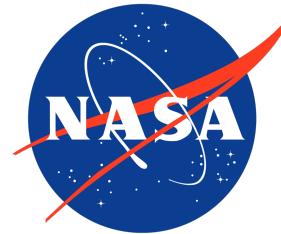


OpenMDAO ワークショップ



金子宗瑚

ミシガン大学 航空宇宙工学科

2023年1月12日 東京大学

スライドのダウンロード

```
$ git clone git@github.com:kanekosh/OpenMDAO_workshop_UTokyo.git
```

自己紹介：金子 宗瑚 (Kaneko, Shugo)

2019.3 九州大学 機械航空工学科卒業

指導教員：矢代茂樹、永井弘人

2019.9 ミシガン大学航空宇宙工学科 博士課程 入学

Multidisciplinary Design Optimization (MDO) Lab

指導教員：Joaquim R. R. A. Martins

2024.4 ミシガン大学卒業予定(?)



AEROSPACE
ENGINEERING
UNIVERSITY of MICHIGAN



本日の流れ

1. OpenMDAOの概要と理論
2. OpenMDAOを用いたオープンソースライブラリの紹介
3. 実践編

OpenMDAOとは？

- NASA Glenn開発の最適化フレームワーク
 - Python、オープンソース
-
- 背景の理論 MAUD¹ はミシガン大学 MDO Lab の考案
 - MDO Lab 卒業生が開発コアメンバーに複数



Google Scholar

🔍

記事

約 1,030 件 (0.02 秒)

期間指定なし
2023 年以降
2022 年以降
2019 年以降
期間を指定...

[HTML] **OpenMDAO**: An open-source framework for multidisciplinary design, analysis, and optimization

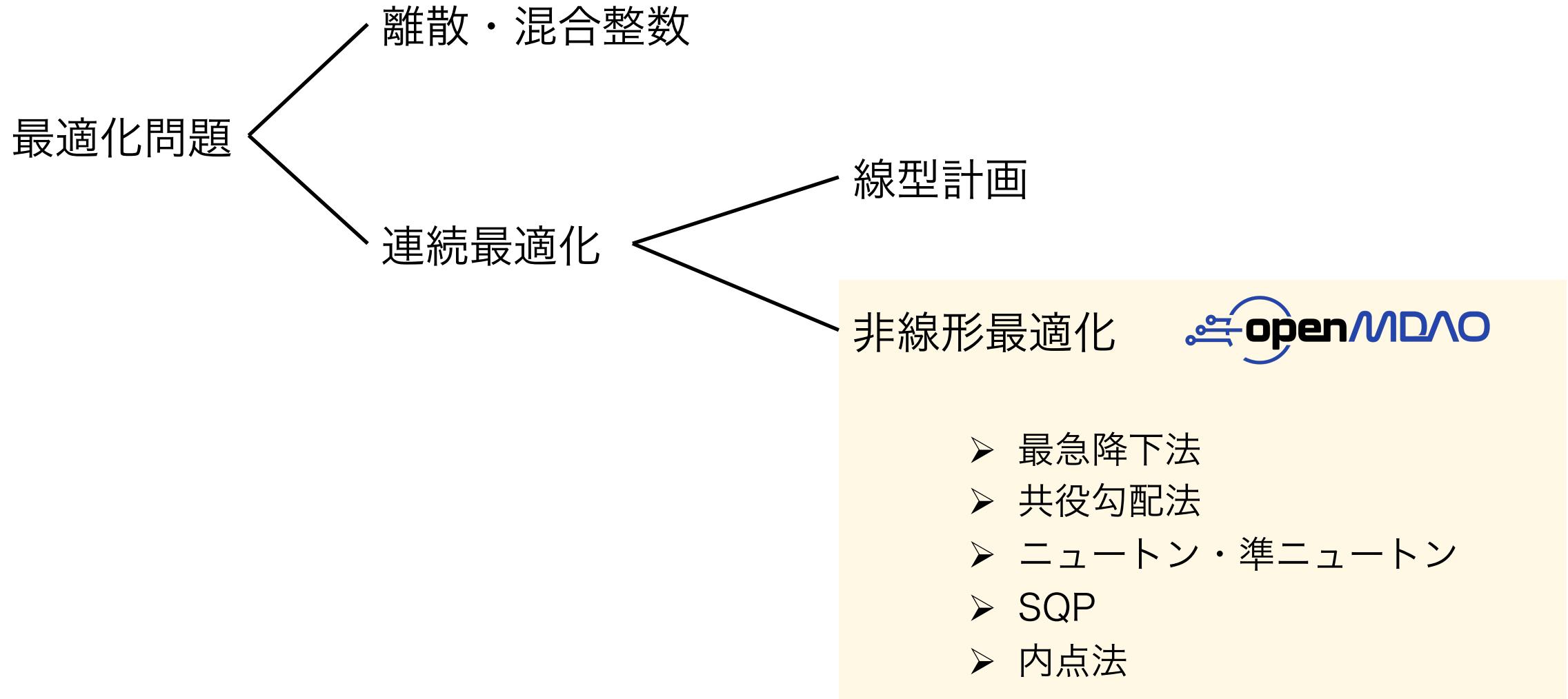
JS Gray, JT Hwang, JRRA Martins, KT Moore... - Structural and ..., 2019 - Springer

... OpenMDAO also provides a framework for computing ... We also summarize a number of OpenMDAO applications ... Given the potential of the OpenMDAO framework, we expect the ...

☆ 保存 羽引用 被引用数: 349 関連記事 全 8 バージョン

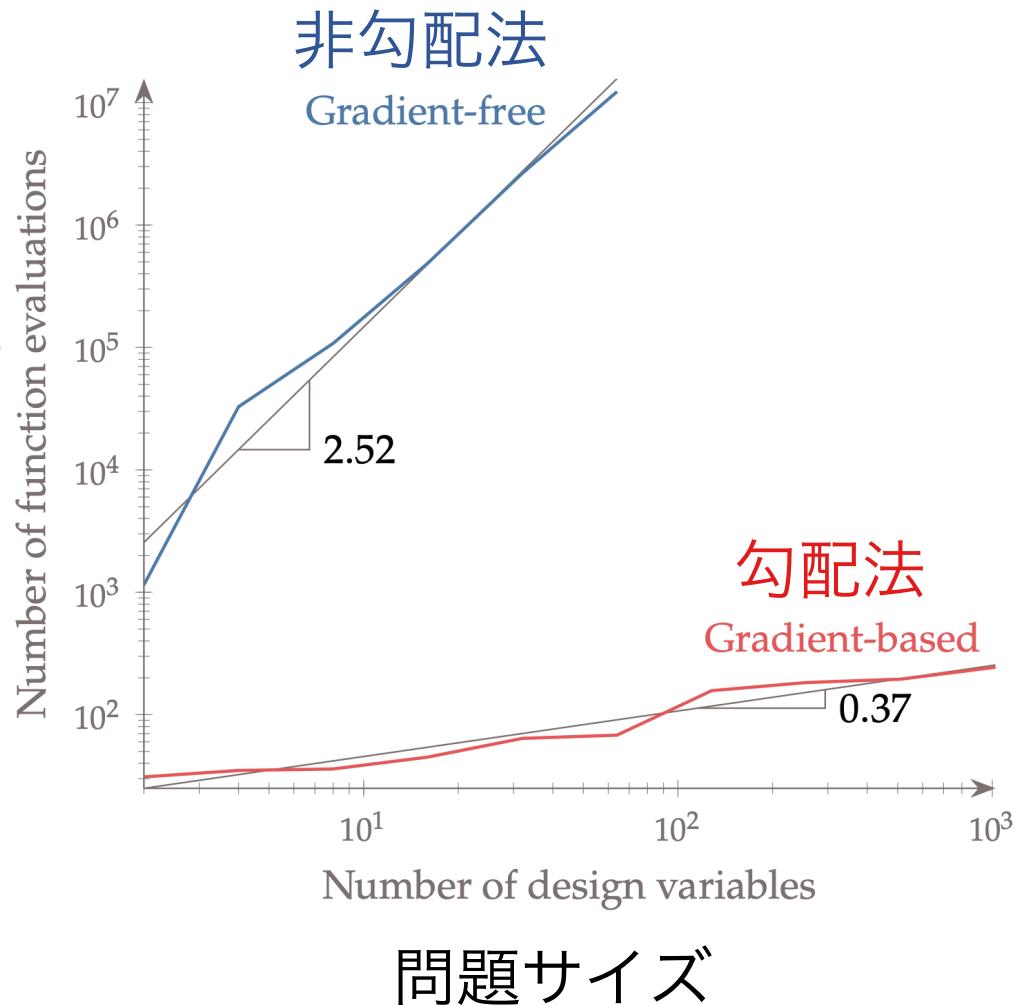
[1] Hwang and Martins, A computational architecture for coupling heterogenous numerical models and computing coupled derivatives, 2018, <https://dl.acm.org/doi/10.1145/3182393>

OpenMDAO：非線形連続最適化向け



連続最適化の解法：勾配法 vs 非勾配法

関数評価
の回数



問題サイズ

勾配法のメリット

- 少ない関数評価 → 低成本
- 大規模問題を解くことができる
- 数学的な局所最適条件 (KKT)

