4.3.2 Multigrid method

A very efficient way of solving the discretized Poisson equation is the *multigrid method* which, together with the main grid, introduces a number of coarser grids, typically with two, four, etc. times the grid spacing in each direction.

Consider the linear system (4.61). Under certain conditions (which are typically met), it can be solved iteratively using

$$f_{k,l} = \frac{f_{k+1,l} + f_{k-1,l} + f_{k,l+1} + f_{k,l-1} - h^2 g_{kl}}{4} \,. \tag{4.71}$$

There are two variants of this iteration scheme. For *Jacobi iteration*, at a given stage the right hand sides are evaluated for all grid points and only then the values f_{kl} are updated. Convergence is two times faster with *Gauss–Seidel* iteration, where Eq. (4.71) is evaluated for each point in sequence, using the latest updated values on the right hand side. Gauss–Seidel iteration also requires only half the memory of Jacobi iteration and has the important advantage of damping the highest wave number. This is the scheme we will consider here.

The iterative solution of Eq. (4.71) is quite slow. While the small scales ($\sim \delta x$) are quickly converging, it is the largest scales that take very long to reach their 'equilibrium' values due to the (local) iteration. For these large scales, however, one would not need the fine grid, and on a coarser grid they would converge much faster. Hence the idea of multigrid methods: Solve the problem iteratively on grids of different resolution by

1. *coarse-graining* (downsampling) using the *restriction matrix R*:

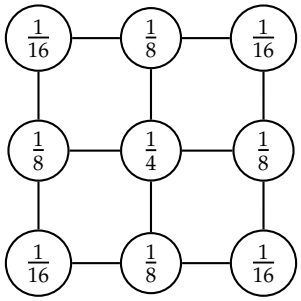
$$\mathbf{r}_{\text{coarse}} = \mathcal{R}\mathbf{r}_{\text{fine}} \,, \tag{4.72}$$

   and

2. *fine-graining* (refining = interpolation), using the *prolongation matrix P*:

$$\delta\mathbf{f}_{\text{fine}} = \mathcal{P}\delta\mathbf{f}_{\text{coarse}} \,. \tag{4.73}$$

A popular choice for the restriction matrix is represented by



This matrix acts as a *lowpass filter*: signals at the Nyquist frequency $f_{\text{Ny,fine}}$ of the fine grid are completely filtered out, while signals at $f_{\text{Ny,coarse}}$ are damped as little as possible.

This is crucial for the scheme to be efficient, as any contribution of $f_{\mathrm{Ny,fine}}$ would only be misinterpreted as a larger frequency on the coarser grid (*aliasing*).

The refinement is done using linear interpolation.

To see how the method works, consider the differential equation

$$\mathcal{L}f = g \ , \tag{4.74}$$

which, discretized at the grid spacing h, becomes

$$\mathcal{L}_h f_h = g_h \ . \tag{4.75}$$

If we have an approximate solution $\tilde{f_h}$, we can introduce the error

$$\delta f_h = f_h - \tilde{f_h} \ , \tag{4.76}$$

and find

$$-\mathcal{L}_h \delta f_h = \mathcal{L}_h \tilde{f_h} - \mathcal{L}_h f_h = \mathcal{L}_h \tilde{f_h} - g_h \ , \tag{4.77}$$

i.e.

$$-\mathcal{L}_h \delta f_h = r_h \ , \tag{4.78}$$

where $r_h \equiv \mathcal{L}_h \tilde{f_h} - g_h$ is the *residual*, which is a measure of how well our approximate solution $\tilde{f}$ solves the original problem.

Coarse-graining r_h to the coarser grid with spacing $H = 2h$,

$$r_H = \mathcal{R} r_h \ , \tag{4.79}$$

we arrive at

$$-\mathcal{L}_H \delta f_H = r_H \ , \tag{4.80}$$

which we solve (this is faster than on the finer grid) to obtain δf_H.

