Automatic Formulation of Lagrangian DAEs

Jae Hyeuk Suk

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science in Mechanical Engineering

University of Washington

2001

Program Authorized to Offer Degree: Mechanical Engineering

University of Washington

Graduate School

This is to certify that I have examined this copy of a master's thesis by

Jae Hyeuk Suk

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Brain C. Fabien

_____

Martin C. Berg

_____

Santosh Devasia

Date: _____

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purpose or by any means shall not be allowed without my written permission.

Signature_____

Date_____

University of Washington

Abstract

Automatic Formulation of Lagrangian DAEs

by Jae Hyeuk Suk

Chair of Supervisory Committee:

Professor Brain C. Fabien
Mechanical Engineering department

This thesis presents a computer package for the automatic formulation of Lagrange's equation of motion. First the basic concepts of Lagrange's equation and its' advantages are presented. The derivation of Lagrange's equation, which include the lagrange multiplier to accommodate constraint equations, will be shown in detail. The generalized Lagrange's equation is used to construct the equations of motion for multi discipline systems.

Next, several important concept of object-oriented programming are introduced. Especially, basic requirements of symbolic computation program, such as creation of symbol object, symbolic calculus, derivative symbolically and so on, are explained in detail. In addition, the way to express symbolic expression is shown with several interfaces and operators which help to make a symbolic computation. To construct the algorithm of derivative, several functions, classes, and templates are introduced.

Using the Lagrange's equation and object-oriented program, the equations of motion of several representative examples are determined. The examples include models from mechanical translational systems, rotational systems, electrical systems, thermal systems, and fluid systems. The results of the computer program are verified by comparison to the analytical solution.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

DAE: Differential-Algebraic Equation. A mathematical description of a dynamic system consist of ordinary differential and algebraic equations.

ODE: Ordinary Differential Equation, a mathematical description of a system.

LAGRANGE'S METHOD: A kind of analysis method to set up the equations of motion using energy terms.

DISPLACEMENT: In a multi-domain approach to lumped-parameter systems modeling, a displacement is the term denoting position and angle in the mechanical domain, charge in the electrical domain, and volume in fluid mechanics domain.

FLOW: In a multi-domain approach to lumped-parameter systems modeling, a flow is the term denoting translational and angular velocities in the mechanical domain, currents in the electrical domain, and fluid flows in the fluid mechanics domain

NEWTON'S METHOD: An analytical method to set up the equations of motion using the free body diagram of the system.

OOP: Object-Oriented Program. such as C++, J++, Java, etc.

LDAE: A DAE resulting from the application of Lagrange's equation of motion to a system

NOMENCLATURE

δ	Variational operator	E	Energy
δq	Virtual displacement	f	Generalized flow
δW	Virtual Work	I	Generalized inductance
Γ	Vector of efforts constraints	M	Inertial matrix
κ, λ, μ	Lagrange's multiplier	P	Translational linear momentum
Υ	Right-hand side terms of LDAE	$\underline{\mathrm{P}}$	Power
ϕ	Displacement constraint equations	q	Generalized displacement
Φ	Vector of displacement constraints	Q	Heat, velocity of flow volume
ψ	Flow Constraint equations	R	Generalized resistance
Ψ	Vector of flow constraints	T	Kinetic energy
C	Generalized capacitance	T^*	Kinetic coenergy
D	Dissipation energy terms	v	Velocity of translational element
e	Generalized energy terms	V	Potential energy
e^d	Dissipation effort	V^*	Potential coenergy
e^a	Applied effort	W	Work

ACKNOWLEDGMENTS

I would like to appreciate to those who have helped me to make my thesis. Without their continuous help, I could not have finished my master course.

Especially, To my advisor, professor Brain C. Fabien, who helped me in terms of academical knowledge throughout the master course, whenever I asked something. To professor Martin C. Berg, who taught me through many classes and supported my systems and control background. To professor James Riley, who encouraged me to challenge the at first quarter in University of Washington. To professor Devasia, who is one of my committee.

To my father, who has always supported me financially and mentally to study even though he has struggled his cancer. Always, he supported me quietly even though I tried to pursue something to get impossible. I always wish that he overcomes his disease.
To my mother, who has made me comfortable during my master instead of any kind of worries. To my sister, who has gave me tips to figure out the problems related to programming. Also to my wife, who has thought my problems with me all times although she had to study her own research. To my classmates, Jong-Jin, Baek-Ho, Joe, Douglas, who has shared their knowledge to me. To my friends, who trust that I could get the master degree.

My thanks to the Department of Mechanical Engineering in University of Washington for supporting my studies.

Chapter 1

INTRODUCTION

1.1 Introduction

Nowadays, it is difficult for complex dynamic system to be analyzed numerically and symbolically. Many research institutes and industrial companies require an accurate analysis of engineering system, such as dynamic system, fluid mechanics, thermodynamics, electrical system and combined complex system. Also, symbolical results are required to use the theoretical study. This thesis presents a computer package for deriving *Lagrange's Equation* of motion for multi discipline systems.

As usual, physical system's many analysts use two methods in many methods - the *Newton method* or the *Lagrange's method* to derive the equations of motion. However, the *Newton method* is difficult to be applied to computer packages because users must finish analyzing a free body diagram completely to make input files. The *Lagrange's method* is easy to be applied to computer packages because users just input a symbolic kinetic co-energy, a potential energy, a dissipation energy, a virtual work, and a constraint equations of system. Of course, there are some algorithms to make easy input files. Also, whatever the method analysts choose; they must make an effort to reduce the minimum-order ordinary differential equation to analyze the system. In order to reduce the effort, this paper presents one suggestion as well as the reason why the *Lagrange's method* is powerful to apply computer package in chapter 2.

As the use of *Object-Oriented Programming* (OOP) increases, mechanical engineers also

need the skill of *Object-Oriented Programming* such as visual basic, C++, J++, Java, etc. Although many commercial computer packages - Matlab, Maple, Mathematica, MuPad and so on are consisted of *object-oriented programming*[1], many engineers don't know and realize the use, importance, and power of *object-oriented programming*. The matrix for *Lagrange's Equation*, is the singular matrix because the *Lagrange's equations* are associated with constraint equations in this paper, which means that we can not have an inverse matrix. For these reasons, this paper shows the simplest concept for making a symbolic object, several required functions to establish the equations of motion using the *Lagrange's method*, and a comprehensive explanation of program in chapter 3.

Finally, this thesis shows several examples - multi-degrees freedom mechanical system, electric system, fluid mechanics and thermal system for testing the computer programming. The thesis also presents the future works that is related to numerical solution using symbolical results.

1.2 Chapter Descriptions

To help the readers where there are interesting results and chapter contents, a brief description of each chapter is introduced.

Chapter 2 : To help the readers understand the concept of the fundamental elements of systems, such as displacement, flow(change rate of displacement), efforts(usually called the force), momentum, and energy terms. After that, the basic analyzing method in mechanical engineering - *Lagrange's Equation* - will be described using the basic elements of systems. Additionally, LDAE (*Lagrange's Differential Algebraic Equations*) will be introduced briefly.

Chapter 3 : The basic requirements for making the program using the *Lagrange's Equation* and the *Object-Oriented Program* are introduced and showed. This chapter can

[1] See the textbook Steeb, Hardy and Shi [33]

help the reader to understand the importance of an *Object-Oriented Program*. The reader will get the reason why many engineers want to use an *Object-Oriented Program* like visual basic, C++, J++, Java and so on.

Chapter 4 : Many examples that represent an each system, like mechanical translation system, mechanical rotational system, electrical system, thermal system, fluid system, are shown to prove the program given by this paper. In addition, the readers can follow the way to make equations of motion of some systems by hand and the program using the generalized *Lagrange's Equation* that is derived in chapter 2.

Chapter 5 : The future works associated with this paper will be described.

Chapter 2

LAGRANGE'S EQUATION OF MOTION

2.1 Introduction

There are many methods to get equations of motion for system, such as dynamics, fluid mechanics, electrical system, thermal system and so on. In basic engineering books Meriam and Kraige [25], Thomson [34], Morse, Tse, and Hinkle [10] have introduced the *Newton's method* because the *Newton's method* is based on the free body diagram, which is a very basic engineering concept in a variety of classes. However, whenever generalized coordinates are required in analyzing, many system textbooks Meirovitch [24], Crandall [8], Layton [20], Udwadia and Kalaba [35] have introduced *Lagrange's equation* for generalized coordinates. The fact that many systems need many coordinates to analyze the system, which we call the multi degree system, is the reason why engineers need generalized coordinates. Basically, it is easy and useful for engineers to analyze the system because they will use energy terms instead of using complex free body diagrams. To understand the concept of *Lagrange's method*, the basic definition and DAE's(*Differential-Algebraic Equations*), which will be used in symbolic computational programs, are presented in this chapter.

2.2 Basic Elements of System

According to Layton [20], [19], the fundamental variables of the system consist of displacements, flows, efforts, and momentum that is shown by table (2.1). The table (2.1) shows how the fundamental variables from each discipline is related.

Basically, the flow(f), which must be dependent on time, can be obtained by differentiating the displacement(q), which also must be dependent on time, in terms of time(t) such

Table 2.1: Basic Element to describe the system

System	Displacement (q)	Flow (f)	Effort (e)	Momentum (P)
Translational	Displacement	Velocity	Force	Linear Momentum
Rotational	Angle	Angular Velocity	Torque	Angular Momentum
Electrical	Charge	Current	Voltage	Flux Linkage
Fluid	Volume	Volume Flow Rate	Pressure	Pressure Momentum
Thermal	Entropy	Entropy Flow Rate	Temperature	-

that

$$\begin{aligned} f &= \frac{d}{dt}(q) \Rightarrow \begin{cases} v = \frac{dx}{dt} & : Mechanical \quad translation \\ \omega = \frac{d\theta}{dt} & : Mechanical \quad rotation \\ i = \frac{dq}{dt} & : Electrical \\ Q = \frac{dV}{dt} & : Fluid \\ \dot{s} = \frac{ds}{dt} & : Thermodynamics \end{cases} \quad or \\ q &= \int (f) dt \end{aligned} \tag{2.1}$$

From the *Newton's 2nd law*, which is called *momentum law*, the effort(e) can be expressed by the time derivative of momentum such that

$$\begin{aligned} e &= \frac{d}{dt}(P) \Rightarrow \begin{cases} F = \frac{dP}{dt} & : Mechanical \quad translation(Newton's 2nd Law) \\ T = \frac{dH}{dt} & : Mechanical \quad rotation(Euler's Equation) \\ v = \frac{d\lambda}{dt} & : Electrical(Faraday's Law) \\ P = \frac{d\Gamma}{dt} & : Fluid \end{cases} \quad or \\ P &= \int (e) dt \end{aligned} \tag{2.2}$$

And the power($\underline{P}$) can be expressed in terms of effort(e) times flow(f) such that

$$\underline{P} = e \cdot f \Rightarrow \begin{cases} \underline{P} = F \cdot v & : Mechanical \quad translation \\ \underline{P} = T \cdot \omega & : Mechanical \quad rotation \\ \underline{P} = V \cdot i & : Electrical \\ \underline{P} = P \cdot Q & : Fluid \\ \underline{P} = T \cdot \dot{s} & : Thermodynamics \end{cases} \tag{2.3}$$

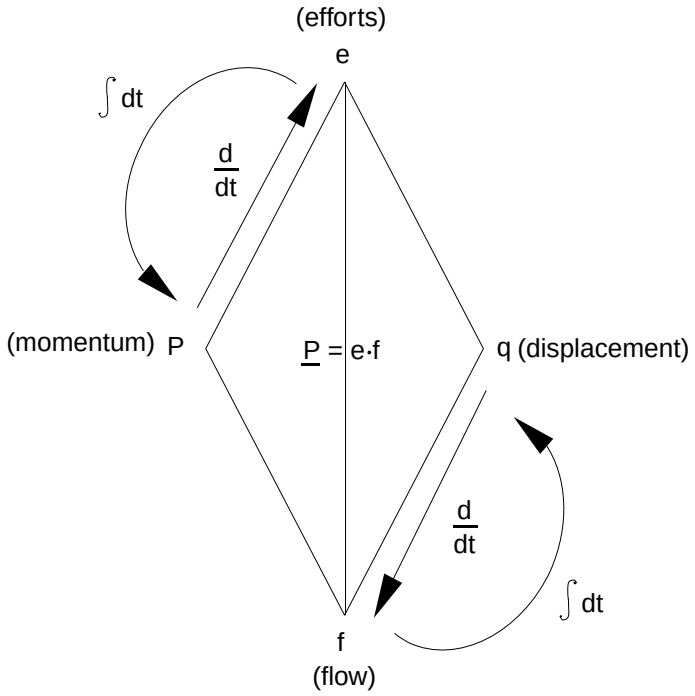

Figure 2.1: Paynter's Diagram

Figure (2.1) illustrates the relationship between the fundamental variables. Additionally, many energy terms, like kinetic energy, potential energy, dissipation energy, and so on, will be used when the *Lagrange's Equation* will be derived later. Therefore, the way to get energy terms will be introduced in this section. First of all, the most fundamental energy, the potential energy, will be expressed such that

$$P.E. = \int e_{cl}(q)dq \tag{2.4}$$

where e_{cl} : function of effort, q : displacement, P.E. : Potential Energy.

Each discipline has its own effort function, therefore using the effort function and equation

(2.4) the potential energy can be obtained for linear systems such that

$$\begin{aligned}
F_a = k\cdot x \Rightarrow P.E = \int F_a dx = \int (k\cdot x)dx = \tfrac{1}{2}k\cdot x^2 \quad &: Mechanical \quad translation \\
\tau_a = k_\theta \cdot \theta \Rightarrow P.E = \int \tau_a d\theta = \int (k_\theta \cdot \theta)d\theta = \tfrac{1}{2}k_\theta \cdot \theta^2 \quad &: Mechanical \quad rotation \\
v_{1\cdot 2} = \tfrac{q}{c} \Rightarrow P.E = \int v_{1\cdot 2} dq = \int \tfrac{q}{c} dq = \tfrac{1}{2}q^2/c \quad &: Electrical \\
P = v/C_f \Rightarrow P.E = \int P dv = \int \tfrac{v}{C_f} dv = \tfrac{1}{2}v^2/C_f \quad &: Fluid
\end{aligned} \tag{2.5}$$

In the same way, kinetic energy term, which is expressed by T, is able to be written such that

$$T = \int f dP \tag{2.6}$$

where f : Flow element, P : Momentum, T : Kinetic Energy

Each discipline has its own momentum term. Using the momentum terms and equation (2.6), the kinetic energy corresponding to each system can be obtained for linear systems

$$\begin{aligned}
P = m\cdot v \Rightarrow T = \int v dP = \int (\tfrac{P}{m})dP = \tfrac{1}{2}P^2/m \quad &: Mechanical \quad translation \\
H = I\cdot \omega \Rightarrow T = \int \omega dH = \int (\tfrac{H}{I})dH = \tfrac{1}{2}H^2/I \quad &: Mechanical \quad rotation \\
\lambda = L\cdot i \Rightarrow T = \int i d\lambda = \int (\tfrac{\lambda}{L})d\lambda = \tfrac{1}{2}\lambda^2/L \quad &: Electrical \\
\Gamma = Q\cdot I_f \Rightarrow T = \int Q d\Gamma = \int \tfrac{\Gamma}{I_f} d\Gamma = \tfrac{1}{2}\Gamma^2/I_f \quad &: Fluid
\end{aligned} \tag{2.7}$$

The last energy term, the dissipation function term, is characterized by a relationship between the effort and flow such that

$$D = \int e_{cl}(f) df \tag{2.8}$$

where e_{cl} : Function of Effort, f : Flow, D : Dissipation Function

, the function of effort can be expressed by the flow. Therefore, the dissipation function terms corresponding to each system can be obtained for linear systems as,

$$\begin{aligned}
F_{cl} = c\cdot v \Rightarrow D = \int F_{cl} dv = \int (c\cdot v)dv = \tfrac{1}{2}c\cdot v^2 \quad &: Mechanical \quad translation \\
\tau_{cl} = c_\theta \cdot \omega \Rightarrow D = \int \tau_{cl} d\omega = \int (c_\theta \cdot \omega)d\omega = \tfrac{1}{2}c_\theta \cdot \omega^2 \quad &: Mechanical \quad rotation \\
v_{cl} = R\cdot i \Rightarrow D = \int v_{cl} di = \int (R\cdot i)di = \tfrac{1}{2}R\cdot i^2 \quad &: Electrical \\
P_{cl} = R_f \cdot Q \Rightarrow D = \int P_{cl} dQ = \int (R_f \cdot Q)dQ = \tfrac{1}{2}R_f \cdot Q^2 \quad &: Fluid \\
T_{cl} = R_t \cdot \dot{s} \Rightarrow D = \int T_{cl} d\dot{s} = \int (R_t \cdot \dot{s})d\dot{s} = \tfrac{1}{2}R_t \cdot \dot{s}^2 \quad &: Thermodynamics
\end{aligned} \tag{2.9}$$

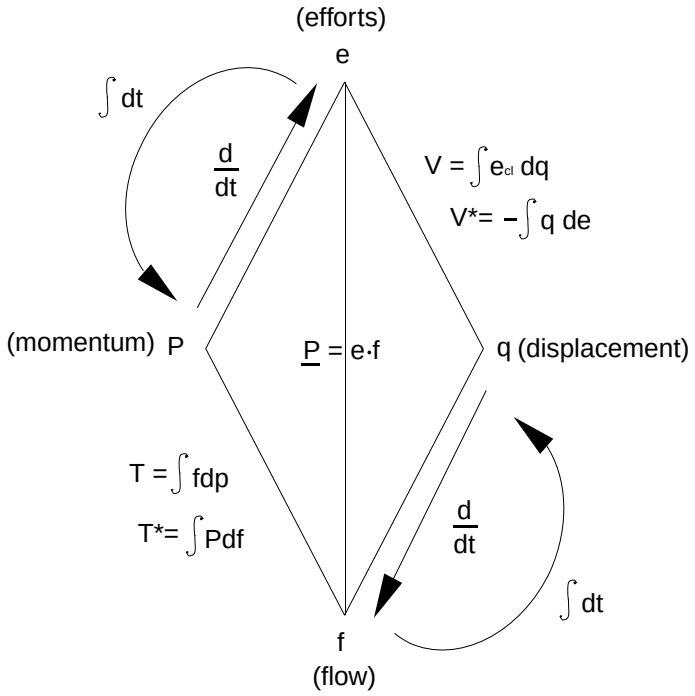

Figure 2.2: Paynter's Diagram

This section show the basic Paynter's diagram figure (2.1). If the energy terms - equations (2.4), (2.6), and (2.8) - will be added to the figure (2.1), the Paynter's diagram will be draw newly like figure (2.2).

In this section, basic elements to describe a system, relationships among elements, and the basic energy terms, have been already introduced. In next section, the *Lagrange's equation* will be derived using these concepts.

2.3 Derivation of Basic Lagrange's Equation

In an other paper [19], *d'Alembert's principle* is used to derive the *Lagrange's Equation.* Also, this paper will show how to derive the *Lagrange's equation* using the *d'Alembert's*

principle. According to the *d'Alembert's principle*, the virtual work done by the applied effort and inertial effort is zero i.e.

$$\delta W = \sum_{i=1}^{N} (e_i - \dot{P}_i) \cdot \delta q_i = 0 \tag{2.10}$$

where e_i : Applied efforts,$\dot{P}_i$: Inertial efforts

Also, the increment in energy equals the increment in work done on the system including power.

$$dE = dW \tag{2.11}$$

dE : Increment in energy, dW : Work by the applied work

The total energy is a sum of kinetic energy and potential energy. The work by the applied efforts is related to the damping force in dynamics, applied voltage in electronic system, and so on. Therefore, equation (2.11) is written such that

$$\underbrace{\delta E = \delta W, \quad \delta E = \delta(T+V)}_{\Downarrow}$$

$$\delta W = \delta(T+V) \tag{2.12}$$

The potential energy is the function of displacement and the kinetic energy is the function of momentum, displacement, and time. Therefore, the right hand side of (2.12) is written such that

$$V = V(q) \qquad \Rightarrow \qquad \delta V(q) = \sum_{i=1}^{N} \frac{\partial V}{\partial q_i} \cdot \delta q_i \tag{2.13}$$

$$T = T(P,q,t) \qquad \Rightarrow \quad \delta T(P,q,t) = \sum_{i=1}^{N} \left(\frac{\partial T}{\partial P_i} \cdot \delta P_i + \frac{\partial T}{\partial q_i} \cdot \delta q_i \right) \tag{2.14}$$

where P_i : Linear momentum in i^{th} coordinate,

q_i : Displacement, t : Time.

