The numerical solution of Poisson's equation using multigrid methods

I. Poisson's equation

The electrostatic potential once the density is given is:

\[ \phi(r) = \int \frac{\rho(r')}{|r - r'|} d^3r' \]  \hspace{1cm} (1.1)

However, it is not easy to perform this numerically. As you can see, for each point \( r \), one has to do an integral on \( r' \). Is there a better way? One approach is to represent the potential as a solution to a differential equation. Indeed, we will show that the required equation is Poisson's equation:

\[ \nabla^2 \phi(r) = -4\pi \rho(r) \]  \hspace{1cm} (1.2)

Where \( \phi(r) \) is the electrostatic potential and \( \rho(r) \) is the charge density. This equation must be solved over all space. It is a way to obtain the potential due to a distribution \( \rho(r) \) of charges in space. If the potential is assumed to decay to zero when far from the origin (\( \phi(r) \to 0, \ r \to \infty \)) then the solution of this equation is just Eq. (1.1). To see this, let us apply the Laplace operator to this equation:

\[ \nabla^2 \phi(r) = \int \rho(r')\nabla^2 \frac{1}{|r - r'|} d^3r' \]  \hspace{1cm} (1.3)

We will show now that the “function” \( \nabla^2 \frac{1}{|x|} \) is very special: it is zero at all \( x \neq 0 \) and its integral over space is finite. Thus it is proportional to a delta-function. Let’s prove it. Assume first that \( x \neq 0 \). Then:
\[ \frac{\partial^2 1}{\partial x^2 |x|} = \frac{\partial^2}{\partial x^2} \frac{1}{\sqrt{x^2 + y^2 + z^2}} = \frac{\partial}{\partial x} \left( -\frac{x}{(x^2 + y^2 + z^2)^{3/2}} \right) \]

\[ = \left( -\frac{1}{(x^2 + y^2 + z^2)^{3/2}} + 3 \frac{x^2}{(x^2 + y^2 + z^2)^{5/2}} \right) \]

\[ = -\frac{1}{(x^2 + y^2 + z^2)^{3/2}} \left( 1 - 3 \frac{x^2}{(x^2 + y^2 + z^2)} \right) \]

with obvious terms for \( \frac{\partial^2}{\partial y^2} \) and \( \frac{\partial^2}{\partial z^2} \):

\[ \frac{\partial^2 1}{\partial y^2 |x|} = -\frac{1}{(x^2 + y^2 + z^2)^{3/2}} \left( 1 - 3 \frac{y^2}{(x^2 + y^2 + z^2)} \right) \]

\[ \frac{\partial^2 1}{\partial z^2 |x|} = -\frac{1}{(x^2 + y^2 + z^2)^{3/2}} \left( 1 - 3 \frac{z^2}{(x^2 + y^2 + z^2)} \right) \]

Therefore, we have, for the sum over the x, y, and z derivatives, i.e. for the Laplacian:

\[ \nabla^2 \frac{1}{|x|} = -\frac{1}{(x^2 + y^2 + z^2)^{3/2}} (3 - 3) = 0 \quad x \neq 0 \quad (1.4) \]

This shows the function is zero everywhere except at the origin where it is divergent.

Next, we show that the integral of this divergent function gives a finite value.

We first make an infinitesimal change in the denominator, rendering the integral finite:

\[ \int \nabla^2 \frac{1}{|x|} d^3x = \lim_{\eta \to 0} \int \nabla^2 \frac{1}{\sqrt{|x|^2 + \eta^2}} d^3x \quad (1.5) \]

Next, notice that the integrand depends only on \( r \) and not on the angles \( \theta \) and \( \phi \) so that these angles can be integrated to \( 4\pi \) and we are left with the 1-dimensional integral. In radial coordinates the Laplacian is replaced by \( \nabla^2 f(r) \rightarrow \frac{1}{r} \left( rf'(r) \right)' \), so:

\[ \int \nabla^2 \frac{1}{|x|} d^3x = \lim_{\eta \to 0} \int_0^\infty \frac{1}{r} \left( \frac{1}{\sqrt{r^2 + \eta^2}} \right)' 4\pi r^2 dr \quad (1.6) \]

Next, we evaluate the second derivative:
\[
\int \nabla^2 \frac{1}{|x|} \, d^3x = \lim_{\eta \to 0} \int_0^\infty \frac{1}{r} \left( \frac{1}{\sqrt{r^2 + \eta^2}} - \frac{r^2}{\sqrt{r^2 + \eta^2}^3} \right) 4\pi r^2 \, dr \\
= \lim_{\eta \to 0} \int_0^\infty \left( \frac{-3}{\sqrt{r^2 + \eta^2}^3} + \frac{3r^2}{\sqrt{r^2 + \eta^2}^5} \right) 4\pi r^2 \, dr
\] (1.7)

