FyTok/SpDM 用户手册（alpha版）

FuYun 开发组

Nov 30, 2023

CONTENTS

Fy**Tok** 是基于托卡马克 Ontology (IMAS DD) 构建的集成建模框架。Fy**Tok** 可以系统性的提供托卡马克数据分析和物理建模功能，可以构建复杂的工作流，并可通过灵活插件机制与第三方物理程序进行交互。对于集成建模和数据分析用户，仅需阅读本文档的第一部分“FyTok 托卡马克模拟器”，即可快速上手使用 Fy**Tok**。

Sp**DM** 是一个 Python 库，主要包含数据集成、建模和可视化工具。Sp**DM** 为 Fy**Tok** 提供了物理语义无关的核心功能和基础架构。对于希望添加数据源或物理程序的用户，建议阅读本文档的第二部分“SpDM (SpDB) 数据集成建模”，了解数据和程序接口，以及辅助工具。

本文档的组织结构

名词表

CRAFT Comprehensive Research Facility for Fusion Technology，聚变堆主机关键系统综合研究设施项目

DD Data Dictionary (数据字典)，本文特指IMAS的数据字典，作为托卡马克数据标准规范。

HTree Hierarchical tree，分层树状结构，是一种数据结构，用于存储层次化的数据。

IDS Interface Data structure, IMAS DD 数据结构中最大的组成单位，对应一个相对独立的物理概念或者装置组件。

IM Integrated Modeling

IMAS Integrated Modelling & Analysis Suite，ITER组织开发的集成建模与分析软件套件

ITER International Thermonuclear Experimental Reactor

Ontology 本体，是（特定领域）信息组织的一种形式，是领域知识规范的抽象和描述，是表达、共享、重用知识的方法

URI Uniform Resource Identifier

版本信息

- 版本：alpha
- 日期：2023-11-28

开发人员

- 于治, Zhi Yu, yuzhi@ipp.ac.cn (ASIPP)
- 刘晓娟, Xiaojuan Liu, lxj@ipp.ac.cn (ASIPP)
- ……

致谢

本工作得到了《聚变堆主机关键系统综合研究设施（CRAFT）》项目，《总控课题：集成数值建模和数据分析系统框架开发》的支持（项目编号：2018-000052-73-01-001228.）

Part I

FyTok 托卡马克模拟器

CHAPTER

ONE

概述

托卡马克是一个由多个复杂子系统耦合而成的大科学系统装置。每个子系统中都蕴含了具有不同时空尺度、不同物理机制的复杂过程。这些物理过程各自成体系，又相互紧密联系。很难用单一数值模型描述整个复杂体系的多时空尺度过程。所以，需要分别对各个子系统上的不同物理过程进行建模，形成针对单独子系统的数值模型，并使子系统间的相互作用以数据的方式在模型中传递。最终，将子系统数值模型整合在一起形成全装置的“模拟器”，即所谓的“集成建模”。

几十年来，聚变领域内针对不同物理过程开发了许多优秀、完善数值模型和模拟分析程序。将这些程序整合在一个的架构下，完成耦合，实现整体装置的模拟。传统程序的开发时间跨越了几个世代，地域上也遍布全球各地，开发语言各异，接口不统一。这些程序的集成耦合是一个极具挑战性的任务。多年来随着托卡马克工程和物理研究的深入，集成建模和分析程序的开发，已经从原本单一、孤立的专有数值计算程序，向开放、灵活的数据处理平台发展，更加强调数据的共享、交流和管理。

为此 ITER 组织开发了集成建模与分析软件套件（Integrated Modelling & Analysis Suite，IMAS）。IMAS 是一个开放的、灵活的数据处理平台，它的核心是数据字典（Data Dictionary，DD）。IMAS 的数据字典是一个包含了托卡马克物理实验的所有物理量的数据模型，它定义了托卡马克物理实验的数据结构、数据格式、数据语义、数据关系等。IMAS 的数据字典是一个标准的、统一的、开放的数据模型，它为托卡马克物理实验的数据处理提供了统一的数据语义，使得不同物理模型之间的数据交互成为可能。IMAS 的数据字典是一个动态的、开放的数据模型，它可以随着托卡马克物理实验的发展而不断完善，可以随着托卡马克物理实验的发展而不断扩展。IMAS 的数据字典是一个开放的、灵活的数据模型，它可以被托卡马克物理实验的数据处理程序所使用，也可以被托卡马克物理实验的数据分析程序所使用。

IMAS 数据字典不仅仅定义了描述托卡马克装置的物理量数据模型，也在其数据的文本描述中涉及了物理量的定义和物理量之间的约束。这使其可以在一定程度上视为托卡马克本体（Ontology）表述。但IMAS 数据字典没有完整给出的物理量之间的计算关系，物理量的计算方法。这些计算关系和计算方法是托卡马克物理实验的物理模型，是托卡马克物理实验的物理模拟器。在 IMAS 数据字典中，物理量之间的约束关系是以文本的形式给出的，这种文本形式的物理模型是一种静态的、不可执行的物理模型。我们需要将这种静态的、不可执行的物理模型转换为动态的、可执行的物理模型，即将物理模型的计算关系和计算方法转换为计算机程序。这种转换过程是一个非常复杂的过程，需要对物理模型的计算关系和计算方法进行解析、分析、转换、优化、编译、执行等一系列操作。这种转换过程是托卡马克数据分析和物理建模中关键的一步，也是最为复杂的一步。“集成建模”中的“建模”就是指这种物理模型的转换过程。或者说，“集成建模”是将静态的 Ontology 描述转换为动态的计算机程序，并集成在一起，以实现对于装置整体行为的分析和预言，可称之为基于 Ontology 的建模。

Tip: Ontology是形而上学的一个基本分支，又译为物性论、存在论或系论。它的英文单词ontology源于希腊语词根ont-（存在）和logia（学问）的组合。Ontology的目的是定义一个专业领域内的词汇，并描述这些词汇之间的关系。它类似于字典或术语表，但不同的是，它能让计算机处理更多内容的细节和结构。

在人工智能和信息领域，Ontology 是（特定领域）信息组织的一种形式，是领域知识规范的抽象和描述，是表达、共享、重用知识的方法。Ontology 是知识图谱构建的关键技术，它的关键在于让计算机能够处理人类的知识。

Fy**Tok** 的设计目标是从托卡马克 Ontology 出发，将物理模型抽象为 Ontology 中的概念，将物理模型的输

入输出抽象为 Ontology 中的属性，将物理模型的参数抽象为 Ontology 中的实例，将物理模型的计算过程抽象为 Ontology 中的关系，将物理模型的计算结果抽象为 Ontology 中的实例。通过将静态数据与具体功能函数/程序绑定，使得用户可以通过简单的配置，实现复杂的数据处理和物理建模。

Fy**Tok** 以 IMAS 数据字典作为核心数据模型，遵循其分层树状（hierarchical tree）结构，通过将具体的数值计算过程绑定到静态的树状结构上，构成动态的计算模型，实现从静态的 Ontology 描述到动态的计算模型的转换。并在此基础上，实现对物理模型的计算关系和计算方法的管理和维护，实现对物理模型的计算关系和计算方法的复用和共享，实现对物理模型的计算关系和计算方法的集成和耦合，实现对物理模型的计算关系和计算方法的分析和优化，实现对物理模型的计算关系和计算方法的执行和调用。

Fy**Tok** 主体是一个Python库。借助 Python 语言生态的优势，Fy**Tok** 提供了灵活的插件机制，可以方便地与其他语言开发的科学计算程序进行集成。

Fy**Tok** 中与具体物理语义无关的基础功能和架构，由项目 Sp**DM** 提供，包括常用数据文件和数据源的读写，数据语义转换，数据集成，数据可视化，外部程序集成，执行流管理等等。

CHAPTER

TWO

FYTOK安装

2.1 ShenMa集群

目前，ShenMa集群内（service108）服务器上FyTok模块是可用的。您能运行下面命令，加载其环境：

```
module load fytok/0.0.0-foss-2022b
```

注：执行该操作后，fytok会自动将其依赖的其他环境一起加载，如Python、数据集成工具SpDM等

如果，您需要使用其他第三方物理模块，可以继续执行下面操作：

```
module load fytok_ext/0.0.0-foss-2022b
```

测试安装：

```
python -c "import fytok; print(fytok.__version__)"
```

2.2 本地安装

在自己维护的Python环境下安装FyTok 要求：

- Python版本3.10+。
- FyTok 依赖 SpDM

```
# 先安装SpDM
pip install --upgrade pip
pip install spdm
## 再安装fytok
## 下载FyTok-0.0.0-py3-none-any.whl
pip install FyTok-0.0.0-py3-none-any.whl
```

如果pip安装指定了安装目录，需先添加安装目录到PYTHONPATH中。否者在默认的~/.local/lib/下面：

```
export PYTHONPATH=${INSTALLPATH}/site-packages:${PYTHONPATH}
```

2.2.1 测试您的安装：

```
cd ~
python -c "import spdm; print(spdm.__version__)"
python -c "import fytok; print(fytok.__version__)"
```

调用 FyTok 包时显示的信息包括：

- 项目 logo，
- 版权信息，
- 版本号（fytok、spdm、extension，）
- 用户信息、
- 时间戳
- 运行标记（tag），包含用户名、系统标签和时间标签

```
import fytok
```

```
2023-11-28 15:59:20,989 [   fytok]     INFO:
##################################################################################################################


    ______      _____     _
   / ____/_  __|_   _|__ | | __
  / /_  / / / /  | |/ _ \| |/ /
 / __/ / /_/ /   | | (_) |   <
/_/    \__, /    |_|\___/|_|\_\
      /____/

 Copyright (c) 2021-present Zhi YU (Institute of Plasma Physics Chinese Academy of Sciences)

 url: https://gitee.com/openfusion/fytok_tutorial
      https://github.com/fusion-yun/fytok_tutorial

 version = alpha  (spdm=beta extension=0.2.2.dev11+g706f667, imas_version=v3_38_1_dirty)

 Run by salmon at 2023-11-28T15:59:20.989168.
 Job ID: fytok_salmon_surface_15744

##################################################################################################################
```

CHAPTER

THREE

模块

3.1 状态树和 Actor

在构建托卡马克 Ontology 数据结构时，相互联系紧密的量形成一棵棵相对独立的树。通常这样的一棵树，会对应一个相对独立的物理概念或者装置组件，称为 **Actor**，而这些量描述了 Actor 状态，称之状态树，在单一时间点上的状态树时间片（TimeSlice）。Actor 的状态会随时间或者迭代过程发生改变。从数据流角度来看，Actor 表述了一个 TimeSlice 演化构成的序列。从执行流角度来看，Actor 可以根据已有的 TimeSlice 和外部输入计算出新的 TimeSlice，更准确地说是 Actor 可以根据外部输入改变状态树。兼顾数据流和执行流，Actor 需要具有时间序列的管理功能，也要能够绑定物理程序实现状态树的演化和更新。存在一种可能，状态树构成的演化序列由外部数据源一次性给出，例如对应于诊断和控制的 Actor ，其状态序列是来源于实验诊断和控制指令的信号量（Signal）。这时 Actor 的任务是从已有时间序列中查找到所需的时间点，分别截取数据，再整合成 TimeSlice。Actor 可以以 TimeSlice 的形式给出其在某一时间点上的状态。

Note: 时间片 TimeSlice 和信号 Signal: Actor的状态随时间演化，在单一时刻状态的集合可称之为时间片 TimeSlice，由多个时间片按照时间顺序排列构成时间序列 TimeSeries 。一个Actor随时间演化，表现为由其状态树组成序列，这种数据结构称作 AoS，(Array of structure)。AoS 数据结构更适宜描述抽象物理概念的演化。当描述控制或诊断时，更多的是采用信号量（Signal）的形式。信号量是由等长的时间序列和物理量的序列组成，物理量可以是标量也可以是矢量或者矩阵。这时 Actor 的状态表述为一组信号组成的树，对应的数据结构称为 SoA（Structure of Array）。AoS 和 SoA 两种结构分别适用于不同的数据抽象，可以相互转换。

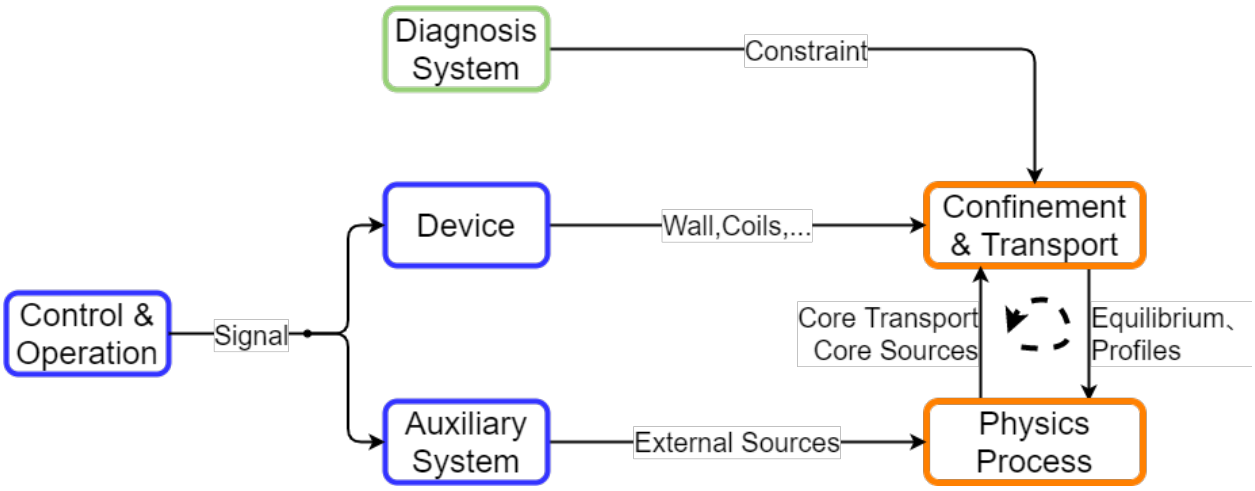
托卡马克 Ontology 规定了 Actor 的物理含义，同时也就明确 Actor 之间的依赖关系。这里的依赖关系可以理解为，当 Actor 更新其状态时需要其他 Actor 的状态作为输入。若将 Actor 视为一个函数，依赖关系定义了其输入参数，输出则写入而描述 Actor 当前状态的 TimeSlice 中。可以将 Ontology 对 Actor 的语义和依赖关系的表述，视作对物理程序 API 的约定。在实际的集成过程中，Fy**Tok** 根据具体程序物理语义将其绑定到对应的 Actor 上，由 Actor 将符合 Ontology 描述的数据转发给外部程序，并收集其输出，转换为同样符合 Ontology 描述的数据结构。

3.2 托卡马克 Ontology 的模块划分

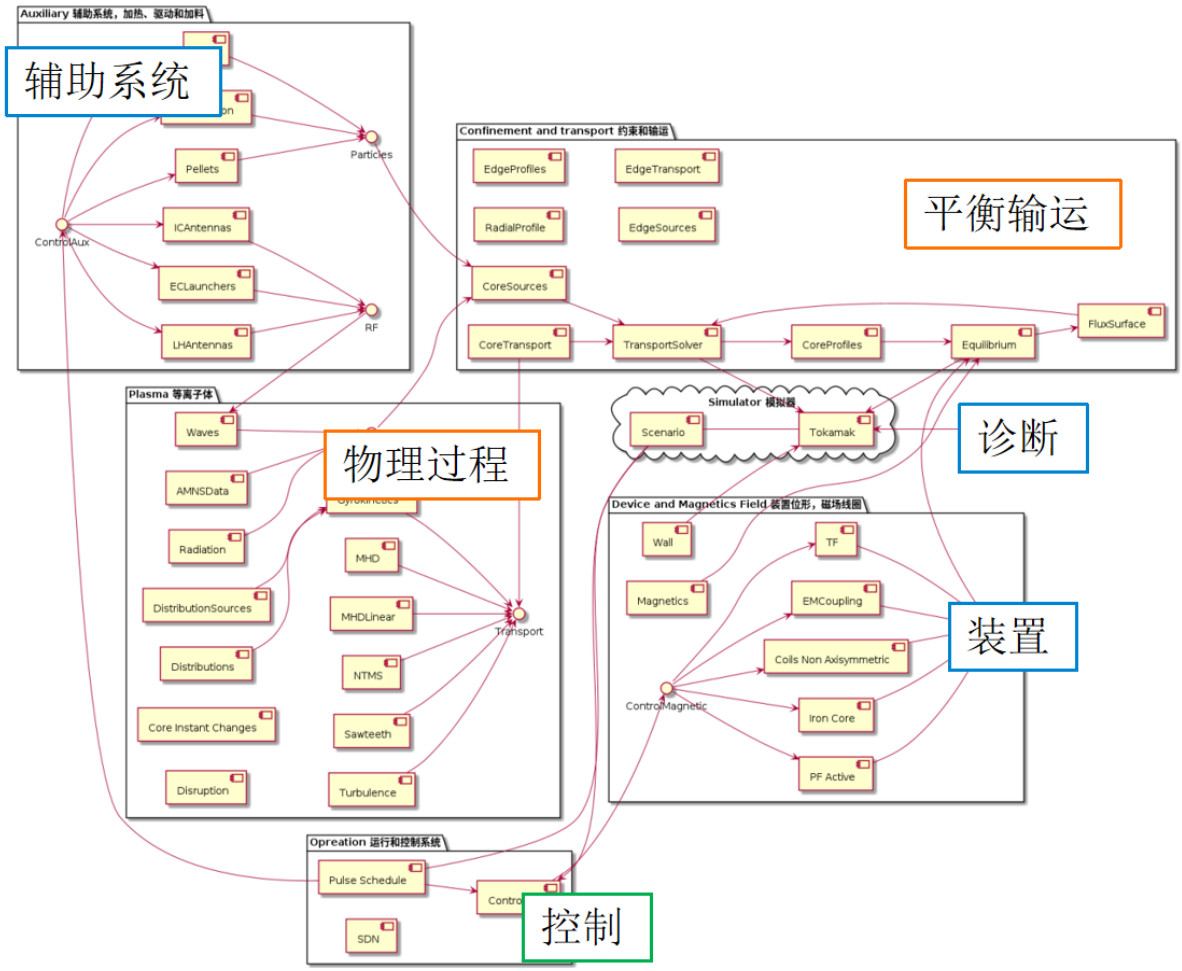
IMAS DD 中最上层的树状结构被称为 IDS（Interface Data structure ）。IMAS 包含近百各 IDS，根据其语义可以分解为以下几各大类（catalogy）：

- 用以描述物理概念和过程：
  - 平衡输运：equilibrium, core_profiles, core_transport, core_sources…
  - 物理过程：wave, MHD, turbulence…
- 用以描述装置中独立的子系统
  - 装置: wall, pf_active…
  - 诊断：magnetics, ECE, Langmuir Probes , Polarimeter…
  - 辅助系统: LH Antennas, ECLauhchers,IC Antennas, NBI, Pellets…
- 用以描述装置运行过程参数：
  - 控制系统: Pulse Schedule , Control

托卡马克的状态演化，可以理解为概略的描述为这几大类 IDS 之间的相互作用结果



更详细的结构如下



3.3 class Module(Actor)

IMAS IDS 大多可以直接作为 Actor。其中，少部分是包含多个 Actor 的集合。如 `core_transport.model`，每个 `model` 对应一个 Actor。在 FyTok 实现中，按照 IDS 构建的 Module 在路径 `fytok.modules` 下，其中包含两个基类：

- `fytok.modules.Utilities.Module`，继承自 `spdm.data.Actor`，增加 `Module.code` 属性（修改自 IDS.code），是所有模块的基类。
- `fytok.modules.Utilities.IDS`，继承自 `fytok.modules.Utilities.Module`，增加 `ids_properties` 属性，符合 IDS 结构。

详见下表：

class Module(*args, **kwargs)
Bases: spdm.data.Actor.Actor

code: fytok.modules.Utilities.Code
对于 Module 的一般性说明。@note code 在 __init__ 时由初始化参数定义，同时会根据 code.name 查找相应的 plugin 。

refresh(*args, time=None, **kwargs) → None
更新当前 Actor 的状态。若 time 为 None 或者与当前时间一致，则更新当前状态树，并执行 self.iteration+=1 否则，向 time_slice 队列中压入新的时间片。

time_slice: spdm.data.TimeSeries.TimeSeriesAoS[spdm.data.TimeSeries.TimeSlice]
时间片序列，保存 Actor 历史状态。@note: TimeSeriesAoS 长度为 n(=3) 循环队列。当压入序列的 TimeSlice 数量超出 n 时，会调用 TimeSeriesAoS.__full__(first_slice)

property current: Type[spdm.data.TimeSeries.TimeSlice]
当前时间片，指向 Actor 所在时间点的状态。

property previous: Type[spdm.data.TimeSeries.TimeSlice]
前一个时间片，指向 Actor 在前一个时间点的状态。

property time: float
当前时间，

property iteration: int
当前时间片执行 refresh 的次数。对于新创建的 TimeSlice，iteration=0

classmethod register(sub_list: Optional[Union[str, list]] = None, plugin_cls=None)
Decorator to register a class to the registry.

class IDS(*args, **kwargs)
Bases: fytok.modules.Utilities.Module

Base class of IDS

ids_properties: fytok.modules.Utilities.IDSProperties
Interface Data Structure properties. This element identifies the node above as an IDS

CHAPTER

FOUR

FYTOK.TOKAMAK

4.1 介绍

fytok.Tokmak 用于执行集成工作流。不在 IMAS DD 的 IDS 定义之内，它包含多个 Module/IDS/Actor，通过驱动 Module 根据物理逻辑更新演化，实现集成建模功能。

4.2 class Tokamak(Actor)

class Tokamak(*args, device: str = tags.not_found, shot: int = tags.not_found, run: int = tags.not_found, time: Optional[float] = None, **kwargs)

Bases: spdm.data.Actor.Actor

__init__(*args, device: str = tags.not_found, shot: int = tags.not_found, run: int = tags.not_found, time: Optional[float] = None, **kwargs)

用于集成子模块，以实现工作流。

现有子模块包括: wall, tf, pf_active, magnetics, equilibrium, core_profiles, core_transport, core_sources, transport_solver

Parameters

- **args** –初始化数据，可以为 dict，str 或者 Entry。输入会通过数据集成合并为单一的HTree，其子节点会作为子模块的初始化数据。
- **device** –指定装置名称，例如，east，ITER, d3d 等
- **shot** –指定实验炮号
- **run** –指定模拟计算的序号
- **time** –指定当前时间
- **kwargs** –指定子模块的初始化数据，，会与args中指定的数据源子节点合并。

property brief_summary: str

综述模拟内容

property title: str

标题，由初始化信息 dataset_fair.description

property tag: str

当前状态标签，由程序版本、用户名、时间戳等信息确定

refresh(*args, **kwargs) → None
更新当前 Actor 的状态。若 time 为 None 或者与当前时间一致，则更新当前状态树，并执行 self.iteration+=1 否则，向 time_slice 队列中压入新的时间片。

