

遺伝子的アルゴリズム サンプルモデル

1. 概要

CommonMPでは、演算の制御を行うプログラムを モデル開発者が組み込む事が可能です。 Ver1.0では、本機能は、限られたものですが、研究を進めるために、是非ともその機能を使用したい方の為に、一例として、遺伝的アルゴリズムを用いて、モデルのパラメータを自動調整するモデルを作成してみました。

2. CommonMPにおける 演算の制御



図2. 1 CommonMP Ver1.0 における演算制御

CommonMP Ver1.0では、基本的に、シミュレーションの時刻は、過去から未来に流れ、図2. 1に示すように、画面上で、開始時刻／終了時刻の設定、演算開始／中断等の制御が行えるようになっています。

しかしながら、モデルによっては、過去 → 未来 → 過去 → 未来 … のように、繰り返し演算を行いたい場合があるかもしれません。そこで、CommonMP Ver1.0では、モデルの一部を グループ化して、グループ要素の内部で モデルの制御論理を ユーザー側で実装する事を可能としております。

2. 1 グループ要素

ここでは、グループ要素について説明します。

例えば、図2. 2 の上段に示すように 5つの要素モデルからなる モデルを作成した場合を考えます。この時、中の3つの要素が互いにループを形成し、組み込んだモデルでは、単純に、過去 → 未来の演算制御では、計算出来ない場合があります。

この時、図2. 2の下段に示すように、例えば、上記3つの要素を グループ化し、3つの要素があたかも一つの大きなモデルとして見なせる様にします。全体で見れば、3つの要素が 直列に並び、通常の演算の制御で計算できます。しかしながら、グループ要素の内部では、お互いに影響を与えながら、繰り返し計算を行う必要があるかもしれません。この時、グループ要素は、外部のシミュレーション時刻の進行とは別に グループ内の3つの要素モデルの演算の順序や時刻の制御を行う必要があります。これらの制御は、どの様なモデルを作成するかによって異なり、CommonMP側では 一意に特定できません。

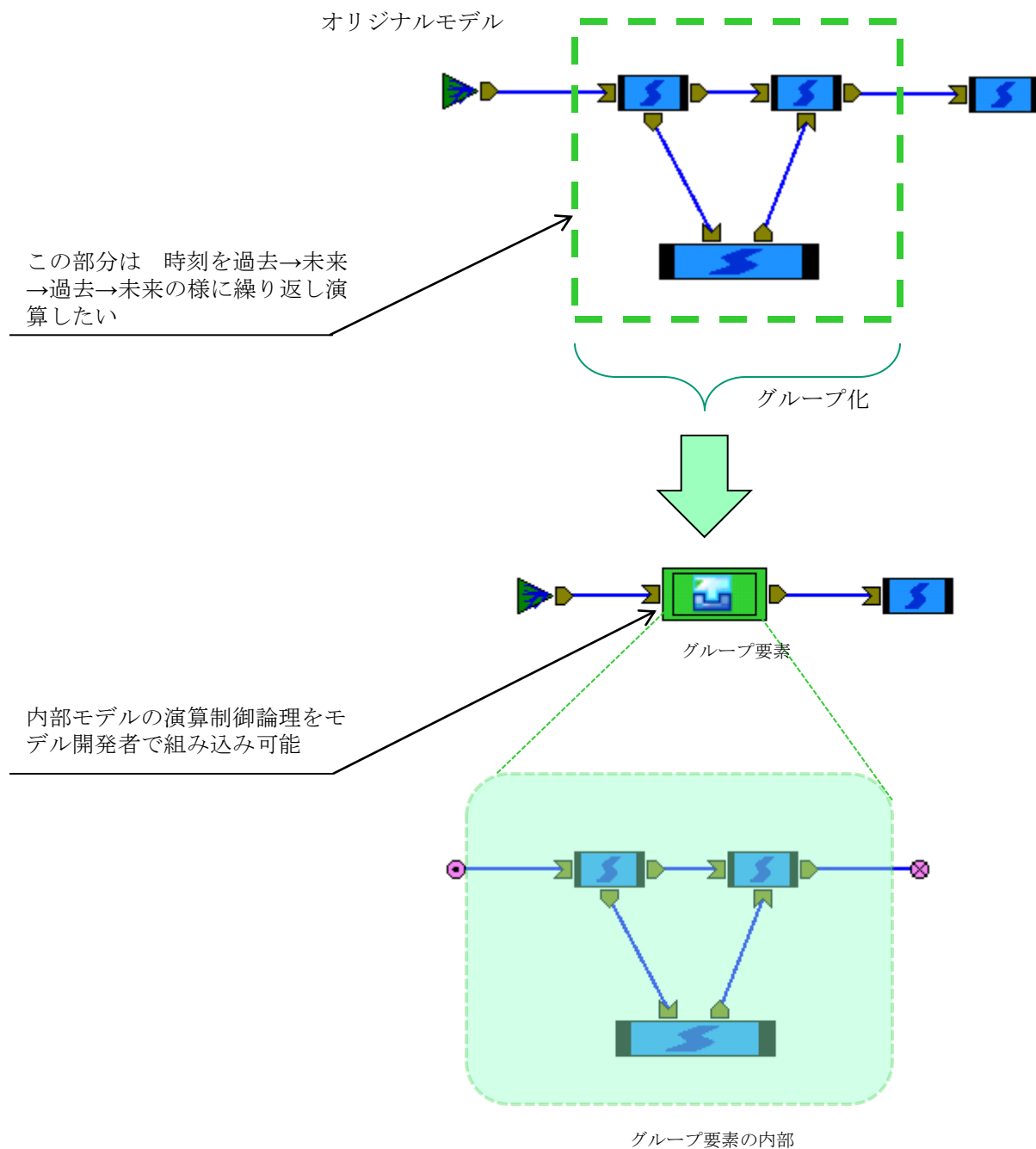


図2. 2 グループ要素の説明

そこで、モデル開発者は、グループの内部に含まれる要素モデルに対して、演算を制御する為の「制御モデル」を開発し、CommonMPへ登録します。（登録は通常の要素モデルと同じ手順で行えます。登録された「制御モデル」はライブラリ管理画面上に表示されます。）

制御モデルは、一般に 各要素に要素モデルを割り当てる手順と同じ要領で グループ要素に割り当てる事が出来ます。（但し、割り当てる事が可能なのは、グループ要素だけです。）

その様子を 図2. 3に示します。

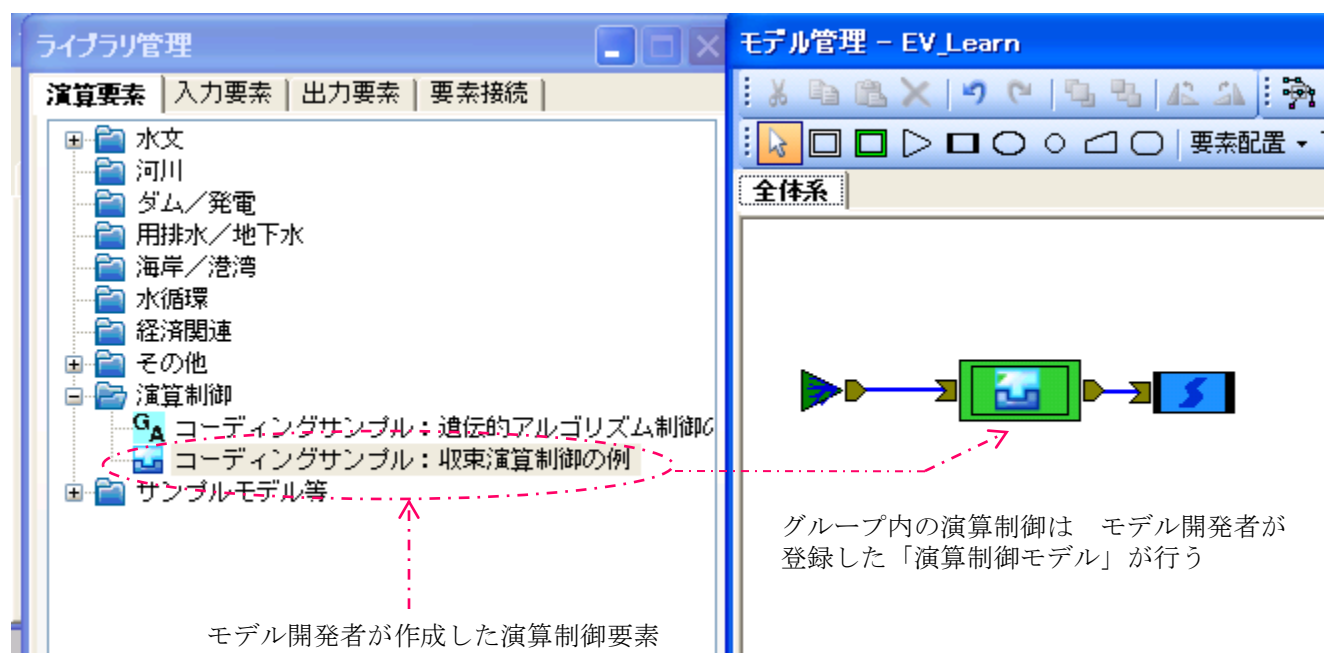


図2. 3 モデル開発者による演算制御論理の組み込み

演算制御要素組込みの例として、CommonMP Ver1.0では、収束演算の例

(McConvergenceSample フォルダ内 の McConvergenceSamplePrj.csproj) を示していますが、さらに 別な例として ここでは、遺伝的アルゴリズムを用いたモデル作成の例を示します。

2. 2 演算制御モデル

図2. 4に 遺伝的アルゴリズムを制御するクラスの派生関係を示します。基本的に モデルを制御するクラスは、モデルを制御する共通処理を組み込んだ「McGrElementCtl」クラスから派生させます。

