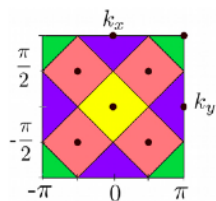
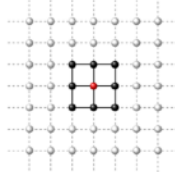




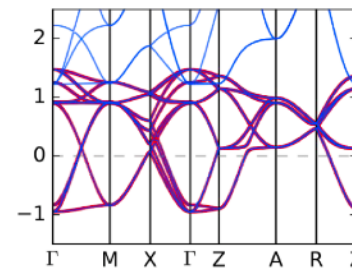
A Software Platform for Quantum Embedding & Diagrammatic Methods

Nils Wentzell

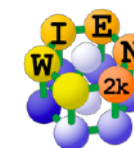




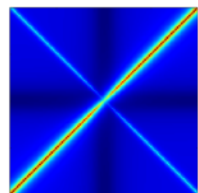
DMFT & Cluster Extensions



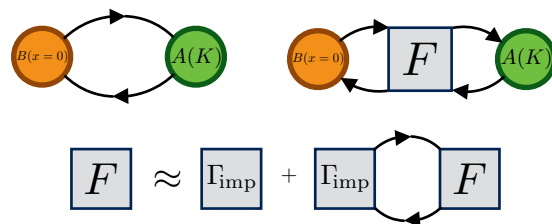
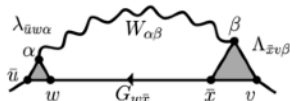
Abinitio DMFT



TRIQS



Diagrammatic & Vertex Methods



Impurity Solvers

ED

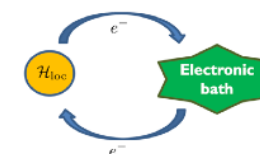
Tensor-Network

CTQMC

NRG

HubbardI

Hartree-Fock

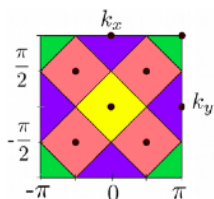




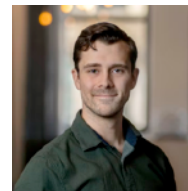
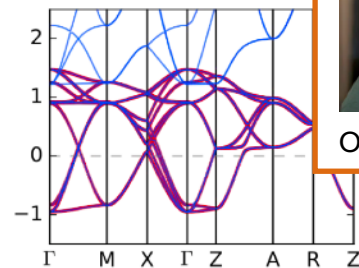
Olivier Parcollet



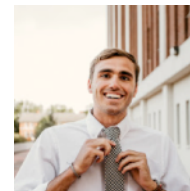
Antoine Georges



DMFT & Cluster Extensions



Olivier Gingras



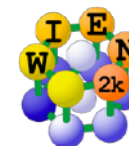
Harrison LaBollita



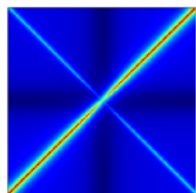
Jennifer Coulter



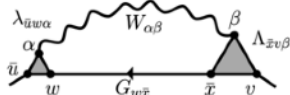
Abinitio DMFT



TRIQS



Diagrammatic & Vertex Methods



Chia-Nan Yeh



Hugo Strand

Impurity Solvers

ED

CTQMC

Tensor-Network

NRG

HubbardI

Hartree-Fock





Michel Ferrero



Toolbox for **R**esearch on **I**nteracting **Q**uantum **S**ystems

O. Parcollet et al. CPC '15 ~ 500 Citations

- Library of Fundamental Building Blocks
- Applications assemble pieces
- High-level Interface in  python™
- Low-level Backend in Modern 





Toolbox for **R**esearch on **I**nteracting **Q**uantum **S**ystems



O. Parcollet et al. CPC '15 ~ 500 Citations



Version 3.3.1



Wentzell released this Sep 3, 2024



- Library of Fundamental Building Blocks
- Applications assemble pieces
- High-level Interface in  python™
- Low-level Backend in Modern 





Toolbox for **R**esearch on **I**nteracting **Q**uantum **S**ystems

O. Parcollet et al. CPC '15 ~ 500 Citations

- Library of Fundamental Building Blocks
- Applications assemble pieces
- High-level Interface in  python™
- Low-level Backend in Modern 



