CIFellows 2020-2021

Mehran Mirramezani

Computing Innovation Fellows

Princeton University, Prof. Adams' Lab

A rapid and automated approach to the design of multistable metastructures

Introduction

- Multistability is an important aspect of mechanical metamaterials designs that allows reconfigurable locally stable systems.
- In motion control, multistability enables large-magnitude movements in a pre-designed way by switching between stable configurations. This can reduce the number of parts, weight, and assembly time of actuators while increasing precision and reliability in many applications e.g., soft robotics.
- Current approaches to multistable design involves deep human expertise, experimental trial-and-error, and costly numerical simulations of continuum mechanics (finite element methods), leading to slow iteration times and suboptimal designs.
- Here, we developed an automated optimization framework that enables the rapid design of a structure with a desired pattern of multistability, leveraging the native differentiability and GPU acceleration of JAX.

Overview

I. input data

- Desired pattern of multistability e.g., a 2D curve of mechanical strain energies (SE), the local minima of which determines stable locations
- Geometric parameters (GPs) with constraints

II. solver

- Construct and discretize geometric models for FEM analysis
- Minimize nonlinear energy function to obtain mechanical deformation
- Compute SE values from mechanical deformation
- Optimize geometric parameters to reduce $||SE-SE_t||_2$

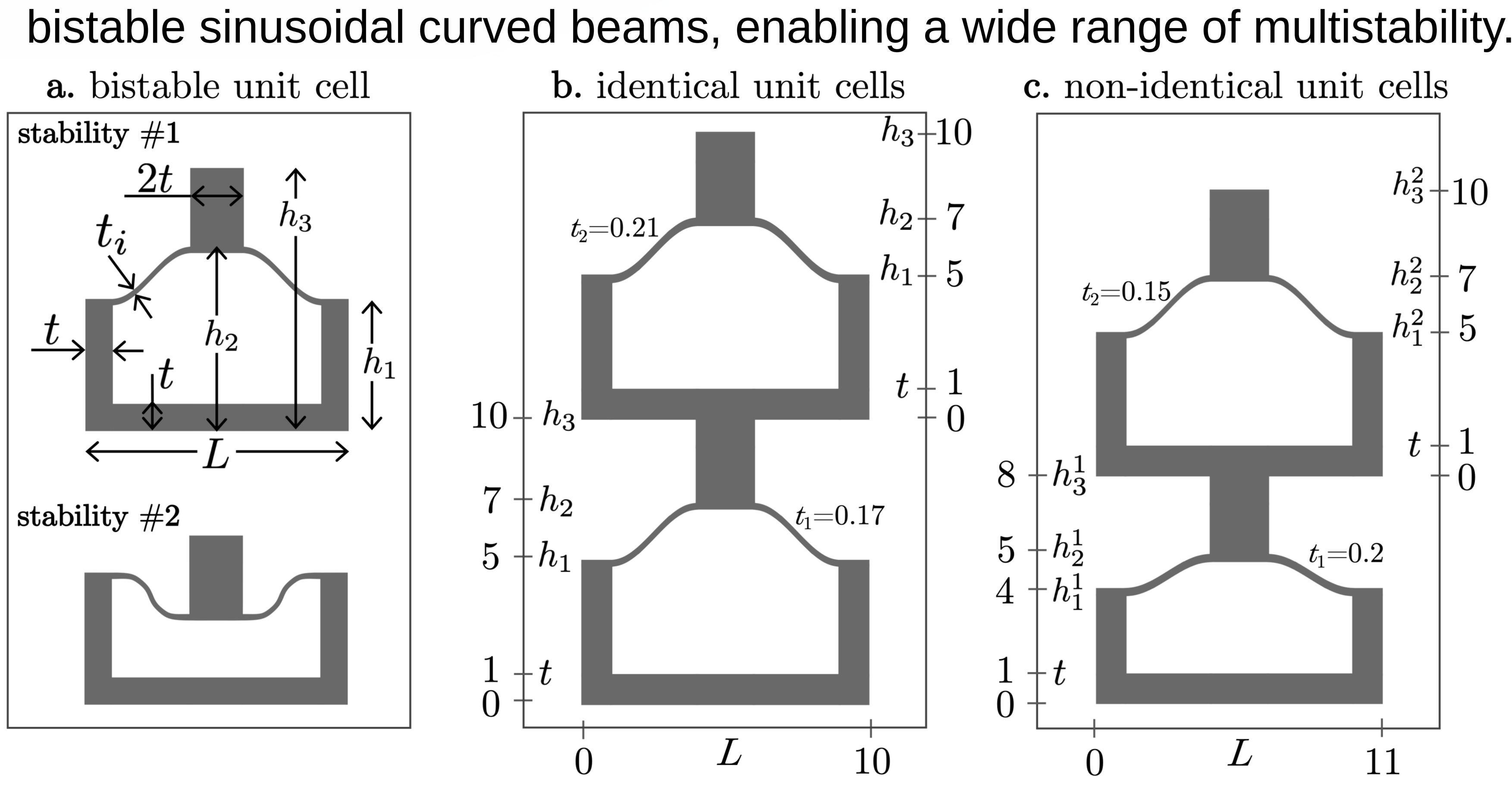
III. output data

- Final optimal design and its mechanical deformation under loading

Methods

a) Structural geometry of metastructures

- Multistable metamaterials consists of multiple identical or non-identical bistable sinusoidal curved beams, enabling a wide range of multistability.

