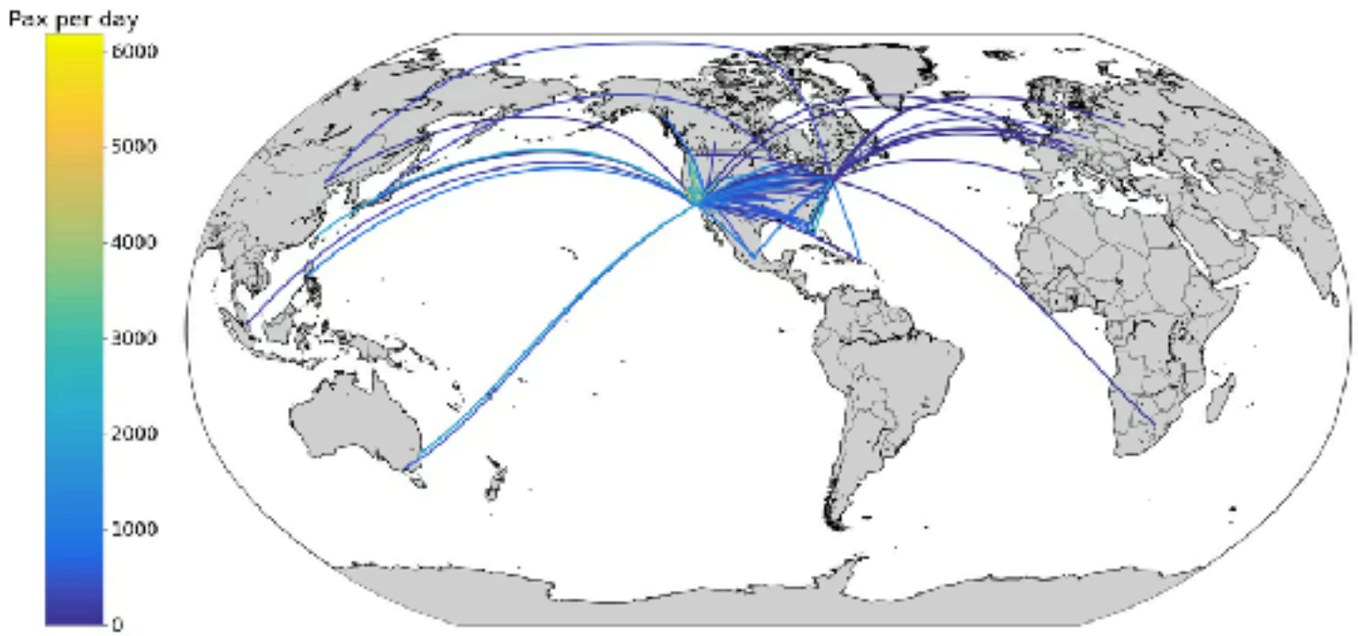
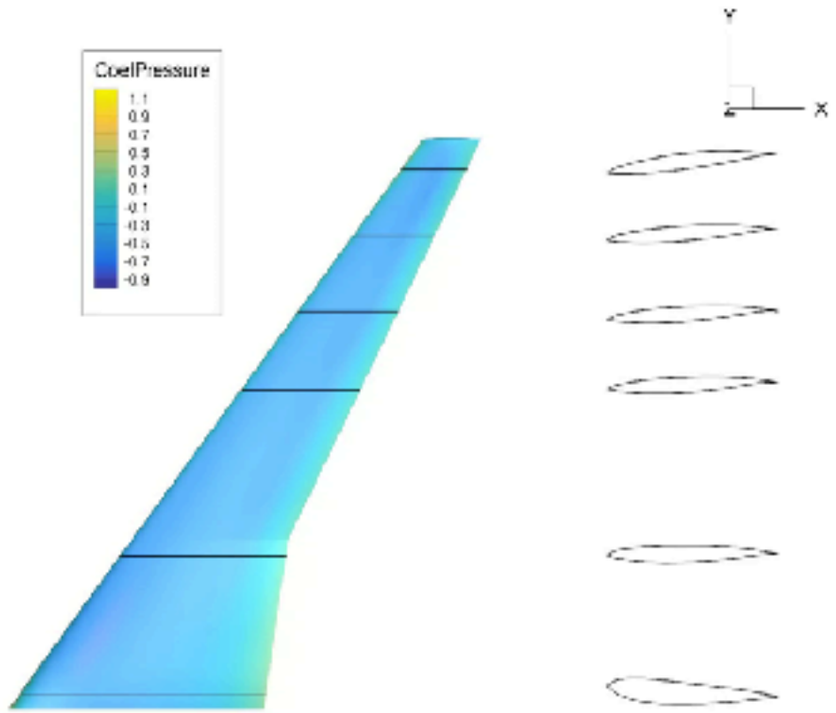
Overview of the MAUD architecture & new applications of OpenMDAO v2





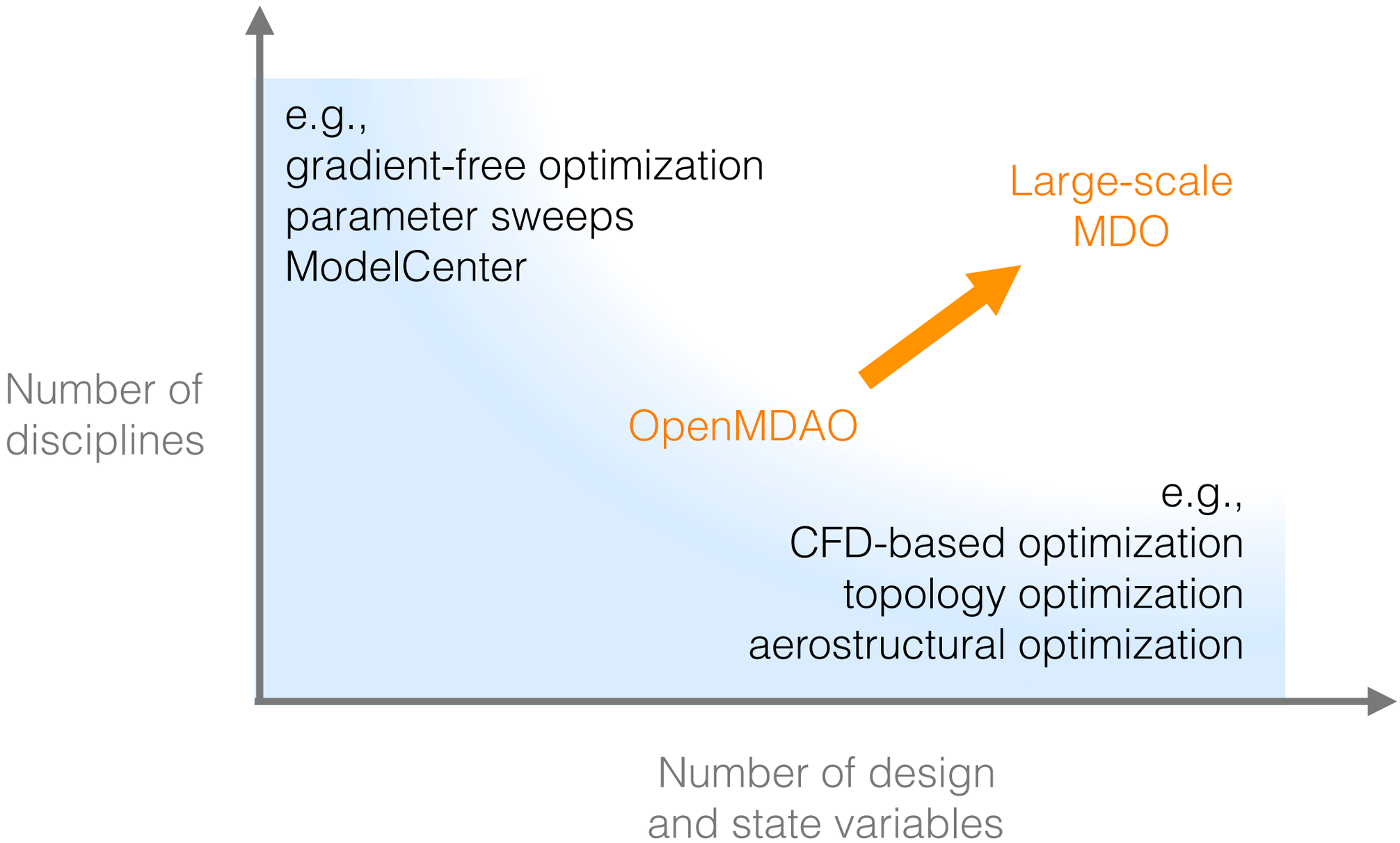
John T Hwang

First European OpenMDAO Workshop
ONERA • October 12, 2017

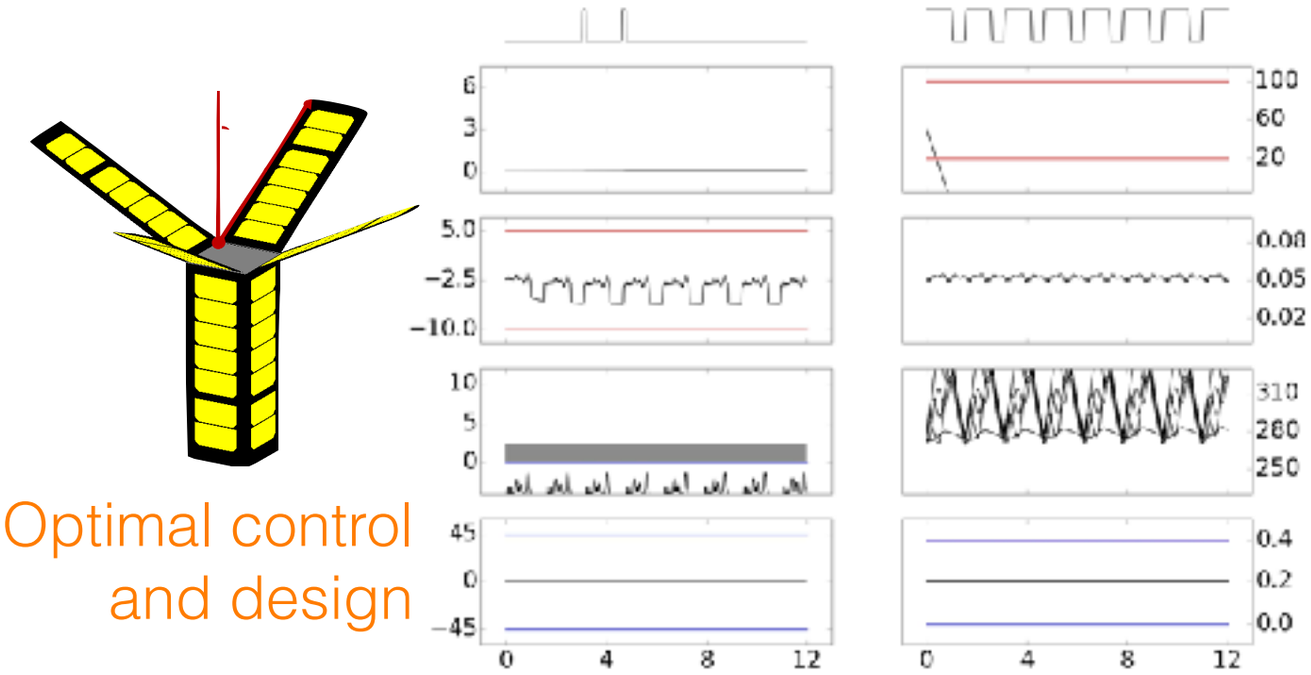
Several new MDO applications have been recently developed in OpenMDAO

- Nanosatellite MDO 2014
- Aircraft design-allocation optimization 2016
- Aircraft trajectory-propulsion optimization 2017
- Electric aircraft MDO 2018
- Aerostructural optimization with morphing 2018
- Topology optimization 2018

The common thread in these examples is large-scale optimization

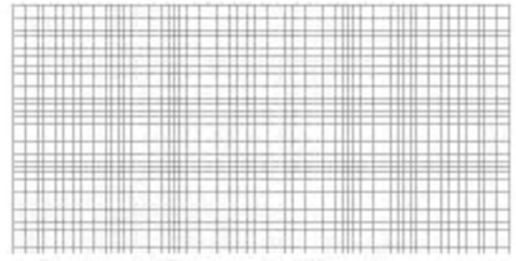


The ability to solve large-scale problems affords significant flexibility

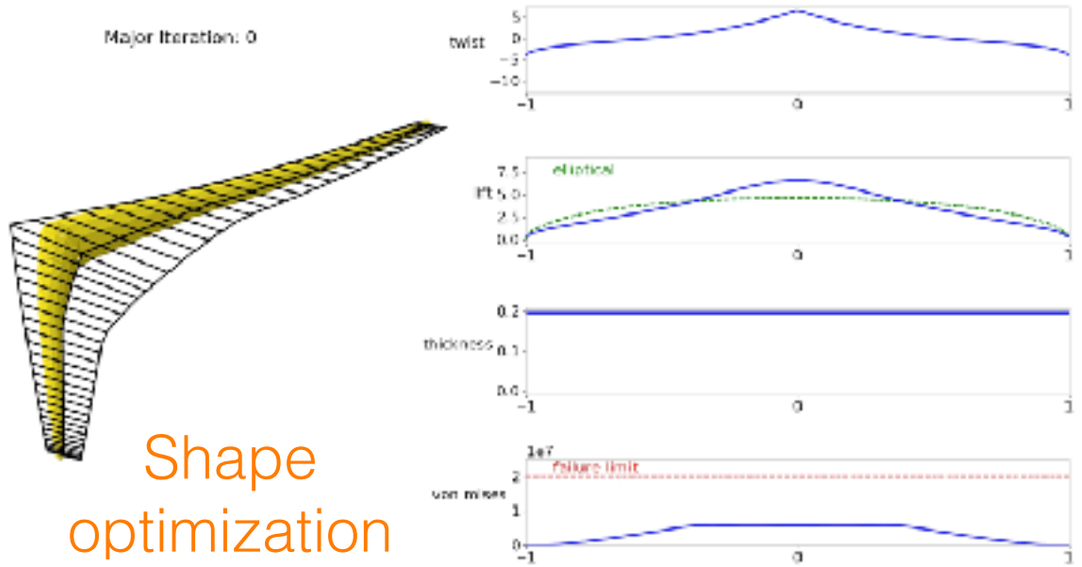


Optimal control and design

Topology optimization



Design-operations optimization

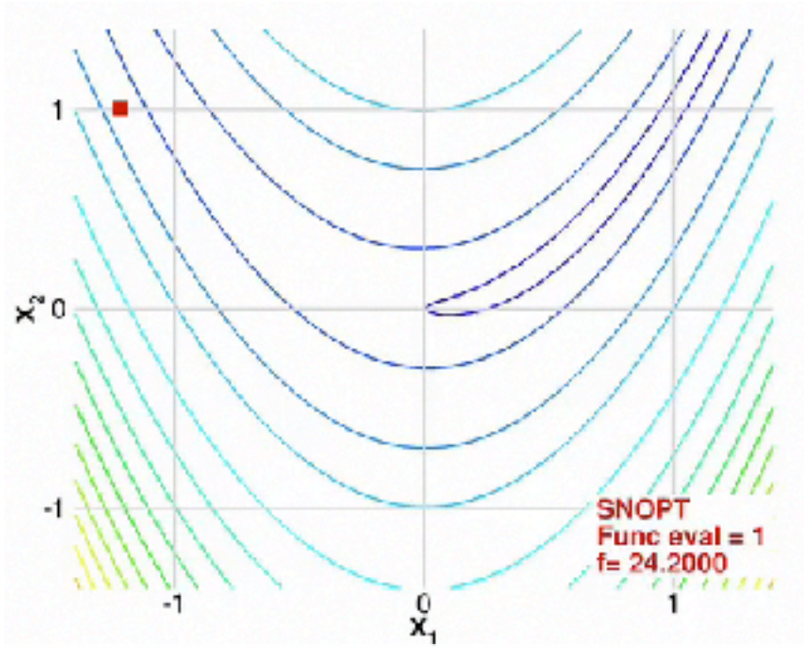


Shape optimization

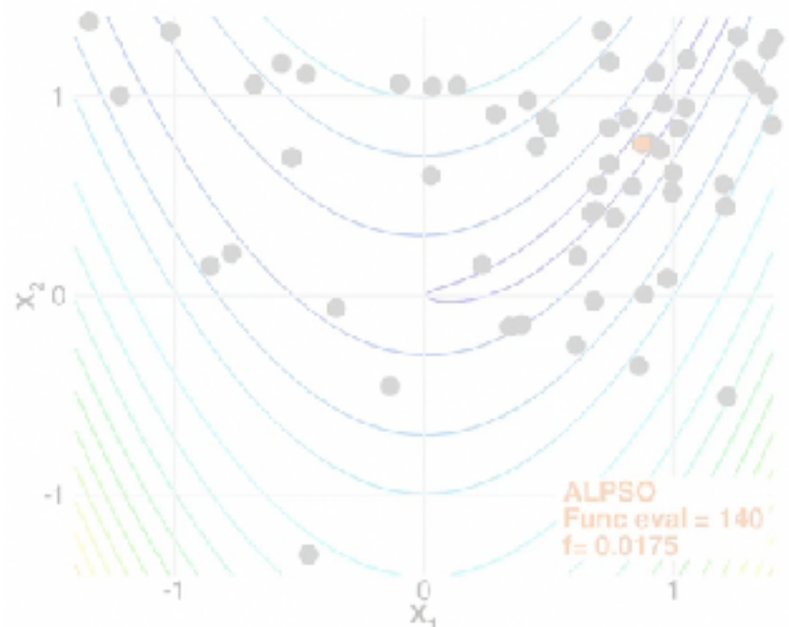
Gradient-based opt. is key to large-scale but derivative computation is the bottleneck

Gradient-based (SNOPT)
41 iterations

Efficient derivative computation is difficult especially with multiple disciplines

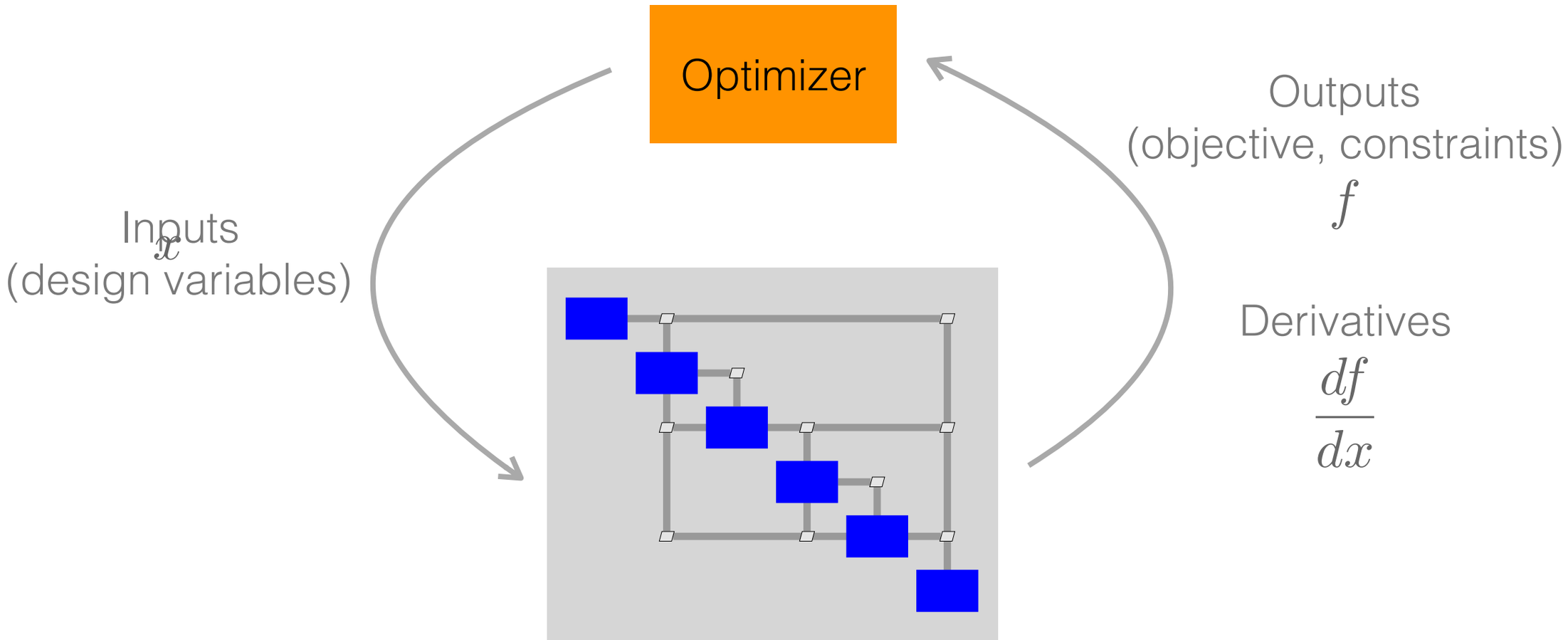


Gradient-free (ALPSO)
1340 iterations



MAUD was developed to simplify efficient differentiation of multidisciplinary models

MAUD: modular analysis and unified derivatives



The most efficient method for computing $\frac{df}{dx}$ is different for each model

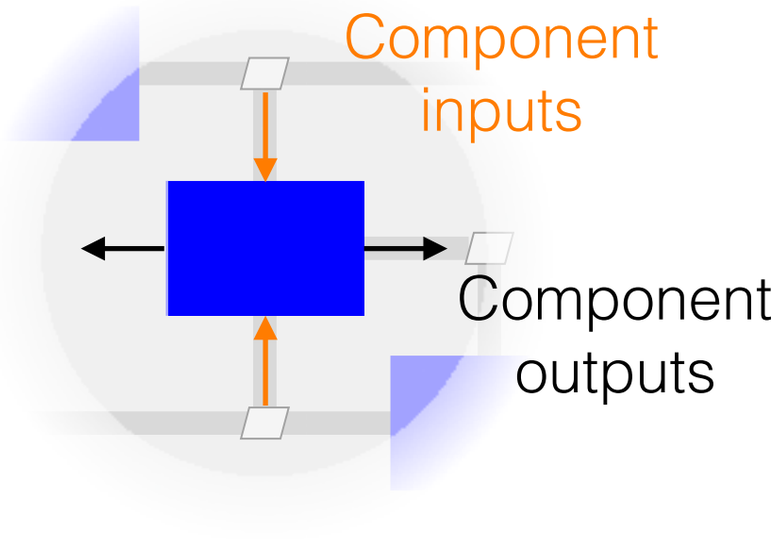
Outline

1. MAUD: mathematical formulation and theory
    - Unification of the methods for computing derivatives
    - Automation of a significant part of derivative computation
2. MAUD: algorithmic aspects
    - Hierarchical solution for minimal overhead
    - Automated parallelization
3. Overview of OpenMDAO v2 applications

Outline

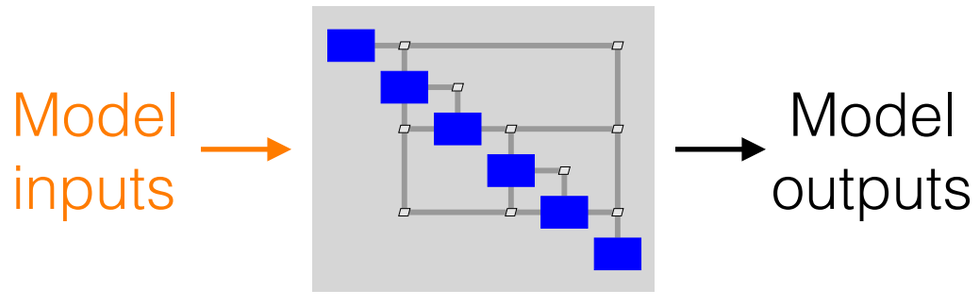
1. MAUD: mathematical formulation and theory
    - Unification of the methods for computing derivatives
    - Automation of a significant part of derivative computation
2. MAUD: algorithmic aspects
    - Hierarchical solution approach
    - Parallel execution
3. Overview of OpenMDAO v2 applications

