TRIQS : A Toolbox for Research in Interacting Quantum Systems

Introduction to Hands-on

Olivier Parcollet

Center for Computational Quantum Physics (CCQ)

Flatiron Institute, Simons Foundation

New York

European Research Council

Established by the European Commission

TRIQS team today

O. Parcollet

Nils Wentzel

Michel Ferrero

Hugo Strand

Alice Moutenet

Flatiron Institute

- Center for Computational Quantum Physics (CCQ), New York
  "Develop the concepts, theories, algorithms and codes needed to solve the quantum many-body problem…"

  https://www.simonsfoundation.org/flatiron/center-for-computational-quantum-physics/

- Supported projects :
  - TRIQS : Quantum Embedded methods (DMFT), diagrammatic
  - iTensor : DMRG, MPS …
  - NetKet : Machine learning & Quantum Many body
  - AFQMC : Auxiliary Field Monte Carlo and applications.

SIMONS FOUNDATION
Advancing Research in Basic Science and Mathematics

What is TRIQS ?

- A Toolkit (Python/C++) to build modern many body computations:
  - Quantum Embedded methods:
    - DMFT. Cluster DMFT.
    - Next generation methods (Trilex, dual fermions/bosons, DΓA,...)
    - State of the art "impurity solvers" for DMFT.
    - Ab-initio strongly correlated materials (DFT+ DMFT). Interface with electronic structure codes.
  - Diagrammatic methods, Monte Carlo, e.g.
    - Eliashberg / GW type equations (superconductivity, spin-fluctuations…)
    - "Diagrammatic" Monte Carlo

DMFT : reminder

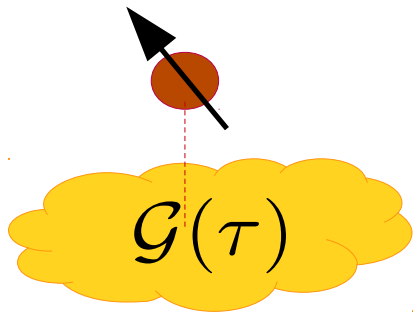
Dynamical Mean Field Theory (DMFT)

Cf D. Sénéchal's lecture

W. Metzner, D. Vollhardt, 1989
A. Georges, G. Kotliar, 1992

- Density Functional Theory
  Independent electrons in an effective periodic potential.
  Interaction taken into account "in average" (Kohn-Sham potential).

- Dynamical Mean Field Theory
  An atom coupled to a bath of non-interacting electrons,
  determined self-consistently.
  The bath represents the other atoms in the crystal.
  Well suited when atomic physics is important (multiplets)



Quantum impurity model

Reminder : Weiss Mean Field Theory

- *Ising model* (Weiss) : A single spin in an effective field.

$H = -J \sum_{ij} \sigma_i \sigma_j$	Ising model.
$m = \langle \sigma \rangle$	Order parameter.
$H_{\text{eff}} = -J h_{\text{eff}} \sigma$	Effective Hamiltonian
$h_{\text{eff}} = zJm$	Weiss Field
$m = \tanh(\beta h_{\text{eff}})$	Solution of the effective Hamiltonian

- Qualitatively correct (phase diagram, second order transition) even if critical exponents are wrong (R.G., Field theory....,)
- Derivation : e.g. large dimension limit on hypercubic lattice

Generalisation for quantum models ?

Dynamical Mean Field Theory

Ising model

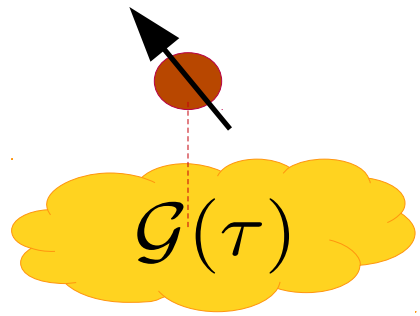
$$H = -J \sum_{ij} \sigma_i \sigma_j$$

$$m = \langle \sigma \rangle$$

$$H_{\text{eff}} = -J h_{\text{eff}} \sigma$$

$$h_{\text{eff}} = zJm$$

$$m = \tanh(\beta h_{\text{eff}})$$



?

Dynamical Mean Field Theory

- Anderson impurity with an effective band determined self-consistently

$$H = \underbrace{\sum_{\sigma=\uparrow,\downarrow} \varepsilon_d c^\dagger_\sigma c_\sigma + U n_\uparrow n_\downarrow}_{\text{Local site}} + \underbrace{\sum_{k,\sigma=\uparrow,\downarrow} V_{k\sigma}(\xi^\dagger_{k\sigma} c_\sigma + h.c.) + \sum_{k,\sigma=\uparrow,\downarrow} \varepsilon_{k\sigma}\xi^\dagger_{k\sigma}\xi_{k\sigma}}_{\text{Coupled to an effective electronic bath}}$$

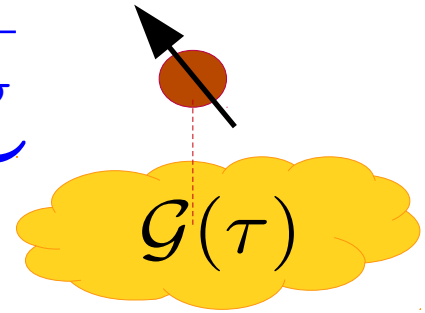
- Action form (Path integral)

$$S = -\int_0^\beta\!\!\int d\tau d\tau' c^\dagger_\sigma(\tau)\mathcal{G}^{-1}_\sigma(\tau-\tau')c_\sigma(\tau') + \int_0^\beta d\tau\, U n_\uparrow(\tau) n_\downarrow(\tau)$$

$$\mathcal{G}^{-1}_\sigma(i\omega_n) \equiv i\omega_n + \epsilon_0 - \underbrace{\sum_k \frac{|V_{k\sigma}|^2}{i\omega_n - \epsilon_{k\sigma}}}_{\Delta_\sigma(i\omega_n)}$$

Bath
"Weiss field"

Hybridization function



Lattice quantities vs impurity quantities

- Dyson equation on the lattice

$$G_{\sigma \mathrm{latt}}(k, i\omega_n) \equiv \frac{1}{i\omega_n + \mu - \epsilon_k - \Sigma_{\sigma \mathrm{latt}}(k, i\omega_n)}$$

- DMFT : the self-energy on the lattice is local :

$$\Sigma_{\sigma \mathrm{latt}}(k, i\omega_n) = \Sigma_{\sigma \mathrm{imp}}(i\omega_n)$$

- DMFT : self-consistency condition

$$G_{\sigma \mathrm{loc}}(i\omega_n) \equiv \sum_k G_{\sigma \mathrm{latt}}(k, i\omega_n) = G_{\sigma \mathrm{imp}}(i\omega_n)$$

- G_{latt} depends on k. There is a Fermi surface in metallic regimes.
- Within DMFT, Z, m*, coherence temperature, finite temperature lifetime of metals are constant along the Fermi surface.

DMFT equations (1 band paramagnetic)

Lattice model

Ising

$$H = -J \sum_{\langle ij \rangle} \sigma_i \sigma_j$$

Hubbard

$$H = -\sum_{\langle ij \rangle} t_{ij} c^\dagger_{i\sigma} c_{i\sigma} + \sum_i U n_{i\uparrow} n_{i\downarrow}$$

Effective model

$$H_{\rm eff} = -J h_{\rm eff} \sigma$$

$$m = \langle \sigma \rangle$$

$$S_{\rm eff} = -\int\int_0^\beta d\tau d\tau' c^\dagger_\sigma(\tau) \mathcal{G}^{-1}_\sigma(\tau - \tau') c_\sigma(\tau') + \int_0^\beta d\tau \; U n_\uparrow(\tau) n_\downarrow(\tau)$$

$$G_{\sigma \rm imp}(\tau) \equiv -\left\langle T c_\sigma(\tau) c^\dagger_\sigma(0) \right\rangle_{S_{\rm eff}}$$

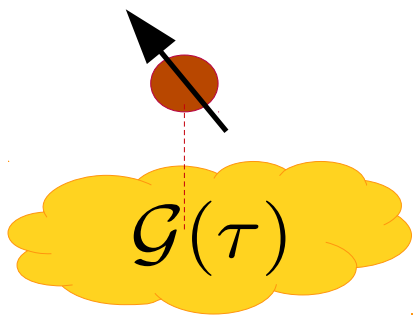
Self consistency condition

$$h_{\rm eff} = zJm$$

$$\Sigma_{\sigma \rm imp}[\mathcal{G}](i\omega_n) \equiv \mathcal{G}^{-1}_\sigma(i\omega_n) - G^{-1}_{\sigma \rm imp}[\mathcal{G}](i\omega_n)$$

$$G_{\sigma \rm imp}[\mathcal{G}](i\omega_n) = \sum_k \frac{1}{i\omega_n + \mu - \epsilon_k - \Sigma_{\sigma \rm imp}[\mathcal{G}](i\omega_n)}$$

Implicit equation for the bath



Solving DMFT : iterative method

Impurity solver *Cf M. Ferrero's lecture tomorrow*

$$S_{\rm eff} = -\int\int_0^\beta d\tau d\tau' c_\sigma^\dagger(\tau)\mathcal{G}_\sigma^{-1}(\tau-\tau')c_\sigma(\tau') + \int_0^\beta d\tau\ U n_\uparrow(\tau) n_\downarrow(\tau)$$

$$G_{\sigma\rm imp}(\tau) \equiv -\left\langle T c_\sigma(\tau) c_\sigma^\dagger(0)\right\rangle_{S_{\rm eff}}$$

$$\Sigma_{\sigma\rm imp}(i\omega_n) \equiv \mathcal{G}_\sigma^{-1}(i\omega_n) - G_{\sigma\rm imp}^{-1}(i\omega_n)$$

$\mathcal{G}$

$G_{\rm imp}$, $\Sigma_{\rm imp}$

Self consistency condition

$$G_{\sigma\rm imp}[\mathcal{G}](i\omega_n) = \sum_k \frac{1}{i\omega_n + \mu - \epsilon_k - \Sigma_{\sigma\rm imp}[\mathcal{G}](i\omega_n)}$$

- In practice, the iterative loop is (almost) always convergent.

Back to TRIQS and hands on

TRIQS: different levels of usage.

- Simplest usage : run a DMFT computation
  - e.g. Vary U, study the Mott transition.
- Write your DMFT self-consistency code [Python]
  - Use building blocks in Python, including "impurity solvers"
- Write high-performance code, e.g. a new impurity solver [C++]
  - Use building blocks in C++ (from TRIQS library) and TRIQS/cpp2py to glue the two languages.
  - Not covered today.
    A little taste of it on Monday. Tutorial CT-INT, Cf later.

