PROCEEDINGS A

royalsocietypublishing.org/journal/rspa

Research

Cite this article: Ojo SO, Trinh LC, Khalid HM, Weaver PM. 2021 Inverse differential quadrature method: mathematical formulation and error analysis. *Proc. R. Soc. A* **476**: 20200815.
https://doi.org/10.1098/rspa.2020.0815

Received: 8 October 2020
Accepted: 23 March 2021

Subject Areas:
applied mathematics, computational mathematics, differential equations

Keywords:
numerical analysis, high-order differential equations, direct approximation, inverse differential quadrature method, error analysis, numerical stability

Author for correspondence:
Saheed O. Ojo
e-mail: saheed.ojo@ul.ie

Electronic supplementary material is available online at https://doi.org/10.6084/m9.figshare.c.5367634.

THE ROYAL SOCIETY PUBLISHING

Inverse differential quadrature method: mathematical formulation and error analysis

Saheed O. Ojo, Luan C. Trinh, Hasan M. Khalid and Paul M. Weaver

Bernal Institute, School of Engineering, University of Limerick, V94 T9PX, Castletroy, Ireland

SOO, 0000-0002-1865-5086

Engineering systems are typically governed by systems of high-order differential equations which require efficient numerical methods to provide reliable solutions, subject to imposed constraints. The conventional approach by direct approximation of system variables can potentially incur considerable error due to high sensitivity of high-order numerical differentiation to noise, thus necessitating improved techniques which can better satisfy the requirements of numerical accuracy desirable in solution of high-order systems. To this end, a novel inverse differential quadrature method (iDQM) is proposed for approximation of engineering systems. A detailed formulation of iDQM based on integration and DQM inversion is developed separately for approximation of arbitrary low-order functions from higher derivatives. Error formulation is further developed to evaluate the performance of the proposed method, whereas the accuracy through convergence, robustness and numerical stability is presented through articulation of two unique concepts of the iDQM scheme, known as Mixed iDQM and Full iDQM. By benchmarking iDQM solutions of high-order differential equations of linear and nonlinear systems drawn from heat transfer and mechanics problems against exact and DQM solutions, it is demonstrated that iDQM approximation is robust to furnish accurate solutions without losing computational efficiency, and offer superior numerical stability over DQM solutions.

1. Introduction

Engineering systems are typically governed by complex high-order differential equations which require numerical methods to provide accurate solutions. To approximate such systems, the domain of interest is discretized by means of interpolation (shape) functions and its higher derivatives, which are defined over a subdomain of interest called elements. Examples of methods available in this context are element-based methods such as finite-element method [1–3], boundary element method [4,5], finite difference method [6–8] or finite volume method [9,10]. On the other hand, a class of high-order mesh-free methods such as radial basis function networks (RBFN) [11–13], element-free Galerkin method [14–16], diffuse element method [17], or high-order collocation methods such as the Chebyshev method [18–20] and the differential quadrature method (DQM) [21–27], have been widely applied for solving engineering and science problems. In all of these methods, target derivatives of an arbitrary function are directly approximated as a weighted sum of the function in the domain. According to Mai-Duy & Tran-Cong [28], in the process of differentiation, errors of function approximation may amplify significantly as influenced by local effects of the approximant. With respect to analysis of engineering structures, such error amplification can affect the accuracy of high-order secondary variables including strains, stresses, moments, and shear forces. To this effect, indirect radial basis function networks (IRBFN) were proposed in [29]. In contrast to the RBFN approach, IRBFN approximate a derivative function using RBFN and then recover the original function by integration, which is less sensitive to noise.

DQM as proposed by Bellman and others [21,22] has received widespread attention in the research community due to spectral accuracy and fast convergence that make it desirable for vast engineering applications. For example, in the field of structural mechanics, the DQM approach has been explored for static analysis [30–32], free vibration analysis [33–35], buckling analysis [29,36] and large deflection post-buckling analysis [37]. In the context of fluid mechanics applications, DQM has been widely applied to obtain solutions of convection problems [38–40] and Navier–Stokes equations [41,42], heat transfer analysis [43,44], and magnetohydrodynamic duct flow problems [45]. It is instructive to note that in [38–40,42] the so-called localized DQM is adopted in combination with RBF since this strategy allows versatility in using meshes or meshless systems [46]. Financial engineering is another area where the merits of DQM have been explored for computational analysis [47].

To generalize DQM for engineering applications, Shu [23] proposed a general approach which admits the use of different base polynomials for computation of DQM weights. According to Shu's approach [23], computation of the differential quadrature (DQ) weights for first-order and higher-order derivatives can be generalized by appropriate choice of base vectors in the linear vector space. Furthermore, Quan & Chang [48] as well as Wang [49] contended that explicit formulation of the weighting coefficients by using test functions can overcome the ill-conditioning arising from a large number of grid points. In this regard, to allow for explicit computation of the DQ weights, setting the base vectors as base polynomials (such as Lagrange polynomials, Chebyshev polynomials or Legendre polynomials) in a linear vector space in the so-called polynomial-based differential quadrature (PDQ) method is recommended [23,48,49]. To mitigate the sensitivity of DQM solutions to grid spacing to the Runge phenomenon, Wang [49] noted that non-uniform grid distribution is required to obtain reliable solutions. Readers are referred to Wang [49] for classical examples of non-uniform grid distributions that guarantee good accuracy, numerical stability and fast convergence. On the other hand, for applications like wave propagation in space where uniformly distributed grid points are required for accurate numerical estimation, the localized DQM (LDQM), which applies DQM approximations within a small neighbourhood of the point of interest, is crucial to keep the balance between the accuracy and stability of the numerical estimates [25]. In recent times, remarkably in computational fluid mechanics applications, the so-called RBF-based DQ method, as well as the LDQM variant, are increasingly being adopted to extend the merits of the DQM, as already noted in [38–40,42]. For the

purpose of the current work, the PDQ-based approach with a non-uniform grid structure is adopted.

Despite the positive contributions of DQM to various engineering fields, application to high-order systems may suffer from numerical inaccuracy and instability on account of error accumulation in the process of differentiation. According to error analysis outlined in Shu [23], the process of differentiation via DQM may incur multiple orders of inaccuracy which increasingly deteriorates for higher-order approximations, especially at the domain boundaries. Relying on the gains of IRBFN, Wu & Ren [50] proposed a 'differential quadrature method based on approximation of the highest derivative' (DQIHD) to reformulate some engineering problems which yielded accurate solutions. Like IRBFN, and in contrast to DQM, DQIHD approximates the highest derivative in a system and obtains lower-order functions by integration of the high-order primary estimate. Although Wu & Ren [50] associated this method, i.e. DQIHD, with DQM, we cannot find any empirical relationship between DQIHD and DQM apart from the use of Lagrange polynomials for computation of weights, as also done by DQM. Moreover, the two methods use different routines to determine the weights, and the properties of resulting matrices for DQM and DQIHD are fundamentally different. In addition to this, Wu & Ren [50] did not provide comparative analysis with DQM to demonstrate the performance of DQIHD. As a result, given the DQM scheme, which primarily estimates the lowest order function in a system, and the DQIHD scheme, which primarily estimates the highest order function in a system, it is not apparent which is the best approach to adopt for system solutions.

Consistent with the idea of indirect approximation, we propose a novel inverse differential quadrature method (iDQM) for system approximation. The proposed iDQM formulation provides a general framework for approximating arbitrary functions of any order in a system and either recover lower-order functions from high-order functions by iDQM operations or obtain high-order functions from lower-order ones by DQM operations. In this context, it is possible to combine the advantages of low-order numerical differentiation and low-order numerical integration to achieve improved results. A specific case of the proposed iDQM scheme is DQIHD in which the highest order derivative in a system is approximated.

Firstly, a formulation for approximation of arbitrary low-order functions from higher derivatives (not necessarily the highest order as in DQIHD) is derived in §2. To circumvent issues arising from computational inefficiency, associated with analytical integration or numerical complexity due to Gaussian integration, we employ a unique strategy to extract iDQM weights by inversion of existing DQM formula, leading to a robust and efficient routine described in §2d, which can admit the use of different base polynomials, like DQM. Then, in §3, formulations of error estimates are developed for *iDQM-by-integration* and *iDQM-by-inversion*, which are subsequently compared with DQM error estimates developed by Shu [23]. Section 4 is dedicated to demonstrating the accuracy of the proposed method through basic implementation of iDQM for functional approximation as well as solution of high-order ordinary and partial differential equations representing linear and nonlinear systems, with examples taken from heat transfer and mechanics fields. This process eventually leads to the establishment of concepts of mixed iDQM (MiDQM) and full iDQM (FiDQM) described in §4b. Consequently, a comprehensive error analysis entailing convergence, robustness, and numerical stability of iDQM operation is illustrated with a boundary value example in §4g and §4h while §5 is dedicated to characterizing the computational efficiency of iDQM. Finally, conclusions of the present study are considered in §6.

(a) Brief introduction to differential quadrature method formulation

According to Weierstrass's first theorem [21], suppose $f(\xi)$ is a real valued continuous function defined in a closed interval $\xi \in [a\ b]$, then there exists a sequence of polynomials $P_n(\xi)$ which converge to $f(\xi)$ as n tends to infinity. Therefore, if $f(\xi)$ represents the solution of a partial

differential equation, then it can be approximated by a polynomial of a degree less than N through the mathematical relation,

$$f(\xi) \approx P_n(\xi) = \sum_{k=0}^{N-1} c_k \xi^k, \tag{1.1}$$

where c_k represents constant weights to be determined and ξ^k are the linearly independent basis vectors in the N-dimensional linear vector space, V_N. Adopting Shu's general approach [23], some sets of base polynomials can be selected to determine the weights, c_k. Furthermore, according to Shu [23], the numerical difficulty of determining the weights as N becomes large can be eliminated by considering Lagrange interpolation polynomials as basis vectors in which $P_n(\xi)$ is then approximated by

$$P_n(\xi) = \sum_{i=1}^{N} f_i l_i(\xi) + E(\xi), \tag{1.2}$$

where $E(\xi)$ is the polynomial approximation error, and $l_i(\xi)$ is a $(N-1)$ degree Lagrange polynomial defined as

$$l_i(\xi) = \frac{M(\xi)}{M^{(1)}(\xi_i)(\xi - \xi_i)}, \tag{1.3}$$

where $M(\xi) = \prod_{i=1}^{N}(\xi - \xi_i)$, and $M^{(1)}(\xi_i) = \prod_{k=1,k\neq i}^{N}(\xi_i - \xi_k)$, and $f(\xi_i)$ is the functional value at a discrete point i. In line with DQM routines outlined in [23], the first derivative of the function $f(\xi)$, $f_i^{(1)}$, at a discrete point i in the domain $\xi \in [a\ b]$ can be realized as follows,

$$f_i^{(1)} = \sum_{j=1}^{N} a_{ij}^{(1)} f_j + E^{\mathrm{dif}(1)}(\xi_i), \quad \text{for } i,j = 1,\ldots,N, \tag{1.4}$$

where $a_{ij}^{(1)} = (M^{(1)}(\xi_i)/M^{(1)})(\xi_j)(\xi_i - \xi_j)$, and $a_{ii}^{(1)} = -\sum_{j=1,j\neq i}^{N} a_{ij}^{(1)}$, for $i = j$.

In equation (1.4), $E^{\mathrm{dif}(1)}(\xi)$ is the approximation error due to first-order numerical differentiation of $f(\xi)$, which is given in [23] as

$$E^{\mathrm{dif}(1)}(\xi) \leq \frac{KM^{(1)}(\xi)}{N!}, \tag{1.5}$$

where K is a constant implied from Shu [23]. For an arbitrary derivative of order m, the DQM approximation is characterized by a weighting coefficient $a_{ij}^{(m)}$ which is evaluated based on the recursive formula,

$$\begin{aligned} a_{ij}^{(m)} &= m\left(a_{ij}^{(1)} a_{ii}^{(m-1)} - \frac{a_{ij}^{(m-1)}}{\xi_i - \xi_j}\right), \quad \text{for } i,j = 1,\ldots,\ m = 2,3,\ldots,N-1, \\ a_{ii}^{(m)} &= -\sum_{j=1,j\neq i}^{N} a_{ij}^{(m)}. \end{aligned} \tag{1.6}$$

In a compact form, the DQM approximation of mth-order derivative of $f(\xi)$ is represented thus

$$\mathbf{F}^{(m)} = \mathbf{D}^{(m)}\mathbf{F} + \mathbf{E}^{\mathrm{dif}(m)} \quad m = 2,3,\ldots,N-1, \tag{1.7}$$

where $\mathbf{F}^{(m)} \in \mathcal{R}^{N\times 1}$ is the mth order derivative of the vector function $\mathbf{F} = [f_1, f_2, \ldots, f_N]^T$, $\mathbf{F} \in \mathcal{R}^{N\times 1}$, while $\mathbf{D}^{(m)} \in \mathcal{R}^{N\times N}$ represents the DQM coefficient matrix for the mth derivative of $\mathbf{F}$ according

to equation (1.4), and $\mathbf{E}^{\mathrm{dif}(m)}$ is the approximation error of the mth-order numerical differentiation of $\mathbf{F}$ given as

$$\mathbf{E}^{\mathrm{dif}(m)}(\xi) = [E^{\mathrm{dif}(m)}(\xi_1), E^{\mathrm{dif}(m)}(\xi_2), \ldots, E^{\mathrm{dif}(m)}(\xi_N)]^T, \quad \mathbf{E}^{\mathrm{dif}(m)} \in \boldsymbol{\mathcal{R}}^{N\times 1} \tag{1.8}$$

Typically, $\mathbf{E}^{\mathrm{dif}(m)} << \mathbf{F}^{(m)}$, so equation (1.8) can be rewritten as

$$\mathbf{F}^{(m)} = \mathbf{D}^{(m)}\mathbf{F} + \mathbf{I}c_0^{(m)} \quad \mathbf{I} = [1, 1, \ldots, 1]^T,\ \mathbf{I} \in \boldsymbol{\mathcal{R}}^{N\times 1}, \tag{1.9}$$

where $c_0^{(m)}$ is the mean approximation error of the mth-order numerical differentiation expressed as

$$c_0^{(m)} = \frac{1}{N}\sum_{i=1}^{N} E^{\mathrm{dif}(m)}(\xi_i). \tag{1.10}$$

2. Inverse differential quadrature method

This section introduces the mathematical formulation for the iDQM. The idea of iDQM involves functional approximation of high-order derivatives of $f(\xi)$ and the subsequent recovering of low-order derivatives by integration of the high-order derivatives. Different approaches to achieve this aim in a computationally efficient and numerically stable manner are presented in the following.

