Automating the implementation of novel EDFs beyond NLO in gradients.

The codes Hephaestos & Tantalus.

W. Ryssens and M. Bender

Center for Theoretical Physics,
Sloane Physics Laboratory,
Yale University.

June 2019

Contents

Yale

Contents

Yale

Skyrme (NLO) functional

Multiple terms

$$E_{\text{tot}} = E_{\text{kin}} + E_{\text{Skyrme}} + E_{\text{Coul}} + E_{\text{cm}} + E_{\text{pair}} \,,$$

The more or less standard Skyrme part is the following

$$E_{\text{Skyrme}} = \sum_{t=0,1} \int d^3r \Big[C_t^{\rho}\, \rho_t^2(\vec{r}) + C_t^{\rho^\alpha}\, \rho_0^{\alpha}(\vec{r})\, \rho_t^2(\vec{r}) + C_t^{\tau}\, \rho_t(\vec{r})\, \tau_t(\vec{r}) + C_t^{\Delta\rho}\, \rho_t(\vec{r})\, \Delta\rho_t(\vec{r}) + C_t^{\nabla\cdot J} \rho_t(\vec{r})\, \nabla\cdot\vec{J}_t(\vec{r}) - C_t^{J} \sum_{\mu,\nu} J_{t,\mu\nu}(\vec{r})\, J_{t,\mu\nu}(\vec{r}) \Big] \,.$$

which has

- Densities: $\rho, \tau, J_{\mu\nu}$
- $\sim$ **10** parameters ...
- ... fitted to a few masses and radii.

Skyrme (NLO) functional

Multiple terms

$$E_{\text{tot}} = E_{\text{kin}} + E_{\text{Skyrme}} + E_{\text{Coul}} + E_{\text{cm}} + E_{\text{pair}} \,,$$

The more or less standard Skyrme part is the following

$$E_{\text{Skyrme}} = \sum_{t=0,1} \int d^3r \Big[C_t^{\rho} \, \rho_t^2(\vec{r}) + C_t^{\rho^{\alpha}} \, \rho_0^{\alpha}(\vec{r}) \, \rho_t^2(\vec{r}) + C_t^{\tau} \, \rho_t(\vec{r}) \, \tau_t(\vec{r}) + C_t^{\Delta\rho} \, \rho_t(\vec{r}) \, \Delta\rho_t(\vec{r}) + C_t^{\nabla\cdot J} \rho_t(\vec{r}) \, \nabla \cdot \vec{J}_t(\vec{r}) - C_t^{I} \sum_{\mu,\nu} J_{t,\mu\nu}(\vec{r}) \, J_{t,\mu\nu}(\vec{r}) \Big] \,.$$

which has

- Densities: $\rho, \tau, J_{\mu\nu}$
- $\sim$ **10** parameters ...
- ... fitted to a few masses and radii.

Easy to calculate!

Yale

Multiple terms

$$E_{\text{tot}} = E_{\text{kin}} + E_{\text{Skyrme}} + E_{\text{Coul}} + E_{\text{cm}} + E_{\text{pair}} \,,$$

The more or less standard Skyrme part is the following

$$E_{\text{Skyrme}} = \sum_{t=0,1} \int d^3r \Big[C_t^{\rho} \, \rho_t^2(\vec{r}) + C_t^{\rho^{\alpha}} \, \rho_0^{\alpha}(\vec{r}) \, \rho_t^2(\vec{r}) + C_t^{\tau} \, \rho_t(\vec{r}) \, \tau_t(\vec{r}) + C_t^{\Delta\rho} \, \rho_t(\vec{r}) \, \Delta\rho_t(\vec{r}) + C_t^{\nabla\cdot J} \rho_t(\vec{r}) \, \nabla \cdot \vec{J}_t(\vec{r}) - C_t^{I} \sum_{\mu,\nu} J_{t,\mu\nu}(\vec{r}) \, J_{t,\mu\nu}(\vec{r}) \Big] \,.$$

which has

- Densities: $\rho, \tau, J_{\mu\nu}$
- $\sim$ **10** parameters ...
- ... fitted to a few masses and radii.

Easy to calculate!

Only half of the story; also time-odd terms!

What we try to reach

Yale

Global microscopic framework to describe

What we try to reach

Yale

Global microscopic framework to describe

Bulk properties

- Masses
- Radii

What we try to reach

Yale

Global microscopic framework to describe

Bulk properties

- Masses
- Radii

Low-energy spectroscopy

- Ground states
- Low-lying excitations

What we try to reach

Yale

Global microscopic framework to describe

Bulk properties

- Masses
- Radii

Low-energy spectroscopy

- Ground states
- Low-lying excitations

Rather good: **HFB17** describes

S.Goriely et al., PRL **102**, 152503 (2009).

- **2149** masses, RMS = **0.581** MeV
  Cf. BE(^{208}Pb) $\sim$ **1.6 G**eV.
- **792** radii , RMS $\approx$ **0.03** fm
  Cf. $R_{\rm rms}$ (^{208}Pb) = $\sim$ **5.5** fm.

What we try to reach

Yale

Global microscopic framework to describe

Bulk properties

- Masses
- Radii

Rather good: **HFB17** describes

S.Goriely et al., PRL **102**, 152503 (2009).

- **2149** masses, RMS = **0.581** MeV
  Cf. BE(^{208}Pb) $\sim$ **1.6 G**eV.
- **792** radii , RMS $\approx$ **0.03** fm
  Cf. $R_{\rm rms}$ (^{208}Pb) = $\sim$ **5.5** fm.

Low-energy spectroscopy

- Ground states
- Low-lying excitations

This is not easy at all . . .

L. Bonneau et al., PRC **76**, 024320.

- Correct J^π $\sim$ 80% (spherical)
- Correct J^π $\sim$ 40% (deformed)
- Performance $\sim$ MIC-MAC.

What we try to reach

Yale

Global microscopic framework to describe

Bulk properties

- Masses
- Radii

Rather good: **HFB17** describes
S.Goriely et al., PRL **102**, 152503 (2009).

- **2149** masses, RMS = **0.581** MeV
  Cf. BE(^{208}Pb) $\sim$ **1.6 G**eV.
- **792** radii , RMS $\approx$ **0.03** fm
  Cf. $R_{\rm rms}$ (^{208}Pb) = $\sim$ **5.5** fm.

Low-energy spectroscopy

- Ground states
- Low-lying excitations

This is not easy at all . . .
L. Bonneau et al., PRC **76**, 024320.

- Correct J^π $\sim$ 80% (spherical)
- Correct J^π $\sim$ 40% (deformed)
- Performance $\sim$ MIC-MAC.

UNEDF-SCIDAC, PRC 89 054314 (2014)

(...) more data points or playing with the χ^2 is not going to change the situation. (...) UNEDF2 marks the end of the (NLO) Skyrme EDF strategy.

NℓLO EDFs

The most general EDF with terms (and densities) with up to 2ℓ gradients

NℓLO EDFs

The most general EDF with terms (and densities) with up to 2ℓ gradients

Examples:

LO	$\rho\rho$, $\vec{s}\cdot\vec{s}$,
NLO	$\rho\Delta\rho, \tau\tau$, $\vec{j}\cdot\vec{j}, \ldots$
N2LO	$\Delta\rho\Delta\rho$, $\tau\nabla\nabla\rho$, $\rho Q, \ldots$
$\ldots$	

$$E_{\rm Sk}^{\rm N2LO} = \int d\vec{r}\left[\mathcal{E}^{(0)}(\vec{r}) + \mathcal{E}^{(2)}(\vec{r}) + \mathcal{E}^{(4)}(\vec{r})\right],$$

$$\mathcal{E}^{(i)}(\vec{r}) = \sum_{t=0,1} \mathcal{E}^{(i)}_{\rm t,e}(\vec{r}) + \mathcal{E}^{(i)}_{\rm t,o}(\vec{r})\,.$$

NℓLO EDFs

The most general EDF with terms (and densities) with up to 2ℓ gradients

- Up to some symmetry restrictions
- An expansion in some parameter? NO
- A systematic enumeration of terms? YES

Examples:

LO	$\rho\rho$, $\vec{s}\cdot\vec{s}$,
NLO	$\rho\Delta\rho, \tau\tau$, $\vec{j}\cdot\vec{j}, \ldots$
N2LO	$\Delta\rho\Delta\rho$, $\tau\nabla\nabla\rho$, $\rho Q, \ldots$
$\ldots$	

$$E_{\rm Sk}^{\rm N2LO} = \int d\vec{r}\left[\mathcal{E}^{(0)}(\vec{r}) + \mathcal{E}^{(2)}(\vec{r}) + \mathcal{E}^{(4)}(\vec{r})\right],$$

$$\mathcal{E}^{(i)}(\vec{r}) = \sum_{t=0,1} \mathcal{E}^{(i)}_{\rm t,e}(\vec{r}) + \mathcal{E}^{(i)}_{\rm t,o}(\vec{r})\,.$$

Yale

NℓLO EDFs

The most general EDF with terms (and densities) with up to 2ℓ gradients

- Up to some symmetry restrictions
- An expansion in some parameter? NO
- A systematic enumeration of terms? YES

Examples:

LO	$\rho\rho$, $\vec{s}\cdot\vec{s}$,
NLO	$\rho\Delta\rho, \tau\tau$, $\vec{j}\cdot\vec{j}, \ldots$
N2LO	$\Delta\rho\Delta\rho$, $\tau\nabla\nabla\rho, \rho Q, \ldots$
...	

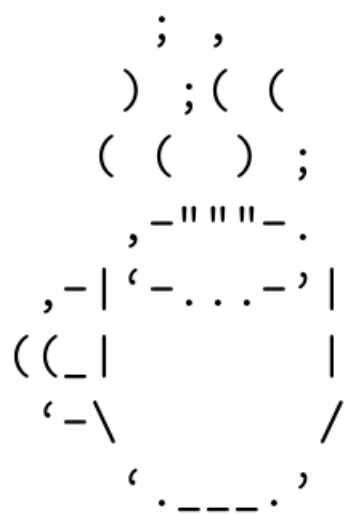
$$E_{\rm Sk}^{\rm N2LO} = \int d\vec{r}\left[\mathcal{E}^{(0)}(\vec{r}) + \mathcal{E}^{(2)}(\vec{r}) + \mathcal{E}^{(4)}(\vec{r})\right],$$

$$\mathcal{E}^{(i)}(\vec{r}) = \sum_{t=0,1} \mathcal{E}^{(i)}_{\rm t,e}(\vec{r}) + \mathcal{E}^{(i)}_{\rm t,o}(\vec{r})\,.$$

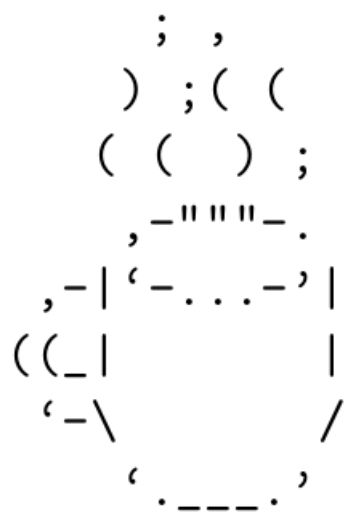
Some history

1. Enumeration up to $\ell = 3$ — B. G. Carlsson et al. PRC **78**, 044326 (2008) (Jyväskylä).
2. Constraints for pseudo-potential — F. Raimondi et al., PRC **83**, 1-15 (2011) (Jyväskylä).
3. Infinite matter studies for $\ell = 2, 3$ — D. Davesne et al. PRC **91**, 064303 (2015) (Lyon).
4. First fit to finite nuclei, SN2LO1 — P. Becker et al. PRC **96**, 044330 (2017) (Lyon).

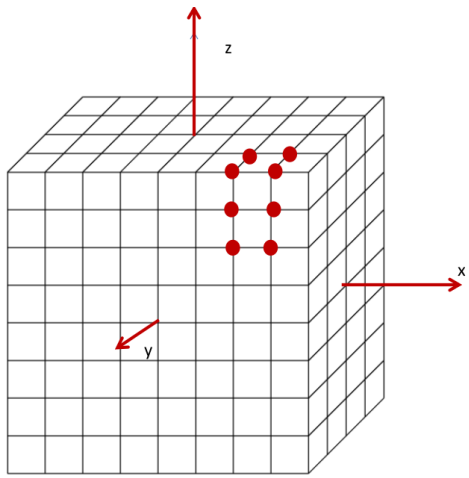
Contents



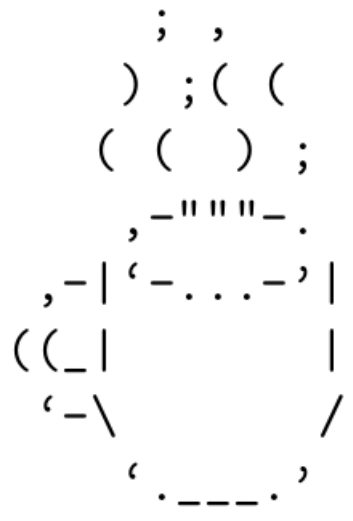
- Mean-field with Skyrme EDFs
  with HF, HF+BCS or HFB pairing.
- 3D coordinate mesh
  Typical sp-basis size: 32000 $\sim$ 64000(!)



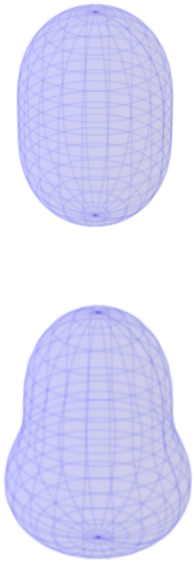
- Mean-field with Skyrme EDFs
  with HF, HF+BCS or HFB pairing.
- 3D coordinate mesh
  Typical sp-basis size: 32000 $\sim$ 64000(!)

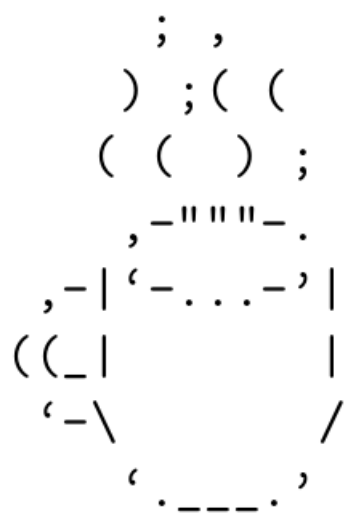


D. Baye et al. J. Phys. A 19 (1986).



- Mean-field with Skyrme EDFs
  with HF, HF+BCS or HFB pairing.
- 3D coordinate mesh
  Typical sp-basis size: 32000 $\sim$ 64000(!)
- 16 possible symmetry combinations
- Fast and easy to use
- Extremely accurate

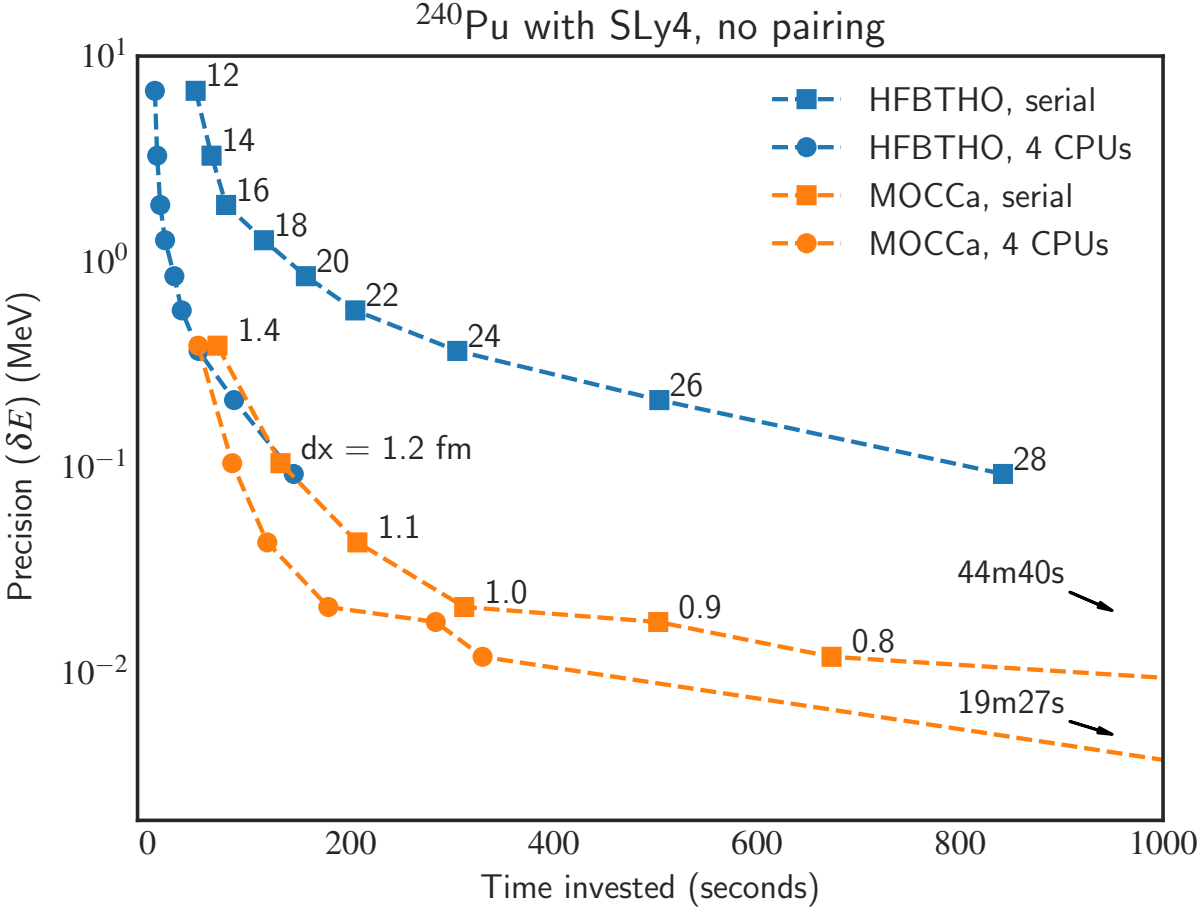




- Mean-field with Skyrme EDFs
  with HF, HF+BCS or HFB pairing.
- 3D coordinate mesh
  Typical sp-basis size: 32000 $\sim$ 64000(!)
- 16 possible symmetry combinations
- Fast and easy to use
- Extremely accurate
- Now up to N2LO

Why coordinate space?

Yale



MOCCa vs. HFBTHOv3

Axial code in HO space

R.N. Perez CPC **220**, 363-375 (2017).