Version 3.3.1



Wentzell released this Sep 3, 2024

Version 4.0



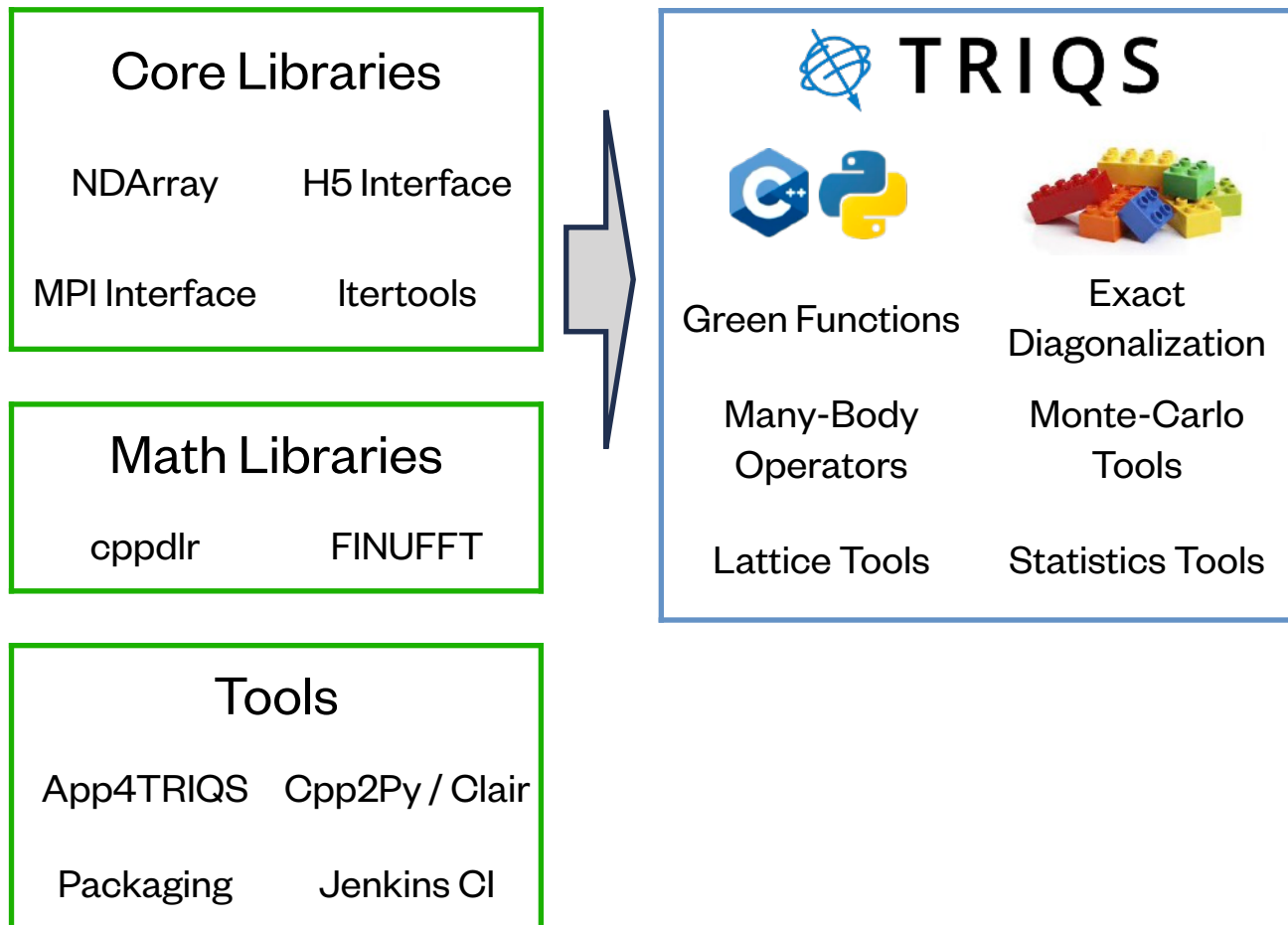
- Release: Dec 2025
- Used for this School



