University of Glasgow

DEPARTMENT OF

AEROSPACE ENGINEERING

Generalized Differential Quadrature and Its Application

Chang SHU and Bryan E. RICHARDS

G. U. Aero. Report 9117

Generalized Differential Quadrature and Its Application

Chang SHU and Bryan E. RICHARDS

G. U. Aero. Report 9117

November, 1991

LIST OF NOTATIONS

t	time
x_i	coordinates in the horizontal direction
y_i	coordinates in the longitudinal direction
N	number of grid points in the x direction
M	number of grid points in the y direction
$f(x)$	a smooth function
$\underline{f}(x)$	approximated function to f(x)
$L_N(x)$	an *N*th order Legendre polynomial
$M(x)$	*N*th order polynomial, $M(x) = (x-x_1)\cdot(x-x_2)\cdots(x-x_N)$
$N(x, x_k)$	*(N-1)*th order polynomial, $M(x) = N(x, x_k)\cdot(x - x_k)$
$M^{(m)}(x)$	*m*th order derivative of function M(x)
a_{ij}	weighting coefficients of the first order derivative
b_{ij}	weighting coefficients of the second order derivative
$w_{ik}^{(n)}$	weighting coefficients of the *n*th order derivative in the x direction
$\overline{w}_{jk}^{(m)}$	weighting coefficients of the *m*th order derivative in the y direction
V_N	an N-dimensional linear polynomial vector space, $V_N(x)$
V_M	a M-dimensional linear polynomial vector space, $V_M(y)$
$V_{N\times M}$	a N×M dimensional linear polynomial vector space, $V_{N\times M}(x,y)$
$r_i(x)$	base polynomials in V_N
$s_j(y)$	base polynomials in V_M
$\Phi_{ij}(x,y)$	base polynomials in $V_{N\times M}$
δ_{ij}	the Kronecker operator
a_{ij}^x	weighting coefficients of the first order derivative in the x direction
a_{ij}^y	weighting coefficients of the first order derivative in the y direction
f_z	the first order derivative of function f with respect to z
$E(f)$	the approximation error of function f
$E_D^{(m)}(f)$	the approximation error of the *m*th order derivative of function f
$L(u)$	a differential operator, which contains spatial derivatives

A	a matrix from the GDQ spatial discretization
λ	an eigenvalue of the matrix A
u	horizontal velocity component
v	vertical velocity component
ω	vorticity
ψ	stream function
Re	Reynolds number
$\nabla 2$	the Laplacian operator
ω_{ij}, ψ_{ij}, u_{ij}, v_{ij}	values of ω, ψ, u, v at x_i, y_j

ABSTRACT

The technique of differential quadrature (DQ) for the solution of a partial differential equation is extended and generalized in this paper. The general formulation for determining the weighting coefficients of the first order derivative is obtained. A recurrence relationship for determining the weighting coefficients of the second and higher order partial derivatives is also obtained, and it is shown that generalized differential quadrature (GDQ) can be considered as a finite difference scheme of the highest order. Three typical formulas of weighting coefficients for the first order derivative are also given in the paper. The error estimations for the function and derivative approximation, and the eigenvalue structures of some basic GDQ spatial discretization matrices have been studied. The application of GDQ to model problems showed that accurate results can be obtained using a small number of grid points.

1. INTRODUCTION

The numerical techniques for the solution of a partial differential equation can be classified into two categories. One is based on the direct discretization of the derivatives and integrals. Another is based on the variational principles or the principles of weighted residuals. The conventional finite difference methods lie in the first category while the finite element and the spectral methods are in the second. Usually, low order methods such as finite differences and finite elements can provide accurate results by using a large number of grid points. However, in some practical applications the numerical solution of a governing equation is required at only a few specified points in a domain. But for acceptable accuracy, conventional finite difference and finite element methods also require the use of a large number of grid points to obtain the solution at those specified points. In seeking a more efficient method using just a few grid points to get an accurate result, Bellman et al [1] introduced a method of differential quadrature, where a partial derivative of a function with respect to a coordinate direction is expressed as a linear weighted sum of all the functional variables at all mesh points along that direction. It is clear that this method is based on the direct discretization of the derivative, and therefore, is in the first category indicated above. Preliminary computational results ([1]-[6]) showed that differential quadrature has potential as an attractive approximation technique. The key technique to differential quadrature is the

means to determine the weighting coefficients for the discretization of any order partial derivative. Bellman et al suggested two methods to determine the weighting coefficients of the first order derivative. One method solves a set of algebraic equations which is obtained by satisfying the linear constrained relation for test functions of x^k, k = 0, 1, ···, N-1, where N is the total number of grid points in a domain. This equation system has a unique solution because the matrix elements are composed of a Vandermonde matrix. Unfortunately, when N is large the inversion of this matrix becomes difficult. This is probably one of the reasons that applications of this scheme so far only use a number of grid points less than or equal to 13. The second method computes the weighting coefficients by an algebraic formulation with coordinates of grid points chosen as the roots of an *N*th order shifted Legendre polynomial. This means that if N is specified, the distributions of grid points are the same for different physical problems. This can provide a major drawback and restrict the application of differential quadrature. In order to overcome this drawback, the generalized differential quadrature technique was developed. The development is discussed in this paper. It is based on the analysis of high order polynomial approximation in the overall domain.

2. DIFFERENTIAL QUADRATURE

For the one dimensional unsteady problem, Bellman et al [1] assume a function u(x,t) to be sufficiently smooth to allow the following linear constrained relation to be satisfied

$$u_x(x_i,t) = \sum_{j=1}^{N} a_{ij} \cdot u(x_j,t) \qquad (2.1)$$

for i = 1, 2, ···, N,

where $u_x(x_i,t)$ indicates the first order derivative of u(x,t) with respect to x at x_i . Substituting (2.1) into a time-dependent partial differential equation yields a set of ordinary differential equations which can be integrated by such well-developed schemes as Runge-Kutta multi-step integration.

The key technique to this procedure is how to determine the weighting coefficients a_{ij}. Bellman et al suggested two ways to carry this out. The first way is to let (2.1) be exact for test functions

$g(x)=x^k$, k=0, 1, ···, N-1, which leads to a set of linear algebraic equations

$$\sum_{j=1}^{N} a_{ij} \cdot x_j^k = k \cdot x_i^{k-1} \qquad (2.2)$$

for i =1, 2, ···, N; k = 0, 1, ···, N-1.

This equation system has a unique solution since its matrix is of Vandermonde form. Unfortunately, when N is large, this matrix is ill-conditioned and its inversion is difficult.

Another way is similar to the first one with an exception that the different test functions

$$g(x) = \frac{L_N(x)}{(x - x_k) \cdot L_N^{(1)}(x_k)} \quad , \quad k = 1, 2, \cdots, N \qquad (2.3)$$

are chosen, where $L_N(x)$ is the Nth order Legendre polynomial and $L_N^{(1)}(x)$ the first order derivative of $L_N(x)$. By choosing x_k to be the roots of the shifted Legendre polynomial, Bellman et al obtained a simple algebraic formulation for a_{ij}

$$a_{ij} = \frac{L_N^{(1)}(x_i)}{(x_i - x_j) \cdot L_N^{(1)}(x_j)} \quad , \quad j \neq i \qquad (2.4a)$$

$$a_{ii} = \frac{1 - 2x_i}{2x_i \cdot (x_i - 1)} \qquad (2.4b)$$

Formulation (2.4) is only valid for coordinates of grid points chosen as the roots of an Nth order Legendre polynomial.

3. GENERALIZED DIFFERENTIAL QUADRATURE

In order to overcome the drawback described above for differential quadrature and to obtain a similar simple formulation for a_{ij}, a method of generalized differential quadrature has been introduced, based on the analysis of the polynomial linear vector space.

3.1 High Order Polynomial Approximation in the Overall Domain

Since any finite range can be transformed into the range of [0, 1] by a simple transformation, we will consider only the range [0, 1] hereafter. It is well known that a continuous function f(x) in the interval [0, 1] can be approximated by an infinite polynomial accurately in

accordance with the Weierstrass polynomial approximation theorem. In practice, a truncated finite polynomial may be used. Some methods, an example being the spectral method, have successfully applied the concept of high order polynomial approximation to the solution of partial differential equations. Following this approach, it is supposed that any smooth function in the interval [0, 1] can be approximated by a *(N-1)*th order polynomial.

It is easy to show that the polynomial of degree less than or equal to N-1 constitutes an N-dimensional linear vector space V_N with respect to the operation of addition and multiplication. From the concept of linear independence, the bases of a linear vector space can be considered as a linearly independent subset which spans the entire space. Here if $r_k(x)$, k=1, 2, ···, N, which are in the space V_N, are the base polynomials, any polynomial in V_N can be expressed as a linear combination of $r_k(x)$, k=1, 2, ···, N, i.e

$$\underline{f}(x) = P_N f = \sum_{k=1}^{N} c_k \cdot r_k(x) \tag{3.1}$$

where P_N is a projection operator of smooth functions onto V_N, c_k is a coefficient, and $\underline{f}(x)$, $r_k(x)$ are in space V_N. The spectral method uses a high order polynomial similar to (3.1) to approximate the function f(x) in the overall domain. But the procedures for the solution of a partial differential equation are quite different. The spectral method, which is based on the principle of the weighted residuals, involves the determination of the coefficients of the base polynomials, namely c_k, while generalized differential quadrature (to be described), which uses this formulation only to determine the weighting coefficients for discretization of any order (less than N) partial derivative, involves the determination of the functional values at grid points.

3.2 Weighting Coefficients of the First Order Derivative

Equation (2.1) is a linear constrained relationship. If the base polynomials $r_k(x)$, k=1, 2, ···, N, satisfy (2.1), so does polynomial $\underline{f}(x)$. And if the base polynomial $r_k(x)$ is chosen to be x^{k-1}, the same equation system as (2.2), given by Bellman's first method, can be obtained. For generality, here the base polynomial $r_k(x)$ is chosen to be the Lagrange interpolation polynomial

$$r_k(x) = \frac{M(x)}{(x - x_k) \cdot M^{(1)}(x_k)} \tag{3.2}$$

where $M(x) = (x-x_1)\cdot(x-x_2)\cdots(x-x_N)$

$$M^{(1)}(x_k) = \prod_{j=1, j\neq k}^{N}(x_k - x_j)$$

$x_1, x_2, \cdots, x_N$ are the coordinates of grid points, and can be chosen arbitrarily.

For simplicity, we set

$$M(x) = N(x, x_k)\cdot(x - x_k) \quad , \quad k = 1, 2, \cdots, N$$

with $N(x_i, x_j) = M^{(1)}(x_i)\cdot\delta_{ij}$, where δ_{ij} is the Kronecker operator.

Thus we have

$$M^{(m)}(x) = N^{(m)}(x, x_k)\cdot(x - x_k) + m\cdot N^{(m-1)}(x, x_k) \qquad (3.3)$$

for m = 1, 2, ···, N-1; k = 1, 2, ···, N

where $M^{(m)}(x)$, $N^{(m)}(x, x_k)$ indicate the mth order derivative of M(x) and $N(x, x_k)$. Substituting (3.2) into (2.1) and using (3.3), we obtain

$$a_{ij} = \frac{M^{(1)}(x_i)}{(x_i - x_j)\cdot M^{(1)}(x_j)} \quad , \quad for\ j \neq i \qquad (3.4a)$$

$$a_{ii} = \frac{M^{(2)}(x_i)}{2M^{(1)}(x_i)} \qquad (3.4b)$$

for i, j =1, 2, ···, N.

