Computational studies of two unconventional approximation schemes for mechanical system responses

A project report submitted in fulfilment of the requirements

for the degree of Master of Technology

by

Vaibhav Pratap Singh

(16807768)

to the

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

July 2021

Certificate

It is certified that the work contained in this project entitled **Computational studies of two unconventional approximation schemes for mechanical system responses** by **Vaibhav Pratap Singh** has been carried out under my supervision and that it has not been submitted elsewhere for a degree to the best of my knowledge.

Prof. Anindya Chatterjee

Professor

Department of Mechanical Engineering

Indian Institute of Technology Kanpur

July 2021

Declaration

This is to certify that the project report titled **Computational studies of two unconventional approximation schemes for mechanical system responses** has been authored by me. It presents the research conducted by me under the supervision of Prof. Anindya Chatterjee

To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted elsewhere, in part or in full, for a degree. Due credit has been attributed to the relevant state-of-the-art and collaborations (if any) with appropriate citations and acknowledgements, in line with established norms and practices.

Vaibhav Pratap Singh

BT-MT (Dual Degree)

Department of Mechanical Engineering

Indian Institute of Technology, Kanpur

Kanpur - 208016

Abstract

Name of the student: **Vaibhav Pratap Singh** Roll No: **16807768**

Degree for which submitted: **M.Tech.** Department: **Mechanical Engineering**

Project title: **Computational studies of two unconventional approximation schemes for mechanical system responses**

Project supervisor: **Prof. Anindya Chatterjee**

Month and year of project submission: **July 2021**

Two different and unconventional approximation techniques for two types of mechanical system responses are developed and presented in this study.

Many systems exhibit decaying responses, and the first problem studied involves approximating such responses. In particular, a new delayed dynamical system is constructed using convolution such that one of ~~the~~ its responses is exactly equal to the desired response curve. The aim of approximating the curve in this way, using a dynamical system, is that the the dynamical system has characteristic roots which lead to exponential curves with a prior guarantee that those exponential curves can be used in linear combination to approximate the desired decaying system response. In implementing the calculation, a combination of discrete and distributed feedbacks are found in the desired dynamical system. That leads to a delay~~ed~~ differential equation or DDE. The characteristic equation ~~for~~ of the ~~delayed differential~~ DDE ~~equation~~ has infinitely many roots, and these roots give us exponential functions that can be used as a basis to fit the response ~~in question~~ of interest. The fit can be done using either 2 norm, infinity norm or any other criterion of interest to the analyst. When we increase the number of terms used in the fit, the quality of the fit improves. The primary advantage of this approach is that the underlined dynamical system and its characteristic equation give us an infinite sequence of roots, so choosing the exponential rates in the expansion

is easy. In contrast, if we pose the problem as one where both the coefficient as well as the exponent have to be fitted ~~and are a part of the~~ as design parameters, then each time we increase the number of functions involved in the fit the whole problem, including finding the exponential roots, ~~have~~ has to be solved afresh.

The second problem is motivated ~~from~~ by an application in robotics. One of the basic manipulation primitives that a robot uses for mechanical tasks involves pushing a manipulated object on a horizontal plane. For ~~the same~~ this task, there is a generalized load (a combination of force~~s~~ and moment~~s~~) that needs to be applied on the body to cause motion. Given a desired motion, finding the loads constitutes a forward problem, which requires some numerical integrals over the contact patch. Given a loading direction, finding the resulting motion and the load magnitude require solution of nonlinear equations, which is more difficult. A way around this, is to solve the problem using approximated limit surfaces. A limit surface is a boundary of a convex set which consists of all possible static and sliding frictional loads, for a given contact ~~surface~~ condition. Limit surfaces ~~have certain desirable properties that make them~~ are suitable for ~~to~~ use in load-to-motion mapping. Since robots may be required to push an object in a particular direction, a simple mapping from load to incipient motion is used. This mapping uses an approximated limit load surface. Approximations for limit surfaces used have included ellipsoids in the simplest case, and multivariate polynomial expansions as a more sophisticated example. We propose a new approximation which uses a small number of symmetrical 3×3 matrices along with fractional powers of simple quadratic forms. With only one matrix, we obtain the ellipsoid approximation. However, with more matrices we obtain more accurate surfaces that are analytically tractable, ~~for~~, e.g., for subsequent motion planning and control. Fitting these 3×3 matrices is easy with simple optimization algorithms. Several numerical examples show that with 2 or 3 such fitted matrices, the fit obtained is excellent. The new method of approximating limit load surfaces presents a significant useful generalization of the popular but somewhat inaccurate ellipsoid approach.

like to thank but can't ----

(of MERL)

Acknowledgements

I ~~would~~ extend my ~~sincerest~~ gratitude to my project advisor Prof. Anindya Chatterjee for guiding and supporting me in the right direction. I ~~would~~ also ~~like to~~ thank Sankalp Tiwari and Devesh Jha for fruitful discussions and multiple insights on the problem statement and for providing me with relevant information to do the work.

I ~~would like to~~ thank my father Mr. Sudesh Kumar Chauhan, my mother Mrs. Nivedita Chauhan and my brother Surya Pratap Singh for supporting and providing me encouragement throughout my years of study. It would have been difficult to complete ~~the project~~ my studies without their assistance.

I'd also ~~like to~~ thank my batchmates Shubham, Manu, Gaurav, Akshay, Umesh, and Akash for their help and encouragement along this journey.

Vaibhav Pratap Singh
Department of Mechanical Engineering
IIT Kanpur

Contents

List of Figures

Abbreviations

LS	**L**imit **S**urface
COR	**C**enter **O**f **R**otation
DDE	**D**elayed **D**ifferential **E**quation

Symbols

P	Load vector
q	motion vector
F	Force
M	Moment
$H(f)$	Failure surface
f, f_k	Force vector
$J1, J2$	Objective functions
α	Scaling factor
$\hat{v}$	Approximated velocity
$\hat{f}$	Approximated force
f_s	Scaled force

Dedicated to my mother
Mrs. Nivedita Chauhan

Chapter 1

Introduction

responses, the

This study present two different and unconventional approximation schemes for two types of mechanical systems, first involving decaying functions and the second involving frictional sliding of rigid bodies. In this way the report has ~~a~~ two ~~part structure to it.~~ distinct parts.

In Chapter 2, we develop and present a new approach to approximate decaying responses of a dynamical system. We use first the hockey stick function as an example to present an approximation strategy for decaying responses. We establish our method and use the same methodology on ~~more decaying~~ some other functions. We comment on the performance of the method and its shortcomings.

In Chapter 3, a ~~significant~~ different approximation problem is addressed, which is relevant in the field of robotic manipulation. We approximate certain closed convex surfaces known as limit surfaces. We study what these limit surfaces are, how they are generated, and why they must be precisely approximated. We examine our formulation and discuss ~~various~~ some fitting strategies. We report the outcomes of the approximation and make comparisons to the previous work done in this field.

The following is an overview of the two problems being investigated. The subsequent chapters ~~delve deeper into these topics.~~ present details. ~~This report comes to a close with concluding remarks on both the themes.~~

The final chapter presents concluding remarks.

1.1 Approximation for decaying functions

Some systems have a scalar response that starts at a high value and then decays to low values with the passage of time. The usual approach is to fit the response using a linear combination of decaying exponentials. When the number of terms in the sum becomes large, then choosing the exponential rates (which can be complex, i.e., with oscillating components) can be difficult. For instance, fitting exponential rates along with the coefficients would require us to solve the whole problem ~~again to~~ afresh just to increase a few terms in the sum. Using an arbitrary set of exponential rates can lead to ill-conditioned coefficient matrices, or a poor fit, hence we required a well defined basis of exponentials ~~rates~~ to fit the function. ~~Moreover,~~ If posed as an optimization problem, there are many local minima and finding a good solution is not guaranteed in advance for arbitrary numbers of terms in the approximation. Here we take an indirect approach to the problem, by constructing a delayed dynamical system whose possible set of free responses includes the specific function of time in question.

1.2 Approximations for limit surfaces in frictional sliding

The second problem studied in this work is ~~inspired from~~ motivated by an application in robotics which involves frictional sliding motions of objects subjected to forces and moments. To understand this better, consider a rigid body sliding on a planar surface. The only interaction of this body with the surface are the ~~frictional~~ contact forces, which are studied here. The contact normal force or pressure is assumed known. If the motion is known, then computing the required loads is straightforward (although it includes evaluation of integrals). Conversely, if the load direction is known, then predicting the initial motion direction is more difficult because it requires solution of nonlinear equations which ~~involve~~ include evaluation of integrals. But it is ~~observed~~ known that all possible static and sliding frictional loads form a convex set, whose boundary is called the limit surface. ~~And these~~ If the limit surface is known, then solving ~~are a way around solving those~~ nonlinear equations is no longer needed. In robotics, there is interest in simple descriptions of force to motion mapping and vice versa. ~~These limit surfaces have certain properties which can be exploited to our advantage.~~ To develop a load motion mapping an approximated limit surface is used. Simple models based on ellipsoids and polynomials have been developed to approximate these limit surfaces in the past. We ~~attempt to develop~~ present a new scheme to approximate these limit surfaces, which uses a small number of symmetrical 3×3 matrices along with fractional powers of simple quadratic forms.

Chapter 2

Approximation ~~for~~ of decaying functions

Make sure you understand the theory & the code!

In this chapter, we propose a scheme to approximate decaying functions as sum s of exponentials. The basic idea is to construct a delayed dynamical system such that one of the responses of the system corresponds to the function that is to be approximated. We develop a scheme to ~~setup~~ set up the aforementioned system using convolution and construct a response that closely approximates the target decaying function. Later ~~on~~ we study how the approximation performs with increasing terms and with different functions.

The next five pages present the basic idea behind the approximation method. These pages draw heavily from an unpublished report written by my thesis advisor Anindya Chatterjee. I have fully reproduced the calculations in that report and then applied that methodology to two different decaying functions. Some shortcomings of the method seen in ~~those~~ the new examples will be discussed at the end of the chapter.

2.1 A dynamical model

We wish to approximate a function $x(t)$ (for $t > 0$) that eventually decays to zero as t increases, using a linear combination of decaying exponentials. We further wish to use the infinity-norm in our approximation. For demonstration of ideas, we use the hockey stick function (Figure 2.1), defined as $x(t) = 1 - t$ for $0 \leq t \leq 1$, and $x(t) = 0$ otherwise.

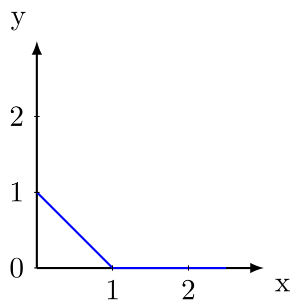


FIGURE 2.1: Hockey stick function

I have forgotten how I did this.

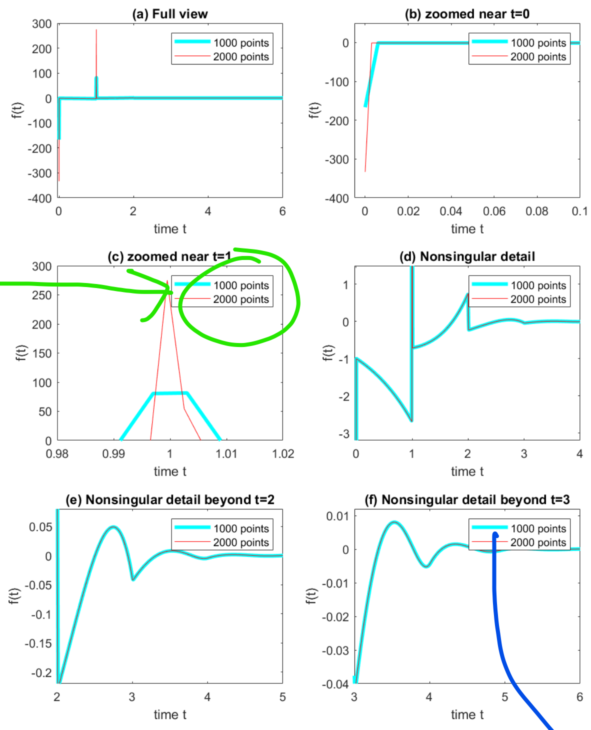
Say something about solution?

In principle, every linear combination of exponentials is the solution to some linear constant coefficient dynamic system. A fairly general form of the same may be expressed using a convolution integral as follows:

$$\dot{x}(t) = -\int_0^t f(\tau)x(t-\tau)\,dx \tag{2.1}$$

We discretize the above equation and find f using a system of simultaneous linear equations. For the hockey stick function, using 2000 points uniformly spaced on $[0, 6]$, we obtain one approximation for f. We do the same for 1000 points as well. Results are shown in Figure 2.2.

match font size, ideally



bigger

adjust location (drag in Matlab)

FIGURE 2.2: First view of $f(t)$

2.2 Characterization of f(t)

It is clear from Figure 2.2 that there are Dirac delta functions at $t = 0$ and $t = 1$. Numerical estimates of their strengths suggest both are of unit magnitude. The rest of $f(t)$ can be well approximated by piece-wise polynomials, on the intervals $(0, 1), (1, 2), (2, 3)$ and so on. so on.
We choose to ignore the nonzero values of f beyond some large enough t (6 in our case). Six such polynomial plots (each of fifth order) are shown in Figure 2.3 (thick cyan: actual value; thin dotted red: polynomial fit).

