HubbardTweezer Documentation

This is an introductory manual for the code on github and used on the calculations in the paper. For scientific principles, please refer to the paper main text.

Dependencies

In order to run the code, you need to install the following packages:

- `scipy` along with `numpy`
- `pymanopt` which depends on `torch`
- `opt_einsum`
- `nlopt`
- `ortools`
- `configobj`

Get started on HubbardTweezer

In general, the code is run by feeding an `ini` file in which required parameters are set.

```
python Hubbard_exe.py parameters.ini
```

The code calculates, and generates results in the same `ini` file.

`ini` file structure

`ini` is a text file format. It has a section-property structure. In one file it has one or more sections. In every section, key-value pairs provide properties:

```
[Section A]
key1 = value1
key2 = value2
...
[section B]
...
```

What the program does is to read parameters set in `[Parameters]` section and write calculation results to the other sections, such as `[Singleband_Parameters]` for single-band Hubbard parameters, `[Equalization_Result]` for equalization solutions and `[Trap_Adjustments]` for how the traps need to be adjusted in experiment to realize the desired Hubbard parameters.

Data type: number vs array

Specifically for the program to read a length=1 array, what needs to do is to add a comma after the number:

```
number = 2 # 2 read as number
tuple = 2, # (2,) read as a tuple
```

Example

Calculate single-band Hubbard parameters for a 2x2 square lattice

We write the input `2x2.ini` file as below:

```
[DVR_Parameters]
N = 20
L0 = 3, 3, 7.2
DVR_dimension = 3
[Trap_Parameters]
V0 = 52.26
waist = 1000,
scattering_length = 1770
laser_wavelength = 780
[Lattice_Parameters]
shape = square
lattice_size = 2, 2
lattice_const = 1550, 1600
lattice_symmetry = True
[Equalization_Parameters]
equalize = False
[Verbosity]
write_log = False
verbosity = 3
```

Then we run the command (make sure give the program correct paths):

```
python Hubbard_exe.py 2x2.ini
```

After calculation result is given by appending contents to the end of `2x2.ini` file:

```
[Singleband_Parameters]
t_ij = "[[0.0, 0.1864950982522663, 0.26852018226162067, 0.003420732961071802], [0.1864950982522671
V_i = "[-9.315215265814913e-12, 6.394884621840902e-14, 9.301004411099711e-12, -5.684341886080802e-
U_i = "[1.2134538518651798, 1.2134538518666564, 1.2134538518681128, 1.2134538518666398]"
wf_centers = "[[-0.725424845124012, -0.7620829309237793], [-0.7254248451240122, 0.7620829309237923
[Trap_Adjustments]
V_offset = "[1.0, 1.0, 1.0, 1.0]"
trap_centers = "[[-0.775, -0.7999999999999999], [-0.775, 0.7999999999999999], [0.775, -0.799999999
waist_factors = "[[1.0, 1.0], [1.0, 1.0], [1.0, 1.0], [1.0, 1.0]]"
[Equalization_Result]
x = "[1.0, -0.775, -0.7999999999999999]"
cost_func_by_terms = "[5.560385638393359e-12, 1.9205510638480715e-12, 3.52929827971391e-11]"
cost_func_value = 3.534519983120769e-11
total_cost_func = 3.5779897142888145e-11
func_eval_number = 0
U_target = 1.2134538518666473
t_target = 0.18649509825197869, 0.26852018226140717
V_target = -1.7763568394002505e-15
scale_factor = 0.1864950982516911
success = False
equalize_status = -1
termination_reason = Not equalized
U_over_t = 6.506625982346819
```

The other sections in the file are not what we are interested in.

