Real-Time Generation for Optimal Robot Motion

Graduate School of Systems and Information Engineering

University of Tsukuba

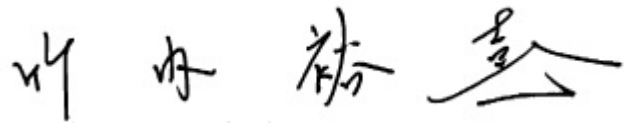
March,2005

Takeuchi Hiroki

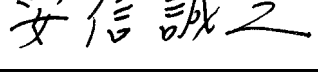
Doctor's Thesis
submitted to Graduate School of Systems and Information Engineering
University of Tsukuba
in partial fulfillment of requirements for the degree of
DOCTOR of ENGINEERING

Takeuchi Hiroki

March 2005

Author 

Takeuchi Hiroki, Graduate School of Systems and Information Engineering

Certified by 

Yasunobu Seiji, Professor, Thesis Adviser

Thesis committee:

Professor,Thesis Adviser:	Yasunobu Seiji
Professor:	Sankai Yoshiyuki
Professor:	Onisawa Takehisa
Associate Professor:	Hori Noriyuki
Associate Professor(Osaka University):	Ohtsuka Toshiyuki

Acknowledgement

The author would first like to express his sincere gratitude to warm encouragement and support of his adviser, Dr. Yasunobu Seiji in pursuing this research. The author gratefully acknowledges the support of The University of Tsukuba and staffs from The Graduate School of Systems and Information Engineering. Especially he also thanks Dr.Ohtsuka Toshiyuki. Dr.Ohtsuka did support to him when he was in Tsukuba University.

The author is also appreciative of my lovely guitars Tokai EA-25 and Taylor-510CE which have comforted him with some songs. Montgomery's book "Green Gables Anne" has given him struggling with the research great hope and various beautiful words about nature of Prince Edward Island.

Abstract

Robot movement generation requires optimization treatment. However, the need for off-line computation makes it difficult to apply traditional optimization techniques to real-time robot control. An important goal is to develop a new algorithm that allows for real-time optimization. The most likely candidate which is called as Receding Horizon Control or Model Predictive Control algorithms have yet to be widely applied to real-time robot control environments.

This thesis uses a legged robot as a control object, one that possesses unstable dynamics and requires specific balance conditions, with the Zero Moment Point balance condition being a particularly important challenge. Equal constraint, proposed in this thesis as a means for meeting such conditions during optimization formulation, overcomes Zero Moment Point problems.

The state variable inequality constraint is a complex challenge because the optimal path must tangentially enter a constrained arc, and one or more time constraint derivatives must equal zero at all entry points. A second challenge addressed in this thesis is the description of a legged robot's swing leg condition as a state variable inequality. Both the nonlinear swing leg and Zero Moment Point balance conditions are involved in Receding Horizon Control formulation.

In this thesis, improvements in the Receding Horizon Control algorithm are discussed, and a new algorithm that allows for real-time robot control is proposed. The real-time optimization techniques described in this paper can be applied to various industrial environments, including aerospace, railway, and automobile manufacturing.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Conventional Research Overview . . . 2
  - 1.2.1 Optimization Theory and Calculus History . . . 2
  - 1.2.2 Early Practical Application of Optimization Theory . . . 3
  - 1.2.3 Real-Time OS and Control . . . 4
  - 1.2.4 Formula Manipulation . . . 4
- 1.3 Objective and Approach . . . 4

2 Application of Optimization Problem to Mechanics **7**
- 2.1 Introduction . . . 7
- 2.2 Definition of Optimization Problem . . . 7
- 2.3 Equal Constraint . . . 8
- 2.4 Gradient Method . . . 9
  - 2.4.1 First Order Gradient of Performance Index . . . 9
  - 2.4.2 Various Sorts of Gradient Method . . . 9
  - 2.4.3 Example of Gradient Method . . . 10
- 2.5 Application of Gradient Method to real-time control robot and Matters . . . 12
- 2.6 Conclusion . . . 13

3 Real-Time Optimization Technique **15**
- 3.1 Introduction . . . 15
- 3.2 Homotopy Method . . . 15
- 3.3 Continuation Method . . . 15
- 3.4 Solve Optmization Problem with Stabilized Continuation Method . . . 17
  - 3.4.1 Formulation . . . 17
- 3.5 Receding Horizon Control . . . 18
- 3.6 Conclusions . . . 19

4 Real-Time Control for Robot **21**
- 4.1 Introduction . . . 21
- 4.2 Numerical Model . . . 21
  - 4.2.1 Lagrange Method . . . 21
  - 4.2.2 Newton-Euler Formulation . . . 22
  - 4.2.3 State Equations . . . 24
- 4.3 Real-Time Control using Modern Control Theory . . . 25

4.4 Singular Point . . . 26
4.5 Real-Time OS . . . 27
4.5.1 Real-Time OS . . . 27
4.5.2 Interrupts . . . 27
4.6 Application to Legged Robot Control . . . 28
4.7 Conclusions . . . 29

5 Equal Constraint for Balance Condition of Legged Robot 31
5.1 Introduction . . . 31
5.2 Receding Horizon Control with Equality Constraint . . . 34
5.3 Formulation . . . 36
5.3.1 Model Expression as a Point Mass . . . 36
5.3.2 Equality Constraints . . . 37
5.3.3 Performance Index . . . 38
5.4 Numerical Simulation . . . 38
5.4.1 Two Dimensional Formulation . . . 38
5.4.2 Three Dimensional Formulation . . . 43
5.5 Mass Behavior and ZMP . . . 48
5.5.1 In case of $x > 0$ . . . 48
5.5.2 In case of Cut Across x=0 . . . 53
5.5.3 Sudden Acceleration . . . 56
5.5.4 Sudden Stop . . . 59
5.6 Weight Matrix . . . 62
5.7 Application . . . 65
5.7.1 In the Case of a Biped . . . 65
5.7.2 In the Case of a Quadruped . . . 66
5.8 Conclusion . . . 66

6 State Variable Inequality Constraint of Swing Leg 69
6.1 Introduction . . . 69
6.2 Modeling of Swing Leg . . . 69
6.3 Performance Index . . . 71
6.4 Numerical Calculation . . . 71
6.5 Constraint for Swing Legs . . . 75
6.5.1 State Variable Constraint . . . 75
6.5.2 Slack Variable Method . . . 75
6.5.3 Performance Index . . . 77
6.5.4 Numerical Calculation(Linear Case) . . . 77
6.5.5 Numerical Calculation (Nonlinear Case) . . . 82
6.6 Conclusion . . . 88

7 Nonlinear Receding Horizon Gradient Method 89
7.1 Introduction . . . 89
7.2 Continuation Method and Gradient . . . 89
7.2.1 Continuation Method . . . 89
7.2.2 Gradient . . . 90

7.2.3 Sampling Interval 90
7.3 Simulation 92
7.3.1 Example 92
7.3.2 Nonlinear Two Link System 96
7.4 Practical Installation into Real-Time System 101
7.5 Singular Point 107
7.6 Conclusion 111

8 Conclusion and Recommendation **113**
8.1 Overall Perspective 113
8.2 Recommendations for Future Research 114

A Transition Matrix **117**

List of Figures

1.1 Main Objective 6

2.1 Two Points Boundary Value Problem 8
2.2.1 x_1(Steepest Descent) 11
2.2.2 x_2(Steepest Descent) 11
2.2.3 λ_1(Steepest Descent) 11
2.2.4 λ_2(Steepest Descent) 11
2.2.5 u_1(Steepest Descent) 11
2.2.6 Performance index value / Number of iteration (Steepest Descent) 11
2.3 Dilemma between off-line optimization and real-time control 12

3.1 Predictor-Corrector Method 16
3.2 Ill case in Predictor-Corrector Method 16
3.3 Solve Optimization Problem with Continuation Method 18
3.4 Moving Horizon 19

4.1 Force and Moment on a Link 22
4.2 Frame Relation 23
4.3 Difference between Optimal Regulator and Receding Horizon Control . . . 26
4.4 Structure of Real-Time OS 27
4.5 Interrupt driven I/O cycle 28
4.6 Control for Legged Robot 29

5.1 Definition of Zero Moment Point 32
5.2 Definition of sagittal plane and frontal plane 32
5.3 Linearization and Reduction 33
5.4 2-D Model in Sagittal Plane 37
5.5.1 x(2-D case) 40
5.5.2 z(2-D case) 40
5.5.3 $\dot{x}$(2-D case) 40
5.5.4 $\dot{z}$(2-D case) 40
5.5.5 u_x(2-D case) 40
5.5.6 u_z(2-D case) 40
5.5.7 u_{ZMP}(2-D case) 40
5.5.8 Sum of errors (2-D case) 41
5.5.9 Left side of equal constraint(2-D case) 41
5.5.10 Right side of equal constraint(2-D case) 41
5.6 3 Dimensional Formulation 43

5.7.1 x(3-D Case) . . . 46
5.7.2 y(3-D Case) . . . 46
5.7.3 z(3-D Case) . . . 46
5.7.4 $\dot{x}$(3-D Case) . . . 46
5.7.5 $\dot{y}$(3-D Case) . . . 46
5.7.6 $\dot{z}$(3-D Case) . . . 46
5.7.7 u_x(3-D Case) . . . 46
5.7.8 u_y(3-D Case) . . . 47
5.7.9 u_z(3-D Case) . . . 47
5.7.10 u_{ZMP_x}(3-D Case) . . . 47
5.7.11 u_{ZMP_y}(3-D Case) . . . 47
5.7.12 Left side of the equal constraint(3-D Case) . . . 47
5.7.13 Right side of the equal constraint(3-D Case) . . . 47
5.7.14 Sum of Error1-6(3-D Case) . . . 47
5.7.15 Mass Behavior Ahead x=0/Behind x=0 . . . 48
5.7.16 In Case of $x > 0$ . . . 49
5.8.1 x(In Case of $x > 0$) . . . 50
5.8.2 y(In Case of $x > 0$) . . . 50
5.8.3 z(In Case of $x > 0$) . . . 50
5.8.4 $\dot{x}$(In Case of $x > 0$) . . . 50
5.8.5 $\dot{y}$(In Case of $x > 0$) . . . 50
5.8.6 $\dot{z}$(In Case of $x > 0$) . . . 50
5.8.7 u_x(In Case of $x > 0$) . . . 50
5.8.8 u_y(In Case of $x > 0$) . . . 51
5.8.9 u_z(In Case of $x > 0$) . . . 51
5.8.10 u_{ZMP_x}(In Case of $x > 0$) . . . 51
5.8.11 u_{ZMP_y}(In Case of $x > 0$) . . . 51
5.9.1 x(Across $x = 0$) . . . 54
5.9.2 y(Across $x = 0$) . . . 54
5.9.3 z(Across $x = 0$) . . . 54
5.9.4 $\dot{x}$(Across $x = 0$) . . . 54
5.9.5 $\dot{y}$(Across $x = 0$) . . . 54
5.9.6 $\dot{z}$(Across $x = 0$) . . . 54
5.9.7 u_x(Across $x = 0$) . . . 54
5.9.8 u_y(Across $x = 0$) . . . 55
5.9.9 u_z(Across $x = 0$) . . . 55
5.9.10 u_{ZMP_x}(Across $x = 0$) . . . 55
5.9.11 u_{ZMP_y}(Across $x = 0$) . . . 55
5.10.1 x(Sudden Acceleration) . . . 57
5.10.2 y(Sudden Acceleration) . . . 57
5.10.3 z(Sudden Acceleration) . . . 57
5.10.4 $\dot{x}$(Sudden Acceleration) . . . 57
5.10.5 $\dot{y}$(Sudden Acceleration) . . . 57
5.10.6 $\dot{z}$(Sudden Acceleration) . . . 57
5.10.7 u_x(Sudden Acceleration) . . . 57

5.10.8 u_y(Sudden Acceleration) . . . 58
5.10.9 u_z(Sudden Acceleration) . . . 58
5.10.10 u_{ZMP_x}(Sudden Acceleration) . . . 58
5.10.11 u_{ZMP_y}(Sudden Acceleration) . . . 58
5.11.1 x(Sudden Stop) . . . 60
5.11.2 y(Sudden Stop) . . . 60
5.11.3 z(Sudden Stop) . . . 60
5.11.4 $\dot{x}$(Sudden Stop) . . . 60
5.11.5 $\dot{y}$(Sudden Stop) . . . 60
5.11.6 $\dot{z}$(Sudden Stop) . . . 60
5.11.7 u_x(Sudden Stop) . . . 60
5.11.8 u_y(Sudden Stop) . . . 61
5.11.9 u_z(Sudden Stop) . . . 61
5.11.10 u_{ZMP_x}(Sudden Stop) . . . 61
5.11.11 u_{ZMP_y}(Sudden Stop) . . . 61
5.12.1 x(Make u_{ZMP} small) . . . 63
5.12.2 y(Make u_{ZMP} small) . . . 63
5.12.3 z(Make u_{ZMP} small) . . . 63
5.12.4 $\dot{x}$(Make u_{ZMP} small) . . . 63
5.12.5 $\dot{y}$(Make u_{ZMP} small) . . . 63
5.12.6 $\dot{z}$(Make u_{ZMP} small) . . . 63
5.12.7 u_x(Make u_{ZMP} small) . . . 63
5.12.8 u_y(Make u_{ZMP} small) . . . 64
5.12.9 u_z(Make u_{ZMP} small) . . . 64
5.12.10 u_{ZMP_x}(Make u_{ZMP} small) . . . 64
5.12.11 u_{ZMP_y}(Make u_{ZMP} small) . . . 64
5.13 Case of Biped . . . 65
5.14 Control Block Diagram . . . 66
5.15 Case of Quadruped . . . 66

6.1 Image of formulization . . . 71
6.2.1 x(No Constraint Case) . . . 73
6.2.2 z(No Constraint Case) . . . 73
6.2.3 θ_1(No Constraint Case) . . . 73
6.2.4 θ_2(No Constraint Case) . . . 73
6.2.5 $\dot{x}$(No Constraint Case) . . . 73
6.2.6 $\dot{z}$(No Constraint Case) . . . 73
6.2.7 $\dot{\theta}_1$(No Constraint Case) . . . 73
6.2.8 $\dot{\theta}_2$(No Constraint Case) . . . 74
6.2.9 u_x(No Constraint Case) . . . 74
6.2.10 u_z(No Constraint Case) . . . 74
6.2.11 u_{θ_1}(No Constraint Case) . . . 74
6.2.12 u_{θ_2}(No Constraint Case) . . . 74
6.2.13 u_{ZMP}(No Constraint Case) . . . 74
6.2.14 Stick Figure(No Constraint Case) . . . 74