4.3 示例

```
from fytok.Tokamak import Tokamak
```

4.4 创建 Tokamak 实例

初始化信息包括

参数	解释
“file+geqdsk://...”	数据源。指向一个 geqdsk 文件，提供平衡信息
device=” ITER”	指定装置名称。根据装置名称，可以自动调取装置的静态信息，例如，wall.limiter 的几何形状，pf_active.coils 极向场线圈的位置等等。默认支持的装置信息，括 EAST、ITER 和 D3D，可通过配置文件扩展其他装置。
shot=900003，	指定放电的炮号。若数据源包含实验数据库，会自动读取相应的炮号作为数据。
run=0	指定模拟运行的编号。若数据源中包含可写入的是模拟数据库，写入模拟结果是会作为数据 index 的一部分。
equilibrium={” code.name: “fy_eq” }	指定子模块 equilibrium 的初始化信息。其中 code.name=” fy_eq”，意为调用路径 fytok.modules.equilibrium.fy_eq 下的插件

tok.refresh(time=5.0) 指定时间。若数据源中包含实验或者模拟数据库，则会查找最接近的时间片导入。

```
tok = Tokamak(
    "mdsplus:///home/salmon/workspace/fytok_data/mdsplus/~t/?enable=efit_east",
    device="east",
    shot=70754,
    run=0,
    equilibrium={"code.name": "fy_eq"},
)

tok.refresh(time=5.0)
```

```
2023-11-29 13:51:12,438 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_xml
2023-11-29 13:51:12,760 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module fytok.plugins.equilibrium.fy_eq
2023-11-29 13:51:12,864 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_mdsplus
2023-11-29 13:51:12,866 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/plugins/
↪data/plugin_mdsplus.py:118:get_tree: Open MDSplus Tree [efit_east] shot=70754
2023-11-29 13:51:12,909 [   fytok]    DEBUG: /home/salmon/workspace/fytok/python/fytok/modules/
↪Utilities.py:121:execute: Execute fytok.plugins.equilibrium.fy_eq-0.0.1 [Zhi YU@ASIPP]
```

4.5 展示概要信息

展示模拟的概要信息，包括

- 装置名称、炮号、运行序号，
- 使用者、运行环境、运行时间、采用的 Ontology 版本
- 调用的模块信心

```
print(tok.brief_summary)
```

```
2023-11-29 13:51:12,944 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module fytok.plugins.transport_solver_numerics.fy_trans
Tokamak simulation :
--------------------------------------------------------------------------------------------------------
                                                Brief Summary
--------------------------------------------------------------------------------------------------------
Dataset Description:

    Device: EAST, Shot: 70754, Run: 0,
    Run by Salmon on Surface at 2023-11-29T13:51:12.940918, base on ontology "modified imas/3"

--------------------------------------------------------------------------------------------------------
Modules:
    transport_solver        : fy_trans-0.0.1 [fytok]
    equilibrium             : fy_eq-0.0.1 [Zhi YU@ASIPP]

    core_profiles           : N/A
    core_transport          :
    core_sources            :
--------------------------------------------------------------------------------------------------------
```

4.6 可视化

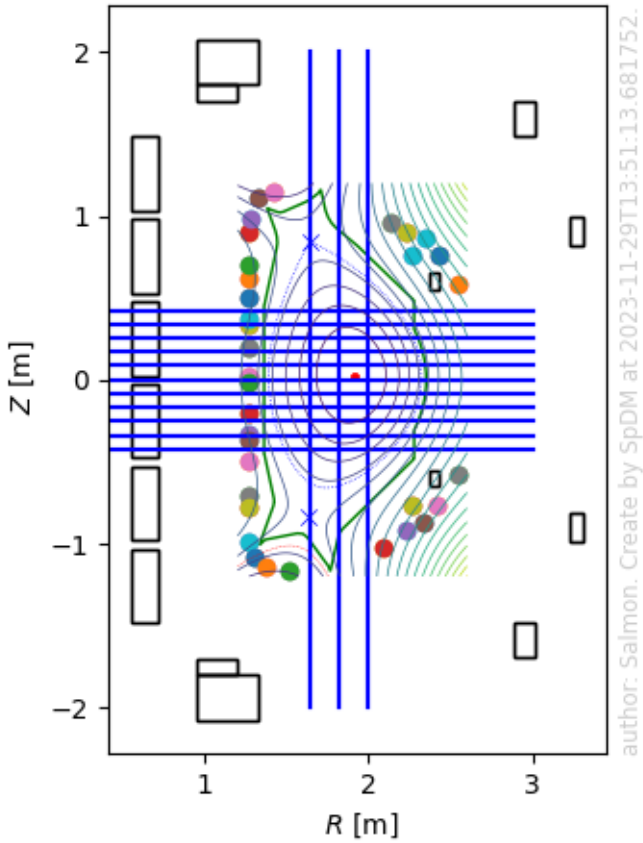
4.6.1 适用于 JupyterLab 环境的可视化接口

FyTok 提供了适用于 JupyterLab 环境的可视化接口，`spdm.view.View.display()` 调用。默认情况下，Tokamak 会将能够获取到的信息尽可能绘制到一张图上。如下图，展示了 pf_active 极向场线圈、magnetic 磁场探针，wall.limiter 壁, interferometer 干涉仪的几何信息，磁平衡信息给出了二维磁面的等高线，最外层磁面、磁轴、X-point等信息。

Note: 具有 IDS 可视化语义的 IDS 都应支持 `spdm.view.View.display()` 可视化，如 wall,pf_active,equilibrium。关于可视化库 spdm.view 的进一步细节请参考第二部分 SpDM 的可视化章节。

```
from spdm.view import View as sp_view
fig=sp_view.display(tok)
```

```
2023-11-29 13:51:12,976 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.view.view_matplotlib
2023-11-29 13:51:13,342 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.mesh.mesh_rectangular
```



4.7 将可视化结果输出为图像文件

通过指定输出文件路径 output=f"output/{tok.tag}_rz.svg"，可以将可视化结果以文件的形式保存。

```
sp_view.display(tok, title=tok.title, output=f"output/{tok.tag}_rz.svg")
```

```
2023-11-29 13:51:14,896 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/view/
↪view_matplotlib.py:71:_figure_post: Write figure to  output/east_70754_0_000500_rz.svg
```

输出的文件如下

```
from IPython.core.display import SVG
SVG(f"output/{tok.tag}_rz.svg")
```

```
<IPython.core.display.SVG object>
```

CHAPTER

FIVE

装置中的子系统

用以记录和描述装置独立的子系统状态的 IDS 包括：

- 装置: wall, pf_active…
- 诊断：magnetics, ECE, Langmuir Probes , Polarimeter…
- 辅助系统: LH Antennas, ECLauhchers,IC Antennas, NBI, Pellets…

数据来源包括：

- 装置静态几何信息；
- 实际控制信号或者设计控制方案；
- 实验诊断数据；

对于这类 IDS，通过数据集成工具将不同来源的数据汇总、映射为符合 IMAS Ontology 的形式。对于 EAST 目前完成数据映射的子系统有：wall, pf_active, magnetics.

Note: 在后续示例中，分别以 `wall, pf_active, magnetics` 子系统作为示例，展示如何使用读取、处理、分析、可视化装置子系统数据。其中数据源包括：本地文件、数据库、远程服务器等。关于数据源的配置请参考的相关文档。

5.1 wall

`wall` 提供对托卡马克装置第一壁的描述（继承自 IMAS Ontology 的定义）

class Wall(*args, **kwargs)

Bases: `fytok.ontology.imas_lastest.wall._T_wall`

Description of the torus wall and its interaction with the plasma

description_2d: spdm.data.AoS.AoS[fytok.ontology.imas_lastest.wall._T_wall_2d]

Set of 2D wall descriptions, for each type of possible physics or engineering configurations necessary (gas tight vs wall with ports and holes, coarse vs fine representation, single contour limiter, disjoint gapped plasma facing components, …). A simplified description of the toroidal extension of the 2D contours is also provided by using the phi_extensions nodes.

5.1.1 创建 Wall 实例

```
from fytok.modules.Wall import Wall

wall = Wall("east://#wall")
```

```
2023-11-29 13:11:53,738 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_xml
```

由于 Wall 的描述来自装置信息，采用 URI east://#wall 指定数据源。其中，east:// 指定了装置 identificer，会根据其匹配合适的装置描述文件。#wall 意为从树中裁剪 wall 所对应的一支。由于不涉及数据库或其他动态数据源，省略了 URI 其他部分。

其中，关于 limiter 几何信息（wall.description_2d[0].limiter.unit[0].outline ）以 r，z 坐标数组的形式描述

```
wall.description_2d[0].limiter.unit[0].outline.r
```

```
array([2.277, 2.273, 2.267, 1.94 , 1.94 , 1.802, 1.773, 1.751, 1.736,
       1.714, 1.707, 1.696, 1.665, 1.656, 1.635, 1.612, 1.478, 1.459,
       1.44 , 1.436, 1.399, 1.379, 1.392, 1.43 , 1.439, 1.442, 1.437,
       1.363, 1.361, 1.361, 1.361, 1.363, 1.421, 1.423, 1.422, 1.418,
       1.331, 1.367, 1.564, 1.597, 1.598, 1.624, 1.754, 1.765, 1.814,
       1.824, 1.825, 1.841, 1.971, 1.971, 2.267, 2.273, 2.277, 2.277,
       2.306, 2.328, 2.343, 2.35 , 2.35 , 2.35 , 2.343, 2.328, 2.306,
       2.277])
```

```
wall.description_2d[0].limiter.unit[0].outline.z
```

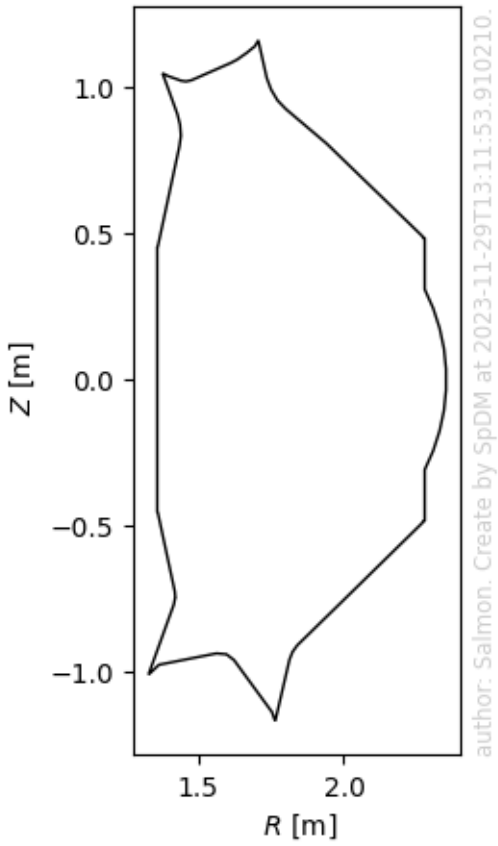
```
array([ 0.485,  0.485,  0.493,  0.809,  0.809,  0.926,  0.956,  0.993,
        1.033,  1.131,  1.162,  1.142,  1.117,  1.111,  1.096,  1.084,
        1.025,  1.021,  1.024,  1.026,  1.039,  1.049,  1.014,  0.909,
        0.873,  0.835,  0.799,  0.456,  0.454,  0.   , -0.454, -0.456,
       -0.725, -0.748, -0.749, -0.77 , -1.011, -0.977, -0.938, -0.941,
       -0.941, -0.961, -1.139, -1.17 , -0.959, -0.934, -0.932, -0.91 ,
       -0.783, -0.783, -0.493, -0.485, -0.485, -0.309, -0.244, -0.176,
       -0.106, -0.036,  0.   ,  0.036,  0.106,  0.176,  0.244,  0.309])
```

5.1.2 可视化效果

```
import spdm.view.View as sp_view

fig = sp_view.display(wall)
```

```
2023-11-29 13:11:53,788 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.view.view_matplotlib
```



5.2 pf_active

pf_active 提供对托卡马克装置极向场线圈的描述（继承自 IMAS Ontology 的定义）

class PFActive(*args, **kwargs)
Bases: fytok.ontology.imas_lastest.pf_active._T_pf_active

coil: AoS[_T_pf_coils]
Active PF coils

5.2.1 创建 PFActive 实例

以本地 MDSplus 数据创建 PFActive 实例：

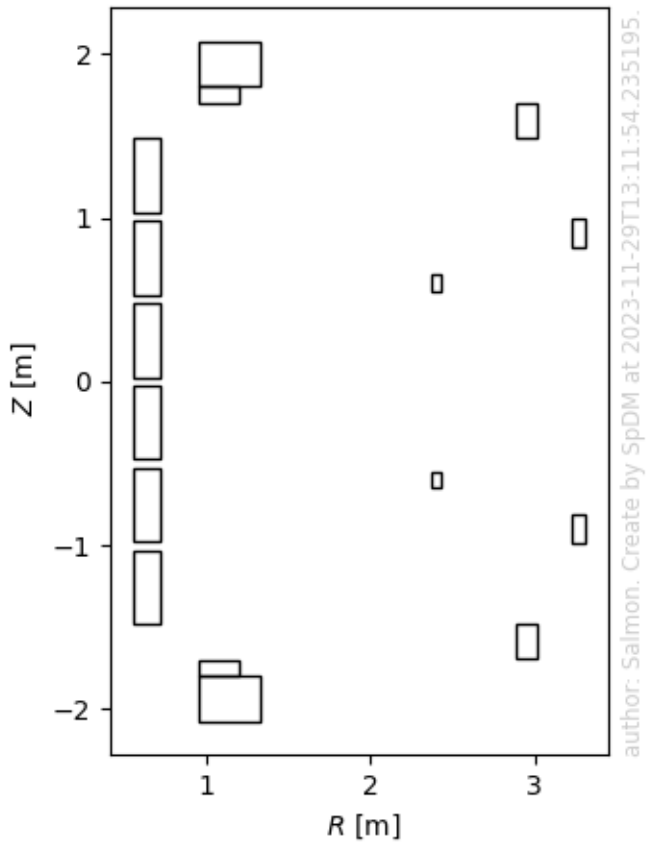
```
from fytok.modules.PFActive import PFActive

pf_active = PFActive(f"east+mdsplus:///home/salmon/workspace/fytok_data/mdsplus/~t/?shot=70754#pf_
↪active")
```

5.2.2 可视化

· 2D： pf 线圈截面

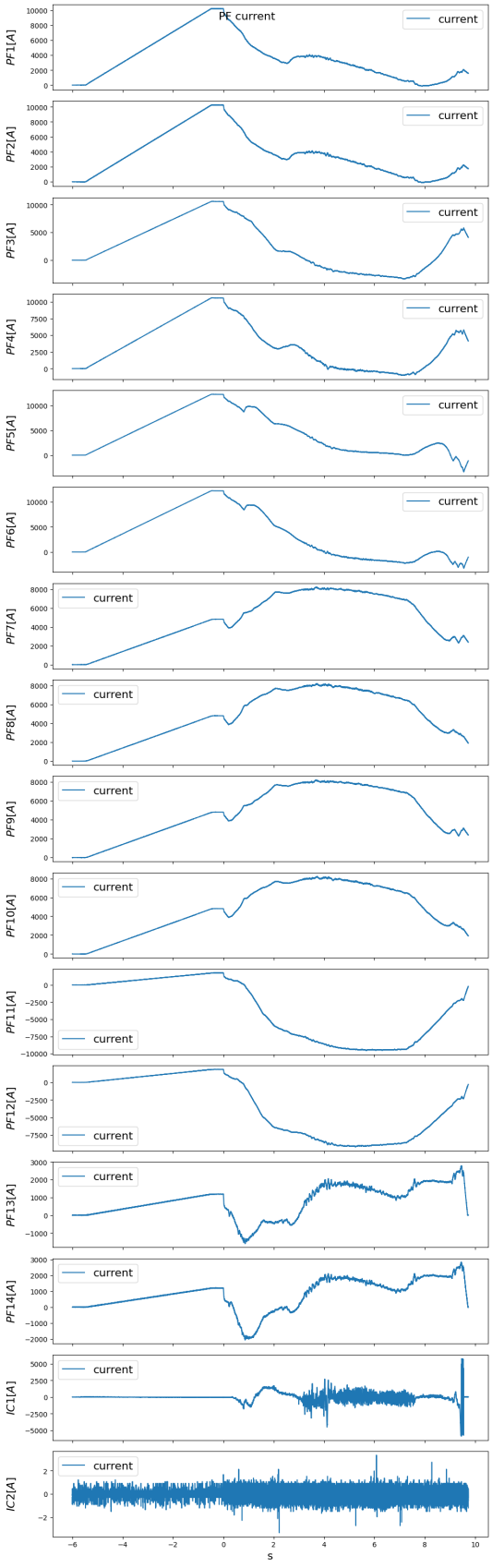
```
import spdm.view.View as sp_view
fig=sp_view.display(pf_active)
```



· 1D： pf 线圈电流随时间变化

```
fig = sp_view.plot(
    [(coil.current, {"y_label": f"{coil.name}[A]"}) for coil in pf_active.coil], x_label="s", title=
↪"PF current"
)
```

```
2023-11-29 13:11:54,748 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_mdsplus
2023-11-29 13:11:54,751 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/plugins/
↪data/plugin_mdsplus.py:118:get_tree: Open MDSplus Tree [pcs_east] shot=70754
```


PF current
current
PF1[A]
PF2[A]
PF3[A]
PF4[A]
PF5[A]
PF6[A]
PF7[A]
PF8[A]
PF9[A]
PF10[A]
PF11[A]
PF12[A]
PF13[A]
PF14[A]
IC1[A]
IC2[A]
S

5.3 magnetics

magnetics 提供对托卡马克磁探针的描述（继承自 IMAS Ontology 的定义）

class Magnetics(*args, **kwargs)

Bases: fytok.ontology.imas_lastest.magnetics._T_magnetics

Magnetic diagnostics for equilibrium identification and plasma shape control.

5.3.1 创建 Magnetics 实例

以远程 MDSplus 数据库创建 Magnetics 实例：

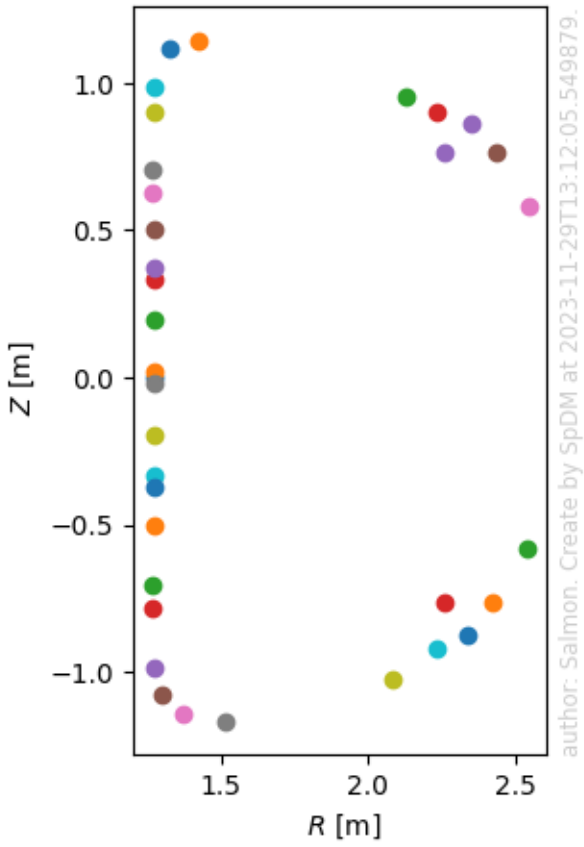
```
from fytok.modules.Magnetics import Magnetics

magnetics = Magnetics(f"east+mdsplus://202.127.204.12?shot=70754#magnetics")
```

5.3.2 可视化

- 2D： 磁探针空间分布

```
import spdm.view.View as sp_view
fig=sp_view.display(magnetics)
```



- 2D： 磁探针信号随时间变化

```
fig = sp_view.plot(
    [(probe.flux, {"label": probe.name, "y_label": f"[Wb]"}) for probe in magnetics.flux_loop],
    x_label="s",
    title=" flux_loop",
)
```

CHAPTER

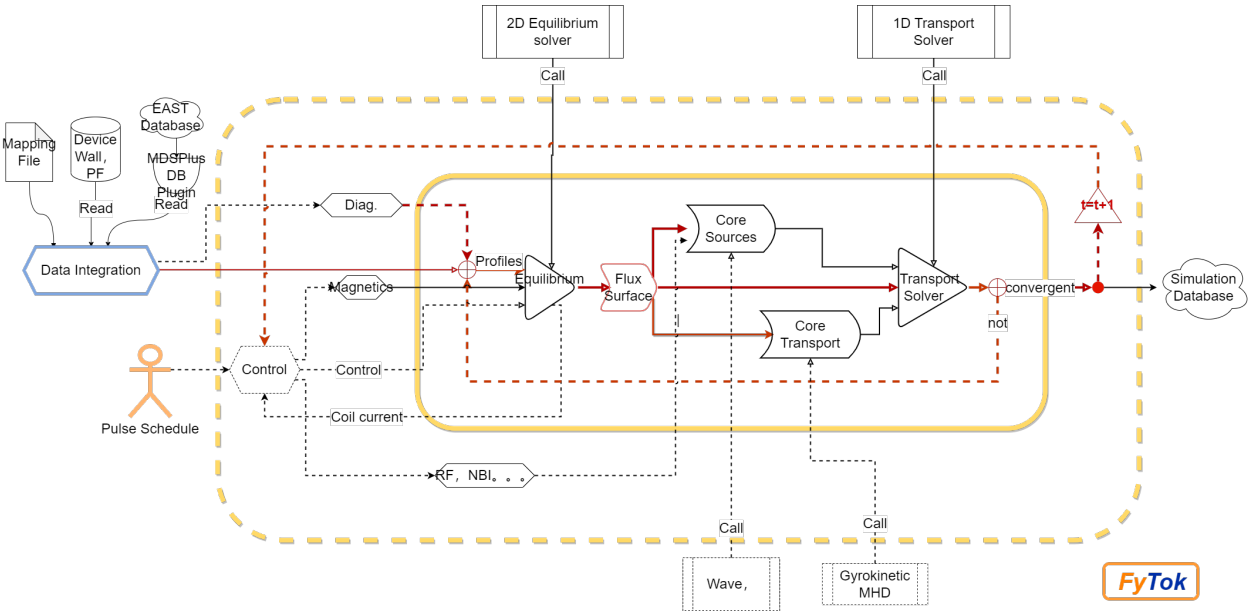
SIX

物理概念和过程

用以描述物理概念和过程的 IDS，数据源自物理建模和数值计算程序的结果，例如：

- 平衡输运：equilibrium, core_profiles, core_transport, core_sources…
- 物理过程：wave, MHD, turbulence…

对于这类 IDS ，通过Module Plugin机制绑定相对独立的功能模块，然后根据 IMAS Ontology 描述的依赖关系，构建工作流实现集成建模。平衡和芯部输运问题式传统集成建模的主要问题，下图为 1.5 维芯部输运的工作流：



6.1 equilibirum

`equilibirum` 提供二维轴对称托卡马克磁平衡的描述。

6.1.1 创建 equilibrium 实例

由 geqdsk 文件创建 equilibrium 实例，默认调用插件 fy_eq，提供基础的磁面分析计算。

```
from fytok.modules.Equilibrium import Equilibrium

equilibrium = Equilibrium({"code": {"name": "dummy"}}, "file+geqdsk://./data/g070754.05000#equilibrium
↪")
```

```
2023-11-29 21:23:44,558 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module fytok.plugins.equilibrium.dummy
2023-11-29 21:23:44,562 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_geqdsk
```