EV8: WR, V. Hellemans, M. Bender and P.-H. Heenen, CPC **187**, 175 - 194 (2015).
Precision: WR, M. Bender and P.-H. Heenen, PRC **92**, 064318 (2015).
Speed: WR, M. Bender and P.-H. Heenen, Eur. Phys. J. A **55**, 93 (2019).

Finite difference derivatives

$$f''(x) \approx \frac{1}{dx^2}\Big[f(x+dx) - 2f(x) + f(x-dx)\Big]$$

Derivatives on a mesh

Finite difference derivatives

$$f''(x) \approx \frac{1}{dx^2}\Big[f(x+dx) - 2f(x) + f(x-dx)\Big]$$

$$f''(x_i) = \frac{1}{dx^2}\sum_j A_{ij} f(x_j)$$

$$A = \begin{pmatrix} \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & 1 & -2 & 1 & 0 & 0\dots \\ \dots & 0 & 1 & -2 & 1 & 0\dots \\ \dots & 0 & 0 & 1 & -2 & 1\dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$

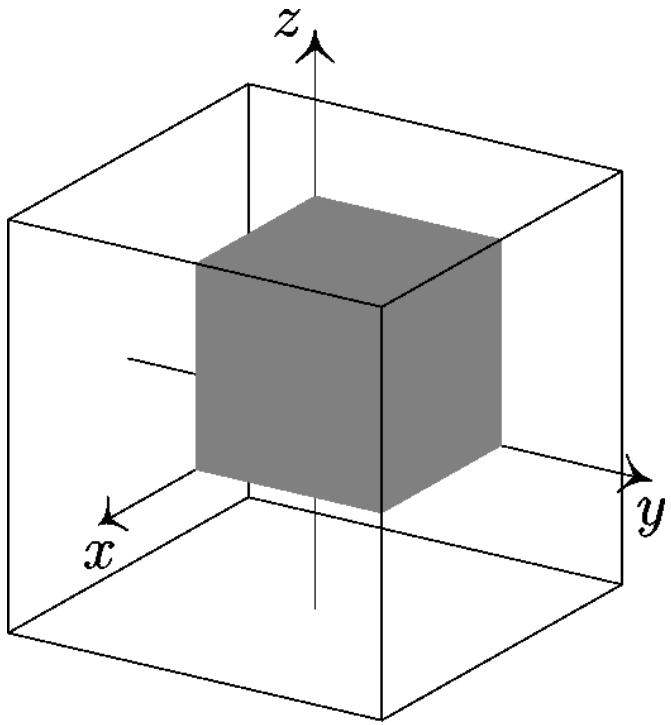
Derivatives on a mesh

Finite difference derivatives

$$f''(x) \approx \frac{1}{dx^2}\Big[f(x+dx) - 2f(x) + f(x-dx)\Big]$$

$$f''(x_i) = \frac{1}{dx^2}\sum_j A_{ij} f(x_j)$$

$$A = \begin{pmatrix} \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & 1 & -2 & 1 & 0 & 0\dots \\ \dots & 0 & 1 & -2 & 1 & 0\dots \\ \dots & 0 & 0 & 1 & -2 & 1\dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$



Symmetries **of** f determine the boundary conditions at $x/y/z = 0$.

Derivatives on a mesh

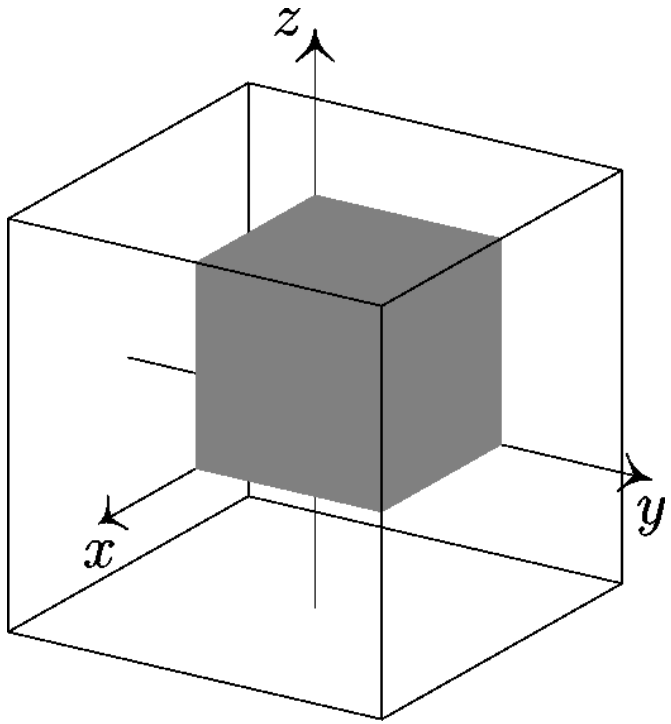
Yale

Finite difference derivatives

$$f''(x) \approx \frac{1}{dx^2}\Big[f(x+dx) - 2f(x) + f(x-dx)\Big]$$

$$f''(x_i) = \frac{1}{dx^2}\sum_j A_{ij} f(x_j)$$

$$A = \begin{pmatrix} \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & 1 & -2 & 1 & 0 & 0\dots \\ \dots & 0 & 1 & -2 & 1 & 0\dots \\ \dots & 0 & 0 & 1 & -2 & 1\dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$



Symmetries **of** f determine the boundary conditions at $x/y/z = 0$.

Actual implementation is pretty complicated

- A is a Lagrange derivative matrix (hence precision)
- 3D, so several boundary conditions $\rightarrow$ several A's.

Contents

Yale

Time-even, locally gauge invariant, N2LO energy density linked to a **central pseudo-potential**

$$
\begin{aligned}
\mathcal{E}^{(4)}_{e,t} = \quad & C_t^{(4)\Delta\rho}\Delta\rho_t(\vec{r})\Delta\rho_t(\vec{r}) + C_t^{(4)M\rho}\left[\rho_t(\vec{r})Q_t(\vec{r}) + \tau_t^2(\vec{r})\right] \\
& + C_t^{(4)M\rho}\sum_{\mu,\nu}\left[2\,\tau_{t,\mu\nu}(\vec{r})\tau_{t,\mu\nu}(\vec{r}) - 2\,\tau_{t,\mu\nu}(\vec{r})\left[\nabla_\mu\nabla_\nu\rho_t(\vec{r})\right]\right] \\
& - C_t^{(4)Ms}\sum_{\mu\nu\kappa}\left[2\,\mathrm{Im}K_{t,\mu\nu\kappa}(\vec{r})\mathrm{Im}K_{t,\mu\nu\kappa}(\vec{r}) - \left[\nabla_\mu J_{t,\mu\nu}(\vec{r})\right]\left[\nabla_\kappa J_{t,\kappa\nu}(\vec{r})\right]\right] \\
& - 4C_t^{(4)Ms}\sum_{\mu\nu} J_{t,\mu\nu}(\vec{r})V_{t,\mu\nu}(\vec{r})\,.
\end{aligned}
$$

Lots of terms with lots of densities

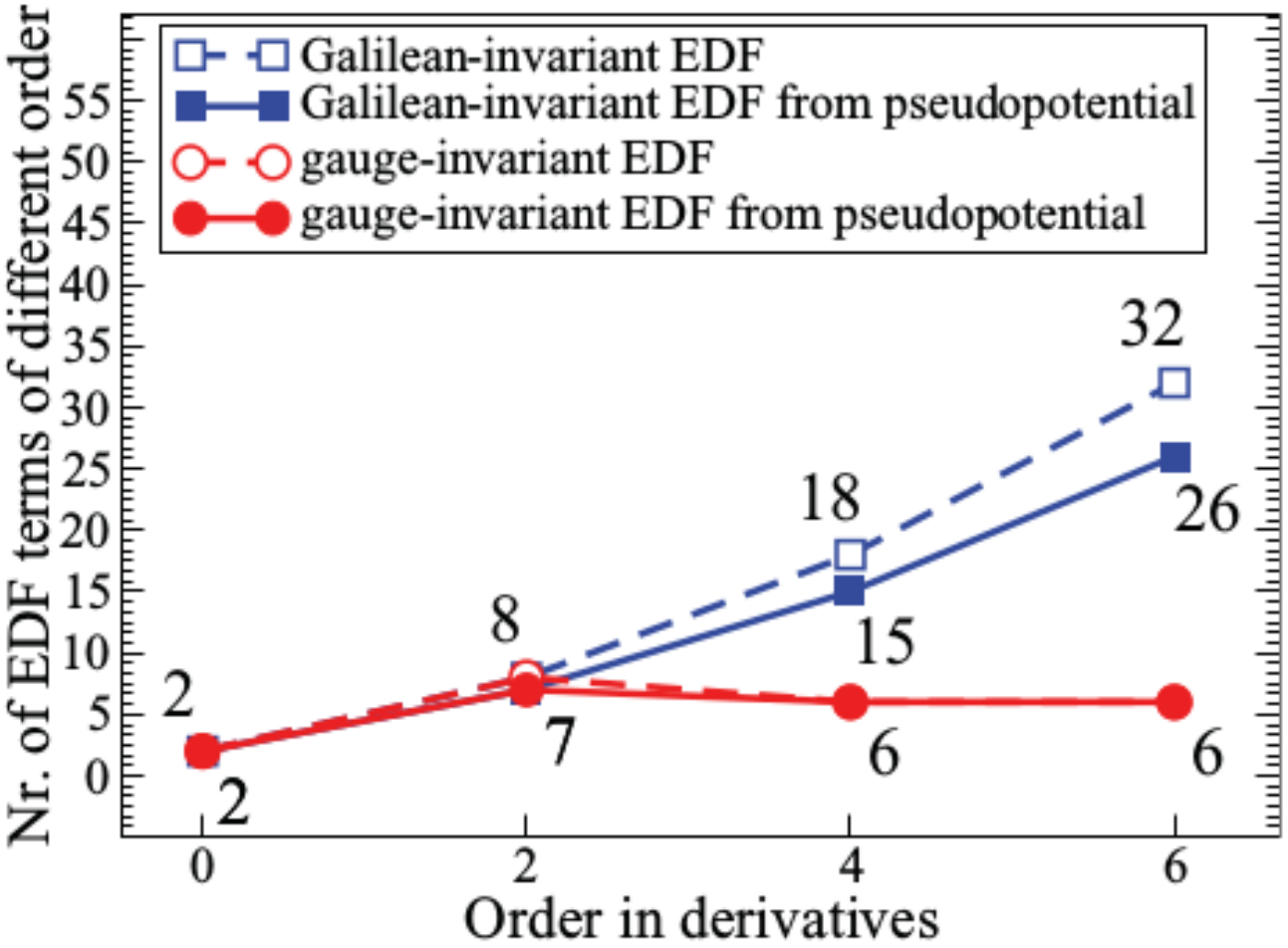


Fig. from F. Raimondi et al., PRC **83**, 054311 (2011).

Implementing this is hard

Yale

Problem: NℓLO functionals are wickedly complicated (even in 3D)

- Calculation of densities (11 of them)
- Correct calculation of derivatives of densities
- Correct calculation of (the action of) single-particle hamiltonian

```
!                                        Symmetries for the derivatives
!                                                        |
DenIn%ImDTN2LO(:,:,:,1,1,it) =                           v              &
&           DeriveX(DenIn%ImKN2LO(:,:,:,1,1,1,it), P, -S, T,2)
DenIn%ImDTN2LO(:,:,:,1,1,it) = DenIn%ImDTN2LO(:,:,:,1,1,it) +          &
&           DeriveY(DenIn%ImKN2LO(:,:,:,2,1,1,it), P, -S, T,1)
DenIn%ImDTN2LO(:,:,:,1,1,it) = DenIn%ImDTN2LO(:,:,:,1,1,it) +          &
&           DeriveZ(DenIn%ImKN2LO(:,:,:,3,1,1,it), P, +S, T,2)
[.............................]
[8 more blocks like this with slightly different indices]
```

Implementing this is hard

Yale

Problem: NℓLO functionals are wickedly complicated (even in 3D)

- Calculation of densities (11 of them)
- Correct calculation of derivatives of densities
- Correct calculation of (the action of) single-particle hamiltonian

Bigger problems:

- Formulation of the functional was not finalized, even at N2LO.
- Densities can be chosen many ways
- Implementation needs to work for symmetry choices

```
!                                        Symmetries for the derivatives
!                                                          |
DenIn%ImDTN2LO(:,:,:,1,1,it) =                             v              &
&           DeriveX(DenIn%ImKN2LO(:,:,:,1,1,1,it), P, -S, T,2)
DenIn%ImDTN2LO(:,:,:,1,1,it) = DenIn%ImDTN2LO(:,:,:,1,1,it) +          &
&           DeriveY(DenIn%ImKN2LO(:,:,:,2,1,1,it), P, -S, T,1)
DenIn%ImDTN2LO(:,:,:,1,1,it) = DenIn%ImDTN2LO(:,:,:,1,1,it) +          &
&           DeriveZ(DenIn%ImKN2LO(:,:,:,3,1,1,it), P, +S, T,2)
[.............................]
[8 more blocks like this with slightly different indices]
```

Implementing this is hard

Yale

Problem: NℓLO functionals are wickedly complicated (even in 3D)

- Calculation of densities (11 of them)
- Correct calculation of derivatives of densities
- Correct calculation of (the action of) single-particle hamiltonian

Bigger problems:

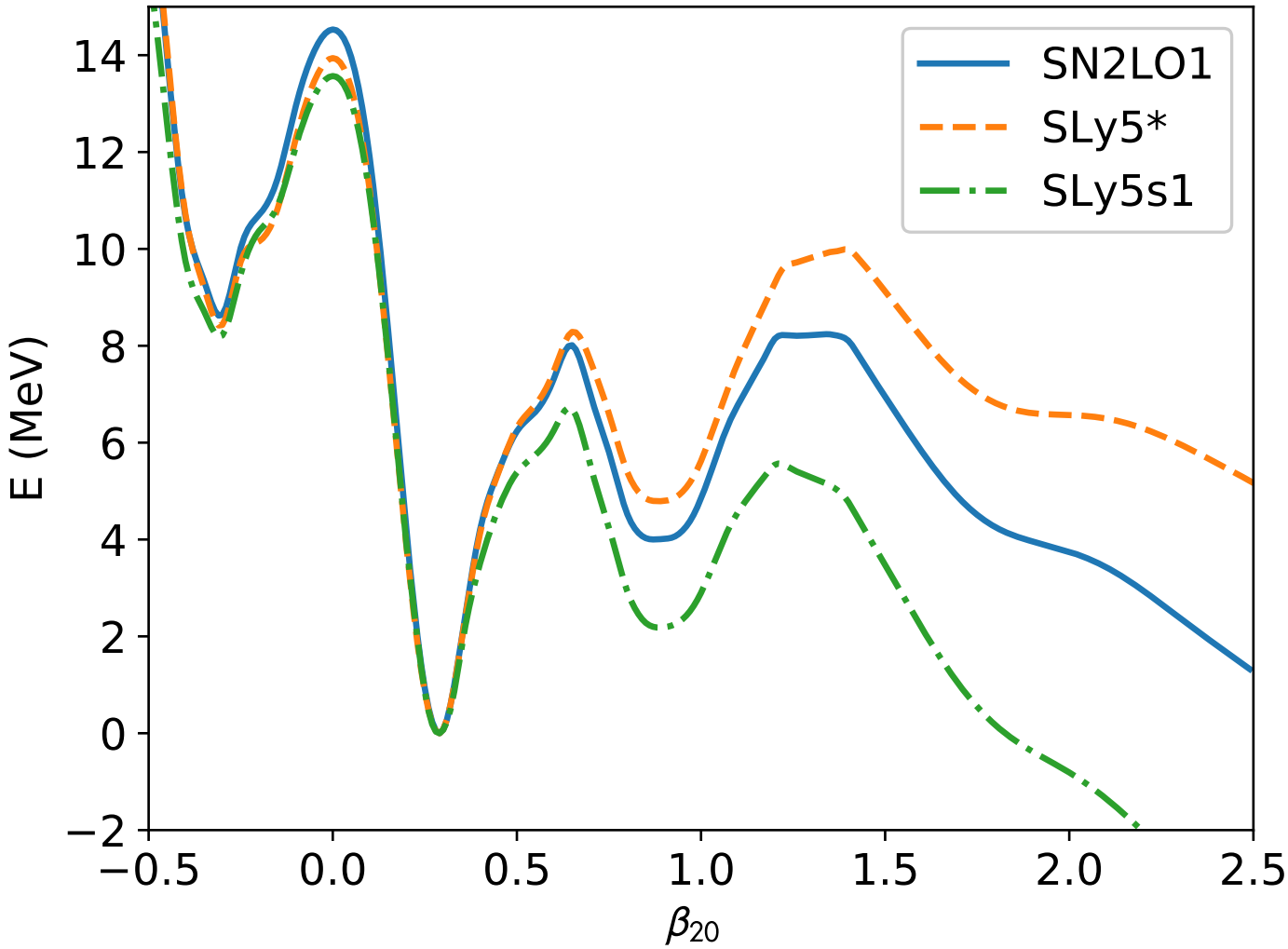
- Formulation of the functional was not finalized, even at N2LO.
- Densities can be chosen many ways
- Implementation needs to work for symmetry choices

Biggest problem: people kept asking for many slightly different EDFs!

```
!                                        Symmetries for the derivatives
!                                                          |
DenIn%ImDTN2LO(:,:,:,1,1,it) =                             v              &
&           DeriveX(DenIn%ImKN2LO(:,:,:,1,1,1,it), P, -S, T,2)
DenIn%ImDTN2LO(:,:,:,1,1,it) = DenIn%ImDTN2LO(:,:,:,1,1,it) +          &
&           DeriveY(DenIn%ImKN2LO(:,:,:,2,1,1,it), P, -S, T,1)
DenIn%ImDTN2LO(:,:,:,1,1,it) = DenIn%ImDTN2LO(:,:,:,1,1,it) +          &
&           DeriveZ(DenIn%ImKN2LO(:,:,:,3,1,1,it), P, +S, T,2)
[.............................]
[8 more blocks like this with slightly different indices]
```

