Daikin AC Pipe Shape Optimization Automation

Final Presentation

Authors:

Anastasiia Rudenko[1]
Davis Funk[2]
Itsuki Imai[3]
Shota Nakagawa[3]

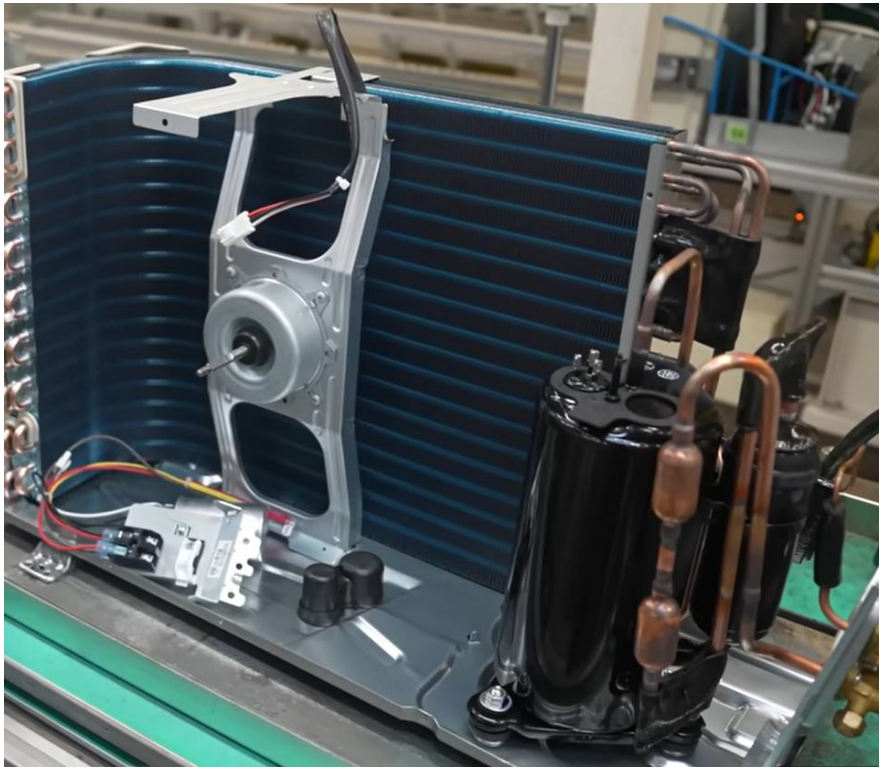
Mentors:

Yuji Ando+
Satoru Takanezawa*
Keiji Tomioka*
Yuki Eda*
Ryota Shiba*

[1] University of Pittsburgh
[2] University of California, Los Angeles
[3] Keio University
+ Tohoku University
* Daikin Mentor

Outline

① Introduction
② Problem Statement
③ Embedding Map Approaches
④ Mathematical Formulation
⑤ Numerical Simulations
⑥ Route Refinement Algorithm
⑦ Conclusions and Future Work
⑧ Q&A

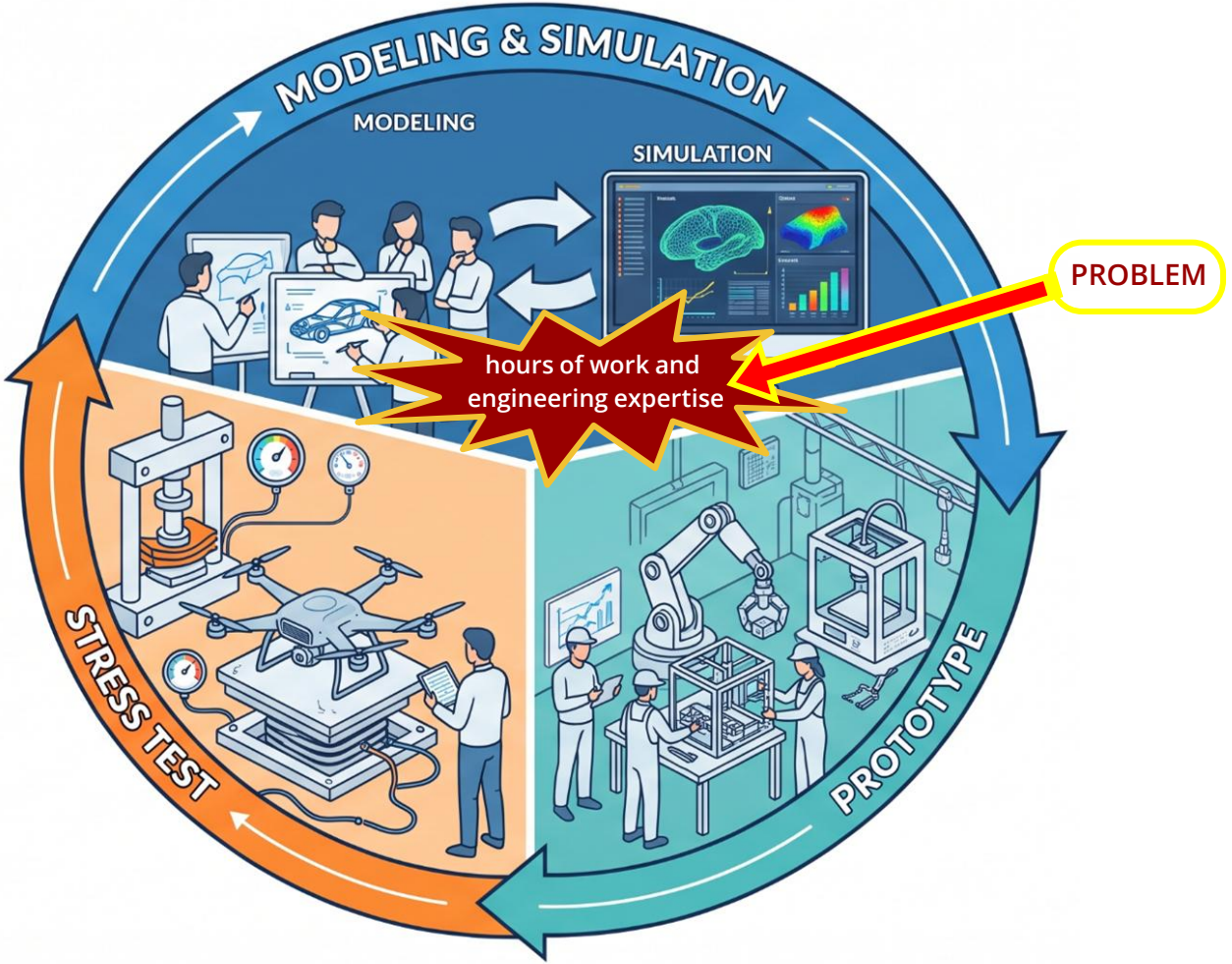


1. **Introduction**
2. Problem Statement
3. Embedding Map Approaches
4. Mathematical Formulation
5. Numerical Simulations
6. Route Refinement Algorithm
7. Conclusions and Future Work
8. Q&A

- current problem
- previous work

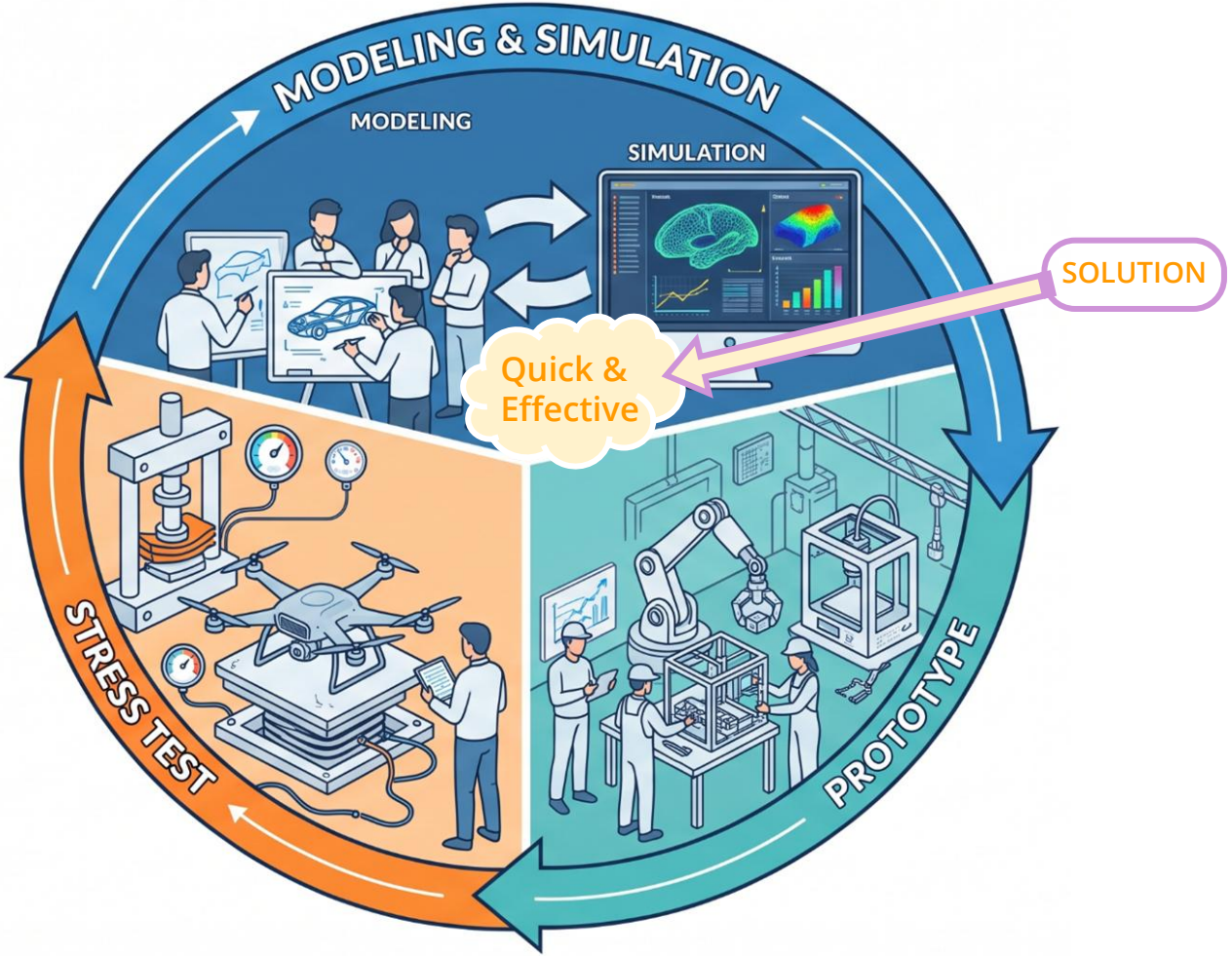
Introduction

Current pipe layout development at Daikin:



Introduction

Our contribution:



Introduction

What is necessary?