Equation (3.4) is a simple formulation for computing a_{ij} without any restriction on choice of grid point x_i. Actually, if x_i is given, it is easy to compute $M^{(1)}(x_i)$, thus a_{ij} for $i \neq j$. The calculation of a_{ii} is based on the computation of the second order derivative $M^{(2)}(x_i)$ which is not easy to be obtained. Next, it will be shown that a_{ii} can be calculated from a_{ij} ($i \neq j$).

According to the theory of a linear vector space, one set of base polynomials can be expressed uniquely by another set of base polynomials. Thus if one set of base polynomials satisfies a linear constrained relationship, say (2.1), so does another set of base polynomials. And since the weighting coefficients are only dependent on the coordinates of grid points if the number of grid points is given, the equation system for determination of a_{ij} derived from one set of base polynomials should be equivalent to that derived from other sets of base polynomials. Thus a_{ij} satisfies the following equation which is obtained by the base polynomial x^k when k=0

$$\sum_{j=1}^{N} a_{ij} = 0 \qquad (3.5)$$

where a_{ii} can be easily determined from a_{ij} ($i \neq j$). Equation (3.4) is a general form for calculating a_{ij}. It follows that if the coordinates of grid points are chosen as the roots of a shifted Legendre polynomial, (3.4) is exactly the same as that given by Bellman's second method.

3.3 Weighting Coefficients of the Second and Higher Order Derivatives

For discretization of the second order derivative, we introduce the following linear constrained relation

$$u_{xx}(x_i, t) = \sum_{j=1}^{N} b_{ij} \cdot u(x_j, t) \qquad (3.6)$$

for i = 1, 2, ···, N

where $u_{xx}(x, t)$ is the second order derivative of u(x, t) with respect to x, and Lagrange interpolated polynomials are chosen as the base polynomials (see 3.2). Using the same approach as for the first order derivative and formulation (3.4), the weighting coefficients b_{ij} are given by

$$b_{ij} = 2 a_{ij} \cdot (a_{ii} - \frac{1}{x_i - x_j}) \quad , \quad for\ j \neq i \qquad (3.7a)$$

$$b_{ii} = \frac{M^{(3)}(x_i)}{3M^{(1)}(x_i)} \qquad (3.7b)$$

for i, j = 1, 2, ···, N.

When $j \neq i$, b_{ij} can be calculated from a_{ij} easily. In a similar analysis to the case of the first order derivative, the equation system for b_{ij} derived from the above Lagrange interpolated polynomials is equivalent to that derived from the base polynomials x^k, k = 0, 1, ···, N-1. Thus b_{ij} should also satisfy the following formulation derived from the base polynomial x^k when k=0

$$\sum_{j=1}^{N} b_{ij} = 0 \qquad (3.8)$$

from which b_{ii} can be easily determined.

Furthermore, for the case of discretization of the higher order derivative, the linear constrained relations are applied as follows

$$u_x^{(m-1)}(x_i,t) = \sum_{j=1}^{N} w_{ij}^{(m-1)} \cdot u(x_j,t) \tag{3.9}$$

$$u_x^{(m)}(x_i,t) = \sum_{j=1}^{N} w_{ij}^{(m)} \cdot u(x_j,t) \tag{3.10}$$

for i = 1, 2, ···, N

where $u_x^{(m-1)}(x_i,t)$, $u_x^{(m)}(x_i,t)$ indicate the *(m-1)*th and *m*th order derivative of u(x,t) with respect to x at x_i, $w_{ij}^{(m-1)}$, $w_{ij}^{(m)}$ are the weighting coefficients related to $u_x^{(m-1)}(x_i,t)$ and $u_x^{(m)}(x_i,t)$. Substituting (3.2) into (3.9), (3.10) and using (3.3), (3.4), a recurrence formulation is obtained as follows

$$w_{ij}^{(m)} = m \cdot (a_{ij} \cdot w_{ii}^{(m-1)} - \frac{w_{ij}^{(m-1)}}{x_i - x_j}) \quad , \qquad j \neq i \tag{3.11}$$

for m = 2, 3, ···, N-1; i, j = 1, 2, ···, N

where a_{ij} are the weighting coefficients of the first order derivative described above. Again, in terms of the analysis of the N-dimensional linear vector space, the equation system for $w_{ij}^{(m)}$ derived from Lagrange interpolated polynomials should be equivalent to that derived from the base polynomials x^k, k=0, 1, ···, N-1. Thus $w_{ij}^{(m)}$ should satisfy the following equation obtained from the base polynomial x^k when k=0

$$\sum_{j=1}^{N} w_{ij}^{(m)} = 0 \tag{3.12}$$

From this formulation, $w_{ii}^{(m)}$ can be easily calculated from $w_{ij}^{(m)}$ ($j \neq i$).

3.4 Extension to the Multi-Dimensional Case

For the two-dimensional approximation of a function f(x,y) in the domain $x \in [0,1]$, $y \in [0,1]$, it is supposed that the value of f(x,b), where b is a constant, $b \in [0, 1]$, can be approximated by an *(N-1)*th order polynomial $P_N(x)$ which constitutes an N-dimensional linear vector space V_N with N base polynomials $r_i(x)$, i=1, 2, ···, N, and the value of f(a,y), where a is a constant, $a \in [0, 1]$, can be approximated by a *(M-1)*th order polynomial $P_M(y)$ which constitutes a M-dimensional linear vector space V_M with M base polynomials $s_j(y)$, j=1, 2, ···, M. The value of function f(x,y) can be approximated by the polynomial $Q_{N\times M}(x,y)$ with the

form

$$Q_{N\times M}(x,y)=\sum_{i=1}^{N}\sum_{j=1}^{M}\bar{c}_{ij}\cdot x^{i-1}\cdot y^{j-1} \tag{3.13}$$

where $\bar{c}_{ij}$ *is a coefficient*

It is clear that $Q_{N\times M}(x,y)$ constitutes a $N\times M$ dimensional linear polynomial vector space $V_{N\times M}$ with respect to the operation of addition and scalar multiplication. It will now be shown that $\Phi_{ij}(x,y) = r_i(x)\cdot s_j(y)$ constitutes the base polynomials in the vector space $V_{N\times M}$. Since $r_i(x)$, $s_j(y)$ are the base polynomials of V_N and V_M, they must be linearly independent, that is

$$\sum_{i=1}^{N} c_i\cdot r_i(x)=0 \quad \textit{only if } c_i=0,\ i=1,\ 2,\ \cdots,\ N \tag{3.14}$$

$$\sum_{j=1}^{M} d_j\cdot s_j(y)=0 \quad \textit{only if } d_j=0,\ j=1,\ 2,\ \cdots,\ M \tag{3.15}$$

Now we see that if

$$\sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}\cdot\Phi_{ij}(x,y)=0 \quad , \quad i.e. \quad \sum_{i=1}^{N}[\sum_{j=1}^{M} c_{ij}\cdot s_j(y)]\cdot r_i(x)=0$$

From (3.14) the following equation is obtained

$$\sum_{j=1}^{M} c_{ij}\cdot s_j(y)=0$$

Finally from (3.15) we obtain $c_{ij} = 0$. Then, $\Phi_{ij}(x, y)$ constitutes the base polynomials in $V_{N\times M}$.

Now it is assumed that the following constrained relations are satisfied for function u(x,y,t) and its first order spatial derivatives

$$u_x(x_i,y_j,t)=\sum_{k=1}^{N} a_{ik}^{x}\cdot u(x_k,y_j,t) \tag{3.16}$$

$$u_y(x_i,y_j,t)=\sum_{k=1}^{M} a_{jk}^{y}\cdot u(x_i,y_k,t) \tag{3.17}$$

for i = 1, 2, ···, N; j = 1, 2, ···, M

where a^x_{ik} , a^y_{jk} are the weighting coefficients related to $u_x(x_i,y_j,t)$ and $u_y(x_i,y_j,t)$ respectively. If all the base polynomials $\Phi_{ij}(x,y)$ satisfy equations (3.16), (3.17), then so does any polynomial in $V_{N\times M}$. Substituting $\Phi_{ij}(x, y)$ into (3.16), (3.17) leads to

$$\sum_{k=1}^{N} a_{ik}^{x}\cdot r_j(x_k)=r_j^{(1)}(x_i) \tag{3.18}$$

$$\sum_{k=1}^{M} a_{ik}^{y} \cdot s_j(y_k) = s_j^{(1)}(y_i) \tag{3.19}$$

where $r_j^{(1)}(x_i)$ represents the first order derivative of $r_j(x)$ at x_i and $s_j^{(1)}(y_i)$ represents the first order derivative of $s_j(y)$ at y_i. From (3.18), (3.19), it is obvious that a^x_{ik} or a^y_{jk} is only related to $r_i(x)$ or $s_j(y)$. Hence the formulation of the one dimensional case can be directly extended to the two dimensional case, that is

$$a_{ij}^{x} = \frac{M^{(1)}(x_i)}{(x_i - x_j) \cdot M^{(1)}(x_j)} \quad , \quad for\, j \neq i \tag{3.20a}$$

$$a_{ii}^{x} = - \sum_{j=1, j\neq i}^{N} a_{ij}^{x} \tag{3.20b}$$

for i, j = 1, 2, ···, N

$$a_{ij}^{y} = \frac{P^{(1)}(y_i)}{(y_i - y_j) \cdot P^{(1)}(y_j)} \quad , \quad for\, j \neq i \tag{3.21a}$$

$$a_{ii}^{y} = - \sum_{j=1, j\neq i}^{M} a_{ij}^{y} \tag{3.21b}$$

for i, j = 1, 2, ···, M

where

$$M^{(1)}(x_i) = \prod_{j=1, j\neq i}^{N} (x_i - x_j)$$

$$P^{(1)}(y_i) = \prod_{j=1, j\neq i}^{M} (y_i - y_j)$$

Similarly, for the second or higher order derivative the recurrence relationship of the weighting coefficients can be obtained as follows

$$w_{ij}^{(n)} = n \cdot (a_{ij}^{x} \cdot w_{ii}^{(n-1)} - \frac{w_{ij}^{(n-1)}}{x_i - x_j}) \quad , \quad j \neq i \tag{3.22a}$$

$$w_{ii}^{(n)} = - \sum_{j=1, j\neq i}^{N} w_{ij}^{(n)} \tag{3.22b}$$

for n = 2, 3, ···, N-1; i, j = 1, 2, ···, N

$$\overline{w}_{ij}^{(m)} = m \cdot (a_{ij}^{y} \cdot \overline{w}_{ii}^{(m-1)} - \frac{\overline{w}_{ij}^{(m-1)}}{y_i - y_j}) \quad , \quad j \neq i \tag{3.23a}$$

$$\overline{w}_{ii}^{(m)} = - \sum_{j=1, j\neq i}^{M} \overline{w}_{ij}^{(m)} \tag{3.23b}$$

for m = 2, 3, ···, M-1; i, j = 1, 2, ···, M

where $w_{ij}^{(n)}$ are the weighting coefficients of the *n*th order derivative of u(x,y,t) with respect to x at x_i, y_j, namely $u_x^{(n)}(x_i,y_j,t)$, and $\overline{w}_{ij}^{(m)}$ the weighting coefficients of the *m*th order derivative of u(x,y,t) with respect to y at x_i, y_j, namely $u_y^{(m)}(x_i,y_j,t)$. They satisfy

$$u_x^{(n)}(x_i,y_j,t) = \sum_{k=1}^{N} w_{ik}^{(n)} \cdot u(x_k,y_j,t) \tag{3.24}$$

$$u_y^{(m)}(x_i,y_j,t) = \sum_{k=1}^{M} \overline{w}_{jk}^{(m)} \cdot u(x_i,y_k,t) \tag{3.25}$$

for i = 1, 2, ···, N; j = 1, 2, ···, M; n = 1, 2, ···, N-1; m = 1, 2, ···, M-1.