Example 2: Equalize Hubbard parameters for a 4-site chain

Here we want to equalize Hubbard parameters for a 4-site chain by `trf` optimization algorithm in `scipy`, without using ghost trap or waist tuning. The input file `4x1_eq.ini` is as below:

```
[DVR_Parameters]
N = 20
L0 = 3, 3, 7.2
DVR_dimension = 3
[Trap_Parameters]
V0 = 52.26
waist = 1000,
scattering_length = 1770
laser_wavelength = 780
[Lattice_Parameters]
shape = square
lattice_size = 4,
lattice_const = 1550,
lattice_symmetry = True
[Equalization_Parameters]
equalize = True
equalize_target = UvT
waist_direction = None
U_over_t = None
method = trf
no_bounds = False
[Verbosity]
write_log = False
verbosity = 3
```

The main difference is in `[Equalization_Parameters]` section. By running the same command as above

```
python Hubbard_exe.py 4x1_eq.ini
```

we get the result:

```
[Equalization_Result]
x = "[1.027480937300892, 1.0083650796908388, -2.354612841530081, -0.7875721977039343]"
cost_func_by_terms = "[0.25916330530145887, 0.07763234841109246, 0.010500917429445717]"
cost_func_value = 0.27074465756771354
total_cost_func = 0.27074465756771354
func_eval_number = 40
U_target = 1.3045697992761218
t_target = 0.20750237974990482, None
V_target = 0
scale_factor = 0.20750237974990482
success = True
equalize_status = 2
termination_reason = `ftol` termination condition is satisfied.
U_over_t = 6.653609207877738
[Trap_Adjustments]
V_offset = "[1.027480937300892, 1.0083650796908388, 1.0083650796908388, 1.027480937300892]"
trap_centers = "[[-2.354612841530081, -0.0], [-0.7875721828027731, -0.0], [0.7875721828027731, -0.
waist_factors = "[[1.0, 1.0], [1.0, 1.0], [1.0, 1.0], [1.0, 1.0]]"
[Singleband_Parameters]
t_ij = "[[0.0, 0.2069326134671067, 0.007771384980274503, 0.0005508604690104194], [0.20693261346711
V_i = "[0.002178965356165463, -0.002178965356165463, -0.0021789653561725686, 0.002178965356165463]
U_i = "[1.3548364097993686, 1.2472824046115605, 1.2472824046115616, 1.3548364097993688]"
wf_centers = "[[-2.3110489105373313, 0.0], [-0.7903690147813551, 0.0], [0.790369014781355, 0.0], [
```

Parameter definitions

Items to input the file

In this section, `Nsite` is the number of trap sites.

`[DVR_Parameters]`

- `N` : (integer) number of DVR grid points from the outermost trap center to the box edges (default: `20`)
- `L0` : (3-entry array) x, y and z direction distances from the outermost trap center to the box edges in unit of x direction waist w_x (default: `3, 3, 7.2`)
- `DVR_dimension` : (integer) DVR grid spatial dimension (default: `1`)

`[Lattice_Parameters]`

- `shape` : (string) lattice shape.

Supported strings: `square`, `Lieb`, `triangular`, `honeycomb`, `defecthoneycomb`, `kagome` and `custom` (default: `square`)

- `lattice_constant`: (tuple or float) the x and y directions lattice spatial scaling, in unit of nm
  if `shape` is `custom`, it is the unit for `site_locations`
  if `shape` is not `custom`, it is lattice spacing
  if only one number is given e.g. `1500`, this means $a_x = a_y$ (default: `1520, 1690`)

If `shape` is not `custom`, the following parameter is read:

- `lattice_size`: (tuple or integer) the number of traps in each lattice dimension
  if only one number is given, this means the lattice is a 1D chain (default: `4,`)

If `shape` is `custom`, the following two parameters are read:

- `site_locations`: (`Nsite` x 2 array) trap centers in unit of `lattice_constant` (default: `None`)
  the `i`-th row is the `(x,y)` coordinate for the `i`-th trap site (`i=0,1,..., Nsite - 1`)
- `bond_links`: (number of bonds x 2 array) used in Hubbard parameter equalization to decide which pairs of sites' tunneling will be equalized (default: `None`)
  each row is a bond, i.e. link between a pair of sites `(i,j)`, with integers `i` and `j` trap site indices

The next parameter specifies whether to use lattice reflection symmetries in the DVR calculation. If this is enabled, only the `(x<=0, y<=0)` quadrant tweezer array parameters, including the trap center locations and the trap depths are used in the calculation. The other quadrants are overwritten by the copy of the `(x<=0, y<=0)` quadrant. Therefore, if the system is not reflection symmetric, please don't set to `True`.

