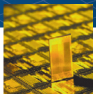


Designing for Reliability and Robustness

By Stuart Kozola

No design is free from uncertainty or natural variation. For example, how will the design be used? How will it respond to environmental factors or to changes in manufacturing or operational processes? These kinds of uncertainty compound the challenge of creating designs that are reliable and robust–designs that perform as expected over time and are insensitive to changes in manufacturing, operational, or environmental factors.

Products Used

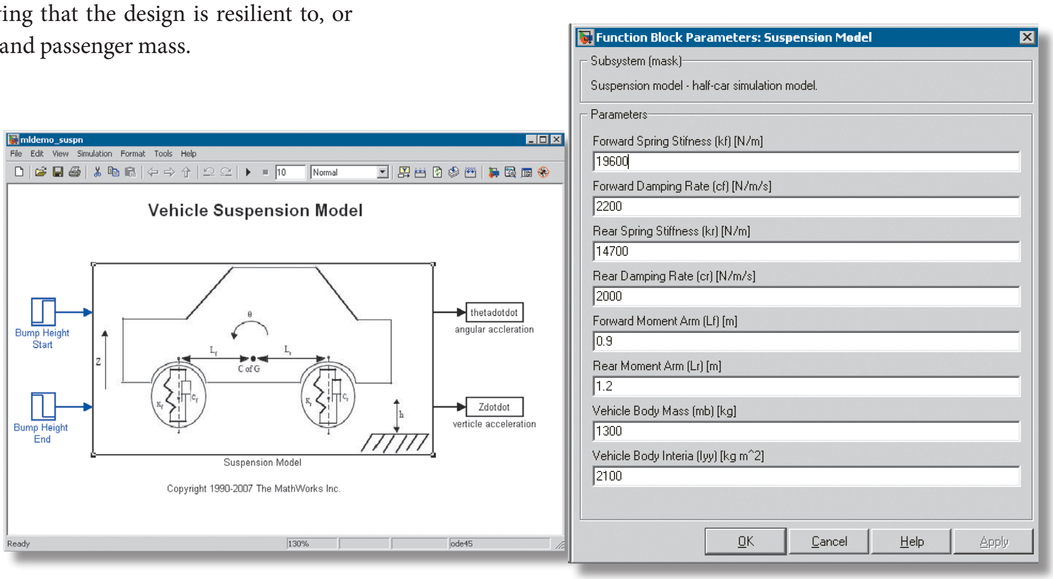
- MATLAB®
- Simulink®
- Statistics Toolbox
- Optimization Toolbox

Using an automotive suspension system as an example, this article describes tools and techniques in MATLAB®, Statistics Toolbox™, and Optimization Toolbox™ that let you extend a traditional design optimization approach to account for uncertainty in your design, improving quality and reducing prototype testing and development effort.

We begin by designing a suspension system that minimizes the forces experienced by front- and rear-seat passengers when the automobile travels over a bump in the road. We then modify the design to account for suspension system reliability; we want to ensure that the suspension system will perform well for at least 100,000 miles. We conclude our analysis by verifying that the design is resilient to, or unaffected by, changes in cargo and passenger mass.

Performing Traditional Design Optimization

Our Simulink® suspension system model (Figure 1) has two inputs—the bump starting and ending height—and eight adjustable parameters. We can modify the four parameters that define the front and rear suspension system stiffness and damping rate: *kf, kr, cf, cr*. The remaining parameters are defined by applying passenger and cargo loading to the vehicle, and are not considered to be design parameters.



Function Block Parameters: Suspension Model

Subsystem (mask)

Suspension model - half-car simulation model.

Parameters

Forward Spring Stifness (kf) [N/m]: 19600

Forward Damping Rate (cf) [N/m/s]: 2200

Rear Spring Stiffness (kr) [N/m]: 14700

Rear Damping Rate (cr) [N/m/s]: 2000

Forward Moment Arm (Lf) [m]: 0.9

Rear Moment Arm (Lr) [m]: 1.2

Vehicle Body Mass (mb) [kg]: 1300

Vehicle Body Interia (Iyy) [kg m^2]: 2100

Figure 1. Simulink® model of an automotive suspension system and dialog showing the model's defining parameters.

