Wannier states for optical lattices v1.01:
Basic user documentation

Richard Walters, Giovanni Cotugno, Tomi H. Johnson,
Stephen R. Clark, and Dieter Jaksch

August 4, 2014

Here we provide the basic documentation, which shows how to install and run the software, and describes the forms of the inputs and outputs.

Contents

1 Purpose of the software

The software calculates and plots the maximally localized Wannier states for a periodic potential in one, two or three dimensions. These form a localized basis of single-particle states in which to re-express a many-body Hamiltonian. The software calculates the localized hopping and interaction Hubbard parameters resulting from this re-expression for the case where the particles interact via a contact potential, which is relevant to cold atom experiments.

More details on this task and the algorithm are found in [1].

2 Installing and using the software

2.1 Requirements

The software is in the form of MATLAB `.m`-files, so to use it you must have MATLAB.

We also ask that if you use the software for calculations that lead to publications or presentations then please remember to cite the project's CCPForge webpage and the publication in which the approach used by the software is described [1].

2.2 Downloading

To get hold of a copy of the source code, please first join the 'Wannier states for optical lattices' project by clicking the 'Request to join project' button on the top right of the project's CCPForge webpage. If you do not already have an account on CCPForge, you will need to do this first. After this, you will be able to download a package containing the MATLAB files and user documentation from the 'Files' section. The source code is also available on request, please contact the project admins at `mlgws-admins@hebron.cse.rl.ac.uk`.

2.3 Installing

Once you have the source code (a `.zip` file), unzip it into a folder of your choice. No pre-compiling is necessary.

2.4 First run

In the directory in which you have just unzipped the source code, open the file `Wannier.m` in MATLAB. This is a header `.m`-file in which you specify all of the input parameters for your maximally localized generalized Wannier states calculation, namely those that specify the lattice potential and some other numerical parameters (see section 3.1 for details). This header file is setup ready to perform a calculation for the two-dimensional square lattice potential. To perform this calculation, simply press F5. Updates regarding the progress of the calculation will be displayed in MATLAB's command window (see section

3.2 for details). Plots of the potential and maximally localized Wannier states will appear. Additionally, a file called `UniqueName.mat` will appear in each of the subfolders in the `Wannier_data` directory. These files contain the other outputs to the calculation (see section 3.3 for details).

To run a calculation for a different potential, change the inputs accordingly. See section 3.1 for a description of the inputs.

As well as the two-dimensional square lattice potential, a user may quickly run calculations for other common potentials. For example, opening `2DKagome.m` in MATLAB and pressing F5 runs a calculation for the two-dimensional Kagomé potential.

3 Description of inputs and outputs

Here we will describe the inputs specified in `Wannier.m` and what is output when that `.m`-file is run. This will allow a user to understand how to change `Wannier.m` so that it performs their desired calculation and how to extract the results of this calculation. When an example is helpful we will use the two-dimensional square lattice potential.

3.1 Inputs

A user who wants to quickly perform a calculation for a new lattice potential only has to alter the inputs specifying the potential and the number of bands they want to include in their system. Those in a rush can therefore skip all sections other than 3.1.1 and 3.1.2.

Those who want a greater control over how and which parts of the algorithm are performed should look at section 3.1.6, while those who wish to change the set of Hubbard parameters that are calculated should see section 3.1.3. To increase the accuracy of the calculation of the Hubbard interaction parameters, see section 3.1.4, and to change the resolution and range of the plots see section 3.1.5.

3.1.1 The potential

We begin by describing how the periodic potential is defined. The direct lattice describes the periodicity of this potential. However, in optical lattice experiments, where potentials are composed of a small number of plane waves, it is usually more convenient to define the potential in terms of the reciprocal lattice.

- The matrix `G` defines a set of primitive reciprocal lattice vectors for the optical lattice, in units of $1/\lambda$, where λ is some length-scale. The length-scale λ may or may not be the approximate wavelength of the lasers creating the optical lattice potential. The number of rows defines the number of dimensions of the system, and there should be the same number of columns as rows. Specifically, the n-th column `[x_n; y_n; z_n]` of `G` specifies a

primitive reciprocal lattice vector $\mathbf{b}_n = (x_n\hat{\mathbf{x}} + y_n\hat{\mathbf{y}} + z_n\hat{\mathbf{z}})/\lambda$, with the obvious generalization for dimensions less than three.

