A Lagrangian Method Based on the Voronoi Diagram for the Incompressible Navier Stokes Equations on a Periodic Domain

Christoph Börgers and Charles S. Peskin

1. Introduction

We consider a fractional step method for the solution of

$$\underline{u}_t + (\underline{u}\cdot\nabla)\underline{u} - \nu\Delta\underline{u} + \nabla p = \underline{f} \ , \quad \nabla\cdot\underline{u} = 0 \tag{1.1}$$

($\underline{u} = \underline{u}(\underline{x},t)$, $p = p(\underline{x},t)$, $\underline{x} \in \mathbb{R}^2$, $t \in \mathbb{R}$, $t > 0$) with the initial condition

$$\underline{u}(\underline{x},0) = \underline{u}_0(\underline{x}) \tag{1.2}$$

and the periodicity conditions

$$\underline{u}(\underline{x} + \underline{k},t) = \underline{u}(\underline{x},t), \ p(\underline{x} + \underline{k},t) = p(\underline{x},t) \ , \ \underline{k} \in Z^2. \tag{1.3}$$

We intend to use this method in problems involving immersed boundaries, modelling the boundaries by a chain of Lagrangian particles connected by springs [10]. In this context, it seems natural to use Lagrangian particles for the fluid as well.

In section 2, we describe a way of adapting the algorithm for the construction of Voronoi meshes suggested in [9] to the case of a periodic domain. We believe that this algorithm can be used in three dimensions with slight modifications. It can also be used as a triangulation algorithm which finds a triangulation with a prescribed set S of corners: The dual graph of the Voronoi mesh (associated with the set S) is such a triangulation, called the "Delaunay triangulation".

In [9], Peskin described discretizations of the Laplace, gradient and divergence operators on arbitrary sets of points in the plane. We review the derivation of these operators in section 3, and we prove that the discrete divergence and gradient operators are weakly consistent with the corresponding continuous operators.

In section 4, we describe a two-level iteration for the solution of discrete Helmholtz equations and present results of numerical experiments with this method

which show that the efficiency of the method is similar to the efficiency of multigrid algorithms for the Helmholtz equation discretized on regular meshes.

In section 5, we discuss the problem of projecting a vector field onto its divergence-free component. We discuss an iterative algorithm which is applicable on unstructured meshes and appears to find quickly a good approximation; convergence to the exact solution of the discrete projection problem is guaranteed but slow.

Finally, we combine these tools to obtain a fractional step method for the Navier-Stokes equations (section 6). Preliminary numerical experiments indicate that we obtain a method which is "of first order" in the sense that the truncation error is roughly reduced by 1/2 if the total number N of fluid markers is multiplied by 4 and the time step is reduced by 1/2. (In two space dimensions, the effective mesh width is $O(N^{-1/2})$.) The work per time step required is O(NlogN) but could easily be reduced to O(N). The method is implicit: there is no stability condition on the size of the time step.

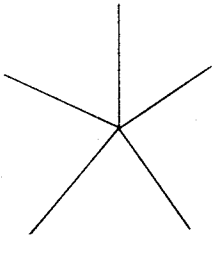
2. Construction of periodic Voronoi diagrams

Let S be a set of points in $\mathbb{R}^2$. For $\underline{x} \in S$, we define

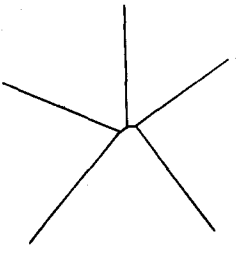
$$P(\underline{x},S) := \{\underline{y} \in \mathbb{R}^2 : \|\underline{y} - \underline{x}\| \leq \|\underline{y} - \tilde{\underline{x}}\| \text{ for all } \tilde{\underline{x}} \in S\} . \tag{2.1}$$

$P(\underline{x},S)$ is an intersection of half planes, i.e. a convex polygon. It is called the "Voronoi polygon" [11] or "Dirichlet region" of $\underline{x}$ with respect to S. The collection of all $P(\underline{x},S)$, $\underline{x} \in S$, is called the "Voronoi diagram" associated with S. We denote it by V(S).

Interior edges in V(S) are "generated" by two points, i.e. they belong to two polygons. Corners are "generated" by three or more points, i.e. belong to three or more polygons. We always assume that every corner is generated by three points. A degenerate case such as



is "resolved" into



This "resolution" of "multiple" corners into sets of "simple" corners linked by edges of length zero is always possible but never unique.

In [7], Shamos and Hoey described a general algorithm for the construction of Voronoi diagrams requiring O(NlogN) operations (N = number of points in S). This is known to be the best possible operation count for the general case.

If additional information on S is available, better operation counts can be achieved. The algorithm in [9] makes use of the assumption that the Voronoi diagram has already been constructed for a set $\tilde{S}$ which is "close to S", and that certain pieces of information about this construction have been saved. The algorithm used in [5] can be viewed as an algorithm which uses $V(\tilde{S})$ to compute V(S). Experiments indicate that both algorithms construct V(S) in linear time.

If S is random with a known distribution, the optimal expected value of operations may be less than O(NlogN). This statement is also supported by numerical experience, see below.

We now consider the case

$$(2.2) \quad S = S_N := \{\underline{X}_j + \underline{k} : 1 \leq j \leq N,\ \underline{k} \in Z^2\} \text{ with } \underline{X}_j \in [0,1)^2 .$$

(In our fractional step method, the $\underline{X}_j$ will be fluid markers.) The corresponding Voronoi diagram can be viewed as a periodic Voronoi diagram in the plane or as a diagram on the torus

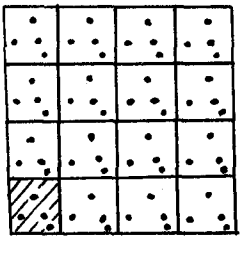
$$(2.3) \quad T := \mathbb{R}^2/Z^2$$

with the metric

$$(2.4) \quad d(\underline{x},\underline{y}) := \min_{\underline{k} \in Z^2} \| \underline{x} - \underline{y} + \underline{k} \| .$$

We describe a modified version of the algorithm in [9] suitable for this case.

It is convenient to use the following storage scheme for $V(S_N)$. We store information about the corners in $V(S_N)$ which lie in $[0;1)^2$. There are exactly 2N such corners.



the unit square

Proof: Consider m by m copies of the points $X_1, \cdots, X_N$:

Consider the corresponding Voronoi diagram. Intersect it with $[0;m]^2$. The result is a polygonal net; we extend this net to one on the sphere $\mathbb{R}^2 \cup \{\infty\}$ by introducing edges that connect the four corners of $[0;m]^2$ with ∞. This introduces a degenerate corner at ∞ which we resolve into two non-degenerate corners connected by an edge of length 0. The net on the sphere has $m^2N + 4$ faces. Since every edge has 2 corners and every corner has 3 edges, the number of corners is $\frac{2}{3}$ times the number of edges. From Euler's formula we now conclude that the total number of corners, including those at ∞, is $2m^2N + 4$.

The (m-2) by (m-2) inner squares are identical up to translation. They contain, say, $(m-2)^2M$ corners. (We want to show M=2N.) For fixed N, the number of corners in the 4m-4 boundary squares is O(m). Therefore

$$(m-2)^2M + O(m) = 2m^2N + 2 .$$

Letting $m \to \infty$, we obtain $M = 2N$.

We call the 2N corners in $[0;1)^2$ the "stored corners". We also call the points $\underline{X}_1,\cdots,\underline{X}_N$ "the stored points". For the j-th stored corner, we store its cartesian coordinates (XC(j),YC(j)) and the square RAD2(j) of its "radius". The "radius" of a corner is its distance from its generating points. Furthermore, we store integer indices

IPT(i,n,j) , i=1,2,3 , n=0,1,2 and
ICR(i,n,j) , i=1,2,3 , n=0,1,2

which have the following meaning: the i-th neighbor corner of the j-th stored corner is the ICR(i,0,j)-th stored corner, shifted by ICR(i,1,j) in the x-direction and by ICR(i,2,j) in the y-direction. The i-th generating point of the j-th stored corner is the IPT(i,0,j)-th stored point, shifted by IPT(i,1,j) in the x-direction and by IPT(i,2,j) in the y-direction.

We now outline the algorithm for the construction of $V(S_N)$:

Step 1: Construct $V(S_1)$.
Step 2: For j=2,•••,N:
  Step 2.1: Construct $V(S_{j-1} \cup \{\underline{X}_j\})$ from $V(S_{j-1})$.
  Step 2.2: Construct $V(S_j)$ from $V(S_{j-1} \cup \{\underline{X}_j\})$.

