FIXSTARS Amplify

量子コンピュータ時代の最適化セミナー

～ブラックボックス最適化を活用した
撹拌機器の設計・運転条件最適化～

本日の予定

第一部

- 会社紹介
- Fixstars Amplify の紹介
- 組合せ最適化事例
- ワークショップ事前準備

第二部

- 組合せ最適化の基本
  - 数の分割ハンズオン

第三部

- ブラックボックス最適化とは
- FMQAの概要とフロー
- FMQAによる設計最適化ハンズオン
  - 問題の説明
  - Amplify-BBOpt 実装
- まとめ

質問は随時 Zoom の Q&A へお願いします

株式会社 Fixstars Amplify (“Amplify”)

- **最適化のための量子コンピューティング基盤の提供**　**（無料・専門知識不要で利用スタートOK！）**
  - 開発環境：**Amplify SDK**（Python基礎知識のみで開発スタート）
  - 実行環境：**Amplify AE** や他社商用ソルバー（後述）

1,100+ ユーザー所属組織数（企業、研究教育機関）

1億3,000万+ ユーザー累計実行回数

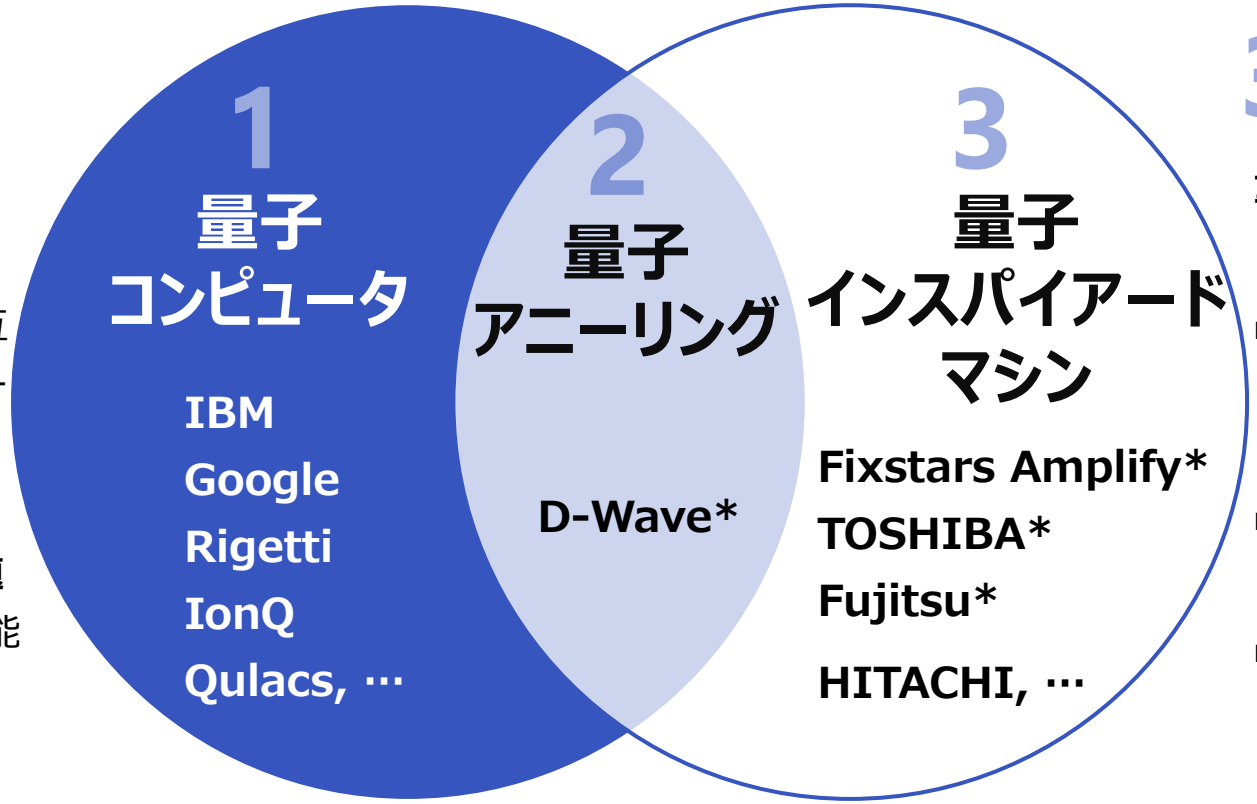
- 2021年10月に設立
- 親会社：株式会社フィックスターズ
  - ソフトウェア高速化のプロフェッショナル集団
  - 日本で初めて D-Wave Systems 社と提携 (2017年)

FIXSTARS Amplify

量子・量子インスパイアード技術 （広く Fixstars Amplify でカバー）

1 量子コンピュータ
（量子ゲート方式）

- 古典汎用コンピュータの上位互換。量子ゲートを操作。エラー訂正機能の無いNISQ型実機がクラウド利用可能
- QAOAにより**組合せ最適化問題** (**QUBO**) を取り扱うことが可能
- 演算規模：**～数100ビット**



3 量子インスパイアードマシン
（半導体技術に基づくイジングマシン）

- 二次の多変数多項式で表される目的関数の**組合せ最適化問題** (**QUBO**) 専用マシン
- 統計物理学におけるイジング模型に由来。様々な実装により実現。
- 演算規模：
  260,000+ビット（*Amplify AE*）

2 量子アニーリング（量子焼きなまし法式のイジングマシン）

- イジングマシンの一種。量子イジング模型を物理的に搭載したプロセッサで実現。
  量子効果を物理的に調整し、自然計算により低エネルギー状態が出力
- **組合せ最適化問題** (**QUBO**) を扱う専用マシン
- 演算規模：**～数1,000ビット**

FIXSTARS Amplify

最適化問題とは

- 最適化
  - さまざまな制約の下で、数ある選択肢の中から、ある観点で最適な選択を決定
    - 製品の高性能化、製造過程の効率化、コスト削減、歩留まり向上
    - 商品の発注計画、効率的な物流ルート、金融資産の運用
    - 災害復旧スケジュール、公共施設の配置、エネルギー需給バランス
- 数理最適化
  - 対象となる問題を数式で記述し、数理的な計算手法で最善策を求める
    - 問題の数式表現：　　数理モデル
    - 数理モデルの構築：　　定式化
- 数理モデルの構成要素
  - 目的関数：　　目的達成度を表す数式 (最小あるいは最大化)
  - 決定変数：　　選択肢となる変数
  - 制約条件：　　決定変数間の取り得る条件式 (制約関数)

} 制約条件を満たす決定変数は実行可能解と呼ばれる

最適化問題の分類

数理最適化問題

- 連続最適化問題
  - 決定変数が連続値
    （従来手法でも求解可能）
- 決定変数が離散値 (整数、0-1整数など)
  - 目的関数の次数
    - 線形 → 整数線形計画問題
      （従来手法で求解可能）
    - **2次多項式**
    - **高次多項式**

