FINITE ELEMENT METHOD WITH QUASI-LINEARIZATION FOR SOLVING BRATU'S PROBLEM

A thesis submitted to the University of Zimbabwe
in partial fulfillment of the degree of Master of Science
in the Faculty of Science

By
Hillary Muzara
Supervisor: Dr. G. T. Marewo
Department of Mathematics
June 2015

Contents

List of Figures

Abstract

This work presented here is the solution of one-dimensional Bratu's problem. The major aim of this research is to master the techniques used to solve the Bratu problem. The nonlinear Bratu's problem is first linearised using the quasi-linearization method and then solved by the finite element method using

1. piecewise linear Lagrange polynomials as basis functions
2. piecewise quadratic Lagrange polynomials as basis functions and
3. hierarchical basis functions.

Unlike other basis functions like the trigonometric functions, the three basis functions used in this research have an advantage that they have small local support, that is, they are only non-zero on a small portion of the given domain. A comparison of the exact solution and the finite element solutions using `Matlab` plots and tabulated results is made. The finite element solutions are validated using both `Matlab's bvp4c` and the Chebyshev spectral collocation method.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Dedication

This work is dedicated to my beloved brother, Happymore.

Acknowledgements

My heartfelt gratitude is extended to the following for their contribution towards the success of this work

- My supervisor, Dr G.T. Marewo for his unmeasurable and tireless support and guidance during the course of the study
- All the mathematics department staff at University of Zimbabwe who assisted me to make this work a success
- My family and friends for their financial and moral support
- God, the Almighty, for the precious gift of life

Chapter 1

Introduction

1.1 Introduction to problem

There are many nonlinear phenomena which are of great importance in various fields of science and engineering. Examples of the nonlinear models that are used in applications include the fuel ignition model of the thermal combustion theory, the model of thermal reaction process, the Chandrasekhar model of the expansion of the universe, questions in geometry and relativity about the Chandrasekhar model, chemical reaction theory, radiative heat transfer and nanotechnology [1, 2, 3]. Such models are classified as Bratu's boundary value problem. In one dimensional planar coordinates, Bratu's model has the form

$$u''(x) + \lambda e^{u(x)} = 0,\ 0 < x < 1 \tag{1.1}$$

together with boundary conditions $u(0) = u(1) = 0$ where λ is a constant. For $\lambda > 0$, this problem has an exact solution which from literature [9, 10, 11, 12] is given by

$$u(x) = -2\ln\left(\frac{\cosh((x-\frac{1}{2})\frac{\theta}{2})}{\cosh(\frac{\theta}{4})}\right) \tag{1.2}$$

where θ is a solution of the equation $\theta = \sqrt{2\lambda}\cosh(\frac{\theta}{4})$. There exists λ_c such that Bratu's problem has no solution when $\lambda > \lambda_c$. It has one solution when $\lambda = \lambda_c$ and two solutions when $\lambda < \lambda_c$ where the critical value $\lambda_c = 3.513830719$ [6, 7] satisfies

$$1 = \frac{1}{4}\sqrt{2\lambda_c}\sinh\left(\frac{\theta_c}{4}\right). \tag{1.3}$$

In n-dimensional coordinates, Bratu's model has the form

$$\Delta u(\mathbf{x}) = -\lambda e^{u(\mathbf{x})},\ \mathbf{x} \in \mathbf{D} \tag{1.4}$$

$$u(\mathbf{x}) = 0,\ \mathbf{x} \in \delta\mathbf{D} \tag{1.5}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, $\Delta = (\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \ldots, \frac{\partial}{\partial x_n})$ is the Laplace operator, $\mathbf{D} \subset \mathbb{R}^n$and $\delta\mathbf{D}$ denotes the boundary of the problem domain $\mathbf{D}$.

In this work we restrict the Bratu problem only to the one-dimensional case. Much work has been done by researchers to solve Bratu's problem. The Adomain decomposition method (ADM) [4, 5] which approximates the analytical solutions in the form of an infinite power series is an example method of solution. Some of the methods which have been used to solve the Bratu problem include the weighted residual method [6], the shooting method [7] and the Sinc-Garlekin method [8]. In this work we use a combination of the finite element method (FEM) and an iterative method to find the numerical solution of the Bratu problem. The results obtained are validated using `Matlab`'s in-built routine `bvp4c` and the Chebyshev spectral collocation method.

1.2 Overview of the numerical methods

1.2.1 Quasi-linearization

The quasi-linearization method (QLM) whose origins are in the theory of dynamic programming was first proposed by Bellman and Kalaba [13]. This method can be viewed as the Newton-Raphson

method applied to nonlinear differential equations. It is a very powerful method for approximating solutions of nonlinear differential equations and makes use of the Taylor series expansion of first order to linearise a nonlinear differential equation. The solution is then approximated as a sequence of the linear equations. Originally, the method was restricted to twice differentiable and strictly concave (or convex) functions. However, great work was done by Lakshmikantham [14] who presented the QLM with the concavity assumption relaxed. This made the QLM applicable to a wider variety of problems.

1.2.2 Finite element methods

The finite element method (FEM) is a computational technique used to obtain approximate solutions of boundary value problems (BVPs) in science and engineering [15]. Many engineering phenomena can be modelled by differential equations together with boundary conditions. In many practical problems the governing equation and the domain are usually complex making it very difficulty to come up with the exact solution of the differential equation, hence the need for approximation of solutions using numerical techniques and digital computations.

The history of the finite element method can be traced back to 1909 when Ritz introduced a method of obtaining approximate solutions of problems in deformable solids. The method included the approximation of the energy functional of known functions with unknown coefficients but had a disadvantage that the functions used had to satisfy the boundary conditions of the problem. The method of Ritz was improved in 1943 by Courant when he introduced a special type of linear functions defined on triangular regions to approximate solutions of torsion problems [16].

Many years later Ray Clough who introduced the term "finite element method" for the first time in 1960 in his paper [17] marked the beginning of the finite element method. The FEM spread widely after 1960 due to the introduction of digital computers. Some other work important in the

development of FEM include the papers by Argyris [18], Turner [19], Hrennikov [20] and many others.

When analysing the behaviour or properties of a complex system, it is easy to consider it as an assemblage of simple elements, then dismantle it and analyse the properties of its elements individually. This is exactly the principle used in the finite element method. The dismantling done in the FEM is the meshing of the physical domain into sub-domains called finite elements. Each element will have interior and exterior nodes, which are points at which the dependent variables will be computed explicitly. Exterior nodes are found at the boundary of the finite element and are used for joining one element to another. The approximate solution will be obtained on the finite element using the nodal values as unknowns and predetermined interpolation functions. The finite element equations are formulated in such a way that the values at each exterior node is the same for the other connecting element hence maintaining continuity.

The basic steps for solving a differential problem using the finite element method are;

1. formulation of the problem in variational form,
2. the finite element dicretization of this formulation and
3. the solution of the resulting finite element equations.

1.2.3 Spectral collocation methods(CSCM)

Spectral methods which became famous in the 1970s are one of the very accurate numerical methods used to solve ordinary and partial differential equations numerically. Spectral methods can achieve upto ten digits of accuracy in problems where other numerical methods like the FEM and finite differences achieve only two or three digits of accuracy [21]. There are mainly three types of

spectral methods which can be identified as collocation, tau and the Garlekin methods [22, 23, 24] distinguished by the type of the trial and test functions used. The choice of the spectral method to use mainly depends on the application. The tau method, discovered by Lanczos [25] is most suited for problems which are non-periodic with complicated boundary conditions. In the Garlekin method, the test and trial functions are considered to be the same. In this work we use the collocation method which is most suited for non-linear problems. The spectral collocation method whose motivation is the finite difference method is in principle similar to the finite element method (FEM) due to the use of basis functions. The major difference is that the finite element method uses local basis functions which have small support, that is, the basis functions are non-zero on small portions of the problem domain whilst the global basis functions used in spectral collocation method are non-zero on the entire domain.

The CSCM has been used to solve the Lane-Emden equation [44] to get more accurate results than those obtained using other methods such as pertubative and nonpertubative techniques [46, 47], quasilinearisation method [48] and Adomain decomposition method [49]. In this work we intend to use the CSCM because of its ability to achieve higher accuracy, given the same grid points, than other methods like the FEM and finite difference methods [45]. It also has excellent error properties with exponential convergence being the fastest possible [50]. However, the CSCM is more difficult to code and can result in heavy loss of accuracy for complicated domains.

Chapter 2

Numerical methods

2.1 Introduction

In this chapter we shall discuss the linearization of equation (1.1) using the quasi-linearisation method. Also presented are the solutions of equation (1.1) using:

1. finite element method with piecewise linear Lagrange polynomials as basis functions
2. finite element method with piecewise quadratic Lagrange polynomials as basis functions
3. finite element method with hierarchical basis functions
4. `Matlab bvp4c` and
5. Chebyshev spectral collocation method.

2.2 Spectral quasi-linearization method

We begin by presenting problems that were solved by other authors using the quasi-linearization method.

Examples of applications of QLM

Example 1. The Lane-Emden equation given by

$$u''(x) + \frac{2}{x}u'(x) + u^m(x) = 0,\ u(0) = 0,\ u'(0) = 0,\ 0 \leq m \leq 5 \tag{2.1}$$

is an equation which occurs in stellar structure [26]. It can be solved analytically for the indices $m = 0$, $m = 1$ and $m = 5$ but is unsolvable analytically when $m = 4$. By making the substitution $u = y/x$ and applying the QLM, Mandelzweig and Tabakin [29], managed to transform the boundary value problem (2.1) to the problem of constructing a sequence $\{u_s\}$ that satisfies

$$u''_{s+1}(x) + m\frac{u_s^{m-1}}{x^{m-1}}u_{s+1}(x) = \frac{m-1}{x^{m-1}}u_s^m(x),\ u_{s+1}(0) = 0,\ u'_{s+1}(0) = 1 \tag{2.2}$$

where $s = 0, 1, 2, \ldots$ and with an initial approximation $u_0(x) = x$, this sequence converges rapidly to the exact solution.

Example 2. The Thomas-Fermi equation

$$\sqrt{x}u''(x) = u^{\frac{3}{2}}(x),\ u(0) = 1,\ u(\infty) = 0 \tag{2.3}$$

is widely used in nuclear Physics. This equation is very difficult to solve because $u''(x) = 0$ when $u < 0$ and also that the solution is very sensitive to the first derivative of the solution at zero. However, it is shown in [29] that by the QLM the resultant iterative scheme

$$\sqrt{x}u''_{s+1}(x) - \frac{3}{2}u_s^{\frac{1}{2}}(x)u_{s+1}(x) = \frac{1}{2}u_s^{\frac{3}{2}}(x),\ u_{s+1}(0) = 1,\ u_{s+1}(\infty) = 0 \tag{2.4}$$

can be easily solved by specifying the boundary condition at infinity directly.

There are many other methods which can be used to solve nonlinear differential equations with the two mostly widely used being the shooting method and the finite difference method [27]. The shooting method [28] fails if one of the solutions of the differential equations under consideration is highly unstable. On the other hand, approximating the solution of a nonlinear system of differential equations using finite difference methods usually results in nonlinear algebraic systems which are difficult to solve. Other methods of solving nonlinear problems include the monotone iterative technique [32, 33, 34] and pertubation techniques [35]. Pertubation techniques also provide a powerful tool of obtaining solutions of nonlinear differential equations but are only appropriate for weakly nonlinear differential equations due to their strong dependence on some small parameters in the equations under consideration.

In this work we intend to use the QLM to solve equation (1.1) because it produces a linear iterative scheme whose iterates converge monotonically. From this scheme we can form a sequence of solutions $u_s(x)$, $s = 0, 1, 2, ...$, which has been shown [36] to have the following properties:

1. The sequence $u_s(x)$ is bounded below and above by $u_0(x)$ (initial solution) and $u(x)$ respectively, and that
2. $u_{s+1}(x) - u_s(x) \geq 0$, $s = 0, 1, 2, ...$

From properties 1 and 2 above, the iterative scheme is bounded and monotone increasing. Using the monotone convergence theorem [37], it can be deduced that the iterative scheme is convergent. The scheme also converges uniformly and quadratically to the solution of the original problem [38]. This iterative scheme produces successive approximations of the solution. For quadratic, monotone and uniform convergence of the QLM to the solution of the problem in question we refer the reader to reference [29]. Another desirable attribute of the QLM is that it is numerically stable as compared to other methods of approximating solutions of nonlinear differential equations [13]. If the initial guess given is close to the true solution, the method converges rapidly to the true solution [31].

Derivation of the QLM formula

Let us consider an n^{th} order nonlinear differential equation of the form

$$\mathbf{F}[\mathbf{u}(x)] = 0, \quad x \in [a, b] \tag{2.5}$$

where x is an independent variable and $\mathbf{u}(x) = (u, u', ..., u^{(n)})$ is a vector of solutions of (2.5). Let $u' = \frac{du}{dx}$ and $u^{(n)} = \frac{d^n u}{dx^n}$, for $n = 2, 3, ...$ As in [30] it is assumed that $\mathbf{z} = (z, z', ..., z^{(n)})$ is an approximate solution of (2.5) which is sufficiently close to the true solution $\mathbf{u}$. Assuming that all the partial derivatives of $\mathbf{F}$ exists, applying Taylor's theorem we get

$$\mathbf{F}[\mathbf{u}] = \mathbf{F}(\mathbf{z}) + \nabla F(\mathbf{z}).(\mathbf{u} - \mathbf{z}) + (\text{higher order terms}) \tag{2.6}$$

Upon ignoring higher order terms equation (2.6) becomes

$$\nabla \mathbf{F}(\mathbf{z}).\mathbf{u} = \nabla \mathbf{F}(\mathbf{z}).\mathbf{z} - \mathbf{F}(\mathbf{z}) \tag{2.7}$$

The solution from (2.7) will not be, generally, the exact solution of (2.5) because of the discarded higher order terms. We will use the initial approximate solution $\mathbf{z}$ as a calculated solution to iteratively compute the new solution $\mathbf{u}$. With this in mind, denote $\mathbf{z}$ and $\mathbf{u}$ by $\mathbf{u}_s$ and $\mathbf{u}_{s+1}$ respectively to get the iterative formula

$$\nabla \mathbf{F}(\mathbf{u_s}).\mathbf{u_{s+1}} = \nabla \mathbf{F}(\mathbf{u_s}).\mathbf{u_s} - \mathbf{F}(\mathbf{u_s}) \tag{2.8}$$

where $s = 0, 1, 2, \ldots$. Since

$$\nabla \mathbf{F}(\mathbf{u_s}).\mathbf{u}_{s+1} = \frac{\partial F(\mathbf{u_s})}{\partial u_s} u_{s+1} + \frac{\partial F(\mathbf{u_s})}{\partial u'_s} \frac{d}{dx}(u_{s+1}) + ... + \frac{\partial F(\mathbf{u_s})}{\partial u_s^{(n-1)}} \frac{d^{n-1}}{dx}(u_{s+1}) + \frac{\partial F(\mathbf{u_s})}{\partial u_s^{(n)}} \frac{d^n}{dx}(u_{s+1})$$

then equation (2.8) can be written in operator form as

$$\mathcal{L} u_{s+1} = \mathcal{L} u_s - F(\mathbf{u}_s) \tag{2.9}$$

where

$$\mathcal{L} = b_0 \frac{d^n}{dx^n} + b_1 \frac{d^{n-1}}{dx^{n-1}} + ... + b_{n-1} \frac{d}{dx} + b_n \tag{2.10}$$

and

$$b_0 = \frac{\partial F(u_s)}{\partial u_s^{(n)}},\ b_1 = \frac{\partial F(u_s)}{\partial u_s^{(n-1)}}, ...,\ b_{n-1} = \frac{\partial F(u_s)}{\partial u_s'} \text{ and } b_n = \frac{\partial F(u_s)}{\partial u_s}$$

The iterative scheme (2.9) is the standard QLM formula used to obtain the $(s+1)$th iterative approximation $u_{s+1}(x)$ of the solution of (2.5).

2.2.1 Application to the Bratu problem

The Bratu problem (1.1) can be transformed to a linear differential problem using the QLM. Equation (1.1) is of second order, thus we have

$$F(u, u', u'') = u''(x) + \lambda e^{u(x)}$$

and

$$\mathcal{L} = b_0 \frac{d^2}{dx^2} + b_1 \frac{d}{dx} + b_2$$

Calculating the coefficients b_0, b_1 and b_2 and substituting into (2.9) we get the iterative scheme

$$u''_{s+1}(x) + \lambda e^{u_s(x)} u_{s+1}(x) = \lambda e^{u_s(x)}(u_s(x) - 1) \tag{2.11a}$$

$$u_{s+1}(0) = u_{s+1}(1) = 0 \tag{2.11b}$$

where $s = 0, 1, 2, ...$ Equation (2.11a) can be used to compute $u_{s+1}(x)$ provided $u_s(x)$ is known. In particular, the initial approximation $u_0(x)$ must be specified so that we compute $u_1(x)$. Once $u_1(x)$ is known, we compute $u_2(x)$ using equation (2.11a) and so on. Also, $u_0(x)$ must satisfy boundary conditions (2.11b). For the sake of brevity, we replace equations (2.11a) and (2.11b) with equations

$$\mathcal{L}(u) = u''(x) + a(x)u(x) = b(x), 0 < x < 1 \tag{2.12a}$$

$$u(0) = u(1) = 0. \tag{2.12b}$$

2.3 Finite element method

2.3.1 Piecewise linear lagrange finite element solution

Variational formulation of the differential problem

To implement the finite element method to the linear differential equation (2.12a) subject to boundary conditions (2.12b), it has to be first converted into its variational or integral form. This is done by making use of the fundamental lemma of variational calculus [39]

Lemma 1. If a function $u(x)$ is continuous and if $\int_a^b u(x)v(x)dx = 0$, for all continuous functions $v(x)$, then $u(x) \equiv 0$ for $a \leq x \leq b$.

