Partial Differential Equations

Last time, we looked at the Jacobi algorithm for solving Laplace's equation on a grid:

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = 0$$

$$V(i,j,k) = \frac{[(V(i+1,j,k) + V(i-1,j,k)](\Delta y)^2(\Delta z)^2 + \ldots}{2\left((\Delta y)^2(\Delta z)^2 + (\Delta x)^2(\Delta z)^2 + (\Delta x)^2(\Delta y)^2\right)}$$

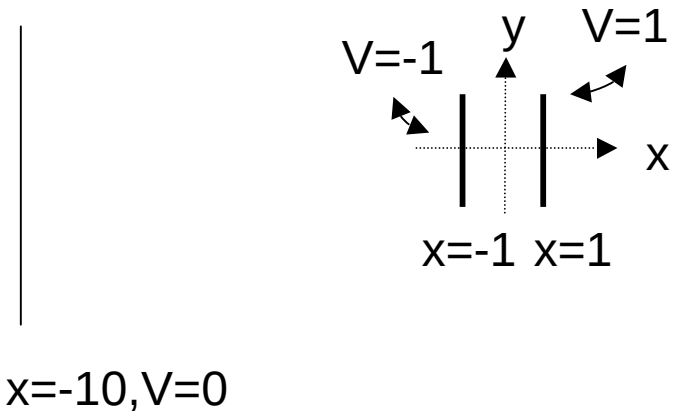
Iterative method (equal step size):

$$V^{(r+1)}(i,j,k) = \frac{1}{6}\begin{bmatrix} V^{(r)}(i+1,j,k) + V^{(r)}(i-1,j,k) + \\ V^{(r)}(i,j+1,k) + V^{(r)}(i,j-1,k) + \\ V^{(r)}(i,j,k+1) + V^{(r)}(i,j,k-1) \end{bmatrix}$$

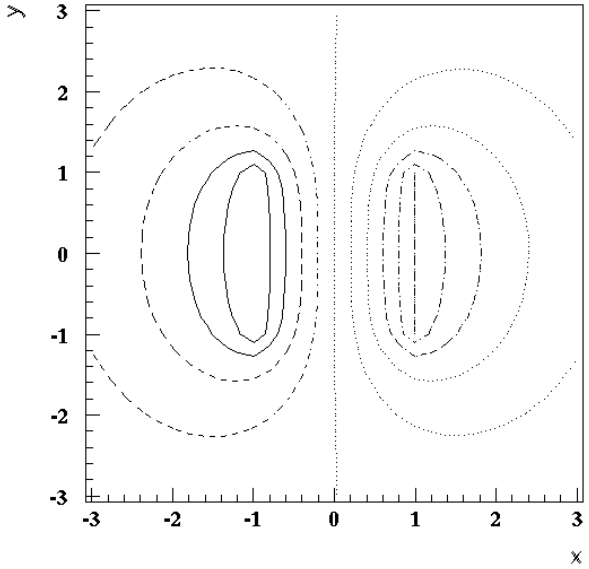
Jacobi Method

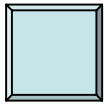
Relaxation Methods and Diffusion

As we discussed last time, the relaxation method propagates the information from the boundary conditions throughout the volume of interest in a way consistent with the differential equations. In this way, we turn a search for a steady state solution into a kind of diffusion problem.



V





Relaxation Methods and Diffusion

Diffusion equation:

$$\frac{\partial \bar{V}(x,y,z,t)}{\partial t} = D\left(\frac{\partial^2 \bar{V}}{\partial x^2} + \frac{\partial^2 \bar{V}}{\partial y^2} + \frac{\partial^2 \bar{V}}{\partial z^2}\right)$$

Diffusion Coefficient

Steady-state limit is the solution we are looking for. In the steady-state limit,

$$\frac{\partial \bar{V}(x,y,z,t)}{\partial t} = 0$$

So,

$\bar{V}(x,y,z,t \to \infty)$ is a solution of Laplace's equation

Matrix Representation of Relaxation

To make the math clearer, look in 1-D. Use a grid with n+1 sites, labelled 0,1,…,n. We will assume equal grid spacing. We use the usual approximation for the second derivative:

$$\frac{d^2V(x_j)}{dx^2} \approx \frac{V(x_{j+1}) - 2V(x_j) + V(x_{j-1})}{h^2} = \frac{V(j+1) - 2V(j) + V(j-1)}{h^2}$$

where h is the grid spacing, and $x_j = j \cdot h$

Boundary Conditions: $V(x_0) = V_0$ $V(x_n) = V_n$ are fixed

So, the Laplace equation on a grid gives a system of linear equations

$$V(j-1) - 2V(j) + V(j+1) = 0$$

n-1 equations for
n-1 unknowns

Matrix Representation of Relaxation

Or, in matrix form

$$\begin{pmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & 0 & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & \ddots & & \\ & 0 & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ \\ V_{n-2} \\ V_{n-1} \end{pmatrix} = \begin{pmatrix} -V_0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ -V_n \end{pmatrix}$$

$$A\vec{V} = \vec{b}$$

$$\vec{V} = A^{-1}\vec{b}$$

i.e., determining the potential on a grid is equivalent to solving a system of inhomogeneous linear equations (SLE's). In one dimension, this can solved with a matrix inversion (to get A^{-1}). More on this type of problem in a future lecture. In several dimensions, the matrix inversion can be very slow and often does not converge, so different techniques are necessary. However, notice the diagonal structure of the matrix.