Example: Pu fission barrier

Yale



W. Ryssens & M. Bender, in preparation

Contents

Two codes

Yale

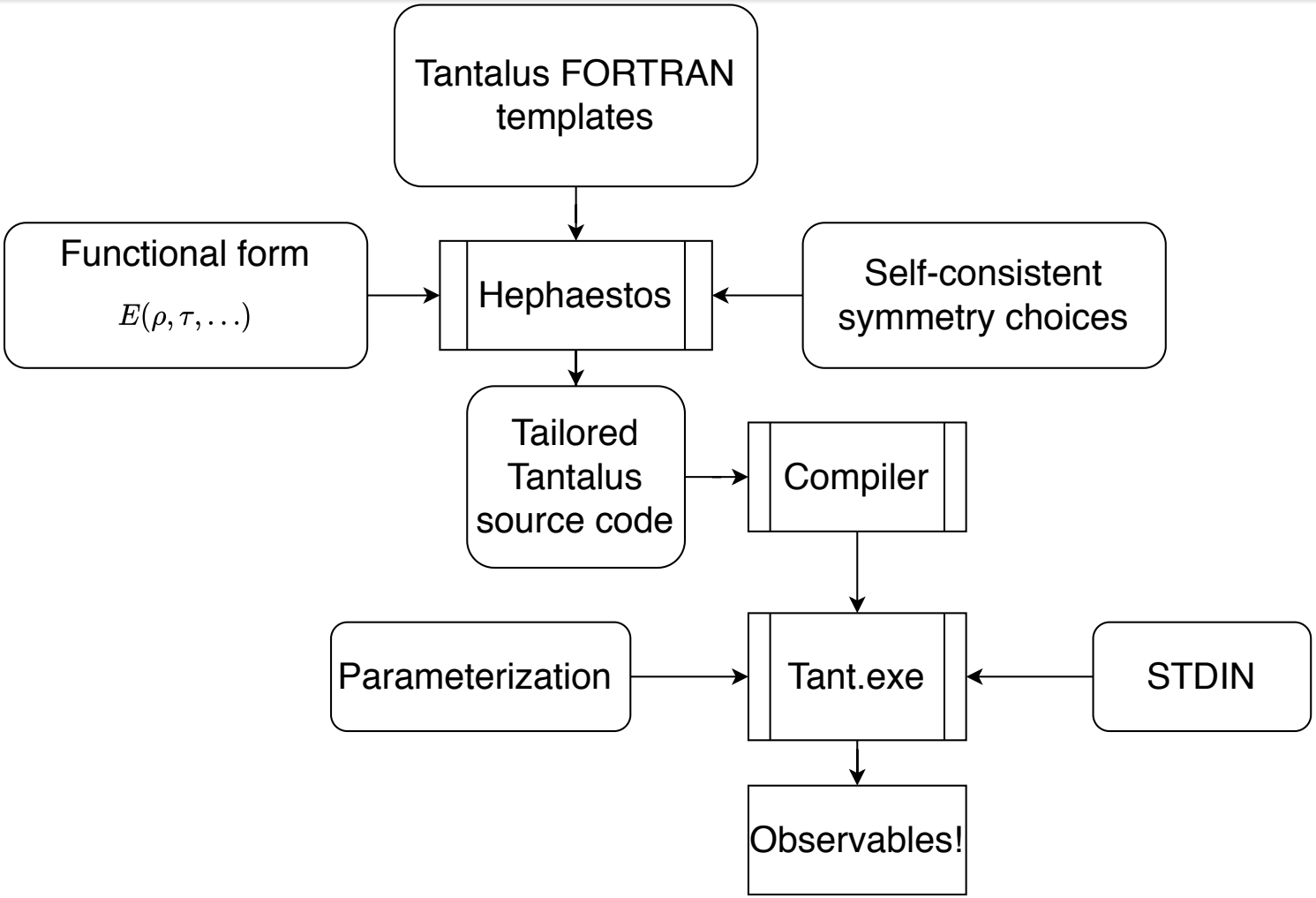
Hephaestos tailors Tantalus to the users specification.

- functional form
- symmetries requested

Yale

Hephaestos tailors Tantalus to the users specification.

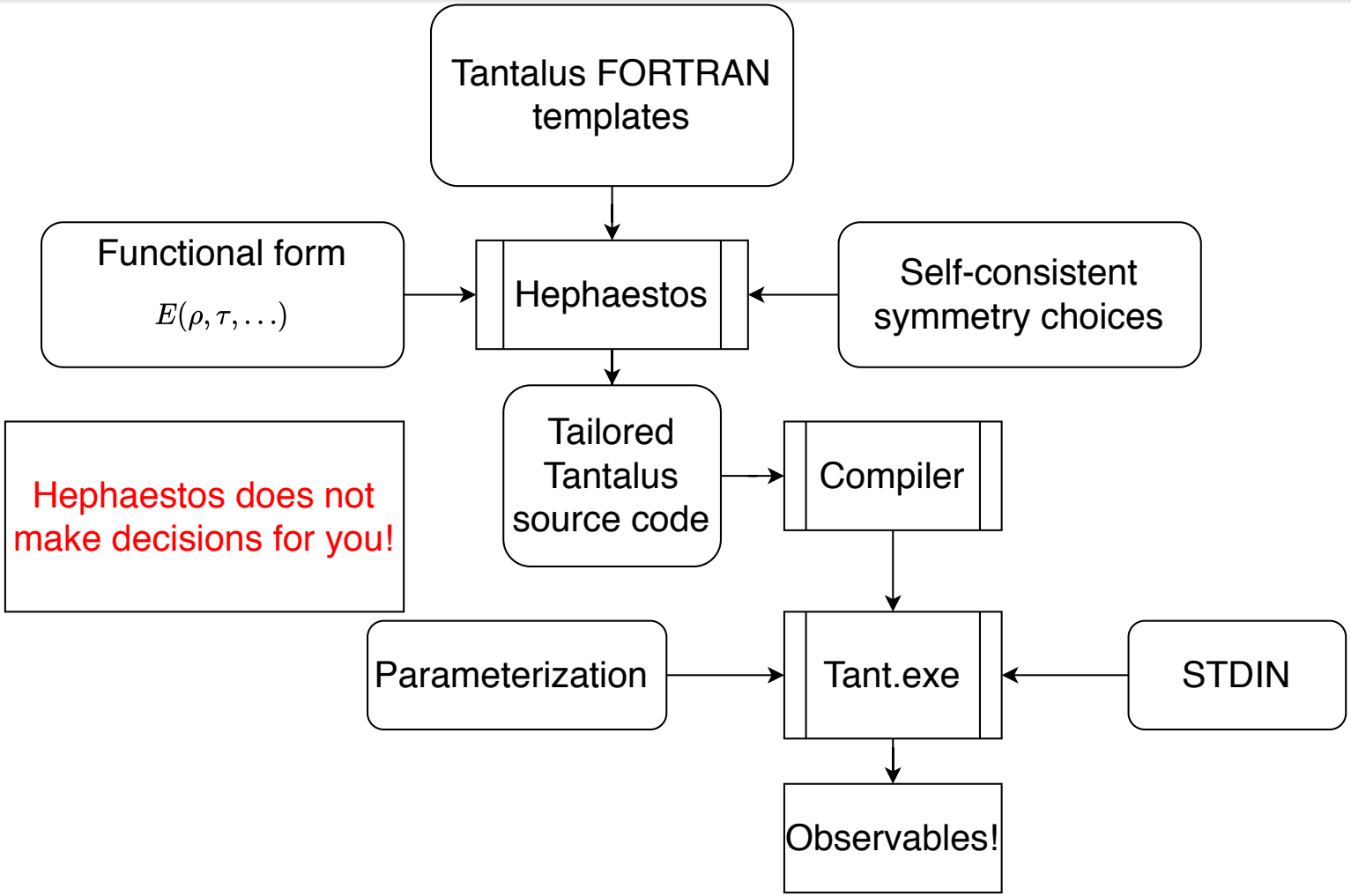
- functional form
- symmetries requested



Yale

Hephaestos tailors Tantalus to the users specification.

- functional form
- symmetries requested



Contents

Yale

The goal

Solve the self-consistent HFB equations

Yale

The goal

Minimize $E(\rho, \tau, \ldots)$
subject to densities being
generated by a HFB state

Yale

The goal

Solve a large-dimensional **constrained optimization problem**

Yale

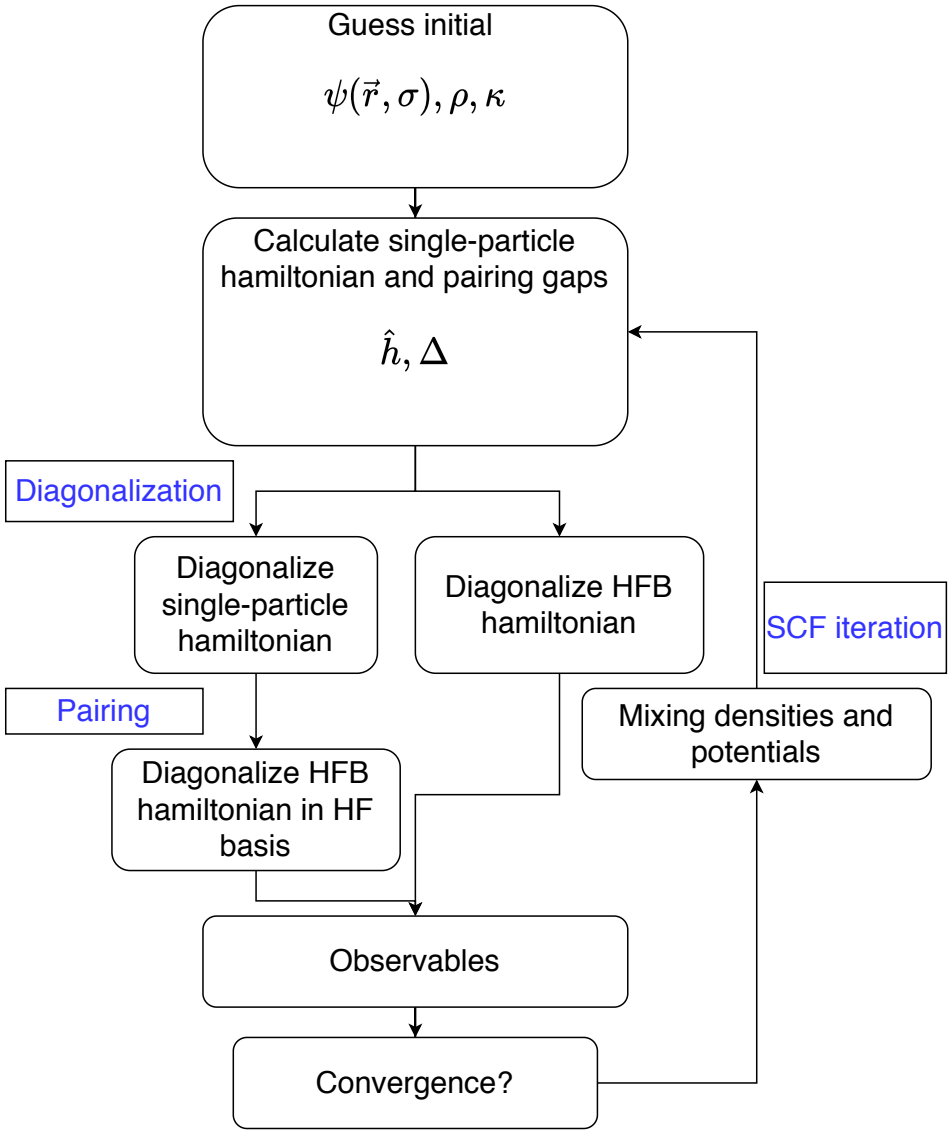
The goal

Solve a large-dimensional **constrained optimization problem**

Three subproblems

- Diagonalization (linear,"predictable")
- Pairing
- SCF iteration (nonlinear,"unpredictable")

with strategies for each.



The structure of a HFB code

Yale

The goal

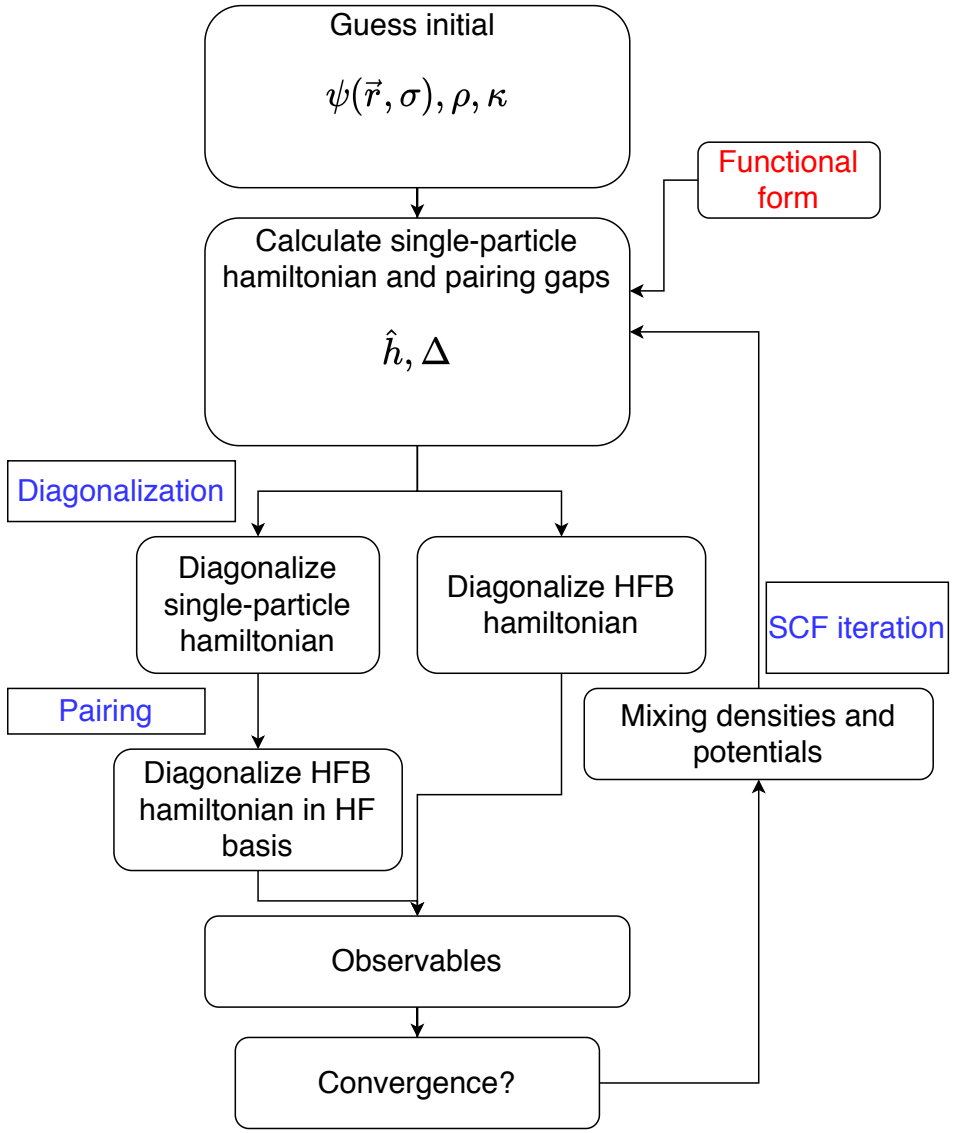
Solve a large-dimensional **constrained optimization problem**

Three subproblems

- Diagonalization (linear,"predictable")
- Pairing
- SCF iteration (nonlinear,"unpredictable")

with strategies for each.

Numerics is (almost) independent of the functional



WR, M. Bender and P.-H. Heenen, Eur. Phys. J. A **55**, 93 (2019).

So what do we need?

Yale

Tantalus

- FORTRAN for speed
- Numerical algorithms for
  - → Diagonalization subproblem
  - → Pairing subproblem
  - → SCF iteration
  - → Dealing with constraints
- Loads of routines for
  - → Calculating expectation values
  - → Input and ouput
  - → ...

So what do we need?

Yale

Tantalus

- FORTRAN for speed
- Numerical algorithms for
  - → Diagonalization subproblem
  - → Pairing subproblem
  - → SCF iteration
  - → Dealing with constraints
- Loads of routines for
  - → Calculating expectation values
  - → Input and ouput
  - → ...

Hephaestos

- Python for string manipulation
- Detect which densities are present in a functional form
- Determine how to calculate them and their derivatives
- Determine $\hat{h}$ and Δ
  a) Determine the mean-field potentials
  b) Determine how to actually calculate matrix elements

So what do we need?

Yale

Tantalus

- FORTRAN for speed
- Numerical algorithms for
  - → Diagonalization subproblem
  - → Pairing subproblem
  - → SCF iteration
  - → Dealing with constraints
- Loads of routines for
  - → Calculating expectation values
  - → Input and ouput
  - → ...

Hephaestos

- Python for string manipulation
- Detect which densities are present in a functional form
- Determine how to calculate them and their derivatives
- Determine $\hat{h}$ and Δ
  - a) Determine the mean-field potentials
  - b) Determine how to actually calculate matrix elements
- Write all of this in appropriate FORTRAN code
- ... and keeping track of symmetry of objects
- Optional: write **efficient** code.