本例では、「McGrElementCtl」の派生クラスである「McSubSystemCtl」から派生し、共通な処理部分は、全て親クラスの論理を使用できるようにしてあります。

本例の遺伝的アルゴリズムによる制御は、主に下記の処理(メソッド処理)を、オーバーライドして 自らの制御処理を実装します。

- Initializeメソッド：演算開始前の初期化を行います。
- Calculateメソッド：演算中の演算制御を行います。
- SetPropertyメソッド：内部制御に必要な情報を設定します。

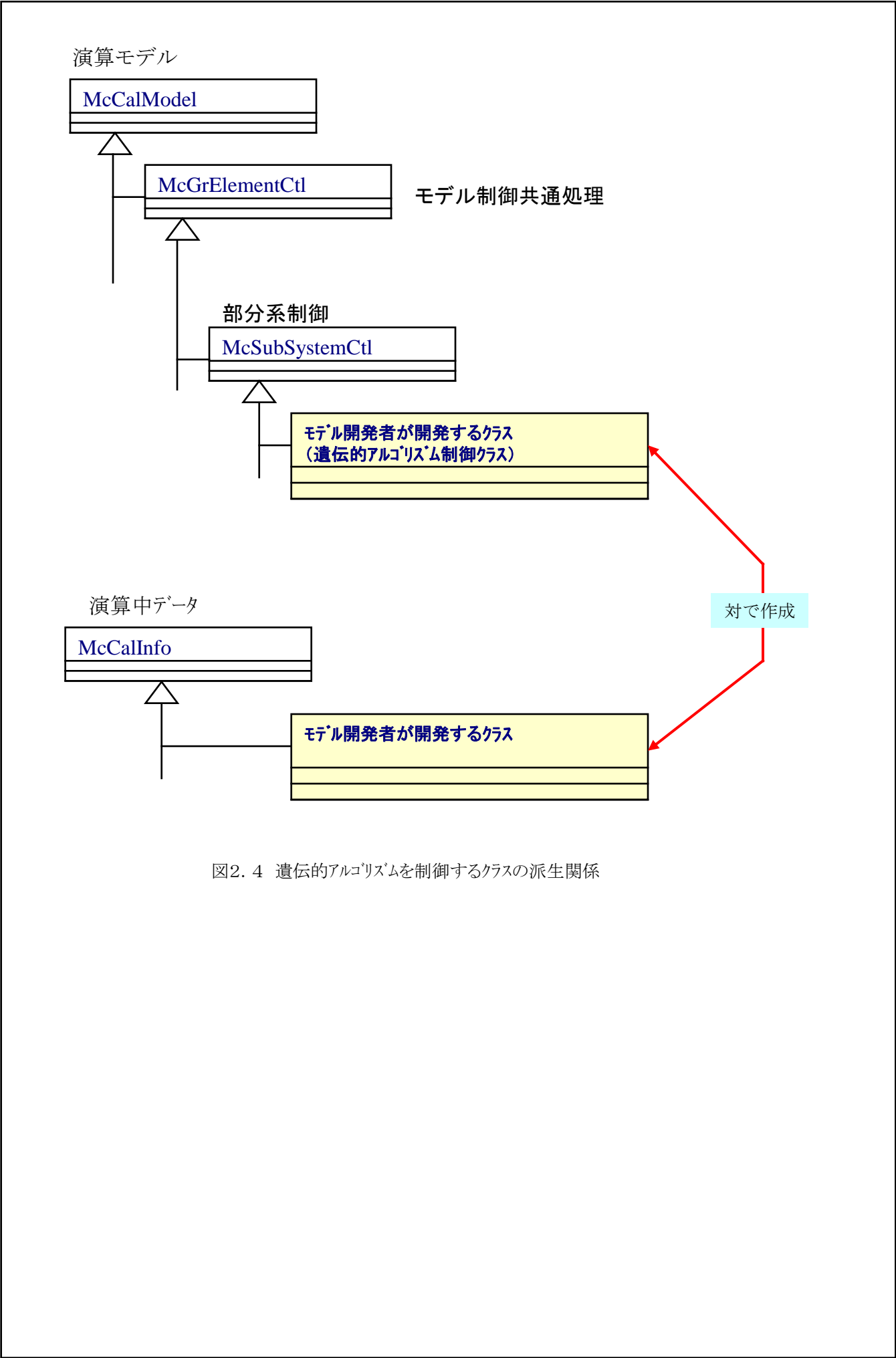


図2. 4 遺伝的アルゴリズムを制御するクラスの派生関係

3. 遺伝的アルゴリズムについて

ここで、遺伝的アルゴリズムについて簡単に触れ、本例では それをどの様に使用しているか説明します。

3. 1 遺伝的アルゴリズムとは

遺伝的アルゴリズムとは、自然界の進化の過程を真似て作られたアルゴリズムで、最適解の探索等によく利用されます。

遺伝的アルゴリズムの使用は下記によって行われます。

- ①問題をソフトウェア的な遺伝子としてコード化する。
- ②空間の中に、それぞれ異なった複数の遺伝子入れ、それらの遺伝子から発生した個体間で競争を行わせる。
- ③競争で勝ち残った(適合性の高い)個体が 次の世代に多く次の遺伝子を残す(淘汰)。
この時、適合性は 探索したい課題に合わせて評価を行う。
すなわち、適合度が高い個体は、より問題の解に近づくように行う。
- ④ 上記③で遺伝子を残す際、次世代の遺伝子生成時に以下の操作を行う。
 - a) 突然変異:ある確率で遺伝子に変異が発生させる。
 - b) 交叉(組み換え) :ある確率で 異なる個体間で遺伝子の一部を交換し合う。
- ⑤次の世代においても、②～④の操作を繰り返す。
この様にして、世代を重ねるに従って、探索が進行する。

3. 2 サンプルにおける遺伝的アルゴリズム適応により解決したい課題

本サンプルにおける 最適化の課題は、下記とします。

- ①現実の河川に対して、河川モデルを作成する。
- ②上記河川モデルが、現実の河川を良くモデリングする為に、パラメータを調整する。
方法としては、現実の河川で観測した値と、河川モデルの計算した値を比較し、河川モデルの計算値が観測値に近くなる様にパラメータを調整する。
(→ 図3. 1を参照)

但し、現実河川のモデルは複雑で 且つ、観測値等の入手が困難である事、 また、ここでは、現実の河川モデルを構築する事が目的ではなく、 演算制御の一例として 遺伝的アルゴリズムの制御を示す事が目的である為、 現実河川の代わりに パラメータ等を予め設定しておいた「**正解モデル**」を 現実河川に見立てて、 サンプルプログラムを作成します。

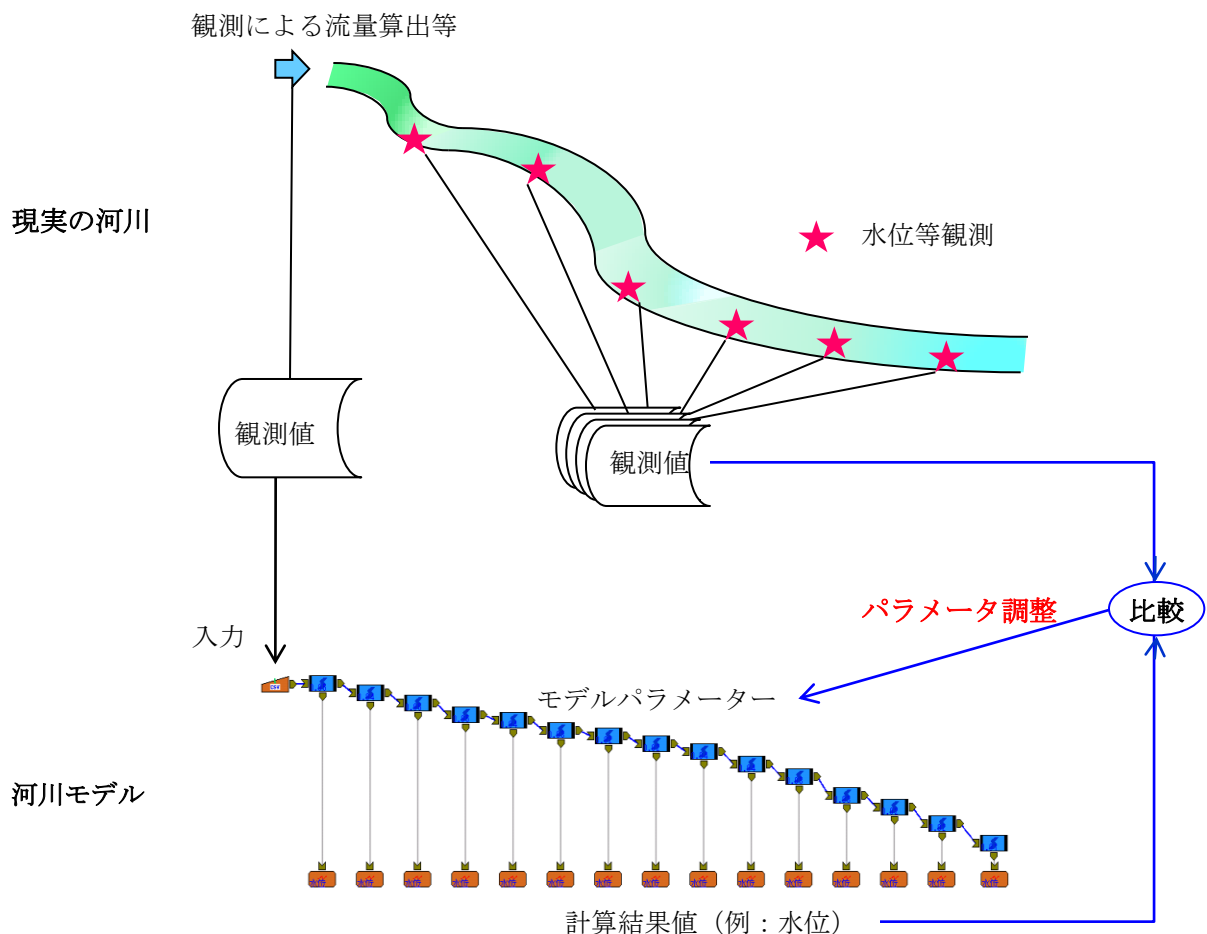


図 3. 1 課題の概念説明

本サンプルで実際に作成するモデルは、下記の通りとします。

①河川モデル

図3. 2に示すように 15個のKinematicWave法を用いた河道要素モデルを直列に接続し、一つの河川モデルとする。

②最適化するパラメーター

本サンプルでは、KinematicWave法を用いた河道計算に用いられるパラメータとして「粗度」を 遺伝的アルゴリズムによって最適化する。他のパラメーター（川幅、勾配等）は変更しない。