(a) First-order inverse differential quadrature method-by-integration

Suppose we let the first derivative of a function $f(\xi)$, $f^{(1)}(\xi)$, be approximated for a fixed N as in equation (1.2),

$$f^{(1)}(\xi) = \sum_{i=1}^{N} f_i^{(1)} l_i(\xi) + E_1(\xi), \tag{2.1}$$

where $f_i^{(1)}$ is the first derivative of $f(\xi)$ at a discrete point and $E_1(\xi)$ is the polynomial approximation error for $f^{(1)}$. The original function $f(\xi)$ can then be recovered from $f^{(1)}$ by integrating equation (2.1) to get

$$f(\xi) = \int f^{(1)}(\xi)\,\mathrm{d}\xi = \sum_{i=1}^{N} f_i^{(1)} \underbrace{\int l_i(\xi) d\xi}_{H_i(\xi)} + c_0 + E_1^{\mathrm{int}(1)}(\xi), \tag{2.2}$$

where $H_i(\xi)$ is a Nth-order polynomial function, c_0 is the constant of integration and $E_1^{\mathrm{int}(1)}(\xi) = \int E_1(\xi)\,\mathrm{d}\xi$. In a compact form, equation (2.2) gives

$$\mathbf{F} = \mathbf{H}^{(1)}\mathbf{F}^{(1)} + \mathbf{I}c_0 + \mathbf{E}_1^{\mathrm{int}(1)}, \quad \mathbf{F}, \mathbf{F}^{(1)}, \mathbf{E}_1^{\mathrm{int}(1)} \in \boldsymbol{\mathcal{R}}^{N\times 1},\ \mathbf{H}^{(1)} \in \boldsymbol{\mathcal{R}}^{N\times N}, \tag{2.3}$$

where $\mathbf{F}^{(1)} = [f_1^{(1)}, f_2^{(1)}, \ldots, f_N^{(1)}]^T$ and $\mathbf{E}_1^{\mathrm{int}(1)} = [E_1^{\mathrm{int}(1)}(\xi_1), E_1^{\mathrm{int}(1)}(\xi_2), \ldots, E_1^{\mathrm{int}(1)}(\xi_N)]^T$. It is convenient to recast equation (2.3) as:

$$\mathbf{F} = \widetilde{\mathbf{H}}^{(1)}\widetilde{\mathbf{F}}^{(1)} + \mathbf{E}_1^{\mathrm{int}(1)}, \quad \widetilde{\mathbf{F}}^{(1)} \in \boldsymbol{\mathcal{R}}^{(N+1)\times 1},\ \widetilde{\mathbf{H}}^{(1)} \in \boldsymbol{\mathcal{R}}^{N\times(N+1)}, \tag{2.4}$$

where $\widetilde{\mathbf{H}}^{(1)} = [\mathbf{H}^{(1)}\ \mathbf{I}]$ and $\widetilde{\mathbf{F}}^{(1)} = [\mathbf{F}^{(1)}\ c_0]^T$.

Equation (2.4) represents the iDQM relation which is used to approximate a function from its first derivative.

(b) Higher-order inverse differential quadrature method-by-integration

Let the mth-order derivative of a continuous function $f(\xi)$ be $f^{(m)}(\xi)$, which is approximated as in equation (1.2),

$$f^{(m)}(\xi) = \sum_{i=1}^{N} f_i^{(m)}(\xi_i) l_i(\xi) + E_m(\xi), \quad \text{for } m > 1. \tag{2.5}$$

$E_m(\xi)$ is the polynomial approximation error for $f^{(m)}$. To obtain $f^{(m)}(\xi)$ from equation (2.5) requires mth order integration, which leads to

$$f(\xi) = \sum_{i=1}^{N} f_i^{(m)} H^{(m)}(\xi) + f_c^{(m-1)}(\xi) + E_m^{\text{int}(m)}(\xi), \tag{2.6}$$

where $H^{(m)}$ is $(N+m-1)$th order polynomial functions containing mth order integral of $l_i(\xi)$, $E_m^{\text{int}(m)}$ is the mth-order integral of E_m, and $f_c^{(m-1)}$is $(m-1)$th polynomial functions of integration constants given as

$$f_c^{(m-1)} = c_0 + \sum_{k=1}^{m-1} c_k \frac{\xi^k}{k!}. \tag{2.7}$$

Note: the subscript and superscript of $E_m^{\text{int}(m)}$ refers, respectively, to the order of function, i.e. $f^{(m)}$, and the order of integration operation.

In consistency with equations (2.2) and (2.3), equation (2.6) can be recast in a compact form as

$$\mathbf{F} = \widetilde{\mathbf{H}}^{(m)} \widetilde{\mathbf{F}}^{(m)} + \mathbf{E}_m^{\text{int}(m)}, \quad \widetilde{\mathbf{F}}^{(m)} \in \boldsymbol{\mathcal{R}}^{(N+m)\times 1}, \ \widetilde{\mathbf{H}}^{(m)} \in \boldsymbol{\mathcal{R}}^{N\times(N+m)}, \tag{2.8}$$

where $\widetilde{\mathbf{H}}^{(m)} = [\mathbf{H}^{(m)} \ (1/(m-1)!)\boldsymbol{\xi}^{m-1} \ (1/(m-2)!)\boldsymbol{\xi}^{m-2} \cdots \mathbf{I}]$, $\mathbf{H}^{(m)} \in \boldsymbol{\mathcal{R}}^{N\times N}$, $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_N]^T$, $\boldsymbol{\xi} \in \boldsymbol{\mathcal{R}}^{N\times 1}$, $\mathbf{E}_m^{\text{int}(m)} = [E_m^{\text{int}(m)}(\xi_1), E_m^{\text{int}(m)}(\xi_2), \ldots, E_m^{\text{int}(m)}(\xi_N)]^T$, $\mathbf{E}_m^{\text{int}(m)} \in \boldsymbol{\mathcal{R}}^{N\times 1}$ and $\widetilde{\mathbf{F}}^{(m)} = [\mathbf{F}^{(m)} \ c_{m-1} \ c_{m-2} \cdots c_0]^T$, $\mathbf{F}^{(m)} \in \boldsymbol{\mathcal{R}}^{N\times 1}$.

Assuming $\mathbf{E}_m^{\text{int}(m)} << \mathbf{F}$, the error term can be dropped from equation (2.8) to get

$$\mathbf{F} \approx \widetilde{\mathbf{H}}^{(m)} \widetilde{\mathbf{F}}^{(m)}. \tag{2.9}$$

(c) Computational aspects of inverse differential quadrature method-by-integration

The matrix of integral coefficients, $\mathbf{H}^{(m)}$, can be computed by analytical integration, which requires a costly symbolic computational operation in MATLAB. Besides, analytical integration may not be feasible for some vector basis functions, making this approach cumbersome and undesirable. On the other hand, Gaussian integration can be used to compute $\mathbf{H}^{(m)}$ efficiently. However, the numerical accuracy and stability of this operation depends on the number of Gauss points, which is typically chosen intuitively. The additional computational variable manifested by Gaussian integration increases the complexity of the *iDQM-by-integration* especially for high-order functions. On this basis, to compute $\mathbf{H}^{(m)}$, we seek a novel alternative that, respectively, resolves the inefficiency and numerical instability bottlenecks of analytical and Gaussian integrations yet sufficiently preserves the accuracy of both methods.

(d) First-order inverse differential quadrature method-by-inversion

To avoid the deficiency imposed by direct integration or Gaussian integration in the computation of $\mathbf{H}^{(m)}$, we now describe an alternative formulation which is computationally efficient and numerically stable to compute the iDQM coefficient matrix.

Assuming the coefficient matrix, $\mathbf{D}$, is invertible, equation (1.7) can be rearranged to make the vector function, $\mathbf{F}$, the subject, in case of first order as

$$\mathbf{F} = \mathbf{D}^{-1}\mathbf{F}^{(1)} - \left[\mathbf{D}^{(-1)}\mathbf{E}^{\mathrm{dif}(1)}\right], \tag{2.10}$$

or alternatively as in equation (1.9),

$$\mathbf{F} = \overline{\mathbf{D}}^{(1)}\mathbf{F}^{(1)} + \mathbf{I}\bar{c}_0, \quad \overline{\mathbf{D}} \in \mathcal{R}^{N\times N}, \tag{2.11}$$

where $\overline{\mathbf{D}}^{(1)} = \mathbf{D}^{-1}$ and $\overline{c_0}$ is the is the mean error distribution for first-order *iDQM-by-inversion*, which is expressed as

$$\overline{c_0} = \left[-\overline{\mathbf{D}}^{(1)}\mathbf{E}^{\mathrm{dif}(1)}\right]_{\mathrm{mean}}. \tag{2.12}$$

To compare equation (2.11) with equation (2.3), we consider derivation of the constant $\overline{c_0}$ in terms of the integration constant c_0. Assuming that the function $f(\xi)$, evaluated at point p in the closed interval $\xi \in [a\ b]$, $f(\xi_p)$, approaches the constant c_0 due to the fact that the polynomial function $l_i(\xi_p) \to 0$, then

$$f(\xi_p) \to c_0 + E^{\mathrm{int}(1)}(\xi_p) \quad \text{s.t. } l_i(\xi_p) \to 0. \tag{2.13}$$

Now, evaluating $f(\xi_p)$ using equations (2.3) and (2.11), the following relation can be established,

$$c_0 + E^{\mathrm{int}(1)}(\xi_p) = \left[\overline{\mathbf{D}}^{(1)}\right]_p \mathbf{F}^{(1)} + \bar{c}_0, \tag{2.14}$$

where $[\overline{\mathbf{D}}^{(1)}]_p \in \mathcal{R}^{1\times N}$ is the pth row vector of matrix $\overline{\mathbf{D}}^{(1)}$. Rearranging equation (2.14) in terms of c_0 gives

$$\bar{c}_0 = c_0 - \left[\overline{\mathbf{D}}^{(1)}\right]_p \mathbf{F}^{(1)} + E^{\mathrm{int}(1)}(\xi_p). \tag{2.15}$$

Substituting equation (2.15) into equation (2.11) gives

$$\mathbf{F} = \underbrace{\left(\overline{\mathbf{D}}^{(1)} - \mathbf{I}\left[\overline{\mathbf{D}}^{(1)}\right]_p\right)}_{\widehat{\mathbf{D}}^{(1)}} \mathbf{F}^{(1)} + \mathbf{I}c_0 + \mathbf{I}E_1^{\mathrm{int}(1)}(\xi_p), \quad \widehat{\mathbf{D}}^{(1)} \in \mathcal{R}^{N\times N}, \tag{2.16}$$

which is recast as

$$\mathbf{F} = \widetilde{\mathbf{D}}^{(1)}\widetilde{\mathbf{F}}^{(1)} + \mathbf{E}_1^{\mathrm{inv}(1)}, \quad \widetilde{\mathbf{D}}^{(1)} \in \mathcal{R}^{N\times(N+1)}, \tag{2.17}$$

where $\widetilde{\mathbf{D}}^{(1)} = [\widehat{\mathbf{D}}^{(1)}\ \mathbf{I}]$ and $\widetilde{\mathbf{F}}^{(1)} = [\mathbf{F}^{(1)}\ c_0]^T$. $\mathbf{E}_1^{\mathrm{inv}(1)} = \mathbf{I}E_1^{\mathrm{int}(1)}(\xi_p)$.

Equation (2.17) is the equivalent of equation (2.4), which is numerically stable and computationally efficient for the approximation of $f(\xi)$ from its first derivative.

