2010

Fast numeric geometric techniques for COMPUTER GENERATED DAE MODELS

Niloofar Mani
Western University

Fast numeric geometric techniques for computer generated DAE models

(Thesis Format: Monograph)

by

Niloofar Mani

Graduate Program
in
Applied Mathematics

A thesis is submitted in partial fulfillment
of the requirements for the degree of
Master of Science

School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor	Examiners
Dr. Greg J. Reid	Dr. David J. Jeffrey
	Dr. Stephen M. Watt
	Dr. Eric Schost

The thesis by

Niloofar Mani

entitled:

Fast numeric geometric techniques for computer generated DAE models

is accepted in partial fulfillment of the requirements for the degree of Master of Science

Date Aug, 23, 2010

Chair of the Thesis Examination Board
Dr. Robert M. Corless

Abstract

Complicated nonlinear systems of ordinary differential equation with constraints (so-called differential algebraic equation (DAE)) arise frequently in applications, and are often so complicated that they are in practice automatically generated by computer modeling and simulation environments. We used the MapleSim software to generate such systems. Missing constraints arising by prolongation (differentiation) of the DAE need to be determined to consistently initialize and stabilize their numerical solution.

In this thesis, we review a fast prolongation method to find hidden constraints, and apply it to systems from MapleSim models. Our symbolic numeric prolongation method avoids the unstable eliminations of exact approaches, and applies to square systems (i.e. systems having the same number of equations and dependent variables). The method is successful provided the prolongations have a block structure, which is efficiently uncovered by Linear Programming. Constrained mechanical systems generated by MapleSim are used to demonstrate the power of the approach. The geometry of the constraints, regarded as the solution set of a positive dimensional polynomial system, is determined by using the new tools of numerical algebraic geometry. We used Bertini, a global homotopy continuation solver, for this purpose. In particular Bertini determines consistent initial conditions on the constraints. These conditions, together with the block structure and an efficient Maple interface enable the efficient numerical solution of the system by standard ODE methods.

Dedicated to my beloved husband, Saeid, the best event of my life.

Acknowledgements

Here I would like to thank those who made this thesis possible. It is my pleasure to thank my M.Sc. thesis supervisor, Dr. Greg Reid for his support and encouragement. During these past two years, he has given me excellent advice about my research, my thesis, and my teaching style. Moreover, I am so thankful for his kind guidance in course projects.

It was an honor for me to be a student in Dr. Robert M. Corless and Dr. Colin Denniston's course. From them I was given a good start to my study.

Many thanks to all members of MapleSoft company, especially Juergen Gerhard, Gilbert Lai, Allan Wittkopf, Austin Roche, Chad Schmitke and Erik Postma. They taught me more than I was expecting to learn about the Maple and MapleSim software during the valuable internship opportunity in this past summer.

I am indebted to Paul Vrbik for every single moment that he has patiently tried to teach me the Maple Software. Thank you, Paul.

Thanks to Wenyuan Wu, for all comments and useful advice about my research.

I like to appreciate all advice I have received from Dan Bates and Jonathan Hauenstein about the Bertini software and homotopy method.

I would like to thank my lovely academic sister, Xuan Liu who has tried her best to replace my family while I was far from them.

Many thanks to all members of the ORCCA lab, the faculty members, staff (especially Audrey Kager) and all my classmates and friends (especially my Canadian sister Nancy Spring) for the ongoing support I have received from them.

I owe my deepest gratitude to my beloved husband, Saeid Kanani for all the support he has given during this time. He has made available his help in a number of ways to improve my living quality and make these two years an enjoyable time. I love you very much, Saeid.

I would also like to show my gratitude to my wonderful mother Zohreh Azarnoush, my late father Dr. Ahmad Mani and my lovely sister Yasaman, of whom I have always been proud.

Love and thanks to my granny, Madar, for all her prayers.

Finally, I would like to express thanks to my mother-in-law, Soha, and Masood Kanani for all their support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The analysis of nonlinear systems of differential algebraic equations (DAEs) occurs commonly in applications. Missing constraints need to be determined for DAEs in order even to initialize and stabilize their numerical solution. A simple example of a DAE is:

$$\begin{aligned} X_{tt} + \lambda X &= 0 \\ Y_{tt} + \lambda Y &= -g \\ X^2 + Y^2 &= 1 \end{aligned} \tag{1.1}$$

The system (1.1) is for the motion of the simple pendulum shown in Figure 1.1.

Figure 1.1: One pendulum system.

Here $X(t)$, and $Y(t)$ give the position of a unit mass on a unit length pendulum under constant gravity g with λ as a Lagrange multiplier. Also $X^2 + Y^2 = 1$ can be regarded as a constraint. In the form (1.1) notice that the initial conditions:

$$\begin{aligned} X(0) &:= X^0, \quad X_t(0) := X_t^0 \\ Y(0) &:= Y^0, \quad Y_t(0) := Y_t^0 \end{aligned} \tag{1.2}$$

can not be independently posed at $t = 0$. Obviously $(X^0)^2 + (Y^0)^2 = 1$. The derivative of the constraint is $2XX_t + 2YY_t = 0$ and so the missing constraint obtained by the differentiation must also be satisfied by the initial conditions. Thus $2X^0X_t^0 + 2Y^0Y_t^0 = 0$. Of course, for the pendulum we can have its constraint removed by changing to polar coordinates as below:

$$X := \sin(\theta), \quad Y := -\cos(\theta). \tag{1.3}$$

Then the DAE is equivalent to the explicit ordinary differential equation (ODE):

$$\theta_{tt} + g\sin(\theta) = 0 \tag{1.4}$$

But often DAEs are too complicated, especially those that are automatically generated, to have their constraints algorithmically removed. Thus an important problem is to determine all missing constraints for DAE so that constant initial conditions satisfying the constraints can be determined for their numerical solution. More general models are partial differential algebraic equations (PDAE). In this thesis we focus on DAEs which are automatically generated by computer software.

Wu, Reid, and Ilie gave a symbolic-numeric approach [1, 2] for the computation of Riquier Bases for PDAE, and showed that it can be useful in finding missing constraints for such systems. In particular they presented new theoretical results, showing that their method can be naturally applied to the approximate solution of PDAE by semi-discretization (i.e. by the method of lines). In this thesis we apply their method to systems of DAEs arising from various models generated by the MapleSim Software [3, 4]. MapleSim is a multi-domain modeling and simulation tool running in Maple.

We used the Wu and Reid's method [1] to determine missing constraints for MapleSim models and find initial conditions to numerically solve them. As described in [2] differential elimination algorithms apply a finite number of differentiations and eliminations to uncover obstructions to formal integrability (i.e. to finitely characterize the relations between all the Taylor coefficients of solutions at a point). Since many numerical solution methods depend on or are equivalent to Taylor expansions, the determination of such obstructions or missing constraints can be essential for such methods. Exact differential elimination algorithms that apply to exact polynomially nonlinear systems of PDAE are given in [5, 6, 7, 8, 9]. Such methods identify all hidden constraints of PDAE systems and the computation of initial conditions and associated formal power series solutions in the neighborhood of a given point.

A major problem in these approaches is the exploding size of prolongations (differentiations) for more than one independent variable. In symbolic approaches much effort has been devoted to control the growth of this size by developing redundancy criteria (for integrability conditions), and making strong use of elimination with respect to rankings to decrease the size of the prolongations [10, 11]. However, symbolic elimination can cause expression swell even in the case of one independent variable, such as for DAE problems arising in multi-body mechanics.

Very little work has been done on the corresponding problems for symbolic-numeric methods. Techniques which are helpful for the symbolic case are often unstable for the approximate case, since the rankings (the differential analogue of term orders or even more primitively, ordered Gaussian elimination) that underlie symbolic methods can cause pivoting on small quantities and result in instability.

References [1, 2] give progress on this problem for a certain class of PDAE. For this class, only prolongations with respect to one independent variable are needed. Rankings are important in [1, 2] but do not cause instability since no eliminations are made. Hence the expression swell due to the eliminations mentioned above is avoided. A suitable ranking is determined by solving an integer linear programming problem to uncover a block structure in the PDAE system.

Another main idea in Wu et al. [1, 2] is that such prolongations are essentially DAE-like, and enable DAE techniques to be generalized to the PDAE case. In that case they generalized a method of Pryce for DAE in the framework of Riquier Theory. In addition, the method in their work yields the method of Pryce [12] for systems of DAE as a special case which is very useful here because all MapleSim systems are DAE.

Prolongation will usually introduce more equations as well as more variables, but not always. If some equations after differentiation do not introduce new variables for the whole system, then there is the possibility that the dimension of the system is lowered. Pryce [12] proposed a method to detect such "chances" that minimize the dimension by taking advantage of the special structure of some systems. Pryce's method was the generalization of a method developed by Pantelides with historical roots in the work of Jacobi (see [13]). References [14, 15] show that Pryce's method can be extended to give a polynomial cost method for the numerical solution of DAE.

Our goal in this thesis is to apply the method of Wu et al. [1, 2] to DAE models generated by the MapleSim package. Our models included: mechanical systems, electrical models, and multi-body systems. We simulated their behavior using the MapleSim software. In our algorithm we derive the DAE system from the model and use our method to simplify them. When a simplified system is obtained, we use the fast prolongation method with respect to one independent variable, usually time t, in order to find missing constraints. Pryce's method helps us to determine the missing constraints and to avoid expression swell. Next, we apply fast prolongation to the system to obtain it a block triangular form.

Bertini is a software package for computation in numerical algebraic geometry (Sommese et al. [16]). In particular this package uses numerical homotopy continuation to find all isolated solutions of polynomial systems. Further it determines (witness) points on all submanifolds (or components) of solutions. We use it to solve the polynomial systems arising in the block structure. Subsequently, it gives us the initial conditions to consistently initialize and then numerically solve the simplified DAE system. Finally [17], using `dsolve` in Maple 13 gives us the numeric solution for the system.

MapleSim models also provide examples of multi-body systems with singularities. See [18, 19, 20] for the analysis of singularities of such systems with closed kinematic loops. In Chapter 6 we will study MapleSim models with singularities.

Arponen et al. in [18] worked on singularities of a benchmark problem called the "Andrews squeezing system". They show that for some physically feasible parameter values this system has singularities. Later, Piipponen in [19] used the same approach on planar linkages. He analyzed closed four-bar mechanisms, closed five-bar mechanisms, and closed six-bar mechanisms for their singularities. In particular he used the tools of computational algebraic geometry to determine the singularities. He also established strategies to find necessary conditions for singularities. The methods are based on Gröbner basis computations and ideal decomposition [21].

The paper [20] by Arponen motivated us to create a two equal-length bar slider crank using MapleSim. The purpose of that paper is to present a new technique for regularizing certain singularities of mechanical systems. Arponen's method is based on the elimination of singularities by reformulating multibody systems of equations in such a way that the cause of singularity is eliminated. In particular, we made systems of a two equal-length bar slider crank to study singular cases using MapleSim. We described the differences of our approach to the approach given in [18, 19, 20].

In Chapter 2, we give an overview of the fast prolongation method and how it produces block triangular structures. We discuss DAE and PDAE and continue by defining of a t-dominated system and a signature matrix for such a system. Then we show how to use this matrix to do the fast prolongation method on a square system and obtain a block triangular structure. Finally, we use the example of a one pendulum system to illustrate these methods.

Chapter 3 provides an introduction to homotopy continuation which is a very powerful and useful method to find isolated solutions of a multivariate polynomial system. We present our homotopy algorithm which is implemented in Maple in order to find the zero set of a polynomial system. In our package we first execute the fast prolongation and subsequently obtain the block structures. The Bertini software for numerical homotopy continuation is used to solve the block structure and give consistent initial conditions for the system. Moreover, the Bertini software is able to characterize positive dimension solution sets by using so-called witness points for polynomial systems. We conclude this chapter with some examples.

Chapter 4 gives a complete description of our method. First we make a system in MapleSim, derive its equations, and simplify them. Next we apply fast prolongation to the simplified system in order to determine its missing constraints. Then we determine the block structure. Finally, we use the block structure to determine initial conditions. The slider crank example is used here to illustrate our method.

In Chapter 5 we apply our method to examples. Firstly, we worked on a non-linear damper with a linear spring which is made in MapleSim by using a so-called custom component. This enables us to add new components to the MapleSim library. The second example we describe is a multi-domain system of a DC-motor attached to a slider crank which has a piecewise function in its system of DAE. The goal of this example is to illustrate the way that we can use Bertini when we have a piecewise function.

Finally in Chapter 6, we discuss a special case of a slider crank which has singularity. We analyze singularity of the model and show how Bertini is used to detect the singularity set of this system.

In our conclusion, Chapter 7, we summarize our results, discuss future work and open problems.

Chapter 2

Fast prolongation method

2.1 Introduction

We work with systems produced by MapleSim such as Electrical, Mechanical, Multibody and other types of systems. The MapleSim package is interfaced to Maple. In particular MapleSim1, MapleSim2, MapleSim3, and MapleSim4 were released by MapleSoft during 2008-2010 to run with Maple 12, 13, and 14. The model equations produced by MapleSim are DAEs.

Wu, Reid and Ilie in [1, 2] introduced a fast prolongation method, a development of (explicit) symbolic Riquier Bases for partial differential algebraic equation (PDAE). Their methods are applicable to a class of PDAEs that are dominated by pure derivatives in one of their independent variables with respect to some (partial) ranking. System of DAEs are dominated by pure derivatives in t, and the fast prolongation methods enable the determination of missing constraints without unstable elimination. It also enables the determination of initial conditions that can be used to solve the system numerically. To begin we present some basic definitions.

Definition 2.1.1. *Let $u \in \mathbb{R}^m$ be an unknown vector valued function in t. An equation of the form $u^{(n)} = F(t, u, \dot{u}, \ddot{u}, \dots)$ is called an ordinary differential equation (ODE) of* *order n. Therefore, a first order ODE can be written as*

$$\dot{u} = F(t, u) \tag{2.1}$$

Initial conditions are easy to state for such ODE. Most generally a DAE is a system in implicit form which is not equivalent to an explicit ODE. For example $f(t, u, \dot{u}) = 0$ where $\frac{\partial f}{\partial \dot{u}}$ is singular. DAEs are distinct from ODEs in that a DAE is not completely solvable for the derivatives of all components of the function u.

One often cited subclass of DAE are those of form:

$$\begin{cases} f(t, u, \dot{u}, \ldots, u^{(\alpha)}) = 0, \\ \qquad g(t, u) = 0, \end{cases} \tag{2.2}$$

Here t is the independent variable (time for us) and $u \in \mathbb{R}^m$ is the m-vector of dependent variables. The function f consists of the equations modeling the dynamics. The function g contains the constraint equations (not hidden ones) which is a set of algebraic equations. And, there is a singularity when the Jacobian of g is not maximal rank [20, 21].

In a similar manner a PDAE is a general systems of PDE which is not in explicit solved form (see Reid, Lin, and Wittkopf [6]).

2.2 Signature matrix of t-dominated systems using rankings

Systems which are t-dominated, are dominated by pure derivatives. We will explain definition of t-dominated systems in 2.2.10.

Definition 2.2.1. ***(Pure-derivative** [1, 2]) Let $\mathbb{F}$ be a field* ($\mathbb{R}$ *or* $\mathbb{C}$ *in our case). Let $x := (x_1, \ldots, x_n)$ be the independent variables (which is just t in our work), and $u = (u^1, \ldots, u^m)$ be the dependent variables for a system of PDAE. A pure derivative with respect to an independent variable x_i, is a derivative of the form $(\frac{\partial}{\partial x_i})^k u^j$ where $k \in N = \{0, 1, 2, \ldots\}$.*

For example u_{tt} is a pure derivative in t, while u_{xt} is not a pure derivative. Systems which are t-dominated, are dominated by pure t-derivatives which are highest with respect to a ranking.

Definition 2.2.2. ***(Jet variables** [1, 2])* *Consider a set of m dependent variables $u^i(x_1, \ldots, x_n)$ which are functions of n independent variables $x_1, \ldots, x_n$. The set of jet variables is the set of indeterminates $\Omega = \{v^i_\alpha | \, \alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n \cup \{0\}, i = 1, \ldots, m\}$ where each member of Ω corresponds to a partial derivative of u by:*

$$v^i_\alpha \leftrightarrow (D_{x_n})^{\alpha_n} \ldots (D_{x_1})^{\alpha_1} u^i(x_1, \ldots, x_n) = D^\alpha \, u^i(x_1, \ldots, x_n). \tag{2.3}$$

An example of this correspondence is:

$$v_{32} \leftrightarrow (D_{x_1})^3 (D_{x_2})^2 u(x_1, x_2). \tag{2.4}$$

Definition 2.2.3. ***(Ranking** [22])* *A positive ranking $\prec$ of Ω is a total ordering on Ω which satisfies:*

$$\begin{gathered} v^i_\alpha \prec v^j_\beta \Rightarrow v^i_{\alpha+\gamma} \prec v^j_{\beta+\gamma}, \\ v^i_\alpha \prec v^i_{\alpha+\gamma} \end{gathered} \tag{2.5}$$

for all $\alpha, \beta, \gamma \in \mathbb{N}^n \cup \{0\}$.

Example 2.2.4. *When there is only one independent variable and one dependent variable, Definition 2.2.3 implies there is only one ranking:*

$$u \prec u_x \prec u_{xx} \prec \ldots. \tag{2.6}$$

Before we define t-dominated systems we need to consider rankings which are consistent with highest t-derivatives [1, 2]. For example, for two independent variables t, x and for each u^j, such a ranking needs to satisfy:

$$u^j \prec u^j_x \prec u^j_{xx} \prec \cdots \prec u^j_t \prec u^j_{tx} \prec \ldots \tag{2.7}$$

It is easy to extend this (partial) ranking to the case when x is a vector (e.g. using lexical order on x). In the general case $t = x_k$ for an x_k-dominated system. However, we caution that t may not represent time for some physical t-dominated systems.

Definition 2.2.5. *The leading derivative of each equation R_i with respect to each u^j using the (partial) ranking* (2.7)*, is denoted by $LD(R_i, u^j)$ which is highest ranked derivatives for R_i respect to u^j.*

Example 2.2.6. *In these systems below we show the leading derivative of each equation R_i respect to the mentioned variable using the (partial) ranking* (2.7)*:*

- $R = \{u_{ttt} - c^2 u_{xx} = 0\}$ *then* $LD(R_1, u) = u_{ttt}$.
- $R = \{w_{xt} - w_t = 0\}$ *then* $LD(R_1, w) = w_{xt}$.
- $R = \{u_{xt} - (v_{tt})^2 = 0, (v_{ttt})^2 + (v_x)^2 = 0\}$ *then* $LD(R_1, u) = u_{xt}$, $LD(R_1, v) = v_{tt}$, *and* $LD(R_2, v) = v_{ttt}$.

Defining a weight map $\varphi : \Omega \to \mathbb{R}$ with respect to $t = x_k$ as below, helps us to hide the details about the differential order of the other independent variables [1, 2]:

$$\varphi(v^i_\alpha) := \begin{cases} \alpha_k, & \text{if } \alpha_p = 0, \text{ for every } p \neq k; \\ \alpha_k + \epsilon, & \text{otherwise} \end{cases} \tag{2.8}$$

where "ϵ" is a symbolic parameter. For purpose of computation $\epsilon > 0$ may be taken small and positive.

Example 2.2.7. *The weight map for these three examples is:*

- $u_{tt} = (D_t)^2 u \Rightarrow \varphi(u_{tt}) = 2$.

- $u_{xxt} = (D_t)^1(D_x)^2 u \Rightarrow \varphi(u_{xxt}) = 1 + \epsilon$.

- $u = (D_{x_n})^0 \dots (D_{x_1})^0 u \Rightarrow \varphi(u) = 0$.

Definition 2.2.8. *(***Signature Matrix** *[1, 2]): By applying* (2.8) *to the leading derivatives of R, we obtain an $\ell \times m$ matrix $(\sigma_{i,j})$ which is called the signature matrix (with respect to t) of R (see Pryce [12] for the DAE case):*

$$(\sigma_{i,j})(R) := \begin{cases} \varphi(LD(R_i, u^j)), \textit{if } R_i \textit{ depends on } u^j \textit{ or any of its derivatives;} \\ -\infty, \qquad \textit{otherwise} \end{cases} \tag{2.9}$$

Example 2.2.9. *Consider $R := \{u_{ttt} - c^2 u_{xx} = 0\}$. The 1×1 signature matrix (with respect to t) for this R is: $\sigma = \varphi(LD(R_1, u)) = \varphi(u_{ttt}) = (3)$. For $R := \{(v_{tttt})^2 - v_{xt} + v_{xx} = 0\}$, $\sigma = \varphi(LD(R_1, v)) = \varphi(v_{tttt}) = (4)$. And for $R := \{w_{xt} - w_t = 0\}$, $\sigma = \varphi(LD(R_1, w)) = \varphi(w_{xt}) = (1 + \epsilon)$.*
The 2×2 signature matrix (with respect to t) of the system $\{u_{xt} - (v_{tt})^2 = 0, (v_{ttt})^2 + (v_x)^2 = 0\}$ with rows corresponding to R_1, R_2 and columns corresponding to u, v is:
$\begin{pmatrix} (1+\epsilon) & 2 \\ -\infty & 3 \end{pmatrix}$.

Definition 2.2.10. *(***t − dominated System** *[1, 2]) R is dominated by pure derivatives in the independent variable t (t-dominated) if there is no ϵ appearing in $(\sigma_{i,j})(R)$.*

Thus from the last example $u_{ttt} - c^2 u_{xx} = 0$ and $(v_{tttt})^2 - v_{xt} + v_{xx} = 0$ are t-dominated. In contrast $w_{xt} - w_t = 0$ and the system $\{u_{xt} - (v_{tt})^2 = 0, (v_{ttt})^2 + (v_x)^2 = 0\}$ are not t-dominated.

A PDAE system which is dominated by pure derivatives with respect to an independent variable x_i, must at least contain such a derivative in each of its equations.

Example 2.2.11. *$u_{ttt} - c^2 u_{xx} = 0$ and $v - u_x = 0$ both contain pure t-derivatives in their equations (u_{ttt} and v respectively). But $u_{xt} - v_{xxt} = 0$ contains neither a pure t or x-derivative, and so is not t-dominated.*

2.3 Generalizing the fast prolongation method

2.3.1 Square systems

Let R be a square (i.e. #equations=#unknowns=m) and t-dominated system. The signature matrix $(\sigma_{i,j})(R)$ contains information on the differential order and ignores details on the coefficients and degrees of items of a system R. Wu and Reid [1] introduced a fast method based on $(\sigma_{i,j})(R)$ to differentiate R with respect to t to include its missing constraints. Pryce's method for square DAEs is a special case (see [13]), and yields a local existence and uniqueness result.

Pryce's method [12] finds the local constraints for a large class of square DAE using only differentiation. This differentiation is called prolongation in the literature and in [1, 2] this construction is generalized to PDAE.

Systems from MapleSim models are systems of DAE with t as an independent variable. Therefore they are t-dominated systems. In the case of square systems from MapleSim models, we use the Wu and Reid fast prolongation method to uncover hidden constraints.

Suppose R_i is differentiated $c_i \geq 0$ times. The new system after differentiation is denoted by $D_t^c R$. Suppose the highest order of u_j that appears in $D_t^c R$ is d_j. From the definition of $(\sigma_{i,j})$, d_j is the largest of $c_i + \sigma_{ij}$, which gives

$$d_j - c_i \geq \sigma_{ij}, \quad \text{for all} \quad i, j. \tag{2.10}$$

There are at most $m + \Sigma d_j$ pure t-derivative jet variables and $m + \Sigma c_i$ equations in $D_t^c R$ (considering independent variables and all non-t-derivatives as parameters). Making the assumption that each equation drops the dimension by 1, the dimension of $D_t^c R$ is $\Sigma d_j - \Sigma c_i$. Finding all the constraints is equivalent to minimizing the dimension of $D_t^c R$. This can be formulated as an integer linear programming (LP) problem in the variables

$c = (c_1, \ldots, c_m)$ and $d = (d_1, \ldots, d_m)$:

$$\begin{cases} \text{Minimize} \quad z = \Sigma d_j - \Sigma c_i, \\ \text{where} \quad d_j - c_i \geq \sigma_{ij}, \\ c_i \geq 0 \end{cases} \tag{2.11}$$

Remark 2.3.1. *Wu et al. [2] say "This integer LP problem is dual to an assignment problem [12]. The task is to choose just one element in each row and column of the signature matrix, then maximize the sum of these m elements. The maximum is called the Maximal Transversal Value (MTV). If this value exists, then problem* (2.11) *has a finite solution. Such problems can be solved (and the existence of MTV can be checked) efficiently and in polynomial time by the Hungarian Method". In [1, 2] they used LPSolve in the optimization package of Maple 10 and their method is implemented here in Maple 13.*

2.3.2 Block triangular structures

After we obtain the number of prolongation steps c_i for each equation, we can construct the partially prolonged system $D_t^c R$ using c. We assume $c_1 \geq c_2 \geq \cdots \geq c_m$, and let $k_c = c_1$, which is closely related [12] to the index of system R. The r-th partial differentiation of a PDE R_j with respect to t is denoted by $R_j^{(r)}$. Then we can partition $D_t^c R$ into $k_c + 1$ parts for $0 \leq i \leq k_c$ by

$$B_i := \{R_j^{(i+c_j-k_c)} : 1 \leq j \leq m, i + c_j - k_c \geq 0\} \tag{2.12}$$

For $0 \leq i < k_c$, B_i has fewer variables than B_{i+1}. The block structure in the case $c_i = c_{i+1} + 1$ is given in Table 2.1.

B_0	B_1	$\dots$	B_{k_c-1}	B_{k_c}
$R_1^{(0)}$	$R_1^{(1)}$	$\dots$	$R_1^{(c_1-1)}$	$R_1^{(c_1)}$
	$R_2^{(0)}$	$\dots$	$R_2^{(c_2-1)}$	$R_2^{(c_2)}$
		$\vdots$	$\vdots$	$\vdots$
		$R_m^{(0)}$	$\dots$	$R_m^{(c_m)}$

Table 2.1: The triangular block structure of $D_t^c R$ for the case $c_i = c_{i+1} + 1$.

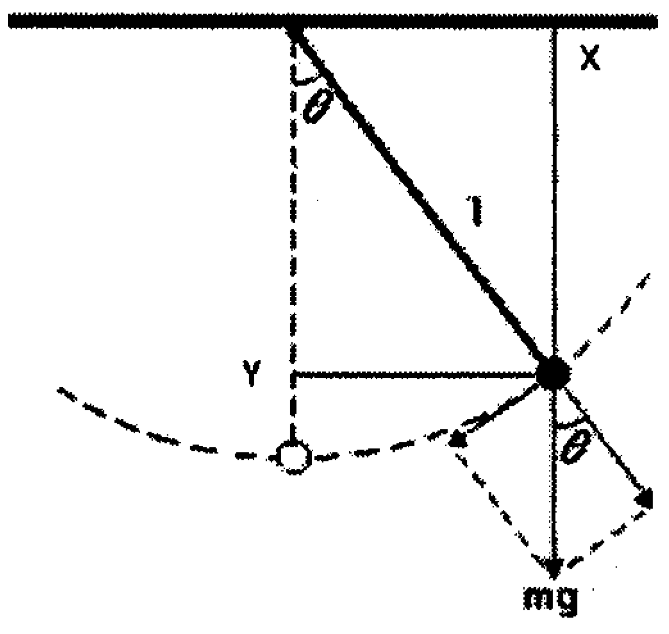


Figure 2.1: One pendulum system.

