ADVENTURE SYSTEM

ADVENTURE_Solid

Static elastic / elastic-plastic / large deformation stress analysis with HDDM solver / parallel CG solver

Version: 1.1

プログラム使用マニュアル

September 2003

ADVENTURE Project

目 次

1 概要

このドキュメントは ADVENTURE Project [1] において開発された、固体静解析のための有限要素法解析ソルバ ADVENTURE_Solid の使用マニュアルである。階層型領域分割法 (Hierarchical Domain Decomposition Method, 以下 HDDM) [2, 3, 4] に基づいた線形方程式ソルバを採用しており、並列計算機環境に対応することで大規模な解析を可能としている。ADVENTURE_Solid は以下のような特徴を持っている。

- 線形方程式ソルバとして HDDM、並列 CG 法および BDD (Balancing Domain Decomposition) 前処理つき HDDM [5, 6, 7] の 3 種類が使用可能である。
- HDDM ソルバによる並列実行では、動的な負荷分散が可能である。
- 解析種類は弾性/弾塑性/幾何学的非線形応力解析。
- 弾塑性/幾何学的非線形解析は荷重/変位制御による増分法。
- 非線形解析には、後退型 Euler 法による応力積分とコンシステント接線剛性を使用。
- 4 面体、6 面体のそれぞれ 1 次、2 次ソリッド要素に対応。
- 対応プラットフォームは Unix, Linux。
- 並列処理ライブラリには MPI [8] を使用し、MPP や ネットワークにより接続された PC または ワークステーションといった多様な並列環境に対応する。もちろん単一プロセッサ上での実行も可能である。

ADVENTURE システムにおいて ADVENTURE_Solid およびその前後における処理の流れは図 1 のようになっている。

(1) メッシュデータの作成 (ADVENTURE_TetMesh)

ADVENTURE_TetMesh を用い、解析対象に対してメッシュ分割を行う。他のメッシュ分割プログラムや、手動で作成したメッシュも、フォーマットを変換することで取り込むことが可能。

(2) 境界条件の設定 (ADVENTURE_BCtool)

ADVENTURE_BCtool を用い、解析対象のメッシュに対して境界条件を付加する。物性値の設定もここで行う。

(3) 領域分割 (ADVENTURE_Metis)

ADVENTURE_Metis を用い、一体型の解析モデルより階層型に領域分割された解析モデルを作成する。並列処理が可能である。

(4) FEM 解析 (ADVENTURE_Solid)

領域分割された解析モデルを入力として、ADVENTURE_Solid により有限要素法解析を行う。並列処理が可能である。

(5) ポストシステム (ADVENTURE_Visual)

ADVENTURE_Visual を用いて解析結果を可視化する。並列処理が可能である。

なお、動作環境は Linux および Unix であり、並列計算には MPI が必要である。MPI には種々の実装が存在するが、フリーなものとしては MPICH [9] 等がある。MPICH は非常に多くの環境をサポートしているため、MPI が用意されていない環境でも多くの場合これをインストールすることで並列版の ADVENTURE_Solid を使用出来るであろう。

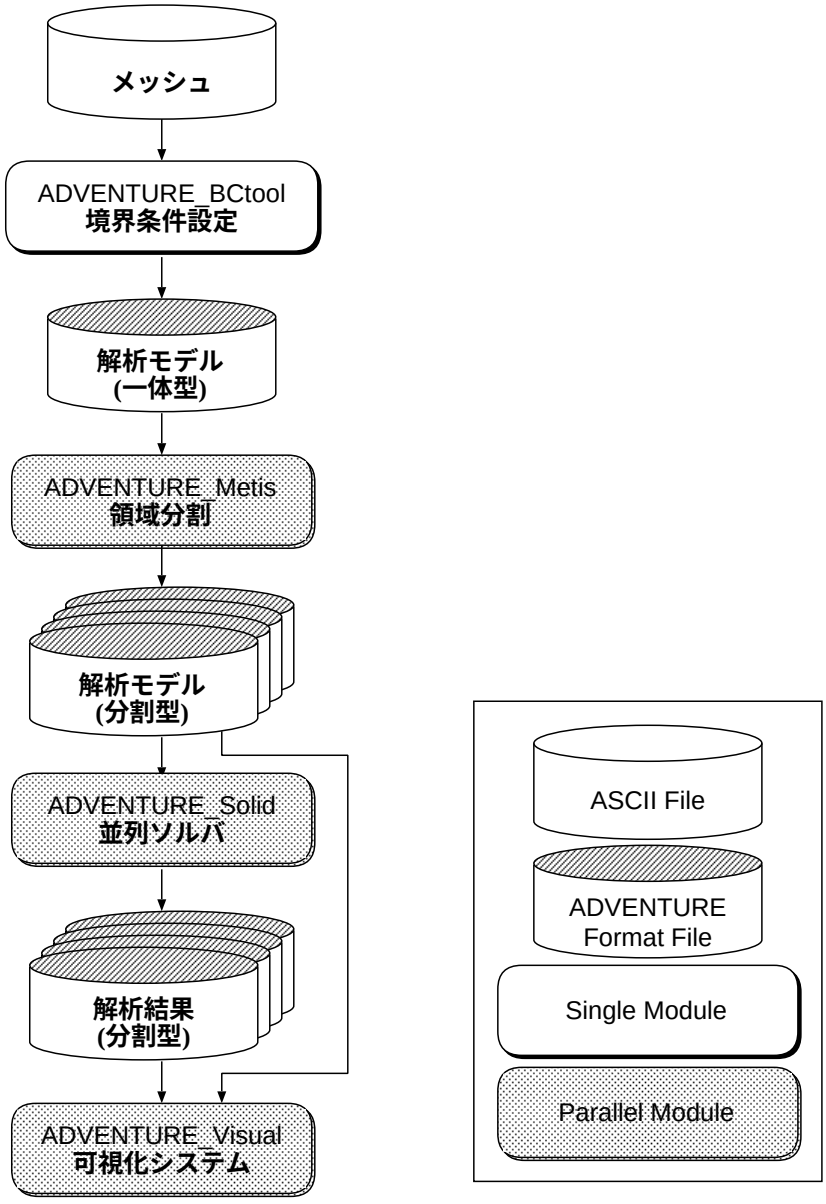


図 1: 全体の流れの概略図

2　線形方程式ソルバと並列方法

2.1　線形方程式ソルバ

ADVENTURE_Solid では、以下の 3 つの線形方程式ソルバを使用することが可能である。

- HDDM ソルバ
  各部分領域内部については直接法で、部分領域間の境界上の自由度については CG 法で解く手法である。また、CG 法の前処理としては対角スケーリングを使用している。

- CG ソルバ
  部分領域内部および領域間境界によらず、全ての自由度に対して CG 法を用いて解く。前処理には対角スケーリングを用いている。

- BDD ソルバ
  HDDM ソルバに対して強力な前処理手法であるバランシング領域分割法 (BDD) を採り入れた解法である。部分領域前処理として Neumann 型前処理を行なう BDD ソルバ、対角スケーリングを行なう BDD-DIAG ソルバとがある。以下では、HDDM ソルバや CG ソルバと比較する場合は両者を合わせて BDD ソルバと示すことがある。

2.2　並列方法

ADVENTURE_Solid では、階層型領域分割法を用いることで並列処理を可能としている。解析領域の階層型への分割を模式的に図示したものが、図 2 である。一階層目の大きな分割単位を”部分”(Part) とし、二階層目の細かい分割単位を”部分領域”(Subdomain) と呼ぶことにする。また、以下特に断らずに単に領域とした場合は、細かい分割単位の方の部分領域を指すことにする。

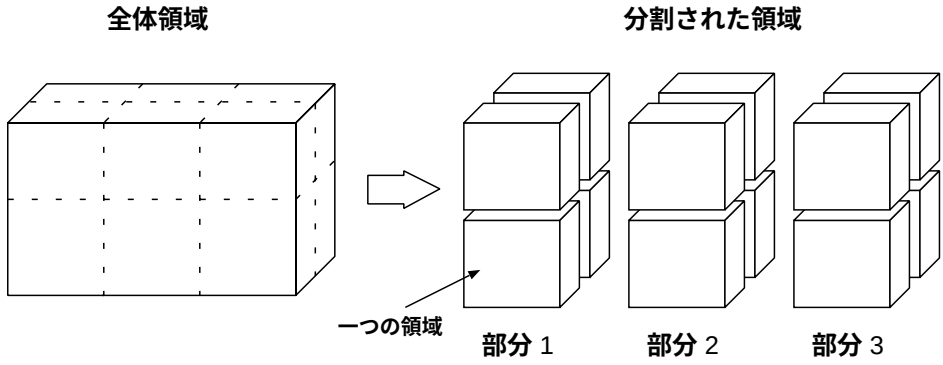


図 2: 階層型領域分割

並列方式	コマンド名	対応ソルバ	領域分割における部分数
シングル版	`advsolid-s`	HDDM, CG, BDD	任意
静的付加分散版	`advsolid-p`	HDDM, CG, BDD	部分数 = 使用ノード数
動的付加分散版	`advsolid-h`	HDDM, BDD	部分数 = 親プロセス数

表 1: 並列方法と対応ソルバ

領域分割は ADVENTURE_Solid の実行に先だって ADVENTURE_Metis により行うが、ADVENTURE_Solid ではこれらの分割された領域の各ノード (プロセス) への割り当て方に対して複数の方法が用意されており、それぞれに適した領域分割を行う必要がある。また、使用するメモリ量や計算時間は分割の仕方および使用する線形方程式ソルバに大きく依存するため、これらのことを考慮して領域分割しておく必要がある。

なお、ADVENTURE_Solid は 並列ライブラリとして MPI を用いており、起動時にはユーザーの指定に応じて複数のプロセス (環境によってはスレッド) が起動される。1 ノード (CPU) あたり 1 プロセスを起動するのが一般的であるため、以下では分かりやすさのためプロセス、ノード、CPU といった言葉は特に区別せず用いている。ただし、1 ノードに対して複数のプロセスを割り当てることももちろん可能である。

ADVENTURE_Solid には並列方法の違いにより、3 つの実行バイナリが用意されている。また、線形方程式ソルバには HDDM, CG, BDD の 3 種類が用意されているが、実行バイナリ毎に使用可能なソルバには違いがある。各並列方法における使用可能な線形方程式ソルバおよび領域分割の仕方は以下のようになっている。

(1) シングル版 (`advsolid-s`)

並列計算は行わず、全ての計算をひとつのプロセスとして実行する。コンパイルおよび実行ともに MPI なしで可能である。部分数、領域数に関する制限はなく、並列用に領域分割した解析モデルをそのまま使用して実行できる (図 3)。基本的に、次の静的負荷分散版において各部分に対して並列実行される計算を、1 プロセス内で順に行うのと同じである。

これを用いることで並列環境が無い環境においても実行可能である。また、並列用に領域分割したモデルを変更せずそのままの分割で実行できるため、並列計算がうまく行かない場合でのチェックのために用いることが出来る。

使用可能な線形方程式ソルバは、HDDM、CG および BDD である。

(2) 静的負荷分散版 (`advsolid-p`)

図 4 に示すように、一つの部分に対する計算を一つのプロセスに静的に割り当てることで並列に計算を行う。次の親子型の動的負荷分散において子に割り振っている仕事を、親が自分で順に行うのと同じである。領域分割における部分数と実行プロセス数が同じあるため、静的負荷分散版で使用するプロセス数が部分数となるようあらかじめ分割しておく必要がある。使用するノード数を部分数として分割し、1 ノード 1 プロセスで実行するのがよいであろう。

次の動的負荷分散版と比べ通信量はかなり押えられるため、各ノードの性能が均質な並列計算環境においては、この静的負荷分散版が有効である。

使用可能な線形方程式ソルバは、HDDM、 CG、BDD である。

(3) 動的負荷分散版 (`advsolid-h`)

図 5 に示すように、各プロセスを各領域の計算を行う子と部分単位で取りまとめを行う親とに分け、子への領域の割り当てを動的に行うことで動的な負荷分散を図る並列方式である。各親には 1 つの部分が静的に割り当てられるため、全プロセスのうち部分数個が親となり、残りのプロセスが子となる。そのため、起動するプロセス数より少ない部分数で領域分割をしておく必要がある。子が多くなった場合、親が一つしかいないとその親に通信が集中してしまい効率が悪くなるため、親も並列化し親の処理も分散できるようになっている。

多くの計算は子が行うため、大部分のプロセスは子に割り当てる方が一般に効率よい計算が行える。例えば、10 プロセスで実行する場合、親すなわち部分数を 1 〜 2 程度にして領域分割しておくのが適当であろう。

使用可能な線形方程式ソルバは、HDDM、BDD である。

計算負荷のバランスを実行中に動的にとることが可能であるが、その反面上記の静的負荷分散版に比べ通信が多くなるため、均質な並列環境では静的負荷分散版の方が一般に効率よく実行できる。逆に、非均質な並列環境や、マシン環境自体は均質でも他に CPU を食うプロセスが走っていて実質的には非均質となるような場合では、この動的負荷分散版がより有利となる。

また、親と子の計算処理は時間的には基本的に重なっていないため、親と子を同じノード上で (異なるプロセスとして) 実行することも可能である。N_{part} を部分数、N_{proc} をプロセス数とすると、MPI の rank が 0 〜 $N_{\mathrm{part}}-1$ までのプロセスが親、rank が N_{part} 〜 $N_{\mathrm{proc}}-1$ の残りのプロセスが子となるようになっているので、それに注意してプロセスを分配すればよい。例えば使用するノード数を 8、親の数を 2 とし、子をノード数と等しい 8 プロセス用いるとすると、全部で 10 プロセスを起動することになる。この場合、例えば mpich の ch_p4 デバイスでは、

```
1  host0
2  host1
3  host0
4  host1
5  host2
6  host3
7  host4
8  host5
9  host6
10 host7
```

のような `machine_file` (付録 C 参照) を用いることで、初めの 2 行にあるホスト (host0, host1) で親を起動し、残りの 3 行目以降の 8 つのホスト (host0 〜 host 7)