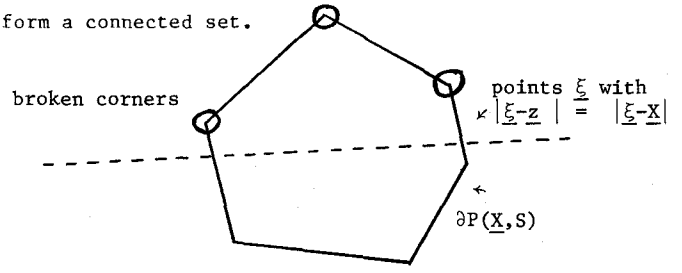
That is, in step 2.1 we add the stored point $\underline{X}_j$ without its periodic images to the otherwise periodic Voronoi diagram. Then in step 2.2 we add the periodic images of $\underline{X}_j$ (all at once).

Step 1 is trivial. The central part of the algorithm is a procedure for the construction of $V(S \cup \{\underline{z}\})$ if $\underline{z} \in S$ and V(S) is known (step 2.1). This procedure was described in [2] and in [9] independently. We also present it here.

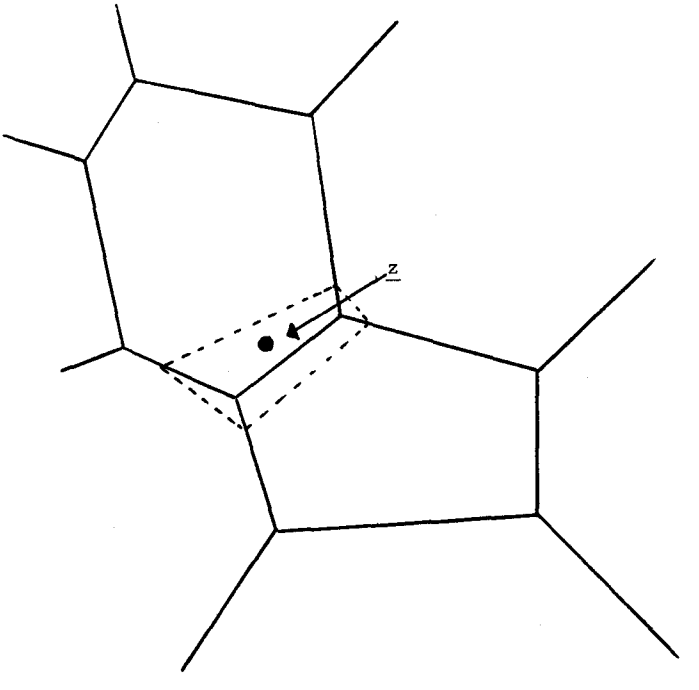
A corner $\underline{c}$ in V(S) is called "broken by $\underline{z}$" if it is not a corner in $V(S \cup \{\underline{z}\})$. $\underline{z}$ breaks $\underline{c}$ if and only if $\underline{z}$ lies in the open disk around $\underline{c}$ with radius $\mathrm{rad}(\underline{c})$, where $\mathrm{rad}(\underline{c})$ is the radius of $\underline{c}$. $\underline{z}$ always breaks at least one corner. (This is, in general, guaranteed only if $\underline{z}$ lies in the convex hull of S. But the convex hull of a set of the form (2.2) is $\mathbb{R}^2$.)

Furthermore, the set of broken corners is connected in V(S).

Proof: For a fixed polygon in V(S), those of its corners which are broken by $\underline{z}$ clearly form a connected set.

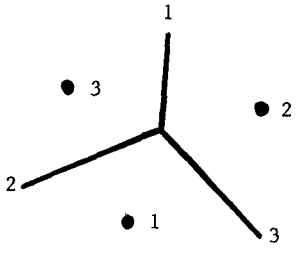


Walking along the boundary of $P(\underline{z},S \cup \{\underline{z}\})$, one walks through all poygons which have corners broken by $\underline{z}$. [This walk was first pointed out to us by D. Goldfarb, unpublished.] Whenever $\partial P(\underline{z},S \cup \{\underline{z}\})$ crosses an edge between two polygons, this edge has exactly one endpoint which is broken by $\underline{z}$. This endpoint is a common corner of the two polygons. Therefore, following $\partial P(\underline{z},S \cup \{\underline{z}\})$ one constructs a walk through all broken corners.



We can therefore find all broken corners in O(1) operations once we know one of them. We can find one broken corner in the following way. First choose some point $\underline{x} \in S$ close to $\underline{z}$. Good ways of choosing $\underline{x}$ will be discussed below. The efficiency of the algorithm mainly depends on how this choice is made. Determine a corner $\underline{c}$ of $P(\underline{x},S)$. (We use an array which contains for each polygon the basic and shift indices of one of its corners for this purpose.) Then search through the graph of corners of V(S), starting at $\underline{c}$, until a broken corner is found.

Once we know all corners broken by $\underline{z}$, we can find all new corners (the corners of $P(\underline{z},S \cup \{\underline{z}\})$) using the fact that each such corner lies on an edge between a broken and an unbroken corner in V(S) (and on each such edge there is a new corner). It is useful (but not necessary) here to store the neighbor corners of a corner $\underline{c}$ in counterclockwise order and to number the generating points such that the edge connecting $\underline{c}$ with its i-th neighbor corner lies opposite to the i-th generating point:



(This ordering seems to be the most important specifically two-dimensional feature of our code.)

It remains to describe step 2.2. To simplify the notation, we take j=N and define

(2.5) $\mathcal{P}_N := P(\underline{X}_N, S_{N-1} \cup \{\underline{X}_N\})$ and

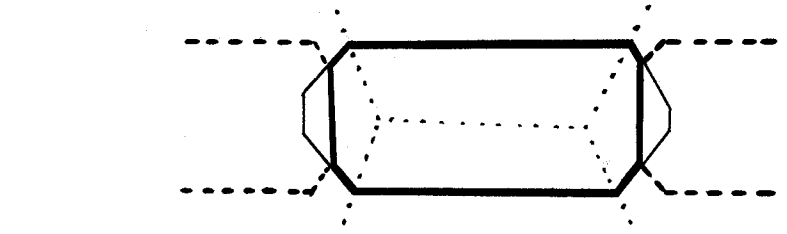
(2.6) $P_N := P(\underline{X}_N, S_N)$.

It is clear then how to get P_N from $\mathcal{P}_N$:

(2.7) $P_N = \mathcal{P}_N \cap ([X_{N1} - 0.5; X_{N1} + 0.5] \times [X_{N2} - 0.5; X_{N2} + 0.5])$.

(X_{N1} and X_{N2} are the coordinates of $\underline{X}_N$.)

We first assume that the decision which corners of $\mathcal{P}_N$ have to be "cut off" to obtain P_N is simply made based on (2.7), checking coordinates. It seems easy then to find the radii of the new corners in $V(S_N)$, their neighbor corners and their neighbor points, as indicated in the figure below:



edges in $V(S_{N-1})$

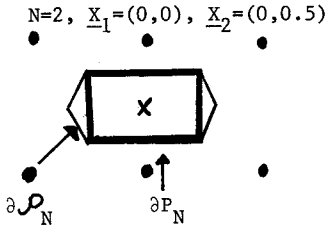
∂P_N

translations of edges of P_N

$\partial \mathcal{P}_N$

(To find the neighbors of the new corners, follow "·····" and "------".)

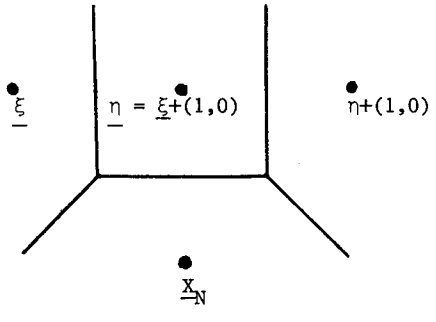
Therefore the description of step 2.2 seems to be completed. This procedure works correctly in most cases; occasionally it breaks down, though. We show a typical situation which leads to a breakdown:



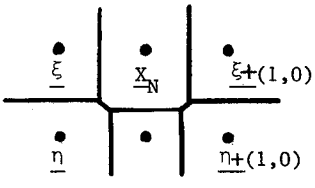
P_N should have exactly 6 corners, two of which are identical up to translation with two others. In arbitrarily but finitely accurate arithmetic, the computed number of corners of P_N may be 4, 5, 6, 7 or 8.