Hands-on menu

1. Get familiar with Python, simple Green function, operators, matplotlib

2. Your first DMFT code : built yourself a IPT solution for DMFT.

3. Solve DMFT, 1 band Hubbard model, with CT-HYB QMC solver

4. Hund's metal: a two band computation with CT-HYB

5. Cluster DMFT : a minimal two patches DCA cluster.
   Mott transition, Fermi Arcs.

6. Lattice models.
   Lindhard function, Two Particle self-consistent approximation (TPSC)

Hands-on menu

1. Get familiar with Python, simple Green function, operators, matplotlib

2. Your first DMFT code : built yourself a IPT solution for DMFT.

3. Solve DMFT, 1 band Hubbard model, with CT-HYB QMC solver

4. Hund's metal: a two band computation with CT-HYB

5. Cluster DMFT : a minimal two patches DCA cluster.
   Mott transition, Fermi Arcs.

6. Lattice models.
   Lindhard function, Two Particle self-consistent approximation (TPSC)

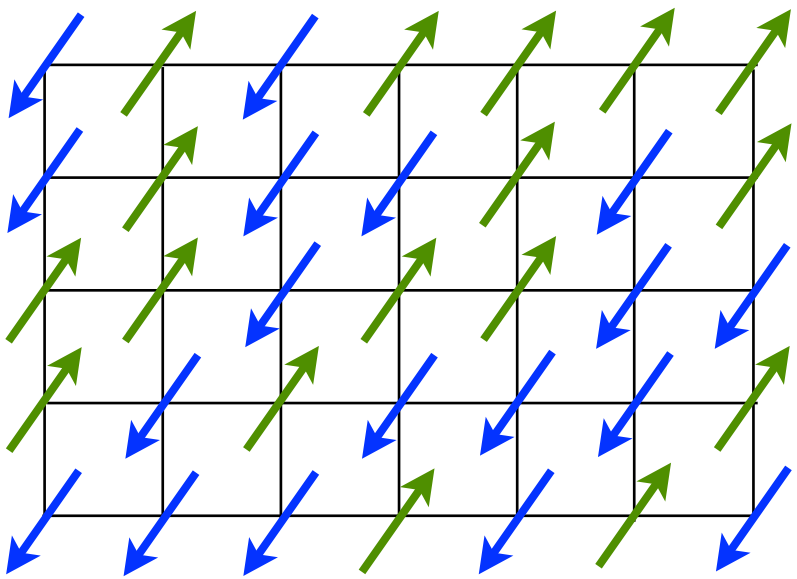
Mott insulator

N. Mott, 50's

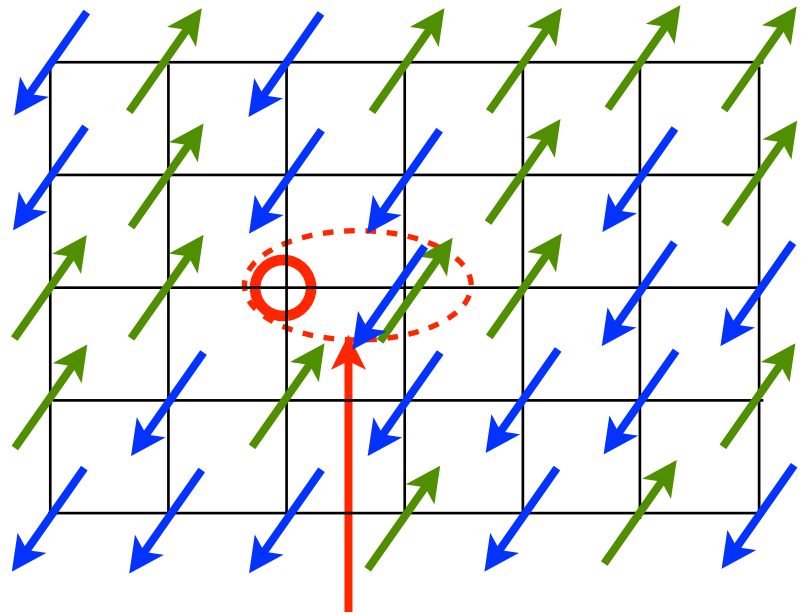
- One electron per site on average (half-filled band).
- Should be a textbook metal.
- If U is large enough, it is an insulator : charge motion frozen.

$$H = -\sum_{\langle ij\rangle,\sigma=\uparrow,\downarrow} t_{ij} c^{\dagger}_{i\sigma} c_{j\sigma} + U n_{i\uparrow} n_{i\downarrow}, \qquad n_{i\sigma} \equiv c^{\dagger}_{i\sigma} c_{i\sigma}$$

$$\delta = 1 - \langle n_{\uparrow} + n_{\downarrow} \rangle$$

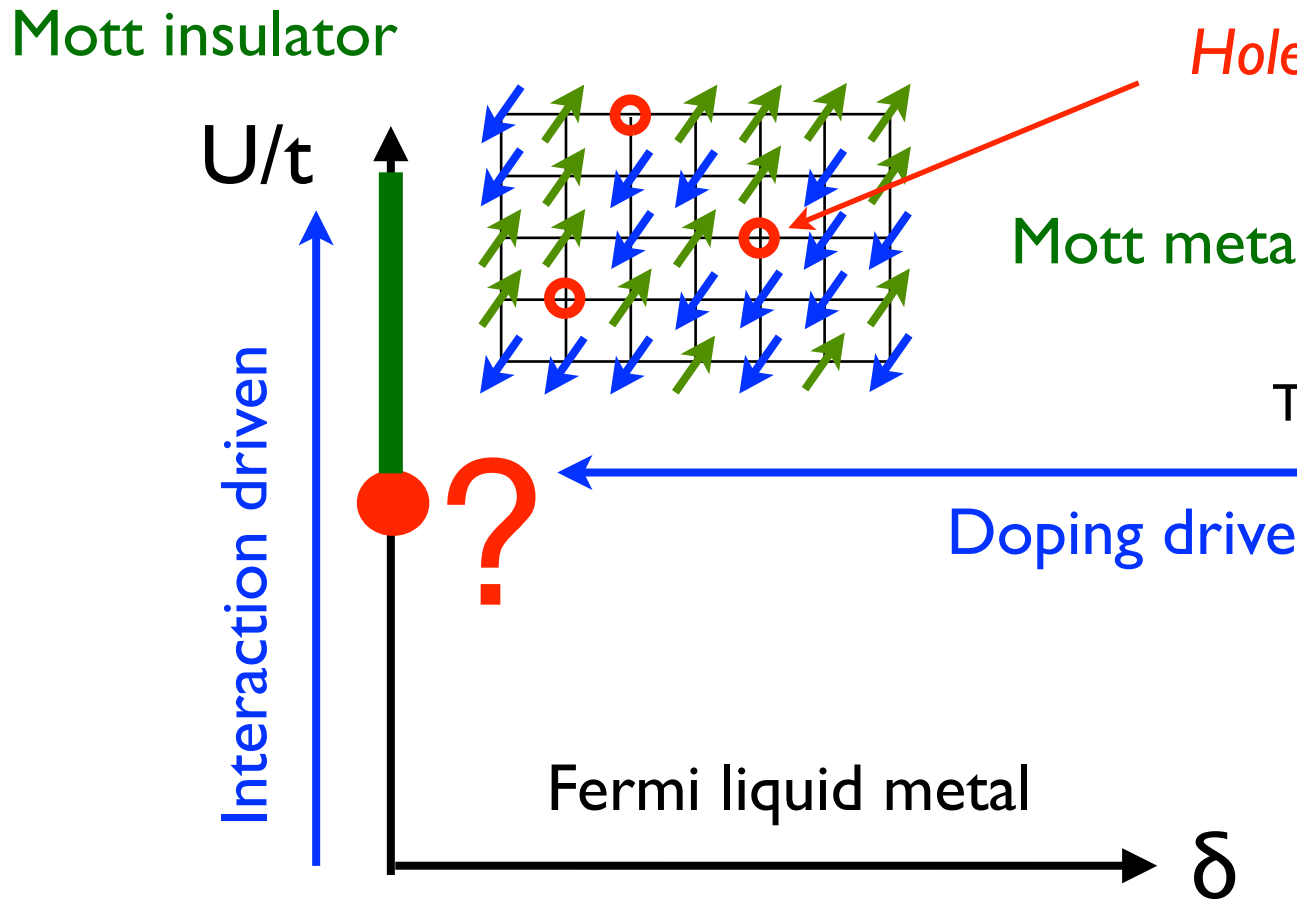


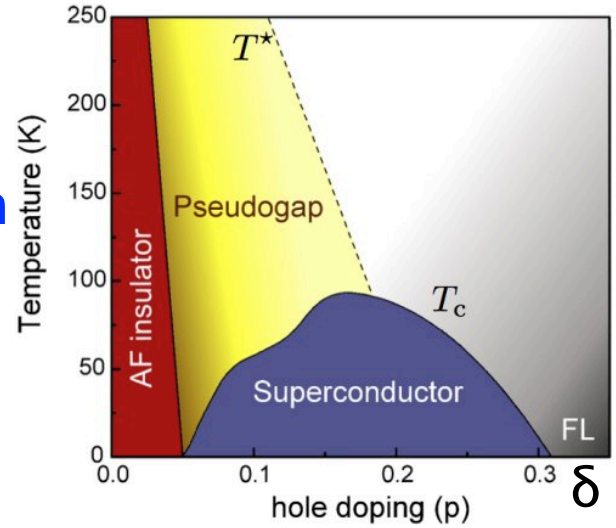
Mott insulator



Large Coulomb repulsion U ~ eV ~ 10^4 K

Doped Mott insulators





High temperature superconductors

- How is a metal destroyed close to a Mott transition ?

Iterated Perturbation Theory (IPT)
A cheap quantum impurity solver

- Anderson model : perturbation in U is regular *(Yosida, Yamada, 70's.)*.
- Bare perturbation theory at second order *(Kotliar-Georges, 1992)*.

$$\Sigma(i\omega_n) \simeq \frac{U}{2} + U^2 \int_0^\beta d\tau \; e^{i\omega_n \tau} \hat{\mathcal{G}}_0(\tau)^3$$

Advantages

- Quick and relatively simple.
- U=0 and U=∞ limit correct !
- Reproduce the main feature of the solution of the Mott transition (see lecture 1).