Manual

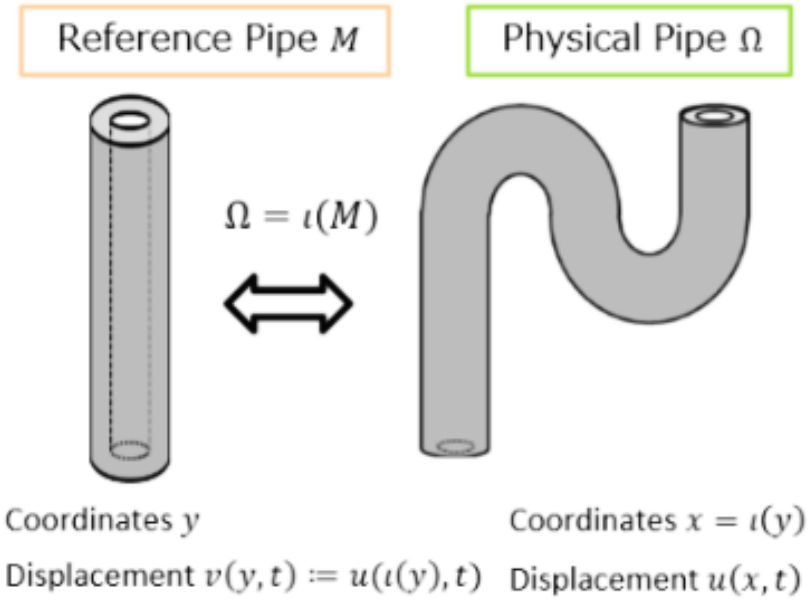
Automatic

- Mathematical model
- Numerical simulation

Introduction

Prior research results:

$$
\begin{aligned}
v(y,0) &= 0 && y \in \overline{M} \\
v(y,t) &= f(y,t) && (y,t) \in D_1 \times \mathbb{R}, \\
[V]n &= 0 && (y,t) \in D_2 \times \mathbb{R}, \\
\rho \frac{\partial^2 v}{\partial t^2} &= \nabla \cdot {}^T[V] && (y,t) \in M \times \mathbb{R}, \\
\iota(y) &= g(y) && y \in D_1, \\
[G]n &= 0 && (y,t) \in D_2 \times \mathbb{R}, \\
[G]\nabla(\mathcal{M}[v,\iota])^2 &= 0 && (y,t) \in M \times \mathbb{R}.
\end{aligned}
$$



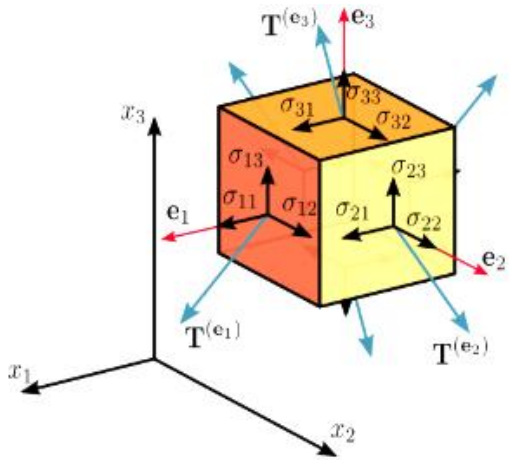
$$
\begin{aligned}
[V] &:= \mu\left[\frac{\partial v}{\partial y}\right] + \mu\left[\frac{\partial \iota}{\partial y}\right]^T\left[\frac{\partial v}{\partial y}\right]\left[\frac{\partial \iota}{\partial y}\right] + \lambda \mathrm{tr}\left(\left[\frac{\partial v}{\partial y}\right]^T\left[\frac{\partial \iota}{\partial y}\right]\right)\left[\frac{\partial \iota}{\partial y}\right], \\
[G] &:= \left(3\left[\frac{\partial \iota}{\partial y}\right]^T\left[\frac{\partial v}{\partial y}\right] + 3\left[\frac{\partial v}{\partial y}\right]^T\left[\frac{\partial \iota}{\partial y}\right] - 2\mathrm{tr}\left(\left[\frac{\partial v}{\partial y}\right]^T\left[\frac{\partial \iota}{\partial y}\right]\right)[I]\right)\left[\frac{\partial v}{\partial y}\right].
\end{aligned}
$$

y : parameter along pipe.
v : displacement.
f, g : given functions.
ρ : density.
ι : the curvature.
M : von Mises stress.

* the figure was provided by Daikin

The Von-Mises stress and the KS Function

- The **Von-Mises Stress** is an effective **maximum stress** in a structure, taken from the full stress tensor and its component stresses.
- It is widely used to determine **failure criteria** in structures.



$$\sigma_{VM} = \sqrt{\frac{1}{2}\left[(\sigma_{11}-\sigma_{22})^2 + (\sigma_{22}-\sigma_{33})^2 + (\sigma_{33}-\sigma_{11})^2 + 6(\sigma_{12}^2+\sigma_{23}^2+\sigma_{31}^2)\right]}$$

The Von-Mises stress and the KS Function

- Since the Von-Mises stress will vary according to both space and time in our problem, this function is **non-smooth** (non-continuous derivatives).
- As a result, we introduce the **KS function**, which smoothly approximates the maximums of the original VM stress, **turning it into a smooth**, differentiable function to be maximized.

$$\tilde{M}_a[v,\iota] = \frac{1}{a}\log\left(\frac{1}{\mathrm{Vol}(M\times[t_0,t_1])}\int_{t_0}^{t_1}\int_M \exp\left(a\sigma_{VM}^2\right)dydt\right)$$

1 Introduction
2 **Problem Statement**
3 Embedding Map Approaches
4 Mathematical Formulation
5 Numerical Simulations
6 Route Refinement Algorithm
7 Conclusions and Future Work
8 Q&A

- our goals

Problem Statement

The Goal

Build upon the prior research done by Daikin.

- Create a **prototype algorithm** that can:
  - simulate the stresses in a 3D pipe structure
  - **optimize** the shape of this pipe to **minimize stresses**.

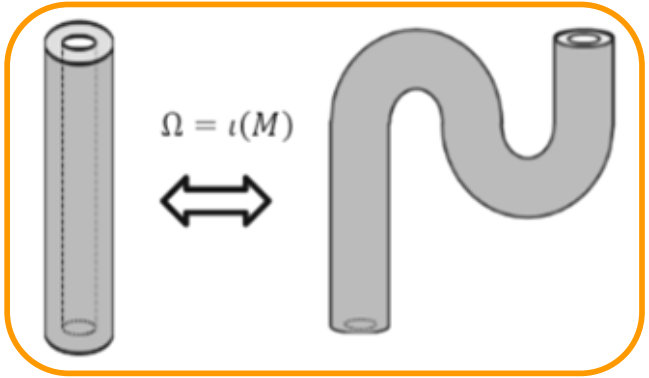
The Steps

- Primary goal: **prototype**.
  - Simplified approach was used.
  - Reason: time constraint.
- After satisfactory prototype:
  - Expand on the mathematical background previously derived by Daikin.
  - Focus on removing non-physical assumptions (det F = 1)

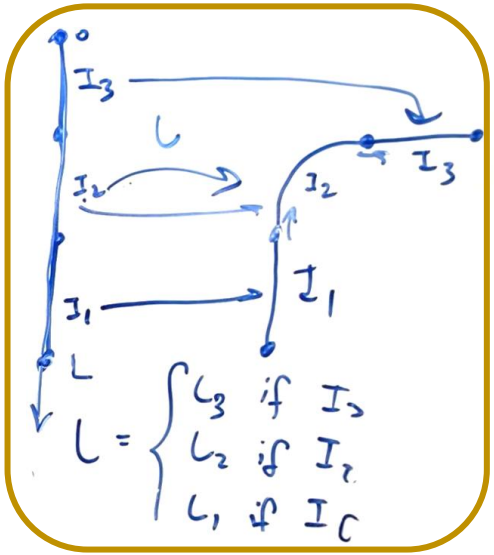
1 Introduction
2 Problem Statement
3 **Embedding Map Approaches**
4 Mathematical Formulation
5 Numerical Simulations
6 Route Refinement Algorithm
7 Conclusions and Future Work
8 Q&A

- possible approaches
- our choice of implementation

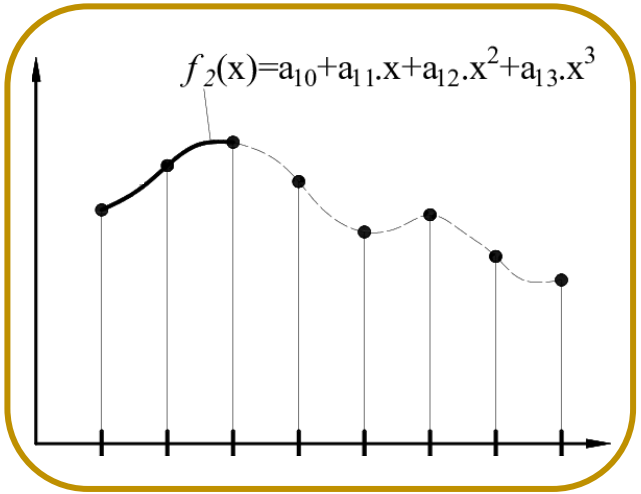
Implementing Embedding Function: How?



