Proceedings of Artificial Intelligence and Machine Learning for Engineering Design
December 14, 2022, Cambridge, USA

MIT 2.S997 Final Project

TORSION SPRING TOPOLOGY OPTIMIZATION FOR EXOSKELETON ACTUATOR

Booker Schelhaas
Department of Mechanical Engineering
Massachusetts Institute of Technology
Email: booker@mit.edu

Samuel Ubellacker
Department of Mechanical Engineering
Massachusetts Institute of Technology
Email: subella@mit.edu

ABSTRACT

Torsional springs are often integrated to joints of exoskeletons to counterbalance high-load maneuvers, such as standing. Classical design methods for such springs consist of a time-consuming iterative approach requiring a trained expert. Instead, we offer an entirely automated approach using a Genetic algorithm and Finite Element Analysis. Our approach can take in any number of design considerations (spring constant, peak stress) and will generate a set of dominant solutions which form a Pareto front. In simulation, we have shown our approach is capable of generating designs with a spring constant within $1 \times 10^{-5} \frac{\mathrm{N}}{\mathrm{m}}$ *of a reference value, while greatly attenuating peak stresses.*

INTRODUCTION

Series elastic actuators (SEA) are becoming a widespread norm for human exoskeletons. These are made by placing a torsional spring in series between the motor and the attachment to the body. This provides precise torque control from measuring the deflection of the spring as well as some built in compliance, making it safer for interacting with the human body. While some optimization work has been done in the past, most of these springs have been designed by hand, with little optimization for performance or lifetime.

This process is painstaking and requires a lot of time. The largest problem with these springs in practices is short lifespans due to high stress concentrations. Most designs can't withstand large deformations without breaking and end up being less effective. It is important for them to be robust and reliable, especially when attached to a human body.

RELATED WORK

Most series elastic actuators have been designed with a manual search over a parametrization [1]. They involve deciding on a couple topologies and running FEA on all of them with changes in the parametrized values (Fig.). This process is mostly manual and time-consuming. Some topology optimization has been performed, however the design space was limited again to already existing topologies and having their parameters as the values to be optimized with FEA as the evaluator [2] [3]. There is yet to be a parametrization that allows more flexibility and options in the topology.

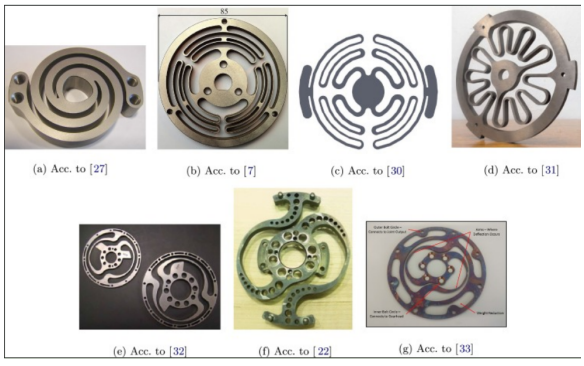

FIGURE 1. CUSTOM TORSIONAL SPRINGS MADE THROUGH PARAMETER SEARCH

OVERVIEW

Our automatated pipeline consists of three stages: mesh generation, mesh evaluation, and mesh optimization. The mesh generation stage will generate a mesh given declarations of the free variables which parameterize the mesh. These meshes are then evaluated by an FEM stage which will return the spring constant reference error and peak stress. These metrics and corresponding mesh parameters are used to inform NSGA-II, which selects a new set of mesh parameters to feed back to the mesh generation stage. This loop continues until a set number of iterations has passed, with the mesh designs improving each iteration.

