JOURNAL OF COMPUTATIONAL PHYSICS 11, 250–268 (1973)

Subgrid Resolution of Flow and Force Fields

O. BUNEMAN*

Institute for Plasma Research, Stanford University, Stanford, California 94305

Received January 2, 1972

By means of spline fitting, both while converting a Lagrangian distribution of field sources onto an Eulerian grid, and while interpolating the field from its Eulerian grid to the Lagrangian positions of the responding elements, one achieves effective resolution down to about one-eighth of a grid mesh unit. One also avoids discontinuities and noise due to cell boundary crossings.

I. HYBRID EULERIAN–LAGRANGIAN SCHEMES

The following mathematically equivalent tasks in computational physics are to be tackled:

1. Determination of the flow due to a distribution of vorticity,
2. Determination of the magnetic field due to a distribution of current,
3. Determination of the electric field due to a distribution of charge,
4. Determination of the gravitational field due to a distribution of mass.

These four problems will be treated in two dimensions.

Generalization to three dimensions is not trivial in applications 1 and 2 [1]. While in applications 3 and 4 such generalization is trivial, it remains expensive computerwise and will not be dealt with here. Only straight, parallel vortex or current filaments, and straight parallel rods of charge or mass will be considered as sources of fields. (The word "source" is not to be understood in just the fluid dynamical sense here, but generally in the sense of "origin" or "cause". The source material, quantitatively, is *circulation* in the fluid dynamical application.) Mathematically, all four problems are the same: the interaction potential is $\ln r^2$ in each case.

The fluid problem is conventionally treated by fully Eulerian methods. The continuous vorticity and the continuous streamfunction are recorded over a grid, dynamical and interaction laws are written as partial differential equations only,

* National Aeronautics and Space Administration Grant NGL-05020-420.

and finite difference schemes are used that require function records only at the discrete gridpoints; subgrid resolution is never required. The interaction potential is not used as such; the ∇^2 operator (for which $\ln r^2$ is the Green's function) is inverted by one of the many finite difference algorithms now available [2].

There are some examples of fully Eulerian schemes being used successfully also in the electrostatic problem. The two-dimensional configuration space is then expanded into four-dimensional phase-space, and Boltzmann's equation, in addition to Poisson's equation, is solved over a grid [3].

There are a few instances, also, of fully *Lagrangian* schemes being used for the solution of these problems [4]: the action upon each element (vortex, filament, charge, particle) due to all the other elements is then calculated by summing all the interaction potentials $\ln |\mathbf{r}_i - \mathbf{r}_j|^2$. The economy of this scheme is discussed in [2] and found to be unfavorable for most cases.

By far most popular is the hybrid Lagrangian/Eulerian scheme where the elements are discrete and in unrestricted positions. They are made to interact with each other through a potential recorded on a regular, fixed mesh. This method is beginning to gain popularity also in the simulation of vortical inviscid flow (Refs. [1, 5]).

It is here that one requires subgrid resolution, both when the source distribution is to be converted from Lagrangian to Eulerian and when the resulting potential or field is to be converted back from Eulerian to Lagrangian for the purpose of getting the local action on the responding elements. In other words, one requires the field at points other than meshpoints due to sources which are not necessarily located at mesh points. A finite-difference Poisson solver is used, in place of the direct interaction kernel ln (distance^2), between the Eulerian source and potential distributions.

II. The Interaction Kernel Array

Before concerning ourselves with the problems of interpolation, let us consider the exceptional case where no interpolations are necessary: a "particle" located at a gridpoint acts on another located at a gridpoint. Ideally, the potential should be $\ln[(I_x - I'_x)^2 + (I_y - I'_y)^2]$ where integers I_x, etc., denote coordinates of gridpoints. (The mesh will be assumed uniform and the same in both dimensions; the spacing will be taken as unit of length.)

However, there exists no simple fast algorithm which generates a potential array from a source array exactly in conformity with the logarithmic formula: after all, what should such an algorithm yield at the gridpoint where one single isolated source is located?

We must be content with something less perfect. Now the nine-point operator, when inverted, yields a potential array in response to a single isolated source which is fairly close to the logarithm except at the origin where, of course, it is finite. A numerical experiment with one's favorite Poisson solver will confirm this, or one can look at Table I, obtained by analytical inversion of the nine-point operator [6], following the steps given in [7] for the five-point operator.

TABLE I

Potentials due to a solitary negative unit (Gaussian) charge at top left. Bracketed: deviations from exact logarithmic potentials.

1.0472	3.6276 (−.0117)	5.0208 (−.0048)	5.8355 (−.0011)
	4.3392 (+.0067)	5.2502 (+.0014)	5.9419 (−.0000)
		5.7195 (+.0007)	6.2046 (+.0003)
			6.5299 (+.0002)

The calculations to be presented here were originally performed using a five-point operator, with somewhat less favorable results. In view of the quadratic spline fitting which is to be employed in the accumulation of the source distribution, one needs a Poisson solver that handles a quadratic source distribution correctly. A nine-point operator does this. We have choosen the version [22]:

$$\nabla^2\Phi \approx \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}\left(\Phi - \frac{1}{12}\nabla^2\Phi\right).$$

One ought to employ Lewis' variational method [8, Eq. 63)] and find the optimal Poisson operator to go with parabolic splines. Such an operator is simple only in Fourier transform space: the one-dimensional Fourier version is given by Langdon [9]. It should lead to better overall performance in numerical simulations without necessarily making the approximation to the logarithm any better, but at the moment our main concern will be with the value of the interaction kernel (or "Green's function") at the origin. In fact, most physicists would "buy" a numerical experiment with an interaction kernel (between elements at *different* locations) that is not exactly logarithmic, provided this kernel is consistently adhered to. However, the finite self-energy, in place of the ideal infinite self-energy, requires more justification and physical interpretation.