Derivatives are computed in two steps: (1) partial derivatives (2) total derivatives



Partial derivatives:

- Derivative of a function with respect to one of its arguments
- A local property computed at the component level
- Methods: FD, CS, AD, symb., …



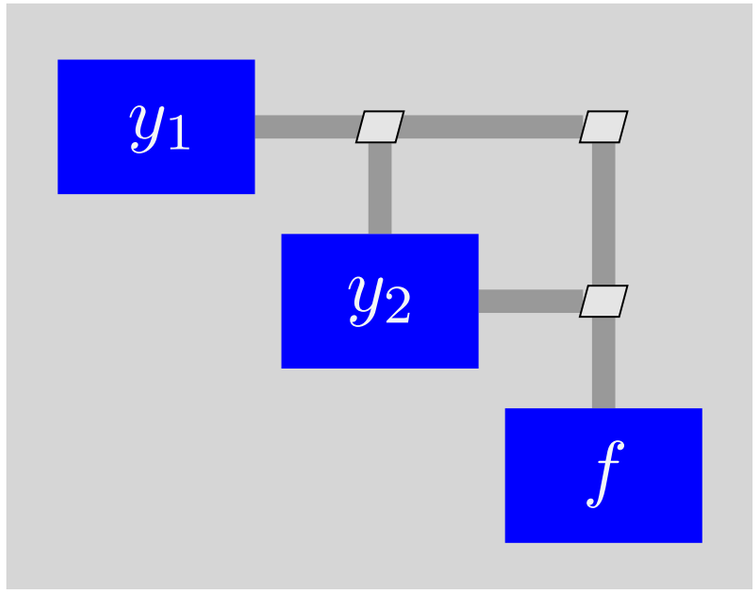
Total derivatives:

- Derivative of a variable with respect to another variable
- A global property computed at the model level
- Methods: FD, chain rule, adjoint, …

  MAUD unifies all of these methods

Examples of total derivative computation

1. Two sequential disciplines: chain rule



$$y_1 = \mathcal{Y}_1(x) \longleftarrow \text{Discipline 1}$$

$$y_2 = \mathcal{Y}_2(x, y_1) \longleftarrow \text{Discipline 2}$$

$$f = \mathcal{F}(x, y_1, y_2) \longleftarrow \text{Objective \& Constraints}$$

Chain rule

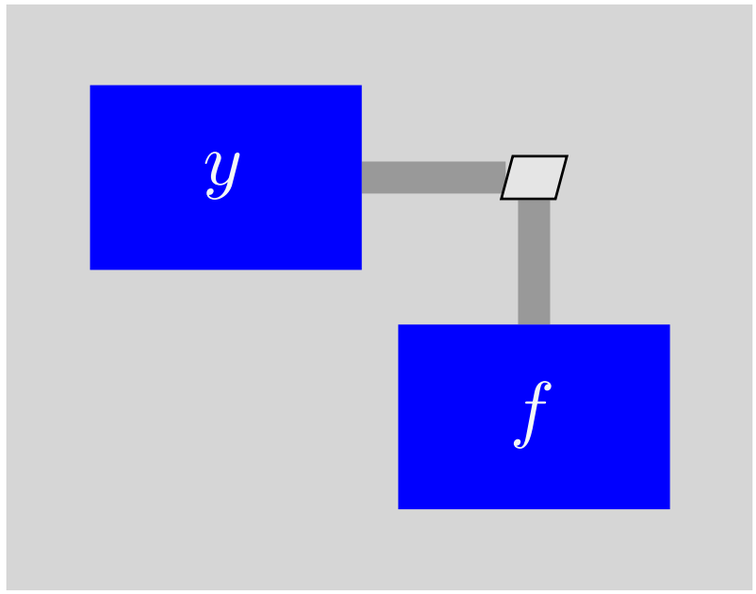
$$\frac{dy_1}{dx} = \frac{\partial \mathcal{Y}_1}{\partial x}$$

$$\frac{dy_2}{dx} = \frac{\partial \mathcal{Y}_2}{\partial x} + \frac{\partial \mathcal{Y}_2}{\partial y_1}\frac{dy_1}{dx}$$

$$\frac{df}{dx} = \frac{\partial F}{\partial x} + \frac{\partial F}{\partial y_1}\frac{dy_1}{dx} + \frac{\partial F}{\partial y_2}\frac{dy_2}{dx}$$

Examples of total derivative computation

2. Implicit discipline: direct/adjoint method



$\mathcal{R}(x, y) = 0$ ← States defined implicitly

$f = \mathcal{F}(x, y)$ ← Objective & Constraints

$$\frac{\partial \mathcal{R}}{\partial y}\frac{dy}{dx} = -\frac{\partial \mathcal{R}}{\partial y}$$

$$\frac{df}{dx} = \frac{\partial \mathcal{F}}{\partial x} + \frac{\partial \mathcal{F}}{\partial x}\frac{dy}{dx}$$

Direct method

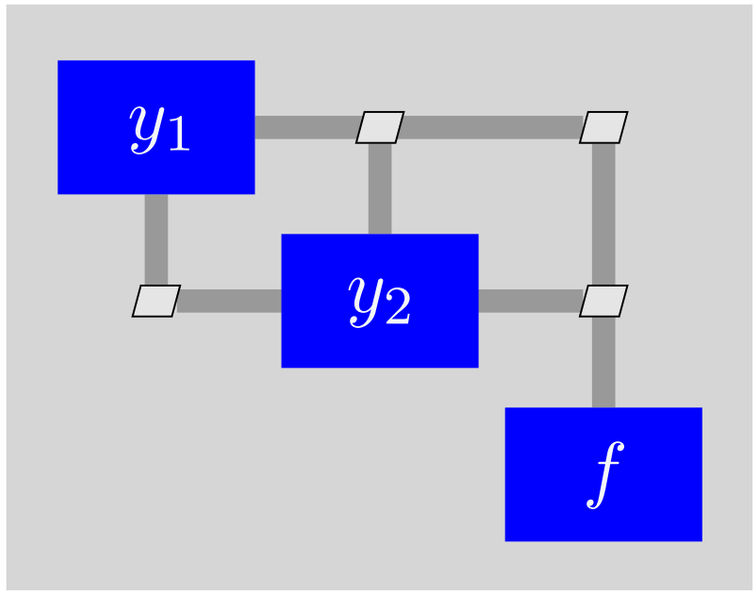
$$\frac{\partial \mathcal{R}}{\partial y}^{T}\ \frac{df}{dr}^{T} = -\ \frac{\partial \mathcal{F}}{\partial y}^{T}$$

$$\frac{df}{dx} = \frac{\partial \mathcal{F}}{\partial x} + \frac{df}{dr}\frac{\partial \mathcal{R}}{\partial x}$$

Adjoint method

Examples of total derivative computation

3. Two coupled disciplines: GSE method



$$y_1 = \mathcal{Y}_1(x, y_2) \longleftarrow \text{Discipline 1}$$

$$y_2 = \mathcal{Y}_2(x, y_1) \longleftarrow \text{Discipline 2}$$

$$f = \mathcal{F}(x, y_1, y_2) \longleftarrow \text{Objective \& Constraints}$$

$$\begin{bmatrix} \mathcal{I} & -\frac{\partial \mathcal{Y}_1}{\partial y_2} \\ -\frac{\partial \mathcal{Y}_2}{\partial y_1} & \mathcal{I} \end{bmatrix} \begin{bmatrix} \frac{dy_1}{dx} \\ \frac{dy_2}{dx} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{Y}_1}{\partial x} \\ \frac{\partial \mathcal{Y}_2}{\partial x} \end{bmatrix}$$

$$\frac{df}{dx} = \frac{\partial \mathcal{F}}{\partial x} + \frac{\partial \mathcal{F}}{\partial y_1}\frac{dy_1}{dx} + \frac{\partial \mathcal{F}}{\partial y_2}\frac{dy_2}{dx}$$

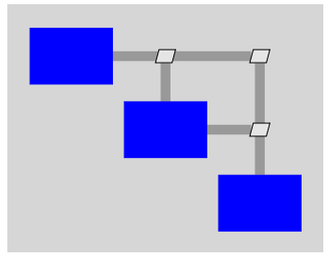
Forward form

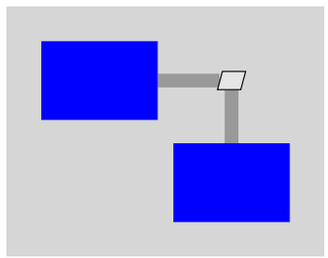
$$\begin{bmatrix} \mathcal{I} & -\frac{\partial \mathcal{Y}_2}{\partial y_1} \\ -\frac{\partial \mathcal{Y}_1}{\partial y_2} & \mathcal{I} \end{bmatrix} \begin{bmatrix} \frac{df}{dy_1} \\ \frac{df}{dy_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial y_1} \\ \frac{\partial \mathcal{F}}{\partial y_2} \end{bmatrix}$$

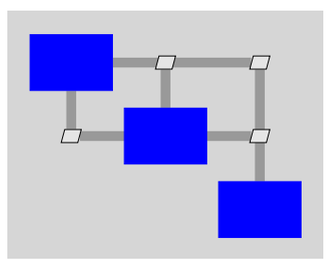
$$\frac{df}{dx} = \frac{\partial \mathcal{F}}{\partial x} + \frac{df}{dy_1}\frac{\partial \mathcal{Y}_1}{\partial x} + \frac{df}{dy_2}\frac{\partial \mathcal{Y}_2}{\partial x}$$

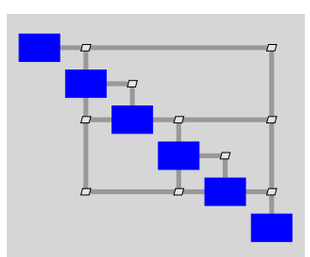
Reverse form

We can unify all methods for computing total derivatives using a single equation









reformulate

$$\mathrm{R(u)} = 0$$

Inverse Function Theorem

If you are curious about the details:

$$\left(\frac{\partial \left(R^{-1} \right)}{\partial r} = \frac{\partial R}{\partial u}^{-1} \right) \qquad \frac{du}{dr} := \frac{\partial \left(R^{-1} \right)}{\partial r}$$

$$\frac{\partial R}{\partial u} \frac{du}{dr} = \mathcal{I} = \frac{\partial R}{\partial u}^{T} \frac{du}{dr}^{T}$$

By choosing the right u & R, each method can be derived from the unifying equation

$$\frac{\partial R}{\partial u}\frac{du}{dr} = \mathcal{I} = \frac{\partial R}{\partial u}^{T}\frac{du}{dr}^{T}$$

Direct method

$$\frac{\partial \mathcal{R}}{\partial y}\frac{dy}{dx} = -\frac{\partial \mathcal{R}}{\partial y}$$

$$\frac{df}{dx} = \frac{\partial \mathcal{F}}{\partial x} + \frac{\partial \mathcal{F}}{\partial x}\frac{dy}{dx}$$

Forward GSE

$$\begin{bmatrix} \mathcal{I} & -\frac{\partial \mathcal{Y}_1}{\partial y_2} \\ -\frac{\partial \mathcal{Y}_2}{\partial y_1} & \mathcal{I} \end{bmatrix} \begin{bmatrix} \frac{dy_1}{dx} \\ \frac{dy_2}{dx} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{Y}_1}{\partial x} \\ \frac{\partial \mathcal{Y}_2}{\partial x} \end{bmatrix}$$

$$\frac{df}{dx} = \frac{\partial \mathcal{F}}{\partial x} + \frac{\partial \mathcal{F}}{\partial y_1}\frac{dy_1}{dx} + \frac{\partial \mathcal{F}}{\partial y_2}\frac{dy_2}{dx}$$

Chain rule

$$\frac{dy_1}{dx} = \frac{\partial \mathcal{Y}_1}{\partial x}$$

$$\frac{dy_2}{dx} = \frac{\partial \mathcal{Y}_2}{\partial x} + \frac{\partial \mathcal{Y}_2}{\partial y_1}\frac{dy_1}{dx}$$

$$\frac{df}{dx} = \frac{\partial F}{\partial x} + \frac{\partial F}{\partial y_1}\frac{dy_1}{dx} + \frac{\partial F}{\partial y_2}\frac{dy_2}{dx}$$

Adjoint method

$$\frac{\partial \mathcal{R}}{\partial y}^{T}\frac{df}{dr}^{T} = -\frac{\partial \mathcal{F}}{\partial y}^{T}$$

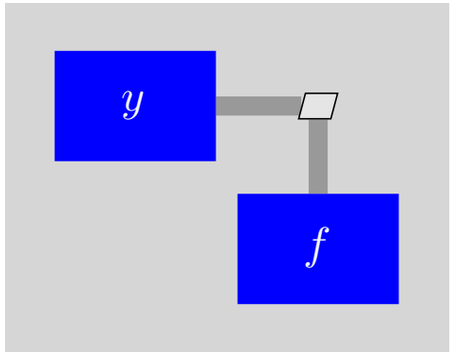
$$\frac{df}{dx} = \frac{\partial \mathcal{F}}{\partial x} + \frac{df}{dr}\frac{\partial \mathcal{R}}{\partial x}$$

Reverse GSE

$$\begin{bmatrix} \mathcal{I} & -\frac{\partial \mathcal{Y}_2}{\partial y_1} \\ -\frac{\partial \mathcal{Y}_1}{\partial y_2} & \mathcal{I} \end{bmatrix} \begin{bmatrix} \frac{df}{dy_1} \\ \frac{df}{dy_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{F}}{\partial y_1} \\ \frac{\partial \mathcal{F}}{\partial y_2} \end{bmatrix}$$

$$\frac{df}{dx} = \frac{\partial \mathcal{F}}{\partial x} + \frac{df}{dy_1}\frac{\partial \mathcal{Y}_1}{\partial x} + \frac{df}{dy_2}\frac{\partial \mathcal{Y}_2}{\partial x}$$

Example: the direct / adjoint method



$$\mathcal{R}(x,y) = 0$$

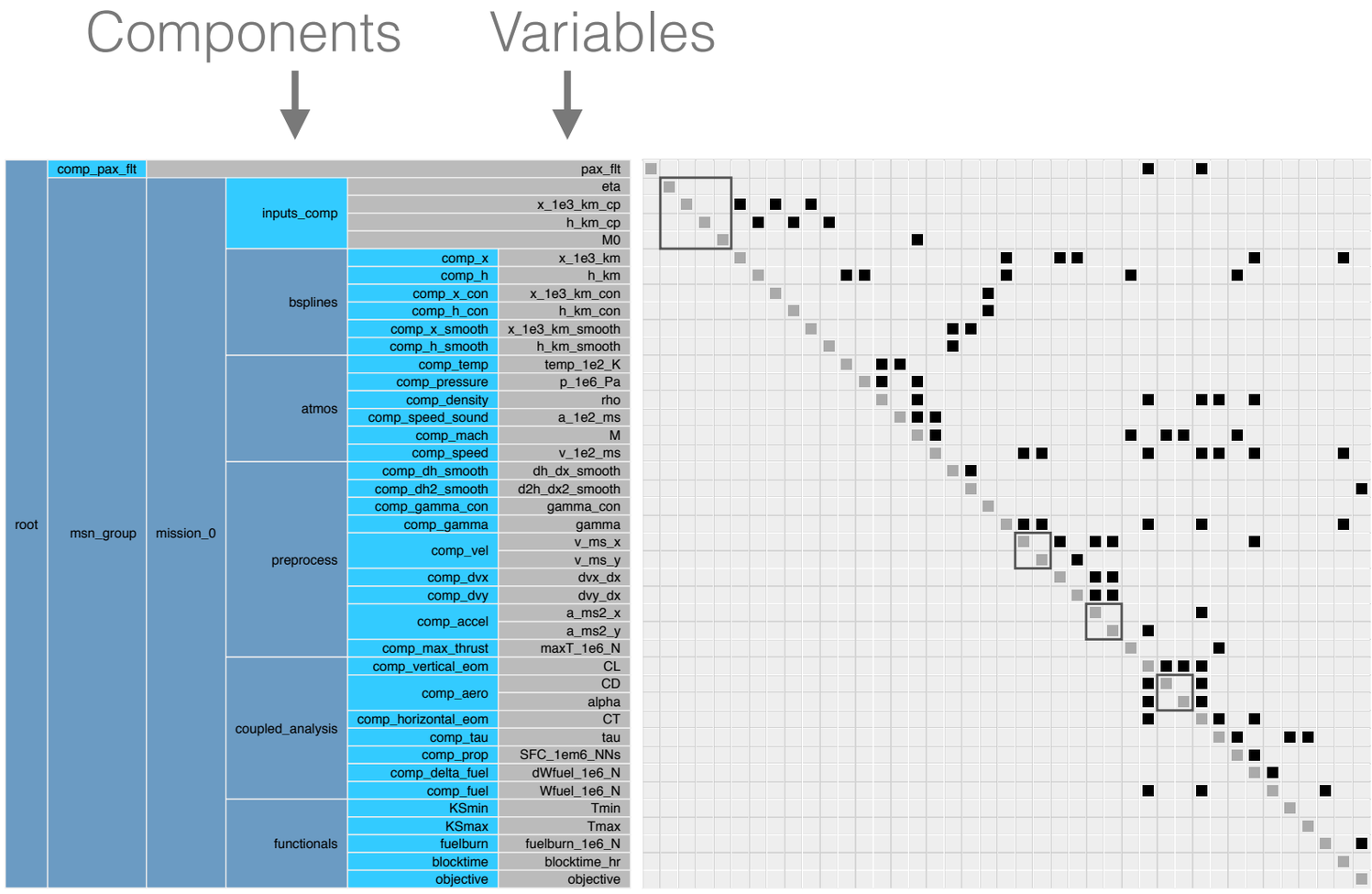
$$f = \mathcal{F}(x,y)$$

$$\rightarrow \quad u = \begin{bmatrix} x \\ y \\ f \end{bmatrix} \qquad R(u) = \begin{bmatrix} x - x^* \\ -\mathcal{R}(x,y) \\ f - \mathcal{F}(x,y) \end{bmatrix}$$

$$\frac{\partial R}{\partial u}\frac{du}{dr} = \mathcal{I} = \frac{\partial R}{\partial u}^T \frac{du}{dr}^T$$

$$\begin{bmatrix} \mathcal{I} & 0 & 0 \\ -\frac{\partial \mathcal{R}}{\partial x} & -\frac{\partial \mathcal{R}}{\partial y} & 0 \\ -\frac{\partial \mathcal{F}}{\partial x} & -\frac{\partial \mathcal{F}}{\partial y} & \mathcal{I} \end{bmatrix} \begin{bmatrix} \mathcal{I} & 0 & 0 \\ \frac{\mathrm{d}y}{\mathrm{d}x} & \frac{\mathrm{d}y}{\mathrm{d}r_y} & 0 \\ \frac{\mathrm{d}f}{\mathrm{d}x} & \frac{\mathrm{d}f}{\mathrm{d}r_y} & \mathcal{I} \end{bmatrix} = \begin{bmatrix} \mathcal{I} & 0 & 0 \\ 0 & \mathcal{I} & 0 \\ 0 & 0 & \mathcal{I} \end{bmatrix} = \begin{bmatrix} \mathcal{I} & -\frac{\partial \mathcal{R}}{\partial x} & -\frac{\partial \mathcal{F}}{\partial x} \\ 0 & -\frac{\partial \mathcal{R}}{\partial y} & -\frac{\partial \mathcal{F}}{\partial y} \\ 0 & 0 & \mathcal{I} \end{bmatrix} \begin{bmatrix} \mathcal{I} & \frac{\mathrm{d}y}{\mathrm{d}x} & \frac{\mathrm{d}f}{\mathrm{d}x} \\ 0 & \frac{\mathrm{d}y}{\mathrm{d}r_y} & \frac{\mathrm{d}f}{\mathrm{d}r_y} \\ 0 & 0 & \mathcal{I} \end{bmatrix}$$

Significance: when building large models, total derivative computation is automated



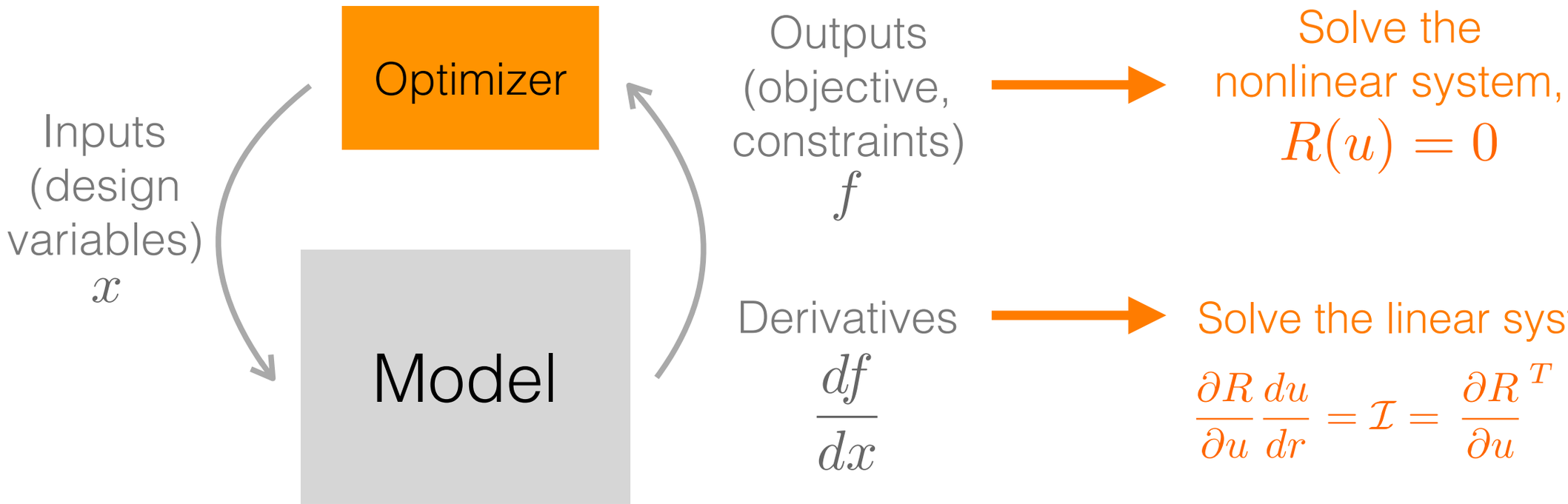
1. We implement our components and their partial derivatives