Drawbacks

- Largely uncontrolled
- Extension beyond 1/2 filling or for cluster do not interpolate well between U=0 and U=∞ *(see however Kajueter-Kotliar, condmat/9509152).*

Hands-on menu

1. Get familiar with Python, simple Green function, operators, matplotlib

2. Your first DMFT code : built yourself a IPT solution for DMFT.

3. Solve DMFT, 1 band Hubbard model, with CT-HYB QMC solver

4. Hund's metal: a two band computation with CT-HYB

5. Cluster DMFT : a minimal two patches DCA cluster.
   Mott transition, Fermi Arcs.

6. Lattice models.
   Lindhard function, Two Particle self-consistent approximation (TPSC)

Hands-on menu

1. Get familiar with Python, simple Green function, operators, matplotlib

2. Your first DMFT code : built yourself a IPT solution for DMFT.

3. Solve DMFT, 1 band Hubbard model, with CT-HYB QMC solver

4. Hund's metal: a two band computation with CT-HYB

5. Cluster DMFT : a minimal two patches DCA cluster.
   Mott transition, Fermi Arcs.

6. Lattice models.
   Lindhard function, Two Particle self-consistent approximation (TPSC)

Hund's metal

Cf review A. Georges. L. De Medici, J. Mravlje, arXiv:1207.3033

- Kanamori Hamiltonian.

$$H_{\rm K} = U\sum_m \hat{n}_{m\uparrow}\hat{n}_{m\downarrow} + U'\sum_{m\neq m'} \hat{n}_{m\uparrow}\hat{n}_{m'\downarrow} + (U'-J)\sum_{m<m',\sigma} \hat{n}_{m\sigma}\hat{n}_{m'\sigma} +$$
$$-J\sum_{m\neq m'} d^+_{m\uparrow}d_{m\downarrow}\,d^+_{m'\downarrow}d_{m'\uparrow} + J\sum_{m\neq m'} d^+_{m\uparrow}d^+_{m\downarrow}\,d_{m'\downarrow}d_{m'\uparrow}$$

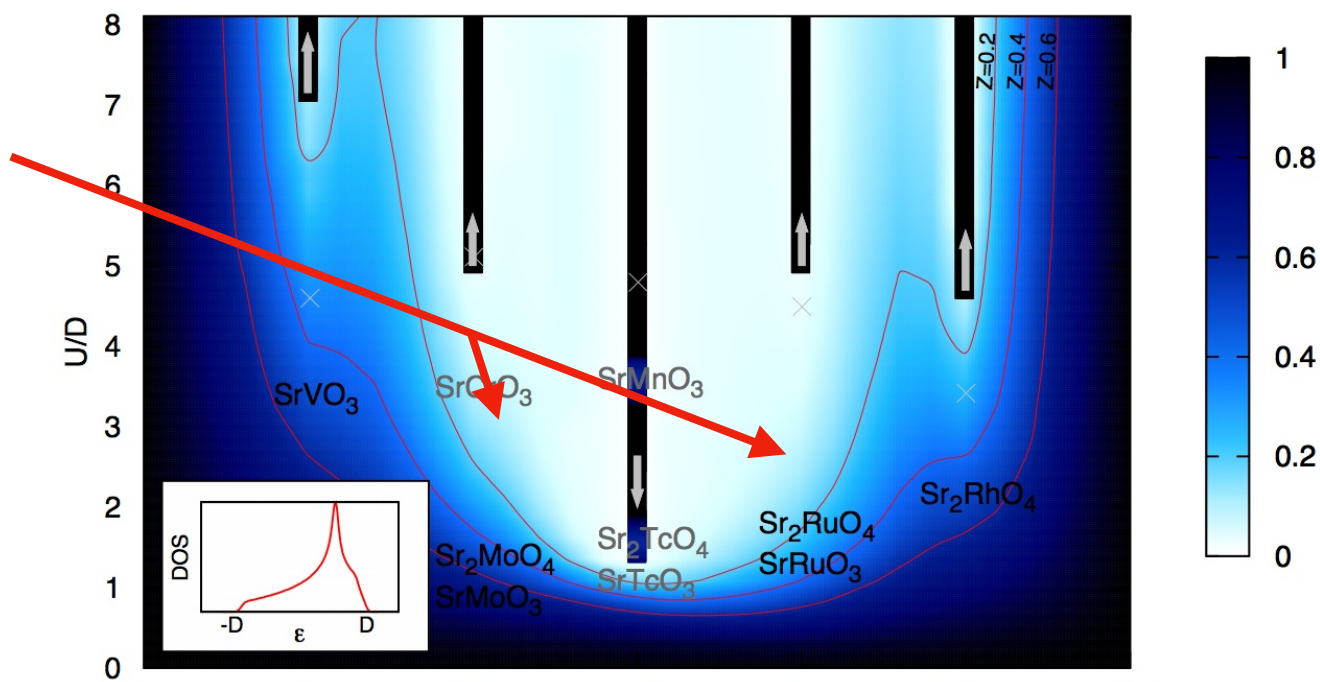
- Effect of Hund's coupling J on the Mott transition and correlation.
  3 orbitals, N= 1,2,3 electrons.

- Strongly correlated metal far from Mott U_c

- Rotationally invariant case $U' = U - 2J$

  J = 0.15U

Z vs U



Hands-on menu

1. Get familiar with Python, simple Green function, operators, matplotlib

2. Your first DMFT code : built yourself a IPT solution for DMFT.

3. Solve DMFT, 1 band Hubbard model, with CT-HYB QMC solver

4. Hund's metal: a two band computation with CT-HYB

5. Cluster DMFT : a minimal two patches DCA cluster.
   Mott transition, Fermi Arcs.

6. Lattice models.
   Lindhard function, Two Particle self-consistent approximation (TPSC)

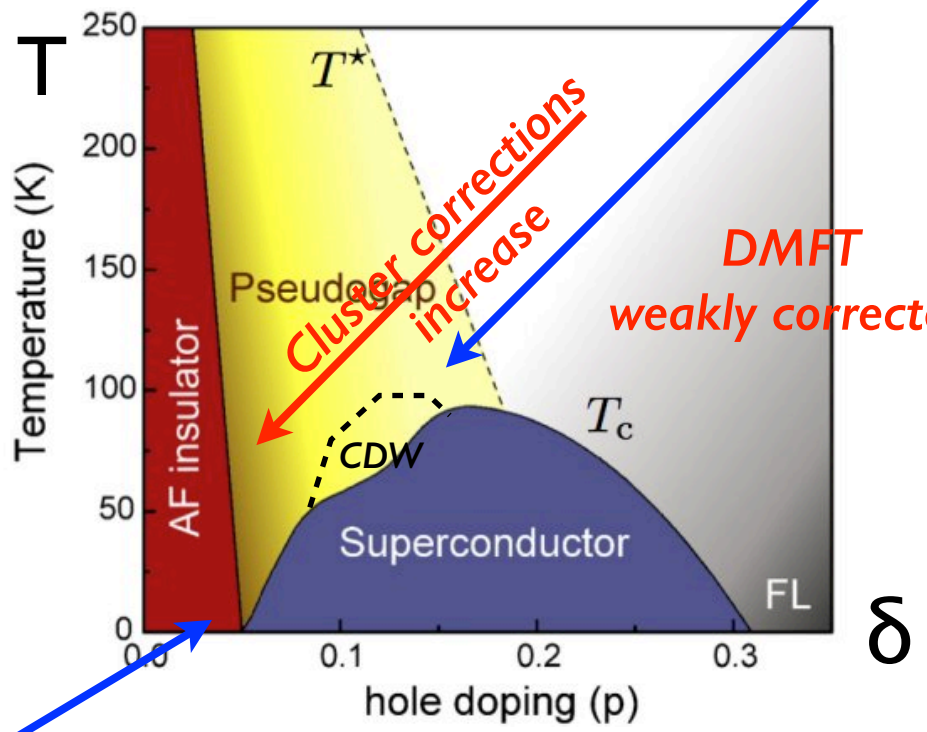
DMFT is high temperature method

"Top to Bottom"

Start from high T/doping

R.G.

Diagrammatic methods



"Bottom to Top"

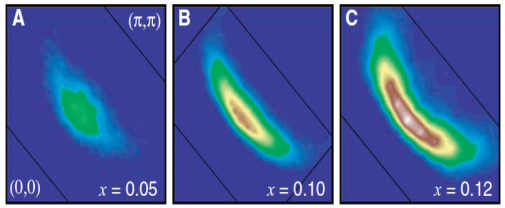
Study the many-body ground state

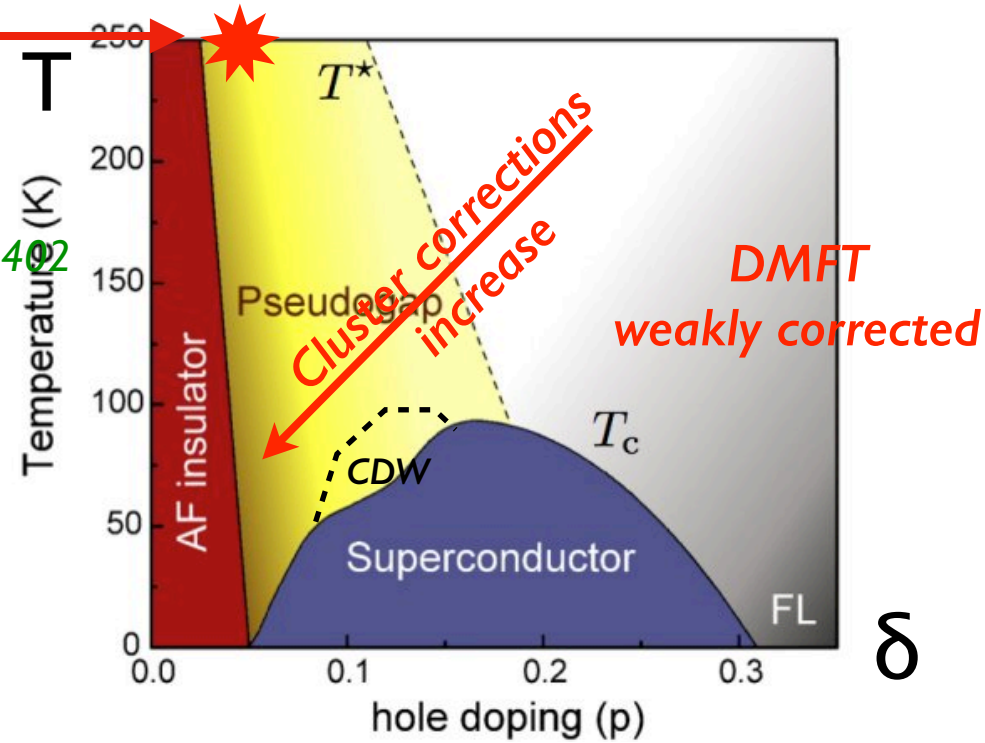
DMRG, PEPS, MERA

Large vs minimal clusters

Converged : Exact solution of Hubbard model in the pseudo-gap !

