# A Differentiable Framework for Aircraft Design Optimization

Peter Sharpe
MIT Aeronautics and Astronautics

BAE Systems

March 22, 2021

# About Me

2nd year graduate student at MIT Int'l Cntr. for Air Transportation (Prof. John Hansman)

Research focuses:

- Aircraft Design
- Multidisciplinary Design Optimization (MDO)
- Computational Aerodynamics

These are all really the same thing!*

*or can be reduced to the same thing

# Outline

01 Motivations and challenges for aircraft design optimization (10 min.)

02 AeroSandbox: a new design framework (20 min.)

03 Applications and case studies (20 min.)

# Aircraft Design Optimization

Motivations and Challenges

# Motivations: Why Optimization?

- Optimization is one of *the* mathematical problems using up the most CPU cycles around the world at this second.
- Engineering design is fundamentally an optimization problem.
  - Sometimes formalized, sometimes not
  - Four key parts:
    - Objective function: The metric you are minimizing
    - Variables: The "knobs" that you have to work with
      - Sets the dimensionality of the design space
    - Constraints: Your requirements
      - Limits the design space to a feasible space
      - Equality constraints are a projection (dimensionality reduction)
      - Can be *active* or *inactive*
    - Parameters: Your assumptions

Problem formulation is *by far* the most important step of design optimization

Firefly

Maximize: $Range$

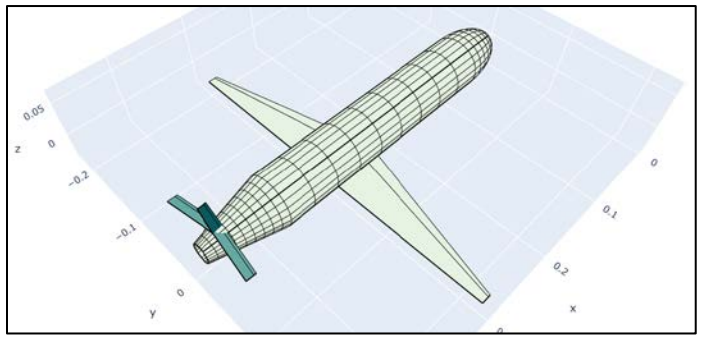
With respect to: $Trajectory, vehicle\ OML, CONOPS, propulsion\ sizing, mechanism\ design, etc.$

Subject to: $Requirements, risk\ metrics, physics, energy\ closure\ over\ 24h$

# Example: The Firefly Design Problem

Simultaneously optimizing...

## Aircraft Design



<u>Design Variables</u>

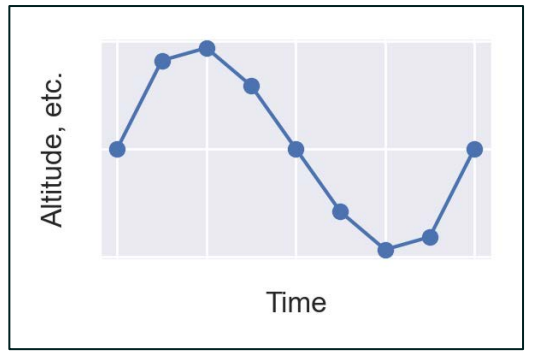
*Hundreds, including:*

- *wing, stabilizer, fuselage geom.*
- *nozzle dimensions*
- *propellant mass and chemistry*
- *insulation thickness*

<u>Parameters</u>

- *canister dimensions*
- *payload dimensions*
- *overall configuration*

## Mission Profile (simulation/control)




<u>Operational Variables</u>

*Thousands, including:*

- *altitude*
- *airspeed*
- *oxamide distribution*
- *ctrl. surf. inputs*

<u>Parameters</u>

- *altitude constraints*
- *speed requirements*
- *propulsion requirements & margins (e.g. no flame-out)*

<u>Objective Function</u>

*Maximize total range*

<u>Constraints</u>

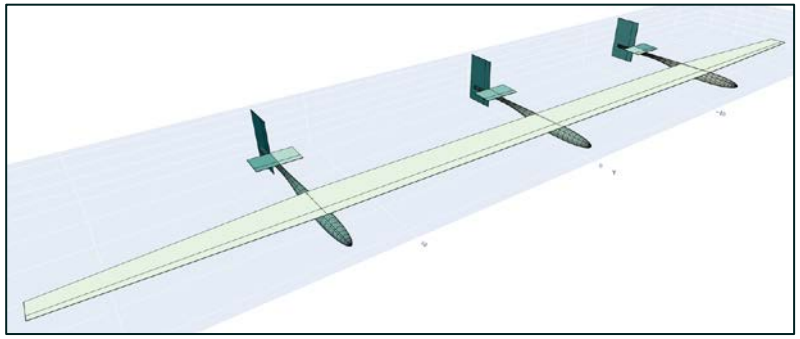
*Physics models*

- *~150 models, 9,000 constraints*

# Example: The Solar Airplane Design Problem

Simultaneously optimizing...

## Aircraft Design



<u>Design Variables</u>

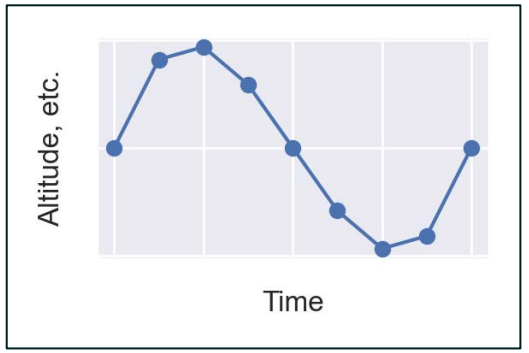
*Hundreds, including:*

- *wing geom. (two-sect.),*
- *tail geom. (one-sect.),*
- *boom geom.,*
- *propeller params.,*
- *battery cap.,*
- *solar panel area,*
- *motor rated power,*
- *wing internal struct.*

<u>Parameters</u>

- *payload mass*
- *# of booms*
- *battery spec. energy*
- *margins*

## Mission Profile (simulation/control)




<u>Operational Variables</u>

*Thousands, including:*

- *altitude*
- *airspeed*
- *throttle*
- *ctrl. surf. inputs*

<u>Parameters</u>

- *min. altitude*
- *wind speed dist.*
- *latitude*
- *day of year*

<u>Objective Function</u>

*Minimize wingspan*

<u>Constraints</u>

*Physics models*

- *~150 models, ~4,000 constraints*

# Optimization Challenges

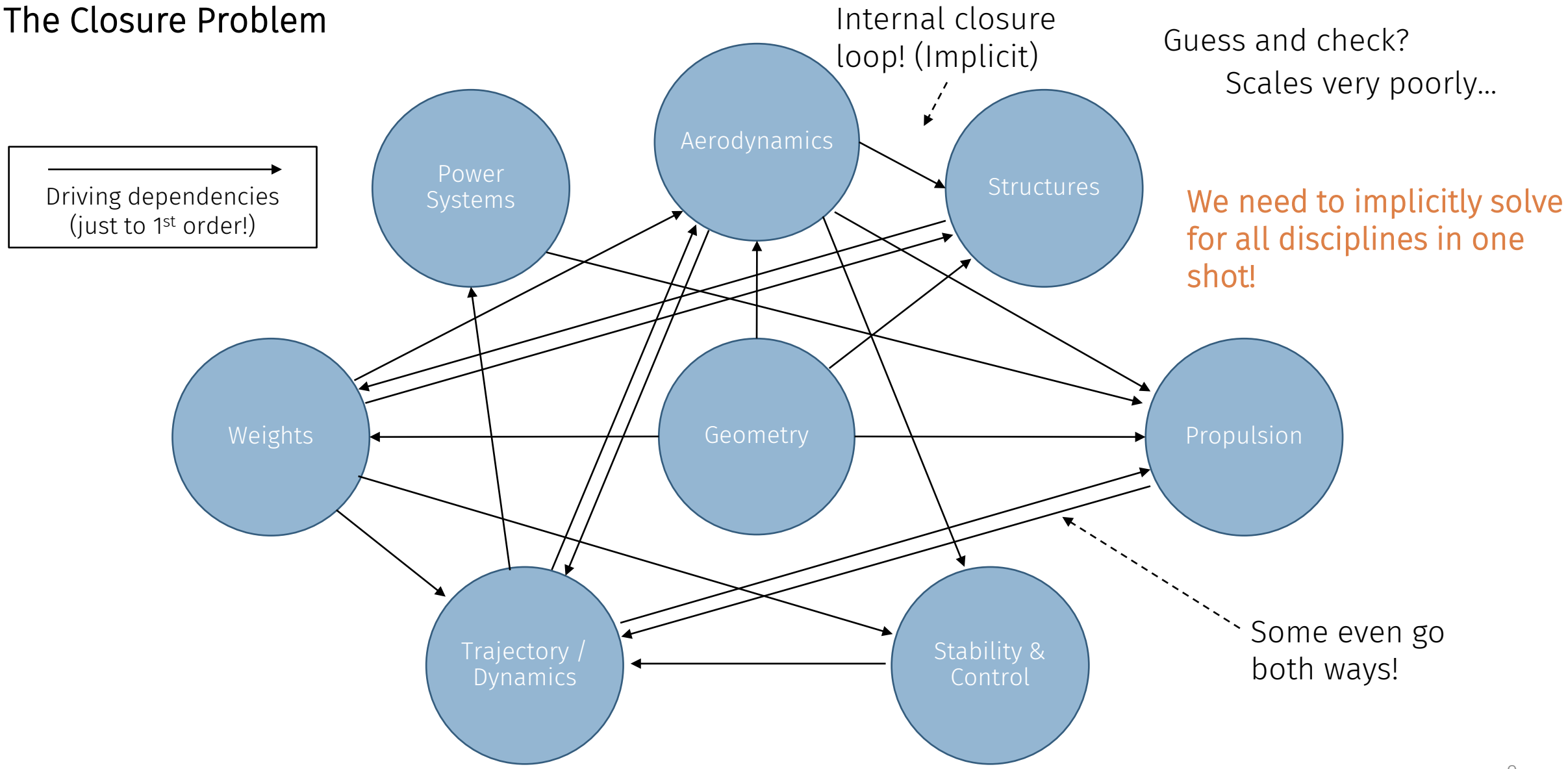
Three key challenges in aircraft design:

- Highly-coupled (multidisciplinary)
  - The closure problem!
- High-dimensional optimization
  - $n_{\text{variables}} \approx \mathcal{O}(10^3)$
  - $n_{\text{constraints}} \approx \mathcal{O}(10^4)$
- Dynamic systems (time-dependent):
  - 787 Dreamliner has a fuel mass fraction of 44% -> large $C_L$ changes
  - Solar airplane: diurnal energy cycle, ascent, wind variation, varying altitude
  - Firefly: range varies by ~2x depending on precise trajectory ($M_{crit}$, $I_{sp}$ trades)

| Typical Disciplines | |
|---|---|
| | Aerodynamics |
| | Structures |
| | Propulsion |
| | Power Systems |
| | Stability & Control |
| | Trajectory/Dynamics |
| | Weights |
| | Mission-specific requirements |
| | … |

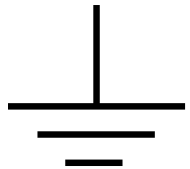
# Addressing Coupled Problems

## The Closure Problem



# Why are Aerospace Problems so Coupled?

- Typical engineering systems:

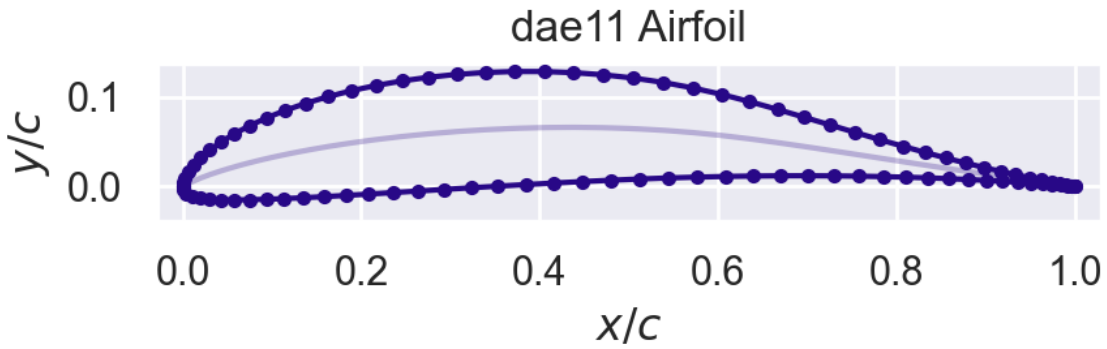


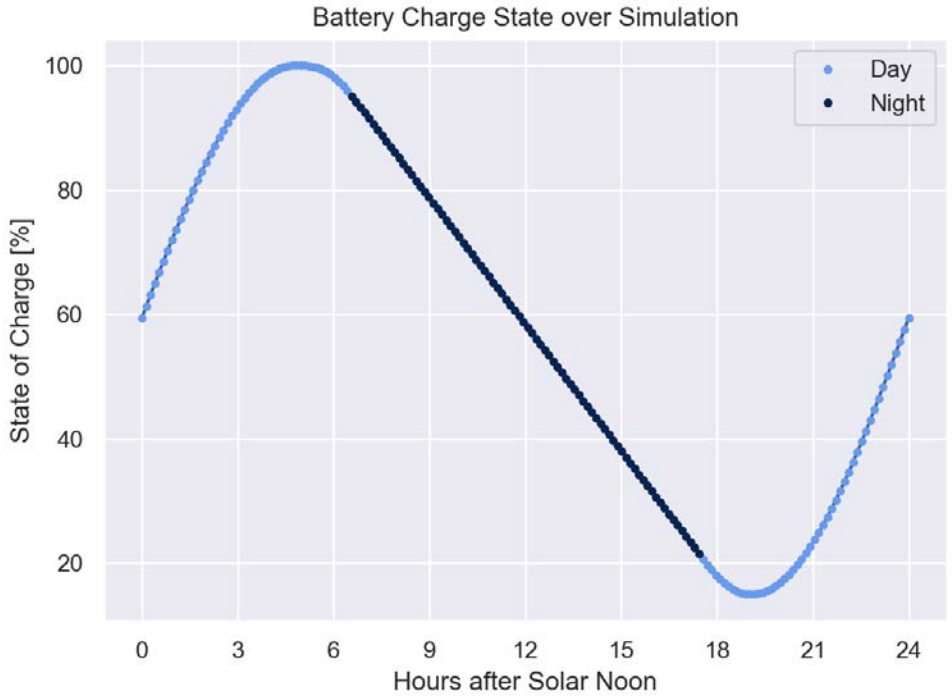
$$T = T_\infty$$

- On free-flying aerospace systems:
  - Nowhere to dump conserved quantities:
    - Mass
    - Momentum
    - Energy
    - Heat
    - Charge
    - etc.
- Each force and moment must be perfectly balanced with all others
- All subsystems must be designed at the same time