デメリット・注意点

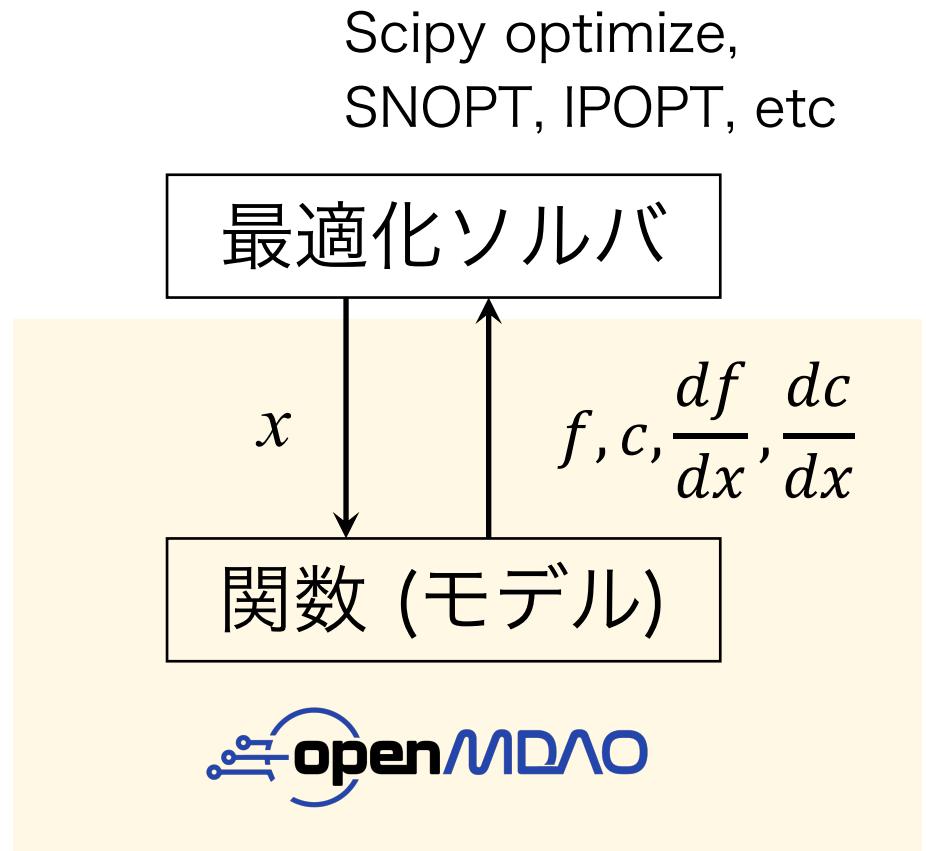
- 局所解に収束する可能性
- 勾配計算が必要
- 勾配が非正確だと収束しない



正確で低成本な勾配計算

OpenMDAOの役割

minimize $f(x)$
by varying x
subject to $c(x) \leq 0$



- 関数(モデル)をユーザーがOpenMDAO上で実装する
- 半自動で微分計算
- 最適化ソルバへのインターフェイス

勾配の計算方法

1. 有限差分近似 (finite difference)

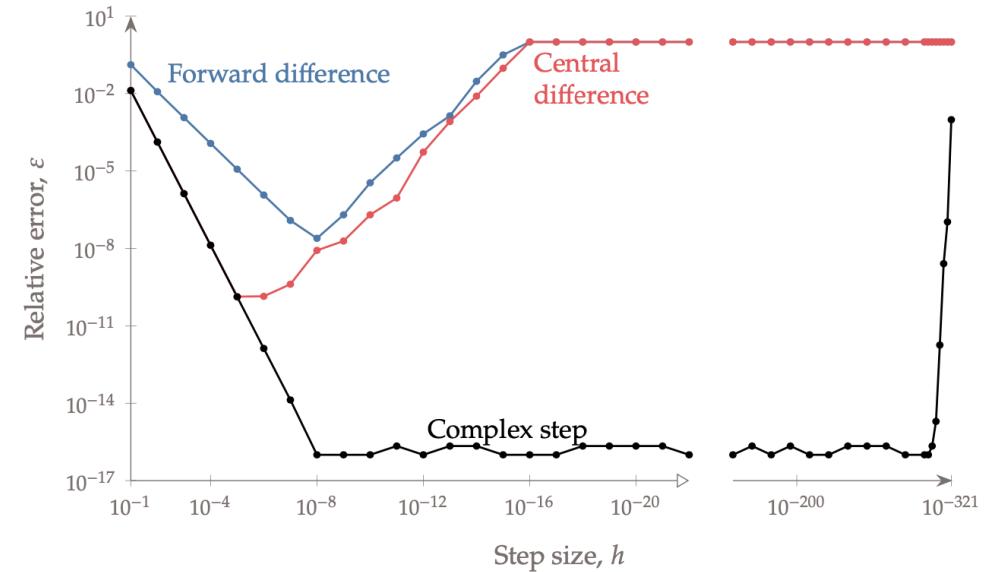
$$\frac{\partial f}{\partial x_j} = \frac{f(x + h\hat{e}_j) - f(x)}{h} + O(h)$$

計算コスト : $O(n_x)$ n_x : 設計変数の数

2. 複素ステップ法 (complex-step difference)

$$\frac{\partial f}{\partial x_j} = \frac{\text{Im} (f(x + ih\hat{e}_j))}{h} + O(h^2)$$

計算コスト : $O(n_x)$



ステップサイズのジレンマ

勾配の計算方法

3. 自動微分 (Automatic differentiation; Algorithmic differentiation, AD)

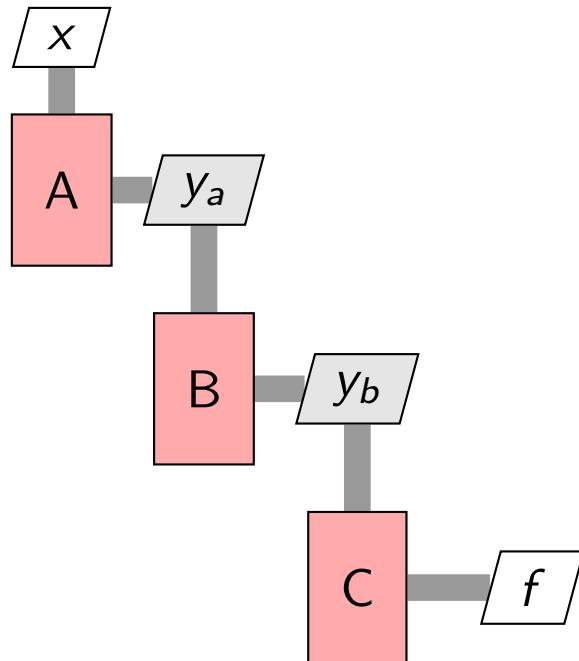
計算コスト : $O(n_x)$ (Forward mode) n_x : 設計変数の数
 $O(n_f)$ (Reverse mode) n_f : 関数の数

勾配の計算方法

4. 解析法 (Implicit analytic methods)



Feed-forwardモデルの場合

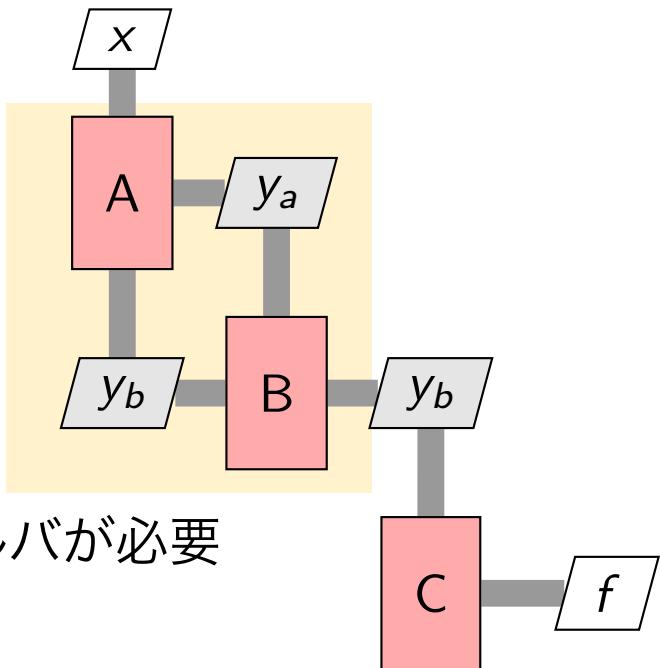


解析法 = 連鎖律 (Chain rule)

$$\frac{df}{dx} = \frac{\partial f}{\partial y_b} \frac{\partial y_b}{\partial y_a} \frac{\partial y_a}{\partial x}$$

連成モデルのChain ruleは複雑

連成モデルの場合

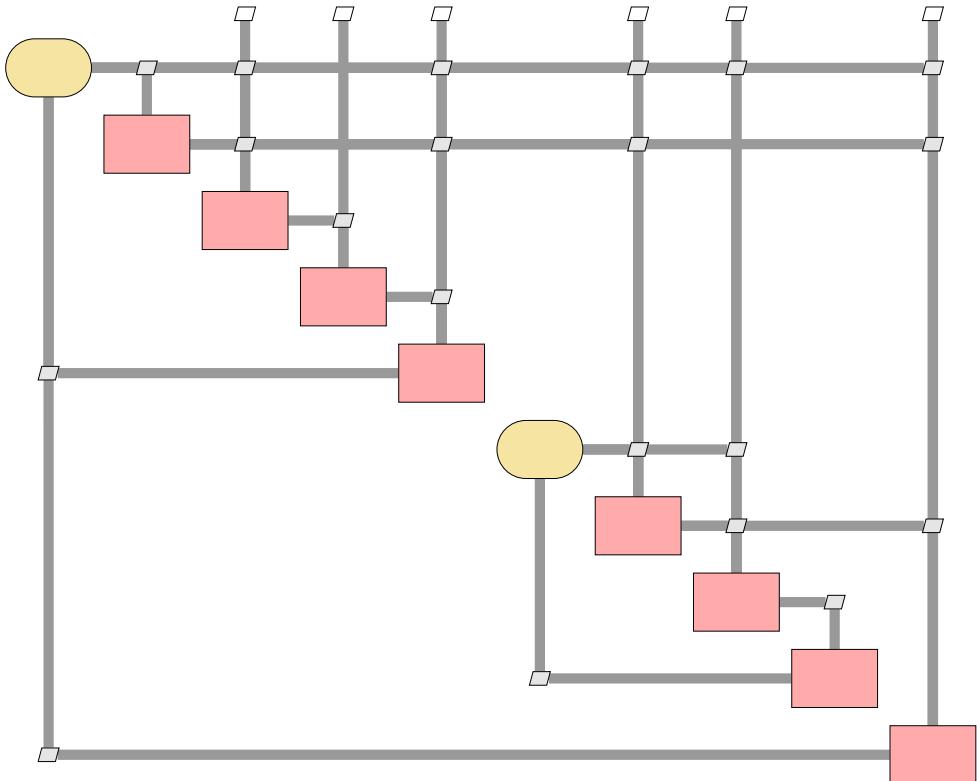


連鎖律 (Chain rule)

$$\frac{df}{dx} = \frac{\partial f}{\partial y_b} \frac{\partial y_b}{\partial y_a} \frac{\partial y_a}{\partial x} + \frac{\partial f}{\partial y_b} \frac{dy_b}{dx} + \frac{\partial f}{\partial y_a} \frac{dy_a}{dx}$$

