BANACH CENTER PUBLICATIONS
VOLUME 3

ADVANCED TECHNIQUES FOR THE DIRECT, NUMERICAL SOLUTION OF POISSON'S EQUATION

ROBERT J. BARKER

Institute for Plasma Research, Stanford University, Stanford, California 94305, U.S.A.

Introduction

Anyone involved in the general field of Computational Physics will almost certainly from time to time during his career seek an optimized, efficient direct numerical solution to Poisson's Equation. This problem arises anytime we seek to determine the value of a potential field, φ, (electrostatic, magnetic, hydrodynamic, or gravitational) generated by a known distribution, ϱ, of "sources" (charges, currents, vortices, or masses). In each case, we must solve an equation of the form:

$$\nabla^2\varphi = \frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} + \frac{\partial^2\varphi}{\partial z^2} = -4\pi\varrho(x, y, z). \tag{1}$$

The accurate determination of the values of φ over a given region of space in which ϱ has been specified is generally difficult and time-consuming. As we pursue the objective of minimizing the execution time of our overall simulation of a given physical system, we almost invariably find that the problem of efficient Poisson-Equation-solving becomes our most serious obstacle. It is, therefore, to this problem of efficient, direct Poisson-solving that this presentation is dedicated.

Our Computational Plasma Physics Group at Stanford University has gained extensive experience in this area. In particular, Professor Oscar Buneman, the head of our group, has developed two techniques which have been found to be optimally effective. The first is called the Double Cyclic Reduction (DCR) technique and should be used whenever limited computer core size is a serious consideration. The second is called the Grid-Insensitive Accurate Space Derivative (GIASD) technique and should be used where optimized accuracy for a given speed is of primary consideration.

Naturally, there exist many other proven Poisson-solvers. For example, many computational physicists still use iterative techniques, such as the Successive Over-Relaxation (SOR) method ([1]). We, however, have found these iterative techniques

to be generally inefficient and frustrating. In addition, there are a number of other direct techniques. One of the most famous of these is Roger Hockney's Fourier Analysis-Cyclic Reduction (FACR) scheme ([2]).

To illustrate the DCR and GIASD techniques, we shall examine only the two-dimensional Poisson's Equation over a rectangular region. Our presentation shall be divided into three major sections. First, we shall explain the Double Cyclic Reduction scheme. Next, we shall look at the details of the Grid-Insensitive Accurate Space Derivative scheme. Finally, we include an Appendix which explains how to deal with non-rectangular regions-of-interest and which illustrates the subtlties of the "cubic splines". (These "cubic splines" play a vital role in the GIASD scheme.)

The double cyclic reduction scheme

To begin with, we assume that we are given an arbitrary source distribution, $\varrho(x, y)$, defined over a rectangular, two-dimensional region. The discreteness of our numerical methods demands that we deal only with the values of $\varrho(x, y)$ which fall on a regular two-dimensional grid. This leaves us with an array of values, $\varrho_{m,n}$, of the form:

$$\begin{array}{ll} & \left|\longleftarrow h \longrightarrow\right| \\ \uparrow & \varrho_{i-1,j-1} \quad \varrho_{i-1,j} \quad \varrho_{i-1,j+1} \\ h & \\ \downarrow & \varrho_{i,j-1} \quad \varrho_{i,j} \quad \varrho_{i,j+1} \\ & \varrho_{i+1,j-1} \quad \varrho_{i+1,j} \quad \varrho_{i+1,j+1} \end{array}$$

The finite-difference Poisson's Equation for the determination of a corresponding $\varphi_{i,j}$ array over the mesh is usually written in either a 5-point or a 9-point form. The 5-point equation reads:

$$\varphi_{i,j-1}+\varphi_{i-1,j}+\varphi_{i,j+1}+\varphi_{i+1,j}-4\varphi_{i,j} = -4\pi h^2\varrho_{i,j} \tag{2}$$

while, in the 9-point scheme with fourth-order error, Kantorovich and Krylov [3] have derived:

$$\begin{aligned} \varphi_{i-1,j+1}+4\varphi_{i,j+1}+\varphi_{i+1,j+1}+4\varphi_{i+1,j}+\varphi_{i+1,j-1}+4\varphi_{i,j-1}+ & \\ +\varphi_{i-1,j-1}+4\varphi_{i-1,j}-20\varphi_{i,j} & \\ = 6h^2\left\{(\nabla^2\varphi)_{i,j}+\frac{h^2}{12}[\nabla^2(\nabla^2\varphi)]_{i,j}\right\} & \\ = -6h^2\left\{4\pi\varrho_{i,j}+\frac{h^2}{12}[\nabla^2(4\pi\varrho)]_{i,j}\right\}. & \end{aligned} \tag{3}$$