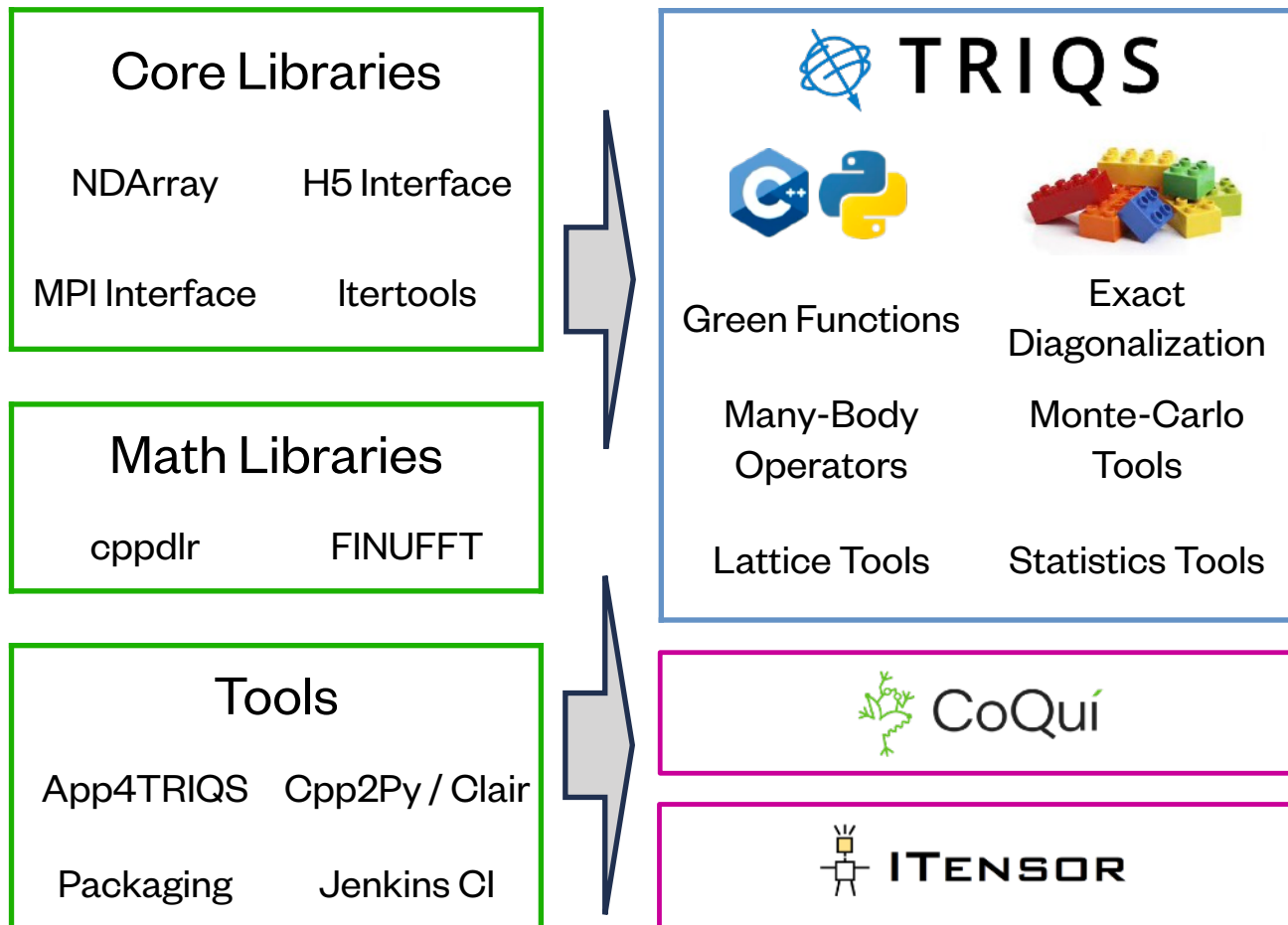
TRIQS Software Stack



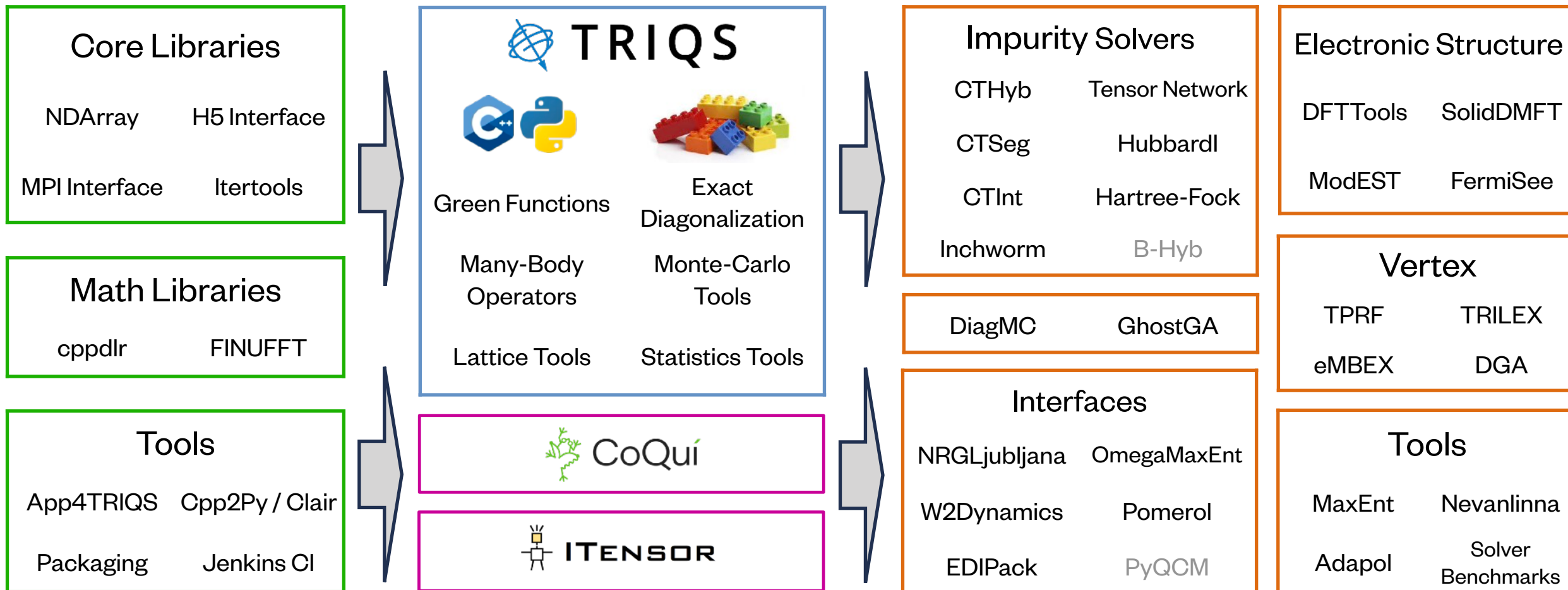
TRIQS Software Stack



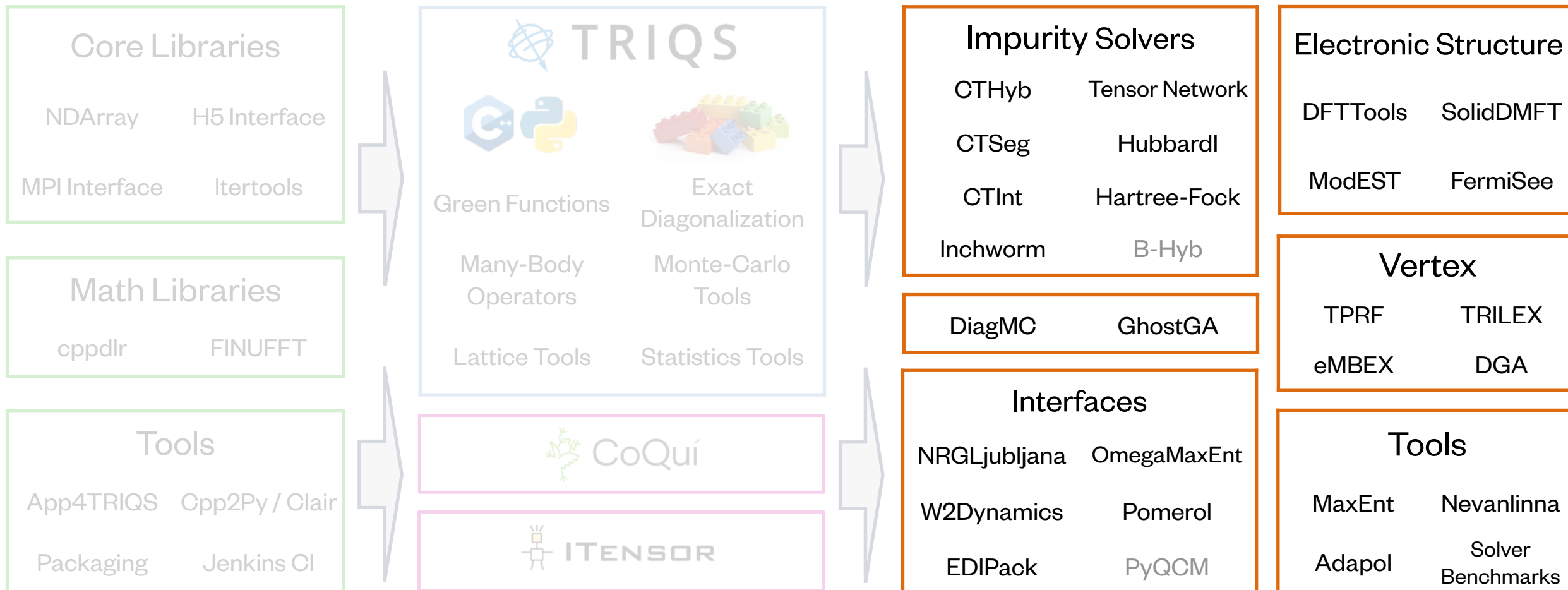
TRIQS Software Stack



TRIQS Software Stack

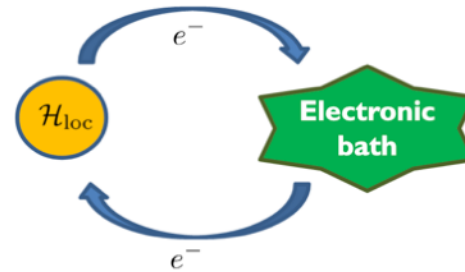


TRIQS Software Stack



Continuous-Time **Hy**bridization Expansion

Quantum Monte-Carlo *P. Seth et al. CPC '15*



- Quantum Impurity Solver
- Hybridization Expansion
- Generic Multi-band/orbital Interactions

- Complex Interactions $\sum_{ijkl} \sum_{\sigma\sigma'} U_{ijkl}^{\sigma\sigma'} c_{\sigma i}^\dagger c_{\sigma' j}^\dagger c_{\sigma' k} c_{\sigma l}$

What can we measure?

$$\langle \mathcal{T} c_{\sigma i}(\tau) c_{\sigma j}^\dagger \rangle$$

$$\langle \mathcal{T} c_{\sigma i}^\dagger(i\omega) c_{\sigma j}(i\omega') c_{\sigma' k}^\dagger(i\omega'') c_{\sigma' l}(0) \rangle$$