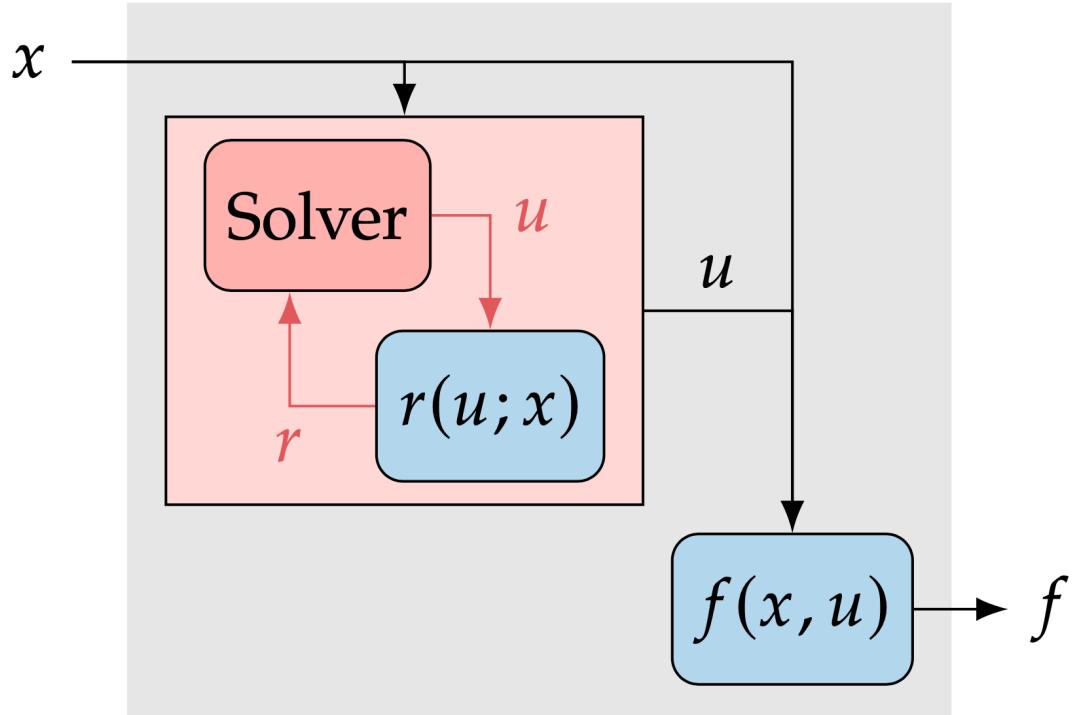
OpenMDAOは、解析法による微分計算を自動で行ってくれる

より複雑な連成モデルの場合



$$\frac{df}{dx} = [\text{ask } \text{openMDAO}]$$

解析法 – 状態変数と残差方程式



x : 設計変数
 f : 目的関数・制約関数
 u : 状態変数 (states)
 r : 残差方程式 (rediduals)
$$r(u; x) = 0$$

例1：有限要素法

$$u = \text{節点変位}, \quad r = Ku - f$$

例2：CFD

$$u = [\rho, \rho u, \rho v, \rho e],$$

解析法 – Direct法 & Adjoint法の導出 (1)

連鎖律(chain rule)より、

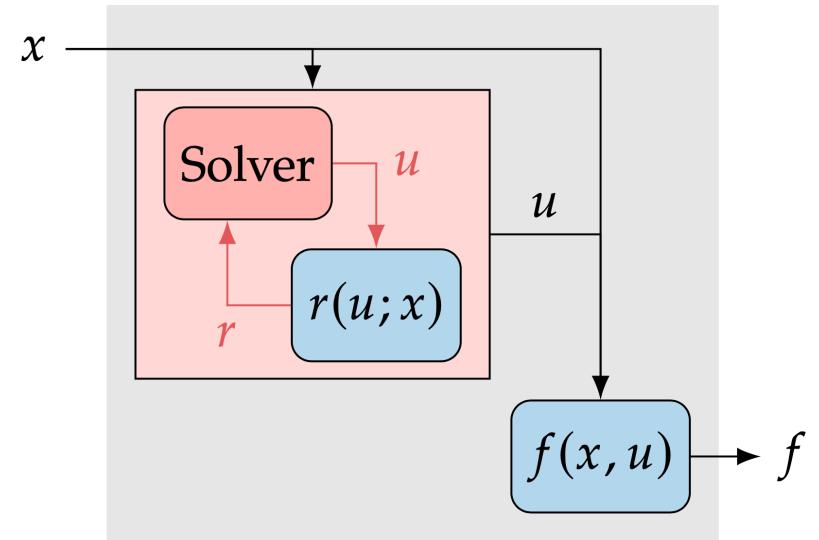
$$\frac{df}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial u} \frac{du}{dx}$$

残差方程式は常に満たされる必要があるから、

$$dr = \frac{\partial r}{\partial x} dx + \frac{\partial r}{\partial u} du = 0$$

上2式より du/dx を消去すると、

$$\frac{df}{dx} = \frac{\partial f}{\partial x} - \frac{\partial f}{\partial u} \frac{\partial r}{\partial u}^{-1} \frac{\partial r}{\partial x}$$



x : 設計変数
 f : 目的関数・制約関数
 u : 状態変数 (states)
 r : 残差方程式 (rediduals)
 $r(u; x) = 0$

解析法 – Direct法 & Adjoint法の導出 (2)

(続き)

$$\frac{df}{dx} = \frac{\partial f}{\partial x} - \frac{\partial f}{\partial u} \frac{\partial r}{\partial u}^{-1} \frac{\partial r}{\partial x}$$

$$\frac{df}{dx} = \begin{matrix} (n_f \times n_x) & (n_f \times n_x) \end{matrix} \begin{matrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial u} \end{matrix} - \underbrace{\begin{matrix} (n_f \times n_u) & (n_u \times n_u) \end{matrix} \begin{matrix} \frac{\partial r}{\partial u}^{-1} \\ \frac{\partial r}{\partial x} \end{matrix}}_{\psi^\top (n_f \times n_u)}$$

$\phi \quad (n_u \times n_x)$

Direct法：コスト $O(n_x)$

1. 偏微分ヤコビ行列を計算する

2. 線型システムを解く : $\frac{\partial r}{\partial u} \phi = \frac{\partial r}{\partial x}$

3. 全微分を計算 : $\frac{df}{dx} = \frac{\partial f}{\partial x} - \frac{\partial f}{\partial u} \phi$

Adjoint法：コスト $O(n_f)$

1. 偏微分ヤコビ行列を計算する

2. 線型システムを解く : $\frac{\partial r}{\partial u}^\top \psi = \frac{\partial f}{\partial u}^\top$

3. 全微分を計算 : $\frac{df}{dx} = \frac{\partial f}{\partial x} - \psi^\top \frac{\partial r}{\partial x}$

OpenMDAOによるAdjoint/Direct法による勾配計算

1. 各コンポーネントの関数・偏微分をユーザーが実装する

- 手計算
- 自動微分
- 有限差分 or Complex-step

2. 偏微分ヤコビ行列 $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial u}, \frac{\partial r}{\partial x}, \frac{\partial r}{\partial u}$ ・ 線型システムを組み立て

- Unified derivative equation (UDE)
- Modular analysis and unified derivatives (MAUD)

Martins and Ning, Engineering Design Optimization, Chapter 6 and 13.

3. 線型システムを解く：Adjoint or Direct or 合わせ技

- 問題サイズによって適した手法を、OpenMDAOが自動で選択

4. 全微分 $\frac{df}{dx}$ を計算

OpenMDAOによるAdjoint/Direct法による勾配計算

1. 各コンポーネントの関数・偏微分をユーザーが実装する

- 手計算
- 自動微分
- 有限差分 or Complex-step

2. 偏微分ヤコビ行列・線型システムを組み立て

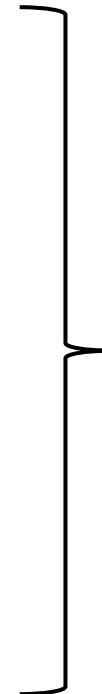
- Unified derivative equation (UDE)
- Modular analysis and unified derivatives (MAUD)

Martins and Ning, Engineering Design Optimization, Chapter 6 and 13.

3. 線型システムを解く：Adjoint or Direct or 合わせ技

- 問題サイズによって適した手法を、OpenMDAOが自動で選択

4. 全微分 $\frac{df}{dx}$ を計算



まとめ：OpenMDAOの理論と概要

- OpenMDAOは非線形連続最適化向け。
 - モデル (関数)とその勾配の実装
 - 最適化ソルバ (例: scipy optimize) とのインターフェイス
- 大規模な最適化問題を解くには、勾配法が必要。
- OpenMDAOは、正確で低成本な勾配計算の実装を簡易化
 - コンポーネントベースのモデル実装
 - Direct & Adjoint法
 - Modular analysis and unified derivatives (MAUD)

参考リソース（理論）



OpenMDAO

Search this book...

Include Source Docs

Welcome to OpenMDAO

Welcome to OpenMDAO

OpenMDAO is an open-source high-performance computing platform for systems analysis and multidisciplinary optimization, written in Python. It enables you to decompose your models, making them easier to build and maintain, while still solving them in a tightly coupled manner with efficient parallel numerical methods.

The OpenMDAO project is primarily focused on supporting gradient-based optimization with analytic derivatives to allow you to explore large design spaces with hundreds or thousands of design variables, but the framework also has a number of parallel computing features that can work with gradient-free optimization, mixed-integer nonlinear programming, and traditional design space exploration.