These may be re-written respectively in the more illuminating schematic forms:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \varphi \end{bmatrix} \rightarrow -4\pi \begin{bmatrix} \varrho \end{bmatrix} \tag{4a}$$

and

$$\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} \cdot \left[\varphi - \frac{h^2}{12}\nabla^2\varphi\right] \rightarrow -4\pi \begin{bmatrix} \varrho \end{bmatrix}. \tag{4b}$$

Through the late 1960's, the concept and processes of Cyclic Reduction were extensively analyzed by Buzbee, Golub, and Nielson ([4]–[6]). This scheme was combined with the powerful Fast Fourier Transform by Roger Hockney in his Fourier Analysis-Cyclic Reduction (FACR) technique. In 1969, Professor Buneman carried Hockney's reduction scheme to its logical extension by devising the Double Cyclic Reduction (DCR) algorithm ([7]). Compared to FACR, this scheme results in a dramatic reduction in the size of the Poisson-solver's object code on the computer. This is, of course, an extremely important consideration when we are working on a computer with a very limited storage capacity.

Five-point DCR

For ease of representation, we shall consider each row of our mesh to be a vector quantity, as in (4) above. We designate the source (or "charge") vectors as $(\vec{s}_0)_j$ and the potential "vectors" as $\vec{\varphi}_j$. An odd number, $N+1$, of the rows are assumed on our over-all mesh and it is further assumed that $\vec{\varphi}_0$, $\vec{\varphi}_N$, and all $(\vec{s}_0)$ are known. The finite-difference Poisson's Equation (4a) relating the $\vec{\varphi}_j$ to the $(\vec{s}_0)$ becomes:

$$\vec{\varphi}_{j-1}-\tilde{T}_0\vec{\varphi}_j+\vec{\varphi}_{j+1} = (\vec{s}_0)_j \tag{5}$$

where $\tilde{T}_0$ is the tridiagonal matrix with coefficients $\{-1, 4, -1\}$.

The essence of cyclic reduction is illustrated by looking at the three successive equations:

$$\begin{aligned} \vec{\varphi}_{j-2}-\tilde{T}_0\vec{\varphi}_{j-1}+\vec{\varphi}_j &= (\vec{s}_0)_{j-1}, \\ \vec{\varphi}_{j-1}-\tilde{T}_0\vec{\varphi}_j+\vec{\varphi}_{j+1} &= (\vec{s}_0)_j, \\ \vec{\varphi}_j-\tilde{T}_0\vec{\varphi}_{j+1}+\vec{\varphi}_{j+2} &= (\vec{s}_0)_{j+1}. \end{aligned}$$

Multiplying the top and bottom equations from the left by $\tilde{T}_0$ and adding the three together yields a new set of equations relating all of the even rows:

$$\vec{\varphi}_{j-2}-\tilde{T}_1\vec{\varphi}_j+\vec{\varphi}_{j+2} = (\vec{s}_1)_j \tag{6}$$

where we have defined:

$$\tilde{T}_1 \equiv \tilde{T}_0^2-2\tilde{I} = (\tilde{T}_0-\tilde{I}\sqrt{2})(\tilde{T}_0+\tilde{I}\sqrt{2})$$

[i.e. -2 factors, $\tilde{T}_0-2\tilde{I}\cos\left(\frac{\text{odd}\,\pi}{4}\right)$] and

$$(\vec{s}_1)_j \equiv \tilde{T}_0(\vec{s}_0)_j+(\vec{s}_0)_{j-1}+(\vec{s}_0)_{j+1}.$$

This process may then be repeated recursively to obtain equations of the form:

$$\vec{\varphi}_{j-2k}-\tilde{T}_k\vec{\varphi}_j+\vec{\varphi}_{j+2k} = (\vec{s}_k)_j \tag{7}$$

with

$$\tilde{T}_{k+1} = \tilde{T}_k^2 - 2\tilde{I}$$

[i.e. -2^{k+1} factors, $\tilde{T}_0 - 2\tilde{I}\cos\left(\frac{\text{odd}\,\pi}{2^{k+2}}\right)$] and

$$(\vec{s}_{k+1})_j = \tilde{T}_k(\vec{s}_k)_j + (\vec{s}_k)_{j-2^k} + (\vec{s}_k)_{j+2^k}$$

for $k = 0, 1, \ldots, (\log_2 N - 1)$, where k is called the "level" of the reduction.

The appearance of the cosine terms in the central coefficients of the tridiagonal matrices is a very interesting phenomenon. It would seem to indicate some type of link with Hockney's Fourier technique but no such link has been analytically derived. This remains to be investigated.

Of course, the objective of this reduction technique is to obtain the equation:

$$\vec{\varphi}_0 - \tilde{T}_K\vec{\varphi}_J + \vec{\varphi}_N = (\vec{s}_K)_J \tag{8}$$

where $J = N/2$ (i.e. the index of the central row), $K = \log_2 N - 1$.

