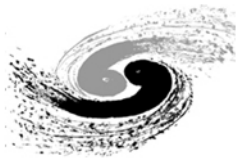


中国科学院高能物理研究所
Institute of High Energy Physics
Chinese Academy of Sciences

国家高能物理科学数据中心
National HEP Data Center

高能所計算中心
IHEP Computing Center

光源实验数据处理及软件框架介绍

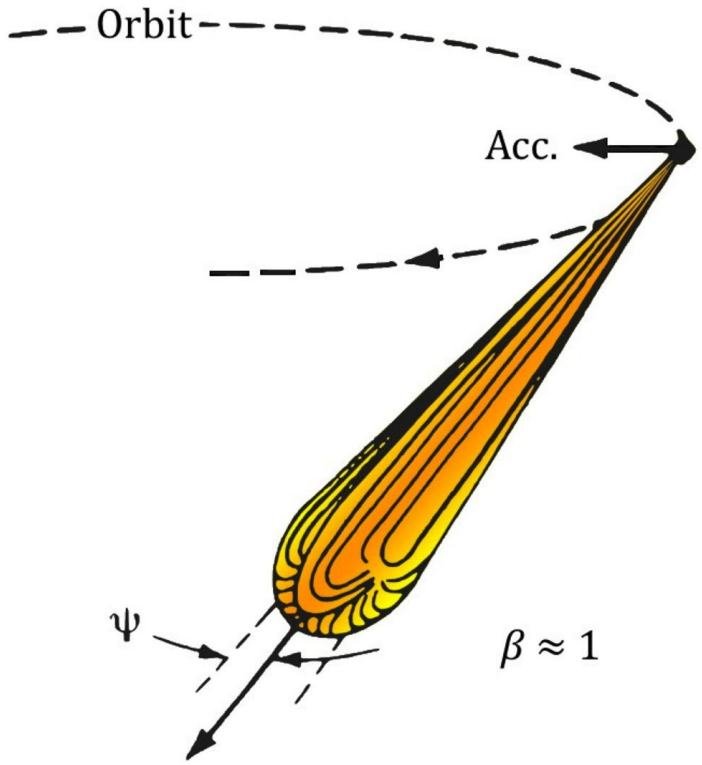
胡誉 (huyu@ihep.ac.cn)

中国科学院高能物理研究所

第五届高能物理计算暑期学校 2024年08月

Introduction to synchrotron radiation

Synchrotron radiation is produced by relativistic charged particles accelerated by magnetic fields. It is observed at particle accelerators



v ≈ c or β = v/c ≈ 1

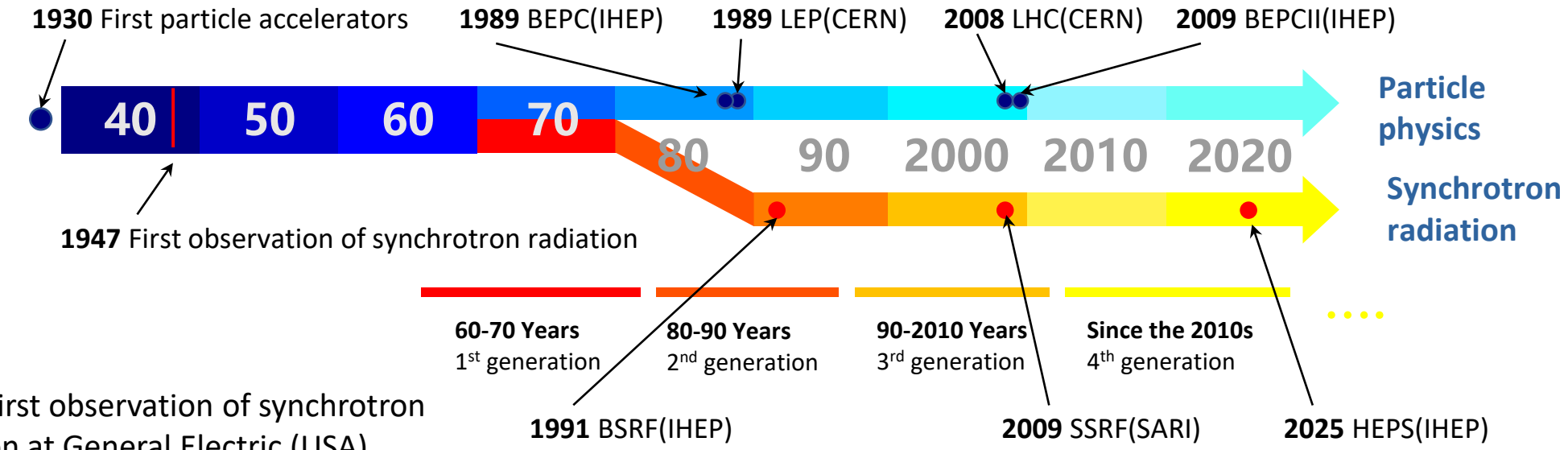
The vertical half-opening angle:

$$\psi \approx \frac{m_0 c^2}{\mathrm{E}} = \frac{1}{\gamma}$$

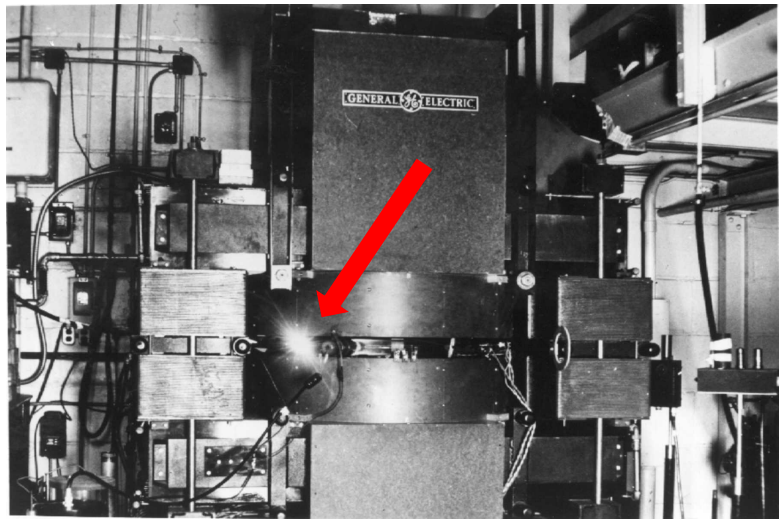
BSRF: E_e = 2.2 GeV, γ = 4 305, ψ = 232 μrad
HEPS: E_e = 6 GeV, γ = 11 800, ψ = 85 μmrad

The emission is concentrated in the forward direction

History of Synchrotron Radiation Sources

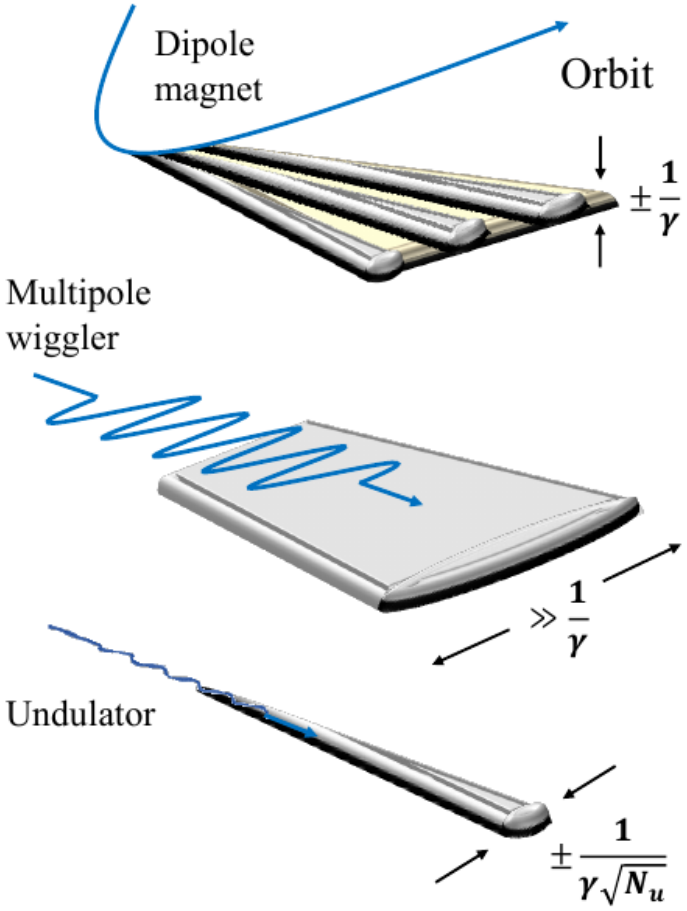


1947: First observation of synchrotron radiation at General Electric (USA).



… followed by decades of parasitic use of Synchrotron radiation on high-energy machines.

History of Synchrotron Radiation Sources



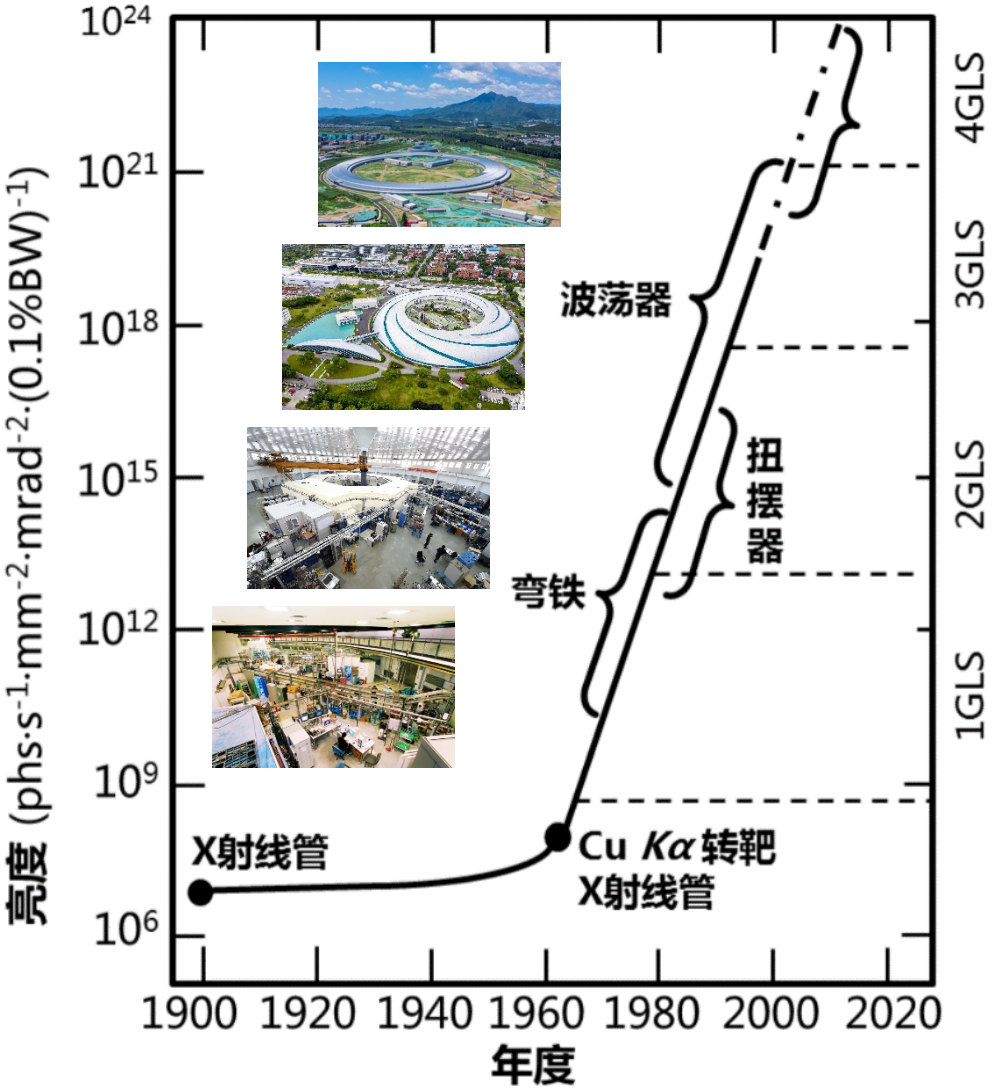
Insertion device

1GLS

2GLS

3GLS

4GLS



Major synchrotron facilities worldwide

There are more than 50 light sources in the world

2009 – PETRA III – 6 GeV

2009 – SSRF – 3.5 GeV

1996 – APS - 7 GeV

1994 – ESRF – 6 GeV

1997 – Super Photon RING 8 GeV

Major synchrotron facilities worldwide

There are more than 50 light sources in the world

2009 – PETRA III – 6 GeV

2025 – HEPS – 6.0 GeV

2009 – SSRF – 3.5 GeV

1996 – APS - 7 GeV

1994 – ESRF – 6 GeV

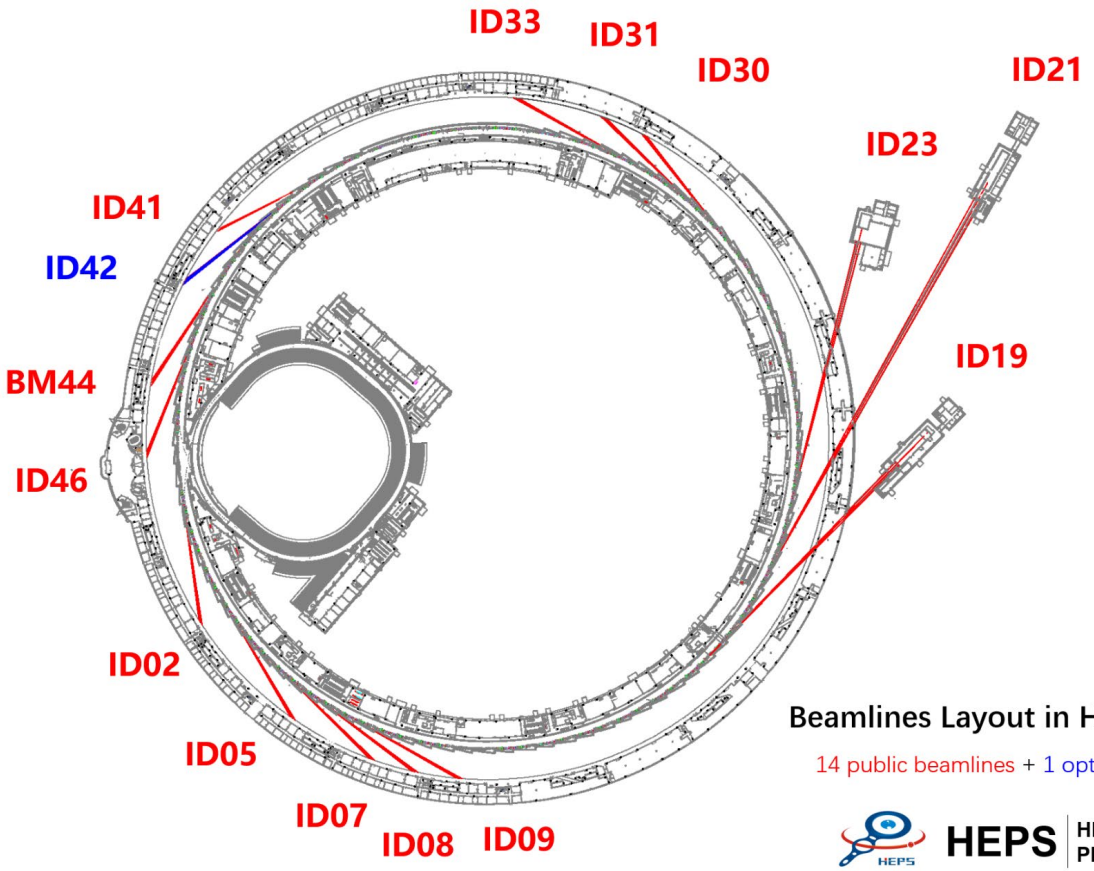
1997 – Super Photon RING 8 GeV

High Energy Photon Source (HEPS)

- New light source in China — High energy, high brightness
- Located in Beijing - about 80KM from IHEP
- Officially approved in Dec. 2017
- The construction was started in the mid-2019
- The whole project will be finished in mid-2025

Main parameters	Unit	Value
Beam energy	GeV	6
Circumference	m	1360.4
Emittance	pm·rad	< 60
Brightness	phs/s/mm^2/mrad^2/0.1%BW	>1x10^{22}
Beam current	mA	200
Injection		Top-up

Beamlines in HEPS phase I



Beamlines Layout in HEPS phase I

14 public beamlines + 1 optics test beamline

HEPS HIGH ENERGY PHOTON SOURCE

Microfocusing X-Ray Protein Crystallography-ID02 Beamline

Low-Dimensional Structure Probe Beamline-ID05

Engineering Materials Beamline-ID07

Hard X-Ray Coherent Scattering Beamline-ID09

Pink Beam SAXS Beamline-ID08

Hard X-Ray Nanoprobe Multimodal Imaging-ID19 Beamline

Hard X-Ray Imaging Beamline-ID21

Structural Dynamics Beamline-ID23

ID30-Transmission X-Ray Microscopic Beamline

ID31-High Pressure Beamline

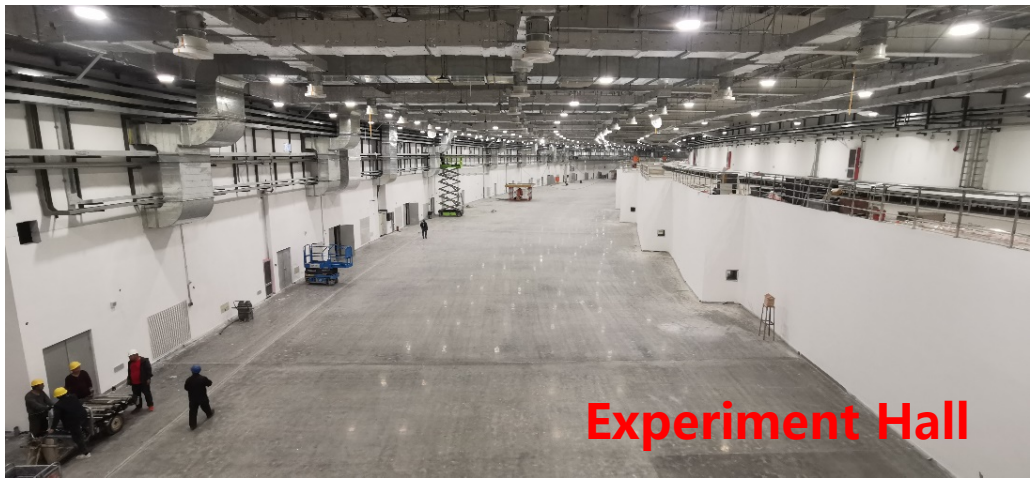
ID33-Hard X-Ray High Resolution Spectroscopy Beamline

BM44-Tender X-Ray Beamline

ID41-High Resolution Nanoscale Electronic Structure Spectroscopy Beamline

ID42-Optics Test Beamline

ID46-X-Ray Absorption Spectroscopy Beamline

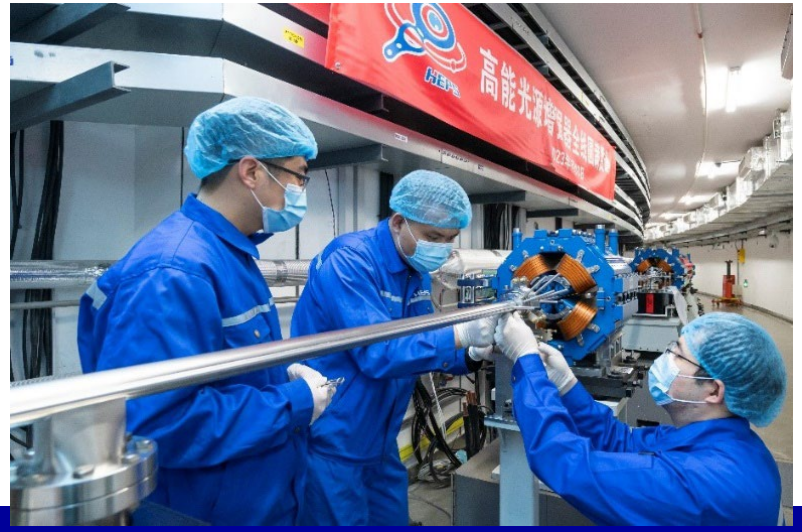


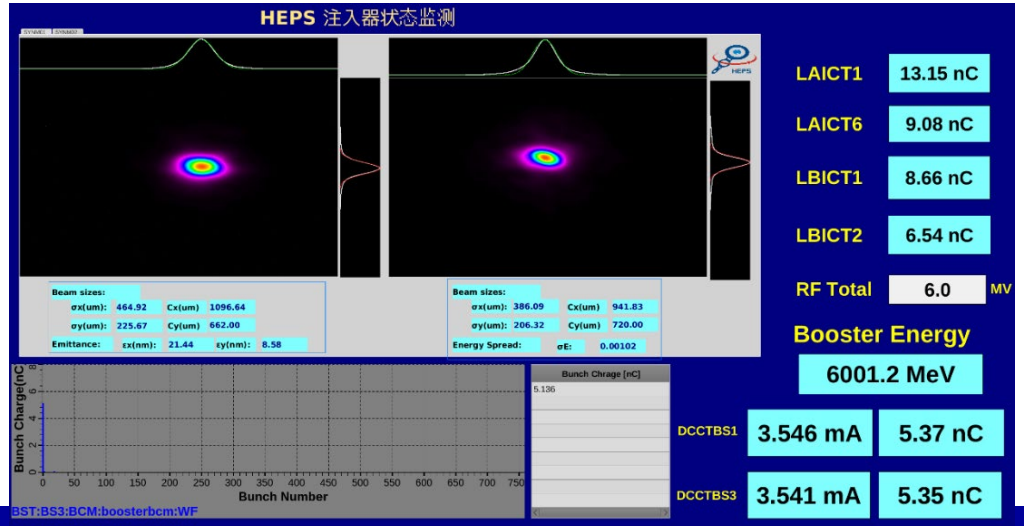
14 public beamlines + 1 optics test beamline in Phase I

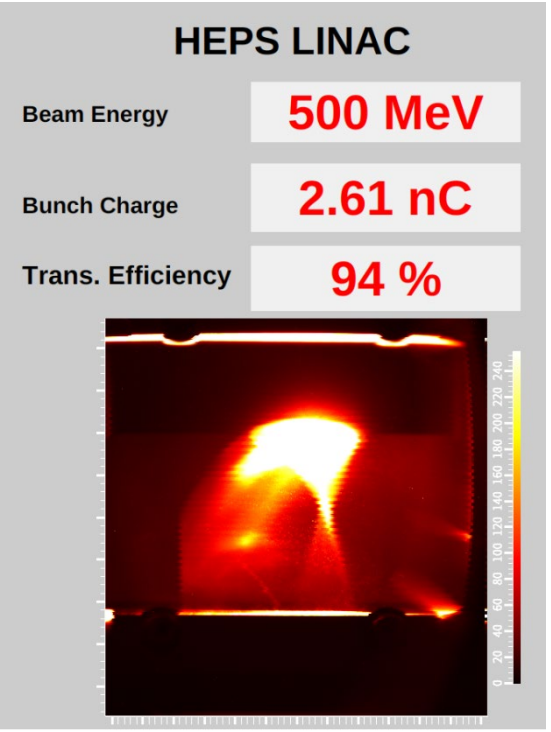
Can accommodate over 90 beamlines in total

Progress of the HEPS project

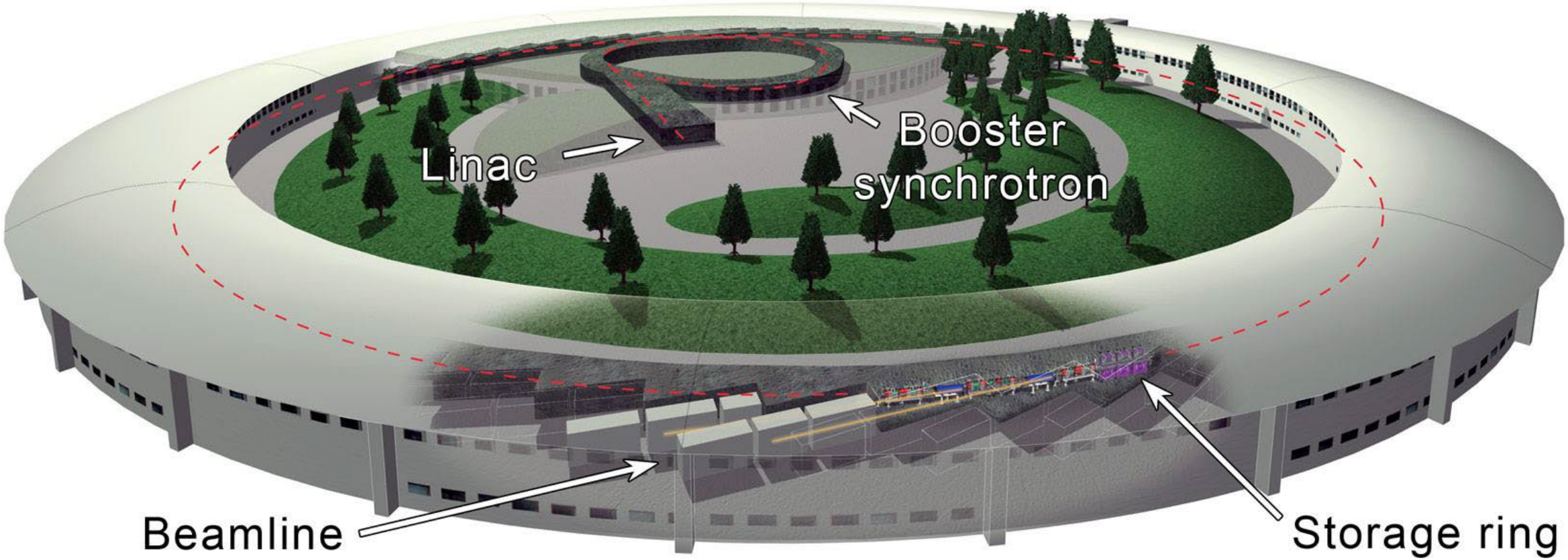
- The construction of the civil structure completed. Now at the stage of equipment installation
- 2023.01, HEPS booster installation completed
- 2023.03, HEPS achieved the first electron beam accelerated to 500 MeV
- 2023.11, Electron beam ramped up to 6 GeV
- 2024.07, HEPS storage ring installation completed
- 1st SR X-ray to be emitted in the near future







