# A Rapid Procedure for Inverting Del-Square with Certain Computers[1]

EDWARD N. LORENZ[2]

*Department of Meteorology, Massachusetts Institute of Technology, Cambridge 02139*

(Manuscript received 24 November 1975, in revised form 26 March 1976)

ABSTRACT

A rapid procedure for inverting a second-order finite-difference form of the two-dimensional del-square operator is presented. The procedure may be used whenever the precision of the computer significantly exceeds the required accuracy of the results; effectively it acquires its speed by using the otherwise unneeded power of the computer represented by the additional precision. A particular variant should prove especially useful in numerical weather prediction, in instances when storage space is at a premium.

## 1. Introduction

There are numerous problems in diverse scientific disciplines which require the inversion of the del-square operator, i.e., the solution of Poisson's equation

$$\nabla^2 p = q, \tag{1.1}$$

for $p$ in terms of $q$, subject to prescribed boundary conditions. As a consequence, numerous procedures for solving Poisson's equation have been developed, not only by mathematicians but also by workers in the fields where the solutions are to be put to use. Extensive bibliographies have been presented by Hockney (1969), Dorr (1970), Buzbee *et al.* (1970), and more recently by Bank (1975).

In meteorology the most familiar problem requiring the inversion of the two-dimensional del-square operator

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{1.2}$$

is undoubtedly the numerical solution of the barotropic vorticity equation

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial \psi}{\partial x}\frac{\partial \zeta}{\partial y} + \frac{\partial \psi}{\partial y}\frac{\partial \zeta}{\partial x} - \beta\frac{\partial \psi}{\partial x}. \tag{1.3}$$

Here $x$ and $y$ are Cartesian coordinates, $t$ is time, $\beta$ is a constant, $\zeta$ and $\psi$ are the vorticity and the stream-function, respectively, and

$$\nabla^2 \psi = \zeta. \tag{1.4}$$

Starting with an initial field of $\zeta$, one must solve (1.4) for $\psi$ before one can compute the time derivative of $\zeta$ and determine future values of $\zeta$ by stepwise integration.

Eq. (1.3) was suggested by Rossby (1939) as being applicable to weather forecasting, and in a slightly modified form, was used by Charney *et al.* (1950) to produce the first moderately successful numerical weather prediction. In the latter work the field of $\zeta$ was represented by its values at a gird of 15 by 18 points, and the corresponding values of $\psi$ were obtained by inverting a finite-difference approximation to del-square. Producing a 24-hour forecast required about 24 hours of computation on the ENIAC computer.

During the quarter century which has since elapsed, the time required to produce a 24 h forecast with a similar grid has been reduced to about 0.1 s. Although the bulk of the gain in speed has resulted from the development of faster computers, at least one order of magnitude has been gained from the discovery of more efficient algorithms, and in particular, algorithms for inverting del-square. The purpose of this paper is to present a procedure for solving (1.1) when $p$ and $q$ are represented by grid-point values, which is suitable for use with many present-day computers, and which appears to be faster than any other procedure currently in use.

Certain developments during the past decade may have served to decrease the interest among meteorologists in such a procedure. One of these is the more widespread use of the primitive equations, where the inversion of del-square is not generally required unless external gravity waves are filtered out. Another is the increased popularity of spectral methods, where the inversion of del-square becomes a trivial problem, although the remaining steps required to solve (1.3) become more cumbersome.

Nevertheless, there remain a number of problems where the use of spectral methods is at best rather awkward. Notable among these are problems where

[1] This work has been sponsored by the Office for Climate Dynamics, National Science Foundation, under Grant OCD 74-03969 A01.

[2] A portion of this work was performed when the writer was at the National Center for Atmospheric Research, Boulder, Colorado 80302. NCAR is sponsored by the National Science Foundation.

evaporation and condensation of water in the atmosphere play a significant role. Since one may not always wish to handle such problems with the primitive equations, it would seem that rapid algorithms for inverting del-square should retain considerable meteorological interest. Finally, fluid dynamicists continue to be interested in Eq. (1.3), often with $\beta=0$, for its own sake, where the motion governed by it has acquired the name "two-dimensional turbulence."

## 2. The marching-correction-marching procedure

Our procedure is based upon a process generally known as "marching." In its unmodified form the process is very simple, but the results which it yields are often worthless. We shall present a modification which can generally produce acceptable results.

