CUMuLOUS – 拡張性を考慮した，大規模多分野問題最適化解析システム –

機体構造グループ
高嵜　浩一

概 要

非定常多分野最適化解析および特殊要素の開発に資する為の解析システム CUMuLOUS について，その解析対象，データ構造，モジュール構造および使用法を記す。基本的な機能に限定された CUMuLOUS 本体，および周辺のデータ操作のためのコード等およびその解説もこの CD-ROM におさめられており，本文でそのリンクを示す。

1　解析システムの概要

1.1　解析対象

本システムは研究活動の上で将来的な解析理論の進展，および実世界の構造様式／運用方式の変化に柔軟に対応して概念設計を円滑に行うことを念頭におき，以下の機能を最終的に実装する事を目標とした。

- 基本的な熱／構造有限要素解析機能（静的，動的，モード解析等）
- 多分野解析機能（軌道解析を含む）
- 並列処理を考慮したデータ構造／アルゴリズム
- 外部からリンク可能なユーザ定義要素モジュール，およびユーザ定義材料特性モジュール
- Fortran90 を採用：行列演算および並列化に適し，かつこの世界では一般的
- ポータブルなコード：用途は解析のみに限らず，将来的には動的加熱試験において，供試体の伝熱モデルを構築し，より高精度な加熱制御や他の用途に使用する事も考慮
- 動的問題の最適化／同定機能：海外に存在する最適化システム（POST,NPDOT,OTIS,VTOTS）[1, 2, 3, 4] と同等以上の機能に加え，大規模有限要素モデルにより適した実装を行う事

この解析システムに想定される適用例としては，以下をあげる事ができる。

- 航空宇宙機の多分野統合概念設計：迅速なパラメトリックスタディ
- 軽量柔軟熱構造の動的／制御連成リアルタイム解析
- ポータブルなコードである事を考慮し，機体構造モデルも組み込んだ加熱試験制御システム

1.2　データ構造

本システムのデータ構造は，それを利用するアルゴリズムと対応して，以下の方針で構築された。

- 記憶領域の節減，処理速度の向上，および並列計算に適したコード
- そのコードに対応した疎行列かつ階層構造のデータ構造
- そのデータに対応した，反復法を基本とした演算アルゴリズム

本システムは元は基本的な伝熱解析有限要素法コードから発展した物で，解析モデルの構築には「要素」，および「節点」を利用する。
しかしながら，本システムにおいてはこれらは広義の意味でこう定義される。

- 「要素」は拘束条件を定義する
- 「節点」は変数／定数を格納する

これにより，システムのデータ構造の一貫性と簡潔さが保たれ，将来のバージョンアップや新しいソルバーや要素の構築が容易となると考えられる。

システムのユーザは一般的な意味での構造要素に限らず，例えば表面の境界条件，また非定常問題における航空機の運動方程式のような拘束条件も形式的に「要素」としてモジュールを作成／リンクする事ができる。

また，CUMuLOUS ではシェル要素の厚さやロッド要素の断面積など，一般的には要素特性として定義される数値も「節点」に格納される。このような仮想の「節点」を複数の要素で共有する事により（図 1），記憶領域を削減できる事に加え，ある一定の解析領域のパラメータを同時に変化させ，または最適化する事が可能になる。

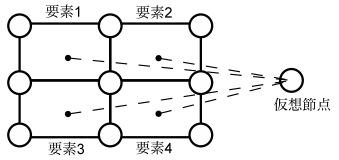

Fig.1　共有される仮想節点の概略図

さらに，感度解析／確率解析／同定解析等で材料特性の一部を変化させたいときも，その材料特性オプション値を負の符号に変更する事によって，（材料特性変化のオプションが用意されている要素であれば,）材料特性値のノミナル値からの変化率を持つ仮想の節点が生成される。

現バージョンでは，どのような問題の拘束条件でも，最終的には連立一次方程式 $\mathbf{H_x} \cdot \mathbf{x} = \mathbf{b}$ に変換される。