Pulling out of the parenthesis the factor \(\frac{1}{\sqrt{r^2 + \eta^2}}\) we obtain after some elementary manipulation:

\[
\int \nabla^2 \frac{1}{|x|} \, d^3x = \lim_{\eta \to 0} \int_0^\infty (-3\eta^2) \frac{4\pi r^2}{\sqrt{r^2 + \eta^2}^5} \, dr
\] (1.8)

In this last integral, it is amazing that we can scale \(\eta\) out by a mere change of variable \(\xi = \frac{r}{\eta}\) This gives:

\[
\int \nabla^2 \frac{1}{|x|} \, d^3x = -12\pi \lim_{\eta \to 0} \int_0^\infty \frac{\xi^2}{\sqrt{\xi^2 + 1}} \, d\xi
\] (1.9)

It is a kind of a miracle that the integral is over the variable \(\xi\) is independent of \(\eta\) so we can just forget the limit process. This technique, where a variable is entered into a seemingly divergent integral only to be eliminated when a convergent form is obtained is called “renormalization technique”. The renormalization thus eliminate completely:

\[
\int \nabla^2 \frac{1}{|x|} \, d^3x = -12\pi \int_0^\infty \frac{\xi^2}{\sqrt{\xi^2 + 1}} \, d\xi = -4\pi
\] (1.10)

Here we used the fact that:

\[
\int_0^\infty \frac{\xi^2}{\sqrt{\xi^2 + 1}} \, d\xi = \int_0^{\pi/2} \frac{\tan^2\alpha}{\sqrt{\tan^2\alpha + 1}} \, d\tan\alpha = \int_0^{\pi/2} \tan^2\alpha \cos^5\alpha \frac{d\alpha}{\cos^2\alpha}
\]

\[
= \int_0^{\pi/2} \sin^2\alpha \cos\alpha \, d\alpha = \int_0^{\pi/2} \sin^2\alpha \, d\sin\alpha = \left( \frac{\sin^3\alpha}{3} \right)_0^{\pi/2} = \frac{1}{3}
\]

Thus:
$$\int \nabla^2 \frac{1}{|\mathbf{x}|} d^3x = -4\pi$$  \hspace{1cm} (1.11)

From the fact that $\nabla^2 \frac{1}{|\mathbf{x}|} = 0$ for all $\mathbf{x} \neq 0$ and its integral is $4\pi$ we conclude:

$$\nabla^2 \frac{1}{|\mathbf{x}|} = -4\pi \delta(\mathbf{x})$$  \hspace{1cm} (1.12)

We see that the differential equation is equivalent to the integral. However, the integral can be pretty difficult to perform numerically while the differential equation is perhaps more efficiently handled. In addition, with a differential equation one can generalize the solution by imposing boundary conditions. For example, when there are metal objects around the potential is constant inside the areas where the metals lie.

So, we will learn how to numerically solve Poisson’s equation as a means of computing the Hartree potential in DFT. This can be an efficient alternative to doing 2-electron integrals in quantum chemical codes. Indeed, when $\rho(\mathbf{r}) = n(\mathbf{r})$ the electrostatic potential $\phi(\mathbf{r})$ is nothing else than the Hartree potential $v_H(\mathbf{r})$. Note, that once the Hartree potential is obtained the Hartree energy $E_H = \frac{1}{2} \int \int \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d^3r' d^3r$ is simply a single integral:

$$E_H = \frac{1}{2} \int n(\mathbf{r})v_H(\mathbf{r})d^3r$$  \hspace{1cm} (1.13)

II. Finite difference formulae

In this section we discuss a technique for producing finite difference formulae to approximate the operation of derivatives on functions on a grid.