(e) Higher-order inverse differential quadrature method-by-inversion

To obtain higher-order iDQM formulae from the DQM counterpart, we revisit equation (2.17) for the first-order derivative, explicitly expressed as

$$\mathbf{F} = \widehat{\mathbf{D}}^{(1)}\mathbf{F}^{(1)} + \mathbf{I}c_0 + \mathbf{I}E_1^{\mathrm{int}(1)}(\xi_p). \tag{2.18}$$

Analogously, we can write derivations of first and second derivatives of $f(\xi)$ in terms of the second and third derivatives, respectively, as:

$$\left.\begin{aligned}\mathbf{F}^{(1)} &= \widehat{\mathbf{D}}^{(1)}\mathbf{F}^{(2)} + \mathbf{I}c_1 + \mathbf{I}E_2^{\mathrm{int}(1)}(\xi_p)\\ \text{and}\quad \mathbf{F}^{(2)} &= \widehat{\mathbf{D}}^{(1)}\mathbf{F}^{(3)} + \mathbf{I}c_2 + \mathbf{I}E_3^{\mathrm{int}(1)}(\xi_p),\end{aligned}\right\} \tag{2.19}$$

where $E_2^{\text{int}(1)}$ and $E_3^{\text{int}(1)}$ are error estimates implied from equation (2.5) after first-order integration, for $m = 2, 3$, i.e.

$$\left.\begin{aligned} f^{(1)}(\xi) &= \sum_{i=1}^{N} f_i^{(2)} H^{(1)}(\xi) + c_1 + E_2^{\text{int}(1)}(\xi) \\ \text{and} \qquad f^{(2)}(\xi) &= \sum_{i=1}^{N} f_i^{(3)} H^{(1)}(\xi) + c_2 + E_3^{\text{int}(1)}(\xi). \end{aligned}\right\} \tag{2.20}$$

By multiplying the first equation of (2.19) by $\widehat{\mathbf{D}}^{(1)}$ while adding $\mathbf{I}c_0$ and $\mathbf{I}E_1^{\text{inv}(1)}$ leads to

$$\underbrace{\widehat{\mathbf{D}}^{(1)}\mathbf{F}^{(1)} + \mathbf{I}c_0 + \mathbf{I}E_1^{\text{inv}(1)}}_{\mathbf{F}} = \widehat{\mathbf{D}}^{(1)}\widehat{\mathbf{D}}^{(1)}\mathbf{F}^{(2)} + \widehat{\mathbf{D}}^{(1)}\mathbf{I}c_1 + \mathbf{I}c_0 + \widehat{\mathbf{D}}^{(1)}\mathbf{I}E_2^{\text{int}(1)}(\xi_p) + \mathbf{I}E_1^{\text{inv}(1)}, \tag{2.21}$$

or alternatively as

$$\mathbf{F} = \widehat{\mathbf{D}}^{(2)}\mathbf{F}^{(2)} + \boldsymbol{\xi}c_1 + \mathbf{I}c_0 + \underbrace{\left(\boldsymbol{\xi}E_2^{\text{int}(1)}(\xi_p) + \mathbf{I}E_1^{\text{inv}(1)}\right)}_{\mathbf{E}_2^{\text{inv}(2)}}, \quad \widehat{\mathbf{D}}^{(2)} \in \mathcal{R}^{N\times N}, \tag{2.22}$$

where $\widehat{\mathbf{D}}^{(2)} = (\widehat{\mathbf{D}}^{(1)})^2$ and it can be proved that $\boldsymbol{\xi} = \widehat{\mathbf{D}}^{(1)}\mathbf{I}$. $\mathbf{E}_2^{\text{inv}(2)}$ is the error due to second-order *iDQM-by-inversion* operation. Repeating the same operation as in equations (2.21) and (2.22) twice for the second equation of (2.19), and adding the last three terms in equation (2.22), leads to

$$\underbrace{\widehat{\mathbf{D}}^{(2)}\mathbf{F}^{(2)} + \boldsymbol{\xi}c_1 + \mathbf{I}c_0 + \mathbf{E}_2^{\text{inv}(2)}}_{\mathbf{F}} = \widehat{\mathbf{D}}^{(2)}\widehat{\mathbf{D}}^{(1)}\mathbf{F}^{(3)} + \widehat{\mathbf{D}}^{(2)}\mathbf{I}c_2 + \widehat{\mathbf{D}}^{(2)}\mathbf{I}E_3^{\text{int}(1)}(\xi_p) + \boldsymbol{\xi}c_1 + \mathbf{I}c_0 + \mathbf{E}_2^{\text{inv}(2)}, \tag{2.23}$$

or alternatively as

$$\mathbf{F} = \widehat{\mathbf{D}}^{(3)}\mathbf{F}^{(3)} + \frac{1}{2}\boldsymbol{\xi}^2 c_2 + \boldsymbol{\xi}c_1 + \mathbf{I}c_0 + \underbrace{\left(\frac{1}{2}\boldsymbol{\xi}^2 E_3^{\text{int}(1)}(\xi_p) + \mathbf{E}_2^{\text{inv}(2)}\right)}_{\mathbf{E}_3^{\text{inv}(3)}}, \quad \widehat{\mathbf{D}}^{(3)} \in \mathcal{R}^{N\times N}, \tag{2.24}$$

where $\widehat{\mathbf{D}}^{(3)} = (\widehat{\mathbf{D}}^{(1)})^3$ and it can be proved that $(1/2)\boldsymbol{\xi}^2 = \widehat{\mathbf{D}}^{(2)}\mathbf{I}$. $\mathbf{E}_3^{\text{inv}(3)}$ is the error due to third-order *iDQM-by-inversion* operation.

Equations (2.17) and (2.19) represent the second- and third-order equivalent of equation (2.8) from which a vector function $\mathbf{F}$ can be recovered from its higher derivatives. Accordingly, the general expression for the mth-order *iDQM-by-inversion* operation is given as

$$\mathbf{F} = \widehat{\mathbf{D}}^{(m)}\mathbf{F}^{(m)} + \frac{1}{(m-1)!}\boldsymbol{\xi}^{m-1}c_{m-1} + , \ldots, + \boldsymbol{\xi}c_1 + \mathbf{I}c_0 + \underbrace{\left(\frac{1}{(m-1)!}\boldsymbol{\xi}^{m-1}E_m^{\text{int}(1)}(\xi_p) + \mathbf{E}_{m-1}^{\text{inv}(m-1)}\right)}_{\mathbf{E}_m^{\text{inv}(m)}}, \tag{2.25}$$

where $\mathbf{E}_m^{\text{inv}(m)}$ is the error due to mth order *iDQM-by-inversion* operation. The error term in equation (2.25) can be dropped since it is very small compared with $\mathbf{F}$. Thus, equation (2.25) is rewritten as

$$\mathbf{F} \approx \widehat{\mathbf{D}}^{(m)}\mathbf{F}^{(m)} + \frac{1}{(m-1)!}\boldsymbol{\xi}^{m-1}c_{m-1} + , \ldots, + \boldsymbol{\xi}c_1 + \mathbf{I}c_0. \quad \widehat{\mathbf{D}}^{(m)} \in \mathcal{R}^{N\times N}. \tag{2.26}$$

Note: the subscript and superscript of $\mathbf{E}_m^{\text{inv}(m)}$ refers, respectively, to the order of function, i.e. $f^{(m)}$, and the order of inverse operation.

If coordinate transformation is performed for the interval $\xi \in [a\ b]$ to $y \in [0\ L]$ while noting from equation (1.3) that $l_i(0) = 0$, equations (2.9) and (2.26) can be transformed to

$$\mathbf{F} \approx \mathbf{H}^{(m)}\mathbf{F}^{(m)} + \frac{1}{(m-1)!}\mathbf{y}^{m-1}f^{m-1}(0) + , \ldots , + \mathbf{y}f^{(1)}(0) + \mathbf{I}f^{(0)} \tag{2.27}$$

and

$$\mathbf{F} \approx \widehat{\mathbf{D}}^{(m)}\mathbf{F}^{(m)} + \frac{1}{(m-1)!}\mathbf{y}^{m-1}f^{m-1}(0) + , \ldots , + \mathbf{y}f^{(1)}(0) + \mathbf{I}f^{(0)}, \tag{2.28}$$

where $\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$, $\mathbf{y} \in \mathcal{R}^{N\times 1}$.

It should be noted that after transformation, point $\xi_p = y_p = 0$. Subsequent derivations will be presented in the new coordinates, y.

(f) Proof of $\mathbf{y}^m = m!\widehat{\mathbf{D}}^{(m)}\mathbf{I}$

Suppose an mth-order polynomial, $f(y) = y^m$ can be approximated by DQM as

$$f(y) = \sum_{i=1}^{N} f(y_i)l_i(y) + E(f), \tag{2.29}$$

where $l_i(y)$, $f(y_i)$ and $E(f)$ are described according to equations (1.2)–(1.5). Differentiating equation (2.29) once according to DQM approximation leads to the discretized equation,

$$\mathbf{F}^{(1)} = \mathbf{D}^{(1)}\mathbf{F} + \mathbf{E}^{\mathrm{dif}(1)}. \tag{2.30}$$

By evaluating the first derivative of the function y^m analytically, equation (2.30) can be rewritten as

$$m\mathbf{y}^{(m-1)} = \mathbf{D}^{(1)}\mathbf{F} + \mathbf{E}^{\mathrm{dif}(1)}, \tag{2.31}$$

which is further simplified (via inversion) after rearrangement in line with procedures in §d as

$$\mathbf{F} = m\overline{\mathbf{D}}^{(1)}\mathbf{y}^{(m-1)} + \mathbf{I}\bar{c}_0. \tag{2.32}$$

In line with equation (2.13), if a point exists in the closed interval $y \in [0\ L]$ where $f(0)$ equals the constant c_0, then $\bar{c}_0$ can be derived in terms of c_0 as

$$\bar{c}_0 = c_0 - m\left[\overline{\mathbf{D}}^{(1)}\right]_1 \mathbf{y}^{(m-1)}. \tag{2.33}$$

Note that equation (2.33) is exact since integration of $f(y) = y^m$ is exact. Additionally, $p = 1$ since $y_p = 0$ is the first point in the interval $y \in [0\ L]$. Then, equation (2.33) can be substituted into equation (2.32) to give

$$\mathbf{F} = \underbrace{\left(\overline{\mathbf{D}}^{(1)} - \mathbf{I}\left[\overline{\mathbf{D}}^{(1)}\right]_1\right)}_{\widehat{\mathbf{D}}^{(1)}} m\mathbf{y}^{(m-1)} + \mathbf{I}c_0, \tag{2.34}$$

which is now simplified as

$$\mathbf{F} = m\widehat{\mathbf{D}}^{(1)}\mathbf{y}^{(m-1)} + \mathbf{I}c_0, \tag{2.35}$$

where c_0 is the intercept of $f(y)$. Since $f(y) = y^m$, such that the intercept $c_0 = 0$, then equation (2.35) becomes

$$\mathbf{F} = \mathbf{y}^m = m\widehat{\mathbf{D}}^{(1)}\mathbf{y}^{(m-1)}, \quad \text{for } m = 1, 2, \ldots \tag{2.36}$$

By considering that $y = \widehat{\mathbf{D}}^{(1)}\mathbf{I}$ from equation (2.36), we can recast equation (2.36) after successive expansion as

$$\mathbf{y}^m = m!\widehat{\mathbf{D}}^{(m)}\mathbf{I}, \tag{2.37}$$

where $\widehat{\mathbf{D}}^{(m)} = (\widehat{\mathbf{D}}^{(1)})^m$. This completes the proof.

3. Inverse differential quadrature method error

It is important to formulate an iDQM error estimate to describe the performance of iDQM in terms of numerical accuracy and numerical stability. Since *iDQM-by-inversion* is adopted in this work to circumvent computational issues arising from analytical or numerical integration, there is also a need to quantify the error estimate due to DQM inversion in order to assess the approximation quality of the proposed *iDQM-by-inversion*. Moreover, as the DQM coefficient matrix is used to compute iDQM weighting factors, a comparison of discrepancy between DQM and *iDQM-by-inversion* error estimates is necessary to confirm improvements and drawbacks of the proposed method.

(a) Error formulation of inverse differential quadrature method-by-integration

Consider a continuously differentiable function up to mth degree, $f(y)$, where m is very high,

$$f(y) = y^m, \quad \text{for } m > 1. \tag{3.1}$$

Approximation of the first-order derivative of f, $f^{(1)}$, according to equation (2.1) gives

$$f_N^{(1)}(y) \approx \sum_{i=1}^{N} f_i^{(1)} l_i(y), \quad \text{s.t. } N << m. \tag{3.2}$$

To determine the approximation error, a function, $F(z)$, is defined such that

$$F(z) = f^{(1)}(z) - f_N^{(1)}(z) - bM_N(z), \tag{3.3}$$

where $M_N(y)$ is a N degree polynomial defined in equation (1.3), and b is a constant. It is noted that when $z = y_1, y_2, \ldots, y_N$, $F(z) = 0$. So, by setting $F(y) = 0$, the approximation error is estimated as

$$E_1(y) = f^{(1)}(y) - f_N^{(1)}(y) = bM_N(y). \tag{3.4}$$

Considering equation (3.3), it is noted that $F(z)$ has N roots, $y_1, y_2, \ldots, y_N$, in the domain $y \in [0\ L]$. Applying Rolle's theorem [51] repeatedly leads to the Nth-order derivative of $F(z)$, $F^N(z)$, which has at least one root, μ, between y_1 and y_N leading to

$$F^N(\mu) = 0, \tag{3.5}$$

while noting that $f_N^{(1)}(z)$ is a polynomial of $(N-1)$ degree. In this instance,

$$b = \frac{f^{N+1}(\mu)}{N!}. \tag{3.6}$$

Substituting equation (3.6) into equation (3.4) gives

$$E_1(y) = \frac{f^{N+1}(\mu)}{N!} M_N(y). \tag{3.7}$$

In general, μ is a function of y.

It is noted in equation (3.7) that $E_1(y_i) = 0$, since $M_N(y_i) = 0$. In line with equation (3.7), let us denote the maximum error of $f^{(1)}$ approximation for a fixed order of y as $E_{1_{\max}}$, which is expressed as

$$E_{1_{\max}}(y) = \frac{\widehat{K}_1 M_N(y)}{N!}, \tag{3.8}$$

where $\widehat{K}_1 = |f^{N+1}(\mu)|_{\max}$. Now, the maximum error estimate of the original function, $f(y)$, recovered via applying *iDQM-by-integration* to equation (3.7) is obtained thus

$$E_{1_{\max}}^{\text{int}(1)}(y) = \frac{\widehat{K}_1 M_N^{\text{int}(1)}(y)}{N!}, \tag{3.9}$$

where $M_N^{\text{int}(1)}$ and $E_{1_{\max}}^{\text{int}(1)}$ are the first-order integrals of $M_N(y)$ and $E_{1_{\max}}$, respectively. Since $\widehat{K}_1$ and N are constants, it can be stated that the total error arising from numerical integration

is constrained by $M_N^{\mathrm{int}(1)}$. To have a clearer understanding of the implication of equation (3.9), consider a specific case of Chebyshev grid distribution, where y_i are the roots of the Chebyshev polynomial, $T_k(y)$, of order k. In this context, y_i can be expressed as

$$y_i = \cos\theta_i, \quad N\theta_i = i\pi \quad \text{for } i = 0, 1, \ldots, N. \tag{3.10}$$

In connection with equation (3.10), the polynomial, $M_N(y)$, can be expressed in terms of $T_k(y)$ as (see [23])

$$M_N(y) = (1 - y^2)T_N^{(1)}(y), \tag{3.11}$$

where $T_N^{(1)}(y)$ is the first-order derivative of $T_N(y)$. Setting $y = cos\theta$ and $T_N(y) = cosN\theta$ in equation (3.11) leads to,

$$M_N(y) = M_N(\theta) = N\sin\theta\ \sin N\theta. \tag{3.12}$$

The first-order integral of $M_N(y)$ in equation (3.12) is expressed as

$$M_N^{\mathrm{int}(1)}(y) = \begin{cases} 0 & \text{for } N = 0 \\ \dfrac{N\cos N\theta}{2N} - \dfrac{N\cos(N\theta - 2\theta)}{4N - 8} - \dfrac{N\cos(N\theta + 2\theta)}{4N + 8} & \text{for } N \neq 0, 2 \\ -\sin^4\theta & \text{for } N = 2. \end{cases} \tag{3.13}$$

Subject to equation (3.10), equation (3.13) can be recast in terms of y_i at the grid points as:

$$M_N^{\mathrm{int}(1)}(y_i) = \begin{cases} 0 & \text{for } N = 0 \\ N(-1)^i \left(\dfrac{1}{2N} - \dfrac{2y_i^2 - 1}{4N - 8} - \dfrac{2y_i^2 - 1}{4N + 8} \right) & \text{for } N \neq 0, 2 \\ -(1 - y_i^2)^2 & \text{for } N = 2. \end{cases} \tag{3.14}$$

As $M_N^{\mathrm{int}(1)}(y_i)$ varies in N by order $O(N/N)$ in equation (3.14), it can be deduced that for a fixed order of y, the accuracy of iDQM operations is not substantially affected after first-order integration as N increases, since $O(N/N)$ is mutually compensating (i.e. varies directly and inversely equally in order of N).