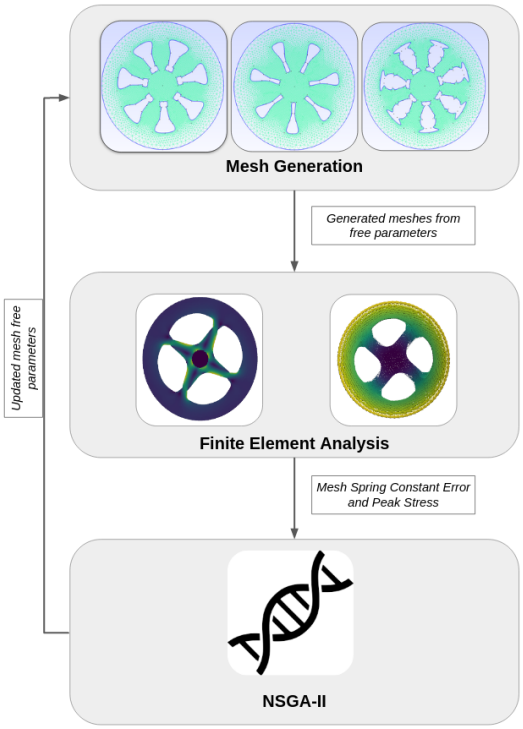

FIGURE 2. AUTOMATED PIPELINE OPTIMIZATION LOOP

MESH GENERATION

There exists an implicit tradeoff between expressiveness and computational feasibility when parameterizing the mesh. If the parameterization is too restrictive, then the search space of the GA is reduced, which may lead to less optimal solutions. On the contrary, if the parameterization is too loose, much more computation is needed to explore the space. We seek to find a parameterization flexible enough to achieve reasonably optimal solutions, and restrictive enough to guide the GA's decisions.

In particular, we heavily rely on radial symmetry. Intuitively, symmetric designs lead to an even distribution of stress which aids in mitigating the peak stress. Additionally, the spring constant should remain the same irrespective of the direction of applied torque, which is easily achieved by a symmetric design.

Our resultant design appears similar to a spoke wheel design (Fig. 3). The number of spokes and the width of each spoke are free design parameters. The shape of each spoke is deduced by cutting out sections parameterized by a B-Spline. The shape of this spline is dependent on the number and location of control points, which are free design parameters. Parameterizing with a B-Spline indeed has a strong bias on the configuration of the spring, but also results in spokes which are smooth and differentiable, which greatly reduces peak stress concentrations.

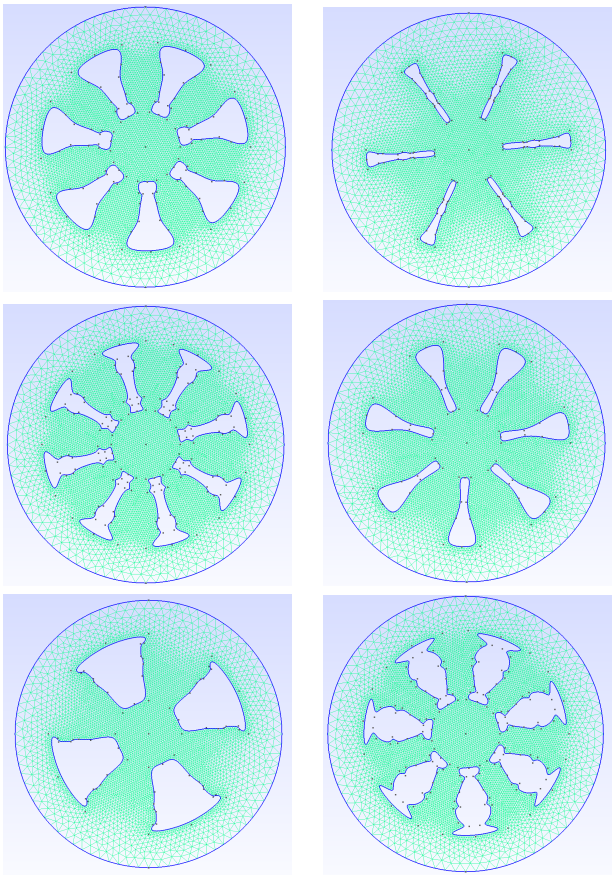

FIGURE 3. RANDOMLY GENERATED MESH DESIGNS

SOFTWARE We use Gmsh [4], an open source finite element generator to create our meshes. We have constructed an automated pipeline in Python that can rapidly generate meshes when given the declarations of each free parameter as input. On average, it takes 0.29 seconds to generate a single mesh, computed from 100 randomly generated meshes.