6.1.2 可视化

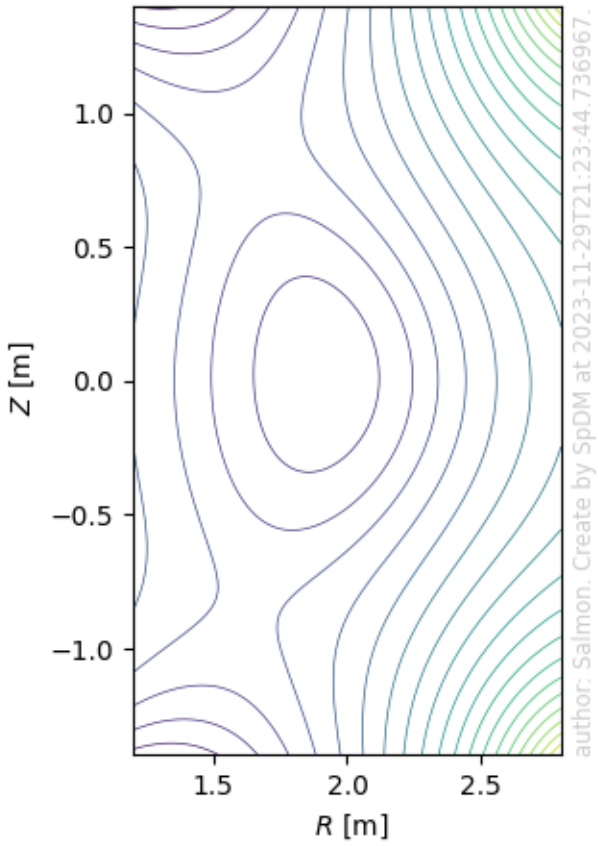
2D 磁面:

给出当前时间片的二维磁场位形（equilibrium.time_slice.current.profiles_2d.psi）

```
import spdm.view.View as sp_view

fig=sp_view.display(equilibrium.time_slice.current.profiles_2d.psi)
```

```
2023-11-29 21:23:44,611 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.view.view_matplotlib
2023-11-29 21:23:44,631 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.mesh.mesh_rectilinear
```

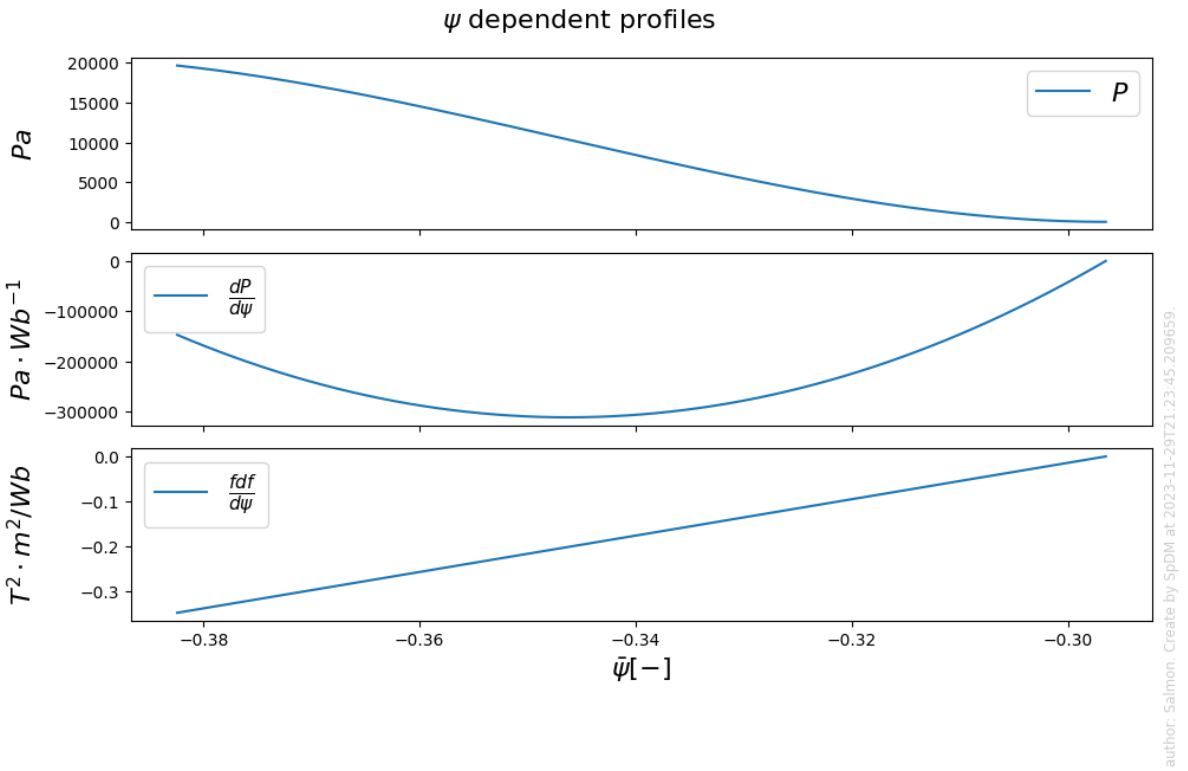


1D 位形分布

绘出当前时间片的一维 profile，磁面坐标为为 ψ equilibrium.time_slice.current.profiles_1d.psi

```
eq_profiles_1d=equilibrium.time_slice.current.profiles_1d
```

```
profs = sp_view.plot(
    [
        eq_profiles_1d.pressure,
        eq_profiles_1d.dpressure_dpsi,
        eq_profiles_1d.f_df_dpsi,
    ],
    x_axis=eq_profiles_1d.psi,
    x_label=r"$\bar{\psi}[-]$",
    title=r"$\psi$ dependent profiles",
)
```



6.1.3 class Equilibrium

除了 ψ, ff', p 等少数基础信息，其他物理量可通过 fy_eq 实时演算获得。

class EquilibriumGlobalQuantities(*args, **kwargs)
Bases: fytok.ontology.imas_lastest.equilibrium._T_equilibrium_global_quantities

class CurrentCentre(*args, **kwargs)
Bases: fytok.modules.Equilibrium.EquilibriumGlobalQuantities.CurrentCentre, spdm.data.sp_property.SpTree

class MagneticAxis(*args, **kwargs)
Bases: fytok.modules.Equilibrium.EquilibriumGlobalQuantities.MagneticAxis, spdm.data.sp_property.SpTree

class Qmin(*args, **kwargs)
Bases: fytok.modules.Equilibrium.EquilibriumGlobalQuantities.Qmin, spdm.data.sp_property.SpTree

magnetic_axis: MagneticAxis
Magnetic axis position and toroidal field

class EquilibriumProfiles1D(*args, **kwargs)
Bases: fytok.ontology.imas_lastest.equilibrium._T_equilibrium_profiles_1d

1D profiles of the equilibrium quantities .. note:

```
- psi_norm is the normalized poloidal flux
- psi is the poloidal flux,
- 以psi而不是psi_norm为主坐标, 原因是 profiles1d 中涉及对 psi 的求导和积分
```

area: Expression
Cross-sectional area of the flux surface

b_field_average: Expression
Flux surface averaged modulus of B (always positive, irrespective of the sign convention for the B-field direction).

b_field_max: Expression
Maximum(modulus(B)) on the flux surface (always positive, irrespective of the sign convention for the B-field direction)

b_field_min: Expression
Minimum(modulus(B)) on the flux surface (always positive, irrespective of the sign convention for the B-field direction)

beta_pol: Expression
Poloidal beta profile. Defined as betap = 4 int(p dV) / [R_0 * mu_0 * Ip^2]

darea_dpsi: Expression
Radial derivative of the cross-sectional area of the flux surface with respect to psi

darea_drho_tor: Expression
Radial derivative of the cross-sectional area of the flux surface with respect to rho_tor

dphi_dpsi: Expression

dpressure_dpsi: Expression
Derivative of pressure w.r.t. psi

dpsi_drho_tor: Expression
Derivative of Psi with respect to Rho_Tor

dvolume_dpsi: Expression
Radial derivative of the volume enclosed in the flux surface with respect to Psi

dvolume_drho_tor: Expression
Radial derivative of the volume enclosed in the flux surface with respect to Rho_Tor

elongation: Expression
Elongation

f: Expression
Diamagnetic function (F=R B_Phi)

f_df_dpsi: Expression
Derivative of F w.r.t. Psi, multiplied with F

geometric_axis: _T_equilibrium_profiles_1d_rz1d_dynamic_aos
RZ position of the geometric axis of the magnetic surfaces (defined as (Rmin+Rmax) / 2 and (Zmin+Zmax) / 2 of the surface)

gm1: Expression
Flux surface averaged 1/R^2

gm2: Expression
Flux surface averaged |grad_rho_tor|^2/R^2

gm3: Expression
Flux surface averaged |grad_rho_tor|^2

gm4: Expression
Flux surface averaged 1/B^2

gm5: Expression
Flux surface averaged B^2

gm6: Expression
Flux surface averaged |grad_rho_tor|^2/B^2

gm7: Expression
Flux surface averaged |grad_rho_tor|

gm8: Expression
Flux surface averaged R

gm9: Expression
Flux surface averaged 1/R

grid

j_parallel: Expression
Flux surface averaged parallel current density = average(j.B) / B0, where B0 = Equilibrium/Global/Toroidal_Field/B0

j_tor: Expression
Flux surface averaged toroidal current density = average(j_tor/R) / average(1/R)

magnetic_shear: Expression
Magnetic shear, defined as rho_tor/q . dq/drho_tor

magnetic_z: Expression

major_radius: Expression

mass_density: Expression
Mass density

minor_radius: Expression

phi: Expression
Toroidal flux

pressure: Expression
Pressure

psi: Expression
Poloidal flux

psi_norm: array_type

q: Expression
only positive when toroidal current and magnetic field are in same direction)

Type Safety factor (IMAS uses COCOS=11

r_inboard: Expression
Radial coordinate (major radius) on the inboard side of the magnetic axis

r_outboard: Expression
Radial coordinate (major radius) on the outboard side of the magnetic axis

rho_tor: Expression
Toroidal flux coordinate = sqrt(phi/(pi*b0)), where the toroidal flux, phi, corresponds to time_slice/ profiles_1d/ phi, the toroidal magnetic field, b0, corresponds to vacuum_toroidal_field/b0 and pi can be found in the IMAS constants

rho_tor_norm: Expression
Normalised toroidal flux coordinate. The normalizing value for rho_tor_norm, is the toroidal flux coordinate at the equilibrium boundary (LCFS or 99.x % of the LCFS in case of a fixed boundary equilibium calculation)

rho_volume_norm: Expression
Normalised square root of enclosed volume (radial coordinate). The normalizing value is the enclosed volume at the equilibrium boundary (LCFS or 99.x % of the LCFS in case of a fixed boundary equilibium calculation)

squareness: Expression

squareness_lower_inner: Expression
Lower inner squareness (definition from T. Luce, Plasma Phys. Control. Fusion 55 (2013) 095009)

squareness_lower_outer: Expression
Lower outer squareness (definition from T. Luce, Plasma Phys. Control. Fusion 55 (2013) 095009)

squareness_upper_inner: Expression
Upper inner squareness (definition from T. Luce, Plasma Phys. Control. Fusion 55 (2013) 095009)

squareness_upper_outer: Expression
Upper outer squareness (definition from T. Luce, Plasma Phys. Control. Fusion 55 (2013) 095009)

surface: Expression
Surface area of the toroidal flux surface

trapped_fraction: Expression
Trapped particle fraction

triangularity

triangularity_lower: Expression
Lower triangularity w.r.t. magnetic axis

triangularity_upper: Expression
Upper triangularity w.r.t. magnetic axis

volume: Expression
Volume enclosed in the flux surface

class EquilibriumProfiles2D(*args, **kwargs)
Bases: `fytok.ontology.imas_lastest.equilibrium._T_equilibrium_profiles_2d`

b_field_r: spdm.data.Field.Field
R component of the poloidal magnetic field

b_field_tor: spdm.data.Field.Field
Toroidal component of the magnetic field

b_field_z: spdm.data.Field.Field
Z component of the poloidal magnetic field

grid: spdm.mesh.Mesh.Mesh
Definition of the 2D grid (the content of dim1 and dim2 is defined by the selected grid_type)

grid_type: fytok.modules.Utilities.Identifier
Selection of one of a set of grid types

j_parallel: spdm.data.Field.Field
Defined as (j.B)/B0 where j and B are the current density and magnetic field vectors and B0 is the (signed) vacuum toroidal magnetic field strength at the geometric reference point (R0,Z0). It is formally not the component of the plasma current density parallel to the magnetic field

j_tor: spdm.data.Field.Field
Toroidal plasma current density

phi: spdm.data.Field.Field
Toroidal flux

psi: spdm.data.Field.Field
Values of the poloidal flux at the grid in the poloidal plane

r: spdm.data.Field.Field
Values of the major radius on the grid

theta: spdm.data.Field.Field
Values of the poloidal angle on the grid

type: fytok.modules.Utilities.Identifier
Type of profiles (distinguishes contribution from plasma, vaccum fields and total fields)

z: spdm.data.Field.Field
Values of the Height on the grid

class EquilibriumTimeSlice(*args, **kwargs)
Bases: fytok.ontology.imas_lastest.equilibrium._T_equilibrium_time_slice

class Equilibrium(*args, **kwargs)
Bases: fytok.modules.Utilities.IDS

Description of a 2D, axi-symmetric, tokamak equilibrium; result of an equilibrium code.

Reference:

- O. Sauter and S. Yu Medvedev, "Tokamak coordinate conventions: COCOS" , Computer Physics Communications 184, 2 (2013), pp. 293–302.

6.2 core_profiles

core_profiles 芯部分布位形。

Note: core_profiles.profiles_1d 以归一化的 $\bar{\rho}$ 为磁面坐标，与 equilibrium.profiles_1d 采用 ψ 作为磁面坐标不同。

6.2.1 创建 core_profiles 实例

由专有数据文件，按照预定义的语义读取并转换成 IMAS Ontology。这里 `iterprofiles` 所代表的 `xls` 文件中保存了一系列芯部物理量分布。

```
from fytok.modules.CoreProfiles import CoreProfiles

core_profiles = CoreProfiles(f"file+iterprofiles://./data/15MA Inductive at burn-ASTRA.xls#core_
↪profiles")
```

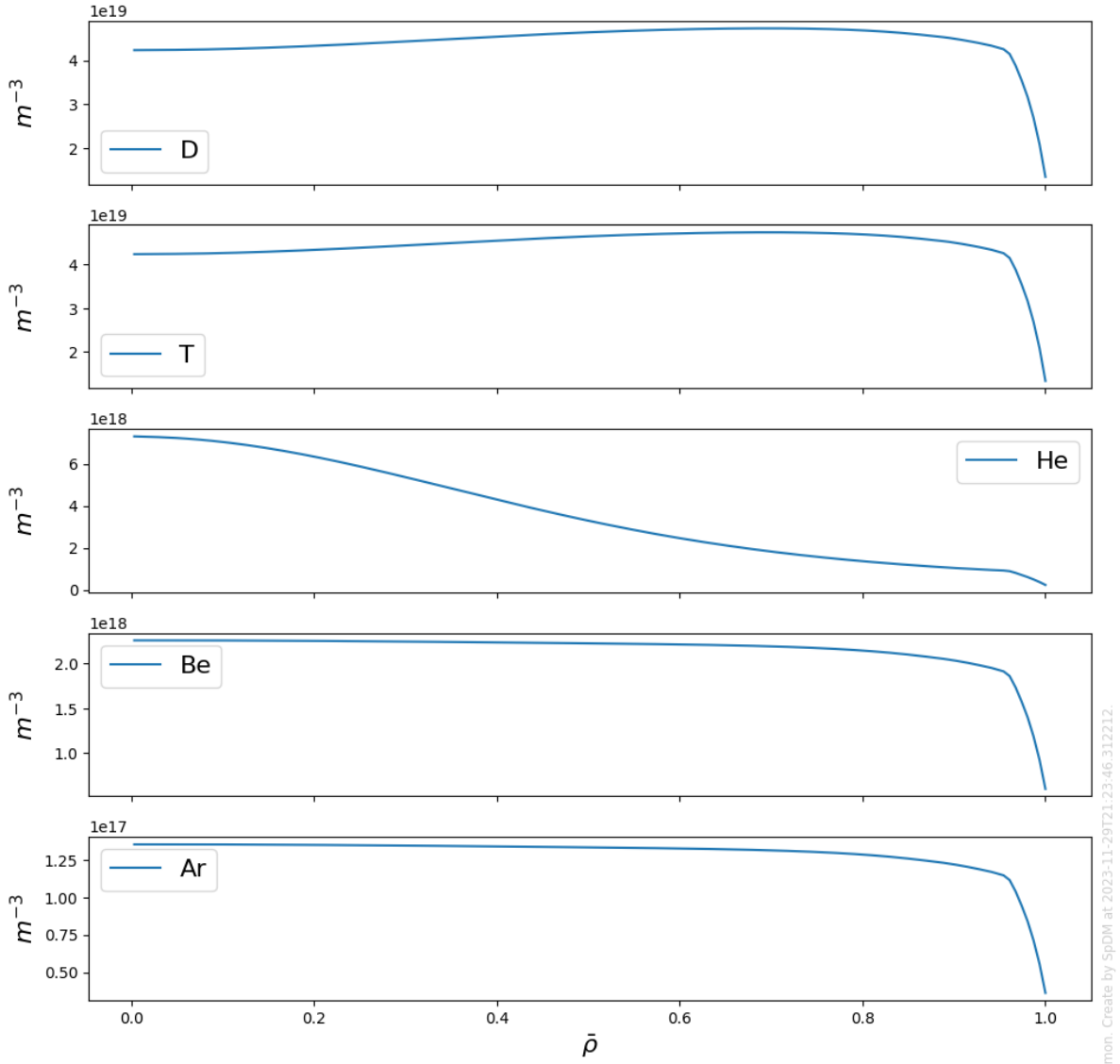
```
2023-11-29 21:23:45,929 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_iterprofiles
```

6.2.2 可视化

绘出一维分布

```
import spdm.view.View as sp_view

fig = sp_view.plot(
    [(ion.density_thermal, ion.label) for ion in core_profiles.time_slice.current.profiles_1d.ion],
    x_value=core_profiles.time_slice.current.profiles_1d.grid.rho_tor_norm,
    x_label=r"$\bar{\rho}$",
    y_label=r"$[m^{-3}]$",
)
```



6.2.3 class CoreProfiles

class CoreProfiles1D(*args, **kwargs)

Bases: fytok.ontology.imas_lastest.utilities._T_core_profiles_profiles_1d

class EFieldVectorComponents(*args, **kwargs)

Bases: fytok.modules.CoreProfiles.CoreProfiles1D.EFieldVectorComponents, spdm.data.sp_property.SpTree

diamagnetic: spdm.data.Expression.Expression

poloidal: spdm.data.Expression.Expression

radial: spdm.data.Expression.Expression

toroidal: spdm.data.Expression.Expression

Electrons
alias of fytok.modules.CoreProfiles.CoreProfilesElectrons

Ion
alias of fytok.modules.CoreProfiles.CoreProfilesIon

Neutral
alias of fytok.modules.CoreProfiles.CoreProfilesNeutral

beta_pol

conductivity_parallel: Expression
Parallel conductivity

coulomb_logarithm

current_parallel_inside: Expression
Parallel current driven inside the flux surface. Cumulative surface integral of j_total

e_field: EFieldVectorComponents
Electric field, averaged on the magnetic surface. E.g for the parallel component, average(E.B) / B0, using core_profiles/vacuum_toroidal_field/b0

electron_collision_time

electrons: CoreProfilesElectrons
Quantities related to the electrons

ffprime: Expression

grid: CoreRadialGrid
Radial grid

ion: AoS[CoreProfilesIon]
Quantities related to the different ion species, in the sense of isonuclear or isomolecular sequences. Ionisation states (or other types of states) must be differentiated at the state level below

j_bootstrap: Expression
Bootstrap current density = average(J_Bootstrap.B) / B0, where B0 = Core_Profiles/Vacuum_Toroidal_Field/ B0

j_non_inductive: Expression
Non-inductive (includes bootstrap) parallel current density = average(jni.B) / B0, where B0 = Core_Profiles/Vacuum_Toroidal_Field/ B0

j_ohmic: Expression
Ohmic parallel current density = average(J_Ohmic.B) / B0, where B0 = Core_Profiles/Vacuum_Toroidal_Field/ B0

j_tor: Expression
Total toroidal current density = average(J_Tor/R) / average(1/R)

j_total: Expression
Total parallel current density = average(jtot.B) / B0, where B0 = Core_Profiles/Vacuum_Toroidal_Field/ B0

magnetic_shear: Expression
Magnetic shear, defined as rho_tor/q . dq/drho_tor

momentum_tor: Expression
Total plasma toroidal momentum, summed over ion species and electrons weighted by their density and major radius, i.e. sum_over_species(n*R*m*Vphi)

n_i_thermal_total: Expression
Total ion thermal density (sum over species and charge states)

n_i_total

n_i_total_over_n_e: Expression
Ratio of total ion density (sum over species and charge states) over electron density. (thermal+non-thermal)

neutral: AoS[CoreProfilesNeutral]
Quantities related to the different neutral species

phi_potential: Expression
Electrostatic potential, averaged on the magnetic flux surface

pprime: Expression

pressure

pressure_ion_total: Expression
Total (sum over ion species) thermal ion pressure

pressure_parallel: Expression
Total parallel pressure (electrons+ions, thermal+non-thermal)

pressure_perpendicular: Expression
Total perpendicular pressure (electrons+ions, thermal+non-thermal)

pressure_thermal: Expression
Thermal pressure (electrons+ions)

q: Expression
only positive when toroidal current and magnetic field are in same direction)

Type Safety factor (IMAS uses COCOS=11

rotation_frequency_tor_sonic: Expression
Derivative of the flux surface averaged electrostatic potential with respect to the poloidal flux, multiplied by -1. This quantity is the toroidal angular rotation frequency due to the ExB drift, introduced in formula (43) of Hinton and Wong, Physics of Fluids 3082 (1985), also referred to as sonic flow in regimes in which the toroidal velocity is dominant over the poloidal velocity

t_i_average: Expression
Ion temperature (averaged on charge states and ion species)

t_i_average_fit: _T_core_profiles_1D_fit
Information on the fit used to obtain the t_i_average profile

time: float

zeff: Expression
Effective charge

zeff_fit: _T_core_profiles_1D_fit
Information on the fit used to obtain the zeff profile

class CoreProfilesTimeSlice(*args, **kwargs)
Bases: spdm.data.TimeSeries.TimeSlice

class CoreProfiles(*args, **kwargs)
Bases: fytok.modules.Utilities.Module

6.3 core_transport

core_transport 芯部输运系数

6.3.1 创建 core_transport 实例

```
import numpy as np
from fytok.modules.CoreTransport import CoreTransport
from spdm.data.Expression import Piecewise, Variable

R0 = 1.85
B0 = 1.8

_x = Variable(0, "rho_tor_norm", label=r"\bar{\rho}_{tor}")

x = np.linspace(0, 1, 128)

# Core profiles
r_ped = 0.96  # np.sqrt(0.88)
i_ped = np.argmin(np.abs(x - r_ped))

# Core Transport

Cped = 0.17
Ccore = 0.4
# Function( profiles["Xi"].values,bs_r_norm)  Cped = 0.2
chi = Piecewise([Ccore * (1.0 + 3 * (_x**2)), Cped], [(_x < r_ped), (_x >= r_ped)], label=r"\chi")
chi_e = Piecewise([0.5 * Ccore * (1.0 + 3 * (_x**2)), Cped], [(_x < r_ped), (_x >= r_ped)], label=r"\
↪chi_e")

D = 0.1 * (chi + chi_e)

v_pinch_ne = -0.6 * D * _x / R0
v_pinch_Te = 2.5 * chi_e * _x / R0
v_pinch_ni = D * _x / R0
v_pinch_Ti = chi * _x / R0

time_slice = {
    "profiles_1d": {
        "grid_d": {"rho_tor_norm": x},
        "electrons": {
            "label": "e",
            "particles": {"d": D, "v": v_pinch_ne},
            "energy": {"d": chi_e, "v": v_pinch_Te},
        },
        "ion": [
            {
                "label": "D",
                "particles": {"d": D, "v": v_pinch_ni},
                "energy": {"d": chi, "v": v_pinch_Ti},
            },
            # {
            #     "label": "T",
            #     "particles": {"d": D, "v": v_pinch_ni},
```