Relative to our problem (2.12a), the function $u(x)$ is the trial solution to the problem and the function $v(x)$ is the weight or test function which is arbitrarily chosen so that it satisfies the same boundary conditions as the trial solution. From lemma 1, requiring the trial solution $u(x) \equiv 0$ is the same as taking the residual function $R(x) = \mathcal{L}(u) - b(x)$, to be zero for $0 \leq x \leq 1$. Hence from equation (2.12a) we have

$$\int_0^1 \left[\mathcal{L}(u) - b(x)\right] v(x)dx = 0, \text{ for all continuous functions v(x) }, \tag{2.13}$$

or

$$A(v, u) = (v, b), \text{ for all continuous functions v(x)}, \tag{2.14}$$

where the bilinear functional

$$A(v, u) = \int_0^1 [v(x)u''(x) + v(x)a(x)u(x)]dx, \tag{2.15}$$

is usually called the strain energy and the L^2 inner product

$$(v, b) = \int_0^1 v(x)b(x)dx$$

is an $\mathcal{L}^2$ inner product. In the strain energy, the product $v(x)u''(x)$ causes different smoothness requirements for the two functions. Symmetry is however introduced by using the integration by parts formula in one dimension to get:

$$A(v,u) = \int_0^1 \big[-v'(x)u'(x) + v(x)a(x)u(x)\big]dx - u(x)'v(x)\big|_0^1. \tag{2.16}$$

Since the solution $u(x)$ is known to be zero at both endpoints $x = 0$ and $x = 1$, the test function is chosen so that it satisfies the same trivial boundary conditions hence making the boundary term to vanish. Now (2.16) takes the form

$$A(v,u) = \int_0^1 \big[-v'(x)u'(x) + v(x)a(x)u(x)\big]dx. \tag{2.17}$$

Integration by parts has added a derivative to the function $v(x)$ so that its selection is restricted to a space where the functions have more continuity than those in $\mathcal{L}^2$. The functions $u(x)$ and $v(x)$ will be required to be elements of the Sobolev space H^1, which is a space of continuous functions where

$$\int_0^1 [(u'(x))^2 + (u(x))^2]dx < \infty.$$

Since the functions $u(x)$ and $v(x)$ satisfy the trivial boundary conditions (2.12b), they are elements of the Sobolev space H_0^1. The variational problem becomes that of determining $u(x) \in H_0^1$ satisfying

$$A(v,u) = (v,b), \quad \forall v(x) \in H_0^1 \tag{2.18}$$

where

$$H_0^1 = \left\{ u(x) \middle| \int_0^1 [(u'(x))^2 + (u(x))^2]dx < \infty, u(0) = 0 \text{ and } u(1) = 0 \right\}.$$

Problem variable approximation

The problem variable approximation is

$$u(x) \approx U(x) = \sum_{c=0}^{E} u_c \phi_c(x) \tag{2.19}$$

where $\phi_c(x),\ \ c = 0, 1, 2, \cdots, E-1, E$ are arbitrarily chosen continuous piecewise linear functions. Since $U(0) = U(E) = 0$, equation (2.19) can be written as

$$U(x) = \sum_{c=1}^{E-1} u_c \phi_c(x) \tag{2.20}$$

Substituting (2.20) into equation (2.18) gives

$$\sum_{c=1}^{E-1} u_c A(v, \phi_c(x)) = (v, b(x)), \text{ for all v : v(0)=v(1)=0} \tag{2.21}$$

which upon choosing functions $E-1$ distinct values $v_1(x), v_2(x), \ldots, v_{E-1}(x)$ of the test function $v(x)$ becomes

$$\sum_{c=1}^{E-1} u_c A(v_d, \phi_c(x)) = (v_d, b(x)),\ d = 1, 2, \cdots, E-1$$

which is a linear system of $E-1$ equations of unknowns u_c, $c = 1, 2, \ldots, \ldots E-1$ since $A(v_d(x), \phi_c(x))$ and $(v_d(x), b(x))$, $d = c = 1, 2, \ldots, E-1$ are constants. The system can be written as

$$(M - K)\mathbf{u} = \mathbf{w} \tag{2.22}$$

where $\mathbf{u} = [u_1, u_2, \ldots, u_{E-1}]^T$ and

$$K = \int_0^1 \begin{pmatrix} v_1'\phi_1' & v_1'\phi_2' & \cdots & v_1'\phi_{E-1}' \\ v_2'\phi_1' & v_2'\phi_2' & \cdots & v_2'\phi_{E-1}' \\ \vdots & \vdots & \ddots & \vdots \\ v_{E-1}'\phi_1' & v_{E-1}'\phi_2' & \cdots & v_{E-1}'\phi_{E-1}' \end{pmatrix} dx \tag{2.23}$$

is the global stiffness matrix with each of its elements given by

$$K_{dc} = \int_0^1 v_d'\phi_c' dx, \text{ where } d, c = 1, 2, \ldots, E-1$$

The elements of the global mass matrix M and the global load vector $\mathbf{w}$ are

$$M_{dc} = \int_0^1 v_d a(x) \phi_c dx \text{ and } w_d = \int_0^1 v_d b dx \tag{2.24}$$

respectively. The values d and c represent the row and column number, respectively, of each entry in the global matrix. Upon solving the linear algebraic system (2.22) we determine the coefficients $u_c,\ c = 1, 2, \cdots, E-1$ for the approximate solution (2.20). The choice of the functions $\phi_c(x),\ c = 1, 2, \cdots, E-1$ determines how good (2.20) is as an approximation of the solution $u(x)$.

Considering the general element $\Omega_e = [x_{c-1}, x_c]$

$$\begin{aligned} K_{dc} &= \int_0^1 v_d' \phi_c' dx \\ &= \sum_{e=1}^{E} \int_{x_{c-1}}^{x_c} v_d' \phi_c' dx \\ &= \sum_{e=1}^{E} K_{dc}^e, \end{aligned}$$

where

$$K_{mn}^e = \begin{cases} \int_{x_{c-1}}^{x_c} v_m'(x)\phi_n'(x)dx & \text{where } m \text{ and } n \text{ are either } c \text{ or } c-1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.25}$$

Similarly

$$M_{dc} = \sum_{e=1}^{E} M_{dc}^e \quad \text{and} \quad \mathbf{w}_d = \sum_{e=1}^{E} \mathbf{w}_d^e,$$

with

$$M_{mn}^e = \begin{cases} \int_{x_{c-1}}^{x_c} v_m(x)a(x)\phi_n(x)dx & \text{where } m \text{ and } n \text{ are either } c \text{ or } c-1 \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\mathbf{w}_m^e = \begin{cases} \int_{x_{c-1}}^{x_c} v_m(x)b(x)dx & \text{where } m \text{ is either } c \text{ or } c-1 \\ 0 & \text{otherwise,} \end{cases}$$

respectively.

Discretizing problem domain and choosing basis functions

The problem domain $[0, 1]$ is partitioned into uniform discrete subintervals $[x_{c-1}, x_c],\ c = 1, 2, 3, \cdots, E$ called finite elements, where

$$0 = x_0 < x_1 < \cdots < x_{E-1} < x_E = 1.$$

The uniform width of each interval is $h = x_c - x_{c-1},\ c = 1, 2, 3, ..., E$ with each endpoint of the subinterval called a node. Each basis function $\phi_c(x)$ is chosen to be a piecewise linear Lagrange polynomial. These polynomials should be continuous and piecewise linear on $[0, 1]$. Such class of polynomials are chosen to be the hat functions

$$\phi_c(x) = \begin{cases} \dfrac{x - x_{c-1}}{h}, & \text{if } \ x \in [x_{c-1}, x_c] \\ \dfrac{x_{c+1} - x}{h}, & \text{if } \ x \in [x_c, x_{c+1}] \\ 0, & \text{otherwise} \end{cases} \tag{2.26}$$

which have the form shown in Figure 2.1

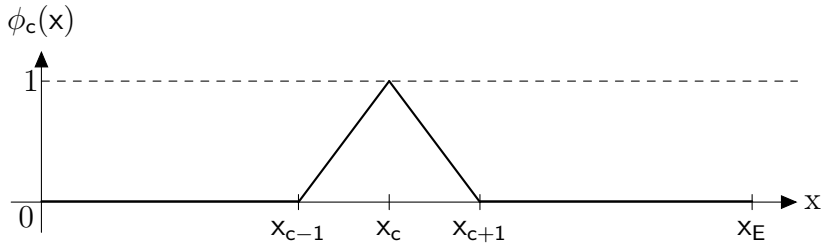


Figure 2.1: The hat function $\phi_c(x)$

The most desirable properties about these functions are that $\phi_c(x)$ has a value of unity at the node c, vanishing at all other nodes. This makes the determination of solutions at the node simple. Secondly, $\phi_c(x)$ is nonzero on the elements containing the node c and zero elsewhere on $[0, 1]$ hence simplifying the solution of the resulting algebraic system. Such functions are said to have local

support.

Computation of element matrices

The variational problem counterpart of equation (2.18) is now constructed by replacing the function $u(x)$ by its approximation given by (2.20) and then choose each test function $V(x) = \phi_r(x)$ by the Garlekin method. The functions $U(x)$ and $V(x)$ belong to a finite dimensional subspace S_0^N of the Sobolev space H_0^1. The basis of the space S_0^N is formed by the functions $\phi_c(x), c = 1, 2, \cdots, E-1$. The variational counterpart problem now consists of determining $U(x)$ satisfying

$$A(V, U) = (V, b), \forall V(x) \in S_0^N. \tag{2.27}$$

Restricting (2.20) to a finite element Ω_e yields the approximate solution

$$U(x) = u_{c-1}\phi_{c-1}(x) + u_c\phi_c(x),\ x \in \Omega_e, \tag{2.28}$$

since $\phi_{c-1}(x)$ and $\phi_c(x)$ are the only nonzero basis functions on Ω_e. These basis functions are both unity at the nodes x_{c-1} and x_c respectively hence we deduce that $U(x_c) = u_c$. Since by the Garlekin method $U(x)$ and $V(x)$ are both chosen from S_0^N,

$$V(x) = v_{c-1}\phi_{c-1}(x) + v_c\phi_c(x),\ x \in \Omega_e. \tag{2.29}$$

The basis functions

$$\phi_{c-1}(x) = \frac{x_c - x}{h} \quad \text{and} \quad \phi_c(x) = \frac{x - x_{c-1}}{h},\ x \in \Omega_e,$$

are shown in Figure 2.2

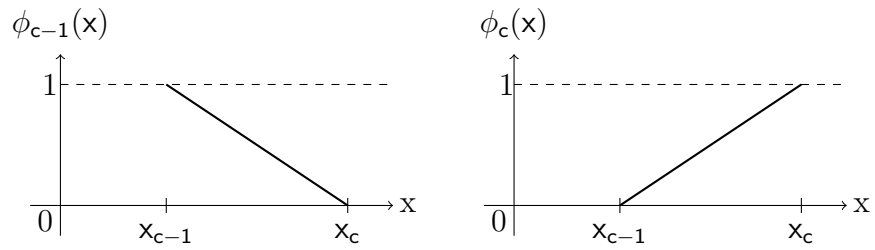


Figure 2.2: Element shape functions

Equation (2.27) can be written as a summation of the contributions of each element on the problem domain $[0, 1]$ as

$$\sum_{c=1}^{E-1} \int_{x_{c-1}}^{x_c} [-V'(x)U'(x) + V(x)a(x)U(x) - V(x)b(x)]dx = 0, \forall V(x) \in S_0^N. \tag{2.30}$$

Element stiffness matrix

The entries of element stiffness matrix are obtained by integrating the first term in (2.30) to get

$$\begin{aligned} \int_{x_{c-1}}^{x_c} V'(x)U'(x)dx &= \int_{x_{c-1}}^{x_c} \begin{bmatrix} v_{c-1} & v_c \end{bmatrix} \begin{bmatrix} \phi'_{c-1} \\ \phi'_c \end{bmatrix} \begin{bmatrix} \phi'_{c-1} & \phi'_c \end{bmatrix} \begin{bmatrix} u_{c-1} \\ u_c \end{bmatrix} dx \qquad (2.31) \\ &= \begin{bmatrix} v_{c-1} & v_c \end{bmatrix} K_c \begin{bmatrix} u_{c-1} \\ u_c \end{bmatrix}, \qquad (2.32) \end{aligned}$$

where

$$K_c = \begin{pmatrix} K_c^{c-1,c-1} & K_c^{c-1,c} \\ K_c^{c,c-1} & K_c^{c,c} \end{pmatrix}, \tag{2.33}$$

is the condensed form of K. The elements of the matrix K_c are

$$K_c^{m,n} = \int_{x_{c-1}}^{x_c} \phi'_m(x)\phi'_n(x)dx, \tag{2.34}$$

where m and n are either $c-1$ or c. For example

$$\begin{aligned} K_c^{c,c-1} &= \int_{x_{c-1}}^{x_c} \phi'_c(x)\phi'_{c-1}(x)dx \\ &= -\int_{x_{c-1}}^{x_c} \left(\frac{1}{h}\right)\left(\frac{1}{h}\right)dx \\ &= -1 \\ &= K_c^{c-1,c}. \end{aligned}$$

Hence

$$K_c = \frac{1}{h} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}. \tag{2.35}$$

Element mass matrix

We proceed in a similar manner for the second term $V(x)a(x)U(x)$ in equation (2.30). We begin by approximating the function $a(x)$ by its linear interpolant so that

$$a(x) \approx a_{c-1}\phi_{c-1}(x) + a_c\phi_c(x), \quad x \in \Omega_e \tag{2.36}$$

where $a_{c-1} = a(x_{c-1})$ and we get

$$\int_{x_{c-1}}^{x_c} V(x)a(x)U(x)dx = \begin{bmatrix} v_{c-1} & v_c \end{bmatrix} M_c \begin{bmatrix} u_{c-1} \\ u_c \end{bmatrix}, \tag{2.37}$$

where

$$M_c = \frac{h}{12}\begin{pmatrix} 3a_{c-1} + a_c & a_{c-1} + a_c \\ a_{c-1} + a_c & a_{c-1} + 3a_c \end{pmatrix}, \tag{2.38}$$

is called the condensed element mass matrix. If the function $a(x) = a$ is constant, then M_c reduces to

$$M_c = \frac{ah_c}{6}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}. \tag{2.39}$$

Element load vector

Similarly if we let

$$b(x) \approx b_{c-1}\phi_{c-1}(x) + b_c\phi_c(x), x \in \Omega_e \tag{2.40}$$

then

$$\begin{aligned} \int_{x_{c-1}}^{x_c} V(x)b(x)dx &\approx \int_{x_{c-1}}^{x_c} \begin{bmatrix} v_{c-1} & v_c \end{bmatrix} \begin{bmatrix} \phi_{c-1}(x) \\ \phi_c(x) \end{bmatrix} \begin{bmatrix} \phi_{c-1}(x) & \phi_c(x) \end{bmatrix} \begin{bmatrix} b_{c-1} \\ b_c \end{bmatrix} dx \\ &= \begin{bmatrix} v_{c-1} & v_c \end{bmatrix} \mathbf{w}_c \end{aligned}$$

where

$$\mathbf{w}_c = \frac{h}{6}\begin{pmatrix} 2b_{c-1} + b_c \\ b_{c-1} + 2b_c \end{pmatrix} \tag{2.41}$$

which is called the condensed element load vector.