The model outputs are angular acceleration about the center of gravity (thetadotddot, $\ddot{\theta}$) and vertical acceleration (zdotdot, $\ddot{z}$). Figure 2 illustrates the model response for our initial design to a simulated bump in the road.

Our goal is to set the parameters *kf, kr, cf,* and *cr* to minimize the discomfort that front- and rear-seat passengers experience as a result of traveling over a bump in the road. We use acceleration as a proxy for passenger discomfort. The design optimization problem can be summarized as follows:

Objective: Minimize peak and total acceleration ($\ddot{z}$, $\ddot{\theta}$)

Design variables: Front/rear spring/shock absorber design (*kr, kr, cf, cr*)

Design constraints: Car is level when at rest.
Suspension system maintains a natural frequency of vibration below 2 Hz.
Damping ratio remains between 0.3 and 0.5.

This problem is nonlinear in both response (Figure 2) and design constraints. To solve it, a nonlinear optimization solver is required. The Optimization Toolbox solver `fmincon` is designed specifically for this type of problem.

We begin by casting our optimization problem into the form accepted by `fmincon`. The table below summarizes the problem formulation that `fmincon` accepts and the definition of the suspension problem in MATLAB syntax.

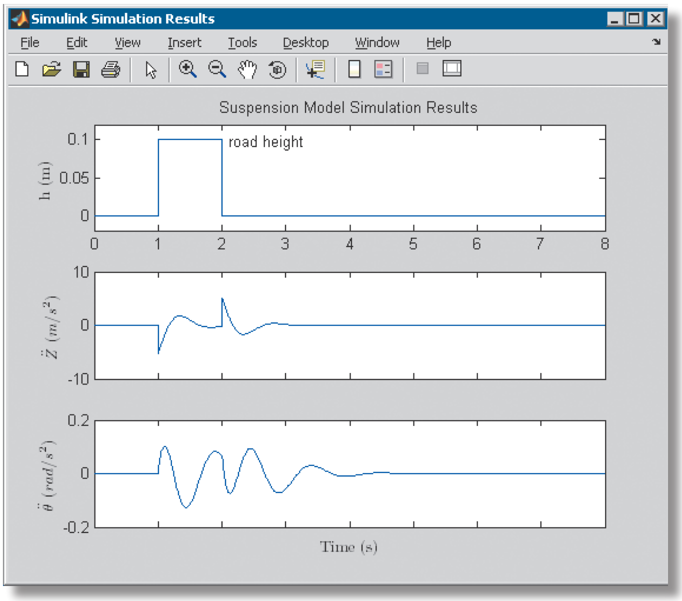


Figure 2. Simulation results for the initial design.

	`fmincon` **Standard Form**	**Suspension Problem (MATLAB M-code)**
Objective	$\min_{x} f(x)$	`myCostFcn(x,simParms)  (see Figure 3)`
Design variables	x	`x = [kf, cf, kr, cr]`
Nonlinear constraints	$C(x) \leq 0$ $C_{eq}(x) = 0$	`mynonlcon(x,simParms)  (see Figure 3)`

Linear constraints	$A \cdot x \leq b$
	$A_{eq} \cdot x = b_{eq}$
		`A   = []; % none for this problem`
		`b   = [];`
		`Aeq = [Lf 0 -Lr 0]; % level car`
		`beq = 0;`