$$\langle \mathcal{T} A(\tau) B(0) \rangle$$

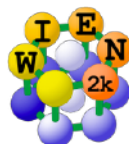
ModEST

triqs.github.io/modest

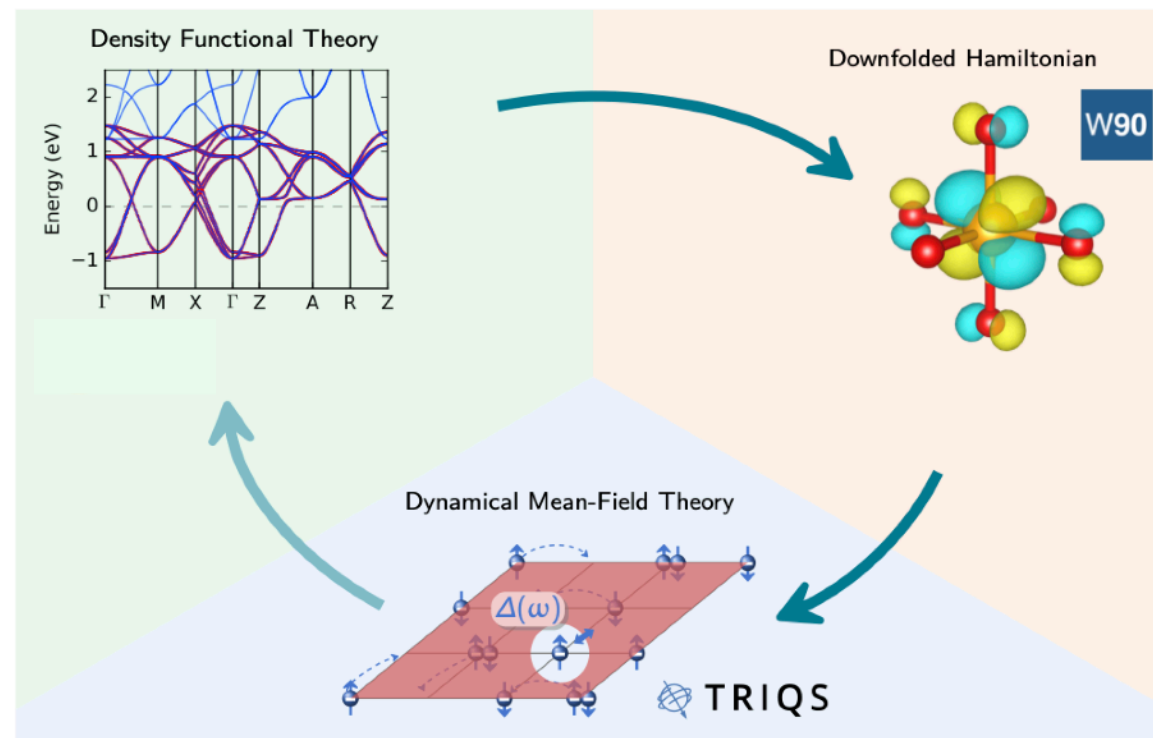


Modular Electronic Structure Toolkit

- Modular, generic DMFT framework
- Unified interface to electronic structure codes (DFT, Wannier functions, MBPT)
- Efficient k-summations
- Comprehensive tooling

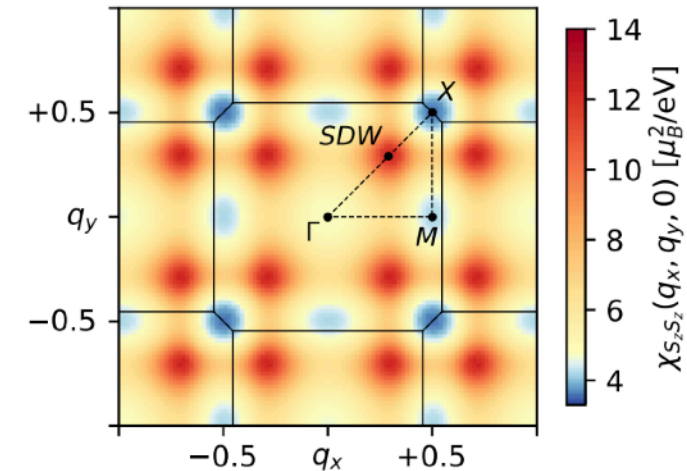


WANNIER90

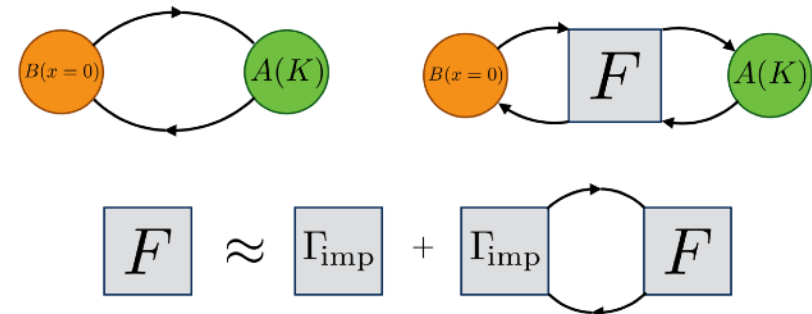


Two-Particle Response Function Tool Box

- Lindhard Susceptibilities
- Random-phase Approximation
- GW Approximation
- Generalized Susceptibilities
- Bethe-Salpeter Equation Solver
- Vertex-Corrected Lattice Susceptibilities



