University of Glasgow

DEPARTMENT OF

AEROSPACE ENGINEERING

Generalized Differential-Integral Quadrature and the Solution of the Boundary Layer Equations

Chang SHU and Bryan E. Richards

G.U. Aero Report 9122

Generalized Differential-Integral Quadrature and the Solution of the Boundary Layer Equations

Chang SHU and Bryan E. Richards

G.U. Aero Report 9122

November, 1991

ABSTRACT

The global methods of generalized differential quadrature (GDQ) and generalized integral quadrature (GIQ) for solutions of partial differential and integral equations are presented in this paper. These methods approximate the derivatives and integrals by a linear combination of all the functional values in the overall domain, where the weighting coefficients can be readily identified. The error estimations of GDQ and GIQ have also been analysed. Application of GDQ and GIQ to solve boundary layer equations demonstrated that accurate numerical results can be obtained using just a few grid points.

1. INTRODUCTION

In seeking an efficient method using just a few grid points to get an accurate solution of a partial differential equation, Bellman et al [1] introduced a global method of differential quadrature, where any partial derivative at a discrete point is approximated by a linear weighted sum of all the functional values in the whole domain. The key to this scheme is how to determine the weighting coefficients. They suggested two methods to determine the weighting coefficients of the first order derivative. One method solves a set of algebraic equations. Unfortunately, when N (the number of grid points) is large the matrix of the equation system is ill-conditioned. The second method computes the weighting coefficients by an algebraic formulation with coordinates of grid points chosen as the roots of an Nth order shifted Legendre polynomial. This means that if N is specified, the distributions of grid points are the same for different physical problems. This restricts the application of, and hence provides a major drawback to, differential quadrature. In order to overcome the drawbacks described above, the generalized differential quadrature (GDQ) was then developed [2], and presented in this paper.

Based on the same concept as GDQ, the generalized integral quadrature (GIQ) was also developed. If a function is continuous in the whole domain, then the integral of the function over a part of the whole domain can be approximated by GIQ with high order accuracy even though the integral domain contains only two points. When the integral domain becomes the whole domain, GIQ reduces to the conventional integral quadrature. Obviously, GIQ greatly improves the accuracy of conventional integral quadrature when the integral domain contains

just a few grid points.

The boundary layer approximation is still an interesting area in CFD because it greatly reduces the computational effort compared with a Navier-Stokes solver. For numerical solution of boundary layer equations, low order finite difference schemes are usually used to discretize the continuity, momentum and energy equations. The reason for not using the integral form of the continuity equation is that the normal velocity obtained by integrating the equation along the normal coordinate is less accurate because some integral domains do not contain sufficient grid points. As will be shown in this paper, the GIQ technique can provide a promising way to obtain the normal velocity accurately by an explicit formulation derived from the integration of the continuity equation. The determination of the normal velocity at any mesh point of the normal coordinate direction has the same order of accuracy. We will use both the GDQ and GIQ techniques in the normal direction for discretizing the derivatives and the integrals. In the streamwise direction, both GDQ and low order finite difference schemes can be used. We will show that the GDQ-GIQ approach can be applied for both the case where the dependent variable is the stream function and the case where the dependent variable is the primitive variable.

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. The key technique to this procedure is how to determine the weighting coefficients a_{ij}. Bellman et al suggested two methods to carry this out. The first method is to let (2.1) be exact for test functions $g_k(x)=x^k$, 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.

polynomial in V_N can be expressed uniquely as a linear combination of $r_k(x)$, k=1, 2, ···, N.

Equation (2.1) is a linear constrained relationship. If the base polynomials $r_k(x)$, k=1, 2, ···, N, satisfy (2.1), so does any polynomial in V_N, 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

$$r_k(x) = \frac{M(x)}{(x - x_k)\cdot M^{(1)}(x_k)} \tag{3.1}$$

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

$$M^{(m)}(x) = N^{(m)}(x, x_k)\cdot(x - x_k) + m\cdot N^{(m-1)}(x, x_k) \tag{3.2}$$

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 (3.1) into (2.1) and using (3.2), we obtain

$$a_{ij} = \frac{M^{(1)}(x_i)}{(x_i - x_j)\cdot M^{(1)}(x_j)} \quad , \quad for\ j \neq i \tag{3.3a}$$

$$a_{ii} = \frac{M^{(2)}(x_i)}{2M^{(1)}(x_i)} \tag{3.3b}$$

for i, j =1, 2, ···, N.

Equation (3.3) 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 easily 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. 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 :

$$\sum_{j=1}^{N} a_{ij} = 0 \tag{3.4}$$

where a_{ii} can be easily determined from a_{ij} ($i \neq j$). Equation (3.3) 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, (3.3) is exactly the same as that given by Bellman's second method.

