un ejemplo que utilizaremos a lo largo de la sección para explicar los algoritmos

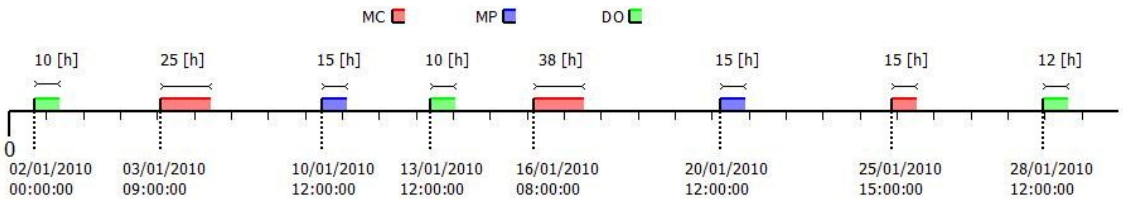

Ilustración 1. Ejemplo de tuplas sobre un bloque

Fecha	Hora	Duración	Tipo
02/01/2010	00:00:00	10	DO
03/01/2010	09:00:00	25	MC
10/01/2010	12:00:00	15	MP
13/01/2010	12:00:00	10	DO
16/01/2010	08:00:00	38	MC
20/01/2010	12:00:00	15	MP
25/01/2010	15:00:00	15	MC
28/01/2010	12:00:00	12	DO

Tabla 1. Tuplas de ejemplo

Para calcular un TBF entendiéndolo como la idea de tiempo entre tuplas de tipo MC y MP, primero, debemos introducir el concepto de *TBT* (*Time Between Tuple*), que se refiere al tiempo entre cualquier tipo de falla. Ambos conceptos, TBF y TBT, necesariamente se calculan entre 2 tuplas. La primera tupla la denominaremos *Tuplas Anterior* y a la segunda, *Tupla Posterior*. Para incorporar la información de las tuplas de tipo MP, es necesario que, cuando obtengamos un TBF o un TBT, sepamos si la *Tupla Posterior* es una MC o no. De esta forma, cada TBF o TBT tendrá una variable de estado: si es MC o no, como muestra la Ilustración 2.

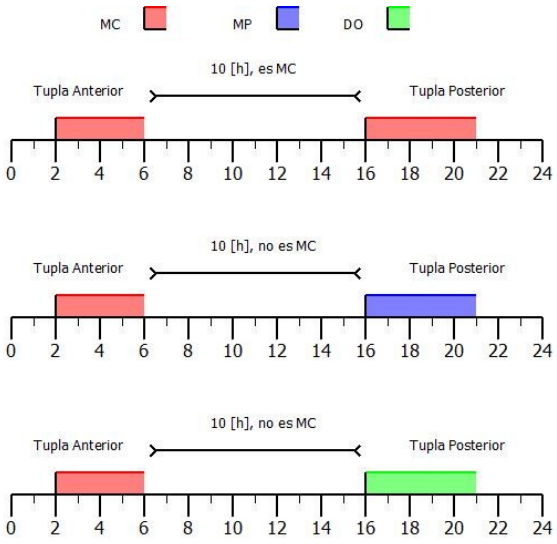

Ilustración 2. Calculo de Tiempo entre Tuplas.

El algoritmo comienza a obtener los TBT desde el fin de la primera falla hasta el inicio de la última. Dado que las detenciones de otro tipo distinto a MC o MP detienen el horómetro del componente, la obtención de los TBFs se debe realizar teniendo en cuenta esta consideración. La Ilustración 3 muestra gráficamente el proceso.

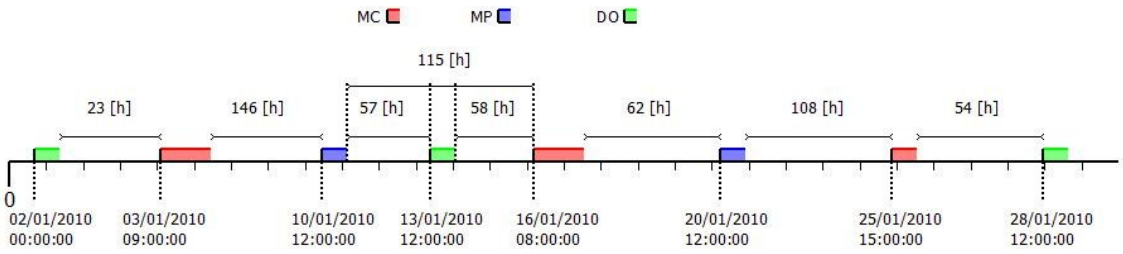

Ilustración 3. Cálculo de TBFs del ejemplo

La Tabla 2 muestra una de las tantas formas posibles de calcular los TBFs, en base al cálculo previo de los TBTs. Por cada tupla, calculamos la cantidad de horas que han pasado desde la fecha 01/01/2012 a las 00:00:00 hasta la fecha de inicio y final. A estas horas, respectivamente, se les denominará *Inicio Relativo* y *Fin Relativo*. Luego, se calcula el TBT utilizando los tiempos relativos

anteriores. Este valor se acumula en la variable *Horómetro*. Si el tipo de la tupla es MC o MP, el horómetro se resetea y se asigna este valor como TBF. Si el tipo de la tupla es cualquier otro tipo, no se crea el TBF y el valor se acumula al horómetro. Al llegar a la última tupla, el último TBT generado se agrega automáticamente como TBF y en caso de no ser MC, se asigna arbitrariamente como si fuera MP. La Tabla 2 resume todo el proceso.

Inicio Relativo	Fin Relativo	Tipo	TBT	Es MC/MP	Horómetro	TBF
24	34	DO	-	No	-	-
57	82	MC	23	Si	23	23
228	243	MP	146	Si	146	146
300	310	DO	57	No	57	-
368	406	MC	58	Si	115	115
468	483	MP	62	Si	62	62
591	606	MC	108	Si	108	108
660	672	DO	54	No	54	-
		MP	-		-	54

Tabla 2. Cálculo de TBFs del ejemplo

Es importante destacar que, luego de calcular el TBF entre 2 fallas, este tiempo se multiplica por un valor llamado **ratio**, que representa la relación entre cantidad de Horas de Operación Anual dividido por la cantidad de horas en un año (8760 [h]).

$$ratio = \frac{Horas\ operacion\ anual}{Horas\ de\ un\ año}$$

Para nuestro ejemplo, el valor de ratio será igual a 1.

Luego, los TBFs se ordenan de menor a mayor y se les asigna un índice correlativo, comenzando desde 1. Además la variable de estado se basa en solamente si el TBF es MC. La Tabla 3 presenta el resultado parcial hasta ahora.

Indice	TBF	Es MC
1	23	Si
2	54	No
3	62	No
4	108	Si
5	115	Si
6	146	No

Tabla 3. Lista de TBFs indexados y ordenados de menor a mayor

El siguiente paso es crear un ranking ajustado que se utilizará para realizar la estimación de parámetros. Este ranking se construye con las siguientes ecuaciones:

$$r_0 = 0$$

$$r_i = r_{i-1} + \frac{(n_r + 1) - r_{i-1}}{(n_r + 1) - (i - 1)} \qquad \forall_{i=\{1,..,n\}}$$

donde

- r_i el valor del ranking actual
- r_{i-1} el valor del ranking anterior
- n_r es la cantidad de TBF's a rankear
- i es el índice correlativo

En este ranking solo se consideran aquellos TBF's asociados a tipos MC, que desde ahora serán conocidos como los valores **x**. Esto quiere decir que el valor r_{i-1} no se calculará si el tipo asociado al TBF es MP. De esta forma, las detenciones MP modifican el ranking final que las detenciones MC tendrán. Para nuestro ejemplo, n=6, entonces, ecuación quedaría:

$$r_i = r_{i-1} + \frac{7 - r_{i-1}}{7 - (i - 1)} \qquad \forall_{i=\{1,..,6\}}$$

La Tabla 4 presenta el resultado parcial hasta ahora en base a ecuación personalizada.

Índice (i)	TBF	Es MC	x	i-1	r_{i-1}	r_i
1	23	Si	23	0	0	1
2	54	No	-	-	-	-
3	62	No	-	-	-	-
4	108	Si	108	3	1	2.5
5	115	Si	115	4	2.5	4
6	146	No	-	-	-	-

Tabla 4. Tabla de Resultados Parciales para los TBFs y su ranking

La **¡Error! No se encuentra el origen de la referencia.** presenta el resultado final del algoritmo.

r	x
1	23
2.5	108
4	115

Tabla 5. Tabla de Resultado Final para algoritmo con Censura a la Derecha

Luego de tener los resultados generados por el algoritmo de Ajuste con Censura a la Derecha se comienza con el ajuste de las curvas. RMES considera 3 casos posibles según la cantidad de datos que este algoritmo entregue:

- Si se generaron entre 0 y 2 datos, no se realiza estimación dado que se posee muy poca información.
- Si se generaron más de 2 datos, se estiman los parámetros para una Exponencial y una Weibull utilizando el método de mínimos cuadrados.

2.2. Estimación exponencial

Los detalles se presentan en los Anexos Análisis Weibull y Análisis Exponencial. Partiremos desde la **¡Error! No se encuentra el origen de la referencia.** que contiene el ranking y los datos asociados. Según lo implementado en RMES, el algoritmo de estimación de parámetros en base al ranking para una distribución exponencial es el siguiente:

$$y_i = 1 - \frac{r_i - 0.3}{n + 0.4}$$

$$\lambda = -\frac{\sum_i^m \ln(y_i) x_i}{\sum_i^m x_i^2}$$

Donde

- x_i es el dato i-ésimo de los TBFs asociados a MC
- n es la cantidad total de tuplas que se consideraron para el ajuste
- m es la cantidad de datos de la **¡Error! No se encuentra el origen de la referencia.**

En nuestro caso particular, n=8 y m=3, resultado que se puede obtener contando las detenciones en la **¡Error! No se encuentra el origen de la referencia.** (las detenciones falsas también se consideran). La Tabla 6 resume los cálculos para el ajuste exponencial, de los cuales podemos desprender el valor del parámetro λ para la exponencial.

x	r	y_i	$ln(y_i)$	x_i^2	$ln(y_i)\, x_i$
23	1.0	0.9166	-0.08701	529	-2.0012
108	2.5	0.7380	-0.30368	11664	-32.7977
115	4.0	0.5595	-0.58066	13225	-66.7769
				25418	-101.5759

Tabla 6. Ajuste Exponencial

Finalmente, el valor de λ será:

$$\lambda = -\frac{-101.5759}{25418} = 0.00399$$

2.3. Estimación Weibull

Los detalles se presentan en los Anexos Análisis Weibull y Análisis Exponencial. La **¡Error! No se encuentra el origen de la referencia.** también nos será útil para el desarrollo del ejemplo. En este caso, en necesario filtrar todos los valores menores o iguales a 0. Dado que en nuestro caso, esto no ocurre, este filtro es innecesario. Los parámetros α y β se calculan con las siguientes ecuaciones:

$$y_i = 1 - \frac{r_i - 0.3}{n + 0.4}$$

$$v_i = \ln(x)$$

$$w_i = \ln(-\ln(y_i))$$

$$a = \mathrm{m}\sum_i^m v_i w_i - \sum_i^m v_i \sum_i^m w_i$$

$$b = \sum_i^m w_i \sum_i^m v_i^2 - \sum_i^m vi \sum_i^m v_i w_i$$

$$c = m\sum_i^m v_i^2 - \left(\sum_i^m vi\right)^2$$

donde m es la cantidad de datos de la **¡Error! No se encuentra el origen de la referencia.**. En nuestro caso, m=3.

Finalmente, los parámetros para la distribución Weibull serán:

$$\alpha = e^{-\frac{b}{a}}$$

$$\beta = \frac{a}{c}$$

x	r	v_i	w_i	v_i^2	$v_i w_i$
23	1.0	3.135494216	-2.441716399	9.831323978	-7.655987646
108	2.5	4.682131227	-1.191772815	21.92235283	-5.580036713
115	4.0	4.744932128	-0.543574052	22.5143809	-2.579221985

	12.56255757	-4.177063266	54.26805771	-15.81524634

Tabla 7. Ajuste Weibull

De Tabla 7 se pueden obtener los valores necesarios para calcular las variables auxiliares a, b y c:

$$a = (3 * -15.8152) - (12.5625 * -4.1770) = 5.0288$$

$$b = (-4.1770 * 54.2680) - (12.5625 * -15.8152) = -28.0011$$

$$c = (3 * 54.2680) - 12.5625^2 = 4.9863$$

Con estos valores, podemos calcular α y β

$$\alpha = e^{-\frac{-28.0011}{5.0288}} = 261{,}9348$$

$$\beta = \frac{5.0288}{4.9863} = 1.0085$$

2.4. Elección de la mejor distribución

Debido al efecto provocado por la censura a la derecha sobre el rankeo de los TBFs, no es posible aplicar alguno de los Test de Hipótesis existentes. Mientras se encuentra algún método que sirva para comparar 2 distribuciones bajo las condiciones de censura a la derecha, utilizaremos el siguiente criterio:

- Sabemos que actualmente solo se estiman 2 distribuciones: Weibull y Exponencial
- Si la estimación del parámetro β de la Weibull esta entre [0.98, 1.02], seleccionará arbitrariamente una Exponencial
- De lo contrario, siempre se seleccionará una Weibull.