Assembling global matrices

Assembling is a process of constructing the global matrices as the summation of the element matrices. In forming the global matrices it is not necessary to expand the element matrices in their condensed form but to only work out their appropriate positions in the global matrix and then add the numbers. In this summation we will consider a uniform width $h = 1/E$ for the elements so that

$$\begin{aligned}
\sum_{c=1}^{E}\int_{x_{c-1}}^{x_c} V'(x)U'(x)dx &= \sum_{c=1}^{E}\begin{bmatrix} v_{c-1} & v_c \end{bmatrix}\frac{1}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} u_{c-1} \\ u_c \end{bmatrix} \\
&= \begin{bmatrix} v_1 \end{bmatrix}\frac{1}{h}\begin{bmatrix} 1 \end{bmatrix}\begin{bmatrix} c_1 \end{bmatrix} + \begin{bmatrix} v_1 & v_2 \end{bmatrix}\frac{1}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \cdots \\
&+ \begin{bmatrix} v_{E-2} & v_{E-1} \end{bmatrix}\frac{1}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} u_{E-2} \\ u_{E-1} \end{bmatrix} + \begin{bmatrix} v_{E-1} \end{bmatrix}\frac{1}{h}\begin{bmatrix} 1 \end{bmatrix}\begin{bmatrix} u_{E-1} \end{bmatrix}.
\end{aligned}$$

The first and last terms have this form due to application of the boundary conditions $u_0 = u_E = v_0 = v_E = 0$. Expanding each vector and matrix to the same dimension we get

$$
\begin{aligned}
\sum_{c=1}^{E}\int_{x_{c-1}}^{x_c} V'(x)U'(x)dx &= \begin{bmatrix} v_1 & v_2 & \cdots & v_{E-1}\end{bmatrix}\frac{1}{h}\begin{bmatrix} 1 & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \\ \vdots \\ u_{E-1}\end{bmatrix} \\
&+ \begin{bmatrix} v_1 & v_2 & \cdots & v_{E-1}\end{bmatrix}\frac{1}{h}\begin{bmatrix} 1 & -1 & & \\ -1 & 1 & & \\ & & & \\ & & & \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \\ \vdots \\ u_{E-1}\end{bmatrix} \\
&+ \cdots \\
&+ \begin{bmatrix} v_1 & v_2 & \cdots & v_{E-1}\end{bmatrix}\frac{1}{h}\begin{bmatrix} & & & \\ & & & \\ & & 1 & -1 \\ & & -1 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \\ \vdots \\ u_{E-1}\end{bmatrix} \\
&+ \begin{bmatrix} v_1 & v_2 & \cdots & v_{E-1}\end{bmatrix}\frac{1}{h}\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \\ \vdots \\ u_{E-1}\end{bmatrix}.
\end{aligned}
$$

This can be written as

$$\sum_{c=1}^{E}\int_{x_{c-1}}^{x_c} V'(x)U'(x)dx = \mathbf{v}^T K\mathbf{u}, \tag{2.42}$$

where

$$K = \frac{1}{h}\begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}$$

is the global stiffness matrix and the vectors of unknown coefficients have the forms

$$\mathbf{v} = \begin{bmatrix} v_1 & v_2 & \cdots & v_{E-2} & v_{E-1} \end{bmatrix}^T \text{ and } \mathbf{u} = \begin{bmatrix} u_1 & u_2 & \cdots & u_{E-2} & u_{E-1} \end{bmatrix}^T.$$

Similarly

$$\sum_{c=1}^{E} \int_{x_{c-1}}^{x_c} V(x)a(x)U(x)dx = \mathbf{v}^T M \mathbf{u} \tag{2.43}$$

and

$$\sum_{c=1}^{E} \int_{x_{c-1}}^{x_c} V(x)b(x)dx = \mathbf{v^T w} \tag{2.44}$$

where the global mass matrix

$$M = \frac{h}{12}\begin{pmatrix} a_0 + 6a_1 + a_2 & a_1 + a_2 & & & & \\ a_1 + a_2 & a_1 + 6a_2 + a_3 & a_2 + a_3 & & & \\ & a_2 + a_3 & a_2 + 6a_3 + a_4 & a_3 + a_4 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{E-2} + a_{E-2} & a_{E-3} + 6a_{E-2} + a_{E-1} & a_{E-2} + a_{E-1} \\ & & & & a_{E-2} + a_{E-1} & a_{E-2} + 6a_{E-1} + a_E \end{pmatrix}$$

is the global stiffness matrix which reduces to

$$M = \frac{ah}{12}\begin{bmatrix} 4 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 4 \end{bmatrix} \tag{2.45}$$

when $a(x) \equiv a$ is constant and

$$\mathbf{w} = \frac{h}{6}\begin{bmatrix} b_0 + 4b_1 + b_2 \\ b_1 + 4b_2 + b_3 \\ \vdots \\ b_{E-2} + 4b_{E-1} + b_E \end{bmatrix} \tag{2.46}$$

is the global load vector. Substituting (2.42) ,(2.43) and (2.44) into (2.30) we get

$$\mathbf{v^T}[(M-K)\mathbf{u} - \mathbf{w}] =, \text{for all } \ V(x) \in S_0^N$$

or

$$(M-K)\mathbf{u} = \mathbf{w} \tag{2.47}$$

for all choices of choices of v . Solving this linear system we get the nodal values u_c, $c = 1, 2, \cdots, E-1$ for the approximation (2.20). Looking closely at (2.42) , without applying boundary conditions, this summation can be written as

$$\sum_{c=1}^{E} \int_{x_{c-1}}^{x_c} V'(x)U'(x)dx = \begin{bmatrix} v_0 & v_1 & \cdots & v_{E-1} & v_E \end{bmatrix} K \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{E-1} \\ u_E \end{bmatrix} \tag{2.48}$$

with the $(E+1)\times(E+1)$ matrix K being the summations of matrices given by

$$K=\frac{1}{h}\left(\begin{bmatrix}1&-1&&&\\-1&1&&&\\&&&&\\&&&&\\&&&&\end{bmatrix}+\begin{bmatrix}&&&&\\&1&-1&&\\&-1&1&&\\&&&&\\&&&&\end{bmatrix}+\cdots+\begin{bmatrix}&&&&\\&&&&\\&&1&-1&\\&&-1&1&\\&&&&\end{bmatrix}+\begin{bmatrix}&&&&\\&&&&\\&&&&\\&&1&-1&\\&&-1&1&\end{bmatrix}\right)$$

which we can write in compact form as

$$K=\sum_{e=1}^{E}K^e \tag{2.49}$$

where

$$K^e=\frac{1}{h}\begin{array}{c}\begin{matrix}1&\dots&e-1&e&\dots&E\end{matrix}\\\begin{pmatrix}0&\dots&0&0&\dots&0\\\vdots&\ddots&\vdots&\vdots&\ddots&\vdots\\0&\dots&1&-1&\dots&0\\0&\dots&-1&1&\dots&0\\\vdots&\ddots&\vdots&\vdots&\ddots&\vdots\\0&\dots&0&0&\dots&0\end{pmatrix}\end{array}\begin{array}{c}1\\\vdots\\e-1\\e\\\vdots\\E\end{array} \tag{2.50}$$

is a sparse matrix and its condensed form is

$$K_c^e=\frac{1}{h}\begin{array}{c}\begin{matrix}e-1&e\end{matrix}\\\begin{bmatrix}1&-1\\-1&1\end{bmatrix}\end{array}\begin{array}{c}e-1\\e\end{array} \tag{2.51}$$

The numbers $e-1$ and e on the boarders of the matrix indicates the appropriate positions of the elements in the global matrix. Similarly we get

$$M_c^e=\frac{h}{12}\begin{array}{c}\begin{matrix}e-1&e\end{matrix}\\\begin{bmatrix}3a_{e-1}+a_e&a_{e-1}+a_e\\a_{e-1}+a_e&a_{e-1}+3a_e\end{bmatrix}\end{array}\begin{array}{c}e-1\\e\end{array} \tag{2.52}$$

and

$$\mathbf{w_c^e} = \frac{h}{6}\begin{bmatrix} 2b_{e-1} + b_e \\ b_{e-1} + 2b_e \end{bmatrix}\begin{matrix} e-1 \\ e \end{matrix} \tag{2.53}$$

as the condensed element mass matrix and element load vector respectively. We demonstrate the assembly procedure by example.

Example 3. We now describe the assembly procedure for the element stiffness matrix on the problem domain $[0, 1]$ with three elements, that is, $E = 3$ and $h = 1/E = 1/3$ to form a global stiffness matrix K of dimension $E + 1 = 4$. Starting with an empty 4×4 global stiffness matrix, the contribution of K^1 to the global matrix K is

$$K = 3\begin{matrix} \\ 0 \\ 1 \\ \\ \end{matrix}\begin{bmatrix} \begin{matrix} 0 & 1 \end{matrix} & \\ \begin{matrix} 1 & -1 \end{matrix} & \begin{matrix} \\ 0 \end{matrix} \\ \begin{matrix} -1 & 1 \end{matrix} & \begin{matrix} \\ 1 \end{matrix} \\ & \\ & \end{bmatrix}$$

and upon adding K^2 we get

$$K = 3\begin{matrix} \\ 1 \\ 2 \\ \\ \end{matrix}\begin{bmatrix} \begin{matrix} 1 & 2 \end{matrix} & \\ \begin{matrix} 1 & -1 \end{matrix} & \\ \begin{matrix} -1 & 1+1 & -1 \end{matrix} & \begin{matrix} \\ 1 \end{matrix} \\ \begin{matrix} -1 & 1 \end{matrix} & \begin{matrix} \\ 2 \end{matrix} \\ & \end{bmatrix}$$

and finally adding K^3 changes K to

$$K = 3\begin{matrix} \\ \\ 2 \\ 3 \end{matrix}\begin{bmatrix} \begin{matrix} 1 & 2 & 3 \end{matrix} & \\ \begin{matrix} 1 & -1 \end{matrix} & \\ \begin{matrix} -1 & 1+1 & -1 \end{matrix} & \\ \begin{matrix} -1 & 1+1 & 1 \end{matrix} & \begin{matrix} 2 \end{matrix} \\ \begin{matrix} -1 & 1 \end{matrix} & \begin{matrix} 3 \end{matrix} \end{bmatrix}$$

After adding K^1, K^2 and K^3 to K and padding with zeros we get

$$K = 3 \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \end{array}$$

The assembly procedure for the global mass matrix follows in a similar manner to yield

$$M = \frac{1}{36} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 3a_0 + a_1 & a_0 + a_1 & 0 & 0 \\ a_0 + a_1 & a_0 + 6a_1 + a_2 & a_1 + a_2 & 0 \\ 0 & a_1 + a_2 & a_1 + 6a_2 + a_3 & a_2 + a_3 \\ 0 & 0 & a_2 + a_3 & a_2 + 3a_3 \end{bmatrix} \end{array} \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array}$$

Similarly the global load vector

$$\mathbf{w} = \frac{1}{18} \begin{bmatrix} 2b_0 + b_1 \\ b_0 + 4b_1 + b_2 \\ b_1 + 4b_2 + b_3 \\ b_2 + 2b_3 \end{bmatrix} \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} \tag{2.54}$$

Imposing boundary conditions

The algebraic system (2.47) can be written as

$$N\mathbf{u} = \mathbf{w} \tag{2.55}$$

where

$$N = M - K \tag{2.56}$$

$$\mathbf{N} = \begin{pmatrix} n_{00} & n_{01} & \cdots & n_{0,E-1} & n_{0E} \\ n_{10} & n_{11} & \cdots & n_{1,E-1} & n_{1E} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ n_{E-1,0} & n_{E-1,1} & \cdots & n_{E-1,E-1} & n_{E-1,E} \\ n_{E,0} & n_{E,1} & \cdots & n_{E,E-1} & n_{E,E} \end{pmatrix}, \mathbf{u} = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{E-1} \\ u_E \end{pmatrix} \text{ and } \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{E-1} \\ w_E \end{pmatrix}$$

hence can be solved easily using any suitable method but only after imposing the boundary conditions. To apply the boundary conditions, we delete the first and last rows of both N and $\mathbf{w}$ and the first and last columns of N. This is because the first and last finite element equations correspond to the nodal values u_0 and u_E respectively which are known from the boundary hence are not needed. The system becomes

$$\begin{pmatrix} n_{10} & n_{11} & \cdots & n_{1,E-1} & n_{1E} \\ n_{20} & n_{21} & \cdots & n_{2,E-1} & n_{2E} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ n_{E-2,0} & n_{E-2,1} & \cdots & n_{E-2,E-1} & n_{E-2,E} \\ n_{E-1,0} & n_{E-1,1} & \cdots & n_{E-1,E-1} & n_{E-1,E} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{E-2} \\ u_{E-1} \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{E-2} \\ w_{E-1} \end{pmatrix}.$$

2.3.2 Piecewise quadratic Lagrange finite element solution

For piecewise quadratic Lagrange finite element solution of problem (2.12a) subject to boundary conditions (2.12b), the same general element $\Omega_e = [x_{c-1}, x_c]$, $c = 1, 2, \cdots, E-1, E$ used for piecewise linear Lagrange finite element solution is considered. The only difference on the element is that besides the two end nodes, a middle node is added to the element as shown in Figure 2.3

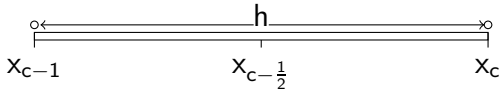


Figure 2.3: A general element for quadratic shape functions

On problem domain $[0, 1]$ we form a mesh

$$0 = x_0 < x_1 < x_2 < \cdots < x_{c-1} < x_{c-\frac{1}{2}} < x_c \cdots < x_{2E-2} < x_{2E-1} < x_{2E} = 1.$$

The length of each element $h = x_c - x_{c-1}$ for each $c = 1, 2, \cdots, E-1, E$ as shown in Figure 2.3. The finite element trial solution is given as

$$U(x) = \sum_{c=0}^{2E} u_{\frac{c}{2}} \phi_{\frac{c}{2}}(x) \tag{2.57}$$

where $\phi_{\frac{c}{2}}(x)$ a piecewise quadratic global basis function which satisfies

$$\phi_c(x) = \begin{cases} 1, & \text{at node c} \\ 0, & \text{otherwise} \end{cases} \tag{2.58}$$

The restriction of the trial function to the element Ω_e is

$$U^e(x) = u_{c-1}\phi_{c-1}(x) + u_{c-\frac{1}{2}}\phi_{c-\frac{1}{2}}(x) + u_c\phi_c(x), \ x \in \Omega_e. \tag{2.59}$$

The restriction of $\phi_c(x)$ to Ω_e is the element shape function $N_c(x)$

$$\phi_c(x) \equiv N_c^e(x), \ \forall x \in \Omega_e. \tag{2.60}$$

where

$$N_c^e(x) = \begin{cases} 1, \text{ at node c} \\ 0, \text{ otherwise} \end{cases} \tag{2.61}$$

Figure 2.4 shows the three element shape functions associated with the nodes x_{c-1}, $x_{c-\frac{1}{2}}$ and x_c on Ω_e

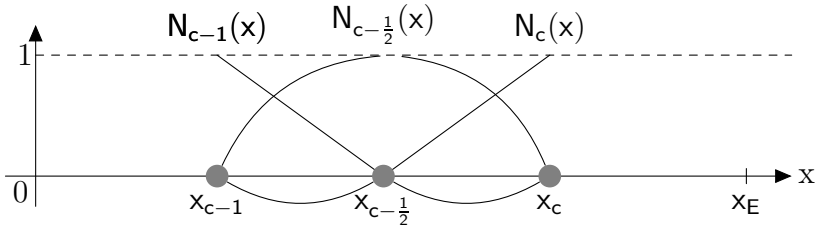


Figure 2.4: Quadratic element shape functions

Using the illustration in Figure (2.4) and with property (2.61) in mind, it follows that $N^e_{c-1}(x_{c-1}) = 1$, $N^e_{c-1}(x_{c-\frac{1}{2}}) = 0$ and $N^e_{c-1}(x_c) = 0$. Using the factor theorem, $(x_{c-\frac{1}{2}} - x)$ and $(x_c - x)$ are factors of the quadratic element shape function $N^e_{c-1}(x)$ such that coupled with normalisation

$$N^e_{c-1}(x) = \frac{2}{h^2}(x_{c-\frac{1}{2}} - x)(x_c - x),\ x \in \Omega_e \tag{2.62}$$

Similarly, the other two element shape functions are

$$N^e_{c-\frac{1}{2}}(x) = \frac{4}{h^2}(x - x_{c-1})(x - x_c) \text{ and } N^e_c(x) = \frac{2}{h^2}(x - x_{c-1})(x - x_{c-\frac{1}{2}}). \tag{2.63}$$

Constructing element matrices

In the construction of element matrices, to conveniently evaluate the involved integrals, the physical element $\Omega_e = [x_{c-1}, x_c]$ on the x-axis is mapped onto the canonical element $\Pi_e = [-1, 1]$ on the ξ-axis using the transformation

$$x(\xi) = \frac{1 - \xi}{2}x_{c-1} + \frac{1 + \xi}{2}x_c, -1 \leq \xi \leq 1 \tag{2.64}$$

See Figure 2.5.

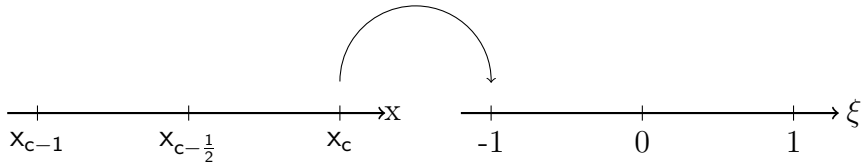


Figure 2.5: Linear transformation

Using the linear transformation (2.64), element shape functions (2.62) and (2.63) on the x-axis are transformed to the equivalent quadratic shape functions

$$N_{-1}(\xi) = \frac{\xi(\xi - 1)}{2},\ N_0(\xi) = 1 - \xi^2 \text{ and } N_1(\xi) = \frac{\xi(\xi + 1)}{2} \tag{2.65}$$

associated with nodes -1, 0 and 1 on the ξ-axis respectively. This far, the finite element trial solution on Ω_e can be written

$$U^e(x) = u_{-1}N^e_{-1}(\xi) + u_0 N^e_0(\xi) + u_1 N^e_1(\xi),\ \xi \in \Pi_e \tag{2.66}$$

with the test function $V(x)$ taking a similar form.