Explicit method

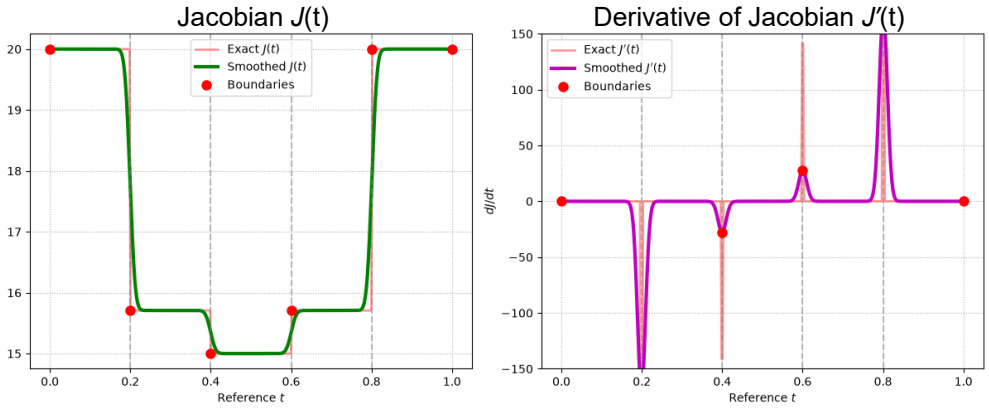


Spline approach



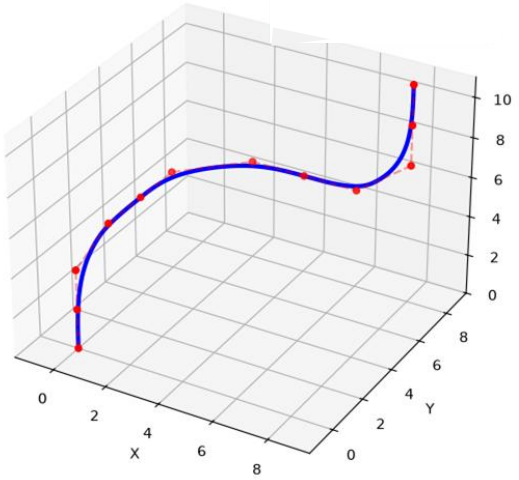
Explicit

- **Good:** very naturally approximates pipe shapes,
- **Major issue:** the gradient is no longer smooth!



vs Spline

- naturally **smooth** (C^2).
- a smooth and well behaved **Jacobian**.
- **built in** implementations in python.

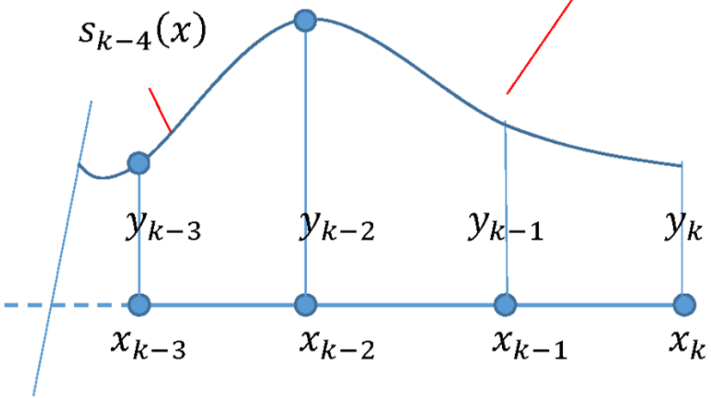


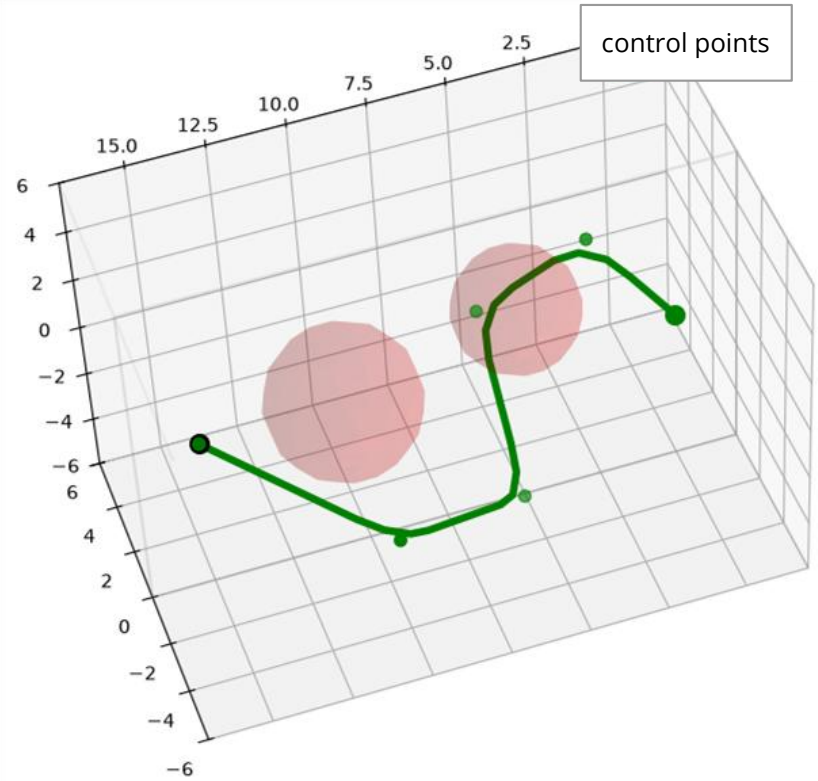
Caveat: How to make industrial pipes?

Spline Polynomial Approach: Industrial Pipes

To achieve even better results with industrially looking pipe shapes, we use a **combination of quadratic and linear splines**.

$$s_{k-3}(x) = a_{k-3} + b_{k-3}x + c_{k-3}x^2$$





1 Introduction
2 Problem Statement
3 Embedding Map Approaches
4 **Mathematical Formulation**
5 Numerical Simulations
6 Route Refinement Algorithm
7 Conclusions and Future Work
8 Q&A

- Weak Form Recap
- Timoshenko Equations
- Optimization Discussion
- Adjoint Formulas
- Gradient Update
- Expanded Theory

Weak Form Derivation

The team initially worked to reformulate the equations that Daikin previously derived into a **weak form**:

$$\int_{t_0}^{t_1}\int_M \mathrm{tr}\left([V]^T\frac{\partial w}{\partial y}\right)+\rho\frac{\partial^2 v}{\partial t^2}\cdot w dy dt=\int_{t_0}^{t_1}\int_{D_1} f(y,t)\cdot[V]\hat{n}dSdt$$

$$\int_M a\left(\nabla(M^2[\iota,v])\cdot[G]\kappa\right)-\mathrm{tr}\left([G]^T\frac{\partial \kappa}{\partial y}\right)dy=\int_{D_1} g\cdot[G]\hat{n}dS$$

$$[V]=\mu\frac{\partial v}{\partial y}+\mu F\left(\frac{\partial v}{\partial y}\right)^T F+\lambda\mathrm{tr}\left(\frac{\partial v}{\partial y}F^T\right)F$$

$$[G]=6\mu^2 A^T\frac{\partial v}{\partial y}+6\mu^2 A\frac{\partial v}{\partial y}-4\mu^2\mathrm{tr}(A)\frac{\partial v}{\partial y},\quad A=\frac{\partial v}{\partial y}F^T$$

First Try: Euler Bernoulli Beam Theory

- We moved to consider a 1D beam element undergoing a 1D displacement
    - But, mapping a 3D space undergoing deformation to a 1D space means the reference pipe experiences **full 3D deformations**.
- Euler Bernoulli **neglects shear deformation**, **assumes plane sections remain plane**, and results in the following set of 4 PDE's:

Bending:

$$\rho A\frac{\partial^2 u_y}{\partial t^2}+\frac{\partial^2}{\partial s^2}\left(EI_z\frac{\partial^2 u_y}{\partial s^2}\right)=0 \qquad \rho A\frac{\partial^2 u_z}{\partial t^2}+\frac{\partial^2}{\partial s^2}\left(EI_y\frac{\partial^2 u_z}{\partial s^2}\right)=0$$

Axial and Torsion:

$$\rho A\frac{\partial^2 u_x}{\partial t^2}-\frac{\partial}{\partial s}\left(EA\frac{\partial u_x}{\partial s}\right)=0 \qquad \rho I_p\frac{\partial^2 \theta}{\partial t^2}-\frac{\partial}{\partial s}\left(GJ\frac{\partial \theta}{\partial s}\right)=0$$

Issues with Euler Bernoulli

$$F = \frac{\partial \iota}{\partial \xi}$$

$$ds = ||F|| d\xi \rightarrow \frac{\partial}{\partial s} = \frac{1}{||F||} \frac{\partial}{\partial \xi}$$

- We transformed back to the reference coordinate, and found the **weak form:**

$$\int_{t_0}^{t_1} \int_0^1 ||F|| \left(\rho A \left(\frac{\partial^2 v}{\partial t^2} \cdot \bar{w} \right) + \rho I_p \frac{\partial^2 \phi}{\partial t^2} \bar{\psi} \right) d\xi dt + \int_{t_0}^{t_1} \int_0^1 \frac{1}{||F||} \left(EA \frac{\partial v_x}{\partial \xi} \frac{\partial \bar{w}_x}{\partial \xi} + GJ_t \frac{\partial \phi}{\partial \xi} \frac{\partial \bar{\psi}}{\partial \xi} \right) d\xi dt$$
$$+ \int_{t_0}^{t_1} \int_0^1 \frac{EI_z}{||F||} \left(\frac{\partial}{\partial \xi} \left(\frac{1}{||F||} \frac{\partial v_y}{\partial \xi} \right) \right) \left(\frac{\partial}{\partial \xi} \left(\frac{1}{||F||} \frac{\partial \bar{w}_y}{\partial \xi} \right) \right) d\xi dt$$
$$+ \int_{t_0}^{t_1} \int_0^1 \frac{EI_y}{||F||} \left(\frac{\partial}{\partial \xi} \left(\frac{1}{||F||} \frac{\partial v_z}{\partial \xi} \right) \right) \left(\frac{\partial}{\partial \xi} \left(\frac{1}{||F||} \frac{\partial \bar{w}_z}{\partial \xi} \right) \right) d\xi dt = 0.$$

- Note: We have **second order derivatives** still, and need to differentiate F.
  - This need for C^1 functions requires higher order shape functions, which are **computationally complex**, and more difficult to implement.
  - ⟶ We decided to move to the **Timoshenko Beam theory**.