From equations (2.13) and (2.14). $\delta(T+V)$ can be calculated such that

$$\delta(T+V) = \sum_{i=1}^{N} \left(\underbrace{\frac{\partial V}{\partial q_i} \cdot \delta q_i}_{V_i} + \underbrace{\frac{\partial T}{\partial P_i} \cdot \delta P_i + \frac{\partial T}{\partial q_i} \cdot \delta q_i}_{T_i} \right) \tag{2.15}$$

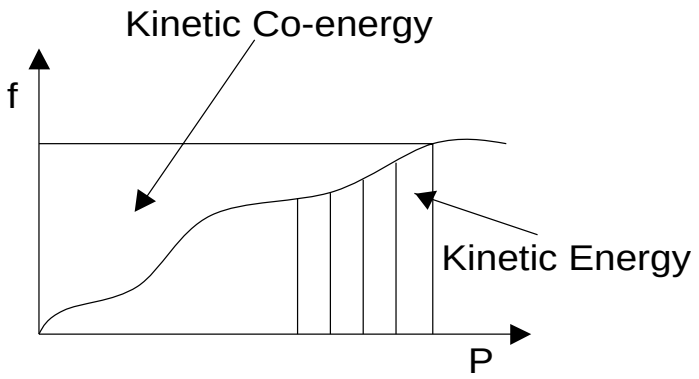

Figure 2.3: Graph of Kinetic Energy

In order to calculate the right hand side of the equation (2.15), the graph of effort versus momentum is used. This graph shows that we can calculate kinetic energy and kinetic co-energy from effort versus momentum graph. Using figure 2.3, we can set up this equation.

$$\sum P_i \cdot f_i = T + T^* \tag{2.16}$$

where P_i : Linear momentum in i^{th} coordinate,
f_i : Change rate of Flow in i^{th} coordinate $\left(f_i = \frac{d}{dt}(q_i)\right)$

Also, equation (2.16) with equation (2.14) can be written such that

$$\begin{aligned} \delta\left(\sum P_i \cdot f_i\right) &= \delta\left(T(P,q,t) + T^*(P,q,t)\right) \\ \Rightarrow \sum \left(f_i \cdot \delta P_i + P_i \cdot \delta f_i\right) &= \delta T(P,q,t) + \delta T^*(P,q,t) \end{aligned} \tag{2.17}$$

In addition, the right hand side of equation (2.17) with equation (2.14) can be expressed such that

$$\delta T + \delta T^* = \sum \left(\frac{\partial T}{\partial P_i} \cdot \delta P_i + \frac{\partial T}{\partial q_i} \cdot \delta q_i + \frac{\partial T^*}{\partial P_i} \cdot \delta P_i + \frac{\partial T^*}{\partial q_i} \cdot \delta q_i\right) \tag{2.18}$$

When rearranging equation (2.17) and (2.18), we can get the following equation.

$$\sum \left(\frac{\partial T}{\partial P_i} - f_i\right) \cdot \delta P_i + \sum \left(\frac{\partial T^*}{\partial f_i} - P_i\right) \cdot \delta f_i + \sum \left(\frac{\partial T}{\partial q_i} + \frac{\partial T^*}{\partial q_i}\right) \cdot \delta q_i = 0 \tag{2.19}$$

If δP_i, δf_i, δq_i are independent on each other, we can get following results from equation (2.19)

$$f_i = \frac{\partial T}{\partial P_i} \tag{2.20}$$

$$P_i = \frac{\partial T^*}{\partial f_i} \tag{2.21}$$

$$\frac{\partial T}{\partial q_i} = -\frac{\partial T^*}{\partial q_i} \tag{2.22}$$

According to Layton's [19], [20] and section 2.1, the increment in work equals the displacement times the effort that can be expressed by the time derivative of linear momentum, i.e. the increment in work can be written by such that

$$e_i \cdot dq_i = \frac{dP_i}{dt} \cdot dq_i = dP_i \cdot \frac{dq_i}{dt} = f_i \cdot dP_i$$

where e_i : efforts, P_i : Linear Momentum,

q_i : Displacement, f_i : Change rate of displacement

Consequently, the increment in work can be expressed by such that

$$e_i \cdot dq_i = \dot{P}_i \cdot dq_i = f_i \cdot dP_i \tag{2.23}$$

The increment in work can be gotten using the equation (2.20) and (2.21).

$$e_i \cdot \delta P_i = \begin{cases} f_i \cdot \delta P_i = \frac{\partial T}{\partial P_i} \cdot \delta P_i \\ \dot{P}_i \cdot \delta q_i = \frac{d}{dt}\left(P_i\right) \cdot \delta q_i = \frac{d}{dt}\frac{\partial T^*}{\partial f_i} \cdot \delta q_i \end{cases} \tag{2.24}$$

From the previous two equations (2.24), we can get the following equation.

$$\frac{\partial T}{\partial P_i} \cdot \delta P_i = \frac{d}{dt}\frac{\partial T^*}{\partial f_i} \cdot \delta q_i = \frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} \cdot \delta q_i \tag{2.25}$$

Equations (2.13) with (2.14) into which is plugged by (2.22) and (2.25) can be written by increment of total energy, like

$$\delta T = \sum_{i=1}^{n}\left(\frac{\partial T}{\partial P_i} \cdot \delta P_i + \frac{\partial T}{\partial q_i} \cdot \delta q_i\right) \tag{2.26}$$

$$= \sum_{i=1}^{n}\left(\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} \cdot \delta q_i - \frac{\partial T^*}{\partial q_i} \cdot \delta q_i\right) \tag{2.27}$$

with eq. (2.13)

$$\Rightarrow \delta(T+V) = \sum_{i=1}^{n} \Big(\underbrace{\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i}}_{T_i} + \underbrace{\frac{\partial V}{\partial q_i}}_{V_i} \Big) \cdot \delta q_i \tag{2.28}$$

In any system, work by the applied effort, as you read earlier, such as damping force, applied voltage, constraint force, must be considered to completely set up the equation of motion. That can be expressed by such that

$$\delta W = \sum_{i=1}^{n} \left(e_i^d + e_i^a \right) \cdot \delta q_i$$

where e_i^d : Dissipational efforts, e_i^a : All other efforts

Damping efforts can be expressed by $e_i^d = -\frac{\partial D}{\partial f_i}$. Therefore,

$$\delta W = \sum_{i=1}^{n} \left(-\frac{\partial D}{\partial f_i} + e_i^a \right) = \left(-\frac{\partial D}{\partial \dot{q}_i} + e_i^a \right) \tag{2.29}$$

Finally, using the equation (2.28) and (2.29) and $\delta E = \delta W$, the "*Lagrange's Equation*" can be derived.

$$\sum_{i=1}^{n} \Big(\underbrace{\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i}}_{\delta(T+V)} + \underbrace{\frac{\partial D}{\partial \dot{q}_i} - e_i^a}_{-\delta W} \Big) \cdot \delta q_i = 0 \tag{2.30}$$

If all the coordinates are independent each other, the *Lagrange's Equation* (2.30) is expressed

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = e_i^a \qquad i = 1, 2, \ldots, n \tag{2.31}$$

where n is the number of coordinates

Up to now, the basic *Lagrange's Equation* is derived for helping to understand a fundamental knowledge.

2.4 Lagrange's Equation with Lagrange's Multiplier

To easily get equations of motion, the constraint equations will be used. In most system, energy functions and coordinates are connected by the constraint equations because the most coordinates are independent on each other. In that case, *Lagrange's multiplier* must be used. Let us consider *Lagrange's Multiplier*.

Theorem 1 (Lagrange's Multiplier) *Given a matrix $A \in \Re^{m\times n}$, let $\vec{s} \in \Re^n$ be a vector that satisfies the relation $A \cdot \vec{s} = 0$. Also for same vector $\vec{s}$, if it is true that $\vec{s}^T \cdot B = 0$, then there exists a vector $\lambda \in \Re^n$ such that*

$$\vec{s}^T \cdot \left(B + A^T\lambda\right) = 0, \quad \forall \vec{s} \tag{2.32}$$

The vector λ is called the Lagrange's Multiplier.

When applied the above *lagrange's multiplier theorem*, i,e, the displacement is plugged into vector $\vec{s}$, the basic *lagrange's equation* (2.31) is plugged into matrix B, the constraint relationship forms are plugged into matrix A, basic *Lagrange's Equation* (2.31) must be changed. Let us derive that. Let us set the ε such that

$$\varepsilon_i = \frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} - e_i^a \tag{2.33}$$

With equation (2.12), equation (2.33) can be written such that

$$\begin{aligned}\delta(T+V) - \delta W &= \sum_{i=1}^{n} (\varepsilon_i \cdot \delta q_i) \\ &= \varepsilon_1 \cdot \delta q_1 + \cdots + \varepsilon_n \cdot \delta q_n = 0\end{aligned} \tag{2.34}$$

If any coordinates are independent on each other and the change of displacement is arbitrary, then the equation (2.34) obviously is satisfied if and only if

$$\varepsilon_i = 0, \quad i = 1, 2, \cdots, n \tag{2.35}$$

However, if the coordinates are not independent, there must exist the form of contraint relationships among the coordinates like

$$\phi_j = \phi_j(q_i, t) = 0 \tag{2.36}$$

where q_i : Displacement in i^{th} coordinate, j = 1,2,$\cdots, m_1$

(m_1 is the number of displacement contraint equations)

Therefore, these constraint equations must make each *Lagrange's Equation* dependent respectively. When considered, a variation of above constraint equations can be expressed in

terms of matrix like

$$\begin{array}{rcccccccc}
\delta\phi_1 &=& \frac{\partial\phi_1}{\partial q_1}\cdot\delta q_1 &+& \frac{\partial\phi_1}{\partial q_2}\cdot\delta q_2 &+& \cdots &+& \frac{\partial\phi_1}{\partial q_n}\cdot\delta q_n = 0 \\
\delta\phi_2 &=& \frac{\partial\phi_2}{\partial q_1}\cdot\delta q_1 &+& \frac{\partial\phi_2}{\partial q_2}\cdot\delta q_2 &+& \cdots &+& \frac{\partial\phi_2}{\partial q_n}\cdot\delta q_n = 0 \\
\vdots & & \vdots & & \vdots & & & & \vdots \\
\delta\phi_m &=& \frac{\partial\phi_m}{\partial q_1}\cdot\delta q_1 &+& \frac{\partial\phi_m}{\partial q_2}\cdot\delta q_2 &+& \cdots &+& \frac{\partial\phi_m}{\partial q_n}\cdot\delta q_n = 0
\end{array} \tag{2.37}$$

The equation (2.37) can be expressed by matrix *Hessian* form.

$$\begin{aligned}
\underbrace{\phi_{q_i}}_{m\times n}\cdot\underbrace{\delta q_i}_{n\times 1} &= \begin{bmatrix} \frac{\partial\phi_1}{\partial q_1} & \frac{\partial\phi_1}{\partial q_2} & \cdots & \frac{\partial\phi_1}{\partial q_n} \\ \frac{\partial\phi_2}{\partial q_1} & \frac{\partial\phi_2}{\partial q_2} & \cdots & \frac{\partial\phi_2}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\phi_m}{\partial q_1} & \frac{\partial\phi_m}{\partial q_2} & \cdots & \frac{\partial\phi_m}{\partial q_n} \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \vdots \\ \delta q_n \end{bmatrix} \\
&= \sum_{j=1}^{m}\frac{\partial\phi_j}{\partial q_i}\cdot\delta q_i = 0
\end{aligned} \tag{2.38}$$

From equation (2.32), (2.33), and (2.38) with the condition equation (2.34), the equation (2.31) can be expressed in terms of independent coordinates such as

$$\frac{d}{dt}\frac{\partial T^*}{\partial\dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial\dot{q}_i} + \sum_{j=1}^{m_1}\frac{\partial\phi_j}{\partial q_i}\cdot\lambda_j - e_i^a = 0 \tag{2.39}$$

where ϕ_j is the function of displacement constraint($\phi_j(q_i,t)=0$),

λ's are the lagrange's multiplier, i = 1,2,$\cdots$, n and j =1,2,$\cdots$,m_1

In the same way, the flow constraint equations also can be included to the equation (2.39), using the *Lagrange's Multiplier Theorem* again. In that case, the *Flow Constraint* equation must be considered to get the *Lagrange's Equation* with flow constraint equations. To get that, let the flow constraint equations such as

$$\psi_j = \psi_j(\dot{q}_i, q_i, t) = 0 \tag{2.40}$$

where $\dot{q}_i$: Change rate of displacement,

j = 1,2,$\cdots$,m_2 (m_2 is the number of flow constraint equation)

Set the equation (2.39), which is the left hand side, ε_i^*. i.e, again use the *Lagrange's Multiplier* same way like displacement constraint equation. When derived through same way,

Lagrange's Equation can be written such

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} + \sum_{j=1}^{m_1} \frac{\partial \phi_j}{\partial q_i} \cdot \lambda_j + \sum_{j=1}^{m_2} \frac{\partial \psi_j}{\partial \dot{q}_i} \cdot \kappa_j = e_i^a \tag{2.41}$$

where ϕ_j is the function of displacement constraint($\phi_j = 0$, j=1,2, $\cdots$, m_1),

ψ_j is the function of flow constraint($\psi_j = 0$, j=1,2, $\cdots, m_1$),

λ_j's and κ_j's are the lagrange's multiplier, and i = 1,2,$\cdots$,n

In addition, the efforts constraint equations $\Gamma_j = \Gamma_j(\dot{q}, q, e_k, t) = 0$ also can be considered such as

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} + \sum_{j=1}^{m_1} \frac{\partial \phi_j}{\partial q_i} \cdot \lambda_j + \sum_{j=1}^{m_2} \frac{\partial \psi_j}{\partial \dot{q}_i} \cdot \kappa_j = e_i^a(e_k) \tag{2.42}$$

where ϕ_j is the function of displacement constraint(ϕ_j=0, j = 1,2,$\cdots$, m_1),

ψ_j is the function of flow constraint(ψ_j=0, j = 1,2,$\cdots$, m_2),

Γ_j is the function of efforts constraint(Γ_j=0, j = 1,2,$\cdots$, m_3),

λ_j's and κ_j's are the lagrange's multiplier, and i = 1,2,$\cdots$, n

Equation (2.42) is called the *general form of Lagrange's equation* which is a basic concept of this paper because it is very easy to set up the equation of motion symbolically.

2.5 Advantage of Lagrange's Equation in Programming

Up to section 2.3, the general *Lagrange's Equation* has been derived using the *d'Alambert's princple* and the concept of *Lagrange's Multiplier*. When considered, equation (2.42) consists in terms of multiplier and energy expressions, such as kinetic co-energy, potential energy, dissipation function, all other energy, and lagrange's multipliers.

If analysts want to set up equations of motion in any multi-degree combined system using the *Lagrange's Equation*, first of all, they must symbolically calculate energy terms. Calculating multipliers without computer is very difficult because displacement, flow, and effort constraint equations are very complex connections among the coordinates generally.

After calculating energy terms symbolically, analysts differentiate the each term for applying the *Lagrange's Equation*. With the same way, the algorithm of given program in this paper is executed.

In addition, if the *Newton's Method*, which is using the free body diagram, is applied, when an electrical system is analyzed, the additional concepts are required. However, the *Lagrange's Equation* does not required.

The given program, in this paper, uses the input file, which just consists of coordinates, constants values, and energy terms, for users to use easily. i.e, many complicated calculi are executed in given program in this paper with the symbolic computation algorithm. This paper will show the symbolic computational concepts in chapter 3.

In this section, general *Lagrange's Equation* is modified in terms of matrix form[1] to easily make input file, which use the equation (2.42), such as

$$\dot{q} = f, \tag{2.43}$$

$$M\dot{f} + \phi_q^T \lambda + \psi_f^T \kappa = \Upsilon, \tag{2.44}$$

$$\phi = 0, \tag{2.45}$$

$$\psi = 0, \tag{2.46}$$

$$\Gamma = 0 \tag{2.47}$$

In odrder to make equation (2.42) matrix form, let Υ, $\dot{p}$, M be defined such as

$$\begin{aligned} M &= \partial^2 T^* / \partial^2 f, \quad \phi_q = \partial\phi/\partial q, \quad \psi_f = \partial\psi/\partial f, \\ \Upsilon &= e_i^a - \frac{\partial T^*}{\partial f} \cdot f - \frac{\partial T^*}{\partial q} + \frac{\partial T^*}{\partial q} - \frac{\partial V}{\partial q} - \frac{\partial D}{\partial f} \end{aligned}$$

Above simple modification helps to get a numerical results[2]. Basically, equations from (2.43) to (2.47) will be used in program in future wowk.

---

[1] See Fabien and Layton's paper [16], [18], [17], and Layton's dissertation [19]

[2] This topic is explained in chapter 5.

In chapter 4. several examples for *Lagrange's Equation*, translation dynamics, rotational dynamics, electrical system, compressive fluid system, and thermal system, are shown with result of program.

Chapter 3

SYMBOLIC SOLUTION FOR LAGRANGE'S EQUATION

3.1 Introduction

In the previous chapter, the *generalized form of Lagrange's Equation* has been derived by using the *d'Almebert Principle* and *Lagrange's Multiplier theorem*. The necessary condition for using the *Lagrange's Equation* in program is introduced in this chapter. In addition, the reason why an *Object-Oriented Program* is powerful in engineering field is introduced concentrating on the brief concept of an *Object-Oriented Program*[1].

In basic symbolic computational programs, many conditions, such as a recognition of symbol, a differentiation which includes the chain rule, a symbolically calculus, and so on are needed to get correct computations. In other many papers Rall [28], [29], and Davenport, Siret, & Tournier's textbook [9], the way to differentiate a symbol corresponding to what users want, has been introduced. In this paper, some important concepts in computational program are introduced focusing on the C++ because given program is used to the C++.

Many demands to achieve advanced technology spur the development in terms of program algorithm. The algorithm of given program will be reported in this chapter with brief explanation[2].

---

[1] Object-Oriented Program be called OOP sometimes.

[2] see the appendix C.

3.2 Concepts

3.2.1 Basic Concept

In oder to get the correct result of *Lagrange's Equation* using the computer, it must have some mandatory conditions such that

- Recognition of Symbol [3] $\leftrightarrow$ Product of the Object
- Symbolic Calculus [4] $\leftrightarrow$ Over Loading in computer language
- Differentiation of Symbol $\leftrightarrow$ For Calculation of *Lagrange's Equation*

The program require a program skill[5] corresponding to the each conditions as you can see above conditions. The basic concept of *Object-Oriented Program* have to be introduced in this section briefly for being satisfied with above mandatories.

The basic concept of *Object-Oriented Program* can be summarized by such items

- Possibility of making our own object
  - By defining the class
  - By our demand to get our goal
- Possibility of using the origin basic function
  - By overloading the function
  - Unnecessary to define again

---

[3] The most parts are following the header file of Hardy, Shi, and Steeb [33].

[4] The basic concepts are following Hardy, Shi, and Steeb [33]

[5] Basic function of C++ are explained in Herbert textbook [31].

Because of these big advantages, engineers keep developing the *Object-Oriented Program.* i.e, it is very convenient for engineers to attach or develop the algorithm to continue former researchers. If these concepts are used, the symbol can be realized, which means the fact that the object of symbol can be created by the *class.* Simply speaking, the most function in *Object-Oriented Programming* should be thought as a kind of object which made by program complier's creaters.

For example, in math header file, the sin wave function is an object. Therefore, the user do not have to define the sin function for using his own program. Otherwise, user can define the overloading to use easily an object when the object's symbol are same as he define. For an example, to calculate the symbols, the overloading of '+'. '-', '×', and '÷'[6] is possible. '+'. '-', '×', and '÷' are defined for calculating integers, doubles and so on. Many overloading examples can be shown in textbooks such as Prata [27], and Campione and Walrath [7].

Nowadays, many *Object-Oriented Programs*, such as visual C++, J++, JAVA, and so on, are made to be satisfied with these conditions. Most commercial packages such as Matlab[7], Maple[8], Mathmetica, MathCad, MuPad and so on, are made by an *Object-Oriented Program.* i.e, users are able to make their own object product in those packages.

3.2.2 Algorithm for Symbol

Some papers, Rall [28], [29],and Jerell [23], have introduced an algorithm with *Object-Oriented Program.* Additionally, the textbook of Hardy, Shi, and Steeb [33] has good examples to realize the above conditions. According to the Hardy, Shi, and Steeb [33], the algorithm for basic four kinds of calculus(+, -, ×, ÷) will be explained in this section.

---

[6] All calculi will be explained another section detail.

[7] See the manual [21].

[8] See the manual [36].

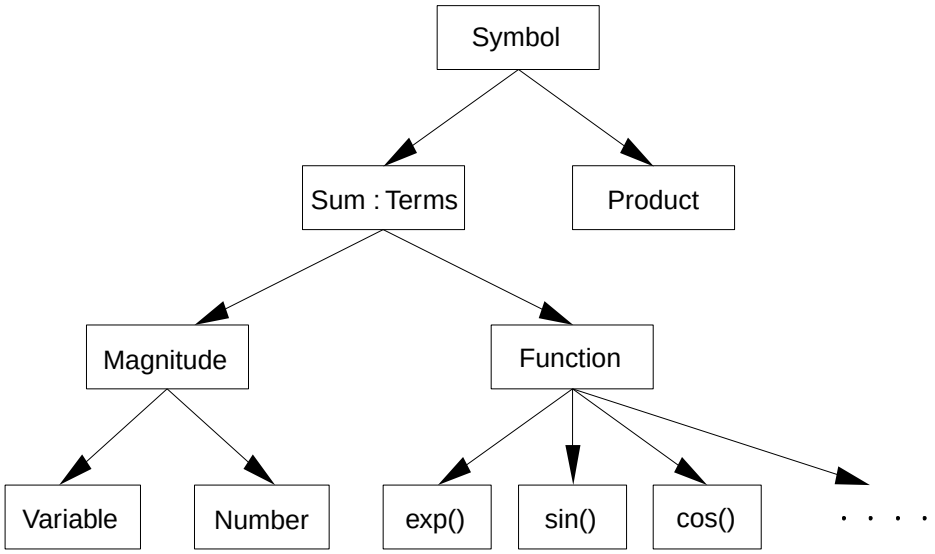

Figure 3.1: Schematic diagram for symbol class

Basically, the purpose of this paper is that symbols, such as x, y, and so on, can be calculated independently. In term of programming, four kinds of calculus must be overloaded because the complier, in this paper C++, has already had them.

The most important things are to make the object of symbols and four kinds of calculus can be realized to be calculated in terms of symbol. The figure 3.1 shows the schematic diagram to make the object of the symbol. Briefly speaking, the class of Symbol has two classes, the Sum and the Product. Two classes, therefore. have inherited the characteristics of the Symbol which makes the symbolic quantities. The Sum class defines symbolic variables and a number. In addition, the Sum class distinguishes the functions, interfaces, and operators. On the other hand, the Product class describes the $\times$ and $\div$. Easily speaking, for this thesis, the Sum class keeps a pointer to class Terms which is shown in figure (3.1). The Terms class has two classes - Magnitude and Function. As you can see in the figure (3.1), the Magnitude class is divided by two classes - Variable and Number. And the Function class is divided by several functions.

The Variable class is associated with the symbolic variable and its own numerical value. The Variable, therefore, needs two data - the name and the value of the variable. There are two storages to save the name and the value of the variable, such as varName() which store the name of symbol and val() which save the numerical value. Internally, those storages are moving for keeping their own characteristics.

The Number class defines a template for numerical data types. The Number class, also, has varName() which can store a *Null* because that is meaningfulness in this class, and val() which store the numerical value. The operator "=" makes Number assign.

The Function class declares following functions.

$$\exp(x),\ \sin(x),\ \cos(x),\ \sinh(x),\ \cosh(x),\ \text{and sqrt (x)}$$

Table 3.1: Return character corresponding to each function

Function	Return character	Function	Return character
exp(x)	e	sqrt(x)	q
ln(x)	l	dot	d
sin(x)	z	cos(x)	c
sinh(x)	j	cosh(x)	w

In Function class, above functions must be distinguished with the other symbol. Any special storage to contain the its function name is required. For an example, the interface of type() returns a character 'e' to recognize that it is an exponential function. The return characters are used in just interface for keeping the name of function. Therefore, an each function must have its own return character for being distinguished in program. The table (3.1) shows the return character[9] corresponding to each function[10]. Because C++ complier

[9] The most return characters are copied in [33]

[10] sqrt(x) = $\sqrt{x}$, ln ⇒ log_e, dot ⇒ differentiation

Table 3.2: Interfaces declared by each class

Class	Interfaces	Functions
Terms	Type()	returns the type - Variable, Number or function
	varName()	returns the name of the Variable or the function
	oprint(ostream& os)	put the name of a Variable and function or the numerical value of a Number on output stream os
Magnitude	val()	returns the numerical value of a Variable or a Number
	set(const T Num)	allots the numerical value num to a Variable or a Number
Function	f(const T& a)	returns the numerical value of the function evaluated at a

is one of *Object-Oriented Program*, everybody can attach and modify new function into the the Fuction class, which is an advantage of *Object-Oriented Program*. For an example, if an integration is required, everybody can define a new return character in the Function and describe the characteristics of integration in other class.