Similar formulations can be obtained for the three dimensional case.

If the functional values at all grid points are obtained, it is easy to determine the functional values in the overall domain in terms of the polynomial approximation, i.e.

$$u(x,y_j,t) = \sum_{i=1}^{N} u(x_i,y_j,t) \cdot r_i(x) \quad , \quad j = 1, 2, \cdots, M \tag{3.26a}$$

$$u(x_i,y,t) = \sum_{j=1}^{M} u(x_i,y_j,t) \cdot s_j(y) \quad , \quad i = 1, 2, \cdots, N \tag{3.26b}$$

$$u(x,y,t) = \sum_{i=1}^{N}\sum_{j=1}^{M} u(x_i,y_j,t) \cdot r_i(x) \cdot s_j(y) \tag{3.26c}$$

where $r_i(x)$, $s_j(y)$ are the Lagrange interpolated polynomials along the x and y direction respectively.

3.5 Comparison with the Highest Order Finite Difference Scheme

For the one dimensional case, supposing the whole domain has N grid points, x_1, x_2, ···, x_N. The *(N-1)*th order finite difference scheme for the first order derivative can be written as a linear sum of the functional values at N grid points, which has the same form as (2.1) where the weighting coefficients are determined by the Taylor series expansion which is usually used in the design of the low order finite difference schemes. Using a Taylor series expansion, $u(x_j,t)$ can be expressed as

$$u(x_j,t) = u(x_i,t) + u^{(1)}(x_i,t)\cdot(x_j-x_i) + \cdots + u^{(k)}(x_i,t)\cdot(x_j-x_i)^k / k! + \cdots + u^{(N-1)}(x_i,t)\cdot(x_j-x_i)^{N-1} / (N-1)! + R_N \tag{3.27}$$

where $u^{(k)}(x_i,t)$ is the *k*th order derivative of u with respect to x at x_i, R_N is the truncated error,

and can be written as

$$R_N = u^{(N)}(\xi_j, t)\cdot(x_j - x_i)^N / N! \quad , \quad \xi_j \in [x_i, x_j] \tag{3.28}$$

Substituting (3.27) into (2.1) yields

$$u_x(x_i,t) = \sum_{j=1}^{N} a_{ij}\cdot\{u(x_i,t) + u^{(1)}(x_i,t)\cdot(x_j - x_i) + \cdots + u^{(N-1)}(x_i,t)\cdot(x_j - x_i)^{N-1}/(N-1)! + R_N\} \tag{3.29}$$

In order to keep the right side of (3.29) consistent with the left side of (3.29) with *(N-1)*th order accuracy, we set

$$\begin{cases} \sum_{j=1}^{N} a_{ij} = 0 \\ \sum_{j=1}^{N} a_{ij}\cdot(x_j - x_i) = 1 \\ \sum_{j=1}^{N} a_{ij}\cdot(x_j - x_i)^k = 0, \; k = 2, 3, \cdots, N-1 \end{cases} \tag{3.30}$$

for i = 1, 2, ···, N.

Equation set (3.30) is another equation system for the determination of the weighting coefficients a_{ij} which are derived from the Taylor series expansion.

As stated above, the equation system for the determination of a_{ij} derived from one set of base polynomials is equivalent to that derived from another set of base polynomials. We will choose only one equation system (2.2) obtained by the base polynomials x^k, k = 0, 1, ···, N-1 and prove that this equation system is equivalent to (3.30) given by the highest order finite difference approach.

It is obvious that the first equation of (2.2) and (3.30) are the same, i.e.

$$\sum_{j=1}^{N} a_{ij} = 0 \tag{3.31}$$

Furthermore, it can be shown that the second equation of the two systems are the same, i.e.

$$\sum_{j=1}^{N} a_{ij}\cdot(x_j - x_i) - 1 = \sum_{j=1}^{N} a_{ij}\cdot x_j - (\sum_{j=1}^{N} a_{ij})\cdot x_i - 1 = \sum_{j=1}^{N} a_{ij}\cdot x_j - 1 = 0 \tag{3.32}$$

Now, assuming that the first (p+1) equations of the two systems are the same, that is

$$\sum_{j=1}^{N} a_{ij}\cdot(x_j - x_i)^k = \sum_{j=1}^{N} a_{ij}\cdot x_j^k - k\cdot x_i^{k-1} = 0 \tag{3.33}$$

for k =0, 1, ···, p; i = 1, 2, ···, N

then using the binary formulation

$$(a-b)^p = a^p - c_p^1 \cdot a^{p-1} \cdot b + \cdots + (-1)^k \cdot c_p^k \cdot a^{p-k} \cdot b^k + \cdots + (-1)^p \cdot b^p \tag{3.34}$$

here c_p^k is the combination of p terms taken k at a time,

and setting a = b = 1, the following expression will be obtained.

$$c_p^1 - c_p^2 + \cdots + (-1)^{k+1} \cdot c_p^k + \cdots + (-1)^{p+1} = 1 \tag{3.35}$$

Using (3.34), the *(p+2)*th equation of (3.30) can be written as

$$\sum_{j=1}^{N} a_{ij} \cdot (x_j - x_i)^{p+1} = \sum_{j=1}^{N} a_{ij} \cdot x_j^{p+1} - c_{p+1}^1 \cdot x_i \cdot \left[\sum_{j=1}^{N} a_{ij} \cdot (x_j^p - \frac{1}{2} \cdot c_p^1 \cdot x_j^{p-1} \cdot x_i + \cdots + \frac{(-1)^p \cdot x_i^p}{p+1}) \right] \tag{3.36}$$

Substituting (3.35), (3.33) into (3.36) leads to

$$\begin{aligned} \sum_{j=1}^{N} a_{ij} \cdot (x_j - x_i)^{p+1} &= \sum_{j=1}^{N} a_{ij} \cdot x_j^{p+1} - (p+1) \cdot x_i^p \cdot \left[c_p^1 - c_p^2 + \cdots + (-1)^{k+1} \cdot c_p^k + \cdots + (-1)^{p+1} \right] \\ &= \sum_{j=1}^{N} a_{ij} \cdot x_j^{p+1} - (p+1) \cdot x_i^p \end{aligned} \tag{3.37}$$

Equation set (3.37) demonstrates that the *(p+2)*th equation of the two systems are the same. Since p is an arbitrary integer only if p $\leqslant$ N-2, it has been proved that the two systems (3.30) and (2.2) are exactly the same.

Similarly, for the case of higher order derivatives, it is easy to show that the weighting coefficients $w_{ij}^{(m)}$ satisfy the following equation system, derived from the *(N-m)*th order finite difference scheme for the *m*th order derivative in the overall domain

$$\begin{cases} \sum_{j=1}^{N} w_{ij}^{(m)} = 0 \\ \sum_{j=1}^{N} w_{ij}^{(m)} \cdot (x_j - x_i)^m = m! \\ \sum_{j=1}^{N} w_{ij}^{(m)} \cdot (x_j - x_i)^k = 0 \quad , \quad k = 1, 2, \cdots, N-1, \text{ but } k \neq m \end{cases} \tag{3.38}$$

It is clear that the first equation of (3.38) is exactly the same as (3.12) for m = 2, ···, N-1. To prove that $w_{ij}^{(m)}$, for 2 $\leqslant$ m $\leqslant$ N-1, satisfies other equations of (3.38), it is supposed that $w_{ij}^{(m-1)}$ satisfies those equations firstly, that is

$$\sum_{j=1}^{N} w_{ij}^{(m-1)} \cdot (x_j - x_i)^k = \begin{cases} (m-1)! & when\ k = m-1 \\ 0 & others \end{cases} \tag{3.39}$$

Using (3.11), now we have, for $1 \leqslant k \leqslant N-1$

$$\sum_{j=1}^{N} w_{ij}^{(m)} \cdot (x_j - x_i)^k = m \cdot w_{ii}^{(m-1)} \cdot \sum_{j=1}^{N} a_{ij} \cdot (x_j - x_i)^k + m \cdot \sum_{j=1}^{N} w_{ij}^{(m-1)} \cdot (x_j - x_i)^{k-1} \tag{3.40}$$

Substituting (3.39), (3.30), (3.12) into (3.40) leads to

$$\sum_{j=1}^{N} w_{ij}^{(m)} \cdot (x_j - x_i)^k = \begin{cases} m! & when\ k = m \\ 0 & others \end{cases} \tag{3.41}$$

Since m is an arbitrary integer only if $2 \leqslant m \leqslant N-1$, it has been proved that $w_{ij}^{(m)}$ satisfies (3.38) exactly. Thus it can be concluded that GDQ is an extension of finite difference methods, and is a highest order finite difference scheme.

As an example, we will show that the discretization of the first order derivative by the GDQ approach in the domain $[x_{i-1}, x_{i+1}]$ is the same as that given from the second order finite difference scheme. Clearly, the domain $[x_{i-1}, x_{i+1}]$ includes three grid points x_{i-1}, x_i, x_{i+1}, and it is known that any smooth function in this domain can be approximated by a polynomial of degree 2, which constitutes a 3-dimensional linear vector space. Thus the weighting coefficients of the first order derivative for this specific case can be determined as follows according to formulation (3.4)

$$M(x) = (x-x_{i-1})\cdot(x-x_i)\cdot(x-x_{i+1}) \tag{3.42}$$

$$M^{(1)}(x_{i-1}) = (x_{i-1}-x_i)\cdot(x_{i-1}-x_{i+1}) \tag{3.43}$$

$$M^{(1)}(x_i) = (x_i-x_{i-1})\cdot(x_i-x_{i+1}) \tag{3.44}$$

$$M^{(1)}(x_{i+1}) = (x_{i+1}-x_{i-1})\cdot(x_{i+1}-x_i) \tag{3.45}$$

and

$$a_{i,i-1} = -\triangle_2 /[(\triangle_1+\triangle_2)\triangle_1] \tag{3.46}$$

$$a_{i,i+1} = \triangle_1 /[(\triangle_1+\triangle_2)\triangle_2] \tag{3.47}$$

$$a_{i,i} = (\triangle_2-\triangle_1)/(\triangle_1\cdot\triangle_2) \tag{3.48}$$

where

$$\triangle_1 = x_i - x_{i-1}$$

$\triangle_2 = x_{i+1} - x_i$

Hence, the first order derivative of a function f can be approximated as

$$f_x(x_i) = \sum_{j=-1}^{1} a_{i,i+j} \cdot f(x_{i+j}) \tag{3.49}$$

It is easy to show that (3.49) is exactly the same as that from the second order finite difference scheme and if the grid is uniform, (3.49) can be reduced to

$$f_x(x_i) = 0.5 \cdot [f(x_{i+1}) - f(x_{i-1})]/\triangle \tag{3.50}$$

where $\triangle = \triangle_1 = \triangle_2$

which is the same as used in the finite difference scheme. In the same manner, the discretization of the first order derivative at x_{i-1} and x_{i+1} can be written as

$$f_x(x_{i-1}) = -\frac{2\Delta_1 + \Delta_2}{(\Delta_1 + \Delta_2)\cdot\Delta_2} \cdot f(x_{i-1}) + \frac{\Delta_1 + \Delta_2}{\Delta_1 \cdot \Delta_2} \cdot f(x_i) - \frac{\Delta_1}{(\Delta_1 + \Delta_2)\cdot\Delta_2} \cdot f(x_{i+1}) \tag{3.51}$$

$$f_x(x_{i+1}) = \frac{\Delta_2}{(\Delta_1 + \Delta_2)\cdot\Delta_1} \cdot f(x_{i-1}) - \frac{\Delta_1 + \Delta_2}{\Delta_1 \cdot \Delta_2} \cdot f(x_i) + \frac{2\Delta_2 + \Delta_1}{(\Delta_1 + \Delta_2)\cdot\Delta_1} \cdot f(x_{i+1}) \tag{3.52}$$

which are exactly the same as those from the second order finite difference scheme. For the overall domain case, it is suggested that such a domain can be divided into N-1 elements with grid points, $x_1, \cdots, x_N$. At location x_i, i = 2, 3, ···, N-1, the first order derivative of a function can be discretized by (3.49) in the element $[x_{i-1}, x_{i+1}]$. It is noted that in the case here, the two neighbouring elements $[x_{i-1}, x_{i+1}]$ and $[x_i, x_{i+2}]$, used for the discretization of the first order derivative at collocation points x_i and x_{i+1}, are overlapped with the region of $[x_i, x_{i+1}]$. This behaviour is different from the standard finite element approach where the neighbouring elements are patched. Similarly, at x_1 and x_N, the discretization of the first order derivative of a function can be obtained by (3.51) in the element $[x_1, x_3]$ and by (3.52) in the element $[x_{N-2}, x_N]$. It can be concluded that any higher order finite difference scheme can be designed using this technique in a straightforward way.