MESH PARAMETERIZATION Each mesh is defined by the following set of parameters:

- r_1 The radius of the inner circle.
- r_2 The inner radius of the outer rim.
- r_3 The outer radius of the outer rim.
- n_s The number of spokes.
- n_c The number of spline control points.
- C A single set of control points' positions.
- w The spoke size ratio, which is the ratio of free space to filled space.

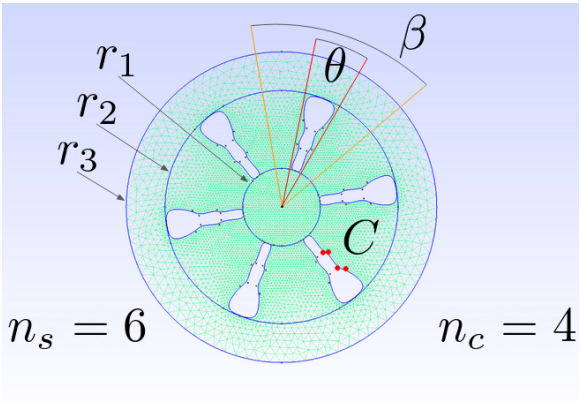

FIGURE 4. A VISUALIZATION OF THE PARAMETERIZATION VARIABLES.

The parameterization process is then applied:

1. The surface is partitioned into n_s even slices, with each slice having an angular distance $\beta = \frac{2\pi}{n_s}$.
2. From the spoke size ratio, we compute the angular distance $\theta = w\beta$. This slice is positioned at the center of the larger partition.
3. A control point c_i is defined by polar coordinate (r_i, θ_i) and belongs to the set of control points $C \in R^{n_c}$. The position of c is determined by the GA and is bounded by $r_1 < r_i < r_2$, $\theta_i \in \theta$.
4. Each control point is placed and then mirrored to enforce a symmetrical cutout.
5. The control points are then connected by a curve either parameterized by lines, spline, or a B-spline, as discussed in the next section.
6. Each cutout is mirrored n_s times radially.
7. The surface is automatically meshed using Gmsh. A distance and thresholding function is used to increase the resolution of the mesh in the proximity of each cutout.

CUTOUT PARAMETERIZATION Because the cutout parameterization greatly effects the resulting design, it deserves special attention. In particular we explored three parameterization types: lines, spline, and B-spline.

Lines. In this parameterization, each control point is simply connected to each other using straight line segments. This parameterization is highly expressive, as $n_c \to \infty$ the cutout can form any imaginable shape. However, line segments are not smooth and will lead to high stress concentrations at junction points, especially when n_c is low.

Spline. Instead of connecting each control point with line segments, a single spline is constrained to pass through each point. This leads to a single smooth curve which reduces peak stress concentrations to a degree and is also highly expressive. However, sharp edges can still form when the distance between consecutive control points is large.

B-Spline. The B-Spline representation treats control points as basis functions instead of constraints. In practice, this leads to control points having a 'gravitational' affect which influences the curve's shape locally. This leads to a smoother and more regularized curve which is harder to explicitly shape.

Lines and splines often create designs with sharp corners and high derivatives (Fig. 5). Intuitively, designs such as these lead to high peak stress concentrations around areas where the curve is rapidly changing. While the B-Spline has less expression in our particular domain, it regularizes the curve which induces smoother curves. We assert that because the optimal solutions of the problem will consist of smooth curves, the added flexibility offered by lines and spline can be safely disregarded. We therefore choose a B-Spline as our primary method of parameterizing the cutouts.

Rarely, the B-Spline parameterization will generate curves which contain self-loops(Fig). It is difficult to prevent this from happening, and Gmsh will loop indefinitely when this occurs. We modified the source code for Gmsh to throw an error when self-loops are detected, so we can handle the exception in code. From the perspective of the GA, self-loops are infeasible designs which should be avoided.