Suppose we have a grid $x_n = nh$. Any function $f(x)$ is represented on the grid using its values on the grid points: $f_n = f(x_n)$. How do we represent derivatives? Since derivatives are linear operators and any function can be written as a linear superposition of plane waves, a very general technique is to consider just a general single plane wave on the grid: $f(x) = e^{ikx}$. 
First, let us state that not all plane waves can be reasonably represented on a grid. For example, the wave \( \sin kx = \frac{1}{2i} (e^{ikx} - e^{-ikx}) \) with \( k = \frac{\pi}{h} \) will be represented on the grid by: \( f_n = \sin \frac{\pi}{h} nh = 0 \). So it is indistinguishable from the function \( f(x) = 0 \). It is also indistinguishable from \( f(x) = e^{ikx} \) with \( k = \frac{2\pi}{h} \) etc. Thus, the fact that we have a finite sampling restricts our description in a fundamental way: the highest frequency representable on a grid is:

\[
k_{\text{max}} = \frac{\pi}{h} \quad k_{\text{min}} = -\frac{\pi}{h}
\]  

(2.1)

Functions with higher frequencies are “aliased” to lower ones. The error incurred by discretization is called the “discretization error”. We call frequencies with \(|k| > \frac{k_{\text{max}}}{2}\) “high frequency” and those with \(|k| < \frac{k_{\text{max}}}{2}\) low frequency:

- low frequency: \(|k|h < \frac{\pi}{2}\)
- high frequency: \(|k|h > \frac{\pi}{2}\)

(2.2)

Of course, high and low are relative superlatives, they are related to the grid spacing \( h \). Clearly, the lower the frequency the better it is described by the grid, since there are many sampling points in each wave length. High frequencies sampled by few points per wave length well.

Now, let us come back to the question of derivatives. We were considering the function \( f(x) = e^{ikx} \). We want to find a local approximation for the derivative \( f'(x_n) = ike^{iknh} \) in terms of \( f_n \) values at consecutive (nearby) grid-points. For example, with 3 consecutive values:

\[
f'(x_n) \approx a_{-1}f_{n-1} + a_0f_n + a_1f_{n+1}
\]

(2.3)

For our plane wave we have:

\[
ike^{iknh} \approx (a_{-1}e^{-ikh} + a_0 + a_1e^{ikh})e^{iknh}
\]

(2.4)

Thus:
\[ ik \approx a_{-1} e^{-ikh} + a_0 + a_1 e^{ikh} \]  

(2.5)

In order to solve this, denote: \( \xi = e^{ikh} \) then \( ikh = \log \xi \) and obtain the polynomial approximation:

\[ \frac{\xi \log \xi}{h} \approx a_{-1} + a_0 \xi + a_1 \xi^2 \]  

(2.6)

Now, how do we find the “best” coefficients \( a_i \)? We want the formula to best reproduce the derivative for small frequencies, since then we know that the original function is well represented on the grid (there is no point at demanding high accuracy of the derivative for functions which are themselves not well represented). So, we want high accuracy when \( k \) is close to zero (low frequency). Thus we are looking at \( \xi \) close to 1. So we Taylor-expand \( g(\xi) = \xi \log \xi \) around \( \xi = 1 \) and use the expansion coefficients for determining \( a_i \). We have: \( g'(\xi) = \log \xi + 1 \), \( g''(\xi) = \frac{1}{\xi} \) so: \( g(1) = 0 \), \( g'(1) = 1 \), \( g''(1) = 1 \) thus, we have:

\[ \xi \log \xi = (\xi - 1) + \frac{1}{2} (\xi - 1)^2 + O(\xi - 1)^3 = \frac{1}{2} (\xi - 1)(\xi + 1) + O(\xi - 1)^3 \]

\[ = \frac{1}{2} (\xi^2 - 1) + O(\xi - 1)^3 \]

keeping only terms to second order. Thus:

\[ \frac{1}{2h} [\xi^2 - 1 + O(h^3)] = a_{-1} + a_0 \xi + a_1 \xi^2 \]  

(2.7)

We now immediately read off the coefficients:

\[ a_{-1} = -\frac{1}{2h} \quad a_0 = 0 \quad a_1 = \frac{1}{2h} \]  

(2.8)

The finite difference formula is:

\[ f_k'(x_n) \approx \frac{f_{n+1} - f_{n-1}}{2h} + O(h^2) \]  

(2.9)

Let us go for higher order accuracy. We include more points:

\[ ik \approx a_{-2} e^{-2ikh} + a_{-1} e^{-ikh} + a_0 + a_1 e^{ikh} + a_2 e^{2ikh} \]  

(2.10)
Once again we want to get the best coefficients:

\[
\frac{\xi^2 \log \xi}{h} \approx a_{-2} + a_{-1} \xi^1 + a_0 \xi^2 + a_1 \xi^3 + a_2 \xi^4
\]  

(2.11)

So, we Taylor-expand \( g(\xi) = \xi^2 \log \xi \) around \( \xi = 1 \). We have: \( g'(\xi) = 2\xi \log \xi + \xi \), \( g''(\xi) = 2 \log \xi + 3 \), \( g'''(\xi) = \frac{2}{\xi} \) and \( g''''(\xi) = \frac{-2}{\xi^2} \). Thus: \( g'(1) = 1 \), \( g''(1) = 3 \), \( g'''(1) = 2 \), \( g''''(1) = -2 \).

