Republic of Iraq
Ministry of Higher Education
and Scientific Research
Al-Nahrain University
College of Science
Department of Mathematics
and Computer Applications

Numerical Solutions of Differential Equations Via G-Spline Based Differential Quadrature Method

A Thesis

Submitted to the College of Science of Al-Nahrain University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Mathematics

By

Mustafa Akram Saeed

(B.Sc. Math. / College of Science / Al-Nahrain University, 2010)

Supervised by

Dr. Osama Hameed Mohammed ***(Asst. Prof.)***

Dr. Fadhel Subhi Fadhel ***(Asst. Prof.)***

December 2012

Safar 1434

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

﴿ ويسئلونك عَنِ الرُّوحِ قُلِ الرُّوحُ من أمرِ ربِي

وَمَا أُوتيتُم مِنَ العِلمِ إِلَّا قليلاً ﴾

صَدَقَ اللهُ العَظِيم

(الإسراء / ٨٥)

Dedication

To …

My Family with Love and Respects

Acknowledgments

My heart and my soul say thanks to Allah, for this generous by making me complete this work.

I would like to express my deep thanks and my respectful gratitude to my highly distinguished guids Dr. Osama Hameed Mohammed and Dr. Fadhel Subhi Fadhel for their supervisions, continuous guidance, encouragements and their helpful suggestions throughout all preparation of this research.

Also, I'm deeply indebted to the staff member of Department of Mathematics and computer applications and to the College of Science, Al-Nahrain University for giving me the chance to complete my study.

Finally, my thanks are extended to the dearest ones in my life my family and friends who gave me the help, support, encouragement and patience.

Mustafa, 2012

Contents

Abstract

This thesis have two main objectives, namely:

1- The first objective is to study the mathematical background of the differential quadrature method and its application to solve boundary value problems of the fourth order ordinary differential equations.

2- The second objective is first about function approximation by G-spline interpolation method. Secondly the numerical solution of two applications relating the vibration of a uniform beam problem which are represented by a boundary value problem of the fourth order ordinary differential equation and the vibration of a square thin plate given by a boundary value problem of the forth order partial differential equation, by using G-spline based differential quadrature method have been obtained.

Introduction

The Differential Quadrature Method (DQM) is a numerical method for evaluating derivatives of a sufficiently smooth function, proposed by Bellman and Casti [1]. The basic idea of DQM comes from Gauss Quadrature method, which can be considered as one of the most simple and accurate methods for calculating the integral numerically. Gauss Quadrature is characterized by approximating a definite integral with a weighting sum of integrand values at a group of so called Gauss points.

Extension is made for finding the derivatives of various orders of a sufficiently smooth function give a rise to DQ in [1], [11]. In the other words, the derivatives of a smooth function are approximated with weighting sum of function values at a group of so called nodes [38].

The key procedure in the DQM lies in the determination of the weighting coefficients. Initially, Bellman and his associates proposed two methods to compute the weighting coefficients for the first order derivative. The first method is based on an ill-conditioned algebraic system of equations. The second method uses a simple algebraic formulation, but the coordinates of the grid points are fixed by the roots of the shifted Legendre polynomial [28]. In earlier applications of the DQM, Bellman's first method was usually used because it allows the case of an arbitrary grid point distribution. However, since the algebraic system of equations related to this method is ill-conditioned, the number of the grid points usually used will be less than 13; this drawback limits the application of the DQM [28].

The DQM and its applications were rapidly developed after the late of 1980, ideas to the innovative work in the computation of the weighting coefficients by many authors see [13], [22], [25], [26] and [27].

As a result, the DQM has emerged as a powerful numerical discretization tool for solving several problems in ordinary and partial differential equations in the past two decades.

Applying boundary conditions is one of the major task in the DQM since a differential equation is underdetermined if boundary conditions are not provided. For a one-dimensional second order partial differential equation, for example, boundary conditions are required at both ends. For such cases, direct DQM can be applied without difficulty. What should be done is to let the function values or the first derivatives at both ends to equal the prescribed values; see [28]. Difficulty, however, arises in applying multi-boundary conditions, for example, to solve forth-order differential equations where two boundary conditions are presented at each end.

Bert and his coworkers [5], [6] and [7], introduced a delta-point a part from the boundary point by a small distance as an additional boundary point and apply the other boundary condition at that point. It is found that, however, the solution accuracy may not be assured. Moreover, there are some difficulties in applying the multi-boundary conditions accurately at the corner points for two-dimensional problems, since singularity may arise.

There are several approaches available in literatures for implementing multi-boundary conditions. Some DQ equations at the inner nodes are replaced by the additional boundary conditions. It is found that; however, the solution accuracy may vary depending on which DQ equations at inner grids are replaced by the boundary conditions. A complete different approach introduced by Wang and Bert [32] is that the boundary conditions are built during the formulation of the weighting coefficients for higher order derivatives. Later, Malik and Bert [16] tried to extend this idea to all boundary conditions. Wang et al. [33] and Wang

et al [34] proposed a method which is called the Differential Quadrature Element Method, or DQEM to applying the multi-boundary conditions the assigning two degrees of freedom to each end point for a fourth-order differential equation. Later Wu and his coworkers [36] and [37] proposed a Generalized Differential Quadrature Rule (GDQR) also introducing multiple degrees of freedom at the boundary points. All boundary conditions may be easily applied by the DQEM or GDQR and accurate solutions can be obtained. Actually, both methods are exactly the same for one-dimensional problems.

Karami and Malekzadeh [12] proposed a method for applying the multi-boundary conditions, in the formulations of the weighting coefficients of third-order and fourth-order derivatives, the second derivatives at the boundary points are viewed as an additional independent variable. Wang and Bert [32] extended the method to all boundary conditions by use the first derivatives at the boundary points as an additional independent variable. Both methods introduces an additional degree of freedom at the boundary points, will computing the weighting coefficients of the first-order derivative by using explicit formula [25], [35].

As compared to the conventional low order finite difference and finite element methods, the DQM can give very accurate numerical results using a considerably smaller number of grid points and hence requiring relatively little computational efforts. So far, the DQM has been efficiently employed in a variety of problems in engineering and physical science problems. A comprehensive review of the DQM has been given by Bert and Malik [4]. DQM may be formulated either through approximation theory or solving system of linear equations.

In this thesis we will employ functions approximation theory using G-spline interpolation to formulate the DQM. Nearly 66 years ago, I.J.

Schoenberg [23] introduced the subject of "spline function" since then splines, have proved to be enormously important in various branches of mathematics, such as approximation theory, numerical analysis, numerical treatment of ordinary, partial differential equations, integral equations statistics, etc. There are several types of spline functions appeared in literatures given by [10], [19] and Stephen [30]. Among these types of spline functions is the so called G-spline interpolation which is necessary to the work of this thesis. In 1968 Schoenberg [23] extended the idea of Hermite for splines to specify that the order of derivatives specified may vary from node to node.

Schoenberg used the term "G-spline" instead of generalized splines because the natural spline term "generalized spline" describes an extension in a different direction [17]. The G-spline is used to interpolate the Hermite Birkhoff-data (problem), the data in this problem are the values of the function and its derivatives but without Hermite's condition that the only consecutives be used at each node. Further, Schoenberge [23] define the G-spline function as a smooth piecewise polynomials, where the smoothness is governed by the incidence matrix, and then proved that G-splines, satisfies what is so called the "minimum norm property", which is used for the optimality of the G-spline function defined mathematically by the following inequality:

$$\int_I [f^{(m)}(x)]^2 dx > \int_I [S^{(m)}(x)]^2 dx$$

This thesis consists of three Chapters:

In Chapter one, the mathematical background of differential quadrature method was given, which cover the following subjects: fundamental concepts of the approximation theory, differential quadrature method, direct differential quadrature method, differential quadrature

analysis of Beams, implementation of boundary conditions and finally a numerical example regarding free vibration analysis of a uniform beam.

In Chapter two which is entitled G-spline-based differential quadrature and its application to a uniform beam problem treat the following subjects: G-spline interpolation, the G-spline interpolation - based differential quadrature method, applying boundary conditions, illustrative example regarding free vibration analysis of a uniform beam using G-spline interpolation-based differential quadrature.

Finally the numerical solution of a thin plates using G-spline-based differential quadrature method which deals with the following concepts: differential quadrature analysis of thin plates, direct substitution of boundary conditions into discrete governing equation, and numerical example illustrating the numerical solution of free vibration analysis of square plates are given in chapter three.

Finally, it is remarkable that all calculations are performed using MATLAB computer package and the computer programs with the G-spline fundamental functions are given in appendices A and B respectively.

Chapter One

The Mathematical Background of Differential Quadrature Method

1.1 Introduction

In this chapter we shall present an introduction to the theory functions approximation and then following by the fundamentals of the differential quadrature method in order to make this thesis of self contained as soon as possible.

Therefore this chapter will consists of five sections, in section 1.2, fundamental concepts of approximation theory was given. In section 1.3, the DQM will be presented while section 1.4 deals with the direct DQM. Finally, the application of DQM to vibration analysis of beams will be given in section 1.5.

1.2 Fundamental Concepts of Approximation Theory:

Suppose C [a, b] is a set of all continuous functions defined on the interval [a, b]. Define a real valued function $f \in$ C [a, b] which has a form to be either complicated or hard to explicitly write.

A general approximation is to use another simpler function $A(f)$ to replace f, given that $A(f)$ is very close to f. Approximation theory is about determining $A(f)$ and how well it works as a replacement of f. To answer this question, we begin with the vector space.

Definition (1.1),[31]:

A vector space (over $\mathbb{R}$) is a set V with two operations "+" and "." satisfying the following properties for all u, v $\in$ V and c, d $\in \mathbb{R}$:

(1) (Additive Closure) u + v $\in$ V.

(2) (Additive Commutativity) u + v = v + u.

(3) (Additive Associativity) (u + v) + w = u + (v + w).

(4) (Zero) There is a special vector 0 $\in$ V such that u + 0 = u for all u in V.

(5) (Additive Inverse) for every u $\in$ V there exists w $\in$ V such that u + w = 0.

(6) (Multiplicative Closure) c . v $\in$ V.

(7) (Distributivity) (c + d).v = c.v + d.v and c.(d + v) = c.d + c.v.

(8) (Associativity) (c . d).v = c. (d . v).

(9) (Unity) 1 . v = v for all v $\in$ V.

Definition (1.2), [31]:

A normed linear space (denoted by V) is a vector space with the additional structure of a norm.

Definition (1.3), [38]:

The norm is a function $\|\ \|$ form V to $\mathbb{R}$, where $\mathbb{R}$ is the set of all real numbers, with the following properties, for each x, y$\in \mathbb{R}$ and each scalar α:

1- $\|x\| \geq 0$.

2- $\|x\| = 0$ if and only if x = 0.

3- $\| \alpha x \| \leq | \alpha | \|x\|$.

4- $\|x + y\| \leq \| x\| + \| y\|$.

Let V be a vector space, let $\phi_1, \phi_2, \cdots, \phi_n$ be vectors in V, and $a_1, a_2, ..., a_n$ be scalars in R. The vector $a_1\phi_1 + a_2\phi_2 + \cdots + a_n\phi_n$ is called a linear combination of $\phi_1, \phi_2, \cdots, \phi_n$ with combination coefficients $a_1, a_2, ..., a_n$ .The vector $a_1\phi_1 + a_2\phi_2 + \cdots + a_n\phi_n$ is the zero vector if and only if $a_1, a_2, ..., a_n$ are all zero. If all vectors in V can be written as linear combinations of $\phi_1, \phi_2, \cdots, \phi_n$ then these vectors form a set called the span of $\phi_1, \phi_2, \cdots, \phi_n$, or in other words say that $\phi_1, \phi_2, \cdots, \phi_n$ span the vector space V. Linear independence and span are two key elements of the so-called basis.

Example (1.1), (polynomial vector space), [38]:

Let Π_n be the set of polynomials of degree less than or equal to n in x with real coefficients. Then it is a vector space. The dimension is n+1. If

$p \in \Pi_n$, then $p = a_0 + a_1 x + \cdots + a_n x^n$. We claim that $1, x, \cdots, x^n$ are the basis for Π_n. To prove this, we need to show that, $1, x, \cdots, x^n$

1- Are linearly independent.

2- They span Π_n

Property 2 is clear from the definition of Π_n . property 1 is direct conclusion from the assumption that $a_0, a_1, \cdots, a_n$ are scalars satisfying the condition $a_0 + a_1 x + \cdots + a_n x^n = 0$.

The following concepts at least point out the possibility of how to construct an approximant based on linear combination.

Theorem (1.1) (Best Approximation), [38]:

Let V be a normed linear space with norm $\| \|$, let $\phi_1, \phi_2, \cdots, \phi_n$ be any linearly independent members of V. Let y be any member of V. Then there are coefficients $a_0, a_1, \cdots, a_n$ which solve the problem of minimizing

$$\left\| y - \sum_{i=1}^{n} a_i \phi_i \right\| \tag{1.1}$$

where y is any member of V.

A solution to the minimizing problem (1.1) is usually called " best approximant " to y from V.

Now, a question may arise. How many solutions are there? The answer is connected with the convexity conditions.

Definition(1.4), [38]:

Let V be a vector space. A subset S of V is convex if for any two members ϕ_1, ϕ_2 of S the set of all members of the line segment $\phi = \lambda \phi_1 + (1 - \lambda)\phi_2$, $0 \le \lambda \le 1$, also belongs to S.

Theorem (1.2) (Uniqueness), [38]

Let V be a normed linear space with strictly convex norm. Then the best approximations problem from finite dimensional subspace are unique. Approximation is transformed into another problem of searching for the basis which defines a normed linear space with strictly convex norm. Depending on the sense in which the approximation is realized, or depending on the norm definition (1.3), there are three types of approximation approaches:

1- Interpolator approximation:

The parameters a_i, for all $i = 1,2,...,n$ are chosen so that on a fixed prescribed set of points x_k, $k = 1,2,\cdots,n$ we have:

$$\sum_{i=1}^{n} a_i \phi_i(x_k) = y_k, \; k = 1,2,...,n \tag{1.2}$$

Sometimes, it is further required that for each i, the first r_i derivatives of the approximant agree with those of y at x_k.

2- Least-square approximation: a_i are chosen so as to minimize:

$$\left\| y - \sum_{i=1}^{n} a_i \phi_i \right\| := \sum_{k=1}^{n} (y_k - \sum_{i=1}^{n} a_i \phi_i(x_k))^2 \tag{1.3}$$

3- Min-Max approximation: the parameters a_i are chosen so as to minimize

$$\left\| y - \sum_{i=1}^{n} a_i \phi_i \right\|_\infty := \max \left| y - \sum_{i=1}^{n} a_i \phi_i \right| \tag{1.4}$$

This thesis will depend on interpolation in the construction of differential quadrature.

1.2.1 Interpolating Bases:

The most employed bases in the theory of interpolation are polynomial basis which are to be detailed in the following. This is the approximants often used in DQM as well.

1.2.1.1 Polynomial Basis:

Choosing $\phi_i = x^i$, $\forall i = 1,2,...,N$ we have polynomials as approximants. Weierstrass theorem guarantees that this is at least theoretically feasible choice.

Theorem (1.3) (Weierstrass Approximation Theorem), [38]:

Let f be a continuous function on the interval [a, b]. Then for any ε > 0, there exist a positive integer n and a polynomial p_n, such that:

$$\max_{x \in [a,b]} \left| f(x) - p_n(x) \right| < \varepsilon \tag{1.5}$$

The proof of Weierstrass approximation theorem can be found in many text-books such as [3].

Weierstrass approximation theorem postulate the existence of some sequence of polynomials converging uniformly to a prescribed continuous function on bounded closed intervals.

In practical applications if f represent the solution of a partial differential equation, then it can be approximated by a polynomial of degree less than n. The conventional form of this approximation is:

$$f(x) = p_n(x) = \sum_{i=0}^{n-1} a_i x^i \tag{1.6}$$

Where a_i are constants.

For the numerical solution of PDE's, one must find out the functional values at a certain discrete points. Now it is supposed that in a closed interval [a,b] there are n mesh points with coordinates $a = x_1 < x_2 < \cdots < x_{n-1} < x_n = b$, and the functional value at each mesh point x_i is $f(x_i)$, for all $i = 1,2,...,n$. The coefficients of the approximated polynomial (1.6) may be determined from the following system of equations

$$\left.\begin{array}{l} a_0 + a_1 x_1 + a_2 x_1^{\,2} + \cdots + a_{n-1} x_1^{\,n-1} = f\,(x_1) \\ a_0 + a_1 x_2 + a_2 x_2^{\,2} + \cdots + a_{n-1} x_2^{\,n-1} = f\,(x_2) \\ \vdots \qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ a_0 + a_1 x_n + a_2 x_n^{\,2} + \cdots + a_{n-1} x_n^{\,n-1} = f\,(x_n) \end{array}\right\} \tag{1.7}$$

The matrix equation (1.7) is of Vandermonde form, which is not singular. Thus equation (1.7) can give a unique solution $a_0, a_1, \cdots, a_{n-1}$. Once the constants are determined, the approximated polynomial is obtained. One the other hand, when n is large, the matrix is highly ill-conditioned and it inversion is very difficult. Then for this case, it is difficult to determine the constants $a_0, a_1, \cdots, a_{n-1}$ from systeam (1.7). The difficulty of determining the approximated polynomial in equation (1.6) can be removed by using Lagrange interpolation polynomial [28].

1.2.1.2 Lagrange Interpolation Polynomials, [38]:

However, polynomial approximants are not efficient in some cases, since oscillation may occurs sense. Taking Lagrange interpolation for instance if $x_1, x_2, \cdots, x_n$ are n distinct points at which the values of the function f are given, then the interpolating polynomial p of degree n, where $n \in N$ is found from the Lagrange interpolation formula:

$$p(x) = \sum_{k=1}^{n} f\,(x_k)\varphi_k(x) \tag{1.8}$$

Where

$$\varphi_k(x) = \prod_{\substack{i=1 \\ j \neq k}}^{n} \frac{x - x_i}{x_k - x_i} \tag{1.9}$$

The error in the approximation is given by

$$\left| p(x) - f\,(x) \right| = \frac{f^{\,(n)}[\xi(x)]}{n!} \prod_{i=1}^{n} (x - x_i) \tag{1.10}$$

Where $\xi(x)$ is in the smallest interval containing $x_1, x_2, ..., x_n$. Introducing the Lebesque function of the following form of

$$\tau_n(x) = \sum_{k=1}^{n} |\varphi_k(x)| \tag{1.11}$$

and a norm

$$\|f\| = \max_{a \le x \le b} |f(x)| \tag{1.12}$$

Then we have

$$\|p\| \le \|\tau_n\| \|f\| \tag{1.13}$$

This estimate is known to be sharp, that is, there exist a function for which the equality holds.

First of all, it should be pointed out that equally spaced points may leads to bad consequences because it can shown that

$$\|\tau_n\| \ge Ce^{n/2} \tag{1.14}$$

As n increases, the function value becomes large and larger, and entirely fails to approximate the function f. In other words, polynomial interpolation can be so bad that it does not yield the correct approximation at all. This situation can be avoided if we have the freedom to choose the interpolation points for the interval [a, b].

Chebyshev nodes in the following are known to be a good choice

$$x_k = \frac{1}{2}\left[a + b + (a-b)\cos\frac{(k-1)\pi}{n-1}\right], \; n \ne 1 \tag{1.15}$$

The maximum value for the associated Lebesque function in this case is

$$\|\tau_n^c\| < \frac{2}{\pi}\log n + 4 \tag{1.16}$$

Using Chebyshev nodes, we obtain the following error bounds for polynomial interpolation

$$\|f - p\| \le 2\left(\frac{b-a}{4}\right)^n \max_{a \le x \le b} \frac{|f^{(n)}(\xi_n)|}{n!} \tag{1.17}$$

1.3 Differential Quadrature Method:

As it is known from the theory of numerical analysis, Gauss quadrature is a numerical integration method. Its basic idea is to approximate a definite integral with a weighted sum of integrand values at a group of nodes in the form:

$$\int_a^b f(x)dx \approx \sum_{j=1}^{N} w_j f(x_j) \tag{1.18}$$

Where x_j, $j = 1,2,...,N$, are nodes and w_j are weighting coefficients. They are determined by solving a system of linear equations. Extending Gauss quadrature to find the derivatives of various orders of a differentiable function gives rise to DQM. In other words, the derivatives of a function are approximated by weighted sums of the function values at a group of nodes. Suppose function f is sufficiently smooth on the interval $[x_1, x_n]$. On that interval, N distinct nodes are defined:

$$x_1 < x_2 < \cdots < x_N \tag{1.19}$$

The function values on these nodes are assumed to be

$$f_1, f_2, \cdots, f_N \tag{1.20}$$

Based on DQ, the first and second order derivatives on each of these nodes are given by

$$\frac{df(x_i)}{dx} \approx \sum_{j=1}^{N} a_{ij} f_j, \qquad i = 1,2,...,N \tag{1.21}$$

$$\frac{d^2 f(x_i)}{dx^2} \approx \sum_{j=1}^{N} b_{ij} f_j, \quad i = 1,2,...,N \tag{1.22}$$

The coefficients a_{ij} and b_{ij} are the weighting coefficients (or simply weights) of the first and second order derivatives with respect to x, respectively.

1.3.1 Computation of the Weighting Coefficients for the First Order Derivative:

Consider a one-dimensional problem over a closed interval [a, b]. It is

supposed that there are N grid points with coordinates as $a = x_1 < x_2 < \cdots < x_N = b$.

Assumed that a function f is sufficiently smooth over the interval [a, b] so that its first order derivative $f'(x)$ at any grid point can be approximated by the following formulation

$$f'(x_i) = \sum_{j=1}^{N} a_{ij} f(x_j), \ for \ i = 1,2,\cdots,N \tag{1.23}$$

Where $f(x_j)$ represents the function value at a grid point x_j, $f'(x_i)$ indicates the first order derivative of f at x_i, and a_{ij} are the weighting coefficients of the first order derivative. The determination of weighting coefficients a_{ij}'s in eq. (1.23) is a key procedure in the DQ approximation. One the weighting coefficients are determined, the bridge to link the derivatives in the governing differential equation and the functional values at the mesh points is established. In other words, with the weighting coefficients, one can easily use the functional value to compute the derivatives. In the following, we shall show that the weighting coefficients can be efficiently computed by employing some explicit formulations.

1.3.1.1 Bellman's Approaches, [22]:

Bellman's and et al. [2] proposed two approaches to compute the weighting coefficients a_{ij} in eq. (1.23). The two approaches are based on the use of two different test functions.

Bellman's first approach: In this approach, the test functions are chosen as

$$g_k(x) = x^k, k = 0,1,\cdots,N-1 \tag{1.24}$$

Obviously, eq. (1.24) gives N test functions. For the weighting coefficients a_{ij} in eq. (1.23), i and j are taken from 1 to N. Thus, the total number of weighting coefficients is $N \times N$. To obtain these weighting coefficients, the N test functions should be applied at N grid points $x_1, x_2, \cdots x_N$ using eq.(1.23)

As a consequence, the following $N \times N$ algebraic equations for a_{ij} are obtained.

$$\left.\begin{aligned} &\sum_{j=1}^{N} a_{ij} = 0 \\ &\sum_{j=1}^{N} a_{ij} \cdot x_j = 1 \\ &\sum_{j=1}^{N} a_{ij} \cdot x_j^k = k \cdot x_i^{k-1}, k = 2,3,...,N-1 \\ &\qquad\qquad\qquad for\ i = 1,2,...,N \end{aligned}\right\} \tag{1.25}$$

System (1.25) has a unique solution because its matrix is of Vandermond form. Unfortunately, and as it is said previously when N is large, the matrix is ill-conditioned and its inversion is very difficult. In the practical application of this approach, N is usually chosen to be small.

Bellman's second approach: This approach is similar to the first approach, but with different test functions

$$g_k(x) = \frac{L_N(x)}{(x - x_k) L_N'(x_k)}, k = 1,2,...,N \tag{1.26}$$

Where $L_N(x)$ is the Legendre polynomial of degree N and $L_N'(x_k)$ is the first order derivative of $L_N(x)$. By choosing x_k to be the roots of the shifted Legendre polynomial and applying eq. (1.26) at N grid points $x_1, x_2, ..., x_N$, Bellman et al. gives a simple algebraic formulation to compute a_{ij} as follows:

$$a_{ij} = \begin{cases} \dfrac{L_N'(x_i)}{(x_i - x_j) L_N'(x_j)}, for\ \ j \neq i \\ \dfrac{1 - 2x_i}{2x_i (x_i - 1)} \qquad ,for\ \ j = i \end{cases} \tag{1.27}$$

Using eq. (1.27), the computation of the weighting coefficients is a simple task. However, this approach is not flexible as the first approach

because the coordinates of the grid points in this approach cannot be chosen arbitrarily. Instead, they should be chosen as the roots of the Legendre polynomial of degree N. So, eq. (1.27) only reflects a special case. Due to the inflexibility associated with the second approach in selecting the grid points, the first approach is usually adopted in practical applications

1.3.1.2 Quan and Chang's Approach, [22]:

To improve Bellman's approaches, many attempts have been made by researchers. One of the most useful approaches is the one introduced by Quan and Chang [20] and [21]. Quan and Chang used the following Lagrange interpolation polynomial as the test functions,

$$g_k(x) = \frac{M(x)}{(x - x_k)M'(x_k)}, k = 1,2,\cdots,N \tag{1.28}$$

Where

$$M(x) = (x - x_1)(x - x_2)\cdots(x - x_N) \tag{1.29}$$

$$M'(x_i) = \prod_{\substack{k=1 \\ k \neq i}}^{N} (x_i - x_k) \tag{1.30}$$

Subsequently, by applying eq. (1.28) at N grid points, they obtained the following algebraic formulations to compute the weighting coefficients a_{ij} ,

$$a_{ij} = \begin{cases} \dfrac{1}{x_j - x_i} \displaystyle\prod_{\substack{k=1 \\ k \neq i}}^{N} \frac{x_i - x_k}{x_j - x_k}, & for\ j \neq i \\ \\ \displaystyle\sum_{k=1, k \neq i}^{N} \frac{1}{x_i - x_k}, & for\ j = i \end{cases} \tag{1.31}$$

When eq. (1.31) is used, there is no restriction on the choice of the grid points.

