PY421/621 – Advanced Computing in Physics

Project #2:

-calculation of an electrostatic potential: solving the equations of Laplace and Poisson by a relaxation algorithm;

-rate of convergence and stability of the relaxation algorithm; methods of Gauss-Jacobi and Gauss-Seidel; overrelaxation and underrelaxation;

-multigrid techniques.

Calculation of an electrostatic potential: the equations of Laplace and Poisson.

We wish to calculate the electrostatic potential $\phi(x)$ determined by some conductors which are kept at fixed potentials $V_1, , V_2 \dots V_m$. ϕ will satisfy the Laplace equation

$$\triangle\phi(x) = 0 \tag{1}$$

where the Laplacian operator is given by

$$\triangle = \partial_x^2 + \partial_y^2 \tag{2}$$

$$\triangle = \partial_x^2 + \partial_y^2 + \partial_z^2 \tag{3}$$

in 2 and 3 dimensions respectively.

Moreover ϕ will have to take value $V_1, , V_2 \dots V_m$ over the regions R_1, $R_2 \dots$ R_m occupied by the conductors. Further boundary conditions may have to be specified (e.g. that ϕ must vanish at ∞) to make the solution uniquely determined. In order to simplify the problem we will assume that the regions $R_1, , R_2 \dots R_m$ enclose completely the region R where ϕ must be determined. We will also assume that all of the regions $R_1, , R_2 \dots R_m$ and R are embedded in a box of size L. All this is illustrated in figure 1 below. In order to proceed to the numerical determination of ϕ we will have to discretize space, which we will do assuming that the sides of the box are divided into N equal

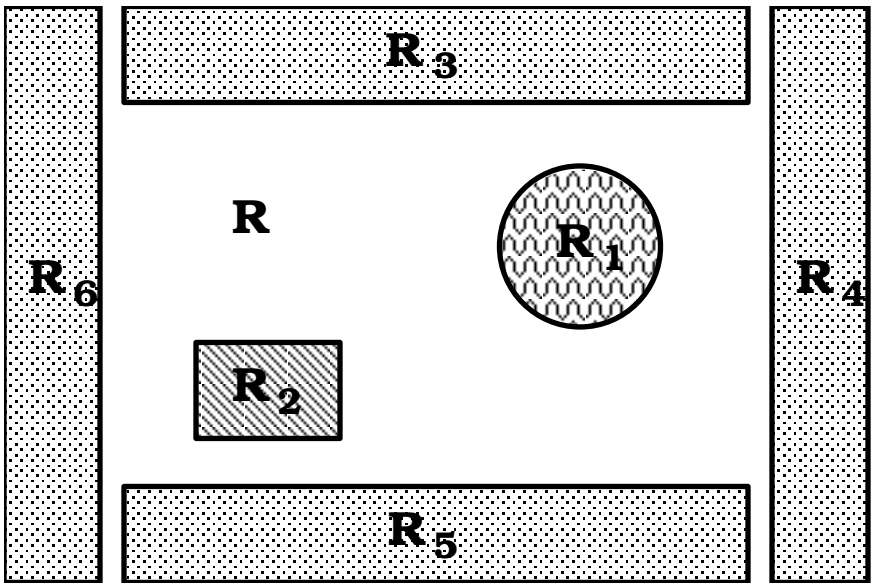

Figure 1: Geometry for the calculation of an electrostatic potential

intervals of width a ($L = Na$). We will then restrict our attention to the values that ϕ takes over the grid of points, or lattice,

$$x = (x_1, x_2) = (n_1 a, n_2 a), \quad n_i = 0 \ldots N-1 \tag{4}$$

in 2 dimensions, or

$$x = (x_1, x_2, x_3) = (n_1 a, n_2 a, n_3 a) \tag{5}$$

in 3 dimensions.

For definiteness we will consider a two dimensional geometry from now on. The values of ϕ will then form a two dimensional array

$$\phi = \phi_{n_1, n_2} \tag{6}$$

Sometimes we will denote the components of this array simply by ϕ_n, where n stands for all the indices n_i.

Using the discretization

$$\partial^2_{x,L}\phi = \frac{\phi(x+a) + \phi(x-a) - 2\phi(x)}{a^2} \tag{7}$$

the Laplace equation becomes

$$\sum_n (\Delta_L)_{m,n}\phi_n = 0 \tag{8}$$

(or even more simply $\Delta_L\phi = 0$) where the matrix elements of the sparse matrix Δ_L are defined implicitly by its action on ϕ:

$$(\Delta_L\phi)_{n_1,n_2} = \frac{1}{a^2}(\phi_{n_1+1,n_2} + \phi_{n_1,n_2+1} + \phi_{n_1-1,n_2} + \phi_{n_1,n_2-1} - 4\phi_{n_1,n_2}) \tag{9}$$

More compactly,

$$(\Delta_L\phi)_n = \frac{1}{a^2}[\sum_{i=1,D}(\phi_{n+\hat{i}} + \phi_{n-\hat{i}}) - 2D\,\phi_n] \tag{10}$$

which is valid in any number of dimensions D and where $\hat{i}$ stands for the vector which has a 1 in position i and all other components equal to 0.

