CGNS 中级程序库

1 引言

本文介绍了一种 CGNS 程序库，它设计来通过向开发人员提供一批方便的 I/O 函数而容易地执行 CGNS。由于使用此程序库并不需要了解 ADF 核心库，因此，它大大简化了与 CGNS 的接口工作。

CGNS 中级程序库以《SIDS-to-ADF 文件映象手册》为基础。它允许阅读和编写该手册中所讲述的所有信息，包括网格坐标、块接口、流动解和边界条件。利用这种中级程序库函数，确保用户应用与 CGNS 数据的内部表示之间的有效交流。

我们假设读者熟悉《CGNS 标准接口数据结构（SIDS）》中的信息以及文件映象手册。非常鼓励用户阅读《CGNS 用户指南》，这包括利用中级程序库编写和阅读包含 CGNS 数据库的简单文件的编程实例。

2 总论

2.1 语言

CGNS 中级程序库用 C 语言编写，但每个函数有一个 Fortran 对应函数。所有函数都用“cg_”起名。Fortran 函数具有与其 C 对应函数相同的名称，附以后缀 “_f”。

2.2 字符串

在 CGNS 中，所有数据结构的名称和标示都限制在 32 个字符。当阅读一个文件时，建议预先规定字符串变量，Fortran 中为 32 个字符，C 语言中为 33 个字符（包括串终止符）。其他字符串，如 CGNS 文件名或描述符文本，在长度上不受限制。

2.3 错误状态

所有 C 函数都要返回一个代表错误状态的整数值。所有 Fortran 函数都有一个附加参数 ier，它含有错误状态的值。错误状态不同于 0，它意味着发生了错误。错误信息可利用第 5 节中描述的 CGNS 程序库的错误处理函数打印出来。在 C 语言和 Fortran 语言包含文件 cgnslib.h 和 cgnslib_f.h 中编写错误代码，。

2.4 类型定义

利用 cgnslib.h 文件中的类型定义确定变量的几种类型。它们用来简化 C 语言中 CGNS 的执行。对这些类型的任何一种变量，这些变量的类型定义为若干可采纳的关键词。采用这种数据

类型的任何 C 语言应用程序都必须包括 cgnslib.h 文件。

在 Fortran 中，同样的关键词定义为包含文件 cgnslib_f.h 中的整数参数。这样的变量应该表示为 Fortran 应用中的整数。采用这种关键词的任何 Fortran 应用都必须包括 cgnslib_f.h 文件。对于每个变量类型 (类型定义) ，支持值（关键词）的列表如下：

```
ZoneType_t          Structured, Unstructured

ElementType_t       NODE, BAR_2, BAR_3, TRI_3, TRI_6, QUAD_4, QUAD_8, QUAD_9,
                    TETRA_4, TETRA_10, PYRA_5, PYRA_14, PENTA_6, PENTA_15, PENTA_18,
                    HEXA_8, HEXA_20, HEXA_27, MIXED, NGON_n

DataType_t          Integer, RealSingle, RealDouble, Character

DataClass_t         Dimensional, NormalizedByDimensional,
                    NormalizedByUnknownDimensional, NondimensionalParameter,
                    DimensionlessConstant

MassUnits_t         Null, UserDefined, Kilogram, Gram, Slug, PoundMass


LengthUnits_t       Null, UserDefined, Meter, Centimeter, Millimeter, Foot, Inch

TimeUnits_t         Null, UserDefined, Second

TemperatureUnits_t
                    Null, UserDefined, Kelvin, Celsius, Rankine, Fahrenheit

AngleUnits_t        Null, UserDefined, Degree, Radian

GoverningEquationsType_t
                    Null, UserDefined, FullPotential, Euler, NSLaminar, NSTurbulent
                    NSLaminarIncompressible, NSTurbulentIncompressible

ModelType_t         Null, UserDefined, Ideal, VanderWaals, Constant, PowerLaw,
                    SutherlandLaw, ConstantPrandtl, EddyViscosity, ReynoldsStress,
                    ReynoldsStressAlgebraic, Algebraic_BaldwinLomax,
                    Algebraic_CebeciSmith, HalfEquation_JohnsonKing,
                    OneEquation_BaldwinBarth, OneEquation_SpalartAllmaras,
                    TwoEquation_JonesLaunder, TwoEquation_MenterSST,
                    TwoEquation_Wilcox

GridLocation_t      Vertex, IFaceCenter, CellCenter, JFaceCenter, FaceCenter,
                    KFaceCenter, EdgeCenter

GridConnectivityType_t
                    Overset, Abutting, Abutting1to1

PointSetType_t      PointList, PointRange, PointListDonor, PointRangeDonor,
                    ElementList, ElementRange

BCType_t            Null, UserDefined, BCAxisymmetricWedge, BCDegenerateLine,
                    BCExtrapolate, BCDegeneratePoint, BCDirichlet, BCFarfield,
                    BCNeumann, BCGeneral, BCInflow, BCOutflow, BCInflowSubsonic,
                    BCOutflowSubsonic, BCInflowSupersonic, BCOutflowSupersonic,
                    BCSymmetryPlane, BCTunnelInflow, BCSymmetryPolar,
                    BCTunnelOutflow, BCWallViscous, BCWall, BCWallViscousHeatFlux,
                    BCWallInviscid, BCWallViscousIsothermal, FamilySpecified

BCDataType_t        Dirichlet, Neumann

RigidGridMotionType_t
                    Null, UserDefined, ConstantRate, VariableRate

ArbitraryGridMotionType_t
                    Null, UserDefined, NonDeformingGrid, DeformingGrid

SimulationType_t
                    TimeAccurate, NonTimeAccurate
```

注意，这些关键词需要像它们在这里所呈现出的那样准确写出，包括大写体和小写体的采

用，要通过程序库来识别。

2.5 获取软件和文档

CGNS 中级程序库软件可以从 CGNS 网址下载，地址是 http://www.CGNS.org/。译编的目标程序可用于各种各样的平台。源程序和 makefiles 也可以下载。

本手册以及其他的 CGNS 文档，可用于来自 CGNS 文档网址的 HTML 和 PDF 格式，地址是 http://www.grc.nasa.gov/www/cgns/。

2.6 本手册的构成

以下几节详细讲述中级程序库函数。前 3 节的内容包括打开和关闭 CGNS 文件（第 3 节）、存取 CGNS 数据库中的特殊节点（第 4 节）和错误处理（第 5 节）。其余几节讲述用来阅读、编写和修改 CGNS 数据库中的节点和数据的函数。这几节基本上遵循《SIDS-to-ADF 文件映象手册》的“详细的 CGNS 节点描述”一节中所采用的构成情况。

在每一小节开始处为节点线，列出了可用的 CGNS 节点标示。

然后介绍了一个表格，说明用于中级程序库函数的语法。首先给出 C 函数，接下来是对应的 Fortran 程序。用正体兰色字表示输入变量，斜体红色字表示输出变量。对于每个函数，右边一栏中列出了可用于该函数的模式（阅读、编写和/或修改）。

于是，就可列出并确定输入和输出变量。

3 打开和关闭 CGNS 文件

Functions	Modes
ier = cg_open(char *filename, int mode, *int *fn*);	r w m
ier = cg_version(int fn, *float *version*);	r w m
ier = cg_close(int fn);	r w m
call cg_open_f(filename, mode, *fn*, *ier*)	r w m
call cg_version_f(fn, *version*, *ier*)	r w m
call cg_close_f(fn, *ier*)	r w m

输入/输出

filename　　CGNS 文件的名称，如果需要，还包括路径名。这种字符变量的长度没有限制。（输入）

mode　　用来打开文件的方式。通常支持的方式是 MODE_READ, MODE_WRITE 和 MODE_MODIFY。（输入）

fn　　CGNS 文件索引数。（对 cg_open 为输入；对 cg_version, cg_close 为输出）

version　　CGNS 程序库的版本数。（输出）

ier　　错误状态。（输出）

cg_open 函数必须总是第一个调用。它打开 CGNS 文件，以便阅读和/或编写，并返回一个索引数 fn。索引数用来在接下来的函数调用中识别 CGNS 文件。可以同时打开几个 CGNS 文件。关于同时打开的文件的数量，本限制取决于平台。在 SGI 工作站上，这种限制设为 100（stdio.h

中的参数 FOPEN_MAX）。

可以下列任何方式之一打开文件：

MODE_READ　　只读方式

MODE_WRITE　　只写方式

MODE_MODIFY　　允许读和/或写

cg_close 函数必须总是最后一个调用。它关闭用索引数 fn 表示的 CGNS 文件，并释放保存 CGNS 数据的存储器。当打开文件进行写入时，cg_close 在关闭文件前将所存 CGNS 数据以存储方式写在磁盘上。因此，如果忽略了，CGNS 文件就不会正确写入。

为了减少存储器的使用，并改进执行速度，像网格坐标或流动解这样的大型数据组实际上不存储在存储器中。而只有它们的 ADF ID 数才保存在存储器中，以便将来参考。当 CGNS 文件以写方式打开时，传给程序库的大型数组立即写入 CGNS 文件，直接在根节点之下。当文件关闭时，这些数组移到其在 CGNS 树中的适当位置。

4 导航 CGNS 文件

Functions	Modes
`ier = cg_goto(int fn, int B, ..., "end");`	r w m
`call cg_goto_f(fn, B, ier, ..., 'end')`	r w m