1.3.1.3 Shu's General Approach, [28]:

This approach was inspired from Bellman's approaches. It covers all the above approaches, including Quan and Chang's approach. Starting from Bellman's two approaches, Shu raised two questions. The first one is why we can use two approaches to compute the weighting coefficients. The second is whether these two approaches give the same weighting coefficients. It was found that these questions many be answered by polynomial approximation and linear vector apace analysis.

As it was shown in section 1.2 that the solution of a PDE can be accurately approximated by a polynomial of high degree. Now, we suppose that the degree of the approximated polynomial is $N-1$. The approximated polynomial constitutes an $N-$dimensional linear vector space V_N with the operation of vector addition and scalar multiplication, and can be expressed in different forms. One popular form is

$$f(x)=\sum_{k=1}^{N} c_k x^{k-1} \tag{1.32}$$

Where c_k is a constant. There exist many sets of base vectors in the linear vector space V_N. So, the base vectors are also called the base polynomials. Four typical sets of the base polynomials are listed as follows,

$$r_k(x)=x^{k-1},\ k=1,2,\cdots,N \tag{1.33a}$$

$$r_k(x)=\frac{L_N(x)}{(x-x_k)L_N'(x_k)},\ k=1,2,\cdots,N \tag{1.33b}$$

$$r_k(x)=\frac{M(x)}{(x-x_k)M'(x_k)},\ k=1,2,\cdots,N \tag{1.33c}$$

$$r_1(x)=1, r_k(x)=(x-x_{k-1})r_{k-1}(x),\ k=2,3,\cdots,N \tag{1.33d}$$

Where $L_N(x)$ is the Legendre polynomial of degree N and $M(x)$ is defined in eq. (1.29). Among the four sets of base polynomial, eq. (1.33b) and (1.33c) are from the Lagrange interpolation polynomials while eq. (1.33d) is from the

Newton interpolation polynomials. eq. (1.33b) is a special case of equation (1.33c) since it is only valid at the Legendre collocation points.

It can be seen that eq. (1.33a) is the same as the test functions of Bellman's first approach while Equation (1.33b) is the same as the test functions of Bellman's second approach. In other words, the test functions of Bellman's two approaches are actually two sets of base polynomials in V_N. Note that eq. (1.23) are a linear operators. Then from the properties of a linear vector space, it is know that if one set of base polynomials satisfies a linear operator such as eq. (1.23), so do other sets of base polynomials, this means that every set of base polynomials would give the same weighting coefficients. Hence, the weighting coefficients do not depend on the choice of the test functions.

On the other hand, on may notice that the difference in Bellman's two approaches only lies in the use of different test functions. The test functions are equivalent to the base polynomials. Thus, the use of different sets of base polynomials will result in different approaches to compute the weighting coefficients. Since there are many sets of base polynomials in the linear vector space V_N, we have many approaches to compute the weighting coefficients.

When the set of base polynomials is given by eq. (1.33a), we can obtain the same Equation system (1.25) as given by Bellman's first approach, and when the set of base polynomials is obtained from eq. (1.33b), we can get the same algebraic formulation (1.27) as given by Bellman's second approach. Here, for generality, we use two sets of base polynomials. The Lagrange interpolation polynomials (1.33c) are taken as the first set of base polynomials. For simplicity, we get

$$M(x) = N(x, x_k)(x - x_k),\ k = 1, 2, \cdots, N \tag{1.34}$$

With $N(x_i, x_j) = M^{(1)}(x_i)\delta_{ij}$, where δ_{ij} is the Kronecker delta function. Using eq. (1.34), then eq. (1.33c) can be simplified to

$$r_k(x) = \frac{N(x, x_k)}{M'(x_k)}, \; k = 1, 2, \cdots, N \tag{1.35}$$

Substituting eq. (1.35) into eq. (1.23) gives

$$a_{ij} = \frac{N'(x_i, x_j)}{M'(x_j)} \tag{1.36}$$

In eq. (1.36), $M'(x_j)$ can be easily computed from eq. (1.30). To evaluate $N'(x_i, x_j)$, we differentiate successively eq. (1.34) with respect to x and obtain the following recurrence formulation will be obtained

$$M^{(m)}(x) = N^{(m)}(x, x_k)(x - x_k) + mN^{(m-1)}(x, x_k) \tag{1.37}$$

for all $m = 1, 2, \ldots, N-1$; $k = 1, 2, \ldots, N$

where $M^{(m)}(x)$ and $N^{(m)}(x, x_k)$ indicate the m−th order derivative of $M(x)$ and $N(x, x_k)$. $N^{(1)}(x_i, x_j)$ can be obtained as follows:

From eq.(1.37) and $m = 1$ one can get:

$$M'(x) = N'(x, x_j)(x - x_j) + N(x, x_j)$$

$$M'(x_i) = N'(x_i, x_j)(x_i - x_j) + N(x_i, x_j)$$

$$M'(x_i) = N'(x_i, x_j)(x_i - x_j) + M'(x_i)\delta_{ij}$$

If $i \neq j$ then

$$M'(x_i) = N'(x_i, x_j)(x_i - x_j)$$

Hence

$$N'(x_i, x_j) = \frac{M'(x_i)}{(x - x_j)} \tag{1.38}$$

And in order to evaluate $N'(x_i, x_i)$ can be evaluate $M''(x)$ from eq.(1.37), for $M = 2$ as follows:

$$M''(x) = N''(x, x_i)(x - x_i) + 2N'(x, x_i)$$

$$M''(x_i) = 2N'(x_i, x_i)$$

Therefore we get

$$N'(x_i, x_i) = \frac{M''(x_i)}{2} \tag{1.39}$$

finally by combining eqs. (1.38) and (1.39) together we have

$$N'(x_i, x_j) = \begin{cases} \dfrac{M'(x_i)}{x_i - x_j}, & for\ i \neq j \\ \dfrac{M''(x_i)}{2}, & for\ i = j \end{cases} \tag{1.40}$$

Substituting eq. (1.40) into eq. (1.36), we obtain

$$a_{ij} = \begin{cases} \dfrac{M'(x_i)}{(x_i - x_j) \cdot M'(x_j)}, & for\ i \neq j \\ \dfrac{M''(x_i)}{2M'(x_i)}, & for\ i = j \end{cases} \tag{1.41}$$

It is observed from eq. (1.41) that, if x_i is given, it is easy to compute $M'(x_i)$ from eq. (1.30), and hence a_{ij} for $i \neq j$. However, the calculation of a_{ii} is based on the computation of the second order derivative $M''(x_i)$ which is not an easy task. This difficulty can be eliminated by using the second set of base polynomials. According to the property of a linear vector space if one set of base polynomials vanishes a linear operator, say eq. (1.23), so does another set of base polynomials. As a consequence, the equation system for the determination of a_{ij} derived from the interpolation polynomials eq. (1.33c) should be equivalent to that derived from another set of base polynomials $x^{k-1}, k = 1, 2, ..., N$ (eq. (1.33a)). Thus a_{ij} satisfies the following equation which is obtained by the base polynomial x^{k-1} when $k = 1$

$$\sum_{j=1}^{N} a_{ij} = 0 \ or \ a_{ii} = -\sum_{j=1, j\neq i}^{N} a_{ij} \tag{1.42}$$

eq. (1.41) and (1.42) are two formulations to compute the weighting coefficients a_{ij} . It is noted that in the development of these two formulations, the two sets of base polynomials were used in the linear polynomial vector space V_N .

1.3.2 Computation of the Weighting Coefficients for the Higher Order Derivatives:

1.3.2.1 Quan and Chang's [22]:

Similarly, to derive formulas for higher order derivatives by using the higher order weighting coefficients, which are expressed as $e_{ij}^{(m)}$ to avoid confusion. They are characterized by recurrence formula:

$$e_{ij}^{(m)} = \begin{cases} m\left(a_{ij} e_{ii}^{(m-1)} - \dfrac{e_{ij}^{(m-1)}}{x_j - x_i} \right), & i, j = 1,2,...,N, i \neq j, m = 2,3,...,N-1 \\ -\sum\limits_{\substack{j=1 \\ j \neq i}}^{N} e_{ij}^{(m)}, & i, j = 1,2,...,N, i = j, m = 2,3,...,N-1 \end{cases} \tag{1.43}$$

Here, it is assume that $a_{ij} = e'_{ij}$ and $b_{ij} = e''_{ij}$ which are the weighting coefficients of the first and second order derivatives with respect to x respectively.

1.3.2.2 Shu's Recurrence Formulation [28]:

For the discretizeation of higher derivatives, the following two linear operators are applied

$$f^{(m-1)}(x_i) = \sum_{j=1}^{N} w_{ij}^{(m-1)} f(x_j) \tag{1.44}$$

$$f^{(m)}(x_i) = \sum_{j=1}^{N} w_{ij}^{(m)} f(x_j) \tag{1.45}$$

for all $i = 1,2,...,N$; $m = 2,3,...,N-1$

Where $f^{(m-1)}(x_i)$, $f^{(m)}(x_i)$ indicate the $(m-1)-th$ and $m-th$ order derivatives of $f(x)$ evaluated to x_i, $w_{ij}^{(m-1)}$ and $w_{ij}^{(m)}$ are the weighting coefficients related to $f^{(m-1)}(x_i)$ and $f^{(m)}(x_i)$ respectively. Two sets of base polynomials will also be used to derive the explicit formulations for $w_{ij}^{(m)}$. The first set of base polynomial is given by eq. (1.35). Substituting eq. (1.35) into eq. (1.44) and (1.45) gives

$$w_{ij}^{(m-1)} = \frac{N^{(m-1)}(x_i, x_j)}{M'(x_j)} \tag{1.46}$$

$$w_{ij}^{(m)} = \frac{N^{(m)}(x_i, x_j)}{M'(x_j)} \tag{1.47}$$

By rewriting eq. (1.46), we obtain

$$N^{(m-1)}(x_i, x_j) = w_{ij}^{(m-1)} M'(x_j) \tag{1.48}$$

Eq. (1.48) is valid for any i and j. On the other hand, from the recurrence formulation (1.37), we have

$$N^{(m-1)}(x_i, x_i) = \frac{M^{(m)}(x_i)}{m} \tag{1.49}$$

$$N^{(m)}(x_i, x_j) = \frac{M^{(m)}(x_i) - mN^{(m-1)}(x_i, x_j)}{x_i - x_j}, for\ i \neq j \tag{1.50}$$

$$N^{(m)}(x_i, x_i) = \frac{M^{(m+1)}(x_i)}{m+1} \tag{1.51}$$

Substituting eq. (1.49) into eq. (1.50) leads to

$$N^{(m)}(x_i, x_j) = \frac{m[N^{(m-1)}(x_i, x_i) - N^{(m-1)}(x_i, x_j)]}{x_i - x_j}, for\ i \neq j \tag{1.52}$$

Eq. (1.52) can be further simplified using eq. (1.48),

$$N^{(m)}(x_i, x_j) = \frac{m[w_{ii}^{(m-1)} M^{(1)}(x_i) - w_{ij}^{(m-1)} M^{(1)}(x_j)]}{x_i - x_j}, \text{for } i \neq j \tag{1.53}$$

Substituting eq.(1.53) into eq. (1.47), and using eq. (1.41), a recurrence formulation is obtained as follows

$$w_{ij}^{(m)} = m\left(a_{ij} w_{ii}^{(m-1)} - \frac{w_{ij}^{(m-1)}}{x_i - x_j}\right) \tag{1.54}$$

$for\ i, j = 1,2,...,N;\ m = 2,3,...,N-1.$

Where a_{ij} is the weighting coefficient of the first order derivative described above. The formulation for $w_{ii}^{(m)}$ can be obtained by substituting eq. (1.51) into eq. (1.47), which gives:

$$w_{ii}^{(m)} = \frac{M^{(m+1)}(x_i)}{(m+1) \cdot M'(x_i)} \tag{1.55}$$

$for\ i, j = 1,2,...,N;\ m = 2,3,...,N-1.$

Obviously, eq. (1.54) offers an easy way to compute the weighting coefficients $w_{ij}^{(m)}$ for $i \neq j$. However, it is very difficult to apply eq. (1.55) to compute the weighting coefficients $w_{ii}^{(m)}$. Again, this difficulty can be overcome by the properties of a linear vector space. In terms of the analysis of the Lagrange interpolation polynomial should be equivalent to that derived from the base polynomial $x^{k-1}, k = 1,2,...,N$ .Thus $w_{ij}^{(m)}$ should satisfy the following equation obtained from the base polynomial x^{k-1} when $k = 1$

$$\sum_{j=1}^{N} w_{ij}^{(m)} = 0 \quad or \quad w_{ii}^{(m)} = -\sum_{j=1, j\neq i}^{N} w_{ij}^{(m)} \tag{1.56}$$

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

1.4 Direct Differential Quadrature Method:

Solving differential equations (ordinary and partial) is one of the most

important ways for engineers, physicists and applied mathematicians which are used to tackle a practical problem. Although, typical analytical techniques of which is separation of variable, their applications are restricted to over-simplified problems. Numerical methods, the most famous of which is the Finite Element Method (FEM), are widely applied today to solve more practical and complicated problems [38].

Since the FEM is a low-order numerical tool. This demands an extensive study of high-order numerical methods. Direct DQM is such a technique, which also provides a cost-effective tool for solving many nonlinear partial differential equations. In the following, however, a linear example is used to illustrate the direct DQM.

The key procedure in the direct DQM is to approximate the derivatives in a differential equation by Equation.

$$\frac{df(x)}{dx} \approx \sum_{k=1}^{N} a_k(x) f(x_k), \quad \text{and} \quad \frac{d^2 f(x)}{dx^2} \approx \sum_{k=1}^{N} b_k(x) f(x_k), \tag{1.57}$$

Substituting eq. (1.57) into the governing equations and equating both sides of the governing equations, we obtain simultaneous equations which can be solved by use of Gauss elimination method or other methods. We will elaborate this point through the following example.

Example 1.1(Bending of Euler Beam) [38]:

A uniform Euler beam under pure bending shown in Fig. (1.1)

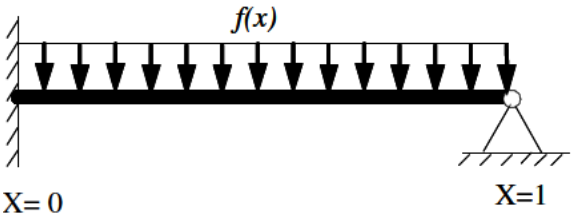

Figure (1.1): Euler beam under uniform loading.

Governed by the following fourth-order Euler-beam equation:

$$EI\frac{d^4w}{dx^4}+f(x)=0,\ 0<x<L \tag{1.68}$$

where EI is the flexural rigidity of the beam, f the external distributed load, and L the length of the beam. Eq. (1.68) may be further transformed to a dimensionless form for the convenience of calculation. With non-dimensionalisation procedures neglected, we obtain

$$\frac{d^4W}{dX^4}+F(x)=0,\ 0<X<1 \tag{1.69}$$

Where $X=x/L$, $W=w/a$, $a=f_0L/EI$, $F(x)=f(x)/f_0$, f_0 is a constant for non-dimensionalisation. As shown in Fig. (1.1) the beam is clamped at the left end and simply supported at the right end. The boundary conditions are then:

$$W=\frac{dW}{dX}=0 \qquad at\ X=0 \tag{1.70a}$$

$$W=\frac{d^2W}{dX^2}=0 \qquad at\ X=1 \tag{1.70b}$$

The exact solution is then given by $W(X)=\frac{1}{48}X^2(5X-2X^2-3)$. Divide the beam domain $0\le X\le 1$ into $N=21$ nodes distributed in the following form:

$$x_i=x_1+\frac{1}{2}\left(1-\cos\frac{i-1}{N-1}\pi\right)(x_N-x_1),\ i=1,2,...,N \tag{1.71}$$

Writing the deflection $\frac{d^4W}{dx^4}$ in the form of eq. (1.43). and substituting the result in eq. (1.69) we obtain yields to

$$\sum_{j=1}^{N}e_{ij}^{(4)}W_j=-F(x_i)\ \ for\ \ i=1,2,...,N \tag{1.72}$$

where $e_{ij}^{(4)}$ are the weighting coefficients of the fourth order derivative.

Inserting eq. (1.21) and (1.22) into eq. (1.70) we obtain the boundary conditions in the discrete form of

$$W_1 = 0, \qquad \sum_{j=1}^{N} e'_{1j} W_j = 0 \tag{1.73a}$$

$$W_N = 1, \qquad \sum_{j=1}^{N} e''_{N,j} W_j = 0 \tag{1.73b}$$

Consider a uniform load with the value $f(x) = f_0$. Then $F(X) = 1$. The deflections of the beam at the nodes is then governed by the system of linear equations

$$\sum_{j=3}^{N-2} C_{ij} W_j = -1 \quad for \quad i = 3,4,...,N-2 \tag{1.74}$$

Where

$$C_{ij} = e^{(4)}_{i,j} + \frac{e^{(4)}_{i,2}(a_{1,j} b_{N,N-1} - a_{1,N-1} b_{N,j}) + e^{(4)}_{i,N-1}(a_{1,2} b_{N,j} - a_{1,j} b_{N,2})}{b_{N,2} a_{1,N-1} - a_{1,2} b_{N,N-1}} \tag{1.75}$$

Solving eq. (1.75)(see program 1 Appendix A) we get the numerical result of the problem which are listed in table (1.1).

Table (1.1)

Deflection of a beam under uniformly distributed Load

X	W(Exact)	W(DQ,N=21)	Error (%)
0	0.0	0.0	0.0
0.00616	-2.3441×10^{-6}	-2.3441×10^{-6}	-3.4562×10^{-11}
0.02447	-3.5917×10^{-5}	-3.5917×10^{-5}	-3.3525×10^{-11}
0.0545	-1.6912×10^{-4}	-1.6912×10^{-4}	-3.2854×10^{-11}
0.09549	-4.8267×10^{-4}	-4.8267×10^{-3}	-3.3289×10^{-11}
0.14645	-1.0324×10^{-3}	-1.0324×10^{-3}	-3.4340×10^{-11}
0.20611	-1.8181×10^{-3}	-1.8181×10^{-3}	-3.5790×10^{-11}
0.273	-2.7701×10^{-3}	-2.7701×10^{-3}	-3.7369×10^{-11}
0.34549	-3.7581×10^{-3}	-3.7581×10^{-3}	-3.9038×10^{-11}
0.42178	-4.6212×10^{-3}	-4.6212×10^{-3}	-4.1047×10^{-11}
0.5	-5.2083×10^{-3}	-5.2083×10^{-3}	-4.3898×10^{-11}
0.57822	-5.4161×10^{-3}	-5.4161×10^{-3}	-4.7370×10^{-11}
0.65451	-5.2139×10^{-3}	-5.2139×10^{-3}	-5.0954×10^{-11}
0.727	-4.6473×10^{-3}	-4.6473×10^{-3}	-5.4255×10^{-11}
0.79389	-3.8218×10^{-3}	-3.8218×10^{-3}	-5.7485×10^{-11}
0.85355	-2.8738×10^{-3}	-2.8738×10^{-3}	-5.9442×10^{-11}
0.90451	-1.9384×10^{-3}	-1.9384×10^{-3}	-6.0730×10^{-11}
0.97553	-5.0892×10^{-4}	-5.0892×10^{-4}	-4.7869×10^{-11}
1	0.0	0.0	0

1.5 Application of DQM to Vibration Analysis of Beams

The vibration of a uniform beam is governed by a fourth-order differential equation. When a numerical method is applied to descretize the spatial derivatives. The ordinary differential equation can be reduced to a set of algebraic equations.

The eigenvalue of the resultant algebraic equation system provide the vibrational frequencies of the problem .Usually, the number of interior grid points is equal to the dimension of the resultant system algebraic equation, thus providing the same number of eigenfrequencies.

Among all the computed eigenfrequencies, only low frequencies are of practical interest [28].

1.5.1 Governing Equations and Boundary Conditions, [28]:

For a beam, three problems are often encountered, i.e., the bending, vibration and column buckling analysis. For a Bernoulli-Euler beam of varying cross-section with length L, the non-dimensional governing equations, for the bending analysis, and for the vibration analysis, and for the vibration analysis, respectively, are

$$s(X)\frac{d^4W}{dX^4}+2\frac{ds(X)}{dX}\frac{d^3W}{dX^3}+\frac{d^2s(X)}{dX^2}\frac{d^2W}{dX^2}+\frac{L^4q(x)}{EI_0}=0 \tag{1.76}$$

$$s(X)\frac{d^4W}{dX^4}+2\frac{ds(X)}{dX}\frac{d^3W}{dX^3}+\frac{d^2s(X)}{dX^2}\frac{d^2W}{dX^2}-\Omega^2W=0 \tag{1.77}$$

$$s(X)\frac{d^4W}{dX^4}+2\frac{ds(X)}{dX}\frac{d^3W}{dX^3}+\frac{d^2s(X)}{dX^2}\frac{d^2W}{dX^2}+\frac{PL^2}{EI_0}\frac{d^2W}{dX^2}=0 \tag{1.78}$$

where $s(X)=EI/EI_0, \Omega^2=\omega^2/EI_0,\ X=x/L,\ EI$ is the beam's flexural rigidity, ρA is the mass per unit length, $q(x)$ is the external distributed load. ω is the dimensional frequency, P is the axial compressive load for a beam of varying cross-section. EI and A are functions of the coordinate x. The governing equation for a beam is a $4-th$ order ordinary differential equation. For a well-posed problem. It requires four boundary conditions. These can be obtained by specifying two boundary conditions at the end

$X = 0$, and another two boundary conditions at the end $X = 1$. Basically, there are three types of boundary conditions. For vibration analysis, these boundary conditions are given as

Simply supported end (SS)

$$\text{W=0} \quad \text{and} \quad \frac{d^2W}{dx^2} = 0 \tag{1.79}$$

Clamped end (C)

$$\text{W=0} \quad \text{and} \quad \frac{dW}{dx} = 0 \tag{1.80}$$

Free end (F)

$$\frac{d^2W}{dX^2} = 0 \quad \text{and} \quad \frac{d^3W}{dX^3} = 0 \tag{1.81}$$

1.5.2 Numerical Discretization

For the numerical computation, the continuous solution is approximated by the functional values at discrete points. Now, we assume that the computational domain $0 \le X \le 1$ is divided into $(N-1)$ intervals with coordinates of the grid points given as $X_1, X_2, ..., X_N$. Although a uniform mesh can be used for the analysis, it is recommended that a non-uniform mesh be used. Here, we adopt the mesh point distribution used by Shu [Shu, 1991],

$$X_i = \frac{1}{2}\left[1 - \cos\left(\frac{i-1}{N-1} \cdot \pi\right)\right], \; i = 1, 2, ..., N, \; N \neq 1 \tag{1.82}$$

With the coordinates of mesh points given by (1.82), the PDQ weighting coefficients can be easily computed. These weighting coefficients can then be used to discretize eqs. (1.76) and (1.78). Using the DQM, where eq. (1.76) can be discretized as:

$$s''(X_i)\sum_{k=1}^{N}c_{ik}''W_k + 2s'(X_i)\sum_{k=1}^{N}c_{ik}'''W_k + s(X_i)\sum_{k=1}^{N}c_{ik}^{(4)}W_k = -\frac{L^4}{EI_0}q(X_i) \tag{1.83}$$

Similarly, eq. (1.77) is discretized as

$$s''(X_i)\sum_{k=1}^{N}c_{ik}''W_k + 2s'(X_i)\sum_{k=1}^{N}c_{ik}'''W_k + s(X_i)\sum_{k=1}^{N}c_{ik}^{(4)}W_k = \Omega^2 W_i \tag{1.84}$$

and eq. (1.78) is discretized by:

$$s''(X_i)\sum_{k=1}^{N}c_{ik}''W_k + 2s'(X_i)\sum_{k=1}^{N}c_{ik}'''W_k + s(X_i)\sum_{k=1}^{N}c_{ik}^{(4)}W_k = -\frac{PL^4}{EI_0}\sum_{k=1}^{N}c_{ik}''W_k \tag{1.85}$$

Where $W_i, i = 1,2,...,N$, is the functional value at the grid point $X_i, s''(X_i)$ and $s'(X_i)$ are respectively the second and first derivatives of $s(X)$ at X_i, $c_{ik}^{(n)}, n = 2,3,4$ are the DQ weighting coefficients of the $n-th$ order derivative. With proper implementation of the boundary conditions, eq. (1.83) can be written in matrix form as

$$[A_d]\{W\} = \{b\} \tag{1.86}$$

Where $[A_d]$ is a matrix, $\{W\}$ is a vector of unknowns and $\{b\}$ is a known vector. The algebraic system (1.86) can be solved using the direct methods such as the LU decomposition, or iterative methods such as the SOR approach. In a similar manner, eq. (1.84) can be written as

$$[A_v]\{W\} = \Omega^2\{W\} \tag{1.87}$$

and eq.(1.85) can be put into

$$[A_b]\{W\} = \frac{PL^2}{EI_0}[B]\{W\} \tag{1.88}$$

where $[A_v], [A_b]$ and $[B]$ are matrices, eqs. (1.87) and (1.88) are eigenvalue systems. The frequency of a free vibration problem can be obtained from the eigenvalues of eq. (1.87). the eigenvalues of eqs. (1.87) and (1.88) can be

calculated using QR algorithm**.**

1.5.3 Implementation of Boundary Conditions:

To solve fourth-order differential equations where two boundary conditions are present at each end.We need to know how to implement such boundary conditions. To properly implement these two boundary conditions, three approaches, namely, the $\delta-$technique, the modified weighting coefficient matrix approach and the approach of directly substituting the boundary conditions into the discrete governing equations will be introduced.

