Nuorium

ユーザガイド
V18

株式会社NTTデータ数理システム

2016年3月

MATHEMATICAL SYSTEMS INCARNATION

目　次

第 1 章　はじめに 1

1.1 Nuorium とは . . . 1

1.1.1 Nuorium の起動方法 . . . 2

1.1.2 ウィンドウ設定 . . . 2

1.2 基本的な使い方 . . . 2

1.2.1 エディタ . . . 3

1.2.2 ログ . . . 4

1.2.3 結果一覧 . . . 4

1.3 モデルとデータの連係 . . . 5

第 2 章　エディタについて 7

2.1 エディタと SIMPLE . . . 7

2.2 エディタの主な機能 . . . 7

2.3 エディタ設定 . . . 12

2.3.1 レイアウト . . . 13

2.3.2 ガター . . . 14

2.3.3 タブ . . . 15

2.3.4 スクロールバー . . . 15

2.3.5 視覚効果 . . . 15

2.3.6 自動補完 . . . 16

2.3.7 折り返し . . . 16

第 3 章　ログおよびビルドと実行について 17

3.1 ログ . . . 17

3.1.1 ログの表示領域 . . . 17

3.1.2 ログの内容 . . . 18

3.2 ビルド . . . 20

3.2.1 実行ファイルの生成 . . . 20

3.2.2 使用できない name 属性について . . . 20

3.3 ビルド & 実行オプション . . . 20

3.3.1 ビルド . . . 21

3.3.2 実行 . . . 22

第 4 章 詳細結果について **23**
4.1 詳細結果画面 . . . 23
4.1.1 詳細結果画面の表示 . . . 23
4.2 詳細結果の表示形式 . . . 24
4.2.1 共通する表示形式 . . . 25
4.2.2 目的関数 . . . 26
4.2.3 パラメータ . . . 26
4.2.4 変数 . . . 27
4.2.5 式 . . . 27
4.3 詳細結果画面上の操作 . . . 27
4.3.1 詳細結果画面の自動更新 . . . 30

第 5 章 データ入出力について **31**
5.1 入力データ . . . 31
5.1.1 csv 形式データ . . . 31
5.1.2 dat 形式データ . . . 32
5.1.3 Numerical Optimizer パラメータ . . . 32
5.1.4 VAP 専用の表形式データ . . . 32
5.2 出力データ . . . 33

第 6 章 ファイル入出力について **35**
6.1 ファイル形式 . . . 35
6.2 ファイルの保存について . . . 36
6.3 エクスポート . . . 36
6.3.1 エクスポートの方法 . . . 36
6.3.2 エクスポート先の階層構造 . . . 37

付録 A 正規表現 **39**
A.1 メタ文字 . . . 39
A.2 エスケープ . . . 40
A.3 例文 . . . 40

索 引 **43**

第 1 章 はじめに

Nuorium とは何か，起動方法，そして基本的な使い方を説明します．Nuorium のひととおりの操作方法を理解することができます．

1.1 Nuorium とは

Nuorium は数理計画問題を計算機上で解くために特別に設計された Numerical Optimizer のための GUI 環境です．内部エンジン“Numerical Optimizer”は，次の 2 つからなる汎用数理計画法パッケージです．

- 数理計画のためのモデリング言語 SIMPLE
- 多様な求解アルゴリズム

Nuorium を使用することで，誰もが Numerical Optimizer の能力を十分に引き出し，数理計画問題の定式化・求解・分析といった一連の流れをスムーズにこなすことができます．Nuorium の代表的な特徴を次に挙げます．

- Nuorium は使いやすさや見やすさのための統一性を意識したデザインが施されています．加えてプログラミングに適した視認性・判読性・可読性が高いフォントを採用しているため，ストレスなく SIMPLE を記述することができます．
- Nuorium なら，モデルの編集環境をいつでもお気に入りの環境へと，簡単にかつ柔軟にカスタマイズすることができます．この他，モダンなエディタに期待する様々な機能を搭載し，基本的な部分から積極的かつ強力に開発を支援します．
- モデルのビルドと実行は，異なる画面を行き来することなく，また特別な環境を必要とすることもなく，Nuorium で完結して行うことができます．もしモデル記述に誤りがあった場合にも，モデル記述のどこで発生しているのか，Nuorium はその手がかりを的確にわかりやすく教えてくれます．
- Nuorium を使うことでモデルの構造を明確に捉え，たとえ複雑なモデルであってもモデルの全容を一望することが可能になります．そして詳細に見たい結果のみを容易に表示でき，膨大な結果を前にしても問題解決に専念することができます．この他，Excel へのコピーも簡単に行うことができます．

以上のように Numerical Optimizer のための場所にふさわしいよう，実に多くの工夫が Nuorium に施されており，すべての人が Numerical Optimizer のポテンシャルを体験することができます．

そう！ Nuorium とは Numerical Optimizer のための場所なのです！

1.1.1 Nuorium の起動方法

Nuorium は Visual Analytics Platform から起動します．Numerical Optimizer モデルアイコンのコンテキストメニューにある「編集」を選択することで起動できます．

1.1.2 ウィンドウ設定

Nuorium の起動時のウィンドウの表示サイズと表示位置の設定は，[メニューバー] → [設定] → [ウィンドウ] を選択することで開かれる設定ウィンドウから行います．設定は 3 種類あります．

- 終了時の設定で起動：終了時の表示サイズと表示位置を設定値として次回起動します．
- 既定値で起動：常に既定の値で次回起動します．
- 常に最大化で起動：常に最大化で次回起動します．

1.2 基本的な使い方

数理計画問題の記述から実行，結果の確認方法などの基本的な使用方法を以下に説明します．

Nuorium の画面構成は図 1.1 のとおりで，大別して次の 3 つの部分からなります．

1. エディタ
2. ログ
3. 結果一覧

これらのうち「エディタ」，「ログ」，「結果一覧」について，基本的な使い方に沿って説明します．

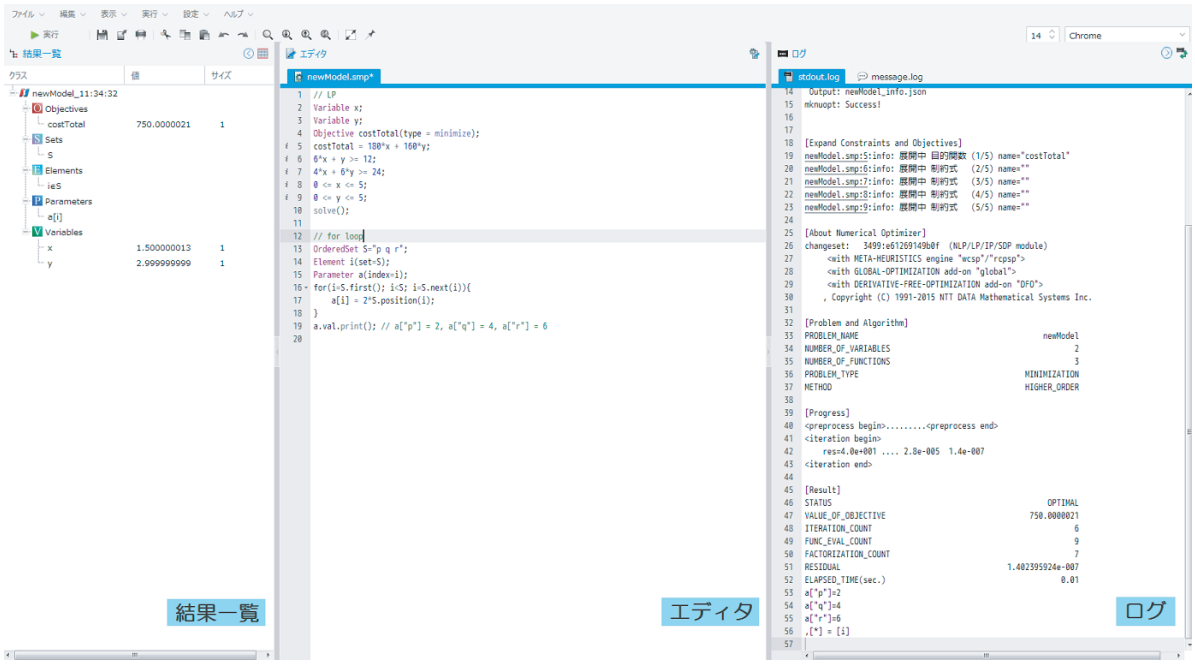


図 1.1 全体画面

1.2.1 エディタ

数理計画問題はモデリング言語 SIMPLE によって記述を行います．記述する領域は画面中央のエディタ領域（図 1.2）です．

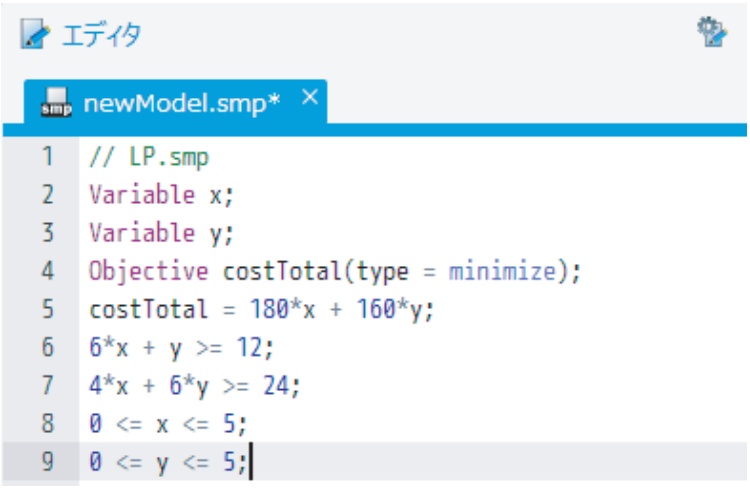


図 1.2 エディタ

例えば次の内容をエディタ領域に記述します．

```
// LP.smp
Variable x;
Variable y;
Objective costTotal(type = minimize);
costTotal = 180*x + 160*y;
6*x + y >= 12;
4*x + 6*y >= 24;
0 <= x <= 5;
0 <= y <= 5;
solve();
x.val.print(); // output to stdout
```

実行はツールバーの実行アイコンを押すか，[メニューバー] → [実行] → [実行] を選択してください．この他，右クリックで表示されるコンテキストメニュー（図 1.3）の実行を選択，もしくは F5 キーまたは Shift-Enter キーによるショートカットキーによっても実行することができます．実行を行うことで，記述された内容の数理計画問題を Numerical Optimizer が求解します．

図 1.3 コンテキストメニュー

実行は SIMPLE のビルドも兼ねています．既にビルドを行っていた場合にはビルドプロセスをスキップし，求解プロセスを実行します．ビルドのみを行うことも，メニューバーやコンテキストメニューなどからできます．なおビルドまたは実行中はファイルを編集できません．