(continues on next page)

(continued from previous page)

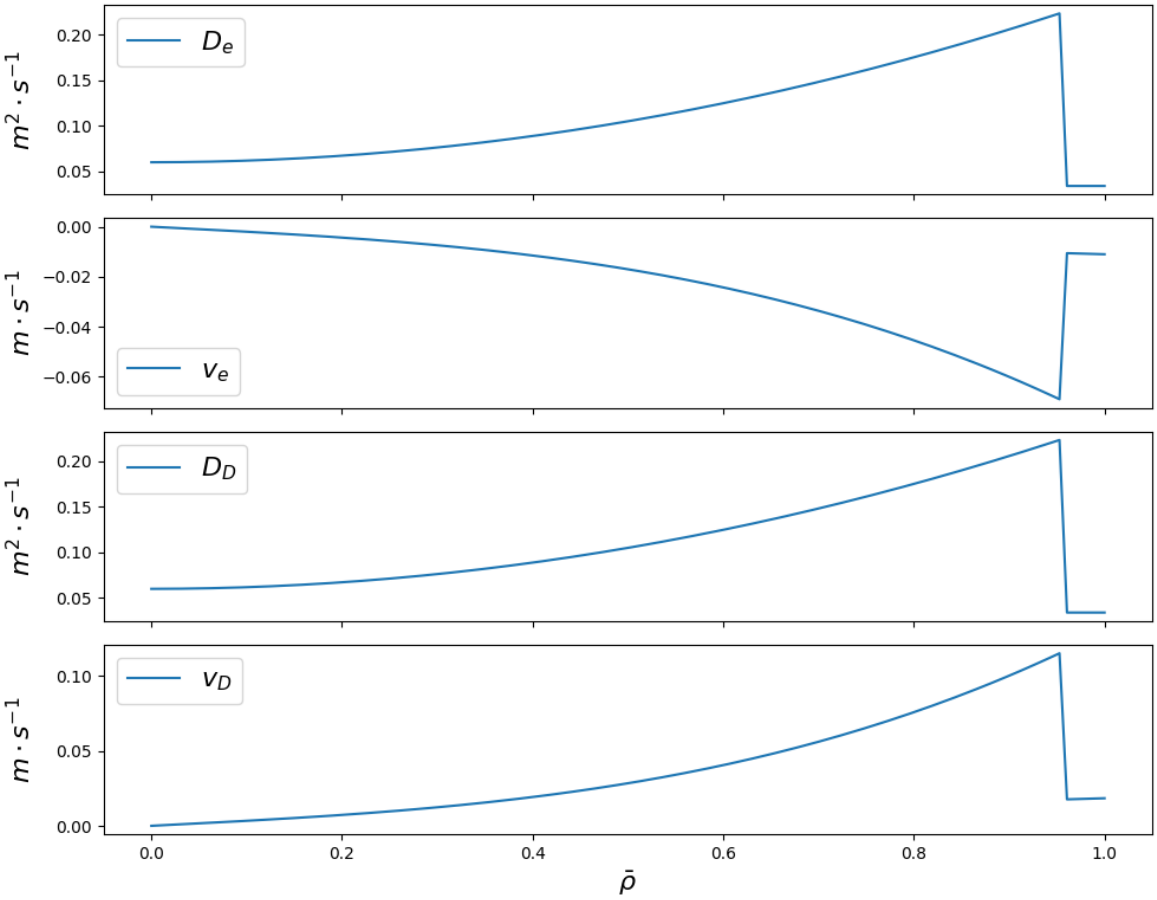
```
            #     "energy": {"d": chi, "v": v_pinch_Ti},
            # },
            # {
            #     "label": "He",
            #     "particles": {"d": D, "v": v_pinch_ni},
            #     "energy": {"d": chi, "v": v_pinch_Ti},
            # },
        ],
    }
}
core_transport = CoreTransport({"model": [{"code": {"name": "dummy"}, "time_slice": [time_slice]}]})
```

6.3.2 可视化

```
import spdm.view.View as sp_view

core_transport_1d = core_transport.model[0].time_slice.current.profiles_1d
fig = sp_view.plot(
    sum(
        [
            [(ion.particles.d, f"$D_{{{ion.label}}}$"), (ion.particles.v, f"$v_{{{ion.label}}}$")]
            for ion in core_transport_1d.ion
        ],
        [(core_transport_1d.electrons.particles.d, f"$D_e$"), (core_transport_1d.electrons.particles.v,
↪ f"$v_e$")],
    ),
    x_value=core_transport_1d.grid_d.rho_tor_norm,
    x_label=r"$\bar{\rho}$",
    y_label=r"$[m^{-3}]$",
)
```

```
2023-11-29 21:23:47,208 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module fytok.plugins.core_transport.model.dummy
```



6.3.3 class CoreTransport

class CoreTransportProfiles1D(*args, **kwargs)

Bases: fytok.ontology.imas_lastest.core_transport._T_core_transport_model_profiles_1d

Electrons

alias of fytok.modules.CoreTransport.CoreTransportElectrons

Ion

alias of fytok.modules.CoreTransport.CoreTransportIon

Neutral

alias of fytok.modules.CoreTransport.CoreTransportNeutral

conductivity_parallel: Expression

Parallel conductivity

e_field_radial: Expression

Radial component of the electric field (calculated e.g. by a neoclassical model)

electrons: fytok.modules.CoreTransport.CoreTransportElectrons

Transport quantities related to the electrons

grid_d: fytok.modules.Utilities.CoreRadialGrid
Grid for effective diffusivities and parallel conductivity

grid_flux: _T_core_radial_grid
Grid for fluxes

grid_v: _T_core_radial_grid
Grid for effective convections

ion: spdm.data.AoS.AoS[fytok.modules.CoreTransport.CoreTransportIon]
Transport coefficients related to the various ion species, in the sense of isonuclear or isomolecular sequences. Ionisation states (and other types of states) must be differentiated at the state level below

momentum_tor: _T_core_transport_model_1_momentum
Transport coefficients for total toroidal momentum equation

neutral: spdm.data.AoS.AoS[fytok.modules.CoreTransport.CoreTransportNeutral]
Transport coefficients related to the various neutral species

time: float

total_ion_energy: _T_core_transport_model_1_energy
Transport coefficients for the total (summed over ion species) energy equation

class CoreTransport(*args, **kwargs)
Bases: fytok.ontology.imas_lastest.core_transport._T_core_transport

6.4 core_sources

6.4.1 class CoreSources

class CoreSourcesProfiles1D(*args, **kwargs)
Bases: fytok.ontology.imas_lastest.core_sources._T_core_sources_source_profiles_1d

conductivity_parallel: Expression
Parallel conductivity due to this source

current_parallel_inside: Expression
Parallel current driven inside the flux surface. Cumulative surface integral of j_parallel

electrons: CoreSourcesElectrons
Sources for electrons

grid: CoreRadialGrid
Radial grid

ion: AoS[CoreSourcesIon]
Source terms related to the different ions species, in the sense of isonuclear or isomolecular sequences. Ionisation states (and other types of states) must be differentiated at the state level below

j_parallel: Expression
Parallel current density source, average(J.B) / B0, where B0 = core_sources/ vacuum_toroidal_field/b0

momentum_tor: Expression
Source term for total toroidal momentum equation

momentum_tor_j_cross_b_field: Expression
Contribution to the toroidal momentum source term (already included in the momentum_tor node) corresponding to the toroidal torques onto the thermal plasma due to Lorentz force associated with radial currents. These currents appear as return-currents (enforcing quasi-neutrality, div(J)=0) balancing radial currents of non-thermal particles, e.g. radial currents of fast and trapped neutral-beam-ions.

neutral: AoS[CoreSourcesNeutral]
Source terms related to the different neutral species

torque_tor_inside: Expression
Toroidal torque inside the flux surface. Cumulative volume integral of the source term for the total toroidal momentum equation

total_ion_energy: Expression
Source term for the total (summed over ion species) energy equation

total_ion_energy_decomposed: _T_core_sources_source_profiles_1d_energy_decomposed_2
Decomposition of the source term for total ion energy equation into implicit and explicit parts

total_ion_power_inside: Expression
Total power coupled to ion species (summed over ion species) inside the flux surface. Cumulative volume integral of the source term for the total ion energy equation

class CoreSources(*args, **kwargs)
Bases: fytok.modules.Utilities.IDS

CHAPTER

SEVEN

MODULE PLUGIN

复杂的模拟通常需要结合许多的物理程序，这些程序可能由社区内不同的研究者提供，并用不同的代码编写。特别是有历史年代的物理程序，或者涉及到许多密集型计算的科学程序和算法，大都使用 C++或者 Fortran 编写的。

为了让他们能够协同工作，需要一个额外的中间层来协调特定的物理代码的执行，并负责数据传递。这一层就是“工作流”。

通常情况下，工作流协调器是用动态编码元素语言实现的，如，FyTok 中选择使用 Python 语言直接来调度不同的物理程序。这样的话，需要一种“封装器”来帮助本地代码语言和调度协调器语言之间充当介质。使用“封装器”将物理程序封装成统一的调度器的组件，提供统一的 API 来调用和使用这些组件。

因此，FyTok 中提供灵活的插件功能来统一封装和组织用户的第三方物理程序，增加物理程序集成的灵活性，进一步降低程序集成的复杂度。

FyTok 中采用基于插件的模块化设计，这种机制运行将多种语言（如 Fortran，C++，Matlab 等）编写的物理代码集成到以 Python 为主体的复杂计算流程中。用于构建托克马克的的 Ontology 清楚描述了和托克马克相关的物理概念或者装置组件，称为 Actor. FyTok 将不同的物理程序绑定到对应的 Actor 上，通过插件机制灵活封装、组织管理、调用。

7.1 目录结构

第三方物理程序无需将代码打包在 FyTok 的框架内，用户仅需将打包好的代码的目录暴露在 FyTok 可检索的路径下。

该目录按照下述规范的组织结构：

```
{work_dir}/python/fytok/plugins/< 模块类型>/< 物理模块名称>
```

其中：

- {work_dir}是用户本地指定的任意目录
- <模块类型>严格遵守 IMAS 中对物理概念或者装置组件描述的分类，常用的有：
  - equilibrium：efit, freegs, ATEC, FyEq…
  - transport_slover: BITS, FyTrans, onetwo,…
  - core_transport/model: cgyro, glf23, gyro, neo, tglf, tgyro,…
  - core_sources/source: genray, cql3d,…
- <物理程序名称>：第三方物理程序名称
  - 若是程序功能和装置相关，建议程序名称 _装置名称的形式命名，如 efit_east

例如，被集成到 FyTok 中的平衡程序 efit 的目录组织：

```
{work_dir}/python/fytok/plugins/equilibrium/efit_east
```

为了方便物理模块的管理、调用和学习推广，FyTok 要求每个物理模块的目录内必须至少包含以下文件：

- README.md：物理模块的说明文档，包含物理模块的功能描述、使用方法、输入输出参数说明等
- __init__.py: 物理模块的封装脚本，用于封装原始的物理模块，提供统一的 API，供调度器调用
- <module_name>.py：物理模块的封装脚本，用于封装原始的物理模块，提供统一的 API，供调度器调用
- 其他：物理模块的其他文件，如物理模块的源代码、测试用例,配置文件等

以 efit_east 模块为例，

- __init__.py

```
# file: __init__.py
### 指定当前目录efit.east.py文件中需要导入运行环境中的模块 EquilibriumEFITEAST

from .efit_east import EquilibriumEFITEAST

### __all__ 关联一个模块列表，当执行 from xx import * 时，就会导入该列表中指定的所有模块

__all__ = ["EquilibriumEFITEAST"]
```

- <module_name>.py

封装后的模块作为 actor 直接作为工作流的组件模块运行，因此，该文件包含以下几个功能：

（1）提供定义明确的应用程序接口，用于工作流中调用，用于 * 调用用户从系统库或二进制可执行文件中提供的本地代码方法。

```
- @Equilibrium.register(["efit_east"]) 明确被集成的物理程序的名称，并将该名称暴露给FyTok.

- 封装后的actor实质上是一个Python 类，如：class EquilibriumEFITEAST() ，用__init.py__ 文件进行管理
```

（2）明确功能逻辑

```
 - refresh(): 迭代当前时间片，更新最后一个时间片

 - advance(): 推进时间片，更新下一个时间片
```

（3）向工作流隐藏原始程序的底层运行方式及复杂性

```
@Equilibrium.register(["efit_east"])
class EquilibriumEFITEAST(FyEqAnalyze):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

    def refresh(
        self,
        *args,
        time: float | None = None,  # unit: s
        magnetics: Magnetics | None = None,
        pf_active: PFActive | None = None,
        tf: TF | None = None,
```

(continues on next page)

(continued from previous page)

```
        wall: Wall | None = None,
        **kwargs,
    ):
        """update the last time slice, base on profiles_2d[-1].psi, and core_profiles_1d, wall, pf_
↪active"""

    def advance(self, *args, dt=0.1, **kwargs):
        """
        Update the next time slice.
        """
        return super().advance(*args, **kwargs)
```

7.2 插件调用

- 添加路径到环境变量

```
export PYTHONPATH={work_dir}/python/fytok/plugins/< 模块类型>/< 物理模块名称>:$PYTHONPATH
```

- 调用：IMAS DD 中每个 IDS 都有 code 子节点，用来描述生成此 IDS 的物理代码的通用信息。FyTok 中使用该组织方式来调用代码。如下例子，可以在当前环境中查找名称为 efit_east 的 actor，并调用它。

```
tok = Tokamak(
        f"east+mdsplus://{WORKSPACE}/fytok_data/mdsplus/~t/?disabled_entry=efit_east&shot={shot}",
        equilibrium={"code": {"name": "efit_east"}}
    )
```

CHAPTER

EIGHT

插件：FY_EQ 二维平衡分析

8.1 主要功能

- 二维平衡分析
- 磁面平均
- 磁坐标转换 (TODO)

8.2 创建 equilibrium 实例

指定调用插件 fy_eq 的 equilibrium 实例。

```
from fytok.modules.Equilibrium import Equilibrium

equilibrium = Equilibrium({"code": {"name": "fy_eq"}}, "file+geqdsk://./data/g070754.05000#equilibrium
↪")
```

```
2023-11-29 21:05:57,373 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module fytok.plugins.equilibrium.fy_eq
2023-11-29 21:05:57,380 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_geqdsk
```

8.3 可视化

8.3.1 2D 磁面

自动确定磁轴 o-point，x-point，和最外层闭合磁面

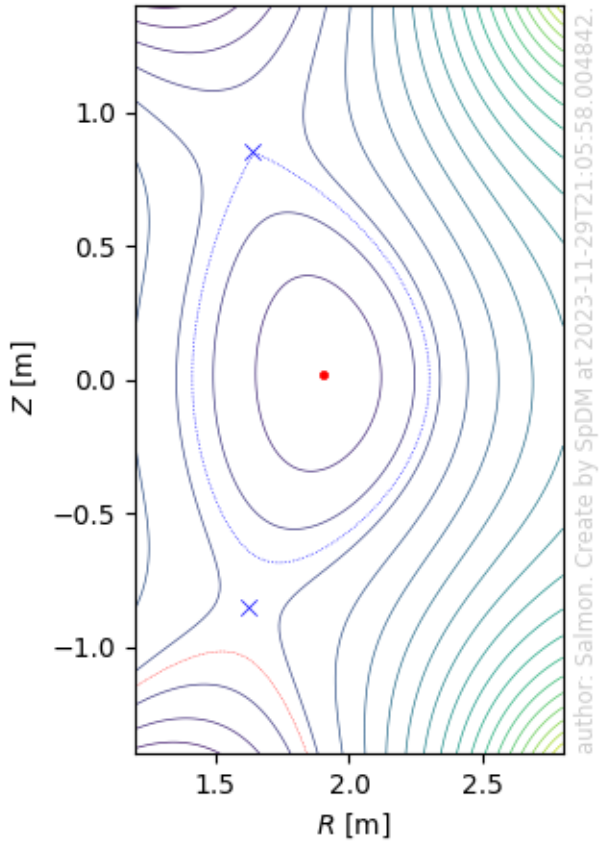
```
import spdm.view.View as sp_view

fig =sp_view.display(equilibrium)
```

```
2023-11-29 21:05:57,532 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.view.view_matplotlib
2023-11-29 21:05:57,581 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.mesh.mesh_rectangular
```

(continues on next page)

(continued from previous page)



8.3.2 1D 磁面坐标的函数

除了基础量 ffprime, pprime, 和二维 psi 外，其他物理量都是通过计算得出的

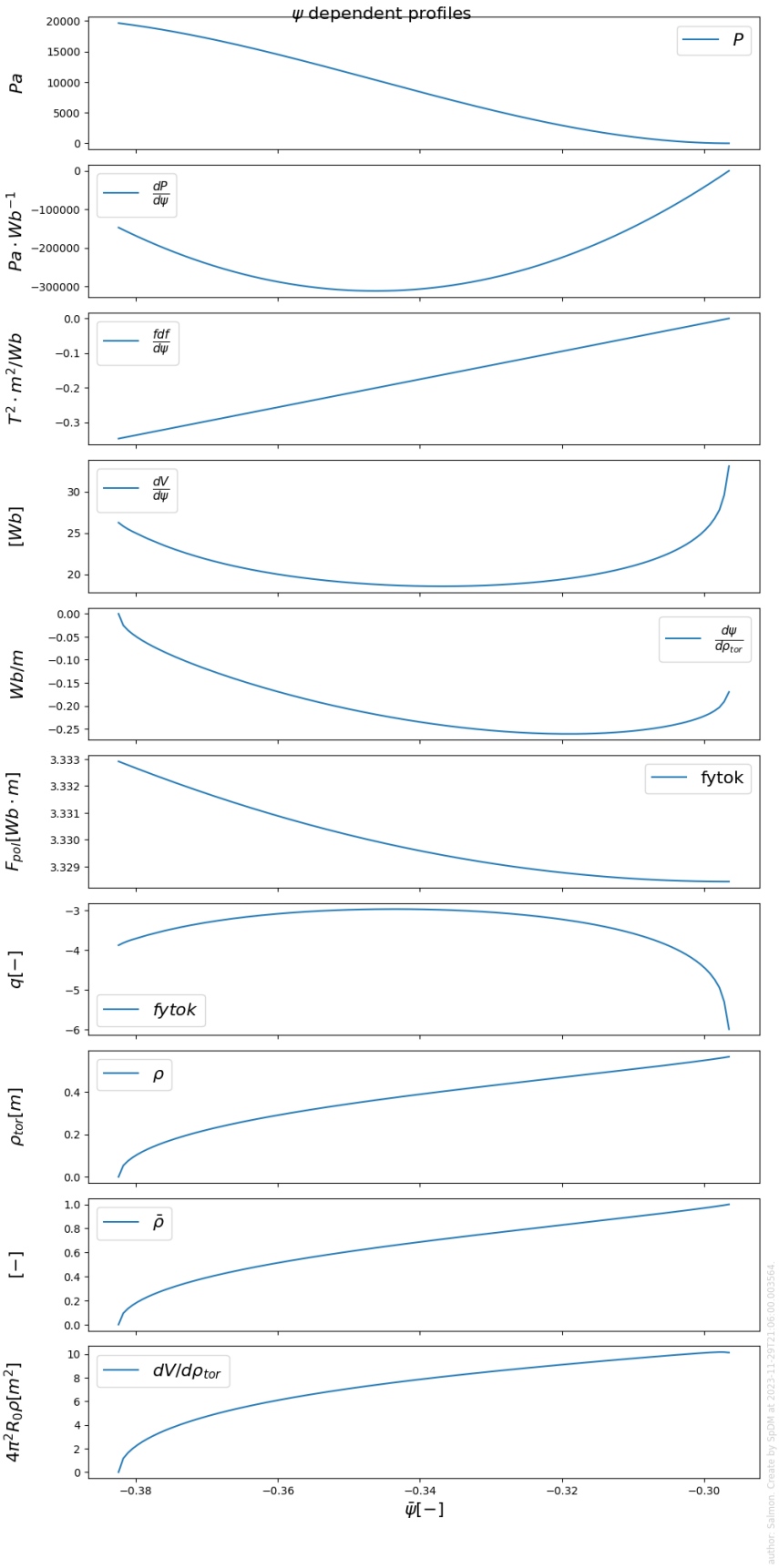
```
eq_profiles_1d = equilibrium.time_slice.current.profiles_1d

profs = sp_view.plot(
    [
        eq_profiles_1d.pressure,
        eq_profiles_1d.dpressure_dpsi,
        eq_profiles_1d.f_df_dpsi,
        ((eq_profiles_1d.dvolume_dpsi, {"label": r"$\frac{dV}{d\psi}$"}), {"y_label": r"$[Wb]$"}),
        (eq_profiles_1d.dpsi_drho_tor, {"label": r"$\frac{d\psi}{d\rho_{tor}}$"}),
        ((eq_profiles_1d.f, {"label": r"fytok"}), {"y_label": r"$F_{pol} [Wb\cdot m]$"}),
        ((eq_profiles_1d.q, {"label": r"$fytok$"}), {"y_label": r"$q [-]$"}),
        ((eq_profiles_1d.rho_tor, {"label": r"$\rho$"}), {"y_label": r"$\rho_{tor}[m]$"}),
        ((eq_profiles_1d.rho_tor_norm, {"label": r"$\bar{\rho}$"}), {"y_label": r"[-]"}),
        ((eq_profiles_1d.dvolume_drho_tor, {"label": r"$dV/d\rho_{tor}$"}), {"y_label": r"$4\pi ^ 2 R_
↪0 \rho[m ^ 2]$"}),
    ],
    x_axis=equilibrium.time_slice.current.profiles_1d.psi,
    x_label=r"$\bar{\psi}[-]$",
```

(continues on next page)