Element stiffness matrix

Like in the linear case,

$$\int_{x_{c-1}}^{x_c} V'(x)U'(x)dx \ = \ \begin{bmatrix} v_{c-1} & v_{c-\frac{1}{2}} & v_c \end{bmatrix} K_c \begin{bmatrix} u_{c-1} \\ u_{c-\frac{1}{2}} \\ u_c \end{bmatrix},$$

where K_c is a symmetric 3×3 matrix given by

$$K_c = \begin{pmatrix} K_c^{c-1,c-1} & K_c^{c-1,c-\frac{1}{2}} & K_c^{c-1,c} \\ K_c^{c-\frac{1}{2},c-1} & K_c^{c-\frac{1}{2},c-\frac{1}{2}} & K_c^{c-\frac{1}{2},c} \\ K_c^{c,c-1} & K_c^{c,c-\frac{1}{2}} & K_c^{c,c} \end{pmatrix}. \tag{2.67}$$

The entries of the matrix are given by

$$K_c^{m,n} = \int_{x_{c-1}}^{x_c} N'_m(x)N'_n(x)dx, \tag{2.68}$$

where m and n are either $c-1$, $c-\frac{1}{2}$ or c. For example

$$\begin{aligned} K_c^{c-1,c-1} &= \int_{x_{c-1}}^{x_c} \frac{dN_{c-1}(x)}{dx}\frac{dN_{c-1}(x)}{dx}dx \\ &= \int_{-1}^{1} \left(\frac{2}{h_c}\right)^2 \frac{dN_{-1}(\xi)}{d\xi}\frac{dN_{-1}(\xi)}{d\xi}\frac{h}{2}d\xi \\ &= \frac{7}{3h}. \end{aligned}$$

The determination of the rest of the entries follows a similar way to get the condensed element

stiffness matrix as

$$K_c = \frac{1}{3h}\begin{pmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{pmatrix} \tag{2.69}$$

Element mass matrix

Similar to the linear case,

$$\int_{x_{c-1}}^{x_c} V(x)a(x)U(x)dx = \begin{bmatrix} v_{c-1} & v_{c-\frac{1}{2}} & v_c \end{bmatrix} M_c \begin{bmatrix} u_{c-1} \\ u_{c-\frac{1}{2}} \\ u_c \end{bmatrix}$$

where the 3×3 symmetric matrix.

$$M_c^e = \begin{pmatrix} M_c^{c-1,c-1} & M_c^{c-1,c-\frac{1}{2}} & M_c^{c-1,c} \\ M_c^{c-\frac{1}{2},c-1} & M_c^{c-\frac{1}{2},c-\frac{1}{2}} & M_c^{c-\frac{1}{2},c} \\ M_c^{c,c-1} & M_c^{c,c-\frac{1}{2}} & M_c^{c,c} \end{pmatrix} \tag{2.70}$$

is the condensed element mass matrix where

$$M_c^{m,n} = \int_{x_{c-1}}^{x_c} N_m(x)a(x)N_n(x)dx$$

and

$$a(x) \approx a_{c-1}N_{c-1}(x) + a_{c-\frac{1}{2}}N_{c-\frac{1}{2}} + a_cN_c(x),\ x \in \Omega_e$$

The entry in row $c-1$ and column $c-1$ of M_c is

$$\begin{aligned} M_c^{c-1,c-1} &= \int_{x_{c-1}}^{x_c} N_{c-1}(x)a(x)N_{c-1}(x)dx \\ &= \int_{x_{c-1}}^{x_c} N_{c-1}(x)\Big(a_{c-1}N_{c-1}(x) + a_{c-\frac{1}{2}}N_{c-\frac{1}{2}}(x) + a_cN_c(x)\Big)N_{c-1}(x)dx \\ &= \frac{13}{140}ha_{c-1} + \frac{1}{21}ha_{c-\frac{1}{2}} - \frac{1}{140}ha_c \end{aligned}$$

Other elements are computed in a similar manner to reveal that

$$M_c = h \begin{pmatrix} \frac{13a_i}{140} + \frac{a_j}{21} - \frac{a_k}{140} & \frac{a_i}{21} + \frac{4a_j}{105} - \frac{2a_k}{105} & -\frac{a_i}{140} - \frac{2a_j}{105} - \frac{a_k}{140} \\ \frac{a_i}{21} + \frac{4a_j}{105} - \frac{2a_k}{105} & \frac{4a_i}{105} + \frac{16a_j}{35} + \frac{4a_k}{105} & -\frac{2a_i}{105} + \frac{4a_j}{105} + \frac{a_k}{21} \\ -\frac{a_i}{140} - \frac{2a_j}{105} - \frac{a_k}{140} & -\frac{2a_i}{105} + \frac{4a_j}{105} + \frac{a_k}{21} & -\frac{a_i}{140} + \frac{a_j}{21} + \frac{13a_k}{140} \end{pmatrix} \tag{2.71}$$

where $a_i = a(x_{c-1})$, $a_j = a(x_{c-\frac{1}{2}})$ and $a_k = a(x_c)$. If $a(x)$ is a constant function, then (2.71) becomes

$$M_c = \frac{ah}{30} \begin{pmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{pmatrix} \tag{2.72}$$

Element load vector

Similarly, if

$$b(x) \approx b_{c-1}N_{c-1}(x) + b_{c-\frac{1}{2}}N_{c-\frac{1}{2}}(x) + b_cN_c(x),\ x \in \Omega_e, \tag{2.73}$$

then

$$\int_{x_{c-1}}^{x_c} V(x)b(x)dx \quad = \quad \begin{bmatrix} v_{c-1} & v_{c-\frac{1}{2}} & v_c \end{bmatrix} \mathbf{w}_c,$$

where

$$\mathbf{w}_c = \frac{h_c}{30} \begin{bmatrix} 4b_{c-1} + 2b_{c-\frac{1}{2}} - b_c \\ 2b_{c-1} + 16b_{c-\frac{1}{2}} + 2b_c \\ -b_{c-1} + 2b_{c-\frac{1}{2}} + 4b_c \end{bmatrix}, \tag{2.74}$$

is the element load vector. Just like in the linear case, finite element equations reduce to linear system

$$(M - K)\mathbf{u} = \mathbf{w}, \text{for all choices of choices of v .} \tag{2.75}$$

Assemble global matrices

Example 4. We illustrate the assembly procedure for global matrices using three elements that is $E = 3$, $h = 1/E = 1/3$ and we have seven nodes. The condensed element stiffness matrix

$$K_c^e = \frac{1}{3h} \begin{array}{c} \begin{array}{ccc} e-1 & e-\frac{1}{2} & e \end{array} \\ \begin{pmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{pmatrix} \end{array} \begin{array}{c} e-1 \\ e-\frac{1}{2} \\ e \end{array} \tag{2.76}$$

is added to K for each $e = 1, 2, 3$ in the appropriate rows and columns. Upon padding with zeros we get

$$K = \begin{array}{c} \begin{array}{ccccccc} 0 & \frac{1}{2} & 1 & \frac{3}{2} & 2 & \frac{5}{2} & 3 \end{array} \\ \begin{bmatrix} 7 & -8 & 1 & 0 & 0 & 0 & 0 \\ -8 & 16 & -8 & 0 & 0 & 0 & 0 \\ 1 & -8 & 7+7 & -8 & 1 & 0 & 0 \\ 0 & 0 & -8 & 16 & -8 & 0 & 0 \\ 0 & 0 & 1 & -8 & 7+7 & -8 & 1 \\ 0 & 0 & 0 & 0 & -8 & 16 & -8 \\ 0 & 0 & 0 & 0 & 1 & -8 & 7 \end{bmatrix} \end{array} \begin{array}{c} 0 \\ \frac{1}{2} \\ 1 \\ \frac{3}{2} \\ 2 \\ \frac{5}{2} \\ 3 \end{array} \tag{2.77}$$

which upon applying the boundary conditions becomes

$$K = \begin{pmatrix} 16 & -8 & 0 & 0 & 0 \\ -8 & 14 & -8 & 1 & 0 \\ 0 & -8 & 16 & -8 & 0 \\ 0 & 1 & -8 & 14 & -8 \\ 0 & 0 & 0 & -8 & 16 \end{pmatrix} \tag{2.78}$$

Similarly, with the condensed element mass matrix

$$M_c^e = \frac{1}{3}\begin{array}{c} \begin{matrix} e-1 & e-\frac{1}{2} & e \end{matrix} \\ \begin{pmatrix} \frac{13a_i}{140}+\frac{a_j}{21}-\frac{a_k}{140} & \frac{a_i}{21}+\frac{4a_j}{105}-\frac{2a_k}{105} & -\frac{a_i}{140}-\frac{2a_j}{105}-\frac{a_k}{140} \\ \frac{a_i}{21}+\frac{4a_j}{105}-\frac{2a_k}{105} & \frac{4a_i}{105}+\frac{16a_j}{35}+\frac{4a_k}{105} & -\frac{2a_i}{105}+\frac{4a_j}{105}+\frac{a_k}{21} \\ -\frac{a_i}{140}-\frac{2a_j}{105}-\frac{a_k}{140} & -\frac{2a_i}{105}+\frac{4a_j}{105}+\frac{a_k}{21} & -\frac{a_i}{140}+\frac{a_j}{21}+\frac{13a_k}{140} \end{pmatrix} \end{array} \begin{matrix} e-1 \\ e-\frac{1}{2} \\ e \end{matrix} \tag{2.79}$$

where $i = e-1$,$j = e-\frac{1}{2}$ and $k = e$, adding M for each $e = 1, 2, 3$ in the appropriate rows and columns which upon applying the boundary conditions becomes

$$M = \begin{pmatrix} p & q & 0 & 0 & 0 \\ q & r & s & t & 0 \\ 0 & s & u & v & 0 \\ 0 & t & v & w & x \\ 0 & 0 & 0 & x & z \end{pmatrix}$$

where

$$p = \frac{4}{105}a_0 + \frac{16}{35}a_{\frac{1}{2}} + \frac{4}{105}a_1,\ q = -\frac{2}{105}a_0 + \frac{4}{105}a_{\frac{1}{2}} + \frac{1}{21}$$
$$r = -\frac{1}{140}a_0 + \frac{1}{21}a_{\frac{1}{2}} + \frac{13}{140}a_1 + \frac{13}{140}a_1 + \frac{1}{21}a_{\frac{3}{2}} - \frac{1}{140}a_2$$
$$s = \frac{1}{21}a_1 + \frac{4}{105}a_{\frac{3}{2}} - \frac{2}{105}a_2,\ t = -\frac{1}{140}a_1 - \frac{2}{105}a_{\frac{3}{2}} - \frac{1}{140}a_2$$
$$u = \frac{4}{105}a_1 + \frac{16}{35}a_{\frac{3}{2}} + \frac{4}{105}a_2,\ v = -\frac{2}{105}a_1 + \frac{4}{105}a_{\frac{3}{2}} + \frac{1}{21}a_1$$
$$w = -\frac{1}{140} + \frac{1}{21}a_{\frac{3}{2}} + \frac{13}{140}a_2 + \frac{13}{140}a_2 + \frac{1}{21}a_{\frac{3}{2}} - \frac{1}{140}a_3,$$
$$x = \frac{1}{21}a_2 + \frac{4}{105}a_{\frac{5}{2}} - \frac{2}{105}a_3 \text{ and } z = \frac{4}{105}a_2 + \frac{16}{35}a_{\frac{5}{2}} + \frac{4}{105}a_3$$

and condensed element load vector

$$\mathbf{w_c^e} = \frac{h}{30}\begin{array}{c} \begin{pmatrix} 4b_{e-1} + 2b_{e-\frac{1}{2}} - b_e \\ 2b_{e-1} + 16b_{e-\frac{1}{2}} + 2b_e \\ -b_{e-1} + 2b_{e-\frac{1}{2}} + 4b_e \end{pmatrix} \end{array} \begin{matrix} e-1 \\ e-\frac{1}{2} \\ e \end{matrix} \tag{2.80}$$

yields the global load vector

$$\mathbf{w} = 10 \begin{pmatrix} 4b_0 + 2b_{\frac{1}{2}} - b_1 & 0 \\ 2b_0 + 16b_{\frac{1}{2}} + 2b_1 & \frac{1}{2} \\ -b_0 + 2b_{\frac{1}{2}} + 4b_1 + 4b_1 + 2b_{\frac{3}{2}} - b_2 & 1 \\ 2b_1 + 16b_{\frac{3}{2}} + 2b_2 & \frac{3}{2} \\ -b_1 + 2b_{\frac{3}{2}} + 4b_2 + 4b_2 + 2b_{\frac{5}{2}} - b_3 & 2 \\ 2b_2 + 16b_{\frac{5}{2}} + 2b_3 & \frac{5}{2} \\ -b_2 + 2b_{\frac{5}{2}} + 4b_3 & 3 \end{pmatrix} \tag{2.81}$$

which upon appying boundary conditions becomes

$$\mathbf{w} = 10 \begin{pmatrix} 2b_0 + 16b_{\frac{1}{2}} + 2b_1 \\ -b_0 + 2b_{\frac{1}{2}} + 4b_1 + 4b_1 + 2b_{\frac{3}{2}} - b_2 \\ 2b_1 + 16b_{\frac{3}{2}} + 2b_2 \\ -b_1 + 2b_{\frac{3}{2}} + 4b_2 + 4b_2 + 2b_{\frac{5}{2}} - b_3 \\ 2b_2 + 16b_{\frac{5}{2}} + 2b_3 \end{pmatrix} \tag{2.82}$$

2.3.3 Finite element solution using hierarchical basis functions

The finite element approximation of the solution of the boundary value problem (2.12a) and (2.12b) using the quadratic hierarchical basis functions is discussed in this Section. Used in the finite element method, hierarchical basis functions have an advantage over the Lagrange polynomials in that the resulting algebraic system is less susceptible to round-off error accumulation at high order. The same general element $\Omega_e = [x_{c-1}, x_c]$ in Figure 2.3 is considered. Unlike in the quadratic Lagrange finite element approximation, the basis functions associated with both end nodes are linear while that associated with the middle node is quadratic as shown in Figure 2.6

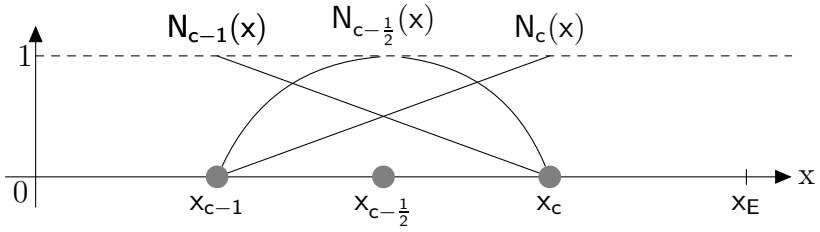


Figure 2.6: Hierarchical shape functions

Similar to the quadratic Lagrange approximation, the physical element Ω_e is transformed to the canonical element $\Pi_e = [-1, 1]$ using transformation (2.64). The restriction of the solution to the canonical element Π_e is given by

$$U(\xi) = u_{c-1}N_{-1}(\xi) + u_{c-\frac{1}{2}}N_0(\xi) + u_cN_1(\xi), \quad \xi \in [-1, 1] \tag{2.83}$$

where $N_{-1}(\xi)$ and $N_1(\xi)$ are linear element shape functions derived from the hat functions in Figure 2.2 using transformation (2.64). Hence

$$N_{-1}(\xi) = \frac{1}{2}(1 - \xi) \text{ and } N_1(\xi) = \frac{1}{2}(1 + \xi), \ \xi \in \Pi_e.$$

Following Szabǿ and Babuska [40] we take

$$N_0(\xi) = \sqrt{\frac{3}{2}} \int_{-1}^{\xi} P_1(\rho)d(\rho),$$

where $P_1(\rho) = \rho$ is the second Legendre polynomial

$$\Rightarrow N_0(\xi) = \frac{3}{2\sqrt{6}}(\xi^2 - 1) \tag{2.84}$$

Constructing element matrices

Element stiffness matrix

Since

$$\int_{x_{c-1}}^{x_c} V'(x)U'(x)dx = \frac{2}{h_c}\int_{-1}^{1} \frac{dV}{d\xi}\frac{dU}{d\xi}\mathrm{d}\xi \tag{2.85}$$

$$= \begin{bmatrix} v_{c-1} & v_{c-\frac{1}{2}} & v_c \end{bmatrix} K_c \begin{bmatrix} u_{c-1} \\ u_{c-\frac{1}{2}} \\ u_c \end{bmatrix} \tag{2.86}$$

where the restrictions of the piecewise quadratic trial and test functions to the element Ω_e are

$$U(\xi) = \begin{bmatrix} u_{c-1} & u_{c-\frac{1}{2}} & u_c \end{bmatrix} \begin{bmatrix} N_{-1} \\ N_0 \\ N_1 \end{bmatrix} \text{ and } V(\xi) = \begin{bmatrix} v_{c-1} & v_{c-\frac{1}{2}} & v_c \end{bmatrix} \begin{bmatrix} N_{-1} \\ N_0 \\ N_1 \end{bmatrix}$$

respectively then the condensed element stiffness matrix K_c given by

$$K_c = \frac{2}{h_c}\int_{-1}^{1} \begin{bmatrix} N'_{-1} \\ N'_0 \\ N'_1 \end{bmatrix} \begin{bmatrix} N'_{-1} & N'_0 & N'_1 \end{bmatrix} \mathrm{d}\xi \tag{2.87}$$

which upon simplification becomes

$$K_c = \frac{1}{h_c}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Element mass matrix

In a similar way,

$$\int_{x_{c-1}}^{x_c} V(x)a(x)U(x)dx = \begin{bmatrix} v_{c-1} & v_{c-\frac{1}{2}} & v_c \end{bmatrix} M_c \begin{bmatrix} u_{c-1} \\ u_{c-\frac{1}{2}} \\ u_c \end{bmatrix} \tag{2.88}$$

Since

$$a(\xi) \approx a_{c-1}N_{-1}(\xi) + a_c N_1(\xi),\ \xi \in [-1, 1]$$

then the element mass matrix is

$$
\begin{aligned}
M_c &= \frac{h_c}{2}\int_{-1}^{1}\begin{bmatrix}N_{-1}\\ N_0\\ N_1\end{bmatrix} a(\xi)\begin{bmatrix}N_{-1} & N_0 & N_1\end{bmatrix} d\xi \\
&= h_c\begin{pmatrix}
\frac{1}{4}a_{c-1}+\frac{1}{12}a_c & -\frac{\sqrt{6}}{20}a_{c-1}-\frac{\sqrt{6}}{30}a_c & \frac{1}{12}a_{c-1}+\frac{1}{12}a_c \\
-\frac{\sqrt{6}}{20}a_{c-1}-\frac{\sqrt{6}}{30}a_c & \frac{1}{10}a_{c-1}+\frac{1}{10}a_c & -\frac{\sqrt{6}}{30}a_{c-1}-\frac{\sqrt{6}}{20}a_c \\
\frac{1}{12}a_{c-1}+\frac{1}{12}a_c & -\frac{\sqrt{6}}{30}a_{c-1}-\frac{\sqrt{6}}{20}a_c & \frac{1}{12}a_{c-1}+\frac{1}{4}a_c
\end{pmatrix}.
\end{aligned}
$$

which simplifies to

$$
M_c = \frac{ah}{6}\begin{pmatrix}
2 & 1 & -\sqrt{\frac{3}{2}} \\
1 & 2 & -\sqrt{\frac{3}{2}} \\
-\sqrt{\frac{3}{2}} & -\sqrt{\frac{3}{2}} & \frac{6}{5}
\end{pmatrix} \tag{2.89}
$$

when $a(x)$ is constant.

