# Simulation of Liquid Rocket Engine Failure Propagation Using Self-Evolving Scenarios

Donovan Mathias, NASA Ames Research Center

Samira Motiwala, University Space Research Association, NASA Ames Research Center

Key Words: spaceflight, rocket engine, dynamic risk assessment, scenario generation

*SUMMARY & CONCLUSIONS*

Traditional probabilistic risk assessment approaches often require failure scenarios to be explicitly defined through event sequences that are then quantified as part of the integrated analysis. This approach becomes difficult when failure propagation paths change as a function of the system operation. Additionally, if the propagation paths represent interactions among even a modest number of components, the scenario count becomes combinatorially intractable. This paper presents an alternate approach for quantifying the probability of failure propagation in such a case. Rather than explicitly defining scenario sequences, simple physical models are created for each of the components. In this way, only the physical states and rules of component interactions must be defined, rather than event sequences for each individual scenario. Initiating failures are introducted into the system, either randomly or as defined by relative likelihood, and the failures cascade through the system via the interaction rules. This process is repeated using Monte Carlo methods and, as a result, the most probable scenarios "self-evolve" in terms of both sequence path and frequency.

This approach was applied to failures occurring in the engine compartment of a space launch vehicle with four liquid rocket engines and four high-pressure helium tanks. Each engine was modeled with key components, such as turbomachinery, combustion chamber, propellant lines, and additional support systems. Three test cases were conducted with different high-energy engine failures. End results of interest included an additional engine-out failure and tank burst, which represent the loss-of-mission (LOM) and loss-of-crew (LOC) failure environments, respectively. Observations show that almost every scenario outcome is unique and that many scenarios involve complex chain reactions that are difficult to predict. This validates the usefulness of the modeling approach in assessing the overall risks to the crew during a launch vehicle abort.

*1 INTRODUCTION*

The National Aeronautics and Space Administration (NASA) strives to achieve and maintain high safety standards for its space launch and exploration systems. Probabilistic risk assessment (PRA) is incorporated into the development and operation of these complex systems to identify driving risk factors and strategies for cost-effectively improving their safety and performance. Crew safety remains NASA's primary goal for human spaceflight missions, and effective abort capabilities are crucial for reducing the overall risk to the crew.

The Engineering Risk Assessment (ERA) team at the NASA Ames Research Center provides simulation-based risk assessment approaches for analyzing crew launch vehicle abort scenarios during the mission ascent phase. The ERA approaches use physics-based models to characterize failure environments in terms of risks posed by blast overpressure, resulting debris field, and fireball thermal radiation [1]. The subsequent propagation of these failure environments is analyzed to evaluate the abort system's ability to escape safely. Failure propagation analysis involves assessing various failure "initiators" and quantifying their relationship with crew-threatening failure environments. The ultimate goal is to identify driving risk factors and guide designers towards effective risk-reducing strategies [2].

Current PRA approaches can pose challenges, however, since they require the explicit definition of failure scenarios through event sequences that are then quantified and integrated into the overall analysis. A list of failure initiators is generated to identify all critical functions required for a successful mission ascent, and each initiator can propagate its failure in multiple ways. Therefore, all possible propagation paths from every possible initiator need to be considered for a comprehensive analysis of all involved risks. This approach becomes difficult when these propagation paths change as a function of the system operation or when the scenario count becomes unmanageable.

This paper presents an alternate approach for quantifying failure propagation probabilities to generate scenarios more organically and intuitively. Rather than explicitly defining scenario sequences, simple physical models of the components are created to generate self-evolving scenarios from a given failure initiator. Each component is described by its ability to generate physical threats to its peers as well as its susceptibility to state disturbances caused by other components. Each component monitors the integrated system state for changes that impact its own operation. In this way, only the rules of component interactions must be defined,

rather than each individual scenario's event sequence. These interaction rules can vary as a function of the system state itself to more accurately represent the evolution of failure. Initiating failures are injected into the system, either randomly or as defined by relative likelihood, and the failures cascade through the system via the interaction rules. This process is repeated using Monte Carlo methods and, as a result, the most probable scenarios "self-evolve" in terms of both sequence path and frequency.