3. Algoritmo Probabilístico

3.1. Clonación y Procesamiento

El primero proceso que se realiza es una clonación completa de la planta/flota a una estructura de árbol muy similar a la implementada en RMES. El objetivo de esto es abstraer todo el proceso probabilístico de forma que el corazón lógico de RMES éste lo más ordenado y limpio posible. De esta forma, la nueva estructura contiene información necesaria para propagar los cálculos probabilísticos desde los bloques a las configuraciones superiores.

Estos nuevos subsistemas se conocen como PSubsystem (por su denominación informática, Probabilistic-Subsystem) y cada uno es un objeto par de un subsistema real en la planta.

A nivel de bloque, existe un PEquipment, el cual extrae de su bloque par la siguiente información:

- La distribución del TBF (puede no existir)
- Si la información que se dispone actualmente del bloque proviene de una importación e datos (en caso contrario, sabremos que se ingresaron manualmente)
- El MTBF
- El MTTR
- El MTTI
- El Tiempo de Operación
- El Costo de Reparación

Debemos dejar en claro que, en caso de que existan tuplas importadas y el usuario tenga seleccionada la opción de extraer la información de allí, la información anterior se obtiene del análisis simple de las tuplas. Por ejemplo, para el MTBF, en realidad, sería el MTBF de las tuplas importadas al bloque.

A nivel de configuración, los datos que se extraen son las tuplas que tenga cargadas. Estos son:

- Si existen datos importados o no
- El MTBF de los datos importados
- El MTTR de los datos importados
- El MTTI de los datos importados

Por supuesto, es no son los indicadores probabilísticos que queremos obtener, pero nos servirán para propósitos posteriores. Con esta información, podemos comenzar a procesar la planta. Se comienza desde los bloques y se sube por configuración hasta llegar al nivel de la planta. Por cada nodo, independiente del tipo, se realizan los siguientes procesos en este orden estricto:

- Calcular la función de confiabilidad R(t)
- Calcular el MTBF
- Calcular el Tiempo de Operación

- Calcular el MTTR
- Calcular el MTTI
- Calcular la disponibilidad (debida al MTTR)
- Calcular la disponibilidad Inherente (debida al MTTI)
- Calcular el costo de mantención

Lo importante tener muy claro que cada vez que visitamos un nivel inferior, se calculan todos los indicadores que el nivel inmediatamente superior podría necesitar. De esta forma, la recursión del nivel superior se limita a obtener los indicadores ya calculados (en *cache*, si se prefiere) de los hijos directos.

3.2. Cálculo de la confiabilidad R(t)

A nivel de **bloque** la confiabilidad se obtiene directamente de la estimación del TBF realizada en base a los siguientes 3 casos posibles:

- Si la distribución es una Weibull, $R(t) = e^{-\left(\frac{t}{\alpha}\right)^{\beta}}$
- Si la distribución es un Exponencial, $R(t) = e^{-\lambda t}$
- Si no existe distribución, se asume una Exponencial con parámetro $\lambda = \frac{1}{MTBF}$. La confiablidad será la misma que en el punto anterior.

A nivel de configuración, el procedimiento es algo más complejo. Comenzaremos con decir que cada tipo de subsistema posee su forma particular de obtener la curva de confiabilidad R(x):

Serie

$$R(t) = \prod_{i=1}^{n} R_i(t)$$

Paralelo

$$R(t) = \begin{cases} R_1(t) + R_2(t) - R_1(t) * R_2(t) & n = 2; \\ R_1(t) + R_2(t) + R_3(t) - R_1(t)R_2(t) - R_1(t)R_3(t) - R_2(t)R_3(t) + R_1(t)R_2(t)R_3(t) & n = 3. \end{cases}$$

Fraccionamiento

$$R(t) = \begin{cases} \sum_{i=1}^{n} I_i * R_i(t) + \left(1 - \sum_{i=1}^{n} I_i\right) * \prod_{i=1}^{n} R_i\,(t) & si\ k = 0 \\ \dfrac{\sum_{i=1}^{n} RushDi(R_1, \dots, R_n, i)}{k} & si\ \ k \neq 0 \end{cases}$$

Redundancia

$$R(t) = RushDi(R_1, \dots, R_n, k)$$

Standby

$$R(t) = \begin{cases} -e^{-\lambda_m t} + \dfrac{\lambda_m}{\lambda_s - \lambda_m} * \left(e^{-\lambda_m t} - e^{-\lambda_s t}\right) & si\ \lambda_s \neq \lambda_m \\ R_s(t) * \left(1 + \lambda_s(t)\right) & si\ \lambda_s = \lambda_m \end{cases}$$

donde

- n es la cantidad de hijos de la configuración
- $R_i(t)$ es la función de confiabilidad del hijo i-ésimo
- I_i es el Impacto del hijo i-ésimo
- Para fraccionamiento y redundancia, k es el nivel de redundancia $(k < n)$
- $RushDi(R_1, \dots, R_n, k)$ es la función Rushdi (ver anexo Rushdi y Rushdi Optimizado)
- Para standby, λ_m es la tasa de falla del componente principal y λ_s es la tasa de falla del componente en standby.

Estas funciones son recursivas en el sentido de que una configuración puede contener otras configuraciones, en tal caso, se ocupará en cada nivel la ecuación correspondiente a la configuración. Este procedimiento, para plantas/flotas pequeñas es suficientemente rápido, pero en presencia de plantas/flotas de gran envergadura y dado su naturaleza recursiva, el tiempo cálculo es extenso, del orden de horas.

En versiones posteriores de RMES, el calculo probabilístico era parte del corazón de esquema lógico-funcional, por ende, era bastante difícil para un informático trabajar con el dado la delicadeza con que se tenía que laborar. Las primeras aproximaciones que se realizaron para resolver el problema de la lentitud del algoritmo se basaron en implementar distintos niveles de *cache*, es decir, estructuras especiales que almacenan los resultados parciales del algoritmo con el

objetivo de evitar la necesidad de recalcular un valor anteriormente obtenido. A pesar de ello, la performance siguió siendo pobre.

La nueva estructura que se diseño, aparte de organizar mejor los procedimientos, es un gran *cache* del algoritmo. De esta forma, se puede tener acceso instantáneo a los resultados probabilísticos para cualquier nodo de la planta/flota. Además, realizar una estimación de la curva de confiabilidad a nivel de configuración utilizando interpolación lineal. Esta aproximación es bastante precisa y se basa en obtener el punto t para todos los puntos de confiabilidad R(t) separados cada 0.01. Es decir, se obtendrá una lista de al menos 100 puntos similar a la mostrada en la Tabla 8.

t_i	$R(t_i)$
$\mathbf{0}$	1
t_1	0.99
t_2	0.98
t_3	0.97
…	…

Tabla 8. Puntos de confiabilidad para aproximación por interpolación lineal

Recordemos que a nivel de configuración, la confiabilidad R(t) no posee una función inversa, por ende, la búsqueda de los puntos t se realiza con un algoritmo de búsqueda binaria. La búsqueda binaria es un procedimiento recursivo bastante simple y común. Lo utilizamos generalmente cuando queremos saber el número telefónico de una persona en base a su nombre (nombre objetivo) revisando una enorme lista ordenada alfabéticamente (una guía telefónica). Lo mejor sería revisar el nombre del individuo que se encuentra exactamente al medio de la lista:

- Si este nombre esta alfabéticamente después de nuestro nombre objetivo, debiéramos revisar entonces el nombre que se encuentre al medio de la mitad de la mitad anterior de la lista, dado que no es posible que se encuentre después.
- Si este nombre esta alfabéticamente antes de nuestro nombre objetivo, ocurre lo contrario, revisamos la mitad de la mitad siguiente de la lista.
- Si este nombre es igual a nuestro nombre objetivo, lo hemos encontrado.

De esta forma, vamos recursivamente buscando en la mitad de la mitad anterior, reduciendo el espacio de búsqueda drásticamente en cada iteración del algoritmo. La salvedad de este algoritmo que sólo se puede utilizar si tenemos bien definidos el inicio y fin absoluto de la lista.

En nuestro caso, dado que no tenemos la inversa de R(t), tampoco sabemos a priori el valor del tiempo t_f tal que $R(t_f)\sim 0$. Además, estrictamente no existe este valor t_f, dado que la función converge a infinito, pero podemos realizar una aproximación bastante razonable de t_f buscando

el valor t tal que $R(t) = 0.01$. Este valor también lo encontraremos con una búsqueda recursiva modificada, en donde antes de realizar la búsqueda, el valor borde inferior t_0' y el borde superior t_f' se buscaran arbitrariamente de forma que $R(t_0') > 0.01$ y $R(t_f') < 0.01$.

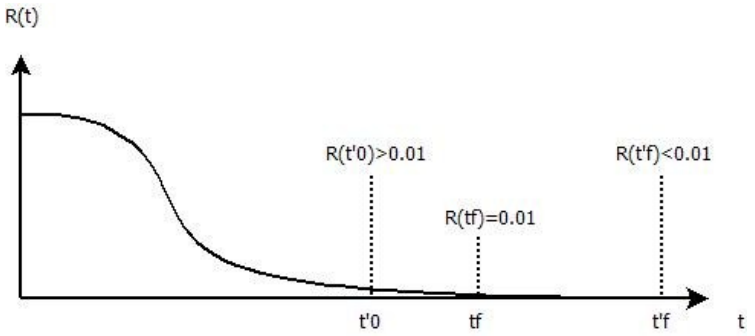

Ilustración 4. Búsqueda recursiva de tf

Luego de haber encontrado el valor de t_f, y sabiendo que $t_0 = 0$, podremos realizar una búsqueda recursiva sobre este intervalo para encontrar todos los puntos t_i presentados en la Tabla 8. Finalmente, para obtener el valor de cualquier punto t' que no este presente en la tabla, se buscará, también con recursión binaria, los puntos t_{i-1} y t_{i+1} tal que $t_{i-1} < t' < t_{i+1}$ y realizamos una interpolación lineal simple, como se muestra en la Ilustración 5.

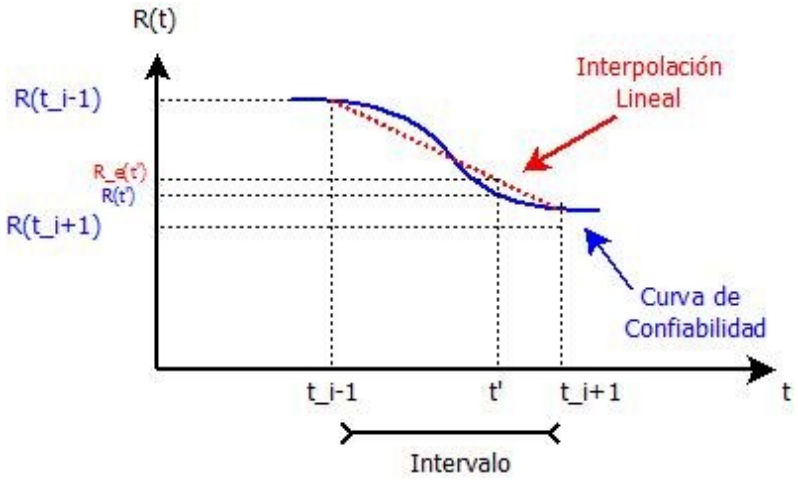

Ilustración 5. Interpolación Lineal para curva de confiabilidad

$R_e(t')$ es el valor de confiabilidad estimado por la interpolación lineal. La diferencia en la estimación observada en la Ilustración 5 es $|R_e(t') - R(t')|$.

Según los test que se realizaron al algoritmo, la máxima diferencia evaluada en todo de la curva de confiabilidad de todos los nodos de la planta testeada no supero el valor de 0.005. Por ejemplo, cuando el valor debiera dar 0.985, el resultado estimado fue 0.980. Esto es un error del 5% en la estimación, lo cual, está dentro de los estándares aceptables.

Finalmente, este algoritmo permite obtener a nivel de configuración una curva estimada de confiabilidad que es el pilar de todos los cálculos que siguen a continuación.

3.3. Cálculo del MTBF

A nivel de bloque, no se realiza ningún procedimiento pues este valor viene dado por el valor del MTBF asignado al inicio de la clonación de la planta.

A nivel de configuración, aunque sabemos que el MTBF es la integral de la curva R(t), la mayoría de las configuraciones posee su forma particular de obtener este valor, lo que aumenta la performance del algoritmo (obtener la integral de una función es generalmente costoso).

Serie

$$MTBF = \frac{1}{\sum_{i=1}^{n} \lambda_i}$$

Paralelo

$$MTBF = \begin{cases} \frac{1}{\lambda_1} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1 + \lambda_2} & if\ n = 2; \\ \frac{1}{\lambda_1} + \frac{1}{\lambda_2} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1 + \lambda_2} - \frac{1}{\lambda_1 + \lambda_3} - \frac{1}{\lambda_2 + \lambda_3} + \frac{1}{\lambda_1 + \lambda_1 + \lambda_3} & if\ n = 3. \end{cases}$$

Fraccionamiento

$$MTBF = \begin{cases} \sum_{i=1}^{n} \frac{I_i}{\lambda_i} + \left(1 - \sum_{i=1}^{n} I_i\right) * \left(\frac{1}{\sum_{i=1}^{n} \lambda_i}\right) & si\ k = 0 \\ \int_0^{t_f} R(t)dt & si\ k \neq 0 \end{cases}$$

Redundancia

$$MTBF = \int_0^{t_f} R(t)\, dt$$

Standby

$$MTBF = \frac{1}{\lambda_m} + \frac{1}{\lambda_s}$$

donde

- n es la cantidad de hijos de la configuración
- $R_i(t)$ es la función de confiabilidad del hijo i-ésimo
- Para fraccionamiento y redundancia, k es el nivel de redundancia $(k < n)$
- λ_i es la tasa de falla del hijo i-ésimo
- I_i es el Impacto del hijo i-ésimo
- Para fraccionamiento y redundancia, k es el nivel de redundancia $(k < n)$
- Para standby, λ_m es la tasa de falla del componente principal y λ_s es la tasa de falla del componente en standby

Recordemos que $\lambda = \frac{1}{MTBF}$ y que en un nivel superior, siempre tendremos los valores calculados del nivel inferior.