6.3 Parameters . . . 75
6.4.1 x(Linear Case) . . . 79
6.4.2 z(Linear Case) . . . 79
6.4.3 θ_1(Linear Case) . . . 79
6.4.4 θ_2(Linear Case) . . . 79
6.4.5 $\dot{x}$(Linear Case) . . . 79
6.4.6 $\dot{z}$(Linear Case) . . . 79
6.4.7 $\dot{\theta}_1$(Linear Case) . . . 79
6.4.8 $\dot{\theta}_2$(Linear Case) . . . 80
6.4.9 slack d(Linear Case) . . . 80
6.4.10 slack $\dot{d}$(Linear Case) . . . 80
6.4.11 u_x(Linear Case) . . . 80
6.4.12 u_z(Linear Case) . . . 80
6.4.13 u_{θ_1}(Linear Case) . . . 80
6.4.14 u_{θ_2}(Linear Case) . . . 80
6.4.15 u_{ZMP}(Linear Case) . . . 80
6.4.16 u_{slack}(Linear Case) . . . 81
6.4.17 ρ_1(Linear Case) . . . 81
6.4.18 ρ_2(Linear Case) . . . 81
6.4.19 Stick Figure(Linear Case) . . . 81
6.5.1 x(Nonlinear Case) . . . 84
6.5.2 z(Nonlinear Case) . . . 84
6.5.3 θ_1(Nonlinear Case) . . . 84
6.5.4 θ_2(Nonlinear Case) . . . 84
6.5.5 $\dot{x}$(Nonlinear Case) . . . 84
6.5.6 $\dot{z}$(Nonlinear Case) . . . 84
6.5.7 $\dot{\theta}_1$((Nonlinear Case) . . . 84
6.5.8 $\dot{\theta}_2$(Nonlinear Case) . . . 85
6.5.9 slack d(Nonlinear Case) . . . 85
6.5.10 slack $\dot{d}$(Nonlinear Case) . . . 85
6.5.11 u_x(Nonlinear Case) . . . 85
6.5.12 u_z(Nonlinear Case) . . . 85
6.5.13 u_{θ_1}(Nonlinear Case) . . . 85
6.5.14 u_{θ_2}(Nonlinear Case) . . . 85
6.5.15 u_{ZMP}(Nonlinear Case) . . . 85
6.5.16 u_{slack}(Nonlinear Case) . . . 87
6.5.17 ρ_1(Nonlinear Case) . . . 87
6.5.18 ρ_2(Nonlinear Case) . . . 87
6.5.19 Stick Figure(Nonlinear Case) . . . 87

7.1 Differential changes in the terminal time . . . 91
7.2 scaling for input variable . . . 92
7.3.1 x_1(dashed line: gradient method, solid line: RHGM) . . . 94
7.3.2 x_2(dashed line: gradient method, solid line: RHGM) . . . 94
7.3.3 u_1(dashed line: gradient method, solid line: RHGM) . . . 94

7.3.4 λ_1(dashed line: gradient method, solid line: RHGM) . . . 94
7.3.5 λ_2(dashed line: gradient method, solid line: RHGM) . . . 94
7.3.6 $Error_1$(RHGM) . . . 94
7.3.7 $Error_2$(RHGM) . . . 95
7.4 Nonlinear Two link system . . . 96
7.5 Experiment System . . . 97
7.6.1 θ_1(Nonlinear Two link System (vertical)) . . . 98
7.6.2 θ_2(Nonlinear Two link System (vertical)) . . . 98
7.6.3 $\dot{\theta}_1$(Nonlinear Two link System (vertical)) . . . 98
7.6.4 $\dot{\theta}_2$(Nonlinear Two link System (vertical)) . . . 98
7.6.5 u_{θ_1}(Nonlinear Two link System (vertical)) . . . 98
7.6.6 u_{θ_1}(Nonlinear Two link System (vertical)) . . . 98
7.6.7 λ_1(Nonlinear Two link System (vertical)) . . . 99
7.6.8 λ_2(Nonlinear Two link System (vertical)) . . . 99
7.6.9 λ_3(Nonlinear Two link System (vertical)) . . . 99
7.6.10 λ_4(Nonlinear Two link System (vertical)) . . . 99
7.6.11 $Error_1$(Nonlinear Two link System (vertical)) . . . 99
7.6.12 $Error_2$(Nonlinear Two link System (vertical)) . . . 99
7.6.13 $Error_3$(Nonlinear Two link System (vertical)) . . . 100
7.6.14 $Error_4$(Nonlinear Two link System (vertical)) . . . 100
7.6.15 Stick Figure(Nonlinear Two link System (vertical)) . . . 100
7.7 Experiment Device) . . . 102
7.8.1 θ_1(from counter)(Experiment) . . . 103
7.8.2 θ_2(from counter)(Experiment) . . . 103
7.8.3 $\dot{\theta}_1$(from counter)(Experiment) . . . 103
7.8.4 $\dot{\theta}_2$(from counter)(Experiment) . . . 103
7.8.5 $\theta_{1reference}$(Experiment) . . . 103
7.8.6 $\theta_{2reference}$(Experiment) . . . 103
7.8.7 $\dot{\theta}_{1reference}$(Experiment) . . . 103
7.8.8 $\dot{\theta}_{2reference}$(Experiment) . . . 104
7.8.9 u_1(Experiment) . . . 104
7.8.10 u_2(Experiment) . . . 104
7.8.11 Stick Figure(Experiment) . . . 104
7.9.1 θ_1(Kinematic Three Links) . . . 109
7.9.2 θ_2(Kinematic Three Links) . . . 109
7.9.3 θ_3(Kinematic Three Links) . . . 109
7.9.4 $\dot{\theta}_1$(Kinematic Three Links) . . . 109
7.9.5 $\dot{\theta}_2$(Kinematic Three Links) . . . 109
7.9.6 $\dot{\theta}_3$(Kinematic Three Links) . . . 109
7.9.7 $Error_1$(Kinematic Three Links) . . . 109
7.9.8 $Error_2$(Kinematic Three Links) . . . 110
7.9.9 $Error_3$(Kinematic Three Links) . . . 110
7.9.10 $Error_4$(Kinematic Three Links) . . . 110
7.9.11 $Error_5$(Kinematic Three Links) . . . 110
7.9.12 $Error_6$(Kinematic Three Links) . . . 110

7.9.13 Stick Figure(Kinematic Three Links) 110

List of Tables

5.1 Simulation Data(2-D Case) 39
5.2 Simulation Data(3-D Case) 45
5.3 Simulation Data(In case of $x > 0$) 49
5.4 Simulation Data(Cut Across x=0) 53
5.5 Simulation Data(Sudden Acceleration) 56
5.6 Simulation Data(Sudden Stop) 59
5.7 Simulation Data(Make u_{ZMP} small) 62

6.1 Two Links Mechanical Parameters 72
6.2 Parameters for Simulation(without constraint) 72
6.3 Parameters for Simulation(Linear Case) 78
6.4 Parameters for Simulation(Nonlinear Case) 83

7.1 Simulation Data 93
7.2 Simulation Data 97
7.3 Experiment Spec and Data 102
7.4 Simulation Data for Kinematic Three Links 108

Legend

Symbol	Definition
g	gravity acceleration
h_{ref}	height of C.G.
x	position along x axis
$\dot{x}$	velocity along x axis
$\ddot{x}$	acceleration along x axis
y	position along y axis
$\dot{y}$	velocity along y axis
$\ddot{y}$	acceleration along y axis
z	position along z axis
$\dot{z}$	velocity along z axis
$\ddot{z}$	acceleration along z axis
x_{ref}	reference trajectory along x axis
y_{ref}	reference trajectory along y axis
z_{ref}	reference trajectory along z axis
$\dot{x}_{ref}$	reference velocity trajectory along x axis
$\dot{y}_{ref}$	reference velocity trajectory along y axis
$\dot{z}_{ref}$	reference velocity trajectory along z axis
u_x	input along x axis
u_y	input along y axis
u_z	input along z axis
u_{zmp}	ZMP input
x_f	x of terminal state variable
y_f	y of terminal state variable
z_f	z of terminal state variable
λ	co-state variable
ϕ	constraint at terminal state
M	Inertial Matrix
V	coliolis term
G	gravity term
H	hamiltonian
ζ	$\frac{d}{dt}F = -\zeta \cdot F$
F	Error of transversality condition $F = \phi_x(x(t,T)) - \lambda(t,T)$
R	weight matrix at performance index
Q	weight matrix at performance index
Δt	time step on real-time axis
$\Delta\tau$	time step on moving evaluated interval
$*$	variable on moving evaluated interval
m_1	mass of $link_1$
m_2	mass of $link_2$
l_1	length of $link_1$

Symbol	Definition
l_2	length of $link_2$
l_3	length of $link_3$
l_1c	length of $link_1$ center of gravity
l_2c	length of $link_2$ center of gravity
${}^i\hat{Z}_i$	$\theta_i \cdot {}^i\hat{Z}_i = [0, 0, \theta_i]^T$
${}^i_{i+1}R$	rotational matrix : $Frame_i -> Frame_{i+1}$
${}^{i+1}_iR$	rotational matrix : $Frame_{i+1} -> Frame_i$

Acronyms and Abbreviations

Symbol	Definition
TPBVP	Two Point Boundary Value Problem
RHC	Receding Horizon Control
MPC	Model Predictive Control
RHGM	Receding Horizon Gradient Method
SCGRA	Sequential Conjugate Gradient Restration Algorithm
MQA	Modified Quasi Linearlization Algorithm
ZMP	Zero Moment Point
C.G.	Center of Gravity
RTOS	Real-time Operating System
2-D	Two Dimensional
3-D	Three Dimensional
OS	Operating System

Chapter 1

Introduction

1.1 Background

Developments in computer technology have made real-time robot control possible at very high levels of intelligence. Current work in control theory is aimed at industrial applications that require optimal performance in terms of maneuverability and non-linearity. A common example is a car's braking system, which requires real-time operations that allow drivers to avoid accidents. Other machines require real-time controllers to avoid instability, which can lead to accidental damage or the complete destruction of systems, people, or objects.

When machines are designed for specialized functions, control is considered a relatively straightforward task. That is not the case today, since robots and other machines are becoming more complex and generalized. Today's engineers must therefore deal with the issue of motion generation. Some of the earliest work in this area involved aerospace applications. One problem in spacecraft design involves generating a vehicle orbit that satisfies a considerable number of requirements, including air conditions, gravity, fuel consumption, upper output limits, and acceleration limits. Similar problems have arisen in other industrial fields as computer have progressed.

The first discussions of the optimization problem can be traced to the 16th century mathematicians Leonhard Euler(1707-1783) and Johann Bernoulli(1667-1748). The tranversality condition was solved by Joseph-Louis Lagrange(1736-1813) in the 17th century. Variations in these methods were developed but not applied for several centuries. Lev Semenovic Pontryagin (1908-1988) started work on a theory of oscillations and automatic control with his physicist friend A. A. Andoronov in the 1930s. However, solvable optimization problems were restricted in the 1950s and 1960s because computer capacity was limited and formula manipulation had yet to emerge. In a book entitled "The Mathematical Theory of Optimal Processes" (1961)[1], Pontryagin claimed Maximal Principle. His proposal made it possible to establish the variation method. Computers could then be used for optimization, which led to the creation of the gradient method. The gradient method was favored by many researchers because of its simplicity and utility, leading to a considerable number of variations-for example, the Sequential Conjugate Gradient Algorithm (SCGRA)[2] and the Modified Quasi Linearlisation Algorithm (MQA)[3].

The formula manipulation tool has had an important role in developing optimization techniques. Software was developed for solving accelerator problems that could not be

solved by hand, and the original tool was subsequently adapted for a variety of mathematical problems. Commercial tools such as Mathematica and Maple have made it possible for computers to be used to solve differential equations and to manipulate formulas in the calculus of variation. These tools raised the bar in terms of effectiveness when the multi-links numerical model was derived for robotics. Advancements in optimization techniques were also made possible by this new environment.

A real-time OS is required to control such real-time oriented machines as robots, intelligent automobiles, and aerospace vehicles. Whereas an ordinal operating system does not ensure time-restricted commitment execution, an RTOS does, making RTOS a requirement in environments where control is measured in terms of milliseconds.

This requirement has fueled a large amount of research in real-time optimization techniques, resulting in a considerable number of gradient method variations. Still, the method suffers from a considerable drawback in the form of off-line calculations that require excessive amounts of time. For instance, several workstation hours are needed to calculate aerospace orbit optimization. Determining an optimal trajectory from preliminary stocks requires a modern approach that utilizes a Receding Horizon Control (RHC) or Model Predictive Control (MPC) algorithm. Both algorithms hold considerable potential for real-time (dynamic) optimization.

1.2 Conventional Research Overview

1.2.1 Optimization Theory and Calculus History

Philosopher and mathematician Bernard Bolzano(1781-1848) was one pioneer in the area of fundamental calculus concepts. However, Bolzano's proofs involved arithmetic, algebra and analysis, whereas Johann Carl Friedrich Gauss(1777-1855) offered proofs of the fundamental theorem of algebra using geometry. The calculus has often been described as arising from the Pythagorean recognition of the difficulty involved in attempting to substitute numerical considerations for continuous geometrical magnitudes. Sir Isaac Newton(1642 - 1727) avoided this via the intuition of continuous motion, and Gottfried Wilhelm Leibniz (1646-1716) evaded the question via the postulate of continuity. In his definition of continuous function, Bolzano asserted that the basis of continuity was found in the limit concept. His definition of a function f(x) as continuous in an interval if for any value of x in the interval the difference $f(x + \delta x) - f(x)$ becomes and remains less than any given quantity for δx that is sufficiently small is essentially the same as that later offered by Augustin-Louis Cauchy (1789-1857). After recognizing that the subject could be explained in terms of limits of finite difference ratios, Bolzano defined the F(x) derivative for any value of x as the F'(x) quantity in which the ratio:

$$\frac{F(x - \delta x) - F(x)}{\delta x} \tag{1.1}$$

indefinitely approaches as δx approaches zero[4] [5].

