Advancing Probabilistic Risk Assessment in Critical Systems Using Python and Time Simulations: The SURE Methodology

Todd Paulos, Ph.D. [a], William Tirone, Andrew Ho[a], and Joshua Bendig

[a]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 USA,

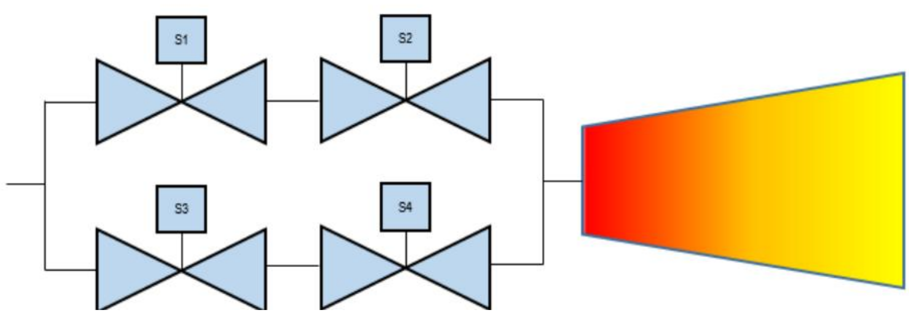
tpaulos@jpl.nasa.gov

Abstract: Probabilistic Risk Assessments (PRAs) are traditionally solved using probabilistic solutions, exponential random variables and rare event approximations. Unfortunately, these methods provide unrealistic predictions when modeling long duration missions based upon comparisons between predictions to future performance. This issue exists with any system that is run without maintenance until failure, or that has a high (non-rare event) failure probability. In these cases, Models of the World (MOWs) should model in time space. Previous work [1] demonstrated that using component life simulations instead of probabilistic solutions yielded vastly different results in both the overall risk number and risk drivers. In this paper, the simulation concept described previously is further developed into a multi-phase model that has been automated to ingest minimal cut sets and basic event failure rate information from a SAPHIRE [2] model, process the inputs into a simulation, run the simulation, and provide results. This methodology, called Simulated Uncertainty and Reliability Estimation (SURE), can also be applied to other types of modeling, such as quantitative fault trees, Reliability Block Diagrams (RBDs) or Markov Models.

1. INTRODUCTION

For review, the monopropellant thruster problem from [1] is shown in Figure 1.

Figure 1: Thruster Model



The thruster is fed propellant through four valves (S1, S2, S3, and S4). In this configuration, the thruster operates when propellant is flowing through either of the two flow paths or both. When the thruster is not in use, both flow paths must be closed to conserve propellant and prevent the spacecraft from gaining unwanted momentum. For simplicity, the only failure modes of the valves considered are Fail Open (FO) and Fail Closed (FC). Additionally, there is now a phase in which the failure occurred. In this case, there are only two phases, PH1 for Phase 1 and PH2 for Phase 2.

2. FAULT TREE

Like the 1-Phase problem, the fault tree is simple and can be easily developed by considering two phases for each failure mode. This requires an expanded Boolean naming convention for the fault tree in which the phase is now part of the naming scheme:

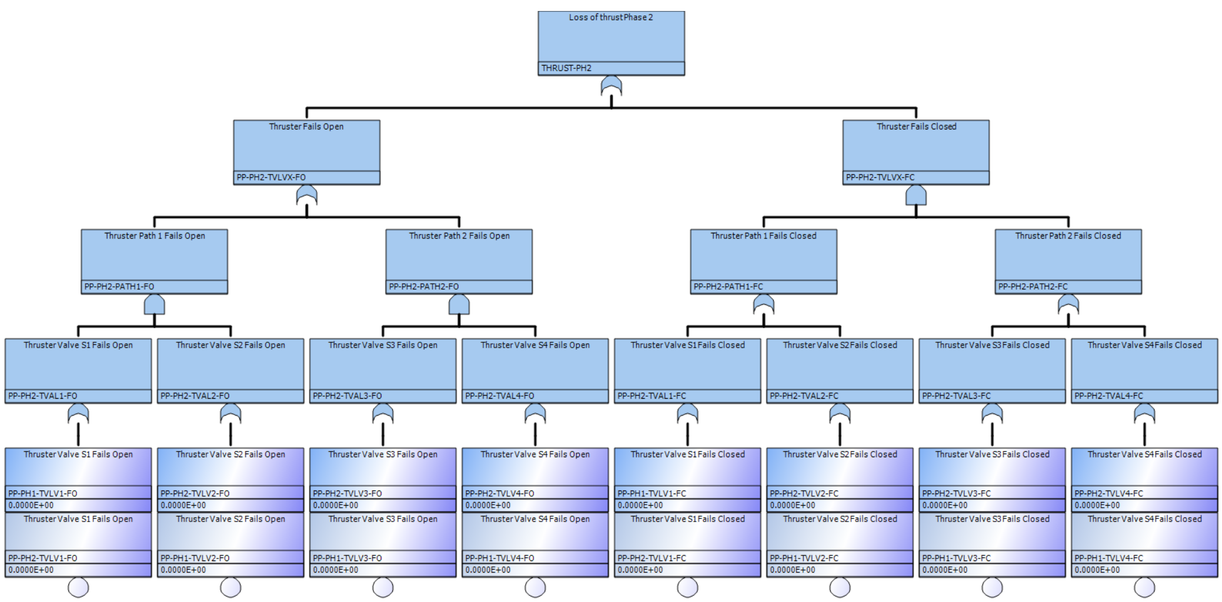
PP-PH?-TVLV?-F?

where,

PP = Propulsion Subsystem,
PH? = PH1 or PH2 for Phase 1 or Phase 2,
TVLV? = Thruster Valve number 1 through 4, and
F? = Fail Open or Fail Closed.

For example, PP-PH2-TVLV3-FC represents the Propulsion Subsystem, Phase 2, Thruster Valve 3 Fails Closed. The fault tree is shown in Figure 2, and the minimal cut sets for the fault tree are shown in Table 1.

Figure 2: Fault Tree for 2 Phase Thruster



The 1-phase model has 6 minimal cut sets (MCS), and the 2-phase model has 24. As the number of phases and model complexity increases, the need to automate simulation development becomes apparent. The minimal cut sets for the 2-phase model are in Table 1.