現行のバージョンでは，変数はすべて倍精度実数で節点に格納される。各節点に格納される変数は，階層構造のインデックスを用いてそのアドレスが指定され，結果的に疎行列として操作できる。各節点には複数の変数が格納されるが，その組み合わせは一意ではなく，その節点を共有する要素に合わせて自動的に記憶領域が確保される。

例えば Var ベクトルの場合，まず各節点ごとに何個の変数が格納されるかを示すインデックス VarIndexOfNode，各節点の各変数の種類（ID.f90 で定義）を示す VarType により，すべての 0 でない可能性のある変数を扱う。またこの性質により，各 VarType には変数が 1 対 1 に対応し，各節点に同じ VarType を持つ 2 つ以上の変数が格納される事はない。上記は荷重ステップに依存しない補助変数ベクトル Para についても同じである。

Hx マトリクスもまた異なる階層構造のインデックスで疎行列として格納される。反復法ソルバーを用いる事から，このマトリクスは各節点に対応するブロック部分行列以外は格納されない。現バージョンでは，速度向上と記憶容量削減のために要素番号，節点番号は 1 から隙間無く番号付けする必要がある。

本システムで確保される基本的な変数をソースコード CUMuLOUS.f90 から抜粋した物を以下に示す。

```
integer, allocatable :: ElemType(:)                ! nElem

integer, allocatable :: OptionIndexOfElem(:)       ! nElem
integer, allocatable :: ElemOption(:)              ! OptionIndexOfElem(nElem)

integer, allocatable :: NodeIndexOfElem(:)         ! nElem
integer, allocatable :: NodeOfElem(:)              ! NodeIndexOfElem(nElem)

integer, allocatable :: ElemIndexOfNode(:)         ! nNode
integer, allocatable :: ElemOfNode(:)              ! ElemIndexOfNode(nNode)

integer, allocatable :: VarIndexOfNode(:)          ! nNode
integer, allocatable :: VarType(:)                 ! VarIndexOfNode(nNode)
real   , allocatable :: Var(:,:)                   ! VarIndexOfNode(nNode), MaxLoadStep

integer, allocatable :: ParaIndexOfNode(:)         ! nNode
integer, allocatable :: ParaType(:)                ! ParaIndexOfNode(nNode)
real   , allocatable :: Para(:)                    ! ParaIndexOfNode(nNode)

integer, allocatable :: HxSupIndexOfNode(:)        ! nNode
integer, allocatable :: HxIndex_Node(:)            ! HxSupIndexOfNode(nNode)
```

```
integer, allocatable :: HxIndex_Var(:)          ! HxSupIndexOfNode(nNode)
integer, allocatable :: HxIndex_Para(:)         ! HxSupIndexOfNode(nNode)
real   , allocatable :: Hx(:,:)                 ! HxIndex(size(HxSupIndexOfNode,1),2), *
real   , allocatable :: Hp(:,:)                 ! HxIndex_Para( last(HxIndex_Para) ) , *

integer, allocatable :: PrecondMtxIndex(:)      ! nNode
real   , allocatable :: PrecondMatrix(:)        ! preconditioner matrix
!                                                 preconditionerType == 1: point jacobi
!                                                 preconditionerType == 2: block diagonal
!                                                 preconditionerType == 3: ILU/IC

real   , allocatable :: b(:,:)                  ! VarIndexOfNode(nNode), *
real   , allocatable :: frequency(:)            ! NoMode * alpha

integer, allocatable :: Fixed(:,:)              ! nFixed, 2 -> [FixedNodeNo FixedVarType]
integer, allocatable :: Sleep(:)                ! nSleep    -> [SleepElemNo]

integer, allocatable :: FixedIndex(:)           ! NoLoadStep or NoLSNode(for TAPFEM)
integer, allocatable :: SleepIndex(:)           ! NoLoadStep or NoLSElem(for TAPFEM)

real   , allocatable :: LoadStep_Time(:)        ! NoLoadStep
real   , allocatable :: LoadStep_dTime(:)       ! NoLoadStep
integer, allocatable :: LoadStep_Order(:)       ! NoLoadStep
```