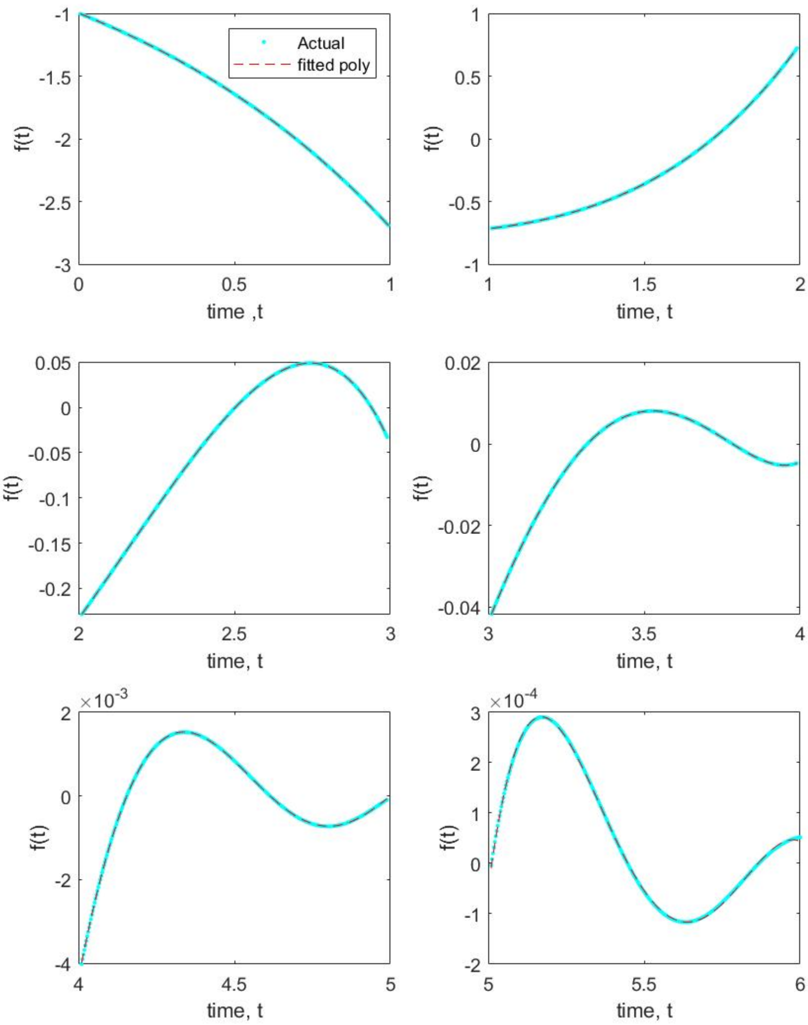


FIGURE 2.3: Piecewise polynomial approximation for $f(t)$

did you retain more digits in the actual calculation? Or is this what was used for plotting?

The following are the polynomials that are used to approximate the function over the interval $[0, 6]$. Here $p_1(t)$ valid for $(0, 1]$, ~~and so on~~. $p_2(t)$ is valid for $(1,2]$ and so on.

$$
\begin{aligned}
p_1(t) &= -0.0137t^5 - 0.0345t^4 - 0.0345t^3 - 0.0345t^2 - 0.03451t - 0.9970 \\
p_2(t) &= 0.0657t^5 - 0.2321t^4 + 0.6695t^3 - 0.6433t^2 + 0.1583t - 0.7332 \\
p_3(t) &= -0.1230t^5 + 1.1803t^4 - 4.7502t^3 + 10.1286t^2 - 10.9627t + 4.2296 \\
p_4(t) &= 0.1091t^5 - 1.7157t^4 + 10.7465t^3 - 33.6881t^2 + 53.2784t - 34.3779 \\
p_5(t) &= -0.0411t^5 + 0.9092t^4 - 7.9743t^3 + 34.6855t^2 - 74.7489t + 63.7709 \\
p_6(t) &= 0.0010t^5 - 0.0407t^4 + 0.5696t^3 - 3.8203t^2 + 12.4353t - 15.8333
\end{aligned}
\tag{2.2}
$$

Now we have established our delay differential equation.

2.3 Delay differential equation

Our system is formed by the dirac delta functions leading to discrete delayed feedback, while the rest of f leading to distributed delayed feedback through integrals, given by eqn (2.3).

$$
\dot{x}(t) = -x(t) + x(t-1) + \int_0^1 p_1(\tau)x(t-\tau)d\tau + ... + \int_5^6 p_6(\tau)x(t-\tau)d\tau \tag{2.3}
$$

The characteristic roots of the above DDE give us a choice of exponential rates to use for the problem of approximating the original function $x(t)$. Let us Lets refer to them as basis 1 (separate) basis1 for the approximation. Now we insert $x(t) = e^{\lambda t}$ in the DDE, carry out the integrations, and obtain the characteristic equation. We solve this equation in Maple. The characteristic equation obtained is

do you? or do you just OBTAIN it?

$$
\begin{aligned}
\lambda^7 = (&\lambda^6+1.991000\lambda^5+2.982700\lambda^4+3.956800\lambda^3+5.060400\lambda^2+4.797600\lambda+9.540000)e^{-1\lambda} \\
&+(-0.132000+0.184800\lambda-0.084100\lambda^5-0.070500\lambda^4-0.034400\lambda^3+0.067200\lambda^2)e^{-6\lambda} \\
&+(5.076000+2.582400\lambda+0.154700\lambda^5+0.154300\lambda^4+0.085000\lambda^3+0.725400\lambda^2)e^{-5\lambda} \\
&+(-18.036000-0.019500\lambda^4-0.935600\lambda^3-4.612800\lambda^2-9.144000\lambda-0.018600\lambda^5)e^{-4\lambda} \\
&+(27.852000-0.016800\lambda^5+0.961700\lambda^4+3.851200\lambda^3+9.787800\lambda^2+14.047200\lambda)e^{-3\lambda} \\
&+(-22.656000-0.991100\lambda^5-2.974400\lambda^4-5.901400\lambda^3-10.030800\lambda^2-11.412000\lambda)e^{-2\lambda} \\
&-1.644000-0.997000\lambda^5-0.995600\lambda^4-0.992000\lambda^3-1.015200\lambda^2-1.000000\lambda^6-0.828000\lambda
\end{aligned}
\tag{2.4}
$$

The above equation is now solved for λ. It is a transcendental equation with infinitely many roots. But the larger roots of such equation usually follow some discernible pattern, so numerically finding several of them is not really difficult. In particular, eventually the real parts change slowly while the imaginary parts are incremented by near-constant amounts. Newton-Raphson can be used to find the roots of this equation, with some manipulations of initial guesses. We use Maple's *Root-Finding* to solve this equation, the first 62 numerical determined roots are shown the Figure 2.4.

Split into real & imag. parts?

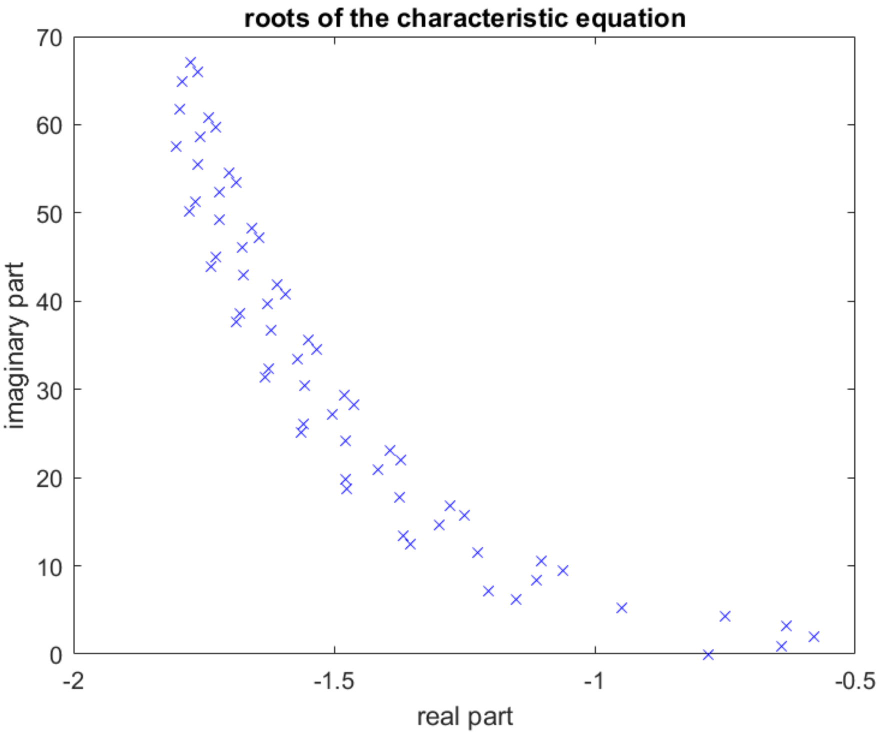
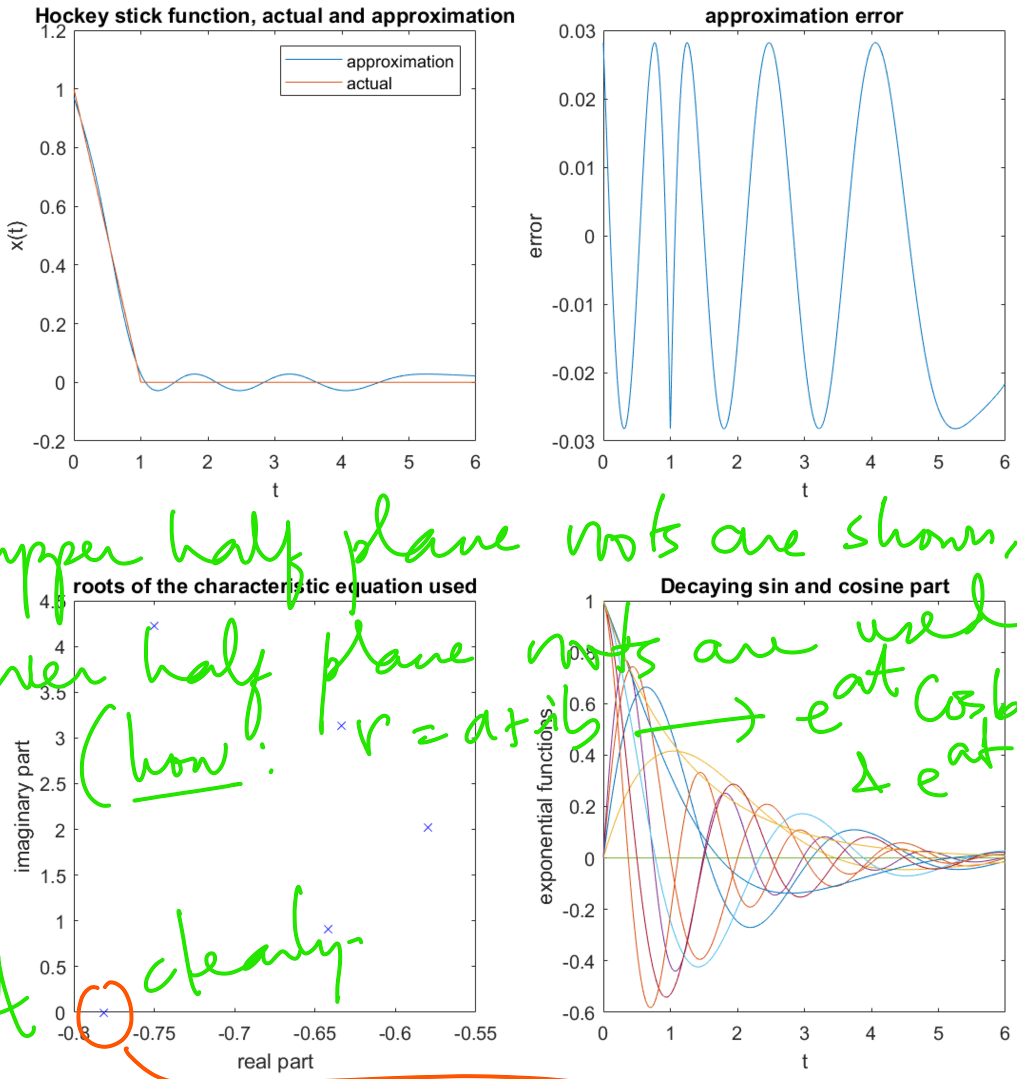


FIGURE 2.4: Roots of characteristic equation

More description

2.4 Least infinity-norm ~~solution~~ error

If we have an over determined system $Ax = b$, then the least squares (or minimum-error in 2-norm) solution is easy, and given by MATLAB in response to simply $A\backslash b$. It is less easy, but still standard, to find the solution x that minimizes $\|Ax - b\|_\infty$. We use MATLAB's linprog to do this. Using the infinity-norm minimizing solver, results obtained using 5 roots (i.e., 10 including complex conjugates) are plotted in Figure 2.5. The results obtained using 8 roots are plotted in Figure 2.6. The results obtained using 62 roots are plotted in Figure 2.7.

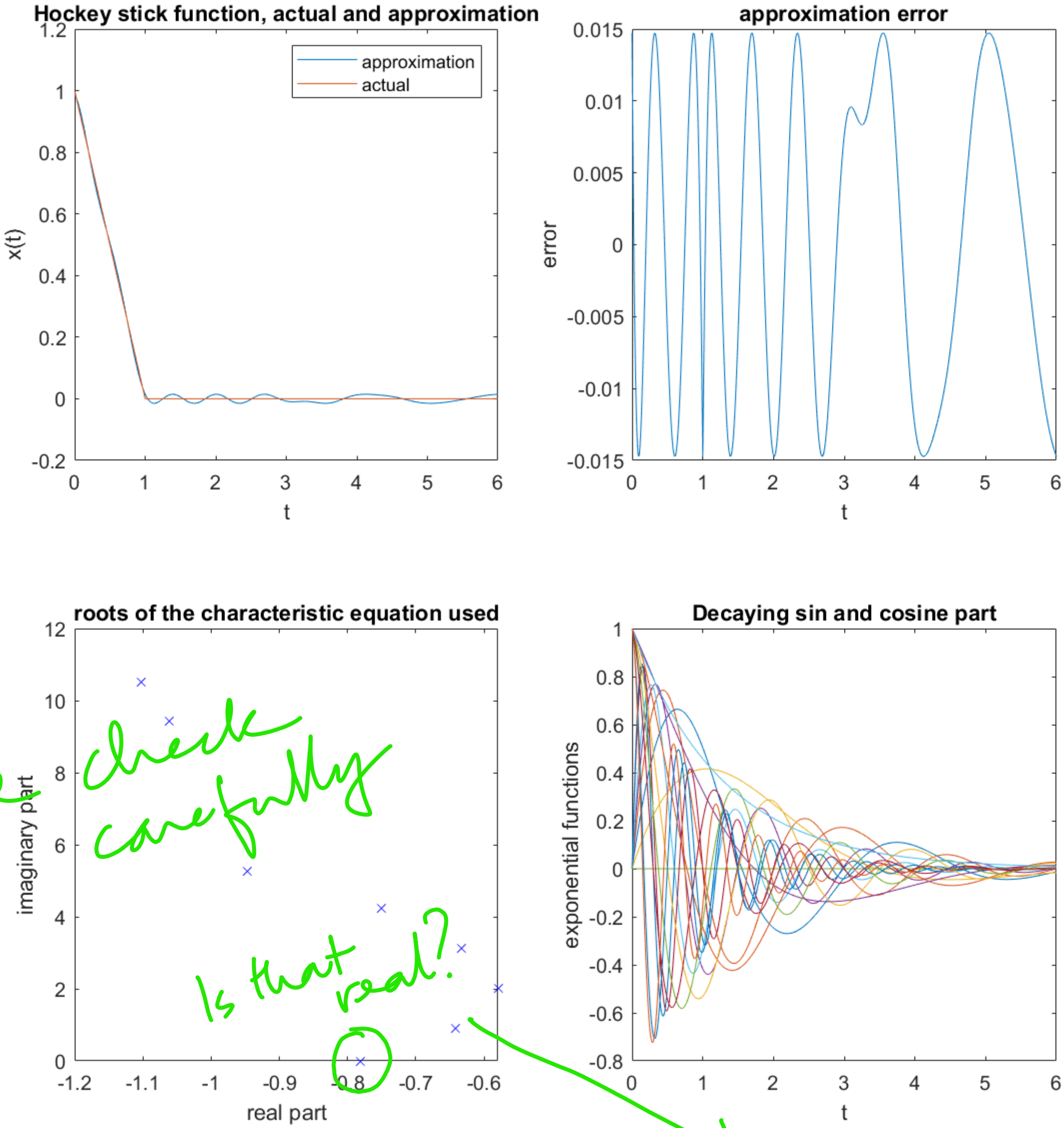


Only upper half plane roots are shown, but corresponding lower half plane roots are used as well (how: $r = a + ib \rightarrow e^{at}\cos bt$ & $e^{at}\sin bt$ both used)

SAY it clearly.

