Fast and Robust Stochastic Structural Optimization

Qiaodong Cui,1 Timothy Langlois,2 Pradeep Sen,1 Theodore Kim3
University of California, Santa Barbara1 Adobe Research2 Yale University3

Abstract
Stochastic structural analysis can assess whether a fabricated object will break under real-world conditions. While this approach is powerful, it is also quite slow, which has previously limited its use to coarse resolutions (e.g., $26 \times 34 \times 28$). We show that this approach can be made asymptotically faster, which in practice reduces computation time by two orders of magnitude, and allows the use of previously-infeasible resolutions. We achieve this by showing that the probability gradient can be computed in linear time instead of quadratic, and by using a robust new scheme that stabilizes the inertia gradients used by the optimization. Additionally, we propose a constrained restart method that deals with local minima, and a sheathing approach that further reduces the weight of the shape. Together, these components enable the discovery of previously-inaccessible designs.

1. Introduction
Recently, a wide range of computational techniques have been developed to assist in the design of objects manufactured using additive fabrication (see, e.g., excellent surveys of the state-of-the-art [LEM+17, MS19]). One key concern is the robustness of the manufactured object (i.e., the conditions in which it will break), and various failure analysis algorithms have been developed to estimate the failure cases of the object in order to improve its design.

Traditionally, failure analysis is performed for worst-case scenarios, and while such analyses are critical in certain scenarios (e.g., bridge construction), they can be overly conservative in others, such as figurine design. In these (more common) situations, it is more realistic to ask whether an object is robust in real-world situations, such as being dropped on the floor or down a flight of stairs. In this case, reinforcing portions of the object that are unlikely to experience an impact because they are already shielded by more robust regions is clearly sub-optimal. Unlike mechanical scenarios should not artificially constrain the design space, and the manufactured designs should be optimized for the “common case.”

Along these lines, Langlois et al. [LSD+16] recently proposed a
stochastic structural analysis} method that used a rigid body simulator to sample a more realistic space of real-world impacts and construct a probability map of predicted failures. This probability map was then incorporated as a constraint into a topology optimization, and used to solve a context-aware inverse design problem that produced geometrical models more robust to realistic impacts. Unfortunately, the method is quite computationally intensive, where even a single optimization step over a small $26 \times 34 \times 28$ model takes over an hour to compute.

We observe that this computational expense arises for two main reasons. First, computing the probability gradients is quadratic in the number of elements. Second, the inertia gradient that arises from the rigid body simulation is computed using a finite difference scheme that leads to instabilities, which in turn negatively impacts the convergence of the optimization. In this paper, we present an approach that is both asymptotically faster and more numerically robust than Langlois et al. [LSD∗16]. We achieve both of these goals directly; no approximation is applied to the original algorithm. In the first case, we show that a careful analysis of the probability gradient can yield a computation that is only linear in the number of elements. In the second case, we use a combination of Gaussian Mixture Models and an alternate central-differencing method to arrive at a more stable version of the inertia gradient.

Even with these improvements, the optimization can still stall at local minima. We therefore introduce a constrained restart method that is able to make further progress and discover interesting new structures that were otherwise inaccessible to the original algorithm. Finally, we show that additional progress can be made by assuming that a thin sheath that is beyond the resolution of the numerical simulation will be printed along the exterior of the object. Our individual technical contributions are as follows:

• Asymptotically faster computation of probability gradients
• A robust scheme for computing inertia gradients that stabilizes the optimization search direction
• A constrained restart strategy that allows the global optimization to climb out of local minima
• A sheathing approach that robustly removes additional mass from the final design

Together, these components comprise a stochastic structural optimization method that is orders of magnitude faster than Langlois et al. [LSD∗16], able to achieve previously-impractical resolutions, and capable of discovering previously-inaccessible designs.

2. Related Work

Since structural optimization was introduced [Ben89, BS95], it has attracted lots of attention in computer graphics. Traditionally, topology optimization involves minimization of compliance. There are many works in computer graphics following this approach, e.g., [WWY∗13, LHZ∗18, LSZ∗14]. However, compliance minimization can overfit to one loading condition, and has difficulty predicting whether objects will fail under realistic conditions.

Instead of minimizing compliance, minimizing the object’s weight subject to a stress constraint offers a better guarantee of the robustness of the object. For example, Lee et al. [LJM12] uses a constraint on the yield stress and minimizes the weight of the object. Similarly, Ulu et al. [UKM19] optimizes the thickness of the shell to minimize the weight, subject to a constraint on the yield stress. Many of these methods use prescribed loadings. This can be useful in some particular cases, e.g., designing a bridge, where the loading can be prescribed in advance. However, in many other cases, the loading might be unknown beforehand.

To capture uncertainty, stochastic finite element analysis has been extensively studied in the engineering community. Stefanou [Ste09] provides an excellent overview. There are two broad classes of methods: 1) the perturbation approach, which uses a Taylor series expansion of the system matrix and solution [LBM86], and 2) the spectral stochastic finite element method, which represents each solution quantity with a series of random Hermite polynomials [GS91]. Monte Carlo simulation (MCS) [PP96] can be used in conjunction with these two methods, which models randomness by solving a deterministic problem many times using different samples of the random variables. These methods are very general, and can consider uncertainties in the loading, geometry, and material behavior of the problem. In our case, we care only about uncertainty in loading conditions, so do not need the complex, expensive machinery to approximate randomness in the system matrix provided by these methods. Our approach (and the previous approach we accelerate [LSD∗16]) is akin to an MCS approach using model reduction to reduce the computational load.

To address uncertainty in the graphics community, worst case structural analysis was introduced by Zhou et al. [ZPZ13]. The method computes a worst case loading scenario where it produces the worst possible stress distribution in the object. Along this line, several different works use the worst case loading to optimize the structure [UMK17, PRZ17, SZB18]. However, it is unknown how often the worst case loading will be present in a realistic scenario, such as a figurine falling and hitting the ground.

To address this limitation, Langlois et al. [LSD∗16] presented a method where the loading of the object was computed from a rigid body simulator that closely mimicked realistic loadings. The work also introduced semantically meaningful failure probabilities that better reflected real-world object failures. The work also presented a structural optimization scheme where the weight of the object was minimized under failure probability constraints. Still, this optimization scheme remained expensive due to the computation of the failure probability gradients.

Our work immediately follows this line. We aim to optimize stochastic structural optimization through three specific contributions: acceleration of the gradient computation, a more robust probability gradient formulation, and a restart strategy to overcome non-optimal local minima during optimization.

3. Stochastic Structural Optimization

Here we briefly summarize the stochastic structural optimization technique of Langlois et al. [LSD∗16].

Notation: We use unbolded lower case for scalars, bolded lower case for vectors (\( \mathbf{0} \)), and bolded upper case for matrices (\( \mathbf{K} \)). When an indexing operation results in a scalar, we unbold the result. We use a colon to denote double-contraction between matrices.
Finite Element Method (FEM): We use a hexahedral uniform grid as our FEM discretization and compute the displacement (u) and element Cauchy stresses (σ) that arise from an external force (f):

\[
\begin{align*}
Ku &= f \\
\sigma &= CBu.
\end{align*}
\]

(1)

Above, \(K \in \mathbb{R}^{3n \times 3n}\) is the stiffness matrix, \(\sigma \in \mathbb{R}^{6m}\) is a vector of per-element Cauchy stresses, \(C \in \mathbb{R}^{6m \times 6m}\) is a constitutive matrix, and \(B \in \mathbb{R}^{6m \times 3n}\) maps \(u\) to Cauchy strains. Stresses and strains are evaluated at the center of each voxel. The quantity \(m\) is the total number of mesh elements, and \(n\) is the number of vertices.

