NTNU
Norwegian University of
Science and Technology

Performance Analysis of Nonlinearly Controlled Motion Systems

Roger Eivind Stenbro

Master of Science in Engineering Cybernetics
Submission date: February 2009
Supervisor: Alexey Pavlov, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

Recently a new method for improving performance of linear control systems has been proposed. The method is based on utilizing nonlinear controllers for linear control systems. It has been shown that for the so-called nonlinear convergent systems one can extend the conventional performance evaluation tools such as frequency response functions from the linear systems theory. Yet, reliable numerical methods for computing such nonlinear frequency response functions still need to be developed.

The goal of this project is to develop such a numerical method for computing frequency response functions (FRF) for a class of convergent nonlinear systems. In particular, since computing such an FRF involves finding a solution to a partial differential equation, the applicability of the finite element method should be investigated. Other methods can be tested as well. The obtained numerical method should be tested on a nonlinearly controlled DVD drive as a benchmark system. Based on the outcome of the development of the numerical procedure, the candidate should propose a sensible performance measure for nonlinearly controlled systems and, if time permits, provide an algorithm for tuning controllers to improve their performance (corresponding to the proposed performance measure).

Assignment given: 21. September 2008
Supervisor: Alexey Pavlov, ITK

Abstract

In this report, a set of nonlinear coupled partial differential equations arising from the extension of frequency response functions to convergent nonlinear systems are studied.

It will be shown how the finite element method can be applied to various versions of the system, starting with a simple linear example, and ending in the finite element model equations for the complete coupled nonlinear system. Further, a solution will be attempted for some of the most usual nonlinear solution techniques. It will be argued that a finite element solution cannot be found if using only linear simplex elements. A possible way forward for the finite element method is presented, but due to several reasons not applied to studied in-depth. Additionally, an alltogether different method is also presented, which was an open option to apply.

The reader is assumed to be familiar with basic linear and nonlinear control theory and terminology. Additionally, some familiarity with the most basic aspects of the finite element method is advantegous, though some theoretical background will be given.

Contents

List of Figures

Chapter 1

Introduction

In industrial control applications *performance* has always been a very important design criterion. For general linear systems, a vast array of techniques and design tools already exist that allow controller tuning based on performance criteria. Exactly what the word performance entails may wary from system to system, though many practical control applications are based around some notion of *tracking* or *path following*, where the system must follow some precalculated trajectory as closely as possible, subject to noise and disturbances. For an excellent background in linear performance analysis, consider [5].

For nonlinear control systems, there is in general a lack of properly formulate performance criteria. Two factors can be mentioned as to why; a (much) higher level of mathematical knowledge is required for the formulation of such critera, which makes it more unapproachable for many practitioners. Further, the complexity of some of the numerical simulations involved have so far proved a limiting factor in practice as well. Simulations running 50-60 hours is not very efficient use of engineer time, especially when they have to be repeated numerous times in a complete synthesis.

For many practical applications purely linear control strategies prove sufficient. However there are also examples of systems where this does not hold, we will see one such example in form of a nonlinearly controlled optical storage drive in this text. Good tracking control is in general coupled with good disturbance attenuation properties at predominantly low frequencies in the linear control systems theory, however for a linear system, this unavoidably means poorer attenuation properties at higher frequencies. If the presence of high-frequency disturbances is likely in addition to the low-frequency content, and if the system requires high tracking performance, this introduces a problem with a purely linear control strategy.

If the system is noiseless, then perfect tracking is always achieveable. However in practice no system is ever noiseless. In particular there is always the presence of measurement noise. Consider the simple linear control system depicted in Figure 1.1. with a plant P, controller C, reference input r, error $\tilde{e}$, output p and measured output $\tilde{p} = p + n$ where n is the measurement noise. The output p of the system is to follow (track) the reference input r as closely as possible. What, then, is a good measure of *performance*? There are two quantities of special interest in this regard.

- How sensitive the system is to changes in the input (reference r).
- How sensitive the system is to noisy disturbances (herein the measurement noise n).

Both effects (reference changes and noise) are commonplace in any control system. A system will deviate from its desired trajectory subject to changes in either the reference or the noise.

In linear systems theory, these quantitites are analyzed according to the so-called *sensitivity* and *complementary sensitivity* functions, denoted $S(s)$ and $T(s)$ respectively. Insertion of the complex frequency $s = j\omega$ gives a frequency-dependent characteristic of how the reference trajectory affects the tracking error through $|S(j\omega)|$ and likewise how the measurement noise affects the output through $|T(j\omega)|$. The applicability of the

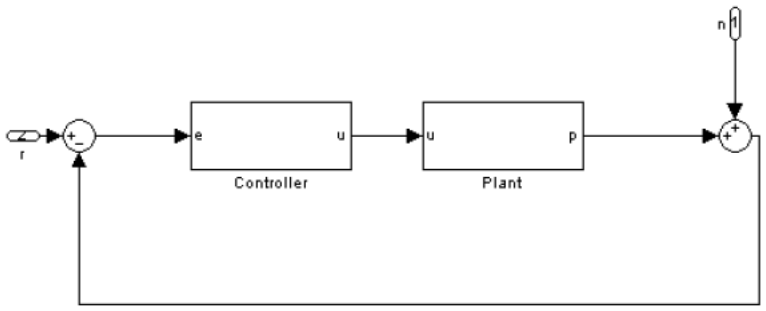

Figure 1.1: A linear control system.

laplace transform to the governing set of ordinary differential equations, and the fact that superposition is valid for any linear system, yield simple and effective analysis tools for performance based on, among other results, these functions.

Another powerful result of the linear systems theory is that of uniqueness of solution. Suppose a linear system, represented by the frequency domain transfer function $G(j\omega)$, is subjected to a general harmonic input of the form $u = a \sin \omega t$. Then, the steady-state output response p of the system will equal

$$p(t) = |G(j\omega)| \sin(\omega t + \phi) \tag{1.1}$$

where $\phi = -\arg(G(j\omega))$ is the phase shift of the output oscillatory response. Thus, when excited by a harmonic input u, linear systems have a unique steady-state response as per (1.1). This steady-state response is unique, and independent of the initial condition(s).

For general nonlinear systems these results vanish. There are two particurlarly vexing aspects of general nonlinear systems that are undesirable in terms of analyzing the steady-state response or results based on that response;

- It may have multiple coexisting solutions.
- The steady-state solution $x_w(t)$ may not be independent of the initial condition(s).

However a class of nonlinear systems exist that do not have these drawbacks; they are referred to as *convergent* systems. While the mathematical definitions will be given, an intuitive textual description is given first. A convergent system will, though nonlinear, have a well-defined *unique* steady-state response. In addition such a system will converge to this response from *any* initial condition. This definition of convergence is very general and applicable to any nonlinear system that satisfy these two criteria. For the systems considered in this report we will mainly be focused on one special class of nonlinear (and convergent) systems; namely the Lur'e systems.

In recent years, research into the properties of convergent systems have yielded some interesting results. As it turns out, for systems with the convergent property, nonlinear counterparts to sensitivity functions can be defined based on knowledge of the steady-state response. Controller design based on these and other performance-based techniques have been tried and tested in practice, as is the example with Philips' DVD drives for use in cars [2].

However, an obstacle is encountered with the calculation of these steady-state responses. Part of this is due to the desire that the system be analyzed with respect to general harmonic inputs, as per the linear systems theory. This creates a set of nonlinear partial differential equations, of which the solution is the sought response. Solving this system of equations is in itself a nontrivial task.

Early on, the solution to the system was simply found "brute force", repeatedly solving the system in state-space (of the form $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + ...$) with respect to harmonic inputs. This was computationally very demanding, and an interesting question arose as to whether applying some known numerical method to the problem could ease the computational demand and make the procedure more practically applicable. This is mainly the focus of this thesis. In particular, the applicability of the finite element method to the system will be investigated.

In fact, there are (now) practically applicable solution methods that may very well outdo a finite element solution, especially the method described in [8], which is also briefly outlined in Chapter 4 of this thesis.

The thesis is organized as follows. In Chapter 2, the theory and application considering general convergent dynamical systems is briefly presented. In Chapter 3, the finite element implementation is considered. In (a brief) Chapter 4, some possible ways forward, with or without the finite element method, is considered. Conclusions are presented in Chapter 5.

Chapter 2

Convergent Systems

This chapter aims to cover the needed background about convergent systems. The theory is mostly based on the papers [1] [2] [3] [4], as well as [22].

The systems considered can in state-space be written in the very general form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}(t)). \tag{2.1}$$

The system of (nonlinear) ordinary differential equations (hereafter ODEs) given by (2.1) has state $\mathbf{x} \in \mathbb{R}^n$ and input —$u(t) \in \mathbb{R}^m$. It is required that the inputs are (at least) piecewise continuous on $\mathbb{R}$. Then, the following defines a convergent system [].

Definition 2.1. The system (2.1) may be called;

- *convergent* if
  1. there exists a unique steady-state solution $\overline{x}_w(t)$ that is defined and bounded on $\mathbb{R}$,
  2. $\overline{x}_w(t)$ is asymptotically stable,
- *uniformly convergent* if $\overline{x}_w(t)$ is uniformly globally asymptotically stable (UGAS),
- *exponentially convergent* if $\overline{x}_w(t)$ is globally exponentially stable (GES).

It is seen that the requirements given in Definition 2.1 are quite general. Any system that possesses a unique steady-state solution, and is asymptotically stable, is a convergent system. All asymptotically stable linear systems are also convergent systems (in fact exponentially convergent). In that case the definition overlaps the usual definition of linear stability.

Definition 2.2. Systems of the form (2.1), that are convergent for some class of inputs $u(t)$, are said to have the uniformly bounded steady-state (UBSS) property if for any $\rho > 0$ there exists constant $M > 0$ such that;

$$|u(t)| \leq \rho \qquad \forall\ t \in \mathbb{R} \qquad \Rightarrow \qquad |\overline{x}_w(t)| \leq M \qquad \forall\ t \in \mathbb{R}. \tag{2.2}$$

In words, if the inputs are absolutely bounded by a constant value ρ, the steady-state output will also be absolutely bounded by some constant M. This is an extension of BIBO stability (bounded input bounded output) which is well known from LTI systems.

The next section establishes that steady-state solutions corresponding to harmonic excitations of the system (2.1), the excitations having varying frequencies and amplitudes, can be characterized by *one* function. In [1], it is referred to as the *nonlinear frequency response function*, and aims to extend such functions found in linear systems theory.

2.1 Nonlinear Response to Harmonic Excitation

Consider systems of the form (2.1), as well as Definition 2.2. The system is excited by inputs $\mathbf{w}(t)$ that are known to be solutions of

$$\dot{\mathbf{w}} = \mathbf{S}(\mathbf{w}), \qquad \mathbf{w} \in \mathbb{R}^n \tag{2.3}$$

with a locally Lipschitz right-hand side. The solution of (2.3) is some function $\mathbf{w}(t, w_0)$ with initial condition $\mathbf{w}(0, w_0) = w_0$. We need a boundedness assumption related to this solution;

Assumption 2.1. Every solution of (2.3) is defined and bounded on $\mathbb{R}$, and $\forall\ r > 0$ there exists $\rho > 0$ such that

$$|w_0| < r \qquad \Rightarrow \qquad |\mathbf{w}(t, w_0)| < \rho \qquad \forall\ t \in \mathbb{R}. \tag{2.4}$$

The above assumption is basically the same boundedness assumption as in Definition 2.2, applied to the system (2.3). To generate general harmonic inputs, we may for example choose $\mathbf{S}(\mathbf{w}) = \mathbf{S}\mathbf{w}$ where the matrix $\mathbf{S}$ fulfills

$$\mathbf{S} = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix}. \tag{2.5}$$

The solution to (2.5) will be $w_1 = \sin \omega t$, $w_2 = \cos \omega t$. Next, one needs to apply this system, acting as a "harmonic signal generator", as input to the general nonlinear convergent system. The following theorem is the result of doing just that.

Theorem 2.1. Let the system (2.1) be uniformly convergent and regular with the UBSS property for the class of continuous bounded inputs. Then, there exists a continuous function $\alpha : \mathbb{R}^3 \to \mathbb{R}^n$ such that for any harmonic excitation of the form $u(t) = a \sin \omega t$, the system has a unique periodic solution

$$\overline{x}_{a\omega}(t) = \Pi(a \sin \omega t, a \cos \omega t, \omega) \tag{2.6}$$

and this solution is UGAS.

Proof. Consider the system (2.3) generating harmonic signals. Consequently, the system (2.1) excited by $u = w_1 = a \sin \omega t$ can be treated as the system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, w_1) \tag{2.7}$$

excited by the solution of (2.3). The above system is regular and uniformly convergent with the UBSS property for the class of continuous bounded inputs. Further one can easily check that the boundedness assumption 2.2 is satisfied. Therefore, by the above theorem, there exists a unique continuous function such that the steady-state solution of system (2.7) equals $\overline{x}_w(t) = \Pi(w_1(t), w_2(t), \omega)$. □

Definition 2.3. The function $\Pi(w_1, w_2, \omega)$ is called the *state frequency response function*. If there in the system is an output operator $y = h(x)$ then the function $h(\Pi(w_1, w_2, \omega))$ is called the *output frequency response function*.

The following lemma is the result that is used to determine the set of partial differential equations that describe the state frequency response function.

Lemma 2.1. Under the conditions of Theorem 2.1, if there exists a continuous function $\Pi(w_1, w_2, \omega)$ differentiable in $w = [w_1 w_2]^T$ and satisfying

$$\frac{\partial \Pi}{\partial w}(w, \omega) S = F(\Pi(w, \omega), w_1), \qquad \forall\ w, \omega \in \mathbb{R}^2 \times \mathbb{R} \tag{2.8}$$

then this $\Pi(w_1, w_2, \omega)$ is the state frequency response function. Conversely, if the state frequency resposne function is differentiable in w, then it is a unique solution of (2.7).

The proof of this statement can be found in []. How straightforward it is to actually find this function

Π is system dependent. For some systems it is possible analytically. If not, one can try and find some approximative solution by numerical means. Further note that (2.8) will result in a partial differential equation for systems with only one state (which are quite a rare sight in control applications), or a set of coupled partial differential equations for system with multiple states. If the system itself is nonlinear, the equations will naturally also be nonlinear, and such systems rarely have analytical solutions.

The above couple of pages have been an excercise in assumptions, theorems, and lemmas. Indeed, they are a condensed version of quite a lot of research that can be found in the referred papers. To round this section off, a more textual description of what has just been presented will be attempted.

Recall that a convergent system is a general nonlinear system which exhibits two defining characteristics; firstly it has a unique steady-state response, and secondly this response is independent of the initial conditions. We wish to analyze the system behaviour subject to harmonic excitations. This is achieved by defining the exosystem (2.3), which generates harmonic signals that can be used as inputs to the general convergent system (2.1). Note that this does not change the convergence property since Assumption 2.1. holds. Next, Theorem 2.1. is presented establishing that the system response to these harmonic excitations may be characterized by only one function. This is where it gets slightly tricky. This function takes three arguments in general; w_1 and w_2 are the solutions of (2.3), and ω is the frequency which naturally is allowed to vary.

Continuing onwards, said function may be found by solving the system of partial differential equations given by (2.8). This solution is in terms of the (w_1, w_2, ω)-coordinates, however for simplicity assume that the frequency ω is constant satisfying $\omega > 0$. Then, the solution is given in terms of the w_1w_2-axes by $\Pi(w_1, w_2)$ where $w_1 = a\sin\omega t$ and $w_2 = a\cos\omega t$.

2.2 Nonlinear Frequency Reponse Functions

In addition, and with a solution to the above problem, functions may be defined which are basically nonlinear counterparts of the frequency response functions so commonly applied in linear control theory. They are referred to in [1] as *generalized sensitivity functions*, and aim to mimic the sensitivity functions which may be found in linear control theory. These will not be restated here, but [5] is an excellent textbook on this subject. To formulate the nonlinear version of these functions, we consider a closed-loop system like the one depicted in Figure 1.1, but with a nonlinear plant and/or controller. In state-space, this may be modelled as;

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{n}) \\ \boldsymbol{p} &= \boldsymbol{h}(\boldsymbol{x}). \end{aligned} \tag{2.9}$$

The error is defined as $\boldsymbol{e} = \boldsymbol{r} - \boldsymbol{p}$. By inspection, this system is convergent for the class of continuous bounded inputs $\boldsymbol{r}$ and $\boldsymbol{n}$. For inputs $n(t) = a\sin\omega t$ and $r(t) = 0$ the system has a unique steady-state periodic solution, which we may denote as $p_{a\omega}(t)$. Conversely, for the inputs $r(t) = a\sin\omega t$ the corresponding steady-state solution $e_{a\omega}(t)$ is also unique. Then, the generalized sensitivity function and the complementary generalized sensitivity function is defined as follows [1].

The functions

$$S(a,\omega) = \frac{||e_{a\omega}||_2}{||r_{a\omega}||_2}, \qquad T(a,\omega) = \frac{||p_{a\omega}||_2}{||n_{a\omega}||_2}, \tag{2.10}$$

are called, respectively, the generalized sensitivity function and the complementary generalized sensitivity function of the convergent system (2.9).

Note that, in light of the fact the superposition does not hold, the functions above only yield information on the system response to actual harmonic inputs. However, this information is usually very valuable, as in industrial control systems it is commonplace to evaluate the performance subject to harmonic disturbances. Additionally, many disturbances that are encountered in practice may be modelled as a harmonic

disturbance, or at the very least closely approximated as harmonic disturbances. For example, the engine vibrations experienced by an optical drive mounted in a car may be viewed as a low-frequent harmonic disturbance.

2.3 Lur'e Systems as Convergent Systems

An early nonlinear feedback control problem was formulated by A. I. Lur'e. Control systems of this form (Figure 2.1), hereafter termed *Lur'e systems*, have a forward LTI path, and a feedback path that contains a memoryless, though possibly time-varying, static nonlinearity. The linear part may be represented by the matrix quadruple $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ such that the linear transfer function in the forward loop is uniquely determined by;

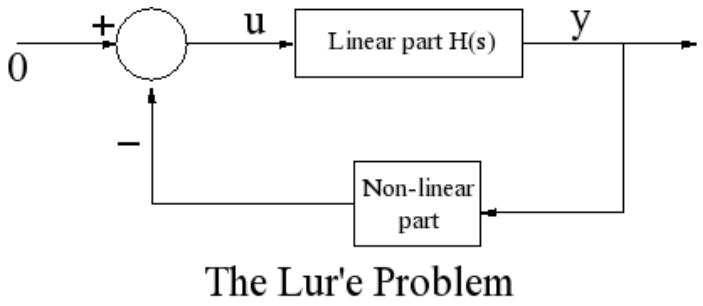

Figure 2.1: A general Lur'e system.

$$\mathbf{G}(j\omega) = \mathbf{C}(j\omega\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}. \tag{2.11}$$

The expression (2.11) should be easily recognized by anyone with some background in linear control theory. The nonlinear part is represented by a sector nonlinearity[1] $\mu(y)$.