③正解モデル(現実の河川の代わり)

予め、正解モデルとして、15の各要素に対して、粗度を決定したモデルを作成しておく。

④最適化に用いる観測データ

本サンプルでは、実際の観測データの代わりに、正解モデルが計算した水位データを観測データと見なして用いる。

⑤最適化されるモデル

進化するモデルは正解モデルと同じ構造であるが、各要素の粗度の初期値を 乱数によって設定する。 遺伝的アルゴリズムにより、進化を重ねたモデルの各要素の粗度が、正解モデルの各要素に設定された粗度の値に近づいてゆく事を確認する。

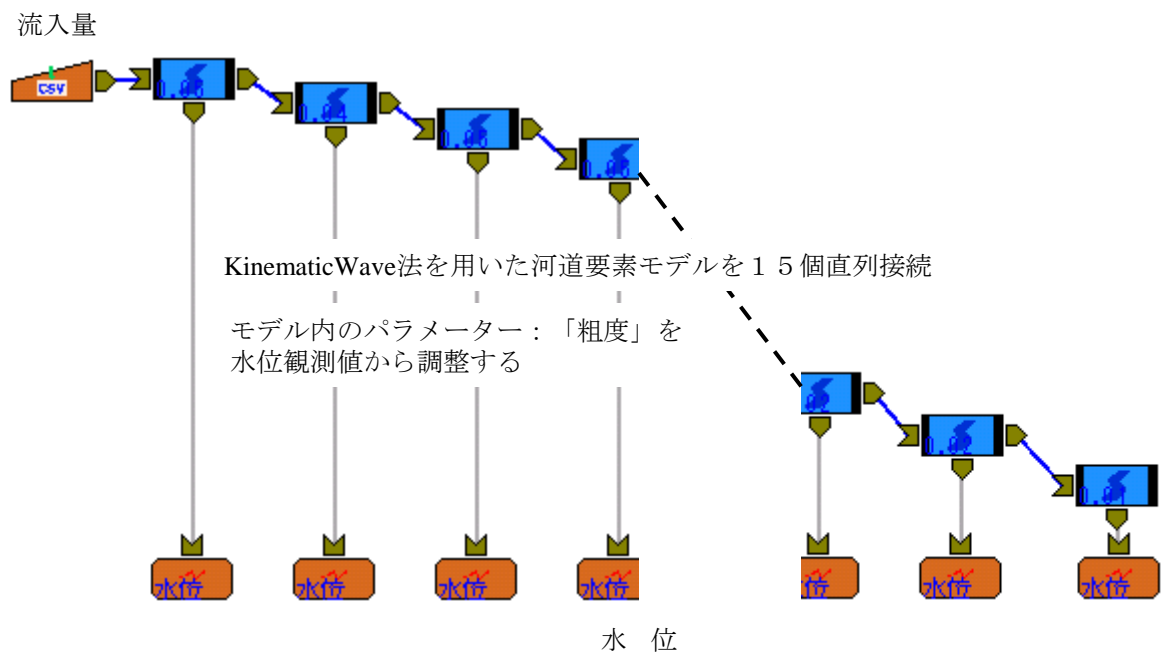


図3.2 サンプル河川モデル

3.3 モデルパラメータの遺伝子コード化

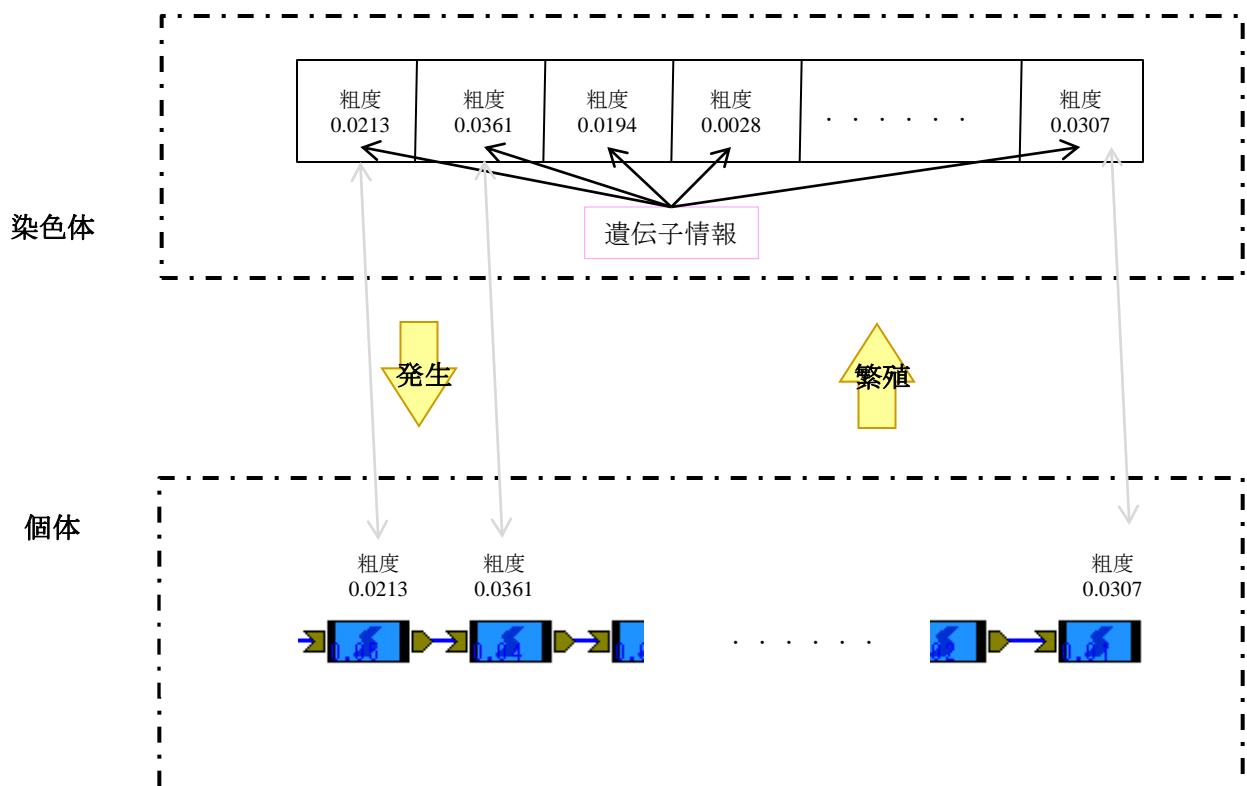


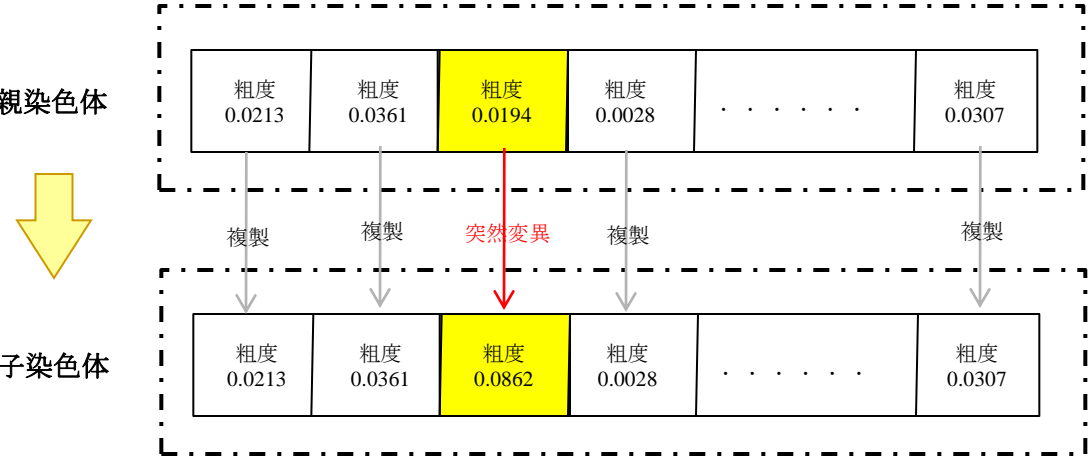
図3.3 遺伝子情報と個体

本サンプルでは、 遺伝子情報として、粗度だけを持ちます。 粗度情報は 図3. 3に示すように、染色体上に直列に配置されています。配列の順序は、個体内の各要素の順序と一致しています。

個体の発生時には、先ず、要素モデルが直列に接続して一つの個体を生成します。 染色体内の粗度値(遺伝子情報)が 順番に読み出されて、発生した個体の各要素モデルのパラメーター値:粗度に設定されます。