Table 1: MCS of 2-Phase Example Problem

Cut Set #	Basic Event Boolean Designator	Description
1	PP-PH1-TVLV2-FO PP-PH2-TVLV1-FO	Thruster Valve S2 Fails Open Thruster Valve S1 Fails Open
2	PP-PH2-TVLV1-FO PP-PH2-TVLV2-FO	Thruster Valve S1 Fails Open Thruster Valve S2 Fails Open
3	PP-PH1-TVLV1-FO PP-PH2-TVLV2-FO	Thruster Valve S1 Fails Open Thruster Valve S2 Fails Open
4	PP-PH1-TVLV1-FC PP-PH1-TVLV3-FC	Thruster Valve S1 Fails Closed Thruster Valve S3 Fails Closed
5	PP-PH1-TVLV2-FC PP-PH1-TVLV3-FC	Thruster Valve S2 Fails Closed Thruster Valve S3 Fails Closed
6	PP-PH1-TVLV1-FC PP-PH2-TVLV3-FC	Thruster Valve S1 Fails Closed Thruster Valve S3 Fails Closed
7	PP-PH2-TVLV2-FC PP-PH2-TVLV3-FC	Thruster Valve S2 Fails Closed Thruster Valve S3 Fails Closed
8	PP-PH2-TVLV1-FC PP-PH2-TVLV3-FC	Thruster Valve S1 Fails Closed Thruster Valve S3 Fails Closed
9	PP-PH1-TVLV2-FC PP-PH2-TVLV3-FC	Thruster Valve S2 Fails Closed Thruster Valve S3 Fails Closed
10	PP-PH1-TVLV3-FC PP-PH2-TVLV2-FC	Thruster Valve S3 Fails Closed Thruster Valve S2 Fails Closed
11	PP-PH1-TVLV4-FC PP-PH2-TVLV2-FC	Thruster Valve S4 Fails Closed Thruster Valve S2 Fails Closed
12	PP-PH1-TVLV3-FC PP-PH2-TVLV1-FC	Thruster Valve S3 Fails Closed Thruster Valve S1 Fails Closed
13	PP-PH1-TVLV4-FC PP-PH2-TVLV1-FC	Thruster Valve S4 Fails Closed Thruster Valve S1 Fails Closed
14	PP-PH1-TVLV1-FC PP-PH1-TVLV4-FC	Thruster Valve S1 Fails Closed Thruster Valve S4 Fails Closed
15	PP-PH1-TVLV2-FC PP-PH1-TVLV4-FC	Thruster Valve S2 Fails Closed Thruster Valve S4 Fails Closed
16	PP-PH1-TVLV1-FC PP-PH2-TVLV4-FC	Thruster Valve S1 Fails Closed Thruster Valve S4 Fails Closed
17	PP-PH2-TVLV2-FC PP-PH2-TVLV4-FC	Thruster Valve S2 Fails Closed Thruster Valve S4 Fails Closed
18	PP-PH2-TVLV1-FC PP-PH2-TVLV4-FC	Thruster Valve S1 Fails Closed Thruster Valve S4 Fails Closed
19	PP-PH1-TVLV2-FC PP-PH2-TVLV4-FC	Thruster Valve S2 Fails Closed Thruster Valve S4 Fails Closed
20	PP-PH2-TVLV3-FO PP-PH2-TVLV4-FO	Thruster Valve S3 Fails Open Thruster Valve S4 Fails Open
21	PP-PH1-TVLV3-FO PP-PH2-TVLV4-FO	Thruster Valve S3 Fails Open Thruster Valve S4 Fails Open
22	PP-PH1-TVLV4-FO PP-PH2-TVLV3-FO	Thruster Valve S4 Fails Open Thruster Valve S3 Fails Open
23	PP-PH1-TVLV1-FO PP-PH1-TVLV2-FO	Thruster Valve S1 Fails Open Thruster Valve S2 Fails Open
24	PP-PH1-TVLV3-FO PP-PH1-TVLV4-FO	Thruster Valve S3 Fails Open Thruster Valve S4 Fails Open

3. ALGORITHM

The goals of this effort are to enhance the simulation by adding multi-phase capability and to automate inputs as much as possible. The simple pseudocode for the simulation is:

1. Import MCS and basic event uncertainty information from SAPHIRE 8, at either the end state or fault tree level.
2. Process the MCS and basic event information into the simulation tool.
3. Identify components with multiple failure modes and phases and mark them for additional handling.
4. Convert basic event rate and uncertainty information for each basic event into a failure time model.
5. Run simulation
    a. Draw failure time values for each basic event.
    b. With every draw for a component failure time:
        i. Compare component failure mode times; first to fail is the only failure mode to occur, and the other failure modes for the component are labeled infinite, as they become impossible to occur.
        ii. Record failure event in the specific phase, marking other phases as impossible.
        iii. Mark specific basic event as failed (component, failure mode and phase) based on phase time definition.
    c. For every MCS, determine failure time (when last event in the MCS occurs).
    d. Determine if MCS resulted in failure at the system level, i.e., did it occur within the mission time?
    e. Compare MCS failure times to determine which failure led to system failure.
6. Compute summary statistics of mission times to failure and drivers.

4. INPUTS

As summarized in [1], the rationale for using gamma distributions is based on the literature [3]. The prior distribution for the failure rate λ is a gamma(α_{prior}, β_{prior}), and the likelihood is considered a Poisson with x events in time t. The posterior is thus a gamma distribution for λ,

$$\text{gamma}(\alpha_{posterior}, \beta_{posterrior}) \tag{1}$$

$$\alpha_{posterior} = \alpha_{prior} + x \tag{2}$$

$$\beta_{posterior} = \beta_{prior} + t \tag{3}$$

Typically, a Jeffreys' noninformative prior is used: gamma(½, 0) [3], which yields Equations (4) and (5)