Let a grid of points be formed by the intersection of a set of horizontal lines (rows) with a set of vertical lines (columns). Let the distance between two adjacent rows or columns be $\delta$. Let $q_{i,j}$ and $p_{i,j}$ denote respectively the values of $q$ and $p/\delta^2$ at the intersection of the row numbered $i$ with the column numbered $j$, and let these quantities be defined for $i=0, \ldots, M-1$ and $j=0, \ldots, N-1$. Given the values of $q_{i,j}$, we seek a solution $p_{i,j}$ of

$$(\Delta p)_{i,j}=q_{i,j}, \tag{2.1}$$

where

$$(\Delta p)_{i,j}=p_{i,j-1}+p_{i-1,j}+p_{i,j+1}+p_{i+1,j}-4p_{i,j} \tag{2.2}$$

is a second-order finite difference approximation to $\nabla^2 p$.

For $i=0$ or $M-1$, or $j=0$ or $N-1$, (2.1) contains implicitly the quantities $p_{-1,j}$, $p_{M,j}$, $p_{i,-1}$, or $p_{i,N}$. We shall describe only the case of periodic boundary conditions, where these quantities are equal respectively to $p_{M-1,j}$, $p_{0,j}$, $p_{i,N-1}$, and $p_{i,0}$. Eq. (2.1) then becomes a system of $MN$ simultaneous equations in $MN$ unknowns, whose matrix of coefficients is singular. No solution exists unless

$$\sum_{j=0}^{N-1}\sum_{i=0}^{M-1} q_{i,j}=0, \tag{2.3}$$

in which case $p_{i,j}$ is determined to within an additive constant. The procedure may be modified to apply to other boundary conditions, but considerable effort may be needed in some cases to formulate a modification with comparable efficiency. We have not attempted any such formulations.

Suppose that the values of $p_{i,j}$ are correctly known everywhere on two adjacent columns, say those where $j=J-1$, and $j=J$. We can then readily determine $p_{i,J+1}$ by setting $j=J$ in (2.1), since the remaining terms in $(\Delta p)_{i,J}$ are known, i.e., we can "march" from columns $J-1$ and $J$ to column $J+1$. Having done so, we can set $j=J+1$ in (2.1) and march to column $J+2$, and likewise ultimately to all columns. If we then use the newly determined values of $p_{i,j}$ to compute $q_{i,J-2}$ and $q_{i,J-1}$ from (2.1), the values so obtained will agree with the correct (given) values.

If instead the values of $p_{i,J-1}$ and $p_{i,J}$ are incorrectly known, perhaps by having merely been guessed at, the marching process may still be executed, but the remaining values of $p_{i,j}$ will be incorrect, as will the values of $q_{i,J-2}$ and $q_{i,J-1}$ computed from (2.1). However the differences between the correct (given) and incorrect (computed) values of $q_{i,J-2}$ and $q_{i,J-1}$ will completely determine the differences between the correct (unknown) and incorrect (assumed) values of $p_{i,J-1}$ and $p_{i,J}$, through a set of $2M$ simultaneous equations. The values of $p_{i,J-1}$ and $p_{i,J}$ may thus be corrected, and the marching process may be repeated to yield correctly the remaining values of $p_{i,j}$. Since the computation in the marching process is minimal, the speed of the whole procedure will depend largely upon the efficiency of the algorithm used to solve the $2M$ equations for the corrections to $p_{i,J-1}$ and $p_{i,J}$.

In practice, this seemingly optimal procedure possesses a serious shortcoming, namely, its instability. Even after "correction" the values of $p_{i,J-1}$ and $p_{i,J}$ will not be entirely correct; at the very least they will possess round-off errors, whose magnitudes will depend upon the precision of the particular computer. In evaluating $p_{i,J+1}$, the round-off error in $p_{i,J}$ will be multiplied by 4 and added to other round-off errors; the resulting larger error will in turn be multiplied by 4 and added to other errors in evaluating $p_{i,J+2}$. Thus, for each additional column to which one marches, the accuracy will fall by about $2\frac{1}{2}$ bits (actually by a factor which approaches $3+8^{\frac{1}{2}}=5.83$ as a limit). Eventually the error may exceed the signal.

This shortcoming is not necessarily fatal. If the computer possesses far greater precision than is actually required, and if in addition $M$ is reasonably small, the procedure may be acceptable. If, for example, the computer has 48-bit precision, and only 28-bit accuracy is required, the procedure should work for $M=10$, where one marches over 8 columns. In effect, the method provides a means of gaining speed by using the otherwise unneeded power of the computer represented by the high precision.