The error for recovering f from its second-order derivative, $f^{(2)}$, can be obtained by redefining equation (3.3) as

$$F(z) = f^{(2)}(z) - f_N^{(2)}(z) - bM_N(z). \tag{3.15}$$

Then, repeating the procedures of equations (3.4)–(3.8) leads to the maximum error for f approximation from $f^{(2)}$ as

$$E_{2_{\max}}(y) = \frac{\widehat{K}_2 M_N(y)}{N!}, \tag{3.16}$$

where $\widehat{K}_2 = |f^{N+2}(\mu)|_{\max}$. The maximum iDQM error estimate of the original function, $f(y)$, recovered via second-order *iDQM-by-integration* gives

$$E_{2_{\max}}^{\mathrm{int}(2)}(y) = \frac{\widehat{K}_2 M_N^{\mathrm{int}(2)}(y)}{N!}. \tag{3.17}$$

Adopting a Chebyshev grid representation as applied in equations (3.10)–(3.14) results in

$$M_N^{\mathrm{int}(2)}(y_i) = (-1)^i \frac{N}{8} \left[\left(4y_i^3 - 3y_i \right) \left[\frac{1}{(N-3)^2} - \frac{1}{(N+3)^2} \right] + 3y_i \left[\frac{1}{(N+1)^2} - \frac{1}{(N-1)^2} \right] \right]. \tag{3.18}$$

A similar approach can be used to recover the maximum iDQM approximation error of $f(y)$ from its third-order derivative, $f^{(3)}$, as,

$$E_{3_{\max}}^{\mathrm{int}(3)}(y) = \frac{\widehat{K}_3 M_N^{\mathrm{int}(3)}(y)}{N!}, \tag{3.19}$$

where $\widehat{K}_3 = |f^{N+3}(\mu)|_{\max}$, and $M_N^{\text{int}(3)}(y)$ in the context of Chebyshev grid is given by

$$M_N^{\text{int}(3)}(y_i) = (-1)^i \frac{N}{16}\left[-(8y_i^4 - 8y_i^2 + 1)\left[\frac{1}{(N-4)^3} + \frac{1}{(N+4)^3}\right] + 4(2y_i^2 - 1)\left[\frac{1}{(N-2)^3} + \frac{1}{(N+2)^3}\right] - \frac{6}{N^3}\right]. \tag{3.20}$$

Equations (3.18) and (3.20) show that $M_N^{\text{int}(2)}$ varies in N by order $O(N/N^2)$ while $M_N^{\text{int}(3)}$ varies in N by order $O(N/N^3)$. This observation illustrates that, for high-order approximation due to *iDQM-by-integration*, the approximation error is scaled by a function which varies inversely in multiple orders of N. Therefore, subject to a fixed order of y, high-order *iDQM-by-integration* operation is potentially stable numerically as N increases.

(b) Error of inverse differential quadrature method-by-inversion

According to equation (2.17), the approximation error for first-order *iDQM-by-inversion* reads

$$E_1^{\text{inv}(1)} = E_1^{\text{int}(1)}(0). \tag{3.21}$$

Noting equation (3.9), the maximum approximation error for first-order *iDQM-by-inversion* becomes

$$E_{1_{\max}}^{\text{inv}(1)} = \frac{\widehat{K}_1 M_N^{\text{int}(1)}(0)}{N!}, \tag{3.22}$$

where $M_N^{\text{int}(1)}(0)$ given in the context of Chebyshev polynomials reads

$$M_N^{\text{int}(1)}(0) = \begin{cases} 0 & \text{for } N = 0 \\ N(-1)^i\left(\dfrac{1}{2N} + \dfrac{1}{4N-8} + \dfrac{1}{4N+8}\right) & \text{for } N \neq 0, 2 \\ -1 & \text{for } N = 2. \end{cases} \tag{3.23}$$

As in equation (3.14), $M_N^{\text{int}(1)}(0)$ varies in N by order $O(N/N)$ implying that, as N increases, the contribution of $M_N^{\text{int}(1)}(0)$ to the total approximation error in equation (3.22) is expected to be minimal. In the same vein, the second-order approximation error according to *iDQM-by-inversion* is obtained according to equation (2.22),

$$E_2^{\text{inv}(2)}(y) = yE_2^{\text{int}(1)}(0) + E_1^{\text{int}(1)}(0). \tag{3.24}$$

By substituting for $E_2^{\text{int}(1)}(0)$ and $E_1^{\text{int}(1)}(0)$ in equation (3.24) using equations (3.9) and (3.17), the maximum approximation error due to second-order *iDQM-by-inversion* operation gives

$$E_2^{\text{inv}(2)} = y\frac{\widehat{K}_2 M_N^{\text{int}(1)}(0)}{N!} + \frac{\widehat{K}_1 M_N^{\text{int}(1)}(0)}{N!}. \tag{3.25}$$

To establish a relationship between $\widehat{K}_2$ and $\widehat{K}_1$, consider the Nth derivative of $f(y)$ expressed as

$$f^N(y) = \frac{m!}{(m-N)!}y^{m-N}. \tag{3.26}$$

Furthermore, a recursive relation can be established between a derivative and its lower-order derivative as

$$f^{N+1}(y) = \frac{m!}{(m-N-1)!}y^{m-N-1} = \frac{(m-N)}{y}f^N(y) \tag{3.27}$$

and

$$f^{N+2}(y) = \frac{m!}{(m-N-2)!}y^{m-N-2} = \frac{(m-N-2)}{y}f^{N+1}(y). \tag{3.28}$$

Noting this relation, it can be established that

$$f^{N+2}(\mu) = \frac{(m-N-1)}{\mu} f^{N+1}(\mu) \tag{3.29}$$

and

$$\widehat{K}_2 = \frac{(m-N-1)}{\mu(y)} \widehat{K}_1. \tag{3.30}$$

Substituting equation (3.30) into equation (3.25), the approximation of the second-order iDQM inversion gives

$$E_{2_{\max}}^{\mathrm{inv}(2)} = \left(\frac{y(m-N-1)}{\mu(y)} + 1 \right) \frac{\widehat{K}_1 M_N^{\mathrm{int}(1)}(0)}{N!}, \tag{3.31}$$

which is further simplified to

$$E_{2_{\max}}^{\mathrm{inv}(2)} = \left(\frac{y(m-N-1)}{\mu(y)} + 1 \right) E_{1_{\max}}^{\mathrm{inv}(1)}. \tag{3.32}$$

Applying similar procedures as in equations (3.24)–(3.32) leads to the approximation error for the third-order DQM inversion:

$$E_{3_{\max}}^{\mathrm{inv}(3)} = \left(\frac{y^2(m-N-3)(m-N-2)}{\mu^2(y)} + \frac{y(m-N-1)}{\mu(y)} + 1 \right) E_{1_{\max}}^{\mathrm{inv}(1)}. \tag{3.33}$$

According to equations (3.31) and (3.33), high-order approximation by *iDQM-by-inversion* incurs progressive error, which increases successively by order N.

(c) Comparison of inverse differential quadrature method error with differential quadrature method error

As noted in Shu [23], the attributable error as a result of first-order DQM approximation is given by

$$E(y) = \frac{f^N(\mu)}{N!} M_N(y), \tag{3.34}$$

where μ is a function of y. Considering equation (3.8), the maximum error of function approximation by DQM is analogously expressed as

$$E_{\max}(y) = \frac{K_1 M_N(y)}{N!}, \tag{3.35}$$

where $K_1 = |f^N(\mu)|_{\max}$. By differentiating equation (3.35) to obtain the error due to first-order numerical differentiation, we get

$$E_{\max}^{\mathrm{dif}(1)}(y) = \frac{K_1 M_N^{\mathrm{dif}(1)}(y)}{N!}, \tag{3.36}$$

where $M_N^{\mathrm{dif}(1)}(y)$ and $E_{\max}^{\mathrm{dif}(1)}$ are the first derivatives of $M_N(y)$ and $E_{\max}$, respectively. According to equation (3.36), the error arising from first-order DQM differentiation is constrained by

$M_N^{\mathrm{dif}(1)}(y)$, since K_1 and N are constants. Adopting Chebyshev grid representation, considering equations (3.11)–(3.12), the first derivative of $M_N(y)$ is expressed as

$$M_N^{\mathrm{dif}(1)}(y) = -\frac{N \sin N\theta \cos\theta + N^2 \cos N\theta}{\sin\theta}. \tag{3.37}$$

Substituting for y_i at the Chebyshev grid points while noting equation (3.10) results in

$$M_N^{\mathrm{dif}(1)}(y_i) = \begin{cases} (-1)^{i+1} N^2 & \text{for } i \neq 0, N \\ -2N^2 & \text{for } i = 0, N. \end{cases} \tag{3.38}$$

In the same vein, the maximum error due to second-order DQM approximation is given as

$$E_{\max}^{\mathrm{dif}(2)}(y) = \frac{K_1 M_N^{\mathrm{dif}(2)}(y)}{N!}. \tag{3.39}$$

Evaluating $M_N^{\mathrm{dif}(2)}(y)$ in the context of Chebyshev polynomials leads to

$$\begin{aligned} M_N^{\mathrm{dif}(2)}(y) = {} & \frac{2N^2 \cos N\theta \cos\theta - N \sin N\theta \sin\theta - N^3 \sin N\theta \sin\theta}{\sin^2\theta} \\ & - \frac{N \sin N\theta \cos^2\theta - N^2 \cos N\theta \sin\theta \cos\theta}{\sin^3\theta}. \end{aligned} \tag{3.40}$$

Substituting for y_i in equation (3.40) at the Chebyshev grid points using equation (3.10) gives

$$M_N^{\mathrm{dif}(2)}(y_i) = \begin{cases} -\dfrac{2}{3} N^2 (1 + 2N^2) & \text{for } i = 0 \\ (-1)^i N^2 \dfrac{y_i}{1 - y_i^2} & \text{for } i \neq 0, N \\ (-1)^i \dfrac{2}{3} N^2 (1 + 2N^2) & \text{for } i = N. \end{cases} \tag{3.41}$$

According to equation (3.38), first-order DQM approximation is constrained by $M_N^{\mathrm{dif}(1)}$ which varies in N by order $O(N^2)$. For this reason, $M_N^{\mathrm{dif}(1)}$ magnifies the total DQM approximation error subject to first-order numerical differentiation. Second-order DQM approximation gives rise to a total approximation error constrained by $M_N^{\mathrm{dif}(2)}$, which varies in N by order $O(N^4)$ leading to further magnification of the total error as the order of numerical differentiation increases to 2. In general, for a fixed order of numerical differentiation or numerical integration, both DQM and iDQM total errors decrease as N increases as they vary each in N by order $O(1/N^N)$. However, the contributions of $M_N^{\mathrm{dif}(1)}$ and $M_N^{\mathrm{int}(1)}$ to the total errors subject to first-order numerical differentiation and numerical integration, respectively, affect the overall accuracy and numerical stability of the approximations. Comparing *iDQM-by-integration* and DQM error representation in this regard, it can be deduced that, for a fixed order of y, the resulting error from first-order *iDQM-by-integration* is less than the first-order DQM approximation, as the total error magnification on account of the order of $M_N^{\mathrm{dif}(1)}$ and $M_N^{\mathrm{int}(1)}$, respectively, in N is higher for DQM than iDQM approximation. This observation is also true for high-order *iDQM-by-integration* operation since, given a fixed order of y, the contribution of $M_N^{\mathrm{int}(1)}$ to the total error is less than the contribution of $M_N^{\mathrm{dif}(1)}$ to the total high-order differentiation error of DQM operations. Although *iDQM-by-inversion* operation progressively magnifies the approximation error by order N, the rate of increase in the order of N for every successive inversion by iDQM operation is less than the rate of increase in the order of N for every successive differentiation by DQM. Therefore, iDQM operations in general potentially demonstrate superior numerical stability than DQM operations.

4. Numerical results and discussions

In this section, we present some illustrations of the numerical accuracy and numerical stability of the proposed iDQM for functional approximation as well as solution of ordinary differential equations (ODEs) and partial differential equations (PDEs) representing linear and nonlinear systems. We further demonstrate how different schemes of iDQM can be implemented using several examples. The results obtained are then benchmarked with DQM and exact solutions to evaluate the performance of iDQM solutions. To show the robustness of iDQM solutions, numerical analyses based on Lagrange basis polynomials on a non-uniform Chebyshev grid structure are performed and the errors computed. In addition, a comprehensive error analysis to examine the performance of iDQM schemes in terms of convergence and error propagation is performed using a fourth-order boundary value problem (BVP). The results are benchmarked against DQM and exact solutions to evaluate the gains of iDQM approximations.