$$\alpha_{posterior} = x + \tfrac{1}{2} \tag{4}$$

$$\beta_{posterior} = t + 0 \tag{5}$$

The simulations use three different failure rates for each basic event, regardless of phase or failure mode: gamma(2,100000), gamma(2,200000), and gamma(2.5,100000). The mission times are 3-, 5-, and 10-years. All simulations are run to 100,000 trials. In this context, the word "simulation" is taken to mean a *set* of trials (e.g., 100,000) used to produce mission failure time and summary statistics. Larger numbers of simulations can be run, but 100k can be run in Excel and is deemed significantly large enough to generate results for comparison purposes. The number of trials used depends on the estimated probabilities of failure.

5. RESULTS

5.1. Probabilistic Results

The results of all simulations using the various tools and basic event rates are compiled in Table 2.

Table 2: Simulation Comparisons to the PRA Results For 100K Trials

	Basic Events	**Gamma(2,100000)**		**Gamma(2,200000)**		**Gamma(2.5,100000)**	
Mission Time	Method	1-Phase	2-Phases	1-Phase	2-Phases	1-Phase	2-Phases
3 years	Excel	6.1E-02	6.2E-02	8.0E-05	5.0E-05	1.6E-01	1.6E-01
	MATLAB sim	5.7E-02		7.0E-05		1.4E-01	
	Python sim	5.5E-02	5.7E-02	9.1E-05	9.9E-05	4.2E-01	1.4E-01
	SAPHIRE	6.7E-01	7.3E-01	2.8E-01	3.1E-01	8.0E-01	8.6E-01
5 years	Excel	4.6E-01	4.6E-01	1.9E-02	1.9E-02	7.1E-01	7.1E-01
	MATLAB sim	3.9E-01		1.8E-02		6.0E-01	
	Python sim	3.8E-01	3.8E-01	1.8E-02	1.8E-02	3.8E-01	3.8E-01
	SAPHIRE	9.2E-01	9.6E-01	5.5E-01	6.1E-01	9.7E-01	1.0E+00
10 years	Excel	9.4E-01	9.5E-01	4.6E-01	4.6E-01	9.2E-01	9.9E-01
	MATLAB sim	8.5E-01		3.8E-01		9.5E-01	
	Python sim	8.5E-01	8.5E-01	3.8E-01	3.8E-01	9.5E-01	8.9E-01
	SAPHIRE	1.0E+00	1.0E+00	9.2E-01	9.6E-01	1.0E+00	1.0E+00

The 1-phase models from Excel, MATLAB, and Python are in close agreement with each other and with the previous work [1]. It is important to note that the values shown in the table are from the *first* simulation. Table 2 does not show that repeated simulations have slightly different results. All three simulations are shown to differ significantly from the SAPHIRE solution, primarily because they assume a model of the world in which many things can go wrong before the system fails, and because they do not use exponential random variables. As shown in the previous work and reproduced here, in the gamma(2,2000000) and 3-year cases, the simulations have a failure probability of around 1E-04, while the SAPHIRE solution has a failure probability of 0.28.

For the 2-phase model, the Excel simulation is extended for sanity checks, but the MATLAB effort is not. The 2-phase simulation results shown in Table 2 are also comparable. The SAPHIRE results are even further apart than the simulations; this is expected given the increase in the number of MCS in the solution. The calculated probability of failure for the 2-Phase SAPHIRE model at 10-years using gamma (2.5,100000) is 0.9999992; the results from the simulation model show a 0.92 probability of failure.

An interesting aspect of this approach is treating mission failure times rather than failure probability. Questions about expected mission life and life quantiles, which help inform asset replacement decisions, are often asked. For example, the simulation results for the 3-year and gamma(2,100000) cases are shown in Figure 3 and Table 3.