These classes have the interfaces such as type(), varName(), and oprint(ostream& os), val(), set(const T num), and f(const T& a)[11]. These interfaces save and send the data. The table (3.2) shows briefly the function of each interfaces and the class which declares itselves. These classes and interfaces can make any kind of symbol expression.

Let us consider examples for making the expression - the figure (3.2)[12] shows the schmetic diagram of making the symbolic expression.

Think about the expression

$$z = a \times b + 1 + \sin(x)$$

and the expression tree in the figure (3.2) (a). Another example is in the figure (3.2) (b).

$$y = 3 \times \sin(x+a) + 8 \times x\hat{}\,2 \times a\hat{}\,4 - 5 + 6 \times \exp(3 \times x + 2 \times b)$$

[11] follow the < math.h > library

[12] The concept of diagram is from the steeb, hardy and shi [33].

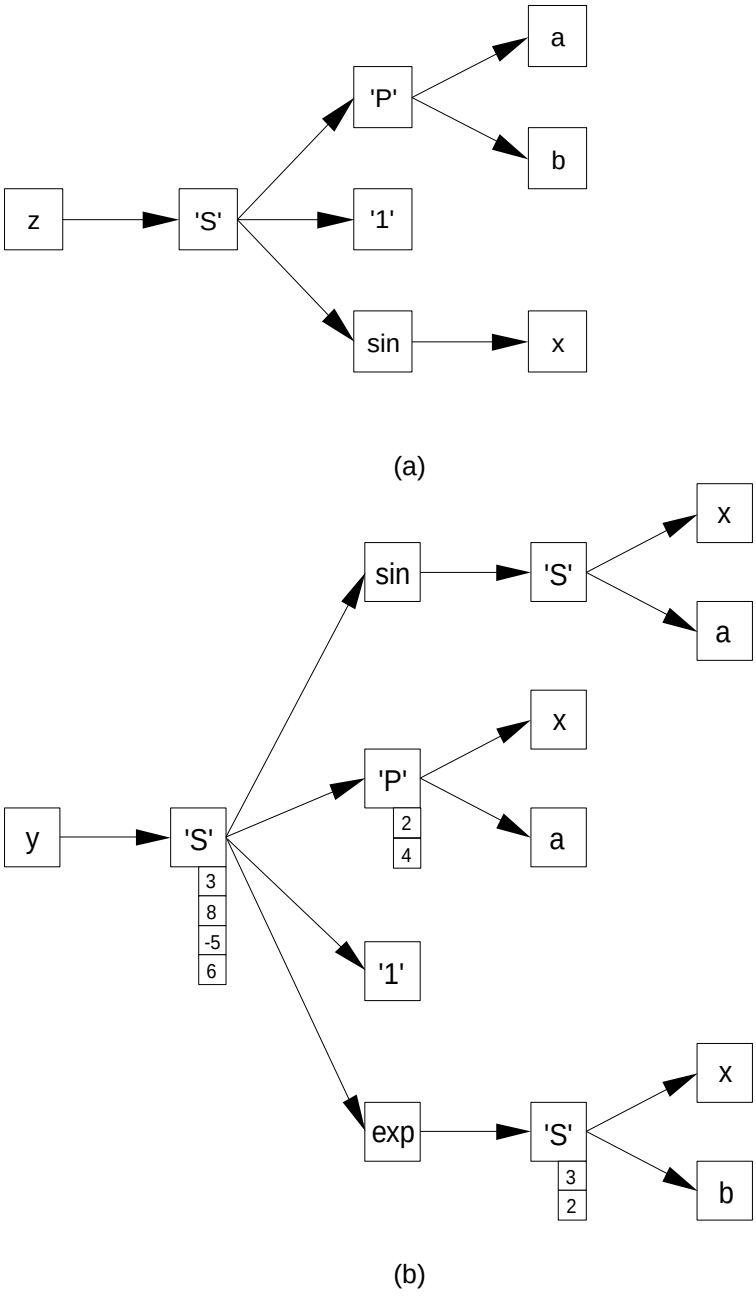

Figure 3.2: Schematic diagram for creating symbol expression (a) z = a×b+1+sin(x), (b) y = 3×sin(x+a)+8×xˆ2×aˆ4-5+6×exp(3×x+2×b)

'S' means the root of an expression tree(in second column) or subtree(in fourth column). The boxes written x, y, z, a, and b, are the Variable node which has two storages - for saving the name and the value of variable. 'P' is a Product which expresses $\times$. The smaller boxes, which is written numbers under 'S' and 'P' boxes, mean a multiplication factor. '1' is just number. sin(x) and exp(x) functions are defined in Function class.

3.3 Algorithm for Operators

Table 3.3: Simple explanation for operators

Expression	Operation
x+y, x+c, c+x	add symbolic variables or numerical constants
x-y, x-c, c-x	subtract symbolic variables or numerical constants
x*y, x*c, c*x	multiply symbolic variables or numerical constants
x/y, x/c, c/x	divide symbolic variables or numerical constants
x=y	assign a symbolic variable or expression
x=c	assign a numerical constant
==, !=	bool operator, compare the equality of two expressions and return 1(true) or 0(false)
power(Sum< T > &s, int n)	raises expression s to power n
dot(Sum< T >& y, Sum< T >& x)	differentiate y with respect to x
coeff()	get the symbolic or numeric coefficients
put(Sum< T >& s1, Sum< T >& s2)	replace s2 for s1
Sum< T >& y.set(const T num)	assigns the numeric constant num to the symbol x
Sum< T >& y.value()	returns the evaluated value of expression y
Sum< T >& y.depend(Sum< T >& x)	y is dependent on x

In this section, mathematical operators, such as +, -, $\times$, $\div$, power(), dot(), coeff(), depend() and so on have been introduced. First of all, let us think operators which are required in a symbolic computational program. If x and y are symbolic variables and c is a

numerical constant, required operators will do following the table (3.3).

3.3.1 Differentiation

The table (3.3) is very helpful as basic operators. Now, let us consider detail functions of each unknown operator, like dot(), coeff(), put(), depend(), and so on. When thinking about a symbolic computation program related to the *Lagrange's Equation*, developer requires the derivative function because the function of derivative is a basic function to get each term of *Lagrange's Equation*. To make the differentiation function, several requirements exist such that

1. $\frac{dx}{dx} = 1$, $\frac{dc}{dx} = 0$, where c is a numerical constant

   Solution : Check the return value[13] of type()

2. $\frac{dy}{dx} = \begin{cases} \frac{dy}{dx} & : \text{y is dependent on x} \\ 0 & : \text{y is independent on x} \end{cases}$

   Solution 1 : Use the function depend()

   Solution 2 : Check the return value of type() when y is independent on x

3. Linear operator $\Rightarrow \frac{d}{dx}[a * u(x) + b * v(x)] = a * \frac{du(x)}{dx} + b * \frac{dv(x)}{dx}$, where a and b are numerical constants, x is an independent symbol, and u(x) and v(x) are functions of x

4. Chain rule $\Rightarrow \frac{d}{dx}f(u(x)) = \frac{df}{du} * \frac{du}{dx}$ where f is a function of u that is dependent on x

   Solution : Differentiate the function f for the function u. After that, multiply with differentiating the function u for x.

The figure (3.3) shows detail execution of the chain rule which is explained very well in Apostol [2]. If f is a function of u which is dependent on x, a derivative for variable x can be

[13] In this case, the return value are 'V' or 'N'.

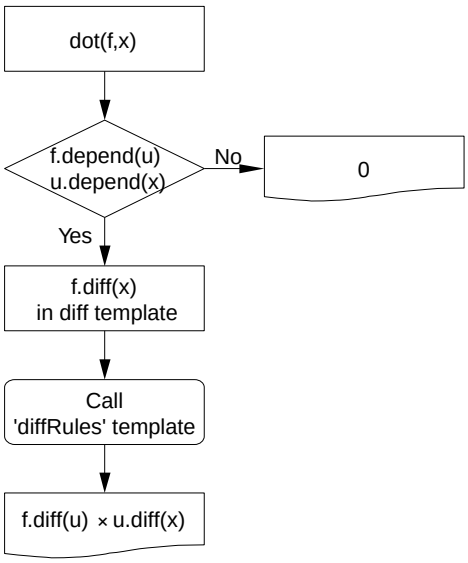

Figure 3.3: A flowchart for explaining chain rule

calculated like the figure (3.3). "dot(f,x)" call the dot template which returns information of f and x to a diff() template. In the diff() template, a test of dependency, which the decision block means is executed. For example, y.depend(x); y.depend(z) declares that y is dependent on x and z. The several operators are related to the depend() - nodepend()[14], isdepend()[15].. After that, call a diffRules template to compute completely the chain rule. In diffRules, derivatives of all special functions, such as sin, cos, sinh, cosh, ln, exp, sqrt and so on, are defined because these functions have specific derivative results. In the same way with the Function class, users can define new derivatives of new function, like gamma function, square wave, impulse function and so on, in the diffRules template.

[14] opposite meaning of depend().

[15] A kind of bool operator

3.3.2 Others

In the table (3.3), several important operators, such as put(), set(), value() and so on, are shown. These operators are explained in this section.
The put(s1,s2) operator replace s2 for s1. This operator is very useful when the *Lagrange's Equation* is calculated. As many derivatives are executed, many other symbols are created. For an example, 'x' is a displacement coordinate and 'dotx' is a flow coordinate. Between two symbols, there is the relationship, like $\frac{dx}{dt} = dotx$ which can be rewritten in terms of operators such that dot(x,t) = dotx. Without put(), dot(x,t) and dotx should be used in a result, which is a little bit messy. Therefore, if the put() operator is used to calculated, the result will be clear than before used the put() operator[16].

Suppose x, and y are symbolic variables, c is a numerical value, y is an algebraic expression in terms of x. The x.set(c) operator assigns the numerical value c in the symbolic variable y. This operator is used when any symbolic variables have specific numerical value[17]. However, set() operator just assigns the numerical value. Other operator, therefore, is needed to get the evaluated algebraic value of expression y. For an example, when users want to get the numerical result instead of sybolic, they just make an input file including the value of constant variable[18]. The y.value() operator evaluates algebraic value of expression y with numerical value x. The x.set(c) always proceeds the y.value().

The y.coeff(x) operator returns the symbolic or numerical x coefficients of the expression y. The y.depend(x) has already been introduced - the decision of dependency[19]. This operator also has to be used for deciding that all coordinates and all energy terms are depend on time. The power(x,c) means x^c simply.

---

[16] See appendix C. Codes

[17] See chapter 4.

[18] See appendix A.

[19] can find this operator in appendix C.

3.4 Miscellaneousness

Until now, the requirements and the skill of symbolic computation are explained. The purpose of this paper is to get equations of motion of system. Let us consider more detail requirements for developing algorithm.

Again, let us consider some conditions for getting equations of motion using symbolic computation program.

1. Decision of a kind of input file
   - DOS window input
   - Text file input[20]
   - Window input
2. Input file[21] consists of
   - Constant
   - Coordinates
   - Energy terms (kinetic energy, potential energy, and dissipation function)
   - Work
   - Constraint equation
3. According to the input file, create symbol objects
4. Decide a dependency for preparing differentiations
5. According to energy terms, differentiate
6. Using the put() operator, simplify an expression

[20] This input type will be used.

[21] See appendix A for more detail information.

7. Assign numerical value to symbols which have specific value

8. Get a result

Chapter 4

EXAMPLES

4.1 Introduction

In this chapter, several representative examples are introduced with completed equations of motion by hands and program which is made by the *Object-Oriented Program* C++. Therefore, the result of program can be compared with that of by hands.

Briefly speaking, the examples include mechanical translational motion, mechanical rotational motion, electrical motion, thermal system, compressive fluid system and combined system. In addition, the method of making input files is explained with first two examples in detail[1].

4.2 Mechanical System - Mechanical Translation

In many dynamics books, translational motion systems are introduced such that spring, damping, external force, and so on. In this section, a spring system with a pendulum will be analyzed by hand. After that, the results are going to be compared to the result using the program.

The figure (4.1) shows briefly sketch for analyzing, i.e, the figure (4.1) can help to draw the *free body diagram*. However, this paper just uses the *Lagrange's Equation*. Therefore, equations of motion of this system are derived using the *Lagrange's Equation*.

---

[1]See Appendix A

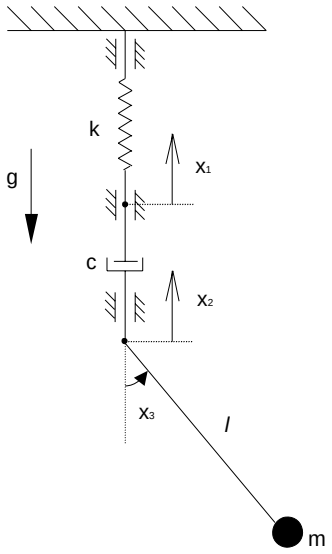

Figure 4.1: Mechanical Translation System - Spring with Pendulum

First of all, the independent coordinates of system must be chosen by inspection such that

$$Coordinates \quad : x_1, x_2, x_3$$

Now, let one of *Lagrange's equation* (2.31). (2.39), (2.41), and (2.42), apply for system of figure (4.1). The equation (2.31) can be applied to get equations of motion because, in this system, the coordinates are independent on each other.
Recall the equation (2.31),

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} = e_i^a \qquad i = 1, 2, \ldots, n \tag{4.1}$$

where e_i^a is applied work corresponding to i^{th} coordinate.

To apply the equation (4.1), the kinetic co-energy, potential energy, dissipation function, and virtual work are expressed in terms of coordinates and some constants such that

$$T^* \quad = \quad \frac{1}{2} m v_{c.o.m}^2 \tag{4.2}$$

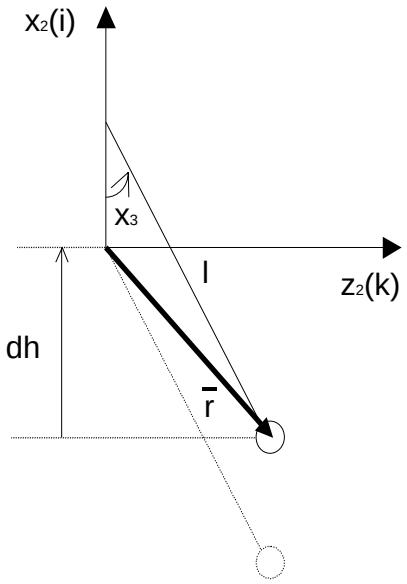

Figure 4.2: Position of Ball in Pendulum

$$V = \frac{1}{2}kx_1^2 \tag{4.3}$$

$$D = \frac{1}{2}c\,(\dot{x}_1 - \dot{x}_2)^2 \tag{4.4}$$

$$\delta W = -mg(\delta h) \tag{4.5}$$

To get the velocity of center of mass ($v_{c.o.m}$) and the change of height (δh), let's draw the position of the ball in pendulum like the Figure (4.2). According to the figure (4.2), the velocity of center of mass and the change of height are calculated by following equations.

$$\widetilde{r} = (x_2 - l\cdot\sin(x_3))\cdot\overline{i} + l\cdot\sin(x_3)\cdot\overline{k} \quad \Rightarrow$$

$$\widetilde{v} = \frac{d\widetilde{r}}{dt} = (\dot{x}_2 + l\cdot\sin(x_3)\cdot\dot{x}_3)\cdot\overline{i} + (l\cdot\cos(x_3))\cdot\dot{x}_3\cdot\overline{k} \tag{4.6}$$

$$h = x_2 - l\cdot\cos(x_3) \Rightarrow \delta h = \delta x_2 + l\cdot\sin(x_3)\delta x_3 \tag{4.7}$$

Using the equation (4.6), the kinetic co-energy term can be expressed with equation (4.2) such that

$$\widetilde{v}_{c.o.m}^2 = \widetilde{v}\cdot\widetilde{v} = (\dot{x}_2 + l\cdot\sin(x_3)\cdot\dot{x}_3)^2 + (l\cdot\cos(x_3)\cdot\dot{x}_3)^2$$

$$
\begin{aligned}
&= \dot{x}_2^2 + l^2 \cdot \sin^2(x_3)\dot{x}_3^2 + 2 \cdot \dot{x}_2 \cdot l \cdot \dot{x}_3 \cdot \sin(x_3) + l^2 \cdot \cos^2(x_3) \cdot \dot{x}_3^2 \\
&= \dot{x}_2^2 + l^2 \cdot \dot{x}_3^2 + 2 \cdot \dot{x}_2 \cdot l \cdot \dot{x}_3 \cdot \sin(x_3) \qquad (4.8)
\end{aligned}
$$

Using the equation (4.8), the kinetic co-energy of this system is expressed by

$$T^* = \frac{1}{2} m v_{c.o.m}^2 = \frac{1}{2} m(\dot{x}_2^2 + l^2 \cdot \dot{x}_3^2 + 2 \cdot \dot{x}_2 \cdot l \cdot \dot{x}_3 \cdot \sin(x_3)) \qquad (4.9)$$

In final, the kinetic co-energy, potential energy, dissipation function, and virtual work can be calculated by the equation (4.2), (4.3), (4.4), (4.5), (4.6), (4.7), and (4.8). Therefore, using the *Lagrange's Equation*, three equations of motion are able to be set up such that

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_1} - \frac{\partial T^*}{\partial x_1} + \frac{\partial V}{\partial x_1} + \frac{\partial D}{\partial \dot{x}_1} = e_{x_1}^a \qquad (4.10)$$

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_2} - \frac{\partial T^*}{\partial x_2} + \frac{\partial V}{\partial x_2} + \frac{\partial D}{\partial \dot{x}_2} = e_{x_2}^a \qquad (4.11)$$

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_3} - \frac{\partial T^*}{\partial x_3} + \frac{\partial V}{\partial x_3} + \frac{\partial D}{\partial \dot{x}_3} = e_{x_3}^a \qquad (4.12)$$

$$
\begin{aligned}
&\text{Where } \delta W = e_{x_1}^a \cdot \delta x_1 + e_{x_2}^a \cdot \delta x_2 + e_{x_2}^a \cdot \delta x_2 \\
&= \text{-mg}\delta h = \underbrace{-mg}_{e_{x_2}^a} \cdot \delta x_2 \underbrace{-mgl\sin(x_3)}_{e_{x_3}^a} \cdot \delta x_3
\end{aligned}
$$

To get the result of differentiation, the $\frac{\partial T^*}{\partial \dot{x}_i}$ terms are expressed such that

$$\frac{\partial T^*}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial T^*}{\partial \dot{x}_1} = & 0 \\ \frac{\partial T^*}{\partial \dot{x}_2} = & m\Big(\dot{x}_2 + l \cdot \dot{x}_3 \cdot \sin(x_3)\Big) \\ \frac{\partial T^*}{\partial \dot{x}_3} = & m\Big(l^2 \cdot \dot{x}_3 + \dot{x}_2 \cdot l \cdot \sin(x_3)\Big) \end{cases} \qquad (4.13)$$

, using the above equations, the $\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_i}$ can be expressed such that

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_1} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_2} = & m\Big(\ddot{x}_2 + l \cdot \ddot{x}_3 \sin(x_3) + l \cdot \dot{x}_3^2 \cos(x_3)\Big) \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_3} = & m\Big(l \cdot \ddot{x}_3 + \ddot{x}_2 \cdot l \cdot \sin(x_3) + \dot{x}_2 \cdot \dot{x}_3 \cdot l \cdot \cos(x_3)\Big) \end{cases} \qquad (4.14)$$

To get the second part of equation (4.10), (4.11), and (4.12) - $\frac{\partial T^*}{\partial x_i}$

$$\frac{\partial T^*}{\partial x_i} \Rightarrow \begin{cases} \frac{\partial T^*}{\partial x_1} = & 0 \\ \frac{\partial T^*}{\partial x_2} = & 0 \\ \frac{\partial T^*}{\partial x_3} = & m \cdot l \cdot \cos(x_3) \cdot \dot{x}_2 \dot{x}_3 \end{cases} \qquad (4.15)$$

To calculate the potential energy term

$$\frac{\partial V}{\partial x_i} \Rightarrow \begin{cases} \frac{\partial V}{\partial x_1} = & k \cdot x_1 \\ \frac{\partial V}{\partial x_2} = & 0 \\ \frac{\partial V}{\partial x_3} = & 0 \end{cases} \tag{4.16}$$

In addition, to get the dissipation function terms

$$\frac{\partial D}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial D}{\partial \dot{x}_1} = & c(\dot{x}_1 - \dot{x}_2) \\ \frac{\partial D}{\partial \dot{x}_2} = & -c(\dot{x}_1 - \dot{x}_2) \\ \frac{\partial D}{\partial \dot{x}_3} = & 0 \end{cases} \tag{4.17}$$

Put together equations (4.14), (4.15), (4.16) and (4.17), the equations of motion of this example can be expressed such that

$$\begin{aligned} kx_1 + c\left(\dot{x}_1 - \dot{x}_2\right) &= 0 \\ m\left(\ddot{x}_2 + l\ddot{x}_3 \sin(x_3) + l\dot{x}_3^2 \cos(x_3)\right) - c\left(\dot{x}_1 - \dot{x}_2\right) &= -mg \\ m\ddot{x}_2 l \sin(x_3) + ml^2 \ddot{x}_3 &= -mgl \sin(x_3) \end{aligned} \tag{4.18}$$

Above equations (4.18) are the result of this example. Until now, equations of motion of this example has derived by hand. Now, the result used the program whose algorithm has been introduced in chapter 3 will be shown. The input file consists of the constant variables, coordinates, energy terms, and work in this example. Although the multiplier which is associated with a constraints equation, actually, must be included to the input file, the multiplier will not be included in this example because all coordinates are independent on each other. The input of the constraint equation must be default value "0". The first input is the number of constant variables that is 6 such as c(damper constant), k(spring constant), g(gravitational constant), l(length of pendulum), m(mass), and a(number 2). When any constant variables can have especial value, the constant value can have it, such as the gravitational constant and constant "2". The next step is the input of the number of displacement coordinates, such as x_1, x_2, x_3 which have already chosen when equations of motion has derived. The flow coordinates can be input like displacement coordinates input.