1.3 モジュール構造

全体システムの現時点での各モジュールのツリー構造を図 2 に示す。システムの頂点に位置するメインモジュール CUMuLOUS の元に，各種ソルバーモジュール，疎行列を扱うモジュール，要素モジュールなどが配置され，さらにその元には基本的なデータ処理や演算を行うモジュールが配置される。各モジュールは独立にバージョンが管理される。

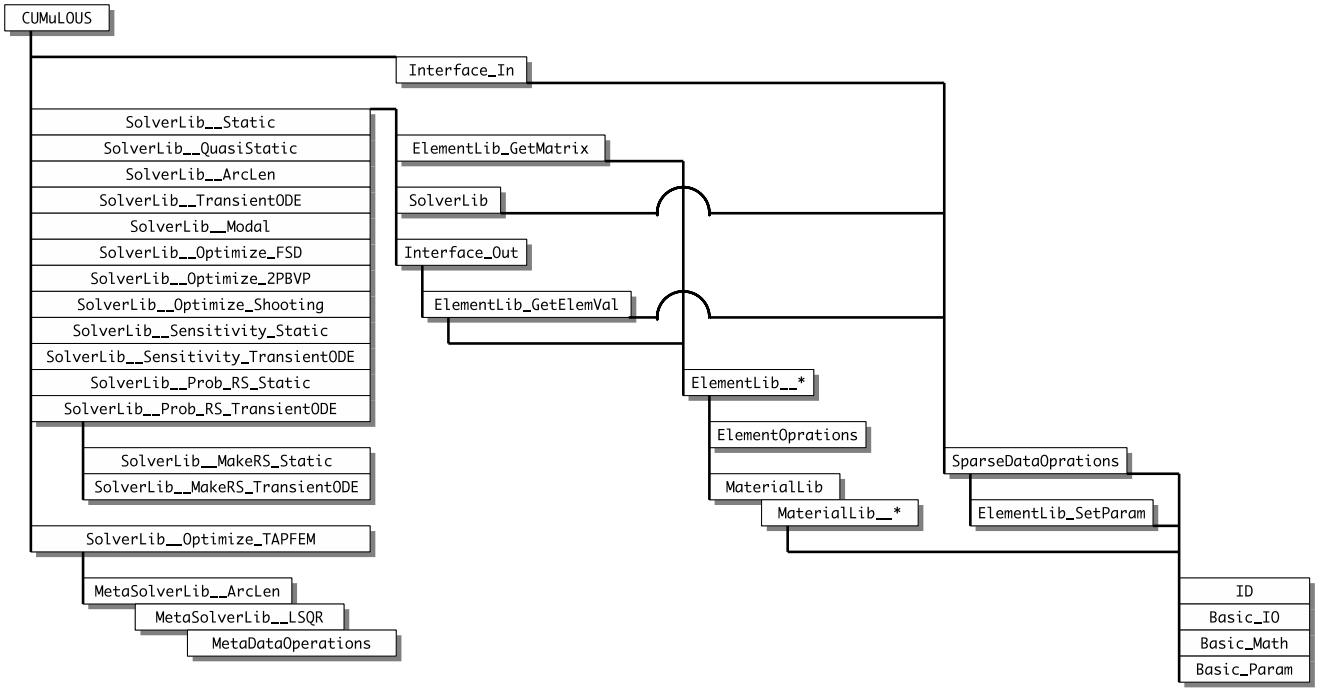

Fig.2　CUMuLOUS システムモジュールのツリー構造の概略図

以下，実装された各モジュールの概要を示す。各モジュールの詳細は各節の名称を用い，

[**名称**].f90

によりそれぞれのソースコードとコメントを含むファイルを参照できる。各名称において，アスタリスクはワイルドカードである。

■ID

ソルバー，要素，材料，変数等の識別番号を整数で定義するモジュール。新しい要素，材料のモジュールを定義した際にはこのモジュールに適宜 ID を割り振り，再コンパイルする。