} **整数非線形計画問題**
（最も解くのが難しい）
（**QUBO**に基づいて求解）

量子・量子インスパイアードが得意な問題

Quadratic	二次形
Unconstrained	制約条件なし
Binary	0-1整数 (二値)
Optimization	計画（最適化）

最適化：線形/非線形な目的関数を最小化するような**連続/離散な決定変数**の値を求める

クラウドサービス：**Fixstars Amplify**

Fixstars Amplify とは

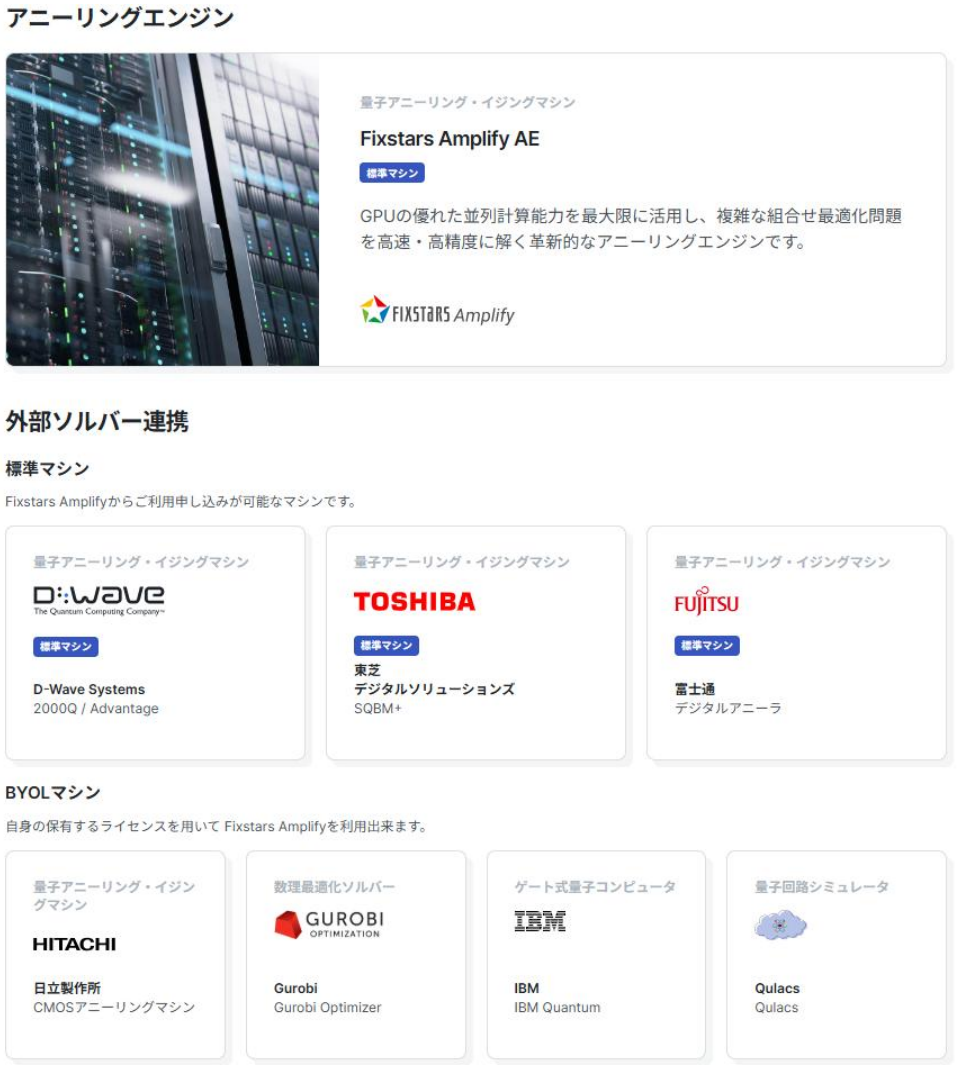
いつでも　開発環境 と 実行環境 がセット
すぐにアプリ開発と実行が出来る

誰でも　ハードウェアや専門的な知識が不要
無料で開発がスタート可能
多くの解説、サンプルコード

高速に　26万ビットクラスの大規模問題の
高速処理と高速実行が可能

あらゆる　一般に公開されている全てのイジング
マシンを利用可能

Fixstars Amplify の対応マシンの一例



標準マシン は、

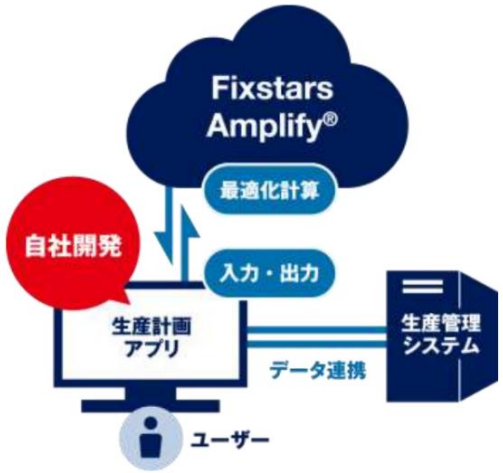
- ベンダ各社と個別マシン利用契約なし、
- 評価・検証用ベーシックプランなら無料、

で利用可能！←「いつでも」、「誰でも」

今後も幅広い対応マシンの追加が続々と行われる予定です！←「あらゆる」

活用領域とユースケース（PoC・実稼働）

生産計画	• 多品種少量生産、保全計画、設備投資、在庫
従業員割り当て	• 食品、輸送、製造
エネマネ	• エネルギーミックス、装置の運転制御
経路	• 配送、船舶、無人搬送車 (AVG)
メディア	• 最適広告配信
研究開発、設計	• 材料設計 • 物理シミュレーション • **ブラックボックス最適化**



Amplify インタビュー　検索

アニーリングマシンの
プログラミング体験

イジングマシンの実行手順

ハンズオンセミナーのメイントピック

1. 数理モデル検討
   解きたい課題の「目的関数」「決定変数」とその「制約条件」を検討する
2. 定式化
   「 多項式」で「目的関数」と「決定変数」を記述 (変換) する
   「決定変数」 に対する「制約条件」を Amplify で表現する
3. モデル変換
   (論理・物理)
   各マシンの仕様や制限に準拠した形式にモデルを変換する
   (例: 二次項に制約がある場合は「グラフマイナー埋め込み」問題を解く)
4. 入力データの準備
   各マシンのSDKやAPI仕様に合わせてQUBO模型 (物理) をデータ化する
5. マシンの実行
   マシンを実行して出力の変数値やエネルギー値(コスト値)を解析する
   上記の逆の手順を**辿り解きたい課題**の「決定変数」を解釈する

Amplify SDK によるサポート

Amplifyの基本的な使用方法（1）

- まずはインポート

```
# Install Amplify SDK to Google Colab
! pip install -q amplify

#Import all functions and classes
from amplify import *
```

- 使用するマシンを選択

```
# Fixstars Amplify AE
client = AmplifyAEClient ()

# Timeout 1s
client.parameters.time_limit_ms = 1000 #ms

# API token
client.token = "AE/XXXXXXXXXXXXXXXXXXXx"
```

その他のクライアントを使用する場合は
ドキュメントを参照
https://amplify.fixstars.com/ja/docs/amplify/v1/clients.html

Amplifyの基本的な使用方法 (2)

- 目的関数の定式化 (多項式)
  - バイナリ多項式の構築

```
# 決定変数生成器を作成
g = VariableGenerator()

# 長さ 2 の決定変数配列を作成
q = g.array("Binary", 2)
```

バイナリ変数 (**“Binary”**) だけでなく、イジング変数 (**“Ising”**) や、整数変数 (**“Integer”**)、実数変数 (**“Real”**) も指定可能

  - 式の構築

```
f = 2 * q[0] * q[1] + q[0] - q[1] + 1
print(f)
# 2 q_0 q_1 + q_0 - q_1 + 1
```

より高次な多項式もOK。マシンが対応していない場合は Amplify SDK が内部的に次数下げを行う

Amplifyの基本的な使用方法 (3)

- モデルの作成とマシンの実行

```
model = Model(f)
result = solve(model, client)
```

目的関数と制約条件からモデルを作成し
使用するクライアントと共に求解

- 結果の取得

```
print(f"objective = {result.best.objective}")
# objective = 0.0

print(f"q = {q.evaluate(result.best.values)}")
# q = [0. 1.]
```

最良解を `best` で指定
目的関数の値を `objective` にて
変数の値を `values` で得る

Amplify SDK によるプログラミング例

```
from amplify import *

# 決定変数を生成
g = VariableGenerator()
q = g.array("Binary", 2)

#目的関数を構築
f = 2 * q[0] * q[1] + q[0] - q[1] + 1

# Amplify モデルを構築
model = Model(f)

#ソルバーの設定
client = AmplifyAEClient()
client.parameters.time_limit_ms = 1000 #ms
client.token = "AE/XXXXXXXXXXXXXXXXXXXx"

# 求解の実行
result = solve(model, client)

# 結果の表示
print(f“objective = {result.best.objective}”)
print(f"q = {q.evaluate(result.best.values)}")
```

1. 定式化
   - 決定変数：スカラーあるいは配列型
   - 目的関数：決定変数による数式処理
   - 制約条件：制約条件の構築及び管理
2. ソルバークライアントの選択
   - ソルバークライアントオブジェクトの構築
   - ほぼ全てのパラメータの設定が可能
3. ソルバーを実行
   - 論理モデルをハードウェアのスペック等に合わせたモデルに変換
   - 適切なモデル変換・定式化手法を選択
4. 解の取得
   - マシンの出力解を逆変換し決定変数の形式で出力

ワークショップ
事前準備（事前メールの内容）

ワークショップの事前準備 (1)

- 【事前メールに記載】ご自身のPC (ブラウザ上) で Python プログラミングを行います。Google Colaboratory を使うので、事前にログイン出来ることを確認をお願いします（要 Google アカウント）

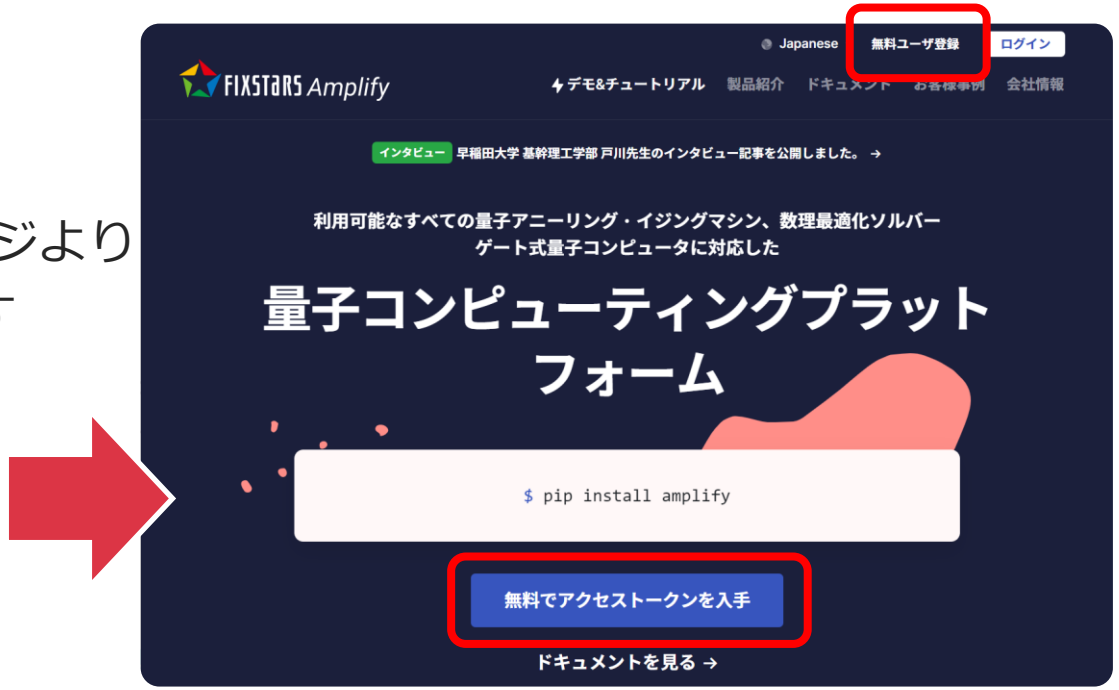
Google Colab 検索

https://colab.research.google.com/

- 【事前メールに記載】 Fixstars Amplify ホームページよりユーザ登録の上、無料トークンの取得をお願いします（1分で終わります）

Fixstars Amplify 検索

https://amplify.fixstars.com/



ワークショップの事前準備 (2)