Example 2.3.2. The Pendulum. *Consider a pendulum of unit mass, under constant gravity g as shown in Figure 2.1. In particular $X(t)$, and $Y(t)$ give the position of the pendulum and $\lambda(t)$ is a Lagrange multiplier. These variables satisfy the system of DAE:*

$$R := \begin{cases} R_1 := X_{tt} + \lambda X = 0 \\ R_2 := Y_{tt} + \lambda Y = -g \\ R_3 := X^2 + Y^2 = 1 \end{cases} \tag{2.13}$$

The signature matrix for (2.13) with rows corresponding to R_1, R_2, and R_3, and columns corresponding to X, Y, and λ, is:

$$\begin{pmatrix} 2 & -\infty & 0 \\ -\infty & 2 & 0 \\ 0 & 0 & -\infty \end{pmatrix} \tag{2.14}$$

Based on this signature matrix we obtain $c = (0, 0, 2)$ and $d = (2, 2, 0)$. Recall that c_i

means the i-th equation needs to be differentiated c_i times ($c_i \geq 0$) and d_j is the highest order of derivative of u^j after the prolongation. Here $k_c = 2$, therefore we order c, d, and R as $c = (2, 0, 0)$, $d = (2, 2, 0)$, and $R := \{X^2 + Y^2 = 1, X_{tt} + \lambda X = 0, Y_{tt} + \lambda Y = -g\}$. Thus the block structure for this system is:

B_0	B_1	B_2
$X^2 + Y^2 = 1$	$2XX_t + 2YY_t = 0$	$2X_t^2 + 2Y_t^2 + 2XX_{tt} + 2YY_{tt} = 0$
		$X_{tt} + \lambda X = 0$
		$Y_{tt} + \lambda Y = -g$

Table 2.2: Block structure for one pendulum system.

Using the first prolongation method yields two missing constraints for this system, which are shown in B_1 and B_2 in the table above.

Chapter 3

Numerical algebraic geometry: A homotopy method

3.1 Introduction

In this chapter we describe the advances made in the numerical solution of systems of multivariate complex polynomial equations. For illustrative purposes we introduce a simple homotopy method and describe how it is used to find isolated solutions of polynomial equations. This method is described by Sommese and Wampler in [23]. Moreover, with Hauenstein and Bates they introduced the Bertini software [24, 25] which is an advanced efficient implementation in C. It is used in our package to find initial conditions for systems of DAEs.

Bertini is an efficient software package for computations in numerical algebraic geometry that use homotopy [16]. It is able to find zero dimensional (isolated) as well as represents positive dimensional solution sets of a system. Solutions can be computed with up to several hundred digits of accuracy. Sommese and Wampler [23] write "For polynomial problem arising in applications in science and engineering, the homotopy method works wonderfully well".

Verschelde [26] has also developed an efficient package for numerical algebraic geometry, PHCPack; in ADA. Other notable packages include Hom4ps, by T. Y. Li and collaborators [27] which finds isolated solutions of polynomial systems. It does not perform numerical algebraic geometry, but leads the field in creating start systems based on mixed volume, which have few diverging paths.

It is appropriate to begin with the definitions of "polynomial," and the "solution of polynomial systems" which are given below from [23].

Definition 3.1.1. *(Polynomials [23]) A function $f(x) : \mathbb{C}^N \to \mathbb{C}$ in N variables $x = (x_1, \dots, x_N)$ is a polynomial if it can be expressed as a sum of terms, where each term is the product of a coefficient and a monomial. In particular, each coefficient is a complex number, and each monomial is a product of variables raised to non-negative integer powers. If $N = 1$ we have an univariate polynomial and otherwise it is multivariate. Let $\alpha = (\alpha_1, \dots, \alpha_N)$ with each α_i a non-negative integer, and write monomials in the form $x^\alpha = \prod_{i=1}^N x_i^{\alpha_i}$. Then, a polynomial f is a function that can be written as*

$$f(x) = \sum_{\alpha \in I} a_\alpha x^\alpha \tag{3.1}$$

where I is a finite set and $a_\alpha \in \mathbb{C}$. The notation $f \in \mathbb{C}[x_1 \dots x_N] = \mathbb{C}[x]$ means f is a polynomial in the variables x with coefficients in $\mathbb{C}$. The total degree of a monomial x^α is $|\alpha| := \alpha_1 + \dots + \alpha_N$ and of the polynomial f(x) is $\max_{\alpha \in I : a_\alpha \neq 0} |\alpha|$. A polynomial system is written as $f : \mathbb{C}^N \to \mathbb{C}^n$ which has n polynomials in N variables.

Remark 3.1.2. *At times we may wish to represent f in other bases than the monomial bases used here, but this suffices for a definition.*

Definition 3.1.3. *(Zero Set, [23]) If $f(x) : \mathbb{C}^N \to \mathbb{C}^n$ is a system of multivariate polynomials, we use the notation $V(f) := f^{-1}(0)$ to represents the solution set of $f(x) = 0$:*

$$V(f) := f^{-1}(0) = \{x \in \mathbb{C}^N | f(x) = 0\}. \tag{3.2}$$

Here $V(f)$ is called the variety of f.

3.2 Homotopy continuation

We present a homotopy continuation method, which is a method for finding all zero-dimensional (isolated) solutions of a multivariate polynomial system. Commenting on alternatives to homotopy methods Sommese and Wampler [23] write: "For low-degree polynomials in one variable, one approach is to reformulate the problem as finding the eigenvalues of the companion matrix, which is convenient due to the wide availability of high quality software for solving eigenvalue problems [28]. For polynomial with high degree, divide-and-conquer techniques may be better [29]." Another alternative to homotopy methods is to use Newton's method. However, Newton's method gives us just one of the roots at a time. In addition, with a poor initial guess, Newton's method can fail. In contrast, homotopy methods are globally convergent and theoretically can find all roots in situations where all alternative methods fail.

The approach of homotopy continuation is to:

- Define a family of problems depending on parameters which contains the system we want to solve.
- Find the solution of the problem for some appropriate point in the parameter space.
- Follow the solution path from the point in the parameter space where we have the solution to the point which is related to the original problem we want to solve.

Consider a polynomial system $f : \mathbb{C}^N \to \mathbb{C}^n$ that we would like to solve by homotopy continuation. We can form a polynomial system g that is related to f but has known, or easily computable, zero set. The polynomials f and g form a homotopy, such as the linear homotopy $H(z,t) = f(1-t) + \gamma t g$ where $\gamma \in \mathbb{C}$ is a random number. I will explain in next section why we need this random γ. For a well-formed homotopy, there are continuous solution paths from the solutions of g to all solutions of f which can be followed using predictor-corrector methods. If $t = 1$ we have $H(z,1) = g$ whose roots are known and if $t = 0$ we have $H(z,0) = f$ whose roots need to be found. Therefore, we can start from $t = 1$ and track our path to $t = 0$. In the next section, I will give an algorithm for this method. Note it is conventional in the literature to start at $t = 1$ and go to $t = 0$, in opposition to the apparently more natural order $0 \leq t \leq 1$. This is because most of the interesting behavior occurs at the final time and representing t by $t = 0$ is preferable for this reason.

A big advantage of this method is that it can easily be parallelized, which means if the starting polynomial g has several solutions, the corresponding solution paths may be tracked on different processors. There is a Message Passing Interface (MPI) version of Bertini whose implementation is a parallelized homotopy method and can be used for big systems of polynomials.

3.2.1 A generic illustrative homotopy algorithm

Consider a univariate polynomial $p(z) := z^d + a_1 z^{d-1} + \cdots + a_d$. We want to show how continuation can be used to find zero set of this polynomial. One of the great advantages of working with the complex numbers is that, by the fundamental theorem of algebra, we know $V(p)$ has exactly d points, including multiplicities. Therefore we can consider a starting system with d known roots and follow construction paths to the roots of $p(z)$. We know how to solve $z^d - 1 = 0$: its d roots are

$$z_k = e^{k2\pi\sqrt{-1}/d} \quad for \quad k = 0, \ldots, d-1. \tag{3.3}$$

We can define a homotopy by

$$H(z,t) := t(z^d - 1) + (1-t)p(z). \tag{3.4}$$

When $t = 1$ we have $H(z,1) = z^d - 1$, with known roots, and when $t = 0$, we have the system $H(z,0) = p(z)$, which we want to solve. We need to track the solution paths as t goes from 1 to 0. The question is how can we numerically do this? The solution paths z_i needs to satisfy $H(z_i(t),t) \equiv 0$ for all t. Therefore:

$$0 \equiv \frac{dH(z_i(t),t)}{dt} = H_z(z_i(t),t)\frac{dz_i(t)}{dt} + H_t(z_i(t),t), \tag{3.5}$$

where $H_z(z,t)$ and $H_t(z,t)$ are the partial derivatives of $H(z,t)$ with respect to z and t respectively. Thus

$$\frac{dz_i(t)}{dt} = -H_z(z_i(t),t)^{-1}H_t(z_i(t),t). \tag{3.6}$$

This is an ordinary differential equation for $z_i(t)$, with initial values at $t = 1$ given by $z_i(1)$ which are starting points for this homotopy. The roots we seek are the values of $z_i(0)$.

There are some deficiencies for this method which we now discuss. For some polynomials, $H(z,t)$ can be constant for some t. Also, $H(z,t)$ may have roots with multiplicity more than one. For example for $p(z) = 5 - z^2$, the homotopy is $H(z,t) = t(z^2-1) + (1-t)(5-z^2)$. These are difficulties at t=$\frac{5}{6}$, because $H(z, \frac{5}{6}) = (\frac{2}{3})z^2$ has a double root and at $t = \frac{1}{2}$, because $H(z, \frac{1}{2}) = 2$ has no solution. Using the following "quick fix," which is called *"gamma trick"* in [30] enables us to avoid these cases. In particular select a random angle $\theta \in [0, 2\pi]$. As usual let $i = \sqrt{-1}$. Then

$$H(z,t) = te^{i\theta}(z^d - 1) + (1-t)p(z) = 0. \tag{3.7}$$

Now, due to the complex factor $e^{i\theta}$, the paths are well defined for all $t \in [0,1]$; with probability one we do not face the difficulties above [23]. One further random angle $\beta \in [0, 2\pi]$ is required to define a suitable homotopy

$$H(z,t) = te^{i\theta}(z^d - e^{di\beta}) + (1-t)p(z) = 0. \tag{3.8}$$

For more details see [23]. Note that at $t = 0$, we have the same target points as before ($\beta = 0$). The non-singular path tracking algorithm that we used in our Maple program can be summarized as follows. To be general we can consider a system with N polynomials, $H(z,t) : \mathbb{C}^N \times \mathbb{R} \to \mathbb{C}^N$. In what follows $H_z = \frac{\partial H}{\partial z}$ is the $N \times N$ Jacobian matrix and $H_t = \frac{\partial H}{\partial t}$ is an $N \times 1$ matrix.

Simple homotopy Algorithm 3.1:

- **Given:** System of equations, $H(z,t) : \mathbb{C}^N \times \mathbb{R} \to \mathbb{C}^N$, initial point z_0 at $t_0 = 1$ such that $H(z_0, t_0) \simeq 0$, and initial step length h; $m = 5$ (# of successive corrections); $k := 0$ (a counter to count # of successful correction steps).

- **Find:** Approximation of a homotopy path $(z(t), t)$ from $t = 1$ to $t = 0$ with $H(z_0, t_0) \simeq 0$.

- **Procedure:** For $i = 0$ while $t_i > 0$ do

1. **Prediction:** Predict solution w_0 by solving (3.9) implicitly (by an Euler method), then decrease t by h:

$$H_z(z_i, t_i) \cdot w_0 := H_z(z_i, t_i) \cdot z_i - H_t(z_i, t_i) \cdot h, \tag{3.9}$$

$$t := t_i - h. \tag{3.10}$$

2. **Correction:** Find the solution w of $H(z, t) = 0$ using Newton's method with start value w_0 and fixed t.

3. **Update:** If (successful correction step)

$$|H(w, t)| < [H(w_0, t)]^2 \tag{3.11}$$

then

$(z_{i+1}, t_{i+1}) := (w, t)$

$i := i + 1$

$k := k + 1$

If $k = m$ then $h := 2 * h$

Go to prediction step 1 to calculate w_0 for the obtained i.

```
else
        $k := 0$
        $h := h/2$
        $t := t_i + h$
        Repeat prediction step with this half size $h$ and $(z_i, t_i)$.
```

Refine endpoint: At $t_n = 0$, correct z_i to high accuracy with Newton's method.

We make some notes about Algorithm 3.1.

In the **Update** step in the Algorithm 3.1:

- The correction step is considered successful in Algorithm 3.1 if (3.11) is satisfied.
- We can cut step length, h, in half on one failure of the correction and to double it, if m successive corrections at the current step length have been successful. Following Sommese and Wampler in [23], we used $m = 5$ in our algorithm in Maple.

Reason for Euler and Newton as prediction and correction method: Expanding the homotopy function in Taylor series yields:

$$H(z + \Delta z, t + \Delta t) = H(z,t) + H_z(z,t)\Delta z + H_t(z,t)\Delta t + \dots, \tag{3.12}$$

If we have a point (z_1, t_1) close to the path, so that, $H(z_1, t_1) \simeq 0$, then $H(z_1 + \Delta z, t_1 + \Delta t) \simeq 0$. Then to first order

$$H_z(z_1, t_1)\Delta z + H_t(z_1, t_1)\Delta t = 0 \tag{3.13}$$

Of course, we can explicitly solve $\Delta z = -H_z^{-1}(z_1, t_1)H_t(z_1, t_1)\Delta t$ from (3.13). However, numerically it is much cheaper to solve the implicit form at each time step than first explicitly symbolically invert H_z. From (3.12) when Δz and Δt are small we can approximate Δz as the solution of

$$\begin{aligned} &H(z_1,t_1) + H_z(z_1,t_1)\Delta z = 0 \Rightarrow \\ &\Delta z = -H_z^{-1}(z_1,t_1)H(z_1,t_1) \end{aligned} \tag{3.14}$$

Equation (3.13) is an Euler prediction step and (3.14) is a Newton correction step.

One strategy to stay close to the path [23] is to keep the h small and the number of iterations in the correction (3.14) small (say ≤ 3). Staying close to the path also helps avoid path jumping. Moreover, in our program in Maple to save computation time, h is bigger at the beginning of the path ($1 \geq t \geq 0.1$), then near the end h is smaller for ($0.1 \geq t \geq 0$). For simplicity but not necessarily efficiency, my program was implemented also using fixed small step size.

3.2.2 Homotopy for other functions

Homotopy approaches are useful not only for polynomial solving but also for solving more general type of systems. The DAE systems we generated all using MapleSim are mostly trigonometric equations which can be converted to polynomials. For example, for equations involving $\sin(\theta)$ and $\cos(\theta)$, we can introduce new variables, $s_\theta := \sin(\theta)$ and $c_\theta := \cos(\theta)$. Then the equation becomes polynomial subject to $s_\theta^2 + c_\theta^2 = 1$. When solutions for s_θ and c_θ are found, the values of θ are easily calculated. However not all equations involving trigonometric functions can be changed to polynomials. Examples include $x + \sin(x) = 0$ and $\sin(x) + \sin(xy) = 0$.

3.2.3 Examples of the homotopy algorithm

In this section we use our simple homotopy algorithm to illustrate homotopy solving.

Example 3.2.1. *The equation* $f(z) = 0$ *in* (3.15) *has four distinct roots.*

$$f(z) := (z-3)(z-\frac{1}{2}-5i)(z-i)(z-7i) = 0. \tag{3.15}$$

The algorithm we implemented in Maple in Section 3.2.1 yields the homotopy paths shown in Figure 3.1 and the solution shown in Table 3.1.

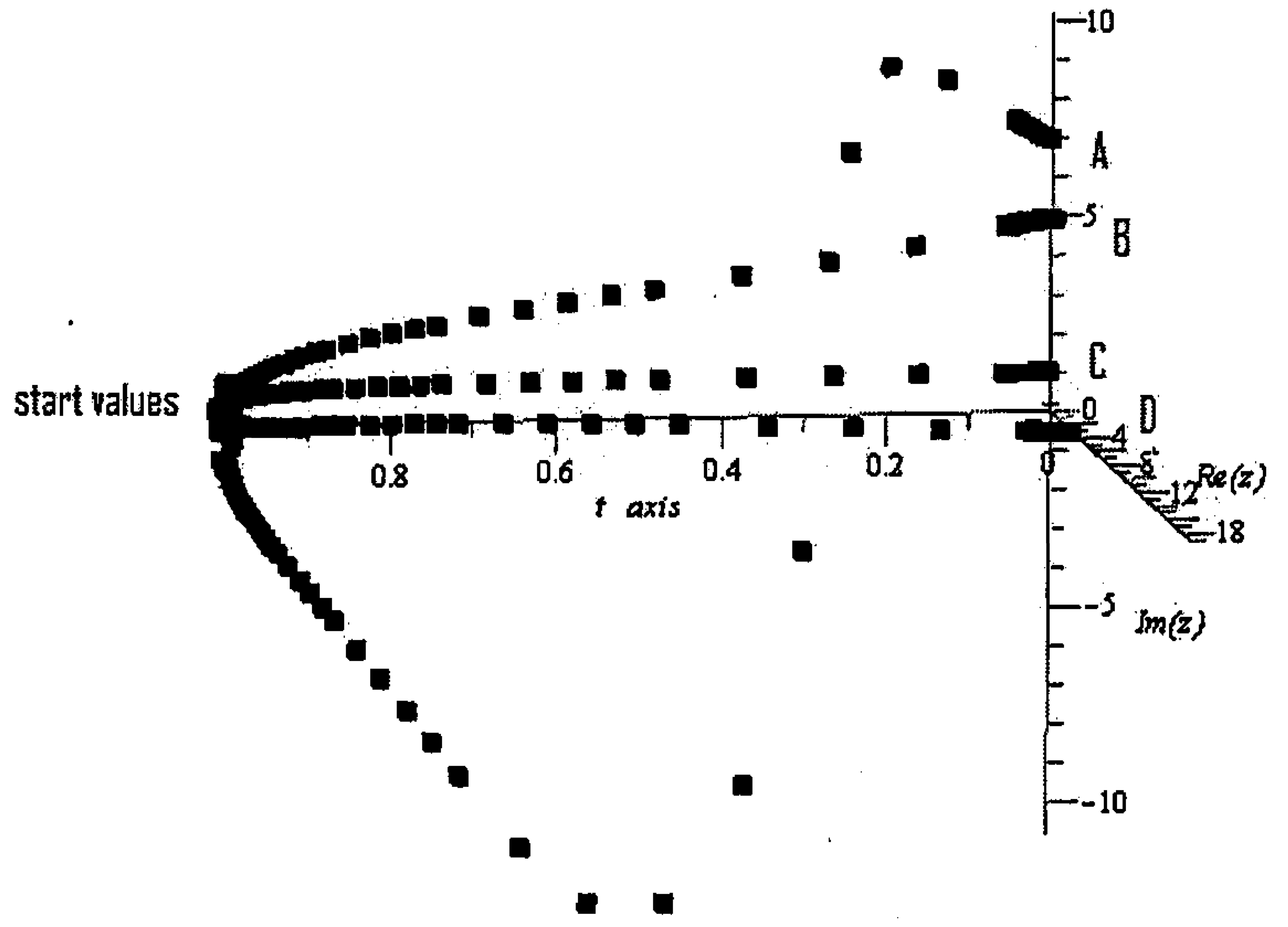


Figure 3.1: Homotopy solution paths for $f(z) := (z-3)(z-\frac{1}{2}-5i)(z-i)(z-7i) = 0$.

In Figure 3.1, the 4 start values are at left of the figure (i.e. at $t = 1$) and the 4 target solutions are at right ($t = 0$). In particular the 4 solutions A, B, C, D are $7i$, $\frac{1}{2}+5i$, i, and 3, respectively.

Roots of start function: $e^{ia}(z^4 - e^{ib})$ Random $a, b \in (0..2\pi)$	Roots of target function: $(z-3)(z-\frac{1}{2}-5i)(z-i)(z-7i)$
$-0.9368574672 + 0.3497114326i$	$0. + 7.000000000i$
$-.3497114324 - .9368574673i$	$3.000000000 + 0i$
$.3497114320 + .9368574675i$	$.5000000000 + 5.000000000i$
$.9368574674 - .3497114322i$	$0. + 1.000000000i$

Table 3.1: Start points and target roots of $(z-3)(z-\frac{1}{2}-5i)(z-i)(z-7i) = 0$ by our homotopy solver.

Example 3.2.2. *The function* $g(z)$

$$g(z) := (z-3)^6 \tag{3.16}$$

in Figure 3.2 has root 3 *with multiplicity* 6. *The homotopy Algorithm 3.1 yields the homotopy paths shown in Figure 3.2 and corresponding solution shown in Table 3.2.*

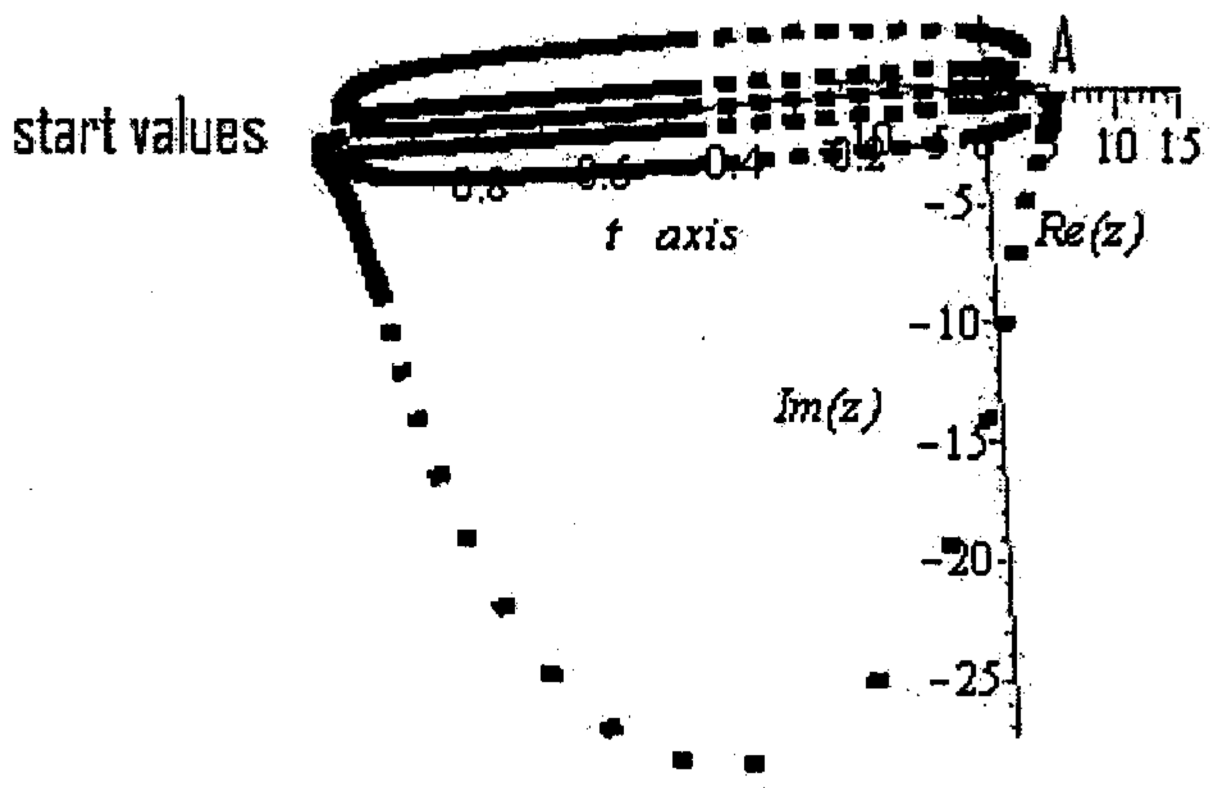


Figure 3.2: Homotopy solution paths for $g(z) := (z-3)^6 = 0$.

In Figure 3.2, the 6 start values are at left of the figure (i.e. $t = 1$) and the 6 target solutions $z = 3$ are at right ($t = 0$).

Roots of start function: $e^{ia}(z^6 - e^{ib})$ Random $a, b \in (0..2\pi)$	Roots of target function: $(z-3)^6$
$-.9861980826 - .1655697494i$	$3.000094330 - 0.1399863868 \times 10^{-4}i$
$-.6364866497 + .7712877185i$	$3.000025356 + 0.7583900569 \times 10^{-4}i$
$-.3497114324 - .9368574673i$	$3.000005939 - 0.7347802653 \times 10^{-4}i$
$.3497114320 + .9368574675i$	$2.999964105 + 0.4427595856 \times 10^{-4}i$
$.6364866493 - .7712877189i$	$2.999958763 - 0.3564671291 \times 10^{-4}i$
$.9861980823 + .1655697508i$	$2.999950164 + 0.4104603269 \times 10^{-5}i$

Table 3.2: Start points and target roots of $(z-3)^6 = 0$ by our homotopy solver.

Example 3.2.3. *The solutions of*

$$h(z) := (z + 15)^2(z - 48.9i)(z - 36.9 + 10.3i)^2(z - \sqrt{2}) = 0 \tag{3.17}$$

is combination of real and complex roots with multiplicity one or more than one (see Figure 3.3 and Table 3.3).

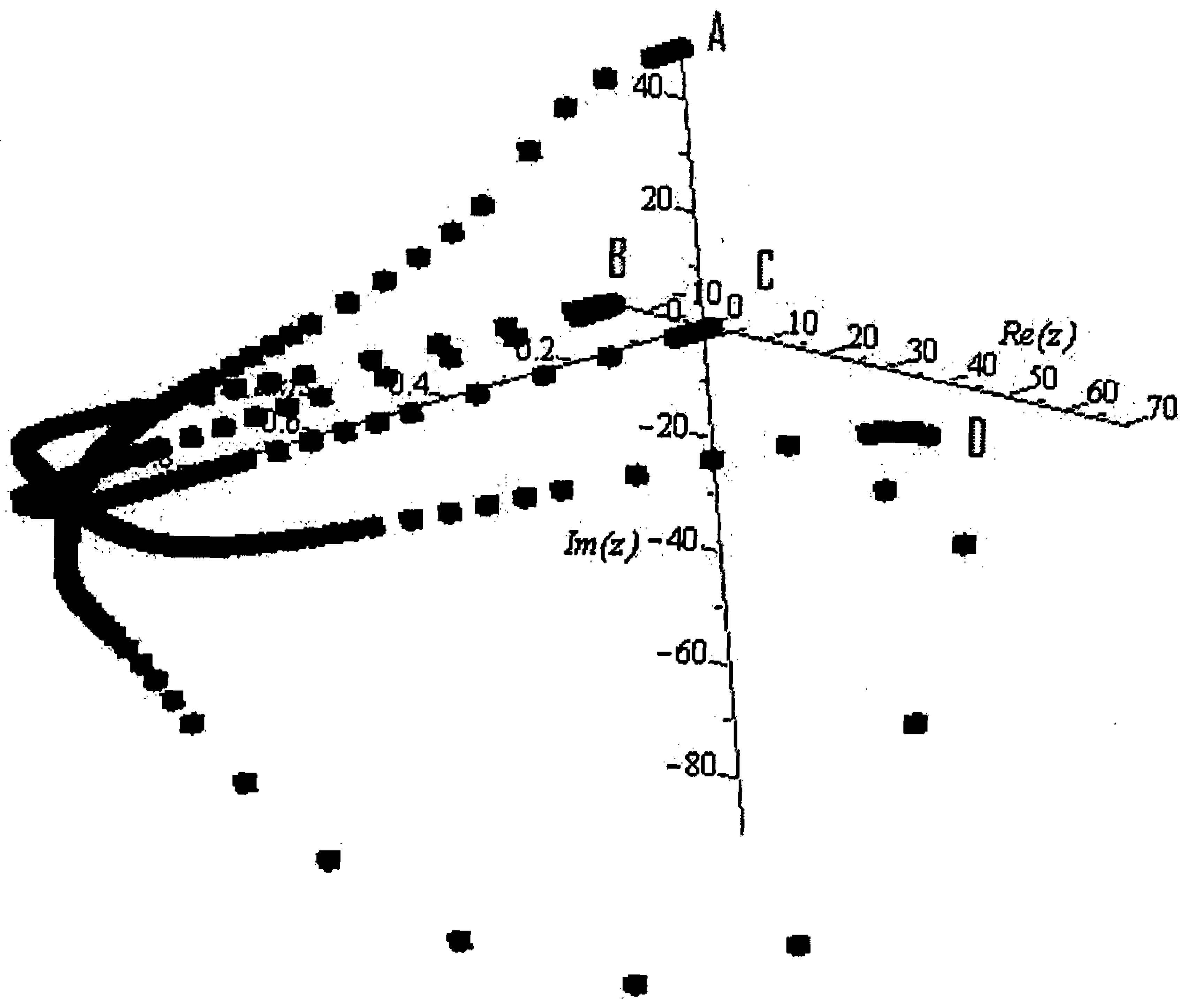


Figure 3.3: Homotopy solution paths of $(z+15)^2(z-48.9i)(z-36.9+10.3i)^2(z-\sqrt{2}) = 0$.