Using the per-element stress,

\[
\sigma_e = \begin{bmatrix}
\sigma_{11} & \sigma_{12} & \sigma_{31} \\
\sigma_{12} & \sigma_{22} & \sigma_{32} \\
\sigma_{31} & \sigma_{32} & \sigma_{33}
\end{bmatrix}
\]

(2)

we compute a scalar, per-element von Mises stress:

\[
S(\sigma_e) = \frac{1}{2} \left( (\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{33} - \sigma_{11})^2 \right) + 3 \left( \sigma_{23}^2 + \sigma_{31}^2 + \sigma_{12}^2 \right).
\]

(3)

A yield stress equals our failure criteria. An element fails if \(S(\sigma_e) > \sigma_f\), and we define “object failure” as the failure of any individual element.

Stochastic Failure Probability: Prior to optimization, we need to estimate an object’s failure probability under various force distributions. Langlois et al. [LSD+16] estimated real-world loadings by using a rigid-body simulation to generate force samples. For each sample, they then computed the maximal von Mises stress across the whole object, and then estimated a probability distribution function (PDF) of maximal stresses. The object fails if its maximal stress is greater than the yield stress, so the survival probability is computed by integrating the PDF from 0 to \(\sigma_f\). The failure probability is one minus the survival probability.

Many force samples (\(\approx 5000\)) are needed to accurately represent the PDF, so principal component analysis (PCA) is used to reduce its computational complexity. We denote each force sample as \(f\), where \(i = 1 \ldots n_s\), and \(n_s\) is the total number of samples. In lieu of performing an FEM analysis for each sample, they compute a reduced force basis, \(F \in \mathbb{R}^{3n \times r}\), where \(r\) is the number of principal components. Each sample is then represented with a reduced configuration, \(\alpha' \in \mathbb{R}^r\), where \(f \approx C\alpha'\). The element stresses for each sample \(i\) can then be computed as:

\[
\sigma_i' = CBK^{-1}C\alpha'.
\]

(4)

By pre-solving \(CBK^{-1}\) for each column in \(F\), significant savings can be achieved when \(r \ll n_s\) (e.g., \(r \approx 100\)).

The normalized, whole-object stress for each \(f\) is then:

\[
s^i = \frac{1}{\sigma_f} \max_e(S(\sigma_e^i)) \quad \text{for} \quad e = 1 \ldots m, \quad i = 1 \ldots n_s
\]

(5)

Using all the stress samples \(s\), we can construct a probability distribution function (PDF) \(p(s)\) for the whole-object stress. The corresponding cumulative distribution (CDF) function is denoted as \(P(s)\). The probability of the object survival \(P(s < 1)\) is then:

\[
P(s < 1) = \int_0^1 p(s) ds.
\]

(6)

Topology Optimization: The failure probability is then used as a constraint in a topology optimization that adds or subtracts materials from some initial design. The overall goal is to reduce the final object weight while satisfying a user defined failure probability \(\Theta\):

\[
\min_{\omega_i} \sum_{i=1}^m \omega_i
\]

s.t. \(P(s < 1) > 1 - \Theta\).

(7)

Above, \(\omega_i\) is a vector of element densities, such that \(\forall e \in (0, 1]\), which is usually initialized to be fully filled (\(\forall e, \omega_e = 1\)). Eqn. 7 is optimized using the Method of Moving Asymptotes (MMA) [Sva02], which requires both the object and constraint gradients. The objective gradient is straightforward to efficiently compute, but the constraint gradient is a major bottleneck because the existing approach is quadratic in the number of elements. We will show that it is possible to reduce its complexity to linear.

4. Our Method

First, in §4.1, we analyze the complexity of computing the probability gradients. By carefully leveraging the structure of the problem, we found the quadratic complexity of previous methods [LSD+16] can be reduced to linear with respect to the number of elements. In practice, this results in a roughly two-order of magnitude speedup.

Next, in §4.2, we show that the existing approach leads to unreliable probability gradients. This negatively impacts the convergence of the optimization and, in turn, the final design. We show how to stabilize these gradients, which leads to higher-quality shapes.

The optimization can still stall at local minima. To address this, we introduce (§4.3) a constrained restart method that identifies and constrains promising structures when the optimization stalls, and applies a perturbation to bump the state out of its local minimum. Finally, we make additional progress by allowing the outer shell of the object to erode, and later restore the visual appearance of the object by adding a lightweight sheath as a post-process.

4.1. Asymptotic Analysis and Acceleration

First, we will analyze the existing method [LSD+16] to show that it runs in \(O(m^3)\). Then we will show that an identical computation can be done in \(O(m)\).

4.1.1. The Previous Quadratic Method

We begin by expanding the derivative of \(p(s)\) from Eqn. 6 in terms of \(s^i\) using the chain rule:

\[
\frac{\partial P(s < 1)}{\partial \omega} = \int_0^1 \sum_{i=1}^m \frac{\partial p}{\partial s^i} \frac{\partial s^i}{\partial \omega} ds.
\]

(8)