個体の繁殖時には、自分と同じ染色体(同じ粗度値を遺伝子情報として持つ)を複製します。 この時、ある確率で「突然変異」および、「交叉」を発生させます。 その様子を 図3. 4に示します。

・突然変異の発生



・交叉の発生

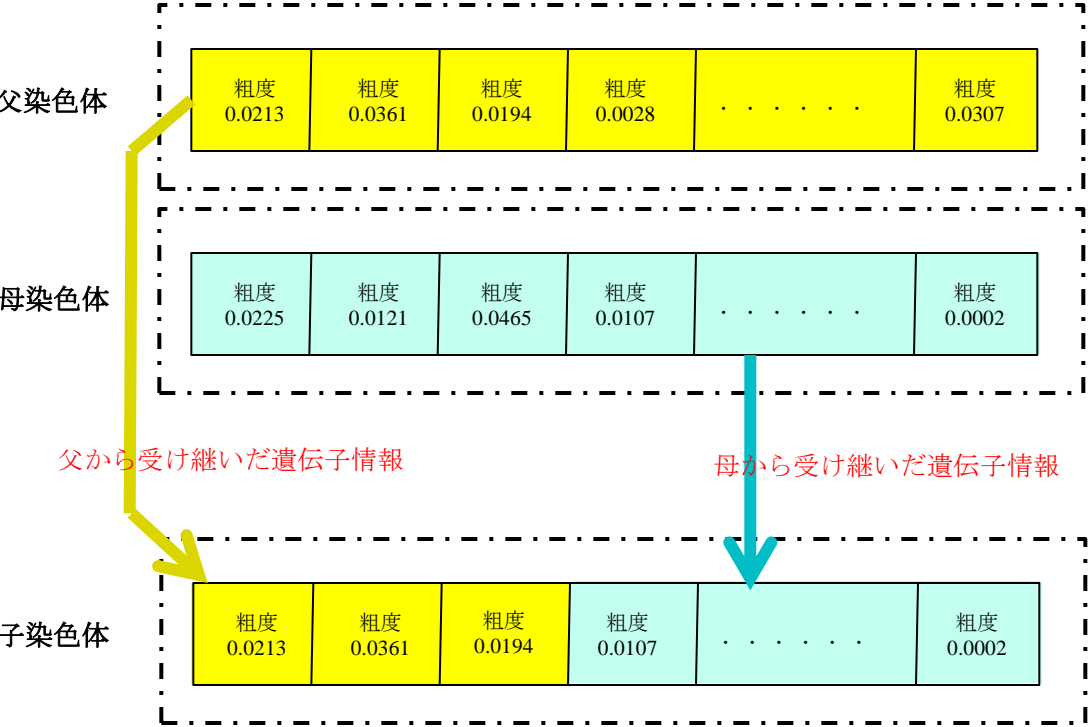


図3. 4 突然変異と交鎖

3. 4 個体の適合度

遺伝的アルゴリズムを適用する場合、各個体を意図した方向に進化させる為に、各個体の適合度を計算(評価関数)する必要があります。
本サンプルでは、個体の適合度を下記の様に定義しました。

$$\begin{aligned} \text{個体の適合度} &= \sum_{\text{個体の生涯}} \{ \log(1 / (2 \times \text{乗誤差})) \} \\ \text{ここに} \quad \text{2乗誤差} &= \sum_{\text{要素モデル数}} (\text{各要素モデル計算値} - \text{観測値})^2 \end{aligned}$$

この式は、河川モデルが計算によって求めた値と 実際の観測値を比較し、その差が小さければ、適合度が高いという事を示しています。
適合度が高い個体ほど、自分の遺伝子を子孫に伝える確率が高くなります。
適合値の計算に、「何故この数式を用いなければならないか？」という問い合わせに対して、論理的な回答はありません。
遺伝的アルゴリズムでは、最適化の評価式が理論的に明確でなくても正しい方向に進化してくれるというメリットがあります。

一般的に、遺伝的アルゴリズムを適応する場合には、①モデルの遺伝子コード化(如何に遺伝子にコード化するかという方法)と、②個体適合度を計算する評価の方法が 重要な 開発要素になりますが、 本サンプルでは その議論を深く掘り下げる事はしません。

3. 5 淘汰

各個体が 生涯に亘って計算した適合度に従って、 次の世代へ残す遺伝子を選択します。
本サンプルでは、伝統的な ルーレット選択を用います。

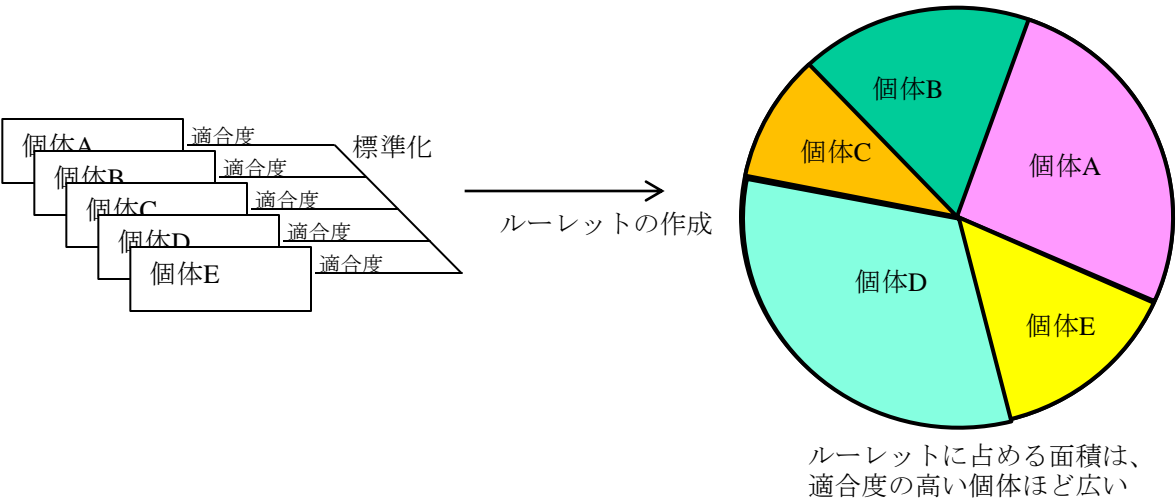


図3. 5に示すように、個体の適合度に従って ブロックの面積が異なるルーレットを作成します。 この時、適合度の高い個体ほど、ルーレットの面積が広くなる様にします。 すると、ルーレットを回したとき、選択される確率が大きくなります。

(注意)適合度をそのまま使用すると、個体間の適合度差が大きい時に、急激に遺伝子の多様性がなくなる為、適当に標準化を行います。

ルーレットによって選択された遺伝子を複製して、次世代の遺伝子とします。 この複製の段階で、ある確率で「突然変異」、「交叉」を発生させます。 選択は個体数分行い、次の世代の遺伝子の数が 現在の個体数と同じになった所で終了します。 尚、本サンプルでは、その世代で最も適合度が大きかった固体(スーパーエリート)について、次の世代に 1個体だけ、必ず完全な複製を残す事にしています。