The situations which cause difficulties are those in which a polygon "wraps around" the torus; for $V(S_N)$, this means that a polygon is adjacent to its own translation. This is most likely to happen at early stages of the construction. It may even happen in $V(S_N)$ with arbitrarily large N.

We therefore have to "cut" $\mathcal{P}_N$ in a more careful way to obtain P_N. Let $\underline{c}$ be a corner of $\mathcal{P}_N$, generated by $\underline{\xi}$, $\underline{\eta}$, $\underline{X}_N$. Let $\underline{c}'$ be a corner of $\mathcal{P}_N$, generated by $\underline{\xi}$ + (1,0), $\underline{\eta}$ + (1,0), $\underline{X}_N$. The following figures show two (possible) such situations.

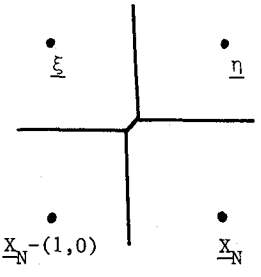


(a)

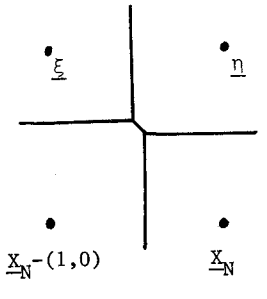


(b)

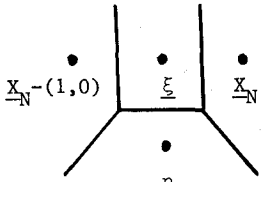
(Case (a) is the one discussed above as an example for a breakdown.) To decide whether $\underline{c}$ is broken by $\underline{X}_N - (1,0)$ in situation (a), we consider the Voronoi diagram $V(\{\underline{\xi};\underline{\eta};\underline{X}_N;\underline{X}_N - (1,0)\})$.



If we draw the diagram like this, we conclude that $\underline{c}$ is not broken by $\underline{X}_N - (1,0)$ (since it exists in the four-point Voronoi diagram). Clearly, we can use the same diagram to decide whether $\underline{c}'$ is broken by $\underline{X}_N + (1,0)$. We conclude that it is. If we had drawn the short edge of length 0 like this:



we would have concluded that $\underline{c}$ is broken whereas $\underline{c}'$ is not. In any case, we make opposite decisions for $\underline{c}$ and $\underline{c}'$, which is correct. We consider case (b) now. Neither $\underline{c}$ nor $\underline{c}'$ are broken here. Again, we can read this off from the Voronoi diagram $V(\{\underline{\xi};\underline{\eta};\underline{X}_N;\underline{X}_N - (1,0)\})$:



This motivates the following algorithm for "cutting":

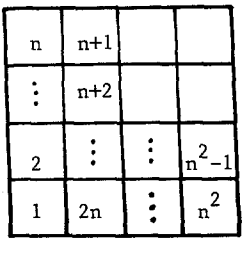
For each corner $\underline{c}$ of $\mathcal{P}_N$, decide whether or not it lies in $[X_{N1} - 0.5;\ X_{N1} + 0.5] \times [X_{N2} - 0.5; X_{N2} + 0.5]$, using the following procedure:
To see whether $\underline{c}$ is closer to $\underline{X}_N$ than to $\underline{X}_N + \underline{k}_0$, $\underline{k}_0 = \pm(1,0)$ or $\underline{k}_0 = \pm(0,1)$, compute

(2.8) $$V(\{\underline{\xi};\underline{\eta};\underline{X}_N;\underline{X}_N + \underline{k}_0\})$$

($\underline{X}_N,\underline{\xi},\underline{\eta}$: generating points of $\underline{c}$). If $\underline{c}$ is a corner in this diagram, it is closer to $\underline{X}_N$ than to $\underline{X}_N + \underline{k}_0$, otherwise it is broken by $\underline{X}_N + \underline{k}_0$. If $\underline{c}'$ is a corner of $\mathcal{P}_N$ generated by $\underline{X}_N,\underline{\xi} - k_0,\underline{\eta} - \underline{k}_0$, use the same diagram to decide whether or not $\underline{c}'$ is broken by $\underline{X}_N - \underline{k}_0$.

In our code, this procedure is simplified. The main simplification is a test whether cutting is necessary at all. (This can be done in a stable way by checking coordinates of corners of $\mathcal{P}_N$.) In the vast majority of cases, no cutting is necessary.

We finally describe two ways of finding a good starting point $\underline{x}$ for the search for the first broken corner. If the points in S are known to be roughly uniformly distributed in $[0;1]^2$, we can divide $[0;1]^2$ into n^2 square cells of equal size with $n^2 \approx N$ and create pointers from those cells to the points in S which they contain. (A linked list is a good data structure to use for this purpose: In each cell, store a pointer to one point in the cell. Then let each point in the cell point to another with the last point containing a stop code.) We then introduce the points cell by cell in the order indicated in the following figure:



A good choice of $\underline{x}$ when introducing $\underline{X}_j$ is then $\underline{X}_{j-1}$. Using this method on sets of points which are random and uniformly distributed in $[0;1]^2$, we obtained the following CPU-times per point in msec on a VAX 11/780:

N	worst case	best case	average
400	19	17	18
800	19	18	18
1200	19	18	18
1600	19	18	19

For each N, we tried 20 different random sets of points.

In our fractional step method, we use this method at time 0. At later times, we use the following method. Assume that $V(\tilde{S})$ has been constructed for a set $\tilde{S} = \{\tilde{\underline{X}}_1;\cdots;\tilde{\underline{X}}_N\} + Z^2$, with $d(\underline{X}_j, \tilde{\underline{X}}_j)$ small, $\tilde{\underline{X}}_j \in [0;1)^2$. (d was defined in (2.4).) When inserting the k-th point $\underline{X}_k$, we start the search for the nearest neighbor with the suitable translation of the j(k)-th point $\underline{X}_{j(k)}$, where j(k) is such that a corner of $P(\tilde{\underline{X}}_{j(k)},\tilde{S})$ was broken when $\tilde{\underline{X}}_k$ was inserted during the construction of $V(\tilde{S})$. (The indices $j(k) \in \{1;\cdots;k\}$ must therefore be saved during the construction of $V(\tilde{S})$.)

Using this method, the construction becomes slightly faster at later times than at time 0.

We conclude this section with some remarks about other possible ways of constructing periodic Voronoi diagrams. It is, of course, possible to reduce the problem to nonperiodic Voronoi diagrams (in the plane): Consider

$$(2.9) \qquad S^* := \{\underline{X}_j + \underline{k}:\ 1 \le j \le N,\ \underline{k} \in Z^2,\ |k_1| \le 1 \text{ and } |k_2| \le 1\}\ .$$

It is easy to see that

$$(2.10) \qquad P(\underline{X}_j,S^*) = P(\underline{X}_j,S) \text{ for all } j\ .$$

Since S^* contains 9N points, this procedure is quite inefficient. The definition of S^* can be modified to reduce the number of auxiliary points, but the algorithm becomes unnecessarily complicated then.

An excellent alternative is the use of an algorithm similar to the one used in [5], which constructs the Delaunay triangulation rather than the Voronoi diagram and seems clearly more efficient than our method. It is also easily vectorizable, which doesn't seem to be the case for our method. Our algorithm may, however, be easier to generalize to Voronoi diagrams in spaces other than $\mathbb{R}^2$ because it uses the metric of the space but no angles. For a spherical version see [1]. From a non-practical point of view, our algorithm has the advantage that it will provably work for any case. Such a proof does not yet exist for the algorithm in [5], while the statement is obvious in the case of our algorithm - at least in the absence of rounding errors.

3. Finite difference operators on arbitrary sets of points

We review the definitions of the discrete Laplace, divergence and gradient operators given in [9]. These definitions are dimension independent. We use the terminology appropriate for 2 dimensions for simplicity.

We introduce the following notations. For $\underline{X} \in S_N$, let $V[\underline{X}]$ be the area of the Voronoi polygon $P(\underline{X},S_N)$. For $\underline{Y} \in S_N$, $\underline{Y} \neq \underline{X}$, let $A[\underline{X},\underline{Y}]$ be the length of the intersection of $P(\underline{X},S_N)$ and $P(\underline{Y},S_N)$. $A[\underline{X},\underline{Y}] = 0$ if $P(\underline{X},S_N)$ and $P(\underline{Y},S_N)$ are not adjacent to each other. If $A[\underline{X},\underline{Y}] \neq 0$, we say that $\underline{X}$ and $\underline{Y}$ are "neighbors" of each other. Our discrete operators couple two distinct points to each other only if they are neighbors. In the following, we let $Nb[\underline{X}]$ be the set of neighbors of $\underline{X}$. (Note that $\underline{X} \in Nb[\underline{X}]$).