# Why are Aerospace Problems so High-Dimensional?

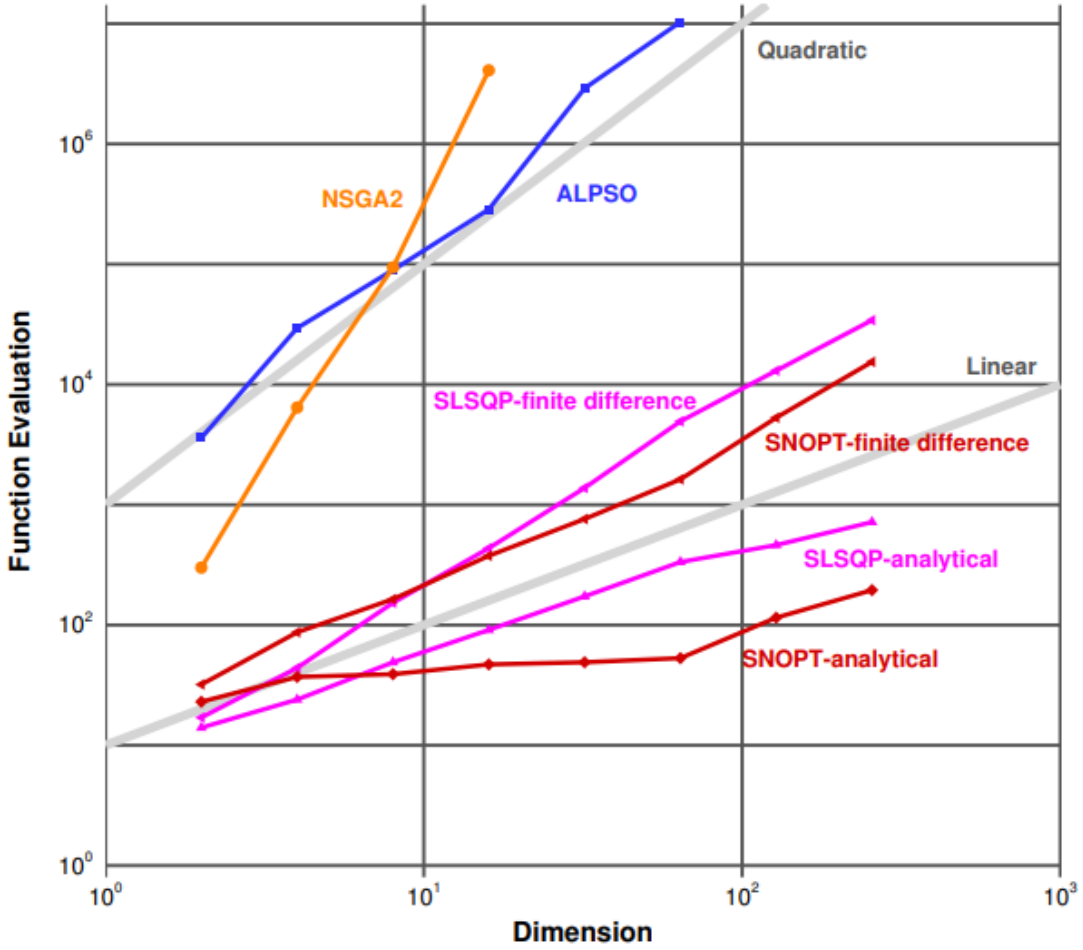
- Two reasons:
  - OML parameterization
    - Smooth, curving surfaces are complicated functions
  - Dynamics parameterization
    - $h(t), \alpha(t), \gamma(t)$, etc. are functions of time
- Function space is infinite-dimensional
  - Discretization can still require hundreds of variables



# High-Dimensional Optimization

- "Curse of Dimensionality"
  - For an $n$-dimensional problem, where each variable has $k$ possible levels:
  - Design space has $k^n$ points – grows exponentially
- Gradient-based optimization is the only tractable approach to large problems
  - Need derivatives (gradient) of objective + constraints
- Gradient computation is the bottleneck. Traditional options:
  - Finite difference: $O(n)$, imprecise
  - Symbolic differentiation: intractable
  - Hand-coded adjoints ("analytical"): efficient, but tedious and error-prone




Dimensionality vs. Computational Cost

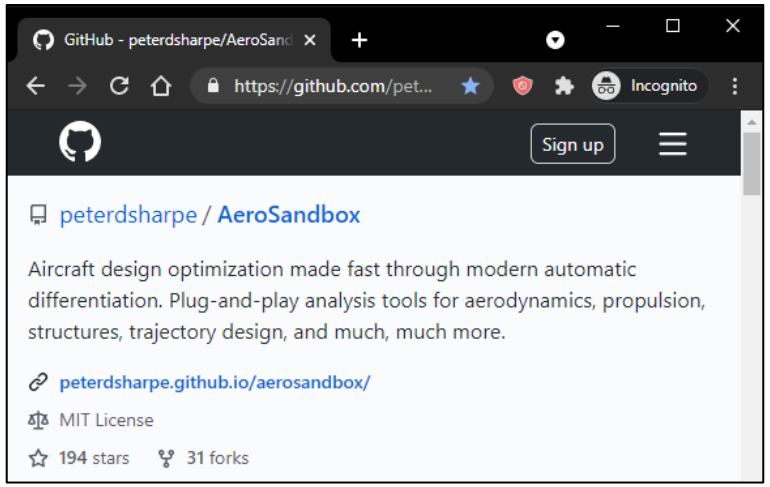
Figure reproduced from J. Martins, 2013 pres., "A Very Short Course on Multidisciplinary Design Optimization"

# AeroSandbox

A Differentiable Aircraft Design Framework

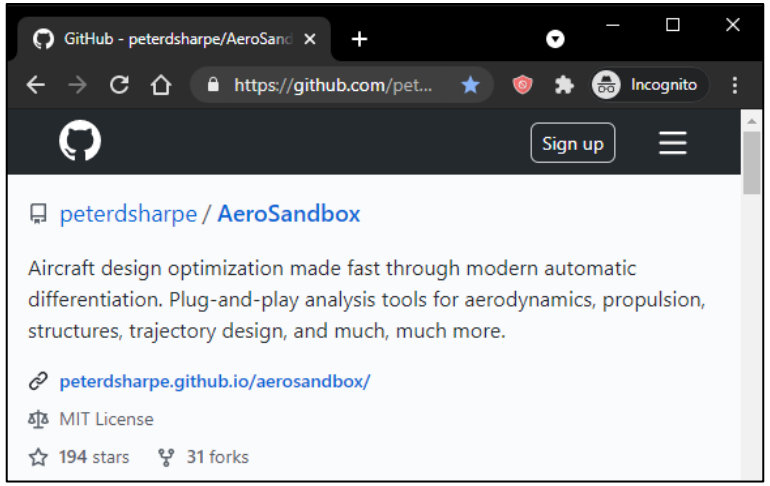
# AeroSandbox Overview

- Key features:
  - An optimization framework for engineered systems
  - Contains hundreds of mutually-compatible differentiable aerospace models
    - Surrogate modeling tools to make new differentiable models from user data (e.g. CFD, flight test)
  - Object-oriented aircraft geometry framework and performance stack

# AeroSandbox Overview

- Code details:
  - Python 3
  - Cross-platform
  - Open-source (MIT license), GitHub-hosted
- A rapidly growing ecosystem:
  - 18 months old, originally created to support the BAE-sponsored Firefly program
  - 126k downloads, 17k in past month
  - 3 student developers, many other internal MIT contributors
  - 19 external contributions to-date from various universities, companies

# “Opti” Optimization Stack

- Rapid learning curve
  - Object-oriented notation
  - Natural syntax
    - Numerics are abstracted away – lets you focus on problem formulation
- Example problem:
  - Minimize drag of a rectangular wing by changing chord and span, given:
    - Sea level atmosphere
    - $V_\infty = 1\ \mathrm{m/s}$, $C_L = 0.4$
    - Fixed wing area $S = 1$

Make an optimization problem

Make design variables and set initial guesses

Model profile drag*

Model induced drag

Constrain the wing area

Set an objective function

Solve

Show results

```
opti = asb.Opti()

chord = opti.variable(init_guess=1)
span = opti.variable(init_guess=1)

Re = velocity * chord / kinematic_viscosity
CD_p = 1.328 * Re ** -0.5

AR = span / chord
CD_i = CL ** 2 / (pi * AR)

opti.subject_to(chord * span == 1)

opti.minimize(CD_p + CD_i)

sol = opti.solve()

print(sol.value(chord), sol.value(span))
```

Imports/constants above

Result: $c = 0.229$ m, $b = 4.37$ m

# “Opti” Optimization Stack

- Opti is fast – *really, really* fast.
  - Speed enables “interactive design”
- “Secret sauce”:
  - Automatic differentiation (AD) everywhere
  - Simultaneous Analysis and Design (SAND) architecture
  - Sparsity exploitation
- Up next:
  - Under the hood
  - Solutions enabled

Example Problem:

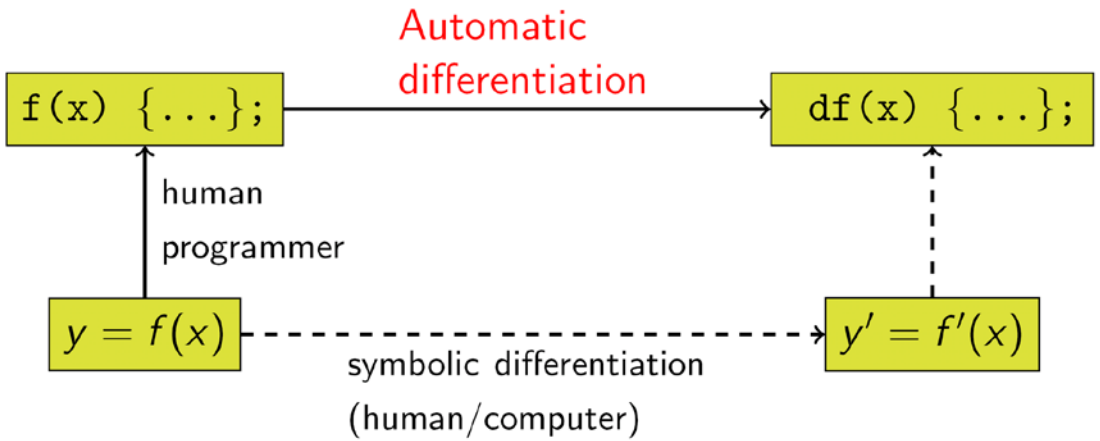
- Firefly design optimization
  - 8 disciplines
  - 3910 variables
  - 8420 constraints
  - 34321 constraint-variable interactions
  - Nonlinear, nonconvex
- 6.91 seconds to solve on a laptop.

```
Number of nonzeros in equality constraint Jacobian...:    22321
Number of nonzeros in inequality constraint Jacobian.:    12021
Number of nonzeros in Lagrangian Hessian.............:    21622

Total number of variables............................:     3910
Total number of equality constraints.................:     2701
Total number of inequality constraints...............:     5719
EXIT: Optimal Solution Found.
      solver  :   t_proc
       nlp_f  |        0
       nlp_g  | 135.00ms
  nlp_grad_f  |   2.00ms
  nlp_hess_l  | 533.00ms
   nlp_jac_g  | 289.00ms
       total  |   6.91 s
```

# Automatic Differentiation (AD)

- A way to compute gradients *of code*
  - Highly related: adjoint methods, "backpropagation"
- For a function $\vec{f}(\vec{x})$: $\mathbb{R}^m \rightarrow \mathbb{R}^n$:
  - 1st-order finite diff.: $\mathcal{O}(mn)$, $\epsilon \approx 10^{-8}$
  - Forward-mode AD: $\mathcal{O}(m)$, $\epsilon \approx 10^{-16}$
  - Reverse-mode AD: $\mathcal{O}(n)$, $\epsilon \approx 10^{-16}$
- Most useful functions have $m \gg n$, so we use reverse-mode AD
  - E.g. for objective functions, $n = 1$
- Calculate an objective function's gradient in $\mathcal{O}(1)$ time instead of $\mathcal{O}(m)$ time