输入/输出

fn　CGNS 文件索引数。（输入）

B　库索引数，其中 1≤B≤nbases。（输入）

…　用来确定到达节点的路径的形参列表。它由不受限制的成对的形参列表组成。每个成对的形参形成 CGNS_NodeLabel, NodeIndex 形式，例如 Zone_t, ZoneIndex。（输入）

end　字符串"end"（或 Fortran 函数中的'end'）必须是最后一个形参，它用来表示形参列表结束了。（输入）

ier　错误状态。下面列出可能值，以及用 cgnslib.h（或 cgnslib_f.h）确定的对应的 C 语言名称（或 Fortran 参数）。

Value	*Name/Parameter*
0	ALL_OK
1	ERROR
2	NODE_NOT_FOUND
3	INCORRECT_PATH

对于非零值，错误信息可能用 cg_error_print ( )打印，如第 5 节中所述。（输出）

这个函数允许通向 CGNS 文件中的任何父节点。父节点是可能具有子节点的一种节点。像 Descriptor_t 这样的不可能有子节点的节点，不受这种函数的支持。

例如

```
ier = cg_goto(fn, B, "Zone_t", Z, "FlowSolutions_t", F, "end");
call cg_goto_f(fn, B, ier, 'Zone_t', Z, 'GasModel_t', 1, 'DataArray_t',
               A, 'end')
```

5 错误处理

Functions	Modes
`error_message = char *cg_get_error();`	r w m
`void cg_error_exit();`	r w m
`void cg_error_print();`	r w m
`call cg_get_error_f(error_message)`	r w m
`call cg_error_exit_f()`	r w m
`call cg_error_print_f()`	r w m

如果在 CGNS 程序库函数的执行过程中发生错误，它由错误状态变量 ier 的非零值表示，那么可以利用 cg_get_error 函数纠正错误信息。于是，cg_error_exit 函数可以用来打印错误信息，并停止程序的执行。换句话说，cg_error_print 可用来打印错误信息，并继续程序的执行。

6 结构分支

6. 1 CGNS 库信息

节点：CGNSBase_t

Functions	Modes
`ier = cg_base_write(int fn, char *basename, int cell_dim, int phys_dim, int *B);`	- w m
`ier = cg_nbases(int fn, int *nbases);`	r - m
`ier = cg_base_read(int fn, int B, char *basename, int *cell_dim, int *phys_dim);`	r - m
`call cg_base_write_f(fn, basename, cell_dim, phys_dim, B, ier)`	- w m
`call cg_nbases_f(fn, nbases, ier)`	r - m
`call cg_base_read_f(fn, B, basename, cell_dim, phys_dim, ier)`	r - m

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（对 cg_base _read 为输入；对 cg_base _write 为输出）
nbases	CGNS 文件 fn 中存在的库的数值。（输出）
basename	库名。（对 cg_base_write 为输入；对 cg_base_read 为输出）
cell_dim	单元的维数；体积单元为 3，面单元为 2。（对 cg_base_write 为输入；对 cg_base_read 为输出）
phys_dim	确定场中矢量所需要的坐标数。（对 cg_base_write 为输入；对 cg_base _read 为输出）
ier	错误状态。（输出）

6. 2 块信息

节点：Zone_t

Functions	Modes
`ier = cg_zone_write(int fn, int B, char *zonename, int *size, ZoneType_t zonetype, int *Z);`	- w m
`ier = cg_nzones(int fn, int B, int *nzones);`	r - m
`ier = cg_zone_read(int fn, int B, int Z, char *zonename, int *size);`	r - m
`ier = cg_zone_type(int fn, int B, int Z, ZoneType_t *zonetype);`	r - m
`call cg_zone_write_f(fn, B, zonename, size, zonetype, Z, ier)`	- w m
`call cg_nzones_f(fn, B, nzones, ier)`	r - m
`call cg_zone_read_f(fn, B, Z, zonename, size, ier)`	r - m
`call cg_zone_type_f(fn, B, Z, zonetype, ier)`	r - m

输入/输出

fn　　CGNS 文件索引数。（输入）

B　　库索引数，其中 1≤B≤nbases。（输入）

Z　　块索引数，其中 1≤Z≤nzones。（对 cg_zone_read, cg_zone_type 为输入；对 cg_zone_write 为输出）

nzones　　B 库中存在的块的数值。（输出）

zonename　　块名称。（对 cg_zone_write 为输入；对 cg_zone_read 为输出）

size　　每个（索引）维数中的顶点、单元和边界顶点的数量。如果在内部节点和边界节点之间给节点分类，那么选择参数 VertexSizeBoundary 必须设为与边界节点数相同。通过缺省，VertexSizeBoundary 等于零，这意味着没对节点分类。这种选择只对非结构块有用。对于结构块，VertexSizeBoundary 在所有方向总是等于 0。

（对 cg_zone_write 为输入；对 cg_zone_read 为输出）

Mesh Type	*Size*

3D structured	`NVertexI, NVertexJ, NVertexK`
	`NCellI, NCellJ, NCellK`
	`NBoundVertexI = 0, NBoundVertexJ = 0,`
	`NBoundVertexK`

2D structured	`NVertexI, NVertexJ`
	`NCellI, NCellJ`
	`NBoundVertexI = 0, NBoundVertexJ = 0`

Unstructured	`NVertex, NCell, NBoundVertex`

(Input for `cg_zone_write`; *output* for `cg_zone_read`)

zonetype　　块的类型。允许的类型为 Structured 和 Unstructured。（对 cg_zone_write 为输入；对 cg_zone_type 为输出）

ier　　错误状态。（输出）

注意，块以字母数字分类，以确保它们总能以相同的顺序检索（对相同模型）。

6.3　模拟类型

节点：SimulationType_t

Functions	Modes
`ier = cg_simulation_type_write(int fn, int B, SimulationType_t SimulationType);`	- w m
`ier = cg_simulation_type_read(int fn, int B, SimulationType_t SimulationType);`	r - m
`call cg_simulation_type_write_f(fn, B, SimulationType, ier)`	- w m
`call cg_simulation_type_read_f(fn, B, SimulationType, ier)`	r - m

输入/输出

fn CGNS 文件索引数。（输入）

B 库索引数，其中 1≤B≤nbases。（输入）

SimulationType 模拟类型。有效的类型是 Null, UserDefined, TimeAccurate 和 NonTimeAccurate 。（对 cg_Simulation_Type_write 为输入：对 cg_Simulation_Type_read 为输出）

ier 错误状态。（输出）

7 描述符

7.1 描述文本

节点：Descriptor_t

Functions	Modes
`ier = cg_descriptor_write(char *name, char *text);`	- w m
`ier = cg_ndescriptors(int *ndescriptors);`	r - m
`ier = cg_descriptor_read(int D, char *name, char **text);`	r - m
`call cg_descriptor_write_f(name, text, ier)`	- w m
`call cg_ndescriptors_f(ndescriptors, ier)`	r - m
`call cg_descriptor_size_f(D, size, ier)`	r - m
`call cg_descriptor_read_f(D, name, text, ier)`	r - m

输入/输出

ndescriptors 本节点下的 Descriptor_t 节点数。（输出）

D 描述符索引数，其中 1≤D≤ndescriptors。（输入）

name Descriptor_t 节点名。(对 cg_descriptor_write 为输入;对 cg_descriptor_read 为输出）

text Descriptor_t 节点中支持的描述符。（对 cg_descriptor_write 为输入；对 cg_descriptor_read 为输出）

size 描述符数据的大小（仅限于 Fortran 接口)。（输出）

ier 错误状态。（输出）

7.2 序数值

节点：Ordinal_t

Functions	Modes
`ier = cg_ordinal_write(int Ordinal);`	- w m
`ier = cg_ordinal_read(int *Ordinal);`	r - m
`call cg_ordinal_write_f(Ordinal, ier)`	- w m
`call cg_ordinal_read_f(Ordinal, ier)`	r - m

输入/输出

Ordinal　　大于 0 的整数。（对 cg_ordinal_write 为输入；对 cg_ordinal_read 为输出）

ier　错误状态。（输出）

8 物理数据

8.1 数组

节点：DataArray_t

Functions	Modes
`ier = cg_array_write(char *ArrayName, DataType_t DataType, int DataDimension, int *DimensionVector, void *Data);`	- w m
`ier = cg_narrays(int *narrays);`	r - m
`ier = cg_array_info(int A, char *ArrayName, DataType_t *DataType, int *DataDimension, int *DimensionVector);`	r - m
`ier = cg_array_read(int A, void *Data);`	r - m
`ier = cg_array_read_as(int A, DataType_t DataType, void *Data);`	r - m
`call cg_array_write_f(ArrayName, DataType, DataDimension, DimensionVector, Data, ier)`	- w m
`call cg_narrays_f(narrays, ier)`	r - m
`call cg_array_info_f(A, ArrayName, DataType, DataDimension, DimensionVector, ier)`	r - m
`call cg_array_read_f(A, Data, ier)`	r - m
`call cg_array_read_as(A, DataType, Data, ier)`	r - m