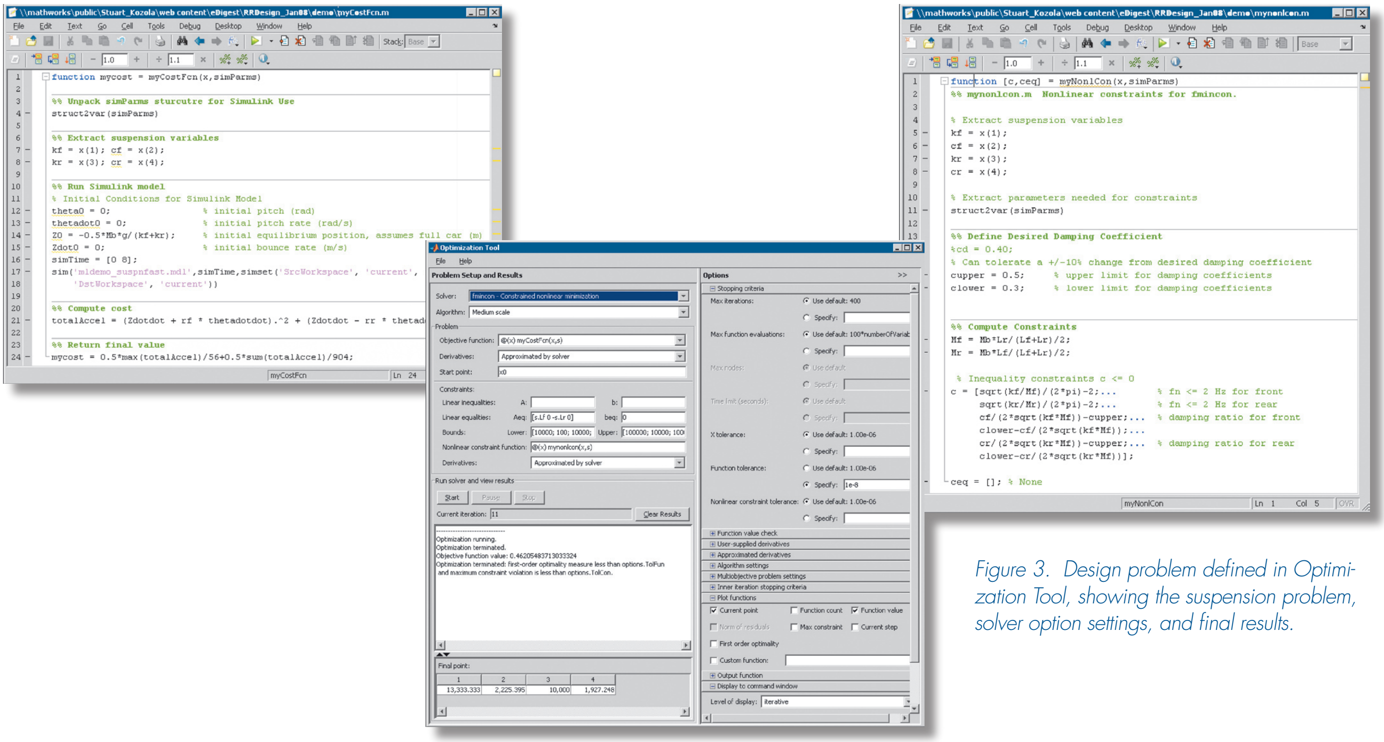
Bound constraints	$lb \leq x \leq ub$	`lb = [10000; 100; 10000; 100];`
		`ub = [100000; 10000; 100000; 10000];`

The design objective is defined as an M-file function `myCostFcn` that accepts two inputs: the design vector x and simParms (Figure 3). x contains our design variables for the suspension system. simParms is a structure that passes in the remaining defining parameters of the Simulink model (*Mb, Lf, Lr,* and *Iyy*). `myCostFcn` runs the suspension model defined by `x` and `simParms` and returns a measure of passenger discomfort, calculated as the weighted average of the peak and total acceleration, as shown in Figure 3. Passenger discomfort is normalized so that our initial design has a discomfort level of 1.

Nonlinear constraints are defined in the M-file function `mynonlcon` that returns values for `c(x)` and `ceq(x)`. The linear and bound constraints are defined as shown in the table as constant coefficient matrices (**A, Aeq**) or vectors (*b, beq, lb, ub*).

Figure 3 shows our problem defined and solved using the Optimization Tool graphical user interface (`optimtool`), which simplifies the tasks of defining an optimization problem, choosing an appropriate solver, setting solver options, and running the solver.

Using a traditional optimization approach, we found that the optimal design was one where x = [*kf, cf, kr, cr*] = [13333, 2225, 10000, 1927].



