JOURNAL OF COMPUTATIONAL PHYSICS 83, 126–147 (1989)

Fast, Adaptive Summation of Point Forces in the Two-Dimensional Poisson Equation

LEON VAN DOMMELEN

Department of Mechanical Engineering,
FAMU/FSU College of Engineering & Supercomputer Computations Research Institute,
Florida State University, Tallahassee, Florida 32306

AND

ELKE A. RUNDENSTEINER

Department of Computer Science,
Florida State University, Tallahassee, Florida 32306

Received June 2, 1987; revised September 30, 1988

Direct summation of the velocity field introduced by point vortices tends to be time consuming since the velocity of each vortex is found as a sum over all other vortices. The resulting number of numerical operations is proportional to the square of the number of vortices. Here a relatively simple procedure is outlined which significantly reduces the number of operations by replacing selected partial sums by asymptotic series. The resulting number of operations appears to vary roughly in proportion to the number of unknowns, corresponding to a "fast" solver. © 1989 Academic Press, Inc.

1. INTRODUCTION

Incompressible flow at high Reynolds number with large-scale separation can be difficult to compute since the vorticity tends to concentrate in limited parts of the flow field. Vortex methods [1] attempt to reduce the number of variables needed for the computation by describing only the vorticity, in its simplest form, by a series of delta-functions or point vortices:

$$\omega = \sum_{i=1}^{N} \Gamma_i \cdot \delta(\mathbf{x} - \mathbf{x}_i). \tag{1}$$

The flow velocity is related to the vorticity by the solution of a Poisson equation, with the vorticity as forcing function, resulting in the stream function. The flow velocity is found by taking the curl of the stream function. The solution for a series

0021-9991/89 $3.00

of delta functions can be found and leads to the following expression for the flow velocity:

$$V_j^* = \frac{\sqrt{-1}}{2\pi} \sum_{\substack{i=1 \\ i \neq j}}^{N} \Gamma_i \frac{1}{Z_j - Z_i}, \qquad j = 1, 2, ..., N, \tag{2}$$

where Z is the complex position $x + \sqrt{-1}y$ and V^* the complex conjugate velocity $u - \sqrt{-1}v$. In general, it will be necessary to add to this flow velocity a solution of a Laplace problem to take care of the boundary conditions.

While the sum in Eq. (2) may easily be evaluated, the number of terms is proportional to the square of the number of vortices N. Thus, the computational effort increases rapidly when the number of vortices increases. In contrast, various mesh-based solution procedures for the Poisson equation are able to find the solution in a computational time roughly proportional to the number of mesh cells. As a result, a point vortex description seems most useful if (a) the number of vortices is much smaller than the number of mesh cells needed to describe the flow (i.e., the vorticity is restricted to a small part of the total domain); (b) the point-singularity description itself is of particular interest and the errors induced by a mesh-based representation must be avoided; or (c) the infinite domain implicit in Eq. (2) is to be preserved. Certainly discrete vortex representations have drawn and continue to draw considerable theoretical and numerical interest. In addition, the Poisson equation is not unique to fluid mechanics; it arises in other fields such as electromagnetism and gravitation. For these reasons, more efficient procedures to evaluate the solution under pointwise forcing are of considerable interest.

Various methods to reduce the computational effort have been proposed. Anderson [2] used a fast Fourier transform method, with corrections for the interactions between nearby vortices. However, some of the mentioned advantages of the vortex method are lost due to the presence of the mesh. In addition, for high accuracy the evaluation of the interactions between neighboring vortices can become computationally intensive.

An alternative approach followed in this paper is, to group the vortices spatially and to approximate the effects induced by each group at larger distances. Appel [3] and Barnes and Hut [4] made approximations using a single replacement element. Yet, using such approximations, high accuracy is difficult to achieve while the algorithm tends to be scalar.

In contrast, the present study uses a Laurent series approximation for the velocity induced by each group. This approximation takes the form

$$V_j^* = \sum_{k=1}^{\infty} \frac{C_k}{(Z_j - Z_0)^k} \tag{3a}$$

$$C_k = \frac{\sqrt{-1}}{2\pi} \sum_i \Gamma_i (Z_i - Z_0)^{k-1}, \tag{3b}$$

where Z_0 is a suitably chosen origin point for the group of vortices and the sum in (3b) extends over all vortices in the considered group. The Laurent series allow the desired accuracy to be maintained by the choice of the truncation of the infinite sum. In addition, when the point j at which the velocity is to be evaluated is sufficiently far distant from the group of vortices, the series converges geometrically and only a limited number of terms is needed for given accuracy. Savings in computational effort result when the number of terms needed for the Laurent series is sufficiently small compared to the number of vortices in the group. For that reason, a minimum group size exists below which further savings are not made. Using the adaptive algorithm on a CYBER 205 computer, Van Dommelen and Rundensteiner [5] found that this group size is of the order of a 100 elements.

At about the same time, similar ideas were developed by Rokhlin [6] and Greengard and Rokhlin [7]. In fact, an adaptive algorithm developed by Carrier, Greengard, and Rokhlin [8] is quite similar to the present one in both the use of Laurent series and the grouping involved. An important difference between the procedures is how the adaptive group structure is addressed. While the procedure [8] is based on five topological sets expressing the relationships between groups, the present procedure is based on an unusual numbering system of the groups. The numbering system is generated simultaneously with the group structure; it leads to a relatively simple and streamlined program logic.

The procedure of Greengard and Rokhlin recasts the Laurent series as Taylor series to achieve further reductions in computational operations, an enhancement not yet incorporated in the present scheme. However, unless the number of vortices is sufficiently large, the possible savings seem limited. Furthermore, not recasting the series offers some compensating advantages, such as reduced storage (only a vanishingly small fraction of the Laurent series expansions need be stored), increased vector length, and less overhead.

In its present form, our procedure can be divided into two parts: generation of an adaptive panel structure, to groups the vortices spatially, and determination of the velocity. The next two sections describe each of these steps in turn.

2. Generation and Numbering of the Panels

In order to use Laurent series effectively, the vortices must be spatially grouped together. Figure 1 illustrates a typical grouping for the example of flow about a circular cylinder. In this example, there are 16,479 vortices outside the cylinder (shown as dots) and an equal number of mirror vortices inside the cylinder (not shown).