1.5.3.1 The $\delta-$technique:

The $\delta-$technique was proposed by Jang, Bert and Striz [8] to eliminate the difficulties in implementing two conditions at a single boundary point. It applies the Dirichlet condition ($W=0$) at the boundary point itself and the derivative condition at its adjacent point which is at a distance δ from the boundary point.

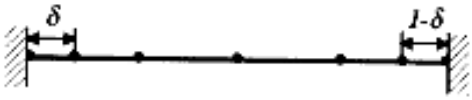

Figure (1.2) Illustration of δ-points for a beam problem.

As shown in Figure (1.2), the δ-points are actually the grid points $X_2=\delta$ and $X_{N-1}=1-\delta$. As an example, when the simply supported condition is specified at the two ends, the derivative condition $\partial^2W/\partial X^2=0$ is discretized at the mesh points X_2 and X_{N-1}, which gives

$$\sum_{k=1}^{N} c''_{2,k} W_k = 0 \qquad (1.89a)$$

$$\sum_{k=1}^{N} c''_{N-1,k} W_k = 0 \qquad (1.89b)$$

Clearly, the derivative condition is not accurately approximated by the

$\delta-$technique. There are two major drawbacks to the $\delta-$technique. One drawback arises from the implementation of the derivative condition at the $\delta-$point. Since it is an approximation to the true boundary condition that should be implemented at the boundary one can expect that the numerical results may depend on the choice of the $\delta-$value. To obtain an accurate numerical solution the values of δ should be chosen to be very small (possibly not greater than 0.0001). However, this small value of δ would result in the second drawback of this technique. When one mesh spacing (δ) is much smaller than the others, the DQ weighting coefficient matrices may become highly ill-conditioned, which when causes the solution to oscillate. As a result, the numerical solution is less accurate.

1.5.3.2 Modification of the Weighting Coefficient Matrices:

The technique was presented by Wang and Bert [32] to overcome the drawbacks of the $\delta-$ technique. For this approach, only one boundary condition is numerically implemented. The other boundary condition (derivative condition) is built into the DQ weighting coefficient matrices. To demonstrate this approach, we consider the following boundary condition (C-SS)

$$W = 0, dW / dX = 0, \text{ at } X = 0 \tag{1.90a}$$

$$W = 0, d^2W / dX^2 = 0, \text{ at } X = 1 \tag{1.90b}$$

Let $[A^{(n)}]$ and $[\widetilde{A}^{(n)}]$ be respectively the original and modified weighting coefficient matrices of the $n-th$ order derivative. Since

$$\frac{d^n W}{dx^n} = \frac{d}{dx}\left(\frac{d^{n-1} W}{dx^{n-1}}\right) \tag{1.91}$$

We then have

$$[A^{(n)}] == [A'][A^{(n-1)}] = [A''][A^{(n-2)}] = \cdots = [A^{(n-1)}][A'] \tag{1.92}$$

Now, consider the DQ analog of the derivative dW / dX at all mesh points

$$\begin{bmatrix} (dW/dX)_1 \\ (dW/dX)_2 \\ \vdots \\ (dW/dX)_{N-1} \\ (dW/dX)_N \end{bmatrix} = \begin{bmatrix} c'_{1,1} & c'_{1,2} & \cdots & c'_{1,N-1} & c'_{1,N} \\ c'_{2,1} & c'_{2,2} & \cdots & c'_{2,N-1} & c'_{2,N} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ c'_{N-1,1} & c'_{N-1,2} & \cdots & c'_{N-1,N-1} & c'_{N-1,N} \\ c'_{N,1} & c'_{N,2} & \cdots & c'_{N,N-1} & c'_{N,N} \end{bmatrix} \begin{Bmatrix} W_1 \\ W_2 \\ \vdots \\ W_{N-1} \\ W_N \end{Bmatrix} \tag{1.93}$$

To satisfy the derivative condition of eq. (1.90a), modify eq. (1.93) by zeroing the first row of the matrix

$$\begin{bmatrix} (dW/dX)_1 \\ (dW/dX)_2 \\ \vdots \\ (dW/dX)_{N-1} \\ (dW/dX)_N \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ c'_{2,1} & c'_{2,2} & \cdots & c'_{2,N-1} & c'_{2,N} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ c'_{N-1,1} & c'_{N-1,2} & \cdots & c'_{N-1,N-1} & c'_{N-1,N} \\ c'_{N,1} & c'_{N,2} & \cdots & c'_{N,N-1} & c'_{N,N} \end{bmatrix} \begin{Bmatrix} W_1 \\ W_2 \\ \vdots \\ W_{N-1} \\ W_N \end{Bmatrix} \tag{1.94}$$

Obviously, the matrix in eq. (1.93) is $[A']$, and the matrix in eq. (1.94) is $[\widetilde{A}']$. To satisfy the boundary condition (1.90a), the second order weighting coefficient matrix $[\overline{A}'']$ can be derived from eq. (1.92)

$$[\overline{A}''] = [A'][\widetilde{A}'] \tag{1.95}$$

In order to satisfy the boundary condition (1.90b) at $X = 1$, $[\overline{A}'']$ should be modified by zeroing its last row, that is,

$$[\tilde{A}''] = \begin{bmatrix} \overline{c}''_{1,1} & \overline{c}''_{1,2} & \cdots & \overline{c}''_{1,N-1} & \overline{c}''_{1,N} \\ \overline{c}''_{2,1} & \overline{c}''_{2,2} & \cdots & \overline{c}''_{2,N-1} & \overline{c}''_{2,N} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \overline{c}''_{N-1,1} & \overline{c}''_{N-1,2} & \cdots & \overline{c}''_{N-1,N-1} & \overline{c}''_{N-1,N} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{1.96}$$

It is noted that the derivative conditions given in Equation (1.90) have been completely built into the matrix $[\widetilde{A}'']$. With $[\widetilde{A}'']$, the modified weighting coefficient matrices $[\widetilde{A}''']$ and $[\widetilde{A}^{(4)}]$ can be computed by

$$[\widetilde{A}'''] = [A'] \cdot [\widetilde{A}''] \tag{1.97}$$

$$[\widetilde{A}^{(4)}] = [A''] \cdot [\widetilde{A}''] \tag{1.98}$$

In the above process, the weighting coefficient matrices are modified through the matrix form. This process is very simple since it just zeros some elements of the matrix.

1.5.3.3 Direct Substitution of the Boundary Conditions into the Discrete Governing Equations:

This approach was proposed by Shu and Du [26] to implement the simply supported, clamped conditions and their combinations. The essence of the approach is that the Dirichlet condition is implemented at the boundary point, while the derivative condition is discretized by the DQM. The discretized conditions at the two ends are then combined to give derivative conditions at the two ends then combined to give the solutions W_2 and W_{N-1}. The expressions for W_2 and W_{N-1} are then substituted into the discrete governing equation which is applied to the interior points $3 \le i \le N-2$. The dimension of the equation system using this approach is $(N-4) \times (N-4)$.

For any combination of the clamped and simply supported conditions at the two ends, the discrete boundary conditions using the DQM can be written as

$$W_1 = 0 \tag{1.99a}$$

$$\sum_{k=1}^{N} c_{1,k}^{(n_0)} W_k = 0 \tag{1.99b}$$

$$W_N = 0 \qquad (1.99c)$$

$$\sum_{k=1}^{N} c_{N,k}^{(n_1)} W_k = 0 \qquad (1.99d)$$

Where n_0 and n_1 may be taken as either 1 or 2.

By choosing the values of n_0 and n_1, eq. (1.99) can given the following four sets of boundary conditions,

$n_0 = 1, n_1 = 1$ .....clamped – clamped

$n_0 = 1, n_1 = 2$ .....clamped – simply supported

$n_0 = 2, n_1 = 1$.....simply supported – clamped

$n_0 = 2, n_1 = 2$.....simply supported – simply supported

Eqs. (1.99a) and (1.99c) can be easily substituted into eq. (1.84) This is not the case for Equations (1.99b) and (1.99d). However, one can couple these two equations together to give two solutions, W_2 and W_{N-1}, as

$$W_2 = \frac{1}{AXN} \sum_{k=3}^{N-2} AXK1 W_k \qquad (1.100a)$$

$$W_{N-1} = \frac{1}{AXN} \sum_{k=3}^{N-2} AXKN W_k \qquad (1.100b)$$

Where

$$AXK1 = c_{1,k}^{(n_0)} c_{N,N-1}^{(n_1)} - c_{1,N-1}^{(n_0)} c_{N,k}^{(n_1)}$$

$$AXKN = c_{1,2}^{(n_0)} c_{N,k}^{(n_1)} - c_{1,k}^{(n_0)} c_{N,2}^{(n_1)}$$

$$AXN = c_{N,2}^{(n_1)} c_{1,N-1}^{(n_0)} - c_{1,2}^{(n_0)} c_{N,N-1}^{(n_1)}$$

According to eq. (1.100), W_2 and W_{N-1} are expressed in terms of $W_3, W_4, ..., W_{N-2}$, and can be easily substituted into eq. (1.84). It should be noted that Equation (1.99) provides four boundary equations. In total, we have N unknowns $W_1, ..., W_N$. In order to close the system, the discretized governing eq. (1.84) has to be applied at $(N-4)$ mesh points. This can be done by applying eq. (1.84) at grid points $X_3, X_4, ..., X_{N-2}$. Substituting eqs. (1.99a), (1.99c) and (1.100) into eq. (1.84) gives

$$s''(X_i)\sum_{k=3}^{N-2}C_1W_k + 2s'(X_i)\sum_{k=3}^{N-2}C_2W_k + s(X_i)\sum_{k=3}^{N-2}C_3W_k = \Omega^2 W_i$$
$$for\ i = 3,4,...,N-2 \qquad (1.101)$$

Where

$$C_1 = c''_{i,k} - \frac{c''_{i,2}AXK1 + c''_{i,N-1}AXKN}{AXN}$$

$$C_2 = c'''_{i,k} - \frac{c'''_{i,2}AXK1 + c'''_{i,N-1}AXKN}{AXN}$$

$$C_3 = c^{(4)}_{i,k} - \frac{c^{(4)}_{i,2}AXK1 + c^{(4)}_{i,N-1}AXKN}{AXN}$$

1.5.4 Numerical Example: Free Vibration Analysis of a Uniform Beam:

The free vibration analysis of a uniform beam given by eq. (1.84) $(s(X)=1)$ is taken an example to show the high efficiency and accuracy of the DQ method. For numerical computations, the PDQ method is used to discretize the derivatives, and the method of directly substituting the boundary conditions into the discrete governing equations is applied to implement the boundary conditions .The mesh point distribution is given by eq. (1.82). Table (1.2) lists the natural frequencies of the first 5 modes for two sets of boundary conditions, that is, simply supported-simply supported (SS-SS), and clamed-simply supported (C-SS) conditions. The DQ results were provided by Shu and Du [24] using 15 mesh points. The exact results given by Blevins [9] are also include in table (1.2) for comparison. It can be observed from table (1.2) that, for the two cases, the DQ results are very accurate although only 15 mesh points are used. Calculations for the results given in table (1.2) are performed by using MATLAB program (see program 2 and 3 Appendix A).

Table (1.2)

Comparison of natural frequencies of a uniform beam with the exact solution given by [9]

	Ω_1	Ω_2	Ω_3	Ω_4	Ω_5
Simply supported-simply supported					
[9]	9.8696	39.4784	88.8264	157.9137	246.7401
PDQ(N=15)	9.8696	39.4784	88.8249	158.0619	248.4716
Clamped-simply supported					
[9]	15.4182	49.9648	104.2477	178.2697	272.0310
PDQ(N=15)	15.4182	49.9648	104.2471	178.4642	273.1126

Chapter Two

G-Spline-Based Differential Quadrature and its Application to a Uniform Beam Problem

2.1 Introduction

This chapter introduces the numerical solution of ODE using G-spline interpolation-based differential quadrature therefore this chapter will consists of six sections in section 2.2, we will present in detail the G-spline interpolation formula while in section 2.3 we will present a numerical example illustrating the construction of G-spline interpolation. The DQ and the computation of weighting coefficients for the first and second derivatives using G-spline interpolation will be given in section 2.4. In section 2.5, applying boundary conditions using the G-spline interpolation formula-based differential quadrature will be given. Finally in section 2.6, Bernoulli-Euler beam problem will be solved by using G-spline based DQM.

2.2 G- Spline Interpolation, [23]:

G-spline interpolatnts was first presented by Schoenberg in 1968 as a tool used to specify the interpolatory conditions:

$$f^{(j)}(x_i) = y_i^{(j)}, \ (i, j) \in e$$

which is called the Hermite-Birkhoff problem. (abbreviated as HB-problem). Schoenberg had developed the idea for splines, where the data of (HB-problem) is the values of the function at the node points in addition to its derivatives without the Hermite condition.

2.2.1 The Hermite-Birkhoff Problem:

As it is already mentioned in the introduction of this chapter, that the G-splines are calculated using the HB-problem, therefore it is convenient in this subsection to discuss the HB-problem and before going forward to give the tractable formal definition of the natural G-spline interpolation, let us consider the nodes:

$$x_1 < x_2 < \ldots < x_k$$

to be distinct and real also, let α be the maximum of the orders of the derivatives to be specified at the nodes. Define an incidence matrix E, by

$$E = [a_{ij}],\ i = 1,2,...,k;\ j = 0,1,...,\alpha$$

where

$$a_{ij} = \begin{cases} 1,\ if\ \ (i,j) \in e \\ 0,\ if\ \ (i,j) \notin e \end{cases}$$

Here $e = \{(i,j)\}$ has been chosen in such a way that i takes the values 1,2, …, k ; one or more times , while $j = \{0,1,...,\alpha\}$ and $j = \alpha$ is attained in at least one element (i,j) of e. Assume also that each row of the incidence matrix E and the last column of E should contain some element equals 1. Let $y_i^{(j)}$ be a prescribed real number for each $(i,j) \in e$. The HB-problem is to find $f(x) \in C^{\alpha}$, which satisfies the interpolatory condition:

$$f^{(i)}(x_i) = y_i^{(j)},\ for\ (i,j) \in e \qquad (2.1)$$

The matrix E will likewise describes the set of eqs. given by system (2.1) if we define the set e by:

$$e = \{(i,j)\ \ | a_{ij} = 1\}$$

Then

$$n = \sum_{i,j} a_{ij}$$

really is the number of interpolatory conditions required to constitute the system (2.1).

Therefore, at each node x_i of the system (2.1), we prescribes the value of $f(x_i)$ and may be also a certain number of consecutive derivatives

$f^{(j)}(x_i)$ for $j = 1,2,...,\alpha_i - 1$; where α_i denotes the number of the required derivatives at x_i, for each i.

The next definition distinguishes between different types of HB- problem, which may be encountered in applications.

Definition (2.1), [23]:

The HB-problem (2.1) is said to be normal provided that (2.1) has a unique solution $f(x)$, which belongs to $\prod_{n-1}$ (where $\prod_{n-1}$ is the set of all polynomials of degree less than or equal to $n-1$). A necessary condition for (2.1) to be normal is the inequality

$$n > \alpha \tag{2.2}$$

which we will assume in our discussion . For $n-1 \le \alpha$, then $n-1 < \alpha$, $f(x) \in \prod_{n-1}$ could not possibly always satisfies the form of system (2.1) involving $f^{(\alpha)}(x)$.

Now, in order to simplify the normality of HB- problem and makes the definition more reliable in applications, assume that system (2.1) is normal, then we shall relate for each $(i,j) \in e$, a unique function $L_{ij}(x)$ (called the fundamental function), which belongs to $\prod_{n-1}$, such that:

$$L_{ij}^{(S)}(x_r) = \delta_{ir}\delta_{js}, \ if \ \ (r,s) \in e$$

Furthermore:

$$L_{ij}^{(s)}(x_r) = \begin{cases} 1, & if \ \ (r,s) = (i,j) \\ 0, & if \ \ (r,s) \neq (i,j) \end{cases} \tag{2.3}$$

The unique solution $f(x) \in \Pi_{n-1}$ of (2.1) may be expressed in terms of the fundamental function $L_{ij}(x)$, as:

$$f(x) = \sum_{(i,j)\in e} y_i^{(j)} L_{ij}(x)$$

Moreover, if $f(x) \in C^{\alpha}$, we may write:

$$f(x) = \sum_{(i,j)\in e} f^{(j)}(x_i) L_{ij}(x) + Rf \qquad (2.4)$$

where Rf is the remainder, for $f(x) \in \prod_{n-1}$, clearly that if Rf = 0 means that (2.4) is an exact interpolant. Formula (2.4) is called the **Hermite-Birkhoff formula.**

2.2.2 On Normal Hermite-Birkhoff Problem and Related Concepts [23]:

The condition that the HB-problem (2.1) is normal may be equivalently expressed by the following requirement:

If:

$$p(x) \in \Pi_{n-1} \qquad (2.5)$$

$$p^{(i)}(x_i) = 0, \text{ if } (i,j) \in e \qquad (2.6)$$

then:

$$p(x) = 0$$

A closely related concept of the normal HB-problem is presented by the following definition:

Definition (2.2):

Let m be a natural number, then the HB-problem (2.1) is said to be m-poised provided that:

$$p(x) \in \Pi_{m-1} \qquad (2.7)$$
$$p^{(j)}(x_i) = 0 \text{ if } (i,j) \in e$$

then :

$$p(x) = 0.$$

The next three lemmas are given in **[**23] and proofed in [18]

Lemma(2.1):

The HB-problem (2.1) is normal if and only if it is n-poised.

Lemma(2.2):

If the HB-problem (2.1) is m-poised, then the inequality $m \leq n$ must hold.

Lemma (2.3):

If the HB-problem (2.1) is m-poised and $1 < m' < m$, then the HB-problem (2.1) is also m'-poised.

Remarks (2.1),[23]:

1-A non-normal system (2.1) may be m-poised for some value $m < n$, as the following example illustrate:

Consider the HB-problem:

$$f(x_1) = y_1, f'(x_2) = y_2', f(x_3) = y_3, x_2 = (x_1 + x_3)/2 \qquad (2.8)$$

Then the incidence matrix E takes the form:

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and the set e is given by:

$$e = \{(1,0),(2,1),(3,0)\}$$

Hence eq. (2.8) is not 3-poised, since:

$$p(x) = (x - x_1)(x - x_3) \neq 0$$

But it satisfies the conditions (2.5) and (2.6). However, one can notice that (2.8) is 2-poised.

2-The condition that system (2.1) is m-poised can expressed as follows:

If:

$$p(x) = \sum_{v=0}^{m-1} a_v \frac{x^v}{v!}$$

then $p^{(j)}(x_i) = 0$, for $(i, j) \in e$, become:

$$\sum_{v=0}^{m-1} a_v \frac{x_i^{v-j}}{(v-j)!} = 0, \ for \ (i, j) \in e$$

where:

$$\frac{x_i^{v-j}}{(v-j)!} = \begin{cases} 1, \ if \ v-j=0 \\ 0, \ if \ v-j<0 \end{cases}$$

Therefore, system (2.1) is m-poised if and only if the matrix with entries $\frac{x_i^{v-j}}{(v-j)!}$ has rank m, where $v = 0,1,\ldots,m-1$; refers to the column of the matrix of entries $\frac{x_i^{v-j}}{(v-j)!}$, while each $(i, j) \in e$ is an indication for a row of the same matrix.

2.2.3 Interpolation by G-Spline [23]

In this section, we shall assume that the HB-problem (2.1) is m-poised and $\alpha < m \le n$, where α, as we mentioned in subsection (2.2.1), is the highest derivative that appears in the interpolation problem.

The definition of G-spline is facilitated by defining a matrix E^* which is obtained from the incidence matrix E by adding $m-\alpha-1$ columns of zeros to the matrix E.

Let $E^* = \left[a_{ij}^*\right]$, where (i = 1,2, …, k; j = 0,1,…, m -1), and:

$$a_{ij}^* = \begin{cases} a_{ij} & if \quad j < \alpha \\ 0, & if \quad j = \alpha+1, \alpha+2, \ldots, m-1 \end{cases}$$

If m = α+1, then $E^* = E$.

Definition (2.3), [23]:

A function $S(x)$ is called natural G-spline for the nodes $x_1, x_2, \ldots, x_k$ and the matrix E^* of order m, provided that it satisfies the

following conditions:

(1) $S(x) \in \prod_{2m-1}$ in $(x_i, x_{i+1}), i = 1,2,...,k-1.$

(2) $S(x) \in \prod_{m-1}$ in $(-\infty, x_1)$ and $(x_k, \infty).$

(3) $S(x) \in C^{m-1}(-\infty, \infty).$

(4) If $a_{ij}^* = 0,$ then $S^{(2m-j-1)}(x)$ is continuous at $x = x_i$; that is, $S^{(2m-j-1)}(x_i - 0) = S^{(2m-j-1)}(x_i + 0)$, where $x_i + 0$ and $x_i - 0$ refers to the right and left hand limits of the function $S^{(2m-j-1)}(x)$.

We denote the set of all natural G-spline interpolation polynomials of given function with nodes $x_1, x_2, ..., x_k$ by:

$$S_m = S\left(E^*, x_1, x_2, \ldots, x_k\right)$$

S_m is a non empty set and this shown by the inclusion relation:

$$\prod_{m-1} \subset S_m$$

Indeed, if $S(x) \in \prod_{m-1}$, then $S(x)$ satisfies all condition from (1) to (4).

Remarks (2.2):-

1- A special case when (2.1) is given by:

$$f(x_i) = y_i, f'(x_i) = y_i', ..., f^{(\alpha_i - 1)}(x_i) = y_i^{(\alpha_i - 1)}, \; i = 1,2,...,k$$

Then the HB-problem is reduced to a Hermite problem.

It is clearly that $\alpha = \max_i (\alpha_i - 1),$ and $\max_i \alpha_i \le m \le n$; $S^{(2m-j-1)}(x)$ is continuous at $x = x_i$, for each $j = \alpha_i, ..., m-1.$ In other words $S^{(v)}(x)$ is continuous at $x = x_i$, for $v = m, m+1, ..., 2m - \alpha_i - 1$ together with condition (3), of definition (2.3) it is concluded that:

$$S(x) \in C^{2m-\alpha_i - 1}, \text{ near } x = x_i, i = 1,2,...,k \qquad (2.9)$$

Conditions (1), (2) of definition (2.3) and eq.(2.9) shows that S_m is identical with the natural spline function of degree 2m-1 having x_i $(i = 1,2,...,k)$ a multiple node of multiplicity α_i, where $\alpha_i \le m$.

2- Another special case, the Lagrange problem which occurs if we assume that $n = k+1$ and $e = \{(i,0), i = 0,1,...,k\}$. In this case, $m = k$ and it may be shown that S_m is identical with the class of natural spline functions of degree 2m-1 having nodes $x_0, x_1, ..., x_k$.