The \(\frac{\partial s^i}{\partial \omega}\) term is computed by replacing the discontinuous max function in Eqn. 5 with a smoother \(L^p\) norm. The density term \(\omega_e\) is also
multiplied beforehand to avoid singularities as shown in [LJM12],
\[ s^j = \| \sqrt{\omega} S(\sigma^j) \|_2, \]
yielding:
\[
\frac{\partial s^j}{\partial \omega} = \frac{1}{\bar{S}^j} \left( \sum_{e=1}^{m} \sqrt{\omega} S(\sigma^j)^p \right)^{1-1} \left( \frac{2}{\overline{\omega} e} S(\sigma^j) + \frac{\partial \omega}{\partial \sigma_e} \overline{\sigma}_e \right)
\]
Combining equation 9 with equation 8, we obtain:
\[
\frac{\partial P(s \leq 1)}{\partial \omega} = \sum_{i=1}^{n} a_i \left( b_i + \left( \frac{\partial \sigma_i}{\partial \omega} \right)^{T} c_i \right)
\]
where
\[
a_i = \int_{0}^{1} \frac{\partial p}{\partial s^i} \left( \sum_{e=1}^{m} \sqrt{\omega} S(\sigma^j)^p \right)^{1-1} \cdot \frac{\partial \omega}{\partial \sigma_e} \overline{\sigma}_e
\]
\[
b_i = \sum_{e=1}^{m} \left( \sqrt{\omega} S(\sigma^j)^p \right)^{1-1} / (2 \sqrt{\omega} e)
\]
\[
c_i = \left( \sqrt{\omega} S(\sigma^j)^p \right)^{1-1} \frac{\partial \omega}{\partial \sigma_e}
\]
\[
\frac{\partial \sigma_i}{\partial \omega}
\]
can be computed from equation 3. Next, from equation 4 we compute:
\[
\frac{\partial \sigma^j}{\partial \omega} = -\text{CBK}^{-1} \frac{\partial F}{\partial \omega} \text{U} \alpha^j + \text{CBK}^{-1} \frac{\partial F}{\partial \omega} \text{U} \alpha^j + \text{CBF} \frac{\partial \alpha^j}{\partial \omega}
\]
where \( \text{U} = \text{K}^{-1} \text{F} \). Here, \( \text{I} \) computes the derivative of the stiffness matrix, \( \text{II} \) computes the gradient of the reduced force basis, and \( \text{III} \) computes the gradient of the reduced force coordinates.
Combining equations 10 and 11, we obtain the final probability derivative, which can be computed as:
\[
\frac{\partial P(s \leq 1)}{\partial \omega} = (\text{K}^{-1} \text{YU}^T) : \frac{\partial \text{K}}{\partial \omega} (\text{K}^{-1} \text{Y}) : \frac{\partial F}{\partial \omega} + x + t
\]
where
\[
Y = \frac{n}{4} B^T C^T c_i \otimes \alpha_i, \quad t = \sum_{i=1}^{n} a_i b_i, \quad x = \sum_{i=1}^{n} \frac{\partial \alpha_i}{\partial \omega} \text{U}\text{B}^T C^T c_i.
\]
The main complexity in Eqn. 12 lies in the second term, \( (\text{K}^{-1} \text{Y}) : \frac{\partial F}{\partial \omega} \). We show in Appendix A for a single element \( e \), the force derivative on the right of the double-contraction can be written as
\[
\frac{\partial F}{\partial \omega} = \bar{F} \omega_e.
\]
4.1.2. Our Linear Method
We first observe that in equation 13, the per-element force basis matrix \( \text{F} \) is fixed, and only the smaller \( \omega_e \in \mathbb{R}^{m \times r} \) ever changes. Second, we observe that the final quantity \( (\text{K}^{-1} \text{Y}) : \frac{\partial F}{\partial \omega} \) is all that matters; the per-element intermediate \( \frac{\partial F}{\partial \sigma_e} \) is not strictly required. Therefore, if we pre-contract \( \bar{F} \) with \( \text{K}^{-1} \text{Y} \), we obtain a smaller matrix, \( \bar{F}^Y \text{K}^{-1} \text{Y} \in \mathbb{R}^{m \times r} \). Each element can then be computed as \( \bar{F}^Y \text{K}^{-1} \text{Y} : \omega_{e} \), which is only \( O(r^2) \) per element.
Specifically, each element must compute the product
\[
ge = (\text{K}^{-1} \text{Y}) : (\text{FW}_{e}).
\]
where \( g_e \) is the entry of the second term in Eqn. 12 for element \( e \). Both \( \text{Z} = \text{K}^{-1} \text{Y} \) and \( \bar{F} \) are static, per-element, \( \mathbb{R}^{m \times r} \) matrices that we choose to rewrite \( g_e \) as
\[
ge = \sum_{i,j} (\text{Z})_{ij} (\bar{F}W_{e})_{ij} = \sum_{i,j,k} (\bar{F}W_{e})_{ijk} (\text{Z})_{ijk} = \sum_{i,j,k} (\bar{F} \text{Z})_{ijk} W_{e,ike}
\]
\[
= (\bar{F}^Y \text{Z}) : \omega_e.
\]
Since \( \bar{F}^Y \text{Z} \in \mathbb{R}^{m \times r} \) is fixed for all \( e \), we can precompute it in \( O(m^2 r) \) time. At runtime, an \( O(r^2) \) contraction is performed over \( m \) elements, yielding an \( O(m r^2) \) overall running time.
So far, we have only examined the derivative of the force basis in Eqn. 13. However, naively evaluating \( \omega_e \) also takes \( O(m^2 r) \) time. We show in Appendix B that this can also be reduced to \( O(m r) \) by leveraging matrix sparsity and similar pre-computations.
We show in Table 1 the running time of computing the total probability gradients using the previous quadratic method (estimated) and our linear method. Our method is asymptotically faster, and accelerates this stage of the method by two orders of magnitude, effectively removing it as the bottleneck of the method.

<table>
<thead>
<tr>
<th></th>
<th>Resolution</th>
<th>[LSD*16]</th>
<th>Our Method</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>28 x 32 x 36</td>
<td>1.84 †</td>
<td>0.041</td>
<td>44.9 x</td>
<td></td>
</tr>
<tr>
<td>28 x 44 x 28</td>
<td>6.18 †</td>
<td>0.044</td>
<td>140 x</td>
<td></td>
</tr>
<tr>
<td>40 x 64 x 60</td>
<td>41.7 †</td>
<td>0.128</td>
<td>326 x</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Running time of Eqn. 13 using the quadratic method of Langlois et al. [LSD*16] and our linear method. †Estimated time.

4.2. Stabilizing the Inertia Gradients
4.2.1. Previous Method
We examine the relevant term from Eqn. 12,
\[
x = \sum_{i=1}^{n} \frac{\partial \omega_i}{\partial \omega} \text{U}^T \text{B}^T C^T c_i,
\]
that measures how changing the voxel densities \( \omega \) influence the rigid body force samples \( \alpha^j \). The previous method [LSD*16] evaluates this term using finite differences.
Given \( n_\alpha \) samples \( \alpha^j \in \mathbb{R}^r \), we assume each entry of \( \alpha^j \) is sampled
from a 1D PDF, and take its finite difference over that distribution. The \( j \)th entry of each force sample \( \alpha_j \) is denoted \( \alpha \). We assume \( \alpha \) is a random variable and that \( \alpha_j \) is drawn from the PDF \( c(\alpha) \).

Instead of computing a finite difference for the sample \( \alpha_j \), we perform a finite difference over its distribution. For any \( \alpha \), we can compute its CDF as \( C(\alpha) \), and retrieve \( \alpha_j \) by sampling its inverse using \( \alpha_j = C^{-1}(u) \), where \( u \) is a uniform random variable. Using the properties of the CDF, we obtain:

\[
\frac{\partial C^{-1}(u)}{\partial \omega} = \frac{\partial u}{\partial \omega} = 0.
\]

\[
\frac{\partial C^{-1}(u)}{\partial \omega} = \frac{\partial C}{\partial \alpha_j} \frac{\partial C^{-1}}{\partial \omega} \bigg|_u + \frac{\partial C}{\partial \omega} \bigg|_\alpha = 0.
\]  

(17)

By manipulating Eqn. 17, we obtain the derivative of \( \alpha \):

\[
\frac{\partial \alpha}{\partial \omega} = -\frac{1}{c(\alpha)} \frac{\partial C}{\partial \omega} \bigg|_{\alpha}.
\]

(18)

We insert \( \alpha = \alpha_j \) into Eqn. 18 to compute the gradient for each random sample. Therefore, an important step in computing the inertia gradients is building the distributions \( C(\alpha) \) and \( c(\alpha) \) over the samples \( \alpha_j \), where \( i = 1 \ldots n \).