3.6 Specific Results for Typical Distributions of Grid Points

In this section, three specific formulations of the weighting coefficients will be given for three typical distributions of grid points: uniform grid; the coordinates chosen as the roots of $T_N(\eta)$ or $|T_N(\eta)| - 1$, where $T_N(\eta)$ is an *N*th order Chebyshev polynomial. Since the complete

weighting coefficients of the second and higher order derivatives can be calculated from those of the first order derivative, and that for the multi-dimensional cases, each direction can be treated as in the **1D** case, then only the weighting coefficients of the first order derivative in the **1D** case are considered.

Case I: Uniform Grid

By a uniform grid it is meant that the grid has the same step sizes. Thus setting

$\triangle x = x_2 - x_1 = x_i - x_{i-1} = x_N - x_{N-1}$, etc.,

one can obtain

$x_j - x_i = (j-i)\cdot\triangle x$

$M^{(1)}(x_i) = (-1)^{N-i}\cdot(\triangle x)^{N-1}\cdot(i-1)!\cdot(N-i)!$, i = 1, 2, ···, N

Thus

$$a_{ij} = (-1)^{i+j}\cdot\frac{(i-1)!\cdot(N-i)!}{\Delta x\cdot(i-j)\cdot(j-1)!\cdot(N-j)!} \qquad (3.53a)$$

for i, j = 1, 2, ···, N, except $j \neq i$

$$a_{ii} = -\sum_{j=1, j\neq i}^{N} a_{ij} \quad , \quad i = 1, 2, \cdots, N \qquad (3.53b)$$

Case II: Coordinates Chosen As the Roots of $|T_N(\eta)|$ - 1

An *N*th order Chebyshev polynomial can be written as

$T_N(\eta) = \cos(N\theta)$ (3.54)

with $\eta = \cos\theta$, $-1 \leqslant \eta \leqslant 1$

Setting $|T_N(\eta)| = 1$ yields

$N\theta = j\pi$, j = 0, 1, ···, N

i.e. $\eta_j = \cos(j\pi/N)$, j = 0, 1, ···, N

where η_j is the coordinate of the grid point in the domain [1, -1]. In this case, the Lagrange interpolated polynomial can be written as

$$r_j(\eta) = \frac{(-1)^{j+1}\cdot(1-\eta^2)\cdot T_N^{(1)}(\eta)}{\bar{c}_j\cdot N^2\cdot(\eta-\eta_j)} \quad , j = 0,\ 1,\ \cdots,\ N \tag{3.55}$$

where $T_N^{(1)}(\eta)$ is the first order derivative of $T_N(\eta)$, and

$$\bar{c}_j = \begin{cases} 2 & when\ j = 0,\ N \\ 1 & others \end{cases}$$

Thus (3.4) can be reduced to

$$a_{ij} = \frac{(-1)^{j+i}\cdot\bar{c}_i}{\bar{c}_j\cdot(\eta_i-\eta_j)} \quad ,\quad i, j = 0,\ 1,\ \cdots,\ N,\ but\ j \neq i \tag{3.56a}$$

$$a_{ii} = -\sum_{j=1, j\neq i}^{N} a_{ij} \quad ,\quad i = 0,\ 1,\ \cdots,\ N \tag{3.56b}$$

It can be seen that (3.56a) is the same as that deduced from the pseudospectral Chebyshev method [8]. To analyse this behaviour, it is known that both spectral methods and finite element methods are based on the principle of the weighted residuals. Spectral methods can be considered as an extension of finite element methods. The difference is that the spectral methods include only one element while finite element methods include many elements. As shown in section 3.5, finite difference methods can also be considered as "finite element" methods which are different from the standard approach in that the elements in a finite difference method are overlapped while the elements in a standard finite element method are patched. But if the whole computational domain is composed of only one element, both finite difference methods, and finite element methods in which the weighting function is taken as the delta function, should give the same results. This is because in this case, one overlapped element and one patched element are the same. From this analysis, it is shown that the GDQ approach should give the same results as the spectral collocation methods if the same distribution of grid points is used, since they can be considered as an extension of the finite difference and finite element methods with only one element. This phenomenon is confirmed in the research which is reported above.

If the physical domain is not [1,-1], but [a,b],then we need to use the following transformation

$x = 0.5\cdot(b-a)\cdot(1-\eta) + a$, where x is the physical coordinate

The weighting coefficients $\bar{a}_{ij}$ in the physical coordinate system can be written as

$$\bar{a}_{ij} = -2 \cdot a_{ij} / (b-a) \quad , \quad i, j = 0, 1, \cdots, N. \tag{3.57}$$

Case III: Coordinates Chosen As the Roots of $T_N(\eta)$

Setting $T_N(\eta) = 0$ yields

$N\theta = 0.5\cdot(2j-1)\pi$, i.e. $\eta_j = \cos[0.5\cdot(2j-1)\pi/N]$, $j = 1, 2, \cdots, N$

It should be noted that η_j is in the domain $[\eta_1, \eta_N]$, where $\eta_1 = \cos(0.5\pi/N)$, $\eta_N = -\eta_1$, and $\eta_1 \neq 1$. In this case, the Lagrange interpolated polynomial can be written as

$$r_j(\eta) = \frac{(-1)^{j+1} \cdot (1-\eta_j^2)^{1/2} \cdot T_N(\eta)}{N \cdot (\eta - \eta_j)} \quad , \quad j = 1, 2, \cdots, N. \tag{3.58}$$

Then (3.4) can be reduced to

$$a_{ij} = \frac{(-1)^{j+i} \cdot (1-\eta_j^2)^{1/2}}{(\eta_i - \eta_j) \cdot (1-\eta_i^2)^{1/2}} \quad , \quad i, j = 1, \cdots, N, \; but \; j \neq i \tag{3.59a}$$

$$a_{ii} = -\sum_{j=1, j\neq i}^{N} a_{ij} \quad , \quad i = 1, 2, \cdots, N \tag{3.59b}$$

Similarly, if the physical domain is [a, b], using the following transformation

$\eta = d_2\cdot(x-a)/(b-a) + d_1$, where $d_1 = \cos[0.5\pi/N]$, $d_2 = -2d_1$,

the weighting coefficients $\bar{a}_{ij}$ in the physical coordinate system can be written as

$$\bar{a}_{ij} = d_2 \cdot a_{ij} / (b-a) = -2\cos[0.5 \cdot \pi / N] \cdot a_{ij} / (b-a) \tag{3.60}$$

for i, j = 1, 2, ···, N.

4. ERROR ESTIMATIONS

The theory and details of GDQ have been described in the previous sections. The errors of the approximations for a function and its derivatives will be estimated in this section.

4.1 The Function Approximation

Firstly, we will discuss the approximation error when f(x) is approximated by an *(N-1)*th

order polynomial, particularly by the Lagrange interpolation polynomial

$$P_N f = \sum_{i=1}^{N} f(x_i) \cdot r_i(x) \tag{4.1}$$

We define the approximation error of f(x) as

$$E(f) = f(x) - P_N f . \tag{4.2}$$

If it is supposed that the Nth order derivative of function f(x) is a constant, say K, then using a Taylor expansion, we can obtain

$$\begin{aligned} f(x) &= f(c) + f^{(1)}(c) \cdot (x-c) + \cdots + f^{(k)}(c) \cdot (x-c)^k / k! + \cdots \\ &\quad + f^{(N-1)}(c) \cdot (x-c)^{N-1} / (N-1)! + f^{(N)}(\xi) \cdot (x-c)^N / N! \\ &= m_0 + m_1 x + m_2 x^2 + \cdots + m_{N-1} \cdot x^{N-1} + K \cdot x^N / N! \end{aligned} \tag{4.3}$$

where c is a constant, and $\xi \in [x, c]$. Since (4.1) is exactly satisfied for a polynomial of degree less than or equal to N-1, we have

$$E(x^k) = 0, \quad \text{when } k = 0, 1, \cdots, N-1. \tag{4.4}$$

Substituting (4.3) into (4.2) and using (4.4), we obtain

$$E(f) = K \cdot E(x^N)/N! \tag{4.5}$$

where

$$E(x^N) = x^N - \sum_{i=1}^{N} x_i^N \cdot r_i(x) \tag{4.6}$$

On the other hand, substituting the *(N-1)*th order polynomial $g(x) = x^N - (x-x_1)\cdot(x-x_2)\cdots(x-x_N)$ $= x^N - M(x)$ into (4.1), we obtain

$$\sum_{i=1}^{N} x_i^N \cdot r_i(x) = x^N - M(x) \tag{4.7}$$

Finally, we get

$$E(f) = K \cdot M(x)/N! \tag{4.8}$$

In most cases, the Nth order derivative of f(x) is not a constant, but may be bounded. In this case, we can turn to another way to analyse E(f). For simplicity, we set $\phi(x) = P_N f$, and define the function F(z) as

$$F(z) = f(z) - \phi(z) - a \cdot M(z) \tag{4.9}$$

Clearly, when $z = x_1, x_2, \cdots, x_N$, $F(z) = 0$. If we set $F(x) = 0$, we get

$$E(f) = f(x) - P_N f = f(x) - \phi(x) = a \cdot M(x) \tag{4.10}$$

Since F(z) has N+1 roots in the domain, by repeated application of Rolle's theorem, the *N*th order derivative of F(z), $F^{(N)}(z)$, is found to have at least one root lying between x_1 and x_N. Let ξ denote this point. We have

$$F^{(N)}(\xi) = 0 . \tag{4.11}$$

From (4.9), we obtain

$$a = f^{(N)}(\xi)/N! , \tag{4.12}$$

$$\text{so, } E(f) = f^{(N)}(\xi)\cdot M(x)/N! . \tag{4.13}$$

Generally, ξ is a function of x.