Theorem (2.1), [23]:

If the HB-problem (2.1) is m-poised, then there exists a unique G-spline function:

$$S(x) = \delta_m\left(E^*;\ x_1,\ x_2,\ \ldots,\ x_k\right)$$

such that:

$$S^{(j)}(x_i) = y_i^{(j)}, \quad for\,(i,j) \in e$$

Under the assumption of theorem (2.1), we now define a G-spline fundamental functions $L_{ij}(x) \in \delta_m$ satisfying:

$$L_{ij}^{(s)}(x_r) = \begin{cases} 0, & if \quad (r,s) \neq (i,j) \\ 1, & if \quad (r,s) = (i,j) \end{cases}$$

If for $f(x) \in C^{\alpha}$, we may write:

$$f(x) = \sum_{(i,j)\in e} f^{(j)}(x_i) L_{ij}(x) + Rf \qquad (2.10)$$

for which the right hand sum presents the G-spline interpolating $f(x)$ at the data of the HB-problem (2.1).

Equation (2.10) is called the G-spline interpolation formula, which is exact for all elements of S_m and in particular for the elements of $\prod_{m-1}$.

The following theorem shows the optimal property the of G-spline

interpolation functions and it may be called the **minimum norm property.**

Theorem (2.2),[23]:

Let $I=[x_0,x_{k+1}]$ such that $x_0<x_1<...<x_{k+1}$ and $f(x)\in C^m(I)$ with $f^{(m-1)}(x)$ is absolutely continuous and $f^{(m)}(x)\in L_2(I)$. If the HB-problem (2.1) to be m-poised, and α < m ≤ n, and let S(x) be the unique G-spline function satisfying the equation:

$$S^{(i)}(x_i)=f^{(j)}(x_i),(i,j)\in e$$

Then:

$$\int_I \left(f^{(m)}(x)\right)^2 dx > \int_I \left(S^{(m)}(x)\right)^2$$

2.2.4 Approximation of Linear Functionals with the Sense of G-Spline Formula:

Let I = [a, b] be a finite interval containing the node points $x_1,x_2,...,x_k$ and let us consider a linear functional,

$\mathcal{L}$f : C^{α} [a, b] $\longrightarrow \mathbb{R}$

of the form:

$$\mathcal{L}f=\sum_{j=0}^{\alpha}\int_a a_j(x)f^{(j)}(x)\,dx+\sum_{j=0}^{\alpha}\sum_{i=0}^{n_j}b_{ji}f^{(j)}(x_{ji}) \qquad (2.11)$$

where the $a_j(x)$ are piecewise continuous functions in I, $x_{ji}\in I$ and b_{ji} are real constants. The function (2.11) may be approximated using the formula:

$$\mathcal{L}f=\sum_{(i,j)\in e}\beta_{ij}f^{(j)}(x_i)+Rf \qquad (2.12)$$

Therefore, in order to find the approximation $\mathcal{L}f$ given by (2.12), which is best in some sense, we propose to determine the real's β_{ij}. Schoenberg [23] states two procedures to determine β_{ij} 's. One of them is the so called Sard procedure, which can be summarized in the following theorem:

Theorem (2.3),[23]:

If $\alpha < m < n$ and the HB-problem (2.1) is m-poised, then Sard's best approximation (2.12) to $\mathcal{L}f$ of order m is obtained by operating with $\mathcal{L}$ on both sides of the G-spline interpolation formula (2.10) of order m. In other words, the coefficients β_{ij} are given by:

$$\beta_{ij} = \mathcal{L}L_{ij}(x)$$

where $L_{ij}(x)$ are the fundamental functions of (2.10).

2.2.5 The Construction of G-Spline:

The construction of the G-spline interpolation formula in a more efficient approach leading to a system of only $m+n$ equations is given as follows:

From conditions (1), (2) and (3) of definition (2.3) it is clear that the most suitable form of a G-spline function S must take the form:

$$S(x) = P_{m-1}(x) + \sum_{i=1}^{k}\sum_{j=0}^{m-1} c_{ij}\frac{(x-x_i)_+^{2m-j-1}}{(2m-j-1)!} \qquad (2.13)$$

where $p_{m-1}(x) \in \Pi_{m-1}$, and

$$(x-x_i)_+^{2m-j-1} = \begin{cases} (x-x_j)^{2m-j-1}, & if\ x > x_j \\ 0, & if\ x \le x_j \end{cases}$$

while the c_{ij} are constants to be determined. Any function of the form (2.13) satisfies conditions (1), (2) and (3) of definition (2.3), except:

$$S(x) \in \prod_{m-1} \quad \text{if} \quad x_k < x$$

and according to the definition of the truncated basis, in equation (2.13), we can see that $S^{(2m-j-1)}(x)$ is continuous at $x = x_i$ if only if $c_{ij} = 0$, while condition (4) of definition (2.3) requires that $S^{(2m-j-1)}(x)$ is continuous if and only if $a_{ij}^{*} = 0$. Leaving out all such terms, we obtain:

$$S(x) = P_{m-1}(x) + \sum_{(i,j)\in e} c_{ij} \frac{(x - x_i)_+^{2m-j-1}}{(2m-j-1)!} \tag{2.14}$$

In order to satisfy (2.14), expand all binomials and equating to zero those coefficients of $x^m, x^{m+1}, ..., x^{2m-1}$, we obtain the equations:

$$\sum_{\substack{(i,j)\in e \\ j \le v}} \frac{c_{ij}}{(2m-j-1)!} \binom{2m-j-1}{2m-v-1} (-x_i)^{v-j} = 0, v = 0,1,...,m-1, \tag{2.15}$$

and also have the equations

$$S^{(i)}(x_i) = y^{(i)}(x_i), (i,j) \in e \tag{2.16}$$

Therefore, we get $n+m$ equations from (2.15) and (2.16) and writing the solution of the unique G-spline so as to exhibit the $y_i^{(j)}$ to get:

$$S(x) = \sum_{(i,j)\in e} y_i^{(j)} L_{ij}(x)$$

Which is the final form of the G-spline approximation function. It is clear that the final form of the G-spline function depends on the fundamental G-spline functions $L_{ij}(x)$, for all $(i,j) \in e$ **.**

2.3 Illustrative Example, [18]:

As an illustration to the discussion given in the subsection (2.2.5), consider the following HB-problem:

Given that:

$$f(-1) = y_1, f'(0) = y_2', f(1) = y_3 \tag{2.17}$$

and to find the G-spline function which interpolate (2.17). In this problem we have $\alpha = 1, n = 3$ and it is clear that the problem is two-poised as given by remarks (2.1)(1).

The incidence matrix is given by:

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and the HB-set e will take the form:

$$e = \{(1,0),(2,1),(3,0)\}$$

Therefore, the G-spline interpolation will be:

$$S(x) = a_0 + a_1 x + \frac{1}{6}c_{10}(x+1)_+^3 + \frac{1}{2}c_{21}x_+^2 + \frac{1}{6}c_{30}(x-1)_+^3 \qquad (2.18)$$

Now, to find the fundamental G-spline functions $L_{10}(x), L_{21}(x)$ and $L_{30}(x)$, we must solve the following linear system of algebraic equations obtained from (2.15) and (2.16) must be solved:

$$\frac{1}{6}c_{10} + \frac{1}{6}c_{30} = 0$$

$$\frac{1}{2}c_{10} + \frac{1}{2}c_{21} - \frac{1}{2}c_{30} = 0$$

$$a_0 - a_1 = y_1$$

$$a_1 + \frac{1}{2}c_{10} = y_2'$$

$$a_0 + a_1 + \frac{8}{6}c_{10} + \frac{1}{2}c_{21} = y_3$$

Which have the solution:

$$c_{10} = \frac{3}{2}y_1 + 3y_2' - y_3$$

$$c_{21} = -3y_1 - 6y_2' + 3y_3$$

$$c_{30} = -\frac{3}{2}y_1 - 3y_2' + \frac{3}{2}y_3$$

$$a_0 = \frac{1}{4}y_1 - \frac{1}{2}y_2' + \frac{3}{4}y_3$$

$$a_1 = -\frac{3}{4}y_1 - \frac{1}{2}y_2' + \frac{3}{4}y_3$$

Therefore, upon substituting $c_{10}, c_{21}, c_{30}, a_0$ and a_1 into (2.18), and writing $S(x)$ in terms of y_1, y_2' and y_3, gives:

$$S(x) = y_1 L_{10}(x) + y_2' L_{21}(x) + y_3 L_{30}(x)$$

Where

$$L_{10} = \frac{1}{4}(1-3x)_+ + \frac{1}{4}(x+1)_+^3 - \frac{3}{2}x_+^2 - \frac{1}{4}(x-1)_+^3$$

$$L_{21} = -\frac{1}{2}(1+x)_+ + \frac{1}{2}(x+1)_+^3 - 3x_+^2 - \frac{1}{2}(x-1)_+^3$$

$$L_{30} = \frac{3}{4}(1+x)_+ - \frac{1}{4}(x+1)_+^3 + \frac{3}{2}x_+^2 + \frac{1}{4}(x-1)_+^3$$

2.4 The G-Spline Interpolation-Based Differential Quadrature Method:

This section may be considered as a generalization to the DQM given in chapter one since as we saw in Remarks (2.2),(2) that Lagrange interpolation polynomial is a especial case of the Hermite-Birkhoff problem especially when $\alpha = 0$.

Suppose the function f is sufficiently smooth on the interval $[x_1, x_N]$, and let us consider an m-poised Hermite-Birkoff problem

$$f^{(j)}(x_i) = y_i^{(j)},\ (i,j) \in e \tag{2.19}$$

on the N−distinct nodes:

$$x_1 < x_2 < \ldots < x_N \tag{2.20}$$

Based on DQ, the first and second order derivatives of f on each of these nodes are given by:

$$\left.\frac{df}{dx}\right|_{x=x_k} = \sum_{(i,j)\in e} a_{k,i}^{(j)} f_i^{(j)}, k = 1,2,...,N \tag{2.21}$$

$$\left.\frac{d^2 f}{dx}\right|_{x=x_k} = \sum_{(i,j)\in e} b_{k,i}^{(j)} f_i^{(j)}, k = 1,2,...,N \tag{2.22}$$

The coefficients $a_{ki}^{(j)}$ and $b_{ki}^{(j)}$ are the weighting coefficients of the first and second order derivatives respectively.

2.4.1 Computation of the Weighting Coefficients for the First and Second Order Derivatives Via G-Spline Interpolation Formula:

To find the weights $a_{ki}^{(j)}$ and $b_{ki}^{(j)}$, consider an m-poised HB-problem to approximate the function f and our purpose is to construct a polynomial of x, which is of the form:

$$f(x) \simeq \sum_{(i,j)\in e} L_{ij}(x) f_i^{(j)} \tag{2.23}$$

satisfying

$$L_{ij}^{(s)}(x_r) = \begin{cases} 1, & if \ \ (i,j) = (r,s) \\ 0, & if \ \ (i,j) \neq (r,s) \end{cases}$$

Then the first and second derivatives of f at any grid point x_k are given by eqs. (2.21) and (2.22) respectively, where $a_{k,i}^{(j)}$ are the coefficients for the first derivative, obtain by the following formula

$$a_{k,i}^{(j)} = \left.\frac{dL_{ij}(x)}{dx}\right|_{x=x_k} \tag{2.24}$$

and similarly $b_{k,i}^{(j)}$ are the coefficients of the second derivative given by

$$b_{k,i}^{(j)} = \left.\frac{d^2 L_{ij}(x)}{dx^2}\right|_{x=x_k} \tag{2.25}$$

In the same manner we may obtain formulas for higher order derivatives

by using the higher order weighting coefficients, which are expressed as $e_{k,i}^{(j,m)}$ to avoid confusion. They are characterized by recurrence

$$e_{k,i}^{(j,m)} = \left.\frac{d^m L_{ij}(x)}{dx^m}\right|_{x=x_k} ,(i,j)\in e, k=1,2,...,N, m=1,2,3,...,N-1 \tag{2.26}$$

Here we assume that $a_{k,i}^{(j)} = e_{k,i}^{(j,1)}$ and $b_{k,i}^{(j)} = e_{k,i}^{(j,2)}$

2.4.2 Direct DQM Using G-Spline Interpolation:

In this subsection we shall follow the same manner that appeared in section 1.4 but the weighting coefficients will be in terms of G-spline interpolation and for comparison purpose we shall take the same example which is the bending of Euler beam.

Example 2.1 (Bending of Euler Beam):

Also it is present in chapter one, the non-dimensionalisation equation of the fourth order Euler beam equation is given eq. (1.69), with the boundary conditions (1.70).

Divide the beam domain $0 \le X \le 1$ into six nodes using Chebyshev points (1.15).

Applying differential quadrature for eq. (1.69) thus we have:

$$\sum_{(i,j)\in e} e_{k,i}^{(j,4)} = -F(x_k), k=1,2,...,6 \tag{2.27}$$

Next two Hermite-Birkhoff sets are considered for finding the approximate solution and therefore we shall divide the solution will be divided into two cases:

Case1:

To construct the approximate solution via G-spline-based DQM an m-poised HB- problem must be chosen.

In this case we shall take 5-poised HB-problem with

$$e_1 = \{(1,0),(2,0),(3,0),(4,0),(5,0),(6,0)\}$$

We shall seek $S_5(x) \in S(E^*, x_1, x_2, x_3, x_4, x_5, x_6)$ where

$$E = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

and for which

$$S_5(x_i) = W_i^{(j)}, (i, j) \in e_1$$

From eq. (2.27) with $F = 1$ and by applying the DQM for eqs. (1.70a) and (1.70b) we get:

$$W_1^{(0)} = 0 \quad W_6^{(0)} = 0$$

$$\sum_{(i,j)\in e_1} e_{1,i}^{(j,1)} W_i^{(j)} = 0 \tag{2.28}$$

$$\sum_{(i,j)\in e_1} e_{6,i}^{(j,2)} W_i^{(j)} = 0 \tag{2.29}$$

and

$$\sum_{(i,j)\in e_1} e_{k,i}^{(j,4)} W_i^{(j)} = -1 \tag{2.30}$$

From eq. (2.28) and (2.29), one can obtain:

$$e_{1,1}^{(0,1)} W_1^{(0)} + e_{1,2}^{(0,1)} W_2^{(0)} + e_{1,3}^{(0,1)} W_3^{(0)} + e_{1,4}^{(0,1)} W_4^{(0)} + e_{1,5}^{(0,1)} W_5^{(0)} + e_{1,6}^{(0,1)} W_6^{(0)} = 0$$

$$e_{6,1}^{(0,2)} W_1^{(0)} + e_{6,2}^{(0,2)} W_2^{(0)} + e_{6,3}^{(0,2)} W_3^{(0)} + e_{6,4}^{(0,2)} W_4^{(0)} + e_{6,5}^{(0,2)} W_5^{(0)} + e_{6,6}^{(0,2)} W_6^{(0)} = 0$$

Hence:

$$\left.\begin{aligned} e_{1,2}^{(0,1)} W_2^{(0)} + e_{1,5}^{(0,1)} W_5^{(0)} &= -e_{1,3}^{(0,1)} W_3^{(0)} - e_{1,4}^{(0,1)} W_4^{(0)} \\ e_{6,2}^{(0,2)} W_2^{(0)} + e_{6,5}^{(0,2)} W_5^{(0)} &= -e_{6,3}^{(0,2)} W_3^{(0)} - e_{6,4}^{(0,2)} W_4^{(0)} \end{aligned}\right\} \tag{2.31}$$

Solving the linear system (2.31) in terms of $W_2^{(0)}$ and $W_5^{(0)}$, we get:

$$W_2^{(0)} = \frac{e_{1,5}^{(0,1)} e_{6,3}^{(0,2)} - e_{1,3}^{(0,1)} e_{6,5}^{(0,2)}}{e_{1,2}^{(0,1)} e_{6,5}^{(0,2)} - e_{1,5}^{(0,1)} e_{6,2}^{(0,2)}} W_3^{(0)} + \frac{e_{1,5}^{(0,1)} e_{6,4}^{(0,2)} - e_{1,4}^{(0,1)} e_{6,5}^{(0,2)}}{e_{1,2}^{(0,1)} e_{6,5}^{(0,2)} - e_{1,5}^{(0,1)} e_{6,2}^{(0,2)}} W_4^{(0)} \tag{2.32}$$

and

$$W_5^{(0)} = \frac{e_{6,3}^{(0,2)} e_{1,2}^{(0,1)} - e_{1,3}^{(0,1)} e_{6,2}^{(0,2)}}{e_{1,5}^{(0,1)} e_{6,2}^{(0,2)} - e_{1,2}^{(0,1)} e_{6,5}^{(0,2)}} W_3^{(0)} + \frac{e_{1,2}^{(0,1)} e_{6,4}^{(0,2)} - e_{1,4}^{(0,1)} e_{6,4}^{(0,2)}}{e_{1,5}^{(0,1)} e_{6,2}^{(0,2)} - e_{1,2}^{(0,1)} e_{6,5}^{(0,2)}} W_4^{(0)} \tag{2.33}$$

upon substituting $W_2^{(0)}$ and $W_5^{(0)}$ in equation (2.30), we shall get for $k = 3$ and 4:

$$e_{k,2}^{(0,4)} \left(\frac{e_{1,5}^{(0,1)} e_{6,3}^{(0,2)} - e_{1,3}^{(0,1)} e_{6,5}^{(0,2)}}{e_{1,2}^{(0,1)} e_{65}^{(0,2)} - e_{1,5}^{(0,1)} e_{6,2}^{(0,2)}} W_3^{(0)} + \frac{e_{1,5}^{(0,1)} e_{6,4}^{(0,2)} - e_{1,4}^{(0,1)} e_{6,5}^{(0,2)}}{e_{1,2}^{(0,1)} e_{6,5}^{(0,2)} - e_{1,5}^{(0,1)} e_{6,2}^{(0,2)}} W_4^{(0)} \right) +$$

$$e_{k,3}^{(0,4)} W_3^{(0)} + e_{k,4}^{(0,4)} W_4^{(0)} +$$

$$e_{k,5}^{(0,4)} \left(\frac{e_{6,3}^{(0,2)} e_{1,2}^{(0,1)} - e_{1,3}^{(0,1)} e_{6,2}^{(0,2)}}{e_{1,5}^{(0,1)} e_{6,2}^{(0,2)} - e_{1,2}^{(0,1)} e_{6,5}^{(0,2)}} W_3^{(0)} + \frac{e_{1,2}^{(0,1)} e_{6,4}^{(0,2)} - e_{1,4}^{(0,1)} e_{6,4}^{(0,2)}}{e_{1,5}^{(0,1)} e_{6,2}^{(0,2)} - e_{1,2}^{(0,1)} e_{6,5}^{(0,2)}} W_4^{(0)} \right) = -1$$

From the above one can get $W_3^{(0)}$ and $W_4^{(0)}$ finally from (2.31) we get $W_2^{(0)}$ and $W_5^{(0)}$.

Following table (2.1) which represent a comparison of the exact solution with the G-spline interpolation-based differential quadrature.

Table (2.1)
Comparison of the approximate solution of example (2.1) (case 1) with the exact solution.

x	W(Exact)	W(DQM)	Error (%)
0	0	0	0
0.09955	-4.828E-004	-1.276E-003	-7.932E-004
0.3456	-3.759E-003	-7.901E-004	-2.969E-003
0.6546	-5.213E-003	-7.844E-004	-4.424E-003
0.9046	-1.936E-003	-4.211E-004	-1.516E-003
1	0	0	0

Case 2:

In this case we shall take a 5-poised Hermite-Birkhoff problem with

$$e_2 = \{(1,0),(2,0),(3,0),(4,1),(5,1),(6,0)\}$$

we shall seek

$S_5(x) \in S(E^*, x_1, x_2, x_3, x_4, x_5, x_6)$ where

$$E = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and for which

$$S_5^{(j)}(x_i) = W_i^{(j)}, (i, j) \in e_2$$

Similarly as we do in the previous case thus we shall have:

$$e_{1,1}^{(0,1)} W_1^{(0)} + e_{1,2}^{(0,1)} W_2^{(0)} + e_{1,3}^{(0,1)} W_3^{(0)} + e_{1,4}^{(1,1)} W_4^{(1)} + e_{1,5}^{(1,1)} W_5^{(1)} + e_{1,6}^{(0,1)} W_6^{(0)} = 0$$

$$e_{6,1}^{(0,2)} W_1^{(0)} + e_{6,2}^{(0,2)} W_2^{(0)} + e_{6,3}^{(0,2)} W_3^{(0)} + e_{6,4}^{(1,2)} W_4^{(1)} + e_{6,5}^{(1,2)} W_5^{(1)} + e_{6,6}^{(0,2)} W_6^{(0)} = 0$$

Hence

$$\left.\begin{aligned} e_{1,2}^{(0,1)} W_2^{(0)} + e_{1,5}^{(1,1)} W_5^{(1)} &= -e_{1,3}^{(0,1)} W_3^{(0)} - e_{1,4}^{(1,1)} W_4^{(1)} \\ e_{6,2}^{(0,2)} W_2^{(0)} + e_{6,5}^{(1,2)} W_5^{(1)} &= -e_{6,3}^{(0,2)} W_3^{(0)} - e_{6,4}^{(1,2)} W_4^{(1)} \end{aligned}\right\} \qquad (2.34)$$

Solving equation (2.34) in terms of $W_2^{(0)}$ and $W_5^{(1)}$ we get

$$W_2^{(0)} = \frac{e_{1,5}^{(1,1)} e_{6,3}^{(0,2)} - e_{1,3}^{(0,1)} e_{6,5}^{(1,2)}}{e_{1,2}^{(0,1)} e_{6,5}^{(1,2)} - e_{1,5}^{(1,1)} e_{6,2}^{(0,2)}} W_3^{(0)} + \frac{e_{1,5}^{(1,1)} e_{6,4}^{(1,2)} - e_{1,4}^{(1,1)} e_{6,5}^{(1,2)}}{e_{1,2}^{(0,1)} e_{6,5}^{(1,2)} - e_{1,5}^{(1,1)} e_{6,2}^{(0,2)}} W_4^{(1)} \qquad (2.35)$$

and

$$W_5^{(1)} = \frac{e_{6,3}^{(0,2)} e_{1,2}^{(0,1)} - e_{1,3}^{(0,1)} e_{6,2}^{(0,2)}}{e_{1,5}^{(1,1)} e_{6,2}^{(0,2)} - e_{1,2}^{(0,1)} e_{6,5}^{(1,2)}} W_3^{(0)} + \frac{e_{1,2}^{(0,1)} e_{6,4}^{(1,2)} - e_{1,4}^{(1,1)} e_{6,5}^{(1,2)}}{e_{1,5}^{(1,1)} e_{6,2}^{(0,2)} - e_{1,2}^{(0,1)} e_{6,5}^{(1,2)}} W_4^{(1)} \tag{2.36}$$

upon substituting $W_2^{(0)}$ and $W_5^{(1)}$ in equation $\sum_{(i,j)\in e} e_{k,i}^{(j,4)} W_i^{(j)} = -1$ we shall get for $k = 3$ and 4:

$$e_{k,2}^{(0,4)} \left(\frac{e_{1,5}^{(1,1)} e_{6,3}^{(0,2)} - e_{1,3}^{(0,1)} e_{6,5}^{(1,2)}}{e_{1,2}^{(0,1)} e_{6,5}^{(1,2)} - e_{1,5}^{(1,1)} e_{6,2}^{(0,2)}} W_3^{(0)} + \frac{e_{1,5}^{(1,1)} e_{6,4}^{(1,2)} - e_{1,4}^{(1,1)} e_{6,5}^{(1,2)}}{e_{1,2}^{(0,1)} e_{6,5}^{(1,2)} - e_{1,5}^{(1,1)} e_{6,2}^{(0,2)}} W_4^{(1)} \right) +$$
$$e_{k,3}^{(0,4)} W_3^{(0)} + e_{k,4}^{(1,4)} W_4^{(1)} +$$
$$e_{k,5}^{(1,4)} \left(\frac{e_{6,3}^{(0,2)} e_{1,2}^{(0,1)} - e_{1,3}^{(0,1)} e_{6,2}^{(0,2)}}{e_{1,5}^{(1,1)} e_{6,2}^{(0,2)} - e_{1,2}^{(0,1)} e_{6,5}^{(1,2)}} W_3^{(0)} + \frac{e_{1,2}^{(0,1)} e_{6,4}^{(1,2)} - e_{1,4}^{(1,1)} e_{6,5}^{(1,2)}}{e_{1,5}^{(1,1)} e_{6,2}^{(0,2)} - e_{1,2}^{(0,1)} e_{6,5}^{(1,2)}} W_4^{(1)} \right) = -1$$

Solving the above equation in terms of $W_3^{(0)}$ and $W_4^{(1)}$ then substituting them into equation (2.34) in order to get $W_2^{(0)}$ and $W_5^{(1)}$.

Following table (2.2) represent a comparison of the exact solution with the G-spine interpolation- based differential quadrature.

Table (2.2)
Comparison of the approximate solution of example (2.1) (case 2)with the exact solution

x	W(Exact)	W(DQM)	Error (%)
0	0	0	0
0.09955	-4.828E-004	-1.720E-003	-1.227E-003
0.3456	-3.759E-003	-3.842E-003	-8.299E-004
0.6546	-5.213E-003	- 6.742E-004	-4.548E-003
0.9046	-1.936E-003	-1.711E-004	-1.764E-003
1	0	0	0

It is noted that all calculations are performed by using Microsoft MATLAB see programs 4 and 5 Appendix A and the fundamental functions $L_{10}(x), L_{20}(x), L_{30}(x), L_{40}(x), L_{50}(x)$ and $L_{60}(x)$ are given in Appendix B in addition to the functions $L_{10}(x), L_{20}(x), L_{30}(x), L_{41}(x)$, $L_{51}(x)$ and $L_{60}(x)$.