This equation specifies the unknown potentials of the central row in terms of the known "source" array and the known boundary row potentials. Having determined these central row values, we may then reverse the reduction process and fill in the unknown row "vectors" in successive steps by applying the formula:

$$\vec{\varphi}_j = \tilde{T}_k^{-1}(\vec{\varphi}_{j+2^k} + \vec{\varphi}_{j-2^k} - (\vec{s}_k)_j) \tag{9}$$

for $k = (\log_2 N - 1), \ldots, 0$.

The operation seems very neat and clean, but it is important to note that the central coefficient of $\tilde{T}_k$ grows very rapidly with each reduction step and is of the order 4^{2^k}. It is therefore advisable to use a revised form of the recursion relation which involves multiplications by the inverse of $\tilde{T}_k$ instead of by $\tilde{T}_k$ itself. Underflow rather than overflow then occurs, but this presents no problem since most computer systems automatically replace an underflowed number with an exact zero and then proceed with the calculations.

This revised procedure involves the formation of the variable:

$$\begin{aligned}(\vec{s}_{k+1})_j = \tilde{T}_k^{-1}[&2(\vec{s}_k)_j - (\vec{s}_{k-1})_{j-r} - (\vec{s}_{k-1})_{j+r} + \\ &+ (\vec{s}_k)_{j-2r} + (\vec{s}_k)_{j+2r} - (\vec{s}_{k-1})_{j-3r} - (\vec{s}_{k-1})_{j+3r}] + \\ &+ [(\vec{s}_k)_j - (\vec{s}_{k-1})_{j-r} - (\vec{s}_{k-1})_{j+r} + (\vec{s}_k)_{j-2r} + (\vec{s}_k)_{j+2r}]\end{aligned} \tag{10}$$

for $k = 1, \ldots, (\log_2 N - 1)$ and $r = 2^{k-1}$ with

$$(\vec{s}_1)_j = \tilde{T}_0^{-1}[2(\vec{s}_0)_j] + [(\vec{s}_0)_{j-1} + (\vec{s}_0)_{j+1}] \quad \text{for} \quad k = 0$$

at each level of the reduction. The unknown potentials at each level are then given by:

$$\vec{\varphi}_j = \tilde{T}_k^{-1}[\vec{\varphi}_{j-2r} + \vec{\varphi}_{j+2r} - (\vec{s}_k)_j] - \tfrac{1}{2}[(\vec{s}_{k-1})_{j-r} + (\vec{s}_{k-1})_{j+r} - (\vec{s}_k)_j] \tag{11a}$$

for $k = (\log_2 N - 1), \ldots, 1$ and $r = 2^{k-1}$.

For $k = 0$, we use the original equation

$$\vec{\varphi}_j = \tilde{T}_0^{-1}[\vec{\varphi}_{j-1} + \vec{\varphi}_{j+1} - (\vec{s}_0)_j]. \tag{11b}$$

Nine-point DCR

Approximately a year and a half ago, Ellen Holden, who was then a graduate student in our research group at Stanford, undertook the task of uniting the compact, efficient DCR method to the superior accuracy of the 9-point basic finite-difference algorithm (Eq. 3). The program which resulted from her work was published in March of 1975. What follows is a brief overview of those results [8].

In matrix form, the 9-point Poisson scheme may be written as:

$$\begin{bmatrix} \tilde{A} & \tilde{T} & & & \\ \tilde{T} & \tilde{A} & \tilde{T} & & 0 \\ & \tilde{T} & \tilde{A} & \tilde{T} & \\ & & \tilde{T} & \ddots & \ddots \\ 0 & & & \ddots & \tilde{T} \\ & & & \tilde{T} & \tilde{A} \end{bmatrix} \cdot \begin{bmatrix} \vec{\varphi}_1 \\ \vec{\varphi}_2 \\ \vec{\varphi}_3 \\ \vdots \\ \vec{\varphi}_N \end{bmatrix} = \begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \\ \vdots \\ \vec{u}_N \end{bmatrix} \tag{12}$$

where

$$\vec{\varphi}_i = \begin{bmatrix} \varphi_{1,i} \\ \varphi_{2,i} \\ \varphi_{3,i} \\ \vdots \\ \varphi_{M,i} \end{bmatrix},$$

i.e., the potential at each mesh point in the ith column,

$$\vec{u}_i = h^2 \begin{bmatrix} \left(\nabla^2\varphi + \frac{2h^2}{4!}\nabla^4\varphi\right)_{1,i} \\ \vdots \\ \left(\nabla^2\varphi + \frac{2h^2}{4!}\nabla^4\varphi\right)_{M,i} \end{bmatrix},$$

$$\tilde{A} \equiv \frac{1}{6h^2}\begin{bmatrix} 20 & 4 & & & 0 \\ 4 & 20 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & 20 & 4 \\ 0 & & & 4 & 20 \end{bmatrix}_{M\times M} \quad \text{and} \quad \tilde{T} \equiv \frac{1}{6h^2}\begin{bmatrix} 4 & 1 & & & 0 \\ 1 & 4 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & 4 & 1 \\ 0 & & & 1 & 4 \end{bmatrix}_{M\times M}$$

and where N = total number of mesh columns, M = total number of mesh rows.