で子を起動する。すなわち host0, host1 では親と子の両方、残りの host2 〜 host7 では子のみを起動することになる。ただし、この指定方法は MPI 実装系に依存するため、他の MPI や mpich の他のデバイスを使用の場合はそれぞれのマニュアルを参考にして頂きたい。また、MPI やマシン環境によっては同一ノード上での MPI による通信は遅い場合もあるため注意を要する。

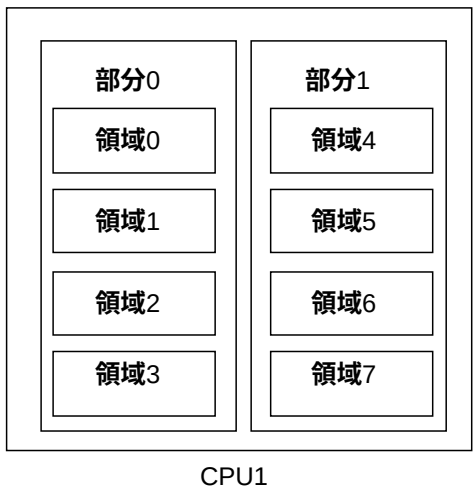


図 3: 領域の CPU への割り当て — シングル版

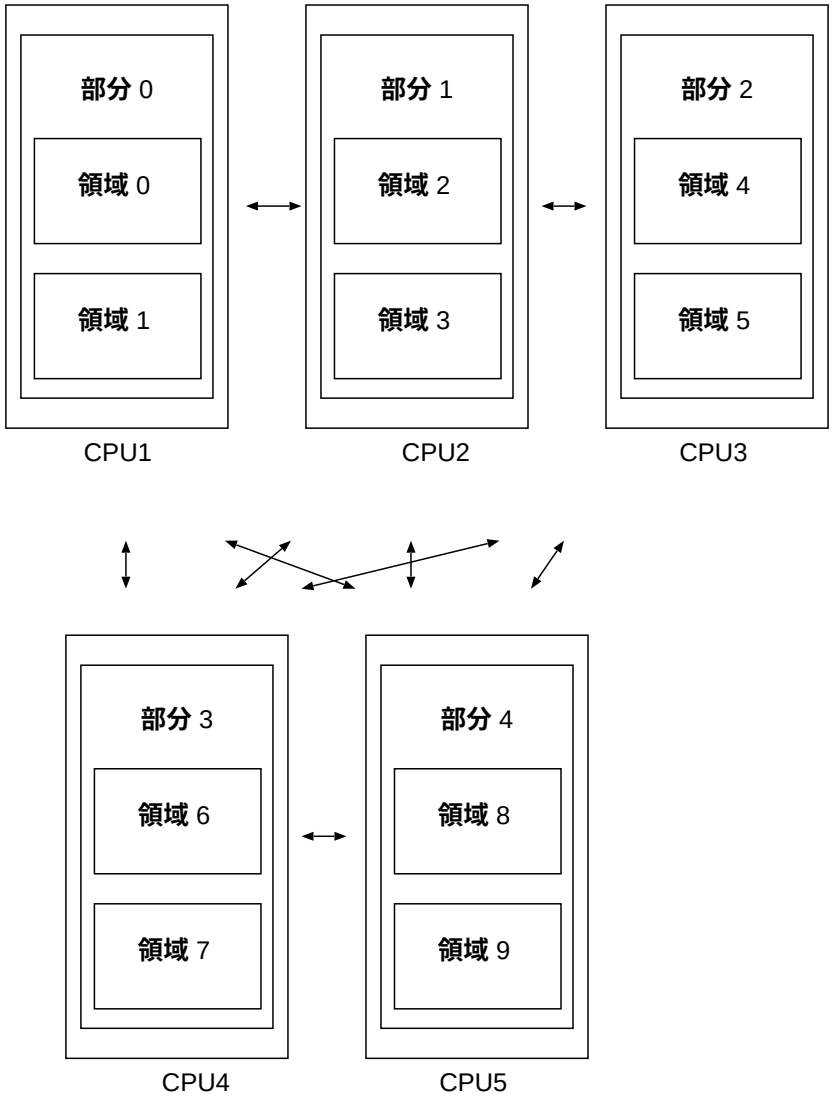


図 4: 領域の CPU への割り当て — 静的負荷分散版

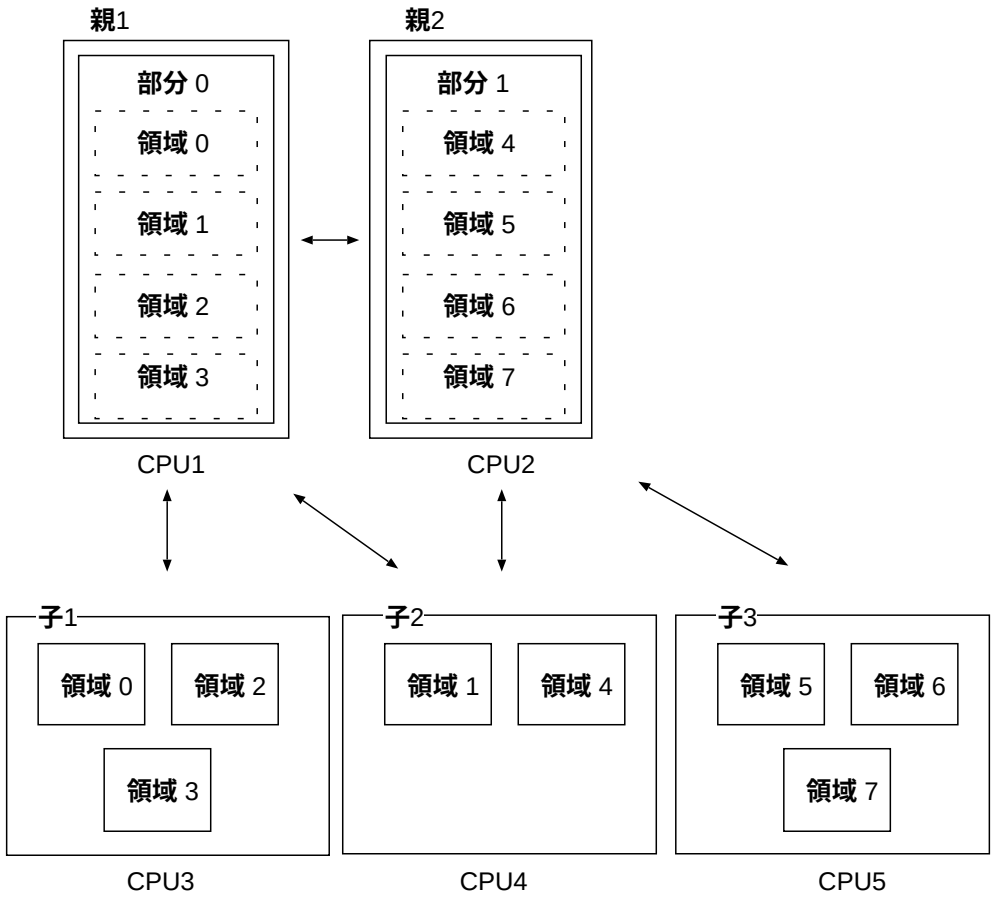


図 5: 領域の CPU への割り当て — 動的負荷分散版

2.3 各ソルバの特徴と領域分割方法

前節では、それぞれの並列方法において領域分割における部分数をどのようにすればよいかを述べたが、基本的に部分数は使用する並列方法といくつ使えるノードがあるかといった実行環境によって決めればよい。一方領域の最適な分割数の決め方はそれほど自明ではないが、使用するノードや親、子の割当が同じであっても領域数の違いは使用するメモリ量、計算時間に大きく影響するため、線形マトリックスソルバに応じて適切な分割を行う必要がある。

以下に示すそれぞれの線形マトリックスソルバの特徴を参考にし、領域分割を行なって頂きたい。

2.3.1 HDDM ソルバ

HDDM ソルバでの使用メモリ量は、それぞれの並列方法によらずほぼトータルの領域数によって決まる。これは、剛性マトリックス (およびその逆マトリックス) を各領域毎にスカイライン法にて記憶しており、これがメモリ使用量の大部分を占めているためである。基本的には粗く分割した方がメモリを多く使用し、細かく分割した方がメモリは少なくて済む。

次に計算時間であるが、まず静的負荷分散版について述べる。この場合、通信量は基本的に部分数すなわち使用ノード数で決まり部分内をどう領域に分割するかにはよらないため、ここでは通信速度に関して考慮する必要はない。

領域内部の自由度に関しては直接法、領域間の自由度に関しては反復法 (CG 法) で解く、ハイブリッド型の線形方程式ソルバを用いており、領域分割数がそれらの混ざる比率を支配するため、領域数が計算時間に大きく影響する。まず 1CG ステップあたりの計算時間に関しては、基本的に細かく分割したほうが直接法を使う比率が少ないため短くなる。一方同じ収束誤差を得るのに必要なステップ数は、直接法の比率が高い粗い分割の方が少なくなる。トータルの解析時間はこれらの掛け合わせであり自明でない。また収束に要するステップ数は境界条件にも依存し、Dirichlet 境界条件である変位境界条件が多いほど収束回数は少なく済むことがわかっている。経験的には Dirichlet 境界条件が多く設定されている場合は細かく分割する方がトータルの解析時間はおおむね少なくすむ。しかし Dirichlet 境界条件が少ないような場合では、分割を細かくするにつれ初めのうちは計算時間が減少するが、途中から逆に増大してしまうため、計算時間の最も短い分割は中間的なものとなる。

また ADVENTURE_Metis では、非常に細かい分割を行った場合、要素を一つも含まないような領域が作られてしまうことがあるが、ADVENTURE_Solid では実行時にそのような領域が見つかると、警告を出して終了するようになっている。これを避けるため、細かく分割する場合の下限として 1 領域あたりの要素数が 20 程度以上となるよう分割しておく必要がある。

結局、経験的には 1 領域あたりの要素数を 20 〜 100 程度にしておけば、ベストか、ベストに近い計算時間が得られるようである。この範囲のうち、変位境界条件が多ければ 1 領域あたりの要素数を少なめにし、少ない場合は多めに設定する。また、4 面体 1 次要素の様に 1 要素中の節点数の少ない要素では、多めにした方がよい。

ADVENTURE_Metis では、部分数 N_{part} (= プロセス数) と、1 部分あたりの領域数 N_{subdom} を指定して実行するため、全要素数を N_{element} とすると、1 領域あたりの要素数 n は

$$n = \frac{N_{\mathrm{element}}}{N_{\mathrm{part}} \times N_{\mathrm{subdom}}} \tag{1}$$

によって与えられる。この値が上に述べたように 20 〜 100 程度となるように分割するとよい。

動的負荷分散版の場合、静的負荷分散版と比べ親子間での通信が生じるが、この通信量は部分内をどのように分割するかに依存する。すなわち、親子間では領域自身や領域間内部境界上のデータを通信する必要があり、細かく分割すればそれだけ通信回数も増え、必要な通信量は増大することとなる。通信速度はそれぞれの並列環境により、通信量は解析モデルに大きく依存するため一概には言えないが、例えば 100Base の Ethernet で接続されたクラスタ環境では、この通信影響は無視できない程度となるであろう。そのため、一般的には上記の静的負荷分散版よりも粗い領域分割を行う方がよい。どの程度粗くするのがよいかは実行環境に大きく依存するため、それぞれの環境にて実際に試して頂きたい。

2.3.2 並列 CC ソルバ

部分内の領域数は 1 である必要があるため、領域分割時には部分内の領域数を 1 として分割しておく。

前処理が単純であるため、一般に多くの反復回数を必要とするが、他の方法と比べると、剛性マトリックスを記憶しないオプション使用時の HDDM ソルバを除き、必要となるメモリ量が最も少なくてすむ手法であり、剛性マトリックスを記憶する HDDM ソルバの約半分程となっている。また、通信量に関しても低く押えられている。

2.3.3 BDD ソルバ

(1) BDD 法

BDD 法とは、領域分割法に対する前処理として有名であった Neumann-Neumann 前処理を拡張した手法として提案された解析手法である。前処理操作としては、

- 部分領域前処理
- コーススペースによるコースグリッド修正

を行なう。ここで、コーススペースとは解析領域より自由度を低くした問題であり、コースグリッド修正とはその問題を解くことにより解の補正を行う方法である。ADVENTURE_Solid で対象としている固体解析では、コーススペースは各部分領域の剛体モードで構築されることが分かっており、部分領域数 × 剛体モードの 6 自由度をもつことになる。

BDD における前処理は、通常の CG 法 1 ステップの数倍のコストを要するが、反復回数を 10 分の 1 以下に抑えることができるため結果として解析の高速化を実現でき

る。特に、 CG 法が収束しにくい問題、例えば重力による体積力を考慮した問題などではその効果を発揮し、反復回数は 20〜30 分の 1 にまで抑えることができる。

(2) BDD 法の実装

ADVENTURE_Solid では、部分領域前処理法が異なる 2 種類の実装を行なっている。

- BDD ソルバ
  部分領域前処理として、オリジナル BDD と同様に Neumann-Neumann 前処理を行なう手法である。前処理効果が非常に高く、一般的に最も良い収束性を示す。しかし、使用メモリ量が大きく、CG 法 1 ステップにおける前処理コストも高くなるという短所がある。

  なお Neumann-Neumann 前処理時には非正則問題を解く必要があり、その解法には特異値分解による一般化逆行列、最小二乗解を求めるなどの手法があるが、これらは基本的に計算コストが高い。そのため ADVENTURE_Solid では、係数行列の対角成分に非負のパラメータ α を加えることで正則化し、その問題の解を近似解として用いる手法を採用している。この α として適切な値を選択しないと BDD の効果が薄れてしまうが、理論に基づいた最適なパラメータの選択方法はまだ示せていない。そこで経験的な指標となるが、およそ 1.0e-2〜1.0e-3 の値を選択するとベストに近い結果が得られるようである (デフォルト値は 1.0e-3)。
- BDD-DIAG ソルバ
  部分領域前処理として、対角スケーリング前処理を行なう手法である。BDD ソルバにおける短所を克服するため開発された手法であり、対角スケーリング前処理のためのコストは無視できる程度のため、使用メモリ量などは HDDM ソルバと同程度であるという特徴がある。一般的に前処理効果は BDD ソルバより劣るが、CG 法 1 ステップのコストが低くなるためトータルの計算時間に大きな差が生じることは少ない。