En caso de que la configuración tenga datos, o sea, posee un $MTBF_{Tuplas}$ obtenido desde las tuplas, el MTBF de la configuración será:

$$MTBF = \frac{1}{\frac{1}{MTBF_{Calculado}} + \frac{1}{MTBF_{Tuplas}}}$$

donde el $MTBF_{Calculado}$ es el que se obtuvo con alguna de las ecuaciones presentadas.

3.4. Cálculo del Tiempo de Operación

A nivel de bloque, no se realiza ningún procedimiento pues este valor viene dado por el valor del Tiempo de operación (TO) asignado al inicio de la clonación de la planta

A nivel de configuración, el cálculo es simple:

$$TO = \frac{MTBF}{2}$$

El tiempo de operación supuestamente debería ser un valor configurado por el usuario. Por ahora, se asignando este valor calculado como tal.

3.5. Cálculo del MTTR

A nivel de bloque, no se realiza ningún procedimiento pues este valor viene dado por el valor del MTTR asignado al inicio de la clonación de la planta.

A nivel de configuración, en caso de que existan datos, o sea, posee un $MTTR_{Tuplas}$ y un $MTBF_{Tuplas}$ obtenido desde las tuplas, el MTTR de la configuración será:

$$MTTR = \left(\left(\sum_{i=1}^{n} \frac{MTTR_i}{MTBF_i}\right) + \frac{MTTR_{Tuplas}}{MTBF_{Tuplas}}\right) * MTBF$$

En este punto, sabemos cual es el valor del MTBF, dado que se calculó cronológicamente primero. Si no existen datos importados, se utilizan las siguientes ecuaciones según sea el caso.

Serie

$$MTTR = \left(\sum_{i=0}^{n} MTTR_i\ \lambda_i\right) * MTBF$$

Paralelo

$$MTTR = \begin{cases} mín(MTTR_1, MTTR_2) & si\ n = 2 \\ mín(MTTR_1, MTTR_2, MTTR_3) & si\ n = 3 \end{cases}$$

Fraccionamiento

$$MTTR = \begin{cases} MTBF * \left(\dfrac{1}{A_{MTTR\_temp}} - 1\right) & si\ k = 0 \\ \dfrac{\sum_{i=1}^{n} MTTR_i}{n} & si\ k \neq 0 \end{cases}$$

Redundancia

$$MTTR = \frac{\sum_{i=1}^{n} MTTR_i}{n}$$

Standby

$$MTTR = mín\{ MTTR_1, MTTR_2\}$$

donde

- n es la cantidad de hijos de la configuración
- $MTTR_i$ es el MTTR del hijo i-ésimo
- λ_i es la tasa de falla del hijo i-ésimo
- Para fraccionamiento y redundancia, k es el nivel de redundancia $(k < n)$

Para fraccionamiento, necesitamos una disponibilidad temporal A_{MTTR_temp} asociada a la configuración. Sabemos que la disponibilidad de cualquier nodo será

$$A = \frac{MTBF}{MTBF + MTTR}$$

En este punto del cálculo solo tenemos el valor del MTBF de la configuración. Por suerte, tenemos una ecuación especial para la disponibilidad de un fraccionamiento con k=0:

$$A_{MTTR\_temp} = \sum_{i=1}^{n} \frac{MTBF_i}{MTBF_i + MTTR_i} + \left(1 - \sum_{i=1}^{n} I_i\right) * \left(\prod_{i=1}^{n} \frac{MTBF_i}{MTBF_i + MTTR_i}\right)$$

donde

- n es la cantidad de hijos de la configuración
- $MTTR_i$ es el MTTR del hijo i-ésimo
- $MTBF_i$ es el MTBF del hijo i-ésimo
- I_i es el Impacto del hijo i-ésimo

3.6. Cálculo del MTTI

A nivel de bloque, no se realiza ningún procedimiento pues este valor viene dado por el valor del MTTI asignado al inicio de la clonación de la planta.

A nivel de configuración, en caso de que existan datos, o sea, posee un $MTTI_{Tuplas}$ y un $MTBF_{Tuplas}$ obtenido desde las tuplas, el MTTR de la configuración será:

$$MTTI = \left(\left(\sum_{i=1}^{n} \frac{MTTI_i}{MTBF_i}\right) + \frac{MTTI_{Tuplas}}{MTBF_{Tuplas}}\right) * MTBF$$

En este punto, sabemos cual es el valor del MTBF, dado que se calculó cronológicamente primero. Si no existen datos importados, se utilizan las siguientes ecuaciones según sea el caso.

Serie

$$MTTI = \left(\sum_{i=0}^{n} MTTI_i \, \lambda_i\right) * MTBF$$

Paralelo

$$MTTI = \begin{cases} mín(MTTI_1, MTTI_2) & si\ n = 2 \\ mín(MTTI_1, MTTI_2, MTTI_3) & si\ n = 3 \end{cases}$$

Fraccionamiento

$$MTTI = \begin{cases} MTBF * \left(\dfrac{1}{A_{MTTI\_temp}} - 1\right) & si\ k = 0 \\ \dfrac{\sum_{i=1}^{n} MTTI_i}{n} & si\ k \neq 0 \end{cases}$$

Redundancia

$$MTTI = \frac{\sum_{i=1}^{n} MTTI_i}{n}$$

Standby

$$MTTI = mín\{MTTI_1, MTTI_2\}$$

donde

- n es la cantidad de hijos de la configuración
- $MTTI_i$ es el MTTI del hijo i-ésimo
- λ_i es la tasa de falla del hijo i-ésimo
- Para fraccionamiento y redundancia, k es el nivel de redundancia $(k < n)$

Para fraccionamiento, necesitamos una disponibilidad temporal A_{MTTI_temp} asociada a la configuración. La disponibilidad inherente debida al MTTI de cualquier nodo será

$$A_{MTTI} = \frac{MTBF}{MTBF + MTTI}$$

En este punto del cálculo solo tenemos el valor del MTBF de la configuración. La ecuación para la disponibilidad inherente de un fraccionamiento con k=0 es:

$$A_{MTTI\_temp} = \sum_{i=1}^{n} \frac{MTBF_i}{MTBF_i + MTTI_i} + \left(1 - \sum_{i=1}^{n} I_i\right) * \left(\prod_{i=1}^{n} \frac{MTBF_i}{MTBF_i + MTTI_i}\right)$$

donde

- n es la cantidad de hijos de la configuración
- $MTTI_i$ es el MTTI del hijo i-ésimo
- $MTBF_i$ es el MTBF del hijo i-ésimo
- I_i es el Impacto del hijo i-ésimo

3.7. Cálculo de la Disponibilidad

La disponibilidad a todo nivel se calcula como

$$A = \frac{MTBF}{MTBF + MTTR}$$

3.8. Cálculo de la Disponibilidad Inherente

La disponibilidad inherente a todo se calcula como

$$A_{MTTI} = \frac{MTBF}{MTBF + MTTI}$$

3.9. Cálculo del Costo de Mantención

A nivel de bloque, el Costo de Mantención (CM) se calcula como

$$MC = \frac{HOA}{MTBF + MTTR} * CR$$

donde

- HOA son las horas de operación anual de la planta
- $MTBF$ es el Tiempo entre Fallas del bloque
- $MTTR$ es el Tiempo de Reparación del bloque
- CR es el Costo de Reparación del bloque

A nivel de configuración, cualquier sea, la ecuación es

$$MC = \sum_{i=1}^{n} MC_i$$

donde

- n es la cantidad de hijos de la configuración
- MC_i es el Costo de Mantención del hijo i-ésimo

4. Modelo Tasa de Fallas

A nivel equipo la tasa de falla $\tau(t)$ se determina como función del tiempo según la curva de distribución del TBF estimada:

- $\tau(t) = \frac{\beta}{\alpha} \times \left(\frac{t}{\alpha}\right)^{\beta-1}$ si el ajuste de curvas entrega distribución Weibull.

- $\tau(t) = \frac{1}{MTBF} = cte$, si el ajuste es distribución exponencial.

A nivel de sistema es razonable calcular la tasa de falla a partir de las curvas de confiabilidad resultantes, de acuerdo a lo siguiente:

$$\tau(t_i) = \frac{R(t_{i-1}) - R(t_i)}{R(t_{i-1}) \times (t_i - t_{i-1})}$$

RMES al graficar una curva de R(t) en el dominio $[0, t_f]$ utiliza solo algunos puntos de la curva y los une con líneas rectas generando visualmente una curva continua. El paso entre un t_i y t_{i-1} se conoce como Δt, es decir:

$$\Delta t = t_i - t_{i-1} \qquad \forall\, i = 1, \dots, n$$

Donde n será la cantidad de puntos a considerar. Por defecto, en RMES es 50.

Para el gráfico de la tasa de falla, se utiliza esta misma idea: el dominio $[0, t_f]$ se divide en n partes y para cada una se obtiene el valor $\tau(t_i)$. Recordemos que luego de realizado el algoritmo probabilístico, en cada configuración se tiene una curva de confiabilidad estimada. De esta forma, la obtención de la curva $\tau(t)$ es directa con un mínimo de uso de recursos del sistema.

Bibliografía

Cuevas, P. (Septiembre de 2012). *Manual Técnico de los Reporte Software R-MES.*

Gonzalez, R. (Septiembre de 2012). *Introducción a Algoritmos en RMES.*

Pages, M. (s.f.). *Weibull Analysis.* Recuperado el 10 de 01 de 2011., de Math Pages: http://www.mathpages.com/home/kmath122/kmath122.htm

Wikipedia. (s.f.). *ANOVA.* Obtenido de Wikipedia: http://es.wikipedia.org/wiki/An%C3%A1lisis_de_la_varianza

Wikipedia. (s.f.). *Contraste de Hipotesis.* Obtenido de Wikipedia: http://es.wikipedia.org/wiki/Contraste_de_hip%C3%B3tesis

Wikipedia. (s.f.). *Distribucion Normal.* Obtenido de Wikipedia: http://es.wikipedia.org/wiki/Distribuci%C3%B3n_normal

Wikipedia. (s.f.). *Hipotesis nula.* Obtenido de Wikipedia: http://es.wikipedia.org/wiki/Hip%C3%B3tesis_nula

Wikipedia. (s.f.). *Series de Taylor.* Recuperado el 01 de 10 de 2012, de Wikipedia: http://es.wikipedia.org/wiki/Serie de Taylor

Wikipedia. (s.f.). *Taylor Series.* Recuperado el 01 de 10 de 2012, de Wikipedia: http://en.wikipedia.org/wiki/Taylor

Wiley, M. (s.f.). *Symmetric Switching Function Approach by Rushdi*. Obtenido de The k-out-of-n System Model: http://media.wiley.com/product_data/excerpt/1X/04713976/047139761X.pdf

Anexos

1. Análisis Weibull

Dada cualquier función *g(x)* con *g(0) = 0* y crezca monótonamente hacia el infinito mientras x tienda al infinito, es posible definir una función de probabilidad acumulada como:

$$F(t) = 1 - e^{-g(t)}$$

Obviamente, la probabilidad en el tiempo *t=0* es cero y crece monótonamente hasta 1 mientras *t* tienda al infinito. La correspondiente función de densidad $f(t)$ es la derivada de $F(t)$:

$$f(t) = g'(x)\, e^{-g(t)}$$

La tasa de falla para una función de densidad dada se define como

$$R(t) = \frac{f(x)}{1 - F(t)}$$

Entonces, la tasa de falla para la función de densidad anterior es:

$$R(t) = \frac{g'(x)\, e^{-g(t)}}{1 - (1 - e^{-g(t)})} = g'(x)$$

Esto nos permite definir una distribución con una tasa de ocurrencia R(t) especificada. Una familia de funciones de tasa de 2 parámetros muy útil es:

$$R(t) = \frac{\alpha}{\beta}\left(\frac{t}{\alpha}\right)^{\beta-1}$$

donde α y β son constantes. Esta familia de distribuciones, llamadas así por el ingeniero Suizo Waloddi Weibull (1887-1979) quien popularizo su uso para análisis de confiabilidad, especialmente para modos de fallas metalúrgicos. El primer paper de Weibull acerca del tema fue publicado en 1939, pero el meto-do no atrajo mucho hasta el año 1950. La "distribución Weibull" también fue estudiada en los años 20 por el estadista Emil Gumbel (1891-1966), que es recordado hoy por su enfrentamiento con los nazis en 1931 cuando se organizó una campaña para obligarlo a salir de su catedra en la Universidad de Heidelberg por su abierto opinión pacifista y puntos de vista anti-nazi. La constante α es llamada parametro de escala, porque escala a la variable t, y la constante β es llamada parámetro de forma, porque determina la forma de la función de tasa. (Ocasionalmente, la variable t en la definición anterior es remplazada por t-g, donde g es un tercer parámetro usado para definir un punto cero adecuado). Respecto del parámetro de forma β:

- Si $\beta > 1$ la tasa crece con t
- Si $\beta < 1$ la tasa decrece con t
- Si $\beta = 1$, la tasa es constante.