Matrix Representation of Relaxation

Formulate now more generally:

In matrix notation $A\vec{V} = \vec{b}$

We will solve the set of linear equations using an iterative approach, as we did in the Jacobi method.

When we solve by iteration, we write $\vec{V}^{(r+1)} = M\vec{V}^{(r)} + N\vec{b}$

e.g., in our 1-D example

$$V^{(r+1)}(i) = \frac{V^{(r)}(i+1) + V^{(r)}(i-1)}{2}$$

Matrix Representation of Relaxation

For convergence, we require $\vec{V}^{(r+1)} = M\vec{V}^{(r)} + N\vec{b} = \vec{V}^{(r)}$

and $\vec{V}^{(r)} = A^{-1}\vec{b}$, which gives

$A^{-1}\vec{b} = MA^{-1}\vec{b} + N\vec{b}$

This is consistent with: $A^{-1} = MA^{-1} + N$, so it holds if $M + NA = E$

where E is the unit matrix

Substituting above gives

$$\vec{V}^{(r+1)} = (E - NA)\vec{V}^{(r)} + N\vec{b} = \vec{V}^{(r)} - N(A\vec{V}^{(r)} - \vec{b})$$

$$\text{or } N^{-1}(\vec{V}^{(r+1)} - \vec{V}^{(r)}) = -(A\vec{V}^{(r)} - \vec{b})$$

Different techniques come from different choices for N^{-1}

Jacobi Method

First, divide the matrix *A* into three parts (we saw that the non-zero elements were clustered around the diagonal):

$$A = L + U + D$$

where *D, L, U,* are the diagonal, lower triangular and upper triangular part of the matrix *A.* For N^{-1}, the diagonal part is chosen for the Jacobi Method which we have been using.

$$N^{-1} = D$$

Which gives

$$M = E - NA = E - D^{-1}(L + U + D) = -D^{-1}(L + U)$$

$$\vec{V}^{(r+1)} = M\vec{V}^{(r)} + N\vec{b} = -D^{-1}(L + U)\vec{V}^{(r)} + D^{-1}\vec{b}$$

$$V^{(r+1)}(i) = -\frac{1}{a_{ii}} \sum_{j \neq i} a_{ij} V^{(r)}(j) + \frac{1}{a_{ii}} b_i$$

Gauss-Seidel Method

In this method, we choose $N = (D + L)^{-1}$

which leads to $$V^{(r+1)}(i) = -\frac{1}{a_{ii}}\left(\sum_{j<i} a_{ij} V^{(r+1)}(j) + \sum_{j>i} a_{ij} V^{(r)}(j) + b_i \right)$$

Advantage is that already updated results used on the fly, leading to a faster convergence.

Solve the parallel plate problem from last time - need 139 iterations to get within the same max variation (instead of 178).

Overrelaxation Gauss-Seidel Method

$$V^{(r+1)}(i) = (1-w)V^{(r)}(j) - \frac{w}{a_{ii}}\left(\sum_{j<i} a_{ij}V^{(r+1)}(j) + \sum_{j>i} a_{ij}V^{(r)}(j) + b_i \right)$$

For convergence, require that $0 < w < 2$

For a square grid with M^2 points, the optimum value is

$$w_{opt} \approx \frac{2}{1+\pi/\sqrt{M}} \qquad w_{Jacobi} = w_{Gauss-Seidel} = 1$$

In our example, $w_{opt} \approx \frac{2}{1+\pi/101} \approx 2(1-0.03) \approx 1.94$

Using this value gives convergence after 112 iterations

Comparison of Methods

Take Laplace equation in 3-D:

$$V^{(r+1)}(i,j,k) = \frac{1}{6}\left[V^{(r)}(i+1,j,k) + V^{(r)}(i-1,j,k) + \cdots\right]$$ Jacobi Method

$$V^{(r+1)}(i,j,k) = \frac{1}{6}\left[V^{(r)}(i+1,j,k) + V^{(r+1)}(i-1,j,k) + \cdots\right]$$ Gauss-Seidel

Define $\Delta V^{(r+1)}(i,j,k) = V^{*(r+1)}(i,j,k) - V^{(r)}(i,j,k)$

where V^* gives the new voltage from the chosen method

(Jacobi, Gauss - Seidel).

