データ圧縮クラスライブラリ PFClib

(Parallel File Compression Library)

基本設計書

Febrary 2014

Version 1.0.0 19 Feb. 2014

目次

第 1 章

機能概要

1.1　機能一覧

PFClib の機能一覧を図 1.1 に示す.

圧縮機能	固有値直交展開(Proper Orthogonal Decomposition)を用いたデータ圧縮を行う。以降 POD データ圧縮と記載する。 圧縮は時間軸（タイムステップ）方向に行う。 入力データの領域分割と圧縮処理時の領域分割に相違があっても対応可能とすること。
圧縮制御機能	処理指示データを読み込むことにより、圧縮処理を制御する。
展開処理	POD 圧縮されたデータを展開する機能 ・領域範囲を指定することによる展開処理 ・インデックス指定（位置指定）による展開処理 ・On the fly 展開処理
ステージングツール	ステージング環境に対応するため、各計算ノード（MPI ランク）毎に必要なファイルを、ランク番号で命名したディレクトリにコピーする。 ステージング対応用のバッチプログラム。

図 1.1　PFClib 機能一覧

設計上の考慮点を以下に示す.

- 任意のプロセス数での圧縮・展開ができること
  但し、圧縮時の並列数は領域分割数×時間軸方向の並列数に一致していること
- ポータビリティがあること
- 将来的には他の圧縮アルゴリズムも追加実装可能なように設計すること

1.2　圧縮／展開イメージ

並列分散データ圧縮の概要イメージを以下に示す.