At first glance one might see a major discrepancy here between the physical reality and the computer model which purports to simulate it. But, in fact, there is a virtue in the finite value of interaction potential at the origin. In most applications the sources carried in the computer model represent extended sources rather than singular sources: In application (2), the current filaments are finite diameter wires. In application (4), the massive particles may be finite diameter stars. In application (3), as well as in some of the gravitational applications, each particle carried in the computer is a "superparticle," standing in for a very large number of smaller physical particles that are spread over a certain area or volume (the "charge clouds" introduced in Ref. [10]. In application (1) each vortex carried in the computer is really a marker in the fluid and should be considered as a finite element of circulation associated with a fluid element of finite area.

If, now, one calculates the true potential created by a finite circular extended source with uniform interior density, one obtains a parabolic potential profile in the interior, joined smoothly to the logarithm outside. It is instructive to calculate the size of source which will give the same potential difference between the origin and some distant point as the finite operator solution. Fitting the constants one obtains a source radius of 0.4511[1] mesh units. In other words, the nine-point Poisson operator automatically leads to a potential profile closely resembling that of a superparticle with very plausible dimensions. If we interpret our tabulated values as the interaction potential between two finite-radius elements, i.e., if the responding element is not infinitesimal but of the same form as the source, the radius of each must be taken as 0.3513 units in order that the central interaction potential and the distant (logarithmic) interaction potential can both be matched to the tabulated values.

One might wish to use an interaction potential corresponding to a superparticle of different size. A spreading algorithm which effectively blows up the elements to about 4 or more cells in area was given by the author under the heading "An Inexpensive Noise Abatement Program" [13]. A brute-force method would be to start with the potential resulting from the nine-point operator and then simply to add or subtract a multiple of the original source distribution, thus raising or lowering the central value of the interaction potential at will. This amounts to changing the factor 1/12 in front of the $\nabla^2\phi$ term on the right of the nine-point operator identification given above. In tampering with this term, we are effectively redistributing the sources prior to inverting the stencil.

It should be noted that this method of smoothing is not like "charge sharing" or "area weighting" [11, 12] which, as shown by Langdon [19], can be combined with interpolation into a single operation, and it becomes almost a matter of

[1] This is the numerical value of $\exp\left(\frac{\pi}{6} + \frac{1}{2} - \gamma - \frac{1}{2}\ln 12\right)$ where γ is Euler's constant.

semantics whether this operation is interpreted as "smoothing" or "interpolation" or both. Rather, tampering with the $\nabla^2\phi/12$ term is equivalent to tailoring the Fourier transform of the operator which connects discrete source and field data (Langdon, loc. cit.).

A good principle for making a choice of particle size or well-depth is to apply Lewis' optimization scheme. As will be reported elsewhere, Lewis' full scheme asks for a 25-point rather than 9-point operator, but when a 9-point approximation to Lewis' scheme is attempted, the well-depth controlling ∇^2-term has to be given the factor 1/12. This justifies our adherence to the uncorrected potentials given in Table I.

III. The Interpolated Field Due to a Cell-Centered Particle

Assuming now that a source of the indicated size is located exactly at a cell center we begin our interpolation studies by letting the source act on a "receiver" that is not at a cell-center. The field, or the potential, must then be interpolated from the available table.

The earliest two-dimensional computer simulations [14] make no attempt at interpolation at all. The field in each cell is taken uniform and obtained by differencing the potential across *two* neighboring cells. Subgrid resolution is not attempted. The onus of resolution is on using a fine mesh. Computing effort is saved at 'move' time, at the cost of increased Poisson solving effort and potential array storage. It is not possible to describe the effective field distribution within each cell by a continuous potential; that is a qualitative defect which could be serious. On the other hand, there is no self-force on any particle when the method is coupled with nearest gridpoint (NGP) for off-center sources, i.e., allocating each source to its *n*earest *g*rid-*p*oint.

In order to advance beyond the original primitive and cheap option of abandoning subgrid resolution one can follow different paths. Typically one can create two field component tables by differencing *adjacent* potentials, and one can then interpolate linearly in these tables [11, 12]. More consistent would be direct quadratic interpolation of the potential and determining the local slopes of the quadratic potential surface. Barnes [15] discussed such methods in a critical survey of different algorithms. Obviously, there is an immediate substantial increase in computer effort when the displacement components of particles relative to the nearest cell center have to be determined and used as weights of several potential array entries, and when they have to be squared or multiplied with each other.

The author, adopting the philosophy of "might as well be hanged for a sheep as a lamb" tried out cubic interpolation through 10 "nearest" gridpoints [16]. He obtained fairly acceptable potential contours *within each cell*, but was disappointed by the still noticeable common failing of all these methods: The fields change discontinuously at the boundaries between cells. Such discontinuities are the cause of severe noise when particles cross into adjacent cells, and they are quite unacceptable when one wants to establish continuous flow fields in application (1).

At this point Bruce Langdon drew the author's attention to the modern technique of interpolation by splines, the very purpose of which is to avoid these