Pierre de Fermat(1601-1665) proposed the principal which light travels in minimum time with calculus of variations. Johannes Bernouil(1667-1748) solved brachistchrone problem

in discrete step using Fermat idea. Issac Newton (1642-1727) used calculus of variations to design minimum drag nose shape of a projectile. Leonard Euler published "The Method of Finding Curves that Show Some Property of Maximum or Minimum. Jean Louis Lagrange (1736-1813) invented a method of variations and multipliers. Euler adopted this idea and proposed Euler-Lagrange equations. Adrean Marie Legendre(1752-1833) proposed the second variation. William Rowan Hamilton (1805-1865) published his work on least action in mechanical systems that involved two partial differential equations. Karl Gustav Jacob Jacobi(1804-1851) proposed Hamilton-Jocobi equation based on Hamilton's result. Karl Wilhelm Theodor Weierstrass (1815-1897) proposed the condition involving excess function, which is predecessor of maximum principle of Pontryagin.

Pontryagin submitted "The Mathematical Theory of Optimal Processes" in 1962; Richard Bellman had already used the term in the 1940s to describe the problem-solving process in which best decisions are found one after another. Bellman also proposed dynamic programming-a method to directly solve the Hamilton-Jacobi-Bellman equation beginning at terminal conditions. However, this method is considered impractical because it requires storing entire extremal fields in computer memory.

The original method for solving optimal problems involved choosing values for unspecified initial conditions and improving estimates of terminal conditions in order to satisfy specific terminal conditions. A major challenge associated with this method is that extremal solutions are too sensitive for estimating initial conditions. The gradient method was proposed to get around this difficulty. Such a direct integration method, which is considered practical for finding extremal solutions, is characterized as an iterative algorithm for improving estimates of control histories to reach optimal and boundary conditions. While the first order gradient method does offer a dramatic improvement for a few iterations, convergence slows considerably as the trajectories approach an optimal solution. For this reason, a second order gradient has been created for later iterations.

1.2.2 Early Practical Application of Optimization Theory

Robert H Goddard(1882-1945) worked on the aerospace trajectory problem (i.e., the optimal thrust series required to reach maximum altitude[6]) as early as 1919. Toward the end of his report, Goddard described a scenario in which a rocket reached the moon and detonated its load of ignitable powder to mark its arrival.

The shooting method was initially adopted to solve the spacecraft orbit problem, primarily because the method was feasible for analyzing conservative systems. However, since Euler-Lagrange equations are considered unstable for aircraft dynamics, the shooting method is considered unfeasible for aircraft dynamics. In an attempt to overcome this instability, the initial value of the Lagrange multiplier from a gradient code was used as an initial estimate. The gradient method (considered sufficient for arriving at an accurate solution) was used to calculate the minimum time for a low-thrust spacecraft to travel from Earth to Mars. The gradient method was also used to determine the minimum time for an F4 fighter to reach the highest altitude for launching a Sparrow missile.

1.2.3 Real-Time OS and Control

A real-time system (used when rigid time requirements are placed on processor operations or data flow) serves as a control device in a dedicated application. Applications that commonly use real-time systems are medical imaging, industrial control, and certain types of displays. Real-time systems have well-defined, fixed time constraints within which processing tasks must be completed. The two primary categories of real-time systems are:

★ **Hard real-time system** These systems ensure that all critical tasks are completed on time. To accomplish this, all system delays (from the retrieval of stored data to the time it takes the operating system to finish a request) are bounded. Most advanced operating systems tend to separate users from their hardware, resulting in uncertainty concerning the amount of time an operation requires. Since virtual memory is almost never found in real-time systems, hard real-time systems cannot be used with time-sharing systems. No existing general-purpose operating systems support hard real-time functionality.

★ **Soft real-time system** These are considered less restrictive systems in which critical real-time tasks are given priority over other tasks and retain priority status they are completed. Kernel delays need to be bounded. Soft real-time systems can be mixed with other types of systems. Given their lack of deadline support, they are considered risky for industrial control purposes. However, because of their expanded functionality, soft real-time systems have been added to most current operating systems, including major versions of UNIX.

1.2.4 Formula Manipulation

Formula manipulation programs have specific differentiation and integration capabilities and supporting simplification, display and input/output editing, and precision arithmetic capabilities. Mathematica, Maple, and Reduce are three commercial examples and Maxima is an open-source example. These tools empower PCs to a) produce multi-link system dynamic equations, and b) work with calculus operations associated with optimization techniques.

1.3 Objective and Approach

A significant issue in robot construction is motion generation; without good motion generation software, even well-built hardware will perform poorly. Although optimization techniques allow for smooth and natural motion, the off-line computing characteristic is considered disadvantageous for real-time robot control.

Based on my conviction that Receding Horizon Control (RHC) will eventually become a key technology for generating high-performance robot motion, the main objectives of this thesis are to show how Receding Horizon Control can be applied to real-time robot control and to start the development process for a Receding Horizon Control algorithm.

This thesis consists of eight chapters.

Overview of optimization problem is introduced in chapter 2. Formulation in optimization problem is defined in section 2.2. A gradient method is explained in section 2.3. The defect points of gradient method are discussed in section 2.3.

Real-time optimization is discussed in chapter 3. Homotopy method is introduced in section 3.1. Continuation method is introduced in section 3.2. Receding Horizon Control theory is explained in section 3.3.

Real-time control is discussed in chapter 4. Numerical model for optimization is introduced in section 4.1. Modern control theory is introduced in section 4.2. How to real-time control robot is discussed in section 4.3.

Chapter 5 gives an application of Receding Horizon Control to legged robot motion generation. Any type of legged robot has to be considered for Zero Moment Point balance condition. How the condition could be involved in formulation of Receding Horizon Control is discussed in this chapter. Two dimensional plane and three-dimensional space simulation are mentioned in section 5.4

Chapter 6 gives an application of Receding Horizon Control to swing leg of legged robot. The notion of Chapter 5 is extended to formulation with swing leg. Constraint of swing leg condition is more complicated than equal constraint in Chapter 5. Inequality constraint and state variable constraint are discussed in this chapter.

Chapter 7 proposes a new algorithm of Receding Horizon Control. To avoid difficulties of complicated matrix manipulation in former algorithm[7][8], this algorithm has been reduced to be simple and basic. Such basic measure could be lead to further development of Receding Horizon Control algorithm. The algorithm is explained in section 7.2. Simulation is introduced in section 7.3.

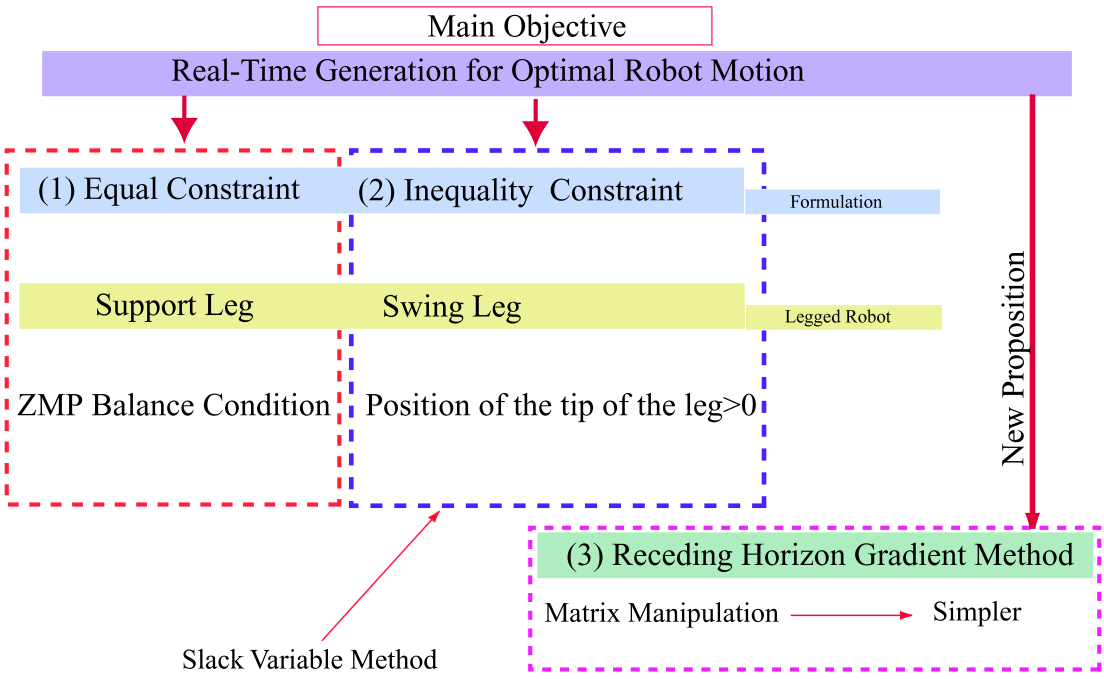


Figure 1.1: Main Objective

Chapter 2

Application of Optimization Problem to Mechanics

2.1 Introduction

There are various types of formulation about optimization. In this section, formulation necessary for mechanical object is briefly explained.

The algorithm of Gradient Method is also explained here. Well known algorithms in Gradient Method are described.

2.2 Definition of Optimization Problem

The state equation treated is multi-variable and nonlinear.

$$\dot{x}(t) = f[x(t), u(t), t] \tag{2.1}$$

x denotes state variable, u denotes input variable, t denotes real-time. x(t):n vector function is determined by u(t):m vector function.

Considered performance index is scalar function. The optimization problem is to find the functions u(t) that minimizes the performance index.

$$J = \phi[x^*(T+t)] + \int_0^{t_f} L[x(t), u(t), t]dt \tag{2.2}$$

ψ denotes terminal conditions. Then Hamiltonian is defined as:

$$H = L[x(t), \lambda(t), u(t), t] + \lambda^T(t) \cdot f[x(t), u(t), t] \tag{2.3}$$

denotes co-state variable, which fills the role as Lagrange multiplier. Euler-Lagrange equation below is derived from Hamiltonian.

$$\dot{\lambda}(t) = -H_x^T \tag{2.4}$$

$$H_u = 0 \tag{2.5}$$

$$\dot{x}(t) = -H_{\lambda}^{T} \tag{2.6}$$

$$\dot{x}(0) = x_0(t) \tag{2.7}$$

$$\lambda(t_f) = \phi_x^T[x(t_f)] \tag{2.8}$$

Euler-Lagrange equation gives the initial condition of the state and the terminal condition of the co-state and its fact implies the notion about two points boundary value problem as Fig.2.1. The initial condition about the state is known but the terminal condition is not known. However, the initial condition about co-state is not known but the terminal condition is known. This improperly paired problem is called as two points boundary value problem. If all condition in Euler-Lagrange equation is satisfied, any gradient is not raised.

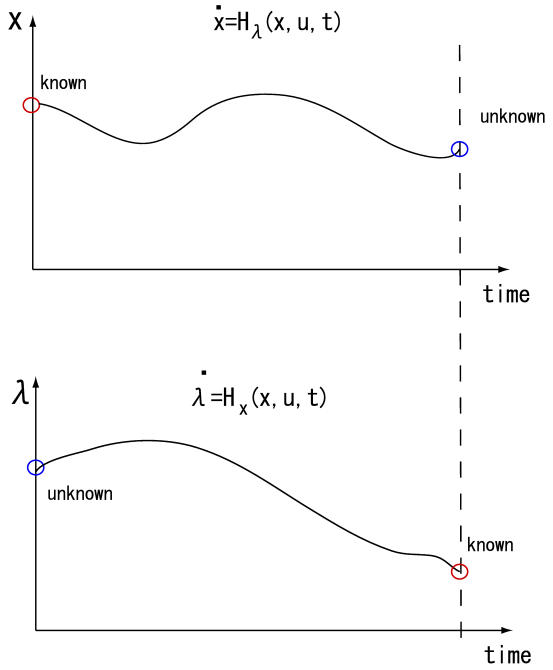


Figure 2.1: Two Points Boundary Value Problem

2.3 Equal Constraint

It takes more than state equation to solve mechanical control object. Limitation of control input, constraint of state variable, particular condition about the mechanics, dynamics change, and etc. could not be described without equal constraint. Let us see how equal constraint is useful in this section. Here we have a equal constraint as:

$$C[x(t), u(t), t] = 0 \tag{2.9}$$

We have to consider this constraint to be involved into Euler-Lagrange equation.A new lagrange multiplier ρ is introduced here and Hamiltonian is:

$$H = L + \lambda^T(t) \cdot f[x(t), u(t), t] + \rho^T(t) \cdot C[x(t), u(t), t] \tag{2.10}$$

To search the control input u to minimize Hamiltonian H means to solve $H_u = 0$. If Euler-Lagrange equation is satisfied, then the equal constraint is also satisfied.

2.4 Gradient Method

Gradient method is well known and has been used in science and engineering field. In this section, the method is explained.

2.4.1 First Order Gradient of Performance Index

Consider the first order variation in J due to variations in the control vector u(t) for fixed time t_0 and t_f,

$$\delta J = [(\frac{\partial \phi}{\partial x} - \lambda^T)\delta x]_{t=t_f} + [\lambda^T \delta x]_{t=t_0} + \int_{t_0}^{t_f} [(H_x + \dot{\lambda}^T(t))\delta x + H_u \delta u]dt \tag{2.11}$$

$$\delta J = \lambda^T(t_0)\delta x_{(t=t_0)} + \int_{t_0}^{t_f} [H_u \delta u]dt \tag{2.12}$$

This equation implies that is the gradient of J if u(t) holds a constant value. If x(t_0) holds a constant, H_u represents the variation in J.

2.4.2 Various Sorts of Gradient Method

Gradient method has various variations nowadays. Such mainstays are introduced here.

1. Sequential Gradient Restoration Algorithm

   Sequential Gradient Restoration method[2] has been introduced to solve nonlinear programming problems. The idea of this method is optimization process is divided into two phases: gradient phase and restoration phase.