To give an example, the line

```
G = 4 * pi * [[1; 0] [0; 1]];
```

specifies a set of primitive reciprocal lattice vectors for the two-dimensional square lattice potential. More precisely, the above line of code specifies the two primitive reciprocal lattice vectors $\mathbf{b}_1$ and $\mathbf{b}_2$, defined by

$$\mathbf{b}_1 = \frac{4\pi}{\lambda}\hat{\mathbf{x}}, \tag{1a}$$

$$\mathbf{b}_2 = \frac{4\pi}{\lambda}\hat{\mathbf{y}}, \tag{1b}$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are the unit vectors in the x and y direction of two-dimensional space, respectively.

- Having specified the lattice and some choice of primitive reciprocal lattice vectors, `hkl` and `vi` specify the potential V up to a constant of proportionality. Specifically, the n-th column `[h_n; k_n; l_n]` in `hkl` specifies a reciprocal lattice vector $\mathbf{G}_n = h_n\mathbf{b}_1 + k_n\mathbf{b}_2 + l_n\mathbf{b}_3$. Further, the n-th value `v_n` in `vi` specifies, up to some constant of proportionality, the coefficient

$$v^{(\mathbf{G}_n)} = \frac{1}{\sqrt{\Upsilon}}\int_{\mathrm{PC}} \mathrm{d}\mathbf{r} V(\mathbf{r}) \mathrm{e}^{-\mathrm{i}\mathbf{G}_n\cdot\mathbf{r}}, \tag{2}$$

of the Fourier series representing the potential $V(\mathbf{r})$. Here the integral is over a primitive unit cell of the direct lattice with volume Υ. All coefficients not specified are assumed to be zero. Therefore this defines the potential

$$V(\mathbf{r}) = \frac{2\pi}{\sqrt{\Upsilon}}\sum_n v^{(\mathbf{G}_n)} \mathrm{e}^{\mathrm{i}\mathbf{G}_n\cdot\mathbf{r}}, \tag{3}$$

up to some positive constant of proportionality.

For the two-dimensional square lattice potential, the lines

```
hkl = [[0; 0] [1; 0] [-1; 0] [0; 1] [0; -1]];
vi = [1 -1 -1 -1 -1];
```

specify the non-zero Fourier coefficients of the potential given in the table below.

$\mathbf{G}_n$	$v^{(\mathbf{G}_n)}$ (up to a positive constant)
$\mathbf{0}$	1
$\mathbf{b}_1$	-1
$-\mathbf{b}_1$	-1
$\mathbf{b}_2$	-1
$-\mathbf{b}_2$	-1

Inserting these into equation (3) gives

$$V(x,y) \propto \left[\sin^2\left(\frac{2\pi x}{\lambda}\right) + \sin^2\left(\frac{2\pi y}{\lambda}\right)\right]. \tag{4}$$

- Once its form has been specified up to some positive constant of proportionality, all that is needed to fully specify the potential is its lattice depth $V_0 = \max_{\mathbf{r}} V(\mathbf{r}) - \min_{\mathbf{r}} V(\mathbf{r})$. This is specified, in units of $E_R = h^2/2\mu\lambda^2$, by the variable `v0`. Here h is Planck's constant and μ is the mass of a particle for which you are aiming to calculate the maximally localized Wannier states, hopping and interaction parameters. Note that if you have not chosen λ to be the wavelength of the lasers creating the optical lattice, then E_R is not strictly the recoil energy, but it still provides a useful energy-scale. The potential, for the purposes of plotting will have the minimum value $\min_{\mathbf{r}} V(\mathbf{r}) = 0$.

  For the two-dimensional square lattice potential, the line

  ```
  v0 = 10;
  ```

  defines $V_0 = 10h^2/2\mu\lambda^2$ and so

$$V(x,y) = 10\frac{h^2}{2\mu\lambda^2}\frac{1}{2}\left[\sin^2\left(\frac{2\pi x}{\lambda}\right) + \sin^2\left(\frac{2\pi y}{\lambda}\right)\right]. \tag{5}$$

3.1.2 Approximations made in the calculation of the Wannier states

In order to make the computations tractable on a computer, several approximations are necessary. We now discuss the variables that control these approximations.

- The calculations always assume that only a small number of the lowest energy bands are important. Physical considerations must be used to determine how many, possibly with the aid of a band structure calculation performed by this software as a first step in calculating the maximally localized Wannier states (see section 3.3.1 for the form of the output of the band structure calculation).

  The variable `groups` specifies which bands are included in the calculation. It also specifies the form of the Wannier states, in particular, between which bands mixing is allowed. If mixing between bands is not allowed then these will be ordinary Wannier states. If mixing is allowed then these will be generalized Wannier states.

  To explain the form of the variable, we will use three examples for the case that only the three lowest energy bands are important. From these it should be clear how to generalize.