The Fix: Timoshenko Beam Theory

- In Timoshenko theory, **shear is no longer neglected**, and the fourth order equation from EB becomes a pair of coupled second order ODEs.
- This **adds two new rotations** to solve for, but provides a nice advantage in the weak formulation:

Note: Only the bending equations change!

$$\rho A \frac{\partial^2 u_x}{\partial t^2} - \frac{\partial}{\partial s}\left(EA\frac{\partial u_x}{\partial s}\right) = 0 \qquad \rho I_p \frac{\partial^2 \theta_x}{\partial t^2} - \frac{\partial}{\partial s}\left(GJ\frac{\partial \theta_x}{\partial s}\right) = 0$$

$$\rho A \frac{\partial^2 u_y}{\partial t^2} + \frac{\partial}{\partial s}\left(\kappa_y AG\left(\frac{\partial u_y}{\partial s} - \theta_z\right)\right) = 0 \qquad \rho I_z \frac{\partial^2 \theta_z}{\partial t^2} - \frac{\partial}{\partial s}\left(EI_z\frac{\partial \theta_z}{\partial s}\right) + \kappa_y AG\left(\frac{\partial u_y}{\partial s} - \theta_z\right) = 0$$

$$\rho A \frac{\partial^2 u_z}{\partial t^2} + \frac{\partial}{\partial s}\left(\kappa_z AG\left(\frac{\partial u_z}{\partial s} + \theta_y\right)\right) = 0 \qquad \rho I_y \frac{\partial^2 \theta_y}{\partial t^2} - \frac{\partial}{\partial s}\left(EI_y\frac{\partial \theta_y}{\partial s}\right) + \kappa_z AG\left(\frac{\partial u_z}{\partial s} + \theta_y\right) = 0$$

Timoshenko Weak Form

- By applying the same transformation, and **converting everything to a weak formulation**, we can recover the following weak form:

All spatial derivatives are 1st order!

$$\text{Axial: } \int_{t_0}^{t_1}\int_0^1 \rho A \frac{\partial^2 v_x}{\partial t^2}\bar{w}_x ||F|| d\xi dt + \int_{t_0}^{t_1}\int_0^1 \frac{EA}{||F||}\frac{\partial v_x}{\partial \xi}\frac{\partial \bar{w}_x}{\partial \xi} d\xi dt = 0$$

$$\text{Torsion: } \int_{t_0}^{t_1}\int_0^1 \rho I_p \frac{\partial^2 \phi_x}{\partial t^2}\bar{\psi}_x ||F|| d\xi dt + \int_{t_0}^{t_1}\int_0^1 \frac{GJ_t}{||F||}\frac{\partial \phi_x}{\partial \xi}\frac{\partial \bar{\psi}_x}{\partial \xi} d\xi dt = 0$$

$$\text{Bending (XY): } \int_{t_0}^{t_1}\int_0^1 \left(\rho A \frac{\partial^2 v_y}{\partial t^2}\bar{w}_y + \rho I_p \frac{\partial^2 \phi_z}{\partial t^2}\bar{\psi}_z\right)||F|| d\xi dt + \int_{t_0}^{t_1}\int_0^1 \frac{EI_z}{||F||}\frac{\partial \phi_z}{\partial \xi}\frac{\partial \bar{\psi}_z}{\partial \xi} d\xi dt$$
$$+ \int_{t_0}^{t_1}\int_0^1 \kappa_y AG\left(\frac{1}{||F||}\frac{\partial v_y}{\partial \xi} - \phi_z\right)\left(\frac{1}{||F||}\frac{\partial \bar{w}_y}{\partial \xi} - \bar{\psi}_z\right)||F|| d\xi dt = 0$$

$$\text{Bending (XZ): } \int_{t_0}^{t_1}\int_0^1 \left(\rho A \frac{\partial^2 v_z}{\partial t^2}\bar{w}_z + \rho I_p \frac{\partial^2 \phi_y}{\partial t^2}\bar{\psi}_y\right)||F|| d\xi dt + \int_{t_0}^{t_1}\int_0^1 \frac{EI_y}{||F||}\frac{\partial \phi_y}{\partial \xi}\frac{\partial \bar{\psi}_y}{\partial \xi} d\xi dt$$
$$+ \int_{t_0}^{t_1}\int_0^1 \kappa_z AG\left(\frac{1}{||F||}\frac{\partial v_z}{\partial \xi} + \phi_y\right)\left(\frac{1}{||F||}\frac{\partial \bar{w}_z}{\partial \xi} + \bar{\psi}_y\right)||F|| d\xi dt = 0$$

Optimization - Overview

- By rewriting our weak form as the operator A, we can formulate the adjoint equation and gradient calculation for this problem.

$$A[v,\lambda] = \int_{t_0}^{t_1} \int_0^1 M(\ddot{v},\lambda) + K(v,\lambda) d\xi dt = 0$$

Gradient → $\frac{d\tilde{M}_a}{d\iota} = \frac{\partial \tilde{M}_a}{\partial \iota} - \frac{\partial}{\partial \iota}(A[v,\lambda]),$ $\frac{\partial \tilde{M}_a}{\partial v} = \frac{\partial}{\partial v} A[v,\lambda]$ ← Adjoint

- Solving our optimization problem now comes down to **two major steps:**
  - Constructing the **adjoint equation** to be solved in tandem with the forward problem.
  - Deriving the equation for the **gradient**, based on the results of both solves.

Step 1 - The Adjoint Method

$$\frac{\partial \tilde{M}_a}{\partial v} = \frac{\partial}{\partial v} A[v, \lambda]$$

$$M\ddot{v}_n + C\dot{v}_n + Kv_n = F_n \leftrightarrow K_{eff} v_n = F_n$$

(Forward FEM)

- Two main approaches are used when constructing the adjoint equations in an optimization problem.
  - We went with the **discrete approach**, as it is mathematically straightforward to construct, and results in a linear algebra problem:

$$A[v_n, \lambda_n] = \sum_{n=0}^{N_t} \lambda_n^T (K_{eff} v_n - F_n) = 0.$$

The integral is now a sum!

$$\frac{\partial}{\partial v_n} A = K_{eff}^T \lambda_n = \frac{\partial \tilde{M}_a}{\partial v_n} + \text{history}$$

The PDE is now a linear equation.

- This stress derivative on the RHS just becomes a number of **vector products**, and involves no actual numerical differentiation.

Step 2 - Finding the Gradient

$$\frac{d\tilde{M}_a}{d\iota} = \frac{\partial \tilde{M}_a}{\partial \iota} - \frac{\partial}{\partial \iota}(A[v, \lambda]),$$

- By utilizing the spline approach, we can **reduce our abstract derivatives** with respect to iota to derivatives with respect to control point position.
- While other properties could be considered, we will start only with **control point position**.
- Combining this notion with our discrete formulas:

$$\begin{aligned}\frac{d\tilde{M}_a}{dP_i} &= \frac{\partial \tilde{M}_a}{\partial P_i} - \frac{\partial}{\partial P_i}\sum_{n=0}^{N_t} \lambda_n^T (K_{eff} v_n - F_n) \\ &= \frac{\partial \tilde{M}_a}{\partial P_i} + \sum_{n=0}^{N_t} \lambda_n^T \left(\frac{\partial F_n}{\partial P_i} - \frac{\partial K_{eff}}{\partial P_i} v_n\right)\end{aligned}$$

Reformulated Continuum Mathematics / Next Steps

- We have also **reformulated the continuum mechanics** approach that Daikin originally provided.
  - We have removed the non-physical det F = 1 condition.
- Removing this assumption changes the shape optimization equations more than the underlying physics.
- **Next Steps:**
  - **Add sound!**
  - **Formulate things in the frequency domain.**

$$[V]^T = \mu F^{-1} A^T + \mu F^{-1} A + \lambda \mathrm{tr} A F^{-1} \qquad [G]^T = -6\mu^2 F^{-1} A^2 - 6\mu^2 F^{-1} A^T A + 2\mu^2 \mathrm{tr}(A) F^{-1} A$$

1 Introduction
2 Problem Statement
3 Embedding Map Approaches
4 Mathematical Formulation
5 **Numerical Simulations**
6 Route Refinement Algorithm
7 Conclusions and Future Work
8 Q&A

- the setup
- current results
- next steps

Numerical Simulations

Part 1: Implementation of all the equations and **physics**

- Uses derived weak form
- Ensures that the pipe behaves the way it would in real life
- Evaluates the stress at each point of the pipe and tests under a simple vibration