The Lur'e systems may be written in state-space as:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\mu(\boldsymbol{y}) + \boldsymbol{F}\boldsymbol{u} \\ \boldsymbol{z} &= -\boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C_y}\boldsymbol{x} + \boldsymbol{D_y}\boldsymbol{u} \end{aligned} \tag{2.12}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, $y \in \mathbb{R}$ is the input vector, and $u \in \mathbb{R}$ is the output vector. The nonlinearity $\mu(\boldsymbol{y})$ is a scalar nonlinearity. The following assumptions must be made in connection with (2.4.1):

1. The matrix $\boldsymbol{A}$ is Hurwitz;
2. The nonlinearity $\mu(\boldsymbol{z})$ is odd and satisfies the slope restriction

$$0 \leq \frac{\mu(z_1) - \mu(z_2)}{z_1 - z_2} \leq \xi, \tag{2.13}$$

   $\forall$ z_1, z_2 and some constant $\xi > 0$;
3. The circle criterion holds;

$$\mathrm{Re}\Big[\boldsymbol{C}(j\omega\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}\Big] > -\frac{1}{\xi}, \qquad \forall\ \omega \in \mathbb{R}. \tag{2.14}$$

[1] This means we have $\frac{\mu(y}{y} \in [a, b]$, where $a < b$.

It is known that, under the above assumptions, the Lur'e system (2.4.1) is exponentially convergent for the class of bounded piecewise inputs. Therefore, for the harmonic input $a \sin \omega t$ this system has a unique periodic steady-state solution $x_{a\omega}(t)$ which corresponding output $y_{a\omega}(t)$ and period $\tau = 2\pi/\omega$.

This is a powerful result, because many practical control system applications may be written as a Lur'e system. In particular, any mass-spring-damper system is a Lur'e system, and any stable mass-spring-damper system, linear or nonlinear, is a convergent system. The applications of mass-spring-damper systems in control practice are virtually endless.

2.4 Case Study: Nonlinearly Controlled Optical Storage Drive

A Case Study (was) intended for this thesis, that of a control problem relating to an optical storage drive, on which the theory can be applied. The resulting model will fit the general class of Lur'e systems that has been outlined. The study is also presented in [2]. A technical study of optical drives in general can be found in [23].

2.4.1 Dynamics and Model

In optical storage units, like CD or DVD drives, the information is stored in a series of indentations in the surface of the disk, called *pits*. Between these pits are sections of flat surface, known as *lands*. The lens projects a beam onto the reflective surface of the disk, and reads this topology of pits and lands which is subsequently converted into binary data. The lens itself is placed on a sledge which is actuated and can move radially on the disk along a base frame. Addiotionally, the lens itself is actuated and can move within the sledge. This situation is depicted in Figure 2.2. There are two control problems of interest for this system; the *long-stroke* motion of the sledge relative to the disk, and the *short-stroke* motion of the lens relative to the sledge. It is the latter that is the control problem which will be presented here.

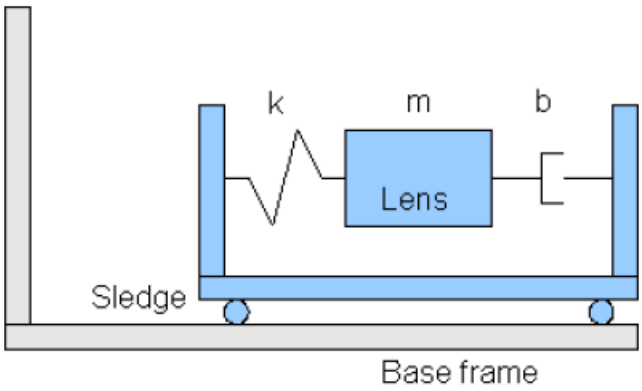

Figure 2.2: The lens mass inside the sledge, moving on a radial base frame.

The lens dynamics are modelled as a standard mass-spring-damper system with mass m, stifness k, and damping b. In addition the actuator dynamics are modelled as a lowpass filter in series with the lens dynamics, such that the (linear) combined lens and actuator dynamics may be represented by the transfer function

$$P(s) = \frac{\omega_a}{(ms^2 + bs + k)(s + \omega_a)}, \tag{2.15}$$

where ω_a is the break frequency of the lowpass filter. The controller is chosen as a linear PID type, with

corresponding transfer function

$$C(s) = \frac{k_p\omega_{lp}^2(s^2 + (\omega_d + \omega_i)s + \omega_d\omega_i)}{\omega_d(s^3 + 2\beta\omega_{lp}s^2 + \omega_{lp}^2 s)} = k_p\frac{1}{s}\frac{\omega_{lp}^2(s^2 + (\omega_d + \omega_i)s + \omega_d\omega_i)}{s^2 + 2\beta\omega_{lp} + \omega_{lp}^2}. \tag{2.16}$$

In (2.16) k_p is the controller gain, ω_i the breakpoint of the integral action, ω_d the breakpoint of the derivative action, ω_{lp} is the breakpoint of the lowpass, and β is a damping parameter. The control system that would result from the above plant and controller is depicted in Figure 1.1, where $\mathbf{x}$ is the state, and $\tilde{\boldsymbol{e}}$ is the measured error signal. The overall control problem is to drive this error to zero, and thus position the lens correctly to read the appropriate data from the disk. For some time in these control problems, the systems were based on linear models such as the one above. While this is a tempting approach due to the relative simplicity of the control problem, it does have its share of drawbacks. In practice, the most important drawback was found to be that of noise and shock suppression over a wide range of frequencies. While the linear system can be tuned to suppress noise very well in one limited frequency range, it typically performs much poorer outside of this range. This is a problem if the control system is frequently subjected to disturbances of very varying frequency. As optical drives became commonplace in cars, on public transport systems, and even designed to be carried around by people jogging, disturbances of varying frequency content also became more and more common. The movement experienced by the system, because of being mounted in car, or carried around in the park, is a disturbance of relatively low frequency. At the same time, scratch marks and smudge is easier to keep on the typical CD or DVD disk than off it, which will contribute to a typical high-frequency disturbance.

With a purely linear control system, a design choice would have to be made as to which of these disturbances to focus on. Clearly, it would be best overall to focus on suppressing the low-frequency content, as it is most common and mostly unavoidable. At the same time, the user might find that his car-mounted stereo performed much poorer in playing his slightly scratched disk than the tabletop stereo at home. The best solution would be to come up with a way to tackle both of the mentioned disturbances at the same time.

A possible solution to this was presented in the paper [2], wherein a *variable-gain* control strategy is suggested. The basic idea is to vary the controller gain k_p such that it ramps up if the measured error (subject to the noisy disturbances) is outside of certain bounds, aiding the control system in repressing said disturbances. A slightly modified version of the control system above is shown in Figure 2.3, where a variable-gain block has been added to the feedback path between the application of the disturbances and the controller. The purpose of this block is to improve the overall performance of the control system. In state-space, the system depicted in Figure 2.3 may be written as:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\gamma(\tilde{\boldsymbol{e}}) + \boldsymbol{B}(\boldsymbol{r} + \boldsymbol{n}), \\ \boldsymbol{p} &= \boldsymbol{C}\boldsymbol{x}, \\ \tilde{\boldsymbol{e}} &= \boldsymbol{r} + \boldsymbol{n} - \boldsymbol{p}. \end{aligned} \tag{2.17}$$

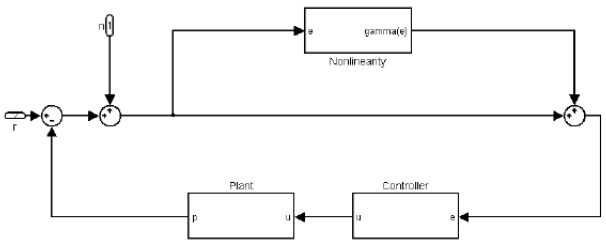

Figure 2.3: The nonlinear control system under consideration.

With $\boldsymbol{r}, \boldsymbol{n} \in \mathbb{R}$ being the (radial) error and the added noise, respectively, $\boldsymbol{x} \in \mathbb{R}^6$ is the state vector, and $\tilde{\boldsymbol{e}}$ is the measured error signal. In the state vector $\boldsymbol{x}$, the variables x_1 through x_6 corresponding to the following. x_1, x_2, x_3 are the derivative, proportional, and integral action of the controller, respectively, x_4 is the force acting on the lens mass, x_5 is the radial position, and x_6 is the radial velocity of the lens mass. The matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are constant and satisfy;

$$\begin{aligned}
\boldsymbol{A} &= \begin{bmatrix} -2\beta\omega_{lp} & -\omega_{lp}^2 & 0 & 0 & -k_p\omega_{lp}^2 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{\omega_a}{\omega_d} & \omega_a(1+\frac{\omega_i}{\omega_d}) & \frac{\omega_a}{\omega_i} & -\omega_a & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{1}{m} & -\frac{k}{m} & -\frac{b}{m} \end{bmatrix}, \\
\boldsymbol{B} &= \begin{bmatrix} k_p\omega_{lp}^2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T, \\
\boldsymbol{C} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.
\end{aligned} \tag{2.18}$$

While the nonlinearity γ is a piecewise linear nonlinearity signifying the wanted change in the controller gain. We have $\gamma(\tilde{e}) = \alpha(\tilde{e} - sgn(\tilde{e})\delta)$ for $|\tilde{e}| > \delta$ and $\gamma(\tilde{e}) = 0$ for $|\tilde{e}| \leq \delta$.

It may easily be shown that the system (2.17) is a convergent Lur'e system of the form (). The system matrix $\boldsymbol{A}$ is Hurwitz due to the stabilizing control design. The nonlinearity clearly satisfies the incremental sector condition for $\xi = \alpha$, and the circle criterion holds for certain values of α. Hence, we may use the presented theory on this system to arrive at the function which describes the steady-state solution.

2.5 Derivation of the PDE Models

It has been hinted earlier in the report that the result of the process of formulating nonlinear frequency response functions is a coupled nonlinear system of partial differential equations. In this section this process will be briefly summarized, and it will be outlined how to derive these equations. The section is concluded with deriving said equations for the case of the optical storage drive system.

Let us first consider a very simple model example.

$$\dot{x} = -x + a\sin\omega t. \tag{2.19}$$

The above is a stable system, harmonically excited with amplitude a and frequency ω. Obviously, this system in linear, and the laplace transformation may be applied to analyze its performance. However, this system is convergent and the theory of the preceding sections is applicable. Since the sinusoid term is the input, we may look at it as the output $u = v_1 = a\sin\omega t$, and we then have the following;

There exists a unique function $x_v(t) = \Pi(v_1, v_2)$ which is the steady-state solution of (2.19), when $u = v_1$ and v_1, v_2 are the outputs generated by the "harmonic signal generator" (2.3).

Insertion of this function into the govering ODE model (recall that $a\sin\omega t = av_1$) directly gives the following *partial* differential equation:

$$\frac{\partial\Pi}{\partial v_1}\omega v_2 - \frac{\partial\Pi}{\partial v_2}\omega v_1 = -\Pi + av_1. \tag{2.20}$$

The above equation now describes the *steady-state* solution of the original ODE model, for a particular choice of a and ω.

Consider the case in which we assign some constant value to ω. In this case, we have a threedimensional solution to the partial differential equation, which, given the theory present, is smooth and unique. The *solution* is further defined in the whole v_1, v_2 plane, so if we could solve this equation for a particular value of the amplitude, we would also necessarily solve the equation for all other values between 0 and a. This

is the primary result a numerical method like the finite element method might be able to achieve - to solve for many different values of the amplitude in a single, smooth solution. If this is possible then it might very well be more effective, computationally, than other methods in which both a and ω must be allowed to vary. If a solution to the partial differential equation can be found, then we may proceed to construct frequency response functions by resolving the equation for different values of ω.

For the case study system, we may simply insert the solution into the state space model (2.17), and obtain the following expression.

$$\frac{\partial \mathbf{\Pi}}{\partial v_1}\omega v_2 - \frac{\partial \mathbf{\Pi}}{\partial v_2}\omega v_1 = -\boldsymbol{A}\mathbf{\Pi} + \boldsymbol{B}\gamma)(\boldsymbol{C}\mathbf{\Pi}) + \boldsymbol{B}v_1. \tag{2.21}$$

The above descripes a *nonlinear coupled system of partial differential equations*. Such systems are usually very nontrivial to solve, and the solution methods may be highly system-dependent. While there are several possible methods that may be studied, in this thesis the finite element method will be applied to this system.

We will also here comment briefly on a couple of particulars of this system. Firstly note that there are no additional conditions that come with the model (2.21), such as boundary conditions, or other conditions that are required in order to establish a unique solution. Few partial differential equations (at least in the authors space of knowledge in this area) have this property. We will note one more aspect, which may be viewed as a connection between the ODE and PDE solutions.

As we have seen, the PDE is obtained by inserting the known solution into the ODE system. It is possible to solve the ODE system, and map the solution over to the v_1, v_2-space in which the PDE solution is defined. To do this, we note that we have $v_1 = \sin \omega t$ and $v_2 = \cos \omega t$ as per the definition of the harmonic signal generator. The unique steady-state PDE solution is defined for some period τ of the ODE solution, and hence we may solve the ODEs, extract the signal values for one period τ, and map this data to a *circle*, of radius a, in the v_1, v_2-system using the following transformation;

$$\begin{aligned} v_1 &= a \sin \omega t \\ v_2 &= a \cos \omega t \\ \Pi &= x_{a\omega}(t). \end{aligned} \tag{2.22}$$

The above is defined $\forall\ t$ in *one* full period of the ODE solution $x_{a\omega}(t)$, and is the solution to the partial differetial equation for a single circle in the v_1, v_2-plane. This leads to the following conclusion.

The domain over which to solve the systems of partial differential equations in the v_1, v_2, Π-space should be a circle of radius a, where a is the maximum considered amplitude of the input signal v_1.

The above will allow us to search for a solution to the problem for all amplitudes between 0 and some maximum a, as long as we keep ω fixed as has been outlined.

Finally, we will briefly discuss the nonlinearity. It was found that most nonlinear solution methods for finite element applications relied on a *differentiable* nonlinearity, both in order to formulate the method, and to be able to prove under what condtions the iterations can be expected to converge. The nonlinearity γ is clearly not differentiable since the signum function is not, and it was decided to approximate this term with a continuous function. As we will see later, this may not have been the best choice of action in this case. At any rate, in this thesis the nonlinearity is approximated with a cubic function, that is $\gamma(\Pi) \approx \Pi^3$. This allows for the finite element model equations to become differentiable. The approximation is reasonable $\forall$ $a < 1$, which points to the fact that we may consider the unit circle as our domain of interest for the time being. The true output of the nonlinear block is shown in Figure 2.4.

It should further be noted that only first-order derivatives will appear in the partial differential equations.

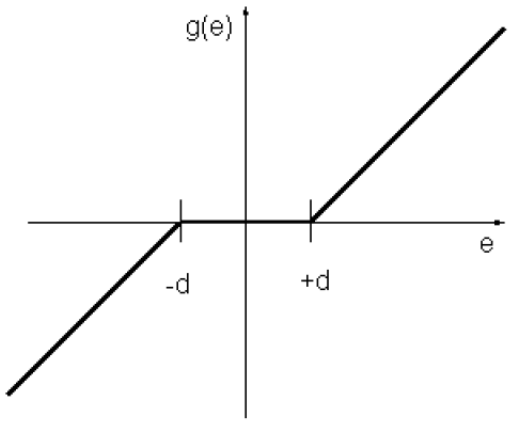

Figure 2.4: The nonlinear block gamma.

Chapter 3

Finite Element Application

This chapter firstly aims to present the Galerkin Finite Element Method in general and certain theoretical and practical details that are needed to solve systems such as the one shown in (2.21). The approach taken here is fairly pragmatic and presents only the most important subjects, those that need to be covered in order to fully understand the implementation. Secondly, this will be used to apply the finite element method to arrive at a finite element model of (2.21). Before we start off, we will briefly summarize the main points of the method;

- Discretize the domain
- Obtain a model which approximates the problem over this domain through the use of piecewise interpolation on each discrete element of the domain
- Assemble the equations to obtain global equilibrium equations for the total domain.

Certain details in each of the above steps will be outlined. For more information, consider one of the several excellent finite element resources in the references, for instance [7], [12], [14], [15], and others.

3.1 The Galerkin Finite Element Method

This section presents all the required background theory for the method of choice; the Galerkin Finite Element Method. We begin with discussing how to, in general, obtain approximative solutions to partial differential equations.

3.1.1 The residual and integral test

It is an advantage to cast the needed formulations and ideas in a general way. To this end, consider the following expression.

$$\boldsymbol{R}(\boldsymbol{u}) = \mathbf{0}. \tag{3.1}$$

Here, $\mathbf{u} \in \mathbb{R}^n$ is a vector of unknowns, and $\boldsymbol{R}(\boldsymbol{u})$ is a matrix function of these unknowns. Expression (3.1) will be thought of to represent any partial differential equation, be it linear or nonlinear, single or sets of equations, and including or not including time as an independent variable. The equation is arranged such that the righthand side becomes identically zero, and the lefthand side $\boldsymbol{R}(\boldsymbol{u})$ will be referred to as the *residual* of the equation. Obviously, if $\boldsymbol{u}$ is the exact solution, the residual is zero. However, we will be searching for approximate solutions and such we may (and most likely will) encounter situations in which the residual is not equal to zero. Checking whether the residual is zero at any given $\boldsymbol{u}$ does not provide anything useful we can use to talk about approximate solutions, it is either zero or not. Hence the first question to be addressed is how one should measure whether an approximate solution, for which the residual is not equal to zero, is a good solution. In short, we are looking for a quality measure on the approximate solution $\tilde{\boldsymbol{u}}$ to (3.1).

One possible choice of such a measure is to integrate the residual over the problem domain (such as the

length in one dimension, the area in two dimensions, and so on). The problem domain will be denoted Ω. Hence, we first construct the integral

$$\int_{\Omega} \boldsymbol{R}(\boldsymbol{u}) \, \mathrm{d}\Omega. \tag{3.2}$$

Think of this as a test; if the residual is identically zero, then the integral will also come out zero. There is a problem however, (3.2) may be zero even if the residual is not. If this was to be used to attempt to prove that the residual corresponds to an actual solution, it would be a flawed test. For example, consider the simple sinusoid depicted in Figure 3.3. Clearly the integral over the length of the graph is zero, however the residual itself may be large. So, some additional thought is needed. One can think of (3.2) as being "blind" to the actual shape of the residual.

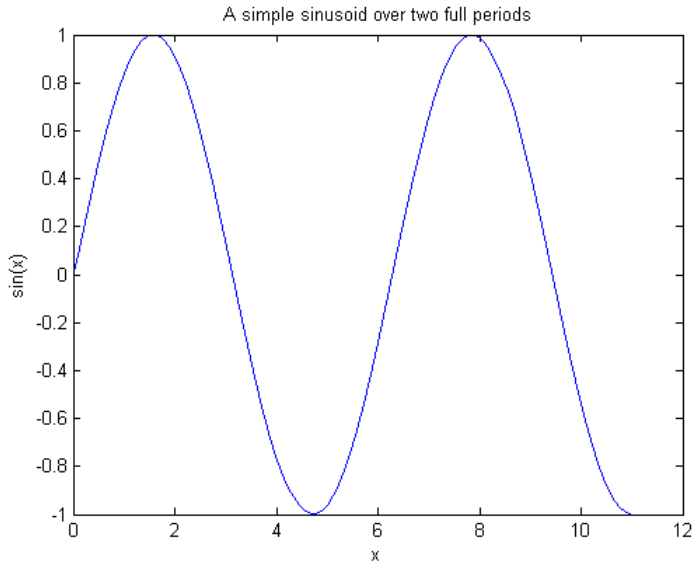

Figure 3.1: A simple sinusoid.