1.2.2 ログ

Nuorium はビルドまたは実行途中にログを出力しユーザを支援します．ログには `stdout.log` または `message.log` があります．それらの概要は次のとおりです．

`stdout.log` にはビルドや実行が成功・失敗したときのログ情報や，SIMPLE 上で記述を行った `.val.print()` などの標準出力への命令の出力が，プレーンテキストで表示されます（図 1.4）．

`message.log` には Nuorium の画面に一時的にポップアップされたメッセージの履歴が表示されます（図 1.5）．

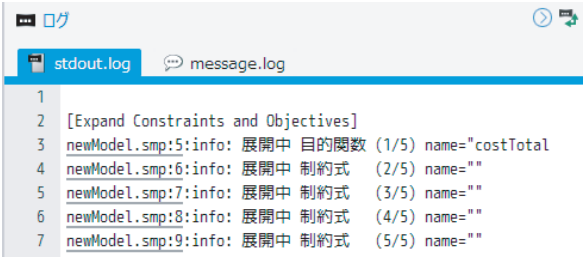


図 1.4 `stdout.log`

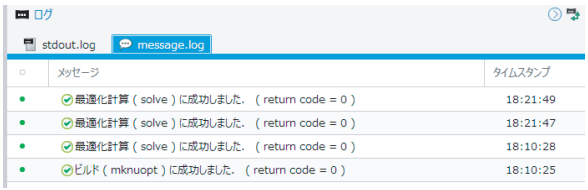


図 1.5 `message.log`

1.2.3 結果一覧

求解プロセスの実行結果の結果一覧は，求解後に画面左側に表示されます（図 1.6）．結果一覧では SIMPLE で記述したモデル中の目的関数，変数，パラメータなどを俯瞰することができます．

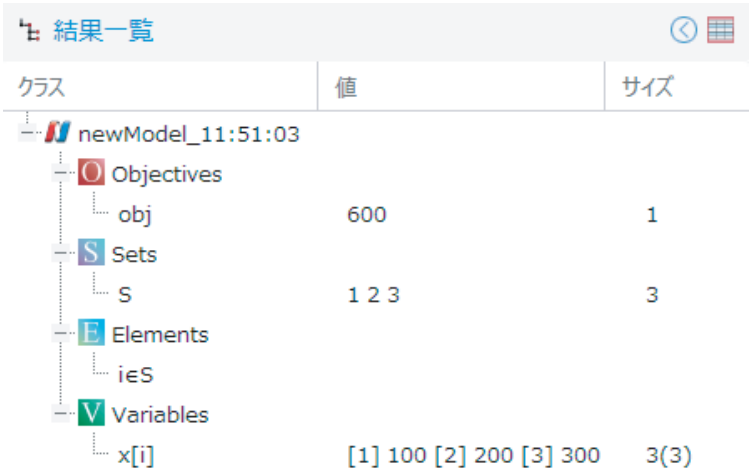


図 1.6　結果一覧

値の列では求解結果の一部情報を確認できます．添字付けされている場合は“[添字] 値”という形式で表示されます．例えば集合 $S = \{1, 2, 3\}$ の要素 i を添字としてもつ変数 x_i があったとします．

```
Set S = "1 2 3";
Element i(set=S);
Variable x(index=i);
```

そしてこのときの x[1]，x[2]，x[3] についての求解結果が，それぞれ 100，200，300 であったとすれば，値の列には次のように表示されます．

```
[1] 100 [2] 200 [3] 300
```

サイズの列は求解で必要とした数を確認できます．これにより問題の規模を捉えることができます．添字付けされている場合は“総数（各添字の総数）”という形式で表示されます．例えば集合 $I = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ の要素 i と，集合 $J = \{1, 2, 3, 4, 5\}$ の要素 j を添字としてもつ変数 $x_{i,j}$ があったとします．

```
Set I = "1 .. 10"; Element i(set=I);
Set J = "1 .. 5"; Element j(set=J);
Variable x(index=(i,j));
```

このとき変数 $x_{i,j}$ は全部で 10×5 で 50 変数あるので，サイズの列には次のように表示されます．

```
50(10,5)
```

結果一覧では見られない詳細な情報については，詳細結果画面から見ることができます．詳細結果は別途「4 詳細結果について」にて後述します．

1.3　モデルとデータの連係

モデルファイルとは別にデータファイル（dat または csv ファイル）が必要な場合，モデルとデー

タの連係は Visual Analytics Platform 上のフロー図（図 1.7）によって行います.

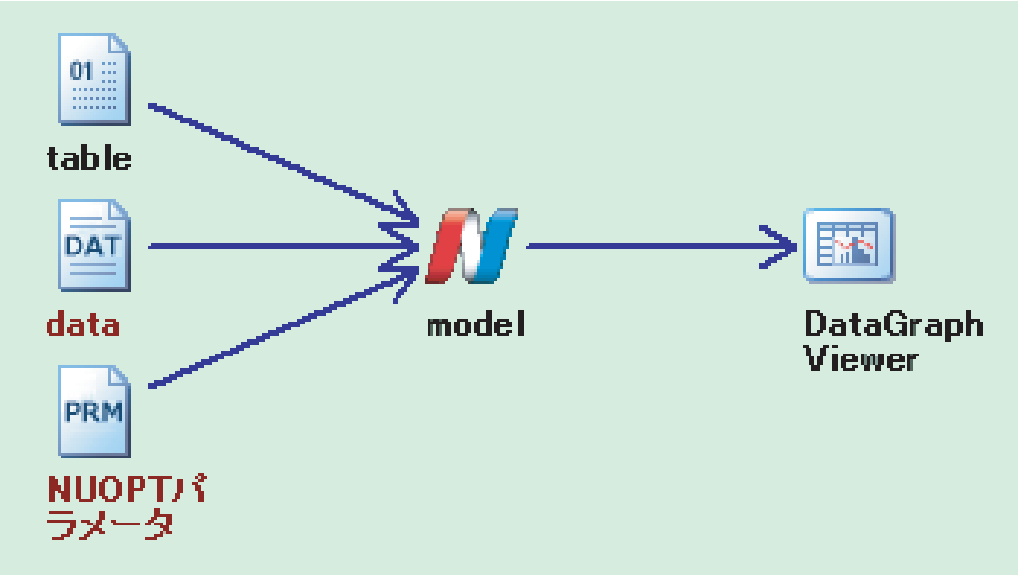


図 1.7 Visual Analytics Platform 上のフロー図

例えば次の内容をエディタ領域に記述したとします.

```
// LP.smp
Variable x, y;
Parameter a, b;
Objective costTotal(type = minimize);
costTotal = a*x + b*y;
6*x + y >= 12;
4*x + 6*y >= 24;
0 <= x <= 5;
0 <= y <= 5;
```

この場合には，a と b がモデルファイル LP.smp のデータであり，これらの具体的な値をデータファイルに記述して，Visual Analytics Platform 上のフロー図を用いてモデルと連係する必要があります.

モデルとデータの連係に関連したデータの入出力については，別途「5.1 データ入出力について」にて後述します.

第2章 エディタについて

エディタの主な機能とエディタ環境の設定方法について説明します．エディタに習熟することで効率的な開発をすることができます．

2.1 エディタとSIMPLE

数理計画問題のモデルを記述するSIMPLEはC++のクラスライブラリとして実装されているため，SIMPLEのシンタックスは基本的な部分でC/C++に準拠しており，C/C++と非常に似通った外観を持ちます．

このためSIMPLEの記述はいくつか確立されているC/C++のコーディングスタイルを引用したり，もしくはSIMPLE独自のコーディングスタイルに沿った記述を行うことが推奨されることがあります．

これらの多くはSIMPLEの記述の整形に関わる事柄ですが，Nuoriumが提供するエディタに備わっている編集機能を用いることで，容易に整形することができます．

エディタはまたテーマ（エディタの色調）に合わせてSIMPLEのキーワードをハイライトするとともに，自動補完機能も備わっているため，簡単にキーワードを思い出したり，キー入力ミスを事前に防ぐことができます．更に自動補完機能ではSIMPLE独自のイディオム（スニペット）も候補に挙げます．

この他，ビルドや実行に伴う編集ファイルへのフィードバックもアノテーションという形で行われます．そして正規表現による高度な検索や，複数のカーソルを展開した同時編集なども可能です．

上記に述べたSIMPLEを記述する上で役立つ機能が，Nuoriumのエディタとして予め付属しています．

2.2 エディタの主な機能

以下にエディタの主要な機能を説明します．これら以外の機能については [メニューバー] → [ヘルプ] → [キーバインド表] を参照してください．

自動インデント

コードブロックを書く際に，タブサイズで選択したサイズで，インデント調整（図2.1）を自動で行います．

```
16 for(i=S.first(); i<S; i=S.next(i)){
17     a[i] = 2*S.position(i);
18 }
```

図2.1　インデント調整

矩形選択

Alt キーを押しながらテキストをマウス選択すると，矩形上に選択（図 2.2）することができます．

```
1  // block selection: Alt-MouseSelection
2  Set I; Element i(set=I);
3  Set J; Element j(set=J);
4  Set K; Element k(set=K);
5  Variable z(index=k);
```

図 2.2　矩形選択

選択領域のコメント化

コメントアウト（またはアンコメント）を行いたい部分を選択し，Ctrl-/または Ctrl-Shift-/をキー入力することで，選択領域をコメントアウト（またはアンコメント）することができます（図 2.3）．

```
1  Variable x;
2  Variable y;
```

```
1  // Variable x;
2  // Variable y;
```

図 2.3　コメントアウト/アンコメント

集中モード

F10 キーを押すことで，エディタ領域のみを開いた状態（集中モード）にすることができます（図 2.4）．なお再び F10 キーを押すことで集中モードを解除することができます．

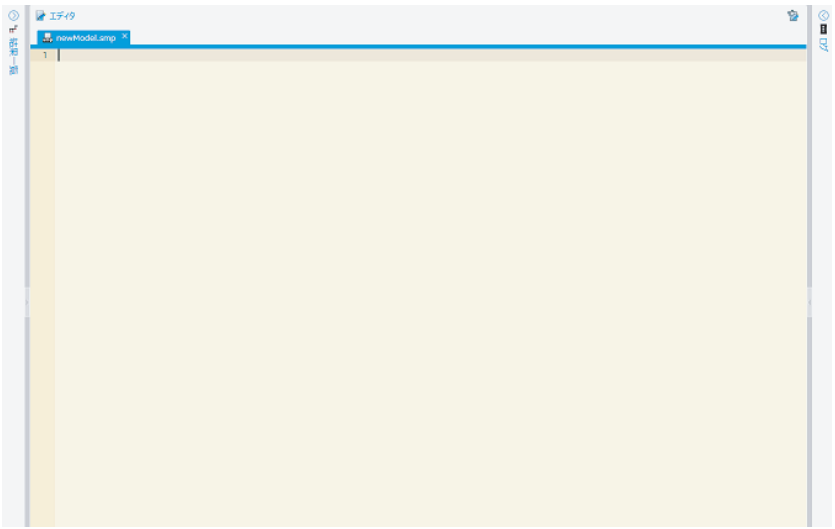


図 2.4　集中モード

自動補完

SIMPLE のコードを編集中に自動補完の候補を挙げます．カーソルキー（↑ ↓ →←）で移動し，タブキーを押すことで，補完候補を選択できます．Ctrl-Space をキー入力することで，カーソルがある場所の文字列を基に，補完候補を挙げることもできます（図 2.5）．補完候補を消して通常の入力モードに戻すには，Esc キーを入力してください．

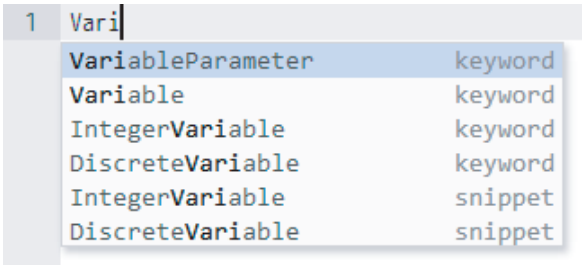


図 2.5　自動補完

補完候補には次の 3 種類があります.