Equation 8 represents a set of inhomogeneous linear equations which must be satisfied by the values ϕ_{n_1,n_2} taken by the potential inside the region R delimited by the conductors. The equations are inhomogeneous because beyond the values taken by ϕ inside R, which are the unknowns of the equations, they also contain the fixed values taken by ϕ over the boundary regions. These fixed values should properly be brought to the r.h.s. of the equations, making their inhomogeneous nature apparent. To bring into evidence the inhomogeneous nature of the equations one can also proceed in a formal way as follows.

Let us denote by ϕ_0 an array which takes values $V_1, , V_2 \ldots V_m$ over the lattice points inside the regions $R_1, , R_2 \ldots R_m$ and 0 over the rest of the lattice. Let us also define

$$\rho = \Delta_L\phi_0 \tag{11}$$

The difference $\bar{\phi} = \phi - \phi_0$ will then satisfy the equation

$$\Delta_L\bar{\phi} = \Delta_L\phi - \Delta_L\phi_0 = -\rho \tag{12}$$

with boundary conditions $\bar{\phi} = 0$ over $R_1, , R_2 \ldots R_m$ (i.e., outside of R). Since inside R $\bar{\phi} = \phi$ it is clear that the equations 12 coincide with the equations we would have obtained by moving in Eqs. 8 the values taken by ϕ over the conductors to the r.h.s. of the equations.

Thus we see that solving the original problem is equivalent to solving the Poisson equation $\Delta_L \bar{\phi} = -\rho$ with Dirichlet boundary conditions (i.e. the field vanishes at the boundary). (Notice, incidentally, that Eq. 12 is the equation we would have encountered if our problem had been to determine the potential in presence of a given distribution of charges with grounded conductors, rather than in presence of a set of conductors kept at fixed potentials. As an interesting follow up, you may consider in more detail what will be the values taken by ρ and think about a physical interpretation of the equivalence between the two computational problems).

We turn now our attention to the solution of Eq. 12 (we will drop the bar from $\bar{\phi}$ henceforth).

In order to deal with a positive semidefinite operator (the Laplacian and its lattice discretization Δ_L are negative semidefinite), and also in order to simplify the notation let us define $A = -\Delta_L$. The Poisson equation thus becomes

$$A\phi = \rho \tag{13}$$

It is useful to separate the diagonal part of A from its non-diagonal part, which we will denote by $-K$. From Eq. 10 and with D=2 we get

$$A = \frac{4}{a^2} I - K \tag{14}$$

where I stands for the identity matrix and K is defined implicitly by

$$(K\phi)_n = \frac{1}{a^2} \sum_{i=1,2} (\phi_{n+\hat{i}} + \phi_{n-\hat{i}}) \tag{15}$$

Inserting Eq. 14 into Eq. 13 this becomes

$$\frac{4}{a^2}\phi = \rho + K\phi \tag{16}$$

and this immediately suggests a possible algorithm for the solution of the equation. Let us consider the iterative procedure

$$\phi \to \phi' = \frac{a^2}{4}\rho + \frac{a^2\, K}{4}\phi \tag{17}$$

If this converges to a stable ϕ, i.e. if eventually $\phi' = \phi$, then clearly the limiting value of the iterative algorithm will be a solution of the Poisson equation.

In general the procedure does converge, albeit the rate of convergence can be sometimes very slow, and the algorithm represented by Eq. 17 goes under the name of Gauss-Jacobi relaxation algorithm.

It is important to note that, although we chose to convert the original Laplace equation with non-zero boundary values into a Poisson equation with Dirichlet boundary conditions, this was only done to emphasize the non-homogeneous nature of the equations and it is not crucial for implementing the relaxation algorithm. Indeed, the relaxation procedure can be formulated directly in terms of the original fields (and this is computationally simpler than going through the introduction of the auxiliary fields ρ and ϕ_0.)

To see this, let us review the relation between the original Laplace equation and the Poisson equation and the working of the relaxation algorithm.

We defined a field ϕ_0 which takes the correct boundary values and vanishes in the interior of the domain. We also defined

$$\rho = -A\phi_0 \tag{18}$$

and set

$$\phi = \phi_0 + \bar{\phi} \tag{19}$$

(we temporarily reinstate the bar here to distinguish between ϕ and $\bar{\phi}$). $\bar{\phi}$ obeys to the equation $A\bar{\phi} = \rho$.