After that, equations of energy terms (4.9), (4.3), and (4.4), must be input with the work corresponding to each coordinate like the figure (4.3). The flow terms are expressed by time

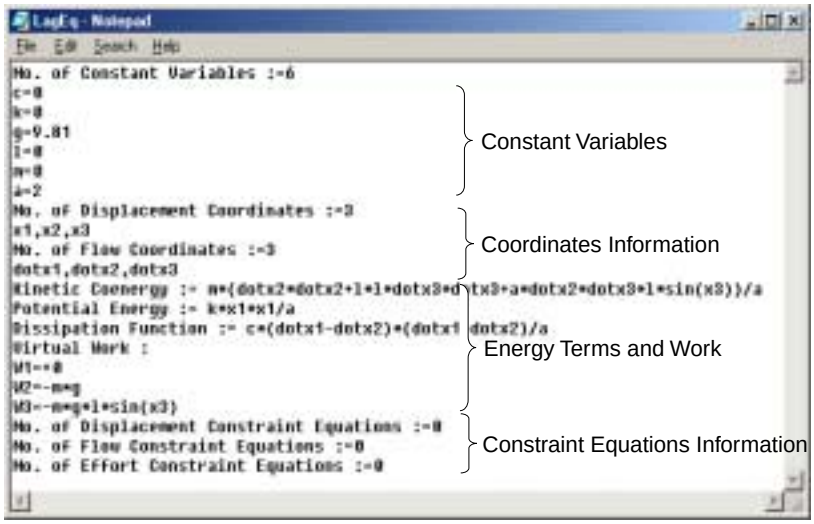

Figure 4.3: Input file for mechanical translation system example

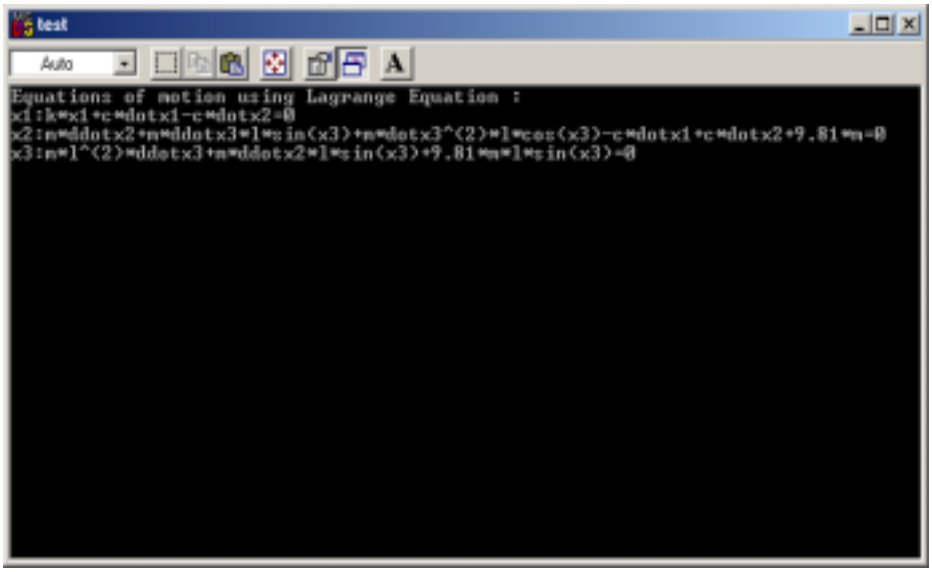

Figure 4.4: The result of translational system example using the program

differentiation of displacement. According to the symbolic program, the time differentiation of some function is expressed by "dot(FUNCTION, t)". But for the user's comfortability, the flow coordinates must be chosen by users[2]. The prefix method can be used in this program.

The last inputs are the constraint equations which is optional inputs. If there is no constraint equation, just input "0"[3] at the number of constraint equation. In this example, all kind of constraint equations don't exist because all coordinates are independent respectively.

Table 4.1: Comparison two result for translational system

Coordinate	Equations of Motion
x_1	*Hand* : $kx_1 + c\,(\dot{x}_1 - \dot{x}_2) = 0$
	program : k * x1 + c*dotx1 - cdotx2 = 0
x_2	*Hand* : m $\left(\ddot{x}_2 + l\ddot{x}_3 \sin(x_3) + l\dot{x}_3^2 \cos(x_3)\right) - c\,(\dot{x}_1 - \dot{x}_2)$ = -mg
	program : m *ddotx2 + m*l*ddotx3*sin(x3)+m*l*dotx3$\hat{}$(2)*cos(x3)-c*dotx1+c*dotx2+9.81*m =0
x_3	*Hand* : m $\ddot{x}_2 l \sin(x_3) + ml^2\ddot{x}_3 = -mgl \sin(x_3)$
	program : m*l$\hat{}$(2)*ddotx3+m*ddotx2*l*sin(x3)+9.81*m*l*sin(x3) =0

According to the above explanation, the input file can be made easily. The figure (4.3) shows the input file for the mechanical translation system example. Additionally, the figure (4.4) show the result of this example using the program. The table (4.1)[4] are showing the same result with that of hands.

4.3 Mechanical System - Rotation

The figure (4.5) shows the mechanical system with rotational elements. In the Conveyor system, the deflection will be neglected for making the system easy. Basically, the way to

[2]The use of prefix or affix - "*dot*" - is highly recommanded for recognizing easily.

[3]0 is default input. See appendix A.

[4]where "ddot" means twice time differentiation, x1 = x_1

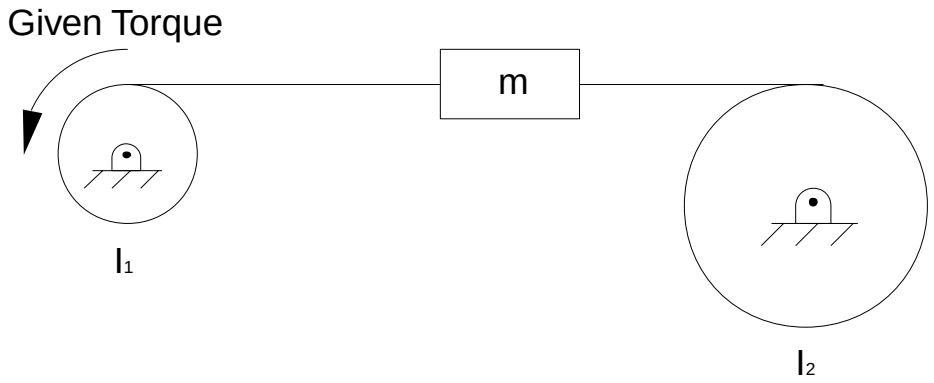

Figure 4.5: Mechanical System - Conveyor Belt

derive the equations of motion is exactly same with the mechanical translational system. However, the figure (4.5) is not free body diagram for choosing the coordinates. Therefore the free body diagram must be drawn to choose the coordinates and decide the energy terms.

The figure (4.6) is the free body diagram for this rotational system. According to the free body diagram, the number of displacements are four, such as x_1, x_2, x_3, and x_4. Of course, independent coordinates can be chosen such as x_1, x_2 or x_1, x_3 or x_2, x_4 or x_3, x_4, because there are two displacement constraint equation such that

$$\begin{aligned} x_1 = r_2 \times x_4 \quad and \quad x_2 = r_1 \times x_3 \\ \Rightarrow \dot{x}_1 = r_2 \times \dot{x}_4 \quad and \quad \dot{x}_2 = r_1 \times \dot{x}_3 \end{aligned} \tag{4.19}$$

However, let us use *Lagrange's equation* (2.39) to follow *Lagrange's equation* with *Lagrange's multiplier*. The equation (4.20) will be with the constraint equations corresponding to displacement. Recall the equation (2.39).

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} + \sum_{j=1}^{m_1} \frac{\partial \phi_j}{\partial q_i} \cdot \lambda_j - e_i^a = 0 \tag{4.20}$$

where ϕ_j is the function of displacement constraint($\phi_j(q_i, t) = 0$),

λ's are the Lagrange's multiplier, i = 1,2,$\cdots$, n and j=1,2,$\cdots$,m_1

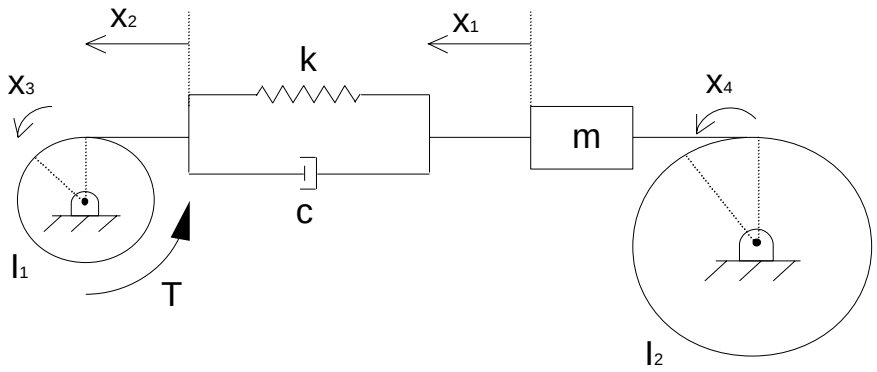

Figure 4.6: Modeling the figure 4.5

To apply the equation (4.20), the kinetic co-energy, potential energy, dissipation function, and virtual work are described in terms of coordinates and some variables such that

$$T^* = \frac{1}{2}I_1\dot{x}_3^2 + \frac{1}{2}I_2\dot{x}_4^2 + \frac{1}{2}m\dot{x}_1^2 \tag{4.21}$$

$$V = \frac{1}{2}k(x_1 - x_2)^2 \tag{4.22}$$

$$D = \frac{1}{2}c(\dot{x}_1 - \dot{x}_2)^2 \tag{4.23}$$

$$\delta W = T\delta x_3 \tag{4.24}$$

When applying to the equation (4.20), energy terms are generated easily. In addition, the constraint equations must be considered because the *Lagrange's equation* with the *Lagrange's multiplier* is used. The constraint equations corresponding to the displacement are modified from the equation (4.20) such that

$$\phi_1 = x_2 - r_1 x_3 = 0 \tag{4.25}$$

$$\phi_2 = x_1 - r_2 x_4 = 0 \tag{4.26}$$

According to the equations (2.39), (4.21), (4.22), (4.23), (4.24), (4.25), and (4.26), four

equations of motion are able to be set up such that

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_1} - \frac{\partial T^*}{\partial x_1} + \frac{\partial V}{\partial x_1} + \frac{\partial D}{\partial \dot{x}_1} + \sum_{j=1}^{m_1} \frac{\partial \phi_j}{\partial x_1} \cdot \lambda_j - e^a_{x_1} = 0$$
$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_2} - \frac{\partial T^*}{\partial x_2} + \frac{\partial V}{\partial x_2} + \frac{\partial D}{\partial \dot{x}_2} + \sum_{j=1}^{m_1} \frac{\partial \phi_j}{\partial x_2} \cdot \lambda_j - e^a_{x_2} = 0$$
$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_3} - \frac{\partial T^*}{\partial x_3} + \frac{\partial V}{\partial x_3} + \frac{\partial D}{\partial \dot{x}_3} + \sum_{j=1}^{m_1} \frac{\partial \phi_j}{\partial x_3} \cdot \lambda_j - e^a_{x_3} = 0$$
$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_4} - \frac{\partial T^*}{\partial x_4} + \frac{\partial V}{\partial x_4} + \frac{\partial D}{\partial \dot{x}_4} + \sum_{j=1}^{m_1} \frac{\partial \phi_j}{\partial x_4} \cdot \lambda_j - e^a_{x_4} = 0$$

$$\begin{aligned}\text{Where } \delta \mathrm{W} &= e^a_{x_1}\delta x_1 + e^a_{x_2}\delta x_2 + e^a_{x_3}\delta x_3 + e^a_{x_4}\delta x_4 \\ &= \underbrace{T}_{e^a_{x_3}} \delta x_3 \text{ and } m_1 = 1{,}2\end{aligned}$$

To get the result of differentiation, the $\frac{\partial T^*}{\partial \dot{x}_i}$ terms are expressed

$$\frac{\partial T^*}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial T^*}{\partial \dot{x}_1} = & m\dot{x}_1 \\ \frac{\partial T^*}{\partial \dot{x}_2} = & 0 \\ \frac{\partial T^*}{\partial \dot{x}_3} = & I_1\dot{x}_3 \\ \frac{\partial T^*}{\partial \dot{x}_4} = & I_2\dot{x}_4 \end{cases} \tag{4.27}$$

, using the above equations, the $\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_i}$ can be expressed such that

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_1} = & m\ddot{x}_1 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_2} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_3} = & I_1\ddot{x}_3 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_4} = & I_2\ddot{x}_4 \end{cases} \tag{4.28}$$

To calculate the second part of the *Lagrange's Equation*,

$$\frac{\partial T^*}{\partial x_i} \Rightarrow \begin{cases} \frac{\partial T^*}{\partial x_1} = & 0 \\ \frac{\partial T^*}{\partial x_2} = & 0 \\ \frac{\partial T^*}{\partial x_3} = & 0 \\ \frac{\partial T^*}{\partial x_4} = & 0 \end{cases} \tag{4.29}$$

To calculate the potential energy term

$$\frac{\partial V}{\partial x_i} \Rightarrow \begin{cases} \frac{\partial V}{\partial x_1} = & k(x_1 - x_2) \\ \frac{\partial V}{\partial x_2} = & -k(x_1 - x_2) \\ \frac{\partial V}{\partial x_3} = & 0 \\ \frac{\partial V}{\partial x_4} = & 0 \end{cases} \tag{4.30}$$

To calculate the dissipation function terms

$$\frac{\partial D}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial D}{\partial \dot{x}_1} = & c(\dot{x}_1 - \dot{x}_2) \\ \frac{\partial D}{\partial \dot{x}_2} = & -c(\dot{x}_1 - \dot{x}_2) \\ \frac{\partial D}{\partial \dot{x}_3} = & 0 \\ \frac{\partial D}{\partial \dot{x}_4} = & 0 \end{cases} \tag{4.31}$$

Finally, using the equation (4.25) and (4.26), the constraint equations' terms are calculated such that

$$\sum_{j=1}^{2} \frac{\partial \phi_j}{\partial x_i} \lambda_j \Rightarrow \begin{cases} \frac{\partial \phi_1}{\partial x_1}\lambda_1 + \frac{\partial \phi_2}{\partial x_1}\lambda_2 = & \lambda_2 \\ \frac{\partial \phi_1}{\partial x_2}\lambda_1 + \frac{\partial \phi_2}{\partial x_2}\lambda_2 = & \lambda_1 \\ \frac{\partial \phi_1}{\partial x_3}\lambda_1 + \frac{\partial \phi_2}{\partial x_3}\lambda_2 = & -r_1\lambda_1 \\ \frac{\partial \phi_1}{\partial x_4}\lambda_1 + \frac{\partial \phi_2}{\partial x_4}\lambda_2 = & -r_2\lambda_2 \end{cases} \tag{4.32}$$

Put together equations (4.28), (4.29), (4.30), (4.31), and (4.32), the equations of motion of this example can be expressed such that

$$m\ddot{x}_1 + k(x_1 - x_2) + c(\dot{x}_1 - \dot{x}_2) + \lambda_2 = 0 \tag{4.33}$$

$$-k(x_1 - x_2) - c(\dot{x}_1 - \dot{x}_2) + \lambda_1 = 0 \tag{4.34}$$

$$I_1\ddot{x}_3 - r_1\lambda_1 = T \tag{4.35}$$

$$I_2\ddot{x}_4 - r_2\lambda_2 = 0 \tag{4.36}$$

where λ_1 and λ_2 are the Lagrange's multiplier,

$$\phi_1 = x_1 - r_2 x_4 = 0,\ \phi_2 = x_2 - r_1 x_3 = 0$$

Equations (4.33), (4.34), (4.35), and (4.36) are the result of this section's example to use displacement constraint equations. This results are confirmed by computer's results. The previous equations (4.33), (4.34), (4.35), and (4.36) are the result of this example. Until

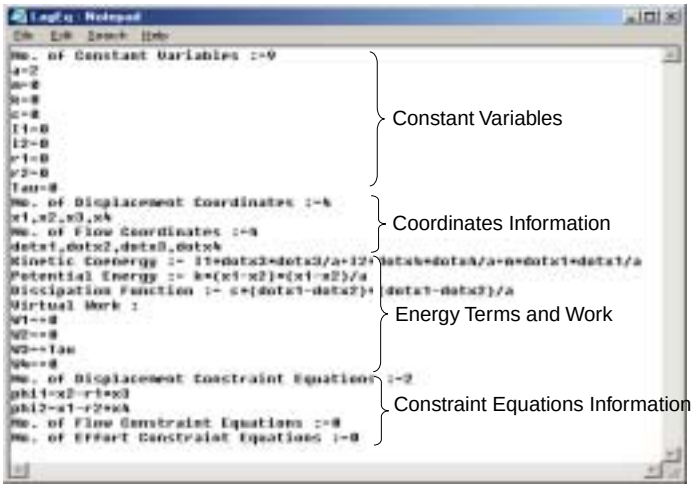

Figure 4.7: Input the constant variables and coordinates for mechanical rotational system example

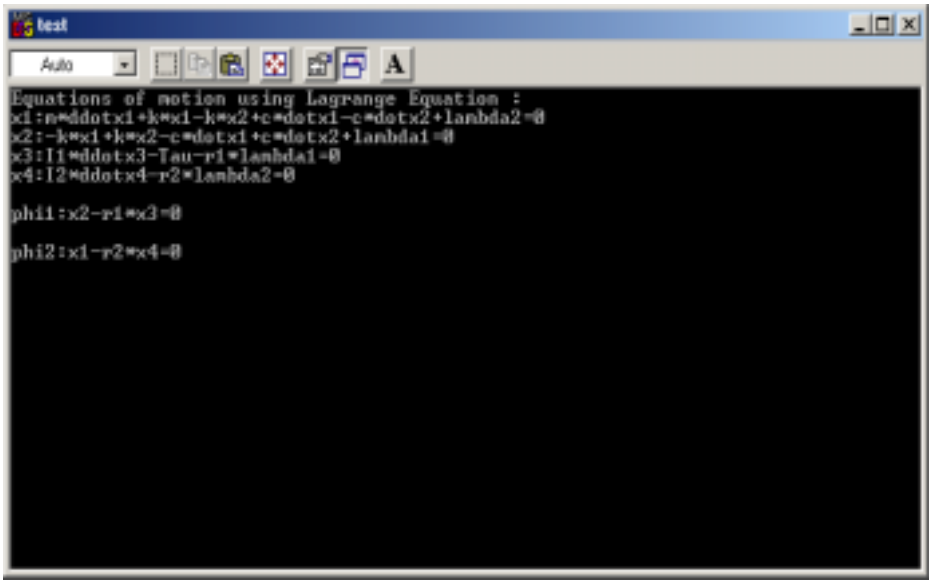

Figure 4.8: The result of rotational system example using the program

now, equations of motion of this example have derived by hand. All input file is similar with the previous input file except the *Lagrange's multiplier*, because coordinates of this example are dependent on each other.

The first input is the number of constant variables that is 8, such as c(damper constant), k(spring constant), I_1(Momentum of disk 1), I_2(Momentum of disk 2), m(mass of belt), r_1(radius of disk 1), r_2(radius of disk 2), and a(number 2). The next step is an input of an number of displacement coordinates, such as x_1, x_2, x_3 and x_4, which have already chosen when equations of motion has derived. The flow coordinates can be input like displacement coordinates input. After that, the energy terms equation (4.21), (4.22), and (4.23), must be input with the work corresponding to each coordinate like figure (4.7). When the work corresponding to each coordinate has been input, the sign (+ or -) must be included. Also, the default input for work is " + 0 " not 0. The last inputs are the

Table 4.2: Comparison two results

Coordinate	Equations of Motion
x_1	*Hand* : $m\ddot{x}_1 + k(x_1 - x_2) + c(\dot{x}_1 - \dot{x}_2) + \lambda_2 = 0$
	program : m*ddotx1+k*x1-k*x2+c*dotx1-c*dotx2+lambda2=0
x_2	*Hand* : $-k(x_1 - x_2) - c(\dot{x}_1 - \dot{x}_2) + \lambda_1 = 0$
	program : -k*x1+k*x2-c*dotx1+c*dotx2+lambda1=0
x_3	*Hand* : $I_1\ddot{x}_3 - r_1\lambda_1 = T$
	program : I1*ddotx3-T-r1*lambda1=0
x_4	*Hand* : $I_2\ddot{x}_4 - r_2\lambda_2 = 0$
	program : I2*ddotx4-r2*lambda2=0

constraint equations which exist in this example. There are two displacement constraint equations like equations (4.20). According to the above explanation, the input file[5] can be made easily.

[5]See Appendix A.

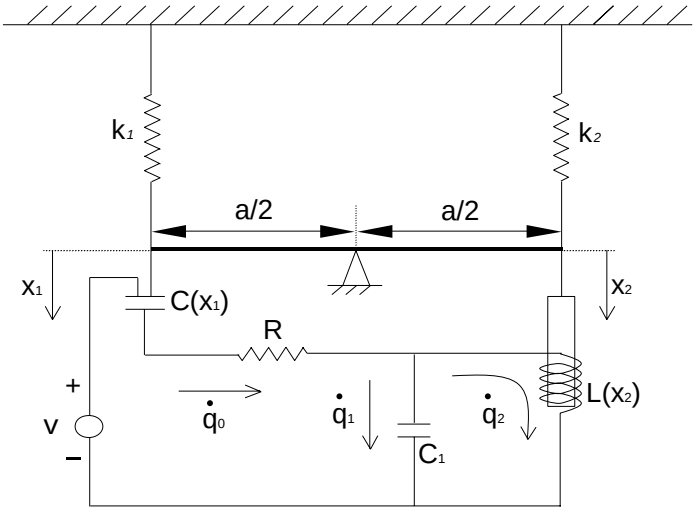

Figure 4.9: Electrical System - Combined System

The figure (4.7) shows the input file for the mechanical translation system example. Additionally, the figure (4.8) show the result of this example using the program. The table (4.2) [6] are showing the same result with that of hand.

4.4 Electrical System

From this section, just the way to make the input file and the result are introduced. The figure (4.9) shows an example of electrical system combined the mechanical translational system with the transformer which causes to make a constraint equation. This system is a simplified example for helping to understand. This electrical system has particular given inductance and capacitors that are expressed such that

$$c(x_1) = c_0/(d - x_1) \tag{4.37}$$

$$L(x_2) = L_0 + bx_2 \tag{4.38}$$

When the input file for this example is made, the first input is the number of constant variables. This example has 9 constant variables, such as c_0, d(constant related to the

[6] where "ddot" means twice time differentiation, x1= x_1...