FIGURE 2.5: Approximation obtained using 5×2 roots

real root?

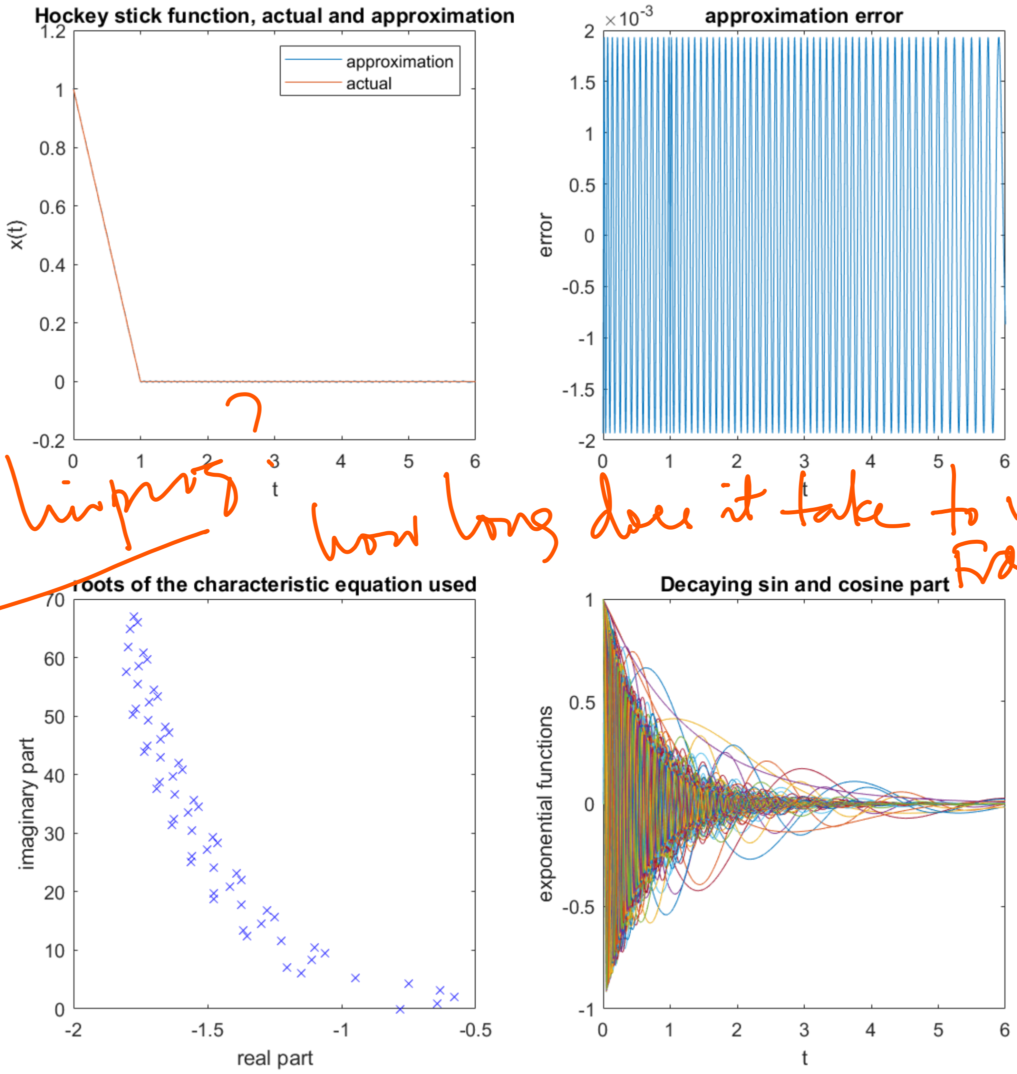


Please check carefully

Is that real?

Then

FIGURE 2.6: Approximation obtained using 8×2 roots



This is nice

Basic linspace?

How long does it take to run? Fraction of a second, or longer?

FIGURE 2.7: Approximation obtained using 62 × 2 roots

The preceding graphs show that as the number of terms increase, the quality of the fit improves. Because the roots are complex, oscillations in the error are expected, but the amplitude of these oscillations decrease s with increase in terms. High frequency sin e and cosine parts are introduced into the system as terms increase, contributing to the system's fast varying parts (sharp transition at $t = 1$), giving better results.

← recall the solution components

Now that we have established our method, we will now reiterate the method on more functions and look ~~the transitions~~ in the next section.

at its performance

apply

2.5 More functions

We will

bases.

We repeat the process for a few more functions, yielding a collection of ~~basis~~. We'll approximate the same functions using these alternative bases to see the changes in approximation and transitions. The functions are approximated from [0,6] and projected until t = 19 to observe the transitions.

f_1 etc, convert it everywhere

u = space

To form the basis, we use the following functions. Let us refer to them as f1, f2, and f3 and the basis obtained from them as basis1, basis2, and basis3 respectively. Furthermore, we also use a simple basis (basis0) obtained from the ~~corresponding~~ DDE: $\dot{x}(t) = -0.1x(t-6)$.

simple

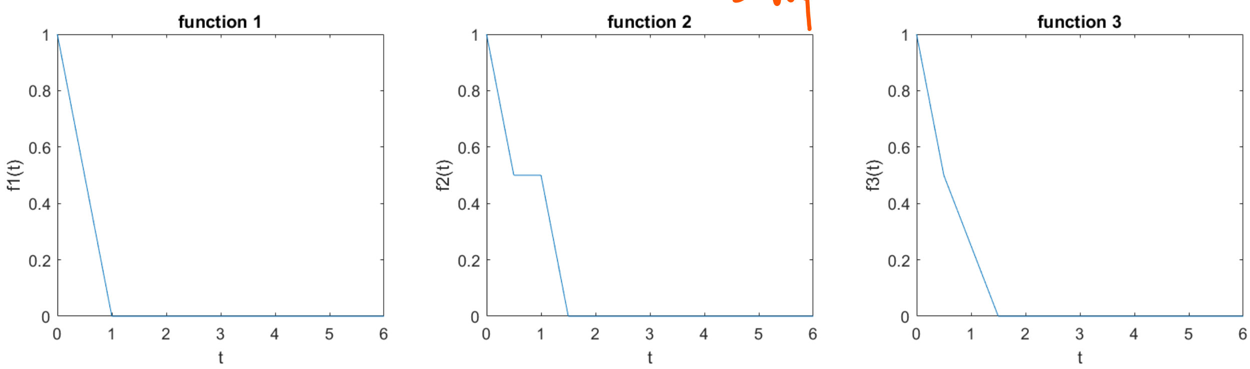
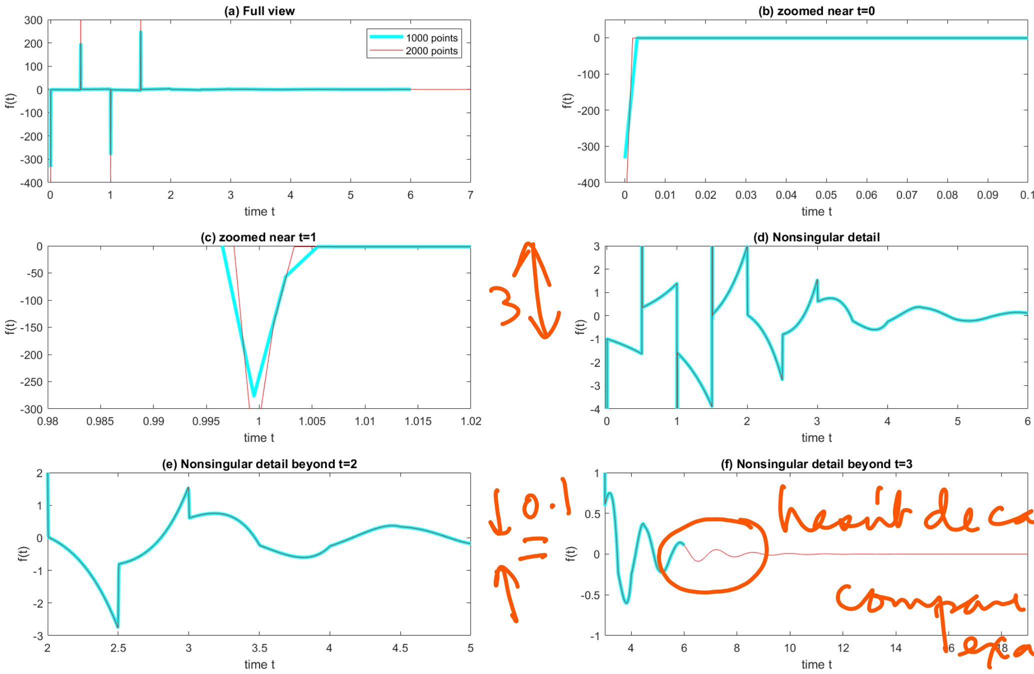


FIGURE 2.8: Function to be approximated

were

Since f1 is the hockey stick function, calculations for it ~~are~~ carried out in the previous sections. ~~so~~ Here we will focus on the other two functions. ~~Reiterating~~ Repeating the process for the functions, we find four dirac-delta functions in f2 and three in f3 as seen in Figures 2.9 and 2.10 respectively. These contribute to the discrete delayed feedbacks and the rest of the $f(t)$ is approximated by piecewise polynomials on the intervals $(0, 0.5), (0.5, 1)$ and so on, giving the distributed delayed feedback through integrals. Fifth order polynomials are used to approximate these, which are shown in Figure 2.11 and 2.12 for f2 and f3 respectively (cyan: actual value; dotted red: polynomial fit).

Now that polynomials are constructed and dirac-delta functions are known, corresponding DDEs for the respectively functions are established. Figure 2.13 shows the roots obtained after setting up the respective DDEs for the functions.



3

0.1

hasn't decayed yet, compare w/ first example

FIGURE 2.9: First view of f for f2. For uniformity of treatment with f1, a maximum delay of 6 is used here as well.

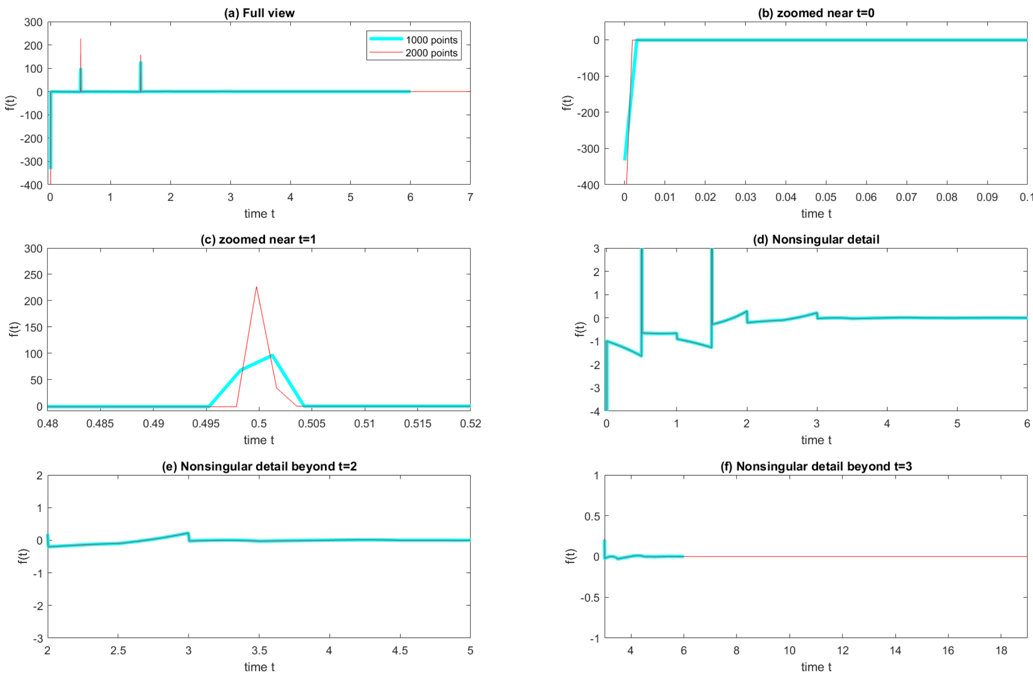


FIGURE 2.10: First view of f for f3. Here the decay seems more satisfactory.