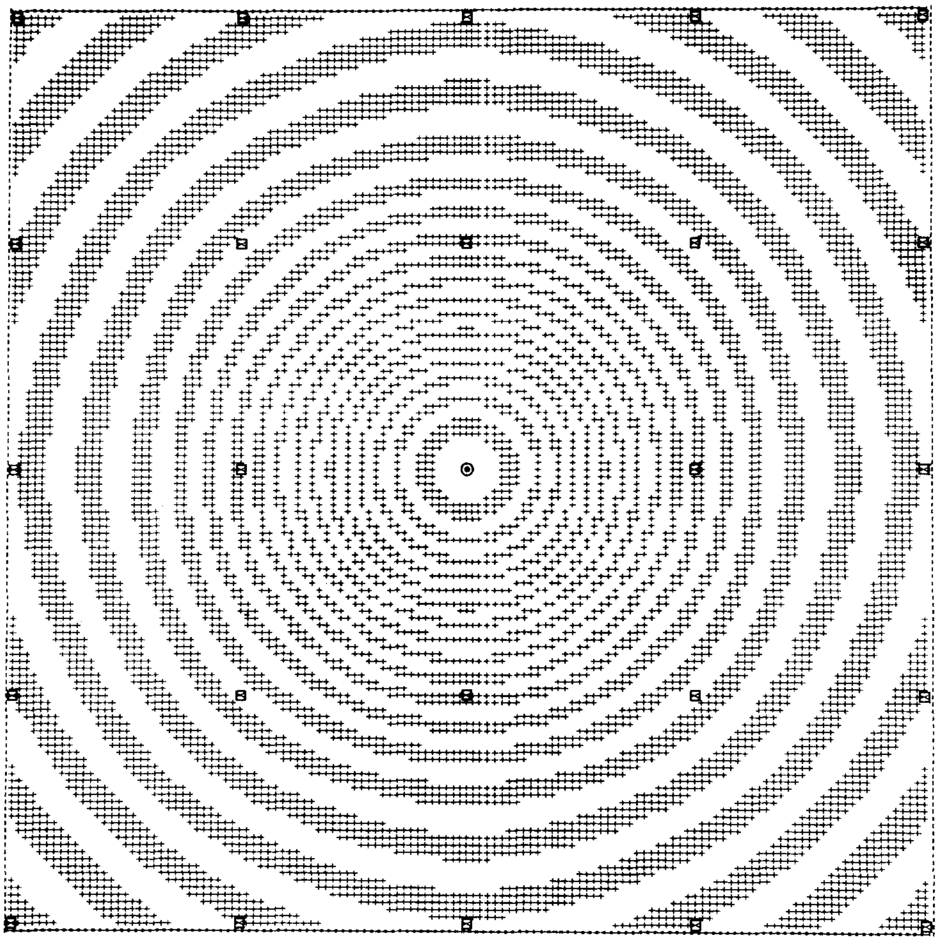

FIG. 1. Potential levels for grid point-centered source, by spline interpolation.
(Note: All figures are print-outs of computer output on an IBM communicating terminal programmed for half-shift. This gives equal horizontal and vertical spacing of print positions, eliminating the 5:3 distortion of common direct printouts. The markers showing the source center (small circle) and the gridpoints (small squares) were emphasized by hand.)

worrisome discontinuities. Bruce Langdon also obliged the author with a concise and lucid explanation of the principle of splines, enabling him to put the splines to work directly. Figure 1 is the outcome of quadratic spline fitting to the entries of Table 1.

The figure provides a contour map of the potential, with alternate levels shown blank and shaded.[2] Any borderline of a shaded region is a contour line. In application (1) this is a flow line, in application (2) a magnetic field line. In applications (3) and (4) the field lines are normal to these contour lines, and their spacing measures field strength. The square marks in the figure indicate the cell centers where the values of Table 1 are supplied. Cell boundaries run, of course, midway between these centers.

There are no visible discontinuities across cell boundaries. One can discern a deviation from circularity in the contours, but such deviations are near the threshold of resolution of the display, which is to *one-thirty second* of a mesh unit. We certainly have good subgrid resolution.

The contours may be compared with the exact contours due to a uniformly dense source of radius 0.4511 units, namely,

$$\Phi = (\pi/3) + (r/0.4511)^2 \qquad \text{up to} \quad r = 0.4511,$$

and

$$\Phi = (\pi/3) + 1 + \ln(r/0.4511)^2 \qquad \text{beyond,}$$

calculated without interpolation. This comparison is given in Fig. 2 where the left-hand side is a version of Fig. 1 with only half the number of contour levels, while the right-hand side shows the ideal potential. At distances beyond one cell unit the discrepancy is almost down to the resolution of the display. Within one cell unit radius the *number* of contours is (a fortiori) the same, but they are displaced relative to each other by as much as 1/16 of a unit. The ideal potential well (right half of Fig. 2) is narrower.

Quadratic splines were fitted in both the x and y directions. This means that in each cell one uses a quadratic in x with coefficients that are quadratic in y or vice versa. Each such doubly quadratic function with its 3-by-3 coefficients is fitted at the cell center and the remaining eight data are primarily determined from the eight closest neighbors. However, the condition of continuity across cell boundaries introduces a further, somewhat weaker dependence on more distant gridpoint values. With the typical 24-bit precision in IBM 360 floating-point arithmetic, this secondary influence reaches to about nine cells away. Section VI below explains the procedure in detail and presents some useful algorithms for its implementation. (The secondary, weaker influence is automatically built into Lewis' scheme which

[2] It is a display of a selected bit in a 129 by 129 array of subtabluated potential values: odd eighths are shown as plus signs, even eighths as blanks (units as in Table 1).