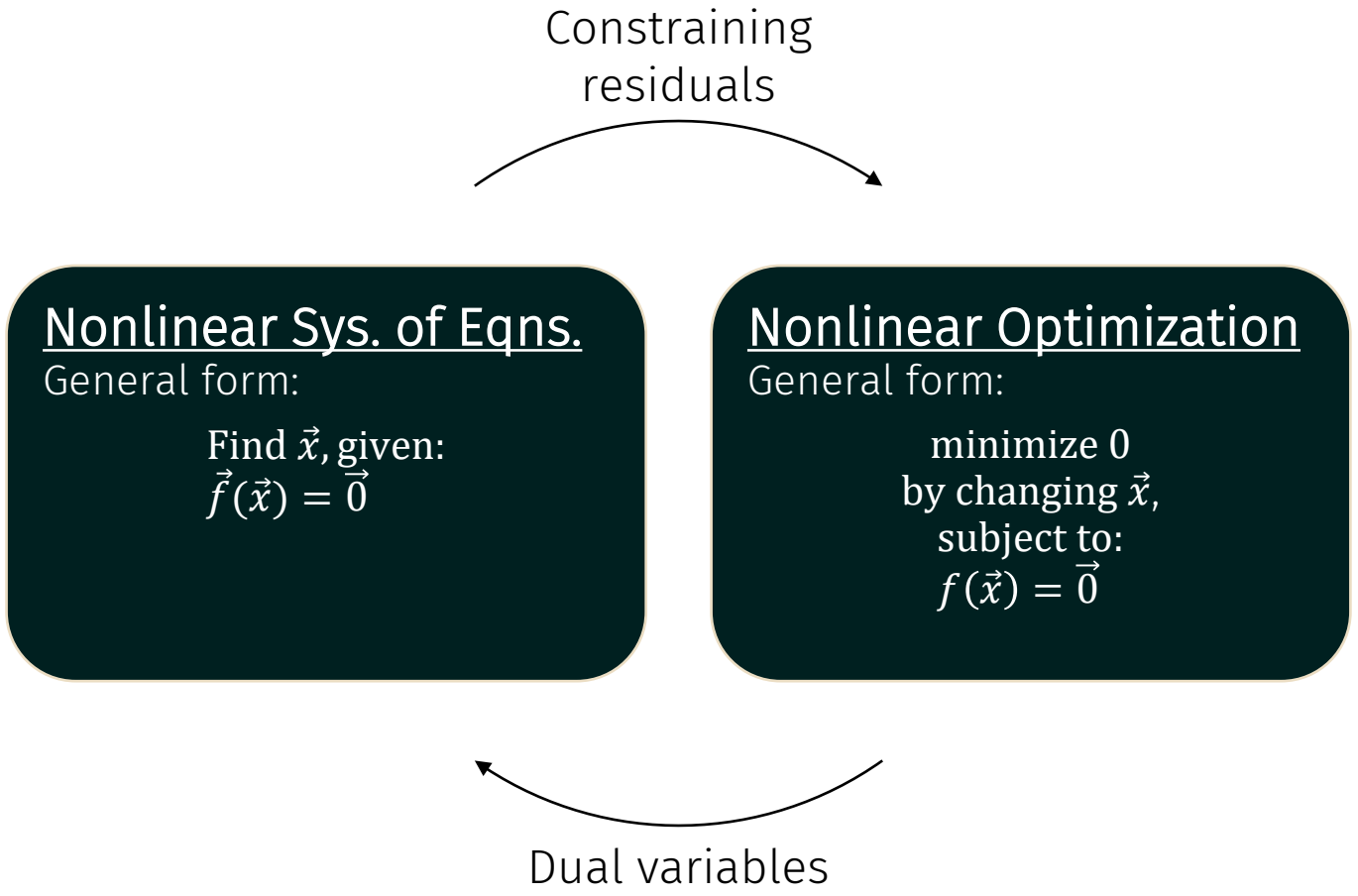
$$\frac{\partial \vec{f}}{\partial \vec{x}} = \overbrace{\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \ddots \end{bmatrix}}^{m} \Big\} n$$




Relationship between automatic and symbolic differentiation (Source: Wikimedia)

# Simultaneous Analysis and Design (SAND)

- Analysis: solving systems of nonlinear equations
  - ODEs, PDEs can be discretized
  - Internal closure loops can be constructed as residuals
- How to optimize:
  - Traditional: wrap analysis in an optimizer (nested)
  - SAND: embed analysis into optimization as constraints
    - Solve governing eqns. and optimality simultaneously
    - Eliminates internal closure loops




* Exploiting this duality for engineering design was pioneered by Raphael Haftka in the 1980s

# Caveat: Glass-Box Requirements

- Both AD and SAND require glass-box models:
  - Requires access to governing equations
  - Analysis coded in Python using **aerosandbox.numpy**
- Options when using legacy black-box tools (e.g. RANS CFD):
  - Surrogate model (synthetic data)
    - Fit analytic model to data
    - Interpolate from data
      - ASB provides differentiable $n$-dimensional b-spline lookup tables
  - Finite-difference a single model (WIP)

Drop-in differentiable replacement of NumPy/SciPy*:

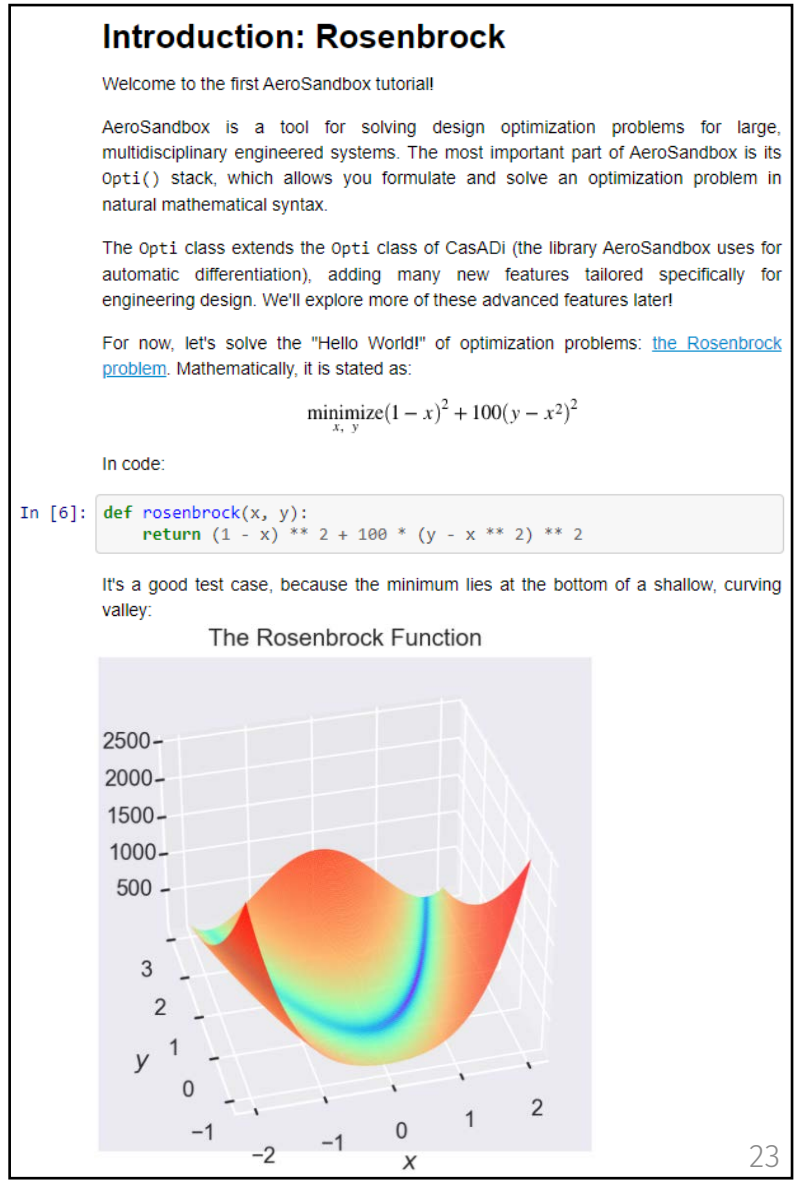
```
# import numpy as np
import aerosandbox.numpy as np
```

- N-dim array operations
- Arithmetic, trig., log/exp
- Conditionals, loops, logical operations
- Linear algebra
  - (linear solves, pseudoinverses, norms, eigenvalues, factorizations)
- Interpolation
- Numerical differentiation/integration
- Advanced: solvers for subproblems:
  - Rootfinders
  - QP and SOCP solvers
  - Adaptive-step integrators
- etc.

*Nearly complete

# Documentation and Testing

- Code is extensively documented
  - Comment:code ratio $> 0.75:1$
  - Inline docs and usage examples for all objects and most functions with **help()** command
- 150+ unit tests automatically run after each codebase edit
  - Caveat: Currently migrating from ASB 2 to ASB 3, a few hiccups for the next week
- Dozens of interactive tutorials on GitHub



## Introduction: Rosenbrock

Welcome to the first AeroSandbox tutorial!

AeroSandbox is a tool for solving design optimization problems for large, multidisciplinary engineered systems. The most important part of AeroSandbox is its `Opti()` stack, which allows you formulate and solve an optimization problem in natural mathematical syntax.

The `Opti` class extends the `Opti` class of CasADi (the library AeroSandbox uses for automatic differentiation), adding many new features tailored specifically for engineering design. We'll explore more of these advanced features later!

For now, let's solve the "Hello World!" of optimization problems: the Rosenbrock problem. Mathematically, it is stated as:

$$\underset{x,\,y}{\text{minimize}}(1-x)^2+100(y-x^2)^2$$

In code:

In [6]:
```
def rosenbrock(x, y):
    return (1 - x) ** 2 + 100 * (y - x ** 2) ** 2
```

It's a good test case, because the minimum lies at the bottom of a shallow, curving valley:

The Rosenbrock Function

# AeroSandbox: Tools and Models

What else is in the box?

# Tools and Models: Overview

- Tools:
  - Geometry engine
  - Dynamics (time-dependence)
  - Surrogate modeling tools
- Models:
  - Aerodynamics
  - Propulsion
  - Power systems
  - Atmosphere and winds
  - Structures and weights
  - …
- Models are just a starting point
  - Not intended to be comprehensive
  - Designed to be easy to edit and extend

# Tools: Dynamics

- Capabilities
  - Solve ODEs, BVPs via direct collocation
  - Simulate dynamic systems
    - Fixed or variable time horizons
    - Multiphase dynamics (e.g. rocket boost and glide)
  - Optimal control
- Implement integrators in one line:
  - Forward/backwards Euler
  - Midpoint method ($2^{nd}$-order)
  - Runge-Kutta (in pre-release)
- Shorthands for 2D aircraft dynamics

Make an optimization problem

Create a time vector

Initialize position

Initialize velocity

Initialize acceleration

Governing eqn.: $F = ma$

Initial condition

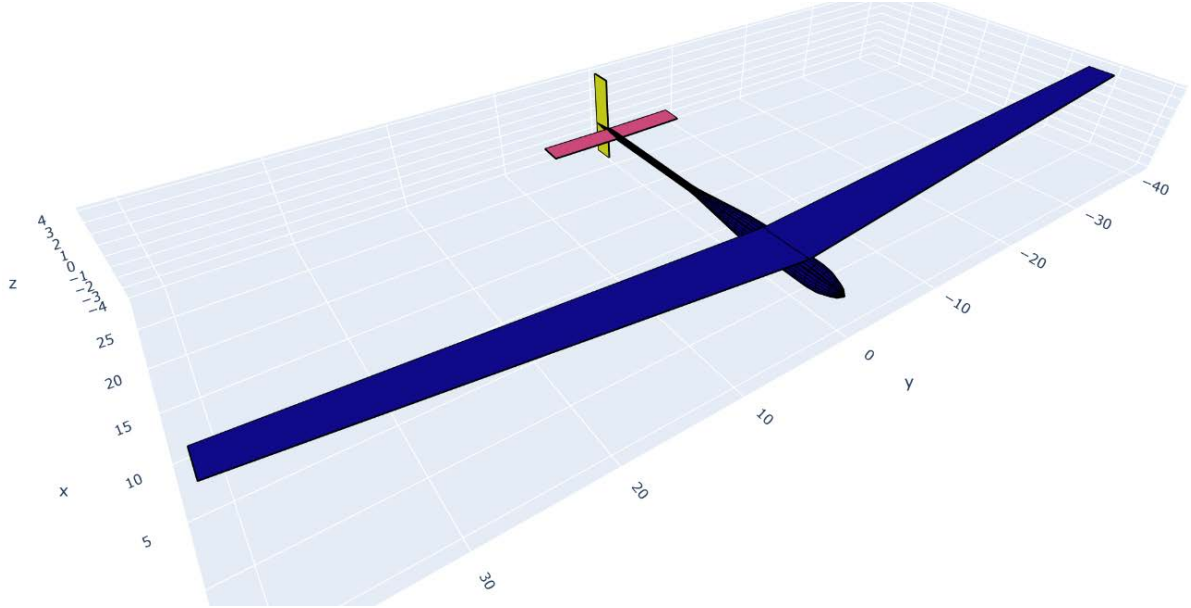
Initial condition

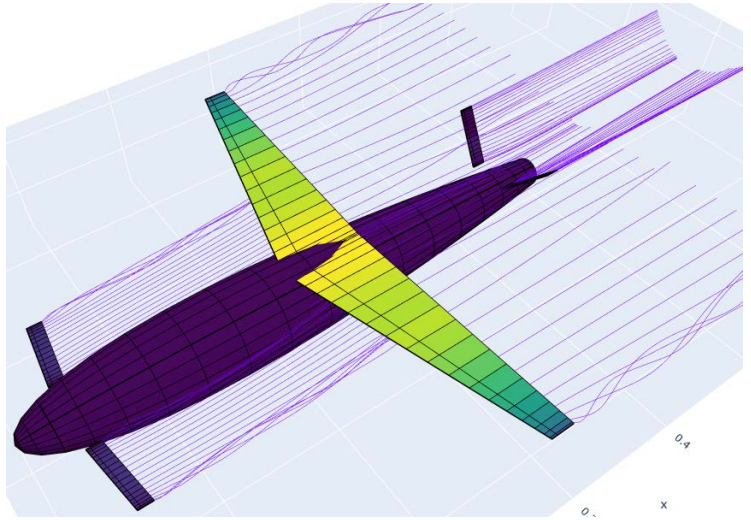
Solve!

```
opti = asb.Opti()

time = np.linspace(0, 1, 50)

position = opti.variable(
    init_guess=np.linspace(0, 1, 50)
)

velocity = opti.derivative_of(
    position, with_respect_to=time,
    method="runge-kutta"
)

accel = opti.derivative_of(
    velocity, with_respect_to=time,
    method="midpoint"
)

opti.subject_to([
    force(time) == mass * accel,
    position[0] == 0,
    velocity[0] == 0,
])

sol = opti.solve()
```

