

収束演算 サンプル

1. 概要

本サンプルでは、水理・水文系とは無関係な例で 収束演算サンプルを作成しました。
収束演算においては、その収束制御用モデルも内部のモデルに依存する為、ここでは、収束演算を行う際の一例として、参照下さい。

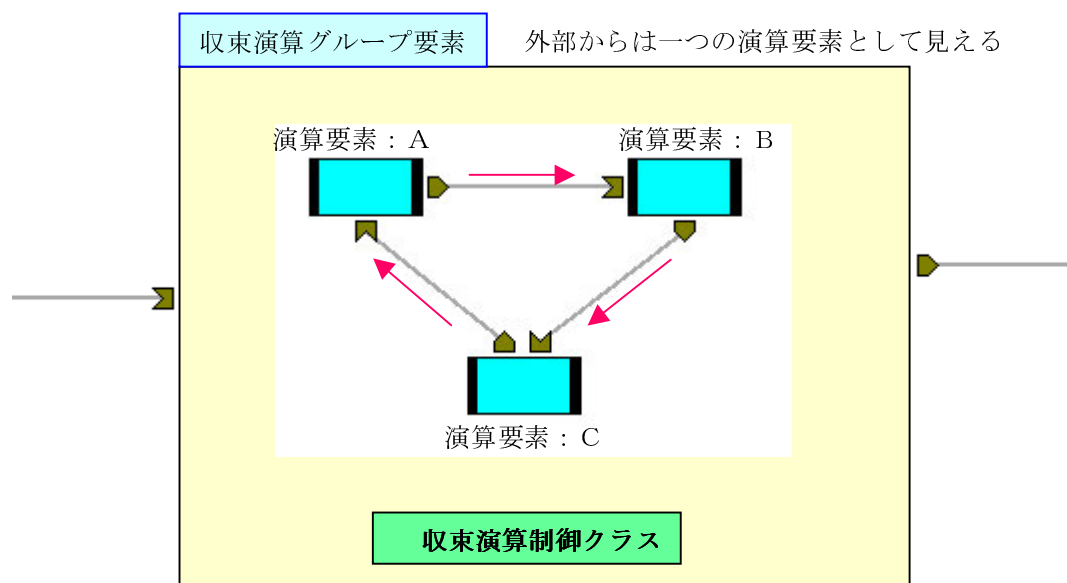


図1. 1 収束演算概念図

「収束演算」とは、例えば、図1. 1の様に 内部に 要素:A, B, Cがお互いに影響を与えながら、演算を繰り返し、ある状態に収束した時、その結果が、演算結果であると見なします。