H. Strand et al. PRB '19



Documentation

triqs.github.io/triqs/unstable



TRIQS
3.2.0

Search docs

- Welcome
- Installation
- Documentation
 - Manual
 - C++ API
 - Python API
 - triqs.atom_diag
 - triqs.dos
 - triqs.fit
 - triqs.gf
 - triqs.lattice
 - triqs.operators
 - triqs.plot
 - triqs.random_generator
 - triqs.stat
 - triqs.sumk
 - triqs.utility
- Applications based on TRIQS
- User guide
- Contributing

» Documentation » triqs.gf » triqs.gf.meshes » triqs.gf.meshes.MeshImFreq

triqs.gf.meshes.MeshImFreq

class triqs.gf.meshes.MeshImFreq

Mesh of Matsubara frequencies

Parameters:

- *beta* (*float*) – Inverse temperature
- *S* (*str*) – Statistic, 'Fermion' or 'Boson'
- *n_iw* (*int* [default=1025]) – Number of positive Matsubara frequencies

Methods

<code>__init__</code> (*args, **kwargs)	Initialize self.
<code>copy</code>	Signature : () -> MeshImFreq Make a copy (clone) of self
<code>copy_from</code>	Signature : (MeshImFreq other) -> None Assignment
<code>first_index</code>	Signature : () -> int
<code>index_to_linear</code>	Signature : (int i) -> int index -> linear index
<code>last_index</code>	Signature : () -> int
<code>positive_only</code>	Signature : () -> bool
<code>set_tail_fit_parameters</code>	Signature : (float tail_fraction, int n_tail_max = 30, std::optional<int> expansion_order = {}) -
<code>values</code>	Signature : () -> PyObject * A numpy array of all the values of the mesh points

Installation

triqs.github.io/triqs/latest/install



- **Anaconda** `conda install -c conda-forge triqs`



- **Ubuntu 24.04** `apt-get install triqs`



- **Docker** `docker pull flatironinstitute/triqs:school2025`



- **Apptainer** `apptainer pull docker://flatironinstitute/triqs:school2025`



TRIQS Tutorials

github.com/triqs/tutorials



The screenshot shows the TRIQS 3.3 share JupyterLab interface. On the left, a file browser displays the directory structure under '/ tutorials / Basics /'. The files listed are: solutions (23 minutes ago), 00a-Introducing_th... (23 minutes ago), 00b-Matplotlib_Exa... (23 minutes ago), 01-Greens_function... (23 minutes ago, selected), 02-Archiving_your_... (23 minutes ago), 03-Operators.ipynb (23 minutes ago), 04-Multivariable_Gr... (23 minutes ago), my_archive.h5 (40 minutes ago), and sample.dat (2 hours ago). A green box highlights the 'Basics' folder and its subfolders: ModelDMFT, AbinitioDMFT, TwoParticleResponse, and CoQui.

The main area displays the '01-Greens_functions.ipynb' notebook. The title is 'TRIQS Green's functions'. The text explains that it is now time to start using some of the tools provided by TRIQS. It states that much of the functionality in TRIQS, while implemented in C++ for optimal performance, is exposed through a Python interface to make it easier to use. From a practical point of view, this means that you can think of TRIQS as a python library, just like numpy or matplotlib.

One of the central objects of a many-body calculation is a Green's function. Green's functions in TRIQS are functions defined on a mesh \mathcal{M} of points that hold values in some domain \mathcal{D} , for example $\mathbb{C}^{2 \times 2}$.

$$G : \mathcal{M} \rightarrow \mathcal{D}$$

A few common Green's function meshes in TRIQS include:

- MeshReFreq - Real-frequencies equally spaced in $[\omega_{min}, \omega_{max}]$
- MeshImFreq - Matsubara Frequencies
- MeshImTime - Imaginary time points equally spaced in $[0, \beta]$
- MeshReTime - Real-time points (not covered in this tutorial)

Let's see how we can **construct a Mesh and print its values**.

```
[ ]: # Import the Mesh type we want to use
from triqs.gf import MeshImTime

# The documentation tells us which parameters we need to pass for the mesh construction
?MeshImTime

[ ]: # Provide the inverse temperature, Statistic, and number of points
tau_mesh = MeshImTime(beta=5, statistic='Fermion', n_tau=11)
```

The status bar at the bottom indicates 'Simple' mode, '0' lines, '1' column, 'TRIQS 3.3 share | Idle', 'Mode: Command', 'Ln 1, Col 1', and '01-Greens_functions.ipynb'.

Getting Started

sdsc-binder.flatironinstitute.org



Sign in with Google

Getting Started

sdsc-binder.flatironinstitute.org



Choose project

Owner

Project

URL to open (optional)

File ▾

launch

Check your currently
running server.

Documentation for users
and Flatiron researchers.

Binder is provided as
service to the community.
All storage is temporary
and regularly purged. Do
not store any sensitive or
critical data.

Getting Started

sdsc-binder.flatironinstitute.org



Choose project

Owner

ccq

Project

triqs

URL to open (optional)

URL to open (optional)

File ▾

launch

Check your currently
running server.

Documentation for users
and Flatiron researchers.