Contents

Yale

Non-local densities

$$\rho(\vec{r}\sigma,\vec{r}'\sigma') = \langle \hat{a}^\dagger_{\vec{r}'\sigma'}\hat{a}_{\vec{r}\sigma}\rangle = \tfrac{1}{2}\,\rho(\vec{r},\vec{r}')\delta_{\sigma\sigma'} + \tfrac{1}{2}\,\langle\sigma'|\hat{\boldsymbol{\sigma}}|\sigma\rangle\cdot\vec{s}(\vec{r},\vec{r}')\,,$$

$$\tilde{\rho}(\vec{r}\sigma,\vec{r}'\sigma') = \sigma'\langle \hat{a}_{\vec{r}'\sigma'}\hat{a}_{\vec{r}\sigma}\rangle = \tfrac{1}{2}\,\tilde{\rho}(\vec{r},\vec{r}')\delta_{\sigma\sigma'} + \tfrac{1}{2}\,\langle\sigma'|\hat{\boldsymbol{\sigma}}|\sigma\rangle\cdot\tilde{\vec{s}}(\vec{r},\vec{r}')\,.$$

Yale

Non-local densities

$$\rho(\vec{r}\sigma,\vec{r}'\sigma') = \langle \hat{a}^\dagger_{\vec{r}'\sigma'}\hat{a}_{\vec{r}\sigma}\rangle = \tfrac{1}{2}\,\rho(\vec{r},\vec{r}')\delta_{\sigma\sigma'} + \tfrac{1}{2}\,\langle\sigma'|\hat{\boldsymbol{\sigma}}|\sigma\rangle\cdot\vec{s}(\vec{r},\vec{r}')\,,$$

$$\tilde{\rho}(\vec{r}\sigma,\vec{r}'\sigma') = \sigma'\langle \hat{a}_{\vec{r}'\sigma'}\hat{a}_{\vec{r}\sigma}\rangle = \tfrac{1}{2}\,\tilde{\rho}(\vec{r},\vec{r}')\delta_{\sigma\sigma'} + \tfrac{1}{2}\,\langle\sigma'|\hat{\boldsymbol{\sigma}}|\sigma\rangle\cdot\tilde{\vec{s}}(\vec{r},\vec{r}')\,.$$

Constructing a local density:

1. Pick one of four non-local densities
2. Pick a set of derivatives acting on $\vec{r}$, and a set on $\vec{r}'$
3. Set $\vec{r} = \vec{r}'$

Local densities at N2LO

Yale

Non-local densities

$$\rho(\vec{r}\sigma,\vec{r}'\sigma') = \langle \hat{a}^\dagger_{\vec{r}'\sigma'}\hat{a}_{\vec{r}\sigma}\rangle = \tfrac{1}{2}\,\rho(\vec{r},\vec{r}')\delta_{\sigma\sigma'} + \tfrac{1}{2}\,\langle\sigma'|\hat{\boldsymbol{\sigma}}|\sigma\rangle\cdot\vec{s}(\vec{r},\vec{r}')\,,$$

$$\tilde{\rho}(\vec{r}\sigma,\vec{r}'\sigma') = \sigma'\langle \hat{a}_{\vec{r}'\sigma'}\hat{a}_{\vec{r}\sigma}\rangle = \tfrac{1}{2}\,\tilde{\rho}(\vec{r},\vec{r}')\delta_{\sigma\sigma'} + \tfrac{1}{2}\,\langle\sigma'|\hat{\boldsymbol{\sigma}}|\sigma\rangle\cdot\tilde{\vec{s}}(\vec{r},\vec{r}')\,.$$

Constructing a local density:

1. Pick one of four non-local densities
2. Pick a set of derivatives acting on $\vec{r}$, and a set on $\vec{r}'$
3. Set $\vec{r} = \vec{r}'$

Examples:

$$Q(\vec{r}) \equiv \Delta\Delta'\rho(\vec{r},\vec{r}')\big|_{\vec{r}=\vec{r}'}$$

$$\tau_{\mu\nu}(\vec{r}) \equiv \nabla'_\mu\nabla_\nu\rho(\vec{r},\vec{r}')\big|_{\vec{r}=\vec{r}'}\quad .$$

Local densities at N2LO

Non-local densities

$$\rho(\vec{r}\sigma,\vec{r}'\sigma') = \langle \hat{a}^\dagger_{\vec{r}'\sigma'}\hat{a}_{\vec{r}\sigma}\rangle = \tfrac{1}{2}\,\rho(\vec{r},\vec{r}')\delta_{\sigma\sigma'} + \tfrac{1}{2}\,\langle\sigma'|\hat{\boldsymbol{\sigma}}|\sigma\rangle\cdot\vec{s}(\vec{r},\vec{r}')\,,$$
$$\tilde{\rho}(\vec{r}\sigma,\vec{r}'\sigma') = \sigma'\langle \hat{a}_{\vec{r}'\sigma'}\hat{a}_{\vec{r}\sigma}\rangle = \tfrac{1}{2}\,\tilde{\rho}(\vec{r},\vec{r}')\delta_{\sigma\sigma'} + \tfrac{1}{2}\,\langle\sigma'|\hat{\boldsymbol{\sigma}}|\sigma\rangle\cdot\tilde{\vec{s}}(\vec{r},\vec{r}')\,.$$

Constructing a local density:

1. Pick one of four non-local densities
2. Pick a set of derivatives acting on $\vec{r}$, and a set on $\vec{r}'$
3. Set $\vec{r} = \vec{r}'$

Examples:

$$Q(\vec{r}) \equiv \Delta\Delta'\rho(\vec{r},\vec{r}')\big|_{\vec{r}=\vec{r}'}$$
$$\tau_{\mu\nu}(\vec{r}) \equiv \nabla'_\mu\nabla_\nu\rho(\vec{r},\vec{r}')\big|_{\vec{r}=\vec{r}'}\quad.$$

Does not guarantee reality!

$$\tau^*_{\mu\nu}(\vec{r}) \neq \pm\tau_{\mu\nu}(\vec{r})$$

Local densities at N2LO

Yale

Non-local densities

$$\rho(\vec{r}\sigma,\vec{r}'\sigma') = \langle \hat{a}^\dagger_{\vec{r}'\sigma'}\hat{a}_{\vec{r}\sigma}\rangle = \tfrac{1}{2}\,\rho(\vec{r},\vec{r}')\delta_{\sigma\sigma'} + \tfrac{1}{2}\,\langle\sigma'|\hat{\boldsymbol{\sigma}}|\sigma\rangle\cdot\vec{s}(\vec{r},\vec{r}')\,,$$
$$\tilde{\rho}(\vec{r}\sigma,\vec{r}'\sigma') = \sigma'\langle \hat{a}_{\vec{r}'\sigma'}\hat{a}_{\vec{r}\sigma}\rangle = \tfrac{1}{2}\,\tilde{\rho}(\vec{r},\vec{r}')\delta_{\sigma\sigma'} + \tfrac{1}{2}\,\langle\sigma'|\hat{\boldsymbol{\sigma}}|\sigma\rangle\cdot\tilde{\vec{s}}(\vec{r},\vec{r}')\,.$$

Constructing a local density:

1. Pick one of four non-local densities
2. Pick a set of derivatives acting on $\vec{r}$, and a set on $\vec{r}'$
3. Set $\vec{r} = \vec{r}'$

Examples:

$$Q(\vec{r}) \equiv \Delta\Delta'\rho(\vec{r},\vec{r}')\big|_{\vec{r}=\vec{r}'}$$
$$\tau_{\mu\nu}(\vec{r}) \equiv \nabla'_\mu\nabla_\nu\rho(\vec{r},\vec{r}')\big|_{\vec{r}=\vec{r}'}\quad.$$

Does not guarantee reality!

$$\tau^*_{\mu\nu}(\vec{r}) \neq \pm\tau_{\mu\nu}(\vec{r})$$

The densities of SN2LO1 represent 104 real functions on the mesh

$$\rho, s_\mu, \tau_{\mu\nu}, j_\mu, J_{\mu\nu}, S_\mu, \Pi_\mu, V_{\mu\nu}, K_{\mu\nu\kappa}, Q$$

Does not include: NLO and N2LO tensor terms, pairing terms, N3LO terms, ...

Things to think about

- Not all possible densities offer meaningful degrees of freedom
  Example:

$$\mathrm{Im}\tau_{\mu\nu}(\vec{r}) = \frac{1}{2}\left[\nabla_\mu \vec{j}_\nu(\vec{r}) - \nabla_\nu \vec{j}_\mu(\vec{r})\right].$$

Things to think about

Yale

- Not all possible densities offer meaningful degrees of freedom
  Example:

$$\mathrm{Im}\tau_{\mu\nu}(\vec{r}) = \frac{1}{2}\left[\nabla_\mu \vec{j}_\nu(\vec{r}) - \nabla_\nu \vec{j}_\mu(\vec{r})\right].$$

We would like no reducible densities, like $\mathrm{Im}\tau_{\mu\nu}(\vec{r})$.

Things to think about

Yale

- Not all possible densities offer meaningful degrees of freedom
  Example:

$$\mathrm{Im}\tau_{\mu\nu}(\vec{r}) = \frac{1}{2}\left[\nabla_\mu \vec{j}_\nu(\vec{r}) - \nabla_\nu \vec{j}_\mu(\vec{r})\right].$$

  We would like no reducible densities, like $\mathrm{Im}\tau_{\mu\nu}(\vec{r})$.
- Many possibilities of densities are related

$$\mathcal{T}(\vec{r}) = (\nabla\cdot\nabla')(\nabla\cdot\nabla')\rho(\vec{r},\vec{r}')\big|_{\vec{r}=\vec{r}'} \qquad = Q(\vec{r}) + \text{derivatives of} \tau_{\mu\nu}$$

Things to think about

Yale

- Not all possible densities offer meaningful degrees of freedom
  Example:

$$\mathrm{Im}\tau_{\mu\nu}(\vec{r}) = \frac{1}{2}\left[\nabla_\mu \vec{j}_\nu(\vec{r}) - \nabla_\nu \vec{j}_\mu(\vec{r})\right].$$

  We would like no reducible densities, like $\mathrm{Im}\tau_{\mu\nu}(\vec{r})$.
- Many possibilities of densities are related

$$\mathcal{T}(\vec{r}) = (\nabla\cdot\nabla')(\nabla\cdot\nabla')\rho(\vec{r},\vec{r}')\big|_{\vec{r}=\vec{r}'} \quad = Q(\vec{r}) + \text{derivatives of} \tau_{\mu\nu}$$

  We would like to make non-redundant choices, like $\mathcal{T}(\vec{r})$ and $Q(\vec{r})$.

Things to think about

Yale

- Not all possible densities offer meaningful degrees of freedom
  Example:

$$\mathrm{Im}\tau_{\mu\nu}(\vec{r}) = \frac{1}{2}\left[\nabla_\mu \vec{j}_\nu(\vec{r}) - \nabla_\nu \vec{j}_\mu(\vec{r})\right].$$

  We would like no reducible densities, like $\mathrm{Im}\tau_{\mu\nu}(\vec{r})$.
- Many possibilities of densities are related

$$\mathcal{T}(\vec{r}) = (\nabla\cdot\nabla')(\nabla\cdot\nabla')\rho(\vec{r},\vec{r}')\big|_{\vec{r}=\vec{r}'} \quad = \mathcal{Q}(\vec{r}) + \text{derivatives of} \tau_{\mu\nu}$$

  We would like to make non-redundant choices, like $\mathcal{T}(\vec{r})$ and $\mathcal{Q}(\vec{r})$.
- Not all possibilities are nicely behaved under time-reversal

$$\tau^T_{\mu\nu}(\vec{r}) = \tau^*_{\mu\nu}(\vec{r}).$$

  We would like to keep "time-odd" and "time-even" densities.

Things to think about

Yale

- Not all possible densities offer meaningful degrees of freedom
  Example:

$$\mathrm{Im}\tau_{\mu\nu}(\vec{r}) = \frac{1}{2}\left[\nabla_\mu \vec{j}_\nu(\vec{r}) - \nabla_\nu \vec{j}_\mu(\vec{r})\right].$$

  We would like no reducible densities, like $\mathrm{Im}\tau_{\mu\nu}(\vec{r})$.
- Many possibilities of densities are related

$$\mathcal{T}(\vec{r}) = (\nabla\cdot\nabla')(\nabla\cdot\nabla')\rho(\vec{r},\vec{r}')\big|_{\vec{r}=\vec{r}'} \quad = \mathcal{Q}(\vec{r}) + \text{derivatives of} \tau_{\mu\nu}$$

  We would like to make non-redundant choices, like $\mathcal{T}(\vec{r})$ and $\mathcal{Q}(\vec{r})$.
- Not all possibilities are nicely behaved under time-reversal

$$\tau^T_{\mu\nu}(\vec{r}) = \tau^*_{\mu\nu}(\vec{r}).$$

  We would like to keep "time-odd" and "time-even" densities.
- The computational cost depends on the densities!

A systematic notation

Yale

New notation

$$D^{\hat{L},\hat{R}}(\vec{r}) = \mathbf{Re}\left\{\hat{L}\hat{R}\,\rho(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \quad , \quad C^{\hat{L},\hat{R}}(\vec{r}) = \mathbf{Im}\left\{\hat{L}\hat{R}\,\rho(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \quad .$$

A systematic notation

Yale

New notation

$$D^{\hat{L},\hat{R}}(\vec{r}) = \mathrm{Re}\left\{\hat{L}\hat{R}\,\rho(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \quad , \quad C^{\hat{L},\hat{R}}(\vec{r}) = \mathrm{Im}\left\{\hat{L}\hat{R}\,\rho(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \quad .$$

$$D^{\hat{L},\hat{R}\sigma}(\vec{r}) = \mathrm{Re}\left\{\hat{L}\hat{R}\,\vec{s}(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \quad , \quad C^{\hat{L},\hat{R}\sigma}(\vec{r}) = \mathrm{Im}\left\{\hat{L}\hat{R}\,\vec{s}(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \quad .$$

A systematic notation

Yale

New notation

$$D^{\hat{L},\hat{R}}(\vec{r}) = \mathrm{Re}\left\{\hat{L}\hat{R}\,\rho(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \quad , \quad C^{\hat{L},\hat{R}}(\vec{r}) = \mathrm{Im}\left\{\hat{L}\hat{R}\,\rho(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \; .$$

$$D^{\hat{L},\hat{R}\sigma}(\vec{r}) = \mathrm{Re}\left\{\hat{L}\hat{R}\,\vec{s}(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \, , \quad C^{\hat{L},\hat{R}\sigma}(\vec{r}) = \mathrm{Im}\left\{\hat{L}\hat{R}\,\vec{s}(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \; .$$

Examples:

$$C_{\mu}^{1,\nabla}(\vec{r}) = \frac{1}{2i}\left(\nabla_{\mu} - \nabla'_{\mu}\right)\rho(\vec{r},\vec{r}')\Big|_{\vec{r}=\vec{r}'} = \vec{j}(\vec{r}).$$

$$D_{q,\mu\nu\kappa}^{\nabla,\nabla\sigma}(\vec{r}) = \left(\nabla'_{\mu}\nabla_{\nu} + \nabla_{\mu}\nabla'_{\nu}\right) s_{q,\kappa}(\vec{r},\vec{r}')\Big|_{\vec{r}=\vec{r}'} = \mathrm{Re}K_{\nu\mu\kappa}(\vec{r}) \, .$$

A systematic notation

Yale

New notation

$$D^{\hat{L},\hat{R}}(\vec{r}) = \mathrm{Re}\left\{\hat{L}\hat{R}\,\rho(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \quad , \quad C^{\hat{L},\hat{R}}(\vec{r}) = \mathrm{Im}\left\{\hat{L}\hat{R}\,\rho(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \, .$$

$$D^{\hat{L},\hat{R}\sigma}(\vec{r}) = \mathrm{Re}\left\{\hat{L}\hat{R}\,\vec{s}(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \, , \quad C^{\hat{L},\hat{R}\sigma}(\vec{r}) = \mathrm{Im}\left\{\hat{L}\hat{R}\,\vec{s}(\vec{r},\vec{r}')\right\}\Big|_{\vec{r}=\vec{r}'} \, .$$

Examples:

$$C_{\mu}^{1,\nabla}(\vec{r}) = \frac{1}{2i}\left(\nabla_{\mu} - \nabla'_{\mu}\right)\rho(\vec{r},\vec{r}')\Big|_{\vec{r}=\vec{r}'} = \vec{j}(\vec{r}).$$

$$D_{q,\mu\nu\kappa}^{\nabla,\nabla\sigma}(\vec{r}) = \left(\nabla'_{\mu}\nabla_{\nu} + \nabla_{\mu}\nabla'_{\nu}\right) s_{q,\kappa}(\vec{r},\vec{r}')\Big|_{\vec{r}=\vec{r}'} = \mathrm{Re}K_{\nu\mu\kappa}(\vec{r}) \, .$$

Formal advantages

- **Extensibility**: up to NℓLO, with tensor terms, ...
- **Reality**: Every density is a real spatial function. (including pairing densities!)
- **Time-reversal**: every density is time-even/odd. (including pairing densities!)
- **Clarity**: operator structure in notation
  - → Some practical recoupling relations
  - → Redundancies and reducibility can be identified "on sight"

Contents

With the new notation, the (time-even) N2LO energy density becomes

$$\begin{aligned}\mathcal{E}^{(4)}_{\mathrm{Sk,e}}(\vec{r}) = \sum_{t=0,1}\Big[&\mathbb{A}^{(4,1)}_{\mathrm{t,e}}\left(\Delta D^{1,1}_t\right)\left(\Delta D^{1,1}_t\right) + \mathbb{A}^{(4,2)}_{\mathrm{t,e}} D^{1,1}_t\, D^{\Delta,\Delta}_t + \mathbb{A}^{(4,3)}_{\mathrm{t,e}} D^{(\nabla,\nabla)}_t D^{(\nabla,\nabla)}_t \\ &+ \mathbb{A}^{(4,4)}_{\mathrm{t,e}}\sum_{\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu} + \mathbb{A}^{(4,5)}_{\mathrm{t,e}}\sum_{\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu}\left(\nabla_\mu\nabla_\nu D^{1,1}_t\right) \\ &+ \mathbb{A}^{(4,6)}_{\mathrm{t,e}}\sum_{\mu\nu} C^{1,\nabla\sigma}_{t,\mu\nu}\left(\Delta C^{1,\nabla\sigma}_{t,\mu\nu}\right) + \mathbb{A}^{(4,7)}_{\mathrm{t,e}}\sum_{\mu\nu\kappa}\left(\nabla_\mu C^{1,\nabla\sigma}_{t,\mu\kappa}\right)\left(\nabla_\nu C^{1,\nabla\sigma}_{t,\nu\kappa}\right) \\ &+ \mathbb{A}^{(4,8)}_{\mathrm{t,e}}\sum_{\mu\nu} C^{1,\nabla\sigma}_{t,\mu\nu} C^{\Delta,\nabla\sigma}_{t,\mu\nu}\Big].\end{aligned}$$

With the new notation, the (time-even) N2LO energy density becomes

$$\begin{aligned}\mathcal{E}^{(4)}_{\mathrm{Sk,e}}(\vec{r}) = \sum_{t=0,1}\Big[&\mathbb{A}^{(4,1)}_{\mathrm{t,e}}\left(\Delta D^{1,1}_t\right)\left(\Delta D^{1,1}_t\right) + \mathbb{A}^{(4,2)}_{\mathrm{t,e}} D^{1,1}_t\, D^{\Delta,\Delta}_t + \mathbb{A}^{(4,3)}_{\mathrm{t,e}} D^{(\nabla,\nabla)}_t D^{(\nabla,\nabla)}_t \\ &+ \mathbb{A}^{(4,4)}_{\mathrm{t,e}}\sum_{\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu} + \mathbb{A}^{(4,5)}_{\mathrm{t,e}}\sum_{\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu}\left(\nabla_\mu\nabla_\nu D^{1,1}_t\right) \\ &+ \mathbb{A}^{(4,6)}_{\mathrm{t,e}}\sum_{\mu\nu} C^{1,\nabla\sigma}_{t,\mu\nu}\left(\Delta C^{1,\nabla\sigma}_{t,\mu\nu}\right) + \mathbb{A}^{(4,7)}_{\mathrm{t,e}}\sum_{\mu\nu\kappa}\left(\nabla_\mu C^{1,\nabla\sigma}_{t,\mu\kappa}\right)\left(\nabla_\nu C^{1,\nabla\sigma}_{t,\nu\kappa}\right) \\ &+ \mathbb{A}^{(4,8)}_{\mathrm{t,e}}\sum_{\mu\nu} C^{1,\nabla\sigma}_{t,\mu\nu} C^{\Delta,\nabla\sigma}_{t,\mu\nu}\Big].\end{aligned}$$

This looks intimidating . . .