The procedure for generating this panel structure is shown in Fig. 2. The first few steps are further illustrated in Fig. 3. The starting domain is taken as the smallest square that encloses all vortices. This square is subdivided into four squares, or subpanels, of equal size (indicated as **A**, **B**, **C**, and **D** in Fig. 3). The vortices are reordered so as to group the vortices in each of the four subpanels together (in the

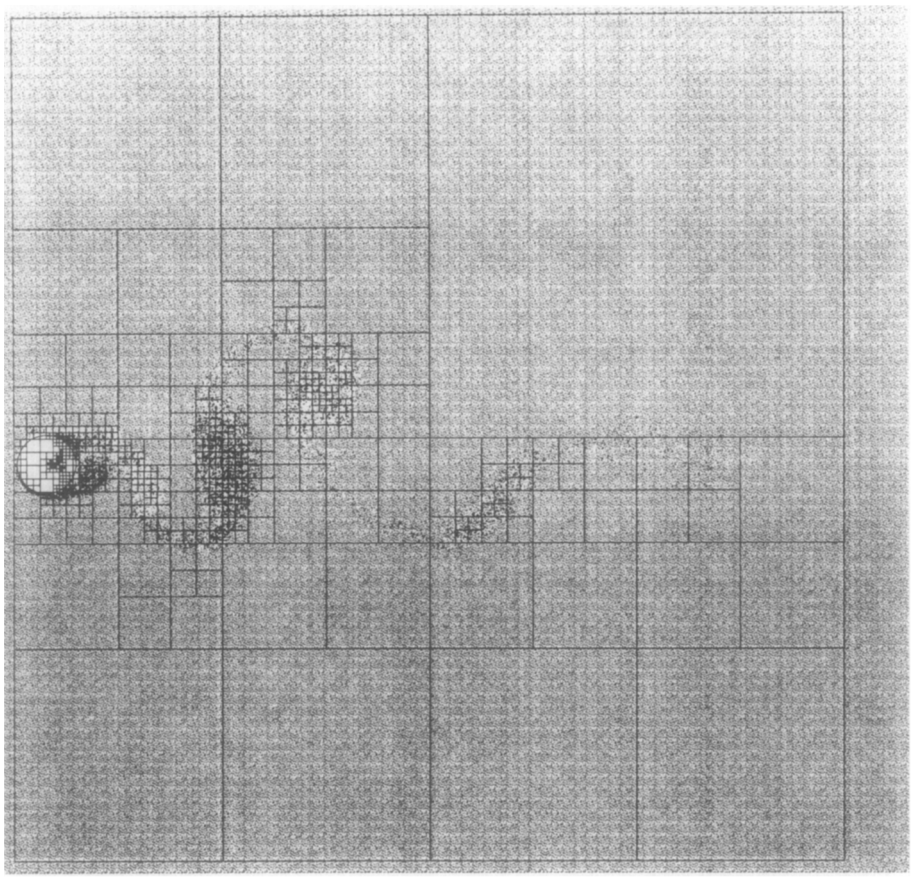

FIG. 1. Example panel structure generated for flow about a circular cylinder. The 16,000 vortices outside the cylinder are shown as dots; an equal number of mirror vortices within the cylinder are not shown.

CYBER 205 implementation, the built-in vector function Q8VCMPRS was used for this purpose).

Information about each of the panels is stored in memory. This information includes the position of the panel and the storage locations of the first and the last vortex within the panel. It also includes an identifying panel number defined later.

After subdivision, execution transfers to the first of the four subpanels generated, and a decision is made whether this panel should be further subdivided. The decision is based on the number of vortices in the panel; if sufficient vortices are present, for example, more than 100, the panel is further subdivided. Meanwhile the panel data for the remaining three subpanels is temporarily stored away in a last-in, first-out buffer. (The last part of the memory allocated for the panel information was used as buffer.)

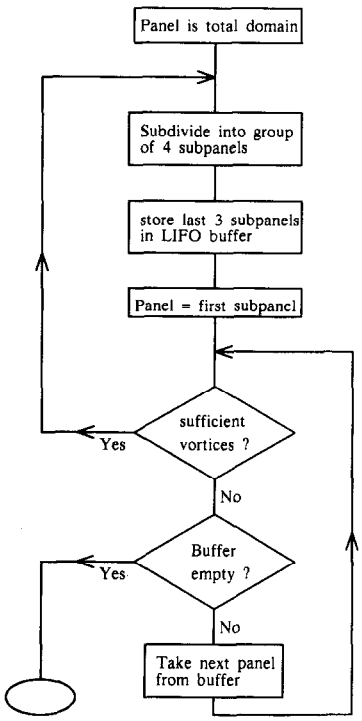

FIG. 2. Basic flow chart for generating the panel structure.

If the considered panel does not contain sufficient vortices, the next panel will be retrieved from the buffer for possible further subdivisions. Proceeding in this manner, the entire domain is subdivided into panels containing a limited number of vortices.

Each of the panels is given a unique number to simplify identification of the panel and its place in the panel structure. Figure 3 illustrates that the storage is always kept in order of increasing panel number. The actual definition of the panel number is illustrated in Fig. 4: all panels which could be created by the subdivision process can be represented as a series of uniform divisions of the domain. For each of these uniform subdivisions, the x- and y-positions can be given a binary number. The four panels generated at the first level of subdivision can be numbered using a one-digit binary number (top of Fig. 4). Each additional level of subdivision requires one additional digit.

Therefore, the binary digits determine the position of the panel. The number of binary digits determines the subdivision level. It follows that the binary digits of the x- and y-positions describe the panels uniquely. The complete information is stored in a single panel number using the following procedure:

(a) Increment each digit in both the x- and y-position by one, so that binary zero becomes 1 and binary one becomes 2.

(b) "Interleave" the resulting digits of the x- and y-positions into a single number, so that the odd digits become the digits of the x-position and the even digits those of the y-position.

(c) Add trailing zeros to obtain a final panel number with a fixed and predetermined number of digits. The 205 procedure chooses a 28 digit panel number.

The procedure is illustrated in Fig. 4 for example panels. Since the highest value of the digits in the obtained panel number is 2, it can be considered as the representation of a number in a base-3 notation.