4.2 The Derivative Approximation

We define the error for the *m*th order derivative approximation as

$$E_D^{(m)}(f) = \frac{\partial^m f}{\partial x^m} - \frac{\partial^m (P_N f)}{\partial x^m} = \frac{\partial^m f}{\partial x^m} - \frac{\partial^m \phi}{\partial x^m} \tag{4.14}$$

where m = 1, 2, ···, N-1. Generally, $E_D^{(m)}(f)$ can be written as

$$E_D^{(m)}(f) = \frac{1}{N!}\cdot\frac{\partial^m [f^{(N)}(\xi)\cdot M(x)]}{\partial x^m} \tag{4.15}$$

Since ξ is an unknown function of x, it is difficult to estimate $E_D^{(m)}(f)$ using (4.15). As a special case, if we assume that the *N*th order derivative of f(x) is a constant, namely K, then from (4.8), we get

$$E_D^{(m)}(f) = K\cdot M^{(m)}(x)/N! \tag{4.16}$$

Although (4.16) is satisfied for the condition of $f^{(N)}(\xi) = K$, it is useful in the error analysis. Firstly, (4.16) has no restriction on x. In other words, x can be any coordinate in the domain. Secondly, similar to the analysis of the order of the truncated error in a low order finite difference scheme, when the order of the truncated error caused by GDQ is studied, we can only consider the *(N+1)*th term in the Taylor series expansion though this term is not the exact error. The *(N+1)*th term of the Taylor series expansion is $f^{(N)}(c)\cdot(x-c)^N/N!$, where c is a constant. So, $f^{(N)}(c)$ can be treated as a constant in this case. Thus the analysis of the function and the derivative approximations is the same as that shown above. For a more general case,

we can use a similar method as in the analysis of the function approximation to do it. Since $g(z) = f(z) - \phi(z)$ has N roots in the domain, according to Rolle's theorem, its *m*th order derivative $g^{(m)}(z)$ has at least (N-m) roots in the domain, namely, $\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_{N-m}$. Thus the function

$$F^{(m)}(z) = g^{(m)}(z) - \underline{a}\cdot\underline{M}(z) = f^{(m)}(z) - \phi^{(m)}(z) - \underline{a}\cdot\underline{M}(z) \qquad (4.17)$$

where

$$\underline{M}(z) = (z-\underline{x}_1)\cdot(z-\underline{x}_2)\cdots(z-\underline{x}_{N-m}) ,$$

vanishes at $\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_{N-m}$. Now, if we set $F^{(m)}(\underline{x}) = 0$, where $\underline{x}$ is different from $\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_{N-m}$, then $F^{(m)}(z)$ has (N-m+1) roots, and

$$E_D^{(m)}[f(\underline{x})] = f^{(m)}(\underline{x}) - \phi^{(m)}(\underline{x}) = \underline{a}\cdot\underline{M}(\underline{x}) \qquad (4.18)$$

Using Rolle's theorem repeatedly (N-m) times, the *(N-m)*th order derivative of $F^{(m)}(z)$ is found to have at least one root ξ, thus

$$\underline{a} = f^{(N)}(\xi)/(N-m)!$$

$$E_D^{(m)}[f(\underline{x})] = f^{(N)}(\xi)\cdot\underline{M}(\underline{x})/(N-m)! \qquad (4.19)$$

Equation (4.19) is satisfied for $\underline{x} \neq \underline{x}_1, \underline{x}_2, \cdots, \underline{x}_{N-m}$. This is guaranteed if $\underline{x}$ is outside the domain of $\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_{N-m}$.

If it is assumed that all the x_i and x are in the interval h, and the *N*th order derivative of function f is bounded, then

$|f^{(N)}(\xi)| \leqslant C$, where C is a positive constant

$|M^{(m)}(x)| \leqslant N\cdot(N-1)\cdots(N-m+1)\cdot h^{N-m}$

$|\underline{M}(\underline{x})| \leqslant h^{N-m}$

so

$$\left|E_D^{(m)}(f)\right| \leq \frac{C\cdot h^{N-m}}{(N-m)!} \qquad (4.20)$$

for $1 \leqslant m \leqslant N-1$

5. EIGENVALUES OF SPECIFIC GDQ SPATIAL DISCRETIZATION MATRICES

Time-dependent problems are usually well-posed by the equation

$$\frac{\partial u}{\partial t} = L(u) \tag{5.1}$$

with proper initial and boundary conditions, where L is an operator which contains the spatial part of the partial differential equations. L is generally a non-linear operator. After discretization by GDQ and linearization of the non-linear terms, (5.1) can be transformed into a set of ordinary differential equations in time

$$\frac{d\mathbf{U}}{dt} = \mathbf{AU} + \mathbf{Q} \tag{5.2}$$

where $\mathbf{U}$ is the vector of the functional values at interior grid points, $\mathbf{Q}$ contains non-homogeneous and boundary values and A is a matrix. According to the stability analysis to (5.2), it is defined that a method is called A-stable if the region of absolute stability includes the region $\mathrm{Re}\{\lambda\triangle t\} \leqslant 0$ (whole left half plane), where λ is an eigenvalue of A. Furthermore, a method is said to be stable for a particular problem if, for sufficiently small $\triangle t > 0$, the product of $\triangle t$ times every eigenvalue of A lies within the region of absolute stability, usually with $\mathrm{Re}\{\lambda\triangle t\} \leqslant 0$ (part region of left half plane). Thus it is very useful to analyse the eigenvalue structure of the GDQ spatial discretization matrix when its stability is studied. In the following, we will study the eigenvalues of three basic GDQ spatial discretization matrices.

5.1 The Convection Operator

Here L(u) is chosen as a convection operator

$$L(u) = -\frac{\partial u}{\partial x} \quad , \qquad \text{on } [0,\ 1] \tag{5.3}$$

with Dirichlet Boundary condition

$$u(0) = f(x) \tag{5.4}$$

Firstly, we consider the three typical distributions of grid points given in section 3.6 to study the influence of the grid points

Case I : basic grid is generated by $|T_N(\eta)| = 1$;

Case II : basic grid is generated by $T_N(\eta) = 0$;

Case III : uniform grid.

The eigenvalues with grid of case I are plotted in Fig. 1. Fig. 2 and Fig. 3 show the eigenvalues with grid of case II and case III respectively. It is clear, from Fig. 1, that the real parts of all the eigenvalues of case I are strictly negative. This is not true for cases II and III. In fact, the real part of the maximum eigenvalue for cases II and III is positive although the modulus of the maximum eigenvalue of these two cases is less than that of case I. It is noted that, for cases II and III, the maximum eigenvalue does always lie in the unstable region. This behaviour is independent of the number of grid points used. Thus it seems to be true that the distribution of the grid points can greatly influence the stability behaviour of a global method such as GDQ. It is also found that the minimum step size near the boundary, for cases II and III, is larger than that for case I.

Here, we pose a question: is the above observation likely to be a major reason to cause stability problems arising from the use of cases II and III? To study this, we introduce a transformation

$$\underline{x} = (1 - \alpha)\cdot(3\cdot x^2 - 2\cdot x^3) + \alpha\cdot x \ , \ \alpha \geqslant 0 \qquad (5.5)$$

where $\underline{x}$ is the transformed coordinate. When $\alpha \leqslant 1$, the transformed grid is stretched near the boundary (i.e. grid points are more concentrated near the boundary) and when $\alpha > 1$, the transformed grid is relaxed near the boundary. Using (5.5), we can get

Case IV : Transformed from Case II with $\alpha < 1$;

Case V : Transformed from Case III with $\alpha < 1$.

To investigate the effect of the minimum step size, under the condition of stability, on the value of the modulus of the maximum eigenvalue, we introduce

Case VI : Transformed from Case I with $\alpha > 0$.

Fig. 4 - 6 display the eigenvalues of cases IV, V and VI. In these cases, the real parts of all the eigenvalues are strictly negative. It is shown in Fig. 6 that when the minimum step size is relaxed near the boundary, the value of the maximum eigenvalue is reduced, thus the time step size is relaxed. It can be concluded from Fig. 4 - 5 that the stretched grid near the

boundary can improve the stability. From here, a question may be posed: what is the behaviour if the grid is stretched at other points? To study it, we introduce another transformation, which stretches the grid near the middle point

$$\underline{x} = \xi = 2\cdot(1 - \beta)\cdot(x^2 - x) + \beta\cdot x \ , \quad \beta \geqslant 0 \ , x \leqslant 0.5 \tag{5.6a}$$

$$\underline{x} = 1 - \xi \ , x > 0.5 \tag{5.6b}$$

Using (5.6), we obtain case VII as

Case VII : Transformed from Case III with $\beta < 1$.

Fig. 7 shows the eigenvalues of case VII. Obviously, the grid stretched near the middle point does not improve the stability behaviour. Actually, the real part of the maximum eigenvalue for all the cases is always positive. When N becomes a large number, the structure of the eigenvalues tends to be symmetrical about the origin.

5.2 The Diffusion Operator

The diffusion operator is chosen as

$$L(u) = \frac{\partial^2 u}{\partial x^2} \ , \quad \text{on } [0,\ 1] \tag{5.7}$$

The boundary condition we will impose is of Dirichlet type

$$u(0) = u(1) = 0 \tag{5.8}$$

or of Neumann type

$$\frac{\partial u}{\partial x}(0) = \frac{\partial u}{\partial x}(1) = 0 \tag{5.9}$$

When the grid of case I was used, the eigenvalues for both the Dirichlet and Neumann boundary conditions are real negative numbers. But the Neumann boundary condition can give smaller eigenvalues than the Dirichlet boundary condition, thus the former may allow a larger time step size to be used. For example, the value of the maximum eigenvalue of N = 31, i.e. -1.5443×10^5, for the Dirichlet condition, can be reduced to -4.6665×10^4 for the Neumann condition. This is also the case when the grid of case II was used in which the eigenvalues for the Dirichlet condition are real numbers but are not for the Neumann condition. When N = 31, the maximum eigenvalue is $(-5.5389\times10^4,\ 0)$ for the Dirichlet

condition, and (-1.6801×10^4, 8.0125×10^3) for the Neumann condition. Although the Neumann condition can give smaller eigenvalues, it may cause stability problems. When the grid of case III was used with the Dirichlet condition, the real part of all the eigenvalues are negative, but when the Neumann condition was applied, the real part of the maximum eigenvalue is positive which can cause the computation to be unstable. Fig.8 shows the eigenvalues with grid of case III for the Dirichlet and Neumann conditions.

5.3 The Convection-Diffusion Operator

We now consider the convection-diffusion operator

$$L(u) = \upsilon \cdot \frac{\partial^2 u}{\partial x^2} - \frac{\partial u}{\partial x} \quad , \quad \text{on } [0,\ 1] \tag{5.10}$$

with a Dirichlet type boundary condition. When ν = O(1), this equation is dominated by convection and diffusion. When ν « O(1), this equation is mainly dominated by convection. It is found that when ν = O(1), the real parts of all the eigenvalues for all of the cases studied are strictly negative, and the case III gives the smallest value of maximum eigenvalue which then allows the use of the largest time step size, but when ν is very small, the real parts of some eigenvalues may be positive, leading to a stability problem. It is found that the minimum ν for keeping stability is greatly affected by the distribution of grid points. For example, to obtain a stable solution, the minimum value of ν is around 0.05 for the grid of case III, and 0.0015 for the grid of cases I and II when N=21. Thus for the case of very small ν, the uniform grid is not recommended. The instability problem can be removed by increasing the number of grid points for all the cases when ν is very small. If the number of grid points is kept the same, it is useful to explore the behaviour if the grid is stretched or relaxed near the boundary. We have found that, for the grid of cases II and III, the real parts of all the eigenvalues can be negative if the grid is stretched near the boundary. This is not true for the grid of case I. Fig. 9 - 10 show the eigenvalues of the convection-diffusion operator with ν = 0.001 for the grid of case I. It is clear from these figures that, when N = 21, the real part of the maximum eigenvalue is positive, but when N = 31, the real parts of all the eigenvalues are strictly negative. Keeping N = 21, when the grid is

stretched near the boundary, then more eigenvalues lie in the right half plane, but when the grid is relaxed near the boundary, the real parts of all the eigenvalues are kept in the left half plane. It may be concluded that for the convection-diffusion operator, the stability can be improved by stretching the grid near the boundary for some cases of grid, and by relaxing the grid near the boundary for other cases of grid.