A remedy that addresses this is to use a "window" function, or more commonly known a test function. Hence, let $\mathbf{w} \in \mathbb{R}^{\mathbf{n}}$ be a vector of such functions, and note that they are for the time being taken to be completely arbitrary. Now, instead of (3.1), construct the following integral.

$$\int_{\Omega} \boldsymbol{w}^T \boldsymbol{R}(\boldsymbol{u}) \, \mathrm{d}\Omega \tag{3.3}$$

The above correctly indicates that the residual alone does not correspond to the exact solution. Equation (3.3) is known as a *weighted residual statement*. Approximation methods that start from a weighted residual statement are known as weighted residual methods.

There is still a small issue with (3.3) however. In order to make sure there are no "bumps" in the residual it would need to be evaluated for an infinite number of functions $\boldsymbol{w}$. Just like evaluation the original residual at every possible $\boldsymbol{u}$, this job will still take an infinite time. An analogy of what (3.3) is actually trying to accomplish is that it is like trying to push a balloon into a box. We may use the fingers of one hand to press down on the balloon such that the top of the balloon is held exactly at the top of the box - this is like the residual being zero with the fingers being test functions. However, evidently the balloon will protrude a little between the fingers and a lot everywhere else. We may then increase the number of test functions - fingers - by using our other hand as well. Then we may recruit our friends and relatives to come assist in pushing down the balloon, and do a better and better job so the balloon in the end does not protrude very much.

With the above analogy we may begin to see how a trial-and-test approximation method may be formulated. Selecting a finite number of suitable functions w_j, $j \in 1, 2, ..., N$, we should be able to keep the residual small, and by increasing the number of test functions the error should reduce. In addition, for each suitable test function we will make the integral (3.3) vanish, which provides the means of calculating N coefficients from these N equations. Then, the domain Ω may also be divided into N separate parts, a solution to (3.3) sought for each with some corresponding test function w_j. The total solution over the entire domain can then be imagined as the sum of each "sub-solution". In fact, this is what the Finite Element Method is all about at the core, and the next few sections will explore these ideas and why they work. The first step is the actual division of the whole domain Ω into a number of subdomains.

3.1.2 Discretising the domain

First, we clarify what is meant by the domain Ω. To this end, return to the expression (3.1). In this text, the domain will always be understood as where we seek to solve the equation, for example within a certain area of $\mathbb{R}^2$ for a two-dimensional problem. While the equation may very well be defined for the entirety of $\mathbb{R}^2$, the domain will be understood as some simply connected, closed subset or subspace of the total definition space for which the equation remains valid. By *simply connected* we mean that the boundaries of this region do not in any way overlap, and by *closed* we mean that the boundary defines some complete shape such that there are no gaps in it. For example, the circle depicted in Figure **??** is a simply connected, closed subspace of $\mathbb{R}^2$.

Next, we need to divide the domain into a finite number of subdomains. These subdomains should be as easy as possible, and the concept of tiling arbitrary domains into a set of triangles is quite old. Hence, the domain Ω will be approximated as a collection of triangles, known as a *triangulation*. As the number of triangles increase, the geometry of this "collection" of triangles will more and more closely resemble that of the actual circle.

The vertices of the triangulation are the *nodes*, while the segments connecting the nodes are *edges*. Evidently, the triangles themselves are the *finite elements*.

For the purposes of this thesis, we will consider a discretization of the unit circle in the v_1, v_2 space using linear triangular elements. Such a discretization is shown in Figure **??**.

3.1.3 Test and trial functions: basis functions on triangulations

It is time to discuss the test and trial functions, and how these may be chosen. From this point, the discussion will be limited to two-dimensional problems, as there is no need to consider higher dimensional problems for the purpose of this thesis. The discussion will be further limited to triangular elements.

The basic idea here is to construct some interpolation over each individual element, then combine these into a total system solution. It assumed that the interpolation may be written as an expression similar to;

$$\Pi = \sum_{i=1}^{n} N_i(v_1, v_2) u_i. \tag{3.4}$$

Where N_i are certain polynomial functions, and u_i are the nodal unknowns we are trying to calculate. It is worth noting that this corresponds to a *displacement-based* finite element method, in that we are only solving for the displacements of u when raised out of the v_1, v_2-plane. These functions are often called *trial* functions.

First, recall that the *test* function is an arbitrary function w used to weigh the residual before integration. The *trial* function is a chosen function which will be assumed to approximate the solution to the partial differential equation within one finite element.

The idea that will be followed here is quite an old one, namely the concept of piecewise linear functions

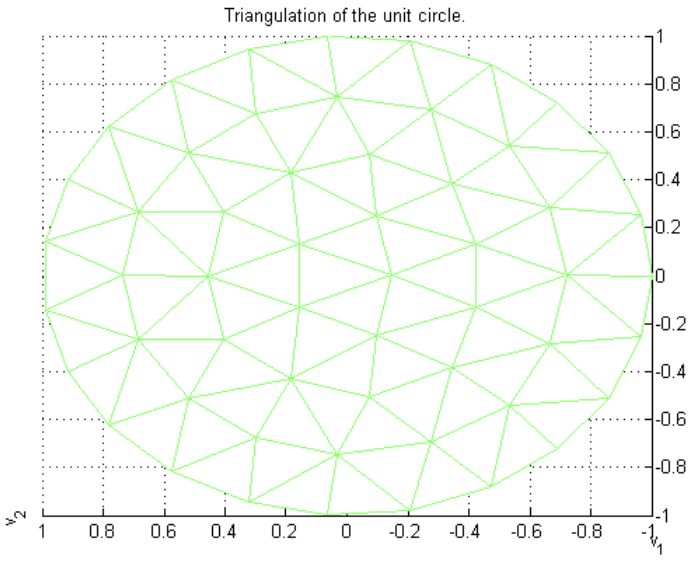

Figure 3.2: A triangulation of the unit circle.

defined over these tilings of the domain Ω. Now some additional points need to be made about the actual form of the partial differential equation contained in the residual, specifically what order derivatives we are dealing with. For the purpose of this thesis only first-order derivatives of the unknown functions need to be considered. At any rate, for equations containing second-order derivatives it is common practice to carry out integration by parts to reduce the second-order terms to first-order terms, hence still allowing for linear interpolants to be used. It the following it is thus assumed that only first-order derivatives are encountered.

With this in hand, interpolation on the triangulation will be treated as a linear combination of functions which geometrically will resemble "tents". Each individual tent is formed by grabbing one of the nodes of a triangle (say node j), and raising it out of the plane of the triangulation (traditionally to unit height). The tent "canvas" is stretched over the edges that connect at node j, and are clamped down by the ring of edges that surround node j. One such particular basis function tent is shown in Figure **??**, along with some points for the integration rules inside the triangle. For those who do not like tents, the term *hat function* may be preferable. The basis function connected to node j will be denoted as N_j.

All the triangles which are connected to node j are said to *support* the function N_j, which is another way of saying that the function N_j is nonzero in these triangles; evidently it is defined to be zero everywhere else. It now remains to find the expression that defines such a function N_j at any point within its support. That would mean writing an expression for each triangle separately. This is certainly one way of doing it, and since the equations will be similar for any triangle it is possible to do so. There is, however, a better way. The alternative viewpoint is to define one single triangle to serve as a *master* element. Over this triangle we can express all nonzero pieces of all basis functions. The triangle most commonly chosen for this is the unit right isosceles triangle, one of which is depicted in FIGURE. Referring to the same figure, there are only three such functions; the three basis functions associated with the nodes at the corners of the element. All the other basis functions in the mesh are identically zero for that particular element. Thus, all that is left is the task of writing down the basis functions on a single triangle. Before this, a very important requirement will be stated.

It is imperative for the convergence of the isoparamteric finite element models that each shape function takes on the value of unity (1) at its home node, and is zero everywhere else in the mesh.

The reasons as to why are not covered but may be found in most introductory textbooks.

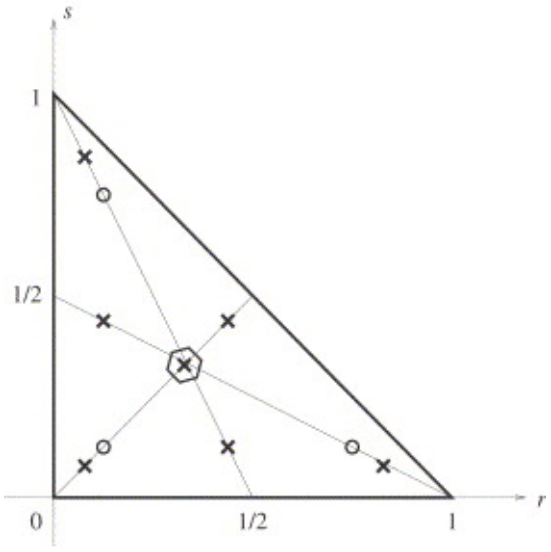

Figure 3.3: The master element, along with some integration points.

Each of the three basis functions is zero along one edge of the triangle. Using the standard unit right isosceles triangle as the master triangle, one can spot that the basis functions associated with nodes 2 and 3 are simply

$$\begin{aligned} N_2(\xi,\eta) &= \xi, \\ N_3(\xi,\eta) &= \eta. \end{aligned} \tag{3.5}$$

As is easily verified, N_2 is zero along the edge 1-3 and assumes the value +1 at node 2; analogous properties hold for N_3. If N_1 should be equal to +1 at node 1, it must be written as

$$N_1 = 1 - \xi - \eta. \tag{3.6}$$

Clearly N_1 vanishes at the edge opposite node 1. These three functions now also satisfy an important interpolation property known as the *Kronecker delta property*. In English, this means that the degree of freedom at each node of the triangle is equal to the value of the interpolated function at the node. Therefore, we make the observation that data sitting at the nodes of the triangle are *naturally interpolated*. One particularly useful quantity that one can interpolate on the standard triangle are the Cartesian coordinates of the nodes in the physical space (x,y);

$$\begin{aligned} x &= \sum_{i=1}^{3} N_i(\xi,\eta) x_i, \\ y &= \sum_{i=1}^{3} N_i(\xi,\eta) y_i, \end{aligned} \tag{3.7}$$

with $[x_i,y_i]$ being the Cartesian coordinates of the three points. The result of this interpolation is now a fixed point $\boldsymbol{p} = [x, y]^T$ in the Cartesian space, and equation (3.7) is a mapping from the pair (ξ,η) to the point (x,y). Substituting for the basis functions, it may be written explicitly as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}. \tag{3.8}$$

Inverting the above to arrive at ξ and η, which could then be substituted into the shape functions, looks tempting but should be resisted. The reason is that quadrature rules for numerical integration are readily

available for the standard triangle, but is much harder on general triangles. This will become especially clear with the applications covered later.

However, since (3.8) is an invertible map[1] from the standard triangle to any given triangle in the Cartesian coordinates, we now have an approach to *evaluating basis functions on a general triangle*. Given a point $(\bar{x}, \bar{y})$ in the Cartesian space, and within the bounds of a triangle, we can use the inverse of the map (3.8) to obtain a pair $(\bar{\xi},\bar{\eta})$ in the standard triangle. We may then evalute $N_i(\bar{\xi}, \bar{\eta})$. This might seem awkward, but normally we wish to evaluate the basis functions in order to perform a numerical integration at a particular *quadrature point* within the standard triangle. Then, $(\bar{\xi},\bar{\eta})$ would be known, and $(\bar{x},\bar{y})$ would be unknown. However since values at the nodes are naturally interpolated we have

$$N_i(\bar{\xi}, \bar{\eta}) = N_i(\bar{x}, \bar{y}). \tag{3.9}$$

Thus if we can calculate the value of N_i at any given node in the standard triangle, we automatically get the value of N_i at the corresponding node in the Cartesian space.

3.1.4 Derivatives of the basis functions

We have already established how to evaluate the value of a basis function at any point within the element. What remains is to derive the expressions necessary to calculate the derivatives of these functions. While this section only does so for the linear triangle, the results are much more general, and the same ideas and formulas may be applied to any element.

Recall that the basis functions corresponding to the linear triangle are

$$N_1 = 1 - \xi - \eta, \qquad N_2 = \xi, \qquad N_3 = \eta. \tag{3.10}$$

Deriving $\partial N_i/\partial x$ and $\partial N_i/\partial y$ is easy enough using the chain rule of differentiation. We get:

$$\begin{aligned} \frac{\partial N_i}{\partial x} &= \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial N_i}{\partial \eta}\frac{\partial \eta}{\partial x} \\ \frac{\partial N_i}{\partial y} &= \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial N_i}{\partial \eta}\frac{\partial \eta}{\partial y} \end{aligned} \tag{3.11}$$

The above may, after some shuffling, be arranged into the following matrix equation.

$$\begin{bmatrix} \frac{\partial N_i}{\partial \xi} & \frac{\partial N_i}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_i}{\partial x} & \frac{\partial N_i}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{3.12}$$

The 2×2 matrix in the above is the *Jacobian* matrix of the mapping $x = x(\xi, \eta)$, $y = y(\xi, \eta)$ of (3.8), denoted as $\boldsymbol{J}$. The elements of $\boldsymbol{J}$ are directly available from (3.8), but it is better to use (3.7) instead. Hence, inserting we have:

$$\boldsymbol{J} = \begin{bmatrix} \sum_{i=1}^{3} \frac{\partial N_i}{\partial \xi} x_i & \sum_{i=1}^{3} \frac{\partial N_i}{\partial \eta} x_i \\ \sum_{i=1}^{3} \frac{\partial N_i}{\partial \xi} y_i & \sum_{i=1}^{3} \frac{\partial N_i}{\partial \eta} y_i \end{bmatrix}. \tag{3.13}$$

For implementation purposes the Jacobian will be expressed as the product $\boldsymbol{J} = \boldsymbol{x}^T \boldsymbol{N}_{\boldsymbol{\Delta}}$ where $\boldsymbol{x}$ collects in each row a coordinate pair (x_i, y_i) of the nodes and $\boldsymbol{N}_{\boldsymbol{\Delta}}$ collects in each row the derivatives of the basis functions with respect to the (ξ, η)-coordinates.

It is further noted that, in the case of the linear triangular element, the Jacobian will be a constant matrix. This is always the case for elements which have straight sides.

[1] At least as long as the triangle does not have all its corners along a single straight line.

3.1.5 Numerical Integration

Here the procedure for integrating functions numerically over the standard triangle will be formulated. We begin by highlighting the role of the Jacobian matrix for this purpose. Consider the following mapping $(\xi,\eta)\rightarrow(x,y)$:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{bmatrix}. \tag{3.14}$$

Now, define the two vectors $[\mathrm{d}\xi, 0]^T$ and $[0, \mathrm{d}\eta]^T$ spanning a parallellogram of area $\mathrm{d}\xi\mathrm{d}\eta$. These are mapped by (3.14) to the vectors

$$\begin{bmatrix} \mathrm{d}\xi \\ 0 \end{bmatrix} \rightarrow \mathrm{d}\xi \begin{bmatrix} \frac{\partial x}{\partial \xi} \\ \frac{\partial y}{\partial \xi} \end{bmatrix}, \qquad \begin{bmatrix} 0 \\ \mathrm{d}\eta \end{bmatrix} \rightarrow \mathrm{d}\eta \begin{bmatrix} \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \eta} \end{bmatrix}. \tag{3.15}$$

Here the square brackets hold the components in the standard Cartesian basis, note that these vectors are *tangent* to the coordinate curves, which consist of the points in the physical space (x, y) that are maps of the curves $\xi = \text{const}$ and $\eta = \text{const}$. The area of this new "hatched" parallellogram is

$$\mathrm{d}\xi\mathrm{d}\eta \begin{bmatrix} \frac{\partial x}{\partial \xi} \\ \frac{\partial y}{\partial \xi} \end{bmatrix} \times \begin{bmatrix} \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \eta} \end{bmatrix}. \tag{3.16}$$

When the above is compared to the expression for the Jacobian matrix in (3.12), it is seen that the two vectors in the cross product are the columns of this matrix. But the cross product of the columns is just the determinant of the matrix. Hence, the mapping (3.14) maps areas as

$$\mathrm{d}\xi\mathrm{d}\eta \rightarrow \mathrm{d}\xi\mathrm{d}\eta \ \det\big[\boldsymbol{J}\big]. \tag{3.17}$$

As a direct consequence of the above, the following expression holds true for the change of coordinates in integrals.

$$\int_{\Omega_{[x,y]}} f(x,y) \ \mathrm{d}x\mathrm{d}y = \int_{\Omega_{[\xi,\eta]}} f(\xi,\eta) \ \det\big[\boldsymbol{J}\big] \ \mathrm{d}\xi\mathrm{d}\eta \tag{3.18}$$

The above equality means that it is possible to determine the value of the integral of f over the element in the global (x,y)-system by integrating f over the master element in the local (ξ,η)-system and multiplying through by the determinant of the Jacobian matrix.

Now, depending on the nature of the function f, the integral in (3.18) may or may not be easy to evaluate. This is one of the reasons one normally chooses to perform all integrations in the finite element method numerically. It is also considered standard within the finite element formulation to perform integrations numerically; the process can be largely automated, easily modified to fit different problems, and offers a high degree of reliability overall.

Hence, approximate the left-hand side expression of (3.18) as follows.

$$\int_{\Omega_{[x,y]}} f(x,y) \ \mathrm{d}x\mathrm{d}y = \sum_{k=1}^{M} f(\xi_k,\eta_k) \ \det\big[\boldsymbol{J}\big] W_k. \tag{3.19}$$

Here, M is the number of integration pairs (ξ_k, η_k) within the master element, and W_k is the weight corresponding to integration point k. Note that we retain the equality in (3.19) since it is generally possible to obtain the exact (analytical) value of the integrand even if it is evaluated numerically. How (and why) to accomplish this is considered next.

It seems natural to try achieving as high an accuracy as possible. One integration rule in which the points and weights are optimized is *Gauss quadrature* []. Another issue is of what *order* the integration should be. It can be shown that, for a Gauss rule, a polynomial of order $2n-1$ is integrated exactly with n function expressions. Tbe set of points and weights that accomplish this is called an integration rule of order n. Choosing this order is important in practice because, first, the cost of analysis increases when a

higher order integration is employed, and second, using a different integration order can affect the results by a large amount. Specifically, if the order of integration is too low, the coefficients may be evaluated very inaccurately - and may even cause the finite element method to diverge. For this reason, it is desirable that full order of integration (giving analytical values of the integrand) is used if possible.

The points and weights to be used in these rules may be found in tables in most textbooks that cover the subject, for example in [12]. The points for the one-point, three-point, and six-point rules may be seen as the cross within a polygon, the circles, and the crosses in Figure **??**, respectively.

3.1.6 Finite Element Nonlinear Analysis

So far, only linear analysis has been mentioned. We have seen the construction of the finite element equilibrium equations of the form

$$\boldsymbol{K}\boldsymbol{U} = \boldsymbol{F}, \tag{3.20}$$

and assumed that the relationship between the response $\boldsymbol{U}$ and the applied force $\boldsymbol{F}$ is entirely linear. That is; if the force is increased to $\alpha\boldsymbol{F}$, for constant α, the corresponding system response is $\alpha\boldsymbol{U}$. Now, we will drop this assumption and discuss how to perform a *nonlinear* analysis within the finite element context.