2. Solving the unifying equation automatically applies the right method

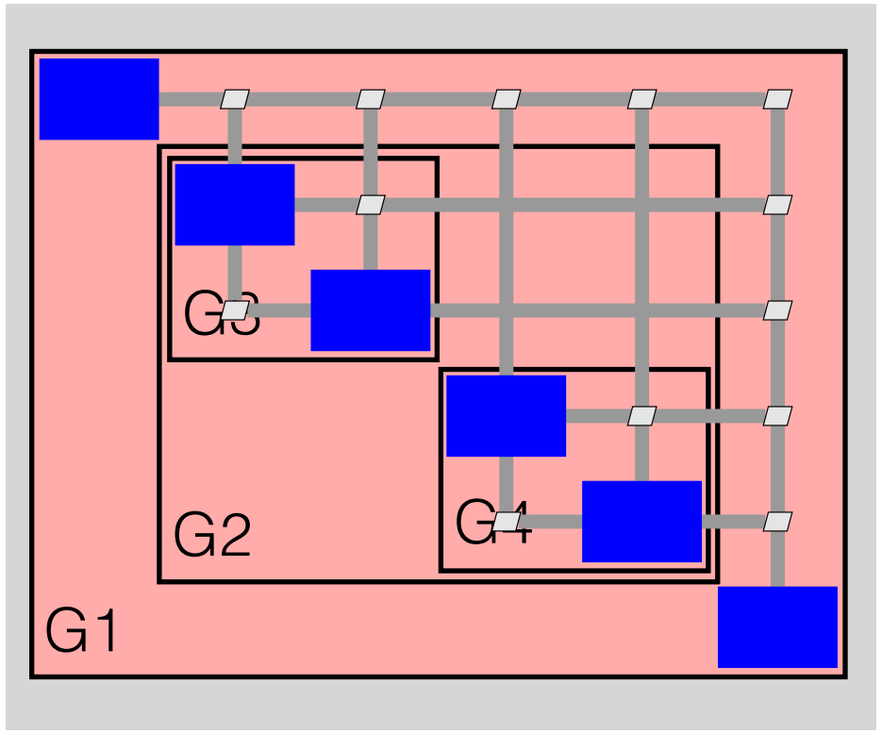
Outline

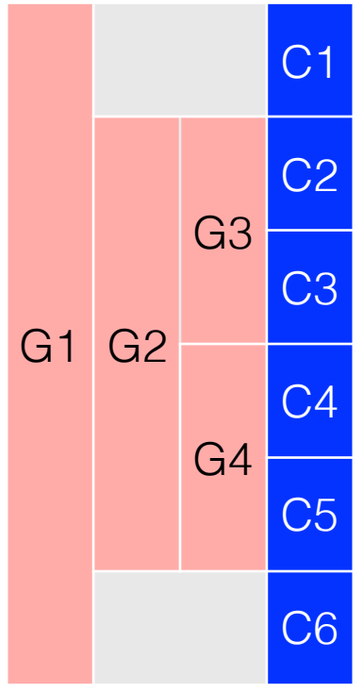
1. MAUD: mathematical formulation and theory
   - Unification of the methods for computing derivatives
   - Automation of a significant part of derivative computation
2. MAUD: algorithmic aspects
   - Hierarchical solution approach
   - Parallel execution
3. Overview of OpenMDAO v2 applications

Running the model now becomes solving a system of equations



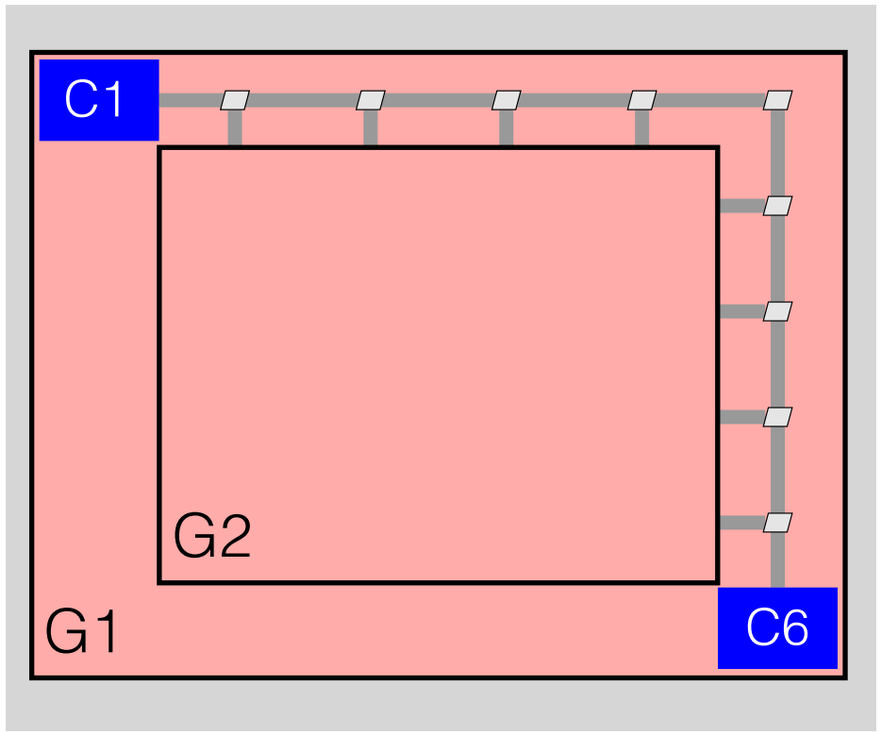
To solve these systems efficiently, we use hierarchical solvers

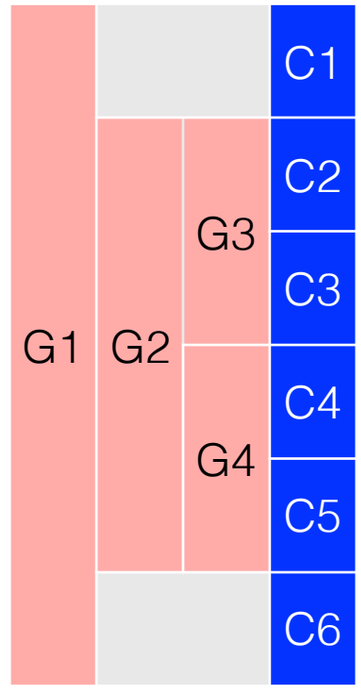




At each level in the hierarchy, we will apply the appropriate solver (nonlinear and linear)

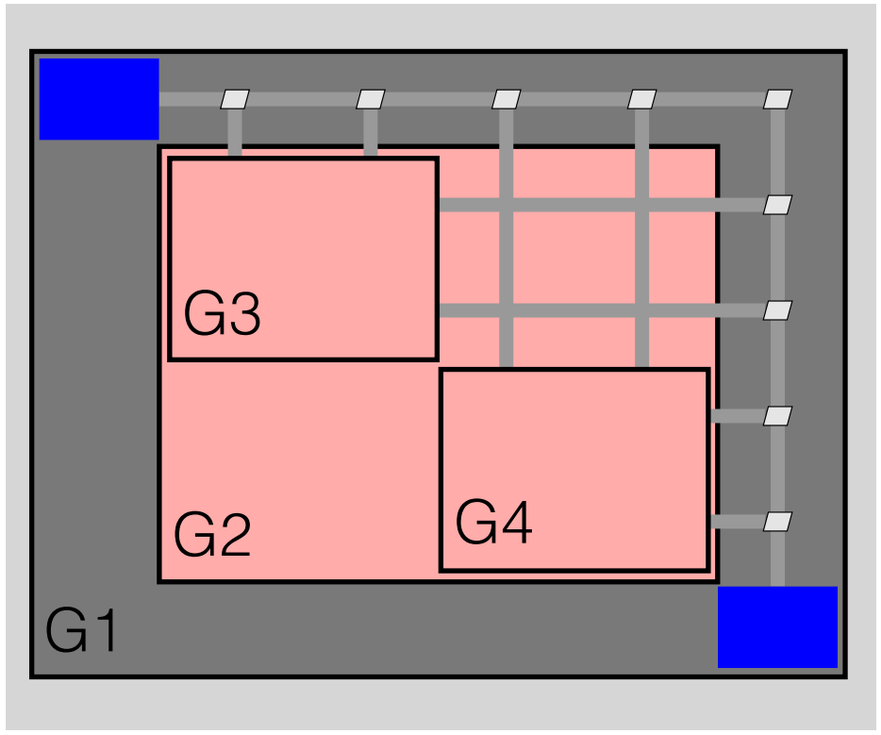
For Group 1, its subsystems are sequential

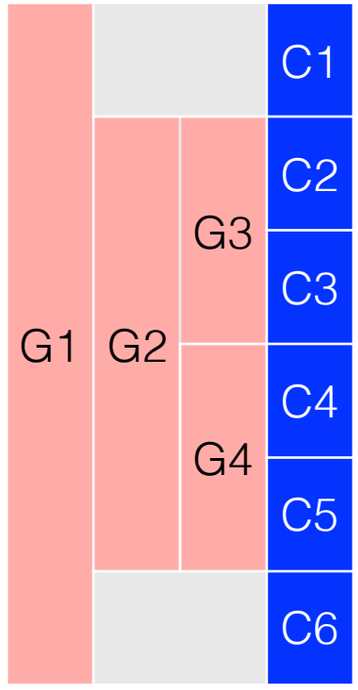




Since all dependencies are feed-forward at this level,
a single block Gauss—Seidel iteration is sufficient

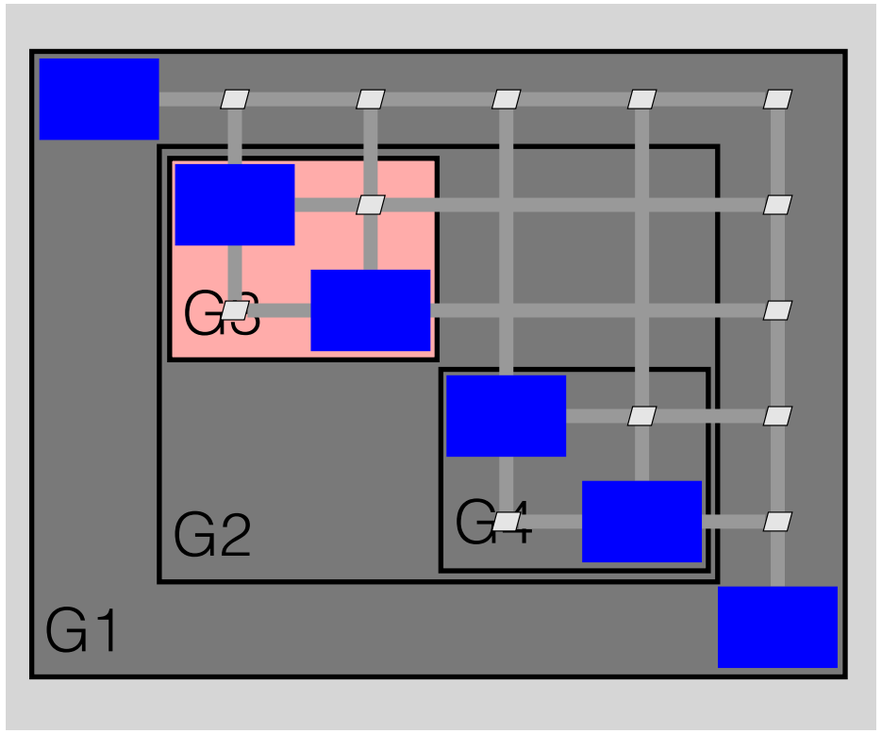
For Group 2, its subsystems are decoupled

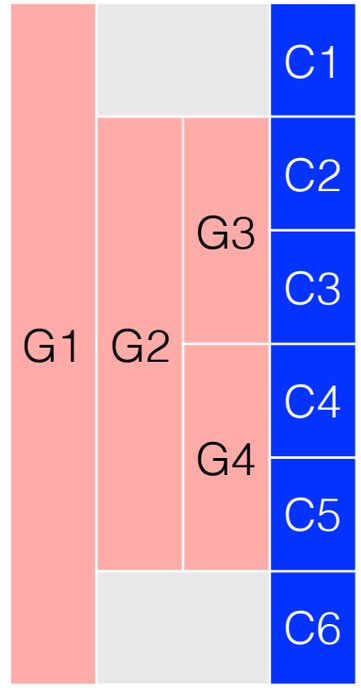




Since there are no dependencies at this level,
a single block Jacobi iteration is sufficient

For Group 3, its subsystems are coupled

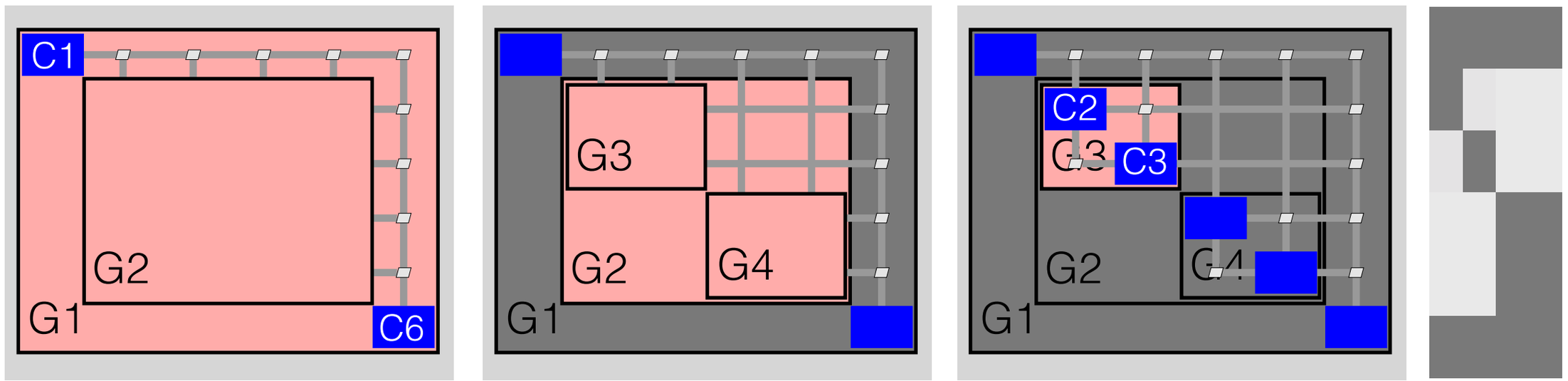




Since the dependencies include a feedback loop,
a (nonlinear or linear) solver is required.

We can parallelize anywhere at any level in the hierarchy

For instance, if we run this model with 4 processors:



In group G1, it does not make sense to parallelize.

C1: 4 processors
G2: 4 processors
C6: 4 processors

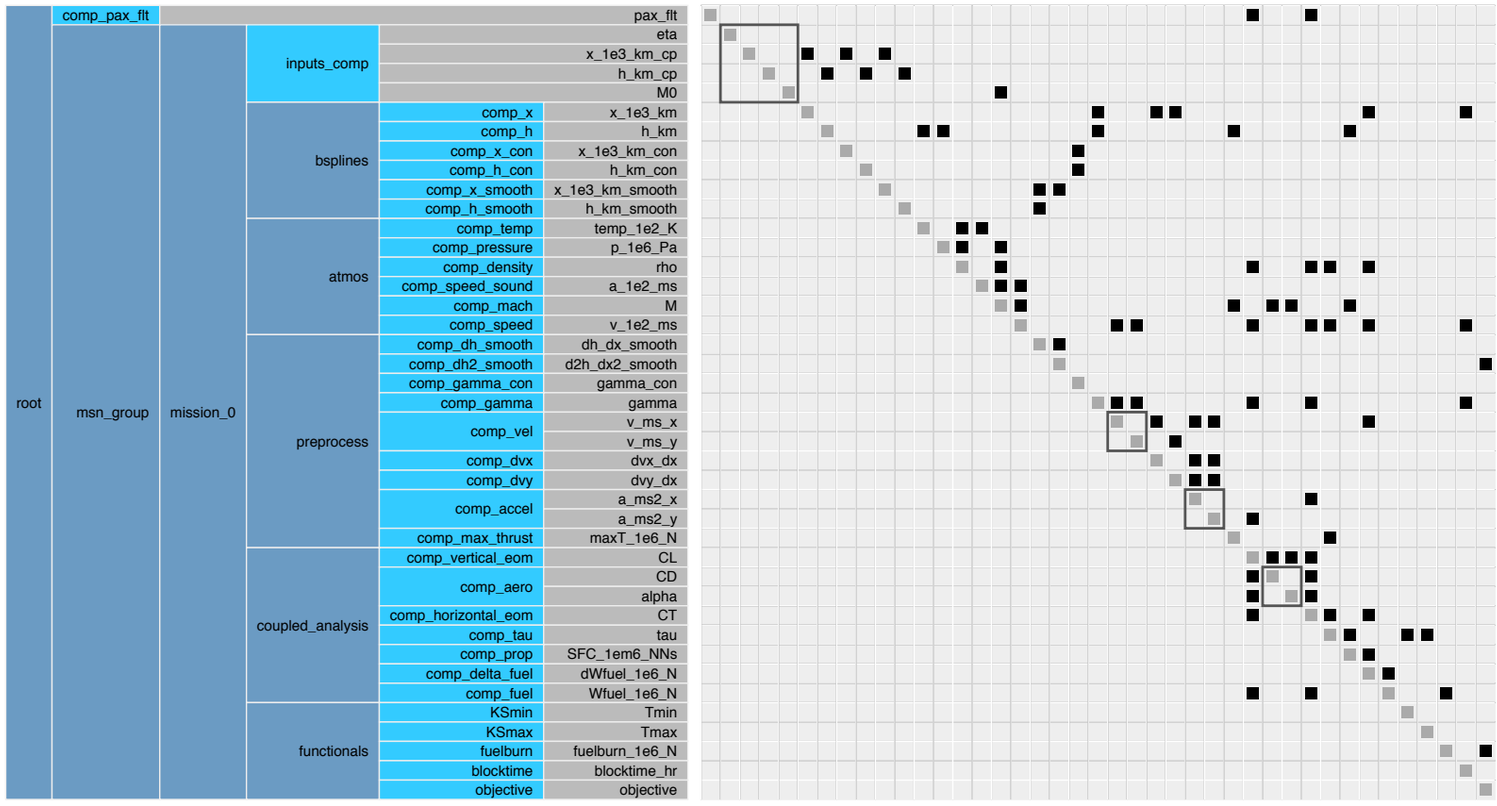
In group G2, it does makes sense to parallelize.

G3: 2 processors
G4: 2 processors

In group G3, we might parallelize based on our solver (say we do).

Comp 2: 1 processors
Comp 3: 1 processors

Significance: we can assign solvers and/or choose to parallelize at any level



OpenMDAO v2 methods and applications

1. New methods
    - Reconfigurability
    - Ozone: ODE and optimal control solver library
2. Recent applications
    - Topology optimization
    - Aircraft design-allocation optimization
    - Electric aircraft MDO

OpenMDAO v2 methods and applications

1. New methods
   - Reconfigurability
   - Ozone: ODE and optimal control solver library
2. Recent applications
   - Topology optimization
   - Aircraft design-allocation optimization
   - Electric aircraft MDO

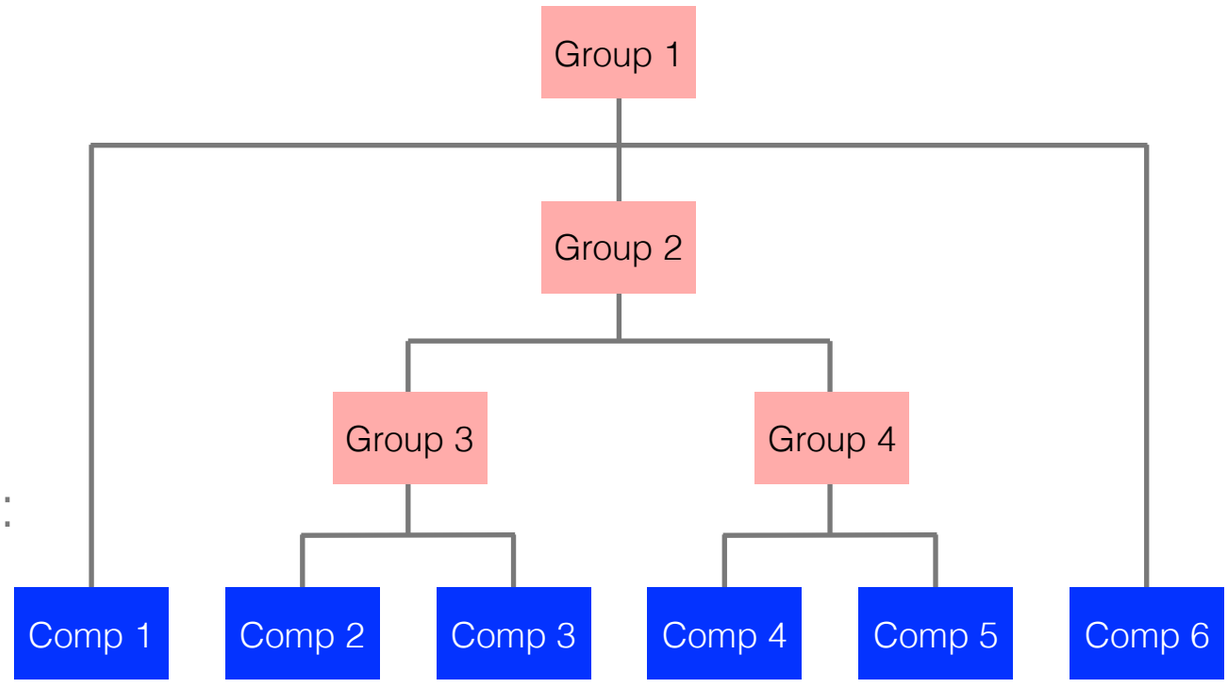
Reconfiguration: changing the model in the middle of running it

Changing variable sizes/adding or removing variables:

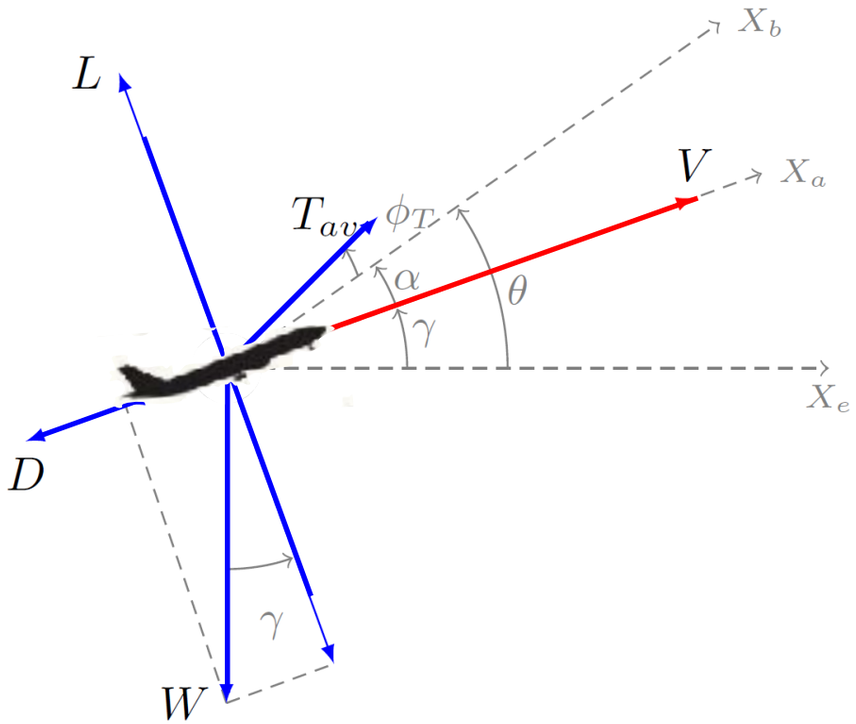
- Adaptive time stepping
- Overset/unstructured CFD
- Adaptive mesh refinement
- Multi-fidelity optimization