This paper outlines how this approach can be used to simulate failure propagation from blast wave and debris field environments. The physics of the model is explained as well as the specific pre-defined states and interaction rules required to run the simulation. The paper concludes with an example of the approach applied to failures occurring in the engine compartment of a space launch vehicle. In this example, four liquid rocket engines are modeled and tested with various high-energy component initiators that are capable of creating explosive failure environments. Observations from this test case are used to assess and cross-validate existing data to better understand the interrelationships between components and their surrounding environment, with the ultimate goal of improving the safety of crew launch systems.

## *2 MODELING APPROACH*

The Monte Carlo framework begins with explicit definitions of each component's physical states and interaction rules. Each component is physically modeled using CAD software, and the geometry is included in the simulation using a triangulated surface mesh. The initial conditions assigned to a component—such as the number, speeds, and directions of fragments released—are characterized by uncertainty distributions so that every failure scenario is effectively unique. Physical properties assigned to each component vessel, such as the amount of its pressurization, help characterize its system state. Each component also has vulnerability criteria that describe its own response to blast pressures and debris impacts.

This section outlines the debris and blast wave propagation models. It describes the initial states and conditions defined for each model as well as the physical mechanisms involved during the propagation.

### *2.1 Debris Propagation*

The debris field environment is strongly dependent on the initial fragment distribution, which is defined in terms of the total number of debris pieces as well as each piece's mass and imparted relative velocity [1]. Masses are assigned by specifying a component's total mass and a distribution type to allocate a mass value for each debris piece. Velocities are assigned by specifying values and distribution types for the speed and distribution angles of each debris piece. A kinetic energy vulnerability criterion is included to describe a component's structural threshold for withstanding an impact.

To initiate the propagation sequence, the user selects the component that serves as the source of the propagation (i.e., the component that fragments and generates a debris cloud). The initiator's debris catalog is randomly generated based on the aforementioned inputs. These debris pieces are treated as point masses that emanate from the source's geometric center with masses and velocities defined from the catalog.

The model calculates a debris strike by determining if the velocity ray of each debris piece intersects each surface mesh triangle of every surrounding component. However, before determining whether the velocity ray intersects the individual triangles of a component's surface mesh, a triangulated "bounding box" is created to first compute whether the ray passes through the vicinity of any given component. This box uses the component's minimum and maximum coordinate values to determine the minimum enclosing box within which all of the component's vertices lie. Each surface of the box is divided into two triangles for a total of 12 triangles (the minimum amount of triangles required for the box's triangulated mesh). This enables the model to save a significant amount of computation time since a component can have hundreds to thousands of triangles comprising its mesh. Figure 1 shows a visual representation of a meshed object surrounded by a meshed bounding box.

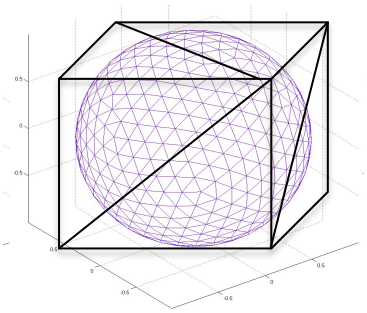

*Figure 1 – Triangulated Sphere with Meshed Bounding Box*

The inclusion of a point on the ray with a triangular plane can be determined by computing the parametric coordinates of the intersection point in the plane [3-4]. Consider a ray from $v_1$ to $v_2$ (with $\boldsymbol{v} = v_2 - v_1$) and a triangle with vertices $p_1$, $p_2$, and $p_3$. The normal of the plane containing the triangle is computed using the cross product:

$$\boldsymbol{n} = (p_2 - p_1) \times (p_3 - p_1) \tag{1}$$

If the ray intersects the plane, then the point of intersection can be evaluated to determine if that point also intersects the triangle. Otherwise, if the ray does not intersect the plane, then no intersections with the triangle are possible. The parametric representation of the ray is given by:

$$x = o + s\,\boldsymbol{v} \tag{2}$$

where $o$ is the origin of the ray, $\boldsymbol{v}$ is the ray direction vector, and $s$ is the "slope-like" parameter corresponding to the intersection point. This parameter can be determined by first considering the implicit representation of the plane:

$$\boldsymbol{n} \cdot x + (-p_1 \cdot \boldsymbol{n}) = 0 \tag{3}$$

Substituting (2) into (3), rearranging terms, and solving for *s gives*:

$$s = -\boldsymbol{n} \cdot (o - p_1) / (\boldsymbol{n} \cdot \boldsymbol{v}) \quad (4)$$