Start values at left of Figure 3.3, $t = 1$ are shown around a circle. At the right there are six roots. Two of the roots have multiplicity two (B and D in Figure 3.3).

Roots of start function: $e^{ia}(z^6 - e^{ib})$ Random $a, b \in (0..2\pi)$	Roots of target function: $(z+15)^2(z-48.9i)(z-36.9+10.3i)^2(z-\sqrt{2})$
$-0.9861980826 - 0.1655697494i$	$-14.99999959 - 3.912940075 \times 10^{-7}i$
$-0.6364866497 + 0.7712877185i$	$-15.00000040 + 3.838379100 \times 10^{-7}i$
$-0.3497114324 - 0.9368574673i$	$36.89999939 - 10.29999981i$
$0.3497114320 + 0.9368574675i$	$-10^{-28} + 48.90000000i$
$0.6364866493 - 0.7712877189i$	$36.89999939 - 10.29999981i$
$0.9861980823 + 0.1655697508i$	$1.414213562 + 0.i$

Table 3.3: Start points and target roots of $(z+15)^2(z-48.9i)(z-36.9+10.3i)^2(z-\sqrt{2}) = 0$.

3.2.4 Discussion

An alternative to using the prediction-correction methods is to only using prediction steps. Thus we only numerically solve the prediction ODE (3.6). We compared these two strategies on some examples. We used dsolve/numeric in Maple with Fehlberg fourth-fifth order Runge-Kutta method (rkf45) and `absolute error`(abserr) $=10^{-7}$ and `relative error` (relerr) =0.000001. On our example the results were obtained more quickly but not as accurately as results from our prediction and correction method with `tolerance`=0.000001. For the example below `dsolve/numeric` computing time for finding all roots is =2.797 seconds but the prediction and correction computing time is =9.391 seconds.

Result by `dsolve/numeric`:	Result by prediction and correction:
$-14.9991594040901 - 0.112989803875263 \times 10^{-2}i$	$-14.99999959 - 3.912940075 \times 10^{-7}i$
$-15.0001157551416 + 0.165970043007754 \times 10^{-2}i$	$-15.00000040 + 3.838379100 \times 10^{-7}i$
$36.9241045638584 - 10.3081715489427i$	$36.89999939 - 10.29999981i$
$0.2169589718836 \times 10^{-5} + 48.8999997696319i$	$-10^{-28} + 48.90000000i$
$36.8924717622899 - 10.2977793703036i$	$36.89999939 - 10.29999981i$
$1.41421355480710 - 1.41056735789722 \times 10^{-8}i$	$1.414213562 + 0.i$

Table 3.4: Comparison between `dsolve/numeric` and prediction-correction for solving $(z+15)^2(z-48.9i)(z-36.9+10.3i)^2(z-\sqrt{2}) = 0$.

3.3 Positive dimensional solutions

Homotopy continuation methods provide efficient numerical algorithms to compute exact approximations of all zero dimensional (isolated) solutions of polynomial systems. However, numerical algebraic geometry treats both zero-dimensional and positive dimensional solution sets. The fundamental parts of the positive dimensional solution set of equations are its irreducible components. These are the algebraic subsets Z of the solution set. The solution sets are separated into a union of irreducible components, so that none of them is contained in the union of the others. Among those components there may be some zero dimensional (isolated) and some positive dimensional ($dim > 0$) solution sets. So-called 'witness sets' are the basic data structure used to describe positive dimensional solution sets.

Consider an irreducible component Z of a system of polynomials $f(x) = 0$. A witness set for Z consists of triple (f, L, W), where L is a random linear space of dimension complementary to dimension of Z and where W is a set of points such that $W := Z \cap L$. The Bertini software is able to give zero dimensional and positive dimensional solutions for a system of $N \times n$ polynomials [31, 32]. In those works it was shown that we can represent a positive dimensional by approximate (so-called witness) points, which are obtained as intersection points of the set with a linear space of complementary dimension. Here we want to illustrate witness sets and how to use them to describe positive dimensional solution sets.

3.3.1 A numerical irreducible decomposition

I start with a motivating example from [31], which is a system of polynomials with several solution components, of different dimensions and degrees. Later, we define the witness set, which is used to represent pure dimensional solution sets of polynomial systems numerically.

Example 3.3.1. *Suppose we wish to find the solution set of* $f : \mathbb{C}^3 \to \mathbb{C}^3$*:*

$$f(x,y,z) = \begin{bmatrix} (y-x^2)(x^2+y^2+z^2-1)(x-0.5) \\ (z-x^3)(x^2+y^2+z^2-1)(y-0.5) \\ (y-x^2)(z-x^3)(x^2+y^2+z^2-1)(z-0.5) \end{bmatrix} = 0 \tag{3.18}$$

In this factored form we can calculate the decomposition of the solution set $z = f^{-1}(0)$ into irreducible solution components, as below:

$$Z = Z_2 \cup Z_1 \cup Z_0 = \{Z_{21}\} \cup \{Z_{11} \cup Z_{12} \cup Z_{13} \cup Z_{14}\} \cup \{Z_{01}\} \tag{3.19}$$

Where

- Z_{21} is the sphere $(x^2+y^2+z^2-1) = 0$,
- Z_{11} is the line $(x = 0.5, z = (0.5)^3)$,
- Z_{12} is the line $(x = \sqrt{0.5}, y = 0.5)$,
- Z_{13} is the line $(x = -\sqrt{0.5}, y = 0.5)$,
- Z_{14} is the twisted cubic $(y - x^2 = 0, z - x^3 = 0)$,
- Z_{01} is the point $(x = 0.5, y = 0.5, z = 0.5)$.

Here Z_{21} is 2-dimensional, Z_{01} is 0-dimensional and Z_{11}, Z_{12}, Z_{13}, and Z_{14} are all 1-dimensional.

3.3.2 Witness sets

Now we define witness sets using [31]. Consider $f := \mathbb{C}^N \to \mathbb{C}^n$ of n polynomials $f := \{f_1, f_2, \dots, f_n\}$ and N unknowns $\mathbf{x} = (x_1, x_2, \dots, x_N)$. Recall we represent the solution set of f by

$$V(f) = \{\mathbf{x} \in \mathbb{C}^N | f(\mathbf{x}) = 0\}. \tag{3.20}$$

Suppose $X \subset V(f) \subset \mathbb{C}^N$ is a pure dimensional algebraic set of dimension i and degree d. By pure, we mean all components of the set have the same dimension. A witness set for X is a data structure which consists the system f, a linear space $L \subset \mathbb{C}^N$ of codimension i, and a set of d points $X \cap L$.

When X is not pure dimensional, a witness set for X can be separated into a list of witness sets, one for each dimension. X has a unique decomposition into irreducible components. Therefore, a witness set for X has a decomposition into the corresponding irreducible witness sets. This is called a numerical irreducible decomposition of X.

Therefore, the irreducible decomposition of the solution set Z in (3.19) is:

$$[W_2, W_1, W_0] = [[W_{21}], [W_{11}, W_{12}, W_{13}, W_{14}], [W_{01}]], \tag{3.21}$$

where the W_i are witness sets for pure dimensional components, of dimension i, partitioned into witness sets W_{ij}'s corresponding to the irreducible components of Z. Thus for this example we have:

- W_{21} contains two points on the sphere $(x^2+y^2+z^2-1) = 0$, cut by a random line,
- W_{11} contains one point on the line $(x = 0.5, z = 0.5^3)$, which is cut by a random plane,
- W_{12} contains one point on the line $(x = \sqrt{0.5}, y = 0.5)$, which is cut by a random plane,
- W_{13} contains one point on the line $(x = -\sqrt{0.5}, y = 0.5)$, which is cut by a random plane,
- W_{14} contains three points on the twisted cubic $(y - x^2 = 0, z - x^3 = 0)$, which is cut by a random plane,
- W_{01} is still just this point $(x = 0.5, y = 0.5, z = 0.5)$.

The witness sets W_{ij} consist of witness sets $\mathbf{W} = \{i, f, L, \mathrm{x}\}$, for $\mathrm{x} \in Z_{ij} \cap L$, where L is a random linear subspace of codimension i (in this example of dimension $3 - i$).

Moreover, as we can observe here $\#W_{ij} = deg(Z_{ij}) = \#(Z_{ij} \cap L)$. If we ask Bertini to solve this polynomial system, for example, as a result it gives a text file which contains $\#W_{ij}$ witness points for each components of various dimension. Furthermore, it is able to show the dimension and degree of each component as well as singular and non-singular solution sets. We will show one example of singular solutions from Bertini in Chapter 6.

Chapter 4

Main algorithm for MapleSim generated models

4.1 Introduction to the MapleSim software

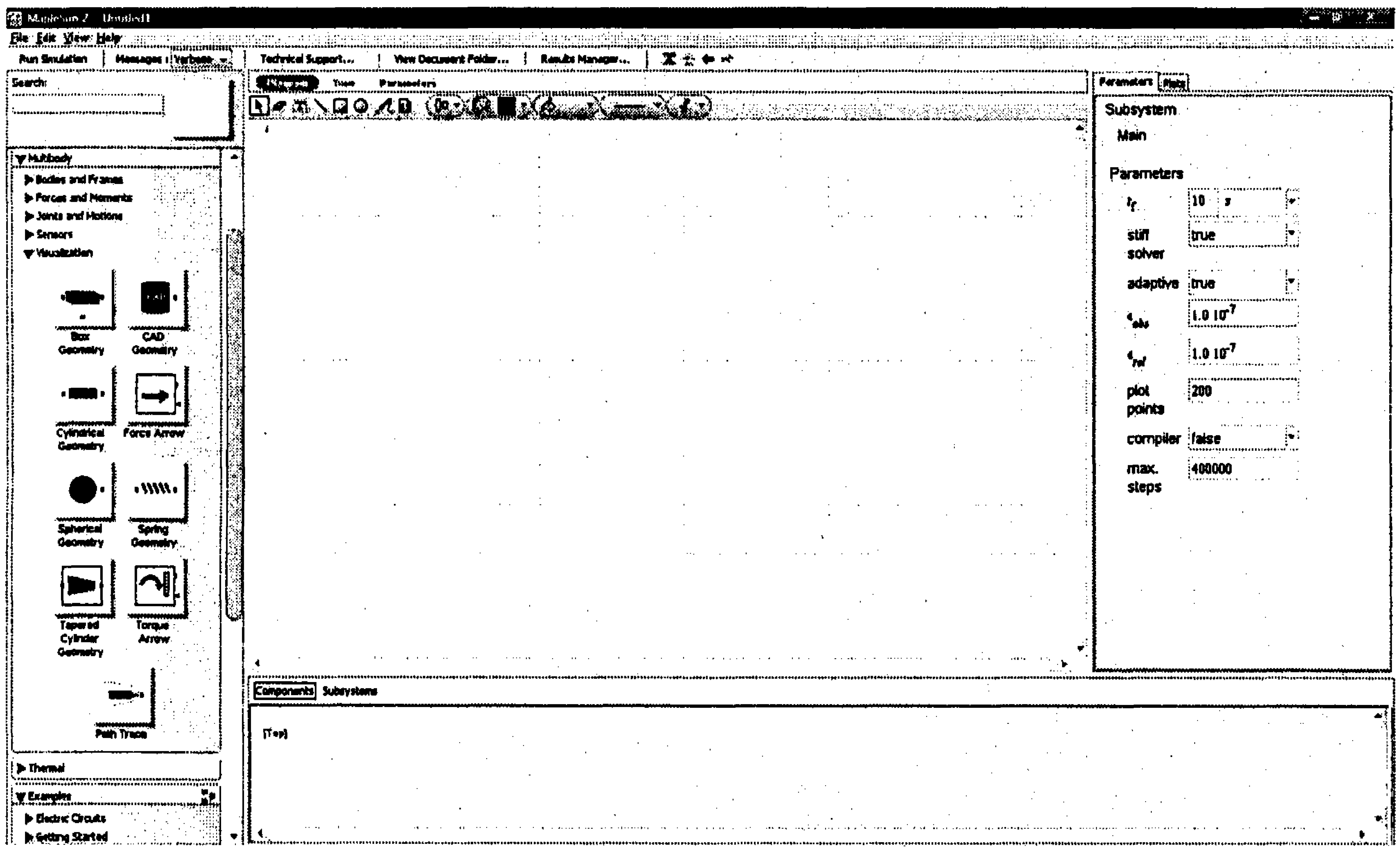
The main goal of this thesis is to efficiently apply the fast numeric geometric techniques discussed in Chapter 2 to a wide variety of physical models generated by the MapleSim software [3].

MapleSim is a high-performance multi domain modeling and simulation tool which uses components from various engineering fields. We can build component diagrams that represent such systems in a graphical form using MapleSim's drag and drop interface. When we obtain such a system, we can simulate its behavior using numeric approaches. Unlike other simulation environments MapleSim exploits Maple's symbolic features.

MapleSim uses the symbolic and numeric capabilities of Maple to generate the mathematical models that simulate the behavior of the physical system. We can, therefore, derive the system's equations in Maple and create concise and numerically efficient models. MapleSim enables the creation of the physical components the user needs without forcing the user to write complex code and mathematical equations. Other physical modeling tools however, demand that we write programming code to create physical components. Consequently, MapleSim can cut project time and cost.

MapleSim also provides control theory tools, which give the ability to optimize and control parameters to minimize rise time and reduce overshoot. Using 3D visualizer features, MapleSim enables us to obtain an immediate insight into behavior of our models.

A typical MapleSim workspace is given in Figure 4.1 below:



Subsystem
Main

Parameters

t_f	10 s
stiff solver	true
adaptive	true
ϵ_{abs}	$1.0\ 10^{-7}$
ϵ_{rel}	$1.0\ 10^{-7}$
plot points	200
compiler	false
max. steps	400000

Figure 4.1: MapleSim environment.

To make a model,'we need to use appropriate MapleSim library icons from the left palette in Figure 4.1. Then we drag and drop the relevant icons on the workspace (the center panel of Figure 4.1). Next we connect the icons in the desired manner. We can use the Parameters pane in the right hand side of Figure 4.1 to change system parameters such as final time, relative error, absolute error, and others.

Moreover, we have the option to choose initial conditions here while making our model. When we click on some components, an *IC* (initial condition) appears in the parameter pane. There are three options for *IC*: `Ignore`; `Treat as guess`; `Strictly Enforced`. `Ignore` means MapleSim ignores the initial conditions of this component given by user, and uses default values which are zero in most of the cases. `Treat as guess` means MapleSim uses the initial condition for this component if it is consistent with other initial values. Otherwise it replaces the initial condition with a consistent one. Finally, `Strictly Enforced` means MapleSim uses the initial condition which is given by the user. If the condition is not consistent it sends an error message to the user.

If we want to measure a quantity for a component such as an angle, speed or acceleration, we need to add a 'probe' to that component. Those values are graphically displayed after simulation.

The best way to derive equations is using the Modelica language which is accessible in MapleSim work space. The Modelica language [33] is a multi-domain modeling language for component-oriented modeling of complex systems. It can model systems containing mechanical, electrical, hydraulic, thermal, control, electric power or process-oriented subcomponents. The Modelica Standard Library contains about 920 generic model components and 615 functions in various domains. Modelica is an open standard for describing physical models and components. Many components in MapleSim are from the Modelica Standard Library. Because these Modelica components have been developed over many years and validated by industry, engineers can have confidence that their MapleSim model will provide an accurate representation of the system. The Modelica file of a MapleSim model can be exported to MapleSim after the model's creation. To obtain the DAE system of the model, we have to open the Modelica file in a Maple worksheet. MapleSim equations after simulation are simplified. However, in the Modelica file they are unsimplified and raw.

Input and output values for the system, can be visualized using Maple. Also MapleSim does index reduction of DAE [6]. Index reduction enables the inclusion of missing constraints.

MapleSim is comparable with Matlab's Simulink. Simulink is also a recent environment for multi-domain simulation and model-based design for dynamic systems integrated with Matlab. In some cases MapleSim differs from this software. We mention just one of differences here [3]. If we need a component which is not available in the MapleSim libraries we can create it in Maple with easy-to-use templates in which we only have to write down the equations that define the component's dynamics. No other modeling products have this ability. In Chapter 5, we give an example of this feature.

4.2 Algorithm for MapleSim generated models

Our algorithm in order to apply fast numeric geometric techniques with MapleSim models is:

Algorithm 4.1 : Fast numeric-geometric techniques for MapleSim

- Given: A model created from MapleSim.
- Find: Numerical solution of system of DAEs for this model.
- Procedure:

1. **Model in MapleSim:** Make a model in MapleSim, do simulation and export Modelica file.
2. **In Maple's worksheet:** Open Modelica file and save it.
3. **Simplification:** Derive the DAE system and its variables. Change variables physical names to the mathematical names such as $X[i]$ when $i = 1 \ldots$ number of variables. Simplify the system.
4. **Fast Prolongation:** Calculate c_i and d_i as described in Section 2.3, by fast prolongation package.
5. **Initial Time:** Set initial time according to the model in MapleSim.
6. **Initial Conditions:** There are two different cases for this part.
   If $c_i = 0$ for all i, we use initial conditions from Modelica file if there is any, or a random initial condition.
   If $\exists i, c_i \neq 0$, a block structure is calculated and we create a bottom-up block by the two steps below:
   (a) **Jet Variables:** Change jet variables to produce a file for Bertini (e.g. use *xtttt* rather than x_{tttt}).

(b) **Bottom-Up Block:** Use bottom-up substitution in block triangular structures [1, 2] to find consistent initial conditions:

Loop: For bottom block to top block repeat:

i. Solve system of this block by Bertini.

ii. Plug Bertini's solution into the next block until all blocks are solved.

7. **Solution:** Use dsolve/numeric to solve the system of DAE from this model with initial conditions from step 6.

8. **Comparison:** Compare our solution to that obtained by MapleSim.

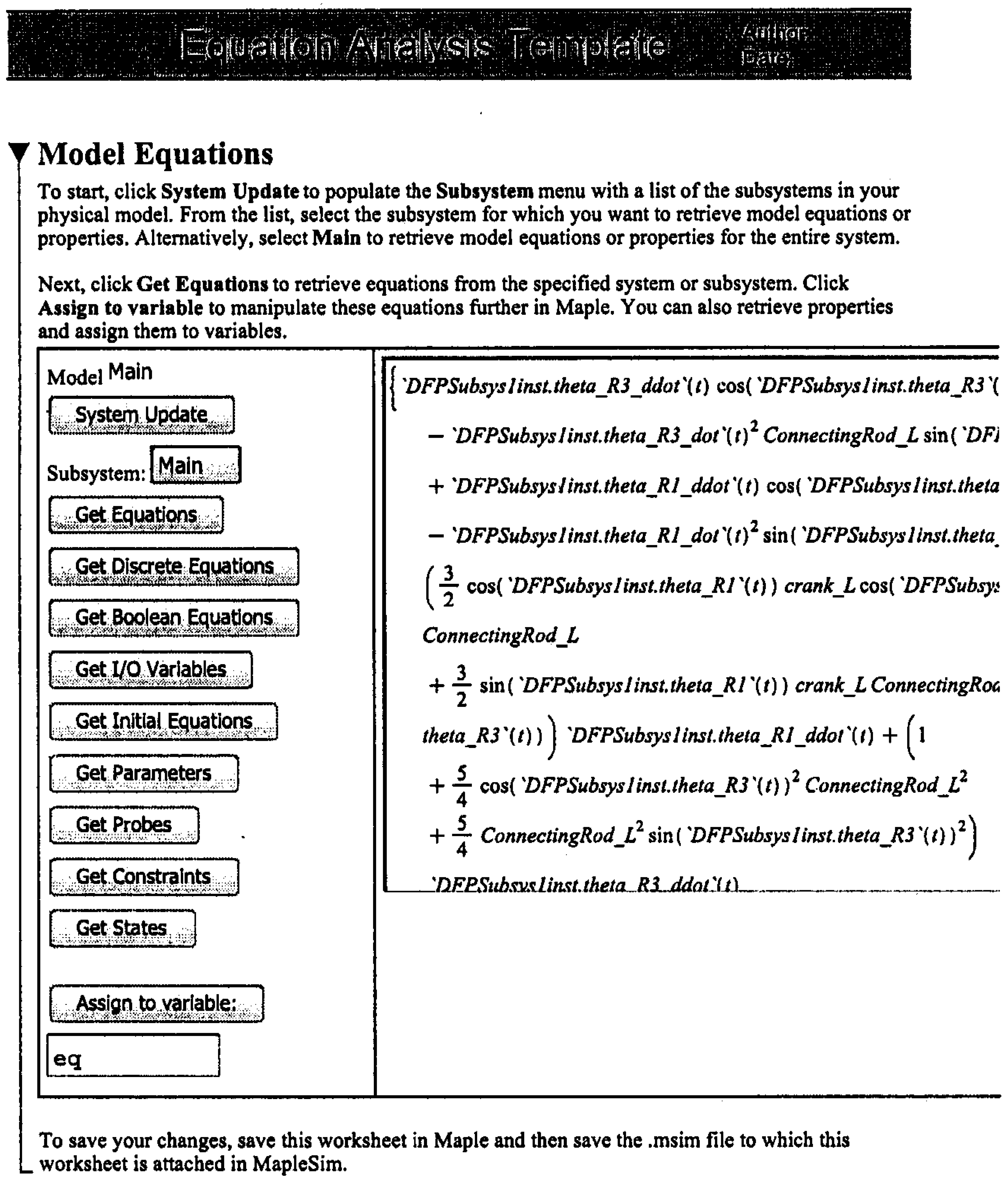
We made an automatic package for Algorithm 4.1. For most models all that is required is to input the Modelica file.

4.2.1 Simplification

The large DAE systems of MapleSim models usually contains many simple equations. MapleSim uses these simple equations to reduce the number of variables and simplify the system of DAEs which can be accessed from a Maple worksheet. Figure 4.2 gives an example of such a simplified system.

However, our goal is to apply alternative procedures to the unsimplified DAE system. While using an automatic command for generating the unsimplified system for a special model we found a bug which was reported to MapleSoft Corporation. This led us to use instead the Modelica file for the unsimplified system of equations. We derive the Modelica file and open it in a Maple Worksheet in order to obtain the unsimplified system of equations in Maple. The Modelica file includes all information about the system such as names, parameters, variables, constants and the system of DAE.

Firstly, we change variable names from Modelica names to a form usable by our Maple programs. Variable names in Modelica are physical names such as "Main.probe2.phi", for example which means the angle of the second probe attached in the main system. Moreover, in Modelica they are not shown as functions of t. Such variable names are changed to $Xi(t)$, where i is from 1 to the number of variables.



Model Equations

To start, click **System Update** to populate the **Subsystem** menu with a list of the subsystems in your physical model. From the list, select the subsystem for which you want to retrieve model equations or properties. Alternatively, select **Main** to retrieve model equations or properties for the entire system.

Next, click **Get Equations** to retrieve equations from the specified system or subsystem. Click **Assign to variable** to manipulate these equations further in Maple. You can also retrieve properties and assign them to variables.

$\Big\{$ `DFPSubsys1inst.theta_R3_ddot`$(t)\cos($`DFPSubsys1inst.theta_R3`$($

$-$ `DFPSubsys1inst.theta_R3_dot`$(t)^2$ $\mathit{ConnectingRod\_L}\sin($`DF`

$+$ `DFPSubsys1inst.theta_R1_ddot`$(t)\cos($`DFPSubsys1inst.theta`

$-$ `DFPSubsys1inst.theta_R1_dot`$(t)^2\sin($`DFPSubsys1inst.theta_`

$\Big(\frac{3}{2}\cos($`DFPSubsys1inst.theta_R1`$(t))\ \mathit{crank\_L}\cos($`DFPSubsy`

$\mathit{ConnectingRod\_L}$

$+\frac{3}{2}\sin($`DFPSubsys1inst.theta_R1`$(t))\ \mathit{crank\_L}\ \mathit{ConnectingRo}$

$\mathit{theta\_R3}$`(t))`$\Big)$ `DFPSubsys1inst.theta_R1_ddot`$(t)+\Big(1$

$+\frac{5}{4}\cos($`DFPSubsys1inst.theta_R3`$(t))^2\ \mathit{ConnectingRod\_L}^2$

$+\frac{5}{4}\ \mathit{ConnectingRod\_L}^2\sin($`DFPSubsys1inst.theta_R3`$(t))^2\Big)$

`DFPSubsys1inst.theta_R3_ddot`(t)

To save your changes, save this worksheet in Maple and then save the .msim file to which this worksheet is attached in MapleSim.

Figure 4.2: Equation Analysis Template in equivalent Maple worksheet.

Next the "_msim_der" used by Modelica for derivative is replaced by Maple's derivation notation. Now we are left with a DAE system in Maple notation. For example:

$$\begin{aligned} &X13(t) = 0, X17(t) = 0, X2(t) - X10(t) = 0, \ldots \\ &X1(t) - \frac{d}{dt}X3(t) = 0, X4 - \frac{d}{dt}X6(t) = 0, \ldots \\ &C_1 X34(t)) + C_2 \cos(X25(t)) - X22(t) = 0, \ldots \end{aligned} \tag{4.1}$$

Such systems contain many 2-term equations of the form $u - v = 0$ or $w - \frac{dz}{dt} = 0$ (1st and 2nd rows in (4.1)). Such equations can be used to simplify the system. We use classical symbolic simplification of the 2-term equations with respect to a ranking $\prec$ so that $u \prec v$, $w \prec \frac{dz}{dt}$. Applying Maple's DEtools[rifsimp] to the 2-term equations, and substitution of the result in the rest of the system can dramatically simplify systems like (4.1). In particular a system with many variables and equations can be simplified to a system with far fewer variables and equations.

This example (see Figure 4.3) will be used to illustrate the algorithm.