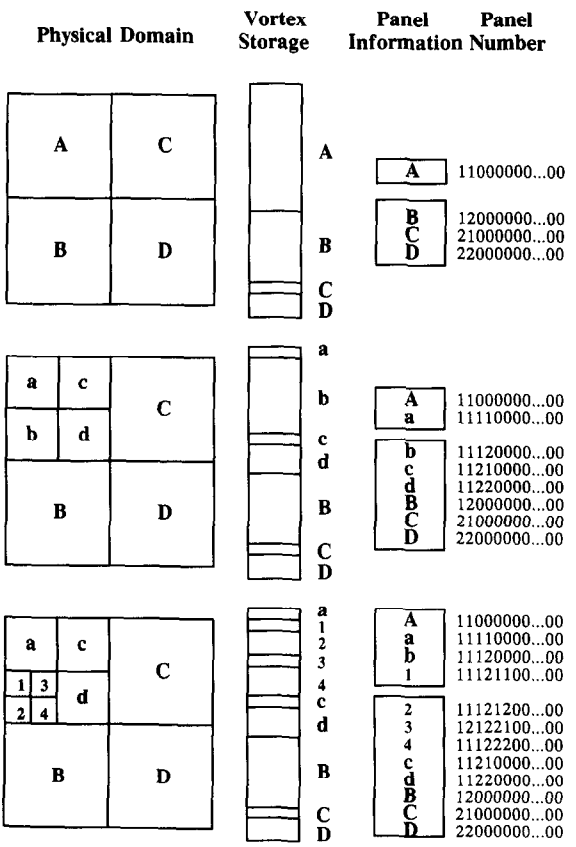

FIG. 3. The first three steps in generating the panel structure of Fig. 1. Shown are: the subsequent divisions of the domain, the order in which the vortices are stored, the order in which the information about the panels is stored, and the numbering of the panels.

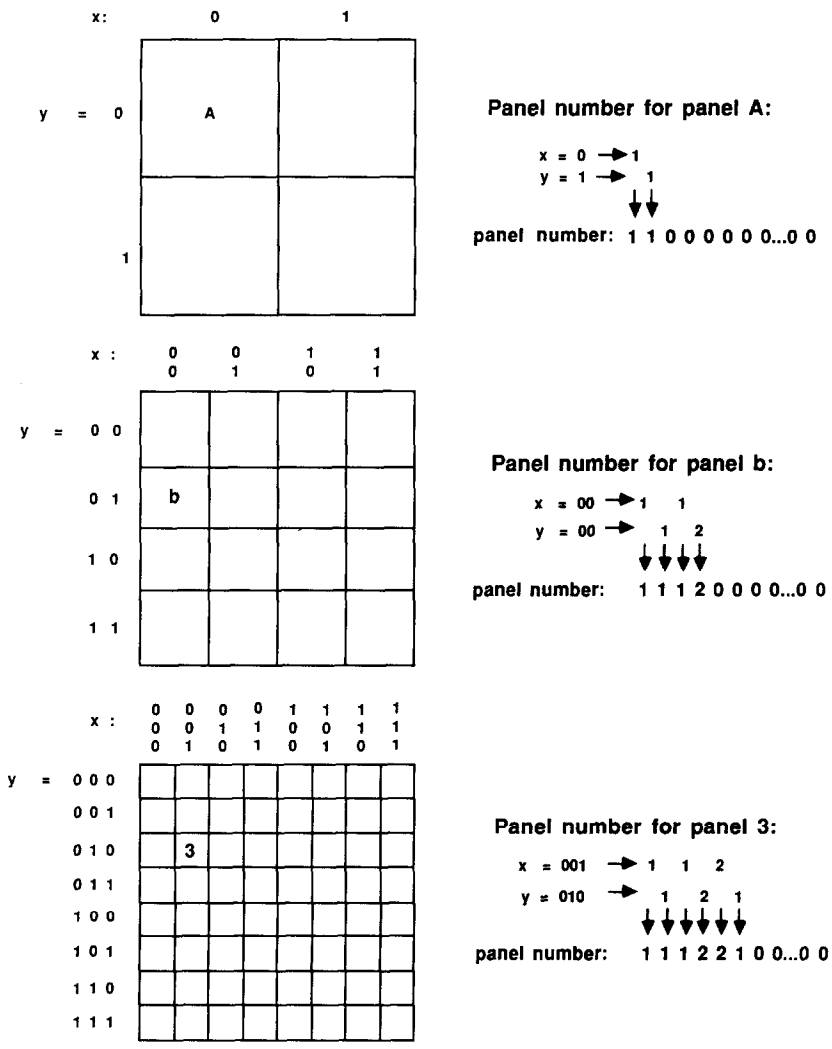

FIG. 4. Definition of the panel number of example panels.

By construction, the panel number contains all the information about the panel: the non-zero digits determine the panel position and the number of pairs of non-zero digits the subdivision level. Particularly important properties are:

(i) For any given panel, the panel numbers of neighboring panels of the same size may be found by simply binary manipulations. (For example, to find the panel at the same y-position but the previous x-position, determine the odd non-zero digits of the panel number, giving the x-location, and do a binary subtraction of unity to find the digits of the sought panel number.)

(ii) For any given panel number, the panel number of the next larger "mother" panel containing the given panel is found by setting the last two non-zero digits to zero. (The last two non-zero digits were generated by the last panel subdivision.)

(iii) Subpanels of any panel have a panel number greater than the original panel, but less than the next panel of equal size. (The subpanels have the same leading digits as the original panel, but non-zero trailing digits.) This property implies that in order of increasing panel number, panels are arranged in "families," with the subpanels always immediately following the panels of which they are a part.

The algorithm for generating the panels described at the start of this section generates them in order of increasing panel number, subdividing the current lowest panel before moving on to the next panel.

Since each subdivision adds two more non-zero digits, the total number of digits in the panel number limits the smallest panel that can be defined. In the 205 implementation, this total number of digits was chosen to be 28, since 28-digit numbers are the largest base-3 numbers than can be stored in a single 205 memory location, saving storage and computational operations. In 28 digit representation, the smallest panel can be about 16,000 times smaller than the original domain, which would seem sufficient for most purposes.

3. Determination of the Velocity

The velocity is determined in a single pass over all panels in the order in which they were generated as described in the previous section. The procedure is outlined in Fig. 5.

For each panel, a "neighborhood" of vortices is established, consisting of the vortices both within the panel itself and in the panels, of at least equal size, sharing a boundary line or a corner point with the considered panel. The vortices in this neighborhood are not summed by the Laurent series expansion of the considered panel. This restriction ensures that the Laurent series converges exponentially. Instead, in evaluating the velocity induced on the neighborhood, the original sum in Eq. (2) is used. This sum is only performed for panels which are not further subdivided; for panels which are subdivided, the velocity is evaluated by means of the subpanels.