If the denominator of $s$ is less than a very small number (close to zero), the ray is considered parallel to the triangular plane and no intersection occurs. If $s = 0$ (i.e., the dot product in the numerator is zero), the ray lies in the triangular plane and again, no intersection occurs. There is also no intersection if $s < 0$ (ray points away from the triangular plane) or if $s > 1$. Otherwise, Equation (2) computes the intersection point of the ray with the plane.

Once the intersection with the plane is confirmed, the inclusion of that point in the triangle itself must be validated. This can be done by first computing the area of the triangle in 3D space and then computing the triangular areas formed by the intersection point with each of the triangle's edges. If the difference between the triangular area and the sum of the other areas is zero (or less than a very small tolerance), then the areas are equivalent and the ray intersects the triangle. Otherwise, the ray only intersects the plane and not the triangle.

If the model results indicate an intersection of the velocity ray with any of the bounding box's triangles, every triangle of the component's meshed surface is searched to determine whether the velocity ray intersects the component. If the model confirms an intersection with any of the component's surface triangles, it calculates the strike time using the debris piece's travelled distance and speed. Since the ray would intersect the component twice (to get both "inside" and back "outside"), only the first strike, corresponding to the earlier strike time, is stored and tabulated.

The debris piece imparts a kinetic energy to the component it strikes, so a kinetic energy threshold is specified as an input for each component to establish the maximum energy impact it can withstand before failing. If the imparted energy exceeds this threshold, and if the component is breakable (i.e., it can generate debris), then the component serves as the new "source" of debris for propagation. Its debris catalog is generated and the model computes whether the new debris pieces intersect any other components. Those that have already generated debris for propagation (i.e., "exploded") become nonexistent and are reduced to a single point so that they cannot be "struck" by subsequent propagations. The process repeats itself until no remaining components are struck or all further propagation possibilities are exhausted.

### *2.2 Blast Wave Propagation*

Blast overpressure is the pressure caused by a shock wave due to a high-energy explosion or tank burst. The intensity of the overpressure environment is typically measured in terms of the peak overpressure value and the time integration of the overpressure distribution (i.e., impulse) [2].

The blast propagation model is based on the Kingery-Bulmash (K-B) equations [5] used to model a spherical, surface explosion. This model uses K-B curve fitting for high-explosive blast propagation characteristics based on 1 kg of TNT at sea level. It generates blast environment parameters from component vessel parameters, ambient environment parameters, and distance from the vessel to a target component. The outputs of the model include incident and reflected peak overpressure as well as incident and reflected impulse.

The component vessel state parameters include its volume, pressure, and specific heat ratio of contained gas. Ambient environment parameters, such as pressure, specific heat ratio, and speed of sound, also govern the shock propagation. These are determined from a given altitude, which the user specifies as part of the failure initiation. After subsequent propagations, however, the altitude changes based on the elapsed time. The new altitude can be determined from an altitude-time history curve, which is typically obtained from a trajectory analysis of the given launch vehicle platform.

Similar to the kinetic energy threshold, an overpressure threshold is pre-defined for each component to establish the maximum peak reflected overpressure it can withstand before failing and becoming a new source of both debris and blast overpressure. The reflected peak overpressure is used because it is the maximum amount of overpressure experienced by the impacted component. Any failed element, either by debris strike or blast overpressure, becomes nonexistent for subsequent propagations.

### *2.3 Model Results*

Using the modeling approach described above, the user needs only to specify the failure initiator to begin the propagation simulation, which continues until the system has catastrophically failed or the failure propagation ends. In this way, the chain-reaction scenarios manifest themselves ("self-evolve") based on the initiator alone. It is clear to see that increasing the number of modeled components increases the number of possible permutations (and other sequence paths) resulting from a single initiator. Eventually, with a large enough number of components, the scenario count becomes combinatorially intractable. This approach allows for a large increase in modeled components for only a modest increase in computation time because the most probable scenarios naturally evolve.