Evidently, a nonlinear finite element model stems from a nonlinear governing partial differential equation. There are no extras involved in setting up the finite element model in such cases; the method of weighted residuals can be used in the same way as with linear equations.

The basic problem in a general nonlinear analysis is to find the state of equilibrium of a system corresponding to the applied forces. These equilibrium conditions can, returning to the same line of thought that was used to describe the residual of the governing equations earlier, be expressed as:

$$\boldsymbol{R} - \boldsymbol{F} = \boldsymbol{0}. \tag{3.21}$$

Here both $\boldsymbol{K}$ and $\boldsymbol{F}$ may be functions of the unknowns $\boldsymbol{U}$. We will think of $\boldsymbol{F}$ as the *internal* variables (most often describing forces or dynamics) of the system, while $\boldsymbol{R}$ represents the *external* variables; in other words the applied forces. The reason for this "'sudden change"' in notation will become apparent.

There are many types of nonlinearities, and for the purposes of this thesis only one will be considered, the *path-dependent* (material) type. With this type, it is necessary to solve the equilibrium equations (3.21) for the entire "'range"' of interest. By range we mean some collection of discrete steps the system need to go through before arriving at an (acceptable) solution. It is always with path-dependent nonlinearities necessary to perform some step-by-step incremental procedure, because the solution history affects the current system equilibrium.

The basic approach of such incremental solutions is to assume that the solution is *known* at some discrete iteration i within the procedure, and that the solution for iteration $i+1$ is required. Within each iteration, some form of (3.21) is solved. The exact setup of the equations depend on the employed method. The most widely used iterative methods in finite element analysis is based on the classical *Newton-Raphson* technique [], which is formally derived in Appendix APP. Here we restate the needed equations. It is assumed that the force term $\boldsymbol{R}$ is independent of the solution $\boldsymbol{U}$, and solve the following for $i = 1, 2, 3, ...$:

$$\begin{aligned} \Delta\boldsymbol{R}^{i-1} &= \boldsymbol{R} - \boldsymbol{F}^{i-1} \\ \boldsymbol{K}^{i-1}\Delta\boldsymbol{U}^{i} &= \Delta\boldsymbol{R}^{i-1} \\ \boldsymbol{U}^{i} &= \boldsymbol{U}^{i-1} + \Delta\boldsymbol{U}^{i} \end{aligned} \tag{3.22}$$

Where $\boldsymbol{K}$ is the tangent stiffness matrix, and we assume given $\boldsymbol{F}^0$ and $\boldsymbol{U}^0$ to be applied in the first iteration. In essence, (3.22) is obtained by linearizing the response of the finite element system about the conditions at iteration $i-1$. In each iteration an out-of-balance force vector is calculated (the first equation), which yields an increment $\Delta\boldsymbol{U}^i$ of the displacements in the second equation. We continue until the out-of-balance force

vector $\Delta \boldsymbol{R}^{i-1}$ or the displacement increments $\Delta \boldsymbol{U}^i$ are sufficiently small. Criteria for doing this break-off are also discussed in Appendix APP.

The Newton-Raphson iteration is so widely used in finite element analysis it represents the primary solution scheme for nonlinear finite element equations. Therefore, before tackling the details in the calculation of the different terms in (3.22), we give two major properties of the Newton-Raphson method that are of interest. Denote $\boldsymbol{U}^*$ as a converged solution.

Property 3.1. If the tangent stiffness matrix $\boldsymbol{K}^{i-1}$ is nonsingular, if the finite element system $\boldsymbol{R} - \boldsymbol{F}$ and its first derivatives with respect to $\boldsymbol{U}^*$ are continuous in a neighborhood of $\boldsymbol{U}$, and if $\boldsymbol{U}^{i-1}$ lies in that neighborhood, then $\boldsymbol{U}^i$ will be closer to to $\boldsymbol{U}^*$ than $\boldsymbol{U}^{i-1}$ and the sequence of iterative solutions generated by the algorithm converges to $\boldsymbol{U}^*$.

Property 3.2. If the tangent stiffness matrix also satisfies

$$||\boldsymbol{K}|_{\boldsymbol{U}_1} - \boldsymbol{K}|_{\boldsymbol{U}_2}|| \leq L||\boldsymbol{U}_1 - \boldsymbol{U}_2||, \tag{3.23}$$

$\forall\ \boldsymbol{U}_1, \boldsymbol{U}_2$ in the neighborhood of $\boldsymbol{U}^*$ and $L > 0$, then the convergence is quadratic. This condition may also be recognized as Lipschitz continuity.

The practical consequence of these two properties is that if the current solution iterate is sufficiently close to the solution $\boldsymbol{U}^*$ and if the tangent stiffness matrix does not change abruptly, we can expect rapid (quadratic) convergence. The assumption is of course that the *exact* tangent stiffness matrix is used in the iteration. On the other hand, if this matrix is not exact and/or changes abruptly, or the current solution iterate is not sufficiently close to the solution, then the iteration may diverge. In a finite element program, the exact tangent stiffness matrix should *always* be used if possible, and if convergence difficulties are still encountered one should firstly decrease the magnitude of the step between iterations, secondly investigate whether the initial guess or iterated solution where divergence occurs is sufficiently close to the actual solution. These topics will also be briefly encountered in later chapters.

Now, it is time to discuss how to calculate the various terms of (3.22) so as to fulfill the requirements for convergence of the nonlinear iteration. An important point is that the correct calculation of $\boldsymbol{F}^{i-1}$ from $\boldsymbol{U}^{i-1}$ is crucial. Any errors in this calculation will, in general, result in an incorrect response prediction. It is also worth to note that errors in this calculation may not cause the method to diverge; in fact it might instead cause the method to converge to some other, slightly different solution, especially if the errors are quite small. This can understandably lead to much confusion as to why the correct solution is not reached. The correct evaluation of the tangent stiffness matrix $\boldsymbol{K}^{i-1}$ is also important. Incorrect evaluation of this matrix often leads to divergence, however for sufficiently small errors the method *may* still converge to the correct solution, as long as $\boldsymbol{F}^{i-1}$ is correctly evaluated. Usually "sufficiently small" means at most a constant deviation from the correct expressions. So the method is a little forgiving towards errors in the individual terms, but if they become too large, it will diverge.

The basic steps in the derivation of the finite element equations are the same as those used in linear analysis; the selection of the interpolation functions and interpolation of element coordinates and displacements with these functions inserted in the governing equations. Also as in linear analysis, we may restrict ourselves to the derivation of the equations within a single element of a specific type, because the global equilibrium equations of the assemblage may be obtained using the same direct summation procedure.

The *general approach* to nonlinear finite element analysis can be likened to a series of lab tests, in which various assumptions are investigated in each step to attempt to gain some clarity of the problem behaviour. It possible, the nonlinear analysis should be preceded by a linear analysis, in which the most important particulars of the system under consideration can be investigated. Unfortunately, nonlinear analysis can be vexing in the regard that for an unfunctional system, the iterations may create a vast amount of information, and the analyst may have problems processing this in a way that will reasonably produce any answers.

3.2 Applications

Let us now attack some actual equations and attempt to solve them with the finite element method. In order to gain more insight into the process and, most importantly, possible hurdles, it is always recommended to start with an "'easy"' example. It is also helpful in order to prove the correctness of the solution algorithms involved. Hence, we will start with a single linear equation,, then gradually work our way up to more challenging tasks.

One-Equation Linear System

Consider the case in which we seek the solution $\Pi(v_1, v_2)$ of the following simple (and single) partial differential equation:

$$\frac{\partial \Pi}{\partial v_1} v_2 - \frac{\partial \Pi}{\partial v_2} v_1 + \Pi - v_1 = 0. \tag{3.24}$$

The above is derived from what is probably the easiest stable harmonically excited system, $\dot{x} = -x + u$ with $u = v_1 = a \sin \omega t$ leads to (3.24) with v_1 being the solution of the exosystem $\dot{\boldsymbol{v}} = \boldsymbol{S}\boldsymbol{v}$, $\boldsymbol{v} = [v_1 \ v_2]^T$. Taking a good look at (3.24) reveals that a linear combination of v_1 and v_2 is a good guess at the solution, and hence let

$$\boldsymbol{\Pi} = \boldsymbol{\Gamma}\boldsymbol{v} = \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix} = \Gamma_1 v_1 + \Gamma_2 v_2. \tag{3.25}$$

be a solution ansatz for (3.24). By inserting (3.25) into (3.24) and arranging the following is obtained:

$$(-\Gamma_2 + \Gamma_1 - 1)v_1 = (-\Gamma_2 - \Gamma_1)v_2 \tag{3.26}$$

such that we have the relations $-\Gamma_2 + \Gamma_1 - 1 = 0 \wedge -\Gamma_2 - \Gamma_1 = 0$, equating to $\Gamma_1 = 1/2 \wedge \Gamma_2 = -1/2$. The solution to (3.24) is then $\Pi(v_1, v_2) = \frac{1}{2}(v_1 - v_2)$, which may be easily checked to be correct by insertion.

Applying the finite element method to (3.24) is done in the standard way. The weighted residual formulation of the equation is found by weighing and integrating. We pick the linear triangle and its corresponding three shape functions N_1, N_2, N_3 as shown in (3.10), and insert for $\Pi = \sum_{i=1}^{3} N_i u_i$. Dropping the summations for brevity, the following expression is the finite element model, within a single element, of (3.24).

$$\int_\Omega \left[N_i \frac{\partial N_j}{\partial v_1} v_2 - N_i \frac{\partial N_j}{\partial v_2} v_1 + N_i N_j - N_i v_1 \right] \mathrm{d}\Omega = 0. \tag{3.27}$$

The element matrix $\boldsymbol{k}$, of size 3×3, is constructed from the terms involving the solution Π, and the element vector $\boldsymbol{f}$, of size 3×1, from the terms *not* involving the solution:

$$\begin{aligned} \boldsymbol{k}^{ij} &= \int_\Omega \left[N_i \frac{\partial N_j}{\partial v_1} v_2 - N_i \frac{\partial N_j}{\partial v_2} v_1 + N_i N_j \right] \mathrm{d}\Omega, \\ \boldsymbol{f}^i &= \int_\Omega N_i v_1 \ \mathrm{d}\Omega. \end{aligned} \tag{3.28}$$

Note that in the above the element vector is assumed to have moved to the right-hand side of the equation.

To check the correctness of the solution, we may use the analytical solution given above, however we may also map the solution of the state-space ODE, for a single amplitude and frequency, to the (v_1, v_2)-coordinates. This will, as described in Section 2.XX, create a circle which *must lie entirely within the plane of the solution.* This approach will be used since it is the only way to test the correctness of the solution in the nonlinear case.

We will scrutinize the individual elements of this system some more, to show the actual expressions that need to be integrated, and to choose a sufficient integration rule for the problem. Hence, take for example term $\boldsymbol{k}^{22}$. This term involves shape function N_2 only, and we may easily see that

$$\boldsymbol{k}^{22} = \int_\Omega \left[N_2 \frac{\partial N_2}{\partial v_1} v_2 - N_2 \frac{\partial N_2}{\partial v_2} v_1 + N_2^2 \right] \mathrm{d}\Omega. \tag{3.29}$$

The above must be written as a function (ξ, η) for the purposes of numerical integration. We have $N_2 = \xi$, further we must map v_1 and v_2 to $v_1(\xi, \eta)$ and $v_2(\xi, \eta)$ through the interpolation as shown in (3.7). The derivatives of N_2 with respect to v_1 and v_2 behave as constants. Let v_{11}, v_{12}, v_{13} be the v_1-coordinates of the element nodes, and similarly for v_2. Then, we end up with the following function to be integrated.

$$\gamma(\xi,\eta) = \xi\frac{\partial N_2}{\partial v_1}\big((1-\xi-\eta)v_{11} + \xi v_{12} + \eta v_{13}\big) - \xi\frac{\partial N_2}{\partial v_2}\big((1-\xi-\eta)v_{21} + \xi v_{22} + \eta v_{23}\big) + \xi^2. \tag{3.30}$$

Now, $\boldsymbol{k}^{22}$ is evaluated as

$$\boldsymbol{k}^{22} = \int_{\Omega_{[\xi,\eta]}} \gamma(\xi,\eta)\ \mathrm{d}\xi\mathrm{d}\eta = \sum_{k=1}^{M} \gamma(\xi_k,\eta_k)\ \det\big[\boldsymbol{J}\big] W_k. \tag{3.31}$$

Similar derivations hold for the calculation of all other terms in $\boldsymbol{k}$ and $\boldsymbol{f}$. Note that the maximum polynomial order of γ in (3.30) is 2, and we may also easily realize that this must hold for all other expressions to be integrated for this particular model as well. Therefore, the three-point integration rule shown in Table TAB should be capable of obtaining the exact values of these integrands, and is chosen for implementation.

The next step is, simply, assembling and solving the equations. Figure 3.4 shows the solution of (3.24) as obtained by the model (3.27). Further, Figure 3.5 shows the same solution, with the aforementioned mapped ODE solution embedded in the surface for four different values of the input amplitude (which equates to the radius in the (v_1,v_2)-system). It is clearly seen in all cases that the solution is correct. It may also easily be checked point-for-point with the analytical solution in this case.

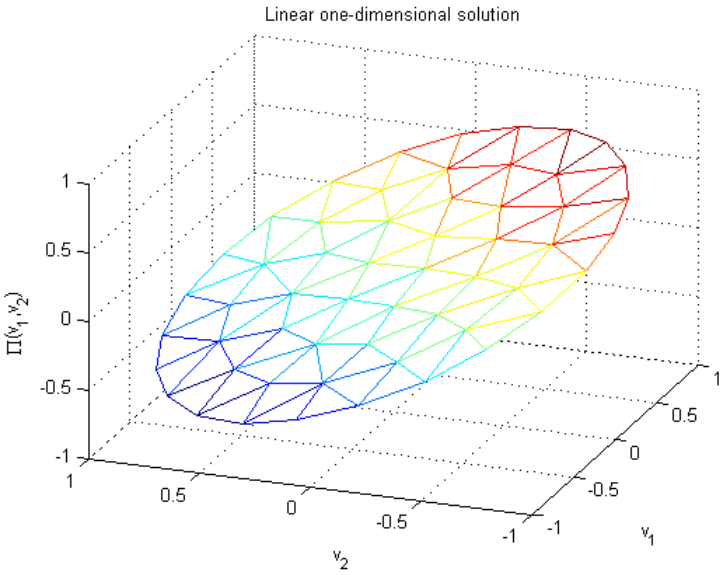

Figure 3.4: Solution of the single-equation linear system.

There is one more aspect that should be investigated before we move on from this simple example; that of the effect of reduced order numerical integration on this system. If the solution can be obtained with reduced integration, it would be more computationally effective. There are no two-point rules on the linear triangle, but we may try the one-point rule. The result of solving the system with this rule is shown in Figure FIG. It is immediately seen that this solution is not very accurate. Hence, we may predict that full-order integration should *always* be used for these systems. This also has ramifications for the nonlinear iteration process; having iteration solutions which are as inaccurate as that shown in Figure FIG would probably

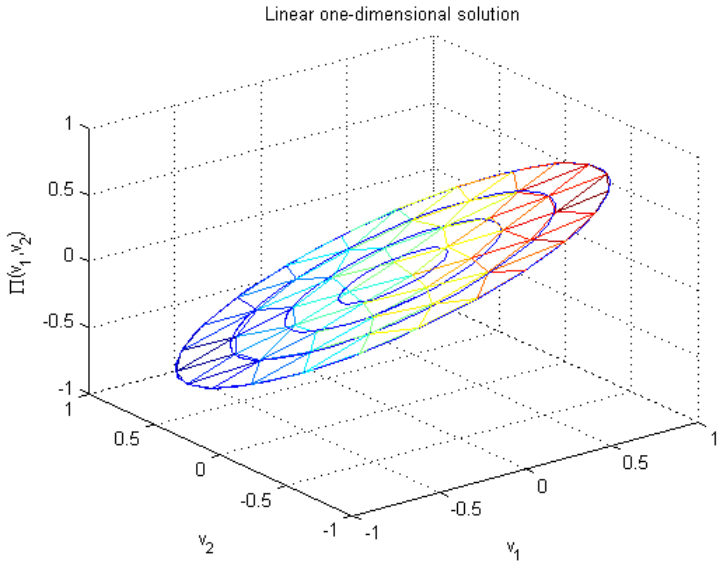

Figure 3.5: Solution of the single-equation linear system.

prevent convergence of the method alltogether.

Next, we treat a similar nonlinear problem.

One-Equation Nonlinear System

A nonlinear equivalent to (3.24) can easily be constructed by introducing a nonlinearity in the state equation. Thus, let the new state equation be $\dot{x} = -x^3 + u$, with $u = v_1$ from the exosystem as before. It is trivial to show that the PDE from which the steady-state solution may be found in this case is

$$\frac{\partial \Pi}{\partial v_1} v_2 - \frac{\partial \Pi}{\partial v_2} v_1 + \Pi^3 - v_1 = 0. \tag{3.32}$$

Applying the same procedure as in the linear case, the following finite element model of (3.32) is obtained.

$$\int_\Omega \left[\sum_{k=1}^{3} \left(N_i \frac{\partial N_j}{\partial v_1} v_2 - N_i \frac{\partial N_j}{\partial v_2} v_1 \right) u_k + N_i \left(\sum_{k=1}^{3} N_k u_k \right)^3 - N_i v_1 \right] \mathrm{d}\Omega = 0. \tag{3.33}$$

In the above we have used that the interpolation of Π is $\Pi = \sum_{i=1}^{3} N_i u_i$. We may further easily find the corresponding $\boldsymbol{k}^{ij}$ and $\boldsymbol{f}^i$:

$$\begin{aligned} \boldsymbol{k}^{ij} &= \int_\Omega \left[\sum_{k=1}^{3} \left(N_i \frac{\partial N_j}{\partial v_1} v_2 - N_i \frac{\partial N_j}{\partial v_2} v_1 \right) u_k + N_i \left(\sum_{k=1}^{3} N_k u_k \right)^3 \right] \mathrm{d}\Omega, \\ \boldsymbol{f}^i &= \int_\Omega N_i w_1 \, \mathrm{d}\Omega. \end{aligned} \tag{3.34}$$

It is important to note that, in the above, it is impossible to write the entire model as $\boldsymbol{ku} = \boldsymbol{f}$ due to the nonlinear term. Instead, we must write it simply as $\boldsymbol{k}(\boldsymbol{u}) = \boldsymbol{f}$. Hence, the summation is kept in the equations. Basically, (3.34) now describes one of the three equations needed to evaluate the three unknowns $u_1 u_2$,u_3. Again, closer inspection of the expressions to be integrated is required. Pick $\boldsymbol{k}^{22}$ as before. The

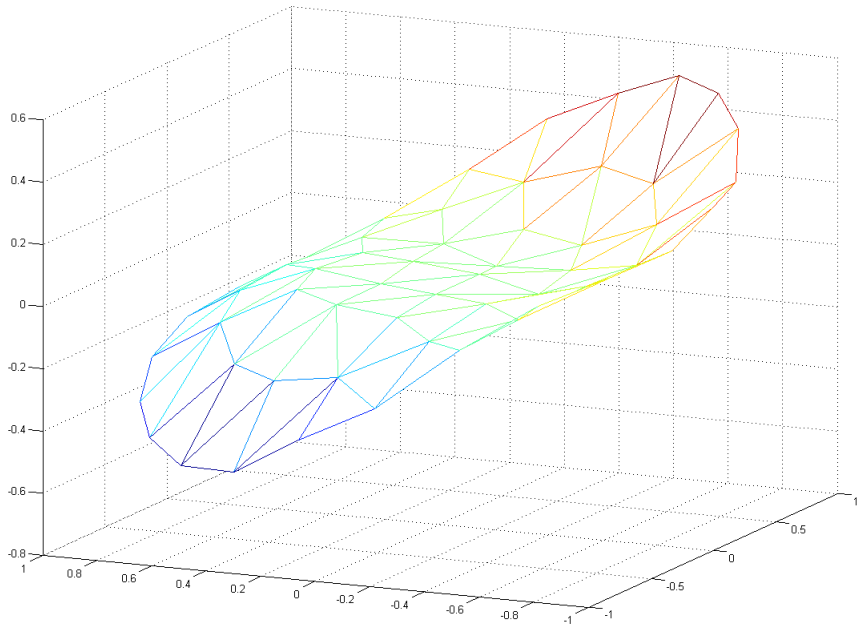

Figure 3.6: Linear one-point integration yields an unstable solution.

contribution from the linear part involving the derivatives is the same as with the linear equation, and the nonlinear part may be written, after expanding the sum:

$$\gamma_{nl}(\xi,\eta) = \xi\big((1-\xi-\eta)u_1 + \xi u_2 + \eta u_3\big)^3. \tag{3.35}$$

As before, $\boldsymbol{k}^{22}$ is obtained by integrating γ_{nl} (along with the linear contribution). The above expression will result in a highest order polynomial of 4. Therefore, the three-point integration rule is *no longer applicable* to this situation. The six-point rule shown in Table TAB is however able to integrate (3.35) exactly.