6. APPLICATION TO MODEL PROBLEMS

In order to validate the capability of GDQ for solving partial differential equations, two test examples, each of which has an analytic solution or has been well studied by other numerical methods, have been chosen for simulation.

6.1 Solutions of the 2D Burger's Equation

We consider the two-dimensional steady problem by solving the 2D Burger equation

$$u_t + c\cdot u_x + d\cdot u_y = \varepsilon\cdot(u_{xx} + u_{yy}) \tag{6.1}$$

with boundary conditions for $t > 0$

$u(x,0,t)=\{1-\exp[(x-1)\cdot c/\varepsilon]\}/[1-\exp(-c/\varepsilon)]$, $u(x,1,t)=0$, $0 \leqslant x \leqslant 1$

$u(0,y,t)=\{1-\exp[(y-1)\cdot d/\varepsilon]\}/[1-\exp(-d/\varepsilon)]$, $u(1,y,t)=0$, $0 \leqslant y \leqslant 1$.

The exact solution to (6.1) is

$$u(x,y) = \frac{1-\exp[(x-1)\cdot c/\varepsilon]}{1-\exp(-c/\varepsilon)} \cdot \frac{1-\exp[(y-1)\cdot d/\varepsilon]}{1-\exp(-d/\varepsilon)} \tag{6.2}$$

Using GDQ, we have employed the grid of cases I, II and III to simulate this problem, and found that, when N = 11, the allowable maximum time step size is 1.1×10^{-3} for case I, 3.2×10^{-3} for case II and 7.10×10^{-3} for case III, and that the converged results for all three cases are nearly the same. This confirms the findings from the stability and eigenvalue studies in the above section. Table I lists the computational results using the grid of case III with N = 11. Some exact results and numerical results given by a time-split MacCormack scheme are also included in this table. For the finite difference simulation, the allowable

Table I The Steady Solution of 2D Burger's Equation

$c = 1.0, \quad d = 2.0, \quad \varepsilon = 0.5$

y	0.20	0.40	0.60	0.80
x	Computed by GDQ (N = M = 11, CPU = 0.44 sec.)			
0.20	0.901911	0.854935	0.750394	0.517749
0.40	0.789693	0.748554	0.657015	0.453316
0.60	0.622288	0.589865	0.517726	0.357209
0.80	0.372555	0.353141	0.309950	0.213851
x	Computed by FD[1] (N = M = 51, CPU = 17.15 sec.)			
0.20	0.901928	0.854973	0.750462	0.517836
0.40	0.789720	0.748616	0.657117	0.453437
0.60	0.622318	0.589932	0.517833	0.357328
0.80	0.372578	0.353191	0.310026	0.213933
x	Exact			
0.20	0.901916	0.854945	0.750410	0.517764
0.40	0.789702	0.748575	0.657046	0.453345
0.60	0.622299	0.589890	0.517764	0.357244
0.80	0.372563	0.353160	0.309979	0.213878

maximum time step size was used. The CPU time required on the IBM 3090 is also shown in the table. It is clear that the GDQ results are more accurate than the finite difference results even though fewer grid points are used, and that they result from considerably less computation time.

6.2 Solution to the 2D Incompressible Navier-Stokes Equations

The GDQ is now applied to solve the driven flow in a rectangular cavity. This is a standard test problem in CFD and it is governed by the Navier-Stokes equations. The non-dimensional vorticity-stream function formulation of the **2D** Navier-Stokes equations can be written as

[1] Second Order Time-Split MacCormack Finite Difference Scheme

$$\omega_t + u\cdot\omega_x + v\cdot\omega_y = \nabla^2\omega/Re \tag{6.3a}$$

$$\nabla^2\psi = \omega \tag{6.3b}$$

where ∇^2 is the Laplacian operator; ω, ψ and Re the vorticity, stream function and Reynolds number; u, v are the components of the velocity in the x and y direction, which can be calculated from the stream function

$$u = \psi_y, \quad v = -\psi_x,$$

with the boundary conditions

$$\psi = \psi_x = 0, \quad \text{at } x = 0,\ 1,\ 0 \leqslant y < 1 \tag{6.4a}$$

$$\psi = \psi_y = 0, \quad \text{at } y = 0\ ,\ 0 \leqslant x \leqslant 1 \tag{6.4b}$$

$$\psi = 0,\ \psi_y = 1, \quad \text{at } y = 1\ ,\ 0 < x < 1 \tag{6.4c}$$

Clearly, the two corners on the upper wall are singular points which in a numerical technique normally cause difficulties in treating the boundary conditions. In numerical simulation, the boundary conditions for ω can be obtained from equation (6.3b). At each solid boundary there is a boundary condition for ω and there are two boundary conditions for ψ (see (6.4)). Using GDQ, (6.3) can be discretized as

$$\frac{d\,\omega_{ij}}{dt} + u_{ij}\cdot\sum_{k=1}^{N} w_{ik}^{(1)}\,\omega_{kj} + v_{ij}\cdot\sum_{k=1}^{M}\overline{w}_{jk}^{(1)}\,\omega_{ik} = \frac{1}{\mathrm{Re}}\cdot\left[\sum_{k=1}^{N} w_{ik}^{(2)}\,\omega_{kj} + \sum_{k=1}^{M}\overline{w}_{jk}^{(2)}\,\omega_{ik}\right] \tag{6.5a}$$

$$\sum_{k=1}^{N} w_{ik}^{(2)}\,\psi_{kj} + \sum_{k=1}^{M}\overline{w}_{jk}^{(2)}\,\psi_{ik} = \omega_{ij} \tag{6.5b}$$

Similarly, the boundary conditions for ψ (6.4) and ω (6.3b) can be discretized by GDQ. It is indicated that the Neumann boundary conditions for ψ can be combined to give the solution

$$\psi_{2,j} = \frac{1}{AXN}\cdot\left[\sum_{k=1,k\neq 2,N-1}^{N}(w_{1,k}^{(1)}\cdot w_{N,N-1}^{(1)} - w_{N,k}^{(1)}\cdot w_{1,N-1}^{(1)})\cdot\psi_{k,j}\right] \tag{6.6a}$$

$$\psi_{N-1,j} = \frac{1}{AXN}\cdot\left[\sum_{k=1,k\neq 2,N-1}^{N}(w_{N,k}^{(1)}\cdot w_{1,2}^{(1)} - w_{1,k}^{(1)}\cdot w_{N,2}^{(1)})\cdot\psi_{k,j}\right] \tag{6.6b}$$

$$\psi_{i,2} = \frac{1}{AYM}\cdot\left[\overline{w}_{1,M-1}^{(1)} + \sum_{k=1,k\neq 2,M-1}^{M}(\overline{w}_{1,k}^{(1)}\cdot\overline{w}_{M,M-1}^{(1)} - \overline{w}_{M,k}^{(1)}\cdot\overline{w}_{1,M-1}^{(1)})\cdot\psi_{i,k}\right] \tag{6.6c}$$

$$\psi_{i,M-1} = \frac{1}{AYM}\cdot\left[-\overline{w}_{1,2}^{(1)} + \sum_{k=1,k\neq 2,M-1}^{M}(\overline{w}_{M,k}^{(1)}\cdot\overline{w}_{1,2}^{(1)} - \overline{w}_{1,k}^{(1)}\cdot\overline{w}_{M,2}^{(1)})\cdot\psi_{i,k}\right] \tag{6.6d}$$

where

$$AXN = w_{N,2}^{(1)}\cdot w_{1,N-1}^{(1)} - w_{1,2}^{(1)}\cdot w_{N,N-1}^{(1)}$$

$$AYM = \overline{w}^{(1)}_{M,2} \cdot \overline{w}^{(1)}_{1,M-1} - \overline{w}^{(1)}_{1,2} \cdot \overline{w}^{(1)}_{M,M-1}$$

With the boundary conditions, the set of (N-2)×(M-2) ordinary differential equations for ω was solved by the 4-stage Runge-Kutta scheme, and the set of (N-4)×(M-4) algebraic equations for ψ was solved by LU decomposition. Noting that the Laplacian operator is a linear operator, we need only decompose the matrix of the equations system once and store the inverted matrix elements for all the following time steps. For numerical simulation, the solutions were obtained in the Reynolds numbers range from 100 to 1000. The grid of case IV shown in above section was used for the GDQ simulation. Mesh sizes of 13×13, 17×15, 21×17 and 23×21 for Reynolds numbers of 100, 200, 400, 1000 were respectively used. The initial values for all variables in the interior points are chosen to be zero.

Table II Parameters Defining the Vortex Centre for Driven Cavity Flow

Re	Reference	Grid	x	y	ψ	ω
100	Ghia et al	129×129	0.6172	0.7344	-0.1034	3.1665
	Present (GDQ)	13×13	0.6150	0.7350	-0.1035	3.1547
	Present (FD)	51×51	0.6200	0.7400	-0.1030	3.1915
200	Ku et al	25×15	0.6023	0.6657	-0.1071	2.6345
	Present (GDQ)	17×15	0.6000	0.6650	-0.1089	2.6686
	Present (FD)	51×51	0.6000	0.6600	-0.1072	2.6673
400	Ghia et al	129×129	0.5547	0.6055	-0.1139	2.2947
	Present (GDQ)	21×17	0.5550	0.6050	-0.1131	2.2794
	Present (FD)	51×51	0.5600	0.6000	-0.1105	2.2428
1000	Ghia et al	129×129	0.5313	0.5625	-0.1179	2.0497
	Present (GDQ)	23×21	0.5300	0.5650	-0.1184	2.0649
	Present (FD)	51×51	0.5400	0.5600	-0.1103	1.9326

For direct comparison of GDQ with conventional numerical techniques, numerical results using a second order time-split MacCormack finite difference scheme for vorticity equation and a preconditioning technique of SIP for the stream function, are also obtained for a uniform grid of mesh size of 51×51. By numerical experiment, the allowable maximum time step size for the finite difference simulation was used. The parameters defining the vortex centre are compared in Table II. The comparison includes the results of the GDQ approximation, the finite difference approximation and results given by Ghia et al [9] and Ku et al [10]. It is clear from Table II that the GDQ approximation is very accurate even though just a few grid points were used, compared with the finite difference approximations using a large number of grid points. Table III shows the CPU time on the IBM 3090 by the GDQ approximation and the time-split MacCormack finite difference approximation. The mesh size used is the same as that given in Table II. We see, clearly, that the GDQ approach requires much less CPU time for accurate results.