Langlois et al. [LSD*16] computed \( C(\alpha) \) and \( c(\alpha) \) by fitting a uniform 1D finite element grid over the samples, which represents them as a sum of basis functions: \( c(\alpha) = \sum_{i=1}^{k} \alpha_i \psi_i(\alpha) \). Here, \( k \) is the number of elements of this 1D finite element grid, \( \alpha_i \) is a shape coefficient, and \( \psi_i(\alpha) \) is a symmetric hat function of element \( i \). A Galerkin method is then used to solve for \( \alpha_i \). However, this approach can fit the data poorly. In the top of Fig. 3, we show the PDF \( c(\alpha) \) that results from this approach using \( n \) elements. For a small \( n \), the histogram is fit poorly. As \( n \) increases, ringing artifacts appear. This leads to instabilities when computing \( \frac{\partial \alpha}{\partial \omega} \) in Eqn. 18 because the ringing artifacts can cause the PDF to become negative.

\[ \text{4.2.2. Stabilized Inertia Gradients} \]

We use Gaussian Mixture Models (GMMs) to address this problem. GMMs are widely used to capture discrete distributions [Bi06], and we found that they yield results superior to the FEM approach.

We first expand \( c(\alpha) \) using a set of Gaussians:

\[
c(\alpha) = \sum_{i=1}^{k} \pi_i \mathcal{N}(\alpha | \mu_i, \sigma_i)
\]

(19)

where \( k \) is the number of Gaussians, \( \pi_i \) is a weight, and \( \mathcal{N}(\alpha | \mu_i, \sigma_i) \) is a Gaussian with mean \( \mu_i \) and variance \( \sigma_i \). The parameters, \( \pi_i, \sigma_i, \mu_i \), can be computed from \( \alpha_j \) using expectation maximization [Bi06], and we found that \( k \approx 10 \) usually yields good results. This leads to the superior distribution representations we show in the bottom of Fig. 3.

Returning to Eqn. 18, \( \frac{\partial \alpha}{\partial \omega} \) is also evaluated using finite differences. We assume that all the potential contact positions of the rigid-body lies on the surface of the object. So the rigid-body can be parameterized by its mass, center of mass, and moment of inertia. As a result, \( \alpha_j \) and its CDF \( C(\alpha) \) can be parameterized using a total of 10 variables. Denoting the parameters as \( M_i, i = 1 \ldots 10 \), the finite difference is computed as:

\[
\frac{\partial C}{\partial \omega} \bigg|_{\alpha} = \sum_{i=1}^{10} \frac{\partial C}{\partial M_i} \bigg|_{\alpha} \frac{\partial M_i}{\partial \omega}.
\]

(20)

The \( \frac{\partial M_i}{\partial \omega} \) is straightforward, and we use a centered difference for \( \frac{\partial C}{\partial M_i} \):

\[
\frac{\partial C}{\partial M_i} \bigg|_{\alpha} = \frac{1}{2} \left( C(M_i + \Delta M) |_{\alpha} - C(M_i - \Delta M) |_{\alpha} \right) / \Delta M.
\]

(21)

Evaluating the above equation involves two extra rounds of rigid body simulation with \( M_i + \Delta M \) and \( M_i - \Delta M \) perturbations applied to each parameter, for a total of 20 rounds.

The resulting probability distribution leads to much stabler inertia gradients, as shown in Fig. 4. In that figure, we show the probability gradients from the first three frames of the optimization as well as two frames near the end. The element densities change only slightly during these frames, and yet the FEM-based gradients oscillate wildly. Using our method, the gradient becomes very stable, and greatly improves the convergence of the optimization (Fig. 12).

\[ \text{4.3. Constrained Restart Strategy} \]

Eqn. 7 has a non-linear constraint and a large number of variables, so we use the Method of Moving Asymptotes (MMA) [Sva02] as our optimization algorithm. The non-linearity of the probability constraint causes the optimization to often fall out of the feasible
region. MMA can recover, but often at the cost of oscillatory behavior that converges to sub-optimal local minima [Sva02]. One popular technique for addressing this well-known problem is the Solid Isotropic Material with Penalization (SIMP) model [BS95], but we found it to be insufficient for our case (see Appendix D).

Instead, we found the following constrained restart method to be effective, which can be viewed as a form of either block-cyclic reduction [GVL13] or sand-filling [BMS15]. We observe that when the optimization stalls, it usually has found a preliminary, but promising, reinforcement structure. Therefore, we perform multiple optimization passes where the promising structures from the previous iteration are used as an initial guess.

We isolate these structures by applying threshold \( c \) to the current solution and constraining the results to \( \omega_c = 1 \) in Eqn. 7. Next, we add a perturbation to push the global solution state out of its current local minimum. We compute extension density field \( \beta \) of the stalled solution \( \omega^{i-1} \) and then use \( \min(\omega^{i-1} + \beta, 1) \) as the initial state for the next pass. The extension field is computed as:

\[
\beta = \max \left( 1 - \frac{\text{SDF}(V)}{b_{\beta}}, 0 \right)
\]  

(22)

where \( \text{SDF} \) is a signed distance field, \( V \) is the set of constrained voxels and \( b_{\beta} \) is a bandwidth parameter. Fig. 6 illustrates these quantities. The strategy essentially inflates the existing reinforcement structure for the next optimization pass.

Using the extension field \( \beta \) and constraints on \( V \), we run the next optimization pass. We found this to be effective in perturbing the solution from local minima and finding sparser and more interesting structures, e.g. as shown in Fig. 7. The complete optimization is listed in Algorithm 1. We found that \( n_{\text{opt}} = 3 \) rounds of optimization with MMA produces converged results. As shown in Fig. 8, \( n_{\text{opt}} > 3 \) produces little change. In all our computations we set \( c = 0.7 \) and \( b_{\beta} = 16 \) grid cells.

Using the extension field \( \beta \) and constraints on \( V \), we run the next optimization pass. We found this to be effective in perturbing the solution from local minima and finding sparser and more interesting structures, e.g. as shown in Fig. 7. The complete optimization is listed in Algorithm 1. We found that \( n_{\text{opt}} = 3 \) rounds of optimization with MMA produces converged results. As shown in Fig. 8, \( n_{\text{opt}} > 3 \) produces little change. In all our computations we set \( c = 0.7 \) and \( b_{\beta} = 16 \) grid cells.

Using the extension field \( \beta \) and constraints on \( V \), we run the next optimization pass. We found this to be effective in perturbing the solution from local minima and finding sparser and more interesting structures, e.g. as shown in Fig. 7. The complete optimization is listed in Algorithm 1. We found that \( n_{\text{opt}} = 3 \) rounds of optimization with MMA produces converged results. As shown in Fig. 8, \( n_{\text{opt}} > 3 \) produces little change. In all our computations we set \( c = 0.7 \) and \( b_{\beta} = 16 \) grid cells.

Using the extension field \( \beta \) and constraints on \( V \), we run the next optimization pass. We found this to be effective in perturbing the solution from local minima and finding sparser and more interesting structures, e.g. as shown in Fig. 7. The complete optimization is listed in Algorithm 1. We found that \( n_{\text{opt}} = 3 \) rounds of optimization with MMA produces converged results. As shown in Fig. 8, \( n_{\text{opt}} > 3 \) produces little change. In all our computations we set \( c = 0.7 \) and \( b_{\beta} = 16 \) grid cells.