Laurent series can be used for all vortices outside the neighborhood of a panel. However, to reduce the computational effort, the Laurent series is only used for those vortices which cannot be evaluated by means of the Laurent series of the next larger "mother" panel: the single Laurent series of the mother panel is more efficient than the four Laurent series of its subpanels. Therefore, the Laurent series of any panel is used only for the vortices within the neighborhood of the mother, but outside the neighborhood of the panel itself.

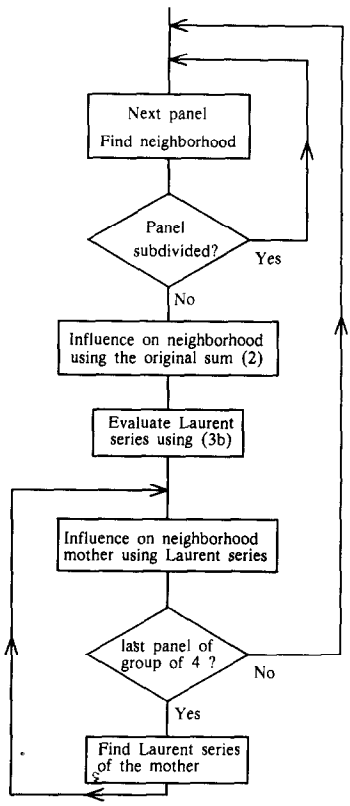

FIG. 5. Basic flow chart for the determination of the velocity.

In this scheme, at each stage the smallest possible number of Laurent series is used, for N vortices resulting in the $O(N \ln N)$ operation counts of the next section. In the procedure of Greengard and Rokhlin [7] this operation count is further reduced to $O(N)$ by recasting the Laurent series as Taylor series; however, the present procedure has the advantage of being less complex and requires only a single sweep over the panel structure to evaluate the velocity.

To incorporate the Taylor series within the present procedure, the evaluation of the neighborhood of the mother would have to modified. For each suitable panel within this neighborhood, the sum (3a) would be replaced by a recasting of the Laurent series into a Taylor series. Additional steps would be needed to transfer the Taylor series of the larger panels to the subpanels and to add the contributions of these series to the velocity.

Clearly, this will increase program complexity and scalar overhead. In addition, it requires that the neighborhood of the mother is described in terms of individual panels. The present procedure describes this neighborhood in terms of a small number of vectors of vortices, increasing vectorization. Furthermore, the present

procedure has a storage advantage: when the final subpanel of any group of 4 is reached, the four Laurent series of the subpanels can be combined into the Laurent series of the mother (bottom of Fig. 5). The four Laurent series of the subpanels can then be discarded; they are no longer needed. As a result, at any time only a small fraction of the Laurent series need be stored. On the other hand, using Taylor series, no obvious way to avoid storing the Taylor series coefficients for each panel is evident. This can be a disadvantage since each series represents a set of coefficients while, in addition, the total number of panels may be difficult to estimate precisely beforehand.

In the actual implementation of the procedure in Fig. 5, the first step is identification of the neighborhood of each panel. The present procedure starts out by identifying the individual digits of the binary x- and y-positions of the panel. By performing unit binary additions and subtractions, the panel numbers of the eight neighboring panels of the same size are found. For each of these eight panel numbers, the corresponding panel is located. In case any of the eight panels is undefined, the panel with the largest panel number less than or equal to the sought one is selected. On behalf of the properties of the panel number, the selected panel will always enclose the sought panel, ensuring the geometric convergence of the Laurent series. Since the panel numbers are ordered, an appropriate search on a scalar machine is binary; the CYBER 205 implementation switches to the vector function Q8SLT when the search interval extends over less than 500 panels.

After the neighboring panels have been located, the storage locations of the vortices in the neighborhood are simply the combination of the storage locations of the vortices in each of the nine panels. The subdivision process of the previous section reordered the vortices so as to group vortices in the same panel in contiguous storage locations or vectors. As a result, the neighborhood is described by at most nine vectors of vortices, and an additional check is made to identify contiguous vectors which can be described by a single vector. (In particular, the four subpanels of the larger panel containing the considered panel describe a single vector of vortices.) Since the number of vortices per panel is never small, the computations remain efficient on the 205.

The next step in the procedure in Fig. 5 is the evaluation of the original sum in Eq. (2) for panels which are not further subdivided. This sum was split into real and imaginary parts and modified to:

$$u_j = \sum_i g_i \frac{y_j - y_i}{(x_j - x_i)^2 + (y_j - y_i)^2 + d_i^2} \tag{4a}$$

$$v_j = \sum_i g_i \frac{x_i - x_j}{(x_j - x_i)^2 + (y_j - y_i)^2 + d_i^2} \tag{4b}$$

$$g_i = \frac{\Gamma_i}{2\pi}. \tag{4c}$$

These expressions are equivalent to the original sum in Eq. (2) when the value of

d_i vanishes. They are equivalent to the sum in Eq. (2) to machine precision when the value of d_i equals the machine epsilon. The addition of the d_i-term in (4a) and (4b) has the advantage of avoiding the singularity in the $i=j$ term while limiting the effect of numerical inaccuracy. For larger values of d_i, the velocity corresponds to vortices with finite core, which tend to improve the numerical properties of a vortex representation [9]. Expressions more elaborate than Eqs. (4a) through (4c) could be used [10]; since they increase the computational time for the original algorithm, they are likely to enhance the relative performance of the present algorithm.

The coefficients of the Laurent series follow from the sum (3b). The coefficients may be split into real and imaginary parts A_k and B_k, leading to the following recursive relationships:

$$a_i^1 = 0, \qquad b_i^1 = g_i \tag{5a), (5b)}$$

$$a_i^{k+1} = a_i^k(x_i - x_0) - b_i^k(y_i - y_0) \tag{5c}$$

$$b_i^{k+1} = a_i^k(y_i - y_0) + b_i^k(x_i - x_0) \tag{5d}$$

$$A_k = \sum_i a_i^k, \qquad B_k = \sum_i b_i^k. \tag{5e), (5f)}$$

In order to avoid possible inaccuracy caused by underflow of terms, the x- and y-positions were measured from the center of the panel and normalized with half the linear panel dimension.