With the new notation, the (time-even) N2LO energy density becomes

$$\begin{aligned}\mathcal{E}^{(4)}_{\mathrm{Sk,e}}(\vec{r}) = \sum_{t=0,1}\Big[&\mathbb{A}^{(4,1)}_{\mathrm{t,e}}\left(\Delta D^{1,1}_t\right)\left(\Delta D^{1,1}_t\right) + \mathbb{A}^{(4,2)}_{\mathrm{t,e}} D^{1,1}_t\, D^{\Delta,\Delta}_t + \mathbb{A}^{(4,3)}_{\mathrm{t,e}} D^{(\nabla,\nabla)}_t D^{(\nabla,\nabla)}_t \\ &+ \mathbb{A}^{(4,4)}_{\mathrm{t,e}}\sum_{\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu} + \mathbb{A}^{(4,5)}_{\mathrm{t,e}}\sum_{\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu}\left(\nabla_\mu\nabla_\nu D^{1,1}_t\right) \\ &+ \mathbb{A}^{(4,6)}_{\mathrm{t,e}}\sum_{\mu\nu} C^{1,\nabla\sigma}_{t,\mu\nu}\left(\Delta C^{1,\nabla\sigma}_{t,\mu\nu}\right) + \mathbb{A}^{(4,7)}_{\mathrm{t,e}}\sum_{\mu\nu\kappa}\left(\nabla_\mu C^{1,\nabla\sigma}_{t,\mu\kappa}\right)\left(\nabla_\nu C^{1,\nabla\sigma}_{t,\nu\kappa}\right) \\ &+ \mathbb{A}^{(4,8)}_{\mathrm{t,e}}\sum_{\mu\nu} C^{1,\nabla\sigma}_{t,\mu\nu} C^{\Delta,\nabla\sigma}_{t,\mu\nu}\Big].\end{aligned}$$

This looks intimidating . . . to you and me, but not to a computer!

With the new notation, the (time-even) N2LO energy density becomes

$$\begin{aligned}\mathcal{E}^{(4)}_{\mathrm{Sk,e}}(\vec{r}) = \sum_{t=0,1}\Big[&\mathbb{A}^{(4,1)}_{\mathrm{t,e}}\left(\Delta D^{1,1}_t\right)\left(\Delta D^{1,1}_t\right) + \mathbb{A}^{(4,2)}_{\mathrm{t,e}} D^{1,1}_t D^{\Delta,\Delta}_t + \mathbb{A}^{(4,3)}_{\mathrm{t,e}} D^{(\nabla,\nabla)}_t D^{(\nabla,\nabla)}_t \\ &+ \mathbb{A}^{(4,4)}_{\mathrm{t,e}} \sum_{\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu} + \mathbb{A}^{(4,5)}_{\mathrm{t,e}} \sum_{\mu\nu} D^{\nabla,\nabla}_{t,\mu\nu}\left(\nabla_\mu\nabla_\nu D^{1,1}_t\right) \\ &+ \mathbb{A}^{(4,6)}_{\mathrm{t,e}} \sum_{\mu\nu} C^{1,\nabla\sigma}_{t,\mu\nu}\left(\Delta C^{1,\nabla\sigma}_{t,\mu\nu}\right) + \mathbb{A}^{(4,7)}_{\mathrm{t,e}} \sum_{\mu\nu\kappa}\left(\nabla_\mu C^{1,\nabla\sigma}_{t,\mu\kappa}\right)\left(\nabla_\nu C^{1,\nabla\sigma}_{t,\nu\kappa}\right) \\ &+ \mathbb{A}^{(4,8)}_{\mathrm{t,e}} \sum_{\mu\nu} C^{1,\nabla\sigma}_{t,\mu\nu} C^{\Delta,\nabla\sigma}_{t,\mu\nu}\Big].\end{aligned}$$

This looks intimidating . . . to you and me, but not to a computer!
Every term is a scalar and can be coded as

```
E(xi) = E(xi)+CC * Density_1(xi,mu,..)*[Derivative] Density_2(xi,mu,...)
```

This kind of notation allows a computer to see the structure

$$D_{\mu}^{1,\nabla} \rightarrow \texttt{D\_I\_N\_m}$$

This kind of notation allows a computer to see the structure

$$D_{\mu}^{1,\nabla} \rightarrow \texttt{D\_I\_N\_m}$$
$$C_{\mu\nu}^{\Delta,\nabla\sigma} \rightarrow \texttt{C\_NkNk\_NmSo}$$

This kind of notation allows a computer to see the structure

$$D_{\mu}^{1,\nabla} \rightarrow \texttt{D\_I\_N\_m}$$
$$C_{\mu\nu}^{\Delta,\nabla\sigma} \rightarrow \texttt{C\_NkNk\_NmSo}$$

Full terms in a functional

ρQ	$D^{1,1}\, D^{\Delta,\Delta}$	E_D_I_I_D_NmNm_NkNk
$\Delta\rho\Delta\rho$	$\Delta D^{1,1} \Delta D^{1,1}$	E_Lap_D_I_I_Lap_D_I_I
$J_{\mu\nu} V_{\mu\nu}$	$C_{\mu\nu}^{1,\nabla\sigma} C_{\mu\nu}^{\Delta,\nabla\sigma}$	E_C_I_NmSo_C_NkNk_NmSo

Contents

How to calculate a complicated object like:

$$\hat{D}^{\hat{L},\hat{R}}(\vec{r}) = \mathrm{Re}\left\{ \sum_{kj\sigma} \rho_{kj} \left[\hat{L}\psi_j^*\right](\vec{r},\sigma) \left[\hat{R}\psi_k\right](\vec{r},\sigma) \right\} ?$$

We represent

$$\psi(\vec{r}) = \begin{pmatrix} \psi_1(\vec{r}) \\ \psi_2(\vec{r}) \\ \psi_3(\vec{r}) \\ \psi_4(\vec{r}) \end{pmatrix} = \begin{pmatrix} \mathrm{Re}\psi(\vec{r},\uparrow) \\ \mathrm{Im}\psi(\vec{r},\uparrow) \\ \mathrm{Re}\psi(\vec{r},\downarrow) \\ \mathrm{Im}\psi(\vec{r},\downarrow) \end{pmatrix}$$

How to calculate a complicated object like:

$$\hat{D}^{\hat{L},\hat{R}}(\vec{r}) = \mathrm{Re}\left\{ \sum_{kj\sigma} \rho_{kj} \left[\hat{L}\psi_j^*\right](\vec{r},\sigma) \left[\hat{R}\psi_k\right](\vec{r},\sigma) \right\} ?$$

We represent

$$\psi(\vec{r}) = \begin{pmatrix} \psi_1(\vec{r}) \\ \psi_2(\vec{r}) \\ \psi_3(\vec{r}) \\ \psi_4(\vec{r}) \end{pmatrix} = \begin{pmatrix} \mathrm{Re}\psi(\vec{r},\uparrow) \\ \mathrm{Im}\psi(\vec{r},\uparrow) \\ \mathrm{Re}\psi(\vec{r},\downarrow) \\ \mathrm{Im}\psi(\vec{r},\downarrow) \end{pmatrix}$$

So

$$\mathrm{Re}\left\{ \sum_{\sigma} \psi_j^*(\vec{r},\sigma)\psi_k(\vec{r},\sigma) \right\} = \psi_{j,1}\psi_{k,1} + \psi_{j,2}\psi_{k,2} + \psi_{j,3}\psi_{k,3} + \psi_{j,4}\psi_{k,4}$$

$$\mathrm{Im}\left\{ \sum_{\sigma} \psi_j^*(\vec{r},\sigma)\psi_k(\vec{r},\sigma) \right\} = \psi_{j,1}\psi_{k,2} - \psi_{j,2}\psi_{k,1} + \psi_{j,3}\psi_{k,4} - \psi_{j,4}\psi_{k,3}$$

How to calculate a complicated object like:

$$\hat{D}^{\hat{L},\hat{R}}(\vec{r}) = \mathrm{Re}\left\{ \sum_{kj\sigma} \rho_{kj} \left[\hat{L}\psi_j^*\right](\vec{r},\sigma) \left[\hat{R}\psi_k\right](\vec{r},\sigma) \right\} ?$$

We represent

$$\psi(\vec{r}) = \begin{pmatrix} \psi_1(\vec{r}) \\ \psi_2(\vec{r}) \\ \psi_3(\vec{r}) \\ \psi_4(\vec{r}) \end{pmatrix} = \begin{pmatrix} \mathrm{Re}\psi(\vec{r},\uparrow) \\ \mathrm{Im}\psi(\vec{r},\uparrow) \\ \mathrm{Re}\psi(\vec{r},\downarrow) \\ \mathrm{Im}\psi(\vec{r},\downarrow) \end{pmatrix}$$

$$\mathrm{Re}\left\{ \sum_{\sigma} \left[\hat{L}\psi_j^*\right](\vec{r},\sigma) \left[\hat{R}\psi_k\right](\vec{r},\sigma) \right\} = \hat{L}\psi_{j,1}\hat{R}\psi_{k,1} + \hat{L}\psi_{j,2}\hat{R}\psi_{k,2} + \hat{L}\psi_{j,3}\hat{R}\psi_{k,3} + \hat{L}\psi_{j,4}\hat{R}\psi_{k,4}$$

Some FORTRAN for ρ

Yale

BEFORE Hephaestos

```
do wave=1,nwt                                  ! nwt = total number of spwfs
    it = 2 ;  if(wave.le.nwn) it = 1! proton or neutron?
    weight  = rho_can(wave)                    ! rho_{wave wave}

    do i=1,mv                                  ! mv = total number of
                                               !      mesh points
$EXPRESSION                              ! The magic happens here
    enddo
enddo
```

AFTER `Hephaestos`

```
do wave=1,nwt                                  ! nwt = total number of spwfs
    it = 2 ;  if(wave.le.nwn) it = 1! proton or neutron?
    weight  = rho_can(wave)                    ! rho_{wave wave}

    do i=1,mv                                  ! mv = total number of
                                               !      mesh points
    ! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    ! Calculation of density D_I_I
    D_I_I(i,it) = D_I_I(i,it) + weight * (&
           &     + DenPsi(i,1,wave) * DenPsi(i,1,wave)&
           &     + DenPsi(i,2,wave) * DenPsi(i,2,wave)&
           &     + DenPsi(i,3,wave) * DenPsi(i,3,wave)&
           &     + DenPsi(i,4,wave) * DenPsi(i,4,wave))
    enddo
    !DenPsi(i,l,wave) = \psi(\vec{r}_i)_{wave,l}
enddo
```

The keystone module

```
from string     import Template
template        = 'ABC${TEST}EF' ; dic = {'TEST', 'D'}
filled_string = template.substitute(dic) # = ABCDEF
```

Yale

The keystone module

```
from string     import Template
template        = 'ABC${TEST}EF' ; dic = {'TEST', 'D'}
filled_string = template.substitute(dic) # = ABCDEF
```

Two templates to the calculation of ρ

```
$NAME(i$IND,it) = $NAME(i$IND,it) + $WEIGHT * (
```

```
$SIGN $LEFTWF(i$LIND,$LCOMP,$LEFTWAVE) *
      $RIGHTWF(i$RIND,$RCOMP,$RIGHTWAVE)
```

$$\begin{aligned}\rho(x_i) = \rho(x_i) + \rho_{jk}[& \\ & \psi_{k,1}(x_i)\psi_{j,1}(x_i) \\ & + \psi_{k,2}(x_i)\psi_{j,2}(x_i) \\ & + \psi_{k,3}(x_i)\psi_{j,3}(x_i) \\ & + \psi_{k,4}(x_i)\psi_{j,4}(x_i)]\end{aligned}$$

```
D_I_I(i,it) = D_I_I(i,it) + weight * (&
&   + DenPsi(i,1,wave)*DenPsi(i,1,wave)&
&   + DenPsi(i,2,wave)*DenPsi(i,2,wave)&
&   + DenPsi(i,3,wave)*DenPsi(i,3,wave)&
&   + DenPsi(i,4,wave)*DenPsi(i,4,wave))
```

The role of operators

Yale

How do we keep track of the action of $\hat{L}$ and $\hat{R}$?

Yale

How do we keep track of the action of $\hat{L}$ and $\hat{R}$?

Consider operators as functions

$$\begin{aligned}
\nabla_\mu &: (\psi_1, \psi_2, \psi_3, \psi_4) \to \left([\nabla_\mu \psi]_1, [\nabla_\mu \psi]_2, [\nabla_\mu \psi]_3, [\nabla_\mu \psi]_4\right) \\
\sigma_x &: (\psi_1, \psi_2, \psi_3, \psi_4) \to (+\psi_3, +\psi_4, +\psi_1, +\psi_2) \\
\sigma_y &: (\psi_1, \psi_2, \psi_3, \psi_4) \to (+\psi_4, -\psi_3, -\psi_2, +\psi_1) \\
\sigma_z &: (\psi_1, \psi_2, \psi_3, \psi_4) \to (+\psi_1, +\psi_2, -\psi_3, -\psi_4)
\end{aligned}$$

The role of operators

Yale

How do we keep track of the action of $\hat{L}$ and $\hat{R}$?

Consider operators as functions

$$\begin{aligned}
\nabla_\mu &: (\psi_1, \psi_2, \psi_3, \psi_4) \to \left([\nabla_\mu \psi]_1, [\nabla_\mu \psi]_2, [\nabla_\mu \psi]_3, [\nabla_\mu \psi]_4\right) \\
\sigma_x &: (\psi_1, \psi_2, \psi_3, \psi_4) \to (+\psi_3, +\psi_4, +\psi_1, +\psi_2) \\
\sigma_y &: (\psi_1, \psi_2, \psi_3, \psi_4) \to (+\psi_4, -\psi_3, -\psi_2, +\psi_1) \\
\sigma_z &: (\psi_1, \psi_2, \psi_3, \psi_4) \to (+\psi_1, +\psi_2, -\psi_3, -\psi_4)
\end{aligned}$$

An operator is a function that takes

$$(l_1, l_2, l_3, l_4) \to \nabla^n_{\mu_1, \ldots \mu_n}(\pm l'_1, \pm l'_2, \pm l'_3, \pm l'_4)$$

Characterized by

- Derivative order n
- Derivative indices $\mu_1, \ldots \mu_n$
- A permutation of $(1, 2, 3, 4)$
- A set of four signs $(\pm, \pm, \pm, \pm)$

And here the true magic happens:

Yale

Functions can be composed !

$$\hat{L} = \nabla_\mu \nabla_\nu \hat{\sigma} \qquad \sim \qquad \mathcal{F} = f \circ g \circ h$$

And here the true magic happens:

Yale

Functions can be composed !

$$\hat{L} = \nabla_\mu \nabla_\nu \hat{\sigma} \qquad \sim \qquad \mathcal{F} = f \circ g \circ h$$

- Derivative order $n = n_1 + n_2$
- Derivative indices $\{\mu\} = \{\mu_1^1, \ldots \mu_{n_1}^1\} + \{\mu_1^2, \ldots \mu_{n_1}^2\}$

- A composite permutation of $(1, 2, 3, 4)$
- $(\pm, \pm, \pm, \pm) = (\pm_1, \pm_1, \pm_1, \pm_1) \times (\pm_2, \pm_2, \pm_2, \pm_2)$

And here the true magic happens:

Yale

Functions can be composed !

$$\hat{L} = \nabla_\mu \nabla_\nu \hat{\sigma} \qquad \sim \qquad \mathcal{F} = f \circ g \circ h$$

- Derivative order $n = n_1 + n_2$
- Derivative indices $\{\mu\} = \{\mu_1^1, \ldots \mu_{n_1}^1\} + \{\mu_1^2, \ldots \mu_{n_1}^2\}$
- A composite permutation of $(1, 2, 3, 4)$
- $(\pm, \pm, \pm, \pm) = (\pm_1, \pm_1, \pm_1, \pm_1) \times (\pm_2, \pm_2, \pm_2, \pm_2)$

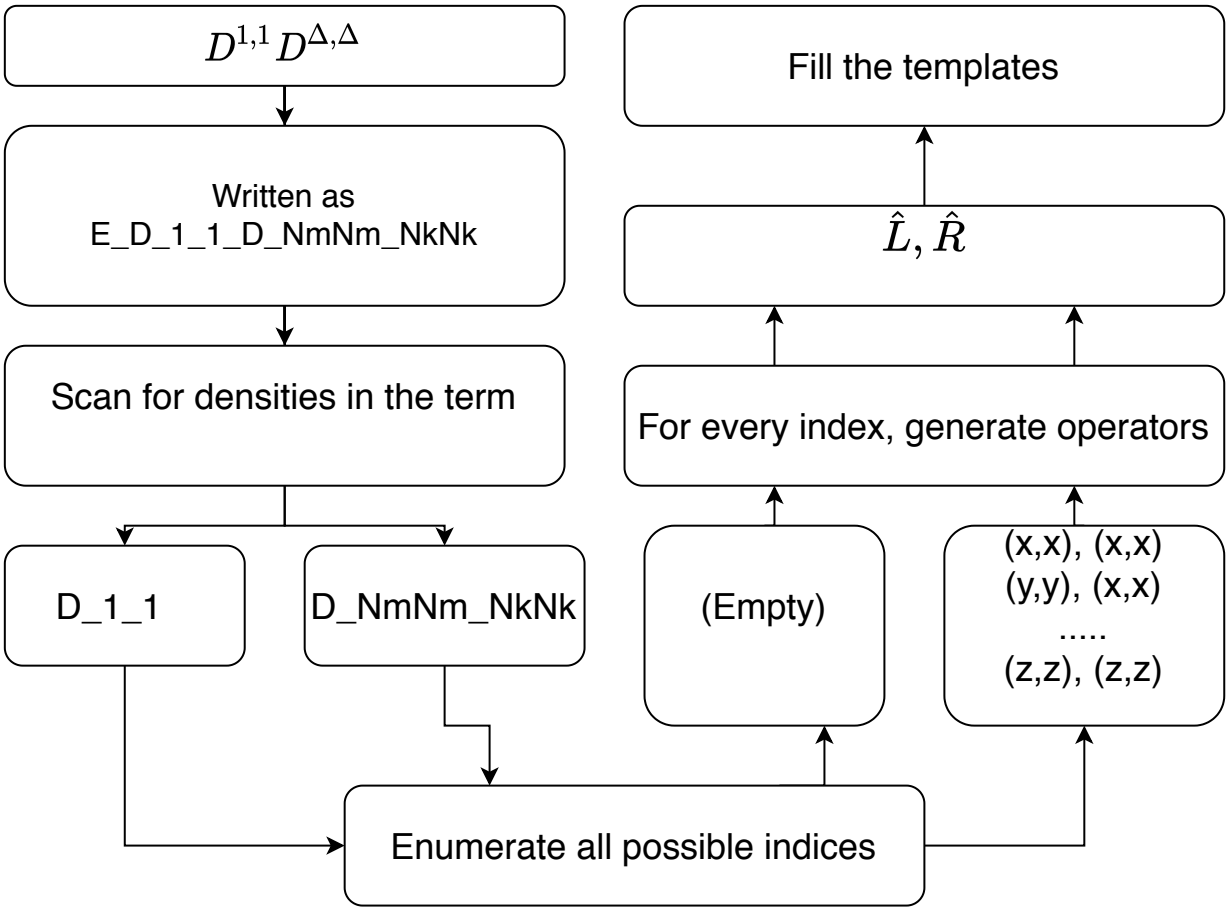
The "only" Python needed

```
def Identity(mu,indices):
  # Identity
def Nabla(mu,indices):
  # Gradient in direction mu
def Sigma(mu,indices):
  # Pauli sigma matrices
```