We define a discrete Laplace operator L by

$$(3.1) \qquad V[\underline{X}](L\phi)(\underline{X}) := \sum_{\underline{Y}\neq\underline{X}} A[\underline{X},\underline{Y}](\phi(\underline{Y})-\phi(\underline{X}))/\|\underline{X}-\underline{Y}\| \quad \text{for } \underline{X} \in S_N ,$$

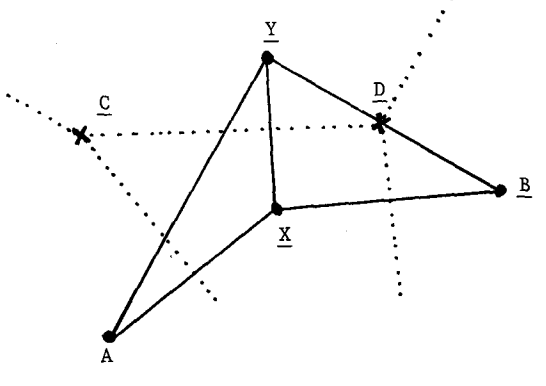
which is motivated by the formula

$$(3.2) \qquad \int_P \Delta\phi(\underline{x})\, d\underline{x} = \int_{\partial P} \frac{\partial\phi}{\partial n}(\underline{x})\, ds$$

(for smooth functions ϕ). The sum in (3.1) is formally infinite but contains only finitely many non-zero terms. As was pointed out by B. Mercier ([6]), the operator in (3.1), seen as a matrix, is the stiffness matrix obtained with piecewise linear finite elements on the Delaunay triangulation.

Proof: Both matrices have zero row sums. Therefore we need to consider non-diagonal entries only. Since both matrices are also symmetric, it makes sense to talk about the "coupling between $\underline{X}$ and $\underline{Y}$" ($\underline{X},\underline{Y} \in S_N$, $\underline{X} \neq \underline{Y}$). We compare the couplings in the matrix in (3.1) and in the finite element stiffness matrix.

Without loss of generality, we assume that $\underline{X} = (0,0)$, $\underline{Y} = (0,1)$. Let $\underline{A}$, $\underline{B}$ be the points in S_N such that $\underline{A}$, $\underline{X}$, $\underline{Y}$ and $\underline{B}$, $\underline{X}$, $\underline{Y}$ are triangles in the Delaunay triangulation:



Let $\underline{C}$ be the center of the circle through $\underline{A}$, $\underline{X}$, $\underline{Y}$, and let $\underline{D}$ be the center of the circle through $\underline{B}$, $\underline{X}$, $\underline{Y}$. $\underline{C}$ and $\underline{D}$ are corners in $V(S_N)$. The claim is that the coupling between $\underline{X}$ and $\underline{Y}$ obtained with piecewise linear finite elements is $\|\underline{C}-\underline{D}\|$. This is shown by straightforward computation of this coupling and $\|\underline{C}-\underline{D}\|$ in terms of the coordinates of $\underline{A}$ and $\underline{B}$. Both turn out to be equal to

$$(3.3) \qquad -\frac{A_2^2 - A_2}{2\,A_1} - \frac{A_1}{2} + \frac{B_2^2 - B_2}{2\,B_1} + \frac{B_1}{2}$$

with $\underline{A} =: (A_1, A_2)$ and $\underline{B} =: (B_1, B_2)$.

Nevertheless, a discretization of the Poisson equation based on (3.1) is not a finite element discretization. The difference lies in the right-hand sides. Let f be a given continuous right-hand side. The value in $\underline{X}$ of the discrete right-hand side obtained with the finite element method is a weighted sum of values of f in neighbors of $\underline{X}$. The sum $M[\underline{X}]$ of the weights (elements of the mass matrix) is one third times the sum of the areas of the triangles to which $\underline{X}$ belongs. Using (3.1), one is lead to using $V[\underline{X}]f(\underline{X})$ as the value of the discrete right-hand side in $\underline{X}$. $V[\underline{X}]$ and $M[\underline{X}]$ are, in general, different from each other. They are, of course, equal "on the average". ($M[\underline{X}]$ is the area of a cell suggested in [4].)

We define a discrete divergence operator next.

$$(3.4) \qquad V[\underline{X}]D\underline{u}(\underline{X}) := \sum_{\underline{Y}} \underline{u}[\underline{Y}] \cdot \frac{\partial V[\underline{X}]}{\partial \underline{Y}} \quad \text{for } \underline{X} \in S_N .$$

Here $\frac{\partial V[\underline{X}]}{\partial \underline{Y}}$ is the gradient of $V[\underline{X}]$ with respect to $\underline{Y}$ (keeping all other points fixed). If $(\underline{X}_1, \cdots, \underline{X}_N) = (\underline{X}_1(t), \cdots, \underline{X}_N(t))$ are moving points, it follows that

$$(3.5) \qquad \frac{d}{dt} V[\underline{X}_j(t)] = V[\underline{X}_j(t)]D\underline{u}(\underline{X}_j(t))$$

for all j, in particular:

If $\underline{X}_1(t), \cdots, \underline{X}_N(t)$ are points in space which move in a discretely divergence-free velocity field, the Voronoi polygons maintain their areas exactly.

(However, the Voronoi polygons do change their areas, usually even quite drastically, if the points are moved in a continuously divergence-free field.)

An explicit formula for $\frac{\partial V[\underline{X}]}{\partial \underline{Y}}$ may be derived as follows. For a given $\underline{Y}$, there are three cases to consider:

i) $\underline{X} = \underline{Y}$

ii) $\underline{X} \in Nb[\underline{Y}]$ (so $\underline{X} \neq \underline{Y}$)

iii) $\underline{X} \neq \underline{Y}$ and $\underline{X} \notin Nb[\underline{Y}]$.

It is obvious that

$$(3.6) \qquad \frac{\partial V[\underline{X}]}{\partial \underline{Y}} = 0 \text{ in case (iii)} .$$

Case (i) may be reduced to case (ii) as follows. From (3.6), it is easy to conclude that

$$(3.7)\qquad \frac{\partial}{\partial \underline{Y}} \left(V[\underline{Y}] + \sum_{\underline{X} \in \mathrm{Nb}(\underline{Y})} V[\underline{X}] \right) = 0 \ ,$$

hence

$$(3.8)\qquad \frac{\partial}{\partial \underline{Y}} V[\underline{Y}] = - \sum_{\underline{X} \in \mathrm{Nb}(\underline{Y})} \frac{\partial V[\underline{X}]}{\partial \underline{Y}} \ .$$

One can also derive a formula dual to (3.8) by considering translation invariance of the Voronoi diagram on the torus. Let the points $\underline{X}_1 \cdots \underline{X}_N$ all move at the same constant velocity $\underline{u}$. Clearly, the diagram translates as a whole and the areas of the polygons do not change. Thus

$$(3.9)\qquad 0 = \frac{d}{dt} V[X] = \sum_{\underline{Y}} \frac{\partial V[X]}{\partial \underline{Y}} \cdot \underline{u}$$

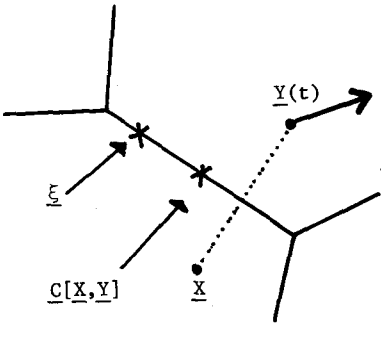
Since $\underline{u}$ was arbitrary

$$(3.10)\qquad 0 = \sum_{\underline{Y}} \frac{\partial V[X]}{\partial \underline{Y}}$$

$$(3.11)\qquad \frac{\partial V[X]}{\partial \underline{X}} = - \sum_{\underline{Y} \neq \underline{X}} \frac{\partial V[X]}{\partial \underline{Y}} = - \sum_{\underline{Y} \in \mathrm{Nb}[\underline{X}]} \frac{\partial V[X]}{\partial \underline{Y}}$$

Note that (3.8) and (3.11) are different formulae for the diagonal derivative. They are both written in terms of $\partial V[\underline{X}]/\partial \underline{Y}$, but (3.8) involves a sum over $\underline{X}$ and (3.11) a sum over $\underline{Y}$. Either of these formulae may be used to express the diagonal derivatives in terms of the off-diagonal derivatives.