- keyword：`Variable`，`Parameter` などの SIMPLE のキーワード
- snippet：SIMPLE で多用される定型文
- local：ユーザが編集中に入力した文字列

snippet による補完候補は，図 2.6 のように補足説明が同時に表示されます．補足説明には snippet の内容が記載されています.

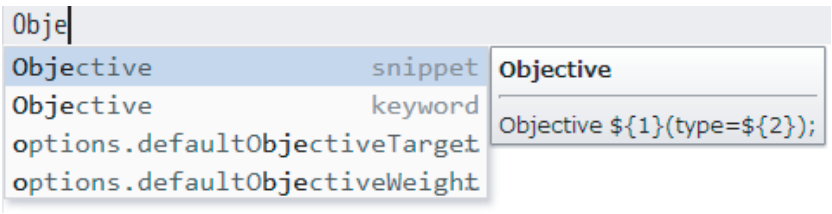


図 2.6　snippet による補完

例えば `Objective ${1}(type=${2});` という snippet は，ユーザによる`${1}`と`${2}`への入力を受け付けることを表しています．今の場合に実際にこの snippet を選択すると，次のように入力されます.

```
Objective |(type=);
```

`|`はカーソルを表していますが，snippet の入力とともに`|`がある位置にカーソルが遷移します．この位置は`${1}`の位置になっており，Tab キーを押すことで，次に示すようにカーソルを`${2}`の位置へ遷移することができます.

```
Objective (type=|);
```

そしてこの状態から Shift-Tab キーを入力すると，`${1}`の位置に遷移することができます.

アノテーション

SIMPLE のコードをビルドまたは実行した際，ログの `stdout.log` に編集を行っていた SIMPLE コードの該当行へのリンクが張られることがあります．このリンク情報はエディタの左端に位置する行番号が記されたガター領域にアノテーション（注釈）として表示されます（図 2.7）．このためアノテーションを表示させるには行番号を表示しておく必要があります.

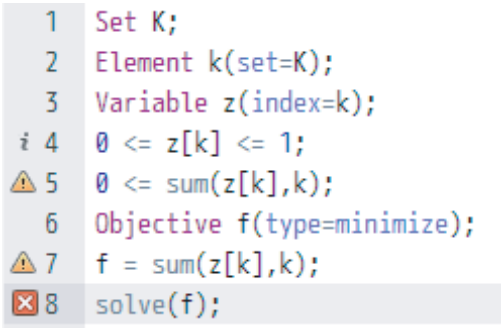


図 2.7　アノテーション

エディタ内では Alt-E または Alt-Shift-E をキー入力することで，各アノテーションへ現在行からジャンプし，その内容を表示することができます（図 2.8 参照）.

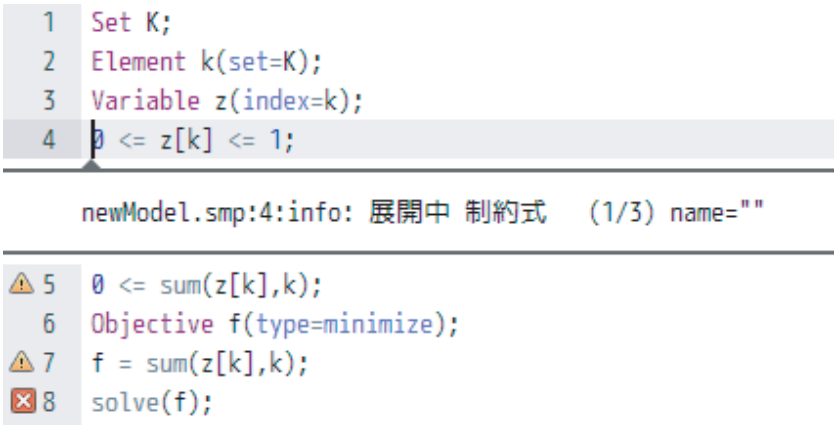


図 2.8　アノテーションへのジャンプ

正規表現検索

文字列検索を行う際に [.*] を選択（図 2.9）することで，正規表現を用いた検索ができます.

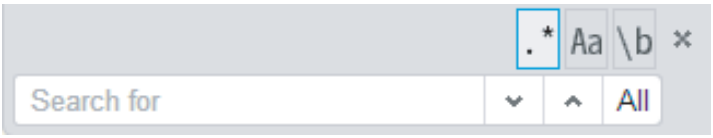


図 2.9　正規表現検索

正規表現で使用するメタ文字やその意味などについては，附録「A 正規表現」を参照してください.

あいまい検索

文字列検索を行う際に [Aa] を選択しなかった場合（図 2.10）には，大文字と小文字を区別しない検索（あいまい検索）ができます．例えば Set と検索した場合に，あいまい検索が有効な場合には，set も検索対象になります.

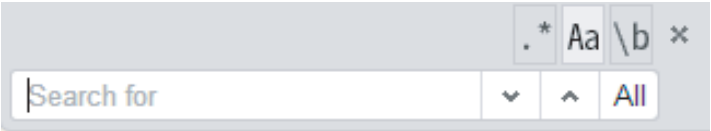


図 2.10　あいまい検索

全文一致検索

文字列検索を行う際に [\b] を選択（図 2.11）することで，文字列が全文一致したもののみ検索ができます．例えば `Variable` と検索した場合に，他に `IntegerVariable` があった場合でも，全文一致していない `IntegerVariable` は候補に挙がらず，`Variable` のみ検索対象になります．

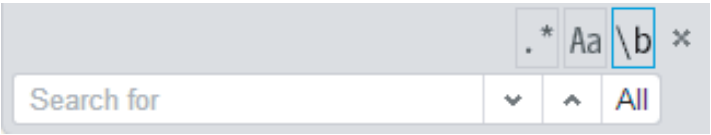


図 2.11　全文一致検索

検索結果の全選択

文字列検索を行って [All] を選択（図 2.12）することで，検索結果を全選択することができます．

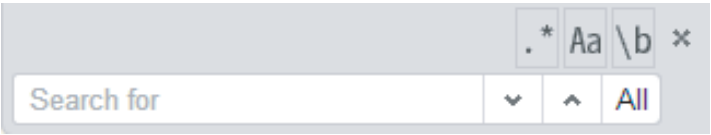


図 2.12　検索結果の全選択

選択領域の折り畳み

任意の領域を選択し，Alt-Shift-L または Ctrl-Shift-F1 をキー入力することで，選択領域にあるコードを折り畳むことができます（図 2.13）．例えば//によるコメント領域のみをすべて折り畳みたい場合には，正規表現検索で`//.*`と検索し [All] を押して，検索結果を全選択した後に，選択領域の折り畳みを実行すると，すべて折り畳むことができます．

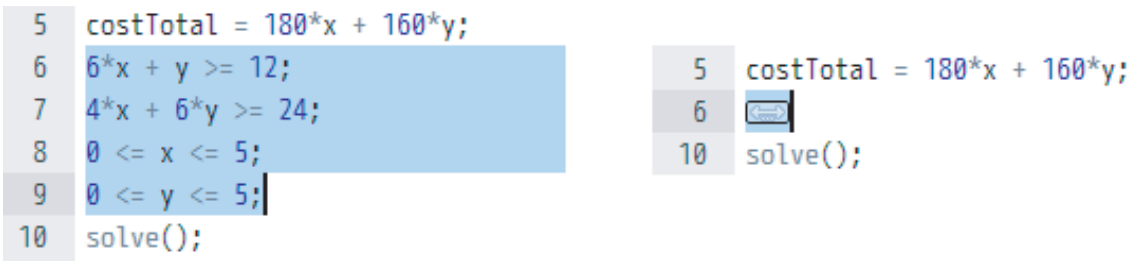


図 2.13　選択領域の折り畳み

なお Ctrl-I によって選択領域を反転することができるので，応用として//によるコメント領域をすべて選択し，Ctrl-I をキー入力して，選択領域を折り畳めば，//によるコメント領域以外をすべて折り畳むことができます．

折り畳みを解除するには，折り畳まっている部分をクリックしてください．もしくは Alt-Shift-0 をキー入力することで，すべての折り畳みを解除することができます．この場合の 0 キーとしては，テンキーの 0 キーを利用することができません．

多重カーソル

Ctrl-Alt-(カーソルキー↑↓→←) をキー入力することで，カーソルを多重化することができます．解除にはカーソルを重ねるか，または Esc キーを入力して解除できます．多重カーソルを使用する場面の例として次を挙げます．

複数行にわたって同一の文字列を入力をする場合

図 2.14 に示すように `Variable` が 3 行にわたってあったとします．ここで Ctrl-Alt-↓-↓などと↓を 2 回続けて入力してください．すると図 2.15 に示すように，カーソルが 3 重になります．このまま図 2.16 のように `Integer` と入力することで，連続変数の宣言を整数変数の宣言へと一度に置き換えることができます．

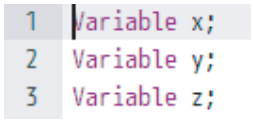


図 2.14

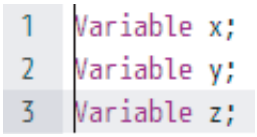


図 2.15

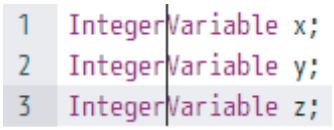


図 2.16

離れた位置の文字列を複数選択する場合

図 2.17 に示すように `Variable` が離れた位置があったとします．カーソルを `Variable` に合わせて，Ctrl-Alt-→-→などと→を 2 回続けて入力してください．すると図 2.18 に示すように，離れた位置にある `Variable` を複数選択することができます．