\\mathworks\public\Stuart_Kozola\web content\eDigest\RRDesign_Jan08\demo\myCostFcn.m

```
function mycost = myCostFcn(x,simParms)

%% Unpack simParms sturcutre for Simulink Use
struct2var(simParms)

%% Extract suspension variables
kf = x(1); cf = x(2);
kr = x(3); cr = x(4);

%% Run Simulink model
% Initial Conditions for Simulink Model
theta0 = 0;                 % initial pitch (rad)
thetadot0 = 0;              % initial pitch rate (rad/s)
Z0 = -0.5*Mb*g/(kf+kr);     % initial equilibrium position, assumes full car (m)
Zdot0 = 0;                  % initial bounce rate (m/s)
simTime = [0 8];
sim('mldemo_suspnfast.mdl',simTime,simset('SrcWorkspace', 'current',
    'DstWorkspace', 'current'))

%% Compute cost
totalAccel = (Zdotdot + rf * thetadotdot).^2 + (Zdotdot - rr * thetad

%% Return final value
mycost = 0.5*max(totalAccel)/56+0.5*sum(totalAccel)/904;
```

Optimization Tool

Problem Setup and Results

Solver: fmincon - Constrained nonlinear minimization

Algorithm: Medium scale

Problem

Objective function: @(x) myCostFcn(x,s)

Derivatives: Approximated by solver

Start point: x0

Constraints:

Linear inequalities: A: b:

Linear equalities: Aeq: [s.Lf 0 -s.Lr 0] beq: 0

Bounds: Lower: [10000; 100; 10000; Upper: [100000; 10000; 100

Nonlinear constraint function: @(x) mynonlcon(x,s)

Derivatives: Approximated by solver

Run solver and view results

Current iteration: 11

```
-----------------------------
Optimization running.
Optimization terminated.
Objective function value: 0.46205483713033324
Optimization terminated: first-order optimality measure less than options.TolFun
 and maximum constraint violation is less than options.TolCon.
```

Final point:

1	2	3	4
13,333.333	2,225.395	10,000	1,927.248

Options

⊟ Stopping criteria

Max iterations: ● Use default: 400 ○ Specify:

Max function evaluations: ● Use default: 100*numberOfVariab ○ Specify:

Max nodes: ● Use default ○ Specify:

Time limit (seconds): ● Use default ○ Specify:

X tolerance: ● Use default: 1.00e-06 ○ Specify:

Function tolerance: ○ Use default: 1.00e-06 ● Specify: 1e-8

Nonlinear constraint tolerance: ● Use default: 1.00e-06 ○ Specify:

⊞ Function value check

⊞ User-supplied derivatives

⊞ Approximated derivatives

⊞ Algorithm settings

⊞ Multiobjective problem settings

⊞ Inner iteration stopping criteria

⊟ Plot functions

☑ Current point ☐ Function count ☑ Function value

☐ Norm of residuals ☐ Max constraint ☐ Current step

☐ First order optimality

☐ Custom function:

⊞ Output function

⊟ Display to command window

Level of display: iterative

\\mathworks\public\Stuart_Kozola\web content\eDigest\RRDesign_Jan08\demo\mynonlcon.m

```
function [c,ceq] = myNonlCon(x,simParms)
%% mynonlcon.m  Nonlinear constraints for fmincon.

% Extract suspension variables
kf = x(1);
cf = x(2);
kr = x(3);
cr = x(4);

% Extract parameters needed for constraints
struct2var(simParms)

%% Define Desired Damping Coefficient
%cd = 0.40;
% Can tolerate a +/-10% change from desired damping coefficient
cupper = 0.5;      % upper limit for damping coefficients
clower = 0.3;      % lower limit for damping coefficients


%% Compute Constraints
Mf = Mb*Lr/(Lf+Lr)/2;
Mr = Mb*Lf/(Lf+Lr)/2;

 % Inequality constraints c <= 0
c = [sqrt(kf/Mf)/(2*pi)-2;...         % fn <= 2 Hz for front
     sqrt(kr/Mr)/(2*pi)-2;...         % fn <= 2 Hz for rear
     cf/(2*sqrt(kf*Mf))-cupper;...    % damping ratio for front
     clower-cf/(2*sqrt(kf*Mf));...
     cr/(2*sqrt(kr*Mf))-cupper;...    % damping ratio for rear
     clower-cr/(2*sqrt(kr*Mf))];

ceq = []; % None
```

Figure 3. Design problem defined in Optimization Tool, showing the suspension problem, solver option settings, and final results.