Element load vector

As done for the linear piecewise approximation,

$$
\begin{aligned}
\int_{x_{c-1}}^{x_c} V(x)b(x)dx &= \frac{h}{2}\int_{-1}^{1} V(\xi)b(\xi)d\xi &(2.90)\\
&= \begin{bmatrix}v_{c-1} & v_{c-\frac{1}{2}} & v_c\end{bmatrix}\mathbf{w}_c &(2.91)
\end{aligned}
$$

where

$$
\mathbf{w}_c = \int_{-1}^{1}\begin{bmatrix}N_{-1}\\ N_0\\ N_1\end{bmatrix} b(x(\xi))\dot{d}\xi \tag{2.92}
$$

and

$$
b(x) \approx b_{c-1}N_{-1}(\xi) + b_cN_1(\xi),\ \xi\in[-1,1] \tag{2.93}
$$

Simplifying (2.92) gives the element load vector as

$$\mathbf{w}_c = \frac{h_c}{6}\begin{bmatrix} 2b_{c-1} + b_c \\ -\sqrt{\frac{3}{2}}(b_{c-1} + b_c) \\ b_{c-1} + 2b_c \end{bmatrix} \tag{2.94}$$

Substituting (2.85),(2.88) and (2.90) into (2.30),we get the linear algebraic system

$$(M - K)\mathbf{u} = \mathbf{w}, \text{ for all choices of } v_c, \ c = 1/2, 1, 3/2, \cdots E - 1 \tag{2.95}$$

Assembly procedure

Example 5. The assembly procedure for K, M and $\mathbf{w}$ for the quadratic hierarchical approximation follows the same pattern as for the Lagrange approximations. The condensed forms of K, M and $\mathbf{w}$ are

$$K_c^e = \frac{1}{h}\begin{array}{c} \begin{matrix} e-1 & e-\frac{1}{2} & e \end{matrix} \\ \begin{bmatrix} 1 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 1 \end{bmatrix} \end{array} \begin{array}{c} e-1 \\ e-\frac{1}{2} \\ e \end{array} \tag{2.96}$$

$$M_c^e = \frac{1}{3}\begin{array}{c} \begin{matrix} e-1 & e-\frac{1}{2} & e \end{matrix} \\ \begin{bmatrix} \frac{1}{4}a_{e-1} + \frac{1}{12}a_e & -\frac{\sqrt{6}}{20}a_{e-1} - \frac{\sqrt{6}}{30}a_e & \frac{1}{12}a_{e-1} + \frac{1}{12}a_e \\ -\frac{\sqrt{6}}{20}a_{e-1} - \frac{\sqrt{6}}{30}a_e & \frac{1}{10}a_{e-1} + \frac{1}{10}a_e & -\frac{\sqrt{6}}{30}a_{e-1} - \frac{\sqrt{6}}{20}a_e \\ \frac{1}{12}a_{e-1} + \frac{1}{12}a_e & -\frac{\sqrt{6}}{30}a_{e-1} - \frac{\sqrt{6}}{20}a_e & \frac{1}{12}a_{e-1} + \frac{1}{4}a_e \end{bmatrix} \end{array} \begin{array}{c} e-1 \\ e-\frac{1}{2} \\ e \end{array} \tag{2.97}$$

$$\mathbf{w_c^e} = \frac{1}{18}\begin{bmatrix} 2b_{e-1} + b_e \\ -\sqrt{\frac{3}{2}}(b_{e-1} + b_e) \\ b_{e-1} + 2b_e \end{bmatrix} \begin{array}{c} e-1 \\ e-\frac{1}{2} \\ e \end{array} \tag{2.98}$$

respectively. With three nodes on the interval $[0, 1]$, we get the global stiffness, mass and load vector matrices as

$$K = \begin{array}{c} \begin{matrix} 0 & \frac{1}{2} & 1 & \frac{3}{2} & 2 & \frac{5}{2} & 3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1+1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1+1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} 0 \\ \frac{1}{2} \\ 1 \\ \frac{3}{2} \\ 2 \\ \frac{5}{2} \\ 3 \end{matrix}, \tag{2.99}$$

$$M = \frac{1}{3} \begin{array}{c} \begin{matrix} 0 & \frac{1}{2} & 1 & \frac{3}{2} & 2 & \frac{5}{2} & 3 \end{matrix} \\ \begin{bmatrix} a & b & c & 0 & 0 & 0 & 0 \\ b & d & e & 0 & 0 & 0 & 0 \\ c & e & f & g & h & 0 & 0 \\ 0 & 0 & g & i & j & 0 & 0 \\ 0 & 0 & h & j & k & l & m \\ 0 & 0 & 0 & 0 & l & o & p \\ 0 & 0 & 0 & 0 & m & p & s \end{bmatrix} \end{array} \begin{matrix} 0 \\ \frac{1}{2} \\ 1 \\ \frac{3}{2} \\ 2 \\ \frac{5}{2} \\ 3 \end{matrix},$$

and

$$\mathbf{w} = \frac{1}{18} \begin{bmatrix} 2b_0 + b_1 \\ -\sqrt{\frac{3}{2}}(b_0 + b_1) \\ b_0 + 2b_1 + 2b_1 + b_2 \\ -\sqrt{\frac{3}{2}}(b_1 + b_2) \\ b_1 + 2b_2 + 2b_2 + b_3 \\ -\sqrt{\frac{3}{2}}(b_2 + b_3) \\ b_2 + 2b_3 \end{bmatrix} \begin{matrix} 0 \\ \frac{1}{2} \\ 1 \\ \frac{3}{2} \\ 2 \\ \frac{5}{2} \\ 3 \end{matrix}, \tag{2.100}$$

respectively.

where

$$a = \frac{1}{4}a_0 + \frac{1}{12}a_1,\ b = -\frac{\sqrt{6}}{20}a_0 - \frac{\sqrt{6}}{30}a_1,\ c = \frac{1}{12}a_0 + \frac{1}{12}a_1,\ d = \frac{1}{10}a_0 + \frac{1}{10}a_1$$

$$e = -\frac{\sqrt{6}}{30}a_0 - \frac{\sqrt{6}}{20}a_1,\ f = \frac{1}{12}a_0 + \frac{1}{4}a_1 + \frac{1}{4}a_1 + \frac{1}{12}a_2,\ g = -\frac{\sqrt{6}}{20}a_1 - \frac{\sqrt{6}}{30}a_2$$

$$h = \frac{1}{12}a_1 + \frac{1}{12}a_2,\ i = \frac{1}{10}a_1 + \frac{1}{a1}a_2,\ j = -\frac{\sqrt{6}}{30}a_1 - \frac{\sqrt{6}}{20}a_2$$

$$k = \frac{1}{12}a_1 + \frac{1}{4}a_2 + \frac{1}{4}a_2 + \frac{1}{12}a_3,\ l = -\frac{\sqrt{6}}{30}a_2 - \frac{\sqrt{6}}{20}a_3,\ m = \frac{1}{12}a_2 + \frac{1}{12}a_3$$

$$o = \frac{1}{10}a_2 + \frac{1}{10}a_3,\ p = -\frac{\sqrt{6}}{30}a_2 - \frac{\sqrt{6}}{20}a_3, \text{ and } s = \frac{1}{12}a_2 + \frac{1}{4}a_3.$$

2.4 The Bvp4c

`Matlab` has a built-in routine which was proposed by Kierzenka and Shampine [41] called `bvp4c`. This `Matlab` program is used to solve a system of n first order differential equations with two-point boundary conditions. It can also be used to solve multipoint boundary problems but in this work, only the solution of two-point BVPs is discussed. The numerical method used by `Matlab`'s `bvp4c` is the finite difference method which makes use of Lobatto formula [42] which is a collocation formula and collocation polynomial that provides a continuous solution that is fourth-order accurate uniformly in a closed interval. To solve any BVP with `Matlab bvp4c`, it should be of first order and of the form

$$\mathbf{u}' = \mathbf{f}(x, \mathbf{u}, \mathbf{p}),\ x \in\ [a, b] \tag{2.101}$$

together with two point boundary conditions

$$\mathbf{bc}(\mathbf{u}a, \mathbf{u}b, \mathbf{p}) = \mathbf{0}. \tag{2.102}$$

where the scalar variable x is the independent variable and the dependent vector variable

$$\mathbf{u}(x) = \begin{bmatrix} u_1(x) \\ u_2(x) \\ \vdots \\ u_n(x) \end{bmatrix} \quad \Rightarrow \quad \mathbf{u}'(x) = \begin{bmatrix} u_1'(x) \\ u_2'(x) \\ \vdots \\ u_n'(x) \end{bmatrix}$$

The columns vectors $\mathbf{u}a$ and $\mathbf{u}b$ corresponding to $\mathbf{u}(a)$ and $\mathbf{u}(b)$ respectively, are the values of the column vector $\mathbf{u}$ evaluated at both end-points a and b. The vector $\mathbf{p}$ which contains unknown parameters is optional. `Matlab`'s `bvp4c` function solves the boundary value problem and returns the solution in the structure we will name `BVPsol` with the syntax given by

```
BVPsol = bvp4c(@odebratu,@bcbratu,initialsol);
```

Here `bvp4c` takes the following arguments:

1. `@odebratu` - a function handle which allows us to invoke the function `odebratu` from any part of the program. It is the one used for evaluating the differential equation and it can take the form

$$d\mathbf{u}dx = \texttt{odebratu}(x, \mathbf{u})$$

2. `@bcbratu` - a function handle like `@odebratu`. The function `bcbratu` is for computing the residual in the boundary conditions, that is, a measure of how much the boundary conditions are not satisfied. `bcbratu` returns a column vector and it takes the form

$$\texttt{bcres} = \texttt{bcbratu}(\mathbf{u}a, \mathbf{u}b)$$

3. `initialsol` - a structure containing the initial guess to the solution which is created using the function `bvpinit`. `initialsol` has the first field x of ordered nodes where the endpoints are defined as

$$\text{a} = \texttt{initialsol}.\text{x}(1) \quad \text{and} \quad \text{b} = \texttt{initialsol}.(\text{end}).$$

The other field **u** contains the initial guess where `initialsol`.u(:,i) is an initial guess of $\mathbf{u}(x(i))$ at node `initialsol`.x(i). The function `initialsol` is used with syntax

```
initialsol=bvpinit(xmesh,uinit);
```

where `xmesh` is the mesh formed on the problem domain $[a, b]$ and `uinit` is the initial guess of the solution.

The value of the solution **u** at each mesh point `xmesh` can be evaluated using the function `deval` as follows

```
uatx=deval(BVPsol,xmesh)
```

2.4.1 Application to the Bratu problem

To transform Bratu's problem (1.1) to a linear differential equation let $u_1(x) = u(x)$ and also $u_2(x) = \dfrac{du(x)}{dx}$, to get

$$\frac{du_2(x)}{dx} = -\lambda e^{u_1(x)}, \tag{2.103a}$$

$$u_1(0) = u_1(1) = 0. \tag{2.103b}$$

2.5 The Spectral Collocation Method

To solve a differential problem on a physical domain $[a, b]$ on the x-axis, it is convenient to transform $[a, b]$ to the interval $[-1, 1]$ on the ξ-axis using the transformation

$$x = \frac{1-\xi}{2}a + \frac{1+\xi}{2}b, \quad -1 \leq \xi \leq 1 \tag{2.104}$$

See Figure 2.7

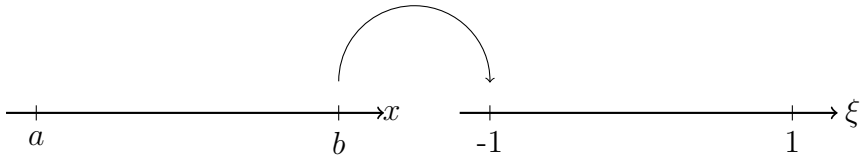


Figure 2.7: Linear transformation

Like the finite element method, some discretization

$$-1 = \xi_E < \xi_{E-1} < \ldots < \xi_0 = 1$$

is made where

$$\xi_c = \cos\left(\frac{\pi c}{E}\right), \ c = 0, 1, 2, \ldots, E$$

are called Chebyshev collocation points. These points are not equally spaced on $[-1, 1]$ and can be viewed as the projection on $[-1, 1]$ of equispaced points on the upper half of the unit circle [21] shown in Figure 2.8

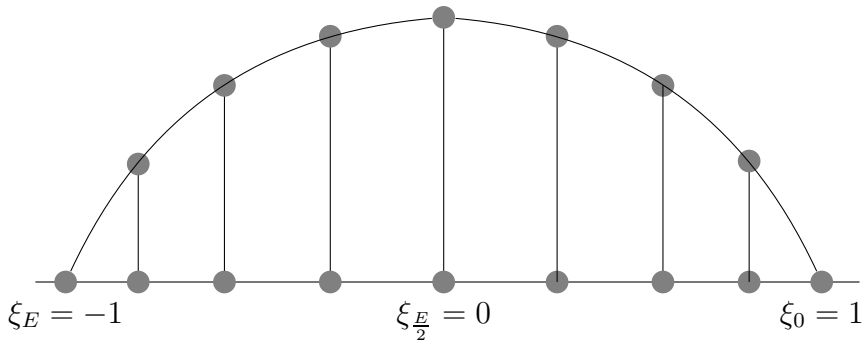


Figure 2.8: Chebyshev points

On $[-1, 1]$, the problem variable approximation is

$$u(\xi) \approx \sum_{c=0}^{E} u_c L_c(\xi) \tag{2.105}$$

where $u_c = u(\xi_c)$ and

$$L_c(\xi) = \prod_{k=0, k\neq c}^{E} \frac{\xi - \xi_k}{\xi_c - \xi_k}$$

is the Lagrange polynomial of degree E associated with node $\xi = \xi_c$. When $E = 1$, equation (2.105) becomes

$$u(\xi) \;\; = \;\; u_0 L_0(\xi) + u_1 L_1(\xi) \tag{2.106}$$

where

$$L_0(\xi) = \frac{1+\xi}{2} \;\text{ and }\; L_1(\xi) = \frac{1-\xi}{2} \tag{2.107}$$

Approximation of derivatives at collocation points gives

$$u'(\xi_0) = \frac{1}{2}u_0 - \frac{1}{2}u_1 \;\text{ and }\; u'(\xi_1) = \frac{1}{2}u_0 - \frac{1}{2}u_1$$

or

$$\mathbf{u}' = D\mathbf{u}$$

where $\mathbf{u} = [u_0 \;\; u_1]^T$ and

$$D = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

is the so-called Chebyshev differentiation matrix when $E = 1$. Similarly, when $E = 2$

$$u(\xi) = u_0 L_0(\xi) + u_1 L_1(\xi) + u_2 L_2(\xi) \tag{2.108}$$

where

$$L_0(\xi) = \frac{\xi(\xi+1)}{2}, \;\; L_1(\xi) = 1 - \xi^2 \;\; \text{and} \;\; L_2(\xi) = \frac{\xi(\xi-1)}{2}$$

Approximation of derivatives at collocation points gives

$$\begin{aligned} u'(\xi_0) &= \frac{3}{2}u_0 - 2u_1 + \frac{1}{2}u_2 \\ u'(\xi_1) &= \frac{1}{2}u_0 - \frac{1}{2}u_2 \\ u'(\xi_2) &= -\frac{1}{2}u_0 + 2u_1 - \frac{3}{2}u_2 \end{aligned} \tag{2.109}$$

or

$$\mathbf{u}' = D\mathbf{u}$$

where $\mathbf{u} = [u_0 \;\; u_1 \;\; u_2]^T$ and

$$D = \begin{pmatrix} \frac{3}{2} & -2 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{2} & 2 & -\frac{3}{2} \end{pmatrix}$$

is the Chebyshev differentiation matrix when $E = 2$. Generally, for any $E = 1, 2, 3, \ldots$, $\mathbf{u}' = D\mathbf{u}$ where D is given by the following theorem.