■Basic_Param

グローバル定数／変数を定義するモジュール。本システムは，関係する変数を関数の引数として渡す事を基本とするが， 例外的にグローバル定数／変数を必要とするときにこのモジュールに追加する。

■Basic_Math

基本的な演算を行う関数のモジュール。

■Basic_IO

基本的なファイル入出力および標準入出力のモジュール。

■Interface_In

CUMuLOUS 各種ファイル入力を扱うためのモジュール。

■Interface_Out

解析結果等をファイルに出力するモジュール。

Basic_Param モジュールで定義される print_level(:) の値に応じて，要素モジュール／ソルバーモジュール／上位モジュール／結果出力それぞれのファイルが生成される。解析結果出力は，節点／要素／時系列／ Gmsh ポストスクリプトデータの最大 4 種類である。

■SparseDataOperations

疎行列データを扱うためのモジュール。

■ElemLib_SetParam

各要素モジュールで使用する基本的な設定値（使用する積分点，形状関数，各節点に格納される変数の種類等）を記述するモジュール。ユーザが要素を追加する際には，このモジュールにも変更を加える。

■ElemLib_GetMatrix

上位のソルバーからコールされ，各要素マトリクスを出力する。基本的な手順としては，各要素モジュールを呼び出す際の各節点の引数を定義し，各要素モジュールのサブルーチンを呼び出す。

■ElemLib__*

各有限要素を記述するモジュール。ユーザーは，所望の機能，つまり拘束条件を持つ「要素」をモジュールとして追加する事ができる。

各要素モジュールの詳細は後に公開予定の Theoretical manual に譲る事とし，ここでは各要素モジュールの名称の

みを記述する。現時点では，その詳細は各要素モジュールのソースコードのコメントを参照されたい。

なお先頭の整数は， ID モジュールで定義されている要素タイプ番号である。同じ要素モジュールを用いても，節点数などにより要素タイプ番号が異なる。また，行末の * は現時点で非公開であり，機能が省かれている事を示す。

00001 : **集中熱容量**

00002-00003 : 1 **次元伝熱要素**

00003-00008 : 2 **次元伝熱ソリッド要素**
00008-00012 : 3 **次元伝熱ソリッド要素**

00013-00016 : 3 **次元伝熱メンブレン要素**

00100 : 1 **次元熱流束境界条件要素**
00101-00102 : 2 **次元熱流束境界条件要素**
00103-00107 : 3 **次元熱流束境界条件要素**

01001 : p **法** 1 **次元伝熱要素**

01002-01003 : p **法** 2 **次元伝熱シェル要素**
01004-01007 : p **法** 3 **次元伝熱シェル要素**

02001 : p **法** 1 **次元アブレーション伝熱要素（*）**
02002 : p **法** 1 **次元溶融要素（*）**
02003 : p **法** 2 **次元アブレーション伝熱シェル要素（*）**
02004 : p **法** 2 **次元溶融シェル要素（*）**
02005-02006 : p **法** 3 **次元アブレーション伝熱シェル要素（*）**
02007-02008 : p **法** 3 **次元溶融要素シェル要素（*）**
02009 : p **法** 2 **次元アクティブクーリング伝熱シェル要素（*）**
02010-02011 : p **法** 3 **次元アクティブクーリング伝熱シェル要素（*）**

02021 : 1 **次元内部輻射要素（**2 **流束法および** Discrete originate **法）**

02022-02023 : 1 **次元内部輻射/伝熱連成要素（**Spherical harmonics P1 **法）**
02024-02028 : 2 **次元内部輻射/伝熱連成要素（**Spherical harmonics P1 **法）**
02029-02032 : 3 **次元内部輻射/伝熱連成要素（**Spherical harmonics P1 **法）**