Part 2: Setting up **optimization** iteration to achieve automation

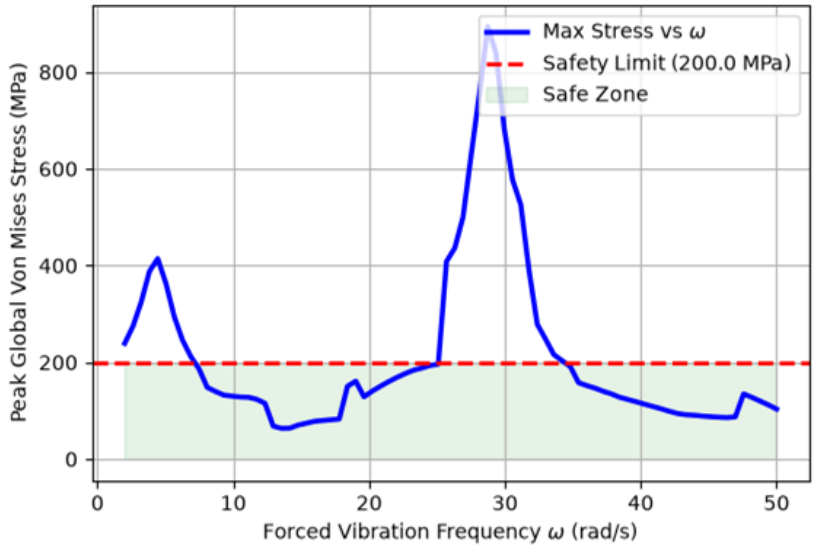
- Automatically updates the pipe design to minimize the stress
- Ideally, allows to incorporate additional constraints, such as obstacle avoidance

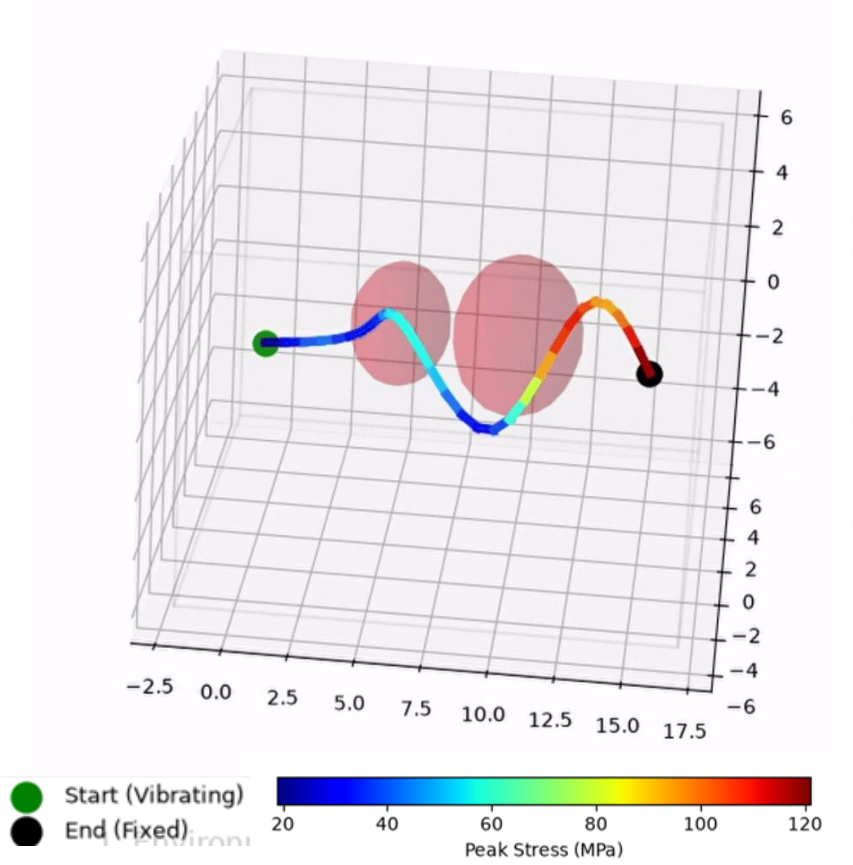
Numerical Simulation

Results

Numerical Simulation – Part 1: Physics ✅

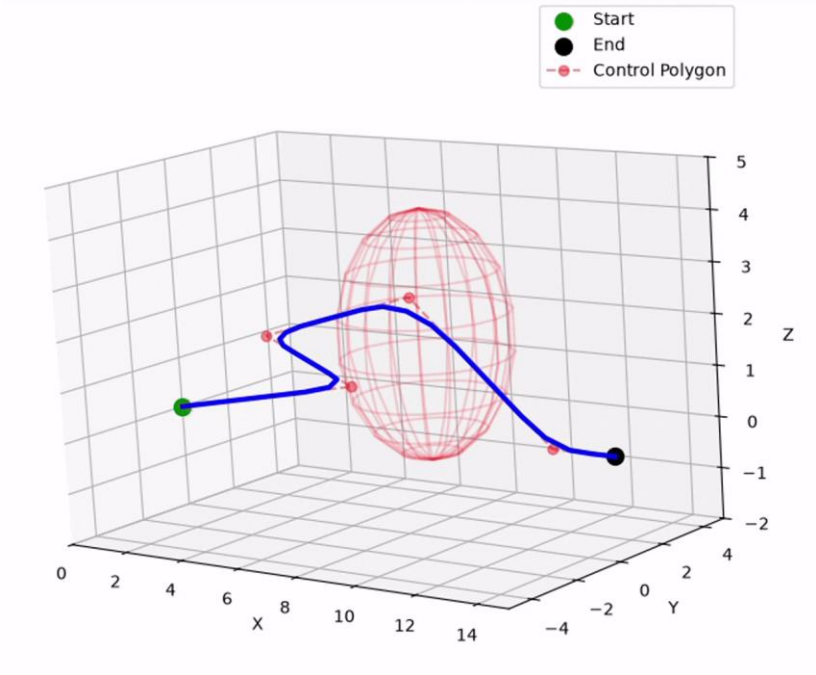
- we **vibrate** one of the ends of the pipe in simple function: sin(ωt)
- we calculate the von Mises **stress** and test it while varying vibration frequency ω