# Tools: Geometry Stack

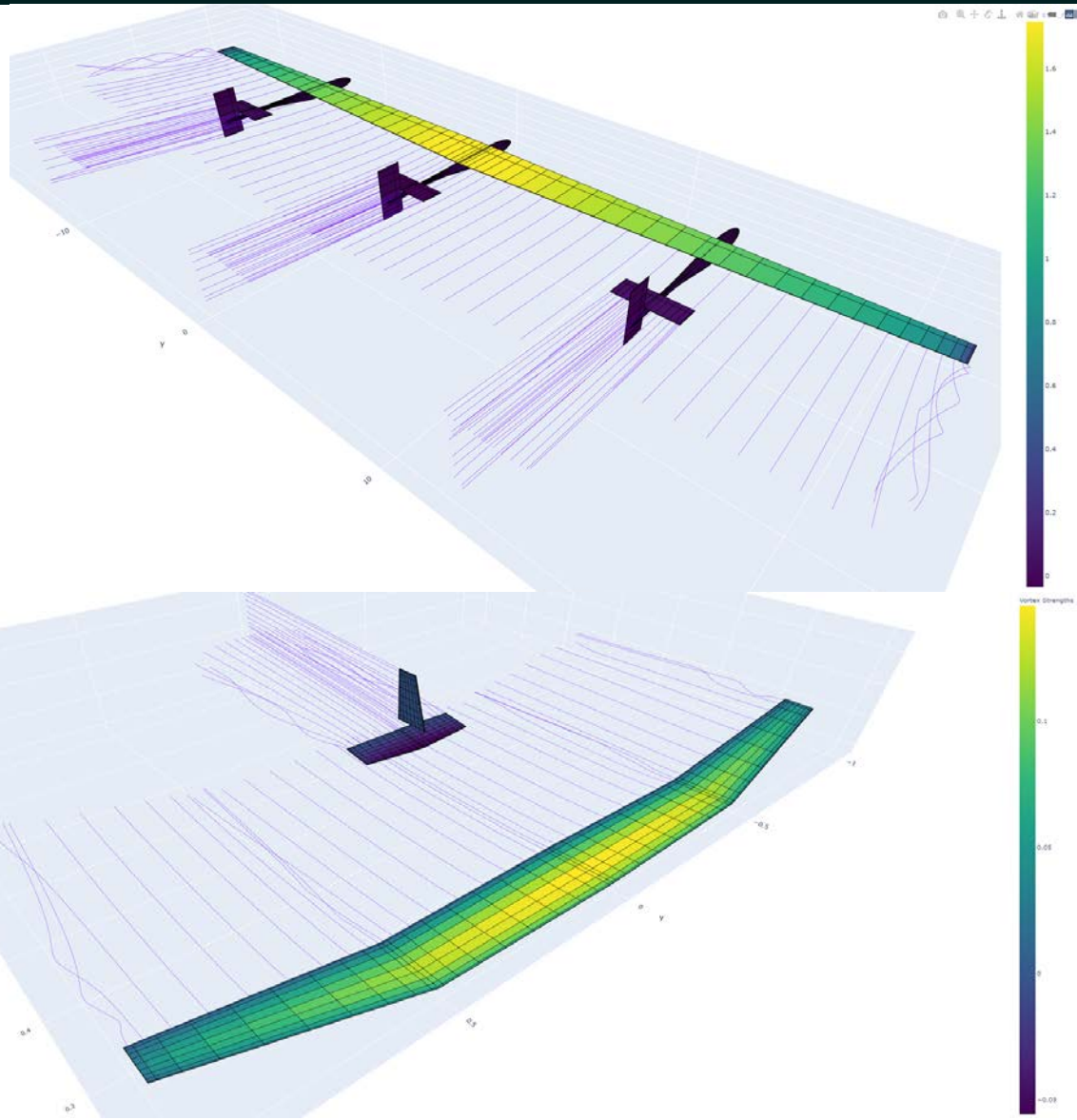
- Object-oriented geometry stack:
  - Airplanes, wings, fuselages, airfoils, custom polygons/polyhedral
  - Lofting, parameter calculation (volume, wetted area, etc.)
  - Thin surface or solid
  - 3D tri/quad surface meshing and visualization
  - File interfaces (*.dat, *.stl, *.avl, …)
- Differentiable – supplies derivatives to analysis tools



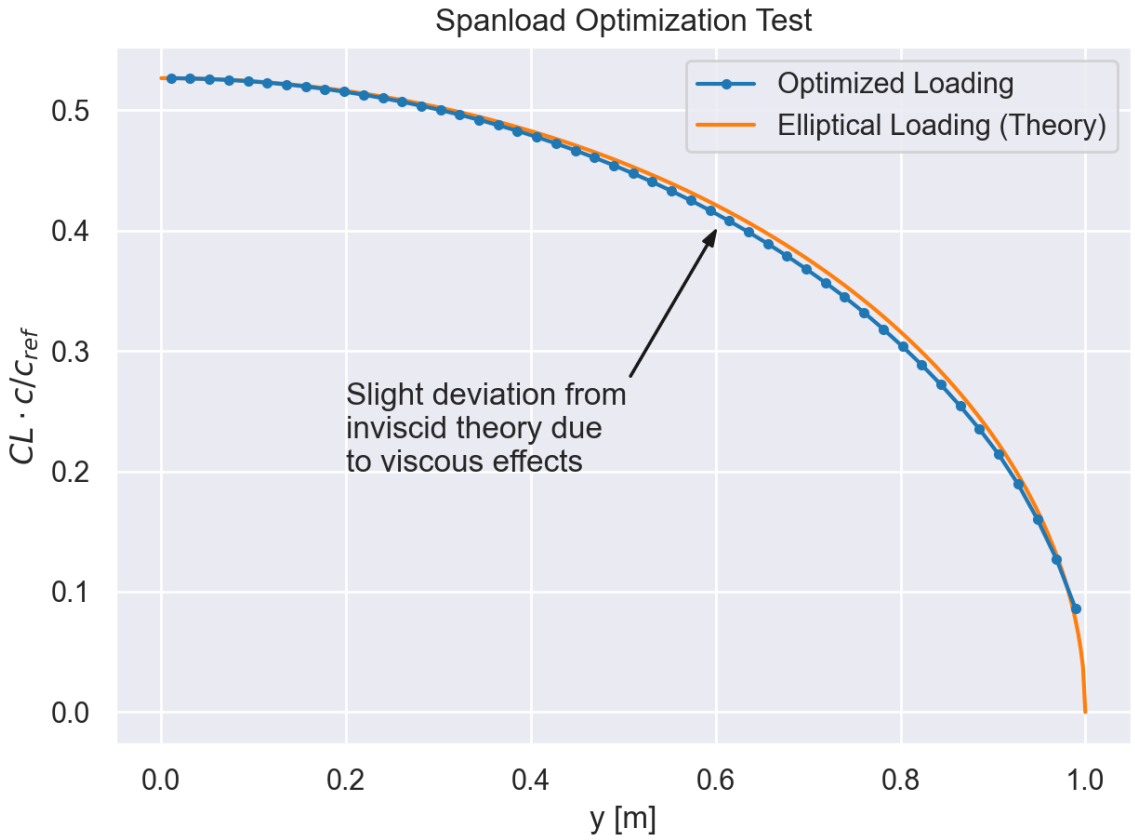
# Models: 3D Aerodynamics

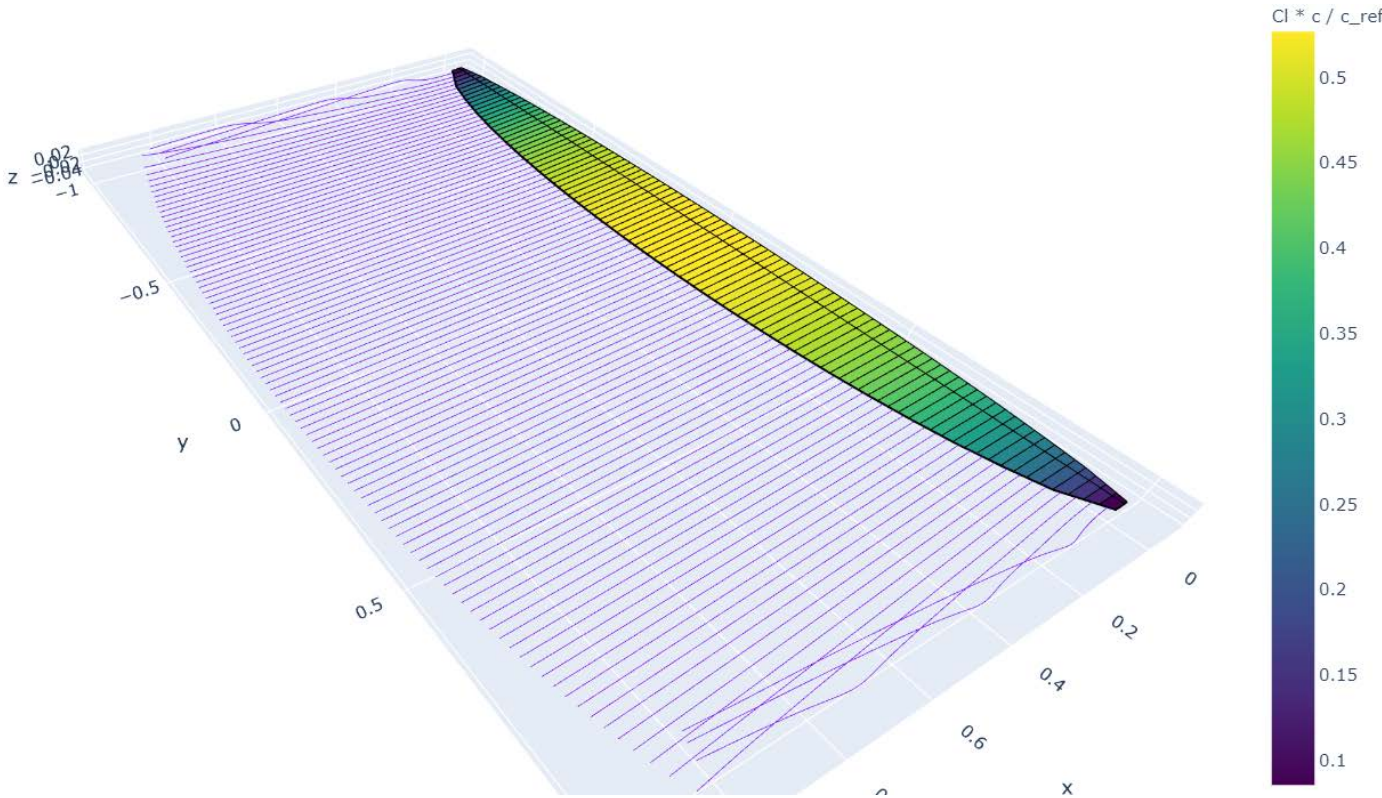
- Options:
  - Semi-empirical buildup
    - Similar to DATCOM
  - Fully nonlinear lifting-line method:
    - Coupled viscous effects
    - Local stall
    - Modern modifications:
      - Nonzero sweep or dihedral
      - Multiple surfaces, fuselages
  - Vortex-lattice model
- Dynamic problems: aerodynamics implicitly solved at each point on trajectory



# Models: 3D Aerodynamics

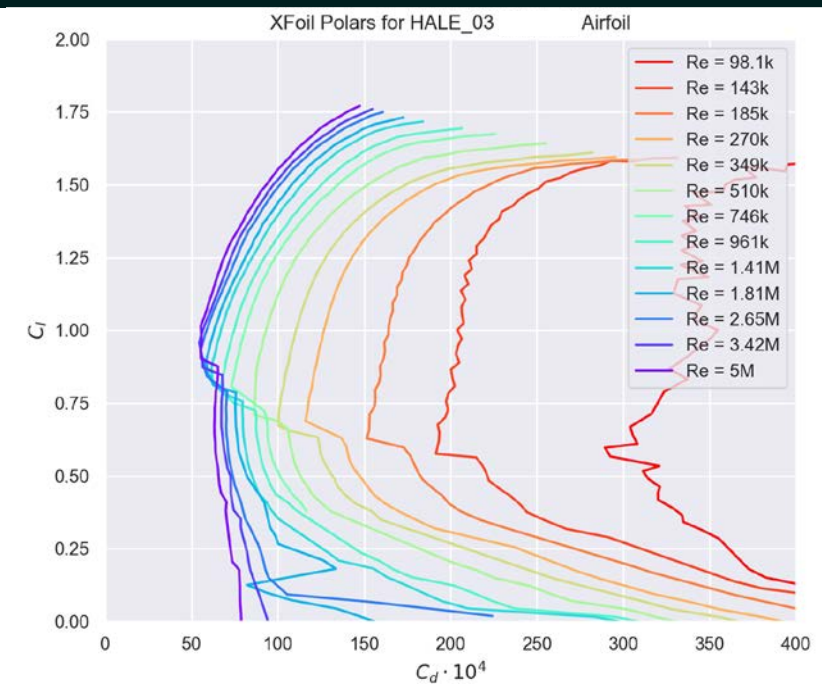
- All three are end-to-end differentiable; optimizer-friendly
- E.g.: Optimization of wing, with viscous effects

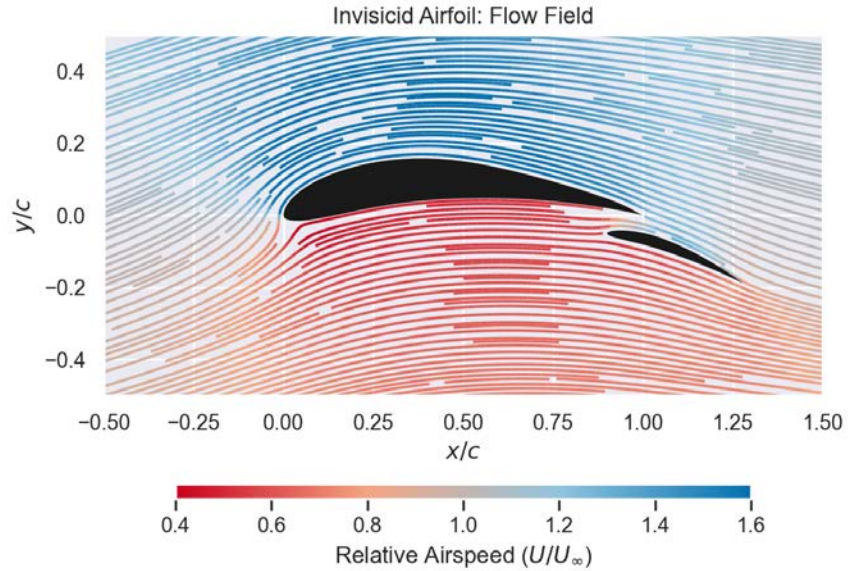


# Models: 2D Aerodynamics

- Options:
  - Interface to XFoil (not differentiable)
    - Useful for data retrieval, surrogate modeling
  - Uncoupled inviscid-viscous IBL model
    - Multielement, ground effect
    - Identical IBL implementation to XFoil
  - In progress: differentiable coupled IBL model (same as XFoil)
    - Will allow airfoil shape optimization coupled with vehicle design, trajectory