   (a) Restoration Phase

   Equation(2.11) and (2.12) are the conditions of the restoration, then the performance index at this phase is defined as Equation(2.13).

$$C(x, u) = 0 \tag{2.13}$$

$$\dot{x} - \phi(x, u) = 0 \tag{2.14}$$

$$P = \int_{t_0}^{t_f} \|\dot{x} - \phi(x, u)\| \, dt + \int_{t_0}^{t_f} \|C[x, u]\| \, dt \tag{2.15}$$

(b) Gradient Phase

The augmented performance index is defined as:

$$J = \phi|_{t=0} + \int_{t_0}^{t_f} [f(x, u) + \lambda^T(\dot{x} - \phi(x, u)) + \rho^T C(x, u)]dt \tag{2.16}$$

Its first order differential is :

$$Q = \int_{t_0}^{t_f} \left\| \dot{\lambda} - f_x^T + \phi_x^T \lambda - C_x^T \rho \right\| dt + \int_{t_0}^{t_f} \left\| f_u^T + \phi_u^T \lambda - C_u^T \rho \right\| dt + \int_{t_0}^{t_f} \left\| \lambda + \phi_x^T \right\| dt \tag{2.17}$$

Equation(2.15) is merely considered in optimization at this phase. The restoration conditions are not considered here. The process of the method iterates these phases alternatively, and the gradient is reduced step by step.

2. Modified Quasi-linearization Algorithm

P and Q are reduced at once in MQA[3]. MQA uses second order variation because partial differential of P and Q. Q is originally first order function, and its partial differential comes to be second order. First order algorithm converges dramatically at first few steps, however its convergence could not last long in later steps. This is because the algorithm uses the gradient going up through first order variation. Then, second order algorith like as MQA is used for making good convergence at the later steps.

2.4.3 Example of Gradient Method

An example[11] solved with steepest descent algorithm. The state equation is defined as.

$$\frac{d}{dt} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} (1 - x_1^2(t) - x_2^2(t))x_1(t) - x_2(t) + u(t) \\ x_1(t) \end{bmatrix} \tag{2.18}$$

Performance index is defined as.

$$J = 2 \cdot (x_1^2(t_f) + x_2^2(t_f)) + \int_{t_0}^{t_f} (x_1^2(t) + x_2^2(t) + u^2(t))dt \tag{2.19}$$

The solution is figured in Fig.2.2. This algorithm is converged within 30 iterations.

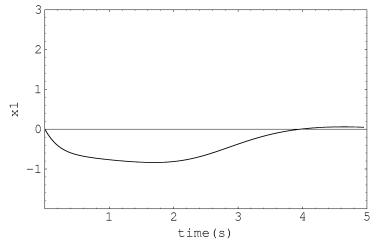


Fig2.2.1 x_1(Steepest Descent)

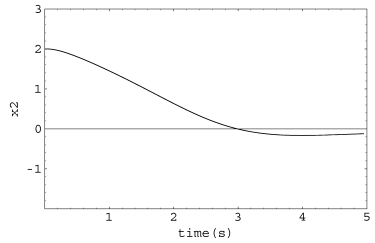


Fig2.2.2 x_2(Steepest Descent)

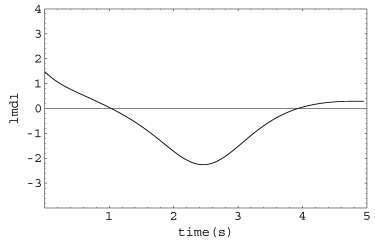


Fig2.2.3 λ_1(Steepest Descent)

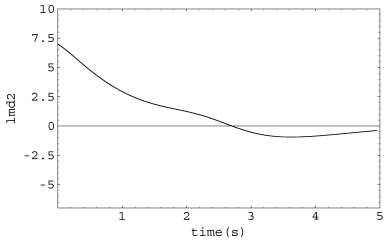


Fig2.2.4 λ_2(Steepest Descent)

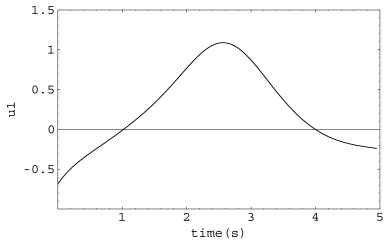


Fig2.2.5 u_1(Steepest Descent)

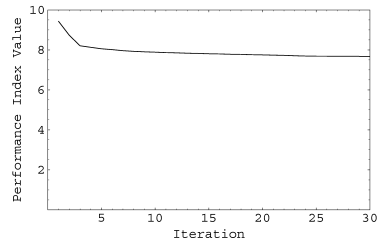


Fig2.2.6 Performance index value / Number of iteration (Steepest Descent)

2.5 Application of Gradient Method to real-time control robot and Matters

The question that inevitably arises in real-time control of robot is how to make motion generation. Robot cannot behave without this solution. Its trajectory or motion must be optimized so as to possess naturally smoothness. Some robot of early date does awkward motion because some trigonometric functions make up the motion function.

However optimization must be needed, its calculation time is too long to execute on real-time controller. For example, 5 seconds simulation needs 1-hour calculation time. Then the measure which pre-optimized trajectories are stored in computer memory was taken. The defect of this measure is lack of flexibility against unexpected happening. The action of the robot is restricted in the measure. It cannot maximize the effect of the optimization.

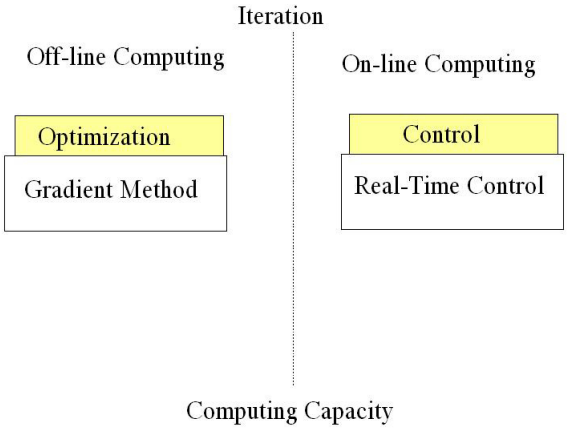


Figure 2.3: Dilemma between off-line optimization and real-time control

2.6 Conclusion

Optimization formulation and application to real-time control is discussed in this chapter. Equal constraint is quite useful to make necessary conditions for mechanics involve with formulation of optimization. Gradient Method is popular to solve optimization problem.

Chapter 3

Real-Time Optimization Technique

3.1 Introduction

Homotopy Method and Continuation Method constitute substantial portion of Receding Horizon Control used in this study. Those fundamentals are explained in this chapter.

3.2 Homotopy Method

A banach space which the sphere with radius r forms is defined as:

$$B = \{x \in X | \, \|r \geq x\|\} \tag{3.1}$$

B $->$ X is put a case that it is compact mapping

$$\begin{aligned} h(x,t) &= (1-\tau)\cdot g(x) + \tau \cdot f(x) \\ &(x,\tau) \in B \times [0,1] \end{aligned} \tag{3.2}$$

This equation has a property as:

$$\begin{aligned} h(x,0) &= g(x) \\ h(x,1) &= f(x) \end{aligned} \tag{3.3}$$

This is called Homotopy of mapping f(x) and g(x). g(x) has the trivial solution g(x_0).

$$h(x,\tau) = (1-\tau)\cdot g(x_0) + t \cdot f(x) \tag{3.4}$$

We can obtain the solution shifting from $\tau = 0$(trivial solution g(x_0)) to $\tau = 1$.

3.3 Continuation Method

To obtain the solution using Homotopy method, the solution trajectory must be tracked along parameter t. Predictor-Corrector Method is orthodox for the tracking. Predictor-Corrector Method is composed of two steps below.

(1)Predictor
The Jacobian of f(x) is defined as:

$$Df(x_0) = \frac{\partial f_i(x)}{\partial x_j}|_{x=x_0} \tag{3.5}$$

Then calculate Df(x_0) at the current point x_0. The current point is redefined as the contact point and a line v(x) directing to Df(x_0) is assumed.

$$Df(x_0)\nu(x) = 0 \tag{3.6}$$

(2)Corrector The length of v(x) is,

$$\|\nu(x)\| = 1 \tag{3.7}$$

A normal line is set from the point which has distance v(x) from the point x_0. The intersection of the normal line with the curve f(x) is set as the next point. Iterating these steps until t=1 and the tracking of the solution trajectory can be done.

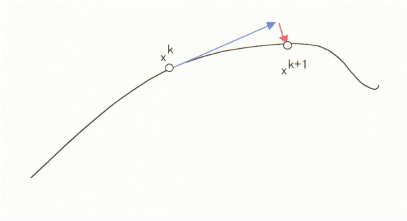


Figure 3.1: Predictor-Corrector Method

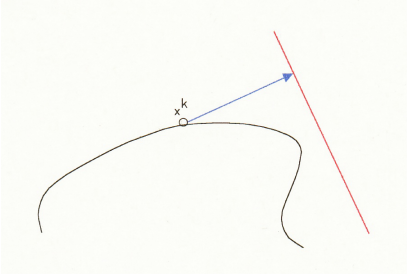


Figure 3.2: Ill case in Predictor-Corrector Method

One of defects of this method is a case, which the curve of the solution trajectory is extremely crooked like as Fig.3.2.

3.4 Solve Optmization Problem with Stabilized Continuation Method

3.4.1 Formulation

Let us consider to solve a generalized optimization problem below with continuation method. $\tau \in [0, 1]$ is defined as a continuation parameter

$$\dot{x}(t) = f[x(t), u(t), t, \tau] \tag{3.8}$$

$$J = \phi[x(t_f)] + \int_0^{t_f} L[x(t), u(t), t, \tau]dt \tag{3.9}$$

One of the advantages to use Continuation Method is that it starts from a trivial solution. Popular one is terminal time $t_f = 0$. Transversality condition comes off as below. Starting from $t_f = 0$, TPBVP extends the terminal time bit at a time.

$$\phi(x(0)) = \lambda(0) \tag{3.10}$$

In order to make the Euler-Lagrange equation transit in a continuous fashion, the perturbation from optimal path of Euler-Lagrange equation. In this case, the terminal time is unspecified.

$$d\lambda(t_f) = \phi_{xx}dx + \frac{\partial}{\partial t}\phi_x dt_f \tag{3.11}$$

The formation of the equation avobe resembles the formation of the equation in Backward Sweep Method[23]. An equation below is assumed to obtain costate variable in Backward Sweep Method.

$$\delta\lambda(t) = S(t)\delta x + c(t)dt \tag{3.12}$$

These S and c are defined as equations below.

$$\frac{d}{dt}S(t) = -A^T \cdot S - S \cdot A + S \cdot B \cdot S - C \tag{3.13}$$

$$\frac{d}{dt}c(t) = -(A^T - S \cdot B) \cdot c \tag{3.14}$$

The matrix A is treated in transition matrix[23](Appendix:A).

$$\frac{d}{dt}\begin{bmatrix}\delta x \\ \delta\lambda\end{bmatrix} = \begin{bmatrix}A & -B \\ -C & -A^T\end{bmatrix}\begin{bmatrix}\delta x \\ \delta\lambda\end{bmatrix} \tag{3.15}$$

$$\begin{aligned} A(t) &= f_x - f_u \cdot H_{uu}^{-1} \cdot H_{ux} \\ B(t) &= f_u \cdot H_{uu}^{-1} \cdot f_u^T \\ C(t) &= H_{xx}^{-1} - H_{xu}^{-1} \cdot H_{uu}^{-1} \cdot H_{ux}^{-1} \end{aligned}$$

The costate variable can be obtained from equations above, and we can also obtain optimal input variable in the equation below.

$$H_u = 0 \tag{3.16}$$

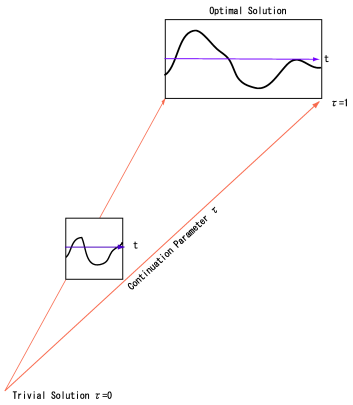


Figure 3.3: Solve Optimization Problem with Continuation Method

3.5 Receding Horizon Control

Receding Horizon Control or Moving Horizon Control has been emerged as a feedback strategy for linear and nonlinear plants. Mayne and Michalska[19], Eaton and Rawlings[21] proposed formulations for nonlinear system. Some stability analysis about linear system without constrains at finite horizon[15][14]. The concept of Receding Horizon Control is to determine the control input that optimizes some open loop performance objective on a time interval extending from the current time to the future terminal time. One of the features is that feedback is incorporated using the measurement to update the optimization problem for the next time step. Ohtsuka and Fujii[7][8] has developed an algorithm using homotopy notion and backward sweep method. This algorithm is used for numerical simulation in Chapter5 and Chapter6. An idea of homotopy method like here is that they had their eye on error of transversality condition. TPBVP on receding horizon has transversality condition:

$$F = \lambda(t_f) - \phi_x(x(t_f)) \tag{3.17}$$

The difference between Receding Horizon Control and previous section is that the evaluated interval of performance index moves along time. Because Receding Horizon Control updates the initial condition of the state variable, it gives state feedback. We have to make a notation to describe such space as $x^*(t, \tau)$. The axis τ means the evaluated interval.

The initial digits of the input variable and costate variable to update on real-time axis. This is feature of Receding Horizon Control. Controller uses only initial array of variables on τ axis.

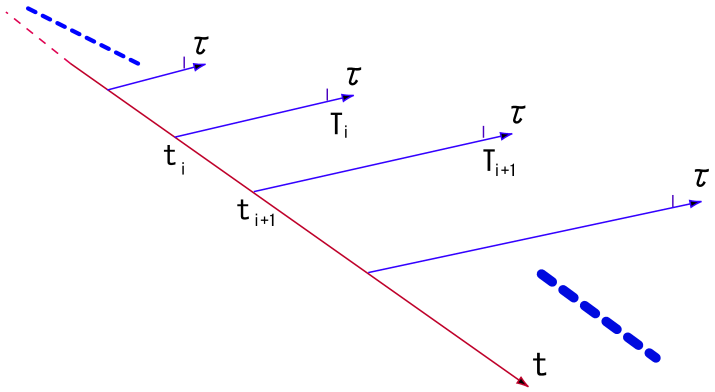


Figure 3.4: Moving Horizon

$$\frac{d}{dt}\lambda(t) = S^*(t,0)\cdot\delta x + c^*(t,0) \tag{3.18}$$

$$H_u = 0 \tag{3.19}$$

* means that the variable is on τ axis.