For the evaluation of the neighborhood of the mother panels in Fig. 5, the Laurent series (3a) is used. Split into real and imaginary parts, the series can be written:

$$U_j^1 = \frac{x_j - x_0}{(x_j - x_0)^2 + (y_j - y_0)^2} \tag{6a}$$

$$V_j^1 = \frac{y_0 - y_j}{(x_j - x_0)^2 + (y_j - y_0)^2} \tag{6b}$$

$$U_j^{k+1} = \frac{U_j^k(x_j - x_0) + V_j^k(y_j - y_0)}{(x_j - x_0)^2 + (y_j - y_0)^2} \tag{6c}$$

$$V_j^{k+1} = \frac{V_j^k(x_j - x_0) - U_j^k(y_j - y_0)}{(x_j - x_0)^2 + (y_j - y_0)^2} \tag{6d}$$

$$u_j = \sum_{k=1} A_k U_j^k - B_k V_j^k \tag{6e}$$

$$v_j = -\sum_{k=1} A_k V_j^k + B_k U_j^k. \tag{6f}$$

It can readily be shown that the terms induced by any individual vortex i converge geometrically with a convergence ratio

$$\frac{|t_k|}{|t_{k+1}|} = \left|\frac{Z - Z_0}{Z_i - Z_0}\right| \tag{7}$$

if Z_0 is the position of the center of the panel. Since the vortices in the eight neighboring panels are excluded from the Laurent series, simple geometry shows that the convergence ratio is at least $3/\sqrt{2}$. Therefore, truncating the Laurent series at 22 terms, each vortex would be summed to a relative erro $6 \cdot 10^{-8}$; about the machine accuracy in half precision on the CYBER 205.

The last step in the procedure in Fig. 5 is the evaluation of the Laurent series of the current mother panel. While this Laurent series could be found using Eq. (3b), it can be found more efficiently from the Laurent series of the subpanels. The contribution of each of the subpanels to the Laurent series of the mother is given by

$$\Delta C'_k = m^k_k C_k + m^k_{k-1} C_{k-1} + \cdots + m^k_1 C_1 \tag{8a}$$

$$m^k_k = 2^{-k} \tag{8b}$$

$$m^k_{k-l} = m^k_{k-l+1} \frac{k-l}{l} \frac{1}{\sqrt{2}} H, \tag{8c}$$

where

$$H = -1 + \sqrt{-1} \tag{8d}$$

$$H = -1 - \sqrt{-1} \tag{8e}$$

$$H = 1 + \sqrt{-1} \tag{8f}$$

$$H = 1 - \sqrt{-1} \tag{8g}$$

for the first through the fourth subpanels, respectively. (The factor 2^{-k} in the above expressions reflects the scaling of Laurent series proportional to the panel size.) The coefficients m^k_{k-l} can be evaluated a priori after which the evaluation of the coefficients vectorizes.

4. Performance

The numerical performance of the present algorithm is difficult to analyze in general. In the following, the analysis has been simplified by assuming that the N vortices are homogeneously distributed over a square. In that case, the domain will be subdivided in panels containing the same number of vortices, n, each. The number of unsubdivided panels is N/n and the number of levels of subdivisions needed is $\log_4(N/n)$.

The total computational time fo find the velocity of the N vortices consists of a number of contributions. First of all, the vortices must be gathered into panels. The first subdivision of the total domain involves N vortices which are first examined on x-position, then on y-position, and correspondingly reordered. The time involved in this step will be written as

$$Nv_G f_{G,N}.$$

The factor v_G is the time needed to compare the x- and y-positions of a vortex with those of the center of the panel, pass the vortex 4 times through the vector function Q8VCMPRS (in the 205 implementation, otherwise to store the vortex twice), and add one to the number of vortices in first the right half of the panel and then to the number of vortices in the subpanel, using Q8SCNT.

The penalty factor $f_{G,N}$ expresses the overhead performed which is independent of the number of vortices involved, such as computing and storing the panel information for the four panels and, on the 205, starting up the vector operations. For a large number of vortices, the operations for the individual vortices dominate the total time and $f_{G,N}$ will approach unity. However, for vector processors such as the 205, the vector operations for the individual vortices are performed with such a speed that $f_{G,N}$ becomes appreciable when the number of vortices becomes less than a few hundred. (For the simple vector operations in half precision on the FSU/DOE 205, the start-up overhead becomes equivalent to the time of execution when the number of vortices is 200). The subscript N in the penalty factor $f_{G,N}$ expresses the representative number of elements or vector length.

In the next level of subdivision, four panels with each $\frac{1}{4}N$ vortices are subdivided, requiring a computational time

$$\tfrac{1}{4}Nv_G f_{G,N/4} \cdot 4 = Nv_G f_{G,N/4}.$$

Since there are $\log_4(N/n)$ levels of subdivisions and the penalty factor increases with decreasing vector length, the total time for finding the panels may conservatively be written as:

$$t_G = Nv_G f_{G,n} \log_4 \left(\frac{N}{n}\right). \tag{9a}$$

The logarithmic factor may be bounded by the maximum number of subdivisions allowed by the machine accuracy [8], but such a bound depends on the particular coding techniques and machine accuracy available and will be avoided here.

In the present algorithm, the original sum in Eq. (2) is used to evaluate the velocity induced by the n vortices in each unsubdivided panel upon its neighborhood of nine panels. If v_Σ is the time needed to evaluate a single term in Eq. (2), the total time can be written, conservatively, as

$$t_\Sigma = n \cdot 9n \cdot v_\Sigma f_{\Sigma,n} \cdot \frac{N}{n}, \tag{9b}$$

neglecting panels that may fall outside the domain, or

$$t_\Sigma = 9n\left[1 - \frac{4}{3}\sqrt{\frac{n}{N}} + \frac{4}{9}\frac{n}{N}\right] v_\Sigma f_{\Sigma,n} N,$$

after correction. Since n will typically be sizably smaller than N, Eq. (9b) will be used.

For each of the unsubdivided panels, K coefficients of the Laurent series (3b) must be determined, requiring a time

$$t_C = n \cdot K \cdot v_C f_{C,n} \cdot \frac{N}{n}. \tag{9c}$$

These Laurent series are next used to determine the velocity induced on the $4 \cdot 9n$ vortices in the neighborhood of the larger panel containing the unsubdivided panel, excluding the $9n$ vortices in the neighborhood of the unsubdivided panel itself. The time needed for N/n unsubdivided panels is

$$K \cdot 27n \cdot v_L f_{L,n} \frac{N}{n}.$$

Similarly, the Laurent series for the $N/4n$ larger panels are used to evaluate the velocity induced upon $27 \cdot 4n$ vortices, neglecting edge effects. With $\log_4(N/4)$ levels of subdivision, the time can be written conservatively as:

$$t_L = K \cdot 27n \cdot v_L f_{L,n} \frac{N}{n} \log_4\left(\frac{N}{n}\right). \tag{9d}$$

The procedure of Greengard and Rokhlin [7] avoids the logarithmic factor, since the number of coefficients in the Taylor series does not increase with panel size. However, the logarithmic factor in (9a) would remain.