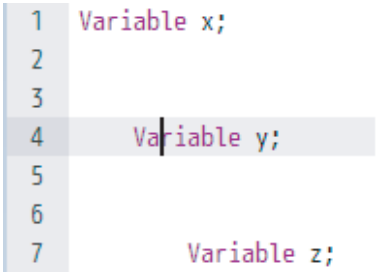


図 2.17

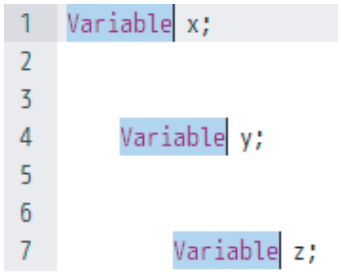


図 2.18

グラフィックドライバで Intel Graphics Controller をご利用の場合には，こちらの設定が優先され画面が回転する場合があります．多重カーソルを使用する場合はグラフィックドライバの設定から画面回転の機能を無効に設定してください．

2.3 エディタ設定

エディタの文字サイズやテーマ（文字色や背景色）などのエディタに関する設定は，[メニューバー] → [設定] → [エディタ]，もしくはエディタパネルの右上にある歯車マークをクリックすることで開か

れる設定ウィンドウ（図 2.19）から行います．

図 2.19　エディタ設定画面

エディタ設定の左部分は設定を確認するためのサンプルコードです．設定の変更がサンプルコードにリアルタイムに反映されますので，自由に記述して設定を調整してください．

設定を反映する場合は右下にある [OK] ボタンを押してください．出荷時の設定に戻す場合は [デフォルト] ボタンを押してください．

エディタ設定の各項目をカテゴリごとに以下に説明します．

2.3.1　レイアウト

フォントサイズ

文字サイズ（最小値 6 /最大値 72）を設定します．

テーマ

エディタのテーマには，配色が明るめの Light 系と暗めの Dark 系の 2 系統があり，合わせて表 2.1 に挙げる 33 種類あります．これらの中からお好みのテーマを設定します．

表 2.1　テーマ一覧

Light 系（14 種類）	Dark 系（19 種類）
Chrome	Ambiance
Clouds	Chaos
Crimson Editor	Clouds Midnight
Dawn	Cobalt
Dreamweaver	idle Fingers
Eclipse	Kr Theme
GitHub	Merbivore
Retro LCD	Merbivore Soft
Solarized Light	Mono Industrial
TextMate	Monokai
Tomorrow	Pastel on dark
XCode	Solarized Dark
Kuroir	Terminal
KatzenMilch	Tomorrow Night
	Tomorrow Night Blue
	Tomorrow Night Bright
	Tomorrow Night Eighties
	Twilight
	Vibrant Ink

パッドサイズ

記述を行う領域の左端と記述を始める位置の間のスペースのサイズ（パッドサイズ（最小値 0/最大値 32））を設定します.

2.3.2　ガター

ガターとは記述領域の左側にある領域（図 2.20）のことで，行番号やアノテーションなどの情報を表示する領域です.

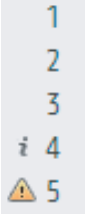


図 2.20　ガター

行番号

ガターの領域に行番号を表示するかどうかを設定します.

{ }, [ ], /* */の折り畳み

{ }, [ ], /* */が行をまたぐとき，その間の記述を折り畳む機能を有効にするかどうかを設定します．折り畳みは括弧の開始行の高さの位置にあるガターの領域に表示された下三角マーク▽から行います（図 2.21，図 2.22 参照）.

```
16 for(i=S.first(); i<S; i=S.next(i)){
17     a[i] = 2*S.position(i);
18 }
```

図 2.21 下三角マーク▽からの折り畳み

```
16 for(i=S.first(); i<S; i=S.next(i)){⏎}
```

図 2.22 折り畳み後

2.3.3 タブ

タブサイズ

タブのサイズ（最小値 2 /最大値 8）を設定します．例えば 4 と設定すると，半角スペース 4 個分の長さを持ったタブに設定します.

ソフトタブ

タブを半角スペースで代用するかどうかを設定します．半角スペースで代用した場合でも，カーソルの移動はタブがあった場合と同じ移動になります.

2.3.4 スクロールバー

水平スクロールバーを常に表示

水平スクロールバーを常に表示します.

垂直スクロールバーを常に表示

垂直スクロールバーを常に表示します.

2.3.5 視覚効果

現在行をハイライト

カーソルがある行をハイライトするかどうかを設定します.

インデント単位の表示

タブまたは半角スペースによるインデント単位を，図 2.23 に示すようにエディタ上に表示するかどうかを設定します．全角スペースによってインデントの単位を与えることはできません.

```
1  12345678
2
```

図 2.23 タブサイズが 4 の場合のインデント単位の表示の様子（2 行目）

制御文字の表示

通常表示されない文字（改行，半角スペース，タブ，終端）をシンボリックに表示するかどうかを設定します．これら表示はエディタのテーマによって異なります．

{ }, [ ], /* */の折り畳みをフェードアウト

{ }, [ ], /* */の折り畳みを行う下三角マーク▽にマウスカーソルが当たっていない場合には，自動的にフェードアウトするかどうかを設定します．

2.3.6　自動補完

( ), { }などの自動補完

括弧のように対になる文字がある場合，最初の文字を入力した際に自動でもう一方の文字を補完するかどうかを設定します．補完対象として次の 5 通りがあります．

- ( )
- { }
- [ ]
- " "
- ’ ’

SIMPLE キーワード

モデリング言語 SIMPLE の予約語（キーワード）の文字列を入力中に，自動補完のリストを表示するかどうかを設定します．自動補完リストの中には SIMPLE の予約語でなくとも，ローカルに記述した文字列も自動で含まれます．補完の程度には次の 4 段階があります．

- なし：自動補完を行いません．
- 標準キーワード：SIMPLE で頻出する標準的なキーワードのみを自動補完リストに含めます．
- 全キーワード：SIMPLE の全キーワードを自動補完リストに含めます．
- 全キーワード & C/C++：SIMPLE の全キーワードと C/C++のキーワードを自動補完リストに含めます．

2.3.7　折り返し

折り返し機能

折り返し文字数に指定した文字数で，行を折り返すかどうかを指定します．

折り返し文字数

行の折り返しを行う文字数（最小値 10/最大値 1000）を指定します．

第3章 ログおよびビルドと実行について

ログをみることでモデルのビルドと実行に関わる情報を取得することができます．結果の検証，モデルのチューニングに役立てることができます．

3.1 ログ

SIMPLE の記述ができたら，次の 2 つのプロセスを経て，求解結果を得ることができます．

ビルドプロセス

SIMPLE の記述自体が正しく行われているかチェックし実行ファイルを生成するプロセス

実行プロセス

Numerical Optimizer が記述された数理計画問題を求解するプロセス

Nuorium はこれらプロセスの挙動をログとしてユーザに表示し，最適化計算がどのように行われているかを知らせます．

通常，モデルの記述には試行錯誤が必須となり，その試行錯誤の中には，単純なビルドエラーから数理計画問題に特有な実行時エラーまで，様々な障壁があります．Nuorium はそれらログが教える解決策や SIMPLE の記述の不備に関する情報と密に連係して，ユーザの試行錯誤をサポートします．

3.1.1 ログの表示領域

ログの表示領域は画面右側（図 3.1）か画面下側（図 3.2）の何れかの領域を選択することができます．表示位置の変更は [メニューバー] → [表示] → [ログ] → [ログの表示位置を交換] または F8 をキー入力して行います．