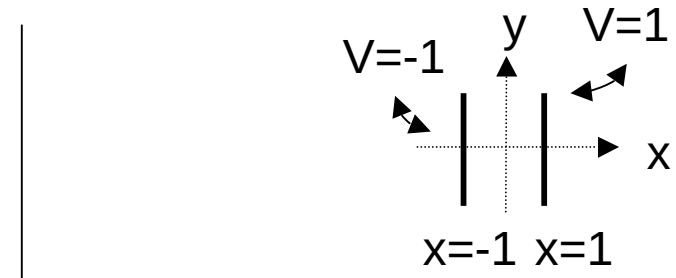
$$V^{(r+1)}(i,j,k) = \alpha \Delta V^{(r+1)}(i,j,k) + V^{(r)}(i,j,k)$$

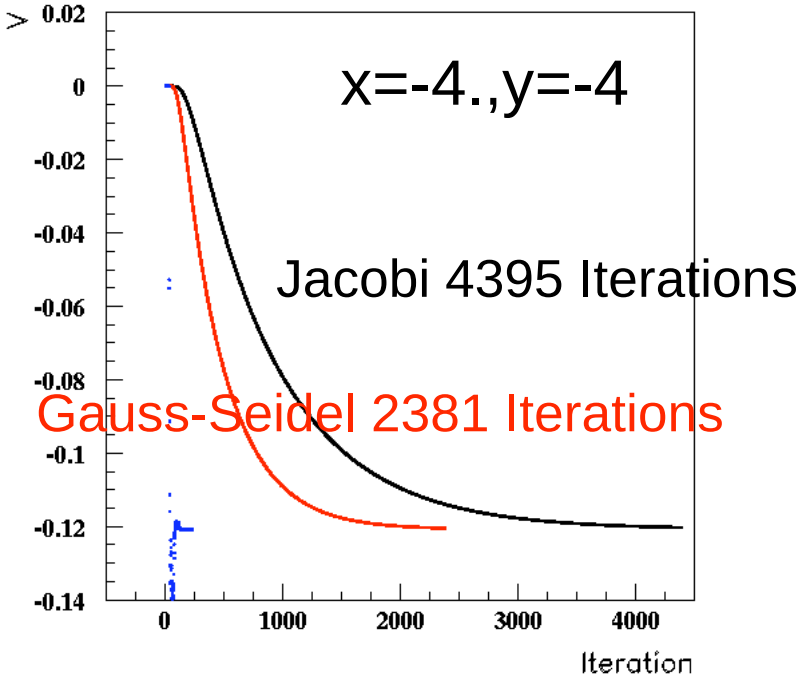
For $\alpha = 1$, $V^{(r+1)}(i,j,k) = V^{*(r+1)}(i,j,k)$

For $\alpha > 1$, we 'overrelax' (speed up changes), $\alpha < 1$ 'underrelax'

Comparison of Speed of Convergence

Parallel plate capacitor problem as in the last lecture. This time, take max variation per iteration 10^{-6}. Compare speed of convergence of different algorithms. Monitor V at different points.

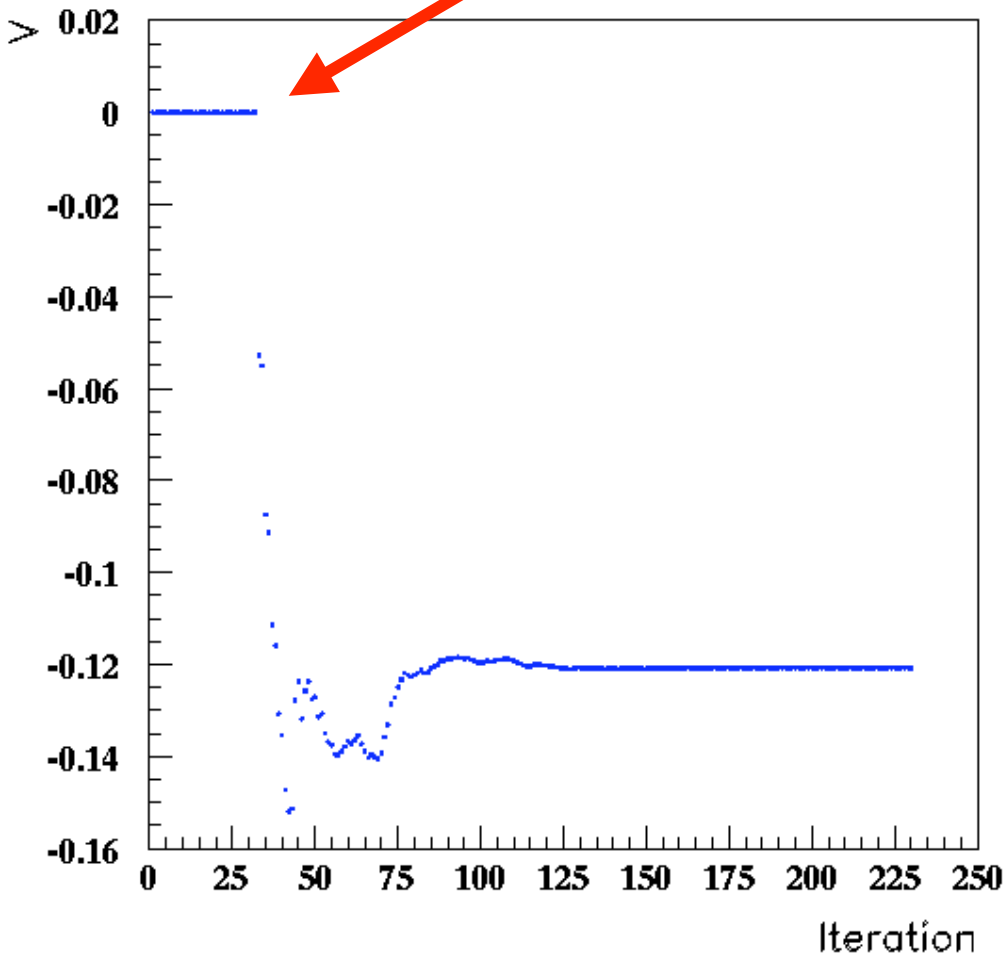




Gauss-Seidel,w_{opt}=*1.94, 230 Iterations*

Comparison of Methods

Voltage at a particular point. Note that nothing changes until 'wave' from boundary has reached the particular point. Then method converges quickly.