Figure 4 shows a standard Optimization Toolbox solution progress plot. The top plot shows the current value of the design variables for the current solver iteration, which at iteration 11 is the final solution. The bottom plot shows the objective function value (passenger discomfort relative to the initial design) across solver iterations. This plot shows that our initial design (iteration 0) had a discomfort level of 1, while the optimal design, found after 11 iterations, has a discomfort level of 0.46 – a reduction of 54% from our initial design.

Ensuring Suspension System Reliability

Our optimal design satisfies the design constraints, but is it a reliable design? Will it perform as expected over a given time period? We want to ensure that our suspension design will perform as intended for at least 100,000 miles.

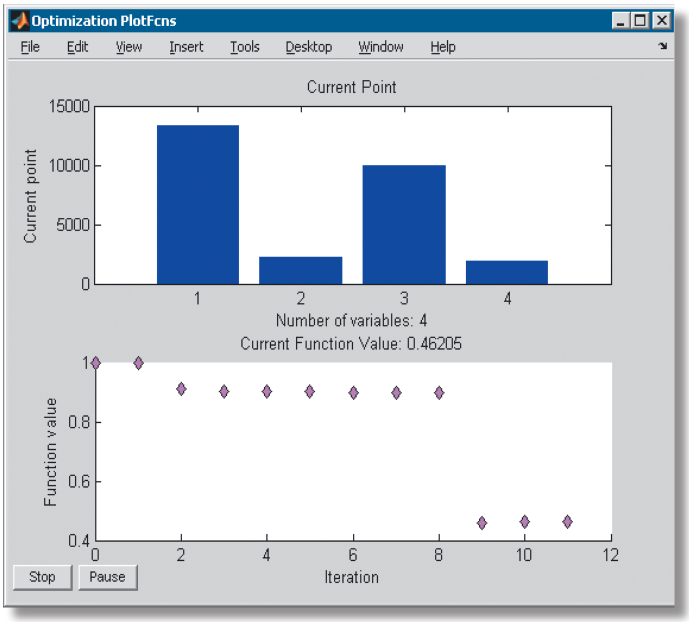


Figure 4. Solution progress plot showing the final value of x (top) and the objective function value as a function of solver iteration (bottom).

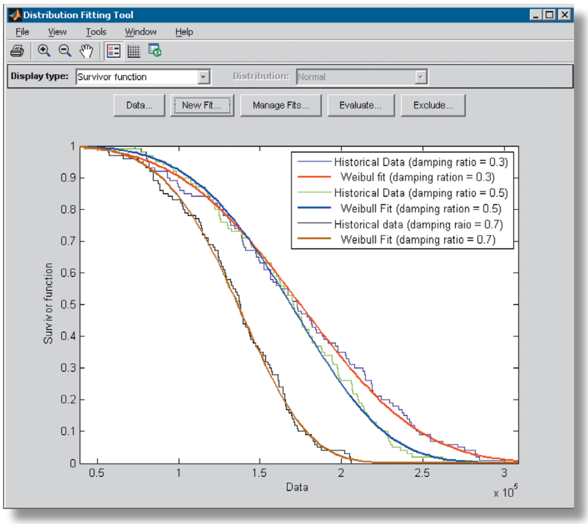

Distribution Fitting Tool
File
View
Tools
Window
Help
Display type:
Survivor function
Distribution:
Normal
Data...
New Fit...
Manage Fits...
Evaluate...
Exclude...
Historical Data (damping ratio = 0.3)
Weibul fit (damping ration = 0.3)
Historical Data (damping ratio = 0.5)
Weibull Fit (damping ration = 0.5)
Historical data (damping raio = 0.7)
Weibull Fit (damping ratio = 0.7)
Survivor function
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.5
1
1.5
2
2.5
3
Data
x 10^5

Figure 5. Fit of a Weibull distribution to five-year survival data for similar suspension system designs. The horizontal axis is the number of miles traveled, and the vertical axis is the survival rate (1- failure rate).