Note: all examples performed in this work are implemented using *iDQM-by-inversion* since it proves more computationally efficient than *iDQM-by-integration*.

(a) Approximation of function and its higher derivatives

Consider the function,

$$f = 0.02(12 + 3y - 3.5y^2 + 7.2y^3)\,(1 + cos4\pi y)\,(1 + 0.8sin3\pi y), \quad \forall\, y \in [0\ 1]. \tag{4.1}$$

Function f and its derivates up to fourth order are approximated by iDQM schemes of different orders. The results shown in figure 1 prove the accuracy of iDQM approximation as they agree satisfactorily with DQM and exact solutions. The measured relative error, ϵ, between iDQM and DQM estimates and exact solution is computed using the relation,

$$\epsilon = \frac{||\mathbf{f} - \mathbf{f}_{\text{approx}}||_2}{||\mathbf{f}||_2}, \tag{4.2}$$

where $\mathbf{f}$ is a vector function consisting of exact values of f evaluated at each point in the domain while $\mathbf{f}_{\text{approx}}$ represents a vector function of approximate estimates of f at each point in the domain. All error plots presented in subsequent examples are computed using equation (4.2).

According to table 1, iDQM estimates of f in equation (4.1) and its derivatives prove more accurate than DQM estimates. This observation is more evident for second- and third-order iDQM schemes in which the total error is less perturbed by numerical differentiation by DQM (to obtain high-order derivatives) or numerical integration by iDQM (to obtain low-order derivatives). Depending on the order of iDQM, the accuracy of functional approximations may fluctuate between low-order derivative approximation and low-order integral approximation, which is beneficial compared with DQM estimates, which accumulate error subject to successive high-order numerical differentiation. Apart from this, the rate of decline in numerical accuracy for DQM estimates is higher than for iDQM estimates, which is a good indication of numerical stability in favour of iDQM. This aspect is discussed further in §4g.

(b) Solution schemes for systems of differential equations by the inverse differential quadrature method

By approximation of higher derivatives instead of the original function, the proposed iDQM formulation presents a unique opportunity to tune the order of a system, which aids in control of numerical accuracy and numerical stability of the system. In this context, two concepts are proposed in the following subsections.

(i) Mixed inverse differential quadrature method

This scheme involves combination of DQM and iDQM in a manner that ensures application of iDQM for approximation of intermediate derivative(s), which is lower than the highest derivative

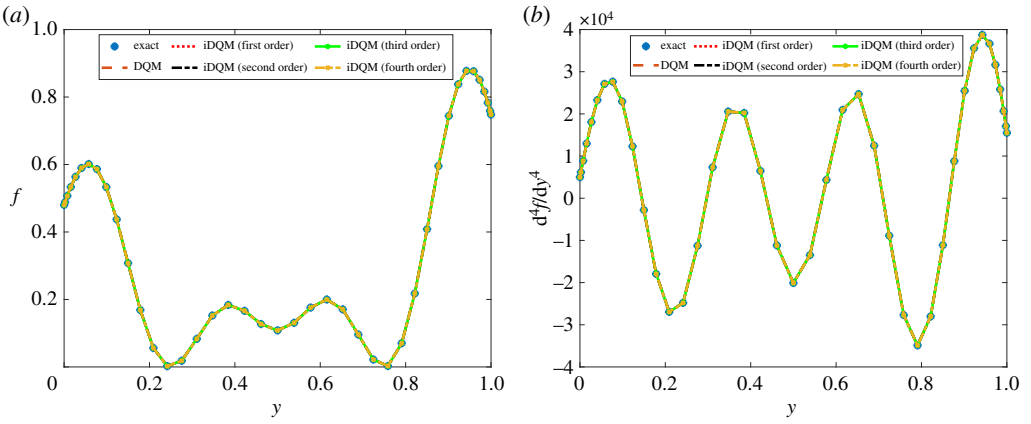

Figure 1. iDQM results implemented for approximation of (*a*) function and (*b*) fourth derivative. (Online version in colour.)

Table 1. Maximum absolute error (relative error) of iDQM and DQM estimates for functional approximation.

	41 points				
	DQM	first-order iDQM	second-order iDQM	third-order iDQM	fourth-order iDQM
f	0	$10^{-15}(10^{-15})$	$10^{-14}(10^{-14})$	$10^{-14}(10^{-14})$	$10^{-12}(10^{-12})$
$f^{(1)}$	$10^{-13}(10^{-15})$	0	$10^{-14}(10^{-14})$	$10^{-13}(10^{-14})$	$10^{-12}(10^{-13})$
$f^{(2)}$	$10^{-11}(10^{-13})$	$10^{-12}(10^{-15})$	0	$10^{-13}(10^{-15})$	$10^{-11}(10^{-13})$
$f^{(3)}$	$10^{-8}(10^{-12})$	$10^{-9}(10^{-14})$	$10^{-11}(10^{-15})$	0	$10^{-11}(10^{-14})$
$f^{(4)}$	$10^{-5}(10^{-11})$	$10^{-7}(10^{-12})$	$10^{-8}(10^{-14})$	$10^{-10}(10^{-15})$	0

in a system, such that lower-order functions can be obtained via iDQM integration while higher-order functions are obtained via DQM differentiation. For example, a fourth-order system of equations can be represented by approximation of the second-order derivative in a MiDQM scheme, in which case the first-order derivative and original function are obtained by numerical integration while third- and fourth-order derivatives are obtained by numerical differentiation. This strategy leads to reduction in the DQM order required for the system solution and, by implication, reduction of the order of the DQM approximation error. This approach is highly promising, and therefore noteworthy, in that it allows tuning of the numerical accuracy of DQM to achieve improved solution. A demonstration of the implementation of this approach is presented for one dimension and two dimensions in appendix A.

(ii) Full inverse differential quadrature method

In contradistinction to its DQM counterpart, FiDQM presents an opportunity to approximate the highest derivative in a system and then apply equation (2.28) to retrieve lower derivatives via iDQM operation. As established in the previous section, given a geometry, the error accrued by integrating a high-order function to get low-order estimates is quite stable numerically compared with differentiating low-order functions to get high-order estimates. As a result, depending on properties such as geometric specifications, boundary conditions or order of a system, high numerical accuracy can be achieved potentially by FiDQM schemes compared with DQM. A demonstration of the implementation of this approach is presented for one dimension and two dimensions in appendix A.

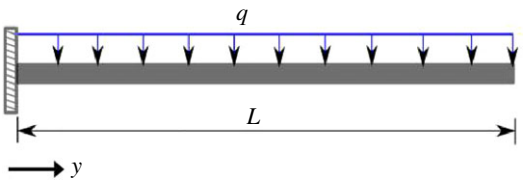

Figure 2. Euler cantilever beam under uniform q load. $L = 10$ m, $b = 0.01$ m, $h = 0.01$ m, $E = 9.05$ GPa and $q = -1$ N. (Online version in colour.)

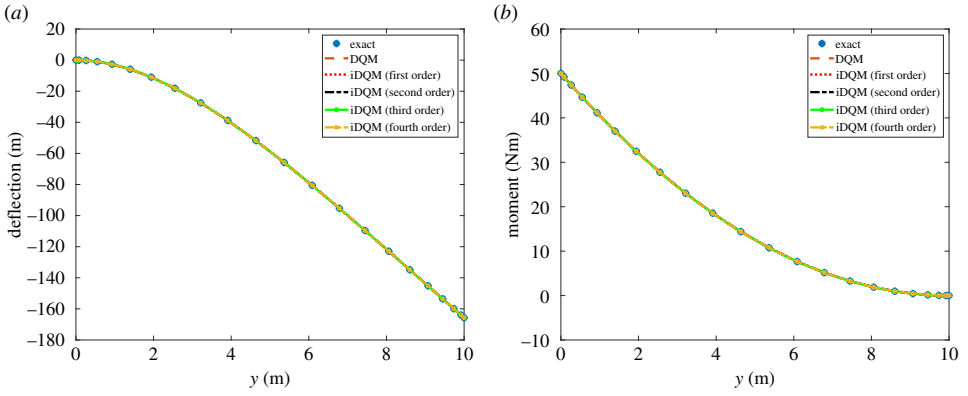

Figure 3. iDQM results for Euler beam (*a*) deflection and (*b*) moment. (Online version in colour.)

(c) Numerical solution of Euler cantilever beam (ODE)

Consider a Euler cantilever beam under a uniformly distributed load, q (figure 2), the governing equation together with the boundary conditions for the beam deflection, w, reads

$$\left.\begin{aligned} &EI\frac{\mathrm{d}^4 w}{\mathrm{d}y^4} = q, \quad \forall y \in [0\ L],\ L = 10\,\mathrm{m} \\ \text{and} \quad &w(y=0) = 0,\ \left.\frac{\mathrm{d}w}{\mathrm{d}y}\right|_{y=0} = 0,\ \left.\frac{\mathrm{d}^2 w}{\mathrm{d}y^2}\right|_{y=L} = 0,\ \left.\frac{\mathrm{d}^3 w}{\mathrm{d}y^3}\right|_{y=L} = 0, \end{aligned}\right\} \tag{4.3}$$

where E is Young's modulus of the beam and I is the second moment of area of the beam's cross-section. According to [52], the exact solution of the beam deflection is given by

$$w(y) = \frac{qy^2(6L^2 - 4Ly + y^2)}{24EI}. \tag{4.4}$$

Given the iDQM discretization scheme described in appendix A, it is noted that the equations arising from iDQM constitute an underdetermined system because the number of unknowns exceed the number of equations. In this context, a pseudoinverse procedure based on truncated singular value decomposition described in [53] is adopted in this work to solve the systems of equations.

According to figure 3, all iDQM estimates of the deflection and moment agree well with exact and DQM solutions showing the accuracy of iDQM solutions. In addition, according to table 2, the relative errors due to iDQM estimates computed for beam deflection, moment and shear force show significant improvement over DQM estimates for the same number of points. This remarkable improvement shows the great potential of iDQM schemes for numerical solution of ordinary differential equations.

Table 2. Maximum absolute error (relative error) of iDQM and DQM estimates for Euler cantilevered beam.

	five points				
	DQM	first-order iDQM	second-order iDQM	third-order iDQM	fourth-order iDQM
deflection	$10^{-12}(10^{-14})$	$10^{-13}(10^{-15})$	$10^{-13}(10^{-16})$	$10^{-13}(10^{-16})$	$10^{-13}(10^{-16})$
moment	$10^{-12}(10^{-14})$	$10^{-13}(10^{-14})$	$10^{-14}(10^{-15})$	$10^{-14}(10^{-15})$	$10^{-14}(10^{-16})$
shear force	$10^{-13}(10^{-14})$	$10^{-14}(10^{-15})$	$10^{-14}(10^{-15})$	$10^{-14}(10^{-15})$	$10^{-14}(10^{-16})$

Table 3. Maximum absolute error (relative error) of iDQM and DQM estimates for temperature and its derivatives.

	30 points		
	DQM	first-order iDQM	second-order iDQM
θ	$10^{-15}(10^{-15})$	$10^{-16}(10^{-16})$	$10^{-15}(10^{-16})$
$\frac{\partial\theta}{\partial\psi}$	$10^{-13}(10^{-14})$	$10^{-15}(10^{-15})$	$10^{-15}(10^{-15})$
$\frac{\partial^2\theta}{\partial\psi^2}$	$10^{-11}(10^{-12})$	$10^{-13}(10^{-13})$	$10^{-14}(10^{-15})$

(d) Nonlinear steady-state solution of heat conduction in slab with temperature-dependent conductivity (ODE)

The problem involves a steady-state heat conduction in a slab with temperature-dependent thermal conductivity in which the non-dimensional form of the temperature (θ) governing equation is expressed as

$$(1+\theta)\frac{\mathrm{d}^2\theta}{\mathrm{d}\psi^2}+\left(\frac{d\theta}{d\psi}\right)^2=0,\quad 0\le\psi\le 1,\ \theta(\psi=0)=0,\ \theta(\psi=1)=1. \tag{4.5}$$

The exact solution of the problem is given in [54] as

$$\theta=-1+\sqrt{1+3\psi}. \tag{4.6}$$

The solution of the nonlinear system, i.e. equation (4.5), is obtained based on Newton–Raphson optimization after iDQM discretization. iDQM results prove accurate with respect to the exact solution and DQM estimates as temperature profile of the slab as well as its first-order derivative match satisfactorily (figure 4). As expected, the error estimates reported in table 3 demonstrate that, although the accuracy of iDQM and DQM estimates of the temperature variable compares equally, iDQM estimates of higher-order derivatives of the temperature variable proves more accurate than DQM estimate, suggesting that error propagation during iDQM operations is less than for DQM operations.

(e) Solution of convection-diffusion equation (PDE)

Consider a steady-state convection-diffusion equation with boundary conditions as follows,

$$\frac{\partial^2u}{\partial x^2}+\frac{\partial^2u}{\partial y^2}-P_e\frac{\partial u}{\partial x}=0,\quad 0\le x,y\le 1,$$
$$u(x,0)=u(x,1)=0,\ 0\le x\le 1,\quad u(0,y)=\sin(\pi y),\ u(1,y)=2\sin(\pi y),\ 0\le y\le 1, \tag{4.7}$$

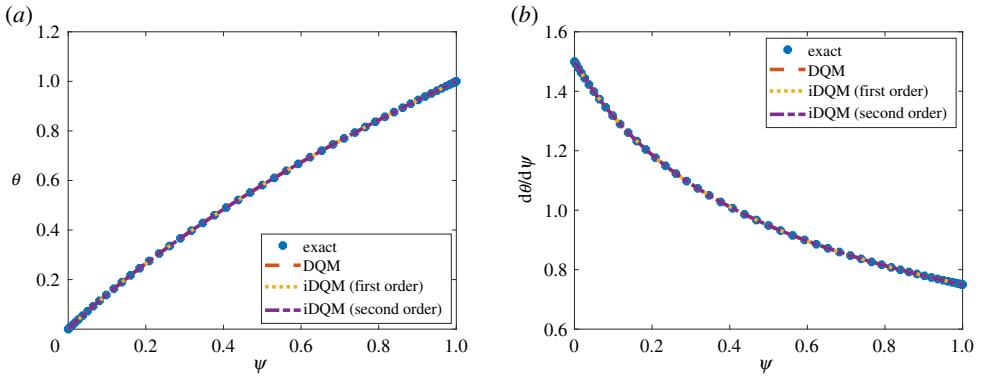

Figure 4. Nonlinear steady-state solution of heat conduction for (*a*) θ and (*b*) $\partial\theta/\partial\psi$. (Online version in colour.)