3.2 Weighting Coefficients of the Second and Higher Order Derivatives

For the case of discretization of the second and higher order derivatives, the linear constrained relations are applied as follows

$$u_x^{(m-1)}(x_i,t) = \sum_{j=1}^{N} w_{ij}^{(m-1)} \cdot u(x_j,t) \tag{3.5}$$

$$u_x^{(m)}(x_i,t) = \sum_{j=1}^{N} w_{ij}^{(m)} \cdot u(x_j,t) \tag{3.6}$$

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, and $w_{ij}^{(m-1)}$, $w_{ij}^{(m)}$ are the weighting coefficients related to $u_x^{(m-1)}(x_i,t)$ and $u_x^{(m)}(x_i,t)$. Substituting (3.1) into (3.5), (3.6) and using (3.2), (3.3), a recurrence formulation is obtained as follows

$$w_{ij}^{(m)} = m \cdot (a_{ij} \cdot w_{ii}^{(m-1)} - \frac{w_{ij}^{(m-1)}}{x_i - x_j}) \quad , \qquad j \neq i \tag{3.7}$$

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

where a_{ij} is 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 :

$$\sum_{j=1}^{N} w_{ij}^{(m)} = 0 \tag{3.8}$$

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

4. GENERALIZED INTEGRAL QUADRATURE

It is supposed that a function f(x) is continuous in the overall domain [a, b], which can be decomposed into N-1 intervals with grid points as x_1=a, x_2, ···, x_N=b. Since f(x) is continuous in the whole domain, it can be approximated by an *(N-1)*th order polynomial. In particular, when the functional values at N grid points are known, f(x) can be approximated by the Lagrange interpolated polynomial which are related to the functional values at all grid points. As a result, the integral of this approximated polynomial over [x_i, x_j] may involve the functional values outside the integral domain. As a general case, it is assumed that the integral of f(x) over a part of the whole domain can be approximated by a linear combination of the functional values in the overall domain with the form

$$\int_{x_i}^{x_j} f(x)\cdot dx = \sum_{k=1}^{N} c_k^{ij}\cdot f(x_k) \tag{4.1}$$

where x_i, x_j are numbers that can be altered. When x_i=a, x_j=b, (4.1) reduces to a traditional numerical integral. In a similar fashion to the analysis in the previous section, the *(N-1)*th order polynomial, which is an approximation to f(x), constitutes an N-dimensional linear vector space. Thus if all the base polynomials satisfy (4.1), so does any polynomial in the space. If the Lagrange interpolated polynomials, $r_k(x)$, k = 1, 2, ···, N, are chosen as the base polynomials, c_k^{ij} can be determined by

$$c_k^{ij} = \int_{x_i}^{x_j} r_k(x)\cdot dx \tag{4.2}$$

The expression of c_k^{ij} is very complicated. Therefore, it is difficult to calculate c_k^{ij} accurately using (4.2). We will turn to another way to determine c_k^{ij}. Setting

$$f(x) = \frac{du(x)}{dx} \quad , \tag{4.3}$$

we see clearly that if f(x) is an *(N-1)*th order polynomial, u(x) should be an *N*th order polynomial. Furthermore, if u(x) is an *N*th order polynomial without a constant term, f(x) can still be an *(N-1)*th order polynomial. Thus it is supposed that u(x) has N terms with the form

$$u(x) = x\cdot(a_0 + a_1\cdot x + \cdots + a_{N-1}\cdot x^{N-1}) \tag{4.4}$$

It is clear from (4.4), that u(x) constitutes an N dimensional linear vector space. One set of its base polynomials can be chosen as

$$p_k(x) = x\cdot r_k(x) \quad , \quad k = 1, 2, \cdots, N \tag{4.5}$$

where $r_k(x)$ is the Lagrange interpolated polynomial. Similar to formulation (2.1), we can set

$$u_x(x_i) = \sum_{j=1}^{N} \underline{a}_{ij} \cdot u(x_j) \tag{4.6}$$

Formulation (4.6) can be written as a vector form

$$\boldsymbol{U}_x = \underline{A}\boldsymbol{U} \tag{4.7}$$

where

$$\boldsymbol{U} = [u(x_1), u(x_2), \cdots, u(x_N)]^T$$

$$\boldsymbol{U}_x = [u_x(x_1), u_x(x_2), \cdots, u_x(x_N)]^T$$

and $\underline{A}$ is a matrix composed by $\underline{a}_{ij}$.

Integrating (4.3) yields

$$u(x) = \int_c^x f(t)\cdot dt + F(c) \tag{4.8}$$

where c is a constant, $c \in [a,b]$, F(c) guarantees that u(x) has no constant term. Setting

$$\boldsymbol{f}^I = [\int_c^{x_1} f(x)\cdot dx, \int_c^{x_2} f(x)\cdot dx, \cdots, \int_c^{x_N} f(x)\cdot dx]^T$$

$$\boldsymbol{I} = [1, 1, \cdots, 1]^T$$

$$\boldsymbol{f} = \boldsymbol{U}_x$$

then (4.7) can be written as

$$\boldsymbol{f} = \underline{A}\cdot(\boldsymbol{f}^I + F(c)\cdot\boldsymbol{I}) \quad . \tag{4.9}$$

Setting

$$W^I = \underline{A}^{-1} \tag{4.10}$$

we then obtain

$$f^I = W^I \cdot f - F(c) \cdot I \quad . \tag{4.11}$$

The scalar form of (4.11) can be written as

$$\int_c^{x_i} f(x) \cdot dx = \sum_{k=1}^{N} w_{ik}^I \cdot f(x_k) - F(c) \tag{4.12}$$

for i = 1, 2, ···, N.