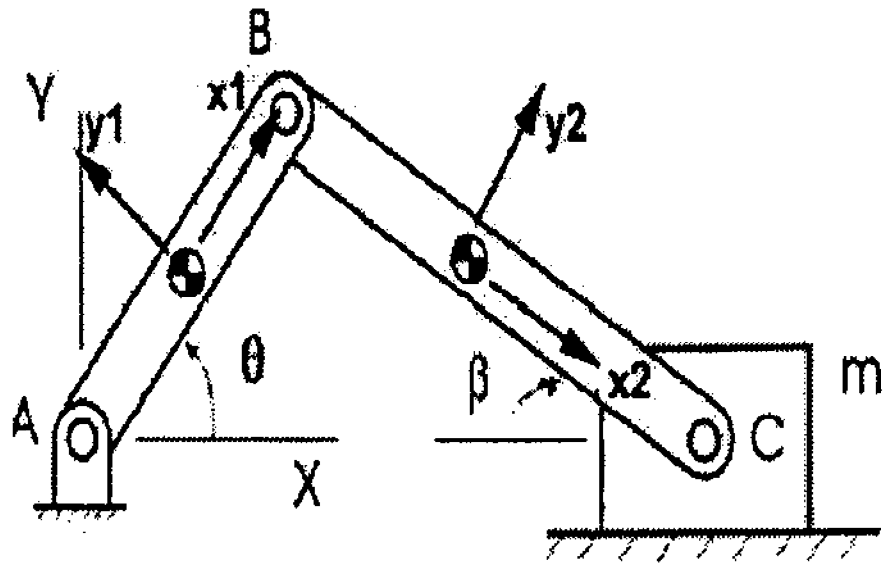


Figure 4.3: Two dimensional slider crank.

The two dimensional slider crank in Figure 4.3 is made using components from the Multibody mechanical library. This system has a revolute joint, A, which is connected to a planar link. This planar link is attached to a connecting rod using a second revolute joint, B. The connecting rod is then connected to a sliding mass by a third revolute joint, C, and the sliding mass is attached to the ground by a prismatic joint. In particular, this system is used to convert rotational motion of the crank to translational motion of the sliding mass or vice versa. For this system gravity is considered to be the only external force, acting along the negative Y axis (the y axis for the inertial frame) [4].

This model in MapleSim consists of two subsystems, one for the crank and one for the connecting rod. See Figure 4.4 for these subsystems which each contains two `Rigid Body Frames` and one `Rigid Body`. A `Rigid Body Frame` is a frame with a fixed displacement and orientation relative to a rigid body center of mass (CoM) frame with associated mass and inertia matrix which is this `Rigid Body` here.

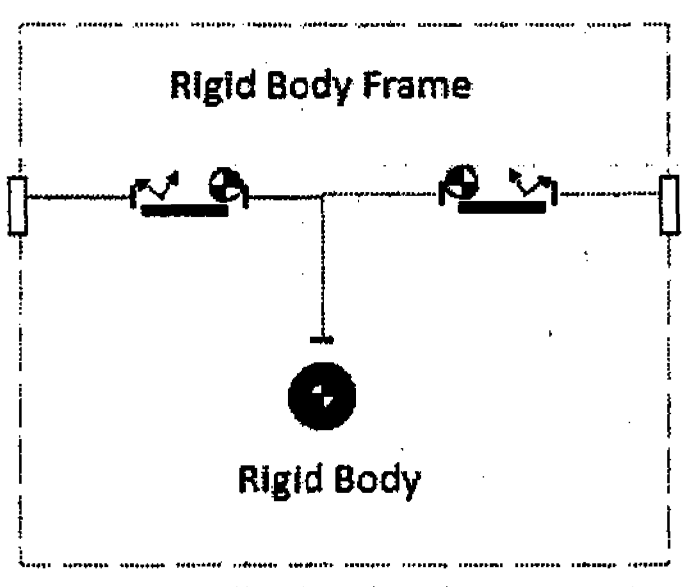
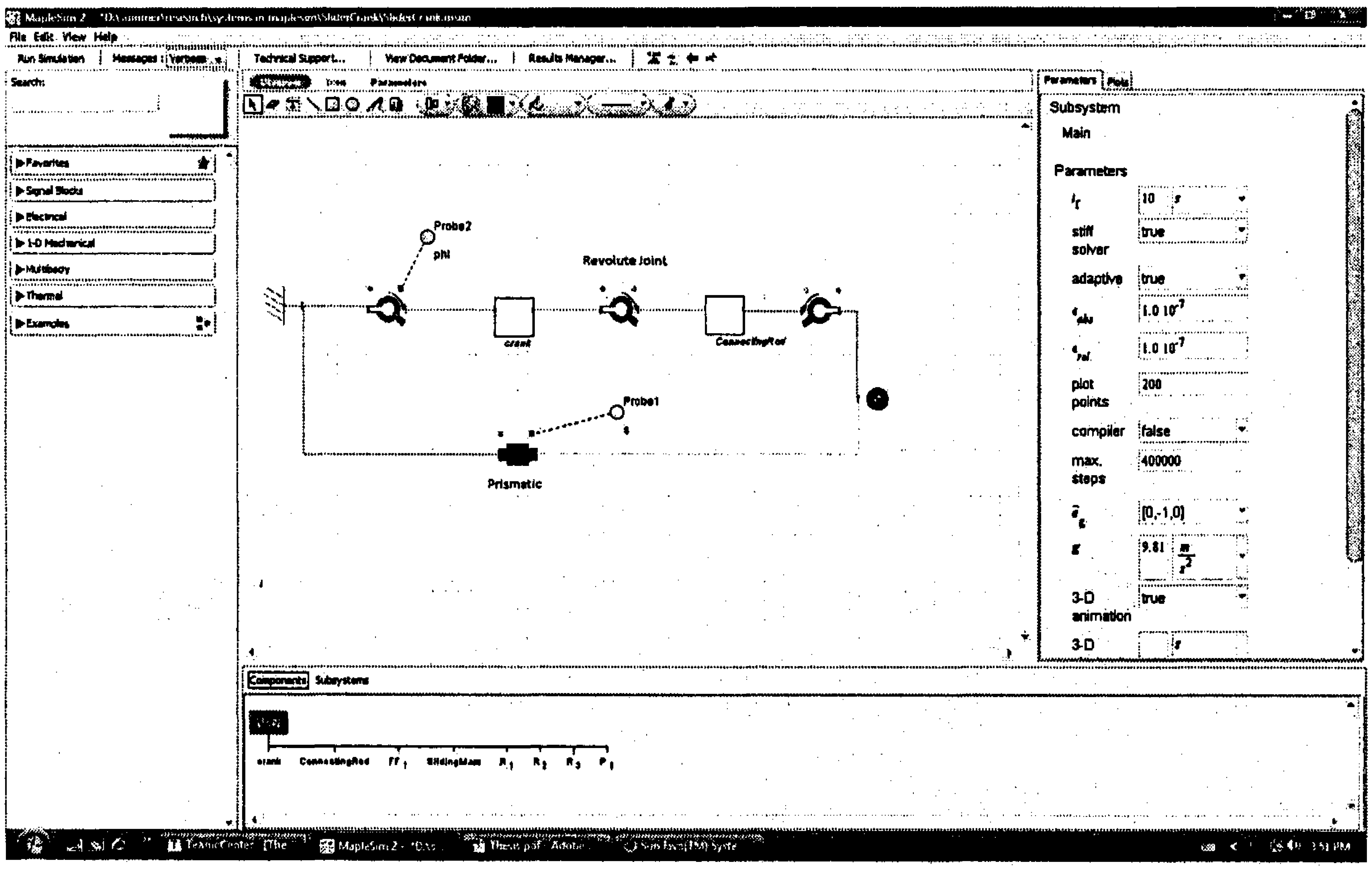


Figure 4.4: Crank or connecting rod subsystem in slider crank in MapleSim.

In MapleSim we choose the length of the crank and connecting rod to be 1 and 2 meters, respectively. We discuss later a singular case where the bars have equal lengths. The model of the slider crank in Figure 4.5 consists of the two mentioned subsystems, three `Revolute` joints, one `Rigid Body` and one `Prismatic` (joint allowing one translational degree of freedom along a given axis). We need MapleSim to measure displacement from A to C and angle θ in Figure 4.3. Therefore we add two probes to our system, to report the length and angle θ. We give an initial condition of $\pi/4$ for the angle θ for the first `Revolute joint`, which also has an attached probe. This initial condition will be transferred to the related Maple worksheet via the Modelica file. The model before simulation is shown in Figure 4.5, and after simulation is given in Figure 4.6.



Subsystem
Main

Parameters

t_f	10 s
stiff solver	true
adaptive	true
ϵ_{abs}	$1.0\ 10^{-7}$
ϵ_{rel}	$1.0\ 10^{-7}$
plot points	200
compiler	false
max. steps	400000
$\vec{e}_g$	[0,-1,0]
g	9.81 $\frac{m}{s^2}$
3-D animation	true
3-D	s

Figure 4.5: Two dimensional slider crank in MapleSim environment.

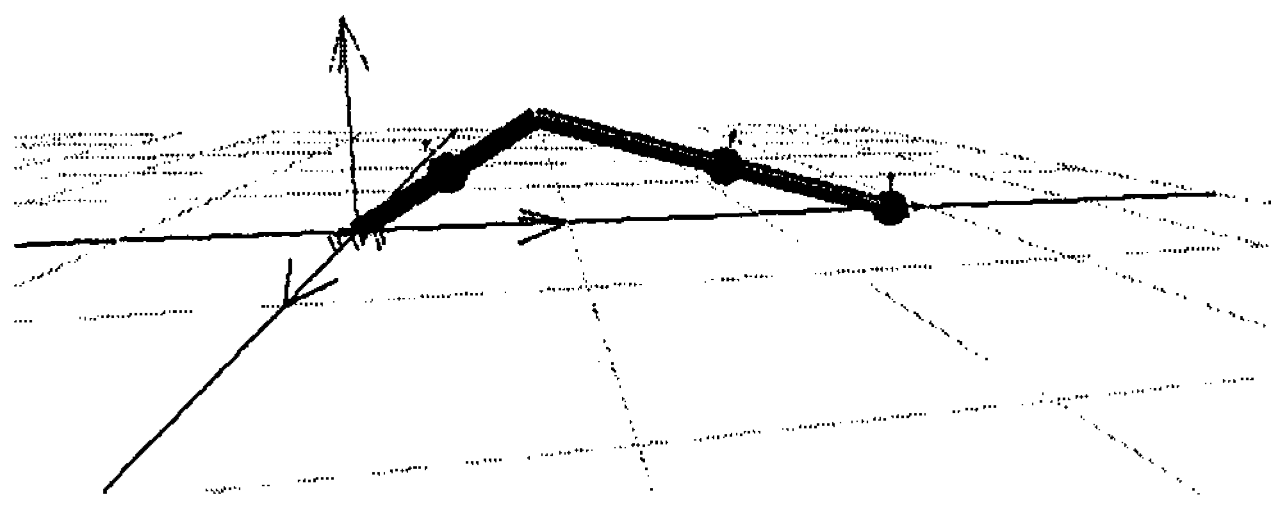


Figure 4.6: Animation of two dimensional slider crank after simulation.

When we derive its DAE equations, we found that there are 23 equations and 23 variables. The Maple form of the unsimplified system is:

$$
\begin{aligned}
&X5(t) = 0, X10(t) = 0, X1(t) = X4(t), X4(t) = X6(t), X2(t) = X7(t),\\
&X7(t) = X9(t), -X3(t) - X5(t) = 0, -X8(t) - X10(t) = 0, X15(t) = X4(t),\\
&X18(t) = X7(t), X14(t) + X3(t) = 0, X14(t) = X12(t), X19(t) = X13(t),\\
&X19(t) + X8(t) = 0, X18(t) = X16(t),\\
&\frac{d}{dt}X16(t) = X21(t), \frac{d}{dt}X17(t) = X23(t), \frac{d}{dt}X21(t) = X20(t), \frac{d}{dt}X23(t) = X22(t),\\
&2\sin(X17(t)) + \sin(X16(t)) = 0,\\
&X15(t) = 2\cos(X17(t)) + cos(X16(t)),\\
&3.25X20(t) + 3X22(t)\cos(X16(t))\cos(X17(t)) + 3X22(t)\sin(X16(t))\sin(X17(t)) -\\
&\cos(X16(t))X11(t) + 24.525\cos(X16(t)) - 3\cos(X16(t))X23(t)^2 sin(X17(t)) +\\
&3\sin(X16(t))X23(t)^2\cos(X17(t)) + \sin(X16(t))X12(t) - X13(t) = 0,\\
&3X20(t)\cos(X16(t))\cos(X17(t)) + 3X20(t)\sin(X16(t))\sin(X17(t)) + 6X22(t) -\\
&2cos(X17(t))X11(t) + 29.43\cos(X17(t)) - 3\cos(X17(t))X21(t)^2\sin(X16(t)) +\\
&3\sin(X17(t))X21(t)^2\cos(X16(t)) + 2\sin(X17(t))X12(t) = 0.
\end{aligned}
\tag{4.2}
$$

DEtools[rifsimp]'s result on the first five rows of (4.2) is:

$$
\begin{aligned}
&\frac{d}{dt}X17(t) = X23(t), \frac{d}{dt}X21(t) = X20(t),\\
&\frac{d}{dt}X23(t) = X22(t), \frac{d}{dt}X9(t) = X21(t),\\
&X1(t) = X6(t), X10(t) = 0, X12(t) = 0, X13(t) = 0, X14(t) = 0,\\
&X15(t) = X6(t), X16(t) = X9(t), X18(t) = X9(t), X19(t) = 0,\\
&X2(t) = X9(t), X3(t) = 0, X4(t) = X6(t), X5(t) = 0, X7(t) = X9(t), X8(t) = 0.
\end{aligned}
\tag{4.3}
$$

If we use the three last rows from (4.3) we can use for example $X6(t)$ instead of $X1(t)$, and remove for example $X10(t)$ because it is zero, therefore out of 23 variables we are left with eight variables below:

$$X11(t), X17(t), X20(t), X21(t), X22(t), X23(t), X6(t), X9(t). \tag{4.4}$$

Moreover, we can use the first two rows of (4.3) to remove $X20(t)$, $X21(t)$, $X22(t)$, and

$X23(t)$ from the variables (4.4) as well. Finally we are left with just four variables, $X6(t)$, $X9(t)$, $X11(t)$, and $X17(t)$. Substitution of the rifsimp's result in the remaining part of the DAE system which is eight rows of (4.2), gives a square system of four variables and four equations:

$$\begin{aligned}
&2\sin(X17(t)) + \sin(X9(t)) = 0, X6(t) = 2\cos(X17(t)) + \cos(X9(t)),\\
&3.25(\frac{d^2}{dt^2}X9(t)) + 3(\frac{d^2}{dt^2}X17(t))\cos(X9(t))\cos(X17(t)) +\\
&3(\frac{d^2}{dt^2}X17(t))\sin(X9(t))\sin(X17(t)) - \cos(X9(t))X11(t) +\\
&24.52\cos(X9(t)) - 3\cos(X9(t))(\frac{d}{dt}X17(t))^2\sin(X17(t)) +\\
&3\sin(X9(t))(\frac{d}{dt}X17(t))^2\cos(X17(t)) = 0,\\
&3(\frac{d^2}{dt^2}X9(t))\cos(X9(t))\cos(X17(t)) + 3(\frac{d^2}{dt^2}X9(t))\sin(X9(t))\sin(X17(t)) +\\
&6(\frac{d^2}{dt^2}X17(t)) - 2\cos(X17(t))X11(t) + 29.43\cos(X17(t)) -\\
&3\cos(X17(t))(\frac{d}{dt}X9(t))^2\sin(X9(t)) + 3\sin(X17(t))(\frac{d}{dt}X9(t))^2\cos(X9(t)) = 0.
\end{aligned} \tag{4.5}$$

Here $X9(t) = \theta$, and $X17(t) = \beta$, and $X6(t)$ is displacement from A to C in Figure 4.3. Also $X11(t)$ is the Lagrange multiplier for the system.

4.2.2 Fast prolongation

After we obtain a simplified system, we use fast prolongation to determine all missing constraints. Using our package, we can quickly calculate the c_i and d_j. Recall c_i is the number of times we have to differentiate (prolong) the i-th equation in the simplified DAE system and d_j is the highest derivative order of variable u^j after the prolongation.

The c_i and d_j below for DAE system of the slider crank, (4.5) with four variables, $X6(t)$, $X9(t)$, $X11(t)$, and $X17(t)$ are obtained in 0.016 seconds:

$$\begin{aligned}
&c_1 = 2,\ c_2 = 0,\ c_3 = 0,\ c_4 = 0,\\
&d_1 = 0,\ d_2 = 2,\ d_3 = 0,\ d_4 = 2.
\end{aligned} \tag{4.6}$$

Therefore, according to the c_i values we have to differentiate first equation in (4.5) twice and there is no differentiation needed for the other three equations. Moreover, because of the d_j values the differentiation order of $X6(t)$ and $X11(t)$ after prolongation turns out to be zero. However, we expect the first order derivative and second order derivative of $X9(t)$, and $X17(t)$ to be in the system.

4.2.3 Solution of the DAE system

At this stage, we set the initial time, t_0 to find initial conditions in order to solve the simplified DAE system. Unlike the crane example in [1], our MapleSim models are invariant under time translation and we can choose initial time $t = 0$. Wu and Reid in [1] used the `stats` command in Maple to set random initial times because t appeared explicitly in non-derivative form in their example.

If $c_i = 0$ for all i, we use initial conditions from the Modelica file, or random initial conditions. I will give an example of this later in Chapter 5.

If $\exists i, c_i \neq 0$, we calculate the block structure. For the slider crank here $c_1 \neq 0$, therefore we compute the block structure. Recall $c_1 = 2$ means we have to prolong (differentiate) the first equation in the simplified DAE system twice. Thus the block structure for the slider crank contains three blocks, B_0, which is called the bottom block, and blocks B_1, B_2 which are given below:

- $B_0 = [2\sin(X17(t)) + \sin(X9(t)) = 0]$.
- $B_1 = [2\cos(X17(t))\frac{d}{dt}X17(t) + \cos(X9(t))\frac{d}{dt}X9(t) = 0]$.
- $B_2 = [-2\sin(X17(t))(\frac{d}{dt}X17(t))^2 + 2\cos(X17(t))\frac{d^2}{dt^2}X17(t) - \sin(X9(t))(\frac{d}{dt}X9(t))^2 + \cos(X9(t))\frac{d^2}{dt^2}X9(t) = 0,\ X6(t) = 2\cos(X17(t)) + \cos(X9(t)), \ldots]$

In order to solve the ODE system for this problem, we have to solve B_0, B_1, and B_2 to find initial conditions. The total Bezout degree of the constraints is high. The Bezout degree of a polynomial system of equations is the product of the degrees of each of the equations. It is an important measure of the complexity of solving the system. However, even though our constraint system has high Bezout degree it has a block triangular structure which enables us to solve efficiently it by bottom up substitution.

In bottom up substitution, we first numerically solve the bottom block (B_0), then substitute the solution into B_1 and solve it. Again, we substitute solution of B_1 into B_2 and solve B_2 and repeat this until we reach to the last B_i.

We use the Bertini package as our numerical solver. However Bertini requires polynomial systems rather than algebraic equations. To obtain polynomial equations for the slider crank, we replace $\sin(X9(t))$, $\cos(X9(t))$, $\sin(X17(t))$, and $\cos(X17(t))$ by $n1(t)$, $n2(t)$, $n3(t)$, and $n4(t)$ respectively. The bottom block becomes $2*n3(t)+n1(t) = 0$ subject to the relevant trigonometric identities $n1(t)^2 + n2(t)^2 = 1$, and $n3(t)^2 + n4(t)^2 = 1$.

To send these blocks to Bertini we need one more change. All variables are replaced by jet variables. Therefore the three blocks become:

- $B_0 = [2*n3+n1 = 0, n1^2+n2^2-1 = 0, n3^2+n4^2-1 = 0]$.
- $B_1 = [2*n4*X17t+n2*X9t = 0]$.
- $B_2 = [-2*n3*X17t^2+2*n4*X17tt-n1*X9t^2+n2*X9tt = 0, X6 = 2*n4+n2, 3.25*X9tt+3*X17tt*n2*n4+3*X17tt*n1*n3-n2*X11+24.525*n2-3*n2*X17t^2*n3+3*n1*X17t^2*n4 = 0, 3*X9tt*n2*n4+3*X9tt*n1*n3+6*X17tt-2*n4*X11+29.43*n4-3*n4*X9t^2*n1+3*n3*X9t^2*n2 = 0]$.

Using the initial condition $X9(t) = \pi/4$ from the Modelica file and solving the system of polynomials from B_0 in Bertini gives values for $n1$, $n2$, $n3$, and $n4$ which we substitute in B_1. Then we solve the single equation in B_1 and get $X9t$, and $X17t$. Substitution of all these in B_2 and solving the resulting polynomial system, we obtain initial conditions for $X9tt$, $X17tt$, $X6$, and $X11$.

After we obtain initial conditions for all variables, Maple's dsolve/numeric command is used to solve the ODE system which is the last block B_i. For almost all MapleSim models, we use an implicit numerical method because the system of DAE from these models are stiff and explicit methods are very expensive.

Finally, as a confirmation we can plot the solution and compare it to the MapleSim result. For the slider crank example, we can plot $X9(t)$, and $X6(t)$ for desired time to compare with two probes that are shown in MapleSim part, θ and distance from A to C in Figure 4.3. In Chapter 5, and 6 we will give more examples for this method.

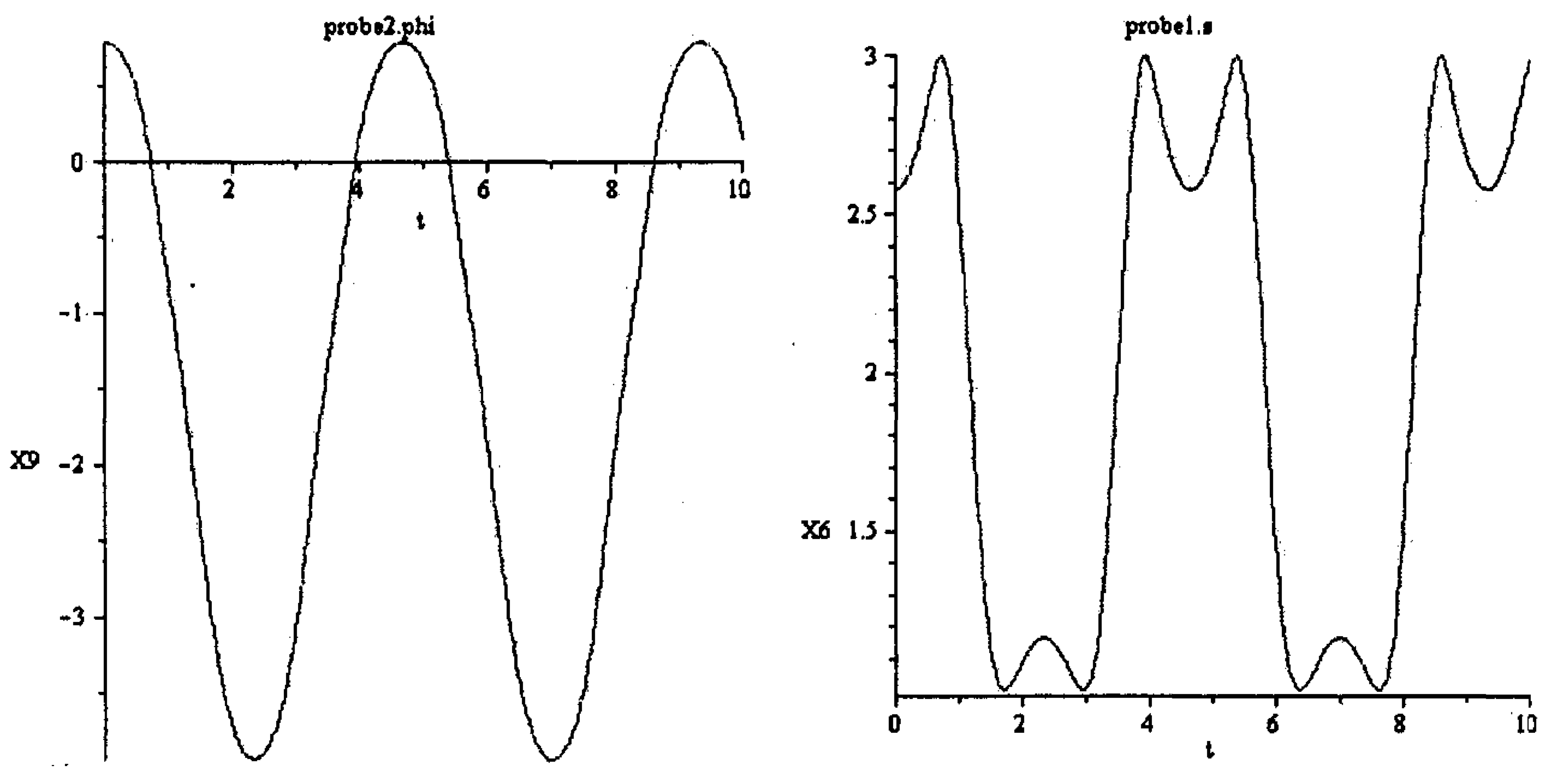


Figure 4.7: Maple result for angle and displacement for slider crank.

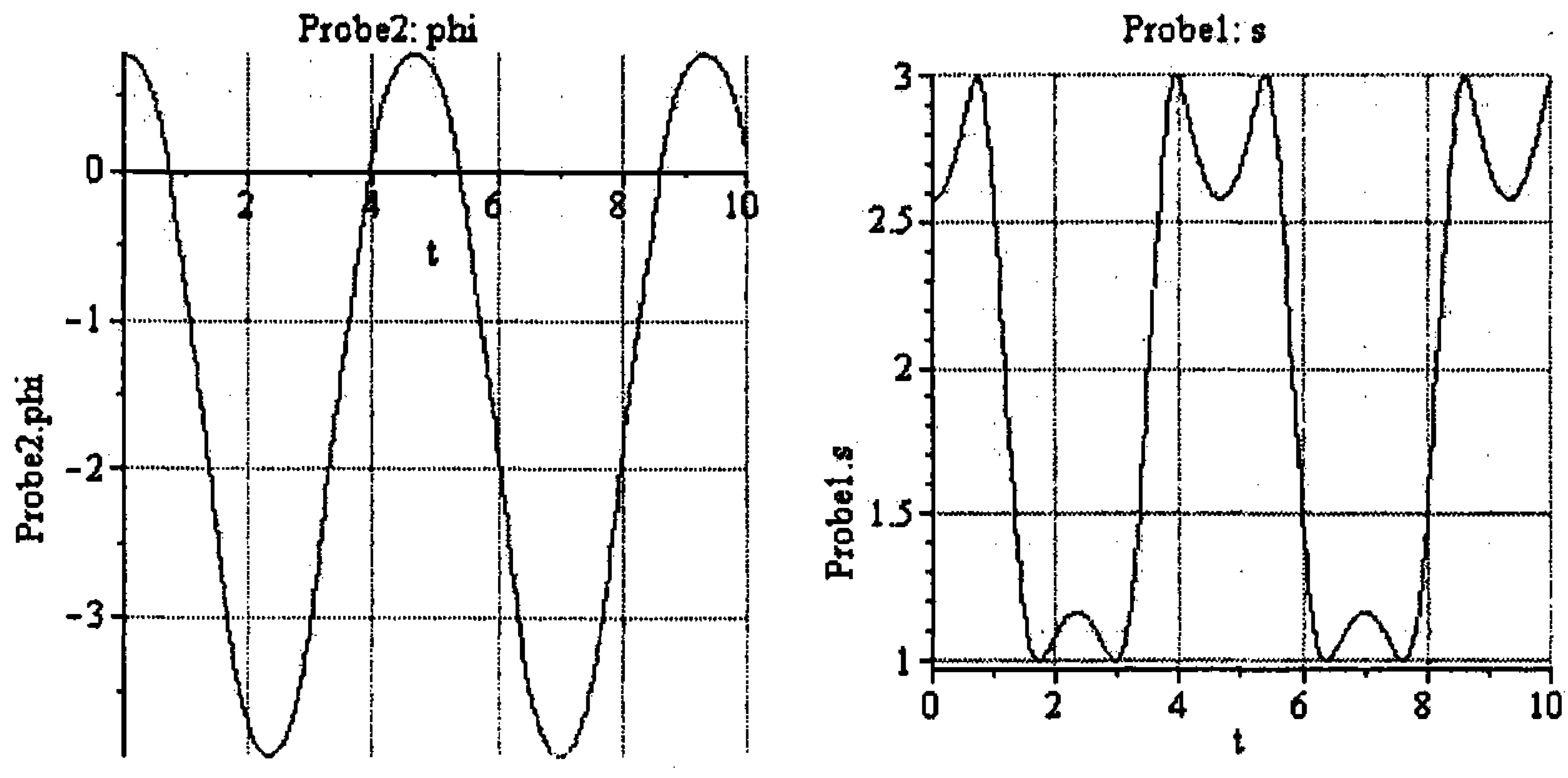


Figure 4.8: MapleSim result for angle and displacement for slider crank.