W. Wu, M. Ferrero, A. Georges, E. Kozik. Arxiv: 1608.08402



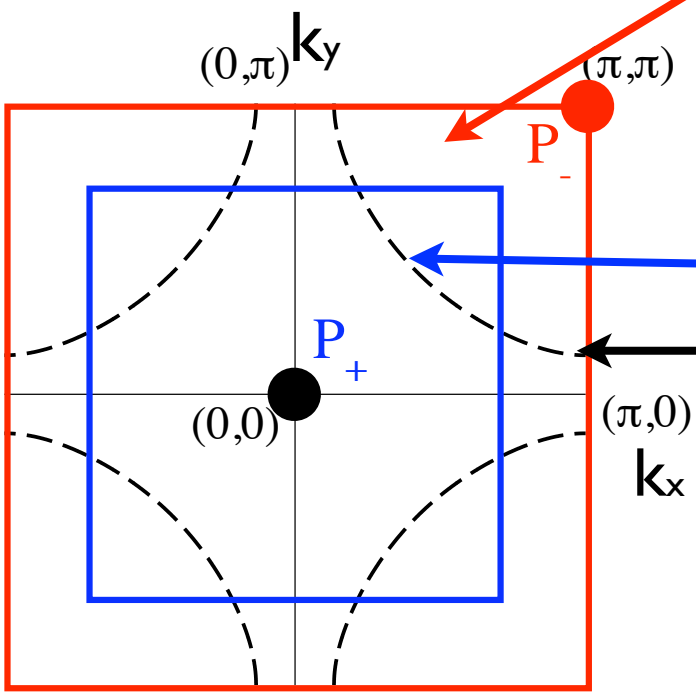


- At high T or δ, intermediate U:
  - Exact solution : can large clusters converge before the sign problem kills the "impurity solver" ?
- At lower T, δ
  - Small clusters capture some important effects (pseudogap, d-SC). Minimal cluster ? Physical picture ?

Minimal cluster DMFT for Fermi Arcs

M. Ferrero, P. S. Cornaglia, L. De Leo, O. Parcollet, G. Kotliar, A. Georges, EPL and PRB 2009

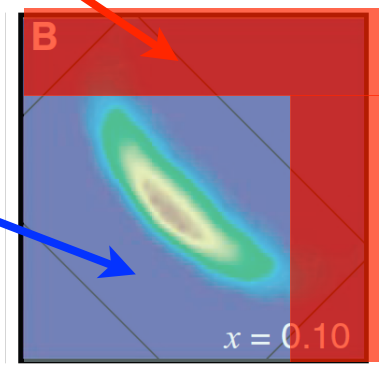
- Two patches patches P+, P- (of equal volume)

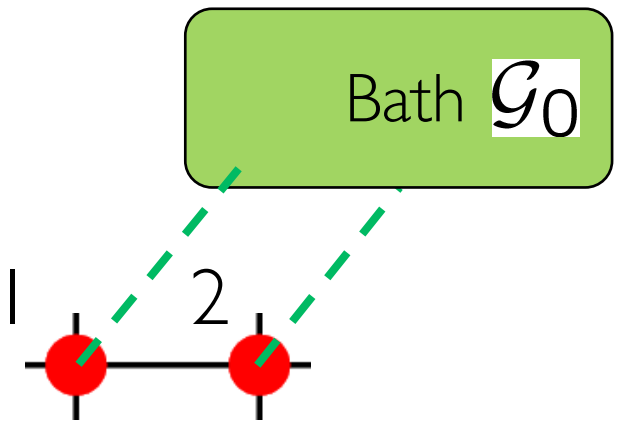


Outer patch P- : Antinodes

Inner patch P+ : Nodes

Free Fermi surface





Two-site Anderson impurity model

Hands-on menu

1. Get familiar with Python, simple Green function, operators, matplotlib

2. Your first DMFT code : built yourself a IPT solution for DMFT.

3. Solve DMFT, 1 band Hubbard model, with CT-HYB QMC solver

4. Hund's metal: a two band computation with CT-HYB

5. Cluster DMFT : a minimal two patches DCA cluster.
   Mott transition, Fermi Arcs.

6. Lattice models.
   Lindhard function, Two Particle self-consistent approximation (TPSC)

A full DMFT computation in 1 slide

Solving DMFT : iterative method

Impurity solver *Cf M. Ferrero's lecture tomorrow*

$$S_{\rm eff} = -\int\int_0^\beta d\tau d\tau' c_\sigma^\dagger(\tau)\mathcal{G}_\sigma^{-1}(\tau-\tau')c_\sigma(\tau') + \int_0^\beta d\tau\ U n_\uparrow(\tau) n_\downarrow(\tau)$$

$$G_{\sigma\rm imp}(\tau) \equiv -\left\langle T c_\sigma(\tau) c_\sigma^\dagger(0)\right\rangle_{S_{\rm eff}}$$

$$\Sigma_{\sigma\rm imp}(i\omega_n) \equiv \mathcal{G}_\sigma^{-1}(i\omega_n) - G_{\sigma\rm imp}^{-1}(i\omega_n)$$

$\mathcal{G}$

$G_{\rm imp}$, $\Sigma_{\rm imp}$

Self consistency condition

$$G_{\sigma\rm imp}[\mathcal{G}](i\omega_n) = \sum_k \frac{1}{i\omega_n + \mu - \epsilon_k - \Sigma_{\sigma\rm imp}[\mathcal{G}](i\omega_n)}$$

- In practice, the iterative loop is (almost) always convergent.

Depends only the d.o.s of free electrons

- The k dependence is only through ε_K for the impurity problem
- Density of states for ε_K

$$D(\epsilon) \equiv \sum_k \delta(\epsilon - \epsilon_k)$$

- Self-consistency condition is a Hilbert transform

$$\tilde{D}(z) \equiv \int d\epsilon \frac{D(\epsilon)}{z - \epsilon} \qquad \text{for} \quad z \in \mathbb{C}$$

$$\begin{aligned} G_{\sigma\text{imp}}[\mathcal{G}](i\omega_n) &= \sum_k \frac{1}{i\omega_n + \mu - \epsilon_k - \Sigma_{\sigma\text{imp}}[\mathcal{G}](i\omega_n)} \\ &= \tilde{D}(i\omega_n + \mu - \Sigma_{\sigma\text{imp}}[\mathcal{G}](i\omega_n)) \end{aligned}$$

Semi circular d.o.s

- A simpler case, when the d.o.s is a semi-circular

$$D(\epsilon)=\frac{1}{2\pi t^2}\sqrt{4t^2-\epsilon^2},\quad |\epsilon|<2t.$$

- Its Hilbert transform can be done explicitly

$$\tilde{D}(\zeta)\equiv\int_{-\infty}^{+\infty}d\epsilon\,\frac{D(\epsilon)}{\zeta-\epsilon},\quad R[\tilde{D}(\zeta)]=\zeta.$$

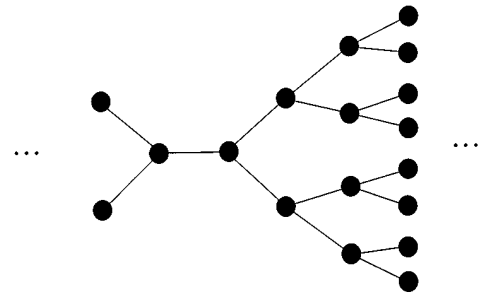
$$\tilde{D}(\zeta)=(\zeta-s\sqrt{\zeta^2-4t^2})/2t^2,\quad R(G)=t^2G+1/G\qquad s=\mathrm{sgn}[\mathrm{Im}(\zeta)]$$

$$G_{\sigma\mathrm{imp}}(i\omega_n)=\tilde{D}(i\omega_n+\mu-\Sigma_{\sigma\mathrm{imp}}(i\omega_n))$$
$$R[G_{\sigma\mathrm{imp}}](i\omega_n)=i\omega_n+\mu-\Sigma_{\sigma\mathrm{imp}}(i\omega_n)$$
$$t^2G_{\sigma\mathrm{imp}}(i\omega_n)+G^{-1}_{\sigma\mathrm{imp}}(i\omega_n)=i\omega_n+\mu-\mathcal{G}^{-1}_{\sigma}(i\omega_n)+G^{-1}_{\sigma\mathrm{imp}}(i\omega_n)$$

$$\mathcal{G}^{-1}_{\sigma}(i\omega_n)=i\omega_n+\mu-\underbrace{t^2G_{\sigma\mathrm{imp}}(i\omega_n)}_{\Delta_\sigma(i\omega_n)}$$

Bethe lattice : summary of equations

- DMFT on the Bethe lattice at $z \to \infty$
- Bethe lattice = semi-circular dos
- Physically meaning full, since semi-circular dos is a reasonable shape



$$S_{\rm eff} = -\int\!\!\int_0^\beta d\tau d\tau' c_\sigma^\dagger(\tau)\mathcal{G}_\sigma^{-1}(\tau-\tau')c_\sigma(\tau') + \int_0^\beta d\tau\, U n_\uparrow(\tau) n_\downarrow(\tau)$$

$$G_{\sigma \rm imp}(\tau) \equiv -\left\langle T c_\sigma(\tau) c_\sigma^\dagger(0)\right\rangle_{S_{\rm eff}}$$

$$\mathcal{G}_\sigma^{-1}(i\omega_n) = i\omega_n + \mu - \underbrace{t^2 G_{\sigma \rm imp}(i\omega_n)}_{\Delta_\sigma(i\omega_n)}$$

- Goal: Solve DMFT equations, self-consistently with CT-INT.

How to do it ?

- Break the DMFT computation into small parts and assemble the computation.

- Which parts ?
  - Local Green functions
  - An impurity solver: e.g. the CT-INT solver.
  - Save the result.
  - Plot it.

Assemble a DMFT computation in 1 slide

- A complete code, using a CT-INT solver (one of the TRIQS apps).
- In Python, with parallelization included (mpi).

- Do not worry about the details of the syntax yet,
  the hands-on are here for that

DMFT computation in 1 slide

```
from pytriqs.gf import *
from ctint_tutorial import CtintSolver


U = 2.5              # Hubbard interaction
mu = U/2.0           # Chemical potential
half_bandwidth=1.0 # Half bandwidth (energy unit)
beta = 40.0          # Inverse temperature
n_iw = 128           # Number of Matsubara frequencies
n_cycles = 10000     # Number of MC cycles
delta = 0.1          # delta parameter
n_iterations = 21  # Number of DMFT iterations

S = CtintSolver(beta, n_iw) # Initialize the solver

S.G_iw << SemiCircular(half_bandwidth) # Initialize the Green's function
```

- Import some basic blocks (Green function, a solver) ...
- Define some parameters and declare a CT-INT solver S
- All TRIQS solvers contains G, G_0, Σ as members with the correct β, dimensions, etc...
- Initialize S.G_iw to a (the Hilbert transform of a) semi-circular dos.