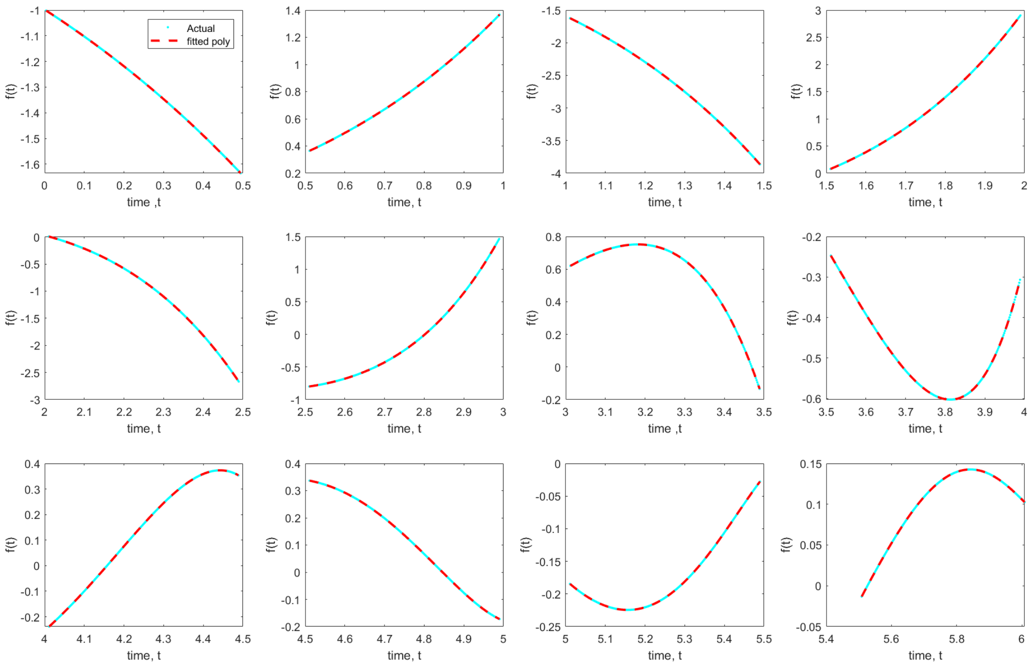


FIGURE 2.11: Piecewise polynomial approximation for f(t) in f2

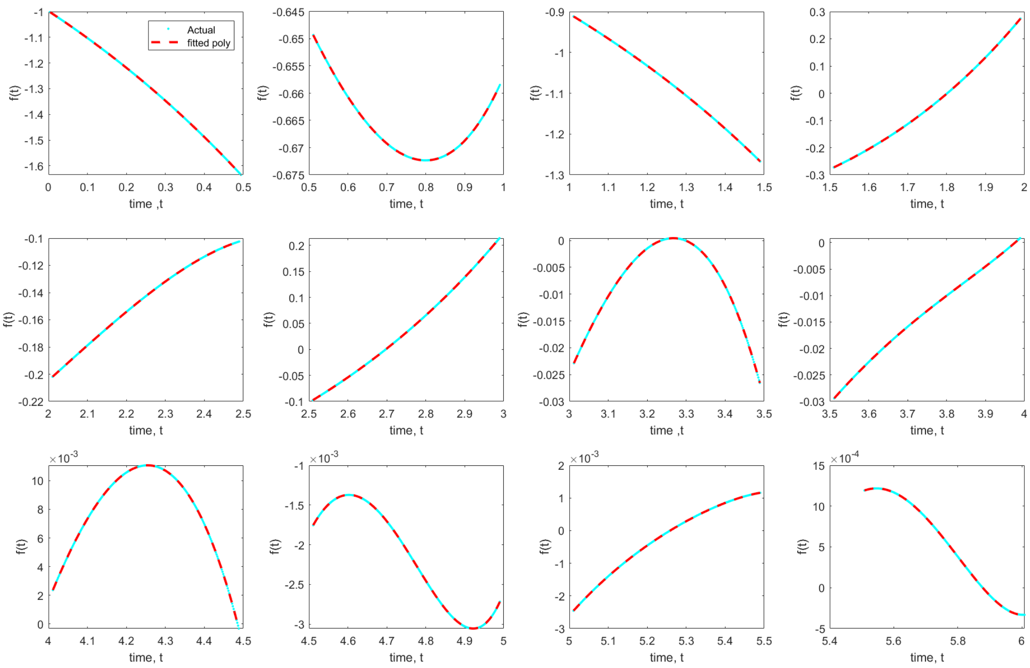
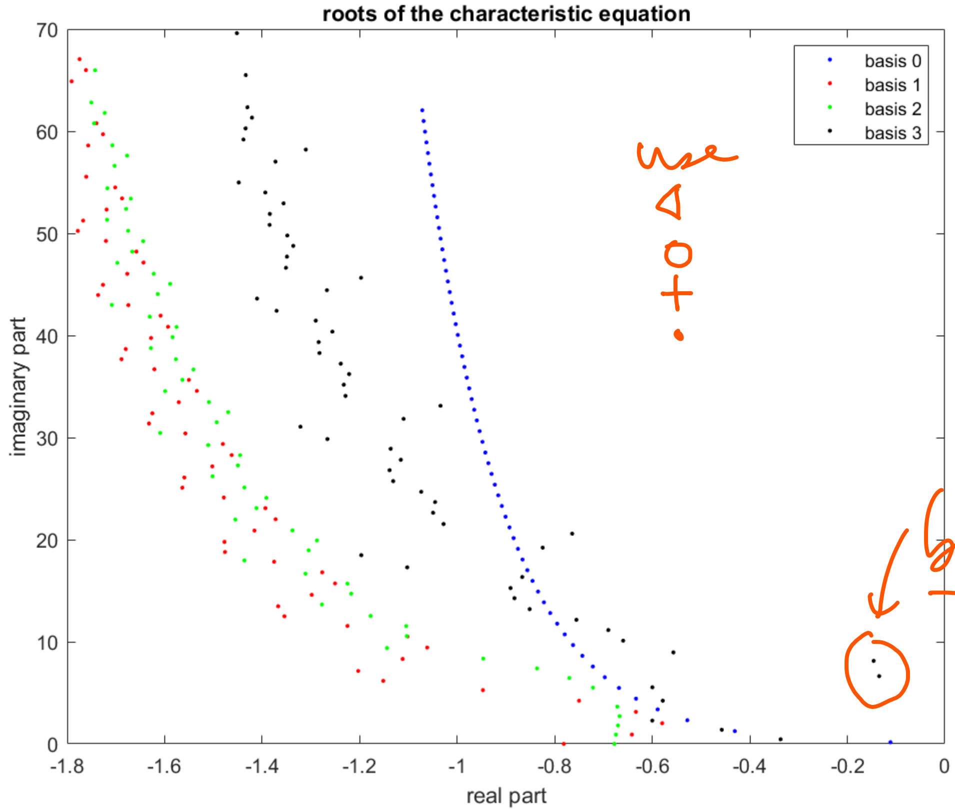


FIGURE 2.12: Piecewise polynomial approximation for f(t) in f3

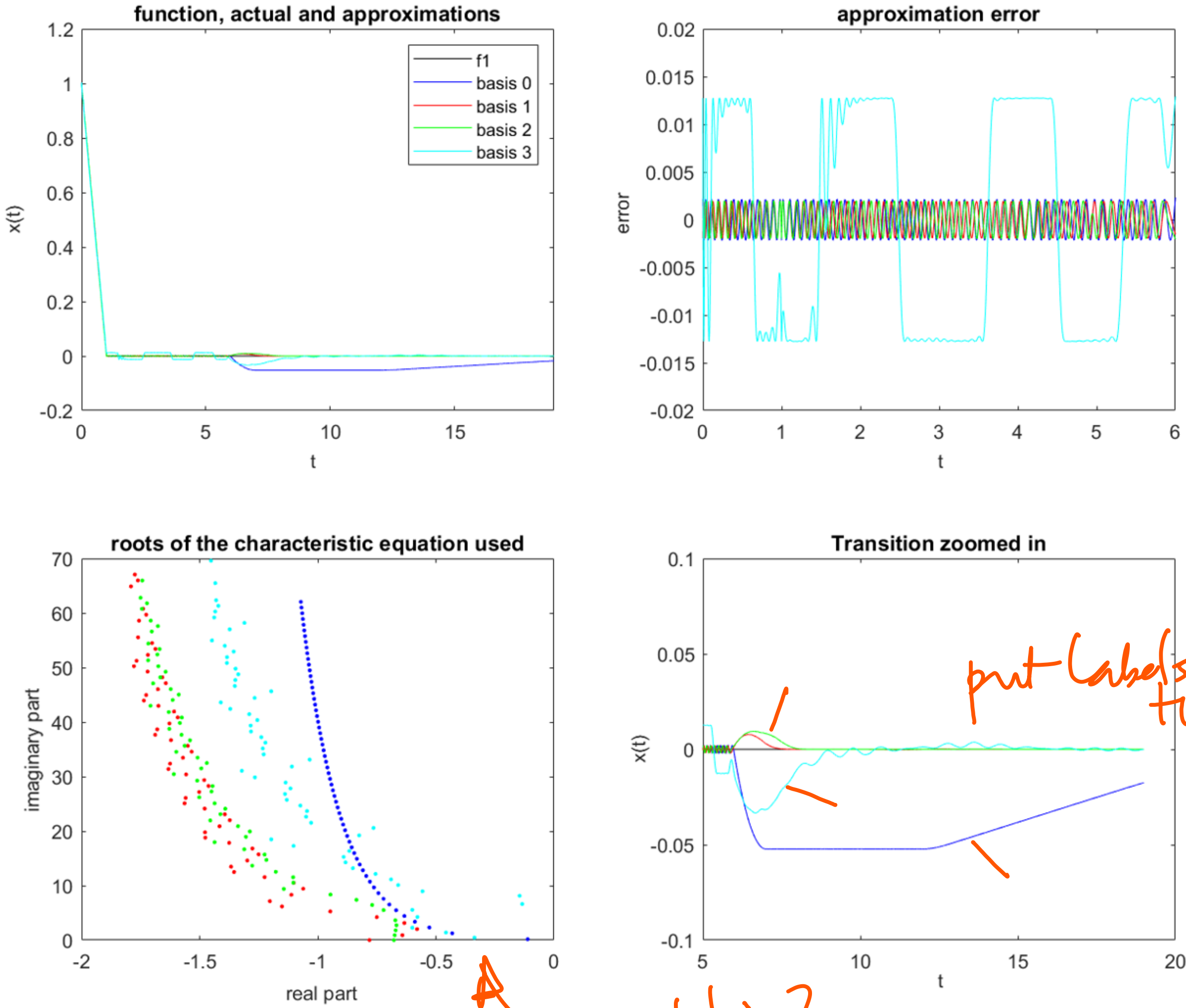


use △ o + •

basis 3?

FIGURE 2.13: Roots of respective characteristic equations

We approximate the above functions individually using all of the bases to obtain the best fit and observe the ~~transitions~~ performance. We use infinity-norm to fit the functions with first 60 roots of each characteristic equation. Figure 2.14 depicts approximation of f1. Figures 2.15 and 2.16 depict the approximation of f2 and f3, respectively.



put Labels like this

repetition?