We now consider case (ii) in which $\underline{Y}$ is a neighbor of $\underline{X}$.



Suppose $\underline{Y}$ is moving and $\underline{X}$ is fixed. Also, all other points are fixed. Let $\underline{\xi}$ be an arbitrary point on the edge common to $P[\underline{X}]$ and $P[\underline{Y}]$. Recall that this edge has length $A[\underline{X},\underline{Y}] \neq 0$. Let $\underline{C}[\underline{X},\underline{Y}]$ be the midpoint (center of mass) of this edge. Now the arbitrary point $\underline{\xi}$ satisfies the equation

$$(3.12) \qquad \|\underline{\xi} - \underline{X}\|^2 = \|\underline{\xi} - \underline{Y}\|^2$$

Differentiating this with respect to t, we get

$$(3.13) \qquad \frac{d\underline{\xi}}{dt} \cdot (\underline{\xi} - \underline{X}) = (\frac{d\underline{\xi}}{dt} - \frac{d\underline{Y}}{dt}) \cdot (\underline{\xi} - \underline{Y})$$

$$(3.14) \qquad \frac{d\underline{\xi}}{dt} \cdot (\underline{Y} - \underline{X}) = \frac{d\underline{Y}}{dt} \cdot (\underline{Y} - \underline{\xi})$$

Dividing this expression by $\|\underline{Y} - \underline{X}\|$, integrating over the edge in question, and noticing that $(\underline{Y} - \underline{X})/\|\underline{Y} - \underline{X}\|$ is the unit normal to this edge, we obtain

$$(3.15) \qquad \frac{d}{dt} V[\underline{X}] = \int \frac{d\underline{\xi}}{dt} \cdot \frac{\underline{Y} - \underline{X}}{\|\underline{Y} - \underline{X}\|} ds = \frac{d\underline{Y}}{dt} \cdot \int \frac{\underline{Y} - \underline{\xi}}{\|\underline{Y} - \underline{X}\|} ds$$

where ds is the element of length along the edge. But $\int ds = A[\underline{X},\underline{Y}]$ and $\int \underline{\xi} ds = A[\underline{X},\underline{Y}]\underline{C}[\underline{X},\underline{Y}]$. Therefore

$$(3.16) \qquad \frac{d}{dt} V[\underline{X}] = \frac{d\underline{Y}}{dt} \cdot \frac{\underline{Y} - \underline{C}[\underline{X},\underline{Y}]}{\|\underline{Y} - \underline{X}\|} A[\underline{X},\underline{Y}]$$

Since $\partial \underline{Y}/\partial t$ was arbitrary, it follows that

$$(3.17) \qquad \frac{\partial V[\underline{X}]}{\partial \underline{Y}} = \frac{\underline{Y} - \underline{C}[\underline{X},\underline{Y}]}{\|\underline{Y} - \underline{X}\|} A[\underline{X},\underline{Y}] \ , \ \underline{X} \in Nb[\underline{Y}]$$

For future reference we resolve this into normal and tangential components:

$$(3.18) \qquad \underline{Y} - \underline{C}[\underline{X},\underline{Y}] = (\underline{Y} - \frac{1}{2}(\underline{X} + \underline{Y})) + (\frac{1}{2}(\underline{X} + \underline{Y}) - \underline{C}[\underline{X},\underline{Y}])$$

$$= \frac{1}{2}(\underline{Y} - \underline{X}) + (\frac{1}{2}\underline{X} + \underline{Y}) - \underline{C}[\underline{X},\underline{Y}])$$

Therefore

$$(3.19) \qquad \frac{\partial V[\underline{X}]}{\partial \underline{Y}} = \frac{1}{2} A[\underline{X},\underline{Y}] \frac{\underline{Y} - \underline{X}}{\|\underline{Y} - \underline{X}\|} + \frac{\underline{T}[\underline{X},\underline{Y}]}{\|\underline{Y} - \underline{X}\|} A[\underline{X},\underline{Y}]$$

where

$$(3.20) \qquad \underline{T}[\underline{X},\underline{Y}] = \frac{1}{2}(\underline{X} + \underline{Y}) - \underline{C}[\underline{X},\underline{Y}]$$

is tangent to the edge in question.

We now make use of the foregoing results to prove that the operator D is weakly consistent (to first order) with the continuous divergence operator. That is, for arbitrary smooth, periodic (scalar and vector) functions ϕ and $\underline{u}$, we shall prove that

$$(3.21)\qquad \sum_k V[\underline{X}_k]\phi(\underline{X}_k)D\underline{u}(\underline{X}_k) = \int_{[0,1]^2} \phi(\nabla\cdot\underline{u})dV + O(h)$$

where

$$(3.22)\qquad h = \operatorname*{Max}_{\underline{x}} \operatorname*{Min}_{\underline{X}\in S} \|\underline{x} - \underline{X}\|$$

(That is, h is the maximum "radius" of the polygons in the diagram.) To show (3.21), we rewrite the left-hand side using the definition of D (Eq. 3.4) and the identity (3.8), which is used here with $\underline{X}$ and $\underline{Y}$ interchanged:

$$\begin{aligned}
&\sum_k V[\underline{X}_k]\phi(\underline{X}_k)D\underline{u}(\underline{X}_k)\\
&= \sum_k \phi(\underline{X}_k) \sum_{\underline{Y}\in S_N} \frac{\partial V[\underline{X}_k]}{\partial \underline{Y}} \cdot \underline{u}(\underline{Y})\\
&= \sum_k \phi(\underline{X}_k) \sum_{\underline{Y}\in Nb[\underline{X}_k]} \{\frac{\partial V[\underline{X}_k]}{\partial \underline{Y}} \cdot \underline{u}(\underline{Y}) - \frac{\partial V[\underline{Y}]}{\partial \underline{X}_k} \cdot \underline{u}(\underline{X}_k)\}
\end{aligned}$$

Next, we substitute into this the formula for $\partial V[\underline{X}]/\partial \underline{Y}$, Eq. 3.19:

$$\begin{aligned}
(3.24)\qquad &\sum_k V[\underline{X}_k]\phi(\underline{X}_k)D\underline{u}(\underline{X}_k)\\
&= \sum_k \phi(\underline{X}_k) \sum_{\underline{Y}\in Nb[\underline{X}_k]} A[\underline{X}_k,\underline{Y}] \frac{\underline{Y} - \underline{X}_k}{\|\underline{Y} - \underline{X}_k\|} \cdot \frac{1}{2}(\underline{u}(\underline{Y}) + \underline{u}(\underline{X}_k))\\
&+ \sum_k \phi(\underline{X}_k) \sum_{\underline{Y}\in Nb[\underline{X}_k]} A[\underline{X}_k,\underline{Y}]\underline{T}[\underline{X}_k,\underline{Y}] \cdot \frac{\underline{u}(\underline{Y}) - \underline{u}(\underline{X}_k)}{\|\underline{Y} - \underline{X}_k\|}
\end{aligned}$$

We analyze these two terms separately. In the last term, since A[X,Y] and $\underline{T}[X,Y]$ are symmetric, we have the equivalent expression

$$(3.25)\qquad \frac{1}{2}\sum_k \sum_{\underline{Y}\in Nb[\underline{X}_k]} (\phi(\underline{X}_k) - \phi(\underline{Y}))A[\underline{X}_k,\underline{Y}]\underline{T}[\underline{X}_k,\underline{Y}] \cdot \frac{\underline{u}(\underline{Y}) - \underline{u}(\underline{X}_k)}{\|\underline{Y} - \underline{X}_k\|}$$

Now $\phi(\underline{X}_k) - \phi(\underline{Y}) = O(h)$, $A[\underline{X}_k,\underline{Y}] = O(h)$, $\underline{T}[\underline{X}_k,\underline{Y}] = O(h)$, and $(\underline{u}(\underline{Y}) - u(\underline{X}_k))/\|\underline{Y} - \underline{X}_k\| = O(1)$. Therefore, each individual term in (3.25) is $O(h^3)$. Moreover, the number of terms is only $O(N) = O(h^{-2})$, since the number of elements in $Nb[\underline{X}_k]$ is $O(1)$. Therefore, the entire expression (3.25) is $O(h)$. In summary, we have shown that

$$(3.26)\qquad \sum_k V[\underline{X}_k]\phi(\underline{X}_k)D\underline{u}(\underline{X}_k)$$
$$= \sum_k \phi(\underline{X}_k) \sum_{\underline{Y}\in Nb[\underline{X}_k]} A[\underline{X}_k,\underline{Y}] \frac{\underline{Y}-\underline{X}_k}{\|\underline{Y}-\underline{X}_k\|}\cdot\frac{1}{2}(\underline{u}(\underline{Y})+\underline{u}(\underline{X}_k)) + O(h)$$