输入/输出

narrays	本节点下的 DataArray _t 节点数。（输出）
A	描述符索引数，其中 1≤A≤narrays。（输入）
Arrayname	DataArray_t 节点名。(对 cg_array_write 为输入；对 cg_array_info 为输出）
Datatype	DataArray_t 节点中支持的数据类型。允许的类型有 Integer, RealSingle, RealDouble 和 Character。（对 cg_array_write, cg_array_read_as 为输入；对 cg_array_info 为输出）
DataDimension	维数。（对 cg_array_write 为输入；对 cg_array_info 为输出）
DimensionVector	每一维中数据元的数值。（对 cg_array_write 为输入；对 cg_array_info 为输出）
Data	数据组。（对 cg_array_write 为输入；对 cg_array_read，cg_array_read_as 为输出）

ier　　错误状态。（输出）

8.2 数据分类

节点：DataClass_t

Functions	Modes
`ier = cg_dataclass_write(DataClass_t dataclass);`	- w m
`ier = cg_dataclass_read(DataClass_t *dataclass);`	r - m
`call cg_dataclass_write_f(dataclass, ier)`	- w m
`call cg_dataclass_read_f(dataclass, ier)`	r - m

输入/输出

Dataclass　　在此级别下节点的数据类型。CGNS 中通常支持的数据类型见下面所述。（cg_dataclass_write 为输入；对 cg_dataclass_read 为输出）

ier　　错误状态。（输出）

CGNS 中通常支持的数据类型为：

Dimensional　　普通的量纲数据。

NormalizedByDimensional　　通过量纲参考量归一化的无量纲数据。

NormalizedByUnknownDimensional　　所有的场和参考数据都是无量纲的。

NondimensionalParameter　　像 M 数和升力系数这样的无量纲参数。

DimensionlessConstant　　像π这样的常数。

这些类型由 cgnslib.h 中的类型定义 DataClass_t 说明，并作为 cgnslib_f.h 中的参数。

8.3 数据转换因子

节点：DataConversion_t

Functions	Modes
`ier = cg_conversion_write(DataType_t DataType, void *ConversionScale, void *ConversionOffset);`	- w m
`ier = cg_conversion_info(DataType_t *DataType);`	r - m
`ier = cg_conversion_read(void *ConversionScale, void *ConversionOffset);`	r - m
`call cg_conversion_write_f(DataType, ConversionScale, ConversionOffset, ier)`	- w m
`call cg_conversion_info_f(DataType, ier)`	r - m
`call cg_conversion_read_f(ConversionScale, ConversionOffset, ier)`	r - m

输入/输出

DataType　　记录转换因子的数据类型。允许的数据类型有 Realsingle 和 Realdouble。（对 cg_conversion_write 为输入；对 cg_conversion_info　为输出）

ConversionScale　　转换因子。（对 cg_conversion_write 为输入；对 cg_conversion_read 为输出）

ConversionOffset　　补偿因子。（对 cg_conversion_write 为输入；对 cg_conversion_read 为输出）

ier　　错误状态。（输出）

DataConversion_t 数据结构包含将无量纲数据转换成“原始”量纲数据的因子；这些因子是

ConversionScale 和 ConversionOffset。转换过程如下：

```
Data(raw) = Data(nondimensional)*ConversionScale + ConversionOffset
```

8.4 量纲单位

节点：DimensionalUnits_t

Functions	Modes
`ier = cg_units_write(MassUnits_t mass, LengthUnits_t length, TimeUnits_t time, TemperatureUnits_t temperature, AngleUnits_t angle);`	- w m
`ier = cg_units_read(MassUnits_t *mass, LengthUnits_t *length, TimeUnits_t *time, TemperatureUnits_t *temperature, AngleUnits_t *angle);`	r - m
`call cg_units_write_f(mass, length, time, temperature, angle, ier)`	- w m
`call cg_units_read_f(mass, length, time, temperature, angle, ier)`	r - m

输入/输出

mass　　质量单位。允许值有 Null, UserDefined, Kilogram, Gram, Slug 和 PoundMass。（对 cg_units_write 为输入；对 cg_units_read 为输出）

length　　长度单位。允许值有 Null, UserDefined, Meter, Centimeter, Millimeter, Foot 和 Inch。（对 cg_units_write 为输入；对 cg_units_read 为输出）

time　　时间单位。允许值有 Null, UserDefined 和 Second。（对 cg_units_write 为输入；对 cg_units_read 为输出）

temperature　　温度单位。允许值有 Null, UserDefined, Kelvin, Celsius, Rankine 和 Fahrenheit。（对 cg_units_write 为输入；对 cg_units_read 为输出）

angle　　角度单位。允许值有 Null, UserDefined, Degree 和 Radian。（对 cg_units_write 为输入；对 cg_units_read 为输出）

ier　　错误状态。（输出）

支持的单位由 cgnslib.h 中的类型定义说明，并作为 cgnslib_f.h 中的参数。

8.5 量纲指数

节点：DimensionalExponents_t

Functions	Modes
`ier = cg_exponents_write(DataType_t DataType, void *exponents);`	- w m
`ier = cg_exponents_info(DataType_t *DataType);`	r - m
`ier = cg_exponents_read(void *exponents);`	r - m
`call cg_exponents_write_f(DataType, exponents, ier)`	- w m
`call cg_exponents_info_f(DataType, ier )`	r - m
`call cg_exponents_read_f(exponents, ier)`	r - m

输入/输出

DataType　　记录指数的数据类型。允许的数据类型有 Realsingle 和 Realdouble。（对 cg_exponents_write 为输入；对 cg_exponents_info 为输出）

Exponents　　5 个量纲单位的 5 个指数值。（对 cg_exponents_write 为输入；对

cg_exponents_read 为输出）

ier 错误状态。（输出）

9 位置和定位

9.1 网格位置

节点：GridLocation_t

Functions	Modes
`ier = cg_gridlocation_write(GridLocation_t GridLocation);`	- w m
`ier = cg_gridlocation_read(GridLocation_t *GridLocation);`	r - m
`call cg_gridlocation_write_f(GridLocation, ier)`	- w m
`call cg_gridlocation_read_f(GridLocation, ier)`	r - m

输入/输出

GridLocation 在网格中的位置。允许的位置有 Null, UserDefined, Vertex, CellCenter, FaceCenter, IFaceCenter, JfaceCenter, KFaceCenter 和 EdgeCenter 。（对 cg_gridlocation_write 为输入；对 cg_gridlocation_read 为输出）

ier 错误状态。（输出）

9.2 外层网格

节点：Rind_t

Functions	Modes
`ier = cg_rind_write(int *RindData);`	- w m
`ier = cg_rind_read(int *RindData);`	r - m
`call cg_rind_write_f(RindData, ier)`	- w m
`call cg_rind_read_f(RindData, ier)`	r - m

输入/输出

RindData 每个计算方向的外层网格的数值。（对 cg_rind_write 为输入;对 cg_rind_read 为输出）

ier 错误状态。（输出）

10 辅助数据

10.1 参考状态

节点：ReferenceState_t

Functions	Modes
`ier = cg_state_write(char *StateDescription);`	- w n
`ier = cg_state_read(char **StateDescription);`	r - n
`call cg_state_write_f(StateDescription, ier)`	- w n
`call cg_state_size_f(Size, ier)`	r - n
`call cg_state_read_f(StateDescription, ier)`	r - n

输入/输出

StateDescription　参考状态的版本描述。(对 cg_state_write 为输入;对 cg_state_read 为输出)

Size　StateDescription 字符串中字符的序号（仅限于 Fortran 接口)。(输出)

ier　错误状态。(输出)

如果没有规定 StateDescription（即空格字符串），那么 cg_state_write 函数就会产生 ReferenceState_t 节点，并且必须调用。在这种节点产生之后，表征 ReferenceState_t 数据结构的描述符、数据组、数据类型和量纲单位可以添加到这种数据结构中。

cg_state_read 函数阅读当地 ReferenceState_t 节点的 StateDescription。在 CGNS 数据库中没有规定 StateDescription，此函数返回一个空行。

10.2 收敛历程

节点：ConvergenceHistory_t

Functions	Modes
`ier = cg_convergence_write(int niterations, char *NormDefinitions);`	- w n
`ier = cg_convergence_read(int *niterations, char **NormDefinitions);`	r - n
`call cg_convergence_write_f(niterations, NormDefinitions, ier)`	- w n
`call cg_convergence_read_f(niterations, NormDefinitions, ier)`	r - n

输入/输出

niterations　记录收敛信息的迭代数。(对 cg_convergence_write 为输入;对 cg_convergence_read 为输出)

NormDefinitions　数据组中记录的收敛信息的描述。(对 cg_convergence_write 为输入;对 cg_convergence_read 为输出)

ier　错误状态。(输出)

cg_convergence_write 函数产生 ConvergenceHistory_t 节点。当记录收敛随时间变化的数据时，它必须首先调用。可能未规定 NormDefinitions（即空格字符串)。在这种节点产生之后，可以添加表征 ConvergenceHistory_t 数据结构的描述符、数据组、数据类型和量纲单位。

cg_convergence_read 函数阅读 ConvergenceHistory_t 节点的 StateDescription。如果在 CGNS 数据库中没有规定 NormDefinitions，此函数就要返回一个空行。

10.3 集成数据

节点：IntegralData_t

Functions	Modes
ier = cg_integral_write(char *Name);	- w m
ier = cg_nintegrals(int *nintegrals);	r - m
ier = cg_integral_read(int Index, char *Name);	r - m
call cg_integral_write_f(Name, ier)	- w m
call cg_nintegrals_f(nintegrals, ier)	r - m
call cg_integral_read_f(Index, Name, ier)	r - m