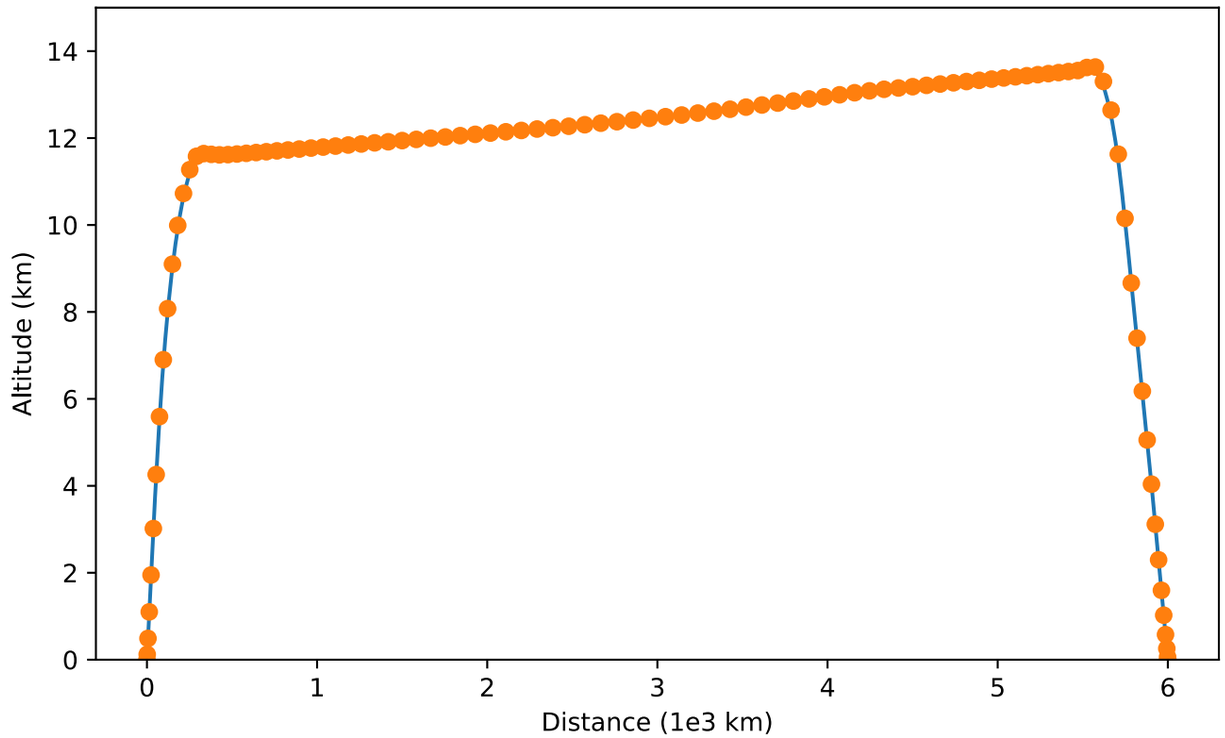
Changing the hierarchy/parallelization:

- Adaptive preconditioners
- Adaptive globalization methods
- Automatically partitioning solvers



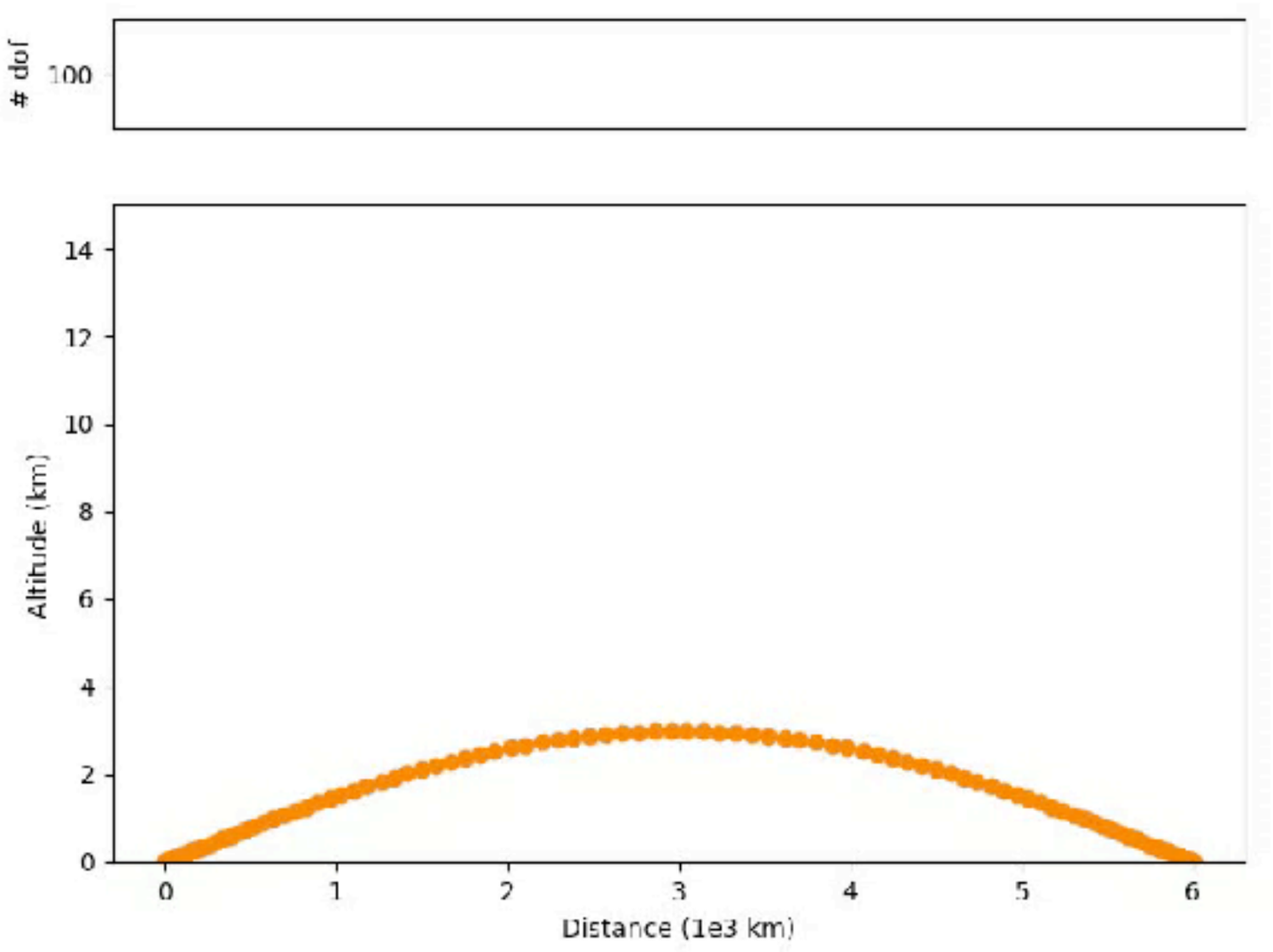
Reconfigurability is used for adaptive optimization of an aircraft mission profile

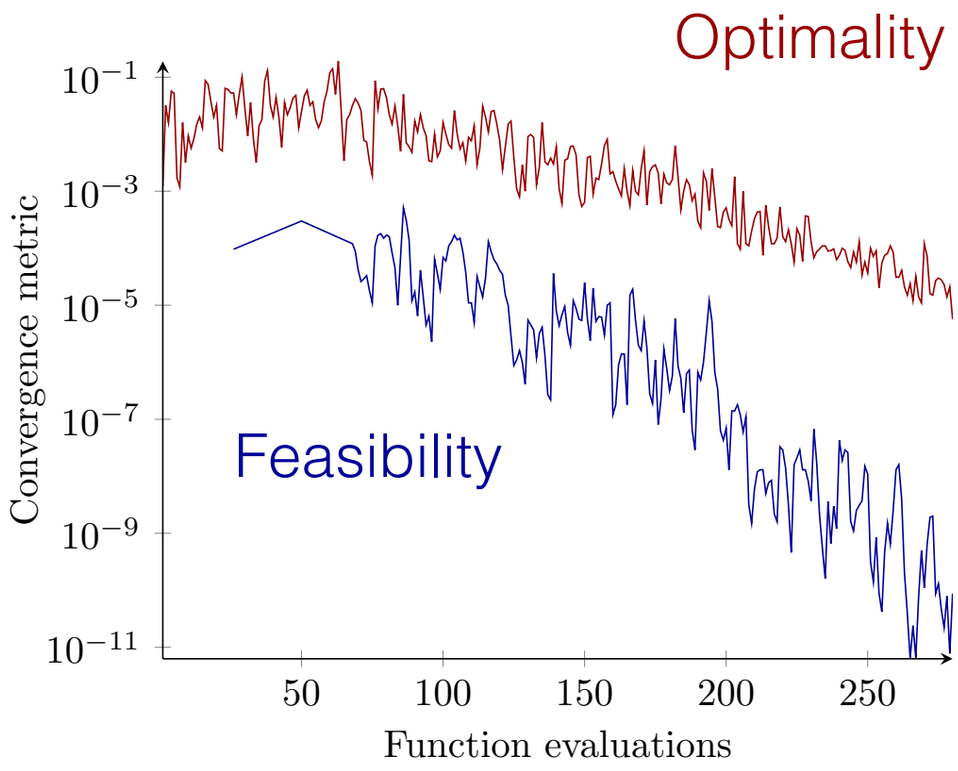




Baseline optimization with no refinement

Minimal fuel-burn altitude optimization: 100 B-spline control points

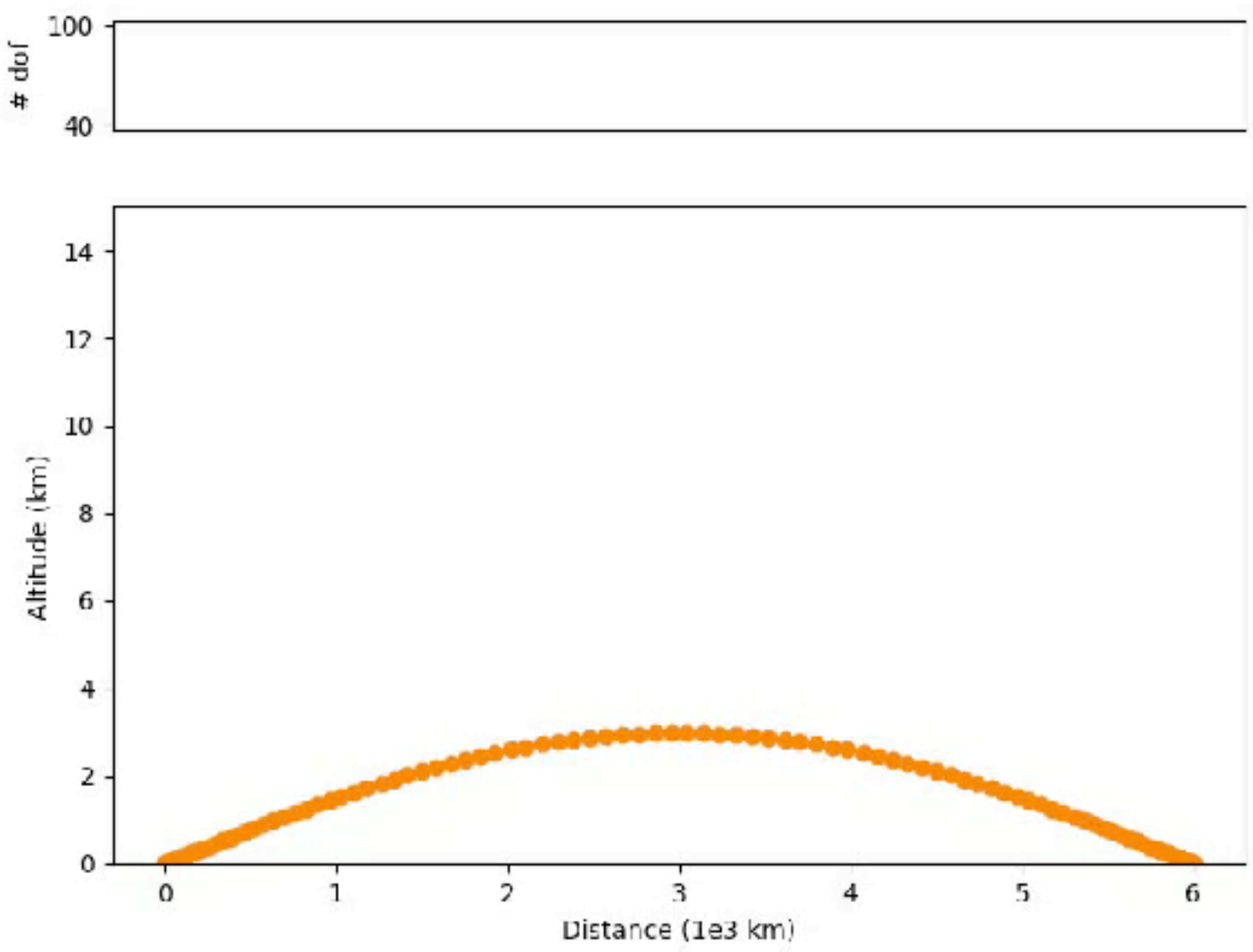


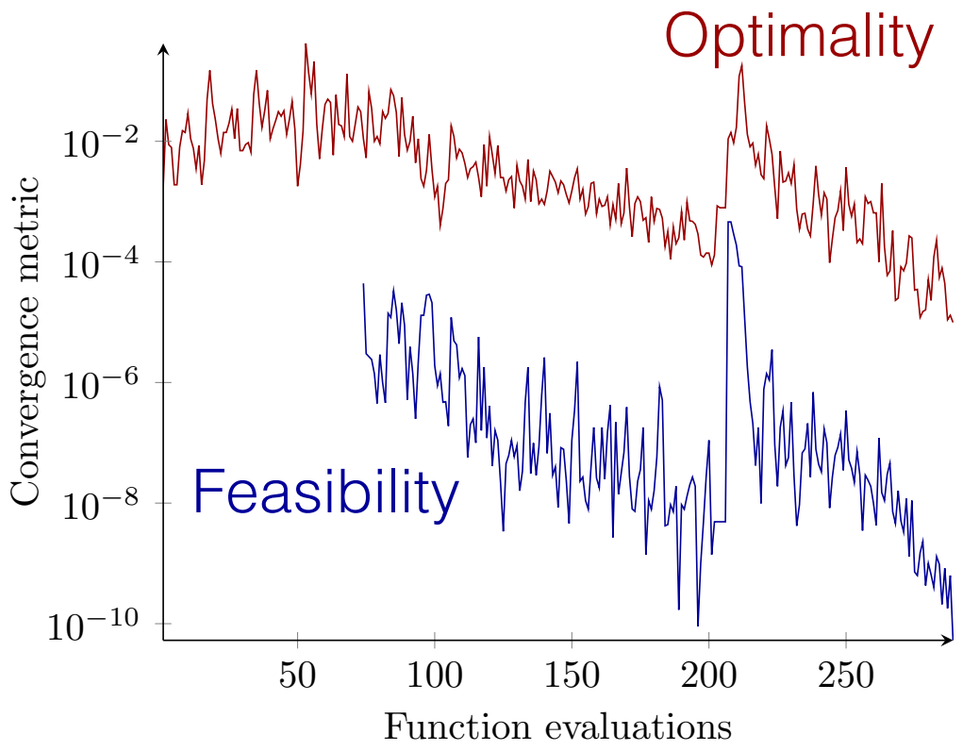


520 seconds

Optimization with refinement

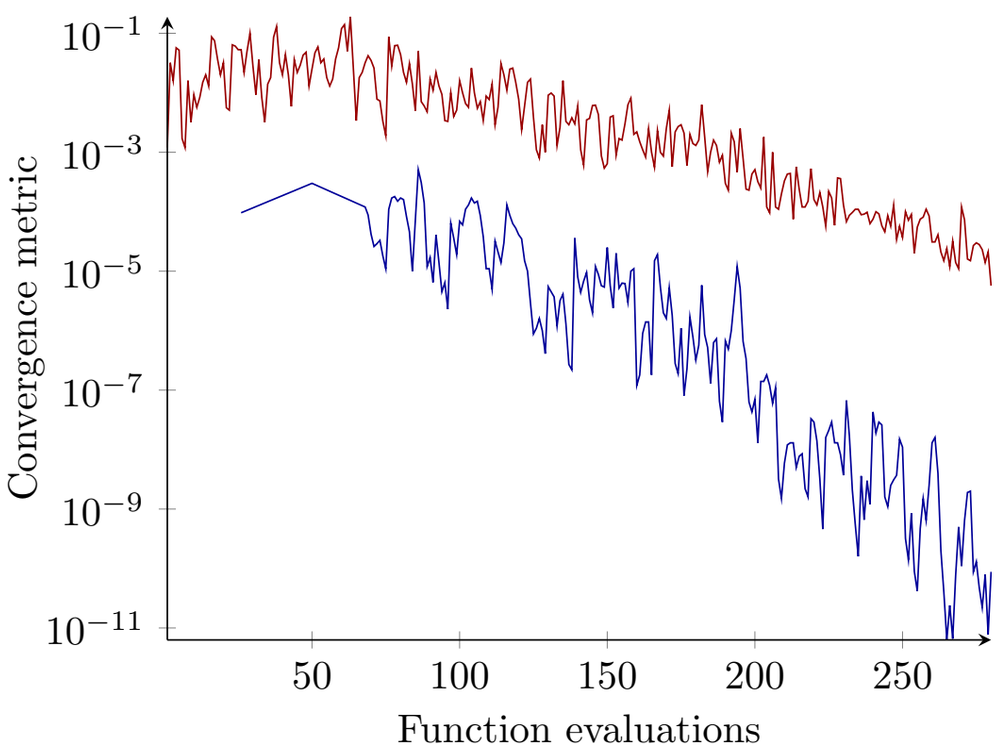
Minimal fuel-burn altitude optimization: 40; 100 degrees of freedom



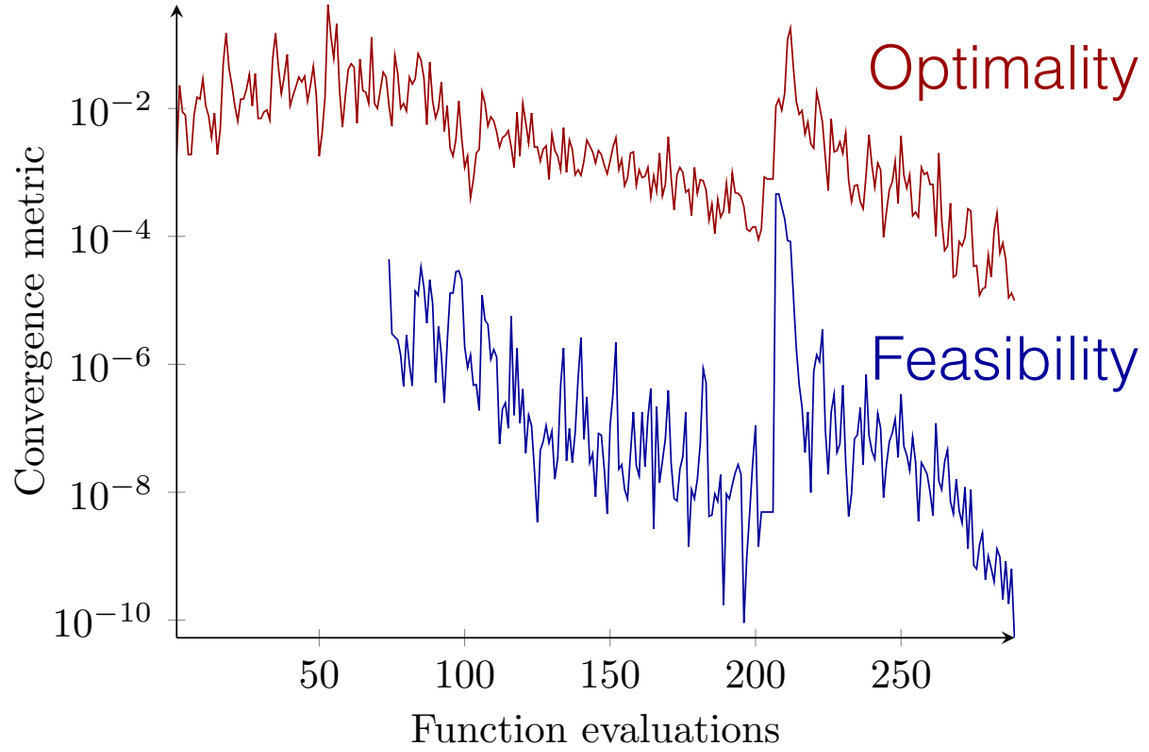


470 seconds

Optimization with a single adaptive refinement is the most efficient



No refinement
(~520 s)



With refinement
(~470 s)

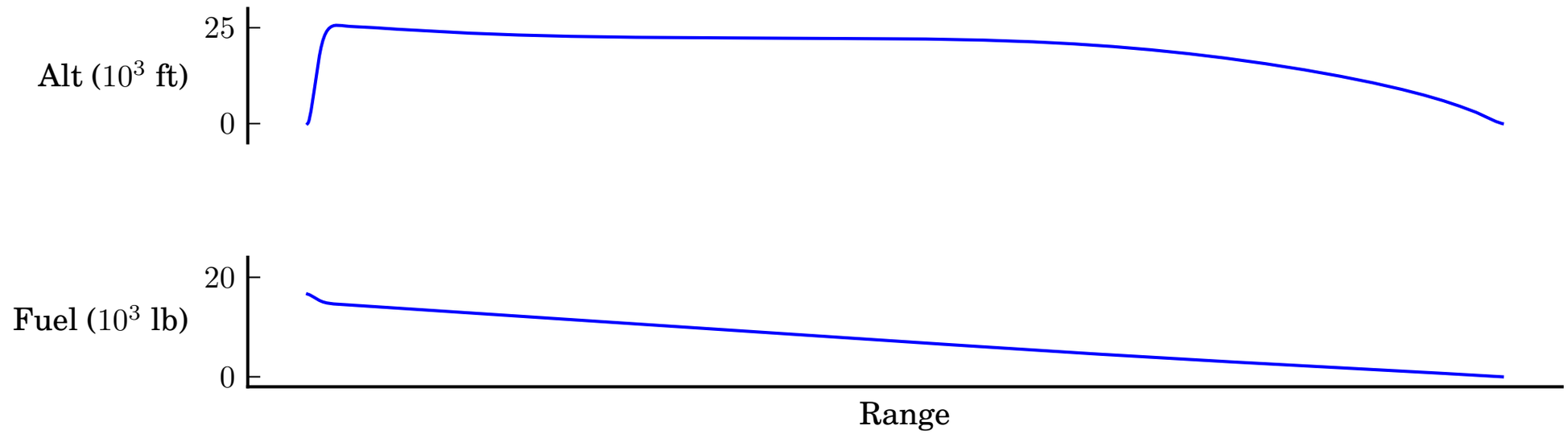
OpenMDAO v2 methods and applications

1. New methods
   - Reconfigurability
   - Ozone: ODE and optimal control solver library
2. Recent applications
   - Topology optimization
   - Aircraft design-allocation optimization
   - Electric aircraft MDO

ODEs or optimal control problems are often found in MDO

For example, in the problem we just saw:

Optimizing the aircraft altitude profile…



…to minimize fuel burn over the mission

$$dW_{fuel} / dt = f(W_{fuel})$$

There are many families of methods but we require them to be differentiated

$$\dot{y} = f(t, y)$$

Explicit methods
(e.g., forward Euler)

$$y_{n+1} = y_n + hf(t_n, y_n)$$

Implicit methods
(e.g., backward Euler)

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1})$$

Multistage methods
(e.g., Runge—Kutta 4)

$$y_{n+1} = y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$

$$k_1 = f(t_n, y_n),$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right),$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right),$$

$$k_4 = f(t_n + h, y_n + hk_3).$$

Multistep methods
(e.g., 2nd order BDF)

$$y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n = \frac{2}{3}hf(t_{n+2}, y_{n+2})$$