Thus

$$\int_{x_i}^{x_j} f(x) \cdot dx = \sum_{k=1}^{N} (w_{jk}^I - w_{ik}^I) \cdot f(x_k) \tag{4.13}$$

$$c_k^{ij} = w_{jk}^I - w_{ik}^I \quad . \tag{4.14}$$

We will discuss how to determine $\underline{A}$ through the two following cases.

Case I: The Integral Domain not Including the Origin

Supposing b > a, it is assumed that the integral domain does not include the origin, i.e. a > 0 or b < 0. Substituting (4.5) into (4.6) yields

$$\underline{a}_{ij} = \frac{x_i}{x_j} \cdot a_{ij} \quad , \qquad when\ i \neq j \tag{4.15a}$$

$$\underline{a}_{ii} = a_{ii} + \frac{1}{x_i} \tag{4.15b}$$

for i, j = 1, 2, ···, N.

Equation (4.15) requires $x_i \neq 0$, for i =1, 2, ···, N. This is guaranteed by the condition of a > 0 or b < 0.

Case II: The Integral Domain Including the Origin

If the integral domain includes the origin, (4.15) may be singular. This problem can be removed by the following transformation :

$$\underline{x} = x + d \tag{4.16}$$

where $\underline{x}$ is the transformed coordinate, and d is a constant which guarantees that the transformed integral domain does not include the origin, i.e. $\underline{a}$ = a + d > 0 or $\underline{b}$ = b + d < 0. Then (4.15) is held in the domain [$\underline{a}$, $\underline{b}$]. Using (4.16), we get $\underline{a}_{ij}$, in this case, as

$$\underline{a}_{ij} = \frac{x_i + d}{x_j + d} \cdot a_{ij} \quad , \qquad when\ i \neq j \tag{4.17a}$$

$$\underline{a}_{ii} = a_{ii} + \frac{1}{x_i + d}$$

for i, j = 1, 2, ···, N. (4.17b)

5. ERROR ESTIMATIONS

5.1 The Function Approximation

Firstly, we discuss the approximation error when f(x) is approximated by an *(N-1)*th order polynomial, in particular by the Lagrange interpolation polynomial :

$$P_N f = \sum_{i=1}^{N} f(x_i) \cdot r_i(x) \tag{5.1}$$

We define the approximation error of f(x) as

$$E(f) = f(x) - P_N f \ . \tag{5.2}$$

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

$$\begin{aligned} f(x) &= f(c) + f^{(1)}(c)\cdot(x-c) + \cdots + f^{(k)}(c)\cdot(x-c)^k / k! + \cdots \\ &\quad + f^{(N-1)}(c)\cdot(x-c)^{N-1} / (N-1)! + f^{(N)}(\xi)\cdot(x-c)^N / N! \\ &= m_0 + m_1 x + m_2 x^2 + \cdots + m_{N-1}\cdot x^{N-1} + K\cdot x^N / N! \end{aligned} \tag{5.3}$$

where c is a constant, and $\xi \in [x, c]$. Since (5.1) is exactly satisfied for a polynomial of degree less than or equal to N-1, we have

$$E(x^k) = 0, \quad \text{when } k = 0, 1, \cdots, N-1. \tag{5.4}$$

Substituting (5.3) into (5.2) and using (5.4), we obtain

$$E(f) = K\cdot E(x^N)/N! \tag{5.5}$$

where

$$E(x^N) = x^N - \sum_{i=1}^{N} x_i^N \cdot r_i(x) \tag{5.6}$$

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 (5.1), we obtain

$$\sum_{i=1}^{N} x_i^N \cdot r_i(x) = x^N - M(x) \tag{5.7}$$

Finally, we get

$$E(f) = K \cdot M(x)/N! \tag{5.8}$$

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

$$F(z) = f(z) - \phi(z) - a \cdot M(z) \ . \tag{5.9}$$

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

$$E(f) = f(x) - P_N f = f(x) - \phi(x) = a \cdot M(x) \ . \tag{5.10}$$

Since F(z) has N+1 roots in the domain, then by repeated application of Rolle's theorem, the Nth 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

$$F^{(N)}(\xi) = 0 \ . \tag{5.11}$$

From (5.9) and (5.11), we obtain

$$a = f^{(N)}(\xi)/N! \ , \tag{5.12}$$

so, $$E(f) = f^{(N)}(\xi) \cdot M(x)/N! \ . \tag{5.13}$$

Generally, ξ is a function of x.