DMFT computation in 1 slide

```
from pytriqs.gf import *
from ctint_tutorial import CtintSolver



U = 2.5              # Hubbard interaction
mu = U/2.0           # Chemical potential
half_bandwidth=1.0 # Half bandwidth (energy unit)
beta = 40.0          # Inverse temperature
n_iw = 128           # Number of Matsubara frequencies
n_cycles = 10000     # Number of MC cycles
delta = 0.1          # delta parameter
n_iterations = 21  # Number of DMFT iterations

S = CtintSolver(beta, n_iw) # Initialize the solver

S.G_iw << SemiCircular(half_bandwidth) # Initialize the Green's function

for sigma, G0 in S.G0_iw: # sigma = 'up', 'down'
  G0 << inverse(iOmega_n + mu - (half_bandwidth/2.0)**2 * S.G_iw[sigma] ) # Set G0
```

$$G_{0\sigma}^{-1}(i\omega_n) = i\omega_n + \mu - t^2 G_{c\sigma}(i\omega_n), \quad \text{for } \sigma = \uparrow, \downarrow$$

- Implement DMFT self-consistency condition

DMFT computation in 1 slide

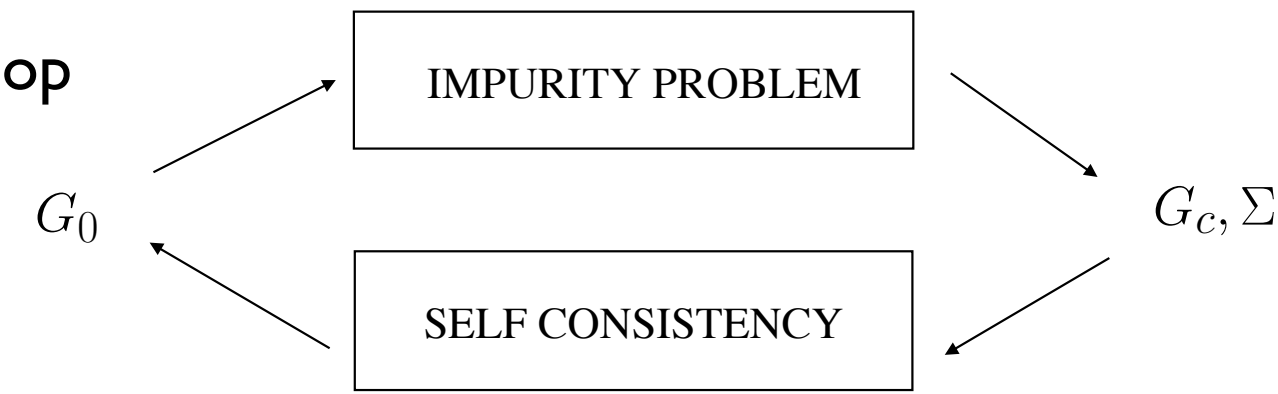
```
from pytriqs.gf import *
from ctint_tutorial import CtintSolver



U = 2.5              # Hubbard interaction
mu = U/2.0           # Chemical potential
half_bandwidth=1.0 # Half bandwidth (energy unit)
beta = 40.0          # Inverse temperature
n_iw = 128           # Number of Matsubara frequencies
n_cycles = 10000     # Number of MC cycles
delta = 0.1          # delta parameter
n_iterations = 21  # Number of DMFT iterations

S = CtintSolver(beta, n_iw) # Initialize the solver

S.G_iw << SemiCircular(half_bandwidth) # Initialize the Green's function

for sigma, G0 in S.G0_iw:
  G0 << inverse(iOmega_n + mu - (half_bandwidth/2.0)**2 * S.G_iw[sigma] ) # Set G0

S.solve(U, delta, n_cycles) # Solve the impurity problem
```

- Call the solver.
- From $G_0(i\omega_n)$ (and various parameters), it computes $G(i\omega_n)$.

DMFT computation in 1 slide

```
from pytriqs.gf import *
from ctint_tutorial import CtintSolver



U = 2.5              # Hubbard interaction
mu = U/2.0           # Chemical potential
half_bandwidth=1.0 # Half bandwidth (energy unit)
beta = 40.0          # Inverse temperature
n_iw = 128           # Number of Matsubara frequencies
n_cycles = 10000     # Number of MC cycles
delta = 0.1          # delta parameter
n_iterations = 21  # Number of DMFT iterations

S = CtintSolver(beta, n_iw) # Initialize the solver

S.G_iw << SemiCircular(half_bandwidth) # Initialize the Green's function

for it in range(n_iterations): # DMFT loop
  for sigma, G0 in S.G0_iw:
    G0 << inverse(iOmega_n + mu - (half_bandwidth/2.0)**2 * S.G_iw[sigma] ) # Set G0

  S.solve(U, delta, n_cycles) # Solve the impurity problem
```

- DMFT iteration loop



DMFT computation in 1 slide

```
from pytriqs.gf import *
from ctint_tutorial import CtintSolver



U = 2.5              # Hubbard interaction
mu = U/2.0           # Chemical potential
half_bandwidth=1.0 # Half bandwidth (energy unit)
beta = 40.0          # Inverse temperature
n_iw = 128           # Number of Matsubara frequencies
n_cycles = 10000     # Number of MC cycles
delta = 0.1          # delta parameter
n_iterations = 21  # Number of DMFT iterations

S = CtintSolver(beta, n_iw) # Initialize the solver

S.G_iw << SemiCircular(half_bandwidth) # Initialize the Green's function

for it in range(n_iterations): # DMFT loop
  for sigma, G0 in S.G0_iw:
    G0 << inverse(iOmega_n + mu - (half_bandwidth/2.0)**2 * S.G_iw[sigma] ) # Set G0

  S.solve(U, delta, n_cycles) # Solve the impurity problem

  G_sym = (S.G_iw['up'] + S.G_iw['down'])/2 # Impose paramagnetic solution
  S.G_iw << G_sym
```

- Enforce the fact that the solution is paramagnetic, cf DMFT lecture. (noise in the QMC would lead to a AF solution after iterations).

DMFT computation in 1 slide

```
from pytriqs.gf import *
from ctint_tutorial import CtintSolver
from pytriqs.archive import HDFArchive


U = 2.5              # Hubbard interaction
mu = U/2.0           # Chemical potential
half_bandwidth=1.0 # Half bandwidth (energy unit)
beta = 40.0          # Inverse temperature
n_iw = 128           # Number of Matsubara frequencies
n_cycles = 10000     # Number of MC cycles
delta = 0.1          # delta parameter
n_iterations = 21  # Number of DMFT iterations

S = CtintSolver(beta, n_iw) # Initialize the solver

S.G_iw << SemiCircular(half_bandwidth) # Initialize the Green's function

for it in range(n_iterations): # DMFT loop
  for sigma, G0 in S.G0_iw:
    G0 << inverse(iOmega_n + mu - (half_bandwidth/2.0)**2 * S.G_iw[sigma] ) # Set G0

  S.solve(U, delta, n_cycles) # Solve the impurity problem

  G_sym = (S.G_iw['up'] + S.G_iw['down'])/2 # Impose paramagnetic solution
  S.G_iw << G_sym

  with HDFArchive("dmft_bethe.h5",'a') as A:
    A['G%i'%it] = G_sym # Save G from every iteration to file as G1, G2, G3....
```

- Accumulate the various iterations in a (hdf5) file

DMFT computation in 1 slide

```
from pytriqs.gf import *
from ctint_tutorial import CtintSolver
from pytriqs.archive import HDFArchive


U = 2.5             # Hubbard interaction
mu = U/2.0          # Chemical potential
half_bandwidth=1.0 # Half bandwidth (energy unit)
beta = 40.0         # Inverse temperature
n_iw = 128          # Number of Matsubara frequencies
n_cycles = 10000    # Number of MC cycles
delta = 0.1         # delta parameter
n_iterations = 21   # Number of DMFT iterations

S = CtintSolver(beta, n_iw) # Initialize the solver

S.G_iw << SemiCircular(half_bandwidth) # Initialize the Green's function

for it in range(n_iterations): # DMFT loop
  for sigma, G0 in S.G0_iw:
    G0 << inverse(iOmega_n + mu - (half_bandwidth/2.0)**2 * S.G_iw[sigma] ) # Set G0

  # Change random number generator on final iteration
  random_name = 'mt19937' if it < n_iterations-1 else 'lagged_fibonacci19937'

  S.solve(U, delta, n_cycles, random_name=random_name) # Solve the impurity problem

  G_sym = (S.G_iw['up']+S.G_iw['down'])/2 # Impose paramagnetic solution
  S.G_iw << G_sym

  with HDFArchive("dmft_bethe.h5",'a') as A:
     A['G%i'%it] = G_sym # Save G from every iteration to file
```

- Change the random generator at the last iteration !

Look at the result (in IPython notebook)

```
A = HDFArchive("dmft_bethe.h5",'r') # Open file in read mode
for it in range(21):
    if it%2: # Plot every second result
    oplot(A['G%i'%it], '-o', mode='I', name='G%i'%it)
```

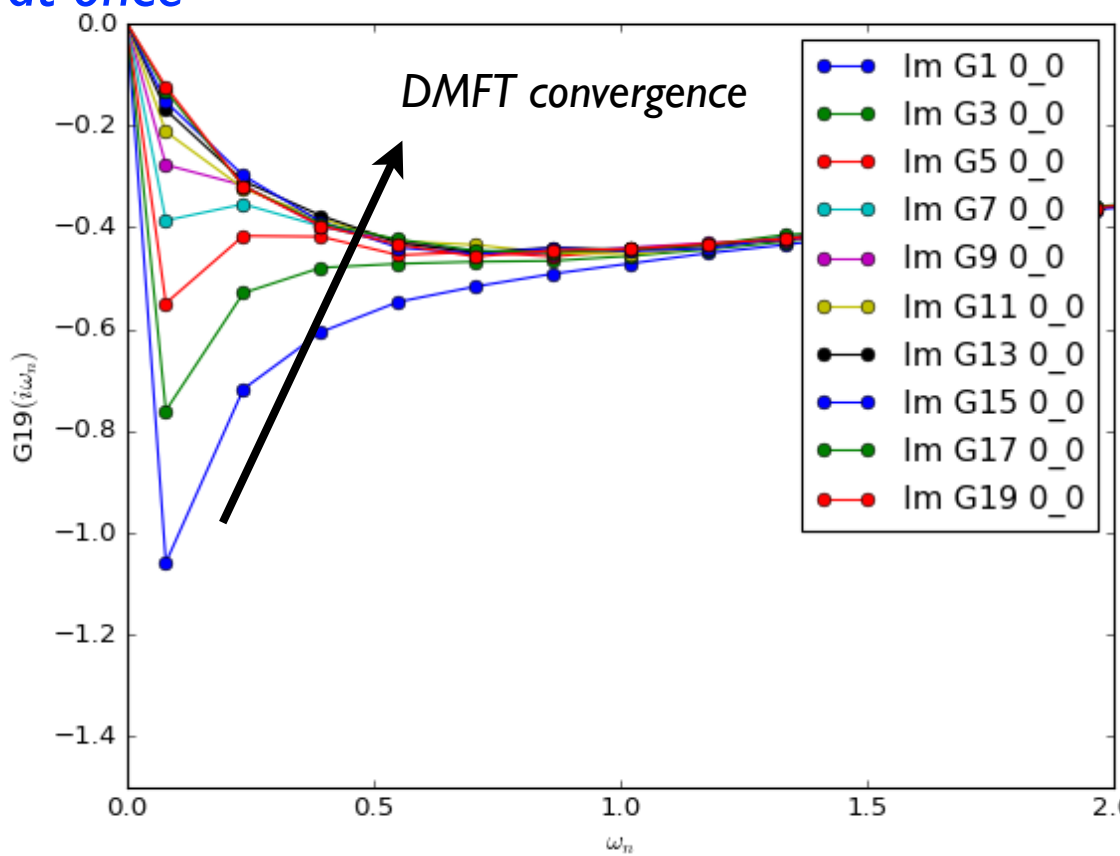
oplot can plot many TRIQS objects via matplotlib