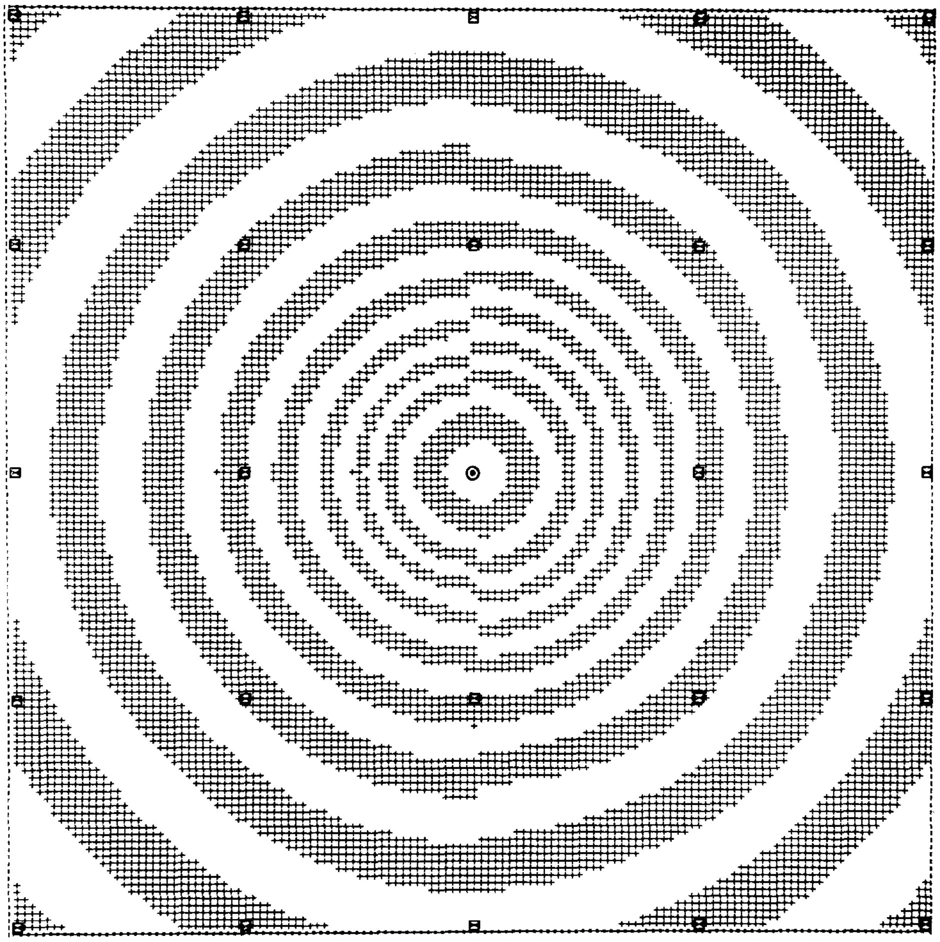

FIG. 2. Left: potential as for Fig. 1, with half the number of levels. Right: uninterpolated potential levels due to ideal solid cylindrical source of radius 0.4511 mesh units.

produces an array of spline coefficients—weights of basis functions—rather than an array of potential values over a grid.)

IV. INTERACTION POTENTIAL DUE TO AN ARBITRARILY PLACED SOURCE

Looking at Figs. 1 and 2 one might become unduly optimistic regarding the power of spline interpolation. One would confidently derive velocity or field components from the direction and spacing of the equipotentials, i.e., by evaluating the partial derivative of our quadratic functions, anywhere in any cell.

However, the good performance of the scheme is due largely to the fortunate placing of the source at a cell center. When the source is moved off-center, results are much less satisfactory, as shown by the distorted and displaced contours in Fig. 3.

In this case, the interaction potential has to be interpolated at both ends, i.e., one writes r^2 as $(x - x')^2 + (y - y')^2$ in the parabolic-logorithmic potential given in Section II and interpolates on all four variables x, x', y, y' when neither the source nor the responding element are at integer locations.

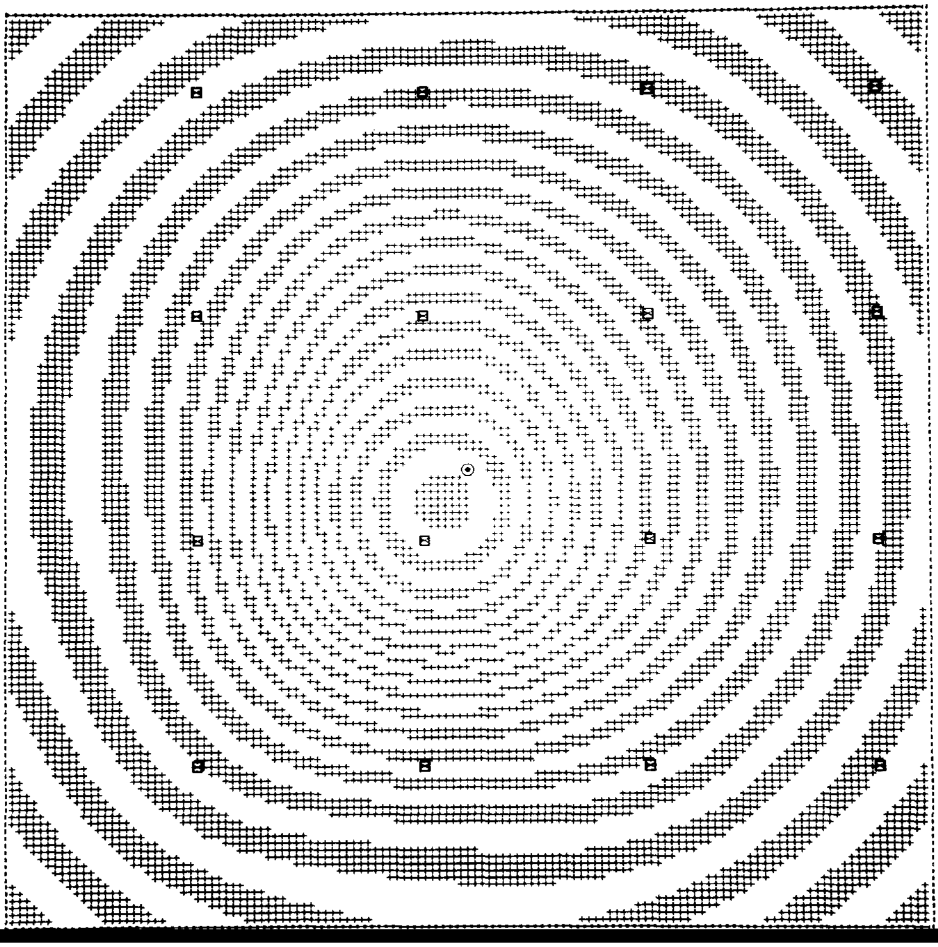

FIG. 3. Potential levels for source with coordinates 5/16, 5/16 relative to a gridpoint, by spline interpolation.

Rather than using the exact logorithmic values for the table of the kernel, we use again the slightly imperfect values readily obtained by nine-point Poisson operator inversion. These values were used for the case of a grid-centered source in Section III, and they are listed in Table 1.

Interpolation on x and y is mathematically the same as interpolation on x' and y'. The same quadratic splines will be used in this new interpolation as were used previously. One can interpret the process of interpolation as a form of "weighting" or "sharing," and this alternative view is often preferred. The equivalence of the two points of view is demonstrated in [19]. Linear weighting or "area" weighting has long been used to improve on the crude and cheap NGP (nearest gridpoint) method of accumulating charges of particles into the charge array. Here we do not only advance from linear to quadratic weighting, but we do this by quadratic spline fitting. This amounts to distributing a charge or source primarily among the nearest nine gridpoints and making a secondary allocation over a wider range—up to 9 mesh units away before the weights become subliminal in IBM 360 precision. This secondary allocation is unnecessary if one uses Lewis' operator which connects spline coefficients of density directly with spline coefficients of potentials.