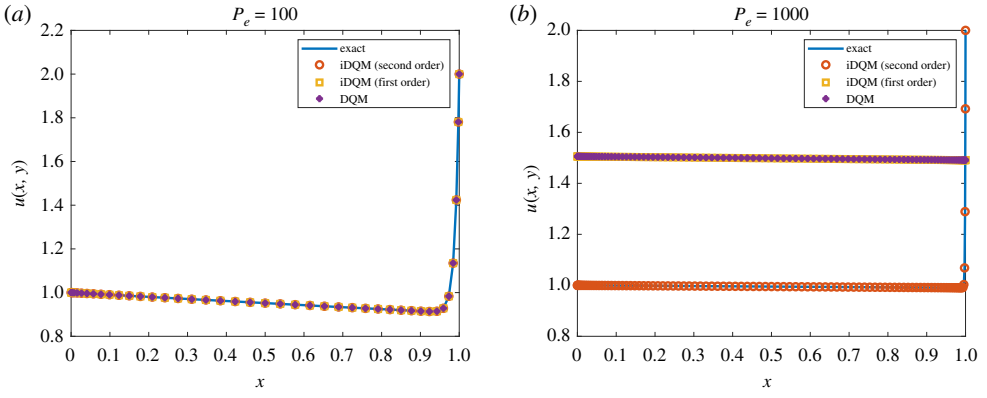

Figure 5. DQM and iDQM $u(x, y)$ estimates at $y = b/2$ for P_e (*a*) 100 and (*b*) 1000. (Online version in colour.)

where P_e is the Peclet number. The exact solution for the given partial differential equation is given in [55] as

$$u(x,y) = \mathrm{e}^{(P_e x/2)}\ \sin(\pi y)\left(\frac{2\mathrm{e}^{((-P_e)/2)}\ \sinh(\sigma x) + \sinh(\sigma(1-x))}{\sinh(\sigma)}\right), \tag{4.8}$$

where $\sigma = \sqrt{\pi^2 + P_e^2/4}$.

According to figure 5, DQM and iDQM values converge to an exact solution of the PDE in equation (4.7) for a 41×41 grid, whereas in figure 5, only second-order iDQM converges to the exact solution for a 101×101 grid. It is quite evident from table 4 that the accuracy of the numerical values has a strong dependence on the Peclet number. For the case of $Pe = 1000$, DQM and first-order iDQM do not converge to the exact solution for the 101×101 grid and further grid refinement fails to improve the solution. According to the findings in [38], an upwind scheme is necessary to obtain accurate solutions for a high Peclet number. Nonetheless, second-order iDQM furnishes an accurate solution for $Pe = 1000$ without an upwind scheme, showing the accuracy of the proposed iDQM. Figure 6 demonstrates the agreement of second-order iDQM with an exact solution over the entire domain.

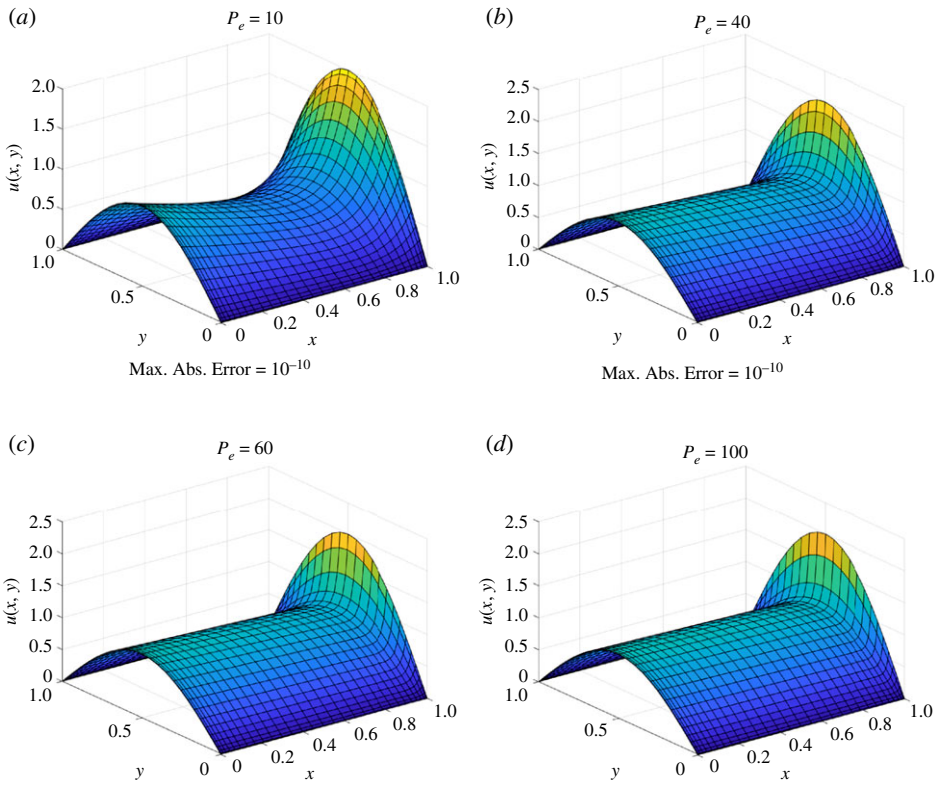

Figure 6. $U(x, y)$ estimate for convection-diffusion equation by iDQM (second order) (31×31 grid). (*a*) Max. Abs. Error$= 10^{-10}$, (*b*) Max. Abs. Error$= 10^{-10}$, (*c*) Max. Abs. Error$= 10^{-7}$ and (*d*) Max. Abs. Error$=10^{-5}$. (Online version in colour.)

Table 4. Maximum absolute error (relative error) of iDQM and DQM $u(x, y)$ estimates.

grid	first-order iDQM	second-order iDQM	DQM
$P_e = 10$			
11×11	$10^{-2}(10^{-3})$	$10^{-5}(10^{-5})$	$10^{-2}(10^{-2})$
15×15	$10^{-5}(10^{-6})$	$10^{-8}(10^{-8})$	$10^{-6}(10^{-6})$
21×21	$10^{-6}(10^{-7})$	$10^{-10}(10^{-11})$	$10^{-11}(10^{-12})$
$P_e = 40$			
11×11	$10^{-1}(10^{-1})$	$10^{-2}(10^{-2})$	$10^{-1}(10^{-1})$
21×21	$10^{-2}(10^{-2})$	$10^{-5}(10^{-6})$	$10^{-2}(10^{-2})$
31×31	$10^{-7}(10^{-7})$	$10^{-10}(10^{-10})$	$10^{-8}(10^{-9})$
$P_e = 100$			
21×21	$10^{-1}(10^{-1})$	$10^{-3}(10^{-3})$	$10^{-1}(10^{-1})$
31×31	$10^{-1}(10^{-1})$	$10^{-5}(10^{-5})$	$10^{-1}(10^{-1})$
41×41	$10^{-5}(10^{-5})$	$10^{-8}(10^{-8})$	$10^{-5}(10^{-5})$
$P_e = 1000$			
101×101	—	$10^{-5}(10^{-5})$	—

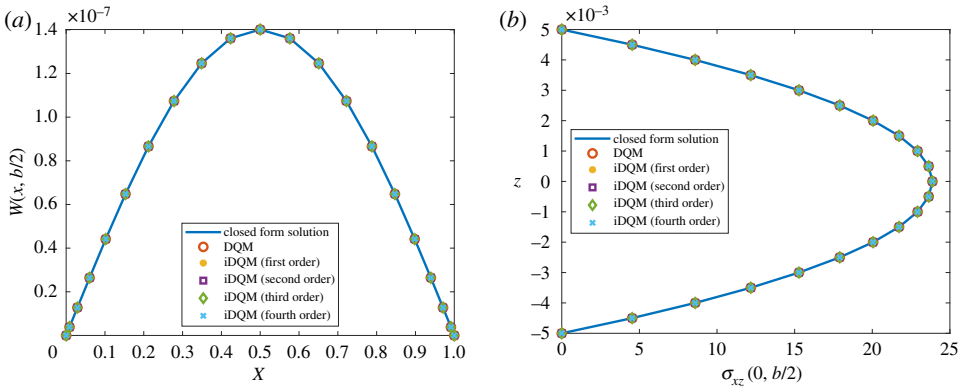

Figure 7. Plot of (*a*) deflection and (*b*) through-the-thickness shear stress, for isotropic plate. (Online version in colour.)

(f) Solution of simply supported isotropic plate under sinusoidally distributed load (PDE)

We consider a thin isotropic plate with dimensions a and b in x and y coordinates, respectively, simply supported on all the edges and under sinusoidally distributed load. The governing differential equation and associated boundary conditions are given as follows,

$$\left.\begin{aligned} &\frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4} = \frac{q}{D}, \quad 0 \le x, y \le a, b \\ &w(x,0) = w(x,a) = w(0,y) = w(b,y) = 0 \\ \text{and} \quad &\frac{\partial^2 w}{\partial x^2}(0,y) = \frac{\partial^2 w}{\partial x^2}(a,y) = \frac{\partial^2 w}{\partial y^2}(x,0) = \frac{\partial^2 w}{\partial y^2}(x,b) = 0, \end{aligned}\right\} \tag{4.9}$$

where w is the transverse deflection of the mid-plane of the plate under loading $q(x,y) = q_0 \sin(\pi x/a)\sin(\pi y/b)$, q_0 is the amplitude of the sinusoidally distributed load, D is the flexural stiffness of the plate given as $D = (Eh^3)/(12(1-\nu^2))$, E is Young's modulus, ν is Poisson's ratio and h is the thickness of the plate. Material and geometric properties of the plate are given by $E = 200\,\text{GPa}$, $\nu = 0.3$, $a = b = 1\,\text{m}$, $h = 0.01\,\text{m}$ and $q_0 = 1\,\text{Pa}$.

The Navier's closed-form solution is simply given by,

$$w(x,y) = \frac{q_0}{\pi^4 D\left(\frac{1}{a^2} + \frac{1}{b^2}\right)^2} \sin\left(\frac{\pi x}{a}\right) \sin\left(\frac{\pi y}{b}\right). \tag{4.10}$$

Solutions of the plate equation obtained by iDQM and DQM are shown in figure 7 where deflections and stresses for the plate match DQM and Navier's solutions demonstrating accuracy of iDQM. Figure 8 shows the deformed planform of the plate under the loading by fourth-order iDQM along with the maximum absolute error in the entire domain. Furthermore, the percentage error of DQM and iDQM estimates in table 5 clearly demonstrates faster convergence for fourth-, third- and second-order iDQM over DQM, highlighting the computational merits of the proposed method.

(g) Error analysis (measure of numerical accuracy)

To appropriately examine the convergence of iDQM, it is important to assess the numerical accuracy of iDQM estimates subject to increased discretization of the domain. In this regard, we

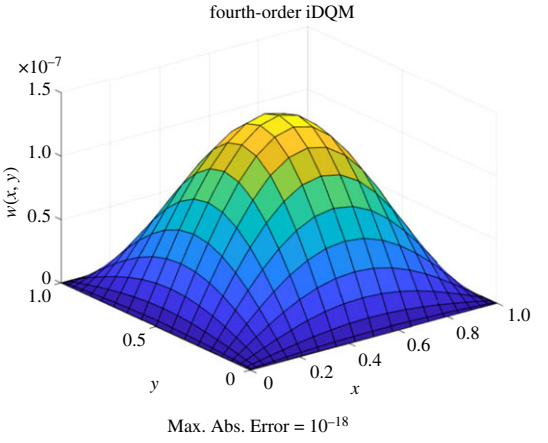

Figure 8. Transverse displacement w of the isotropic plate (21×21 grid). (Online version in colour.)