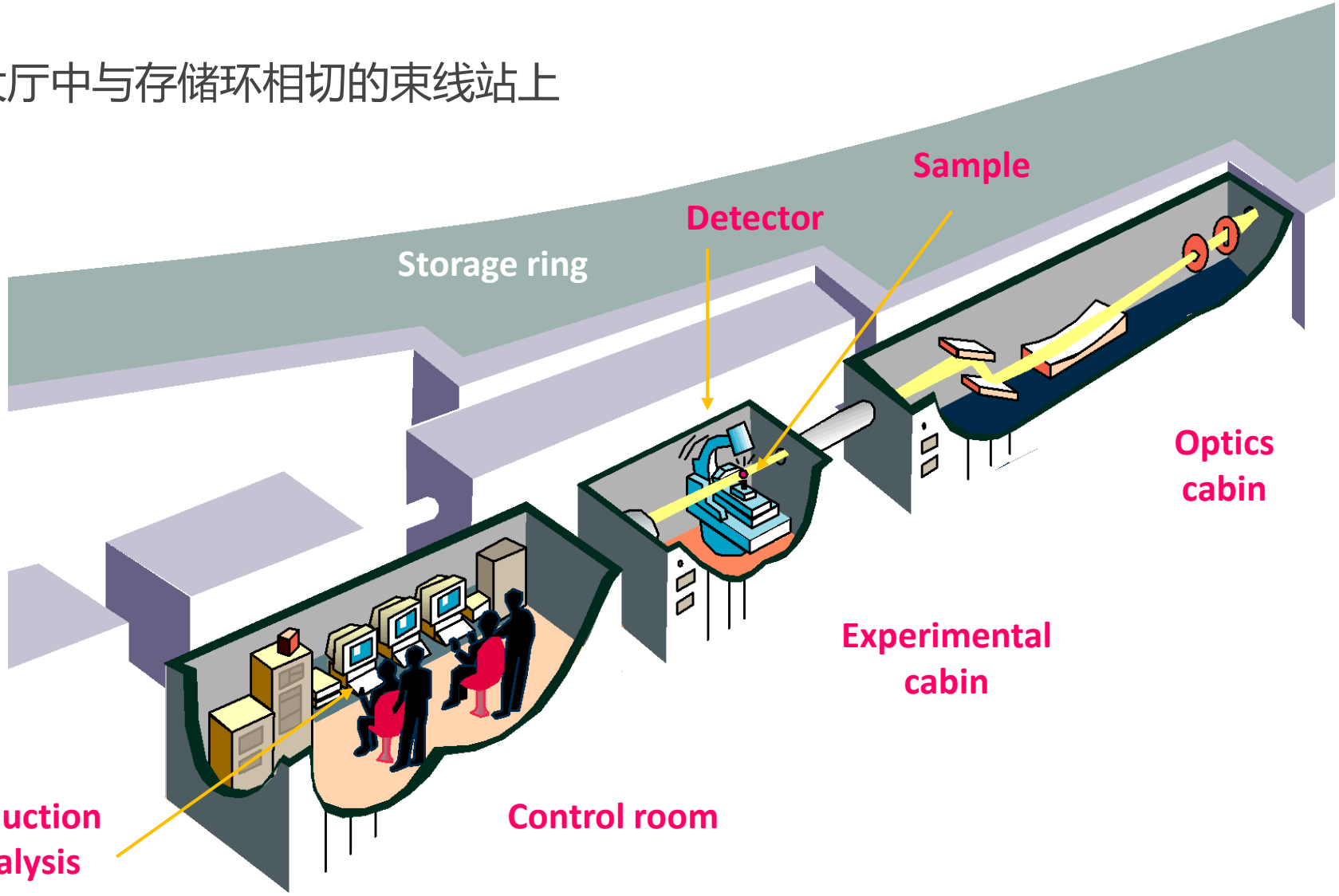
The layout of a synchrotron source



A typical synchrotron beamline

电子发出的x射线被引到实验大厅中与存储环相切的束线站上

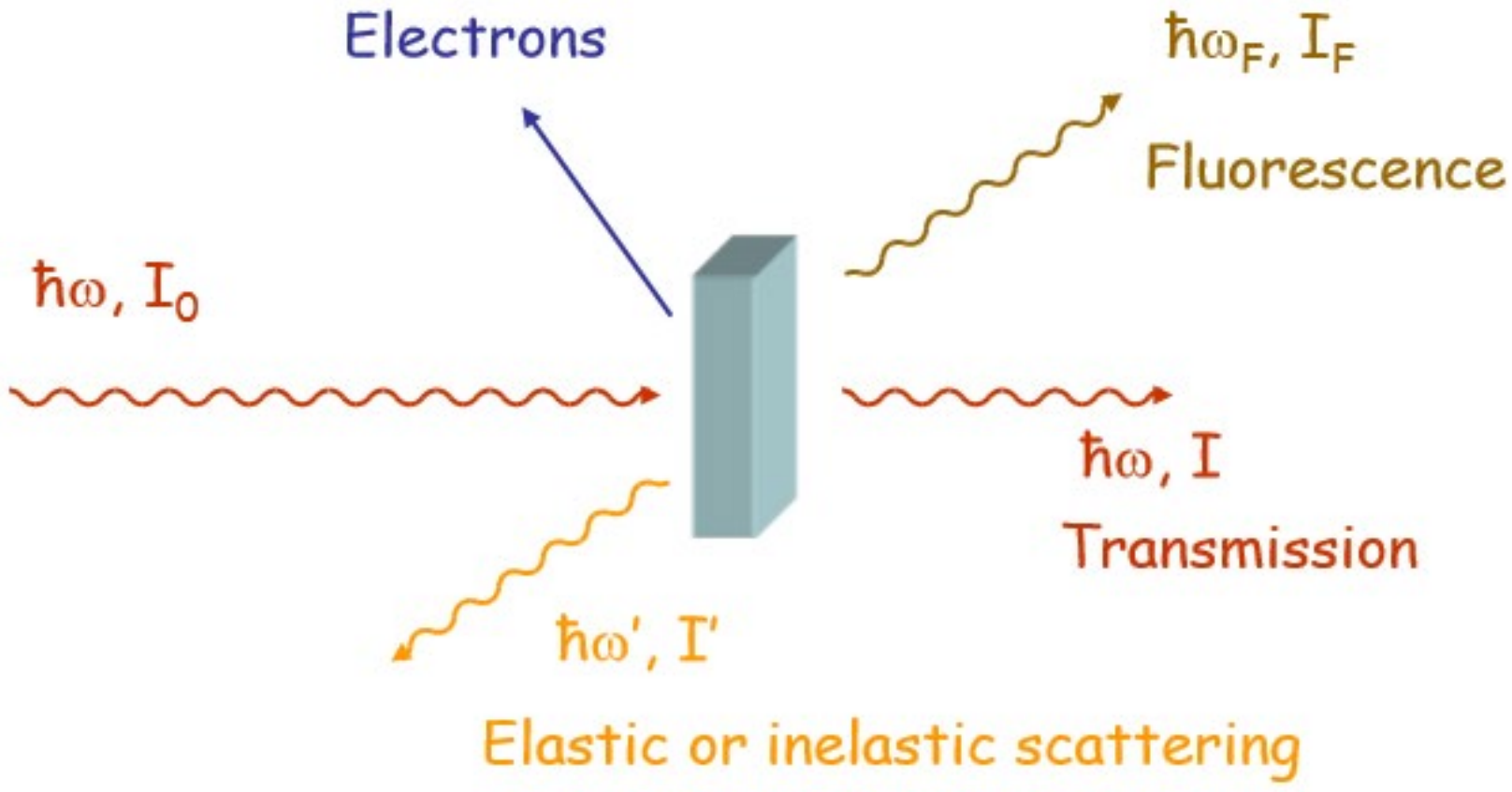
每条光束线站都针对特定的研究技术而设计



Synchrotron radiation - applications

Fundamental and applied studies on materials and living matter

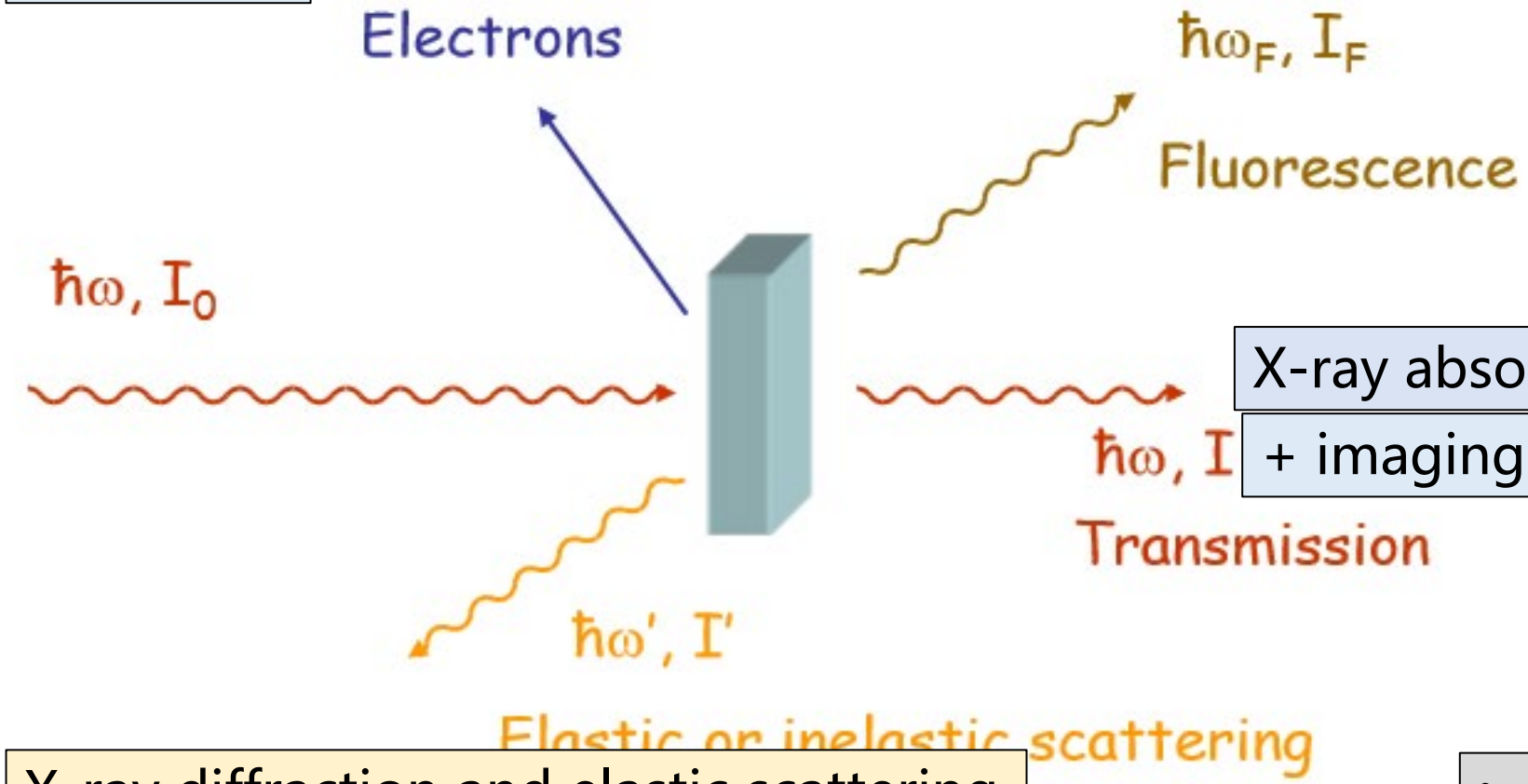
X-ray – matter interaction



X-ray – matter interaction: the classical techniques

Photoemission spectroscopy

+ imaging



X-ray absorption spectroscopy

+ imaging

X-ray diffraction and elastic scattering

+ imaging

- Diffraction & Scattering
- Spectroscopy
- Imaging

Types of synchrotron light experiments

衍散射

- 光被样品本身衍射，形成衍射图样
- 通过对衍射图样研究，可以获得原子结构、体相、表界面等纳米尺度结构信息

谱学

- 光穿过样品被吸收，在另一侧可以测量到被吸收后的光谱
- 通过光谱分析，可以获得电子能级、能带结构、自旋和轨道磁矩、原子配位等信息

成像

- 光从样品透射，形成样品内部的对比图像
- 分析投影图像，可以获得形貌结构、磁畴结构、元素及化学态空间分布等信息

……

X-ray diffraction and (in)elastic scattering

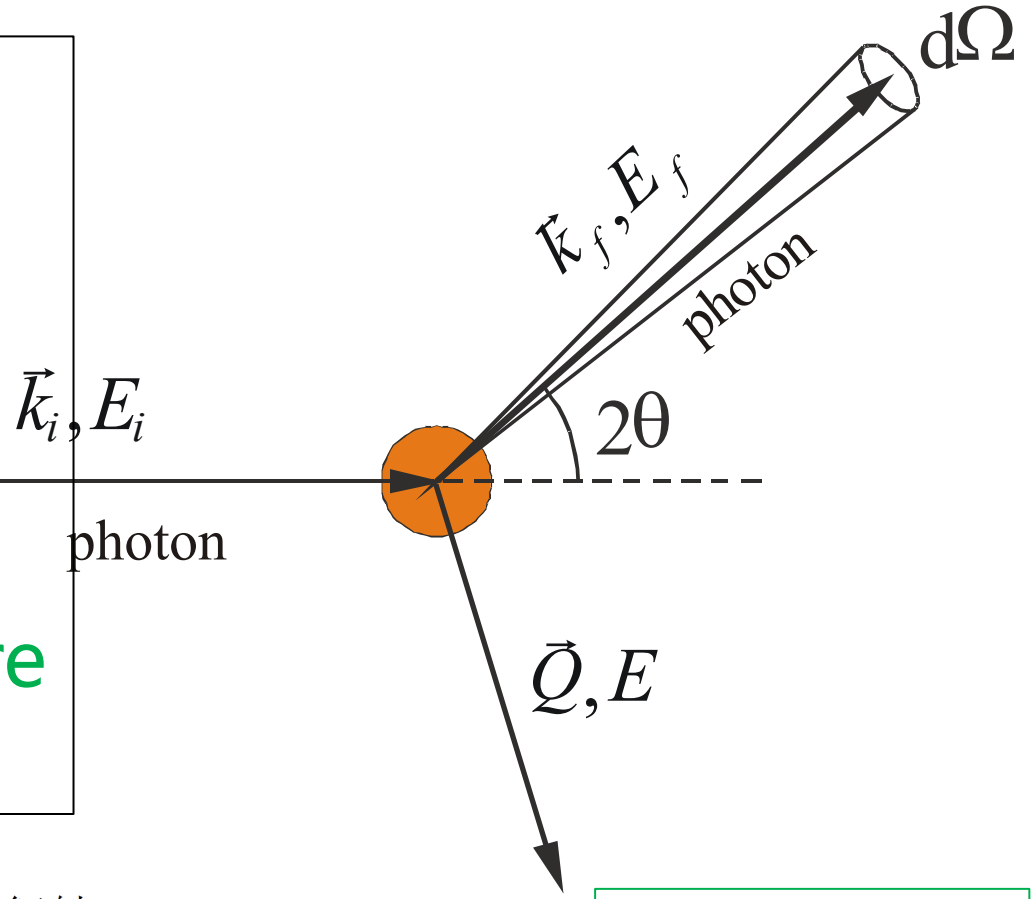
Elastic scattering

$E_f = E_i$

$|\vec{k_f}| = |\vec{k_i}|$

$Q = 4\pi/\lambda \cdot \sin(\theta)$

"determine where the atoms are"



散射
- 弹性($E_i = E_o$) → 相干叠加 → 衍射
- 非弹性 ($E_i \neq E_o$)

$|\vec{k_i}| = 2\pi/\lambda$

$\vec{Q} = \vec{k_f} - \vec{k_i}$

$E = E_f - E_i$

Inelastic scattering

$E_f < E_i$

$|\vec{k_f}| < |\vec{k_i}|$

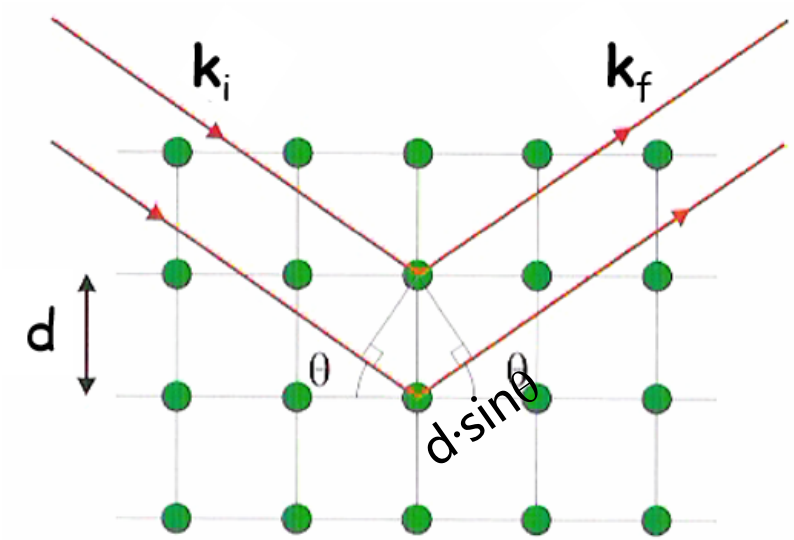
$Q = 4\pi/\lambda \cdot \sin(\theta)$

$E = E_f - E_i$

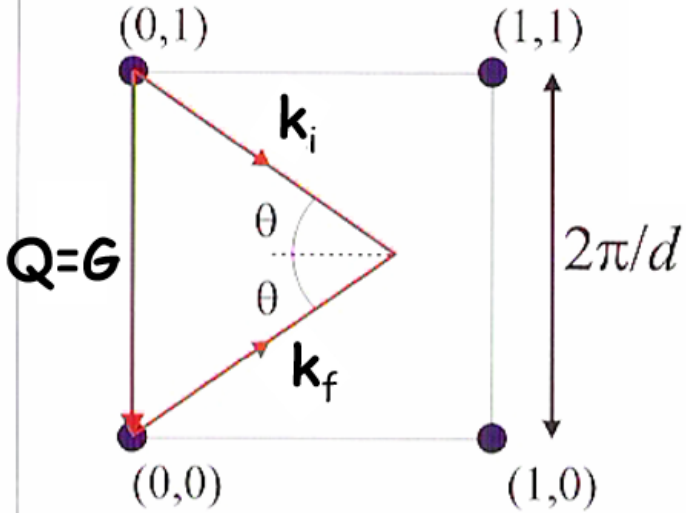
"determine where the atoms are and how they move"

Crystal diffraction: Bragg's law and laue condition

A **single crystal** is a **periodic array of atoms in 3D**.
Description as the Fourier transform of a 3D array of δ-**functions** in **reciprocal space**:
$G_{hkl}(\vec{Q})$ with integers h,k,l (Miller indices).
Diffraction only occurs when $\vec{Q} = \vec{G}$: **Laue condition**
This is geometrically equivalent to $\mathbf{n \cdot \lambda = 2 d_{hkl} \cdot \sin\theta}$: **Bragg law**

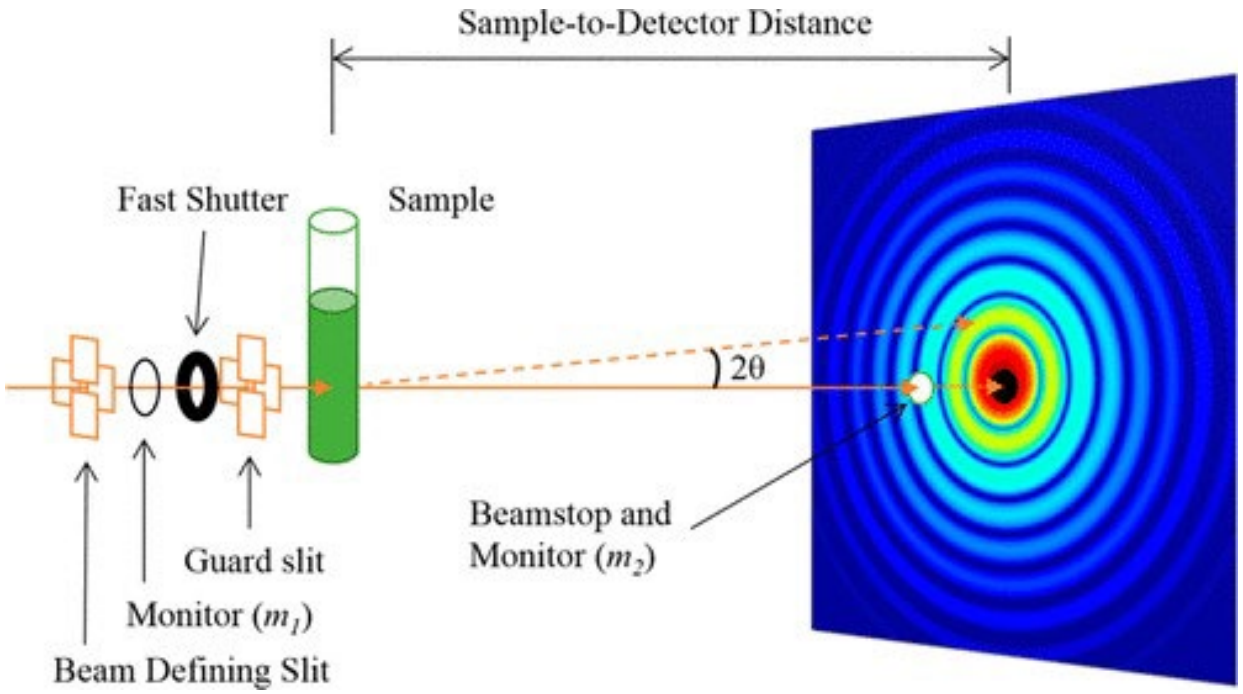


d is the lattice spacing of planes hkl

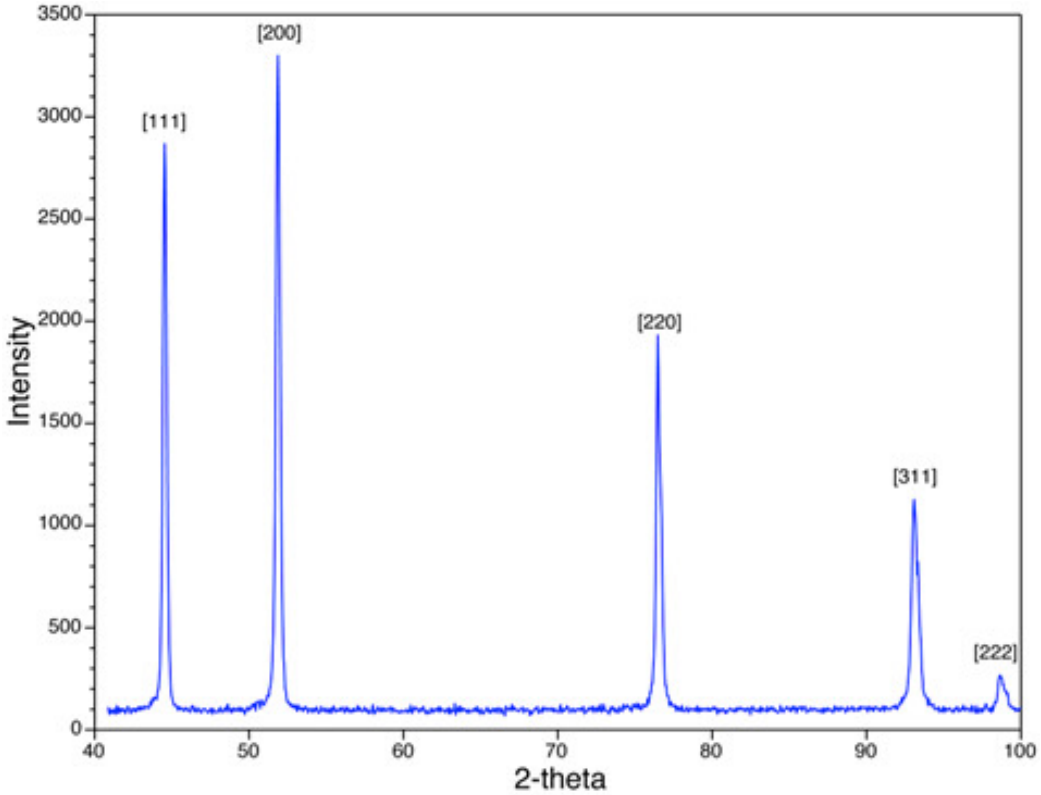


$$|\vec{Q}| = 4\pi \cdot \sin(\theta)/\lambda$$
or
$$|\vec{G}| = 2\pi/d_{hkl}$$

X-ray diffraction —— data analysis



Integral



衍射强度谱

X-ray diffraction —— data analysis

峰的位置

面间距：点阵参数、空间群

峰的强度

定量相分析，获得三维晶体结构

$$I = I_0 \frac{e^4}{m^2c^4} \frac{\lambda^3}{32\pi R} \frac{1}{V_0^2} F^2 P \frac{1+\cos^2 2\theta}{\sin^2\theta\cos\theta} e^{-2M} \frac{1}{2u}$$