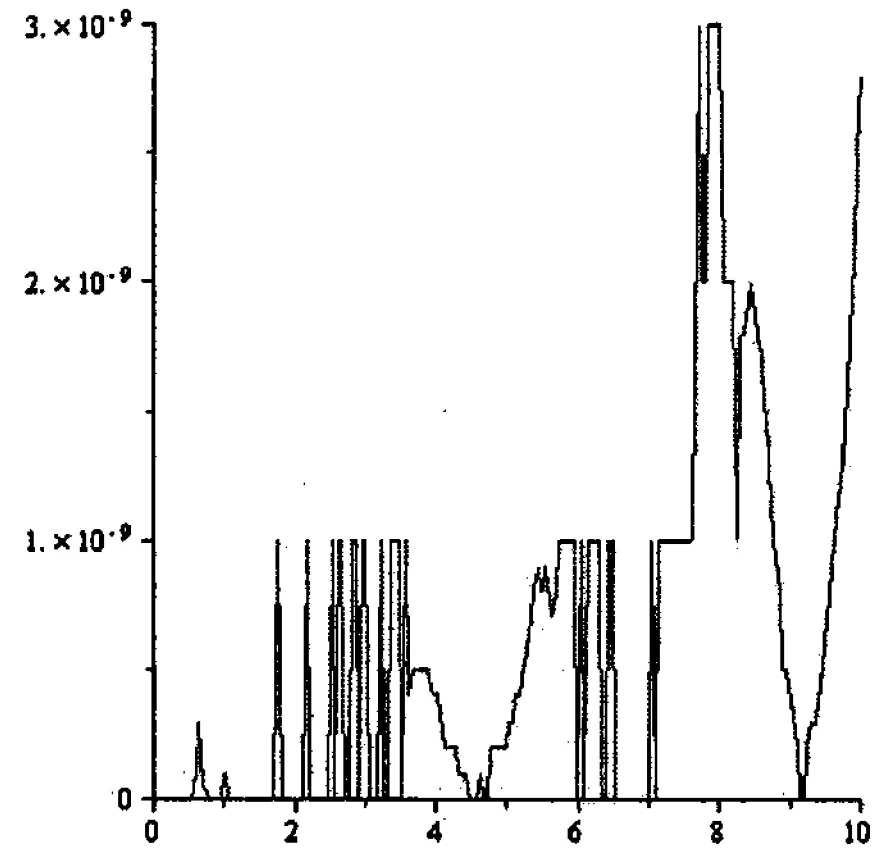


Figure 4.9: Absolute error for Probe2.phi.

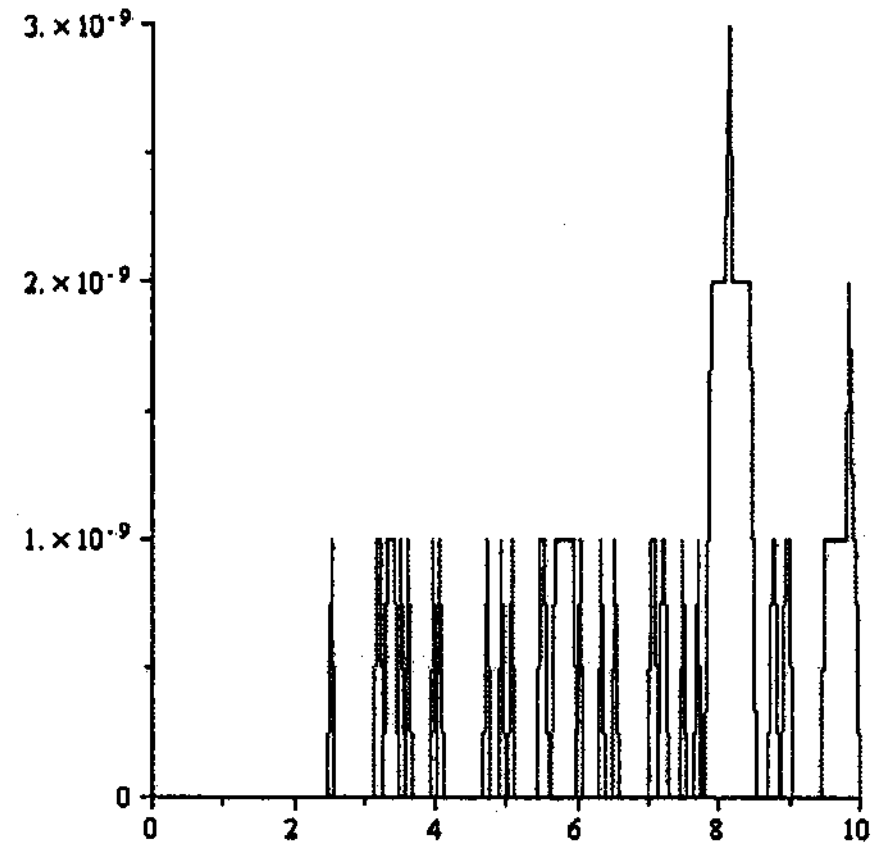


Figure 4.10: Absolute error for Probe1.s.

4.3 Simplification of various type of DAE systems generated by MapleSim

Not all of the DAE systems which we worked on are like (4.1). For example the DAE system of RLC-Circuit model is:

$$
\begin{aligned}
&X8(t) = 0, X21(t) = X27(t), X1(t) - X4(t) = 0, \ldots \\
&X1(t) - C_1 \frac{d}{dt} X6(t) = 0, X14 - C_2 \frac{d}{dt} X9(t) = 0, \ldots \\
&X21(t) = C_5 + \text{msim/PIECEWISE}(X28(t) < C_6, 0, \mathit{true}, C_4)
\end{aligned}
\tag{4.7}
$$

In (4.7) X28(t) is "_msim_time" which is time or t in the system. Also msim/PIECEWISE is the symbol for piecewise function in the Modelica file:

$$
X21(t) = C_5 + \begin{cases} 0 & \text{for} \quad t < C_6 \\ C_4 & \qquad \text{otherwise} \end{cases}
\tag{4.8}
$$

An example of a different class of MapleSim DAE systems is the Heating Transfer system:

$$
\begin{aligned}
&X5(t) = 0, X7(t) = 0, X1(t) - X2(t) = 0, \ldots \\
&X19(t) - C_1 \frac{d}{dt} X1(t) = 0, X3 - C_2 \frac{d}{dt} X20(t) = 0, \ldots \\
&X19(t) - X5(t) + X25(t) + X32(t) = 0, \ldots
\end{aligned}
\tag{4.9}
$$

Another vary interesting DAE system is that for a 3D pendulum with drag in MapleSim:

$$
\begin{aligned}
&X69(t) = 0, X70(t) = 0, X44(t) - X17(t) = 0, \ldots \\
&X95(t) - C_1 \frac{d}{dt} X96(t) = 0, X94 - C_2 \frac{d}{dt} X50(t) = 0, \ldots \\
&X78(t) = \sin(X50(t)), X59(t) = \cos(X50(t)) \cos(X97(t)), \ldots \\
&X47(t) = C_3 + \text{msim/PIECEWISE}(X1(t) < C_4, 0, \mathit{true}, C_5) \ldots \\
&-2\cos(X96(t)) \sin(X50(t)) \sin(X97(t)) X49(t) + X94(t) X95(t) \cos(X50(t)) \ldots
\end{aligned}
\tag{4.10}
$$

For this system we can use the first three rows of (4.10) to do simplification. However we are unable to use DEtools[rifsimp] for equations similar to the third row of (4.10) since sin and cos functions appear in the system. We first obtain DEtools[rifsimp]'s simplification of first two rows of (4.10) and substitute it into the rest of (4.10). We then use equations from third row of (4.10) to again substitute, for example $\sin(X50(t))$ for $X78(t)$ and $\cos(X50(t))\cos(X97(t))$ for $X59(t)$. This reduces the number of variables and equations further.

There are some other DAE systems from MapleSim models which combine feature of (4.1) and (4.7) or (4.1) and (4.9) such as a slider crank and a DC-motor attached together and a centrifuge. Similarly, we can use DEtools[rifsimp] on redundant equations and substitute the result in the rest of the DAE system to simplify it. Simplification results for different MapleSim models by this method is summarized in this table below:

MapleSim Model	Before simplification #Eqns×#Vars	After simplification #Eqns×#Vars
2D Slider Crank	23×23	4×4
3D Slider Crank	24×24	7×7
Rotating Pendulum	26×26	2×2
Gimbal	27×27	3×3
RLC-circuit	32×32	1×1
Furuta Pendulum	34×34	2×2
Triplets	39×39	3×3
Heating transfer system	41×41	1×1
DC-motor	57×57	1×1
Non-linear Spring Damper	57×57	2×2
Five-Pendulums	65×65	5×5
One-loop-Pendulum	69×69	8×8
Slider crank+DC-motor	72×72	5×5
Centrifuge	100×94	3×3
Two-loops-Pendulum	102×102	14×14
3-D Pendulum	117×96	3×3
Ice Tank	186×186	5×5

Table 4.1: Simplification result for various MapleSim models.

4.4 Fast Prolongation on DAE systems generated by MapleSim

After we obtain a simplified system, the fast prolongation method is used to uncover all hidden constraints. The fast prolongation is very fast-not more than 0.05 seconds was taken on all models here. This is done in a computer with 2.66 GHz Intel Core processor and 6.00 GB memory (RAM). Fast prolongation timing for different MapleSim models that we did is given in this table below:

MapleSim Model	Time in seconds
2D Slider Crank	0.016
3D Slider Crank	0.016
Rotating pendulum	0.015
Gimbal	0.015
RLC-circuit	0.016
Furuta	0.031
Triplets	0.001
Heating Transfer System	0.016
DC-motor	0.015
Non-linear Spring Damper	0.001
Five-pendulum	0.031
One-loop-pendulum	0.046
Slider crank+DC motor	0.001
Centrifuge	0.016
Two-loops-pendulum	0.062
3-D Pendulum	0.006
Ice Tank	0.016

Table 4.2: Fast prolongation timing for various MapleSim models.

Chapter 5

Some MapleSim Models

5.1 Introduction

We applied the algorithm of Chapter 4 to many MapleSim models. Some of them are pre-prepared examples such as, centrifuge, heating transfer system, ice-tank, etc. They are accessible from MapleSim's examples menu. However, we built others ourselves using sources such as [3, 4].

We will discuss user created examples using a so-called 'custom components' which are special feature of MapleSim. Custom components help us to make new components that are not already in MapleSim by first writing the equations for that component in a Maple worksheet.

We have substantial experience while working with the MapleSim models and we can not talk about all of it here. In this and the next chapter, we try to summarize some special and more significant cases that we faced. The first model we consider is a model with custom components. This model does not have any missing constraints. The second model is a multi-domain DC-motor attached to a slider crank. In this model we show how to deal with a piecewise functions while using the Bertini software which requires just polynomial equations. We made these models using [4].

5.2 Models with custom components

In addition to over 300 available components, we can extend the MapleSim library by creating custom components that are based on mathematical models that we define. Custom physical components can be created by writing down the equations in proper mathematical notation in a Maple template. A custom component can contain a particular subsystem and provide specialized functionality. By using the Custom Component Template, which is a Maple worksheet included in the MapleSim document folder, we perform the following tasks in Maple to create a custom component:

- Define the component equations and properties that determine the behavior of the component (for example, parameters, input/output ports, and variables).
- Test and analyze our mathematical model.
- Define and add input and output to the component.
- Generate the component and make it available in MapleSim.

5.2.1 A non-linear damper with a linear spring

To make this model [4], we used the custom component template to create a non-linear spring damper component defined by differential equations. The equation defined in this part are based on the `Translational Spring Damper` component in MapleSim. In this case, the stiffness and damping coefficients are replaced with functions that are added as inputs to the component. The spring-damper system is shown in Figure 5.1.

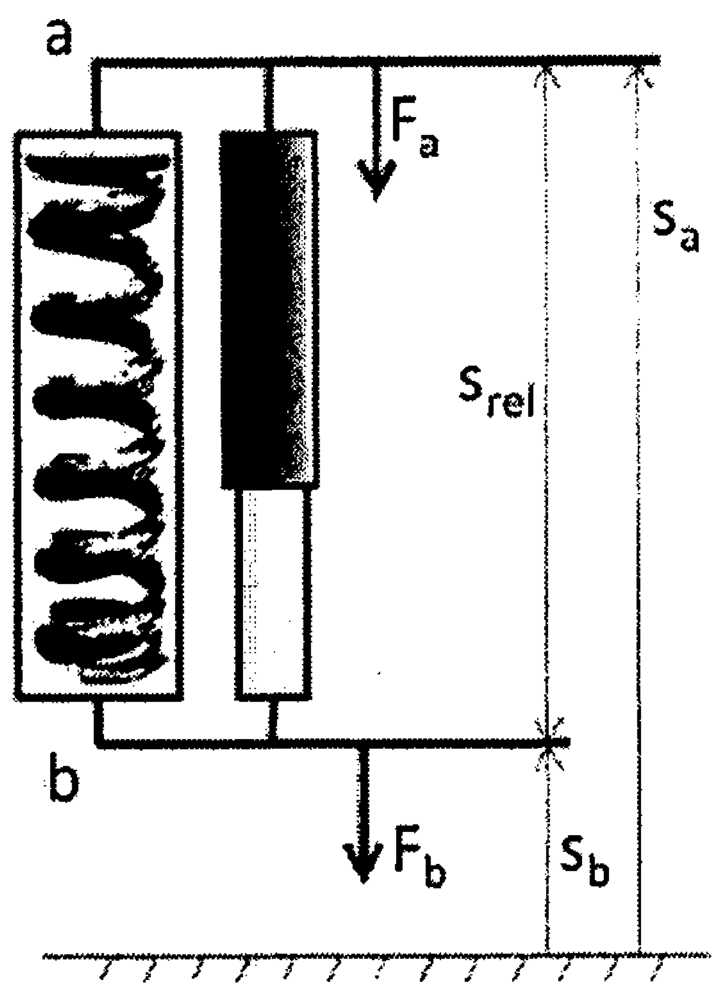


Figure 5.1: The spring-damper system.

As previously mentioned, MapleSim components are based on equations. To define the equations related to this component the end points, a and b, can be defined as the 'ports' for the component. The equations are derived relative to these ports. The general equation of the motion is:

$$d * \frac{d}{dt} s_{rel}(t) + c * s_{rel}(t) = F(t). \tag{5.1}$$

Here d is the damping coefficient, and c is the stiffness of this spring. The quantity s_{rel} is the relative displacement between two ports s_a and s_b which is defined by

$$s_{rel} = s_a - s_b. \tag{5.2}$$

By a force analysis of the system, we have:

$$F(t) = F_b(t). \tag{5.3}$$

$$F_a(t) + F_b(t) = 0. \tag{5.4}$$

All of the above equations are necessary to define the behavior of this component. We open a custom component template, through a document folder in MapleSim workspace and write these equations above in the provided document space. The quantities $F_a(t)$, $F_b(t)$, $c(t)$, $d(t)$ and s_a, s_b are input and output variables, respectively. When we define everything in this template, the `Generate MapleSim Custom Component` button at the end of the page adds this new component, $NLMSD_1$ in Figure 5.2, to the MapleSim workspace. Also, MapleSim allows us to attach more files to our model. For instance, we can provide the relative displacement of the damper (0, 0.05, 0.1, 0.2, 0.25, 0.3) and values of the damping coefficient (750, 500, 250, 75, 250, 650) in an external Excel file.

We now make a subsystem called `Nonlinear Damper`. In order to do that, we need a procedure `Constant` to generate a real constant signal. This Constant produces stiffness of the spring which is 1000 here. Also, we add a procedure `Gain` which outputs the product of a gain value which is chosen 1 here with the input signal, and a `1D lookup table`, useful for transferring data from the Excel file. Finally a `Position Sensor` is added that measures the absolute position of a translational port. Figure 5.2 displays this subsystem.

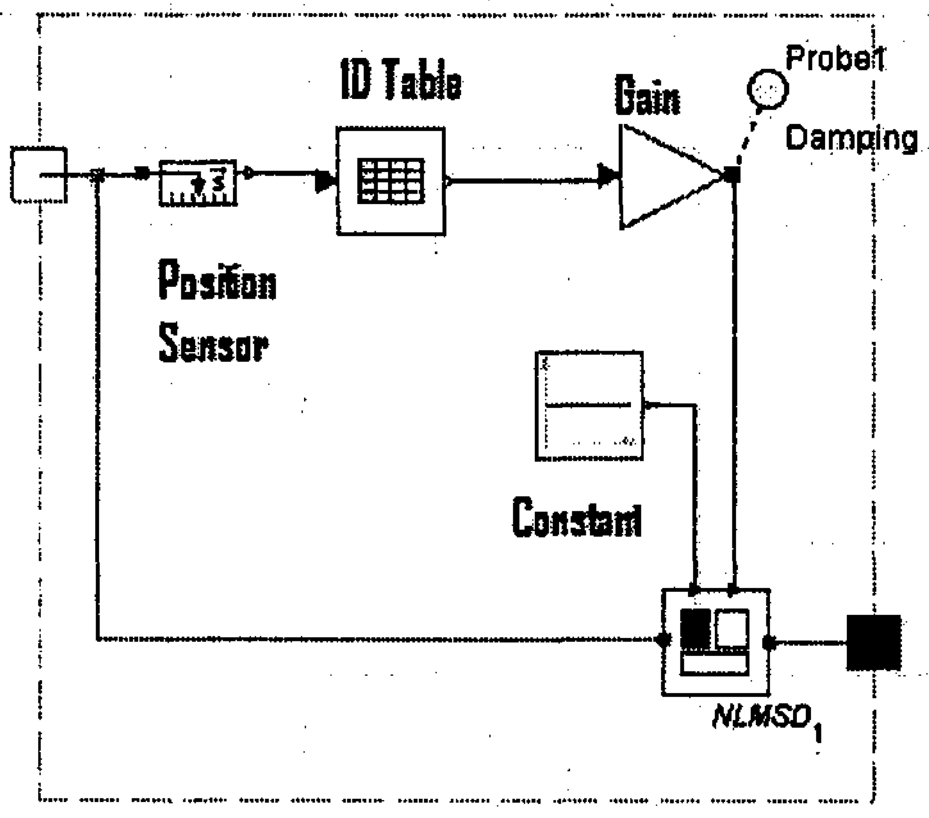


Figure 5.2: The non-linear damper subsystem in MapleSim.

To finalize the main system, we need a `Sliding Mass`, which is a sliding mass with inertia, and a `Force`, for the external force acting on a drive train element as input signal, and `Step` which generates a real step signal with the height 100 here. Mass of `Sliding Mass` is chosen, for instance, to be 100 Kg. The final system of this model in MapleSim is given below in Figure 5.3.

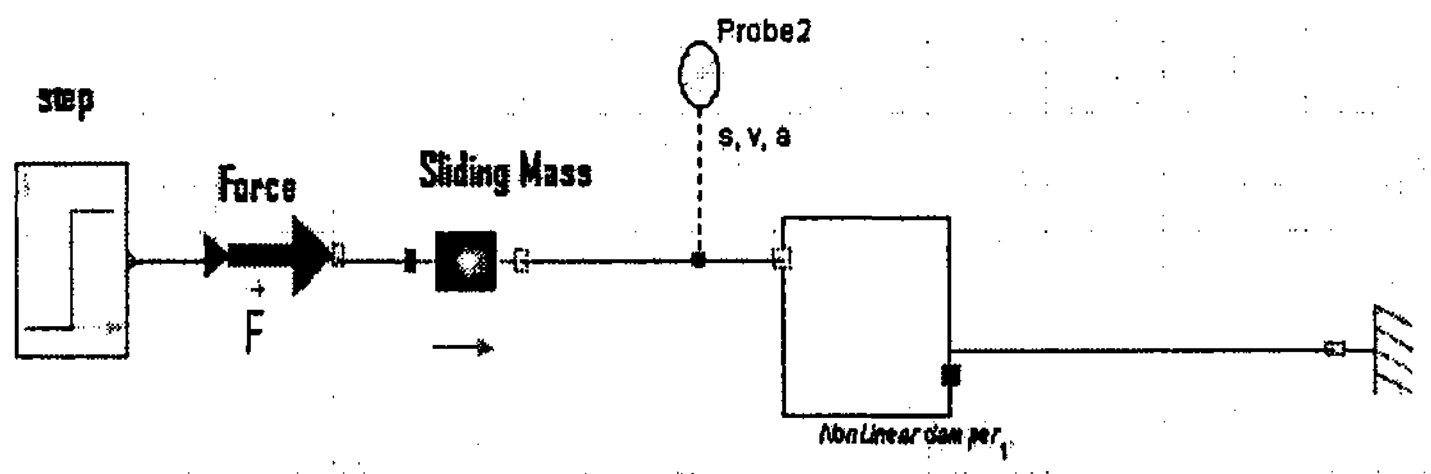


Figure 5.3: The non-linear damper with linear spring in MapleSim.

We ask MapleSim to measure the damping, displacement, speed, and acceleration by two probes which we added to the subsystem and the main system, respectively. They are shown in Figures, 5.2 and 5.3 as `Probe1-Damping`, `Probe2-s`, `Probe2-v`, and `Probe2-a`. To implement our algorithm, a Modelica file is made in MapleSim and is derived in a Maple worksheet. All information about the system including system of DAE is transferred from Modelica format. The DAE system of this model is a system of 57 equations and 57 variables of form:

$$\begin{aligned}
&X13(t) = 0, X1(t) = X7(t), X2(t) - X3(t) = 0, \ldots\\
&X4(t) - C_1\frac{d}{dt}X6(t) = 0, X8 - C_2\frac{d}{dt}X10(t) = 0, \ldots\\
&X7(t) = \text{msim/PIECEWISE}(X28(t) < C_3, 0, true, C_4), \qquad (5.5)\\
&X34(t) = \text{msim/PIECEWISE}(X33(t) < 0.2, \text{msim/PIECEWISE}(\ldots), true, \ldots),
\end{aligned}$$

where

$$X28(t) = \_\text{msim\_time} = t$$

Using DEtools[rifsimp] on the first two lines of (5.5) and substitution of the result into the next two lines of (5.5) yields a simplified 2 × 2 system of equations:

$$100\frac{d^2}{dt^2}X9(t) + X41(t)\frac{d}{dt}X9(t) + 1000X9(t)) = \begin{cases} 0 & t < 0 \\ 100 & \text{otherwise} \end{cases} \tag{5.6}$$

In (5.6), $X41(t)$ which is the damping coefficient is generated with linear interpolation based on the position of the mass.

$$X41(t) = \begin{cases} g(X9(t)) & X9(t) < .2 \\ h(X9(t)) & \text{otherwise} \end{cases}$$

and

$$g(X9(t)) = \begin{cases} C_1 - C_2X9(t) - C_3X9(t)^3 & \text{if} \quad X9(t) < 0.05 \\ K(X9(t)) & \text{otherwise} \end{cases}$$

$$K(X9(t)) = \begin{cases} C_4 - C_5X9(t) - C_6(X9(t) - 0.05)^2 + C_7(X9(t) - 0.05)^3 & \text{if} \quad X9(t) < .1 \\ C_8 - C_2X9(t) + C_9(X9(t) - .1)^2 + C_{10}(X9(t) - .1)^3 & \text{otherwise} \end{cases}$$

$$h(X9(t)) = \begin{cases} C_{11} + C_{12}X9(t) + C_{13}(X9(t) - .2)^2 + C_{14}(X9(t) - .2)^3 & \text{if} \quad X9(t) < .25 \\ C_{15} + C_{16}X9(t) + C_{17}(X9(t) - .25)^2 - C_{18}(X9(t) - .25)^3 & \text{otherwise} \end{cases}$$

Also $X9(t)$ =`Probe2-s`, $X41(t)$ =`Probe1-Damping`. Moreover, another outcome of DEtools[rifsimp] is

$$\frac{d}{dt}((\texttt{Probe2-s})) = \texttt{Probe2-v}, \tag{5.7}$$

$$\frac{d}{dt}(\texttt{Probe2-v}) = \texttt{Probe2-a}. \tag{5.8}$$

Next, we use fast prolongation to determine any hidden constraints. We find there are no missing constraints because:

$$c_1 = 0, c_2 = 0, d_1 = 2, d_2 = 0. \tag{5.9}$$

This result means that we do not need any prolongation of 2×2 system of equation, (5.6). Based on the algorithm in Chapter 4 and the fast prolongation result, we find initial conditions. If there are no initial conditions from the Modelica file, we generate random initial conditions. For this system there are no initial conditions coming from the Modelica file; therefore we are free to define our own:

$$X41(0) = 0, X9(0) = 0, D(X9)(0) = 0. \tag{5.10}$$

Finally, Maple's `dsolve/numeric` is applied to the 2×2 system of equations (5.6) with initial condition (5.10) to determine $X41(t)$ and $X9(t)$. We can plot the solutions and compare them with the MapleSim result. To avoid expensive calculation in Maple, we used standard finite difference approximation to calculate and plot the second derivative of $X9(t)$ which is `Probe2-a` [34]. We have:

$$D(f)(a) \simeq \frac{1}{h}\left(-\frac{1}{3}f(a-h) - \frac{1}{2}f(a) + f(a+h) - \frac{1}{6}f(a+2h)\right) \tag{5.11}$$

Therefore we can use the first derivative from `dsolve/numeric` result to approximate the second derivative of the desired variable:

$$D^2(f)(a) \simeq \frac{1}{h}\left(-\frac{1}{3}D(f)(a-h) - \frac{1}{2}D(f)(a) + D(f)(a+h) - \frac{1}{6}D(f)(a+2h)\right) \tag{5.12}$$

The MapleSim results and our Maple worksheet results are the same for damping, displacement, speed, and acceleration. In Figures 5.4 and 5.5 there is a comparison between displacement and speed. There is also a close agreement for damping and acceleration.