Receding Horizon Control predicts future performance from current time until $t + T$. Receding Horizon Control is different from full-time optimization. The full-time optimization evaluates performance index in full range of the time. However, advantage of Receding Horizon Control is that it can treat with unexpected happening real-timely.

3.6 Conclusions

Homotopy Method and Continuation Method are the basic building block of Receding Horizon Control algorithm. Backward Sweep Method is well known method to obtain optimal solution of gradient method traditionally. Receding Horizon algorithm is composed of Continuation Method and Backward Sweep method. Chapter5 and Chapter 6 are explained using this algorithm.

Chapter 4

Real-Time Control for Robot

4.1 Introduction

Real-time control of a robot requires specific items. The numerical model of a robot can be described as mechanical link system. Then some method of its derivation has been considered. The popular one is Lagrange Method, another one is Newton-Euler Method. Those methods are briefly introduced in this chapter.

Real-time OS is also indispensable item. What take a look at its contexture is important to consider real-time optimization technique.

4.2 Numerical Model

The robot must be numerically modeled when real-time control of the robot is done. In robotics engineering, methods to build numerical model of link mechanism has been developed. Such methods are Lagrange Method, Newton-Euler Method, and so on.

4.2.1 Lagrange Method

Lagrange Method derives numerical model from kinetic energy and potential energy. Because Lagrange Method uses generalized coordinates, arbitrary coordinates without Cartesian coordinates. It has flexibility that some restrictions can be added when the problem is formulated. This method has been effective after some formula manipulation softs emerged. K is defined as kinetic energy and P as potential energy. Lagrange function L is,

$$L = K - P \tag{4.1}$$

The variable of generalized coordinates denotes q:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} \tag{4.2}$$

4.2.2 Newton-Euler Formulation

Newton-Euler formulation is force and moment -based, and requires an ability to describe all the forces and moments acting upon the different components of the link-system.

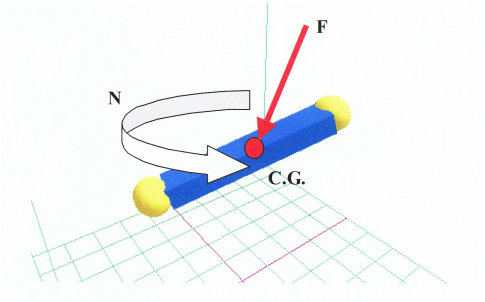


Figure 4.1: Force and Moment on a Link

The degree of the freedom of each component is six. To describe the movement, translational motion: three and rotational motion: three are needed. D denotes kinetic momentum, E denotes angular momentum.

$$F = \frac{dD}{dt} \tag{4.3}$$

$$N = \frac{dE}{dt} \tag{4.4}$$

The relation between links "i" and "i+1" is described as rotational matrix. The rotational matrix, which is from link "i" to "i+1" is defined as:

$${}^{i}_{i+1}R = \begin{bmatrix} \cos\theta_{i+1} & -\sin\theta_{i+1} & 0 \\ \sin\theta_{i+1} & \cos\theta_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

The inverse operation of it is defined as:

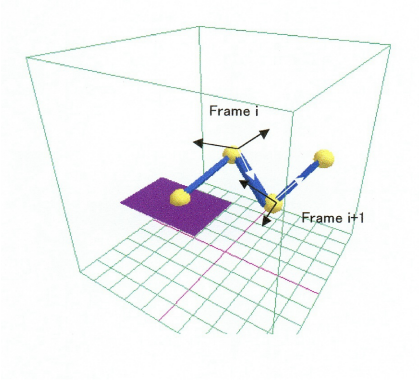
$${}^{i+1}_{i}R = \begin{bmatrix} \cos\theta_{i+1} & \sin\theta_{i+1} & 0 \\ -\sin\theta_{i+1} & \cos\theta_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

This notation is based on [10]. Angular velocity, which is from the coordinate "i" to the coordinate "i+1", is defined as:

$${}^{i}\omega_{i+1} \tag{4.7}$$

The formulation is recursive and simple sequence. It is divided to inward iterations and outward iterations.

(Inward Iteration) Link $0 -> $ The end effector: (Angular Velocity)



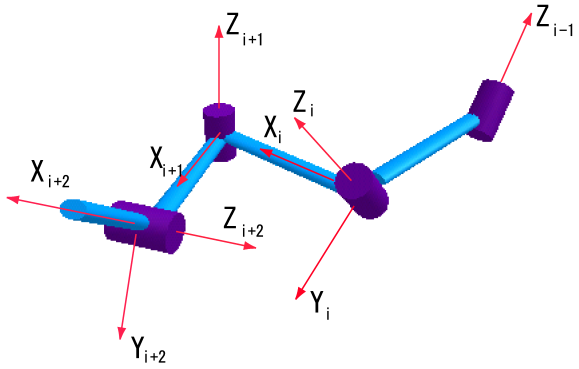


Figure 4.2: Frame Relation

$$
\begin{aligned}
{}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \\
\theta_i \cdot {}^{i}\hat{Z}_i &= \begin{bmatrix} 0 \\ 0 \\ \theta_i \end{bmatrix}
\end{aligned}
$$

(Angular Acceleration)

$$
{}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + {}^{i+1}_{i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \tag{4.8}
$$

(Translation Acceleration)

$$
{}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i) \tag{4.9}
$$

(Translation Acceleration at the center of the gravity)

$$
{}^{i+1}\dot{v}_{Ci+1} = ({}^{i+1}\dot{\omega}_{i+1} \times {}^{i}P_{Ci+1} + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{Ci+1}) + {}^{i+1}\dot{v}_{i+1}) \tag{4.10}
$$

(Force of Translation motion)

$$^{i+1}F_{i+1} = m_{i+1} \cdot {}^{i+1}\dot{v}_{Ci+1} \tag{4.11}$$

(Moment of Translation Motion)

$$^{i+1}N_{i+1} = {}^{Ci+1}I_{i+1} \cdot {}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{Ci+1}I_{i+1}{}^{i+1}\omega_{i+1} \tag{4.12}$$

(Outward Iteration) The end effector $->$ Link 0:

$$^{i}f_i = {}^{i}_{i+1}R^{i+1}f_{i+1} + {}^{i}F_i \tag{4.13}$$

$$^{i}n_i = {}^{i}N_i + {}^{i}_{i+1}R^{i+1}n_{i+1} + P_{Ci} \times {}^{i}F_i + {}^{i}P_{i+1} \times {}^{i}_{i+1}R^{i+1}f_{i+1} \tag{4.14}$$

4.2.3 State Equations

To describe the robot model as a state equations, joint angle and joint angular velocity are tend to be defined as the state variables, joint torques are defined as input variables.

$$\frac{d}{dt}\begin{bmatrix}\theta_1(t)\\ \vdots \\ \theta_n(t)\\ \dot{\theta}_1(t)\\ \vdots \\ \dot{\theta}_n(t)\end{bmatrix} = \begin{bmatrix}\dot{\theta}_1(t)\\ \vdots \\ \dot{\theta}_n(t)\\ M^{-1}(u - V(\Theta, \dot{\Theta}) - G(\Theta))\end{bmatrix} \tag{4.15}$$

$$\Theta = \begin{bmatrix}\theta_1(t)\\ \vdots \\ \theta_n(t)\end{bmatrix}, u = \begin{bmatrix}u_1(t)\\ \vdots \\ u_n(t)\end{bmatrix}, M : inertial\ matrix, V : colioris\ term, G : gravity\ term$$

Describing nonlinearlity causes complication to be feasible. In the era when there is no formula manipulation software, Three-Dimensional description was too much difficult to be feasible. Then such measurements below were considered:

★ Two 2-D equations constitute a 3-D model.

★ The perturbation model is used and the model is simplified.

★ Eliminate some nonlinear terms

★ Linearization

After formula manipulation software has been emerged, this problem has not been critical.

4.3 Real-Time Control using Modern Control Theory

Various methods have been developed to control robot or another subjects. Classical control theory, H infinity, Fuzzy, Neuro-control, Adaptive control, have been applied into real-time control of robot. In such applications, most controversial feature is how to generate its motion, especially each joint trajectory.

The basic in modrn control theory is optimal regulator. However, optimal regulator was not designed for robotics originally. Optimal regulator treats state equation as a linear problem.

$$\dot{x} = A \cdot x + B \cdot u \tag{4.16}$$

Matrix A and B should be time invariant. Such equation could not treat nonlinear dynamics like as robot arm.

$$u(t) = M(\Theta(t)) \cdot \ddot{\Theta}(t) + V(\Theta(t), \dot{\Theta}(t)) + G(\Theta(t)) \tag{4.17}$$

One of the most popular method to handle nonlinear mechanical links is nonlinear compensation method. The control input has nonlinear terms.

$$\begin{aligned} u(t) = M(\Theta(t)) \cdot (K_p \cdot (x_{ref} - x(t)) + K_v \cdot (\dot{x}_{ref} - \dot{x}(t)) \\ + K_i \cdot \int (x_{ref} - x(t))dt + V(\Theta(t), \dot{\Theta}(t)) + G(\Theta(t)) \end{aligned} \tag{4.18}$$

If the control input above is acted into the control object of a nonlinear mechanical link system, the nonlinear term of the dynamics is canceled.

$$\ddot{\Theta}(t) = K_p \cdot (x_{ref} - x(t)) + K_v \cdot (\dot{x}_{ref} - \dot{x}(t)) + K_i \cdot \int (x_{ref} - x(t))dt \tag{4.19}$$

Then the dynamics is changed to linear equation, and optimal regulator theory could be applied. This is main story of nonlinear compensation method. The nonlinear term is compensated at this method, however, designer have to define the trajectories of the angular position, velocity, and acceleration of the joints. Optimal regulator acts only at these error among reference and current states. Angular position, velocity, and acceleration of one link arm like as Fig.4.3 must be defined by a designer. Designer has to consider how to use gravity term(nonlinear term) well. The link behavior which makes an effective use of nonlinear term could not be generated automatically.

Autonomous generation needs somewhat intelligence or designers support. Eventual result always reached to optimization. Optimization could generate a motion closed to natural motion by animal or human. It is recognized as least energy consumption, least time, or some least criteria.

If optimization technique treat such problem, the problem is formulated as a TPBVP. Although the initial and terminal conditions are defined, the transition of the trajectories rely on optimization process. The links behavior is generated automatically and it makes an effective use of nonlinear term.

The critical path was the calculation time of optimization. To avoid this, some measurements were considered.

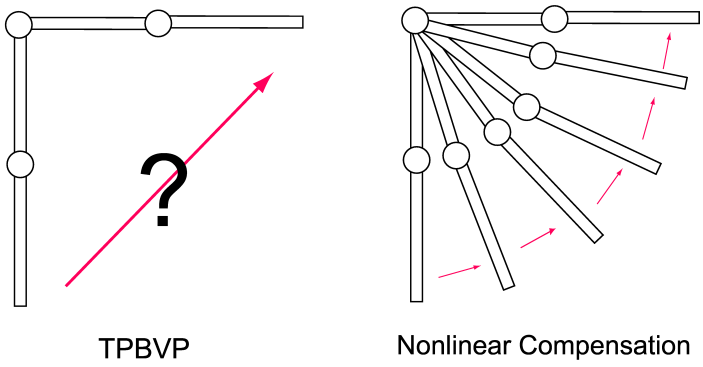


Figure 4.3: Difference between Optimal Regulator and Receding Horizon Control

4.4 Singular Point

The relation between Cartesian coordination and Joint angle coordination is defined as:

$$\dot{x} = J \cdot \dot{\Theta} \tag{4.20}$$

J is called as Jacobian. We always use Cartesian coordination to order a robot and have to translate it to joint angle coordination using Jacobian.

$$\dot{\Theta} = J^{-1} \cdot \dot{x} \tag{4.21}$$

J consists of trigonometric functions and J^{-1} diverges if the posture goes to singular point. To avoid singular point is one of major problems in robotics field.

Jacobian has one more problem that the calculation needs long time if the link system is large scaled. Computing of the inverse of the Jacobian matrix takes long time. This is critical path for real-time control.

Furthermore, if the link system is redundant system, it becomes difficult to obtain the inverse of the Jacobian. A common practice in such case is to make pseudo inverse.

$$J^{+} = (J^{T} \cdot J)^{-1} J^{T} \tag{4.22}$$

J^{+} is merely pseudo and it is not real inverse. If we use this to generate robot motion, it is far from optimal motion.

Receding Horizon Control eliminates these problems because it does not use Jacobian matrix. These problems are discussed in Chapter7.

4.5 Real-Time OS

4.5.1 Real-Time OS

The round robin scheduling used in Unix cannot treat periodical task processing. If the system falls in deadlock, which task will over the deadline cannot be expected. Multi-task real-time operating system treats that the timing control will not interfere[13]. It has both functional decomposition and time decomposition. The level to make processes parallelized varies three below:

1. Fine-graded: Statement level
2. Middle-graded: Iteration level
3. Coarse-graded: Function level

Generally the embedded operating system is a coarse-graded.

Real-time system is divided to thread model (Itron, VxWorks) and process model (RT-Linux). Address space is independent in each process of the process model. The reliability of the process model is high because the data of process-process is protected. Even if it has bugs, it is protected. Process model is a Coarse-graded. Object oriented is a thread model. A large scaled system needs a thread model because of development efficiency. Then, such two models merits are utilized to multi-process/multi-thread model (QNX,Lynx,OS-9,v3,OSE). UNIX and Windows also multi-process/multi-thread model. Linux converted to real-time operating system has varied like as RT-Linux or Time-Sys Linux. RT-Linux does not have protected function of process model, and if it has a bug, it will be crashed. Time-Sys Linux has a protect function but the accuracy of time lacks a digit than kernel space execution.