- `lattice_symmetry`: (bool) use lattice x- and y-reflection symmetry (default: `True`)

`[Trap_Parameters]`

- `scattering_length`: (float) scattering length in unit of Bohr radius a_0 (default:

`1770`)

- `waist`: (tuple or float) x and y direction waist (w_x, w_y) in unit of nm (default: `1000, 1000`)
  if only one is set it means $w_x = w_y$
- `atom_mass`: (float) atom mass in unit of amu (default: `6.015122`)
- `laser_wavelength`: (float) laser wavelength in unit of nm (default: `780`)
- `zR`: (tuple or float, optional) x and y direction Rayleigh range ($z_{R,x}$, $z_{R,y}$) in unit of nm (default: `None`)
  `None` means calculated from `waist` and `laser_wavelength`

> **Set trap depths for each trap**
>
> The trap depths of each trap is $\text{trap depth} = V_{\text{offset}} \times V_0$
> where V_0 is a number specifying the frequency scale and V_{offset} is an array of scale factors of each trap. They are the two next parameters listed.

- `V0`: (float) trap depth frequency scale in unit of kHz (default: 104.52)

input in `[Trap_Adjustment]`

- `V_offset`: (`Nsite`-entry array) trap depth factors for each trap (default: `None`)
  if `lattice_symmetry` is `True`, only the `(x<=0,y<=0)` quadrant of the lattice will be used, and the rest of the trap depths input will be overwritten
  if `equalize` is `True`, `V_offset` information is overridden by `x`, see details in "input in `[Equalization_Result]`" section
  `None` means $V_{\text{offset}} = 1$ over the entire lattice

`[Hubbard_Settings]`

- `Nintgrl_grid`: (integer) number of grid points in trapezoidal numerical integration of U (default: `200`)
- `band`: (integer) number of bands to be calculated in Hubbard model (default: `1`)
- `offdiagonal_U`: (bool) calculate multi-site interaction U_{ijkl} (default: `False`)
  if it is `True`, it calculates and stores a tensor of N_{site}^4 elements
  only `band=1` is supported

`[Equalization_Parameters]`

- `equalize`: (bool) whether equalize Hubbard parameters or not (default: `False`)
- `equalize_target`: (string) target Hubbard parameters to be equalized (default: `vT`)

> **Explain equalization target**
>
> 1. lowercase `u`, `v`, `t`: to equalize Hubbard parameters without target values, meaning the program minimizes the variance of the Hubbard parameter
> 2. uppercase `U`, `V`, `T`: to equalize Hubbard parameters to target values, meaning the program minimizes the difference between Hubbard parameters and target values
> 3. Multiple letters can be used together, e.g. `uT` means to equalize `u` to a uniform but not specific target value and `T` to target values

- `method`: (string) optimization algorithm to equalize Hubbard parameters (default: `trf`)
  available algorithms: `trf`, `Nelder-Mead`, `SLSQP`, `L-BFGS-B`, `cobyla`, `praxis` and `bobyqa`

Equalization proposal: adjust waist

- `waist_direction`: (optional, string) direction of waist adjustment. `x`, `y`, `xy` are supported
  `None` means no waist adjustment (default: `None`)

Equalization proposal: ghost trap

`shape=custom` is not supported by ghost trap adjustment.

- `ghost_sites`: (optional, bool) add ghost sites to the lattice (default: `False`)
- `ghost_penalty`: (optional, tuple) 2-entry tuple (factor, threshold) of the ghost penalty added to the cost function (default: `1, 1`)
  threshold is in unit of kHz

> **Explain ghost penalty**
>
> ghost_penalty determines how the penalty is added to the equalization cost function. The formula is as below:

> $\text{penalty} = \text{factor} \times \exp[-6(q - \text{threshold})]$

`[Verbosity]`

- `write_log` : (optional, bool) print parameters of every step to log file (default: `False`)
  see `[Equalization_Log]` in output sections
- `verbosity` : (optional, integer `0~3`) levels of how much information printed (default: `0`)

input in `[Equalization_Result]`

- `x` : (optional, 1-D array) initial trap parameters for equalization as a 1-D array used as initial guess for equalization
- `U_over_t` : (float) Hubbard U/t ratio (default: `None`)
  `None` means this value is calculated by the ratio of $\text{avg}U/\text{avg}t_x$ in initial guess

Items output by the program

Here the integers `N` is the number of sites, and `k` is the number of bands.