输入/输出

Name	IntegralData_t 数据结构的名称。（对 cg_integral_write 为输入；对 cg_ integral _read 为输出）
nintegrals	本节点下的 IntegralData_t 节点数。（输出）
Index	集成数据索引数，其中 1≤Index≤nintegrals。（输入）
ier	错误状态。（输出）

10.4 用户定义的数据

节点：UserDefinedData_t

Functions	Modes
ier = cg_user_data_write(char *Name);	- w m
ier = cg_nuser_data(int *nuserdata);	r - m
ier = cg_user_data_read(int Index, char *Name);	r - m
call cg_user_data_write_f(Name, ier)	- w m
call cg_nuser_data_f(nuserdata, ier)	r - m
call cg_user_data_read_f(Index, Name, ier)	r - m

输入/输出

nuserdata	本节点下的 UserDefinedData_t 节点数。（输出）
Name	UserDefinedData_t 节点的名称。（对 cg_user_data_write 为输入；对 cg_ user_data _read 为输出）
Index	用户定义的数据索引数，其中 1≤Index≤nuserdata。（输入）
ier	错误状态。（输出）

11 网格技术要求

11.1 块网格坐标

节点：GridCoordinates_t

GridCoordinates_t 节点用来描述与特殊块有关的网格。原始网格必须用称为 GridCoordinates 的 GridCoordinates_t 节点来描述。可以采用其他的 GridCoordinates_t 节点，以及用户定义的名称，用来在多个时间步长或迭代时存储网格。除了在 SIDS 和文件映象手册中讨论 GridCoordinates_t 节点外，还在 SIDS 手册中讨论了 ZoneIterativeData_t 和 ArbitraryGridMotion_t 节点。

以下三个函数可用于任何 GridCoordinates_t 节点：

Functions	Modes
`ier = cg_grid_write(int fn, int B, int Z, char *GridCoordName, int *G);`	- w n
`ier = cg_ngrids(int fn, int B, int Z, int *ngrids);`	- w n
`ier = cg_grid_read(int fn, int B, int Z, int G, char *GridCoordName);`	r - n
`call cg_grid_write_f(fn, B, Z, GridCoordName, G, ier)`	- w n
`call cg_ngrids_f(fn, B, Z, ngrids, ier)`	- w n
`call cg_grid_read_f(fn, B, Z, G, GridCoordName, ier)`	r - n

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
G	网格索引数，其中 1≤G≤ngrids。（对 cg_grid_read 为输入；对 cg_grid_write 为输出）
ngrids	对于 Z 块，GridCoordinates_t 节点数。（输出）
GridCoordinateName	GridCoordinates_t 节点的名称。注意，名称“GridCoordinates_t”是专供原始网格用的，并且必须是要定义的第一个 GridCoordinates_t 节点。（对 cg_grid_write 为输入；对 cg_grid_read 为输出）
ier	错误状态。（输出）

下列函数只适用于称为 GridCoordinates 的 GridCoordinates_t 节点，用于某一个块中的原始网格。对于某一个块中附加的 GridCoordinates_t 节点，必须读出其坐标，并利用第 8.1 节中介绍的 cg_array_xxx 函数编写。

Functions	Modes
`ier = cg_coord_write(int fn, int B, int Z, DataType_t datatype, char *coordname, void *coord_array, int *C);`	- w m
`ier = cg_ncoords(int fn, int B, int Z, int *ncoords);`	r - m
`ier = cg_coord_info(int fn, int B, int Z, int C, DataType_t *datatype, char *coordname);`	r - m
`ier = cg_coord_read(int fn, int B, int Z, char *coordname, DataType_t datatype, int *range_min, int *range_max, void *coord_array);`	r - m
`call cg_coord_write_f(fn, B, Z, datatype, coordname, coord_array, C, ier)`	- w m
`call cg_ncoords_f(fn, B, Z, ncoords, ier)`	r - m
`call cg_coord_info_f(fn, B, Z, C, datatype, coordname, ier)`	r - m
`call cg_coord_read_f(fn, B, Z, coordname, datatype, range_min, range_max, coord_array, ier)`	r - m

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
C	坐标数据组索引数，其中 1≤C≤ncoords。（对 cg_coord_info 为输入；对 cg_coord_write 为输出）
ncoords	对于 Z 块，坐标数据组的数值。（输出）

datatype	编写坐标数据组的数据类型。允许的类型有 Realsingle 和 RealDouble。（对 cg_coord_write，cg_coord_read 为输入；对 cg_coord_info 为输出）
rang_min	较低范围索引（例如 imin, jminj, kmin）。（输入）
rang_max	较高范围索引（例如 imax, jmax, kmax）。（输入）
coord_array	规定范围的坐标值的数据组。（对 cg_coord_write 为输入；对 cg_coord_ read 为输出）
ier	错误状态。（输出）

11.2 单元连接性

节点：Elements_t

Functions	Modes
`ier = cg_section_write(int fn, int B, int Z, char *ElementSectionName, ElementType_t type, int start, int end, int nbndry, int *Elements, int *S);`	- w n
`ier = cg_parent_data_write(int fn, int B, int Z, int S, int *ParentData);`	- w n
`ier = cg_nsections(int fn, int B, int Z, int *nsections);`	r - n
`ier = cg_section_read(int fn, int B, int Z, int S, char *ElementSectionName, ElementType_t *type, int *start, int *end, int *nbndry, int *parent_flag);`	r - n
`ier = cg_ElementDataSize(int fn, int B, int Z, int S, int *ElementDataSize);`	r - n
`ier = cg_elements_read(int fn, int B, int Z, int S, int *Elements, int *ParentData);`	r - n
`ier = cg_npe(ElementType_t type, int *npe);`	r w n
`call cg_section_write_f(fn, B, Z, ElementSectionName, type, start, end, nbndry, Elements, S, ier)`	- w n
`call cg_parent_data_write_f(fn, B, Z, S, ParentData, ier)`	- w n
`call cg_nsections_f(fn, B, Z, nsections, ier)`	r - n
`call cg_section_read_f(fn, B, Z, S, ElementSectionName, type, start, end, nbndry, parent_flag, ier)`	r - n
`call cg_ElementDataSize_f(fn, B, Z, S, ElementDataSize, ier)`	r - n
`call cg_elements_read_f(fn, B, Z, S, Elements, ParentData, ier)`	r - n
`call cg_npe_f(type, npe, ier)`	r w n

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
ElementSectionName	Elements_t 节点的名称。（对 cg_section_write 为输入；对 cg_section_ read 为输出）
type	单元的类型。见 2.4 节中 ElementType_t 的符合条件的类型。（对 cg_section_write，cg_npe 为输入；对 cg_section_ read 为输出）
start	数据段中第一个单元的索引。（对 cg_section_write 为输入；对 cg_section_ read 为输出）
end	数据段中最后一个单元的索引。（对 cg_section_write 为输入；对 cg_section_ read 为输出）
nbndry	数据段中最后一个边界单元的索引。如果单元未分类，则设为 0（对

	cg_section_write 为输入；对 cg_section_ read 为输出）
nsections	较低范围索引（例如 imin, jminj, kmin）。（输入）
S	单元段索引数，其中 1≤S≤nsections。（对 cg_parent_ data_write, cg_section_read, cg_ElementDataSize, cg_elements_ read 为输入；对 cg_section_ write 为输出）
parent_flag	是否确定父型数据的标志。若存在父型数据，则 parent_flag 设为 1；否则设为 0。（输出）
ElementDataSize	单元连接性数据的数值。（输出）
Elements	单元连接性数据。（对 cg_section_write 为输入；对 cg_elements_ read 为输出）
ParentData	对于边界或接口单元，这个数据组包括共用单元的小单元和小单元界面。（输出）
npe	type 类型的单元的节点数。（输出）
ier	错误状态。（输出）

12 结果数据

12.1 流动解

节点：FlowSolution_t

以下所述三个函数用来产生并获得有关 FlowSolution_t 节点的信息。

Functions	Modes
`ier = cg_sol_write(int fn, int B, int Z, char *solname, GridLocationType_t location, int *S);`	`- w n`
`ier = cg_nsols(int fn, int B, int Z, int *nsols);`	`r - n`
`ier = cg_sol_info(int fn, int B, int Z, int S, char *solname, GridLocationType_t *location);`	`r - n`
`call cg_sol_write_f(fn, B, Z, solname, location, S, ier)`	`- w n`
`call cg_nsols_f(fn, B, Z, nsols, ier)`	`r - n`
`call cg_sol_info_f(fn, B, Z, S, solname, location, ier)`	`r - n`

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
S	流动解索引数，其中 1≤S≤nsols。（对 cg_sol_info 为输入；对 cg_sol_ write 为输出)
nsols	Z 块的流动解的数值。（输出）
solname	流动解的名称。（对 cg_sol_write 为输入；对 cg_sol_info 为输出)
location	记录结果的网格位置。通常允许的位置有 Vertex, CellCenter, IFaceCenter, JFaceCenter 和 KFaceCenter。（对 cg_sol_write 为输入；对 cg_sol_info 为输出）
ier	错误状态。（输出）