【事前メールに記載】

- 取得されたトークンを用いて、トークンチェック用サンプルコードが動くか確認をお願いします。

  https://colab.research.google.com/drive/1-Jh2nlhWO97OO96WgBwCSOQmP8BVa6-T （※URLはZoomのチャット欄を参照）

- サンプルコードは閲覧のみ可能な状態です。**「ファイル」→「ドライブにコピーを保存」**の上、ご自身のトークンを入力してください。その後、Shift + Enterで実行下さい。

```
! pip install amplify
token = "AE/******************************" # ご自身のトークンを入力
```

- ご自身のトークン番号は、Amplifyウェブページよりご確認いただけます。

- 実行後、以下の結果が出力されればOKです。

```
result: [q_0, q_1] = [1. 1.] (f = 0.0)
```

ハンズオン①

- トークンチェックサンプルコードで遊んでみましょう。
  - 決定変数配列サイズを変更

```
# バイナリー変数q_0, q_1を定義
q = VariableGenerator().array("Binary", 2)
```

例

```
# バイナリー変数q_0, q_1, q_2, q_3を定義
q = VariableGenerator().array("Binary", 4)
```

  - 目的関数の多項式を変更

```
# 目的関数 1 - q_0 * q_1 を定義
f = 1 - q[0] * q[1]
```

例

```
# 目的関数 1 - q_0 * q_1 *q_2 * q_3 + q_0 * q_2 - 2 * q_3 + 3 * q_2 - q_0 を定義
f = 1 - q[0] * q[1] *q[2] * q[3] + q[0] * q[2] - 2 * q[3] + 3 * q[2] - q[0]
```

FIXSTARS Amplify

「数の分割問題」のハンズオン

通常の組合せ最適化
（ブラックボックス最適化への導入）

数の分割問題

- 与えられた n 個の整数 $a_0, \cdots, a_{n-1}$ を二つの集合に分ける。集合内の数の和が、もう一方の集合内の数の和と等しくなるようできるか?

NP完全問題：とても難しい問題として知られている → 全通り試すしか方法は無い

数の分割問題（具体例と解法の方針）

具体例

`{2,10,3,8,5,7,9,5,3,2}` の10個の数の完璧な分割は見つけられるか？

答　え

- 存在する
  - `{2,3,5,7,10}` と `{2,3,5,8,9}`
  - どちらも和は 27
- 分割方法は 23 通り存在する (対称を除く)

どうやって解くか？

・ひとつの『数』がどちらの集合に分割されるか全通り試す → $2^{10} = 1024$通り
・効率のよい厳密な方法は知られていない・・・
(もし発見されたら大騒ぎ)

数の分割問題（定式化）

最適化問題：数の分割において最も惜しい組合せは何か？

- **目的関数**

  {集合1の和} – {集合2の和} の絶対値を最小化

- **決定変数**

  数 a_i がどちらの集合に属するかを s_i で表す

  - a_i = { 2,10, 3, 8, 5, 7, 9, 5, 3, 2}
  - s_i = {-1, 1,-1, 1,-1,-1, 1, 1, 1, 1}

数理モデル

- 目的関数 $$f = \left| \sum_{i=0}^{N-1} s_i a_i \right| \quad (s_i \in \{-1, +1\})$$

$\sum s_i a_i$ は、自然と
{『1』の集合の和} – {『-1』の集合の和}
となる！

数の分割問題（バイナリへの式変形）

0-1整数二次計画問題への変換

- Quadratic Unconstrained Binary Optimization (QUBO) 式

$$f = \left| \sum_{i=0}^{N-1} s_i a_i \right| \quad (s_i \in \{-1, +1\})$$

$\sum s_i a_i$ は、自然と
{『1』の集合の和} – {『-1』の集合の和}
となる！

$$\rightarrow \left(\sum_{i=0}^{N-1} s_i a_i \right)^2 \quad (s_i \in \{-1, +1\})$$

絶対値を二次式で表す

$$\rightarrow \left(\sum_{i=0}^{N-1} (2q_i - 1) a_i \right)^2 \quad (q_i \in \{0, +1\})$$

±1をバイナリ変数で表す

数の分割問題（定式化の具体例）

問　題

- a_i ={2,10,3,8,5,7,9,5,3,2} の10個の数の完璧な分割は見つけられるか？

決定変数

- $q_i = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\} (q_i \in \{0,1\})$ で集合0又は集合1、どちらに所属するかを表す

目的関数

$$f = \left(\sum_{i=1}^{N} (2q_i - 1)a_i \right)^2$$

目的関数を展開

$$f = \begin{pmatrix} 2(2q_0 - 1) + 10(2q_1 - 1) + 3(2q_2 - 1) + 8(2q_3 - 1) + 5(2q_4 - 1) \\ +7(2q_5 - 1) + 9(2q_6 - 1) + 5(2q_7 - 1) + 3(2q_8 - 1) + 2(2q_9 - 1) \end{pmatrix}^2$$

数の分割問題（プログラム）

1. 定式化

- 問題の定義と決定変数生成器による決定変数の生成

```
a = [2, 10, 3, 8, 5, 7, 9, 5, 3, 2]
q = amplify.VariableGenerator().array("Binary", len(a))
```

- 目的関数、$f = \left(\sum_{i=1}^{N}(2q_i - 1)a_i\right)^2$、の定式化（①②③は同等）

①

```
f = ((2 * q - 1) * a).sum() ** 2
```

②

```
f = 0
for i in range(len(a)):
    f += (2 * q[i] - 1) * a[i]
f **= 2
```

③

```
f = amplify.sum((2 * q - 1) * a) ** 2
```

色々な書き方が出来る

2. 実　行

```
result = amplify.solve(f, client)
```

3. 結　果

```
q = [1, 1, 1, 0, 1, 1, 0, 0, 0, 0], f = 0.0, w = 27
```

得られた目的関数の値0

各集合の合計値27

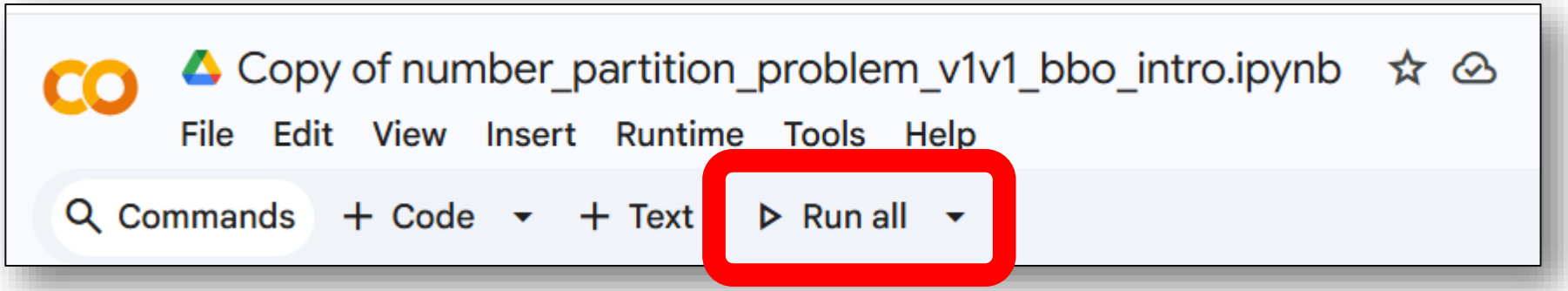
各数字に対して、集合0か、集合1か

FIXSTARS Amplify

ハンズオン②

- 数の分割ハンズオンコードを触ってみましょう。
  https://colab.research.google.com/drive/11vI2sHbtP_aaZHOTjsGYoGcvw2dHhD1z

  1. ドライブにコピーを保存
  2. APIトークンをコピペ
  3. サンプルコードを上から実行して、完璧な分割が得られるか確認
  4. 課題1～3を可能な限りチャレンジ
     （後日取り組み＆不明点はご質問でもOK）



