Towards Efficient Ways of Estimating Failure Probability of Mechanical Structures Under Interval Uncertainty

Michael Beer[1], Marco De Angelis[1], and Vladik Kreinovich[2]

[1]Institute for Risk and Uncertainty, School of Engineering
University of Liverpool, UK
mbeer@liverpool.ac.uk, Marco.De-Angelis@liverpool.ac.uk
[2]Department of Computer Science, University of Texas at El Paso
El Paso, TX 79968, USA, vladik@utep.edu

1. Case of Full Knowledge: Reminder

- The textbook approach to estimating the failure probability of mechanical structures assumes that:
  - we know the pdf $f(\theta)$ of quantities $\theta = (\theta_1, \ldots, \theta_n)$ describing the structure and its environment;
  - we know a *limit function* $g(\theta)$ such that:
    - stable states correspond to $g(\theta) > 0$, while
    - failures correspond to $g(\theta) < 0$.
- Once we know this information, we can find the desired failure probability P as the integral $P = \int_{g(\theta)<0} f(\theta)\, d\theta$.
- P can be computed by a (somewhat time-consuming) Monte-Carlo algorithm (MCA).

2. Case of Interval Uncertainty: A Description

- We often do not know the exact probabilities:
  - usually, we know that the distribution belongs to a certain family (e.g., that it is normal),
  - but we only know the bounds $\underline{p}_i \le p_i \le \overline{p}_i$ on the corresponding parameters p_i.
- Similarly, we know the general parametric expression for the limit function (e.g., that $g(\theta)$ is linear or quadratic).
- However, we only know the bounds on the actual (unknown) values of the corresponding parameters p_j.
- For each combination of parameters p_i, we can compute the failure probability $P_f(p_1, \ldots, p_m)$.
- We want to compute the range

$$\left[\underline{P}, \overline{P}\right] \stackrel{\text{def}}{=} \left\{ P_f(p_1, \ldots, p_n) : p_i \in \left[\underline{p}_i, \overline{p}_i\right] \right\}.$$

3. Linearization is Usually Possible

- Each interval can be represented as $[\widetilde{p}_i - \Delta_i, \widetilde{p}_i - \Delta_i]$, where $\widetilde{p}_i$ is a midpoint and Δ_i is half-width.
- Possible values p_i are $p_i = \widetilde{p}_i + \Delta p_i$, with $|\Delta p_i| \le \Delta_i$, so $P_f(p_1, \ldots, p_m) = P_f(\widetilde{p}_1 + \Delta p_1, \ldots, \widetilde{p}_m + \Delta p_m)$.
- The values Δ_i are usually reasonable small, hence the values Δp_i are also small; thus:
  - we can expand P_f into Taylor series
  - and keep only linear terms in this expansion:

$$P_f(\widetilde{p}_1 + \Delta p_1, \ldots) = \widetilde{P} + \sum_{i=1}^{m} c_i \cdot \Delta p_i, \widetilde{P} \stackrel{\text{def}}{=} P_f(\widetilde{p}_1, \ldots), c_i \stackrel{\text{def}}{=} \frac{\partial P}{\partial p_i}.$$

- Here, $\max(c_i \cdot \Delta p_i) = |c_i| \cdot \Delta_i$, so the range of P_f is

$$\left[\widetilde{P} - \Delta, \widetilde{P} + \Delta\right], \text{ where } \Delta = \sum_{i=1}^{m} |c_i| \cdot \Delta_i.$$

4. Towards an Algorithm

- To compute $\Delta = \sum_{i=1}^{m} |c_i| \cdot \Delta_i$, we need to find c_i.
- If we replace one of $\widetilde{p}_i$ with $\widetilde{p}_i + \Delta_i$, then, due to linearization, we get

$$P_i \stackrel{\text{def}}{=} P_f(\widetilde{p}_1, \ldots, \widetilde{p}_{i-1}, \widetilde{p}_i + \Delta_i, \widetilde{p}_{i+1}, \ldots, \widetilde{p}_m) = \widetilde{P} + c_i \cdot \Delta_i.$$