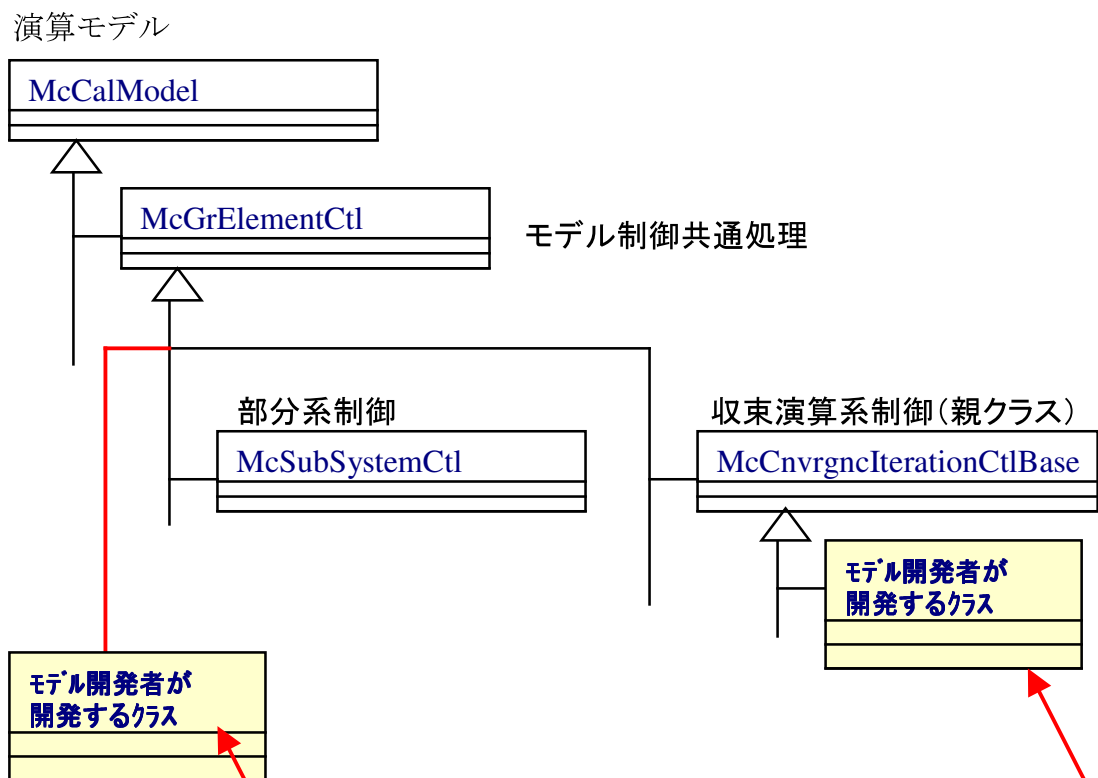
この計算においては、各要素が繰り返し計算を行い、各要素の状態が収束したか否かを判断する等、CommonMPが提供する「全体系(あるいは部分系)での演算系制御」は、対応出来ない部分があります。そこで、収束演算を行うモデルは、一つの収束演算系として一塊に纏め、外からは一つの演算要素として見なし、その内部で 収束演算計算を行います。

これを「収束演算グループ要素」と呼びます。「収束演算グループ要素」内部には、実際の計算を行う「演算要素モデル」が接続されています。また、「収束演算グループ要素」を制御する処理モデルも一つの「演算要素モデル」と見なすことができます。

収束演算は、扱うモデル等により、収束条件、制御方法が異なる為、CommonMPで 汎用的収束演算処理を作成することは現実的ではありません。したがって、モデル開発者が、自開発モデルに応じた収束演算制御を開発する必要があります。

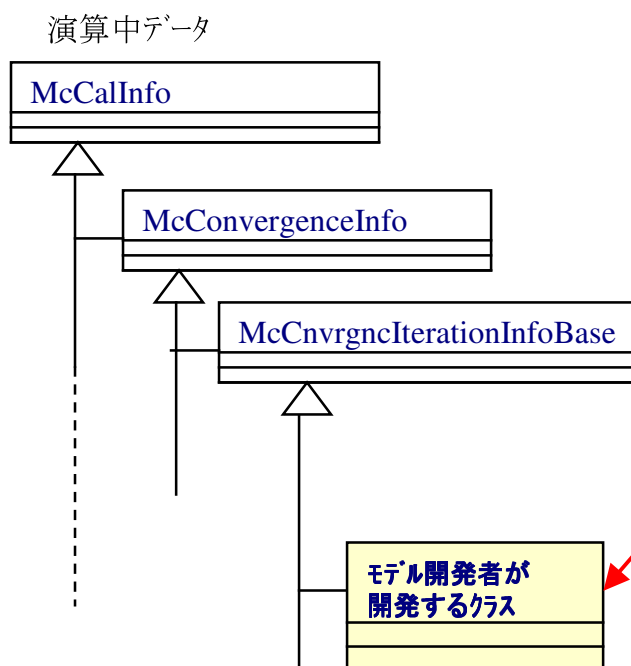
そこで、CommonMPでは、収束演算も扱えるフレームワークとして 図1. 2に示すように 収束演算制御用の「収束演算系制御基底クラス([McCnvrngIterationCtlBase](#))」及び、それに対応する「収束演算用演算中データクラス基底クラス([McCnvrngIterationInfoBase](#))」を準備します。