Using the extension field \( \beta \) and constraints on \( V \), we run the next optimization pass. We found this to be effective in perturbing the solution from local minima and finding sparser and more interesting structures, e.g. as shown in Fig. 7. The complete optimization is listed in Algorithm 1. We found that \( n_{\text{opt}} = 3 \) rounds of optimization with MMA produces converged results. As shown in Fig. 8, \( n_{\text{opt}} > 3 \) produces little change. In all our computations we set \( c = 0.7 \) and \( b_{\beta} = 16 \) grid cells.
Algorithm 1 Incremental Shape Optimization

Input: User-defined geometry, user defined failure probability Θ

1: procedure SHAPE OPTIMIZATION
2:   ω₁ ← 1
3:   Optimize ω₁ using MMA to convergence
4:   V ← voxels in ω₁ with density larger than c
5: for i from 2 to n_opt do
6:     Compute β from V
7:     ω_i ← min(ω_{i-1} + β, 1)
8:     Constrain densities of V to 1
9:     Optimize ω_i using MMA to convergence
10:    V ← voxels in ω_i with density larger than c
11: end for
12:   V ← ∅
13:   ω_f ← ω_m
14:   Optimize ω_f using MMA to convergence
15: end procedure

4.4. Sheathing Post-Process

Many algorithms constrain the exterior of the object during the optimization [UMK17, LSD*16]. However, the final object then has an outer shell that is the thickness of the (quite coarse) voxel grid. However, as the shell is thickened, it can become the main reinforcement structure in the model [UMK17], which biases the optimization towards shell-like designs, and results in heavier objects. Instead, we only constrain the shell at force contact locations, allowing the optimizer to find a lighter reinforcement structure, and allowing surface regions which are unlikely to experience contact to be hollowed out. To preserve the surface geometry, we perform a post-process that adds a thin “sheath” of material, far below the resolution that we can simulate, along the original surface. We show in §5.4 that this sheathing has a minimal impact on the object’s performance.

5. Implementation and Results

Figure 9: Force contact locations for different examples. Surfaces marked red are possible contact locations.

5.1. Implementation Details

We initially voxelize the surface mesh according to the resolutions in Tbl. 4. We use Bullet [C*13] to obtain rigid body force samples. For all our examples, we use the following scenario: the shape falls 1 meter with a random initial orientation and small random angular velocity, and hits a flat plane. We record the three initial contact events when the shape hits the ground. We use n_s = 5000 rigid body simulations in all our examples and kept 90% of the total variance when performing PCA on the force samples. To avoid checkerboard patterns, we augment our cost function with the energy term of from Schumacher et al. [SBR*15]. We constrain the rigid body contact locations to ω_e = 1 to ensure that they do not change during optimization. As shown in Fig 9, these locations are usually very sparse and lie on the object’s convex hull. We use the following material parameters: Young’s modulus = 2.2 GPa, density = 1.037 g/cm³, and yield stress = 0.031 GPa. The object is scaled so that the maximal bounding box dimension is 15 cm.

We implemented our algorithm in C++. OpenMP multi-threading was used whenever possible, Eigen [GJ*10] is used for most matrix operations, Intel’s Paradiso was used to solve linear systems, and Armadillo [SC16] was used for the GMMs. All our results were run on a desktop with 192 GB of memory and a 2.4 GHz, 20-core Intel 6148.

5.2. Optimization Results

Fig. 10 shows that our method adds densities to locations of potential high stress. In particular, fragile locations such as the head of the beaver, the extremities of the camel, the handle and spout tip of the teapot, the beak corner (commissure) on the crow (Fig. 1), and

Figure 10: Left: The teapot, camel, beaver, and tiger surface meshes. Middle: Final optimized density field. Right: Final meshed results, rendered translucently to show internal structure.
the pelvis of the Armadillo (Fig. 15, top) are reinforced during optimization. Conversely, material is removed from low-stress regions. We do not constrain the surface, so interesting contact-dependent structures form, such as the removal of the beaver’s back, the tiger’s chest, and the divot between the camel’s humps.

![Figure 11: Left: Optimized result of a beaver with its tail. Collision with its back becomes very unlikely, so the region is entirely hollowed out. Right: With the tail removed, the back experiences collisions, so the material remains.](image)

To further explore the dependence of our algorithm on real-world contacts, we optimized a beaver model with and without its tail (Fig. 11). The optimizer aggressively subtracts material along the beaver’s back when collisions in that region become unlikely. Fig. 12 compares our results to Langlois et al. [LSD*16], and shows that our method consistently produces a superior (lighter) object. Tbl. 2 shows the survival probability remains similar, and we achieve up to 3.22× lighter objects. Fig. 13 shows the object weights and survival probabilities during optimization. Langlois et al. [LSD*16] stalls early, while our method makes steady progress.

![Figure 12: The Langlois et al. [LSD*16] algorithm vs. ours. Left: Original shape. Middle: Stalled result from Langlois et al. Right: Improved result using ours.](image)

5.4. Optimization Validation

We ran several comparisons to validate our sheathing post-process. First, we compared the results of the optimization with and without shell constraints (Tbl. 3). The shell constraint consistently produces heavier results, even through the survival probability is almost identical. We then added a thin sheath to the results without the shell constraint. The survival probability remains essentially the same (in the Armadillo case, it actually improves), and the object weight remains significantly below that found using the shell constraint.

![Table 3: Results with and without the shell constraint, and with the sheathing post-process from §4.4.](image)

5.3. Post-Processing

As is common in many algorithms, we optimize over continuous densities (Fig. 14, left). To obtain the final mesh (Fig. 14, middle), we use marching cubes on the density field [LC87] to obtain the $\omega = 0.5$ isocontour. Since we did not constrain the exterior shell of the object, we attach a thin shell of width $\frac{dx}{2}$ using the SDF of the original mesh, where $dx$ is length of one hexahedron (Fig. 14, right). We show in §5.4 that this thin shell has a negligible effect on the final object’s weight and survival probability.

![Table 2: The final object weight and survival probabilities of our method and Langlois et al. [LSD*16]. Our method consistently produces lighter objects with similar survival probabilities.](image)

Additionally, we visualized the von Mises stresses for several models in Fig. 15. The heavier shell-constrained models produce
larger rigid body impact forces, and therefore larger stresses. The sheath post-processed models have almost the same stress distribution as the original shell-unconstrained results, which indicate that the regions of likely impact have been effectively reinforced.

6. Physical Validation

We printed five copies of our Raven model with the sheath and five copies without. The results are shown in Fig. 2.

We then dropped each model from a height of 1.5 meters until a breakage occurred. Each drop used a random initial orientation, and the results are listed in Tbl. 5. We computed the expected survival probabilities from these breakage statistics, and found that they are close to the probability predicted by our simulation.

In Fig. 16 we show two breakage patterns from the sheathed and un-sheathed objects. Breaks occur around the beak or contact locations. In all five of our tests, as predicted by our simulation, the sheath did not come into contact with the ground.
Table 4: Timing breakdown (in seconds), volume reduction, and survival probability across different examples. The volume reduction is computed with final after the sheathing post-process.