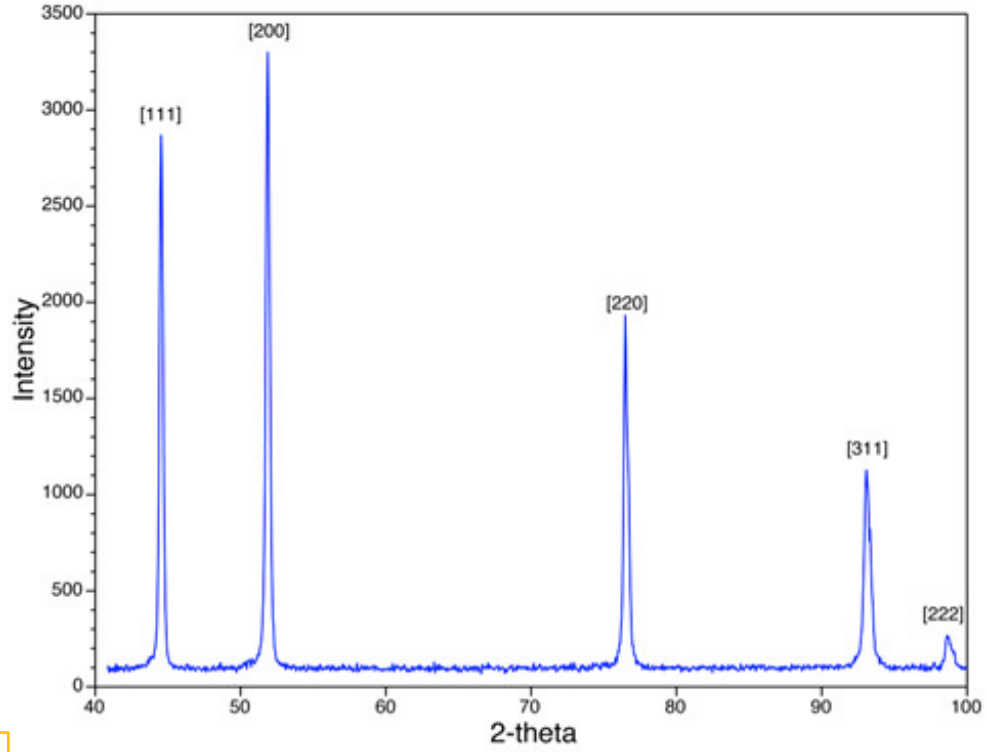
1. 结构因子
2. 多重性因子
3. 洛伦兹因子
4. 吸收因子
5. 温度因子

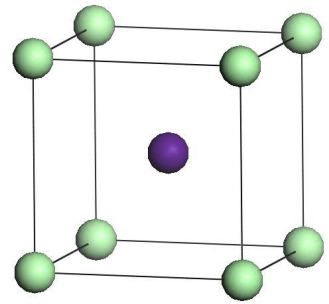
$$I_{hkl} \propto |F_{hkl}|^2$$

结构因子：

$$F_{hkl} = \sum_{j=1}^{m} N_j f_j \exp\left[2\pi i\left(hx_j + ky_j + lz_j\right)\right]$$

结构因子定量表征了原子个数、原子位置以及原子种类对衍射强度影响规律的参数





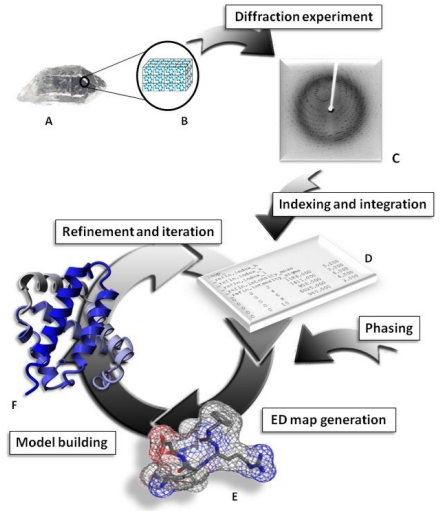
晶体结构

峰的形状

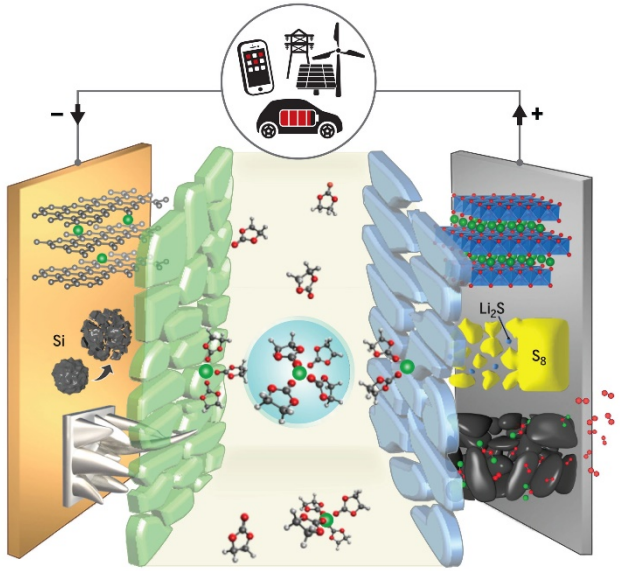
晶粒尺寸、结晶性

Applications of x-ray diffraction & scattering

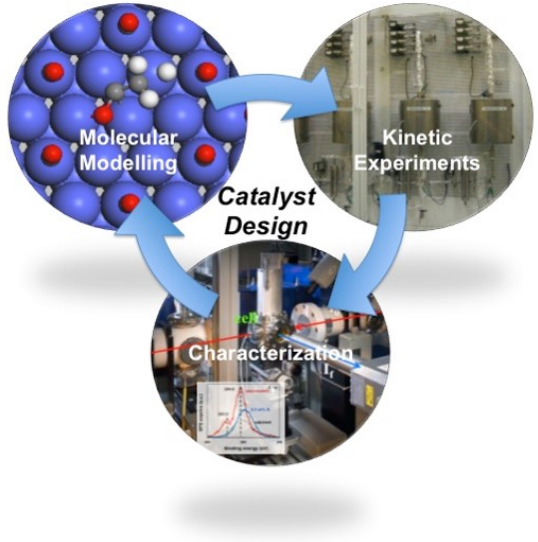
PDF，小角散射，广角散射，X射线衍射



生物大分子



电池研发



催化科学

医药

聚合物及复合材料

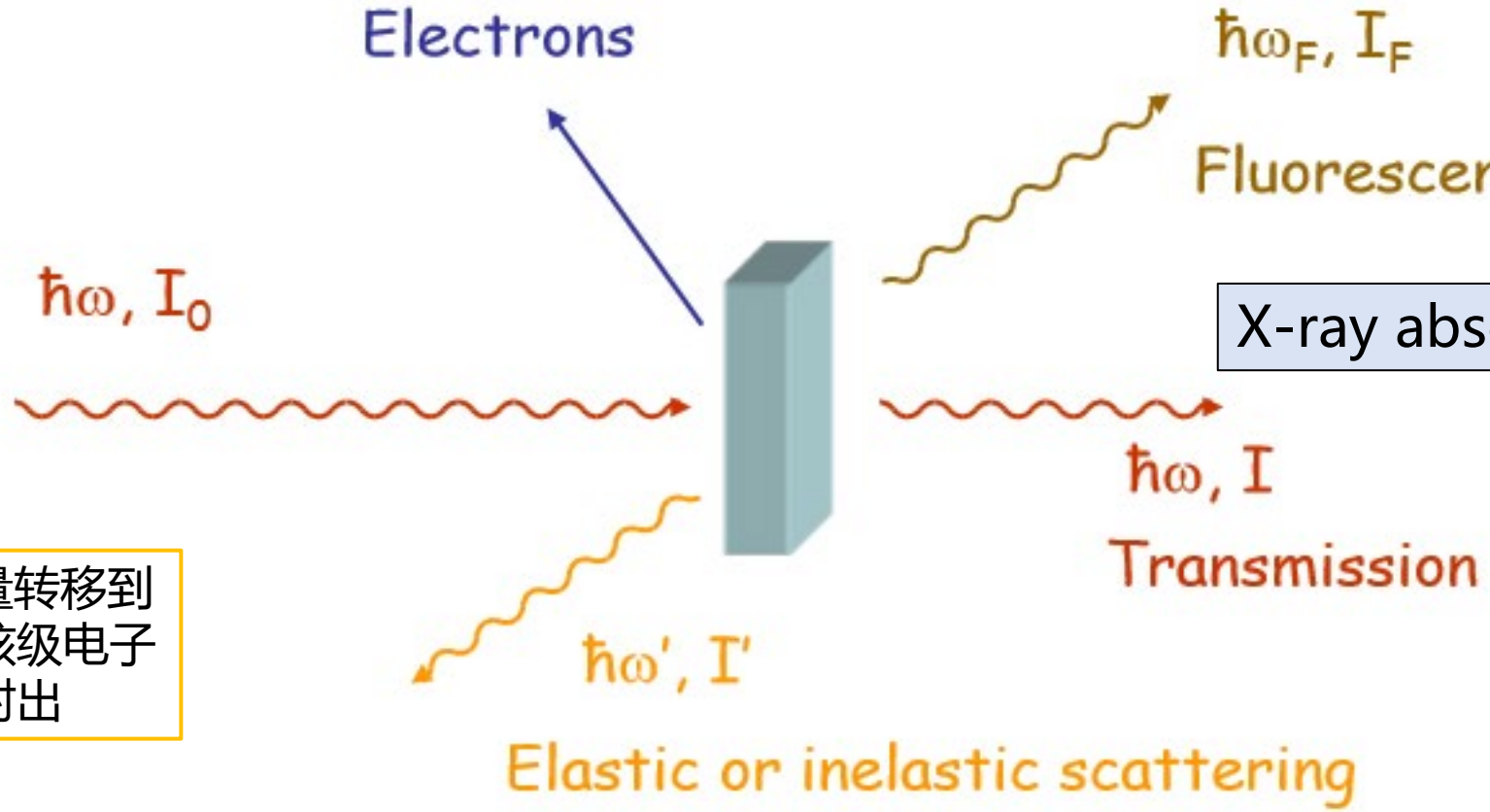
地质学

纳米材料

增材制造

……

X-ray spectroscopy



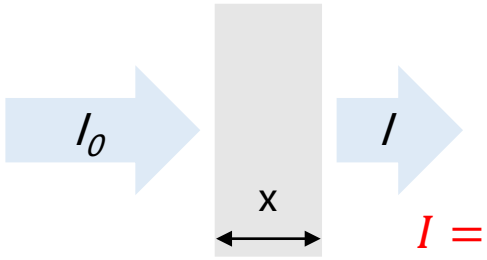
X 射线的能量转移到核级电子，核级电子从原子中发射出

X-ray absorption spectroscopy

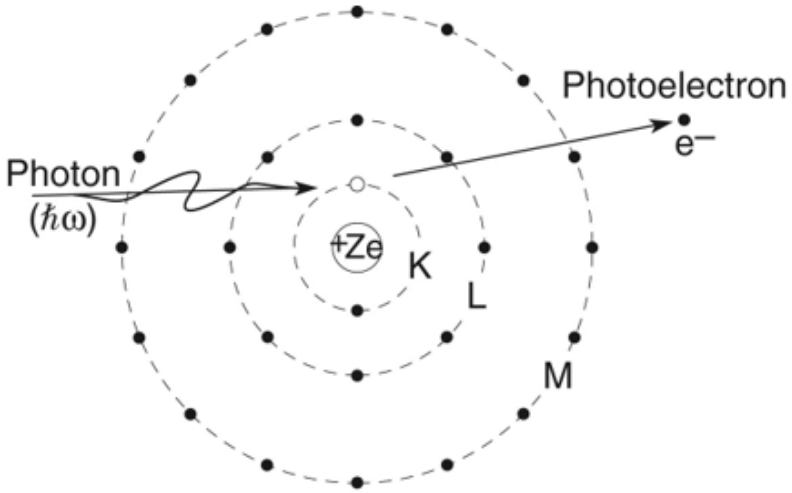
当光子能量与电子层跃迁能量相近时，光子会被原子共振吸收，吸收增强

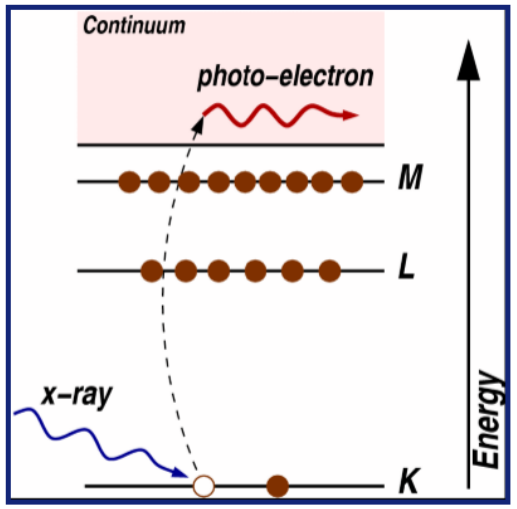
- X-ray Absorption Fine Structure, XAFS
- X-ray Absorption Near Edge Structure, XANES
- Extended X-ray Absorption Fine Structure, EXAFS

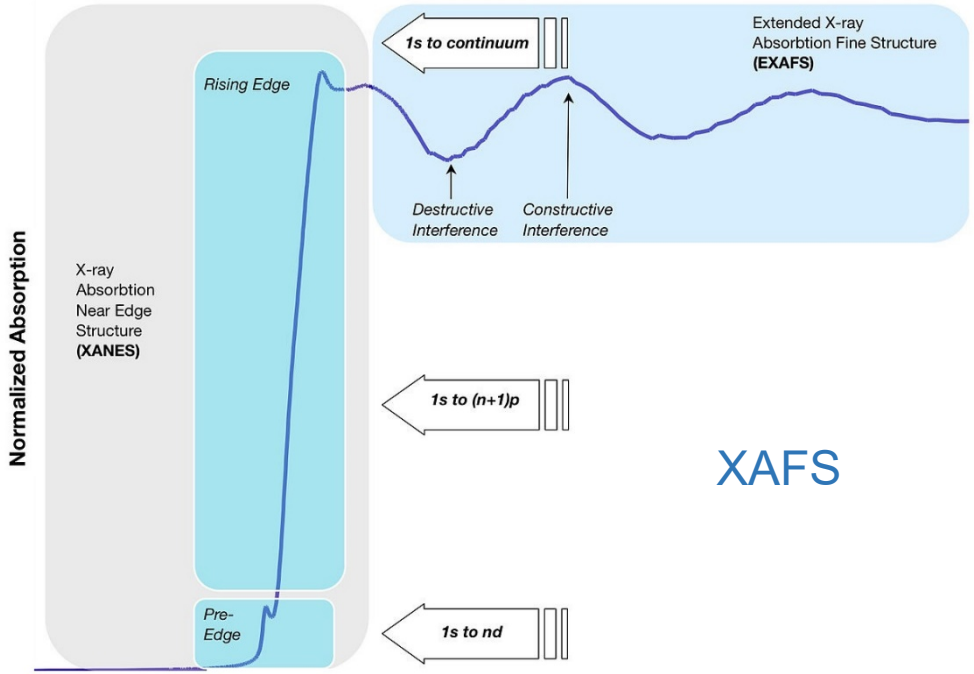
XAFS = XANES+EXAFS



$$I = I_0 e^{-\mu x}$$







XAFS

覆盖几乎所有元素的许多特征吸收边

X-ray absorption spectroscopy-data analysis

预处理：

能量校准、归一化、背景减除

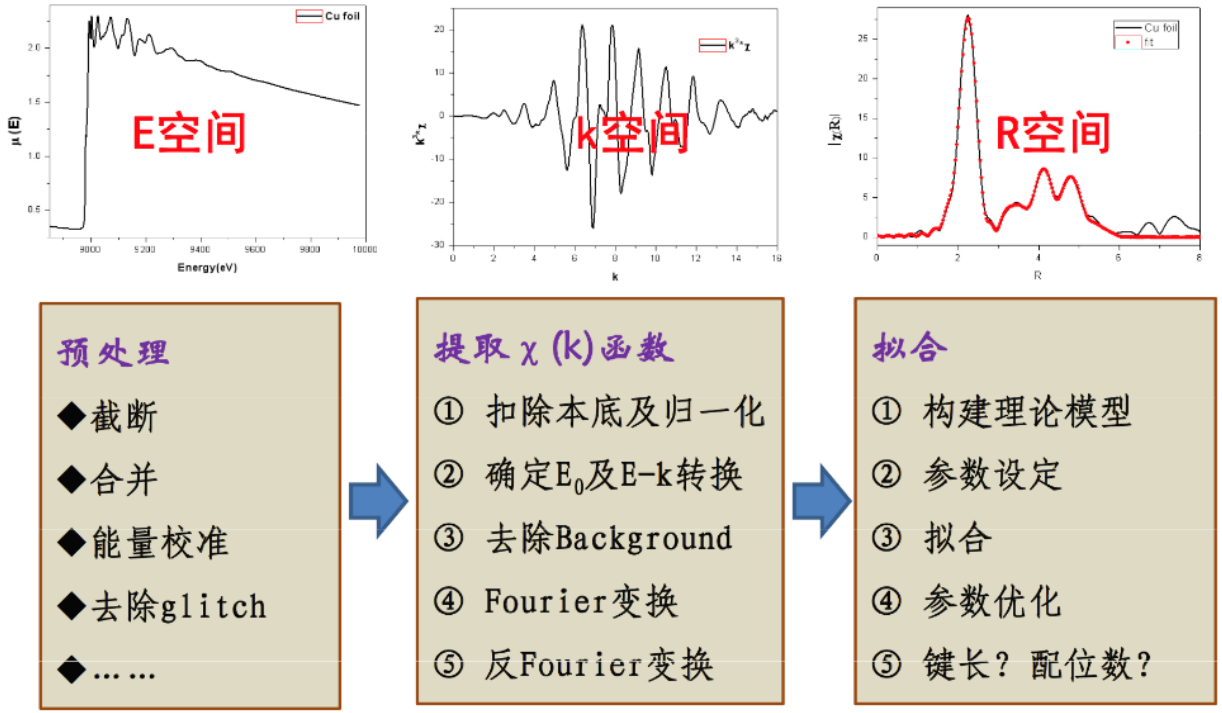
XANES:

LCA、PCA、指纹分析、模拟、标准样本比对（判断化合价状态和配位环境）

EXAFS:

小波分析，拟合，模拟

配位键长，配位数，原子种类



Applications of x-ray spectroscopy

从 XAFS 光谱的不同区域可以获得许多有关材料结构的信息

化学与催化

磁学

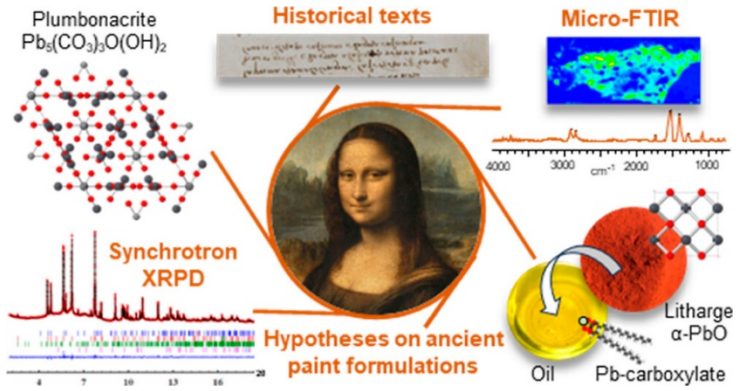
超导

地球与行星科学

环境科学

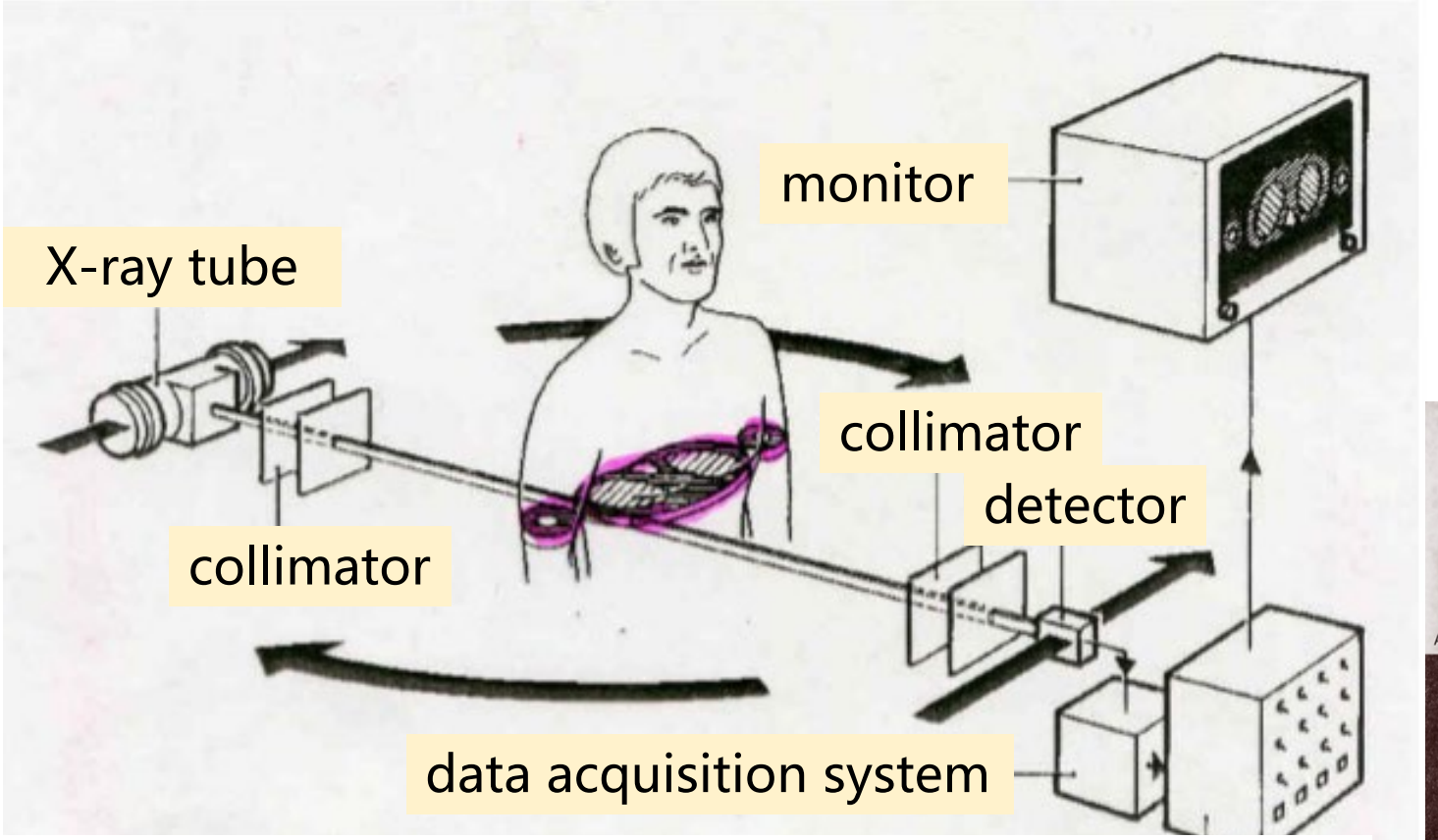
文化遗产

.....