The end result of the Monte Carlo simulation for a given initiator is the frequency of component failures of interest and associated sequence of events. The model records the probabilities of each component's observed failure occurrences as well as the probability breakdown of each victim component's failure from all contributing culprit components. The model also tracks every failure sequence to determine the probability of any particular sequence occurring and identify the most likely sequence of events. This enables the user to estimate the scenario likelihoods, which can be used to make informative risk assessment decisions.

## 3 LIQUID ROCKET ENGINE EXAMPLE

The failure propagation approach is applied to an assembly of liquid-fuelled cryogenic rocket engines in the engine bay of a generic space launch vehicle. The components are created using computer-aided design (CAD) software and are roughly based on a highly simplified version of Aerojet Rocketdyne's J2X, a liquid oxygen/liquid hydrogen engine originally planned for use on NASA's Ares I launch vehicle [6]. Critical high-energy components of the engine include the main combustion chamber (MCC), hydrogen fuel turbopump (FTP), oxygen turbopump (OTP), and gas generator (GG).

The end results of interest include the failure propagation paths that lead to either a loss-of-mission (LOM) or loss-of-crew (LOC) outcome. Since a launch vehicle design may allow for one engine out (i.e., one engine can "fail" or shut down), a second engine out would compromise the mission and prompt an abort, causing a LOM. A second engine out would occur if the propagation path from a failure initiator eventually reached a critical component of another engine and caused it to fail. A high-pressure helium tank burst could cause a catastrophic failure to the vehicle stage. The severity of such a failure is dependent on many factors, but for this analysis high-pressure tank burst is assumed to lead to LOC.

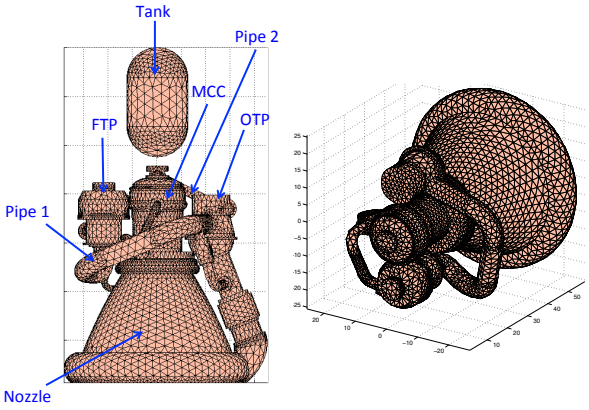


*Figure 2 – Simplified Engine Mesh Model*

The 3D engine CAD geometry is imported into NASTRAN, a finite element analysis tool that generates a surface mesh of the object with defined vertices, faces, and surface normals. This information is then imported into the failure propagation model. Figure 2 shows the simplified mesh model consisting of the MCC, FTP, OTP, fuel and hot gas pipes, and nozzle.

The engine bay assembly consists of four engines and four high-pressure helium tanks. Figure 3 shows a representation of the generic launch vehicle engine bay configuration. The engines are positioned in a circular arrangement and the helium tanks are positioned between two adjacent engines at a distance above them. Each engine is rotated 90 degrees with respect to adjacent engines for rotational symmetry. The engine was sized to fit the engine bay, each with a length of about 1.44 meters and maximum diameter of about 1.3 meters. Each tank has a length of about 0.76 meters and maximum diameter of about 0.42 meters. There are 28 components in total: 6 components per engine and one component for each tank. Table 1 summarizes the number of surface triangles for each component type.

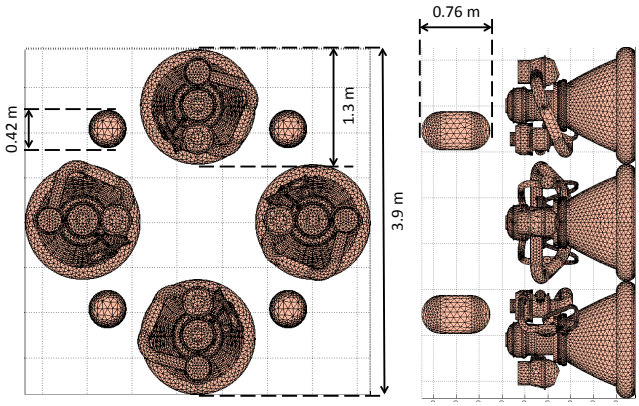


*Figure 3 – Engine Bay Configuration: Top View (Left) and Side View (Right)*

| Component | Number of Surface Triangles |
|---|---|
| MCC | 2930 |
| FTP | 2084 |
| OTP | 942 |
| Nozzle | 6076 |
| Pipe 1 | 1392 |
| Pipe 2 | 2536 |
| Tank | 1016 |