We separate A into its diagonal and non-diagonal parts:

$$A = \frac{4}{a^2} I - K \tag{20}$$

Then, starting from $\bar{\phi} = 0$ we iterate the step

$$\bar{\phi} \to \frac{a^2}{4}(\rho + K\bar{\phi}) \tag{21}$$

Let us denote with superscripts $(0), (1) \ldots$ the successive values taken by the fields. We have

$$\bar{\phi}^{(1)} = \frac{a^2}{4}\rho = \frac{a^2}{4} K\phi_0 - \phi_0 \tag{22}$$

or, directly in terms of ϕ,

$$(\bar{\phi}^{(1)} + \phi_0) = \phi^{(1)} = \frac{a^2}{4} K\phi_0 = \frac{a^2}{4} K\phi^{(0)} \tag{23}$$

At the next iteration we find

$$\bar{\phi}^{(2)} = \frac{a^2}{4}\rho + \frac{a^2}{4}K\bar{\phi}^{(1)} = \frac{a^2}{4}K\phi_0 - \phi_0 + \frac{a^2}{4}K\bar{\phi}^{(1)} \tag{24}$$

i.e., adding to both sides ϕ_0,

$$(\bar{\phi}^{(2)} + \phi_0) = \phi^{(2)} = \frac{a^2}{4}K\phi_0 + \frac{a^2}{4}K\bar{\phi}^{(1)} = \frac{a^2}{4}K\phi^{(1)} \tag{25}$$

etc...

In conclusion, we see that, if we calculate step by step the successive values of $\bar{\phi}$ and use them to calculate the corresponding values of $\phi = \phi_0 + \bar{\phi}$, we obtain the same results we would have obtained applying the relaxation procedure directly to ϕ (and we now drop the bar once again).

Another important consideration has to do with the possible application of Fourier transform techniques to the problem under consideration. In view of what we have learned about the Fourier transform in the course of the first project, it might seem that going to Fourier space would be a much better way to solve the equations of Laplace and Poisson.

Consider for instance Eq. 13, $A\phi = \rho$, but with the simplifying assumption that the field ϕ satisfies periodic boundary conditions (we recall that this is not the case in the problem we are considering, where the boundary conditions are that ϕ vanishes over all the regions $R_1, R_2, \ldots$ in Fig. 1.)

In this case we could indeed solve the problem by using the Fourier transform, as follows.

We introduce the Fourier transform of ϕ

$$\tilde{\phi}_{k_1,k_2} = \frac{1}{N}\sum_{n_1,n_2} \exp(2\pi\imath\frac{n_1k_1 + n_2k_2}{N})\phi_{n_1,n_2} \tag{26}$$

$$\phi_{n_1,n_2} = \frac{1}{N}\sum_{k_1,k_2} \exp(-2\pi\imath\frac{n_1k_1 + n_2k_2}{N})\tilde{\phi}_{k_1,k_2} \tag{27}$$

and similarly for ρ.

From the explicit form of A it is then easy to see that in terms of $\tilde{\phi}$, $\tilde{\rho}$ Eq. 13 becomes

$$\frac{2}{a^2}[2 - \cos(\frac{2\pi k_1}{N}) - \cos(\frac{2\pi k_2}{N})]\tilde{\phi}_{k_1,k_2} = \tilde{\rho}_{k_1,k_2} \tag{28}$$

Solving this equation and inverting the Fourier transform gives then immediately the solution for ϕ.

However, this procedure relies on the fact that the Fourier components are eigenvectors of the discretized Laplacian operator. This set of eigenvectors is determined not only by the local form of the discretized Laplacian, but also by the boundary conditions, which should be considered an integral part (in the literal, not mathematical, sense of the word) of the definition of any operator. With our actual problem, where the field must vanish on the boundary of a domain with a complex geometry, the Fourier modes are no longer eigenvectors of the discretized Laplacian operator, and so using a Fourier transform would be of little help.

In conclusion, there are some cases where, because of the special form of the operator and of the associated boundary conditions, the equations to be solved become diagonal in Fourier space. The use of a Fourier transform represents then a very efficient way to solve the problem. In more general cases, however, this will not happen and then relaxation methods or other algorithms suitable for very sparse matrices (e.g. the algorithm of conjugate gradients) provide the most cost effective and frequently the only viable procedures for bringing the problem to a solution.

Rate of convergence and stability of the relaxation algorithm.

In order to study the rate of convergence of the Gauss-Jacobi relaxation algorithm it is convenient to introduce an (artificial) time dependence and to consider the equation

$$\partial_t \phi(t) = -A\phi(t) + \rho \tag{29}$$

with some initial value for $\phi(t = 0)$ which for the moment we do not need to specify (generally it will be 0 or some initial guess at the solution).

It is clear that, if the time evolution converges to a steady state $\phi(\infty)$, this will satisfy

$$-A\phi(\infty) + \rho = 0 \tag{30}$$

i.e., it will be a solution of our original Poisson equation.

Let us now integrate Eq. 29 numerically with what is known as the Euler or one-step method. The Euler algorithm consists in replacing the derivative

with respect to time in Eq. 29 with its forward difference approximation $\partial_t \phi(t) \approx \frac{\phi(t+dt)-\phi(t)}{dt}$. This produces the iterative algorithm

$$\frac{\phi(t+dt)-\phi(t)}{dt} = -A\phi(t) + \rho \tag{31}$$

i.e.

$$\phi \rightarrow \phi' = \phi - A\, dt\phi + \rho\, dt = (1 - \frac{4}{a^2}dt)\phi + K\, dt\phi + \rho\, dt \tag{32}$$

(Notice that in this equation and in all equations below dt does not represent a differential, but rather a finite, albeit generally small, increment of t.)

In some future lecture we will see that the one-step algorithm gives origin to a rather poor approximation to the actual solution of a time evolution equation, but it is adequate for our present purposes. After all, the time evolution we have just introduced is an artificial one and we are not interested in solving it accurately.