Poisson Equation

In the parallel plate example we worked out, we did not have any sources. They are easily implemented (Laplace $\rightarrow$ Poisson Equation)

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)V(x,y,z)=\frac{\rho(x,y,z)}{\varepsilon_0}$$

Gauss-Seidel with Overrelaxation

$$V^{(r+1)}(i,j,k)=(1-w)V^{(r)}(i,j,k)-\frac{w}{6}\begin{pmatrix}V^{(r+1)}(i-1,j,k)+V^{(r)}(i+1,j,k)+\\ V^{(r+1)}(i,j-1,k)+V^{(r)}(i,j+1,k)+\\ V^{(r+1)}(i,j,k-1)+V^{(r)}(i,j,k+1)+\\ \frac{\rho(i,j,k)}{\varepsilon_0}h^2\end{pmatrix}$$

with $$\rho(i,j,k)=\frac{\iiint\rho(x,y,z)dxdydz}{\Delta x\Delta y\Delta z}$$

and *h* the cell size (assumed same in all directions)

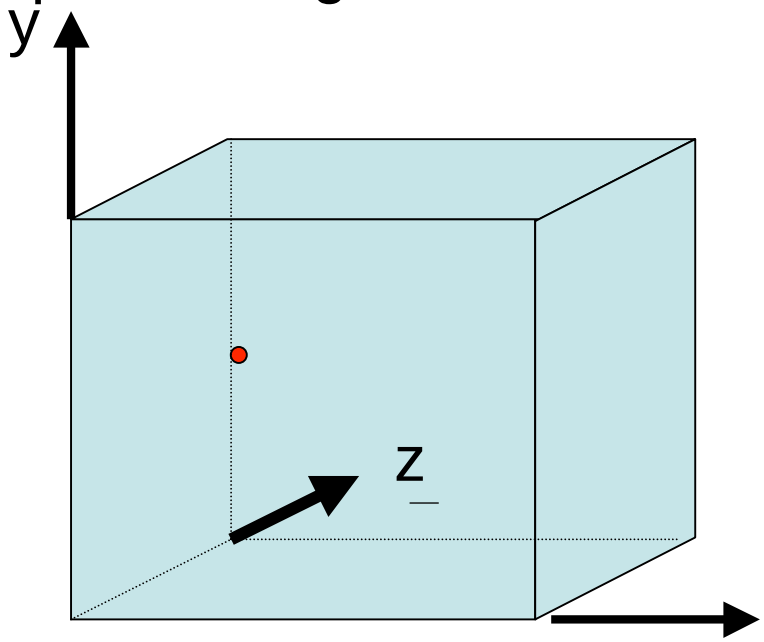
Matrix Representation

Or, writing $\rho' = \rho / \varepsilon_0$ 1-D example of matrix

$$\begin{pmatrix} -2-k^2h^2 & 1 & & & & & & \\ 1 & -2-k^2h^2 & 1 & & & & & \\ & 1 & -2-k^2h^2 & 1 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & & \ddots & & & \\ & & & & 1 & -2-k^2h^2 & 1 & \\ & & & & & 1 & -2-k^2h^2 & \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ \vdots \\ V_{n-2} \\ V_{n-1} \end{pmatrix} = \begin{pmatrix} -V_0 - h^2\rho'_1 \\ -h^2\rho'_2 \\ -h^2\rho'_3 \\ \vdots \\ \vdots \\ -h^2\rho'_{n-2} \\ -V_n - h^2\rho'_{n-1} \end{pmatrix}$$

Point Charge in a Box

Let's try it out on a point charge in a box:



Walls have V=0

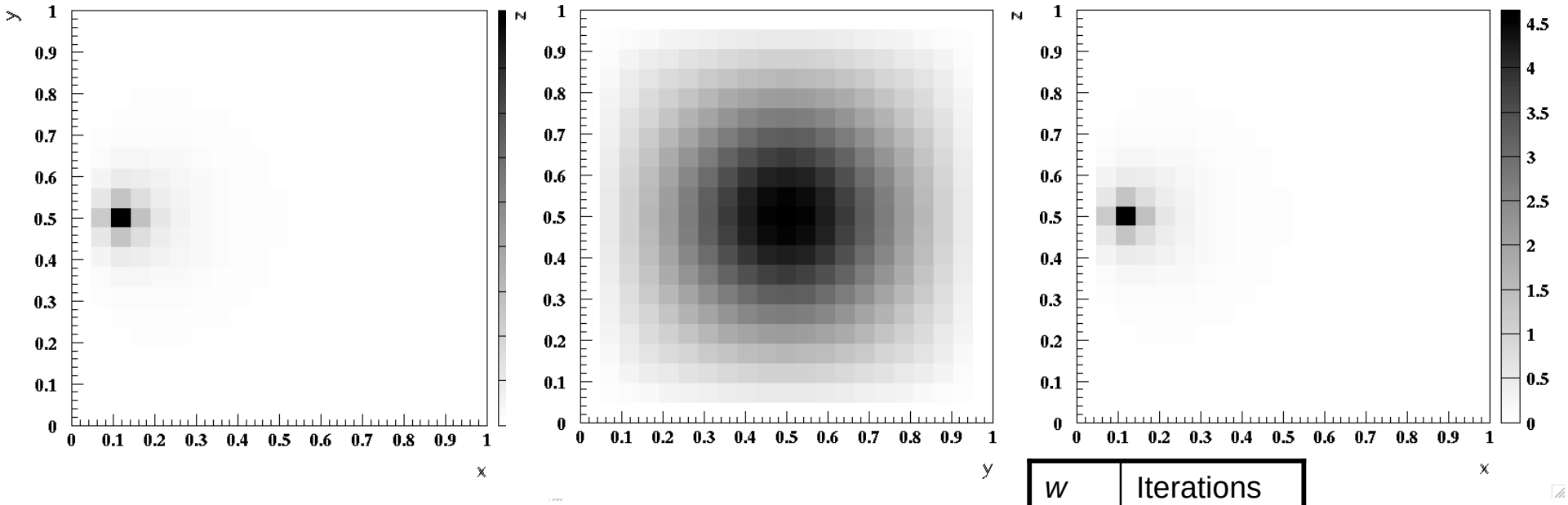
Box extends in x: 0-1,y:0-1,z:0-1.
The point charge is located at (x,y,z)=(0.101,0.501,0.501)
Take 20 intervals in each dimension (21^3=9261 grid points)

$$\iiint \frac{\rho(x,y,z)}{\varepsilon_0} dxdydz = \iiint \frac{q\delta(0.101,0.501,0.501)}{\varepsilon_0} dxdydz = 1.0$$

Point Charge in a box

Using the overrelaxation method with *w*=1.8, converge after 63 iterations (variation less than 10^{-6}).

