Risk Unifying Frequency Functional

Technical Report 2015-003
June 11, 2015

Graduate Program in Operations Research & Industrial Engineering
The University of Texas at Austin
Austin, Texas, USA
1 Introduction

As part of the response to General Safety Issue 191 (GSI-191), the South Texas Project Nuclear Operating Company (STPNOC) wishes to perform an assessment of the estimated frequency of loss of coolant accident (LOCA) events that fall into a risk-informed category. In particular, they wish to determine the frequency of LOCA events that would generate a quantity of fiber fines that exceed a given threshold. The Risk Unifying Frequency Functional (RUFF) detailed in this report describes a method for estimating this frequency using as input a set of welds that generate and transport enough fiber fines to exceed the threshold, as well as the minimum break size that produces an amount of fines greater than the threshold.

Section 2 describes the sources of data for LOCA frequency by break size and the set of welds for which a break of some size produces an amount of fiber fines in excess of the given threshold. Section 3 details the calculation method used to find the total frequency or breaks that fall into the risk-informed category. Finally, Section 4 presents the source code of an implementation of RUFF in Python.

2 Data Sources

2.1 LOCA Frequencies

Tregoning, Abramson, and Scott [1] determined a plant-wide frequency and impact (in terms of flow rate) of different size breaks in PWRs and BWRs through elicitation of experts. The flow rates in the elicitation were converted to break diameters and presented in tables as frequencies by break size. Tregoning et al. are widely referenced for LOCA frequencies used in plant PRAs (see [2] for PRA LOCA initiating event frequency prior data). Table 1 presents the LOCA frequencies by break size used in RUFF.

Table 1: NUREG-1829 (Tregoning et al., 2008, Table 7.19) estimates for the mean, median, 5th percentile, and 95th percentile exceedance frequency values estimates for 25-year plant operation, using geometric means aggregation. RUFF can also accept estimates using 40-year plant operation and/or arithmetic means.

<table>
<thead>
<tr>
<th>Break Size (in.)</th>
<th>Category</th>
<th>5th %ile</th>
<th>50th %ile</th>
<th>Mean</th>
<th>95th %ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.500</td>
<td>1</td>
<td>6.80E-05</td>
<td>6.30E-04</td>
<td>1.90E-03</td>
<td>7.10E-03</td>
</tr>
<tr>
<td>1.625</td>
<td>2</td>
<td>5.00E-06</td>
<td>8.90E-05</td>
<td>4.20E-04</td>
<td>1.60E-03</td>
</tr>
<tr>
<td>3.000</td>
<td>3</td>
<td>2.10E-07</td>
<td>3.40E-06</td>
<td>1.60E-05</td>
<td>6.10E-05</td>
</tr>
<tr>
<td>7.000</td>
<td>4</td>
<td>1.40E-08</td>
<td>3.10E-07</td>
<td>1.60E-06</td>
<td>6.10E-06</td>
</tr>
<tr>
<td>14.000</td>
<td>5</td>
<td>4.10E-10</td>
<td>1.20E-08</td>
<td>2.00E-07</td>
<td>5.80E-07</td>
</tr>
<tr>
<td>31.000</td>
<td>6</td>
<td>3.50E-11</td>
<td>1.20E-09</td>
<td>2.90E-08</td>
<td>8.10E-08</td>
</tr>
</tbody>
</table>

2.2 Welds and Break Sizes

CASA Grande is a software tool that was developed to analyze accident sequences in a realistic time-dependent manner to determine the probabilities of various failures potentially leading to core damage from a spectrum of location-specific pipe breaks (i.e., LOCA events) [3]. As a preprocessing step to RUFF, CASA
Grande runs are performed to identify all weld locations, and the corresponding smallest break sizes that produce more than the allowable amount of fiber fines.