モデル開発者は、上記に示した「基底クラス」からクラスを派生して、収束演算制御等のクラスの製作を行います。



<注意>

フレームワークとして収束演算関連の親クラスを準備しているが、現CommonMP フレームワークが想定していない制御が必要な場合には、上のクラスから派生させても良い。



対で作成

図1. 2 収束演算制御関連クラス

2. 収束演算制御

2. 1 収束演算グループ要素

収束演算は、収束演算グループ要素内部で 収束計算を行います。

収束演算を制御する場合、内部の要素の状態が 如何なる状態にあるかを認識しておく必要があります。したがって、収束演算グループ要素内に含まれている 各々の演算要素モデルの演算中データ(各演算要素の **McCalInfo**)、及び各モデル間の伝送データ(**McTranInfo**)等を 保持しています。これらの情報は、収束演算系制御基底クラス(**McCnvrncIterationCtlBase**)が使用する収束演算用演算中データ基底クラス(**McCnvrncIterationInfoBase**) にメンバー変数として持ちます。これらのメンバー変数は、**protected**で宣言されている為 派生クラス側からも参照できます。但し、クラス自身にそれらの Getter,Setter メソッドを設けている為、それらのメソッドを使用することを奨励します。

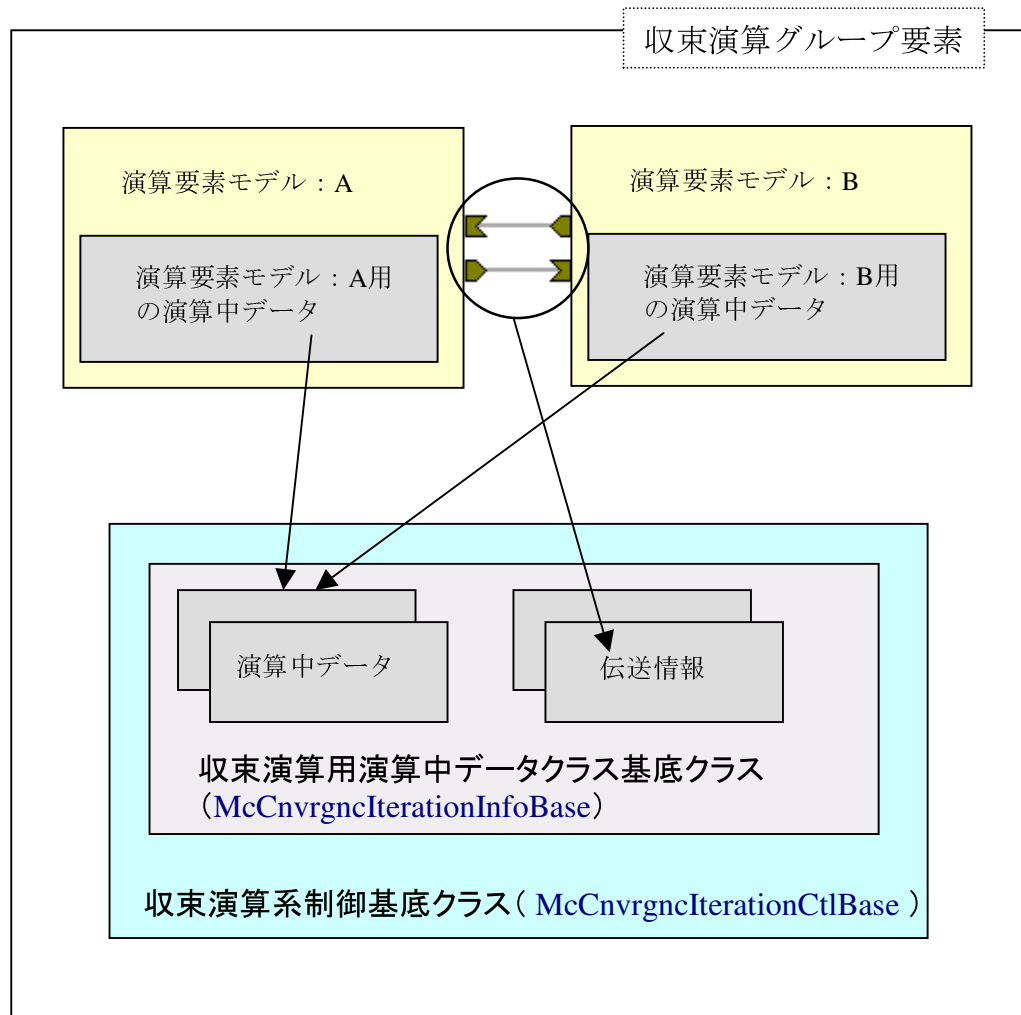


図2. 1 収束演算関連モデル構造概念図

2. 2 収束演算制御

収束演算グループ要素は、外部からは、一つの演算要素として見えます。しかしグループ内では、通常のグループ化要素とは異なる処理を行っています。

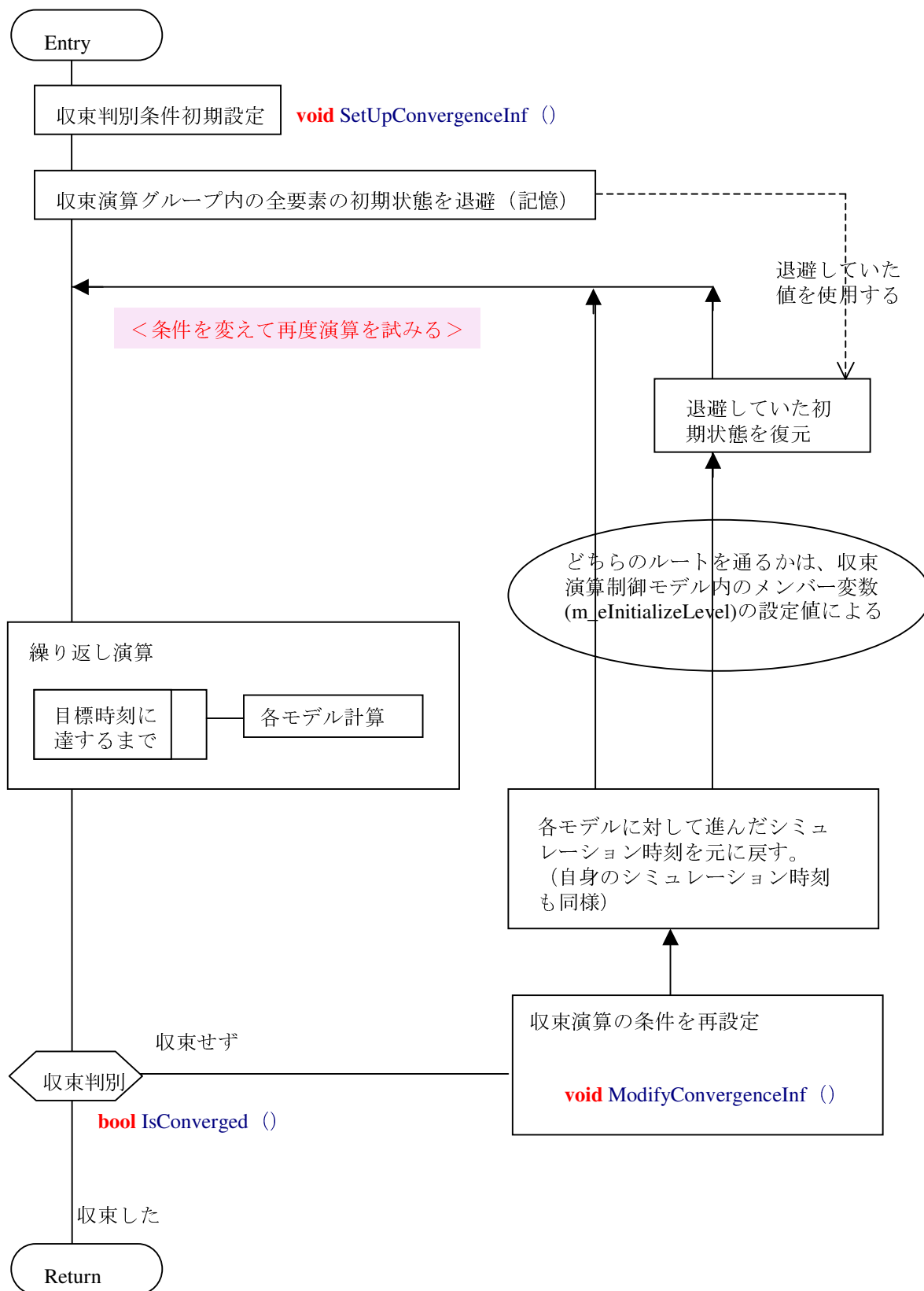


図 2. 2 収束演算グループ内の処理

図2. 2に示すように、収束演算では、

- ①演算開始前に、初期条件を退避して記憶しておきます。
- ②ある条件を設定し、特定時刻(或いは回数)まで モデル内の要素の演算を繰り返します。
- ③収束条件を満足していたならば演算を終了します。
- ④収束していなければ、ある条件を変更し その他の情報は退避していた初期条件によって 演算要素を初期化し、再度計算を行います(時刻も戻します)。
- ⑤以降、収束するまで ③、④を繰り返します。
(永遠に収束しない場合に備えて、最大繰り返し数は設定しておきます)

モデル開発者が作成する 収束演算系制御クラス([McCnvrncIterationCtlBase](#)派生)
で、収束演算として特別に 実装すべき主なメソッドは 下記に示す通りです。

- 1) 収束判別条件初期設定

protected override void SetUpConvergenceInf(

- 2) 収束判別

protected override bool IsConverged(

- 3) 収束演算の条件を再設定

protected override void ModifyConvergenceInf

その他、実装すべきメソッドは、親クラス側で **abstract** 宣言してある為、
実装時にコンパイルエラーとして認識できます。

<補足>

収束演算グループ要素内の要素モデルが、外部のモデルと接続される場合(図2. 3)
グループの内外は、中継端子を経由して接続されている為、収束演算中の演算要素
モデルが出力した情報が全て、外部へ伝送されるわけではありません。中継端子は、
収束演算グループ要素の収束が終了した後、その収束結果だけを、外部に対して出力
します。