Potential in three views

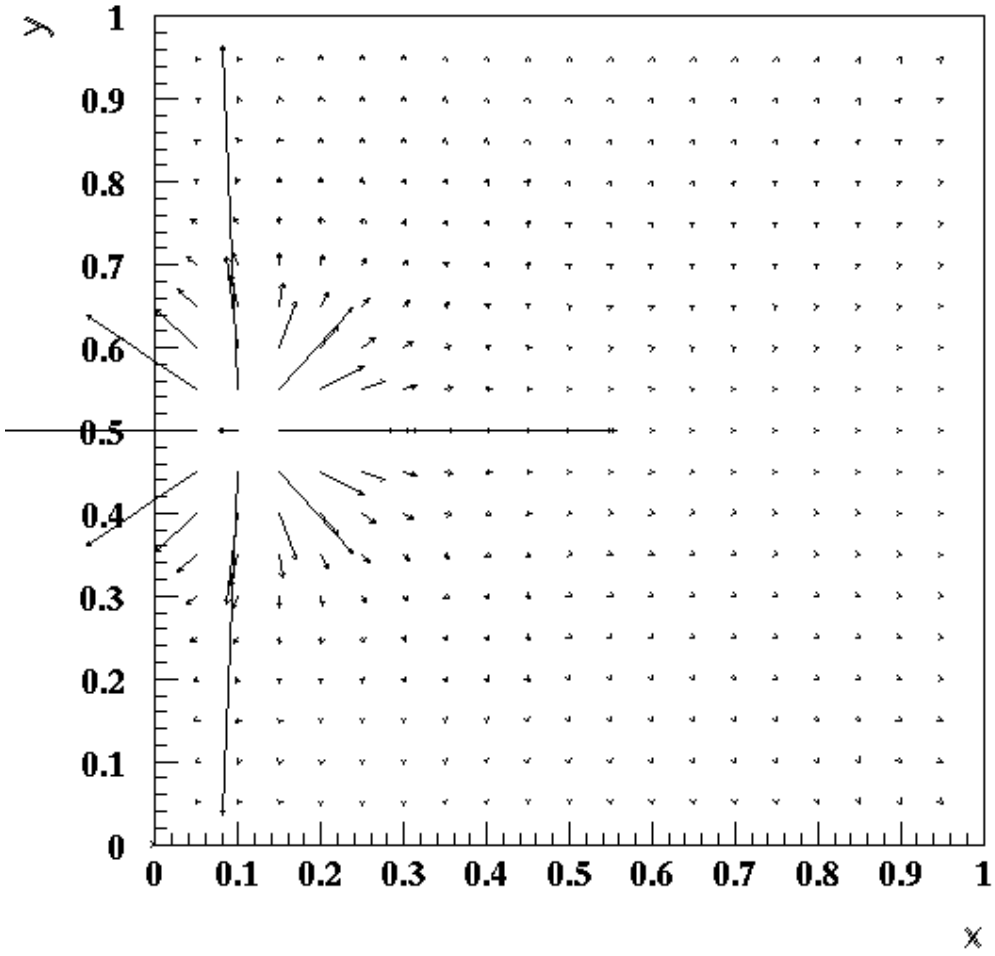


w	Iterations
1.0	306
1.2	213
1.4	141
1.6	82
1.8	63
1.9	127

Check convergence with different values of *w:*

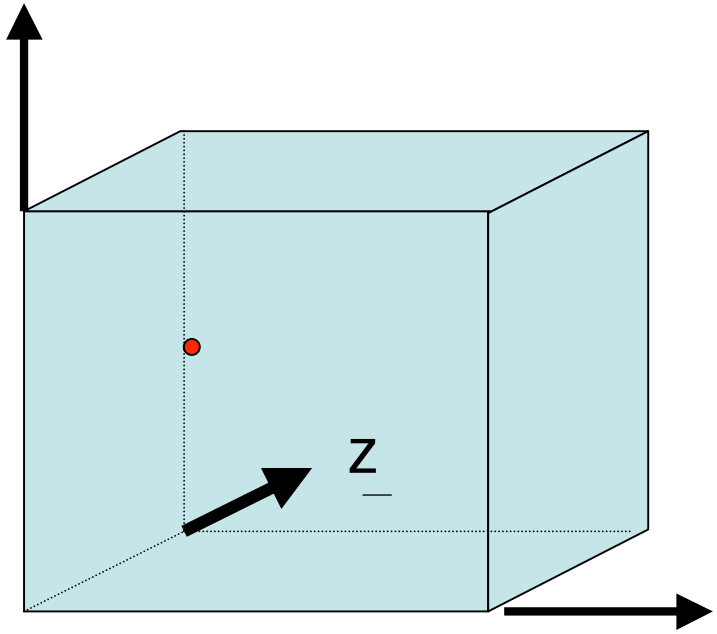
Point Charge in a Box

Electric Field



Point Charge in Box

Now we try a finer grid to see more detail.



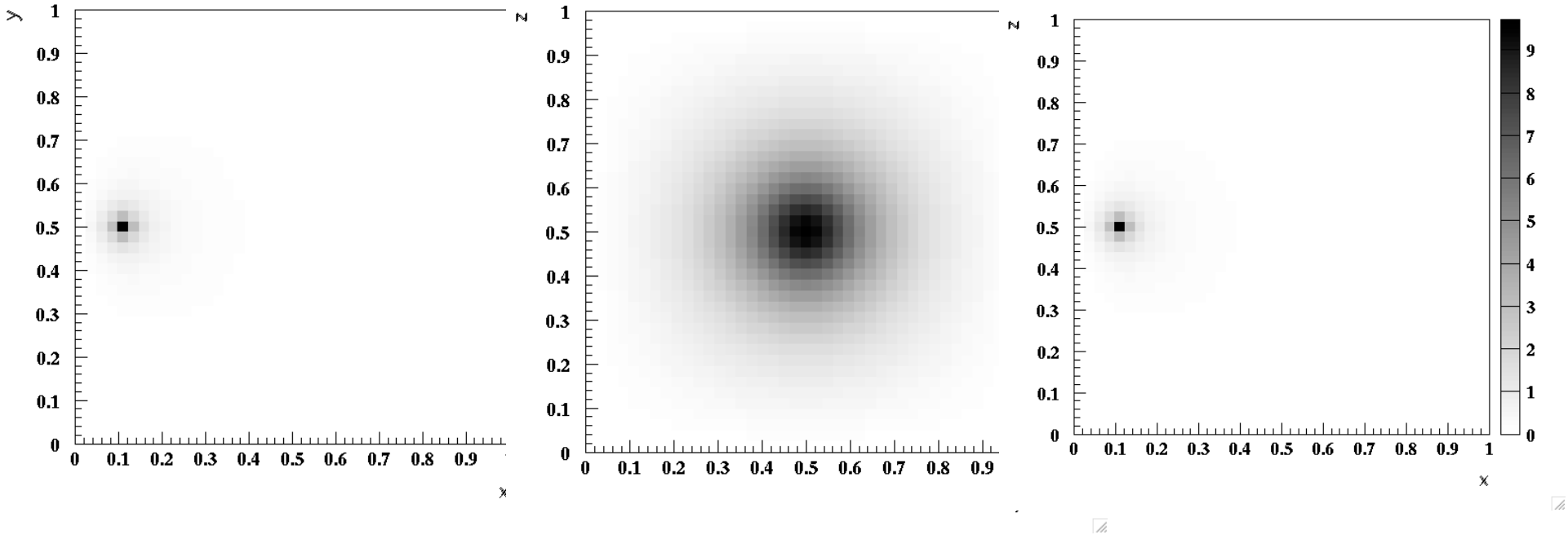
Walls have V=0

Take 40 intervals in each dimension (41^3=68921 grid points)
With *w=1.8* need 141 iterations. Calculation time increases by factor $(2^3)*(141/63)\approx 16$

Point Charge in a box

Using the overrelaxation method with *w*=1.8, converge after 141 iterations (variation less than 10^{-6}).