Table III The CPU Time Taken by Driven Cavity Flow Simulation

Re	100	200	400	1000
GDQ (seconds)	4.27	6.69	16.99	33.79
FD (seconds)	442.73	536.98	601.50	732.90

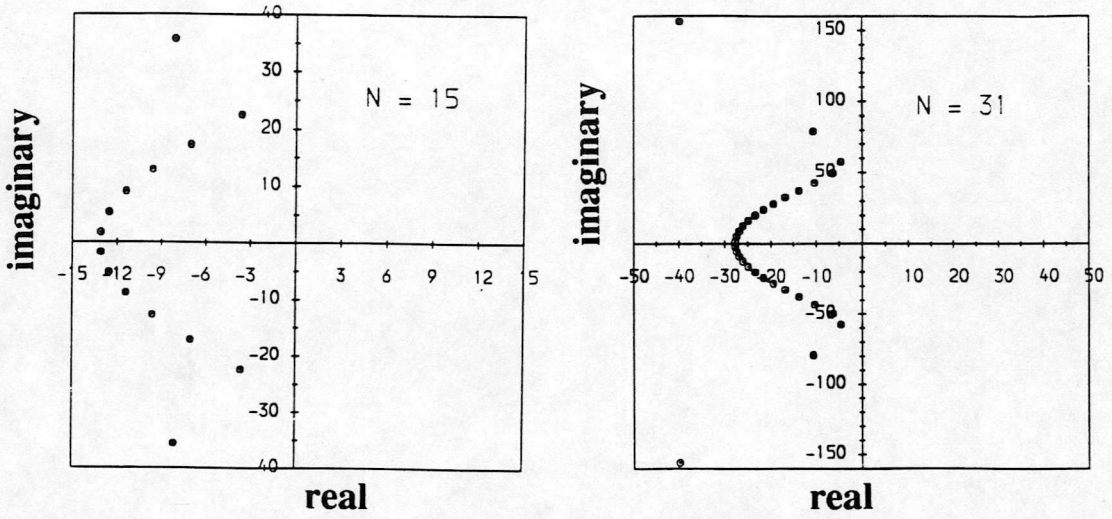
7. CONCLUSIONS

The technique of differential quadrature has been extended to a general case. Generalized differential quadrature has overcome the difficulty of differential quadrature in obtaining the weighting coefficients of the first order derivative when a large number of grid points with an arbitrary distribution are applied. Furthermore, a recurrence relationship has been established for determination of the weighting coefficients of the second and higher order derivatives. It has been proved that the GDQ approach can be considered as the highest order finite difference scheme, and when the coordinates of grid points are chosen as the roots of a

Chebyshev polynomial, the formulation of the first order derivative discretization obtained by GDQ is exactly the same as that given by the Chebyshev pseudospectral method. The error estimations for the function and derivative approximation have also been analysed. The distribution of the grid points was found to have a considerable influence on the stability condition. Grid stretching near the boundary can improve the stability, but the grid stretching near the middle point makes it worse even though the minimum step size is very small. Application of GDQ to solve Burger's equation and the 2D Navier-Stokes equations showed that accurate numerical results can be obtained using just a few grid points, and require much less computational time.

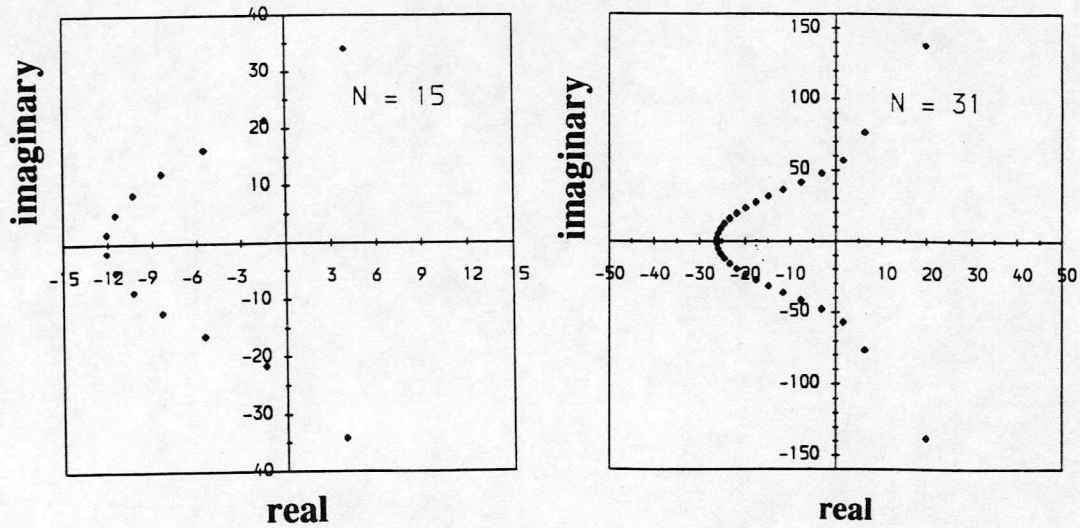
REFERENCES

1. R. Bellman, B.G. Kashef and J. Casti, 'Differential Quadrature: A Technique for the Rapid Solution of Nonlinear Partial Differential Equations', *J. Comput. Phys.* **10**, (1972), 40-52
2. R. Bellman, G. Naadimuthu, K.M. Wang, and E.S. Lee, 'Differential Quadrature and Partial Differential Equations: Some Numerical Results', *J. Math. Anal. Appl.* **98**, (9184), 220-235
3. F. Civan and C.M. Sliepcevich, 'Application of Differential Quadrature to Transport Processes', *J. Math. Anal. Appl.* **93**, (1983), 206-221
4. F. Civan and C.M. Sliepcevich, 'Differential Quadrature for Multi-Dimensional Problems', *J. Math. Anal. Appl.* **101**, (1984),423-443
5. J.O. Mingle, 'The Method of Differential Quadrature for Transient Nonlinear Diffusion', *J. Math. Anal. Appl.* **60**, (1977), 559-569
6. S.K. Jang, C.M. Bert, and A.G. Striz, 'Application of Differential Quadrature to Static Analysis of Structural Components', *Int. J. Numer. Methods Eng.* 28, (1989), 561-577
7. C. Shu, 'Generalized Differential-Integral Quadrature and Application to the Simulation of Incompressible Viscous Flows Including Parallel Computation', *PhD Thesis*, University of Glasgow, July 1991
8. U. Ehrenstein and R. Peyret, 'A Chebyshev Collocation Method for the Navier-Stokes Equations with Application to Double-Diffusive Convection', *Int. J. Numer. Methods Fluids* **9**, (1989), 427-452
9. U.Ghia, K.N. Ghia, and C.T. Shin, 'High Resolutions for Incompressible Flow Using the Navier-Stokes Equations and A Multi-Grid Method', *J. Compt. Phys.* **48**, (1982), 387-411
10. H.C. Ku and D. Hatziavramidis, 'Solutions of the Two-Dimensional Navier-Stokes Equations by Chebyshev Expansion Methods', *Comput. Fluids* **13**, (1985), 99-113



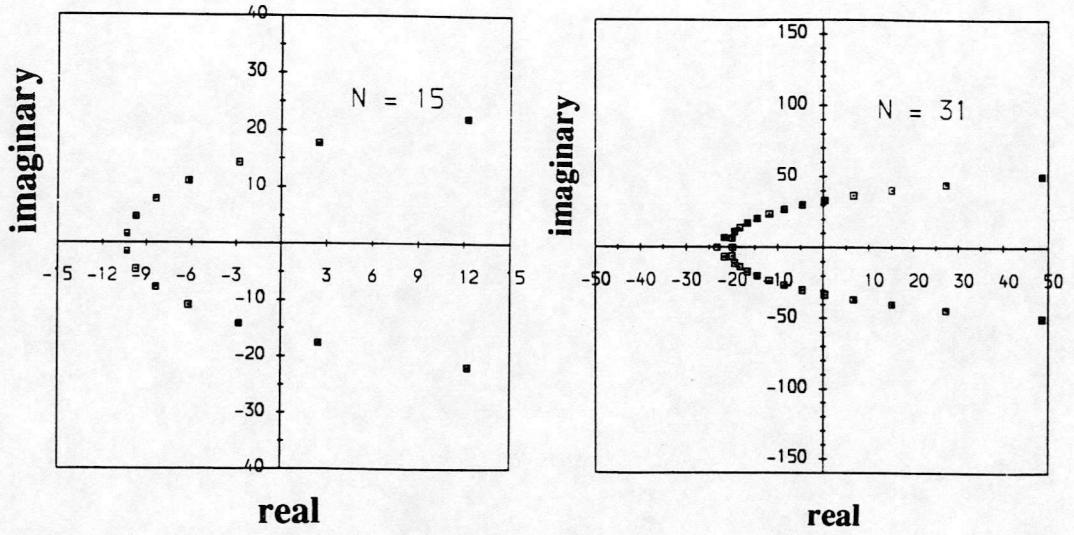
(a) $|\lambda|_{max}$ =36.5, $|\Delta x|_{min}$=0.0125 (b) $|\lambda|_{max}$=161.1, $|\Delta x|_{min}$=0.00274

Fig. 1 Convection Operator Eigenvalues with Grid of Case I



(a) $|\lambda|_{max}$ =34.4, $|\Delta x|_{min}$=0.0219 (b) $|\lambda|_{max}$=139.4, $|\Delta x|_{min}$=0.00513

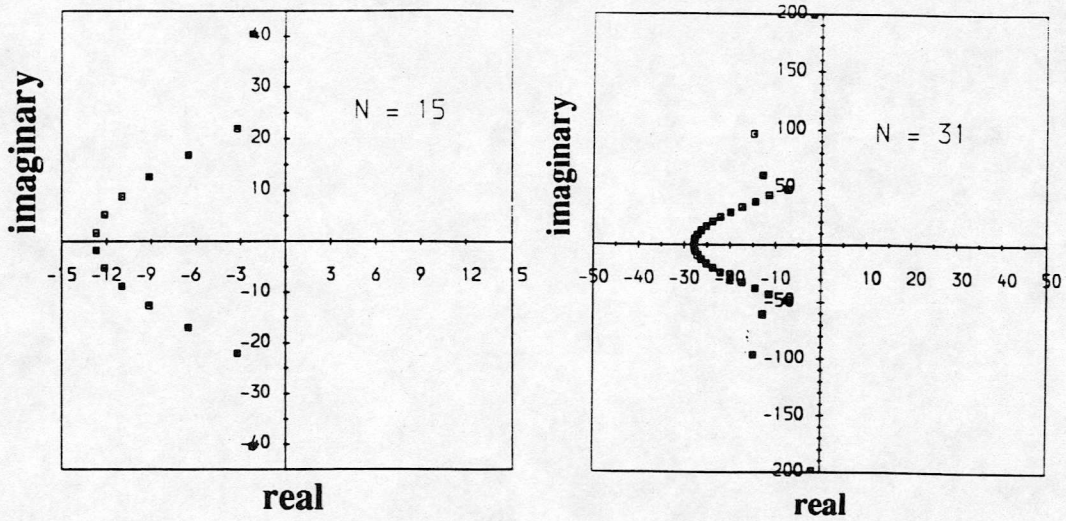
Fig. 2 Convection Operator Eigenvalues with Grid of Case II



(a) $|\lambda|_{max}$ =25.1, $|\Delta x|_{min}$=0.0714

(b) $|\lambda|_{max}$=69.89, $|\Delta x|_{min}$=0.03333

Fig. 3 Convection Operator Eigenvalues with Grid of Case III



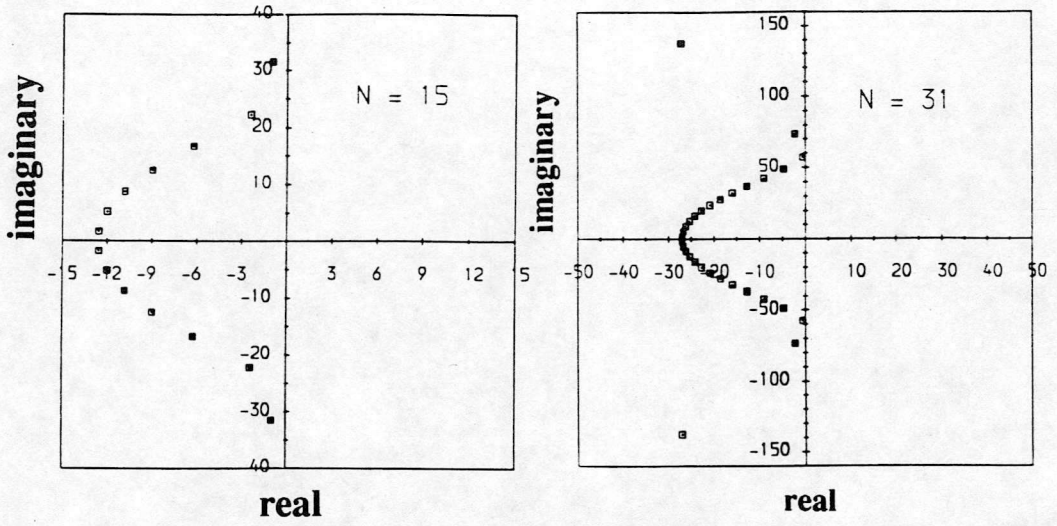
(a) $|\lambda|_{max}$ =40.5, $|\Delta x|_{min}$=0.0159