*Table 1 – Component Surface Triangle Count*

### 3.1 Test Case Inputs

Tables 2a and 2b summarize the debris and blast propagation model inputs, respectively. The MCC and tanks are given a range for the number of debris pieces generated for each Monte Carlo trial, a velocity magnitude of 300 m/s for each debris piece, and uniform distributions for each angle of spread. The turbopumps, however, do not come fully apart and are therefore assigned a fixed number of blades (as debris pieces) that propagate at a much higher speed of 1,000 m/s. Their distribution is uniform circumferentially with respect to the spin axis and normal around the spin axis normal. The rest of the components (both pipes and nozzle) are treated as static elements—they are allowed to be struck but do not generate any debris fields themselves. The pipes can fail in the sense that a leakage can occur if struck with enough energy or pressure, but the nozzle is assumed to not fail since the majority of the component does not have any pressurized liquids or gasses flowing through it. All defined values are conservative and every debris strike is treated as a failure.

Blast model inputs also only consider components that can physically break. The MCC and tanks are the only pressurized components. The tanks are highly pressurized and have a much lower overpressure threshold than the MCC and turbopumps.

| Component | # Pieces | Velocity (m/s) | Phi (deg) | Phi Type | Theta (deg) | Theta Type | Mass (kg) |
|---|---|---|---|---|---|---|---|
| MCC | 2-20 | 300 | 0-360 | Uniform | 0-360 | Uniform | 45 |
| FTP | 10 | 1000 | 0 mean, 15 std | Normal | 0-360 | Uniform | 0.23 |
| OTP | 10 | 1000 | 0 mean, 15 std | Normal | 0-360 | Uniform | 0.23 |
| Tank | 5-100 | 300 | 0-360 | Uniform | 0-360 | Uniform | 45 |

| Component | Pressure (psi) | Specific Heat Ratio | Volume ($m^3$) | OP Failure (psi) |
|---|---|---|---|---|
| MCC | 1350 | 1.4 | 0.024 | 500 |
| FTP | - | 1.4 | 0.029 | 500 |
| OTP | - | 1.4 | 0.028 | 500 |
| Tank | 4500 | 1.4 | 0.086 | 100 |

*Table 2 – Input Parameters for (a) Debris Field (Top) and (b) Blast Wave (Bottom)*

### *3.2 Test Case Results*

Three test case scenarios were considered: source failures from the MCC, FTP, and OTP. Only one engine was tested since it was assumed that the rotational symmetry would yield similar results for the other engines, whose inputs are all identical. Each test case simulation was comprised of 1,000 Monte Carlo trials at a 10-kilometer altitude (arbitrarily chosen). Probabilities of resulting debris strikes for every case were intuitive: components closer to the initiating failure source were struck more often than those further away. Additionally, strike probabilities from the FTP initiator were higher overall, which is also intuitive considering it is closer in proximity to surrounding components than the MCC or OTP. Table 3 summarizes the engine out and tank burst failure probabilities from each simulation. As mentioned, every strike is conservatively considered to cause a failure, so the probabilities are highly exaggerated. No failures due to blast wave propagation were observed.

| Initiator | Engine Out | Tank Burst |
|---|---|---|
| MCC | 0.688 | 0.419 |
| FTP | 0.857 | 0.425 |
| OTP | 0.648 | 0.341 |

*Table 3 – Engine Out and Tank Burst Probabilities for Each Initiator*

An interesting observation from the stored sequence data for each test case is the uniqueness of the specific chain of failure propagation events. In the first case where the MCC is the initiator, about 76% of the Monte Carlo trials resulted in a unique sequence of events (i.e., had a probability of 1E-3). Of the 24% that experienced recurring sequences, the most probable outcomes included no strikes from the initiator (5.7%) and the initiator only striking its own nozzle (4.6%). The remaining sequence outcomes, with probabilities less than 2% each, did not involve strikes to any other engine component (i.e., no chain reaction).