Then we fine-grain (interpolate) δf_H onto the finer grid,

$$\delta \tilde{f_h} = \mathcal{P} \delta f_H \ , \tag{4.81}$$

and calculate the new value of $\tilde{f_h}$ as

$$\tilde{f_h}^{\mathrm{new}} = \tilde{f_h} + \delta \tilde{f_h} \ . \tag{4.82}$$

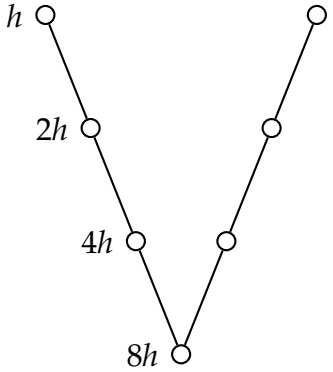
Formulated as a recipe, this two-grid method reads:

0. Start with a guess $\tilde{f_h}$ on the fine grid (e.g. choose 0).
1. Calculate the residual $r_h = \mathcal{L}_h \tilde{f_h} - g_h$, and coarse-grain it, $r_H = \mathcal{R} r_h$.
2. Solve $\mathcal{L}_H \, \delta f_H = -r_H$ on the coarser grid, fine-grain the correction, $\delta \tilde{f_h} = \mathcal{P} \delta f_H$, and add it to the initial guess. $\tilde{f_h}^{\mathrm{new}} = \tilde{f_h} + \delta \tilde{f_h}$.
3. Do one Gauss–Seidel iteration.
4. Continue with 1 until $\delta \tilde{f_h}$ is small enough.

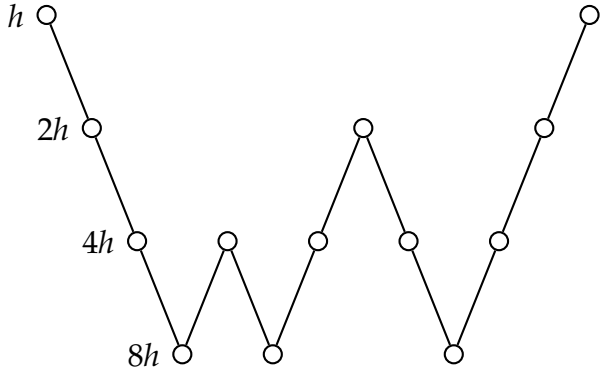
To turn this from a two- into a multi-grid method, we simply use another two-grid scheme for obtaining δf_H at stage 2, etc. The best way to code a multigrid method is obviously recursive.

Note 1: The numerical cost for the multigrid method is roughly $\propto N = N_x N_y$ and thus comparable to Fourier methods. However, multigrid methods can be used for equations with variable coefficients and also for nonlinear equations.

Note 2: The multigrid scheme thus described is referred to as 'V' cycle (the name should be evident from the scheme below). There are other popular cycles like the 'W' cycle.



The 'V' cycle.



A 'W' cycle (many other W cycles exist).

4.4 Parabolic problems

The heat conduction equation

$$\frac{\partial T}{\partial t} = \chi \frac{\partial^2 T}{\partial x^2} \tag{4.83}$$

is the prototype of a parabolic differential equation. To solve it numerically, we need to discretize in space and time:

$$T_k^l \equiv T(x_k, t_l) \; . \tag{4.84}$$

Starting from an initial condition T_k^l, which we assume to be known everywhere, we need to construct the solution at the next time, T_k^{l+1}.

4.4.1 Explicit scheme

A simple scheme is obtained from the discretization

$$\frac{T_k^{l+1} - T_k^l}{\delta t} = \chi \; \frac{T_{k-1}^l - 2T_k^l + T_{k+1}^l}{\delta x^2} \; . \tag{4.85}$$

We can explicitly solve for the unknown T_k^{l+1},

$$T_k^{l+1} \; = \; T_k^l + \frac{\chi \, \delta t}{\delta x^2} \left(T_{k-1}^l - 2T_k^l + T_{k+1}^l \right) \; = \; C T_{k-1}^l + (1-2C) T_k^l + C T_{k+1}^l \; , \tag{4.86}$$