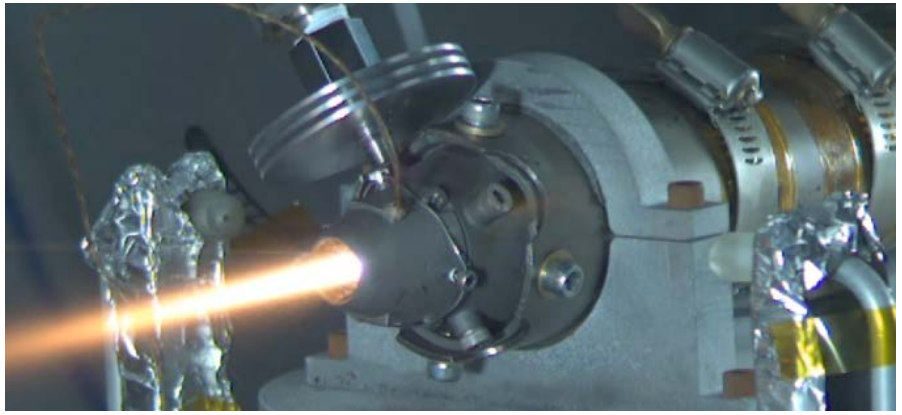


# Models: Propulsion and Power Systems

- Propulsion
  - Propeller
    - Disc-actuator model with viscous modifications
    - Differentiable blade-element model (same as QProp)
  - Rocket
    - Isentropic relations with empirical corrections for nozzle flow
  - Turbomachinery (WIP)
- Power Systems
  - Solid rocket propellants
    - Burn rate suppressants (oxamide)
  - Batteries, solar cells
  - Avgas, Jet A (WIP)

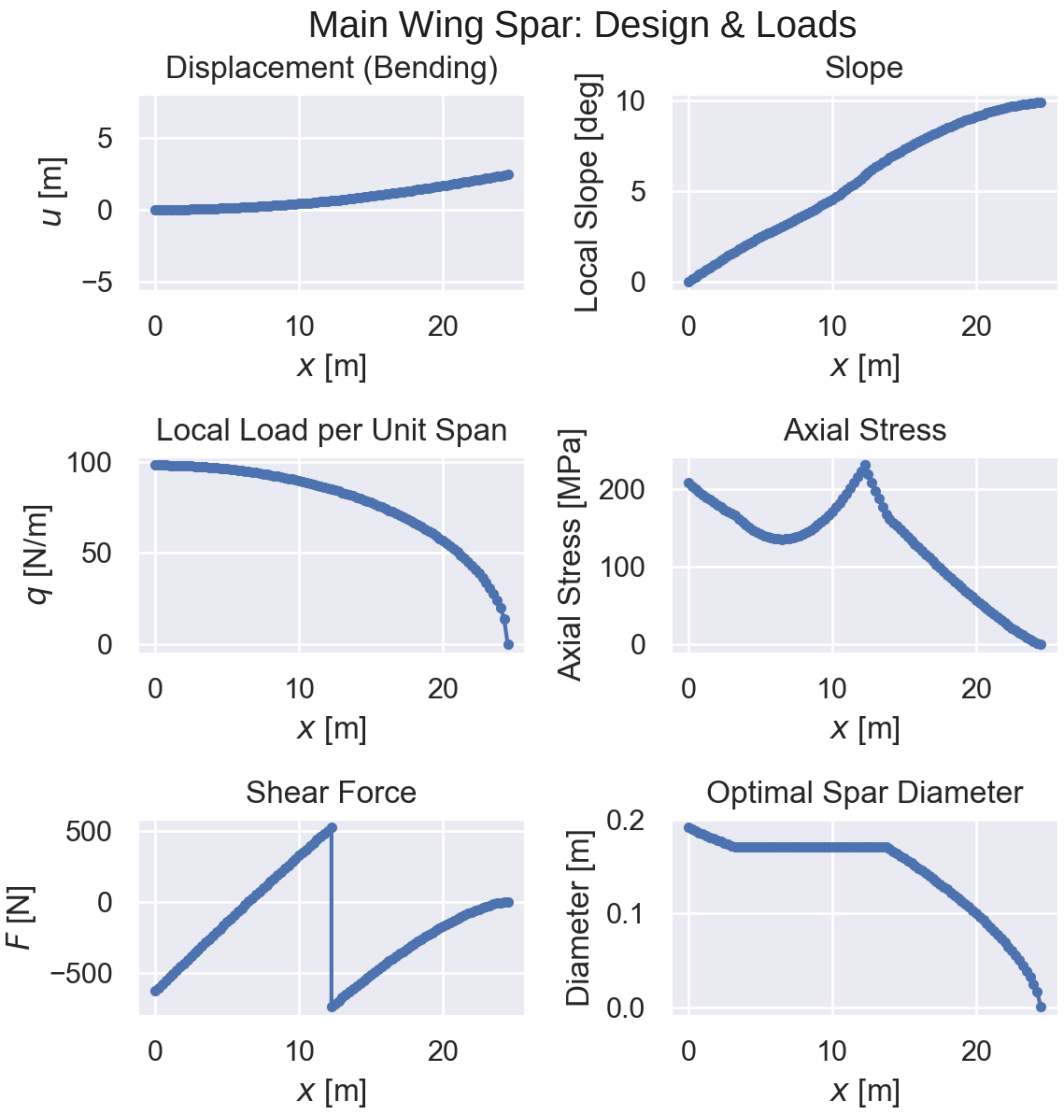

Electra.aero UAM demonstrator: a development program using AeroSandbox for takeoff performance analysis



Firefly slow-burn-motor static fire at MIT: data used to calibrate AeroSandbox's rocket performance library

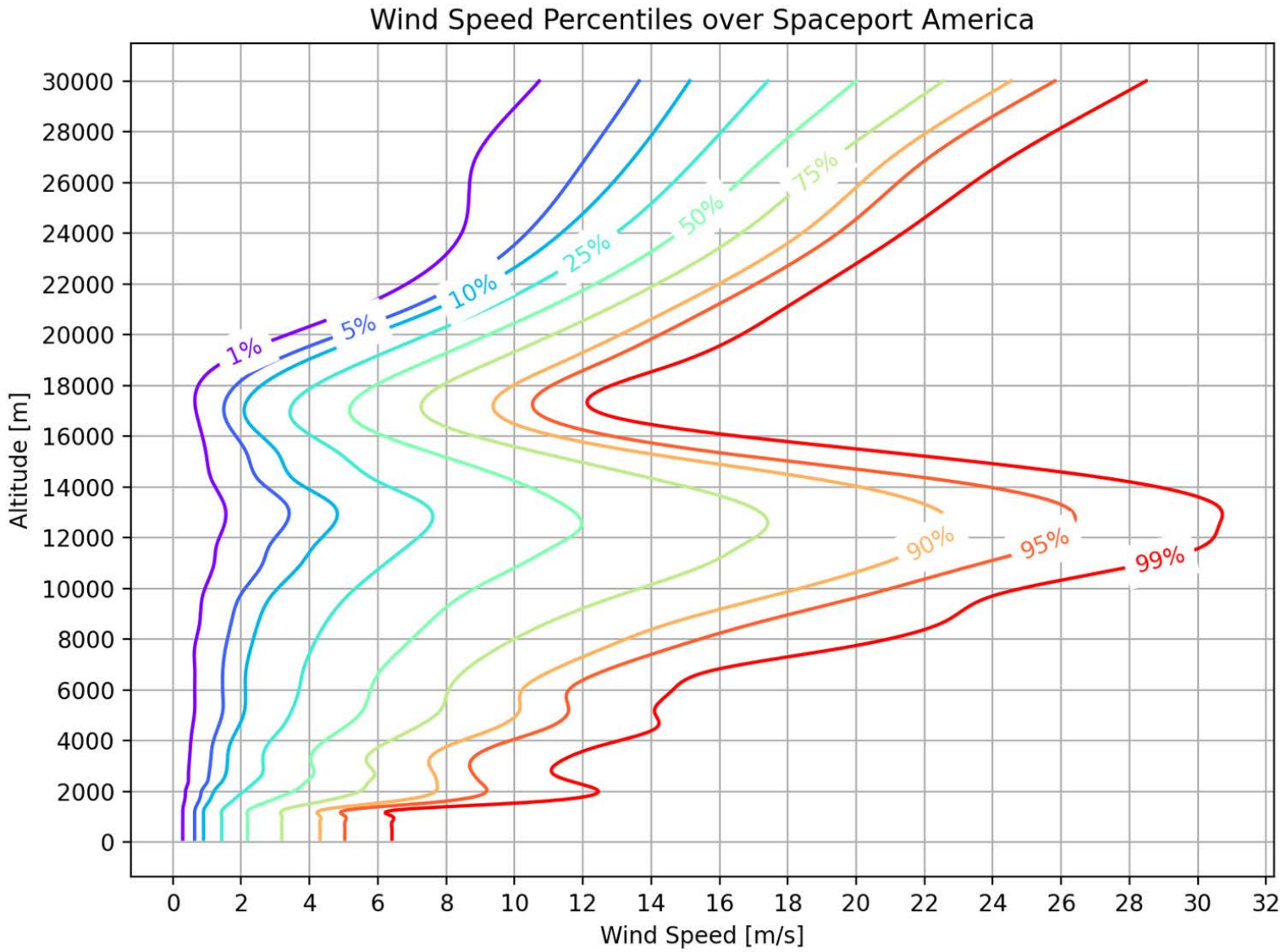
# Models: Structures & Weights

- Structures
  - 6-DoF Euler-Bernoulli beam model
    - Bending, torsion
  - Statistical models based on existing aircraft
- Weights
  - Component-wise CG buildup
    - Moments of inertia
    - Stability derivatives easily computed via AD
      - Derivatives of stability derivatives
- Stability options:
  - Static margin
  - Constrain stability derivatives
  - Constrain poles
    - Optimizing through an eigenvalue decomposition!



# Models: Atmosphere and Winds

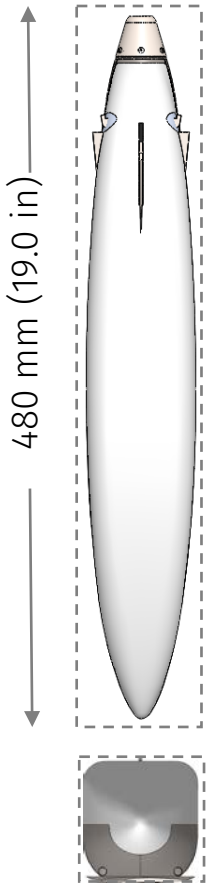
- Atmosphere
  - Differentiable atmosphere enabling trajectory optimization
- Wind speed modeling
  - Statistical model
  - By using probabilistic models, risk can be your objective function
    - Enables robust design



# Application: Firefly

A Mach 0.8 rocket-propelled micro-UAV

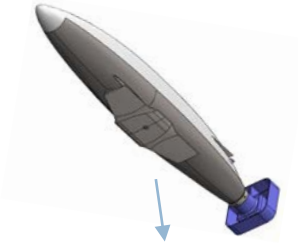
# Firefly: Requirements and Configuration



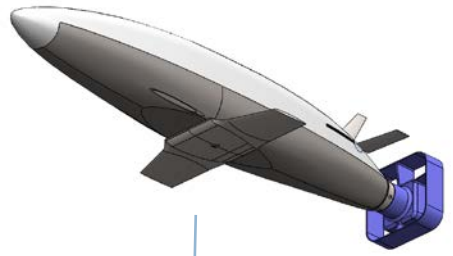

Requirements:

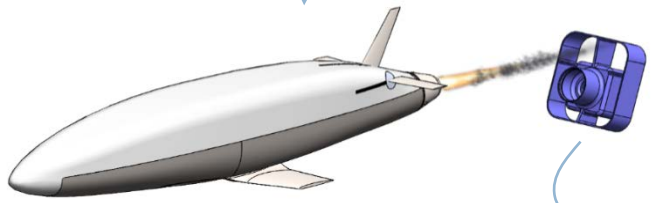
- Mach 0.8 powered flight + glide
- Size < 70 x 70 x 480 mm
- Mass < 2 kg
- Maximize total range

Drop



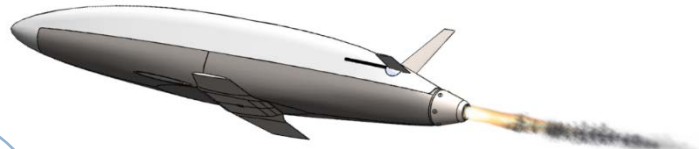
Unfold and stabilize



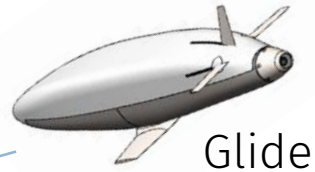


Ignite rocket

Discard stabilizer



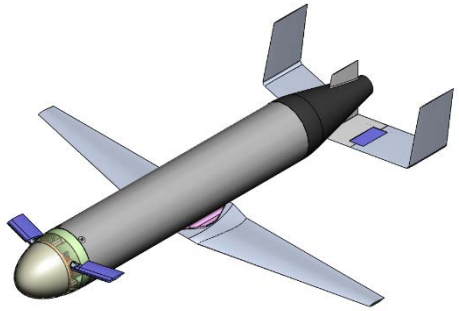
Powered flight
(~2 minutes)