If we now define

$$q \equiv 4\pi\varrho$$

and use a five-point discretization of $\nabla^2 q$, then the basic equation which we wish to solve is:

$$[8 \text{ points} - 20\varphi_{i,j}] = \frac{h^2}{2}[8q_{i,j} + q_{i+1,j} + q_{i-1,j} + q_{i,j+1} + q_{i,j-1}]. \tag{13}$$

Paralleling the analysis used in Buneman's 5-point DCR scheme, we write:

$$\begin{aligned} &\tilde{A}\vec{\varphi}_1+\tilde{T}\vec{\varphi}_2 = \vec{u}_1, \\ &\tilde{T}\vec{\varphi}_{j-1}+\tilde{A}\vec{\varphi}_j+\tilde{T}\vec{\varphi}_{j+1} = \vec{u}_j, \quad \text{for} \quad j = 2, 3, \ldots, r-1, \\ &\tilde{T}\vec{\varphi}_{r-1}+\tilde{A}\vec{\varphi}_r = \vec{u}_r, \quad \text{for} \quad r = 2^{k+1}-1, \end{aligned} \tag{14}$$

where k is the total number of cyclic reductions which will be performed. Multiplying through by $\tilde{T}^{-1}$ in equations (14) from the left yields the first reduction level:

$$\vec{\varphi}_{j-2}+[2\tilde{I}_s-(\tilde{T}^{-1}\tilde{A})^2]\vec{\varphi}_j+\vec{\varphi}_{j+2} = \tilde{T}^{-1}\vec{u}_{j-1}+\tilde{T}^{-1}\vec{u}_{j+1}-(\tilde{T}^{-1}\tilde{A})\tilde{T}^{-1}\vec{u}_j$$

where $j = 2, 4, \ldots, r-1$ and $\vec{\varphi}_0 \equiv \vec{\varphi}_{r+1} \equiv \vec{0}$, $\tilde{I}_s \equiv$ the identity matrix of order s, $s \equiv$ the number of interior mesh rows.

Now define

$$\tilde{P} \equiv \tilde{T}^{-1}\tilde{A}$$

and

$$\begin{aligned} (\vec{u}_1)_j &= \tilde{T}^{-1}\vec{u}_{j-1}+\tilde{T}^{-1}\vec{u}_{j+1}-\tilde{P}\tilde{T}^{-1}\vec{u}_j \\ &= \tilde{P}_1\tilde{P}^{-1}\tilde{T}^{-1}\vec{u}_j+\tilde{T}^{-1}\vec{u}_{j-1}+\tilde{T}^{-1}\vec{u}_{j+1}-2\tilde{P}^{-1}\tilde{T}^{-1}\vec{u}_j \end{aligned}$$

where $\tilde{P_1} \equiv 2I_s-\tilde{P}^2$.

If we further define:

$$(\vec{p}_1)_j \equiv \tilde{P}^{-1}\tilde{T}^{-1}\vec{u}_j = \tilde{A}^{-1}\vec{u}_j$$

and

$$(\vec{g}_1)_j \equiv \tilde{T}^{-1}\vec{u}_{j-1}+\tilde{T}^{-1}\vec{u}_{j+1}-2(\vec{p}_1)_j$$

then

$$(\vec{u}_1)_j = \tilde{P}_1(\vec{p}_1)_j+(\vec{g}_1)_j \quad \text{for} \quad j = 2, 4, \ldots, r-1$$

so that:

$$\begin{aligned} \vec{\varphi}_{j-2}+\tilde{P}_1\vec{\varphi}_j+\vec{\varphi}_{j+2} &= (\vec{u}_1)_j \quad \text{for} \quad j = 2, 4, \ldots, r-1 \\ &= \tilde{P}_1(\vec{p}_1)_j+(\vec{g}_1)_j. \end{aligned}$$

Similarly, after "n" reductions, we have:

$$\vec{\varphi}_{j-2^n}+P_n\vec{\varphi}_j+\vec{\varphi}_{j+2^n} = \tilde{P}_n(\vec{p}_n)_j+(\vec{g}_n)_j \tag{15}$$

where

$$j = m2^n \quad \text{and} \quad m = 1, 2, \ldots, 2^{k-n+1}-1$$

where:

$$\tilde{P}_n = 2\tilde{I}_s-(\tilde{P}_{n-1})^2,$$

$$\begin{aligned} (\vec{p}_n)_j &= (\vec{p}_{n-1})_j-\tilde{P}_{n-1}^{-1}[(\vec{p}_{n-1})_{j-2^{n-1}}+(\vec{p}_{n-1})_{j+2^{n-1}}-(\vec{g}_{n-1})_j], \\ (\vec{g}_n)_j &= (\vec{g}_{n-1})_{j-2^{n-1}}+(\vec{g}_{n-1})_{j+2^{n-1}}-2(\vec{p}_n)_j, \end{aligned}$$

with

$$(\vec{p}_{n-1})_0 \equiv (\vec{p}_{n-1})_{2^{k+1}} \equiv (\vec{g}_{n-1})_0 \equiv (\vec{g}_{n-1})_{2^{k+1}} \equiv \vec{0}.$$

The remaining steps of the solution process parallel those of the five-point DCR scheme above. We shall not repeat these details.