5.2 The Derivative Approximation

We define the error for the mth order derivative approximation as

$$E_D^{(m)}(f) = \frac{\partial^m f}{\partial x^m} - \frac{\partial^m (P_N f)}{\partial x^m} = \frac{\partial^m f}{\partial x^m} - \frac{\partial^m \phi}{\partial x^m} \tag{5.14}$$

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

$$E_D^{(m)}(f) = \frac{1}{N!} \cdot \frac{\partial^m [f^{(N)}(\xi) \cdot M(x)]}{\partial x^m} \tag{5.15}$$

Since ξ is an unknown function of x, it is difficult to estimate $E_D^{(m)}(f)$ using (5.15). As a special case, if we assume that the Nth order derivative of f(x) is a constant, namely K, then from (5.8), we get

$$E_D^{(m)}(f) = K \cdot M^{(m)}(x)/N! \tag{5.16}$$

Although (5.16) is satisfied for the condition of $f^{(N)}(\xi) = K$, it is useful in the error analysis. Firstly, (5.16) 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

$$F^{(m)}(z) = g^{(m)}(z) - \underline{a}\cdot\underline{M}(z) = f^{(m)}(z) - \phi^{(m)}(z) - \underline{a}\cdot\underline{M}(z) \qquad (5.17)$$

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

$$E_D^{(m)}[f(\underline{x})] = f^{(m)}(\underline{x}) - \phi^{(m)}(\underline{x}) = \underline{a}\cdot\underline{M}(\underline{x}) \ . \qquad (5.18)$$

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 from (5.17), (5.18), we have

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

$$E_D^{(m)}[f(\underline{x})] = f^{(N)}(\xi)\cdot\underline{M}(\underline{x})/(N-m)! \qquad (5.19)$$

Equation (5.19) is satisfied for $\underline{x} \neq \underline{x}_1, \underline{x}_2, \cdots, \underline{x}_{N-m}$.

5.3 The Integral Approximation

The error of the numerical integral of f(x) in the domain $[x_i, x_j]$ is defined as

$$E_I(f, x_i, x_j) = \int_{x_i}^{x_j} [f(x) - \phi(x)]\cdot dx$$

$$= \int_{x_i}^{x_j} f(x)\cdot dx - \sum_{k=1}^{N}(w_{jk}^{I} - w_{ik}^{I})\cdot f(x_k) \tag{5.20}$$

where w_{ij}^{I} is the weighting coefficient of the integral described in the previous section. For a general case, using (5.13), we get

$$E_I(f, x_i, x_j) = \frac{1}{N!}\cdot \int_{x_i}^{x_j} f^{(N)}(\xi)\cdot M(x)\cdot dx \tag{5.21}$$

If the integral domain is $[x_i, x_{i+1}]$, then M(x) does not change its sign in $[x_i, x_{i+1}]$. By using the second mean-value theorem, (5.21) can be reduced to

$$E_I(f, x_i, x_{i+1}) = \frac{f^{(N)}(\eta)}{N!}\cdot \int_{x_i}^{x_{i+1}} M(x)\cdot dx \tag{5.22}$$

Generally, M(x) may change its sign in the domain $[x_i, x_j]$, but |M(x)| is always positive in the domain. If it is assumed that $|f^{(N)}(\xi)| \leqslant C$, then (5.21), (5.22) can be rewritten as

$$|E_I(f, x_i, x_j)| \leq \frac{C}{N!}\cdot \int_{x_i}^{x_j} |M(x)|\cdot dx \tag{5.23}$$

$$|E_I(f, x_i, x_{i+1})| \leq \frac{C}{N!}\cdot \left| \int_{x_i}^{x_{i+1}} M(x)\cdot dx \right| \tag{5.24}$$

6. SOLUTIONS OF BOUNDARY LAYER EQUATIONS

We now apply the GDQ-GIQ approach to solve the boundary layer equations. Two cases are considered, each of which is illustrated by a test example.

6.1 Stream Function Chosen as the Dependent Variable

For simplicity, we choose the Blasius boundary layer as a test example, which is governed by

$$\frac{\partial^3 f}{\partial \eta^3} + f\cdot \frac{\partial^2 f}{\partial \eta^2} = 0 \tag{6.1}$$

with boundary conditions

$$f = 0, f_\eta = 0, \quad \text{when } \eta = 0 \tag{6.2a}$$

$$f_\eta = 1, \qquad \text{when } \eta \to \infty \tag{6.2b}$$

Setting $u = f_\eta$ and introducing an unsteady term on the right side of (6.1), this equation can be written as the two following equations, which can then be solved by GDQ and GIQ.

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial \eta^2} + f \cdot \frac{\partial u}{\partial \eta} \tag{6.3}$$

$$f = \int_0^\eta u \cdot d\eta + f(0) \tag{6.4}$$

For numerical simulation, the infinite interval in the η direction can be truncated to the finite interval [0, 3]. Using GDQ and GIQ in the domain [0, 3], we can discretize equations (6.3), (6.4) respectively as

$$\frac{d u_j}{dt} = \sum_{k=1}^{M} w_{jk}^{(2)} \cdot u_k + f_j \cdot \sum_{k=1}^{M} w_{jk}^{(1)} \cdot u_k \tag{6.5}$$

$$f_j = \sum_{k=1}^{M} (w_{jk}^I - w_{1k}^I) \cdot u_k + f_1 \tag{6.6}$$

for j = 1, 2, ···, M,

where M is the number of grid points, $w_{ij}^{(m)}$ are the weighting coefficients of the mth order derivative of the function with respect to η and w_{ij}^I are the weighting coefficients of the integral along the η direction. The boundary conditions (6.2) become

$$u_1 = 0, \quad u_M = 1, \quad f_1 = 0 \tag{6.7}$$

which are easily implemented in the solution procedure. It is clear that there are two boundary conditions for u. As we will show, another boundary condition for u can be implemented if it is necessary. Referring to equation (4.12), the discretization of (6.4) can also be written as

$$f_j = \sum_{k=1}^{M} w_{jk}^I \cdot u_k - f(0) + f_1 = \sum_{k=1}^{M} w_{jk}^I \cdot u_k \tag{6.8}$$