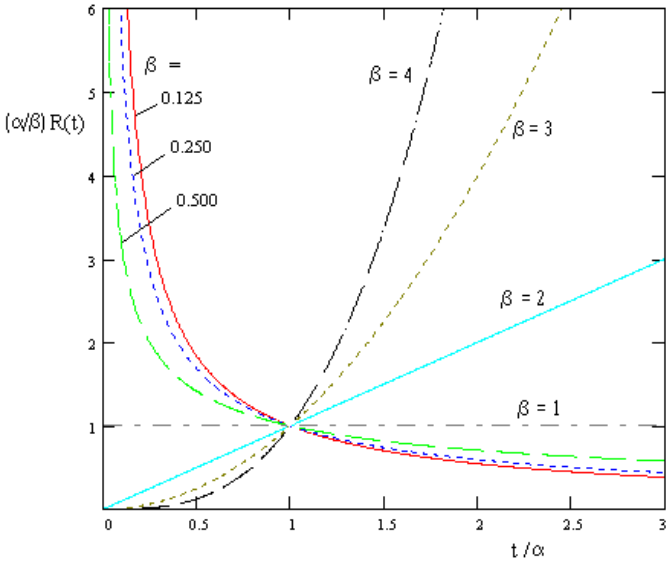

Figura 1. Tasa de falla.

En el último caso, la distribución Weibull se iguala a una distribución Exponencial. La forma de la función tasa de falla se ilustra en la figura 1.

Dado que *R(t)* es igual a $g'(t)$, integramos esta función para obtener:

$$g(t) = \left(\frac{t}{\alpha}\right)^{\beta}$$

Claramente, para cualquier valor positivo de α y β la función g(t) crece monótonamente desde 0 hasta el infinito mientras t se lo haga también, lo que nos lleva a definir una probabilidad de densidad acumulada:

$$F(t) = 1 - e^{-\left(\frac{t}{\alpha}\right)^{\beta}}$$

Ecuación 1. Función de probabilidad Weibull (pdf)

y su correspondiente función de densidad:

$$f(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} e^{-\left(\frac{t}{\alpha}\right)^{\beta}}$$

Supongamos que tenemos un conjunto de n procesos, donde n es un número grande, cada uno comienza operando continuamente al mismo tiempo t = 0 (entiéndase proceso como intervalo de tiempo de buen funcionamiento; con esto se pretende simplificar el problema del funcionamiento continuo de un equipo, asumiendo que cada vez que este falle y restablezca su funcionamiento la variable de tiempo t vuelve a cero, de esta forma t no representa tiempo de continuidad asociada

a fecha, si no que a la duración de los procesos). Cada proceso tiene una distribución acumulada de falla de tipo Weibull dada por Ecuación 1 para parámetros fijos α y β.

El número esperado N(t) de procesos para el tiempo *t* (es decir, del conjunto *n* de procesos, la cantidad de ellos que aun seguirán en funcionamiento hasta el tiempo *t*) es:

$$N(t) = F(t)n$$

$$N(t) = \left(1 - e^{-\left(\frac{t}{\alpha}\right)^{\beta}}\right)n$$

Ecuación 2. Numero esperado de procesos

Reordenando la Ecuación 2:

$$1 - \frac{N(t)}{n} = e^{-\left(\frac{t}{\alpha}\right)^{\beta}}$$

Aplicando y multiplicando por -1

$$ln\left(\frac{1}{1 - \frac{N(t)}{n}}\right) = \left(\frac{t}{\alpha}\right)^{\beta}$$

Aplicando nuevamente *ln(.):*

$$ln\left(ln\left(\frac{1}{1 - \frac{N(t)}{n}}\right)\right) = \beta \ln(t) - \beta \ln(\alpha)$$

Ecuación 3. Ecuación Estimación Weibull

Dada un conjunto inicial de $n = 100$ de procesos que comiencen en $t = 0$ y acumulando horas continuamente posteriormente, supongamos que la primera falla ocurre en un tiempo $t = t_1$. Aproximadamente, podríamos decir que el número esperado de fallas al tiempo de la primera falla es 1 (o sea, aproximadamente $N(t_1) = 1$), entonces, según la ecuación 9, $F(t_1) = N(t_1)/n = 1/100$. Sin embargo, esto (decir que $N(t_1) = 1$) no es muy óptimo, dado que estadísticamente la primera falla probablemente ocurra algo antes de que el número esperado de fallas alcance 1. Para entender el porqué de esto, consideremos un conjunto que consista únicamente de un sólo proceso, en tal caso, el número esperado de fallas a un tiempo *t* cualquiera sería meramente $F(t)$, el cual sólo se aproxima a 1 mientras t tienda al infinito, y sin embargo, el tiempo medio de falla es en el tiempo $t = t_{mediana}$ donde $F(t_{mediana}) = 0{,}5$. En otras palabras, hay una probabilidad de 0.5 de que la falla ocurra antes que el tiempo medio y de un 0,5 de que ocurra después. De ahí que para un conjunto de tamaño 1, el número esperado de fallas al tiempo medio de la primera falla es sólo de 0,5.

En general, dado un conjunto de n procesos, cada uno con una función de densidad $f(t)$, la probabilidad individual de que alguno falle al tiempo t_m es $F(t_m) = N\ (t_m)/n$. Definiendo el valor por Φ, la probabilidad exacta de que *j* procesos fallen y $n - j$ no lo hagan al tiempo t_m sigue una distribución binomial [4] y es:

$$P(j;n) = \binom{n}{x} \Phi^j (1-\Phi)^{n-j}$$

Y por consiguiente, la probabilidad de que j o más fallen al tiempo t_m es:

$$P(\geq j;n) = \sum_{i=j}^{n} \binom{n}{i} \Phi^j (1-\Phi)^{n-i}$$

Esto representa la probabilidad de que la j-étimo (de n) falla ocurra en el tiempo t_m, y por supuesto, el complemento es la probabilidad de que la j-étimo aun no ocurra al tiempo t_m.

Por lo tanto, dado que la j-ésima falla ocurre al tiempo t_m, el valor "medio" de $F(t_m) = \Phi$ se obtiene igualando $P(\geq j;n) =$ 0,5 en la ecuación de arriba y resolviendo para Φ. Este valor se conoce como el rango medio y puede ser computado numéricamente. Un enfoque alternativo es utilizar la muy buena aproximación:

$$F(t_m) = \frac{j-0{,}3}{n+0{,}4}$$

Este es el valor (en vez de j/n) que se debería asignar a $N(t_j)/n$ para la j-ésima falla.

Luego, podemos remplazar este valor en la Ecuación 3, obteniendo

$$ln\left(ln\left(\frac{1}{1-\frac{j-0{,}3}{n+0{,}4}}\right)\right) = \beta \ln(t) - \beta \ln(\alpha)$$

Ecuación 4. Ecuación Estimación Weibull modificada

Utilizando regresión simple (ver sección 3 del anexo) y usando como variables auxiliares

$$x_j = \ln(t_j)$$

$$y_i = ln\left(ln\left(\frac{1}{1-\frac{j-0{,}3}{n+0{,}4}}\right)\right)$$

Obtenemos

$$y_j = \beta x_j - \beta \ln(\alpha)$$

Luego

$$\beta = \frac{k\sum x_j y_j - \sum x_j \sum y_j}{k\sum x_j^2 - (\sum x_j)^2}$$

Ecuación 5. Beta

$$\alpha = exp\left[\frac{\sum y_j \sum x_j^2 - \sum x_j \sum x_j y_j}{-\beta\left(k\sum x_j^2 - (\sum x_j)^2\right)}\right]$$

Ecuación 6. Alpha

donde k es la cantidad de datos.

La dependencia de la tasa de falla $R(t)$ en el tiempo de cada proceso es también una razón que consideramos en el conjunto coherente de procesos cuyas edades están sincronizadas $(t_0 = 0)$. Esto simplifica enormemente el análisis. En situaciones más realistas, el conjunto de proceso cambia, y la "edad" de cada proceso en el conjunto será diferente, de igual manera, la tasa con la cual las horas operacionales se acumulan es una función de tiempo. Mas generalmente, podríamos considerar un conjunto de procesos tales que cada uno posee su "tiempo propio":

$$t_j = m_j(t - g_j)$$

para todo t mas grande que g_j, donde t es el tiempo, g_j es la fecha de nacimiento del j-étimo proceso y m_j es factor de uso operacional. Este tiempo propio es entonces la variable de tiempo para la función de densidad de la Weibull para el j-ésima proceso y la tasa de fallas genera para todo el conjunto en un tiempo dado se compone de todas las tasas individuales. En un conjunto no coherente, cada proceso tiene distintas distribuciones de falla

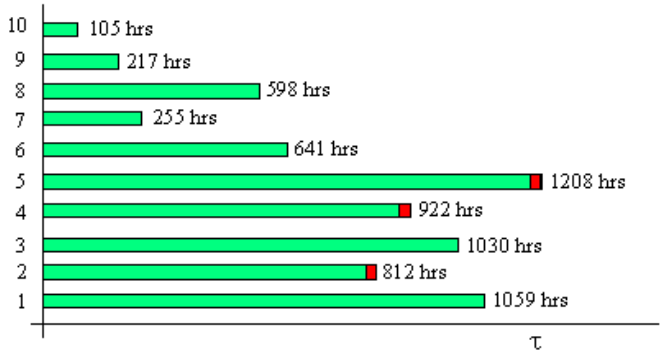

Ilustración 6. Procesos coherentes con censura a la derecha.

La Ilustración 6 muestra procesos coherentes. Los procesos 2, 4 y 5 representan procesos detenidos debido a fallas. El resto, continúan su funcionamiento acumulando horas de funcionamiento a sus respectivas tasas. Ellos son llamados puntos "censurados a la derecha" o "suspendidos", dado que imaginamos que su funcionamiento fue suspendido antes de que el proceso fallara. No sabemos cuando realmente podrían haber fallado, por lo que no pueden ser usados directamente como puntos para el ajuste de la distribución, pero de todas maneras podemos utilizar el hecho de que ellos acumulan horas sin fallas. La aproximación usual en análisis de confiabilidad es primero rankear todos los datos de acuerdo a sus horas acumuladas, tal como muestra la figura 3.

Failure Rank	Overall Rank	Hours	
	1	105	
	2	217	
	3	255	
	4	598	
	5	641	
1	6	812	failed
2	7	922	failed
	8	1030	
	9	1059	
3	10	1208	failed

Ilustración 7. Rankeo de datos para ajuste Weibull.

A continuación, asignamos un ajuste al ranking para los procesos que efectivamente falla. Si denotamos k_j al ranking respecto a todos los datos y $r(j)$ al ranking ajustado asociados a los procesos que fallan (definiendo $r(0) = 0$), el ranking ajustado de la j-esima falla viene dado por la siguiente formula:

$$r(j) = r(j-1) + \frac{N + 1 - r\,(j-1)}{N + 1 - (k_j - 1)}$$

Luego, para el ejemplo de la figura 2 tenemos:

$$r(1) = r(0) + \frac{10 + 1 - r\,(0)}{10 + 1 - (6 - 1)} = 1{,}667$$

$$r(2) = r(1) + \frac{N + 1 - r\,(1)}{N + 1 - (7 - 1)} = 3{,}533$$

Y estos son los valores que debemos usar para el ajuste Weibull. Luego, en la ecuación 28 en vez de utilizar j, utilizamos $r(j)$ asociado a los procesos que efectivamente fallaron.

$$F(t_m) = \frac{r(j) - 0{,}3}{n + 0{,}4}$$

Finalmente, nuestras variables auxiliares para al ajuste se modifican.

$$x_j = \ln(t_j)$$

$$y_j = ln\left(ln\left(\frac{1}{1-\frac{r(j)-0{,}3}{n+0{,}4}}\right)\right)$$

Dejando la Ecuación 4 como

$$ln\left(ln\left(\frac{1}{1-\frac{r(j)-0{,}3}{n+0{,}4}}\right)\right) = \beta\ln(t) - \beta\ln(\alpha)$$

Ecuación 7. Ecuación Estimación Weibull modificada con rankeo

Y las Ecuación 5. Betaecuaciones Ecuación 5 y Ecuación 6 para los parámetros β y α respectivamente no se modifican.

2. Análisis Exponencial

El análisis para esta distribución es símil a la distribución Weibull, pero forma mucho más simplificada dado que esta distribución es un caso especial de la Weibull cuando β = 1. Con ello, la tasa de falla será constante y se define como:

$$\lambda(t) = \frac{1}{\lambda}$$

Luego, las funciones de densidad y probabilidad acumulada se define como, respectivamente:

$$f(t) = \lambda\, e^{-\lambda t}$$

$$F(t) = 1 - \, e^{-\lambda t}$$

Notar que en las formulas análogas con Weibull simplemente se remplaza β = 1 y α= 1/λ. Para el ajuste, realizamos los mismos procedimientos que en el análisis Weibull:

$$N(t) = \left(1 - \, e^{-\lambda t}\right)n$$

$$1 - \frac{N(t)}{n} = e^{-\lambda t}$$

$$ln\left(1 - \frac{N(t)}{n}\right) = -\lambda t$$

Utilizando el mismo concepto de rankeo explicado anteriormente obtenemos:

$$ln\left(1 - \frac{r(j) - 0{,}3}{n + 0.4}\right) = -\lambda t$$

Utilizando las siguientes variables auxiliares:

$$x = t$$

$$y = ln\left(1 - \frac{r(j) - 0{,}3}{n + 0.4}\right)$$

Obtenemos simplemente:

$$y = -\lambda x$$

Y aplicando regresión lineal, detallada en la sección del Apéndice 4.2, obtenemos:

$$\lambda = -\frac{\sum xy}{\sum x^2}$$

3. Regresión Lineal Simple

La regresión lineal simple se basa en solucionar el siguiente problema:

$$\min_{A,B} \sum (y_i - ye_i)^2$$

donde ye_i es el error de estimación y se puede modelar como

$$ye_i = Ax_i + B$$

xi es la variable independiente e y_i es la variable dependiente.

