Multigrid

A highly versatile integration grid class for C++

Tobias Stollenwerk

April 23, 2012

Contents

1 Introduction

The multigrid class is a highly versatile integration grid class for `C++`. It features the integration of multiple grid regions in order to resolve sharp peaks or steps in the integrand function. Hereby one can choose between equidistant, tangential and logarithmically dense grid regions. Although the number of grid regions is not bounded, there is an inverse mapping useful for fast interpolation purposes. Intersection and overlap of different grid region is possible will be dealt with by favouring the better resolve grid region.

The original purpose of the multigrid class was to resolve a lot of sharp Lorentz-like peaks in a self-consistent calculation on strongly correlated electron systems.

2 Quick Start Guide

The purpose of this section is to give you a brief introduction of how the multigrid is used in practice. It may suffice for the most applications.

2.1 Installation

At the moment the only possiblility to use the multigrid class is to directly incorporate the source file into your program. It is planned to create a library version. At first download the latest version of the multigrid class from

```
https://github.com/tstollenw/multigrid
```

unpack it and put the files

```
multigrid.cpp
multigrid.h
grid.cpp
grid.h
mesh.cpp
mesh.h
```

into your program directory. Include and link the files in the usual way. There is an example program `main.cpp` and the corresponding makefile `makefile` which will create an excecutable `example.out`.

2.2 First Steps

The numerical calculation of an integral over some function $f(\omega)$

$$I = \int d\omega f(\omega)$$

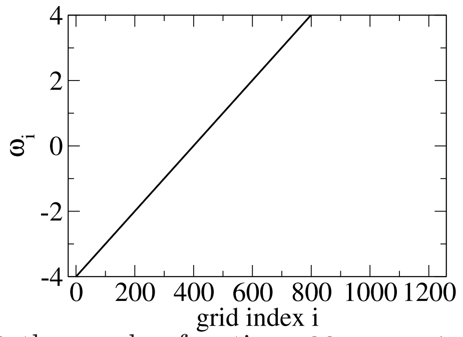
can be written as

$$I = \sum_{i=0}^{M} d\omega_i f(\omega_i)$$

where the ω_i is the discrete integration grid and $d\omega_i$ are the corresponding weights. The integration grid is a mapping from a discrete index $i \in \{0, M\}$ to a contineous variable ω_i. The multigrid class provides both the integration grid and its weights (which are calculated by the trapez rule) as well as an inverse mapping (back from any ω to the nearest index i).

In the following we will show the basic functionallity of the multigrid class by some simple examples. All these examples are more or less all part of the example program `main.cpp`. At first we will create a simple equidistant grid with from -4 to 4 with a resolution of 0.01 by

```
multigrid mgrid;
mgrid.add_gr_equi(-4, 4, 0.01);
mgrid.create();
```



After the declaration of the multigrid named `mgrid`, the member function `add_gr_equi` adds an equidistant grid region to the grid. The grid is created by invoking the `create` member function and it is accessed over its member variables (see table 2.2).

Name	Type	Description
`M`	integer	Number of grid points
`omega`	STL-Vector	Grid
`domega`	STL-Vector	Weights of the grid
`omega_min`	double	Minimum grid value (equal to `omega[0]`)
`omega_max`	double	Maximum grid value (equal to `omega[M]`)
`inverse`	returns integer	Inverse mapping: $\omega \to i$

Table 2.1: Most important members of the multigrid

The calculation of the above integral from -4 to 4 is then done by

```
double I=0;
for (int i=0; i<=mgrid.M; j++)
{
        I += f(mgrid.omega[i]) * mgrid.domega[i];
}
```

2.3 Logarithmically Dense Grid Regions

To resolve steps or very sharp peaks in the integrand function one needs a lot of integration grid points at specific regions. The multigrid class provides a tool to solve such problems: logarithmically dense grid regions (LGR). An LGR is determined by four variables, i.e. the center of the grid region ω_0 which corresponds for example to the position of a peak in the integrand function, the half width of the grid region ω_1, the maximal resolution $d\omega_{max}$ at the center of the grid region and the minimal resolution $d\omega_{min}$ at the edges of the grid region (see figure 2.3).

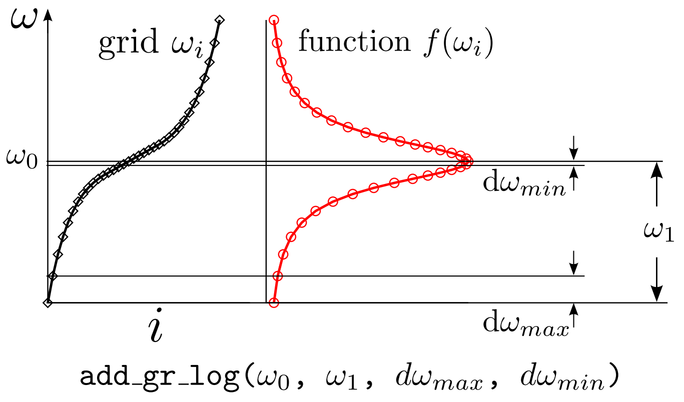
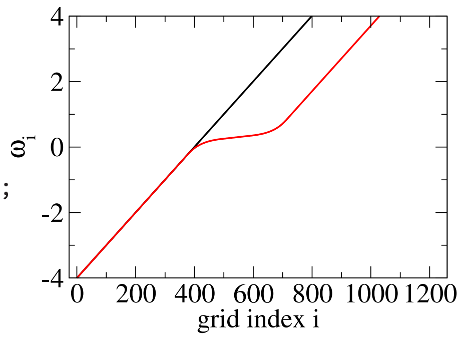