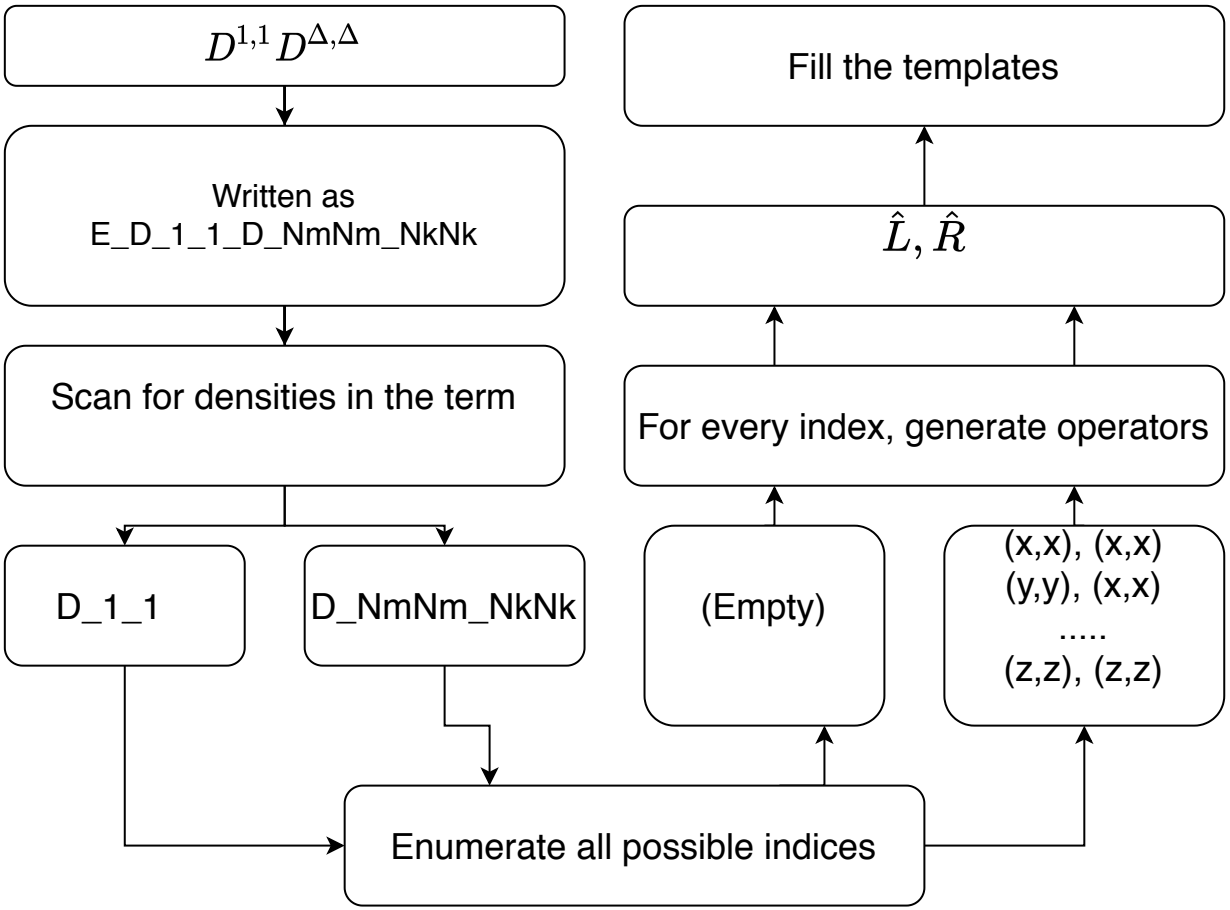
```
def Current(mu,indices):
  # Turn a D into a C
def TR(mu,indices):
# Incorporate time-reversal
def Combine( L , R ):
  # Composition rules
```

The Hephaestos logic (up to here)

Yale



The Hephaestos logic (up to here)



Things that take a lot of lines of code

- Enumeration of the indices ($\geq$95% time)
- Writing miscellaneous statements
- Doing things optimally
- ...

```
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! Calculation of density D_NmNm_NkNk
D_NmNm_NkNk(i,it) = D_NmNm_NkNk(i,it) + weight * (&
 &     + DenddPsi(i,1,1,wave) * DenddPsi(i,1,1,wave)&
 &     + DenddPsi(i,1,2,wave) * DenddPsi(i,1,2,wave)&
 &     + DenddPsi(i,1,3,wave) * DenddPsi(i,1,3,wave)&
 &     + DenddPsi(i,1,4,wave) * DenddPsi(i,1,4,wave)&
 &     + DenddPsi(i,1,1,wave) * DenddPsi(i,4,1,wave)&
 &     + DenddPsi(i,1,2,wave) * DenddPsi(i,4,2,wave)&
 &     + DenddPsi(i,1,3,wave) * DenddPsi(i,4,3,wave)&
 &     + DenddPsi(i,1,4,wave) * DenddPsi(i,4,4,wave)&
 &     + DenddPsi(i,1,1,wave) * DenddPsi(i,6,1,wave)&
 &     + DenddPsi(i,1,2,wave) * DenddPsi(i,6,2,wave)&
 &     + DenddPsi(i,1,3,wave) * DenddPsi(i,6,3,wave)&
 &     + DenddPsi(i,1,4,wave) * DenddPsi(i,6,4,wave)&
 &     + DenddPsi(i,4,1,wave) * DenddPsi(i,1,1,wave)&
 &     + DenddPsi(i,4,2,wave) * DenddPsi(i,1,2,wave)&
 &     + DenddPsi(i,4,3,wave) * DenddPsi(i,1,3,wave)&
 &     + DenddPsi(i,4,4,wave) * DenddPsi(i,1,4,wave)&
 &     + DenddPsi(i,4,1,wave) * DenddPsi(i,4,1,wave)&
 &     + DenddPsi(i,4,2,wave) * DenddPsi(i,4,2,wave)&
 &     + DenddPsi(i,4,3,wave) * DenddPsi(i,4,3,wave)&
 &     + DenddPsi(i,4,4,wave) * DenddPsi(i,4,4,wave)&
 &     + DenddPsi(i,4,1,wave) * DenddPsi(i,6,1,wave)&
 &     + DenddPsi(i,4,2,wave) * DenddPsi(i,6,2,wave)&
 &     + DenddPsi(i,4,3,wave) * DenddPsi(i,6,3,wave)&
 &     + DenddPsi(i,4,4,wave) * DenddPsi(i,6,4,wave)&
 &     + DenddPsi(i,6,1,wave) * DenddPsi(i,1,1,wave)&
 &     + DenddPsi(i,6,2,wave) * DenddPsi(i,1,2,wave)&
 &     + DenddPsi(i,6,3,wave) * DenddPsi(i,1,3,wave)&
 &     + DenddPsi(i,6,4,wave) * DenddPsi(i,1,4,wave)&
 &     + DenddPsi(i,6,1,wave) * DenddPsi(i,4,1,wave)&
 &     + DenddPsi(i,6,2,wave) * DenddPsi(i,4,2,wave)&
 &     + DenddPsi(i,6,3,wave) * DenddPsi(i,4,3,wave)&
 &     + DenddPsi(i,6,4,wave) * DenddPsi(i,4,4,wave)&
 &     + DenddPsi(i,6,1,wave) * DenddPsi(i,6,1,wave)&
 &     + DenddPsi(i,6,2,wave) * DenddPsi(i,6,2,wave)&
 &     + DenddPsi(i,6,3,wave) * DenddPsi(i,6,3,wave)&
 &     + DenddPsi(i,6,4,wave) * DenddPsi(i,6,4,wave))
```

```
!- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! Calculation of E_D_I_I_D_NmNm_NkNk
Edensity = 0.0_dp

! indices = ()
EDensity(:,3) = Edensity(:,3) + sum(D_I_I(:,:),2) &
&                             *sum(D_NmNm_NkNk(:,:),2)

[....]

E_D_I_I_D_NmNm_NkNk(1,1) = dv * ( &
& (B_D_I_I_D_NmNm_NkNk(1,2)+0.5*B_D_I_I_D_NmNm_NkNk(2,2)) &
& * sum(Edensity(:,3)))
```

Contents

The single-particle hamiltonian

Yale

To do the actual optimization, we need to represent the single-particle hamiltonian

$$\begin{aligned} h(\vec{r}\sigma, \vec{r}'\sigma') = & h_e^{(0)}(\vec{r}\sigma, \vec{r}'\sigma') + h_o^{(0)}(\vec{r}\sigma, \vec{r}'\sigma') \\ & + h_e^{(2)}(\vec{r}\sigma, \vec{r}'\sigma') + h_o^{(2)}(\vec{r}\sigma, \vec{r}'\sigma') \\ & + h_e^{(4)}(\vec{r}\sigma, \vec{r}'\sigma') + h_o^{(4)}(\vec{r}\sigma, \vec{r}'\sigma') \,. \end{aligned}$$

The single-particle hamiltonian

To do the actual optimization, we need to represent the single-particle hamiltonian

$$\begin{aligned} h(\vec{r}\sigma,\vec{r}'\sigma') = & \, h_e^{(0)}(\vec{r}\sigma,\vec{r}'\sigma') + h_o^{(0)}(\vec{r}\sigma,\vec{r}'\sigma') \\ & + h_e^{(2)}(\vec{r}\sigma,\vec{r}'\sigma') + h_o^{(2)}(\vec{r}\sigma,\vec{r}'\sigma') \\ & + h_e^{(4)}(\vec{r}\sigma,\vec{r}'\sigma') + h_o^{(4)}(\vec{r}\sigma,\vec{r}'\sigma') \, . \end{aligned}$$

Example:

$$\hat{h}_e^{(4)}|\psi_i\rangle = \Delta F^{\Delta,\Delta}(\vec{r})\Delta|\psi_i\rangle - \frac{i}{2}\sum_{\mu\nu}\left[G_{\mu\nu}^{\Delta,\nabla\sigma}(\vec{r})\Delta\nabla_\mu\hat{\sigma}_\nu + \Delta G_{\mu\nu}^{\Delta,\nabla\sigma}(\vec{r})\nabla_\mu\hat{\sigma}_\nu\right]|\psi_i\rangle \, ,$$

With fields that can be constructed as

The single-particle hamiltonian

To do the actual optimization, we need to represent the single-particle hamiltonian

$$\begin{aligned} h(\vec{r}\sigma,\vec{r}'\sigma') = & \, h_e^{(0)}(\vec{r}\sigma,\vec{r}'\sigma') + h_o^{(0)}(\vec{r}\sigma,\vec{r}'\sigma') \\ & + h_e^{(2)}(\vec{r}\sigma,\vec{r}'\sigma') + h_o^{(2)}(\vec{r}\sigma,\vec{r}'\sigma') \\ & + h_e^{(4)}(\vec{r}\sigma,\vec{r}'\sigma') + h_o^{(4)}(\vec{r}\sigma,\vec{r}'\sigma') \, . \end{aligned}$$

Example:

$$\hat{h}_e^{(4)}|\psi_i\rangle = \Delta F^{\Delta,\Delta}(\vec{r})\Delta|\psi_i\rangle - \frac{i}{2}\sum_{\mu\nu}\left[G_{\mu\nu}^{\Delta,\nabla\sigma}(\vec{r})\Delta\nabla_\mu\hat{\sigma}_\nu + \Delta G_{\mu\nu}^{\Delta,\nabla\sigma}(\vec{r})\nabla_\mu\hat{\sigma}_\nu\right]|\psi_i\rangle \, ,$$

With fields that can be constructed as

$$F^{\hat{L},\hat{R}}(\vec{r}) \equiv \frac{\delta E(\mathrm{R})}{\delta D^{\hat{L},\hat{R}}(\vec{r})} \, , \qquad G^{\hat{L},\hat{R}}(\vec{r}) \equiv \frac{\delta E(\mathrm{R})}{\delta C^{\hat{L},\hat{R}}(\vec{r})} \, ,$$

The single-particle hamiltonian

To do the actual optimization, we need to represent the single-particle hamiltonian

$$\begin{aligned} h(\vec{r}\sigma,\vec{r}'\sigma') = & \, h_e^{(0)}(\vec{r}\sigma,\vec{r}'\sigma') + h_o^{(0)}(\vec{r}\sigma,\vec{r}'\sigma') \\ & + h_e^{(2)}(\vec{r}\sigma,\vec{r}'\sigma') + h_o^{(2)}(\vec{r}\sigma,\vec{r}'\sigma') \\ & + h_e^{(4)}(\vec{r}\sigma,\vec{r}'\sigma') + h_o^{(4)}(\vec{r}\sigma,\vec{r}'\sigma') \, . \end{aligned}$$

Example:

$$\hat{h}_e^{(4)}|\psi_i\rangle = \Delta F^{\Delta,\Delta}(\vec{r})\Delta|\psi_i\rangle - \frac{i}{2}\sum_{\mu\nu}\left[G_{\mu\nu}^{\Delta,\nabla\sigma}(\vec{r})\Delta\nabla_\mu\hat{\sigma}_\nu + \Delta G_{\mu\nu}^{\Delta,\nabla\sigma}(\vec{r})\nabla_\mu\hat{\sigma}_\nu\right]|\psi_i\rangle \, ,$$

With fields that can be constructed as

$$F^{\hat{L},\hat{R}}(\vec{r}) \equiv \frac{\delta E(\mathrm{R})}{\delta D^{\hat{L},\hat{R}}(\vec{r})} \, , \qquad G^{\hat{L},\hat{R}}(\vec{r}) \equiv \frac{\delta E(\mathrm{R})}{\delta C^{\hat{L},\hat{R}}(\vec{r})} \, ,$$

Automating the fields F, G

Yale

For bilinear functionals, derivatives are very easy

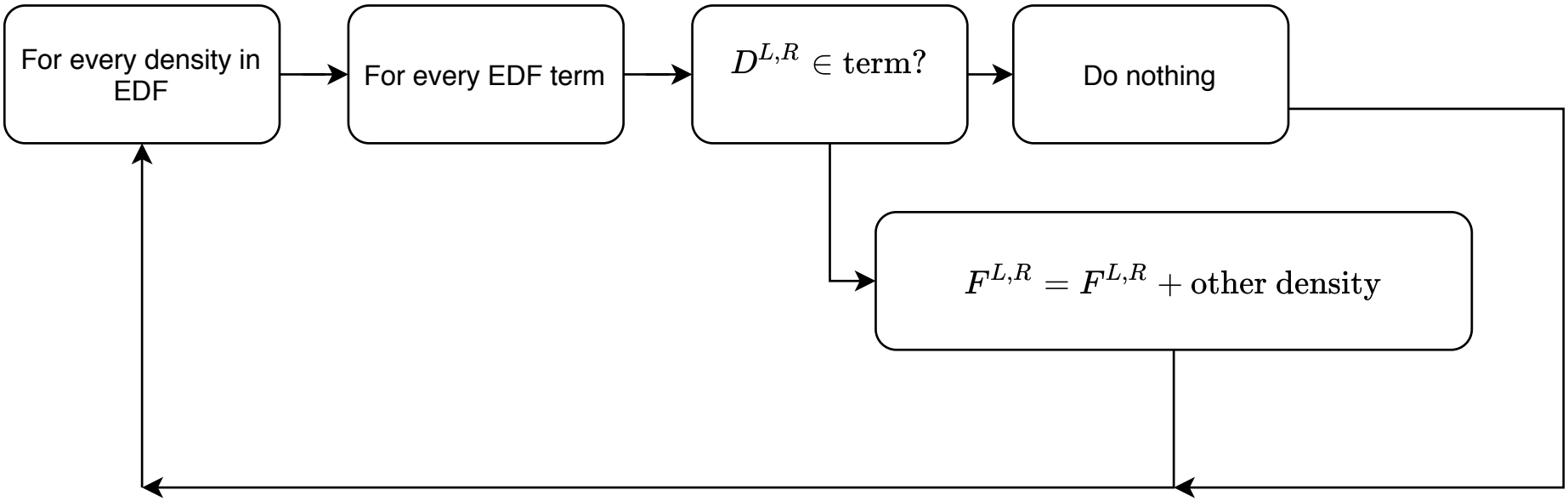
$$\frac{\partial}{\partial D^{1,1}}\left(D^{1,1}D^{\Delta,\Delta}\right) = D^{\Delta,\Delta}\,, \qquad \frac{\partial}{\partial D^{\Delta,\Delta}}\left(D^{1,1}D^{\Delta,\Delta}\right) = D^{1,1}\,.$$

Yale

For bilinear functionals, derivatives are very easy

$$\frac{\partial}{\partial D^{1,1}}\left(D^{1,1}D^{\Delta,\Delta}\right) = D^{\Delta,\Delta}\,, \qquad \frac{\partial}{\partial D^{\Delta,\Delta}}\left(D^{1,1}D^{\Delta,\Delta}\right) = D^{1,1}\,.$$

All the code needs to do



Automating the action of $\hat{h}$

Yale

The variation with respect to $\hat{D}^{\hat{L},\hat{R}}$ is particularly easy to write down

$$\left[\hat{h}\psi\right](\vec{r},\sigma) \sim \left[\hat{L}\,F^{\hat{L},\hat{R}}\,\hat{R}\,\psi\right](\vec{r},\sigma)\,.$$

The variation with respect to $\hat{D}^{\hat{L},\hat{R}}$ is particularly easy to write down

$$\left[\hat{h}\psi\right](\vec{r},\sigma) \sim \left[\hat{L}\,F^{\hat{L},\hat{R}}\,\hat{R}\,\psi\right](\vec{r},\sigma)\,.$$

Hence anything can be coded as

$$\left[\hat{h}\psi\right](\vec{r},\sigma) \sim \left[\hat{L}\Big(F^{\hat{L},\hat{R}}(\vec{r}) \times \left[\hat{R}\psi\right]\Big)\right](\vec{r},\sigma)$$

This is truly the most difficult part of a HFB code to write by hand

Automating the action of $\hat{h}$

Yale

The variation with respect to $\hat{D}^{\hat{L},\hat{R}}$ is particularly easy to write down

$$\left[\hat{h}\psi\right](\vec{r},\sigma) \sim \left[\hat{L}\,F^{\hat{L},\hat{R}}\,\hat{R}\,\psi\right](\vec{r},\sigma)\,.$$

Hence anything can be coded as

$$\left[\hat{h}\psi\right](\vec{r},\sigma) \sim \left[\hat{L}\Big(F^{\hat{L},\hat{R}}(\vec{r}) \times \left[\hat{R}\psi\right]\Big)\right](\vec{r},\sigma)$$

This is truly the most difficult part of a HFB code to write by hand

But this can be achieved again by composing Python functions!