(3) BDD 法の並列化

BDD は 10 年ほど前から研究され始めているが、並列化の実現例は少ない。ADVENTURE_Solid では、BDD を HDDM と同様に階層型で管理を行うことで並列化を実現している。よって、 Neumannn-Neumann 前処理に必要な部分領域データは、解析に用いる剛性行列と同じ方法で作成、記憶を行っている。しかし、コースグリッド修正は全解析領域のデータによって構築される問題 (コース問題) を解くことになるため、階層型による管理は難しい。そこで、コース問題における係数行列 (コースマトリクス) を全プロセッサで分割し記憶することで並列化を実現している。ここで、コースグリッド修正は各 CG ステップで右辺ベクトルが変わるだけの問題を解くことになるため、CG 初期ステップにおいてコースマトリクスの並列 LU 分解を行い、それを記憶することで計算時間の短縮を行っている。なお、コースマトリクスにはスカイライン法に基づいた記憶方式を採用している。

ここで、コース問題の自由度数は部分領域数に関係するため、部分領域数が増加する大規模自由度問題ではコースマトリクスの LU 分解コストは非常に大きくなる。

そこで、ADVENTURE_Solid ではコースマトリクスを分割記憶する際に意図的にデータを欠落させる (不完全コースマトリクス) ことで、計算量の軽減、並列性の向上、使用メモリ量の軽減を実現できる手法が選択できる (ただし、シングル版を除く)。前処理効果は劣ってしまうが、CG 法 1 ステップのコストが低くなり、トータルの計算時間で上回る場合もある。特に問題規模が大きい場合や使用プロセッサ数が多い場合は計算速度が飛躍的な向上が期待できる。なお、ここでのデータ欠落は解析解の精度には影響しないことが経験的に分かっている。

これにより、ADVENTURE_Solid における BDD 法としては、

ソルバ	実行時オプション
BDD	`-solver bdd`
BDD + 不完全コースマトリクス	`-solver bdd -iLU`
BDD-DIAG	`-solver bdd-diag`
BDD-DIAG + 不完全コースマトリクス	`-solver bdd-diag -iLU`

が選択できることになる。それぞれの特徴は一般的に、BDD は 使用メモリ量が大きいが収束性が非常に良い、BDD-DIAG は使用メモリ量が HDDM と同程度で収束性が良く、不完全コースマトリクスを使用すると収束性が悪くなるが使用メモリ量が小さく大規模問題では計算速度の向上が期待できるものであり、問題設定や使用プロセッサ数によって使い分けることになる。なお、最も使用メモリ量が小さいのは「BDD-DIAG+不完全コースマトリクス」である。

(4) BDD における領域分割数

BDD は基本的に HDDM への前処理として搭載しているため、2.3.1 節で紹介している領域分割数の決め方と大きく異なる点はない。しかし、コースグリッド修正におけるコーススペースの自由度が部分領域数 × 6(剛体モードの自由度) となることから、あまり細かい分割を行うとコースマトリクスの LU 分解にかかるコストが増大してしまう。そのため一概には言えないが、BDD を指定しない場合の最適分割数よりも若干粗い分割を行う方がよい。経験的には、静的負荷分散版で 1 領域の要素数が 60〜120 程度、動的負荷分散で 150〜250 程度にしておけば、ベストかそれに近い計算時間が得られるようである。なお、使用メモリの観点では、BDD ソルバでは HDDM ソルバに比べて約 1.5〜2 倍のメモリ、BDD-DIAG ソルバでは約 1.2 〜 1.5 倍のメモリが必要となるため、メモリ領域が不足しそうな場合はさらに細かい分割を行う必要がある。

3　解析機能

ADVENTURE_Solid は、材料特性として弾性解析、弾塑性解析、また幾何学的非線形性として、大変位、大ひずみを取り扱うことが出来る。これらの可能な組合せは以下のようになっている。

- 線形弾性解析
- 弾性大変位微小歪み解析 (Total Lagrange 法)
- 弾性大変位大歪み解析 (Updated Lagrange 法)
- 弾塑性解析
- 弾塑性大変位微小歪み解析 (Total Lagrange 法)
- 弾塑性大変位大歪み解析 (Updated Lagrange 法)

これらの解析における共通の機能を挙げておく。

3.1　全解析における共通機能

3.1.1　並列処理機能

シングル版を含め以下の 3 パターンが可能であり、環境に応じて使用できる。

- シングル CPU での実行
- 静的な負荷分散による並列処理
- 動的な負荷分散による並列処理

また、全ての線形方程式ソルバが使用可能である。

3.1.2　要素

使用できる要素は以下の 4 種類のソリッド要素である (付録 A 参照)。ただし、 要素の混在には対応していないため、全て同一の要素を用いてモデルを作成する必要がある。

- 4 面体 1 次要素 (1 積分点)
- 4 面体 2 次要素 (4 または 5 積分点)
- 6 面体 1 次要素 (8 積分点、オプションで体積歪みまたはせん断歪みに関する次数低減積分)
- 6 面体 2 次要素 (27 積分点)

3.1.3 境界条件

付加できる境界条件は以下のものである。

- 節点強制変位
- 節点集中荷重 (面荷重は ADVENTURE_BCtool にて節点集中荷重に変換する)

3.1.4 体積力

重力による自重を付加可能である。

3.2 線形弾性解析機能

3.2.1 熱応力解析

熱膨張を考慮した解析を行なう。線膨張係数、参照温度、各節点での温度を入力ファイル中で指定しておく必要がある。

3.2.2 材料モデリング

等方的な材料物性に対応し、以下の材料物性値が使用できる。

- ヤング率
- ポアソン比
- 質量密度 (自重負荷時に使用)
- 線膨張係数 (熱応力解析時に使用)
- 参照温度 (熱応力解析時に使用)

複数の材料を含む解析にも対応しており、その場合は各材料毎に上記の物性値を指定する。

3.2.3 解析結果出力

以下の量が解析結果として出力可能であり、実行時に選択出来る。

- 変位 (節点)
- 反力 (節点)
- 応力テンソル (要素/積分点/節点)
- 相当応力 (要素/積分点/節点)

- ひずみテンソル (要素/積分点/節点)
- 主応力とその方向ベクトル (要素/積分点/節点)
- 主ひずみとその方向ベクトル (要素/積分点/節点)

3.3 非線形解析機能

線形弾性解析以外の非線形解析においては、ひずみ増分理論による荷重および変位を増分とした増分解析を行っている。解析方法の定式化に関しては、[4, 10] を参照されたい。

大まかな処理の流れは図 6 に示すように、大きく 3 重のループとなっている。最も外側のループは増分ステップのループであり、増分ステップの内側では Consistent 接線剛性を用いた Newton-Raphson 法による反復を行っている。これにより、比較的大きな増分ステップをとることが可能となっている。各 Newton-Raphson ループの内側では階層型領域分割法における CG 法の反復を行っている。

なお、熱応力解析については線形弾性解析でのみの対応であり、非線形解析では未対応となっている。

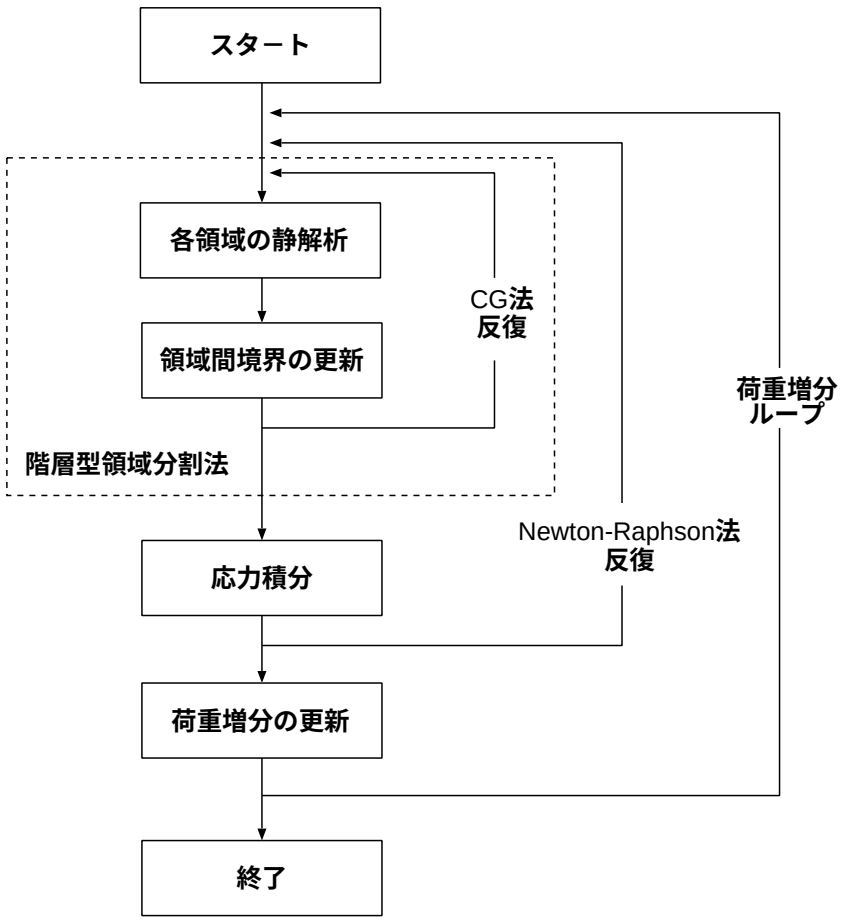


図 6: 非線形解析における処理の流れ

3.3.1 弾塑性モデリング

関連流れ則、von Mises 降伏条件でのバイリニア型の硬化関数を用いた解析が可能であり、弾性解析に加え以下の物性値を用いたモデリングが可能である。

- 加工硬化係数
- 初期降伏応力

3.3.2 増分ステップの制御

増分のステップ数や刻み幅はユーザが適切に設定して解析を実行する必要があるが、初回のステップでは大きな増分をとり、その後は細かい増分をとるといった、増分幅の制御が可能となっている。指定方法に関しては第 5.2.4 節を参照のこと。

3.3.3 解析結果出力

線形弾性解析における出力に加え以下の量が解析結果として出力可能であり、実行時に選択出来る。

- 塑性ひずみテンソル (要素/積分点/節点)
- 相当塑性ひずみ (要素/積分点/節点)
- 降伏応力 (要素/積分点/節点)
- 降伏領域 (要素/積分点)

またこれらの出力は、増分ステップ毎にも可能であり、どの増分時に出力するかも指定可能である。

4 入出力データ

4.1 入出力ファイルの流れ

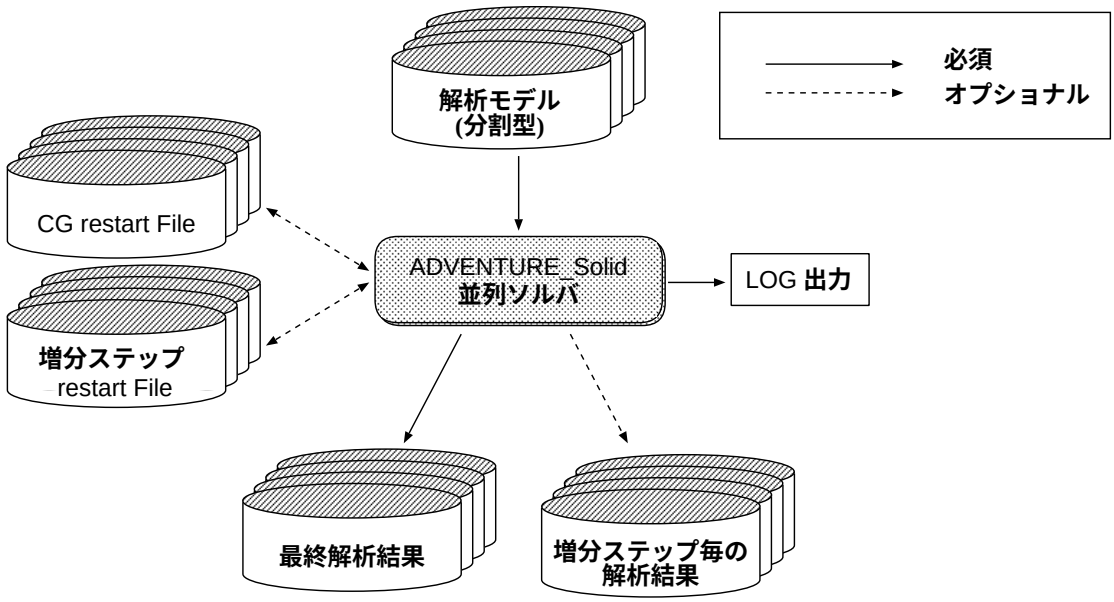


図 7: 入出力ファイルの流れ

ADVENTURE_Solid における入出力ファイルは、図 7 のようになっている。画面へのログ出力以外のファイルは全て、ADVENTURE File フォーマットであり、各部分ごとに 1 ファイルとなっている。

入力ファイルとなるのは階層型に領域分割された ”解析モデルファイル” である。これは解析に先だってドメインディコンポーザ ADVENTURE_Metis によって作成する。

ADVENTURE_Solid からの出力ファイルは、節点変位や応力などの解析により求められた物理量を持つ、”解析結果ファイル” である。出力する物理量は選択可能である。非線形解析時には増分ステップ毎の出力も可能である。これらも階層型に領域分割された形式での出力となる。