02100 : 1 **次元輻射強度境界条件要素（要素タイプ** 02021 **と共に使用）**

02101 : 1 **次元輻射強度境界条件要素（要素タイプ** 02022-02023 **と共に使用）**
02102-02103 : 2 **次元輻射強度境界条件要素（要素タイプ** 02024-02028 **と共に使用）**
02104-02108 : 3 **次元輻射強度境界条件要素（要素タイプ** 02029-02032 **と共に使用）**

02210 : 1 **次元アブレーション境界条件要素（要素タイプ** 02001 **と共に使用）**

00050 : **集中構造質量要素**
00051 : 2 **次元ロッド要素**
00052 : 3 **次元ロッド要素**
00053-00057 : 2 **次元構造ソリッド要素**

00058-00061 : 3 **次元構造ソリッド要素**
00062 : 2 **次元ビーム要素**
00063 : 3 **次元ビーム要素**

00064-00065 : 3 **次元構造メンブレン要素**
00066-00067 : 3 **次元構造シェル要素**

00150 : **集中力要素**
00151 : **集中モーメント要素**

00161-00162 : 2 **次元圧力境界条件要素**
00163-00167 : 3 **次元圧力境界条件要素**

00181-00182 : 2 **次元分布質量要素**
00183-00187 : 3 **次元分布質量要素**

02502 : 2 **次元飛行力学要素**
02503 : 3 **次元飛行力学要素**

02504 : 2 **次元軌道要素**
02505 : 3 **次元軌道要素**
02549 : **万有引力要素**

04001 : 1 **次元リミット要素**

05000 : **微分要素**
05001 : **微分要素（**MinMax **問題用）**

10000 : **時間軸要素（*）**

■MaterialLib

上位のソルバーモジュール（後述）から呼び出され，与えられたパラメータに対して，要求された非線形材料特性を返すモジュール。

■MaterialLib__*

各材料特性に対応した関数を記述するモジュール。上位モジュール MaterialLib から呼び出される。

本システムの特徴の一つはこのように材料特性も関数で記述する点である。コンパイルする手順がある一方で，ユーザは自身が持つ複雑な材料特性などを用いた解析を行う事が可能となる。これはオープンソースのソフトウェアであるために容易に実現可能な事である。

もし与えられた表形式のデータを補間することにより材料特性を定義したい場合は別途そのようなモジュールを作成する事で実現できる。

■SolverLib

上位のソルバーモジュールから呼び出され，線形連立 1 次方程式のソルバー等を提供する。問題の種類に応じて，CUMuLOUS では複数の解法に対応したソルバーモジュールが用意されている。本システムでは，大規模問題および並列化への対応の点から，先述のように現時点では基本的に反復解法のみを用いる。

現バージョンにおいて対応している一次形式を解くソルバーは以下の通り。

- CG
- BiCGStab
- BiCRSafe

これらのソルバーには前処理行列が適用される。ユーザーは，入力ファイルの一つ CUMuLOUS_SolverParam.inp において，モジュール Basic_Param の変数 preconditionerType を以下のように整数で記す事で，その値に対応する前処理行列を指定できる。

- 0: 前処理行列なし
- 1: 対角行列の逆行列（0 の対角要素がある場合には自動的に 1 に置き換えられる）
- 2: 各節点に対応するブロック対角行列の逆行列

また，モード解析で必要になる実数固有値問題のために，Subspace 法も実装されている。

■SolverLib_*

下位モジュール SolverLib を利用し，各種解析に対応したソルバーモジュール。各解法の例が最終章の解析例で紹介されている場合は，この CD-ROM の/CUMuLOUS/manual/tutorial ディレクトリに入力ファイルの解説がある。

■SolverLib_Static

非線形定常問題を解くモジュール。

■SolverLib_QuasiStatic

構造モデルに対する非線形準定常問題を解くモジュール。ユーザーは，モデルに働く力または圧力を入力ファイル CUMuLOUS_SolverParam.inp に記述する事で，それらの境界条件と釣り合う慣性力を自動的に算出する。このモジュールは，航空機全機等の支持されていない構造問題に使用する。そのため，使用する要素は構造要素に限られる。