3 RUFF Methodology

We start with the list of welds and corresponding break sizes taken from CASA Grande as described with presented Section 2.2. With this step completed, we have data that can be thought of as ordered pairs consisting of a weld index and a break size. For now, assume that \( I \) weld locations are in the risk-informed category and these locations are indexed by \( i = 1, \ldots, I \). Each weld location \( i \) then has a corresponding break size \( D_{\text{small}}^i \) which caused it to be placed in the risk-informed category. It is possible that for a single weld, multiple break scenarios caused it to be put in this category. If so, define \( D_{\text{small}}^i \) to be the smallest such break size.

The goal is to determine the overall frequency of events that generate too many fiber fines. First, for each weld \( i \) in the risk-informed category, the goal is to determine the frequency of breaks that exceed \( D_{\text{small}}^i \). This is called \( F(D_{\text{small}}^i) \) and is the frequency of unacceptable events caused by that particular weld. Then, the overall frequency of unacceptable events caused by breaks in the risk-informed category is simply the sum of these frequencies:

\[
\Phi = \sum_{i=1}^{I} F(D_{\text{small}}^i).
\]

The three factors that determine the frequency of breaks at any particular weld \( F(D_{\text{small}}^i) \) for the purposes of our calculation are:

- the percentile of the probability distribution that governs LOCA frequency by break size;
- the smallest break size for the weld that generates a sufficient amount of fiber fines \( D_{\text{small}}^i \); and,
- the conditional probability that the break of size \( D_{\text{small}}^i \) came from a specific weld.

Using a given percentile for initiating frequencies as input, we calculate the frequency of breaks of a given size across the STPNOC plant, \( f(D_{\text{small}}^i) \), by looking up the value under the appropriate percentile in Table 1. In the event that a break size falls between two categories, linear interpolation is used to estimate the frequency of breaks of size \( D_{\text{small}}^i \) across the plant; this is a conservative method of estimation of frequency, as shown in [4]. In the event that a percentile other than the 5th, 50th or 95th is assessed, we can use a Johnson distribution fit to the three percentiles by category to obtain estimates for a given percentile. The methodology for fitting Johnson percentiles to the NUREG-1829 percentiles is shown in detail in [5].

Because a partial break of a weld is possible, any weld whose inner diameter is of size \( D_{\text{small}}^i \) or greater is able to generate a break of size \( D_{\text{small}}^i \). We assume that any such weld in the plant has an equal likelihood of breaking, ignoring the GSI-191 assumptions. Therefore, let \( TW_i \) be the total number of welds of size \( D_{\text{small}}^i \) or greater. Then, the conditional probability that weld \( i \) is responsible for a break of size \( D_{\text{small}}^i \) is

\[
P_i(D_{\text{small}}^i) = \frac{1}{TW_i}.
\]

Therefore, the frequency of LOCA events that generate a sufficient amount of fiber fines to be a risk-informed event at weld \( i \) is

\[
F(D_{\text{small}}^i) = f(D_{\text{small}}^i)P_i(D_{\text{small}}^i) = \frac{f(D_{\text{small}}^i)}{TW_i},
\]

and the total risk-informed event frequency \( R \) is therefore

\[
\Phi = \sum_{i=1}^{I} F(D_{\text{small}}^i) = \sum_{i=1}^{I} \frac{f(D_{\text{small}}^i)}{TW_i}.
\]
4 Source Code

This implementation of RUFF is a Python script developed at the University of Texas at Austin under STPNOC grant BO4425. The open-source PANDAS library (http://pandas.pydata.org/) is used in the FIDOE implementation.