All major ODE integration schemes are unified by general linear methods (GLM)

$$\dot{y} = f(t, y)$$

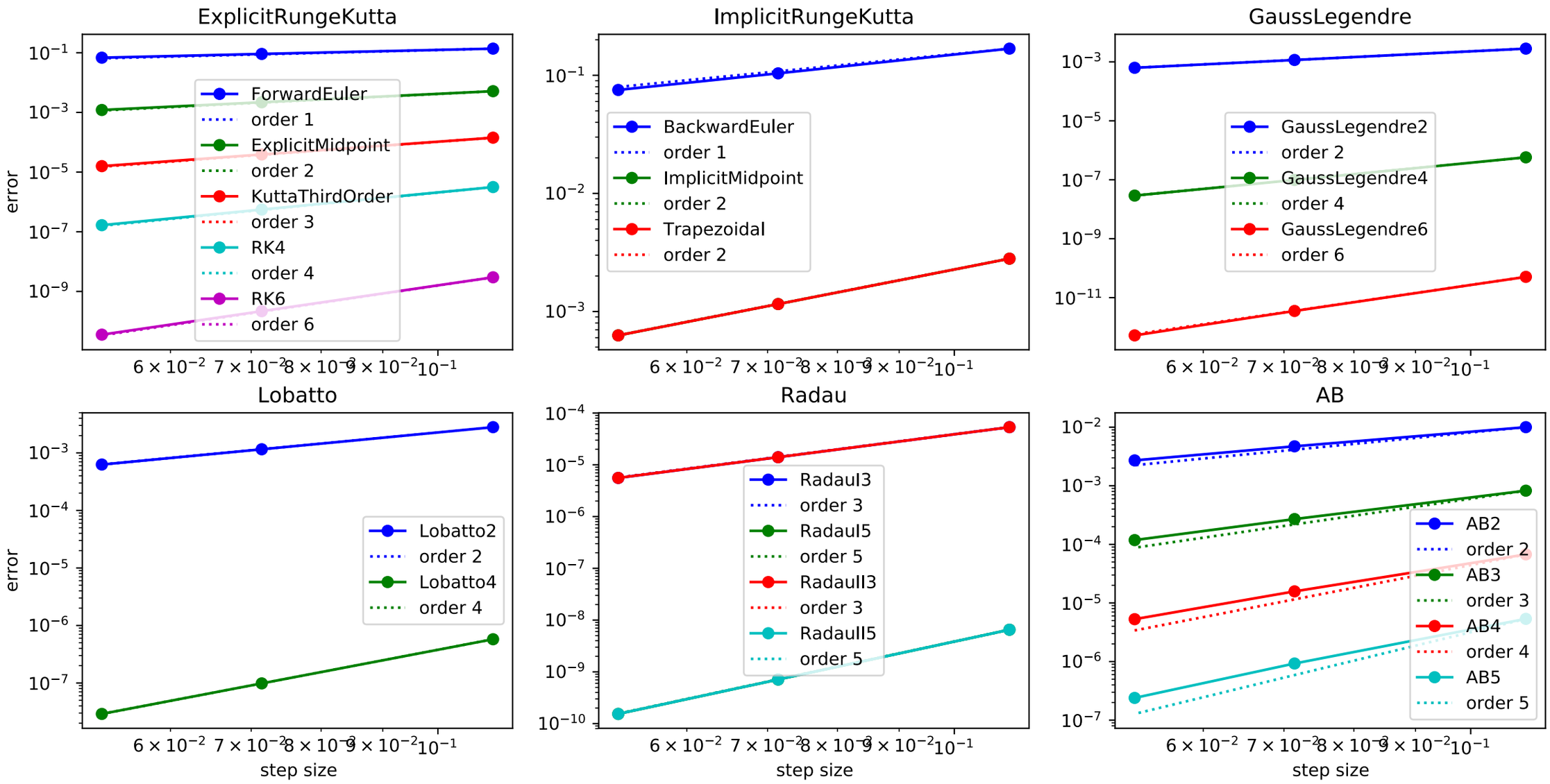
General linear methods formulation:

$$Y_i = \sum_{j=1}^{s} h a_{ij} F_j + \sum_{j=1}^{r} u_{ij} y_j^{[n-1]}, \quad i = 1, \ldots, s,$$

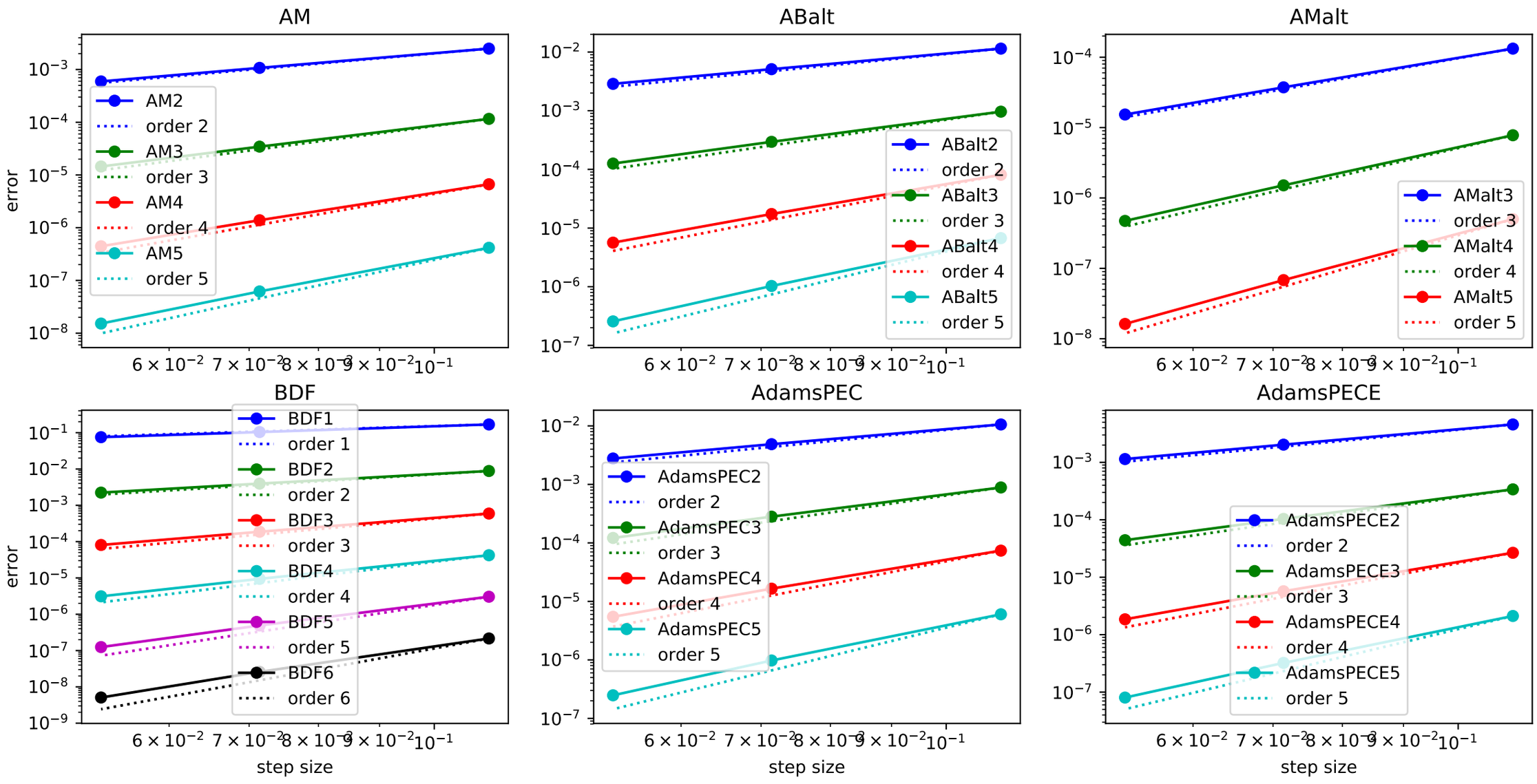
$$y_i^{[n]} = \sum_{j=1}^{s} h b_{ij} F_j + \sum_{j=1}^{r} v_{ij} y_j^{[n-1]}, \quad i = 1, \ldots, r,$$

The coefficient matrices [a_{ij}, b_{ij}, u_{ij}, v_{ij},] determines the scheme.
e.g., $a_{ij}=b_{ij}=u_{ij}=v_{ij}=1$ for forward Euler

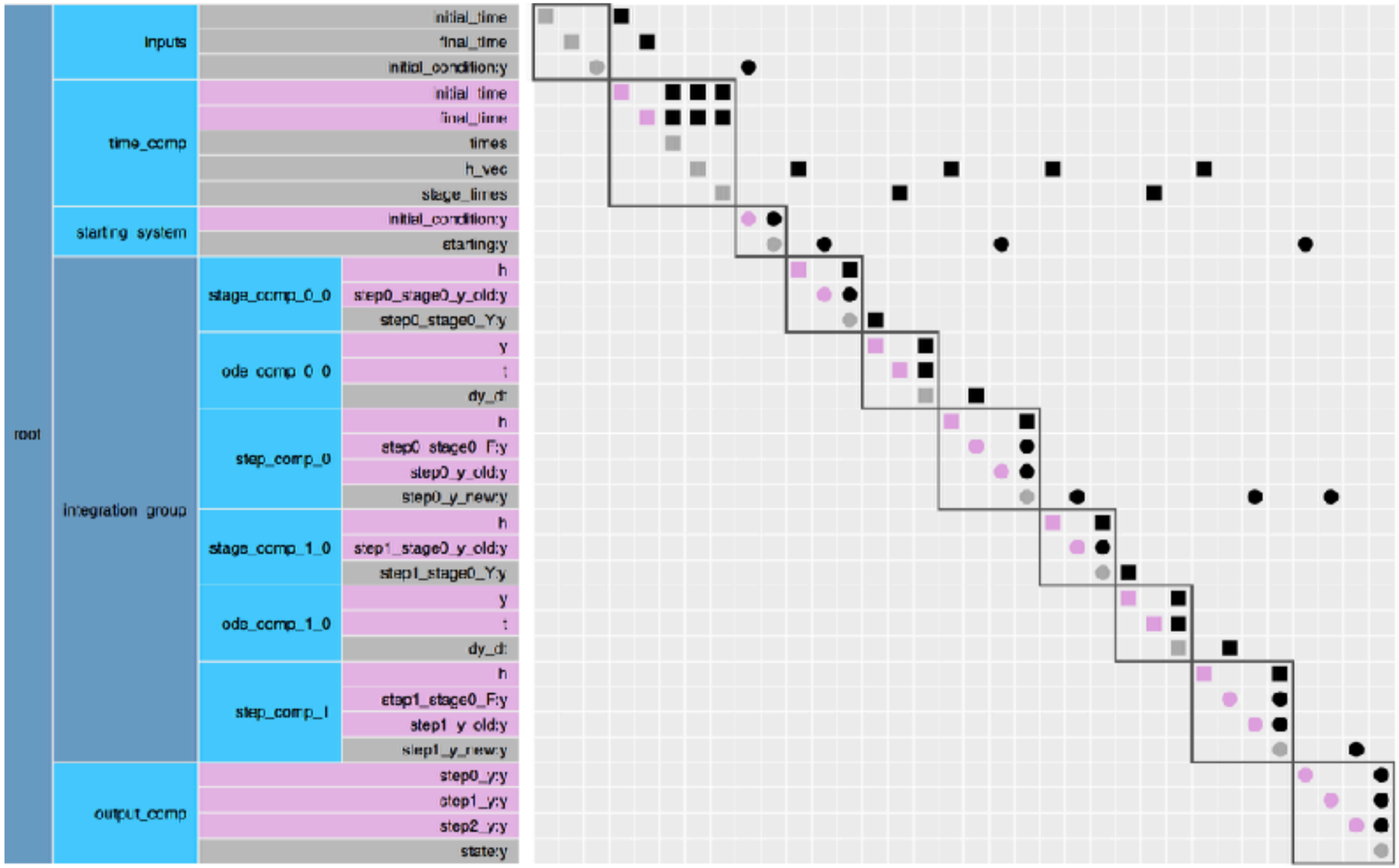
Ozone is a new ODE solver library for OpenMDAO v2 based on GLM



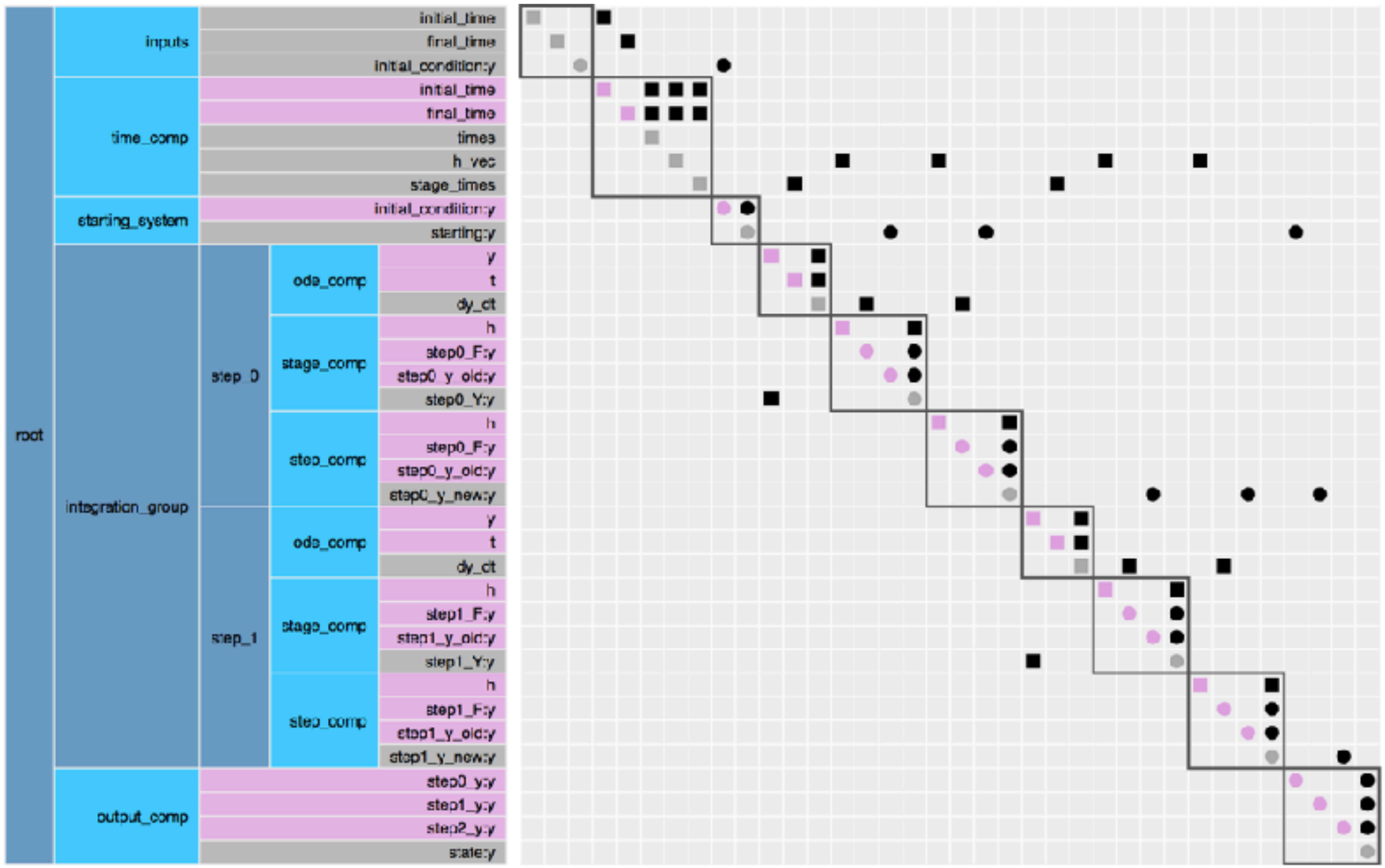
Ozone is a new ODE solver library for OpenMDAO v2 based on GLM



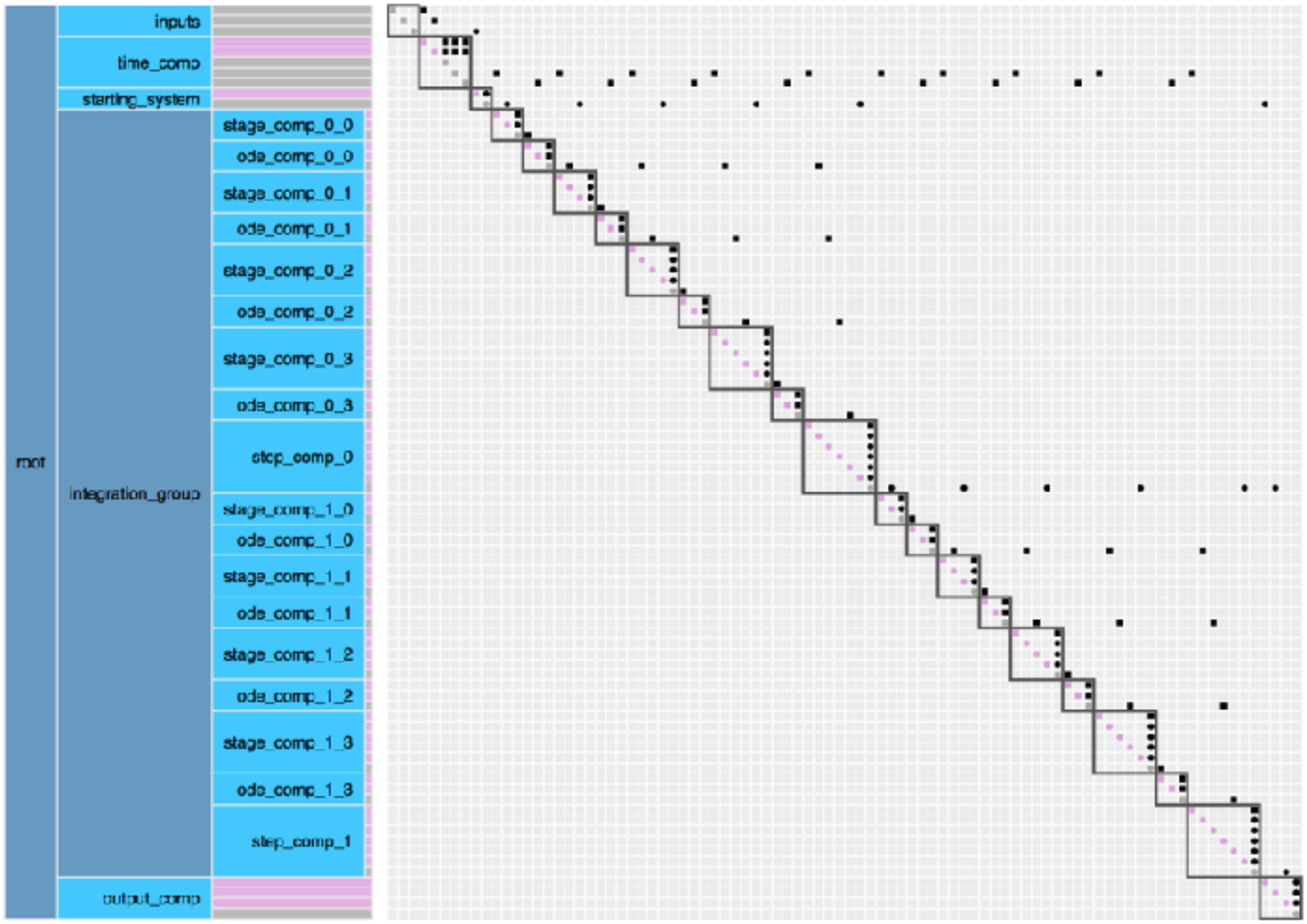
2 steps of forward Euler (time-marching)



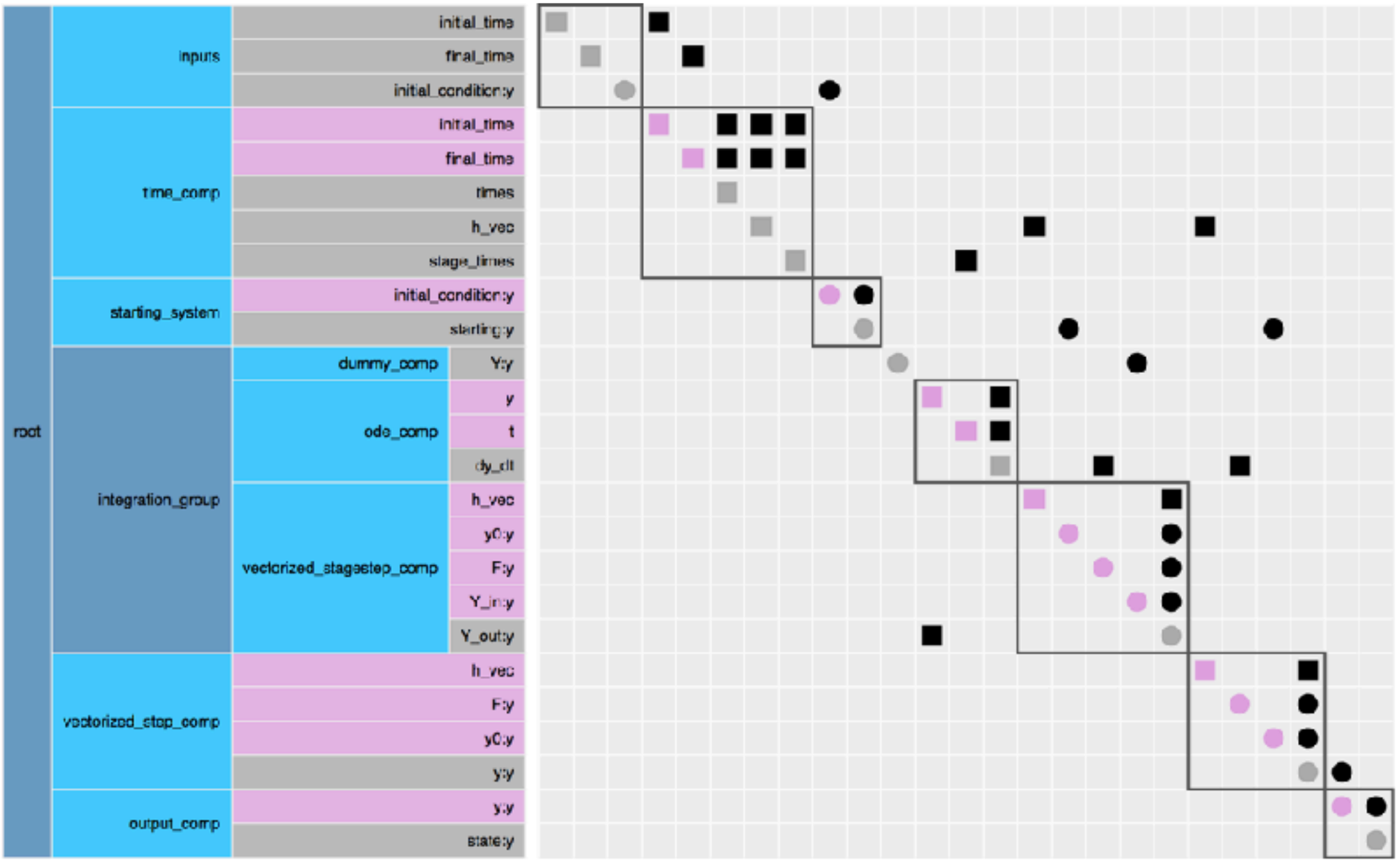
2 steps of backward Euler (time-marching)



2 steps of RK4 (time-marching)