If instead $M=64$, a value frequently used in two-dimensional turbulence studies, the procedure as presented would be useless with any of today's standard computers. It could be made to work by writing a multiple (not simple double) precision subroutine, but this would presumably more than nullify any gain in speed which would otherwise be realized.

An obvious modification which would partially remove the shortcoming would be to march both forward and backward from columns $J-1$ and $J$, using (2.1) with $j=J-1$, then $j=J-2$, etc., for the backward marching. With the hypothetical precision and required accuracy previously assumed, the procedure should work for $M=18$, where one marches over 8 columns in each direction.

A still further modification would be to march

forward and backward not from just one pair of adjacent columns but from several pairs. Assuming again that the precision of the computer exceeds the required accuracy, one can, for any particular value of $M$, choose enough pairs to make the procedure acceptable. It is this modification which forms the basis for our procedure.

### 3. The modified marching-correction-marching procedure

The procedure which we propose is simplest when one marches forward and backward over the same number of columns, say $K-1$, from each of $L$ pairs of adjacent columns. Accordingly, we shall present the case where $N=2KL$, where $K>1$ and $L>1$ and $K$ is not too large. We could extend the procedure to arbitrary values of $N$ by marching over different numbers of columns from different pairs of columns. The speed of any computer program will depend to a considerable extent upon the efficiency of the algorithm used to perform the complex Fourier transformation which appears in the "correction" process. The procedure can therefore be especially fast when $M$ is a power of 2.

In this section we shall present the formulas which are to be executed in solving (2.1), in their order of execution. Only these formulas need appear in a computer program. In the following section we shall present additional formulas, which are needed in order to demonstrate that the procedure of this section is valid. The reader may prefer to proceed at this point to Section 4, referring back to Section 3 whenever references to formulas in Section 3 occur.

If (2.1) is to be solved many times with the same values of $K$, $L$ and $M$, as will be the case in the numerical integration of (1.3), certain quantities should be evaluated once and for all, and stored. These include the cosines and sines

$$c_m=\cos(2\pi m/M), \tag{3.1}$$

$$s_m=\sin(2\pi m/M), \tag{3.2}$$

for $m=0, \cdots, M-1$, which appear implicitly in the complex Fourier transformation, the "multipliers"

$$a_m=2-c_m, \tag{3.3}$$

the "roots"

$$\lambda_0=(K-1)/(K+1), \tag{3.4}$$

$$\lambda_m=a_m-(a_m^2-1)^{\frac{1}{2}}, \tag{3.5}$$

for $m=1, \cdots, M-1$, and the functions of $\lambda_m$ which appear in (3.11), (3.12), and (3.21)-(3.26).

The formulas which follow are to be executed each time (2.1) is solved. We begin by setting

$$p'_{i,j}=0 \tag{3.6}$$

for $i=0, \ldots, M-1$ and $j=2lK-1$ or $2lK$, with $l=0, \ldots, L$. We march forward and backward by letting

$$p'_{i,j}=-p'_{i,j-2}-p'_{i-1,j-1}-p'_{i+1,j-1}+4p'_{i,j-1}+q_{i,j-1} \tag{3.7}$$

for $i=0, \ldots, M-1$ and $j=2lK+1, 2lK+2, \ldots, 2lK+K-1$, with $l=0, \ldots, L-1$, and

$$p'_{i,j}=-p'_{i,j+2}-p'_{i-1,j+1}-p'_{i+1,j+1}+4p'_{i,j+1}+q_{i,j+1} \tag{3.8}$$

for $i=0, \ldots, M-1$ and $j=2lK-2, 2lK-3, \ldots, 2lK-K$, with $l=1, \ldots, L$. When the quantities $p'_{-1,j}$ and $p'_{M,j}$ appear on the right side of (3.7) and (3.8), they are to be set equal to $p'_{M-1,j}$ and $p'_{0,j}$ respectively. Having completed the marching, we let

$$q''_{i,j}=q_{i,j}-(p'_{i,j-1}+p'_{i-1,j}+p'_{i,j+1}+p'_{i+1,j}-4p'_{i,j}) \tag{3.9}$$

for $i=0, \ldots, M-1$ and $j=2lK+K-1$ or $2lK+K$, with $l=0, \ldots, L-1$.