オンラインデモ & チュートリアル

Amplify デモ 検索

https://amplify.fixstars.com/ja/demo

デモアプリケーション

ピクロスパズルの求解

プログラミング難易度 ★★★☆☆

複雑な定式化の例として、数字で与えられるヒントを元にマスを塗り、絵を完成させるパズルゲーム、ピクロスを解くアプリを開発します。

デモアプリ　サンプルコード

チュートリアル応用編

ブラックボックス最適化 (1)

プログラミング難易度 ★★★★☆

複雑で未知な目的関数にも適用可能な、機械学習と組み合わせ最適化を組み合わせたブラックボックス最適化手法を紹介し、Amplifyを用いて実装します。

サンプルコード

チュートリアル応用編

ブラックボックス最適化 (2)

プログラミング難易度 ★★★★☆

機械学習と量子アニーリング・イジングマシンを活用するブラックボックス最適化の適用例として、疑似的な高温超電導を実現する材料探索を取り扱います。

サンプルコード

チュートリアル応用編

ブラックボックス最適化 (3)

プログラミング難易度 ★★★★☆

化学プラントにおける生産量を最大化するための運転条件最適化を行います。最適化には、機械学習モデルに基づくブラックボックス最適化と化学反応に関する物理シミュレーションを用います。

サンプルコード

チュートリアル応用編

ブラックボックス最適化 (4)

プログラミング難易度 ★★★★☆

流体機器設計に不可欠な翼型の最適化問題を取り上げます。最適化には、組み合わせ最適化及び機械学習に基づくブラックボックス最適化と流体シミュレーションを用い、翼の揚抗比を最大化するように翼型の探索を行います。

サンプルコード

デモアプリケーション

容量制約つき運搬経路問題 (CVRP)

プログラミング難易度 ★★★★☆

運送業における効率的な配送計画の策定やごみ収集や道路清掃における訪問順序の最適化等での応用が期待される容量制約つき運搬経路問題（CVRP）を取り扱います。

デモアプリ　サンプルコード

チュートリアル応用編

ブラックボックス最適化 (5)

プログラミング難易度 ★★★★☆

ブラックボックス最適化により、商業施設による交通集中が発生し得る都市における、交通渋滞を低減するような信号機群の最適制御を実施します。最適化の実施及び実証には、マルチ・エージェント・シミュレーションによる交通シミュレーションを用います。

サンプルコード

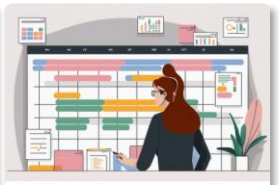
チュートリアル応用編

定式化による交通信号機の最適化

プログラミング難易度 ★★★★☆

都市における渋滞を最小化するために、刻一刻と変化する交通状況に応じ、組合せ最適化を用いてリアルタイムに信号機の最適制御を実施します。また、その様な信号機制御を実施した際の都市の交通量をシミュレーションします。

サンプルコード



チュートリアル応用編

10. 整数長ジョブスケジューリング問題

プログラミング難易度 ★★★☆☆

あらかじめ決まった数のジョブとマシンがあり、それぞれのジョブにかかる時間が分かっているとします。それぞれのジョブをいずれかのマシンに割り当てます。ジョブスケジューリング問題では、最も早く全ジョブが完了するような割り当て方を求めます。

サンプルコード

チュートリアル基礎編

画像のノイズ除去

プログラミング難易度 ★★★☆☆

画像のノイズ除去を行うアプリケーションを開発します。

サンプルコード

チュートリアル応用編

会議室割当問題

プログラミング難易度 ★★★☆☆

制約条件を用いて定式化するアプリケーションの例として会議室割当問題のアプリケーションを開発します。

サンプルコード

チュートリアル応用編

タクシーマッチング問題

プログラミング難易度 ★★★☆☆

目的関数と制約条件を用いて定式化するアプリケーションの例としてタクシーマッチング問題のアプリケーションを開発します。

サンプルコード

デモアプリケーション

グラフ彩色問題

プログラミング難易度 ★★★☆☆

Fixstars Amplifyによる、グラフ彩色問題の定式化を体験します。

デモアプリ　サンプルコード

デモアプリケーション

巡回セールスマン問題

プログラミング難易度 ★★★★☆

Fixstars Amplifyによる、巡回セールスマン問題の定式化を体験します。

デモアプリ　サンプルコード

デモアプリケーション

数独

プログラミング難易度 ★★★☆☆

Fixstars Amplifyによる、数独の定式化を体験します。

デモアプリ　サンプルコード

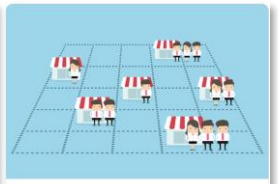
デモアプリケーション

ライドシェア

プログラミング難易度 ★★★★★

集合型ライドシェアの最適化アプリケーションを体験します。

デモアプリ　サンプルコード



デモアプリケーション

タスク割当問題

プログラミング難易度 ★★★★★

店舗とタスクに従業員を割り当てる組合せ最適化問題のアプリケーションを体験します。

デモアプリ　サンプルコード

デモアプリケーション

ポートフォリオ最適化

プログラミング難易度 ★★★★☆

リスクとリターンを考慮した株式ポートフォリオの最適化アプリケーションを体験します。

デモアプリ　サンプルコード

ブラックボックス最適化

通常の組合せ最適化とブラックボックス最適化

通常の組合せ最適化

- **目的関数** を定式化可能（例：QUBO）
  - 数の分割問題（**差** を最小化）
    $f = [\Sigma(2q_i - 1)a_i]^2$
  - 経路最適化（**経路距離** を最小化）
    $f = \Sigma\Sigma\Sigma d_{i,j} q_{n,i} q_{n+1,j} \cdots$

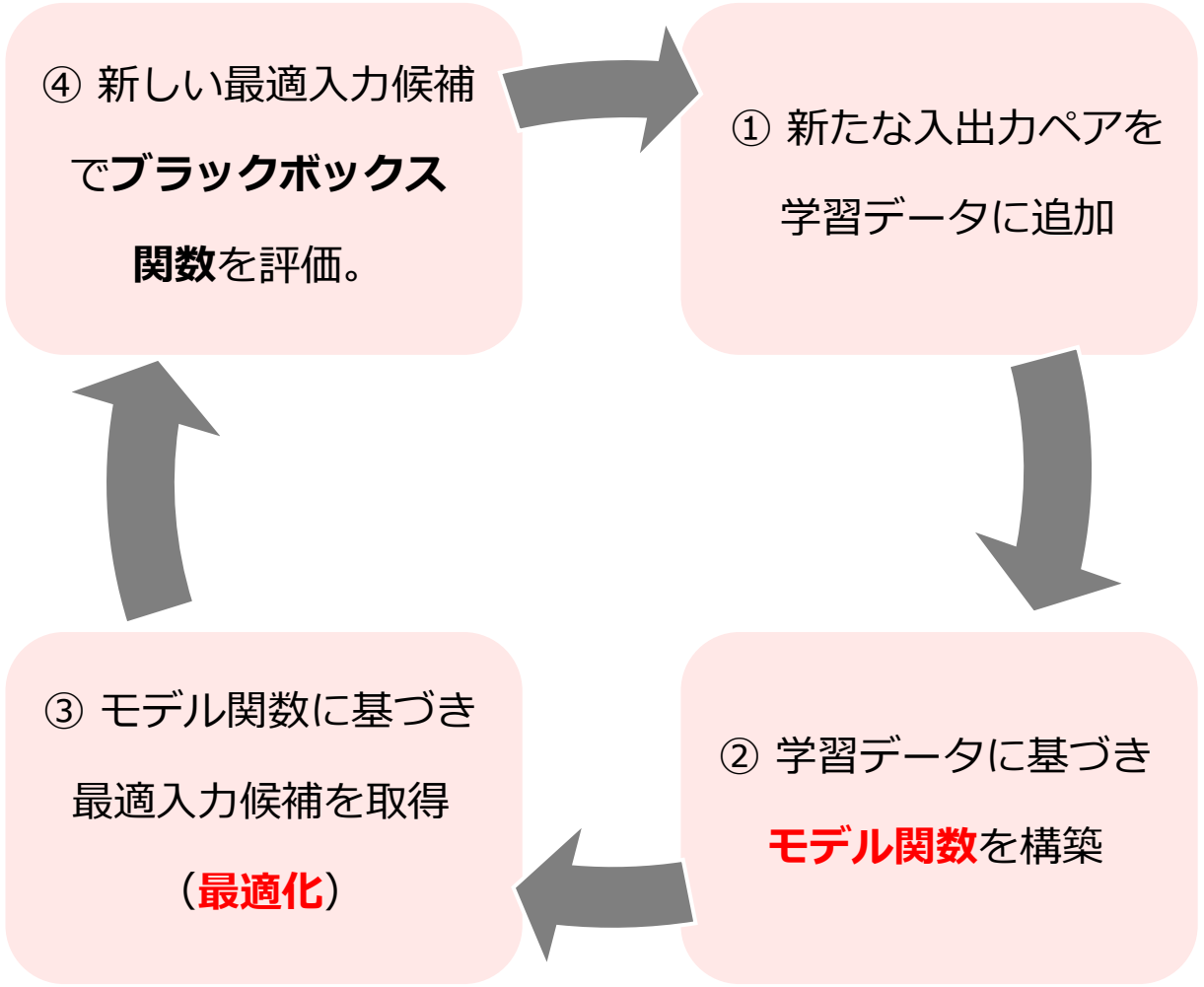
- **最適化の実施**
  - イジングマシンなどより、定式化された目的関数を最小化する