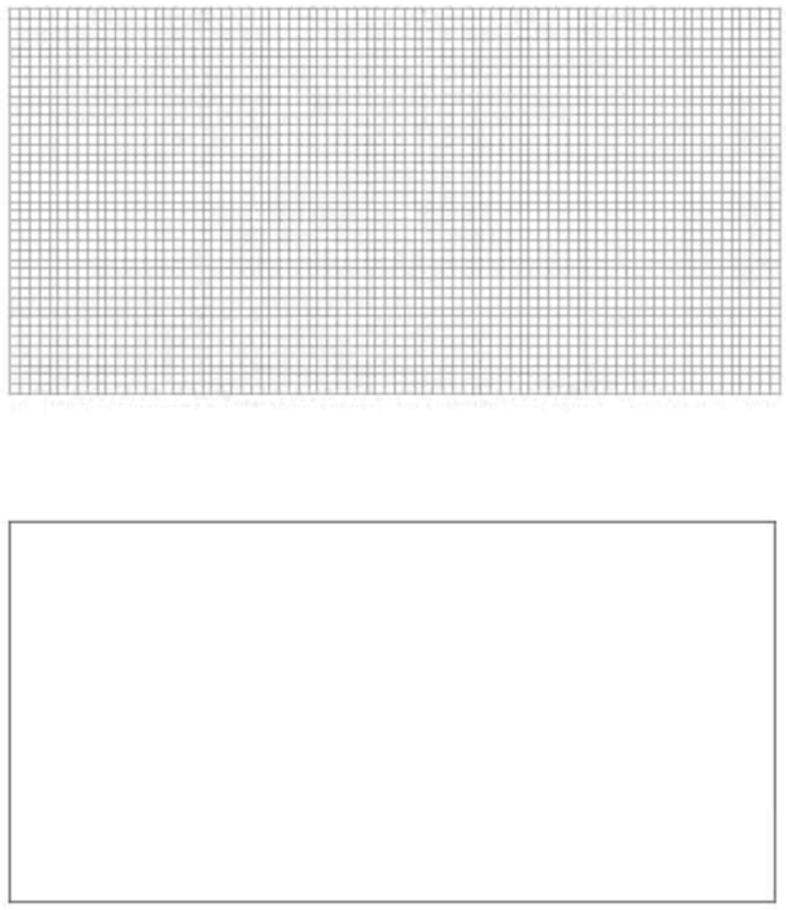
A new vectorized formulation

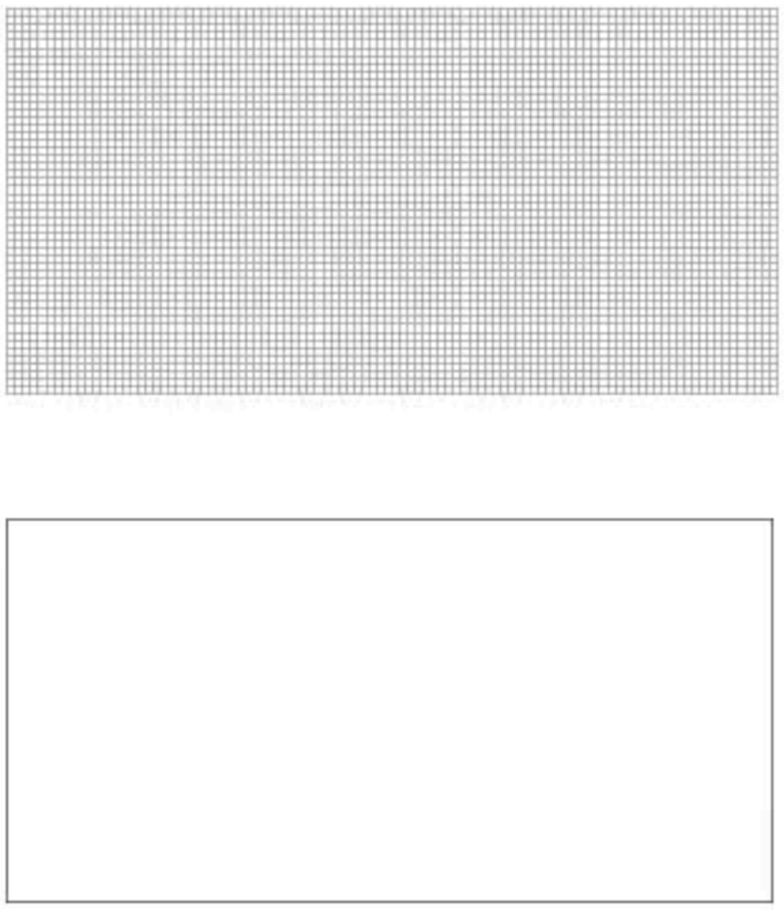


OpenMDAO v2 methods and applications

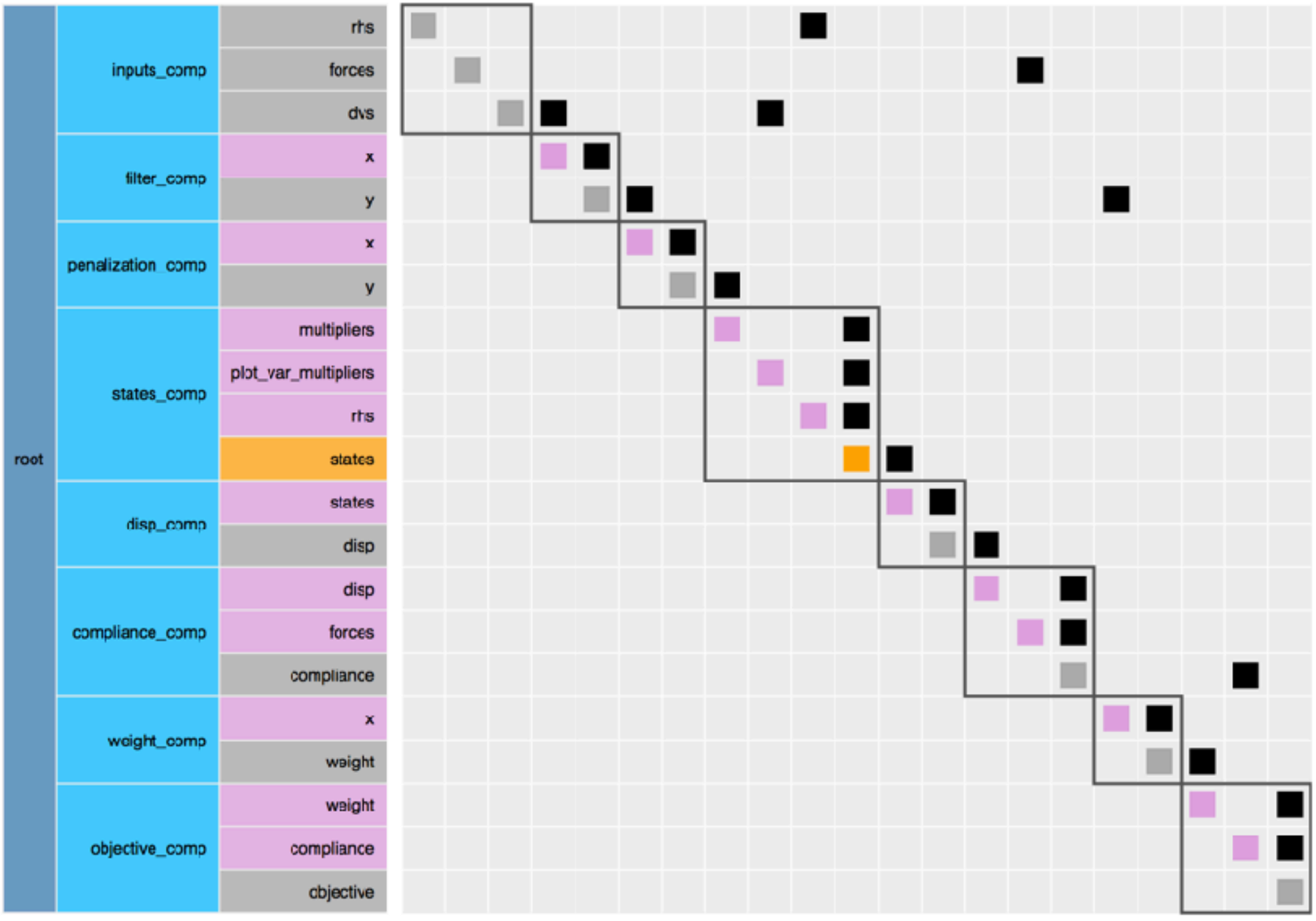
1. New methods
   - Reconfigurability
   - Ozone: ODE and optimal control solver library
2. Recent applications
   - Topology optimization
   - Aircraft design-allocation optimization
   - Electric aircraft MDO

Topology optimization (SIMP) treats every element's density as a continuous variable





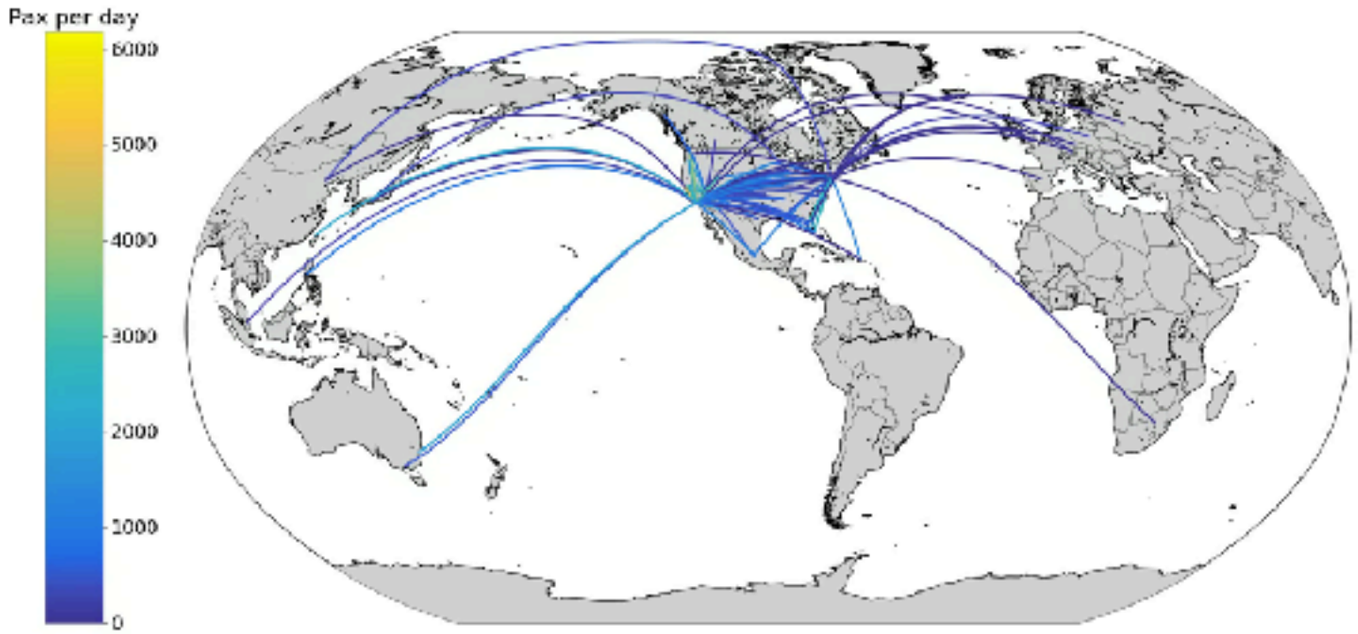
The 2-D FEM solver and the topology optimization are all entirely in OpenMDAO

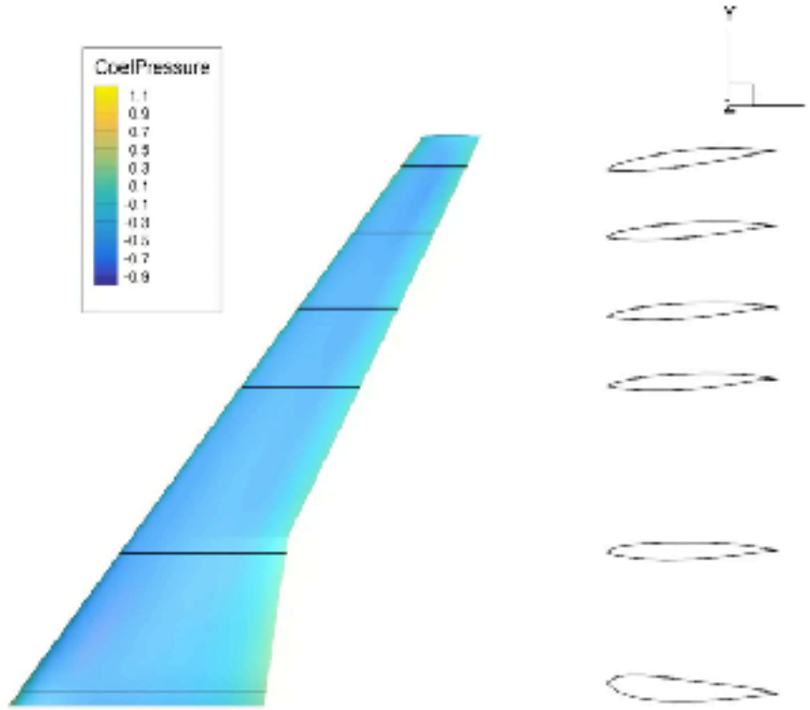


OpenMDAO v2 methods and applications

1. New methods
   - Reconfigurability
   - Ozone: ODE and optimal control solver library
2. Recent applications
   - Topology optimization
   - Aircraft design-allocation optimization
   - Electric aircraft MDO

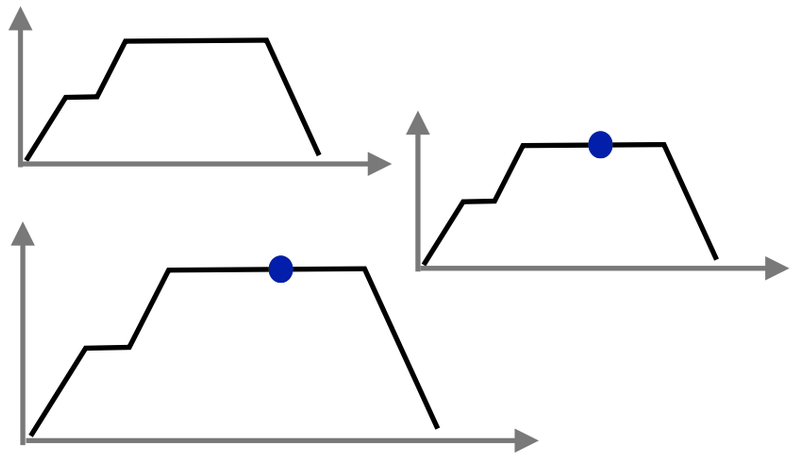
Allocation-mission-design optimization





Background: aerostructural optimization studies minimize estimated fuel burn

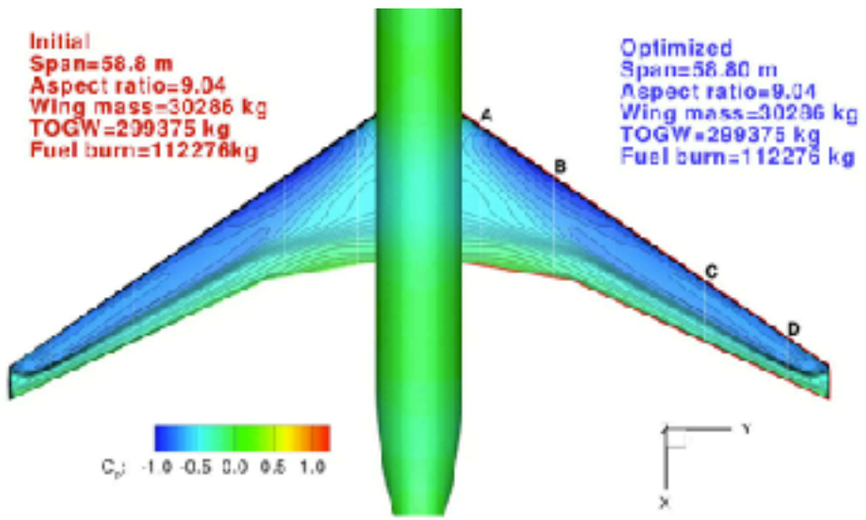
Select a single design mission range, R



Select points $(M_1, C_{L,1}) \ldots (M_n, C_{L,n})$ and their weights $w_1 \ldots w_n$

minimize $w_1\, fb(R, M_1, C_{L,1}) + \ldots + w_n\, fb(R, M_n, C_{L,n})$

with respect to the aircraft design



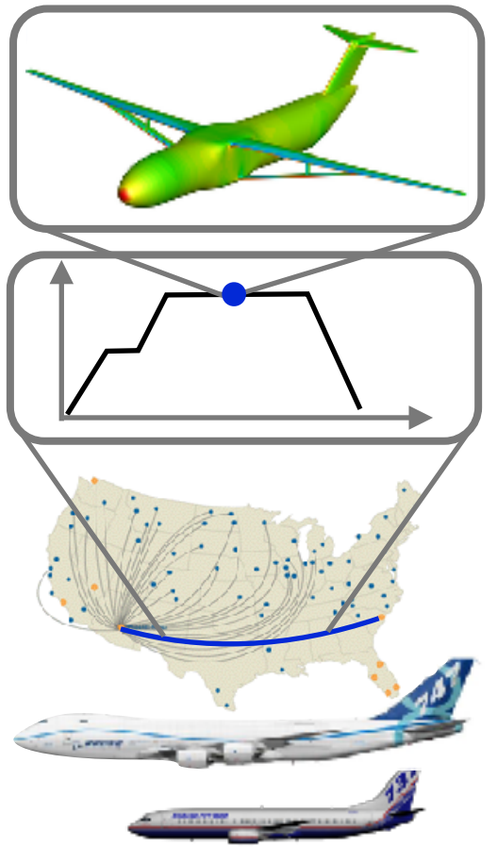
In reality, the allocation, mission profiles, and design should be optimized together

maximize airline industry-level profit

with respect to design: shape, twist, area, sweep

mission: Mach number, altitude profile

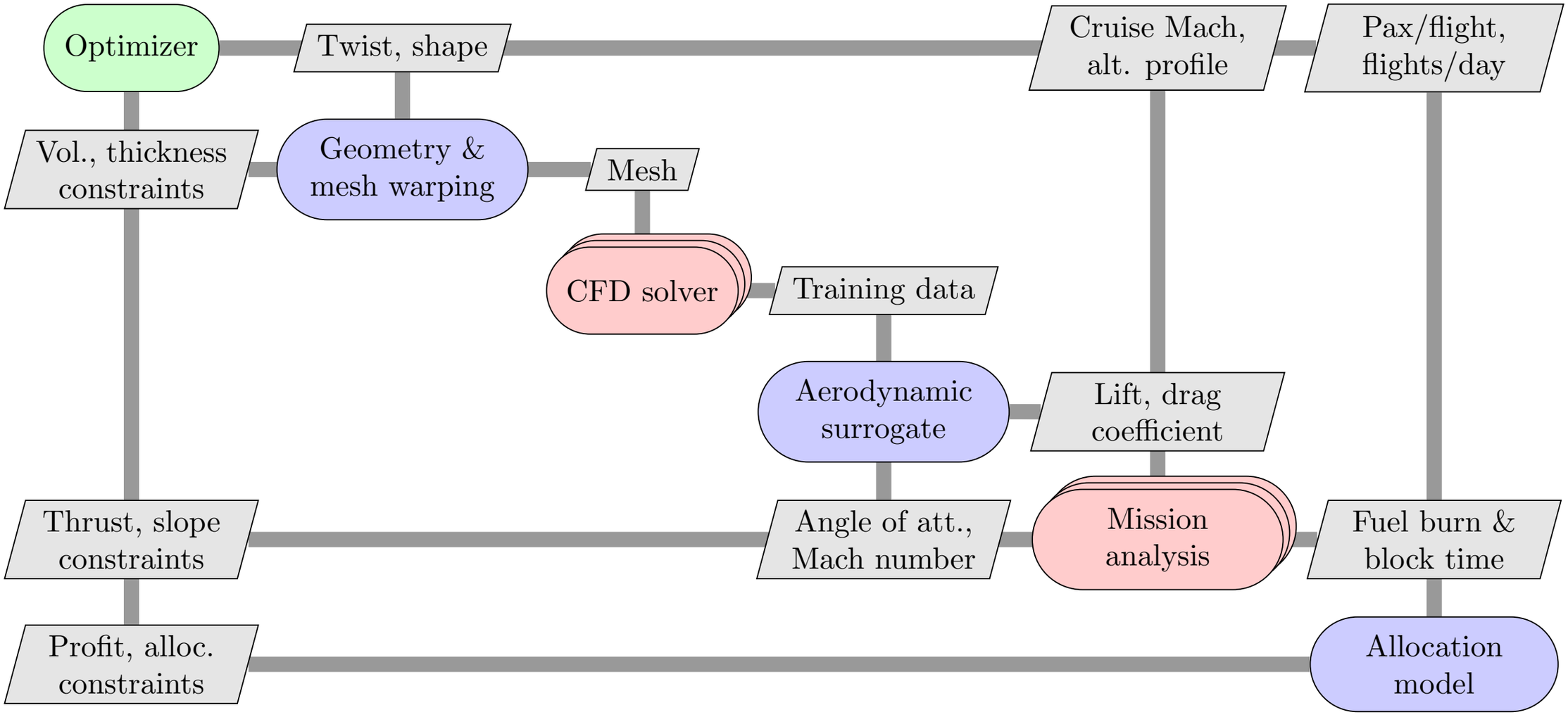
airline allocation: flights per day



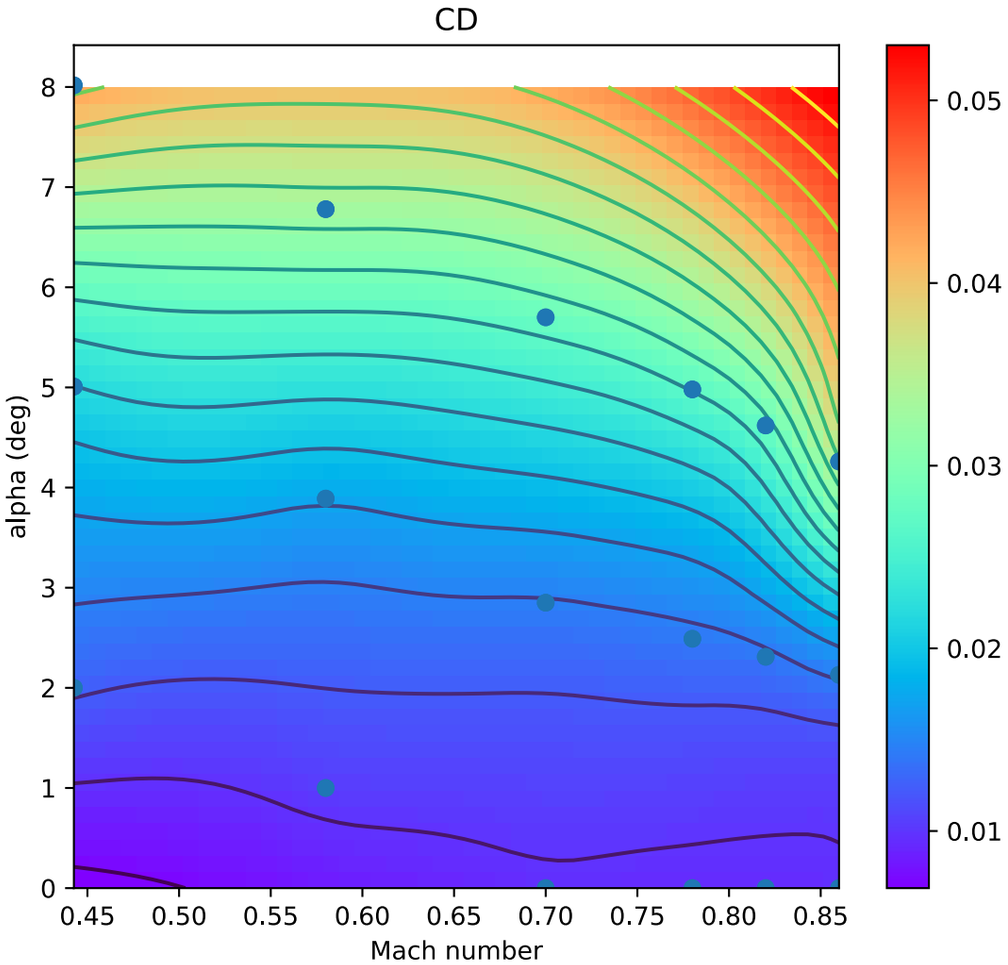
Why?