区域	特征	可获得的信息
边前	● 因电子跃迁至外层空轨道导致 ● 跃迁受选律约束	● 原子局域结构 ● 氧化态 ● 成键情况
吸收边	● 给出电离能（吸收阈能）的数值	● 氧化态（化学位移） ● 吸收边常随氧化态的升高向高能区移动
XANES （特指多重散射的影响）	● 由低能光电子的多重散射现象导致 ● 具有大的散射截面	● 近邻原子位置，键长与键角 ● 几何结构
EXAFS	● 由高能光电子的单散射导致	● 配位半径

X-ray imaging – x-ray radiography

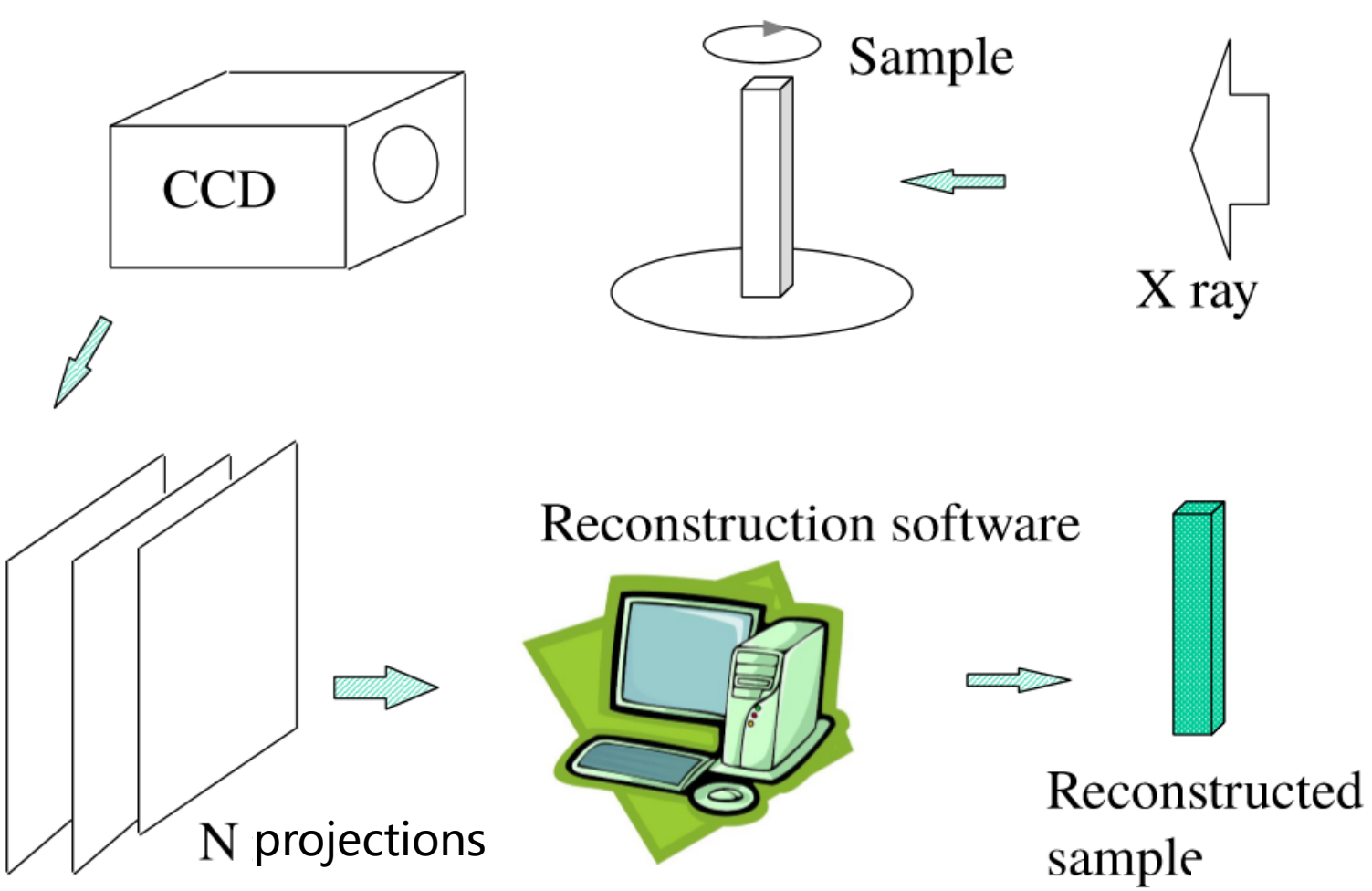


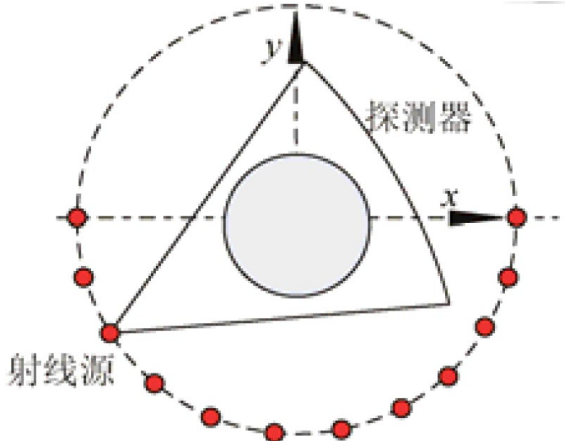
X-ray radiography: the oldest X-ray imaging technique
good lateral resolution
easily available in any lab or hospital

but no depth resolution = no three dimensional information

X-ray imaging – principle of x-ray tomography

360°(180°)rotation of the sample; at each Δθ take an image; 3D reconstruction from N projections





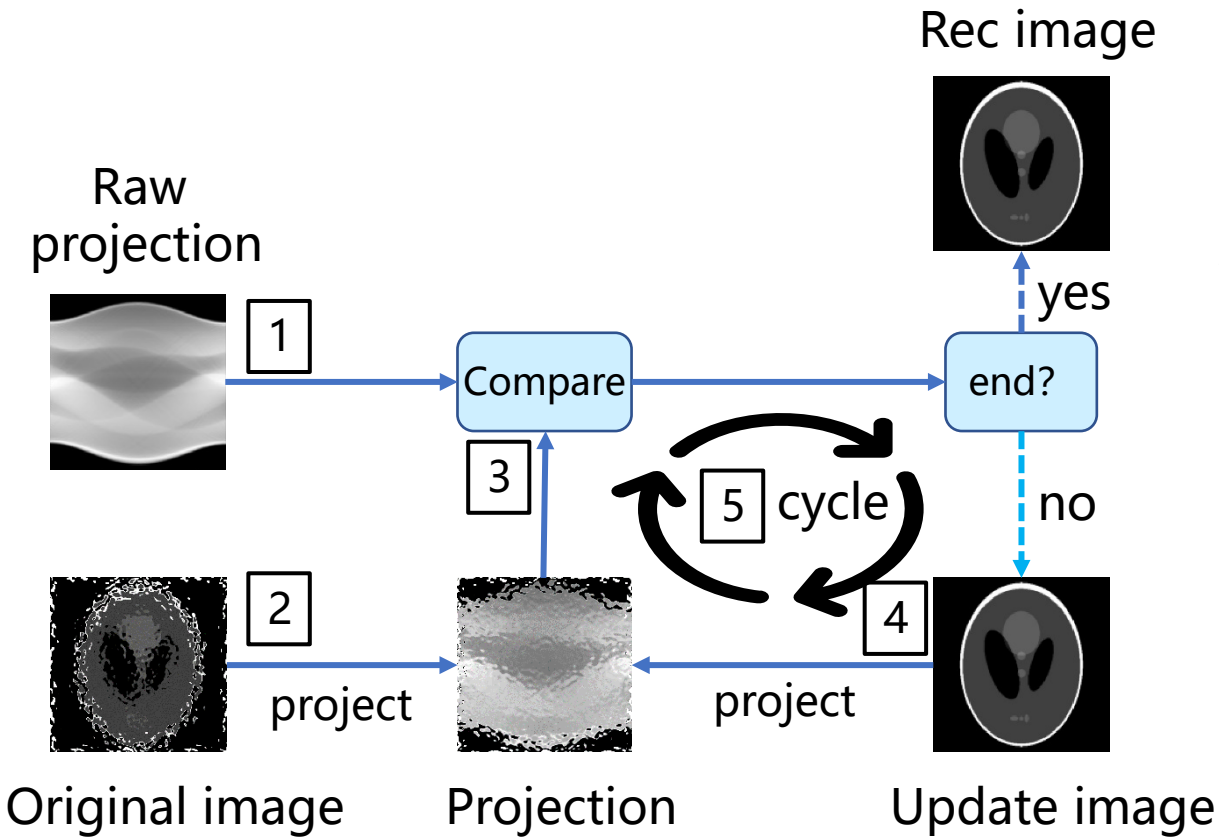
$N \cdot \Delta\theta = 360°(180°)$

compute intensive

Computed Tomography（CT）

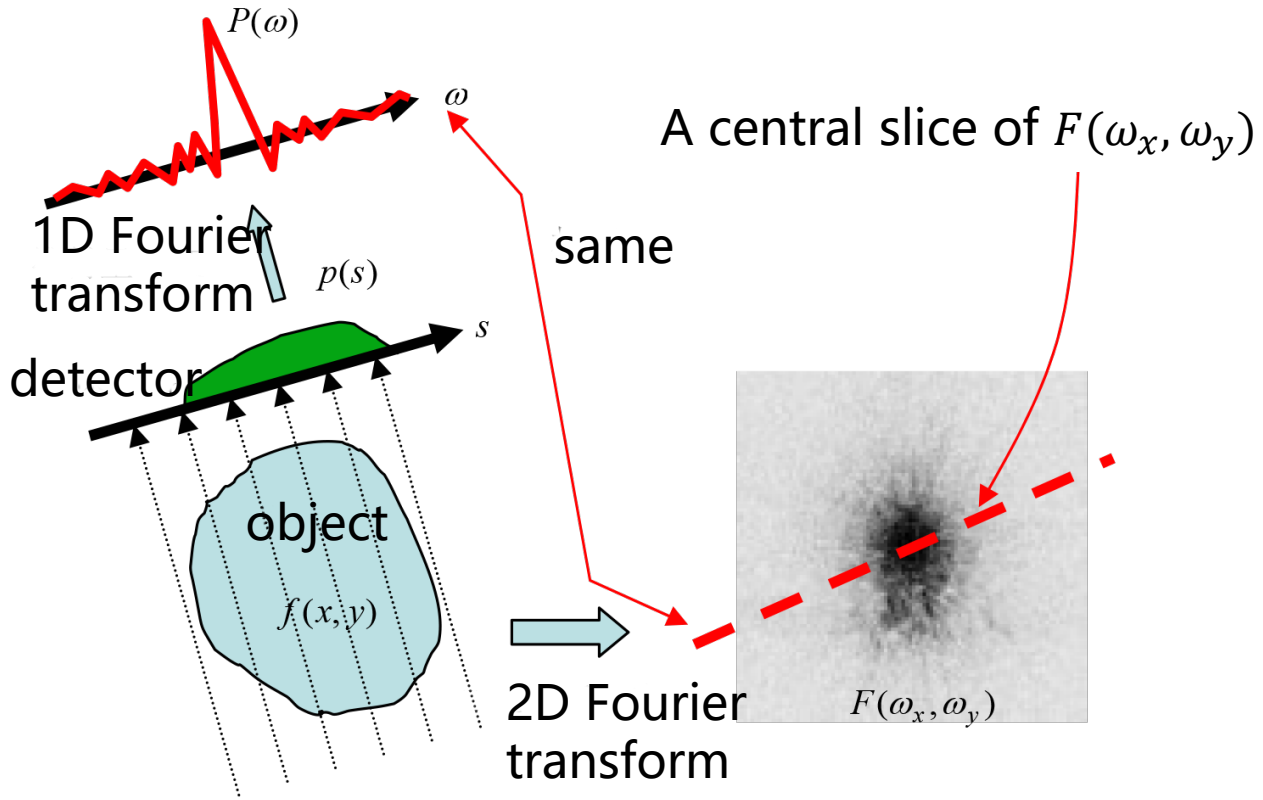
X-ray imaging – tomography reconstruction

Iterative reconstruction algorithms



$$\vec{x}^{下一个} = \vec{x}^{当前} - 反投影_{射线}\left\{\frac{投影_{射线}(\vec{x}^{当前}) - 测量值_{射线}}{归一化因子}\right\}$$

Analytical reconstruction algorithm(FBP)



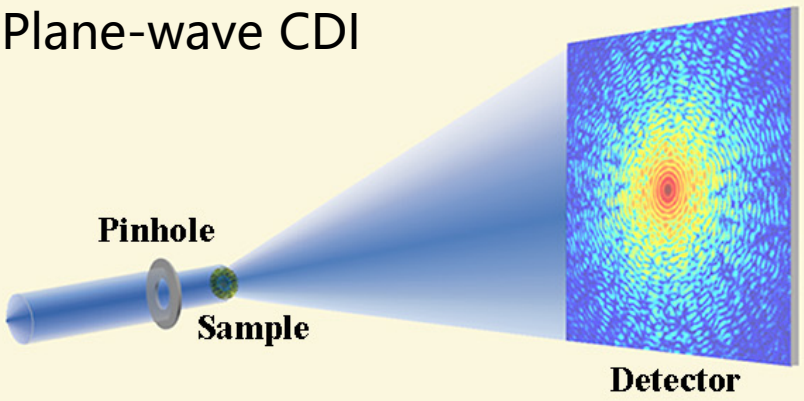
projection data → filtered data → reconstructed data

$$p(s,\theta) \xrightarrow{\text{ramp filter}} q(s,\theta) \xrightarrow{\text{back projection}} f(x,y)$$

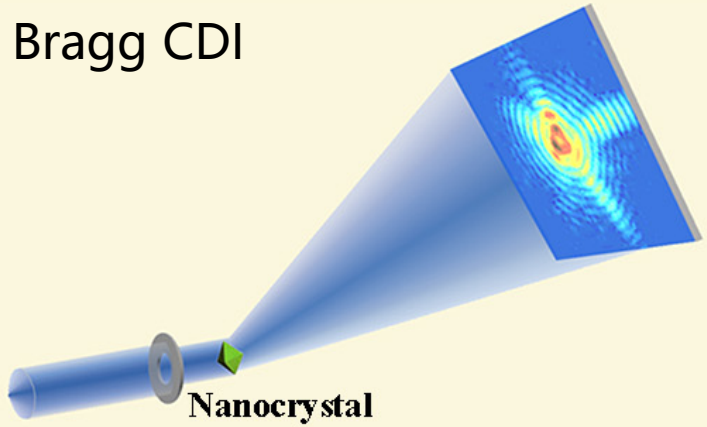
X-ray imaging – Coherent diffraction imaging

Reconstruct the structure of an object by analyzing the diffraction pattern of coherent light