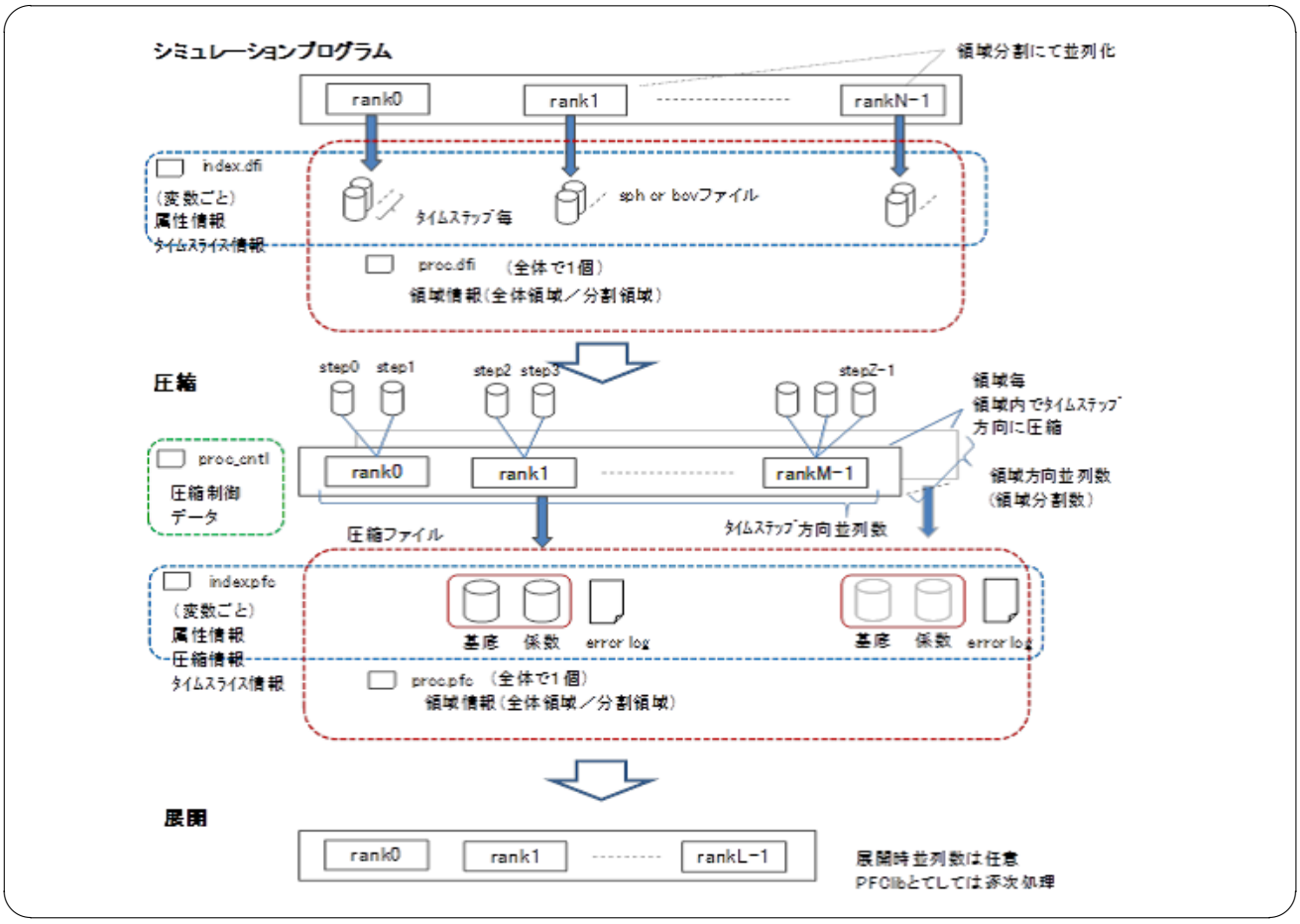


図 1.2　圧縮／展開イメージ概要図

シミュレーションプログラムの領域分割数、圧縮時の領域分割数、展開時の領域分割数は任意である．圧縮時の実行並列数に関しては、以下を満たす必要がある．

圧縮時の実行並列数 = 領域分割数×タイムステップ数で決まる並列数

タイムステップ数で決まる並列数に関しては、5Appendix をご参照のこと．

POD 圧縮の圧縮アルゴリズムに関しては、論文等をご参照のこと．

データ展開に関しては、ユーザプログラムの中で行います．具体的なプログラムは可視化プログラムおよびデータ処理プログラムが想定される．

第 2 章

プログラム構造

(1) 圧縮プログラム構造

圧縮プログラム構造図を図 2.1 に示す.