\[
\frac{\xi^2 \log \xi}{h} = \frac{1}{h} \left( (\xi - 1) + \frac{3}{2} (\xi - 1)^2 + \frac{2}{6} (\xi - 1)^3 - \frac{2}{24} (\xi - 1)^4 \right)
\]

(2.12)

Thus:

\[
\begin{align*}
a_{-2} &= \frac{1}{12h} \\
a_{-1} &= -\frac{8}{12h} \\
a_0 &= 0 \\
a_1 &= \frac{8}{12h} \\
a_2 &= -\frac{1}{12h}
\end{align*}
\]

(2.13)

The derivative is (VWM):

\[
f'(x_n) \approx \frac{f(x_{n-2}) - 8f(x_{n-1}) + 8f(x_{n+1}) - f(x_{n+2})}{12h} + O(h^4)
\]

(2.14)

The same technique can be applied to the second derivative. We know that \( f''_k(x_n) = (ik)^2 f_k(x_n) \). Thus:

\[
(ik)^2 \approx a_{-1} e^{-ikh} + a_0 + a_1 e^{ikh}
\]

(2.15)

Put \( \xi = e^{ikh} \) and so: \( \log \xi = ikh \rightarrow (ik)^2 = \left( \frac{\log(\xi)}{h} \right)^2 \). Thus

\[
\frac{\xi [\log \xi]^2}{h^2} = a_{-1} + a_0 \xi + a_1 \xi^2
\]

(2.16)

We Taylor-expand:

\[
\xi [\log \xi]^2 = (\xi - 1)^2 + O(h^4)
\]

(2.17)

Thus:

\[
\frac{\xi^2 - 2\xi + 1}{h^2} = a_{-1} + a_0 \xi + a_1 \xi^2
\]

(2.18)
And so the coefficients can be read off:

\[ a_{-1} = \frac{1}{h^2}, \quad a_0 = -\frac{2}{h^2}, \quad a_1 = \frac{1}{h^2} \]  

(2.19)

And:

\[ f''(x) = \frac{f(x - h) - 2f(x) + f(x + h)}{h^2} + O(h^2) \]

(2.20)

The next order:

\[ (ik)^2 \approx a_{-2} e^{-2ikh} + a_{-1} e^{-ikh} + a_0 + a_1 e^{ikh} + a_2 e^{i2kh} \]  

(2.21)

Or:

\[ \frac{\xi^2 [\log \xi]^2}{h^2} \approx a_{-2} + a_{-1} \xi + a_0 \xi^2 + a_1 \xi^3 + a_2 \xi^4 \]  

(2.22)

Now:

\[ \xi^2 [\log \xi]^2 = (\xi - 1)^2 + (\xi - 1)^3 - \frac{1}{12} (\xi - 1)^4 + O(h^6) \]

\[ = -\frac{1}{12} + \frac{4}{3} \xi - \frac{5}{2} \xi^2 + \frac{4}{3} \xi^3 - \frac{1}{12} \xi^4 + O(h^6) \]  

(2.23)

Thus:

\[ a_{-2} = -\frac{1}{12h^2}, \quad a_{-1} = \frac{4}{3h^2}, \quad a_0 = -\frac{5}{2h^2}, \quad a_1 = \frac{4}{3h^2}, \quad a_2 = -\frac{1}{12h^2} \]  

(2.24)

And so (VWM):

\[ f'''(x) = \frac{-f(x - 2h) + 16f(x - h) - 30f(x) + 16f(x + h) - f(x + 2h)}{12h^2} + O(h^4) \]

This is a useful formula, since it gives high accuracy for only 5 function evaluations at each grid point.

Suppose we are next to the left grid boundary. Then we cannot go to the left. Still we can use our method. For first derivative:

\[ f_k'(x_n) \approx a_0 f_k(x_n) + a_1 f_k(x_{n+1}) + a_2 f_k(x_{n+2}) \]  

(2.25)
Then:

\[ i k \approx a_0 + a_1 e^{ikh} + a_2 e^{2ikh} \]  

(2.26)

Thus:

\[ \log \frac{\xi}{h} \approx a_0 + a_1 \xi + a_2 \xi^2 \]  

(2.27)

Then:

\[ \log \xi = \frac{1}{2} (-3 + 4\xi - \xi^2) + O(h^3) \]  

(2.28)

So:

\[ -3 + 4\xi - \xi^2 \]  

(2.29)