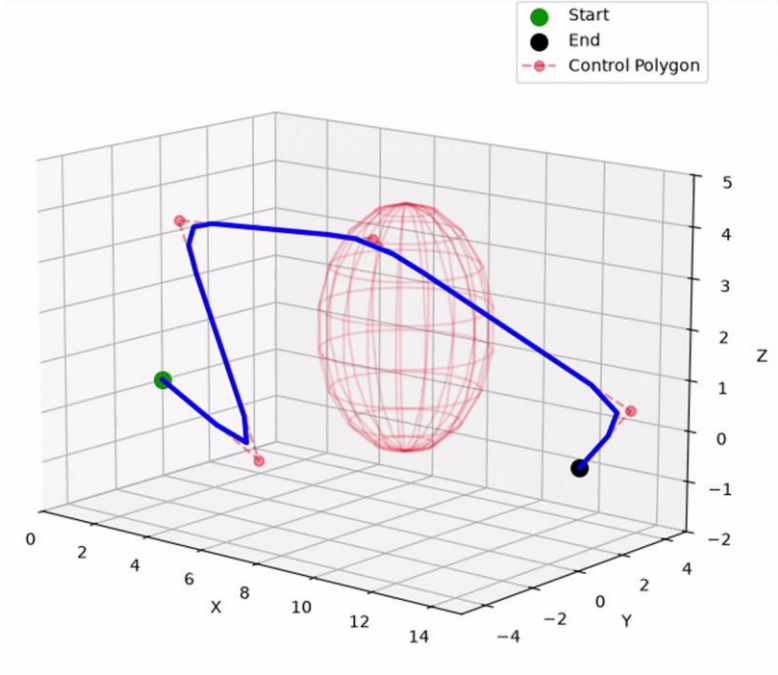


Part 2: Optimization – Results ✅

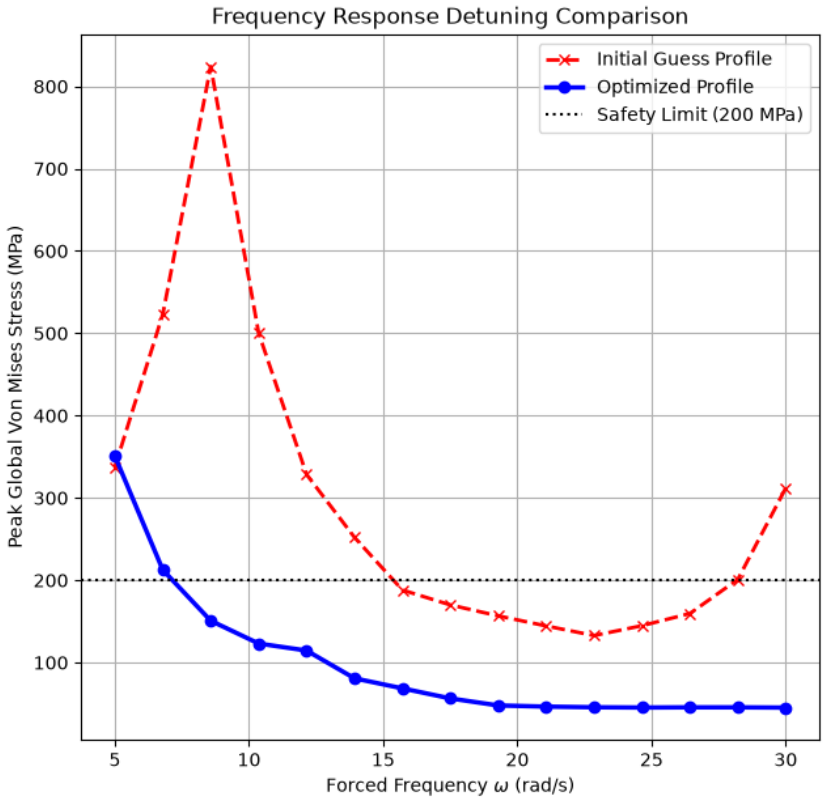
Initial guess

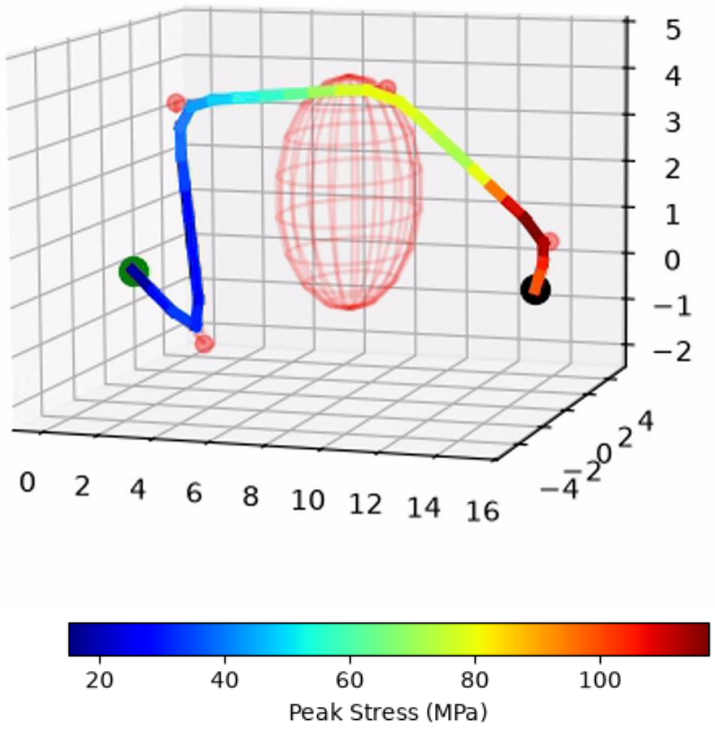


Optimized pipe



Part 2: Optimization – Stress Results





Optimization Implementation – Adjoint

Problems:

- **Memory** complexity — adjoint runs backwards in time and requires saving the entire forward trajectory u_n(t) to memory.
- Needs implementation of dedicated **algorithms**
- Require workaround for **non-smooth logic** and functions

Optimization Implementation – Pivot

Time constraint >>> pivot to **Finite Differences** in the implementation.

Reasons:

- implemented in python
- more accessible for a time-limited project

Limitations: **slow**.

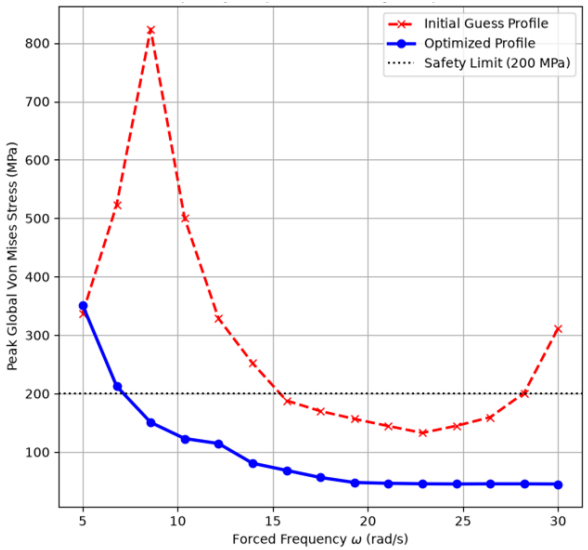
- finite differences: evaluates all the directions of the gradient
- adjoint: points in the correct direction.

** The adjoint method is still preferred.*

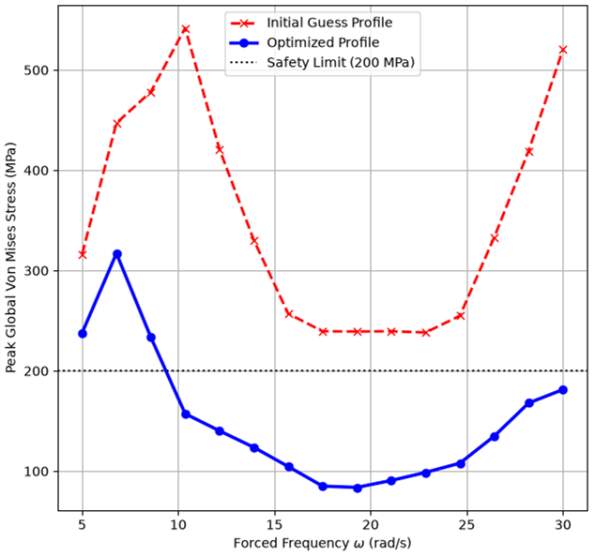
Numerical Simulation

Caveats

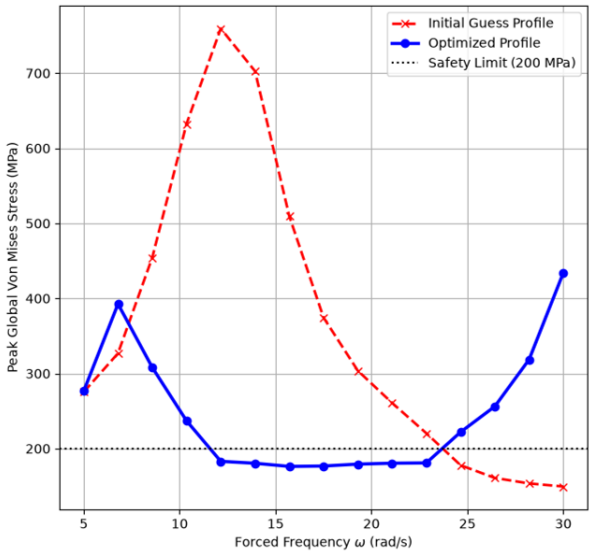
Optimization – Results: Comparison



#1



#2



#3

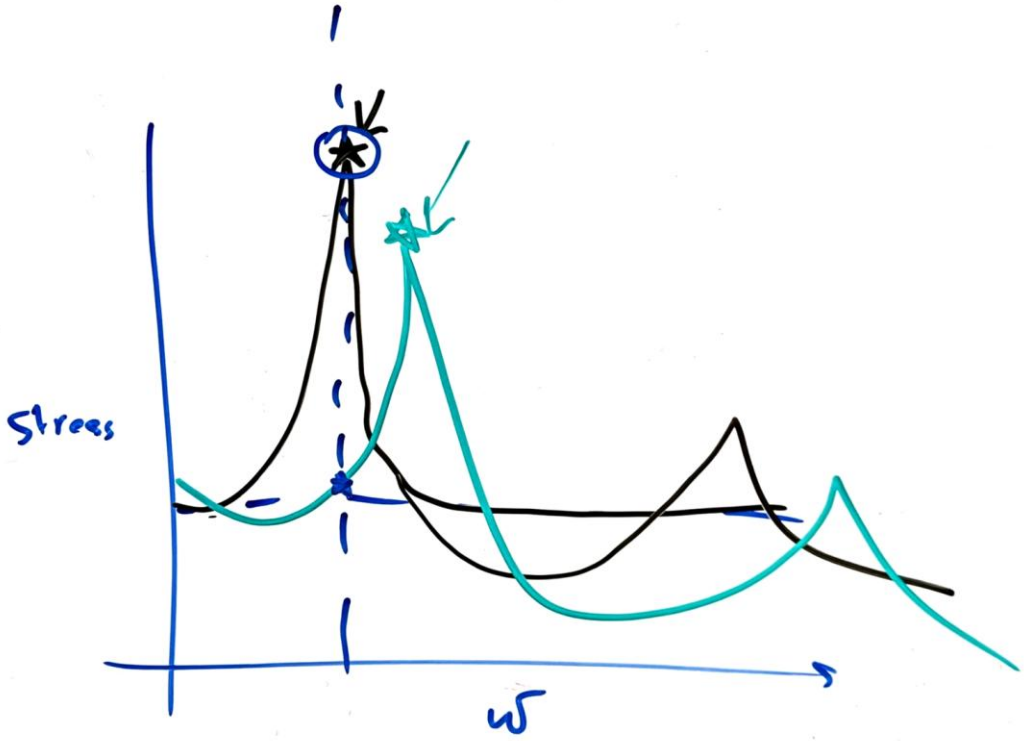
Optimization – Issue

Optimization KS function:

- finds the point of max stress
- minimizes the stress **at that frequency ω**

The problem:

- sometimes it just **shifts** the resonance point without minimizing global max stress

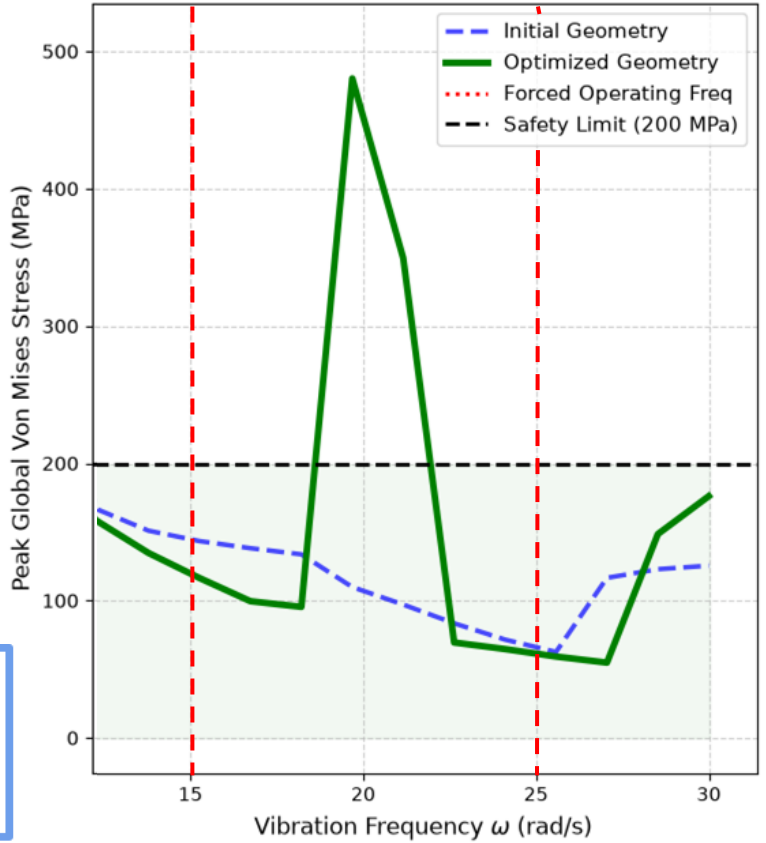


Optimization – Issue

KS function does not depend on frequency ω

Have to manually add frequencies to test on

It can still fail!