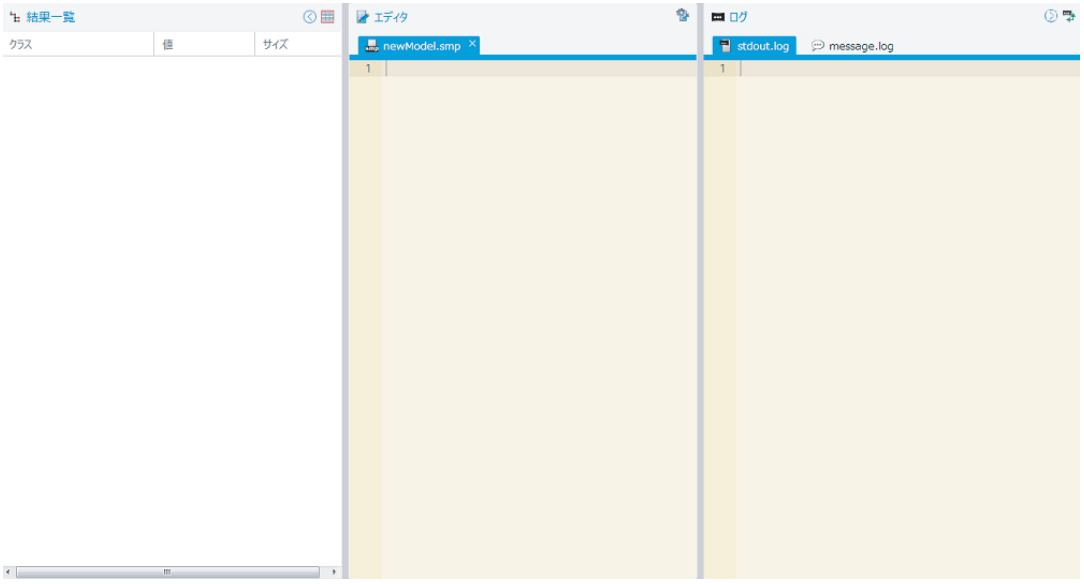


図 3.1　ログの表示領域（画面右側の場合）

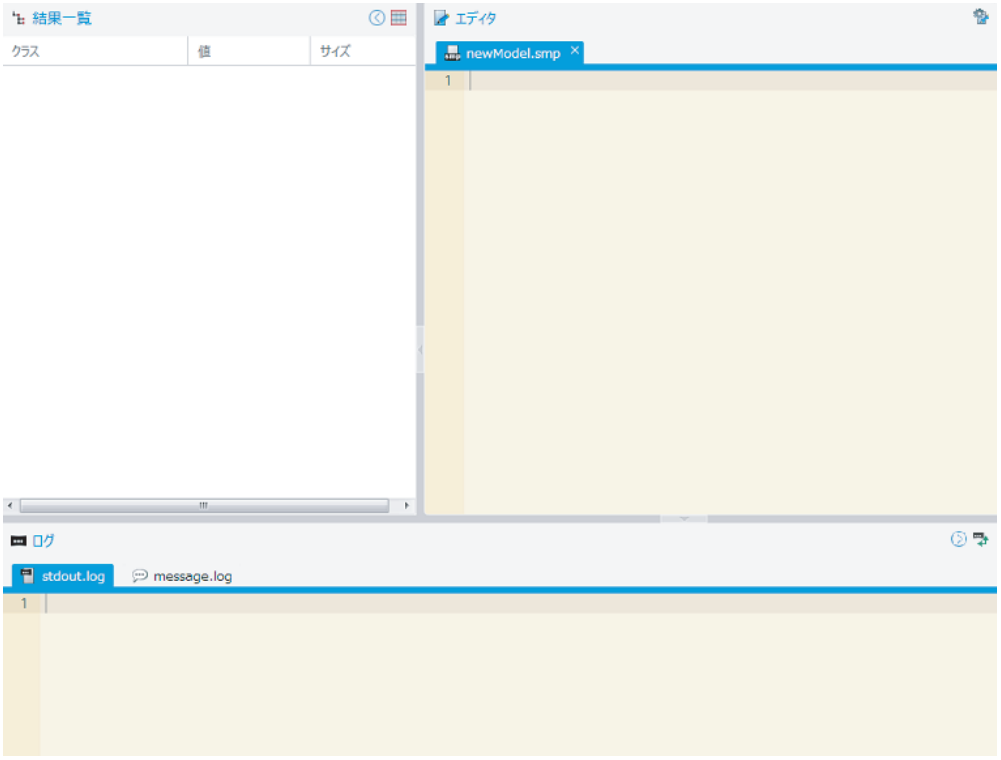


図 3.2　ログの表示領域（画面下側の場合）

3.1.2　ログの内容

ログは `stdout.log` と `message.log` の 2 種類があります．これらの内容は次のとおりです．

`stdout.log`

Nuorium は SIMPLE のビルドメッセージおよび Numerical Optimizer の実行に伴う標準出力を

stdout.log に表示します（図 3.3）．なおビルドおよび実行を行う度に，表示領域をクリアして最新のログに更新します．

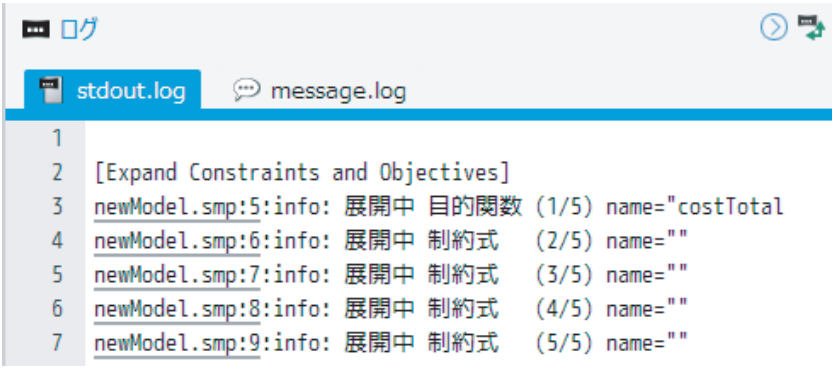


図 3.3 stdout.log

stdout.log に表示するメッセージのうち，行番号を記載したメッセージには，Nuorium がエディタ内の該当行へのリンクとアノテーションを自動で追加します．マウスカーソルを合わせてクリックすることで，該当行にカーソルをジャンプさせることができます．

メッセージには次のレベルがあり，識別しやすいようにそれぞれ色分けをします．

- エラー（赤）
- 警告（黄）
- 情報（灰）

message.log

Nuorium は画面右上に一時的に表示されるメッセージの履歴を message.log に表示します（図 3.4）．

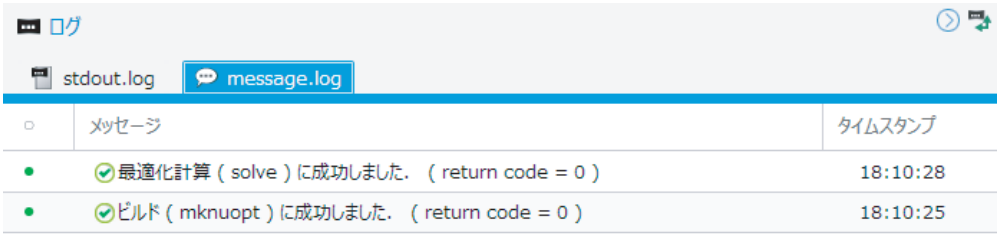


図 3.4 message.log

1 列目にはメッセージの内容に応じた次の 4 つのカテゴリを表示します．

- エラー（赤）
- 警告（黄）
- 情報（灰）
- 成功（緑）

2 列目にはメッセージを表示します．

3 列目にはメッセージのタイムスタンプを表示します．

3.2 ビルド

3.2.1 実行ファイルの生成

ビルドが成功すると実行ファイル（exe ファイル）が生成されます．このあとファイル編集すると，生成された実行ファイルは自動的に消去されます．新しく編集したファイルについては，新たに再びビルドを行ってください．

またビルドが成功して実行ファイル（exe ファイル）が生成された場合には，ファイル名の左のアイコンが exe アイコンに変化します（図 3.5）．ファイル編集を行って実行ファイルが消去されると，元のファイルアイコンに変化します．

図 3.5　exe アイコン

3.2.2 使用できない name 属性について

Nuorium では name 属性として次の引数を用いることができません．

- 二重引用符（"）
- 半角コンマ（,）
- 制御文字（\r，\n，\t など）
- name 引数の重複．以下はその例です．

```
Variable x(name="a"), y(name="a");
```

- name 引数とインスタンス名の重複．以下はその例です．

```
Variable x(name="y");
Variable y;
```

3.3 ビルド & 実行オプション

ビルドまたは実行の際にオプションを付加することができます．[メニューバー] → [実行] → [オプション] または F7 キーから，ビルド & 実行オプション画面（図 3.6 参照）を開いてオプション設定を行ってください．

図 3.6 ビルド & 実行オプション画面

オプションの内容をビルドと実行について，それぞれ説明すると次のとおりです．

3.3.1 ビルド

ビルド時に並列化オプションを付与

チェックをするとビルド時に-parallel オプションをつけてコンパイルを行います．この指定によって分枝限定法や wcsp を利用する際に並列化計算を行うことができます．合わせてスレッド数の”上限”を SIMPLE のコードに options.bbthreads の値として記述してください．

例えば options.bbthreads = 2; と記述した場合には，stdout.log の途中で次のようなメッセージが表示され，2 スレッドまで立ち上がっていることが確認できます．

```
...
<iteration begin>
        Parallel up to 2 threads
dllpath = C:\Program Files (x86)\Mathematical Systems Inc\NUOPT\bin\tbb.dll
...
```

ビルド時に詳細を出力

チェックをするとビルド時に-v オプションをつけてコンパイルを行います．コンパイル環境の詳細な確認が可能になり，コンパイル失敗時の原因究明に役立ちます．

3.3.2　実行

実行時にデータオプションを付与

チェックをするとビルド時に-data オプションをつけてコンパイルを行います．後述する「データ」の入力フォームに書いた一時的な情報を入力情報として実行することができます．

データ

-data オプションを有効にした際に入力情報となるデータを記述するフォームです．書式はデータファイル（.dat ファイル）と同じです[1]．

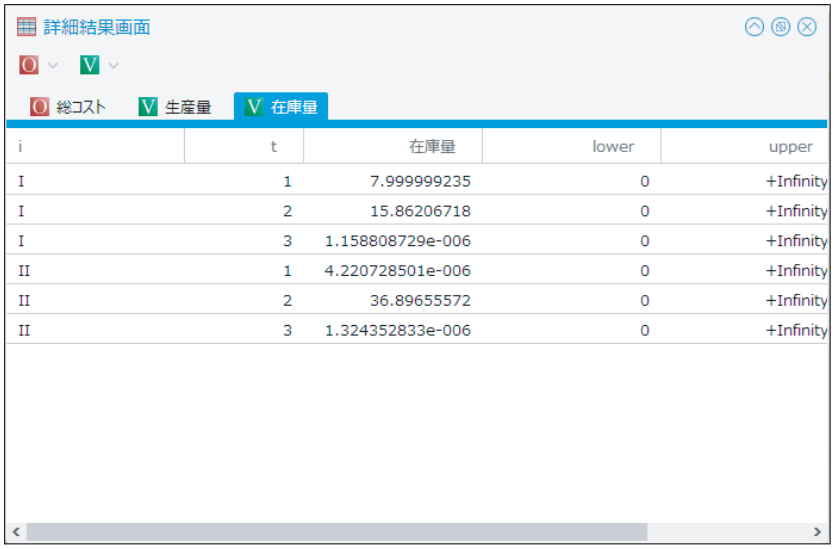
[1]書式に関しては『数理システム Numerical Optimizer/SIMPLE マニュアル』のデータファイルの章をご参照ください．

第 4 章　詳細結果について

詳細結果をみることでモデルの最適化結果に関する情報を取得することができます．変数の値のみならず，目的関数，制約式，パラメータ，式の値も表示することができるため，最適化計算への入力と結果の両方を確認することができます．

4.1　詳細結果画面

詳細結果画面は最適化計算の詳細な結果をグリッド状に表示する画面（図 4.1）です．



i	t	在庫量	lower	upper
I	1	7.999999235	0	+Infinity
I	2	15.86206718	0	+Infinity
I	3	1.158808729e-006	0	+Infinity
II	1	4.220728501e-006	0	+Infinity
II	2	36.89655572	0	+Infinity
II	3	1.324352833e-006	0	+Infinity

図 4.1　詳細結果画面

4.1.1　詳細結果画面の表示

詳細結果画面を表示する方法は次の 2 通りがあります．

1. [メニューバー] → [表示] → [詳細結果を表示]，もしくは結果一覧の右上にある「詳細結果アイコン」（図 4.2）をクリックすると表示します．
   詳細結果画面が表示されている状態で，再度，詳細結果アイコンをクリックすると詳細結果画面が非表示となります．

図 4.2　詳細結果アイコン

2. 結果一覧（図 4.3）から表示したい対象（目的関数，制約式，パラメータ，変数，式）をダブルクリックすると表示します．
   詳細結果画面を表示している状態で，再度，結果一覧の表示したい対象をダブルクリックすると，ダブルクリックした対象の詳細結果を表示します．
   また，図 4.3 では結果一覧にパラメータの値の列が表示されていません．詳細結果画面でも同様です．理由はパラメータの出力がデフォルトでは行われないためです．このような場合は詳細結果アイコンをクリックした場合と同様に，最も左のタブが選択された状態で表示することになります．