Finite Element Analysis

To simulate applying a torque on the mesh we use SFEPY, a python library built for high customization of FEA simulations [5]. To achieve the desired simulation, we used a linear elasticity integral, while applying a known displacement of N degrees to the outermost edge of the disk. We then calculated the resultant torque by summing the moments experienced around the inner fixed circle (Fig 7).

To find the spring constant, we ran the simulation various times for different rotations and plotted the deformation vs the resultant moment experienced (fig X). From this we found that the actual plot has quadratic properties (Fig 8), which is similar

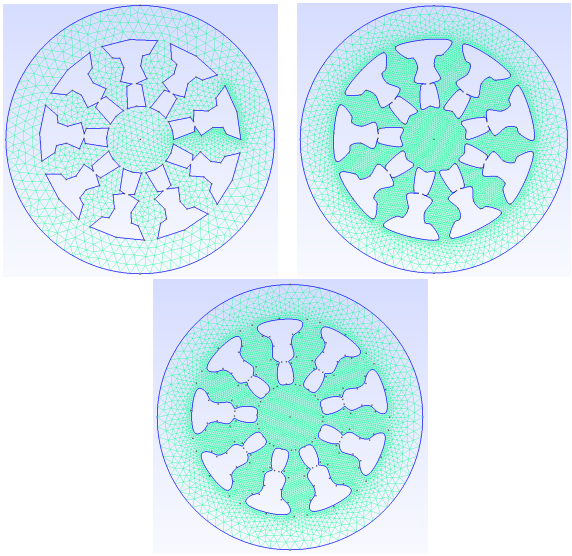

FIGURE 5. VISUALIZATION OF CUTOUT PARAMETERIZATIONS: LINES (TOP LEFT), SPLINE (TOP RIGHT), AND B-SPLINE (BOTTOM)

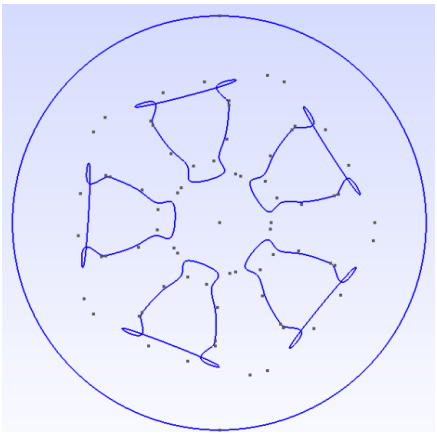

FIGURE 6. SELF-LOOPS IN B-SPLINE PARAMETERIZATION.

to what has been found in previous works. For running our algorithm, this appeared to be too computationally expensive, so we decided to go with a linear approximation instead (Fig 9). This allowed us to only do one simulation per spring, along with a 0 point, and calculate the constant by finding the slope of the line between the two. The plot of the von mises stress can be seen in Fig 10.

GENETIC ALGORITHM

We are presented with a multi-objective optimization problem which is highly non-convex. This form of problem cannot be solved with traditional gradient based optimization techniques

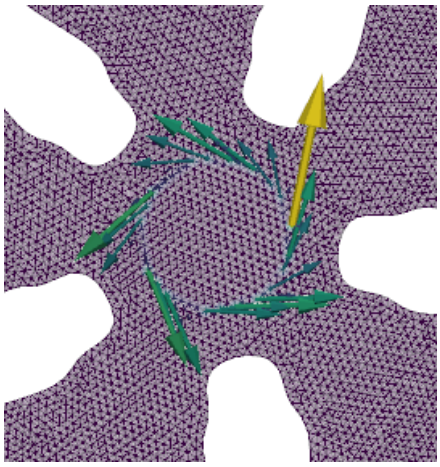

FIGURE 7. MOMENTS EXPERIENCED IN SIMULATION.