<table>
<thead>
<tr>
<th>Model</th>
<th>Resolution</th>
<th># Iters</th>
<th>Sampling</th>
<th>GMMs</th>
<th>FEM</th>
<th>Grad</th>
<th>Total</th>
<th>Volume Reduction</th>
<th>Survival Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Armadillo</td>
<td>56 × 64 × 52</td>
<td>189</td>
<td>72.52</td>
<td>31.40</td>
<td>8.36</td>
<td>289.17</td>
<td>401.45</td>
<td>67.4%</td>
<td>0.706</td>
</tr>
<tr>
<td>Camel</td>
<td>20 × 64 × 52</td>
<td>142</td>
<td>70.14</td>
<td>20.00</td>
<td>3.71</td>
<td>140.34</td>
<td>234.19</td>
<td>64.8%</td>
<td>0.734</td>
</tr>
<tr>
<td>Beaver(tail)</td>
<td>56 × 28 × 64</td>
<td>162</td>
<td>60.55</td>
<td>22.56</td>
<td>6.28</td>
<td>200.84</td>
<td>290.23</td>
<td>70.4%</td>
<td>0.617</td>
</tr>
<tr>
<td>Raven</td>
<td>32 × 64 × 40</td>
<td>159</td>
<td>62.11</td>
<td>31.00</td>
<td>5.00</td>
<td>260.70</td>
<td>328.18</td>
<td>73.1%</td>
<td>0.624</td>
</tr>
<tr>
<td>Teapot</td>
<td>64 × 36 × 44</td>
<td>168</td>
<td>60.80</td>
<td>43.00</td>
<td>7.51</td>
<td>282.51</td>
<td>393.83</td>
<td>71.7%</td>
<td>0.737</td>
</tr>
<tr>
<td>Dog</td>
<td>40 × 64 × 60</td>
<td>103</td>
<td>74.69</td>
<td>32.44</td>
<td>11.8</td>
<td>341.81</td>
<td>460.71</td>
<td>77.4%</td>
<td>0.754</td>
</tr>
<tr>
<td>Tiger</td>
<td>32 × 40 × 64</td>
<td>134</td>
<td>67.36</td>
<td>31.04</td>
<td>4.93</td>
<td>170.77</td>
<td>274.10</td>
<td>67.1%</td>
<td>0.813</td>
</tr>
<tr>
<td>Penguin</td>
<td>36 × 48 × 40</td>
<td>97</td>
<td>21.74</td>
<td>42.11</td>
<td>11.5</td>
<td>180.51</td>
<td>255.85</td>
<td>71.3%</td>
<td>0.767</td>
</tr>
<tr>
<td>Rabbit</td>
<td>28 × 44 × 48</td>
<td>103</td>
<td>17.28</td>
<td>35.15</td>
<td>6.57</td>
<td>99.72</td>
<td>158.72</td>
<td>68.9%</td>
<td>0.662</td>
</tr>
<tr>
<td>Hand</td>
<td>40 × 32 × 24</td>
<td>98</td>
<td>30.72</td>
<td>28.28</td>
<td>3.80</td>
<td>57.66</td>
<td>120.46</td>
<td>68.8%</td>
<td>0.868</td>
</tr>
<tr>
<td>Molar</td>
<td>32 × 28 × 48</td>
<td>91</td>
<td>33.41</td>
<td>41.87</td>
<td>5.45</td>
<td>127.03</td>
<td>207.76</td>
<td>71.3%</td>
<td>0.761</td>
</tr>
<tr>
<td>Panda</td>
<td>28 × 32 × 36</td>
<td>95</td>
<td>17.22</td>
<td>35.12</td>
<td>4.54</td>
<td>90.30</td>
<td>147.18</td>
<td>68.2%</td>
<td>0.776</td>
</tr>
</tbody>
</table>

Table 5: Number of experimental drops of the sheathed and un-sheathed objects before breakage. Our predicted probability of remaining intact is respectively 0.632 and 0.624. The expected probability of remaining intact computed from the breakage statistics and assuming a binomial distribution is respectively 0.688 and 0.666.

<table>
<thead>
<tr>
<th>weight</th>
<th># of drops</th>
<th>weight</th>
<th># of drops</th>
</tr>
</thead>
<tbody>
<tr>
<td>44.02 g</td>
<td>4</td>
<td>41.68 g</td>
<td>5</td>
</tr>
<tr>
<td>45.00 g</td>
<td>4</td>
<td>42.62 g</td>
<td>3</td>
</tr>
<tr>
<td>44.84 g</td>
<td>3</td>
<td>42.8 g</td>
<td>3</td>
</tr>
<tr>
<td>44.44 g</td>
<td>3</td>
<td>41.77 g</td>
<td>2</td>
</tr>
<tr>
<td>44.89 g</td>
<td>2</td>
<td>41.26 g</td>
<td>2</td>
</tr>
</tbody>
</table>

7. Discussion and Future Work

We have described an asymptotically faster and more robust method for stochastic structural optimization. We reduced the previous quadric complexity to linear, which results in a two order of magnitude speed-up. By stabilizing the gradients and utilizing a constrained restart method, we achieve better convergence.

We hold the contact points fixed during the optimization, which enables us to use a finite difference method to compute the inertia gradients (i.e. the gradient of the rigid body simulator). However, this means that our method cannot handle topological changes along the surface, so it limited to examples where the external surface is prescribed. As a direction for future work, allowing the gradients to incorporate shape changes would broaden the possible application areas.

While the constrained restart method works in practice, it remains to be seen if the reinforcement structure can be identified and constrained within a single optimization pass to improve convergence. Even with our improvements, computing the probability gradient can still be a bottleneck due to its dense linear algebra operations, which limits the resolution of our method. One way to reduce this cost could be to use a sparse grid [LHZ*18].

References

by mapping a rigid body force $f'$ via a projection matrix $J$: $f' = Jf'$. Each rigid body force sample $f' \in \mathbb{R}^{3n}$ is of the form:

$$f' = [l_1 \ldots l_n] [\text{com}\ t\ \text{com}]$$

where $l_i$ is the contact forces sampled at the surface of the object, $n$ is the total number of possible contact positions, and $\text{com}, t$ are the inertial force and torque acting on the center of the mass.

The matrix $J$ is of the form:

$$J = [J^m J^{\text{com}} J^r]$$

where $J^m \in \mathbb{R}^{3n \times 3n}$ maps the surface contact points to volumetric element vertices, and $J^{\text{com}}, J^r \in \mathbb{R}^{3n \times 3}$ maps the inertial forces and torques acting on the center of mass to element vertices.