Figure 3: Distribution of Mission Failure Times

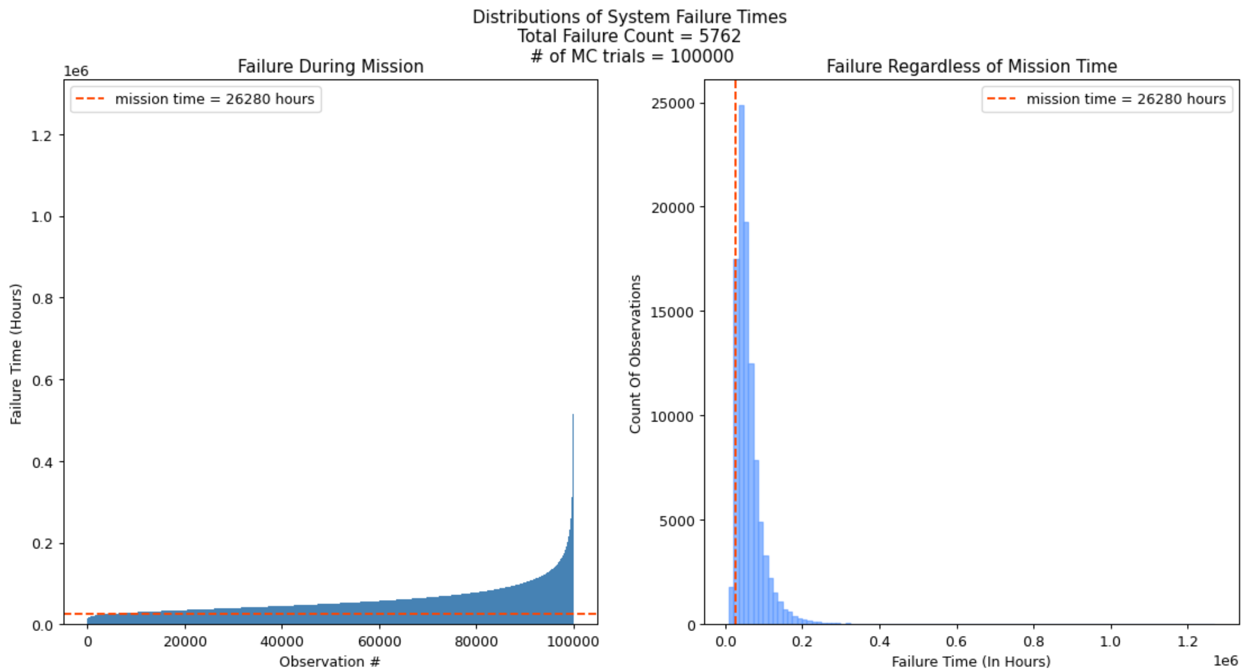


Table 3: Mission Failure Time Statistics for 3-Year Mission and Gamma(2, 100000)

Quantile	Failure In Mission Time (Hrs)	Failure Regardless of Mission Time (Hrs)
count	5762	100000
mean	22793	60817
std	2761	40480
min	9203	9203
5%	17392	25633
10%	18779	29318
25%	21171	37302
50%	23415	50351
75%	25012	71111
90%	25793	102203
95%	26065	129442
max	26280	1270554

5.2. Risk Drivers

As many have said, the power of conducting PRA is not in the numbers generated; it lies in the systems analysis and risk driver prioritization. However, another significant difference between an exponential model and a simulation model is that, in the former, the results are always the same, whereas in the latter, both the failure probability and risk rankings change from simulation to simulation.

A concerted effort is typically needed to interpret MCS results of any complex PRA, regardless of how the model is solved. The results are calculated using basic events, and they need to be further analyzed, processed, and understood at a higher level for managers and decision-makers. Solving using simulation adds an extra dimension of complexity. Table 4 lists the results from one simulation. It should be noted that unless the same seed is used from simulation to simulation, the results will slightly vary.

A list of prioritized MCS of a complex system is difficult to sort through and characterize at an actionable level for anyone. Imagine 10,000, 100,000, or 1,000,000 or more cut sets. It takes an effort to analyze this volume and put them into a handful of scenarios that drive the risk. No single risk driver in Table 4 tells the risk story; looking only at the top two risks gives a false impression of the dominant risks. Scenarios that result from valves failing open, despite being the first two on the list, account for only 36% of the total, while scenarios that result from valves failing to close account for 64% of the risk.