- Thus, $|c_i| \cdot \Delta_i = \left| P_i - \widetilde{P} \right|$ and hence $\Delta = \sum_{i=1}^{m} \left| P_i - \widetilde{P} \right|$.
- *Algorithm:* compute $\widetilde{P} = P_f(\widetilde{p}_1, \ldots, \widetilde{p}_m)$, m values $P_i = P_f(\widetilde{p}_1, \ldots, \widetilde{p}_{i-1}, \widetilde{p}_i + \Delta_i, \widetilde{p}_{i+1}, \ldots, \widetilde{p}_m)$, then

$$\Delta = \sum_{i=1}^{m} \left| P_i - \widetilde{P} \right| \text{ and } \left[\widetilde{P} - \Delta, \widetilde{P} + \Delta \right].$$

- This algorithm requires $m+1$ calls to MCA: to compute $\widetilde{P}$ and m values P_i.

5. Towards a Faster Algorithm

- When the number of parameters m is large, $m+1$ calls may be too long.
- We can use the property of Cauchy distribution

$$\rho_\delta(x) = \frac{\delta}{\pi} \cdot \frac{1}{1 + \dfrac{x^2}{\delta^2}}:$$

  - if η_i are independently Cauchy-distributed with parameters Δ_i,
  - then $\eta \stackrel{\text{def}}{=} \sum_{i=1}^{m} c_i \cdot \eta_i$ is Cauchy-distributed with parameter $\Delta = \sum_{i=1}^{m} |c_i| \cdot \Delta_i$.

- Once we get simulated Cauchy-distributed values η, we can estimate Δ by the Maximum Likelihood method.
- We also need to scale η_i to the interval $[-\Delta_i, \Delta_i]$ on which the linear approximation is applicable.

6. Faster Algorithm

- First, we compute $\widetilde{P} = P_f(\widetilde{p}_1, \ldots, \widetilde{p}_m)$.
- For some N (e.g., 200), for $k = 1, \ldots, N$, repeat:
  - use the random number generator to compute $r_i^{(k)}$, $i = 1, 2, \ldots, m$, uniformly distributed on $[0, 1]$;
  - compute Cauchy distributed values as
$$c_i^{(k)} = \tan(\pi \cdot (r_i^{(k)} - 0.5));$$
  - compute the largest value K of the values $\left|c_i^{(k)}\right|$;
  - compute simulated "actual values" $p_i^{(k)} = \widetilde{p}_i + \dfrac{\Delta_i \cdot c_i^{(k)}}{K}$;
  - apply MCA and compute
$$\Delta P^{(k)} = K \cdot \left(P_f\left(p_1^{(k)}, \ldots, p_i^{(k)}, \ldots, p_m^{(k)}\right) - \widetilde{P}\right).$$

7. Faster Algorithm (cont-d)

- We have computed

$$\Delta P^{(k)} = K \cdot \left(P_f\left(p_1^{(k)}, \ldots, p_i^{(k)}, \ldots, p_m^{(k)}\right) - \widetilde{P} \right).$$

- Then, we compute $\Delta \in \left[0, \max_k \left|\Delta P^{(k)}\right|\right]$ by applying the bisection method to the equation

$$\frac{1}{1 + \left(\dfrac{\Delta P^{(1)}}{\Delta}\right)^2} + \ldots + \frac{1}{1 + \left(\dfrac{\Delta P^{(N)}}{\Delta}\right)^2} = \frac{N}{2}.$$

- We stop when we get Δ with accuracy $\approx 20\%$ (accuracy 1% and 1.2% is approximately the same).
- The Cauchy-variate algorithm requires $N \approx 200$ calls to MCA.
- So, when $m \gg 200$, it is much faster than the above linearization-based algorithm.