Theorem 1. For each $E \geq 1$, let the rows and columns of the $(E+1) \times (E+1)$ Chebyshev differentiation matrix D be indexed from 0 to E. The entries of this matrix are

$$\begin{aligned} D_{00} &= \frac{2E^2+1}{6}, \;\; D_{EE} = -\frac{2E^2+1}{6} \\ D_{cc} &= -\frac{\xi_c}{2(1-\xi_c^2)}, \;\; c = 1, 2, ..., E-1 \\ D_{ci} &= \frac{a_c(-1)^{c+i}}{a_i(\xi_c - \xi_i)}, \;\; c \neq i, \;\; i, c = 1, 2, ..., E-1. \end{aligned}$$

where

$$a_c = \begin{cases} 2, & c = 0, E \\ 1, & -1 \leq c \leq 1 \end{cases}$$

Since $\mathbf{u}' = D\mathbf{u}$, then

$$\mathbf{u}'' = (\mathbf{u}')' = (D\mathbf{u})' = D\mathbf{u}' = D(D\mathbf{u}) = D^2\mathbf{u}$$

Generally,

$$\frac{d^p\mathbf{u}}{d\xi^p} = D^p\mathbf{u}$$

for each $p = 1, 2, \ldots$. Evaluating a given differential equation

$$a_n(\xi)\frac{d^n u}{d\xi^n} + a_{n-1}(\xi)\frac{d^{n-1}u}{d\xi^{n-1}} + \ldots + a_1(\xi)\frac{du}{d\xi} + a_0(\xi) = f(\xi) \tag{2.110}$$

at each $\xi_0, \xi_1, \ldots, \xi_N$ gives the system

$$A_n\frac{d^n\mathbf{u}}{d\xi^n} + A_{n-1}\frac{d^{n-1}\mathbf{u}}{d\xi^{n-1}} + \ldots + A_1\frac{d\mathbf{u}}{d\xi} + A_0\mathbf{u} = \mathbf{f} \tag{2.111}$$

of n^{th} order differential equations where

$$A_n = \text{diag}\{a_n(\xi_0), \ldots, a_n(\xi_E)\} = \begin{pmatrix} a_n(\xi_0) & & & \\ & a_n(\xi_1) & & \\ & & \ddots & \\ & & & a_n(\xi_E) \end{pmatrix} \tag{2.112}$$

is an $(E+1) \times (E+1)$ diagonal matrix,

$$\mathbf{u} = \begin{bmatrix} u(\xi_0) & u(\xi_1) & \ldots & u(\xi_E) \end{bmatrix}^T \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} f(\xi_0) & f(\xi_1) & \ldots & f(\xi_E) \end{bmatrix}^T$$

Using Chebyshev differentiation equation, (2.111) is replaced by

$$(A_n D^n + A_{n-1}D^{n-1} + \ldots + A_1 D + A_0)\mathbf{u} = \mathbf{f}$$

or in short

$$A\mathbf{u} = \mathbf{f} \tag{2.113}$$

where

$$A = A_n D^n + A_{n-1} D^{n-1} + \ldots + A_1 D + A_0$$

Suppose differential equation (2.110) is subject to boundary conditions $u(-1) = \alpha$ and $u(1) = \beta$, then we include them into equation (2.113) as follows

$$\left(\begin{array}{cccc} 1 & 0 & \ldots & 0 \\ \hline & & & \\ & & A & \\ & & & \\ \hline 0 & \ldots & 0 & 1 \end{array}\right) \left(\begin{array}{c} u_0 \\ \hline \\ \mathbf{u} \\ \\ \hline u_E \end{array}\right) = \left(\begin{array}{c} \beta \\ \hline \\ \mathbf{f} \\ \\ \hline \alpha \end{array}\right) \tag{2.114}$$

Since the first equation is for determining u_0 which is known from boundary condition $u(1) = \beta$ we do not need first equation so we delete it and replace with $u_0 = \beta$. Similarly, last equation is replaced with $u_E = \alpha$. Once $u_0, u_1, \ldots, u_E$ are known upon solving linear system (2.114), equation (2.105) determines the value of u at any $\xi \in [-1, 1]$.

2.5.1 Application to the Bratu's problem

So far (1.1) has been transformed by quasi-linearization to

$$u''_{s+1}(x) + \lambda e^{u_s(x)} u_{s+1}(x) = \lambda e^{u_s(x)} (u_s(x) - 1), \ s = 0, 1, 2, \ldots \tag{2.115a}$$

$$u_{s+1}(0) = u_{s+1}(1) = 0 \tag{2.115b}$$

which upon using Chebyshev differentiation is replaced by

$$D^2 \mathbf{u}_{s+1} + B\mathbf{u}_{s+1} = \mathbf{r}_s, \ s = 0, 1, 2, \ldots \tag{2.116}$$

or

$$A\mathbf{u}_{s+1} = \mathbf{r}_s, \;\; s = 0, 1, 2, \ldots \tag{2.117a}$$

$$u_{s+1}(0) = u_{s+1}(1) = 0 \tag{2.117b}$$

where

$$\begin{aligned} B &= \text{diag}\{\lambda e^{u_s(x)}\}, \\ A &= D^2 + B, \\ \mathbf{r}_s(\mathbf{u}_s) &= \lambda e^{\mathbf{u}_s} \circ (\mathbf{u}_s - \mathbf{i}) \end{aligned}$$

$\mathbf{i} = \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix}^T$ and $A \circ F$ denotes the Hadamard product, a matrix of the same size as A and F whose elements are given by

$$[A \circ F]ij = [A]_{ij}[F]_{ij}$$

Hence $\circ$ denotes elementwise multiplication for matrices. Before we solve the linear system (2.117a) we include boundary conditions (2.117b) as follows

$$\left(\begin{array}{cccc} 1 & 0 & \ldots & 0 \\ \hline & & & \\ & & A & \\ & & & \\ \hline 0 & \ldots & 0 & 1 \end{array}\right)\left(\begin{array}{c} 0 \\ \hline \\ \mathbf{u}_{s+1} \\ \\ \hline 0 \end{array}\right) = \left(\begin{array}{c} 0 \\ \hline \\ \mathbf{r}_s \\ \\ \hline 0 \end{array}\right)$$

In order to generate subsequent approximations $\mathbf{u}_{s+1}$, $s = 1, 2, \ldots, E$, we choose the initial approximation

$$\mathbf{u}_0 = \begin{bmatrix} x_0(1-x_0) \\ x_1(1-x_1) \\ \vdots \\ x_E(1-x_E) \end{bmatrix} \tag{2.118}$$

so that boundary conditions (2.117b) are satisfied. Hence, successive approximations are

$$\mathbf{u}_{s+1} = A^{-1}\mathbf{r}_s$$

for each $s = 0, 1$

Chapter 3

Results and discussion

3.1 Introduction

In this chapter we present the results of the finite element solutions, `Matlab bvp4c` solution as well as the Chebyshev spectral collocation method solution of the Bratu's problem and their discussion.

3.2 Finite element solution using piecewise linear Lagrange polynomials as basis functions (LFEM)

The linear system (2.47) is solved using the `Matlab` code given by **Program** 1 in Appendix A for $E = 20$ and $\lambda = 1$. A comparison between the FE solution using piecewise linear Lagrange basis functions and the exact solution (1.2) is done in Figure 3.1

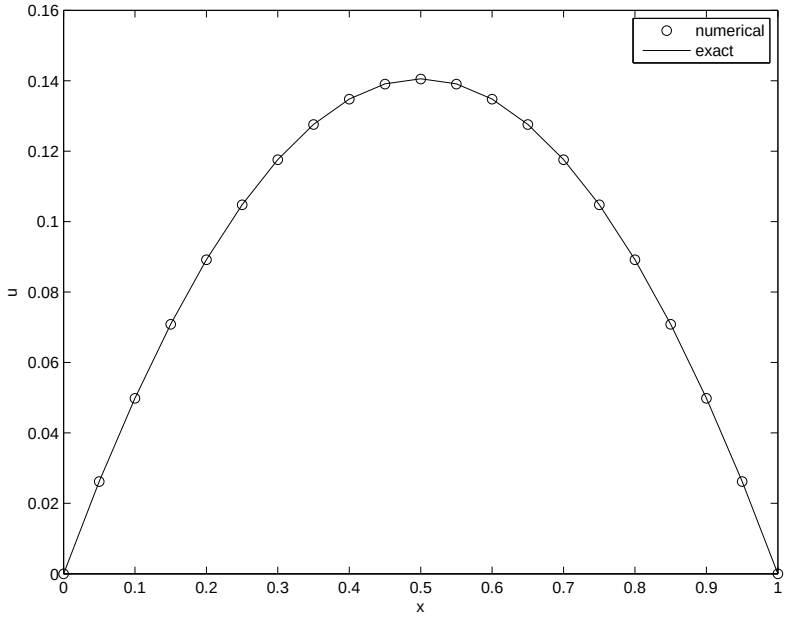


Figure 3.1: FE solution using piecewise linear Lagrange basis functions.

A closer comparison of the results shown in Figure 3.1 is done in Table 3.1 and it shows that the finite element solution using piecewise linear Lagrange basis functions gives a good approximation to the exact solution with values agreeing up to 4 decimal places with a maximum absolute error 0.5×10^{-4}.

x	exact solution $u(x)$	LFEM	absolute error($\times 10^{-4}$)
0.1	0.04985	0.04983	0.2
0.2	0.08919	0.08916	0.3
0.3	0.11761	0.11757	0.4
0.4	0.13479	0.13474	0.5
0.5	0.14054	0.14049	0.5
0.6	0.13479	0.13474	0.5
0.7	0.11761	0.11757	0.4
0.8	0.08919	0.08916	0.3
0.9	0.04985	0.04982	0.3

Table 3.1: Results for FEM solution with linear Lagrange basis

3.3 Finite element solution using piecewise quadratic Lagrange polynomials as basis functions (QFEM)

The `Matlab` code for the solution of (2.75) with piecewise quadratic lagrange basis functions is given in **Program** 2 in Appendix A. The comparison of the results of the exact solution against the FE solution using piecewise quadratic Lagrange basis functions is shown in Figure 3.2 and Table 3.2. The results from Table 3.2 show a good approximation of the exact solution by the FE solution using quadratic lagrange basis functions with values agreeing up to 3 decimal places. The comparisons made in Table 3.1 and Table 3.2 show that results for LFEM with a maximum absolute error 0.5×10^{-4} are closer to the exact solution than those QFEM with a maximum absolute error 0.23×10^{-3}.

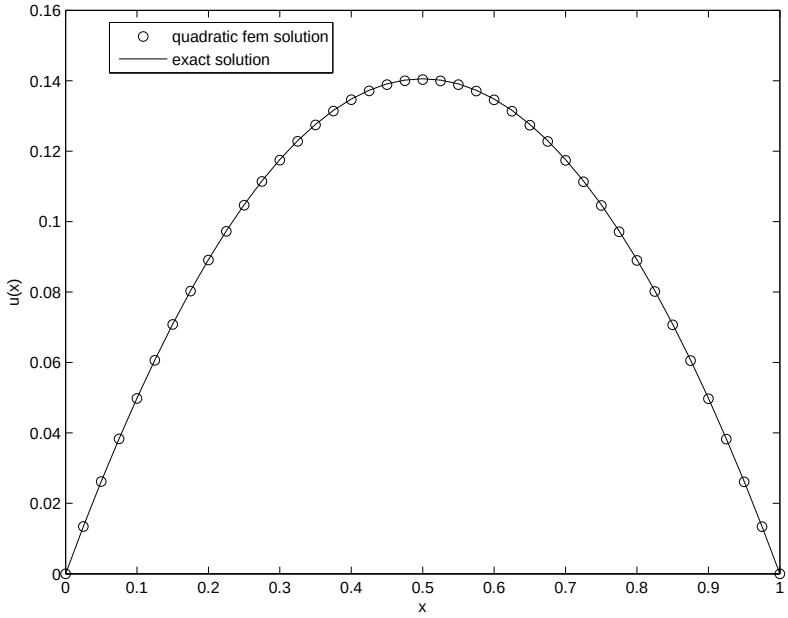


Figure 3.2: FE solution using piecewise quadratic Lagrange basis functions.

A closer comparison is presented in Table 3.2

x	exact solution $u(x)$	QFEM	absolute error($\times 10^{-3}$)
0.1	0.04985	0.04979	0.06
0.2	0.08919	0.08909	0.10
0.3	0.11761	0.11746	0.15
0.4	0.13479	0.13460	0.19
0.5	0.14054	0.14033	0.21
0.6	0.13479	0.13457	0.22
0.7	0.11761	0.11738	0.23
0.8	0.08919	0.08898	0.21
0.9	0.04985	0.04970	0.15

Table 3.2: Results for FE solution using piecewise quadratic Lagrange basis functions

3.4 Finite element solution using quadratic hierarchical basis functions (QHFEM)

We solved the FE equations given by (2.95) on a mesh of 20 elements with $\lambda = 1$ using the `Matlab` code given in **Program 3** to get results with shown in Figure 3.3.

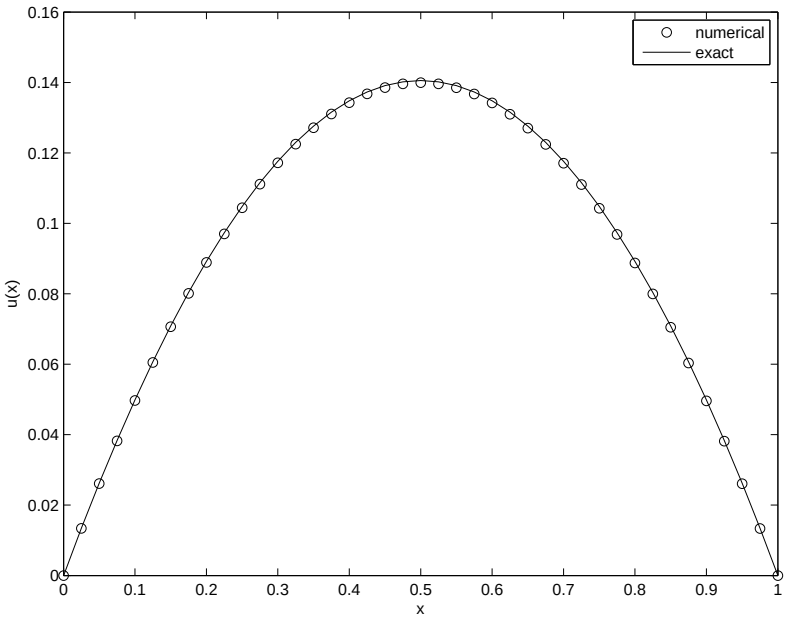


Figure 3.3: FE solution using piecewise quadratic hierarchical basis functions.

Numerical and exact results are in good agreement. A closer comparison is done in Table 3.3.

x	exact solution $u(x)$	QHFEM	absolute error($\times 10^{-3}$)
0.1	0.04985	0.04967	0.18
0.2	0.08919	0.08886	0.33
0.3	0.11761	0.11714	0.47
0.4	0.13479	0.13423	0.56
0.5	0.14054	0.13995	0.59
0.6	0.13479	0.13423	0.56
0.7	0.11761	0.11713	0.48
0.8	0.08919	0.08883	0.36
0.9	0.04985	0.04965	0.20

Table 3.3: Results for FE solution using piecewise quadratic hierarchical basis functions

The results in Table 3.3 show that the FE solution using hierarchical basis functions and the exact solution agree up to 3 decimal places and the solution of the former with maximum error 0.59×10^{-3} is less closer to the exact solution as the quadratic FE solution with maximum absolute error 0.23×10^{-3}.

3.5 Results from using bvp4c to solve Bratu's problem

As in the FE solution, we solve equation (2.103a) subject to boundary conditions (2.103b) using `Matlab`'s `bvp4c` on a mesh of 20 elements with $\lambda = 1$ using the `Matlab` code shown in **Program** 4 in Appendix A. The results of the comparison between the exact solution and the `Matlab bvp4c` is shown in Figure 3.4.

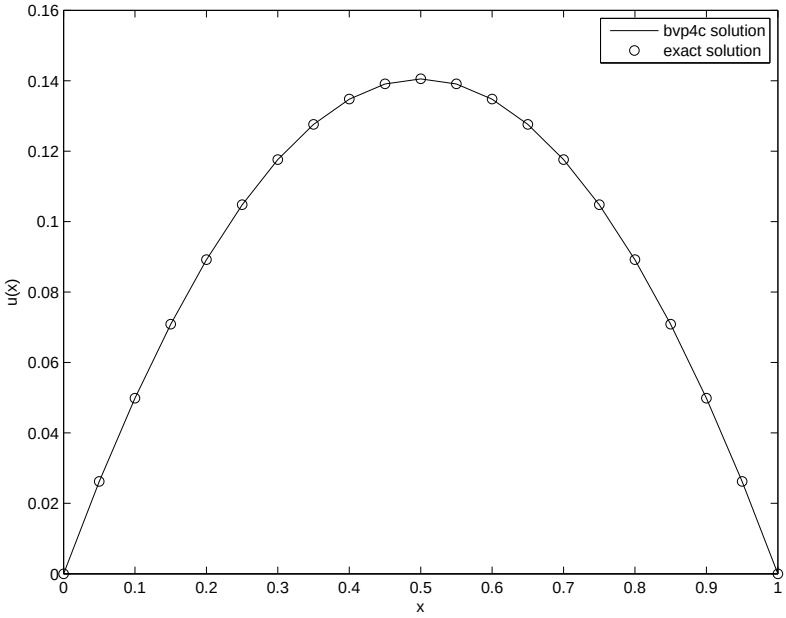


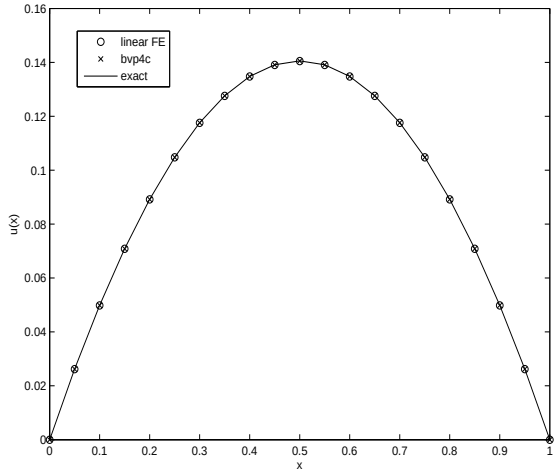
Figure 3.4: Matlab bvp4c solution of the Bratu problem.