The MCS are in three groupings, although this may seem counterintuitive given that the basic event inputs have the same failure rates and the exposure times are all the same. As gamma random variables, later events are more likely than earlier ones. Drivers 1 through 6 have both failures occur in Phase 2, with *approximately* the same likelihood. Again, the results will slightly change from simulation to simulation, but these six are always the top six if the simulation is repeated.

Table 4: Mission Risk Drivers

#	Minimal Cut Set	Fail Count in Mission	Fail Count Regardless of Mission	Proportion in Mission	Proportion Regardless of Mission Time
1	PP-PH2-TVLV3-FO/PP-PH2-TVLV4-FO	949	21469	0.164699757	0.21469
2	PP-PH2-TVLV1-FO/PP-PH2-TVLV2-FO	943	21425	0.163658452	0.21425
3	PP-PH2-TVLV2-FC/PP-PH2-TVLV3-FC	839	16343	0.145609163	0.16343
4	PP-PH2-TVLV1-FC/PP-PH2-TVLV3-FC	854	13799	0.148212426	0.13799
5	PP-PH2-TVLV2-FC/PP-PH2-TVLV4-FC	820	13645	0.142311697	0.13645
6	PP-PH2-TVLV1-FC/PP-PH2-TVLV4-FC	805	11220	0.139708435	0.1122
7	PP-PH1-TVLV2-FO/PP-PH2-TVLV1-FO	51	233	0.008851093	0.00233
8	PP-PH1-TVLV1-FC/PP-PH2-TVLV4-FC	62	206	0.010760153	0.00206
9	PP-PH1-TVLV4-FO/PP-PH2-TVLV3-FO	45	206	0.007809788	0.00206
10	PP-PH1-TVLV1-FO/PP-PH2-TVLV2-FO	48	205	0.008330441	0.00205
11	PP-PH1-TVLV1-FC/PP-PH2-TVLV3-FC	43	198	0.007462687	0.00198
12	PP-PH1-TVLV3-FO/PP-PH2-TVLV4-FO	43	195	0.007462687	0.00195
13	PP-PH1-TVLV3-FC/PP-PH2-TVLV2-FC	51	175	0.008851093	0.00175
14	PP-PH1-TVLV3-FC/PP-PH2-TVLV1-FC	45	156	0.007809788	0.00156
15	PP-PH1-TVLV4-FC/PP-PH2-TVLV2-FC	47	136	0.00815689	0.00136
16	PP-PH1-TVLV2-FC/PP-PH2-TVLV4-FC	38	128	0.006594932	0.00128
17	PP-PH1-TVLV4-FC/PP-PH2-TVLV1-FC	30	125	0.005206526	0.00125
18	PP-PH1-TVLV2-FC/PP-PH2-TVLV3-FC	32	119	0.005553627	0.00119
19	PP-PH1-TVLV1-FC/PP-PH1-TVLV4-FC	5	5	0.000867754	0.00005
20	PP-PH1-TVLV1-FC/PP-PH1-TVLV3-FC	4	4	0.000694203	0.00004
21	PP-PH1-TVLV2-FC/PP-PH1-TVLV3-FC	3	3	0.000520653	0.00003
22	PP-PH1-TVLV2-FC/PP-PH1-TVLV4-FC	2	2	0.000347102	0.00002
23	PP-PH1-TVLV3-FO/PP-PH1-TVLV4-FO	2	2	0.000347102	0.00002
24	PP-PH1-TVLV1-FO/PP-PH1-TVLV2-FO	1	1	0.000173551	0.00001

The second grouping of 12 MCS (numbers 7 to 18) has one failure in Phase 1 and the second in Phase 2. The last grouping is the six MCS where both failures occur in Phase 1. Again, this is expected given the failure rates, use of gamma distributions, and the mission time. It is important to note that in some simulations, not all MCS with both failures occurring in Phase 1 appear in the results. The probability is low, so low that the 100,000 trials may not always result in that scenario failing.

6. DISCUSSION

Gamma random variables are not memoryless, and this has a significant impact on failure probability. Tables 5, 6, and 7 show various λt values comparing exponential and gamma distributions.