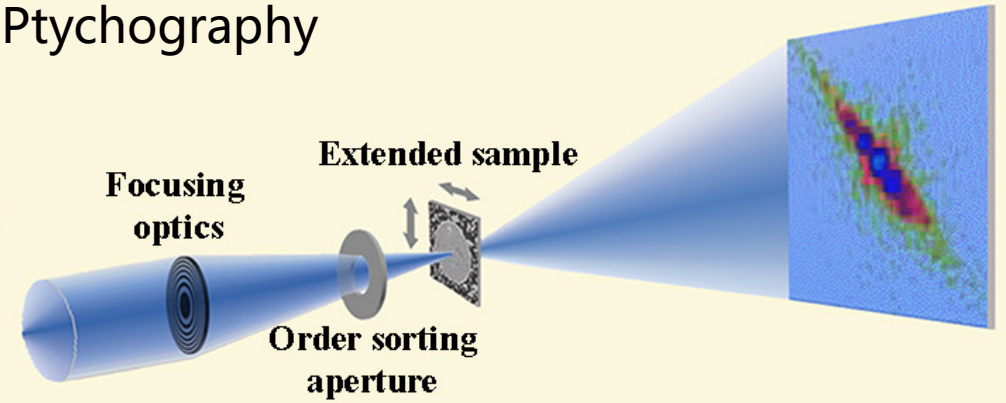
Plane-wave CDI



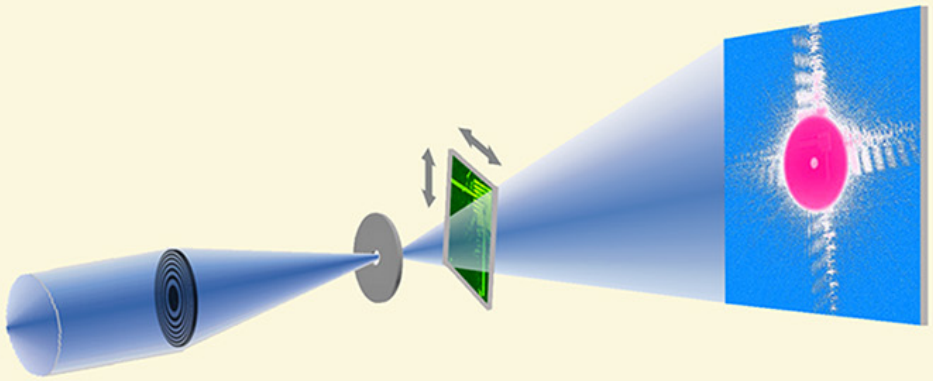
Bragg CDI



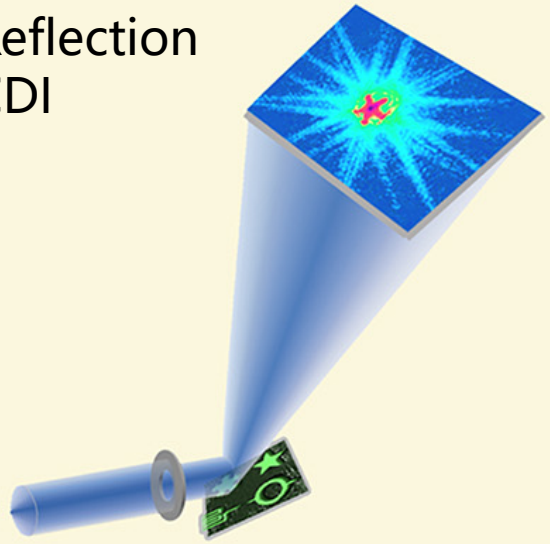
Ptychography

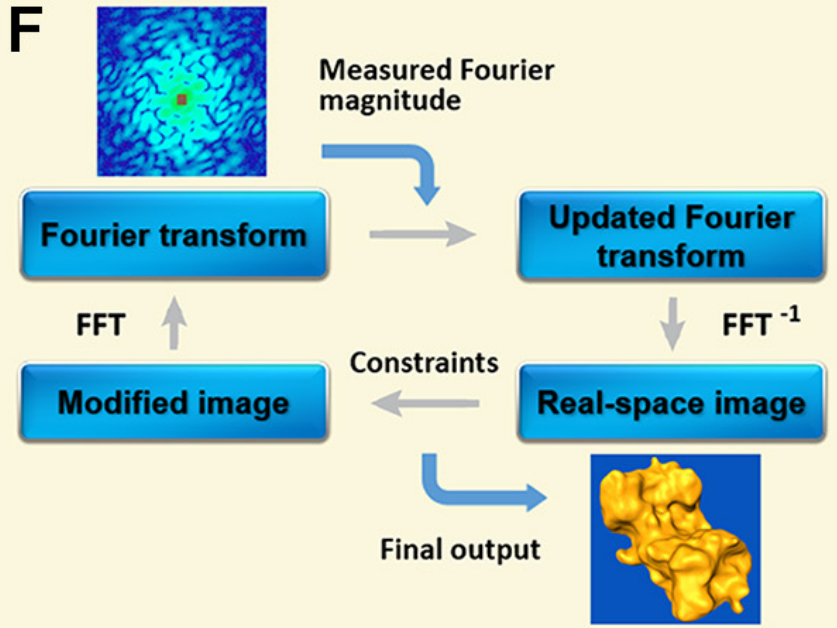


Fresnel CDI

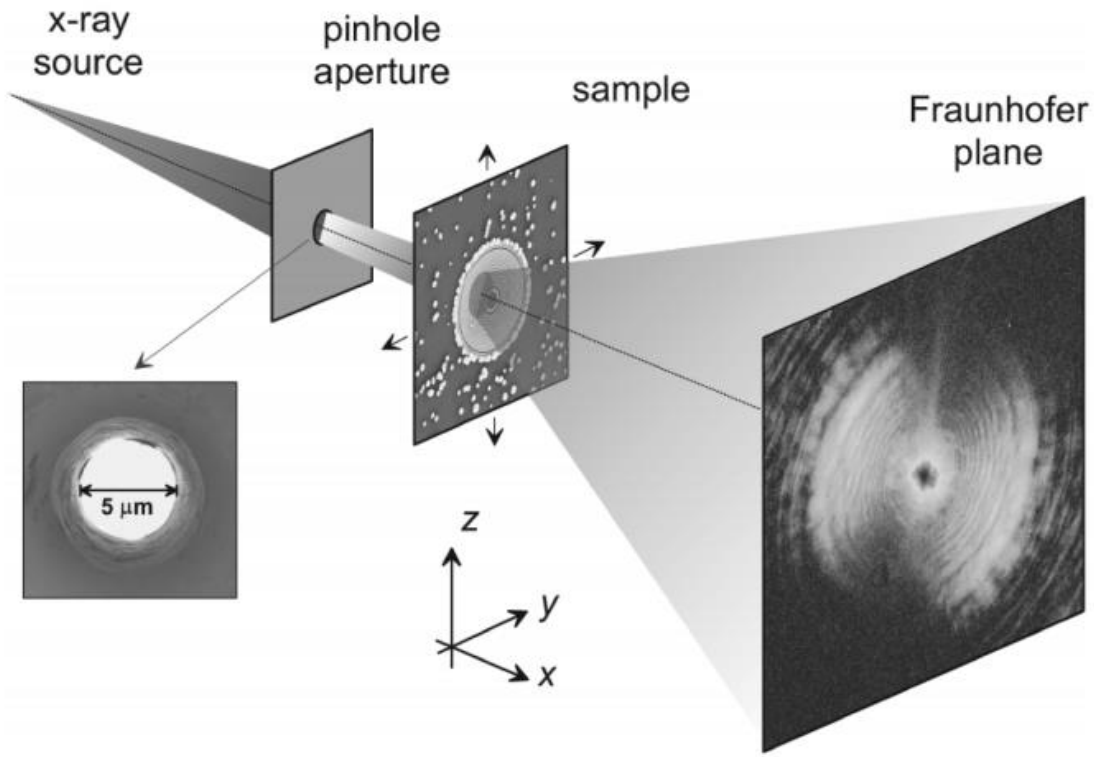


Reflection CDI





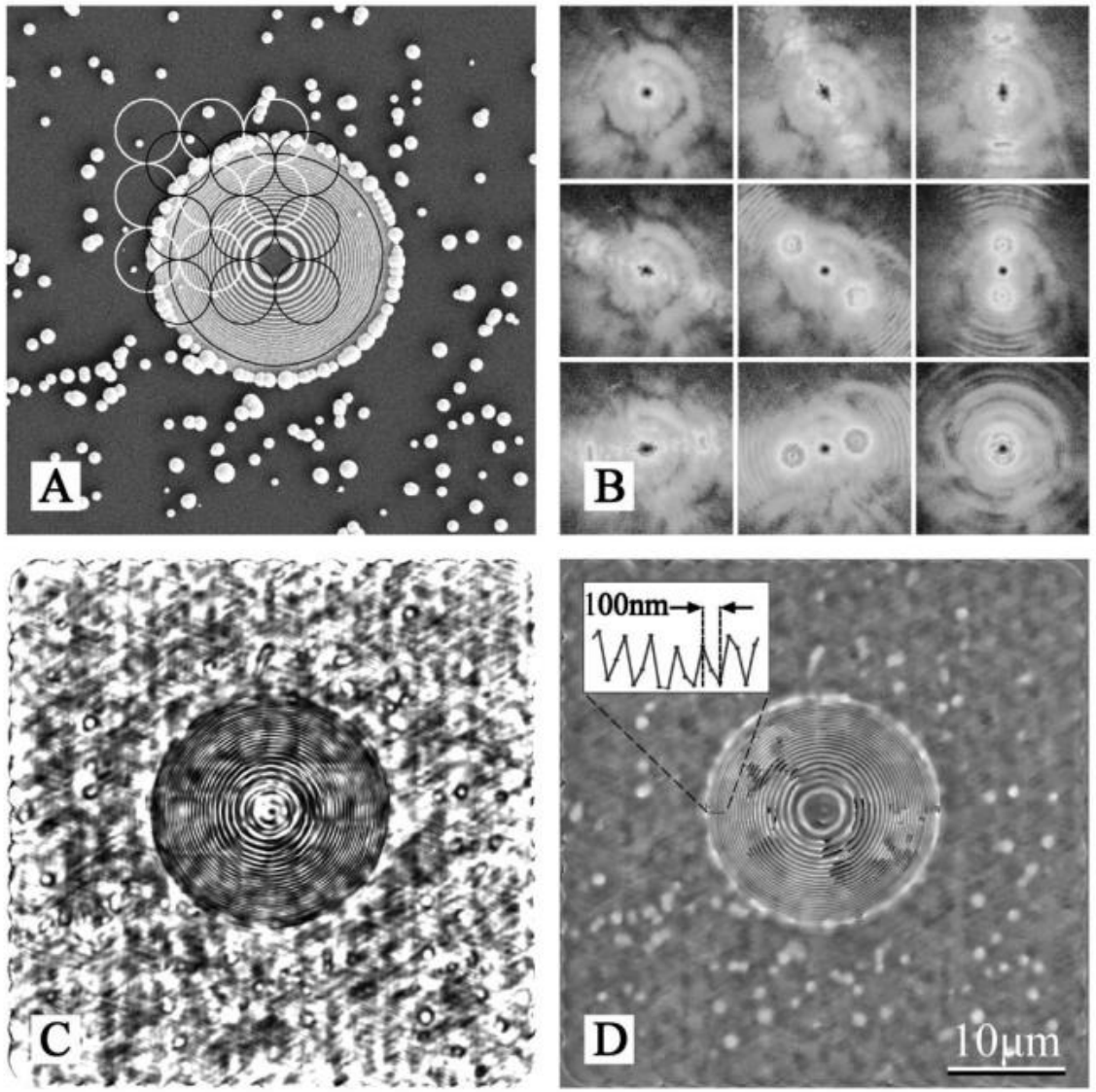
X-ray imaging – Ptychography



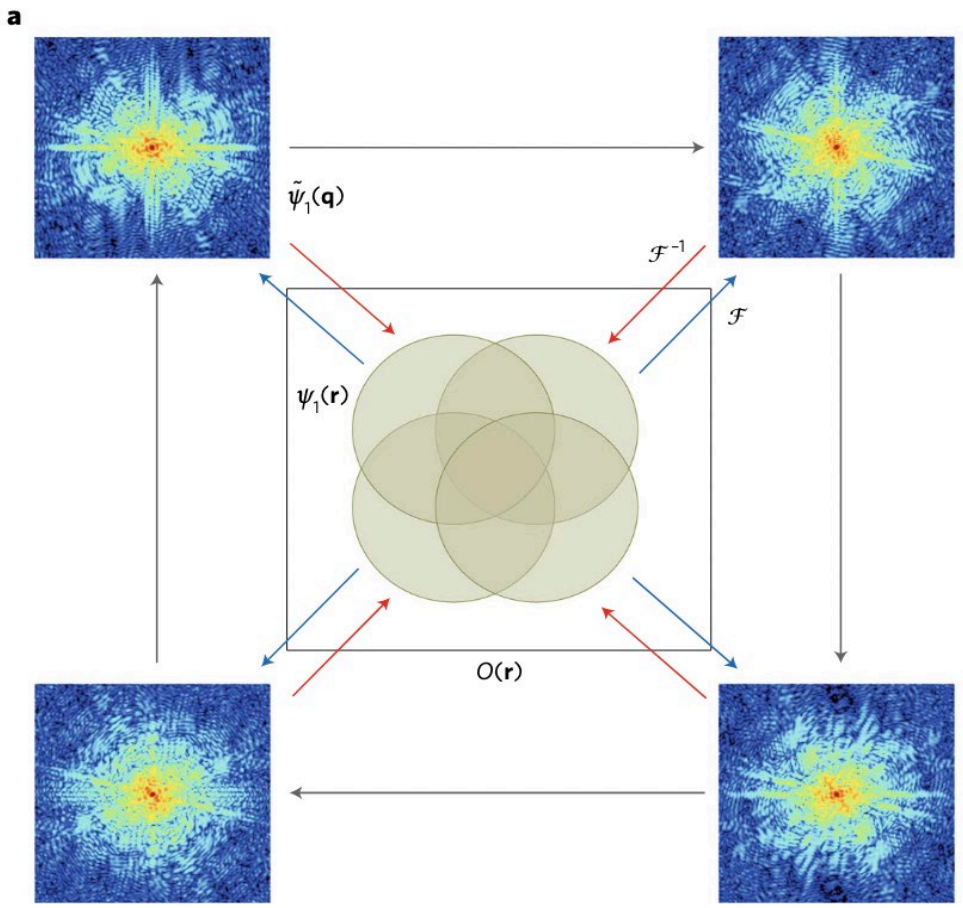
对物体进行区域重叠的空间扫描

允许以纳米级空间分辨恢复物体的结构

从一系列部分重叠的区域收集衍射图样

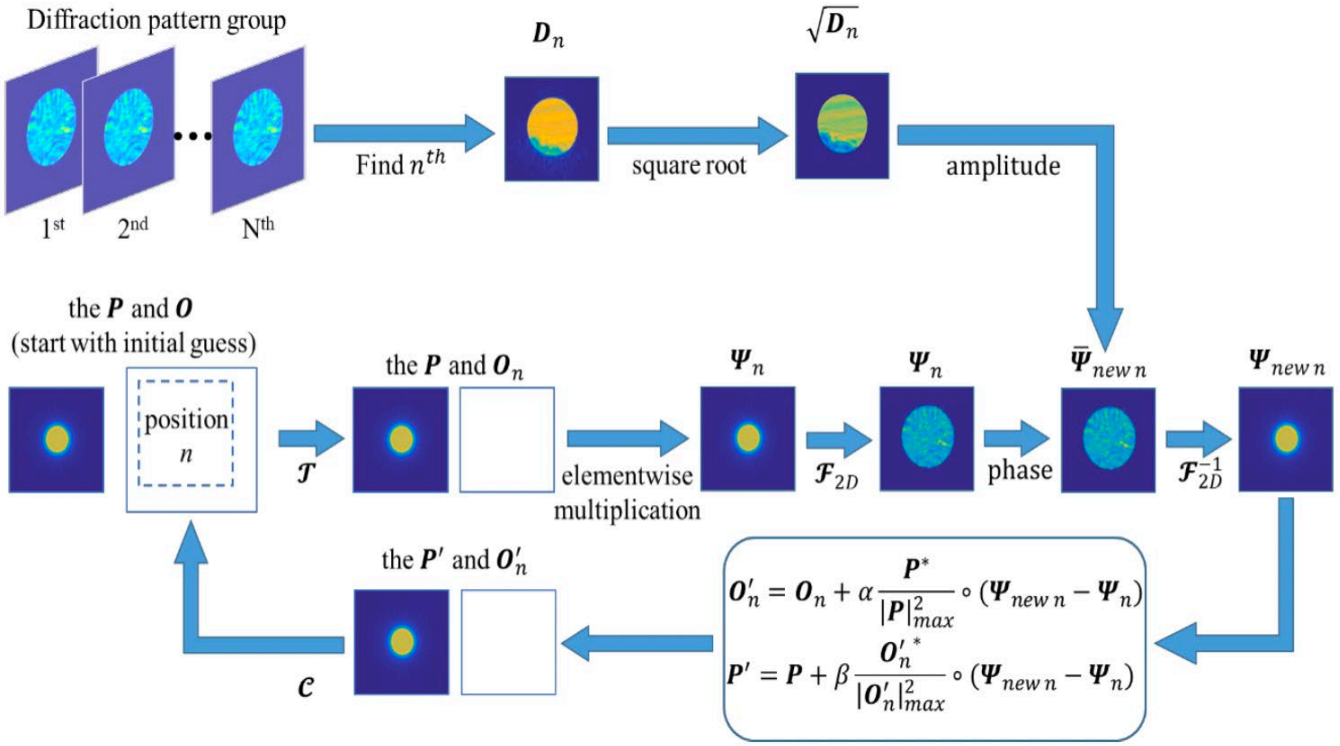


X-ray imaging – Ptychography



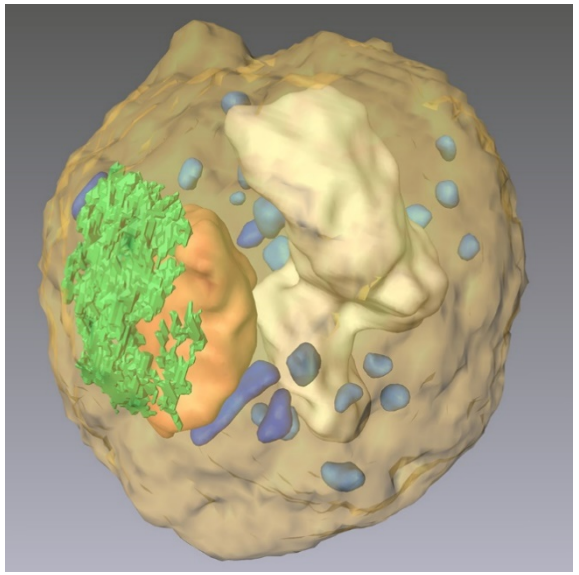
Extended Ptychographical iterative engine（ePIE）

- Probe function *P*, object function *O* 均未知
- 每一步迭代同时更新 *P(r),O(r)*



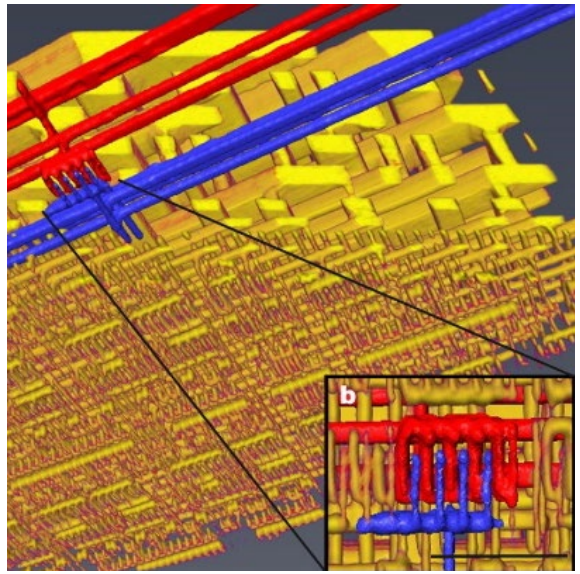
Applications of x-ray imaging

生物学



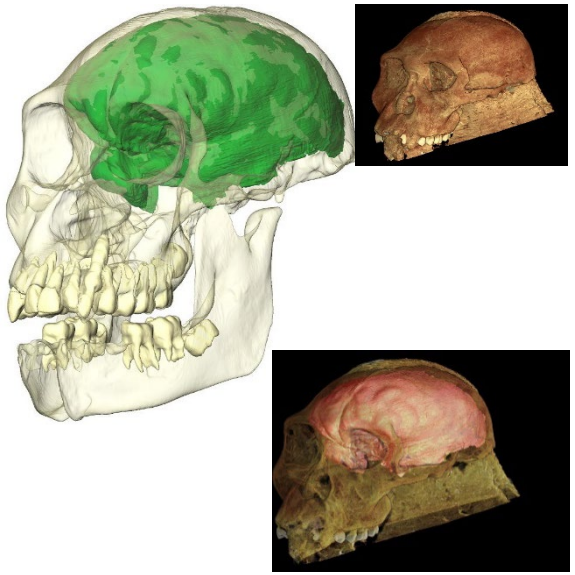
CDI显示酵母孢子细胞内的三维细胞器, 包括 细胞核，内质网，液泡，线粒体，和颗粒

集成电路



15纳米分辨率的集成电路，显示具有最精细结构的有源层视图

古生物学



南方古猿sediba的大脑结构，一些学者认为是现代人类真正祖先的新物种

材料科学

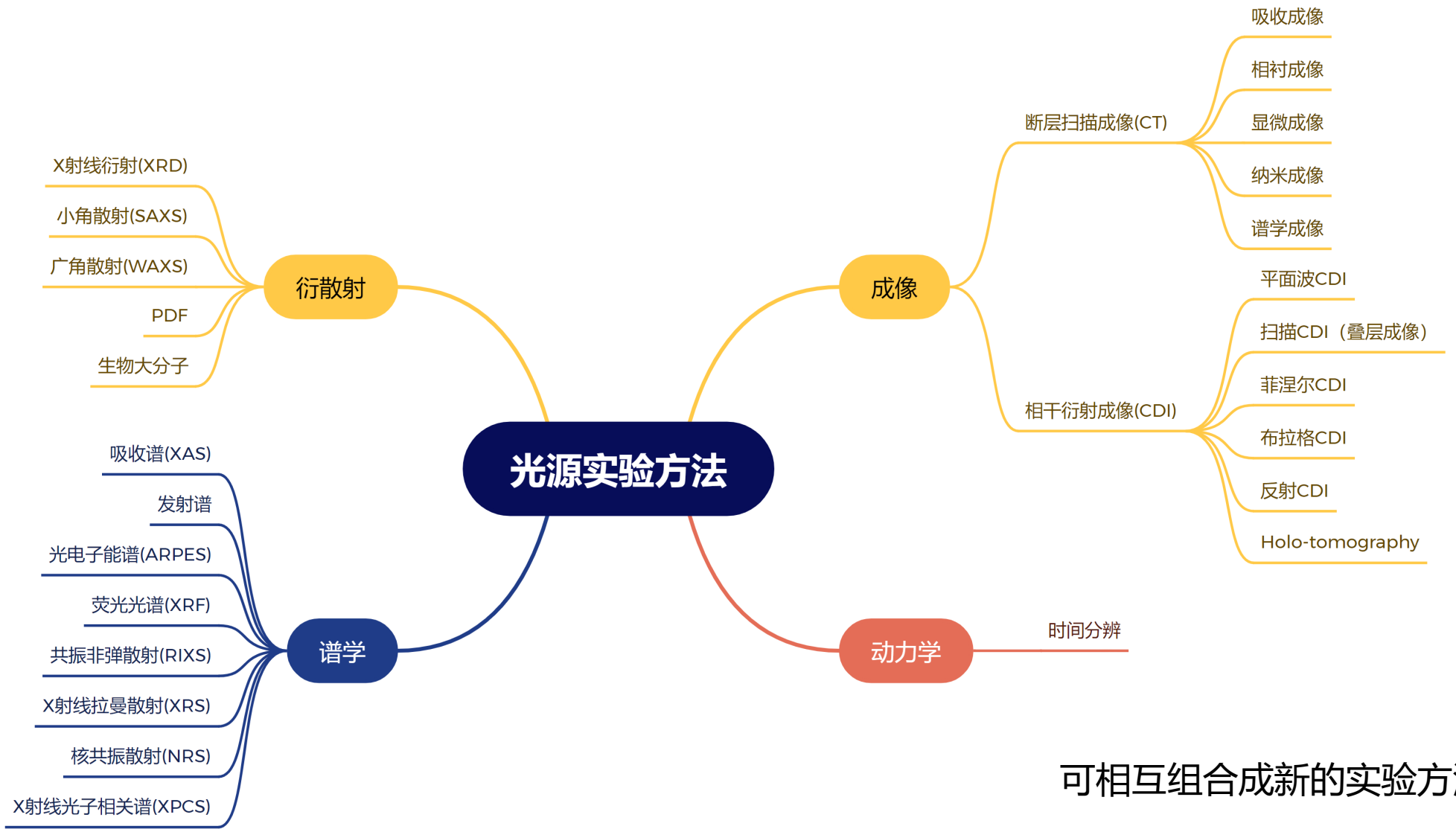
纳米科学

考古学

神经科学

.....

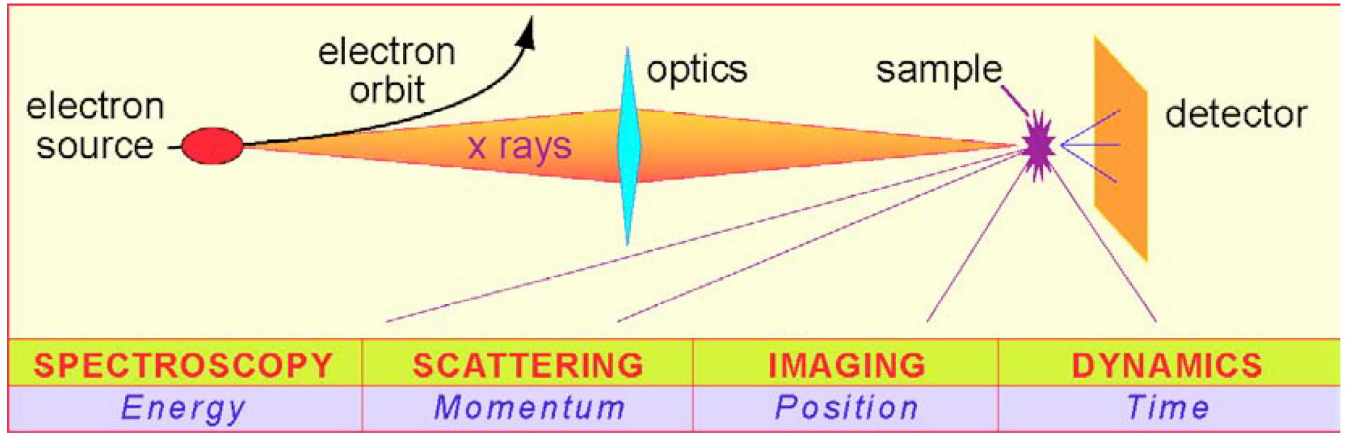
Experimental methods on light sources

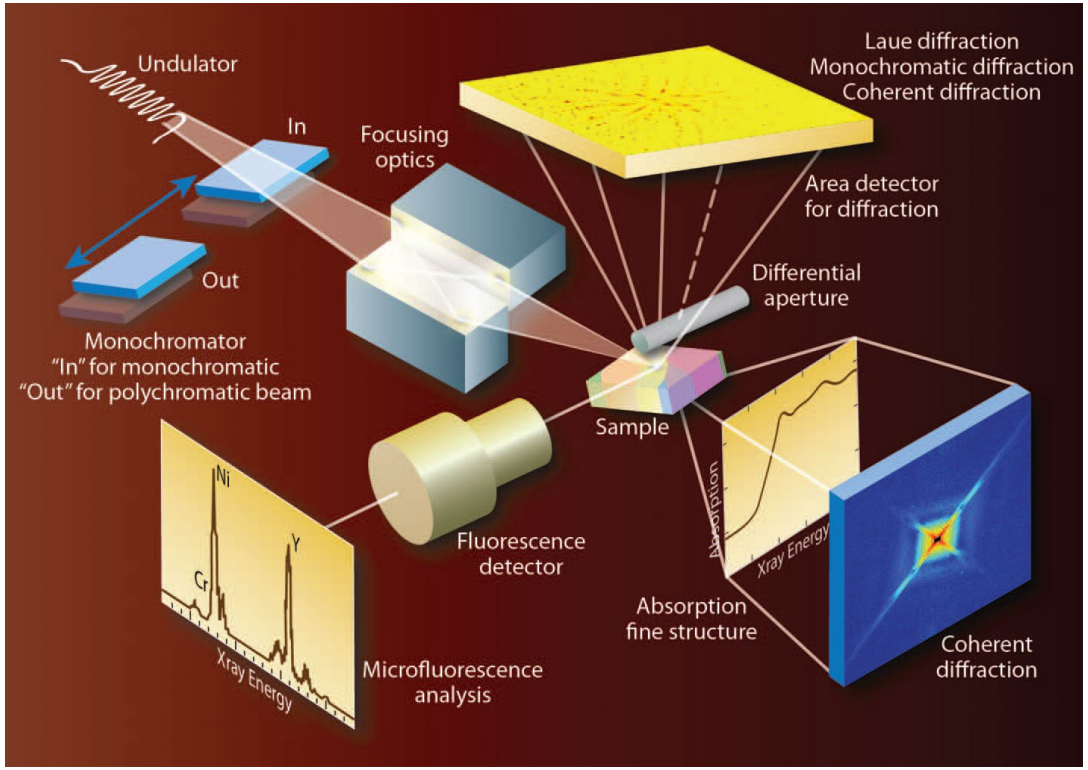


可相互组合成新的实验方法

Data Challenges @ next gen. light sources

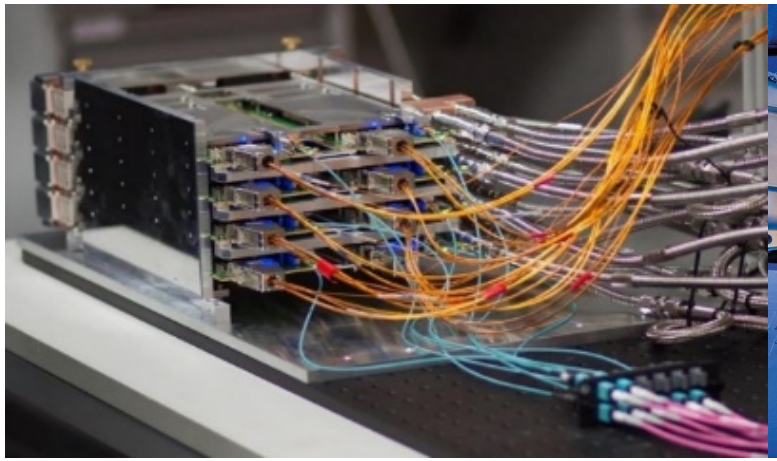
- 学科多样性，实验方法繁多，谱学、衍散射、成像 ……
- X射线光源技术的发展，不断催生出更加复杂的新方法、新技术以及新研究领域，需要新的学科软件及算法
  - 多模态实验需要结合多个样本、技术和设备的数据
  - 原位和动态加载实验需要实时反馈和自主控制
- 不同的光束线站以及科学目标，实验数据通量和容量存在巨大差异
- 源源不断的来自不同领域和背景的新用户

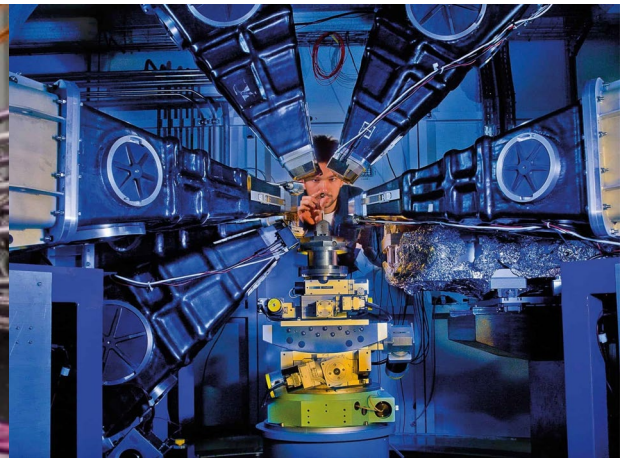


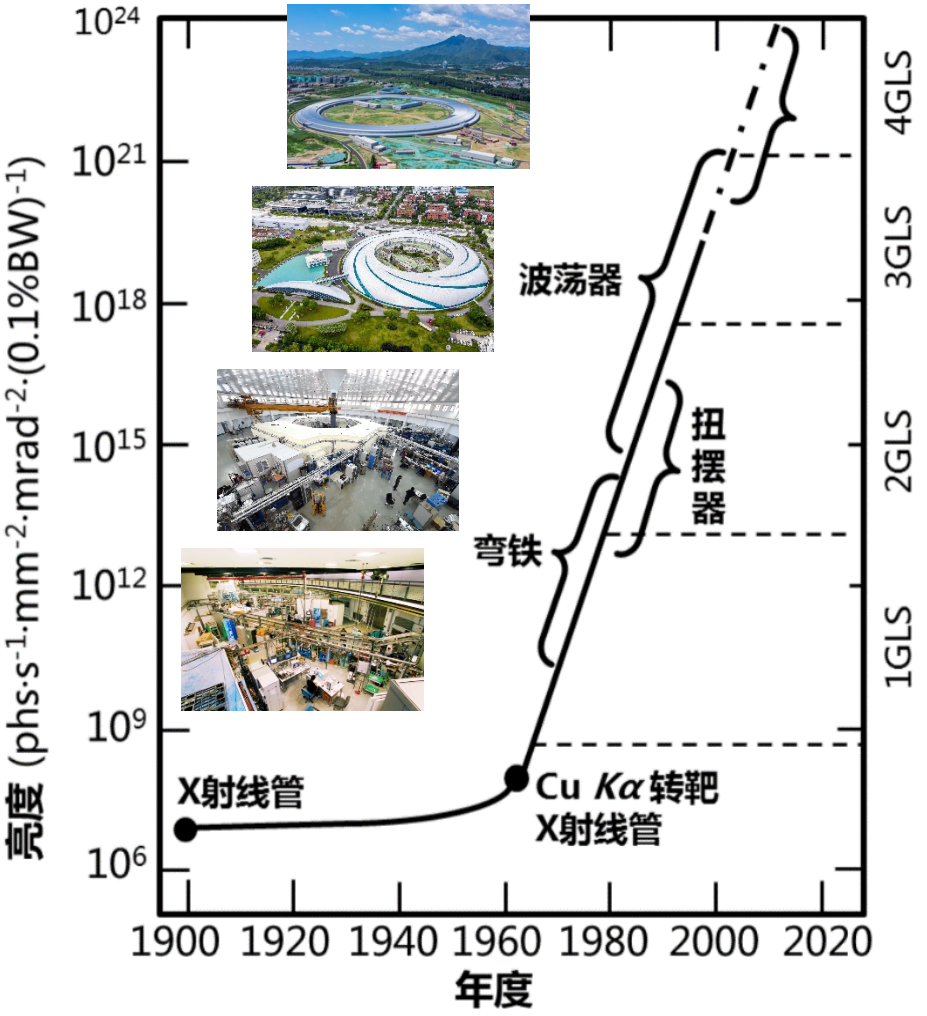


Data Challenges @ next gen. light sources

- **亮度相对三代光源提升了 2-3 个量级**
  - 在更短的时间内产生更加海量的具有更多细节信息的原始数据
- **X 射线探测器能力不断提高:**
  - 更宽的动态范围，更快的读出速率，更大的像素阵列（32bits，20KHz，28kx10k)
  - 更大的帧数，更高的帧率 => 更多的原始数据