Equation (6.8) provides another boundary condition for u, i.e.

$$\sum_{k=1}^{M} w_{1k}^I \cdot u_k = f_1 = 0 \tag{6.9}$$

It is noted that if (6.8) is used, then (6.9) should be implemented as another boundary condition for u. The set of ordinary differential equations (6.5) can be solved by the 4-stage Runge-Kutta scheme. We have studied the difference between the use of (6.6) and (6.8), (6.9). It is found that when (6.8), (6.9) are used, that is, the three boundary conditions are employed, the allowable time step size is much larger than that when (6.6) is used, that is, only two boundary conditions are implemented. For example, when N = 12, the allowable

time step size is 1.3×10^{-2} if (6.8), (6.9) are used, and is 1.0×10^{-3} if (6.6) is used. As a result, for the convergence criterion of the maximum residual reduced by 4 orders, (6.8) and (6.9) require 385 time steps and 1.03 seconds of CPU time on the IBM 3090, but (6.6) needs 5119 time steps and 12.92 seconds of CPU time. In addition, it is found that (6.8), (6.9) can give more accurate results than (6.6). For the test problem, the exact value of the wall shear stress is 1.3284. Equations (6.8), (6.9) give 1.3286 using N = 12 while (6.6) gives 1.3298 using N = 12. Fig. 1 shows the computed and the exact velocity profile of the Blasius boundary layer. Good agreement between computed and exact solutions has been achieved.

6.2 Primitive Variable Chosen as the Dependent Variable

For demonstration, we consider the two-dimensional unsteady viscous flow past a circular cylinder started impulsively from rest. This problem has been chosen as a test example by many researchers for the study of unsteady boundary layer behaviour. Unlike the steady boundary layer equations, there are arguments as to whether there exists a finite time singularity in the solution of the unsteady counterparts. For the test problem, some researchers (e.g. Bodonyi and Stewartson [3], Liakopoulos [4]) claimed that there is a finite time singularity in the solution procedure, while others (e.g. Cebeci [5]) suggested that there is no finite time singularity.

The non-dimensional form of the governing equations for this problem [4] is

$$u_x + v_y = 0 \tag{6.10}$$

$$u_t + u \cdot u_x + v \cdot u_y = u_e \cdot \frac{du_e}{dx} + u_{yy} \tag{6.11}$$

with initial condition

$$u(x,y,0) = u_e(x) = \sin(x), \quad (y \neq 0) \tag{6.12}$$

and boundary conditions

$$u(x,0,t) = v(x,0,t) = 0 \tag{6.13}$$

$$u(x,\infty,t) = u_e(x) = \sin(x) \tag{6.14}$$

$$u(0,y,t) = 0 . \tag{6.15}$$

The computational domain in the y direction can be obtained by truncating the infinite domain

to [0, 35]. Using GDQ and GIQ in the y direction, (6.10), (6.11) can be discretized as

$$v_{ij} = -\sum_{k=1}^{M} (w_{jk}^{I} - w_{1k}^{I}) \cdot (u_x)_{ik} + v_{i1} \tag{6.16}$$

$$\frac{du_{ij}}{dt} + u_{ij} \cdot (u_x)_{ij} + v_{ij} \cdot \sum_{k=1}^{M} w_{jk}^{(1)} \cdot u_{ik} = \sin(x) \cdot \cos(x) + \sum_{k=1}^{M} w_{jk}^{(2)} \cdot u_{ik} \tag{6.17}$$

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

where N is the number of grid points in the x direction, M is the number of grid points in the y direction, $w_{ij}^{(m)}$ are the weighting coefficients of the *m*th order derivative of the function with respect to y and w_{ij}^{I} are the weighting coefficients of the integral along the y direction. When the derivative of u_x is discretized by a second order finite difference scheme, the solution procedure can be marched along the x direction, but when GDQ is also used in the x direction, the marching technique is invalid. In this case, the 4-stage Runge-Kutta scheme can be used for the solution of the resultant ordinary differential equations.

The use of GDQ in the x direction is still attractive although it may increase the storage. Since GDQ can achieve the same accuracy using just a few grid points as a low order finite difference scheme using a large number of grid points, the total number of the degrees of freedom can be greatly reduced if GDQ and GIQ are used in all the coordinate directions. Thus the total storage and the computational efforts required may be reduced. For the present case, we will use GDQ and GIQ to discretize the spatial derivatives and the integral, and use the 4-stage Runge-Kutta scheme to solve the resultant ordinary differential equations.