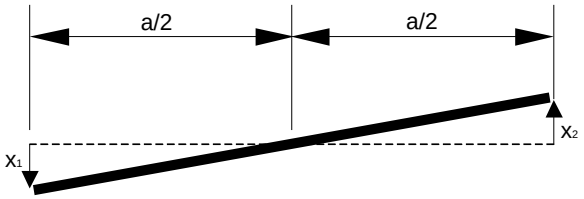

Figure 4.10: Geometric figure in electrical system

capacitor function), b, L_0(constant related to the inductance function), c_1(other capacitor constant), R(resistor), a(length of lever), k(spring constant), and 2(number). The number of constant variables must be input and each variables are input the symbol whatever user can choose. However, the constant variables and coordinates must be distinguished [7]. After that, the number of coordinates is 5, which will be the second input, according to the figure (4.9), the coordinates and the flow are chosen such that

$$\text{Coordinates ; } x_1,\ x_2,\ q_0,\ q_1,\ q_2$$
$$\text{Flows ; } \dot{x}_1,\ \dot{x}_2,\ \dot{q}_0,\ \dot{q}_1,\ \dot{q}_2$$

The lever in the system creates the displacement constraint equations because of the geometric figure (4.10). According to the figure (4.10), it is very obvious to prove that $x_1 = -x_2$. The constraint equation corresponding to the displacement, therefore, is following equation.

$$\phi = x_1 + x_2 = 0 \tag{4.39}$$

Due to the electrical currency, the flow constraint equation exits such that

$$\psi = \dot{q}_0 - \dot{q}_1 - \dot{q}_2 = 0 \tag{4.40}$$

Let us think each energy term, which will be an important part of input file in program.

$$\begin{aligned} T^* &= \frac{1}{2}L(x_2)\dot{q}_2^2 = \frac{1}{2}(L_0 + bx_2)\dot{q}_2^2 \\ V &= \frac{1}{2}kx_1^2 + \frac{1}{2}kx_2^2 + \frac{1}{2}\frac{q_0^2}{c(x_2)} + \frac{1}{2}\frac{q_1^2}{c_1} \end{aligned} \tag{4.41}$$

[7] Don't use same symbol

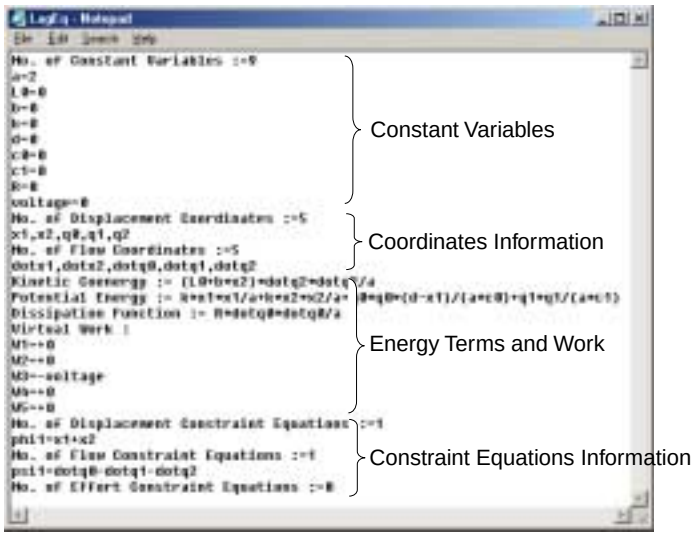

Figure 4.11: The typical example of input file

$$= \frac{1}{2}kx_1^2 + \frac{1}{2}kx_2^2 + \frac{1}{2}\frac{q_0^2(d-x_1)}{c_0} + \frac{1}{2}\frac{q_1^2}{c_1} \tag{4.42}$$

$$D = \frac{1}{2}R\dot{q}_0^2 \tag{4.43}$$

$$\delta W = e^a_{x_1}\delta x_1 + e^a_{x_2}\delta x_2 + e^a_{q_0}\delta q_0 + e^a_{q_1}\delta q_1 + e^a_{q_2}\delta q_2$$

$$= \underbrace{v}_{e^a_{q_0}}\delta q_0 \tag{4.44}$$

Using the equation (4.41), (4.42), (4.43), (4.39), and (4.40) with the coordinates and constant variables, input file which is called "*LagEq.txt*", can be made like the figure (4.11). When making input file, users must be careful for using symbols. There must be errors when users use same symbol, also when users use 'KE', 'PE', 'D', which are defined already[8]. After making the input file, just click the batch file which is called *Jae*. Finally, the equations of motion are obtained such as figure (4.12).

In appendix B, equations of motion of this example are obtained by hands and compared.

[8]See appendix A.

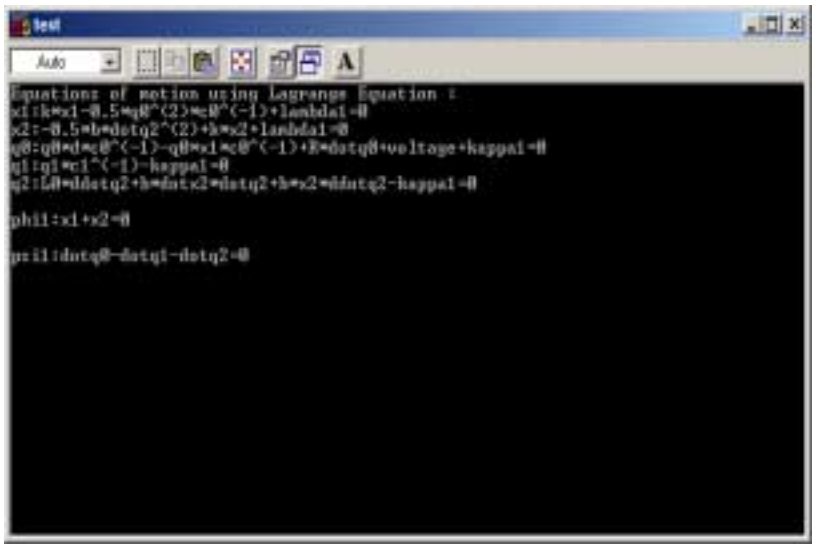

Figure 4.12: The result of electrical system using the program

4.5 Thermal System with Entropy

The figure (4.13) shows the blending system[9] with the electrical system. Two identical liquids of different temperature T_1 and T_2 which are blending in the tank. In addition, there are two flow rate Q_1 and Q_2 which are perfectly mixed in a blender of volume V. The mixed liquids is heated by an electric heater with constant rate Q_{hgen} which generates constant temperature T_Q. Basically, heat transfer governs the thermal system. Also basic analysis method of heat transfer is introduced in Holman's textbook [13]. In this section, to analyze the system using lumped modeling method, the blending system can be drawn again like figure (4.14).

Actually, resistance R_{T1} and R_{T2} are calculated such that

$$R_{t1} = \frac{1}{\rho c_p Q_1} \tag{4.45}$$

$$R_{t2} = \frac{1}{\rho c_p Q_2} \tag{4.46}$$

where ρ is density, c_p is specific heat.

[9] See J. Lowen Shearer & Bohdan T. Kulakowski text book [14].

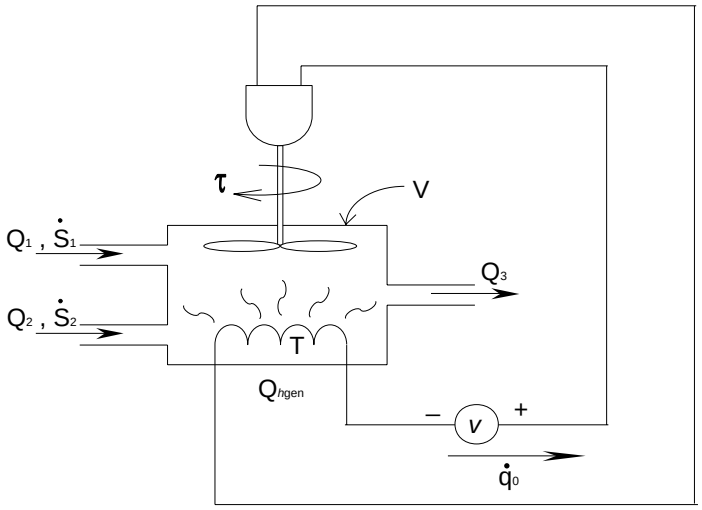

Figure 4.13: The blending system with electrical device

The capacitor C_t, also, can be calculated such that

$$C_t = \rho c_p V \tag{4.47}$$

where ρ is density, c_p is specific heat and V is volume of blending system.

According to the figure (4.14), the flow constraint equation is obtained

$$\psi_1 = \dot{S}_3 - \dot{S}_1 - \dot{S}_2 = 0 \tag{4.48}$$

Because of fluid flow, the equivalent system for fluid flow like figure (4.15) According to the figure (4.15), the flow constraint equation is obtained such that

$$\psi_2 = Q_3 - Q_1 - Q_2 = 0 \tag{4.49}$$

The electrical system consists of resistor, which generates temperature, and DC motor, which operates fans. The effort constraint equation corresponding to the DC motor exists such that

$$\Gamma = \tau - k_\tau = 0 \tag{4.50}$$

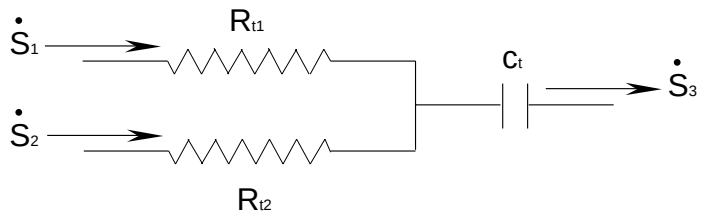

Figure 4.14: The equivalent system with blending system

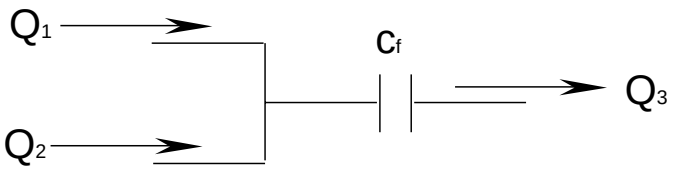

Figure 4.15: The equivalent system with fluid flow

Before making an input file, choose the coordinates of system such that

$$\begin{aligned} Displacement \quad &: \quad \underbrace{V_1, V_2, V_3}_{fluid}, \underbrace{S_1, S_2, S_3}_{thermal}, \underbrace{q_0}_{electric}, \underbrace{\theta}_{fan} \\ Flow \quad &: \quad \overbrace{\dot{V}_1, \dot{V}_2, \dot{V}_3}, \overbrace{\dot{S}_1, \dot{S}_2, \dot{S}_3}, \overbrace{\dot{q}_0}, \overbrace{\dot{\theta}} \end{aligned}$$

Until now, basic concept of blending system with electrical system is introduced. In order to make the input file, count the number of constant variables, such as two(number '2'), τ(torque at DC motor), k_τ(torque constant), c_f(fluid capacitor constant), c_t(thermal capacitor constant), R_{t1}, R_{t2}(thermal resistor constant), R_e(electrical resistor), T(temperature), v(supply voltage), and I(momentum of fan) - 11. The number of displacement coordinates

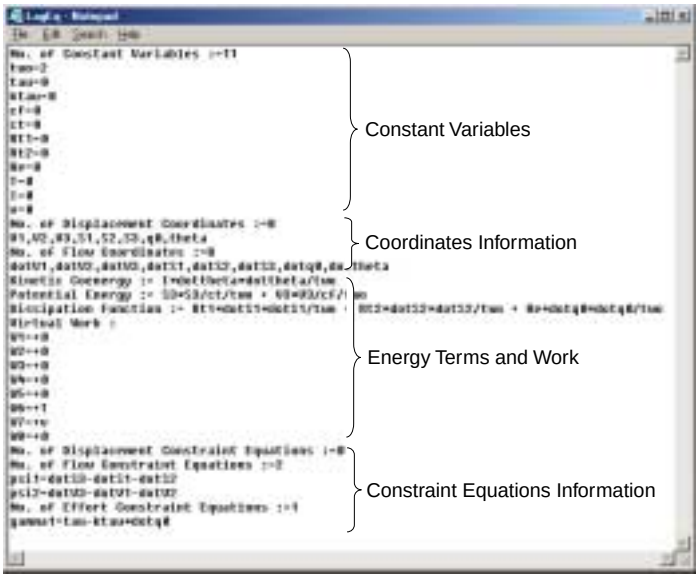

Figure 4.16: The input file for thermal system

is 8. Let us consider the energy terms[10] and work corresponding to coordinates.

$$KE = \frac{1}{2} I \dot{\theta}^2 \tag{4.51}$$

$$PE = \frac{1}{2}\frac{S_3^2}{c_t} + \frac{1}{2}\frac{V_3^2}{c_f} \tag{4.52}$$

$$D = \frac{1}{2} R_{t1} \dot{S}_1^2 + \frac{1}{2} R_{t2} \dot{S}_2^2 + \frac{1}{2} R_e \dot{q}_0^2 \tag{4.53}$$

$$\begin{aligned} \delta W &= e^a_{V_1}\delta V_1 + e^a_{V_2}\delta V_2 + e^a_{V_3}\delta V_3 + e^a_{S_1}\delta S_1 + e^a_{S_2}\delta S_2 + e^a_{S_3}\delta S_3 \\ &\quad + e^a_{q_0}\delta q_0 + e^a_\theta \delta\theta \\ &= T\delta S_3 + v\delta q_0 \end{aligned} \tag{4.54}$$

Finally, required information for making input file - equations (4.51), (4.52), (4.53), (4.54), (4.48), (4.49), and (4.50) - is obtained completely. The figures (4.16) and (4.17) show input file and result for thermal system. In appendix B, results by hand is derived.

[10] Assume that all thermal element simplify in terms of R_{t1}, R_{t2}, and c_t.

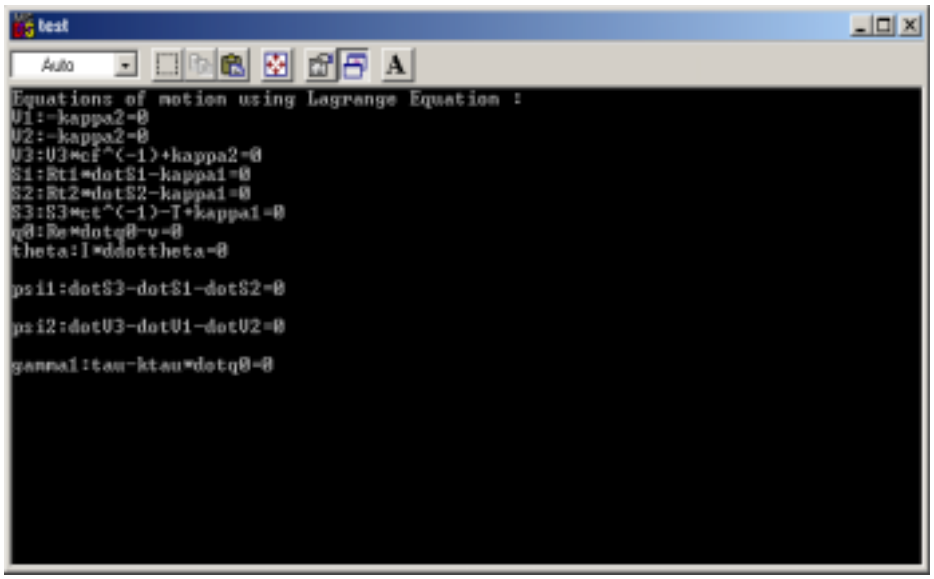

Figure 4.17: The result of thermal system

4.6 Fluid Mechanics System with Compressive Fluid

The figure (4.18) shows a pneumatic position[11] system with chime solenoid[12] and a rack and pinion system which transfer a rotational motion to a translational motion. The pneumatic system is applied in many system, such as measurement and automatic control in a process industries, for heating and ventilating controls and in certain military and aerospace systems.

First of all, let us assume about the pneumatic system[13]. It works from a pressure source of clean air which is a compressive fluid. The atmospheric pressure P_{atm} can serve as a secondary reference, but the primary reference is a perfect vacuum so that all pressures

[11]Form woods and Lawrence [15]

[12]Form Crandall [?]

[13]From J. Lowen Shearer & Bohdan T. Kulakowski [14] and Oagata [26].

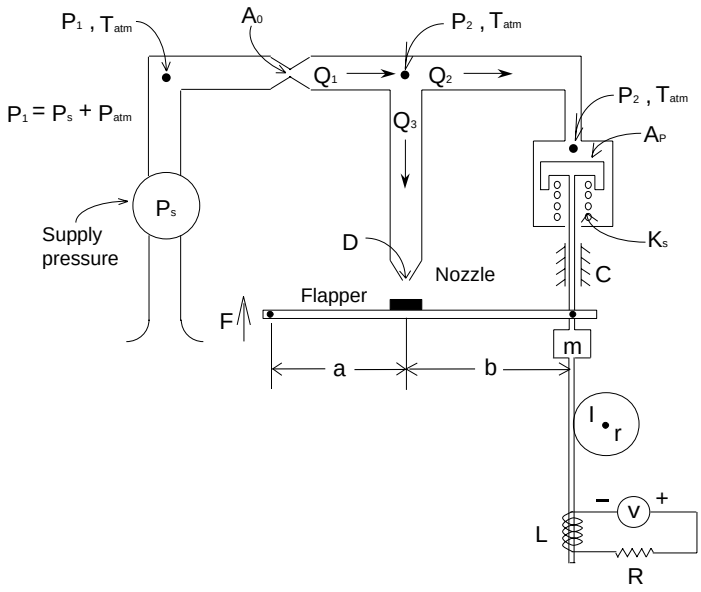

Figure 4.18: The pneumatic position system with electrical device

are expressed as absolute pressures[14].

According to the figure (4.19), the air flow with γ, local weight density of the air, makes an constraint equation obviously such that

$$\begin{aligned} \psi &= Q_1 - Q_2 - Q_3 \\ &= Q_{1w}\gamma - Q_{2w}\gamma - Q_{3w}\gamma = 0 \end{aligned} \tag{4.55}$$

Before another analysis, let think about resistors and capacitor in figure (4.19). Relationship equations of resistors and capacitor must be checked because fluid of this system is air. The Allen, Edward & Sabersky's textbook [30] and Frank's textbook [37] introduced the method to get relationship equations such that

$$\frac{Q_{wNLR}}{C_d A_0} = \begin{cases} \frac{C_2 P_u}{(T_u)^{0.5}}, & for \quad 0 < \frac{P_d}{P_u} < 0.5 \\ \frac{2C_2 P_u}{(T_u)^{0.5}} \left[\frac{P_d}{P_u} \left(1 - \frac{P_d}{P_u} \right) \right]^{0.5} & for \quad 0.5 < \frac{P_d}{P_u} < 1 \end{cases} \tag{4.56}$$

where

[14] Except constant supply pressure P_s

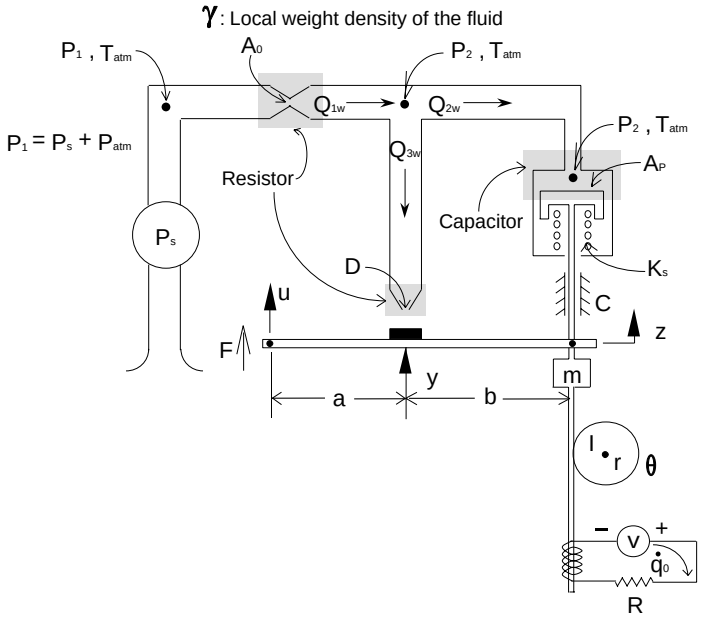

Figure 4.19: The analysis of the pneumatic position system

$$
\begin{aligned}
C_d &= discharge \quad coefficient, \quad 0.85 \quad for \quad air \\
A_0 &= orifice \quad area \\
Q_{wNLR} &= Weight \quad rate \quad of \quad flow \\
C_2 &= g\sqrt{\frac{k}{R}\left(\frac{k+1}{2}\right)^{(k+1)/(k-1)}} = 0.532 \quad for \quad air \\
P_u &= upstream \quad pressure \\
P_d &= downstream \quad pressure \\
T_u &= upstream \quad temperature \\
R &= the \quad perfect \quad gas \quad constant \\
k &= the \quad specific \quad heat \quad ratio, \quad 1.4 \quad for \quad air
\end{aligned}
$$

For orifice A_0, therfore, using the equation (4.56)

$$Q_{1w} = \frac{Q_1}{\gamma} = \frac{C_{d1} A_0 C_2 P_1}{(T_{atm})^{0.5}} \tag{4.57}$$

For the flapper-nozzle orifice, using the equation (4.56)

$$Q_{3w} = \frac{Q_1}{\gamma} = C_{d2}\pi D y_0 C_2 P_2 \left[\frac{\left(\frac{P_{atm}}{P_2}\right)\left(1 - \frac{P_{atm}}{P_2}\right)}{T_{atm}} \right]^{0.5} \tag{4.58}$$

Both equations (4.57) and (4.58) are the effort constraint equation expressed such that

$$\Gamma_1 = \frac{Q_1}{\gamma} - \frac{C_{d1} A_0 C_2 P_1}{(T_{atm})^{0.5}} = 0 \tag{4.59}$$

$$\Gamma_2 = \frac{Q_3}{\gamma} - C_{d2}\pi D y_0 C_2 P_2 \left[\frac{\left(\frac{P_{atm}}{P_2}\right)\left(1 - \frac{P_{atm}}{P_2}\right)}{T_{atm}} \right]^{0.5} = 0 \tag{4.60}$$

Equation (4.59) and (4.60) are used in the input file.