Having distributed a single off-center charge to neighboring gridpoints and applied a nine-point Poisson operator inversion to the charge array, one then applies another spline-fitting in order to get the potential everywhere, between the gridpoints as well as at the gridpoints.

Figure 3 shows the result of such a multiple operation for the case where $x = 3/16$ and $y = 5/16$, chosen as a "typical" off-center position. The display is arranged so that the source is in the middle of the picture and the gridpoints are offset.

We note considerable distortion of the equipotentials from the ideal (the right half of Fig. 2) or from the on-center configuration of Fig. 1. The potential well has in fact been dragged over towards the nearest gridpoint, and a count of contours, starting from the distant contours which agree tolerably with those of Fig. 1, shows that one contour has been lost. The well is less deep or, for a positive charge, the peak is lower.

The centers of the outer contours agree well with the position of the source but the centers of the inner contours are displaced from where they ought to be up to 1/8 of a mesh unit. A systematic exploration of the full range of possible grid locations relative to the source shows this to be about the worst displacement.

Regarding the well depth, one gets more severe effects as the source moves toward the center between four nearest gridpoints. In this position there is no displacement, but one finds a more drastic reduction of well depth. The contours are shown in Fig. 4.

When one moves the grid into various positions relative to the source, the

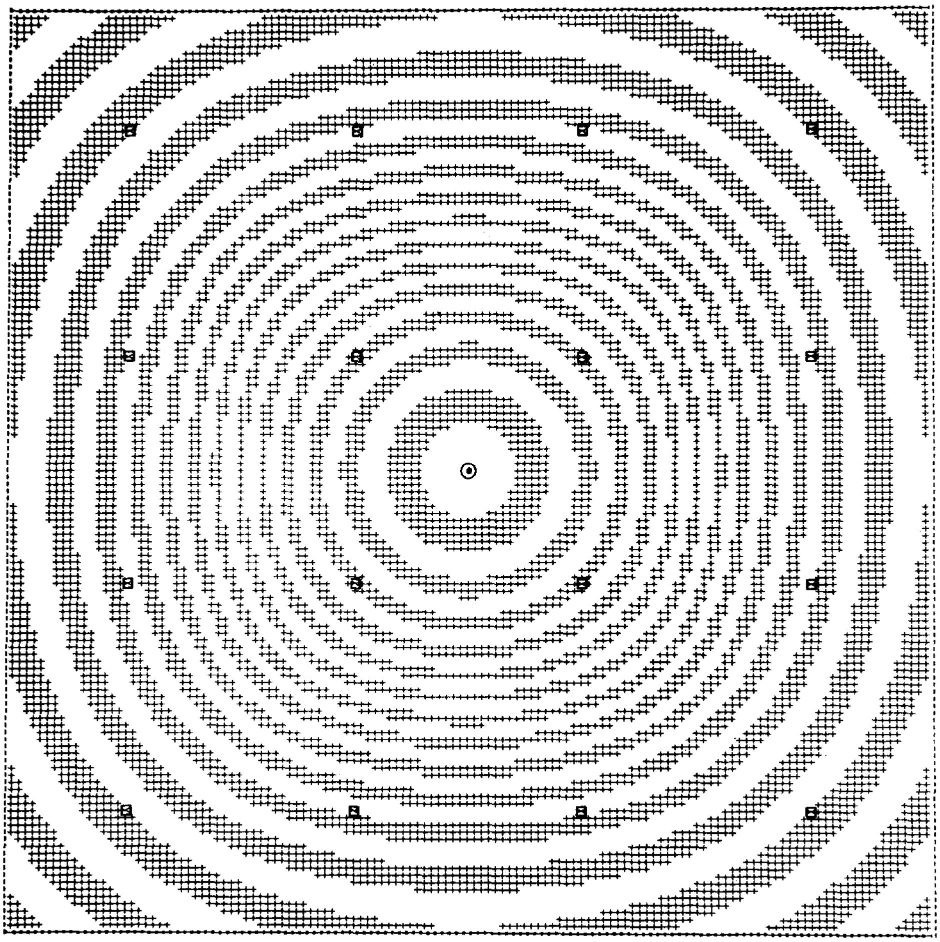

FIG. 4. Potential levels for source midway between four gridpoints, by spline interpolation.

potential well is gently pulled out of shape or, putting it another way, when a source moves through a grid, some wobbles in its position and strength are introduced by the grid. These effects are strictly local: The contours more than 2 mesh points away remain fairly steady relative to the source. Since we have used splines, no discontinuities are introduced by the grid.

The uncertainty of position introduced by the grid within the vicinity of the source is about one-eighth of a mesh unit, i.e., less than the implied source radius. In view of this, one would be wasting information if one were to spread the source deliberately over a larger area than that which resulted naturally from the Poisson-solving algorithm. On the other hand, one could hardly go much further in the other direction, that of narrowing the source; indeed, the more the singularity at the origin stands out against the rest of the entries in Table 1, the less accurately can it be reproduced by interpolation from a table of values taken in the surrounding flatter domain. For this is what our scheme amounts to: Instead of passing on the data for a contour map in the form of the original benchmarks which included the bottom of the well or top of the peak, we only supply data obtained by interpolation from the original benchmarks on the slopes of the well or peak.

To get the same localization (1/8 mesh units) with NGP, one would have to quadruple the meshpoints in each dimension and the extra cost in storage (or buffering in and out) and Poisson solving time must be weighed against the cost of the spline interpolations. However, one would then also forego the benefit of higher quality field mapping at moderate distances from the source (about 1/16 of our present mesh units), and one would miss the benefit of the smoothness of the spline interpolated fields.