For numerical simulation, the mesh size used is 21×31. It is found that reverse flow first starts at $\theta=180^0$ and time, t=0.644 which is in agreement with other researchers' results. As time increases, the point of zero wall shear moves along the surface of the cylinder towards the steady state value $\theta_G = 104.5^0$ (position of the Goldstein singularity). However, the computation cannot reach steady state resolution because the numerical instability breaks down the calculation at $t \cong 3.0$. Fig. 2 shows the instantaneous streamlines computed by the GDQ-GIQ approach. Clearly, when t = 2.5, some wiggles occur in the streamlines. This is because GDQ is a global method, and when the solution develops a singularity at a point, this singularity will spread in the whole computational field. To study this, we use a

second order finite difference scheme to discretize the derivatives in the x direction, the derivatives in the y direction being discretized by GDQ. We call this scheme the GDQ-GIQ-FD approach for convenience. Figure 3 shows the instantaneous streamlines computed by the GDQ-GIQ-FD approach. The mesh size used is 81×31. Compared with Fig. 2, when $t < 2.0$, the results for both approaches are nearly the same, but when $t > 2.0$, the GDQ-GIQ-FD approach gives a considerable improvement over the GDQ-GIQ approach. Fig. 4 displays the wall shear distributions. The solid lines in the figure are the results of the GDQ-GIQ-FD approach, and the symbols are the results of the GDQ-GIQ approach. It is clear that when $t > 2.0$, the GDQ-GIQ results are less accurate. Fig. 5 shows the position of zero wall shear, where the dashed line is the position of Goldstein singularity. It is seen that the unsteady computation cannot reach the position of the Goldstein singularity. Table I lists the present and other researchers' results of the position and the time of the zero wall shear.

Table I Comparison of the Time and Position of the Zero Wall Shear Stress

References	180^0	166^0	146^0	138^0	124^0	110^0
Bar-Lev and Yang [6]	0.644	0.660	0.778	0.876	1.204	2.188
Cebeci [5]	0.640	0.660	0.780	0.872	1.192	2.200
Present (GDQ-GIQ)	0.644	0.664	0.790	0.874	1.193	2.098
Present (GDQ-GIQ-FD)	0.644	0.668	0.791	0.878	1.196	2.204

7. CONCLUSIONS

The global methods of generalized differential and integral quadrature have been presented in this paper, based on the analysis of a polynomial linear vector space. GDQ approximates any spatial derivative at a collocation point by a linear sum of all the functional values in the whole domain, where the weighting coefficients of the first order derivative are given by a simple algebraic formulation, and the weighting coefficients of the second and higher order derivatives are determined by a recurrence relationship. If the function is continuous in the whole domain, then GIQ approximates the integral of the function over a part of the whole domain (including the case of a whole domain) by a linear sum of all the functional values in the

whole domain. The weighting coefficients in GIQ can be determined from those of GDQ. Application of GDQ-GIQ approach to solve boundary layer equations demonstrated that accurate numerical results can be achieved using just a few grid points. In the applications of present scheme, the dependent variable can be the stream function or the primitive variable. It is found that when the computational field has a singularity at some point, the GDQ-GIQ approach is less efficient.

The first author acknowledges the support of the University postgraduate scholarship of Glasgow and an ORS award from U. K. government during this study.

REFERENCES

[1] R. Bellman, B.G. Kashef and J. Casti, 1972, Differential Quadrature: A Technique for the Rapid Solution of Nonlinear Partial Differential Equations, *J. Comput. Phys.* **10**, pp. 40-52

[2] C. Shu, 1991, Generalized Differential-Integral Quadrature and Application to the Simulation of Incompressible Viscous Flows Including Parallel Computation, *PhD Thesis*, University of Glasgow

[3] R.J. Bodonyi and K. Stewartson, 1977, The Unsteady Laminar Boundary Layer on a Rotating Disk in a Counter-Rotating Fluid, *J. Fluid Mech.* **79**, pp. 669-688

[4] A. Liakopoulos, 1988, Pseudospectral Solutions of Separated Flows, *AIAA Paper* 88-3643-CP

[5] T. Cebeci, 1979, The Laminar Boundary Layer on a Circular Cylinder Started Impulsively from Rest, *J. Compt. Phys.* **31**, pp. 153-172

[6] M. Bar-Lev and H.T. Yang, 1975, Initial Flow Field over an Impulsively Started Circular Cylinder, *J. Fluid Mech.*, **72**, pp. 625-647

Figure Captions

Figure 1 Velocity Profile of the Blasius Boundary Layer

Figure 2 Streamlines past a Circular Cylinder, Computed by GDQ-GIQ Approach.
(a) t = 1.5; (b) t = 2.0; (c) t = 2.5

Figure 3 Streamlines past a Circular Cylinder, Computed by GDQ-GIQ-FD Approach.
(a) t = 2.0; (b) t = 2.5; (c) t = 2.8

Figure 4 Wall Shear Distributions of the Unsteady Boundary Layer.