We begin the correction process by performing the complex Fourier transformation

$$G_{m,l}+IH_{m,l}=\sum_{i=0}^{M-1}(q''_{i,2lK+K-1}+Iq''_{i,2lK+K})\exp(2\pi Iim/M) \tag{3.10}$$

for $m=0, \ldots, M-1$ and $l=0, \ldots, L-1$, where $G_{m,l}$ and $H_{m,l}$ are *real*. Here and subsequently we use $I$ for the imaginary unit to distinguish it from the row index $i$. We then let

$$G'_{m,l}=\tfrac{1}{2}(1+\lambda_m)(G_{m,l}+H_{M-m,l})-\tfrac{1}{2}(1-\lambda_m)(H_{m,l}-G_{M-m,l}), \tag{3.11}$$

$$H'_{m,l}=\tfrac{1}{2}(1+\lambda_m)(G_{m,l}+H_{M-m,l})+\tfrac{1}{2}(1-\lambda_m)(H_{m,l}-G_{M-m,l}), \tag{3.12}$$

for the same values of $m$ and $l$. When $G_{M,l}$ and $H_{M,l}$ appear in (3.11) and (3.12), they are to be set equal to $G_{0,l}$ and $H_{0,l}$.

To obtain quantities $E'_{m,l}$ and $F'_{m,l}$ from $G'_{m,l}$ and $H'_{m,l}$ we first set

$$E''_{0,0}=0, \tag{3.13}$$

$$F''_{0,0}=0, \tag{3.14}$$

and let

$$E''_{0,l}=G'_{0,l-1}-E''_{0,l-1}+2F''_{0,l-1}, \tag{3.15}$$

$$F''_{0,l}=H'_{0,l-1}-F''_{0,l-1}+2F''_{0,l}, \tag{3.16}$$

for $l=1, \ldots, L$. We then set

$$E'_{0,0}=0, \tag{3.17}$$

$$F'_{0,0}=\tfrac{1}{2}L^{-1}F''_{0,L}, \tag{3.18}$$

and obtain $E'_{0,l}$ and $F'_{0,l}$ by repeating (3.15) and (3.16) with double primes replaced by single primes. Likewise for $m=1, \ldots, M-1$ we set

$$E''_{m,0}=0, \tag{3.19}$$

$$F''_{m,L}=0, \tag{3.20}$$

and let

$$E''_{m,l}=-\lambda_m^K H'_{m,l-1}+\lambda_m^{2K}E''_{m,l-1} \tag{3.21}$$

for $l=1, \ldots, L$ (in that order), and

$$F''_{m,l}=-\lambda_m^K G'_{m,l}+\lambda_m^{2K}F''_{m,l+1} \tag{3.22}$$

for $l=L-1, \ldots, 0$ (in that order). We then set

$$E'_{m,0}=(1-\lambda_m^N)E''_{m,L}, \tag{3.23}$$

$$F'_{m,L}=(1-\lambda_m^N)F''_{m,0}, \tag{3.24}$$

and obtain $E'_{m,l}$ and $F'_{m,l}$ by repeating (3.21) and (3.22) with double primes replaced by single primes.

To complete the correction process we let

$$E_{m,l}=\tfrac{1}{2}(1-\lambda_m)^{-1}(E'_{m,l}+F'_{m,l}) +\tfrac{1}{2}(1+\lambda_m)^{-1}(E'_{M-m,l}-F'_{M-m,l}), \tag{3.25}$$

$$F_{M-m,l}=\tfrac{1}{2}(1-\lambda_m)^{-1}(E'_{m,l}+F'_{m,l}) -\tfrac{1}{2}(1+\lambda_m)^{-1}(E'_{M-m,l}-F'_{M-m,l}), \tag{3.26}$$

for $m=0, \ldots, M-1$ and $l=0, \ldots, L-1$, and set $F_{0,l}$ equal to $F_{m,l}$. We then perform the inverse Fourier transformation

$$p_{i,2lK-1}+Ip_{i,2lK} =M^{-1}\sum_{m=0}^{M-1}(E_{m,l}+IF_{m,l})\exp(-2\pi Iim/M) \tag{3.27}$$

for $i=0, \ldots, M-1$ and $l=0, \ldots, L-1$, where $p_{i,2lK-1}$ and $p_{i,2lK}$ are *real*, and set $p_{i,N-1}$ and $p_{i,N}$ equal to $p_{i,-1}$ and $p_{i,0}$.

Finally, we complete the solution of (2.1) by repeating the marching procedure (3.7) and (3.8) with the primes removed from the variables.