ブラックボックス最適化 (BBO)

- **定式化が困難な** ブラックボックス **目的関数**（解析・実験）
  - 低 **損失** 流体デバイス形状の同定
  - 高 **性能** 材料・構造トポロジーの探索
  - モデルの **誤差** を最小化するハイパーパラメータ最適化

- **最適化の方法**
  - シミュレーションや実験を併用しながら、**試行錯誤**に基づくアプローチ

ブラックボックス最適化のフロー



QA-BBO

モデル関数とQAを使うBBO手法の総称

- **FMQA** (Kitai, et al., *Phys. Rev. Res.*, 2020)
  - **モデル関数 → FM** **F**actorization **M**achine
  - **最適化 → QA** **Q**uadratic-optimization **A**nnealing
- **Kernel-QA**
  - **モデル関数 → Kernel model**
  - **最適化 → QA**

QA-BBO の特徴

- **高次元**の最適化問題にも強い！
  （次元の呪い）
- **制約条件**にも強い！

ブラックボックス最適化 活用例

材料分野に限らず、幅広い分野へ適用可能

QA-BBO: 活用例 (Amplify チュートリアル)

Amplify デモ　検索

チュートリアル応用編

ブラックボックス最適化 (2)

プログラミング難易度 ★★★★☆

機械学習と量子アニーリング・イジングマシンを活用するブラックボックス最適化の適用例として、疑似的な高温超電導を実現する材料探索を取り扱います。

サンプルコード

材料最適化

FMQA × 物理モデル

チュートリアル応用編

ブラックボックス最適化 (3)

プログラミング難易度 ★★★★☆

化学プラントにおける生産量を最大化するための運転条件最適化を行います。最適化には、機械学習モデルに基づくブラックボックス最適化と化学反応に関する物理シミュレーションを用います。

サンプルコード

化学プラント運転条件最適化

FMQA × 化学シミュレーション

チュートリアル応用編

ブラックボックス最適化 (4)

プログラミング難易度 ★★★★☆

流体機器設計に不可欠な翼型の最適化問題を取り上げます。最適化には、組み合わせ最適化及び機械学習に基づくブラックボックス最適化と流体シミュレーションを用い、翼の揚抗比を最大化するように翼型の探索を行います。

サンプルコード

翼形状最適化

FMQA × 流体シミュレーション

チュートリアル応用編

ブラックボックス最適化 (5)

プログラミング難易度 ★★★★☆

ブラックボックス最適化により、商業施設による交通集中が発生し得る都市における、交通渋滞を低減するような信号機群の最適制御を実施します。最適化の実施及び実証には、マルチ・エージェント・シミュレーションによる交通シミュレーションを用います。

サンプルコード

信号制御最適化

FMQA × マルチ・エージェント・シミュレーション

チュートリアル応用編

ブラックボックス最適化 (6)

プログラミング難易度 ★★★★☆

ブラックボックス最適化により、撹拌性能に影響を与える設計パラメータに対して、混合効率が最大化されるような撹拌機の最適設計を実施します。最適化の実施および評価には、濃度分布に基づく簡易的な撹拌シミュレーションを用います。

サンプルコード

機器設計最適化

FMQA × 撹拌シミュレーション

FIXSTARS Amplify

QA-BBO: 活用例 (Amplify ユーザー)

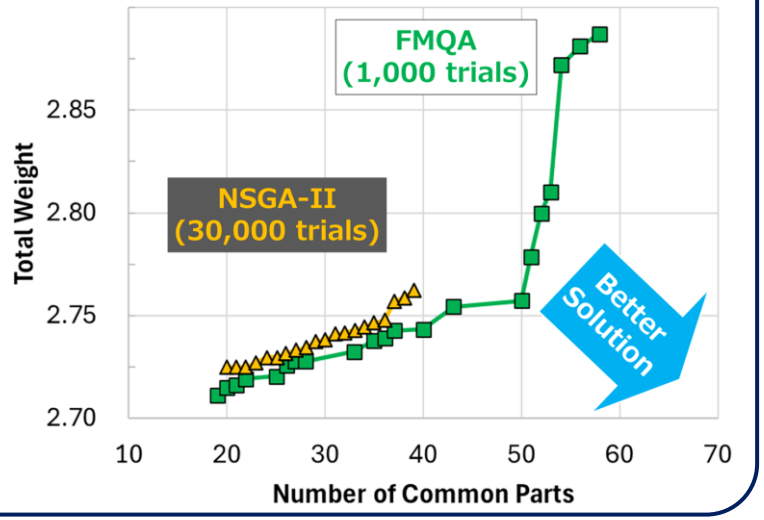
活用領域 化学、 創薬、 食品、 自動車、 電機、 通信、 重工、 エネルギー、ヘルスケア・・・

- 非線形現象の逆問題
- 機械学習：コスト↓精度↑
- 設計開発における部品選定
- 材料配合最適化
- 多目的最適化
- 物理モデルの簡略化

QA-BBO: 実際の応用事例

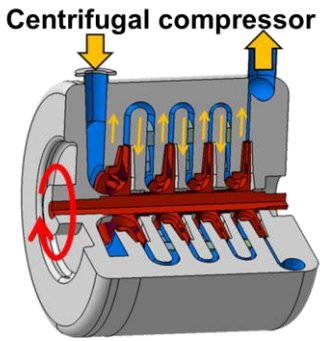
マツダ 様

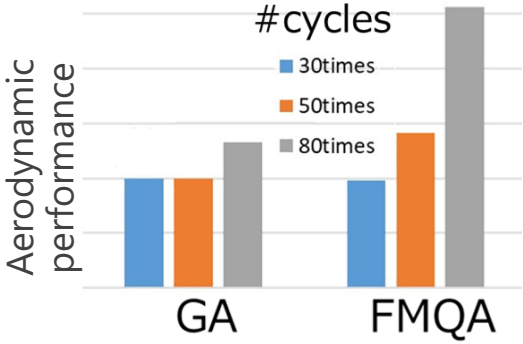
- 市販車両を対象した、複数車種同時設計最適化問題
  実数変数 200 以上の大規模問題
- **多目的最適化：**車体の軽量化と共通部品点数の最大化
- **制約条件：50 以上の制約条件**
  （衝突性能、製造制約、構造制約など）
- 従来手法 (GA/BO) と比較し、3%程度のシミュレーションコストでの最適化を実現。従来手法と同等以上の解を見つけることに成功



川崎重工業 様

- **ターボ機械**の設計最適化問題
- 従来より商用最適化ソフトによる遺伝的アルゴリズム (GA) を使用
- **最適化規模が大きくなると**最適解の求解までに時間がかかり、開発期間が長期化するといった課題
- 従来ツールに比べ、**同じシミュレーション回数**でも**より優れた解**が得られた