連続して実行できる時間が限られている環境などでも解析可能なように、途中までの計算結果を一旦ファイルにセーブし、その時点から計算を再開するためのリスタート機能がある。使用できるリスタートファイルには 2 種類あり、(1) CG リスタートファイル、(2) 増分ステップリスタートファイルである。線形弾性解析のリスタートには CG リスタートファイルを用い、非線形解析のリスタートには増分ステップリスタートファイルを用いる。

4.2 単位系について

入力ファイルでの単位系の指定機能や、プログラム内部での単位系の変換機能は含まれておらず、入力データ作成時に矛盾のない単位系を使用しておく必要がある。

4.3 入出力を行うプロセス

ファイルの入出力を行うプロセスは、静的負荷分散版では全プロセス、動的負荷分散版では親プロセスとなっている。全ての入出力ファイルは各部分毎に 1 ファイルとなっており、それぞれの部分を担当するプロセスにより独立して入出力が行われる。ファイルのパス名はファイル名に含まれる部分番号を除いて全てのプロセスに対して同じものが用いられるため、同じパスで参照できる必要がある。並列環境としてネットワーク接続したワークステーション等を用いる場合には、NFS にてファイルを共有し各マシンから同じパスで参照できるようにしておくのが便利である。そうできない場合には、ftp 等で入力ファイルをあらかじめ各マシンに送り、共通のパス名で参照できる位置に用意しておく必要がある。

ファイルの入出力は、デフォルトでは排他的に行うようになっている。すなわち、各プロセスが同時に入出力を行うことは無く、部分番号順に順次ディスクにアクセスする。これは、並列環境としてネットワーク接続したワークステーションにおいて NFS 等でファイル共有している場合、一度にアクセスが集中するとパフォーマンスが非常に悪くなることを避けるためである。ただし、各プロセスが扱うファイルは独立しているため、個々のマシンのローカルディスクを用いることも可能であり、この場合並列にアクセスしても問題ない。ローカルディスクを用いるなど、並列にファイルアクセス可能な場合は、ADVENTURE_Solid を `-file-para` オプション (第 5.2.8 節参照) つきで実行することで並列にファイルにアクセスを行うようになる。ただしローカルディスクを使用する場合は、上に述べたようにあらかじめ各マシン上の同じパス名で参照できるディレクトリに解析モデルファイルを用意しておく必要がある。

4.4 入力データ

第 4.1 節で示したように、ADVENTURE_Solid 実行に先だって用意する入力ファイルは領域分割された FEM 解析モデルであり、以下の手順で作成する。

(1) メッシュファイルの作成

(2) 境界条件、物性値を設定し、解析モデルファイルを作成する。

(3) 領域分割を行い、領域分割型の解析モデルファイルを作成する。

以下に各手順を説明する。

4.4.1 メッシュファイル

まず、ASCII 形式のテキストファイルとして解析対象に対するメッシュファイルを作成する。使用できる要素は、4 面体、6 面体のそれぞれ 1 次、2 次ソリッド要素である。ただし、異なる要素の混在は不可であるため、全て同一の要素を用いてメッシュを作成する必要がある。

4 面体要素メッシュに関しては ADVENTURE_TetMesh を使用して作成することが出来る。使用方法は ADVENTURE_TetMesh のマニュアルを参考して頂きたい。また、他

のメッシュ作成ツールや手動でメッシュを作成した場合も、以下に示すフォーマットに合わせることで、4 面体、6 面体ともに ADVENTURE システムに取り込むことが可能である。

以下に示すのは、6 面体 1 次要素による メッシュファイルの例である。これは一辺の長さが 2 の立方体を節点数 27、要素数 $2 \times 2 \times 2 = 8$ でメッシュ化したものである。

```
8
       0       1       4       3       9      10      13      12
       1       2       5       4      10      11      14      13
       3       4       7       6      12      13      16      15
       4       5       8       7      13      14      17      16
       9      10      13      12      18      19      22      21
      10      11      14      13      19      20      23      22
      12      13      16      15      21      22      25      24
      13      14      17      16      22      23      26      25

27
  -1.00000000e+00  -1.00000000e+00  -1.00000000e+00
   0.00000000e+00  -1.00000000e+00  -1.00000000e+00
   1.00000000e+00  -1.00000000e+00  -1.00000000e+00
  -1.00000000e+00   0.00000000e+00  -1.00000000e+00
   0.00000000e+00   0.00000000e+00  -1.00000000e+00
   1.00000000e+00   0.00000000e+00  -1.00000000e+00
  -1.00000000e+00   1.00000000e+00  -1.00000000e+00
   0.00000000e+00   1.00000000e+00  -1.00000000e+00
   1.00000000e+00   1.00000000e+00  -1.00000000e+00
  -1.00000000e+00  -1.00000000e+00   0.00000000e+00
   0.00000000e+00  -1.00000000e+00   0.00000000e+00
   1.00000000e+00  -1.00000000e+00   0.00000000e+00
  -1.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00   0.00000000e+00   0.00000000e+00
   1.00000000e+00   0.00000000e+00   0.00000000e+00
  -1.00000000e+00   1.00000000e+00   0.00000000e+00
   0.00000000e+00   1.00000000e+00   0.00000000e+00
   1.00000000e+00   1.00000000e+00   0.00000000e+00
  -1.00000000e+00  -1.00000000e+00   1.00000000e+00
   0.00000000e+00  -1.00000000e+00   1.00000000e+00
   1.00000000e+00  -1.00000000e+00   1.00000000e+00
  -1.00000000e+00   0.00000000e+00   1.00000000e+00
   0.00000000e+00   0.00000000e+00   1.00000000e+00
   1.00000000e+00   0.00000000e+00   1.00000000e+00
  -1.00000000e+00   1.00000000e+00   1.00000000e+00
   0.00000000e+00   1.00000000e+00   1.00000000e+00
   1.00000000e+00   1.00000000e+00   1.00000000e+00
```

まず 1 行目が全要素数である。続く 2〜9 行目は各行に 1 要素分の要素コネクティビティが並んでいる。コネクティビティはその要素を構成する節点番号の並びとして表され、例のように 6 面体 1 次要素なら 8 つの節点番号が各行に並ぶ。この時の節点番号の並べ順は、各要素ごとに決まっており、付録 A に示すようになっている。

次の 11 行目は全節点数である。続く 12〜38 行目は、各節点の座標値が節点番号順に 1

行に 1 節点分のデータとして並ぶ。各行の 1 節点の座標値は、x、y、z の順である。なお、節点番号は 0 から (全節点数 -1) まで連続的に並ぶものとする。

4.4.2 FEM 解析モデル (一体型)

メッシュファイルが用意できたら、次はそのメッシュに対して ADVENTURE_BCtool を用いて境界条件を付加する。また、各種の物性値もここで設定する。ここで作成される解析モデルファイルは、ADVENTURE File フォーマットとなる。設定方法に関しては、ADVENTURE_BCtool のマニュアルを参考のこと。

付加できる境界条件は、以下のものであり、全ての解析に対して有効である。

- 変位境界条件
  強制変位境界条件である。基本的に荷重条件で境界条件を与える場合でも、最低限剛体モードを消去するための 6 自由度は設定する必要がある。
- 荷重境界条件
  解析対象に付加する荷重境界条件である。面荷重による指定の場合も、ここで等価節点力に変換され出力される。

また、以下の物性値は解析種類によらず全ての解析に対して必要である。

- ヤング率 (実数のスカラ)
- ポアソン比 (実数のスカラ)

弾塑性解析時には、上記に加え以下の物性値も必ず設定する必要がある。

- 初期降伏応力 (実数のスカラ)
- 加工硬化係数 (実数のスカラ)

重力を付加する場合は、以下も設定する必要がある。

- 質量密度 (実数のスカラ)
- 重力加速度 (実数の 3 次元ベクトル)

また、熱応力解析時 (線形弾性解析時のみ有効) には以下も設定する必要がある。

- 線膨張係数 (実数のスカラ)
- 参照温度 (実数のスカラ)
- 節点温度 (各節点に対して実数のスカラ)

ここで作成される一体型解析モデルファイルは、バイナリ形式の ADVENTURE Format ファイルである。ADVENTURE Format とは、FEM において使用されるデータを表現するための汎用フォーマットとして ADVENTURE Project より提唱されたデータ格納形式である [11]。このフォーマットでは、節点座標や要素コネクティビティといったデータをそれぞれ種類毎に ADV Document と呼ばれる単位で管理する。一つのファイル中にはこの ADV Document が複数含まれ、それらによって一つの解析モデルが構成される。より詳しくは [11] を参考のこと。また、実際のデータ例は本パッケージ中の `sample_data/` ディレクトリ以下に ADVENTURE Format のもの、およびそれをテキスト化したものが含まれているのでそちらを御覧頂きたい。

なお、ADVENTURE Format ファイルを読み書きするためのライブラリとして、ADVENTURE_IO が ADVENTURE Project より提供されており、ADVENTURE_Solid でもこのライブラリを使用し入出力を行っている。ADVENTURE_IO モジュールを利用することで、各ユーザが直接 ADVENTURE Format ファイルを読み書きすることが可能である。

また、 ADVENTURE Format ファイルの中身をテキスト化するプログラム、 advshow が本パッケージ中に用意してあり、これを用いることで ADVENTURE Format ファイルの中身を知ることが出来る。使用方法については、付録 B.2 を参照のこと。

4.4.3 FEM 解析モデルの領域分割

前節で述べた解析モデルファイルに対し、階層型領域分割モジュール ADVENTURE_Metis を用いて領域分割を行い、ADVENTURE_Solid のための領域分割型 FEM 解析モデルを作成する。ADVENTURE_Metis 自体も MPI により並列化されているため、並列計算環境においての実行が可能となっている。

実行方法等は ADVENTURE_Metis のマニュアルを参照頂きたい。また、ADVENUTE_Solid を実行する上でどのような分割を行えばよいかについては、第 2.2 節を参考のこと。

ここで作成される分割型解析モデルファイルも ADVENTURE Format にて作成されるが、これは領域分割のために若干の拡張を行ったフォーマットとなっている。具体的なフォーマットに関しては、実際のデータ例として本パッケージの `sample_data/` ディレクトリ以下に領域分割された ADVENTURE Format の解析モデル、およびそれをテキスト化したものが含まれているのでそちらを御覧頂きたい。これらのファイルの読み書きも ADVENTURE_IO により可能である。

4.5 解析結果ファイル

4.5.1 出力できる物理量

解析した結果は、解析モデルと同様に部分ごとに 1 ファイルの領域分割の ADVENTURE Format ファイルとしてで出力される。出力される物理量は選択可能である。また非線形解析においては増分ステップごとでの出力も可能である。弾性解析時に出力可能な物理量おのびその出力点は、表 2 となっている。弾塑性解析時にはさらに表 3 の各量も出力可能である。

ここで降伏領域は、積分点での出力の場合は降伏曲面上にあれば 1、無ければ 0 となる整数である。要素での出力の場合は要素内積分点での値の平均 (実数) である。

また、変位と反力に関してはもともと節点上にて得られる量であるため、節点のみでの出力となっている。その他の量に関しては、内部的には積分点で評価している量であり、要素平均や、節点上での出力では、積分点でのデータから評価している。要素平均データでは、積分点でのデータを算術平均することで求めている。また、節点データへの変換は、まず積分点データから各要素の節点に外挿し、その各要素に対して外挿された節点上の量を算術平均することで求めている。

表 2 にあるラベル名とは、それぞれの物理量を特定できるようにつけられている名前である。解析結果ファイルは ADVENTURE Format であり、ADVENTURE Format では節点または要素上のデータを表すための汎用 Document として FEGenericAttribute (FEGA) が用意されている。ここでの出力にはこれを階層型領域分割用に拡張した HDDM_FEGenericAttribute (HDDM_FEGA) という Document を使用している。出力の物理量はこの HDDM_FEGA Document として記述されており、それぞれの物理量を特定するための名前としてつけられているのがこのラベル名である。ADVENTURE_IO ライブラリの関数を用いてデータを読み込む場合は、このラベル名を指定することでそれぞれのデータにアクセスできる。また、後述の解析結果データをマージするツール `hddmmrg` においてもこのラベル名を指定してデータを取り出すようになっている。

デフォルトでの出力データは、節点変位と節点相当応力となっている。他の量を出力する場合は、オプション指定にて行う。また、非線形解析ではステップごとの出力が可能であり、出力間隔も指定できるようになっている。デフォルトではステップごとの出力は行わないので、必要な場合は実行時オプションにて指定する。

最終的な解析結果ファイルと増分ステップごとの解析結果ファイルはファイル名が異なり、また出力できるデータはそれぞれ別々に指定するようになっている。

4.5.2 解析結果のポスト処理

ADVENTURE_Visual を用いることで、解析結果の可視化が可能である。また、ADVENTURE_Visual に含まれる機能以外のポスト処理を各ユーザが行えるよう、本パッケージ中に hddmmrg というプログラムが用意されている。これは領域分割型の ADVENTURE Format である解析結果ファイルを一体型にマージし、ASCII フォーマットにて分割前の節点/要素番号をつけて出力するツールである。hddmmrg では表 2 に示されているラベル名を指定することでそれぞれのデータの抽出を行う。使用方法に関しては、付録 B.1 を参照のこと。