  Example A:

```
groups = {1,2,3};
```

This disallows interband mixing, corresponding to the calculation of the maximally localized ordinary Wannier states.

Example B:

```
groups = {[1,2,3]};
```

This allows mixing between all bands, corresponding to the calculation of the maximally localized generalized Wannier states.

Example C:

```
groups = {1,[2,3]};
```

Mixing is allowed between bands 2 and 3 only, corresponding to the calculation of the maximally localized ordinary Wannier state for the lowest band, and the maximally localized generalized Wannier states for the first two excited bands.

For the two-dimensional square lattice potential,

```
groups = {1,[2,3]};
```

is an appropriate choice, since the first two excited bands are degenerate while the lowest band is separated in energy.

- For the calculations, the software works in a truncated Fourier space. Expansions in terms of the reciprocal lattice vectors are truncated and terms relating to reciprocal lattice vectors with magnitudes more than $G_{\max}$ are ignored. This effectively limits the spatial resolution to $2\pi/G_{\max}$, and therefore it should be checked that the potential and Wannier states plotted vary on length-scales shorter than this. The variable `Gmax` specifies $G_{\max}$ in units of $1/\lambda$.

  Typically, a choice

  ```
  Gmax = 100;
  ```

  leading to $G_{\max} = 100/\lambda$ and a spatial resolution $\pi\lambda/50$ is usually large enough to result in the accurate description of the Bloch and Wannier states.

- Integrals over a primitive cell of $\mathbf{k}$-space are replaced by a sum over a discrete mesh of wave-vectors. Specifically, the mesh is formed by the wave-vectors $\mathbf{k} = \mathbf{G}/N$ within some primitive cell of the reciprocal lattice, where $\mathbf{G}$ is a reciprocal lattice vector. This is equivalent to working with a periodic system with only N primitive cells in each dimension, and therefore strictly the relation $N \gg 1$ should be satisfied in order for boundary effects to be negligible. The variable `N` specifies the value of N.

  A choice of

```
N = 20;
```

is usually adequate.

3.1.3 Hubbard parameters of interest

The Hubbard hopping interaction parameters are

$$t^{mn}_{\mathbf{R}\mathbf{R}'} = -\int \mathrm{d}\mathbf{r}\; w^{m*}_{\mathbf{R}}(\mathbf{r})\; h_0\; w^{n}_{\mathbf{R}'}(\mathbf{r}), \tag{6a}$$

$$U^{mnop}_{\mathbf{R}\mathbf{R}'\mathbf{R}''\mathbf{R}'''} = g\int \mathrm{d}\mathbf{r}\; w^{m*}_{\mathbf{R}}(\mathbf{r}) w^{n*}_{\mathbf{R}'}(\mathbf{r}) w^{o}_{\mathbf{R}''}(\mathbf{r}) w^{p}_{\mathbf{R}'''}(\mathbf{r}), \tag{6b}$$

where $w^n_{\mathbf{R}}(\mathbf{r})$ is the band-n Wannier function located at site $\mathbf{R}$ and $h_0 = \hbar^2\nabla^2/2\mu + V(\mathbf{r})$ is the single-particle Hamiltonian. Since the Wannier functions are related by translations

$$w^n_{\mathbf{R}}(\mathbf{r}) = w^n_{\mathbf{R}'}(\mathbf{r} + \mathbf{R}' - \mathbf{R}), \tag{7}$$

this reduces the number of Hubbard parameters that need to be calculated. Specifically, all hopping parameters can be worked out from $t^{mn}_{\mathbf{R0}}$. Further, a large number of the interaction parameters (usually the significant parameters) can be worked out from $U^{mmnn}_{\mathbf{00RR}}$, $U^{mmnn}_{\mathbf{000R}}$, $U^{mmmn}_{\mathbf{00RR}}$, $U^{mmmn}_{\mathbf{000R}}$, $U^{mmmn}_{\mathbf{00R0}}$ and $U^{mmnn}_{\mathbf{R0R0}}$.

Our software calculates the dimensionless quantities $t^{mn}_{\mathbf{R0}}/E_R$, $U^{mmnn}_{\mathbf{00RR}}\tilde{g}/gE_R$, $U^{mmnn}_{\mathbf{000R}}\tilde{g}/gE_R$, $U^{mmmn}_{\mathbf{00RR}}\tilde{g}/gE_R$, $U^{mmmn}_{\mathbf{000R}}\tilde{g}/gE_R$, $U^{mmmn}_{\mathbf{00R0}}\tilde{g}/gE_R$ and $U^{mmnn}_{\mathbf{R0R0}}\tilde{g}/gE_R$ for all m and n but only for specific $\mathbf{R}$ (so that this number is finite!). Here $\tilde{g} = E_R\lambda^D$ and D is the dimension. The specific values of $\mathbf{R}$ are determined by the user in `mbSites`.