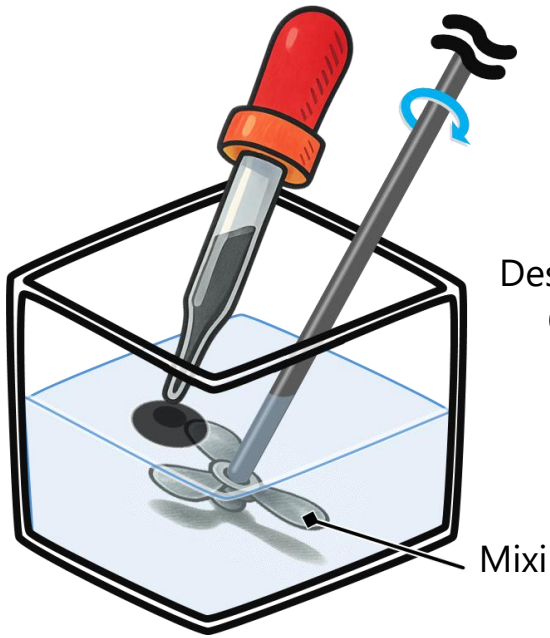
ブラックボックス最適化ハンズオン

問題設定と目的関数

問題設定

撹拌機器設計における
設計パラメータの最適化

- 撹拌性能を最大化（撹拌後の濃度分散を最小化）するように設計パラメータを最適化



- 目的関数（ブラックボックス）

入力
設計パラメータ $(x_0, x_1, x_2, x_3, x_4)$

入力値の評価
与えられた設計パラメータに基づく撹拌機で、一定時間攪拌シミュレーションを実施

撹拌中の濃度分布の時間発展

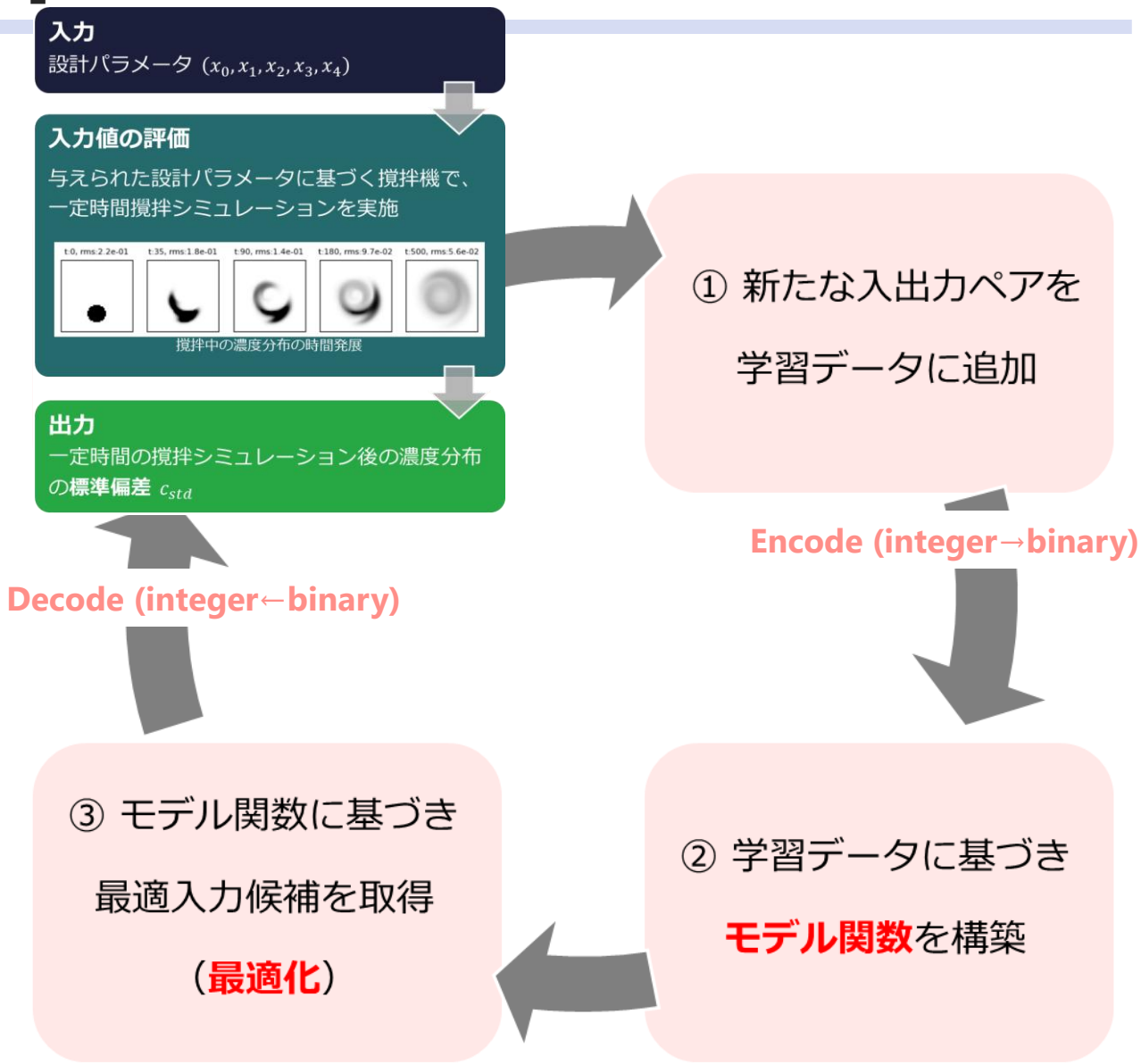
出力
一定時間の撹拌シミュレーション後の濃度分布の**標準偏差** c_{std}

QA-BBO活用の3方針

- **Amplify SDK + PyTorch**
  - 実装コスト：大
  - 柔軟性：大
- **Amplify-BBOpt ★**

  amplify.fixstars.com/ja/docs/amplify-bbopt/v1/
  - 実装コスト：小～中
  - 柔軟性：中
- **Web App**

  (Amplify-BBOpt Studio)



ブラックボックス最適化ハンズオン

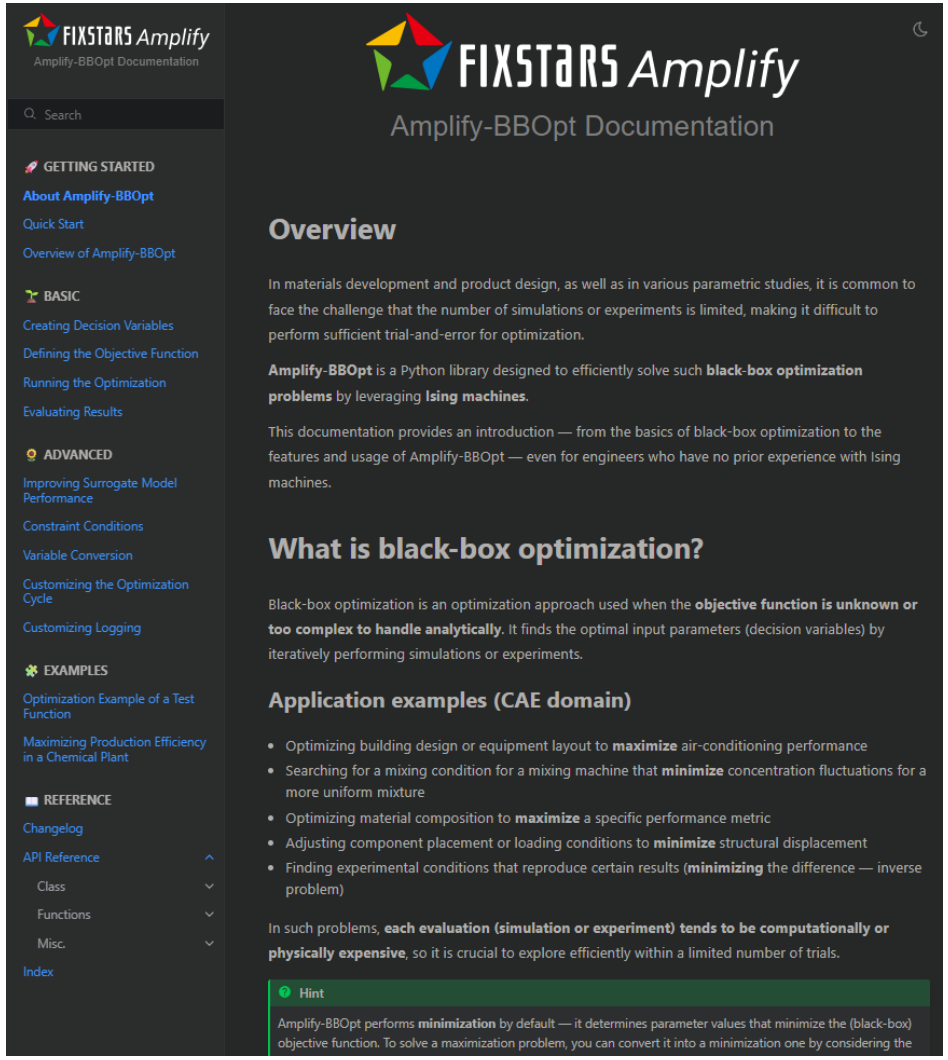
Amplify-BBOptによる実装

Amplify-BBOpt: 特徴

- **QA-BBO** の簡単実装：
  - 目的関数の定義（**シミュレーション**や**実験**）
  - イジングマシンによる最適解の**効率的な探索**
  - 過去の結果に基づき、**次に実施するべき**シミュレーション・実験への**入力条件を提示**
- Amplify-BBOpt で使えるアルゴリズム：
  - **F**actorization **M**achine with **Q**uadratic-optimization **A**nnealing (**FMQA**) [1]
  - Polynomial-Based **Kernel** with **Q**uadratic-optimization **A**nnealing (**Kernel-QA**) [2]

[1] Kitai, K., Guo, J., Ju, S., Tanaka, S., Tsuda, K., Shiomi, J., Tamura, R. Phys. Rev. Res. 2, 013319 (2020).

[2] Y. Minamoto and Y. Sakamoto, arXiv:2501.04225 (2025).



amplify.fixstars.com/ja/docs/amplify-bbopt/v1/

実装 (1/4): ブラックボックスな目的関数

```python
from amplify_bbopt import blackbox, IntegerVariable


@blackbox
def bbfunc(x0: int = IntegerVariable(bounds=(2, 10)),
           x1: int = IntegerVariable(bounds=(5, 20)),
           x2: int = IntegerVariable(bounds=(0, 45)),
           x3: int = IntegerVariable(bounds=(1, 5)),
           x4: int = IntegerVariable(bounds=(1, 4))) -> float:
    s = MixingSimulator(x0, x1, x2, x3, x4)
    c_std = s.simulate(duration=500)
    s.plot_evolution()
    print(f"{c_std=:.3f}")
    return c_std
```