Making the grid four times finer in each dimension increases Poisson solving time by more than a factor of 16 (a factor 20 would be representative; see [2]), assuming core storage is available. The cost of quadratic spline interpolation, on the other hand, can be estimated from the following considerations:

(a) Generating tables of spline coefficients adds 50–80 % to the execution of the fastest direct Poisson codes (see Section VI), but nothing to the popular two-dimensional transform codes.

(b) When advancing from NGP to linear interpolation (area weighting) in two dimensions, one introduces four multiplications and eight additions per element per time step and per interpolated variable (density, potential, field component). When advancing from linear to quadratic interpolation, one introduces another six multiplications and 18 additions per element per time step and per variable.

In view of the fact that in plasma and gravitational applications the total number of particles has to be large for various reasons, such as noise suppression and good resolution in momentum space as well as in x–y space, the cost balance could easily go against quadratic spline interpolation here, i.e., more grid points and a simpler "mover" may be cheaper.

In sum, quadratic spline interpolation gives satisfactory subgrid resolution and good economy for handling isolated elements or strongly localized sporadic concentrations. Regarding collective effects in plasmas and gravitating systems, there may be some doubt as to its economy. If it is used, one may speculate that *some* subgrid physics will be simulated adequately, and mesh sizes as large as a Debye length or a Larmor radius may be acceptable. One knows that mesh sizes of several Debye lengths lead to disastrous stroboscopic grid effects ("aliassing"), which can, however, be alleviated by higher order interpolation, i.e., by improved subgrid resolution [19].

V. Self-Effect and Self-Energy

Of major concern to the physicists (who have been struggling with self-energy problems of particles for nearly a century) will be the self-propulsion and variable self-energy exhibited by the elements in our scheme.

Clearly an off-center vortex will make itself precess, as seen in Fig. 3. An off-center charge will propel itself away from the nearest gridpoint while an off-center mass will gravitate towards that gridpoint. The self-energy of a particle does not remain constant while it moves through the grid: it is highest at a gridpoint (Fig. 1) and lowest midway between four gridpoints (Fig. 4).

In spite of this variability of self-energy, there exists a pseudo-energy for an assembly of charged or gravitating particles which is conserved to *second* order in the time step. If one forms the kinetic energy from the displacements during a time step and adds to this the potential energy of the particles at their final position in the field created by them in their initial positions, one obtains an energy-like expression which is time-symmetric for our scheme. (To check this, one only needs to multiply the central difference acceleration statement by the total displacement through two time-steps—and one can also check that a magnetic field parallel to the charged rods, if introduced time symmetrically as in [17], will have no effect on this energy balance.) The time-symmetry of the total (kinetic plus potential) energy so defined is due to using the same interpolation scheme for the sources and the responders—the interaction kernel array itself is symmetric already.[3] From the time symmetry it follows directly that the energy error is quadratic in the time-step. Langdon [19] found the kernel and interpolation symmetry to be a sufficient condition for energy conservation in the limit of zero time-step. In a recent communication [9] he emphasizes this feature with respect to the use of splines. As long as we are interpolating potentials, not fields, from their gridpoint values, we are operating in the mode described by Eqs. (28, 29) in Ref. [19]. This mode permits exact energy conservation.

Regarding the self-force and the ugly (but, as just explained, not crippling) variable self-energy, we observe that (1) these features do not occur with NGP and (2) they would not occur if, instead of interpolating for the potential, we interpolated instead for the field components from two field component tables. These latter tables (for an isolated source at the origin) are antisymmetric in one of the two indices, symmetric in the other. The choice of a finite potential value at the origin which occupied us in Section III would be immaterial and hence particle size (provided it is less than a mesh unit) plays no part. Antisymmetry forces the field entry at the origin to be zero.

[3] An optimal Lewis operator, in place of our nine-point Poisson operator, would likewise result in energy conservation—after all, it is derived from an energy principle (Ref. [18]).

Now let $\mathbf{E}_{ij}$ be the vector array of gridpoint fields and let $w_{ij}(\mathbf{r})$ be the weights for an element at $\mathbf{r}$. Then the field at $\mathbf{r}$ due to a source at $\mathbf{R}$,

$$\sum_{i,j,l,m} w_{ij}(\mathbf{r})\, \mathbf{E}_{i-l,j-m} w_{lm}(\mathbf{R})$$

is antisymmetric with respect to interchange of $\mathbf{r}$ and $\mathbf{R}$. This is Newton's third law and results in zero self-force as well as strict momentum conservation even for finite time-step. However, one cannot readily construct a quasi-energy for this scheme, even if one requires only conservation to second order in the time-step (see [19] for the case $\delta t \to 0$).

Thus one has the mutually exclusive choice between momentum and "energy" conservation, in that one can use spline interpolation on the field components *or* the potential array. We have presented the latter case in our calculations and diagrams, for illustrative purposes, and to save array space. An important application of spline interpolation will be to electromagnetic codes which, perferentially, employ field components [20] and which, on account of the three-dimensionality of the important applications, call for the drastic economy of a very open mesh.

VI. The Weights Resulting from Spline Fitting

Of the technical detail in the calculations the spline fitting algorithm is worth reporting. One can then get a feel for how a source is "shared" between the gridpoints and how information is gathered from the gridpoints for the evaluation of the local fields.

We present the procedure for one dimension only—extension to two dimensions is trivial. We consider a potential function $F(x)$ whose values are given at integer points $x = n$ as an array F_n. In the range $n - 0.5 < x < n + 0.5$ we fit a parabola which we write in the form

$$F(x) = F_n{}^0 + (x - n)\, \delta F_n{}^0 + ((x - n)^2 + 1/4)\, \delta^2 F_n{}^0/2, \tag{1}$$

where $F_n{}^0$, $\delta F_n{}^0$ and $\delta^2 F_n{}^0$ should be considered, in the first place, just as names for the coefficients appropriate to the n-th interval. Notice the extra "1/4" included in the quadratic term. Without this, we would have an ordinary quadratic interpolation formula; $F_n{}^0$ would then be identified with F_n while $\delta F_n{}^0$, $\delta^2 F_n{}^0$ would be the first and second central differences in the $F_n{}^0$ (or F_n) array. In the presence of the "1/4" term the $F_n{}^0$ array cannot be identical with the F_n array, as seen when one puts $x = n$. Let us concentrate on the continuity across cell boundaries first.