Risk Unifying Frequency Functional (RUFF)
Alex Zolan
Updated April 20, 2015

The purpose of the program is to estimate the frequency of critical breaks that can occur. We assume that any pipe that has a diameter as large or larger than any critical break size could experience such a break, and that each possible pipe has the same chance of having such a break.

import pandas
import scipy
import scipy.stats
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

class NUREG_1829_Freqs(object):
    
    def __init__(self, nureg_file):
        
        df = pandas.read_csv(nureg_file)
        self.categories = df.Category.values
        self.sizes = df.Break_Size.values
        self.means = df.Mean.values
        self.P5 = df.P5.values
        self.P50 = df.P50.values
        self.P95 = df.P95.values
        if 'gamma' in df.columns.values:
            self.gamma = df.gamma.values
        else: self.gamma = None
        if 'delta' in df.columns.values:
            self.delta = df.delta.values
        else: self.delta = None
        if 'xi' in df.columns.values:
            self.xi = df.xi.values
        else: self.xi = None
        if 'lamb' in df.columns.values:
            self.lamb = df.lamb.values
        else: self.lamb = None
def findFirstExceedingIndex(self, size):
    """Finds the index of the first weld size that is larger than the given input.
    size -- break size, in inches
    retval - index from sizes object"""
    assert size >= self.sizes[0], "Size outside of NUREG."
    for idx, s in enumerate(self.sizes):
        if s >= size: return idx
    return -1

def getFrequency(self, size, stat):
    """Returns the exceedance frequency of a given break size uses NUREG 1829 values and linear interpolation to find the estimate for that category.
    size -- break size, in inches
    stat -- desired summary statistic
    retval - summary statistic frequency for break size"""
    idx = self.findFirstExceedingIndex(size)
    assert idx >= 0, "Size outside of NUREG Found. Aborting."
    if idx == 0: return self.getStat(0, stat)
    lower = self.getStat(idx-1, stat)
    upper = self.getStat(idx, stat)
    frac = (size-self.sizes[idx-1])/(self.sizes[idx] - self.sizes[idx-1])
    return lower + (upper-lower)*frac

def getStat(self, idx, stats):
    """Returns a summary statistic or percentile for a given NUREG Category,
    based on the object desired.
    idx -- index of the desire list to return
    stats -- list/array of desired summary statistics - this could be from the table of initial values given, or it could be an array of percentile (float)
    retval - frequency from NUREG-1829, or Johnson Distribution percentile"""
    if stats == ['P5']: return self.P5[idx]
    if stats == ['P50']: return self.P50[idx]
    if stats == ['P95']: return self.P95[idx]
    if stats == ['Mean']: return self.means[idx]
    elif self.lamb != None:
        return scipy.stats.johnsonsb.ppf(stats,
                                           self.gamma[idx],
                                           self.delta[idx],
                                           loc=self.xi[idx],
                                           scale=self.lamb[idx])
    else: assert False, "No distribution information given."

class LOCAEventCalculator(object):
    """This class acts as the calculator for LOCA Events. It calls
frequencies from the NUREG_1829_Freqs object, and determines
the probability of a particular pipe breaking by finding the
number of pipes that could handle such a break.