Amplify-BBOpt では、決定変数の定義をブラックボックス関数の定義で実施する。

入力
設計パラメータ $(x_0, x_1, x_2, x_3, x_4)$

入力値の評価
与えられた設計パラメータに基づく撹拌機で、一定時間攪拌シミュレーションを実施

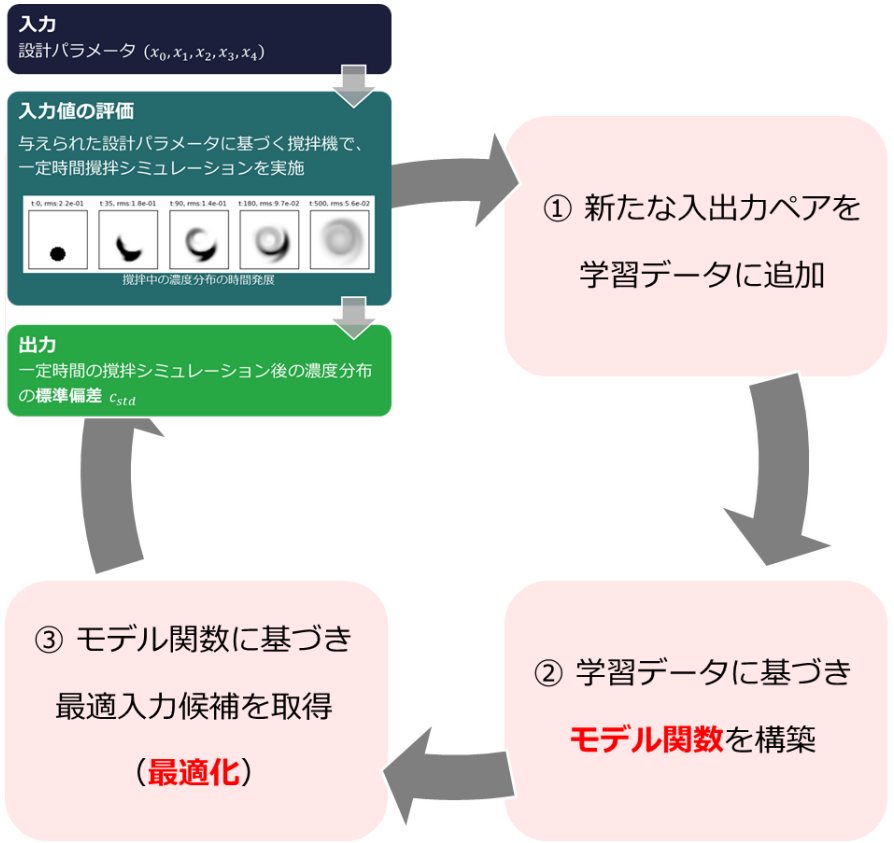
撹拌中の濃度分布の時間発展

出力
一定時間の撹拌シミュレーション後の濃度分布の**標準偏差** c_{std}

実装 (2/4): 初期学習データ

```
# 初期学習データ（入力ベクトル10サンプル）
x_init = np.array([
    [ 2,  6,  0,  1,  4],
    [ 4, 18, 30,  1,  2],
    ...
            ])

# 初期学習データ（対応する出力ベクトル10サンプル）
y_init = np.array([
    0.15041756,
    0.17410071,
...
    ])
```



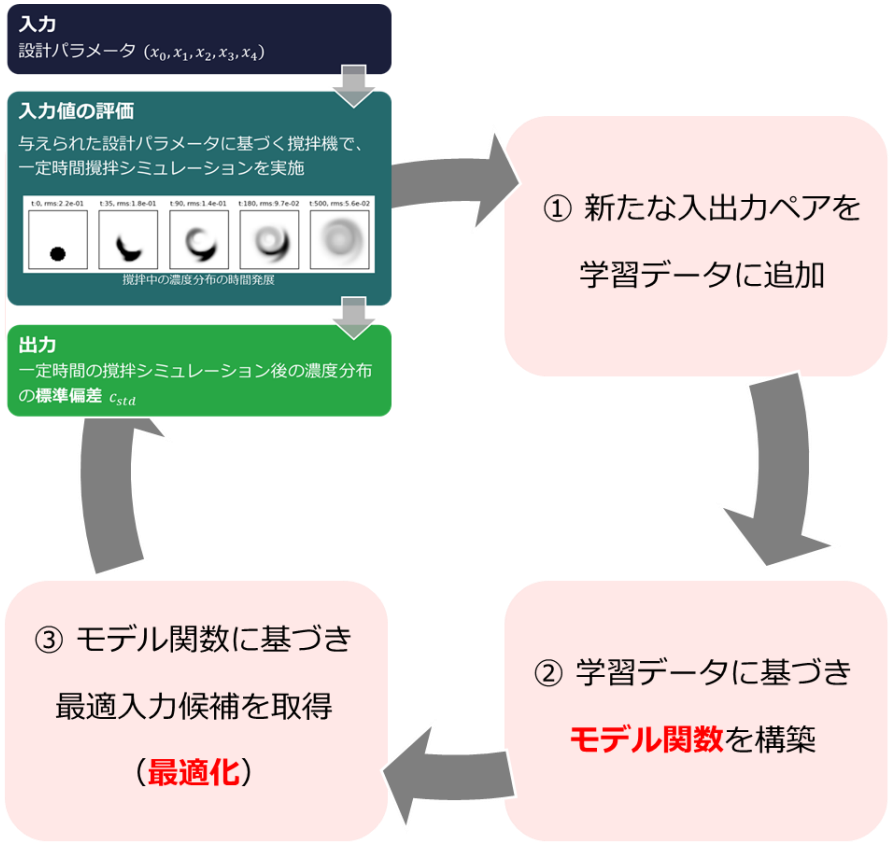
ごく少量の学習データサンプルが必要
既存のデータがない場合、Amplify-BBOpt で初期学習データを生成することも可能。

実装 (3/4): ソルバーの設定

```
from amplify import AmplifyAEClient
from datetime import timedelta


# ソルバークライアントを Amplify AE に設定
client = AmplifyAEClient()
client.token = "XXXXXXXXXXXXXXXXXXX"  API トークン
client.parameters.time_limit_ms = timedelta(milliseconds=2000)
```

- Fujitsu DA
  - **FujitsuDA4Client()**
- TOSHIBA SQBM+
  - **ToshibaSQBM2Client()**
- D-Wave Systems
  - **DWaveSamplerClient()**
  - **LeapHybridSamplerClient()**
  - **LeapHybridCQMSamplerClient()**



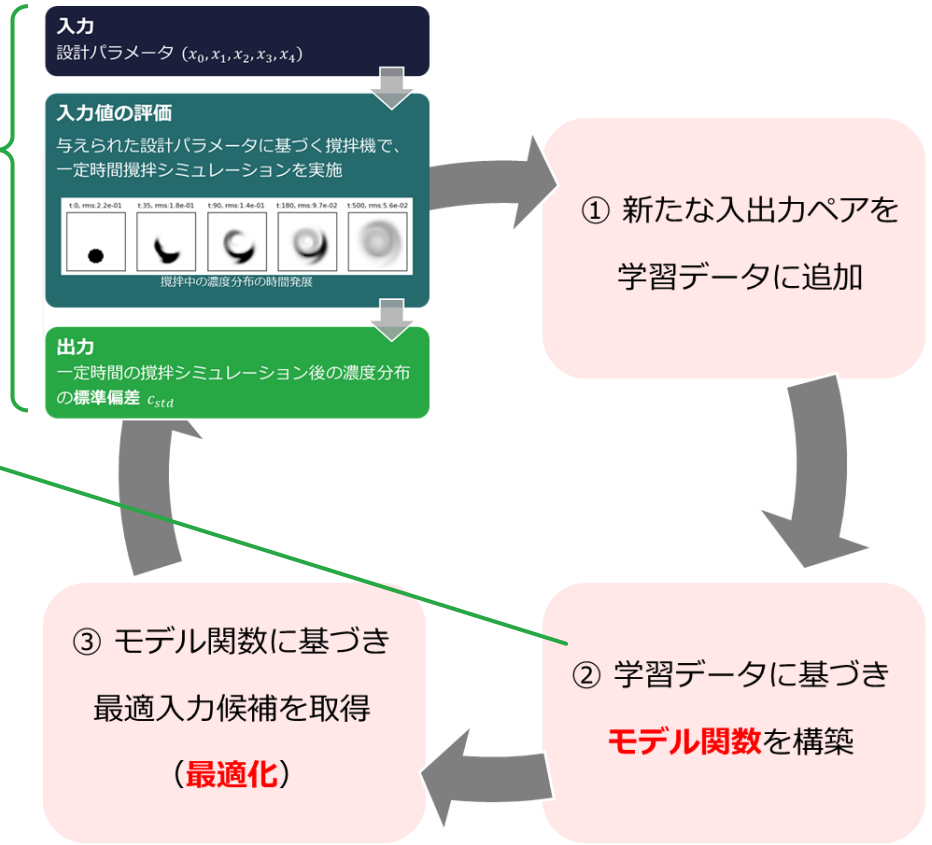
実装 (4/4): 最適化サイクル

```python
from amplify_bbopt import Optimizer, FMTrainer, Dataset


optimizer = Optimizer(blackbox = bbfunc,
                      trainer =  FMTrainer(),
                      client = client,
                      training_data = Dataset(x_init, y_init))
                      # (x_init, y_init): 初期学習データ


# 最適化サイクルを実施
optimizer.optimize(num_iterations=20)
```