<https://openmdao.org/newdocs/versions/latest/main.html>

Structural and Multidisciplinary Optimization (2019) 59:1075–1104
<https://doi.org/10.1007/s00158-019-02211-z>

RESEARCH PAPER

OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization

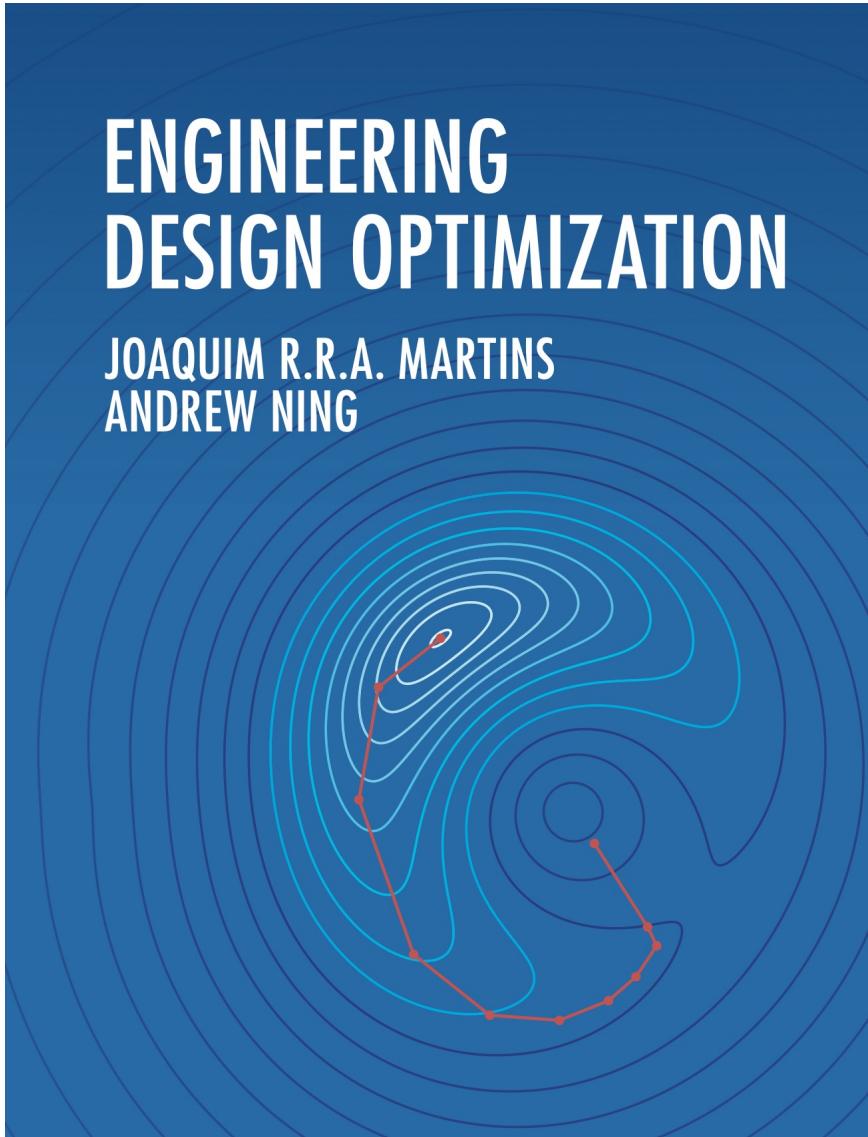
Justin S. Gray¹ · John T. Hwang² · Joaquim R. R. A. Martins³ · Kenneth T. Moore⁴ · Bret A. Naylor⁴

<https://link.springer.com/article/10.1007/s00158-019-02211-z>

OpenMDAO
ドキュメント

OpenMDAO 論文

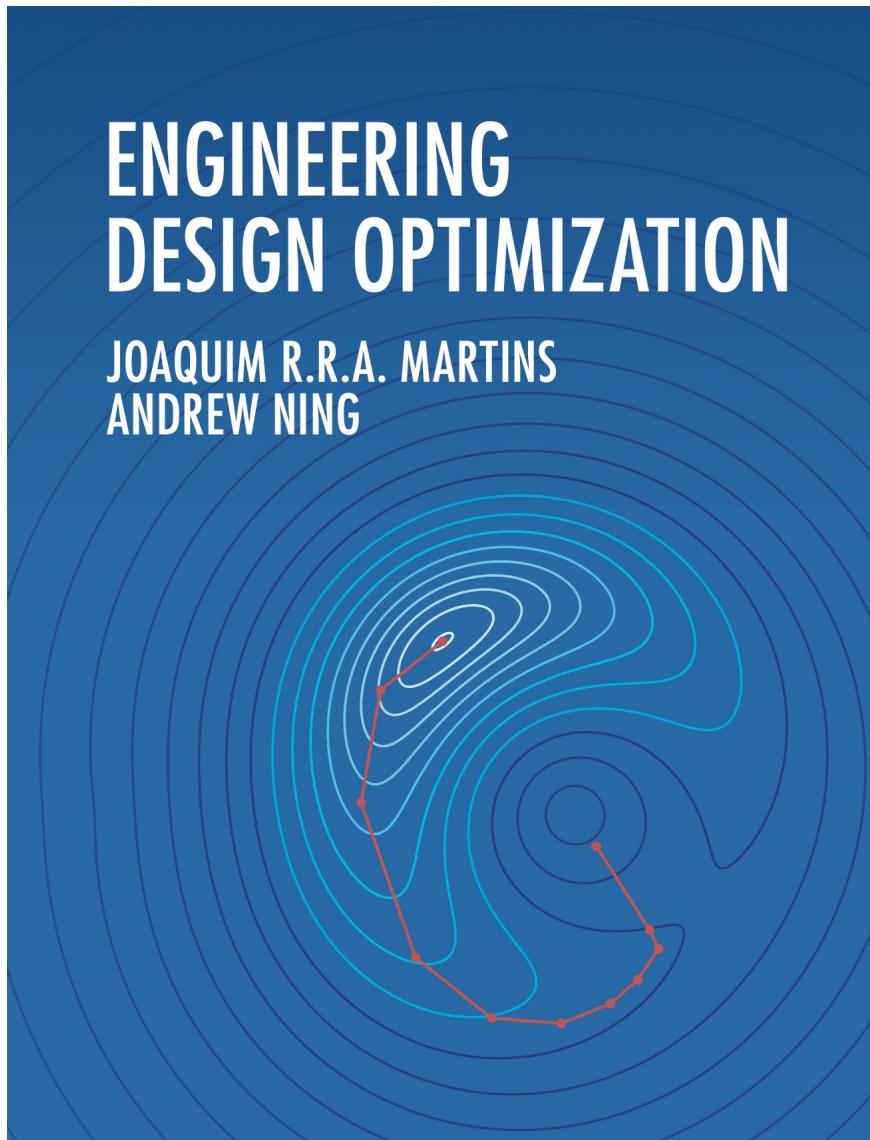
参考リソース (理論)



<https://mdobook.github.io>

設計最適化の教科書
PDF版は無料でダウンロード可能

参考リソース (理論)



OpenMDAOに関する項目

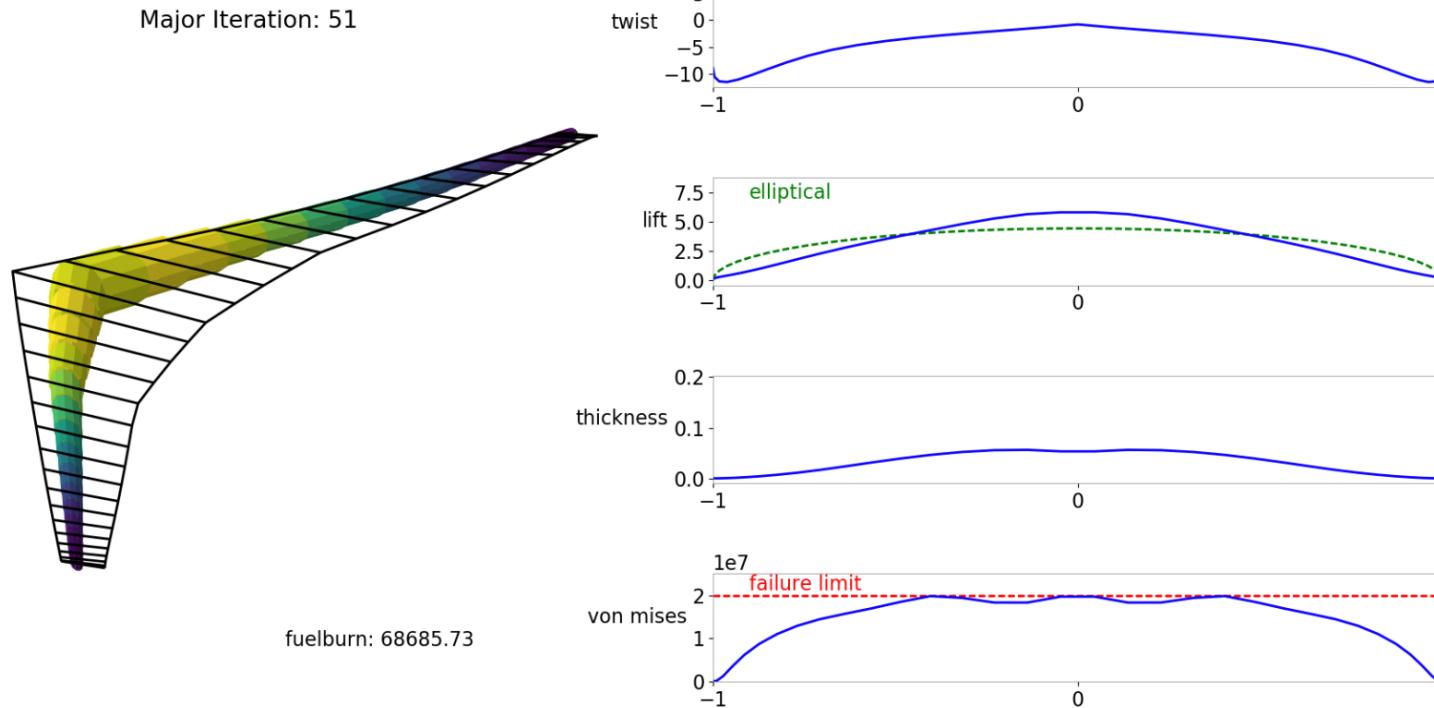
- Chapter 1 header and Sec. 1.1: Idea of optimization, motivation
- Sec. 1.2: Problem formulation
- Sec. 1.4.1-1.4.3: Gradient-based vs gradient-free optimization algorithms (see Fig. 1.23 for cost comparison), local vs global search, mathematical vs heuristic approaches
- Sec. 1.5: How to select an optimization algorithm
- Sec. 3.3: Introduction to governing equations residuals and function (explicit and implicit)
- Sec. 3.6: Overview of governing equation solvers
- Chapter 4 headerSec. 4.1-4.2: Basic idea of gradient-based optimization
- Example 4.18 and Sec. 4.6: Comparison of gradient-based algorithms, summary
- Chapter 5 header and Sec. 5.1: Basic idea of constrained optimization
- Sec 5.8: Summary of constrained optimization
- Chapter 6: Derivatives. Whole chapter recommended. Particularly important Sec. 6.7 (implicit analytic methods including adjoint), Sec. 6.8 (sparse Jacobians), Sec. 6.9 (UDE), and Sec. 6.10 (summary)
- Sec. 7.1: When to use gradient-free algorithms
- Chapter 13 header and 13.1: Introduction to MDO
- Sec. 13.2: Coupled models and solvers (includes MAUD in Sec. 13.2.6)
- Sec. 13.3.3: Implicit analytic coupled derivatives
- Tip 13.4: OpenMDAO
- Sec. 13.6: MDO summary