## 4. Justification of the procedure

Eqs. (3.6)-(3.9) assure us that

$$(\Delta p')_{i,j}=q_{i,j}-q''_{i,j} \tag{4.1}$$

if $j=2lK+K-1$ or $2lK+K$, and

$$(\Delta p')_{i,j}=q_{i,j} \tag{4.2}$$

otherwise. It follows that the desired solution $p_{i,j}$ may be obtained by adding to $p'_{i,j}$ a quantity $p''_{i,j}$ such that

$$(\Delta p'')_{i,j}=q''_{i,j} \tag{4.3}$$

if $j=2lK+K-1$ or $2lK+K$, and

$$(\Delta p'')_{i,j}=0 \tag{4.4}$$

otherwise. It is furthermore sufficient to determine $p''_{i,j}$ when $j=2lK-1$ or $2lK$; these values may be added to the corresponding values of $p_{i,j}$ (which vanish), after which (3.7) and (3.8), without primes, will yield the remaining values of $p_{i,j}$. It remains to establish a procedure for solving the $2LM$ equations obtained by eliminating from (4.3) and (4.4) the $(N-2L)M$ variables $p''_{i,j}$ for which $j\neq 2lK-1$ or $2lK$, and to show that this procedure may be carried out by executing (3.10)-(3.27).

For this purpose we introduce the alternative Fourier transformation

$$Q_{m,j}=\sqrt{2}\sum_{i=0}^{M-1} q''_{i,j}\cos(2\pi im/M+\pi/4), \tag{4.5}$$

$$P_{m,j}=\sqrt{2}\sum_{i=0}^{M-1} p''_{i,j}\cos(2\pi im/M+\pi/4). \tag{4.6}$$

We include the phase angle $\pi/4$ so that $Q_{m,j}$ and $Q_{M-m,j}$ will contain different information, as will $P_{m,j}$ and $P_{M-m,j}$. Eqs. (4.3) and (4.4) now become

$$P_{m,j-1}-2a_mP_{m,j}+P_{m,j+1}=Q_{m,j} \tag{4.7}$$

if $j=2lK+K-1$ or $2lK+K$, and

$$P_{m,j-1}-2a_mP_{m,j}+P_{m,j+1}=0 \tag{4.8}$$

otherwise, $a_m$ being given by (3.3). We must now establish a procedure for solving the $M$ systems, one for each value of $m$, of $2L$ equations obtained by eliminating from (4.7) and (4.8) the variables $P_{m,j}$ for which $j\neq 2lK-1$ or $2lK$.

To set up these equations, we let $c_{m,0}=0$ and $c_{m,1}=1$ for each value of $m$, and, for $k=2, \ldots, K$, let

$$c_{m,k+1}=2a_mc_{m,k}-c_{m,k-1}. \tag{4.9}$$

From repeated application of (4.8) we find that, in two-row matrix notation,

$$\begin{pmatrix}P_{m,2lK+K-2}\\P_{m,2lK+K-1}\end{pmatrix}=\begin{pmatrix}-c_{m,K-2} & c_{m,K-1}\\-c_{m,K-1} & c_{m,K}\end{pmatrix}\begin{pmatrix}P_{m,2lK-1}\\P_{m,2lK}\end{pmatrix}, \tag{4.10}$$

$$\begin{pmatrix}P_{m,2lK+K}\\P_{m,2lK+K+1}\end{pmatrix} =\begin{pmatrix}c_{m,K} & -c_{m,K-1}\\c_{m,K-1} & -c_{m,K-2}\end{pmatrix}\begin{pmatrix}P_{m,2lK+2K-1}\\P_{m,2lK+2K}\end{pmatrix}, \tag{4.11}$$

whence, from (4.7),

$$\begin{pmatrix}c_{m,K} & -c_{m,K+1}\\-c_{m,K-1} & c_{m,K}\end{pmatrix}\begin{pmatrix}P_{m,2lK-1}\\P_{m,2lK}\end{pmatrix} +\begin{pmatrix}c_{m,K} & -c_{m,K-1}\\-c_{m,K+1} & c_{m,K}\end{pmatrix}\begin{pmatrix}P_{m,2lK+2K-1}\\P_{m,2lK+2K}\end{pmatrix} =\begin{pmatrix}Q_{m,2lK+K-1}\\Q_{m,2lK+K}\end{pmatrix}. \tag{4.12}$$

Each system of $2L$ equations is represented by (4.12), with one value of $m$, and with $l=0, \ldots, L-1$. The quantities $P_{m,2LK-1}$ and $P_{m,2LK}$ are equal to $P_{m,-1}$ and $P_{m,0}$, since $2LK=N$.