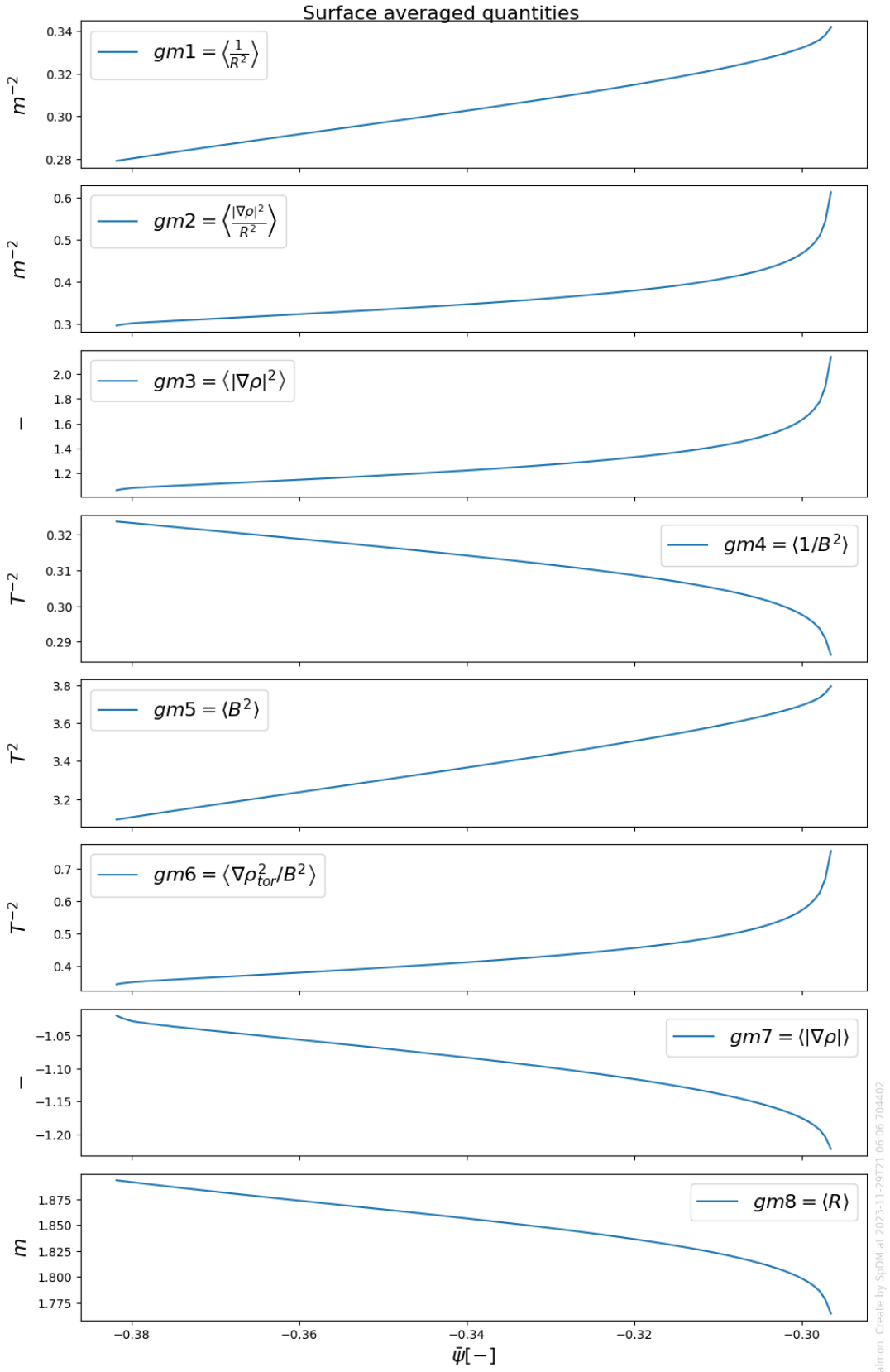
(continued from previous page)

```
    title=r"$\psi$ dependent profiles",
)
```


ψ dependent profiles
P
Pa
dP/dψ
Pa · Wb⁻¹
fdf/dψ
T² · m²/Wb
dV/dψ
[Wb]
dψ/dρ_tor
Wb/m
fytok
F_pol[Wb · m]
fytok
q[−]
ρ
ρ_tor[m]
ρ̃
[−]
dV/dρ_tor
4π²R₀ρ[m²]
ψ̃[−]

8.4 1D：磁面平均给出几何量

```
surfs = sp_view.plot(
    [
        (eq_profiles_1d.gm1, {"label": r"$gm1=\left<\frac{1}{R^2}\right>$"}),
        (eq_profiles_1d.gm2, {"label": r"$gm2=\left<\frac{\left|\nabla \rho\right|^2}{R^2}\right>$"}),
        (eq_profiles_1d.gm3, {"label": r"$gm3=\left<\left|\nabla \rho\right|^2\right>$"}),
        (eq_profiles_1d.gm4, {"label": r"$gm4=\left<1/B^2\right>$"}),
        (eq_profiles_1d.gm5, {"label": r"$gm5=\left<B^2\right>$"}),
        (eq_profiles_1d.gm6, {"label": r"$gm6=\left<\nabla \rho_{tor}^2/ B^2 \right>$"}),
        (eq_profiles_1d.gm7, {"label": r"$gm7=\left<\left|\nabla \rho\right|\right>$"}),
        (eq_profiles_1d.gm8, {"label": r"$gm8=\left<R\right>$"}),
    ],
    x_axis=equilibrium.time_slice.current.profiles_1d.psi.__array__()[1:],
    # x_value=tok.equilibrium.time_slice.current.profiles_1d.psi_norm[1:],
    x_label=r"$\bar{\psi}[-]$",
    title=r"Surface averaged quantities",
)
```


Surface averaged quantities
$gm1 = \left\langle \frac{1}{R^2} \right\rangle$
$gm2 = \left\langle \frac{|\nabla\rho|^2}{R^2} \right\rangle$
$gm3 = \langle |\nabla\rho|^2 \rangle$
$gm4 = \langle 1/B^2 \rangle$
$gm5 = \langle B^2 \rangle$
$gm6 = \langle \nabla\rho_{tor}^2/B^2 \rangle$
$gm7 = \langle |\nabla\rho| \rangle$
$gm8 = \langle R \rangle$
m^{-2}
m^{-2}
$-$
T^{-2}
T^2
T^{-2}
$-$
m
$\bar{\psi}[-]$
author: Salmon, Create by SpDM at 2023-11-29T21:06:06.704402

CHAPTER

NINE

插件：FY_TRANS 输运方程求解

9.1 主要功能

- 汇总 core_transport.model 和 core_sources.source
- 求解输运方程

9.2 创建 Tokamak 实例

输入参数中 transport_solver 的内容，用于指定输运方程求解器为 fy_trans，其中 equations 指定了待求解输运方程的主物理量和边界条件类型。

```
from fytok.Tokamak import Tokamak
import numpy as np

tokamak = Tokamak(
    f"file+iterprofiles://./data/15MA Inductive at burn-ASTRA.xls",
    f"file+geqdsk://./data/g900003.00230_ITER_15MA_eqdsk16HR.txt",
    device="iter",
    shot=900003,
    core_transport={"model": [{"code": {"name": "dummy"}}]},
    core_sources={"source": [{"code": {"name": "dummy"}}]},
    transport_solver={
        "code": {
            "name": "fy_trans",
            "parameters": {
                "rho_tor_norm": np.linspace(0.01, 0.995, 128),
                "control_parameters": {
                    "bvp_rms_mask": [0.96],
                    "hyper_diff": 0.0001,
                    "max_nodes": 512,
                    "verbose": 2,
                },
            },
        },
        "primary_coordinate": {"index": (index := 0), "label": r"\bar{rho}_{tor_norm}"},
        "equations": [
            # fmt: off
            {"identifier": "ion/D/density_thermal",    "boundary_condition": [2, 1]},
            {"identifier": "ion/T/density_thermal",    "boundary_condition": [2, 1]},
```

(continues on next page)

(continued from previous page)

```
            {"identifier": "ion/D/temperature",        "boundary_condition": [2, 1]},
            {"identifier": "ion/T/temperature",        "boundary_condition": [2, 1]},
            {"identifier": "electrons/temperature",    "boundary_condition": [2, 1]},
            # fmt: on
        ],
    },
)
```

```
2023-11-29 21:16:26,124 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_xml
2023-11-29 21:16:26,558 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_iterprofiles
2023-11-29 21:16:26,635 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_geqdsk
```

9.3 调用求解器 transport_solver

给定初始值和边界值，通过多次非线性迭代求解，直到收敛。并可根据需要，自动加密网格。

```
tokamak.transport_solver.refresh(
    time=0.0,
    initial_value=[1.0e19, 1.0e19, 1000, 1000, 1000],
    boundary_value=[[[0.0], [2.0e19]], [[0.0], [2.0e19]], [[0.0], [1000]], [[0.0], [1000]], [[0.0],␣
↪[1000]]],
)
```

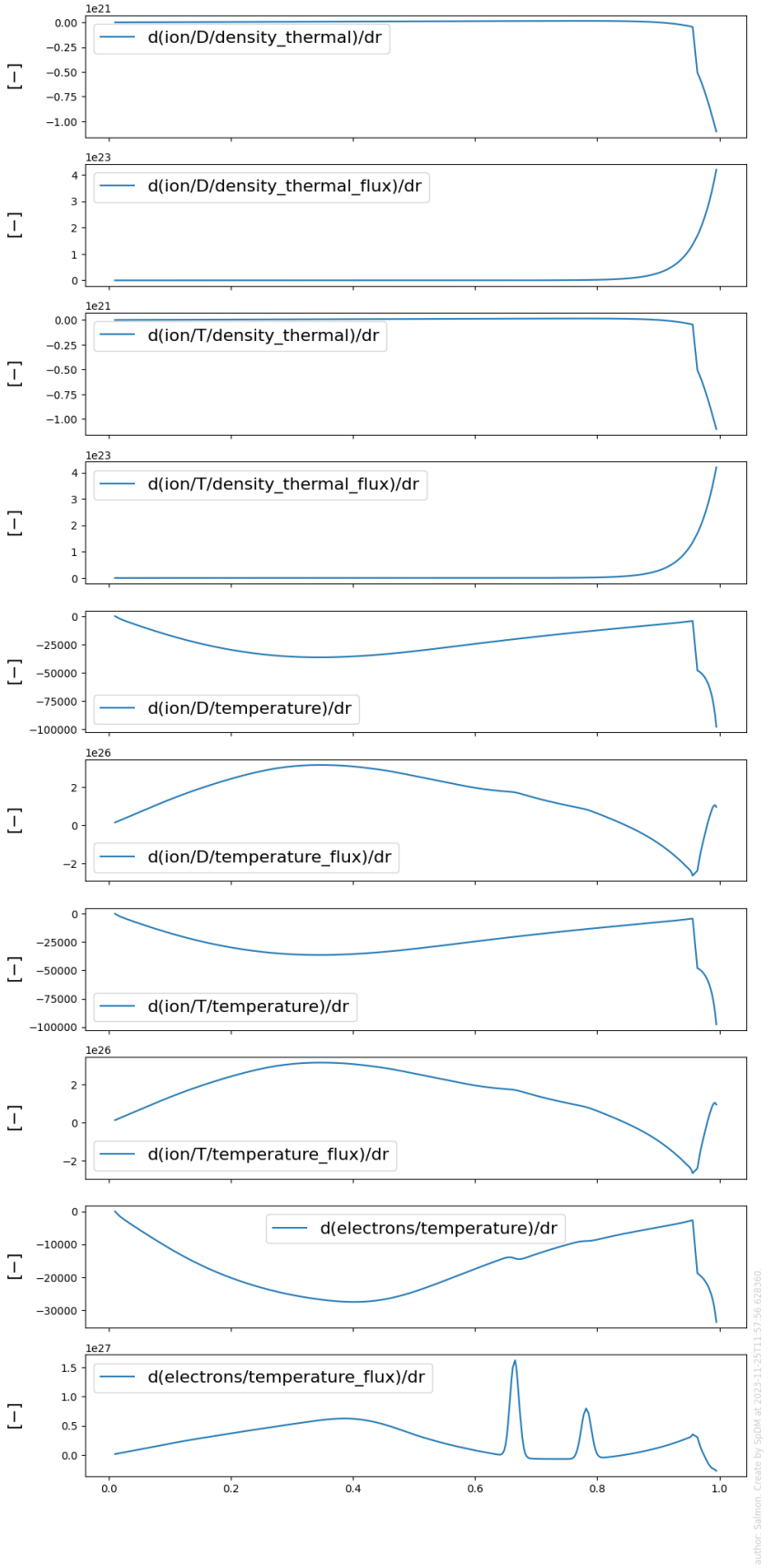
```
2023-11-29 21:16:29,372 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module fytok.plugins.transport_solver_numerics.fy_trans
2023-11-29 21:16:29,764 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module fytok.plugins.equilibrium.fy_eq
2023-11-29 21:16:29,796 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.mesh.mesh_rectangular
2023-11-29 21:16:32,038 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module fytok.plugins.core_transport.model.dummy
2023-11-29 21:16:32,094 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module fytok.plugins.core_sources.source.dummy
2023-11-29 21:16:32,162 [   fytok]    DEBUG: /home/salmon/workspace/fytok/python/fytok/modules/
↪Utilities.py:121:execute: Execute fytok.plugins.transport_solver_numerics.fy_trans-0.0.1 [fytok]
2023-11-29 21:16:32,164 [   fytok]     INFO: Solve transport equations : ion/D/density_thermal,ion/
↪T/density_thermal,ion/D/temperature,ion/T/temperature,electrons/temperature
  Iteration    Max residual  Max BC residual  Total nodes    Nodes added
      1          7.37e+01       3.54e+04          128            142
      2          2.51e-02       3.72e-09          270            31
      3          6.99e-03       2.84e-14          301             9
      4          3.73e-03       3.20e+01          310             1
      5          9.50e-04       2.84e-14          311             0
Solved in 5 iterations, number of nodes 311.
Maximum relative residual: 9.50e-04
Maximum boundary residual: 2.84e-14
2023-11-29 21:16:39,469 [   fytok]    DEBUG: /home/salmon/workspace/fytok/python/fytok/plugins/
↪transport_solver_numerics/fy_trans.py:688:execute: Solve BVP success: The algorithm converged to␣
↪the desired accuracy. , 5 iterations
```

9.4 中间变量

```
from spdm.view import View as sp_view
```

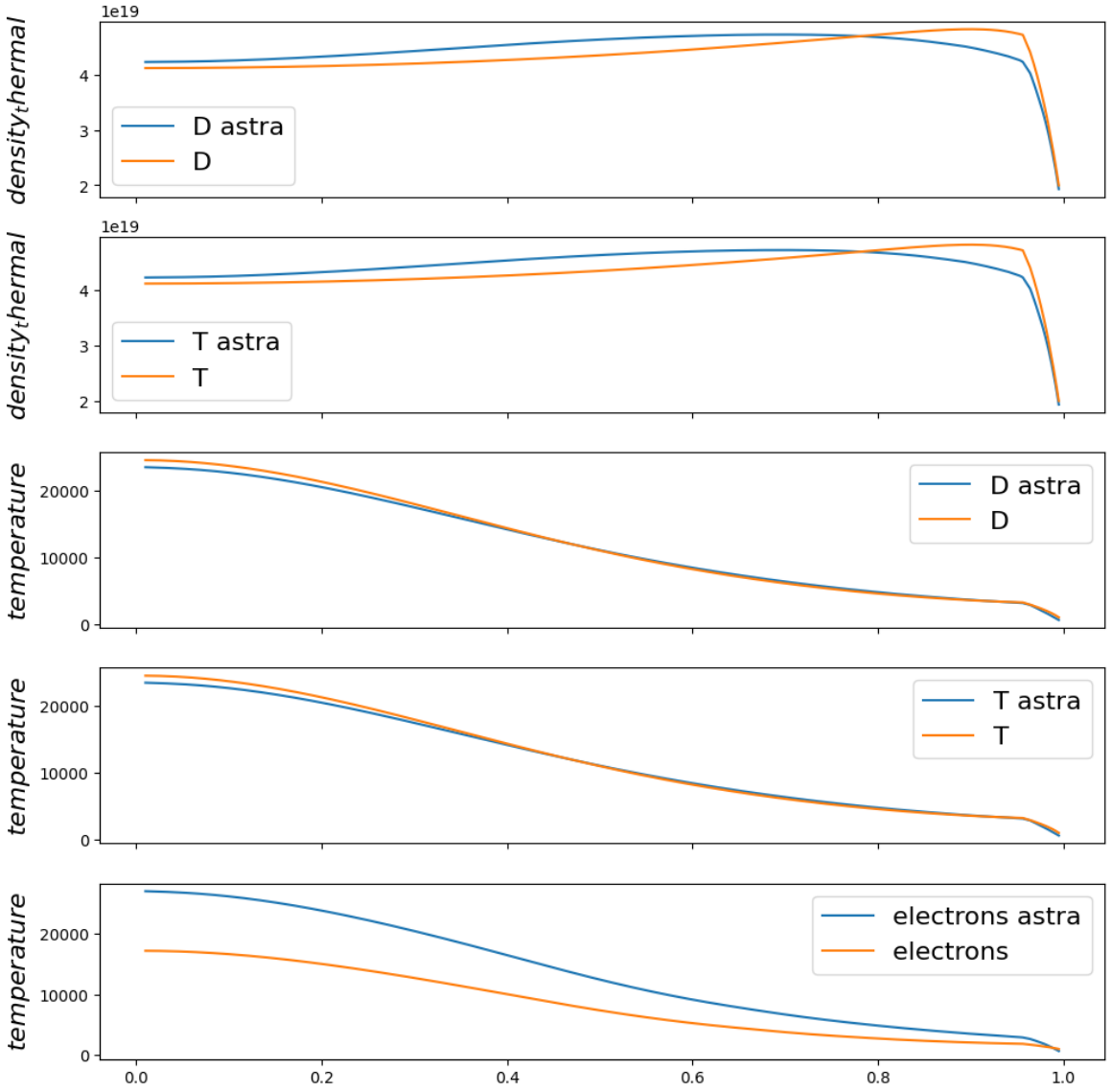
```
solver_1d = tokamak.transport_solver.time_slice.current
fig = sp_view.plot(
    sum(
        [
            [
                # (equ.primary_quantity.profile, equ.primary_quantity.identifier),
                # (equ.primary_quantity.flux, f"{equ.primary_quantity.identifier}_flux"),
                (equ.primary_quantity.d_dr,f"d({equ.primary_quantity.identifier})/dr"),
                (equ.primary_quantity.dflux_dr, f"d({equ.primary_quantity.identifier}_flux)/dr"),
            ]
            for equ in solver_1d.equation
        ],
        [],
    ),
    x_axis=solver_1d.grid.rho_tor_norm,
)
```

```
2023-11-25 11:57:55,423 [   fytok]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.view.view_matplotlib
```


d(ion/D/density_thermal)/dr
d(ion/D/density_thermal_flux)/dr
d(ion/T/density_thermal)/dr
d(ion/T/density_thermal_flux)/dr
d(ion/D/temperature)/dr
d(ion/D/temperature_flux)/dr
d(ion/T/temperature)/dr
d(ion/T/temperature_flux)/dr
d(electrons/temperature)/dr
d(electrons/temperature_flux)/dr
[−]
author: Salmon. Create by SpDM at 2023-11-25T11:57:36.628360.

9.5 计算结果

```
core_profiles_1d = tokamak.core_profiles.time_slice.current.profiles_1d
fig = sp_view.plot([
    ([
        (core_profiles_1d.get(equ.primary_quantity.identifier), rf"{equ.primary_quantity.identifier.
↪split('/')[-2]} astra"),
        (equ.primary_quantity.profile, rf"{equ.primary_quantity.identifier.split('/')[-2]} ")
    ],{"y_label": equ.primary_quantity.identifier.split("/")[-1]} ) for equ in solver_1d.equation
],
    x_axis=solver_1d.grid.rho_tor_norm
)
```



Part II

SpDM (SpDB) 数据集成和建模工具

CHAPTER

TEN

概述

Sp**DM** 的设计目标是从 Ontology 出发，将物理模型抽象为 Ontology 中的概念，将物理模型的输入输出抽象为 Ontology 中的属性，将物理模型的参数抽象为 Ontology 中的实例，将物理模型的计算过程抽象为 Ontology 中的关系，将物理模型的计算结果抽象为 Ontology 中的实例。通过将静态数据与具体功能函数/程序绑定，使得用户可以通过简单的配置，实现复杂的数据处理和物理建模。

SpDM 是一个自主开发的Python库，用于处理EAST数据分析中涉及的多种科学数据格式。它以MAS IDS为标准规范的本体，引入数据集成的思想将以不同语义和格式存储的数据在统一的数据模型下进行检索、查询。

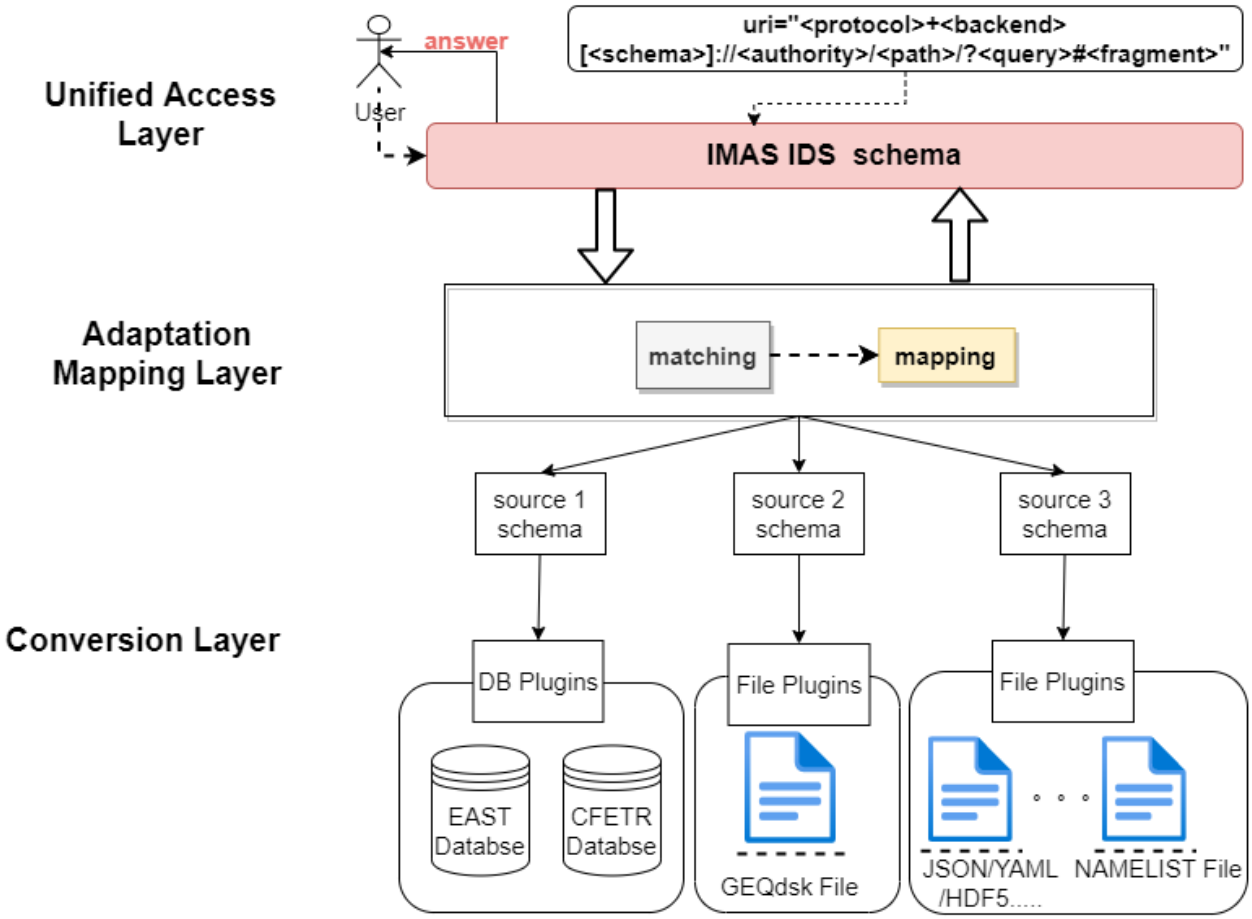
SpDM 专门为集成建模分析系统FyTok的数据交互而设计，但是不一定与之绑定，独立于 Fy**Tok** 仍然可以使用。**SpDM** 可以读取和处理常用的多种类型的科学数据格式，包括：Python原生的数据格式：字典、List；半结构化数据结构，如：Namelist、JSON、XML、HDF5、netCDF等；结构化数据结构，如：Gdskfile，、Inputfile等；远程数据库系统，如EAST实验的MDSplus及CFETR设计的MDSplus数据库等。Sp**DM** 将这些数据格式映射在以IMAS IDS为标准的数据交互语义，将数据绑定到 IMAS DD，以便在 IMAS DD 名称空间下统一异构多源数据。但是不依赖于IMAS ，不需要安装IMAS 。Sp**DM** 读取的数据在内存中以 Python 中数据格式进行交互，其数据的语义是严格遵守 IMAS IDS 的树状结构。Sp**DM** 也可以将数据写入到常用的多种类型的科学数据格式。