FIGURE 2.14: Approximation of function 1 using different basis

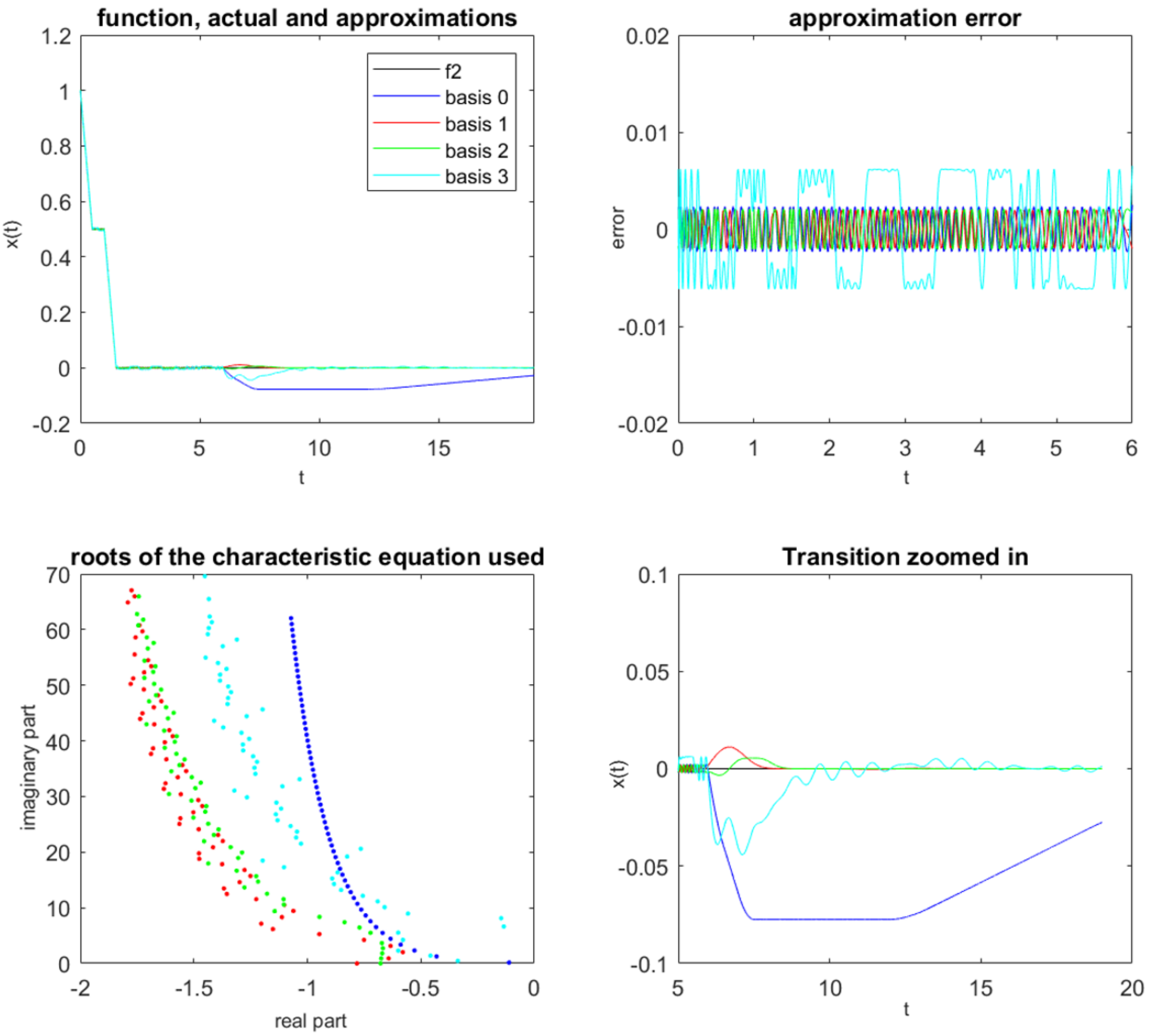


FIGURE 2.15: Approximation of function 2 using different basis

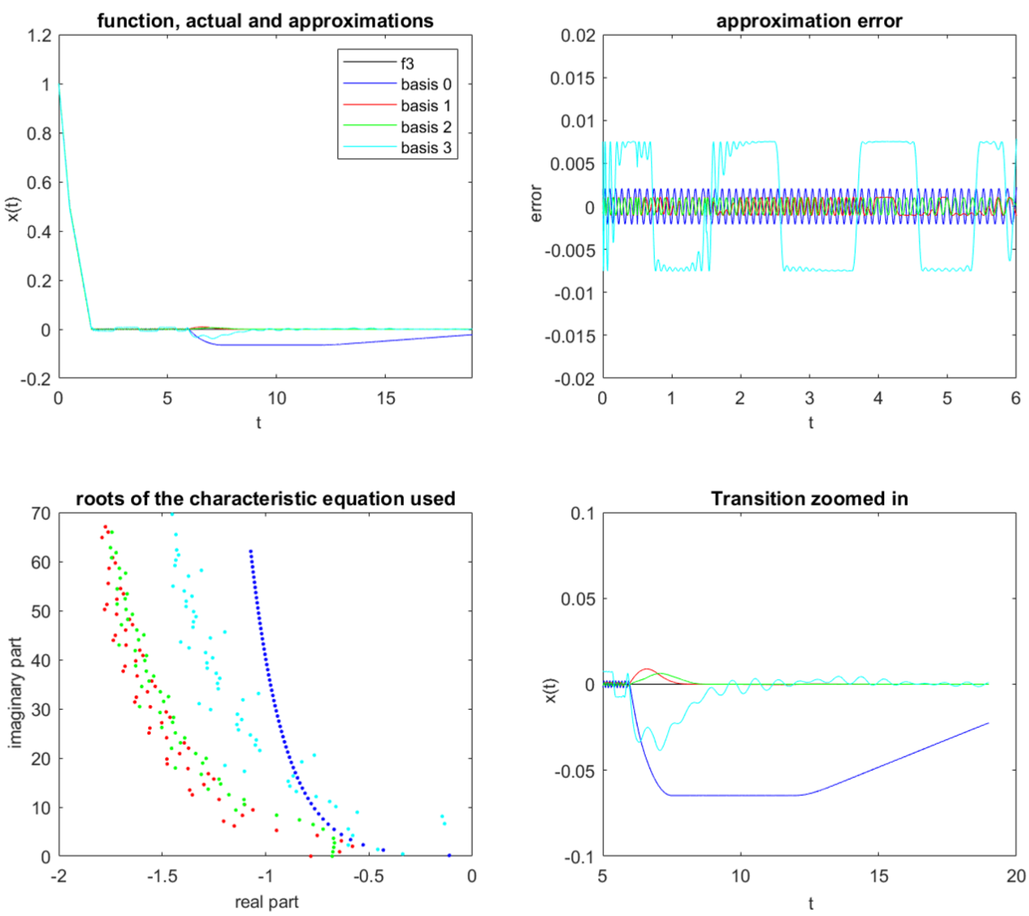


FIGURE 2.16: Approximation of function 3 using different basis

The basis3 (cyan) struggles to approximate any of the functions since it has two roots with slow decaying components, whereas the other functions produce an excellent fit until the transition, as shown in the preceding figures. Because we are fitting our exponential functions till a finite distance ($t = 6$), we see a large response with slow decay after that. Since the exponents were only fitted to match the response up to a certain time, this is to be expected. After $t = 6$, the basis0 has a large response which eventually decreases whereas the other basis have relatively smaller responses decaying eventually. The equal amplitudes in the error with time is due to the fact that we are using infinity norm as our criteria for the fit. Had we used least square fit the error amplitude would vary with time.

To summarise, the approach works well for functions with faster decaying characteristic roots. The method offers excellent fit up-to a finite distance beyond which we get an undesired response which eventually decays.

Future work may investigate the role of the few right-most roots (slowest decaying exponentials) as well as the length of the interval used (here, T=6).

Ha ha. Just realized your writeup has no references! Cite the minimum references at appropriate places - like for characteristic roots of DDEs earlier, and Goyal et al & Mason for limit surfaces! Mention Dr. Devesh Jha here, too.

Chapter 3

Approximation ~~for~~ Of limit surfaces for frictional sliding

+ The motion itself may involve some combination of translation and rotation.

This chapter focuses on the report's second problem, which is motivated from an application in robotics. One of the basic manipulation tasks a robot can perform is pushing an object kept on a horizontal plane. ~~For the same, a~~ A combination of a force~~s~~ and a moment~~s~~ is required to overcome the frictional loads to cause motion. + For a particular generalized motion direction, these generalized frictional loads vectors can be calculated by simply summing up the frictional forces at the contact points or calculating integrals over the contact patch. But estimating the motion direction for a given frictional load requires solutions to ~~some~~ non-linear equations involving integrals, which is more difficult. Interestingly, all the possible configurations of these frictional loads for a given contact surface form a convex set, and the boundary of this set is called the limit surface. Need to put references here. These limit surfaces have ~~very~~ some desirable properties, which are utilized to develop ~~models for~~ load-motion mapping. Since robots may be forced to push objects in a specific direction due to some obstacles or mechanical limitations, these models may be used in path and motion planning decisions. These models require an approximated limit surface. Simple ~~models~~ approximations have been developed in the past ~~to approximated~~ for these limit surfaces, a brief overview of which is presented in 3.1. In this chapter, we develop our own ~~model to~~ approximation for these limit surfaces which is more accurate than those developed in the past.

We start by reviewing some of the past work in this field. In subsequent sections, we define and study these limit surfaces, develop a method to generate them and show several examples of LS. We ~~further establish~~ then present our method to approximate them, ~~analyze~~ study its performance and compare it to previous work.

don't abbreviate.

3.1 Related work

Oh. Some lit. review present here. Look at the earlier chapter though.

Wrenches have force and moment parallel, in 2D case moment = 0 & line of action is to be found. You have done something else.

In this section, we review some literature which is closest to our proposed work . One of the earliest ~~work~~ papers that presented modeling of limit surfaces was presented in [1]. Our work is mostly inspired from the analysis presented in [1]. ~~In [1]~~ Those authors show that all possible static and sliding ~~frictional wrenches~~ Load vectors (generalized) form a convex set whose boundary is called the limit surface.

In [2], authors show that the limit surface can be approximated by a three-dimensional ellipsoid. These ellipsoids are fitted by calculating the maximum friction force and maximum moment, and then used to calculate the major and minor axis as well as the tilt angles of the ellipsoid. This model was proposed in 1996, and was designed to reduce the computational load required to generate an approximate limit surface. Now, with ~~advancements in technology~~ easier computing, more complex models can be constructed giving more accurate approximations.

More recently, reference [3] presented a framework of representing planar sliding force-motion models using homogeneous even-degree sum-of-squares (sos) convex polynomials. These can be identified by solving a semi-definite program, and the set of applied wrenches can be obtained by 1-sublevel set of a convex polynomial [3].

The work presented in [1] shows an elliptical ellipsoidal (does it, in fact? or is it [2]?) approximation of the limit surface which is computationally simple to evaluate. Consequently, this model has been widely used for various manipulation tasks [4, 5, 6, 7, 8, 9].

3.2 Analytical model for pusher slider system

To better understand the relationship of frictional loads and motion, we establish an analytical model for the pusher slider system. We specific some notations and the configurations of the slider, which will be held constant throughout the study.

Consider the case of a rigid body sliding on a plane. This body's only interaction with the surface is ~~frictional~~ through contact forces. The normal force or pressure ~~or~~, and coefficient of friction at the point of contact or surface ~~is~~, are assumed known. The magnitude of frictional force at each location is determined solely by the orientation and direction of slipping of the body, not by the ~~magnitude~~ velocity of slipping.

A reference point C is chosen as our body's geometrical center and positioned at the origin for ~~our~~ convenience. A unit motion vector **q** has ~~the~~ components which are the reference point's translation for math quantities in your text, use $. $

Discrete forces ⇒ sums. Distributed forces ⇒ integrals w/ dA not dx or dy.

velocity and the angular velocity of the ~~reference point~~ body, which we write as $\mathbf{q} = [q_x, q_y, \omega]$. The velocities are normalized ~~non-dimensionalized~~ by a ~~the~~ characteristic length of the body. The net frictional load is $\mathbf{P}$=$[F_x, F_y, M]$ where F_x and F_y are the net forces and M is the moment about the z ~~z~~ axis. The forces are also normalized ~~non-dimensionalized~~ by multiplication with the characteristic length. $\mathbf{P}$ is the required external load to overcome the frictional forces to cause motion. These forces and moment can be calculated as integrals over the contact patch. Let $f = [f_{ax}, f_{ay}]$ be the frictional force at a contact point A on the patch with coordinates as $r_a = [x_a, y_a]$. The net frictional forces as be calculated as

(Move later)

not correct. Fix this.

$$F_x = \int_A f_{ax} dx \quad F_y = \int_A f_{ay} dy \quad M = \int_A x_a f_{ax} - y_a f_{ay} dA \tag{3.1}$$

And when we deal with surface with discrete contact points instead of a patch, the integrals change to summations over all the contact points, and rest stays the same.

All the possible combinations of these frictional load $\mathbf{P}$ over the contact patch, form a convex set. The boundary of this set is called the limit surface. The limit surface has the following properties-

- if $\mathbf{P}$ is inside the limit surface, then $\mathbf{q} = 0$
- $\mathbf{q}$ is normal to the limit surface, given the surface is smooth and unique normal exists
- At the edges of the limit surface, $\mathbf{q}$ is non-unique for a particular $\mathbf{P}$
- At the facets (flat patches), the $\mathbf{q}$ is same for all $\mathbf{P}$ lying on the facet

Authors in [1] provide a formal proof of the existence of limit surfaces as well as their properties. Since we are only interested in approximating these surface, We won't elaborate on the formal proofs.

To construct these limit surfaces, all we need is set of all possible frictional loads $\mathbf{P}$. Since, for a given motion, load calculation constitutes a forward problem and easy to compute, we use this to our advantage. We consider a large number of motion vectors $\mathbf{q}$ uniformly distributed on the surface of a sphere (we used 2000 such vectors) and calculate the corresponding $\mathbf{P}$ in load space for each $\mathbf{q}$, using equation 3.1.

With this method we obtain a set of possible frictional loads $\mathbf{P}$, which are a part of the limit surface. These loads can be plotted in load space (F_x and F_y on x and y axis respectively, with M on the z axis) to analyse. In the next section we will look at these surfaces in depth for a few cases and prepare our fitting data for the approximation.

3.3 Limit surfaces and data generation

We shall look at these limit surfaces for some trivial cases. Lets consider an arbitrary body with a single contact point at $(1, 0)$, the calculated the limit surface (a curve in this case) is plotted in Figure 3.1 in load space.

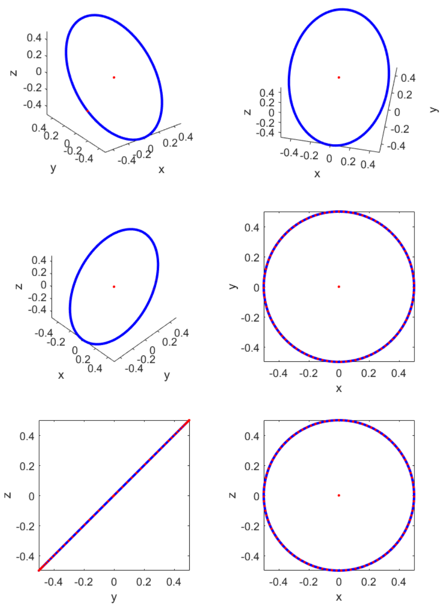


FIGURE 3.1: Limit curve for a single contact point

Consider a two point contact case (a rigid bar with 2 contact point at the ends) as shown in Figure 3.2. The limit surface calculated is plotted in Figure 3.3.

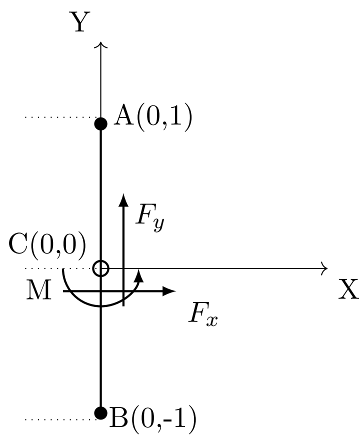


FIGURE 3.2: A rigid bar supported at the ends

Figure 3.3 shows four red circles on the limit curve; these circles form in a specific case when one of the contact points becomes the COR, about which frictional forces cannot be calculated. As long as the frictional force at that contact point is less than the allowable frictional force, the motion will continue. As a result, the surface has flat patches or facets, and it can be concluded that the number of facets on the surface equals two times the number of contact points. This is in agreement with the property stated earlier that is, on the facets there is a non-uniqueness in $\mathbf{P}$ and the $\mathbf{q}$ is unique throughout it.