Table 5: λt = 1E-3

Failure Rate (λ) Per Time	Time t	Number of Failures in Time t	Minimum Time to Failure (10^6 Trials)
1.00E-04	10	0	349.2782
1.00E-05	100	0	4,802.795
1.00E-06	1000	0	40,266.53
1.00E-07	10000	0	472,746.8

Columns 1 and 2 of Table 5 show various combinations of λt that equal 1E-3, such as λ = 1E-4 and t = 10, or λ = 1E-5 and t = 100, etc. The third column shows the number of trials out of 1 *million* that failed, using mission time t and λ from a gamma distribution. In every case, the failure count is zero. Column 4 shows the minimum time to failure from the million trials, which is typically two orders of magnitude greater than the mission time (Column 2). This is a major contrast with treating failure rates as exponential random variables and calculating the probability of failure at 1E-3 for each MCS. Given that some PRA solutions can have 1000s to 100,000s of MCS, this has a significant impact on the system failure probability.

Tables 6 and 7 are similar, showing values of λt = 1E-4 and 1E-5 (α = 0.5). In all cases, although the rare-event approximation of an exponential distribution yields a failure probability, the simulation results show that no failures occur in 1 million trials. Combined with the large number of minimal cut sets that often exist, exponential solutions yield high-probability values that are unrealistic. Any reliability analysis that uses exponential random variables has similar issues.

Table 6: λt = 1E-4

Failure Rate (λ) Per Time	Time t	Number of Failures in Time t	Minimum Time to Failure (10^6 Trials)
1.00E-05	10	0	4,201.016
1.00E-06	100	0	43,236.67
1.00E-07	1000	0	381,401.3
1.00E-08	10000	0	4,306,800

Table 7: λt = 1E-5

Failure Rate (λ) Per Time	Time t	Number of Failures in Time t	Minimum Time to Failure (10^6 Trials)
1.00E-06	10	0	37,135.85
1.00E-07	100	0	365,918.1
1.00E-08	1000	0	3,938,313

7. FUTURE WORK

SURE by itself is a methodology that can be applied through a variety of ways. For this work, it has been implemented into a Python script residing in a JupyterLab 4.5.1 environment [4]. This script was developed specifically to handle the PRA efforts at JPL, meaning the naming convention, failure modes, special cases, uncertainty distributions, etc., are all known and consistent. Some individual PRA model characteristics may be unique, and these are hard coded into the tool. Efforts are being made to make it a stand-alone script to be run outside the JupyterLab environment, to be more flexible in phasing, to allow easier input of phases, times, and component failure modes, and to add code enhancements for reporting and error handling.

8. CONCLUSIONS

The SURE methodology is presented as an alternative solution to risk and reliability models when mission times are long and failure probabilities are high, i.e., the constant failure rate and rare event approximations typically used are no longer valid. The Python-based simulation successfully replicates previous results from Excel and MATLAB and adds additional functionality and capability. The simulation can process outputs from SAPHIRE directly, including MCS from fault trees and end states, as well as basic event-uncertainty data. The simulation can handle a sizable model and multiphase scenarios, and it can run millions of trials per simulation (although the upper limit has not been tested).

It should be noted that all analysts should be cognizant that what may work for some may not work for others. It is up to the analysts to determine what makes sense in their model applications and what provides reasonable results. The application presented here is one possible use of the methodology.

Acknowledgement

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] T. Paulos, C. Smith and A. Ho. "*Reliability Modeling of Complex Components Using Simulation*," Proceedings of the 16th International Conference on Probabilistic Safety Assessment and Management (PSAM 16), (2022). Available at https://psam16.org/.

[2] https://saphire.inl.gov/#/ (last accessed May 5, 2026).

[3] C.L. Atwood et al. "Handbook of Parameter Estimation for Probabilistic Risk Assessment," NUREG/CR-6823, (2003). Available at https://www.nrc.gov/docs/ML0329/ML032900131.pdf.

[4] Project Jupyter Contributors. Jupyter Version 4.5.1. Available at https://jupyter.org/ or https://github.com/jupyterlab/jupyterlab (last accessed May 5, 2026).