Maple worksheet result:

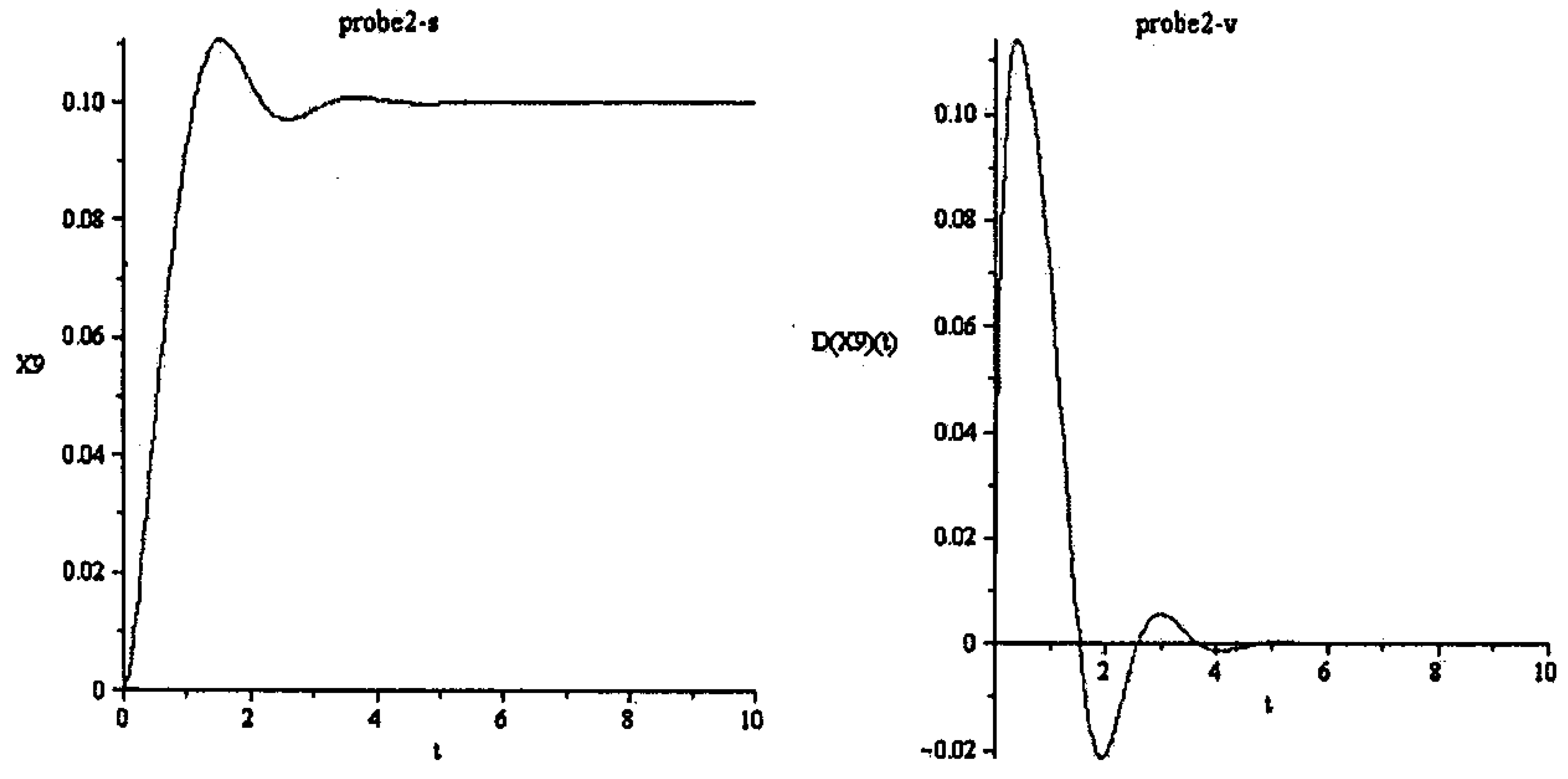


Figure 5.4: Maple result for displacement and speed for the non-linear damper with a linear spring.

MapleSim result:

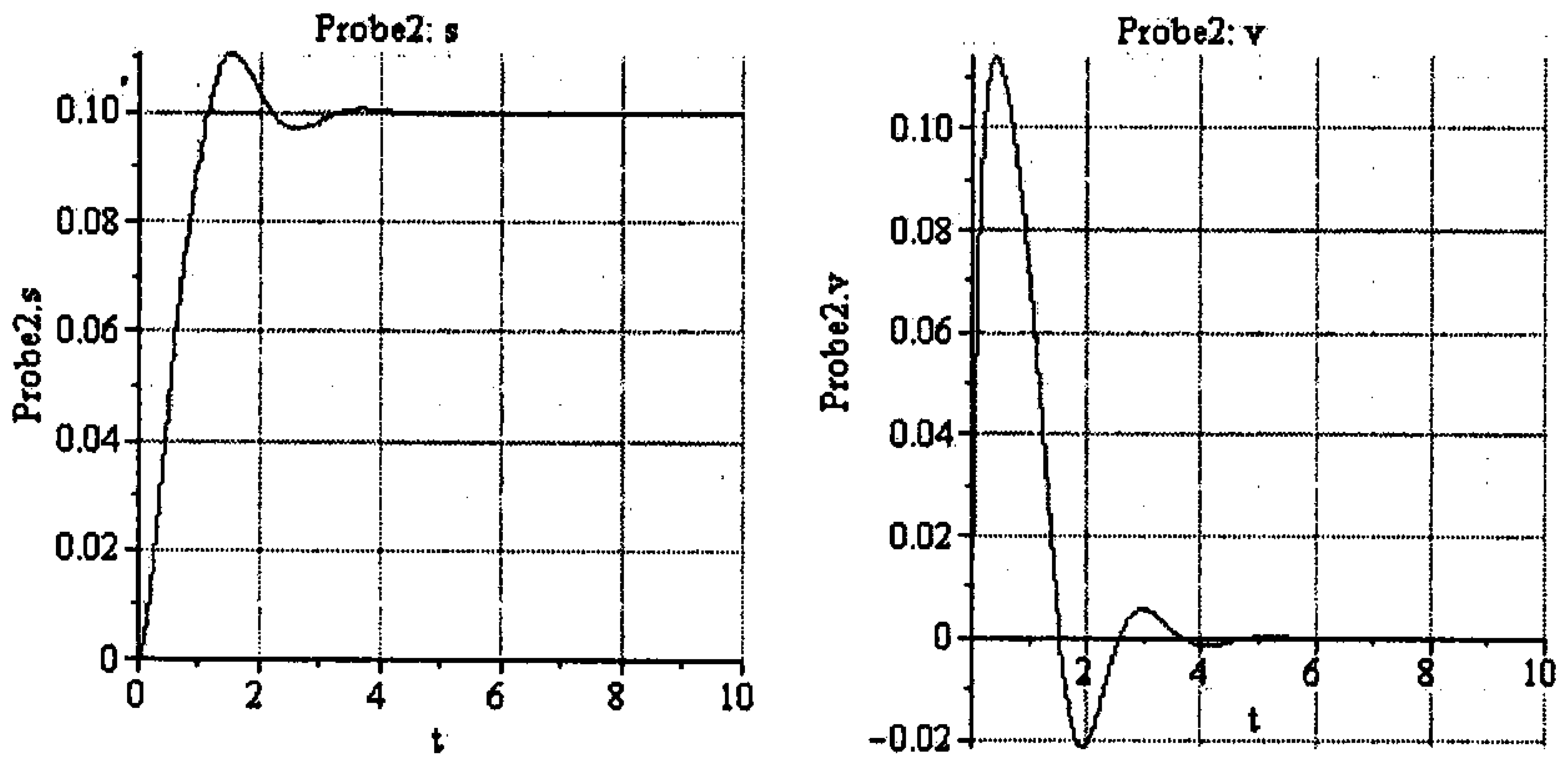


Figure 5.5: MapleSim result for displacement and speed for the non-linear damper with a linear spring.

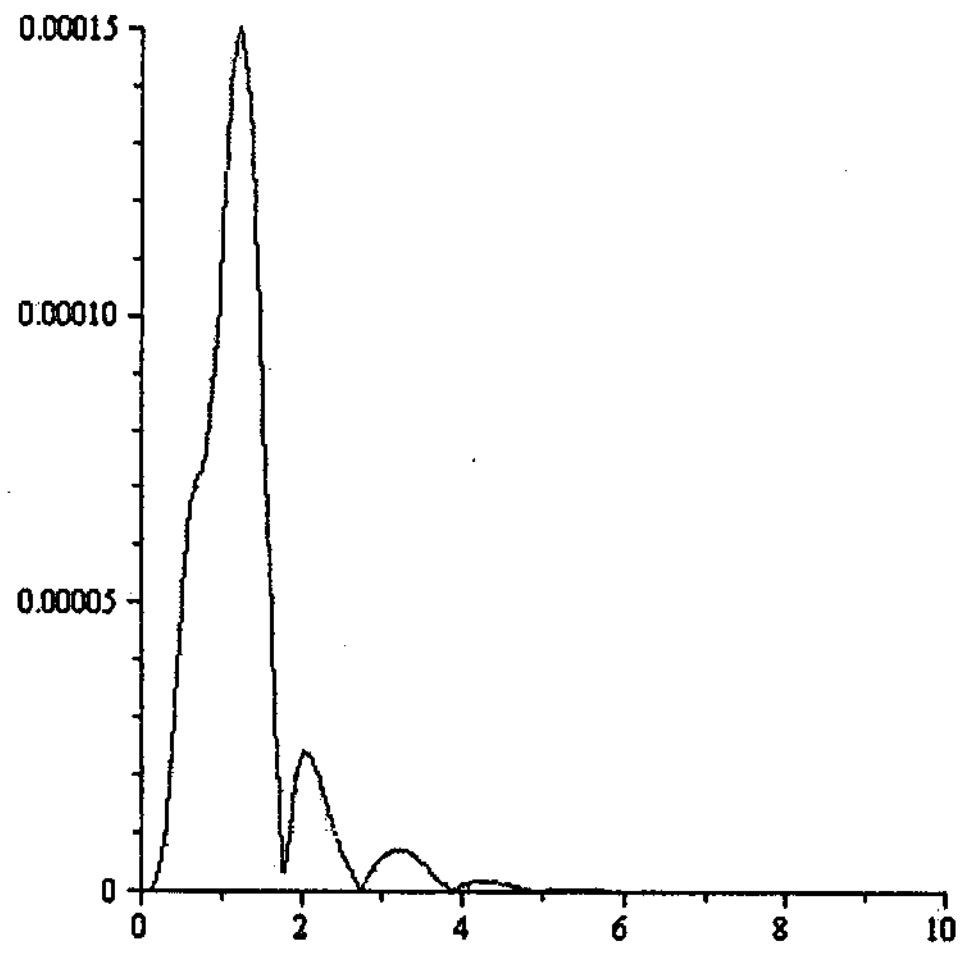


Figure 5.6: Absolute error for probe2-s.

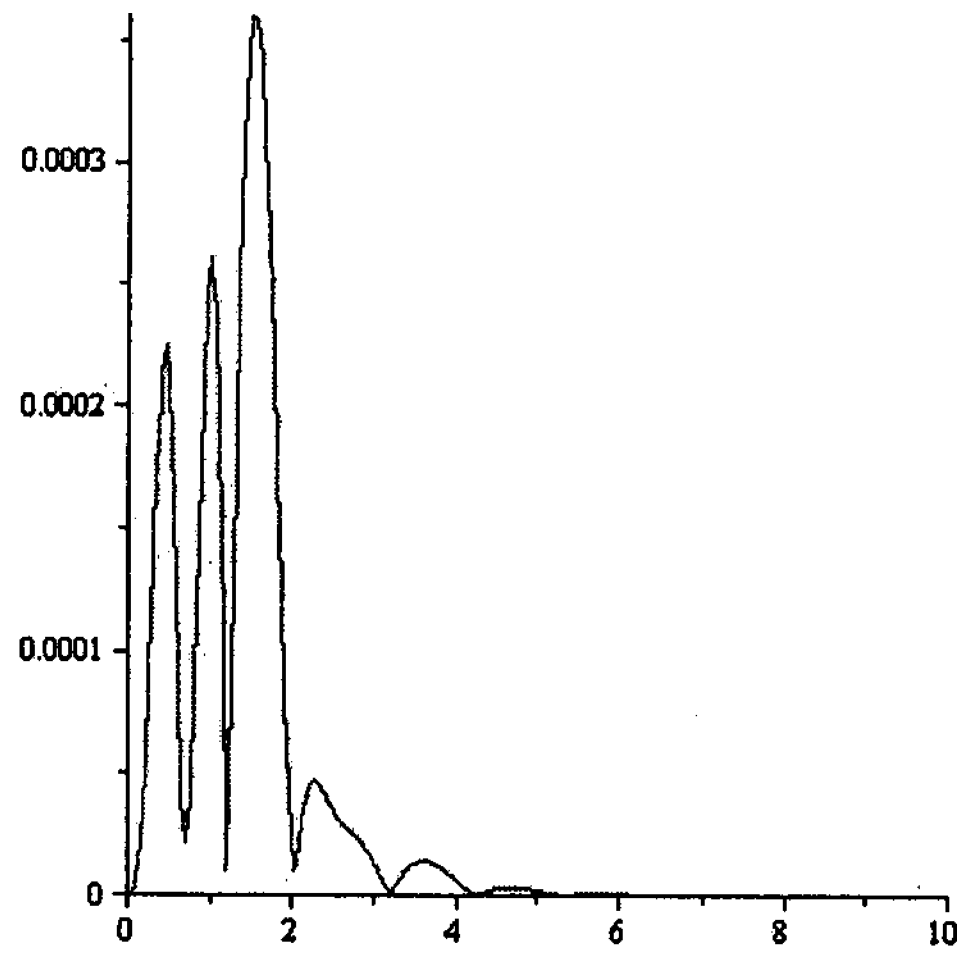


Figure 5.7: Absolute error for probe2-v.

5.3 Multi-domain system in MapleSim

In MapleSim there are many models that are built by combination of two or more systems from different fields. An example is a slider crank which is attached to a DC-motor. The DAE system of this model contains a piecewise function and due to the attached slider crank it has missing constraints. What should we do when we have a piecewise function and we need to use Bertini which accepts just polynomials? In the following example we answer this question.

5.3.1 A DC-motor attached to a slider crank

A DC-motor is an electric motor that runs on direct current (DC) electricity. This system uses electrical energy to produce mechanical energy. Here, we attached this system to a slider crank to observe the speed of slider crank [3]. This MapleSim model is given in Figure 5.8.

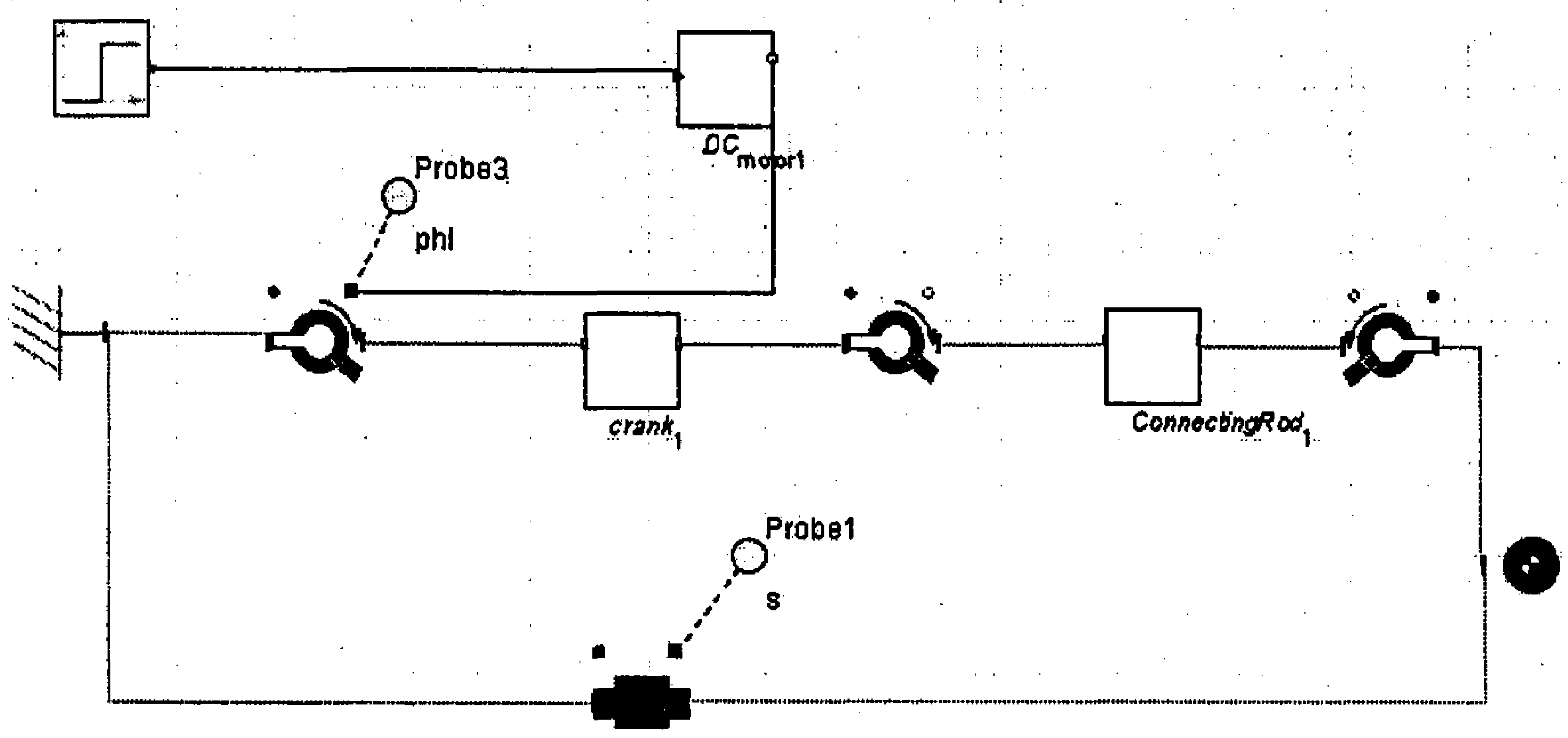


Figure 5.8: The slider crank system attached to a DC-motor in MapleSim.

The DC-motor subsystem in MapleSim which contains an RLC-Circuit with an electromotive force (`EMF`), `Inertia`, and `Rotational Damper` is shown in Figure 5.9.

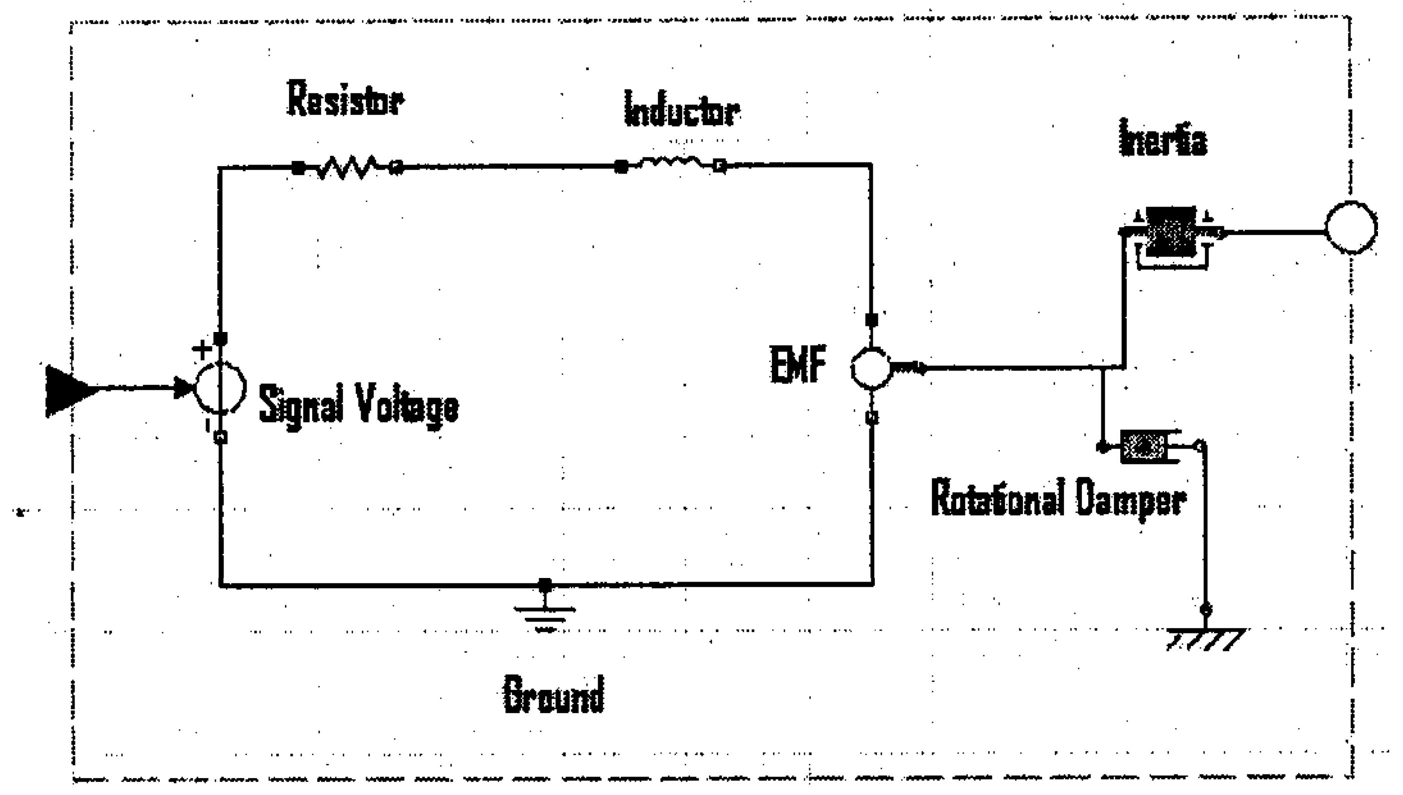


Figure 5.9: The DC-motor subsystem in MapleSim.

An electromotive force (`EMF`) is an electric/mechanic transformer, `Inertia` is a 1-D rotational component with inertia, and `Rotational Damper` is a linear 1-D rotational damper. Parameters in this subsystem are chosen to be resistance of `Resistor` 1 Ω, inductance of `Inductor` $\frac{1}{2}$ Henry, transformation coefficient of `EMF` 0.01 $\frac{Nm}{A}$. The DC-motor is connected to a `Step` which generates a real step signal. We choose the height of this step, for instance, to be 2000.

The DAE system of this model from the Modelica file is 72×72 system of the forms :

$$\begin{aligned}
&X15(t) = 0, X20(t) = 0, X14(t) - X16(t) = 0, \dots \\
&X7(t) - \frac{d}{dt}X8(t) = 0, X23 - \frac{d}{dt}X21(t) = 0, X30 - \frac{d}{dt}X60(t) = 0, \dots \\
&X3(t) = \text{msim/PIECEWISE}(X4(t) < 0, 0, true, 2000), \\
&2\sin(X55(t)) + \sin(X54(t)) = 0, X53(t) = 2\cos(X55(t)) + \cos(X54(t)), \dots
\end{aligned} \tag{5.13}$$

Here $X4(t)$ = _msim_time = t. We use DEtools[rifsimp] on the first two rows of (5.13) and substitute the result into the rest of the DAE system. This simplifies the system to a 5×5 system of equations. Moreover, we replaced the msim/PIECEWISE function in the system by a Maple piecewise function. The simplified system is:

$$\begin{aligned}
&2\sin(X55(t)) + \sin(X68(t)) = 0,\\
&\frac{1}{2}\frac{d}{dt}(X9(t)) + X9(t) + 0.01\frac{d}{dt}(X68(t)) = \begin{cases} 0 & \text{for} \quad t < 0 \\ 2000 & \text{otherwise} \end{cases},\\
&X53(t) = 2\cos(X55(t)) + \cos(X68(t)),\\
&3\frac{d^2}{dt^2}(X68(t))\cos(X68(t))\cos(X55(t)) + 3\frac{d^2}{dt^2}(X68(t))\sin(X68(t))\sin(X55(t)) + \ldots,\\
&3.26\frac{d^2}{dt^2}(X68(t)) + 3\frac{d^2}{dt^2}(X55(t))\cos(X68(t))\cos(X55(t)) + 3\frac{d^2}{dt^2}(X55(t))\sin(X68(t))\ldots
\end{aligned} \tag{5.14}$$

Here $X68(t)$ = `Probe3-phi`, $X53(t)$ = `Probe1-s`, and $X55(t)$ is the angle made by `ConnectingRod` subsystem with ground in Figure 5.8. Also $X9(t)$ is direct current of electricity, and $X49(t)$ is the Lagrange multiplier for the system.

At this point, we apply fast prolongation for this system with variables $X9(t)$, $X49(t)$, $X53(t)$, $X55(t)$, $X68(t)$ and obtain:

$$\begin{aligned}
&c_1 = 2,\ c_2 = 0,\ c_3 = 0,\ c_4 = 0,\ c_5 = 0\\
&d_1 = 1,\ d_2 = 0,\ d_3 = 0,\ d_4 = 2,\ d_5 = 2.
\end{aligned} \tag{5.15}$$

Therefore, based on (5.15), we need to differentiate first equation of (5.14) twice to obtain missing constraints. However, there is no need to differentiate the other equations in this system, since for $i > 1$ we have $c_i = 0$. The missing constraints are:

$$\begin{aligned}
2\cos(X55(t))\frac{d}{dt}(X55(t)) + \cos(X68(t))\frac{d}{dt}(X68(t)) &= 0,\\
-2\sin(X55(t))(\frac{d}{dt}(X55(t)))^2 + 2\cos(X55(t))\frac{d^2}{dt^2}(X55(t)) &-\\
\sin(X68(t))(\frac{d}{dt}(X68(t)))^2 + \cos(X68(t))\frac{d^2}{dt^2}(X68(t)) &= 0
\end{aligned} \tag{5.16}$$

Then, we make three block structures in order to find consistent initial conditions. These are obtained by the bottom up block method using Bertini. Recall, Bertini is unable to work with algebraic equations which contain functions such as sin and cos. So we have to transform these functions to produce a polynomial system:

$$\begin{aligned}&\sin(X55(t)) = n1(t), \cos(X55(t)) = n2(t),\\ &\sin(X68(t)) = n3(t), \cos(X68(t)) = n4(t).\end{aligned} \tag{5.17}$$

Moreover, we have to add two relevant trigonometric identities $n1(t)^2 + n2(t)^2 - 1 = 0$, and $n3(t)^2 + n4(t)^2 - 1 = 0$ and change to jet variable notation. The final block structure which goes to Bertini is:

- $B_0 = [2 * n1 + n3 = 0,\ n1^2 + n2^2 - 1 = 0,\ n3^2 + n4^2 - 1 = 0]$.
- $B_1 = [2 * n2 * X55t + n4 * X68t = 0]$.
- $B_2 = [-2 * n1 * X55t^2 + 2 * n2 * X55tt - n3 * X68t^2 + n4 * X68tt = 0,\ X53 = 2 * n2 + n4,\ \frac{1}{2} * X9t + X9 + 0.01 * X68t = \begin{cases} 0 & \text{for} \quad t < 0 \\ 2000 & \text{otherwise} \end{cases},$ $3 * X68tt * n4 * n2 + 3 * X68tt * n3 * n1 + 6 * X55tt - 2 * n2 * X49 + 29.43 * n2 - 3 * n2 * X68t^2 * n3 + 3 * n1 * X68t^2 * n4 = 0,\ 3.26 * X68tt + 3 * X55tt * n4 * n2 + 3 * X55tt * n3 * n1 - n4 * X49 + 24.525 * n4 - 3 * n4 * X55t^2 * n1 + 3 * n3 * X55t^2 * n2 + 0.1 * X68t - 0.01 * X9 = 0]$.