The next step is to bring the integral $\int\phi(\nabla\cdot\underline{u})dV$ into a similar form. We have

$$(3.27)\qquad \int_{[0;1]^2}\phi(\nabla\cdot\underline{u})dV = \sum_k \phi(\underline{X}_k)\int_{P[\underline{X}_k]}(\nabla\cdot\underline{u})dV + O(h)$$
$$= \sum_k \phi(\underline{X}_k) \sum_{\underline{Y}\in Nb[\underline{X}_k]} A[\underline{X}_k,\underline{Y}] \frac{\underline{Y}-\underline{X}_k}{\|\underline{Y}-\underline{X}_k\|}\cdot\underline{u}^*[\underline{X}_k,\underline{Y}] + O(h)$$

where $\underline{u}^*[\underline{X}_k,\underline{Y}]$ is the average of $\underline{u}$ over the edge common to $P[\underline{X}_k]$ and $P[\underline{Y}]$. Subtracting Eq. (3.27) from (3.26) we get

(3.28)
$$\sum_k V[\underline{X}_k]\phi(\underline{X}_k)D\underline{u}(\underline{X}_k) - \int\phi(\nabla\cdot\underline{u})dV + O(h)$$
$$= \sum_k \phi(\underline{X}_k) \sum_{\underline{Y}\in Nb[\underline{X}_k]} A[\underline{X}_k,\underline{Y}] \frac{\underline{Y}-\underline{X}_k}{\|\underline{Y}-\underline{X}_k\|}\cdot\{\frac{1}{2}(\underline{u}(\underline{Y})+\underline{u}(\underline{X}_k)) - \underline{u}^*[\underline{X}_k,\underline{Y}]\}$$
$$= \frac{1}{2}\sum_k\sum_{\underline{Y}\in Nb[\underline{X}_k]} (\phi(\underline{X}_k)-\phi(\underline{Y}))A[\underline{X}_k,\underline{Y}] \frac{\underline{Y}-\underline{X}_k}{\|\underline{Y}-\underline{X}_k\|}\cdot\{\frac{1}{2}(\underline{u}(\underline{Y})+\underline{u}(\underline{X}_k)) - \underline{u}^*[\underline{X}_k,\underline{Y}]\}$$

As above, we have $\phi(\underline{X}_k) - \phi(\underline{Y}) = O(h)$, $A[\underline{X}_k,\underline{Y}] = O(h)$, $\{\frac{1}{2}(\underline{u}(\underline{Y}) + u(\underline{X}_k)) - \underline{u}^*[\underline{X}_k,\underline{Y}]\} = O(h)$, and $(\underline{Y}-\underline{X}_k)/\|\underline{Y}-\underline{X}_k\| = O(1)$. Thus, the individual terms are $O(h^3)$ and there are (as before) $O(N) = O(h^{-2})$ terms in the sum. It follows that the entire expression is $O(h)$ and hence that

$$(3.29)\qquad \sum_k V[\underline{X}_k]\phi(\underline{X}_k)D\underline{u}(\underline{X}_k) - \int_{[0;1]^2}\phi(\nabla\cdot\underline{u})dV = O(h)$$

as claimed.

This completes the proof that D and $\nabla\cdot$ are weakly consistent. In fact, they are <u>not</u> pointwise consistent. This was first noticed by M. McCracken, who constructed counterexamples (unpublished).

Note that the foregoing argument is exactly the same in three dimensions. We just have to define $\underline{C}[\underline{X}_k,\underline{Y}]$ as the "center of mass" of the face common to $P[\underline{X}_k]$ and $P[\underline{Y}]$, and $A[\underline{X}_k,\underline{Y}]$ as the area of that face. Then $A[\underline{X}_k,\underline{Y}] = O(h^2)$ and $N = O(h^{-3})$, so the extra factors of h cancel.

Next, we introduce an operator $\underline{G}$ which is the negative adjoint of D with respect to the discrete L^2 inner product

$$(3.30)\qquad (\phi,\psi) := \sum_k \phi(\underline{X}_k)\psi(\underline{X}_k)V[\underline{X}_k]$$

$$(3.31)\qquad (\underline{u},\underline{v}) = \sum_k \underline{u}(\underline{X}_k)\cdot\underline{v}(\underline{X}_k)V[\underline{X}_k]$$

The definition of $\underline{G}$ is

$$V[\underline{X}]\underline{G}\phi(\underline{X}) = -\sum_{\underline{Y}} \phi(\underline{Y}) \frac{\partial V[\underline{Y}]}{\partial \underline{X}} \tag{3.32}$$

It follows directly from the definitions of D and $\underline{G}$ that

$$(\underline{u},\underline{G}\phi) + (D\underline{u},\phi) = 0 \tag{3.33}$$

which is the analog of the continuous identity

$$\int (\underline{u}\cdot\nabla\phi)dV + \int (\nabla\cdot\underline{u})\phi dV = 0 \tag{3.34}$$

The weak consistency of $\underline{G}$ with the continous gradient operator follows directly from these identities and the weak consistency of D. That is, combining Eqs. (3.29), (3.33), and (3.34), we obtain

$$\sum_k V[\underline{X}_k]\underline{G}\phi(\underline{X}_k)\cdot\underline{u}(\underline{X}_k) = \int_{[0;1]^2} \underline{u}\cdot\nabla\phi \; dV + O(h)$$

4. Solution of discrete Helmholtz equations on the Voronoi mesh

In this section, we consider the fast numerical solution of

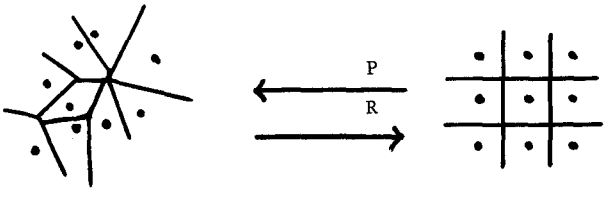
$$-L\phi + c\phi = f \quad (c \geq 0) . \tag{4.1}$$

We use a two-level iteration which is identical with the well-known multigrid correction cycle (see [8]), replacing coarsening of the grid by "regularization". We give a motivating derivation of the method here.

Consider a regular grid of the form

$$G^h = \{((i + 0.5)h,(j + 0.5)h) : i,j \in Z\} \tag{4.2}$$

with $h = 1/n$ and $n \approx \sqrt{N}$. Let R be a linear interpolation operator which maps functions defined on S_N onto functions defined on G^h. let P be a linear interpolation operator in the opposite direction.



We introduce the short notation

(4.3) $A := -L + cI$,

(4.4) $B :=$ discretization of $-\Delta + cI$ on the grid, using the standard 5-point operator.

We consider the following simple iterative method for (4.1).

(4.5) $\phi^{n+1} := \phi^n + PB^{-1}R(f - A\phi^n)$.

(If c=0, B^{-1} does not exist. "$B^{-1}r$" is obtained as follows in that case: Subtract the constant component from r, find some solution v of Bv = r and subtract the constant component from v.)

If P and R are chosen in a simple, straightforward way (see below), the iteration (4.5) converges very poorly if at all. The reason is that P and R introduce averaging, which prevents highly oscillatory erros from converging fast. Therefore (4.5) has to be supplemented by a method which is efficient on highly oscillatory errors; we choose a suitable relaxation method for this. We obtain then the two-level correction cycle

(4.6)
Step 1: m relaxation sweeps
Step 2: (4.5)
Step 3: l relaxation sweeps

m and l are small integers, typically 1 or 2. To solve problems with B (as required in (4.5)), we use the Fast Fourier Transform. This is the only piece in our fractional step method which costs O(NlogN) operations rather than O(N) operations per time step. Instead of Fast Fourier Transform, an approximate solver (for example a multigrid solver) requiring only O(N) operations could be used without significant deterioration of the convergence rates, see [8].