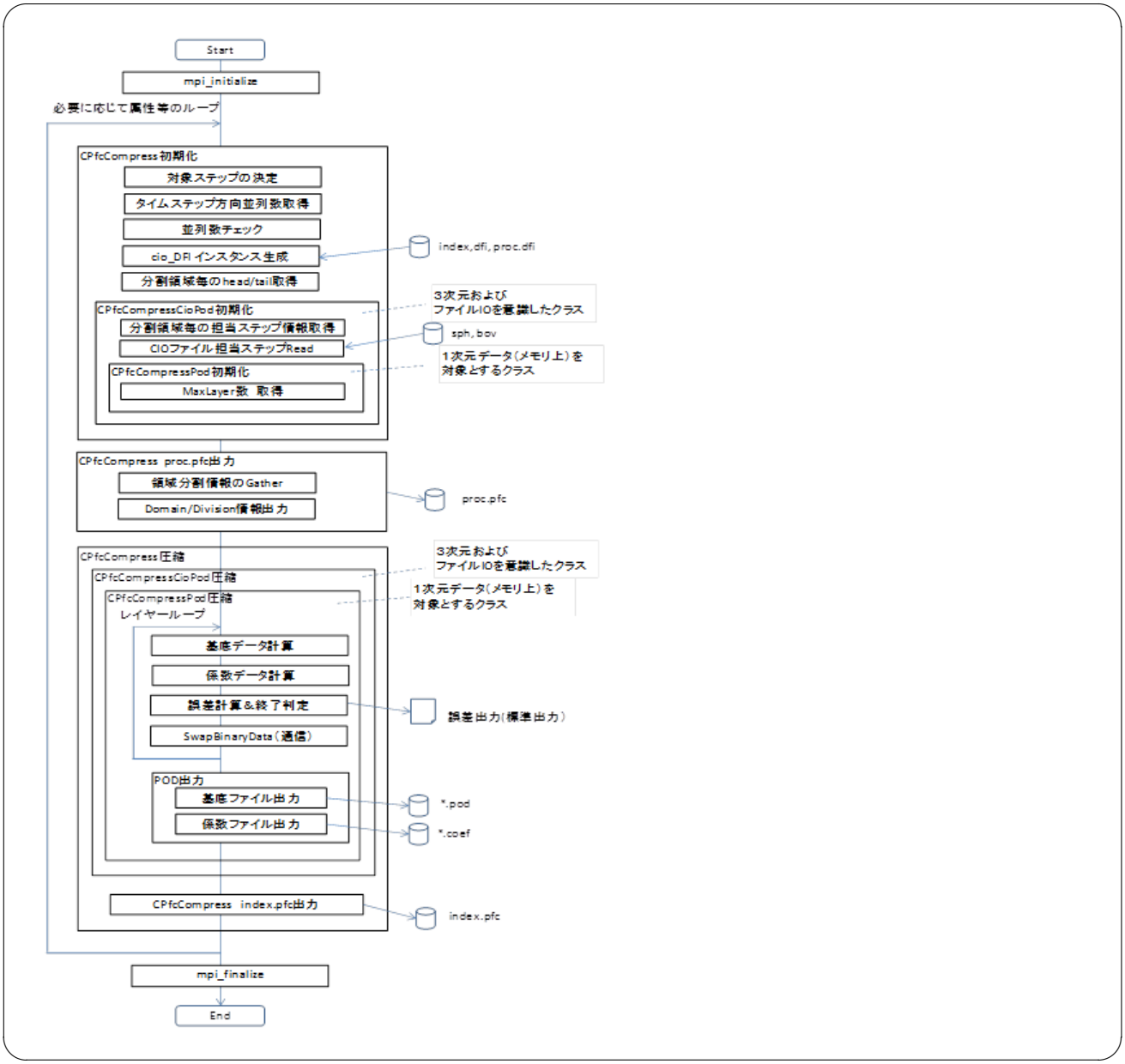


図 2.1　圧縮プログラム構造図

```
*****************************************
  PFC POD 圧縮 プログラム構造
*****************************************

main
1
1 MPI_Init
1
1 CPfcCompress::Init()
1 2
1 2 PFC::MakeDirectory( m_outDirPath );
1 2 CPfcCioDfiUtil::GetDfiTimeStepList()
1 2 CPfcFunction::GetPodParallel()
1 2 CPfcCioDfiUtil::GetDfiDomain()
1 2 cio_DFI::ReadInit()
1 2 CPfcFunction::GetPodRegionID()
1 2 CPfcFunction::CalcHeadTail()
1 2
1 2 CPfcCompressCioPod::Init() ( PfcCompressCioPod.cpp )
1 2 3
1 2 3 CPfcFunction::GetPodRegionID()
1 2 3 CPfcFunction::GetPodStepInfo()
1 2 3
1 2 3 CPfcCompressCioPod::ReadCioFile_IJKN()    IJKN 順読み込み
1 2 3 4 CPfcCompressCioPod::ReadCioFile()
1 2 3 4 5 cio_DFI::GetcioDomain()
1 2 3 4 5 cio_DFI::ReadData()
1 2 3 4 cio_DFI::GetcioDomain()
1 2 3 4 cio_DFI::ReadData()
1 2 3
1 2 3 CPfcCompressPod::Init()
1 2 3 4 CPfcFunction::GetPodParallel()
1 2 3 4 CPfcFunction::GetPodMaxLayer()
1 2 3 4 CPfcFunction::GetPodRegionID()
1 2 3 4 CPfcFunction::GetPodStepInfo()

1 CPfcCompress::WriteProcFile()
1 2 CPfcCompress::GatherAndCreateDivisionInfo()
1 2 3 MPI_send/MPI_recv
1 2 CPfcDomain::Write()
1 2 CPfcDivision::Write()
1
1 CPfcCompress::WriteData()
1 2
1 2 CPfcCompressCioPod::WriteData()
1 2 3
1 2 3 CPfcCompressPod::WriteData()
1 2 3 4
1 2 3 4 --- layer < max_layer のループ (start) -------------------------
1 2 3 4
1 2 3 4 CPfcCompressPod::CalcPodBase()
1 2 3 4 5 PFC_matrix_Trans_d()
1 2 3 4 5 PFC_dgemm()
1 2 3 4 5 PFC_dgeev()
1 2 3 4 5 PFC_dgemv()
1 2 3
1 2 3 4 CPfcCompressPod::CalcPodCoef()
1 2 3 4 5 PFC_ddot()
1 2 3
1 2 3 4 CPfcCompressPod::CheckFinish()
1 2 3 4 5 MPI_Reduce()
1 2 3 4 5 MPI_Bcast()
1 2 3
1 2 3 4 CPfcCompressPod::SwapBinaryData()
1 2 3 4 5 MPI_Isend/MPI_Irecv/MPI_Waitall
1 2 3
1 2 3 4 --- layer < max_layer のループ (end) -------------------------
1 2 3
1 2 3 4 CPfcCompressPod::Output()
1 2 3 4 5
```

```
1 2 3 4 5 CPfcCompressPod::WritePodBaseFile()
1 2 3 4 5 6 CPfcMpiCom::GetMaxInt()
1 2 3 4 5 6 CPfcMpiCom::GatherV_DataDouble()
1 2 3 4 5 6 PFC::MakeDirectory()
1 2 3 4 5 6 CPfcPodFile::WriteBaseFile()
1 2 3 4 5
1 2 3 4 5 CPfcCompressPod::WritePodCoefFile()
1 2 3 4 5 6 CPfcMpiCom::GatherDataDouble()
1 2 3 4 5 6 CPfcMpiCom::GatherDataInt()
1 2 3 4 5 5 CPfcPodFile::WriteCoefFile()
1 2 3
1 2 CPfcCompress::WriteIndexPfcFile()
1 2 3 CPfcFileInfo::Write()
1 2 3 CPfcCompressInfo::Write()
1 2 3 CPfcFilePath::Write()
1 2 3 cio_Unit::Write()
1 2 3 CPfcTimeSlice::Write()
1
1 MPI_Finalize()
2
```

(2) 展開プログラム構造

展開プログラム構造図を図 2.2 に示す.

