

Besides all the approximations we have to do, no computer can solve an infinite continuous problem. We have to discretize our equations somehow. Octopus uses a grid in real space to solve the Kohn-Sham equations. That is, functions are represented by their value over a set of points in real space. Normally the grid is equally spaced, but also non-uniform grids can be used. The shape of the simulation region may also be tuned to suit the geometric configuration of the system.

Contents

- 1 Grid
 - ◆ 1.1 Double grid (experimental)
- 2 Box
 - ◆ 2.1 Zero boundary conditions

Grid

In Octopus functions are represented by a set of values that correspond to the value of the function over a set of points in real space. By default these points are distributed in a uniform grid, which means that the distance between points is a constant for each direction. It is possible to have grids that are not uniform, but as this is a bit more complex we will discuss it later.

In this scheme, the separation between points, or spacing, is a critical value. When it becomes large the representation of functions gets worse and when it becomes small the number of points increases, increasing memory use and calculation time. This value is equivalent to the energy cutoff used by plane-wave representations.

In Octopus you can choose the Spacing of your simulation by the Spacing variable. If you set this as a single value it will give the same spacing for each directions (for example `Spacing=0.3`). If you want to have different spacings for each coordinate you can specify Spacing as a block of three real values.

If you are working with the default pseudopotential species of Octopus they come with a recommended Spacing value and you don't need to give it in the input file. Normally this default values are around 0.4 [b] (~0.2 [Å]), but you may need smaller spacing in some cases. Do not rely on these default values for production runs.

Double grid (experimental)

The double-grid technique is a method to increase the precision of the representation of the pseudopotentials in the grid that has been recently integrated in Octopus (not available in Octopus 2.1 and previous versions). To activate this technique, set the variable DoubleGrid to `yes`. The use of a double grid increases the cost of the transfer of the potential to the grid, but as in most cases this is done only a few times per run, the overhead to the total computation time is negligible. The only exception is when the atoms are displaced while doing a time-dependent simulation, where the double grid can severely increase the computation time.

Box

We also have to select a finite domain of the real space to run our simulation, which is known as the simulation box. Octopus can use several kinds of shapes of box. This is controlled by the variable BoxShape. Besides standard shapes Octopus can take shapes given by a user-defined function or even by an image.

The way to give the size of the simulation box changes for each shape, but for most of them it is given by the variable Radius.

Zero boundary conditions

By default Octopus assumes zero boundary conditions, that is, wavefunctions and density are zero over the boundary of the domain. This is the natural boundary condition when working with finite systems.

In this case choosing an adequate box size is very important: if the box is too small the wavefunctions will be forced to go to zero unnaturally, but if the box is too large, a larger number of points is needed, increasing calculation time and memory requirements.

Previous [Manual:Hamiltonian](#) - Next [Manual:Output](#)

Back to [Manual](#)