In principle we can solve (4.12) by another marching-correction-marching procedure. With assumed values for $P_{m,-1}$ and $P_{m,0}$, we could solve for $P_{m,2K-1}$ and $P_{m,2K}$, then $P_{m,4K-1}$ and $P_{m,4K}$, etc., and then obtain

corrections to $P_{m,-1}$ and $P_{m,0}$ and solve again for $P_{m,2K-1}$ and $P_{m,2K}$ etc. However, this procedure proves to be unstable (except when $m=0$) and hence unsuitable, because $c_{m,2K-1}$, $c_{m,2K}$ and $c_{m,2K+1}$ are large quantities.

We can use a stable marching-correction-marching procedure after transforming the variables. Comparing (4.5) and (3.10) we see that

$$G_{m,l}+H_{M-m,l}=Q_{m,2lK+K-1}+Q_{m,2lK+K}, \quad (4.13)$$

$$H_{m,l}-G_{M-m,l}=-Q_{m,2lK+K-1}+Q_{m,2lK+K}, \quad (4.14)$$

whence it follows from (3.11) and (3.12) that

$$\begin{pmatrix} G'_{m,l} \\ H'_{m,l} \end{pmatrix}=\begin{pmatrix} 1 & \lambda_m \\ \lambda_m & 1 \end{pmatrix}\begin{pmatrix} Q_{m,2lK+K-1} \\ Q_{m,2lK+K} \end{pmatrix}. \quad (4.15)$$

Likewise, from (4.6) and the inverse of (3.27) it follows that

$$E_{m,l}+F_{M-m,l}=P_{m,2lK-1}+P_{m,2lK}, \quad (4.16)$$

$$F_{m,l}-E_{M-m,l}=-P_{m,2lK-1}+P_{m,2lK}, \quad (4.17)$$

whence it follows from (3.25) and (3.26) that

$$\begin{pmatrix} E'_{m,l} \\ F'_{m,l} \end{pmatrix}=\begin{pmatrix} 1 & -\lambda_m \\ -\lambda_m & 1 \end{pmatrix}\begin{pmatrix} P_{m,2lK-1} \\ P_{m,2lK} \end{pmatrix}. \quad (4.18)$$

It is in fact because of (4.15) and (4.18) that we chose to introduce the quantities $G'_{m,l}$, $H'_{m,l}$, $E'_{m,l}$ and $F'_{m,l}$.

We now observe that except when $m=0$, $\lambda_m$ is the smaller root of the equation

$$\lambda_m{}^2-2a_m\lambda_m+1=0, \quad (4.19)$$

the larger root being $\lambda_m{}^{-1}$. It therefore follows by induction that

$$c_{m,k}=(\lambda_m{}^{-k}-\lambda_m{}^k)/(\lambda_m{}^{-1}-\lambda_m). \quad (4.20)$$

From (4.20) it follows that

$$\begin{pmatrix} 1 & \lambda_m \\ \lambda_m & 1 \end{pmatrix}\begin{pmatrix} c_{m,K} & -c_{m,K+1} \\ -c_{m,K-1} & c_{m,K} \end{pmatrix} =\begin{pmatrix} 0 & -\lambda_m{}^{-K} \\ \lambda_m{}^K & 0 \end{pmatrix}\begin{pmatrix} 1 & -\lambda_m \\ -\lambda_m & 1 \end{pmatrix}, \quad (4.21)$$

$$\begin{pmatrix} 1 & \lambda_m \\ \lambda_m & 1 \end{pmatrix}\begin{pmatrix} c_{m,K} & -c_{m,K-1} \\ -c_{m,K+1} & c_{m,K} \end{pmatrix} =\begin{pmatrix} 0 & \lambda_m{}^K \\ -\lambda_m{}^{-K} & 0 \end{pmatrix}\begin{pmatrix} 1 & -\lambda_m \\ -\lambda_m & 1 \end{pmatrix}. \quad (4.22)$$

In view of (4.15), (4.18), (4.21) and (4.22), Eq. (4.12) reduces to

$$\begin{pmatrix} 0 & -\lambda_m{}^{-K} \\ \lambda_m{}^K & 0 \end{pmatrix}\begin{pmatrix} E'_{m,l} \\ F'_{m,l} \end{pmatrix} +\begin{pmatrix} 0 & \lambda_m{}^K \\ -\lambda_m{}^{-K} & 0 \end{pmatrix}\begin{pmatrix} E'_{m,l+1} \\ F'_{m,l+1} \end{pmatrix}=\begin{pmatrix} G'_{m,l} \\ H'_{m,l} \end{pmatrix}. \quad (4.23)$$

The $2L$ scalar equations represented by (4.23), with one value of $m$ and with $l=0, \ldots, L-1$, now separate into two systems of $L$ equations. Each of these may be solved by a *stable* marching-correction-marching procedure, by assuming a value for $E'_{m,0}$ and marching forward in $l$, and assuming a value for $F'_{m,L}$ and marching backward in $l$. Stability is assured by multiplying only by positive powers of $\lambda_m$, which are small. The explicit procedure is given by (3.19)–(3.24).