To estimate the suspension's reliability, we use historical maintenance data for similar suspension system designs (Figure 5). The horizontal axis represents the driving time, reported as miles. The vertical axis shows how many suspension systems had degraded performance requiring repair or servicing. The different sets of data apply to suspension systems with different damping ratios. The damping ratio is defined as

$$\eta = \frac{c}{2\sqrt{kM}}$$

where c is the damping coefficient (*cf* or *cr*), k is the spring stiffness (*kf* or *kr*), and M is the amount of mass supported by the front or rear suspension. The damping ratio is a measure of the stiffness of the suspension system.

We fit Weibull distributions to the historical data using the Distribution Fitting Tool (`dfittool`). Each fit provides a probability model that we can use to predict our suspension system reliability as a function of miles driven. Collectively, the three Weibull fits let us predict how the damping ratio affects the suspension system reliability as a function of miles driven. For example, the optimal design found previously has a damping ratio for the front and rear suspension of 0.5. Using the plots in Figure 5, we can expect that after 100,000 miles of operation, our design will have 88% of the original designs operating without the need for repair. Conversely, 12% of the original designs will require repair before 100,000 miles.

We want to improve our design so that it has a 90% survival rate at 100,000 miles of operation. We add this reliability constraint to our traditional optimization problem by adding a nonlinear constraint to `mynonlcon`.

```
% maximum probability of shock absorber failure
Plimit = 0.90;
A  = @(dampRatio) -1.0129e+005. *...
 dampRatio.^2 -28805. * dampRatio + 2.18 31e+005;
B  = @(dampRatio) 1.6865. * dampRatio.^2 -1.8534. *...
 dampRatio + 4.1507;
Ps = @(miles, dampRatio) 1 -...
 wblcdf(miles, A(dampRatio), B(dampRatio));

% Add inequality constraint to existing constraints
% keep original constraints
c = [c;
% front reliability constraint
Plimit- Ps(cdf, Mileage);...
% rear reliability constraint
Plimit- Ps(cdr, Mileage)];
```

We solve the optimization problem as before using `optimtool`. The results, summarized in Figure 6, show that including the reliability constraint changed the design values for cf and cr and resulted in a slightly higher discomfort level. The reliability-based design still performs better than the initial design.

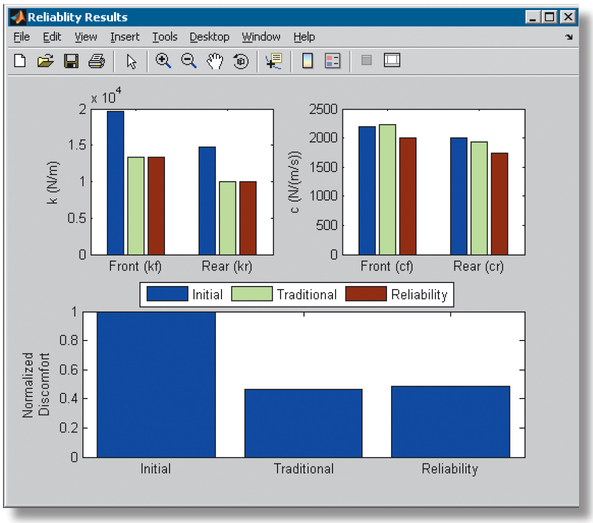


Figure 6. Summary of results showing how design parameters trade off with performance.

Optimizing for Robustness

Our design is now reliable—it meets our life and design goals—but it may not be robust. Operation of the suspension-system design is affected by changes in the mass distribution of the passengers or cargo loads. To be robust, the design must be insensitive to changes in mass distribution.

To account for a distribution of mass loadings in our design, we use Monte Carlo simulation, repeatedly running the Simulink model for a wide range of mass loadings at a given design. The Monte Carlo simulation will result in the model output having a distribution of values for a given set of design variables. The objective of our optimization problem is therefore to minimize the mean and standard deviation of passenger discomfort.

