LEO2D Manual

Stephen Carr

26th August 2019

Contents

1 Introduction

Locality for electronic obserables in 2D materials (LEO2D) is a code designed to perform parallelized tight-binding calculations in models of twisted 2D heterostructures.

LEO2D is explicitly parallelized with the standard Message Passing Interface (MPI) library, and cannot be run in serial. The MPI communication is designed in a master-worker queuing system, where a master MPI rank (rank 0) sends out and receives jobs from a series of workers (rank > 0). The master determines the heterostructure's sparsity pattern, organizes a queue of calculations, and saves results. Each worker performs a specific realization of the heterostructure hamiltonian, performs relevant electronic structure computations, and then returns results to the master. For example, when calculating local density of states in finite-systems, the workers handle different local environments. For periodic systems, the workers perform band-structure sampling by creating the hamiltonian at different k-points. There is also implicit multi-threading included in the Matrix Kernel Library (MKL), whose routines make up the majority of the work in LEO2D's electronic-structure calculations.

2 Getting Started

It is highly recommended that the Intel Matrix Kernal Library (MKL) be used to improve performance on intel hardware. The MKL includes optimized BLAS and LAPACK routines, which take the majority of wall-time for large tight-binding calculations.

2.1 Installation

LEO2D is compiled with the cmake utility. Make a copy of the `CMakeLists_template.txt` named `CMakeLists.txt` in the same directory.

There are a number of options which can be set to `ON` or `OFF` in this file for controlling installation settings:

- **LINK_FFTW**: Some older compilers do not come with FFTW included by default. This will try to find a FFTW library on the local environment and link it the LEO2D.
- **MKL_INTEL**: Assumes the MKL library is included in the compiler (works with most modern icc installations).
- **MKL_GNU**: For use with non-intel compilers (e.g. gcc), will look for the MKL library in the local environment.
- **MKL_OFF**: Will instead use the open-source C Library Eigen for matrix arithmetic. Eigen has fairly good matrix-vector multiplication, but its diagonalization is much slower than the MKL LAPACK routines.

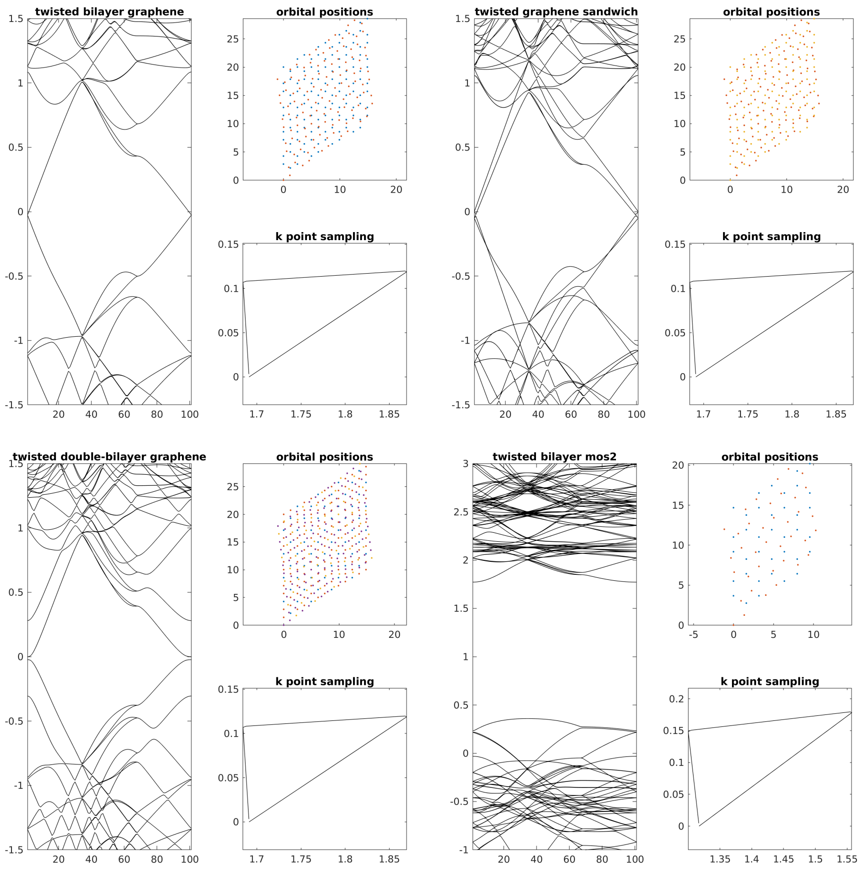

Figure 1: Supercell band-structure examples.

When finished, the executable `LEO2D` should be available in the `app/` directory, and a partner library `libleo2d.a` will be in the `lib/` directory.

2.2 Running Examples

Once compiled, try running LEO2D on any of examples provided in the `examples` directory. For example, to run a supercell band-structure calculation for twisted bilayer graphene, go to the `example/tblg_sc` directory and call LEO2D like so:

```
mpirun -n 2 ../../app/LEO2D tblg_sc_hstruct.in
```

Here we are calling a 2-core instance of LEO2D using the settings in `tblg_sc_hstruct.in`. The same can be done for any of the other examples. Note that the `examples/old/` directory is no longer maintained.

There are four twisted supercell examples: bilayer graphene (`tblg_sc`), graphene sandwich (`tswg_sc`), double-bilayer graphene (`tdbg_sc`), and bilayer MoS_2

(`tbmos2_sc`). There is also and example of a density of states calculation for a twisted finite flake (`tblg_flake`) and for a bandstructure comparison for monolayer graphene with and without a uniform strain (`mlg_strain`).

2.3 Common Errors

3 Parameters

4 Code Overview