From Eq. 32 we see that if we choose a time step

$$dt = \frac{a^2}{4} \tag{33}$$

there will be no diagonal terms in the r.h.s. of Eq. 32 and the iterative equation itself will reduce to the equation that defines the Gauss-Jacobi relaxation algorithm

$$\phi \rightarrow \phi' = K\frac{a^2}{4}\phi + \rho\frac{a^2}{4} \tag{34}$$

Thus we see that the Gauss-Jacobi relaxation procedure can also be interpreted as the discretization of the time evolution introduced in Eq. 29. This is useful to investigate the rate of convergence and stability of the algorithm (and also to understand its variants, such as the overrelaxation and underrelaxation algorithms).

In order to analyze the rate of convergence of the Gauss-Jacobi relaxation procedure let us denote by ϕ_{exact} the actual solution of Eq. 13 and, borrowing a terminology commonly employed in the study of multigrid methods, let us define as the "error", $\delta\phi$, at any given step the difference between the current iterate and the exact solution:

$$\delta\phi = \phi - \phi_{exact} \tag{35}$$

Notice that the error is generally not known in the course of the relaxation. Indeed, it is clear from the equation above that knowing the error is tantamount to knowing the exact solution.

Since ϕ_{exact} satisfies

$$A\phi_{exact} = \rho \tag{36}$$

substituting into the equation

$$\frac{\phi(t+dt) - \phi(t)}{dt} = -A\phi(t) + \rho \tag{37}$$

we find that the error satisfies

$$\frac{\delta\phi(t+dt) - \delta\phi(t)}{dt} = -A\delta\phi(t) \tag{38}$$

In other words, we find that that in the basic iterative step the error changes according to

$$\delta\phi \rightarrow \delta\phi' = (I - A\,dt)\delta\phi \tag{39}$$

(Notice that, as observed above, $\delta\phi$ is not a known variable, as opposed to ϕ itself. Nevertheless $\delta\phi$ is a well defined mathematical quantity, like ϕ_{exact}, so that it makes sense to talk about its variation at each iterative step.)

We can now expand $\delta\phi$ into the basis formed by the eigenvectors of A:

$$\delta\phi = \sum_i c_i \eta^{(i)} \tag{40}$$

where the eigenvectors $\eta^{(i)}$ satisfy

$$A\eta^{(i)} = \lambda_i \eta^{(i)} \tag{41}$$

and λ_i are the eigenvalues.

Substituting into Eq. 39 we see that the coefficient of the expansion change according to

$$c_i \rightarrow c_i' = r_i c_i \tag{42}$$

with

$$r_i = (1 - \lambda_i dt) \tag{43}$$

In order for the relaxation procedure to converge all of the factors r_i corresponding to a non-vanishing component of the error $c_i\eta^{(i)}$ must have an

absolute value smaller than one. Moreover, the greater the difference $1-|r_i|$, the faster will be the rate of convergence of the corresponding component of ϕ.

This brings us to consider, then, the range of the eigenvalues λ_i. In the study of the blurring of the image (or, equivalently, of the corresponding diffusion equation) we saw that the eigenvalues of the discretized Laplacian over a square lattice with periodic boundary conditions ranged from 0 to $8/a^2$. In the case under consideration, while the differential form of the operator is still the Laplacian, A is a different operator than in the case of the blurring of the image because of the different boundary conditions. The eigenvectors and eigenvalues of A can no longer be found analytically (although with some effort they could be calculated computationally) and 0, in particular, will no longer be an eigenvalue, in presence of Dirichlet boundary conditions. However, on general grounds, we can say that the eigenvalues of A will range from a minimum value $\lambda_{\min}$ which will be quite insensitive to the discretization of the system, to a maximum value $\lambda_{\max}$ which will be very much the same as in the case of the blurring of the image. This is because the lowest eigenvalues of the discretized Laplacian and the corresponding eigenvectors will reflect the long range properties of the continuum system. (To get an intuitive idea of the lowest eigenvectors, we could think of cutting out a metal lamina in the shape of the region R, clamping it at the boundary of the region. If we hit the lamina it will vibrate and the lowest eigenvectors will have the shape of the lowest normal modes of vibration.) On the other hand the highest eigenvalues will reflect local properties of the lattice and be quite insensitive to the boundary conditions. Thus they will be very much the same as when A is defined over a square with periodic boundary conditions. In correspondence to this range of eigenvalues of A we will have

$$(1-\lambda_{\max}dt) \leq r_i \leq (1-\lambda_{\min}dt) \tag{44}$$

or, with $dt = a^2/4$ (see Eq. 33),

$$\left(1-\frac{\lambda_{\max}a^2}{4}\right) \leq r_i \leq \left(1-\frac{\lambda_{\min}a^2}{4}\right) \tag{45}$$