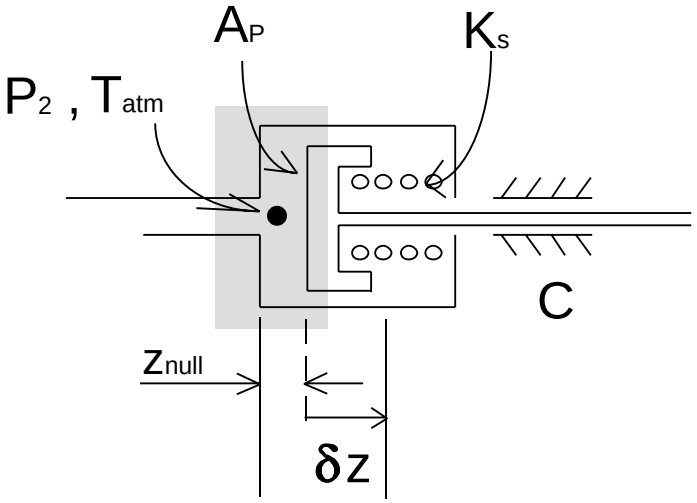

Figure 4.20: The piston position

For the piston's capacitor which is drawn in figure (4.20) and isothermal case[15], following equations can be used for calculating a capacitance(C_{fw}) of pneumatic system such that

$$C_{fw} = \begin{cases} \frac{gV(t)}{RT_1} + \frac{gP_1(t)A_p^2}{RT_1 k_s}, & for \quad `slow' \quad changes \quad in \quad P_1 \\ \frac{gV(t)}{kRT_1} + \frac{gP_1(t)A_p^2}{RT_1 k_s}, & for \quad `fast' \quad changes \quad in \quad P_1 \end{cases} \tag{4.61}$$

where

[15] Explain later detail.

$$
\begin{aligned}
P_1(t) &= pressure \quad at \quad 1, \quad function \quad of \quad time \\
V(t) &= volume \quad of \quad piston \\
k_s &= spring \quad constant \\
A_p &= area \quad of \quad piston \\
g &= gravitational \quad constant \\
R &= the \quad perfect \quad gas \quad constant \\
k &= the \quad specific \quad heat \quad ratio, \quad 1.4 \quad for \quad air
\end{aligned}
$$

According to the figure (4.20) and equation (4.61), the volume of capacitor will be $A_p(z_{null} + \delta z)$. Therefore, the C_{fw} is expressed such that

$$C_{fw} = \frac{gA_p}{RT_{atm}}\Big(z_{null} + \delta z + \frac{P_2 A_p}{k_s}\Big) \tag{4.62}$$

Using the equation (4.62), relationship between Q_{3w}, and P_2 can be expressed such that

$$
\begin{aligned}
Q_{2w} &= C_{fw}\frac{dP_2}{dt} \Rightarrow \\
V_{2w} = \frac{V_2}{\gamma} &= \int C_{fw}\frac{dP_2}{dt} \\
&= \int \frac{gA_p}{RT_{atm}}\Big(z_{null} + \underbrace{\delta z}_{=A_p\frac{P_2 - P_{atm}}{k_s}} + \frac{P_2 A_p}{k_s}\Big)\frac{dP_2}{dt}dt \\
&= \int \frac{gA_p}{RT_{atm}}\Big(z_{null} - A_p\frac{P_{atm}}{k_s} + 2\frac{P_2 A_p}{k_s}\Big)\frac{dP_2}{dt}dt \\
&= \frac{gA_p}{RT_{atm}}\left\{\Big(z_{null} - A_p\frac{P_{atm}}{k_s}\Big)t + \frac{P_2^2 A_p}{k_s}\right\}
\end{aligned} \tag{4.63}
$$

Using the equation (4.63), the third effort constraint equation can be made such that

$$\Gamma_3 = \frac{V_2}{\gamma} - \frac{gA_p}{RT_{atm}}\left\{\Big(z_{null} - A_p\frac{P_{atm}}{k_s}\Big)t + \frac{P_2^2 A_p}{k_s}\right\} = 0 \tag{4.64}$$

The equation (4.64) is used in input file also.

Because of the lever, there is constraint equation. The lever of the flapper is not deformed, which means the lever is a rigid body, to simplify. Therefore, the lever makes a constraint equation. The constrain equation corresponding to displacements, can be ob-

tained applying superposition because u and z affect y respectively such that

$$\delta y = \frac{b}{a+b}\delta u \tag{4.65}$$

$$\delta y = \frac{-a}{a+b}\delta z \tag{4.66}$$

Because equations (4.65) and (4.66) are linear, these equations can be combined such that

$$\begin{aligned} \delta y &= \frac{-a}{a+b}\delta z + \frac{b}{a+b}\delta u \\ \phi_1 &= y + \frac{a}{a+b}z - \frac{b}{a+b}u = 0 \end{aligned} \tag{4.67}$$

The equation (4.67) will be used in input file. Also, assume that changes in P_2 occur slowly, which means that the chamber is an isothermal chamber.

In the figure (4.18), an electrical system is included with the rack and pinion. The electrical system expresses an chime solenoid[16] This chime solenoid is connected be the rack and pinion system which makes a constraint equation corresponding to displacement coordinates. The figure (4.21) shows a relationship between rack and pinion such that

$$\begin{aligned} z &= -r\theta \\ \phi_2 &= z + r\theta = 0 \end{aligned} \tag{4.68}$$

According to the figure (4.18), displacement coordinates are chosen such that

$$Displacement \quad coordinates \quad : \quad u, y, z, \theta \tag{4.69}$$

Also, the system has 4 flow coordinates such that

$$Flow \quad coordinates \quad : \quad Q_1, Q_2, Q_3, \dot{q}_0 \tag{4.70}$$

Therefore, using expressions (4.69) and (4.70),all coordinates are expressed like

$$displacement \quad coordinates \quad : \quad u, y, z, \theta, V_1, V_2, V_3, q_0 \tag{4.71}$$

$$flow \quad coordinates \quad : \quad \dot{u}, \dot{y}, \dot{x}, \dot{\theta}, Q_1, Q_2, Q_3, \dot{q}_0 \tag{4.72}$$

[16] See the textbook of Crandall [?].

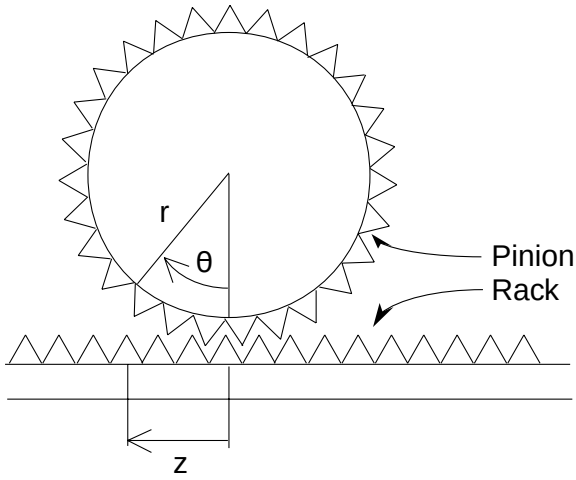

Figure 4.21: The relationship between rack and pinion

Equations (4.71) and (4.72) are the coordinates input.

Before energy terms are obtained, define the constant variables. There are many constant variables, such as one (number 1), two(number 2), a(lever length), b(lever length), m(mass), g(gravitational constant, 9.81), k_s(spring constant), c(damper constant), I(momentum), L(electrical Inductance), r(pinion's radius), T(torque), F(input force), v(voltage), pi(pi, 3.14), γ(local weight density of the air), Dn(diameter of nozzle), A_0(area of orifice), A_p(area of bellows), C_{d1}, C_{d2}(discharge coefficient, 0.85 for air), $C_2(0.532(degR)^{0.5}$/sec for air), P_1(pressure at 1), P_2(pressure at 2), P_s(supply constant pressure), P_{atm}(pressure in atmosphere), T_{atm}(temperature in atmosphere), R(the perfect gas constant), k(the specific heat ratio, 1.4 for air), y_{null}(length without any force), z_{null}(length when P_2=P_{atm}) and R_e(resistance in electrical system). Total of constant variables is 32. Let us consider the energy terms - Kinetic co-energy, Potential energy, Dissipation function such that

$$K.E = \frac{1}{2}m\dot{z}^2 + \frac{1}{2}L\dot{q}_0^2 + \frac{1}{2}I\omega^2 \tag{4.73}$$

$$P.E = \frac{1}{2}k_s z^2 \tag{4.74}$$

$$D.E = \frac{1}{2}c\dot{z}^2 + \frac{1}{2}R\dot{q}_0^2 \tag{4.75}$$

After that, figure out the work corresponding to each coordinate. There are several

sources, such as supply pressure, electrical supply, torque, pressure due to the piston actuator, input force, and gravitational force. The work corresponding to each coordinate, therefore, can be expressed such that

$$\left.\begin{aligned} W_u &= F \\ W_y &= 0 \\ W_z &= -A_p(P_2 - P_{atm}) - mg \\ W_\theta &= 0 \\ W_{Q_1} &= P_s \\ W_{Q_2} &= P_2 \\ W_{Q_3} &= P_2 \\ W_{\dot{q}_0} &= v \end{aligned}\right\} \tag{4.76}$$

Finally, all things associated with the input file are obtained. Let us make the input file - "*LagEq.txt*" - like figure (4.22), using equations (4.71), (4.72), (4.73), (4.74), (4.75), (4.76), (4.67), (4.68), (4.55), (4.59), (4.60), and (4.64).

Using the input file like the figure (4.22), the result of *Lagrange's Equation* can be obtained like the figure (4.23). Also, you can confirm the result by hand in appendix B.

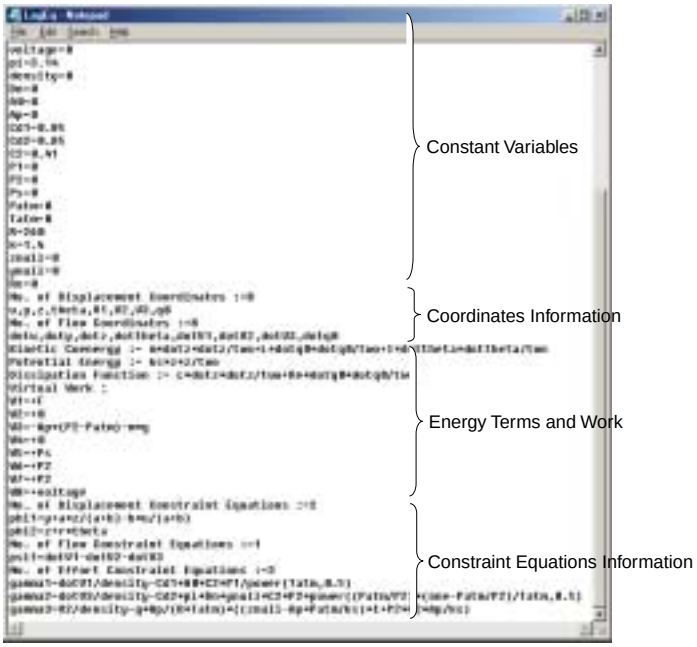

Figure 4.22: The input file for the fluid system example

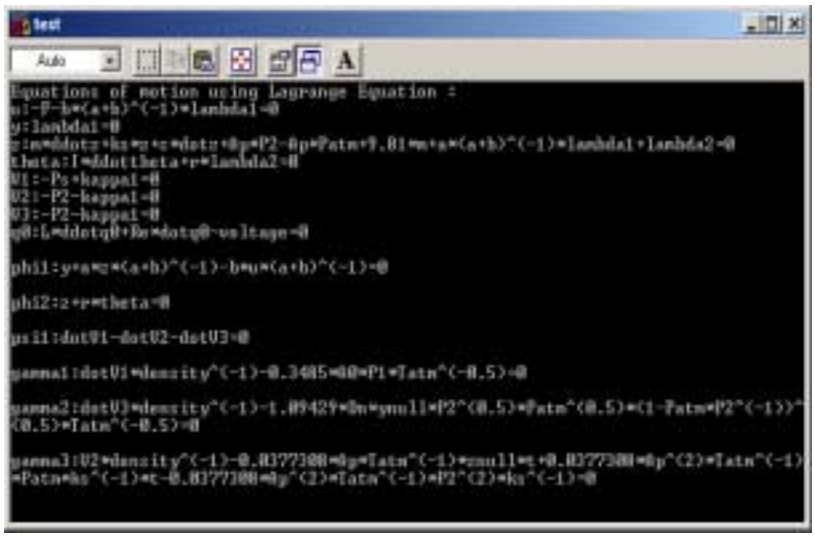

Figure 4.23: The output for the fluid system example

Chapter 5

CONCLUSION AND FUTURE WORKS

Before this chapter, the *Lagrange's Equation*, the *Object-Oriented Program*, and the algorithm using the *Lagrange's Equation* and the C++ have been introduced in detail. In addition, 5 examples show us that the program works very well.

The reason why this algorithm is developed is due to the fact that the commercial package, like Mat lab, Mathmetica, Maple, MuPad and so on, have a problem when they deal with singular matrix with their own manual and functions. Because of this reason, this paper provide some algorithm as a part of solution about that problem.

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} + \sum_{j=1}^{m_1} \frac{\partial \phi_j}{\partial q_i} \cdot \lambda_j + \sum_{j=1}^{m_2} \frac{\partial \psi_j}{\partial \dot{q}_i} \cdot \kappa_j = e_i^a(e_k) \tag{5.1}$$

where ϕ_j is the function of displacement constraint(ϕ_j=0, j = 1,2,$\cdots$, m_1),

ψ_j is the function of flow constraint(ψ_j=0, j = 1,2,$\cdots$, m_2),

Γ_j is the function of efforts constraint(Γ_j=0, j = 1,2,$\cdots$, m_3),

λ_j's and κ_j's are the Lagrange's multiplier, and i = 1,2,$\cdots$, n

When applied to the matrix form[1] for solving numerically, the generalized *Lagrange's Equation* (5.1) must make the singular matrix because of constraint equations. Using the *Lagrange's Equation*, equations of motion of multi-degree problems can be obtained easily. Also, equations of motion of the typical electrical system can be calculated.

In chapter 3, the given program is satisfied with many conditions for making symbol objects, calculating among symbol objects, differentiating some function for dependent or

[1] See the section 2.5

independent variables, and so on. The following conditions are necessary conditions for making symbolic computation.

1. Recognition of Symbol

2. Symbolic Calculus

3. Differentiation of Symbol

4. Define operators

During checking these conditions, several advantages of the *Object-Oriented Program* have been introduced - the ability of attachment and modification with classes and templates is the most useful advantage, i.e, using this advantage, many other functions - are not included in the program - can be added. Because of this advantage, many developers of commercial packages use the *Object-Oriented Program*. Therefore, users can add their own function to the commercial packages easily. In order to make the input file, the way to consist of input file is introduced. When a very complex effort constrain equation is considered, effort must be expressed in terms of other parameter such as 'solve' command in Maple. However, given program[2] can not calculate. This part also will be future work.

In chapter 4, many examples are solved by hand and the program[3] with a little explanation of each system. The source codes are in the **ftp://abs-5.washington.edu/jsuk/sym** with the manual[4] for using the program.

As future works, the many other functions, such as integration, impulse function, and so on, are added in the program. The integration require another class like diffRules for defining special integration. The impulse function, also, is defined its own characteristics

[2]Users must solve that by hand.

[3]Chapter 4 and appendix B show same result

[4]Also, see appendix A.

and has the its own return character for recognizing it in program.

In addition, the numerical solution can be obtained by connecting other program, the purpose of which is to get the numerical values. That will be the next stage for making a completed program.

BIBLIOGRAPHY

[1] Jonathan Alberts. *Contribution to the Stability Analysis and Numerical Solution of Differentail-Algebraic Systems.* Ph.D Dissertation, University of Washington, 1999.

[2] Tom M. Apostol. *Mathematical Analysis.* Addison-Wesley Publishing Company, Inc, 1975.

[3] V.I. Arnorld. *Mathematical Methods of Classical Mechanics.* Springer-Verlag New York Inc., 1989.

[4] Gustavson F.G. Brayton R.K. and Hachtel G.D. A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas. In *Proceedings of the IEEE, 60(1)*, pages 98–108, 1972.

[5] Stephen La Vern Campbell. *Singular systems of differential equations.* Pitman Advanced Publishing Program, 1980.

[6] Stephen La Vern Campbell. *Singular systems of differential equations2.* Pitman Advanced Publishing Program, 1982.

[7] Mary Campione and Kathy Walrath. *The Java Tutorial.* Addison Wesley, 1996.

[8] Stephen H. Crandall. *Dynamics of Mechanical and Electromechanical Systems.* McGraw-Hill, 1968.

[9] Davenport J. H. & Siret Y. & Tournier E. *Computer Algebra: Systems and Algorithms for Algebraic Computations.* Academic Press, London, 1993.

[10] Ivan E. Morse Francis S. Tse and Rolland T. Hinkle. *Mechanical Vibrations Theory and Application.* Allyn and Bacon, 1978.

[11] Werner Greub. *Linear Algebra.* Springer-Verlag New York Inc., 1981.

[12] Steeb W. H. *Problems and Solutions in Theoretical and Mathematical Physics.* World Scientific , Singapore, 1996.

[13] Jack P. Holman. *Heat Transfer.* McGraw-Hill, 1990.

[14] J. Lowen Shearer & Bohdan T. Kulakowski. *Dynamic Modeling and Control of Engineering Systems.* Macmillan Publishing Company, 1990.

[15] Robert L. Woods & Kent L. Lawrence. *Modeling and Simulation of Dynamics Systems.* Prentice Hall, 1997.

[16] Brain C. Fabien & Richard A. Layton. Automatic integration of the lagrangian daes of motion. *Mathematical and Computer Modelling of Dynamics Systems*, 2000.

[17] Brain C. Fabien & Richard A. Layton. Modelling and simulation of physical systems i: An introduction to lagrangian daes. In *4th IASTED Int. Conf., Robotics and Manufacturing*, Honolulu, 1996.

[18] Brain C. Fabien & Richard A. Layton. Systematic modelling using lagrangian daes. *Mathematical and Computer Modelling of Dynamics Systems*, in review.

[19] Richard A. Layton. *Analytical System Dynamics.* Ph.D Dissertation, University of Washington, 1995.

[20] Richard A. Layton. *Principles of Analytical System Dynamics.* Springer-Verlag, 1998.

[21] Manual. *Matlab: The Language of Technical Computing.* The Math Works Inc., 1998.

[22] Eberhard Griepentrog & Roswitha Marz. *Differential-Algebraic Equations and Their Numerical Treatment.* Teubner Verlagsgesellschaft, Leipzig, 1986.

[23] Jerrell M.E. Automatic differentiation in c++. *Journal of Object Oriented Programming*, pages 17–24, 1990.

[24] Leonard Meirovitch. *Introduction to Dynamics and Control.* John Wiley & Sons, 1985.

[25] James L. Meriam and L.G. Kraige. *Engineering Mechanics.* Wiley, 1993.

[26] Katsuhiko Ogata. *Mordern Control Enegineering.* Prentice Hall, 1997.

[27] Stephen Prata. *C++ Prime Plus.* The Waite Group, Inc., 1995.

[28] L. B. Rall. Application of software for automatic differentiation in numerical computation. *Computing*, Suppliment 2:141–156, 1980.

[29] L. B. Rall. Differentiation in pascal-sc. *ACM Trans. on Math. Software*, 10:161–184, 1984.

[30] Allan J. Acosta & Edward G. Hauptmann Rolf H. Sabersky. *Fluid Flow : A First Course in Fluid Mechanics.* Macmillan Publishing Company, New York, 1989.

[31] Herbert Schildt. *C: The Complete Reference.* McGraw-Hill, 1987.

[32] Gilbert Strang. *Linear Algebra and Its Application.* Saunders, 1986.

[33] Yorick Hardy Tan Kiat Shi, Willi-Hans Steeb. *Symbolic C++ An Introduction to Computer Algebra using Object-Oriented Programming.* Springer, 2000.

[34] William T. Thomson. *Theory of Vibration with Applications.* Prentice Hall, 1988.

[35] Firdaus E. Udwadia and Robert E. Kalaba. *Analytical Dynamics.* Cambridge University Press, 1996.

[36] Char B. W. *First Leaves - A Tutorial Introduction to MAPLE V.* Springer-Verlag, New York, 1991.

[37] Frank M. White. *Fluid Mechanics.* McGraw-Hill, 1986.

Appendix A

MANUAL HOW TO MAKE INPUT FILE

In chapter4, first two examples, which are representatives of translational and rotational system, have detail explanation for input file. In this appendix, the basic way how users can make appropriate input file has been introduced.

The figure (A.1) shows a typical example input file. The numbers of the figure means the procedure of making input file and is explained. With the figure (A.1), following explanation matching the number will be helpful for user to make the input file. Because the input file exists like the figure (A.1), modifying the input file is easy.

1. A number of constant variables

   Key words [1] : "No. of Constant Variables :="

2. Each constant variable can be alloted their own value if needed.

   - Default value is "0" [2]
   - The number of constant variables must be same with symbols of the constant variables
   - The symbols of constant variable can be chosen by users.

3. A number of *displacement coordinates*

   Key words : "No. of Displacement Coordinates :="

---

[1] When the program is run, the program will check the *Key Words* to recognize and create the object. Users must keep to make exact input file.

[2] When users want it to use in just symbolic computation, users must use the default value.

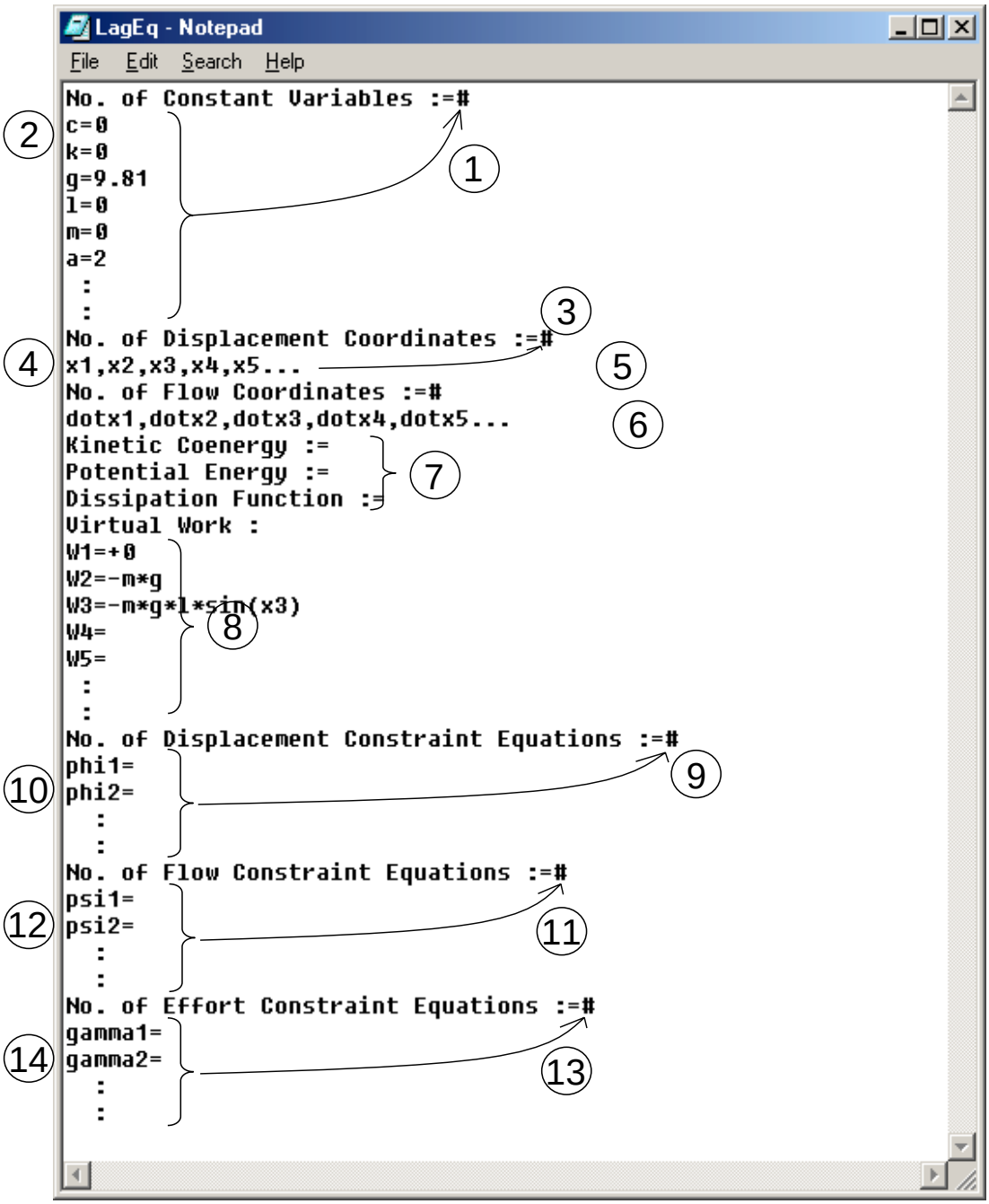

Figure A.1: A typical example of input file

4. Symbols of displacement coordinate can be picked

   - Each displacement coordinate are separated by ","
   - General coordinates are easy to be distinguished like x_1, x_2, x_3, $\cdots$
   - The number of displacement coordinates must be same with symbols of the displacement coordinates
   - Don't use same symbols with constants

5. A number of *flow coordinates*

   Key words : "No. of Flow Coordinates :="

   - Strictly, the number of flow coordinates is same with the number of displacement coordinates

6. Symbols of Flow coordinate can be picked

   - Each flow coordinate is separated by ","
   - The number of flow coordinates must be same with symbols of the flow coordinate
   - Generally, the flow coordinates are similar with the displacement coordinates
   - Displacement coordinates are easy to distinguished

7. Energy terms

   - Kinetic co-energy

     Key words : "Kinetic Coenergy :="

   - Potential energy

     Key words : "Potential Energy :="

   - Dissipation Function

     Key words : "Dissipation Function :="

8. Work

   Key words : "Virtual Work :="

   - According to the *Lagrange's Equation*, work functions corresponding to each coordinate are required.
   - Work must have the sign(+ or -)
   - The number of work functions must be same with the number of coordinates

9. A number of *displacement constraint equations*

   Key words : " No. of Displacement Constraint Equations :="

   - If there is no displacement constraint equation, default number "0" must be written.