本日の流れ

1. OpenMDAOの概要と理論
2. OpenMDAOを用いたオープンソースライブラリの紹介
3. 実践編

OpenAeroStruct：翼の空力・構造連成解析と最適化

- Vortex lattice method (VLM) + 1D beam FEM



<https://github.com/mdolab/OpenAeroStruct>

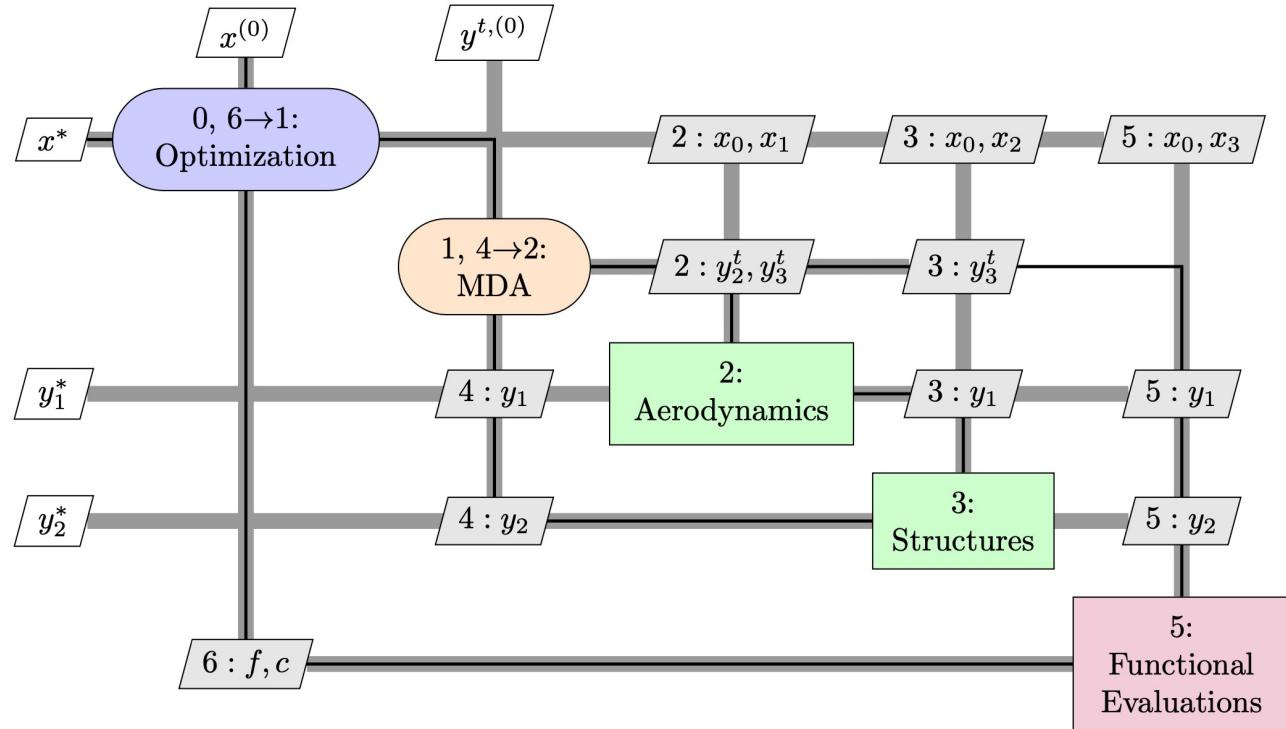


AEROSPACE
ENGINEERING
UNIVERSITY OF MICHIGAN



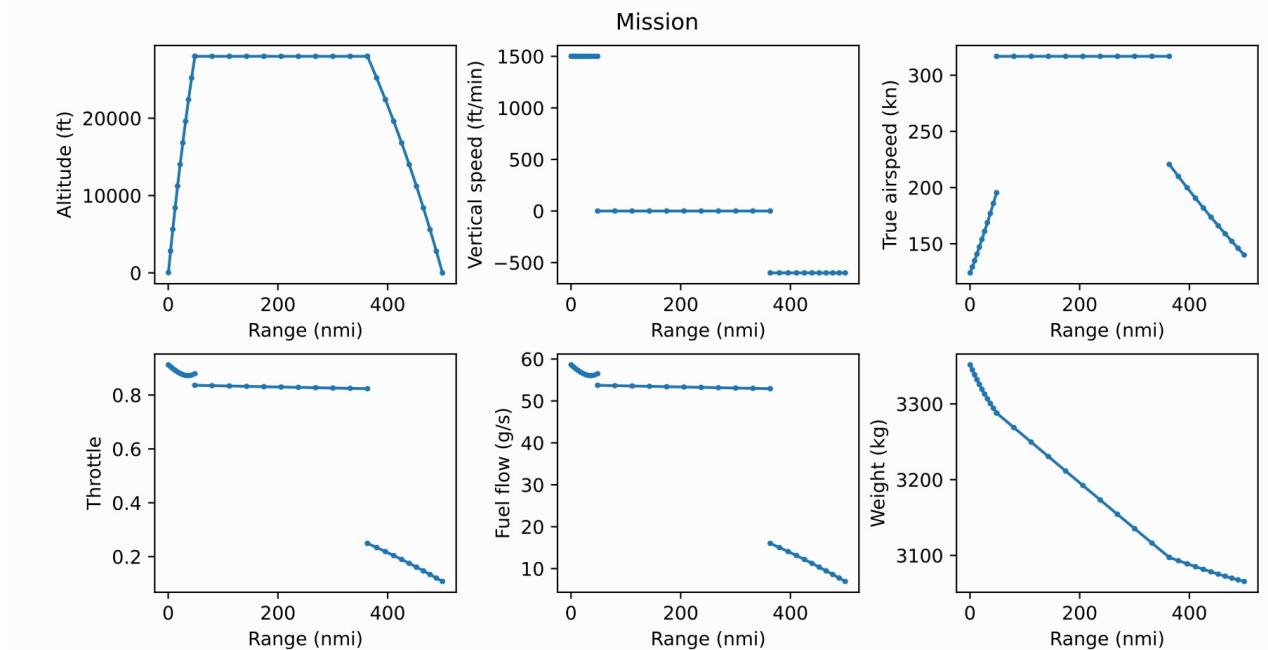
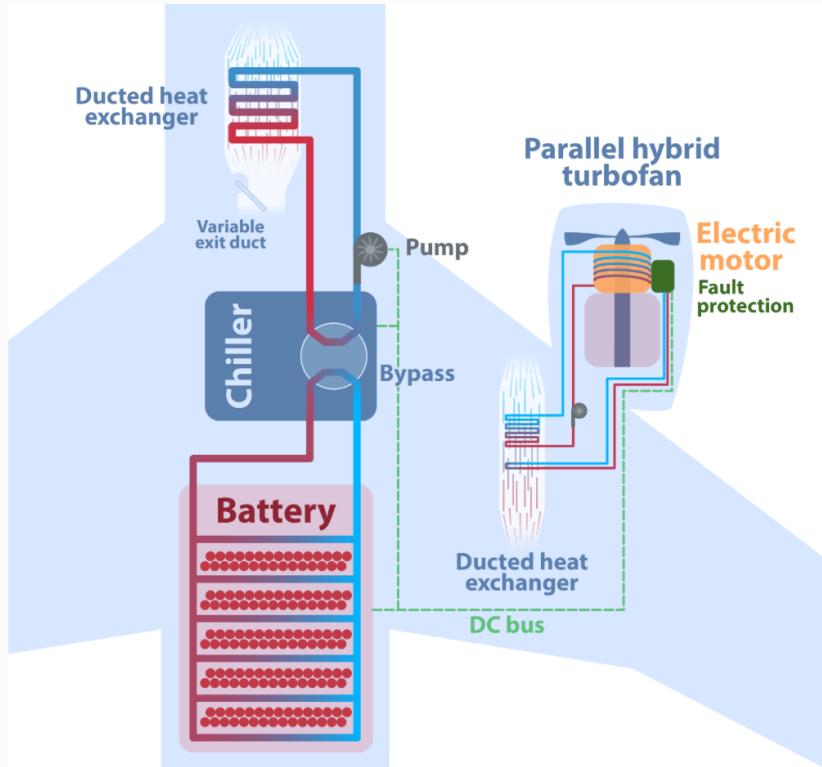
OpenAeroStructの実装