Yale

Action of $\hat{h}$ corresponding to $C^{\Delta,\nabla\sigma}$

```
temp = 0.0
do i=1,mv
    !                                          R[psi]= nabla_x sigma_x [psi]
    !                                                  |
    !                                                  v
    temp(i,1) =  temp(i,1) +  G_NkNk_NS(i,1,1,it) * dpsi(i,1,4)
    temp(i,2) =  temp(i,2) -  G_NkNk_NS(i,1,1,it) * dpsi(i,1,3)
    temp(i,3) =  temp(i,3) +  G_NkNk_NS(i,1,1,it) * dpsi(i,1,2)
    temp(i,4) =  temp(i,4) -  G_NkNk_NS(i,1,1,it) * dpsi(i,1,1)
    !                                  ^
    !                                  |
    !                            multiplication with field
enddo
.........
[8 more copies with slightly different indices ]
.........

!  Acting with L = Laplacian
call Derive_lap(temp(:,1), sx(1), sy(1), sz(1), laptemp(:,1))
call Derive_lap(temp(:,2), sx(2), sy(2), sz(2), laptemp(:,2))
call Derive_lap(temp(:,3), sx(3), sy(3), sz(3), laptemp(:,3))
call Derive_lap(temp(:,4), sx(4), sy(4), sz(4), laptemp(:,4))

! Adding the final result to the other terms
hpsi(:,1:4) =  hpsi(:,1:4) +  laptemp(:,1:4)
```

Contents

(ALMOST HANDS-ON EXAMPLE)

Two ways to calculate the energy

$$E = \int d\vec{r}\,\mathcal{E}(\vec{r}) = \frac{1}{2}E_{\text{kin}} + \sum_i v_i^2 \epsilon_i + E_{\text{pair}} + \text{rearrangement terms}$$

Automation can enforce consistency

Yale

Two ways to calculate the energy

$$E = \int d\vec{r}\,\mathcal{E}(\vec{r}) = \frac{1}{2}E_{\mathrm{kin}} + \sum_i v_i^2 \epsilon_i + E_{\mathrm{pair}} + \text{rearrangement terms}$$

"Magic formula test" for ^{12}C with SN2LO1

(no coulomb, coarse mesh, small box)

Code	Integrating EDF	Summing spwfs	Relative error
MOCCa	-98.82**125342164**	-98.82**074979838**	10^{-5}

Two ways to calculate the energy

$$E = \int d\vec{r}\,\mathcal{E}(\vec{r}) = \frac{1}{2}E_{\mathrm{kin}} + \sum_i v_i^2 \epsilon_i + E_{\mathrm{pair}} + \text{rearrangement terms}$$

"Magic formula test" for ^{12}C with SN2LO1
(no coulomb, coarse mesh, small box)

Code	Integrating EDF	Summing spwfs	Relative error
MOCCa	-98.82**125342164**	-98.82**074979838**	10^{-5}
Tantalus	-98.82335969697	-98.82335969697	$< 10^{-14}$

Contents

Status of the code(s)

Yale

Input functional forms

- Must be bilinear . . .
- . . . but may be density-dependent
- No restrictions on pairing part

  Example: SkP EDF
- Time-odd terms OK!

Tantalus; actual mean-field

- Feature parity with EV8

  WR et al. CPC 187 (2015).
- Improved
  - → speed
  - → precision
  - → useability
- No feature parity with MOCCa

Status of the code(s)

Yale

Input functional forms

- Must be bilinear . . .
- . . . but may be density-dependent
- No restrictions on pairing part

  Example: SkP EDF
- Time-odd terms OK!

Tantalus; actual mean-field

- Feature parity with EV8

  WR et al. CPC 187 (2015).
- Improved
  - → speed
  - → precision
  - → useability
- No feature parity with MOCCa

Future

- Trilinear, quadrilinear, . . . , n-linear terms

  cf. SLyMR0 J. Sadoudi et al. Phys. Scr. T154, 14013 (2013).
- General symmetry breaking for the mean-field part
- LaTeX output (input?)
- Multi-reference implementations with the same machinery!

Status of the code(s)

Yale

Hephaestos/Tantalus has been used already to

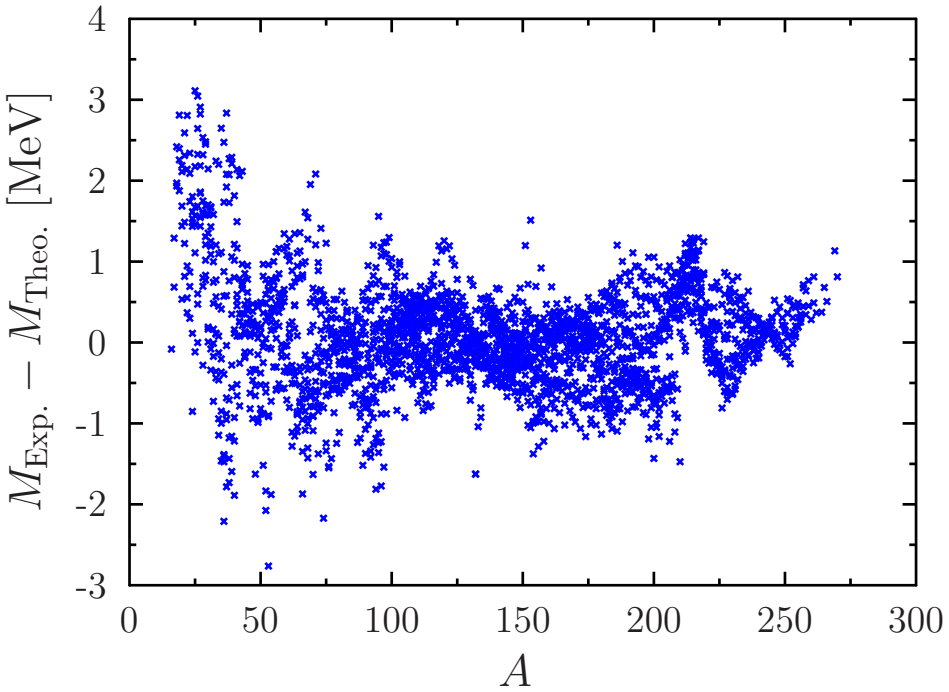
- Bugcheck of the fitting code (WHISKY) used for SN2LO1
- Check the equivalency of different N2/3LO functional forms
- . . . including some forms with N2LO tensor terms
- Construct toy N3LO parameterizations

Status of the code(s)

Yale

Hephaestos/Tantalus has been used already to

- Bugcheck of the fitting code (WHISKY) used for SN2LO1
- Check the equivalency of different N2/3LO functional forms
- . . . including some forms with N2LO tensor terms
- Construct toy N3LO parameterizations
- Construct new EDF fits!



- NLO Skyrme with some corrections
- First 3D $\vec{r}$-space fits
- RMS on **2408** masses:
  - New fit: **0.590** MeV.
  - BSK27: **0.512** MeV.
  - UNEDF0: **1.4** MeV
- Neural networks for fitting (G. Scamps)

Work of G. Scamps, E. Olsen and S. Goriely (ULB).

Conclusions

Yale

Automation of functional implementation is **my** (the?) future.

- Functionals get more and more complicated.

  Ex: tensor terms, NℓLO, NℓLO without restrictions, SLyMRℓ, Fayans, REGℓd, ...
- Time to implementation grows quickly with complexity ...
- ...but is nothing compared to debugging time.

Conclusions

Yale

Automation of functional implementation is **my** (the?) future.

- Functionals get more and more complicated.

  Ex: tensor terms, NℓLO, NℓLO without restrictions, SLyMRℓ, Fayans, REGℓd, ...
- Time to implementation grows quickly with complexity ...
- ...but is nothing compared to debugging time.

At least for bilinear Skyrme functionals

- Automation has been achieved: NLO, N2LO, N3LO, ...
- The most important ingredient: abstraction of densities.

Thanks!

Yale

Thanks for. . .

. . . the **help**:

Cool (T=0) people

- P.-H. Heenen ULB
- M. Bender IP2I
- P. Becker IP2I
- D. Davesne IP2I
- J. Meyer IP2I

Thanks!

Yale

Thanks for...

...the **help**:

Cool (T=0) people

- P.-H. Heenen ULB
- M. Bender IP2I
- P. Becker IP2I
- D. Davesne IP2I
- J. Meyer IP2I

Hot ($T \geq 0$) people

- Y. Alhassid Yale
- P. Fanto Yale
- S. Vartak Yale
- S. Jensen Yale

Thanks!

Yale

Thanks for. . .

. . . the **help**:

Cool (T=0) people

- P.-H. Heenen ULB
- M. Bender IP2I
- P. Becker IP2I
- D. Davesne IP2I
- J. Meyer IP2I

Hot ($T \geq 0$) people

- Y. Alhassid Yale
- P. Fanto Yale
- S. Vartak Yale
- S. Jensen Yale

Stellar ($\star$) people

- S. Goriely ULB
- G. Scamps ULB
- E. Olsen ULB

Thanks!

Yale

Thanks for. . .

. . . the **help**:

Cool (T=0) people

- P.-H. Heenen ULB
- M. Bender IP2I
- P. Becker IP2I
- D. Davesne IP2I
- J. Meyer IP2I

Hot ($T \geq 0$) people

- Y. Alhassid Yale
- P. Fanto Yale
- S. Vartak Yale
- S. Jensen Yale

Stellar ($\star$) people

- S. Goriely ULB
- G. Scamps ULB
- E. Olsen ULB

. . . the **CPU time**!

- NERSC, DOE, USA.
- GRACE, YALE, USA.
- CÉCI network, Belgium.
- CC of IN2P3 , France.

Thanks!

Yale

Thanks for. . .

. . . the **help**:

Cool (T=0) people

- P.-H. Heenen ULB
- M. Bender IP2I
- P. Becker IP2I
- D. Davesne IP2I
- J. Meyer IP2I

Hot ($T \geq 0$) people

- Y. Alhassid Yale
- P. Fanto Yale
- S. Vartak Yale
- S. Jensen Yale

Stellar ($\star$) people

- S. Goriely ULB
- G. Scamps ULB
- E. Olsen ULB

. . . the **CPU time**!

- NERSC, DOE, USA.
- GRACE, YALE, USA.
- CÉCI network, Belgium.
- CC of IN2P3 , France.

. . . your **attention**!

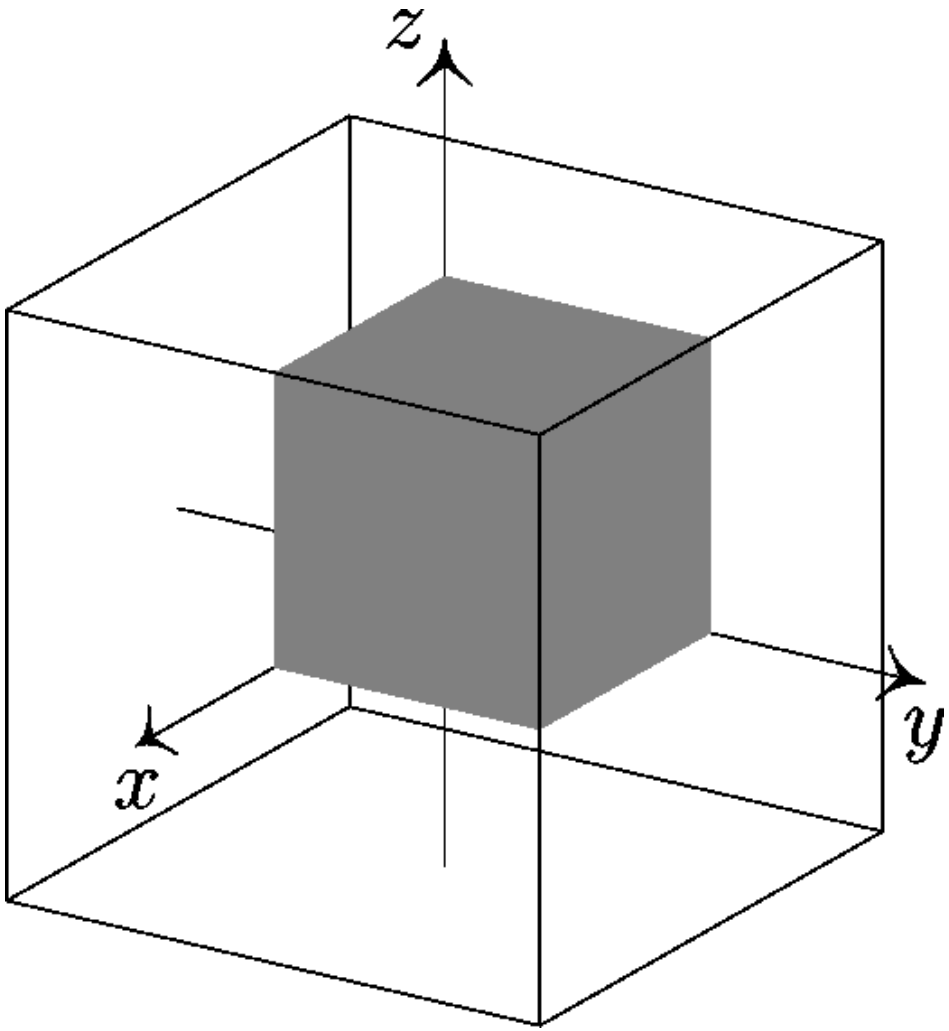
Yale

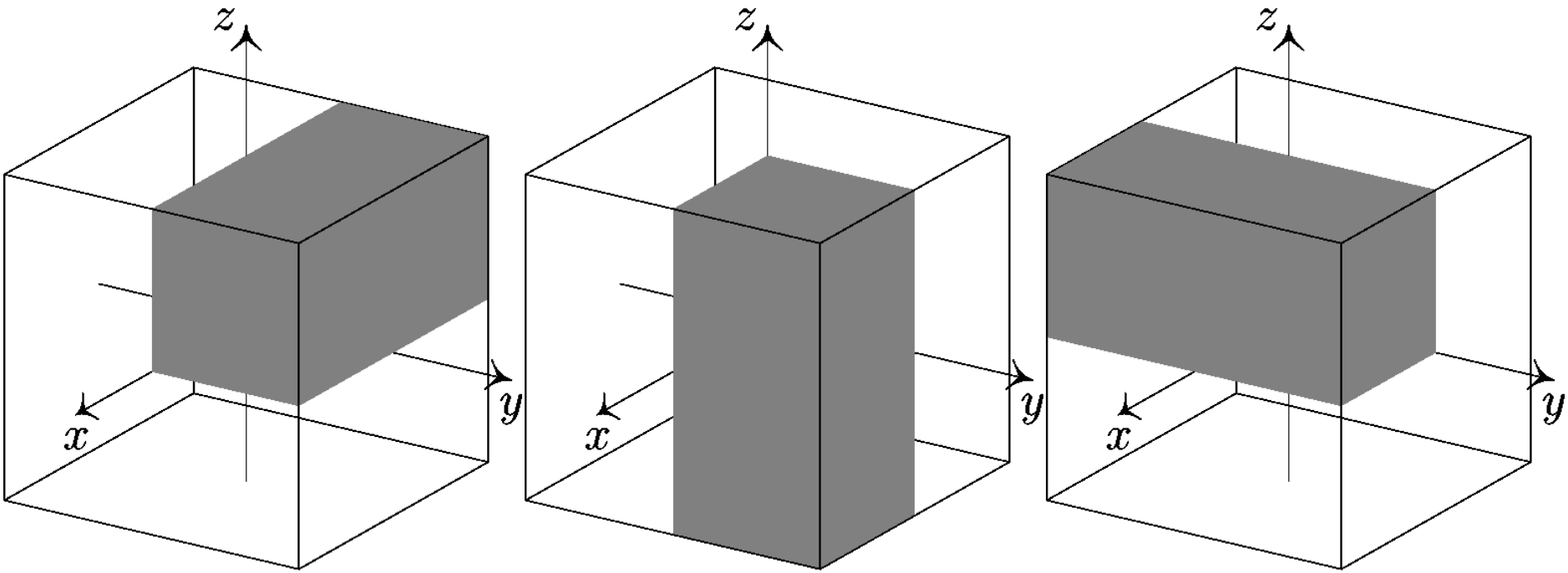
Bonus!