3. 6 遺伝的アルゴリズムの計算制御

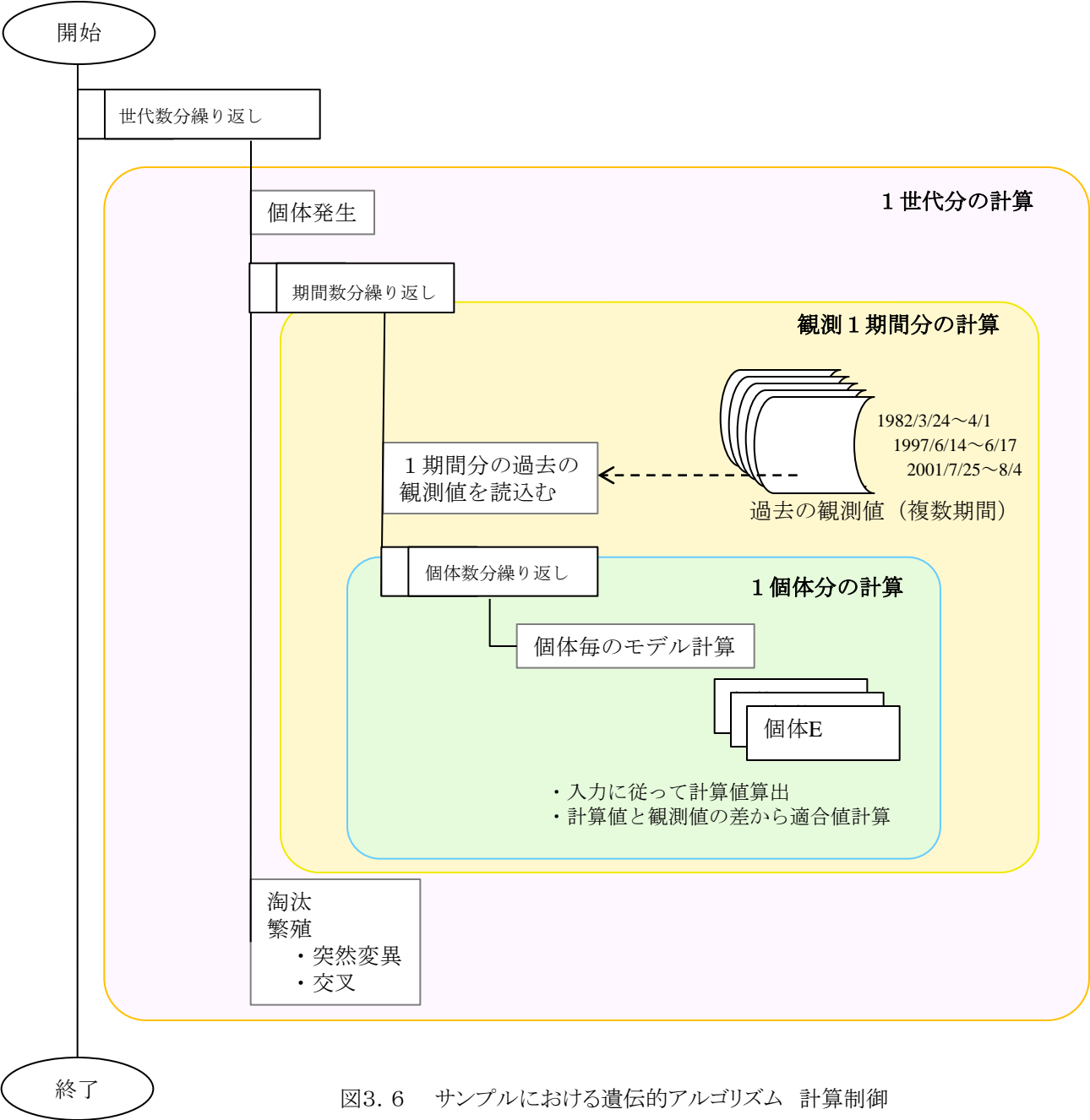


図3. 6 サンプルにおける遺伝的アルゴリズム 計算制御

図3. 6に 本サンプルにおける 遺伝的アルゴリズムの制御の概念を示します。
制御は 3つのループが 入れ子構造になっています。

一番内側のループは、発生した個体毎の計算を制御するループで 一回のループで一つの個体のモデル演算が行われます。例えば、1997/6/14～6/17 の観測データを用いて、個体Aが モデルの計算を行い、その計算結果と観測データの差から、自モデルの適合度を算出します。これを、全ての個体について行います。

次に、ループ制御は一つ外に移行し、次の観測データ(例えば2001/7/25～8/4)を用いて同じ様に、すべての個体の適合度を計算します。このループでは、観測期間に応じて、モデル内部のシミュレーション期間が変更され その都度、要素モデルを初期化し、所定期間演算を行う必要があります。

全ての観測データについて計算を行ったところで、個体の生涯が終了します。そこで、演算制御は、一番外側のループに移行し、既に述べた方法で、次の世代に残す遺伝子を選択し、その遺伝子の複製を作成します。

この様に、本モデルの制御内では、シミュレーション時刻は、過去から未来に一樣に流れるのではなく、過去から未来、未来から過去へと不連続的に制御する必要があります。

3. 7 遺伝的アルゴリズムの計算の実際

図3. 7に 実際に進化中の遺伝的アルゴリズム計算を行うモデルの構造を示しています。

上段は、CommonMP上で構築したモデルのハードコピーで、左は全体、右は 制御用のグループ要素の内部です。このモデル画面に対応した 各クラスの間関係を 図の下に示しています。制御用グループ要素(図中: **GAControlSample**) クラスの負担を軽減する為、**GAControlSample**クラスは、制御処理だけを行わせ、遺伝子の管理や、個体発生等の処理は 内部に 進化舞台(図中: **GAEvolutionSample**) クラスを設け、その中で行わせています。舞台クラス内部には、個体として、実際の河川モデル(図中: **GAKinematicWaveEvolutionModel**) が、複数存在しています。

進化中は、流入量を入力要素から読み込み、個体、すなわち河川モデル(**GAKinematicWaveEvolutionModel**) に水位を計算させ、その結果と、入力要素から読み込んだ 観測データを比較して 各個体(河川モデル)の適合度を 計算します。

世代の終わりに、その適合度から ルーレット選択により 次の世代の染色体(図中: **GACHromosomeSample**)を残し、また、化石として スーパーエリートの遺伝子を ファイル出力すると共に、スーパーエリートの適合度を 画面に出力します。

一つの世代が終了すると、先代の 残した染色体 (**GACHromosomeSample**) から次の世代の個体 (**GAKinematicWaveEvolutionModel**) を 発生し、同様の計算を行います。

本モデルでは、CommonMPの画面で設定する時刻と、モデルが実際に動作する時刻は関係がありません。従って、CommonMPの計算開始スタートボタンを押下して、計算が開始しても、表示される時刻には意味が無く、単に 計算が進行している事を示すのみとなります。(CommonMP Ver1.0では、繰り返し計算に対応した 制御画面はリリースされていません) 図3. 7のモデルで 適合度のモニター画面を表示状態にしておけば、何世代まで進化したかを知る事ができます。

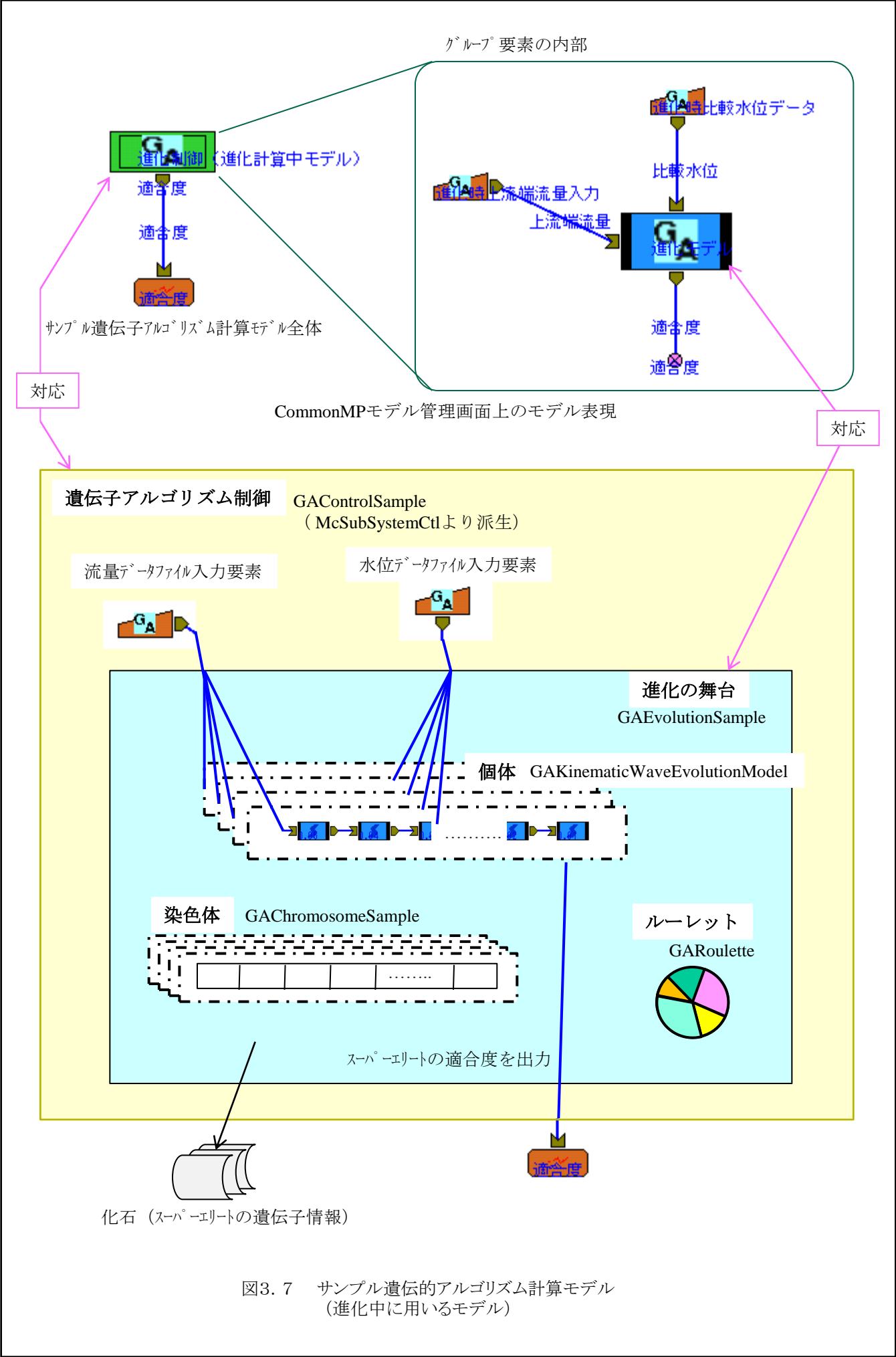


図3. 8は、個体 (`GAKinematicWaveEvolutionModel`) の内部を示したものです。
個体は、`CommonMP`の要素モデル親クラス (`McForecastModelBase`) から派生しています。
また、`KinematicWave`法による河道計算を 河道計算要素 (`GAKinematicWaveBox`) クラスに纏め、15個の `GAKinematicWaveBox` を直列に結合しています。
本サンプルでは、個体内部に `CommonMP`の要素モデルを持つと 制御処理が複雑になる為、`GAKinematicWaveBox`は、`CommonMP`の要素モデルクラスではなく、独立のクラスとして作成しています。