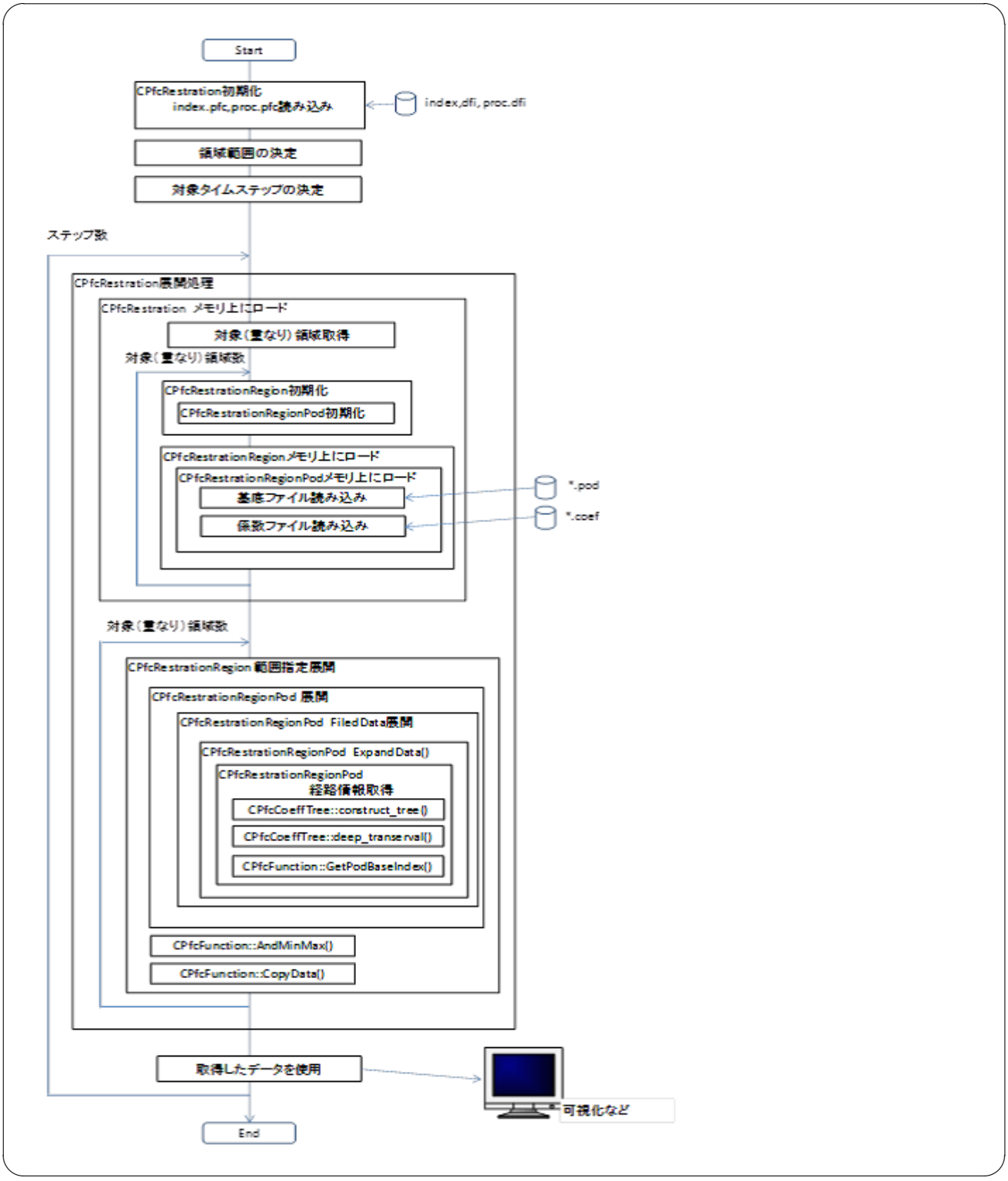


図 2.2　展開プログラム構造図

```
*****************************************
  PFC POD 展開 プログラム構造
   ( 圧縮ファイルから１タイムステップずつ読み込むパターン )
*****************************************

main
1
1
1 CPfcRestration::Init()
1 2
1 2 PFC::PFCPath_DirName()
1 2 CPfcFunction::GetPodParallel()
1 2 CPfcTextParser::getTPinstance()
1 2 CPfcFileInfo::Read()
1 2 CPfcCompressInfo::Read()
1 2 CPfcFilePath::Read()
1 2 CPfcUnitList::Read()
1 2 CPfcTimeSlice::Read()
1 2 CPfcRestration::GetTimeStepList()
1 2 PFC::cioPath_FileName()
1 2 PFC::cioPath_ConnectPath()
1 2 CPfcDomain::Read()
1 2 CPfcDivision::Read()
1
1 CPfcRestration::GetHeadTail()
1 2 CPfcFunction::CalcHeadTail()
1
1 CPfcRestration::GetTimeStepList()
1
1 --- ステップ数数ループ (start) -------------------------
1
1 CPfcRestration::ReadData()  head/tail 指定
1 2
1 2 CPfcRestration::LoadCompressDataOnMem()
1 2 3 CPfcDivision::CheckReadRegion()
1 2  3 --- 対象となる領域数ループ (start) -------------------------
1 2  3 CPfcRestrationRegion::Init()
1 2  3 4 CPfcRestrationRegionPod::Init()
1 2  3 CPfcRestrationRegion::LoadCompressDataOnMem()
1 2  3 4 CPfcRestrationRegionPod::LoadCompressDataOnMem()
1 2 3 4 5 CPfcPodFile::ReadBaseFile()
1 2 3 4 5 CPfcPodFile::ReadCoefFile()
1 2  3 --- 対象となる領域数ループ ( end ) -------------------------
1 2
1 2 --- 対象となる領域数ループ (start) -------------------------
1 2
1 2 CPfcRestrationRegion::ReadDataInRange()
1 2 3 CPfcRestrationRegionPod::ReadData()
1 2 3 4 CPfcRestrationRegionPod::ReadFieldData()
1 2 3 4 5 CPfcRestrationRegionPod::ExpandData()
1 2 3 4 5 6 CPfcRestrationRegionPod::GetExpandRowInfo()