8. Need to Take Model Inaccuracy Into Account

- Often, the given finite-parametric family of distributions is only an approximation.
- Similarly, the given family of limit functions is only an approximation.
- As a result of the model uncertainty:
  - the value $C = C_f(p_1, \ldots, p_m)$ produced by the approximate model
  - is, in general, different from the actual failure probability $P = P_f(p_1, \ldots, p_m)$.
- Often, the only available information about the model inaccuracy $C - P$ is the upper bound δ: $|C - P| \leq \delta$.
- Let us analyze how this inaccuracy affects our estimations.

9. What If We Use the Linearized Algorithm

- In the linearized case, the desired upper endpoint $\overline{C}$ has the form $\overline{C} = \widetilde{C} + \sum_{i=1}^{m} \left| \widetilde{C} - C_i \right|$.
- Each of the values $\widetilde{C}$ and $\widetilde{C}_i$ is accurate only with accuracy δ.
- As a result, $\left| \overline{C} - \overline{P} \right| \le (2m+1) \cdot \delta$.
- Similarly, $|\underline{C} - \underline{P}| \le (2m+1) \cdot \delta$.
- As an interval which is guaranteed to contain the actual failure probability P, we can thus take

$$\left[\underline{C} - (2m+1) \cdot \delta, \overline{C} + (2m+1) \cdot \delta \right].$$

- When the number m of parameters is large, the approximation error $(2m+1) \cdot \delta$ becomes significant.
- How can we decrease this approximation error?

10. Analysis of the Problem

- The actual maximum $\overline{P}$ is attained when $\Delta p_i = \varepsilon_i \cdot \Delta_i$, where $\varepsilon_i \stackrel{\text{def}}{=} \text{sign}(c_i)$.
- For these ε_i, we have $C_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots) \geq \overline{P} - \delta$.
- Thus, for $\overline{C} \stackrel{\text{def}}{=} \max_{\varepsilon} C_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots)$, we get $\overline{C} \geq \overline{P} - \delta$.
- On the other hand, for each ε, we have
$$C_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots) \leq P_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots) + \delta.$$
- Since $P_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots) \leq \overline{P}$, we conclude that
$$C_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots) \leq \overline{P} + \delta.$$
- Thus, for $\overline{C} = \max_{\varepsilon} C_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots)$, we have
$$\overline{C} \leq \overline{P} + \delta.$$
- So, the maximum $\overline{C}$ provides a δ-approximation to $\overline{P}$.

11. First New Algorithm

- For all 2^m possible combinations of values $\varepsilon_1 \in \{-1, 1\}$, $\ldots$, $\varepsilon_m \in \{-1, 1\}$, we estimate $C_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots)$.
- Then, we compute the largest of these estimates $\overline{C}$.
- We can then guarantee that $\left|\overline{C} - \overline{P}\,\right| \leq \delta$.
- Similarly, the smallest of the values $C_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots)$ is δ-close to $\underline{P}$: $|\underline{C} - \underline{P}\,| \leq \delta$.
- The above algorithm requires at least 2^m estimates, which for large m is unrealistically large.
- It is known that if we want to find $\widetilde{P}$ with accuracy δ, we cannot use fewer than exponentially many calls.
- How can we decrease the uncertainty in estimating without increasing the number of calls too much?

12. Towards the Second New Algorithm

- Let us compute a new difference

$$P_f(\ldots, \widetilde{p}_i + \Delta_i, \ldots) - P_f(\ldots, \widetilde{p}_i - \Delta_i, \ldots) = 2c_i \cdot \Delta_i.$$

- When $P_f(\ldots)$ is known with accuracy δ, we have $2c_i \cdot \Delta_i$ with accuracy 2δ, hence $|c_i| \cdot \Delta_i$ with accuracy $\delta < 2\delta$.
- So, for each $i = 0, \ldots, m$, we compute

$$E_i = C_f(\widetilde{p}_1 + \Delta_1, \ldots, \widetilde{p}_i + \Delta_i, \widetilde{p}_{i+1} - \Delta_{i+1}, \ldots, \widetilde{p}_m - \Delta_m).$$

- We take the average $\widetilde{E}$ of all E_i as an estimate for $\widetilde{P}$, and $\widetilde{\Delta} \stackrel{\text{def}}{=} \frac{1}{2} \cdot \sum_{i=1}^{m} |E_i - E_{i-1}|$ as an estimate for Δ.
- This algorithm requires $m + 1$ calls to C_f, but its accuracy is $(m + 1) \cdot \delta$ – twice better.