Figure 5 Position and Time of the Zero Wall Shear.

Tables

Table I Comparison of the Time and Position of the Zero Wall Shear Stress

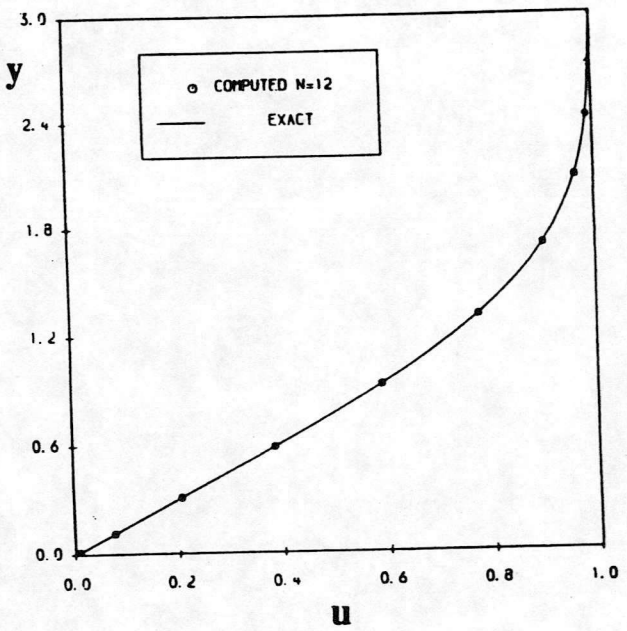
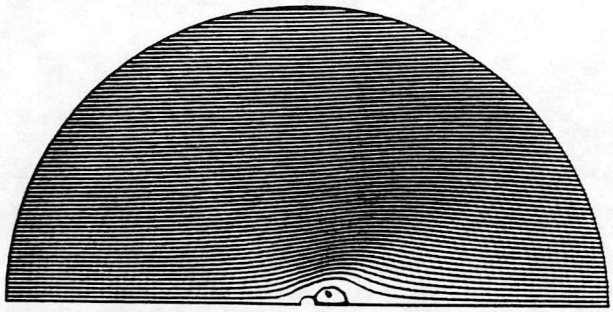
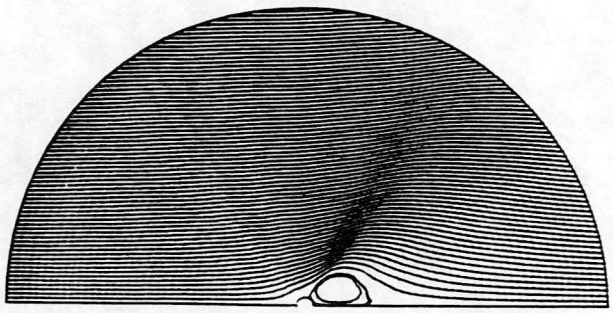


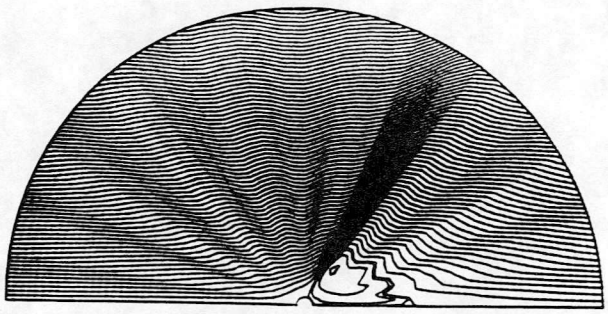
Fig. 1 Velocity Profile of the Blasius Boundary Layer