1 2 3 4 5 6 7 CPfcCoeffTree::construct_tree()
1 2 3 4 5 6 7 CPfcCoeffTree::deep_transerval()
1 2 3 4 5 6 7 CPfcFunction::GetPodBaseIndex()
1 2 3 CPfcFunction::AndMinMax()
1 2 3 CPfcFunction::CopyData()
1 2 --- 対象となる領域数ループ  (end) -------------------------
1 2
1 2 CPfcRestration::DeleteCompressDataOnMem()
1
1 --- ステップ数数ループ ( end ) -------------------------
```

第 3 章

ソースファイル一覧

ソースファイル (‘.cpp のみ) の一覧を図 3.1 に示す.

区分	ファイル名	内容
Compress	PfcCompress.cpp	圧縮クラス
	PfcCompress_WriteIndexPfcFile.cpp	圧縮クラス(index.pfc 出力)
	PfcCompress_WriteProcFile.cpp	圧縮クラス(proc.pfc 出力)
	PfcCompressCioPod.cpp	CIO(3 次元) を意識したクラス、ファイル IO あり
	PfcCompressPod.cpp	POD 圧縮 1 次元データクラス
	PfcCompressPod_CalPod.cpp	基底データ、係数データ計算
	PfcCompressPod_CheckFinish.cpp	誤差計算＆終了判定
	PfcCompressPod_Output.cpp	POD ファイル出力
	PfcCompressPod_SwapBinaryData.cpp	データスワップ
	PfcMatrix.cpp	マトリック計算クラス
Compress command	PfcCompressCmd.cpp	圧縮コマンドクラス
	PfcCompressCmdElm.cpp	圧縮コマンド要素クラス
Compress その他	PfcMpiCom.cpp	通信用クラス
	PfcCioDfiUtil.cpp	DFI utility クラス
Restration	PfcRestration.cpp	展開クラス
	PfcRestrationRegion.cpp	展開 領域クラス
	PfcRestrationRegionPod.cpp	POD 展開 領域クラス(１次元データ)
	PfcRestrationRegionPod_Expand.cpp	POD 展開 処理
	PfcCoeffTree.cpp	POD 展開ツリークラス
POD ファイル	PfcPodFile.cpp	ＰＯＤ基底ファイル、係数ファイル IO
index.pfc proc.pfc ファイル	PfcTextParser.cpp	テキストパーサ
	PfcCompressInfo.cpp	
	PfcDivision.cpp	
	PfcDomain.cpp	
	PfcFileInfo.cpp	
	PfcFilePath.cpp	
	PfcRegion.cpp	
	PfcTimeSlice.cpp	
	PfcUnitList.cpp	
共通	PfcFunction.cpp	PFC 関数 （POD 用関数を含む）
	PfcPathUtil.cpp	ファイルパス関連関数
	PfcPerfMon.cpp	測 定 用 ク ラ ス (Performance Monitor:Compress/Restration 用)