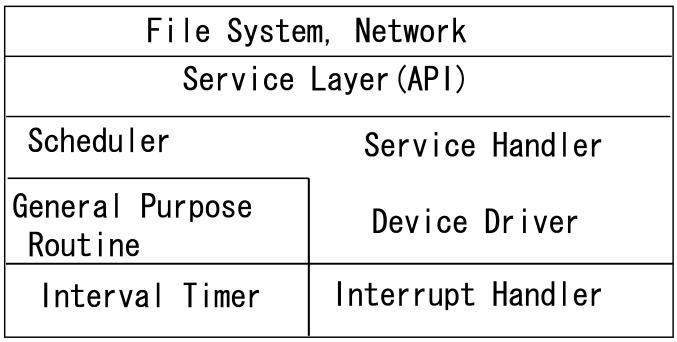


Figure 4.4: Structure of Real-Time OS

4.5.2 Interrupts

The CPU hardware has a wire called the interrupt-request line. When the CPU detects that a controller has asserted a signal on the interrupt request line, the CPU saves small

amount of state, and jumps to the interrupt-handler routine at a fixed address in memory. The interrupt handler determines the cause of the interrupt, performs the necessary processing, and executes a return from interrupt instruction to return the CPU to the execution state prior to the interrupt. This basic interrupt mechanism enables the CPU to respond to an asynchronous event, such as a device controller becoming ready for service. Most CPUs have two interrupt request lines. One is the non-maskable interrupt, which is reserved for events such as unrecoverable memory errors. The second is maskable. It can be turned off by the CPU before the execution of critical instruction sequences that must not be interrupted. Device controllers to request service use the maskable interrupt. The interrupt mechanism accepts an address - a number that selects a specific interrupt handling routine from a small set. In most architecture, this address is an offset in a table called the interrupt vector. Hitachi Super-H2 has hardware interrupt vector table and the respond time is several tens - hundreds of nanoseconds. Hitachi Super-H3 has software interrupt vector table and the respond time is several microseconds. The practical time to respond for interrupt is interrupt time + interrupt mask time. The interrupt mask time is based on the time, which is the longest system call time to be taken. Hitachi Super-H3 has several - several tens of microseconds in thread model, several tens of microseconds - several milliseconds in process model.

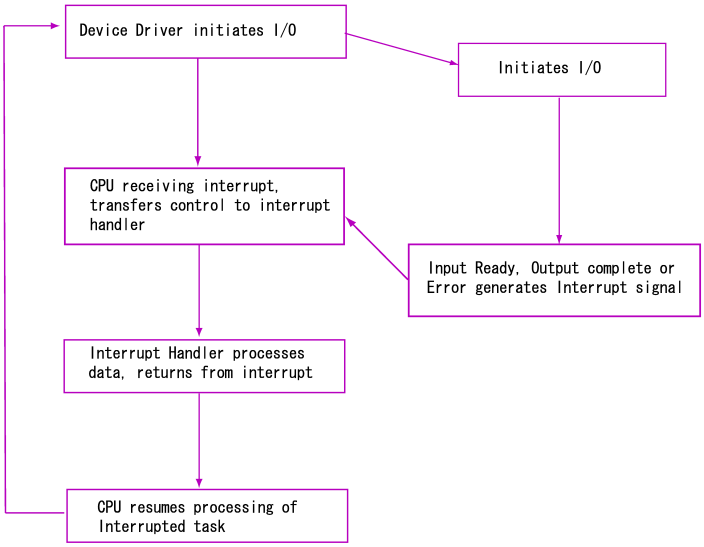


Figure 4.5: Interrupt driven I/O cycle

4.6 Application to Legged Robot Control

Modern control theory enabled robot precisely control as acceleration, velocity, and position level. One of strong progresses in robot control is force control. Sophisticated force sensor has enabled a controller to add compliance control, and then robot has been

able to do some dexterity such as holding Tofu by end-effectors. Control of legged robot is higher leveled control than another kind of robot because its must be dynamically balanced. Almost early stage of legged robots does static walk, but nowadays many legged robots have done dynamic walk. ZMP (Zero Moment Point) must be somewhat controlled in dynamic walk.

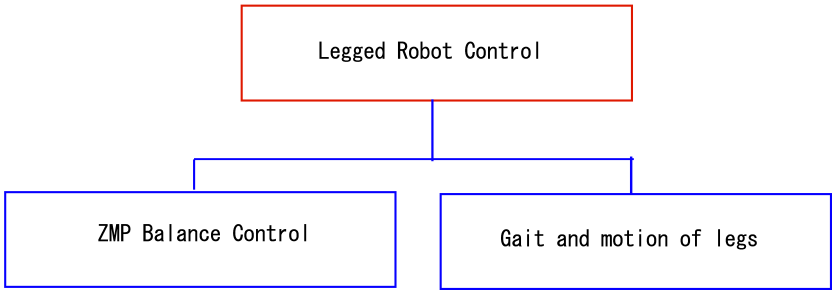


Figure 4.6: Control for Legged Robot

4.7 Conclusions

Items for real-time control of robot are discussed in this chapter. Newton-Euler Method to make numerical model for robot is useful item. Formulation of optimization problem at later chapters treats this model.

To manage real-time control, the knowledge about real-time OS is essential.RT-Linux,Tornado,T-Engine,etc. are based on real-time OS architecture.

Accumulation of various knowledge is needed for real-time control of a robot.

Chapter 5

Equal Constraint for Balance Condition of Legged Robot

5.1 Introduction

Legged robot has specific balance condition attributable to the unstable dynamics. It needs to contrive ways to involve such condition into formulation.

While the legged robot is standing on one foot, the point at which the center of gravity (C.G.) of the robot is projected onto the ground must be located on the sole plane to enable it static walk. While standing on two feet, there must be a point on the plane, which connects both the soles. While standing on four feet, there must be a point on the polygon, which consist of the four soles. While the robot moves, in order to be stabilized dynamically and to walk, the same concept is required. Generally, this is called ZMP (Zero Moment Point [16][17]Fig.5.1). ZMP within sagittal plane(Fig.5.2) can be expressed as follows from link i=0 to i=n.

ZMP is the point on the ground where ground reaction forces are applied.

$$x_{zmp} = \frac{-\Sigma_{i=0}^{n} m_i(-g + \ddot{z}_i)x + \Sigma_{i=0}^{n} m_i \ddot{x}_i z_i}{\Sigma_{i=0}^{n} m_i(-g + \ddot{z}_i)} \tag{5.1}$$

g is gravity acceleration and m is the mass of each link. If the "M" is represented for the whole mass,

$$x_{zmp} = \frac{-M(-g + \ddot{z})x + M\ddot{x}z}{M(-g + \ddot{z})} \tag{5.2}$$

This equation means that the sum total of moment of the point-mass around the origin of the coordinate balances with the moment generated by the ZMP distance from the origin and the reaction force from the ground. If the ZMP is located in the polygon constituted by the soles as well as the point at which the C.G. projects itself onto the ground in a static walk, the robot is stabilized and a dynamic walk can be carried out. If the ZMP runs-over from this polygon, it will cause the robot to fall and it cannot continue to walk. An attempt to converge the ZMP to a referenced ZMP trajectory by using feedback control in recent years has been performed [18]. Then, how a ZMP reference trajectory could be generated poses the next problem.

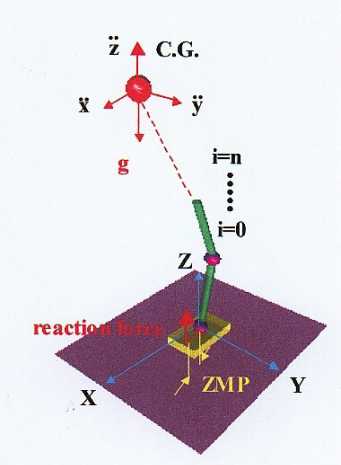


Figure 5.1: Definition of Zero Moment Point

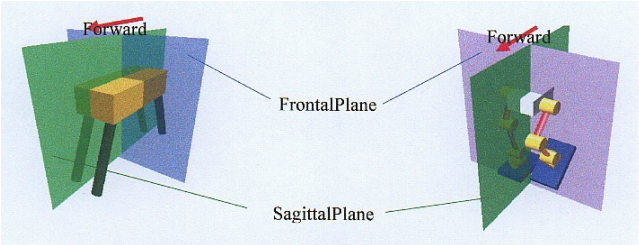


Figure 5.2: Definition of sagittal plane and frontal plane

In the conventional research of legged robot, there are two variables "x" and "z" in Equation(5.2) and poses a problem in solving the ZMP variable. Because it is not solved uniquely. An optimization problem must be solved to obtain a solution. If the condition , which holds a center of gravity position at fixed height, is added, we can avoid this problem temporarily. Then, the variable $\ddot{z}$ in the equation is set to 0, the equation could be described as follows, and a pseudo solution is uniquely obtained(Fig.5.3).

In the conventional legged robot research, one of big problems was to compute ZMP by Equation(5.2) since there are two variables of "x" and "z". To avoid this problem, some treatments had been concerned[22]. The main concept was to make the numerical model unique. Nonlinear dynamic equation is linearized adding constraints and reduced to unique equation like as Fig.5.3

A constraint below is added:

$$z = k \cdot x + h_{ref} \tag{5.3}$$

h_{ref} denotes a fixed height. Then we have linear equation:

$$\ddot{x} = \frac{g}{h_{ref}} \cdot x + \frac{1}{m \cdot h_{ref}} \cdot \tau \tag{5.4}$$

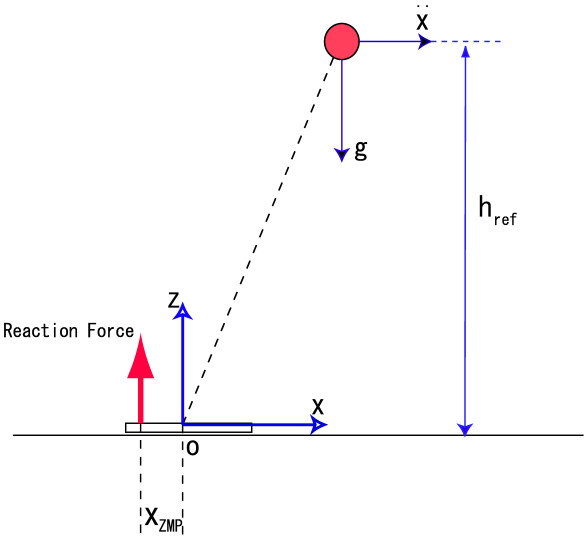


Figure 5.3: Linearization and Reduction

τ denotes ankle joint torque. The value of ZMP can be obtained from τ and sensed value of floor reaction force. Then the ZMP could be in proportion with the acceleration of x. One of the defects in this equation is that the torque necessary for whole the robot is collected on the ankle joint. Redundancy, which the robot possesses, could not be utilized effectively. Equation(5.3) is defined arbitrarily by designer.

★ Excessive torque for whole the robot is converged to the ankle joint

★ Robot motion is restricted because the constraint is adopted (The motion is on a linear line)

★ Solutions for another joints without the ankle joint could not be obtained

The second item implies that robot motion is not natural. What it takes to utilize redundancy of a robot is optimization. Since iterative calculation is needed in the optimization by the gradient method, such technique usually turns into off-line calculation. However, in a robot control that the real-time performance is required, off-line optimization is disadvantageous. When an unexpected situation appears, it could not be coped with.

Many engineering applications require real-time solutions of optimization problems. However, traditional algorithms for digital computers may not provide real-time optimization. An attractive and promising approach was introduced to real-time solutions for optimization problems known as Receding Horizon Control [7] [8] [19] [20]. This new optimization technique goes into the practical usage stage. Since Receding Horizon Control does not use a gradient method for optimization, it can carry out calculation processing of the optimal solution in short time such as a real-time control interval. Although much research has been conducted in respect to the theory, applying Receding Horizon Control

to robotics still has no actual example. This paper describes ZMP control of the legged robot using Receding Horizon Control proving that real-time optimization is available. Furthermore, it proposes a method of generating the optimum ZMP reference [28].

5.2 Receding Horizon Control with Equality Constraint

At the first stage in the history of Receding Horizon Control, Receding Horizon Control has been proposed for the linear system [14]. Then, Chen and Shaw [15], Mayne and Michalska [19] applied Receding Horizon Control to the general nonlinear system. Mayne and Michalska [20] is described the Robust design technique of Receding Horizon Control. Ohtsuka and Fujii [7] [8] developed the practical nonlinear control system design technique of Receding Horizon Control.

Receding Horizon Control formulation without constraints has been performed backwards. In this chapter, Receding Horizon Control containing the equality constraint is focused on and explained.
The state equation to treat,

$$\dot{x}(t) = f[x(t), u(t), t] \tag{5.5}$$

As equality constraint,

$$C[x(t), u(t), t] = 0 \tag{5.6}$$

If equality constraint condition can be used, it is convenient when formulizing a problem like real-time control of a robot.