物理量	出力点	ラベル名
変位	節点	Displacement
反力	節点	ReactionForce
応力テンソル	要素	Stress
応力テンソル	積分点	Stress@IntegrationPoint
応力テンソル	節点	NodalStress
相当応力	要素	EquivalentStress
相当応力	積分点	EquivalentStress@IntegrationPoint
相当応力	節点	NodalEquivalentStress
歪みテンソル	要素	Strain
歪みテンソル	積分点	Strain@IntegrationPoint
歪みテンソル	節点	NodalStrain
最大主応力	要素	MaximumPrincipalStress
最大主応力	積分点	MaximumPrincipalStress@IntegrationPoint
最大主応力	節点	MaximumNodalPrincipalStress
中間主応力	要素	MiddlePrincipalStress
中間主応力	積分点	MiddlePrincipalStress@IntegrationPoint
中間主応力	節点	MiddleNodalPrincipalStress
最小主応力	要素	MinimumPrincipalStress
最小主応力	積分点	MinimumPrincipalStress@IntegrationPoint
最小主応力	節点	MinimumNodalPrincipalStress
最大主応力の方向ベクトル	要素	EigenvectorOfMaximumPrincipalStress
最大主応力の方向ベクトル	積分点	EigenvectorOfMaximumPrincipalStress@IntegrationPoint
最大主応力の方向ベクトル	節点	EigenvectorOfMaximumNodalPrincipalStress
中間主応力の方向ベクトル	要素	EigenvectorOfMiddlePrincipalStress
中間主応力の方向ベクトル	積分点	EigenvectorOfMiddlePrincipalStress@IntegrationPoint
中間主応力の方向ベクトル	節点	EigenvectorOfMiddleNodalPrincipalStress
最小主応力の方向ベクトル	要素	EigenvectorOfMinimumPrincipalStress
最小主応力の方向ベクトル	積分点	EigenvectorOfMinimumPrincipalStress@IntegrationPoint
最小主応力の方向ベクトル	節点	EigenvectorOfMinimumNodalPrincipalStress
最大主歪み	要素	MaximumPrincipalStrain
最大主歪み	積分点	MaximumPrincipalStrain@IntegrationPoint
最大主歪み	節点	MaximumNodalPrincipalStrain
中間主歪み	要素	MiddlePrincipalStrain
中間主歪み	積分点	MiddlePrincipalStrain@IntegrationPoint
中間主歪み	節点	MiddleNodalPrincipalStrain
最小主歪み	要素	MinimumPrincipalStrain
最小主歪み	積分点	MinimumPrincipalStrain@IntegrationPoint
最小主歪み	節点	MinimumNodalPrincipalStrain
最大主歪みの方向ベクトル	要素	EigenvectorOfMaximumPrincipalStrain
最大主歪みの方向ベクトル	積分点	EigenvectorOfMaximumPrincipalStrain@IntegrationPoint
最大主歪みの方向ベクトル	節点	EigenvectorOfMaximumNodalPrincipalStrain
中間主歪みの方向ベクトル	要素	EigenvectorOfMiddlePrincipalStrain
中間主歪みの方向ベクトル	積分点	EigenvectorOfMiddlePrincipalStrain@IntegrationPoint
中間主歪みの方向ベクトル	節点	EigenvectorOfMiddleNodalPrincipalStrain
最小主歪みの方向ベクトル	要素	EigenvectorOfMinimumPrincipalStrain
最小主歪みの方向ベクトル	積分点	EigenvectorOfMinimumPrincipalStrain@IntegrationPoint
最小主歪みの方向ベクトル	節点	EigenvectorOfMinimumNodalPrincipalStrain

表 2: 出力可能な物理量

物理量	出力点	ラベル名
塑性歪みテンソル	要素	PlasticStrain
塑性歪みテンソル	積分点	PlasticStrain@IntegrationPoint
塑性歪みテンソル	節点	NodalPlasticStrain
相当塑性歪み	要素	EquivalentPlasticStrain
相当塑性歪み	積分点	EquivalentPlasticStrain@IntegrationPoint
相当塑性歪み	節点	NodalEquivalentPlasticStrain
降伏応力	要素	YieldStress
降伏応力	積分点	YieldStress@IntegrationPoint
降伏応力	節点	NodalYieldStress
降伏領域	要素	PlasticState
降伏領域	積分点	PlasticState@IntegrationPoint

表 3: 弾塑性解析時において出力可能な物理量

5 実行方法

第 2.2 節で述べたように、ADVENTURE_Solid ではシングル版の `advsolid-s` および静的負荷分散型の `advsolid-p` と動的負荷分散型の `advsolid-h` の 2 種類の並列版、計 3 つの実行モジュールがある。解析を行うには、環境に応じてこれらのうちの一つを用いて実行すればよい。

シングル版は、MPI 無しでコンパイル、実行が可能となっている。2 つの並列版は MPI を用いた並列化を行っているため、コンパイル、実行には MPI が必要である。MPI には種々の実装がありコンパイルや実行方法はそれぞれの実装系に依存している。ここでは多くのマシン環境に対応している mpich [9] における実行方法を述べるが、他の MPI 実装を用いる場合は、適宜該当部分をその実装系にあったものに置き換えることで実行できる。

起動の方法は、シングル版の場合

`%` `advsolid-s` [*options*] *data_dir*

となる。

mpich での並列版の実行には、以下のように `mpirun` コマンドを用いる。

`%` `mpirun` [*options_for_mpirun*] `advsolid-p` [*options*] *data_dir*

または

`%` `mpirun` [*options_for_mpirun*] `advsolid-h` [*options*] *data_dir*

ここで [*options_for_mpirun*] は `mpirun` コマンドに対するオプション (付録 C を参照) である。mpich 以外を使用する場合は `mpirun` [*options_for_mpirun*] の部分をその実行環境にそって置き換える必要がある。

[*options*] は ADVENTURE_Solid 自身に対するオプションであり、基本的に 3 つの実行モジュール全てに対して共通である。このオプション指定により、解析種類の指定や種々の設定を行う。詳しくは後述する。

最後の *data_dir* は必須オプションであり、入出力データファイルのあるトップディレクトリを指定する。この下のディレクトリおよびファイル名は次節に示すようになる。

5.1 入出力ファイル名

デフォルトでの入出力ファイル名は以下のようになっている。*data_dir* が ADVENTURE_Solid 実行時の必須引数として指定する、入出力データファイルのトップディレクトリであり、各ファイルはこの *data_dir* 以下に置かれる。

- 解析モデルファイル:
  data_dir`/model/advhddm_in_`P`.adv`
- 最終解析結果ファイル:
  data_dir`/result/advhddm_out_`P`.adv`
- 増分ステップごとの解析結果ファイル:
  data_dir`/result/advhddm_incrout_`S`_`P`.adv`

○ CG リスタートファイル:

data_dir/cg-res/advhddm_cgres_P.adv

○ 増分ステップリスタートファイル:

data_dir/incr-res/advhddm_incrres_S_P.adv

ここで P は部分番号、S は増分ステップ番号を示している。

5.2 実行時オプション

実行時に可能なオプションは以下の通りである。

5.2.1 解析種類の指定

解析種類の指定等に関するオプションとして、以下のものがある。これらの指定が無ければ線形弾性解析を行う。

- `-ep`
  弾塑性解析を行う。モデル作成時に加工硬化係数と初期降伏応力を指定しておく必要がある。

- `-tl`
  Total Lagrange 法による幾何学的非線形解析を行う場合指定する。弾性解析、弾塑性解析共に有効であり、大変位微小歪問題を扱うことが出来る。下の `-ul` オプションとは共存しないのでいずれかを指定する。

- `-ul`
  Updated Lagrange 法 による幾何学的非線形解析を行う場合指定する。弾性解析、弾塑性解析共に有効であり、大変位大歪問題を扱うことが出来る。上の `-tl` オプションとは共存しないのでいずれかを指定する。

可能な解析と指定オプションの関係は以下の表のようになる。

解析種類	オプション
線形弾性解析	
弾性大変位微小歪み解析	`-tl`
弾性大変位大歪み解析	`-ul`
弾塑性解析	`-ep`
弾塑性大変位微小歪み解析	`-ep -tl`
弾塑性大変位大歪み解析	`-ep -ul`

表 4: 解析種類と指定オプション

線形弾性解析以外は増分法による非線形解析であるため、後述の増分ステップ指定オプションを適切に設定する必要がある。

自重による荷重を加える場合は、以下のオプションを指定する。この場合、モデル作成時に質量密度と重力加速度を指定しておく必要がある。また、非線形解析を行う場合には、後述の `-incr-step` オプションのサブオプション `--bf-width` を用いることで、各増分ステップにおいて加味する荷重増分をコントロールできるようになっている。

- `-gravity`
  重力による自重を考慮する場合指定する。

また、熱応力解析を行なう場合は以下のオプションを指定する。この場合、モデル作成時に線膨張係数、参照温度、節点温度を指定しておく必要がある。熱応力解析は線形弾性解析でのみ使用可能である。

- `-thermal`
  熱応力解析時に指定する。

5.2.2 要素に関するオプション

- `-selective-intg volume`
  要素積分において、体積歪みに関する選択的次数低減積分を行う。6 面体 1 次要素に対してのみ有効である。

- `-selective-intg shear`
  要素積分において、せん断歪みに関する選択的次数低減積分を行う。6 面体 1 次要素に対してのみ有効である。

- `-tet10-integ5`
  4 面体 2 次要素を使用する場合に、要素積分において 5 点積分を行う。指定しない場合は 4 点積分である。

5.2.3 出力データ指定オプション

デフォルトで出力するデータは、最終解析結果として出力するものが、変位と節点相当応力、増分ステップごとの出力は無し、となっている。それぞれ出力したいデータを変更する場合には、以下のオプションを用いる。

- `-result` [*sub_options*]
  最終解析結果ファイルへ出力するデータを指定する。

- `-no-result` [*sub_options*]
  最終解析結果ファイルへ出力しないデータを指定する。

- `-incr-result` [*sub_options*]
  増分ステップごとに出力する解析結果データを指定する。

- `-no-incr-result` [*sub_options*]
  増分ステップごとに出力しない解析結果データを指定する。

実際の出力データ種類は、これらのオプションに対する以下のサブオプション *sub_options* により指定する。サブオプションは、一つの (`-result` といった) オプションに続けて複数指定することが出来る。

- `--disp`
  節点変位
- `--reac`
  節点反力
- `--estr`
  要素相当応力
- `--estr-n`
  節点相当応力
- `--estr-i`
  積分点相当応力
- `--str`
  要素応力テンソル
- `--str-n`
  節点応力テンソル
- `--str-i`
  積分点応力テンソル
- `--stra`
  要素歪みテンソル
- `--stra-n`
  節点歪みテンソル
- `--stra-i`
  積分点歪みテンソル
- `--prstr`
  要素主応力 (主応力 3 成分とその固有ベクトル)
- `--prstr-n`
  節点主応力 (主応力 3 成分とその固有ベクトル)
- `--prstr-i`
  積分点主応力 (主応力 3 成分とその固有ベクトル)
- `--prstra`
  要素主歪み (主歪み 3 成分とその固有ベクトル)

- `--prstra-n`
  節点主歪み (主歪み 3 成分とその固有ベクトル)
- `--prstra-i`
  積分点主歪み (主歪み 3 成分とその固有ベクトル)
- `--plstra`
  要素塑性歪みテンソル (弾塑性解析時のみ有効)
- `--plstra-n`
  節点塑性歪みテンソル (弾塑性解析時のみ有効)
- `--plstra-i`
  積分点塑性歪みテンソル (弾塑性解析時のみ有効)
- `--eqplstra`
  要素相当塑性歪み (弾塑性解析時のみ有効)
- `--eqplstra-n`
  節点相当塑性歪み (弾塑性解析時のみ有効)
- `--eqplstra-i`
  積分点相当塑性歪み (弾塑性解析時のみ有効)
- `--ystr`
  要素降伏応力 (弾塑性解析時のみ有効)
- `--ystr-n`
  節点降伏応力 (弾塑性解析時のみ有効)
- `--ystr-i`
  積分点降伏応力 (弾塑性解析時のみ有効)
- `--elpl`
  要素降伏領域 (弾塑性解析時のみ有効)
- `--elpl-i`
  積分点降伏領域 (弾塑性解析時のみ有効)

例えば、オプションとして `"-result --disp --str --estr-n --stra-n"` を指定した場合、最終解析結果ファイルに節点変位、要素応力テンソル、節点相当応力、節点歪みテンソルが出力される。

5.2.4 増分ステップコントロールオプション

非線形解析では以下のオプションを用いて増分ステップを適切に指定する必要がある。

- `-incr-step` n [*sub_options*]
  n 回の増分ステップ解析を行う。デフォルトでは変位と荷重の境界条件、および (オプション指定時のみ) 自重に対し、$1/n$ をかけたものが 1 ステップにおける増分となる。このオプションは複数指定することが出来、指定した順に順次増分ステップが刻まれる。

このオプションには以下のようなサブオプションが指定可能であり、`-incr-step` n に続けて (複数) 指定することが出来る。

- `--bc-width` x
  解析モデルファイルにおける強制変位と荷重の境界条件に対し、x をかけたものを 1 増分ステップにおける変位・荷重増分とする。指定しない場合は $1/n$ (n はステップ数) となる。

- `--bf-width` x
  体積力 (自重) に対して x をかけたものを 1 増分ステップにおける自重の増分とする。指定しない場合は $1/n$ (n はステップ数) となる。

- `--output-interval` n
  この `-incr-step` オプションにおいて指定される増分ステップ中で、増分ステップ n 回毎に増分ステップ解析結果ファイルを出力する。デフォルトでは出力しない。

- `--output-last`
  この `-incr-step` オプションで指定する増分ステップの最終回に増分ステップ解析結果ファイルを出力する。デフォルトでは出力しない。

- `--resout-interval` n
  この `-incr-step` オプションにおいて指定される増分ステップ中で、増分ステップ n 回毎に増分ステップリスタートファイルを出力する。デフォルトでは出力しない。

- `--resout-last`
  この `-incr-step` オプションで指定する増分ステップの最終回に増分ステップリスタートファイルを出力する。デフォルトでは出力しない。

例えば、初めに変位・荷重増分幅を 0.1 で 5 回、さらに変位・荷重増分幅 0.05 で 10 回の増分で弾塑性解析を行う場合は、`"-ep -incr-step 5 --bc-width 0.1 -incr-step 10 --bc-width 0.05"` を指定する。この場合、それぞれのステップにおいてかけられている変位、荷重境界条件は、解析モデルファイルで与えたものに対して、図 8 の縦軸をかけたものが積算の変位、荷重となる。また増分ステップ中の解析結果を出力する例として、変位と要素応力を初めの 5 回の増分ステップではその最後に、次の 10 回のステップでは 2 ステップ毎に出力する場合には、`"-ep -incr-step 5 --bc-width 0.1 --output-last -incr-step 10 --bc-width 0.05 --output-interval 2 -incr-result --disp --str"` のように指定する。