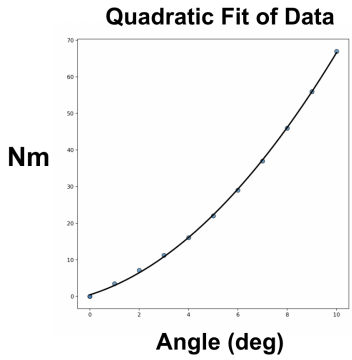

FIGURE 8. QUADRATIC FIT OF FEA SIMULATION DATA.

due to the prevalence of local minima. We instead opt for an evolutionary-based genetic algorithm approach. In particular, we use NSGA-II (Non-dominated sorting genetic algorithm).

NSGA-II mimics a natural selection type approach where mutations and mixing of strong candidates from one generation form the parents of the next generation. More specifically, NSGA-II generates offerings using crossover and mutation methods and the best offspring are selected using a non-dominated sorting and crowding distance comparison.

In our problem, we begin with a initial population of randomly generated meshes. Each mesh is evaluated through the FEA to determine their fitness. The NSGA-II algorithm is then applied to this population to create the next generation of offspring. This cycle is repeated until the required number of generations is met.

We use Pymoo [6] to implement our NSGA-II algorithm.

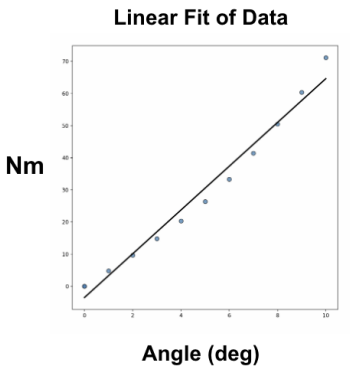

FIGURE 9. LINEAR FIT OF FEA SIMULATION DATA.

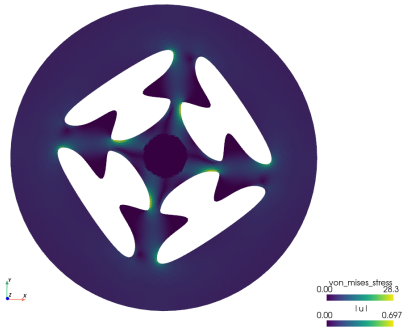

FIGURE 10. VON MISES STRESS OF A SPRING MESH.

RESULTS

We ran NSGA-II with an intial population size of 75 and a total of 500 generations. All other parameters were set to Pymoo's default values. In total, the simulation took 10 hours with a GTX 1080 Ti GPU, 64gb ram, and i9-7920X CPU.

To reduce the computational cost of the problem, we held some free parameters in our mesh generation fixed. In particular, we set the number of spokes to 4 and the number of control points to 5. The GA can modify the positions of the 5 control points and the width of the cutouts.

The resulting pareto front (Fig. 11) shows an interesting phenomena. The pareto front is comprised of 3 disjoint clusters. Indeed, the meshes belonging to a particular cluster all share the same archetype, with only minute differences in node positions or spoke width. Between clusters, the differences are more perceptible. This suggests our GA has experience a sort of natural selection where three designs have dominated all others. This could be due to several factors: too small of initial population, too little generation, or insufficient hyperparameters such as mutation rate. It could also likely be an artifact of FEM simulation, which can sometimes produce a drastically different output for a small perturbation of input.

The meshes with the lowest spring constant error and lowest

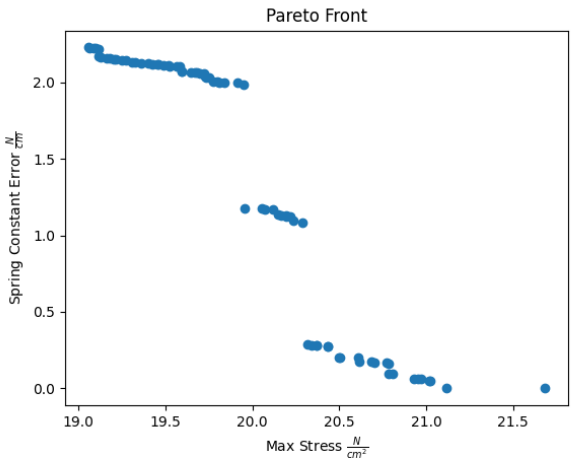

FIGURE 11. PARETO FRONT AFTER 500 GENERATIONS.

peak stress are shown in Fig. 12 and Fig. 13. We also display a mesh from the middle cluster which offers a compromise between the two metrics (Fig. 14).