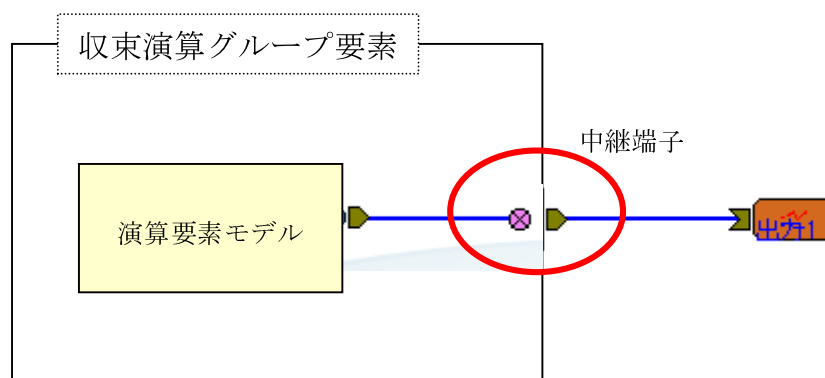


図2. 3 中継端子

3. 収束サンプル問題

わざわざCommonMPを使用して計算するまでもありませんが、収束演算の簡単な例として以下の様な問題を考えます。

<問題>

変数X, Y が存在するとして、

$$aX + bY = C \quad a, b, C \text{は定数} \quad \text{①式}$$

$$\alpha = XY \quad \text{②式}$$

①式を満足させながら ②式で求めた α が最大となる様に X, Yの値を求める。

此処に、 $a=100$ 、 $b=100$ 、 $C=1000$ とする。

上記問題を収束演算によって解くことを考えます。

1. 先ず、初期状態として $X=0.5$ の時を考えると、

$$Y = (1000 - 100 \times 0.5) / 100 = 9.5$$

この時 $\alpha = 0.5 \times 9.5 = 4.75$

2. 次のステップで

$\delta X = 1.0$ として $X = X + \delta X$ とし 同様の計算を行います。

同様に計算すると

$$X = 1.5, Y = 8.5, \alpha = 12.75$$

3. 次第にX値を大きくすると

$$X = 4.5, Y = 5.5, \alpha = 24.75$$

$$X = 5.5, Y = 4.5, \alpha = 24.75$$

4. ここで、増加していた値が同じ値となった為、極大値はその間にあると推定して

$$X = X - \delta X, \quad \delta X = \delta X / 10 \quad \text{として}$$

また、同じ計算を行う。

$$X = 4.51, Y = 5.49, \alpha = 24.7599$$

5. このようにして 極大値を挟んだ時に δX 値を次第に小さくしていき、

$\delta X < \varepsilon$ より小さくなったとき X, Yは収束したと判断する。

$$X = Y = 5.0 \text{となるはずである。}$$

4. サンプルモデル接続例

CommonMPの内部要素モデルとして、入力値Xにより 出力 $Y = (C - aX) / b$ を計算しそのY値を出力する演算要素モデルを 作成します。

収束演算テスト用要素モデル([McCnvElmSampleTestModel](#)) がその実装モデルとなります。

また、収束演算制御モデルとしては、3章で述べた α 値がと δX 値を判別して収束制御を行うクラスを作成します。 収束演算制御のコーディング例クラス([McSampleCnvrgrncIteration](#))がその実装モデルとなります。

収束演算の例の構造定義ファイルを ホームディレクトリ¥temp¥下に

ConvModelSample.xml として用意しております。

それを読み込んで、テストして下さい。(モデルの各メソッドにブレークポイントを置くと動作が理解できると思います)

収束演算のサンプルとして準備した ConvModelSample.xml を読み込んだ時のモデルを図4. 1に示します。

「全体系」タグで 緑色の箱型要素に コーディングサンプル:収束演算制御の例 のモデルが割り付けられています。 前記モデル(緑色)をセレクトし、ポップアップメニューから「グループ内部表示」―「開く」を選択すると、新しいタグが表示され、グループ内部の要素モデルの配置が確認できます。