10. Symbols of displacement constraint equation and equations are written.

    - The number of displacement constraint equations must be same with symbols of the displacement constraint equation
    - Generally the symbol of displacement constraint equation is "phi"

11. A number of *flow constraint equations*

    Key words : " No. of Flow Constraint Equations :="

    - If there is no flow constraint equation, default number "0" must be written.

12. Symbols of flow constraint equation and equations are written.

    - The number of displacement constraint equations must be same with symbols of the displacement constraint equation
    - Generally the symbol of flow constraint equation is "psi"

13. A number of *Effort constraint equations*

    Key words : " No. of Effort Constraint Equations :="

    - If there is no Effort constraint equation, default number "0" must be written.

14. Symbols of effort constraint equation and equations are written.

    - The number of displacement constraint equations must be same with symbols of the displacement constraint equation
    - Generally the symbol of effort constraint equation is "gamma"

Following items are caution for making input file.

- Don't use same symbol in same input.
- Don't use pre-defined symbol, such as KE, PE, D, lambda, kappa, t, rhs, lhs, cons, and eq.
- Don't miss 'Key Word'
- Don't change the input file name.

Appendix B

DERIVATION OF EXAMPLES BY HANDS

B.1 Electrical System

This system require the *Lagrange's Equation* with displacement and flow constraint equation (2.41) because there are two constraint equations. Recall that

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} + \sum_{j=1}^{m_1} \frac{\partial \phi_j}{\partial q_i} \cdot \lambda_j + \sum_{j=1}^{m_2} \frac{\partial \psi_j}{\partial \dot{q}_i} \cdot \kappa_j = e_i^a \tag{B.1}$$

where ϕ_j is the function of displacement constraint(=0, j=1,2, $\cdots$, m_1),

ψ_j is the function of flow constraint(= $0, j = 1, 2, \cdots, m_1$),

λ_j's and κ_j's are the Lagrange's multiplier, and i = 1,2,$\cdots$,n

Let us think each energy term, which will be an important part of input file in program.

$$T^* = \frac{1}{2}L(x_2)\dot{q}_2^2 = \frac{1}{2}(L_0 + bx_2)\dot{q}_2^2 \tag{B.2}$$

$$\begin{aligned} V &= \frac{1}{2}kx_1^2 + \frac{1}{2}kx_2^2 + \frac{1}{2}\frac{q_0^2}{c(x_2)} + \frac{1}{2}\frac{q_1^2}{c_1} \\ &= \frac{1}{2}kx_1^2 + \frac{1}{2}kx_2^2 + \frac{1}{2}\frac{q_0^2(d - x_1)}{c_0} + \frac{1}{2}\frac{q_1^2}{c_1} \end{aligned} \tag{B.3}$$

$$D = \frac{1}{2}R\dot{q}_0^2 \tag{B.4}$$

$$\delta W = v\delta q_0 \tag{B.5}$$

To apply the equation (B.1), $\frac{\partial T^*}{\partial \dot{x}_i}$ must be calculated such that

$$\frac{\partial T^*}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial T^*}{\partial \dot{x}_1} = & 0 \\ \frac{\partial T^*}{\partial \dot{x}_2} = & 0 \\ \frac{\partial T^*}{\partial \dot{q}_0} = & 0 \\ \frac{\partial T^*}{\partial \dot{q}_1} = & 0 \\ \frac{\partial T^*}{\partial \dot{q}_2} = & (L_0 + bx_2)\dot{q}_2 \end{cases}$$

From the previous equations, $\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_i}$ term can be calculated such that

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_1} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_2} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_0} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_1} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_2} = & (L_0 + bx_2)\ddot{q}_2 + b\dot{x}_2\dot{q}_2 \end{cases} \tag{B.6}$$

In this example, $\frac{\partial T^*}{\partial x_i}$ exits because kinetic co-energy term include x_2 term.

$$\frac{\partial T^*}{\partial x_i} \Rightarrow \begin{cases} \frac{\partial T^*}{\partial x_1} = & 0 \\ \frac{\partial T^*}{\partial x_2} = & \frac{b}{2}\dot{q}_2^2 \\ \frac{\partial T^*}{\partial q_0} = & 0 \\ \frac{\partial T^*}{\partial q_1} = & 0 \\ \frac{\partial T^*}{\partial q_2} = & 0 \end{cases} \tag{B.7}$$

The potential energy terms ($\frac{\partial V}{\partial x_i}$) are able to expressed such that

$$\frac{\partial V}{\partial x_i} \Rightarrow \begin{cases} \frac{\partial V}{\partial x_1} = & -\frac{q_0^2}{2c_0} \\ \frac{\partial V}{\partial x_2} = & kx_2 \\ \frac{\partial V}{\partial q_0} = & \frac{q_0(d-x_1)}{c_0} \\ \frac{\partial V}{\partial q_1} = & \frac{q_1}{c_1} \\ \frac{\partial V}{\partial q_2} = & 0 \end{cases} \tag{B.8}$$

To calculate the dissipation function terms ($\frac{\partial D}{\partial \dot{x}_i}$)

$$\frac{\partial D}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial D}{\partial \dot{x}_1} = & 0 \\ \frac{\partial D}{\partial \dot{x}_2} = & 0 \\ \frac{\partial D}{\partial \dot{q}_0} = & R\dot{q}_0 \\ \frac{\partial D}{\partial \dot{q}_1} = & 0 \\ \frac{\partial D}{\partial \dot{q}_2} = & 0 \end{cases} \tag{B.9}$$

Finally, using the equation (4.39) and (4.40), the constraint equations' terms are calculated such that

$$\sum_{j=1}^{1} \frac{\partial \phi_1}{\partial x_i} \lambda_j \Rightarrow \begin{cases} \frac{\partial \phi}{\partial x_1}\lambda = & \lambda \\ \frac{\partial \phi}{\partial x_2}\lambda = & \lambda \\ \frac{\partial \phi}{\partial q_0}\lambda = & 0 \\ \frac{\partial \phi}{\partial q_1}\lambda = & 0 \\ \frac{\partial \phi}{\partial q_2}\lambda = & 0 \end{cases} \tag{B.10}$$

$$\sum_{j=1}^{1} \frac{\partial \phi_1}{\partial x_i} \kappa_j \Rightarrow \begin{cases} \frac{\partial \psi}{\partial x_1}\kappa = & 0 \\ \frac{\partial \psi}{\partial x_2}\kappa = & 0 \\ \frac{\partial \psi}{\partial q_0}\kappa = & \kappa \\ \frac{\partial \psi}{\partial q_1}\kappa = & -\kappa \\ \frac{\partial \psi}{\partial q_2}\kappa = & -\kappa \end{cases} \tag{B.11}$$

Put together equations (B.6), (B.7), (B.8), (B.9), (B.10), and (B.11), the equations of motion of this example can be expressed such that

$$kx_1 - \frac{q_0^2}{2c_0} + \lambda = 0 \tag{B.12}$$

$$-\frac{b}{2}\dot{q}_2^2 + kx_2 + \lambda = 0 \tag{B.13}$$

$$\frac{q_0(d - x_1)}{c_0} + R\dot{q}_0 + \kappa = v \tag{B.14}$$

$$\frac{q_1}{c_1} - \kappa = 0 \tag{B.15}$$

$$(l_0 + bx_2)\ddot{q}_2 + b\dot{x}_2\dot{q}_2 - \kappa = 0 \tag{B.16}$$

$$\phi = x_1 + x_2 = 0 \tag{B.17}$$

$$\kappa = \dot{q}_0 - \dot{q}_1 - \dot{q}_2 = 0 \tag{B.18}$$

Following table (B.1) shows the Comparison between the results by hands and by program.

Table B.1: Comparison two result for electrical system

Coordinate	Equations of Motion
x_1	*Hand* : $kx_1 - \frac{q_0^2}{2c_0} + \lambda = 0$
	program : k*x1-0.5*q0^(2)*c0^(-1)+lambda1=0
x_2	*Hand* : $-\frac{b}{2}\dot{q}_2^2 + kx_2 + \lambda = 0$
	program : -0.5*b*dotq2^(2)+k*x2+lambda1=0
q_0	*Hand* : $\frac{q_0(d-x_1)}{c_0} + R\dot{q}_0 + \kappa = v$
	program : q0*d*c0^(-1)-q0*x1*c0^(-1)+R+dotq0-v+kappa1=0
q_1	*Hand* : $\frac{q_1}{c_1} - \kappa = 0$
	program : q1*c1^(-1)-kappa1=0
q_2	*Hand* : $(l_0 + bx_2)\ddot{q}_2 + b\dot{x}_2\dot{q}_2 - \kappa = 0$
	program : L0*ddotq2+b*dotx2*dotq2+b*x2*ddotq2-kappa1=0

B.2 Thermal System with Entropy

The generalized *Lagrange's Equation* will be used for getting equations of motion of the example of thermal system with electrical system such that

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} + \sum_{j=1}^{m_1}\frac{\partial \phi_j}{\partial q_i}\cdot\lambda_j + \sum_{j=1}^{m_2}\frac{\partial \psi_j}{\partial \dot{q}_i}\cdot\kappa_j = e_i^a(e_k) \qquad \text{(B.19)}$$

where ϕ_j is the function of displacement constraint(=0, j = 1,2,$\cdots$, m_1),

ψ_j is the function of flow constraint(=0, j = 1,2,$\cdots$, m_2),

Γ_j is the function of efforts constraint(=0, j = 1,2,$\cdots$, m_3),

λ_j's and κ_j's Γ_j's are the Lagrange's multiplier, and i = 1,2,$\cdots$, n

Using equations of energy terms (4.51), (4.52), and (4.53) with equation (B.19), each terms of *Lagrange's Equation* can be calculated like following equations.

$\frac{\partial T^*}{\partial \dot{x}_i}$ must be calculated such that

$$\frac{\partial T^*}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial T^*}{\partial \dot{V}_1} = & 0 \\ \frac{\partial T^*}{\partial \dot{V}_2} = & 0 \\ \frac{\partial T^*}{\partial \dot{V}_3} = & 0 \\ \frac{\partial T^*}{\partial \dot{S}_1} = & 0 \\ \frac{\partial T^*}{\partial \dot{S}_2} = & 0 \\ \frac{\partial T^*}{\partial \dot{S}_3} = & 0 \\ \frac{\partial T^*}{\partial \dot{q}_0} = & 0 \\ \frac{\partial T^*}{\partial \dot{\theta}} = & I\dot{\theta} \end{cases} \tag{B.20}$$

From equations (B.20, $\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_i}$ term can be calculated such that

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial T^*}{\partial \dot{V}_1} = & 0 \\ \frac{\partial T^*}{\partial \dot{V}_2} = & 0 \\ \frac{\partial T^*}{\partial \dot{V}_3} = & 0 \\ \frac{\partial T^*}{\partial \dot{S}_1} = & 0 \\ \frac{\partial T^*}{\partial \dot{S}_2} = & 0 \\ \frac{\partial T^*}{\partial \dot{S}_3} = & 0 \\ \frac{\partial T^*}{\partial \dot{q}_0} = & 0 \\ \frac{\partial T^*}{\partial \dot{\theta}} = & I\ddot{\theta} \end{cases} \tag{B.21}$$

The second terms in equation (B.19) - $\frac{\partial T^*}{\partial x_i}$ - does not exit because kinetic co-energy term does not include displacement coordinates term. The third terms - potential energy

terms ($\frac{\partial V}{\partial x_i}$) - are able to expressed such that

$$\frac{\partial V}{\partial x_i} \Rightarrow \begin{cases} \frac{\partial V}{\partial V_1} = & 0 \\ \frac{\partial V}{\partial V_2} = & 0 \\ \frac{\partial V}{\partial V_3} = & \frac{V_3}{c_f} \\ \frac{\partial V}{\partial S_1} = & 0 \\ \frac{\partial V}{\partial S_2} = & 0 \\ \frac{\partial V}{\partial S_3} = & \frac{S_3}{c_t} \\ \frac{\partial V}{\partial q_0} = & 0 \\ \frac{\partial V}{\partial \theta} = & 0 \end{cases} \tag{B.22}$$

To calculate the dissipation function terms ($\frac{\partial D}{\partial \dot{x}_i}$)

$$\frac{\partial D}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial D}{\partial \dot{V}_1} = & 0 \\ \frac{\partial D}{\partial \dot{V}_2} = & 0 \\ \frac{\partial D}{\partial \dot{V}_3} = & 0 \\ \frac{\partial D}{\partial \dot{S}_1} = & R_{t1}\dot{S}_1 \\ \frac{\partial D}{\partial \dot{S}_2} = & R_{t2}\dot{S}_2 \\ \frac{\partial D}{\partial \dot{S}_3} = & 0 \\ \frac{\partial D}{\partial \dot{q}_0} = & R_e\dot{q}_0 \\ \frac{\partial D}{\partial \dot{\theta}} = & 0 \end{cases} \tag{B.23}$$

Also, constraint equations (4.48) and (4.49) are used such that

$$\sum_{j=1}^{2} \frac{\partial \phi_j}{\partial \dot{x}_i}\kappa_j \Rightarrow \begin{cases} \frac{\partial \psi_1}{\partial \dot{V}_1}\kappa_1 + \frac{\partial \phi_2}{\partial \dot{V}_1}\kappa_2 = & -\kappa_2 \\ \frac{\partial \psi_1}{\partial \dot{V}_2}\kappa_1 + \frac{\partial \phi_2}{\partial \dot{V}_2}\kappa_2 = & -\kappa_2 \\ \frac{\partial \psi_1}{\partial \dot{V}_3}\kappa_1 + \frac{\partial \phi_2}{\partial \dot{V}_3}\kappa_2 = & \kappa_2 \\ \frac{\partial \psi_1}{\partial \dot{S}_1}\kappa_1 + \frac{\partial \phi_2}{\partial \dot{S}_1}\kappa_2 = & -\kappa_1 \\ \frac{\partial \psi_1}{\partial \dot{S}_2}\kappa_1 + \frac{\partial \phi_2}{\partial \dot{S}_2}\kappa_2 = & -\kappa_1 \\ \frac{\partial \psi_1}{\partial \dot{S}_3}\kappa_1 + \frac{\partial \phi_2}{\partial \dot{S}_3}\kappa_2 = & \kappa_1 \\ \frac{\partial \psi_1}{\partial \dot{q}_0}\kappa_1 + \frac{\partial \phi_2}{\partial \dot{q}_0}\kappa_2 = & 0 \\ \frac{\partial \psi_1}{\partial \dot{\theta}}\kappa_1 + \frac{\partial \phi_2}{\partial \dot{\theta}}\kappa_2 = & 0 \end{cases} \tag{B.24}$$

Finally, equations of motion of this example can be obtained using equations (B.21), (B.22), (B.23), (B.24), and (4.50) such that

$$-\kappa_2 = 0 \tag{B.25}$$

$$-\kappa_2 = 0 \tag{B.26}$$

$$\frac{V_3}{c_f} + \kappa_2 = 0 \tag{B.27}$$

$$R_{t1}\dot{S}_1 - \kappa_1 = 0 \tag{B.28}$$

$$R_{t2}\dot{S}_2 - \kappa_1 = 0 \tag{B.29}$$

$$\frac{S_3}{c_t} + \kappa_1 = T \tag{B.30}$$

$$R_e\dot{q}_0 = v \tag{B.31}$$

$$I\ddot{\theta} = 0 \tag{B.32}$$

Table B.2: Comparison two result for thermal system

Coordinate	Equations of Motion	Coordinate	Equations of Motion
V_1	*Hand* : $-\kappa_2 = 0$ *program* : -kappa2=0	S_1	*Hand* : $R_{t1}\dot{S}_1 - \kappa_1 = 0$ *program* : Rt1*dotS1-kappa1=0
V_2	*Hand* : $-\kappa_2 = 0$ *program* : -kappa2=0	S_2	*Hand* : $R_{t2}\dot{S}_2 - \kappa_1 = 0$ *program* : Rt2*dotS2-kappa1=0
V_3	*Hand* : $\frac{V_3}{c_f} + \kappa_2 = 0$ *program* : V3*cf(-1)+kappa2=0	S_3	*Hand* : $\frac{S_3}{c_t} + \kappa_1 = T$ *program* : S3*ct(-1)-T+kappa1=0
q_0	*Hand* : $R_e\dot{q}_0 = v$ *program* : Re*dotq0-v=0	θ	*Hand* : $I\ddot{\theta} = 0$ *program* : I*ddottheta=0

The table (B.2) shows the Comparison between the results by hands and by program. Of course, three effort constraint equations must be included in the solution.

B.3 Fluid Mechanics System with Compressive Fluid

The generalized *Lagrange's Equation* will be used for getting equations of motion of the example of fluid mechanics. Recall the equation (2.42)

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} + \sum_{j=1}^{m_1} \frac{\partial \phi_j}{\partial q_i} \cdot \lambda_j + \sum_{j=1}^{m_2} \frac{\partial \psi_j}{\partial \dot{q}_i} \cdot \kappa_j = e_i^a(e_k) \qquad \text{(B.33)}$$

where ϕ_j is the function of displacement constraint(=0, j = 1,2,$\cdots$, m_1),

ψ_j is the function of flow constraint(=0, j = 1,2,$\cdots$, m_2),

Γ_j is the function of efforts constraint(=0, j = 1,2,$\cdots$, m_3),

λ_j's and κ_j's Γ_j's are the Lagrange's multiplier, and i = 1,2,$\cdots$, n

Using equations of energy terms (4.73), (4.74), and (4.75) with equation (B.33), each terms of *Lagrange's Equation* can be calculated like following equations.