`[Singleband_Parameters]`

The Hubbard parameters for the single-band Hubbard model, unit kHz.

- `t_ij` : (`N` x `N` array) tunneling matrix between sites `i` and `j`
- `V_i` : (`N` x 1 array) on-site potential at site `i`
- `U_i` : (`N` x 1 array) on-site Hubbard interaction at site `i`
- `wf_centers` : (`N` x 2 array) calculated Wannier orbital center positions

output in `[Trap_Adjustment]`

The factors to adjust traps to equalize Hubbard parameters.

- `V_offset` : (`N` x 1 array) factor to scale individual trap depth, the same item as in the input section
  resulting trap depth $V_{\text{trap}} = V_{\text{offset}} \times V_0$
- `trap_centers` : (`N` x 2 array) trap center position in unit of `waist_x` and `waist_y`

- `waist_factors` : (`N` x 2 array) factor to scale trap waist, resulting x and y waist $w_{x,y} = \text{waist\_factors}_{x,y} \times w_{x,y}$

output in `[Equalization_Result]`

This section lists the equalization status and result.

- `x` : (1-D array) the optimal trap parameters to equalize Hubbard parameters, the same item as in the input part
- `cost_func_by_terms` : (3-entry array) cost function values C_U, C_t, C_V by terms of U, t, and V
- `cost_func_value` : (float) weighted cost function value `feval` to be minimized $\text{feval} = w_1 \times C_U + w_2 \times C_t + w_3 \times C_V$
- `total_cost_func` : (float) equal-weighted total cost function value $C = C_U + C_t + C_V$
- `func_eval_number` : (integer) number of cost function evaluations
- `success` : (bool) minimization success
- `equalize_status` : (integer) termination status of the optimization algorithm
- `termination_reason` : (string) termination message given by the optimization algorithm
- `U_over_t` : (float) U/t ratio, the same item as in the input section

`[Equalization_Log]` (optional)

Log of equalization process, turn on/off by `write_log`. Each item is an array of values introduced in `[Equalization_Result]` bearing the same key name, of which each row refers to one iteration step.

`[Multiband_Parameters]` (optional)

Multiband Hubbard parameters in unit of kHz, turn on if `band > 1`. Parameters have the same format as in `[Singleband_Parameters]`, labeled by band index.

For example, `t_1_ij` is the tunneling matrix between sites `i` and `j` for the 1st band, and `U_12_i` is the on-site Hubbard interaction at site `i` between 1st and 2nd bands.

Code structure

The code consists of two modules `DVR` and `Hubbard`. Their main modules are explained below.

1. `DVR`: DVR spectra calculations
    - `DVR.core`: `DVR` base class to calculate DVR spectra
    - `DVR.const`: constants used in DVR calculations
    - `DVR.wavefunc`: wavefunction calculations
2. `Hubbard`: Hubbard parameter calculations
    - `Hubbard.core` : `MLWF` class to construct maximally localized Wannier funcitons (MLWFs)
    - `Hubbard.equalizer` : `HubbardParamEqualizer` class to equalize Hubbard parameters over all lattice sites
    - `Hubbard.riemann`: functions for Riemannian manifold optimization in constructing MLWFs
    - `Hubbard.eqinit`: functions to initialize trap parameters for equalization
    - `Hubbard.io`: logger and functions to read and write Hubbard parameters in equalization
    - `Hubbard.lattice`: `Lattice` class to define lattice geometry
    - `Hubbard.ghost`: `GhostTrap` class to add ghost traps to the lattice
3. `tools`: tools for data analysis
    - `tools.integrate`: functions to calculate 3D numerical integrals
    - `tools.point_match`: function to match and label MLWFs to the traps
    - `tools.reportIO`: functions to read and write `ini` files
4. `Hubbard_exe.py` : execute script to read inputs and write out Hubbard parameters for given lattice