使用上の注意としては，系が特異にならないよう，2 次元では 3 自由度，3 次元では 6 自由度の拘束をかけておく必要がある。（解析結果は，それらの拘束に対する反力は 0 となる。）

■SolverLib_ArcLen

非線形性の強い定常問題に対し，Arc Length Method を用いて解くためのモジュール。

■SolverLib_TransientODE

非線形非定常問題を解くモジュール。

現バージョンでは，陰解法を用い，常微分方程式 $\mathbf{C}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{0}$ を解く。質量マトリクスを形式上使用しないため，構造問題に使用する場合には，要素マトリクス M は自動的に要素マトリクス C に組み込まれる。

ユーザーは，各荷重ステップの時間，荷重ステップ時間幅，荷重ステップ内での定数の補間次数を指定できる。

■SolverLib_Modal

サブスペース法を用いた，モード解析のためのモジュール。

■SolverLib_Sensitivity_Static

定常問題に対する感度解析モジュール。パラメータのノミナル値を中心として，ユーザーが指定した変化幅により数値差分を行う。

■SolverLib_Sensitivity_TransientODE

非定常問題に対する感度解析モジュール。前項の定常問題を非定常問題に拡張した物である。

■SolverLib__Optimize_FSD

Fully stressed design 法を用い，重量最適化問題を解くモジュール。現バージョンでは以下の制約がある。

- 静的解析および準静的解析に対応
- 各材料強度に対応した安全余裕のみを用いており，剪断／圧縮に対する安定性の安全余裕は考慮されていない

■SolverLib__Optimize_Shooting

Shooting 法を用いた非定常最適化問題を解くモジュール。他の非定常最適化問題解法（2PBVP, 時間軸有限要素）よりも高速だが，解ける問題は比較的単純な物に限られる。

■SolverLib__Prob_RS_Static

応答曲面法とモンテカルロ法を併用した，定常問題に対する確率解析モジュール。破壊確率が少ない領域を扱う場合，Importance sampling 法を使用しているため，少ない乱数生成で同じ確かさの解が得られる。

■SolverLib__Prob_RS_TransientODE

応答曲面法とモンテカルロ法を併用した，非定常問題に対する確率解析モジュール。前項の定常問題を非定常問題に拡張した物である。

■SolverLib__Optimize_2PBVP

Pierson のアルゴリズム [5] を用いた，二点境界値問題を解くためのモジュール。

■SolverLib__Optimize_TAPFEM

時間軸に p 法有限要素を用い，非定常構造モデルに対して以下の機能を持ったソルバーモジュール。（現バージョンでは，このソースコードは公開されておらず，したがってその機能も省かれている。）

- 任意の時点での拘束条件
- 時間軸有限要素を組み合わせる事により，複数の初期／終端条件，および周期的な問題を解く機能

この解法の特徴としては以下が上げられる。

- LSQR 法を二度用いた簡潔な Gradient/Correction procedure
- 非定常解析に対して時間方向の pFEM を適用。柔軟な動的解析精度および管理が可能。
- SQP に代表される方法をベースにした最適化と拘束条件を同時に解くソルバーに比べ，各イタレーションで現実的な解が得られる。

このアルゴリズムのため，別途，メタデータを扱うためのモジュール `MetaDataOperations` と，そのメタデータを用いたソルバーモジュール `MetaSolverLib_*` が用意されている。

詳細については文献 [6] を参照されたい。

2 解析システムの使用法

2.1 コンパイル方法

本システムは，Ubuntu linux 8.04 上の GNU Fortran compiler 4.2.4 および Intel Fortran compiler 10.1 の 2 コンパイラでそれぞれ動作を確認している。本システムをコンパイルするには，ソースコードを展開したディレクトリにおいて，以下のコマンドをタイプする。