We have to choose P, R and the relaxation scheme. We discuss the choice of P and R first. We list some "desirable properties":

CP : P preserves constants:

$P(1)(\underline{X}_j) = 1$ for all j .

CR : R preserves constants:

$R(1)(\underline{x}) = 1$ for all $\underline{x} \in G^h$.

IOP : P preserves the property of having discrete integral 0:

If $\sum_{\underline{x} \in G^h} \phi(\underline{x})h^2 = 0$, then $\sum_k (P\phi)(\underline{X}_k)V[\underline{X}_k] = 0$.

IOR : R preserves the property of having discrete integral 0:

If $\sum_k \Phi(\underline{X}_k)V[\underline{X}_k] = 0$, then $\sum_{\underline{x}} (R\Phi)(\underline{x})h^2 = 0$.

IOP and IOR state the preservation of the compatibility condition if c=0; for our multigrid algorithm, it is desirable to have IOR at least, to ensure solvability of the problems on G^h if c=0.

We focus attention on operators defined in the form

$$(4.7) \qquad (P\phi)(\underline{X}_k) := \sum_{\underline{x} \in G^h} \phi(\underline{x})\delta_h(\underline{X}_k - \underline{x})h^2 ,$$

$$(4.8) \qquad (R\Phi)(\underline{x}) := \sum_k \Phi(\underline{X}_k)\delta_h(\underline{x} - \underline{X}_k)V[\underline{X}_k] ,$$

where δ_h is a spread-out model of the delta distribution with support of (linear) size $O(h)$. ((4.7), (4.8) imply $P=R^*$, where R^* is the adjoint of R with respect to the discrete L^2-products on S_N and on G^h.) It is clear then that CR cannot be satisfied in general. (It is possible that no point in S_N lies close to $\underline{x} \in G^h$, in which case $R\Phi(\underline{x})$ is 0 no matter what Φ is.) CP, on the other hand, is easy to satisfy, for example with

$$(4.9) \qquad \delta_h(\underline{x}) = \delta_h(x_1)\delta_h(x_2) , \text{ with}$$

$$(4.10) \qquad \delta_h(x) = \begin{cases} (1 - \frac{|x|}{h}) \text{ if } |x| \leq h , \\ 0 \text{ otherwise} \end{cases}$$

(piecewise bilinear interpolation). It is easy to conclude from this that IOR is satisfied while IOP is not. In fact, R preserves discrete integrals in general. These conclusions depend on nothing but the fact that $P = R^*$.

Eqs. (4.7) - (4.10) specify our choice of P and R.

It remains to choose a relaxation scheme. Using Gauss-Seidel relaxation, we obtain a performance which is disappointing in comparison with many multigrid methods on regular meshes. The following table contains the reduction of the discrete L^2-norm of the residual with Gauss-Seidel relaxation, m=2 and n=1, after

1,3 and 5 two-level iterations. For each N, we performed experiments on 10 different random sets S_N (uniformly distributed in the unit square). We show the worst and the best results as well as the geometric means over all 10 experiments. In all cases, the continuous problem being solved has the solution

(4.11) $$\phi(x) = \sin(2\pi(x - 2y)) .$$

The Helmholtz constant is c = 10.0.

N	worst case	best case	geometric mean
	0.22E+00	0.69E-01	0.13E+00
100	0.34E-02	0.37E-04	0.23E-02
	0.56E-03	0.28E-06	0.60E-04
	0.26E+00	0.16E+00	0.22E+00
400	0.17E-02	0.29E-03	0.78E-02
	0.26E-03	0.84E-05	0.57E-03
	0.29E+00	0.21E+00	0.25E+00
1600	0.18E-02	0.39E-03	0.86E-02
	0.30E-03	0.13E-04	0.76E-03

The results can be improved by a modification of the relaxation method. The modified relaxation scheme is defined as follows. Introduce the notation

(4.12) $$(L\phi)(\underline{X}_k) =: \sum_{\underline{Y}} L[\underline{X}_k, \underline{Y}]\phi(\underline{Y}) .$$

When "relaxing $\underline{X}_k$", determine the set of all neighbors $\underline{Y}$ of $\underline{X}_k$ for which

(4.13) $$L[\underline{X}_k, \underline{Y}] \geq -\delta\, L[\underline{X}_k, \underline{X}_k] ,$$

where $\delta \in [0;1]$ remains to be chosen. Change values in all these neighbors and in $\underline{X}_k$ simultaneously such that their equations become satisfied. If δ is at least 0.5, it follows that the blocks which are relaxed simultaneously are at most of size 2. (To see this, recall that the diagonal elements of L are negative, that the off-diagonal elements are positive or zero with at least 3 positive elements in each row, and that the row sums of L are zero. Thus there is at most one neighbor $\underline{Y}$ of any point $\underline{X}_k$ for which (4.13) is satisfied when $\delta \geq 0.5$.) Therefore we choose δ = 0.5. We repeat the above experiments with this modified relaxation scheme. The results, for c = 10.0, are as follows:

N	worst case	best case	geometric mean
	0.90E-01	0.30E-01	0.52E-01
100	0.55E-04	0.15E-05	0.81E-04
	0.70E-06	0.33E-07	0.77E-06
	0.15E+00	0.96E-01	0.12E+00
400	0.11E-03	0.17E-04	0.48E-03
	0.18E-05	0.18E-06	0.40E-05
	0.16E+00	0.13E+00	0.14E+00
1600	0.12E-03	0.28E-04	0.63E-03
	0.33E-05	0.60E-06	0.12E-04

The results for c = 0.0 look hardly different:

N	worst case	best case	geometric mean
	0.83E-01	0.33E-01	0.54E-01
100	0.70E-04	0.20E-05	0.10E-03
	0.98E-06	0.27E-07	0.71E-06
	0.16E+00	0.10E+00	0.12E+00
400	0.12E-03	0.19E-04	0.53E-03
	0.19E-05	0.19E-06	0.45E-05
	0.16E+00	0.13E+00	0.15E+00
1600	0.13E-03	0.29E-04	0.65E-03
	0.34E-05	0.72E-06	0.12E-04

(Here we have projected the discrete right-hand side onto its constant-free part to ensure that there is a solution.)

We still obtain convergence rates which seem to increase with growing N. Good multigrid methods have convergence rates which are bounded independent of the number of unknowns, the bound being far below 1; see [8]. We do not know whether the method presented here has this property. It is, in any case, satisfactory when used within our fractional step method, see section 6.

5. Numerical projection of a vector field onto its divergence free part

The last tool needed for our fractional step method is an algorithm which orthogonally projects a given vector field $\underline{u}$ on S_N onto the kernel of the discrete divergence operator D.

We want to find $\mathbb{P}\underline{u}$ and q such that

$$\underline{u} = \mathbb{P}\underline{u} + \underline{G}q \tag{5.1}$$

and

(5.2) $\quad D\underline{P}u = 0 .$

To do this exactly, we have to solve

(5.3) $\quad D\underline{G}q = Du .$

No difficulty arises from the fact that $D\underline{G}$ is singular. Du satisfies the compatiblity condition and $\underline{G}q$ is unique.

On a regular square grid with mesh width h, $D\underline{G}$ is the standard 5-point discretization of the Laplace operator with mesh width 2h. Simple relaxation schemes have no smoothing effect for this discretization, as can easily be verified by Fourier analysis. (We ignore the fact that the system can be decoupled into 4 smaller systems which can be solved easily, since this will not be the case on irregular meshes.) The difficulty is related to lack of "discrete ellipticity", see [3].

We therefore consider replacing the discrete projection operator

(5.4) $\quad I - \underline{G}(D\underline{G})^{-1}D$

by

(5.5) $\quad I - \underline{G}L^{-1}D .$

($D\underline{G}$ and L are singular; nevertheless "$G(D\underline{G})^{-1}D$" and "$GL^{-1}D$" have obvious well-defined meanings.)

(5.5) is not a projection operator. In fact, its discrete L^2-norm

(5.6) $\quad \|I - GL^{-1}D\| = \rho(I - \underline{G}L^{-1}D)$

is often larger than 1. (The eigenvalues with largest absolute value must be smaller than -1 if that is the case.) If this norm is smaller than 1, then

(5.7) $\quad \lim_{j\to\infty} (I - \underline{G}L^{-1}D)^j = I - \underline{G}(D\underline{G})^{-1}D .$

Since (5.5) is a discretization of the continuous projection operator, it is to be expected that eigenvalues corresponding to smooth eigenfunctions are larger than -1. This motivates the following modification of (5.5):

(5.8) $\quad I - \underline{G}(I + \omega L)^k L^{-1} D ,$

with $\omega \approx 1/\rho(L)$ but $\omega < 1/\rho(L)$, k integer. For a fixed k, $(I + \omega L)^k$ is consistent with the identity operator and (5.8) can therefore still be considered a discretization of the continuous projection operator. For sufficiently large k, the powers of (5.8) converge to $I - \underline{G}(D\underline{G})^{-1}D$.