And we have:

\[ a_0 = \frac{-3}{2h} \quad a_1 = \frac{2}{h} \quad a_2 = -\frac{1}{2h} \]  

(2.30)

And:

\[ f'(x_n) = \frac{-3f(x_n) + 4f(x_{n+1}) - f(x_{n+2})}{2h} + O(h^2) \]  

(2.31)

The second derivative will be:

\[ \frac{(\log \xi)^2}{h^2} \approx a_0 + a_1 \xi + a_2 \xi^2 \]  

(2.32)

So:

\[ \frac{1 - 2\xi + \xi^2 + O(h^3)}{h^2} \approx a_0 + a_1 \xi + a_2 \xi^2 \]  

(2.33)

So:

\[ a_0 = \frac{1}{h^2} \quad a_1 = \frac{-2}{h^2} \quad a_2 = \frac{1}{h^2} \]  

(2.34)

And:
\[
\frac{d^2}{dx^2} f(x_n) = \frac{f(x_{n+1}) - 2f(x_n) + f(x_{n-1})}{h^2} + O(h)
\] (2.35)

This formula is of low quality. Next:

\[
\frac{\log \xi}{h^2} \approx a_0 + a_1 \xi + a_3 \xi^3 + a_4 \xi^4 + a_5 \xi^5
\] (2.36)

The expansion is \((\log \xi)^2 = \frac{15}{4} - \frac{77}{6} \xi + \frac{107}{6} \xi^2 - \frac{13}{2} \xi^3 + \frac{61}{12} \xi^4 - \frac{5}{6} \xi^5 + O(h^6)\). Thus:

\[
\frac{d^2}{dx^2} f(x_n) = \frac{45f_n - 154f_{n+1} + 214f_{n+2} - 156f_{n+3} + 61f_{n+4} - 10f_{n+5}}{12h^2} + O(h^4)
\] (2.37)

III. Discretization

In order to solve numerically Poisson’s equation we need to represent the functions \(\phi(r)\) and \(\rho(r)\) in some numerical way. One approach is to use a “grid”. We use a grid to describe functions. The grid spans a portion of 1D, 2D or 3D space. For example, we will work with 2D. The physical space is a 2D box of lengths \(L_x\) and \(L_y\). In x dimension we have \(N_x + 1\) points and in y dimension \(N_y + 1\) thus the pairs of integers \((i,j)\), \(i = 0, ..., N_x\) and \(j = 0, ..., N_y\) designate \((N_x + 1)(N_y + 1)\) points in the plane. The grid spacing in each direction is assumed, for simplicity equal: \(h = \frac{L_x}{N_x} = \frac{L_y}{N_y}\). We will suppose that the function is strictly zero when \(i = 0\) or \(i = N_x\) or \(j = 0\) or \(j = N_y\).

The charge density is given on the grid as: \(\rho_{ij} = \rho(x_i, y_j)\). \(\phi_{ij}\) must be obtained by solving Poisson’s equation. To do this, we must define the Laplacian on the grid. We use the finite difference formula derived in the previous section.

Thus:

\[
\left[ \frac{\partial^2}{\partial x^2} \phi \right]_{ij} = \frac{\phi_{i,j-1} - 2\phi_{i,j} + \phi_{i,j+1}}{h^2} + O(h^2)
\] (2.38)

And similar expression for \(\partial_y\).

To study the approximation, let us use a 1D example. For a grid with grid-spacing \(h\), the maximal frequency represented on the grid is given by:
\[
\lambda_{\text{min}} = 2h \rightarrow k_{\text{max}} = \frac{2\pi}{\lambda_{\text{min}}} = \frac{\pi}{h}
\] (2.39)

Let us then look at the following analysis. Consider again the Laplacian operation on a wave \( \phi_n = \varepsilon e^{ikx_n} \):

\[
[\nabla^2 \varepsilon e^{ikx}]_n \approx \varepsilon e^{ikx_n} e^{-ikh-2\epsilon h} = 2\varepsilon e^{ikx_n} \frac{\cos kh - 1}{h^2}
\] (2.40)

Compare this with the exact solution, \( k^2 \varepsilon e^{ikx_n} \), we see that the approximation implies:

\[-k^2 \varepsilon e^{ikx_n} \approx \frac{2\cos kh - 1}{h^2} \cdot \frac{\varepsilon}{2} \cdot \frac{\cos kh - 1}{h^2} \].

For \( kh = 1 \) we have \( \cos kh = 1 - \frac{1}{2} (kh)^2 + O((kh)^4) \), so the approximation is reasonable. But once the frequency \( k \) is high, so \( kh \) is considerably larger than 1, this approximation is very bad (see figure below). As mentioned above, the error incurred by discretization is called the “discretization error”.