Potential in three views

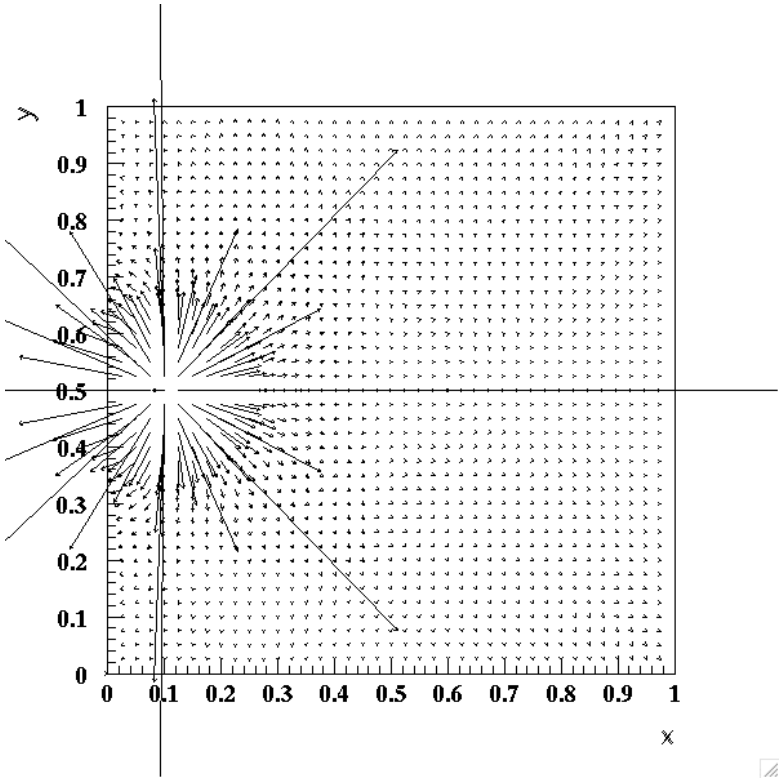


Check convergence with different values of *w:*

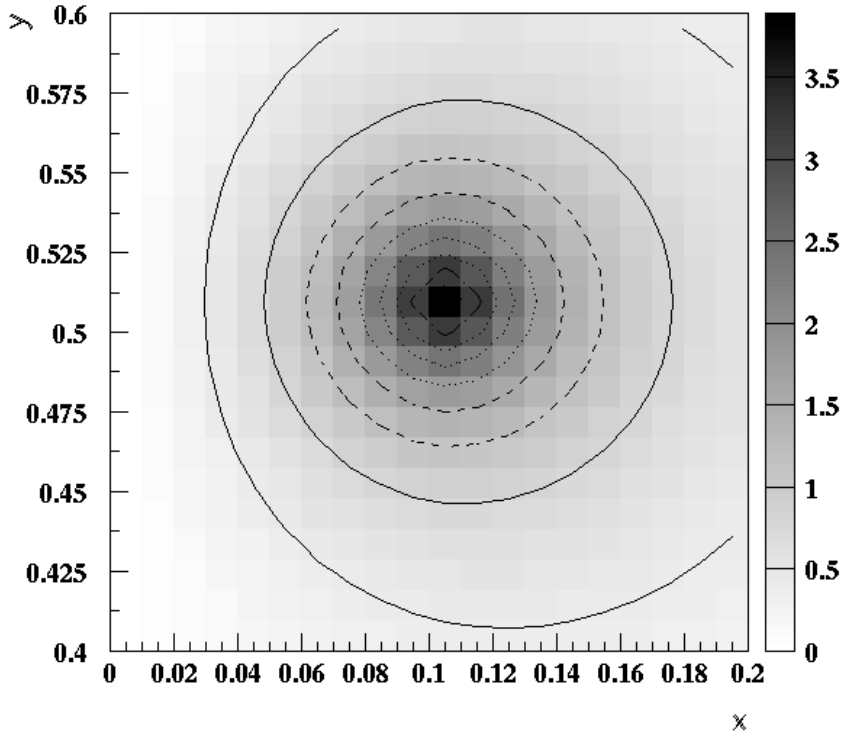
w	Iterations
1.6	300
1.8	141
1.9	137

Point Charge in a Box

Electric Field

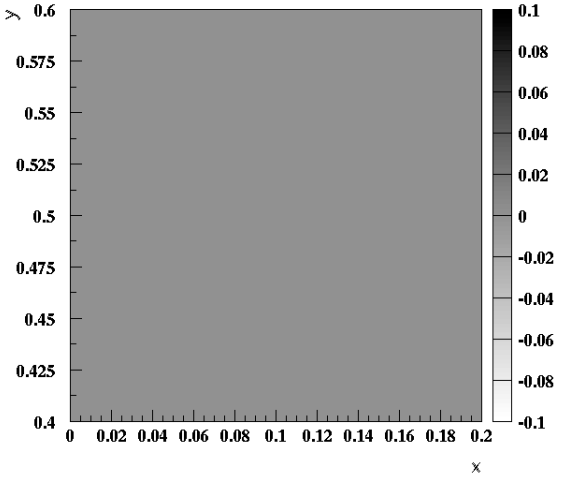


I also tried a 101^3 grid. Single precision calculation did not converge. In double precision, converged after 282 iterations.