If we substitute $8/a^2$ for $\lambda_{\max}$ we see that at the lower range the factors reach down to a value

$$r_{\min} = \left(1-\frac{8}{a^2}\,\frac{a^2}{4}\right) = -1 \tag{46}$$

With $|r_{\min}| = 1$ the corresponding component of the error will not be reduced, but it will not grow either, and thus, since we may expect the corresponding c_i in the initial value of the error to be very small, this lack of convergence for a single component of the error will not be a shortcoming of the method. Something similar can be said of the other short range components of the error, which will decrease at a slow rate (since the corresponding r_i will be close to -1), but will again typically be very small in the initial value of the error. On the other side of the range of the r_i factors we encounter the long range components of the error. These will all decrease, since the corresponding r_i will be smaller or equal to

$$r_{\max} = \left(1 - \frac{\lambda_{\min} a^2}{4}\right) < 1 \tag{47}$$

However the rate of convergence will be limited by the rate of decay of the component of the error corresponding to $\lambda_{\min}$. This will be given by

$$|\log r_{\max}| = \left|\log\left(1 - \frac{\lambda_{\min} a^2}{4}\right)\right| \approx \frac{\lambda_{\min} a^2}{4} \tag{48}$$

Since, as we discussed above, $\lambda_{\min}$ is rather insensitive to the discretization, we see from Eq. 48 that the rate of convergence of the relaxation algorithm will decrease like a^2 when the grid is made finer. This goes under the name of "critical slowing down" and constitutes a serious limitation of the relaxation technique. Critical slowing down is not specific to the problem we chose to solve. It is a rather general feature of the relaxation method.

Overrelaxation and underrelaxation.

If we fix the parameter dt in the basic relaxation step to values larger or smaller than the value that makes the diagonal elements vanish (cfr. Eq. (21)) we obtain the overrelaxation and underrelaxation algorithms, respectively. There are limits, however, to the amount of overrelaxation one can implement without inducing instabilities. In the problem we studied we can underrelax, but we cannot make dt larger than $a^2/4$ because this would push the factors r_i for the short range modes to values smaller than -1, with $|r_i| > 1$, causing an instability. Thus we cannot overrelax. In other cases overrelaxation is possible, but the amount of overrelaxation that can be applied is in general quite limited, as we will show by considering the following example.

Let us assume that the equation we have to solve has the form

$$-\Delta_L \phi + m^2 \phi = \rho \tag{49}$$

We define now

$$A = -\Delta_L + m^2 \tag{50}$$

so that the equation to solve is still $A\phi = \rho$.

We write again the iteration step in the form

$$\phi \to \phi' = \phi - A\,dt\,\phi + \rho\,dt = [1 - (\frac{4}{a^2} + m^2)\,dt]\,\phi + K\,dt\,\phi + \rho\,dt \tag{51}$$

where we have paid attention to the m^2 term in the diagonal part of A.

For the straightforward relaxation algorithm we take

$$dt = dt_0 = \frac{a^2}{4 + m^2\,a^2} \tag{52}$$

Let us define an over(under)relaxation parameter ω and set

$$dt = \omega dt_0 = \omega \frac{a^2}{4 + m^2\,a^2} \tag{53}$$

In terms of ω the iterative step becomes

$$\phi \to \phi' = (1 - \omega)\phi + \omega \frac{K\,a^2}{4 + m^2\,a^2}\phi + \omega \frac{a^2}{4 + m^2\,a^2}\rho \tag{54}$$

The evolution of the error will be given by a similar formula without the inhomogeneous term

$$\delta\phi \to \delta\phi' = (1 - \omega)\delta\phi + \omega \frac{K\,a^2}{4 + m^2\,a^2}\delta\phi \tag{55}$$

In terms of the components c_i of the error in the basis formed by the eigenvectors of K (which are the same as the eigenvectors of A) the same equation becomes

$$c_i \to c_i' = r_i c_i \tag{56}$$

with

$$r_i = 1 - \omega + \omega \frac{\kappa_i a^2}{4 + m^2\,a^2} \tag{57}$$

κ_i being the eigenvalues of K.

In order to continue with a quantitative discussion we must assume that the range of eigenvalues of K is similar to the one when the fields ϕ and ρ are defined over a square with periodic boundary conditions. In this case we have seen that the eigenvalues of the discretized Laplacian range between $-8/a^2$ and 0 and those of $K = \Delta_L + 4I/a^2$ range over

$$-\frac{4}{a^2} \leq \kappa_i \leq \frac{4}{a^2} \tag{58}$$

with the highest eigenvalues corresponding to the long range eigenvectors.

In correspondence with Eq. 58 we find

$$r_{min} \leq r_i \leq r_{max} \tag{59}$$

with

$$r_{min} = 1 - \omega - \omega \frac{4}{4 + m^2\, a^2} \tag{60}$$

$$r_{max} = 1 - \omega + \omega \frac{4}{4 + m^2\, a^2} \tag{61}$$

It is useful to recast these formulae as follows. We define a parameter of order a^2

$$\epsilon = 1 - \frac{4}{4 + m^2\, a^2} = \frac{m^2\, a^2}{4 + m^2\, a^2} \tag{62}$$

We then have

$$r_{max} = 1 - \omega\epsilon \tag{63}$$

$$r_{min} = 1 - 2\omega + \omega\epsilon \tag{64}$$

Since ω will typically not be much less than one, Eq. 64 can be more informatively recast as

$$|r_{min}| = 1 - \omega\epsilon + 2(\omega - 1) \tag{65}$$

From these equations we can easily read off the effect that overrelaxation and underrelaxation will have on convergence.