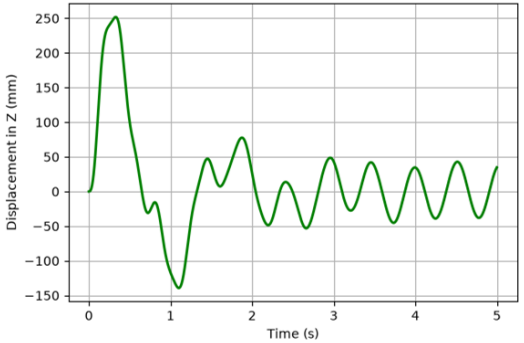


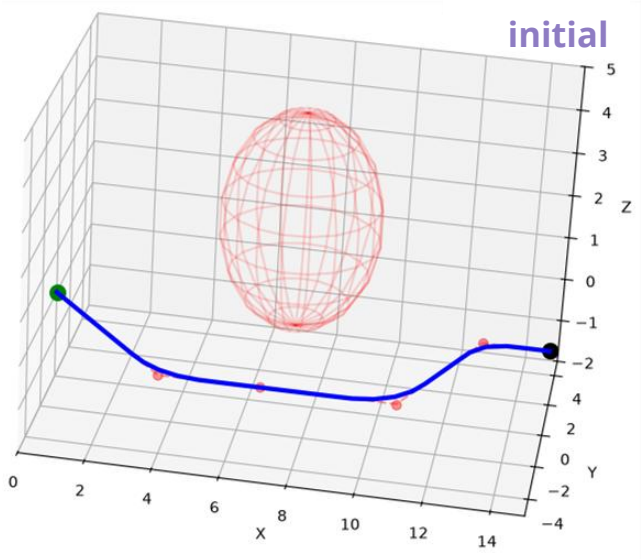
Possible future direction: *try solving this problem in the frequency domain where the frequency is parameter and we have control over it.*

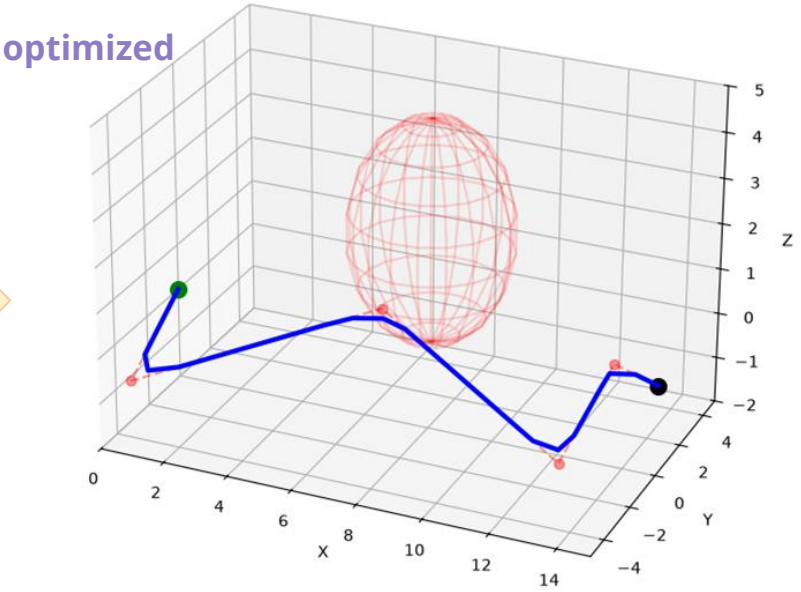
Optimization – Issue #2

Occasionally, the **optimized pipe becomes flat** (2D).

Educated guess: because the **test vibration** is being applied along only one axis (Z-axis).





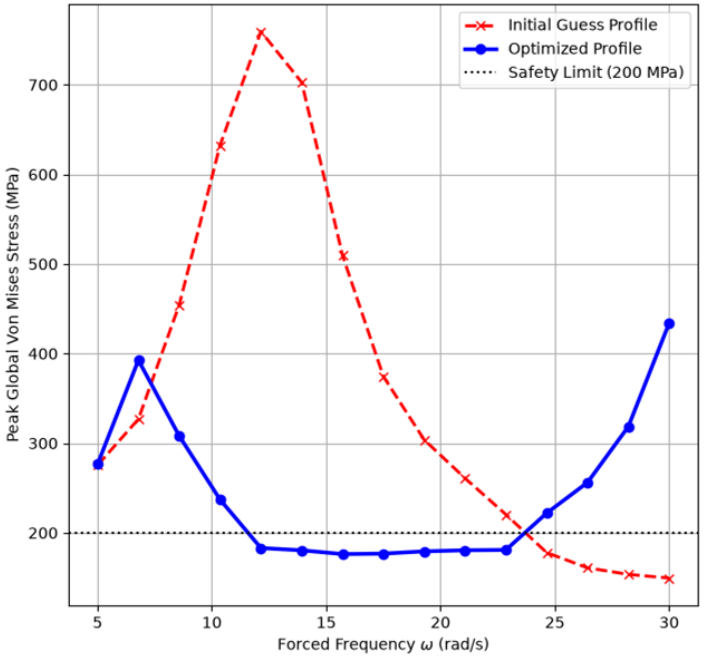


Numerical Simulation – Next Steps

Physics and one-frequency Optimization are complete! ✅

Avenues of future work:

- Fully implement the **adjoint method**
- Apply testing over more complicated **vibration function**
- Smooth out the shape of the optimized pipe to **avoid sharp angles**
- Move away from KS function to the **frequency domain analysis**



① Introduction
② Problem Statement
③ Embedding Map Approaches
④ Mathematical Formulation
⑤ Numerical Simulations
⑥ **Route Refinement Algorithm**
⑦ Conclusions and Future Work
⑧ Q&A

- core idea
- the practical purpose
- the algorithm

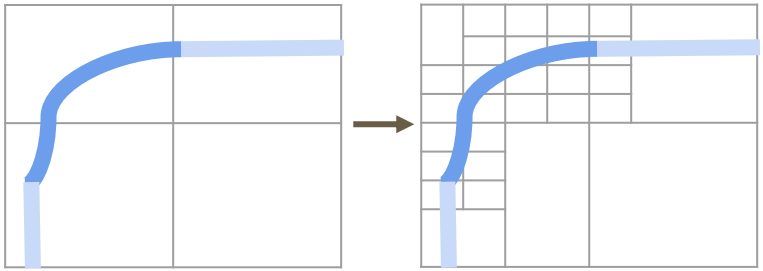
Route Refinement Algorithm (Core Idea)

Core idea: Keep both of the **coverage** and the **efficiency**

- Coverage: **Categorize & Perturb**



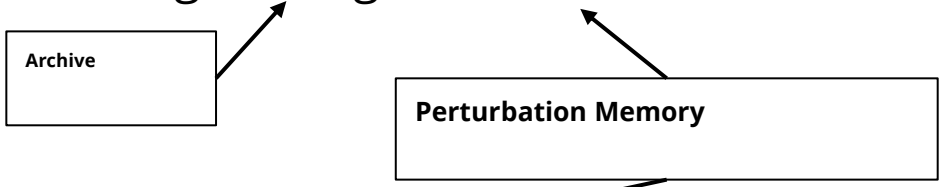
- Efficiency: **Coarse to fine**



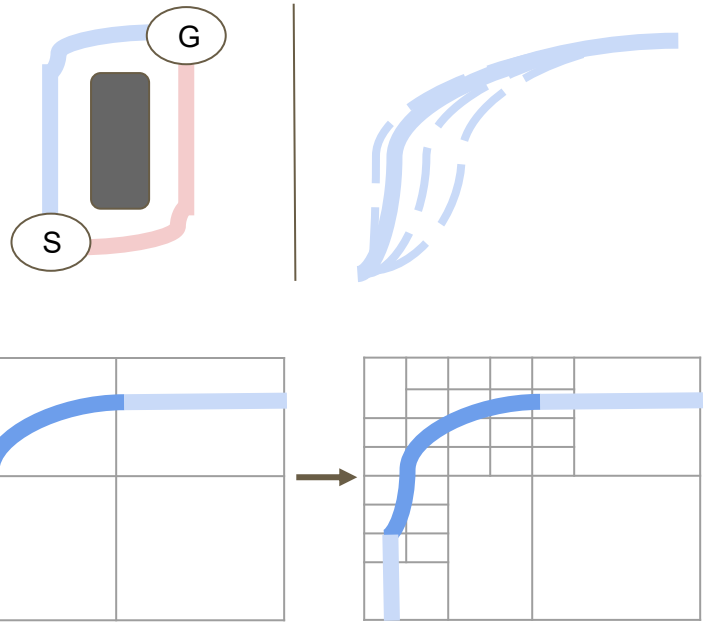
Route Refinement Algorithm (Core Idea)

Core idea: Keep both of the **coverage** and the **efficiency**

- Coverage: Categorize & Perturb



- Efficiency: Coarse to fine



New Methods for Practical Purpose



New Methods for Practical Purpose

1. Not vertical or horizontal!

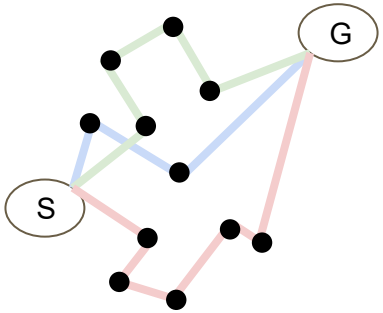


2. Extra U-turn parts?

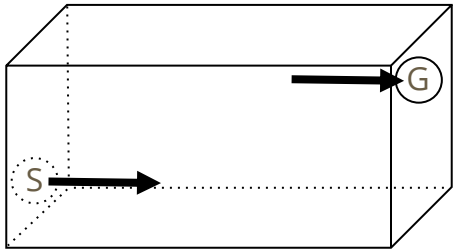
3. Endpoint has a direction!