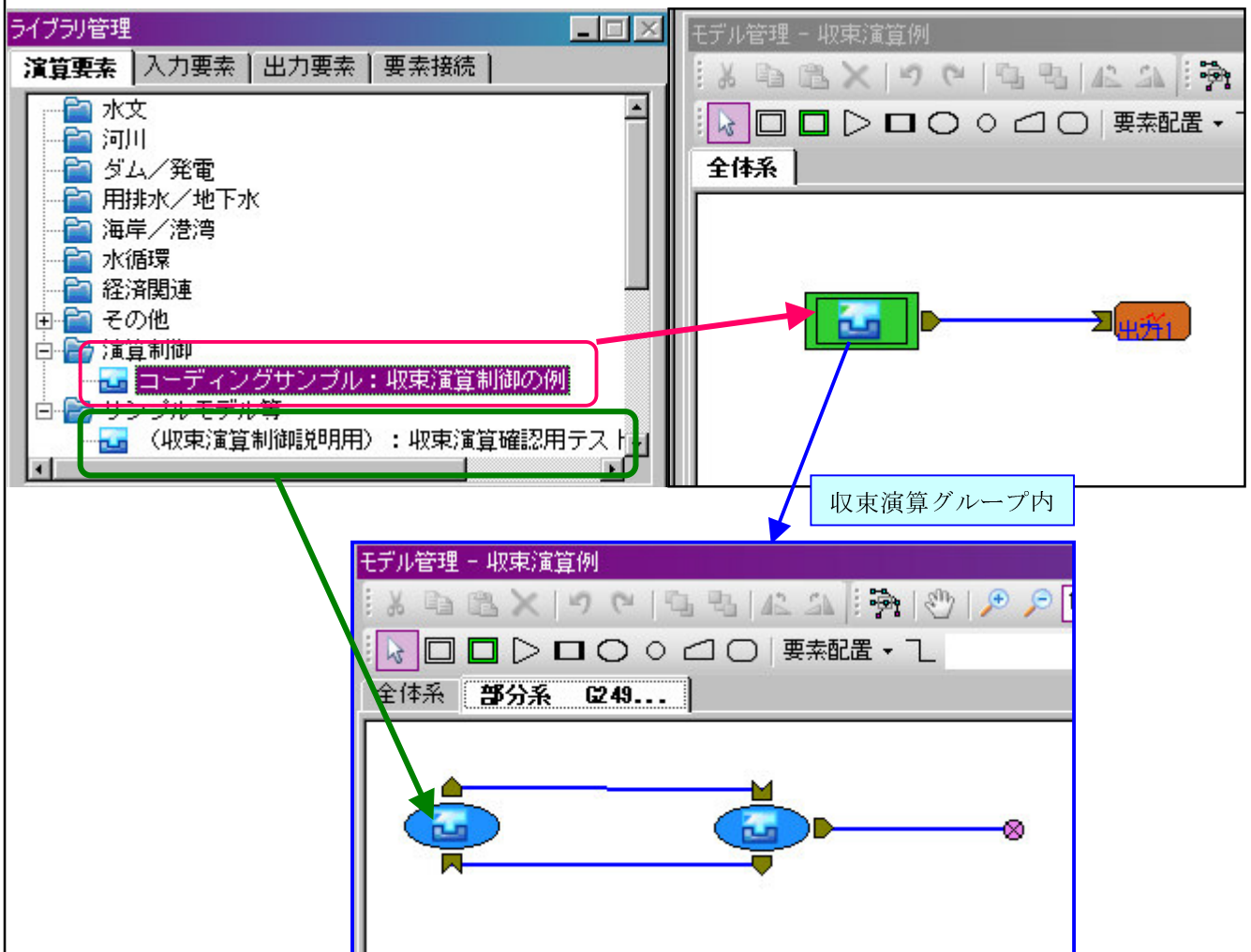


図 4. 1 収束演算の構造

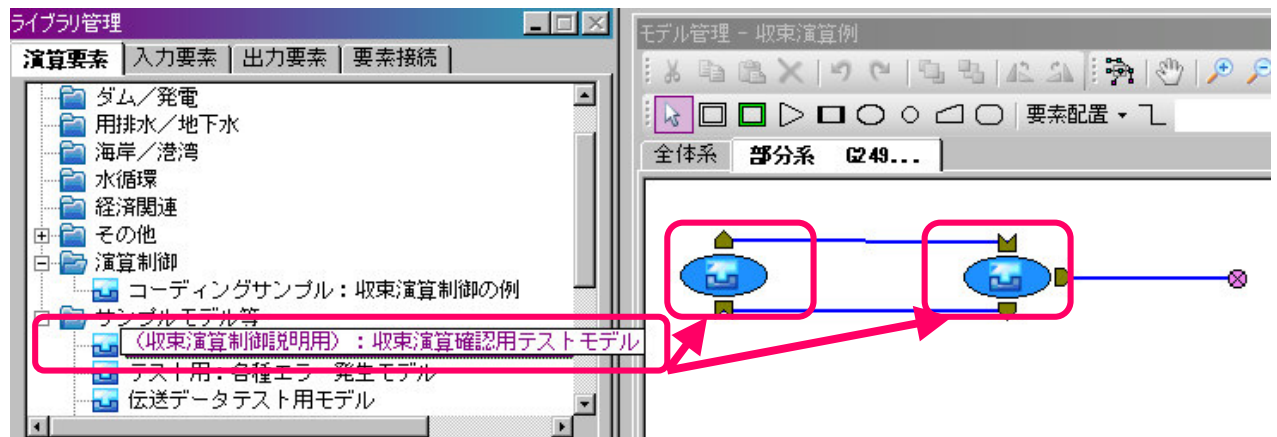


図 4. 2 収束テストモデルの接続

図4. 2 において どちらか一方要素が X 、もう一方が Y の値を計算します。

本サンプル問題では、演算の時刻は、意味をなしません。しかしながら、実際の水理・水文モデルにおけるモデルでは、時刻制御が重要となりますので、演算要素モデル間の接続は、敢えて、時系列情報(ポイント時系列データクラス)を用いております。