2.5 Applying Boundary Conditions

In this section, a treatment for the boundary conditions using the G-spline interpolation formula-based differential quadrature will be given. The direct substitution of boundary conditions into the discrete governing equations will be considered in order to find the numerical solution of the free vibration beam problem that is appeared in eq. (1.77).

2.5.1 Direct Substitution of Boundary Conditions Using G – Spline Interpolation:

For a vibration of a uniform beam problem the governing differential equation as we mentioned previously in section (1.5) is:

$$s(X)\frac{d^4W}{dX^4}+2\frac{d\,s(X)}{dX}\frac{d^3W}{dX^3}+\frac{d^2s(X)}{dX^2}\frac{d^2W}{dX^2}-\Omega^2W=0 \qquad (2.37)$$

Also, for a well-posed problem, it requires four boundary conditions. These can be obtained by specifying two boundary conditions at the end point $X=0$, and another two boundary conditions at the end point $X=1$. In this subsection next, two types of boundary conditions will be considered which are (1.79) and (1.80).

A combination of these two boundary conditions is used in the numerical experiment in the next section.

The selection of locations of the sampling points plays an important role in the accuracy of the solution of the differential equations. Using uniform grids can be considered to be convenient and easy selection method. Quite frequently the DQM delivers more accurate solution using the so called Chebyshev Gauss Lobatto points given by (1.82).

For the approximate solution of the free vibration analysis of a uniform beam (2.37) with the boundary conditions given by eqs. (1.79)

and (1.80) using the G-spline interpolation formula-based DQM, we first discritize the interval [0, 1] such that $0 = x_1 < x_2 < \cdots < x_N = 1$

The idea of the followed approach is similar to the given in subsection (1.5.3.3). The descretized derivative conditions at the two ends are then combined to give the solutions $W_2^{(j_2)}$ and $W_{N-1}^{(j_5)}$, where j_2 and j_5 represent the minimum order derivative of W at x_2 and x_{N-1} respectively. The expressions for $W_2^{(j_2)}$ and $W_{N-1}^{(j_5)}$ are then substituted into the discrete governing equation

$$s''(X_i)\sum_{(i,j)\in e} e_{k,i}^{(j,2)} W_i^{(j)} + 2s'(X_i)\sum_{(i,j)\in e} e_{k,i}^{(j,3)} W_i^{(j)} + s(X_i)\sum_{(i,j)\in e} e_{k,i}^{(j,4)} W_i^{(j)} = \Omega^2 W_i^{(j)} \qquad (2.38)$$

With the interior points $3 \le i \le N-2$.

For any combination of the clamped and simply supported conditions at the two ends, the discrete boundary conditions using the DQM may be written as

$$W_1^{(0)} = 0 \qquad (2.39a)$$

$$\sum_{(i,j)\in e} e_{1,i}^{(j,n_0)} W_i^{(j)} = 0 \qquad (2.39b)$$

$$W_N^{(0)} = 0 \qquad (2.39c)$$

$$\sum_{(i,j)\in e} e_{N,i}^{(j,n_1)} W_i^{(j)} = 0 \qquad (2.39d)$$

Where n_0 and n_1 may be taken as 1 or 2.

We shall treat only the following two sets of boundary conditions

$$\left.\begin{array}{l} n_0 = 1, n_1 = 2 \ldots \text{clamped} - \text{simply supported} \\ n_0 = 2, n_1 = 2 \ldots \text{simply supported} - \text{ simply supported} \end{array}\right\} \qquad (2.40)$$

Equation (2.39a) and (2.39c) can be easily substituted into equation (2.38) and it is clear that this is not the case for eq. (2.39b) and (2.39d).

However one can couple these two equations together to give the solutions $W_2^{(j_2)}$ and $W_{N-1}^{(j_5)}$ as:

$$W_2^{(j_2)} = \frac{1}{AXN} \sum_{(i,j)\in e^*} AXK1 \cdot W_i^{(j)} \qquad (2.41a)$$

$$W_{N-1}^{(j_5)} = \frac{1}{AXN} \sum_{(i,j)\in e^*} AXKN \cdot W_i^{(j)} \qquad (2.41b)$$

Where:

$$e^* = e / \{(2, j_2), (N-1, j_5)\}$$

Hence, for $(i,j) \in e^*$, we have

$$AXK1 = e_{1,i}^{(j,n_0)} e_{N,N-1}^{(j,n_1)} - e_{1,N-1}^{(j,n_0)} e_{N,i}^{(j,n_1)}$$
$$AXKN = e_{1,2}^{(j,n_0)} e_{N,i}^{(j,n_1)} - e_{1,i}^{(j,n_0)} e_{N,2}^{(j,n_1)}$$
$$AXN = e_{N,2}^{(j,n_1)} e_{1,N-1}^{(j,n_0)} - e_{1,2}^{(j,n_0)} e_{N,N-1}^{(j,n_1)}$$

According to eq. (2.41a) and (2.41b) $W_2^{(j_2)}$ and $W_{N-1}^{(j_5)}$ are expressed in terms of $W_i^{(j)}$, $(i,j) \in e^*$ and can be easily substituted into eq. (2.38) .

In order to find the values of $W_i^{(j)}$ for $(i,j) \in e^*$ the discretized governening equation (2.38) has to be applied at the interior ordered Paris $(i,j) \in e^*$. Substituting eqs. (2.39a), (2.39c), (2.41a) and (2.41b) into eq. (2.38) gives

$$s''(X_i) \sum_{(i,j)\in e^*} C_1 W_i^{(j)} + 2s'(X_i) \sum_{(i,j)\in e^*} C_2 W_i^{(j)} +$$
$$s(X_i) \sum_{(i,j)\in e^*} C_3 W_i^{(j)} = \Omega^2 W_i^{(j)}, (i,j) \in e^* \qquad (2.42)$$

Where

$$C_1 = e_{k,i}^{(j,2)} - \frac{e_{k,2}^{(j,2)} AXK1 + e_{k,N-1}^{(j,2)} AXKN}{AXN}$$
$$C_2 = e_{k,i}^{(j,3)} - \frac{e_{k,2}^{(j,3)} AXK1 + e_{k,N-1}^{(j,3)} AXKN}{AXN}$$

$$C_3 = e_{k,i}^{(4)} - \frac{e_{k,2}^{(j,4)} AXK1 + e_{k,N-1}^{(j,4)} AXKN}{AXN}$$

It is noted that the eq. (2.42) has (N-4) equations with (N-4) unknowns which can be written in matrix form, as:

$$[A]\{W\} = \Omega^2 \{W\} \qquad (2.43)$$

Where Ω^2 represent the eigenvalues of the above system

2.6 Illustrative Example: Free Vibration Analysis of a Uniform Beam Using G-Spline Interpolation-Based DQM:

The free vibration analysis of a uniform beam as given in equation (2.38) with $(s(x) = 1)$ will be treated in this section, in which two combinations of boundary conditions given by (2.40) are considered.

Applying the approach given in subsection (2.5.1) by considered two different sets of HB-problems in the first one the usual differential quadrature occurs since we assume that $N = K + 1$ and $e = \{(i,0),\ i = 1,2,...,k+1\}$ the HB-problem can be reduced to a Lagrange problem. While the second set represents that our approach can be considered as a generalization to the usual differential quadrature and this can be illustrated by the following two cases:

Case1:

To construct the approximate solution via G-spline-based DQM an m-poised HB-problem must chosen. In this case we shall take a 5-poised HB- problem with

$$e_1 = \{(1,0),(2,0),(3,0),(4,0),(5,0),(6,0)\}$$

we shall seek

$S_5(x) \in S(E^*, x_1, x_2, x_3, x_4, x_5, x_6)$ where

$$E = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

and for which

$$S_5(x_i) = W_i^{(j)}, (i, j) \in e_1$$

First applying the DQM for the simply supported-simply supported boundary conditions using the node points (1.71) with $N = 6$ thus we have

$$W_1^{(0)} = 0 \tag{2.44a}$$

$$\sum_{(i,0)\in e_1} e_{1,i}^{(0,2)} \cdot W_i^{(0)} = 0 \tag{2.44b}$$

$$W_6^{(0)} = 0 \tag{2.44c}$$

$$\sum_{(i,0)\in e_1} e_{6,i}^{(0,2)} \cdot W_i^{(0)} = 0 \tag{2.44d}$$

And the governing equation therefore becomes

$$\sum_{(i,0)\in e_1} e_{ki}^{(0,4)} W_i^{(0)} = \Omega^2 W_i^{(0)}, (i,0) \in e_1 \tag{2.45}$$

Equation (2.44a) and (2.44c) can be easily substituted into equation (2.45).

From equations (2.44b) and (2.44d) one can get $W_2^{(0)}$ and $W_5^{(0)}$ as follows:

$$W_2^{(0)} = \frac{1}{AXN} \sum_{(i,0)\in e_1^*} AXK1 \cdot W_i^{(0)} \tag{2.46a}$$

$$W_5^{(0)} = \frac{1}{AXN} \sum_{(i,0)\in e_1^*} AXKN \cdot W_i^{(0)} \tag{2.46b}$$

Where

$$e_1^* = \{(3,0),(4,0)\}$$

$$\left.\begin{aligned} AXK1 &= e_{1,i}^{(0,2)}\ e_{6,5}^{(0,2)} - e_{1,5}^{(0,2)}\ e_{6,i}^{(0,2)} \\ AXKN &= e_{1,2}^{(0,2)}\ e_{6,i}^{(0,2)} - e_{1,i}^{(0,2)}\ e_{6,2}^{(0,2)} \\ AXN &= e_{6,2}^{(0,2)}\ e_{1,5}^{(0,2)} - e_{1,2}^{(0,2)}\ e_{6,5}^{(0,2)} \end{aligned}\right\} for\ (i,0) \in e_1^*$$

Thus we have $W_2^{(0)}$ and $W_5^{(0)}$ in terms of $W_3^{(0)}$ and $W_4^{(0)}$ and after substituting into eq. (2.45) hence we get two equations in two unknowns written by eq. (2.43).

The natural frequency of the simply supported - simply supported given according to this case in table (2.3).

Secondly applying the DQM for the clamped – simply supported boundary conditions will yields to

$$W_1^{(0)} = 0 \qquad (2.47a)$$

$$\sum_{(i,0)\in e_1} e_{1,i}^{(0,1)} W_i^{(0)} = 0 \qquad (2.47b)$$

$$W_6^{(0)} = 0 \qquad (2.47c)$$

$$\sum_{(i,0)\in e_1} e_{6,i}^{(0,2)} W_i^{(0)} = 0 \qquad (2.47d)$$

And the governing equation still as given in (2.45)

Similarly eqs. (2.47a) and (2.47c) can be easily substituted into eq. (2.45).

For eqs. (2.47b) and (2.47d) we can get $W_2^{(0)}$ and $W_5^{(0)}$ in terms of $W_3^{(0)}$ and $W_4^{(0)}$ as given in (2.46) with

$$\left.\begin{aligned} AXK1 &= e_{1,i}^{(0,1)}\ e_{6,5}^{(0,2)} - e_{1,5}^{(0,1)}\ e_{6,i}^{(0,2)} \\ AXKN &= e_{1,2}^{(0,1)}\ e_{6,i}^{(0,2)} - e_{1,i}^{(0,1)}\ e_{6,2}^{(0,2)} \\ AXN &= e_{6,2}^{(0,1)}\ e_{1,5}^{(0,2)} - e_{1,2}^{(0,1)}\ e_{6,5}^{(0,2)} \end{aligned}\right\} for\ (i,0) \in e_1^*$$

Similarly substituting $W_2^{(0)}$ and $W_5^{(0)}$ into equation (2.45) similarly we get two equations into two unknowns written by equation (2.43).

The natural low frequency of the clamped - simply supported will be given for this case in table (2.3).

It is noted that all the calculations are performed by a computer programs written by MATLAB see programs 6 and 7 Appendix A.

Comparison of the low natural frequency of a uniform beam and the exact solution is given in table (2.3). Where the exact solution is given in Blevins, [9].

Table (2.3)

Comparison of low natural frequency (Ω) of a uniform beam using G-spline interpolation-based DQM with the exact solution (case 1)

Boundary Conditions	G-spline	Exact	Error%
SS-SS	9.8669	9.8696	-0.0027
C-SS	15.4682	15.4182	0.05

Case2:

In this case we shall take a 5-poised HB- problem will be take with

$$e_2 = \{(1,0),(2,0),(3,0),(4,1),(5,1),(6,0)\}$$

we shall seek

$$S_5(x) \in S(E^*, x_1, x_2, x_3, x_4, x_5, x_6)$$ where

$$E = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and for which

$$S_5^{(j)}(x_i) = W_i^{(j)}, \ (i, j) \in e_2$$

Similarly applying the DQM for the simply supported-simply supported boundary conditions using the node points (1.71) with $N = 6$ thus we have

$$W_1^{(0)} = 0 \tag{2.48a}$$

$$\sum_{(i,j)\in e_2} e_{1,i}^{(j,2)} W_i^{(j)} = 0 \tag{2.48b}$$

$$W_6^{(0)} = 0 \tag{2.48c}$$

$$\sum_{(i,j)\in e_2} e_{6,i}^{(j,2)} W_i^{(j)} = 0 \tag{2.48d}$$

And the governing equation therefore becomes

$$\sum_{(i,j)\in e_2} e_{k,i}^{(j,4)} W_i^{(j)} = \Omega^2 W_i^{(j)}, (i,j) \in e_2 \tag{2.49}$$

Equation (2.48a) and (2.48c) can be easily substituted into eq. (2.49). For eqs. (2.48b) and (2.48d) we can get $W_2^{(0)}$ and $W_5^{(1)}$ as follows:

$$W_2^{(0)} = \frac{1}{AXN} \sum_{(i,j)\in e_2^*} AXK1 \cdot W_i^{(j)} \tag{2.50a}$$

$$W_5^{(1)} = \frac{1}{AXN} \sum_{(i,j)\in e_2^*} AXKN \cdot W_i^{(j)} \tag{2.50b}$$

Where

$$e_2^* = \{(3,0),(4,1)\}$$

$$\left.\begin{aligned} AXK1 &= e_{1,i}^{(j,2)} e_{6,5}^{(1,2)} - e_{1,5}^{(1,2)} e_{6,i}^{(j,2)} \\ AXKN &= e_{1,2}^{(0,2)} e_{6,i}^{(j,2)} - e_{1,i}^{(j,2)} e_{6,2}^{(0,2)} \\ AXN &= e_{6,2}^{(0,2)} e_{1,5}^{(1,2)} - e_{1,2}^{(0,2)} e_{6,5}^{(1,2)} \end{aligned}\right\} for\ (i,j) \in e_2^*$$

Thus we have $W_2^{(0)}$ and $W_5^{(1)}$ in terms of $W_3^{(0)}$ and $W_4^{(1)}$ and after substituting into eq. (2.49) hence we get two equations in two unknowns written by eq. (2.43).

The low natural frequency of the simply supported - simply supported is given in table (2.4).

Secondly Applying the DQM for the clamped – simply supported boundary conditions therefore give:

$$W_1^{(0)} = 0 \qquad (2.51a)$$

$$\sum_{(i,j)\in e_1} e_{1,i}^{(j,1)} W_i^{(j)} = 0 \qquad (2.51b)$$

$$W_6^{(0)} = 0 \qquad (2.51c)$$

$$\sum_{(i,j)\in e_1} e_{6,i}^{(j,2)} W_i^{(j)} = 0 \qquad (2.51d)$$

The governing equation will be in the form of eq. (2.49). Eqs. (2.51a) and (2.51c) can be easily substituted into eq. (2.49).

From equations (2.51b) and (2.51d) one can find $W_2^{(0)}$ and $W_5^{(1)}$ in terms of $W_3^{(0)}$ and $W_4^{(1)}$ as given in equations (2.49) with

$$\left.\begin{aligned} AXK1 &= e_{1,i}^{(j,1)}\ e_{6,5}^{(1,2)} - e_{1,5}^{(1,1)}\ e_{6,i}^{(j,2)} \\ AXKN &= e_{1,2}^{(0,1)}\ e_{6,i}^{(j,2)} - e_{1,i}^{(j,1)}\ e_{6,2}^{(0,2)} \\ AXN &= e_{6,2}^{(0,1)}\ e_{1,5}^{(1,2)} - e_{1,2}^{(0,1)}\ e_{6,5}^{(1,2)} \end{aligned}\right\} for\ (i,j)\in e_2^*$$

Substituting $W_2^{(0)}$ and $W_5^{(1)}$ into (2.49) then the eigenvalue equations written by (2.43) will be found.

The low natural frequency of the clamped - simply supported will be given in table (2.4).

It is noted that all the calculations are performed by a computer programs written by MATLAB see Appendix A programs 8 and 9.

Following table (2.4) represent a comparison of the low natural frequency of a uniform beam with the exact solution.

Table (2.4)

Comparison of low natural frequency (Ω) of a uniform beam using G-spline interpolation-based DQM with the exact solution (case 2)

Boundary Conditions	G-spline	Exact	Error%
SS-SS	9.5773	9.8696	-0.1923
C-SS	15.7376	15.4182	0.3194

Chapter Three

Numerical Solution of Thin Plates problem Using G-Spline-Based Differential Quadrature Method

3.1 Introduction

This chapter is devoted to find the numerical solution of a thin plate problem using G-spline-based DQM which consists of four sections. In section 3.2 the differential quadrature analysis of a thin plate problem will be presented, while section 3.3 is related to the direct substitution of the boundary conditions into the discrete governing equation. Finally a numerical example will be given in section 3.4.

3.2 Differential Quadrature Analysis of Thin Plates

In this section, the application of the G-spline interpolation-based DQM will be illustrated in the structural and vibration analysis of thin plates problem. In which, only the rectangular plate will be considered.

3.2.1 The Governing Equations and Boundary Conditions

Deflection, free vibration and buckling are two typical problems for a plate. The non-dimensional governing equations for these two cases can be written as [28]:

Plate deflection

$$\frac{\partial^4 W}{\partial X^4} + 2\lambda^2 \frac{\partial^4 W}{\partial X^2 \partial Y^2} + \lambda^4 \frac{\partial^4 W}{\partial Y^4} = \frac{a^4 q(X,Y)}{D} \tag{3.1}$$

transverse vibration of thin, isotropic plates

$$\frac{\partial^4 W}{\partial X^4} + 2\lambda^2 \frac{\partial^4 W}{\partial X^2 \partial Y^2} + \lambda^4 \frac{\partial^4 W}{\partial Y^4} = \Omega^2 W \tag{3.2}$$

buckling of a plate under uniaxial compression

$$\frac{\partial^4 W}{\partial X^4} + 2\lambda^2 \frac{\partial^4 W}{\partial X^2 \partial Y^2} + \lambda^4 \frac{\partial^4 W}{\partial Y^4} = \frac{N_x a^2}{D} \frac{\partial^2 W}{\partial X^2} \tag{3.3}$$

Where W is the dimensionless mode shape function, $q(X,Y)$ is the external distributed load, Ω is the dimensionless frequency, $X = x/a$ and $Y = y/b$ are dimensionless coordinates, a and b are the

lengths of the plate edges, $\lambda = a/b$ is the aspect ratio, and N_x is the uniaxial load. Furthe more, $\Omega = \omega a^2\sqrt{\rho/D}$, where ω is the dimensionless circular frequency, $D = E\,h^3/[12(1-\nu^2)]$ is the flexural rigidity, E, v, ρ and h are Young's modulus, Poisson's ratio, the density of the plate material, and the plate thickness, respectively. It should be mentioned that the above equations do not cover all the cases. For example, for free vibration of the anisotropic plats, eq. (3.2) has to be changed to include more terms. Eq. (3.3) can be modified to consider buckling under different compressions.

The governing equation for a thin plate is a 4^{th} order partial differential equation with respect to X and Y. It requires two boundary conditions at each edge. There are three basic boundary conditions. For free vibration analysis, these boundary conditions are

Simply supported edge (SS)

$$W\big|_{X=0} = 0, \qquad \left.\frac{\partial^2 W}{\partial X^2}\right|_{X=1} = 0 \tag{3.4a}$$

$$W\big|_{Y=0} = 0, \qquad \left.\frac{\partial^2 W}{\partial Y^2}\right|_{Y=1} = 0 \tag{3.4b}$$

Clamped edge (C)

$$W\big|_{X=0} = 0, \qquad \left.\frac{\partial W}{\partial X}\right|_{X=1} = 0 \tag{3.5a}$$

$$W\big|_{Y=0} = 0, \qquad \left.\frac{\partial W}{\partial Y}\right|_{Y=1} = 0 \tag{3.5b}$$

Free edge (F)

$$\left.\frac{\partial^2 W}{\partial X^2}+\nu\lambda^2\frac{\partial^2 W}{\partial Y^2}\right|_{X=0}=0,\quad \left.\frac{\partial^3 W}{\partial X^3}+(2-\nu)\lambda^2\frac{\partial^3 W}{\partial X\,\partial Y^2}\right|_{X=1}=0 \qquad (3.6a)$$

$$\left.\lambda^2\frac{\partial^2 W}{\partial Y^2}+\nu\frac{\partial^2 W}{\partial X^2}\right|_{Y=0}=0,\quad \left.\lambda^2\frac{\partial^3 W}{\partial Y^3}+(2-\nu)\frac{\partial^3 W}{\partial X^2\partial Y}\right|_{Y=1}=0 \qquad (3.6b)$$

and

$$\frac{\partial^2 W}{\partial X\,\partial Y}=0 \qquad (3.6c)$$

at the corner of two adjacent free edges.

3.2.2 Numerical Discretization of the Problem

The computational domain of a rectangular plate is $0\le X\le 1, 0\le Y\le 1$. For numerical computation, we need to perform a mesh generation first. As for the case of the plate, the mesh generation in the X and Y directions are given by:

$$X_i=\tfrac{1}{2}\left[1-\cos\left(\frac{i-1}{N-1}\pi\right)\right],\ i=1,2,...,N \qquad (3.7a)$$

$$Y_r=\tfrac{1}{2}\left[1-\cos\left(\frac{r-1}{M-1}\pi\right)\right],\ r=1,2,...,M \qquad (3.7b)$$

Where N and M are the number of the grid points in the X and Y directions respectively. With the coordinates of the mesh points given by eq.(3.7), the G-spline-based DQ weighting coefficients can be easily computed. These weighting coefficients can be used to discretize eqs. (3.1) and (3.3). Let $e_{k,i}^{(j,n)}$ be the DQ weighting coefficients of the n−th order derivative in the X direction, and $\bar{e}_{h,r}^{(s,m)}$ be the DQ weighting

coefficients of the m−th order derivative in the Y direction. Using the DQM, eq. (3.1) may be discretized as

$$\sum_{(i,j)\in e_1} e_{k,i}^{(j,4)} W_{i,r}^{(j,s)} + 2\lambda^2 \sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} e_{k,i}^{(j,2)} \bar{e}_{h,r}^{(s,2)} W_{i,r}^{(j,s)} + \lambda^4 \sum_{(r,s)\in e_2} \bar{e}_{h,r}^{(s,4)} W_{i,r}^{(j,s)}$$
$$= \frac{a^4 q_{i,r}}{D}, (i,j)\in e_1, (r,s)\in e_2, k = 1,2,...,N,\ h = 1,2,...,M \qquad (3.8)$$

and eq. (3.2) is discretized as

$$\sum_{(i,j)\in e_1} e_{k,i}^{(j,4)} W_{i,r}^{(j,s)} + 2\lambda^2 \sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} e_{k,i}^{(j,2)} \bar{e}_{h,r}^{(s,2)} W_{i,r}^{(j,s)} + \lambda^4 \sum_{(r,s)\in e_2} \bar{e}_{h,r}^{(s,4)} W_{i,r}^{(j,s)}$$
$$= \Omega^2 W_{i,r}^{(i,s)}, (i,j)\in e_1, (r,s)\in e_2, k = 1,2,...,N,\ h = 1,2,...,M \qquad (3.9)$$

In a similar manner, eq. (3.3) is discretized as

$$\sum_{(i,j)\in e_1} e_{k,i}^{(j,4)} W_{i,r}^{(j,s)} + 2\lambda^2 \sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} e_{k,i}^{(j,2)} \bar{e}_{h,r}^{(s,2)} W_{i,r}^{(j,s)} + \lambda^4 \sum_{(r,s)\in e_2} \bar{e}_{h,r}^{(s,4)} W_{i,r}^{(j,s)}$$
$$= \frac{a^2 N_x}{D} \sum_{(i,j)\in e_1} e_{k,i}^{(j,2)} W_{i,r}^{(j,s)}, (i,j)\in e_1, (r,s)\in e_2\ k = 1,2,...,N,\ h = 1,2,...,M \qquad (3.10)$$

Where $W_{i,r}^{(j,s)} = \left.\frac{\partial^{j+s} W}{\partial^{j+s}}\right|_{(x_i, y_r)}$ is $(j+s)^{th}$ derivative at the grid point (X_i, Y_r). After implementation of the boundary conditions, eq. (3.8) can be written in matrix form as it is given in eq (1.86), similarly, eq. (3.9) can be given in the same form as eq. (1.87), and eq. (3.10) can also be given into a similar form as in eq. (1.88). The solution techniques for the three matrix forms have been described in chapter two.