At the limits of the interval, $x = n \pm 1/2$, the formula yields

$$F(n \pm 1/2) = F_n{}^0 \pm \tfrac{1}{2}\delta F_n{}^0 + \tfrac{1}{4}\delta^2 F_n{}^0,$$
$$F'(n \pm 1/2) = \delta F_n{}^0 \pm \delta^2 F_n{}^0,$$

and if $\delta F_n{}^0$, $\delta^2 F_n{}^0$ are now interpreted as first and second central differences of the yet undetermined array $F_n{}^0$, we obtain

$$F(n \pm \tfrac{1}{2}) = (F_n{}^0 + F_{n\pm 1}^0)/2,$$
$$F'(n + \tfrac{1}{2}) = F_{n+1}^0 - F_n{}^0,$$
$$F'(n - \tfrac{1}{2}) = F_n{}^0 - F_{n-1}^0,$$

showing that we get the same values for function and slope if we work from the neighboring cells, using the same array of coefficients $F_n{}^0$.

Applying (1) at $x = n$, one gets

$$F_n = (1 + \delta^2/8)\, F_n{}^0; \tag{2}$$

in other words, the coefficients $F_n{}^0$ are found from F_n by inverting the difference operator $1 + \delta^2/8$. The "Greens function" for this operator is readily obtained, and one solves

$$F_n{}^0 = \sum_m \frac{\sqrt{2}\, F_m}{(-3 - \sqrt{8})^{|n-m|}}\,. \tag{3}$$

Thus, the coefficients are primarily given by the local value F_n, but with an additional dependence on the nearest neighbors. The dependence gets exponentially weaker with distance and, as indicated before, when $|\, n - m \,|$ exceeds 9, the weighting factor in (3) drops below detectability in 24 bit precision. For this reason we have not mentioned boundary conditions specifically—in any case, the examples discussed here are for infinite space: after all, the whole picture in Figs. 1–4 only extends 2 cells away from the source.

Spline interpolation, or weighting, is therefore a two-stage process: first comes the inversion of $1 + \delta^2/8$ which results in an exponentially attenuated set of weights for the entire neighborhood of the particle. Then comes the construction of the parabola, resulting in weights which are quadratic in x, to be applied at the three closest mesh points.

The two-stage weighting operation is carried out twice, namely, at the input end—before Poisson inversion—when charge or circulation is shared over the surrounding gridpoints and again at the output end when the field is to be deduced from the potentials on the gridpoints in the environment. On each occasion an inversion of the $1 + \delta^2/8$ operator must be carried out, or the kernel

$\sqrt{2}(-3 - \sqrt{8})^{-|n-m|}$ must be applied. One might think that at one end of the entire procedure the operator is applied directly while at the other end it is inverted, and one might be tempted to cancel the two steps against each other. This is not so: $1 + \delta^2/8$ is inverted both times, or the kernel is applied both times. There is a further doubling of these operations due to working in two dimensions.

The operator $1 + \delta^2/8$ can be inverted very easily when one works in Fourier transforms, as required by many Poisson solving schemes: for the harmonic characterized by the variation $\exp(2\pi ikz/N)$ the inverse operator becomes a numerical divisor of value $(3 + \cos 2\pi k/N)/4$. This can be built into the set of pretabulated multipliers for generating Fourier harmonics of potential from Fourier harmonics of density and should not lengthen the execution time per step.

Without Fourier-transforms, the inversion is more conveniently done by "cyclic reduction" [21] than by applying the Greens function as in Eq. (3). The three-term recurrence relation (2) for $F_n{}^0$ may be written in the form

$$F_n{}^0 = F_n - (F^0_{n+1} + F^0_{n-1} - 2F_n)/6 \text{ to be used for } n = 1, 3, 5, 7,... \qquad (4)$$

in order to eliminate all odd-indexed unknowns and to obtain the three-term recurrence relation for even-indexed unknowns, namely,

$$F_n{}^0 = F_n{}^I + (F^0_{n-2} + F^0_{n+2} - 2F_n{}^I)/34 \text{ to be used for } n = 2, 6, 10,... \qquad (5)$$

with $F_n{}^I$ defined by

$$F_n{}^I = F_n - (F_{n-1} + F_{n+1} - 2F_n)/4 \qquad \text{to be used for} \quad n = 2, 4, 6, 8,... . \qquad (6)$$

Using (5) at the indicated n-values, one eliminates unknowns with twice-odd index and constructs the three-term recurrence relation connecting unknowns with indices divisible by 4, namely,

$$F_n{}^0 = F_n^{II} + (F^0_{n-4} + F^0_{n+4} - 2F_n^{II})/1154 \text{ to be used for } n = 4, 12, 20,... . \qquad (7)$$

with the definitions

$$F_n^{II} = F_n{}^I + F^I_{n-2} + F^I_{n+2} - 2F_n{}^I)/36 \text{ to be used for } n = 4, 8, 12, 16,... . \qquad (8)$$

Again, using (7) for the listed indices one advances to the next level and obtains

$$F_n{}^0 = F_n^{III} + (F^0_{n-8} + F^0_{n+8} - 2F_n^{III})/1331714 \text{ to be used for } n = 8, 24, 40,... \qquad (9)$$

with the definitions

$$F_n^{III} = F_n^{II} + (F^{II}_{n-4} + F^{II}_{n+4} - 2F_n^{II})/1156 \text{ to be used for } n = 8, 16, 24,... . \qquad (10)$$

In the next step the denominator[4] in the expression for $F_n{}^0$ becomes so large ($> 10^{12}$) that we can equate $F_n{}^0$ with F_n^{IV}, defined in analogy with the preceding definitions 6, 8, 10. Hence:

$$F_n{}^0 = F_n^{III} + (F_{n-8}^{III} + F_{n+8}^{III} - 2F_n^{III})/1331716 \text{ to be used for } n = 16, 32, \ldots. \quad (11)$$

The eight statements 6, 8, 10, 11, 9, 7, 5, 4, *taken in that order* and looped through the listed indices in each case, constitute the program for inverting $1 + \delta^2/8$. The superscripts on all F_n may be ignored: at each step the old value of F_n may be overwritten by the new value. The number of operations is comparable to the number of operations which are carried out $\log_2 N$ times in Poisson solvers that do *not* employ *two*-dimensional Fourier transforms. Since the algorithm is repeated four times, one can expect a 50–80 % increase of execution time for the field calculation step in typical applications.