Retrieve Gi from the file, and use it at once

Imaginary part only

NB

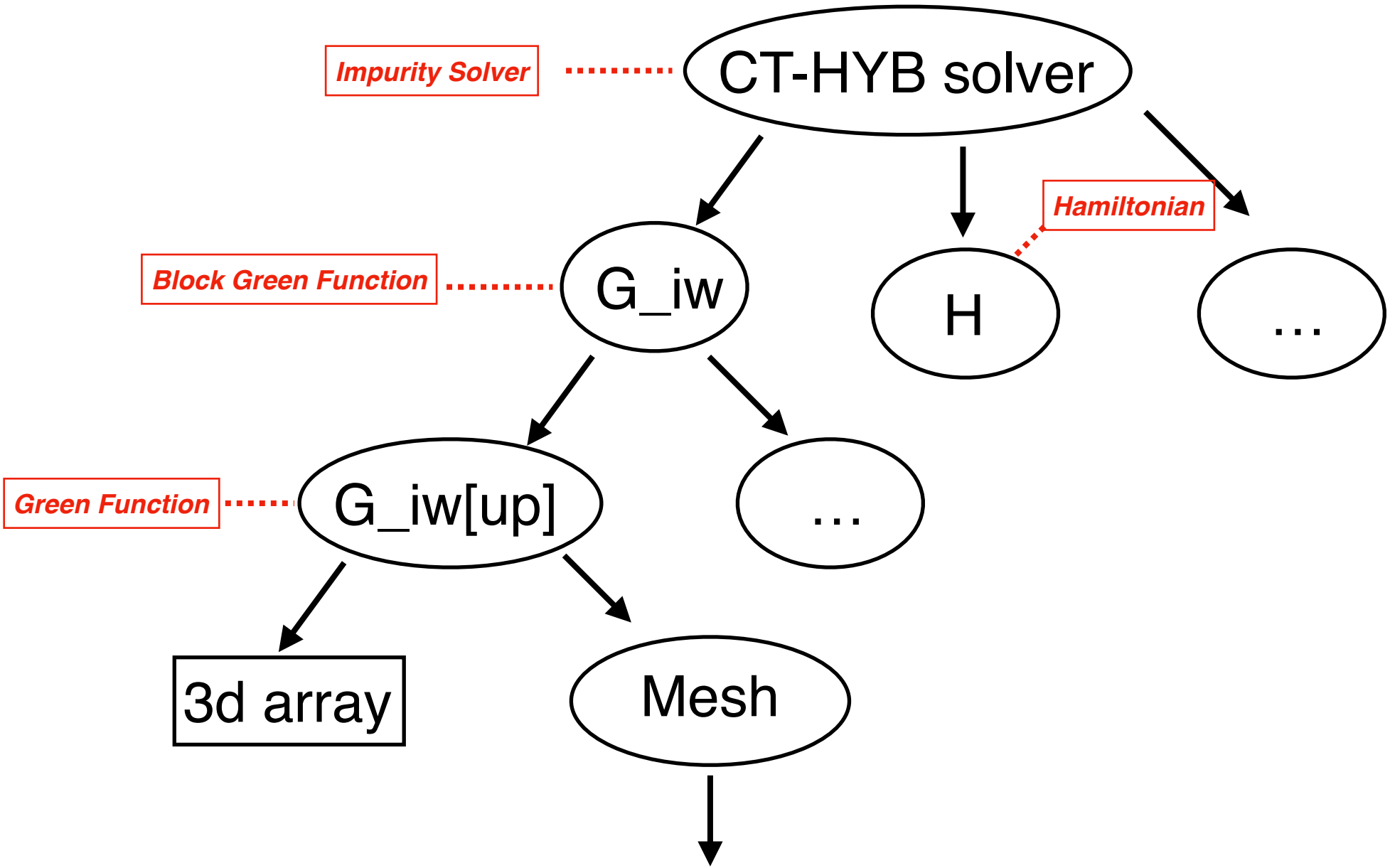
lines are guide to the eyes, only Matsubara frequency point matters



HDF5

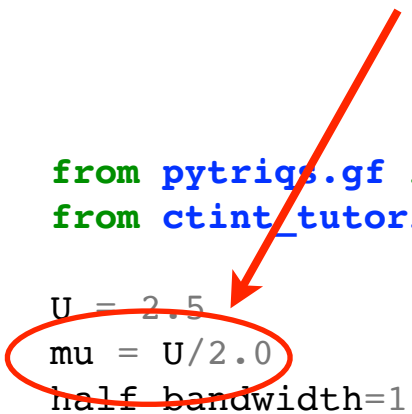
- Language agnostic (python, C/C++, F90).
- Compressed binary format hence compact, but also portable.
- Dump & reload objects in one line.
  Forget worrying about format, reading files, conventions.
- $G(\omega)(n_1,n_2)$ a 3d array of complex numbers, i.e. 4d array of reals.
  No natural convention in a 2d text file.

HDF5 : hierarchical tree structure, like directory



FAQ : Where was the input file ?

- Traditional way: a monolithic program, input files, output files.
- TRIQS way : write your own script with full control
- No input file as text : no need to parse it,
  we can do operations on the fly, use Python to prepare data ...



```
from pytriqs.gf import *
from ctint_tutorial import CtintSolver

U = 2.5                 # Hubbard interaction
mu = U/2.0              # Chemical potential
half_bandwidth=1.0 # Half bandwidth (energy unit)
beta = 40.0             # Inverse temperature
n_iw = 128              # Number of Matsubara frequencies
n_cycles = 10000    # Number of MC cycles
delta = 0.1             # delta parameter
n_iterations = 21   # Number of DMFT iterations
```

• • •

Summarize what we have done so far

- A fully functional DMFT code.
- Computation & data analysis: all in Python.
- Green functions, solvers as Python classes.
- Since it is a script, it is easy to change various details, e.g.
  - Self-consistency condition: enforce paramagnetism
  - Change random generator
  - Change starting point (e.g. reload G from a file).
  - Improve convergence (e.g. mixing quantities over iterations).
  - Measure e.g. susceptibilities only at the end of the DMFT loop

CT-INT implementation

- As a simple example of a TRIQS based application

- https://github.com/TRIQS/tutorials.git, directory cint_tutorial.

- 1 band, CT-INT QMC code.
  - 150 lines of C++.
  - Python wrappings
    automatically generated by TRIQS/cpp2py from the C++ code.

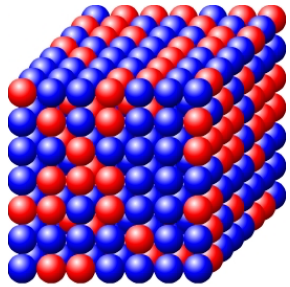
Why a library rather than a monolithic program ?

Library vs monolithic program

- Better to have a language to express your calculation.
- Many-body approaches are quite versatile.
- Illustration with DMFT methods
  - Many impurity solvers (QMC, ED, NCA, IPT, ...)
    Same interface: use several ones, depending on regimes.
  - Various impurity models (# of orbitals, symmetries, clusters)
  - Many self-consistency conditions
  - No "general" DMFT code.

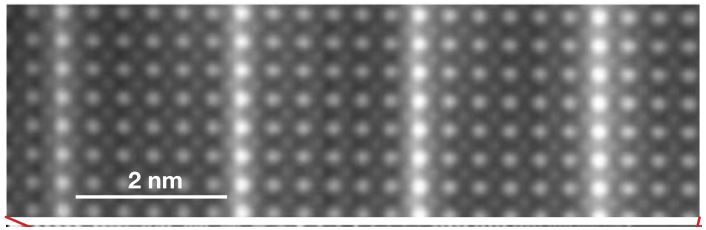
DMFT is quite versatile

- Disordered systems



  - Two impurity models

- Correlated interfaces.



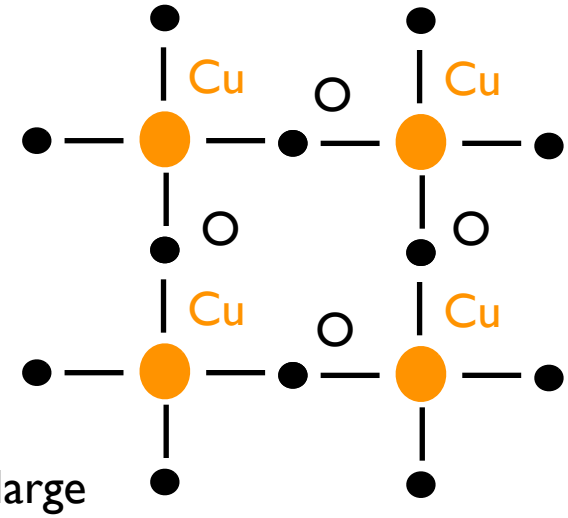
$SrTiO_3/LaTiO_3$

Ohtomo et al, Nature 2002

  - One impurity per layer

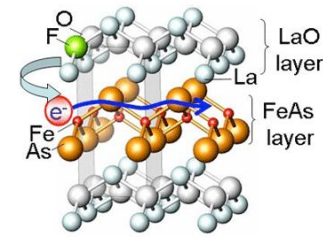
- Multiband/realistic systems

$$\Sigma(\omega) = \begin{pmatrix} \Sigma^{\mathrm{imp}}(\omega) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$



  - Self-consistency in large unit cell (Cu + 2 O)
    $\Sigma_{ab}(\omega)$ a 3x3 matrix
  - Impurity model on Cu, 1 band : $\Sigma^{\mathrm{imp}}(\omega)$ 1x1 matrix

- DFT + DMFT



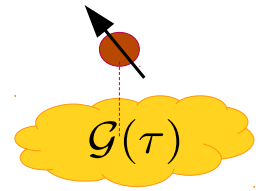
  - Interface with electronic structure codes (project on Wannier functions, etc).