We choose initial time to be zero and send the bottom block, B_0, to Bertini. When Bertini gives back the result, we substitute that result in the next block and send it to Bertini again. This is continued up to B_2, except here for B_2 we need to do one more job due to the piecewise function, $\begin{cases} 0 & \text{for} \quad t < 0 \\ 2000 & \text{otherwise} \end{cases}$. This piecewise function should be evaluated at 'initial time'=0 since Bertini does not accept piecewise functions in its input file. In addition to substitution of Bertini's solution from other block in B_2, we need to substitute $t = 0$ and evaluate B_2 at this point as well. Therefore, that special equation with piecewise function in B_2 is changed to $\frac{1}{2} * X9t + X9 + 0.01 * X68t = 2000$ and Bertini is able to solve it.

Chapter 6

Models with Singularities

6.1 Introduction to models with singularities

There are many works done on singularities of multibody systems. This motivated us to investigate singular cases arising in MapleSim models. Arponen, Piipponen, and Tuomela in [18, 19] analyzed the singularities of planar multibody mechanisms such as the "Andrews squeezing system", and various closed bar mechanisms.

The equation of the systems above take the Lagrangian form:

$$\begin{cases} f(t, u, u_t, u_{tt}, \ldots, \lambda) = 0, \\ g(t, u) = 0, \end{cases} \tag{6.1}$$

where the function f describes the dynamical equations and g gives the constraints. Here $u \in \mathbb{R}^n$ are the (generalized) position coordinates, u_t, and u_{tt} are first and second derivatives, respectively, and λ represents the Lagrange multipliers. A singular point is where the number of degrees of freedom of the system changes. Mathematically these are the points where the rank of the Jacobian of g drops and is not full rank [20, 21].

There are some differences between [18, 19, 20] and our work. For example [18, 19, 20] do not consider the actual dynamical equations and analyze only the constraints given by g. However, in our approach we work with the whole system.

Based on [20], the singularities can be handled in one of the following ways:

- Avoided, or
- Compensated, or
- Eliminated.

Avoiding singularities mean we keep solution paths away from singularities. The works [18, 19] are directed at finding the set of such singularities for multibody systems. Arponen [20] shows that a two bar slider crank admits a singularity if and only if the lengths of the bars are equal. Thus a slider crank with different length bars we considered in Section 4.2.1 is an example of the techniques of avoiding singularities. Compensating for singularities means using techniques to overcome or compensate for the singularity. Elimination means that we find the singularity's cause and reformulate the equations in such a way that the cause of singularity is eliminated. Arponen in [20] uses a different approach to [18, 19] in that he presents a new method for eliminating singularities for a large class of multibody models.

The papers [18, 19, 20] use algebraic tools such as fitting ideals, Gröbner bases etc, which work only for polynomial systems. They used the `Singular` freeware [35] which is a computer algebra system for polynomial computations with special emphasis on the needs of commutative algebra, algebraic geometry, and singularity theory. Since multibody equations contain trigonometric functions, to use algebraic techniques, the constraints equations need to be changed to polynomial form. We also need to change from trigonometric to polynomial form to use the Bertini software for polynomial systems.

There are two ways to formulate models of multibody dynamics. The first one uses a minimum number of variables with constraint forces eliminated, e.g. in slider crank using r, θ as the variables where r is the bar length and θ is the angle which is made by the bar. This is an example of the elimination method. The other formulation uses Cartesian coordinates, keeping the constraint forces within the equations of motion which are augmented by the constraints. This is called the augmentation method and the system is said to be in descriptor form.

The method in [20] by Arponen for the slider crank works in descriptor form. Arponen used Cartesian coordinates (x_i, y_i) based on positions of the two bars. Our method for the slider crank is an elimination method. We start with r and θ and only convert to polynomial form for these equations being solved by Bertini.

6.1.1 Equal-length bar slider crank

We consider an equal-length bar (length=1 meter) slider crank (see Figure 6.1). This system is known [20] to admit a singularity.

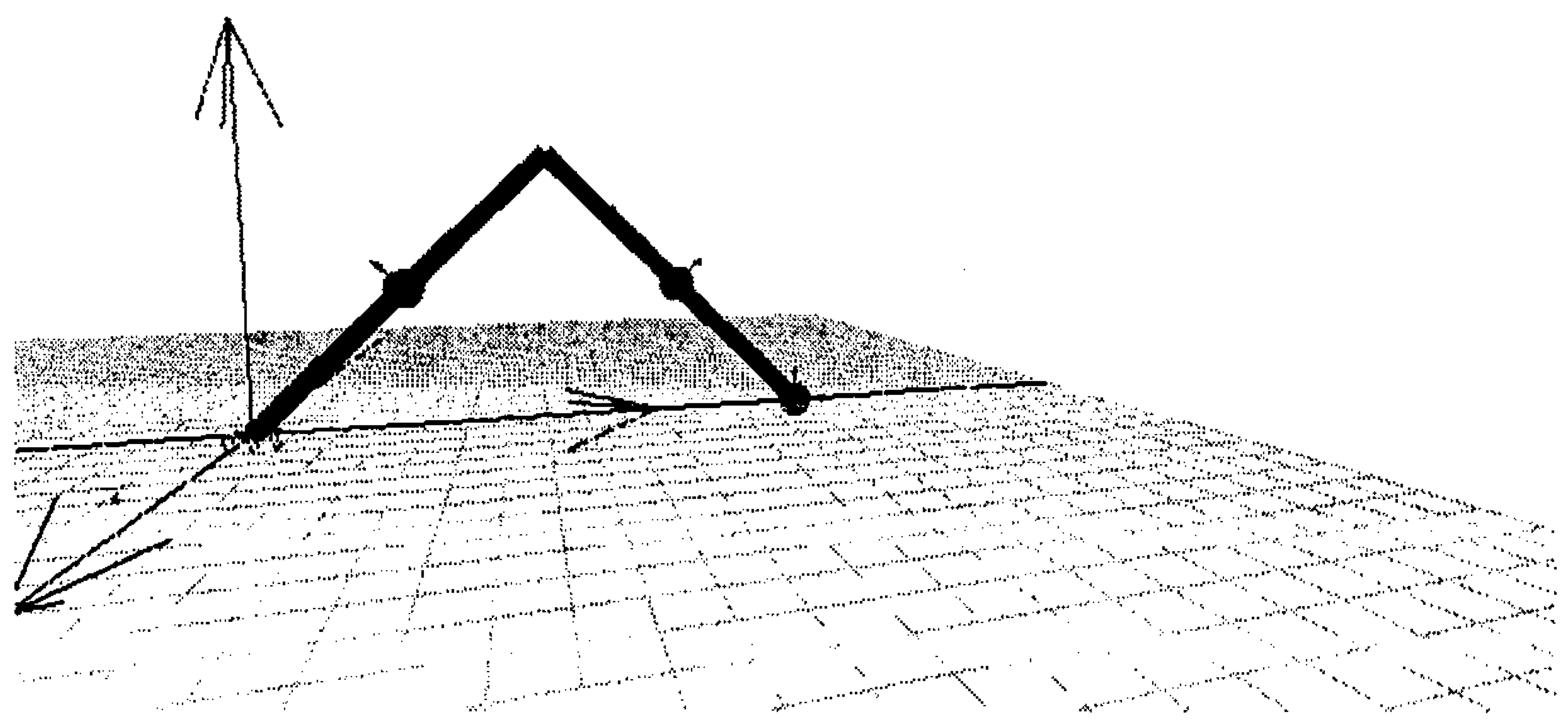


Figure 6.1: Equal-length bar slider crank.

In a similar manner to [20] we analyze the constraints. We start with bottom block. Finally, after constructing all blocks use of Bertini enables us to find the singular points. We now give a formal definition [18] of singular points:

Definition 6.1.1. *Let $f : \mathbb{R}^n \to \mathbb{R}^k$ be any smooth map where $k < n$ and let df be its Jacobian matrix. Let $M = f^{-1}(0) \subset \mathbb{R}^n$ be the zero set of f. A point $q \in M$ is a singular point of M, if df does not have maximal rank at q.*

We observe that the MapleSim simulation is very slow near the singularity. This occurs because as the Jacobian becomes singular the step size is made smaller and smaller to retain accuracy.

In particular we set the final time, t_f in MapleSim to be 10 seconds but found that the simulation stops at 1.605 seconds with a message indicating a possible singularity at $t = 1.625$. We generate a Modelica file for our system and transfer all the system's information to a Maple worksheet. The slider crank figure in MapleSim simulation at the singular point corresponds to the animations below in Figures 6.2 and 6.3. In particular we observe as does Arponen [20] that the singularities occur where the bars overlap.

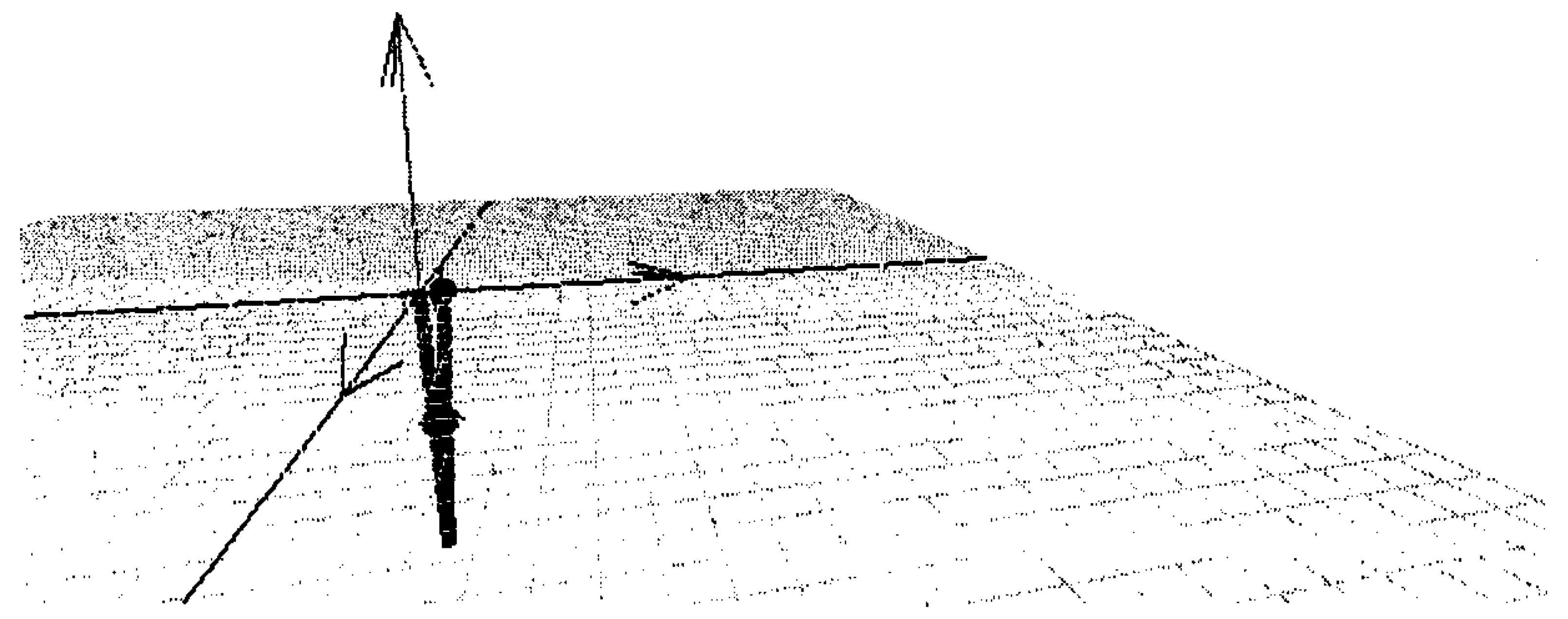


Figure 6.2: Near singular configuration of the equal-length bar slider crank.

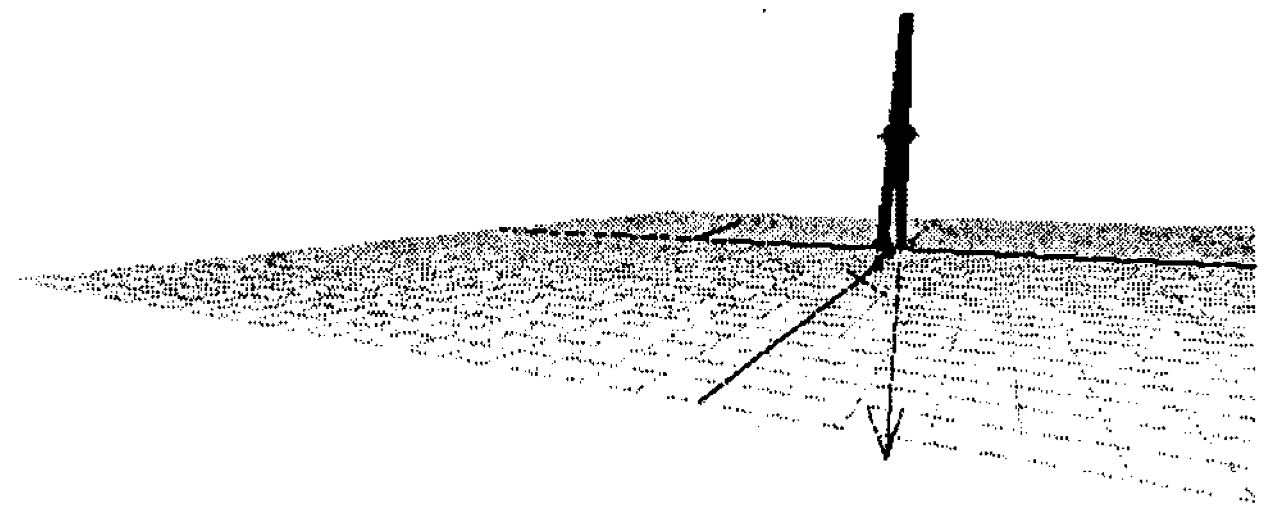


Figure 6.3: Near singular configuration of the equal-length bar slider crank.

As discussed in Section 4.2.1 the DAE system of this model is a 23×23 system which simplifies to a 4×4 system. Then, using the fast prolongation package, two missing constraints are found. They are used to determine the blocks:

- $B_0 = [\sin(X17(t)) + \sin(X9(t)) = 0]$.
- $B_1 = [\cos(X17(t))\frac{d}{dt}X17(t) + \cos(X9(t))\frac{d}{dt}X9(t) = 0]$.
- $B_2 = [-\sin(X17(t))(\frac{d}{dt}X17(t))^2 + \cos(X17(t))\frac{d^2}{dt^2}X17(t) - \sin(X9(t))(\frac{d}{dt}X9(t))^2 + \cos(X9(t))\frac{d^2}{dt^2}X9(t) = 0,\ X6(t) = \cos(X17(t)) + \cos(X9(t)), \ldots]$

The variables $X9(t)$, $X17(t)$ are indicated in Figure 6.4.

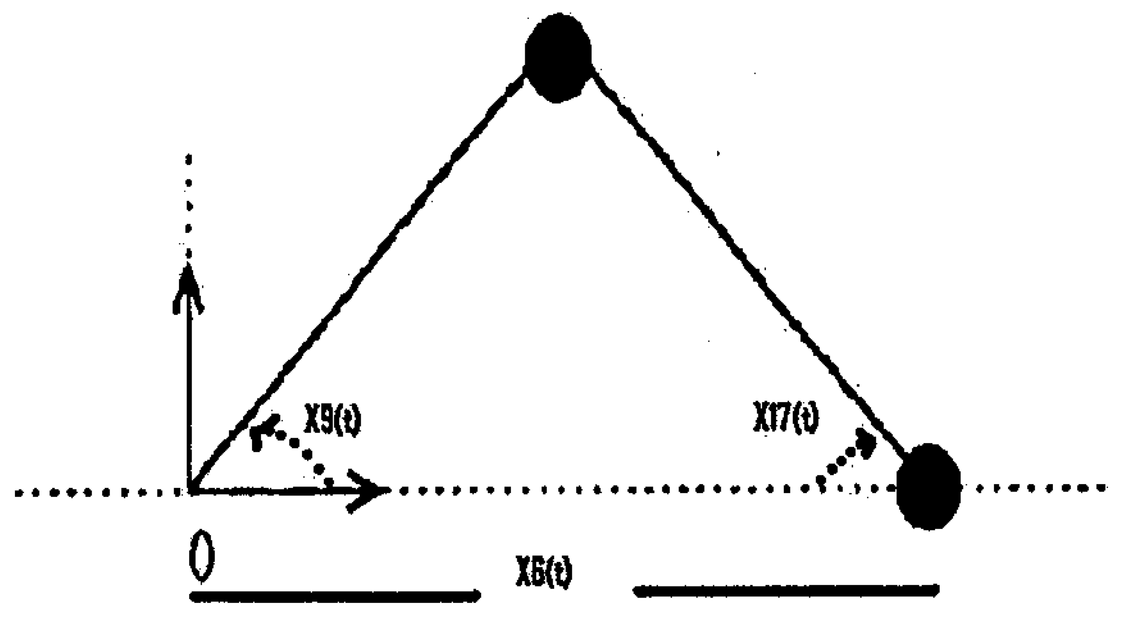


Figure 6.4: Variables for the slider crank.

In terms of probes, $X9(t)$ is `Probe2-phi` and $X6(t)$ is `Probe1-s`. Unlike the slider crank in [20] that only uses Cartesian coordinates, we only change our system to such polynomial coordinates system when required by the Bertini software. The blocks after changing to such coordinates are:

- $B_0 = [n3 + n1 = 0, n1^2 + n2^2 - 1 = 0, n3^2 + n4^2 - 1 = 0]$.
- $B_1 = [n4 * X17t + n2 * X9t = 0]$.
- $B_2 = [-n3 * X17t^2 + n4 * X17tt - n1 * X9t^2 + n2 * X9tt = 0, X6 = n4 + n2, 3.25 * X9tt + 1.5 * X17tt * n2 * n4 + 1.5 * X17tt * n1 * n3 - n2 * X11 + 24.525 * n2 - 1.5 * n2 * X17t^2 * n3 + 1.5 * n1 * X17t^2 * n4 = 0, 1.5 * X9tt * n2 * n4 + 1.5 * X9tt * n1 * n3 + 2.25 * X17tt - n4 * X11 + 14.715 * n4 - 1.5 * n4 * X9t^2 * n1 + 1.5 * n3 * X9t^2 * n2 = 0]$.

Here $\sin(X9(t)) = n1$, $\cos(X9(t)) = n2$, $\sin(X17(t)) = n3$, and $\cos(X17(t)) = n4$. We use the initial condition $X9(t) = \pi/4$ from the Modelica file. We use bottom up substitution with Bertini in order to find all initial conditions for this system. When Maple's `dsolve/numeric` is used it gives a max number of steps exceeded (`maxfun`) error, due to the singularity. Graphs of `Probe2.phi` ($X9(t)$) close to the singular point are given in Figures 6.5 and 6.6.

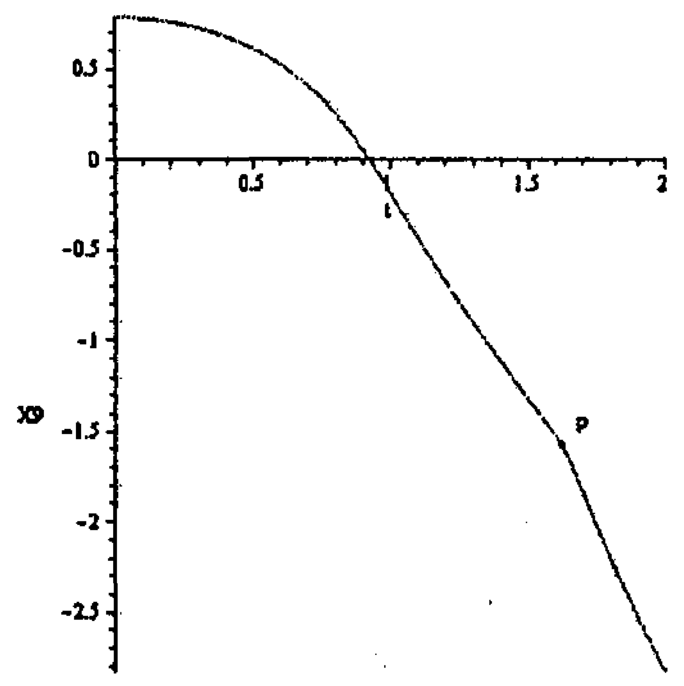


Figure 6.5: Probe2.phi result close to singular point P by `dsolve` for DAE system of the equal-length bar slider crank.

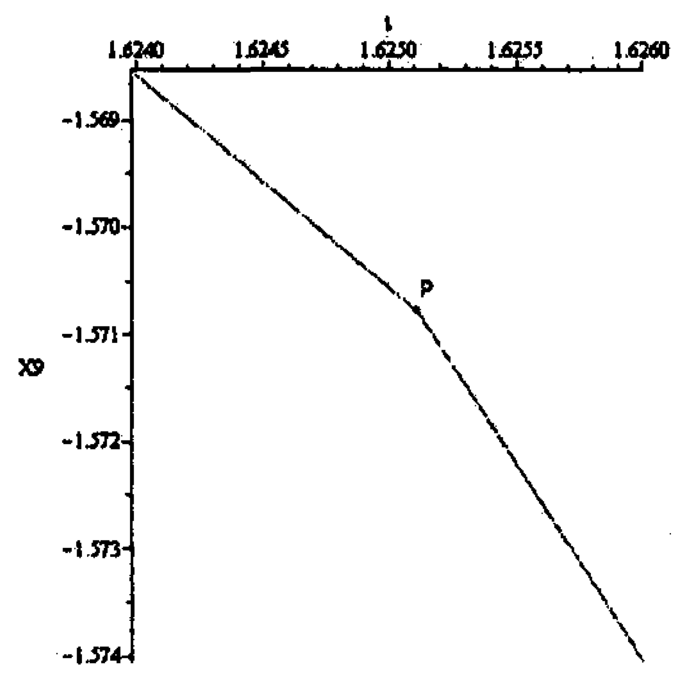


Figure 6.6: Probe2.phi result close to singular point P of the equal-length bar slider crank.

The result from MapleSim and Maple worksheet are the same as each other before this singular point, at $t = 1.625$:

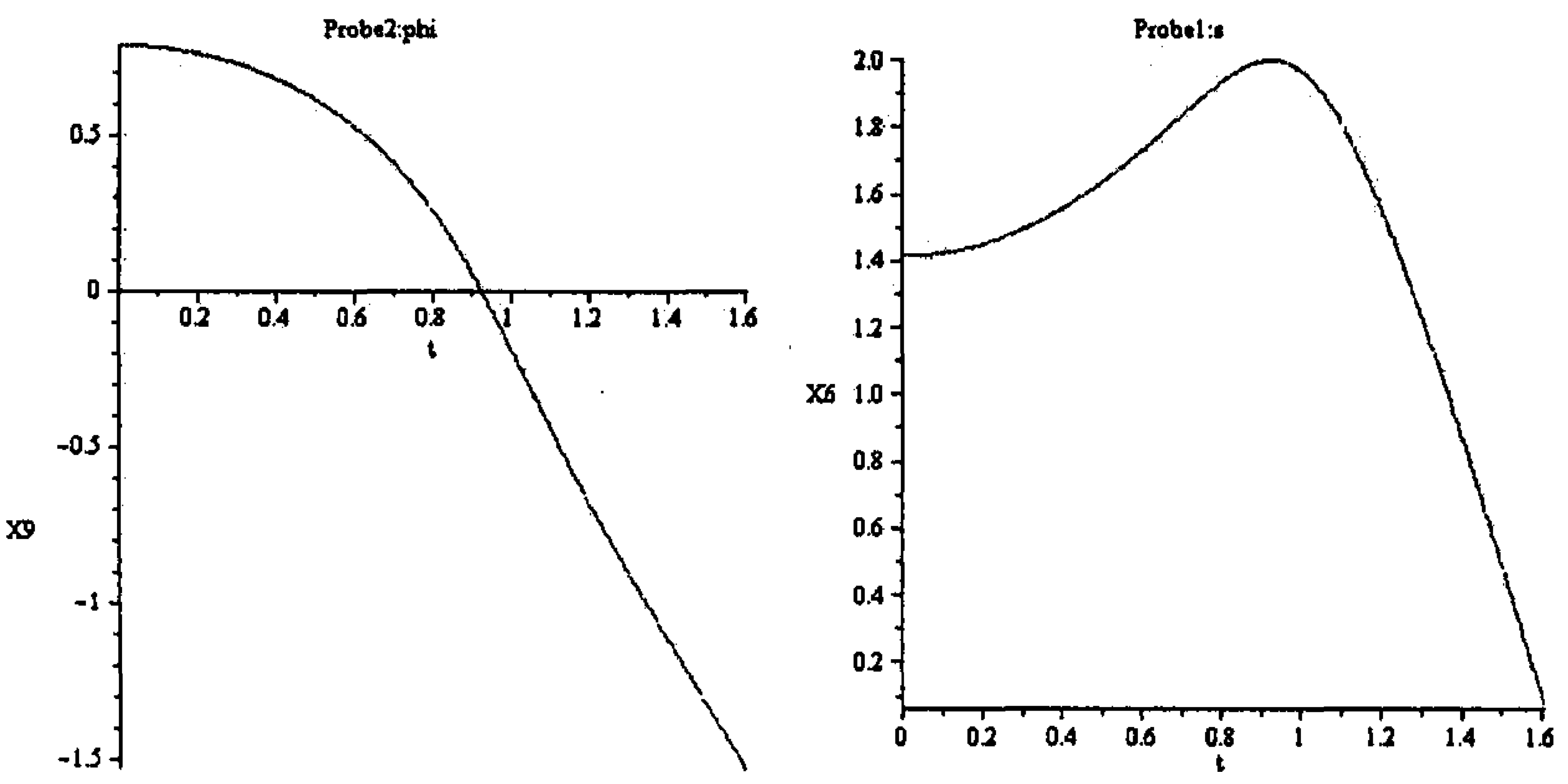


Figure 6.7: Maple result for the equal-length bar slider crank before singularity at $t = 1.625$.