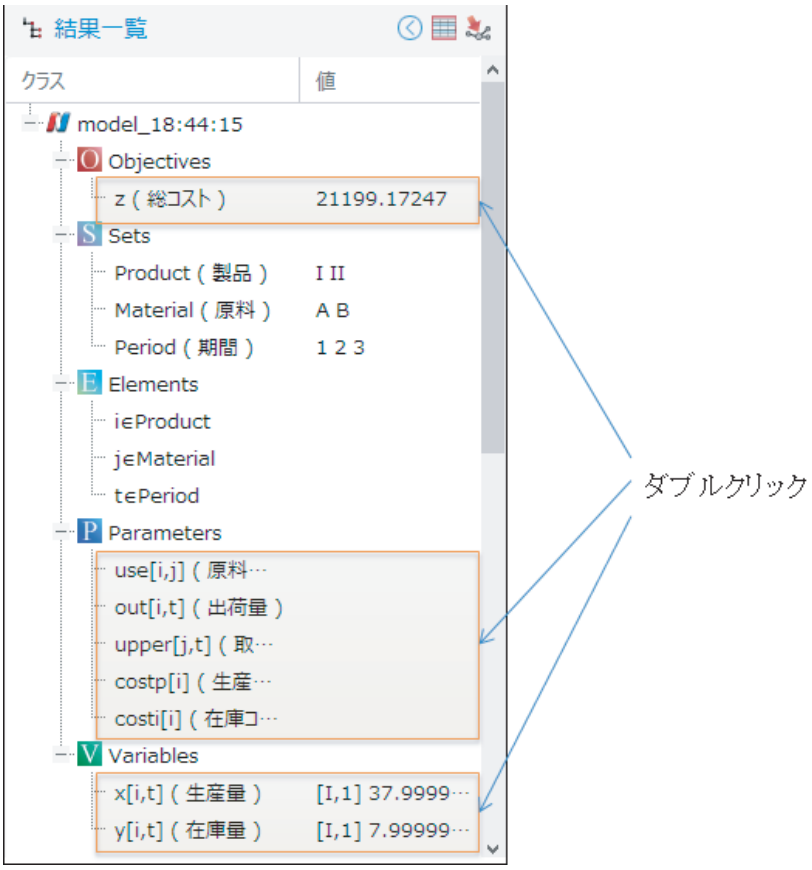


図 4.3　結果一覧

4.2 詳細結果の表示形式

結果表示を行う SIMPLE の主なクラスは次のとおりです．

目的関数

- `Objective`

パラメータ

- `Parameter`

変数

- `Variable`

式

- Expression
- Constraint

詳細結果画面の表示形式は目的関数，パラメータ，変数，式のそれぞれで共通する事柄とそうでない事柄に分けることができます．

そこではじめに共通する表示形式について説明し，その後に目的関数，パラメータ，変数，式，それぞれの表示形式について順に説明します．

4.2.1 共通する表示形式

配列（添字を持つ値）の表示

配列の場合は配列の次元の数だけ列を追加します．n 次元の配列の場合には次のようになります（図 4.4）．

- 1 列目：1 次元目の添字
- 2 列目：2 次元目の添字
- …
- n 列目：n 次元目の添字
- n+1 列目：値

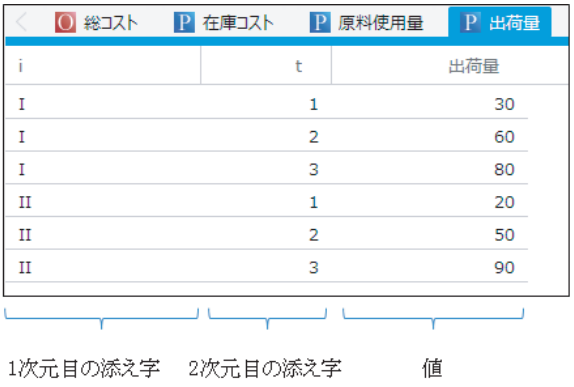


図 4.4 2 次元配列（添え字を持つ値）の表示例

添字を持たない値（スカラー）の表示

スカラーは 0 次元の配列と定義し，1 行 1 列の表示となります（図 4.5）．

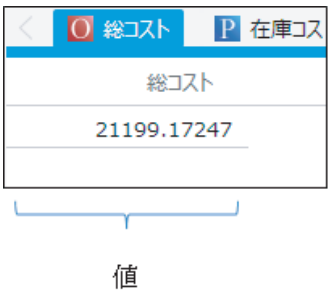


図 4.5 添え字を持たない値（スカラー）の表示例

最適化を繰り返し実行した場合の表示

モデルの中で複数回 `solve()` 関数を呼び出した（複数回最適化計算を行った）場合は，各々の項目の結果は最大で `solve()` 関数を呼び出した回数個生成されます．詳細結果ではこの複数個の結果を 1 つの結果として表示します（図 4.6）．

`solve()` を n 回呼び出した際の i 回目の最適化計算の結果を「計算番号 i の結果」とよぶと，表示の規則は次のようになります．

- 1 列目：計算番号
- 2 列目：1 次元目の添字
- …
- n+1 列目：n 次元目の添字
- n+2 列目：値

特にスカラーの場合は 0 次元の配列と考えるため次のようになります．

- 1 列目：計算番号
- 2 列目：値

図 4.6　`solve()` が何度もよばれた場合の結果例

4.2.2　目的関数

目的関数は必ずスカラーな値であるため，「共通する表示形式」で述べたスカラー表示のとおりとなります．値の列には最適化計算で得られた目的関数の値を表示します．

4.2.3　パラメータ

パラメータは配列，スカラーともにあります．「共通する表示形式」のとおりの表示となります．値の列には最適化計算で用いたパラメータの値を表示します．

パラメータの値はデフォルトでは表示しません．表示/非表示の切替はモデル記述において `options.outputParameter` に次の値を設定することで行えます．

- 1:表示
- 0:非表示（デフォルト）

例えば，表示する場合はモデルに次の一行を `solve()` 関数の前に記述します．

```
options.outputParameter = 1;
```

4.2.4 変数

変数は配列，スカラーともにあります．値の列には最適値を与える解の値を表示します．

変数については「共通する表示形式」で述べた値の列の右側に，以下の 3 列が追加されます（図 4.7）．

- 変数の下限値
- 変数の上限値
- 双対変数の値

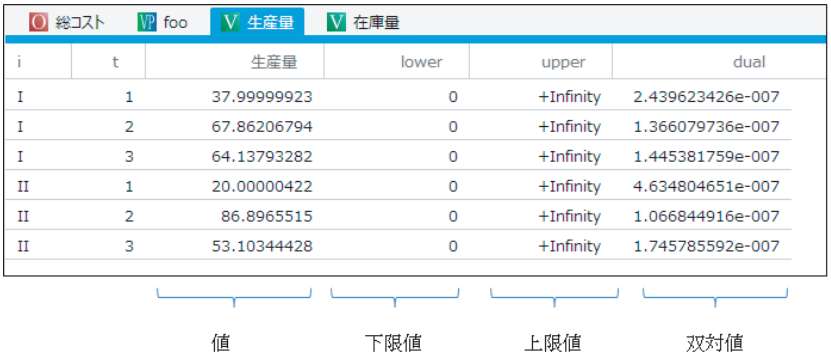


図 4.7　変数の表示例

変数の上下限値として「+Infinity」と「-Infinity」の表示は正負の無限大を表し，上下限値の設定がないことを意味しています．つまり上限値として「+Infinity」の場合は上限値の設定がなく，下限値として「-Infinity」の場合は下限値の制限がありません．

4.2.5 式

式は配列，スカラーともにあります．「共通する表示形式」のとおりの表示となります．値の列には最適化計算で得られた式の値を表示します．

式の値はデフォルトで表示します．表示/非表示の切替はモデル記述において `options.outputExpression` に次の値を設定することで行えます．

- 1:表示（デフォルト）
- 0:非表示

例えば，非表示とする場合はモデルに次の一行を `solve()` 関数の前に記述します．

```
options.outputExpression = 0;
```

4.3 詳細結果画面上の操作

詳細結果画面（図 4.8）で可能な操作は以下になります．

- 選択ボタン / タブの選択
- フィルタ
- 領域選択とクリップボードへのコピー

図 4.8　選択ボタン，タブ，グリッド

これらの操作を順に説明すると次のとおりです．

選択ボタン / タブの選択

詳細結果を確認したい項目名を選択ボタンから選ぶ，または，タブを直接クリックするとその画面が表示できます．

フィルタ

フィルタは列ごとに条件を入力でき，すべての条件が満たされている行のみ表示します．また，列は型情報（数値型または文字列型）を保持しています．数値型の場合は数値の範囲または一致の条件を入力できます（図 4.9）．文字列型の場合は文字列を入力でき，入力文字列が含まれている行を条件が満たされている行とします（図 4.10）．