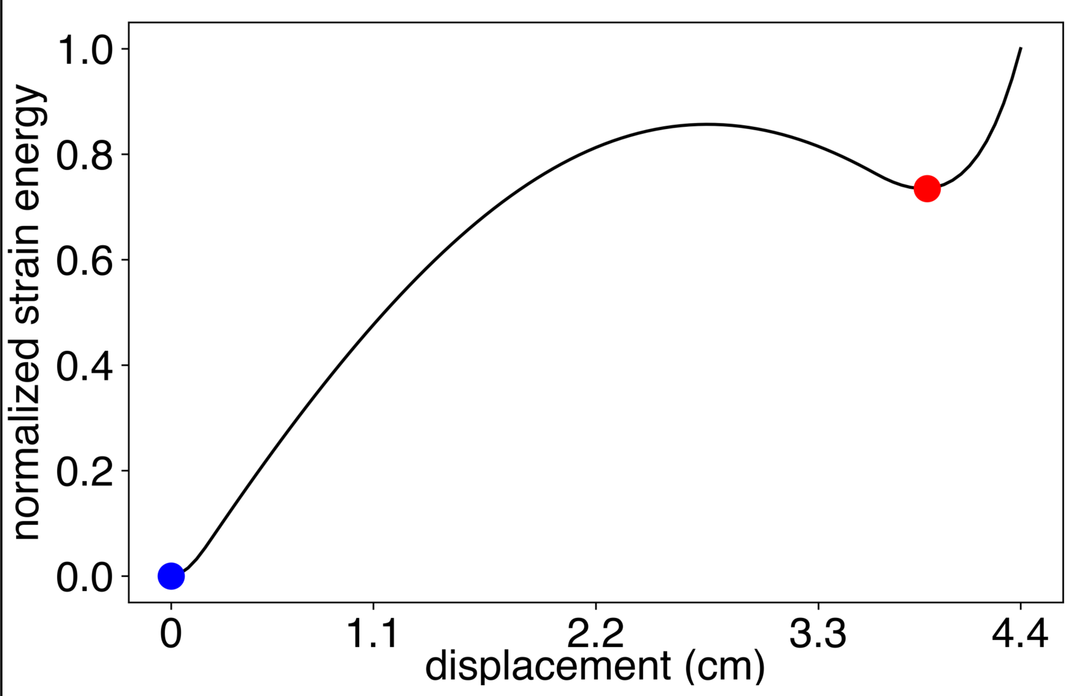


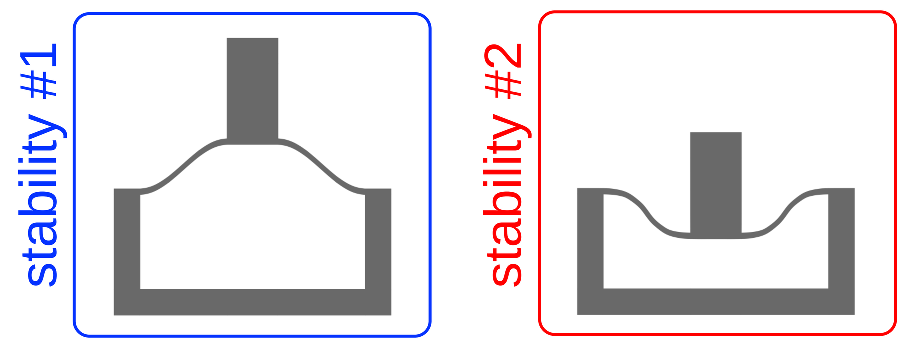
b) JAX implementation

```
def construct_simulation(geom_params,….)
  def construct_geomtry(geom_params,….)  # FEM analysis
      - construct/discretize geometric domain
      - return nonlinear energy density function W (u,x,X,…)
  @jax.jit
  def simulate(geom_params)
      - minimize W to obtain mechanical deformation "u"
      - return strain energy SE(u) values
  return simulate
def loss_fn(geom_params)
    SE = simulate(geom_params)
return ||SE-SE_t||_2
grad_loss = jit(grad(loss_fn(geom_params)))
geom_params = geom_params – lr × grad_loss
```