Table 5. Percentage error of DQM and iDQM estimates for plate solution.

grid	first-order iDQM	second-order iDQM	third-order iDQM	fourth-order iDQM	DQM
$w(a/2, b/2)$					
5×5	47.28	20.77	20.77	1.97	36.90
7×7	15.76	0.01	0.01	0.01	2.24
11×11	0.00	0.00	0.00	0.00	0.00
$\sigma_{xx}(a/2, b/2, h/2)$					
5×5	22.97	9.52	9.52	1.35	25.64
7×7	7.33	0.01	0.01	0.01	1.41
11×11	0.00	0.00	0.00	0.00	0.00
$\sigma_{xz}(0, b/2, 0)$					
5×5	14.30	6.55	6.55	0.04	15.75
7×7	3.99	0.24	0.24	0.01	1.21
11×11	0.14	0.00	0.00	0.00	0.00

consider a BVP with the following set-up,

$$\left.\begin{aligned} &\frac{\mathrm{d}^4U}{\mathrm{d}y^4} - 2\frac{\mathrm{d}^2U}{\mathrm{d}y^2} + U = 0, \quad \forall\, y \in [0\ 4] \\ \text{and} \quad &U(y=0) = 0,\ \left.\frac{\mathrm{d}^2U}{\mathrm{d}y^2}\right|_{y=0} = 2,\ U(y=1) = e,\ \left.\frac{\mathrm{d}^2U}{\mathrm{d}y^2}\right|_{y=1} = 3e. \end{aligned}\right\} \tag{4.11}$$

The exact solution of equation (4.11) is expressed as $U = \cosh y + \cos y$. After solving these equations using different iDQM schemes in accordance with the implementation procedures described in the appendix, the convergence of iDQM and DQM solutions for U and its higher derivatives based on Lagrange polynomial basis are shown in figure 9. According to figure 9, iDQM estimates provide improved convergence over DQM estimates in all cases considered. Clearly, DQM estimates show accumulation of error as the order of the numerical differentiation increases. This observation can be attributed to the perturbation of the total error caused by high-order derivatives of $M_N(y)$, which increasingly affects the accuracy of DQM estimates especially

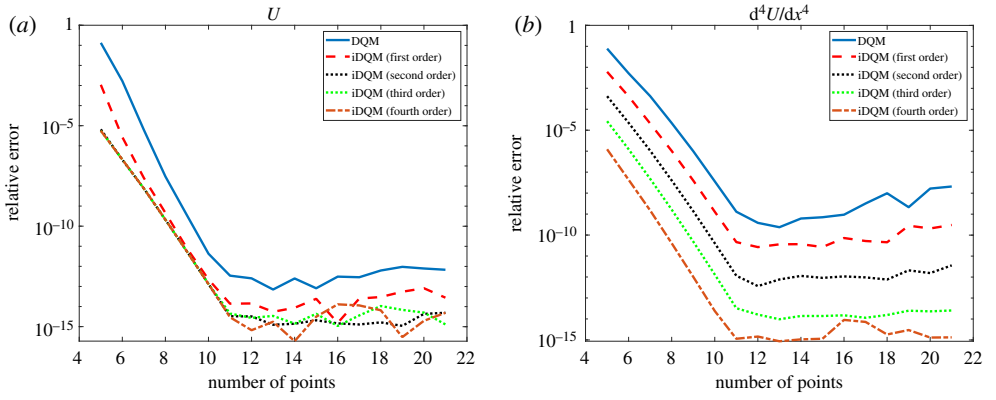

Figure 9. Relative error of BVP solutions for (*a*) U and (*b*) $\partial^4 U/\partial y^4$. (Online version in colour.)

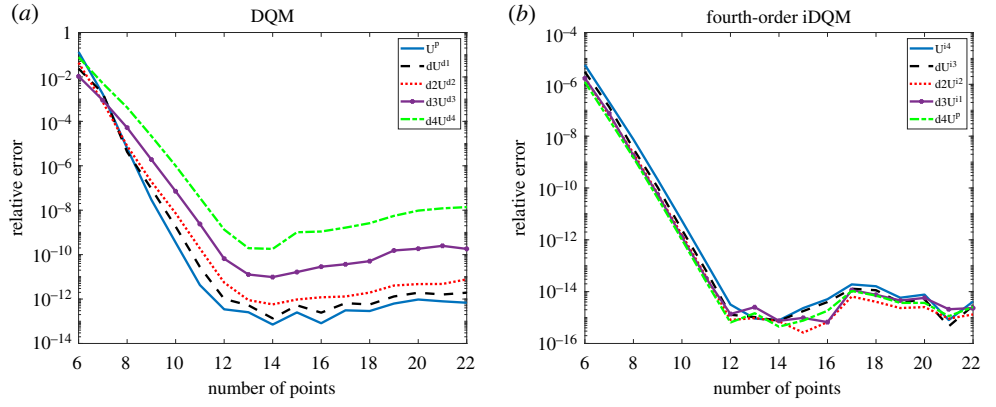

Figure 10. Error propagation of BVP solution based on Lagrange basis polynomial for (*a*) DQM estimate and (*b*) iDQM estimate. (Online version in colour.)

at the boundary, subject to increase in N. On the other hand, iDQM estimates are less perturbed by a high-order integral of $M_N(y)$, which varies inversely to a high order of N (as in *iDQM-by-integration*) or varies with N at a lesser rate than DQM (as in *iDQM-by-inversion*), leading to improved stability of the approximation error.

(h) Error propagation (measure of numerical stability)

As a measure of numerical stability, propagated error arising from approximation of low-order functions from high-order estimates (by using iDQM) or approximation of high-order functions from lower-order estimates (by using DQM) is examined in this section. Figure 10 shows error propagation of the BVP example in §4g, in which high-order functions (labelled superscript d) are computed from primary estimates (labelled superscript p) using DQM operation. On the other hand, low-order functions (labelled superscript i) are computed from primary estimates by *iDQM-by-inversion*.

According to figure 10, the errors resulting from high-order integration by *iDQM-by-inversion* are minimal, indicating numerical stability of iDQM operation. In the case of DQM, errors propagate by multiple orders for successive differentiation operation. While the error accumulation for low-order DQM approximations seems tolerable, the multiple order increment in the error due to high-order differentiation can cause inaccuracy which, in turn, leads to

Table 6. Computational efficiency of iDQM and DQM approximates.

time requirement (nb^2)					
DQM	first-order iDQM	second-order iDQM	third-order iDQM	fourth-order iDQM	ϵ_{max}
example (4g)					
3328	1859	1584	1584	1584	10^{-13}
example (4c)					
320[a]	320	320	245	320	10^{-13}
example (4d)					
12696	17 576	17 576	—	—	10^{-15}
space requirement (nb)					
example (4g)					
208	143	132	132	132	10^{-13}
example (4c)					
40[a]	40	40	35	40	10^{-13}
example (4d)					
552	676	676	—	—	10^{-15}

[a] The DQM estimates converges to 10^{-12}.

numerical instability. As already mentioned in §3c, numerical stability of the DQM solution is significantly affected by the high-order derivative of $M_N(y)$, which increases geometrically as the order of N increases. Therefore, on increasing N, high-order derivatives of $M_N(y)$ quickly offset the total error to reach a lower error bound. However, the lower error bound of low-order estimates from high-order estimates remains stable after *iDQM-by-inversion*, indicating low error propagation and improved numerical stability.

5. Computational efficiency of the inverse differential quadrature method

To measure the computational efficiency of iDQM, we consider a comparison of the time and space memory requirements for the different benchmark problems to converge to a fixed value of the maximum absolute error ϵ_{max} for DQM and iDQM estimates. In this regard, the bandwidth (b) and the total primary degrees of freedom (n) of the final matrix **A** for a given algebraic system **Ax=b** are computed according to table 6.

Some benchmark examples (4c, 4d and 4g) are chosen in table 6 to reflect different types of analysis, boundary conditions and nonlinearities that directly affect the computational complexities of a given numerical problem. According to table 6, iDQM approximation preserves the order of the numerical complexities of the problem in terms of time and space requirements as DQM. It is worth noting that the DQM estimate for example (4c) fails to converge to the threshold absolute maximum error ϵ_{max}, i.e. 10^{-13}. Thus, it is concluded that iDQM approximation preserves the computational efficiency of DQM approximation.

6. Conclusion

This study proposes a novel iDQM for numerical analysis of engineering systems. Given a system of high-order differential equations, the proposed iDQM approximates high-order variables rather than the original function which can be subsequently recovered by integration. To deal with issues bordering on computational inefficiency of analytical integration and numerical

complexity of Gaussian integration, this study develops a novel strategy which relies on inversion of the existing formula of the DQM to compute the required iDQM weighting factors. Furthermore, to evaluate the performance of iDQM solutions, detailed derivations of iDQM error estimates based on integration and DQM inversion are developed in this work, which are then compared with DQM error estimates outlined in [23]. In the context of the iDQM scheme, two implementation approaches identified as Mixed iDQM and Full iDQM are proposed to obtain solutions of the examples provided in this work. Remarkably, the concept of Mixed iDQM provides an excellent opportunity to control the accuracy of system solutions by combining the numerical advantages of low-order differentiation and low-order integration to achieve an improved solution. Subsequently, a demonstration of iDQM implementation for functional approximation, and numerical solutions of systems of high-order ordinary differential equations and partial differential equations representing linear and nonlinear systems, prove that iDQM operations are potentially robust to furnish accurate solutions to numerical systems. Finally, an appraisal of the convergence and numerical stability of the iDQM approach suggests that, compared with DQM, improved convergence can be obtained for systems solution and improved numerical stability is guaranteed by using the proposed method without loss of computational efficiency.

Data accessibility. This article has no additional data.

Authors' contributions. S.O.O. and L.C.T. contributed to the development of iDQM theory and implementation for one-dimensional examples. H.M.K. contributed to iDQM implementation for multidimensional cases. P.M.W. is a contributing author who also supervised the project, and contributed to numerical analysis. All authors gave final approval for publication and agree to be held accountable for the work performed therein.

Competing interests. We declare we have no competing interests.

Funding. The authors acknowledge funding from the Science Foundation Ireland (SFI) and Temporally VARIable COMPosite (VARICOMP) grant no. (15/RP/2773) under its Research Professor programme.

Appendix A

(a) One-dimensional inverse differential quadrature method discretization

Consider a fourth-order ordinary differential equation,

$$\frac{\mathrm{d}^4 w}{\mathrm{d}y^4} = q, \tag{A 1}$$

then the various numerical schemes in discretized form can be realized as follows:

DQM

$$\mathbf{D}^{(4)}\mathbf{w} = \mathbf{q}, \quad \mathbf{w} = [w_1, w_2, \ldots, w_N]^T, \ \mathbf{q} = [q_1, q_2, \ldots, q_N]^T, \quad \mathbf{w}, \mathbf{q} \in \mathcal{R}^{N\times 1} \tag{A 2}$$

First-order iDQM

$$\left.\begin{aligned} &\mathbf{D}^{(3)}\mathbf{w}^{(1)} = \mathbf{q}, \quad \mathbf{w} = \widetilde{\mathbf{D}}^{(1)}\widetilde{\mathbf{w}}^{(1)}, \ \widetilde{\mathbf{w}}^{(1)} = [\mathbf{w}^{(1)} \ w(0)]^T \\ \text{and} \quad &\widetilde{\mathbf{w}}^{(1)} \in \mathcal{R}^{(N+1)\times 1}, \ \widetilde{\mathbf{D}}^{(1)} = [\widehat{\mathbf{D}}^{(1)} \ \mathbf{I}], \ \widetilde{\mathbf{D}}^{(1)} \in \mathcal{R}^{N\times(N+1)}. \end{aligned}\right\} \tag{A 3}$$

Second-order iDQM

$$\left.\begin{aligned} &\mathbf{D}^{(2)}\mathbf{w}^{(2)} = \mathbf{q}, \quad \mathbf{w} = \widetilde{\mathbf{D}}^{(2)}\widetilde{\mathbf{w}}^{(2)}, \quad \widetilde{\mathbf{w}}^{(2)} = [\mathbf{w}^{(2)} \ w^{(1)}(0) \ w(0)]^T \\ \text{and} \quad &\widetilde{\mathbf{w}}^{(2)} \in \mathcal{R}^{(N+2)\times 1}, \quad \widetilde{\mathbf{D}}^{(2)} = [\widehat{\mathbf{D}}^{(1)} \ \mathbf{y} \ \mathbf{I}], \ \widetilde{\mathbf{D}}^{(2)} \in \mathcal{R}^{N\times(N+2)}. \end{aligned}\right\} \tag{A 4}$$

Third-order iDQM

$$\left.\begin{aligned} &\mathbf{D}^{(1)}\mathbf{w}^{(3)} = \mathbf{q}, \quad \mathbf{w} = \widetilde{\mathbf{D}}^{(3)}\widetilde{\mathbf{w}}^{(3)}, \ \widetilde{\mathbf{w}}^{(3)} = [\mathbf{w}^{(3)} \ w^{(2)}(0) \ w^{(1)}(0) \ w(0)]^T \\ \text{and} \quad &\widetilde{\mathbf{w}}^{(3)} \in \mathcal{R}^{(N+3)\times 1}, \quad \widetilde{\mathbf{D}}^{(3)} = \left[\widehat{\mathbf{D}}^{(3)} \ \frac{\mathbf{y}^2}{2} \ \mathbf{y} \ \mathbf{I}\right], \ \widetilde{\mathbf{D}}^{(3)} \in \mathcal{R}^{N\times(N+3)}. \end{aligned}\right\} \tag{A 5}$$

Fourth-order iDQM

$$\left.\begin{aligned} &\mathbf{I}_d\mathbf{w}^{(4)} = \mathbf{q}, \quad \mathbf{w} = \widetilde{\mathbf{D}}^{(4)}\widetilde{\mathbf{w}}^{(4)},\ \widetilde{\mathbf{w}}^{(4)} = [\mathbf{w}^{(4)}\ w^{(3)}(0)\ w^{(2)}(0)\ w^{(1)}(0)\ w(0)]^T, \\ \text{and} \quad &\widetilde{\mathbf{w}}^{(4)} \in \mathcal{R}^{(N+4)\times 1}, \quad \widetilde{\mathbf{D}}^{(4)} = \left[\widehat{\mathbf{D}}^{(3)}\ \frac{\mathbf{y}^3}{6}\ \frac{\mathbf{y}^2}{2}\ \mathbf{y}\ \mathbf{I}\right], \quad \widetilde{\mathbf{D}}^{(4)} \in \mathcal{R}^{N\times(N+4)}. \end{aligned}\right\} \tag{A 6}$$

All coefficient matrices used in this section are as defined in §2. $\mathbf{I}_d \in \mathcal{R}^{N\times N}$ is an identity matrix, N being the number of points chosen in the domain. $\mathbf{D}$, $\widetilde{\mathbf{D}}$ matrix and $\mathbf{I}$ represent DQM and iDQM coefficients defined in §2.