Lo que se desea es minimizar el error entre el valor de y_i, el cual es un dato del problema, respecto a la estimación $ye_i = Ax_i + B$, donde x_i tambien es un dato del problema. Finalmente, queremos

$$\min_{A,B} \sum (y_i - Ax_i - B)^2$$

Ecuación 8. Minimización del error cuadrático

Teniendo ambos conjuntos de datos $\bar{x}$ e $\bar{y}$ (de igual tamaño) es posible solucionar el problema planteado por la Ecuación 8 derivando la función de error parcialmente respecto A y B, igualando a cero y substituyendo el resultado de una ecuación sobre la otra.

Derivando parcialmente para A:

$$\begin{aligned}\frac{d}{\partial A}\sum (y_i - Ax_i - B)^2 &= 2\sum (y_i - Ax_i - B)\frac{d}{\partial A}(y_i - Ax_i - B)\\ &= 2\sum (y_i - Ax_i - B) * -x_i\end{aligned}$$

e igualando a cero:

$$\sum (y_i - Ax_i - B) * -x_i = 0$$

$$-\sum x_i y_i + A\sum x_i^2 + \sum Bx_i = 0$$

$$A = \frac{\sum x_i y_i - \sum Bx_i}{\sum x_i^2}$$

Derivando parcialmente para B:

$$\begin{aligned}\frac{d}{\partial B}\sum (y_i - Ax_i - B)^2 &= 2\sum (y_i - Ax_i - B)^2\frac{d}{\partial B}(y_i - Ax_i - B)\\ &= 2\sum (y_i - Ax_i - B) * -1\end{aligned}$$

e igualando a cero:

$$\sum (y_i - Ax_i - B) = 0$$

$$\sum y_i - A \sum x_i - nB = 0$$

$$B = \frac{\sum y_i - A \sum x_i}{n}$$

Uniendo las ecuaciones obtenidas en 64 y 70, obtenemos que:

$$A = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2}$$

$$B = \frac{\sum y_i - A \sum x_i}{n}$$

Para el caso de que nuestra ecuación lineal sea aun más simple, o sea, $ye_i = Ax_i$, el procedimiento es similar:

Derivando parcialmente para A:

$$\frac{d}{\partial B} \sum (y_i - Ax_i - B)^2 = 2 \sum (y_i - Ax_i) \frac{d}{\partial A} (y_i - Ax_i)$$
$$= 2 \sum (y_i - Ax_i) * -x_i$$

e igualando a cero:

$$\sum (y_i - Ax_i) * -x_i = 0$$

$$-\sum x_i y_i + A \sum x_i^2 = 0$$

$$A = \frac{\sum x_i y_i}{\sum x_i^2}$$

4. Tabla ANOVA

El análisis de la varianza parte de los conceptos de regresión lineal. De forma similar a lo explicado en la sección del anexo Regresión Lineal Simple,**Regresión Lineal Simple** el primer concepto fundamental es que todo valor observado puede expresarse mediante la siguiente función:

$$Y = B_0 + B_1 X + e$$

Ecuación 9: Modelo lineal simple

Donde Y es **el valor observado** (variable dependiente), y X el valor que toma la variable independiente. B_0 es una constante que, en la recta de regresión, equivale a la ordenada en el origen. B_1 es otra constante que equivale a la pendiente de la recta. e es una variable aleatoria que añade a la función cierto error el pronóstico del valor estimado del real.

Por tanto, a la **función de pronóstico** la podemos llamar Y':

$$Y' = B_0 + B_1 X$$

Ecuación 10. Función de pronóstico lineal

Siendo un poco más prácticos con la explicación, imaginemos que tenemos 2 variables que intentaremos explicar con un modelo lineal: la variable X y la variable Y, la última depende de la primera. Mas específicamente, nos referimos a que tenemos un cierto número de datos, pares (X,Y), o sea, datos reales que hemos obtenido de alguna manera, y queremos saber si existe una transformación matemática tal que al aplicarla a cualquier valor X del conjunto obtendremos un valor estimado Y'que es muy similar al valor observado Y, el cual sabemos que debemos obtener.

Si usamos el modelo de la Ecuación 10, hemos elegido dentro de toda la gama de modelos existentes, el modelo lineal simple. En tal caso, lo que deseamos en encontrar el valor de B_0y B_1 de tal forma el valor de pronóstico Y' sea lo más similar posible al valor real observado Y, ojala, en todos los casos en que podamos realizar una comparación. Es aquí donde entra el concepto de error cuadrático medio:

$$ECM = \sum (Y - Y')^2$$

Ecuación 11.Error cuadrático medio

Con este concepto, podemos ya implementar algún algoritmo que resuelva el problema. Por ejemplo podríamos realizar los siguientes pasos:

1. Elegir un valor al azar de B_0y B_1
2. Obtenemos Y e Y' para cada par de puntos (X,Y)
3. Calculamos la diferencia y la elevamos al cuadrado, o sea, calculamos el ECM.
4. Sumamos todos los ECM.

Con ello obtendremos un resultado, un número que nos dice cuan acertada es nuestra aproximación de los valores B_0y B_1que hemos elegido. Como toda métrica, este número por si sólo no nos sirve de nada, dado que solo tiene sentido si lo comparamos con otro que se obtuvo de la misma manera. Por ello, podríamos elegir nuevamente al azar otros valores B_0y B_1 y volver a realizar los pasos anteriormente descritos con la esperanza que el resultado final sea menor que el que obtuvimos al principio. Es bastante claro que este algoritmo, para nuestro problema, es bastante poco eficiente. Es aquí donde la estimación de los valores B_0y B_1 explicados en detalle en el anexo Regresión Lineal Simple es un algoritmo mucho mejor.

Pero lo que debemos tener completamente claro es que **asumimos que nuestro conjunto de puntos (X,Y) pueden ser explicados con un modelo lineal simple**, lo cual, puede no ser así. Entonces, de lo único que tenemos certeza es que **el modelo lineal simple nos entrega la mejor recta que se ajusta a nuestros datos**.

Ahora, lo que nos interesa saber es que si nuestra estimación utilizando el modelo lineal simple es lo suficientemente bueno como para poder utilizarlo de la manera que mejor nos acomode. Para ello, necesitamos realizar un análisis de la varianza de nuestro modelo.

Observando la Ecuación 9 y la Ecuación 10, es fácil ver que

Sabiendo este concepto, podemos operar con esta ecuación de la siguiente forma:

$$Y = Y' + e$$

La media de la variable Y será representada como $\overline{Y}$. Si restamos este valor a la ecuación anterior:

$$Y - \overline{Y} = Y' + e - \overline{Y}$$

Sabemos que el error e se define como

$$e = Y - Y'$$

Por tanto

$$Y - \overline{Y} = Y' + (Y - Y') - \overline{Y}$$

Y reorganizando la ecuación:

$$Y - \overline{Y} = (Y' - \overline{Y}) + (Y - Y')$$

Ahora hay que tener en cuenta que la media de los valores observados es exactamente igual a la media de los valores estimados (no se explicará este supuesto) , o sea, $\overline{Y} = \overline{Y'}$, obtenemos

$$Y - \overline{Y} = (Y' - \overline{Y'}) + (Y - Y')$$

Elevando al cuadrado y aplicamos sumatoria obtenemos que

$$\sum(Y-\overline{Y})^2=\sum((Y'-\overline{Y'})+(Y-Y'))^2$$

Desarrollando la ecuación

$$\sum(Y-\overline{Y})^2=\sum(Y'-\overline{Y'})^2+\sum(Y-Y')^2+\sum 2\,(Y'-\overline{Y'})(Y-Y')$$

El último término es una Suma Cruzada de Cuadrados (el numerador de la covarianza), y la covarianza en este caso es cero (por las propiedades de la regresión lineal, la covarianza entre el error y la variable independiente es cero).

Por tanto:

$$\overbrace{\sum(Y-\overline{Y})^2}^{ECG}=\overbrace{\sum(Y'-\overline{Y'})^2}^{ECE}+\overbrace{\sum(Y-Y')^2}^{ECM}$$

Donde ECG es el Error Cuadrático General (o Total) y ECE es el Error Cuadrático de Estimación (o de Tratamientos). Finalmente, la tabla ANOVA para nuestro caso particular de regresión lineal simple (que solo trabaja con 2 variables) será:

Fuente Variación	Suma Cuadrados	Grados Libertad	Cuadrado Medio	F
Intergrupo	ECE	1	$T=ECE$	$F_{TSV}=\frac{T}{E}$
Intragrupo	ECM	n – 2	$E=\frac{ECM}{n-2}$	
Total	ECG	n – 1		

En base a los innumerables estudios que se han realizado sobre el tema, sabemos que este valor F que obtenemos sigue un comportamiento que puede ser explicado por un distribución F de Fisher.

¿Pero que quiere decir esto? Imaginemos que realizamos un estudio de estatura de los niños prescolares de cierto colegio y descubrimos que la mayoría de ellos posee una estatura promedio de 70 cm con desviación standard de 5 cm y, por simplicidad, decidimos utilizar una curva normal con estos valores. Esto nos dice que es probable que un niño pequeño mida entre 55 y 85 cm, pero es muy poco probable tenga una estatura fuera de este intervalo. Supongamos que medimos a un individuo y obtenemos 1.80 m de estatura. Utilicemos únicamente el siguiente contraste de hipótesis (Wikipedia, Contraste de Hipotesis) para saber si el individuo es un niño o no:

- H0: el individuo es un niño

- H1: el individuo NO es un niño

Donde HO, conocida como Hipótesis nula (Wikipedia, Hipotesis nula), se cumplirá si la estatura del individuo probablemente pertenece al grupo definido por la distribución normal que estimamos anteriormente bajo un cierto **nivel de confianza**.

Se define **nivel de confianza** como el riesgo máximo que se acepta correr a la hora de rechazar H0, o sea, el riesgo de rechazarla cuando en realidad es cierta.

Si no se cumple H0, aceptaremos H1. Entonces, lo que podríamos hacer es calcular las estaturas límites según el nivel de confianza y la distribución:

- Existiría un limite inferior $E_{inferior}$ en base al **valor inverso de la CDF** de una distribución normal para un valor 0.05
- Existiría un limite superior $E_{superior}$ en base al **valor inverso de la CDF** de una distribución normal para un valor 0.95

Finalmente, podemos comparar la estatura que medimos y ver si pertenece al rango $[E_{inferior}, E_{superior}]$. Para nuestro caso, obtendríamos que el individuo probablemente no es un niño. En realidad, esta forma de saber si aceptamos H0 no se utiliza, dado que es mucho más sencillo definir un nivel de confianza, por ejemplo, un 90%, y luego obtener el valor CDF para el punto de la estatura del individuo y ver si este valor pertenece al rango [0.05, 0.95]. La idea final es comprender el proceso y entender que ambos formas son igual de válidas. Como nota aparte, este procedimiento se conoce como análisis de 2 colas, dado que se estudian los 2 bordes extremos de la distribución.

También es importante dejar en claro el concepto de **nivel de confianza** y se refiere más precisamente a cuanto estamos dispuestos a "equivocarnos". Supongamos que tenemos un niño de 83 cm. El pertenece al rango definido para los niños prescolares, pero su probabilidad es de 0.97 (un valor arbitrario para no complicar el tema y explicar el punto). Si utilizamos el mismo criterio anterior, un 90% de certeza, el test de hipótesis rechazaría a este niño, pues es muy alto para su edad y probablemente no sea un infante (falso positivo). Pero si utilizáramos un 95% de certeza, nuestro rango cambiaría a [0.025, 0.975] y se aceptaría H0.

No entraremos en discutir el tema de los falsos positivos o falsos positivos, pues no entran en el ámbito de este anexo, pero existe mucho información del tema en la literatura.

Habiendo comprendido los conceptos anteriores, introduciremos el concepto de **nivel de significancia α** que es simplemente el complemento de nuestro **nivel de confianza NC**

$$\alpha = 1 - NC$$

Nuestro contraste de linealidad se realizará bajo las siguientes hipótesis:

- H0: el modelo es lineal.
- H1: el modelo NO es lineal.

Nuestro indicador será el valor F_{TSV} obtenido de la tabla ANOVA y se conoce como Test Statistical Value (TSV). Si nuestro modelo es lineal, es probable que el indicador F_{TSV} tenga un comportamiento descrito por una función F de Fisher. Se dice que esta distribución es de una cola, dado que solo es necesario analizar el borde extremo derecho.

En términos muy coloquiales, la distribución de Fisher modela un comportamiento asociado a distancia relativa de nuestro modelo a uno perfectamente lineal. En estos términos, el mejor resultado que podríamos obtener del estadístico F_{TSV} es cero (la linealidad es perfecta). Como se puede ver, F_{TSV} será siempre un valor positivo, dado que la distancia relativa no puede menor que cero. Es por esto que solo se analiza el borde extremo derecho.