The performance index is defined as,

$$J = \phi[x^*(T+t)] + \int_t^{t+T} L[x^*(t,\tau), u^*(t,\tau)]d\tau \tag{5.7}$$

Receding Horizon Control has added superscript * to the variable on the time-axis τ which moves, in order that the evaluation section may move with time. The left side of the bracket "(t,τ)" means the real time "t", and the right side of this bracket means the time on the τ axis. Hamiltonian is described as,

$$H = L + \lambda^{*T} f + \rho^{*T} C \tag{5.8}$$

$\lambda^*, \rho^*$are co-state variables. Euler-Lagrange equations are described as,

$$\dot{\lambda}^*(t,\tau) = -H_x^T \tag{5.9}$$

$$H_u = 0 \tag{5.10}$$

$$\dot{x}^*(t,\tau) = -H_\lambda^T \tag{5.11}$$

$$C[x^*(t,\tau), u^*(t,\tau), \tau] = 0 \tag{5.12}$$

$$\dot{x}^*(t,0) = x(t) \tag{5.13}$$

$$\lambda^*(t,T) = \phi_x^T[x^*(t,T)] \tag{5.14}$$

It considers obtaining a solution using the continuation method [7] [8]. The perturbation from an optimal path is described as:

$$\delta\dot{x} = f_x\delta x + f_u\delta u + f_\rho\delta\rho \tag{5.15}$$

$$\delta\dot{\lambda} = -H_{xx}\delta x - f_x^T\delta\lambda - H_{xu}\delta u - H_{x\rho}\delta\rho \tag{5.16}$$

$$H_{ux}\delta x + f_u^T\delta\lambda + H_{uu}\delta u - H_{u\rho}\delta\rho = 0 \tag{5.17}$$

$$C_u\delta u + C_x\delta x = 0 \tag{5.18}$$

Here,δu and $\delta\rho$ are eliminable if Equation(5.17),Equation(5.18) are solved as simultaneous equations. Then Equation(5.15),Equation(5.16) are described as:

$$\frac{d}{dt}\begin{bmatrix}\delta x\\ \delta\lambda\end{bmatrix} = \begin{bmatrix}A & B\\ C & D\end{bmatrix}\begin{bmatrix}\delta x\\ \delta\lambda\end{bmatrix} \tag{5.19}$$

$$\begin{aligned}
A &= f_x + f_\rho\cdot H_{u\rho}^{-1}\cdot H_{ux} - f_\rho\cdot H_{u\rho}^{-1}\cdot H_{uu}\cdot C_u^{-1}\cdot C_x - f_u\cdot C_u^{-1}\cdot C_x\\
B &= f_\rho\cdot H_{u\rho}^{-1}\cdot f_u^T\\
C &= -H_{xx} + H_{xu}\cdot C_u^{-1}\cdot C_x - H_{x\rho}\cdot H_{u\rho}^{-1}\cdot H_{ux} + H_{x\rho}\cdot H_{u\rho}^{-1}\cdot H_{uu}\cdot C_u^{-1}\cdot C_x\\
D &= -f_x^T - H_{x\rho}\cdot H_{u\rho}^{-1}\cdot f_u^T
\end{aligned}$$

The subsequent calculation method follows the continuation method which Ohtsuka and Fujii developed [7] [8]. This is explained briefly below. This technique pursues the optimal solution so that the error F of the transversality conditions of an Euler-Lagrange equation is converged to 0.

$$\frac{d}{dt}F[\lambda(t), x(t), T(t)] = Coeff\cdot F[\lambda(t), x(t), T(t)] \tag{5.20}$$

Thus, F can be stabilized. The equation below is assumed here.

$$\delta\lambda^*(\tau,t) = S^*(t,\tau)\cdot\delta x(t,\tau) + c^*(t,\tau)dt \tag{5.21}$$

This is substituted for Equation(5.19).

$$S_\tau^* = D\cdot S^* - S^*\cdot A - S^*\cdot B\cdot S^* + C \tag{5.22}$$

$$c_\tau^* = (D - S^* \cdot B) \cdot c^* \tag{5.23}$$

This terminal value is acquired from Equation(5.21). An $S^*(t,0), c^*(t,0)$ will be acquired if it finds the integral from the terminal value along time reversely.

$$\dot{\lambda}(t) = S^*(t,0) \cdot \dot{x}(t) + c^*(t,0) \tag{5.24}$$

Then, optimized $\lambda(t)$ will be obtained if it integrates with the upper equation on real time. Also optimized u(t) can be obtained from $H_u = 0$. See Ohtsuka and Fujii [7] [8] for detail.

5.3 Formulation

5.3.1 Model Expression as a Point Mass

Modeling of robot mechanics has been studied for many years. One of the ways is to describe nonlinearity of the robot precisely. However such dynamic equations leads to a result that the equation itself is too much complicated to treat. Such modeling also could not be applicable to even similar type robots. From this point of view, a concept of simple modeling has been emerged. Modeling simply leads to be basic and flexible to apply it to various type controlled objects. The fundamental treatment should be formulated at first, and then applied to more complicated modeling properly. However, overdo of reduction and linearization in modeling leads to many defects mentioned in 5.1 Introduction. The modeling must be simple, but also, must be with wide application.

In this study, when a legged robot is modeled, the whole robot is treated as a point mass of most fundamental case. Treating the whole robot center of gravity as inverted pendulum is a technique generally performed. According to such simple modeling, a method applicable to a biped robot, a quadruped robot, and other multi-legged type robots can be proposed so that Chapter5.7 may describe. First, in order to help understanding, it deals with a problem at a 2-dimensional plane. Chapter5.4.2 describes what extended this to 3 dimensions. As an input of a system, it sets setting up the acceleration of axis "x" and "z",$u_x = \ddot{x}, u_z = \ddot{z}$. Gravity is applied to a perpendicular lower part.

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ z(t) \\ \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ u_x(t) \\ u_z(t) - g \end{bmatrix} \tag{5.25}$$

Although this is considered on Sagittal plane, if it considers at Frontal plane,

$$\frac{d}{dt}\begin{bmatrix} y(t) \\ z(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{y}(t) \\ \dot{z}(t) \\ u_y(t) \\ u_z(t) - g \end{bmatrix} \tag{5.26}$$

This modeling does not include ankle joint torque explicitly. Then the problem mentioned in 5.1 Introduction could not be emerged. The potential of the redundancy of the

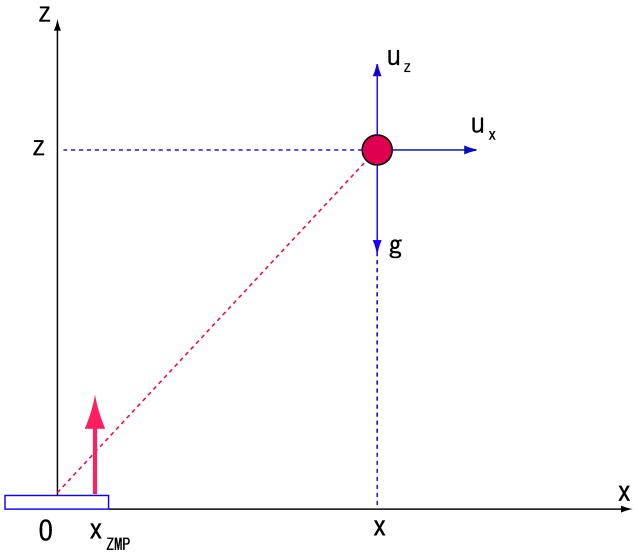


Figure 5.4: 2-D Model in Sagittal Plane

robot could be left and it produces extensive utility. Furthermore, we can obtain ZMP trajectory.

5.3.2 Equality Constraints

Difficulties in taking the ZMP variable into a state equation as a state variable complicates the problem in formulation. Because the right side of Equation(5.2) has the dimension of acceleration, it causes differentiation of acceleration. Then, the use of the equality constraint expressing ZMP eliminates this problem.

$$x_{zmp}(t)(u_z(t) - g) = x(t)(u_z(t) - g) - z(t)u_x(t) \tag{5.27}$$

This means that the total moment by the acceleration inputs and gravity balances with the moment by the ZMP distance and reaction force from the ground like as Fig.5.3. ZMP is the point of satisfying Equation(5.27). If ZMP is located in the polygon constituted by the sole plane, it can support the reaction force without generating any moment. Furthermore, this paper proposes that the 3rd input u_{zmp} substitute for x_{zmp}, then this idea compliments using the ZMP variable in formulation. It is not necessary for u_{zmp} to be included in the state equation.

$$u_{zmp}(t)(u_z(t) - g) = x(t)(u_z(t) - g) - z(t)u_x(t) \tag{5.28}$$

If ZMP is treated as one of the inputs, when an optimum solution is calculated, the optimum ZMP input will be obtained simultaneously. At this point, the equality constraint has an important role.

5.3.3 Performance Index

The performance index is created using norm of inputs. Here, features include that the term of u_{zmp} is added in the performance index. Thus, by setting up the solution , which minimizes the norm of each axial acceleration and the ZMP sway, will be calculated. Considering that the ZMP may not sway within the sole of the robot, this has lead to the design of the robot's sole to be as small as possible.

$$\begin{aligned} J &= X^T \cdot S_f \cdot X + \int_t^{t+T} (X^T \cdot Q \cdot X + U^T \cdot R \cdot U) d\tau \\ X &= [x_f - x(t,T), y_f - y(t,T), \dot{x}_f - \dot{x}(t,T), \dot{y}_f - \dot{y}(t,T)]^T \\ U &= [u_x^*(t,\tau), u_z^*(t,\tau), u_{zmp}^*(t,\tau)]^T \end{aligned} \tag{5.29}$$

S_f and R are diagonal weight matrix.

5.4 Numerical Simulation

5.4.1 Two Dimensional Formulation

The example of numerical simulation is shown in Fig.5.5. The used parameter carried out in 0.5m/s in horizontal speed and the perpendicular direction speed of 0.0m/s in the mass of m = 1.0kg at the initial state and the terminal state. The initial state is $(x, z, \dot{x}, \dot{z})$= (-0.25m, 0.8m, 0.5m/s ,0.0m/s), the terminal state is $(x, z, \dot{x}, \dot{z})$= (0.25m, 0.8m, 0.5m/s ,0.0m/s). R = diag(1.0, 1.0,1.0) and S_f=diag (1.0, 4400.0, 1.0,0.0). trajectory "x,z,$\dot{x}$, $\dot{z}$" follows on moving to the terminal state of Fig.5.5.1-5.5.4, and signs that the ZMP input "u_{zmp}" also transit as Fig.5.5.7. u_x, u_z are also obtained as Fig.5.5.5,Fig.5.5.6. In this example, the action of the legged robot which advances 0.5m in 1.0s at 0.8m height is exaggerated, but this is to be understandable. Even in such a big action the optimum the ZMP input can also be generated by this technique. Fig.5.5.9,Fig.5.5.10 comparing the left side and right side of the equality constraint equations of the following to see if the solution is suitable.

$$u_{zmp}(t)(u_z(t) - g) = x(t)(u_z(t) - g) - z(t)u_x(t) \tag{5.30}$$

Balance is mostly maintained by the Fig.5.5.9,Fig.5.5.10. It can be understood that the optimum ZMP input is generated appropriately. The CPU time taken for this calculation is 0.09s by a Linux OS PC with a Celeron processor 333MHz. It can be completed in a sufficiently short time to 1.0s of the simulation time. The control interval is 2ms. It is understood that real-time control is possible. Since the ZMP input is generable on real time, as compared with the conventional technique that the optimum ZMP trajectory is beforehand generated by off-line calculation, it is very convenient. Fig.5.5.8 is the transition of the value of error F from the "transversality condition". The optimum solution is achieved as this value F is close to 0. Although the value F overshoots from 0, the value return to 0 quickly in this figure. This figure provides the evidence that the optimum solution is achieved.

Table 5.1: Simulation Data(2-D Case)

Simulation Time	1.0 s
dt	2.0 ms
Continuation Terminal Time	T=0.2
Initial Condition	$[0.0, 0.0, 0.8, 0.5, 0.0]^T$
Terminal Condition	$[0.25, 0.8, 0.5, 0.0]^T$
S_f	diag[1,540,1,1]
R	diag[1,1,1]
Q	diag[0,0,0,0]
Number of time steps of tau axis	5
ζ	450

$$F[\lambda(t), x(t), T(t)] = \lambda^*(t, T) - \phi_x^T[x^*(t, T)] \tag{5.31}$$

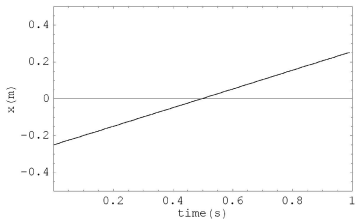


Fig5.5.1 x(2-D case)

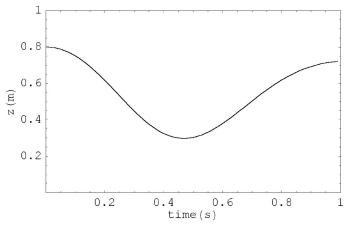


Fig5.5.2 z(2-D case)

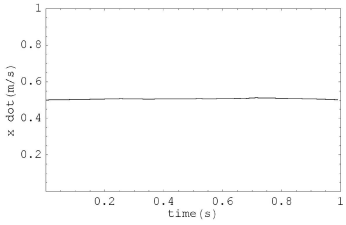


Fig5.5.3 $\dot{x}$(2-D case)

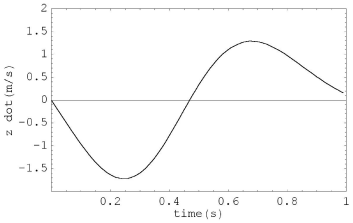


Fig5.5.4 $\dot{z}$(2-D case)

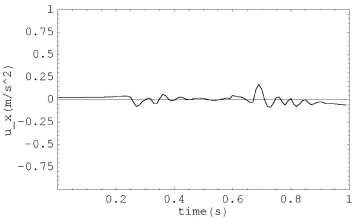


Fig5.5.5 u_x(2-D case)

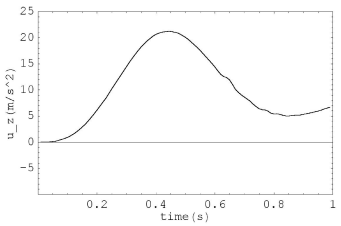


Fig5.5.6 u_z(2-D case)

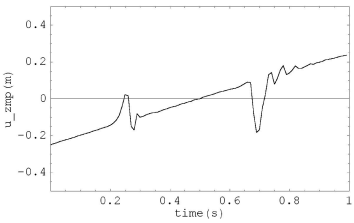


Fig5.5.7 u_{ZMP}(2-D case)

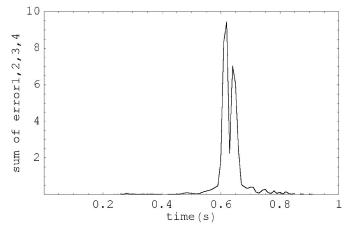


Fig5.5.8 Sum of errors (2-D case)

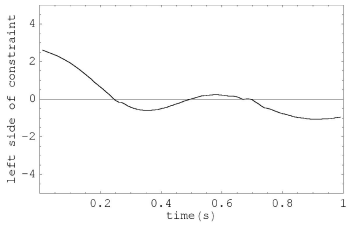


Fig5.5.9 Left side of equal constraint(2-D case)

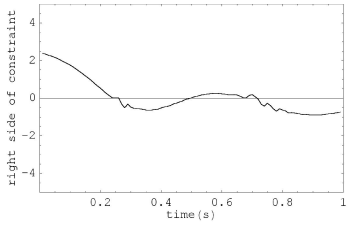


Fig5.5.10 Right side of equal constraint(2-D case)

5.4.2 Three Dimensional Formulation

Although the preceding chapter stated at the two dimensional plane in order to give intelligible explanation, formulation in the three dimensional space is also available. The state equation of the three dimensional mass is described as,

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \\ u_x(t) \\ u_y(t) \\ u_z(t) - g \end{bmatrix} \tag{5.32}$$