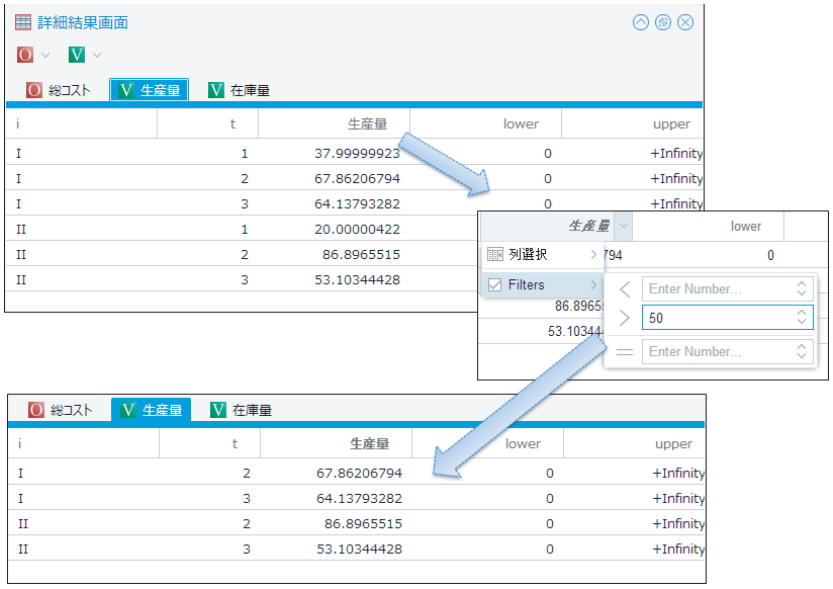


図 4.9 数値型のフィルタ

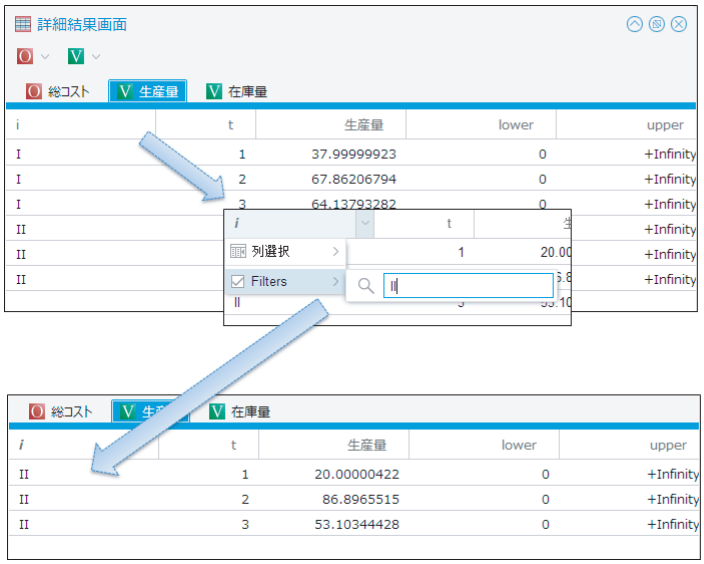
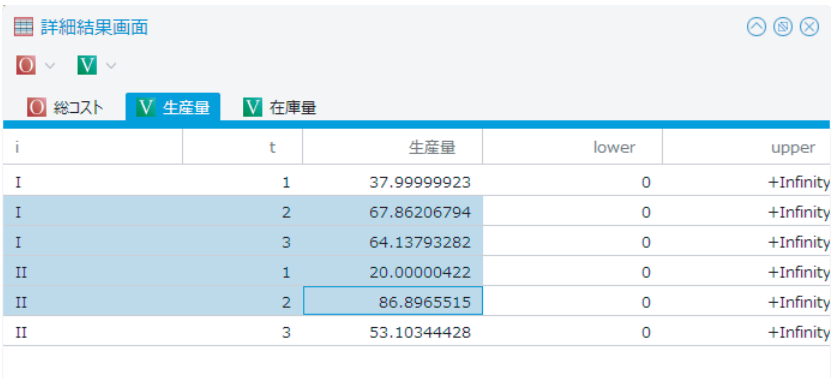


図 4.10 文字列型のフィルタ

領域選択とクリップボードへのコピー

詳細結果のグリッドの領域をマウスで選択し，Ctrl-C を押下することでその選択情報をクリップボードにコピー（図 4.11）することができます．クリップボードにはタブ区切りのテキストデータとして格納されているため，Excel やテキストエディタ等にペーストすることができます．



i	t	生産量	lower	upper
I	1	37.99999923	0	+Infinity
I	2	67.86206794	0	+Infinity
I	3	64.13793282	0	+Infinity
II	1	20.00000422	0	+Infinity
II	2	86.8965515	0	+Infinity
II	3	53.10344428	0	+Infinity

図 4.11　選択領域のコピー

4.3.1　詳細結果画面の自動更新

詳細結果画面を表示している場合に最適化を実行すると，詳細結果画面の表示は最新の情報に自動的に更新されます．

しかし，次の場合には，これまで表示されていた計算結果が削除されますのでご注意ください（図 4.12）．

- 最適化の実行が失敗した場合
- ビルドエラーの場合

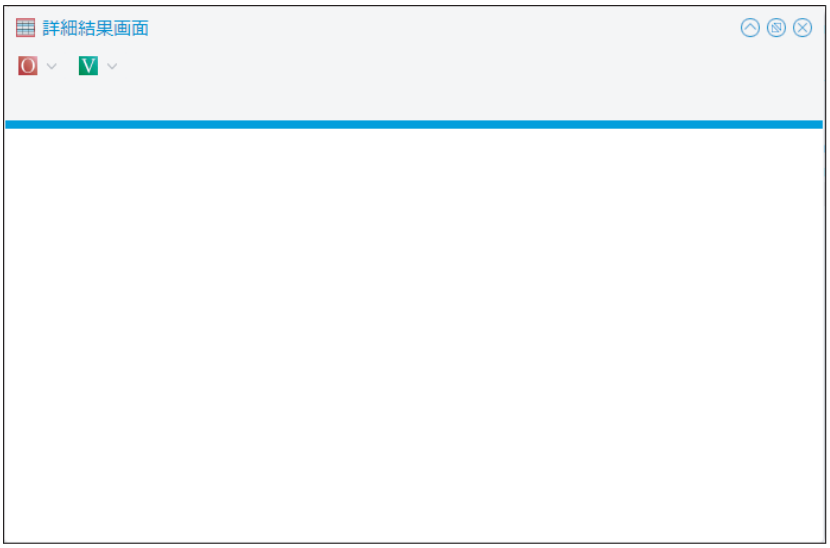


図 4.12　ビルドエラー時の詳細結果画面

第5章 データ入出力について

Visual Analytics Platform（以下 VAP と略称）上では，アイコンベースの処理フローを描くことにより，Nuorium で作った最適化モデルをデータ分析ツール等と手軽に連係させることができます．

5.1 入力データ

Nuorium で編集するモデルファイル（smp ファイル）に記述された数理計画問題のモデルと連係できる入力データは次の 4 種類です．

- csv 形式データ
- dat 形式データ
- Numerical Optimzier パラメータ
- VAP 専用の表形式データ

以下ではこれらについて順に説明しますが，モデルの入力データとして指定するには，何れの場合も Visual Analytics Platform 上のフローで Numerical Optimizer モデルアイコンの上流に配置して指定します．入力データアイコンをモデルアイコンの上流に配置するには，入力データアイコンからモデルアイコンへ「矢印線」を引く必要があります．「矢印線」の引き方については VAP マニュアルの「矢印線」の項を参照してください．

5.1.1 csv 形式データ

csv ファイルをデータとして入力したい場合は，VAP 上に csv ファイルをドラッグ & ドロップし，ドロップされたアイコンをダブルクリックして「データインポート」の設定を行い，Numerical Optimizer モデルアイコンの上流に配置して，入力データとして指定します．

csv ファイルの具体的な書き方については，『Numerical Optimizer / SIMPLE マニュアル』の「データファイル」の章を参照してください．また，データインポートについては VAP マニュアルの「データインポート」の項を参照してください．

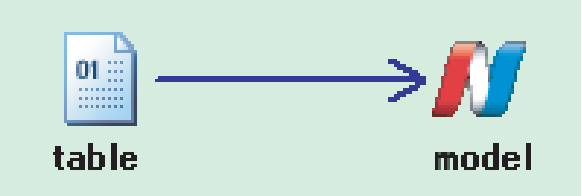


図 5.1　csv 形式データの入力

5.1.2　dat 形式データ

dat ファイルをデータとして入力したい場合は，VAP 上に dat ファイルをドラッグ & ドロップして Numerical Optimizer モデルアイコンの上流に配置して，入力データとして指定します．

dat 形式データは Numerical Optimizer 専用の形式のデータであり，具体的な書き方については，『Numerical Optimizer/SIMPLE マニュアル』の「データファイル」の章を参照してください．

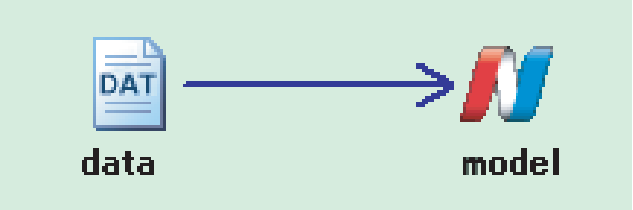


図 5.2　dat 形式データの入力

5.1.3　Numerical Optimizer パラメータ

Numerical Optimizer パラメータは Numerical Optimizer の各種のパラメータを設定する際に用いる専用のデータ形式です．

Numerical Optimizer パラメータの設定は，VAP 上で「Numerical Optimizer パラメータ」アイコンを配置して，「Numerical Optimizer パラメータ」アイコンを編集することで行うことができます．

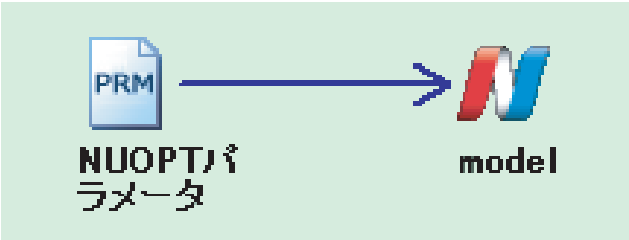


図 5.3　Numerical Optimzer パラメータの設定

設定を行った Numerical Optimizer パラメータを入力データとして指定するには，他の入力データの形式と同様に Numerical Optimizer モデルアイコンの上流に配置して，入力データとして指定します．

5.1.4　VAP 専用の表形式データ

VAP 専用の表形式データを入力することもできます．つまり，Numerical Optimizer モデルアイコンの上流に VAP のスクリプトアイコン，クリーニングアイコン，フィルタリングアイコン等を配置して，上流に配置した各アイコンの処理結果を入力とすることができます．上流に配置する各アイコンについては『VAP マニュアル』を参照してください．

5.2 出力データ

Numerical Optimizer が計算して求めた変数と目的関数の値を VAP 上のフローの下流に流すことができます．下流に流すデータの設定は [メニューバー] → [実行] → [VAP への出力設定]，もしくは結果一覧の右上にある「VAP への出力設定ボタン」（図 5.4 参照）をクリックすることで開かれる「VAP への出力設定」のウィンドウから行います．

図 5.4　VAP への出力設定ボタン

ビルドまたは実行後に変数と目的関数がリストされるので，チェックを入れて下流に流す出力データを設定します．ただし，ファイル名として使用できない次の文字が含まれた名前のデータは，下流に流すことはできません．

```
\ / : * ? " < > |
```

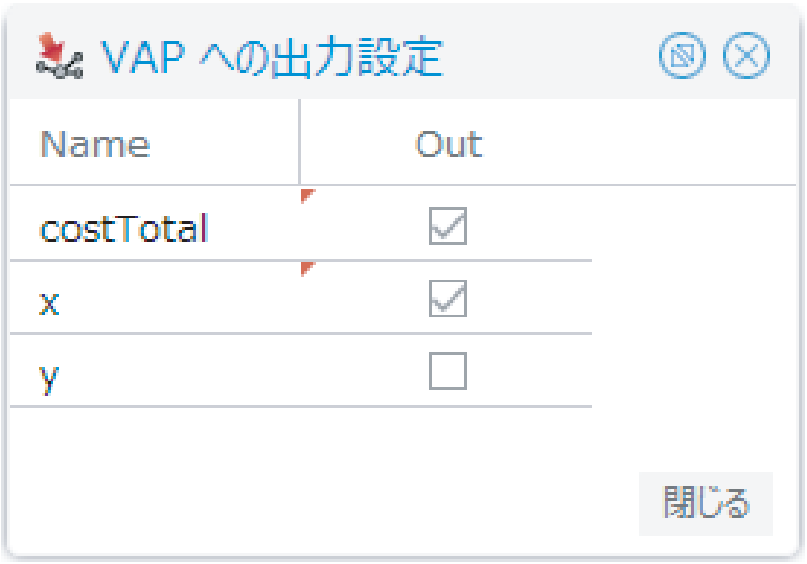


図 5.5　VAP への出力設定の画面

Nuorium を終了後，VAP のフロー上で Numerical Optimizer モデルアイコンの出力データを確認することができます．アイコンの入出力データの確認方法については VAP マニュアルの「入出力マッチング」の項を参照してください．