増分ステップリスタートファイルを用いて、解析を再開するには以下のオプションを用いる。

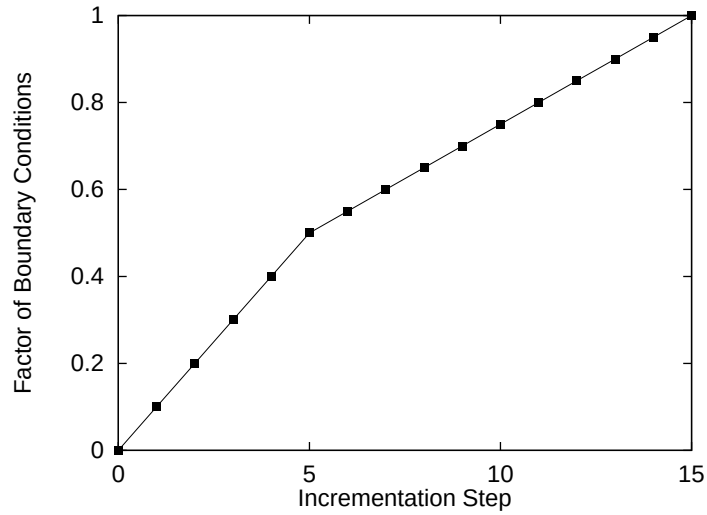


図 8: 増分ステップ指定例

- `-use-incr-resin` n
  前回の実行において出力された、増分ステップ n におけるリスタートファイルを読み込み、そこから解析を再開する。

5.2.5 反復法のコントロールオプション

ADVENTURE_Solid では、全体剛性マトリックスによる線形方程式を解くために CG 法、1 増分ステップにおいて Newton-Raphson 法による反復計算を行っており、それらをコントールするためにいくつかのオプションがある。まず、CG 法に関するオプションには以下のものがある。

- `-cg-tol` x
  収束判定のためのトレランスを指定する。これは、増分ステップ、Newton-Raphson 反復、CG 反復全てにおける初回での CG 残差に対する相対誤差であり、CG 反復においてこれより相対誤差が小さくなった時点で収束とする。非線形解析では、Newton-Raphson 法のトレランスより小さくしておく必要がある。デフォルト値は 1.0×10^{-6} である。

- `-cgloop-max` n
  CG 反復回数の上限を指定する。デフォルトは 10000 になっている。大きな規模の解析ではこれでは収束が得られない場合があるので、大規模解析ではより大きな値を指定すること。

- `-nokeep-kmat`
  HDDM ソルバでのみ使用可能なオプションである。デフォルトでは、剛性マトリックスは CG 法の初回に作成したものを記憶しておき、CG 反復中はその記憶したマ

トリックスを計算に使用するが、このオプションを指定することで、剛性マトリックスを記憶せず各 CG 反復ステップ毎に作成するようになる。メモリ使用量が大幅に少なくなるが、計算時間は大幅に増える。

- `-use-cg-resin`
  CG リスタート入力ファイルを読み込み、その時点からの解析を再開する。線形弾性解析においてのみ有効である。デフォルトでは読み込まない。

- `-resout-cgstep` n
  CG リスタートファイルの出力を CG ステップ n 回おきに行う。n を 0 とすれば出力しない。デフォルトでは出力しない。

- `-resout-cglast`
  CG リスタートファイルの出力を最後の CG ステップで行う。収束時、および収束せずループ回数の上限に達した場合共に出力を行う。デフォルトでは出力しない。

また、BDD 法ソルバ用のオプションとして、以下のものがある。

- `-iLU`
  BDD ソルバ、BDD-DIAG ソルバで有効なオプションである。コースグリッド修正時に、不完全コースマトリクスを使用する。ディフォルトは使用しない。

- `-ginv-alpha` x
  BDD ソルバでのみ有効なオプションである。Neumann-Neumann 前処理時に必要な正則化パラメータ α を x とする。デフォルトは 1.0e-3 である。

- `-resout-bdd-cmat`
  BDD ソルバ、BDD-DIAG ソルバで有効なオプションである。LU 分解されたコースマトリクスをファイルに出力する。デフォルトでは出力しない。

- `-use-bdd-cmat`
  BDD ソルバ、BDD-DIAG ソルバで有効なオプションである。コースマトリクスをファイルより読み込み、前処理に使用する。デフォルトでは読み込まない。

さらに Newton-Raphson 法のコントロール用オプションとして、以下のものがある。

- `-newton-tol` x
  収束判定のためのトレランスを指定する。これは、増分ステップ、Newton-Raphson 反復、CG 反復ともに初回における CG 残差に対する相対誤差であり、Newton-Raphson 法の残差がこれ以下となると収束したと見なされる。CG 法に対するトレランスより大きくする必要がある。デフォルト値は 1.5×10^{-6} である。

- `-newton-max` n
  反復回数の上限を指定する。デフォルトは 10 になっている。これで収束が得られないような場合はこれを大きく設定することでこの上限値を緩和できる。ただし、これを大きくとるより増分ステップの刻みを細かくする方がよいであろう。

5.2.6 入出力ファイル名の変更オプション

第 5.1 節に示したように、入出力に用いるファイルの指定方法は、基本的にそれらのトップディレクトリのみを指定し、そこからのファイル、ディレクトリ名はデフォルト値を用いるようになっている。これを変更する場合、以下のオプションを指定する。ここで S は増分ステップ番号、P は部分番号を示している。

- `-model-file` *file*
  入力解析モデルファイル名を *file* とする。実際のファイル名は、これに `_`P`.adv` をつけたものとなる。デフォルトは `advhddm_in` である。

- `-model-dir` *dir*
  入力解析モデルファイルのあるサブディレクトリ名を *dir* とする。デフォルトは `model` である。

- `-result-file` *file*
  最終の解析結果ファイル名を *file* とする。実際のファイル名は、これに `_`P`.adv` をつけたものとなる。デフォルトは `advhddm_out` である。

- `-result-dir` *dir*
  最終の解析結果ファイルのおかれるサブディレクトリ名を *dir* とする。デフォルトは `result` である。

- `-incr-result-file` *file*
  増分ステップ解析結果ファイル名を *file* とする。実際のファイル名は、これに `_`S`_`P`.adv` をつけたものとなる。デフォルトは `advhddm_incrout` である。

- `-incr-result-dir` *dir*
  増分ステップ解析結果ファイルのおかれるサブディレクトリ名を *dir* とする。デフォルトは `result` である。

- `-incr-resin-file` *file*
  増分ステップリスタート入力ファイル名を *file* とする。実際のファイル名は、これに `_`S`_`P`.adv` をつけたものとなる。デフォルトは `advhddm_incrres` である。

- `-incr-resin-dir` *dir*
  増分ステップリスタート入力ファイルのあるサブディレクトリ名を *dir* とする。デフォルトは `incr-res` である。

- `-incr-resout-file` *file*
  増分ステップリスタート出力ファイル名を *file* とする。実際のファイル名は、これに `_`S`_`P`.adv` をつけたものとなる。デフォルトは `advhddm_incrres` である。

- `-incr-resout-dir` *dir*
  増分ステップリスタート出力ファイルのおかれるサブディレクトリ名を *dir* とする。デフォルトは `incr-res` である。

- `-cg-resin-file` *file*
  CG リスタート入力ファイル名を *file* とする。実際のファイル名は、これに *_P*.`adv` をつけたものとなる。デフォルトは `advhddm_cgres` である。

- `-cg-resin-dir` *dir*
  CG リスタート入力ファイルのあるサブディレクトリ名を *dir* とする。デフォルトは `cg-res` である。

- `-cg-resout-file` *file*
  CG リスタート出力ファイル名を *file* とする。実際のファイル名は、これに *_P*.`adv` をつけたものとなる。デフォルトは `advhddm_cgres` である。

- `-cg-resout-dir` *dir*
  CG リスタート出力ファイルのおかれるサブディレクトリ名を *dir* とする。デフォルトは `cg-res` である。

- `-bdd-dir` *dir*
  BDD ソルバでのみ有効なオプションで、コースマトリクス入出力ファイルのあるサブディレクトリ名を *dir* とする。デフォルトは `bdd` である。

- `-bdd-cmat-file` *file*
  BDD ソルバでのみ有効なオプションで、コースマトリクス入出力ファイル名を *file* とする。デフォルトは `advhbdd_cmat` である。

5.2.7 ソルバ指定オプション

- `-solver` *type*
  線形方程式ソルバを指定する。有効な it type は `hddm`, `cg`, `bdd` または `bdd-diag` のいずれかであり、それぞれ HDDM ソルバ、CG ソルバ、BDD ソルバ、BDD-DIAG ソルバを用いる場合に指定する。ただし、`advsolid-h` では CG ソルバは利用できない。デフォルトは BDD ソルバである。

5.2.8 その他のオプション

その他に、以下のオプションがある。

- `-file-para`
  ファイル入出力を各プロセスが並列に行う (第 4.3 節を参照のこと)。

- `-memlimit` *n*
  各プロセスが使用するメモリの上限を *n* MByte とし、これを越えた場合、その時点で実行を停止する。解析規模が大きく実メモリで足りるかどうか不明だが、スワップはさせたくない場合、1 プロセスが使用できる実メモリ量を指定するといった用途に用いる。これによりチェックされるメモリ量は、実際には内部で動的に確保す

るメモリ量でありプロセスが使用する全メモリではないが、実質的にはこれらはほぼ等しいであろう。

- `-v` または `-version`
  バージョンを表示する。

また、オプションの簡単な説明を表示するために以下のものが用意されている。

- `-help` または `-h`
  メインのヘルプメッセージを表示する。
- `-help-output`
  出力データ指定のためのヘルプメッセージを表示する。
- `-help-incr`
  増分ステップ指定のためのヘルプメッセージを表示する。
- `-help-iter`
  CG 法、Newton-Raphson 法のコントロールオプション指定のためのヘルプメッセージを表示する。
- `-help-bdd`
  BDD ソルバ固有のオプションに関するヘルプメッセージを表示する。

5.3 ADVENTURE_Solid 実行スクリプト `advsolid`

本パッケージには `advsolid-s`, `advsolid-p`, `advsolid-h` の 3 つの実行モジュールに加え、これらの起動用スクリプト `advsolid` が用意されている。これを用いることで、直接実行モジュールを起動するよりも以下のような点で扱いやすいようになっている。

- 設定ファイルにオプション設定を記述できる。
- 並列版を mpich の `mpirun` を用いて実行する場合等では、`advsolid-p`, `advsolid-h` が置かれているパスまで含めてコマンドラインに指定しなくてはいけないが、`advsolid` スクリプトを使う場合はそれが不要になる。
- シングル版、静的負荷分散版、動的負荷分散版のいずれも起動可能

実行方法は、

```
% advsolid [-show] [-log logfile] [-single|-para|-parahddm]
          [options_for_mpirun] [-conf confile| -- ] [solver_options] [data_dir]
```

となる。
各オプションの意味は以下のようになる。

- `-show`
  実際には実行せず、何を実行するかを表示する。

- `-log` *logfile*
  画面出力されるされる実行ログをファイル *logfile* にも出力する。

- `-single|-para|-parahddm`
  シングル版を実行の場合 `-single`、静的負荷分散版を実行の場合`-para`、動的負荷分散版を実行の場合`-parahddm` を指定する。

- *options_for_mpirun*
  `mpirun` を用いて実行する場合、`mpirun` に対するオプションを指定する。

- `-conf` *conffile* または `--`
  設定をファイル *conffile* から読み込む。設定ファイルを使用しない場合は `--` としておき、その後 ADVENTURE_Solid に対するオプションを指定する。

- *solver_options*
  第 5.2 節に記述した、ADVENTURE_Solid に対するオプションを指定する。

- *data_dir*
  第 5.2 節に記述した、解析データのトップディレクトリを指定する。これは ADVENTURE_Solid の必須オプションであるため、設定ファイルで指定していない場合は必ず指定する必要がある。

これらのオプションは設定ファイルを指定する`-conf` *conffile* を除いて、設定ファイル *conffile* に記述しておくことが出来る。

設定ファイルは Bourne shell スクリプトとして `advsolid` により読み込まれる。このとき以下の変数が解釈されるので、これらの変数にオプション値を設定することで指定する。

- `MODE`
  シングル版を実行の場合 `single`、静的負荷分散版を実行の場合 `para`、動的負荷分散版を実行の場合 `parahddm` を指定する。上記のコマンドラインオプション `-single`, `-para`, `-parahddm` のいずれかが指定されている場合は、コマンドラインオプションが優先される。いずれも指定されていない場合はシングル版で実行される。

- `MPIRUN`
  並列実行する場合の MPI の起動コマンドを指定する。mpich の場合は `mpirun` であり、これがデフォルトである。このような起動コマンドが無い MPI 実装では"`MPIRUN=`"と空で設定しておく。対応するコマンドラインオプションはない。

- `MPIOPTS`
  並列実行する場合の MPI の起動コマンドに対するオプションを指定する。デフォルトの設定は無い。このような起動コマンドが無い MPI 実装では設定しない。この `MPIOPTS` 変数と、コマンドラインオプションの *options_for_mpirun* が共に設定されている場合は、"`$MPIOPTS` *options_for_mpirun*" のように、設定ファイル中で指定したもの、コマンドラインで指定したものの順に両方使用される。

- LOGFILE
  画面出力されるされる実行ログをセーブするファイル名を設定する。ファイルにセーブしない場合は、設定しない。上記のコマンドラインオプション -log *logfile*, が指定されている場合は、コマンドラインオプションが優先される。デフォルトではログファイルは作成しない。

- PROGOPTS
  第 5.2 節に記述した、ADVENTURE_Solid に対するオプションを設定する。この PROGOPTS 変数と、コマンドラインオプションの *solver_options* が共に設定されている場合は、"$PROGOPTS *solver_options*" のように、設定ファイル中で指定したもの、コマンドラインで指定したものの順に両方使用される。デフォルトでは何も設定されていない。