Time is further needed to combine the Laurent series of the unsubdivided panels into these of the larger panels. In the 205 implementation, each coefficient C_k of the larger panel was written as an inner product between the vector of $4K$ coefficients of the subpanels and a corresponding vector of coefficients m_I^k, Eqs. (8a) through (8g). The time needed is:

$$t_I = \frac{4}{3}\frac{K^2}{n} v_I f_{I,4K} N. \tag{9e}$$

The contribution most difficult to estimate is the overhead involved in addressing the panel structure. For each panel, the neighborhood needs to be established, as well as the neighborhood of the next larger panel. Most of the operations involved will roughly be proportional to the number of panels: N/n unsubdivided ones, $N/4n$ next larger ones, and so on, a total of less than $4N/3n$ panels. However, the binary

search to find the nine neighboring panels requires operations proportional to $\log_2(4N/3n)$. The time for overhead will therefore be written as

$$t_0 = \left[\frac{4}{3}s_0 + \frac{8}{3}s_s \log_4\left(\frac{N}{n}\right)\right]\frac{N}{n}, \tag{9f}$$

where s_0 and s_s are representative computational times for each panel and for each binary search, respectively, neglecting $\log_2 4/3$. The symbol s was used here instead of v in order to indicate that the operations involved are largely scalar.

Using these various contributions to the computational time, the decision when to stop subdivision of the panels can be addressed. Collecting all contributions, the total time needed to find the velocity becomes:

$$t = \left(v_G f_{G,n} + 27 v_L f_{L,n} K + \frac{8}{3}\frac{s_s}{n}\right) N \log_4 \frac{N}{n} + \left(9 v_\Sigma f_{\Sigma,n} n + v_C f_{C,n} K + \frac{4}{3} v_I f_{I,4K} \frac{K^2}{n} + \frac{4}{3}\frac{s_0}{n}\right) N. \tag{10}$$

In estimating the relative importance of the terms, it will be assumed that the number of vortices N is large. Indeed, the number of vortices must be sizably larger than the typical number of terms in the Laurent series in order for the algorithm to be useful.

Under the limiting process where both N and the number of vortices per panel n tend to infinity, corresponding to relatively few large panels, the dominant term in Eq. (10) is the time, Eq. (9b), for the original sum, as could be expected. Since this term is proportional to n, decreasing the number of vortices per panel leads to corresponding reductions in computation time.

However, when decreasing the number of panels, adverse affects must eventually occur. The penalty factor $f_{\Sigma,n}$ increases when the value of n decreases, since the start up time increases in relative importance. On a two pipe 205, the vector start up becomes dominating when the number of vortices becomes less than 200, limiting further reductions in the time needed for the original sum.

On the other hand, the time needed for other operations increases while n decreases. For example, the time for doing the Laurent series (9d) increases when n decreases below a certain limit, since the penalty factor increases. This term contains the relatively large numerical factor $27K$, so that appreciable increases in the penalty factor tend to be important. In addition, the scalar times, which can be relatively large on a 205, are inversely proportional to n.

It may be concluded that for sufficiently many vortices, the computational time first decreases with the number of vortices per panel and then increases. As a result, a number of vortices per panel exists for which the present algorithm performs optimally.

For that reason, in generating the panels, the present algorithm decides whether

to subdivide panels further based on the number of vortices in the panel. Further subdivisions are only made when the number of vortices is greater than some minimum value n, chosen a priori.

Table I provides examples of the influence of the value of n on the computational time. In this case, the vortices were approximately homogeneously distributed over the interior of a circle, grouped in rings. It appears from Table I that the minimum number of vortices to subdivide a panel on a 205 should be roughly 200. Fortunately, the precise value used appears to have relatively little influence on the results.

In addition to the computational time, the numerical errors in the algorithm are important. For p vortices of strength Γ located on a ring of radius R, the velocity induced is

$$V_j^* = \sqrt{-1}\,\frac{p\Gamma}{2\pi Z_j}\,\frac{Z_j^p}{Z_j^p - R^p}$$

TABLE I

Computational Time and Numerical Errors for Vortices Homogeneously Distributed within a Circle Using 23 Term Asymptotic Expansions

Number of vortices	1000	2000	4000	8000	16,000	32,000	64,000
Time for summation, CPU seconds							
Original	0.10	0.36	1.38	5.39	21.35	86.32	356.78
100	0.11	0.35	0.71	1.95	3.55	9.35	17.17
200	0.11	0.27	0.67	1.61	3.55	8.12	17.10
400	0.10	0.27	0.82	1.61	4.58	8.12	22.53
Ratio of improvement							
200	0.9	1.3	2.0	3.3	6.0	10.6	20.9
Maximum error in the velocity, percent							
Original	0.005	0.019	0.020	0.040	0.080	0.160	0.319
100	0.004	0.007	0.009	0.013	0.016	0.022	0.024
200	0.004	0.006	0.009	0.013	0.016	0.021	0.024
400	0.005	0.006	0.012	0.043	0.021	0.021	0.030
Mean square error in the velocity, percent							
Original	0.003	0.006	0.012	0.023	0.046	0.093	0.187
200	0.002	0.004	0.005	0.007	0.009	0.012	0.013
Average error in the velocity, percent							
Original	0.003	0.005	0.010	0.020	0.041	0.082	0.164
200	0.002	0.003	0.005	0.006	0.008	0.010	0.012

or

$$V_j^* = \sqrt{-1}\,\frac{(p-1)\Gamma}{4\pi Z_j}$$

if the vortex j is located on the same ring. By summation over all rings the analytical solution can be found and compared to the obtained results.

Table I list the maximum deviation in either velocity component from the analytical solution, expressed in a percentage of the velocity at the perimeter of the circle. The present algorithm shows considerably better accuracy than the original sum, which may be due to the summation of the terms in groups. In the original algorithm, the individual terms were added to an increasingly large total, leading to a loss of significant digits.