Given $n$ rigid body force samples, $L = [l_1 \ldots l^m]$, the reduced force basis $\bar{F}$ is the eigenvectors of the covariance matrix of the force samples, $JL$. Therefore, we have $JLL^T\bar{f} = FA\bar{f}$, where $A \in \mathbb{R}^{3n \times 3}$ is a diagonal matrix of eigenvalues of the covariance matrix $JLL^T$. From here we follow the procedure of [XSZB15]. We have:

$$\frac{\partial JLL^T\bar{f}}{\partial \omega} = \frac{\partial F}{\partial \omega} \frac{\partial F^T}{\partial \omega} + \frac{\partial A}{\partial \omega} \bar{f} + FA \frac{\partial F^T}{\partial \omega}$$

Multiplying Eqn. 25 with $\bar{F}^T$ from the left and $\bar{F}$ from the right yields:

$$\bar{F}^T \frac{\partial JLL^T\bar{f}}{\partial \omega} \bar{F} = \bar{F}^T \frac{\partial F}{\partial \omega} \bar{F} + \frac{\partial A}{\partial \omega} \bar{f} + FA \frac{\partial F^T}{\partial \omega}$$

Denote $B_e = \bar{F}^T \frac{\partial JLL^T}{\partial \omega} \bar{F}$. The matrix $W_e = \bar{F}^T \frac{\partial A}{\partial \omega}$ is antisymmetric [XSZB15], therefore $\frac{\partial A}{\partial \omega} = \text{diag}(B_e)$, so we have:

$$B_e - \text{diag}(B_e) = W_e A + AW_e^T$$

Exploiting the skew-symmetry of $W_e$ and setting $\lambda = \text{diag}(A)$, we can derive a simple update equation:

$$(\lambda_i - \lambda_j)W_{ij} = B_{ij}$$

where $B_{ij} = B_e - \text{diag}(B_e)$. We solve this equation for each $W_{ij}$. Finally we can compute the approximate gradients as $\frac{\partial F}{\partial \omega} = \bar{F} W_e C$.

The matrix $J$ has a closed form derivative and we compute the derivative of $LL^T$ using finite differences of the rigid body force samples, so $\frac{\partial JLL^T}{\partial \omega}$ is straightforward to evaluate.

**Appendix B:** Enhanced Evaluation for $W_e$.

To evaluate $W_e$, first we need to evaluate $B_e$, and then compute $W_e$ according to equations 27 and 28. Using equation 25 and expanding derivatives, we have:

$$B_e = B_e^1 + B_e^2 + (B_e^3)^T$$

$$B_e^1 = \bar{F}^T \frac{\partial JLL^T}{\partial \omega} \bar{F}$$

$$B_e^2 = \bar{F}^T \frac{\partial JLL^T}{\partial \omega} \bar{F}$$

First we consider the evaluation of $B_e^1$. Because only the $\frac{\partial F}{\partial \omega}$ factor is different from element to element, we can precompute $LL^T \bar{F} = \mathcal{O}(m + nr)$ for all the elements. The major complication is to compute $\frac{\partial F}{\partial \omega}$ efficiently for all elements.
From Appendix C, \( \frac{\partial J}{\partial \omega} \) is composed of three parts:

\[
\frac{\partial J}{\partial \omega} = \begin{bmatrix} 0 & J^T & J^T + J^\text{com} J^c \end{bmatrix}
\]

(30)

where \( J^T, J^T, J^T \in \mathbb{R}^{3n \times 3} \) is constant for all the elements, \( J^c, J^c, J^c \in \mathbb{R}^{3n \times 3} \) is different from element to element (but its sparsity is of \( O(c) \), where \( c \) is a constant), \( J^\text{com} \in \mathbb{R}^{3m \times 3} \) is a constant matrix, and \( J^c, J^c \in \mathbb{R}^{3 \times 3} \) differs from element to element (but it is a small matrix).

Computing \( F^T \frac{\partial J}{\partial \omega} \) naively for all elements poses a complexity of \( O(m^2r) \). However,

\[
F^T \frac{\partial J}{\partial \omega} = \begin{bmatrix} 0 & F^T J^T & F^T J^T + F^T J^\text{com} J^c \end{bmatrix}
\]

(31)

where \( F^T J^T, F^T J^T \) and \( X = F^T J^\text{com} \) can be precomputed for all elements in \( O(mr) \). The only terms which are needed per-element are \( F^T J^c, F^T J^c \) and \( XJ \). Because the three matrices \( J^c, J^c, J^c \) have a constant number of non-zero entries, this can be done for all elements in \( O(mr) \). Therefore, computing \( F^T \frac{\partial J}{\partial \omega} \) can be reduced from \( O(m^2r) \) to \( O(mr) \). The final matrix products \( F^T \frac{\partial J}{\partial \omega} \) and \( (LL^T) J \) take \( O(mr^2) \) for all elements. Therefore, \( B^c \) can be computed in \( O(mr^2) \).

Now consider the evaluation of \( B^c \). Finite differences are used to evaluate \( \frac{\partial (LL^T)}{\partial \omega} \).

\[
\frac{\partial (LL^T)}{\partial \omega} = \sum_{i=1}^{10} \frac{\partial (LL^T)}{\partial M_i} \frac{\partial M_i}{\partial \omega} = \sum_{i=1}^{10} \frac{\partial (LL^T)}{\partial M_i} \frac{\partial M_i}{\partial \omega}
\]

(32)

\( \frac{\partial (LL^T)}{\partial M_i} \) are 10 matrices which can be precomputed in \( O(n_n n_e) \) for all elements. The matrix product \( (F^T J) \frac{\partial (LL^T)}{\partial \omega} (J^T F) \) takes \( O(m(r_n^2 + r_n^2)) \) for all elements. This is the total complexity for \( B^c \).

This part can also be accelerated through further precomputation. Denote \( A_i = \frac{\partial (F^T F)}{\partial M_i}, H = F^T J \in \mathbb{R}^{(m+6)} \). We have:

\[
B^c = H \sum_{i=1}^{10} A_i \frac{\partial M_i}{\partial \omega} H^T = \sum_{i=1}^{10} HA_i H^T \frac{\partial M_i}{\partial \omega}
\]

(33)

Since only \( \frac{\partial M_i}{\partial \omega} \) changes between elements, we can precompute \( HA_i H^T \in \mathbb{R}^{(m+6)} \) before doing the summation. This reduces the complexity for evaluation of \( B^c \) to \( O(mr^2) \), and also reduces the total complexity for \( B^c \) to \( O(mr^2) \).

**Appendix C: Evaluation of Derivative \( \frac{\partial J}{\partial \omega} \)**

Here we explain how to compute the derivative \( \frac{\partial J}{\partial \omega} \). The general form of \( J \) is shown in equation 24. The matrix \( J^\text{com} \) maps a surface index to full volume index. Because the FEM mesh does not change during optimization, \( \frac{\partial \omega}{\partial \omega} = 0 \). We only need to consider last 6 columns of \( J^\text{com} \).

Before going into the details of \( J^\text{com} \) and \( J \), denote the total mass of the object as \( M \), and the total number of element nodes as \( n \). The mass for each node \( i \) is \( m_i \), the fill ratio for each element \( e \) is \( \omega_e \), and the mass of each element is \( d_e = \omega_e \rho \), where \( \rho \) is the density.

Because we use hexahedral elements in the FEM mesh, we compute the node mass as:

\[
m_i = \sum_{j \in N(i)} \frac{1}{8} d_i
\]

(34)

where \( N(i) \) denotes the set of element indices adjacent to node \( i \).