This leads to the following algorithm.

Given $\underline{u}$, generate $\underline{u}^0, \underline{u}^1, \underline{u}^2, \cdots$ with $\underline{u}^j \to \mathbb{P}\underline{u}$
as follows:
$\underline{u}^0 := \underline{u}$.
Given $\underline{u}^j$, define
$k := 0$; $\underline{u}^{j,0} := \underline{u}^j$, $q^{j,1} := L^{-1}D\underline{u}^{j,0}$; $\underline{u}^{j,1} := \underline{u}^{j,0} - Gq^{j,1}$.

(5.9) While $\|\underline{u}^{j,k+1}\| > \|\underline{u}^j\|$:
$k := k+1$; $q^{j,k+1} := \omega L q^{j,k}$; $\underline{u}^{j,k+1} := \underline{u}^{j,k} - Gq^{j,k+1}$.
$\underline{u}^{j+1} := \underline{u}^{j,k+1}$.

In our experiments, this algorithm usually chooses $k = 0$ when computing $\underline{u}^1$ from $\underline{u}^0$, $k > 0$ for $j > 1$, sometimes even $k \gg 0$.

We give a numerical example. Consider

(5.10) $$\underline{u}(\underline{x}) = (\sin(2\pi x_1)\cos(2\pi x_2)\ ,\ 0)\ .$$

The divergence-free part of $\underline{u}$ is

(5.11) $$0.5(\sin(2\pi x_1)\cos(2\pi x_2)\ ,\ -\cos(2\pi x_1)\sin(2\pi x_2))\ .$$

Computing the divergence-free part of $\underline{u}$ numerically, we obtain truncation errors with the following discrete L^2-norms:

N	iteration	worst case	best case
25	1	0.35	0.21
	2	0.34	0.19
	3	0.36	0.18
	4	0.34	0.17
	5	0.37	0.17
	10	0.36	0.16
	20	0.37	0.16
100	1	0.17	0.12
	2	0.16	0.11
	3	0.16	0.11
	4	0.16	0.11
	5	0.16	0.11
	10	0.17	0.11
	20	0.17	0.11
400	1	0.079	0.067
	2	0.076	0.060
	3	0.077	0.062
	4	0.077	0.064
	5	0.078	0.064
	10	0.079	0.063
	20	0.078	0.065
1600	1	0.040	0.035
	2	0.040	0.033
	3	0.043	0.034
	4	*	0.035
	5	*	0.036
	10	*	0.037
	20	*	0.037

For each N, 10 sets of points were tested (random, uniformly distributed). "*" means: The iteration was aborted because k became larger than 50.

One iteration generates an approximation to the continuous solution which is as good or even better than the approximation obtained after more iterations. What we have gained by replacing (5.5) with (5.9) is guaranteed stability, not higher accuracy.

It is a surprising fact that the truncation error does decrease with growing N here, even though D is not pointwise consistent with the divergence operator. We also observe that the variance of the truncation error seems to decrease with growing N.

6. A fractional step method for the Navier-Stokes equations

We consider the following fractional step method for the Navier-Stokes equations:

$$
\begin{aligned}
&\frac{\tilde{u}^{n+1} - u^n}{\Delta t} - \nu\, L^n \tilde{u}^{n+1} = f^{n+1} \\
&u^{n+1} = P^n \tilde{u}^{n+1} \\
&X^{n+1} = X^n + \Delta t\, u^{n+1} .
\end{aligned} \tag{6.1}
$$

Here L^n denotes the discrete Laplacian on "S_N^n", the set S_N at time $n\Delta t$; the points in that set are the components of the infinite vector X^n. P^n denotes the orthogonal projection onto the kernel of D^n, where D^n is the discrete divergence operator on S_N^n.

Writing $P^n \tilde{u}^{n+1} =: \tilde{u}^{n+1} - G^n q^{n+1}$, we obtain

$$
\frac{u^{n+1} - u^n}{\Delta t} - \nu\, L^n \tilde{u}^{n+1} + \frac{G^n q^{n+1}}{\Delta t} = f^{n+1} \text{ and}
$$

$$
D^n u^{n+1} = 0 . \tag{6.2}
$$

We therefore call $p^{n+1} := q^{n+1}/\Delta t$ the "approximation for the pressure".

We notice that (6.1) is stable in the sense that

$$
\| u^{n+1} \|_{(n)} \leq \| u^n \|_{(n)} \tag{6.3}
$$

where "$\| \cdot \|_{(n)}$" denotes the discrete L^2-norm on S_N^n. This follows because L^n is symmetric and negative semidefinite in the corresponding inner product. Using the projection method described in section 5, stability is guaranteed even though we do not apply the operator P^n exactly.

As a test example, we use

$$
\begin{aligned}
u_1(x,t) &= \sin(\tfrac{\pi}{2}t)\cos(2\pi x_1)\sin(2\pi x_2) \\
u_2(x,t) &= -\sin(\tfrac{\pi}{2}t)\sin(2\pi x_1)\cos(2\pi x_2) \\
p(x,t) &= \sin(\tfrac{\pi}{2}t)\cos(2\pi x_1)\cos(2\pi x_2)
\end{aligned} \tag{6.4}
$$

with $\nu = 0.1$. We use

$$
\Delta t = 1/\sqrt{N} \tag{6.5}
$$

and measure discrete L^2-norms of the errors in the velocity at time 1. In every time step, we use only one step of the iteration (5.9). Initially, the X_j are at the positions $((k_1 + 0.5)h, (k_2 + 0.5)h)$, $0 \leq k_1 \leq \sqrt{N}-1$, $0 \leq k_2 \leq \sqrt{N}-1$. If all Helmholtz and Poisson problems are solved up to rounding error accuracy, we obtain:

N	error in u_1	error in u_2
100	0.35	0.35
400	0.12	0.12
1600	0.067	0.067

It is reasonable to expect that iteration up to rounding error accuracy is necessary only at time 0. At later time, the values from the previous time step should provide an excellent initial guess, and little work should be required to improve this guess such that the truncation error level is reached. We repeat our experiments with only one two-level cycle with m=2, l=1, confirming these considerations:

N	error in u_1	error in u_2
100	0.35	0.35
400	0.12	0.12
1600	0.065	0.065

Acknowledgement

This work was supported by the Department of Energy at the Courant Mathematics and Computing Laboratory of New York University under contract DE-AC02-76ER03077. Börgers was also supported by a Dean's Dissertation Fellowship at New York University.

Bibliography

[1] J.M. Augenbaum and C.S. Peskin: "On the Construction of the Voronoi Mesh on a Sphere"; in press. J. Comput. Phys.

[2] A.Bowyer: "Computing Dirichlet Tessellations"; Computer J. 24, 162, 1981.

[3] A. Brandt: "Guide to Multigrid Development"; in Multigrid Methods, Lecture Notes in Mathematics, vol. 961, Springer-Verlag, 1982.

[4] J. Dukowicz: "Lagrangian Fluid Dynamics Using the Voronoi-Delaunay Mesh"; in Numerical Methods for Coupled Problems, Pineridge Press, Swansea, U.K., 1981.

[5] M.J. Fritts: "Two-Dimensional Lagrangian Fluid Dynamics Using Triangular Grids"; in Finite-Difference Techniques for Vectorized Fluid Dynamics Calculations, David L. Book, ed., Springer-Verlag, 1981.

[6] B. Mercier: personal communication.

[7] M.I. Shamos and D. Hoey: in Proc. 16th Annual IEEE Symposium on Foundations of Computer Science, Berkeley, 1975.

[8] K. Stueben and U. Trottenberg: "Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications"; in Multigrid Methods, Lecture Notes in Mathematics, vol. 961, Springer-Verlag, 1982.

[9] C.S. Peskin: "A Lagrangian Method for the Navier-Stokes Equations with Large Deformations"; preprint.

[10] C.S. Peskin: "Numerical Analysis of Blood Flow in the Heart"; J. Comput. Phys. 25, 220-252, 1977.

[11] G. Voronoi: in Z. Reine Angew. Math. 134, 198, 1908.