- 各コンポーネントの関数・偏微分を実装する
- コンポーネントを組み立てる
- 残りの連成微分(coupled derivative) 計算は自動



prob_vars	
indep_vars	
thickness_bsp	thickness
twist_bsp	twist
mesh	mesh
wing	radius
struct_setup	tube
nodes	nodes
assembly	K
wing	create_rhs forces
coupled	fem disp_aug
struct_states	disp disp
wing	def_mesh def_mesh
aero_states	aero_geom
assembly	AIC
rhs	rhs
root	circulations circulations
wing_loads	forces wing_sec_forces
wing_perf	loads
aero_funcs	liftcoeff CI
viscousdrag	CDv
liftdrag	L
coeffs	D
CD	CL1
CL	CDi
thicknessconstraint	CD
structural_weight	CL
vonmises	CL
failure	failure
total_perf	thickness_intersects
struct_funcs	structural_weight
cg_location	cg_location
vonmises	vonmises
failure	failure
CL_CD	CL
fuelburn	CD
L_equals_W	fuelburn
CG	weighted_obj
moment	L_equals_W
	total_weight
	cg
	CM

OpenConcept：航空機ミッション解析、電動航空機設計



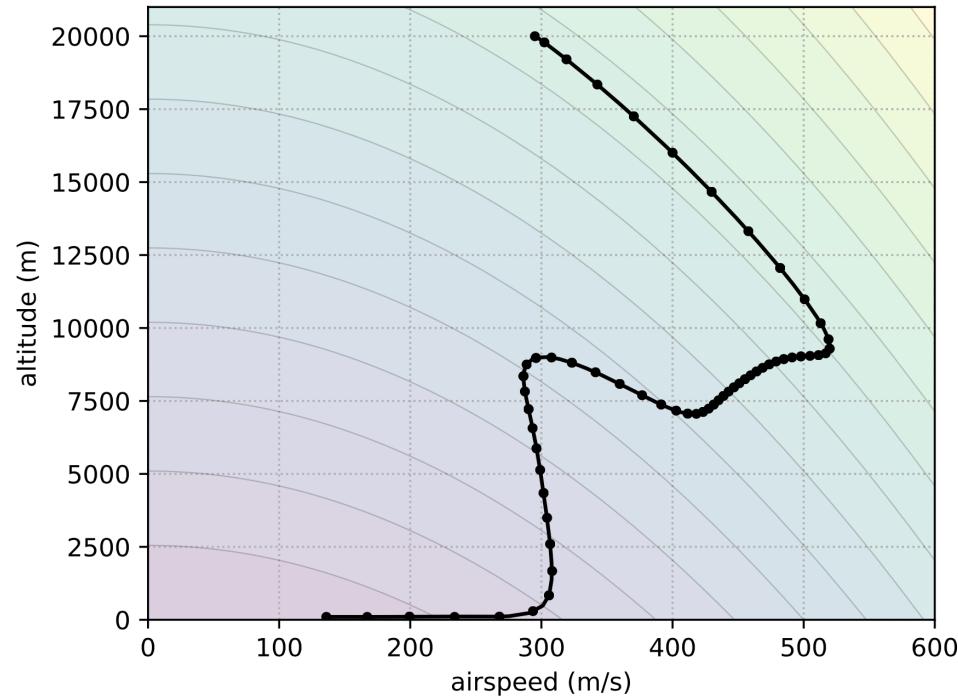
<https://github.com/mdolab/openconcept>



AEROSPACE
ENGINEERING
UNIVERSITY OF MICHIGAN

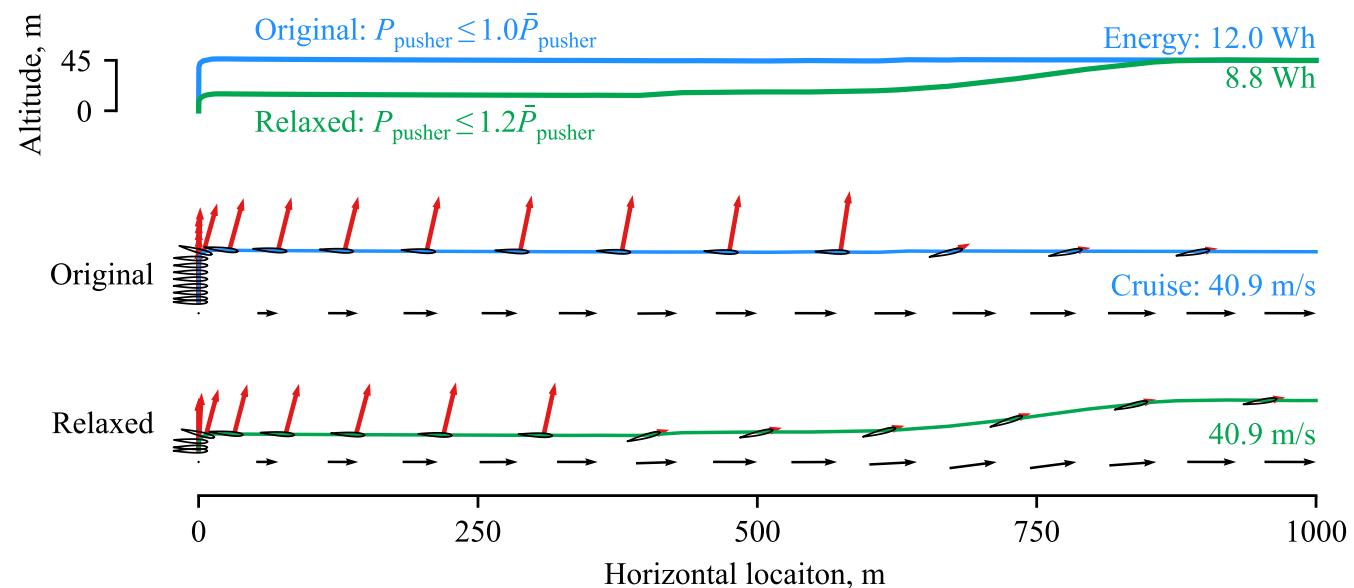
MDO
Lab

Dymos : 軌道最適化ライブラリ



例1. 超音速機の最短時間上昇

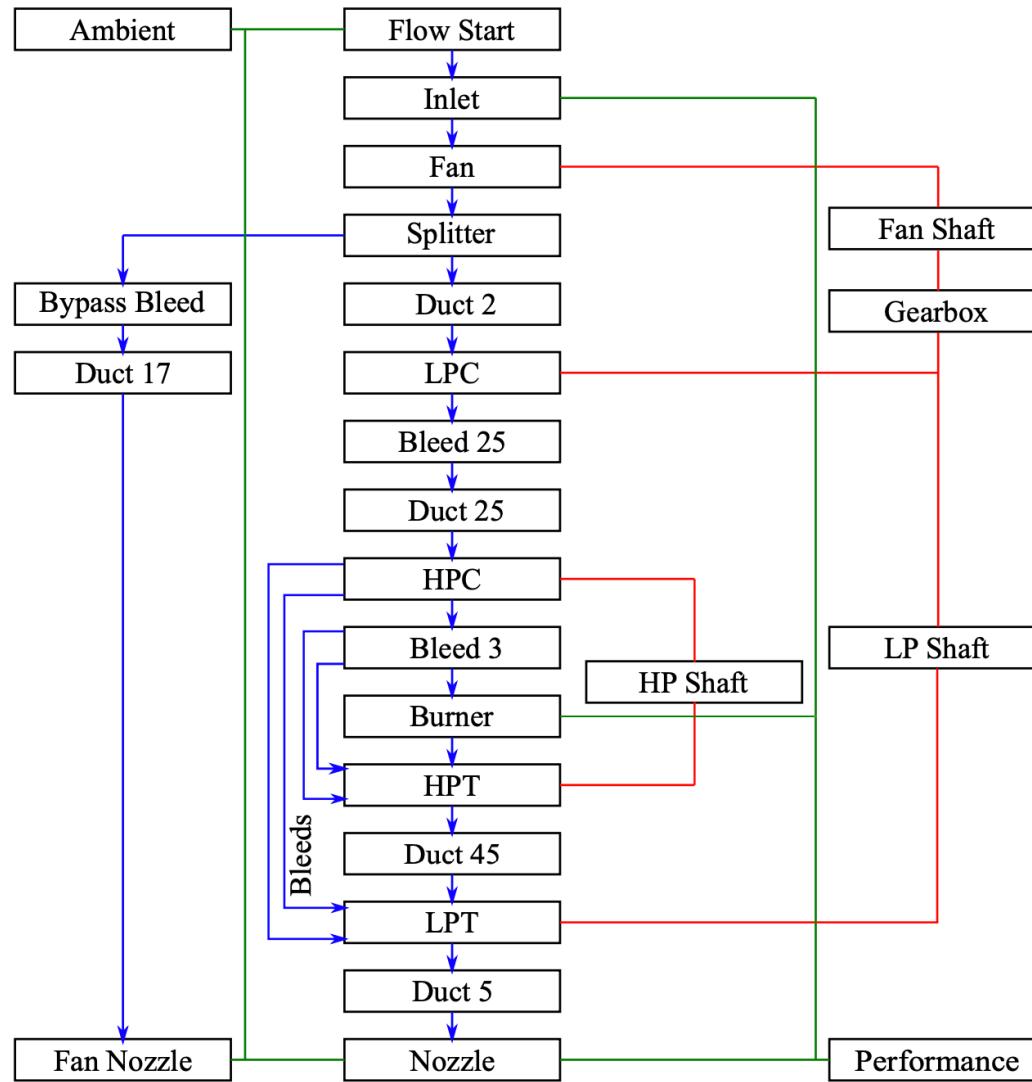
Falck et al., AIAA 2019-0976



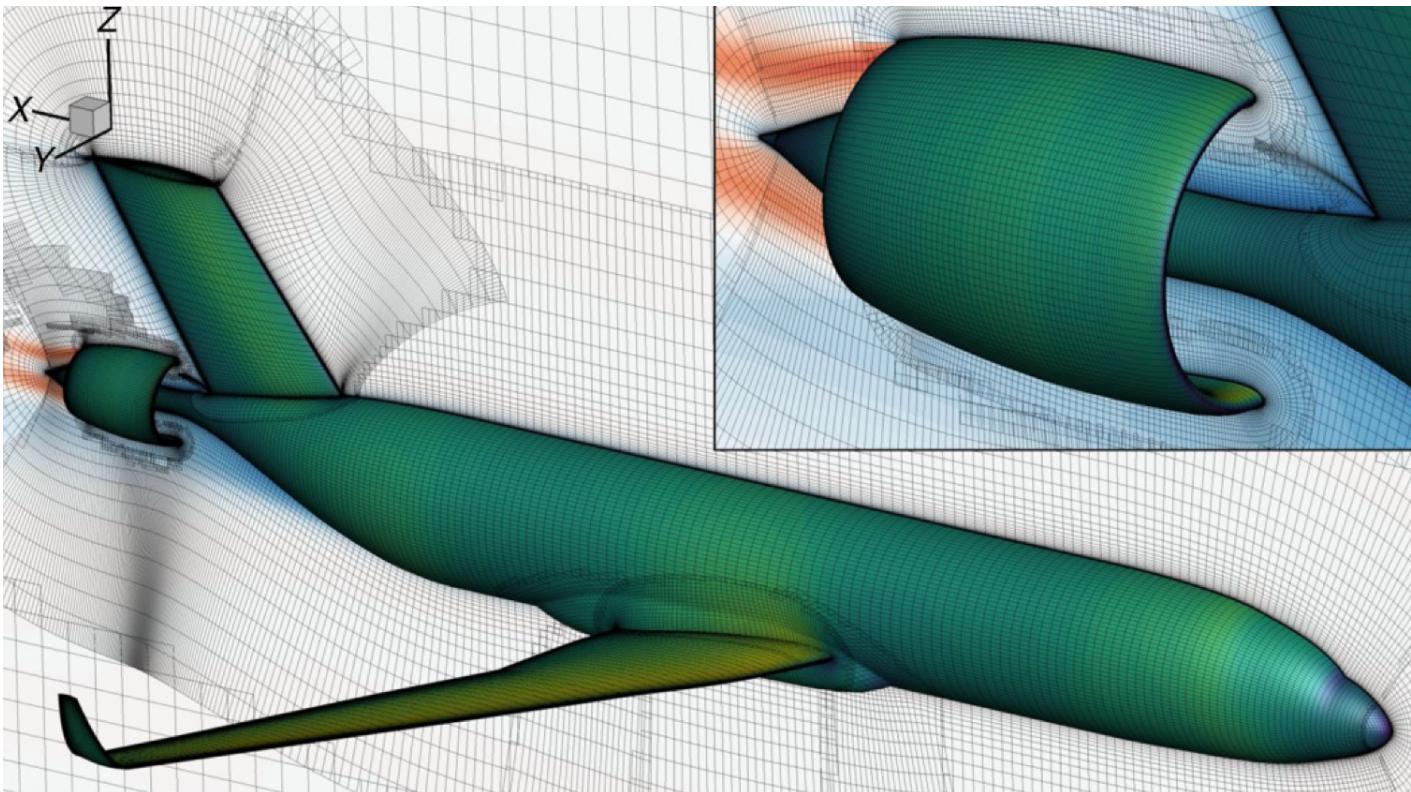
例2. UAVの概念設計・離陸軌道の同時最適化

Kaneko et al., 10th Autonomous VTOL Technical Meeting, 2023

pyCycle : 熱力学サイクルモデリング



MPhys : High-fidelity multiphysics



例. 境界層吸い込み(BLI)航空機の最適設計

Yildirim et al., *Journal of Aircraft*, 2022, 59, 4

<https://github.com/OpenMDAO/mphys>



AEROSPACE