下列函数用来读和写存储在 FlowSolution_t 节点中的结果数据组。

Functions	Modes
`ier = cg_field_write(int fn, int B, int Z, int S, DataType_t datatype, char *fieldname, void *solution_array, int *F);`	- w n
`ier = cg_nfields(int fn, int B, int Z, int S, int *nfields);`	r - n
`ier = cg_field_info(int fn, int B, int Z, int S, int F, DataType_t *datatype, char *fieldname);`	r - n
`ier = cg_field_read(int fn, int B, int Z, int S, char *fieldname, DataType_t datatype, int *range_min, int *range_max, void *solution_array);`	r - n
`call cg_field_write_f(fn, B, Z, S, datatype, fieldname, solution_array, F, ier)`	- w n
`call cg_nfields_f(fn, B, Z, S, nfields, ier)`	r - n
`call cg_field_info_f(fn, B, Z, S, F, datatype, fieldname, ier)`	r - n
`call cg_field_read_f(fn, B, Z, S, fieldname, datatype, range_min, range_max, solution_array, ier)`	r - n

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
S	流动解索引数，其中 1≤S≤nsols。（输入）
F	结果数据组索引数，其中 1≤F≤nfields。（对 cg_field_info 为输入；对 cg_field_write 为输出）
nfields	流动解 S 中的数据组的数值。（输出）
datatype	写结果数据组的数据类型。允许的数据类型有 Integer, Realsingle 和 Realdouble。（对 cg_field_write，cg_field_read 为输入；对 cg_field_info 为输出）
fieldname	结果数据组的名称。在命名结果数据组时，强烈建议采用 SIDS 命名习惯，以确保文件兼容。（对 cg_field_write，cg_field_read 为输入；对 cg_field_info 为输出）
rang_min	较低范围索引（例如 imin, jminj, kmin）。（输入）
rang_max	较高范围索引（例如 imax, jmax, kmax）。（输入）
solution_array	规定范围的结果值的数据组。（对 cg_field_write 为输入；对 cg_field_ read 为输出）
ier	错误状态。（输出）

对于由 rang_min 和 rang_max 规定的范围，函数 cg_field_read 返回结果数据组的名称 fieldname。数据组返回到用于 datatype 所需要的数据类型。此数据类型不需要与在文件中存储数据的类型相同。在 CGNS 文件中以双精度存储的结果数据组不能返回到像单精度一样的应用，反之亦然。

在 Fortran 中，如果变量 Solution_array 定义为 2D 或 3D 的数据组，那么用户就必须保证每个方向分配的维数与要求的范围所需要的维数精确匹配。例如，为了阅读具有 3×5 顶点的 2D 块的所有结果值，就必须用(3,5)的大小来表示数据组。如果数据组表示得更大(比如(6,9))，那么返回的数组值就会出错。为避免发生这样的问题，数据组可简单地分配为 1D 数据组(比如(15))。当采用 1D 数据组时，所表示的大小可以大于要求的数据的数值。

12.2 离散数据

节点：Discretedata_t

Discretedata_t 节点用来存储数据场，这种数据场通常并不标识为流动解的一部分，比如通量或方程截断误差。

Functions	Modes
`ier = cg_discrete_write(int fn, int B, int Z, char *DiscreteName, int *D);`	- w n
`ier = cg_ndiscrete(int fn, int B, int Z, int *ndiscrete);`	r - n
`ier = cg_discrete_read(int fn, int B, int Z, int D, char *DiscreteName);`	r - n
`call cg_discrete_write_f(fn, B, Z, DiscreteName, D, ier)`	- w n
`call cg_ndiscrete_f(fn, B, Z, ndiscrete, ier)`	r - n
`call cg_discrete_read_f(fn, B, Z, D, DiscreteName, ier)`	r - n

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
D	离散数据索引数，其中 1≤D≤ndiscrete。（对 cg_discrete_read 为输入；对 cg_discrete_write 为输出）
ndiscrete	在 Z 块内，Discretedata_t 数据结构的数值。（输出）
DiscreteName	Discretedata_t 数据结构的名称。（对 cg_discrete_write 为输入；对 cg_discrete_read 为输出）
ier	错误状态。（输出）

13 网格连接性

13.1 一对一连接性

节点：GridConnectivityltol_t

下列两个函数可用于获得 CGNS 数据库中关于所有一对一块接口的信息。

Functions	Modes
`ier = cg_nitoi_global(int fn, int B, int *nitoi_global);`	r - n
`ier = cg_itoi_read_global(int fn, int B, char **connectname, char **zonename, char **donorname, int **range, int **donor_range, int **transform);`	r - n
`call cg_nitoi_global_f(fn, B, nitoi_global, ier)`	r - n
`call cg_itoi_read_global_f(fn, B, connectname, zonename, donorname, range, donor_range, transform, ier)`	r - n

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
nltol_global	B 库中一对一接口的总数，存储在 GridConnectivityltol_t 节点内。（也就是说，这不包括可能存储在 GridConnectivity_t 节点内的一对一接口，用于通用的块接口。）注意，函数 cg_nltol（下面予以介绍）可用来获得特

	定块内一对一接口的数值。（输出）
connectname	接口名称。（输出）
zonename	第一个块的名称，用于 B 库中所有的一对一接口。（输出）
donorname	第二个块的名称，用于 B 库中所有的一对一接口。（输出）
range	第一个块的点的范围，用于 B 库中所有的一对一接口。（输出）
donor_range	本块的点的范围，用于 B 库中所有的一对一接口。（输出）
transform	确定两个块相对取向的变换矩阵的简化符号。给出这个变换，用于 B 库中所有的一对一接口。详细情况见《SIDS 手册》中 GridConnectivityltol_t 的描述。（输出）
ier	错误状态。（输出）

下列函数用于读出和写入特定块内的一对一连接性数据。

Functions	Modes
`ier = cg_1to1_write(int fn, int B, int Z, char *connectname, char *donorname, int *range, int *donor_range, int *transform, int *I);`	- w n
`ier = cg_n1to1(int fn, int B, int Z, int *n1to1);`	r - n
`ier = cg_1to1_read(int fn, int B, int Z, int I, char *connectname, char *donorname, int *range, int *donor_range, int *transform);`	r - n
`call cg_1to1_write_f(fn, B, Z, connectname, donorname, range, donor_range, transform, I, ier)`	- w n
`call cg_n1to1_f(fn, B, Z, n1to1, ier)`	r - n
`call cg_1to1_read_f(fn, B, Z, I, connectname, donorname, range, donor_range, transform, ier)`	r - n

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
I	接口索引数，其中 1≤I≤nltol。（对 cg_ltol_read 为输入；对 cg_ltol_write 为输出）
nltol	Z 块中一对一接口的数值，存储在 GridConnectivityltol_t 节点内。（也就是说，这不包括可能存储在 GridConnectivity_t 节点内的一对一接口，用于通用的块接口。）（输出）
connectname	接口名称。（对 cg_ltol_write 为输入；对 cg_ltol_read 为输出）
donorname	与本块邻接的块的名称。（对 cg_ltol_write 为输入；对 cg_ltol_read 为输出）
range	本块的点的范围。（对 cg_ltol_write 为输入；对 cg_ltol_read 为输出）
donor_range	donor 块的点的范围。（对 cg_ltol_write 为输入；对 cg_ltol_read 为输出）
transform	确定两个块相对取向的变换矩阵的简化符号。详细情况见《SIDS 手册》中 GridConnectivityltol_t 的描述。（对 cg_ltol_write 为输入；对 cg_ltol_read 为输出）
ier	错误状态。（输出）

13.2 广义连接性

节点：GridConnectivity_t

Functions	Modes
`ier = cg_conn_write(int fn, int B, int Z, char *connectname, GridLocationType_t location, GridConnectivityType_t connect_type, PointSetType_t ptset_type, int npnts, int *pnts, char *donorname, ZoneType_t donor_zonetype, PointSetType_t donor_ptset_type, DataType_t donor_datatype, int ndata_donor, void *donor_data, int *I);`	- w m
`ier = cg_nconns(int fn, int B, int Z, int *nconns);`	r - m
`ier = cg_conn_info(int fn, int B, int Z, int I, char *connectname, GridLocationType_t *location, GridConnectivityType_t *connect_type, PointSetType_t *ptset_type, int *npnts, char *donorname, ZoneType_t *donor_zonetype, PointSetType_t *donor_ptset_type, DataType_t *donor_datatype, int *ndata_donor);`	r - m
`ier = cg_conn_read(int fn, int B, int Z, int I, int *pnts, DataType_t donor_datatype, void *donor_data);`	r - m
`call cg_conn_write_f(fn, B, Z, connectname, location, connect_type, ptset_type, npnts, pnts, donorname, donor_zonetype, donor_ptset_type, donor_datatype, ndata_donor, donor_data, I, ier)`	- w m
`call cg_nconns_f(fn, B, Z, nconns, ier)`	r - m
`call cg_conn_info_f(fn, B, Z, I, connectname, location, connect_type, ptset_type, npnts, donorname, donor_zonetype, donor_ptset_type, donor_datatype, ndata_donor, ier)`	r - m
`call cg_conn_read_f(fn, B, Z, I, pnts, donor_datatype, donor_data, ier)`	r - m