To solve the system (3.34), it is necessary to implement an iterative method as described in Section 3.1.6. Newton-Raphsons method is the choice, and we must now evaluate the basic equations to be solved using this procedure. Refer to Section 3.1.6 or Appendix B for more details.

We start by treating the *external* force term, denoted as $\boldsymbol{R}$. It was assumed that this was independent of the solution $\boldsymbol{U}$, and indeed it is:

$$\boldsymbol{R}_i = \int_\Omega N_i v_1 \, \mathrm{d}\Omega. \tag{3.36}$$

In other words, this term does not change with the iterations. Next, consider the *internal* contribution $\boldsymbol{F}$, which, for iteration k, becomes

$$\boldsymbol{F}_i^k = \int_\Omega \left[\sum_{k=1}^{3} \left(N_i \frac{\partial N_k}{\partial v_1} v_2 - N_i \frac{\partial N_k}{\partial v_2} v_1 \right) u_k + N_i \left(\sum_{k=1}^{3} N_k u_k \right)^3 \right] \mathrm{d}\Omega. \tag{3.37}$$

The final piece of the puzzle is the tangent stiffness matrix, which is the derivative of $\boldsymbol{F}$ with respect to the nodal variables.

$$\boldsymbol{T}_{ij}^k = \frac{\partial \boldsymbol{F}_i^k}{\partial \boldsymbol{U}_i^k} = \int_\Omega \left[\sum_{k=1}^{3} \left(N_i \frac{\partial N_k}{\partial v_1} v_2 - N_i \frac{\partial N_k}{\partial v_2} v_1 \right) + 3 N_i N_j \left(\sum_{k=1}^{3} N_k u_k \right)^2 \right] \mathrm{d}\Omega. \tag{3.38}$$

The notation may appear a little confusing in some of the above expressions. We will write out one element of $\boldsymbol{F}_i^k$ and the corresponding contribution to $\boldsymbol{T}_{ij}^k$ to hopefully make these slightly convoluted expressions clearer. Pick $\boldsymbol{F}_1^k$, and let u_1^k,u_2^k,u_3^k be the three unknowns inside a particular element for iteration k. Then, we have:

$$\begin{aligned}\boldsymbol{F}_1^k = \int_\Omega \Bigg[&\left(N_1\frac{\partial N_1}{\partial v_1}v_2 - N_1\frac{\partial N_1}{\partial v_2}v_1\right)u_1^k + \left(N_1\frac{\partial N_2}{\partial v_1}v_2 - N_1\frac{\partial N_2}{\partial v_2}v_1\right)u_2^k + \left(N_1\frac{\partial N_3}{\partial v_1}v_2 - N_1\frac{\partial N_3}{\partial v_2}v_1\right)u_3^k \\ &+ N_1\left(N_1u_1^k + N_2u_2^k + N_3u_3^k\right)^3\Bigg]\,\mathrm{d}\Omega.\end{aligned} \tag{3.39}$$

The above expression is one of three equations for any given element and gives rise to three elements of the tangent stiffness matrix, found by differentiating $\boldsymbol{F}_1^k$ with respect to u_1^k, u_2^k, and u_3^k, respectively:

$$\begin{aligned}T_{11} &= \int_\Omega \left[\left(N_1\frac{\partial N_1}{\partial v_1}v_2 - N_1\frac{\partial N_1}{\partial v_2}v_1\right) + 3N_1^2\left(N_1u_1^k + N_2u_2^k + N_3u_3^k\right)^2\right]\mathrm{d}\Omega, \\ T_{12} &= \int_\Omega \left[\left(N_1\frac{\partial N_2}{\partial v_1}v_2 - N_1\frac{\partial N_2}{\partial v_2}v_1\right) + 3N_1N_2\left(N_1u_1^k + N_2u_2^k + N_3u_3^k\right)^2\right]\mathrm{d}\Omega, \\ T_{13} &= \int_\Omega \left[\left(N_1\frac{\partial N_3}{\partial v_1}v_2 - N_1\frac{\partial N_3}{\partial v_2}v_1\right) + 3N_1N_3\left(N_1u_1^k + N_2u_2^k + N_3u_3^k\right)^2\right]\mathrm{d}\Omega.\end{aligned} \tag{3.40}$$

Similar derivations naturally hold for the remaining six entries of the tangent stiffness matrix. Note that we keep the integration for the terms in $\boldsymbol{T}_{ij}^k$ as well. Since we differentiate with respect to u_k, the integration, which is in terms of ξ and η, is simply treated as a constant. Vice versa, u_k may be treated as constants when performing the integration.

We now have all the needed terms to perform the Newton-Raphson iterative procedure. Denote u_1^{k-1}, u_2^{k-1} and u_3^{k-1} as the values of $\boldsymbol{U}$ belonging to a particular element, for iteration $k-1$. Then, the system to be solved in iteration k of the Newton-Raphson method is:

$$\begin{bmatrix}T_{11} & T_{12} & T_{13}\\ T_{21} & T_{22} & T_{23}\\ T_{31} & T_{32} & T_{33}\end{bmatrix}^{k-1}\begin{bmatrix}\Delta u_1^k\\ \Delta u_2^k\\ \Delta u_3^k\end{bmatrix} = \begin{bmatrix}R_1\\ R_2\\ R_3\end{bmatrix} - \begin{bmatrix}F_1^{k-1}\\ F_2^{k-1}\\ F_3^{k-1}\end{bmatrix}, \qquad u_i^k = u_i^{k-1} + \Delta u_i^k. \tag{3.41}$$

As has been mentioned in Appendix APP, the equilibrium equations for the total assemblage of elements may be constructed using the same procedure as in linear analysis, even with the terms involved in the Newton-Raphson procedure.

Again, the highest polynomial order in any term in (3.41) is 4, and a six-point rule is sufficient for exact integration.

We may now proceed to implement the above iteration and attempt to solve the system (3.41). Unfortunately, the method does not seem to be able to converge. To gain some insight into what is happening, consider the error norm $||\boldsymbol{R} - \boldsymbol{F}||$, in other words a norm of the residual of the equations. Figure 3.7 shows what is happening to this expression. Clearly, we have *divergent oscillations*. Some effort must now be made towards establishing why this is occuring.

The first steps should obviously be checking whether there are any errors in the equations or their implementation. Consider the equations. The basic steps that lead to the complete finite element model equations are;

1. The weighing and integrating of the partial differential equations (weighted residual statement).

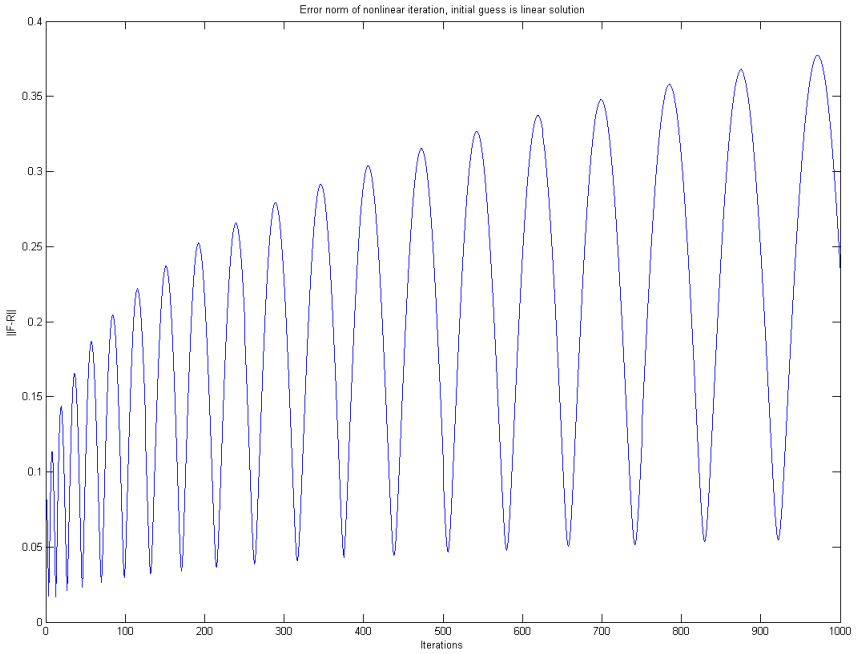

Figure 3.7: The error norm of the nonlinear solution where the initial guess is the linear solution.

2. The choice of some approximating interpolation $\Pi = \sum_i N_i u_i$ on the individual mesh elements.
3. The insertion of this approximation into the governing model.
4. Assembly to establish the equilibrium equations over the whole assemblage of elements.

Let us consider these steps in turn to see whether any errors could have been made here. The first step is obviously correct in that it makes mathematical sense, as has already been argued in Section 3.1.1. The second step only involves the choice of the shape functions N_i to be used, that these are correct has been argued in Section 3.1.3. The third step is again correct because the expressions are still mathematically valid. The fourth step sees us arrive at the total global system of equations, which is correct since the same algorithm has been used to arrive at the linear equilibrium equations, which as we have seen yield the correct solution. It may also be proven mathematically. In fact, the correctness of the mathematics and the algorithms involved may easily be argued through the support of a correctly functioning linear solution.

Let us dwell some further with the expressions. It can easily be seen that the nonlinear finite element model is similar to the linear model, except for the one nonlinear term. Hence, we may reuse the expressions obtained from a linear model (these may also be very effectively precomputed) and concentrate only on the effect the nonlinear term, $N_i\big(\sum_{i=1}^{3} N_i u_i\big)^3$, has on the implementation of the equations.

Firstly, we state that, at the end of each iteration, this expression will be a number. Therefore, it is possible to handle the nonlinear term in the code in the same way that the v_1 and v_2 terms are handled where the multiply their respective derivatives in the expressions. We may precalculate the interpolated expression $\Pi = N_1 u_1 + N_2 u_2 + N_3 u_3$ in the same way that $v_1 = N_1 v_{11} + N_2 v_{12} + N_3 v_{13}$ is calculated. Then, We may weigh the interpolated version of Π with the correct shape function (depending on what node of the element we are currently treating), and proceed to integrate the expression numerically. It has already been covered that the numerical integration rule is correct and can exactly recreate the polynomials that are involved.

In addition, we may show that the nonlinear iteration is working as intended by substituting the linear system into the nonlinear iteration. This must converge in the first iteration with the exact solution, which it does.

We may also try other methods or modifications of the standard Newton-Raphson procedure to see if convergence can be reached. A simple modification is to decrease the step length by a factor $\alpha < 1$ (typically α lies in the 0.30-0.50 range), such that the solution update becomes

$$u_i^k = u_i^{k-1} + \alpha \Delta u_i^k. \tag{3.42}$$

The above may now be attempted for varying values of α, $0 < \alpha < 1$. No values of α has been found that ensure convergence, so this is clearly a dead end as well.

Another method may be to try a different solver for the equations. The matlab function *fsolve* is often used to solve systems of nonlinear equations. By default, *fsolve* uses a trust-region dogleg method [], which is a quite different approach to the Newton-Raphson method. The nonlinear equation system $\boldsymbol{F} - \boldsymbol{R}$ may be implemented in the same way as earlier for this purpose, and the *fsolve* function called with some initial guess $\boldsymbol{u}_0$. Again, the correctness of the implementation is most easily verified by solving the linear system with the same method, which indeed yields the correct solution.

In fact, this approach, for a simple nine element mesh, converges to a solution with an error norm that satisfies $||\boldsymbol{R} - \boldsymbol{F}|| \leq 10^{-16}$, or in other words, machine accuracy. The solution is depicted in Figure 3.8 along with a certain simulated single circle along edge of the circle.

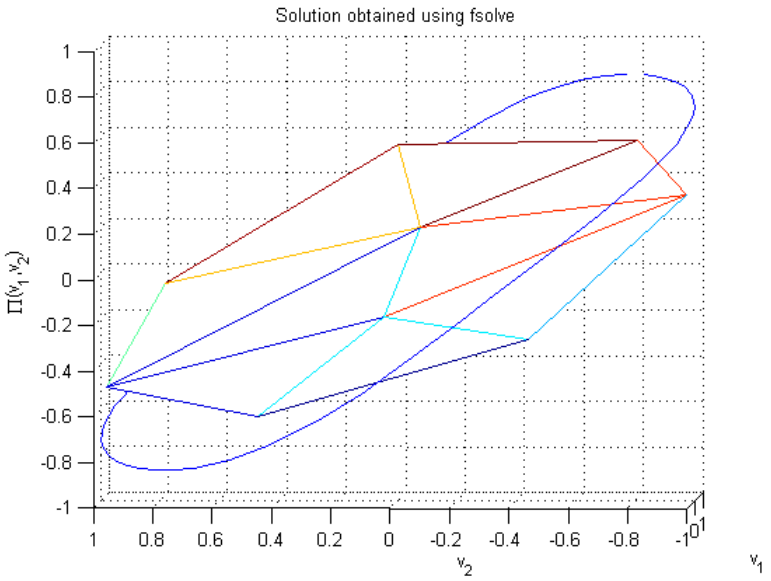

Figure 3.8: The "solution" to the nonlinear problem obtained using fsolve.

It is seen that the solution is not at all very accurate, *yet it is a solution to the equation system* (it has already been argued that the equations themselves are correct, and correctly implemented). This may easily be seen as a conflicting (not to mention confusing) result. Further investigation is no doubt needed. It is well known that the solution accuracy should increase with increasing mesh density for finite element methods, so we may next try to solve the system on a denser mesh. The mesh depicted in Figure **??** has 55 elements. Trying to solve the nonlinear equations on this mesh yields no convergence for any attempted method, including *fsolve*.

The next step to try may be to vary the initial guess, and overall checking whether Property 3.1 is fulfilled. It may easily be found that the tangent stiffness matrix is always nonsingular (except in extreme cases

of divergence when the solution has blown up too much). Hence, we will try moving the initial guess closer and closer to the actual solution. The actual solution may be found by manually assigning values that are in close agreement with the simulated solution to the nodes. If this is done to good accuracy, it will *sligthly* better the error norm of the residual in the Newton-Raphson iteration. We may also attempt to polish a root by picking out the solution iterate for which the error is smallest (typically occurs within the first 25 iterations), and using this new solution as the guess with which to restart the iterations. This will also slightly better the error norm, but not to a degree where it can be argued a converged solution is reached, see Appendix B for some discussion around these error norms and how they should be interpreted. The term 'slightly better the error norm' here refers to a change of about one order of magnitude (10^{-2} to 10^{-3}) for the iterations. Figure 3.7 shows the error norm plotted as a function of the iteration number if the initial guess is the linear solution, while Figure 3.9 shows the same situation after several steps of 'root-polishing' has been taken. Additionally, Figure 3.10 shows the error norm when the initial guess is rather arbitrary. Obviously no acceptable converged solution is reached. At any rate, using a method which relies this much on manual labor would not be very well adaptable to solutions over finer mesh sizes, which would be required for good accuracy.

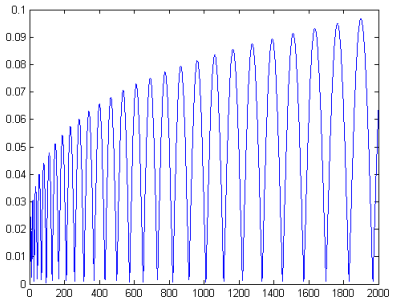

Figure 3.9: The error norm of the nonlinear solution after some root polishing has been attempted.

Let us look at cases where the Newton-Raphson method produces the kind of response we have seen. In fact, the kind of nonlinearities we have restricted ourselves to very much pose a problem for this method in some situations. It is relatively easy to (and commonplace as an example of failure of Newton-based methods) find a third-order equation or system which may fail to converge to a solution if the initial guess is too inaccurate. *However, as long as a solution can be found, the iteration should converge if the initial guess is accurate enough.* In this case then, we clearly have a situation in which no solution can be found. It is also interesting to look at how the response actually develops. It appears to be *diverging* to the linear solution.

Let us briefly go back to the linear solution and explore some peculiarities of this model. Firstly, note that we have insofar not touched the aspect of *boundary conditions*. Simply put, the original partial differential equations on which the finite element modelling is based does not come with any specific boundary conditions that are known to be required in order to find an unique solution to the system. *If boundary conditions are not stated by the PDE model, then they do not have to be included in the finite element model.* Indeed in the linear case, the exact solution is obtained without the addition of any boundary conditions. If one adds the correct boundary conditions to the system, it will not change the response in any way. If incorrect boundary conditions are added (even if only one node is incorrect, and even if the error is minimal), then the solution to the linear system will be divergent. Even so, one could attempt to use boundary conditions in the nonlinear case. The correct values of the boundary nodes may be manually assigned to good accuracy as has been described above. For the nine-element mesh, we would now only need to actually compute values for the two nodes at the mesh interior - all other nodal values are fixed. This may be tried - in fact if it is, no solution can be found to the system in any case.