図 3.1　ソースファイル一覧

第 4 章

ファイル仕様

4.1　ファイル仕様

4.1.1　インデックスファイル（index.pfc）仕様

index.pfc ファイルはファイル情報（FileInfo），圧縮情報（CompressInfo），ファイルパス情報（FilePath），単位系（UnitList），時系列データ（TimeSlice）の５つのブロックで構成されています．

以下に，index.pfc ファイルの仕様とサンプルをブロック毎に示します．

ファイル情報（FileInfo）の仕様

```
FileInfo
{
  DirectoryPath = "./hoge"   // フィールドデータの存在するディレクトリ
  Prefix        = "vel"      // ベースファイル名                (※ 1)
  FileFormat    = "pod"      // ファイルタイプ，拡張子          (※ 1)
  GuideCell     = 0          // 仮想セル数 ( =0 固定 )
  DataType      = "Float64"  // データタイプ                    (※ 2)
  Endian        = "little"   // データのエンディアン            (※ 3)
  ArrayShape    = "ijkn"     // 配列形状                        (※ 4)
  Component     = 3          // 成分数（スカラーは不要）        (※ 4)
}
```

(※ 1) ファイル名
　　並列時 [Prefix]_id[領域 ID:6 桁].[ext]
　　逐次時 [Prefix].[ext]
　　pod の係数ファイルの拡張子は coef とする
(※ 2) Float64
　　圧縮前データが Float32 の場合は圧縮時の誤差が大きくなるので Float64 に変換する
(※ 3) little,big
(※ 4) ijkn,nijk
　　ijkn:(imax,jmax,kmax,Component)
　　nijk:(Component,imax,jmax,kmax)

圧縮情報（CompressInfo）の仕様

```
CompressInfo
{
  CompressFormat  = "pod"    // 圧縮形式              (※ 5)
  CompressError   = 0.01     // 利用者指定誤差率（%）
  CalculatedLayer = 4        // 計算レイヤー数        (※ 6)
  Version         = "1.0.0"  // 圧縮形式のバージョン
  StartStep       = 0        // 開始ステップ
  EndStep         = 90       // 終了ステップ
}
```

(※ 5) pod pod 以外は reserve
(※ 6) pod の時のみ有効
　　精度確保のため，指定された誤差率より計算レイヤー数が変更される可能性あり

ファイルパス（FilePath）の仕様

```
FilePath
{
  DfiPath    = "vel.dfi"      // CIO DFI ファイルパス
  PfcProcess = "proc.pfc"     // PFC proc ファイル名
                              //   並列情報，領域情報（全体，分割）
}
```

単位系（UnitList）の仕様

```
UnitList
{
  Length {
    Unit       = "NonDimensional"    // (NonDimensional, m, cm, mm)
    Reference  = 1.000000e+00
  }
  Pressure {
    Unit       = "NonDimensional"
    Reference  = 0.000000e+00
    Difference = 1.176300e+00        // 圧力差 (Pa)
  }
  Velocity {
    Unit       = "NonDimensional"
    Reference  = 1.000000e+00
  }
}
```

時系列データ（TimeSlice）の仕様

```
TimeSlice
{
  Slice[@] {            // ファイル出力回数分
    Step = 0                    // 出力ステップ
    Time = 0.0                  // 出力時刻
    AverageTime =               // 平均時間（必要に応じて出力）
    AverageStep =               // 平均化したステップ数（必要に応じて出力）
    VectorMinMax {
      Min = 0.000000e+00
      Max = 0.000000e+00
    }
    MinMax[@] {         // Component 個
      Min  = -1.56e-2           // 最小値
      Max  = 8.2e-01            // 最大値
    }
  }
  Slice[@] {
    Step = 10                   // 出力ステップ
    Time = 3.125e-2             // 出力時刻
    AverageTime =               // 平均時間（必要に応じて出力）
    AverageStep =               // 平均化したステップ数（必要に応じて出力）
    VectorMinMax {
      Min = 0.000000e+00
      Max = 0.000000e+00
    }
    MinMax[@] {         // Component 個
      Min  = -4.000938e-05      // 最小値
      Max  = 2.169153e-04       // 最大値
    }
  }
  Slice{@} {
    :
    :
  }
}
```