13. Towards the Third New Algorithm

- For $c_i \geq 0$, the maximum $\overline{P}$ is attained when $\Delta p_i = \Delta_i$.
- When can we conclude that $c_i > 0$?
- We know that $E_i - E_{i-1}$ is a 2δ-approximation to $2c_i \cdot \Delta_i$.
- So, if $E_i - E_{i-1} \geq 2\delta$, we can conclude that $c_i \geq 0$.
- Thus, after computing each E_i, we mark i as $i \in S^+$ if $E_i - E_{i+1} \geq 2\delta$, and $i \in S^-$ if $E_i - E_{i+1} \leq -2\delta$.
- Let $s = \#(S^- \cup S^+)$ and $S_0 = -(S^- \cup S^+)$.
- Then, we compute $p_i^+ = \widetilde{p}_i + \Delta_i$ when $i \in S^+$, $p_i^+ = \widetilde{p}_i - \Delta_i$ when $i \in S^-$, and $p_i^+ = \widetilde{p}_i$ else.
- We then compute $C^+ = C_f(p_1^+, \ldots, p_m^+)$ and

$$\overline{E} = C^+ + \frac{1}{2} \cdot \sum_{i \in S_0} |E_i - E_{i-1}| .$$

14. Third New Algorithm (cont-d)

- Similarly, we compute $p_i^- = \widetilde{p}_i - \Delta_i$ when $i \in S^+$, $p_i^- = \widetilde{p}_i + \Delta_i$ when $i \in S^-$, and $p_i^- = \widetilde{p}_i$ when $i \in S_0$.
- We then compute $C^- = C_f(p_1^-, \ldots, p_m^-)$ and

$$\underline{E} = C^- - \frac{1}{2} \cdot \sum_{i \in S_0} |E_i - E_{i-1}| \, .$$

- The estimates $\overline{E}$ and $\underline{E}$ approximate $\overline{P}$ and $\underline{P}$ with accuracy $(m + 1 - s) \cdot \delta$.
- This algorithm requires $m + 3$ calls to C_f.
- s is large if there are many parameters which significantly affect the failure probability.
- In this case, we get a drastic improvement in accuracy – at the expense of having only two more calls to C_f.

15. Similar Modification of the First Algorithm

- First, we compute the differences

$$E_i = C_f(\widetilde{p}_1 + \Delta_1, \ldots, \widetilde{p}_i + \Delta_i, \widetilde{p}_{i+1} - \Delta_{i+1}, \ldots, \widetilde{p}_m - \Delta_m).$$

- We mark i as $i \in S^+$ if $E_i - E_{i+1} \geq 2\delta$, as $i \in S^-$ if $E_i - E_{i+1} \leq -2\delta$, else as $i \in S_0$.
- To estimate $\overline{P}$, we estimate 2^{m-s} values

$$C_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots), \text{ where:}$$

  - $\varepsilon_i = 1$ for $i \in S^+$,
  - $\varepsilon_i = -1$ for $i \in S^-$, and
  - we take all possible combinations of the values $\varepsilon_i \in \{-1, 1\}$ for the remaining $m - s$ indices $i \in S_0$.

- The largest of these estimates is then returned as an estimate $\overline{C}$ for $\overline{P}$.