Para aceptar H0, existen 2 formas. Ambas se representan en la Ilustración 8. El valor crítico TSV se obtiene de la función inversa de CDF de Fisher para el valor 1 - p-value, a su vez, el valor CV se obtiene de la misma forma para el valor $1 - \alpha$.

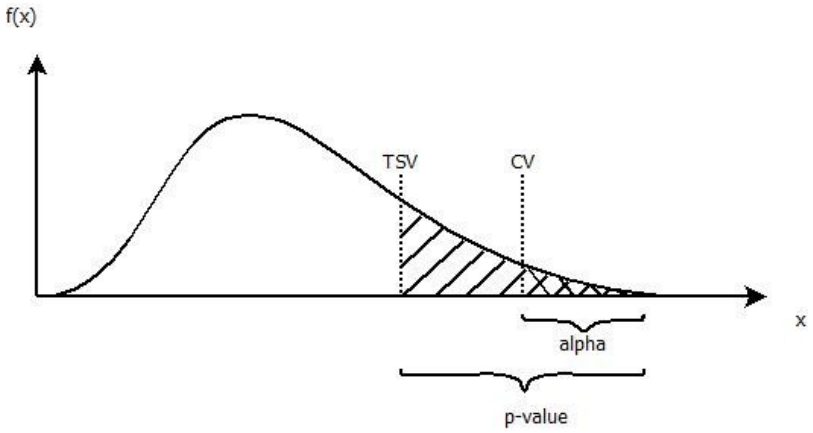

Ilustración 8. Valores críticos

Como se puede observar, mientras menor sea nuestro TSV, mayor certeza tendremos de que nuestro modelo es lineal. Por lo tanto, aceptaremos nuestra hipótesis nula H0 en cualquiera de estos dos casos:

- Si F_{TSV} es menor que F_{CV}
- Si p-value es mayor que α

En la práctica académica, generalmente se enseña la primera opción debido a que se disponen de tablas de la función de Fisher que aproximan los resultados. Computacionalmente, es común utilizar la segunda opción debido a que ocupa menos recursos del sistema.

SÓLO PARA USO INTERNO DE CGS

5. Test de Hipótesis

5.1. Introducción

Muchos métodos estadísticos asumen una distribución subyacente en la derivación de sus resultados. Tales métodos se denominan *paramétricos* (aquellos que no asumen una distribución subyacente en los datos se denominan *no paramétricos*).

Existen procedimientos formales para evaluar la correctitud de la distribución elegida y estos son los Test de Hipótesis, basados en la Teoría Estadística, los cuales generalmente son numéricamente complejos y requieren de algún tipo de soporte computacional.

Un Test de Hipótesis es una *regla* o *huincha de medir*, una herramienta que permite calcular la *distancia* que existe entre un conjunto de datos y una distribución candidata. El concepto de *distancia* lo define propiamente cada Test de Hipótesis, es así como el test ChiCuadrado utiliza su método particular para calcular distancia que difiere de Kolmogorov-Smirnov. Esta *distancia* formalmente se conoce como TSV (Test Statistical Value, o Valor Estadístico del Test).

Para saber cuán bueno o cuán malo es nuestro TSV existen distribuciones que rigen el comportamiento del estadístico. Por ejemplo, para el test ChiCuadrado, la distribución que rige al estadístico es una ChiCuadrado (de ahí su nombre). En sí, lo único que nos dice el test es si nuestro TSV es un valor común (probable) dentro de la distribución del estadístico.

Por ejemplo, pensar en el hecho de intentar decidir si a un animal es un humano únicamente analizando su altura. Es probable que mientras mayor sea la altura del animal que estemos midiendo (TSV), diremos con mayor seguridad de que el animal NO es un ser humano. Al contrario, si altura es pequeña, podremos decir que el animal tiene una probabilidad de ser un humano, o de ser un mono, o de ser un perro, etc. Lo único que nos dice el test es que si el animal es candidato a ser humano. Es en este punto es donde tiene sentido el término Valor Crítico (CV, Critical Value), asociado otro término muy importante que es el Nivel de Significancia α.

El Nivel de Significancia α permite obtener un CV para el test en particular que se este realizando. Supongamos que D(x) es la función de distribucion acumulada CDF del estadístico de distancia de nuestro test. La relacion entre CV y α es la siguiente:

$$D(x > CV) = \alpha$$

Es decir, la probabilidad de que un número se encuentre entre $[CV, \infty]$ es α. Si por ejemplo nuestro TSV se encuentra en este intervalo, es poco probable que el individuo u objeto asociado con al indicador pertenezca al conjunto definido por la variable D(x). Esto último, es la forma simple de definir el concepto de lo que se conoce como Hipótesis Nula **H0**. La negación de la Hipótesis Nula se conoce como Hipótesis Alternativa **H1**. Siguiendo con nuestro ejemplo, queremos saber si el animal podría ser humano o no, entonces diremos que:

- H0: el animal es un ser humano.
- H1: el animal NO es un ser humano.

Podemos decir, sin ser estrictos, que un valor crítico para la altura de un ser humano es de 2.15 [m] con un nivel de significancia de 0.05. Un ser que mida más que esta altura, probablemente no es humano. Se cumple que:

$$D_{humano}(x > 2.15) = 0.05$$

Supongamos que ahora tenemos otra categoría: los primates pequeños. Diremos que:

- H0: el animal es un primate.
- H1: el animal NO es un primate.

Un valor critico podría ser 1.40 [m] con el mismo Nivel de Significancia de 0.05. Si nuestro animal mide más de tal altura, es poco probable que sea un primate. Se cumple que:

$$D_{primate}(x > 1.40) = 0.05$$

Supongamos ahora que hemos tomado un ser aleatorio del reino animal. Medimos su altura y obtenemos 1.30 [m]. Aplicando ambos Test de Hipótesis, para los seres humanos y para los primates, los dos nos dirían que las respectivas H0 son verdaderas, o sea, que el animal puede ser tanto como un ser humano o como un mono. Finalmente, es en este punto es donde nace el último término importante denominado Valor P (P-Value, en inglés). El Valor P es la probabilidad que existe entre el intervalo $[TSV, \infty]$. Observemos la Ilustración 9.

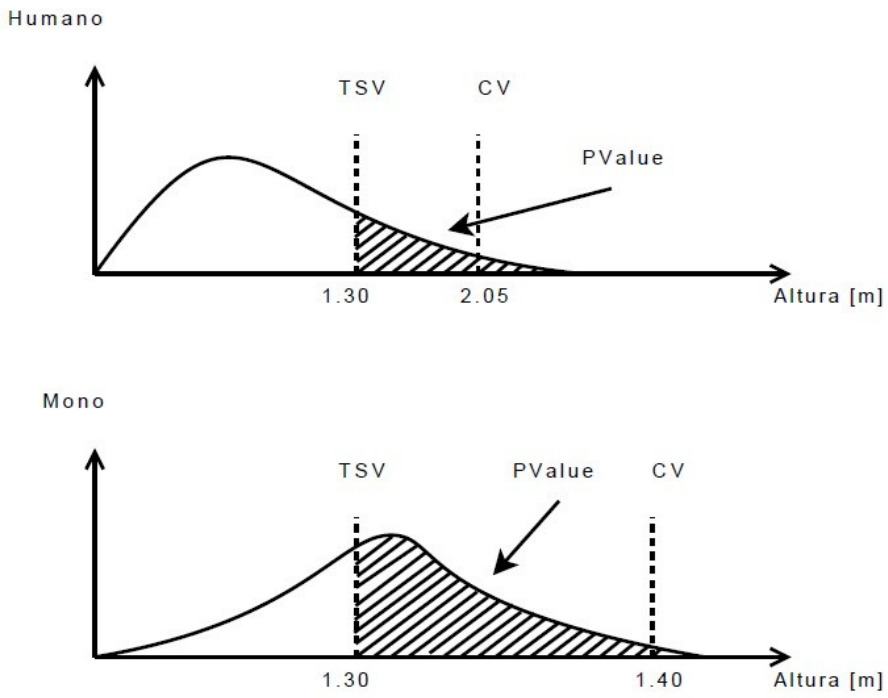

Ilustración 9. Humanos vs primates

Podemos ver que el área abarcada para la distribución asociada a los monos es mayor que el área asociada a la distribución de los seres humanos. O sea:

$$D_{primate}(x > 1.30) > D_{humano}(x > 1.30)$$

$$p - value_{primate} > p - value_{humano}$$

Diremos finalmente que es más probable que el animal se trate de un mono que de un ser humano. Por supuesto, podemos equivocarse y errar en el supuesto H0, pero es lo mejor que se puede hacer en base a la información que se tiene.

5.2. ChiCuadrado

El test de hipótesis ChiCuadrado cabe dentro de los procedimientos denominados *tests de área*, dado que utiliza la función de densidad (PDF) en el proceso.

Para evaluar los datos, existe un esquema bien definido. Primero, se asume que los datos provienen de cierta distribución (i.e: Normal). Luego, se estiman los parámetros de la distribución candidata (i.e: media y varianza) o simplemente se obtienen apriori de la experiencia. Tal proceso lleva a crear una hipótesis *compuesta* de distribución, dado que varios elementos juntos deben ser

verdaderos, la cual se denomina Hipótesis Nula H0, la negación de que los datos no se explican con distribución candidata se conoce como Hipótesis Alternativa H1. Luego, se testea nuestra distribución usando el conjunto de datos. Finalmente, H0 es rechazada si alguno (o varios) de los elementos en la hipótesis H0 no son soportados por los datos.

El test ChiCuadrado esta basado conceptualmente sobre la PDF de la distribución candidata. Si la distribución es correcta, ésta debería abarcar el rango del conjunto de datos de forma muy precisa (pensar en datos que se distribuyen como una normal N(0,1) y nosotros asumimos una normal N(1000,1000), claramente, los rangos no coinciden). De este rango, se eligen convenientemente valores que lo dividan en varios subintervalos. Luego, se calcula el número de puntos en cada intervalo (al igual que en un histograma de frecuencia absoluta). Estos puntos son denominados valores ''observados''. Luego, se calcula la cantidad de puntos que deberían haber caído en esos mismos subintervalos, de acuerdo a la PDF de la distribución candidata. Éstos son denominados valores ''esperados'' y el Test ChiCuadrado requiere de al menos 5 de ellos en cada subintervalo. Finalmente, se comparan estos 2 resultados. Si ellos concuerdan (probabilísticamente), entonces los datos soportan la distribución candidata. De lo contrario, se rechaza la suposición. La fórmula (estadístico o indicador que sigue algún comportamiento) utilizada para saber la diferencia entre los valores ''esperados'' y ''observados'' sigue una distribución ChiCuadrado. De ahí el nombre del test.

La fórmula para el estadístico ChiCuadrado es la siguiente:

$$\mathrm{X}^2 = \sum_{i=1}^{k} \frac{(e_i - o_i)^2}{e_i} \sim \mathrm{X}^2_{k-1-nep}$$

donde

- e_i es el número de datos esperados en el subintervalo i $(e_i > 5)$
- o_i es el número de datos observados en el subintervalo i
- K es el total de subintervalos en el rango
- n es el total de datos para la implementación del Test ChiCuadrado $(n \geq 5k)$
- nep es el número estimado de parámetros (depende de la distribución)
- k-1-nep son los grados de libertad o **DF** para la distribución ChiCuadrado $(DF > 0)$

De todo esto, se pueden desprender algunas observaciones:

- Por ejemplo, para una exponencial, la cantidad de subintervalos mínima k a escoger debe ser de 3, según la formula de DF donde $DF = k - 1 - neq > 0$
- Luego, teniendo el número de intervalos k, se debe cumplir también que $n \geq 5k$. O sea, que para nuestra misma exponecial deberíamos tener al menos 15 datos.
- Para una Weibull, el k mínimo es 4, por ende, la cantidad de datos mínima es de 20.

- Como conclusión, este test necesita de varios datos para realizar el procedimiento.
- La elección de los puntos que definen el inicio y el fin de los subintervalos es aparentemente arbitraria. No se exige que los subintervalo sean iguales en tamaño de rango, sólo se pide que en cada uno de ellos la cantidad de valores esperados sea mayor que 5. Además, estos puntos no necesariamente tiene que pertenecer al conjunto de datos. El test necesita de cierto criterio para la elección de subintervalos basado en la PDF/CDF de la distribución candidata.

Para claridad del test, se presentará el siguiente ejemplo. La Tabla 9 presenta datos ordenados que deseamos contrastar con una distribución Normal que asumimos como candidata. La Tabla 10 resume algunos indicadores para los datos.

6.1448	6.6921	6.7158	7.7342	9.6818	12.3317	12.5535	13.0973	13.6704
14.0077	14.7975	15.3237	15.5832	15.7808	15.7851	16.2981	16.3317	16.8147
16.8860	17.5166	17.5449	17.9186	18.5573	18.8089	19.2541	19.5172	19.7322
21.9602	23.2046	23.2625	23.7064	23.9296	24.8702	25.2669	26.1908	26.9989
27.4122	27.7297	28.0116	28.2206	28.5598	29.5209	30.0080	31.2306	32.5446

Tabla 9. Datos para Test de Bondad de Ajuste Normal

Variable	N	Media	Mediana	StdDev	Min	Max	Q1	Q3
Valor	45	19.50	18.56	7.05	6.14	32.54	15.06	25.73

Tabla 10. Estadísticas Descriptivas para ChiCuadrado

Según la Tabla 10, estimamos una distribución Normal N(19.5; 7.05). Dado que la distribución normal utiliza 2 parámetros, podemos calcular los grados de libertad de nuestro test ChiCuadrado como DF=k-1-2. Podemos elegir sin riesgo un valor para k=5, dado que DF > 0 (DF=2) y se cumple que $n \geq 5k$ (45≥ 15).