Grid-Insensitive Accurate Space Derivative method

When one is dealing with the problem of simulating a plasma, the work is divided into two major sections. First, given an initial charge/plasma distribution, we must solve Poisson's Equation to determine the electric potential on a given array of grid points. Knowing our potential function over the region-of-interest, we then interpolate between the known values to determine the slope of the function (i.e. the electric field) at the positions of our individual charged particles. It is, of course, this value of the electric field at the particle's position which gives us, via the Lorentz Law, the force acting on the particle. In this manner, we know how to move the individual particles over the given time-step.

As a general rule, significantly more computing time is required to Poisson-solve for the value of the electric potential at a given data point than is required for field and force calculations for each particle. It is, therefore, generally desirable to use a coarse mesh for Poisson-solving, while employing a considerably finer grid for specifying particle positions. The question of precisely *how* coarse a Poisson grid we should employ for optimal efficiency, is answered by considering the total number of charged particles we are using in the simulation. If we are working with a very large number of particles, then the cost of interpolating electric field values to each particle position could well outweigh the cost of simply using a finer Poisson-grid with a more simple, low-order particle-moving scheme.

This highlights an important point; in the final analysis, our major interest is to determine the slope of the potential function at the particle positions. Poisson-solving over a fixed, regular grid is an artificial, intermediate step. Our Poisson-solver should therefore be designed for optimal ease-of-transition to random electric field interpolations. This was the primary consideration which led to the development of the Grid-Insensitive Accurate Space Derivative (GIASD) technique [9].

The essence of GIASD rests in the use of cubic "splines" (see Appendix II) for distributing the charge over the Poisson-grid and for interpolating the electric field, combined with the use of Fourier Analysis to actually carry out the Poisson-solving. This combination results in an extremely accurate, straightforward technique.

The heart of this method, namely its use of Fourier harmonics as the working basis functions, is derived from the Accurate Space Derivative (ASD) technique developed by Dr. Jeno Gazdag of the IBM Research Center in Palo Alto, California. We use Fourier harmonics for the following reasons:

1) They guarantee that our approximated functions will have infinite smoothness. All derivatives are continuous and interpolation will be efficient.

2) There is translational invariance. This implies that the results do not depend locally on whether a particle is on a grid line or somewhere in between because, in reality, there is no longer any grid.

3) The harmonics are orthogonal and remain separable in terms which are quadratic. This is an important consideration when one is using an Action Principle to derive the working equations of the physical system.

4) Electrodynamics is traditionally analyzed in Fourier space.

5) Differentiations become multiplications by k_x and k_y.

6) If the cut-off of the spectrum is made according to the magnitude $|\vec{k}| = k_{\max}$ of the $\vec{k}$-vector, then the function representation in real space will be locally isotropic.

Our method of procedure for implementing GIASD may be described as follows. First, we are given a distribution of charges at the locations. Normally, our first concern is how to directly distribute these charges onto the given Poisson-solving grid. However, since we are Poisson-solving in Fourier space, this does not concern us now. Rather, we immediately Fourier transform the charge distribution to obtain the charge coefficients,

$$q_{\vec{k}} = \sum_{\mu} \exp(-i\vec{k}\vec{r}_{\mu}).$$

Now, to preserve isotropy in real space, we introduce the isotropic "form-factor", $P(|\vec{k}|)$ to obtain, instead:

$$q_{\vec{k}} = P(|\vec{k}|) \sum_{\mu} \exp(-i\vec{k} \cdot \vec{r}_{\mu}). \tag{16}$$

The question now arises as to what shape our form-factor, $P(|\vec{k}|)$, shall take. If we were to let $P = \Omega$ then the transform into real space would yield charges with cross-sectional density profiles in the form of sinc functions:

Fig. 1

This is undesirable because this endows all of our charges with negative "haloes", and an attenuation only like $1/r$ (from the center). (Thus, the attenuation of the charges would be as weak as the attenuation of the potential fields which the charges create.) After some experimentation, it appeared that we should make P gaussian in form:

$$P = \exp(-\tfrac{1}{2}R^2k^2).$$

This transforms in real space into another gaussian

$$\exp(\tfrac{1}{2}r^2/R^2)$$

which is an aesthetically pleasing distribution cross-section to give to our charges. In practise, ease-of-calculation dictates that we use the bell-shaped cubic spline curves for P instead of using strict gaussians. (N. B.—Coincidentally, the cubic spline curve is the curve which represents the 4-random walk statical distribution function.) This curve transforms in real space to a very similar curve which has only small positive "haloes", the first of which has a height only one-five hundreth that of the center peak. These haloes are attenuated like $(1/r^4)$, which is very desirable.