GNU Fortran compiler **の場合**

```
$ ./gcc/make_cumulous_all
```

Intel Fortran compiler **の場合**

```
$ ./ifc/make_cumulous_all
```

その結果，同じディレクトリに実行ファイル cumulous が生成される。

2.2 入力ファイル

本システムが使用する共通の入力ファイルを以下に示す。これらのファイルはすべて ASCII テキスト形式であり，CUMuLOUS_*.inp の名称となる。

CUMuLOUS_ControlParam.inp	: **解析の基本的な条件を定義。最初に読み込まれる。**
CUMuLOUS_ElemType.inp	: **要素タイプ**
CUMuLOUS_ElemOption.inp	: **要素オプション**
CUMuLOUS_NodalValue.inp	: **節点に格納される各変数**
CUMuLOUS_NodeConnection.inp	: **各要素の節点の結合**
CUMuLOUS_FixedVar.inp	: **固定境界条件の定義**
CUMuLOUS_SleepElem.inp	: **（各荷重ステップにおいて）使用しない要素の定義**
CUMuLOUS_SolverParam.inp	: **使用するソルバーのパラメータを規定**

これらの詳細および各解析モジュールに対応して適宜要求される入力ファイルの解説は，付録の解析例で示す CD-ROM 内の /CUMuLOUS/manual/tutorial ディレクトリを参照されたい。

2.2.1 Gmsh[7] による CUMuLOUS 用プリ／ポストプロセッサ

本システムはソルバー機能に特化しており，ユーザーが適宜プリ／ポストプロセッサを選んでこのソルバーと連携して利用する事を想定している。

しかしながら，現時点では最低限の機能として，Gmsh をポストプロセッサとして使用するための出力サブルーチンが CUMuLOUS 内に実装されている。

またプリプロセッサとしては，Gmsh の形状記述ファイル*.geo とメッシュ情報記述ファイル*.msh から CUMuLOUS 入力ファイルを出力するための python[8] で記述されたスクリプト GCPP，および航空宇宙機構造用の階層オブジェクト型プリプロセッサ Osh（同じく python で開発，現時点で非公開）が実装されている。

GCPP は Python バージョン 2.5 で動作が確認されている。対象となる解析手法は，現時点では Solve_Static，Solver_CK，Solver_Modal である。スクリプト本体および解説は CD-ROM 内の /GCPP ディレクトリに格納されている。

後述の解析例の中の GCPP を利用した例もあわせて参照されたい。

2.3 実行方法

先述の実行形式 cumulous を必要なところに移動し，そのファイルへのパスを通せば，本システムは，必要な入力ファイルを ASCII テキスト形式で記述した後に

```
$ cumulous
```

とプロンプトから入力することによりシステムが起動する。各入力ファイルはコマンドを入力したカレントディレクトリから読み出される。

2.4 出力ファイル

基本的な出力ファイルは以下の 4 つである。各ソルバーに応じた追加の出力ファイルについては本 CD-ROM の /tutorial フォルダに記述がある。

result_nodalvalue.dat	: **各節点の変数**
result_elemvalue.dat	: **各要素の変数**

result_timehist.dat : **ユーザが指定した変数についての時系列出力データ**
result_gmsh.pos : Gmsh **ポストプロセッシングのためのテキストファイル**

先述のように，Gmsh をポストプロセッサとして使用するための出力サブルーチンがシステム内に記述されている。ポストプロセッシングのための Gmsh 入力ファイル result_gmsh.msh を生成するためには，CUMuLOUS_ControlParam.inp において，print_level の 4 番目のパラメータを 2 以上に記述する。

print_level の設定に応じて他の標準出力およびファイル出力内容が変更される。print_level の詳細については /tutorial/static/ を参照されたい。

2.5 付録 1: CUMuLOUS ソースコード

本解析システムのうち，基本的な機能に限定されたソースコードは添付の CD-ROM の /CUMuLOUS/src/ ディレクトリに格納されている。

2.6 付録 2 : GCPP ソースコード

プリポストプロセッサ Gmsh の形状データ／メッシュデータを CUMuLOUS インプットデータに変換するスクリプト GCPP.py および基本的な演算を行う BasicProc.py は添付 CD-ROM の /GCPP/ ディレクトリに格納されている。