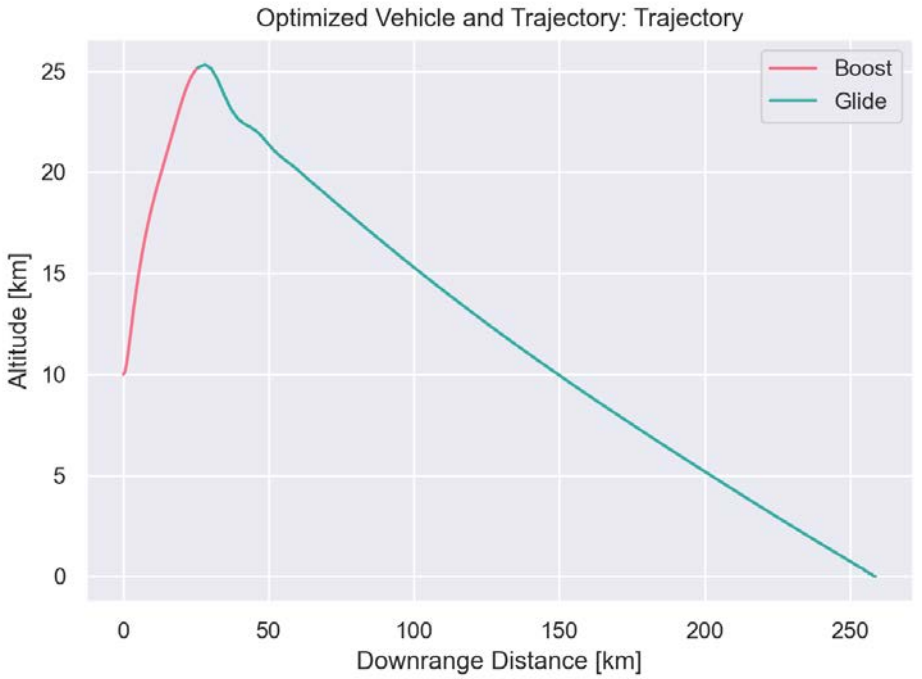
Glide
(4-6 minutes)

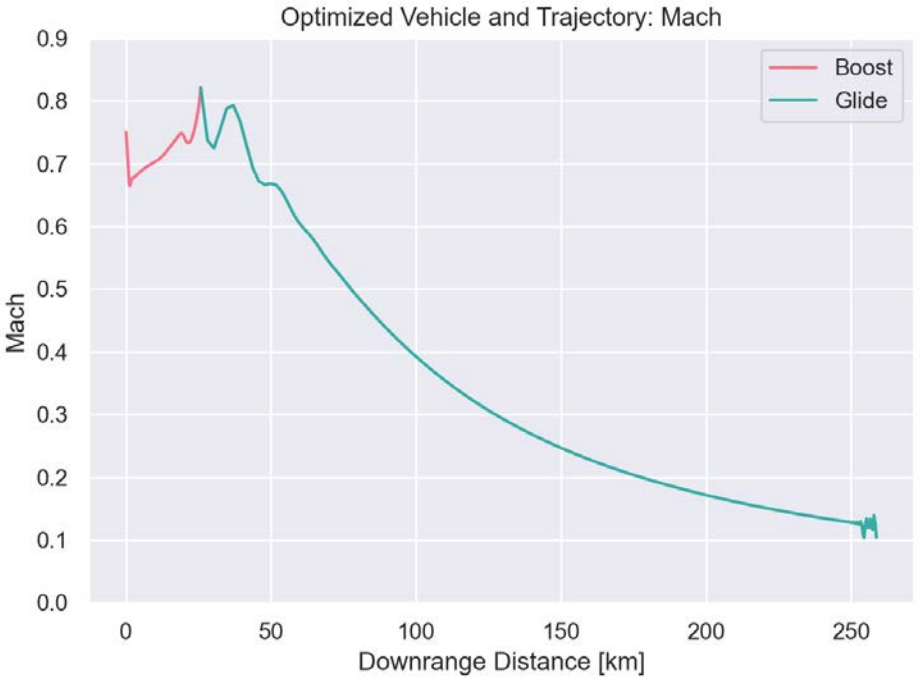
Disposal

# Firefly: Range Optimization



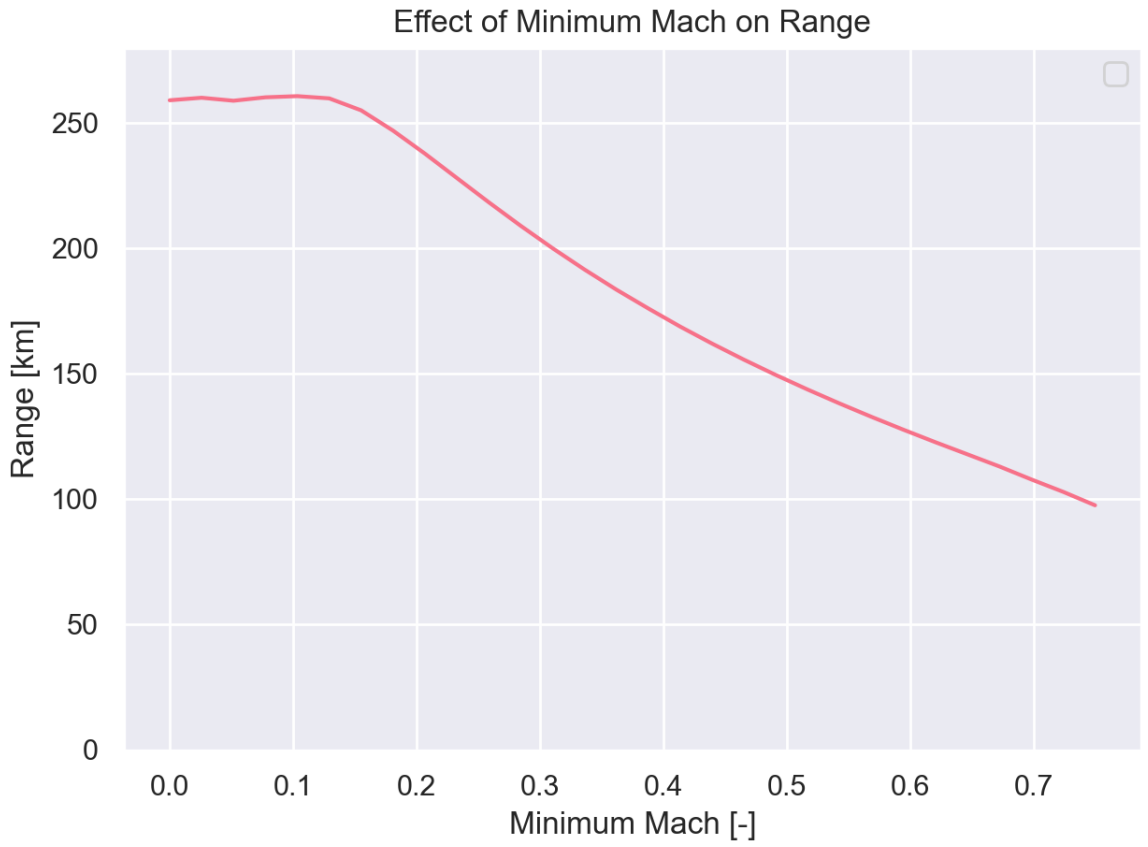
- Assumptions
  - Launch at 10 km, Mach 0.75
  - Maximizing total range
  - Simultaneously optimizing vehicle design and mission strategy
- Results:
  - Range: 258 km
  - “Boost-glide” trajectory
  - Vehicle becomes a rocket-propelled glider
    - Very large lifting surfaces: 426 mm wingspan
    - High boost airspeed, but speed drops precipitously during glide ($L/D_{max}$)



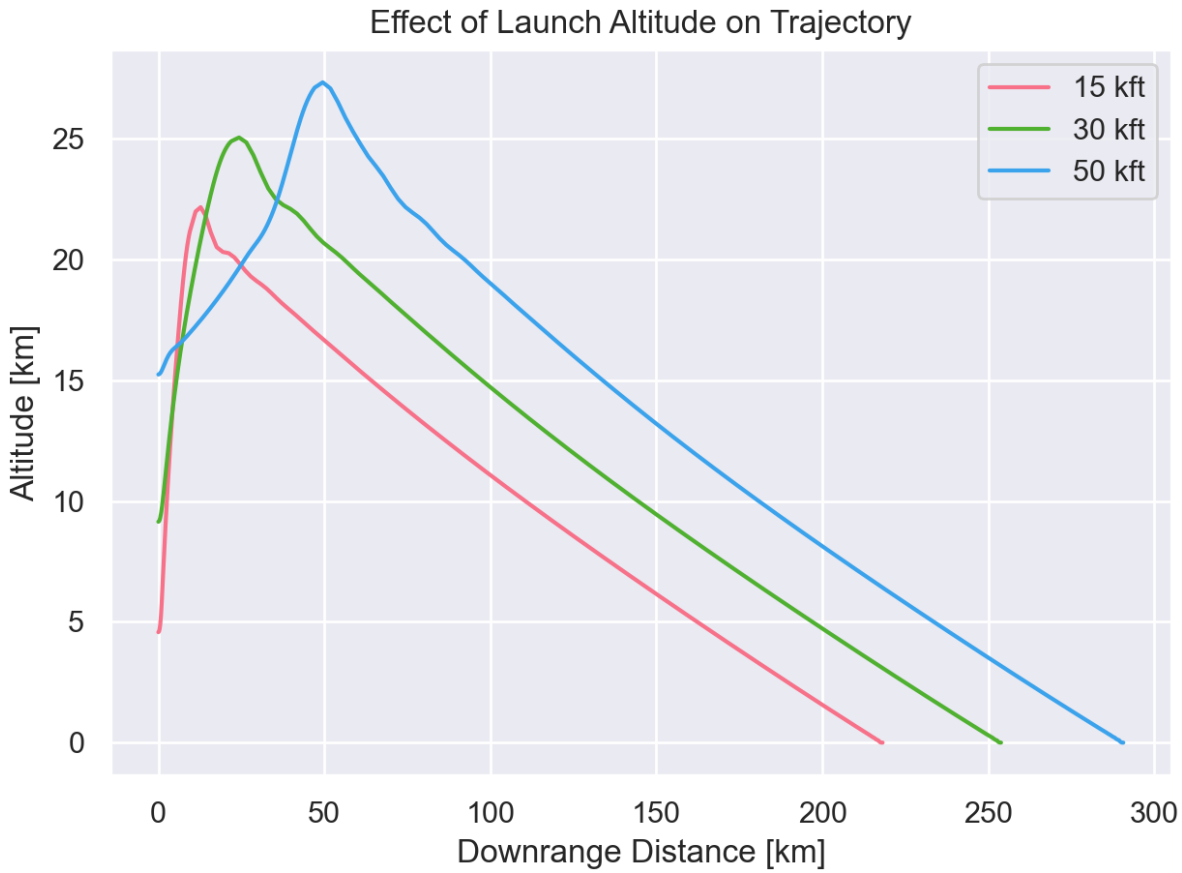
# Firefly: Understanding the Design Space

- Observation: Glide speed is very low ($M \approx 0.1$) near end of flight
- How would total range change if we enforced a minimum glide speed?
  - Initially, no change (loose constraint)
  - At $M \approx 0.12$, constraint becomes tight and starts reducing range
  - Glide performance almost zero for $M \geq 0.75$ case.
    - Vehicle exploits qualities favorable to boost performance



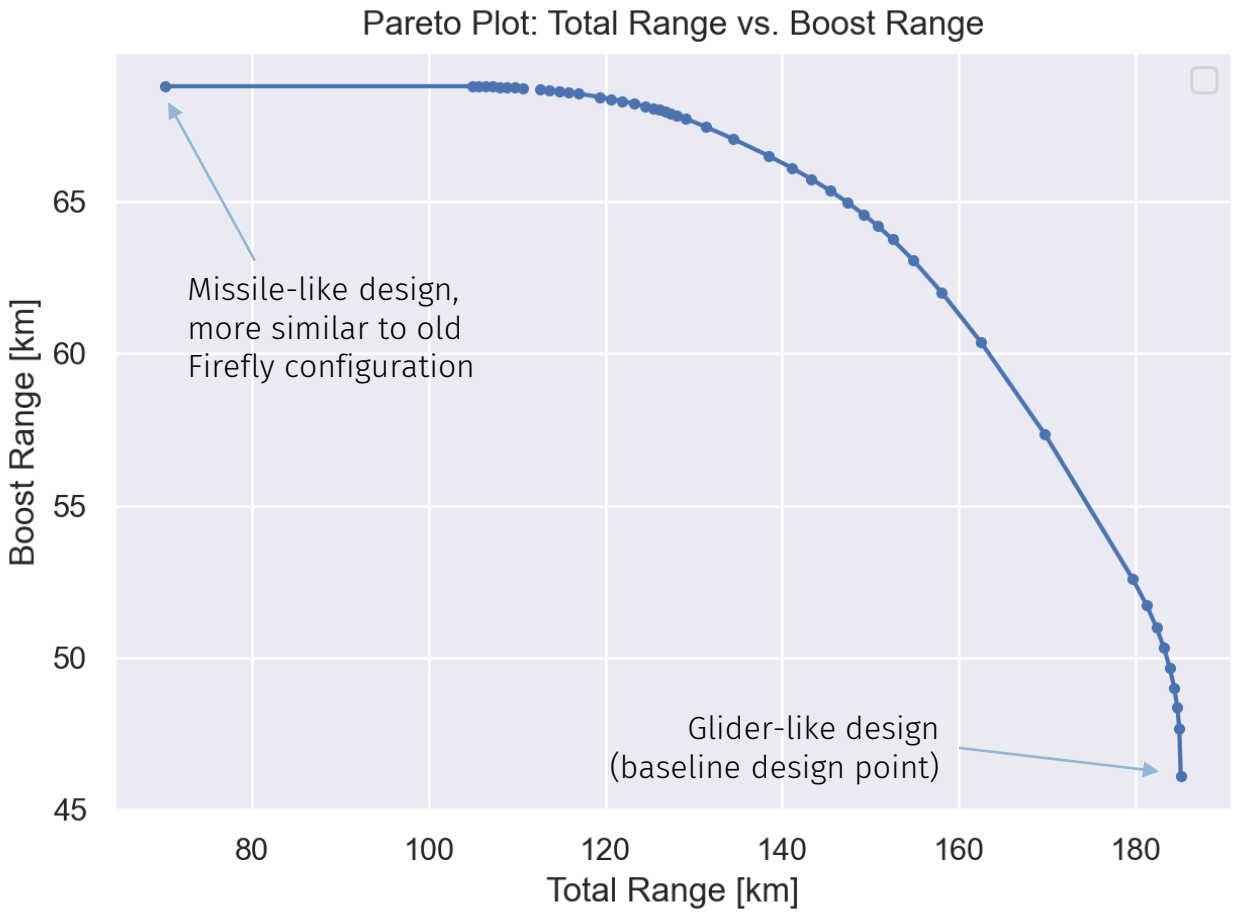
# Firefly: Effect of Launch Altitude

- Effect of varying launch altitude
  - 15 kft launch: 218 km range
  - 30 kft launch: 254 km range
  - 50 kft launch: 291 km range
- Sensitivity of range w.r.t. launch altitude: 6.8 meters/meter
  - Optimizer is directly manipulating control surface deflections to fly trajectory
  - Sensitivity of an optimal control problem