Having chosen a $P(|\vec{k}|)$, our charge distribution transformation is complete and we are ready to attack Poisson's Equation. In Fourier space, this is very easy. The equation reduces to simply:

$$k^2\varphi_{\vec{k}} = q_{\vec{k}} = P(|\vec{k}|) \sum_{\mu} \exp(-i\vec{k} \cdot \vec{r}_{\mu}). \tag{17}$$

The only annoying aspect of this equation is the prospect of having to evaluate the quantity

$$\exp(-i\vec{k} \cdot \vec{r}_{\mu})$$

for every particle position, $\vec{r}_{\mu}$, at every time step. To avoid this, we call upon our friends, the cubic splines, and use the approximation

$$\exp(ikx) \approx \sum_{j=[x]-1}^{[x]+2} g_j S_3(x-j), \tag{18}$$

where $[x]$ is defined as the largest integer contained in x. For this expression (18), we have the recurrence relation

$$g_{j-1} + 4g_j + g_{j+1} = 6\exp(ikj). \tag{19}$$

This formula (19) is solved by

$$g_j = \frac{3\exp(ikj)}{2+\cos(k)}. \tag{20}$$

So that we are left with

$$\exp(ikx) \approx \sum_{j} \frac{3\exp(ikj)S_3(x-j)}{2+\cos(k)}. \tag{21}$$

Plugging equation (21) into our Poisson's Equation (17), we obtain in two-dimensional form

$$k^2\varphi_{\vec{k}} = \frac{9P(|\vec{k}|)}{(2+\cos(k_x))(2+\cos(k_y))} \sum_{j}\sum_{n} \exp(-ik_x j - ik_y n) \times \times \sum_{\mu} S_3(x_\mu - j)S_3(y_\mu - n) \tag{22}$$

$$= [\ldots] \text{ times the discrete Fourier Transform of the charge array.}$$

Given the Fourier components of our complete potential function, we might be tempted at this point to blindly push ahead and back-transform to obtain the real φ function values on the given array of data points and then be left to worry about putting together an interpolation technique to obtain values between grid points, at the positions of the charges. Instead of this, let us pause for a moment to consider the situation. Our ultimate objective is to calculate how our charges move during a given time step. For this, we need the values of the electric field, $\vec{E}(\vec{r}_{\mu})$, at the charge positions, $\vec{r}_{\mu}$. If we have $\vec{E}(\vec{r}_{\mu})$, then we need merely apply:

$$\ddot{\vec{r}}_\mu = \frac{q_\mu}{m}\,\vec{E}(\vec{r}_\mu).$$

But, in one-dimension

$$E(x_\mu) = -\left.\frac{\partial\varphi}{\partial x}\right|_{x_\mu}.$$

Since we have Fourier analyzed φ, we have

$$E = \sum_j ik_j P(k_j)\varphi_{k_j}\exp(-ik_j x_\mu)$$

which, upon again employing spline interpolation to the harmonics, becomes

$$\begin{aligned}(23)\qquad E &= \sum_j ik_j P(k_j)\varphi_{k_j}\sum_n \frac{3\exp(ik_j n)}{2+\cos(k_j)}\cdot S_3(x_\mu - n)\\ &= \sum_n\sum_j ik_j\left\{\frac{3P(k_j)}{2+\cos(k_j)}\varphi_{k_j}\right\}\exp(ik_j n)\cdot S_3\\ &= \sum_n\sum_j ik_j\{\varphi_{\text{eff}}\}_{k_j}\exp(ik_j n)\cdot S_3(x_\mu - n).\end{aligned}$$

From equation (23), we may say that the Fourier components of the spline coefficients of the *effective* potential are simply

$$\begin{aligned}(24)\qquad \{\varphi_{\text{eff}}\}_{k_j} &= \left(\frac{3P}{2+\cos(k_x)}\right)\left(\frac{3P}{2+\cos(k_y)}\right)\varphi_{\vec{k}}\\ &= \left(\frac{9P(|\vec{k}|)}{k(2+\cos(k_x))(2+\cos(k_y))}\right)^2 \times \begin{pmatrix}\text{The discrete Fourier}\\ \text{Transform of the}\\ \text{charge array}\end{pmatrix}.\end{aligned}$$

If we now take the discrete Fourier synthesis of these Fourier components:

$$\sum_j\sum_n \{\varphi_{\text{eff}}\}_{\vec{k}}\exp(ik_x j + ik_y n)$$

what we obtain are then the effective real-space potential coefficients for the complete array of cubic splines centered on each interpolation grid point. We may pretabulate the slopes of the splines to whatever degree of resolution we desire. Thus, the knowledge of these potential spline coefficients affords us a very quick, direct method for determining the electric field at a given charge location. This, in turn, tells us how to move the given charges. This completes our simulation for an individual time step.