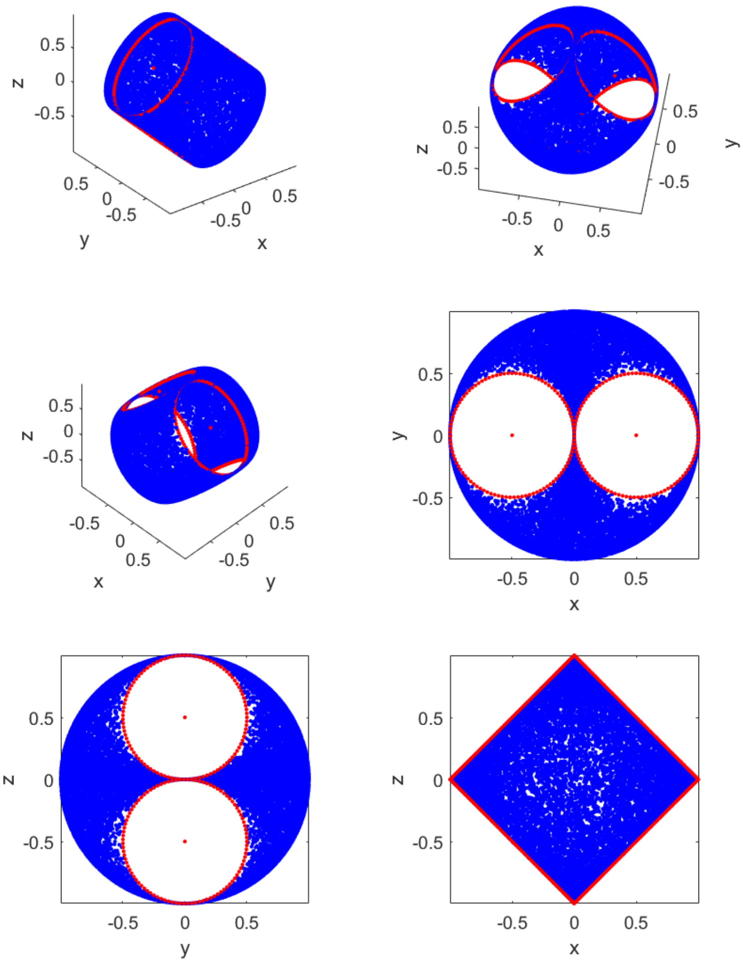


FIGURE 3.3: Limit surface for a rigid bar in Figure(3.2)

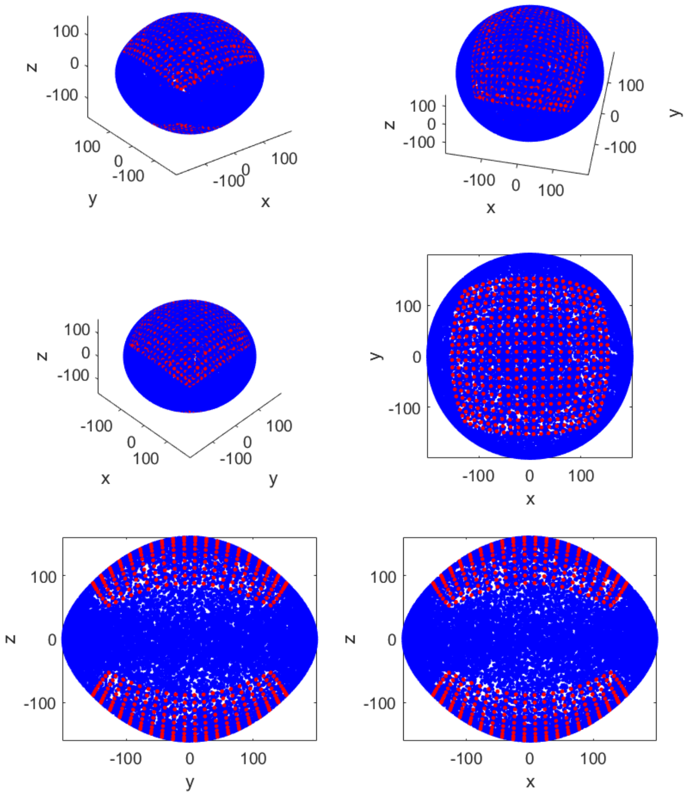


FIGURE 3.4: Limit surface for a continuous square patch

In the case of continuous contact patches (we took 400 uniformly distributed points to discretize the surface), these facets vanish, resulting in a continuous limit surface, as seen in Figure 3.4. Figure 3.4 shows the limit surface for a continuous square patch of length 2 and centred at the origin.

This data of **P** and **q** can now be utilised as fitting data for our approximation, but it won't offer us good facets estimates, when dealing with the case of discrete contact points because none of our data points are on the facets. To address this problem, we calculate a few additional data points, focusing on the facets.
We consider each contact point to be the COR, and calculate velocities and forces on the other contact points while uniformly applying random forces ranging from zero to the maximum allowable frictional force in every direction at the COR. We get our fitting data by combining these data points with the original set. Now, with fitting data set prepared we can move onto our approximation method.

3.4 Approximation method

We want to approximate the limit surface as seen in Figures 3.3 and 3.4 such that for every frictional load **P** we can find the corresponding incipient motion **q** and vice versa using our formulation. We prepare the fitting data set of **P** and **q** as described in previous section.

We define a failure surface $H_i(f)$, (subscript i denotes the number of terms in approximation) as follows-

$$H_1 = f^T A f \quad \text{1 term app} \tag{3.2}$$

$$H_2 = f^T A f + (f^T B f)^{1/2} \quad \text{2 term app} \tag{3.3}$$

$$H_3 = f^T A f + (f^T B f)^{2/3} + (f^T C f)^{1/3} \quad \text{3 term app} \tag{3.4}$$

The failure surface adheres to the following-

- $H(f) < 1$, corresponds to f lying inside the limit surface, implying $v = 0$.
- $H(f) = 1$ implies f lies on the limit surface
- The corresponding velocity is given by the normalized unit gradient, $v = \frac{\nabla H(f)}{\|\nabla H(f)\|}$.

Here f and v are the load and the motion vectors respectively. A, B, C are 3×3 matrices. We define A, B, C such that they are symmetrical and positive semi definite, via the cholesky decomposition. $H(f)$ also obeys the following properties-

- Symmetry: $H(f) = H(-f)$
- Scalability: α is a scaling parameter, that can be computed such that $H(\alpha f) = 1$, for any f
- Invertibility: We can compute a f numerical such that $\frac{\nabla H(f)}{\|\nabla H(f)\|} = v$, for a given v

We chose the following objective functions to get a fit for the limit surface. We fit out failure surface using two types of objective functions **J1** and **J2**. The following are the objective functions with increasing terms in approximation. We will consider upto 4 terms in our analysis, the subscript denotes the number of terms in the failure surface.

$$J1_1 = \sum_{k=1}^{n} (f_k^T A f_k - 1)^2 \tag{3.5}$$

$$J1_2 = \sum_{k=1}^{n} (f_k^T A f_k + (f_k^T B f_k)^{1/2} - 1)^2 \tag{3.6}$$

$$J1_3 = \sum_{k=1}^{n} (f_k^T A f_k + (f_k^T B f_k)^{2/3} + (f_k^T C f_k)^{1/3} - 1)^2 \tag{3.7}$$

$$J2_1 = \sum_{k=1}^{n} (f_k^T A f_k - 1)^2 + \|v_k - \hat{v}\|^2 \tag{3.8}$$

$$J2_2 = \sum_{k=1}^{n} (f_k^T A f_k + (f_k^T B f_k)^{1/2} - 1)^2 + \|v_k - \hat{v}\|^2 \tag{3.9}$$

$$J2_3 = \sum_{k=1}^{n} (f_k^T A f_k + (f_k^T B f_k)^{2/3} + (f_k^T C f_k)^{1/3} - 1)^2 + \|v_k - \hat{v}\|^2 \tag{3.10}$$

Here f_k and v_k is are the load and motion vector from the fitting data, $\hat{v}$ is the approximated motion vector and A, B, C are matrices that are found by minimizing the objective function J wrt to parameters in matrices A, B, C using simple optimization strategy. We use MATLAB's inbuilt solver $fminunc$ to optimize the parameters of the matrices. $fminunc$ uses the BFGS Quasi-Newton method with a cubic line search procedure for optimization.

Now that A, B, C are known, we want to construct the limit surface with our formulation. For the same, We chose a generalized load and scale it such that it lies on the limit surface, i.e:- we enforce $H(f) = 1$. let f_k be the generalized load and let α be a scaling parameter such that αf_k causes slip, i.e:- αf_k resides on the limit surface. Therefore αf_k must satisfy $H(\alpha f_k) = 1$ -

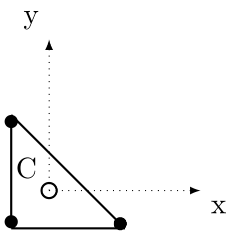
$$\alpha^2 f_k^T A f_k = 1 \quad \text{1 term app} \tag{3.11}$$

$$\alpha^2 f_k^T A f_k + (\alpha^2 f_k^T B f_k)^{1/2} = 1 \quad \text{2 term app} \tag{3.12}$$

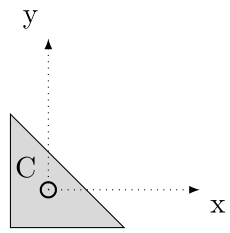
$$\alpha^2 f_k^T A f_k + (\alpha^2 f_k^T B f_k)^{2/3} + (\alpha^2 f_k^T C f_k)^{1/3} = 1 \quad \text{3 term app} \tag{3.13}$$

The above equations are simple polynomial equations in α and can be easily solved. With the aforementioned method we can construct the limit surface once the matrices are fitted. Moreover, we can determine forces that cause slip in any arbitrary direction by considering a unit vector in that direction and scaling it such that it lies on the limit surface.

With our approximation established, we study our method for two particular cases. First for a right angled triangle with contact points at the each of the vertices and second for the same triangle but with a continuous contact patch as shown in Figure 3.5. The reference point C is taken as the centroid of the triangles and positioned at the origin. We assume uniform friction distribution in both the cases. We generated 2000 data points for the fit and additionally 900 points on the facets for the first case to complete our fitting data. Similarly a testing data of 2000 data points was also generated which will be used for error measurement in the next section.



(A) Triangle with three contact points

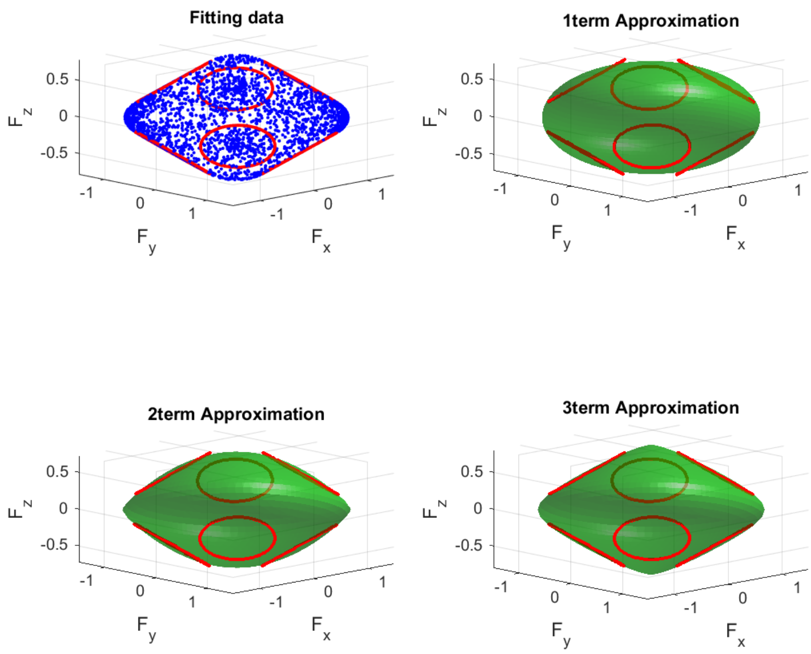


(B) Triangle with continuous contact patch

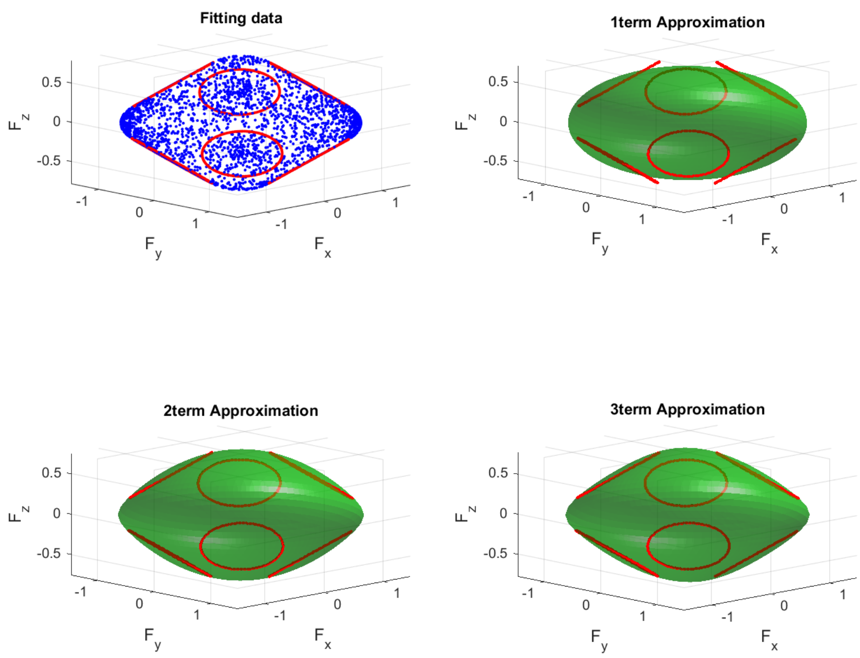
FIGURE 3.5: Two different slider configurations

With the data set ready, the matrices are fitted for the approximations and the limit surfaces are then generated using these matrices. Figure 3.6 shows the limit surfaces generated for the case of three contact point using one, two, and three term approximation along with the fitting data. Figure 3.7 shows the same for the case of a continuous contact patch.

Figure 3.6 illustrates the fitting data set and the approximated surface. In Figure 3.6a it can be observed that as the terms in approximation increase the features of the facets are captured effectively. One term approximation which is basically an ellipsoid approximation struggles to capture the facets whereas three term approximation is able to capture these features. Figure 3.6b shows the results of adding an extra velocity error in our objective function, the corresponding failure surface may not be as accurate as in Figure 3.6a but, should give us better velocity approximations as compared to **J1**. This is further studied in the next section.