Overrelaxation will make the error reduction factors r_i smaller (it will move them farther away from 1, towards 0) for the *long range* modes, corresponding to the largest eigenvalues of the matrix K. This is welcome for the problem we are actually interested in, namely the behavior of the continuum system,

since the long range modes are those that correspond to true physical features rather than lattice artifacts. However there is the danger of inducing instabilities on the other end of the spectrum. Indeed, increasing ω will also move $|r_{min}|$ towards 1 and, eventually, also to a value > 1. This will happen if $-\omega\epsilon + 2(\omega - 1) > 0$, i.e. for

$$\omega > \frac{1}{1 - \epsilon/2} \approx 1 + \frac{\epsilon}{2} \tag{66}$$

Thus we see that the possible range for the overrelaxation is quite limited. If we try to overrelax too much, although we gain in rate of convergence for the physical modes, we induce an instability signaled by the growth of rapidly varying lattice artifacts.

On the other hand underrelaxation never leads to instabilities, but it slows down convergence of the physical, long range modes. Its only advantage is that it increases the rate of reduction of the components of the error along the most rapidly varying modes and therefore has a smoothing effect on the field. It can be useful in multilevel algorithms, such as the multigrid techniques which we will briefly discuss later.

The algorithm of Gauss-Seidel and considerations about its parallel implementation.

A programmer implementing the basic relaxation step (Eq. 54 with $\omega = 1$) on a serial machine and in a somehow hasty and careless manner could easily make a mistake leading to a code that does something slightly different than what Eq. 54 specifies.

Consider the following Fortran code

```
INTEGER :: N,x,y
REAL :: m,a,aux1,aux2
REAL, DIMENSION(0:N-1,0:N-1) :: phi,rho

aux1=1./(4+(a*m)**2)
aux2=aux1*a**2
```

```
DO y=0,N-1
   DO x=0,N-1
      phi(x,y)=aux1*(phi(MOD(x+1,N),y)  &
      +phi(MOD(x+N-1,N),y)+phi(x,MOD(y+1,N))  &
      +phi(x,MOD(y+N-1,N)))+aux2*rho(x,y)
   ENDDO
ENDDO
```

(Note: the use of the MOD function is probably not the most efficient way to implement periodic boundary conditions, but we will not be concerned with code performance here.)

Although these line of code may seem to implement Eq. 54, they do not. Equation 54 states that a new field ϕ' should be defined entirely in terms of the current value ϕ of the field. The above code replaces the current value of ϕ with an upgraded value ϕ' site by site. This means that two of the field components in the r.h.s. of the assignment statement will be indeed components of ϕ, but the other two will already be components of ϕ'.

Yet, in spite of the fact that the code above does not implement the Gauss-Jacobi relaxation procedure, it still does give origin to a well defined algorithm, which goes under the name of Gauss-Seidel relaxation algorithm. It is not as straightforward to study the convergence properties of this algorithm as it is for the algorithm of Gauss-Jacobi and we will not analyze it in detail. In order to formalize the procedure, one must index the lattice sites in the exact sequential order in which they are scanned by the do loops in the code. For instance, in the example above we would introduce a single index i ranging from 0 to $N^2 - 1$ and would assign sites as follows

$$\begin{aligned} i = 0 &\leftrightarrow x = 0, y = 0 \\ i = 1 &\leftrightarrow x = 1, y = 0 \\ i = 2 &\leftrightarrow x = 2, y = 0 \\ &\dots \\ i = N &\leftrightarrow x = 0, y = 1 \\ i = N + 1 &\leftrightarrow x = 1, y = 1 \\ i = N + 2 &\leftrightarrow x = 2, y = 1 \\ &\dots \end{aligned}$$

$$\text{etc.} \tag{67}$$

With this indexing, let us denote by K_U and K_L the upper and lower components of the matrix K, i.e. define

$$K = K_U + K_L \tag{68}$$

where all of the elements of K_U (K_L) on and below (above) the diagonal are zero (we recall that K has no diagonal elements by definition).

Then with a little thought we can easily convince ourselves that the Gauss-Seidel basic iterative step is described by the following equation:

$$\phi \rightarrow \phi' \tag{69}$$

with

$$(1 - \frac{K_L\, a^2}{4 + m^2\, a^2})\phi' = \frac{K_U\, a^2}{4 + m^2\, a^2}\phi + \frac{a^2}{4 + m^2\, a^2}\rho \tag{70}$$

That is, going through all of the sites of the lattice has the effect of replacing ϕ with a new field ϕ' given by Eq. 70.

The algorithm of Gauss-Seidel of course requires less memory than the algorithm of Gauss-Jacobi (since there is no need to keep stored two copies of the field). It is an interesting and important fact that its convergence and stability properties are also better. Thus, on a serial machine at least, it would generally represent a better option than the algorithm of Gauss-Jacobi.