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
I	离散数据索引数，其中 1≤I≤nconns。（对 cg_conn_info, cg_conn_read 为输入；对 cg_conn_write 为输出）
nconns	Z 块的接口名称。（输出）
connectname	接口名称。（对 cg_conn_write 为输入；对 cg_conn_info 为输出）
lacation	用于确定点集的网格的位置。通常允许的位置有 Vertex 和 CellCenter。（对 cg_conn_write 为输入；对 cg_conn_info 为输出）
connect_type	定义的接口的类型。允许的类型有 Overset, Abutting 和 Abuttingltol。（对 cg_conn_write 为输入；对 cg_conn_info 为输出）
ptset_type	定义本块中接口的点集的类型；或者是 PointRange 或者是 PointList。（对 cg_conn_write 为输入；对 cg_conn_info 为输出）
donor_ptset_type	定义 donor 块中接口的点集的类型；或者是 PointListDonor 或者是 CellListDonor。（对 cg_conn_write 为输入；对 cg_conn_info 为输出）
npnts	定义本块中接口的点的数值。对于 PointRange 的 ptset_type，npnts 也有两个。对于 PointList 的 ptset_type，npnts 是 PointList 中的点的数值。（对 cg_conn_write 为输入；对 cg_conn_info 为输出）
ndata_donor	定义 donor 块中接口的点或单元的数值。（对 cg_conn_write 为输入；对 cg_conn_info 为输出）

donorname	与本块邻接的块的名称。（对 cg_conn_write 为输入；对 cg_conn_info 为输出）
donor_datatype	在文件中存储 donor 点的数据类型。与中级程序库的 2.0 版本一样，唯一允许的类型是 Integer。只是对反向兼容性，在这些函数中才进行 donor_datatype 的讨论。（对 cg_conn_write，cg_conn_read 为输入；对 cg_conn_info 为输出）
pnts	定义本块中接口的点的数据组。（对 cg_conn_write 为输入；对 cg_conn_read 为输出）
donor_data	定义 donor 块中接口的点或单元的数据组。（对 cg_conn_write 为输入；对 cg_conn_read 为输出）
donor_zonetype	donor 块的类型。允许的类型有 Structured 和 Unstructured。（对 cg_conn_write 为输入；对 cg_conn_info 为输出）
ier	错误状态。（输出）

注意，利用 cg_goto 和 cg_array_xxx 函数来存取保存在 InterpolantsDonor 数据组中的内插因子，分别在第 4 节和第 8.1 节中进行了讨论。

13.3 重叠洞

节点：Overset Holes_t

Functions	Modes
`ier = cg_hole_write(int fn, int B, int Z, char *holename, GridLocationType_t location, PointSetType_t ptset_type, int nptsets, int npnts, int *pnts, int *I);`	- w m
`ier = cg_nholes(int fn, int B, int Z, int *nholes);`	r - m
`ier = cg_hole_info(int fn, int B, int Z, int I, char *holename, GridLocationType_t *location, PointSetType_t *ptset_type, int *nptsets, int *npnts);`	r - m
`ier = cg_hole_read(int fn, int B, int Z, int I, int *pnts);`	r - m
`call cg_hole_write_f(fn, B, Z, holename, location, ptset_type, nptsets, npnts, pnts, I, ier)`	- w m
`call cg_nholes_f(fn, B, Z, nholes, ier)`	r - m
`call cg_hole_info_f(fn, B, Z, I, holename, location, ptset_type, nptsets, npnts, ier)`	r - m
`call cg_hole_read_f(fn, B, Z, I, pnts, ier)`	r - m

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
I	重叠洞索引数，其中 1≤I≤nholes。（对 cg_hole_info, cg_hole_read 为输入；对 cg_hole_write 为输出）
nholes	Z 块中重叠洞的数值。（输出）
holename	重叠洞的名称。（对 cg_hole_write 为输入；对 cg_hole_info 为输出）
lacation	用于确定点集的网格的位置。通常允许的位置有 Vertex 和 CellCenter。（对 cg_hole_write 为输入；对 cg_hole_info 为输出）
ptset_type	可利用点或单元的范围，或者利用重叠洞中所有点或单元的离散列表来确定

	重叠洞的范围。如果采用点或单元的范围，则 ptset_type 设为 PointRange。如果采用点或单元的离散列表，则 ptset_type 设为 PointList。（对 cg_hole_write 为输入；对 cg_hole_info 为输出）
nptsets	用来确定洞的点集的数值。如果 ptset_type 为 PointRange，则可采用几个点集。如果 ptset_type 为 PointList，则只允许采用一个点集。（对 cg_hole_write 为输入；对 cg_hole_info 为输出）
npnts	点集中点（或单元）的数值。对于 PointRange 的 ptset_type，npnts 也有两个值。对于 PointList 的 ptset_type，npnts 为 PointList 中点或单元的数值。（对 cg_hole_write 为输入；对 cg_hole_info 为输出）
pnts	点集中点或单元的数据组。（对 cg_hole_write 为输入；对 cg_hole_read 为输出）
ier	错误状态。（输出）

14 边界条件

14.1 边界条件类型和位置

节点：BC_t

Functions	Modes
`ier = cg_boco_write(int fn, int B, int Z, char *boconame, BCType_t bocotype, PointSetType_t ptset_type, int npnts, int *pnts, int *BC);`	- w m
`ier = cg_boco_normal_write(int fn, int B, int Z, int BC, int *NormalIndex, int NormalListFlag, DataType_t NormalDataType, void *NormalList);`	- w m
`ier = cg_nbocos(int fn, int B, int Z, int *nbocos);`	r - m
`ier = cg_boco_info(int fn, int B, int Z, int BC, char *boconame, BCType_t *bocotype, PointSetType_t *ptset_type, int *npnts, int *NormalIndex, int *NormalListFlag, DataType_t *NormalDataType, int *ndataset);`	r - m
`ier = cg_boco_read(int fn, int B, int Z, int BC, int *pnts, void *NormalList);`	r - m
`call cg_boco_write_f(fn, B, Z, boconame, bocotype, ptset_type, npnts, pnts, BC, ier)`	- w m
`call cg_boco_normal_write_f(fn, B, Z, BC, NormalIndex, NormalListFlag, NormalDataType, NormalList, ier)`	- w m
`call cg_nbocos_f(fn, B, Z, nbocos, ier)`	r - m
`call cg_boco_info_f(fn, B, Z, BC, boconame, bocotype, ptset_type, npnts, NormalIndex, NormalListFlag, NormalDataType, ndataset, ier)`	r - m
`call cg_boco_read_f(fn, B, Z, BC, pnts, NormalList, ier)`	r - m

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）

BC	边界条件索引数，其中 1≤BC≤nbocos。（对 cg_boco_normal_write, cg_boco_ info, cg_boco_read 为输入；对 cg_boco_write 为输出）
nbocos	Z 块中边界条件的数值。（输出）
boconame	边界条件的名称。（对 cg_boco_write 为输入；对 cg_boco_info 为输出）
bocotype	规定的边界条件的类型。见 2.4 节中 BCType 的符合条件的类型。注意，如果 bocotype 为 FamilySpecified，那么就要对边界所属的族规定边界条件的类型。可利用 cg_fambc_read 和 cg_fambc_write 读和写有关族的边界条件的类型，如 16.3 节所述。（对 cg_boco_write 为输入；对 cg_boco_info 为输出）
ptset_type	可利用点或单元的范围，或者利用（应用边界条件的）所有点或单元的离散列表来确定边界条件的范围。如果采用点或单元的范围，则 ptset_type 设为 PointRange。如果采用点或单元的离散列表，则 ptset_type 设为 PointList。（对 cg_boco_write 为输入；对 cg_boco_info 为输出）
npnts	定义边界条件范围的点集中点的数值。对于 PointRange 的 ptset_type，npnts 也有两个值。对于 PointList 的 ptset_type，npnts 为 PointList 中点的数值。（对 cg_boco_write 为输入；对 cg_boco_info 为输出）
pnts	定义边界条件范围的点数据组。（对 cg_boco_write 为输入；对 cg_boco_read 为输出）
NormalIndex	表示边界条件补块法向的计算坐标方向的索引矢量。（对 cg_boco_normal_write 为输入；对 cg_boco_info 为输出）
NormalListFlag	对于 cg_boco_normal_write，NormalListFlag 是表示在 NormalList 中是否定义法向的一个标志；如果要定义，则为 1，如果不定义，则为 0。对于 cg_boco_info，如果在 NormalList 中定义法向，NormalListFlag 则为补块数 phys_dim 中点的数值，定义场中矢量所需要的坐标的数值。如果在 NormalList 中不定义法向，NormalListFlag 则为 0。（对 cg_boco_normal_write 为输入；对 cg_boco_info 为输出）
NormalDataType	用来定义法向的数据类型。允许的类型有 RealSingle 和 RealDouble。（对 cg_boco_normal_write 为输入；对 cg_boco_info 为输出）
NormalList	指向块内部的边界条件补块的矢量列表法向。（对 cg_boco_normal_ write 为输入；对 cg_boco_info 为输出）
ndataset	本边界条件的边界条件数据组的数值。（输出）
ier	错误状态。（输出）

14.2 边界条件数据组

节点：BCDataSet_t

Functions	Modes
`ier = cg_dataset_write(int fn, int B, int Z, int BC, char *DatasetName, BCType_t BCType, int *Dset);`	- w m
`ier = cg_dataset_read(int fn, int B, int Z, int BC, int DSet, char *DatasetName, BCType_t *BCType, int *DirichletFlag, int *NeumannFlag);`	r - m
`call cg_dataset_write_f(fn, B, Z, BC, DatasetName, BCType, Dset, ier)`	- w m
`call cg_dataset_read_f(fn, B, Z, BC, DSet, DatasetName, BCType, DirichletFlag, NeumannFlag, ier)`	r - m