We replace the single simulation call in `myCostFcn` with the Monte Carlo simulation and optimize for the set of design values that minimize the mean and standard deviation of total passenger discomfort. We'll assume that the mass distribution of passengers and trunk loads follow Rayleigh distributions and randomly sample the distributions to define conditions to test our design performance.

```
nRuns = 10000;
% front passengers - adults (kg)
front = 40 + raylrnd(40, nRuns, 1);
% rear passengers – includes children
back = 1.36 + raylrnd(40, nRuns, 1);
% additional mass for luggage (kg)
trunk = raylrnd(10, nRuns, 1);
```

The total mass, center of gravity, and moment of inertia are adjusted to account for the changes in mass distribution of the automobile.

```
% total mass
mcMb = Mb + front + back + trunk;
% update center of mass
mcCm = (front. * rf - back. * rr - trunk. * rt)./mcMb;
% Adjust moment of inertia
mcIyy = Iyy + front. * rf.^2 + back. *...
 rr.^2 + trunk. * rt.^2- mcMb. * mcCm.^2;
```

The optimization problem, including the reliability constraints, is solved as before in `optimtool`. The results are shown in Figure 7. The robust design has an average discomfort level that is higher than that in the reliability-based design, resulting in a design with higher damping coefficient values.

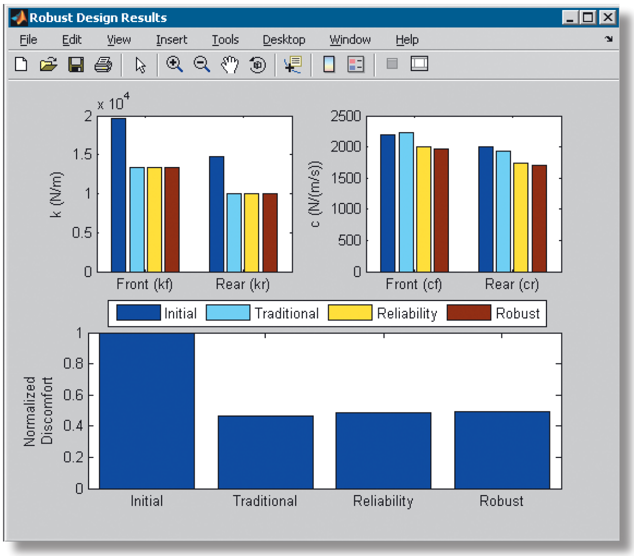


Figure 7. Summary of design optimization results.

The discomfort measure reported in Figure 7 is an average value for the robust design case. Figure 8 displays a scatter-matrix plot that summarizes variability seen in discomfort as a result of different mass loadings for the robust design case.

The diagonals show a histogram of the variables listed on the axis. The plots above and below the diagonal are useful for quickly finding trends across variables. The histograms for front, back, and trunk represent the distribution of the inputs to our simulation. The histogram for discomfort shows that it is concentrated around the value 0.47 and is approximately normally distributed. The plots below the diagonal do not show a strong trend in discomfort with trunk loading levels, indicating that our design is robust to changes in this parameter.