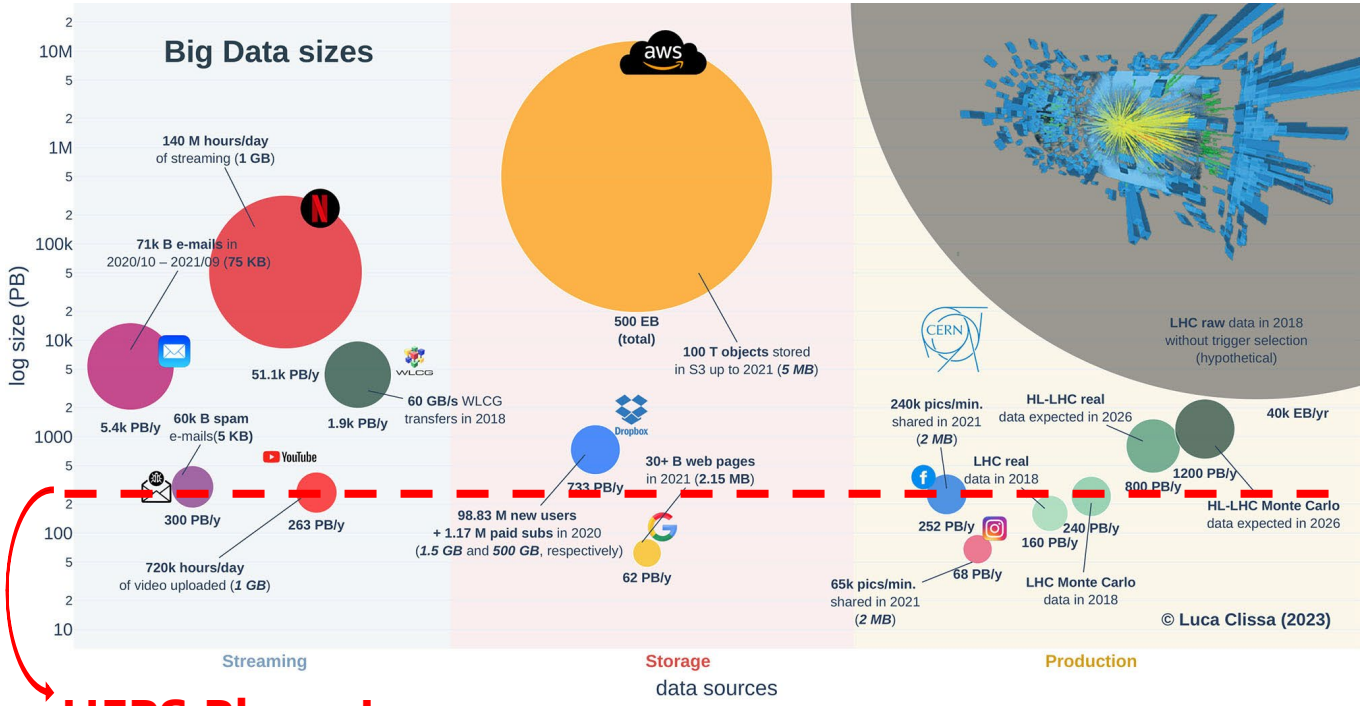




Development of synchrotron radiation light source

Data Challenges @ next gen. light sources

- HEPS 一期(15个线站)数据产生率接近PB/天，每年将产生超过200 PB的数据
- HEPS 总共能容纳超过90条线站，更多的数据
- 数据容量将很快达到 EB 量级



Data volume of HEPS Phase I Beamlines:

Beamlines	Burst output (TB/day)	Average output (TB/day)
Engineering Materials	600.00	200.00
Hard X-ray Multi-analytical Nanoprobe	500.00	200.00
Structural Dynamics	8.00	3.00
Hard X-ray Coherent Scattering	10.00	3.00
Hard X-ray High Energy Resolution Spec.	10.00	1.00
High Pressure	2.00	1.00
Hard X-Ray Imaging	1000.00	250.00
X-ray Absorption Spectroscopy	80.00	10.00
Low-Dimension Structure Probe	20.00	5.00
Biological Macromolecule Microfocus	35.00	10.00
pink SAXS	400.00	50.00
High Res. Nanoscale Elec. Struc. Spec.	1.00	0.20
Tender X-ray beamline	10.00	1.00
Transmission X-ray Microscope	25.00	11.20
Test beamline	1000.00	60.00
Total average:		**805**

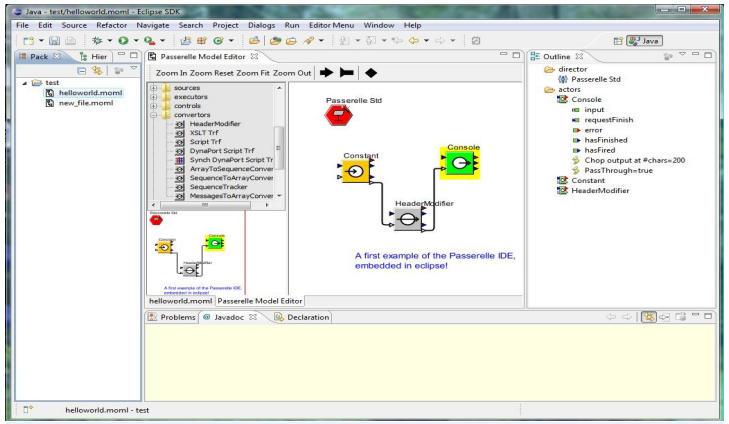
35

Data Challenges @ next gen. light sources

- 先进光源上大规模科学数据的处理和管理变得越来越具有挑战性
- 需要发展和集成先进的分析和管理工具
  - 提供海量科学数据的存储、组织和管理
  - 为方法学软件和算法的多样化发展提供通用的底层软件框架支持
  - 在实验过程中，进行实时数据分析和快速反馈，提供决策指导和修正实验过程，并优化数据采集
  - 在实验结束后，处理海量多模态数据，帮助用户快速完成实验数据分析、获取科研成果，加速科学发现
  - 提供可伸缩的分布式异构算力支持，满足不同科学目标不同规模的计算分析需求
  - 满足不同技术水平的用户需求

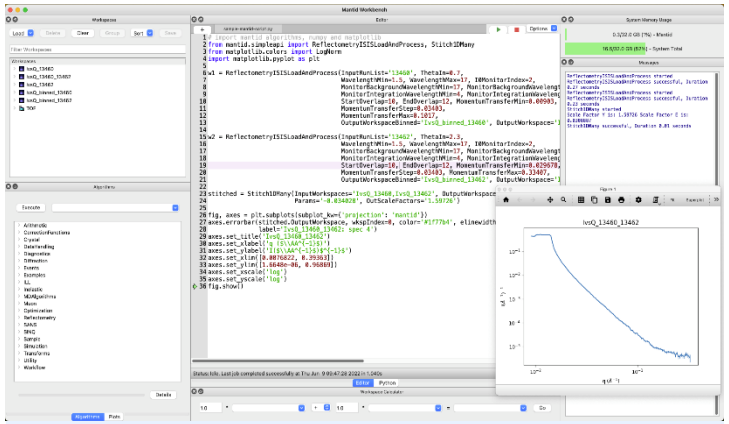
光源数据处理软件发展现状

- 全球先进光源装置升级换代，数据通量爆发性增长，新的实验技术不断涌现，数据处理方面面临着巨大挑战
- 经过十几年的积累，形成了：EDNA（美国）、Mantid（英国）、DAWN（英国）等框架软件及众多方法学软件



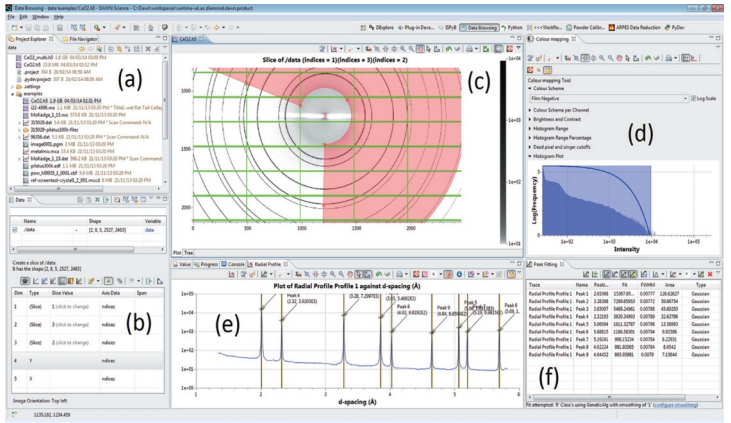
EDNA: ESRF

- Python + Java
- 架构基于Plugin
- 数据模型通过UML语言定义
- 框架不够完善，学习曲线陡峭
- 高通量解决方案比较落后



Mantid: ISIS, SNS, ESS

- Python + C
- 成熟的软件框架和用户界面
- 数据对象实现较为复杂
- 算法依赖于复杂的数据对象
- 没有高通量解决方案

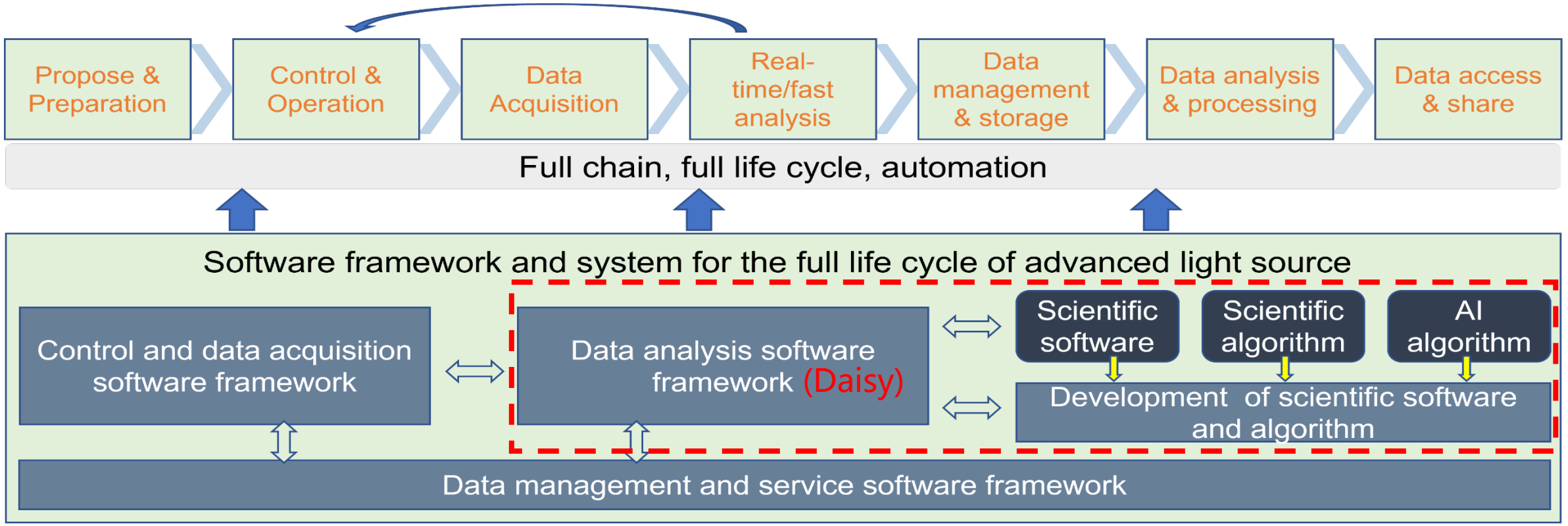


DAWN: Diamond

- JAVA + Eclipse-RCP + SWT
- 相对成熟的软件框架和用户界面
- 开发难度大，需要较大的团队
- 与未来主流基于Python的数据处理模块接口研制困难

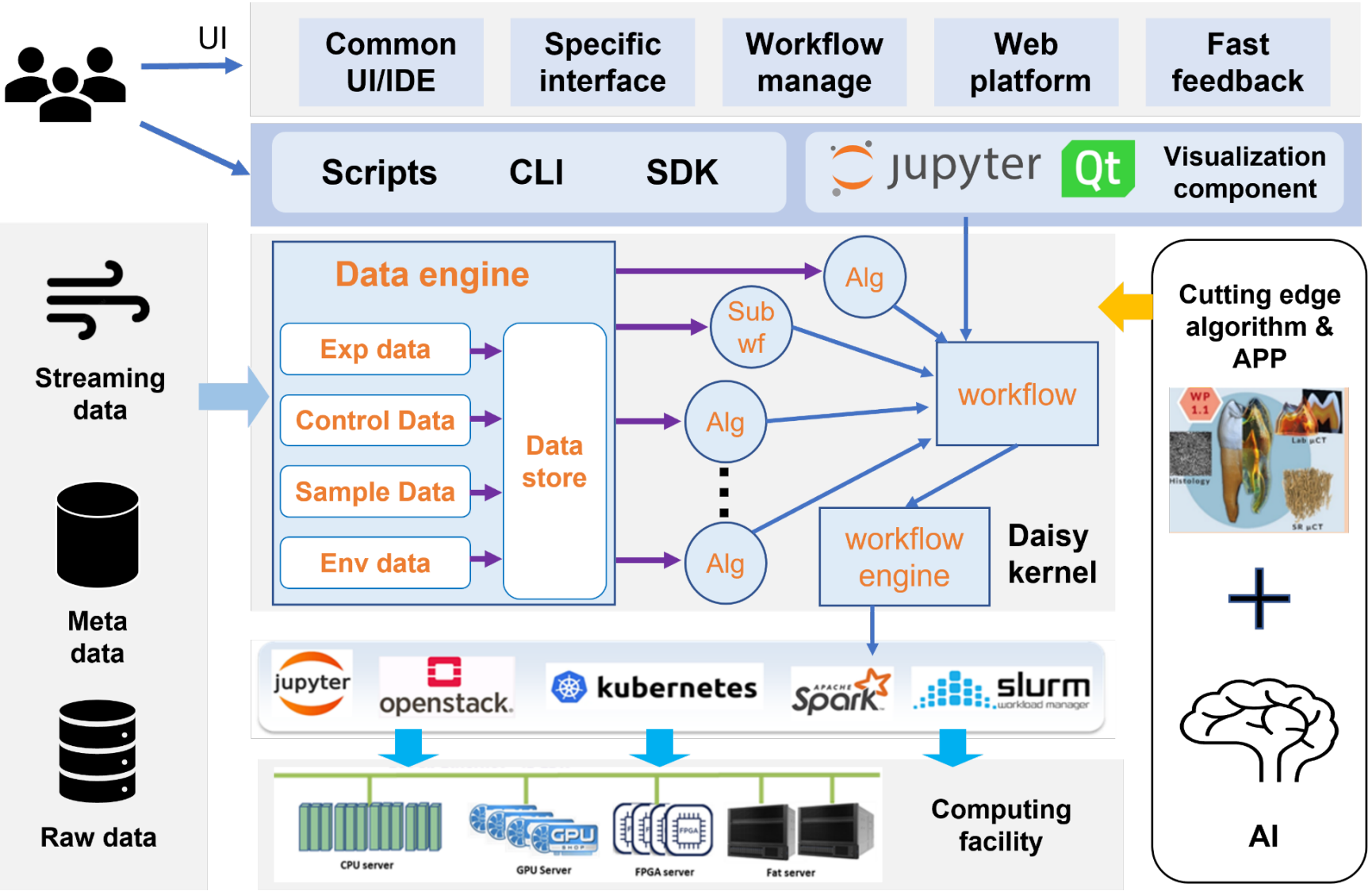
无法满足先进光源快反馈和超高通量的数据处理的需求

先进光源全生命周期的软件系统



支持先进光源科学数据全生命周期的跟踪和管理

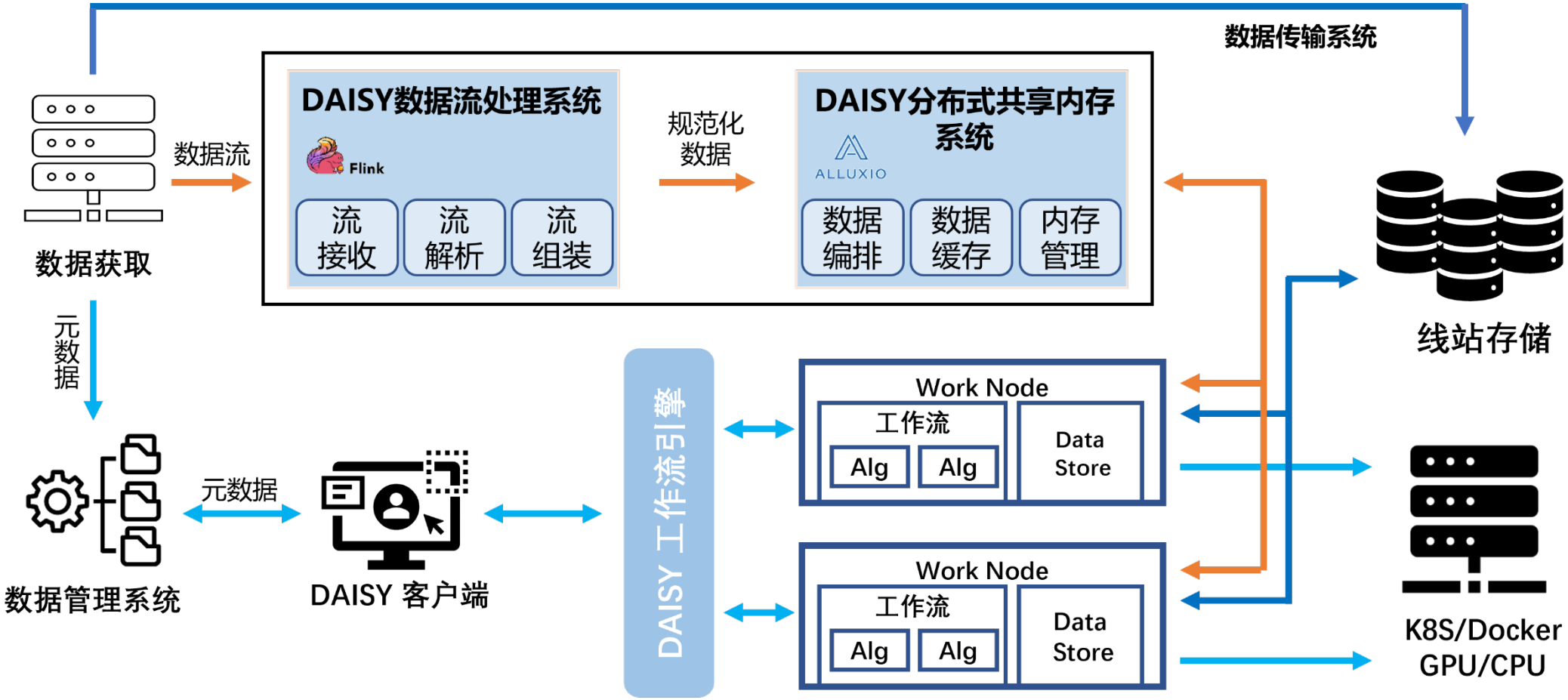
数据处理软件框架总体架构（Daisy）



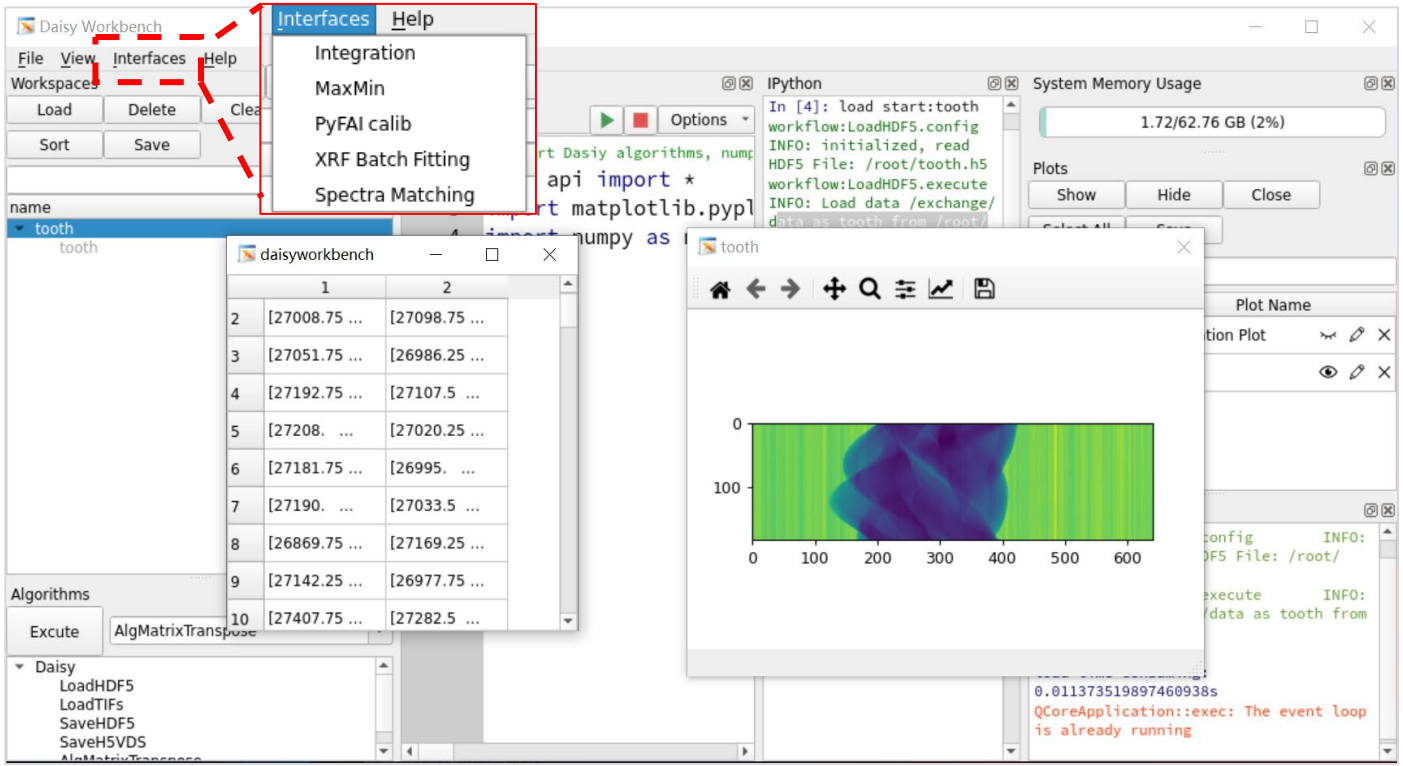
- 数据处理软件框架核心
- 满足新一代光源数据处理需求的衍生技术模块
  - 应对高通量数据I/O、多模态数据解析、多源数据接入的数据对象管理
  - 应对不同规模、不同通量、低延迟数据处理需求的弹性异构计算集群算力支持
  - 服务于学科方法学软件集成和发展的用户软件接口和软件开发环境支持
- 基于软件框架的学科专用应用软件以及针对灵活数据处理需求的通用工作流编排系统

Daisy 数据处理流程

- 支持流处理和批处理
- 支持交互式处理和自动化处理



Daisy 图形化用户接口



Daisy workbench:

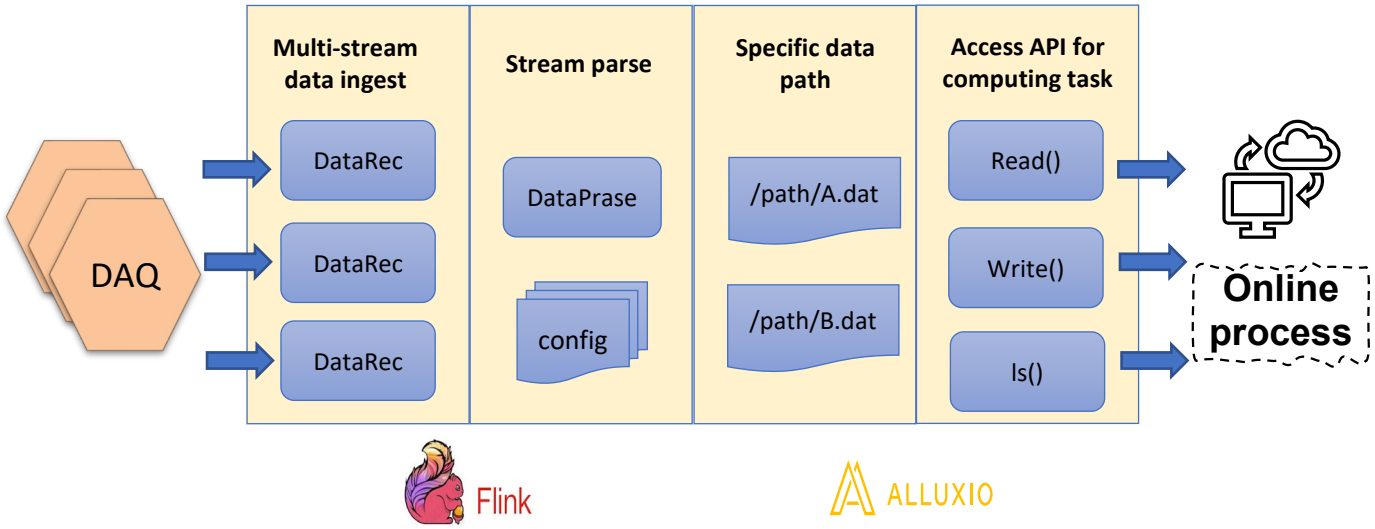
- 通用用户界面，基于 PyQt5
- 包含数据对象列表，算法列表，数据展示/可视化，Log信息，系统内存监控，以及提供给开发者的 IDE 等模块
- 面向多样化学科方法学的专用用户界面接口

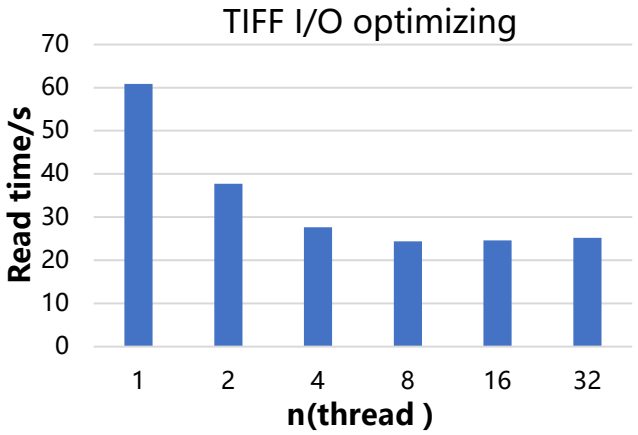
Web data analysis platform:

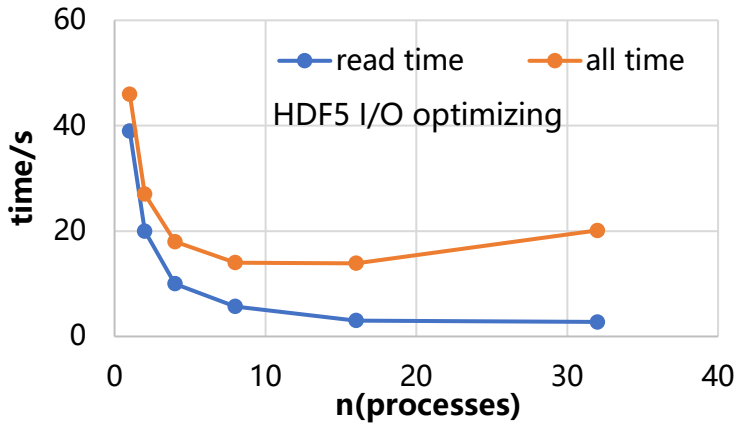
- 基于 Jupyterlab 生态，开箱即用
- 利用容器技术封装开发和运行时环境
- 通过K8s提供弹性可伸缩的计算资源
- 终端+用户友好界面，适合不同专业程度的用户

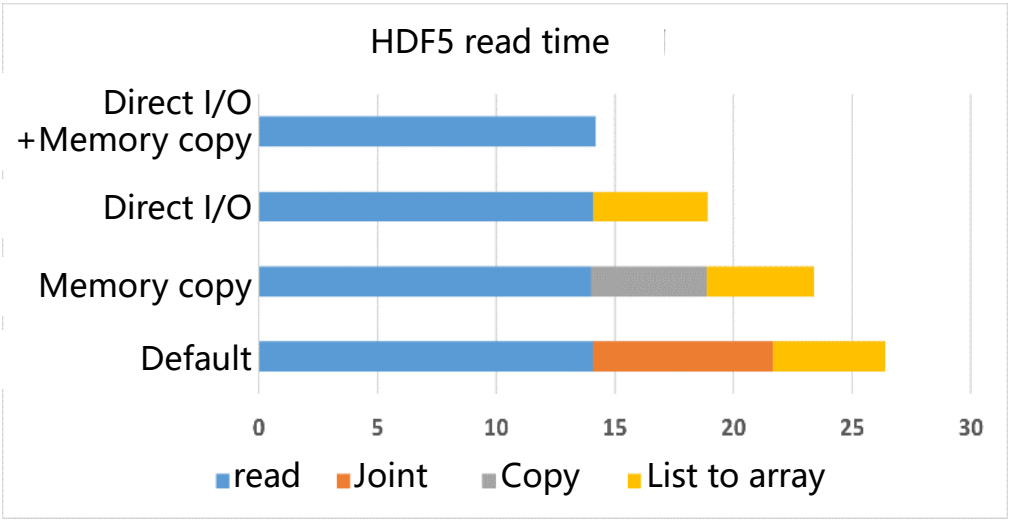
Daisy I/O module

- 提供统一的批处理 I/O 接口，屏蔽底层体系结构和数据结构的差异
- 采用多种方法对数据 I/O 进行优化
  - HDF5文件：多进程并行读写，内存拷贝，异步I/O
  - TIFF文件：多线程并行读写，异步I/O
- 基于流处理、分布式共享内存的流数据处理方案



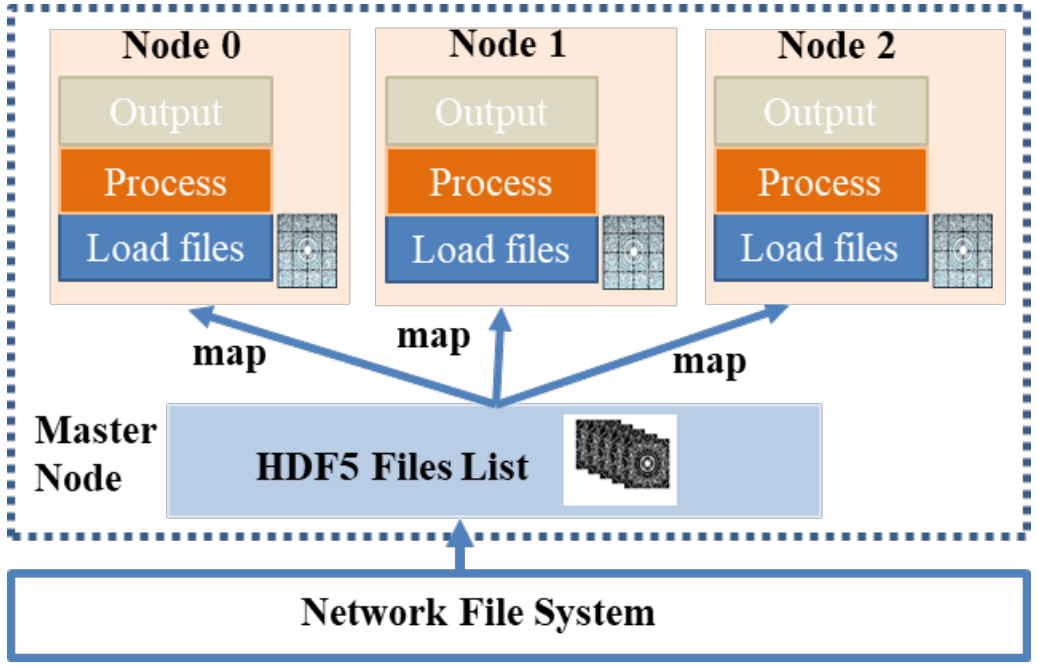




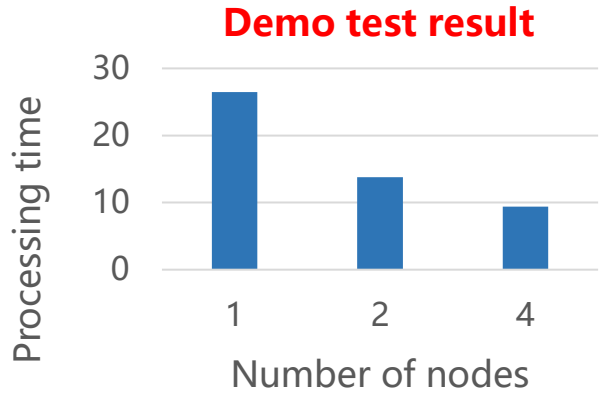


Daisy distributed computing engine

- HEPS成像实验的单个数据集将达到 TB 量级
- 实验用户希望数据处理时间少于数据采集时间，单机无法满足需求
- 借助分布式技术向多机多卡进行扩展，支持分布式异构算力
- 在 CT 重建任务中，加速效果明显，采集时间内重建6k相机数据
- 为计算模型提供统一的编程接口API，降低并行编程的复杂性



Mode	Detector pixel	Projections number	Data rate	Dataset (TB)	Acquisition time	Daily data (TB/d)	Annual data (PB/y)
Powder CT	6k×6k	6k	1.08 GB/s	0.432	6.3 min.	78	9.4
High voxel CT	28kx10k	28k	1.1 GB/s	15.68	240 min.	87	10.4
Fast CT	5k×4k	5k	1.7 GB/s	0.1	1 min.	98	5.9

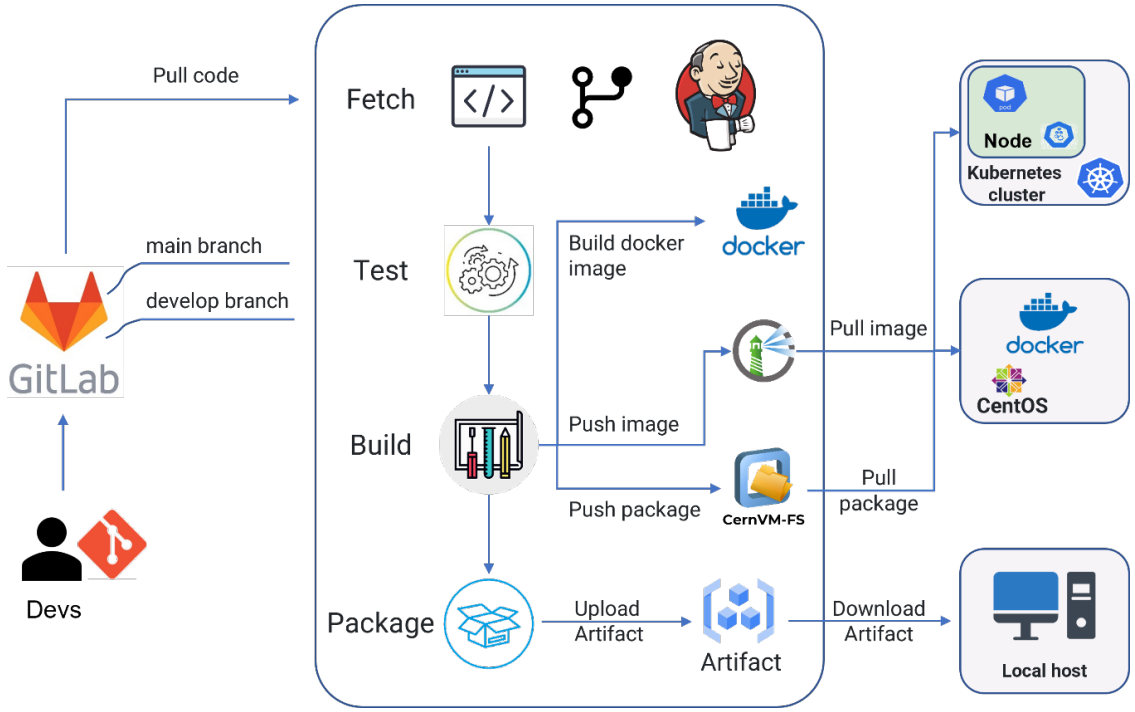


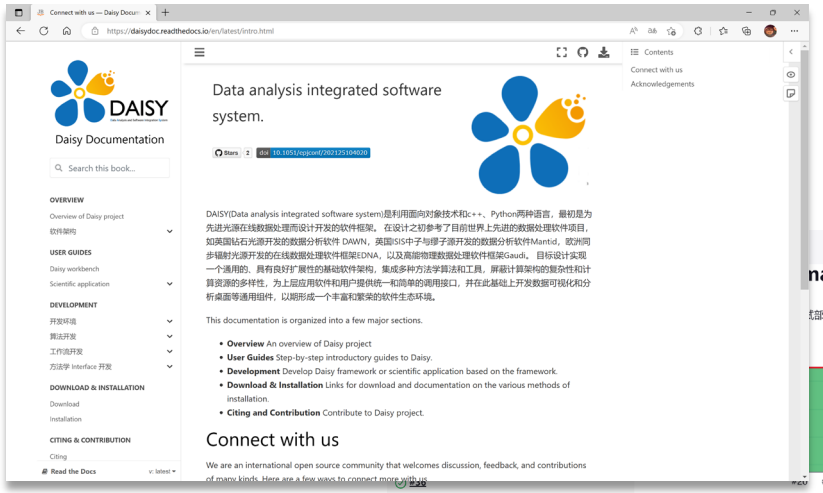
User and software ecosystem support

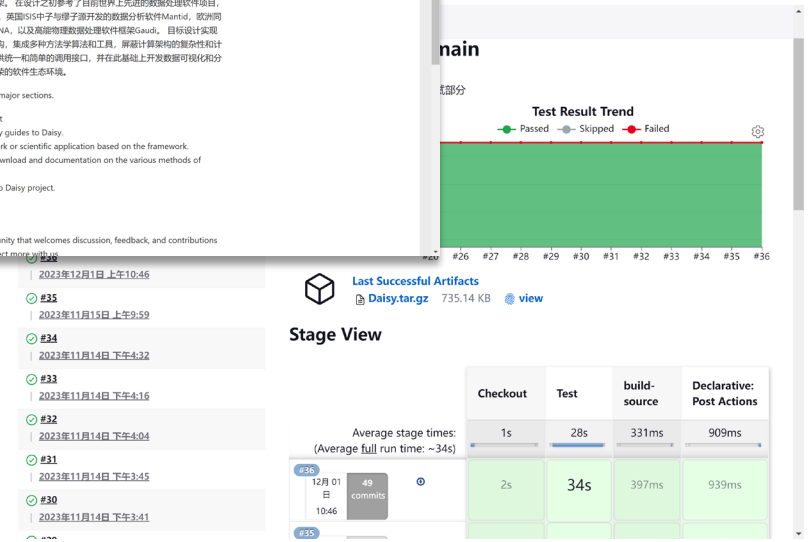
软件开发可持续集成，可持续交付，可持续部署系统

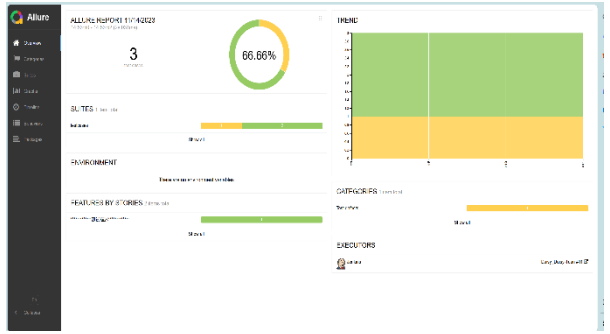
- 自动化软件开发服务 CI/CD 平台，实现多开发者代码集成—>代码测试—>代码编译—>软件部署全流程自动化
- 简化并加快软件开发生命周期

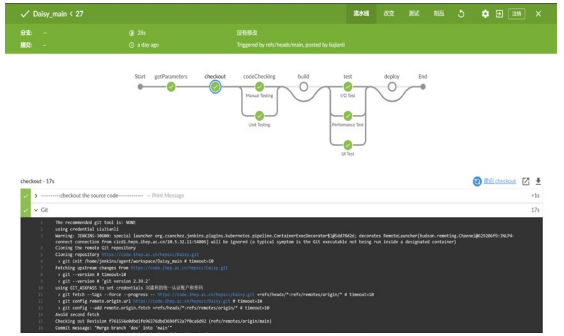
用户使用文档，开发者开发指南











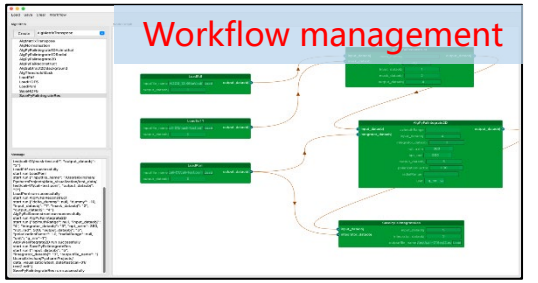
Daisy applications for synchrotron radiation

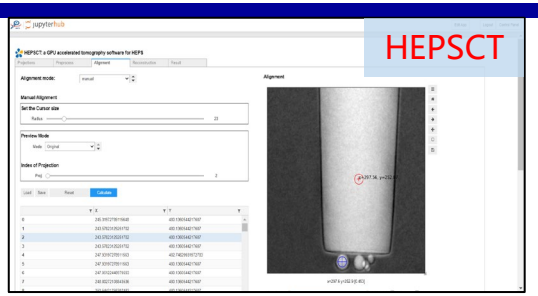
覆盖成像、衍散射、谱学多个学科：

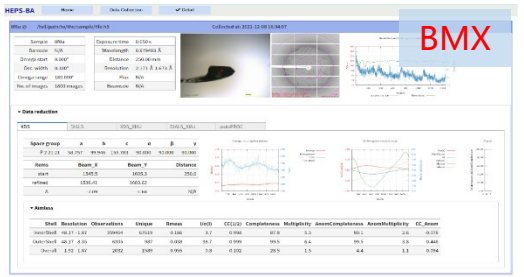
- CT 数据处理软件 HEPSCT
- 叠层成像相位恢复软件 Hepsptycho
- 衍散射 PDF 应用 Daisy-PDF
- 生物大分子结构解析应用 Daisy-BMX
- 吸收谱学谱线分析软件（谱学匹配XASMatch和PCA&LCF成分分析应用）
- 荧光光谱批处理软件 XRF-mapping
- Daisy 工作流管理系统

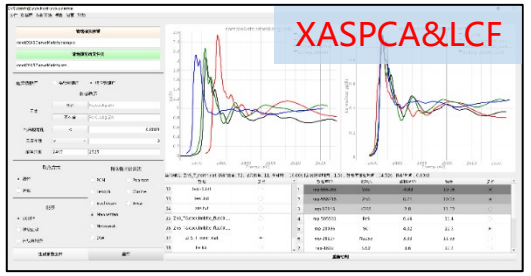
计划（正在）开发：

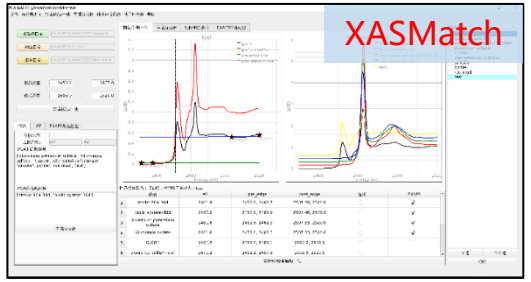
- 小角数据处理软件
- Holotomography
- XPCS
- Bragg CDI
- ……

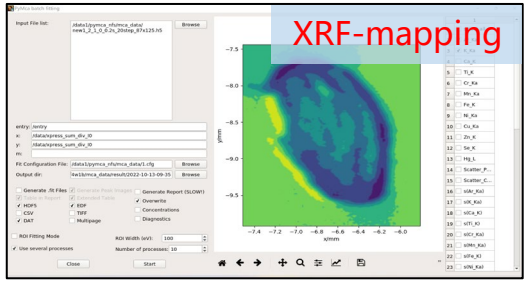


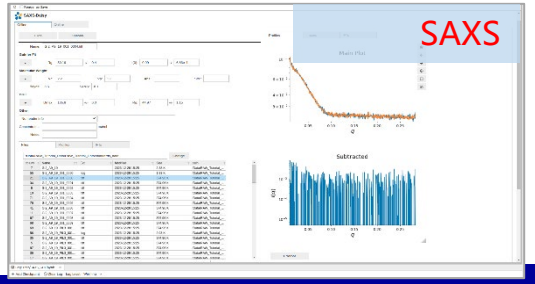


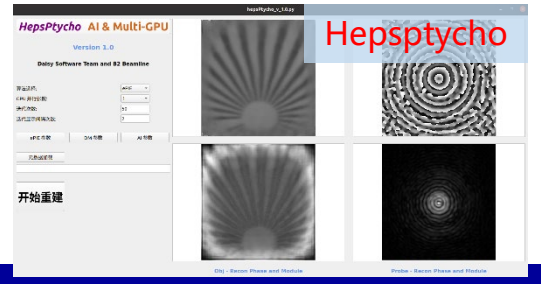








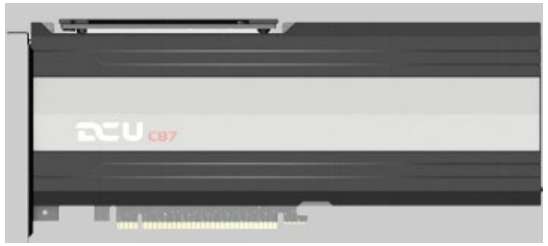




Heterogeneous computing support

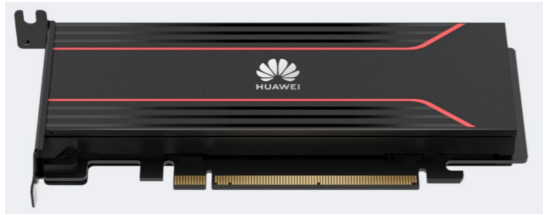
CT 数据处理软件适配

- 常用 GPU 加速 CT 软件 ASTRA，UFO，Tomocupy，能够借助曙光 DCU Z100 进行加速计算



叠层成像数据处理软件适配

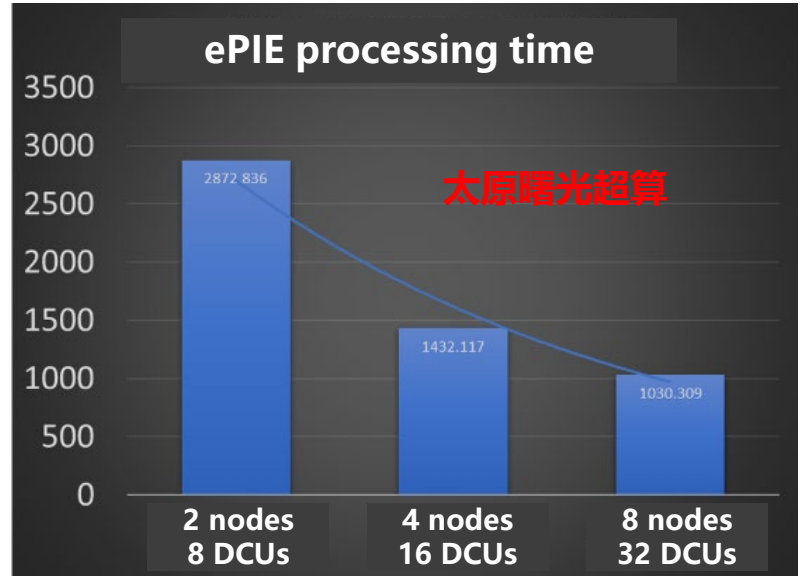
- ePIE、DM 等相位恢复算法可以在曙光 DCU Z100上运行
- 支持太原曙光超算集群大规模分布式并行



人工智能算法适配

- 自研AI算法 W1-Net，在曙光DCU Z100和华为昇腾910上均适配成功

软件	UFO	ASTRA	Tomocupy	ePie，DM	W1-Net
适配平台	曙光 DCU	曙光DCU	曙光DCU	曙光DCU	曙光DCU、华为昇腾910、天数智芯
性能	与 A100 相当	—	—	A100 25%	曙光是A100 66%，华为GPU表现较差