SpDM 可以为用户提供：

- 以IMAS IDS为标准的数据交互语义，但是不依赖于IMAS ，不需要安装IMAS 。
- 独立的python包，易安装、上手。可独立于ShenMa集群运行，安装在任何有python3.10+的环境下。
- 囊括日常科研工作中常用数据格式的处理，且以插件形式管理，开发者用户易对其进行扩展。
- 读取的数据在内存中以Python中数据格式进行交互，其数据的语义是严格遵守IMAS IDS的树状结构。
- 写数据是灵活的。

10.1 设计思想

SpDM 是一个数据集成工具。它基于标准的数据模型，为用户提供一种全局的中介模式，将来自不同、异构的数据源集成到一个全局的地址空间，通过唯一的URI实现对数据的统一访问，即是数据集成研究的范畴。它的实现，包含三层构架：面向用户的统一访问层及底层的映射层和转化层。



10.2 SpDM的处理对象

SpDM 是一个通用的数据集成工具，意在将聚变研究中常用的数据格式都统一映射在标准的语义表述下，同时降低用户处理对不同格式的数据源的门槛。**SpDM** 可以处理常用的多种类型的科学数据格式，包括：

- Python原生的数据格式：字典、List。
  - 直接在内存中交互
- 半结构化数据结构，如：Namelist、JSON、XML、HDF5、netCDF等。
  - 按照半结构化数据的已有的树状路径查询
- 结构化数据结构，如：Gdskfile，、Inputfile等。
  - 数据量比较小，拿回来全部放在内存中，直接访问。
- 远程数据库系统，如EAST实验的MDSplus及CFETR设计的MDSplus数据库等。
  - 将原始的数据源映射在标准的树状结构的语义下。
  - 支持延迟执行
  - 支持不同类型数据的集成，统一的入口访问。
    - 静态装置描述数据

* 动态实验测量数据

CHAPTER

ELEVEN

SPDM安装

11.1 ShenMa集群

目前，ShenMa集群内（service108）服务器上SpDM模块是可用的。你能运行下面命令，加载其环境：

```
module load spdm/0.0.0-foss-2022b
```

另外，如果你想使用MDSplus 后端，需要load：

```
module load MDSplus-Python/7.131.5-gfbf-2022b-Python-3.10.8
```

测试安装：

```
python -c "import spdm; print(spdm.__version__)"
```

11.2 本地安装

依赖:Python 3.10+。

SpDM的发行包已上传在pip 仓库中，运行pip install直接安装:

```
pip install --upgrade pip
pip install spdm
```

或者，也可以下载打包好的whl源码包安装：

```
wget https://gitee.com/SimPla/SpDM/releases/tag/0.3.0-rc
pip install --upgrade pip
pip install spdm
```

如果pip安装指定了安装目录，需先添加安装目录到PYTHONPATH中。否者在默认的~/.local/lib/下面：

```
export PYTHONPATH=${INSTALLPATH}/site-packages:${PYTHONPATH}
```

测试你的安装：

```
cd ~
python -c "import spdm; print(spdm.__version__)"
```

CHAPTER

TWELVE

数据绑定

HTree 是一个类，用来进行数据绑定。它的实例化对象是一个层次化的树形结构，每个节点都是一个 HTree，HTree 对象的属性可以通过点号访问，也可以通过索引访问。

12.1 将Python中原生数据映射到 HTree 中

```
from spdm.data.HTree import HTree, List, Dict

class Data(Dict):
    pass

### 传递一个普通的字典结构给Data
data = Data(
    {
        "name": "Alice",
        "age": 25,
        "hobbies": ["reading", "painting", "yoga"],
        "address": [
            {"street": "123 Main St", "city": "Anytown", "state": "CA", "zip": "12345"},
            {
                "street": "456 Oak St",
                "city": "Othertown",
                "state": "NY",
                "zip": "67890",
            },
            {"street": "789 Elm St", "city": "Somewhere", "state": "CO", "zip": "24680"},
        ],
    },
    # _entry=open_entry(....)
)
```

```
## 访问字典中的数据
data.get("name")
```

```
'Alice'
```

```
data.get("address")
```

```
[{'street': '123 Main St', 'city': 'Anytown', 'state': 'CA', 'zip': '12345'},
 {'street': '456 Oak St', 'city': 'Othertown', 'state': 'NY', 'zip': '67890'},
 {'street': '789 Elm St', 'city': 'Somewhere', 'state': 'CO', 'zip': '24680'}]
```

```
### 但是address不支持直接切片访问。
data.get("address[1]")
```

```
<tags.not_found: 0>
```

```
### address只能以整体拿回来，然后再切片访问
data.get("address")[1]
```

```
{'street': '456 Oak St', 'city': 'Othertown', 'state': 'NY', 'zip': '67890'}
```

12.2 sp_tree 装饰器

sp_tree 装饰器的作用是根据 class 中定义的 type_hint 生成 property，并于数据源对应 key 的 value 绑定，且在访问 property 时自动根据 type_hint 进行类型转换。

```
from spdm.data.sp_property import sp_tree, sp_property
from spdm.data.AoS import AoS
import pprint

### sp_tree装饰器支持在Data类中声明字典中的每个元素key及其value的数据类型，这样就可以直接访问字典中的元素。
@sp_tree
class Data:
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)

    name: str
    age: int
    hobbies: List[str]
    address: List[Dict]
    # address: AoS[Dict]
    # address: AoS[Addrees]


data = Data(
    {
        "name": "Alice",
        "age": '25',
        "hobbies": ["reading", "painting", "yoga"],
        "address": [
            {"street": "123 Main St", "city": "Anytown", "state": "CA", "zip": "12345"},
            {
                "street": "456 Oak St",
                "city": "Othertown",
                "state": "NY",
                "zip": "67890",
```

(continues on next page)

(continued from previous page)

```
            },
            {"street": "789 Elm St", "city": "Somewhere", "state": "CO", "zip": "24680"},
        ],
    },
    # _entry=open_entry(....)
)
```

```
## 发现声明过的元素其颜色都是蓝色的
data.name
```

```
'Alice'
```

```
### 除了声明元素的key值，同时可以强制转化原有数据的数据类型
type(data.age)
```

```
int
```

```
###
data.address
```

```
[{'street': '123 Main St', 'city': 'Anytown', 'state': 'CA', 'zip': '12345'},
 {'street': '456 Oak St', 'city': 'Othertown', 'state': 'NY', 'zip': '67890'},
 {'street': '789 Elm St', 'city': 'Somewhere', 'state': 'CO', 'zip': '24680'}]
```

```
## zip颜色是灰色，因为Dict里面的元素没有一一声明，
data.address[0].zip
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/scratch/jupytertest/workspace_fytok/fytok_tutorial/tutorial/BO6_sp_property_old.ipynb Cell 14␣
 ↪line 1
----> <a href='vscode-notebook-cell://ssh-remote%2Bjupytertest-service108-direct/scratch/
 ↪jupytertest/workspace_fytok/fytok_tutorial/tutorial/BO6_sp_property_old.ipynb
 ↪#X52sdnNjb2RlLXJlbW9OZQ%3D%3D?line=0'>1</a> data.address[0].zip

AttributeError: 'Dict' object has no attribute 'zip'
```

```
### 支持对LIST元素的直接切片访问，而不需要拿回整个字典
data.address[0]._cache
```

```
{'street': '123 Main St', 'city': 'Anytown', 'state': 'CA', 'zip': '12345'}
```

12.3 AoS 结构体数组

```
### 增加Address类，定义address的数据类型

@sp_tree
class Addrees:
    street: str
    city: str
    state: str
    zip: int


@sp_tree
class Data:
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)

    name: str
    age: int
    hobbies: List[str]
    address: AoS[Addrees]


data = Data(
    {
        "name": "Alice",
        "age": '25',
        "hobbies": ["reading", "painting", "yoga"],
        "address": [
            {"street": "123 Main St", "city": "Anytown", "state": "CA", "zip": "12345"},
            {
                "street": "456 Oak St",
                "city": "Othertown",
                "state": "NY",
                "zip": "67890",
            },
            {"street": "789 Elm St", "city": "Somewhere", "state": "CO", "zip": "24680"},
        ],
    },
    # _entry=open_entry(....)
)
```

```
### zip的颜色变成了蓝色，因为在Addrees类中声明了zip
data.address[0].zip
```

```
12345
```

```
data.address[1].street
```

```
'456 Oak St'
```

```
data.address[0]._cache
```

```
{'street': '123 Main St', 'city': 'Anytown', 'state': 'CA', 'zip': 12345}
```

12.4 访问属性

```
@sp_tree
class Addrees:
    street: str
    city: str
    state: str
    zip: int
    building: str = "#12345" # default value
    neighbour: str


@sp_tree
class Data:
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)

    name: str
    age: int
    hobbies: List[str]
    address: AoS[Addrees]


data = Data(
    {
        "name": "Alice",
        "age": 25,
        "hobbies": ["reading", "painting", "yoga"],
        "address": [
            {"street": "123 Main St", "city": "Anytown", "state": "CA", "zip": "12345"},
            {
                "street": "456 Oak St",
                "city": "Othertown",
                "state": "NY",
                "zip": "67890",
            },
            {"street": "789 Elm St", "city": "Somewhere", "state": "CO", "zip": "24680"},
        ],
    },
    # _entry=open_entry(....)
)
```

```
data.address[0].building
```

```
'#12345'
```

```
data.address[0].neighbour="zhangshan"
```

```
### 应用才会生效
data.address[1].building
```

```
'#12345'
```

```
data.address[0].neighbour
```

```
'zhangshan'
```

```
data.address[1].neighbour = "lisi"
```

```
data.address[0]._cache
```

```
{'street': '123 Main St',
 'city': 'Anytown',
 'state': 'CA',
 'zip': '12345',
 'building': '#12345',
 'neighbour': 'zhangshan'}
```

```
data.address[1].neighbour
```

```
{'street': '456 Oak St',
 'city': 'Othertown',
 'state': 'NY',
 'zip': '67890',
 'building': '#12345',
 'neighbour': 'lisi'}
```

12.4.1 增加函数sp_property，定义函数

```
@sp_tree
class House:
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)
        pprint.pprint((args, kwargs))

    level: int = 4
    length: float
    width: float

    @sp_property(units="m^2")
    def area(self) -> float:
        return self.width * self.length


@sp_tree
class Addrees:
```

(continues on next page)

(continued from previous page)

```
    street: str
    city: str
    state: str
    zip: int
    building: str = "#12345"
    neighbour: str
    house: House = sp_property(label="big house", default_value={"level": 2, "area": 1000})


@sp_tree
class Data:
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)

    name: str
    age: int
    hobbies: List[str]
    address: AoS[Addrees]


data = Data(
    {
        "name": "Alice",
        "age": 25,
        "hobbies": ["reading", "painting", "yoga"],
        "address": [
            {"street": "123 Main St", "city": "Anytown", "state": "CA", "zip": "12345"},
            {
                "street": "456 Oak St",
                "city": "Othertown",
                "state": "NY",
                "zip": "67890",
                "house": {"length": 5, "width": 10},
            },
            {"street": "789 Elm St", "city": "Somewhere", "state": "CO", "zip": "24680"},
        ],
    },
    # _entry=open_entry(....)
)
```

```
house=data.address[1].house
```

```
(({'length': 5, 'width': 10},),
 {'_entry': None,
  '_parent': <__main__.Addrees object at 0x7efc7ea7f760>,
  'default_value': {'area': 1000, 'level': 2},
  'label': 'big house',
  'name': 'house'})
```

```
house.level
```

```
4
```

```
house.length
```

```
5.0
```

```
house.width
```

```
10.0
```

```
house.area
```

```
50.0
```

```
House.area.metadata.get("units")
```

```
'm^2'
```

```
data.get("address/3/street", "I don't know")
```

```
"I don't know"
```

```
for address in data.address:
    print(address.street)
```

```
123 Main St
456 Oak St
789 Elm St
tags.not_found
```

```
data.address[0].street = "456 Main St"
```

```
data.address[0].street
```

```
'456 Main St'
```

CHAPTER

THIRTEEN

统一数据入口 ENTRY

Entry 是数据访问的统一入口，其指向树状结构中任意一个节点，并可对树结构遍历。

13.1 对 Python 原生数据的访问

```
from spdm.data.Entry import Entry
### 接收字典
example_dict = {
    'name': 'Alice',
    'age': 25,
    'hobbies': ['reading', 'painting', 'yoga'],
    'address': [
        {
            'street': '123 Main St',
            'city': 'Anytown',
            'state': 'CA',
            'zip': '12345'
        },
        {
            'street': '456 Oak St',
            'city': 'Othertown',
            'state': 'NY',
            'zip': '67890'
        },
        {
            'street': '789 Elm St',
            'city': 'Somewhere',
            'state': 'CO',
            'zip': '24680'
        }
    ],
    'spouse': {
        'spouse_name':
            {
                'name':'Bob',
                'age': 27,
            },
        'name': 'Bob',
        'age': 27,
        'hobbies': ['music', 'skiing', 'reading']
    }
}
```

(continues on next page)

(continued from previous page)

```
### example_dict作为参数传入Entry，生成Entry对象
dict_entry = Entry(example_dict)
### dict_entry是一个Entry对象，可以使用Entry的方法。如.get()方法，获取Entry对象的值
dict_entry.get()
```

```
{'name': 'Alice',
 'age': 25,
 'hobbies': ['reading', 'painting', 'yoga'],
 'address': [{'street': '123 Main St',
   'city': 'Anytown',
   'state': 'CA',
   'zip': '12345'},
  {'street': '456 Oak St', 'city': 'Othertown', 'state': 'NY', 'zip': '67890'},
  {'street': '789 Elm St',
   'city': 'Somewhere',
   'state': 'CO',
   'zip': '24680'}],
 'spouse': {'spouse_name': {'name': 'Bob', 'age': 27},
  'name': 'Bob',
  'age': 27,
  'hobbies': ['music', 'skiing', 'reading']}}
```

```
### 接收list
my_list = [
    [1, 2, 3, 4, 5],
    [6, 7, 8, 9, 10],
    [11, 12, 13],
    [14, 15, 16, 17],
    [18, 19, 20, 21, 22, 23, 24]
]
### my_list作为参数传入Entry，生成Entry对象
list_entry = Entry(my_list)
list_entry.get()
```

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

13.2 基本数据操作

- get()：读取数据
- put()：写入数据
- keys()：获得关键节点
- count()：计算当前节点的元数个数

13.2.1 .get()方法

- get(path): 读取数据,返回数据对象的值.
  - path是空的时候,返回当前根节点的数据对象
  - path参数是一个路径,路径可以是字符串,也可以是一个包含list索引的字符串的组合。
  - 相当于.child()+.fetch()操作

```
### 不指定path，返回整个节点的值
dict_entry.get()
```

```
{'name': 'Alice',
 'age': 25,
 'hobbies': ['reading', 'painting', 'yoga'],
 'address': [{'street': '123 Main St',
   'city': 'Anytown',
   'state': 'CA',
   'zip': '12345'},
  {'street': '456 Oak St', 'city': 'Othertown', 'state': 'NY', 'zip': '67890'},
  {'street': '789 Elm St',
   'city': 'Somewhere',
   'state': 'CO',
   'zip': '24680'}],
 'spouse': {'spouse_name': {'name': 'Bob', 'age': 27},
  'name': 'Bob',
  'age': 27,
  'hobbies': ['music', 'skiing', 'reading']}}
```

```
### 指定path，path表示方法1：'key1.key2.key3'。
### address是dict中的一个list，可以使用索引获取list中的元素，如address[0]，address[1]
dict_entry.get('address[0].street')
```

```
'123 Main St'
```

```
### 指定path，path表示方法2：['key1/key2/key3']。
dict_entry.get('address/0/street')
```

```
'123 Main St'
```

13.2.2 .put()方法

- put(key:value): 写入数据，返回Entry path对象.
  - key是一个路径,路径可以是字符串,也可以是一个包含list索引的字符串的组合。
  - 输出的path是一个Entry对象，可以继续进行get等操作。
  - 相当于.child()+.update()操作

```
### 根节点中增加height节点，值为170
dict_entry.put('height','170')
```

```
<spdm.data.Entry.Entry at 0x7fbdf051b3d0>
```

```
### 获取新增加的height节点的值
dict_entry.get('height')
```

```
'170'
```

```
### 给address[0]节点增加元素
dict_entry.put('address[0].phone','17756014979')
```

```
<spdm.data.Entry.Entry at 0x7fbdf051b640>
```

```
### 检查iphone节点是否增加成功
dict_entry.get('address[0]')
```

```
{'street': '123 Main St',
 'city': 'Anytown',
 'state': 'CA',
 'zip': '12345',
 'phone': '17756014979'}
```

```
### 增加新的address list节点的元数
### **"Not implemented "**
dict_entry.put('address[3]',{'street': '901 Elm St','city': 'Somewhere','state': 'CN','zip': '25780'})
```

```
dict_entry.get('address[2]')
```

```
{'street': '901 Elm St', 'city': 'Somewhere', 'state': 'CN', 'zip': '25780'}
```

13.2.3 .keys()方法

- keys():获得节点关键字

```
[*dict_entry.keys()]
```

```
['name', 'age', 'hobbies', 'address', 'spouse', 'height']
```

13.2.4 .count方法

- count:计算当前节点子节点个数

```
dict_entry.count
```

```
{'name': 'Alice',
 'age': 25,
 'hobbies': ['reading', 'painting', 'yoga'],
 'address': [{'street': '123 Main St',
   'city': 'Anytown',
   'state': 'CA',
   'zip': '12345'},
  {'street': '456 Oak St', 'city': 'Othertown', 'state': 'NY', 'zip': '67890'},
  {'street': '789 Elm St',
   'city': 'Somewhere',
   'state': 'CO',
   'zip': '24680'}],
 'spouse': {'spouse_name': {'name': 'Bob', 'age': 27},
  'name': 'Bob',
  'age': 27,
  'hobbies': ['music', 'skiing', 'reading']}}
```

```
### 计算当前节点的元数个数
print(f'address的元数个数是',dict_entry.child("address").count)
print(f'子节点spouse.spouse_name的元数个数是',dict_entry.child("spouse.spouse_name").count)
```

```
address的元数个数是 [{'street': '123 Main St', 'city': 'Anytown', 'state': 'CA', 'zip': '12345'}, {
↪'street': '456 Oak St', 'city': 'Othertown', 'state': 'NY', 'zip': '67890'}, {'street': '789 Elm
↪St', 'city': 'Somewhere', 'state': 'CO', 'zip': '24680'}]
子节点spouse.spouse_name的元数个数是 {'name': 'Bob', 'age': 27}
```

13.3 树遍历

- child(path):将指标移动到目标子节点，返回是个entry
- paraent(path):将指标移动到目标父节点，返回是个entry
- next():将指标移动到目标节点内元素的兄弟，返回是个entry
- 该功能只对元素是List有效

```
### 指标游走到当前节点"spouse.spouse_name"
child_data = dict_entry.child("spouse.spouse_name")
child_data.fetch()
```

```
{'name': 'Bob', 'age': 27}
```

```
### 指标移到当前节点的paraent节点，并获取数据
paraent_tree = child_data.parent
paraent_tree.fetch()
```

```
{'spouse_name': {'name': 'Bob', 'age': 27},
 'name': 'Bob',
 'age': 27,
 'hobbies': ['music', 'skiing', 'reading']}
```

```
dict_entry.child("address[0]").parent.fetch()
```

```
[{'street': '123 Main St', 'city': 'Anytown', 'state': 'CA', 'zip': '12345'},
 {'street': '456 Oak St', 'city': 'Othertown', 'state': 'NY', 'zip': '67890'},
 {'street': '789 Elm St', 'city': 'Somewhere', 'state': 'CO', 'zip': '24680'}]
```

```
### 当当前节点的元素是LIST时，指标可以移动遍历到其节点内所有value
print(dict_entry.child("address[0]").next().fetch())
print(dict_entry.child("address[1]").next().fetch())
```

```
{'street': '456 Oak St', 'city': 'Othertown', 'state': 'NY', 'zip': '67890'}
{'street': '789 Elm St', 'city': 'Somewhere', 'state': 'CO', 'zip': '24680'}
```

13.3.1 补充操作

灵活支持数据读写和更新：

- fetch(): 获取指标所指向的节点的数据，返回是对应的数值
- insert(value):在指标所指向的数据中插入新的数据，在原有数据基础上增加新的。
- update(value):在指标所指向的数据中更新数据，修改原有的数值。
- for_each():在指标所指向的数据中遍历数据

```
### for_each用来遍历dict中delist
for v in dict_entry.child("address").for_each():
    print(v)
```

```
(0, {'street': '123 Main St', 'city': 'Anytown', 'state': 'CA', 'zip': '12345'})
(1, {'street': '456 Oak St', 'city': 'Othertown', 'state': 'NY', 'zip': '67890'})
(2, {'street': '789 Elm St', 'city': 'Somewhere', 'state': 'CO', 'zip': '24680'})
```

```
### insert插入增加一个值，使之前的数变成list
dict_entry.child("age").insert(50)
dict_entry.child("age").fetch()
```

```
25
```

```
dict_entry.child("name").insert({"first":"Alice","last":"Smith"})
## 注意，插入更多数据后，原来的数据被组成了LIST
dict_entry.child("name").fetch()
```

```
['Alice', {'first': 'Alice', 'last': 'Smith'}]
```

```
### update是更新之前的整个数值
print(f'原来的age是',dict_entry.child("age").fetch())
dict_entry.child("age").update(100)
print(f'现在的age是',dict_entry.child("age").fetch())
# dict_entry.child("age").get()
```

```
原来的age是 90
现在的age是 100
```

13.4 统一URI访问 open_entry

在磁约束聚变研究中，科学数据是重要的处理对象。按照普遍的数据科学中分类，大体上被分为

- 结构化数据：如大型的MDSplus数据库
- 半结构化数据：如NameList,JSON,netCDF,HDF5，XML,YAML等
- 非结构化数据：如GDSKfile,InPut文件

每种数据都有其特定的读取、处理方式，SpDM通过对底层Entry功能封装，提供统一的Open_Entry标准API来访问、处理这些数据，使得用户可以通过同一种访问入口读取不同类型的数据。