Luego, seleccionaremos arbitrariamente los puntos finales para cada subintervalo: 14, 17, 22 y 26. Ellos definirán 5 subintervalos o celdas, donde cada una contendrá más de 5 valores esperados.

La Tabla 11 muestra los resultados intermedios del algoritmo.

End	StdEnd	CumProb	CellProb	Expected	Observed	$(e-o)^2/e$
14	-0.78014	0.217654	0.217654	9.7944	9	0.06443
17	-0.35461	0.361441	0.143787	6.4704	10	1.92535
22	0.35461	0.638559	0.277118	12.4703	9	0.96574
26	0.92199	0.821732	0.183173	8.2428	6	0.61024
∞	∞	0.999999	0.178300	8.0235	11	1.10420
		Totales	1.000000	45.0010	45	4.67000

Tabla 11. Resultados Intermedios

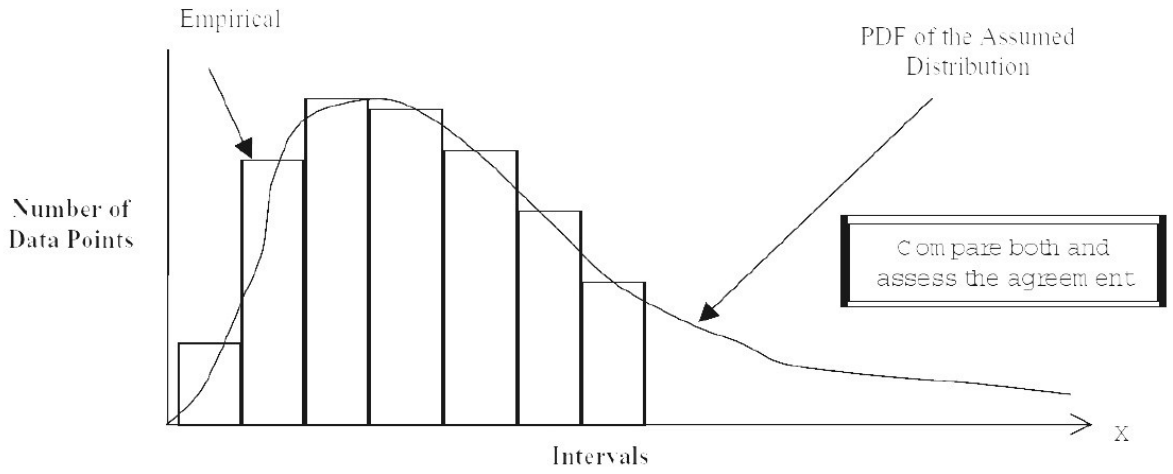

Ilustración 10. Aproximación Conceptual del Test ChiCuadrado

La primera columna muestra los puntos finales de cada subintervalo. La segunda, muestra el punto final estandarizado. La tercera, muestra sus valores acumulados (obtenidos desde una tabla Normal común y corriente). Por ejemplo, para el primer punto final (14), su valor estandarizado será:

$$StdEnd_1 = \frac{14 - 19.5}{7.05} = -0.78014$$

Y su valor acumulado (la probabilidad entre $[-\infty, 0.78014]$ será:

$$CumProb_1 = N_{19.5;7.05}(-0.78014) = 0.2176$$

En la cuarta columna se muestra la probabilidad del área de la celda, la cual se calcula restando las probabilidades acumuladas de la columna anterior inmediatamente adyacentes. Por ejemplo:

- $CellProb_1 = 0.217654 - 0 = 0.217654$
- $CellProb_2 = 0.361441 - 0.217654 = 0.143787$
- … y así con el resto.

La quinta columna contiene la multiplicación del área de la celda por la cantidad de datos (45) dando a lugar el valor esperado para cada celda. La sexta columna es el valor observado que se puede obtener directamente de la tabla Tabla 9 contando los valores la cantidad de valores que pertenecen al rango en estudio.

La última columna presenta el estadístico parcial para cada celda. La sumatoria de esta columna es el estadístico para el test:

$$X^2 = \sum_{i=1}^{k} \frac{(e_i - o_i)^2}{e_i} = 4.67$$

Este resultado se debe comparar con el valor crítico para ChiCuadrado. Este se obtiene directamente desde una tabla ChiCuadrado. Con un nivel de significancia α de 0.05 y con 2 grados de libertad el valor crítico es:

$$X_2^2(0.05) = 5.99$$

En vista de que el valor calculado 4.67 es menor que el valor crítico 5.99, se acepta la hipótesis nula y se dice que con un error del 5% es probable de que los datos se comporten como una N(19.5; 7.05).

5.3. Kolmogorov-Smirnov

A diferencia del Test ChiCuadrado, este procedimiento se denomina *test de distancia* dado que trabaja con la función de densidad acumulada (CDF). Al igual que el Test Anderson-Darling, es uno de los mejores test en presencia de pocos datos.

Su implementación sigue una serie bien definida de pasos. Primero, se asume una distribución candidata para los datos y se estima sus parámetros. Tal proceso conduce a la creación de la hipótesis nula H0, en donde varias de los supuestos de la hipótesis deberán ser verdaderos para que ésta sea aceptada. La negación será la hipótesis alternativa H1.

En un test de distancia, cuando la distribución candidata es correcta, la CDF teórica (denotada F_0) sigue muy de cerca a la Función de Densidad Acumulada Empírica, una función escalón de la CDF (denotada F_n). Esto se ilustra conceptualmente en la Ilustración 11. Aqui, los datos se encuentran ordenados, $\{x_1 \leq x_2 \leq \cdots \leq x_n\}$ y la distribución candidata tiene una CDF, F_0. Luego, se obtienen los correspondientes valores estadísticos para el test de bondad de ajuste (GoF). Si concuerdan, probabilísticamente, los datos soportan a la distribución candidata. De lo contrario, se rechaza la distribución. Existen varias formas de realizar el test KS. A continuación se explicará una de ellas.

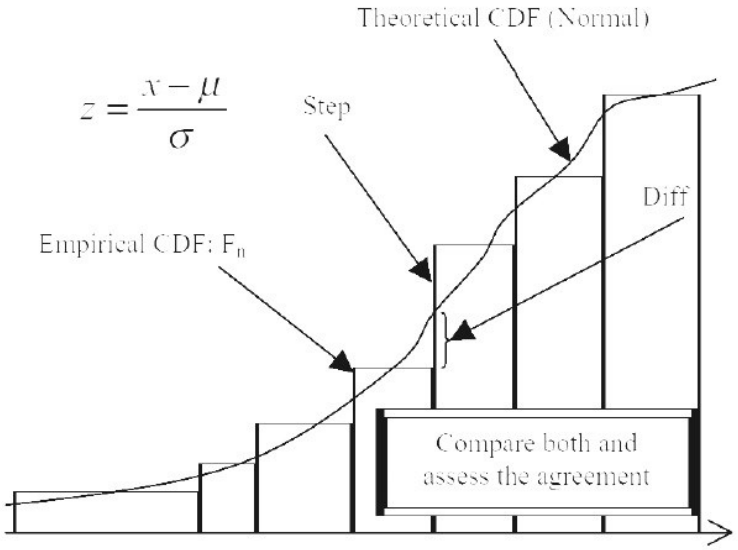

Ilustración 11. Aproximación Conceptual del Test Kolmogorov-Smirnov

Primero, se ordenan los datos y se obtiene los parámetros de la distribución candidata. Luego, se obtienen la función CDF teórica (F_0) y la función CDF empírica (F_n).

$F_0(x_i)$ es la función de distribución acumulada (de la distribución candidata) evaluada en el punto x_i y $F_n(x_i)$ es la función de distribuión acumulada empírica obtenida de la proporción de la cantidad de datos menores que x_i en el conjunto de datos de tamaño n.

Dado que KS es un test de distancia, necesitamos encontrar la distancia máxima entre las dos funciones $|F_0 - F_n|$. Estas se definen como:

$$F_0(x_i) = P_0(x < x_i) = CDF(x_i)$$

$$F_n(x_i) = \frac{\#\ de\ x's\ \leq x_i}{n} = \frac{i}{n} \quad i = 1, \dots, n$$

Definamos para cada punto x_i:

$$D^+ = F_n - F_0$$

$$D^- = F_0 - F_n$$

El estadístico KS se define como:

$$D = Maximo\ de\ todos\ los\ D^+ y\ D^-$$

La lógica de KS es la siguiente: si la máxima separación entre la CDF candidata y la CDF empírica es pequeña, entonces aceptamos que la distribución candidata explicada al conjunto de datos. De lo contrario, se rechaza.

Para una mejor compresión del test, se utilizará un ejemplo. La tabla \ref{tab:11} contiene seis datos con los que se trabajará durante la explicación del test. La tabla \ref{tab:12} contiene estadísticas de los datos. Se asumirá que los datos siguen una distribución Normal(μ,σ).

294.2	308.5	313.1	317.7	322.7	338.7

Tabla 12. Datos para Test KS GoF

Variable	n	Media	Mediana	StdDev
Datos	6	315.82	315.40	14.85

Tabla 13. Estadísticas Descriptivas para KS

Como ejemplo, calcularemos los valores para el primer punto de la tabla \ref{tab:11}. Asumiremos que los datos se comportan como una $N(315.82, 14.85)$. Esta será nuestra distribución candidata.

$$F_0(294.2) = N_{315.82;14.85}\left(\frac{294 - 315.82}{14.85}\right) = 0.0727$$

$$F_n(294.2) = \frac{1}{6} = 0.167$$

$$F_{n-1}(294.2) = \frac{0}{6} = 0$$

Los estadísticos D^+y D^- serán:

$$D^+(294.2) = F_n(294.2) - F_0(294.2) = 0.167 \ - \ 0.0727 \ = \ 0.0939$$

$$D^-(294.2) = F_0(294.2) - F_{n-1}(294.2) = 0.0727 - 0 \ = \ 0.0727$$

El resto de los valores se presenta en la tabla \ref{tab:13}.

Row	DataSet	F_0	F_n	F_{n-1}	D^+	D^-
1	294.1	0.072711	0.16667	0.00000	0.093955	0.072711
2	308.5	0.311031	0.33333	0.16667	0.022302	**0.144365**
3	313.1	0.427334	0.50000	0.33333	0.007266	0.094001
4	317.7	0.550371	0.66667	0.50000	0.116295	0.050371
5	322.7	0.678425	0.83333	0.66667	**0.154908**	0.011759

6	338.7	0.938310	1.00000	0.83333	0.061690	0.104977
					0.154908	0.144365

Tabla 14. Valores intermedios para test de normalidad con KS

El resultado final para el TSV de KS para este conjunto de datos y la distribución candidata es:

$$D = \max\{D^+, D^-\} = 0.1549$$

Este resultado se debe comparar con el valor crítico. Este se obtiene directamente desde una tabla Kolmogorov-Smirnov. Con un nivel de significancia α de 0.05 y con una cantidad de datos igual a 6 el valor crítico es:

$$KS_6(0.05) = 0.521$$

En vista a que el valor calculado 0.1549 es menor que el valor crítico 0.521 se acepta la hipótesis nula y se dice que con un error del 5% es probable de que los datos se comporten como una N(315.82, 14.85).

6. Rushdi

La explicación conceptual del algoritmo se encuentra en (Wiley). A continuación, sólo detallará lo esencial del algoritmo para su comprensión. La fórmula original está propuesta para configuraciones de tipo k-out-of-n:G. Es recursiva y se expresión es la siguiente:

$$R(i,j) = p_j R\,(i-1, j-1) + q_j R\,(i, j-1) \quad (1)$$

El término R(i , j) se refiere a la confiabilidad del sistema i-out-of-j:G, es decir, de una configuración que necesita de que al menos i componentes funcionen para que el sistema no se detenga. Para efectos prácticos, inicialmente sería la configuración en fraccionamiento/redundancia en estudio, en donde *i* es la cantidad de equipos que deben estar en funcionamiento del total *j* para que la configuración no falle. Definimos:

- p_j se refiere a la confiabilidad del componente *j* de la configuración.
- q_j se refiere al complemento de la confiabilidad p_j, $q_j = 1 - p_j$.

La recursión debe tener condiciones de borde para su detención. Estas condiciones son las siguientes:

$$R(0,j) = 1$$

$$R(j+1,j) = 0$$

Ilustraremos el algoritmo con un ejemplo. Se calculará la confiabilidad de una configuración 1-out-of-2:G, donde el primer componente tiene confiabilidad 0.8 y el segundo, 0.9. Entonces:

- $p_1 = 0{,}8$
- $p_2 = 0{,}9$
- $q_1 = 0{,}2$
- $q_2 = 0{,}1$

$$R(1,2) = 0{,}9 * R\,(0,1) + 0{,}1 * R\,(1,1) \quad (4)$$

$$R(0,1) = 1$$

$$R(1,1) = 0{,}8 * R\,(0,0) + 0{,}1 * R\,(1,0) \quad (5)$$

$$R(0,0) = 1$$

$$R(1,0) = 0$$

Luego

$$R(1,2) = 0{,}9 + 0{,}1 * 0{,}8 \quad (6)$$

$$R(1,2) = 0{,}98$$

Finalmente, la confiabilidad de la configuración 1-out-of-2:G es de 0,98.