IV. Solving Poisson's equation by iterative smoothing

How do we solve? We denote the exact solution of the discretized (algebraic) equation by \( \phi_n \):
How close is the exact solution $\bar{\phi}_n$ of the algebraic equation to the exact analytical solution of the differential equation $\phi''(x) = f(x)$ is a matter of how well the problem is represented on the grid. Since we know discretization errors are proportional to $O(h^2)$ we know how to control this error.

Suppose now we have an approximate solution $f_n$ to the discretized, algebraic, equation. It differs from the exact solution by the “algebraic error” vector:

$$\tau_n = \bar{\phi}_n - \phi_n$$

A measure of this kind of error is $\tau = \sqrt{\frac{1}{N} \sum_n \tau_n^2}$ ($N$ is the total number of grid points).

Another measure is the residual vector:

$$r_n = f_n - (L\phi)_n$$

Again, the norm of this vector is a global indicator called “the residual”: $r = \sqrt{\frac{1}{N} \sum_n r_n^2}$.

How can we reduce these indicators once we have a guess to the solution $\phi_n$?

We can try to improve it by iteration. In Eq. (3.2) we “solve” for $\bar{\phi}_n$ and make it an iteration:

$$\phi_n^{\text{better}} = \frac{1}{2}(\phi_{n-1} + \phi_{n+1} - h^2 f_n)$$

We are assuming that the boundary conditions are $\phi_0 = \phi_N = 0$ so we do not change these values: only the inner gridpoint values of $\phi_n$ are changed. Let us take an example. The function we take is a “spike” $f_n = \delta_{n,N}$ where $N + 1$ is the number of grid points.

The interval is $[-1,1]$. Thus $h = \frac{2}{N}$. For $N = 20$, we have $h = 0.1$ and the iterations show the following convergence:
We find that the error indicators can be reduced by this method but the process is very slow. Initially it looks promising but after a few iterations it slows down. When $N$ is larger the situation is worse: after 80 iterations the residual was reduced by just a little more than a factor 2.

In what way is $\phi_i^{\text{better}}$ this really better? Let’s check. The difference between $\phi$ and $\bar{\phi}$, i.e. the “error“ is a combination of waves:

$$\bar{\phi}_n = \phi_n + \sum_k \epsilon_k e^{ikx_n}$$  \hspace{1cm} (3.6)

Because of linearity of the problem, we can treat each wave separately, so suppose just:

$$\bar{\phi}_n = \phi_n + \sum_k \epsilon_k e^{ikx_n} \to \bar{\phi}_n = \phi_n^{\text{better}} + \sum_k \epsilon_k^{\text{better}} e^{ikx_n}$$  \hspace{1cm} (3.7)

Where $k$ can be any wave-vector. Plugging this in (3.5), using (3.2), we have:

$$\phi_n^{\text{better}} = \frac{1}{2}(\phi_{n-1} + \phi_{n+1} - h^2 f_n) = \bar{\phi}_n - \epsilon_k e^{ikx_n} \cos kh$$  \hspace{1cm} (3.8)

Thus $\epsilon_k^{\text{better}} = -\epsilon_k \cos kh$:

$$s_k \equiv \left|\frac{\epsilon_k^{\text{better}}}{\epsilon_k}\right| = |\cos kh|$$  \hspace{1cm} (3.9)

$s_k$ is the reduction factor of the error at frequency $k$. For medium frequencies, $kh \gg \frac{p}{2}$ we see that the iteration effectively reduces the error ($s_k$ small). But for low frequency, at $kh \ll 1$ or high, at $kh \approx \pi$, the iteration is very inefficient ($s_k$ close to 1).
Thus we see that the first few iterations may be efficient since they reduce efficiently all the medium components. So the total error drops dramatically in the first iterations. But after that there are no more medium frequencies. Only high or low frequency errors remain. For these the iterations have virtually no effect! In the example we studied, we needed hundreds of iterations to get reasonable convergence in the coarse grid when \( h = 0.1 \). In a finer grid, when \( h = 0.01 \) the solution, since it has a cusp, has a much higher content of high frequencies and these are not well suppressed by the iterations.