Binder is provided as
service to the community.
All storage is temporary
and regularly purged. Do
not store any sensitive or
critical data.

Getting Started

sdsc-binder.flatironinstitute.org



Choose project

Owner

ccq

Project

triqs

URL to open (optional)

URL to open (optional)

File ▾

launch

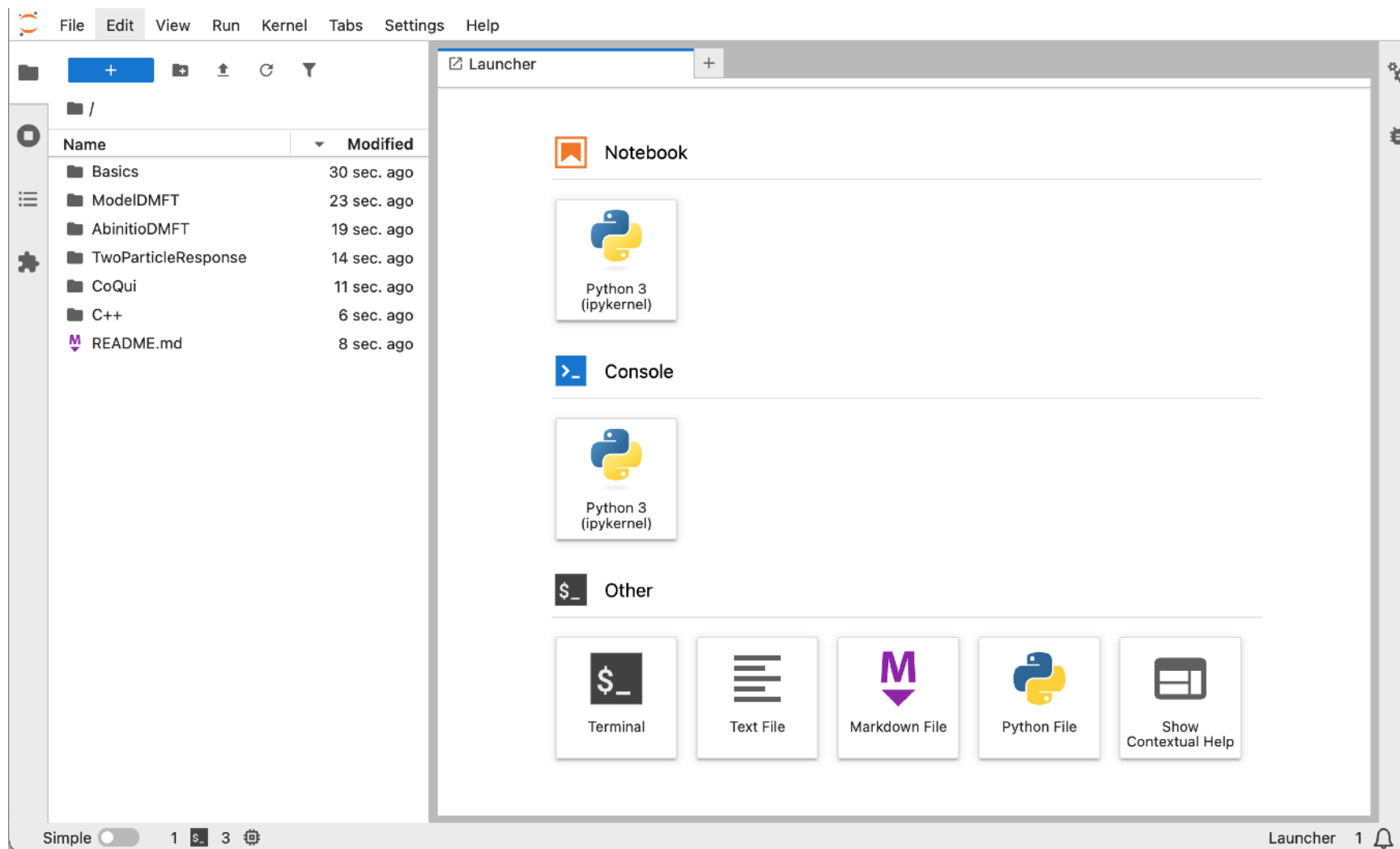
Check your currently
running server.

Documentation for users
and Flatiron researchers.

Binder is provided as
service to the community.
All storage is temporary
and regularly purged. Do
not store any sensitive or
critical data.

Getting Started

sdsc-binder.flatironinstitute.org



Getting Started

sdsc-binder.flatironinstitute.org



File Edit View Run Kernel Tabs Settings Help

/ Basics /

Name	Modified
solutions	1 hr. ago
00a-Introducing_the_ipyt...	1 hr. ago
00b-Matplotlib_Example...	1 hr. ago
01-Greens_functions.ipynb	1 min. ago
02-Archiving_your_data.i...	1 hr. ago
03-Operators.ipynb	1 hr. ago
04-Multivariable_Green_f...	1 hr. ago
sample.dat	1 hr. ago

Launcher

01-Greens_functions.ipynb

Python 3 (ipykernel)

TRIQS Green's functions

It is now time to start using some of the tools provided by TRIQS.