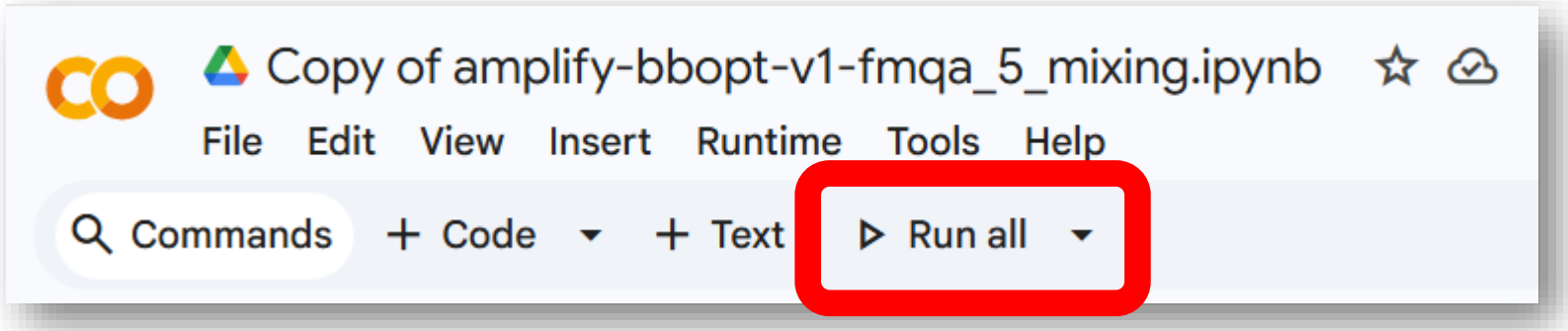
- 様々な制約条件も追加で考慮可能 [Link]



ハンズオン③

- ブラックボックス最適化サンプルコードを触ってみましょう。
  https://colab.research.google.com/drive/1a8XP0-wsDbrWa8Lp3PGDhonbuAX_hIrx

1. ドライブにコピーを保存
2. APIトークンをコピペ
3. 設計最適化サンプルコードを上から実行
4. 度々、新しく発見された設計パラメータが目的関数値のベストを更新していくことを確認
5. 最適化履歴を確認



Fixstars Amplify
ご利用プラン

クラウド利用料

個人単位のプラン
～ 主に研究者・開発者向け ～

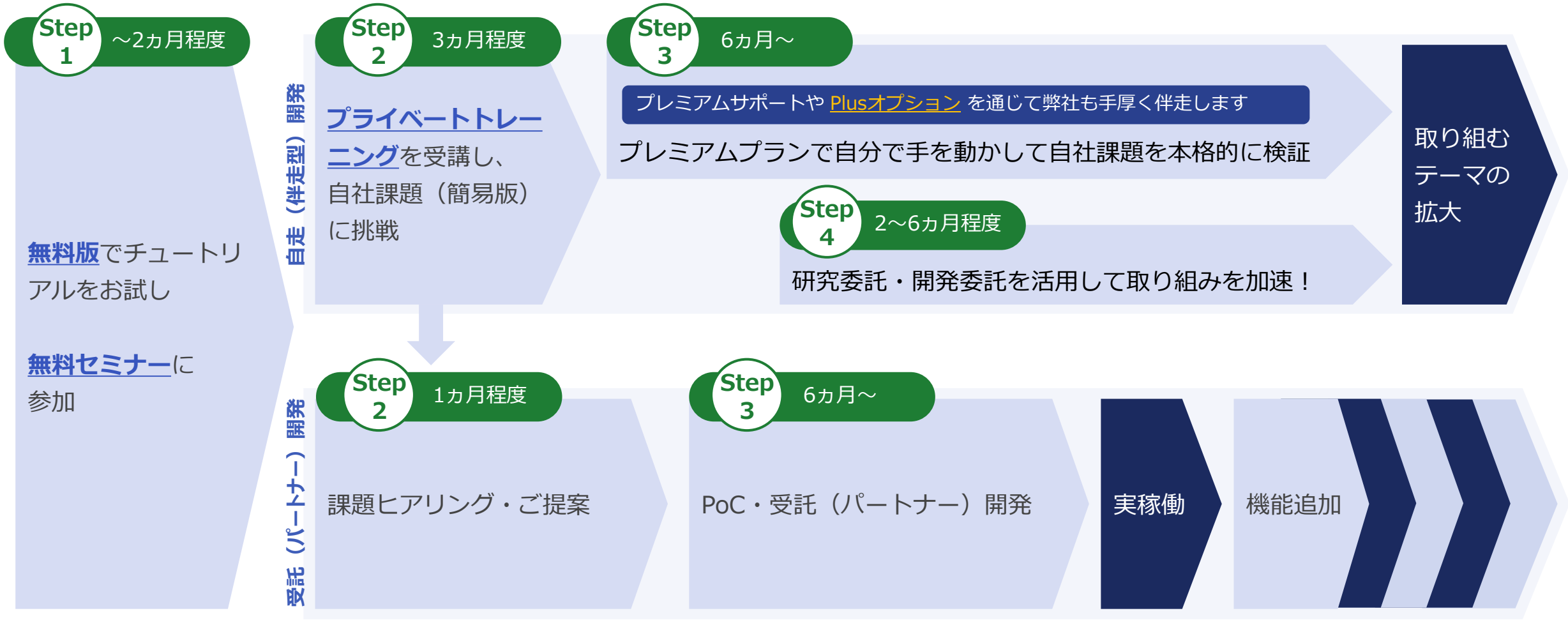
(金額は税抜)	ベーシック	スタンダード	プレミアム	Sプレミアム
月額利用料	**無料**	**10万円** (1名) **30万円** (最大5名)	**20万円** (1名) **60万円** (最大5名)	**30万円** (1名) **90万円** (最大5名)
計算環境	**スモール**	**ミディアム**	**ラージ**	**スーパーラージ**
利用GPU (マルチGPUオプションあり)	NVIDIA V100	NVIDIA V100	NVIDIA A100	NVIDIA H100
1ジョブの実行時間 (実行時間延長オプションあり)	10秒	1分	10分	15分
月間実行回数上限 (実行回数追加オプションあり)	制限の可能性あり	無制限		
東芝 SQBM+オプション	無料	30万円 (1名)、90万円 (最大5名)		
富士通 DAオプション	無料	50万円 (1名)、150万円 (最大5名)		
D-Wave の利用	無料プログラム (3分/月)			
サポート	**ベーシック**	**スタンダード**	**プレミアム**	**プレミアム**
Plus オプション	-	-	月額50万/人	

定式化や実装を**手厚く**伴走支援します！

FIXSTARS Amplify

自立開発向け・受託開発向けのおすすめの進め方

二次・非線形を上手に使いこなせるように、**弊社と一緒に**取り組みを進めていきましょう！



セミナー・トレーニングのご紹介

https://amplify.fixstars.com/ja/news/seminar

お客様の実際の課題解決をご支援するために、**無料セミナー**や**有償トレーニング**を提供しています。

無料セミナー・ワークショップ

ビジネス向け、エンジニア向けに分けて
開催しています！

ビジネス向け

製造業向け量子コンピュータ時代のDXセミナー

見える化、予測・分析、その先の最適化へ

組合せ最適化問題や量子アニーリング・イジングマシンの概要をご紹介したのち、製造業における組合せ最適化を活用したDX推進の一例として、生産計画最適化や生産ラインのシフト最適化などの事例とデモをご紹介いたします。「Fixstars Amplify」を通じて量子アニーリング・イジングマシンを活用することで、どのようなビジネス上の効果が期待できるのかを感じていただきたいと思います。

エンジニア向け

製造業向け量子コンピュータ時代のDXセミナー

最適化の中身を覗いてみよう

製造業における組合せ最適化を活用したDX推進の一例として、生産計画最適化、勤務シフト最適化などの事例を用いて、問題設定の考え方、目的関数や制約条件の定式化、実装のポイントなどを実際のコードを見ながら解説します。また、サンプルコードを用いて、ご自身の環境で実際に量子アニーリング・イジングマシンを動かす体験をしていただきます。

企業向けプライベートトレーニング

お客様が抱える実際の課題やデータを使った
カスタムメイド のトレーニングです！

全4回のレクチャーとお客様に実施いただく「課題」を含む約1.5か月のコースです。コースの前半では、量子アニーリング・イジングマシン専用の開発／実行環境であるFixstars Amplifyを用いてPython言語による組合せ最適化アプリケーション開発方法を学びます。後半では、お客様が抱える実際の課題やデータを使ったトレーニングを実施します。量子アニーリング・イジングマシンを使って実課題の解決に取り組んでみたい方に最適なコースです。

今後のセミナー予定・情報発信

定期的に無料セミナーを開催しています！

2026/6/11 （受付中）
「AIとの違いと意思決定プロジェクトの進め方」

経営層・マネジメント層のための最適化入門セミナー。AIによる予測を、意思決定に落とし込むために。

2026/6/25（予定）
「エネルギーマネジメント最適化 ハンズオン」

エネルギーマネジメント最適化をハンズオンで実施。

2026/7/16 （予定）
「ブラックボックス最適化技術解説」

量子・量子インスパイアード技術による、ブラックボックス最適化成功のヒントを解説

2026/7/30 （予定）
「シフト最適化ハンズオン」

従業員割り当ての最適化をハンズオンで実施。効率化だけでなく、スキル値や平準化を考慮した計画を立案します。

@FixstarsAmplify

ご質問・ご不明点がありましたら、お問い合わせフォームでお気軽にご連絡下さい
https://amplify.fixstars.com/ja/contact