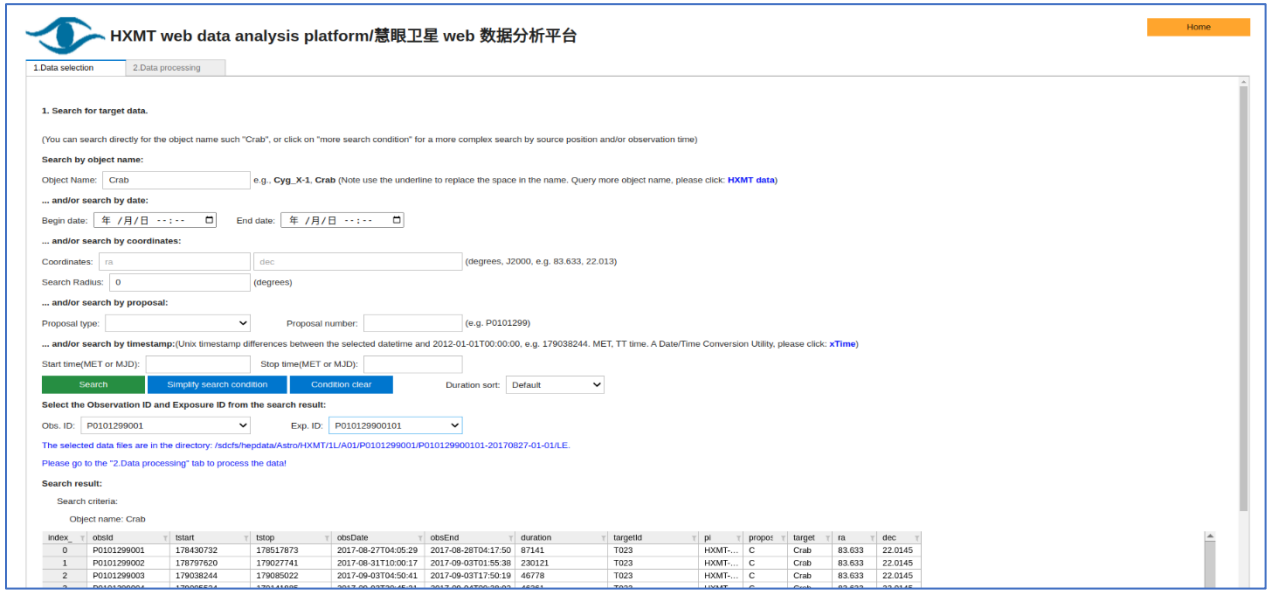
Application of Daisy in space astronomy

可能的应用场景

- 数据处理，数据分析，数据产品生成
- 探测器模拟，观测模拟
- 整合现有软件资源，形成共性软件包

HXMT web 数据处理平台

- 基于 jupyterlab 的 HXMT web 数据分析平台
- 通过 web 浏览器为用户提供数据处理环境和服务



https://sdccompute.ihep.ac.cn/

Svom 数据产品生成软件集成

- 软件框架应用于 **Svom** 卫星二级数据产品生成
- fits 数据文件读写算法的集成，部分数据产品生成算法

eXTP 卫星数据处理软件集成

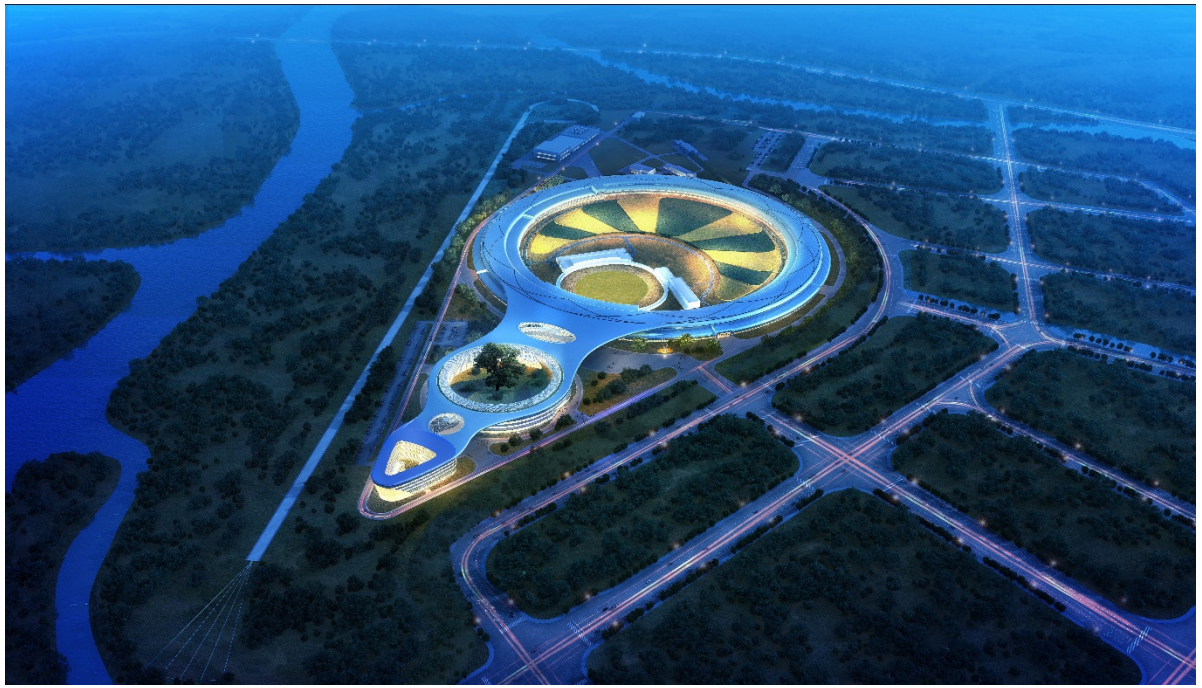
- 数据模板生成，数据分割等算法的集成

总结

- 同步辐射光源的实验变得更快、更复杂、更多样化
- 新一代光源上的X射线实验能够在短时间内采集到海量数据
- 光源上不同实验方法的数据处理需求存在差异，且新的实验方法不断涌现
- 急需基础软件框架提供底层共性支持，并开发先进的软件和算法

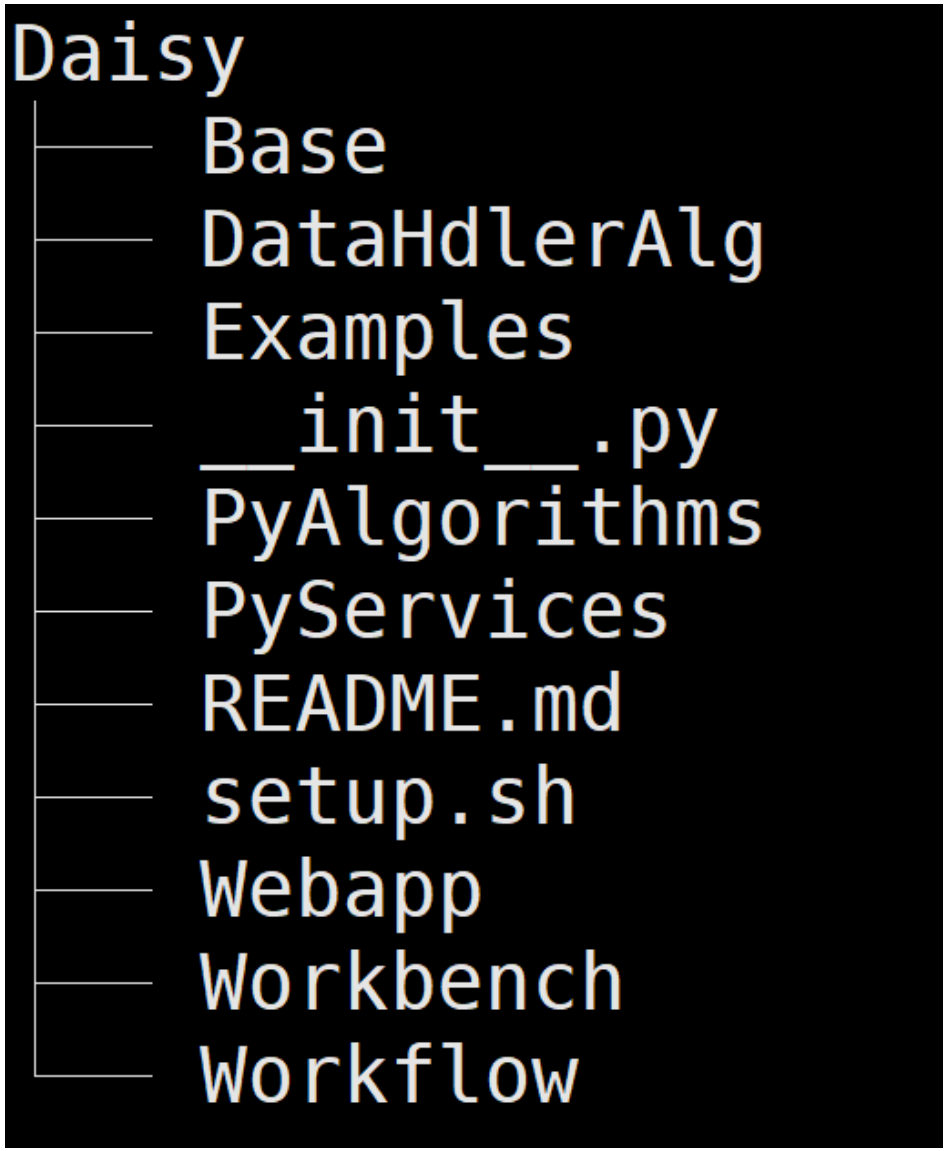
https://daisy.ihep.ac.cn/

谢谢！



Backup

Daisy代码组织结构



- **Base**：Daisy 的核心基类，包括data store, algorithm, workflow，workflow engine以及service等
- **PyAlgorithms**：领域方法学模型即数据处理逻辑的具体实现，如积分算法，转轴矫正算法等。不同学科的代码将有更细致的组织
- **Workflow**: 用户完整业务逻辑的实现，如CT图像重建，材料原子结构定量分析及建模，数据产品生成等
- **DataHdlerAlg**: 不同类型数据 I/O 的实现，如HDF5，Tiff，fits等
- **Workbench**: 通用用户界面
- **Webapp**: 基于 web 的应用软件
- **PyServices**: 外部公共服务，如条件数据库查询等
- **Examples**: 用户开发示例

方法学应用开发示例 —— 算法实现（Python）

```python
import numpy as np
import tomopy
from Daisy import DaisyAlg
class AlgTomopyRecon(DaisyAlg):
    def __init__(self, name):
        super().__init__(name)

    def initialize(self):
        self.data = self.get("DataStore").data()
        self.LogInfo("initialized, Tomopy Reconstruction")
        return True

    def execute(self, input_dataobj, theta, center, alg_type, output_dataobj):
        projs  = self.data[input_dataobj]
        thetas = self.data[theta]
        dataobj = tomopy.recon(projs, thetas, center=center, algorithm=alg_type)
        self.data[output_dataobj] = dataobj
        return True

    def finalize(self):
        self.LogInfo("finalized")
        return True
```

A

继承算法基类，实现 initialize、execute 和 finalize 三个方法。方法学的数据处理逻辑在 **execute** 中实现

一个算法一般包括一组**输入数据对象**，一组**输出数据对象**，一组**计算参数**。

从数据仓库中拿到输入数据

实现具体的数据处理逻辑

将处理结果写回数据仓库，写回的数据可被其它模块访问，或保存到输出文件

方法学应用开发示例 —— Workflow 实现（Python）

```
@Daisy.Singleton
class WorkflowCTReconstruct(Daisy.PyWorkflow):
 def execute(self):
  self.engine['loadhdf5'].execute(input_path='/entry/tomo', output_dataobj='tomodata')
  self.engine['loadhdf5'].execute(input_path='/entry/dark', output_dataobj='darkdata')
  self.engine['loadhdf5'].execute(input_path='/entry/flat', output_dataobj='flatdata')
  self.engine['normalize'].execute(projs_dataobj='tomodata',darks_dataobj='darkdata',\
                                   flats_dataobj='flatdata', output_dataobj='normdata')
  self.engine['angles'].execute(input_dataobj='normdata', output_dataobj='thetas')
  self.engine['minuslog'].execute(input_dataobj='normdata', output_dataobj='mlogdata')
  self.engine['reconstruct'].execute(input_dataobj='mlogdata', theta='thetas',\
                                     center=1030, alg_type='fbp',output_dataobj='recodata')
  self.engine['savehdf5'].execute(input_dataobj='recodata',output_path='/entry/reco')

wf = WorkflowCTReconstruct('WorkflowCTReconstruct')
wf.initialize(workflow_engine='PyWorkflowEngine',\
              workflow_environment = init_dict, algorithms_cfg = cfg_dict)
wf.execute()

data =wf.data_keys()
algs =wf.algorithm_keys()

wf.finalize()
```

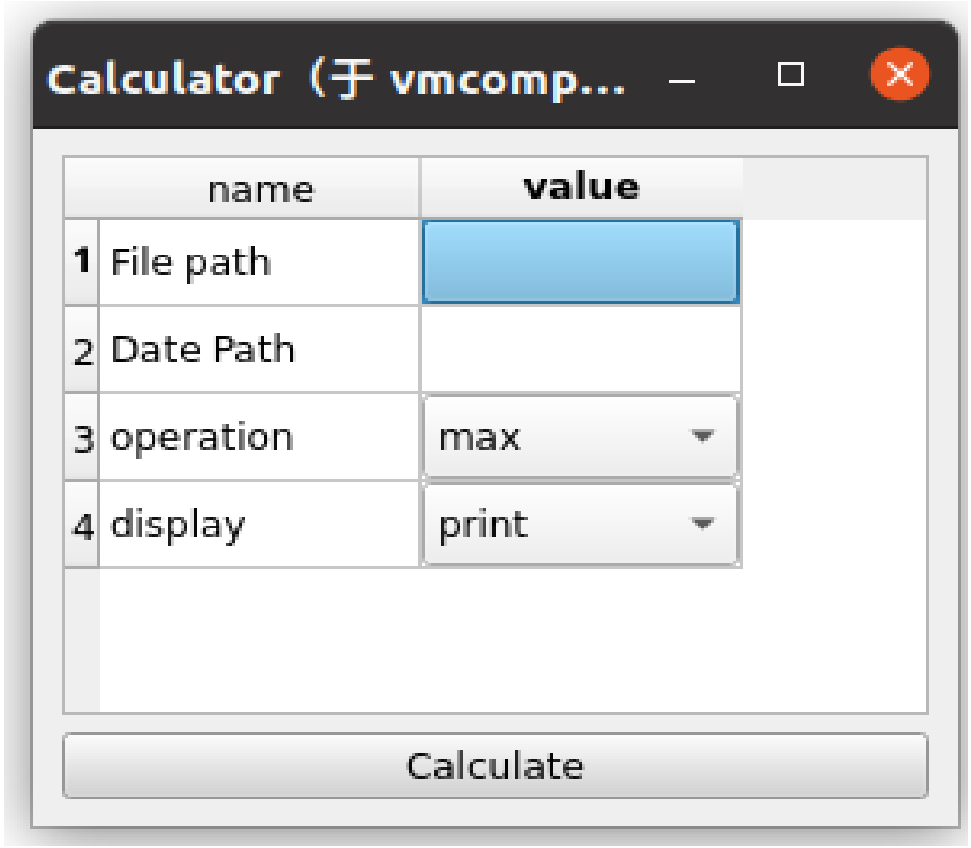
B

工作流和算法一样，实现 initialize、execute 和 finalize 三个方法。在**execute**方法中按序列调用算法或者子工作流，实现具体的业务逻辑。

工作流初始化时需要指定工作流引擎，工作流引擎通过算法名创建和调用特定算法和工作流模块，并通过数据仓库管理算法所需的输入输出数据对象。

算法对象初始化参数列表和算法运行参数可以由 JSON 对象或者 Python 字典对象表示

方法学应用开发示例 —— 方法学 interface 开发



Interface 功能：

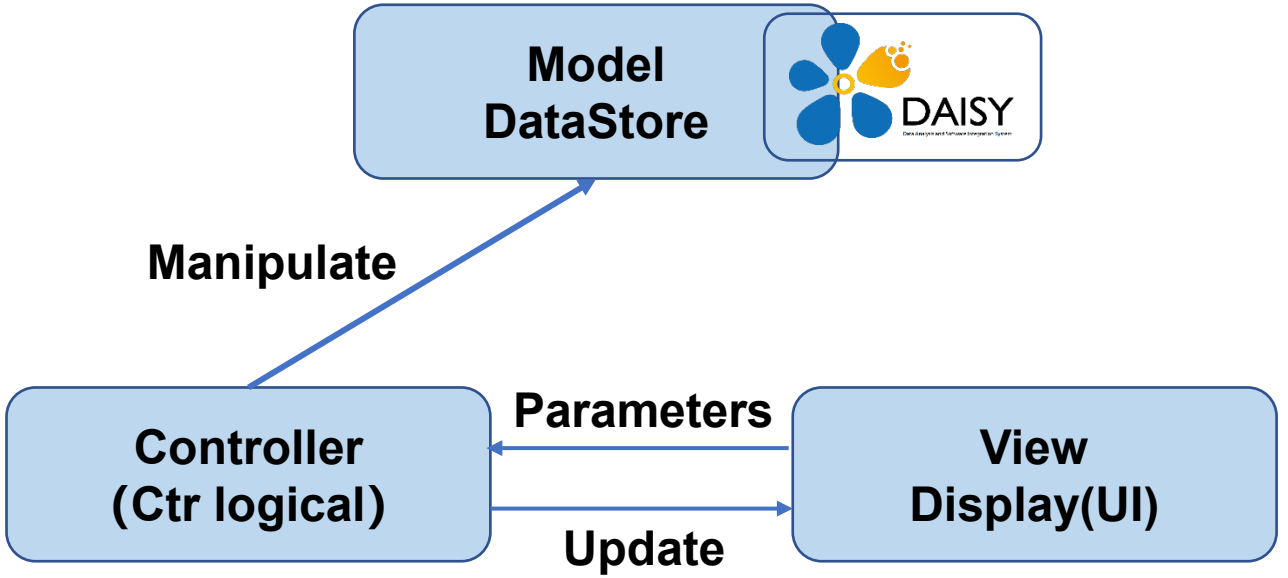
1.调用 Daisy 算法从 HDF5 文件中读取矩阵数据。

2.调用 Daisy 算法求矩阵数据的最大值或最小值。

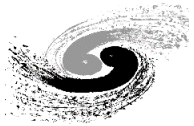
3.打印结果。

建议开发设计模式：MVC/MVP

- Model 是业务模型，View 层是界面，Controller 用来调度 View 和 Model。
- 用户界面和业务逻辑分离，使得一个程序可以使用不同的表现形式，同一个界面下面也可以有不同的业务实现，使得代码具有可扩展性、可复用性，维护方便。
- 本例中 Daisy 的数据处理逻辑即包含在 Model。



方法学应用开发示例 —— 方法学 interface 开发



开发流程：

1. 在 Daisy/Workbench/windows/interface/ 目录下创建该 interface 的子目录，本例为 Calculator/。
2. Calculator 目录下创建必要的文件并实现相应的代码，本例的代码目录如下：

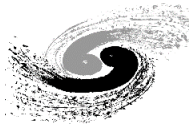
```
1 ├── CalculatorManager.py
2 ├── Controller.py
3 ├── __init__.py
4 ├── Model.py
5 └── View.py
```

CalculatorManager.py文件为该 interface 的入口程序。

CalculatorManager.py：

```python
from windows.interface.Calculator.Model import Model
from windows.interface.Calculator.View import View
from windows.interface.Calculator.Controller import Controller

"""
A wrapper class for setting the main window of the interface
"""
class CalculatorManager(QtWidgets.QMainWindow):

    def __init__(self, parent=None):
        super(CalculatorManager, self).__init__(parent)  # noqa
        self.setWindowFlag(Qt.Window)
        self.setAutoFillBackground(True)

        self.window = QtWidgets.QMainWindow()
        demo_view = View()
        demo_model = Model()
        # create controller
        self.controller = Controller(demo_view, demo_model)
        # set the view for the main window
        self.setCentralWidget(demo_view)
        self.setWindowTitle("Calculator")

    def show(self) -> None:
        super(CalculatorManager, self).show()
        self.activateWindow()
```

方法学应用开发示例 —— 方法学 interface 开发



开发流程：

1. 在 Daisy/Workbench/windows/interface/ 目录下创建该 interface 的子目录，本例为 Calculator/。
2. Calculator 目录下创建必要的文件并实现相应的代码，本例的代码目录如下：

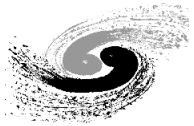
```
1 ├── CalculatorManager.py
2 ├── Controller.py
3 ├── __init__.py
4 ├── Model.py
5 └── View.py
```

Model.py 调用Daisy 算法’ LoadHDF5’ 和 ‘MaxMin’ 实现HDF5 文件的加载以及矩阵数据极值的求解

Model.py：

```python
from api import work_flow

class Model(object):
    def __init__(self):
        # Get the algorithms from Daisy workflowengine
        self.MaxMin = work_flow.engine['MaxMin']
        self.Loadh5 = work_flow.engine['LoadHDF5']

    def CalMaxmin(self, Filepath, DataPath, operation):
        # Load matrix data from the HDF5 file
        self.Loadh5.config({'inputfile_name': Filepath})
        self.Loadh5.execute(input_path = DataPath, output_dataobj='data_h5')
        # Calculate the maximum or minimum of the matrix
        if operation == "max":
            self.MaxMin.execute(input_dataobj='data_h5',  output_dataobj='maxmin', showMin=False)
        elif operation == "min":
            self.MaxMin.execute(input_dataobj='data_h5',  output_dataobj='maxmin', showMin=True)
        # Get the result from the datastore
        self.result = work_flow.engine.datastore['maxmin']
        return self.result
```

方法学应用开发示例 —— 方法学 interface 开发



开发流程：

1. 在 Daisy/Workbench/windows/interface/ 目录下创建该 interface 的子目录，本例为 Calculator/。
2. Calculator 目录下创建必要的文件并实现相应的代码，本例的代码目录如下：

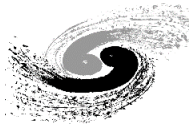
```
1 ├── CalculatorManager.py
2 ├── Controller.py
3 ├── __init__.py
4 ├── Model.py
5 └── View.py
```

View.py 添加窗口小部件实现图形化用户界面

View.py：

```
class View(QtWidgets.QDialog):
    # In PyQt signals are the first thing to be defined in a class:
    displaySignal = QtCore.pyqtSignal()
    btnSignal = QtCore.pyqtSignal()

    def __init__(self, parent=None):
        # Call QDialog's constructor
        super(View, self).__init__(parent)

        # Initialise the widgets for the view (this can also be done from Qt Creator
        self.table = QtWidgets.QTableWidget()
        self.table.setWindowTitle("MVP Demo")
        self.table.resize(600, 250)
        self.table.setRowCount(5)
        self.table.setColumnCount(2)
        self.table.setHorizontalHeaderLabels("name;value;".split(";"))

        self.FilePath = ''

        # Set display values in the widgets
        keys = ['File path', 'Date Path', 'operation',  'display', 'result']
        self.combo = {}
        self.create_File_selction(0, 1, 'File')
        self.create_combo_table(2, 1, 'operations')
        self.create_combo_table(3, 1, 'display')
        for row in range(len(keys)):
            self.set_names(keys[row], row)

        # Initialise layout of the widget and add child widgets to it
        grid = QtWidgets.QGridLayout()
        grid.addWidget(self.table)
```

方法学应用开发示例 —— 方法学 interface 开发



开发流程：

1. 在 Daisy/Workbench/windows/interface/ 目录下创建该 interface 的子目录，本例为 Calculator/。
2. Calculator 目录下创建必要的文件并实现相应的代码，本例的代码目录如下：

```
1 ├── CalculatorManager.py
2 ├── Controller.py
3 ├── __init__.py
4 ├── Model.py
5 └── View.py
```

Controller.py 连接 Model 和 View 模块，实现界面控制逻辑

Controller.py：

```python
class Controller(object):
    # Pass the view and model into the presenter
    def __init__(self, demo_view, demo_model):
        self.model = demo_model
        self.view = demo_view

        # Define the initial view
        # Note that, in the view, the drop-down could be replaced with a set of
        # tick boxes and this line would remain unchanged - an advantage of
        # decoupling the presenter and view
        self.view.set_options('operations', ['max', 'min'])
        self.view.set_options('display', ['print', 'update', 'print and update'])
        self.printToScreen = True
        self.view.hide_display()

        # Connect to the view's custom signals
        self.view.btnSignal.connect(self.handle_button)
        self.view.displaySignal.connect(self.display_update)

    # The final two methods handle the signals
    def display_update(self):
        display = self.view.get_display()
        if display == 'update':
            self.printToScreen = False
            self.view.show_display()
        elif display == 'print':
            self.printToScreen = True
            self.view.hide_display()
        else:
            self.printToScreen = True
            self.view.show_display()
```

方法学应用开发示例 —— 方法学 interface 开发

开发流程：

3. 在 Daisy/Workbench/windows/MainWindowManager.py 中主窗口的 setup 函数中进行 interface 窗口实例化。

```
from windows.interface.Integration.CalculatorManager import CalculatorManager
self.interfaces_calculator = CalculatorManager(self)
```

4. 在 create_interfaces_action 中，为该 interface 在主窗口的 interface 菜单下添加一个按钮连接到该 interface。

```
create_action(
                self,
                "calculator",
                on_triggered=self.interfaces_calculator.show,
                shortcut_context=Qt.ApplicationShortcut,
            ),

```