3.3 Direct Substitution of Boundary Conditions into Discrete Governing Equation

The idea behind this approach is the same as that for the beam problem. It was presented by Shu and Du [24], to implement the simply

supported and clamped conditions. The derivatives in the boundary condition are also discretized by the DQM. The discrete form of any combination of the clamped and simply supported conditions may be given as

$$W_{1,r}^{(j,s)} = 0, W_{N,r}^{(j,s)} = 0, W_{i,1}^{(j,s)} = 0, W_{i,M}^{(j,s)} = 0 \qquad (3.11a)$$

$$\sum_{(i,j)\in e_1} e_{1,i}^{(j,n_0)} W_{i,r}^{(j,s)} = 0 \ \ at \ \ X = 0 \qquad (3.11b)$$

$$\sum_{(i,j)\in e_1} e_{N,i}^{(j,n_1)} W_{i,r}^{(j,s)} = 0 \ \ at \ \ X = 1 \qquad (3.11c)$$

$$\sum_{(r,s)\in e_2} \bar{e}_{1,r}^{(s,m_0)} W_{i,r}^{(j,s)} = 0 \ \ at \ Y = 0 \qquad (3.11d)$$

$$\sum_{(r,s)\in e_2} \bar{e}_{M,r}^{(s,m_1)} W_{i,r}^{(j,s)} = 0 \ \ at \ Y = 1 \qquad (3.11e)$$

where n_o, n_1, m_0 and m_1 are taken as either 1or 2, where 1 is used for the clamped edge condition and 2 is used for the simply supported edge condition. n_o, n_1, m_0 and m_1 correspond to the edges of X = 0, X = 1, Y = 0, Y = 1, respectively. It is noted that eq. (3.11a) corresponds to the Dirichlet boundary condition at the four edge of the plate, and eqs. (3.11b), (3.11c), (3.11d) and (3.11e) result from the derivative boundary conditions. Obviously, eq. (3.11a) can be easily substituted into Equation (3.9). However, eqs. (3.11b), (3.11c), (3.11d) and (3.11e) cannot be directly substituted into eq. (3.9). This difficulty can be easily overcome. Using the same procedure as for the beam, eqs. (3.11b), (3.11c) can be coupled to give two solution $W_{2,r}^{(j_1,s)}$ and $W_{N-1,r}^{(j_2,s)}$, where j_1 and j_2 represent the minimum partial order derivative of W with respect to X at X_2 and X_{N-1} respectively, which are located at the grid points shown by the symbol ○ in Fig. (3.1)

$$W_{2,r}^{(j_1,s)} = \frac{1}{AXN} \sum_{(i,j)\in e_1^*} AXK1 \cdot W_{i,r}^{(j,s)} \qquad (3.12a)$$

$$W_{N-1,r}^{(j_2,s)} = \frac{1}{AXN} \sum_{(i,j)\in e_1^*} AXKN \cdot W_{i,r}^{(j,s)} \qquad (3.12b)$$

$$for\ r = 3,4,...,M-2,\ e_1^* = e_1 \setminus \{(2,j_1),(N-1,j_2)\}$$

Where

for $(i,j) \in e_1^*$

$$AXN = e_{N,2}^{(j,n_1)} e_{1,N-1}^{(j,n_0)} - e_{1,2}^{(j,n_0)} e_{N,N-1}^{(j,n_1)}$$
$$AXK1 = e_{1,i}^{(j,n_0)} e_{N,N-1}^{(j,n_1)} - e_{1,N-1}^{(j,n_0)} e_{N,i}^{(j,n_1)}$$
$$AXKN = e_{1,2}^{(j,n_0)} e_{N,i}^{(j,n_1)} - e_{1,i}^{(j,n_0)} e_{N,2}^{(j,n_1)}$$

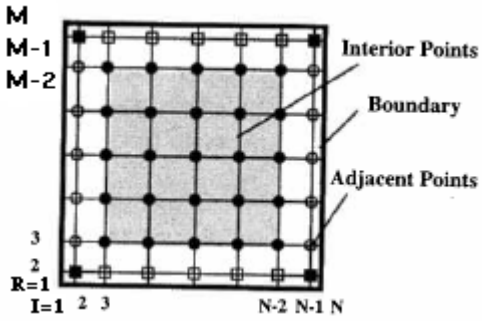

Figure (3.1) Illustration of interior and adjacent points for a rectangular plate

Similarly, eqs. (3.11d) and (3.11e) can be coupled to give two solutions $W_{i,2}^{(j,s_1)}$ and $W_{i,M-1}^{(j,s_2)}$, where s_1 and s_2 represent the minimum partial order derivative of W with respect to Y at Y_2 and Y_{M-1} respectively. Which are located at the grid points shown by the symbol □ in Fig. (3.1),

$$W_{i,2}^{(j,s_1)} = \frac{1}{AYM} \sum_{(r,s)\in e_2^*} AYK1 \cdot W_{i,r}^{(r,s)} \qquad (3.13a)$$

$$W_{i,M-1}^{(j,s_2)} = \frac{1}{AYM} \sum_{(r,s)\in e_2^*} AYKM \cdot W_{i,r}^{(r,s)} \qquad (3.13b)$$

$$for\ i = 3,4,...N-2,\ e_2^* = e_2 \setminus \{(2,s_1),(M-1,s_2)\}$$

where

for $(r,s)\in e_2^*$

$$AYN = e_{M,2}^{-(s,m_1)-(s,m_0)} e_{1,M-1} - e_{1,2}^{-(s,m_0)-(s,m_1)} e_{M,M-1}$$
$$AYK1 = e_{1,r}^{-(s,m_0)-(s,m_1)} e_{M,M-1} - e_{1,M-1}^{-(s,m_0)-(s,m_1)} e_{M,r}$$
$$AYKM = e_{1,2}^{-(s,m_0)-(s,m_1)} e_{M,r} - e_{1,r}^{-(s,m_0)-(s,m_1)} e_{M,2}$$

For the points near the four corners shown by the symbol ■ in Fig. (3.1), the four eqs. (3.11b), (3.11c), (3.11d) and (3.11e) have to be coupled to provide the following four solutions:

$$W_{2,2}^{(j_1,s_1)} = \frac{1}{AXN}\frac{1}{AYM}\sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} AXK1\cdot AYK1\cdot W_{i,r}^{(j,s)} \qquad (3.14a)$$

$$W_{N-1,2}^{(j_2,s_1)} = \frac{1}{AXN}\frac{1}{AYM}\sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} AXKN\cdot AYK1\cdot W_{i,r}^{(j,s)} \qquad (3.14b)$$

$$W_{2,M-1}^{(j_1,s_2)} = \frac{1}{AXN}\frac{1}{AYM}\sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} AXK1\cdot AYKM\cdot W_{i,r}^{(j,s)} \qquad (3.14c)$$

$$W_{N-1,M-1}^{(j_2,s_2)} = \frac{1}{AXN}\frac{1}{AYM}\sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} AXKN\cdot AYKM\cdot W_{i,r}^{(j,s)} \qquad (3.14d)$$

With eqs. (3.11a), (3.12), (3.13) and (3.14), all the boundary conditions can be directly substituted into eq. (3.9). As a result, the final system eigenvalue equation (3.9) of becomes:

$$\sum_{(i,j)\in e_1} C_1 W_{i,r}^{(j,s)} + 2\lambda^2 \sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} C_2 W_{i,r}^{(j,s)} + \lambda^4 \sum_{(r,s)\in e_2} C_3 W_{i,r}^{(j,s)} = \Omega^2 W_{i,r}^{(j,s)},$$
$$(i,j)\in e_1,(r,s)\in e_2 \qquad (3.15)$$

Where

$$C_1 = e_{k,i}^{(j,4)} - \frac{e_{k,2}^{(j,4)}\ AXK1 + e_{k,N-1}^{(j,4)}\ AXKN}{AXN},$$

$$C_2 = e_{k,i}^{(j,2)}\ \bar{e}_{h,r}^{(s,2)} - \frac{(AXK1 \cdot e_{k,2}^{(j,2)} + AXKN \cdot e_{k,N-1}^{(j,2)})}{AXN}\ \bar{e}_{h,r}^{(s,2)} -$$

$$\frac{(AYK1 \cdot \bar{e}_{h,2}^{(s,2)} + AYKM \cdot \bar{e}_{h,M-1}^{(s,2)})}{AYM} e_{k,i}^{(j,2)} +$$

$$\frac{(AXK1 \cdot AYK1 \cdot e_{k,2}^{(j,2)} \cdot \bar{e}_{h,2}^{(s,2)} + AXKN \cdot AYK1 \cdot e_{k,N-1}^{(s,2)} \bar{e}_{h,2}^{(s,2)})}{AXN \cdot AYM} +$$

$$\frac{(AXK1 \cdot AYKM \cdot e_{k,2}^{(j,2)} \bar{e}_{h,M-1}^{(s,2)} + AXKN \cdot AYKM \cdot e_{k,N-1}^{(j,2)} \bar{e}_{h,M-1}^{(s,2)})}{AXN \cdot AYM}$$

$$C_3 = \bar{e}_{h,r}^{(s,4)} - \frac{\bar{e}_{h,2}^{(s,4)} \cdot AYK1 + \bar{e}_{h,M-1}^{(s,4)} \cdot AYKM}{AYM}$$

Eq. (3.15) gives a system of $(N-4)\times(M-4)$ algebraic equations with $(N-4)\times(M-4)$ unknowns.

3.4 The Free Vibration Analysis of Square Plates:

In this section the free vibration analysis of square plates as given in equation (3.9) with $(\lambda = 1)$ will be solve numerically using proposed approach.

Applying the approach given in section (3.3) by considering two different sets of HB- problems to find the solution of such problem which are considered in the following cases:

Case1:

In this case we shall take a 5-poised HB- problems given for X and Y, respectively by the sets

$$e_1 = \{(1,0),(2,0),(3,0),(4,0),(5,0),(6,0)\}$$

$$e_2 = \{(1,0),(2,0),(3,0),(4,0),(5,0),(6,0)\}$$

With the node points given by (3.7) taking $N=6$ and $M=6$. First applying the G-spline interpolation-based DQM for the simply supported – simply supported – simply supported – simply supported boundary conditions, thus we have for $i,r=1,2,3,4,5,6$:

$$W_{1,r}^{(0,0)}=0, W_{6,r}^{(0,0)}=0, W_{i,1}^{(0,0)}=0, W_{i,6}^{(0,0)}=0 \qquad (3.16a)$$

$$\sum_{(i,0)\in e_1} e_{1,i}^{(0,2)} W_{i,r}^{(0,0)}=0 \ \ at\ \ X=0 \qquad (3.16b)$$

$$\sum_{(i,0)\in e_1} e_{6.i}^{(0,2)} W_{i,r}^{(0,0)}=0 \ \ at\ \ X=1 \qquad (3.16c)$$

$$\sum_{(r,0)\in e_2} \bar{e}_{1,r}^{(0,2)} W_{i,r}^{(0,0)}=0 \ \ at\ Y=0 \qquad (3.16d)$$

$$\sum_{(r,0)\in e_2} \bar{e}_{6,r}^{(0,2)} W_{i,r}^{(0,0)}=0 \ \ at\ Y=1 \qquad (3.16e)$$

Equations (3.16b) and (3.16c) may be coupled to give two solution $W_{2,r}^{(0,0)}$ and $W_{5,r}^{(0,0)}$, which are given for $r=3,4$ by:

$$W_{2,r}^{(0,0)}=\frac{1}{AXN}\sum_{(i,0)\in e_1^*} AXK1\cdot W_{i,r}^{(0,0)} \qquad (3.17a)$$

$$W_{5,r}^{(0,0)}=\frac{1}{AXN}\sum_{(i,0)\in e_1^*} AXKN\cdot W_{i,r}^{(0,0)} \qquad (3.17b)$$

Where:

$$e_1^*=\{(3,0),(4,0)\}$$

$$\left.\begin{aligned} AXN &= e_{6,2}^{(0,2)} e_{1,5}^{(0,2)} - e_{1,2}^{(0,2)} e_{6,5}^{(0,2)} \\ AXK1 &= e_{1,i}^{(0,2)} e_{6,5}^{(0,2)} - e_{1,5}^{(0,2)} e_{6,i}^{(0,2)} \\ AXKN &= e_{1,2}^{(0,2)} e_{6,i}^{(0,2)} - e_{1,i}^{(0,2)} e_{6,2}^{(0,2)} \end{aligned}\right\} for\ (i,0)\in e_1^*$$

Similarly, eqs. (3.16d) and (3.16e) can be coupled to give two solutions $W_{i,2}^{(0,0)}$ and $W_{i,5}^{(0,0)}$ given by:

$$W_{i,2}^{(0,0)} = \frac{1}{AYM} \sum_{(r,0)\in e_2^*} AYK1 \cdot W_{i,r}^{(0,0)} \qquad (3.18a)$$

$$W_{i,5}^{(0,0)} = \frac{1}{AYM} \sum_{(r,0)\in e_2^*} AYKM \cdot W_{i,r}^{(0,0)} \qquad (3.18b)$$

$for\ i = 3,4$

where

$$e_2^* = \{(3,0),(4,0)\}$$

$$\left.\begin{aligned} AYM &= \bar{e}_{6,2}^{(0,2)} \bar{e}_{1,5}^{(0,2)} - \bar{e}_{1,2}^{(0,2)} \bar{e}_{6,5}^{(0,2)} \\ AYK1 &= \bar{e}_{1,r}^{(0,2)} \bar{e}_{6,5}^{(0,2)} - \bar{e}_{1,5}^{(0,2)} \bar{e}_{6,r}^{(0,2)} \\ AYKM &= \bar{e}_{1,2}^{(0,2)} \bar{e}_{6,r}^{(0,2)} - \bar{e}_{1,r}^{(0,2)} \bar{e}_{6,2}^{(0,2)} \end{aligned}\right\} for\ (r,0) \in e_2^*$$

Four eq.s (3.16b), (3.16c), (3.16d) and (3.16e) have to be coupled to provide the following four solutions

$$W_{2,2}^{(0,0)} = \frac{1}{AXN}\frac{1}{AYM} \sum_{(i,0)\in e_1} \sum_{(r,0)\in e_2} AXK1 \cdot AYK1 \cdot W_{i,r}^{(0,0)} \qquad (3.19a)$$

$$W_{5,2}^{(0,0)} = \frac{1}{AXN}\frac{1}{AYM} \sum_{(i,0)\in e_1} \sum_{(r,0)\in e_2} AXKN \cdot AYK1 \cdot W_{i,r}^{(0,0)} \qquad (3.19b)$$

$$W_{2,5}^{(0,0)} = \frac{1}{AXN}\frac{1}{AYM} \sum_{(i,0)\in e_1} \sum_{(r,0)\in e_2} AXK1 \cdot AYKM \cdot W_{i,r}^{(0,0)} \qquad (3.19c)$$

$$W_{5,5}^{(0,0)} = \frac{1}{AXN}\frac{1}{AYM} \sum_{(i,0)\in e_1} \sum_{(r,0)\in e_2} AXKN \cdot AYKM \cdot W_{i,r}^{(0,0)} \qquad (3.19d)$$

With eqs. (3.16a), (3.17), (3.18) and (3.19), all the boundary conditions can be directly substituted into Equation (3.9). As a result, the final eigenvalue equation system becomes

$$[A]\{W_{i,r}^{(0,0)}\} = \Omega^2 \{W_{i,r}^{(0,0)}\}\ for\ (i,0) \in e_1, (r,0) \in e_2 \qquad (3.20)$$

The low natural frequency of the simply supported – simply supported - simply supported – simply supported will be given in table (3.1).

Secondly applying the DQM for the clamped –clamped – clamped –clamped boundary conditions therefore for $i, r = 1,2,3,4,5,6$ yields to:

$$W_{1,r}^{(0,0)} = 0, W_{6,r}^{(0,0)} = 0, W_{i,1}^{(0,0)} = 0, W_{i,6}^{(0,0)} = 0 \qquad (3.21a)$$

$$\sum_{(i,0)\in e_1} e_{1,i}^{(0,1)} W_{i,r}^{(0,0)} = 0 \ \ at \ \ X = 0 \qquad (3.21b)$$

$$\sum_{(i,0)\in e_1} e_{6.i}^{(0,1)} W_{i,r}^{(0,0)} = 0 \ \ at \ \ X = 1 \qquad (3.21c)$$

$$\sum_{(r,0)\in e_2} \bar{e}_{1,i}^{(0,1)} W_{i,r}^{(0,0)} = 0 \ \ at \ Y = 0 \qquad (3.21d)$$

$$\sum_{(r,0)\in e_2} \bar{e}_{6,i}^{(0,1)} W_{i,r}^{(0,0)} = 0 \ \ at \ Y = 1 \qquad (3.21e)$$

Equations (3.21b) and (3.21c) can be coupled together to give two solution $W_{2,r}^{(0,0)}$ and $W_{5,r}^{(0,0)}$. As given in (3.17) with

$$\left.\begin{aligned} AXN &= e_{6,2}^{(0,1)} e_{1,5}^{(0,1)} - e_{1,2}^{(0,1)} e_{6,5}^{(0,1)} \\ AXK1 &= e_{1,i}^{(0,1)} e_{6,5}^{(0,1)} - e_{1,5}^{(0,1)} e_{6,i}^{(0,1)} \\ AXKN &= e_{1,2}^{(0,1)} e_{6,i}^{(0,1)} - e_{1,i}^{(0,1)} e_{6,2}^{(0,1)} \end{aligned}\right\} for\,(i,0) \in e_1^*$$

Similarly, eqs. (3.21d) and (3.21e) can be coupled also to give two solutions $W_{i,2}^{(0,0)}$ and $W_{i,5}^{(0,0)}$. As given in (3.18) with

$$\left.\begin{aligned} AYM &= \bar{e}_{6,2}^{(0,1)} \bar{e}_{1,5}^{(0,1)} - \bar{e}_{1,2}^{(0,1)} \bar{e}_{6,5}^{(0,1)} \\ AYK1 &= \bar{e}_{1,r}^{(0,1)} \bar{e}_{6,5}^{(0,1)} - \bar{e}_{1,5}^{(0,1)} \bar{e}_{6,r}^{(0,1)} \\ AYKM &= \bar{e}_{1,2}^{(0,1)} \bar{e}_{6,r}^{(0,1)} - \bar{e}_{1,r}^{(0,1)} \bar{e}_{6,2}^{(0,1)} \end{aligned}\right\} for\,(r,0) \in e_2^*$$

Four eqs. (3.21b), (3.21c), (3.21d) and (3.21e) which have to be coupled to provide by (3.19), with AXN, AXK1, AXKN, AYM, AYK1, and AYKM are give above.

The natural low of frequency of the clamped – clamped – clamped –clamped boundary conditions will be given in table (3.1).

It is noted that all the calculations are performed by a computer programs written in MATLAB (see Appendix A) programs 10 and 11.

Table 3.1

Comparison of natural low frequency (Ω) of a square plate using G-spline interpolation-based differential quadrature with the exact solution given by Leissa [14] using case1.

Boundary Conditions	Ω(DQM)	Ω[14]
SS-SS-SS-SS	19.0665	19.0970
C-C-C-C	36.4037	36.4441

Case2:

In this case we shall consider another 5-poised HB sets for X and Y respectively given by:

$$e_1 = \{(1,0),(2,0),(3,0),(4,1),(5,1),(6,0)\}$$

$$e_2 = \{(1,0),(2,0),(3,0),(4,1),(5,1),(6,0)\}$$

With the node points given by (3.7) with $N = 6$ and $M = 6$ and first applying the G-spline interpolation-based DQM for the simply supported – simply supported – simply supported – simply supported boundary conditions, thus for $i, r = 1,2,3,4,5,6$ getting:

$$W_{1,r}^{(j,s)} = 0, W_{6,r}^{(j,s)} = 0, W_{i,1}^{(j,s)} = 0, W_{i,6}^{(j,s)} = 0 \qquad (3.22a)$$

$$\sum_{(i,j)\in e_1} e_{1,i}^{(j,2)} W_{i,r}^{(j,s)} = 0 \ \ at \ \ X = 0 \qquad (3.22b)$$

$$\sum_{(i,j)\in e_1} e_{6.i}^{(j,2)} W_{i,r}^{(j,s)} = 0 \ \ at \ \ X = 1 \qquad (3.22c)$$

$$\sum_{(r,s)\in e_2} \bar{e}_{1,i}^{(s,2)} W_{i,r}^{(j,s)} = 0 \ \ at\ Y\ = 0 \qquad (3.22d)$$

$$\sum_{(r,s)\in e_2} \bar{e}_{6,i}^{(s,2)} W_{i,r}^{(j,s)} = 0 \ \ at\ Y\ = 1 \qquad (3.22e)$$

Equations (3.22b) and (3.22c) can be coupled together to give two solutions, namely $W_{2,r}^{(0,s)}$ and $W_{5,r}^{(1,s)}$ and for $r = 3,4$ are defined by:

$$W_{2,r}^{(0,s)} = \frac{1}{AXN} \sum_{(i,j)\in e_1^*} AXK1 \cdot W_{i,r}^{(j,s)} \qquad (3.23a)$$

$$W_{5,r}^{(1,s)} = \frac{1}{AXN} \sum_{(i,j)\in e_1^*} AXKN \cdot W_{i,r}^{(j,s)} \qquad (3.23b)$$

Where:

$$e_1^* = \{(3,0),(4,1)\}$$

$$\left.\begin{aligned} AXN &= e_{6,2}^{(j,2)} e_{1,5}^{(j,2)} - e_{1,2}^{(j,2)} e_{6,5}^{(j,2)} \\ AXK1 &= e_{1,i}^{(j,2)} e_{6,5}^{(j,2)} - e_{1,5}^{(j,2)} e_{6,i}^{(j,2)} \\ AXKN &= e_{1,2}^{(j,2)} e_{6,i}^{(j,2)} - e_{1,i}^{(j,2)} e_{6,2}^{(j,2)} \end{aligned}\right\} for\ (i,j) \in e_1^*$$

Similarly, eqs. (3.22d) and (3.22e) can be coupled to give two solutions $W_{i,2}^{(j,0)}$ and $W_{i,5}^{(j,1)}$.

$$W_{i,2}^{(j,0)} = \frac{1}{AYM} \sum_{(r,s)\in e_2^*} AYK1 \cdot W_{i,r}^{(j,s)} \qquad (3.24a)$$

$$W_{i,5}^{(j,1)} = \frac{1}{AYM} \sum_{(r,s)\in e_2^*} AYKM \cdot W_{i,r}^{(j,s)} \qquad (3.24b)$$

$for\ i\ = 3,4$

where

$$e_2^* = \{(3,0),(4,1)\}$$

$$\left.\begin{aligned} AYM &= \bar{e}_{6,2}^{(s,2)} \bar{e}_{1,5}^{(s,2)} - \bar{e}_{1,2}^{(s,2)} \bar{e}_{6,5}^{(s,2)} \\ AYK1 &= \bar{e}_{1,r}^{(s,2)} \bar{e}_{6,5}^{(s,2)} - \bar{e}_{1,5}^{(s,2)} \bar{e}_{6,r}^{(s,2)} \\ AYKM &= \bar{e}_{1,2}^{(s,2)} \bar{e}_{6,r}^{(s,2)} - \bar{e}_{1,r}^{(s,2)} \bar{e}_{6,2}^{(s,2)} \end{aligned}\right\} for\ (r,s) \in e_2^*$$

Four eqs. (3.22b), (3.22c), (3.22d) and (3.22e) have to be coupled also to provide the following four solutions

$$W_{2,2}^{(j_1,s_1)} = \frac{1}{AXN}\frac{1}{AYM}\sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} AXK1\cdot AYK1\cdot W_{r,s}^{(j,s)} \quad (3.25a)$$

$$W_{5,2}^{(j_2,s_1)} = \frac{1}{AXN}\frac{1}{AYM}\sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} AXKN\cdot AYK1\cdot W_{r,s}^{(j,s)} \quad (3.25b)$$

$$W_{2,5}^{(j_1,s_2)} = \frac{1}{AXN}\frac{1}{AYM}\sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} AXK1\cdot AYKM\cdot W_{r,s}^{(j,s)} \quad (3.25c)$$

$$W_{5,5}^{(j_2,s_2)} = \frac{1}{AXN}\frac{1}{AYM}\sum_{(i,j)\in e_1}\sum_{(r,s)\in e_2} AXKN\cdot AYKM\cdot W_{r,s}^{(j,s)} \quad (3.25d)$$

With eqs. (3.22a), (3.23), (3.24) and (3.25), all the boundary conditions can be directly substituted into eq. (3.9). As a result, the final eigenvalue equation system becomes

$$[A]\{W_{i,r}^{(j,s)}\} = \Omega^2\{W_{i,r}^{(j,s)}\} for\ (i,j)\in e_1,(r,s)\in e_2 \quad (3.26)$$

The natural low frequency of the simply supported – simply supported - simply supported – simply supported will be given in table (3.2).