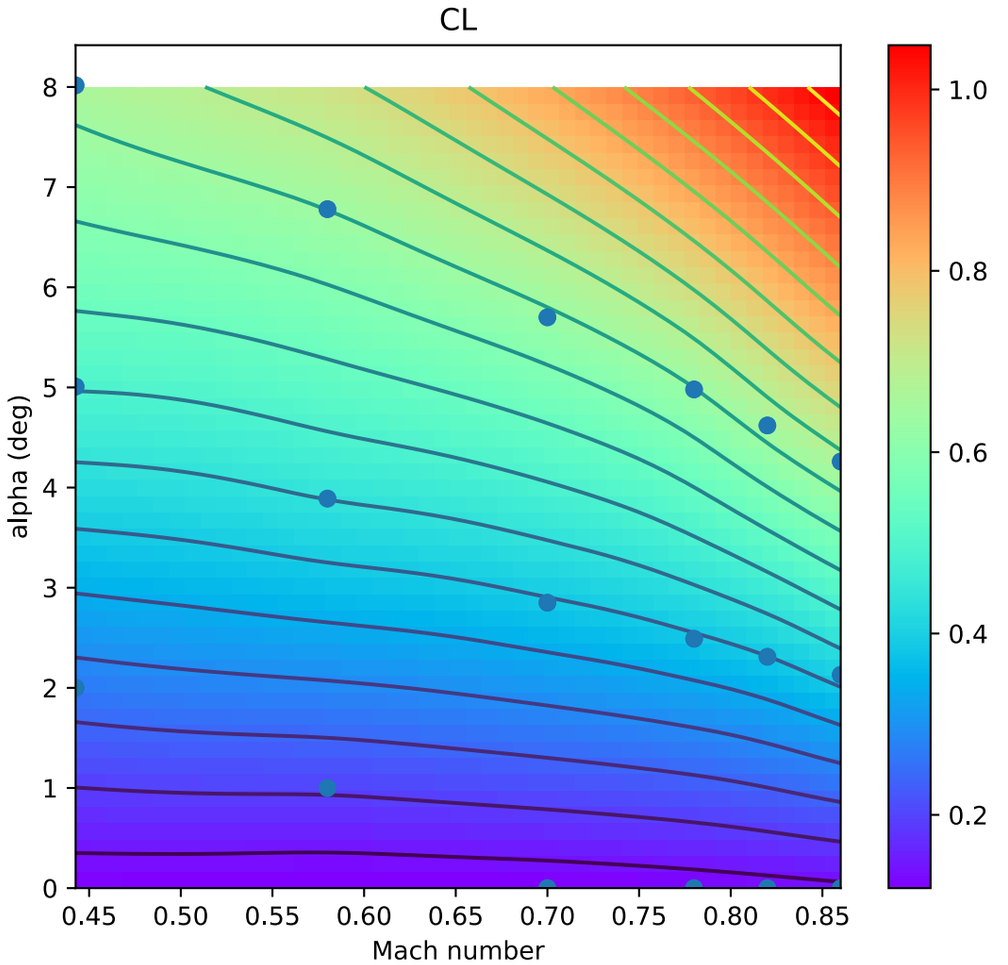
- Aircraft often flown below their design range
- Determine optimal design/sizing based on current fleets
- Model next-generation designs and morphing technology