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
BC	边界条件索引数，其中 1≤BC≤nbocos。（输入）
Dset	数据组索引数，其中 1≤Dset≤ndataset。（对 cg_dataset_read 为输入；对 cg_dataset_write 为输出）
DatasetName	数据组名称。（对 cg_dataset_write 为输入；对 cg_dataset_read 为输出）
BCType	数据组的边界条件类型。（对 cg_dataset_write 为输入；对 cg_dataset_read 为输出）
DirichletFlag	显示数据组是否包括 Dirichlet 数据的标志。（输出）
NeumannFlag	显示数据组是否包括 Neumann 数据的标志。（输出）
ier	错误状态。（输出）

14.3 边界条件数据

节点：BCData_t

Functions	Modes
`ier = cg_bcdata_write(int fn, int B, int Z, int BC, int Dset, BCDataType_t BCDataType);`	- w m
`call cg_bcdata_write_f(fn, B, Z, BC, Dset, BCDataType, ier)`	- w m

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	块索引数，其中 1≤Z≤nzones。（输入）
BC	边界条件索引数，其中 1≤BC≤nbocos。（输入）
Dset	数据组索引数，其中 1≤Dset≤ndataset。（输入）
BCDataType	数据组中边界条件的类型。允许的边界条件类型有 Dirichlet 和 Neumann（输入）
ier	错误状态。（输出）

15 方程技术要求

15.1 流动方程组

节点：FlowEquationSet_t

Functions	Modes
`ier = cg_equationset_write(int EquationDimension);`	`- w m`
`ier = cg_equationset_read(int *EquationDimension, int *GoverningEquationsFlag, int *GasModelFlag, int *ViscosityModelFlag, int *ThermalConductModelFlag, int *TurbulenceClosureFlag, int *TurbulenceModelFlag);`	`r - m`
`ier = cg_equationset_chemistry_read(int *ThermalRelaxationFlag, int *ChemicalKineticsFlag);`	`r - m`
`call cg_equationset_write_f(EquationDimension, ier)`	`- w m`
`call cg_equationset_read_f(EquationDimension, GoverningEquationsFlag, GasModelFlag, ViscosityModelFlag, ThermalConductModelFlag, TurbulenceClosureFlag, TurbulenceModelFlag, ier)`	`r - m`
`call cg_equationset_chemistry_read_f(ThermalRelaxationFlag, ChemicalKineticsFlag, ier)`	`r - m`

输入/输出

EquationDimension　控制方程的维数：它是描述流动的空间变量的数值。（对 cg_equationset_write 为输入；对 cg_equationset_info 为输出）

GoverningEquationsFlag　表示此 FlowEquationSet_t 节点是否包括控制方程定义的标志：如果不包括则为 0，如果包括则为 1。（输出）

GasModelFlag　表示此 FlowEquationSet_t 节点是否包括气体模型定义的标志：如果不包括则为 0，如果包括则为 1。（输出）

ViscosityModelFlag　表示此 FlowEquationSet_t 节点是否包括粘性模型定义的标志：如果不包括则为 0，如果包括则为 1。（输出）

ThermalConductModelFlag　表显示此 FlowEquationSet_t 节点是否包括热传导模型定义的标志：如果不包括则为 0，如果包括则为 1。（输出）

TurbulenceClosureFlag　表示此 FlowEquationSet_t 节点是否包括湍流结束定义的标志：如果不包括则为 0，如果包括则为 1。（输出）

TurbulenceModelFlag　表示此 FlowEquationSet_t 节点是否包括湍流模型定义的标志：如果不包括则为 0，如果包括则为 1。（输出）

ThermalRelaxationFlag　表示此 FlowEquationSet_t 节点是否包括热松弛模型定义的标志：如果不包括则为 0，如果包括则为 1。（输出）

ChemicalKineticsFlag　表示此 FlowEquationSet_t 节点是否包括化学动力学模型定义的标志：如果不包括则为 0，如果包括则为 1。（输出）

ier　错误状态。（输出）

15.2 控制方程

节点：GoverningEquations_t

Functions	Modes
`ier = cg_governing_write(GoverningEquationsType_t Equationstype);`	- w m
`ier = cg_governing_read(GoverningEquationsType_t *EquationsType);`	r - m
`ier = cg_diffusion_write(int *diffusion_model);`	- w m
`ier = cg_diffusion_read(int *diffusion_model);`	r - m
`call cg_governing_write_f(EquationsType, ier)`	- w m
`call cg_governing_read_f(EquationsType, ier)`	r - m
`call cg_diffusion_write_f(diffusion_model, ier)`	- w m
`call cg_diffusion_read_f(diffusion_model, ier)`	r - m

输入/输出

EquationsType　控制方程的类型。允许的类型有 Null, UserDefined, FullPotential, Euler, NSLaminar, NSTurbulent, NSLaminarIncompressibleNS 和 NSTurbulentIncompressible。（对 cg_governing_write 为输入；对 cg_governing _read 为输出）

diffusion_model　确定控制方程中包括哪些耗散项的标志。它仅适用于具有结构网格的 N-S 方程。详细情况见 SIDS 手册中 GoverningEquations_t 的讨论。（对 cg_diffusion_write 为输入；对 cg_diffusion_read 为输出）

ier　错误状态。（输出）

15.3 辅助模型

节点：GasModel_t, ViscosityModel_t, ThermalConductivityModel_t, TurbulenceClosure_t, TurbulenceModel_t, ThermalRelaxationModel_t, ChemicalKineticsModel_t

Functions	Modes
`ier = cg_model_write(char *ModelLabel, ModelType_t ModelType);`	- w m
`ier = cg_model_read(char *ModelLabel, ModelType_t *ModelType);`	r - m
`call cg_model_write_f(ModelLabel, ModelType, ier)`	- w m
`call cg_model_read_f(ModelLabel, ModelType, ier)`	r - m

输入/输出

ModelLabel　定义模型的 CGNS 标示。CGNS 支持的模型为：

- GasModel_t
- ViscosityModel_t
- ThermalConductivityModel_t
- TurbulenceClosure_t
- TurbulenceModel_t
- ThermalRelaxationModel_t
- ChemicalKineticsModel_t

（输入）

ModelType　允许选择 ModelLabel 的模型类型之一（见下）。（对 cg_model_write 为输入；对 cg_model_read 为输出）

ier　错误状态。（输出）

可供各种模型所用的类型是：

GasModel_t	Null, UserDefined, Ideal, VanderWaals, CaloricallyPerfect, ThermallyPerfect, ConstantDensity, RedlichKwong
ViscosityModel_t	Null, UserDefined, Constant, PowerLaw, SutherlandLaw
ThermalConductivityModel_t	Null, UserDefined, PowerLaw, SutherlandLaw, ConstantPrandtl
TurbulenceModel_t	Null, UserDefined, Algebraic_BaldwinLomax, Algebraic_CebeciSmith, HalfEquation_JohnsonKing, OneEquation_BaldwinBarth, OneEquation_SpalartAllmaras, TwoEquation_JonesLaunder, TwoEquation_MenterSST, TwoEquation_Wilcox
TurbulenceClosure_t	Null, UserDefined, EddyViscosity, ReynoldsStress, ReynoldsStressAlgebraic
ThermalRelaxationModel_t	Null, UserDefined, Frozen, ThermalEquilib, ThermalNonequilib
ChemicalKineticsModel_t	Null, UserDefined, Frozen, ChemicalEquilibCurveFit, ChemicalEquilibMinimization, ChemicalNonequilib

16 族

16.1 族定义

节点：Family_t

Functions	Modes
`ier = cg_family_write(int fn, int B, char *FamilyName, int *Fam);`	- w m
`ier = cg_nfamilies(int fn, int B, int *nfamilies);`	r - m
`ier = cg_family_read(int fn, int B, int Fam, char *FamilyName, int *nFamBC, int *nGeo);`	r - m
`call cg_family_write_f(fn, B, FamilyName, Fam, ier)`	- w m
`call cg_nfamilies_f(fn, B, nfamilies, ier)`	r - m
`call cg_family_read_f(fn, B, Fam, FamilyName, nFamBC, nGeo, ier)`	r - m

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
nfamilies	B 库中族的数值。（输出）
Fam	族索引数，其中 1≤Fam≤nfamilies。（对 cg_family_read 为输入；对 cg_family_write 为输出）
FamilyName	族的名称。（对 cg_family_write 为输入；对 cg_family_read 为输出）
nFamBC	用于此族的边界条件的数值，应该是 0 或 1。（输出）
nGeo	用于此族的几何参考的数值。（输出）
ier	错误状态。（输出）

16.2 几何参考

节点：GeometryReference_t

Functions	Modes
`ier = cg_geo_write(int fn, int B, int Fam, char *GeoName, char*FileName, char *CADSystem, int *G);`	- w m
`ier = cg_geo_read(int fn, int B, int Fam, int G, char *GeoName, char *FileName, char *CADSystem, int *nparts);`	r - m
`ier = cg_part_write(int fn, int B, int Fam, int G, char *PartName, int *P);`	- w m
`ier = cg_part_read(int fn, int B, int Fam, int G, int P, char *PartName);`	r - m
`call cg_geo_write_f(fn, B, Fam, GeoName, FileName, CADSystem, G, ier)`	- w m
`call cg_geo_read_f(fn, B, Fam, G, GeoName, FileName, CADSystem, nparts, ier)`	r - m
`call cg_part_write_f(fn, B, Fam, G, PartName, P, ier)`	- w m
`call cg_part_read_f(fn, B, Fam, G, P, PartName, ier)`	r - m