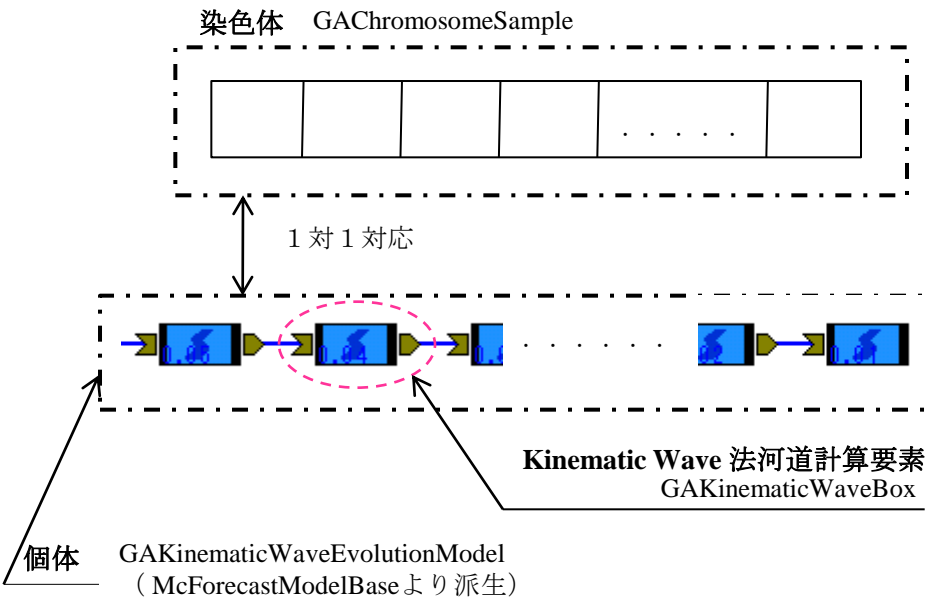


図3. 8 個体モデルの構造

4. 進化計算

4. 1 モデルの進化計算の実行

ここでは、実際に 進化計算を行わせて見ます。

①CommonMPの起動

CommonMP¥Source¥HYMCO¥OptionImpl¥ModelDeveloperExpressEditionフォルダー下の TestModelDeveloperMainExp.sln を ダブルクリックして、Microsoft社のVisual Studio C# から デバッグモードで CommonMPを立ち上げます。

②サンプルプロジェクトを開く

メニュー「プロジェクト」―「開く」から下記プロジェクトを開きます。

＜プロジェクトグループ＞ SampleModelDevelop

＜プロジェクト名＞ GASample_Learn.cmprj

(構造定義ファイルは¥SampleModelDevelop¥SCF¥GASample¥EV_Learn.xmlに定義されています。)

すると、図4. 1のようなモデルが生成されます。

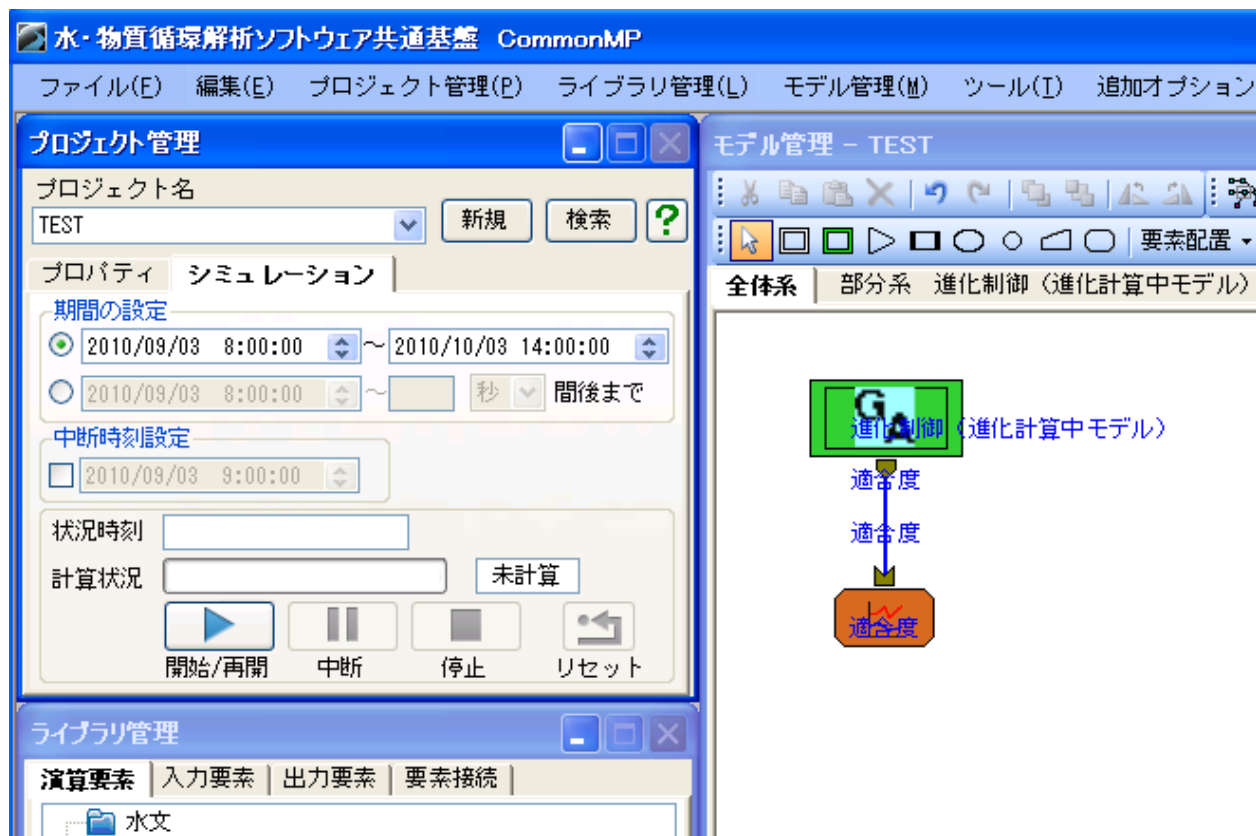


図4. 1 遺伝的アルゴリズム計算用モデル

③モニター画面表示

計算を開始する前に、図4. 2の通り、進化の適合度をモニターする画面を表示しておきます。

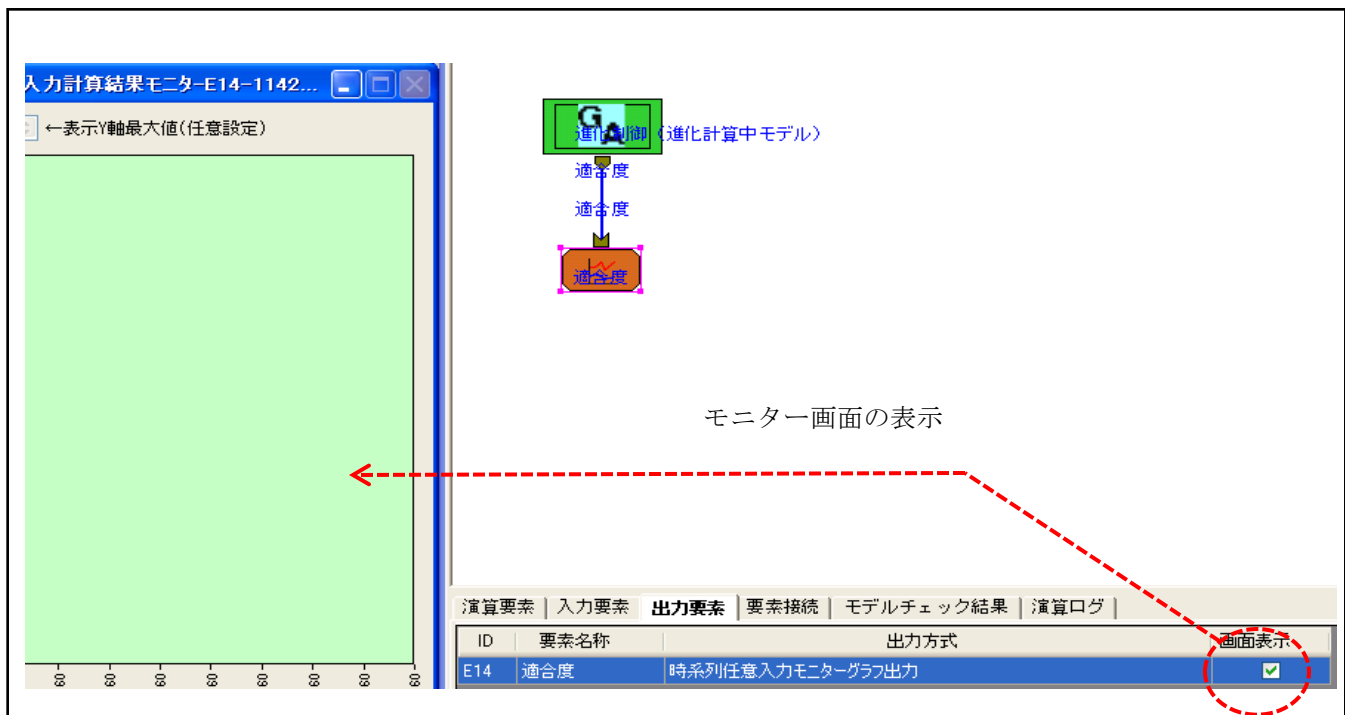


図4. 2 モニター画面の表示

④計算開始

プロジェクト管理画面上で、計算期間を1ヶ月程度にして 計算を開始します。

(注意)

進化計算には、時間がかかります。 マシン性能によりますが、
本サンプルモデルでは、CPUクロック数 3GHz程度のマシンを使用して
1世代の計算に 1分程度かかりました。

<補足>

本サンプルでは、観測値を、3パターン準備しています。

それらのデータは、フォルダー

CommonMPData¥SampleModelDevelop¥InputData¥GASample

下に置かれた下記ファイル

<モデルに入力する流入量データ>

EVTestFileIn1.csv、EVTestFileIn2.csv、EVTestFileIn3.csv

<比較用水位データ>

EVTestFileRef1.csv、EVTestFileRef2.csv、EVTestFileRef3.csv

内に 記述されています。

尚、このデータをプログラム内で管理する情報として、GAPatternDefinition.xml 内に、観測期間等が記述されています。

4. 2 進化計算の結果

図4. 1は、個体数50個で 約450世代進化させた時の、各世代でのスーパーエリートの適合度の推移を モニター画面で表示したものです。 また、各世代のスーパーエリートの遺伝情報は、化石として ファイルに出力しています。