モニター画面を表示し、動作させた場合には、必ず収束値: $X=5$ の値が表示されます。(図4. 3参照)

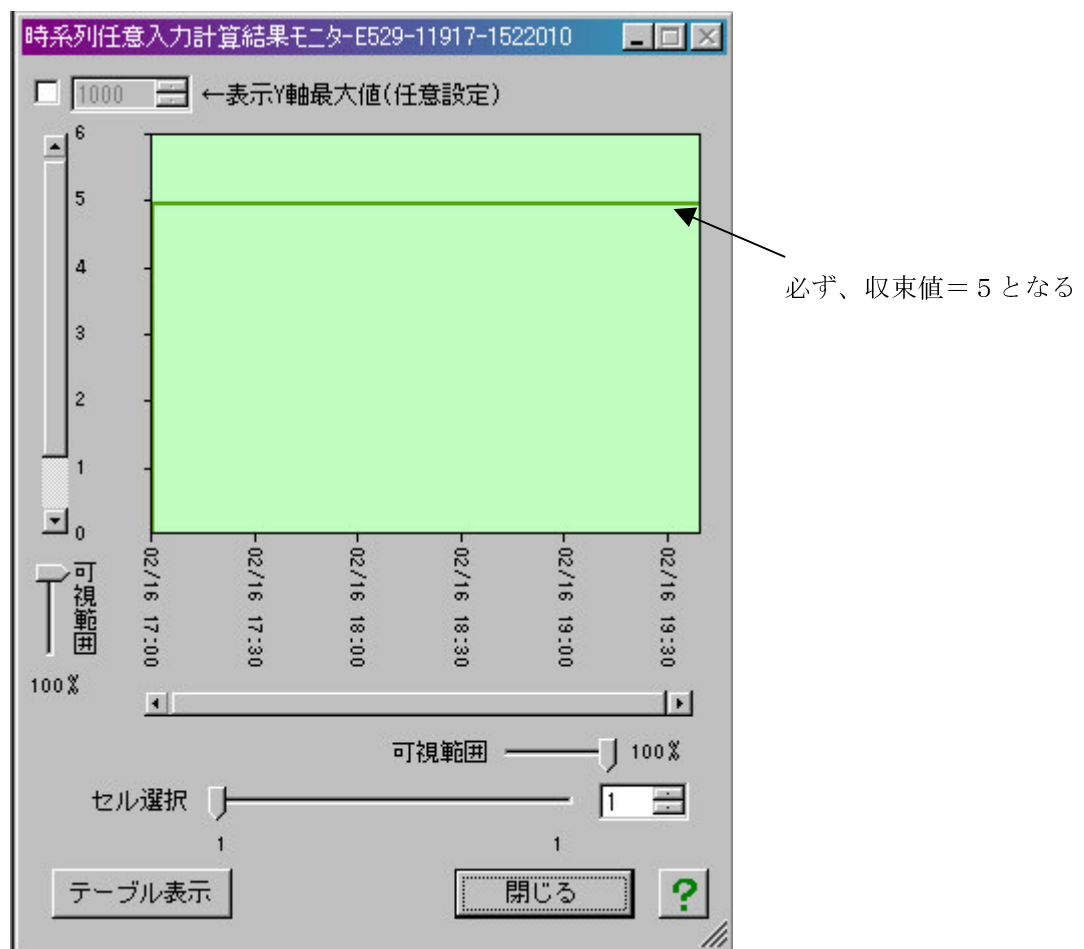


図 4. 3 演算結果出力