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Fam	族索引数，其中 1≤Fam≤nfamilies。（输入）
G	几何参考索引数，其中 1≤G≤nGeo。（对 cg_geo_read, cg_part_write 和 cg_part_read 为输入；对 cg_geo_write 为输出）
P	几何实体索引数，其中 1≤P≤nparts。（对 cg_part_read 为输入；对 cg_part_write 为输出）
GeoName	GeometryReference_t 节点的名称。（对 cg_geo_write 为输入；对 cg_geo_read 为输出）
FileName	几何文件的名称。（对 cg_geo_write 为输入；对 cg_geo_read 为输出）
CADSystem	几何格式。（对 cg_geo_write 为输入；对 cg_geo_read 为输出）
nparts	几何实体的序号。（输出）
PartName	FileName 文件中几何实体的数值。（对 cg_geo_write 为输入；对 cg_geo_read 为输出）
ier	错误状态。（输出）

16.3 族边界条件

节点：FamilyBC_t

Functions	Modes
`ier = cg_fambc_write(int fn, int B, int Fam, char *FamBCName, BCType_t BCType, int *BC);`	- w n
`ier = cg_fambc_read(int fn, int B, int Fam, int BC, char *FamBCName, BCType_t *BCType);`	r - n
`call cg_fambc_write_f(fn, B, Fam, FamBCName, BCType, BC, ier)`	- w n
`call cg_fambc_read_f(fn, B, Fam, BC, FamBCName, BCType, ier)`	r - n

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Fam	族索引数，其中 1≤Fam≤nfamilies。（输入）
BC	族边界条件索引数，必须等于 1。（对 cg_fambc_read 为输入；对 cg_fambc_write 为输出）
FamBCName	FamilyBC_t 节点的名称。（对 cg_fambc_write 为输入；对 cg_fambc_read 为输出）
BCType	用于族的边界条件的类型。见 2.4 节中 BCType_t 的符合条件的类型。（对 cg_fambc_write 为输入；对 cg_fambc_read 为输出）
ier	错误状态。（输出）

16.4 族名称

节点：FamilyName_t

Functions	Modes
`ier = cg_famname_write(char *FamilyName);`	- w n
`ier = cg_famname_read(char *FamilyName);`	r - n
`call cg_famname_write_f(FamilyName, ier)`	- w n
`call cg_famname_read_f(FamilyName, ier)`	r - n

输入/输出

FamilyName	族名称。（对 cg_famname_write 为输入；对 cg_famname_read 为输出）
ier	错误状态。（输出）

17 随时间变化的数据

17.1 库迭代数据

节点：BaseIterativeData_t

Functions	Modes
`ier = cg_biter_write(int fn, int B, char *BaseIterName, int Nsteps);`	- w n
`ier = cg_biter_read(int fn, int B, char *BaseIterName, int *Nsteps);`	r - n
`call cg_biter_write_f(fn, B, BaseIterName, Nsteps, ier)`	- w n
`call cg_biter_read_f(fn, B, BaseIterName, Nsteps, ier)`	r - n

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
BaseIterName	BaseIterativeData_t 节点的名称。（对 cg_biter_write 为输入；对 cg_biter_read 为输出）
Nsteps	时间步长或迭代的数值。（对 cg_biter_write 为输入；对 cg_biter_read 为输出）
ier	错误状态。（输出）

17.2 块迭代数据

节点：ZoneIterativeData_t

Functions	Modes
`ier = cg_ziter_write(int fn, int B, int Z, char *ZoneIterName);`	- w n
`ier = cg_ziter_read(int fn, int B, int Z, char *ZoneIterName);`	r - n
`call cg_ziter_write_f(fn, B, Z, ZoneIterName, ier)`	- w n
`call cg_ziter_read_f(fn, B, Z, ZoneIterName, ier)`	r - n

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	族索引数，其中 1≤Z≤nzones。（输入）
ZoneIterName	ZoneIterativeData_t 节点的名称。（对 cg_ziter_write 为输入；对 cg_ziter_read 为输出）
ier	错误状态。（输出）

17.3 刚性网格运动

节点：RigidGridMotion_t

Functions	Modes

`ier = cg_rigid_motion_write(int fn, int B, int Z,`
`char *RigidGridMotionName,`
`RigidGridMotionType_t RigidGridMotionType, int *R);`
	- w n

`ier = cg_n_rigid_motions(int fn, int B, int Z, int *n_rigid_motions);`	r - n

`ier = cg_rigid_motion_read(int fn, int B, int Z, int R,`
`char *RigidGridMotionName,`
`RigidGridMotionType_t RigidGridMotionType);`
	r - n

`call cg_rigid_motion_write_f(fn, B, Z, RigidGridMotionName,` `RigidGridMotionType, R, ier)`	- w n
`call cg_n_rigid_motions_f(fn, B, Z, n_rigid_motions, ier)`	r - n
`call cg_rigid_motion_read_f(fn, B, Z, R, RigidGridMotionName,` `RigidGridMotionType, ier)`	r - n

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）

Z	族索引数，其中 1≤Z≤nzones。（输入）
RigidGridMotionName	RigidGridMotion_t 节点的名称。（对 cg_rigid_motion_write 为输入；对 cg_rigid_motion_read 为输出）
RigidGridMotionType	刚性网格运动的类型。允许的类型有 Null, UserDefined, ConstantRate 和 VariableRate。（对 cg_rigid_motion_write 为输入；对 cg_rigid_motion_read 为输出）
n_rigid_motions	Z 块内 RigidGridMotion_t 节点的数值。（输出）
R	刚性旋转索引数，其中 1≤R≤n_rigid_motions。（对 cg_rigid_motion_read 为输入；对 cg_rigid_motion_write 为输出）
ier	错误状态。（输出）

17.4 任意网格运动

节点：ArbitraryGridMotion_t

Functions	Modes
`ier = cg_arbitrary_motion_write(int fn, int B, int Z, char *ArbitraryGridMotionName, ArbitraryGridMotionType_t ArbitraryGridMotionType, int *A);`	- w m
`ier = cg_n_arbitrary_motions(int fn, int B, int Z, int *n_arbitrary_motions);`	r - m
`ier = cg_arbitrary_motion_read(int fn, int B, int Z, int A, char *ArbitraryGridMotionName, ArbitraryGridMotionType_t ArbitraryGridMotionType);`	r - m
`call cg_arbitrary_motion_write_f(fn, B, Z, ArbitraryGridMotionName, ArbitraryGridMotionType, A, ier)`	- w m
`call cg_n_arbitrary_motions_f(fn, B, Z, n_arbitrary_motions, ier)`	r - m
`call cg_arbitrary_motion_read_f(fn, B, Z, A, ArbitraryGridMotionName, ArbitraryGridMotionType, ier)`	r - m

输入/输出

fn	CGNS 文件索引数。（输入）
B	库索引数，其中 1≤B≤nbases。（输入）
Z	族索引数，其中 1≤Z≤nzones。（输入）
ArbitraryGridMotionName	ArbitraryGridMotion_t 节点的名称。（对 cg_arbitrary_motion_write 为输入；对 cg_arbitrary _motion_read 为输出）
ArbitraryGridMotionType	任意网格运动的类型。允许的类型有 Null, UserDefined, NonDeformingGrid 和 DeformingRate。（对 cg_arbitrary_motion_write 为输入；对 cg_ arbitrary _motion_read 为输出）
n_arbitrary_motions	Z 块内 ArbitraryGridMotion_t 节点的数值。（输出）
A	任意网格运动索引数，其中 1≤A≤n_arbitrary_motions。（对 cg_arbitrary_motion_read 为输入；对 cg_arbitrary_motion_ write 为输出）
ier	错误状态。（输出）

18 链接

Functions	Modes
ier = cg_link_write(char *nodename, char *filename, char *name_in_file);	- w n
ier = cg_is_link(int *path_length);	r - n
ier = cg_link_read(char **filename, char **link_path);	r - n
call cg_link_write_f(nodename, filename, name_in_file, ier)	- w n
call cg_is_link_f(path_length, ier)	r - n
call cg_link_read_f(filename, link_path, ier)	r - n

输入/输出

nodename　　产生如 GridCoordinates 的链接节点的名称。（输入）

filename　　链接文件的名称，或者如果在相同的文件内链接，空字符串的名称。（对 cg_link_write 为输入；对 cg_link_read 为输出）

name_in_file　　链接指向的节点的路径名，它可以是简单名，也可以是复合名，例如 Bade/Zone 1/GridCoordinates。（输入）

path_length　　链接节点的路径名长度。如果节点不是链接的，则返回 0 值。（输出）

link_path　　链接指向的节点的路径名。（输出）

ier　　错误状态。（输出）

在调用这些程序之前，利用第 4 节介绍的 cg_goto(_f)将它们放置在文件中的适当位置。

如果采用 cg_link_write，在产生链接的时候，要链接的节点并非必须存在。但是，当采用链接时，如果要链接的节点不存在，就会产生错误。

只有支持下一级节点的节点才会支持这些链接。

对 C 语言函数 cg_link_read 的返回值，存储器由程序库分配。如果不再需要，此存储器应该由用户清空。