breaksFile -- location of the file that contains all pipes and
the weld break sizes that would cause a significant event
weldsFile -- location of the file that contains a summary of
the number of welds of each size/type

```
def __init__(self,breaksFile,weldsFile):
    self.breaks_df = pandas.read_csv(breaksFile)
    self.welds_df = pandas.read_csv(weldsFile)

def getPipesOfExceedingSize(self,breakSize):
    """returns the number of pipes in from the welds dataframe
that have a diameter that meets or exceeds a given break
size, given the input breakSize."""
    return scipy.sum(self.welds_df[self.welds_df.pipe_type
        >= breakSize].number_of_welds.values)

def getSumOfAllBreaks(self,stats,nuregFile):
    """calculates the expected frequency of LOCA events based
on calculating the exceedance frequency of the break size
and then dividing by the number of pipes that could have
a break of that size in the plant (as given by the welds
file). This term is calculated for each pipe in the
pipebreaks file (when a nonzero break size is included)
and then summed to get the result.

nuregFile -- table of NUREG-1829 frequencies.
retval -- expected frequency of LOCA events/CY."""
nureg = NUREG_1829_Freqs(nuregFile)
if len(stats) == 1: sum_freqs = 0.0
else: sum_freqs = scipy.float64(0.0)*stats
for i,rowdata in self.breaks_df.iterrows():
    if self.breaks_df.Break_size[i] == 0: continue
    breakFreq = nureg.getFrequency(
        self.breaks_df.Break_size[i],stats
    )
    numPipes = self.getPipesOfExceedingSize(
        self.breaks_df.Break_size[i]
    )
    sum_freqs += breakFreq / numPipes
    #print self.breaks_df.Break_size[i],breakFreq,numPipes
return sum_freqs

def getStratifiedSamplingEstimate(self,samples,
    nuregFile,pctCI):
    """Returns a point and interval estimate of deltaCDF using
the underlying Johnson distributions assumed to govern
LOCA frequency."""
    #samples = getUniformStratifiedSamples(strata,
    #    samplesPerStrata)
cdfs = self.getSumOfAllBreaks(samples, nuregFile)
var = scipy.sum(scipy.var(cdfs, 0) / len(samples[0]))
pointEst = scipy.mean(cdfs)
HW = scipy.norm.ppf(1 - (1 - pctCI) / 2) * scipy.sqrt(var)
return pointEst, var, HW

def getUniformStratifiedSamples(strata, samplesPerStrata):
    """Uses Python’s RNG to create random uniforms in a stratified sampling format, in 2D array form. Assumes uniform strata.
strata - number of bins to create samples
samplesPerStrata - number of samples per strata
retval - 2D array of uniform random numbers, organized by strata in 'strata' rows, 'samples per strata' columns""
    lb = scipy.arange(strata) * (1.0 / strata)
    lb = lb.reshape((strata, 1))  # lower bound of each bin
    ub = lb + 1.0 / strata  # upper bound of each bin
    x = scipy.random.sample(strata * samplesPerStrata).reshape(strata, samplesPerStrata)
    x = x * (ub - lb) + lb
    return x

if __name__ == "__main__":
    weldsFile = raw_input("Please enter the name of the welds inputs file: ")
    breaksFile = raw_input("Please enter the name of the pipe/break sizes file: ")
    nuregFile = raw_input("Please enter the name of the NUREG frequencies file: ")
    locas = LOCAEventCalculator(breaksFile, weldsFile)
P5Freq = locas.getSumOfAllBreaks(["P5"], nuregFile)
P50Freq = locas.getSumOfAllBreaks(["P50"], nuregFile)
P95Freq = locas.getSumOfAllBreaks(["P95"], nuregFile)
meanFreq = locas.getSumOfAllBreaks(["Mean"], nuregFile)
print ("Total expected frequency of events at NUREG P5: "+
str(P5Freq)+" events/CY")
print ("Total expected frequency of events at NUREG P50: "+
str(P50Freq)+" events/CY")
print ("Total expected frequency of events at NUREG P95: "+
str(P95Freq)+" events/CY")
print ("Total expected frequency of events at NUREG Mean: "+
str(meanFreq)+" events/CY")
# Create Stratified Sampling Estimates. We will use CRNs.
try:
samples = getUniformStratifiedSamples(50, 1000)
cdfs = locas.getSumOfAllBreaks(samples, nuregFile)
# We have uniform bins, and the same number of samples per bin, so this is just the mean of all observations.
pointEst = scipy.mean(cdfs)
# Variance = sum of (P(bin)"* "var(bin) / (samples in bin))
var = ((1.0 / len(samples)**2) * scipy.sum(scipy.var(cdfs, 1))
    / len(samples[0]))
HW = (scipy.stats.norm.ppf(1 - 0.05 / 2) *
    scipy.sqrt(var))  # 95% CIs
interval_lb = pointEst - HW
interval_ub = pointEst + HW
print "Johnson Parameters included in input file."
print ("Point estimate for delta-CDF for 50 bins, "+
"1000 samples per bin:\n"+str(pointEst)+
" events/CY")
print ("95\% CI on frequency of events under Johnson "+
"distribution (events/CY):\n"+str(interval_lb)+
","+str(interval_ub)+")
except AssertionError:
    print "No Johnson parameters given - no CI can be output."

References