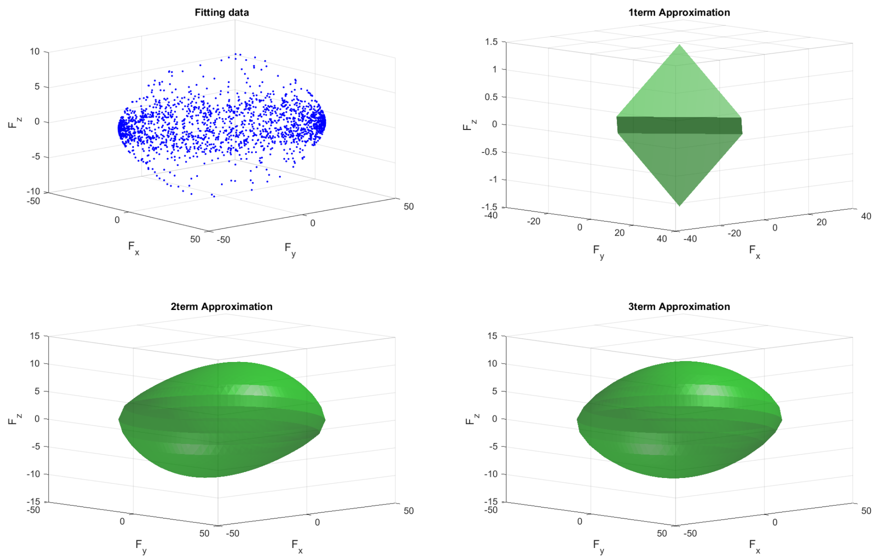


(A) Approximations using J1

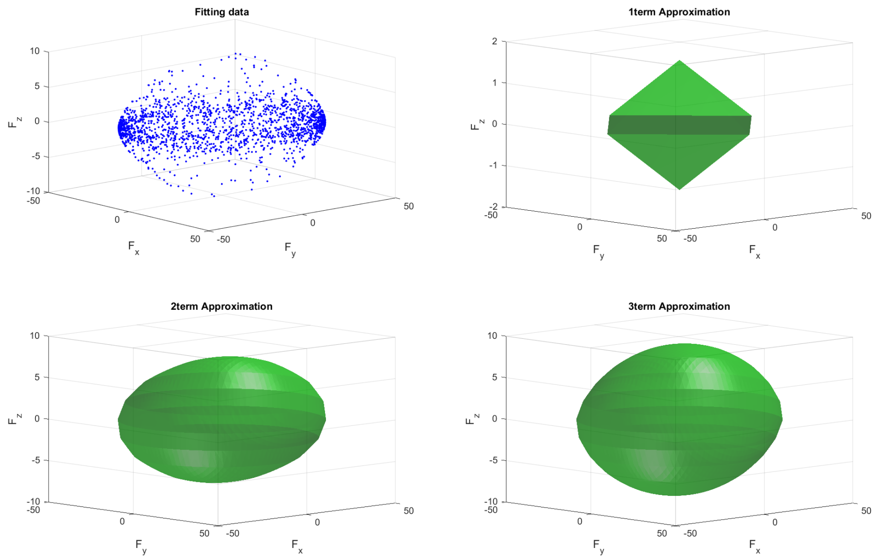


(B) Approximations using J2

FIGURE 3.6: Approximated LS for 3 point contact



(A) Approximations using J1



(B) Approximations using J2

FIGURE 3.7: Approximated LS for continuous patch

Figure 3.7 shows the fitting data and the approximated surface for the continuous contact patch. It can be seen that the one term approximation does not perform well, as it maps most of the data set to the center of the surface. Higher terms of J1 and J2 fits the surface well.

We try another method to optimize the two term approximation with the same objective function, but here we minimize the objective function wrt B. Lets call this second approximation and note that A is only symmetrical while B is symmetrical and positive semi definite. A is solved in a least square sense using the eqn(3.14) and the objective function is minimized wrt parameters in matrix B.

$$f_k^T A f_k = 1 - (f_k^T B f_k)^{1/2} \tag{3.14}$$

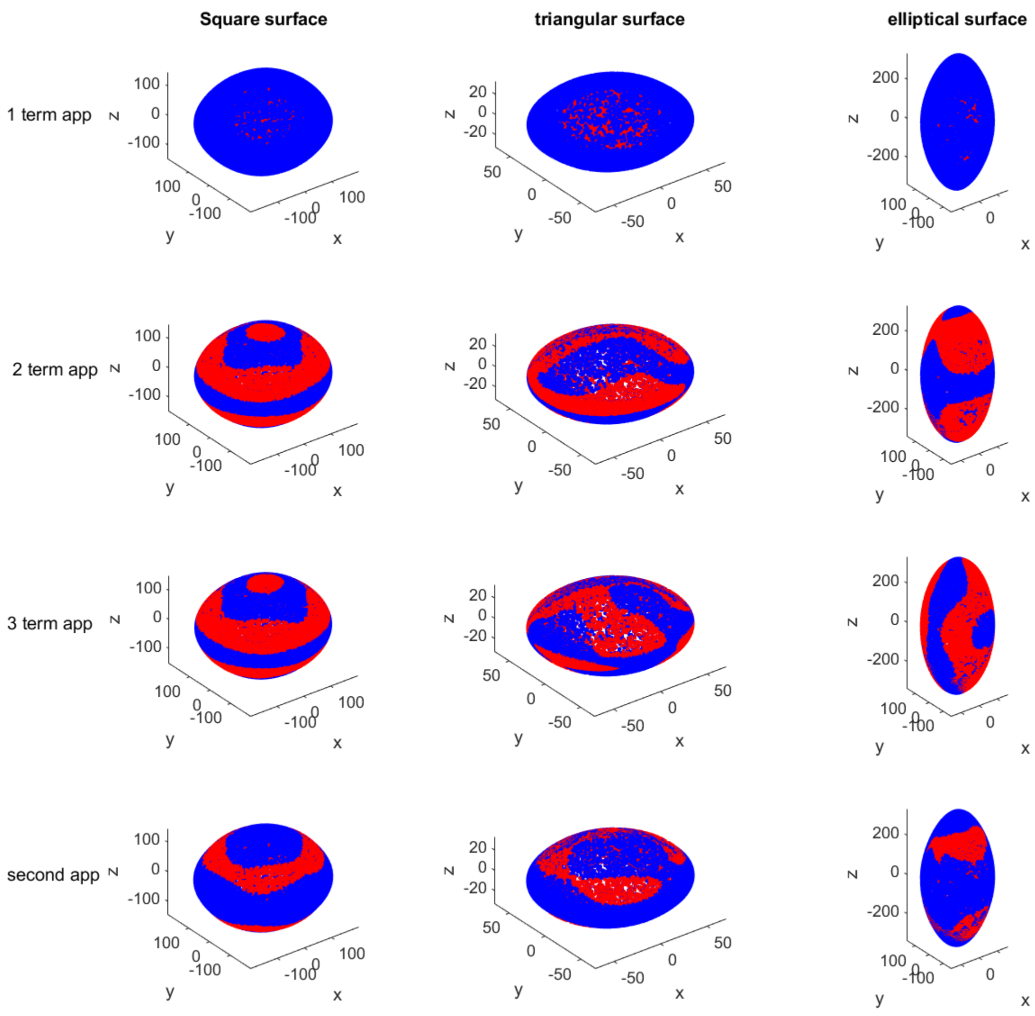


FIGURE 3.8: Approximations: blue- original red- approximation

Figure 3.8 shows upto three term approximation of limit surfaces for three different continuous surfaces, square, triangular and elliptical. The blue surface is the original data and the red surface is the approximated surface. It is clear that the one term approximation (ellipsoidal approximation) struggles, whereas the higher term approximations perform better.

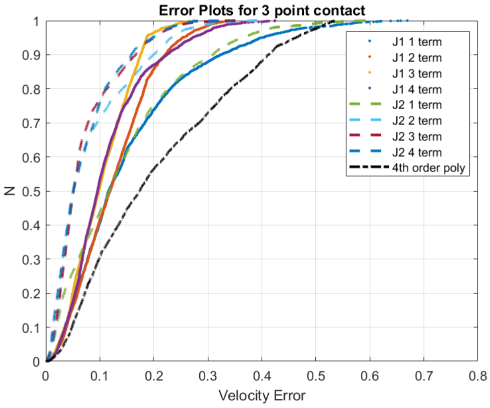
3.5 Error measure

We established our method and saw several examples that gave us an idea how our formulation performs with increasing terms. In this section, we look at the performance of the formulation in terms of errors. For the same, we define some error measures and then plot the empirical distribution of the errors to examine its performance. To study this, two types of error measure are defined, force based and velocity based. Here we refer loads and motion vectors as force and velocity vectors respectively.

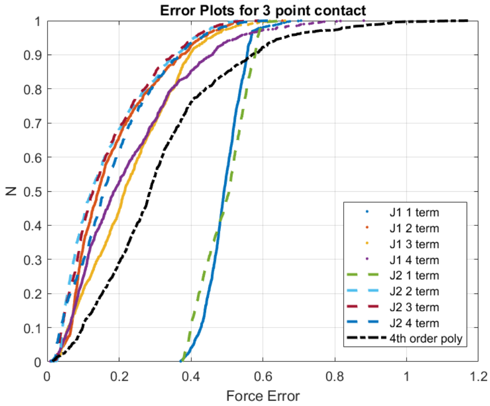
In velocity based error, our aim is to examine at the errors between the original motion vector and the approximated motion vector the formulation generates. For a testing data (f, v), approximated velocity $\hat{v}$ is calculated using normalised surface normal given by the gradient, $\hat{v} = \frac{\nabla H(f)}{\|\nabla H(f)\|}$ and error is defined as the norm of the difference between the two, i.e.- $\|v - \hat{v}\|$. Since v and $\hat{v}$ are unit vectors, error can range from zero to two.

In force based error, our aim is to study the errors between the original load vectors and the approximated load vector. Here we take a slightly different approach to generate our original load vectors. 1000 unit vectors lying uniformly on the surface of the sphere are considered, which are then scaled using equations 3.11, 3.12, such that they lie on the limit surface. Lets call them f_s (scaled force), these are forces generated using our formulation. Now, using f_s velocity v_s is calculated using normalised unit gradient at f_s, $v_s = \frac{\nabla H(f_s)}{\|\nabla H(f_s)\|}$. For a given motion direction, calculation of frictional forces is easy. So, using v_s, corresponding forces f are calculated by equation 3.1. These forces serve as our original force vectors, which are then compared with our approximated forces. Hence, force error is defined as the norm of the difference between f_s and f, i.e.- $\|f_s - f\|$.

The empirical distribution of the errors are plotted in Figures 3.9 and 3.10 to compare the different fits and objective functions. A fourth order convex polynomial fit as described in [3] is fitted using a least square method and superimposed on the plots for comparison.

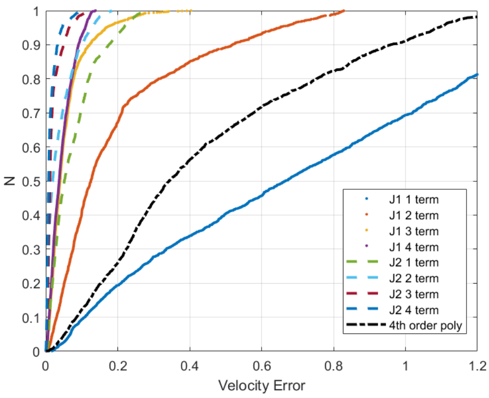


(A) Velocity error distribution

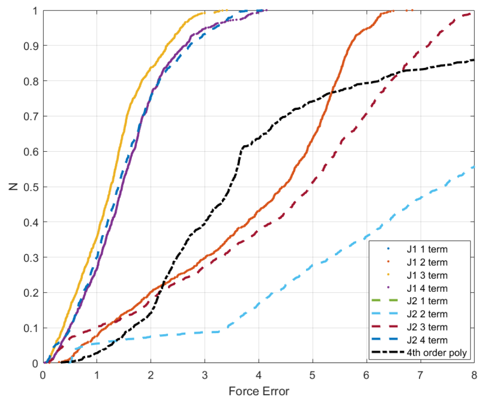


(B) Force error distribution

FIGURE 3.9: Error plots for 3 point contact



(A) Velocity error distribution



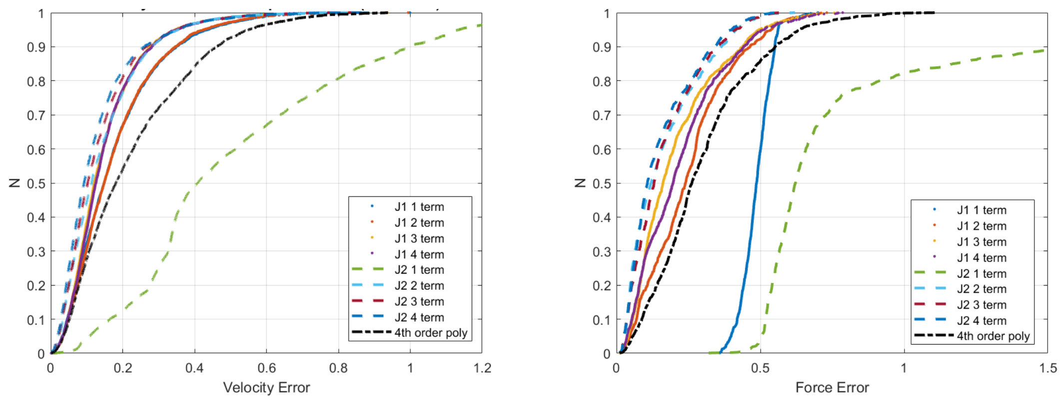
(B) Force error distribution

FIGURE 3.10: Error plots for continuous distribution

In velocity error distributions, J2 performs better than J1 as expected and with higher terms quality of the approximation increases (100% of the points have errors less than 0.3). Almost every approximation performs better than the fourth order polynomial in case of three point contact. Same observations can be drawn in case of continuous contact patch except that fourth order polynomial performs better that one term approximation fitted using J1 which is basically an ellipsoid.

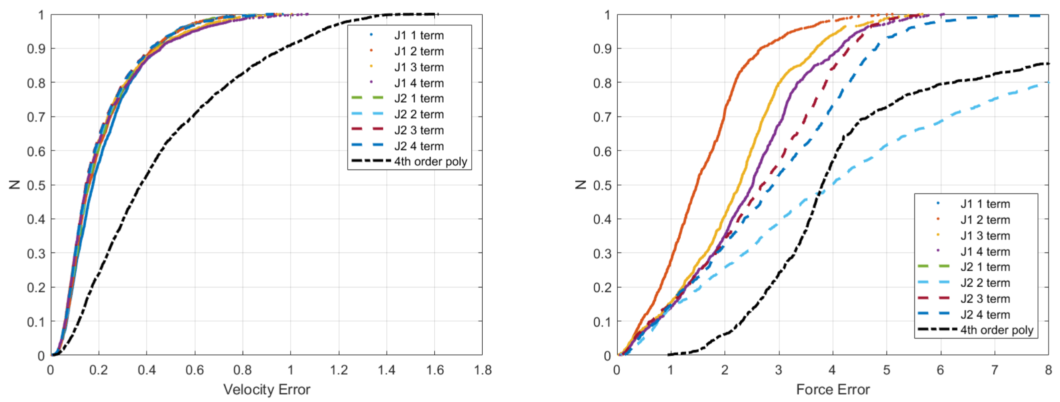
In force error distributions, higher term approximations fitted using J2 perform similar to two term approximation fitted using J1. We observe that one term approximations perform poorly in case of three point contact which is expected since they are ellipsoidal approximation and wont be able to capture the features of the facets. In case of three point contact, higher order approximation perform much better than the fourth order polynomial. And in case of continuous contact patch approximations fitted using J1 perform much better than those fitted using J2, as J1 is designed to fit the failure surface. Higher terms approximations perform better than the fourth order polynomial.

To test the robustness of our formulation we add noise to the training as well as testing data and repeat the above process to verify the results. With noise introduced in the data the results don't vary drastically as seen in Figures 3.11 and 3.12. But a change can be noticed in velocity error plot for the continuous case (Figure 3.12a), With the noise introduced in the system J1 and J2 perform equally contrary to Figure 3.10a, where J2 outperformed J1 suggesting it might be over fitting the velocity data. It can be taken care of by introducing a multiplication factor in the velocity error term in J2.