El algoritmo más óptimo para la implementación de este cálculo sugiere el uso de una matriz dos dimensional que represente el diagrama de flujo de la figura 1.

La posición (i; j) representa la confiabilidad *R(i, j)*. Los nodos negros en la primera fila con i = 0 son los "nodos fuente" con valores iniciales igual a 1, o sea, *R(0 , j)* = 1. Los nodos blancos en la posición *i = j + 1* son "nodos fuente" con valor inicial 0, o sea, *R(j +1 , j)* = 0, ∀ j ≥ 0. Los valores de los otros nodos, por ejemplo, *(i , j),* se calculan sumando el producto la entrada superior izquierda por p_j al producto de la entrada izquierda por q_j .

El algoritmo actualmente implementado es el siguiente:

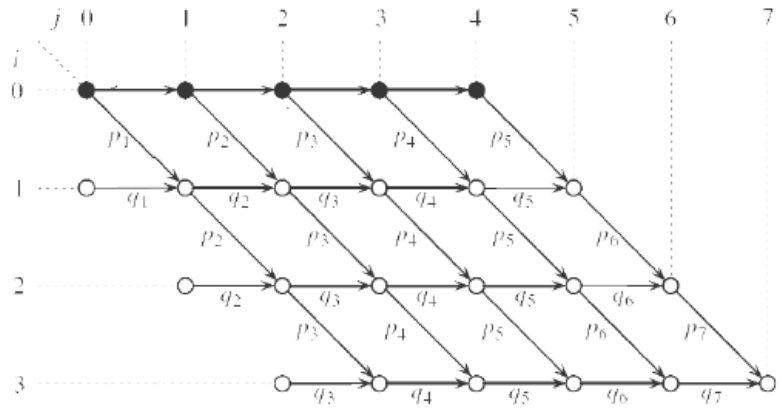

Ilustración 12. Diagrama de fiujo obtenido para R(3,7)

```
double array[ ] [ ] = new double[k + 1][n + 1];
for (int  j = 0; j < n; j++) {
        array[0][j] = 1.0;
}
for (int i = 1; i <= k; i++) {
        for (int j = i; j <= n; j++) {
                double p = ss.getSubsystem(j - 1).R(t);
                array[i][j] = p * array[i - 1][j - 1] + (1 - p) * array[i][j - 1];
        }
}
return array[k][n];
```

Se puede observar que el valor de retorno es el del último nodo de la matriz. Los valores k y n representan a la configuración k-out-of-n:G. La variable *p* almacena la confiabilidad del nodo hijo

$j-1$. Esta confiabilidad es la calculada en el tiempo t. También es posible utilizar la confiabilidad por defecto que se calcula con un *t* igual al tiempo de operación de la configuración.

7. Rushdi Optimizado

Este es el algoritmo Rushdi que realmente esta implementado en RMES para cálculo de confiabilidad de sistemas redundantes. Mejora la performance y el consumo de memoria respecto al original. Pero antes de entrar en detalle, es necesario comprender a cabalidad el original (referirse al anexo Rushdi). A continuación, se explicará el algoritmo original paso a paso y las mejoras que se pueden realizar.

Digamos que se tiene una configuración 3-out-of-4:G. Supongamos que los valores de la confiabilidad de las 4 configuraciones se conocen y se almacenan en el vector "P":

$$P = [p_0 \quad p_1 \quad p_2 \quad p_3 \ ]$$

El algoritmo original crearía la siguiente matriz A:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Y luego igualaría la primera fila a 1.

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Después, se comenzaría por calcular la celda $a_{1,1}$ usando los valores de las celdas $a_{0,0}$ y $a_{1,0}$ junto con el valor p_0 del vector:

$$a_{1,1} = p_0 * a_{0,0} + (1 - p_0) * a_{1,0}$$

Notar que $a_{0,i}$ = 1 y $a_{1,0}$ = 0, por lo tanto, para este caso particular $a_{1,1} = p_0$

$$A = \begin{pmatrix} a_{0,0} & 1 & 1 & 1 & 1 \\ a_{1,0} & a_{1,1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Continuando con el algoritmo, se procede a calcular la siguiente celda, $a_{1,2}$, el cual se obtendrá de las celdas $a_{0,1}$ y $a_{1,1}$ junto con el valor p_1 del vector:

$$a_{1,2} = p_1 * a_{0,1} + (1 - p_1) * a_{1,1}$$

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & 1 & 1 & 1 \\ a_{1,0} & a_{1,1} & a_{1,2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Finalmente las últimas celdas se calcularían de la siguiente manera:

$$a_{1,3} = p_2 * a_{0,2} + (1 - p_2) * a_{1,2}$$

$$a_{1,4} = p_3 * a_{0,3} + (1 - p_3) * a_{1,3}$$

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & 1 \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

En la siguiente iteración, el cálculo comienza en la celda $a_{2,2}$ y algoritmo es idéntico:

$$a_{2,2} = p_1 * a_{1,1} + (1 - p_1) * a_{2,1}$$

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & 1 \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,1} & a_{2,2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$a_{2,3} = p_2 * a_{1,2} + (1 - p_2) * a_{2,2}$$

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & 1 \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,1} & a_{2,2} & a_{2,3} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$a_{2,4} = p_3 * a_{1,3} + (1 - p_3) * a_{2,3}$$

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & 1 \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Y en la siguiente y última iteración el cálculo comienza en la celda $a_{3,3}$ hasta llegar a la celda $a_{3,4}$ valor que retorna el algoritmo.

$$a_{3,3} = p_2 * a_{2,2} + (1 - p_2) * a_{3,2}$$

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & 1 \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ 0 & 0 & a_{3,2} & a_{3,3} & 0 \end{pmatrix}$$

$$a_{3,4} = p_3 * a_{2,3} + (1 - p_3) * a_{3,3}$$

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & 1 \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ 0 & 0 & a_{3,2} & a_{3,3} & a_{3,4} \end{pmatrix}$$

De todo el proceso se pueden desprender que en cada paso unitario del algoritmo se necesita únicamente saber el valor de la celda superior izquierda y de la celda inmediatamente izquierda respecto de la actual. Mientras el algoritmo avanza, los valores que se calcularon para las primeras filas no se vuelven a utilizar. Esto lleva a pensar que sólo es necesario utilizar un vector de tamaño *n* (la cantidad de nodos hijos) que almacene los valores por cada iteración.

Teniendo presente este esquema, es necesario aislar la primera iteración del resto, debido a que los valores iniciales de las celdas superiores deben ser igual a 1, lo que implica que la fórmula puede ser modificada para este caso particular. Ubicaremos gráficamente un vector V dentro la matriz A para observar el comportamiento del algoritmo modificado. El vector V almacenará los valores del algoritmo.

$$V = [v_0 \quad v_1 \quad v_2 \quad v_3]$$

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & v_0 & v_1 & v_2 & v_3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Para la primera iteración se cumple que $a_{0,i}$ = 1, por lo tanto la fórmula puede ser modificada de la siguiente manera:

$$a_{1,j} = p_{j-1} + (1 - p_{j-1}) * a_{1,j-1} \quad \forall j = 1 \dots n$$

Reemplazando para el vector V:

$$v_j = p_j + (1 - p_j) * a_{1,j} \quad \forall j = 0 \dots n-1$$

Para eliminar la dependencia con la matriz A, el valor de la celda $a_{1,j}$ simplemente es el valor de la celda calculada en el paso anterior, la cual puede ser referenciada directamente. Para primer elemento, v_0 v_{-1} no existe y se considera con valor 0, por ende, la fórmula puede ser aún más concisa.

$$v_0 = p_j$$

$$v_j = p_j + (1 - p_j) * v_{j-1} \quad \forall j = 0 \dots n-1$$

Gráficamente, el algoritmo seria el siguiente:

$$v_0 = p_0$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & v_0 & - & - & - \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$v_1 = p_1 + (1 - p_1) * v_0$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & v_0 & v_1 & - & - \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$v_2 = p_2 + (1 - p_2) * v_1$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & v_0 & v_1 & v_2 & - \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$v_3 = p_3 + (1 - p_3) * v_2$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & v_0 & v_1 & v_2 & v_3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Para la siguiente iteración, es necesario modificar el criterio anterior. El valor de la celda v_1 es el que debe ser calculado, entonces, siguiendo la fórmula:

$$v_1 = p_1 * \hat{v}_0 + (1 - p_1) * v_0$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & \hat{v}_0 & - & v_2 & v_3 \\ 0 & v_0 & v_1 & - & - \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

En donde $\hat{v}_i$ se refiere al valor antiguo de la celda *i* del vector V antes de ser calculada. Particularmente, para el primer paso de la iteración, no se realiza cálculo sobre v_0, además, su valor es v_0 = 0, y se debe de recordar que $\hat{v}_i$ tiene un valor calculado en la iteración anterior, por lo tanto, sólo es necesario modificar la fórmula:

$$v_1 = p_1 * \hat{v}_0$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & \hat{v}_0 & - & v_2 & v_3 \\ 0 & 0 & v_1 & - & - \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

El problema de esto es que se pierde el valor original de v_1, el cual se usa en el paso siguiente de la iteración. Por lo tanto, el valor de v_1 debe ser almacenado en una variable auxiliar antes de realizar el cálculo. Esta variable se llamará "actualOldValue", o AOV

$$AOV = v_1$$

$$v_1 = p_1 * \hat{v}_0$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & \hat{v}_0 & AOV & v_2 & v_3 \\ 0 & 0 & v_1 & - & - \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Para la siguiente iteración, es necesario utilizar el valor *AOV* y v_1 para calcular el nuevo valor de la celda v_2, pero también es necesario almacenar el valor de la celda v_2 anterior al cálculo, dado que se necesita el valor antiguo para el cálculo de la siguiente celda en la iteración. Se usará la variable AOV para almacenar el valor de v_2 previo al cálculo, pero además, se debe almacenar el valor que AOV tendrá almacenado. Para ello, se usará la variable "upperOldValue", o UOV. Esta se usará para realizar el cálculo de la celda actual v_2. Entonces:

$$UOV = AOV$$

$$AOV = v_2$$

$$v_2 = p_2 * UOV + (1 - p_2) * v_1$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & \hat{v}_0 & UOV & AOV & v_3 \\ 0 & 0 & v_1 & v_2 & - \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Para la siguiente iteración, no es necesario agregar más variables, sólo actualizar las existentes:

$$UOV = AOV$$

$$AOV = v_3$$

$$v_3 = p_3 * UOV + (1 - p_3) * v_2$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & \hat{v}_0 & - & UOV & AOV \\ 0 & 0 & v_1 & v_2 & v_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Para la última iteración, se ocupa el mismo criterio anterior. Sabemos que $\mathbf{v_1} = 0$, entonces el primer paso de la iteración sería:

$$AOV = v_2$$

$$v_2 = p_2 * \hat{v}_1$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & - & - & - & - \\ 0 & - & \hat{v}_1 & AOV & v_3 \\ 0 & - & 0 & v_2 & 0 \end{pmatrix}$$

El último paso de la iteración sería:

$$UOV = AOV$$

$$AOV = v_3$$

$$v_3 = p_3 * UOV + (1 - p_3) * v_2$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & - & - & - & - \\ 0 & - & \hat{v}_1 & UOV & AOV \\ 0 & 0 & 0 & v_2 & v_3 \end{pmatrix}$$

El código de este algoritmo es el siguiente:

```
private static double rushdiGeneral(double[ ] ssR, int k) {

    int n = ssR.length;
    double upperOldValue;
    double actualOldValue;
    double array[ ] = new double[n];
    int i, j;


    array[0] = ssR[0];
    for (j = 1; j < n;  j++)  {
    array[j] = ssR[j] + (1 - ssR[j]) * array[j - 1];
    }


    for (i = 1; i < k; i++) {
        actualOldValue = array[i];
        array[i] = ssR[i] * array[i - 1];
```


```
        for (j = i + 1; j < n; j++) {
            upperOldValue = actualOldValue;
            actualOldValue = array[j];
            array[j] = ssR[j] * upperOldValue + (1 - ssR[j]) * array[j - 1];
        }

    }

return array[n - 1];

}
```

En donde *ssR* es el vector de contiene la confiabilidad de los hijos de la máquina y k es la mínima cantidad de equipos en operación. El valor de k puede ir entre 1 y n, donde n es la cantidad de elementos del vector *ssR*. Para el caso particular k = 1 se implementó un algoritmo recursivo que tiene mejor performance que el presente algoritmo, pero únicamente para un valor de k = 1, para cualquier otro valor, su performance es pobre y de hasta menor calidad que el algoritmo rushdi original.

El código se basa en la siguiente fórmula:

$$R(i,j) = p_j R\,(i-1, j-1) + \; q_j R\,(i, j-1)$$

Y esta es su implementación:

```
private static double rushdiOneMinToWork(double[] ssR, int n)
        if (n == 0) {
            return 0;
        }
    return ssR[n - 1] + (1 - ssR[n - 1]) * rushdiOneMinToWork(ssR, n - 1);
}
```

En donde *ssR* es el vector de contiene la confiabilidad de los hijos de la máquina y *n* un parámetro que varía con la recursión, inicialmente es la cantidad de elementos del vector *ssR.*