(b) Two-dimensional inverse differential quadrature method discretization

Consider a second-order partial differential equation,

$$\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} = q, \tag{A 7}$$

then the various numerical schemes in discretized form can be realized as follows:

DQM

$$\left.\begin{aligned} &\mathbf{D}_x^{(2)}\mathbf{w}\mathbf{I}_d^T + \mathbf{I}_d\mathbf{w}\mathbf{D}_y^{(2)^T} = \mathbf{q} \\ \text{and} \quad &\mathbf{w} = \begin{bmatrix} w(x_1,y_1) & \cdots & w(x_1,y_n) \\ \vdots & \ddots & \vdots \\ w(x_n,y_1) & \cdots & w(x_n,y_n) \end{bmatrix},\ \mathbf{w} \in \mathcal{R}^{N\times N},\ \mathbf{D}_x = \mathbf{D}_y = \mathbf{D}. \end{aligned}\right\} \tag{A 8}$$

First-order iDQM

$$\left.\begin{aligned} &\widetilde{\mathbf{D}}_x^{(1)}\widetilde{\mathbf{w}}^{(1)}\mathbf{I}_{dy}^{(1)^T} + \mathbf{I}_{dx}^{(1)}\widetilde{\mathbf{w}}^{(1)}\widetilde{\mathbf{D}}_y^{(1)^T} = \mathbf{q}, \quad \widetilde{\mathbf{w}}^{(1)} \in \mathcal{R}^{(N+1)\times(N+1)}, \\ &\widetilde{\mathbf{w}}^{(1)} = \begin{bmatrix} w(0,0) & w_y(0,y_1) & \cdots & w_y(0,y_n) \\ w_x(x_1,0) & w_{xy}(x_1,y_1) & \cdots & w_{xy}(x_1,y_n) \\ \vdots & \vdots & \ddots & \vdots \\ w_x(x_n,0) & w_{xy}(x_n,y_1) & \cdots & w_{xy}(x_n,y_n) \end{bmatrix}, \\ \text{and} \quad &\widetilde{\mathbf{D}}_x^{(1)} = \widetilde{\mathbf{D}}_y^{(1)} = [\mathbf{I}\ \widehat{\mathbf{D}}^{(1)}],\ \mathbf{I}_{dx}^{(1)} = \mathbf{I}_{dy}^{(1)} = [\mathbf{0}\ \mathbf{I}_d] \end{aligned}\right\} \tag{A 9}$$

Second-order iDQM

$$\left.\begin{aligned} &\widetilde{\mathbf{D}}_x^{(2)}\widetilde{\mathbf{w}}^{(2)}\mathbf{I}_{dy}^{(2)^T} + \mathbf{I}_{dx}^{(2)}\widetilde{\mathbf{w}}^{(2)}\widetilde{\mathbf{D}}_y^{(2)^T} = \mathbf{q},\ \widetilde{\mathbf{w}}^{(2)} \in \mathcal{R}^{(N+2)\times(N+2)}, \\ &\tilde{\mathbf{w}}^{(2)} = \begin{bmatrix} w(0,0) & w_y(0,0) & w_{y^2}(0,y_1) & \cdots & w_{y^2}(0,y_n) \\ w_x(0,0) & w_{xy}(0,0) & w_{xy^2}(0,y_1) & \cdots & w_{xy^2}(0,y_n) \\ w_{x^2}(x_1,0) & w_{x^2y}(x_1,0) & w_{x^2y^2}(x_1,y_1) & \cdots & w_{x^2y^2}(x_1,y_n) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ w_{x^2}(x_n,0) & w_{x^2y}(x_n,0) & w_{x^2y^2}(x_n,y_1) & \cdots & w_{x^2y^2}(x_n,y_n) \end{bmatrix}, \\ \text{and} \quad &\widetilde{\mathbf{D}}_x^{(2)} = \widetilde{\mathbf{D}}_y^{(2)} = [\mathbf{I}\ \mathbf{y}\ \widehat{\mathbf{D}}^{(2)}],\ \mathbf{I}_{dx}^{(2)} = \mathbf{I}_{dy}^{(2)} = [\mathbf{0}\ \mathbf{0}\ \mathbf{I}_d] \end{aligned}\right\} \tag{A 10}$$

References

1. Reddy J. 2006 *An introduction to the finite element method*. McGraw-Hill series in mechanical engineering. UK: McGraw-Hill.
2. Logan DL. 2011 *A first course in the finite element method*. USA: Cengage Learning.
3. Cook R, Malkus D, Plesha M. 1989 *Concepts and applications of finite element analysis*. USA: Wiley.

4. Ang W. 2007 *A Beginner's course in boundary element methods*. USA: Universal Publishers.
5. Beer G, Smith I, Duenser C. 2008 *The boundary element method with programming: for engineers and scientists*. Vienna, Austria: Springer.
6. Grossmann C, Roos H, Stynes M. 2007 *Numerical treatment of partial differential equations*. Universitext. Berlin Heidelberg, Germany: Springer.
7. Hoffman J, Frankel S. 2018 *Numerical Methods for Engineers and Scientists*. Boca Raton, FL: CRC Press.
8. Smith G. 1985 *Numerical solution of partial differential equations: finite difference methods*. Oxford applied mathematics and computing science series. Oxford, UK: Clarendon Press.
9. Eymard R, Gallouët T, Herbin R. 2000 Finite volume methods. In *Handbook of numerical Analysis*. vol. 7, pp. 713–1018.
10. Toro EF. 2013 *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Germany: Springer Science & Business Media.
11. Powell M. 1987 Radial basis functions for multivariable interpolation: a review. *Algorithms for the Approximation of Functions and Data* (eds JC Mason, MG Cox), pp. 143–167. Oxford, UK: Clarenden Press.
12. Broomhead D, Lowe D. 1988 Multivariable functional interpolation and adaptive networks. *Complex Syst.* **2**, 321–355.
13. Powell M. 1987 Radial basis functions approximations to polynomials. In *Proc. 12th Biennial Numerical Analysis Conf., 1987*.
14. Dolbow J, Belytschko T. 1998 An introduction to programming the meshless Element F reeGalerkin method. *Arch. Comput. Methods Eng.* **5**, 207–241. (doi:10.1007/BF02897874)
15. Lu Y, Belytschko T, Gu L. 1994 A new implementation of the element free Galerkin method. *Comput. Methods Appl. Mech. Eng.* **113**, 397–414. (doi:10.1016/0045-7825(94)90056-6)
16. Hegen D. 1996 Element-free Galerkin methods in combination with finite element approaches. *Comput. Methods Appl. Mech. Eng.* **135**, 143–166. (doi:10.1016/0045-7825(96)00994-2)
17. Nayroles B, Touzot G, Villon P. 1992 Generalizing the finite element method: diffuse approximation and diffuse elements. *Comput. Mech.* **10**, 307–318. (doi:10.1007/BF00364252)
18. Wright K. 1964 Chebyshev collocation methods for ordinary differential equations. *Comput. J.* **6**, 358–365. (doi:10.1093/comjnl/6.4.358)
19. Dolapçi İT. 2004 Chebyshev collocation method for solving linear differential equations. *Math. Comput. Appl.* **9**, 107–115. (doi:10.3390/mca9010107)
20. Boyd JP. 2001 *Chebyshev and Fourier spectral methods*. USA: Courier Corporation.
21. Bellman N, Roth RS. 2012 *Methods in approximation: techniques for mathematical modelling*, vol. 26. Springer Science & Business Media.
22. Bellman R, Kashef B, Casti J. 1972 Differential quadrature: a technique for the rapid solution of nonlinear partial differential equations. *J. Comput. Phys.* **10**, 40–52. (doi:10.1016/0021-9991(72)90089-7)
23. Shu C. 2012 *Differential quadrature and its application in engineering*. UK: Springer Science & Business Media.
24. Bert CW, Malik M. 1996 Differential quadrature method in computational mechanics: a review. *Appl. Mech. Rev.* **49**, 1–28. (doi:10.1115/1.3101882)
25. Zong Z, Zhang Y. 2009 *Advanced differential quadrature methods*. Boca Raton, FL: CRC Press.
26. Shu C, Richard B. 1992 Parallel simulation of incompressible viscous flows by generalized differential quadrature. *Comput. Syst. Eng.* **3**, 271–281. (doi:10.1016/0956-0521(92)90112-V)
27. Shu C, Richards BE. 1992 Application of generalized differential quadrature to solve two-dimensional incompressible Navier-Stokes equations. *Int. J. Numer. Methods Fluids* **15**, 791–798. (doi:10.1002/fld.1650150704)
28. Mai-Duy N, Tran-Cong T. 2003 Approximation of function and its derivatives using radial basis function networks. *Appl. Math. Model.* **27**, 197–220. (doi:10.1016/S0307-904X(02)00101-4)
29. Kang K, Bert CW. 1997 Flexural–torsional buckling analysis of arches with warping using DQM. *Eng. Struct.* **19**, 247–254. (doi:10.1016/S0141-0296(96)00057-0)
30. Striz AG, Weilong C, Bert CW. 1994 Static analysis of structures by the quadrature element method (QEM). *Int. J. Solids Struct.* **31**, 2807–2818. (doi:10.1016/0020-7683(94)90070-1)
31. Ojo S, Patni M, Weaver P. 2019 Comparison of weak and strong formulations for 3D stress predictions of composite beam structures. *Int. J. Solids Struct.* **178**, 145–166. (doi:10.1016/j.ijsolstr.2019.06.016)
32. Ojo S, Weaver P. 2019 3D static analysis of patched composite laminates using a multidomain differential quadrature method. *Compos. Struct.* **229**, 111389. (doi:10.1016/j.compstruct.2019.111389)

33. Tornabene F, Fantuzzi N, Bacciocchi M. 2018 Strong and weak formulations based on differential and integral quadrature methods for the free vibration analysis of composite plates and shells: convergence and accuracy. *Eng. Anal. Bound. Elem.* **92**, 3–37. (doi:10.1016/j.enganabound.2017.08.020)
34. Kang K, Bert C, Striz A. 1995 Vibration analysis of shear deformable circular arches by the differential quadrature method. *J. Sound Vib.* **183**, 353–360. (doi:10.1006/jsvi.1995.0258)
35. Striz A, Chen W, Bert C. 1997 Free vibration of plates by the high accuracy quadrature element method. *J. Sound Vib.* **202**, 689–702. (doi:10.1006/jsvi.1996.0846)
36. Wang X, Bert C, Striz A. 1993 Differential quadrature analysis of deflection, buckling, and free vibration of beams and rectangular plates. *Comput. Struct.* **48**, 473–479. (doi:10.1016/0045-7949(93)90324-7)
37. Ojo SO, Weaver P. 2021 A generalized nonlinear strong Unified Formulation for large deflection analysis of composite beam structures. In *AIAA Scitech 2021 Forum*. p. 0698.
38. Chan Y, Shen L, Wu C, Young D. 2014 A novel upwind-based local radial basis function differential quadrature method for convection-dominated flows. *Comput. Fluids* **89**, 157–166. (doi:10.1016/j.compfluid.2013.10.032)
39. Shen L, Young D, Lo D, Sun C. 2009 Local differential quadrature method for 2-D flow and forced-convection problems in irregular domains. *Numer. Heat Transfer B: Fundam.* **55**, 116–134. (doi:10.1080/10407790802605430)
40. Lo D, Young D, Tsai C. 2007 High resolution of 2D natural convection in a cavity by the DQ method. *J. Comput. Appl. Math.* **203**, 219–236. (doi:10.1016/j.cam.2006.03.021)
41. Shu C, Chew Y, Khoo B, Yeo K. 1994 Global method for solving incompressible Navier-Stokes equations in the general coordinate system.
42. Lo D, Young D, Murugesan K. 2006 An accurate numerical solution algorithm for 3D velocity–vorticity Navier–Stokes equations by the DQ method. *Commun. Numer. Methods Eng.* **22**, 235–250. (doi:10.1002/cnm.817)
43. Malekzadeh P, Rahideh H, Karami G. 2006 A differential quadrature element method for nonlinear transient heat transfer analysis of extended surfaces. *Numer. Heat Transfer A Appl.* **49**, 511–523. (doi:10.1080/10407780500436840)
44. Fidanoglu M, Kömürgöz G, Özkol I. 2014 Application of Differential Quadrature Method (DQM) to Heat Transfer and Entropy Generation Problems. In *Advanced Materials Research* vol. 1016, pp. 769–773. Trans Tech Publ.
45. Wang T, Shen L, Young D, Chen C. 2018 Local iRBF-DQ method for MHD duct flows at high Hartmann numbers. *MOJ Appl. Bio. Biomech.* **2**, 00047. (doi:10.15406/mojabb.2018.02.00047)
46. Shen L, Tseng K, Young D. 2013 Evaluation of multi-order derivatives by local radial basis function differential quadrature method. *J. Mech.* **29**, 67–78. (doi:10.1017/jmech.2012.121)
47. Xionghua W, Zhihong D. 2002 Differential quadrature method for pricing American options. *Numer. Methods Partial Dif. Equ. Int. J.* **18**, 711–725. (doi:10.1002/num.10028)
48. Quan J, Chang C. 1989 New insights in solving distributed system equations by the quadrature method–I. Analysis. *Comput. Chem. Eng.* **13**, 779–788. (doi:10.1016/0098-1354(89)85051-3)
49. Wang X. 2015 *Differential quadrature and differential quadrature based element methods: theory and applications*. USA: Butterworth-Heinemann.
50. Wu X, Ren Ye. 2007 Differential quadrature method based on the highest derivative and its applications. *J. Comput. Appl. Math.* **205**, 239–250. (doi:10.1016/j.cam.2006.04.055)
51. Besenyei A. 2012 A brief history of the mean value theorem, *Talk slides*, September12.
52. Wikipedia contributors. 2020 Eulerbeam theory — Wikipedia, The Free Encyclopedia. [Online; accessed 30 September 2020].
53. Seiler MC, Seiler FA. 1989 Numerical recipes in C: the art of scientific computing. *Risk Anal.* **9**, 415–416. (doi:10.1111/j.1539-6924.1989.tb01007.x)
54. Finlayson B. 2003 *Nonlinear analysis in chemical engineering*. USA: Ravenna Park Publishing.
55. Mai-Duy N, Dalal D, Le TTV, Ngo-Cong D, Tran-Cong T. 2018 A symmetric integrated radial basis function method for solving differential equations. *Numer. Methods Partial Differ. Equ.* **34**, 959–981. (doi:10.1002/num.22240)