4.1.2　プロセス情報ファイル（proc.pfc）仕様

proc.pfc ファイルはドメイン情報（Domain），分割領域情報（Division）の２つのブロックで構成されています．

以下に，proc.pfc ファイルの仕様とサンプルをブロック毎に示します．

ドメイン情報（Domain）の仕様

```
Domain
{
  GlobalOrigin   = (-3.00, -3.00, -3.00)      // 計算空間の起点座標
  GlobalRegion   = ( 6.00,  6.00,  6.00)      // 計算空間の各軸方向の長さ
  GlobalVoxel    = (64, 64, 64)               // 計算領域全体のボクセル数
  GlobalDivision = (2, 2, 2)                  // 計算領域の部分領域の分割数
}
```

分割領域情報（Division）の仕様

```
Division {{
  Region[@] { //  分割領域の数あり（上記の例では 2x2x2->8 個 データあり）
    ID        = 0                    // 分割した領域の ID
    VoxelSize = (32, 32, 32)         // ボクセルサイズ
    HeadIndex = (1, 1, 1)            // 始点インデクス　グローバルで (1,1,1) からスタート
    TailIndex = (32, 32, 32)         // 終点インデクス　グローバルで (64, 64, 64) が終端
  }
  Region[@] {
  ...
}
```

4.1.3　圧縮制御情報ファイル（pfc_cntl）仕様

pfc_cntl ファイルは圧縮制御情報（PfcCompressCntl）のブロックで構成されています.

以下に，pfc_cntl ファイルの仕様とサンプルを示します.

圧縮制御情報（PfcCompressCntl）の仕様

```
PfcCompressCntl
{
  DomainDivision  = (2, 2, 2)   // 計算領域の部分領域の分割数　※必須

  ItemCntl[@] { // 属性（変数）ごとに指定する

    InputDfiPath    = "vel.dfi" // CIO の index.dfi のパス     ※必須
                                // Input となる DFI パスを指定
    OutputDirectoryPath = "./"  // 出力ディレクトリパス

    CompressFormat  = "pod" // 圧縮形式    ※必須
    CompressError   = 0.01  // 許容誤差率（%）      ※省略可（省略時：0.01）

    StartStep       = 100   // 圧縮開始ステップ　※省略可（省略時：dfi ファイルに従う）※ 1
    EndStep         = 200   // 圧縮終了ステップ　※省略可（省略時：dfi ファイルに従う）※ 1

    ProcFileSave    = “ON”// proc.pfc ファイル出力の有無　(ON/OFF) ）　※必須　※ 2
    OptSave         = "ON"  // 検証用：全ての圧縮段階の基底ベクトルと係数の組を残す
                            // ※省略可（省略時： "OFF" ）
  }

  ItemCntl[@] {
    ・・・
  }
  ・・・
}
```

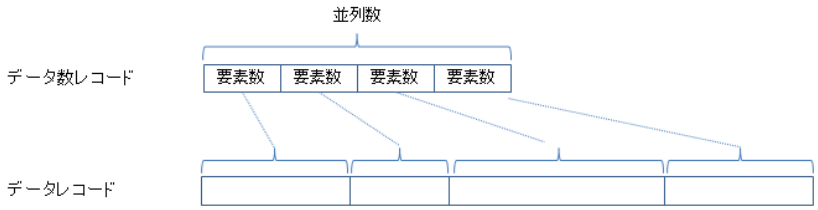
(※ 1) 同一ファイル内で同じ属性（変数）に対して，複数の開始ステップ〜終了ステップを指定することは出来ません.
(※ 2) 全体で proc.pfc ファイルは 1 ファイルあれば良い．複数 Save 指定された場合など，proc.pfc ファイルが既に存在する場合は上書きされる.

表 4.1　基底ファイル

レコード名	型	サイズ	出力数	内容
エンディアンチェックレコード	整数	4byte	1	整数 1 固定 Read 時に 1 として読み込めたかどうかをチェックして エンディアンを判定する
データタイプレコード	整数	4byte	1	整数 2 固定 2：倍精度浮動小数点
タイムステップ数レコード	整数	4byte	1	タイムステップ数
並列数レコード	整数	4byte	1	圧縮時の並列数
レイヤー数レコード	整数	4byte	1	圧縮時の計算レイヤー数
データ数レコード	整数	4byte	並列数	各並列ごとのデータ（要素）数
データレコード	倍精度	8*データ数	並列数	各並列ごとのデータ