Cluster DMFT

Cf lecture by D. Sénéchal

- DMFT: 1 atom (Anderson impurity) + effective self-consistent bath

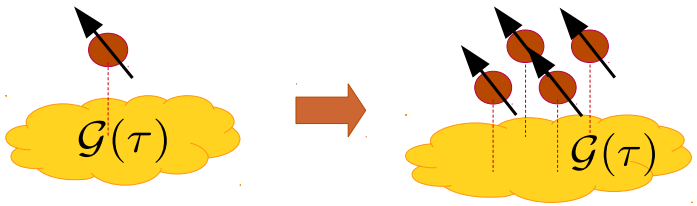
$$\Sigma(k,\omega) \approx \Sigma_{\text{impurity}}(\omega)$$



- Clusters = a systematic expansion around DMFT.
- Control parameter = size of cluster / momentum resolution.
  Systematic benchmarks, cf *J. LeBlanc et al., PRX 5 (2015)*

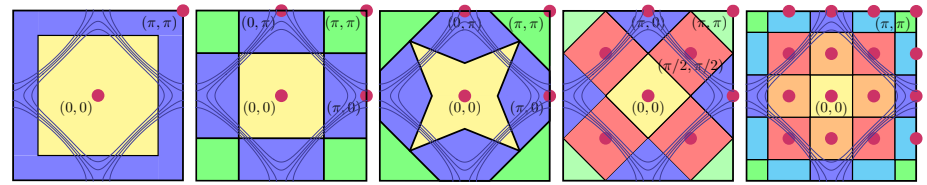
Real space clusters (C-DMFT)

Lichtenstein, Katsnelson 2000
Kotliar et al. 2001



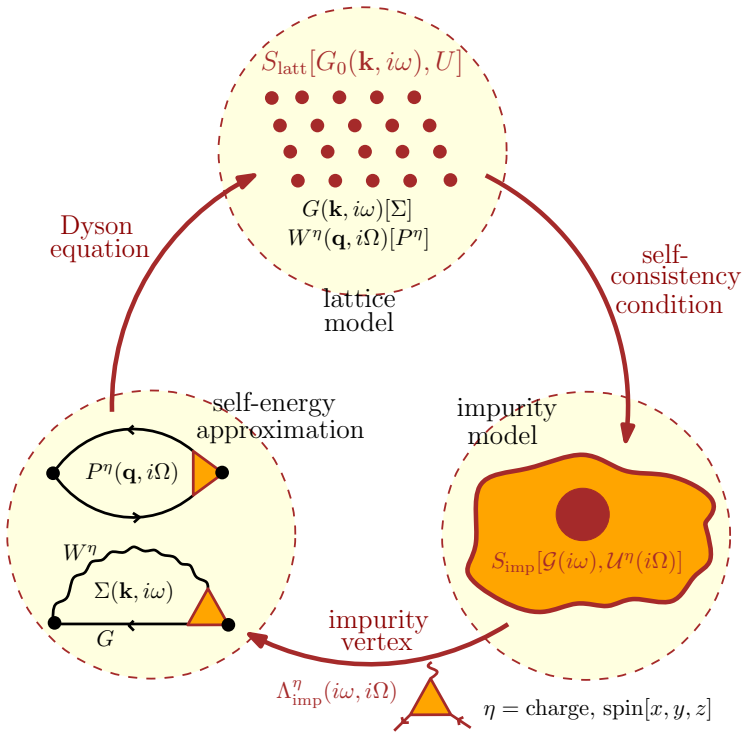
Reciprocal space (DCA)
clusters Brillouin zone patching

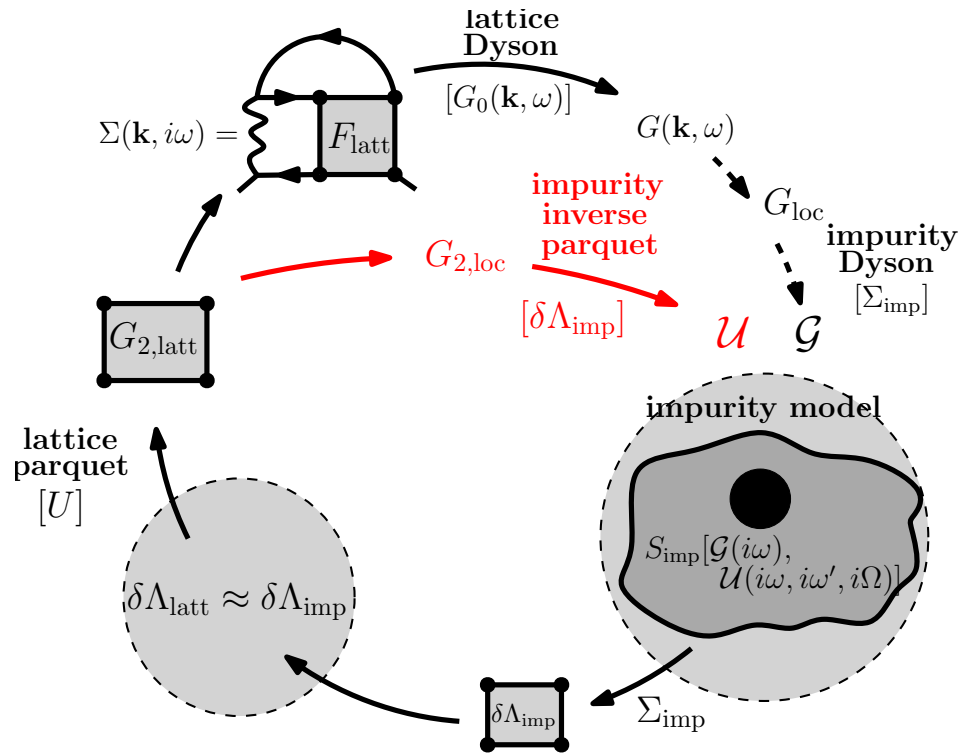
Hettler et al. '98, ...



Beyond DMFT

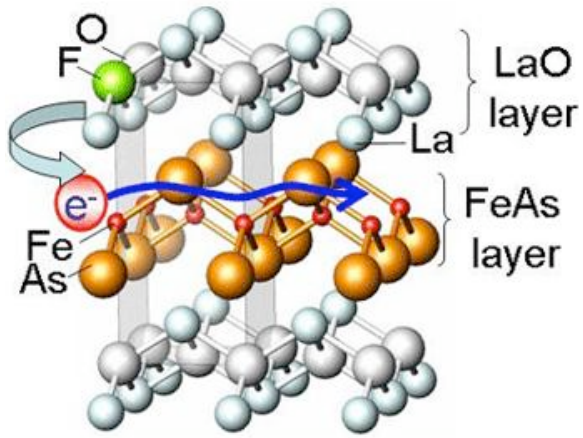
- Atomic approximation but for which quantity ?
- DMFT is only the simplest of a family of approximations.
- Local Mott physics + spin-fluctuations, Bethe Salpeter, Parquet. e.g. DΓA *Toschi '07,* Trilex, Quadrilex, *T.Ayral, O.Parcollet, 2015-2016*
- Need to assemble more complex methods with more complex objects : G(k,ω), Vertex Γ(ω,ν,ν').



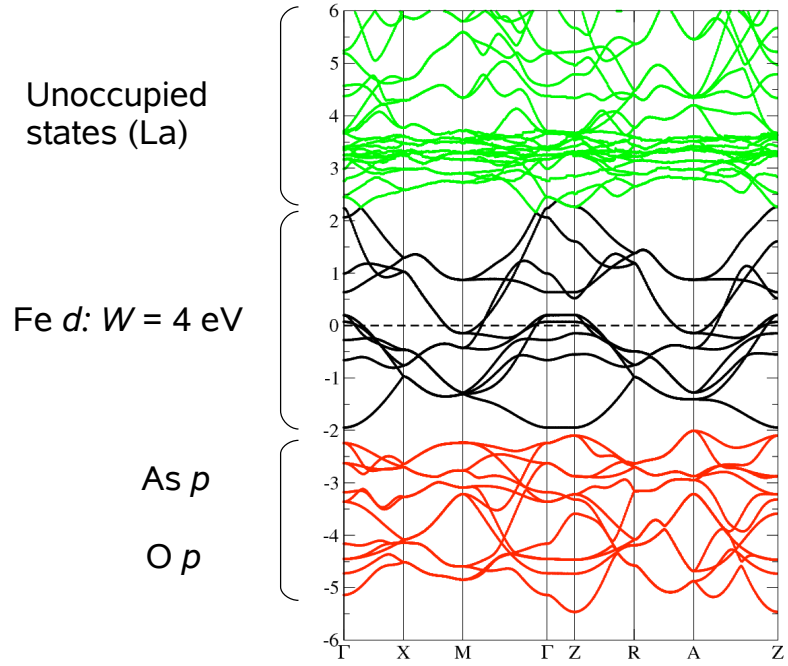


Mix with electronic structure codes

- Cf Lectures next week.
- (Much) more of the same kind of manipulations:
  Extract the Green function of the correlated orbitals.
  Project on Wannier functions.
  Embed the self-energy of the correlated orbital (downfolding).

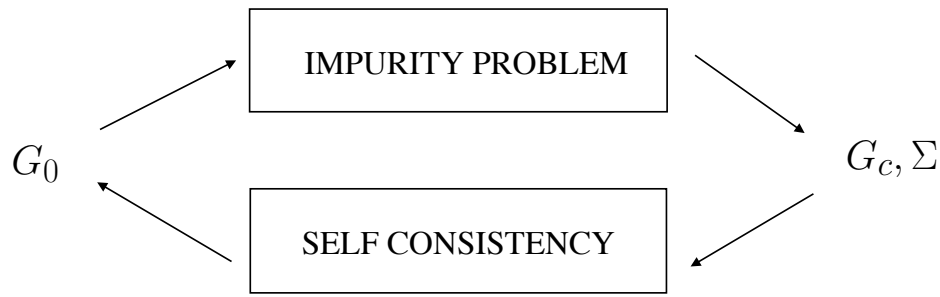


Fe-Based (2008)



Need for a library

Hard but well defined problem



Easy to write, but many variants

- No "general" DMFT code.
- A simple language able to write all of these, and much more.
- A library : extending Python/C++ to express the basic concepts of our field (Green functions, Bravais lattice, Brillouin zone, etc).
- Questions:
  design : which blocks ? how to compose them ?

TRIQS package : overview

Technical goals

- Basic blocks for our field (e.g. Green functions, MC tools).
- Simplicity : what is simple should be coded simply !
- High performance :
  - Human time : reduce the cost of writing codes.
  - Machine time : run quickly. Zero cost abstraction (modern C++).

Reproducibility & Correctness

- Codes should be open source
  Diminish development time/effort with libraries
- Code clarity : written to be read/understood.
  Libraries (std, triqs, …) make code smaller, easier to understand.
  Code review.
- Version control, test driven development.
  tools : git, google test.
- Easy installation, with environment.
  "Container" techniques : docker, singularity

TRIQS project : a modular structure

CTHYB
Impurity solver

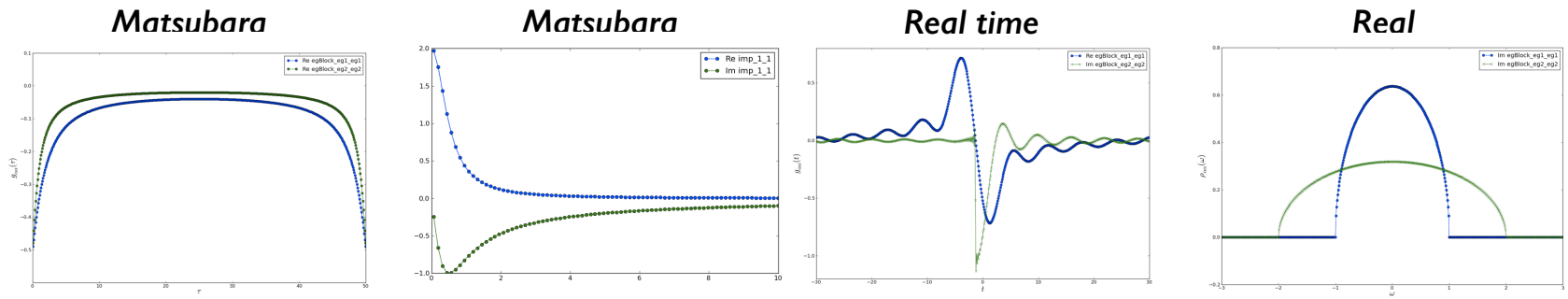
DFTTools
Interface to electronic structure codes

Your app.