Figure 2.1: Logarithmically dense grid region to resolve a peaked integrand function

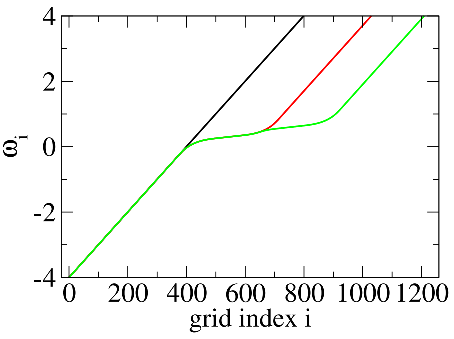
It is created by the function add_gr_log(ω_0, ω_1, $d\omega_{max}$, $d\omega_{min}$). For example the following code adds a LGR on top of the equidistant gridregion from above. Note that since the equidistant grid region was added first, it determines the outer boundaries of the whole grid (here from -4 to 4). The first added grid region is therefore a special one and is called the basic grid region.

```
multigrid mgrid;
mgrid.add_gr_equi(-4, 4, 0.01);
mgrid.add_gr_log(0.3,0.5,0.001,0.01);
mgrid.create();
```



The strength of the multigrid is that one can add now more and more grid regions on top of each other. The create function will take care of calculating intersection points between the grid regions by favouring the better resolved grid region. In the following example the there are two intersecting LGR on top of an equidistant grid region.

```
multigrid mgrid;
mgrid.add_gr_equi(-4, 4, 0.01);
mgrid.add_gr_log(0.3,0.5,0.001,0.01);
mgrid.add_gr_log(0.6,0.5,0.001,0.01);
mgrid.create();
```



These are only the basic features of the multigrid class. There is an algorithm which decides where to cut grid regions if there is intersection or even skip a particular grid region in special cases. The decisive element is the grid resolution exactly at the center of a given grid region ω_0. This is called the peak point. Hereby it is possible to add hundreds of grid regions on top of each other without losing the resolution at every single peak point. In figure 2.3 there is an example for the necessity for multiple LGR in the integration grid. The integrand function has several sharp peaks which has to be resolved. Each peak is resolved by a LGR.

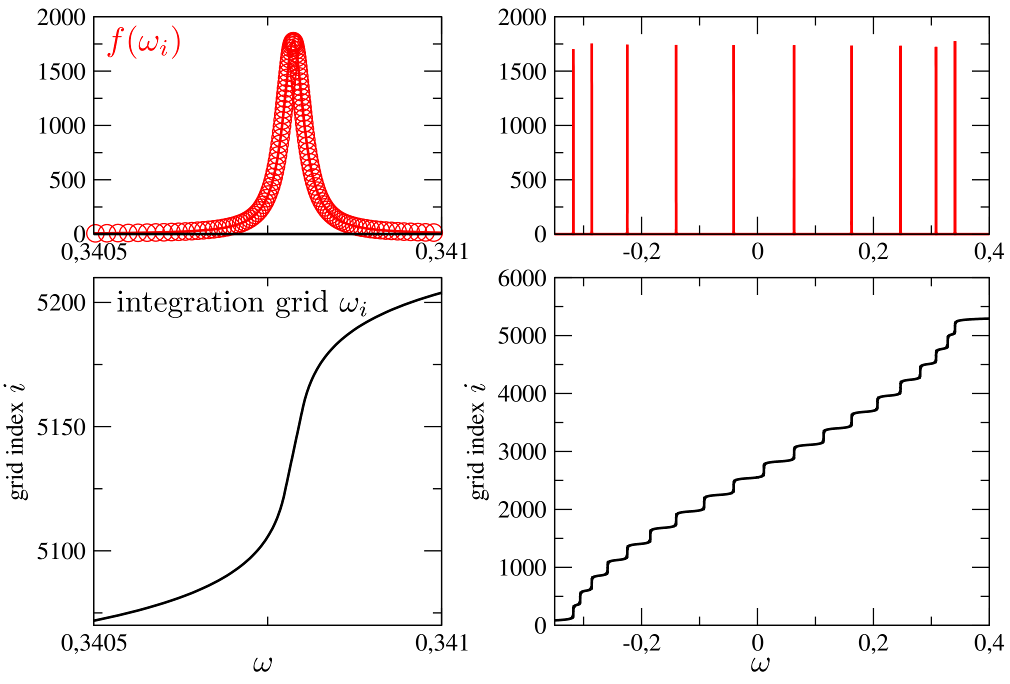


Figure 2.2: Multigrid with various logarithmically dense grid regions to resolve a multiple peaked integrand function