Let’s try another type of iteration. Adding \( 2\omega \bar{\phi}_n \) (where \( \omega \) is arbitrary) to both sides of Eq. (3.2), we obtain:

\[
2(1 + \omega)\bar{\phi}_n = \bar{\phi}_{n-1} + 2\omega \bar{\phi}_n + \bar{\phi}_{n+1} - h^2 f_n
\]

(3.10)

And so, instead of Eq. (3.5), we have now the iteration:

\[
\phi_n^{\text{better}} = \frac{1}{2(1 + \omega)}(\phi_{n-1} + 2\omega \phi_n + \phi_{n+1} - h^2 f_n)
\]

(3.11)

This iteration has the arbitrary parameter which gives it an additional degree of freedom.

When we take \( \omega = 1 \) we have for our problem:

\[
\phi_n^{\text{better}} = \bar{\phi}_n - \varepsilon_k e^{ikx_n} \frac{(e^{-ikh} + 2\omega e^{ikh})}{2 + 2\omega} = \bar{\phi}_n - \varepsilon_k e^{ikx_n} \frac{\cos kh + \omega}{1 + \omega}
\]

(3.12)

And the reduction factor is:

Figure 2: Same as Figure 1 but for \( \omega = 1 \).

It seems nothing substantial has been gained: convergence is slow. But let’s wave analyze it:
\[ s_k = \frac{\cos k h + \omega}{1 + \omega} \quad s_k = \frac{\cos k h + w}{1 + w} \] (3.13)

The effect of the additional degree of freedom is shown in the graph, where we take 4 cases: \( w = 0, 0.5, 1, 2 \). The first of these is our previous iteration:

![Graph showing reduction factor vs frequency for several values of \( \omega \).](image)

Figure 3: reduction factor vs frequency for several values of \( \omega \).

We see that the choice \( \omega = 1 \) is special since it yields an iteration that acts as a "smoother": the higher the frequency the more efficient is the iteration at reducing the error. With these iterations the spike charge problem above will be much better behaved. Yet, even then one needs a huge number of iterations because the low frequency components decay very slowly: After several iterations of the smoother, any initial error is smoothed and has only long wavelengths: the high frequencies were "killed". These low frequency errors are not efficiently damped and the iterations become inefficient.

Another type of smoother can be obtained by taking \( \omega = 0 \) iteration "in-place", i.e. iterate on:

\[ \phi_n^{\text{better}} = \frac{1}{2} (\phi_{n-1}^{\text{better}} + \phi_{n+1} - h^2 f_n) \] (3.14)

This is called the Gauss-Seidel iteration. We can analyze it as:

\[ \varepsilon_k^{\text{better}} e^{ikx_n} = \frac{1}{2} (\varepsilon_k^{\text{better}} e^{ikx_{n-1}} + \varepsilon_k^* e^{ikx_{n+1}}) \] (3.15)
Thus:

$$\varepsilon_k^{\text{better}} (2 - e^{-ikh}) = \varepsilon_k e^{ikh} \rightarrow s_k \equiv \left| \frac{1}{2 - e^{-ikh}} \right| = \frac{1}{\sqrt{5 - 4 \cos kh}}$$

(3.16)

The successive over-relaxation Gauss-Seidel (SOR-GS) iteration is obtained by working in place and in additional averaging the old iterant and the new one:

$$\phi_n^{\text{better}} = (1 - \tilde{\omega}) \phi_n + \frac{1}{2} \left( \phi_{n-1}^{\text{better}} + \phi_{n+1} - h^2 f_n \right)$$

(3.17)

The GS method overshoots the solution; the averaging procedure reduces this.

The analysis gives:

$$\varepsilon_k^{\text{better}} = (1 - \tilde{\omega}) \varepsilon_k + \frac{1}{2} \left( \varepsilon_k^{\text{better}} e^{-ikh} + \varepsilon_k e^{ikh} \right) \rightarrow s_k = \left| \frac{(1 - \tilde{\omega}) + \frac{1}{2} e^{ikh}}{(1 - \frac{1}{2} \tilde{\omega} e^{-ikh})} \right|$$

(3.18)

The performance as smoothers is given in the graph:

Figure 4: The Gauss-Seidel with various successive relaxation coefficients.

The $\omega = 1$ and $\tilde{\omega} = \frac{2}{3}$ behave similarly. However the $\omega = 1$ method damps higher frequencies somewhat better. The benefit of the GS-SOR method is that it needs less memory since everything is done in place.
V. The multigrid method

We select a smoother. It efficiently reduces the error at high frequencies, but it is very inefficient for low ones. Can we turn this deficiency into a winning advantage? The answer, pioneered by Brandt[1] is yes. Since the error is smooth after iterations, we should move the problem to a coarse grid and try to determine the error there. By doing this we should not lose much, because we are only looking for the long wavelengths, and on a coarse grid all operations are twice as fast (in 1D, and 4 or 8 times as fast in 2D or 3D). Some of the low frequency components in the fine grid are now high frequencies in the coarse grid. So they can be effectively reduced by additional smoothing iterations in that grid. If necessary we can then go to even coarser grid.