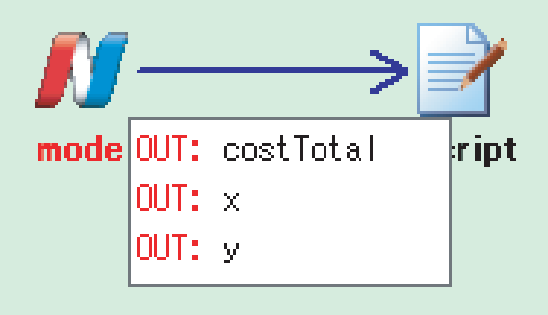


図 5.6　出力設定の確認

第 6 章　ファイル入出力について

Nuorium が編集対象とするファイル形式，ファイルの保存に関わること，そしてエクスポートについて説明します．

6.1　ファイル形式

編集対象

Nuorium で編集可能なファイルは次の拡張子のファイルに限ります．

- smp（モデリング言語 SIMPLE で数理計画問題が記述されたファイル）

ファイルを編集中には，ファイル名の左に対応するファイルアイコンが表示されます（図 6.1）．また編集によって変更が加えられたファイルのファイル名の右側には，*のマークが付きます（図 6.2）．

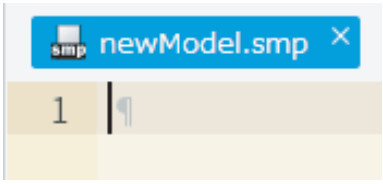


図 6.1　ファイルアイコン

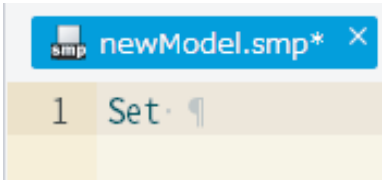


図 6.2　変更が加えられたファイル

改行コードと文字コード

Nuorium での改行コードと文字コードの取り扱いは以下のようになっています．

- 読み込まれたファイルの場合，ファイルの編集と書き込みは元の改行コードと文字コードで行われます．
- 新規に生成されたファイルの場合，ファイルの編集と書き込みは改行コードが CRLF で文字コードが SJIS で行われます．

機種依存文字

ファイルに機種依存文字を用いた場合には，正常に保存されない可能性があります．ファイルの記述には機種依存文字を使用しないことを推奨します．

ファイルサイズの上限

Nuorium のエディタで扱えるファイルサイズの上限は 10 MB です．10 MB を超過したファイルは開くことができません．

6.2 ファイルの保存について

Visual Analytics Platform 上に配置され Nuorium で編集が行われるファイルは，Visual Analytics Platform が管理しています．このためファイルの保存は，外部からドラッグ＆ドロップによって Visual Analytics Platform に持ち込まれた元のファイルには行われません．

6.3 エクスポート

Nuorium は編集を行っている SIMPLE ファイルやビルドして生成された実行ファイル（exe ファイル）などの一連の生成物を外部にエクスポートすることができます．

6.3.1 エクスポートの方法

エクスポートは [メニューバー] → [エクスポート] より開かれるエクスポート画面（図 6.3）から行います．エクスポート画面ではエクスポートを行うファイルの一覧が表示されており，エクスポート先のフォルダを選択するフォームがあります．エクスポート先を指定して [OK] ボタンを押してエクスポートを行ってください．モデルファイルの名称でフォルダが自動生成されエクスポートが完了します．

図 6.3　エクスポート画面

6.3.2 エクスポート先の階層構造

実行によって求解結果が得られた状況で，エクスポートを行うと次に示す階層構造が自動で生成されます.

- (エクスポート先に自動生成したフォルダ)/
  - smp ファイル（編集対象のモデルファイル）
  - exe ファイル（ビルドで生成した実行ファイル）
  - build.log（ビルドに伴うログ情報）
  - json ファイル（モデル構造の情報）
  - run.log（実行に伴うログ情報）
  - result/
    - solfile.txt（実行結果の情報）
    - progress.txt（実行の進捗情報の要約）
    - result_info.json（実行結果に伴うモデル構造の情報）
    - csv ファイル（実行結果の詳細情報）

ビルドや実行が完了しておらず，はじめから存在していないファイルがある場合には，それらは存在しないためエクスポートされません．ファイルの有無に関わる各状況とエクスポートによって生成される階層構造は次のとおりです.

ファイルを開いただけの状況

- (エクスポート先に自動生成したフォルダ)/
  - smp ファイル（編集を行っていたモデルファイル）

ビルドの成功の可否に関わらずビルドログが得られた状況

- (エクスポート先に自動生成したフォルダ)/
  - smp ファイル（編集を行っていたモデルファイル）
  - build.log（ビルドに伴うログ情報）

ビルドが成功した状況

- (エクスポート先に自動生成したフォルダ)/
  - smp ファイル（編集を行っていたモデルファイル）
  - exe ファイル（ビルドで生成した実行ファイル）
  - build.log（ビルドに伴うログ情報）
  - json ファイル（モデル構造の情報）

実行の求解の可否に関わらず実行ログが得られた状況

- (エクスポート先に自動生成したフォルダ)/
  - smp ファイル（編集を行っていたモデルファイル）
  - exe ファイル（ビルドで生成した実行ファイル）
  - build.log（ビルドに伴うログ情報）
  - json ファイル（モデル構造の情報）

- run.log（実行に伴うログ情報）

付録 A 正規表現

Nuorium のエディタでは，Javascript 準拠の正規表現を用いた検索が可能です．正規表現の詳細に関しては Javascript による正規表現の文献[2]をご参照ください．

以下では正規表現に用いるメタ文字とその意味，および例文を幾つか記載します．

A.1 メタ文字

正規表現で使用するメタ文字とその意味を表 A.1 に記載します．共通するよくある注意事項としては次があります．

- -は [ ] で囲まれたときのみ，メタ文字となります．
- [ ] で囲まれた中でメタ文字となるのは，^，-，] のみです．つまり . や?などはメタ文字とはみなされません．
- ^は [ ] で囲まれたときと，そうでないときとで異なる 2 つの意味を持つメタ文字です．

表 A.1　メタ文字一覧

メタ文字	説明
^	行の先頭にマッチします．
$	行の末尾にマッチします．
*	直前の文字の 0 回以上の繰り返しにマッチします．
+	直前の文字の 1 回以上の繰り返しにマッチします．
?	直前の文字の 0 回か 1 回の出現にマッチします．
.	改行文字以外のどの 1 文字にもマッチします．
(x)	x にマッチし，マッチした内容を記憶します．
(?:x)	x にマッチしますが，マッチした内容は記憶しません．
x(?=y)	x に y が続く場合のみ x にマッチします．
x(?!y)	x に y が続かない場合のみ x にマッチします．
x|y	x または y にマッチします．
{n}	直前の文字がちょうど n 回出現するものにマッチします．
{n,m}	直前の文字が少なくとも n 回，多くても m 回出現するものにマッチします．
[xyz]	囲まれた文字の何れかにマッチします． -を用いて [x-z] のように文字の範囲を指定することも可能です．
[^xyz]	囲まれていない文字の何れかにマッチします．

[2] 正規表現-JavaScript|MDN（https://developer.mozilla.org/ja/docs/Web/JavaScript/Guide/Regular_Expressions）など

表 A.1 メタ文字一覧

メタ文字	説明
\b	単語の区切りにマッチします.
\B	単語の区切り以外にマッチします.
\d	数字にマッチします.
\D	数字以外にマッチします.
\s	半角スペース，全角スペース，タブを含む 1 つの空白文字にマッチします.
\S	空白文字以外の 1 文字にマッチします.
\t	タブにマッチします.
\w	英数字およびアンダースコアにマッチします.
\W	英数字およびアンダースコア以外にマッチします.

A.2 エスケープ

メタ文字に使用されるメタ文字を通常の文字として扱いたい場合には，\を用いてエスケープしてください．対応表を表 A.2 に記載します.

表 A.2 エスケープ前後の対応表

エスケープ前	エスケープ後
\	\\
^	\^
$	\$
*	*
+	\+
?	\?
.	\.
(	\(
)	\)
{	\{
}	\}
[	\[
]	\]
|	\\|
-	\-

A.3 例文

表 A.3 に正規表現の例を記載します.

表 A.3 正規表現の例文

正規表現	説明
".*"	文字列を選択
(Variable|IntegerVariable)	Varibale または IntegerVariable を選択
//.*$	//によるコメントのみ選択
options\..*=.*;	options 設定を選択
.*==.*$	等式制約がある行を選択
.*(<=|>=).*$	不等式制約がある行を選択
.*(==|<=|>=).*$	制約条件がある行を選択
^(?!xyz).*$	xyz から始まる行以外の行を選択

索　引

記号・数字

* 35

C

csv ファイル 5, 31

D

dat ファイル 5, 31

E

Excel 29
exe ファイル 20, 36

I

Infinity 27

K

keyword 9

L

local 9

M

message.log 4, 19

N

Numerical Optimizer モデルアイコン 31
Numerical Optimzier パラメータ 31

O

options.outputExpression 27
options.outputParameter 26

S

snippet 9
solfile 37
stdout.log 4, 9, 18

V

VAP 31
VAP への出力設定 33
Visual Analytics Platform 2, 6, 31

あ

あいまい検索 10
アノテーション 9
アンコメント 8

い

インデント単位の表示 15

え

エクスポート 36
エスケープ 40
エディタ 3

お

折り返し 16
折り畳み 11, 15, 16

か

改行コード 35
下限値 27

ガター 14

き

機種依存文字 35
行番号 15

く

矩形選択 8

け

結果一覧 4
検索結果の全選択 11

こ

コメントアウト 8

し

式 25, 27
自動インデント 7
自動補完 8, 16
集中モード 8
出力データ 33
上限値 27
詳細結果画面 23

す

スカラー 25

せ

正規表現 39
正規表現検索 10
全文一致検索 11

そ

双対変数 27
属性 20
ソフトタブ 15

た

多重カーソル 12
タブサイズ 15

て

データオプション 22
テーマ 13

に

入力データ 31

は

配列 25
パッドサイズ 14
パラメータ 24, 26

ひ

表形式データ 31
ビルド 20

ふ

フォントサイズ 13

へ

並列化オプション 21
変数 24, 27

め

メタ文字 39

も

目的関数 24, 26
文字コード 35

ろ

ログ 4