Secondly Applying the DQM for the clamped –clamped – clamped –clamped boundary conditions therefore for $i, j = 1,2,3,4,5,6$ getting:

$$W_{1,r}^{(j,s)} = 0, W_{6,r}^{(j,s)} = 0, W_{i,1}^{(j,s)} = 0, W_{i,6}^{(j,s)} = 0 \quad (3.27a)$$

$$\sum_{(i,j)\in e_1} e_{1,i}^{(j,1)} W_{i,r}^{(j,s)} = 0 \ at\ X = 0 \quad (3.27b)$$

$$\sum_{(i,j)\in e_1} e_{6.i}^{(j,1)} W_{i,r}^{(j,s)} = 0 \ at\ X = 1 \quad (3.27c)$$

$$\sum_{(r,s)\in e_2} \bar{e}_{1,i}^{(s,1)} W_{i,r}^{(j,s)} = 0 \ at\ Y = 0 \quad (3.27d)$$

$$\sum_{(r,s)\in e_2} \bar{e}_{6,i}^{(s,1)} W_{i,r}^{(j,s)} = 0 \ at\ Y = 1 \quad (3.27e)$$

Equations (3.27b), (3.27c) can be coupled to given two solution $W_{2,r}^{(0,s)}$ and $W_{5,r}^{(1,s)}$ a given in (3.17) with

$$\left.\begin{aligned} AXN &= e_{6,2}^{(j,1)} e_{1,5}^{(j,1)} - e_{1,2}^{(j,1)} e_{6,5}^{(j,1)} \\ AXK1 &= e_{1,i}^{(j,1)} e_{6,5}^{(j,1)} - e_{1,5}^{(j,1)} e_{6,i}^{(j,1)} \\ AXKN &= e_{1,2}^{(j,1)} e_{6,i}^{(j,1)} - e_{1,i}^{(j,1)} e_{6,2}^{(j,1)} \end{aligned}\right\} for\ (i,j) \in e_1^*$$

Similarly, Equation (3.27d) and (3.27e) can be coupled to give two solutions $W_{i,2}^{(j,0)}$ and $W_{i,5}^{(j,1)}$ a given in (3.18) with

$$\left.\begin{aligned} AYM &= \bar{e}_{6,2}^{(s,1)} \bar{e}_{1,5}^{(s,1)} - \bar{e}_{1,2}^{(s,1)} \bar{e}_{6,5}^{(s,1)} \\ AYK1 &= \bar{e}_{1,r}^{(s,1)} \bar{e}_{6,5}^{(s,1)} - \bar{e}_{1,5}^{(s,1)} \bar{e}_{6,r}^{(s,1)} \\ AYKM &= \bar{e}_{1,2}^{(s,1)} \bar{e}_{6,r}^{(s,1)} - \bar{e}_{1,r}^{(s,1)} \bar{e}_{6,2}^{(s,1)} \end{aligned}\right\} for\ (r,s) \in e_2^*$$

Four eqs. (3.27b), (3.27c), (3.27d) and (3.27e) have to be coupled to provide by (3.25), with AXN, AXK1, AXKN, AXM, AXK1 and AXKM are given above.

The natural low frequency of the clamped – clamped – clamped – clamped will be given in table (3.2).

It is noted that all the calculations are performed by a computer programs written by MATLAB see Appendix A programs 12 and 13.

Table 3.2

Comparison of natural low frequency (Ω) of a square plate using G-spline interpolation-based differential quadrature with the exact solution given by Leissa [14] using case 2

Boundary Conditions	Ω(DQM)	Ω[14]
SS-SS-SS-SS	19.1797	19.0970
C-C-C-C	36.9222	36.4441

Conclusion and Recommendations for Future work

From the present study of this thesis, the following conclusion may be drawn:

1- Differential quadrature method can be considered as a powerful tool to find the numerical solution of differential equations.
2- The G-spline-based DQM can be considered as a generalization to the usual DQM.
3- The G-spline-based DQM gave reasonable results in which we used a small number of node points in the computations.

For the future work, we may suggest the following:

1- Using the DQM to solve non-local boundary value problems.
2- Studying the numerical solution of fractional order differential equations using polynomial based DQM or spline-based DQM.
3- Studying the numerical solution of differential equations using G-spline-based DQM with the modification of weighting coefficient matrices.
4- Studying the numerical solution of partial fractional order differential equations using G-spline-based DQM.

References

[1] Bellman R. and Casti, J., "Differential quadrature and long-term integration", Journal of Mathematical Analysis and Applications, Vol. 34, pp. 235-238, 1971.

[2] Bellman R., Kashef B. G. and Casti J., "Differential quadrature: A technique for the rapid solution of nonlinear partial differential Equations", Journal of Computational Physics, Vol. 10, pp. 40-52, 1972.

[3] Bellman R. E. and Roth R. S., " Methods in approximation: techniques for mathematical modeling", D. Reidel Publishing Company, Dordrecht, Holland, 1986.

[4] Bert C. W. and Malik M., "Differential quadrature method in computational mechanics: A review", Appl. Mech. Rev., Vol. 49, pp. 1-27, 1996.

[5] Bert C. W., Wang X. and Striz A. G.,“Differential quadrature for static and free vibration analysis of anisotropic plates”, International Journal of Solids and Structures, Vol. 30, pp. 1737-1744, 1993.

[6] Bert C.W., Wang, X. and Striz, A.G., “Static and free vibration analysis of beams and plates by differential quadrature method”, Acta Mechanica, Vol.102, pp. 11-24, 1994a.

[7] Bert C. W., Wang X. and Striz, A. G., “Convergence of the DQM in the analysis of anisotropic plates”, Journal of Sound and Vibration, Vol. 170, pp. 140-144, 1994b.

[8] Bert C. W., Jang S.K. and Striz A. G., "Nonlinear bending analysis of orthotropic rectangular plates by the method of differential quadrature ", Computational Methods, Vol. 5, pp. 217-226, 1989.

[9] Blevins R. D., "Formulas for natural frequency and mode shape", Robert E. Krieger Publishing Company, Malanar, Florida, 1984.

[10] Deboor C.,"A practical guide to splines", Springer-Verlag New York, Inc., 1978.

[11] Jalaal M., Soleimani S., Domairry G., Ghasemi E., Bararnia H., Mohammadi F. and Barari A., "Numerical simulation of electric field in complex geometries for different electrode arrangements using meshless local MQ-DQ method", Journal of Electrostatics, Vol. xxx, pp. 1-8, 2011.

[12] Karami G. and Malekzadeh P., “A new differential quadrature methodology for beam analysis and the associated differential quadrature element method”, Computational Methods in Applied Mechanics and Engineering, Vol. 191, pp. 3509-3526, 2002.

[13] Korkmaz A. and Dag I., "A differential quadrature algorithm for simulations of nonlinear Schrödinger equation", Computers and Mathematics with Applications, Vol. 56, pp. 2222–2234, 2008.

[14] Leissa A. W., "The free vibration of rectangular plates", Journal Sound Vibration, Vol. 31, pp. 257-293, 1973.

[15] Lienhard J. H., "A heat transfer textbook", Prentice-Hall, Englewood Cliffs, N.J., 1987.

[16] Malik M. and Bert C.W., “Implementing multiple boundary conditions in the DQ solution of high-order PDE’s: Application to

free vibration of plates", International Journal of Numerical Methods in En- gineering, Vol. 39, pp. 1237-1258, 1996.

[17] Micula G., "A variational approach to spline function theory", Rend. Sem. Mat. University Pol. Torino, Vol. 61, pp. 3, Splines and Radial Functions, 2003.

[18] Mohammed, O. H., "Functions Approximation using G-spline and its Generalization to Two- Dimensional spaces", Ph.D. Thesis, College of Science, AL-Nahrain University, 2006.

[19] Powell M. J., "Approximation theory and methods", Cambridge University Press, 1981.

[20] Quan J. R. and Chang C. T., "New insights in solving distributed system equations by the quadrature methods-I ", Comput. Chem. Engrg., Vol. 13, pp. 779-788, 1989a.

[21] Quan J. R. and Chang C. T., "New insights in solving distributed system equations by the quadrature methods-II ", Comput. Chem. Engrg., Vol. 13, pp. 1017-1024, 1989b.

[22] Quan J. R. and Chang C. T., "New sightings in involving distributed system equations by the quadrature methods-II", Comput. Chem. Engrg., Vol. 13, pp. 71017–71024, 1989c.

[23] Schoenberg I. J., " On the Ahlberg-Nilson extension of spline interpolation: the G-spline and their optimal properties", Journal Mathematical Analysis Application, Vol. 21, pp. 207-231, 1968.

[24] Shu C. and Du H., "Implementation of clamped and simply supported boundary conditions in the GDQ free vibration analysis

of beam and plates", International Journal Solids Structures, Vol. 34, Iss. 7, pp. 819-835, 1997.

[25] Shu C. and Richards B. E., "Application of generalized differential quadrature to solve two dimensional incompressible Navier Stokes equations", International Journal Numerical Methods Fluids, Vol. 15, pp. 791–798, 1992.

[26] Shu C. and Xue H., "Explicit computation of weighting coefficients in the harmonic differential quadrature", Journal of Sound and Vibration, Vol. 204, Iss. 3, pp. 549–555, 1997.

[27] Shu C. and Wu Y. L.,"Integrated radial basis functions-based differential quadrature method and its performance", International Journal Numerical Methods Fluids, Vol. 53, pp. 969–984, 2007.

[28] Shu C., "Differential quadrature and its application in engineering", National University of Singapore, 10 Kent Ridge crescent, Singapore 119260, 2000.

[29] Shu C., " Generalized differential-integral quadrature and application to the simulation of incompressible viscous flows including parallel computation", Ph.D., thesis, University of Glasgow, U. K., 1991.

[30] Stephen W., "An introduction to the mathematics and construction of splines", Addix. software consultancy limited, Version 1.6, September, 2002.

[31] Tom D. and Andrew W., "Linear Algebra in Twenty Five Lectures", March 27, 2012.

[32] Wang X. and Bert C. W., " A new approach in applying differential quadrature to static and free vibrational analysis of beams and

plates", Journal Sound Vibration, Vol. 162, Iss. 3, pp. 566-572, 1993.

[33] Wang X., Gu H. and Liu, B., “On Buckling Analysis of Beams and Frame Structures by the Differential Quadrature Element Method”, Proceedings of Engineering Mechanics, Vol. 1, pp. 382-385, 1996.

[34] Wang X. W. and Gu. H. Z., “Static analysis of frame structures by the differential quadrature element method”, International Journal of Numerical Methods in Engineering, Vol. 40, pp. 759-772, 1997.

[35] Wang Y., "Differential Quadrature Method & Differential Quadrature Element Method-Theory and Practice", Ph.D. Dissertation, Nanjing University of Aeronautics and Astronautics, China (in Chinese), 2001.

[36] Wu T. Y. and Liu G. R., “Application of generalized differential quadrature rule to sixth-order differential equations”, Communications in Numerical Methods in Engineering, Vol. 16, pp. 777-784, 2000.

[37] Wu T. Y. and Liu G. R., “Free vibration analysis of circular plates with variable thickness by the generalized differential quadrature rule”, International Journal of Solids and Structures, Vol. 38, pp.7967-7980, 2001.

[38] Zhi Z. and Yingyan Z., "Advanced Differential Quadrature Methods", Taylor & Francis Group, LLC, 2009.

Computer programs

Program 1:-

```
n= 21 ;
x(1) = 0 ;
x(n) = 1 ;
for i = 1 :n-4
tt(i)=-1;
end
for i = 2:n-1
x(i)=x(1)+(1/2)*(1-cos((i-1)*(3.142)/(n-1)))*(x(n)-x(1));
end
for i = 1:n
w(i)=(1/48)*(x(i)^2)*((5*x(i)-2*(x(i)^2)-3.0));
end
for i = 1 : n
for j = 1 : n
z = 1;
for k = 1:n
if (k ~= i) && (k ~= j)
r1 = (x(i)-x(k))/(x(j)-x(k));
z = z * r1;
end
end
if i ~= j ;
a(i,j) = (1/(x(j)-x(i)))*z ;
end
end
end
for i = 1 : n
a(i,i) = 0 ;
for j = 1 : n
if i ~= j ;
a(i,i) = ( a(i,i)+ a(i,j));
end
end
a(i,i) = -1* a(i,i) ;
end
for i = 1 : n
for j = 1 : n
r(i,j)=0;
for k = 1 : n
r(i,j)= r(i,j)+(a(i,k)*a(k,j));
end
b(i,j)=r(i,j);
end
end
for i = 1 : n
for j = 1 : n
```

```
r1=0;
for k = 1 : n
r1= r1+(a(i,k)*b(k,j));
end
e3(i,j)=r1;
end
end
for i = 1 : n
for j = 1 : n
r2(i,j)=0;
for k = 1 : n
r2(i,j)=r2(i,j)+(a(i,k)*e3(k,j));
end
e4(i,j)=r2(i,j);
end
end
for i = 3 : n-2
for j = 3 : n-2
t1= a(1,j)*b(n,n-1)-a(1,n-1)*b(n,j);
t2= a(1,2)*b(n,j)-a(1,j)*b(n,2);
t3  = b(n,2)*a(1,n-1)-a(1,2)*b(n,n-1);
v(i,j) = e4(i,j)+((e4(i,2)*t1+e4(i,n-1)*t2)/t3);
end
end
for i = 1 : n-4
for j = 1 : n-4
c1(i,j)= v(i+2,j+2);
end
end
c2 = inv (c1)*tt';
for j = 3 : n-2
u = u+ a(1,j)*c2(j-2);
y = y + b(n,j)*c2(j-2);
end
k(1)=-1*u;
k(2)=-1*y;
s(1,1)=a(1,2);
s(1,2)=a(1,20);
s(2,1)=b(21,2);
s(2,2)=b(21,20);
mm = inv (s)*k';
ww(1)=0.0;
ww(2)=mm(1);
for i=3:n-2
ww(i)=c2(i-2);
end
w(n-1)=mm(2);
ww(n)=0.0;
(abs(w)-abs(ww))'
```

Program 2:-

```
n= 15 ;
x(1) = 0 ;
x(n) = 1 ;
for i = 1 : 17
tt(i)=-1;
end
for i = 2:n-1
x(i)=x(1)+(1/2)*(1-cos((i-1)*(3.142)/(n-1)))*(x(n)-x(1));
end
for i = 1:n
w(i) = (1/48)*(x(i)^2)*((5*x(i)-2*(x(i)^2)-3.0));
end
for i = 1 : n
for j = 1 : n
z = 1;
for k = 1:n
if (k ~= i) && (k ~= j)
r1 = (x(i)-x(k))/(x(j)-x(k));
z = z * r1 ;
end
end
if i ~= j ;
a(i,j) = (1/(x(j)-x(i)))*z ;
end
end
end
for i = 1 : n
a(i,i) = 0 ;
for j = 1 : n
if i ~= j ;
a(i,i) = ( a(i,i)+ a(i,j));
end
end
a(i,i) = -1* a(i,i) ;
end
b=a*a;
e3=a*b;
e4=a*e3;
for i = 3 : n-2
for j = 3 : n-2
t1= b(1,j)*b(n,n-1)-b(1,n-1)*b(n,j);
t2= b(1,2)*b(n,j)-b(1,j)*b(n,2);
t3  = b(n,2)*b(1,n-1)-b(1,2)*b(n,n-1);
v(i,j) = e4(i,j)+((e4(i,2)*t1+e4(i,n-1)*t2)/t3);
end
end
for i = 1 : n-4
for j = 1 : n-4
c1(i,j)= v(i+2,j+2);
end
```

```
end
v1=(eig(v));
for i=1:n-4
(v1(i))^.5
end
```

Program 3:-

```
n= 15 ;
x(1) = 0 ;
x(n) = 1 ;
for i = 1 : 17
tt(i)=-1;
end
for i = 2:n-1
x(i)= x(1)+(1/2)*(1-cos((i-1)*(3.142)/(n-1)))*(x(n)-
x(1));
end
for i = 1:n
w(i) = (1/48)*(x(i)^2)*((5*x(i)-2*(x(i)^2)-3.0));
end
for i = 1 : n
for j = 1 : n
z = 1;
for k = 1:n
if (k ~= i) && (k ~= j)
r1 = (x(i)-x(k))/(x(j)-x(k));
z = z * r1 ;
end
end
if i ~= j ;
a(i,j) = (1/(x(j)-x(i)))*z ;
end
end
end
for i = 1 : n
a(i,i) = 0 ;
for j = 1 : n
if i ~= j ;
a(i,i) = ( a(i,i)+ a(i,j));
end
end
a(i,i) = -1* a(i,i) ;
end
b=a*a;
e3=a*b;
e4=a*e3;
for i = 3 : n-2
for j = 3 : n-2
t1= b(1,j)*a(n,n-1)-b(1,n-1)*a(n,j);
t2= b(1,2)*a(n,j)-b(1,j)*a(n,2);
t3= b(n,2)*a(1,n-1)-b(1,2)*a(n,n-1);
```

```
v(i,j) = e4(i,j)+((e4(i,2)*t1+e4(i,n-1)*t2)/t3);
end
end
for i = 1 : n-4
for j = 1 : n-4
c1(i,j)= v(i+2,j+2);
end
end
v1=(eig(v));
for i=1:n-4
(v1(i))^.5
end
```

Program 4:-

```
n= 6;
x(1) = 0 ;
x(n) = 1 ;
for i = 1 : n-4
tt(i)=-1;
end
a=[69.0972,-51.2641.5906,-169.6349,170.8088,-
85.5966;44.8789,-97.8936,104.3358,102.2127,101.9542,-
50.9635;8.8931,-18.8908,15.2475,-11.7038,13.1703,-
6.7216;-0.6083,1.4726,-2.90442,-0.6976,4.3496,-
1.6216;0.9749,-0.8755,1.4228,-3.6459,-2.7209,5.4315;-
0.9078,2.0384,-2.9197,5.6686,-20.8738,16.97796];
b=a*a;
e3=a*b;
e4=a*e3;
for i = 3 : n-2
for j = 3 : n-2
t1= a(1,j)*b(n,n-1)-a(1,n-1)*b(n,j);
t2= a(1,2)*b(n,j)-a(1,j)*b(n,2);
t3  = b(n,2)*a(1,n-1)-a(1,2)*b(n,n-1);
v(i,j) = e4(i,j)+((e4(i,2)*t1+e4(i,n-1)*t2)/t3);
end
end
for i = 1 : n-4
for j = 1 : n-4
c1(i,j)= v(i+2,j+2);
end
end
c2 = inv (c1)*tt';
format long e;
for j = 3 : n-2
u = u+ a(1,j)*c2(j-2);
y = y + b(n,j)*c2(j-2);
end
k(1)=-1*u;
k(2)=-1*y;
s(1,1)=a(1,2);
```

```
s(1,2)=a(1,5);
s(2,1)=b(6,2);
s(2,2)=b(6,5);
mm = inv (s)*k';
ww(1)=0.0;
ww(2)=mm(1);
for i=3:n-2
ww(i)=c2(i-2);
end
ww(n-1)=mm(2);
ww(n)=0.0;
abs((abs(w)-abs(ww)))'
```

Program 5:-

```
n= 6 ;
x(1) = 0 ;
x(n) = 1 ;
for i = 1 : n-4
tt(i)=-1;
end
a=[-31.089941056156209505,-
12.23057741217772963,0.77968683595852671497,-
18.453027890343230902,-67.76557439272856797,-
12.737478218682955771;31.462632673736228255,8.55664669146
90994018,-3.1589433681645365545,-
39.413189727876957288,142.02076329965116237,26.5420617342
75329012;-39.100582338249030684,-12.681462357270821679,-
1.98164509005807730 47,67.096870399794979987,-
198.92631460106778661,-35.97074306959821994;-
6.857165106827839516,-2.9264003479616642098,-
1.0132240340995923634,15.572244071285122453,-
50.939335688638180131,-8.5384766821064232152;-
4.2976118828138608537,-1.6666600543338908889,-
0.82218507370751093234,11.066403445120947639,-
18.526860515951349241,-
2.7261574346852199537;20.961890720669004935,8.39639816099
71580041,3.7648635847823997444,-
46.053352732815160677,124.67113591572975694,22.1661595540
054397];
 b=a*a;
 e3=a*b;
 e4=a*e3;
 for i = 3 : n-2
 for j = 3 : n-2
 t1= a(1,j)*b(n,n-1)-a(1,n-1)*b(n,j);
 t2= a(1,2)*b(n,j)-a(1,j)*b(n,2);
 t3= b(n,2)*a(1,n-1)-a(1,2)*b(n,n-1);
 v(i,j) = e4(i,j)+((e4(i,2)*t1+e4(i,n-1)*t2)/t3);
 end
 end
 for i = 1 : n-4
```

```
for j = 1 : n-4
c1(i,j)= v(i+2,j+2);
end
end
c2 = inv (c1)*tt';
format long e;
for j = 3 : n-2
u = u+ a(1,j)*c2(j-2);
y = y + b(n,j)*c2(j-2);
end
k(1)=-1*u;
k(2)=-1*y;
s(1,1)=a(1,2);
s(1,2)=a(1,5);
s(2,1)=b(6,2);
s(2,2)=b(6,5);
mm = inv (s)*k';
ww(1)=0.0;
ww(2)=mm(1);
for i=3:n-2
ww(i)=c2(i-2);
end
ww(n-1)=mm(2);
ww(n)=0.0;
((abs(w)-abs(ww)))'
```

Program 6:-

```
n=6;
a=[69.09719985911381365,44.87886386461249491,8.8931247930
295905263,-.60825729333936220064,.37448032895763460070,-
.907817665373853087 86;-151.26408151768037419,-
97.89357021388116992,-
18.890881193353345692,1.4725962584529261853,-
.875469956598808400 13,2.03840291511047200 54;166.590574306
62880177,104.33583623781664348,15.247473736065912130,-
2.9044160365662253014,1.4228339691646464165,-
2.9196718673154255307;-169.63487768745532748,-
102.31273612034090082,-11.703828191467317361,-
.69756771628189845849,-
3.6458955204879273155,5.6686264480545005592;170.808803056
09099342,101.95422048810383070,13.170289965783140359,4.34
962935176186918 09,-2.7209006179462262763,-
20.873843965251741921;-85.59661984836753540,-
50.96349354939456879,-6.721567816775886375,-
1.6117677538230514370,5.4311368799616276354,16.9879695388
96380639];
b=a*a;
e3=a*b;
e4=a*e3;
for i = 3 : n-2
```

```
for j = 3 : n-2
t1= b(1,j)*b(n,n-1)-b(1,n-1)*b(n,j);
t2= b(1,2)*b(n,j)-b(1,j)*b(n,2);
t3= b(n,2)*b(1,n-1)-b(1,2)*b(n,n-1);
v(i,j)= e4(i,j)+((e4(i,2)*t1+e4(i,n-1)*t2)/t3);
end
end
for i = 1 : n-4
for j = 1 : n-4
c1(i,j)= v(i+2,j+2);
end
end
v1=(eig(v));
for i=1:n-2
(v1(i))^.5
end
```

Program 7:-

```
n=6;
a=[69.09719985911381365,44.87886386461249491,8.8931247930
295905263,-.608257293339362200 64,.37448032895763460070,-
.907817665373853087 86;-151.26408151768037419,-
97.89357021388116992,-
18.890881193353345692,1.4725962584529261853,-
.875469956598808400 13,2.038402951511047200 54;166.590574306
62880177,104.33583623781664348,15.247473736065912130,-
2.904416036566225301 4,1.4228339691646464165,-
2.91967186731542553 07;-169.63487768745532748,-
102.31273612034090082,-11.703828191467317361,-
.697567716281898458 49,-
3.645895520487927315 5,5.6686264480545005592;170.808803056
09099342,101.95422048810383070,13.170289965783140359,4.34
962935176186918 09,-2.7209006179462262763,-
20.899843965251741921;-85.59261984836753540,-
50.18949354939456879,-6.721567816775886375,-
1.611755753823051437 0,5.4315368799616276354,16.9879695388
96380639];
b=a*a;
e3=a*b;
e4=a*e3;
for i = 3 : n-2
for j = 3 : n-2
t1= a(1,j)*b(n,n-1)-a(1,n-1)*b(n,j);
t2= a(1,2)*b(n,j)-a(1,j)*b(n,2);
t3= a(n,2)*b(1,n-1)-a(1,2)*b(n,n-1);
v(i,j)= e4(i,j)+((e4(i,2)*t1+e4(i,n-1)*t2)/t3);
end
end
for i = 1 : n-4
for j = 1 : n-4
c1(i,j)= v(i+2,j+2);
```

```
end
end
v1=(eig(v));
for i=1:n-2
(v1(i))^.5
end
```