(a) t = 1.5

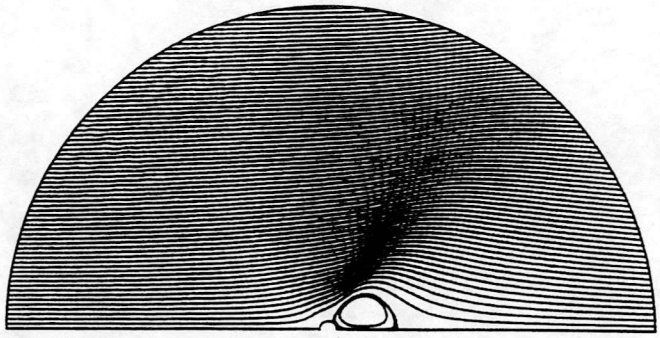


(b) t = 2.0

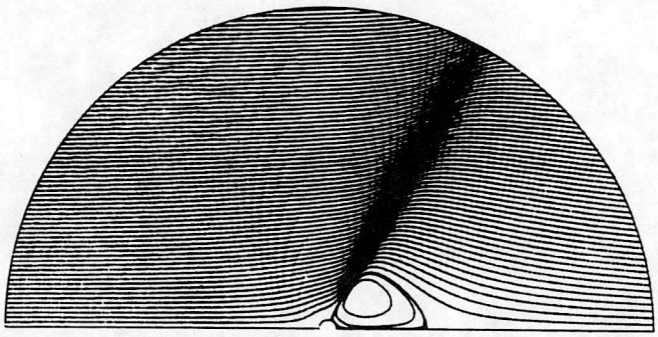


(c) t = 2.5

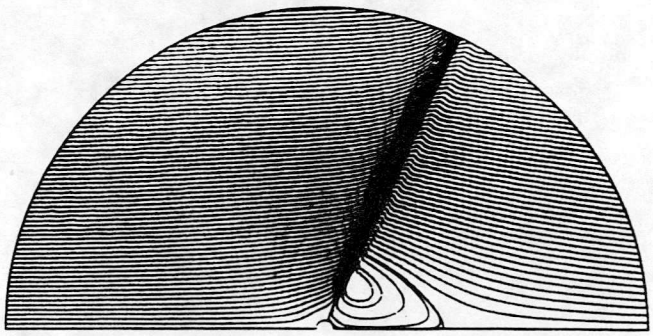
Fig. 2 Streamlines past a Circular Cylinder, Computed by GDQ-GIQ Approach



(a) t = 2.0



(b) t = 2.5



(c) t = 2.8

Fig. 3 Streamlines past a Circular Cylinder, Computed by GDQ-GIQ-FD Approach

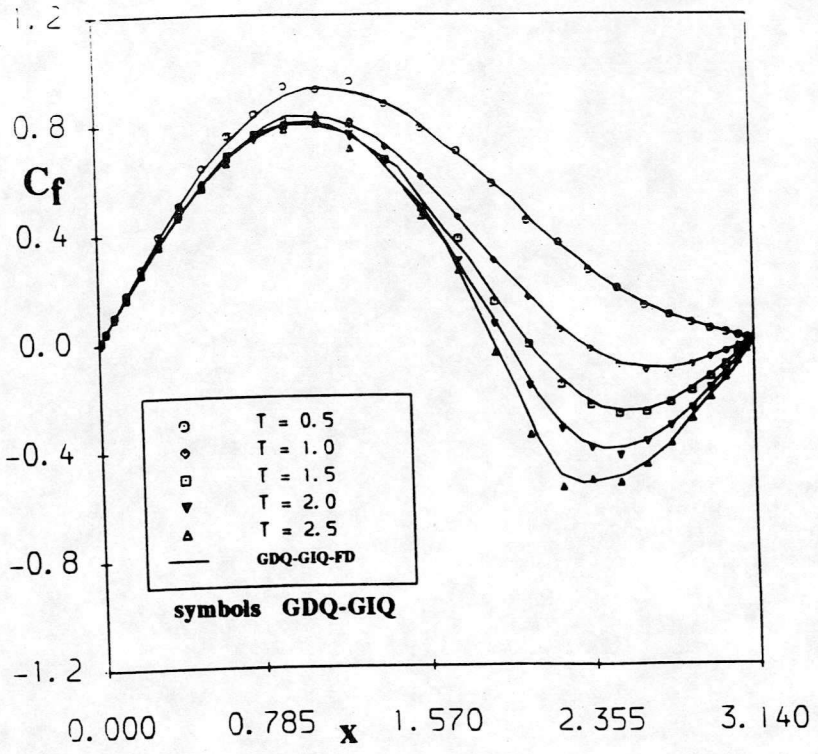


Fig. 4 Wall Shear Distributions of the Unsteady Boundary Layer

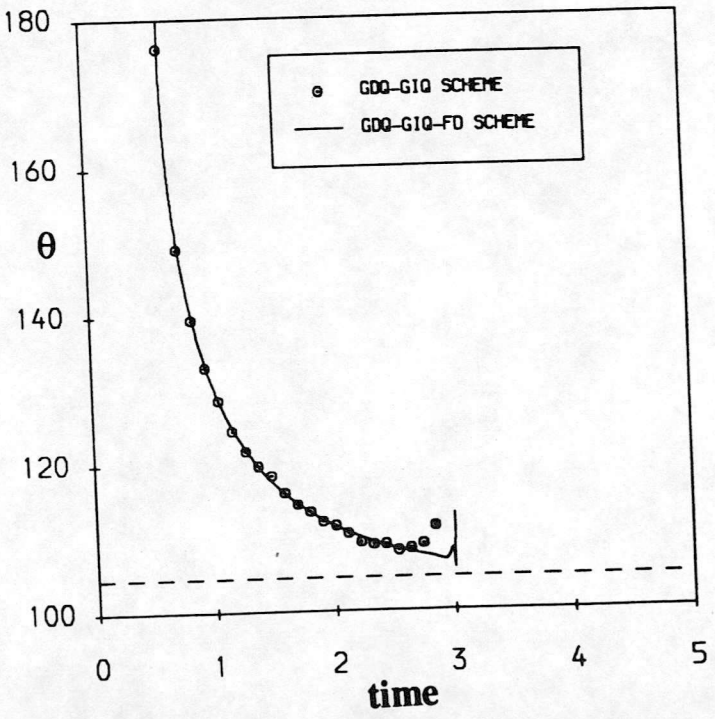


Fig. 5 Position and Time of the Zero Wall Shear