First of all, $\frac{\partial T^*}{\partial \dot{x}_i}$ must be calculated such that

$$\frac{\partial T^*}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial T^*}{\partial \dot{u}} = & 0 \\ \frac{\partial T^*}{\partial \dot{y}} = & 0 \\ \frac{\partial T^*}{\partial \dot{z}} = & m\dot{z} \\ \frac{\partial T^*}{\partial \dot{\theta}} = & I\dot{\theta} \\ \frac{\partial T^*}{\partial \dot{V_1}} = & 0 \\ \frac{\partial T^*}{\partial \dot{V_2}} = & 0 \\ \frac{\partial T^*}{\partial \dot{V_3}} = & 0 \\ \frac{\partial T^*}{\partial \dot{q}_0} = & L\dot{q}_0 \end{cases}$$

From the above equations, $\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_i}$ term can be calculated such that

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{d}{dt}\frac{\partial T^*}{\partial \dot{u}} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{y}} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{z}} = & m\ddot{z} \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{\theta}} = & I\ddot{\theta} \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{V_1}} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{V_2}} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{V_3}} = & 0 \\ \frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_0} = & L\ddot{q}_0 \end{cases} \tag{B.34}$$

$\frac{\partial T^*}{\partial x_i}$ does not exit because kinetic energy term does not include x_2 term. The potential energy terms ($\frac{\partial V}{\partial x_i}$) are able to expressed such that

$$\frac{\partial V}{\partial x_i} \Rightarrow \begin{cases} \frac{\partial V}{\partial u} = & 0 \\ \frac{\partial V}{\partial y} = & 0 \\ \frac{\partial V}{\partial z} = & k_s z \\ \frac{\partial V}{\partial \theta} = & 0 \\ \frac{\partial V}{\partial V_1} = & 0 \\ \frac{\partial V}{\partial V_2} = & 0 \\ \frac{\partial V}{\partial V_3} = & 0 \\ \frac{\partial V}{\partial q_0} = & 0 \end{cases} \tag{B.35}$$

To calculate the dissipation function terms ($\frac{\partial D}{\partial \dot{x}_i}$)

$$\frac{\partial D}{\partial \dot{x}_i} \Rightarrow \begin{cases} \frac{\partial D}{\partial \dot{u}} = & 0 \\ \frac{\partial D}{\partial \dot{y}} = & 0 \\ \frac{\partial D}{\partial \dot{z}} = & c\dot{z} \\ \frac{\partial D}{\partial \dot{\theta}} = & 0 \\ \frac{\partial D}{\partial \dot{V_1}} = & 0 \\ \frac{\partial D}{\partial \dot{V_2}} = & 0 \\ \frac{\partial D}{\partial \dot{V_3}} = & 0 \\ \frac{\partial D}{\partial \dot{q}_0} = & R\dot{q}_0 \end{cases} \tag{B.36}$$

Also, constraint equations (4.67), (4.68), and (4.55) are used such that

$$\sum_{j=1}^{2} \frac{\partial \phi_j}{\partial x_i} \lambda_j \Rightarrow \begin{cases} \frac{\partial \phi_1}{\partial u}\lambda_1 + \frac{\partial \phi_2}{\partial u}\lambda_2 = & \frac{b}{a+b}\lambda_1 \\ \frac{\partial \phi_1}{\partial y}\lambda_1 + \frac{\partial \phi_2}{\partial y}\lambda_2 = & \lambda_1 \\ \frac{\partial \phi_1}{\partial z}\lambda_1 + \frac{\partial \phi_2}{\partial z}\lambda_2 = & \lambda_2 - \frac{a}{a+b}\lambda_1 \\ \frac{\partial \phi_1}{\partial \theta}\lambda_1 + \frac{\partial \phi_2}{\partial \theta}\lambda_2 = & r\lambda_2 \\ \frac{\partial \phi_1}{\partial V_1}\lambda_1 + \frac{\partial \phi_2}{\partial V_1}\lambda_2 = & 0 \\ \frac{\partial \phi_1}{\partial V_2}\lambda_1 + \frac{\partial \phi_2}{\partial V_2}\lambda_2 = & 0 \\ \frac{\partial \phi_1}{\partial V_3}\lambda_1 + \frac{\partial \phi_2}{\partial V_3}\lambda_2 = & 0 \\ \frac{\partial \phi_1}{\partial q_0}\lambda_1 + \frac{\partial \phi_2}{\partial q_0}\lambda_2 = & 0 \end{cases} \tag{B.37}$$

$$\sum_{j=1}^{1} \frac{\partial \psi_j}{\partial \dot{x}_i} \kappa_j \Rightarrow \begin{cases} \frac{\partial \psi_1}{\partial \dot{u}}\kappa_1 = & 0 \\ \frac{\partial \psi_1}{\partial \dot{y}}\kappa_1 = & 0 \\ \frac{\partial \psi_1}{\partial \dot{z}}\kappa_1 = & 0 \\ \frac{\partial \psi_1}{\partial \dot{\theta}}\kappa_1 = & 0 \\ \frac{\partial \psi_1}{\partial V_1}\kappa_1 = & \kappa_1 \\ \frac{\partial \psi_1}{\partial V_2}\kappa_1 = & -\kappa_1 \\ \frac{\partial \psi_1}{\partial V_3}\kappa_1 = & -\kappa_1 \\ \frac{\partial \psi_1}{\partial \dot{q}_0}\kappa_1 = & 0 \end{cases} \tag{B.38}$$

Finally, equations of motion of this example can be obtained using equations (B.34), (B.35), (B.36), (B.37), and (B.38) such that

$$\frac{b}{a+b}\lambda_1 = F \tag{B.39}$$

$$\lambda_1 = 0 \tag{B.40}$$

$$m\ddot{z} + k_s z + c\dot{z} + \lambda_2 - \frac{a}{a+b}\lambda_1 = -A_p(P_2 - P_{atm}) - mg \tag{B.41}$$

$$I\ddot{\theta} + r\lambda_2 = 0 \tag{B.42}$$

$$\kappa_1 = P_s \tag{B.43}$$

$$-\kappa_1 = P_2 \tag{B.44}$$

$$-\kappa_1 = P_2 \tag{B.45}$$

$$L\ddot{q}_0 + R\dot{q}_0 = v \tag{B.46}$$

Following table (B.3) shows the Comparison between the results by hands and by program. Of course, three effort constraint equations must be included in the solution.

Table B.3: Comparison two result for fluid mechanics system

Coordinate	Equations of Motion
u	*Hand* : $\frac{b}{a+b}\lambda_1 = F$
	program : -F+b*(a+b)^(-1)*lambda1=0
y	*Hand* : $\lambda_1 = 0$
	program : lambda1=0
z	*Hand* : $m\ddot{z} + k_s z + c\dot{z} + \lambda_2 - \frac{a}{a+b}\lambda_1 = -A_p(P_2 - P_{atm}) - mg$
	program : m*ddotz+ks*z+c*dotz+Ap*P2-Ap*P2+m*g-a*(a+b)^(-1)*lambda1+lambda2=0
θ	*Hand* : $I\ddot{\theta} + r\lambda_2 = 0$
	program : I*ddottheta+r*lambda2=0
V_1	*Hand* : $\kappa_1 = P_s$
	program ::-Ps+kappa1=0
V_2	*Hand* : $-\kappa_1 = P_2$
	program : P2-kappa1=0
V_3	*Hand* : $-\kappa_1 = P_2$
	program : P2-kappa1=0
q_0	*Hand* : $L\ddot{q}_0 + R\dot{q}_0 = v$
	program : L*ddotq0+R*dotq0-v=0

Appendix C

CODES

This header files [1] are making the symbolic object to make the input files and set up the equations of motion of each system. Anybody can see the header files in ftp://abs-5.me.wahsington.edu/jsuk.[2] However, simple codes for reading the input file are here.[3]

```
#include<iostream.h> // To use basic input, output command
#include<fstream.h> // To use file input command
#include<ostream.h> // To use file output command
#include<ctype.h> // To use character function command
#include<math.h> // To use the math character like sin, cos, etc.
#include<stdio.h> // To use the c command for input, output
#include<stdlib.h> #include<string.h> #include<float.h>

// To define several useful number
#define MAX_LENGTH 512
#define KEYWORD 30
#define VARIABLE 25

// To convert character to number(use ascii codes)
int trans(char x[3]) {
    int X;
    for(int i = 0 ; i < 3 ; i ++)
    {
        if (isalpha(x[i]) || ispunct(x[i]) )
```

[1] This parts are modified from the header files in Hardy, Shi, and Steeb [33]

[2] These header files are too long to write here. Just check in the 'ftp'

[3] More detail explanation have attached

```
        {
            // alert the errors
            cout << "Because of " ;
            cout << x[i] ;
            cout << ", your inputs are wrong!" << endl;
        }
    }

    if (strlen(x) == 3)
    {
        X = (x[0] -48)*100 + (x[1] -48)*10 + (x[2]-48);
        return X;
    }
    else if (strlen(x) == 2)
    {
        X = (x[0] -48)*10 + (x[1] -48);
        return X;
    }
    else (strlen(x) == 1) ;
    {
        X = x[0]-48;
        return X;
    }
}

void main() {
    cout << "-------------------------------------------------------------
    ----------" << endl;
    cout << "| The purpose of this program is to set up the equations of mo
    tion.    |" << endl;
    cout << "| If you want to use this program, you need the knowledge of l
    agrange  |" << endl;
    cout << "| equation. You can keep you following manual, you can easily
```

```
get the  |" << endl;
cout << "| equations of motion.
         |" << endl;
cout << "|                                          Developed by Jae Hyeuk
 Suk     |" << endl;
cout << "-----------------------------------------------------------------
--------" << endl << endl;

cout << "----------------------File Name--------------------------" <<
endl;
cout << "| First of all, you must open the \"LagEq.txt\" to        |" <<
endl;
cout << "| input. You can choose the name of input txt file which|" <<
endl;
cout << "| must include \".txt\". And, your result will be recorded|" <<
endl;
cout << "| dos command which you can hold.                        |" <<
endl;
cout << "-----------------------------------------------------------" <<
endl << endl;

// To open the input file
ifstream fin;
fin.open( "LagEq.txt" , ios::in | ios::nocreate);

// To opent the output file -> To create new cpp file
ofstream fout("test.cpp");

fout << "#include<iostream.h>" << endl;
fout << "#include<fstream.h>" << endl;
fout << "#include<stdlib.h>" << endl;
fout << "#include<math.h>" << endl;
fout << "#include<stdio.h>" << endl;
```

```
fout << "#include<string.h>" << endl;
fout << "#include \"MSymbol.h\"" << endl;
fout << "#include \"Matrix.h\"" << endl << endl;
fout << "void main()" << endl;
fout << "{" << endl;

// Basic symbols -> don't use these symbol in your program!
fout << "Sum<double> KE(\"KE\",0), PE(\"PE\",0) ,D(\"D\",0), t(\"t\",0);"
 << endl;

// Constant Variables
char con[MAX_LENGTH], n_const[3];
int no_const;
fin.getline(con,MAX_LENGTH, '=');
if(strcmp(con,"No. of Constant Variables :"))
{
    cout << "Input Error during inputting constant variables!" << endl;
}
else
{
    fin.getline(n_const, MAX_LENGTH, '\n');
    no_const = trans(n_const);                  //Convert 'char' -> 'int'
    char consta[50][VARIABLE], var[50][VARIABLE];
    for(int a = 1; a < no_const+1; a++)
    {
        fin.getline(consta[a], VARIABLE, '=');
        fout << "Sum<double> " << consta[a]<< "(\"" << consta[a] << "\",0)
        ;" << endl;
        fin.getline(var[a], VARIABLE, '\n');
        if(strcmp(var[a], "0")) {
            fout << "double " << consta[a] << a << a << "=" << var[a] <<
            ";" << endl;
            fout << consta[a] << ".set(" << consta[a] << a << a << "); "
```

```
            << endl;
            fout << consta[a] << "=" << consta[a] << a << a << ";" << endl;
        }
    }
}

// Displacement Coordinates
char d_codi_tag[MAX_LENGTH], n_codi[3];
char d_codi[30][VARIABLE];
int no_codi;
fin.getline(d_codi_tag,MAX_LENGTH, '=');
if(strcmp(d_codi_tag,"No. of Displacement Coordinates :"))
{
    cout << "Input Error during inputting Displacement Coordinates!" << endl;
}
else
{
    fin.getline(n_codi, MAX_LENGTH, '\n');
    no_codi = trans(n_codi);                 //Convert 'char' -> 'int'
    int b = 1;
    while(b < no_codi+1)
    {
        if(b < no_codi)
        {
            fin.getline(d_codi[b], VARIABLE, ',');
            fout << "Sum<double> " << d_codi[b] <<"(\"" << d_codi[b] <<
            "\",0), rhs" << b;
            fout << "(\"rhs" << b << "\",0), lhs" << b << "(\"lhs" << b <<
             "\",0), cons";
            fout << b << "(\"cons" << b << "\",0), eq" << b << "(\"eq" << b
            << "\",0);" <<endl;
            fout << "cons" << b << ".depend(" << d_codi[b] << ");" << endl;
            fout << "KE.depend(" << d_codi[b] << ");" << endl;
```

```
            fout << "PE.depend(" << d_codi[b] << ");" << endl;
            fout << "D.depend(" << d_codi[b] << ");" << endl;
            fout << d_codi[b] << ".depend(t); "<< endl;
        }
        else if (b = no_codi)
        {
            fin.getline(d_codi[b], VARIABLE, '\n');
            fout << "Sum<double> " << d_codi[b] <<"(\"" << d_codi[b] <<
            "\",0), rhs" << b;
            fout << "(\"rhs" << b << "\",0), lhs" << b << "(\"lhs" << b <<
            "\",0), cons";
            fout << b << "(\"cons" << b << "\",0), eq" << b << "(\"eq" << b
            << "\",0);" <<endl ;
            fout << "cons" << b << ".depend(" << d_codi[b] << ");" << endl;
            fout << "KE.depend(" << d_codi[b] << ");" << endl;
            fout << "PE.depend(" << d_codi[b] << ");" << endl;
            fout << "D.depend(" << d_codi[b] << ");" << endl;
            fout << d_codi[b] << ".depend(t); "<< endl;
        }
        b++;
    }
}

// Flow Coordinates
char f_codi_tag[MAX_LENGTH], fn_codi[3];
char f_codi[30][VARIABLE];
int fno_codi;
fin.getline(f_codi_tag,MAX_LENGTH, '=');
if(strcmp(f_codi_tag,"No. of Flow Coordinates :"))
{
    cout << "Input Error during inputting Flow Coordinates!" << endl;
}
else
```

```
{
    fin.getline(fn_codi, MAX_LENGTH, '\n');
    fno_codi = trans(fn_codi);                  //Convert 'char' -> 'int'
    int c = 1;
    while(c < no_codi+1)
    {
        if(c < no_codi)
        {
            fin.getline(f_codi[c], VARIABLE, ',');
            fout << "Sum<double> " << f_codi[c] <<"(\"" << f_codi[c] <<
            "\",0), d" << f_codi[c];
            fout << "(\"d" << f_codi[c] << "\",0);" << endl;
            fout << "cons" << c << ".depend(" << f_codi[c] << ");" << endl;
            fout << f_codi[c] <<".depend(t);" << endl;
            fout << "KE.depend(" << f_codi[c] << ");" << endl;
            fout << "PE.depend(" << f_codi[c] << ");" << endl;
            fout << "D.depend(" << f_codi[c] << ");" << endl;
            fout << f_codi[c] << ".depend(t); "<< endl;
        }
        else if (c = fno_codi)
        {
            fin.getline(f_codi[c], VARIABLE, '\n');
            fout << "Sum<double> " << f_codi[c] <<"(\"" << f_codi[c] <<
            "\",0), d" << f_codi[c];
            fout << "(\"d" << f_codi[c] << "\",0);" << endl;
            fout << "cons" << c << ".depend(" << f_codi[c] << ");" << endl;
            fout << f_codi[c] <<".depend(t);" << endl;
            fout << "KE.depend(" << f_codi[c] << ");" << endl;
            fout << "PE.depend(" << f_codi[c] << ");" << endl;
            fout << "D.depend(" << f_codi[c] << ");" << endl;
            fout << f_codi[c] << ".depend(t); "<< endl;
        }
        c++;
```

```
        }
    }

    // Energy Terms
    // Kinetic Co-energy
    char k[MAX_LENGTH], KE[MAX_LENGTH];
    fin.getline(k, MAX_LENGTH,'=');
    if(strcmp(k,"Kinetic Coenergy :"))
    {
        cout << "Input Error during inputting Kinetic Coenergy!" << endl;
    }
    else
    {
        fin.getline(KE,MAX_LENGTH,'\n');
        fout << "KE = " << KE << ";" << endl;
    }

    // Potential Energy
    char p[MAX_LENGTH], PE[MAX_LENGTH];
    fin.getline(p, MAX_LENGTH,'=');
    if(strcmp(p,"Potential Energy :"))
    {
        cout << "Input Error during inputting Potential Energy!" << endl;
    }
    else
    {
        fin.getline(PE,MAX_LENGTH,'\n');
        fout << "PE = " << PE << ";" << endl;
    }

    // Dissipation Function
    char d[MAX_LENGTH], D[MAX_LENGTH];
    fin.getline(d, MAX_LENGTH,'=');
```

```
if(strcmp(d,"Dissipation Function :"))
{
    cout << "Input Error during inputting Dissipation Function!" << endl;
}
else
{
    fin.getline(D,MAX_LENGTH,'\n');
    fout << "D = " << D << ";" << endl;
}

// Work corresponding to each coordinates

char w_tag[MAX_LENGTH];
char w[30][VARIABLE], w_var[30][MAX_LENGTH];
fin.getline(w_tag, MAX_LENGTH, '\n');
if(strcmp(w_tag,"Virtual Work :"))
{
    cout << "Input Error during inputting Virtual Work!" << endl;
}
else
{
    for(int d = 1 ; d < no_codi+1 ; d++)
    {
        fin.getline(w[d],MAX_LENGTH,'=');
        fin.getline(w_var[d],MAX_LENGTH,'\n');
        if(!strcmp(w_var[d],"+0"))
        {
            strcpy(w_var[d]," ");
        }

    }
}
```

```
// Constraint Equations
// Displacement Constraint
char dis_cons_tag[MAX_LENGTH], n_dis_cons[3];
char dis_cons[30][VARIABLE], dis_cons_var[30][MAX_LENGTH];
int no_dis_cons;
fin.getline(dis_cons_tag, MAX_LENGTH, '=');
if(strcmp(dis_cons_tag,"No. of Displacement Constraint Equations :"))
{
    cout << "Input Error during inputting Displacement Constraint Equation!"
    << endl;
}
else
{
    fin.getline(n_dis_cons, MAX_LENGTH, '\n');
    no_dis_cons = trans(n_dis_cons);
    if(no_dis_cons != 0)
    {
        for (int e = 1 ; e < no_dis_cons+1; e++)
        {
            fin.getline(dis_cons[e], MAX_LENGTH, '=');
            fout << "Sum<double> " << dis_cons[e] <<"(\"" << dis_cons[e] <<
            "\",0), lambda";
            fout << e << "(\"lambda" << e << "\",0) ;" << endl;
            for (int ee = 1; ee < no_codi+1 ; ee++)
            {
                fout << dis_cons[e] << ".depend(" << d_codi[ee] << "); "<<
                endl;
            }
            fin.getline(dis_cons_var[e], MAX_LENGTH, '\n');
            fout << dis_cons[e] << " = " << dis_cons_var[e] << ";" << endl;
        }
    }
}
```

```
// Flow Constraint
char flw_cons_tag[MAX_LENGTH], n_flw_cons[3];
char flw_cons[30][VARIABLE], flw_cons_var[30][MAX_LENGTH];
int no_flw_cons;
fin.getline(flw_cons_tag, MAX_LENGTH, '=');
if(strcmp(flw_cons_tag,"No. of Flow Constraint Equations :"))
{
    cout << "Input Error during inputting Flow Constraint Equation!" << endl;
}
else
{
    fin.getline(n_flw_cons, MAX_LENGTH, '\n');
    no_flw_cons = trans(n_flw_cons);
    if(no_flw_cons != 0)
    {
        for (int e = 1 ; e < no_flw_cons+1; e++)
        {
            fin.getline(flw_cons[e], MAX_LENGTH, '=');
            fout << "Sum<double> " << flw_cons[e] <<"(\"" << flw_cons[e]
            << "\",0), kappa";
            fout << e << "(\"kappa" << e << "\",0) ;" << endl;
            for (int ee = 1; ee < no_codi+1 ; ee++)
            {
                fout << flw_cons[e] << ".depend(" << f_codi[ee] << "); "<<
                 endl;
            }
            fin.getline(flw_cons_var[e], MAX_LENGTH, '\n');
            fout << flw_cons[e] << " = " << flw_cons_var[e] << ";" << endl;
        }
    }
}
```

```
// Effort Constraint
char efo_cons_tag[MAX_LENGTH], n_efo_cons[3];
char efo_cons[30][VARIABLE], efo_cons_var[30][MAX_LENGTH];
int no_efo_cons;
fin.getline(efo_cons_tag, MAX_LENGTH, '=');
if(strcmp(efo_cons_tag,"No. of Effort Constraint Equations :"))
{
    cout << "Input Error during inputting Effort Constraint Equation!" <<
     endl;
}
else
{
    fin.getline(n_efo_cons, MAX_LENGTH, '\n');
    no_efo_cons = trans(n_efo_cons);
    if(no_efo_cons !=0)
    {
        for (int f = 1 ; f < no_efo_cons+1; f++)
        {
            fin.getline(efo_cons[f], MAX_LENGTH, '=');
            fout << "Sum<double> " << efo_cons[f] <<"(\"" << efo_cons[f] <<
             "\",0);" << endl;
            for (int ff = 1; ff < no_codi+1 ; ff++)
            {
                fout << efo_cons[f] << ".depend(" << f_codi[ff] << "); "<<
                 endl;
            }
            fin.getline(efo_cons_var[f], MAX_LENGTH, '\n');
            fout << efo_cons[f] << " = " << efo_cons_var[f] << ";" << endl;
        }
    }
}

int q = 1;
```

```
if(no_dis_cons+no_flw_cons !=0)
{
    while(q< no_codi +1)
    {
        fout << "cons" << q << "=";
        if(no_dis_cons > 0)
        {
            int r = 1;
            while (r < no_dis_cons+1)
            {
                if (r < no_dis_cons)
                {
                    fout << "dot(" << dis_cons_var[r] << "," << d_codi[q]
                    << ") * lambda" << r << " + " ;
                }
                else if (r = no_dis_cons)
                {
                    fout << "dot(" << dis_cons_var[r] << "," << d_codi[q]
                    << ") * lambda" << r ;
                }
                r = r+1 ;
            }
        }


        if(no_flw_cons > 0)
        {
            int s = 1;
            fout << " + " ;
            while (s < no_flw_cons+1)
            {
                if (s < no_flw_cons)
                {
```

```
                    fout << "dot(" << flw_cons_var[s] << "," << f_codi[q]
                    << ") * kappa" << s  << " + " ;
                }
                else if (s = no_flw_cons)
                {
                    fout << "dot(" << flw_cons_var[s] << "," << f_codi[q]
                    << ") * kappa" << s  << ";" << endl ;
                }
                s = s+1;
            }
        }
        else if (no_flw_cons == 0)
        {
            fout << ";" << endl;
        }
        q = q+1;

    }
}
else
{
    for(int i = 1; i < no_codi+1; i++)
    {
        fout << "cons" << i << " = 0;" << endl;
    }
}

for(int t = 1; t < no_codi +1; t++)
{
    fout << "rhs" << t << "=" << "dot(dot(KE," << f_codi[t] << "),t) ; "
    << endl;
}
```

```
for (int u = 1 ; u < no_codi +1 ; u++)
{
    fout << "lhs" << u << "=" << " dot(KE," << d_codi[u] << ") - dot(PE,"
    << d_codi[u] << ") - dot(D,";
    fout << f_codi[u] << ") " << w_var[u] << ";" << endl;
}

for (int v = 1 ; v < no_codi +1 ; v++)
{
    fout << "eq" << v << " = " << "rhs" << v << "- lhs" << v << ";" << endl;
    for(int vv = 1; vv < no_codi+1 ; vv++)
    {
        fout << "eq" << v << ".put(dot(" << d_codi[vv] << ",t)," <<
        f_codi[vv] << ");" << endl;
        fout << "eq" << v << ".put(dot(" << f_codi[vv] << ",t),d" <<
        f_codi[vv] << ");" << endl;
    }
}

// The results
fout << "cout << \"Equations of motion using Lagrange Equation :\" <<
endl;" << endl;
for(int x =1; x < no_codi +1 ; x++)
{
    fout << "cout << \"" << d_codi[x] << ":\"" << "<< eq" << x <<
    ".value() + cons" << x << "  << \"=0\" << endl;" << endl;
}

// To express the constraint equations in results.
if(no_dis_cons !=0)
{
    for(int v=1; v<no_dis_cons +1 ; v++)
    {
```

```
            fout << "cout << endl << \"phi" << v << ":\" << phi" << v <<
            "<< \"=0 \"<< endl;" << endl;
        }
    }

    if(no_flw_cons !=0)
    {
        for(int vv=1; vv<no_flw_cons +1 ; vv++)
        {
            fout << "cout << endl << \"psi" << vv << ":\" << psi" << vv <<
            "<< \"=0 \" << endl;" << endl;
        }
    }

    if(no_efo_cons !=0)
    {
        for(int w=1; w<no_efo_cons +1 ; w++)
        {
            fout << "cout << endl << \"gamma" << w << ":\" << gamma" << w <<
            "<< \"=0 \" << endl;" << endl;
        }
    }



    fout << "getchar();" << endl;
    fout << "}" << endl;
    getchar();
}
```