At any rate, and as has been mentioned, so long as the original partial differential equation model does not explicitly include any other conditions, we should not expect to need to model them. The way the boundary conditions are obtained in this case - is actually by solving the equations. There is no other way

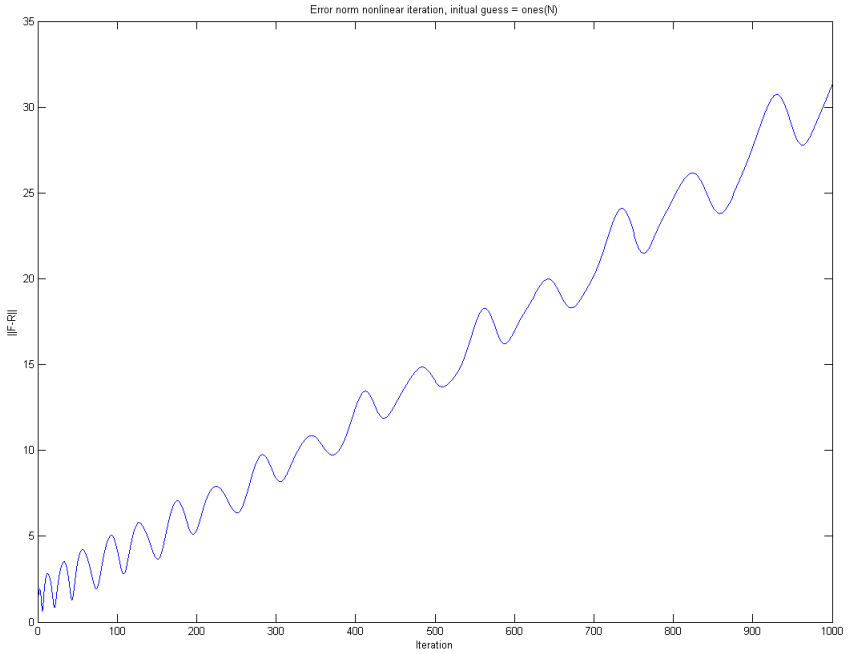

Figure 3.10: The error norm of the nonlinear solution where the initial guess is arbitrary.

to compute the values of this surface solely by working on the ordinary differential equations from which the model ultimately stems. Hence, arguing that we need boundary conditions to stabilize the solution is like arguing that we need to solve the equations, in order to solve the equations. It does not make perfect sense. Also, because there are enough support conditions to uniquely recreate the solution in the linear case, and the mesh has not changed, it should still be possible to recreate the solution without boundary information. However, it *may* also be that the original model is incomplete - though this is not likely, looking at the particulars of the linear solution.

Maybe it is time to actually take a look at the surface we are trying to calculate. Recall that the linear equations yielded a linear surface. Now, obviously, we are looking at a nonlinear surface. As before, we may calculate single circles with different amplitudes of the input. Figures 3.11, 3.12, and 3.13 show three different angles of this surface, for values of the amplitude $a = 0.05, 0.25, 0.50, 0.75, 1.00$. This gives an idea of what the surface actually looks like in threedimensional space. Now, evidently, staring at these figures in the report may not be the best solution - it is hard to convey threedimensionality using five circles "'shot"' from different angles and projected on flat paper. It is recommended that the Matlab .fig file that corresponds to FIG is studied instead, see Appendix C detailing the content of the attached files. We are looking to get some basic information about the shape of this surface. The shape is cubic - evidently governed by the chosen nonlinearity. It is very unlikely that the linear elements can adapt to this situation.

Now, it might seem that it has been argued that linear elements are generally insufficient to solve nonlinear problems. This is *not* the case. Certainly, for all PDEs whose solution has a well-behaved parabolic shape, for instance, linear elements are perfectly admissible. The process of solving the nonlinear equations using linear elements can be alikened to that of bending a jigsaw puzzle, each individual piece of jigsaw representing one linear element. Initially, as with the linear system, we are okay because we are solving a flat surface. In the nonlinear case, we may think that we are beginning to bend and twist this jigsaw. Initially, the jigsaw will give in and adapt somewhat to this action, but if the bending or twisting becomes too severe, it will

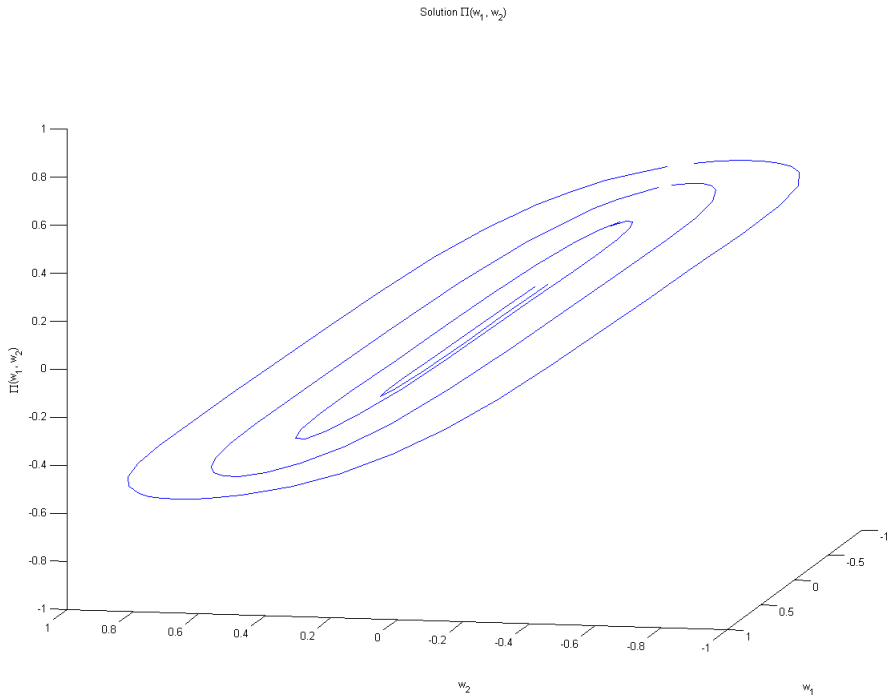

Figure 3.11: A simulated version of the nonlinear surface, seen slightly from above.

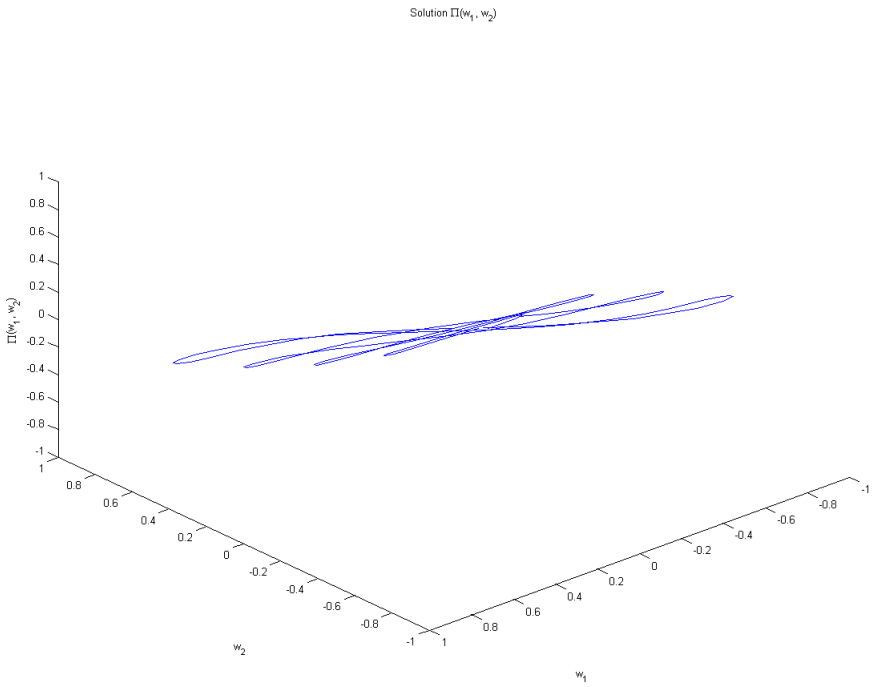

Figure 3.12: A simulated version of the nonlinear surface, seen from the side.

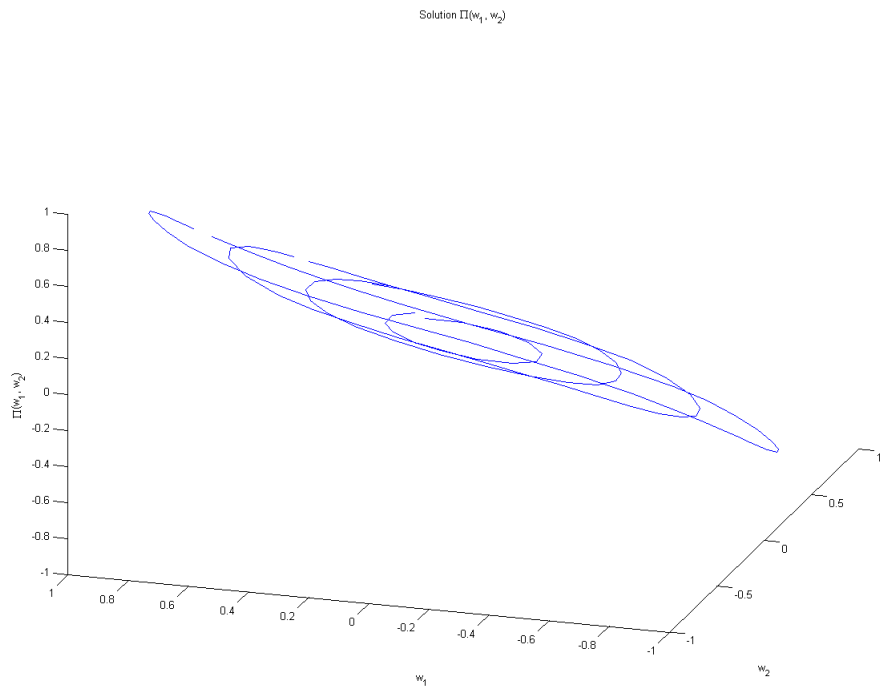

Figure 3.13: A simulated version of the nonlinear surface, seen slightly from below and the side.

break, because the individual pieces of jigsaw are stiff and cannot be properly adjusted. Here is the main realization about why convergence cannot be reached, for this nonlinear system, using linear elements:

The linear elements are much too stiff.

Simply put, the linear elements cannot properly adapt to the flow of the surface in order to achieve a solution with good enough accuracy to ensure convergence.

To properly understand why some finite elements fail in certain situations, it is necessary to know something about the mathematical relationship between the mesh geometry, interpolation errors, and stiffess matrix conditioning. This is an active research field within finite element methods, and [18] [19] are excellent introductory papers in this field. Any depth study into this is beyond the scope of this thesis, however some general comments can be made that shed a slightly more mathematical light on the failure of the linear elements for this nonlinear application.

The basic idea of the finite element method application herein is to, by interpolation on a triangular mesh, construct a function that attempts to approximate some "true" function, whose exact identity is not perfectly known. Herein, the sole purpose of the elements is to be a basis for the triangulation - and the primary criterion of fitness in this regard is how much the interpolated function differs from the true function. There are two types of this *interpolation error* that matter for most applications; the difference between the interpolated function and the true function, and the difference between the gradient of the interpolated function and the gradient of the true function. Let us return to the situation in which we tried to heighten the mesh density to see if it improved the convergence (which it did not). Initially, one might think that this points to some basic mistake in either the equations or the implementation of the method - after all, if the true function is smooth, one would expect to see significant error reduction in the interpolated function simply by making the triangles smaller. Indeed this is true, however what such reduction there is can easily be "nullified" by added errors in the gradient. *Errors in the gradient can be surprisingly important.*

The *curvature* of the true function (the surface we are trying to approximate) certainly plays a role also. Given some vector-valued function $\boldsymbol{f}(\boldsymbol{p})$, the curvature is defined by

$$\kappa = \boldsymbol{d}^T \boldsymbol{H}(\boldsymbol{p})\boldsymbol{d} \tag{3.43}$$

Where $\boldsymbol{H}(\boldsymbol{p})$ is the *Hessian* matrix of $\boldsymbol{f}$, and $\boldsymbol{d}$ is a unit direction vector. In short, the curvature of the surface along a certain direction it is the second derivative along that direction. The approximative power of the element depends largely on how well it can predict the curvature of the surface where the element is placed. While we for this application do not know $\boldsymbol{f}$, we do have information about its shape, as described in an earlier paragraph. At certain points in the surface, the curvature will change signs. We will now present a statement regarding the suitability of linear elements for such situations:

Statement 3.1. *If any linear element overlaps a point in which the curvature of the surface changes sign, the error in the interpolated function and its gradient can be large.*

The above is intuitive - the predictive capability of any linear element is only good enough to affix a flat surface, which has zero curvature and a constant directional derivative. In an iterative method, such elements may induce significant error to the system by effectively "changing" their appearance between iterations - oscillating between a positive and negative direction. Naturally, exactly how large the errors are is problem dependent, and an analysis of the situation would require knowledge of both the true function and the interpolating function. However, we may construct a small thought experiment to see what may be happening in this case. Consider an element which has two nodes on one side of a curvature sign change, and the last node on the other side. The linear element can obviously only replicate a linear displacement field *within* the element. Therefore, it cannot accurately predict the actual response, and will tend to either slope in the direction of the curvature before the curvature sign change, or in the direction of the curvature after the curvature sign change. So at least one node in the element may be fairly inaccurately predicted. Depending on how the nonlinear iteration is updating the solution, the element may also change direction between iterations. Exactly how and when this may be occuring is dependent on many factors, and is fairly futile to attempt to analyze for a large system. In addition, the element is coupled to other elements, so the inaccuracies tend to "spread" across the entire surface, even if they are severe only for a few elements or nodal points. These effects are hard to analyze without the proper tools to do so. It is enough to conclude that significant errors may be introduced in the finite element model due to the possibility of having such behaviour.

We can now also attempt to shed some light on why the solution accuracy seems to decrease with increased mesh density. Simply put, the more elements there are in the mesh for an unstable method, the worse we can expect the errors to become. For example, the more elements experience curvature changes "from within", the more error overall will be added to the system.

We may also point out that the consistency requirement that is vital for convergence, as described in Appendix A, does not hold for the linear simplex element model. Hence, the method does not converge.

One can conclude this discussion with the following statement, to which all signs now point.

Statement 3.2. *For the nonlinear systems considered in this thesis, and using only linear elements, the combined effect of interpolation errors, discretization errors, and inaccurate gradient information will prevent convergence of the finite element method as applied to these systems.*

There is one aspect which might have struck the reader as strange by now. Why are we not attempting to properly quantify the effect of these error sources on the system? Again, the problem is too complex for it to be any use at all. Even if we could, for example, compute some values which tell us how the gradient information changes with varying conditions, there would be nothing to compare these numbers to. Here is another statement.

Statement 3.3. *In the finite element method, the only information available to recreate the solution comes*

from the choice of elements and the integration of the element shape (interpolating) functions.

Again - fairly obvious. However it also means that a test in which we, for example, were to manually assign the correct values of the solution and insert these into the equations to test whether the equations are actually correct, can be flawed, since the interpolation information is not accurate enough to be able to predict the exact behaviour of the unknowns. Hence, even these tests may produce answers that can be confusing.

So, the linear elements will have to be scrapped. Unfortunately, this took a long time to properly realize. Is there now a way forward with the finite element method for this application? Possibly - we may simply consider nonlinear elements instead. A nonlinear element can curve in the cartesian space - which obviously adds much accuracy to the solution and may be enough to ensure convergence of the method. We will briefly discuss such possible ways forward in a later chapter.

Two-Equation Linear System

While we have now discussed the particulars of the nonlinear system for some time, we must also be able to solve a *coupled* system of partial differential equations. Again, we will start with something simple. One of the easiest two-equation systems is a linear mass-spring-damper system of the form $m\ddot{x} + d\dot{x} + kx = u$ where x is the position, m is the mass, d is the damping, k is the spring constant and u a driving force. This system may trivially be represented in state space as

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -\frac{b}{m}x_1 - \frac{k}{m}x_2 + \frac{1}{m}u \end{aligned} \tag{3.44}$$

Assume for added simplicity that $m = b = k = 1$. Again let $u = w_1$ be the driving force as generated by the exosystem. Then, we have a function $\Pi(w_1, w_2)$ as before, that is the solution to the above system with respect to harmonic inputs. The insertion of this function into the ODE model yields the following set of partial differential equations to be solved.

$$\begin{aligned} \frac{\partial \Pi_1}{\partial w_1}w_2 - \frac{\partial \Pi_1}{\partial w_2}w_1 - \Pi_2 &= 0 \\ \frac{\partial \Pi_2}{\partial w_1}w_2 - \frac{\partial \Pi_2}{\partial w_2}w_1 + \Pi_1 + \Pi_2 - w_1 &= 0 \end{aligned} \tag{3.45}$$

The main idea behind solving this coupled system with the finite element method is to construct an extended system matrix consisting of $p \times p$ submatrices where p is the number of unknowns in the equation system. Submatrix number (i, j) with $i, j \in [1\ p]$ must then reconstruct the contribution in equation number i from unknown number j. With these ideas it is quite simple to see that one gets the following system matrix and vector for the system (3.45).

$$\boldsymbol{K} = \begin{bmatrix} [\boldsymbol{k}^{11}] & [\boldsymbol{k}^{12}] \\ [\boldsymbol{k}^{21}] & [\boldsymbol{k}^{22}] \end{bmatrix}, \qquad \boldsymbol{F} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{bmatrix}, \tag{3.46}$$

where

$$
\begin{aligned}
\boldsymbol{k}_{ij}^{11} &= \int_{\Omega} \left[N_i \frac{\partial N_j}{\partial w_1} w_2 - N_i \frac{\partial N_j}{\partial w_2} w_1 \right] \, \mathrm{d}\Omega, \\
\boldsymbol{k}_{ij}^{12} &= \int_{\Omega} \left[-N_i N_j \right] \, \mathrm{d}\Omega, \\
\boldsymbol{k}_{ij}^{21} &= \int_{\Omega} \left[N_i N_j \right] \, \mathrm{d}\Omega, \\
\boldsymbol{k}_{ij}^{22} &= \int_{\Omega} \left[N_i \frac{\partial N_j}{\partial w_1} w_2 - N_i \frac{\partial N_j}{\partial w_2} w_1 + N_i N_j \right] \, \mathrm{d}\Omega, \\
\boldsymbol{f}_i^1 &= 0, \\
\boldsymbol{f}_i^2 &= \int_{\Omega} \left[N_i w_1 \right] \, \mathrm{d}\Omega.
\end{aligned}
\tag{3.47}
$$

Again, assembly of this total system follows the steps already outlined. Also, note that this system is cyclically coupled - it is not possible to reduce the complexity of the system by removing any of the unknowns. Hence, it has to be solved "as is" - solving all the unknowns simultaneously in one solution. Fortunately, the finite element method is easily extendable to this situation. It is enough to formulate the finite element model as usual, and extend the global system matrix and vector in such a way that both unknowns can be tackled in the same structure.

One more important point should be noted, about the expressions shown in (3.47). Since we are using the same shape functions in all expressions, the four stiffness matrices are actually constructed from only two unique expressions;

$$
\begin{aligned}
I_1 &= \int_{\Omega} \left[N_i \frac{\partial N_j}{\partial w_1} w_2 - N_i \frac{\partial N_j}{\partial w_2} w_1 \right] \, \mathrm{d}\Omega, \\
I_2 &= \int_{\Omega} \left[N_i N_j \right] \, \mathrm{d}\Omega.
\end{aligned}
\tag{3.48}
$$

And we may see that we can construct the entire global matrix for the finite element solution of (3.45) by precalculating these expressions (which, obviously, also are the same expression that turn up in the previous one-equation problems), then combining them to form the required system. This is computationally effective, since the required matrices in (3.48) are constant for any fixed mesh, and may be very effectively precomputed and stored. Let us now combine these expressions and solve the two-equation coupled system. This done, the solution for Π_1 and Π_2 are depicted in Figures 3.14 and 3.15 respectively, with as before the simulated single-circle response embedded in the surface to prove that it is indeed the correct solution to these unknowns. As expected, the response is entirely linear.