For random walk computations, the mean square or average errors may be more relevant than the maximum error, since only averaged quantities are relevant. Both these errors show behavior similar to the maximum error.

TABLE II

As Table I, but Using 13 Term Expansions

Number of vortices	1000	2000	4000	8000	16,000	32,000	64,000
Time for summation, CPU seconds							
Original	0.10	0.36	1.38	5.39	21.36	86.35	356.72
100	0.09	0.26	0.57	1.40	2.79	6.61	13.03
200	0.09	0.25	0.55	1.07	2.79	6.91	13.08
400	0.10	0.25	0.79	1.43	4.27	6.92	20.40
Ratio of improvement							
100	1.1	1.4	2.4	3.9	7.7	13.1	27.4
Maximum error in the velocity, percent							
Original	0.005	0.010	0.020	0.040	0.080	0.160	0.319
100	0.003	0.005	0.007	0.009	0.011	0.013	0.015
200	0.003	0.006	0.007	0.011	0.011	0.016	0.015
400	0.005	0.006	0.011	0.012	0.019	0.016	0.025
Mean square error in the velocity, percent							
Original	0.003	0.006	0.012	0.023	0.046	0.093	0.187
100	0.002	0.003	0.004	0.005	0.006	0.007	0.009
Average error in the velocity, percent							
Original	0.003	0.005	0.010	0.020	0.041	0.082	0.164
100	0.002	0.002	0.003	0.004	0.005	0.006	0.008

The results in Table I were obtained by expanding all Laurent series to machine precision, 23 terms. Yet in view of the final errors in the results, there appears little justification in demanding such accuracy, unless special provisions are made to avoid accumulation of the round-off errors. Results for 13 term Laurent series are presented in Table II. Remarkably, the resulting errors prove somewhat lower than those in the 23 term expansion. A good explanation of this effect cannot be given; however, the maximum possible error in truncating the Laurent series is only 0.006%, small compared to the final errors. On the other hand, the final terms in the Laurent series correspond to the fastest Fourier components: for that reason truncating the series may have some averaging effect on the round-off errors.

For the case of Tables I and II, the vorticity occupied most of the domain under consideration. A somewhat different case arises when the vortices are evenly spaced along the perimeter of a circle. Since the vorticity is now sparsely distributed, the present procedure will generate a considerable number of empty panels, and it

TABLE III

Computational Time and Numerical Errors for Vortices Homogeneously Distributed on a Circle Using 13 Term Asymptotic Expansions

Number of vortices	1000	2000	4000	8000	16,000	32,000	64,000
Time for summation, CPU seconds							
Original	0.10	0.36	1.38	5.41	21.38	86.32	356.75
50	0.08	0.17	0.36	0.78	1.65	3.53	7.30
100	0.06	0.14	0.31	0.65	1.42	3.04	6.28
200	0.06	0.17	0.34	0.75	1.51	3.36	7.19
Ratio of improvement							
100	1.7	2.6	4.5	8.3	15.1	28.4	56.8
Maximum error in the velocity, percent							
Original	0.024	0.061	0.128	0.271	0.547	1.100	—
50	0.012	0.023	0.046	0.093	0.180	0.360	0.702
100	0.012	0.024	0.046	0.094	0.181	0.361	0.702
200	0.013	0.026	0.049	0.098	0.184	0.363	0.705
Mean square error in the velocity, percent							
Original	0.005	0.010	0.018	0.041	0.079	0.143	—
100	0.004	0.008	0.012	0.031	0.059	0.096	0.149
Average error in the velocity, percent							
Original	0.001	0.002	0.003	0.006	0.012	0.024	—
100	0.000	0.001	0.001	0.001	0.002	0.002	0.002

might seem that this would adversely affect computational time. However, the data in Table III shows that performance improves. This appears to agree with the observations of Carrier, Greengard, and Rokhlin [8].

The simplified case of vortices arranged on a single horizontal line can shed some light on this increase in efficiency. Repeating the previous analysis with suitable modifications, the total time becomes:

$$t = \left[2v_G f_{G,n} + 6v_L f_{L,n} K + 8 \frac{s_s}{n} \right] N \log_4 \frac{N}{n} + \left[3v_\Sigma f_{\Sigma,n} n + v_C f_{C,n} K + 4v_I f_{I,4K} \frac{K^2}{n} + 4 \frac{s_0}{n} \right] N. \tag{12}$$

Comparison with Eq. (10) does show that the time for scalar panel overhead has increased. On the other hand, the time needed for the direct summation has improved, since the empty panels decrease the number of vortices in the neighborhood of unsubdivided panels from $9n$ to $3n$. In addition, the time for the Laurent series has decreased, since $6n$ rather than $27n$ vortices need to be

TABLE IV

Computational Time and Numerical Errors for Vortices on the Perimeter of a Circle for a MicroVAX II

Number of vortices	400	800	1600	3200	6400	12,800	25,600
Time for summation, CPU seconds							
Original	9.1	36.3	151.9	614.2	2470.8	9867.3	39469.2
30	5.0	10.9	24.5	56.3	127.0	272.3	604.1
40	4.7	10.6	25.1	55.4	128.5	286.3	597.8
50	4.7	10.6	25.0	55.5	133.5	292.5	616.3
60	4.7	10.6	26.4	55.7	138.9	303.3	635.6
Mean square error in the velocity, percent							
Original	0.00000	0.00000	0.00001	0.00004	0.00007	0.00019	0.00036
40	0.00001	0.00001	0.00001	0.00002	0.00004	0.00010	0.00018
Additional array storage used, 4 byte words							
40	496	3289	3907	5040	7281	11374	20827
Percentage CPU time for various steps							
t_O	62	50	48	43	42	41	34
t_{L_1}	30	14	8	9	9	7	7
t_{L_2}	0	26	34	38	40	43	50

determined from the Laurent series of the unsubdivided panels; the remaining contributions are found using the combined Laurent series.

Next, while scalar operations are relatively slow on the 205, the corresponding total times are proportional to $1/n$, which is of order 10^{-2}. On the other hand, the times for the direct sum and the Laurent series are proportional to $9n$ and $27K$, respectively, which are of order 10^2. For that reason, the scalar times need not dominate even for quite slow scalar processing.

While the present algorithm still leads to reduced errors, the differences are not so pronounced as in Tables I and II. The reason may be that in this case the vortices in the immediate vicinity of each other dominate the errors (the sum approximates a singular integral). Those vortices are still summed in the same way as in the original algorithm.