ENGINEERING
UNIVERSITY OF MICHIGAN



本日の流れ

1. OpenMDAOの概要と理論
2. OpenMDAOを用いたオープンソースライブラリの紹介
3. 実践編
 1. Paraboloid 問題
 2. Sellar 問題

インストール

- Python 3 のインストール

- OpenMDAOのインストール

```
$ pip install openmdao
```

- サンプルコードのダウンロード

```
$ git clone git@github.com:kanekosh/OpenMDAO_workshop_UTokyo.git
```

例 1 : Paraboloid 問題

minimize: $f(x, y) = (x - 3)^2 + xy + (y + 4)^2 - 3$

by varying: x, y

subject to: $0 \leq x + y \leq 10$

出典 : OpenMDAO ドキュメント

https://openmdao.org/newdocs/versions/latest/basic_user_guide/single_disciplinary_optimization/first_analysis.html

例 1: $f(x, y)$ の実装

サンプルファイル: `paraboloid_1.py`

実行: `$ python paraboloid_1.py`

例 1: $f(x, y)$ の微分の実装

テンプレファイル: `paraboloid_2.py`

【タスク】 `compute_partials()` メソッド内で、偏微分を実装

回答ファイル: `paraboloid_2_sol.py`

実行(偏微分値のverification): \$ `python paraboloid_2.py`

例 1：最適化の実行

minimize: $f(x, y) = (x - 3)^2 + xy + (y + 4)^2 - 3$

by varying: x, y

subject to: $0 \leq x + y \leq 10$

サンプルファイル: `paraboloid_optimize.py`

実行: \$ `python paraboloid_optimize.py`

例 2: Sellar 問題 (シンプルな連成問題)

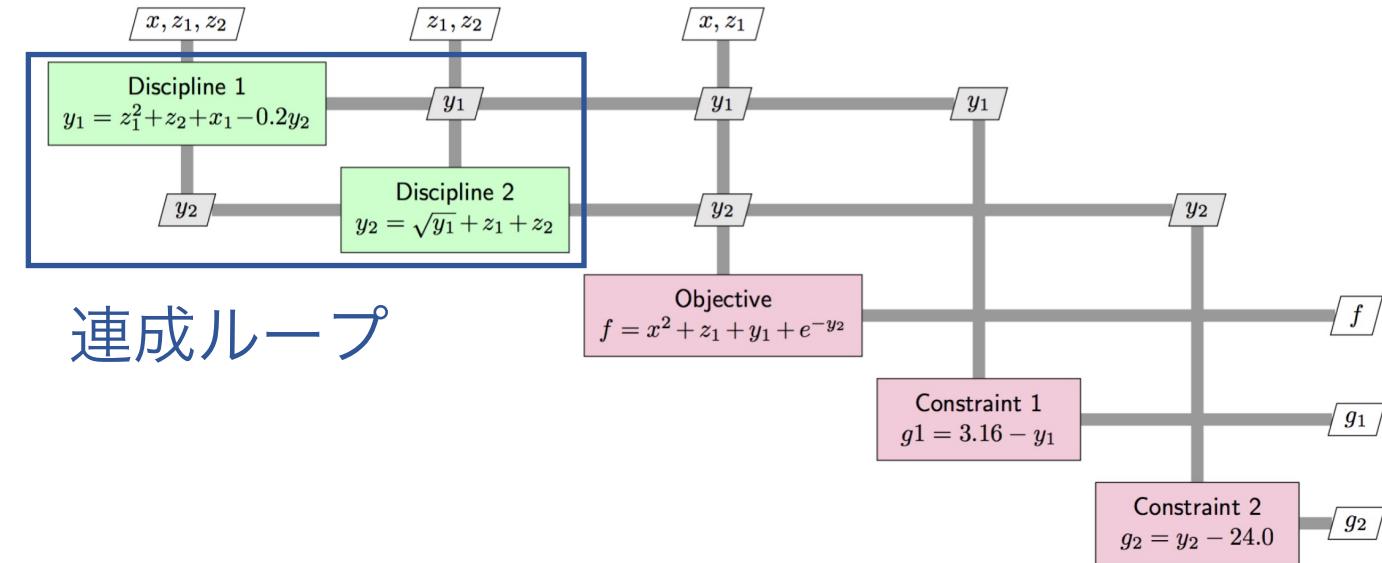
minimize: $f(x, y, z) = x^2 + z_1 + y_1 + e^{-y_2}$

by varying: x, z_1, z_2

subject to: $3.16 - y_1 \leq 0$

$y_2 - 24 \leq 0$

while solving: $y_1 = z_1^2 + z_2 + x - 0.2y_2$
 $y_2 = \sqrt{y_1} + z_1 + z_2$



出典 : OpenMDAO ドキュメント

https://openmdao.org/newdocs/versions/latest/basic_user_guide/multidisciplinary_optimization/sellar.html