16. Modified First Algorithm (cont-d)

- To estimate $\underline{P}$, we estimate 2^{m-s} values

$$C_f(\widetilde{p}_1 + \varepsilon_1 \cdot \Delta_1, \ldots), \text{ where:}$$

  - $\varepsilon_i = -1$ for $i \in S^+$,
  - $\varepsilon_i = 1$ for $i \in S^-$, and
  - we take all possible combinations of the values $\varepsilon_i \in \{-1, 1\}$ for the remaining $m - s$ indices $i \in S_0$.

- The smallest of these estimates is then returned as an estimate $\underline{C}$ for $\underline{P}$.
- We get the same accuracy δ with $(m+1)+2\cdot 2^{m-s} \ll 2^m$ calls to C_f.

17. Towards a Fourth Algorithm

- To get the best accuracy δ, we need all 2^{m-s} combinations of $m-s$ values $\varepsilon_i = \pm 1$.
- If we only have time for $\le m+3$ combinations, then we can gain accuracy $\le (m-s+1)\cdot\delta$.
- What if we have more computation time but still not enough to try all 2^{m-s} combinations?
- It then makes sense to select $g > 1$ and divide $m-s$ parameters $i \in S_0$ into $\dfrac{m-s}{g}$ groups G_ℓ of size g.
- Within each G_l, we try all combinations, to estimate:
  - $s_1^+ \stackrel{\text{def}}{=} \widetilde{P} + \sum\limits_{i\in G_1} |c_i|\cdot\Delta_i$ (for the first group) and
  - $s_\ell \stackrel{\text{def}}{=} \sum\limits_{i\in G_\ell} |c_i|\cdot\Delta_i$ (for all other groups $\ell > 1$).
- Then, we add up the resulting estimates (same for $\underline{P}$).

18. Towards a Similar Modification of the Cauchy Variate Algorithm

- Due to model inaccuracy, we only know the values $P(p_1^{(k)}, \ldots)$ and $\widetilde{P}$ with accuracy δ.
- Thus, the computed value $\widetilde{\Delta}^{(k)} = K \cdot \left(C_f\left(p_1^{(k)}, \ldots\right) - \widetilde{C} \right)$ is $(2K \cdot \delta)$-close to the desired values $\Delta P^{(k)}$:

$$\Delta P^{(k)} \in \left[\widetilde{\Delta}^{(k)} - 2K \cdot \delta, \widetilde{\Delta}^{(k)} + 2K \cdot \delta \right].$$

- In the formula for Δ, Δ increases with each

$$s_k \stackrel{\text{def}}{=} \left(\Delta P^{(k)} \right)^2.$$

- Thus, to find the largest possible value of Δ, we need to take the largest possible value $\overline{s}_k$ of $\left(\Delta P^{(k)} \right)^2$:

$$\overline{s}_k = \left(\left| \widetilde{\Delta}^{(k)} \right| + 2K \cdot \delta \right)^2.$$

19. Resulting Algorithm

- First, we estimate $\widetilde{C} = C_f(\widetilde{p}_1, \ldots, \widetilde{p}_m)$.
- Then, we compute K and $p_i^{(k)}$ as in the original Cauchy deviate algorithm, and estimate $\widetilde{\Delta}^{(k)}$ and

$$\overline{s}_k = \left(\left|\widetilde{\Delta}^{(k)}\right| + 2K \cdot \delta\right)^2.$$

- After that, we compute $\Delta \in \left[0, \sqrt{\max_k \overline{s}_k}\right]$ by applying the bisection method to the equation

$$\frac{1}{1 + \dfrac{\overline{s}_1}{\Delta^2}} + \ldots + \frac{1}{1 + \dfrac{\overline{s}_N}{\Delta^2}} = \frac{N}{2}.$$

- Finally, we return the range $\left[\widetilde{C} - \Delta, \widetilde{C} + \Delta\right]$.

20. Parallelization and Acknowledgments

- One can easily see that the above algorithms are easily parallelizable.
- This work was supported in part by the National Science Foundation grants:
  - HRD-0734825,
  - HRD-1242122, and
  - DUE-0926721.