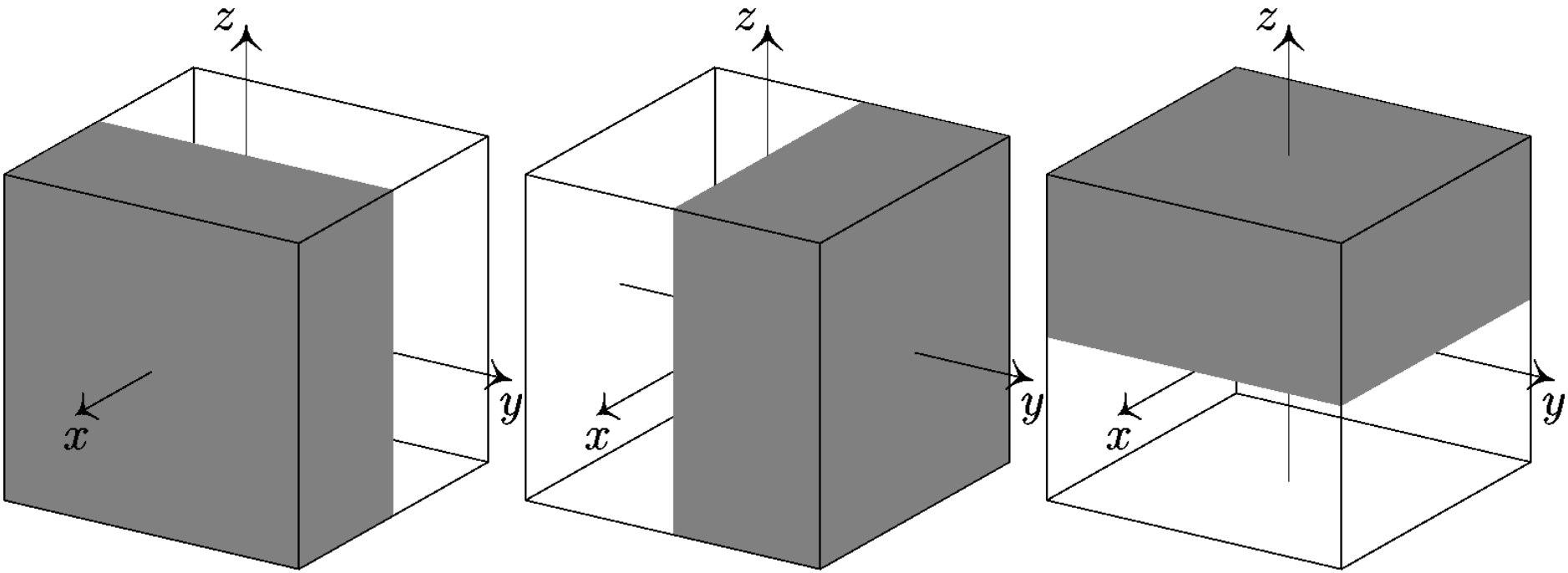
Things I try to pay attention to

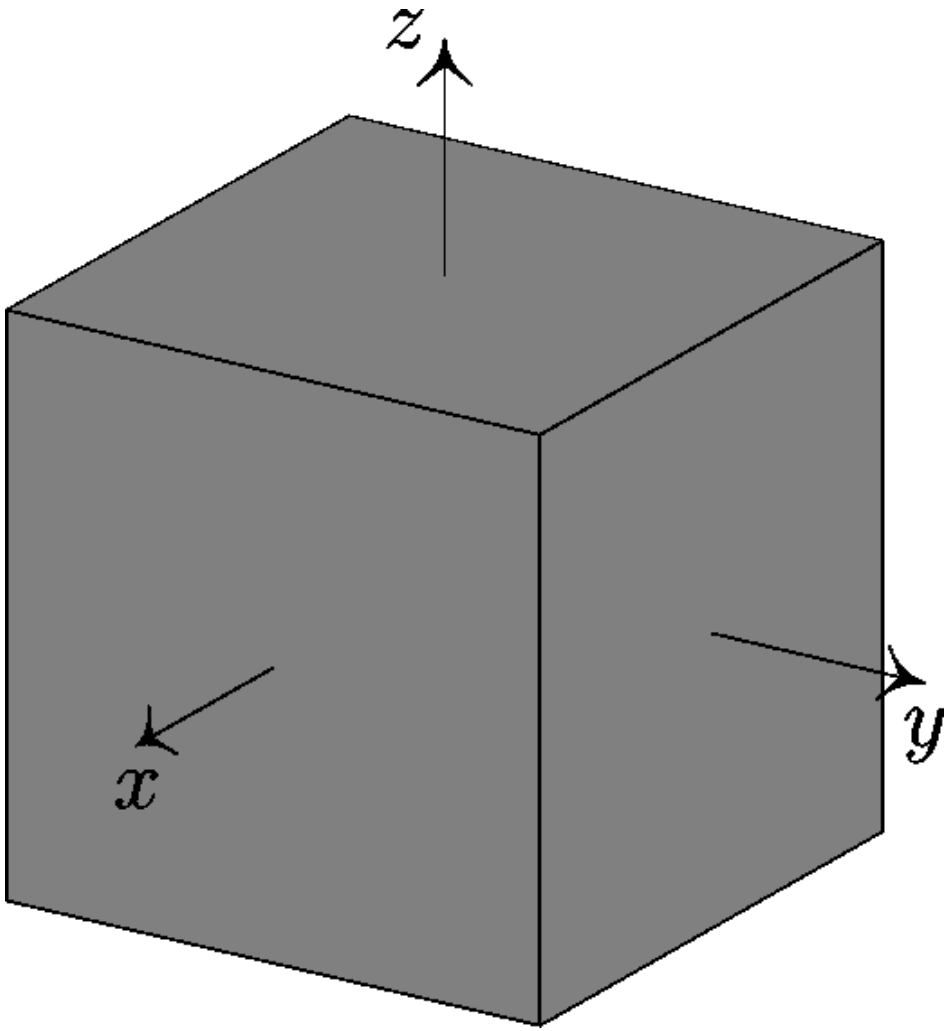
Yale

- Make sure you make everything abstract enough
  - → It was not clear to me when starting on N2LO functionals that these structures were so regular
  - → Even when I thought of the densities, I did not realize I would be able to do the fields and $\hat{h}$
- Make an effort to keep things human(-readable) at every stage
  - → It is easy to automate comments
  - → It is easy to automate print statements
  - → Make every code output logs
  - → Put in as much consistency checks as you can possibly think of
- Don't make your code TOO clever
  - → Hephaestos makes no decisions for me
  - → Density dependences are somewhat 'by hand', because they are not worth automating.


z
x
y

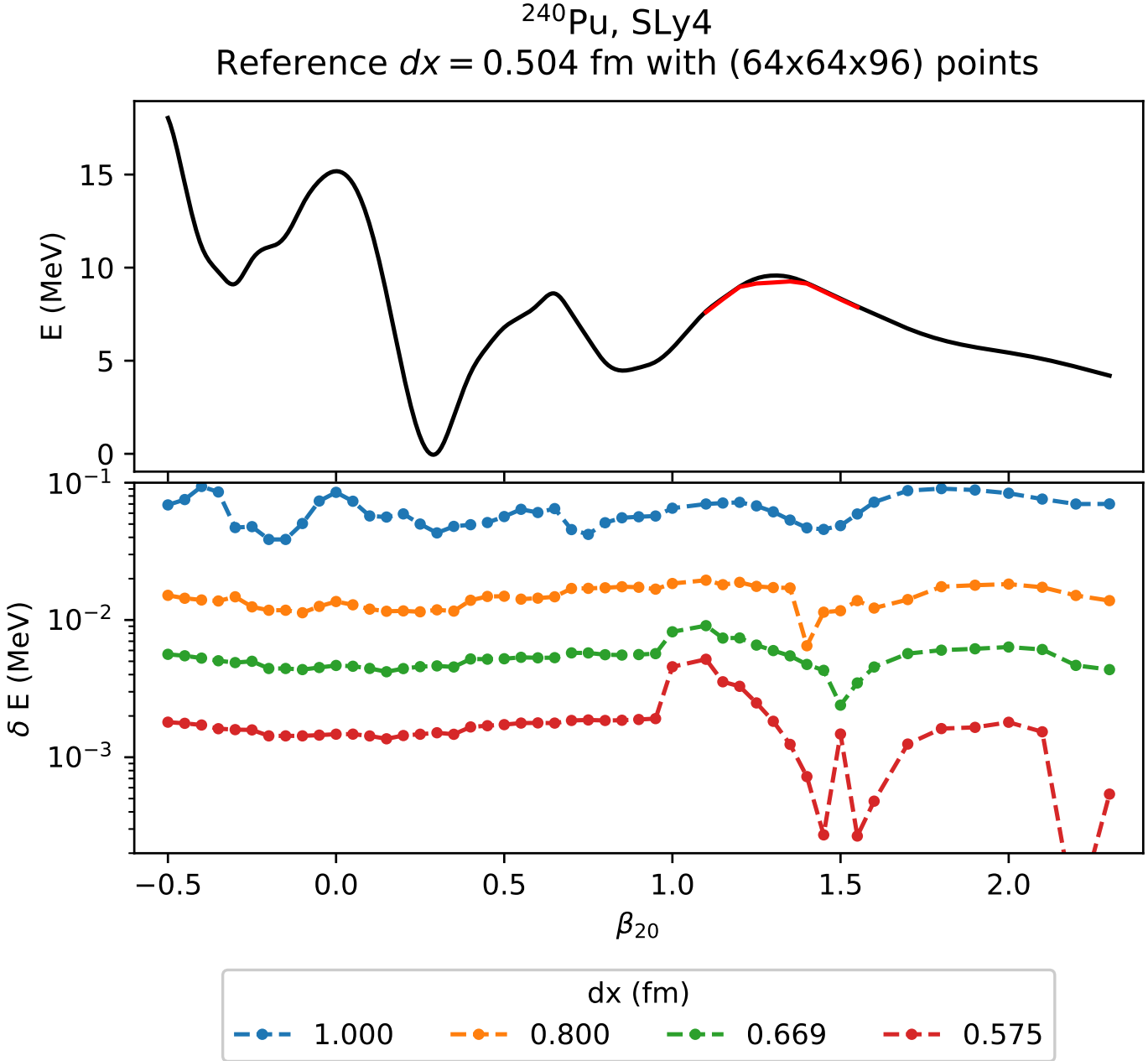

z
y
x
z
y
x
z
y
x

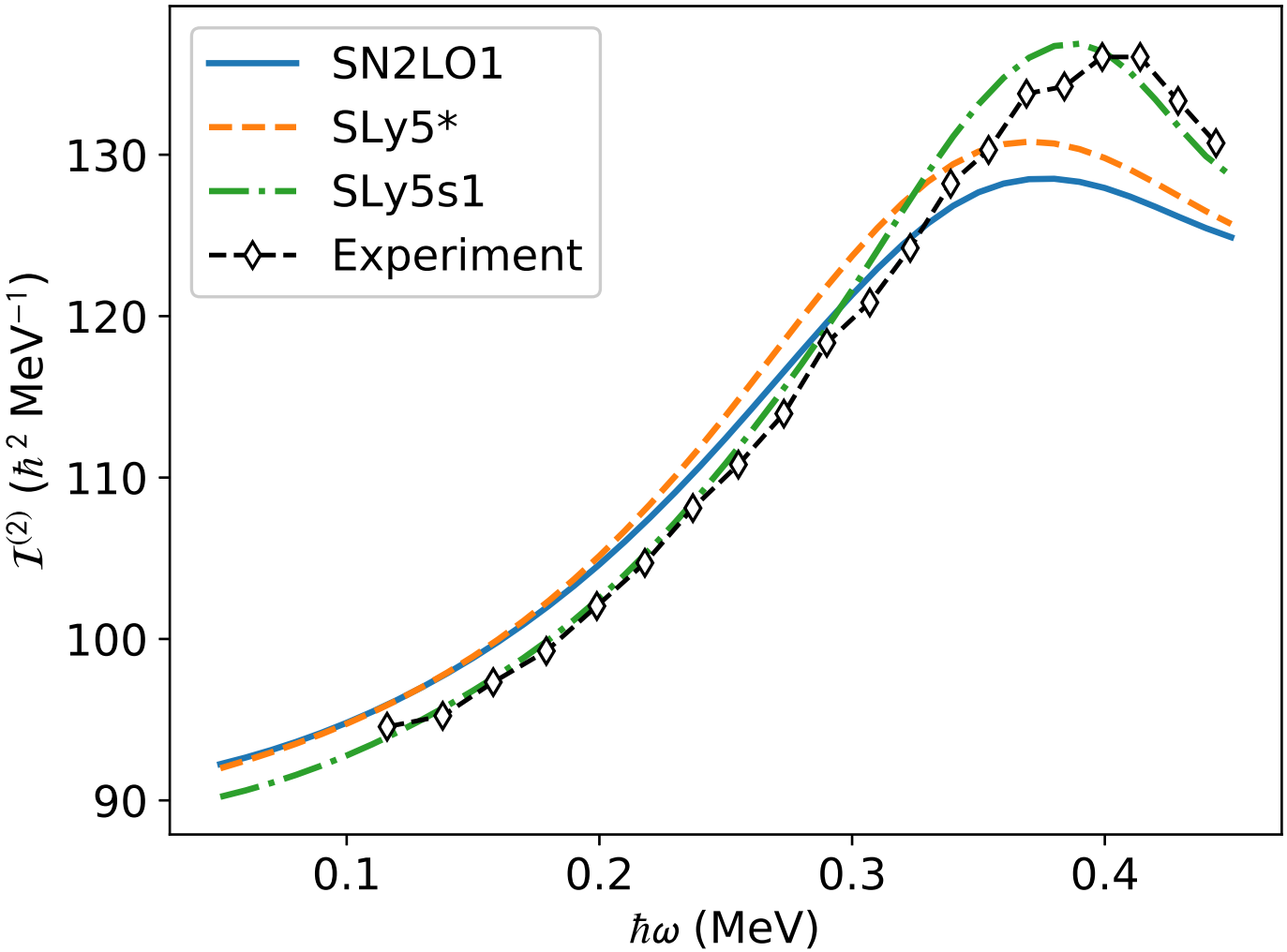

z
x
y
z
x
y
z
x
y

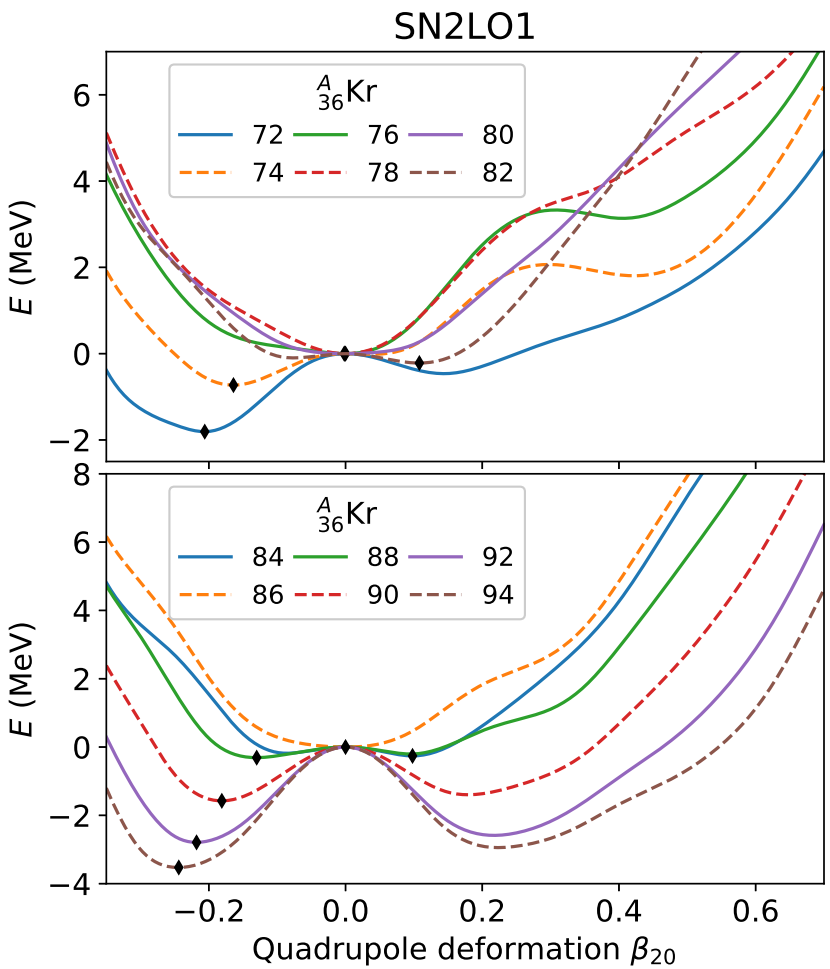

z
x
y

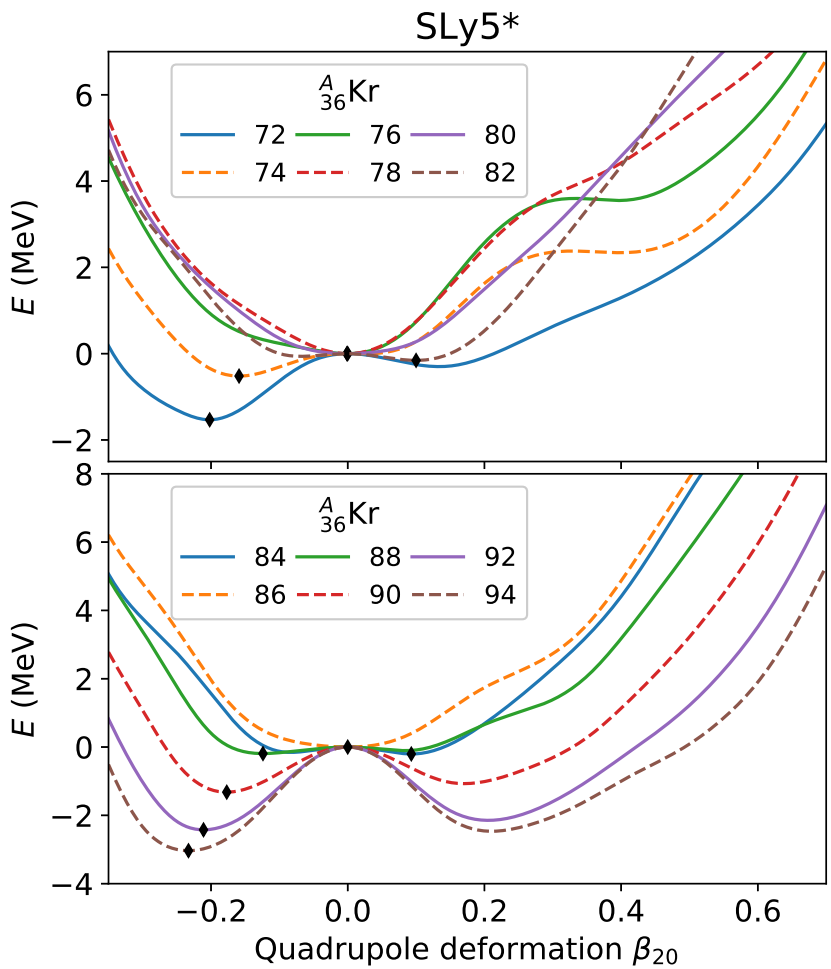
Why coordinate space?

Yale




SN2LO1
SLy5*
SLy5s1
Experiment
130
120
110
100
90
$\mathcal{I}^{(2)}$ ($\hbar^2$ MeV^{-1})
0.1
0.2
0.3
0.4
$\hbar\omega$ (MeV)


SN2LO1
$^{A}_{36}$Kr
72 76 80
74 78 82
84 88 92
86 90 94
E (MeV)
Quadrupole deformation β_{20}


SLy5*
$^{A}_{36}$Kr
72 76 80
74 78 82
84 88 92
86 90 94
E (MeV)
Quadrupole deformation β_{20}