(A) Velocity error distribution (B) Force error distribution

FIGURE 3.11: Error plots for 3 point contact with noisy data



(A) Velocity error distribution (B) Force error distribution

FIGURE 3.12: Error plots for continuous contact patch with noisy data

Chapter 4

Conclusions

With concluding remarks and future scope of study, we outline our work on both areas.

4.1 Decaying functions

4.1.1 Concluding remarks

We devised a method to approximate a system's decaying response as a sum of exponentials. We constructed a DDE leading to characteristic equation and utilised its roots to form a basis for approximating the functions. With a high number of terms in the approximation, the method provides an excellent fit. We evaluated the method for some more decaying functions, in which some shortcomings of the methods were noticed. The method gave a poor fit with for a function which had slower decaying roots in its characteristic equation. The method provides a good fit till a finite distance upto which it is fitted, beyond which undesirable responses are observed which eventually decay.

4.1.2 Future scope

In the proposed method, there are a few more aspects to look into. We used integrals to build our DDE, but what other forms of equations may work can be investigated. To discover and approximate f, we employ an ad hoc method that must be addressed individually for each function. It remains to be seen whether an analytic solution for f can be found. Better optimization methods can be utilised once a basis are obtained.

4.2 Limit surfaces

4.2.1 Concluding remarks

We gave a quick introduction of limit surfaces, their importance and previous research into them. In the past, approximations for these limit surfaces have included ellipsoids in the simplest case, and multivariate polynomial expansions, a more sophisticated example. We proposed a new approximation which uses a small number of symmetrical 3×3 matrices along with fractional powers of simple quadratic forms. We fit our matrices using different techniques, concluding that there is a trade-off between load and motion predictions when determining the objective function. One (J2) would provide a better motion approximation, whereas the other (J1) would better fit the limit surface. We evaluated our formulation in a few scenarios and plotted errors, noting that higher term approximations are better than the commonly used ellipsoidal approximation and also outperforms the more complex fourth order convex polynomial fit.

4.2.2 Future scope

This study's analysis was based on theoretical data. It remains to be seen how the formulation performs when tested against various experimental data sets. Because multiple local minima were discovered while fitting the matrices, more sophisticated optimization procedures may result in superior fits.

Appendix A

Matlab codes

A.1 Approximation for decaying functions

A.1.1 Finding delay feedback

```
function f=findf(t,x)
dt=t(2)-t(1);
xd=[x(2:end)-x(1:end-1);0]/dt;
n=length(x);
A=zeros(n);
for k=1:n
    f=zeros(n,1);
    f(k)=1;
    c=conv(x,f);
    A(:,k)=c(1:n)*dt;
end
f=A\xd;
end
```

A.1.2 Infinity norm solver

```
function x=inf_norm_sol(A,b)
% solves Ax=b approximately such that Ax-b is minimized in the
% infinite-norm as opposed to x=A\b, which minimizes Ax-b in the 2-norm.
% It is assumed that A has n rows and m columns, with n >= m,
% and that b has n rows and ONE column
[n,m]=size(A);
```

```
f=[1;zeros(n+m,1)];
% op=optimset('maxiter',2000) this is an optional command
AA=[zeros(n,1),-eye(n),A];
B=zeros(2*n+1,n+m+1);
B(:,1)=-ones(2*n+1,1);
B(2:n+1,2:n+1)=eye(n);
B(n+2:2*n+1,2:n+1)=-eye(n);
z=zeros(2*n+1,1);
% x=linprog(f,B,z,AA,b,[],[],[],op); % use this, in place of the
% following line, if options are set above
x=linprog(f,B,z,AA,b);
x=x(n+2:end);
end
```

A.2 Approximation for limit surface

A.2.1 Data generation

```
function [F,v]=datagen
%% normal generated
num=2000;
n=randn(num,3);
for i=1:num
    n(i,:)=n(i,:)/norm(n(i,:));
end
v=n;
%% forces generated
r=[[-40,-40,0];[-40,80,0];[80,-40,0]]; % vertices of the triangle
r=r/150;
[m1,m2]=size(r);
mu=(0.5)*ones(m1); % friction distribution
for j=1:num
    for i=1:m1
        vk(i,:)=n(j,:)+cross([0,0,n(j,3)],r(i,:)); % calculating velocity of each vertex
        f(i,:)=mu(i)*vk(i,1:2)/norm(vk(i,1:2));    % calculating fx and fy
        mk(i,:)=cross(r(i,:),[f(i,:),0]);           % calculating moment
    end
    F(j,:)=[sum(f(1:end,:)),sum(mk(1:end,end))];
end
v1=[];
% to calculate data for facets
for j=1:m1
```

```
    rc=r(j,:);          % position of COR
    omega=1;
    n1=cross([0,0,omega],-rc); % velocity of reference point
    n1(3)=omega;
    n1=n1/norm(n1);
    for i=1:length(r(:,1))
        vk(i,:)=-cross(r(i,:)-rc,[0,0,omega]); % Velocity at vertices
        if rc==r(i,:)       % force at COR is taken 0, will be adding random values in
    ↪ next section
            f(i,:)=[0,0];
            mk(i,:)=cross(r(i,:),[f(i,:),0]);
        else
            f(i,:)=mu(i)*vk(i,1:2)/norm(vk(i,1:2)); % forces at other vertices
            mk(i,:)=cross(r(i,:),[f(i,:),0]);
        end
    end
    F1(j,:)=[sum(f(1:end,:)),sum(mk(1:end,end))]; % total force
    v1=[v1;n1]; % respective velocity
end
F2=-F1; % to account for clockwise and anticlockwise
v2=-v1;
ff=[];
vv=[];
cnt=150;    % no of data points on the facets
theta=linspace(0,2*pi,cnt); % will be adding force at COR in all directions
for i=1:m1
    for j=1:cnt
        t=rand/2;   % friction distribution at COR (anything less than muN )
        f1=t*[sin(theta(j)),cos(theta(j))];
        mk(i,:)=cross(r(i,:),[f1,0]);
        f=[f1,mk(i,3)]+F1(i,:); % adding force at cor to the total force
        ff=[ff;f];
        %plot3(f(1,1),f(1,2),f(1,3),'.r')
        vv=[vv;v1(i,:)];    % accounting for corresponding velocity
    end
    % same process below when direction of rotation is reversed
    for j=1:cnt
        t=rand/2;
        f1=t*[sin(theta(j)),cos(theta(j))];
        mk(i,:)=cross(r(i,:),[f1,0]);
        f=[f1,mk(i,3)]+F2(i,:);
        ff=[ff;f];
        %plot3(f(1,1),f(1,2),f(1,3),'.r')
        vv=[vv;v2(i,:)];
    end
```

```
end
v=[v;vv]; %
F=[F;ff];
```

A.2.2 Velocity error

```
function [Ev]=getappV(q)
t=q(1:6);
A=[[t(1);0;0],[t(2:3);0],[t(4:6)]];
A=A'*A;
if length(q)>6
    t=q(7:12);
end
B=[[t(1);0;0],[t(2:3);0],[t(4:6)]];
B=B'*B;
if length(q)>12
    t=q(13:18);
end
C=[[t(1);0;0],[t(2:3);0],[t(4:6)]];
C=C'*C;
if length(q)>18
    t=q(19:24);
end
D=[[t(1);0;0],[t(2:3);0],[t(4:6)]];
D=D'*D;
E=0;
Ev=[];
load FV_test_noise
[n1,n2]=size(F);
Fapp=[];
vapp=[];
for k=1:n1
    f=F(k,:);
    va=v(k,:);
    if length(q)==6
        g=2*f*A;
        g=g/norm(g);
        vapp=[vapp;g];
        Ev=[Ev;norm(va-g)];

    elseif length(q)==12
        g=2*f*A+(1/2)*((f*B*f')^(-1/2))*(2*f*B);
        g=g/norm(g);
```

```
        vapp=[vapp;g];
        Ev=[Ev;norm(va-g)];

    elseif length(q)==18
        g=2*f*A+(1/3)*((f*B*f')^(-2/3))*(2*f*B)+(2/3)*((f*C*f')^(-1/3))*(2*f*C);
        g=g/norm(g);
        vapp=[vapp;g];
        Ev=[Ev;norm(va-g)];

    elseif length(q)==24

        g=2*f*A+(1/4)*((f*B*f')^(-3/4))*(2*f*B)+(2/4)*((f*C*f')^(-2/4))*(2*f*C)+(3/4)*((f*D*f')^(-1/4))*(2*f*D);
        g=g/norm(g);
        vapp=[vapp;g];
        Ev=[Ev;norm(va-g)];
    end

end
Ev=sort(Ev);
end
```

A.2.3 Force approximation and error

```
function [Fapp,vapp]=getappF(q)
t=q(1:6);
A=[[t(1);0;0],[t(2:3);0],[t(4:6)]];
A=A'*A;
if length(q)>6
    t=q(7:12);
end
B=[[t(1);0;0],[t(2:3);0],[t(4:6)]];
B=B'*B;
if length(q)>12
    t=q(13:18);
end
C=[[t(1);0;0],[t(2:3);0],[t(4:6)]];
C=C'*C;
if length(q)>18
    t=q(19:24);
end
D=[[t(1);0;0],[t(2:3);0],[t(4:6)]];
D=D'*D;
```

```
E=0;
Ev=[];
load Fsphere
[n1,n2]=size(F);
Fapp=[];
vapp=[];
for k=1:n1
    f=F(k,:);
    if length(q)==6
        m=0;
        E=E+(f*A*f'-1)^2;
        alpha=(1/(f*A*f'))^1/2;
        if any(imag(alpha))
            disp(alpha);
            error('alpha is imaginary')
        end

        Fapp=[Fapp;alpha*f];
        f=alpha*f;
        g=2*f*A;
        g=g/norm(g);
        vapp=[vapp;g];

    elseif length(q)==12
        m1=0.5;
        E=E+(f*A*f'+(f*B*f')^m1 -1)^2;
        c=roots([f*A*f',(f*B*f')^m1,-1]);
        cr=c(imag(c)==0);
        cr=cr(real(cr)>=0);
        cc=abs(cr-1);
        [m,n]=min(cr);
        alpha=cr(n);
        if any(imag(alpha))
            disp(alpha);
            disp(cr(n));
            disp(cr);
            error('alpha is imaginary')
        end
        Fapp=[Fapp;alpha*f];

        f=alpha*f;
        g=2*f*A+(1/2)*((f*B*f')^(-1/2))*(2*f*B);
        g=g/norm(g);
        vapp=[vapp;g];

```

```
    elseif length(q)==18
        m1=1/3; m2=2/3;
        E=E+(f*A*f'+(f*B*f')^m1+(f*C*f')^m2 -1)^2;
        c=roots([f*A*f',(f*C*f')^m2,(f*B*f')^m1,-1]);
        cr=c(imag(c)==0);
        cr=cr(real(cr)>=0);
        cc=abs(cr-1);
        [m,n]=min(cr);
        alpha=(cr(n))^(3/2);
        if any(imag(alpha))
            disp(alpha);
            disp(cr(n));
            disp(cr);
            error('alpha is imaginary')
        end
        Fapp=[Fapp;alpha*f];

        f=alpha*f;
        g=2*f*A+(1/3)*((f*B*f')^(-2/3))*(2*f*B)+(2/3)*((f*C*f')^(-1/3))*(2*f*C);
        g=g/norm(g);
        vapp=[vapp;g];


    elseif length(q)==24
        m1=1/4; m2=2/4; m3=3/4;
        E=E+(f*A*f'+(f*B*f')^m1+(f*C*f')^m2 +(f*D*f')^m3 -1)^2;
        c=roots([f*A*f',(f*D*f')^m3,(f*C*f')^m2,(f*B*f')^m1,-1]);
        cr=c(imag(c)==0);
        cr=cr(real(cr)>=0);
        cc=abs(cr-1);
        [m,n]=min(cr);
        alpha=(cr(n))^(2);
        if any(imag(alpha))
            disp(alpha);
            disp(cr(n));
            disp(cr);
            error('alpha is imaginary')
        end
        Fapp=[Fapp;alpha*f];

        f=alpha*f;
        g=2*f*A+(1/4)*((f*B*f')^(-3/4))*(2*f*B)+(2/4)*((f*C*f')^(-2/4))*(2*f*C)+(3/4)*((f*
    ↪ D*f')^(-1/4))*(2*f*D);
        g=g/norm(g);
```

```
        vapp=[vapp;g];
    end

end
```

```
function [EF]=F_error
load fvn_j2_1term % contains Fapp and vapp generated from above code
[num,num1]=size(Fa);
n=va;
%% forces generated
%r=[[0,1,0];[cosd(30),-sind(30),0];[-cosd(30),-sind(30),0]];
r=[[-40,-40,0];[-40,80,0];[80,-40,0]];
r=r/150;
[m1,m2]=size(r);
mu=(0.5)*ones(m1);
EF=[];
EFr=[];
for j=1:num
    for i=1:m1
        vk(i,:)=n(j,:)+cross([0,0,n(j,3)],r(i,:));
        f(i,:)=mu(i)*vk(i,1:2)/norm(vk(i,1:2));
        mk(i,:)=cross(r(i,:),[f(i,:),0]);
    end
    F(j,:)=[sum(f(1:end,:)),sum(mk(1:end,end))];
    EF=[EF;norm(F(j,:)-Fa(j,:))];
end
v=n;
EF=sort(EF);
end
```

Bibliography

[1] S. Goyal, A. Ruina, and J. Papadopoulos, "Planar sliding with dry friction part 1. limit surface and moment function," *Wear*, vol. 143, no. 2, pp. 307–330, 1991.

[2] R. D. Howe and M. R. Cutkosky, "Practical force-motion models for sliding manipulation," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 557–572, 1996.

[3] J. Zhou, R. Paolini, J. A. Bagnell, and M. T. Mason, "A convex polynomial force-motion model for planar sliding: Identification and application," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 372–377.

[4] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *The international journal of robotics research*, vol. 15, no. 6, pp. 533–556, 1996.

[5] M. R. Dogar and S. S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 2123–2130.

[6] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1578–1585.

[7] K. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 30–37.

[8] W. Zhang, S. Seto, and D. K. Jha, "Cazsl: Zero-shot regression for pushing models by generalizing through context," 2020.

[9] N. Chavan-Dafle and A. Rodriguez, “Prehensile pushing: In-hand manipulation with push-primitives,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 6215–6222.