例 2: Discipline 1 の実装

テンプレファイル: `sellar_components.py`

【タスク】 `setup()`メソッド内に、`input/output`を定義する

【タスク】 `compute()`メソッドに関数計算を実装する

回答ファイル: `sellar_components_sol.py`

$$\text{minimize: } f(x, y, z) = x^2 + z_1 + y_1 + e^{-y_2}$$

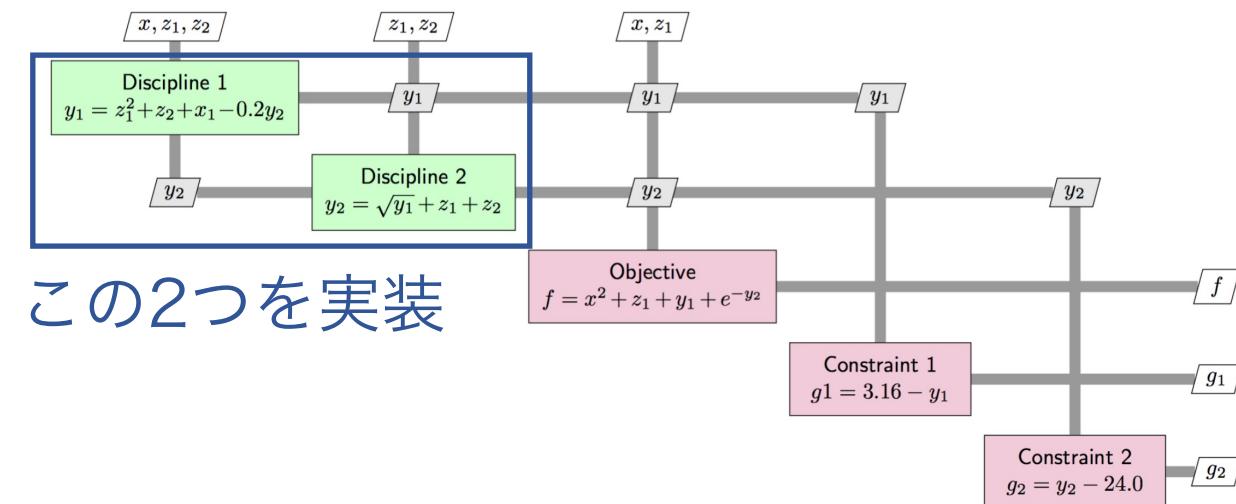
by varying: x, z_1, z_2

subject to: $3.16 - y_1 \leq 0$

$$y_2 - 24 \leq 0$$

$$\text{while solving: } y_1 = z_1^2 + z_2 + x - 0.2y_2$$

$$y_2 = \sqrt{y_1} + z_1 + z_2$$



例2: 連成モデルの構築・非線形ソルバの実装

サンプルファイル: `sellar_group.py`

モデルの可視化: `om.n2(prob)`

minimize: $f(x, y, z) = x^2 + z_1 + y_1 + e^{-y_2}$

by varying: x, z_1, z_2

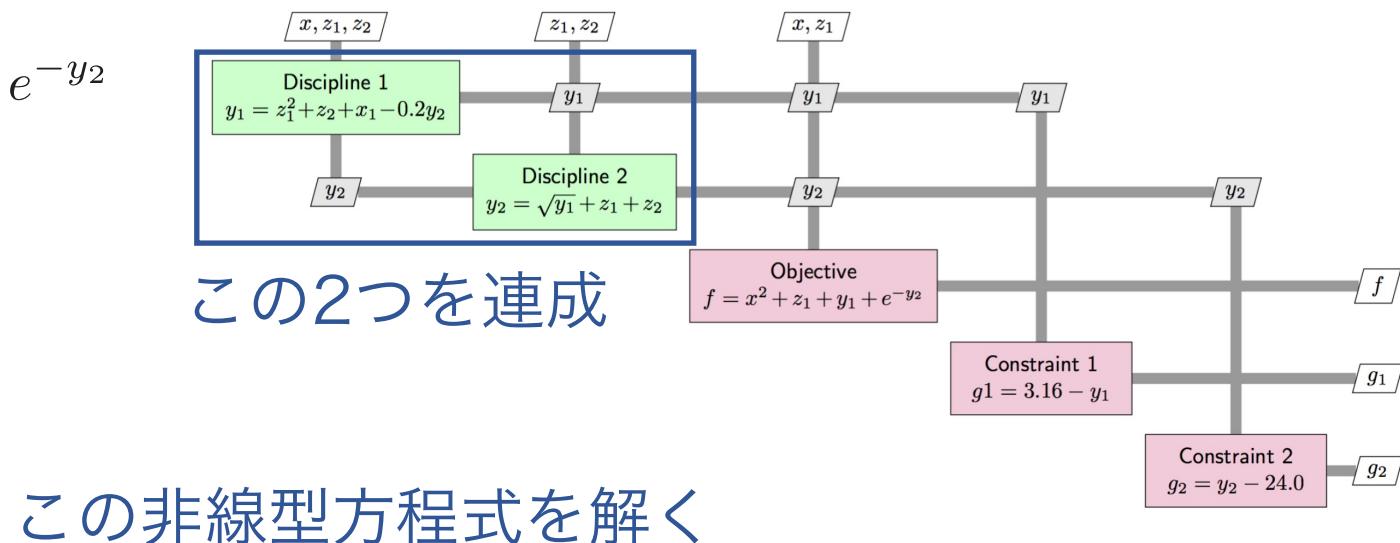
subject to: $3.16 - y_1 \leq 0$

$$y_2 - 24 \leq 0$$

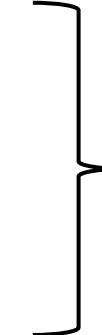
while solving:

$$y_1 = z_1^2 + z_2 + x - 0.2y_2$$

$$y_2 = \sqrt{y_1} + z_1 + z_2$$



OpenMDAOで使用可能な非線形ソルバ

- Nonlinear Block Gauss-Seidel
 - Nonlinear Block Jacobi
 - Broyden (準Newton)
 - Newton
 - 勾配が必要。
 - 最も収束が早い(ことが多い)
- 
- 勾配は不要

例 2：最適化の実行

テンプレファイル： sellar_optimize.py

【タスク】 設計変数、目的関数、制約を定義する

回答ファイル： sellar_optimize_sol.py

```
Optimization terminated successfully      (Exit mode 0)
Current function value: [3.18339395]
Iterations: 11
Function evaluations: 12
Gradient evaluations: 11
Optimization Complete
-----
minimum found at
x = 4.401421628747386e-15
z = [1.97763888e+00 1.96644286e-14]
minimum objective
f = 3.1833939532752136
```

例2：[おまけ] 連成モデルを `promotes` を使って構築する

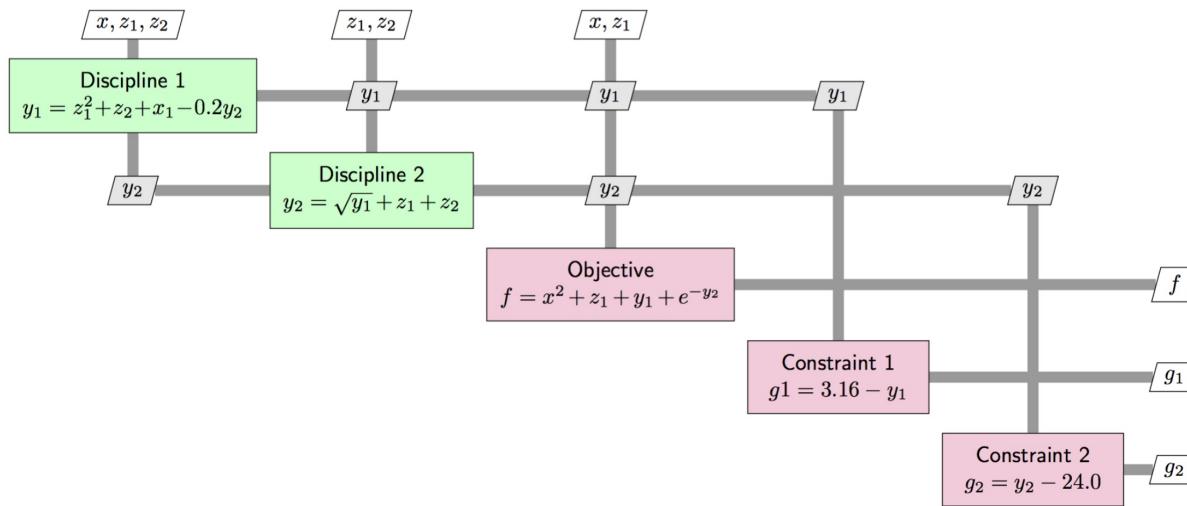
テンプレファイル： `sellar_group_via_promotes.py`

【タスク】 `connect` の代わりに、`promotes` で変数 `y1`, `y2` をリンク

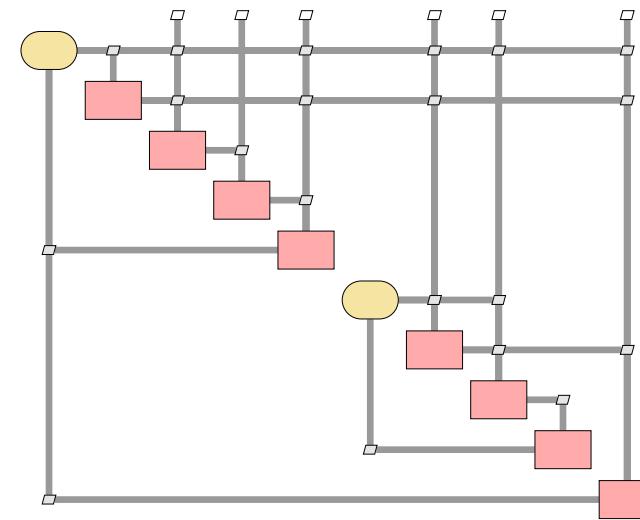
回答ファイル： `sellar_group_via_promotes_sol.py`

まとめ

- シンプルな問題なら、自分でスクラッチから実装した方が早い
- 複雑な問題ほど、OpenMDAOが真価を発揮する
 - コンポーネント数が多い場合
 - 連成(サイクル)がある場合
 - コンポーネントベースの実行 → モデル組替が簡単



Sellar問題の勾配を、Chain ruleで導出するのはやや複雑



複雑な問題構造の場合、OpenMDAOは強力

参考リソース（実践）



OpenMDAO

Search this book...

Include Source Docs

Welcome to OpenMDAO

Welcome to OpenMDAO

OpenMDAO is an open-source high-performance computing platform for systems analysis and multidisciplinary optimization, written in Python. It enables you to decompose your models, making them easier to build and maintain, while still solving them in a tightly coupled manner with efficient parallel numerical methods.

The OpenMDAO project is primarily focused on supporting gradient-based optimization with analytic derivatives to allow you to explore large design spaces with hundreds or thousands of design variables, but the framework also has a number of parallel computing features that can work with gradient-free optimization, mixed-integer nonlinear programming, and traditional design space exploration.

<https://openmdao.org/newdocs/versions/latest/main.html>

The screenshot shows the YouTube channel page for 'OpenMDAO'. At the top, there's a header with the channel logo, name (@OpenMDAO), and subscriber count (618). Below the header is a navigation bar with links for HOME, VIDEOS (which is underlined), PLAYLISTS, COMMUNITY, CHANNELS, ABOUT, and a search bar. A 'Subscribe' button is located on the right side of the header. The main content area displays four video thumbnails under the heading 'Recently uploaded': 1) 'Explicit vs implicit systems' (8:06), 2) 'Driver Scaling Report' (19:33), 3) 'Types of gradient-free optimizers' (10:38), and 4) 'Why optimization convergence matters' (11:30). Below each thumbnail, there's a brief description and the upload date ('59 views • 6 days ago' for the first video).

<https://www.youtube.com/@OpenMDAO>

OpenMDAO ドキュメント

OpenMDAO Youtubeチャンネル