＜化石の出力先＞

¥CommonMPData¥SampleModelDevelop¥OutputData¥GASample¥Fossil

＜化石ファイル名＞

SuperElite.datG + 世代数

図から、世代と共に、適合度が次第に増加している事がわかります。
尚、モニター画面の横軸は、表示は時間となっていますが、本当は、世代数を表しています。

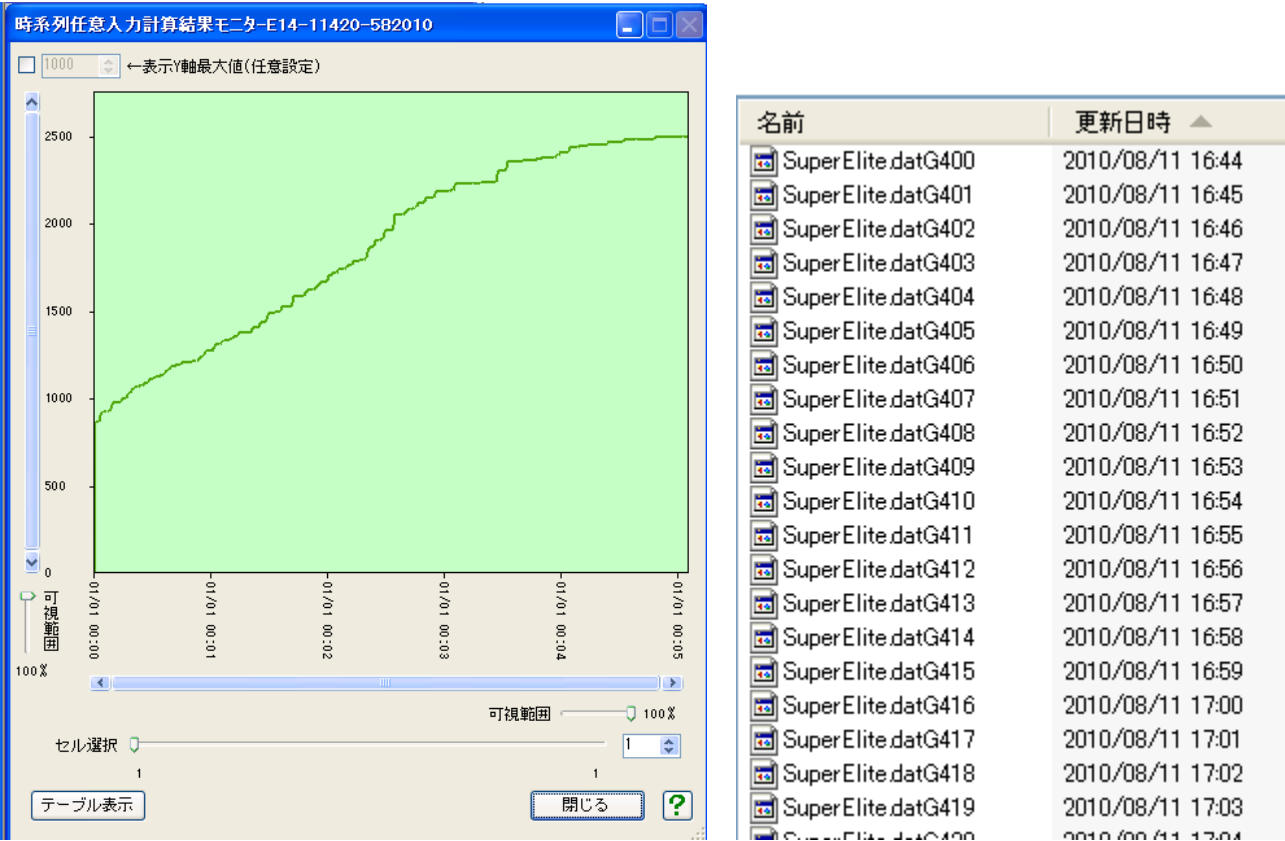


図4. 1 スーパーエリートの適合度の推移と化石情報

5. 進化したモデルによる予測計算

折角、モデルを進化させたので、次に進化したモデルを用いて、本当に粗度が最適化されているかを調べてみます。進化済みのモデルに対して、進化中には使用しなかった入力を与えて、その計算結果と 予め 観測していた 観測値を比較します。

尚、ここでの観測値とは、「正解モデル」の出力を示しています。

5. 1 進化したモデルによる予測実行

進化済みのモデルの計算は下記によって行います。

①CommonMPの起動

CommonMP¥Source¥HYMCO¥OptionImpl¥ModelDeveloperExpressEditionフォルダー下の TestModelDeveloperMainExp.sln を ダブルクリックして、Microsoft社のVisual Studio C# から ディバッグモードで CommonMPを立ち上げます。

②サンプルプロジェクトを開く

メニュー「プロジェクト」-「開く」から下記プロジェクトを開きます。

<プロジェクトグループ> SampleModelDevelop

<プロジェクト名> GASample_Cal.cmprj

(構造定義ファイルは¥SampleModelDevelop¥SCF¥GASample¥EV_Cal.xmlに定義されています。)