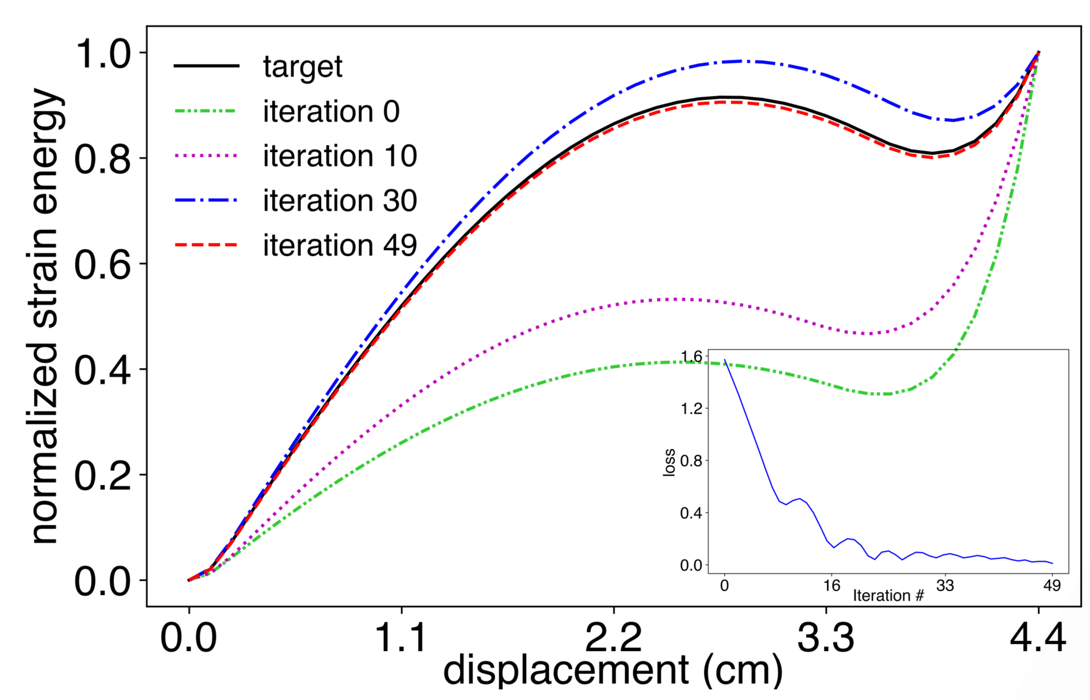
Results

a) Bi-stability analysis of a unit cell



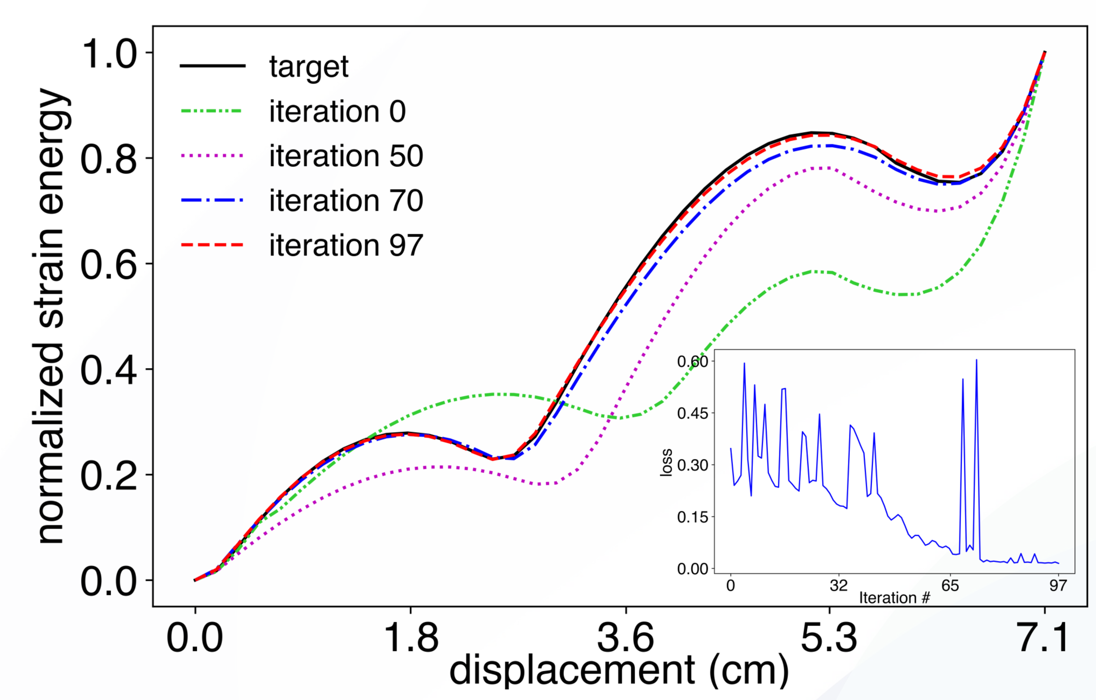


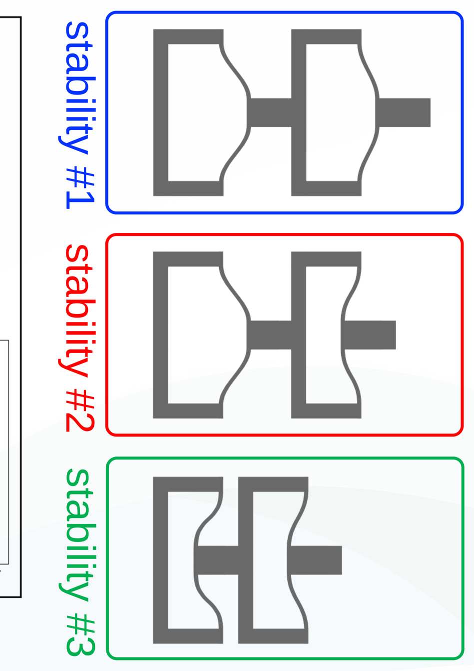
b) Optimize GPs to match a bi-stability



	L	t	h_1	h_2	h_3	t_1
Itr. #0	10.24	0.88	5.28	7.21	9.63	0.19
Itr. #10	10.15	0.97	5.29	7.19	9.72	0.27
Itr. #30	10.19	0.93	5.15	7.33	9.71	0.28
Itr. #49	10.18	0.94	5.18	7.30	9.73	0.28

c) Optimize GPs to match a multistability with constrains





	L	t	h_1^1	h_1^2	h_2^1	h_2^2	h_3^1	h_3^2	t_1	t_2
Itr. #0	10.00	1.00	5.00	5.00	6.95	6.24	11.00	11.00	0.19	0.21
Itr. #50	10.00	0.96	5.21	4.74	6.75	6.51	10.91	11.04	0.16	0.22
Itr. #70	10.00	0.94	5.30	4.61	6.65	6.63	10.91	11.02	0.17	0.20
Itr. #97	10.00	0.93	5.32	4.57	6.63	6.67	10.90	11.06	0.17	0.20

CRA Computing Research Association
CCC Computing Community Consortium Catalyst
NSF