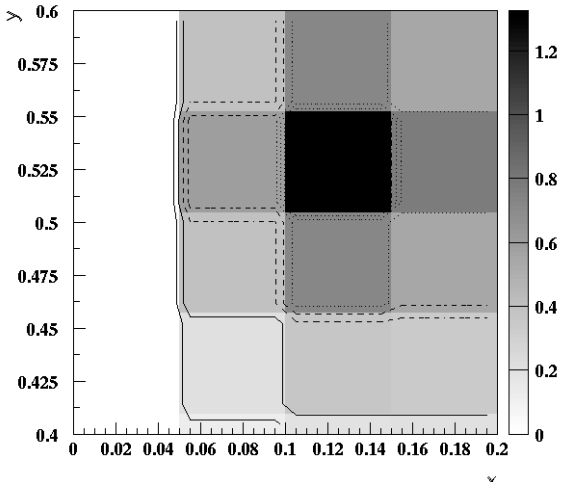


Point Charge in a Box

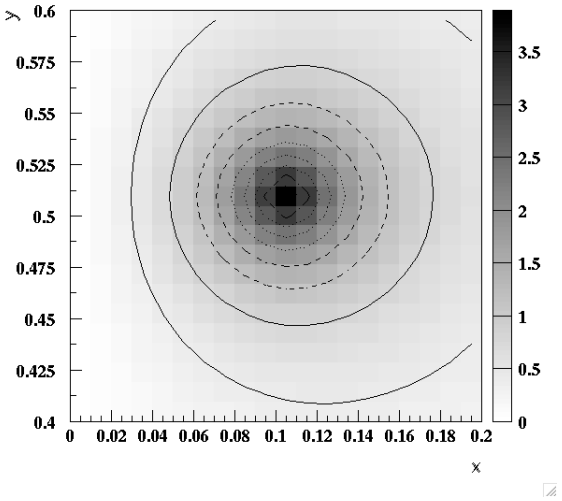
Start for 21^3 grid



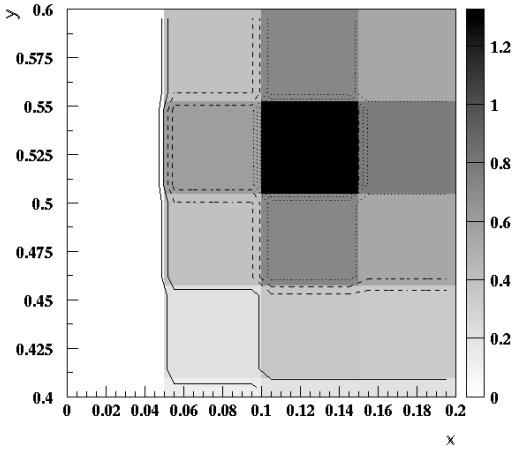
End for 21^3 grid



End for 101^3 grid



Start for 101^3 grid



Suppose have an approximate solution to the Poisson equation, f_1
Now use a coarser grid to find a correction, f_2 by calculating

$$r_1 = \nabla^2 f_1 + \frac{\rho}{\varepsilon_0} \qquad \left(\nabla^2 f = \frac{\rho}{\varepsilon_0} \right)$$

$$\nabla^2 f_2 = -r_1$$ on a coarser grid.

This solution will also not be exact, so can define another residuum

$$r_2 = \nabla^2 f_2 + \frac{\rho}{\varepsilon_0} \qquad \left(\nabla^2 f = \frac{\rho}{\varepsilon_0} \right)$$

$$\nabla^2 f_3 = -r_2$$

etc. Interpolate coarse grid corrections to fine grid.

Multigrid Methods

Another approach is to start on a coarse grid, and use the solution to provide the guess for the next iteration. Let's see how much faster our 101^3 grid converges if we start with the solution from the 21^3 grid. We try the following algorithm to set the starting values of the grid:

$$V_{101}(i,j,k) = V_{21}(i',j',k') \quad \text{if} \quad \begin{aligned} 5(i'-1) < (i-1) < 5i' \qquad & 1 < i < 101 \\ 5(j'-1) < (j-1) < 5j' \qquad & 1 < j < 101 \\ 5(k'-1) < (k-1) < 5k' \qquad & 1 < k < 101 \end{aligned}$$

w	Δ	21^3 steps	101^3 steps
1	10^{-4}		730
1	10^{-4}	117	266
1.90	10^{-4}		134
1.90	10^{-4}	88	131

} Big gain in speed

Here, overrelaxation faster.
In general, multigrid better,
but the two don't mix well.

Exercizes

1. Write a program with two point charges in a box where the walls are at potential zero. Place the two charges in the center in y,z and at x values 10% of the box size from each end wall. Solve for the potential and the electric field using the Gauss-Seidel method with overrelaxation.

2. Try the same problem with a coarse grid, which is then used to give input values for the fine grid. Compare the results to using the fine grid from the beginning.