4.2　PFC ファイル仕様

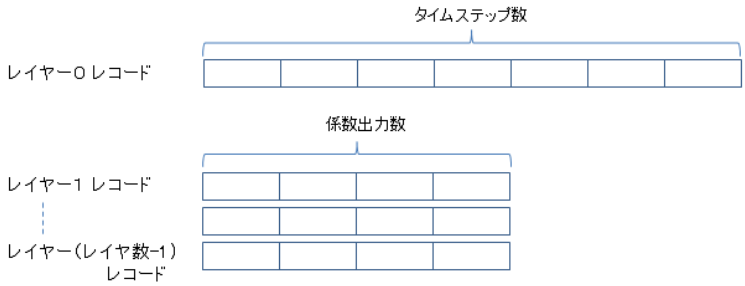
4.2.1　基底ファイル



4.2.2　係数ファイル

表 4.2　係数ファイル

レコード名	型	サイズ	出力数	内容
エンディアンチェックレコード	整数	4byte	1	整数 1 固定 Read 時に 1 として読み込めたかどうかをチェックして エンディアンを判定する
データタイプレコード	整数	4byte	1	整数 2 固定 2：倍精度浮動小数点
タイムステップ数レコード	整数	4byte	1	タイムステップ数
係数出力数レコード	整数	4byte	1	係数出力数=2^((int)log2(タイムステップ数）) （並列数× 2）
レイヤー数レコード	整数	4byte	1	圧縮時の計算レイヤー数
データレコード（レイヤ 0）	倍精度	8byte	数	レイヤー 0 のデータ
データレコード（レイヤ 1 以降）	倍精度	8byte	係数出力数	レイヤー 1 以降のデータ



4.3　PFC ファイル仕様 (デバッグ用)

4.3.1　基底ファイル (デバッグ用)

表 4.3　基底ファイル (デバッグ用)

レコード名	型	サイズ	出力数	内容
エンディアンチェックレコード	整数	4byte	1	整数 1 固定 Read 時に 1 として読み込めたかたどうかをチェックして エンディアンを判定する
データタイプレコード	整数	4byte	1	整数 2 固定 2：倍精度浮動小数点
タイムステップ数レコード	整数	4byte	1	タイムステップ数
並列数レコード	整数	4byte	1	圧縮時の並列数
レイヤー数レコード	整数	4byte	1	圧縮時の計算レイヤー数
データ数レコード	整数	4byte	1	出力データ（要素）数
データレコード	倍精度	8*データ数	1	圧縮データ

ファイル名 並列時 [Prefix]_id[領域 ID:6 桁]-[領域内の連番:5 桁]_layer[レイヤー No:2 桁].pod
・各レイヤー毎に出力する
・各ランク毎に出力する（ランクで集めるとサイズが大きくなりすぎるため）
・領域数が 0 であっても、ファイル名に領域 ID を含む。

第 5 章

Appendix

5.1 タイムステップ毎の Max レイヤー数＆並列数

タイムステップ数に応じた Max レイヤー数とステップ方向の並列数の例を以下に示す.

POD圧縮方式
タイムステップ数→maxレイヤー数およびstep方向圧縮並列数

step数	log2(step数)	max_layer	ベース step数	step方向 圧縮 並列数
		(int)log2(step数)	2**max_layer	ベース step数/2
1	0.00	0	1	-
2	1.00	1	2	1
3	1.58	1	2	1
4	2.00	2	4	2
5	2.32	2	4	2
6	2.58	2	4	2
7	2.81	2	4	2
8	3.00	3	8	4
9	3.17	3	8	4
10	3.32	3	8	4
11	3.46	3	8	4
12	3.58	3	8	4
13	3.70	3	8	4
14	3.81	3	8	4
15	3.91	3	8	4
16	4.00	4	16	8
17	4.09	4	16	8
18	4.17	4	16	8
19	4.25	4	16	8
20	4.32	4	16	8
21	4.39	4	16	8
22	4.46	4	16	8
23	4.52	4	16	8
24	4.58	4	16	8
25	4.64	4	16	8
26	4.70	4	16	8
27	4.75	4	16	8
28	4.81	4	16	8
29	4.86	4	16	8
30	4.91	4	16	8
31	4.95	4	16	8
32	5.00	5	32	16
33	5.04	5	32	16
34	5.09	5	32	16
35	5.13	5	32	16
36	5.17	5	32	16
37	5.21	5	32	16
38	5.25	5	32	16
39	5.29	5	32	16
40	5.32	5	32	16
41	5.36	5	32	16
42	5.39	5	32	16
43	5.43	5	32	16
44	5.46	5	32	16
45	5.49	5	32	16
46	5.52	5	32	16
47	5.55	5	32	16
48	5.58	5	32	16
49	5.61	5	32	16
50	5.64	5	32	16
64	6.00	6	64	32
128	7.00	7	128	64
256	8.00	8	256	128
512	9.00	9	512	256
1024	10.00	10	1024	512

図 5.1　Appendix