On a parallel machine, of course, the implementation of the Gauss-Seidel algorithm in the form given above is impossible. The dependencies in the loop make the procedure intrinsically serial. However a variant of the algorithm can be implemented on a parallel machine as well.

Let us index the sites of the lattice in the following manner: we still scan the lattice in the x first then y lexicographic order, but, to begin with, *we index only the even sites* (i.e. the sites for which MOD(x+y,2)=0). Then, after we have indexed all the even sites, we proceed to the enumeration of all odd sites.

We notice now that the matrix K *connects only even to odd sites* and viceversa, but has no matrix elements between sites of the same parity. More specifically, with the ordering we have chosen, K_U has non-vanishing matrix elements only from odd sites to even sites, K_L only from even sites to odd

sites. Let us therefore separate ϕ and ϕ' into there even and odd components ϕ_e, ϕ_o, ϕ'_e, ϕ'_o (these even and odd components are at the same time the first half or upper half and the second half or lower half of the components of the arrays). Then the basic iterative step of Eq. 70 takes the form

$$\phi'_e = \frac{K_U\, a^2}{4 + m^2\, a^2}\phi_o + \frac{a^2}{4 + m^2\, a^2}\rho_e \tag{71}$$

$$\phi'_o - \frac{K_L\, a^2}{4 + m^2\, a^2}\phi'_e = \frac{a^2}{4 + m^2\, a^2}\rho_o \tag{72}$$

This shows that the iterative step really breaks up into two substeps

$$\phi_e \to \phi'_e \tag{73}$$

with

$$\phi'_e = \frac{K_U\, a^2}{4 + m^2\, a^2}\phi_o + \frac{a^2}{4 + m^2\, a^2}\rho_e \tag{74}$$

and

$$\phi_o \to \phi'_o \tag{75}$$

with

$$\phi'_o = \frac{K_L\, a^2}{4 + m^2\, a^2}\phi'_e + \frac{a^2}{4 + m^2\, a^2}\rho_o \tag{76}$$

But there are no more dependencies in these two separate steps. This means that the even sites and odd sites components of the fields can be separately upgraded in parallel.

This method of resolving the dependencies in an algorithm by subdividing the lattice into two sublattices formed by the even and odd sites respectively goes under the name of red-black or checkerboard algorithm. It is crucial for the vectorized or parallel implementation of several computational techniques, beyond the Gauss-Seidel method which we have been considering now. Checkerboard algorithms can be implemented very efficiently on parallel machines, but one has to pay some attention of the arrangement of the data, which is not as straightforward as when one deals with the whole lattice at one time. As a final remark, we should also observe that the the Gauss-Seidel algorithm in its checkerboard implementation, although still advantageous in memory utilization and for better convergence properties, is not identical to its original serial version. As Eqs. 69, 70 clearly point out, the mathematical details of the basic relaxation step do depend of the

specific break-up of K and different orderings of the sites will give origin to different break-ups.

Multigrid techniques.

We have seen how the rate of convergence of relaxation algorithms depends on the spectrum of eigenvalues of the discretized differential operator. The closer the factors r_{max} and $|r_{min}|$ are to one, the slower becomes the rate of convergence. On the other hand, the example we considered in some detail has shown us that the distance of r_{max} from 1 is proportional to the square of the lattice spacing (cfr. Eqs. 62 and 63) and this is a rather typical situation. This suggests that we might improve the convergence by working on a coarser lattice, with larger lattice spacing a. However, a coarser lattice will be, per se, obviously unsatisfactory because of the poorer resolution it provides. The way out of the dilemma lies in the observation that the modes responsible for slow convergence, i.e. the eigenvectors associated with the largest values of $r^{(i)}$, are typically the long-range modes, those exhibiting the slowest variation of the field. These can be well reproduced also on a coarser lattice. We can therefore put together the advantages that derive from the faster convergence on a coarser lattice and the higher resolution of a finer lattice if we adopt a "multigrid" technique, which uses both lattices. As a matter of fact, the scheme can be iterated, and one can use several lattices of different degrees of coarseness. The term "multigrid methods" refers indeed, in its generality, to those techniques that use multiple scales of discretization.

Let us now consider now the multigrid technique in some more detail.

We wish to solve Eq. 13:

$$A\phi = \rho$$

We take ϕ to represent the actual value of the field, as stored in the memory of the computer, at some definite point in the iterative procedure. We will use again the symbol ϕ_{exact} to characterize the actual solution of Eq. 13.

We define a residue (note: r is an array, just as ϕ and ρ)

$$r = \rho - A\phi \tag{77}$$

and a correction or error (as in Eq. 35, but with the opposite sign and with a different symbol)

$$e = \phi_{exact} - \phi \tag{78}$$

From Eqs. 13,77,78 and the fact that ϕ_{exact} satisfies $A\phi_{exact} = \rho$ we see that the error obeys the equation

$$Ae = r \tag{79}$$

Of course, we do not know e. Knowing e would be equivalent to knowing the exact solution. Also, it is important to notice that at any iterative step the original problem (solving Eq. 13) can be replaced with the problem of solving the equation for the error, Eq. 79. Indeed, having defined r from the current value of ϕ, if we solve Eq. 79, we can then obtain ϕ_{exact} simply by adding to the current iterate ϕ the correction e.