Generally, there is a good agreement between the exact solution and the `bvp4c` solution. A close comparison of the results in Figure 3.4 is done in Table 3.4

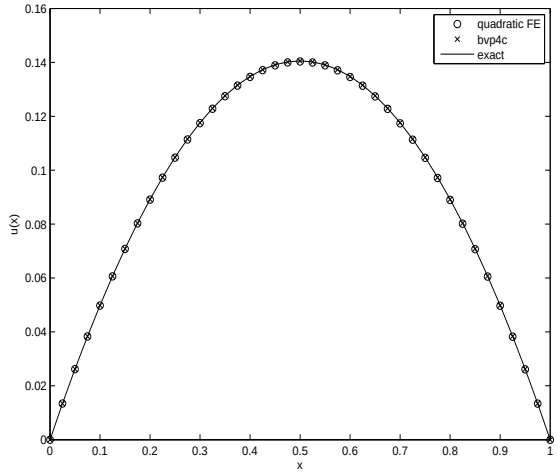
x	exact solution $u(x)$	`bvp4c`	absolute error($\times 10^{-4}$)
0.1	0.04985	0.04984	0.07029
0.2	0.08919	0.08918	0.13436
0.3	0.11761	0.11759	0.18586
0.4	0.13479	0.13477	0.21929
0.5	0.14054	0.14052	0.23088
0.6	0.13479	0.13477	0.21929
0.7	0.11761	0.11759	0.18586
0.8	0.08919	0.08918	0.13436
0.9	0.04985	0.04984	0.07029

Table 3.4: Results for exact solution against bvp4c solution

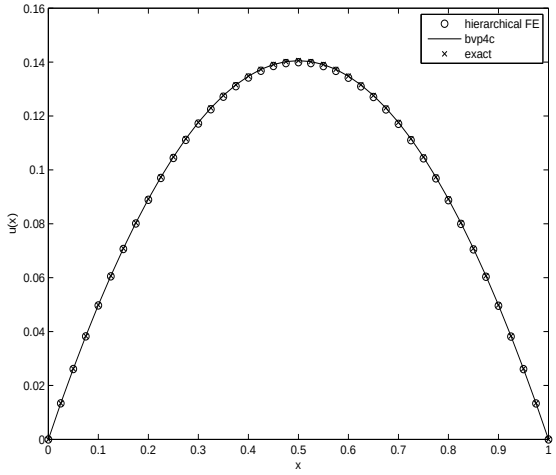
The results shown in Table 3.4 show that `bvp4c` solution gives results which are correct to within 4 decimal places of the exact solution. A comparison of the `bvp4c`, linear FE, quadratic FE and the hiererchical basis functions solutions is made in Figure 3.5.



(a)



(b)



(c)

Figure 3.5: A comparison of FE solutions and Matlab bvp4c solution

A closer comparison of results in Figure 3.5 is made in Table 3.7

x	exact	`bvp4c`	LFEM	QFEM	QHFEM
0.1	0.04985	0.04984	0.04983	0.04979	0.04969
0.2	0.08919	0.08918	0.08916	0.08909	0.08889
0.3	0.11761	0.11759	0.11757	0.11746	0.11717
0.4	0.13479	0.13477	0.13474	0.13460	0.13426
0.5	0.14054	0.14052	0.14049	0.14033	0.13995
0.6	0.13479	0.13477	0.13474	0.13457	0.13420
0.7	0.11761	0.11759	0.11757	0.11738	0.11707
0.8	0.08919	0.08918	0.08916	0.08898	0.08875
0.9	0.04985	0.04984	0.04983	0.04970	0.04957

Table 3.5: Results for FE solutions against bvp4c solution

According to results from Table 3.5 it can be deduced that the four solutions can be arranged in order of decreasing accuracy as follows:

1. `bvp4c`
2. linear FE
3. quadratic FE
4. hierarchical basis functions solution.

The results of the comparison between the number of iterations and amount of time required to achieve specified accuracy for a given number of elements is done in Table 3.6.

No. of elements(E)	No. of iterations	Decimal places	Runtime(s)
	LFEM		
20	70	4	0.11394
40	19	4	0.03651
500	1728	6	81.9406
	QFEM		
20	5	4	0.005715
40	4	4	0.018514
500	35	6	7.209471
	QHFEM		
20	3	4	0.013925
40	3	4	0.024309
500	25	6	4.805692

Table 3.6: Comparison of iteration number against runtime for LFEM, QFEM and QHFEM

Results from Table 3.6 shows that piecewise linear approximation is computationally intensive since it requires too large a mesh $E = 500$ and too many iterations and runtime to achieve accuracy of up to 6 decimal places. Increasing the polynomials from linear to quadratic and then hierarchical reduces the number of iterations and runtime to achieve the required degree of accuracy.

3.6 Solution using Chebyshev spectral collocation method (CSCM)

Solution of problem (2.117a) using the Chebyshev spectral collocation method on a mesh of 20 element with $\lambda = 1$ was done using the `Matlab` code given as **Program 5** in Appendix A. A comparison of the Chebyshev spectral collocation solution, the exact solution and `Matlab bvp4c` solution is done in Figure 3.6. They look the same and a closer comparison of the results of the CSCM solution and exact solution is done in Table 3.7

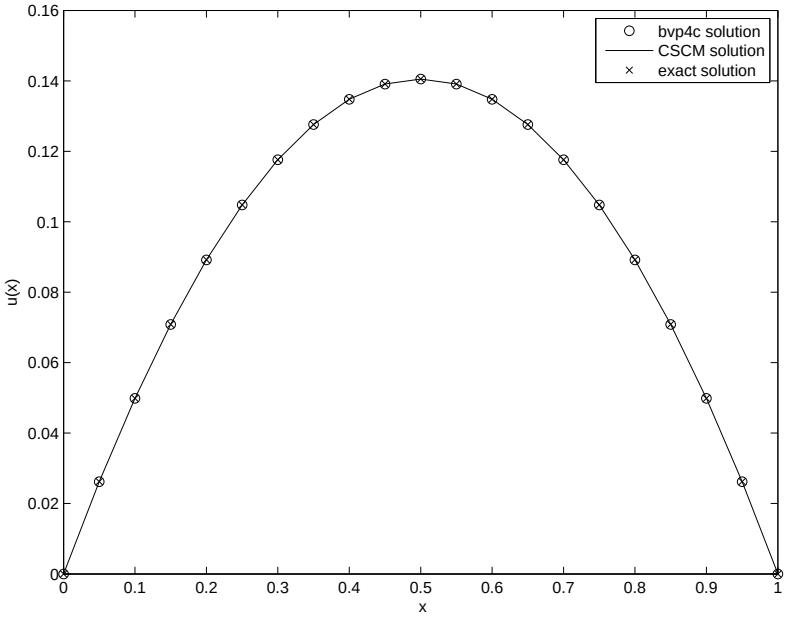


Figure 3.6: Comparison of bvp4c and CSCM solutions with the exact solution

x	exact solution $u(x)$	Matlab bvp4c	ξ	CSCM solution $u(\xi)$
0.1	0.049846791245413	0.049839767018395	-0.8	0.049846791245413
0.2	0.089189934628823	0.089176512344122	-0.6	0.089189934628823
0.3	0.117609095767941	0.117590532668624	-0.4	0.117609095767941
0.4	0.134790253884190	0.134768355546353	-0.2	0.134790253884189
0.5	0.140539214400472	0.140516160101251	0	0.140539214400472
0.6	0.134790253884190	0.134768355546353	0.2	0.134790253884190
0.7	0.117609095767941	0.117590532668624	0.4	0.117609095767941
0.8	0.089189934628823	0.089176512344122	0.6	0.089189934628823
0.9	0.049846791245413	0.049839767018395	0.8	0.049846791245413

Table 3.7: Results of the exact solution against CSCM solution

The results in Table 3.7 show that the CSCM is more accurate than `bvp4c` with the solutions agreeing up to 15 decimal places in the former.

Chapter 4

Conclusion and future work

In this work the one-dimensional Bratu problem is solved in two steps. First the differential equation is linearized using quasilinearization, then linearized equation is solved using

- FEM linear and quadratic polynomial basis functions
- Chebyshev spectral collocation method
- `Matlab's bvp4c`.

A comparison of the numerical results is made with the exact solution. Generally, there is good agreement. Results obtained show that the current form of the finite element is not favourable. In terms of accuracy, the finite element solutions are correct to 4 decimal places of the exact solution whereas Chebyshev spectral collocation method gave results correct to 13 decimal places easily. The accuracy of results was found to improve in the order HFEM, QFEM, LFEM and CSCM. The large mesh size and too many iterations needed to achieve desired accuracy make the finite element method using linear lagrange polynomials computationally intensive. This computational

complexity could be reduced by improving polynomial degree to quadratic and hierarchical basis functions. Achieving convergent results and ease of implementation was found to improve in the order LFEM, QFEM, HFEM and CSCM. Future work could include the use of methods other than the Garlekin method for constructing the trial space.

Appendix A

Computer Code

Program 1

```
% piecewise linear approximation
clear all % start with a clean workspace
E=500; h=1/E; % shall form a mesh on [0,1] with E elements of length h each
Ke=(1/h)*[1 -1; -1 1]; % element stiffness matrix
K=zeros(E+1); M=zeros(E+1); w=zeros(E+1,1); % initialize global matrices
x=[0:h:1]' ;% form uniform mesh on [0,1]
u=x.*(1-x); % initial guess u_0 of u
lambda=1; % choose constant in a of
%u''_{s+1}+a(u_s(x))u_s(x)=b(u_s(x)); s=0,1,...
ndp=6; tol=eval(sprintf('5e-%d',ndp+1)); % require accuracy to ndp
%decimal places?
du=1; % initialize 'distance' between successive u values
% must begin with du > tol. error checking needed here?????
it=0; % initialize iteration number
```

```
% begin exact solution
t0 = 1;
options = optimset('Display','iter','TolFun',1e-12);
tt = fsolve(@(tt) tt - sqrt(2*lambda)*cosh(tt/4),t0,options);
tt = tt(1);
uex = -2*log(cosh(0.5*(x-0.5)*tt)/cosh(0.25*tt)) % exact solution
% end exact solution
tic % start stopwatch
%-------------------------------------------------------------------------
while du>=tol % perform iteration as long as desired accuracy
    %not yet attained
    uold=u; % remember last u
    b=lambda*exp(u).*(u-1); a=lambda.*exp(u); % b(u_s(x)) and a(u_s(x))
    for e=1:E % assemble global matrices
    K([e e+1],[e e+1])=K([e e+1],[e e+1])+Ke; % stiffness matrix
    M([e e+1],[e e+1])=M([e e+1],[e e+1])+(h/12)*[
        3*a(e)+a(e+1) a(e)+a(e+1)
        a(e)+a(e+1) a(e)+3*a(e+1)
    ]; % mass matrix
    w([e e+1])=w([e e+1])+(h/6)*[
        2*b(e)+b(e+1)
        b(e)+2*b(e+1)
    ];% load vector
    end
    A=M-K; % FE equations are (A(u_s(x)))*u_{s+1}(x)=w(u_s(x))
    %where A(u_s(x))=M(u_s(x))-K
    % include boundary conditions;
    AA=A(2:E,2:E); % delete first, last rows of A
    ww=w(2:E); % delete first, last columns of w
    u=AA\ww; % compute u_{s+1}(x) given u_s(x)
```

```
    it=it+1 % increment iteration number
    u=[0; u; 0]; % u together with boundary values
    du=norm(u-uex,inf); % how close is current u to exact solution?
    fprintf('%10.0f\t %10.6f\n',it,du)
end
toc % end stopwatch
plot(x,u,'ro',x,uex,'b')
legend('numerical','exact')
xlabel('x')
ylabel('u')
%-------------------------------------------------------------------------
```

Program 2

```
%FE solution with quadratic Lagrange basis functions.
E=500; h=1/E; % form mesh on [0,1] with E elements of length h each
Ke=(1/(3*h))*([7 -8 1; -8 16 -8; 1 -8 7]); % element stiffness matrix
K=zeros(2*E+1);M=zeros(2*E+1); w=zeros(2*E+1,1);%initialize global matrices
x=[0:h/2:1]'; % form uniform mesh on [0,1]
u=x.*(1-x); % initial guess u_0 of u for quadratic fem
lambda=1; % choose constant in a of u''_{s+1}+a(u_s(x))u_s(x)=b(u_s(x))
% for s=0,1,...
ndp=6; tol=eval(sprintf('5e-%d',ndp+1));%accuracy to ndp decimal places?
du=1; % initialize 'distance' between successive u values
% begin exact solution
it=0;
t0 = 1;
options = optimset('Display','iter','TolFun',1e-12);
tt = fsolve(@(tt) tt - sqrt(2*lambda)*cosh(tt/4),t0,options);
```

```
tt = tt(1);
%x=x(1:2:end);
uex = -2*log(cosh(0.5*(x-0.5)*tt)/cosh(0.25*tt)); % exact solution
%-------------------------------------------------------------------------
tic % start stopwatch
while du>=tol %perform iteration as long as desired accuracy not attained
uold=u; % remember last u
%coefficients for quadratic basis fem
a=lambda.*exp(u);
b=lambda.*exp(u).*(u-1);

%-------------------------------------------------------------------------
  for e=1:E % assemble global matrices
    K([2*e-1 2*e 2*e+1],[2*e-1 2*e 2*e+1])=K([2*e-1 2*e 2*e+1],[
        2*e-1 2*e 2*e+1])+Ke;% stiffness matrix
%-------------------------------------------------------------------------
    m=(13/140)*a(e)+(1/21)*a(e+1)-(1/140)*a(e+2);
    n=(1/21)*a(e)+(4/105)*a(e+1)-(2/105)*a(e+2);
    o=(-1/140)*a(e)+(-2/105)*a(e+1)-(1/140)*a(e+2);
    q=(4/105)*a(e)+(16/35)*a(e+1)+(4/105)*a(e+2);
    r=(-2/105)*a(e)+(4/105)*a(e+1)+(1/21)*a(e+2);
    s=(-1/140)*a(e)+(1/21)*a(e+1)+(13/140)*a(e+2);
%-------------------------------------------------------------------------
    M([2*e-1 2*e 2*e+1],[2*e-1 2*e 2*e+1])=M([
    2*e-1 2*e 2*e+1],[2*e-1 2*e 2*e+1])+(h)*[
    [m  n o
     n  q  r
     o  r  s   ]

];% mass matrix
```

```
    w([2*e-1 2*e 2*e+1])=w([2*e-1 2*e 2*e+1])+(h/30)*[
        4*b(e)+2*b(e+1)-b(e+2)
        2*b(e)+16*b(e+1)+2*b(e+2)
        -b(e)+2*b(e+1)+4*b(e+2)
    ];% load vector
 end

 %-------------------------------------------------------------------------
 %solution for quadratic fem
    A=M-K; % FE equations are (A(u_s(x)))*u_{s+1}(x)=w(u_s(x)) where
    %A(u_s(x))=M(u_s(x))-K
    % include boundary conditions;
    AA=A(2:2*E,2:2*E); % delete first, last rows of A
    ww=w(2:2*E); % delete first, last columns of w
    u=AA\ww; % compute u_{s+1}(x) given u_s(x)
    %error = norm(u-uex,inf)
    it=it+1 % increment iteration number
    u=[0; u; 0]; % u together with boundary values
    du=norm(u-uold,inf); % how close is current u to previous u?
    fprintf('%10.0f\t %10.6f\n',it,du)
end
 toc % end stopwatch
 format long
 %u=u(1:2:end);
 plot(x,u,'ro',x,uex,'b')
 legend('quadratic fem solution','exact solution')
 xlabel('x')
 ylabel('u(x)')
 %title('Quadratic Lagrange FE solution ')
 %-------------------------------------------------------------------------
```