We outline the complete procedure again as follows:

1) We are given an array of charges in real space.

2) We assume that each charge has a density distribution in the shape of a cubic spline.

3) Now execute a finite Fourier analysis of this shaped-charge array.

4) Solve Poisson's Equation in Fourier space. This entails the formation of the Fourier components of the spline coefficients of the *effective* potential by multiplying each charge coefficient by

$$\left(\frac{9P(|\vec{k}|)}{k(2+\cos(k_x))(2+\cos(k_y))}\right)^2.$$

5) Execute a finite Fourier synthesis of these products to obtain the complete array of cubic spline coefficients for the *effective* potential.

6) Apply direct spline interpolation to determine how to move the charges to their new positions for the next time step.

Appendix I — Treatment of irregular boundaries [10], [11]

It is important to note that both DCR methods are designed for the solution of Poisson's Equation over a rectangular grid with straight rectangular boundaries, over which boundaries the potential is known. However, irregular, non-rectangular boundaries can still be dealt with using DCR if we simply introduce the straightforward device of employing a "Capacitance Matrix". This is accomplished as follows:

Suppose we are given a boundary (normally an electrically-conducting boundary) over which the potential is specified:

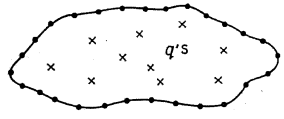

Fig. 2

$\varphi_i = (\varphi_i)_0$ at N points along the given boundary.

Inside the boundary, there is a given, known charge distribution. Our first step is to surround the given boundary with a rectangular boundary as tightly as possible:

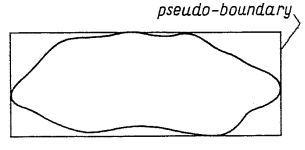

Fig. 3

We place no charges between the two boundaries. Now we use any direct method to solve the problem with the given q's, ignoring the real irregular inner boundary.

Next, we compare the φ's thus calculated for the N points on the inner boundary with the given values of φ and make a record of the N differences:

$$\Delta\varphi_1, \Delta\varphi_2, \Delta\varphi_3, \Delta\varphi_4, \ldots, \Delta\varphi_N.$$

At this point, we calculate the matrix, $\tilde{P}$, whose elements, $P_{i,k}$, are the potentials at the kth inner boundary data point which is created by placing a unit charge on the ith inner boundary data point. The "capacitance matrix", $\tilde{C}$, is then

$$\tilde{C} \equiv \tilde{P}^{-1}.$$

Thus, the creation of this matrix requires N Poisson Equation solves and one matrix inversion. Now we calculate the *effective* charges, q_i', given by

$$q_i' = \sum_k C_{ik} \Delta\varphi_k$$

which must be placed at the inner boundary data points to make $\varphi = \varphi_0$ where required. Now, we solve again; this time using the outer rectangular boundaries, but leaving the q_i' in place over the inner boundary. The solution thus obtained is valid everywhere inside the inner boundary.

Thus, for each time step, the $\Delta\varphi_i$'s and, from these, the q_i''s must be recalculated. The original "capacitance matrix", however, need only be calculated once, at the beginning of our computer run, and then used throughout the rest of the problem (assuming no boundary deformation).

Appendix II — Cubic "splines"

The function which we define as a cubic "spline", may sometimes be referred to by other authors as cubic "shape function" or as a cubic "interpolant". Whatever the name, we specify it by the equation:

$$S_3(x) = \begin{cases} \frac{2}{3} - \frac{1}{2}x^2(2-|x|) & \text{for} \quad |x| \leqslant 1, \\ \frac{1}{6}(2-|x|)^3 & \text{for} \quad 1 \leqslant |x| \leqslant 2, \\ 0 & \text{for} \quad 2 \leqslant |x|, \end{cases}$$

where the unit of length in x is defined by the spacing between grid data points. The curve is pictured in Figure 4 below.