To be precise, in terms of the popular choice

$$\mathbf{a}_i = 2\pi\epsilon_{ijk}\frac{\mathbf{b}_j \times \mathbf{b}_k}{\mathbf{b}_i \cdot (\mathbf{b}_j \times \mathbf{b}_k)},$$

for the primitive lattice vectors $\mathbf{a}_i$ of the direct lattice, where ϵ_{ijk} is the Levi-Civita symbol (and similarly in lower dimensions), the direct lattice vectors are written $\mathbf{R} = n_1\mathbf{a}_1 + n_2\mathbf{a}_2 + n_3\mathbf{a}_3$ and thus labelled by integers n_i. Each column of the matrix `mbSites` gives a set of integers n_i labelling a direct lattice vector $\mathbf{R}$ for which the Hubbard parameters above are calculated.

3.1.4 Approximations made in the calculation of the Hubbard interaction parameters

Once the gauge (specifying the relationship between the Bloch states and Wannier states) corresponding to the maximally localized Wannier states is found, there are some additional approximations made in calculating the Hubbard interaction parameters. We now discuss the variables that control these approximations.

- The Hubbard hopping parameters are calculated directly from the gauge and thus no further approximations are required. However, the interaction parameters are calculated by evaluating the integrals of Wannier functions over real space, as in equation 6b.

  These integrals are approximated by discrete sums corresponding to a mesh of points in real space. Specifically, in terms of equation 8, the mesh will include points $\mathbf{r} = n_1\mathbf{a}_1/N_1 + n_2\mathbf{a}_2/N_2 + n_3\mathbf{a}_3/N_3$ for integer n_i. The i-th element of the row vector `mbDensity` specifies N_i, the number of points per primitive cell along the i-th direction of the lattice.

  We find, for a two-dimensional calculation, that

  ```
  mbDensity = [10 10];
  ```

  is usually adequate.

  If the `mbDensity` defined by the user has too few rows for the number of dimensions of their calculations, the values from the first row automatically replace those that are missing.

- While the above approximation has reduced the integral to a sum over a discrete but infinite set of points, to make this set finite we further neglect all but a set of points around the origin. Specifically, we neglect contributions from all points other than those satisfying $\mathbf{r} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + x_3\mathbf{a}_3$ with $m_i - 1/2 \leq x_i \leq p_i + 1/2$. The range-setting parameters m_i and p_i are integers. This set of points should be large enough that it will capture all significant contributions to the overlap for the sites specified in `mbSites` (cf. section 3.1.3).

  The first (second) element in the i-th row of matrix `mbCells` specifies m_i (p_i). We find, for a two-dimensional calculation, that

  ```
  mbCells = [-5 5; -5 5];
  ```

  is usually adequate.

  If the `mbCells` defined by the user has too few rows for the number of dimensions of their calculations, the values from the first row automatically replace those that are missing.

3.1.5 Plotting parameters

Similarly to the Hubbard interaction integrand, the potential and maximally localized Wannier functions must be evaluated on a grid of points in real space before they are plotted. The variables `plotDensity` and `plotCells` are defined in analogy to `mbDensity` and `mbCells` (cf. section 3.1.4).

We recommend, for a balance between resolution and computational resources,

```
plotCells = [-1 1];
plotDensity = 100;
```

for one-dimensional problems,

```
plotCells = [-1 1; -1 1];
plotDensity = [25 25];
```

for two-dimensional problems, and

```
plotCells = [-1 1; -1 1; -1 1];
plotDensity = [5 5 5];
```

for three-dimensional problems.

If the `plotDensity` or `plotCells` defined by the user has too few rows for the number of dimensions of their calculations, the values from the first row automatically replace those that are missing.

3.1.6 Calculation admin

Here we describe the parameters that allow the user to tell the software what to calculate and where to save to/load from.

- Start by assigning a unique string `filename` for your calculation, e.g.

  ```
  filename = 'UniqueName';
  ```

  This will decide the name of the saved outputs files (in the case above it will be `UniqueName.mat`) and thus choosing two calculations with the same name may cause outputs to be overwritten or the incorrect data to be loaded.