3.3 Application to Case Study

It is time to apply the theory of the previous section to the case study. Consider the system as stated in (.). This particular set has six partial differential equations in the six unknowns $\Pi_1, \cdots, \Pi_6$. As per the techniques of the finite element method, an approximation of the following form is assumed for the six

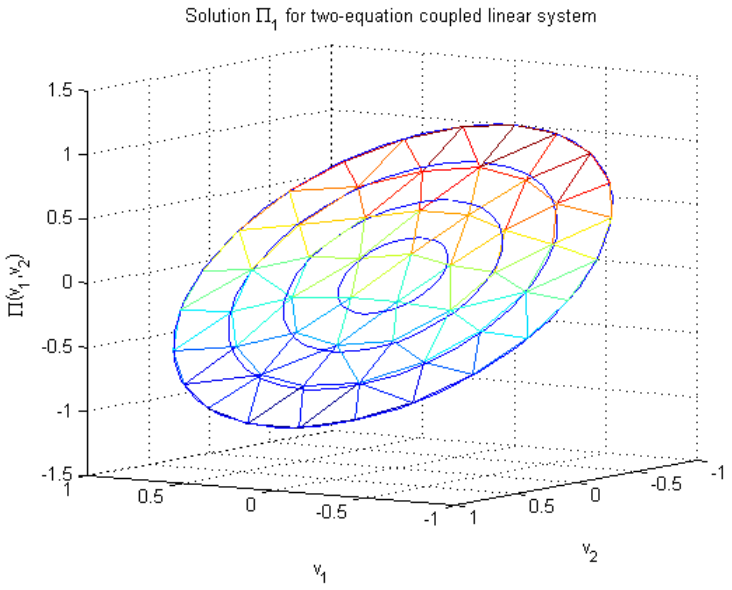

Figure 3.14: The computed linear solution for Π_1 with a coupled system.

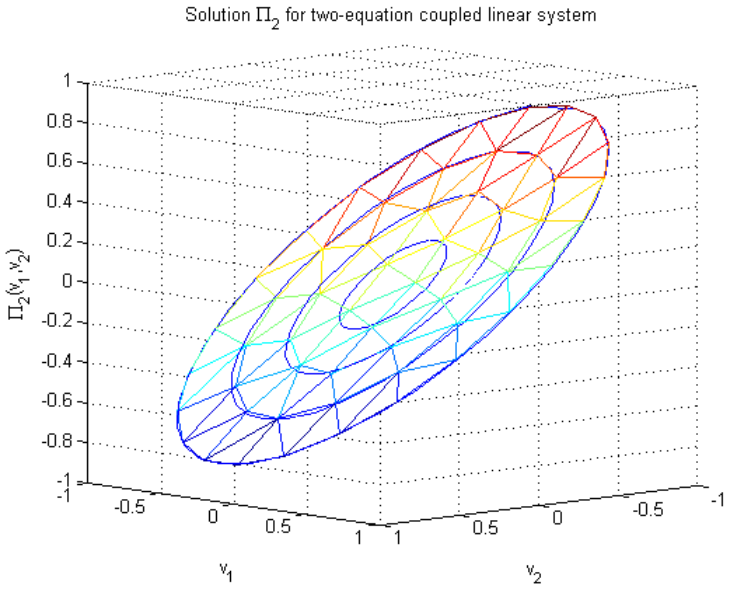

Figure 3.15: The coupled linear solution for Π_2 with a coupled system.

unknowns in the problem.

$$\begin{aligned}
\Pi_1(w_1,w_2) &= \sum_{i=1}^{3} N_i^e u_i^e, & \Pi_2(w_1,w_2) &= \sum_{i=1}^{3} N_i^e v_i^e,\\
\Pi_3(w_1,w_2) &= \sum_{i=1}^{3} N_i^e w_i^e, & \Pi_4(w_1,w_2) &= \sum_{i=1}^{3} N_i^e q_i^e,\\
\Pi_5(w_1,w_2) &= \sum_{i=1}^{3} N_i^e r_i^e, & \Pi_6(w_1,w_2) &= \sum_{i=1}^{3} N_i^e s_i^e.
\end{aligned} \tag{3.49}$$

Here the N_i^e are the approximating shape functions, u_i^e are the unknown nodal values in the mesh corresponding to Π_1, v_i^e correspond to the unknown nodal values of Π_2, and so on through s_i^e going with Π_6. As we have seen with the two-equation system, the total global system matrix for this case will have one "submatrix" for each nonzero entry of the corresponding system of ODEs. Hence, the global system matrix will have the general form shown below.

$$\begin{bmatrix}
[K^{11}] & [K^{12}] & [0] & [0] & [K^{15}] & [0]\\
[K^{21}] & [K^{22}] & [0] & [0] & [0] & [0]\\
[0] & [K^{32}] & [K^{33}] & [0] & [0] & [0]\\
[K^{41}] & [K^{42}] & [K^{43}] & [K^{44}] & [0] & [0]\\
[0] & [0] & [0] & [0] & [K^{55}] & [K^{56}]\\
[0] & [0] & [0] & [K^{64}] & [K^{65}] & [K^{66}]
\end{bmatrix}
\begin{bmatrix} \boldsymbol{u}\\ \boldsymbol{v}\\ \boldsymbol{w}\\ \boldsymbol{q}\\ \boldsymbol{r}\\ \boldsymbol{s} \end{bmatrix}
=
\begin{bmatrix} \boldsymbol{F^1}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0} \end{bmatrix}. \tag{3.50}$$

The weighted residual equations as per the Galerkin finite element method are also straightforwardly found. Choose a weight function w, then weigh and integrate;

$$\begin{aligned}
\int_\Omega w\left[\frac{\partial\Pi_1}{\partial v_1}\omega v_2 - \frac{\partial\Pi_1}{\partial v_2}\omega v_1 + 2\beta\omega_{lp}\Pi_1 + \omega_{lp}^2\Pi_2 + k_p\omega_{lp}^2\Pi_5 - \omega_{lp}^2\Pi_5^3 - \omega_{lp}^2 v_1\right]\,\mathrm{d}\Omega = 0\\
\int_\Omega w\left[\frac{\partial\Pi_2}{\partial v_1}\omega v_2 - \frac{\partial\Pi_2}{\partial v_2}\omega v_1 - \Pi_1\right]\,\mathrm{d}\Omega = 0\\
\int_\Omega w\left[\frac{\partial\Pi_3}{\partial v_1}\omega v_2 - \frac{\partial\Pi_3}{\partial v_2}\omega v_1 - \Pi_2\right]\,\mathrm{d}\Omega = 0\\
\int_\Omega w\left[\frac{\partial\Pi_4}{\partial v_1}\omega v_2 - \frac{\partial\Pi_4}{\partial v_2}\omega v_1 - \frac{\omega_a}{\omega_d}\Pi_1 - \omega_a(1+\frac{\omega_i}{\omega_d})\Pi_2 - \frac{\omega_a}{\omega_i}\Pi_3 + \omega_a\Pi_4\right]\,\mathrm{d}\Omega = 0\\
\int_\Omega w\left[\frac{\partial\Pi_5}{\partial v_1}\omega v_2 - \frac{\partial\Pi_5}{\partial v_2}\omega v_1 - \Pi_6\right]\,\mathrm{d}\Omega = 0\\
\int_\Omega w\left[\frac{\partial\Pi_6}{\partial v_1}\omega v_2 - \frac{\partial\Pi_6}{\partial v_2}\omega v_1 - \frac{1}{m}\Pi_4 + \frac{k}{m}\Pi_5 + \frac{b}{m}\Pi_6\right]\,\mathrm{d}\Omega = 0
\end{aligned} \tag{3.51}$$

As before, we have the linear triangular mesh with linear triangular shape functions. Then, we may easily insert the interpolation into the equations above to obtain the finite element model of the case study system.

$$\begin{aligned}
\int_\Omega N_i\left[\frac{\partial N_j}{\partial v_1}\omega v_2 - \frac{\partial N_j}{\partial v_2}\omega v_1 + 2\beta\omega_{lp}N_j + \omega_{lp}^2 N_j + k_p\omega_{lp}^2 N_j - \omega_{lp}^2(\sum_{k=1}^{3} N_k u_k)^3 - \omega_{lp}^2 v_1\right]\,\mathrm{d}\Omega = 0 \\
\int_\Omega N_i\left[\frac{\partial N_j}{\partial v_1}\omega v_2 - \frac{\partial N_j}{\partial v_2}\omega v_1 - N_j\right]\,\mathrm{d}\Omega = 0 \\
\int_\Omega N_i\left[\frac{\partial N_j}{\partial v_1}\omega v_2 - \frac{\partial N_j}{\partial v_2}\omega v_1 - N_j\right]\,\mathrm{d}\Omega = 0 \\
\int_\Omega N_i\left[\frac{\partial N_j}{\partial v_1}\omega v_2 - \frac{\partial N_j}{\partial v_2}\omega v_1 - \frac{\omega_a}{\omega_d}N_j - \omega_a(1+\frac{\omega_i}{\omega_d})N_j - \frac{\omega_a}{\omega_i}N_j + \omega_a N_j\right]\,\mathrm{d}\Omega = 0 \\
\int_\Omega N_i\left[\frac{\partial N_j}{\partial v_1}\omega v_2 - \frac{\partial N_j}{\partial v_2}\omega v_1 - N_j\right]\,\mathrm{d}\Omega = 0 \\
\int_\Omega N_i\left[\frac{\partial N_j}{\partial v_1}\omega v_2 - \frac{\partial N_j}{\partial v_2}\omega v_1 - \frac{1}{m}N_j + \frac{k}{m}N_j + \frac{b}{m}N_j\right]\,\mathrm{d}\Omega = 0
\end{aligned} \tag{3.52}$$

We skip the attempt at solving a linear version of these equations since neglecting the nonlinearity in this case, and with the constants at their defined values, generates a system which is very badly conditioned. In fact, trying to find a solution to the governing ODEs in this case is futile - no solution can be found. Here we hit another numerical obstacle in the solution of this system - the system matrices as generated with the finite element method for this case will inherit any conditioning problems, which may destabilize the method when an inversion of the system matrix is performed. At any rate, we have already outlined how to solve a coupled system, as long as it does not induce any additional numerical problems in the system matrices. It must be stressed that the matrices that are presented are correct - however in this particular case, there is no solution to the governing ODE system if the system is simply linearized by neglecting or linearizing the nonlinearity. Hence, the finite element model is not supposed to find a solution either.

Nonlinear analysis on the system is also skipped - the system will inherit the problems of the one-equation nonlinear system, as well as possible other obstacles with the inversion of system matrices destabilizing the method due to the numerical values, which adds yet another error source to the equation.

Chapter 4

Ways Forward

In the previous chapter, we have shown that a linear simplex element is not applicable in order to recreate the solution to the problems considered in this thesis. Is there now a way forward? Well, me may as suggested attempt to use other elements, or we may use a different method alltogether. First, we will cover two theories as to how work may progess with the finite element method.

4.1 Nonlinear Elements in the Finite Element Method

Since we are solving a nonlinear problem, it may be wise to consider nonlinear elements as well. By nonlinear finite element we mean that the shape functions will, in general, be nonlinear polynomials of the local coordinates. While the linear triangle can only exactly recreate linear surfaces for example, the quadratic triangle can exactly recreate a parabolic surface. Such a triangle has six nodes instead of three, and can curve in the cartesian space. Naturally, the accuracy of the solution would improve significantly using such elements, and it may even be enough to obtain a convergent method. Since the surface we are trying to establish has a cubic shape, the gradient will be parabolic. Hence, the quadratic element should be able to interpolate the gradient exactly, while still leaving some errors in the interpolation of the actual surface. This may be enough to ensure convergence - the only way to know for sure is to actually try it.

Unfortunately, no freely available meshing tool that can generate such a mesh, that is useful for the purposes of this project has been found. Hence, one can only stipulate the theory that it might work, and attempt to show the way forward.

In fact, if one could get the mesh information in a form that could easily be implemented in Matlab, it would not be hard to modify the finite element model to account for this new mesh. We simply need the shape functions of the quadratic triangle, and we may do all the necessary derivations from there. A quadratic triangle is shown in 4.1, and the shape functions for this triangle are, denoting $\zeta = 1 - \xi - \eta$:

$$\boldsymbol{N}^T = \begin{bmatrix} \zeta(2\zeta - 1) \\ \xi(2\xi - 1) \\ \eta(2\eta - 1) \\ 4\zeta\xi \\ 4\xi\eta \\ 4\zeta\eta \end{bmatrix}. \tag{4.1}$$

We may now obtain the Jacobian as before, and through the shape functions and the Jacobian terms obtain the cartesian derivatives as before. Then, we would have a finite element interpolation $\Pi = \sum_{i=1}^{6} N_i u_i$ over the quadratic element. All the previously discussed factors of numerical integration and assembly of the equations are unchanged, except for the Jacobian matrix also becoming a function of position. This is easy to deal with however as we will always know the position values at any given point, so they may just be inserted directly.

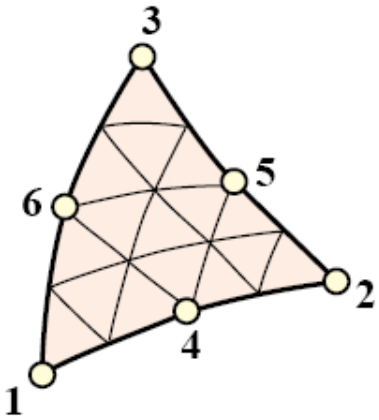

Figure 4.1: A general quadratic triangular element.

If the quadratic element is not sufficient, then one may try a *cubic* element also, like the 10-node cubic element shown in 4.2. The shape functions for this element are, respectively; $N_1 = \frac{1}{2}\zeta(3\zeta - 1)(3\zeta - 2)$, $N_1 = \frac{1}{2}\xi(3\xi - 1)(3\xi - 2)$, $N_3 = \frac{1}{2}\eta(3\eta - 1)(3\eta - 2)$, $N_4 = \frac{9}{2}\zeta\xi(3\zeta - 1)$, $N_5 = \frac{9}{2}\zeta\xi(3\xi - 1)$, $N_6 = \frac{9}{2}\xi\eta(3\xi - 1)$, $N_7 = \frac{9}{2}\xi\eta(3\eta - 1)$, $N_8 = \frac{9}{2}\zeta\eta(3\eta - 1)$, $N_9 = \frac{9}{2}\zeta\eta(3\zeta - 1)$, $N_{10} = 28\zeta\xi\eta$. Again, the same procedures as before may be used to obtain a finite element model using this interpolation. This cubic interpolation should be able to exactly interpolate *both* the actual response *and* the gradient, something that may just be required for convergence of this method. However, as it can be seen, this will yield a quite complex analysis process.

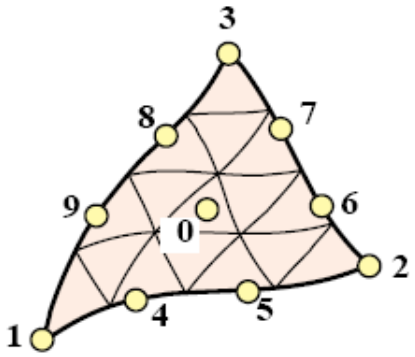

Figure 4.2: A general cubic triangular element.

Some tools would be required to make the process of testing these assumptions even remotely feasible.

4.2 Direct computation of the frequency response functions

We have seen that the response of a nonlinear system to general harmonic disturbance may be represented as a certain system of partial differential equations. However, it is also certainly possible to use a "brute-force" method, in which the ODE system is simulated subject to harmonic inputs $u = a\sin\omega t$ for ranges of a

and ω. Hence, we may compute these solutions for the desired ranges of the amplitude and frequency, and simply construct the frequency response functions directly from this information. This is *very* ineffective computationally, and was abandoned as soon as better solutions surfaced. Hence, it should not be considered.

4.3 Fast computation of the frequency response functions

While the title of this section is slightly dubious, it actually mimics the title of the paper [4]. In this paper, an iterative numerical method for computing periodic responses to periodic excitations for Lur'e-type nonlinear systems is derived. Addtionally, the procedure has guaranteed convergence for arbitrary initial values. In this procedure, periodic responses are represented in terms of the Fourier coefficients. We will not restate these equations here as they can be found in the referred paper, but the main points will be briefly outlined.

In fact, the convergence of this procedure can be proved to be exponential (which is not the case for a finite element solution). The procedure is based on doing the nonlinear update in the time domain, where this is effective, and transferring back and forth between the frequency domain (where the rest of the update is perfomed) and the time domain using the fast fourier transform. Hence, it is a very effective procedure.

Chapter 5

Conclusions

In this thesis, we have attempted to solve a coupled nonlinear set of partial differential equations using certain numerical techniques. The main focus has been on the finite element method application, which has been shown to be unfunctional, in the nonlinear cases, for simplex meshes.

At some point in the work, it had probably been favorable to the overall result if the finite element method had been abandoned, and other methods pursued instead, especially the method so briefly described in Section 4.3. This would have allowed for the solution of the system to be found, and thus progress could have been made towards performance analysis and controller tuning of the case study. Evidently, this path was not chosen. It may be advantegous to give an explanation as to why - clearly it would have been the better choice?

In short, the author (unfortunately) chose to keep going with the finite element application, trying to gain more insight into the workings of the method, believing that a solution could be found with this method. If so, it would have been a good contribution to the original problem. At a certain point in the work, the author was reasonably certain of the findings of this report, and the choice was now between abandoning the finite element method as a solution candidate, and instead carrying on with one of the applicable methods described earlier. However, there was also only a limited time left before the thesis deadline. This was like being stuck between a rock and a hard place - should one concentrate on developing the already established findings, possibly making a breakthrough, or should these be abandoned, and a solution attempted to be found with a different method in a much more limited timeframe? The choice was the former - based on a wrongfully held viewpoint that a breakthrough could be accomplished - and it may very well have been a poor choice. In addition, the "alternative" method is in itself not particularly straightforward and would in all probability require a good amount of time to properly adapt to the situation, and for the author to absorb the theory needed to apply said method, time that may no longer be available.

The abscence of proper tools that can be applied to the situation has been glaring. There is a significant part of the project work, at least in terms of man-hours, that is not done proper justice previously in the thesis - that of looking for software tools that may aid in the process, and trying out many different such tools. However, it has been found that all of the publicly available tools for this problem have some limitations that would render them practically useless for the application. The following lists what such tools should be able to accomplish, should they be considered applicable, and it must be stressed that the below is viewed as *requirements* of software tools for this problem.

- It must be possible to construct circular meshes of cubic triangles.
- The tools must allow for a coupled nonlinear problem to be user-defined over this mesh.
- There must be some solver capability that is able to solve the problem (nonlinear solver).
- The output data must be easily accessible for postprocessing work, including analysis of the data, and more importantly, use of the solution data to generate the needed functions for performance analysis.

- As an addition to the above point, since the software intended for use for the performance analysis is Matlab, there should be an easy way to import the data into Matlab, perhaps through an interface, or saving the data to disk in an easily readable format.

When we look at all of the above, we can begin to see that the availability of *free* software tools that can meet *all* of these demands is probably quite low. In fact, the author has found it to be relatively nonexistent. Many different tools have been tried - some may allow you to define a mesh, but not define or solve the problem, some may have capability to solve the problem, but not to define the mesh. Most have no interface towards Matlab, the few free tools who do, naturally, cannot solve the problem. It may now be thought that it could be possible to combine the capabilities of different tools, instead of looking for the one perfect tool that will do the job, however this has been shown to be infeasible as well, mainly because there is no proper interfacing between these tools. In short, trying to combine several tools to obtain a solution ends up being a worse "hack" than just attempting to do it yourself.