Much of the functionality in TRIQS, while implemented in C++ for optimal performance, is exposed through a Python interface to make it easier to use. In practice, this means you can treat TRIQS as a Python library, just like NumPy or matplotlib.

One of the central objects of a many-body calculation is a Green's function. Green's functions in TRIQS are functions defined on a mesh \mathcal{M} of points that hold values in some domain \mathcal{D} , for example $\mathbb{C}^{2 \times 2}$

$$G : \mathcal{M} \rightarrow \mathcal{D}$$

A few common Green's function meshes in TRIQS include:

- `MeshReFreq` - Real-frequencies equally spaced in $[\omega_{min}, \omega_{max}]$
- `MeshImFreq` - Matsubara Frequencies
- `MeshImTime` - Imaginary time points equally spaced in $[0, \beta]$
- `MeshReTime` - Real-time points (not covered in this tutorial)

Let's see how we can **construct a Mesh and print its values**.

```
[ ]: # Import the Mesh type we want to use
from triqs.gf import MeshImTime

# The documentation tells us which parameters we need to pass for the mesh construction
?MeshImTime
```

Simple 1 4 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 01-Greens_functions.ipynb 1

Getting Started

sdsc-binder.flatironinstitute.org



File Edit View Run Kernel Tabs Settings Help

/ Basics /

Name	Modified
solutions	1 hr. ago
00a-Introducing_the_ipyt...	1 hr. ago
00b-Matplotlib_Example...	1 hr. ago
01-Greens_functions.ipynb	1 min. ago
02-Archiving_your_data.i...	1 hr. ago
03-Operators.ipynb	1 hr. ago
04-Multivariable_Green_f...	1 hr. ago
sample.dat	1 hr. ago

Download Current Folder as an Archive
Paste
New File
New Notebook
New Folder
Shift+Right Click for Browser Menu

TRIQS Green's functions

It is now time to start using some of the tools provided by TRIQS.

Much of the functionality in TRIQS, while implemented in C++ for optimal performance, is exposed through a Python interface to make it easier to use. In practice, this means you can treat TRIQS as a Python library, just like NumPy or matplotlib.

One of the central objects of a many-body calculation is a Green's function. Green's functions in TRIQS are functions defined on a mesh \mathcal{M} of points that hold values in some domain \mathcal{D} , for example $\mathbb{C}^{2 \times 2}$

$$G : \mathcal{M} \rightarrow \mathcal{D}$$

A few common Green's function meshes in TRIQS include:

- MeshReFreq - Real-frequencies equally spaced in $[\omega_{min}, \omega_{max}]$
- MeshImFreq - Matsubara Frequencies
- MeshImTime - Imaginary time points equally spaced in $[0, \beta]$
- MeshReTime - Real-time points (not covered in this tutorial)

Let's see how we can **construct a Mesh and print its values**.

```
[ ]: # Import the Mesh type we want to use
from triqs.gf import MeshImTime

# The documentation tells us which parameters we need to pass for the mesh constructio
?MeshImTime
```

Simple 1 4 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 01-Greens_functions.ipynb 1

Getting Started

sdsc-binder.flatironinstitute.org



The screenshot displays the JupyterLab interface. On the left, a file browser shows a directory structure with files like '01-Greens_functions.ipynb'. A context menu is open over the file browser, showing options like 'Download Current Folder as an Archive', 'Paste', 'New File', 'New Notebook', and 'New Folder'. The main area shows a code editor with the title '01-Greens_functions.ipynb' and the content 'TRIQS Green's functions'. The text describes the TRIQS library and its Python interface. A mathematical expression $G : \mathcal{M} \rightarrow \mathcal{D}$ is shown. Below the text, a list of common Green's function meshes is provided: MeshReFreq, MeshImFreq, MeshImTime, and MeshReTime. At the bottom, a code cell is partially visible, showing imports for MeshImTime.

Owner: ccq

Project: triqs

TRIQS Green's functions

It is now time to start using some of the tools provided by TRIQS.

Much of the functionality in TRIQS, while implemented in C++ for optimal performance, is exposed through a Python interface to make it easier to use. In practice, this means you can treat TRIQS as a Python library, just like NumPy or matplotlib.

One of the central objects of a many-body calculation is a Green's function. Green's functions in TRIQS are functions defined on a mesh \mathcal{M} of points that hold values in some domain \mathcal{D} , for example $\mathbb{C}^{2 \times 2}$

$$G : \mathcal{M} \rightarrow \mathcal{D}$$

A few common Green's function meshes in TRIQS include:

- MeshReFreq - Real-frequencies equally spaced in $[\omega_{min}, \omega_{max}]$
- MeshImFreq - Matsubara Frequencies
- MeshImTime - Imaginary time points equally spaced in $[0, \beta]$
- MeshReTime - Real-time points (not covered in this tutorial)

Let's see how we can **construct a Mesh and print its values**.

```
[ ]: # Import the Mesh type we want to use
from triqs.gf import MeshImTime

# The documentation tells us which parameters we need to pass for the mesh construction
?MeshImTime
```