すると、図5. 1のようなモデルが生成されます。

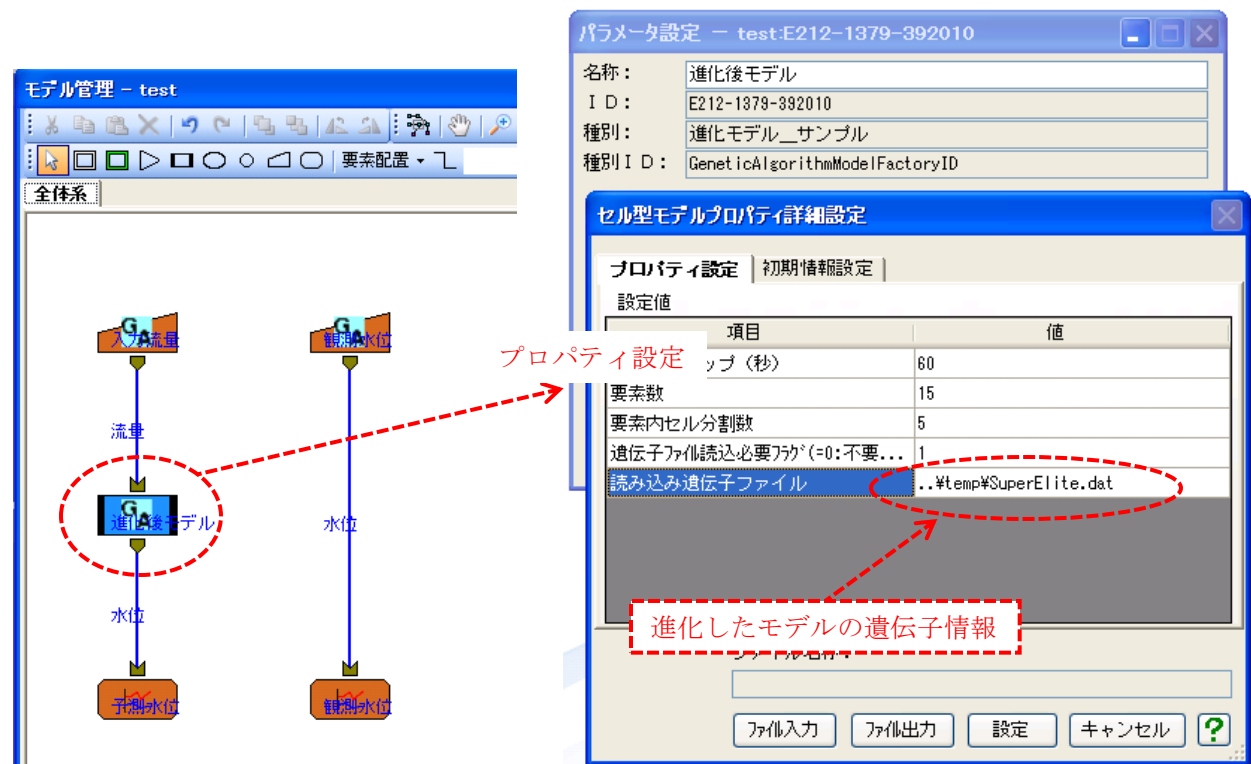


図 5. 1 最適化確認用の動作用モデル

画面の右側が、評価用のモデルで、入力に流量を与えると、計算した水位が出力され、モニター画面に表示されます。

画面の左側は、比較用に 予め準備した水位データがそのままモニター画面上に表示されます。

＜入力データ＞

CommonMPData¥SampleModelDevelop¥InputData¥GASample¥ EVTestFileIn9.csv

＜比較用水位データ＞

CommonMPData¥SampleModelDevelop¥InputData¥GASample¥ EVTestFileRef9.csv

③進化した遺伝子情報の設定

図5. 1のモデルのプロパティ画面を開き、図に示すように 化石として残された遺伝子情報を記述したファイル名を設定します。

④モデル実行

評価用のデータの期間は、2010/7/1～7/8 なので、図5. 2の様に 計算期間を それに合わせて設定し、計算を開始します。

⑤計算結果

モニター画面を表示すると、図5. 3の様に 計算された結果と、比較データが画面に表示されます。（モニター画面には、河道の縦断水位が表示されます）

＜補足＞

進化中とは異なり、進化済みのモデルを使用する場合には、演算の時刻は 一般のモデルと同じように過去から未来に様に流れます。

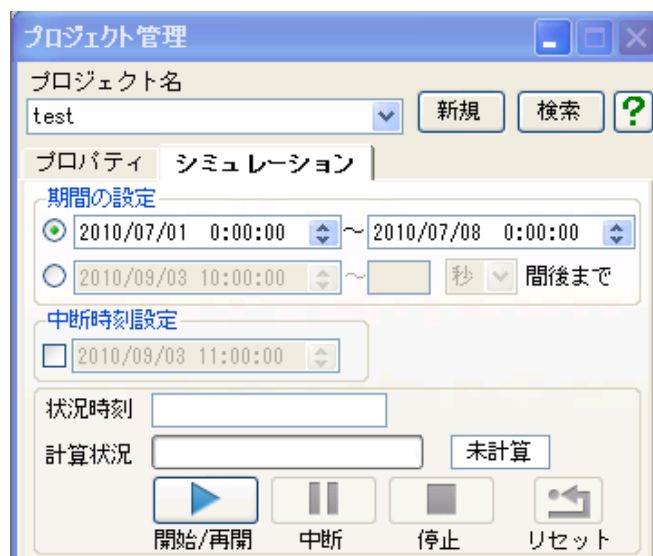


図 5. 2 計算開始（期間の設定）

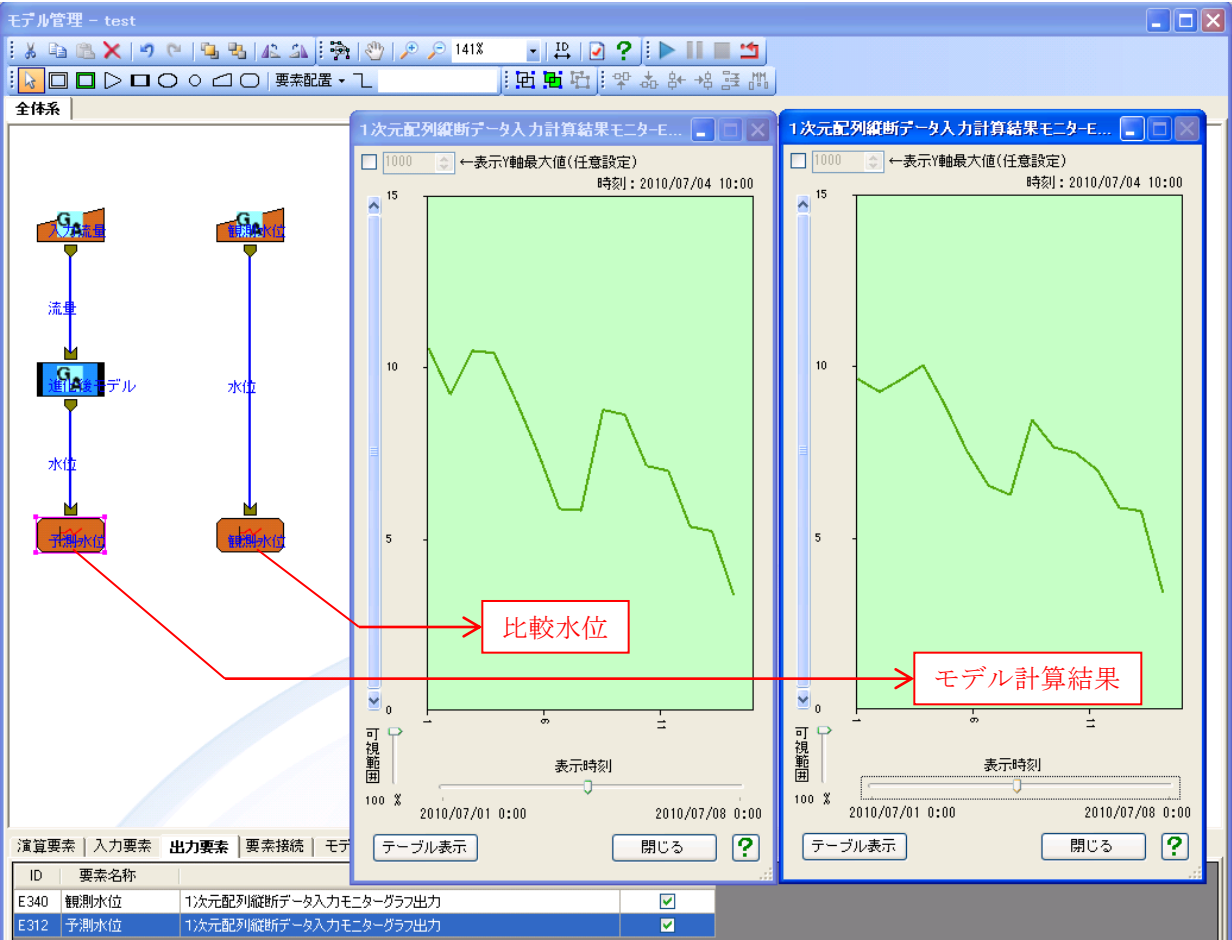


図 5. 3 計算結果

5. 2 遺伝的アルゴリズムによる進化の様子

図5. 4は、各世代の遺伝子情報を元に構築されたモデルが計算した河道縦断水位と、真の水位(正解モデルの計算値)を示したものです。 世代を経るに従って、計算値が、真の値に近づいている事がわかります。

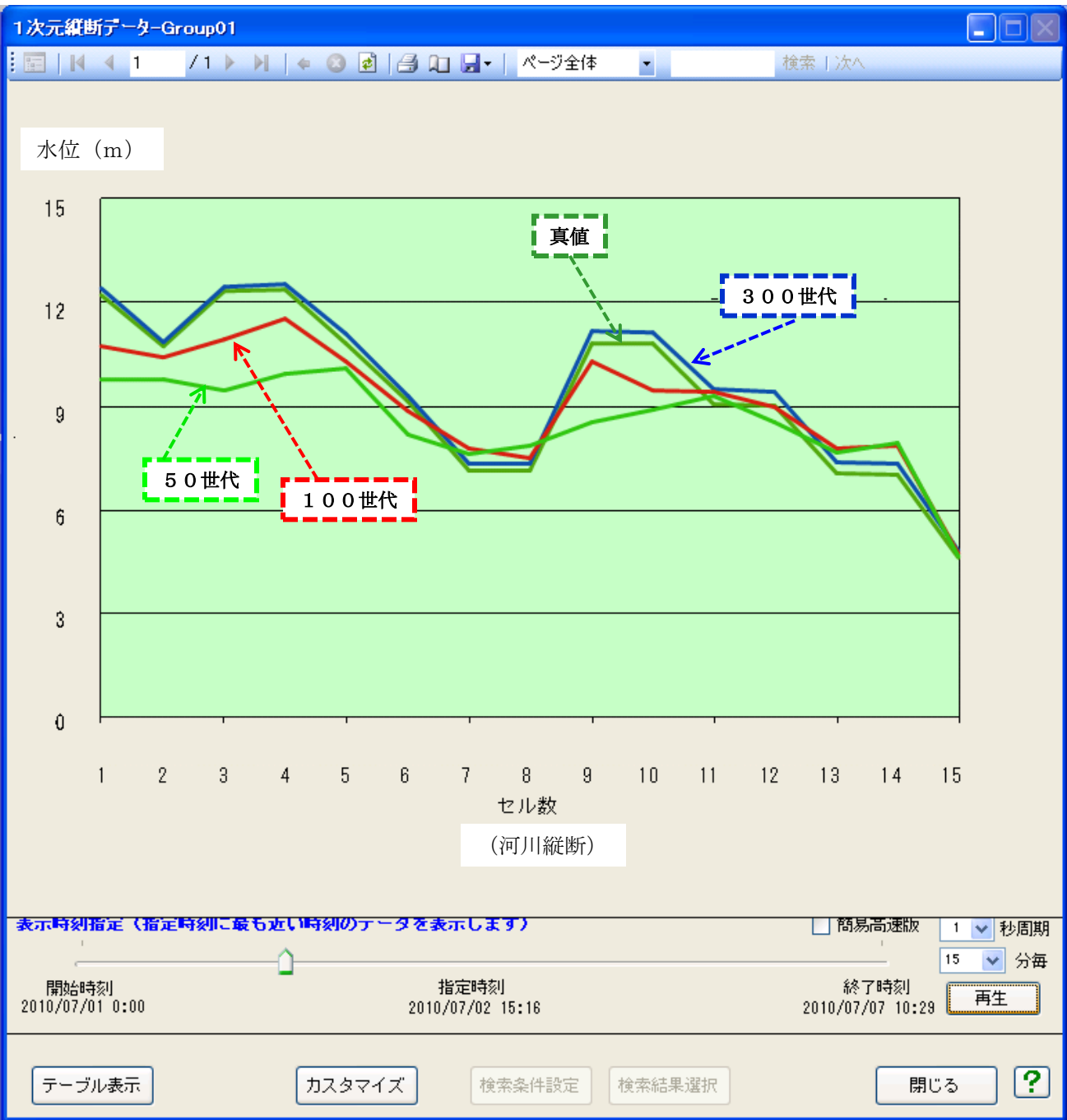


図5. 4 進化の状況

6. 終わりに

CommonMP自身は、その上で動作するモデルを規制するものではありません。その事を示す為に、本サンプルでは、CommonMP Ver1.0でリリースされた機能を用いて、繰り返し計算の例として 遺伝的アルゴリズムを用いた計算の例を示しました。しかしながら 本サンプルの作成によってもわかるように CommonMP Ver1.0では 未だ機能的に不十分な部分があり、動作させたいモデルを構築するに困難を来たす事もあります。また、画面としても 制約が多くあります。

ユーザーからの多くの御要求により、機能が充実され、色々な局面でも使用して頂ける事を目指します。