- DATADIR
  第 5.2 節に記述した、解析データのトップディレクトリを指定する。この DATADIR 変数と、コマンドラインオプションの *data_dir* が共に設定されている場合は、コマンドラインオプションでの指定が優先される。これは ADVENTURE_Solid の必須オプションであるため、どちらかで必ず指定する必要がある。

設定ファイルの例は以下のようになる。

```
##### set parallel mode  ########################
 MODE=para

##### program name of mpirun ####################
 MPIRUN=/usr/bin/mpirun

##### options for mpirun ########################
 MPIOPTS="-np 2"

##### set if you want save log to file ##########################
 LOGFILE="run.log"

##### Options for AdvSolid ######################
PROGOPTS="-result --str --stra"

##### Data directory to be analyzed ######################
DATADIR=cube_p2d2
```

この場合、静的負荷分散版で 2 ノードにて弾性解析を行う。PROGOPTS の指定より、結果出力にデフォルトの変位と節点相当応力に加え、要素応力テンソル、要素歪みテンソルの出力を設定している。また、解析するデータは cube_p2d2 ディレクトリ以下と設定している。

この設定ファイルを advsolid.conf とすると実行は、

```
% advsolid -conf advsolid.conf
```

とすればよい。

また、この設定ファイルを使用するが、解析データのみディレクトリ cube_p2d2 から another_model に変更する場合は、

```
% advsolid -conf advsolid.conf another_model
```

とすることで実行可能である。

6 コンパイルとインストール

6.1 コンパイル

コンパイルに必要なものは、C コンパイラと MPI のコンパイル環境、ADVENTURE_IO モジュールである。MPI がインストールされていない環境では、フリーソフトウェアの MPICH [9] 等を使用することが出来るので、あらかじめインストールしておく。ただし、シングル版のみを使用する場合は、MPI が無くてもコンパイル、実行することが可能である。ADVENTURE_IO はあらかじめコンパイルしておく必要がある。また、簡単なログ解析用ツールが perl で記述されているため、必須ではないが perl もインストールされているのが望ましい。

コンパイルの手順は、

(1) `./configure` [*options*]

(2) `make`

である。どちらもトップディレクトリにて行う。`configure` は環境に依存する部分を解決し、適切な `Makefile` を作成するためのシェルスクリプトである。

`configure` に渡せる主なオプションを以下に示す。ただし、以下で用いるディレクトリ名には絶対パスを指定すること。

- `--prefix=`*install_dir*
  インストール先のトップディレクトリを *install_dir* にする。デフォルトは `$HOME/ADVENTURE` である。

- `--with-advio=`*directory*
  ADVENTURE_IO が上記で指定する (あるいはデフォルトの) *install_dir* 以外にインストールされている場合に、そのインストール先ディレクトリを指定する。

- `--with-mpicc=`*command*
  MPI プログラム用の C コンパイラ名を示す。デフォルトは `mpicc` である。見つからない場合は並列版のコンパイルは行わない。

- `--with-mpi-cflags=`*CFLAGS*
  MPI プログラムをコンパイルする場合必要な C のコンパイルオプションを指定する。 例えば MPI のインクルードファイルを指定する必要がある場合は、`--with-mpi-cflags="-I/usr/local/include/mpi"` などと指定する。MPI プログラムのコンパイル時には、ここで指定したものに加えシングル版用に設定されているものの両方が用いられる。

- `--with-mpi-libs=`*LIBS*
  MPI プログラムをリンクする場合必要なオプションを指定する。例えば MPI のライブラリを明示する必要がある場合は、`--with-mpi-libs="-L/usr/local/lib/mpi -lmpi"` などと指定する。MPI プログラムのリンク時には、ここで指定したものに加えシングル版用に必要なものの両方が用いられる。

- `--enable-optimize`
  コンパイル時に最適化を行う。これにより設定されるオプション以外の最適化オプションをつけてコンパイルしたい場合は、下の書式を使用する。

- `--enable-optimize=`*CFLAGS*
  CFLAGS を最適化用のオプションとして、コンパイル時に最適化を行う。

また、シングル版や、シングル版、並列版共通部分のコンパイルに使用される C コンパイラを変更するには、以下の環境変数が使用できる。

- `CC`
  C コンパイラ名を設定する。

- `CFLAGS`
  C コンパイラに対するオプションを設定する。

- `LIBS`
  リンクする必要のあるライブラリが他にあれば指定する。

これらは、`./configure` 実行前に`./configure` を実行するシェル中にて設定しておく。例えば、C shell の場合、

```
% setenv CC /usr/local/bin/cc
% setenv CFLAGS "-O2 -g -Wall"
% ./configure
```

等とし、Bourne shell の場合、

```
$ CC=/usr/local/bin/cc
$ export CC
$ CFLAGS="-O2 -g -Wall"
$ export CFLAGS
$ ./configure
```

等として設定する。

`configure` スクリプトを使用することで、多くの環境ではコンパイル可能と思われるが、うまくいかない場合には、`Makefile` のサンプルが各ディレクトリに用意してあるのでそれを用いてコンパイルする。それぞれのディレクトリにて `Makefile.sample` を `Makefile` にコピーし、それを各環境に合わせて編集して各ディレクトリ毎に make して頂きたい。ただし、`solver/` の下をコンパイルする前に `libfem/` の下をコンパイルしておく必要がある。

6.2 インストール

`configure` スクリプトを用いてコンパイルした場合は、トップディレクトリにて、

```
% make install
```

とすることで、作成された実行ファイルおよびマニュアルが *install_directory* 以下にインストールされる。

configure を用いず Makefile.sample を Makefile にコピーしてコンパイルした場合は、上記の libfem を除く各ディレクトリにて

```
% make install
```

を行う。実行ファイルのインストール先ディレクトリは Makefile 中の INSTALL_BINDIR として定義されている。デフォルトでは、$HOME/ADVENTURE/bin となっている。またドキュメントのインストール先ディレクトリは Makefile 中の INSTALL_DOCDIR で定義され、デフォルトでは、$HOME/ADVENTURE/doc/AdvSolid となっている。

Appendix

A　使用可能な要素タイプ

ADVENUTE_Solid では表 5で示す 4 種類のソリッド要素に対応している。ただし一つの解析モデル中で使用できるのは 1 種類のみであり、一つの解析モデル中での異なる要素の混在には対応していない。

要素タイプ	節点数	積分点数
4 面体 1 次	4	1
4 面体 2 次	10	4 (5)
6 面体 1 次	8	8
6 面体 2 次	20	27

表 5: 使用可能な要素

A.1　4 面体 1 次要素

(1) 節点

節点数は 4 であり、要素コネクティビティでの各節点の節点番号の並び順は、図 9 のようになっている。

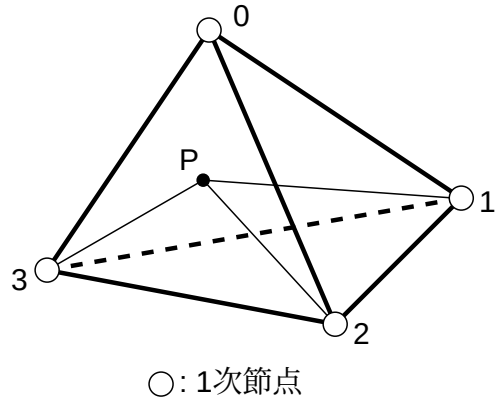


図 9: 4 面体 1 次要素

(2) 積分点

積分点数は 1 であり、積分点は体積座標 (L_0, L_1, L_2, L_3) を用いて 表 6 となっている。ここで体積座標は図 9 で示される点 P を

$$L_0 = \text{4 面体 P123 の体積/4 面体 0123 の体積} \quad (2)$$
$$L_1 = \text{4 面体 P023 の体積/4 面体 0123 の体積} \quad (3)$$
$$L_2 = \text{4 面体 P013 の体積/4 面体 0123 の体積} \quad (4)$$
$$L_3 = \text{4 面体 P012 の体積/4 面体 0123 の体積} \quad (5)$$

で表す座標系である。

積分点番号	L_0	L_1	L_2	L_3
0	1/4	1/4	1/4	1/4

表 6: 4 面体 1 次要素の積分点

A.2　4 面体 2 次要素

(1) 節点

節点数は 10 であり、要素コネクティビティでの各節点の節点番号の並び順は、図 10 のようになっている。

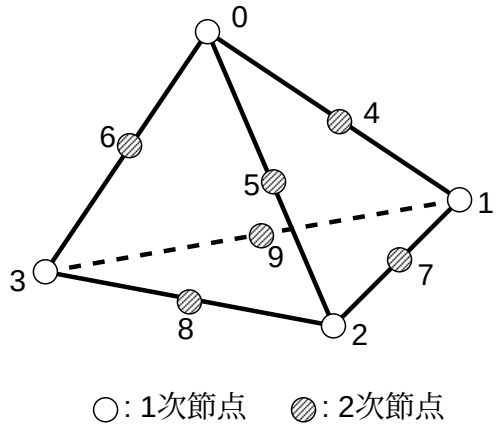


図 10: 4 面体 2 次要素

(2) 積分点

積分点番号	L_0	L_1	L_2	L_3
0	β	α	β	β
1	β	β	α	β
2	β	β	β	α
3	α	β	β	β

表 7: 4 面体 2 次要素の積分点 (4 点積分)

積分点数はデフォルトでは 4 であり、積分点は 4 面体 1 次要素と同様に図 9 および式 (2) 〜 (5) で表される体積座標 (L_0, L_1, L_2, L_3) で表される体積座標を用いて、表 7 となっている。ただし、表 7 において、

$$\begin{aligned} \alpha &= 0.58541019662496845446 \\ \beta &= 0.13819660112501051518 \end{aligned}$$

である。

また、実行時のオプション指定により 5 点積分を行うことが可能である。この場合の積分点は、同様に体積座標を用いて表 8 となっている。

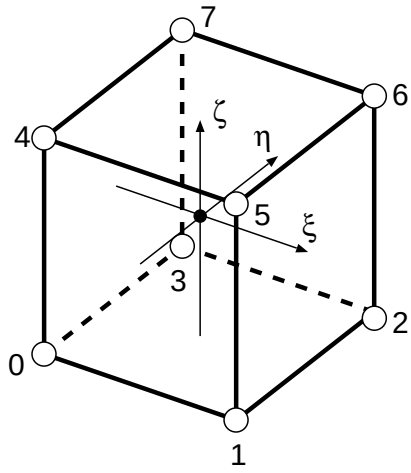
積分点番号	L_0	L_1	L_2	L_3
0	1/4	1/4	1/4	1/4
1	1/6	1/2	1/6	1/6
2	1/6	1/6	1/2	1/6
3	1/6	1/6	1/6	1/2
4	1/2	1/6	1/6	1/6

表 8: 4 面体 2 次要素の積分点 (5 点積分)

A.3　6 面体 1 次要素

(1) 節点

節点数は 8 であり、要素コネクティビティでの各節点の節点番号の並び順は、図 11 のようになっている。



○: 1次節点

図 11: 6 面体 1 次要素

(2) 積分点

積分点番号	ξ	η	ζ
0	$-1/\sqrt{3}$	$-1/\sqrt{3}$	$-1/\sqrt{3}$
1	$1/\sqrt{3}$	$-1/\sqrt{3}$	$-1/\sqrt{3}$
2	$-1/\sqrt{3}$	$1/\sqrt{3}$	$-1/\sqrt{3}$
3	$1/\sqrt{3}$	$1/\sqrt{3}$	$-1/\sqrt{3}$
4	$-1/\sqrt{3}$	$-1/\sqrt{3}$	$1/\sqrt{3}$
5	$1/\sqrt{3}$	$-1/\sqrt{3}$	$1/\sqrt{3}$
6	$-1/\sqrt{3}$	$1/\sqrt{3}$	$1/\sqrt{3}$
7	$1/\sqrt{3}$	$1/\sqrt{3}$	$1/\sqrt{3}$

表 9: 6 面体 1 次要素の積分点

積分点数は 8 であり、図 11 で示される正規化座標 (ξ, η, ζ) (ただし $-1 < \xi, \eta, \zeta < 1$) を用いて、表 9 となっている。

A.4　6 面体 2 次要素

(1) 節点

節点数は 20 であり、要素コネクティビティでの各節点の節点番号の並び順は、図 12 のようになっている。

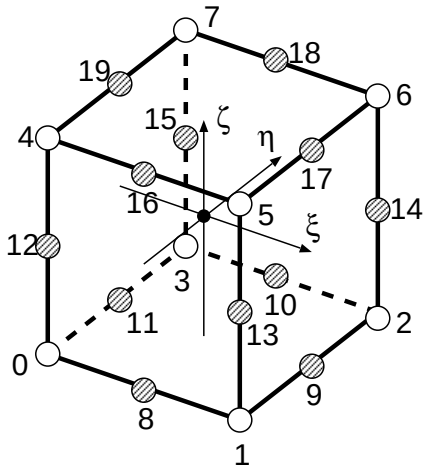


図 12: 6 面体 2 次要素

(2) 積分点

積分点数は 27 であり、6 面体 1 次要素と同様に図 12 で示される正規化座標 (ξ, η, ζ) (ただし $-1 < \xi, \eta, \zeta < 1$) を用いると、積分点は 表 10 となっている。