`open_entry` 中支持标准的URL语法来访问不同的数据。用户无需关心后台的数据格式及来源。

13.4.1 标准的URI表述

URI遵守标准的RFC3986的语法，在其基础上进行扩展。

```
<schema>+<protocol>+<backend>://< authority >/<path>/?<query>#<fragment>
```

- <schema> 则指定了访问资源的“语义”。
- <protocol>指定了资源的协议，如远程的MDSplus或者本地的数据文件格式
- <backend>：指定了访问资源的“格式；
- <authority> 组件指定了管理资源的权限，通常包括主机通常包括主机名和端口号。
- <path>组件指定资源的分层路径，可包括多个层级，每个层级之间用资源的分层路径，可包括多个层级，以斜线（/）分隔。
- <query> 组件指定随资源请求发送到服务器的查询字符串
- <fragment> 组件指定指向特定资源的片段标识符

13.5 本地文件

13.5.1 针对文件的URL表达式：

```
file_entry = open_entry(f"file+format://{file_path}}")
```

- open_entry() :API入口
- file：处理对象是文件
- format：指定文件格式
- file_path：文件路径

13.5.2 处理gdskfile文件

```
### 初始化环境，import open_entry
from spdm.data.Entry import open_entry

### 指定输入文件的目录workdir
workdir = "/scratch/jupytertest/workspace_fytok/fytok_tutorial/tutorial/"

### 使用open_entry打开一个gdskfile文件
file_entry = open_entry(f"file+geqdsk://{workdir}/data/g070754.05000/")


### 指定子节点的标识符
# file_entry = open_entry(f"file+geqdsk://{workdir}//data/g900003.00230_ITER_15MA_eqdsk16HR.txt/
↪#equilibrium")
```

```
## file_entry是一个Entry对象,BO2节介绍的针对entry的对象的操作均可使用
## 获得该entry树的所有节点的标识符
[*file_entry.keys()]
```

```
['wall', 'equilibrium']
```

```
## 指标移到wall节点，获得数据
file_entry.child("wall").fetch()
```

```
{'description_2d': [{'limiter': {'unit': [{'outline': {'r': array([1.35838, 1.35838, 1.35838, 1.
↪36314, 1.4371 , 1.43721, 1.44164,
              1.4297 , 1.39169, 1.39151, 1.37929, 1.399  , 1.43623, 1.45919,
              1.47791, 1.54494, 1.61204, 1.63506, 1.66054, 1.66519, 1.6955 ,
              1.70668, 1.71388, 1.714  , 1.73649, 1.76324, 1.80199, 1.80237,
              1.93972, 1.97063, 2.07495, 2.1771 , 2.27925, 2.35   , 2.35   ,
              2.35   , 2.27925, 2.1771 , 2.07495, 1.9728 , 1.97063, 1.92059,
              1.87055, 1.82051, 1.79281, 1.76511, 1.75421, 1.67782, 1.60143,
              1.48426, 1.36709, 1.33132, 1.37874, 1.42615, 1.40528, 1.3844 ,
              1.36353, 1.35838, 1.35838, 1.35838, 1.35838]),
       'z': array([ 0.      ,  0.227   ,  0.454   ,  0.455735,  0.798332,  0.798821,
              0.838195,  0.908908,  1.01426 ,  1.01473 ,  1.04859 ,  1.03946 ,
```

(continues on next page)

(continued from previous page)

```
                1.0256   ,  1.02095 ,  1.02514 ,  1.05454 ,  1.08394 ,  1.09586 ,
                1.114    ,  1.11765 ,  1.14183 ,  1.16211 ,  1.13093 ,  1.13044 ,
                1.033    ,  0.969832,  0.92567 ,  0.925347,  0.809223,  0.783389,
                0.68838 ,  0.58904 ,  0.4897  ,  0.49    ,  0.      , -0.49    ,
               -0.4897   , -0.58904 , -0.68838 , -0.78772 , -0.783389, -0.832375,
               -0.881361, -0.930347, -1.05034 , -1.17034 , -1.13917 , -1.03465 ,
               -0.930138, -0.95338 , -0.976622, -1.01072 , -0.879327, -0.747939,
               -0.65062 , -0.5533  , -0.455981, -0.454   , -0.227   ,  0.      ,
                0.      ])}}}]}}]}
```

```
### 指定子节点的标识符
file_entry_eq = open_entry(f"file+geqdsk://{workdir}//data/g900003.00230_ITER_15MA_eqdsk16HR.txt/
↪#equilibrium")
```

```
### equilibrium节点的子节点
[*file_entry_eq.keys()]
```

```
['time', 'vacuum_toroidal_field', 'time_slice']
```

13.5.3 HDF5 文件

```
hdf5_entry = open_entry(f"file+hdf5://{workdir}/data/test_eq.h5")
```

```
2023-11-16 21:05:44,677 [    spdm]    DEBUG: /gpfs/fuyun/projects/fuyun/spdm/python/spdm/plugins/
↪data/plugin_hdf5.py:196:open: Open HDF5 File /scratch/jupytertest/workspace_fytok/fytok_tutorial/
↪tutorial//data/test_eq.h5 mode=Mode.read
```

```
[*hdf5_entry.keys()]
```

```
['equilibrium', 'wall']
```

```
hdf5_entry.child("equilibrium/time_slice/0/boundary").fetch()
```

```
{'geometric_axis': {'r': 1.85000002, 'z': 0.0},
 'outline': {'r': array([1.41273642, 1.41389295, 1.41822664, 1.425     , 1.43209863,
        1.43980543, 1.45      , 1.4625    , 1.475     , 1.4875    ,
        1.5       , 1.5125    , 1.525     , 1.54090029, 1.55857183,
        1.575     , 1.5875    , 1.60822115, 1.625     , 1.64220094,
        1.675     , 1.7       , 1.72689129, 1.7625    , 1.7875    ,
        1.8125    , 1.8375    , 1.86378618, 1.893304  , 1.92133389,
        1.94791747, 1.9731425 , 1.99710604, 2.01989722, 2.0415908 ,
        2.0625    , 2.0875    , 2.1125    , 2.13503267, 2.15095696,
        2.175     , 2.19360357, 2.2125    , 2.22883407, 2.24833148,
        2.2625    , 2.275     , 2.2875    , 2.29561793, 2.3       ,
        2.30052821, 2.29718556, 2.2875    , 2.275     , 2.2625    ,
        2.25      , 2.23203494, 2.2125    , 2.19533868, 2.175     ,
```

(continues on next page)

(continued from previous page)

```
        2.15      , 2.13201545, 2.1125    , 2.0875    , 2.0625    ,
        2.0375    , 2.0125    , 1.9875    , 1.9625    , 1.9353857 ,
        1.9       , 1.875     , 1.85      , 1.8125    , 1.7875    ,
        1.75      , 1.7125    , 1.6875    , 1.65      , 1.625     ,
        1.6       , 1.57788566, 1.5625    , 1.5375    , 1.525     ,
        1.5049414 , 1.4875    , 1.475     , 1.4625    , 1.45      ,
        1.43769196, 1.42957712, 1.42310696, 1.41633893, 1.41307553,
        1.41273642]),
 'z': array([ 0.        ,  0.065625  ,  0.13125   ,  0.19194756,  0.240625  ,
         0.284375  ,  0.33423411,  0.38775715,  0.43549313,  0.47904169,
         0.5193792 ,  0.55706639,  0.59230534,  0.634375  ,  0.678125  ,
         0.71687204,  0.74420412,  0.7875    ,  0.82429578,  0.85525714,
         0.83954523,  0.82526535,  0.809375  ,  0.78869278,  0.77356616,
         0.75746309,  0.74052226,  0.721875  ,  0.7       ,  0.678125  ,
         0.65625   ,  0.634375  ,  0.6125    ,  0.590625  ,  0.56875   ,
         0.54659888,  0.51855995,  0.48849663,  0.459375  ,  0.4375    ,
         0.40193967,  0.371875  ,  0.33829558,  0.30625   ,  0.2625    ,
         0.22505378,  0.1856974 ,  0.13510461,  0.0875    ,  0.03997558,
        -0.021875  , -0.065625  , -0.12525664, -0.17499408, -0.2128994 ,
        -0.24493408, -0.284375  , -0.32114872, -0.35      , -0.38083695,
        -0.4150113 , -0.4375    , -0.46029528, -0.48734772, -0.51226288,
        -0.53529371, -0.55661962, -0.57635108, -0.59453317, -0.6125    ,
        -0.63356282, -0.64675334, -0.65804837, -0.6720044 , -0.67853898,
        -0.68429772, -0.68302709, -0.67729193, -0.66033755, -0.6422052 ,
        -0.61782529, -0.590625  , -0.56865804, -0.52684039, -0.50279392,
        -0.459375  , -0.41579306, -0.38041418, -0.34056526, -0.29466146,
        -0.240625  , -0.196875  , -0.153125  , -0.0875    , -0.021875  ,
         0.        ])}}
```

```
## 或者使用. 来表示字符串
hdf5_entry.child("equilibrium.time_slice[0].boundary.geometric_axis").fetch()
```

```
{'r': 1.85000002, 'z': 0.0}
```

13.5.4 NetCDF 文件

```
nc_entry = open_entry(f"file+hdf5://{workdir}/data/test.nc")
```

```
2023-11-16 21:10:30,889 [    spdm]    DEBUG: /gpfs/fuyun/projects/fuyun/spdm/python/spdm/plugins/
↪data/plugin_hdf5.py:196:open: Open HDF5 File /scratch/jupytertest/workspace_fytok/fytok_tutorial/
↪tutorial//data/test.nc mode=Mode.read
```

```
[*nc_entry.keys()]
```

```
['coherent_wave']
```

```
nc_entry.child("coherent_wave").fetch()
```

```
{'0': {'profiles_1d': {'0': {'power_density': array([0.        , 0.        , 0.        , 0.
↪, 0.99999802,
          0.99999918, 0.99999994, 1.00000037, 1.00000072, 1.00000098,
          1.0000012 , 1.00000132, 1.00000145, 1.00000163, 1.00000158,
          1.00000156, 1.00000155, 1.00000155, 1.00000154, 1.00000155,
          1.00000153, 1.00000149, 1.00000145, 1.00000139, 1.00000134,
          1.00000129, 1.00000124, 1.00000125, 1.00000131, 1.00000145,
          1.00000149, 1.0000015 , 1.00000144, 1.00000136, 1.00000122,
          1.0000011 , 1.00000096, 1.00000085, 1.00000083, 1.00000088,
          1.00000105, 1.00000128, 1.0000014 , 1.00000142, 1.00000148,
          1.0000015 , 1.0000015 , 1.00000145, 1.0000012 , 1.00000149]),
   'current_parallel_density': array([-0.00000000e+00, -0.00000000e+00, -0.00000000e+00, -0.
↪00000000e+00,
           1.99364358e-13,  2.14821878e-08,  3.93642769e-05,  2.86688349e-03,
           9.60013670e-02,  1.23434308e+00,  1.03052517e+01,  3.40975907e+01,
           1.27767832e+02,  8.03338628e+02,  4.84207827e+02,  3.60250285e+02,
           2.84093429e+02,  2.66870953e+02,  2.27307283e+02,  2.29816800e+02,
           1.81405673e+02,  1.20656896e+02,  8.05569657e+01,  4.50482215e+01,
           2.53651230e+01,  1.52108096e+01,  7.64395581e+00,  7.25623510e+00,
           1.17114376e+01,  4.32715441e+01,  6.48528120e+01,  7.52389104e+01,
           4.14450706e+01,  1.96846041e+01,  4.89572505e+00,  1.43632231e+00,
           3.62878642e-01,  6.95244839e-02,  1.75151033e-02, -2.21662387e-02,
          -3.89583106e-01, -3.95056278e+00, -1.38417137e+01, -1.74193858e+01,
          -3.01090468e+01, -3.86729141e+01, -4.03153212e+01, -2.48381992e+01,
          -1.19357956e+00, -3.86867567e-01]),
   'grid': {'rho_tor_norm': array([0.01, 0.03, 0.05, 0.07, 0.09, 0.11, 0.13, 0.15, 0.17, 0.19, 0.
↪21,
           0.23, 0.25, 0.27, 0.29, 0.31, 0.33, 0.35, 0.37, 0.39, 0.41, 0.43,
           0.45, 0.47, 0.49, 0.51, 0.53, 0.55, 0.57, 0.59, 0.61, 0.63, 0.65,
           0.67, 0.69, 0.71, 0.73, 0.75, 0.77, 0.79, 0.81, 0.83, 0.85, 0.87,
           0.89, 0.91, 0.93, 0.95, 0.97, 0.99])},
   'electrons': {'power_density_thermal': array([0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
↪0.00000000e+00,
           2.47389160e-16, 2.76673170e-11, 5.32673700e-08, 4.03489236e-06,
           1.38610968e-04, 1.84668042e-03, 1.61529270e-02, 5.60366441e-02,
           2.02266735e-01, 1.14922075e+00, 7.58349000e-01, 6.15352146e-01,
           5.20135233e-01, 5.42653550e-01, 4.89433633e-01, 5.31448764e-01,
           4.30602818e-01, 2.90238020e-01, 1.97551038e-01, 1.13034650e-01,
           6.64365651e-02, 4.17000387e-02, 2.33857989e-02, 2.59507709e-02,
           5.11777186e-02, 2.04015882e-01, 3.02458440e-01, 3.40749100e-01,
           1.83455844e-01, 8.28459551e-02, 1.98170713e-02, 5.64636162e-03,
           1.33097798e-03, 2.43573510e-04, 1.11680512e-04, 2.92732068e-04,
           3.38213156e-03, 3.54961697e-02, 1.23920281e-01, 1.53797565e-01,
           2.62316497e-01, 3.26007311e-01, 3.21541597e-01, 1.92326851e-01,
           1.34692247e-02, 9.24638613e-03])}}}}}
```

```
[*file_entry.keys()]
```

```
['wall', 'equilibrium']
```

```
### 访问远程EAST MDS数据库中70754炮的数据，
shot_num = 70754
time_slice = 10
entry = open_entry("east+mdsplus://202.127.204.12?enable=efit_east&shot={shot_num}")
```

```
[*file_entry.keys()]
```

```
['time', 'vacuum_toroidal_field', 'time_slice']
```

```
file_entry.get("vacuum_toroidal_field")
```

```
{'r0': 6.2, 'b0': [-5.3]}
```

```
file_entry.child("vacuum_toroidal_field").fetch()
```

```
{'r0': 6.2, 'b0': [-5.3]}
```

13.6 远程数据库

Note:

注:MDSplus数据库的访问，仅在等离子所内网有效。

在 **SpDM** 中，使用open_entry()建立统一的访问入口,支持远程MDSplus Server数据库的访问，及本地MDSplus的数据库中数据的访问。支持两个有用的功能：

（1）**不同数据源的自动集成**

SpDM中提供 **open_entry** 建立访问链接，**wall**，**pf**，**tf**，**magnetics** 等被映射的数据均可以通过该链接入口访问。数据源来自于静态的XML文件，动态的MDS数据库中的不同tree：east,pcs_east,efit_east等

SpDM中数据的访问方式是按照IDS的树状结构逐层访问。

（2）**针对大型的数据库系统，“指标游走，懒惰执行”发挥重要作用**

SpDM的Entry支持指标游走，懒惰加载数据功能。这对大型的数据库系统非常重要。SpDM后台已经自动集成了不同数据源的数据，对应于不同的IDS条目中。所请求的条目或者某个条目中可能存储了大量的数据，如果请求的时候便立即从底层访问层后端读取所有数据，可能需要很长时间才能完成。通常情况下，用户可能只需要个别的数据子集，”懒惰加载”使得用户仅仅建立链接，只有在需要的时候才真正获取数据，这样会利于加速。

- **.child(path)**：将链接指针移动到指定的树节点，建立新的链接。返回是新的path。
- **.get(path)**：获得给定path节点的全部数据。返回时具体数值。（get=child+fetch）

13.6.1 针对数据库系统的URI表达式：

访问远程数据库系统

```
entry = open_entry("device+mdsplus://202.127.204.12?enable=efit_east&shot={shot_num}")
```

- device：指定数据来源的装置
- mdsplus：处理对象是MDS数据库
- 202.127.204.12：EAST mds数据库访问地址
- enable：同时可以获取efit_east数据
- shot：指定炮号

访问本地的MDSplus数据库

```
...
entry_local = open_entry(f"east+mdsplus://{DATA_PATH}/mdsplus/~t/?enable=efit_east&shot=70745")
```

- device：指定数据来源的装置,如east
- mdsplus：处理对象是MDS数据库
- {DATA_PATH}/mdsplus/~t/ 本地数据库的目录
- enable：同时可以获取efit_east数据
- shot：指定炮号

加载基本环境

```
### import基本环境
from spdm.data.Entry import open_entry
from spdm.utils.logger import logger
import MDSplus
import os
```

```
### 指定mapping文件的路径
os.environ["SP_DATA_MAPPING_PATH"] = "/gpfs/fuyun/projects/fuyun/fytok/python/fytok/_mapping"
```

访问远程EAST MDSplus数据库

```
### 访问远程EAST MDS数据库中70754炮的数据，
shot_num = 70754
time_slice = 10
entry_reomte = open_entry("east+mdsplus://202.127.204.12?enable=efit_east&shot={shot_num}")
```

```
wall = entry_reomte.child("wall")
```

```
wall.get()
```

```
{'ids_properties': {'comment': {},
  'provider': 'Guo, Yong',
  'creation_date': '2020-10-12',
  'homogeneous_time': 2},
 'description_2d': {'type': {'name': 'equilibrium', 'index': 1},
  'limiter': {'type': {'name': 'limiter', 'index': 0},
   'unit': {'name': 'limiter',
    'closed': 1,
    'outline': {'r': array([2.277, 2.273, 2.267, 1.94 , 1.94 , 1.802, 1.773, 1.751, 1.736,
           1.714, 1.707, 1.696, 1.665, 1.656, 1.635, 1.612, 1.478, 1.459,
           1.44 , 1.436, 1.399, 1.379, 1.392, 1.43 , 1.439, 1.442, 1.437,
           1.363, 1.361, 1.361, 1.361, 1.363, 1.421, 1.423, 1.422, 1.418,
           1.331, 1.367, 1.564, 1.597, 1.598, 1.624, 1.754, 1.765, 1.814,
           1.824, 1.825, 1.841, 1.971, 1.971, 2.267, 2.273, 2.277, 2.277,
           2.306, 2.328, 2.343, 2.35 , 2.35 , 2.35 , 2.343, 2.328, 2.306,
           2.277]),
     'z': array([ 0.485,  0.485,  0.493,  0.809,  0.809,  0.926,  0.956,  0.993,
            1.033,  1.131,  1.162,  1.142,  1.117,  1.111,  1.096,  1.084,
            1.025,  1.021,  1.024,  1.026,  1.039,  1.049,  1.014,  0.909,
            0.873,  0.835,  0.799,  0.456,  0.454,  0.   , -0.454, -0.456,
           -0.725, -0.748, -0.749, -0.77 , -1.011, -0.977, -0.938, -0.941,
           -0.941, -0.961, -1.139, -1.17 , -0.959, -0.934, -0.932, -0.91 ,
           -0.783, -0.783, -0.493, -0.485, -0.485, -0.309, -0.244, -0.176,
           -0.106, -0.036,  0.   ,  0.036,  0.106,  0.176,  0.244,  0.309])},
    '@id': '0'}},
  'vessel': {'unit': {'name': 'EAST',
    'identifier': 'EAST',
    'annular': {'outline_inner': {'closed': 1,
      'r': array([2.7286, 2.7069, 2.664 , 2.6007, 2.5178, 2.417 , 2.2997, 2.1681,
            2.0244, 1.8708, 1.709 , 1.5435, 1.3902, 1.2814, 1.2402, 1.237 ,
            1.237 , 1.237 , 1.237 , 1.237 , 1.237 , 1.237 , 1.237 , 1.237 ,
            1.237 , 1.2402, 1.2814, 1.3902, 1.5435, 1.709 , 1.8708, 2.0244,
            2.1681, 2.2997, 2.417 , 2.5178, 2.6007, 2.664 , 2.7069, 2.7286]),
     'z': array([ 0.083289,  0.24846 ,  0.40942 ,  0.56347 ,  0.708   ,  0.84058 ,
            0.95892 ,  1.061   ,  1.1452  ,  1.2095  ,  1.2454  ,  1.2439  ,
            1.1907  ,  1.075   ,  0.91857 ,  0.75173 ,  0.58468 ,  0.41763 ,
            0.25058 ,  0.083526, -0.083526, -0.25058 , -0.41763 , -0.58468 ,
           -0.75173 , -0.91857 , -1.075   , -1.1907  , -1.2439  , -1.2454  ,
           -1.2095  , -1.1452  , -1.061   , -0.95892 , -0.84058 , -0.708   ,
           -0.56347 , -0.40942 , -0.24846 , -0.083289])},
     'outline_outer': {'closed': 1,
      'r': array([2.844 , 2.8204, 2.7737, 2.7046, 2.6143, 2.5045, 2.3771, 2.2343,
            2.0786, 1.9127, 1.7389, 1.5637, 1.3995, 1.2722, 1.2108, 1.199 ,
            1.199 , 1.199 , 1.199 , 1.199 , 1.199 , 1.199 , 1.199 , 1.199 ,
```

(continues on next page)

(continued from previous page)

```
            1.199 , 1.2108, 1.2722, 1.3995, 1.5637, 1.7389, 1.9127, 2.0786,
            2.2343, 2.3771, 2.5045, 2.6143, 2.7046, 2.7737, 2.8204, 2.844 ]),
     'z': array([ 0.089216,  0.26608 ,  0.43828 ,  0.6028  ,  0.75671 ,  0.89735 ,
             1.0223  ,  1.1292  ,  1.2164  ,  1.282   ,  1.3197  ,  1.3165  ,
             1.2605  ,  1.1444  ,  0.98288 ,  0.80532 ,  0.62636 ,  0.4474  ,
             0.26844 ,  0.08948 , -0.08948 , -0.26844 , -0.4474  , -0.62636 ,
            -0.80532 , -0.98288 , -1.1444  , -1.2605  , -1.3165  , -1.3197  ,
            -1.282   , -1.2164  , -1.1292  , -1.0223  , -0.89735 , -0.75671 ,
            -0.6028  , -0.43828 , -0.26608 , -0.089216])}},
   '@id': '0'}},
 '@id': '0'}}
```