The matrix \( J^\text{com} \) maps the center of mass force to each element. It is a tiled diagonal matrix of \( 3 \times 3 \) of the following form:

\[
J^\text{com} = \frac{1}{M} \begin{bmatrix} m_1 & 0 & 0 \\
0 & m_1 & 0 \\
0 & 0 & m_1 \\
\end{bmatrix}
\]

(35)

To compute \( \frac{\partial m_i}{\partial \omega} \) for element \( e \), we need to evaluate:

\[
\frac{\partial m_i}{\partial \omega} = \frac{\partial M_i}{\partial \omega} \frac{\partial M}{\partial \omega} \frac{\partial m_i}{\partial M}
\]

(36)

The second term on the right hand side of equation 36 is non zero for all \( i = 1, \ldots, n \), because all element nodes contribute to the total mass \( M \).

\[
\frac{\partial M_i}{\partial \omega} = -\sum_{k \in N(e)} \frac{\partial m_k}{\partial \omega} \frac{m_k}{M^2}
\]

(37)

Because element \( e \) contributes \( \frac{1}{8} d_e \) to node \( m_k \), we have

\[
-\frac{1}{8} \sum_{k \in N(e)} \frac{\partial d_e}{\partial \omega} \frac{m_k}{M^2} = \frac{1}{8} \sum_{k \in N(e)} \frac{\rho}{M^2} m_k = -\rho \frac{m_k}{M^2}, \quad i = 1, \ldots, n
\]

(38)

Denote this part of the derivative as \( J^c \). It is a dense tiled matrix of the form:

\[
J^c = \frac{\rho}{M^2} \begin{bmatrix} m_1 & 0 & 0 \\
0 & m_1 & 0 \\
0 & 0 & m_1 \\
\end{bmatrix}
\]

(39)

where every \( i = 1 \ldots n \) is tiled with the \( 3 \times 3 \) diagonal matrix.

Now consider the first part of equation 36, which is non zero only for nodes \( i \) that are adjacent to element \( e \). We have:

\[
\frac{\partial m_i}{\partial \omega} = \frac{\partial m_i}{\partial \omega} \frac{1}{M} \frac{M}{\partial \omega} \frac{\rho}{M} = \frac{\rho}{8M}, \quad for \ k \in N(e)
\]

(40)

This part of the derivative is \( J^c \), a sparse matrix of the form:

\[
J^c = \begin{bmatrix} \rho & 0 & 0 \\
0 \frac{\rho}{M} & 0 \\
0 0 \frac{\rho}{M} \\
\end{bmatrix}
\]

(41)
where only node indices \( k \in N(e) \) are filled with the \( 3 \times 3 \) diagonal matrix. So in summary,
\[
\frac{\partial J^\text{com}}{\partial \omega_c} = J^c + J_e^c
\]

The matrix \( J^2 \) maps the center of mass torque to the elements. It is a tiled \( 3 \times 3 \) skew-symmetric matrix, where each matrix denotes a cross product:
\[
J^2 = \frac{1}{M} \begin{bmatrix}
\vdots & \vdots & \vdots \\
0 & -r_{z_i}m_i & -r_{y_i}m_i \\
-r_{z_i}m_i & 0 & r_{x_i}m_i \\
0 & r_{x_i}m_i & 0 \\
\vdots & \vdots & \vdots
\end{bmatrix}
\]

where \( r_i = p_i - q \) denotes the vector pointing from the center of mass \( (q) \) to the position of node \( i \) \( (p_i) \). Subscripts \( x, y, z \) denote the respective components of that vector. To compute its derivative, consider the \( z \) component:
\[
\frac{\partial (r_{z_i}m_i/M)}{\partial \omega_c} = \frac{\partial (p_{z_i} - q_z) m_i}{\partial \omega_c} + r_{z_i} \frac{\partial (m_i/M)}{\partial \omega_c}, \quad i = 1, \ldots, N
\]
The second term of 44 is computed with equations 36 and 40, and results in a constant part which is included in equation 49, as well as the following non-constant sparse term:
\[
J_e^2 = \frac{\rho}{8M} \begin{bmatrix}
\vdots & \vdots & \vdots \\
0 & r_{z_i} & -r_{y_i} \\
-r_{z_i} & 0 & r_{x_i} \\
0 & r_{x_i} & 0 \\
\vdots & \vdots & \vdots
\end{bmatrix}
\]

Similar to equation 41, only node indices \( k \in N(e) \) are filled. For the first term in equation 44, we have:
\[
\frac{\partial (p_{z_i} - q_z) m_i}{\partial \omega_c} = \frac{\partial (p_{z_i} - q_z) m_i}{\partial \omega_c}
\]

Since \( q_k = \sum_{k=1}^{N} p_k m_k / M \), denote \( s = \sum_{k=1}^{N} p_k m_k \), we have:
\[
-\frac{\partial (s/M) m_i}{\partial \omega_c} = -\frac{\partial s}{\partial \omega_c} m_i + \frac{q_s m_i}{M^2} \frac{\partial M}{\partial \omega_c}, \quad i = 1, \ldots, N
\]
The second part of this equation is equal to \( \rho \frac{q_{z_i}}{M^2} \) for all \( i \), so we have the total contribution to the constant part of derivative as:
\[
(q_i - r_{ze}) \frac{q_{m_i}}{M^2}, \quad i = 1, \ldots, N
\]
we denote this part as \( J^e \), which is constant for all elements.
\[
J^e = \frac{\rho}{M^2} \begin{bmatrix}
\vdots & \vdots & \vdots \\
0 & (q_i - r_{z_i})m_i & -(q_i - r_{y_i})m_i \\
-(q_i - r_{z_i})m_i & 0 & (q_i - r_{x_i})m_i \\
(q_i - r_{y_i})m_i & -(q_i - r_{x_i})m_i & 0 \\
\vdots & \vdots & \vdots
\end{bmatrix}
\]

For the first term of equation 47:
\[
\frac{\partial \mathbf{m}_i}{\partial \omega_c} \frac{\mathbf{m}_i}{M^2} = -\sum_{k \in N(e)} \mathbf{p}_{ek} \frac{\partial \mathbf{m}_i}{\partial \omega_c} M^2 = -\sum_{k \in N(e)} \frac{1}{8} \mathbf{p}_{ek} \rho \frac{m_i}{M^2}, \quad i = 1, \ldots, N
\]
The summation of node position \( \sum_{k \in N(e)} \frac{1}{8} \mathbf{p}_{ek} \) is the \( z \) component of the position of element \( e \). Using the \( z \) component of the centroid of element \( e \) \( (t_{ze}) \), equation 50 can be computed as:
\[
\frac{\partial \mathbf{J}}{\partial \omega_c} = \left[ \begin{array}{ccc}
0 & J^c & J_e^c \\
0 & J^c & J_e^c \\
\end{array} \right] + \left[ \begin{array}{ccc}
\mathbf{J}^e & \mathbf{J}^e & \mathbf{J}^e
\end{array} \right] = \mathbf{J}^\text{com}
\]

**Appendix D: Optimization with different SIMP parameters**

SIMP penalizes intermediate densities in structural optimization by exponentiating them when computing the stiffness matrix. This makes the intermediate densities uneconomical in the optimization, attempting to force them to go to either zero or one. The exponent is a parameter, where increasing it will further penalize the intermediate densities, but render the optimization more difficult.

In figure 5, we use SIMP with an exponent of 5. We experimented with different choices and found it does not improve the results. As shown in figure 17, small SIMP exponents (2 or 3) lead to an unacceptable number of intermediate densities. Increasing the exponent above 10 often stall the optimization at the very beginning. We need a different approach to overcome the local minima.