When $m=0$, the equations represented by (4.12) possess a singular matrix, reflecting the fact that (2.1) determines $p_{i,j}$ only to within an additive constant. Since $a_0=1$, the roots of (4.19) become equal, and the procedure used when $m\neq 0$ is unfeasible.

Instead we choose to define $\lambda_0$ by (3.4). Eqs. (4.15) and (4.18) are still valid, and it is readily verified that $c_{0,k}=k$, whence it follows that

$$\begin{pmatrix} 1 & \lambda_0 \\ \lambda_0 & 1 \end{pmatrix}\begin{pmatrix} c_{0,K} & -c_{0,K+1} \\ -c_{0,K-1} & c_{0,K} \end{pmatrix} =\begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & -\lambda_0 \\ -\lambda_0 & 1 \end{pmatrix}, \quad (4.24)$$

$$\begin{pmatrix} 1 & \lambda_0 \\ \lambda_0 & 1 \end{pmatrix}\begin{pmatrix} c_{0,K} & -c_{0,K-1} \\ -c_{0,K+1} & c_{0,K} \end{pmatrix} =\begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix}\begin{pmatrix} 1 & -\lambda_0 \\ -\lambda_0 & 1 \end{pmatrix}, \quad (4.25)$$

whereupon (4.12) reduces to

$$\begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} E'_{0,l} \\ F'_{0,l} \end{pmatrix}+\begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix}\begin{pmatrix} E'_{0,l+1} \\ F'_{0,l+1} \end{pmatrix}=\begin{pmatrix} G'_{0,l} \\ H'_{0,l} \end{pmatrix}. \quad (4.26)$$

The $2L$ scalar equations represented by (4.26) may be solved by a stable marching-correction-marching procedure, using forward marching only. The explicit procedure is given by (3.13)–(3.18).

## 5. Concluding remarks

The Fourier transformation (4.5) and the subsequent formula (4.15) provide straightforward definitions of $G'_{m,l}$ and $H'_{m,l}$. The Fourier transformation (3.10) and the formulas (3.11) and (3.12) define precisely the same quantities, but the definitions seem less natural, since there is no logical reason for regarding $q''_{i,2lK+K-1}$ and $q''_{i,2lK+K}$ as real and imaginary parts of the same physical quantity. We nevertheless present (3.10)–(3.12), together with the inverse transformations (3.25)–(3.27), in the recommended procedure, because the rapid algorithms available for complex Fourier transformations may then be used.

In a computer program based upon (3.1)–(3.27), further savings may be realized by eliminating some of the indicated multiplications. For example, the

factor $M^{-1}$ in (3.27) could be moved to (3.25) and (3.26) and included as a factor in the precomputed functions of $\lambda_m$, thus effectively redefining $E_{m,l}$ and $F_{m,l}$. Likewise the factor $\lambda_m^K$ in (3.21) and (3.22) could be moved to (3.11) and (3.12) by redefining $G'_{m,l}$ and $H'_{m,l}$. Finally, execution of (3.7) and (3.8) may, for the first indicated values of $j$ only, be replaced respectively by equating $p_{i,2lK+1}$ to $q_{i,2lK}$ and $p'_{i,2lK-2}$ to $q_{i,2lK-1}$, thus eliminating the indicated multiplications of 0 by 4, and the additions of some zeros.

We have written a FORTRAN program for the solution of Poisson's equation, based on (3.1)–(3.27), and containing the improvements just noted. We have applied it to a number of cases where the given values of $q_{i,j}$ were produced by a random-number generator. For $M=64$ and $N=64$, with $K=8$ and $L=4$, the time required to invert del-square, exclusive of the precomputation of the stored quantities, is 0.0195 seconds on the CDC 7600 computer at NCAR. The words in the computer possess 48 bits (about 14 decimal places), and the results are accurate to 8 decimal places. We have not applied the procedure to any cases where $M$ is not a power of 2.