To explain how this idea is implemented, let us write the Poisson equation on the grid with spacing $h$ (called henceforth $h$-grid) as:

$$ (L^h f^h)_n = g^h_n $$  \hspace{1cm} (4.1)

$L$ is the $h$-grid discretized Laplacian operator. The current approximation, after 2-3 smoothing iterations is $f^h_n$, the exact solution is $\bar{f}^h_n$ and the error:

$$ y^h_n = \bar{f}^h_n - f^h_n $$  \hspace{1cm} (4.2)

We don’t know what the error $y$ is, but since we used a smoother, we know it is composed of mostly low frequency components: it is smooth. While the error is unknown, the residual can be calculated: it is the deviance of the charge density from the Laplacian of the potential:

$$ r^h_n = g^h_n - (L^h f^h)_n $$  \hspace{1cm} (4.3)

Now, watch this:

$$ (L^h y)_n = (L^h \bar{f}^h - L^h f^h)_n = g^h_n - (L^h f^h)_n = r^h_n $$  \hspace{1cm} (4.4)

We calculated the laplacian of the error and and found that it obeys a Poisson equation and that the residual is the new charge density! Since the error $y$ is known to be smooth,
i.e. it is dominated by low frequency components, we can solve this new Poisson equation on a **coarser grid** – a grid with spacing $2h$.

We use the fine-to-coarse operator $I^{2h}_h$ to designate a method of transforming the residual (new "charge density") to the coarse grid. We also designate $L^{2h}$ the discretization of the Laplacian operator on the coarse $2h$ grid:

$$
(L^{2h}f^{2h})_n = g^{2h}_n
$$

(4.5)

Where

$$
(I^{2h}_h r^h)_n = g^{2h}_n
$$

(4.6)

Once $f^{2h}$ is found, we transform it back to the fine grid using a “coarse-to-fine transform”:

$$
y^h = I^{h}_{2h} f^{2h}
$$

(4.7)

We then add $y^h$ to $f^h$ to obtain a new iterant which is closer to the exact result $\bar{f}^h$ and iterate a few more times to smooth again on the fine grid. This additional iteration is usually necessary because some high frequencies which were aliased into the coarse grid now reappear in the fine one.

The process can be repeated. Furthermore, the passage to coarser and coarser grids is advantageous, until a coarse enough grid is reached where a matrix inversion is fast. Suppose problem can is solved with 10 smoother iterations in each level so smoother work on the $l$ level is $10N_l$ where $N_l = 2^{dl}$ is the number of grid points in the $l$th level and where $d$ is the dimension of the problem ($d = 1,2,3$). The total amount of work is $10 \sum_{l=1}^{L} N_l = 10 \sum_{l=1}^{L} N_l = 10 \frac{2^{dL}}{2^d-1} (N_L - 1)$. Thus, in general, the amount of work to solve the problem is of the order $10N_L$.

There are several ways to coarsen, i.e. move the residual from the fine to the coarse grid $I^{2h}_h$. The most straightforward is the injection, which in 1D is:

$$
f^{2h}_n = I^{2h}_h r^h = r^{h}_{2n}
$$

(4.8)
This is directly generalized to 2 and 3D.

The transformation from the coarse to the fine grid can be done by interpolation:

\[
f_n^h \leftarrow f_n^h + (I^h_{2h} y)_n = f_n^h + \left\{ \begin{array}{ll}
y_k^{2h} & n = 2k \\
\frac{y_k^{2h} + y_{k+1}^{2h}}{2} & n = 2k + 1
\end{array} \right. \tag{4.9}
\]

A convenient description is:

\[
I^h_{2h} = \begin{pmatrix}
\frac{1}{2} & 1 & \frac{1}{2}
\end{pmatrix}
\]

(4.10)

Meaning that the point in the coarse grid donates a full weight to itself in the fine grid and half to each neighbor point. Overall it has a weight of 2. This is fine since the long wavelength shape is preserved this way and short wavelength distortions will be easily fixed by smoothing iterations. In 2D we have

\[
I^h_{2h} = \begin{pmatrix}
\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\
\frac{1}{2} & 1 & \frac{1}{2} \\
\frac{1}{4} & \frac{1}{2} & \frac{1}{4}
\end{pmatrix}
\]

(4.11)

With overall weight 4. You can read more about multigrid in the book “A Multigrid Tutorial” by Briggs et al [2].