The full cycle of source weighting ("charge sharing") and flow or field evaluation now proceeds according to the following flow-sheet of operations:

The first version of the source array is created by distributing each source among its 9 nearest neighbors in two dimensions. The 3-by-3 weight matrix is the outer product of the 3 weights in the x direction and the 3 weights in the y direction. The three weights in the x direction are:

$$\begin{aligned} &\tfrac{3}{4} - (x - n)^2 \text{ at the nearest gridpoint,}\\ &\tfrac{1}{2}(x - n - \tfrac{1}{2})^2 \text{ at the gridpoint on the left,}\\ &\tfrac{1}{2}(x - n + \tfrac{1}{2})^2 \text{ at the gridpoint on the right.}\end{aligned}$$

Having constructed the first version, one obtains the second version by convolution with the kernel $\sqrt{2}/(-3 - \sqrt{8})^{|n-m|}$, in x as well as in y. This amounts to applying the inverse of the operator $(1 + \delta^2/8)$ and is most conveniently done by Fourier transforming or by the algorithm programmed in Eqs. (4)–(11) above.

After this, the Poisson operator is inverted and then the preceding step is repeated. In the final potential table quadratic interpolation is used with multipliers as given in Eq. (1). This must be done in x and in y, amounting in fact to doing three y interpolations, namely, at the three nearest x levels, followed by one interpolation in x. A local potential value is thus obtained.

To get the local y velocity or x force one y-interpolates at the three nearest x levels as described in the preceding paragraphs, but then one evaluates the x-derivative of (1). This means one applies only linear weights to the three last-calculated coefficients. Gradients (or curls) are formed by differentiation of the *splines* fitted to the fixed grid potential values.

[4] The algorithm for generating the denominators is to start with the number -4 and then to alternate subtractions of 2 with squaring.

For the x velocity or y force one makes the obvious changes to the procedure just described.

The Poisson inversion and the two inversions of $1 + \delta^2/8$ may be carried out in any order if the boundaries are more than 8 cells away from the sources. For the purpose of generating the diagrams, the algorithm of Eq. (4)-(11) was applied twice in x and twice in y to the array of Table 1. From the newly created table a set of benchmarks was calculated by interpolating in each cell with the same x and y displacements from the center. This is equivalent to a Poisson inversion of the source array consisting of the weights due to a single displaced source, and it includes the final $1 + \delta^2/8$ inversion preparatory to the final interpolations. The latter were then performed at 16641 ($= 129^2$) points covering an area of 4-by-4 cells including cell boundaries.

ACKNOWLEDGMENT

This work was supported by NASA Grant NGL 05-020-420.

REFERENCES

1. O. BUNEMAN, "Vortices, the Particles of Fluid and Gasdynamics," Proc. 4th Conf. on Numerical Simulation of Plasmas, N.R.L., Washington, DC, 1970.
2. R. W. HOCKNEY, *Methods Computational Phys.* **9** (1970), 135.
3. J. A. BYERS AND J. KILLEEN, *Methods Computational Phys.* **9** (1970), 241.
4. V. R. WATSON, Computer simulation of rarefied plasmas, in "Rarefied Gasdynamics, 7th Symposium," Academic Press, New York, 1971.
5. J. P. CHRISTIANSEN, "Numerical Simulation of Hydrodynamics by the Method of Point Vortices," UKAEA Culham Laboratory Report CLM-P282, Abingdon, 1971.
6. O. BUNEMAN, "Analytic Inversion of Nine-Point Poisson Operator," SUIPR Report No. 447, Stanford University, 1972.
7. O. BUNEMAN, *J. Computational Phys.* **8** (1971), 500.
8. H. R. LEWIS, *Methods Computational Phys.* **9** (1970), 307.
9. A. B. LANGDON, "Energy Conserving Plasma Simulation Algorithms," U.C. Lawrence Radiation Laboratory Report UCRL 72869, June 1971.
10. C. K. BIRDSALL AND D. FUSS, *J. Computational Phys.* **3** (1969), 494.
11. R. L. MORSE, *Methods Computational Phys.* **9** (1970), 213.
12. C. K. BIRDSALL, A. B. LANGDON AND H. OKUDA, *Methods Computational Phys.* **9** (1970), 241.
13. O. BUNEMAN, "An Inexpensive Noise Abatement Program," 5th Conf. on Computer Simulation of Plasma, paper D7, Iowa City, November 1971.
14. R. W. HOCKNEY, *Phys. Fluids* **9** (1966), 1826.
15. C. BARNES, "Getting the Most Out of Your Poisson Solver," 5th Conf. on Computer Simulation of Plasma, paper D2, Iowa City, November 1971.
16. O. BUNEMAN, "Sub-Grid Resolution by Cubic Weighting and Cubic Interpolation," 5th Conf. on Computer Simulation of Plasma, paper D7, Iowa City, November 1971.

17. O. Buneman, *J. Computational Phys.* **1** (1967), 517.
18. H. R. Lewis, *J. Computational Phys.* **6** (1970), 136.
19. A. B. Langdon, *J. Computational Phys.* **6** (1970), 247.
20. O. Buneman, Fast numerical procedures for computer experiments on relativistic plasmas *in* "Relativistic Plasmas" (Buneman and Pardo, Eds.) p. 205, Benjamin, New York, 1968.
21. R. S. Varga, "Matrix Iterative Analysis," p. 195, Prentice-Hall, Princeton, NJ, 1962.
22. L. Collatz, "The Numerical Treatment of Differential Equations," 3rd Ed., p. 542, last case, Springer-Verlag, Berlin, 1960.