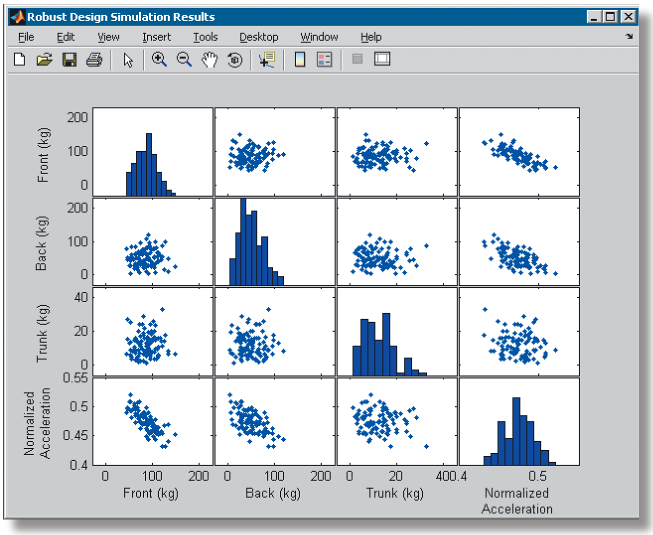
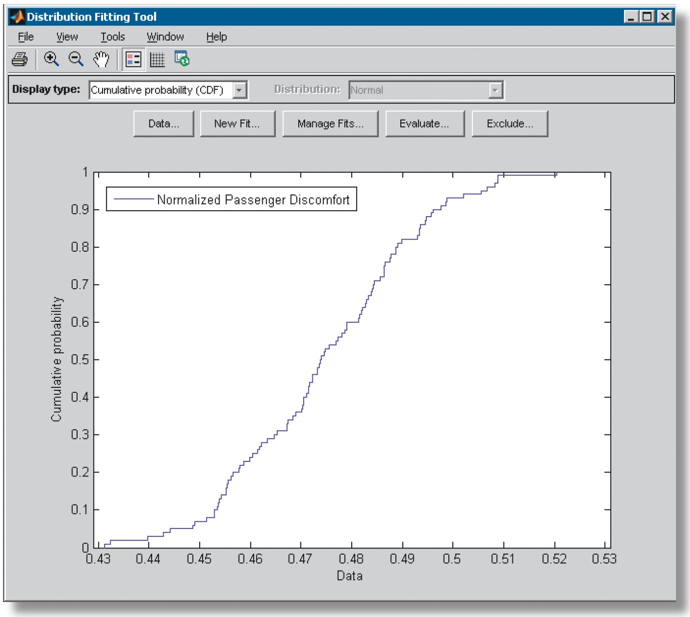


Figure 8. Selected results from the robust design solution. Plotted results include front-passenger loading, back-passenger loading, cargo or trunk loading, and normalized passenger discomfort.



Display type: Cumulative probability (CDF)

Distribution: Normal

Figure 9. Cumulative distribution plot of discomfort results.

There is a definite trend associated with the front loading on discomfort. Front loading appears to be approximately linear with a min of 0.43 and a max of 0.52. A trend between back loading and discomfort can also be seen, but from this plot it is difficult determine if it is linear. From this plot, it is difficult to determine whether our design is robust with respect to back loading.

Using a cumulative probability plot of the discomfort results (Figure 9), we estimate that 90% of the time, passengers will experience less than 50% of the discomfort they would have experienced with the initial design. We can also see that our new design maintains a normalized level of discomfort below 0.52 nearly 100% of the time. We therefore conclude that our optimized design overall is robust to expected variation in loadings and will perform better than our initial design.

Design Trade-Offs

This article showed how MATLAB, Statistics Toolbox, and Optimization Toolbox can be used to capture uncertainty within a simulation-based design problem in order to find an optimal suspension design that is reliable and robust.

We began by showing how to reformulate a design problem as an optimization problem that resulted in a design that performed better than the initial design. We then modified the optimization problem to include a reliability constraint. The results showed that a trade-off in performance was required to meet our reliability goals.

We completed our analysis by including the uncertainty that we would expect to see in the mass loading of the automobile. The results showed that we derived a different design if we accounted for uncertainty in operation and quantified the expected variability in performance. The final design traded performance to maintain reliability and robustness. ■

Resources

VISIT
www.mathworks.com

TECHNICAL SUPPORT
www.mathworks.com/support

ONLINE USER COMMUNITY
www.mathworks.com/matlabcentral

DEMOS
www.mathworks.com/demos

TRAINING SERVICES
www.mathworks.com/training

THIRD-PARTY PRODUCTS AND SERVICES
www.mathworks.com/connections

Worldwide CONTACTS
www.mathworks.com/contact

E-MAIL
info@mathworks.com

91545v00 03/08

For More Information

- "Using Statistics to Analyze Uncertainty in System Models."
  MATLAB Digest, May 2007
- "Improving an Engine Cooling Fan Using Design for Six Sigma Techniques."
  MATLAB Digest, May 2006
- "Modeling Survival Data with Statistics Toolbox."
  MATLAB Digest, November 2004
- *www.mathworks.com/company/newsletters*