Among other direct procedures which have been used to solve Poisson's equation, the one which ours resembles most closely appears to be one of those presented by Bank (1975). This procedure also uses a marching process, and differs from ours mainly in the details of the "correction" process.

Our procedure is also related to one recently described by Sweet (1974), even though the steps actually executed are rather different. In each procedure the original system of $MN$ equations is effectively replaced at an intermediate stage by a system of $MN/K$ equations, in which the unknowns are the values of $p_{i,j}$ on a set of $N/K$ columns. However, in Sweet's procedure these columns are equally spaced, while in ours they occur as $N/2K$ pairs of adjacent columns, the pairs being equally spaced. The use of pairs of adjacent columns permits the use of the marching process.

Our procedure is easily converted into one for solving the equation

$$(\Delta p)_{i,j}-c^2p_{i,j}=q_{i,j} \tag{5.1}$$

which appears in numerous physical problems, and in particular in numerical weather prediction when certain multi-layer models are used. We replace the constant 2 in (3.3) by $2+c^2/2$ and the coefficients 4 in (3.7)–(3.9) by $4+c^2$, and we use (3.5) in place of (3.4), and (3.19)–(3.24) in place of (3.13)–(3.18), even when $m=0$.

Finally, we note that our procedure may be particularly useful when the inversion of del-square is being performed for the purpose of solving the barotropic vorticity equation, and when $M$ and $N$ are large enough to render storage space of some concern. We first march forward from the first pair of columns and backward from the second pair, and obtain values of $q''_{i,K-1}$ and $q''_{i,K}$. We then discard the values of $p'_{i,j}$ obtained during the process before we march forward from the second pair of columns and backward from the third pair to obtain $q''_{i,3K-1}$ and $q''_{i,3K}$, etc.

Later, during the second execution of the marching process, we first determine the values of $p_{i,j}$ on and between the first and second pairs of columns. We immediately evaluate and store the corresponding nonlinear terms in (1.3), representing the advection of vorticity, and then discard the values of $p_{i,j}$ used in the evaluations before we determine the values of $p_{i,j}$ on and between the second and third pairs of columns, etc.

We note that in any event $MN$ locations [or preferably $(M+2)$ $(N+2)$, to make efficient use of cyclic continuity] are required to store the vorticity field. With a minimum-storage time-differencing scheme, such as the $N$-cycle scheme proposed by the writer (Lorenz, 1971), a similar number of locations will suffice to store the time derivatives of vorticity. It follows that in addition to these approximately $2MN$ locations, which would be needed in any case, only $2M(K+1)$ locations are needed to store $p'_{i,j}$, or, subsequently, $p_{i,j}$, while $2M(L+1)$ locations, plus a small amount of working space, will suffice for $q'_{i,2lK+K-1}$ and $q'_{i,2lK+K}$, or, subsequently, $G_{m,l}$ and $H_{m,l}$, $E_{m,l}$ and $F_{m,l}$, and finally $p_{i,2K-1}$ and $p_{i,2K}$. The procedure, modified for solving (5.1), should likewise use a minimum of storage space when a multi-layer model is used for numerical weather prediction.

*Acknowledgment.* The writer is grateful to Dr. Roland Sweet of NCAR for the opportunity to engage in a number of productive discussions concerning numerical solutions of Poisson's equation.

## REFERENCES

Bank, R. E., 1975: Marching algorithms for elliptic boundary value problems. Ph.D. thesis, Harvard University.

Buzbee, B. L., G. H. Golub and C. W. Neilson, 1970: On direct methods for solving Poisson's equation. *SIAM J. Num. Anal.* 7, 627–656.

Charney, J. G., R. Fjørtoft and J. von Neumann, 1950: Numerical integration of the barotropic vorticity equation. *Tellus*, 2, 237–254.

Dorr, F. W., 1970: The direct solution of the discrete Poisson equation on a rectangle. *SIAM Rev.*, 12, 248–263.

Hockney, R. W., 1969: The potential calculation and some applications. *Methods in Computational Physics*, Vol. 9, Academic Press, 131–211.

Lorenz, E. N., 1971: An $N$-cycle scheme for stepwise numerical integration. *Mon. Wea. Rev.*, 99, 644–648.

Rossby, C. G., 1939: Relation between variations in the intensity of the zonal circulation of the atmosphere and the displacements of the semipermanent centers of action. *J. Marine Res.*, 2, 38–55.

Sweet, R. A., 1974: A generalized cyclic reduction algorithm. *SIAM J. Num. Anal.*, 11, 506–520.