Program 8:-

```
n=6;
a=[-31.089941056156209505,-
12.23057741217772963,0.77968683595852671497,-
18.453027890343230902,-67.76557439272856797,-
12.737478218682955771;31.462632673736228255,8.55664669146
90994018,-3.1589433681645365545,-
39.413189727876957288,142.02076329965116237,26.5420617342
75329012;-39.100582338249030684,-12.681462357270821679,-
1.98164509005807730 47,67.096870399794979987,-
198.92631460106778661,-35.97074306959821994;-
6.857165106827839516,-2.9264003479616642098,-
1.0132240340995923634,15.572244071285122453,-
50.939335688638180131,-8.5384766821064232152;-
4.2976118828138608537,-1.66666005433389088 89,-
0.82218507370751093234,11.066403445120947639,-
18.526860515951349241,-
2.7261574346852199537;20.887890720669004935,8.39639816099
71580041,3.4048635847823997444,-
46.023352732815160677,124.97113591572975694,22.2661595540
054397];
b=a*a;
e3=a*b;
e4=a*e3;
for i = 3 : n-2
for j = 3 : n-2
t1= b(1,j)*b(n,n-1)-b(1,n-1)*b(n,j);
t2= b(1,2)*b(n,j)-b(1,j)*b(n,2);
t3= b(n,2)*b(1,n-1)-b(1,2)*b(n,n-1);
v(i,j) = e4(i,j)+((e4(i,2)*t1+e4(i,n-1)*t2)/t3);
end
end
for i = 1 : n-4
for j = 1 : n-4
c1(i,j)= v(i+2,j+2);
end
end
v1=(eig(v));
for i=1:n-2
(v1(i))^.5
end
```

Program 9:-

```
n=6;
a=[-31.089941056156209505,-
12.23057741217772963,0.77968683595852671497,-
18.453027890343230902,-67.76557439272856797,-
12.737478218682955771;31.462632673736228255,8.55664669146
90994018,-3.1589433681645365545,-
39.413189727876957288,142.02076329965116237,26.5420617342
75329012;-39.100582338249030684,-12.681462357270821679,-
1.9446450900580773047,67.996870399794979987,-
198.92631460106778661,-35.97074306959821994;-
6.857165106827839516,-2.9264003479616642098,-
1.0132240340995923634,15.5722440712851224 53,-
50.939335688638180131,-8.5384766821064232152;-
4.2976118828138608537,-1.6666600543338908889,-
0.82218507370751093234,11.066403445120947639,-
18.726860515951349241,-
2.7261574346852199537;20.887890720669004935,8.99639816099
71580041,3.1048635847823997444,-
46.022352732815160677,124.990135915729756 94,22.2661595540
054397];
b=a*a;

e3=a*b;
e4=a*e3;
for i = 3 : n-2
for j = 3 : n-2
t1= a(1,j)*a(n,n-1)-a(1,n-1)*a(n,j);
t2= a(1,2)*a(n,j)-a(1,j)*a(n,2);
t3= a(n,2)*a(1,n-1)-a(1,2)*a(n,n-1);
v(i,j) = e4(i,j)+((e4(i,2)*t1+e4(i,n-1)*t2)/t3);
end
end
for i = 1 : n-4
for j = 1 : n-4
c1(i,j)= v(i+2,j+2);
end
end
v1=(eig(v));
for i=1:n-2
(v1(i))^.5
end
```

Program 10:-

```
n=6;
a=[69.09959985911381365,44.9700486461249491,8.99247930295
905263,-.60625729333936220064,.37448032895763460070,-
.90781766537385308786;-151.26408151768037419,-
97.89357021388116992,-
18.890881193353345692,1.4725962584529261853,-
.87546995659880840013,2.038402951511047200 54;166.590574306
```

```
62880177,104.33583623781664348,15.247473736065912130,-
2.90441603656622530l4,1.42283396916464641l5,-
2.91967186731542553l7;-169.63487768745532748,-
102.31273612034090082,-11.703828191467317361,-
.697567716281898458l9,-
3.64589552048792731l55,5.66862644805450055l92;170.808803056
09099342,101.95422048810383070,13.170289965783140359,4.34
962935176186918l09,-2.7209006179462262763,-
20.873843965251741921;-87.59561984836753540,-
50.96349354939456879,-6.721567816775886375,-
1.621755753823051437l0,5.43153687996162763l54,16.9779695388
96380639];
b=a*a;
c=a*b;
e4=a*c;
for i = 3 : n-2
for j = 3 : n-2
ij=(n-4)*(i-3)+(j-3)+1;
AXN=-1*( a(n,2)*a(1,n-1)-a(1,2)*a(n,n-1));
AYN= -1*(a(n,2)*a(1,n-1)-a(1,2)*a(n,n-1));
for k1=3:n-2
kj=(n-4)*(k1-3)+j-3+1;
AXK1= a(1,k1)*a(n,n-1)-a(1,n-1)*a(n,k1);
AXKN= a(1,2)*a(n,k1)-a(1,k1)*a(n,2);
PDD(ij,kj)=e4(i,k1)-((AXK1)*e4(i,2)+(AXKN)*e4(i,n-
1))/AXN;
end
for k2=3:n-2
ik=(n-4)*(i-3)+k2-3+1;
AYK1=a(1,k2)*a(n,n-1)-a(1,n-1)*a(n,k2);
AYKN= a(1,2)*a(n,k2)-a(1,k2)*a(n,2);
PDD(ij,ik)=PDD(ij,ik)+e4(j,k2)-(e4(j,2)*(AYK1)+e4(j,n-
1)*(AYKN))/AYN;
end
end
end
for i = 3 : n-2
for j = 3 : n-2
ij=(n-4)*(i-3)+(j-3)+1;
for k1=3:n-2
for k2=3:n-2
ik=(n-4)*(k1-3)+(k2-3)+1;
AXK1= a(1,k1)*a(n,n-1)-a(1,n-1)*a(n,k1);
AXKN= a(1,2)*a(n,k1)-a(1,k1)*a(n,2);
AYK1= a(1,k2)*a(n,n-1)-a(1,n-1)*a(n,k2);
AYKN= a(1,2)*a(n,k2)-a(1,k2)*a(n,2);
r1=b(i,k1)*b(i,k2)-((AXK1*b(i,2)+AXKN*b(i,n-
1))/AXN)*b(j,k2);
r2=(AYK1*b(j,2)+AYKN*b(j,n-1))/AYN)*b(i,k1);
r3=((AXK1*AYK1*b(i,2)*b(j,2)+AXKN*AYK1*b(i,n-
1)*b(j,2))/AXN*AYN);
```

```
r4=((AXK1*AYKN*b(i,2)*b(j,n-1)+AXKN*AYKN*b(i,n-1)*b(j,n-
1))/AXN*AYN);
PDD(ij,ik)=PDD(ij,ik)+2*(b(i,k1)*b(j,k2)-
(AXK1*b(i,2)*(b(j,k2)-b(j,2)*AYK1/AYN)+AXKN*b(i,n-
1)*(b(j,k2)-b(j,2)*AYK1/AYN))/AXN-
(AYK1*b(j,2)*b(i,k1)+AYKN*b(j,n-1)*(b(i,k1)-
(b(i,2)*AXK1+b(i,n-1)*AXKN)/AXN))/AYN);
end
end
end
end
v1=(eig(PDD));
for i=1:(n-4)^2
v2(i)=v1(i)^.5;
end
v3=sort(v2);
```

Program 11:-

```
n=6;
a=[69.09719985911381365,44.8742486461249491,8.89312479302
95905263,-.60825729333936220064,.37448032895763460070,-
.90781766537385308786;-151.26408151768037419,-
97.89357021388116992,-
18.8908811933533456 92,1.4725962584529261853,-
.87546995659880840013,2.03840291511047200 54;166.590574306
62880177,104.33583623781664348,15.247473736065912130,-
2.90441603656622530 14,1.42283396916464641 65,-
2.91967186731542553 07;-169.63487768745532748,-
102.31273612034090082,-11.703828191467317361,-
.69756771628189845849,-
3.64589552048792731 55,5.66862644805450055 92;170.808803056
09099342,101.95422048810383070,13.170289965783140359,4.34
962935176186918 09,-2.72090061794622627 63,-
20.87384396525174192 1;-87.595619848367535 40,-
50.96349354939456879,-6.721567816775886375,-
1.62175575382305143 70,5.43153687996162763 54,16.9779695388
96380639];
b=a*a;
e3=a*b;
e4=a*e3;
for i = 3 : n-2
for j = 3 : n-2
ij=(n-4)*(i-3)+(j-3)+1;
AXN=-1*( b(n,2)*b(1,n-1)-b(1,2)*b(n,n-1));
AYN= -1*(b(n,2)*b(1,n-1)-b(1,2)*b(n,n-1));
for k1=3:n-2
kj=(n-4)*(k1-3)+j-3+1;
AXK1= b(1,k1)*b(n,n-1)-b(1,n-1)*b(n,k1);
AXKN= b(1,2)*b(n,k1)-b(1,k1)*b(n,2);
```

```
PDD(ij,kj)=e4(i,k1)-((AXK1)*e4(i,2)+(AXKN)*e4(i,n-
1))/AXN;
end
for k2=3:n-2
ik=(n-4)*(i-3)+k2-3+1;
AYK1= b(1,k2)*b(n,n-1)-b(1,n-1)*b(n,k2);
AYKN= b(1,2)*b(n,k2)-b(1,k2)*b(n,2);
PDD(ij,ik)=PDD(ij,ik)+e4(j,k2)-(e4(j,2)*(AYK1)+e4(j,n-
1)*(AYKN))/AYN;
end
end
end
for i = 3 : n-2
for j = 3 : n-2
ij=(n-4)*(i-3)+(j-3)+1;
for k1=3:n-2
for k2=3:n-2
ik=(n-4)*(k1-3)+(k2-3)+1;
AXK1= b(1,k1)*b(n,n-1)-b(1,n-1)*b(n,k1);
AXKN= b(1,2)*b(n,k1)-b(1,k1)*b(n,2);
AYK1= b(1,k2)*b(n,n-1)-b(1,n-1)*b(n,k2);
AYKN= b(1,2)*b(n,k2)-b(1,k2)*b(n,2);
r1=b(i,k1)*b(i,k2)-((AXK1*b(i,2)+AXKN*b(i,n-
1))/AXN)*b(j,k2);
r2=((AYK1*b(j,2)+AYKN*b(j,n-1))/AYN)*b(i,k1);
r3=((AXK1*AYK1*b(i,2)*b(j,2)+AXKN*AYK1*b(i,n-
1)*b(j,2))/AXN*AYN);
r4=((AXK1*AYKN*b(i,2)*b(j,n-1)+AXKN*AYKN*b(i,n-1)*b(j,n-
1))/AXN*AYN);
PDD(ij,ik)=PDD(ij,ik)+2*(b(i,k1)*b(j,k2)-
(AXK1*b(i,2)*(b(j,k2)-b(j,2)*AYK1/AYN)+AXKN*b(i,n-
1)*(b(j,k2)-b(j,2)*AYK1/AYN))/AXN-
(AYK1*b(j,2)*b(i,k1)+AYKN*b(j,n-1)*(b(i,k1)-
(b(i,2)*AXK1+b(i,n-1)*AXKN)/AXN))/AYN);
end
end
end
end
v1=(eig(PDD));
for i=1:(n-4)^2
v2(i)=v1(i)^.5;
end
v3=sort(v2);
```

program 12:-

```
n=6;
a=[-31.089941056156209505,-
6.23057741217772963,0.77968683595852671497,-
18.453027890343230902,-67.76557439272856797,-
12.737478218682955771;31.462632673736228255,8.55664669146
90994018,-3.1589433681645365545,-
```

```
39.413189727876957288,142.02076329965116237,26.5420617342
75329012;-39.100582338249030684,-12.681462357270821679,-
1.9816450900580773047,67.096870399794979987,-
198.92631460106778661,-35.97074306959821994;-
6.857165106827839516,-2.9264003479616642098,-
1.0132240340995923634,15.572244071285122453,-
50.939335688638180131,-8.5384766821064232152;-
4.2976118828138608537,-1.6666600543338908889,-
0.82218507370751093234,11.066403445120947639,-
18.526860515951349241,-
2.7261574346852199537;20.961890720669004935,8.39639816099
71580041,3.7648635847823997444,-
46.053352732815160677,124.67113591572975694,22.1661595540
054397];
b=a*a;
e3=a*b;
e4=a*e3;
for i = 3 : n-2
for j = 3 : n-2
ij=(n-4)*(i-3)+(j-3)+1;
AXN=-1*( b(n,2)*b(1,n-1)-b(1,2)*b(n,n-1));
AYN= -1*(b(n,2)*b(1,n-1)-b(1,2)*b(n,n-1));
for k1=3:n-2
kj=(n-4)*(k1-3)+j-3+1;
AXK1= b(1,k1)*b(n,n-1)-b(1,n-1)*b(n,k1);
AXKN= b(1,2)*b(n,k1)-b(1,k1)*b(n,2);
PDD(ij,kj)=e4(i,k1)-((AXK1)*e4(i,2)+(AXKN)*e4(i,n-))/AXN;
end
for k2=3:n-2
ik=(n-4)*(i-3)+k2-3+1;
AYK1= b(1,k2)*b(n,n-1)-b(1,n-1)*b(n,k2);
AYKN= b(1,2)*b(n,k2)-b(1,k2)*b(n,2);
PDD(ij,ik)=PDD(ij,ik)+e4(j,k2)-(e4(j,2)*(AYK1)+e4(j,n-
1)*(AYKN))/AYN;
end
end
end
for i = 3 : n-2
for j = 3 : n-2
ij=(n-4)*(i-3)+(j-3)+1;
for k1=3:n-2
for k2=3:n-2
ik=(n-4)*(k1-3)+(k2-3)+1;
AXK1= b(1,k1)*b(n,n-1)-b(1,n-1)*b(n,k1);
AXKN= b(1,2)*b(n,k1)-b(1,k1)*b(n,2);
AYK1= b(1,k2)*b(n,n-1)-b(1,n-1)*b(n,k2);
AYKN= b(1,2)*b(n,k2)-b(1,k2)*b(n,2);
r1=b(i,k1)*b(i,k2)-((AXK1*b(i,2)+AXKN*b(i,n-
1))/AXN)*b(j,k2);
r2=((AYK1*b(j,2)+AYKN*b(j,n-1))/AYN)*b(i,k1);
```

```
r3=((AXK1*AYK1*b(i,2)*b(j,2)+AXKN*AYK1*b(i,n-
1)*b(j,2))/AXN*AYN);
r4=((AXK1*AYKN*b(i,2)*b(j,n-1)+AXKN*AYKN*b(i,n-1)*b(j,n-
1))/AXN*AYN);
PDD(ij,ik)=PDD(ij,ik)+2*(b(i,k1)*b(j,k2)-
(AXK1*b(i,2)*(b(j,k2)-b(j,2)*AYK1/AYN)+AXKN*b(i,n-
1)*(b(j,k2)-b(j,2)*AYK1/AYN))/AXN-
(AYK1*b(j,2)*b(i,k1)+AYKN*b(j,n-1)*(b(i,k1)-
(b(i,2)*AXK1+b(i,n-1)*AXKN)/AXN))/AYN);
end
end
end
end
v1=(eig(PDD));
for i=1:(n-4)^2
v2(i)=v1(i)^.5;
end
v3=sort(v2);
```

program 13:-

```
n=6;
a=[-31.089941056156209505,-
6.23057741217772963,0.32968683595852671497,-
18.453027890343230902,-67.76557439272856797,-
12.737478218682955771;31.462632673736228255,8.55664669146
90994018,-3.1589433681645365545,-
39.413189727876957288,142.02076329965116237,26.5420617342
75329012;-39.100582338249030684,-12.681462357270821679,-
1.9816450900580773047,67.096870399794979987,-
198.92631460106778661,-35.97074306959821994;-
6.857165106827839516,-2.9264003479616642098,-
1.0132240340995923634,15.572244071285122453,-
50.939335688638180131,-8.5384766821064232152;-
4.2976118828138608537,-1.66666005433389088889,-
0.82218507370751093234,11.066403445120947639,-
18.526860515951349241,-
2.7261574346852199537;20.961890720669004935,8.39639816099
71580041,3.7648635847823997444,-
46.053352732815160677,124.67113591572975694,22.1661595540
054397];
b=a*a;
c=a*b;
e4=a*c;
for i = 3 : n-2
for j = 3 : n-2
ij=(n-4)*(i-3)+(j-3)+1;
AXN=-1*( a(n,2)*a(1,n-1)-a(1,2)*a(n,n-1));
AYN= -1*(a(n,2)*a(1,n-1)-a(1,2)*a(n,n-1));
for k1=3:n-2
kj=(n-4)*(k1-3)+j-3+1;
AXK1= a(1,k1)*a(n,n-1)-a(1,n-1)*a(n,k1);
```

```
AXKN= a(1,2)*a(n,k1)-a(1,k1)*a(n,2);
PDD(ij,kj)=e4(i,k1)-((AXK1)*e4(i,2)+(AXKN)*e4(i,n-
1))/AXN;
end
for k2=3:n-2
ik=(n-4)*(i-3)+k2-3+1;
AYK1=a(1,k2)*a(n,n-1)-a(1,n-1)*a(n,k2);
AYKN= a(1,2)*a(n,k2)-a(1,k2)*a(n,2);
PDD(ij,ik)=PDD(ij,ik)+e4(j,k2)-(e4(j,2)*(AYK1)+e4(j,n-
1)*(AYKN))/AYN;
end
end
end
for i = 3 : n-2
for j = 3 : n-2
ij=(n-4)*(i-3)+(j-3)+1;
for k1=3:n-2
for k2=3:n-2
ik=(n-4)*(k1-3)+(k2-3)+1;
AXK1= a(1,k1)*a(n,n-1)-a(1,n-1)*a(n,k1);
AXKN= a(1,2)*a(n,k1)-a(1,k1)*a(n,2);
AYK1= a(1,k2)*a(n,n-1)-a(1,n-1)*a(n,k2);
AYKN= a(1,2)*a(n,k2)-a(1,k2)*a(n,2);
r1=b(i,k1)*b(i,k2)-((AXK1*b(i,2)+AXKN*b(i,n-
1))/AXN)*b(j,k2);
r2=((AYK1*b(j,2)+AYKN*b(j,n-1))/AYN)*b(i,k1);
r3=((AXK1*AYK1*b(i,2)*b(j,2)+AXKN*AYK1*b(i,n-
1)*b(j,2))/AXN*AYN);
r4=((AXK1*AYKN*b(i,2)*b(j,n-1)+AXKN*AYKN*b(i,n-1)*b(j,n-
1))/AXN*AYN);
PDD(ij,ik)=PDD(ij,ik)+2*(b(i,k1)*b(j,k2)-
(AXK1*b(i,2)*(b(j,k2)-b(j,2)*AYK1/AYN)+AXKN*b(i,n-
1)*(b(j,k2)-b(j,2)*AYK1/AYN))/AXN-
(AYK1*b(j,2)*b(i,k1)+AYKN*b(j,n-1)*(b(i,k1)-
(b(i,2)*AXK1+b(i,n-1)*AXKN)/AXN))/AYN);
end
end
end
end
v1=(eig(PDD));
for i=1:(n-4)^2
v2(i)=v1(i)^.5;
end
v3=sort(v2);
```

Fundamentals G-Spline Basis

$$L_{10} = 1 - 16.713x + 78.366x^2 - 143.193x^3 + 93.468x^4 - 55.423(x-0)_+^9 + 110.86(x-.0955)_+^9 - 110.985(x-.3456)_+^9 + 111.195(x-.6546)_+^9 - 111.41(x-.9046)_+^9 + 55.763(x-1)_+^9$$

$$L_{20} = 20.379x - 127.056x^2 + 257.962x^3 - 177.142x^4 + 177.142(x-0)_+^9 - 221.75(x-.0955)_+^9 + 221.999(x-.3456)_+^9 - 222.419(x-.6546)_+^9 + 222.85(x-.9046)_+^9 - 111.541(x-1)_+^9$$

$$L_{30} = -5.24x + 72.038x^2 - 192.538x^3 + 151.627x^4 - 110.985(x-0)_+^9 + 221.999(x-.0955)_+^9 - 222.249(x-.3456)_+^9 + 222.669(x-.6546)_+^9 - 223.101(x-.9046)_+^9 + 111.666(x-1)_+^9$$

$$L_{40} = 2.519x - 37.745x^2 + 129.453x^3 - 120.173x^4 + 111.195(x-0)_+^9 - 222.419(x-.0955)_+^9 + 222.669(x-.3456)_+^9 - 223.091(x-.6546)_+^9 + 223.523(x-.9046)_+^9 - 111.877(x-1)_+^9$$

$$L_{50} = -1.681x + 25.66x^2 - 92.759x^3 + 94.766x^4 - 111.41(x-0)_+^9 + 222.85(x-.0955)_+^9 - 223.101(x-.3456)_+^9 + 223.523(x-.6546)_+^9 - 223.956(x-.9046)_+^9 + 112.094(x-1)_+^9$$

$$L_{60} = .737x - 11.273x^2 + 41.076x^3 - 42.545x^4 + 55.763(x-0)_+^9 - 111.541(x-.0955)_+^9 + 111.666(x-.3456)_+^9 - 111.877(x-.6546)_+^9 + 112.094(x-.9046)_+^9 - 56.105(x-1)_+^9$$

$$L_{10} = 1 - 17.597x + 90.409x^2 - 174.268x^3 + 116.155x^4 - 42.812(x-0)_+^9 + 68.302(x-0.0955)_+^9 - 25.613(x-0.3458)_+^9 - 26.239(x-0.6546)_+^8 + 6.394(x-0.9046)_+^8 + 0.122(x-1)_+^9$$

$$L_{20} = 22.056x - 150.009x^2 + 318.05x^3 - 221.722x^4 + 86.24(x-0)_+^9 - 137.588(x-0.0955)_+^9 + 51.595(x-0.3458)_+^9 + 52.856(x-0.6546)_+^8 - 12.881(x-0.9046)_+^8 - 2.221(x-1)_+^9$$

$$L_{30} = -7.198x + 99.57x^2 - 271.589x^3 + 216.346x^4 - 101.248(x-0)_+^9 + 161.532(x-0.0955)_+^9 - 60.573(x-0.3458)_+^9 - 62.054(x-0.6546)_+^8 + 15.123(x-0.9046)_+^8 + 0.29(x-1)_+^9$$

$$L_{41} = -0.752x + 11.049x^2 - 36.161x^3 + 32.963x^4 - 19.359(x-0)_+^9 + 30.886(x-0.0955)_+^9 - 11.582(x-0.3458)_+^9 - 11.865(x-0.6546)_+^8 + 2.892(x-0.9046)_+^8 + 0.055(x-1)_+^9$$

$$L_{51} = -0.606x + 8.835x^2 - 28.158x^3 + 24.221x^4 - 11.705(x-0)_+^9 + 18.674(x-0.0955)_+^9 - 7.002(x-0.3458)_+^9 - 7.174(x-0.6546)_+^8 + 1.748(x-0.9046)_+^8 + 0.033(x-1)_+^9$$

$$L_{60} = 2.739x - 39.97x^2 + 127.807x^3 - 110.779x^4 + 57.82(x-0)_+^9 - 92.246(x-0.0955)_+^9 + 34.591(x-0.3458)_+^9 + 35.437(x-0.6546)_+^8 - 8.636(x-0.9046)_+^8 - 0.165(x-1)_+^9$$

المستخلص

الغرض الرئيسي لهذه الرسالة يدور حول هدفين:

١- الهدف الاول هو حول دراسة الخلفية الرياضية لطريقة التفاضلات التربيعية وطريقة تكوينها ومن ثم استخدامها لحل معادلات تفاضلية حدودية اعتيادية من الرتبة الرابعة.

٢- الهدف الثاني هو اولا حول تقريب الدوال باستخدام طريقة دوال السبلاين - G.
ثانيا الحل العددي لمسئلتين لتطبيقين هما اهتزاز عمود منتظم ممثلة على شكل معادلة تفاضلية حدودية اعتيادية من الرتبة الرابعة واهتزاز صفيحة سميكة مربعة معطاة على شكل معادلة تفاضلية حدودية جزئية من الرتبة الرابعة باستخدام طريقة التفاضلات التربيعية وبالاعتماد على دوال السبلاين -G ثم الحصول عليها.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم
قسم الرياضيات وتطبيقات الحاسوب

الحلول العددية للمعادلات التفاضلية باستخدام طريقة التفاضلات التربيعية المعتمدة على دوال السبلاين– G

رسالة
مقدمة الى قسم الرياضيات وتطبيقات الحاسوب، كلية العلوم جامعة النهرين
وهي جزء من متطلبات نيل درجة ماجستير علوم في الرياضيات

من قبل
مصطفى اكرم سعيد
(بكلوريوس رياضيات / كلية العلوم / جامعة النهرين، ٢٠١٠)

باشراف

د.أسامة حميد محمد
(أ.م.)

د. فاضل صبحي فاضل
(أ.م.)

صفر
١٤٣٤

كانون الأول
٢٠١٢