添付の gcpp.sh は，bash 上で

```
$ gcpp
```

とタイプすれば自動的に GCPP.py を呼び出して CUMuLOUS 入力ファイルを生成する。そのためには，環境変数 PYTHONPATH にスクリプト GCPP.py へのパスを追加する。（例）

```
PYTHONPATH=~/Codes/python/GCPP/
```

2.7 付録 3 : 例題

本節では，例題を通して本システムの実際の使用法について紹介する。各ソルバーの解説と実際の入力ファイル，そして解析モデルについてはそれぞれの項にパスが表示されているため，その詳細を参照されたい。各例題は，添付の圧縮ファイルを解凍した先のオンラインマニュアル (/CUMuLOUS/manual/index.html) から参照できるが，各項目には直接のパスも記す。

Gmsh 入出力ファイルが用意されている場合には，各解析例の入力ファイルと同じディレクトリ内に以下のファイルが用意されている。

GCPP.inp : GCPP **スクリプト用入力ファイル**
untitled.geo : Gmsh **形状定義ファイル**
untitled.msh : Gmsh **メッシュファイル**
result_gmsh.pos : Gmsh **ポストプロッセシング用入力ファイル**

2.7.1 加熱を受ける二次元要素の熱応力解析

/CUMuLOUS/manual/tutorial/static/static.html

2.7.2 トラス構造の準静的解析

/CUMuLOUS/manual/tutorial/quasistatic/quasistatic.html

2.7.3 トラス構造の非定常解析

/CUMuLOUS/manual/tutorial/transient/structural_truss/structural_truss.html

2.7.4 三次元構造の非線形非定常伝熱解析

/CUMuLOUS/manual/tutorial/transient/thermal_3dim/thermal_3dim.html

2.7.5 二次元構造の振動解析

/CUMuLOUS/manual/tutorial/modal/modal.html

2.7.6 一次元熱伝導問題の感度解析

/CUMuLOUS/manual/tutorial/sensitivity/sensitivity.html

2.7.7 一次元熱伝導問題の確率解析

/CUMuLOUS/manual/tutorial/probabilistic/probabilistic.html

2.7.8 FSD による最適化問題

/CUMuLOUS/manual/tutorial/optimization/FSD/FSD.html

2.7.9 トラス構造の二点境界値最適制御問題

/CUMuLOUS/manual/tutorial/optimization/2PBVP/structural_truss/structural_truss.html

2.7.10 滑空機の二次元運動の二点境界値最適制御問題

/CUMuLOUS/manual/tutorial/optimization/2PBVP/dynamics2D/dynamics2D.html

参考文献

[1] G. L. Brauer, D. E. Cornick, and R. Stevenson, Capabilities and of the Program Applications to Optimize Simulated Trajectories (POST) *NASA CR-2770*, 1977.

[2] Hargraves, C. R. and Paris, S. W., Direct Trajectory Optimization Using Nonlinear Programming and Collocation, *J. Guidance*, Vol.10, No.4, pp.338–342, 1987.

[3] http://otis.grc.nasa.gov/

[4] Bless, Robert R.; Queen, Eric M.; Cavanaugh, Michael D.; Wetzel, Todd A.; Moerder and Daniel D., Variational Trajectory Optimization Tool Set: Technical description and user's manual, *NASA-TM-4442*, 1993.

[5] Willoughby, J. K. and Pierson, B. L., A Constraint-Space Conjugate Gradient Method for Function Minimization and Optimal Control Problems, *Int. J. Control*, Vol.14, No.6, pp.1121–1135, 1971.

[6] 高嵜　浩一, An Optimization Code for Nonlinear Transient Problems of a Large Scale Multidisciplinary Mathematical Model, *Preprints of 26th International Symposium on Space Technology and Science (ISTS)*, 2008.
http://archive.ists.or.jp/upload_pdf/2008-c-17.pdf

[7] http://www.geuz.org/gmsh/

[8] http://www.python.jp/Zope/