α = 0.71

(b) $|\lambda|_{max}$=199.1, $|\Delta x|_{min}$=0.00336

α = 0.65

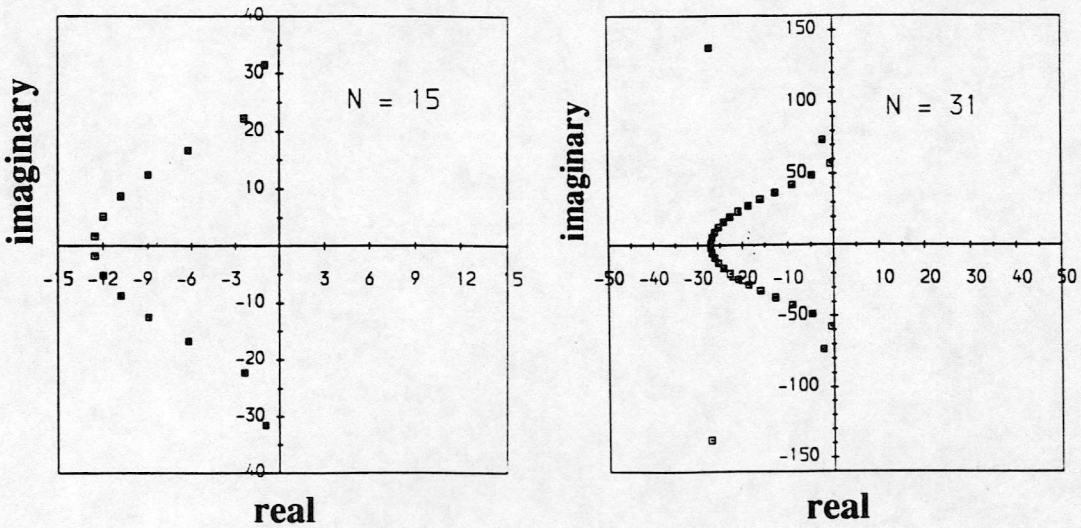
Fig. 4 Convection Operator Eigenvalues with Grid of Case IV



(a) $|\lambda|_{max}$ =34.5, $|\Delta x|_{min}$=0.0163 (b) $|\lambda|_{max}$=140.4, $|\Delta x|_{min}$=0.00341

α = 0.03 α = 0.005

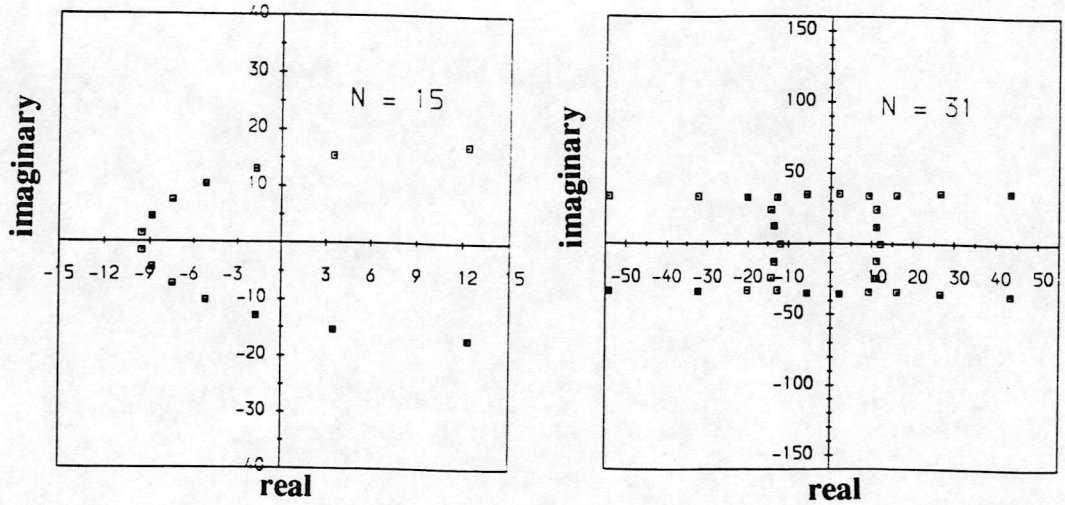
Fig. 5 Convection Operator Eigenvalues with Grid of Case V



(a) $|\lambda|_{max}$ =31.5, $|\Delta x|_{min}$=0.0159 (b) $|\lambda|_{max}$=140.3, $|\Delta x|_{min}$=0.00315

α = 1.28 α = 1.15

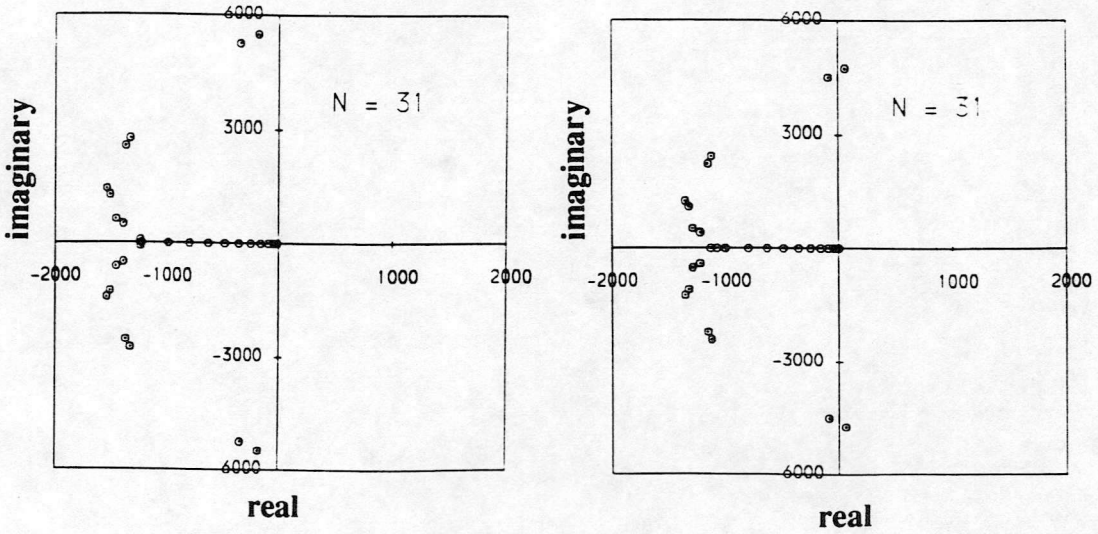
Fig. 6 Convection Operator Eigenvalues with Grid of Case VI



(a) $|\lambda|_{max}$ =21.1, $|\Delta x|_{min}$=0.01204 (b) $|\lambda|_{max}$=63.81, $|\Delta x|_{min}$=0.00844

β = 0.03 β = 0.20

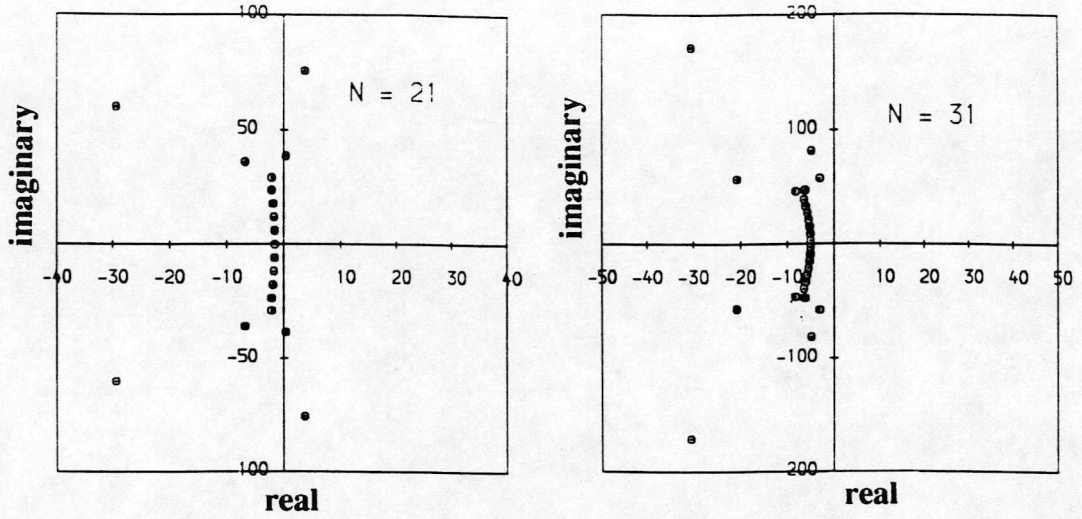
Fig. 7 Convection Operator Eigenvalues with Grid of Case VII



(a) Dirichlet Type Condition (b) Neumann Type Condition

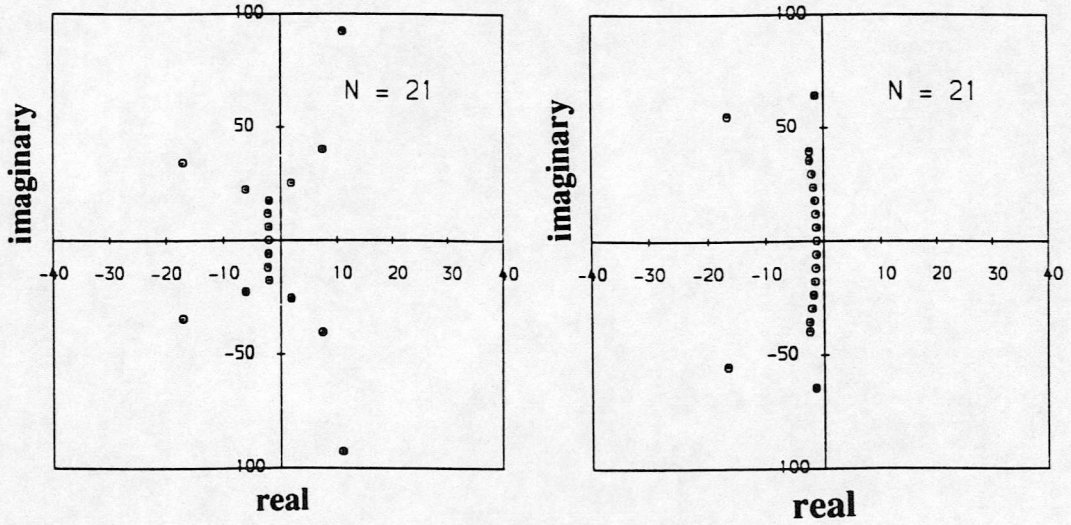
$|\lambda|_{max}$= 5.493×10^3 $|\lambda|_{max}$= 4.734×10^3

Fig. 8 Diffusion Operator Eigenvalues with Grid of Case III



(a) $\nu = 0.001$, $|\lambda|_{max}=75.75$ (b) $\nu = 0.001$, $|\lambda|_{max}=173.81$

Fig. 9 Diffusion-Convection Operator Eigenvalues with Grid of Case I



(a) Stretched near the boundary (b) Relaxed near the boundary

$\nu=0.001$, $\alpha=0.8$, $|\lambda|_{max}=92.22$ $\nu=0.001$, $\alpha=1.2$, $|\lambda|_{max}=64.35$

Fig. 10 Diffusion-Convection Operator Eigenvalues with Grid of Case VI