TRIQS library
The basic blocks

TRIQS library

- "Green functions" containers (any function on a mesh)
  G(ω), G(r, τ), G(k,ω), Vertex Γ(ω,ν,ν').
- Tools for Monte-Carlo



- Many-body operators to write Hamiltonians.
- Basic solid state physics notion : Bravais Lattices, Brillouin zone, density of states, Hilbert transform.
- Interfaces to save/load in HDF5 files,
  plot interactively in the ipython notebook.

TRIQS/CTHYB.

A state of the art quantum impurity solver for multi-orbital systems

Cf M. Ferrero's lecture

TRIQS/CTHYB

https://triqs.ipht.cnrs.fr/1.x/applications/cthyb/

cthyb

a generic quantum impurity solver based on the triqs library

Install Documentation Issues About CTHYB

The hybridization-expansion solver

The TRIQS-based hybridization-expansion solver allows to solve the generic problem of a quantum impurity embedded in a conduction bath for an arbitrary local interaction vertex. The "impurity" can be any set of orbitals, on one or several atoms. To be more specific, the Hamiltonian of the problem has the form:

$$\hat{H} = \sum_{k,\alpha} \epsilon_{k,\alpha} c^\dagger_{k,\alpha} c_{k,\alpha} + \sum_{k,\alpha} V_{k,\alpha} (c^\dagger_{k,\alpha} d_\alpha + h.c.) - \mu \sum_\alpha d^\dagger_\alpha d_\alpha + \hat{\sum}_{\alpha\beta} h_{\alpha\beta} d^\dagger_\alpha d_\beta + \frac{1}{2} \sum_{\alpha\beta\gamma\delta} U_{\alpha\beta\gamma\delta} d^\dagger_\alpha d^\dagger_\beta d_\delta d_\gamma .$$

Here the operators $c^\dagger$ construct a fermion in the bath, while the operators $d^\dagger$ construct a fermion on the impurity. In this problem, the hybridization function Δ between the bath and the impurity is given by:

$$\Delta_{\alpha,\beta}(i\omega_n) = \sum_k \frac{V_{k,\alpha} V^*_{k,\beta}}{i\omega_n - \epsilon_{k,\alpha}},$$

so that the non-interacting Green's function of the impurity is:

$$\hat{G}_0^{-1}(i\omega_n) = i\omega_n + \mu - \hat{h} - \hat{\Delta}(i\omega_n).$$

With the knowledge of G_0 and the matrix $U_{\alpha\beta\gamma\delta}$, the quantum impurity solvers find the interacting Green's function G of the problem. Learn how to use it in the Documentation.

Quick search

Go

Enter search terms or a module, class or function name.

CTHYB

- Expansion in coupling to the bath
- “Any" local Hamiltonian (3, 5 bands, low temperature), including spin-orbit.

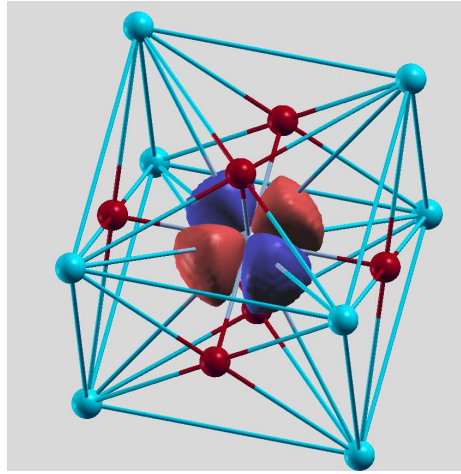
TRIQS/DFTTools.

Interface to electronic structure codes

TRIQS / DFTTools

Markus Aichhorn, Leonid Pourovskii, Priyanka Seth, Veronica Vildosola, Manuel Zingl, Oleg E. Peil, Xiaoyu Deng, Jernej Mravlje, Gernot J. Kraberger, Cyril Martins, M. Ferrero, O. Parcollet Comp. Phys. Comm. 204, 200 (2016), arXiv:1511.01302

- Ab-initio DMFT + DFT.
  Cf lectures by Kotliar & Haule next week.
- A TRIQS python interface with a growing number of electronic structure codes:
  - Wien2k
  - Wannier90
  - Vasp (to be published ?).



$SrVO_3$

Web sites

TRIQS web site

https://triqs.github.io/

triqs

a Toolbox for Research on Interacting Quantum Systems

Install Reference Tutorials Applications Issues About TRIQS

Welcome

TRIQS (**T**oolbox for **R**esearch on **I**nteracting **Q**uantum **S**ystems) is a scientific project providing a set of C++ and Python libraries to develop new tools for the study of interacting quantum systems.

The goal of this toolkit is to provide high level, efficient and simple to use libraries in C++ and Python, and to promote the use of modern programming techniques.

TRIQS is free software distributed under the GPL license.

TRIQS 1.4-dev

This is the homepage of the TRIQS release 1.4 (in development). For the changes in 1.4, Cf changelog page

TRIQS applications

Based on the TRIQS toolkit, several full-fledged applications are also available. They allow for example to solve a generic quantum impurity model or to run a complete LDA+DMFT calculation.

Developed in a collaboration between IPhT Saclay and Ecole Polytechnique since 2005, the TRIQS library and applications have allowed us to address questions as diverse as:

- Momentum-selective aspects on cuprate superconductors (with various cluster DMFT methods)
- Degree of correlation in iron-based superconductors (within an LDA+DMFT approach)
- Fermionic Mott transition and exploration of Sarma phase in cold-atoms

Python & C++

The libraries exist at two complementary levels: on the one hand, C++ libraries allow to quickly develop high-performance low-level codes; on the other hand python libraries implement the most common many-body objects, like Green's functions, that can be manipulated easily in python scripts.

This duality is a real advantage in the development of new many-body tools. Critical parts where performance is essential can be written in C++ (e.g. a quantum impurity solver) while the data analysis, preparation of the inputs or interface with other programs can be done at the very user-friendly python level.

cea

ÉCOLE POLYTECHNIQUE

cnrs

erc

European Research Council

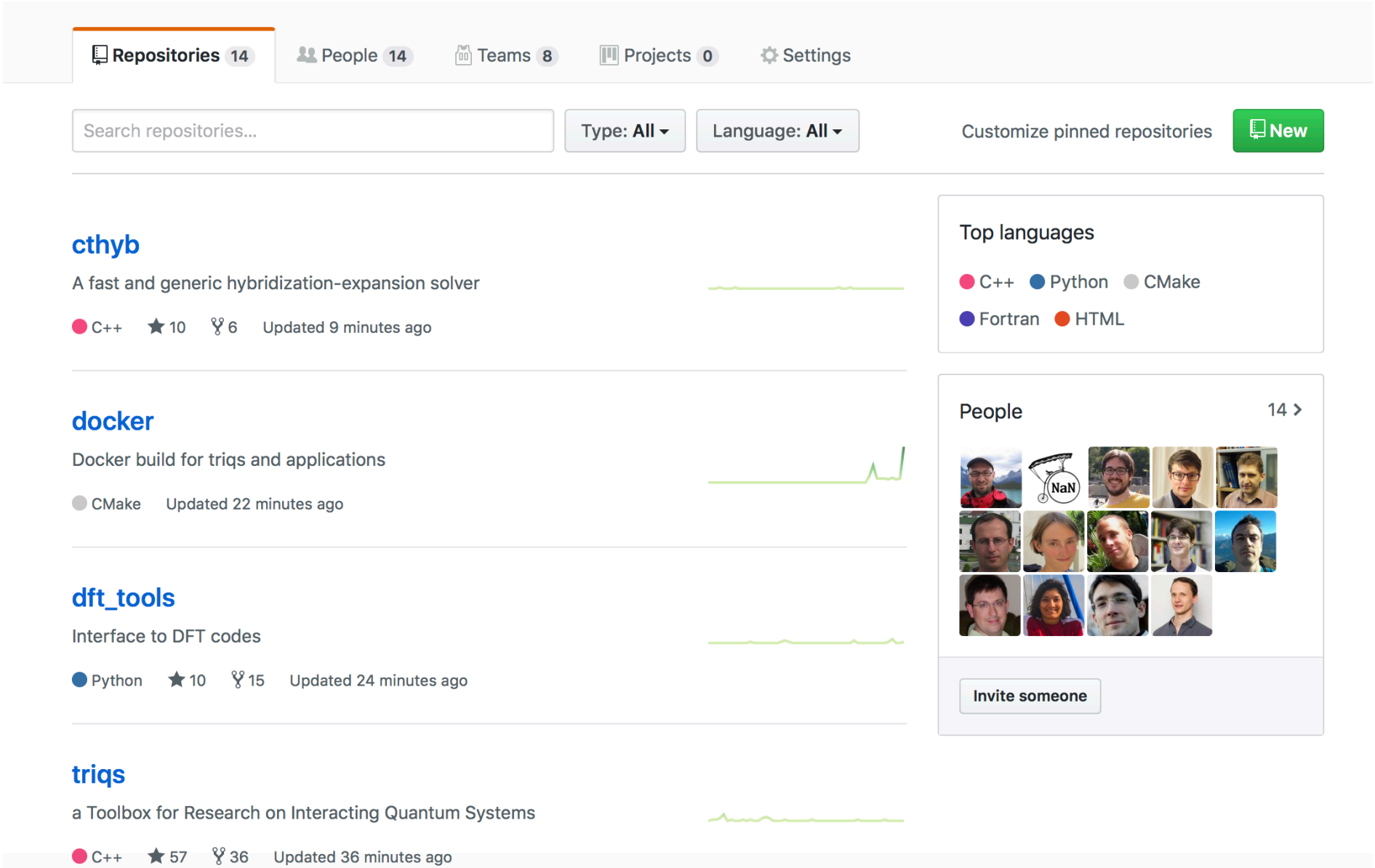
Quick search

Go

Enter search terms or a module, class or function name.

The Github site

https://github.com/TRIQS



TRIQS releases

- 2.x series :
  - Just released 2.0 (used in this school).
  - Upgrade to multiple components.
  - C++17.
  - Packaging : Cf Nils' talk
- 3.x series : Python 3, …

More applications, DMFT solvers, and upgrade

- CT-HYB : Vertex computations.
- CT-HYB "segment picture",
  an optimised code for density-density interaction.
- CT-INT multiband, cluster, with general retarded interaction.
- RISB quick impurity solvers.
- Susceptibility computations in DMFT :
  Bethe-Salpeter, Vertex computations

Thank you for your attention