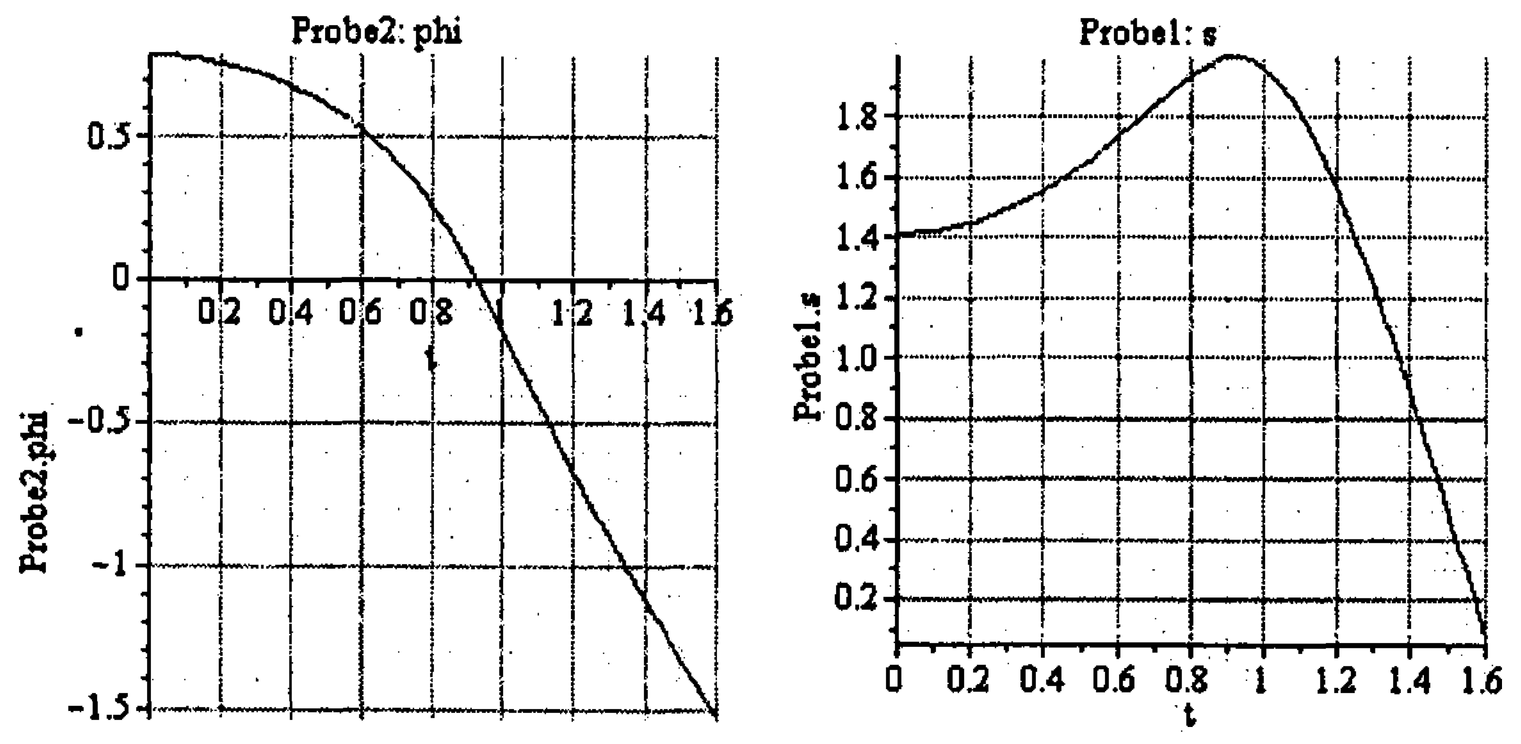


Figure 6.8: MapleSim result for the equal-length bar slider crank before singularity.

Now we want to investigate the origin of the singular point. Based on Figures 6.2 and 6.3 the singular point occurs when:

$$X9(t) = \pi/2,\ X17(t) = -\pi/2,\ X6(t) = 0,$$
$$n1 = 1,\ n2 = 0,\ n3 = -1,\ n4 = 0. \tag{6.2}$$

or

$$X9(t) = 3\pi/2,\ X17(t) = -3\pi/2,\ X6(t) = 0,$$
$$n1 = -1,\ n2 = 0,\ n3 = 1,\ n4 = 0. \tag{6.3}$$

Like Arponen in [20] and Piipponen in [19], we want to find the polynomial decomposition of the bottom block which contains all constraints. Decomposition is a generalization of polynomial factorization (see [21]). Therefore, we use Bertini to decompose the bottom block:

$$n3 + n1 = 0, n1^2 + n2^2 - 1 = 0, n3^2 + n4^2 = 1. \tag{6.4}$$

The Bertini input file for this system of polynomials is:

```
CONFIG
TRACKTYPE: 1;
END;
INPUT
variable_group n1, n3, n2, n4;
function f1,f2,f3;
f1=n3+n1;
f2=n1^2+n2^2-1;
f3=n3^2+n4^2-1;
END;
```

Figure 6.9: A sample Bertini input file.

We choose TRACKTYPE:= 1 in the file 6.9 (TRACKTYPE:= 0 by default). This instructs Bertini to use its positive dimension solver. Recall from Chapter 3, Bertini can give dimension and degree of the components in the decomposition of a polynomial system. It also gives some information about singularities. Bertini shows that there are two components each degree two and dimension one which are both nonsingular:

- Component-1: $C_1 : n1 + n3 = 0,\ n2 - n4 = 0.$

- Component-2: $C_2 : n1 + n3 = 0,\ n2 + n4 = 0.$

In angular coordinates this is equivalent to:

- Component-1: $\sin(X9(t)) = -\sin(X17(t))$, $\cos(X9(t)) = \cos(X17(t))$.
- Component-2: $\sin(X9(t)) = -\sin(X17(t))$, $\cos(X9(t)) = -\cos(X17(t))$.

Thus one of the cases is:

- Component-1: $T_1 : X17(t) = -X9(t)$.
- Component-2: $T_2 : X17(t) = X9(t) - \pi$.

Figures 6.10 and 6.11 shows the components T_1 and T_2.

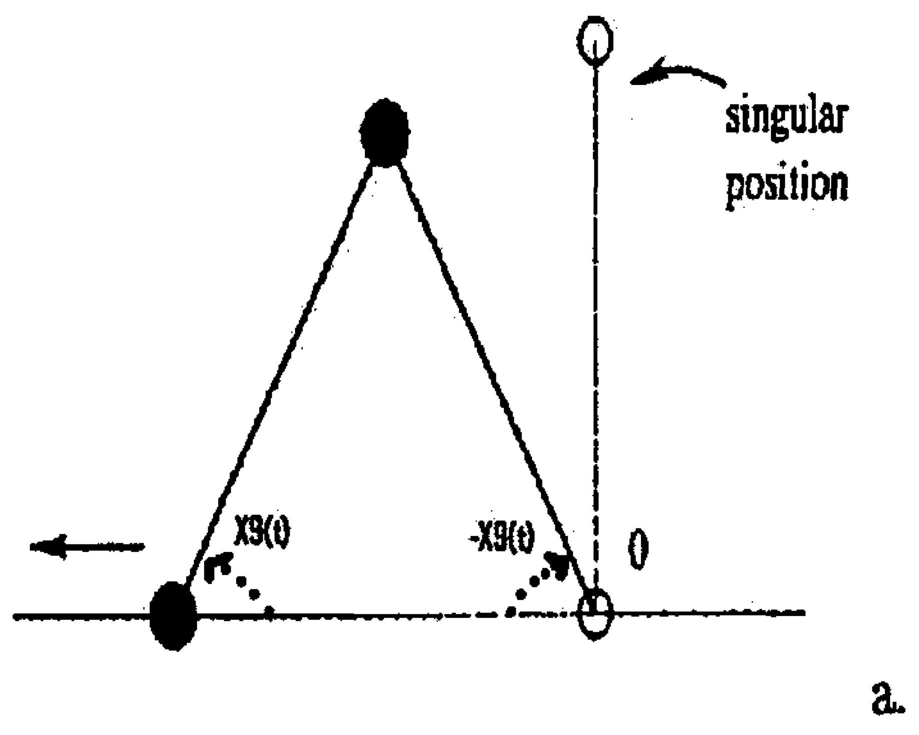


Figure 6.10: First component of bottom block for equal-length bar slider crank.

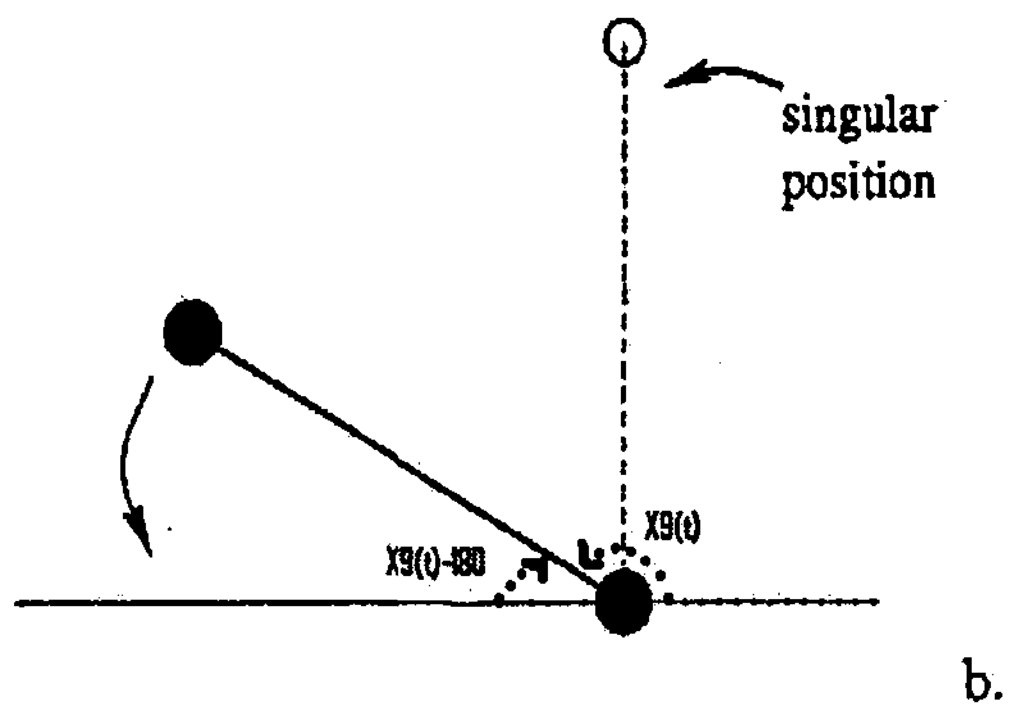


Figure 6.11: Second component of bottom block for equal-length bar slider crank.

As we found in [19, 20], the singular points are at the intersection $C_1 \cap C_2$ of these two components:

$$C_1 \cap C_2 = \{n1 = 1,\ n2 = 0,\ n3 = -1,\ n4 = 0\} \text{ or } \{n1 = -1,\ n2 = 0,\ n3 = 1,\ n4 = 0\}. \tag{6.5}$$

Now we want to show that the Jacobian of (6.4) is full rank for these two components and it is not full rank at their intersection. The Jacobian of (6.4) respect to $[n1, n2, n3, n4]$ is:

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 2n1 & 2n2 & 0 & 0 \\ 0 & 0 & 2n3 & 2n4 \end{bmatrix} \tag{6.6}$$

We can check and see that rank of the Jacobian at each point of (6.5) is 2 which is not full rank ($= 3$). However at C_1 and C_2 it is 3 which is full rank ($= 3$) away from $C_1 \cap C_2$.

Now I would like to show bottom block decomposition using r, θ coordinates (i.e using the elimination method).

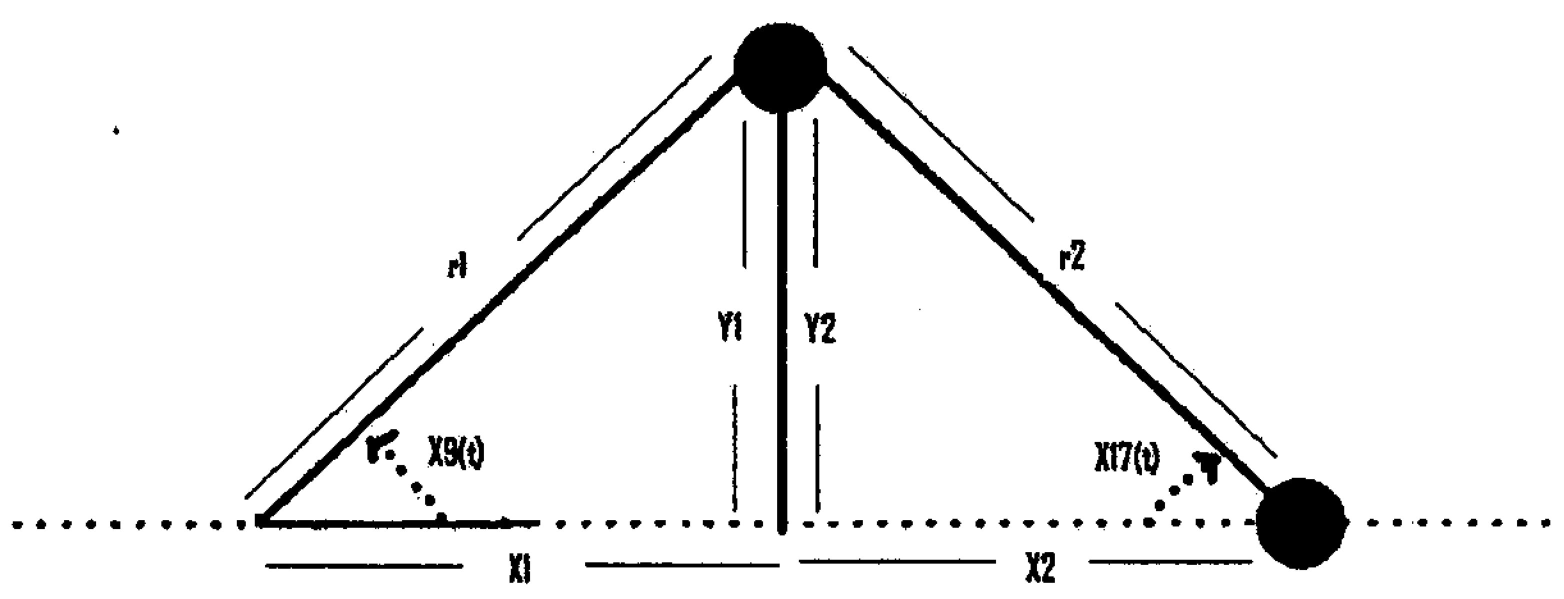


Figure 6.12: Equal-length bar slider crank position.

In Figure 6.12

$$X1 = r1\cos(X9(t)),\ Y1 = r1\sin(X9(t)),$$
$$X2 = r2\cos(X17(t)),\ Y2 = -r2\sin(X17(t)). \tag{6.7}$$

Therefore, the bottom block constraints will be:

$$\begin{aligned} X1^2 + Y1^2 &= 1, \\ X2^2 + Y2^2 &= 1, \\ Y1 &= Y2. \end{aligned} \tag{6.8}$$

In angular coordinate (6.8) is:

$$\begin{aligned} r1^2 - 1 &= 0, \\ r2^2 - 1 &= 0, \\ r1\sin(X9(t)) + r2\sin(X17(t)) &= 0. \end{aligned} \tag{6.9}$$

The decomposition of (6.9) is:

- Component-1: $r1 = 1$, $r2 = 1$, $\sin(X9(t)) + \sin(X17(t)) = 0$,
- Component-2: $r1 = 1$, $r2 = -1$, $\sin(X9(t)) - \sin(X17(t)) = 0$,
- Component-3: $r1 = -1$, $r2 = 1$, $-\sin(X9(t)) + \sin(X17(t)) = 0$,
- Component-4: $r1 = -1$, $r2 = -1$, $\sin(X9(t)) + \sin(X17(t)) = 0$,

Components 2, 3, and 4 are physically impossible since the bar lengths, $r1$, $r2$ are positive and one. The only physically possible component is Component-1 which yields the cases:

$$[r1 = 1,\ r2 = 1,\ \sin(X9(t)) + \sin(X17(t)) = 0] = \begin{cases} T_1 : X17(t) = -X9(t), \\ T_2 : X17(t) = X9(t) - \pi. \end{cases} \tag{6.10}$$

Thus using elimination method (r, θ system) we can have two components which are the same as before related to Figures 6.10 and 6.11. Therefore the two methods give the same result.

Recall from Chapter 3, the Bertini software is able to give information about singular and non-singular solutions of a system. In order to do that, we can convert B_0, B_1, and

B_2 to a system of polynomials. In particular Bertini's input file contains:

$$
\begin{aligned}
& n3 + n1 = 0, \\
& n1^2 + n2^2 - 1 = 0, \\
& n3^2 + n4^2 - 1 = 0, \\
& n4 * X17t + n2 * X9t = 0, \\
& - n3 * X17t^2 + n4 * X17tt - n1 * X9t^2 + n2 * X9tt = 0, \\
& X6 = n4 + n2, \\
& 3.25 * X9tt + 1.5 * X17tt * n2 * n4 + 1.5 * X17tt * n1 * n3 - n2 * X11 + \\
& 24.525 * n2 - 1.5 * n2 * X17t^2 * n3 + 1.5 * n1 * X17t^2 * n4 = 0, \\
& 1.5 * X9tt * n2 * n4 + 1.5 * X9tt * n1 * n3 + 2.25 * X17tt - n4 * X11 + \\
& 14.715 * n4 - 1.5 * n4 * X9t^2 * n1 + 1.5 * n3 * X9t^2 * n2 = 0.
\end{aligned}
\tag{6.11}
$$

Bertini shows that there are six two-dimensional components shown in Table 6.1.

singular	dim	Equations of components
N	2	$X17t = X9t$, $X17tt = X9tt$, $X6 = 0$, $n1 + n3 = 0$, $n2 + n4 = 0$
N	2	$X17t = -X9t$, $X17tt = -X9tt$, $n1 + n3 = 0$, $n2 - n4 = 0$
Y	2	$X17t = X9t$, $X17tt = X6 = X9tt = 0$, $n1 = -1$, $n2 = 0$, $n3 = 1$, $n4 = 0$
Y	2	$X17t = -X9t$, $X17tt = X6 = X9tt = 0$, $n1 = 1$, $n2 = 0$, $n3 = -1$, $n4 = 0$
Y	2	$X17t = X9t$, $X17tt = X6 = X9tt = 0$, $n1 = 1$, $n2 = 0$, $n3 = -1$, $n4 = 0$
Y	2	$X17t = -X9t$, $X17tt = X6 = X9tt = 0$, $n1 = -1$, $n2 = 0$, $n3 = 1$, $n4 = 0$

Table 6.1: Six components from Bertini for solving all blocks of equal-length bar slider crank

Chapter 7

Conclusion and Future Work

The analysis of nonlinear systems of DAEs is becoming more common in applications. For such systems, it is required to uncover all hidden constraints in order to determine their initial conditions and numerically solve them. In general, this can be more challenging when DAEs are complicated and automatically generated by a computer software. In this thesis, we worked on DAE systems which arise from MapleSim software. MapleSim is a multi-domain modeling and simulation tool which enables us to make models from diverse fields. This software is powered by the Maple software and systems of DAEs from the MapleSim models can be transferred to Maple worksheet.

Wu, Reid and Ilie introduced a fast numeric geometric approach [1, 2] in the framework of Riquier Bases for PDAEs. In those works, it is shown that this approach can be very useful in finding all missing constraints for such systems. Their methods are applicable to a class of PDAEs that are square systems dominated by pure derivatives in one of their independent variables with respect to some partial ranking. They showed for this class, only differentiations (prolongations) with respect to one independent variable are needed to determine missing constraints. Their method is called the fast prolongation method.

Since system of DAEs are dominated by pure derivatives in t (time here), we used the fast prolongation method to find missing constraints and consistent initial conditions for solving DAE from MapleSim. One of the contribution of this thesis is to show that the fast prolongation method can be successfully applied to many MapleSim models from diverse fields.

We created some of the models and others were obtained from MapleSim example menus. The systems of DAEs from MapleSim usually contain many simple redundant equations. Another contribution is to show that these can be used to dramatically reduce the number of variables and equations in the system. In particular we used Maple's DEtools[rifsimp] on those redundant equations to simplify the systems. Next, the fast prolongation package is applied to the simplified system in order to find missing constraints. If any missing constraints are found, the block structure is computed. We use the block structure and Bertini to find initial conditions and finally numerically solve the DAE system. There is a discussion about our algorithm for MapleSim generated DAE models in Chapter 4. A slider crank system in MapleSim is used here to illustrate our approach.

The Bertini software is a global homotopy continuation solver introduced by Sommese et al. [16]. Bertini applies to polynomial systems rather than algebraic equations to find zero dimensional (isolated) solution as well as representing positive dimensional solution sets of polynomial systems. It uses witness points to represent positive dimensional solution sets (see Chapter 3). In Chapter 3 we also discussed homotopy continuation methods.

In Chapter 5 we consider two MapleSim models: a nonlinear damper with a linear spring and a DC-motor attached to a slider crank system. We made a nonlinear damper with a linear spring using a so-called custom component. Such custom components are a powerful feature of MapleSim and allow us to add new components to the MapleSim library. A DC-motor attached to a slider crank system is a multi-domain model with a piecewise function. In particular we successfully used our methods with Bertini on such a system. We discuss the way Bertini should be used when we have piecewise function which is not acceptable in the Bertini input file.

Another goal in this thesis was to work on models with singularities. In particular the work on singularities [18, 19, 20] by Arponen, Piipponen, and Tuomela motivated us to study the singularities of an equal-length bar slider crank. In Chapter 6 we compare our method on singularities with the methods in [18, 19, 20]. Also we show how to use Bertini to detect singularities.

The fast prolongation method only works on square DAE systems. Although most of the MapleSim models have equal number of variables and equations after simplification, non-square (overdetermined) DAE systems can be generated. An important future work is to extend this method to non-square systems.

Only a few of the MapleSim models we studied so far have missing constraints. An important future work is to study MapleSim models with missing constraints.

Another important future research area is to study models with singularities. Arponen, Piipponen, and Tuomela analyzed the singularities of planar multibody mechanisms such as the "Andrews squeezing system", and various closed bar mechanisms. Making these models in MapleSim and analyzing their singularities using the techniques of the thesis would be a very interesting future work.

The last but not least open question and future work is how to apply fast prolongation method to systems of DAEs from hybrid models. Hybrid models are models with switching. In these models there is at least one condition on the system which discontinuously changes the DAE system if that condition is satisfied. For example the DAE system may contain a condition such as.

$$\text{if} \quad X3(t) < 5 \quad \text{then} \quad X9(t) = -5 \quad \text{otherwise} \quad \frac{d}{dt}X8(t) = 5. \tag{7.1}$$

Therefore the DAE system can discontinuously change. A very interesting open question is how to apply fast prolongation to this kind of system.

Bibliography

[1] W. Wu and G. Reid. Symbolic-numeric Computation of Implicit Riquier Bases for PDE. *Proc. of ISSAC'07. ACM*, pages 377–385, 2007.

[2] W. Wu, G. Reid, and S. Ilie. Implicit Riquier Bases for PDAE and their Semi-Discretizations. *Journal of Symbolic Computaion*, 44:923–941, 2009.

[3] Maplesoft Corporation. *MapleSim: Multidomain modeling and simulation tool. Available at www.maplesoft.ca*, 2009, Access date:August 27, 2010.

[4] Maplesoft a division of Waterloo Maple Inc. *MapleSim User's Guide*, 2009.

[5] F. Boulier, D. Lazard, F. Ollivier, and M. Petitot. Representation for the radical of a finitely genereted differential ideal. *Proc. ISSAC 1995. ACM Press*, pages 158–166, 1995.

[6] G.J. Reid, P. Lin, and A.D. Wittkopf. Differential-Elimination completion Algorithms for DAE and PDAE. *Studies in Applied Mathematics*, 106(1):1–45, 2001.

[7] E. Hubert. Notes on triangular sets and triangulation-decomposition algorithms ii: Differential Systems in: U. Langer., and F. Winkler. (eds.). *Symbolic and Numerical Scientific Computations. In: LNCS*, 2630, Springer-Verlag Heidelberg, 2003.

[8] E. Mansfield. *Differential Grobner Bases.* Ph.D. Thesis, University of Sydney, 1991.

[9] W.M. Seiler. *Involution-The Formal Theory of Differential Equations and its Applications in Computer Algebra and Numerical Analysis.* Habilitation Thesis, University of Mannheim, 2002.

[10] F. Boulier. *Récriture algébrique dans les systèmes d'équations différentielles en vue d'applications dans les Science du Vivant.* Habilitation Thesis, 2006.

[11] A. Wittkopf. *Algorithms and Implementations for Differential Elimination.* Ph.D. Thesis, Simon Fraser University, 2004.

[12] J.D. Pryce. A simple structure analysis method for DAEs. *BIT*, 41:364–394, 2001.

[13] F. Ollivier. *Historical Material on Jacobi's Bound (including translations).* http://www.lix.polytechnique.fr/ ollivier/JACOBI/jacobi.htm, Access date=August 27, 2010.

[14] S. Ilie, R.M. Corless, and G. Reid. Numerical solution of index-1 differential algebraic equations can be computed in polynomial time. *Numerical Algorithms*, 41(2):161–171, 2006.

[15] R.M. Corless and S. Ilie. Polynomial cost for solving IVP for high-index DAE. *BIT Numerical Mathematics*, 48:29–49, 2008.

[16] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C. W. Wampler. *Bertini: Software for Numerical Algebraic Geometry. Available at www.nd.edu/ sommese/bertini.*, 2008.

[17] L.F. Shampine and R.M. Corless. Initial value problems for ODEs in problem solving environments. *Journal of Computational and Applied Mathematics*, 125:31–40, 2000.

[18] T. Arponen, S. Piipponen, and J. Tuomela. Analysing singularities of a benchmark problem. *Multibody System Dynamics*, 19:227–253, 2008.

[19] S. Piipponen. Singularity analysis of planar linkages. *Multibody System Dynamics*, 22:223–243, 2009.

[20] T. Arponen. Regularization of Constraint Singularities in Multibody Systems. *Multibody System Dynamics*, 6(4):355–375, 2001.

[21] D. Cox, J. Little, and D. OŚhea. *Ideals, Varieties, and Algorithms, An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Springer, New York.

[22] C.J. Rust. *Rankings of derivatives for elimination algorithms and formal solvability of analytic partial differential equations.* Ph.D. Thesis, University of Chicago, 1998.

[23] A.J. Sommese and C.W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science.* World Scientific Press, Singapore, 2005.

[24] D.J Bates. *Theory and applications in numerical algebraic geometry.* Ph.D. Thesis, University of Notre Dame, 2006.

[25] J.D. Hauenstein. *Regeneration, local dimension, and applications in numerical algebraic geometry.* Ph.D. Thesis, University of Notre Dame, 2009.

[26] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software*, 25(2):251–276, 1999, Software available at http://www.math.uic.edu/~jan.

[27] T.L. Lee., T.Y. Li., and C.H. Tsai. HOM4PS-2.0: A software package for solving polynomial systems by the polyhedral homotopy continuation method. 2008, Software available at http://www.mth.msu.edu/~li/.

[28] S. Goedecker. Remark on algorithms to find roots of polynomials. *SIAM Journal on Scientific computing*, 15(5):1059–1063, 1994.

[29] V.Y. Pan. Solving a polynomial equation: some history and recent progress. *SIAM Rev.*, 39(2):187–220, 1997.

[30] A.P. Morgan and A.J. Sommese. A homotopy for solving general polynomial systems that respects m-homogeneous structures. *Appl. Math. Comput.*, 24(2):101–113, 1987.

[31] A.J. Sommese, J. Verschelde, and C.W. Wampler. Solving polynomial systems equation by equation, *In Algorithms in Algebraic Geometry* edited by A. Dickenstein, F.-O. Schreyer, and A.J. Sommese. *IMA Volumes in Mathematics and Its Applications*, 146:133–152, 2007.

[32] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. Software for numerical algebraic geometry: a paradigm and progress toward its implementation.*Software for Algebraic Geometry,* edited by M.E. Stillman, N. Takayama, and J. Verschelde. *IMA Volume in Mathematics and its Applications*, 148:1–14, 2008.

[33] M. Otter and H. Elmqvist. Modelica Languages, Libraries, Tools, Workshop, and EU-Project RealSim. *German Aerospace Center, Oberpfaffenhofen, Germany and Dynasim AB, Lund, Sweden*, 2001.

[34] L.M. Milne-Thomson. *The Calculus Of Finite Differences.* Macmillan And Company, London, 1933.

[35] G.M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3.0. A Computer Algebra System for Polynomial Computations. *Centre for Computer Algebra, University of Kaiserslautern, Software available at http://www.singular.uni-kl.de.*, 2005.