- The user may then wish to choose which parts of the maximally localized Wannier state calculation to perform. As default, the whole computation will be performed, but the user may wish to only perform part of the calculation, for example, (i) if they think they obtained a non-global minimum of the spread and wish to run the minimization part of the algorithm again starting from a different random set of basis states (ii) think their spread is yet to converge to a minimum and want to run more iterations without starting the whole calculation again.

  To introduce the choices on offer to the user, we will briefly describe the stages of the calculation (see [1] for an explanation of each stage):

  1. The band structure and Bloch states are calculated. This data is saved. The algorithm finds a gauge (relative to the Bloch states) corresponding to the set of maximally localized ordinary Wannier states. This data is saved. If the user has disallowed band mixing then the algorithm skips to stage 3, otherwise it continues.

2. The algorithm finds a gauge (relative to the sam initial Bloch states) corresponding to a set of generalized Wannier states with a reduced inter-band contribution to the spread. Immediately after this, the gauge is transformed such that the intra-band contribution to the spread is minimized (the inter-band contribution is unaltered).
3. The gauge corresponding to the maximally localized generalized Wannier states is found by steepest-descent minimization. This data is saved.
4. The maximally localized Wannier states are explicitly built and the corresponding Hubbard parameters calculated. This data is saved.
5. Plots of the potential and maximally localized Wannier states are displayed.

The parameter `recalcBloch` specifies, in the case that there is a save-file called `UniqueName.mat` containing the band structure, whether or not to recalculate the band structure (and the whole of stage 1) anyway. It should be set to `'true'` if you are trying to overwrite an earlier calculation of the same name, but `'false'` if you want to save time by loading the band structure and gauge corresponding to the maximally localized ordinary Wannier states from an earlier calculation and start at stage 2. Note that if no previous output exists then stage 1 will be performed whether `recalcBloch` is `'true'` or `'false'`.

The parameter `reloadComposite` specifies, in the case that there is a save-file called `UniqueName.mat` containing the gauge corresponding to the maximally localized generalized Wannier states (saved after stage 3), whether or not to start at stage 2 in this gauge. It should be set to `'true'` if you do and `'false'` if you do not.

The parameter `disentangle` specifies whether stage 2 is ever performed. It should be set to `'true'` if it may be performed, but `'false'` if you do not and want to skip straight to stage 3.

The parameter `recalcComposite` specifies whether stages 2 and 3 are performed or whether the calculation skip to stage 4 if a previous output to these stages is found. It should be set to `'true'` if stages 2 and 3 are to be performed, but `'false'` if you want to skip stages 2 and 3 if a previous output exists. Note that if no previous output exists then stages 2 and 3 will be performed whether `recalcComposite` is `'true'` or `'false'`. Although if `disentangle` is set to `'false'` then stage 2 will not be performed either way.

The parameter `recalcManyBody` specifies whether stage 4 is performed or whether the calculation skip to stage 5 if a previous output to stage 4 is found. It should be set to `'true'` if stage 4 is to be performed, but `'false'` if you want to skip stage 4 if a previous output exists. Note that if no previous output exists then stage 4 will be performed whether `recalcManyBody` is `'true'` or `'false'`.

The parameter `makePlots` specifies whether stage 5 is performed. It should be set to `'true'` if it is to be performed, but `'false'` if not.

We will now describe the most common combinations. The following is the default and will perform the full calculation:

```
recalcBloch = 'true';
reloadComposite = 'false';
disentangle = 'true';
recalcComposite = 'true';
recalcManyBody = 'true';
makePlots = 'true';
```

If you think that your calculation got stuck in a non-global minimum (see section 4) and wish to run the calculation again from stage 2 (since stage 1 will be same each time), then choose

```
recalcBloch = 'false';
reloadComposite = 'false';
disentangle = 'true';
recalcComposite = 'true';
recalcManyBody = 'true';
makePlots = 'true';
```

to save some time.

If you think that your previous calculation did not perform enough iterations of the steepest-descent algorithm and want to perform more starting from where you were, then choose

```
recalcBloch = 'false';
reloadComposite = 'true';
disentangle = 'false';
recalcComposite = 'true';
recalcManyBody = 'true';
makePlots = 'true';
```

If you want to save time by performing the steepest-descent algorithm without the initialization procedure, then choose

```
recalcBloch = 'true';
reloadComposite = 'true';
disentangle = 'false';
recalcComposite = 'true';
recalcManyBody = 'true';
makePlots = 'true';
```

If you are only interested in recomputing the Hubbard parameters (stage 4), say you want to compute a different set of them, then choose

```
recalcBloch = 'false';
reloadComposite = 'true';
disentangle = 'false';
recalcComposite = 'false';
recalcManyBody = 'true';
makePlots = 'false';
```