访问远程本地MDSplus数据库

```
# ### 访问本地MDSplus路径中中的70754炮的数据
shot_num = 70754
time_slice = 10
DATA_PATH = "/scratch/jupytertest/workspace_fytok/fytok_data"
entry_local = open_entry(f"east+mdsplus://{DATA_PATH}/mdsplus/~t/?shot=70745")
```

open_entry建立了指向数据库系统的链接entry_local，并且自动集成了已经映射的wall，pf，tf，magnetics等数据。

entry_local作为访问这些数据的入口。自动继成Entry的一切针对指针和数据的操作。

```
### .child操作会将链接进一步指向wall
wall = entry_local.child("wall")
### 打印wall的类型，仍然是个entry
type(wall)
wall.get()
```

```
{'ids_properties': {'comment': {},
  'provider': 'Guo, Yong',
  'creation_date': '2020-10-12',
  'homogeneous_time': 2},
 'description_2d': {'type': {'name': 'equilibrium', 'index': 1},
  'limiter': {'type': {'name': 'limiter', 'index': 0},
   'unit': {'name': 'limiter',
    'closed': 1,
    'outline': {'r': array([2.277, 2.273, 2.267, 1.94 , 1.94 , 1.802, 1.773, 1.751, 1.736,
           1.714, 1.707, 1.696, 1.665, 1.656, 1.635, 1.612, 1.478, 1.459,
           1.44 , 1.436, 1.399, 1.379, 1.392, 1.43 , 1.439, 1.442, 1.437,
           1.363, 1.361, 1.361, 1.361, 1.363, 1.421, 1.423, 1.422, 1.418,
           1.331, 1.367, 1.564, 1.597, 1.598, 1.624, 1.754, 1.765, 1.814,
           1.824, 1.825, 1.841, 1.971, 1.971, 2.267, 2.273, 2.277, 2.277,
           2.306, 2.328, 2.343, 2.35 , 2.35 , 2.35 , 2.343, 2.328, 2.306,
           2.277]),
     'z': array([ 0.485,  0.485,  0.493,  0.809,  0.809,  0.926,  0.956,  0.993,
            1.033,  1.131,  1.162,  1.142,  1.117,  1.111,  1.096,  1.084,
            1.025,  1.021,  1.024,  1.026,  1.039,  1.049,  1.014,  0.909,
            0.873,  0.835,  0.799,  0.456,  0.454,  0.   , -0.454, -0.456,
           -0.725, -0.748, -0.749, -0.77 , -1.011, -0.977, -0.938, -0.941,
```

(continues on next page)

(continued from previous page)

```
           -0.941, -0.961, -1.139, -1.17 , -0.959, -0.934, -0.932, -0.91 ,
           -0.783, -0.783, -0.493, -0.485, -0.485, -0.309, -0.244, -0.176,
           -0.106, -0.036,  0.   ,  0.036,  0.106,  0.176,  0.244,  0.309])},
    '@id': '0'}},
  'vessel': {'unit': {'name': 'EAST',
    'identifier': 'EAST',
    'annular': {'outline_inner': {'closed': 1,
      'r': array([2.7286, 2.7069, 2.664 , 2.6007, 2.5178, 2.417 , 2.2997, 2.1681,
             2.0244, 1.8708, 1.709 , 1.5435, 1.3902, 1.2814, 1.2402, 1.237 ,
             1.237 , 1.237 , 1.237 , 1.237 , 1.237 , 1.237 , 1.237 , 1.237 ,
             1.237 , 1.2402, 1.2814, 1.3902, 1.5435, 1.709 , 1.8708, 2.0244,
             2.1681, 2.2997, 2.417 , 2.5178, 2.6007, 2.664 , 2.7069, 2.7286]),
      'z': array([ 0.083289,  0.24846 ,  0.40942 ,  0.56347 ,  0.708   ,  0.84058 ,
              0.95892 ,  1.061   ,  1.1452  ,  1.2095  ,  1.2454  ,  1.2439  ,
              1.1907  ,  1.075   ,  0.91857 ,  0.75173 ,  0.58468 ,  0.41763 ,
              0.25058 ,  0.083526, -0.083526, -0.25058 , -0.41763 , -0.58468 ,
             -0.75173 , -0.91857 , -1.075   , -1.1907  , -1.2439  , -1.2454  ,
             -1.2095  , -1.1452  , -1.061   , -0.95892 , -0.84058 , -0.708   ,
             -0.56347 , -0.40942 , -0.24846 , -0.083289])},
     'outline_outer': {'closed': 1,
      'r': array([2.844 , 2.8204, 2.7737, 2.7046, 2.6143, 2.5045, 2.3771, 2.2343,
             2.0786, 1.9127, 1.7389, 1.5637, 1.3995, 1.2722, 1.2108, 1.199 ,
             1.199 , 1.199 , 1.199 , 1.199 , 1.199 , 1.199 , 1.199 , 1.199 ,
             1.199 , 1.2108, 1.2722, 1.3995, 1.5637, 1.7389, 1.9127, 2.0786,
             2.2343, 2.3771, 2.5045, 2.6143, 2.7046, 2.7737, 2.8204, 2.844 ]),
      'z': array([ 0.089216,  0.26608 ,  0.43828 ,  0.6028  ,  0.75671 ,  0.89735 ,
              1.0223  ,  1.1292  ,  1.2164  ,  1.282   ,  1.3197  ,  1.3165  ,
              1.2605  ,  1.1444  ,  0.98288 ,  0.80532 ,  0.62636 ,  0.4474  ,
              0.26844 ,  0.08948 , -0.08948 , -0.26844 , -0.4474  , -0.62636 ,
             -0.80532 , -0.98288 , -1.1444  , -1.2605  , -1.3165  , -1.3197  ,
             -1.282   , -1.2164  , -1.1292  , -1.0223  , -0.89735 , -0.75671 ,
             -0.6028  , -0.43828 , -0.26608 , -0.089216])}},
    '@id': '0'}},
  '@id': '0'}}
```

```
### 支持按照连续字符串格式，继续移动链接到wall中的下一个子节点中,outline仍然是个entry
outline = entry_local.child("wall.description_2d[0].limiter.unit[0].outline")
### 获取outline的数据
print(outline.get())
```

```
{'r': array([2.277, 2.273, 2.267, 1.94 , 1.94 , 1.802, 1.773, 1.751, 1.736,
       1.714, 1.707, 1.696, 1.665, 1.656, 1.635, 1.612, 1.478, 1.459,
       1.44 , 1.436, 1.399, 1.379, 1.392, 1.43 , 1.439, 1.442, 1.437,
       1.363, 1.361, 1.361, 1.361, 1.363, 1.421, 1.423, 1.422, 1.418,
       1.331, 1.367, 1.564, 1.597, 1.598, 1.624, 1.754, 1.765, 1.814,
       1.824, 1.825, 1.841, 1.971, 1.971, 2.267, 2.273, 2.277, 2.277,
       2.306, 2.328, 2.343, 2.35 , 2.35 , 2.35 , 2.343, 2.328, 2.306,
       2.277]), 'z': array([ 0.485,  0.485,  0.493,  0.809,  0.809,  0.926,  0.956,  0.993,
        1.033,  1.131,  1.162,  1.142,  1.117,  1.111,  1.096,  1.084,
        1.025,  1.021,  1.024,  1.026,  1.039,  1.049,  1.014,  0.909,
        0.873,  0.835,  0.799,  0.456,  0.454,  0.   , -0.454, -0.456,
       -0.725, -0.748, -0.749, -0.77 , -1.011, -0.977, -0.938, -0.941,
       -0.941, -0.961, -1.139, -1.17 , -0.959, -0.934, -0.932, -0.91 ,
       -0.783, -0.783, -0.493, -0.485, -0.485, -0.309, -0.244, -0.176,
       -0.106, -0.036,  0.   ,  0.036,  0.106,  0.176,  0.244,  0.309])}
```

SpDM中数据以字典形式在内存中直接交互

```
[entry_local.get("wall").keys()]
```

```
[dict_keys(['ids_properties', 'description_2d'])]
```

SpDM中list的访问

```
### 例如，b_field_pol_probe探针有多个，以list形式存在
type(entry_local.get("magnetics")["b_field_pol_probe"])
```

```
list
```

```
### 每个探针下面是一个字典结构，获得其keys
entry_local.get("magnetics.b_field_pol_probe[0]").keys()
```

```
2023-11-15 14:01:08,583 [    spdm]    DEBUG: /gpfs/fuyun/projects/fuyun/spdm/python/spdm/plugins/
↪data/plugin_xml.py:274:_convert: ('*', ['magnetics', 'b_field_pol_probe', 0, 'position'])
```

```
dict_keys(['name', 'identifier', 'position', 'toroidal_angle', 'field', '@id'])
```

```
entry_local.child("magnetics.b_field_pol_probe[0].position").fetch()
```

```
2023-11-15 14:01:06,046 [    spdm]    DEBUG: /gpfs/fuyun/projects/fuyun/spdm/python/spdm/plugins/
↪data/plugin_xml.py:274:_convert: ('*', ['magnetics', 'b_field_pol_probe', 0, 'position'])
```

```
[{'r': 1.2905, 'z': 0.0, '@id': '*'}]
```

SpDM中数据按照树状结构层层访问

```
### SpDM中数据按照层层路径进行访问:wall.description_2d[0].limiter.unit.outlin
### 获取outline的数据:
entry_local.get("wall.description_2d[0].limiter.unit.outline")
```

```
{'r': array([2.277, 2.273, 2.267, 1.94 , 1.94 , 1.802, 1.773, 1.751, 1.736,
        1.714, 1.707, 1.696, 1.665, 1.656, 1.635, 1.612, 1.478, 1.459,
        1.44 , 1.436, 1.399, 1.379, 1.392, 1.43 , 1.439, 1.442, 1.437,
        1.363, 1.361, 1.361, 1.361, 1.363, 1.421, 1.423, 1.422, 1.418,
        1.331, 1.367, 1.564, 1.597, 1.598, 1.624, 1.754, 1.765, 1.814,
        1.824, 1.825, 1.841, 1.971, 1.971, 2.267, 2.273, 2.277, 2.277,
        2.306, 2.328, 2.343, 2.35 , 2.35 , 2.35 , 2.343, 2.328, 2.306,
        2.277]),
```

(continues on next page)

(continued from previous page)

```
'z': array([ 0.485,  0.485,  0.493,  0.809,  0.809,  0.926,  0.956,  0.993,
        1.033,  1.131,  1.162,  1.142,  1.117,  1.111,  1.096,  1.084,
        1.025,  1.021,  1.024,  1.026,  1.039,  1.049,  1.014,  0.909,
        0.873,  0.835,  0.799,  0.456,  0.454,  0.   , -0.454, -0.456,
       -0.725, -0.748, -0.749, -0.77 , -1.011, -0.977, -0.938, -0.941,
       -0.941, -0.961, -1.139, -1.17 , -0.959, -0.934, -0.932, -0.91 ,
       -0.783, -0.783, -0.493, -0.485, -0.485, -0.309, -0.244, -0.176,
       -0.106, -0.036,  0.   ,  0.036,  0.106,  0.176,  0.244,  0.309])}
```

CHAPTER

FOURTEEN

数据文件 FILE

spdm.data.File 用来处理非结构化及半结构化数据文件。

14.1 初始化环境

```
from spdm.data.File import File
from spdm.utils.logger import logger
from pathlib import Path
```

14.2 非结构化数据

```
with File("./data/g070754.05000", mode="r", format="GEQdsk") as fid:
    doc = fid.read()
    eq_test = doc.dump()
```

```
#### eq_test是Python中的字典，其key遵守IMAS IDS的组织结构
eq_test.keys()
```

```
dict_keys(['wall', 'equilibrium'])
```

```
#### 查看equilibrium中的数据
eq_test["equilibrium"].keys()
```

```
dict_keys(['time', 'vacuum_toroidal_field', 'time_slice'])
```

14.3 半结构化数据

14.3.1 NameList 文件

```
### 以GENRAY的输入文件genray.dat为例，它是一个NameList格式的文件。
with File(f"./data/genray.dat", format= "namelist" ,mode="r") as oid:
### 仅仅建立链接，而没有拿回数据
    output_entry = oid.read()
### 拿回所有数据
    output_data = oid.read().dump()
### 获取namelist中的关键字key
logger.info(output_data.keys())
### 通过child获取某个子节点的值，而不需要把整个树读回来
logger.info(output_entry.child("main_lobes").__value__)
```

```
2023-11-15 14:18:53,280 [    spdm]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_namelist
2023-11-15 14:18:53,356 [    spdm]     INFO: dict_keys(['main_lobes', 'genr', 'tokamak', 'wave',
↪'scatnper', 'dispers', 'numercl', 'output', 'plasma', 'species', 'varden', 'denprof', 'tpopprof',
↪ 'vflprof', 'zprof', 'tprof', 'grill', 'rz_launch_grill_tab', 'eccone', 'dentab', 'dentab_
↪nonuniform_line', 'temtab', 'temtab_nonuniform_line', 'tpoptab', 'tpoptab_nonuniform_line',
↪'vflowtab', 'vflowtab_nonuniform_line', 'zeftab', 'zeftab_nonuniform_line', 'read_diskf',
↪'emission', 'ox', 'adj_nml', 'edge_prof_nml', 'lsc_approach_nml', 'edctsctab', 'jpartsctab'])
2023-11-15 14:18:53,357 [    spdm]     INFO: OrderedDict([('total_grills', 5),
             ('main_grills', 4),
             ('bt_direction', 1),
             ('power_fraction_factor', 0.8248)])
```

```
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value -0.319356437496 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.25 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 2.35087179772 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value -0.0903859428184 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 2.3432537639 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.135873276762 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 2.26070683323 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.35694008914 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.5 is not assigned to any variable and has been removed.
```

(continues on next page)

(continued from previous page)

```
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.001 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.7 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.9 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.5d is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.12 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 1.0 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.2 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.3 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.9.d0 is not assigned to any variable and has been removed.
  warnings.warn(
/home/salmon/.local/lib/python3.10/site-packages/f90nml/parser.py:827: UserWarning: f90nml:␣
↪warning: Value 0.0 is not assigned to any variable and has been removed.
  warnings.warn(
```

```
### 获取namelist中的关键字key
logger.info(output_data.keys())
### 通过child获取某个子节点的值，而不需要把整个树读回来
logger.info(output_entry.child("main_lobes").__value__)
```

```
2023-11-15 14:19:06,575 [    spdm]     INFO: dict_keys(['main_lobes', 'genr', 'tokamak', 'wave',
↪'scatnper', 'dispers', 'numercl', 'output', 'plasma', 'species', 'varden', 'denprof', 'tpopprof',
↪ 'vflprof', 'zprof', 'tprof', 'grill', 'rz_launch_grill_tab', 'eccone', 'dentab', 'dentab_
↪nonuniform_line', 'temtab', 'temtab_nonuniform_line', 'tpoptab', 'tpoptab_nonuniform_line',
↪'vflowtab', 'vflowtab_nonuniform_line', 'zeftab', 'zeftab_nonuniform_line', 'read_diskf',
↪'emission', 'ox', 'adj_nml', 'edge_prof_nml', 'lsc_approach_nml', 'edctsctab', 'jpartsctab'])
2023-11-15 14:19:06,580 [    spdm]     INFO: OrderedDict([('total_grills', 5),
             ('main_grills', 4),
             ('bt_direction', 1),
             ('power_fraction_factor', 0.8248)])
```

```
#### NotImplementedError: TODO: NAMELISTFile.write
# with File(f"{DATA_INPUT}/test-eq.h5", mode="w", format="namelist") as oid:
#     oid.write(eq_test)
```

14.3.2 netCDF 文件

```
### 以GENRAY的剖面输入文件genray_profs_in.nc为例，它是一个netCDF格式的文件。
with File(f"./data/genray_profs_in.nc", format= "NetCDf" ,mode="r") as oid:
### 仅仅建立链接，而没有拿回数据
    output_entry = oid.read()
### 拿回所有数据
    output_data = oid.read().dump()
### 获取namelist中的关键字key
logger.info(output_data.keys())
logger.info(output_data["dmass"].data)
### 通过child获取某个子节点的值，而不需要把整个树读回来
# logger.info(output_entry.child("dmass").__value__)
```

```
2023-11-15 14:19:21,349 [    spdm]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_netcdf
2023-11-15 14:19:21,373 [    spdm]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/plugins/
↪data/plugin_netcdf.py:149:open: Open NetCDF File ./data/genray_profs_in.nc mode=Mode.read
2023-11-15 14:19:21,379 [    spdm]     INFO: dict_keys(['charge', 'dmass', 'en', 'eqdsk_name', 'nj
↪', 'nspecgr', 'r', 'temp', 'zeff', 'title'])
2023-11-15 14:19:21,380 [    spdm]     INFO: [1.00000000e+00 1.83619995e+03 3.67239990e+03 2.
↪20343994e+04]
```

```
#### 写.nc文件
with File(f"./data/test-eq.nc", mode="w", format="netcdf") as oid:
    oid.write(eq_test)
with File(f"./data/test-eq.nc", format= "NetCDf" ,mode="r") as oid:
    test_data = oid.read().dump()
    logger.info(test_data.keys())
    logger.info(test_data["wall"])
```

```
2023-11-15 14:19:32,717 [    spdm]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/plugins/
↪data/plugin_netcdf.py:149:open: Open NetCDF File ./data/test-eq.nc mode=Mode.create|write
2023-11-15 14:19:32,741 [    spdm]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/plugins/
↪data/plugin_netcdf.py:149:open: Open NetCDF File ./data/test-eq.nc mode=Mode.read
2023-11-15 14:19:32,745 [    spdm]     INFO: dict_keys(['wall', 'equilibrium'])
2023-11-15 14:19:32,746 [    spdm]     INFO: {'description_2d': {'0': {'limiter': {'unit': {'0': {
↪'outline': {'r': array([1.35838, 1.35838, 1.35838, 1.36314, 1.4371 , 1.43721, 1.44164,
       1.4297 , 1.39169, 1.39151, 1.37929, 1.399  , 1.43623, 1.45919,
       1.47791, 1.54494, 1.61204, 1.63506, 1.66054, 1.66519, 1.6955 ,
       1.70668, 1.71388, 1.714  , 1.73649, 1.76324, 1.80199, 1.80237,
       1.93972, 1.97063, 2.07495, 2.1771 , 2.27925, 2.35   , 2.35   ,
       2.35   , 2.27925, 2.1771 , 2.07495, 1.9728 , 1.97063, 1.92059,
       1.87055, 1.82051, 1.79281, 1.76511, 1.75421, 1.67782, 1.60143,
       1.48426, 1.36709, 1.33132, 1.37874, 1.42615, 1.40528, 1.3844 ,
       1.36353, 1.35838, 1.35838, 1.35838, 1.35838]),
                                                              'z': array([ 0.      ,  0.227   ,␣
↪ 0.454   ,  0.455735,  0.798332,  0.798821,
        0.838195,  0.908908,  1.01426 ,  1.01473 ,  1.04859 ,  1.03946 ,
        1.0256  ,  1.02095 ,  1.02514 ,  1.05454 ,  1.08394 ,  1.09586 ,
        1.114   ,  1.11765 ,  1.14183 ,  1.16211 ,  1.13093 ,  1.13044 ,
        1.033   ,  0.969832,  0.92567 ,  0.925347,  0.809223,  0.783389,
        0.68838 ,  0.58904 ,  0.4897  ,  0.49    ,  0.      , -0.49    ,
       -0.4897  , -0.58904 , -0.68838 , -0.78772 , -0.783389, -0.832375,
       -0.881361, -0.930347, -1.05034 , -1.17034 , -1.13917 , -1.03465 ,
```

(continues on next page)

(continued from previous page)

```
       -0.930138, -0.95338 , -0.976622, -1.01072 , -0.879327, -0.747939,
       -0.65062 , -0.5533  , -0.455981, -0.454   , -0.227   ,  0.      ,
        0.      ])}}}}}}}}
```

14.3.3 HDF5 文件

```
### 以testall.h5，它是一个HDF5格式的文件。
with File(f"./data/testall.h5", format= "HDF5" ,mode="r") as oid:
### 仅仅建立链接，而没有拿回数据
    output_entry = oid.read()
### 拿回所有数据
    output_data = oid.read().dump()
### 获取namelist中的关键字key
logger.info(output_data.keys())
logger.info(output_data["coherent_wave"]["global_quantities"])

### 通过child获取某个子节点的值，而不需要把整个树读回来
# logger.info(output_entry.child("coherent_wave.current_tor").__value__)
```

```
2023-11-15 14:19:43,508 [    spdm]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/utils/
↪sp_export.py:66:sp_load_module: Load module spdm.plugins.data.plugin_hdf5
2023-11-15 14:19:43,511 [    spdm]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/plugins/
↪data/plugin_hdf5.py:197:open: Open HDF5 File ./data/testall.h5 mode=Mode.read
2023-11-15 14:19:43,518 [    spdm]     INFO: dict_keys(['coherent_wave'])
2023-11-15 14:19:43,519 [    spdm]     INFO: {'current_tor': 461821.31203568814,
 'frequency': 4600000000.0,
 'power': 19375694355948.902}
```

```
#### 写.nc文件
with File(f"./data/test-eq.hdf5", mode="w", format="HDF5") as oid:
    oid.write(eq_test)
with File(f"./data/test-eq.hdf5", format= "HDF5" ,mode="r") as oid:
    test_data = oid.read().dump()
    logger.info(test_data.keys())
    logger.info(test_data["wall"])
```

```
2023-11-15 14:19:47,011 [    spdm]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/plugins/
↪data/plugin_hdf5.py:197:open: Open HDF5 File ./data/test-eq.hdf5 mode=Mode.create|write
2023-11-15 14:19:47,034 [    spdm]    DEBUG: /home/salmon/workspace/fytok/SpDM/python/spdm/plugins/
↪data/plugin_hdf5.py:197:open: Open HDF5 File ./data/test-eq.hdf5 mode=Mode.read
2023-11-15 14:19:47,044 [    spdm]     INFO: dict_keys(['equilibrium', 'wall'])
2023-11-15 14:19:47,046 [    spdm]     INFO: {'description_2d': [{'limiter': {'unit': [{'outline':
↪{'r': array([1.35838, 1.35838, 1.35838, 1.36314, 1.4371 , 1.43721, 1.44164,
       1.4297 , 1.39169, 1.39151, 1.37929, 1.399  , 1.43623, 1.45919,
       1.47791, 1.54494, 1.61204, 1.63506, 1.66054, 1.66519, 1.6955 ,
       1.70668, 1.71388, 1.714  , 1.73649, 1.76324, 1.80199, 1.80237,
       1.93972, 1.97063, 2.07495, 2.1771 , 2.27925, 2.35   , 2.35   ,
       2.35   , 2.27925, 2.1771 , 2.07495, 1.9728 , 1.97063, 1.92059,
       1.87055, 1.82051, 1.79281, 1.76511, 1.75421, 1.67782, 1.60143,
       1.48426, 1.36709, 1.33132, 1.37874, 1.42615, 1.40528, 1.3844 ,
       1.36353, 1.35838, 1.35838, 1.35838, 1.35838]),
```

(continues on next page)

(continued from previous page)

```
                                                'z': array([ 0.      ,  0.227   ,  0.454   ,
↪ 0.455735,  0.798332,  0.798821,
       0.838195,  0.908908,  1.01426 ,  1.01473 ,  1.04859 ,  1.03946 ,
       1.0256  ,  1.02095 ,  1.02514 ,  1.05454 ,  1.08394 ,  1.09586 ,
       1.114   ,  1.11765 ,  1.14183 ,  1.16211 ,  1.13093 ,  1.13044 ,
       1.033   ,  0.969832,  0.92567 ,  0.925347,  0.809223,  0.783389,
       0.68838 ,  0.58904 ,  0.4897  ,  0.49    ,  0.      , -0.49    ,
      -0.4897  , -0.58904 , -0.68838 , -0.78772 , -0.783389, -0.832375,
      -0.881361, -0.930347, -1.05034 , -1.17034 , -1.13917 , -1.03465 ,
      -0.930138, -0.95338 , -0.976622, -1.01072 , -0.879327, -0.747939,
      -0.65062 , -0.5533  , -0.455981, -0.454   , -0.227   ,  0.      ,
       0.      ])}}}]}}}]}
```

Note: （1）其他的半结构化数据,YAML,JSON等，访问形式类似。

INDEX

Non-alphabetical

A

B

C

D

E

F

G

H

I

J

M

N

Z