# Firefly: Multi-objective Optimization and Pareto Efficiency

- Tools to allow multi-objective optimization
  - E.g. trade-off between total range and boost range
- Pareto frontier
  - Each point represents a unique airplane and trajectory
- Shows where large gains can be found in one metric in exchange for minor setbacks in another




*Note: figure based on outdated assumptions, for illustration only

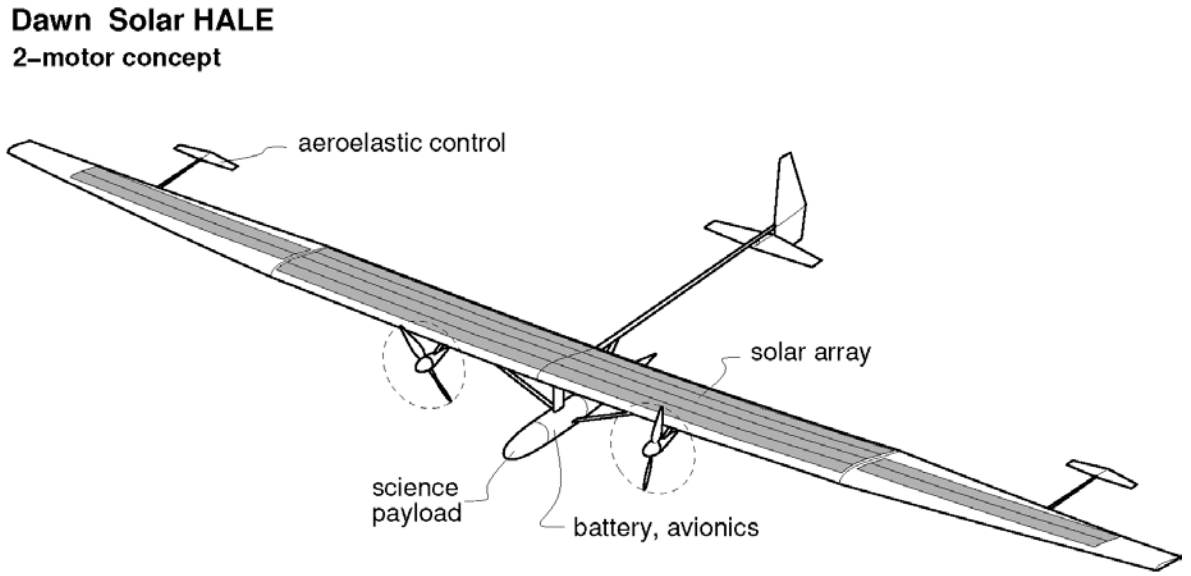
# Application: Solar Airplane (“Dawn”)

A stratospheric airplane supporting atmospheric science research

# Dawn: Requirements and Configuration

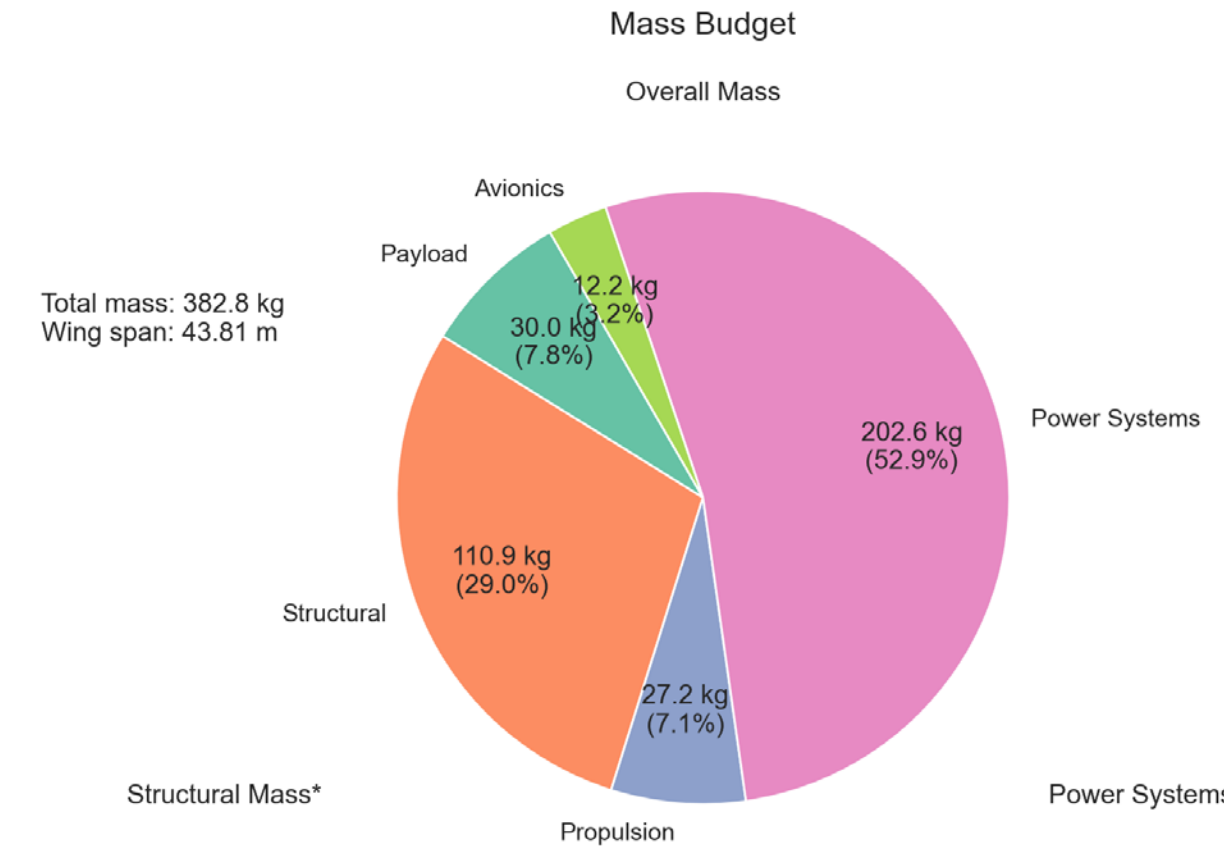
Requirements:

- Unmanned platform
- Payload:
  - 30 kg
  - Power: 500 W day, 150 W night
- Mission:
  - 6 weeks continuous flight
  - July – Aug., continental U.S. (CONUS)
  - 60,000 ft. (18.3 km) minimum altitude
  - Ability to relocate on command, stationkeep against winds
  - Survive gusts during ascent and thunderstorm overflight




# Dawn: Mass Budgets

- Component-wise mass buildup
  - Visualization tools to increase interpretability
- By default, entire airplane and trajectory designed simultaneously
- Selective freezing of design variables as subsystems become more “locked in”
  - Per-variable or by user-specified categories
  - Enables off-design performance analysis



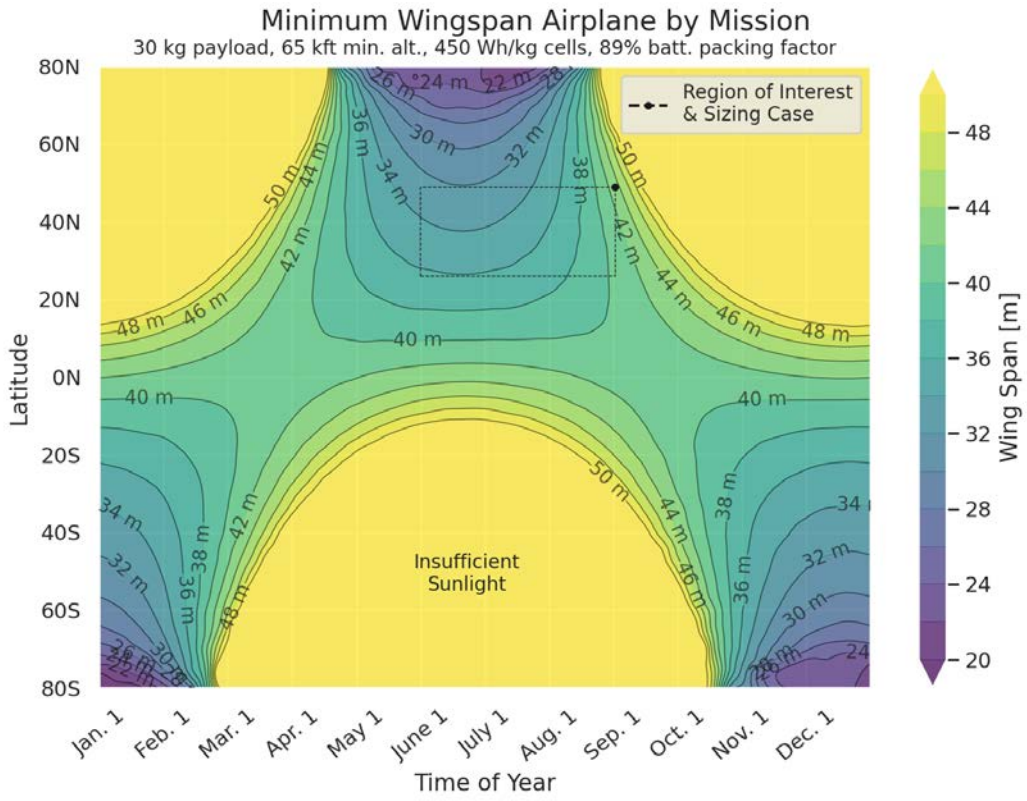

* percentages referenced to total aircraft mass

# Dawn: Carpet Plots and Envelope Exploration

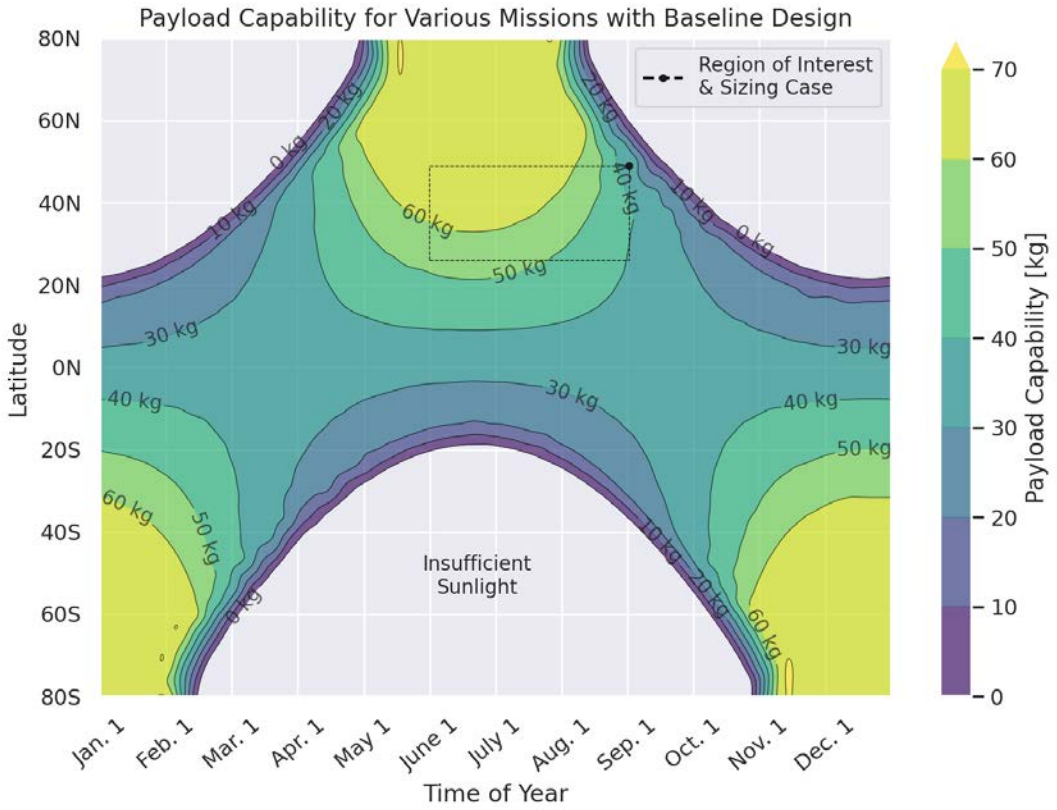
Optimization speed enables sweep studies on mission parameters

- Plots are the result of 3000 optimization runs, each with ~4000 variables (~30 min. total runtime on laptop, serial)