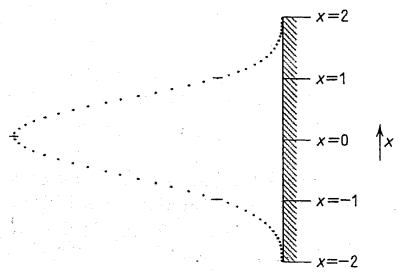

Fig. 4

In GIASD, these splines are used to approximate Fourier harmonics. It is, therefore, important to see just how well they do behave as sinusoid interpolants. Such a test has been completed and two examples of the results are pictured below. In the first case (Figure 5), a data point spacing of $5\pi/6$ is used. We see that the splines soften the erratic nature of the function and refuse to swing out as far as the pure

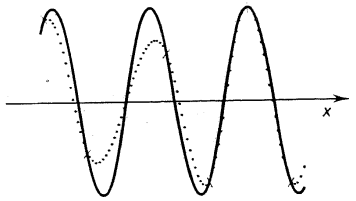

Fig. 5

sinusoid fitted through all the data points. On the other hand, if we space our data points by $\pi/2$ and specify them at the extremes and node points of the sinusoid, then the true curve and the interpolated curve are virtually indistinguishable.

Acknowledgements

I wish to gratefully acknowledge the assistance of my Research Supervisor, Professor Oscar Buneman, in the preparation of these lectures. My thanks also go to Drs. Wakulicz and Moszyński of the Banach Center and to Prof. Jaworski of the Warsaw Polytechnic University for the kindness and hospitality they showed towards me during my brief stay in the Polish capital.

These lectures were undertaken with the support of the United States National Science Foundation (NSF Grant ENG73-03722).

References

[1] R. W. Hockney, *The potential calculation and some applications, Methods in Computational Physics*, Volume 9, edited by Berni Alder, Academic Press, New York 1970, pp. 164–167.
[2] Ibid., pp. 146–155.
[3] L. V. Kantorovich and V. I. Krylov, *Approximate Methods of Higher Analysis*, Interscience Publishers Inc., New York 1958.
[4] B. L. Buzbee, G. H. Golub, and C. W. Nielson, *The method of odd/even reduction and factorization with application to Poisson's equation*, Stanford University Computer Science Department (SUCSD) Technical Report No. 128 (April 1969).
[5] —, —, —, *The method of odd/even reduction and factorization with application to Poisson's equation*, Part II, SUCSD Technical Report No. 155 (March 1970).
[6] —, —, —, *On direct methods for solving Poisson's equations*, SIAM Journ. Numer. Anal. 7, 4 (Dec. 1970).
[7] O. Buneman, *A compact non-iterative Poisson solver*, Stanford University Institute for Plasma Research (SUIPR) Report No. 294 (May 1969).

[8] Ellen Holden, *Solution of the nine-point Poisson operator by cyclic reduction*, SUIPR Report No. 619 (March 1975).
[9] O. Buneman, *Variationally optimized, grid-insensitive vortex tracing*, Proceedings of the Fourth International Conference on Numerical Methods in Fluid Dynamics, Springer-Verlag, Berlin (June 1974).
[10] J. Allan George, *The use of direct methods for the solution of discrete Poisson equation on non-rectangular regions*, SUCSD Technical Report No. 159 (June 1970).
[11] B. L. Buzbee, *et. al.*, *The direct solution of the discrete Poisson equation on irregular regions*, SUCSD Technical Report No. 195 (Dec. 1970).

Presented to the Semester
Mathematical Models and Numerical Methods
(February 3–June 14, 1975)

BANACH CENTER PUBLICATIONS
VOLUME 3

RECENT RESULTS AND OPEN PROBLEMS IN ANALYTIC COMPUTATIONAL COMPLEXITY

J. F. TRAUB

Departments of Computer Science and Mathematics, Carnegie-Mellon University, Pittsburgh, Pa. 15213, USA

There are many types of computational complexity depending on the model being studied. These include algebraic, combinatoric, analytic and parallel complexity. Recent progress in algebraic complexity is summarized by Borodin and Munro [1] and the state of the art in analytic complexity is covered in Traub [8]. Material on parallel complexity may be found in Traub [7] and Heller [4].

Although we will focus here on analytic complexity, we begin by reporting a recent algebraic result which was stimulated by a question in analytic complexity.

Given a power series for f we wish to calculate the first n terms of the reverse power series. The classical algorithms for doing this are at least $O(n^3)$. Brent and Kung [2] have shown that this problem can be solved with complexity $O((n\log n)^{3/2})$. No non-trivial lower bound is known for this problem. They also show that the problem of reversion of power series is equivalent to the problem of calculating the first n terms of the polynomial resulting from the composition of two polynomials of degree n.

Finally they show (Brent and Kung [3]) that the polynomial consisting of the first n terms of the reverse power series can be evaluated in $O(n\log n)$. This result gives us an upper bound on the combinatory cost of certain iterations.

We turn to recent results and open problems in analytic computational complexity, or more specifically in iterative computational complexity.

Let f be a nonlinear operator, $f\colon D \subset B_1 \to B_2$ where B_1 and B_2 are two Banach spaces. Let α be a simple zero of f and let x_i be a sequence of approximations for α which are generated by an algorithm Φ. We shall consider here only implicit problems, that is problems where only certain functionals of f are available.

Let e_i represent some measure of the error of x_i. For example e_i might represent

$\|x_i-\alpha\|$, absolute error,