If you are only interested in quickly viewing the plots (stage 5), then choose

```
recalcBloch = 'false';
reloadComposite = 'true';
disentangle = 'false';
recalcComposite = 'false';
recalcManyBody = 'false';
makePlots = 'true';
```

- As well as deciding what to calculate, the user has some flexibility over how it is calculated. For instance, for one-dimensional potentials, the user could opt to use the Parallel Transport algorithm [2], instead of the usual initialization procedure (stage 2), to reduce the inter-band spread. To do this, set

  ```
  parallelTransport = 'true';
  ```

  To perform the matrix multiplication needed to calculate the Hubbard hopping parameters, our software uses a multiproduct technique that saves processing time at the expense of an increased use of memory. If you are having memory problems (possible in three-dimensional calculations) then set

  ```
  calcMode = 'loop';
  ```

  At the beginning of stage 2 of the calculation (see above) the Bloch states of different bands are randomly permuted at each point in **k**-space. This is to help avoid non-global minima. If the user would prefer a more deterministic calculation then set

  ```
  randomise = 'false';
  ```

  The default values for these parameters are

  ```
  parallelTransport = 'false';
  calcMode = 'multiprod';
  randomise = 'true';
  ```

- Finally the parameters governing the precise inner-workings of the spread-reducing algorithms themselves can be altered. Specifically, (i) `iterIso`, (ii) `iterDis` and (iii) `iterComp` determined the number of iteration steps used during (i) the steepest-descent minimization of the intra-band spread for a single band [2, 1], as used in stage 1 and 2, (ii) each use of the disentangling procedure to reduce the inter-band spread [3, 1], as used in stage 2, and (iii) the steepest-descent minimization of the total spread for a single band [2, 1], as used in stage 3. The defaults are

  ```
  iterIso = 1000;
  iterDis = 1000;
  iterComp = 1000;
  ```

  During iterative procedure (ii) `alphaDis` controls the mixing of solutions from the current and previous stage of the iteration [3, 4], where the value `1` (`0`) corresponds to the current (previous) stage only. Reduce from `1` if there are stability problems.

  Similarly, `epsilon` controls the step size of the steepest-descent algorithm [2, 4] used in iterative procedures (i) and (iii). The defaults are

  ```
  epsilon = 1;
  alphaDis = 1;
  ```

3.2 Displayed outputs

The software lets the user keep track of the status of a calculation by displaying outputs in the MATLAB command window. As well as letting the user know how far into the calculation the software is, the main purpose for these outputs is that the user can check that iterative minimization procedures are converging sensibly. Here we discuss some of what is displayed, referring to the 5 stages of the calculation described in section 3.1.6.

1. The first stage begins with a band structure calculation which finds the Bloch states and energies at each point in the **k**-space mesh. The software outputs, e.g.,

   ```
   q = 31/221
   ```

   to indicate that the Bloch states and energies for 31 out of a total 221 **k**-points have been computed.

   The software then goes on to change the phases of these Bloch states so as to iteratively minimize the spread of their Fourier transform (ordinary Wannier states), one band at a time. An output

   ```
   Iteration = 400, omegaD = 4.749e-3
   ```

indicates that after 400 iterations, the spread of the ordinary Wannier states for the current band is $4.749 \times 10^{-3}\lambda$.

If the user asked for the band structure data to be loaded from a file, but no file was found, then the user will be alerted at the beginning of this stage.

2. The software allows mixing between groups of bands specified in `groups`. For each group it (if not disallowed by the user) starts by using the Souza et al. algorithm to perform a series of gauge transformations that optimally disentangles one band at a time using an iterative procedure, where optimally means the gauge-independent part of the spread for that band is minimized [3, 1]. An output

   ```
   n = 1/2: iteration 40, omegaI = 1.22995
   ```

   means that the software is extracting the first band of two for this group, and after 40 iterations the gauge-independent part to the spread for the band being disentangled is 1.22995λ.

   The software then repeats what it did at the end of stage 1 with similar displayed outputs.

   If the user asked for the starting gauge to be loaded from a file, but no file was found, then the user will be alerted at the beginning of this stage.

3. The software then applies the steepest-descent algorithm to minimize the total spread of each group of bands specified in `groups`. An output

   ```
   Iteration = 90, omegaOD = 0.007365, omegaD = 0.001993
   ```

   specifies that for the current group, after 90 iterations, the inter-band contribution to the spread is 0.007365λ and the intra-band contribution is 0.001993λ.