The basic idea of the multigrid method is that both r and e will be approximated on a coarser lattice, where an approximate calculation of e will be performed. The algorithm proceeds through the following steps:

1) start from a given ϕ, calculate r according to

$$r = \rho - A\phi \tag{80}$$

2) project r onto a coarser lattice (the details will be given later) to obtain r^c. Notice that r^c is an array defined over a lattice with much fewer points than the original one and has correspondingly fewer components;

3) define a projection A^c of the original operator over the coarser lattice;

4) starting from an initial value $e^c = 0$ perform some number of steps of an iterative procedure for solving Eq. 79 on the coarser lattice: $A^c\, e^c = r^c$;

5) interpolate the error e^c onto the original fine lattice (again, detail will be given later) to obtain the fine-grid correction e and add this to ϕ to obtain a new value for the field.

Notice that all of the five steps in the algorithm correspond to well defined procedures, that could be implemented by suitable subroutines. It is indeed convenient to introduce a subroutine like notation also for purposes of explanation. Let us thus make reference to the five steps above in terms of "subroutines":

1) residue(r, phi, rho) - This subroutine receives the current value of the field ϕ and the array ρ and returns r. We do not add A to the list of arguments, because the very sparse matrix A is never stored explicitly, but is rather "hard-wired" in the subroutine. It would be appropriate, however, to pass

to the subroutine the parameters which enter in the definition of A, such as the lattice spacing a and m. We will leave these implicit, nevertheless.

2) project(rc, r) - The meaning is obvious.

3) define_Ac - As we will see, all this module amounts to is a calculation of the parameters of the coarse grid operator A^c. This will often be totally straightforward, e.g it may reduce to the redefinition $a = 2 * a$ and will not require a separate subroutine.

4) relax(ec, r, ec0, nit) - Starting from e_0^c (which may be 0) perform nit steps of the iterative procedure used to solve $A^c\, e^c = r^c$.

5) interpolate(phi, ec) - Given the correction e^c over the coarser lattice project it first onto the fine lattice to obtain an array e and add e to the current value of ϕ: $\phi = \phi + e$. This overwrites the initial value of ϕ, which is passed as argument to the subroutine.

The definition of the projection from the fine lattice to the coarse lattice is not unique. Several different projection methods can be used. The definition itself of the coarse lattice leaves some degree of arbitrariness. But, in any case, the projection must conform to the underlying principle that we wish to have a mechanism for approximating well the long-range modes responsible for slow convergence on the coarser lattice. We will define now two possible projection procedures for a two-dimensional square lattice. These are the most obvious procedures, those that would be used in most applications. Moreover, one can easily derive from these two examples guidelines for implementing more complex projections, for the cases where such may be needed.

A) We take the sites of the coarse lattice to be the centers of 2×2 cells of the finer lattice and define the coarse lattice residue as the average of the fine lattice residue over the cell. In program notation, assuming that r and rc have been dimensioned as `r(0:N-1,0:N-1)` and `rc(0:N/2-1,0:N/2-1)` we would have (for all `i,j` ranging over `0:N/2-1`, and with proper attention to the boundary conditions):

```
rc(i,j)=0.25*(r(2*i,2*j)+ r(2*i+1,2*j)+ r(2*i+1,2*j+1)
        +r(2*i,2*j+1))
```

B) We identify the sites of the coarse lattice with the sites of even parity of the fine lattice and take the new residue to be a weighted average of the

residues at the site itself (weight 1), at the nearest neighbor sites (weight 1/2, the fine site is shared between two coarse sites) and at the next nearest neighbor sites (weight 1/4, the fine site is shared among four coarse sites):

```
rc(i,j)=0.25*(r(2*i,2*j)+ 0.5*(r(2*i+1,2*j)+ r(2*i,2*j+1)
        + r(2*i-1,2*j)+r(2*i,2*j-1))+0.25*(r(2*i+1,2*j+1)
        +r(2*i-1,2*j+1)+r(2*i-1,2*j-1)+r(2*i+1,2*j-1)))
```

Correspondingly, the interpolation operator will be as follows:

A) the coarse correction e^c will be added (with weight 1) to the value of the field at all the 4 (fine lattice) sites of the cell;

B) the coarse correction e^c will be added with weight 1 to the field at the same site in the fine lattice, with weight 1/2 to the fields at the nearest neighbor sites and with weight 1/4 to the fields at the next nearest neighbor sites. (Notice that all the field elements over the fine lattice will receive contributions with weights that add up to 1).

Finally, about the coarse lattice operator A^c, the most important point is that it must have eigenvectors and eigenvalues that approximate well, in the part of the spectrum corresponding to the long range modes, those of the original operator A. In many cases, especially in connection with the discretization of simple differential operators such as the Laplace operator, the prescription to follow will be obvious. In the examples of differential operators we have considered in the previous lectures as coarse grid operator we would choose the same one we used on the fine grid, but with a replaced by $2a$. However, in some more elaborate problems, it is not obvious how to choose an optimal operator, and this may become a topic of research.