## Fixed assumptions, rubber airplane

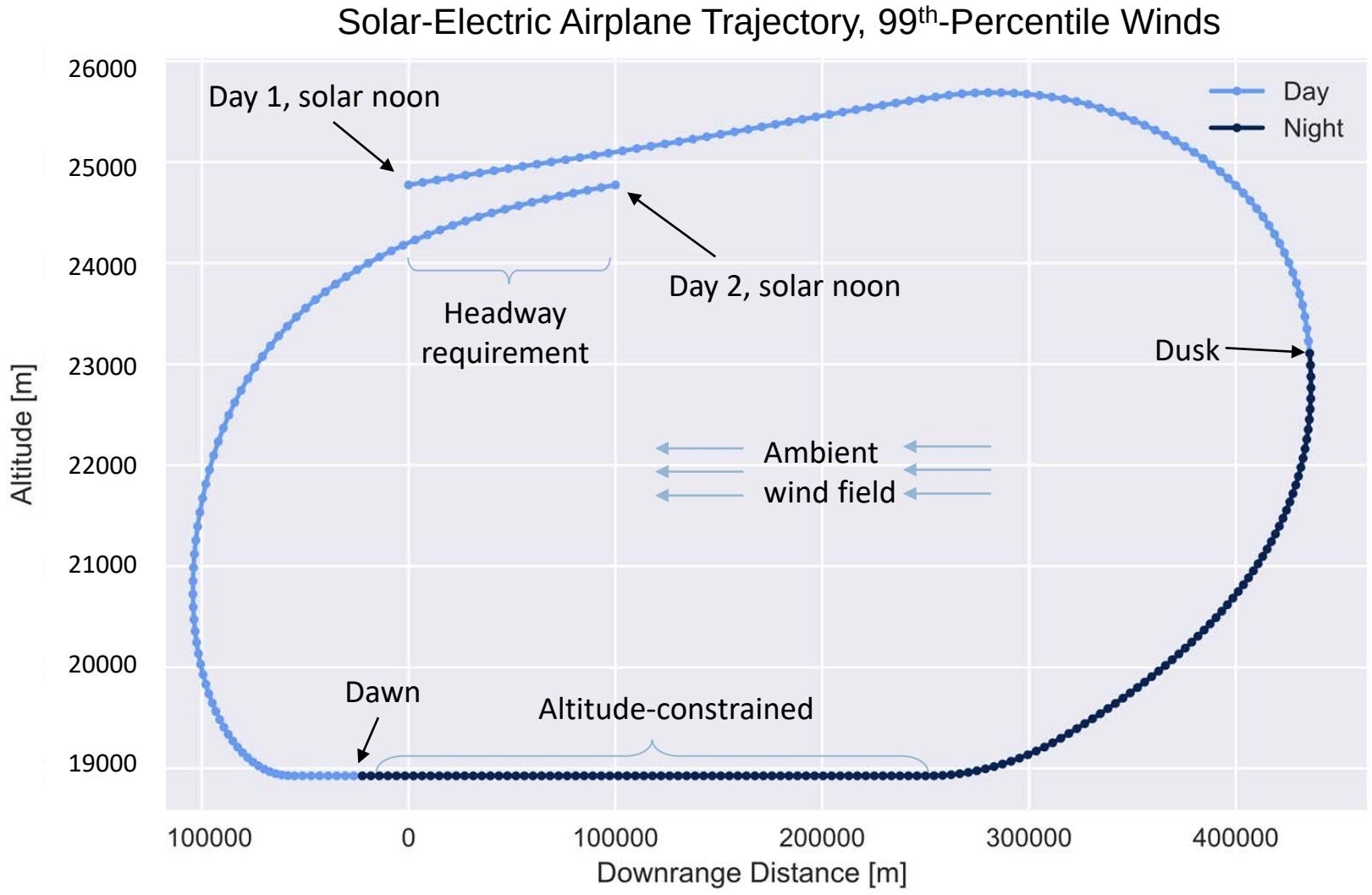



## Rubber assumptions, fixed airplane



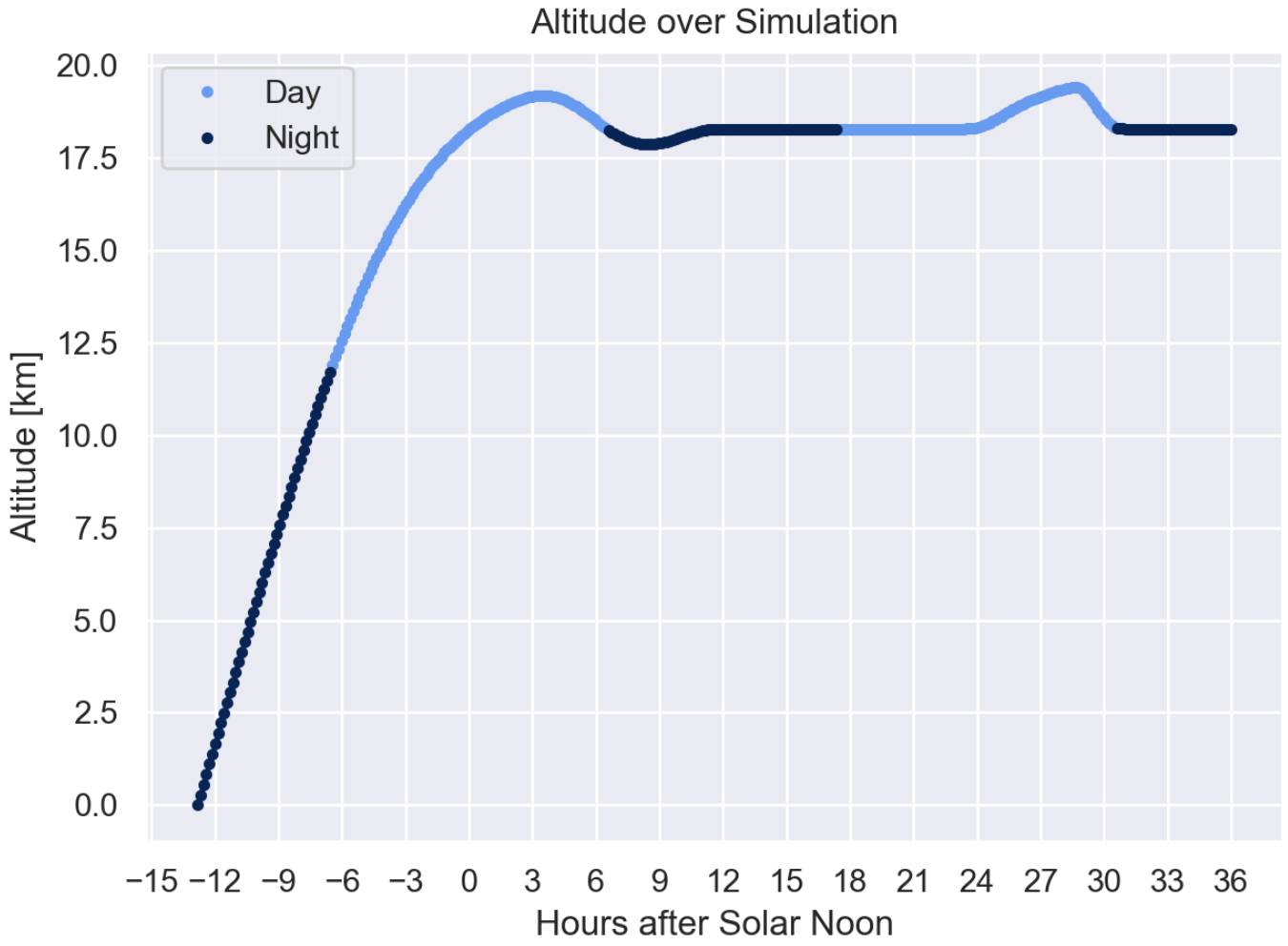
# Dawn: Cruise Dynamics

- Diurnal power injection leads to unsteady optimal trajectories
  - Altitude cycling
  - Airspeed cycling
- Initialized to steady flight
  - AeroSandbox finds unintuitive optimal control solutions
- Simultaneously designing airplane and trajectory



# Dawn: Ascent Dynamics

- Goal:
  - Find minimum-energy ascent profile
  - Find optimal takeoff time relative to solar noon
    - Precise timing required
    - Gust, cloud cover constraints
- Results:
  - Optimal takeoff at ~11 p.m., relative to solar noon
  - Steep initial ascent followed by shallow entrance to diurnal cycle



# Design Sensitivities

$$\frac{\partial(\text{wingspan})}{\partial(\text{variable})}$$

- Design sensitivities obtained "for free"
- Inform subsystem-level decisions, e.g.:
  - Motor efficiency vs. motor mass
  - Marginal benefit of dollars spent on advanced technologies
  - Identifying active and inactive constraints in requirements
    - What's limiting the design?

| Figure of Merit | Sensitivities of Minimum Achievable Wingspan |
|---|---|
| Any added mass | 0.303 m/kg |
| Any added power draw* | 0.0172 m/W |
| Any added drag | 0.705 m/N |
| Battery spec. energy | -0.110 m/(Wh/kg) |
| Battery packing fraction** | -0.546 m/(%) |
| Solar cell efficiency** | -0.984 m/(%) |
| Solar cell area density | 18.4 m/(kg/m^2) |
| Propeller COP** | -0.819 m/(%) |
| Motor Efficiency** | -0.773 m/(%) |
| Nighttime Altitude Req. | 4.13 m/km |

* Assumed constant over 24 hours
** Derivatives w.r.t. percentages are in terms of absolute percentage points (i.e. 25% → 26%), not multiplied percentages

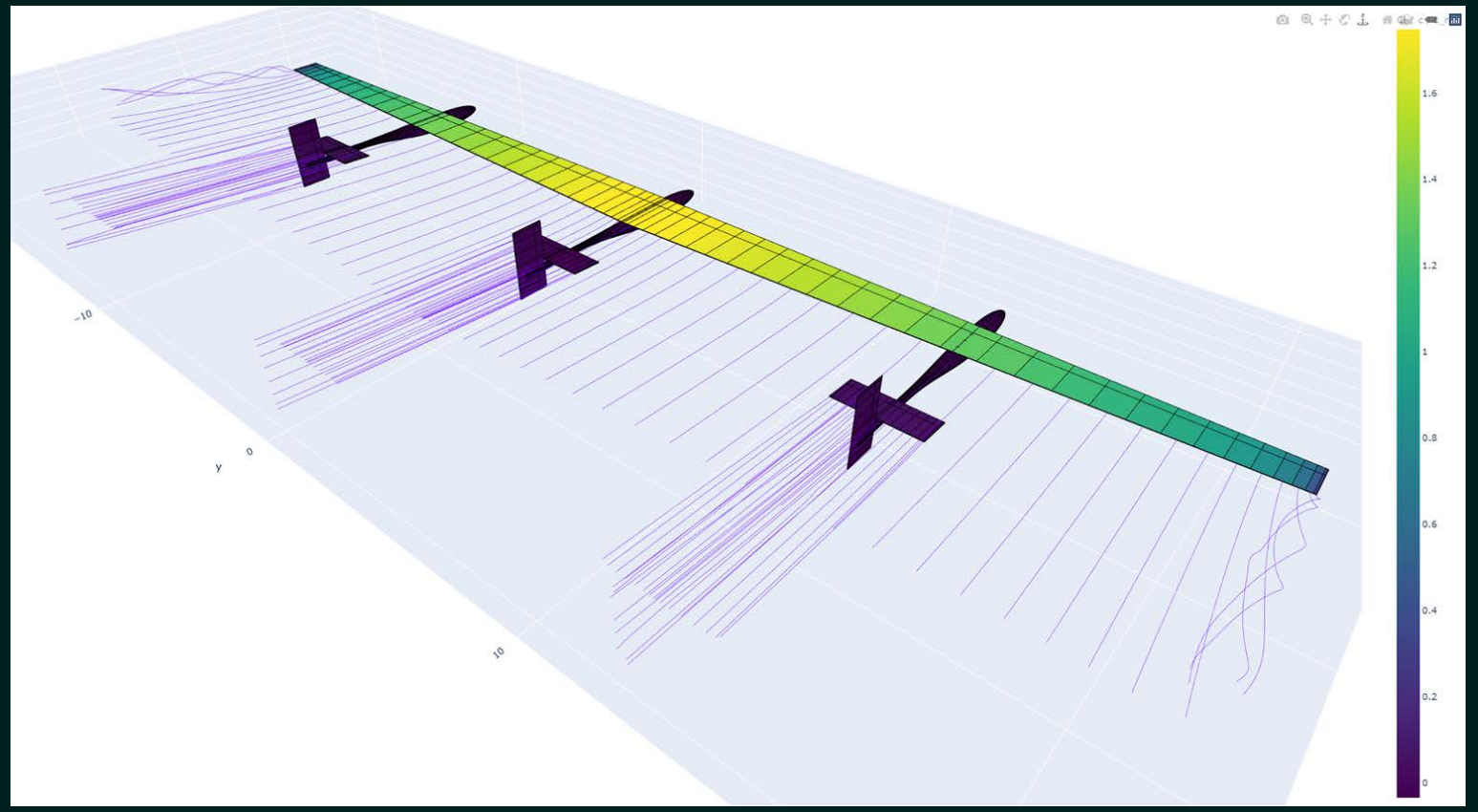
# Final Notes: Optimization Caveats

- Engineering time is often (implicitly) part of your objective function
  - 80% of the results come from 20% of the work – low-fidelity models go exceptionally far
  - Identify sensitive models and assumptions, then focus on refining those
- Do not blindly trust an optimizer
  - An optimizer only solves the problem you give it
    - Often, we forget constraints that seem "intuitive"
    - If any flaw exists in your model, an optimizer will exploit it – garbage in, garbage out
  - Without adding *margin*, optimized designs are almost always fragile
    - An optimizer will often naturally drive to the edge of the feasible space
      - In nature, optima are usually not near extremes
  - Make sure your optimized designs pass the TLAR ("That looks about right")
- 90% of design is asking the right question

# Final Notes: Key Take-Aways

- AeroSandbox is a differentiable framework for engineering design optimization
  - Includes lots of useful aerospace models, but the real paradigm shift is the optimization framework
- AD and SAND improve optimization speed by orders of magnitude
  - Fast enough to support:
    - Interactive design
    - Real-time design discussions between subteams
    - Complex design and optimal control problems
  - Numerics abstracted; designed to “just work”
- Tested in numerous ongoing aircraft development programs

# Thank you and happy optimizing!




https://github.com/peterdsharpe/AeroSandbox