積分点番号	ξ	η	ζ
0	$-\sqrt{3/5}$	$-\sqrt{3/5}$	$-\sqrt{3/5}$
1	0	$-\sqrt{3/5}$	$-\sqrt{3/5}$
2	$\sqrt{3/5}$	$-\sqrt{3/5}$	$-\sqrt{3/5}$
3	$-\sqrt{3/5}$	0	$-\sqrt{3/5}$
4	0	0	$-\sqrt{3/5}$
5	$\sqrt{3/5}$	0	$-\sqrt{3/5}$
6	$-\sqrt{3/5}$	$\sqrt{3/5}$	$-\sqrt{3/5}$
7	0	$\sqrt{3/5}$	$-\sqrt{3/5}$
8	$\sqrt{3/5}$	$\sqrt{3/5}$	$-\sqrt{3/5}$
9	$-\sqrt{3/5}$	$-\sqrt{3/5}$	0
10	0	$-\sqrt{3/5}$	0
11	$\sqrt{3/5}$	$-\sqrt{3/5}$	0
12	$-\sqrt{3/5}$	0	0
13	0	0	0
14	$\sqrt{3/5}$	0	0
15	$-\sqrt{3/5}$	$\sqrt{3/5}$	0
16	0	$\sqrt{3/5}$	0
17	$\sqrt{3/5}$	$\sqrt{3/5}$	0
18	$-\sqrt{3/5}$	$-\sqrt{3/5}$	$\sqrt{3/5}$
19	0	$-\sqrt{3/5}$	$\sqrt{3/5}$
20	$\sqrt{3/5}$	$-\sqrt{3/5}$	$\sqrt{3/5}$
21	$-\sqrt{3/5}$	0	$\sqrt{3/5}$
22	0	0	$\sqrt{3/5}$
23	$\sqrt{3/5}$	0	$\sqrt{3/5}$
24	$-\sqrt{3/5}$	$\sqrt{3/5}$	$\sqrt{3/5}$
25	0	$\sqrt{3/5}$	$\sqrt{3/5}$
26	$\sqrt{3/5}$	$\sqrt{3/5}$	$\sqrt{3/5}$

表 10: 6 面体 2 次要素の積分点

B ツール類

ADVENTURE_Solid のアーカイブ中には、ADVENTURE_Solid 本体に加え以下のようなツール類が含まれている。

B.1 解析結果の一体型データへの変換 hddmmrg

hddmmrg は、ADVENTURE_Solid により得られる領域分割型 ADVENTURE Format の解析結果データを一体型データにマージし、テキストファイルとして出力するプログラムである。出力フォーマットは単純であるので、解析結果に対し何らかの処理を施したい場合に使用されたい。ただし、可視化モジュールの ADVENTURE_Visual では ADVENTURE_Solid が出力する領域分割型の解析結果ファイルを直接読むため不要である。

実行方法は、

% hddmmrg [*options*] *label* *data_dir*

である。ここで *data_dir* は、領域分割されたモデル、解析結果ファイルの置かれるトップディレクトリであり、advsolid の実行時の引数と同じものを指定する。*label* は抽出するデータを識別するための名前であり、ADVENTURE_Solid 実行時に出力されたものを表 2 中にあるラベル名により指定する。後述するように、解析結果ファイル中にどのラベルのデータが出力されているかも hddmmrg を用いて知ることが出来る。

hddmmrg は 1 回の実行につき一種類 (1 ラベル名) のデータを扱うため、複数種類のデータをマージする場合は、それぞれに対して hddmmrg を実行する。

入力ファイルのデフォルトは解析モデルファイルが *data_dir*/model/advhddm_in_P.adv で、解析結果ファイルが *data_dir*/model/advhddm_out_P.adv である。ここで P は部分番号である。

指定できるオプションは以下の通りである。

- -modelfile *file*
  ADVENTURE_Metis により作成された領域分割型の解析モデルファイルを *file* とする (_P.adv を除いて指定する) 。デフォルトは model/advhddm_in である。
- -resultfile *file*
  ADVENTURE_Solid により作成された、領域分割型の解析結果ファイル名を指定する (_P.adv を除いて指定する) 。デフォルトは result/advhddm_out である。
- -itemlist *file*
  一部の節点または要素データのみを取り出したい場合に、その (一体型での) 節点または要素番号を記したファイルを用意し、ここでそのファイル名 *file* を指定する。デフォルトでは使用せず、全節点 (または要素) に対してで出力する。*file* のフォーマットは ASCII 形式であり、1 行目に出力したい節点 (要素) 数、続いて各行にその節点 (要素) 番号を並べる。
- -h
  ヘルプの表示

また、解析結果ファイル中にどのようなラベル名を持つデータが入っているかを表示するには、

`% hddmmrg [`*options*`] -showlabel` *data_dir*

とすることで行える。上記のオプション中、`-resultfile` *file* が使用できるので必要ならこれを用いてデータファイルを指定する。

`hddmmrg` により一体型にマージされた解析結果のフォーマットは、以下のようになる。

```
label=Displacement
num_items=125

0:   0.00000000e+00   0.00000000e+00   0.00000000e+00
1:  -1.96064988e-06  -1.96064988e-06  -2.77081012e-06
2:  -7.69281443e-07  -1.93681695e-06  -2.30353339e-06
3:  -4.05759629e-21  -1.97623614e-06  -2.21997832e-06
4:   7.69281443e-07  -1.93681695e-06  -2.30353339e-06
5:   1.96064988e-06  -1.96064988e-06  -2.77081012e-06
6:  -1.93681695e-06  -7.69281443e-07  -2.30353339e-06
                           .
                           .
                           .
```

ここで、1 行目は出力したラベル名、2 行目は出力した節点 (要素) 数である。4 行目からがマージされたデータであり、各行に 1 節点 (要素) のデータ (この場合は変位) であり、先頭に節点 (要素) 番号を付けて出力される。各行のデータはスカラデータなら 1 つ、ベクトルなら x, y, z の順に 3 つ、応力や歪みテンソルでは xx, yy, zz, xy, yz, zx の順に 6 つ並ぶ。

また積分点上のデータに関しては、1 行に 1 要素分が出力される。同様にまず行の先頭に要素番号がつき、続いて第 A 節にて示されている積分点順に各積分点データが並ぶ。例えば応力 σ の場合、

```
label=Stress@IntegrationPoint
num_items=64

0: σ_{0,xx}  σ_{0,yy}  ...  σ_{0,zx}  σ_{1,xx}  σ_{1,yy}  ...
1: σ_{0,xx}  σ_{0,yy}  ...  σ_{0,zx}  σ_{1,xx}  σ_{1,yy}  ...
                           .
                           .
                           .
                           .
```

となる。ただし各行の先頭の番号は要素番号であり、$\sigma_{i,xx}$ 等は各要素内の積分点 i の応力成分を示している。

B.2 ADVENTURE Format ファイルを表示する `advshow`

バイナリ形式である ADVENTURE Format のファイルを、テキスト化するツールである。使用方法は、

`% advshow [`*options*`] file1 ...`

であり、*file1 ...* の所にテキスト化したいファイルを指定する。ファイル名は複数指定可能である。デフォルトでの出力は標準出力である。

オプションは以下の通りである。

- `-o` *file*
  標準出力でなく、ファイル *file* に出力する。

- `-p`
  データ部分は出力せず、プロパティ部分のみ出力する。

- `-h`
  ヘルプメッセージを出力する。この場合、*file1 ...* のファイル名の指定は不要である。

B.3 `advsolid` のログを解析する `log2*`

ADVENTURE_Solid が標準エラー出力に出力するログ出力を幾つかの用途向けに整形するための簡単なスクリプトである。perl で記述されているため、使用には perl がインストールされている必要がある。以下の 3 種類のものが用意してある。

- `log2cnv-cg`
  CG 法の収束の様子をグラフ化しやすいよう、プロット用のプログラムに入力しやすい形に整形して出力する。出力フォーマットは、各行毎に 1 ステップのデータであり、各行のデータは、左から (1) CG 反復回数、(2) 相対残差、(3) 絶対残差、(4) 経過時間 [秒]、である。

- `log2cnv-nr`
  増分ステップ解析時の Newton-Raphson 法の収束の様子をグラフ化しやすいよう、プロット用のプログラムに入力しやすい形に整形して出力する。出力フォーマットは `log2cnv-cg` と同様に、各行毎に 1 ステップのデータであり、各行のデータは、左から (1) それまでの積算 CG 反復回数、(2) 相対残差、(3) 絶対残差、(4) 経過時間 [秒]、(5) Newton-Raphson 法反復回数、である。

- `log2info`
  各 ノードが使用したメモリ量や、計算時間のサマリを表示する。

ログの入力方法は 3 つのプログラム共に、(1) 標準入力、(2) ファイルのいずれも可能である。ファイルから入力する場合は、各プログラムの引数としてファイル名を指定すればよい。画面のログをファイルとしてセーブしておくには、シェルのリダイレクション機能を用いるか、`advsolid` スクリプトの `-log` オプションでログファイルを指定することで可能である。

例えば gnuplot を用いる場合、ファイル `run.log` にセーブされている ADVENTURE_Solid の実行ログから CG 残差をプロットするには

```
% log2cnv-cg run.log > cgconv.dat
% gnuplot
gnuplot> set logscale y
gnuplot> plot "cgconv.dat" using 1:2
```

とすれば、y 軸をログスケールにして CG ステップ vs. 相対残差のグラフを得ることが出来る。また、最後の行の ”1:2” の部分を ”1:3” とすれば CG ステップ vs. 絶対残差、”4:2” とすれば経過時間 vs. 相対残差のグラフを得る。

C　MPICH の使用方法

ADVENTURE_Solid の並列版、`advsolid-p` と `advsolid-h` では、並列ライブラリとして MPI [8] を用いている。MPI にはさまざまな実装系があるが、ここでは開発時に主に用いている mpich [9] によるプログラムの実行方法について簡単に説明する。mpich は MPP マシンからネットワーク接続による PC、ワークステーションまで、大変多くのプラットフォームをサポートするフリーソフトであり、あらかじめ用意されている MPI がない場合でも大抵の環境ではこれを使用することが出来るであろう。PC 2 台に Linux を入れ MPICH をインストールすれば、すぐに並列計算が可能となるわけである。

ここではネットワーク接続されたワークステーションによる並列環境 (MPICH において ch_p4 デバイスと呼ばれている)、における実行方法を簡単に述べる。詳細については、mpich のマニュアルを参照頂きたい。

C.1　準備

mpich ではリモートホスト上でプログラムの実行させるために、UNIX コマンドである rsh を用いている (他の方法もあるが、これが最も簡単であろう)。このため、並列プログラムを実行するユーザーは、適切に記述された `.rhosts` という名前のファイルをホームディレクトリに用意しておく必要がある。

ホームディレクトリが NFS により各ホスト間で共有されている場合は、そのホームディレクトリ上に用意すればよい。`.rhosts` ファイルには、ホスト名、ユーザー名を書いた行を、使用するホストごとに、記述する。例えば、ユーザー名が `user` で、ホスト名が `host0`、`host1`、`host2`、`host3`、`host4`、の 5 台のマシンを使うとすると、`.rhosts` ファイルの内容は、

```
1  host0 user
2  host1 user
3  host2 user
4  host3 user
5  host4 user
```

となる。

ただし各ユーザー単位でなく、`/etc/hosts.equiv` 等にてシステム全体として rsh を許可することも可能であり、その場合は、各ユーザーが `.rhosts` を用意する必要が無い。

C.2　実行

MPICH を用いて並列化されたプログラム *program* を実行するには、コマンドプロンプトから次のように入力する。

`%` `mpirun` [*options_for_MPICH*] *program* [*options_for_program*]

MPICH に対するオプション *options_for_MPICH* のうちよく使われるものは以下のものである。なお、詳細は mpich のマニュアルを参照のこと。

- `-np` *number_of_hosts*

number_of_hosts に実行するホストの数を指定する。

- `-machinefile` *machine_file*

  並列計算を行うホストを、デフォルトの設定でなく *machine_file* に書かれたホストに変更する。例えば以下のようになる。

```
1 host0
2 host1
3 host2
4 host3
5 host4
```

  このファイルの先頭にかかれたホストから順に並列実行プロセスとして割り当てられるため、必ずしも実際に実行するホスト数とこのファイル中のホスト数が等しい必要はない。

参考文献

[1] ADVENTURE Project Home Page: `http://adventure.q.t.u-tokyo.ac.jp/`

[2] G. Yagawa and R. Shioya: Parallel Finite Elements on a Massively Parallel Computer with Domain Decomposition, *Computing Systems in Engineering*, **4**, Nos. 4-6 (1993), 495-503.

[3] G. Yagawa and R. Shioya: Massively Parallel Finite Element Analysis, Asakura-Shoten, (1998) (in Japanese).

[4] T. Miyamura, H. Noguchi, R. Shioya, S. Yoshimura and G. Yagawa: Massively Parallel Elestic-Plastic Finite Element Analysis Uning the Hierarchical Domain Decomposition Method, *Transactions of Japan Society of Mechanical Engineers (JSME)*, **65-A**, No. 634 (1999), 1201-1208 (in Japanese).

[5] J. Mandel: Balancing Domain Decomposition, *Communications on Numerical Methods in Engineering*, **9** (1993), 233-241.

[6] R. Shioya, M. Ogino, H. Kanayama and D. Tagami: Parallel Finite Element Analysis with a Balancing Preconditioning Technique, *Lectures in Numerical Simulation in Engineering*, (2001), 30-38.

[7] R. Shioya, M. Ogino, H. Kanayama and D. Tagami: Parallel Finite Element Analysis with a Balancing Domain Decomposition Method, *Computational Mechanics, New Frontiers for the New Millennium, Proceedings of the First Asian-Pacific Congress on Computational Mechanics*, (2001), 133-138.

[8] MPI Home Page: `http://www-unix.mcs.anl.gov/mpi/`

[9] MPICH Home Page: `http://www-unix.mcs.anl.gov/mpi/mpich`

[10] T. Hisada and H. Noguchi: Nonlinear Finite Element Method: Fundamentals and Applications, Maruzen, (1995) (in Japanese).

[11] T. Miyamura, S. Tanaka, H. Takubo, S. Yoshimura and G. Yagawa: Standardization of Input/Output Data in Large Scale Parallel Computational Mechanics System, *Internet Transactions of Japan Society for Computational Engineering and Science (JSCES)*, No. 20000028 (2000) (in Japanese),
`http://homer.shinshu-u.ac.jp/jsces/trans/trans2000/No20000028.pdf`