Table IV shows computational times for a scalar version of the algorithm. As may be expected, the scalar version can address smaller groups of vortices more efficiently than the 205 version, resulting in some additional savings.

The total time CPU time used can be divided into contributions t_0 to perform the original sum (2), t_{L_1} to perform the Laurent series of panels which are not further subdivided, and t_{L_2} for the Laurent series of panels which are. Table IV shows those contributions for the case that $n = 40$. Recasting of Laurent series into Taylor series as used by Greengard and Rokhlin [7] could be used to reduce the required time t_{L_2}.

Comparison of Table IV with the data of Carrier, Greengard, and Rokhlin [8] does suggest a significant increase in storage due to such a recasting.

5. Concluding Remarks

The present algorithm is concerned with fast solution of the 2-dimensional Poisson equation under pointwise forcing. Since the actual application is not the true subject of this paper, only a concise description of the one considered here will be given: Lagrangian flow computations using a random walk simulation of diffusion effects similar to [9]. A relatively simple removal of the singular behavior, Eq. (4), was used in computed examples such as Fig. 1. In most computations, the chosen vortex diameter was 0.675 times the random step size $\sigma = \sqrt{2\nu\,\Delta t}$. The normal boundary condition was satisfied by means of mirror vortices within the cylinder. To satisfy the tangential boundary condition, after each predictor-corrector step all vortices within a distance $1.27\,\sigma$ from the wall, a thin sub-layer of the boundary layer, were removed. Next the slip velocity at the wall was evaluated and integrated to find the amount of circulation needed to satisfy the no-slip condition. This circulation was subsequently assigned to a ring of vortices a distance $0.675\,\sigma$ away from the wall and spaced $1.27\,\sigma$ apart. The procedure leads to the same flux of vortices through the cutoff at $1.27\,\sigma$ as a homogeneous distribution of vortices within the cutoff. In order to reduce the random fluctuations introduced by strong vortices, the number of vortices placed at each location along the wall was chosen

to give an approximately uniform vortex strength. At least one vortex was placed at each location if the local circulation was non-zero. Some experiments varying the given numerical values, or using an exponential vortex core rather than Eq. (4), were performed, but results were ambiguous due to the random noise. The random step sizes were taken from a data base of 8000 random numbers, starting from a randomly chosen position.

The purpose off this paper was to show how the computational time can be greatly reduced using Laurent series, allowing a much larger number of vortices to be included. The use of Laurent series or replacement elements to save computational time is not a new notion [6]; however, the present method renders the application effective by gathering the point forces into an adaptive, ordered panel structure. The contribution of the present paper is therefore primarily a programming technique which allows an easily addressable adaptive description of irregular distributions of points. Moreover, it is quite suited for vector processing and requires little storage. It seems simpler and possibly more vectorizable than the procedure of Carrier, Greengard, and Rokhlin [8].

The evidence of the Tables I through IV shows that the present algorithm is "fast" in the sense that the computational time roughly doubles when the number of vortices doubles. For the original sum in Eq. (2), the computational time becomes larger by a factor four instead. For that reason the savings in computational time increase with the number of vortices.

In fact, the time estimates in Eqs. (10) and (12) show the computational time to be proportional to $N \log N$, similar to the fast Fourier transform solutions of the Poisson equation such as Hockney's FACR algorithm, which needs $N \log_2(\log_2 N)$ operations. However, a closer study of Eqs. (10) and (12) shows that for typical values $K \sim 20$ and $n \sim 100$, the coefficient of the $N \log N$ term in the present algorithm will be numerically quite large.

For that reason, one of the motivations mentioned in the Introduction should still be present in order to adopt an algorithm such as the present one.

An interesting question is whether the present algorithm is applicable to 3-dimensional Poisson problems. This would make it possible to address such problems as the motion of stars in galaxies and 3-dimensional flows with sparse vortex geometry. Most of the procedures in Sections 2 and 3 carry through immediately by the simple step of including the digits of the third coordinate in the panel numbers. However, the straightforward generalization of the Laurent series to spherical harmonics as applied by Greengard and Rokhlin [11] has the disadvantage that the number of terms added for each order of accuracy increases. A procedure based on fast Fourier transforms proposed by Greengard and Rokhlin [12] can significantly reduce the effort.

The present procedure of generating and addressing a complex panel structure does not need to be restricted to solution of the Poisson equation, but could be used for other problems involving groups of points in which the interaction between elements of different groups can be simplified when the distance between the groups is sufficient.

ACKNOWLEDGMENTS

The authors acknowledge the support of the Supercomputer Computations Research Institute and the Florida State University through use of its 205. During parts of this investigation, the first author was supported by NASA Ames and the AFOSR, and the second author by the SCRI.

REFERENCES

1. A. LEONARD, *Annu. Rev. Fluid Mech.* **17**, 523 (1985).
2. C. R. ANDERSON, *J. Comput. Phys.* **62**, 111 (1936).
3. A. W. APPEL, *SIAM J. Sci. Stat. Comput.* **6**, 85 (1985).
4. J. BARNES AND P. HUT, *Nature* **342**, 446 (1986).
5. L. L. VAN DOMMELEN AND E. A. RUNDENSTEINER, "Adaptive-panel vortex summation for the CYBER 205," 38th Annual Meeting of the Division of Fluid Dynamics of the American Physical Society, Nov. 22–24, 1985.
6. V. ROKHLIN, *J. Comput. Phys.* **60**, 187 (1985).
7. L. GREENGARD AND V. ROKHLIN, *J. Comput. Phys.* **73**, 325 (1987).
8. J. CARRIER, L. GREENGARD, AND V. ROKHLIN, Yale University, Department of Computer Science Report No. YALEU/DCS/RR-496, 1987; *SIAM J. Sci. Stat. Comput.* **9**, No. 4 (1988).
9. A. J. CHORIN AND P. S. BERNARD, *J. Comput. Phys.* **37**, 423 (1973).
10. J. T. BEALE AND A. J. MAJDA, *J. Comput. Phys.* **58**, 188 (1984).
11. L. GREENGARD AND V. ROKHLIN, Yale University, Department of Computer Science Report No. YALEU/DCS/RR-515, 1987 (unpublished).
12. L. GREENGARD AND V. ROKHLIN, Yale University, Department of Computer Science Report No. YALEU/DCS/RR-602, 1988 (unpublished).