New Methods for Practical Purpose

- **Angle-based** approach
- **U-turn** insertion
- Fixed endpoint **vectors**







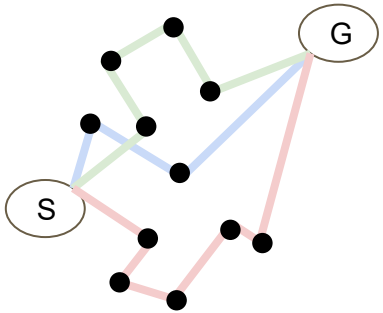
Overall Flow

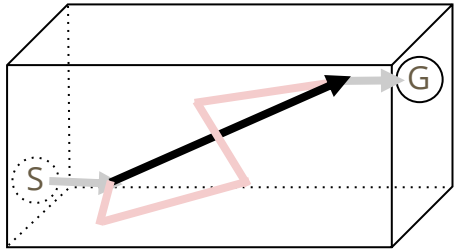
1. Generation & Categorization (= **Archive Generation**)
    1-1. Generation
    1-2. Moderate Elimination
    1-3. Upfront U-turn Insertion
    1-4. Categorization
2. Perturbation & Evaluation & Elimination (= **Archive Update**)
    2-1. Curve Perturbation
    2-2. U-turn Insertion

1 Generation & Categorization

1-1. Generate angle-based routes

indicators: number of curves, how deviate from the “main axis” →





1-2. Moderately narrow down for prevention of bias

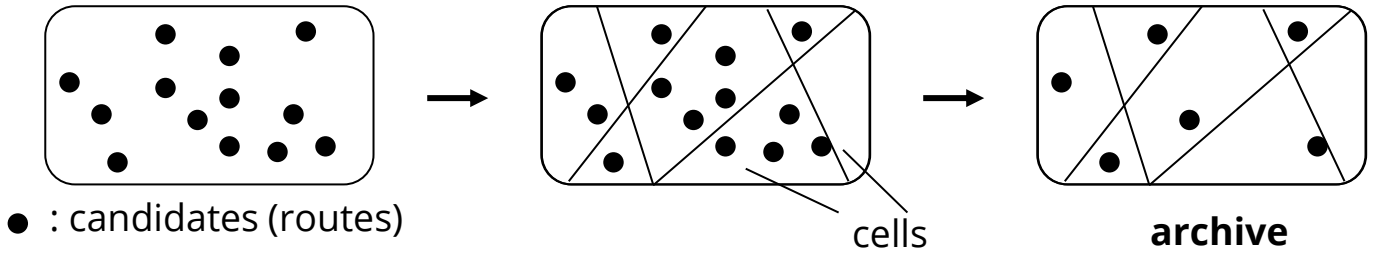
1 Generation & Categorization

1-3. Insert U-turns for variety



1 Generation & Categorization

1-4. Make cells (standardized categories) and an archive



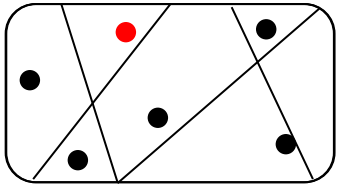
Merits of archive

- Keeping **diversity** of routes
- All of the routes are **survivors**

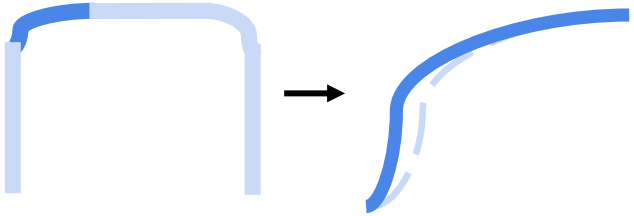
2 Perturbation & Evaluation & Elimination

2-1. Iterate the following (1)-(8)

(1) Pick up one route (cell) from the archive



(2) Perturb one curve



(3) Evaluate it locally (coarsely) and Keep improvement

2 Perturbation & Evaluation & Elimination

2-1. Iterate the following (cont.)

(4) Iterate (2) and (3) (focusing on one curve)

(2) Perturbation

(3) Local Evaluation / Deferred Acceptance

- with **updating direction/score memories**

(5) Iterate (4) for each curve

2 Perturbation & Evaluation & Elimination

2-1. Iterate the following (cont.)

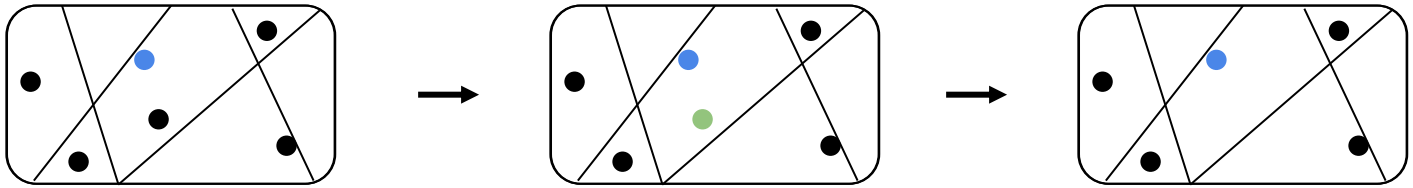
(6) Assign the kept one ( ) to an appropriate cell

(If the cell is empty, ( ) is accepted)

(7) Compare the existing one ( ) and the kept one ( )

with Non-local (fine) evaluation

(8) If the latter wins, it replace the former



2 Perturbation & Evaluation & Elimination

2-2. Try U-turn insertion

For each cell (route), iterate the following

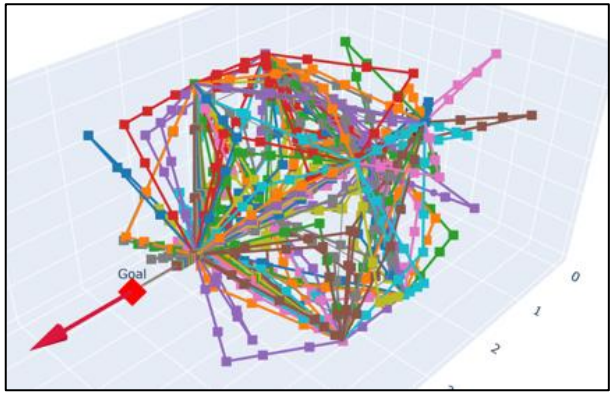
- Choose one leg (between curves) of the route
- Generate U-turn inserted route (•)
- Check the cell which (•) belongs to
  - If the cell is empty, (•) is accepted
  - Otherwise, comparison happens

Iterate 2-1 and 2-2

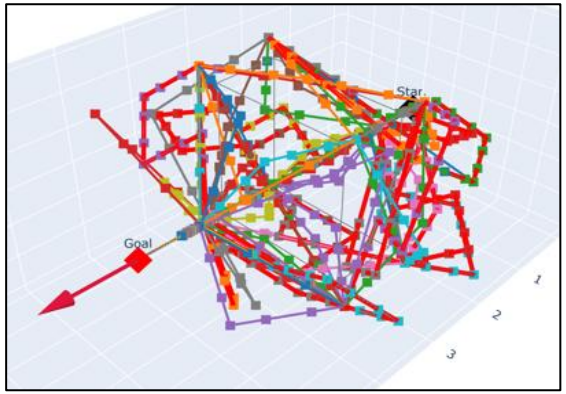
→ **Updated Archive** (a set of distinct & refined routes)

Route Refinement Algorithm (Example)

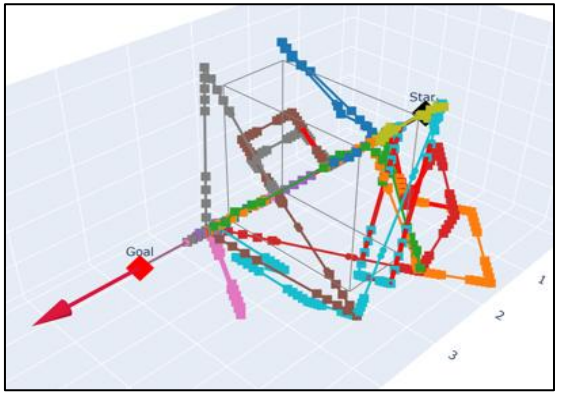
Generated Routes



After the 1st Step



After the 2nd Step



U-turn Insertion

Categorization

Iterated Update

1. Introduction
2. Problem Statement
3. Embedding Map Approaches
4. Mathematical Formulation
5. Numerical Simulations
6. Route Refinement Algorithm
7. **Conclusions and Future Work**
8. Q&A

- summary
- future work
- acknowledgements

Conclusion & Future Work

- The MVP mathematics are now finished! ✅
- We have a working optimization pipeline using the more cumbersome finite difference (direct gradient) method. ✅
- The route optimization algorithm has reached a working state. ✅

- Next steps: ⏳
  - The continuum theory can be further improved.
  - Finish working with the finite difference method, and focus on implementing the adjoint optimization formula.
  - Integrate the full optimization/simulation code with the pipe routing algorithm.

Our Sponsors















G-RIPS
Sendai 2026

主催

- 東北大学・数理科学共創社会センター (MathCCS)
- 東北大学・知の創出センター (TFC)
- 東北大学・材料科学高等研究所 (WPI-AIMR)

共催

- 理化学研究所・数理創造研究センター (iTHEMS)
- Institute for Pure & Applied Mathematics, UCLA