Program 3

```
% piecewise quadratic hierarchical approximation
% clear all % start with a clean workspace
E=500; h=1/E; % shall form a mesh on [0,1] with E elements of length h each
Ke=(1/h)*([1 0 -1; 0 2 0; -1 0 1 ]); % element stiffness matrix
% initialize global matrices
K=zeros(2*E+1);M=zeros(2*E+1); w=zeros(2*E+1,1);
x=(0:h/2:1)';% form uniform mesh on [0,1]
u=x.*(1-x); % initial guess u_0 of u
% choose constant in a of u''_{s+1}+a(u_s(x))u_s(x)=b(u_s(x)); s=0,1,...
lambda=1;
 % require accuracy to ndp decimal places?
ndp=6; tol=eval(sprintf('5e-%d',ndp+1));
du=1; % initialize 'distance' between successive u values
% must begin with du > tol. error checking needed here?????
it=0; % initialize iteration number
% begin exact solution
t0 = 1;
options = optimset('Display','iter','TolFun',1e-12);
tt = fsolve(@(tt) tt - sqrt(2*lambda)*cosh(tt/4),t0,options);
tt = tt(1);
x=x(1:2:end);
uex = -2*log(cosh(0.5*(x-0.5)*tt)/cosh(0.25*tt));% exact solution
% end exact solution
tic % start stopwatch
% perform iteration as long as desired accuracy not yet attained
while du>=tol
    uold=u; % remember last u
```

```
a=lambda*exp(u);
b=lambda*exp(u).*(u-1); % b(u_s(x)) and a(u_s(x))
for e=1:E % assemble global matrices
K([2*e-1 2*e 2*e+1],[2*e-1 2*e 2*e+1])=K([2*e-1 2*e 2*e+1],[
    2*e-1 2*e 2*e+1])+Ke;% stiffness matrix
%-------------------------------------------------------------------------
%expressions for mass matrix elements
m=(1/4)*a(e)+(1/12)*a(e+1);
n=(-sqrt(6)/20)*a(e)-(sqrt(6)/30)*a(e+1);
o=(1/12)*a(e)+(1/12)*a(e+1);
q=(1/10)*a(e)+(1/10)*a(e+1);
r=(-sqrt(6)/30)*a(e)-(sqrt(6)/20)*a(e+1);
u=(1/12)*a(e)+(1/4)*a(e+1);
%-------------------------------------------------------------------------
M([2*e-1 2*e 2*e+1],[2*e-1 2*e 2*e+1])=M([
2*e-1 2*e 2*e+1],[2*e-1 2*e 2*e+1])+(h)*[
m  n  o
n  q  r
o  r  u
];% mass matrix
w([2*e-1 2*e 2*e+1])=w([2*e-1 2*e 2*e+1])+(h/6)*[
    2*b(e)+b(e+1)
   (-sqrt(3/2))*(b(e)+b(e+1))
   b(e)+2*b(e+1)
];% load vector
end
% FE equations are (A(u_s(x)))*u_{s+1}(x)=w(u_s(x))
%where A(u_s(x))=M(u_s(x))-K
A=M-K;
% include boundary conditions;
```

```
    AA=A(2:2*E,2:2*E); % delete first, last rows of A
    ww=w(2:2*E); % delete first, last columns of w
    u=AA\ww; % compute u_{s+1}(x) given u_s(x)
    it=it+1 % increment iteration number
    format long
    u=[0; u; 0]; % u together with boundary values
    du=norm(u-uold,inf); % how close is current u to previous u?
    %fprintf('%10.0f\t %10.6f\t %10.6f\n',it,u(E/2),uex(E/2))
    %du=norm(u-uex,inf) % how close is current u to exact solution?
    fprintf('%10.0f\t %10.6f\n',it,du)
end
%error=u-uex;
toc % end stopwatch
u=u(1:2:end);
plot(x,u,'ro',x,uex,'b')
legend('numerical','exact')
xlabel('x')
ylabel('u(x)')
%title('FE solution with quadratic hierarchical basis functions')
%-------------------------------------------------------------------------
```

Program 4

```
%Bratu type u''(x)+exp(u(x))=0 solution using matlab bvp4c
function bvpBratu %function definition
lambda=1;% using the value lambda=1
E=20;%number of  elements
h=1/E;%lenght of each element
solinit=bvpinit([0:h:1],[-1 0]);% initial guess of the solution
```

```
BVPsol=bvp4c(@twoode,@twobc,solinit);%solving the differential equation
xmesh=[0:h:1];% forming a mesh of 20 elements
uatx=deval(BVPsol,xmesh);% evaluating the solution at mesh points
s=uatx(1,:);
%-------------------------------------------------------------------------
%exact solution
t0=1;
tt = fsolve(@(tt) tt - sqrt(2*lambda)*cosh(tt/4),t0);
tt = tt(1);
uex = -2*log(cosh(0.5*(xmesh-0.5)*tt)/cosh(0.25*tt));
%-------------------------------------------------------------------------
plot(xmesh,uatx(1,:),'r',xmesh,uex,'o')
xlabel('x')
ylabel('u(x)')
legend('bvp4c solution','exact solution')
%title('bvp4c solution of Bratu problem')
end
%-------------------------------------------------------------------------
function dudx=twoode(xmesh,uatx)
%evaluating the differential equation
dudx=[uatx(2);-exp(uatx(1))];
end
%-------------------------------------------------------------------------
function res = twobc(uatxa,uatxb)
%combuting residues in the boundary conditions
res=[uatxa(1); uatxb(1)];
end
%-------------------------------------------------------------------------
```

Program 5

```
function bvpCheb%function definition
E =20;h=1/E;% shall form a mesh on [-1,1] with E elements of length h each
[DifMatrx , x] = cheb(E);%computing differentiation matrix
xmesh=[-1:h:1];% form a mesh on [-1,1]
D2 = DifMatrx^2; D2 = D2(2:E,2:E);%applying the boundary condition
u = x.*(1-x);%initial guess of the solution
u=u(2:E);%applying boundar conditions
du = 1; it = 0;%initialise iteration number to zero
D21=D2+diag(exp(u));%equation in terms of differentiation matrices
%-------------------------------------------------------------------------
tic%start stopwatch
while du > 1e-4 % fixed-point iteration
f1=exp(u).*(u-1);
f=exp(u)/4;
unew = -D2\f;
du = norm(unew-u,inf);
u = unew; it = it+1;
end
toc%stop stopwatch
u = [0;u;0];%solution u(x)
format long
uu = polyval(polyfit(x,u,E),xmesh);
%-------------------------------------------------------------------------
%bvp4c solution
initialsol=bvpinit(-1:h:1,[-1 1]); % initial guess
BVPsol=bvp4c(@odebratu,@bcbratu,initialsol); % solving the d.e
uatx=deval(BVPsol,xmesh);%evaluating solution at mesh points
s=uatx(1,:);
%-------------------------------------------------------------------------
% begin exact solution
```

```
xmesh=[0:h:1];
lambda=1;
t0 = 1;
options = optimset('Display','iter','TolFun',1e-12);
tt = fsolve(@(tt) tt - sqrt(2*lambda)*cosh(tt/4),t0,options);
tt = tt(1);
uex = -2*log(cosh(0.5*(xmesh-0.5)*tt)/cosh(0.25*tt)); % exact solution
% end exact solution
%-------------------------------------------------------------------------
% plotting the solution
s=uatx(1,:);
s=s(1:2:end);
uu=uu(1:2:end);
plot(xmesh,s,'ro',xmesh,uu,xmesh,uex,'x')
xlabel(' x')
ylabel('u(x)')
legend('bvp4c solution','CSCM solution','exact solution')
%title('bvp4c,exact and CSCM solution')
end
%-------------------------------------------------------------------------
function dudx = odebratu(xmesh,uatx)
%evaluate differential equation for different coefficients a(x)
a=1/4;
dudx = [uatx(2);
-a*exp(uatx(1))];
end
%-------------------------------------------------------------------------
function [bcres] = bcbratu(uatxa,uatxb)
% compute residuals in boundary conditions
bcres= [uatxa(1);
```

```
uatxb(1)];
end
%-------------------------------------------------------------------------
```

Appendix B

Nomenclature

Ω_e	=	physical domain associated with element e
c	=	number of edges
x_{c-1}	=	vertex associated with edge $c-1$
$x_{c-\frac{1}{2}}$	=	vertex associated with edge $c-\frac{1}{2}$
x_c	=	vertex associated with edge c
E	=	number of elements
u	=	exact solution
v	=	test/weight function
U	=	trial solution
V	=	approximation of the weight function
ϕ_{c-1}	=	basis function associted with node $c-1$
$\phi_{c-\frac{1}{2}}$	=	basis function associted with node $c-\frac{1}{2}$
ϕ_c	=	basis function associted with node c
N_{c-1}	=	element shape function associted with node $c-1$

$N_{c-\frac{1}{2}}$	=	element shape function associted with node $c-\frac{1}{2}$
N_c	=	element shape function associted with node c
Π_e	=	canonoical problem domain
K_c	=	element stiffness matrix
M_c	=	element mass matrix
$\mathbf{w_c}$	=	element load vector
b_c	=	nodal value associated with node x_c
$E(x)$	=	Young's modulus
$N(x)$	=	resultant force (N)
L	=	length (m)
$A(x)$	=	cross sectional area (m^2)
$p(x)$	=	distributed force (N)

References

[1] Buckmire, R: Application of a Mickens finite-difference scheme to the cylindrical Bratu-Gelfand problem. Numer. Methods Partial Differ. Equ. 20(3), 327-337 (2004)

[2] Mounim, AS, de Dormale, BM: From the fitting techniques to accurate schemes for the Liouville-Bratu-Gelfand problem. Numer. Methods Partial Differ. Equ. 22(4), 761-775 (2006)

[3] Li, S, Liao, SJ: Analytic approach to solve multiple solutions of a strongly nonlinear problem. Appl. Math. Comput. 169, 854-865 (2005)

[4] G. Adomian, A review of the decomposition method and some recent results for nonlinear equations, Computers & Mathematics with Applications, vol. 21, no. 5, pp. 101127, 1991.

[5] G. Adomian, Solving frontier problems modelled by nonlinear partial differential equations, Computers & Mathematics with Applications, vol. 22, no. 8, pp. 9194, 1991.

[6] Aregbesola, Y. Numerical solution of Bratu problem using the method of weighted residual, Electronic Journal of Southern African Mathematical Sciences Association 3 (01), 17, 2003.

[7] Ascher, U.M., Matheij, R. and Russell, R.D. Numerical solution of boundary value problems for ordinary differential equations (SIAM, Philadelphia, PA, 1995)

[8] Caglara, H, Caglarb, N, zer, M: B-spline method for solving Bratus problem. Int. J. Comput. Math. 87(8), 1885-1891 (2010)

[9] Hassan, H. N., Semary, M. S. Analytic approximate solution for the Bratus problem by optimal homotopy analysis method. Communications in Numerical Analysis, 2013. doi.org/10.5899/2013/cna-00139.

[10] Ascher, U.M., Matheij, R. and Russell, R.D. Numerical solution of boundary value problems for ordinary differential equations (SIAM, Philadelphia, PA, 1995).

[11] Boyd, J.P. An analytical and numerical study of the two-dimensional Bratu equation, J.Scien. Computing 1 (2), 183206, 1986.

[12] Boyd, J. P. Chebyshev polynomial expansions for simultaneous approximation of two branches of a function with application to the one-dimensional Bratu equation, Appl. Math. Comput. 142, 189200, 2003.

[13] R.E. Bellman, R.E. Kalaba, Quasilinearization and Nonlinear Boundary-Value Problems, Elsevier Publishing Company, New York, 1965.

[14] V. Lakshmikantham, An extension of the method of quasilinearization, J. Optim. Theory Appl., 82 (1994), 315-321.

[15] David V. Hutton. Fundamentals Of Finite Element Analysis, The McGraw-Hill companies,New York,2004, page 1.

[16] Courant R. Variational methods for the solution of problems of equilibrium and vibrations. Bulletin of the American Mathematical Society, 1943, Vol. 49, P. 1-23.

[17] Clough R. W. The finite element method in plane stress analysis. Proc. American Society of Civil Engineers (2nd Conference on Electronic Computation, Pitsburg, Pennsylvania), 1960, Vol. 23, P. 345-378.

[18] Argyris J. H. Energy theorems and structural analysis. Aircraft Engineering, 1954, Vol. 26, Part 1 (Oct. Nov.), 1955, Vol. 27, Part 2 (Feb. May).

[19] Turner M. J., Clough R. W., Martin H. C. and Topp L. J. Stiffness and deflection analysis of complex structures. Journal of Aeronautical Science, 1956, Vol. 23, No. 9, P. 805-824.

[20] Hrennikov A. Solution of problems in elasticity by the frame work method. Journal of Applied Mechanics, 1941, Vol. 8, P. 169-175.

[21] Trefethen, L. N., Spectral Methods in MATLAB, SIAM, Philadelphia, PA,2000.

[22] Canuto, C, Hussaini, M.Y., Quarteroni, A., Zang, T.A., Spectral Methods in Fluid Dynamics, Springer Verlag, 1987.

[23] Gottlieb, D., Orszag, S.A., Numerical Analysis of Spectral Methods: Theory and Applications, SIAM, 1977.

[24] Gottlieb, D., Hussaini, M.Y., Orszag, S.A., Theory and Applications of Spectral Methods, in Spectral Methods for Partial Differential Equations, Ed. by R.G. Voigt, D. Gottlieb, M.Y. Hussaini, SIAM-CBMS, P. 1-54, 1984.

[25] Lanczos, C., Applied Analysis, Prentice Hall Inc., Englewood Cliffs, N. J.,1956

[26] Chandrasekhar, S. An Introduction to the Study of Stellar Structure. New York: Dover, pp. 84-182, 1967.

[27] R. J. Sylvester and F. Meyer, Two Point Boundary Problems by Quasilinearization, Journal of the Society for Industrial and Applied Mathematics, Vol. 13, No. 2 (Jun.,1965), pp. 586-602

[28] Liu, C. "Efficient shooting methods for the second-order ordinary differential equations." Computer Modeling in Engineering and Sciences 15.2 (2006): 69.

[29] V.B. Mandelzweig, F. Tabakin, Quasilinearization approach to nonlinear problems in physics with application to nonlinear ODEs, Computer Physics Comm., 141 (2001), 268-281.

[30] C. Chun, Iterative methods improving Newtons method by the decomposition method, Computers & Mathematics with Applications, vol. 50, no. 1012, pp. 15591568, 2005.

[31] Sandile S. Motsa and Precious Sibanda, On Extending the Quasilinearization Method to Higher Order Convergent Hybrid Schemes Using the Spectral Homotopy Analysis Method, Journal of Applied Mathematics, vol. 2013, Article ID 879195, 9 pages, 2013. doi:10.1155/2013/879195

[32] Vatsala, A. S. "Monotone iterative technique for singular systems of differential equations." Nonlinear Analysis and Applications 109 (1987): 579-582.

[33] Ladde, GS, Lakshmikantham, V, Vatsala, AS: Monotone Iterative Techniques for Nonlinear Differential Equations.Pitman, Boston (1985)

[34] Pao, C. V. Review: G. S. Ladde, V. Lakshmikantham, and A. S. Vatsala, Monotone iterative techniques for nonlinear differential equations . Bulletin (New Series) of the American Mathematical Society 18 (1988), no. 1, 65–67. http://projecteuclid.org/euclid.bams/1183554440

[35] Bellman, Richard Ernest. Perturbation techniques in mathematics, engineering and physics. Courier Dover Publications, 2003.

[36] R. Kalaba, On nonlinear differential equations, the maximum operation and monotone convergence, J. Math. Mech. 8 1959) 519.

[37] R.G. Bartle, D.R. Sherbert, (2000) Introduction to Real Analysis 3ed. John Wiley & Sons.

[38] Agarwal et al.: Method of quasilinearization for a nonlocal singular boundary value problem in weighted spaces. Boundary Value Problems 2013, 2013:261.

[39] Henwood, D., and Bonet, J. (1996). Finite elements: A gentle introduction. Houndmills, England: MacMillan.

[40] B. Szabǿ and I. Babuska. Finite Element Analysis. John Wiley and Sons, New York, 1991,page 38.

[41] J. Kierzenka and L. F. Shampine, A BVP solver that controls residual and error, Journal of NumericalAnalysis, Industrial and Applied Mathematics, vol. 3, no. 1-2, pp. 2741, 2008.

[42] Gonzlez Pinto, S., S. Prez Rodrguez, and J. I. Montijano Torcal. "On the numerical solution of stiff IVPs by Lobatto IIIA Runge-Kutta methods." Journal of computational and applied mathematics 82.1 (1997): 129-148.

[43] Clenshaw, C. W. 1962. Chebyshev series for mathematical functions. National Physical Laboratory Mathematical Tables, Volume 5, H.M.S.O.,London, 1962.

[44] Parand, Kourosh, and Mehdi Shahini. "Rational Chebyshev collocation method for solving nonlinear ordinary differential equations of Lane-Emden type."

[45] Chen, Hsin-Chu, T. L. Horng, and Y. H. Yang. "Reflexive decompositions for solving Poisson equation by Chebyshev pseudospectral method." Proceedings of Neural, Parallel, and Scientific Computations 4 (2010): 98-103.

[46] Bender, C.M., Milton, K.A., Pinsky, S.S. and Simmons Jr. L.M., A new perturbative approach to nonlinear problems, J. Math. Phys., 30 (1989) 1447-55.

[47] Shawagfeh, N.T., Nonperturbative approximate solution for Lane-Emden equation, J. Math. Phys., 34 (1993) 4364-69.

[48] Mandelzweig V.B. and Tabakin F., Quasilinearization approach to nonlinear problems in physics with application to nonlinear ODEs, Comput. Phys. Commun., 141 (2001) 268-281.

[49] Wazwaz, A.M., A new algorithm for solving differential equations of Lane-Emden type, Appl. Math. Comput., 118 (2001) 287-310.

[50] Wei, Yunxia, and Yanping Chen. "Convergence analysis of the spectral methods for weakly singular Volterra integro-differential equations with smooth solutions." Advances in Applied Mathematics and Mechanics 4.1 (2012): 1-20.

[51] Jean-Paul Berrut & Lloyd N. Trefethen (2004). "Barycentric Lagrange Interpolation". SIAM Review 46 (3): 501517. doi:10.1137/S0036144502417715.