However, there does exist software tools that would *probably* be sufficient to aid in the solution of this problem with the finite element method. While they may be no sufficiently capable free software tools, there are several *commerically* available finite element packages that boast all of the requirements in the above list. The drawback - they cost a significant amount of money, and naturally a lot more than it is realistic to spend on software for a masters project, especially when reaching an acceptable solution with the software cannot be viewed as guaranteed. The "cheapest" software solution that *might* be able to deal with the problem is currently priced at $895 for a limited academic license.

Much time has also been seemingly "lost" trying to quantify the effects of various potential error sources on the system, when in fact these error cannot be properly quantified. In short, it may very well not be possible at all to come up with an error bound which dictates the point in which the finite element solution becomes unstable for this particular system, and in fact, such error bounds are rare in the literature. There exists a small amount of somewhat available information in the form of certain research papers and other documents, such as [17] [18], which can be found in the references. To give a specific example, consider the effect of inaccurate gradient information on the system. To properly quantify these effects, it would be necessary to obtain the exact (or at least converged) values of the gradient at every nodal point. Without this information, the issue cannot be further studied.

Some clarifications must also be made with regards to the outcome of the earlier work on this problem by the author, which was described in a previously delivered project report. While many of the findings of that report were correct, there were also a few either incorrect or misguided ones, including a couple of pretty severe errors, as it turns out. For example, the previous project focuses on the effect of boundary conditions on the problem in a way that is incorrect. Hence, the following few paragraphs will in some ways be an excercise in "debunking" some of the authors previous work, but it needs to be done in order to be perfectly clear about some of the properties of this problem. For example, on the last paragraph of page 43 of the previous report, the following can be read: [quote]...a quick convergence test for the implementation may be to simply stipulate some arbitrary boundary conditions and check if the solution still converges smoothly...[unquote].

This statement is not just wrong, it is *complete nonsense*. The issues of boundary conditions and why they should not have to be considered for this problem has already been lit. In addition, trying to add arbitrary boundary conditions to the problem should destabilize the method alltogether, be it the nonlinear or linear version. While the following is evident by now, it also needs to be stated; *the previously presented solution to the nonlinear problem is incorrect.*

In fact, the linear solution presented in the previous work is also dubious. It was generated using the boundary conditions - again a completely false assumption. *It was probably incorrect.* In this thesis, it is conclusively shown how to, in general, deal with the linear systems. Naturally - for linear systems we do not need this amount of complexity to analyze its performance, so this is fairly useless overall. Additionally, there are further problems with the case study problem.

At this stage the author cannot honestly recommend further study into the solution of this probem using the finite element method, as the process of doing would probably be too involved, especially given that there are at least one perfectly workable solution method that is both easier, more practical, and in all probability more efficient than a finite element solution. If the author is correct, and a 10-node triangular element is required to interpolate the solution to a degree where convergence can be reached, it would severely affect the overall computational time of the method.

Unfortunately, the project has proven, in some ways "too hard" to be entirely tractable for the author. The application of the finite element method quickly diverges into advanced finite element studies, which are in no way included in the previous coursework of the studies. Significant and important time is lost just "searching" for that one piece of good information - simply because it is not known where to find it. The amount of finite element books in the references do not do the actual time and effort spent pouring over books, lecture notes, publications, and various other online resources justice either. It has proven fairly fruitless - most textbooks consider only problems which are in all honesty more tractable than the problem considered here, and the amount of ideas and insight that can be gained from simply "studying more" is very limited.

When studying finite element methods, there are two basic approaches - to learn to apply the methods to general engineering problems, and to pursue research into the methods themselves. The author has opted to go with the first approach - believing that as an engineering problem, this is achievable, and by studying the application of the finite element method to other engineering problems enough insight and information will be gained that the solution may also be found for this problem. It is fairly obvious that this was a wrongful assumption as well. In this case, the problem has more than one "peculiarity" which is not shared by any other system, or discussed by any textbook, the author has seen.

The problem **may** also not be convex, an impression that is conveyed partly by the involved expressions and partly by the shape of the surface. Certainly, many cubic problems are in general not convex, and again in general, convex problems are hard to solve using common iterative methods. For example, Newton-based methods and line search based methods tend to fail for nonconvex problems, which also adds to the headache considering the issue of stabilizing the considered systems. While this paragraph should be taken with a pinch of salt, as the author is unaware how to conclusively show whether the problem is nonconvex or not, it would probably add significantly to the complexity involved in solving the system should it be nonconvex.

It is, given all of these considerations, fair to say that the solution of the problems considered in this thesis with the finite element method is simply this:

A dead end.

Due to all the listed problems, it is the authors belief that the problem is not scalable on the level of engineering problems wherein the finite element method may relatively easily be applied. Further study in this direction would require an effort which is prohibitive, considering the availability of other methods which are in all honesty better than a finite element solution will be, as well as the complete abscence of proper tools that would greatly aid such investigations. For example, as we have briefly touched, the solution method proposed in [4] is exponentially convergent, does not require any tools or extras per se. A finite element based method will at most attain quadratic convergence rates in general - sometimes far less.

While this thesis does not in any way deliver in terms of computation of frequency reponse functions, it is believed that the findings in the thesis still deserve some merit. It must be stressed that it is very hard to quantify the effects of some of the individual error sources and pecularities that are encountered with these systems, when the readily available literature does not at all deal with these problems. At the same time, while it is unfortunate that a different method was not attempted, when such a method was available, it was in the end a choice of how to proceed, and given the authors uncertainty about the findings for some time, as well as other aspects that were briefly discussed, was not perceived as an alltogether easy choice.

Additionally, the thesis debunks some of the earlier work in the same direction, arguing several incorrect

aspects of this work, albeit it was performed by the same author. Still, more clarity into the workings of the finite element method as applied to these systems has been achieved - unfortunately these new findings have had a tendency to lead to more questions that are even harder to answer.

It should also be noted that finite element methods does not play a mentionable role in any course the author has attended, it is entirely self-taught. While sometimes self-taught is well-taught, the author has found the problem at hand too convoluted, and useful information too hard to come across, to make more progress than what the thesis currently outlines.

Now, it would have been custom to outline some possible future work in these directions. No such work will be outlined. It is not good use of time overall, since it should be clear that the finite element method is not particularly applicable. Any future work should use the already functional method [4] on this problem.

Appendix A

Convergence in Finite Element Methods

This appendix covers the topic of convergence as it applies to finite element methods to some extent. Because of the number of topics involved, which are here mostly briefly discussed, this was omitted in large from the main part of the thesis.

Let us assume that the problem we wish to solve has a nice and smooth solution. What are then the conditions for a finite element model to converge to this solution in the limit of the mesh size approaching zero? For the models considered in this text, there are two sufficient conditions for convergence to occur; *consistency* and *stability*.

A.1 Consistency

Consistency is a front for two individual requirements, *completeness* and *compatibility*. The present discussion will be centered around static problems, but may be extended to time-dependent problems (more or less) straightforwardly.

A.1.1 Completeness

Simply put, the elements must have enough *approximative power* to capture the analytical solution in the limit of the mesh refinement. This is mainly an intuitive statement; exactly whether a particular element fulfills this requirement will in general depend on the problem being solved.

A.1.2 Compatibility

To state this requirement succinctly, we introduce the concept of a *patch*. A patch is the set of all elements attached to a given node. Further, we may define a finite element patch trial function by the union of all shape functions "activated" by setting a certain degree of freedom in the patch to unity, while the other degrees of freedom are zero. Such a function "propagates" only over the patch, and is zero beyond it. This property follows from the local-support requirement; a shape function for node i should vanish on all sides or faces that do not include i. We may now continue in this way, obtaining a patch trial function over the entire finite element mesh. If this function is *conforming*, then compatilbility holds. By conforming we mean that the patch trial functions must be continuous between elements, and at least piecewise differentiable within the elements.

In practice, it is enough that the mesh is matching. That is, there are no elements in the mesh that share edges with elements whose node number for that particular edge is different. For example, trying to combine linear and quadratic triangles anywhere in the mesh would not lead to a matching mesh, and hence

compatibility would not be satisfied.

Naturally, this also means that as long as we use the same element for all elements in the mesh, compatibility is *always* satisfied.

A.2 Stability

Stability may be informally characterized as ensuring that the finite element model enjoys the same solution uniqueness properties as the analytical solution of the mathematical model.

Additionally, since the finite element method can handle arbitrary assemblies of elements, including individual elements, this property is required to hold at the element level.

In the present outline we will be considering stability at the element level. Stability is *not* a property of shape functions per se, but of the implementation of the element as well as its geometrical definition. It involves two subordinate requirements; rank sufficiency, and Jacobian positiveness. Rank sufficiency is the most important aspect. Jacobian positiveness corresponds to an element which is not distorted in a way such that it overlaps itself, and for the problems considered in this report such a scenario will never occur.

Rank sufficiency will in these problems be guaranteed on the circulat triangulated mesh because there are enough support conditions, if in addition the numerical integration rule is sufficient for exact integration.

Appendix B

Solution Methods for Nonlinear Algebraic Equations

This Appendix describes the details concerning solution methods for nonlinear algebraic equations, as applies to the finite element method in the nonlinear case. The focus is on iterative techniques as was briefly discussed in Section 3.1.5.

The most frequently used iteration schemes for the solution of nonlinear finite element equations are the Newton-Raphson iteration. Here this method will be derived in a more formal manner. Various solution methods are discussed extensively in readily available literature, and some excellent references are [12] and [7].

B.1 The Newton-Raphson Method

The finite element equilibrium requirements amount to finding a solution of the equations

$$\boldsymbol{f}(\boldsymbol{U}^*) = \boldsymbol{0}, \tag{B.1}$$

where

$$\boldsymbol{f}(\boldsymbol{U}^*) = \boldsymbol{R}(\boldsymbol{U}^*) - \boldsymbol{F}(\boldsymbol{U}^*) \tag{B.2}$$

The solution is then denoted as $\boldsymbol{U}^*$. Assume that the iterative solution has established $\boldsymbol{U}^{i-1}$, then a Taylor series expansion gives:

$$\boldsymbol{f}(\boldsymbol{U}^*) = \boldsymbol{f}(\boldsymbol{U}^{i-1}) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{U}}\right|^{i-1} \cdot (\boldsymbol{U}^* - \boldsymbol{U}^{i-1}) + \text{h.o.t.} \tag{B.3}$$

Where h.o.t means higher order terms. These can be assumed to be very small and are neglected. Now substitue (B.1) and (B.2) into the above, which leads to

$$\left.\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}}\right|^{i-1} \cdot (\boldsymbol{U}^* - \boldsymbol{U}^{i-1}) = \boldsymbol{R} - \boldsymbol{F}^{i-1}. \tag{B.4}$$

In the above it is assumed that the externally applied loads $\boldsymbol{R}$ are independent of the solution. We can now calculate an increment in the displacements $\boldsymbol{U}$:

$$\boldsymbol{K}^{i-1}\Delta\boldsymbol{U}^i = \boldsymbol{R} - \boldsymbol{F}^{i-1} \tag{B.5}$$

Where $\boldsymbol{K}^{i-1}$ is the current tangent stiffness matrix. The improved displacement solution is

$$\boldsymbol{U}^i = \boldsymbol{U}^{i-1} + \Delta\boldsymbol{U}^i \tag{B.6}$$

The iteration as above is continued until appropriate convergence criteria are satisfied.

A characteristic of this iteration is that a new tangent stiffness matrix is used in *each* iteration, which is why this method is also referred to as the full Newton-Raphson method. Figure FIG illustrates the process of solution for a single degree of freedom system. The nonlinear response characteristics are such that convergence is rapidly obtained. However, it is easy to image a more complex characteristic with a starting point of iteration which does not converge. The situation in Figure FIG describes the general idea, but is only a well-behaved single degree-of-freedom system. In the solution of systems with many degrees of freedom, the response curves will in general be rather nonsmooth and complicated.

Considering the Newton-Raphson iteration it is recognized that in general the major computational cost per iteration lies in the calculation and factorization of the tangent stiffness matrix. Since these calculations can be quite expensive when large-order systems are considered, the use of a modification of the full algorithm may be effective. One such modification is to only use the inital stiffness matrix $\boldsymbol{K}^0$ in the iterations, and hence operate on the system

$$\boldsymbol{K}^0 \Delta \boldsymbol{U}^i = \boldsymbol{R} - \boldsymbol{F}^{i-1} \tag{B.7}$$

again with initial guess $\boldsymbol{U}^0$. In this case only one version of the tangent stiffness matrix needs to be calculated, that of the initial guess. This "initial stress" method corresponds to a linearization of the response about the initial configuration and may converge slowly or even diverge. In general, it has worse convergence properties than the full Newton-Raphson iteration, however it may be a more computationally effective method for the cases on which it works.

Additionally, it is possible to hit an in-between of the two above variants by calculating a new tangent stiffness matrix only at certain iteration intervals or based on some measure of the progress of the algorithm.

B.2 Convergence Criteria

Realistic criteria should be used for termination in any iteration method. At the end of each iteration, the solution must be checked to see whether it has converged within preset tolerances or whether the iteration is diverging. If the tolerances are too loose, inaccurate results are obtained, and if they are too strict, the method may never terminate with an accepted solution. The objective in this section is to briefly discuss some convergence criteria.

Since we are seeking the displacements corresponding to iteration i, it is realistic to require that the displacements at the end of each iteration be within a certain tolerance of the true solution. Hence, one convergence criterion could be

$$\frac{||\Delta \boldsymbol{U}^i||}{||\boldsymbol{U}||} \leq \epsilon_1, \tag{B.8}$$

where ϵ_1 is some convergence tolerance. The vector $\boldsymbol{U}$ is not known and must be approximated in some way. Frequently, it is enough to use the solution for the previous attempted iterate for this purpose, however there are cases in which a convergence test such as the above may terminate and still be far away from the actual solution, for example if the calculated displacements only change a little from iteration to iteration, but continue to change for many iterations.

A second convergence criterion may be to measure the out-of-balance load vector. For example, we may require that the norm of this vector be within a certain tolerance of the initial load increment:

$$||\boldsymbol{R} - \boldsymbol{F}^i|| \leq \epsilon_2 ||\boldsymbol{R} - \boldsymbol{F}^0||. \tag{B.9}$$

A problem with the above is that the displacement solution does not at all enter into the termination criterion. In some cases it is possible that the out-of-balance loads are small yet the solution is much in error. The following may be noted about the two above criteria in general:

The convergence criteria in (B.8) and (B.9) should only be used with very small values of ϵ_1 and ϵ_2.

A third criterion may include some indication of both when the forces and the displacements are in equilibrium. A criterion which measures this is based on the increment of internal energy between the iterations, and reads:

$$\Delta(\boldsymbol{U}^i)^T\big(\boldsymbol{R}-\boldsymbol{F}^{i-1}\big) \leq \epsilon_3\big(\Delta(\boldsymbol{U}^0)^T\big(\boldsymbol{R}-\boldsymbol{F}^0\big)\big) \tag{B.10}$$

In practice the above is often an attractive measure.

Appendix C

Use of the Accompanying Files

The files that accompany the thesis may be used to generate some of the figures.

To start, locate and execute*run.m* in the Master directory. This should attempt to solve the nonlinear single equation using fsolve for a nine element mesh.

After (and only after) this step, type *Lin2DSol* to bring up the plots for the linear solutions.

Execute *FSolNL* to run an iteration attempt using the Newton-Raphson method on the simple mesh.

The figures that are used in the report may be found in the Figures subdirectory. They mostly have understandable namings. In particulars, the plot of the surface that was mentioned in the report, is filed under the name *SolutionPISimul.fig*.

Appendix D

The Mesh Generator

The mesh generator that is used in this thesis for generation and manipulation of the meshes is called DistMesh, and is Copyright Per-Olof Persson and Gilbert Strang with the Massachussetts Institute of Technology. The generator is freely available software written in Matlab, and is distributed under the terms of the GNU General Public License published by the Free Software Foundation; license version 2 or later.

The generator can handle virtually any shape in two or three dimensions, but is only able to triangulate the regions using simplex triangular elements.

More information about the mesh generator can be found in [11], as well as at its homepage;

www-math.mit.edu/ persson/mesh/.

References

1. A. Pavlov, N. van de Wouv, H. Nijmeijer. Frequency response functions for nonlinear convergent systems, *IEEE Trans. Automatic Control, 52(6), 1154-1159 (2007).*

2. M. Heertjes, E. Pastink, N. van de Wouv, H. Nijmeijer. Experimental Frequency-Domain Analysis of Nonlinear Controller Optical Storage Drives. *IEEE Trans. on Control Systems Technology, Vol. 14, No. 3, May 2006.*

3. A. Pavlov, N. van de Wouv, H. Nijmeijer. *Uniform output regulation of nonlinear systems; a convergent dynamics approach.* Birkhauser, Boston, 2005.

5. S. Skogestad, I. Postlethwaite. *Multivariable Feedback Control Analysis and Design.* John Wiley and Sons, England, 2007.

6. H.K. Khalil. *Nonlinear systems*, 2nd ed. Prentice Hall, Upper Saddle River, 1996.

7. J.N. Reddy. *Nonlinear finite element analysis.* Oxford University Press, 2004.

8. A. Pavlov, N. van de Wouw. *Fast computation of frequency response functions for a class of nonlinear systems.* Preprint submitted to 47th IEEE Conference on Decision and Control.

9. Y.W.Kwon, H.Bang. *The finite element method usling Matlab.* CRC Press, 1997.

10. S.S. Rao. *The finite element method in engineering.* Elsevier Science Technology Books, 2004.

11. P.O. Persson, G. Strang, *A simple mesh generator in Matlan.* SIAM Review Vol. 46 (2) 2004.

12. K.J. Bathe. *Finite Element Procedures.* Prentice Hall, 1996.

13. S.C. Brenner, L. Ridgway Scott. *The mathematical theory of finite element methods.* Springer Science+Business Media, LLC, 2008.

14. O.C. Zienkiewicz, R.L. Taylor. *The Finite Element Method - Volume 1: The Basis.* Butterworth-Heinemann, 2000.

15. O.C. Zienkiewicz, R.L. Taylor. *The Finite Element Method - Volume 1: Solid Mechanics.* Butterworth-Heinemann, 2000.

16. O.C. Zienkiewicz, R.L. Taylor. *The Finite Element Method - Volume 1: Fluid Dynamics.* Butterworth-Heinemann, 2000.

17. J.R. Shewchuk. *What Is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy, and Quality Measures.* Unpublished preprint, 2002.

18. J.R. Shewchuk. *What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measures.* Eleventh International Meshing Roundtable (Ithaca, New York), pages 115-126, Sandia National Laboratories, September 2002.

19. J.E. Flaherty. *Finite Element Analysis.* Lecture Notes, CSCI, Math 6860, Spring 2000.

20. G.F. Carey. *Computational Grids: Generation, Adaptation, and Solution Strategies.* Series in Computational and Physical Processes in Mechanics and Thermal Science, Taylor and Francis, New York, 1997.

21. G.F. Carey, J.T. Oden. *Finite elements: computational aspects, volume III.* Prentice Hall, Englewood Cliffs, 1984. **22**. S.G. Stan. *The CD-ROM drive; a Brief System Description.* Dordrecht, the Netherlands: Kluwer, 1998.