36 CFD analyses are performed to train a surrogate model in the loop

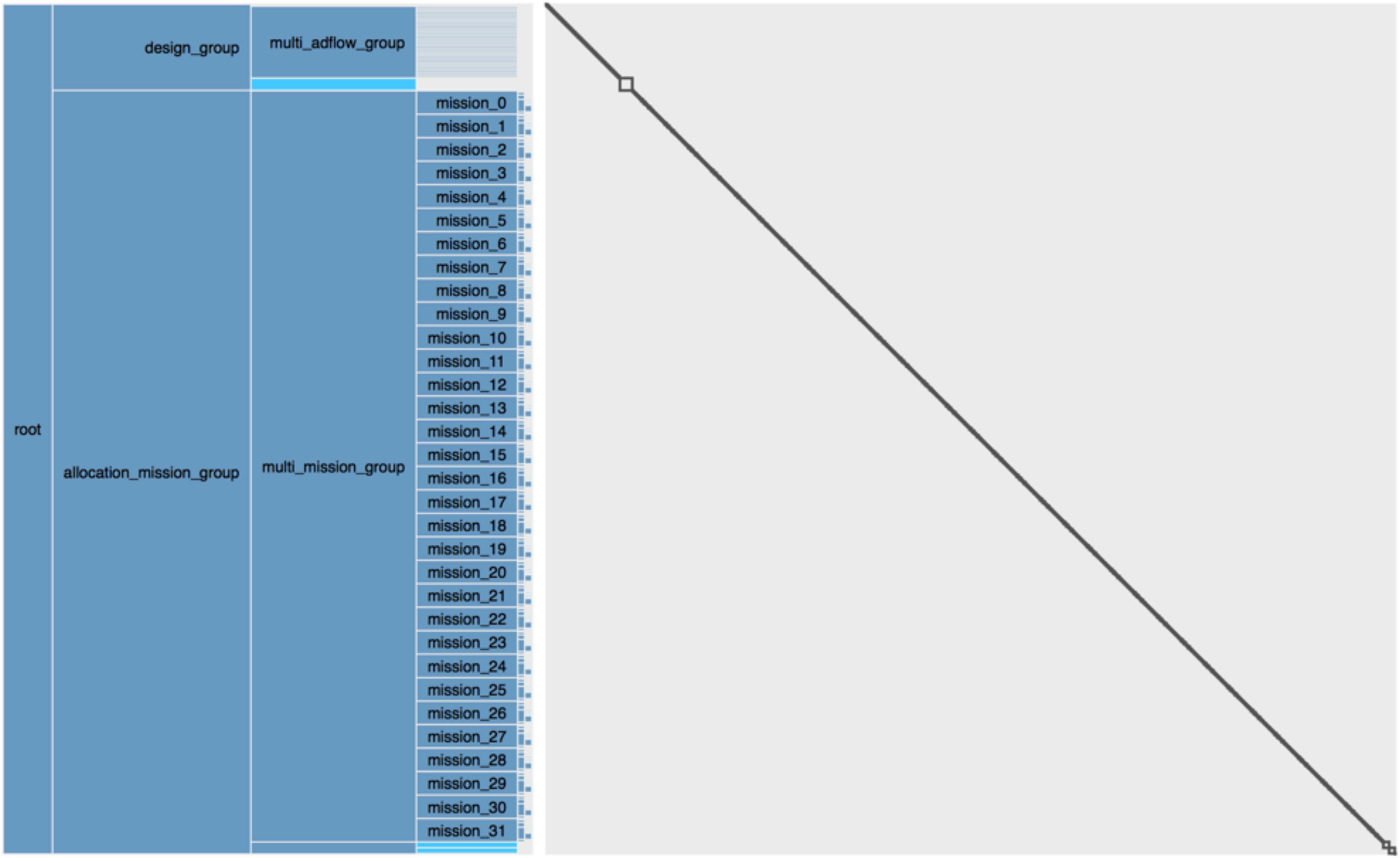


We developed a new surrogate model based on nonlinear minimal-energy splines

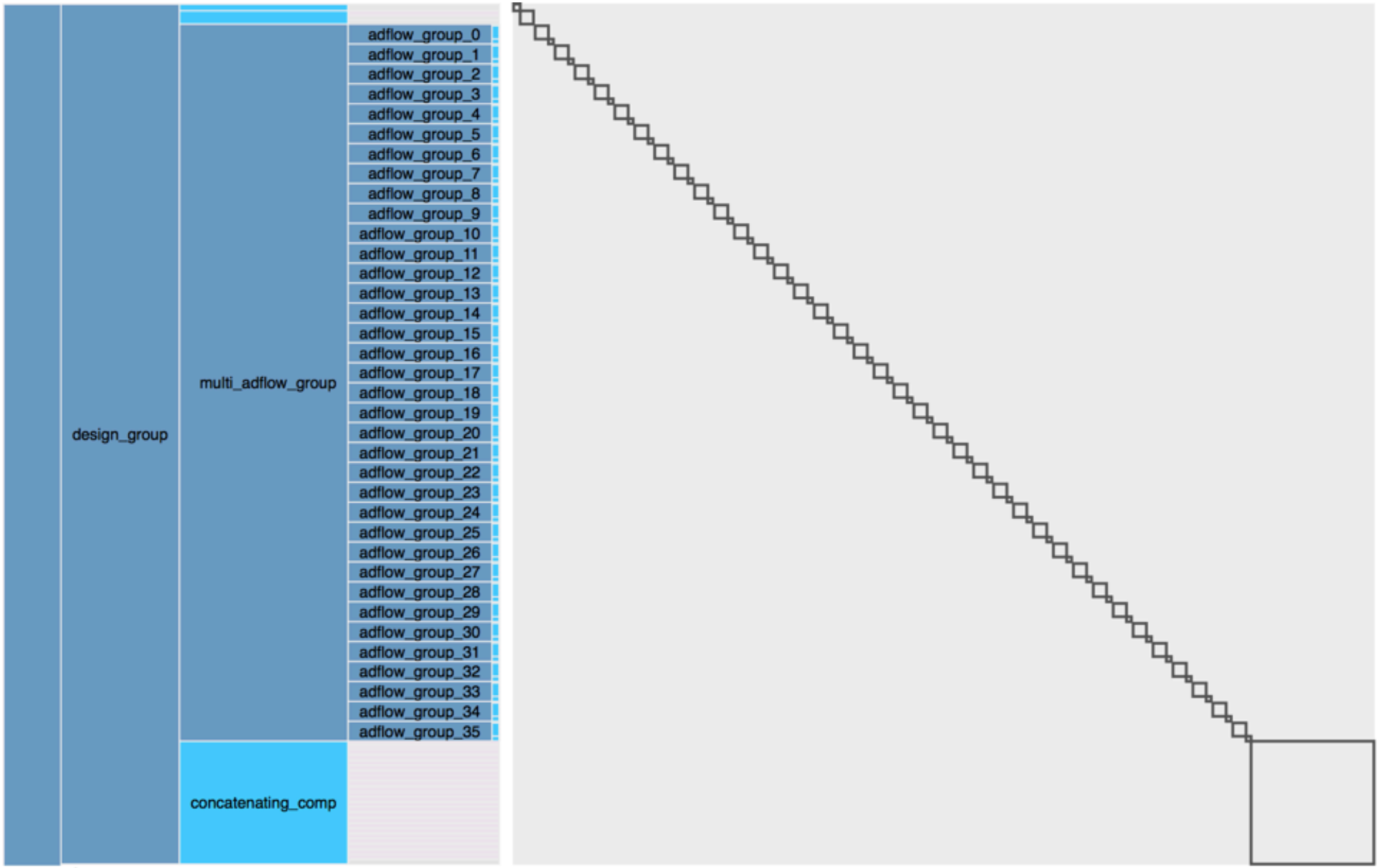




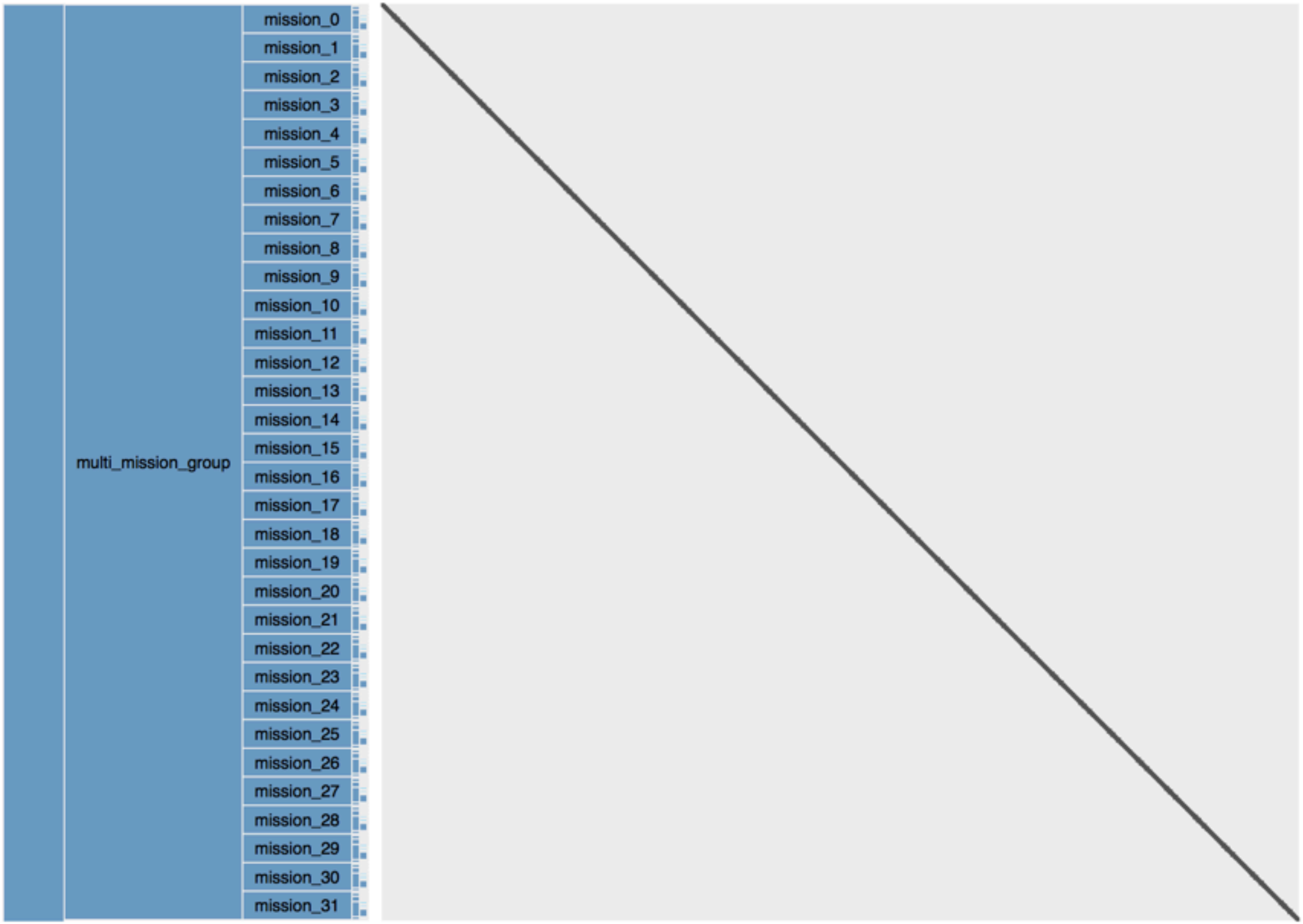
Overall model



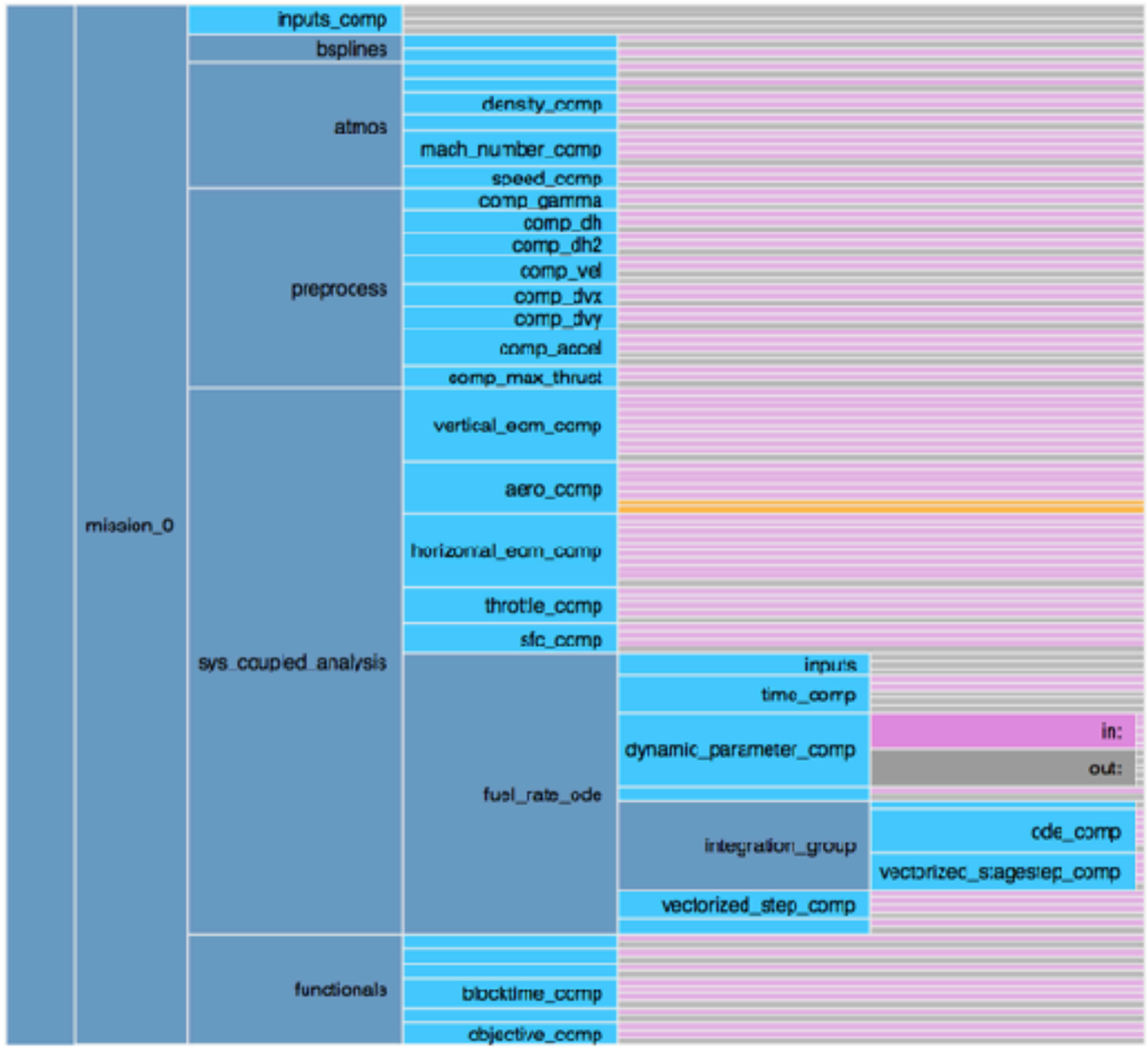
36 CFD groups

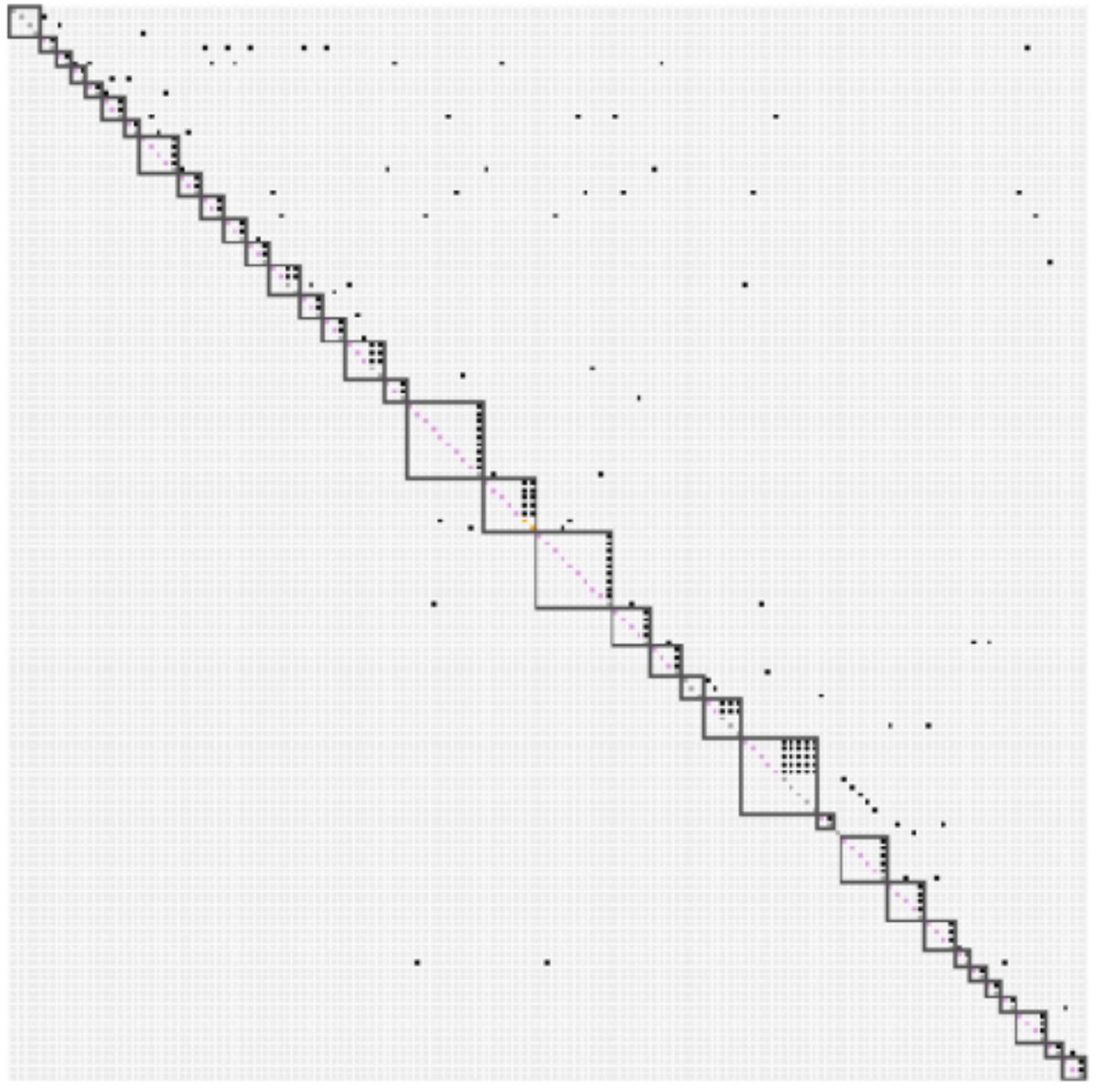


128 mission analysis groups



Inside a single mission analysis

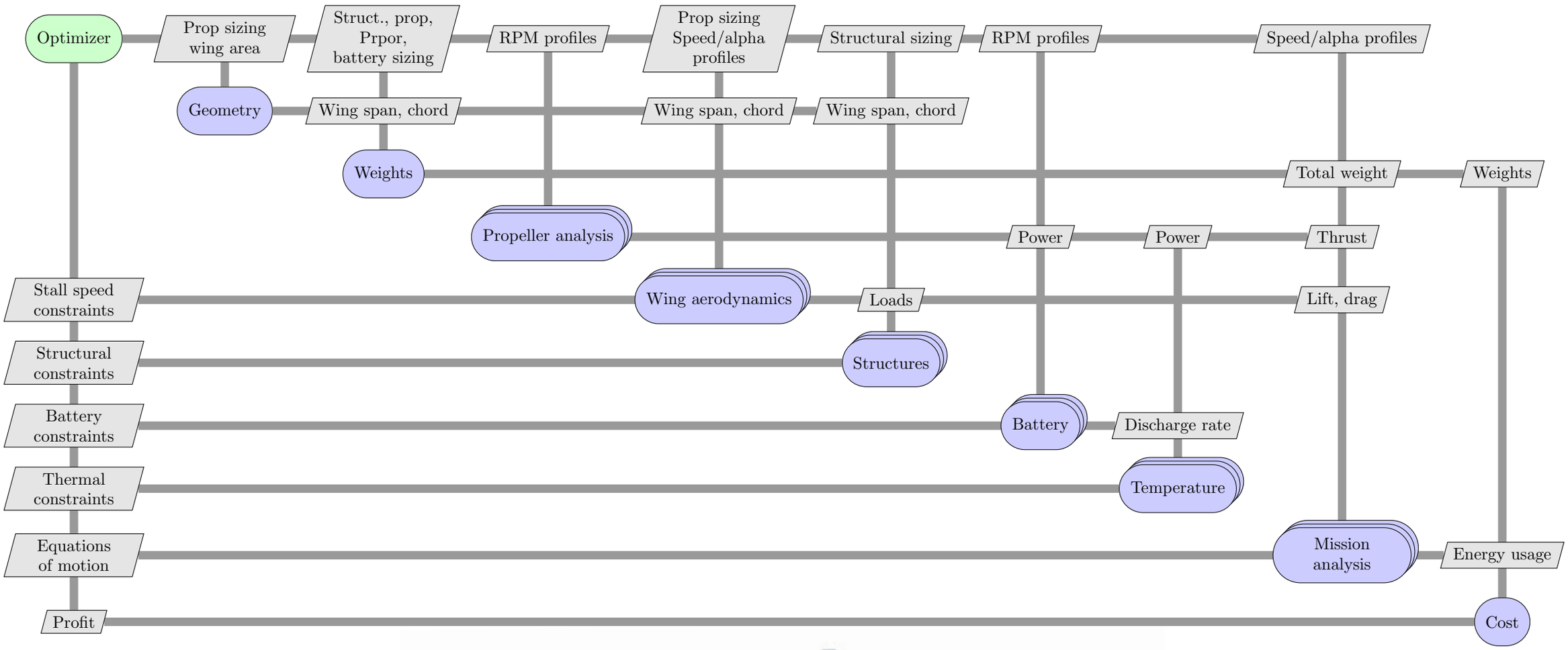




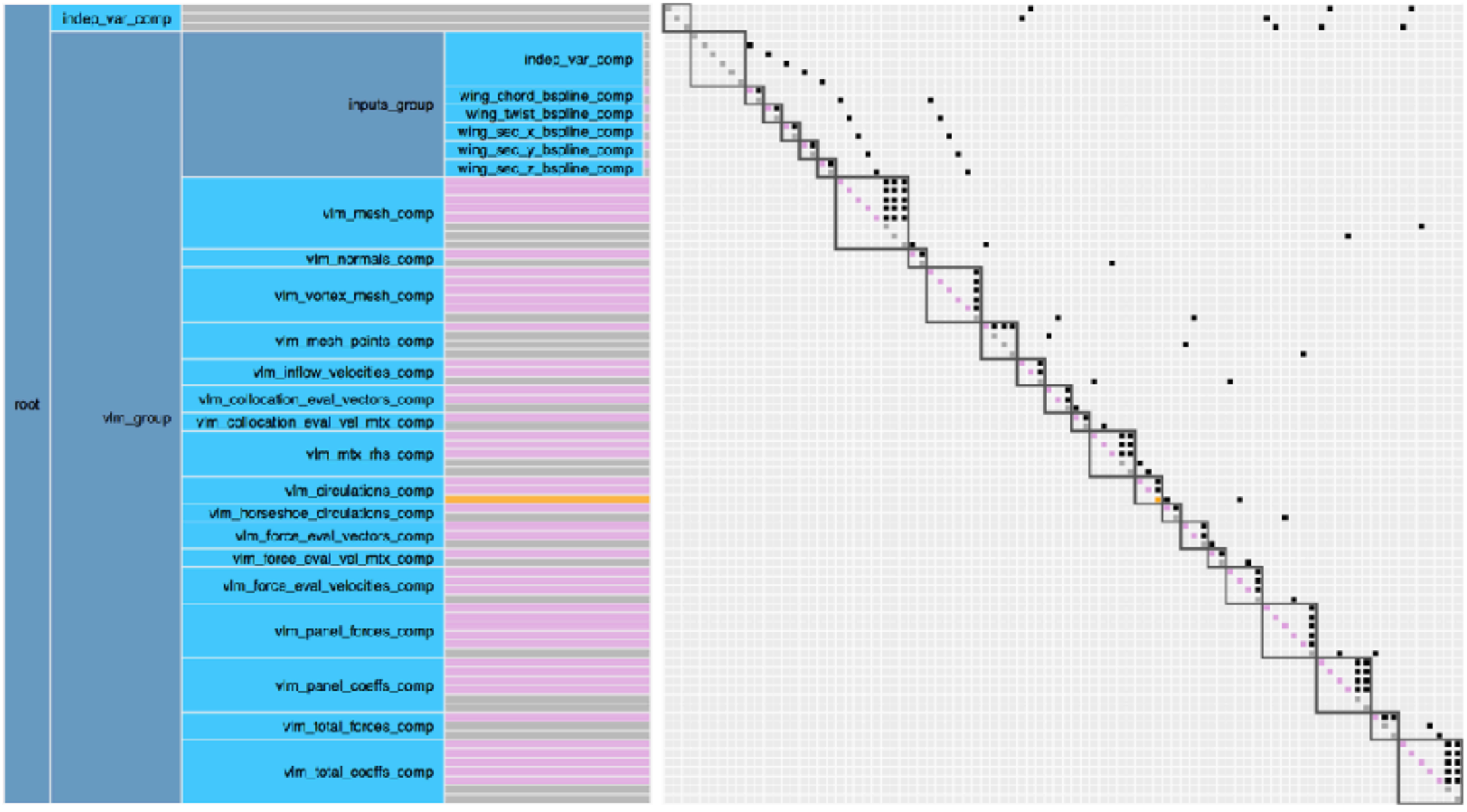
OpenMDAO v2 methods and applications

1. New methods
   - Reconfigurability
   - Ozone: ODE and optimal control solver library
2. Recent applications
   - Topology optimization
   - Aircraft design-allocation optimization
   - Electric aircraft MDO

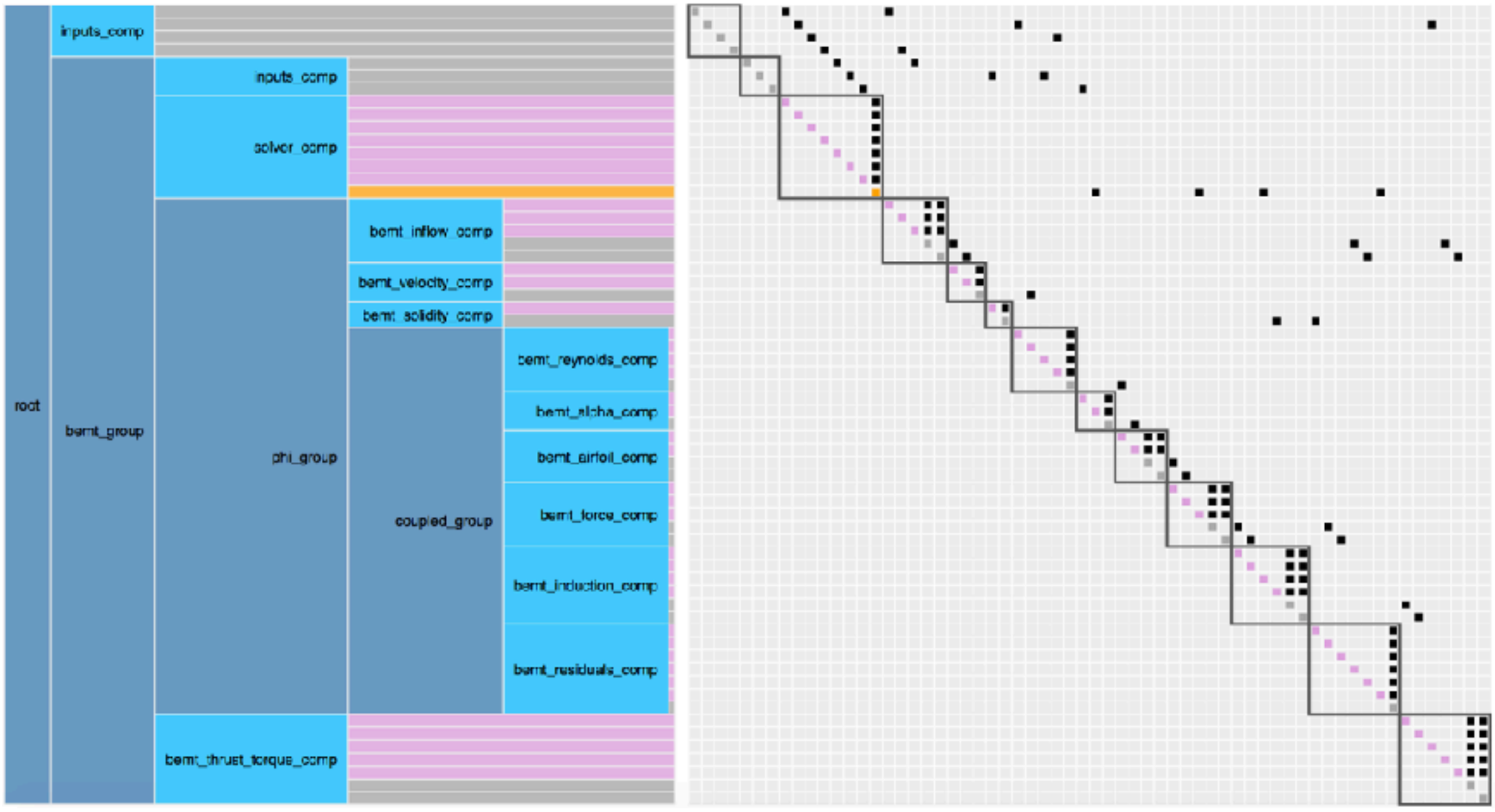
Large-scale design-mission optimization of an on-demand mobility electric aircraft



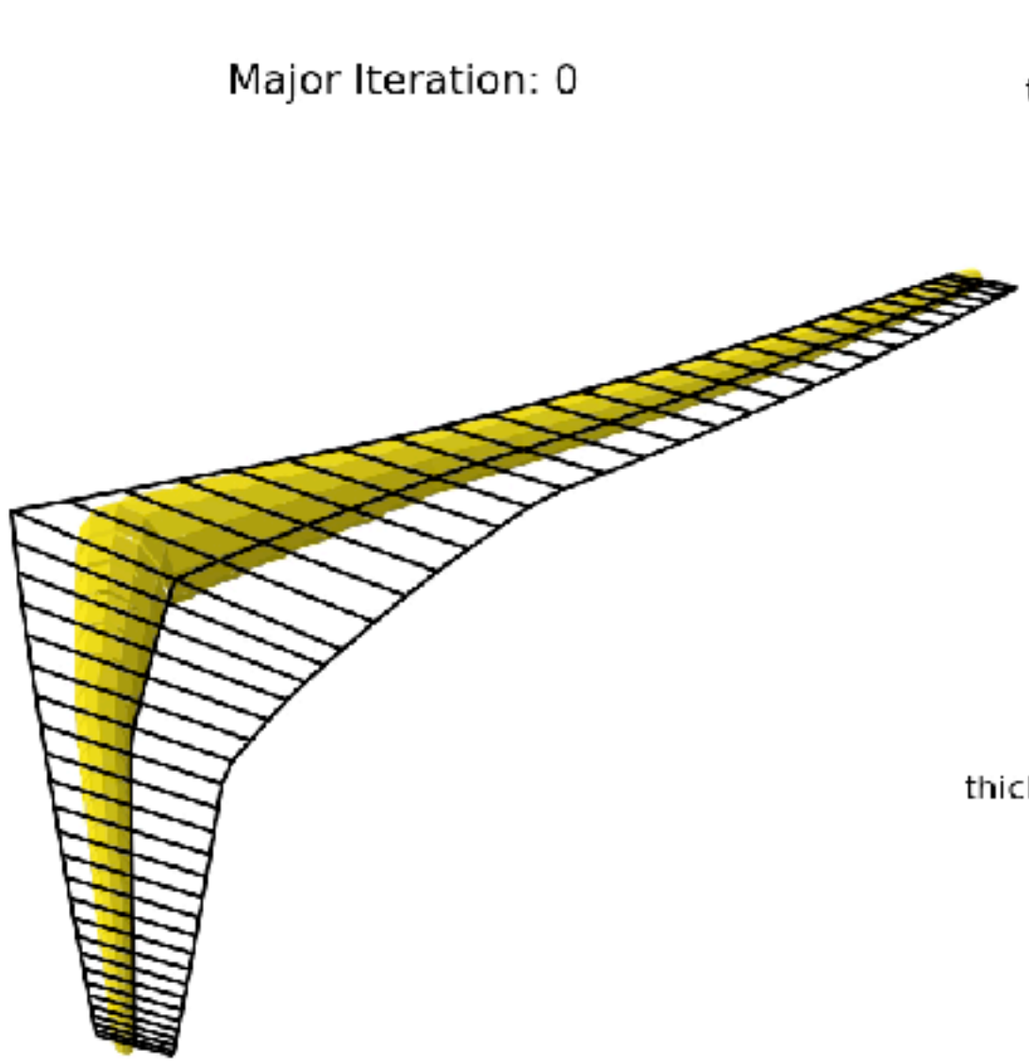
The 2-way propeller-wing interaction is modeled using the vortex lattice method

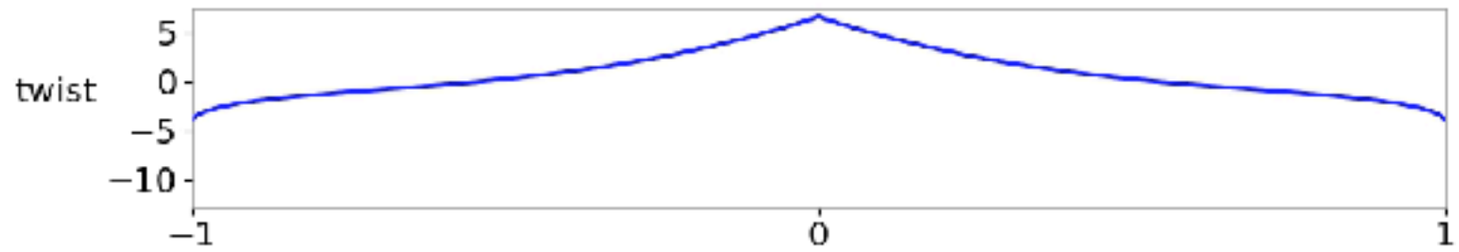


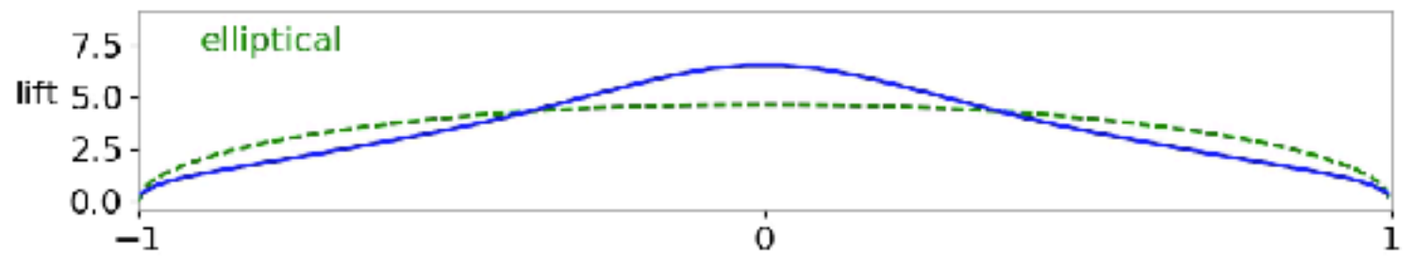
This is coupled to blade element momentum theory with slipstream evolution

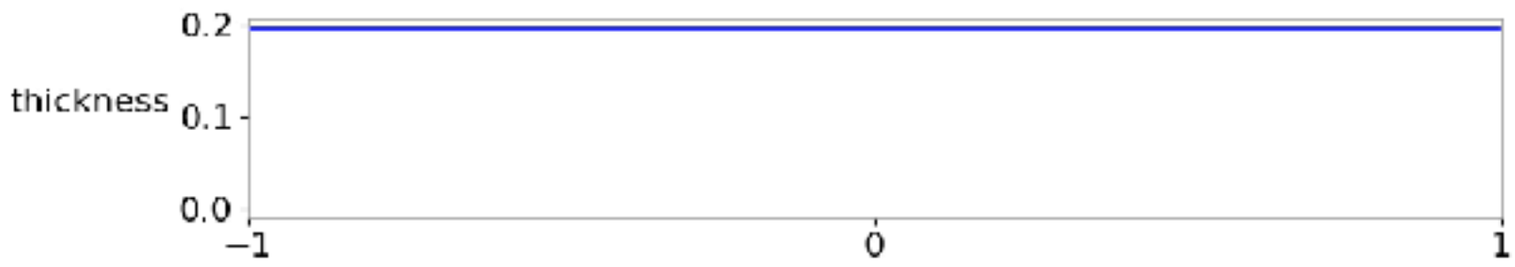


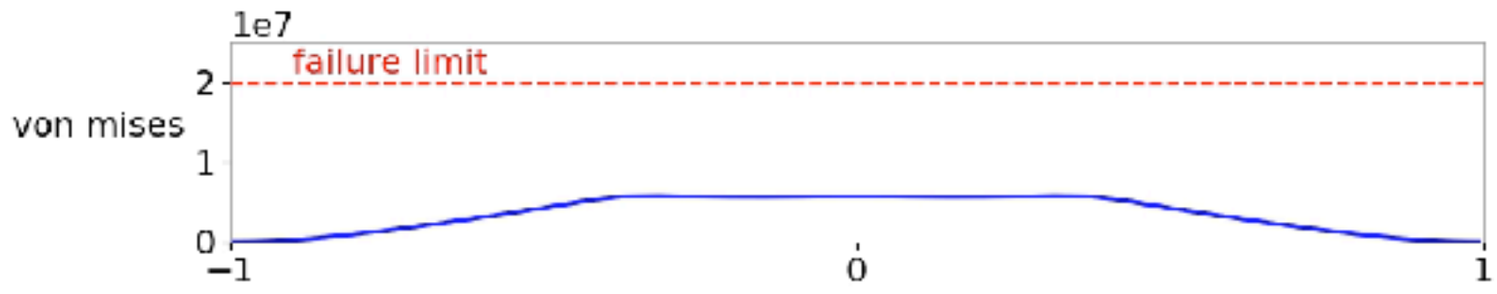
The VLM is also coupled to a 1-D FEA solver with spatial beam elements



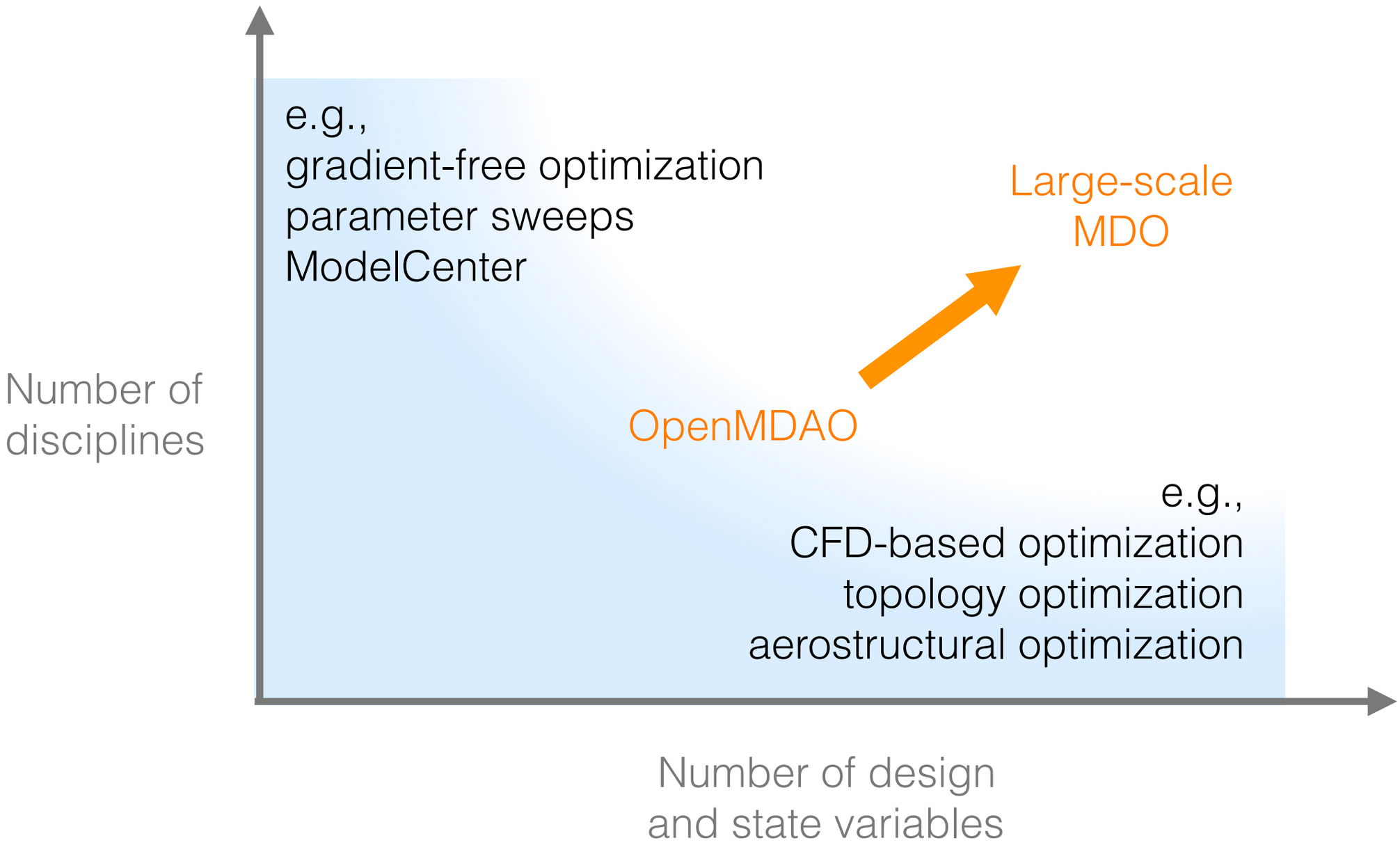








MAUD simplifies derivative computation, but many challenges remain for large-scale



Acknowledgments

- Transformational Tools and Technologies (T3) Project

- NASA Aeronautics Research Mission Directorate

- John P. Jasa for his help on the aircraft allocation-mission-design work

- Other collaborators: Prof. Joaquim R. R. A. Martins, Dr. Hayoung Chung, Prof. H. Alicia Kim, Drayton W. Munster, Prof. Andrew Ning

- The OpenMDAO team, especially Bret A. Naylor and Justin S. Gray

- Nathalie, Thierry, and Remi for organizing this workshop!

www.openmdao.org

www.nasa.gov