4. For this stage, the user is informed when the calculations have been performed.

5. The potential (Figure 1) and the n-th maximally localized generalized Wannier functions (Figure $n + 1$) are plotted. The resolution and range of the plots are determined by the inputs discussed in section 3.1.5, while the units of the Wannier functions are $\lambda^{-D/2}$ where D is the dimension.

3.3 Saved outputs

The outputs are in formats such that the user can post-process and present the information in anyway they wish. We now describe these formats.

3.3.1 Band structure

The band structure is saved in the file

`Wannier_data/Bloch_data/UniqueName.mat`

where `'UniqueName'` is the string the user assigned to `filename`.

The **k**-points for which the Bloch states and energies are calculated are stored in `bloch.Q.Mesh` which is a matrix of size $D \times N^D$. Here N is the number of points along each direction of the reciprocal lattice chosen by the user through input `N`. Each column of `bloch.Q.Mesh` is the vector in the Cartesian basis that specifies a **k**-point in units of $1/\lambda$. For example

```
bloch.Q.Mesh(:,85) = pi*[-2; -4];
```

tells us that **k**-point 85 is $\mathbf{k} = (-2\hat{\mathbf{x}} - 4\hat{\mathbf{y}})\,\pi/\lambda$.

The non-discarded reciprocal lattice vectors **G** are stored in the matrices `recip.Gk` and `recip.Gfrac`. Both are $D \times M$, where M is the number of non-discarded reciprocal lattice vectors. Each column of `recip.Gk` and `recip.Gfrac` gives the vector that specifies a non-discarded reciprocal lattice vector **G**. For `recip.Gk` this vector uses Cartesian coordinates, as was the case for `bloch.Q.Mesh`. Alternatively and redundantly `recip.Gfrac` uses the primitive reciprocal lattice vectors $\mathbf{b}_i$, as basis states. So

```
recip.Gfrac(:,31) = [5; 7];
```

tells us that non-discarded reciprocal lattice vector 31 is $\mathbf{G} = 5\mathbf{b}_1 + 7\mathbf{b}_2$.

The Bloch energies are stored in the $J \times N^D$ matrix `bloch.Energy`, where the first index labels the band (so row m corresponds to band m, as will always be the case) and the second index labels **k**. The order of the **k** vectors is the same as that in `bloch.Q.Mesh` (as will always be the case). The energies are in units of the recoil energy $E_R = h^2/2\mu\lambda^2$ (cf. section 3.1.1).

The Fourier coefficients $c_m^{(\mathbf{k},\mathbf{G})}$ describing the Bloch states are stored in the $M \times N^D \times J$ array `bloch.State`, where the first index labels **G** and the second index labels **k**, and the third index labels the band. The coefficients are dimensionless. To be precise, for a Bloch function

$$\psi_m^{(\mathbf{k})}(\mathbf{r}) = \mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\mathbf{r}} u_m^{(\mathbf{k})}(\mathbf{r}), \tag{8}$$

the coefficients are given by

$$c_m^{(\mathbf{k},\mathbf{G})} = \frac{1}{\sqrt{\Upsilon}} \int_{\mathrm{PC}} \mathrm{d}\mathbf{r} u(\mathbf{r}) \mathrm{e}^{-\mathrm{i}\mathbf{G}\cdot\mathbf{r}}. \tag{9}$$

3.3.2 The spreads of the maximally localized ordinary Wannier states

The spreads of the maximally localized ordinary Wannier states are saved in the file

`Wannier_data/Isolated_data/UniqueName.mat`

where `'UniqueName'` is the string the user assigned to `filename`.

The row vectors `wannier90.OmegaI`, `wannier90.OmegaD` and their vector sum `wannier90.Omega` give the phase invariant ($\Omega^n_{I,D}$), phase dependent (Ω^n_D) and total (Ω^n) spreads of each maximally localized ordinary Wannier state, in units of λ (see [1] for notation).

3.3.3 Gauge and spreads for maximally localized generalized Wannier states

The gauge and spreads of the maximally localized generalized Wannier states are saved in the file

```
Wannier_data/Isolated_data/UniqueName.mat
```

where `'UniqueName'` is the string the user assigned to `filename`.

The $J \times J \times N^D$ array `wannier90.U` stores the gauge matrices $U^{(\mathrm{k})}$ (see [2, 1]). Specifically, the first (second) index of `wannier90.U` indexes the columns (rows) of $U^{(\mathrm{k})}$.

The gauge here is in relation to the Bloch states defined by `bloch.State`.