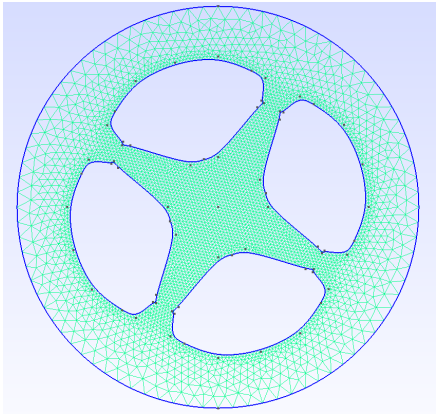

FIGURE 12. DESIGN WITH SMALLEST SPRING CONSTANT ERROR.

CONCLUSION

In this paper we designed a novel way of parametrizing a custom torsion spring for a series elastic actuator, and used a GA to optimize the topology in a multi-objective optimization problem. We found that our algorithm was able to optimize for both spring constant error and peak Von Mises stress resulting from a simplified FEA simulation.

We think that future work can still be done in order to allow the GA more freedom in how it can create spokes, with the goal

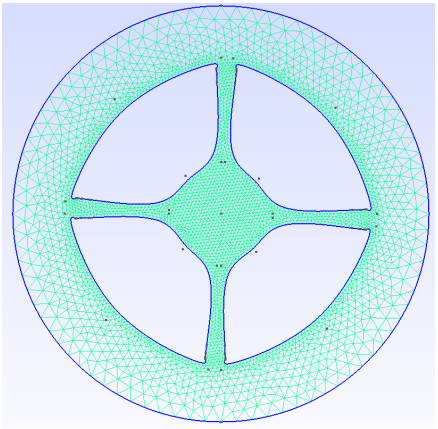

FIGURE 13. DESIGN WITH SMALLEST PEAK VON MISES STRESS.

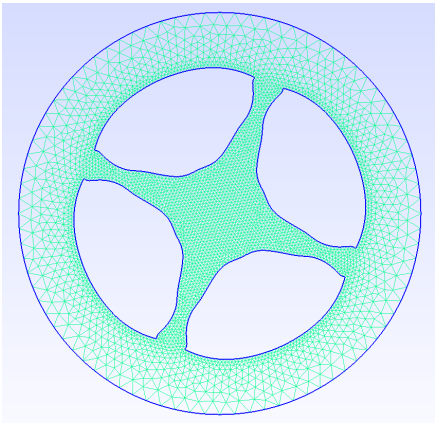

FIGURE 14. DESIGN COMPROMISING BETWEEN SPRING CONSTANT ERROR AND PEAK STRESS.

of generating a wider variety of outputs and a more distributed Pareto Front. Furthermore, we believe it would be valuable to compare the outputs from this setup to existing spring topologies.

ACKNOWLEDGMENT

Thank you to Professor Faez Ahmed and Lyle Regenwetter for the support and guiding us in the process.

REFERENCES

[1] Zhou, M., and Wang, S., 2021. "A method to determine the topology of custom torsional elastic element for the lightweight rotary series elastic actuator". *Journal of Physics*.

[2] Carpino, G., and Accoto, D., 2012. "A novel compact torsional spring for series elastic actuators for assistive wearable robots". *Journal of Mechanical Engineering Science*.

[3] Yildirim, M., and Sendur, P., 2021. "Design and development of a durable series elastic actuator with an optimized spring topology". *Journal of Mechanical Engineering Science,* ***235***.

[4] Geuzaine, Christophe and Remacle, Jean-Francois. Gmsh.

[5] Cimrman, Robert and Lukeš, Vladimír. Sfepy.

[6] Blank, J., and Deb, K., 2020. "pymoo: Multi-objective optimization in python". *IEEE Access,* ***8***, pp. 89497–89509.