For the OTP initiator, about 67% of the sequences were unique. Of the 33% that experienced recurring sequences, the most probable outcome was the initiator striking its own engine's larger pipe (5.8%). The remaining sequence outcomes, with probabilities less than 3% each, did not involve strikes to any other engine component. For the FTP initiator, almost 90% of the Monte Carlo trials resulted in a unique sequence of events. The highest recurring sequence of the remaining 10% was the initiator striking its own MCC and smaller pipe (1.3%) and the initiator striking just the smaller pipe (1%). The remaining recurring sequence outcomes had probabilities of less than 1% each and did not involve strikes to other engine components.

The unpredictability of these self-evolving scenarios shows how pre-defining a list of propagation paths can miss a significant amount of the involved risk. Many of the critical LOM and LOC outcomes do not manifest from a direct strike by the initiator, but rather through a series of complicated chain reactions that cannot be predicted by intuition alone. This fact validates the use of the described modeling approach, which can be instrumental in assessing and mitigating risks to crew systems during a launch vehicle abort.

## *REFERENCES*


1. Mathias, D. L., Go, S., Gee, K., and Lawrence, S. "Simulation Assisted Risk Assessment Applied to Launch Vehicle Conceptual Design," *Annual Reliability and Maintainability Symposium,* 2008.
2. Stamatelatos, Michael and Dezfuli, Homayoon. *Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners*, 2nd Ed. Prepared for Office of Safety and Mission Assurance, Washington, D.C., December 2011.
3. Badouel, Didier. "An Efficient Ray-Polygon Intersection," *Graphics Gems I*, 1990.
4. Akenine-Moller, Tomas, Eric Haines, and Naty Hoffman. "Intersection Test Methods - Ray/Triangle Intersection." *Real-Time Rendering*. Wellesley, MA: A K Peters, Ltd., 2008.
5. Kingery, Charles, and Bulmash, Gerald, "Airblast Parameters from TNT Spherical Air Burst and Hemispherical Surface Burst," Ballistic Research Laboratory Technical Report ARBRL-TR-02555, April 1984.
6. Wade, Mark. "J-2X." *Encyclopedia Astronautica*. http://www.astronautix.com/engines/j2x.htm.


## *BIOGRAPHIES*


Donovan L. Mathias
NASA Ames Research Center
Mail Stop 258-5, Rm. 202
Moffett Field, CA 94035, USA

e-mail: donovan.mathias@nasa.gov


Donovan Mathias is an Aerospace Engineer in the NASA Advanced Supercomputing (NAS) Division at NASA Ames Research Center. He has been at Ames since 1992, during which time he has worked extensively in the field of computational modeling. He has spent the last fourteen years developing risk assessment tools and creating risk models that incorporate physics-based analyses. He has served as PI for the Simulation Assisted Risk Assessment (SARA) project, has

performed risk assessments for multiple NASA architecture studies, and was the Crew Safety and Reliability Manager for the Ares I Launch Vehicle. Currently, Donovan is the Engineering Risk Assessment Team Lead at Ames Research Center. Dr. Mathias earned his B.S. and M.S. degrees in Aeronautical Engineering from California Polytechnic State University, San Luis Obispo and his Ph.D. in Aeronautics and Astronautics from Stanford University.

Samira A. Motiwala
University Space Research Association (USRA)
NASA Ames Research Center
Moffett Field, CA 94035, USA

e-mail: samira.a.motiwala@nasa.gov

Samira Motiwala is an Aerospace Engineering intern sponsored by USRA's Education Associates Program (EAP) at the NASA Ames Research Center. She has been at Ames working in the NASA Advanced Supercomputing (NAS) Division since 2013, supporting projects for the Engineering Risk Assessment team. Samira earned her B.S. degree in Aerospace Engineering from California State Polytechnic University, Pomona, and her M.S. degree in Aeronautics and Astronautics from Stanford University.