The $J \times J \times N^D \times B$ array `wannier90.Mmn` stores the overlaps $M^{(\mathbf{k},\mathbf{b})}_{mn}$ (see [2, 1]). Specifically, the first (second) index of `wannier90.Mmn` indexes the columns (rows) of matrix $M^{(\mathbf{k},\mathbf{b})}$. The third index labels $\mathbf{k}$ while the last index labels the neighbour $\mathbf{b}$ [2, 1], of which there are a number B.

The row vectors `wannier90.OmegaD` and `wannier90.Omega` give the gauge dependent (Ω^n_D) and total (Ω^n) spreads of each maximally localized generalized Wannier state, in units of λ. The row vector `wannier90.GroupOmegaOD` gives the contribution to the inter-band spread (Ω_{OD}) from each group of bands specified in `groups`. The row vector `wannier90.GroupOmegaI` gives the contribution to the gauge independent spread (Ω_I) from each group of bands.

3.3.4 Wannier states and Hubbard parameters

The gauge and spreads of the maximally localized generalized Wannier states are saved in the file

```
Wannier_data/Many_body_data/UniqueName.mat
```

where `'UniqueName'` is the string the user assigned to `filename`.

The $\mathbf{r}$-points for which the values of the Wannier functions are stored is specified in `manyBody.SCell.Mesh` which is a matrix of size $D \times P$. Here P is the number of $\mathbf{r}$-points, which is determined from the users choice of `mbDensity` and `mbCells` (cf. section 3.1.3). For example, if the system is two-dimensional, then the choice

```
mbDensity = [10 10];
mbCells = [-5 5; -5 5];
```

leads to $P = (10 \times 11 + 1)^2 = 12321$ points. As `bloch.Q.Mesh` did for **k**-points, each column of `manyBody.SCell.Mesh` stores the Cartesian coordinates of an **r**-point in units of λ.

The matrix `manyBody.Sites` is equivalent to the user-defined `mbSites`, which specifies a set of S direct lattice vectors $\mathbf{R}$ (cf. section 3.1.3).

The $P \times J \times S$ tensor `manyBody.W` gives the values $w^n_{\mathbf{R}}(\mathbf{r})$ at each of these **r**-points for each n and $\mathbf{R}$ in units of $\lambda^{-D/2}$. The first, second and third indices correspond to the **r**-points for each n and $\mathbf{R}$ respectively.

The $J \times J \times S$ tensor `manyBody.J` gives the hopping parameters $t^{mn}_{\mathbf{R0}}/E_R$ for all m, n and $\mathbf{R}$ (first, second and third index respectively).

Similarly, the tensors `Ujjnn UOjnn Ujjmn UOjmn UjOmn UjOjO` in `manyBody` store the interaction parameters $U^{mmnn}_{\mathbf{00R}}\tilde{g}/gE_R$, $U^{mmnn}_{\mathbf{000}}\tilde{g}/gE_R$, $U^{mmmn}_{\mathbf{00R}}\tilde{g}/gE_R$, $U^{mmmn}_{\mathbf{000}}\tilde{g}/gE_R$, $U^{mmmn}_{\mathbf{00R0}}\tilde{g}/gE_R$ and $U^{mmnn}_{\mathbf{R0R0}}\tilde{g}/gE_R$ for all m, n and $\mathbf{R}$ (first, second and third index respectively). Here $\tilde{g} = E_R\lambda^D$

4 Detecting non-global minima

Despite designing an initialization procedure for the calculation of the maximally localized generalized Wannier stats precisely to avoid getting stuck in non-global minima, occasionally the steepest-descent minimization algorithm still finds a non-global minimum.

If a non-global minimum is obtained, this should be clear to user from the fact that the Wannier states are non-real, and also from the unusual (non-converging) behaviour of the spreads (displayed in the MATLAB command window as the calculation is running).

Running the calculation again with `recalcComposite`, `disentangle` and `randomise` all set to `'true'` should find the global minimum.

5 More information needed, or any suggestions?

Please let us know via email: `mlgws-admins@hebron.cse.rl.ac.uk`. Thank you.

References

[1] R. Walters, G. Cotugno, T. H. Johnson, S. R. Clark, and D. Jaksch, *Ab initio derivation of Hubbard models for cold atoms in optical lattices*, preprint arXiv:1303.2213 (2013).

[2] N. Marzari, and D. Vanderbilt, Phys. Rev. B **56**, 12847 (1997).

[3] I. Souza, N. Marzari, and D. Vanderbilt, Phys. Rev. B **65**, 035109 (2001).

[4] R. Walters, Ultracold Atoms in Optical Lattices: Simulating Quantum Spin Systems (DPhil. Thesis, University of Oxford, 2013); available here.