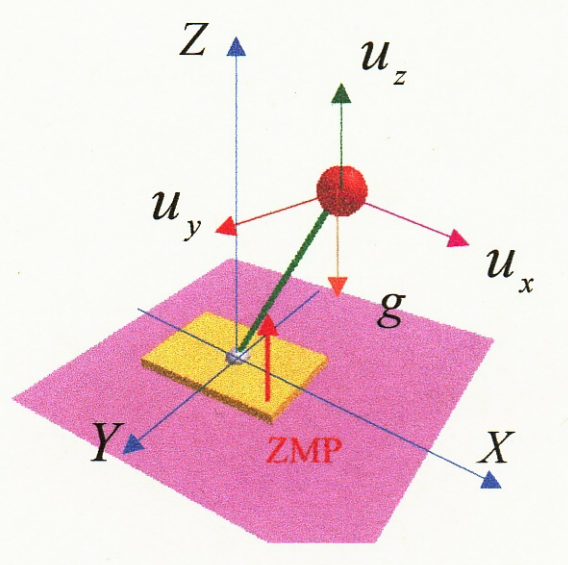


Figure 5.6: 3 Dimensional Formulation

The equality constraint of the ZMP balance along the x axis and the balance along the y axis must be taken into consideration.

$$x_{zmp}(t)(u_z(t) - g) = -u_x(t)z(t) + (u_z(t) - g)x(t) \tag{5.33}$$

$$y_{zmp}(t)(u_z(t) - g) = -u_y(t)z(t) + (u_z(t) - g)y(t) \tag{5.34}$$

These two equations are collected in order to reduce the number of the equality constraint.

$$u_x(t)z(t)(y_{zmp}(t) - y(t)) = u_y(t)z(t)(x_{zmp}(t) - x(t)) \tag{5.35}$$

The performance index makes the minimum norm of each axial acceleration and ZMP inputs.

$$J = X^T \cdot S_f \cdot X + \int_t^{t+T} (X^T \cdot Q \cdot X + U^T \cdot R \cdot U) d\tau$$

$$U = [u_x^*(t,\tau), u_y^*(t,\tau), u_z^*(t,\tau), u_{x_{zmp}}^*, u_{y_{zmp}}^*]^T$$

$$\begin{aligned} X = [&x_f - x(t,T), y_f - y(t,T), z_f - z(t,T), \\ &\dot{x}_f - \dot{x}(t,T), \dot{y}_f - \dot{y}(t,T), \dot{z}_f - \dot{z}(t,T)]^T \end{aligned} \tag{5.36}$$

The calculation result of this 3-D formulation is shown in Fig.5.7. The initial state ($x, y, z, \dot{x}, \dot{y}, \dot{z}$)= (-0.125m, -0.125m, 0.800m, 0.500m/s, 0.500m/s, 0.000m/s) and the terminal state($x, y, z, \dot{x}, \dot{y}, \dot{z}$)= (0.125m, 0.125m, 0.800m, 0.500m/s, 0.500m/s,0.000m/s). R = diag (1.0, 1.0, 1.0, 1.0, 1.0) and S_f = diag (100.0, 600.0, 3400.0, 0.1, 2.0, 0.1). Fig.5.7.1-Fig.5.7.4 show the transitions of state variables on each axes. Fig.5.7.7-Fig.5.7.9 show transitions of the inputs of each axes. Fig.5.7.10 and Fig.5.7.11 show ZMP inputs along axes x and y. The figures show signs that it has fitted in less than ± 0.05m on the x axis, and less than ± 0.1m on the y axis. ZMP sway inside this range means that the appropriate sole for the leg can be designed. Moreover, Fig.5.7.12 and Fig.5.7.13 and show the left side value and right side value of Equation(5.33). These results fulfill the equality constraint for the most part. Error F is suppressed well(Fig.5.7.14). This means that the calculation result follows the optimum path well. The calculation time required is 0.35s for this simulation time of 0.5s. In this calculation, the time unit of integration was set to 200 μs.

Table 5.2: Simulation Data(3-D Case)

Simulation Time	0.5 s
dt	2.0 ms
Continuation Terminal Time	T=0.2
Initial Condition	$[0.0, 0.0, 0.8, 0.25, 0.25, 0.0]^T$
Terminal Condition	$[0.125, 0.125, 0.8, 0.25, 0.25, 0.0]^T$
S_f	diag[100,600,3400,0.1,2,0.1]
R	diag[1,1,1,51,1]
Q	diag[0,0,0,0,0,0]
Number of time steps of *tau* axis	5
ζ	4500

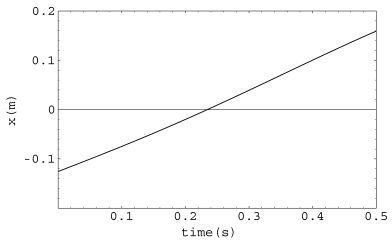


Fig5.7.1 x(3-D Case)

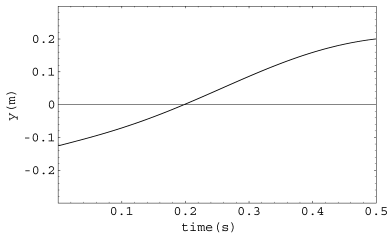


Fig5.7.2 y(3-D Case)

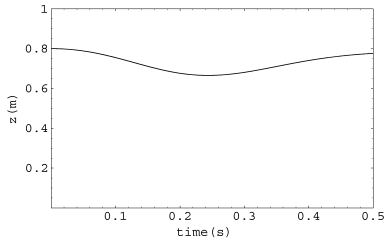


Fig5.7.3 z(3-D Case)

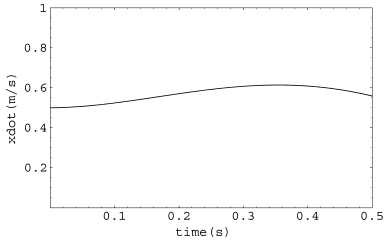


Fig5.7.4 $\dot{x}$(3-D Case)

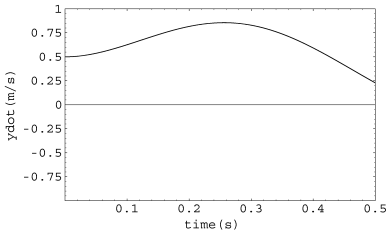


Fig5.7.5 $\dot{y}$(3-D Case)

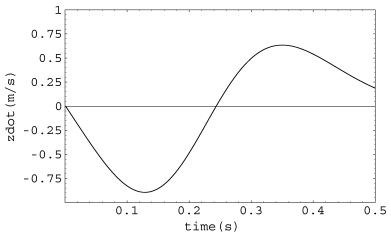


Fig5.7.6 $\dot{z}$(3-D Case)

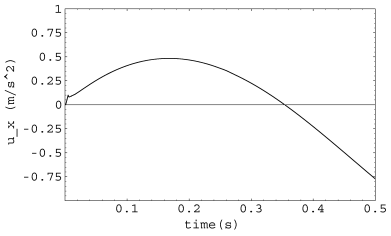


Fig5.7.7 u_x(3-D Case)

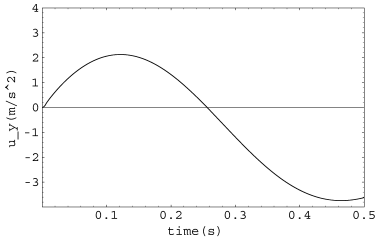


Fig5.7.8 u_y(3-D Case)

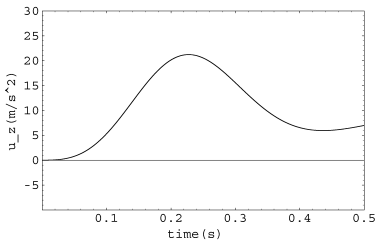


Fig5.7.9 u_z(3-D Case)

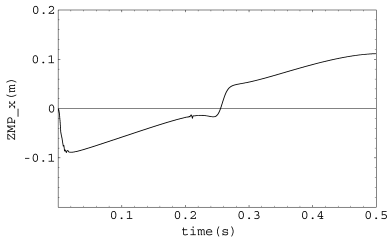


Fig5.7.10 u_{ZMP_x}(3-D Case)

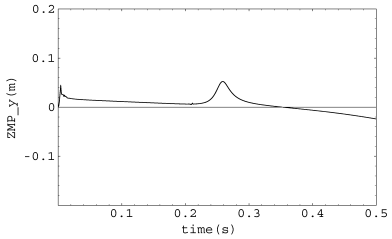


Fig5.7.11 u_{ZMP_y}(3-D Case)

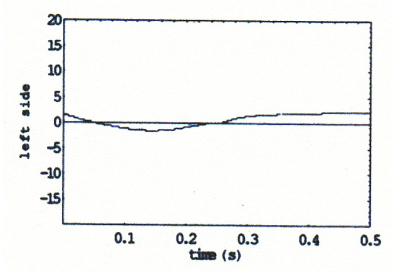


Fig5.7.12 Left side of the equal constraint(3-D Case)

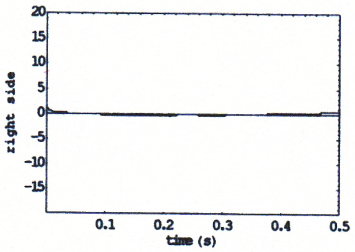


Fig5.7.13 Right side of the equal constraint(3-D Case)

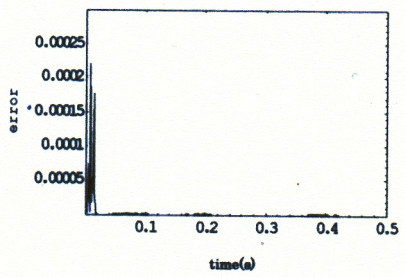


Fig5.7.14 Sum of Error1-6(3-D Case)

5.5 Mass Behavior and ZMP

The ZMP balance condition varies with the mass movement. Here let's confine the mass movement solely to the x-axis like as Fig.5.4. The ZMP balance condition in case of $x < 0$, the mass is pulled by gravity and thrust force along x axis is needed. However, the condition in case of $x > 0$, the mass is pushed by gravity and the movement along x-axis is easy. Some simulation results reflect such mass behavior.

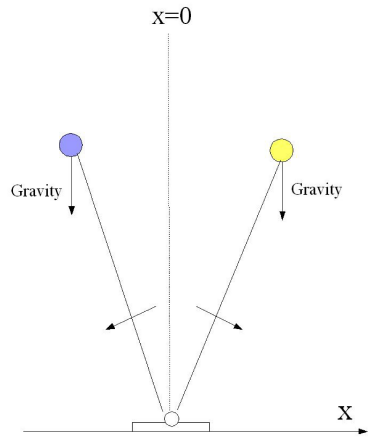


Fig5.7.15 Mass Behavior Ahead x=0/Behind x=0

5.5.1 In case of $x > 0$

If the movement of the mass starts from $x > 0$ and the velocity of the movement is constant, the ZMP balance condition does not fluctuate so much. This case is base of another cases.

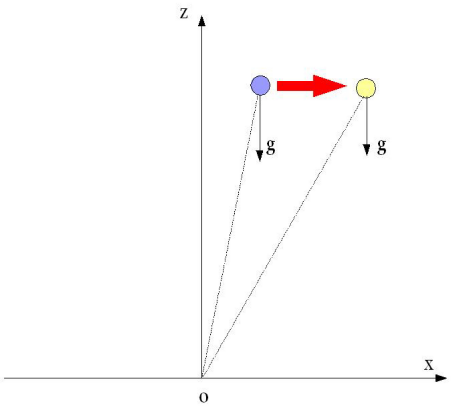


Fig5.7.16　In Case of $x > 0$

Table 5.3: Simulation Data(In case of $x > 0$)

Simulation Time	1.0 s
dt	2.0 ms
Continuation Terminal Time	T=0.5
Initial Condition	$[0.0, 0.0, 0.8, 0.5, 0.0, 0.0]^T$
Terminal Condition	$[0.5, 0.0, 0.8, 0.5, 0.0, 0.0]^T$
S_f	diag[1,1,100,0.1,0.1,100]
R	diag[1,1,1,5,5]
Q	diag[1,1,1,1,1,1]
Number of time steps of *tau* axis	50
ζ	300

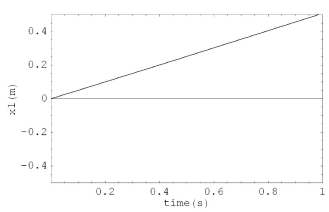


Fig5.8.1 x(In Case of $x > 0$)

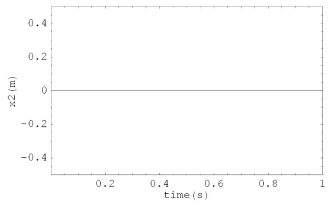


Fig5.8.2 y(In Case of $x > 0$)

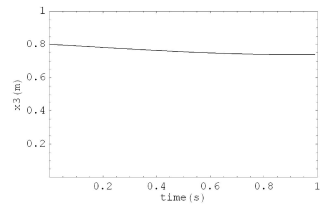


Fig5.8.3 z(In Case of $x > 0$)

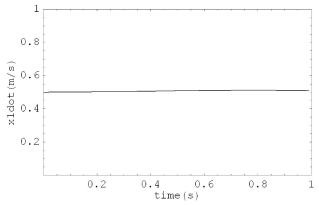


Fig5.8.4 $\dot{x}$(In Case of $x > 0$)

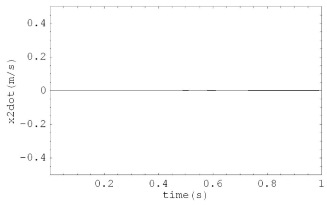


Fig5.8.5 $\dot{y}$(In Case of $x > 0$)

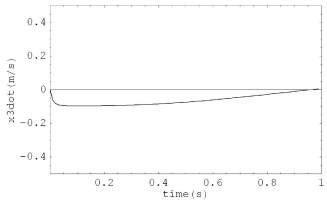


Fig5.8.6 $\dot{z}$(In Case of $x > 0$)

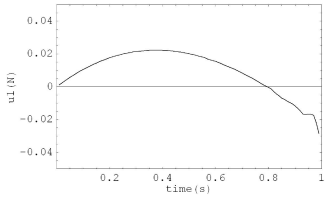


Fig5.8.7 u_x(In Case of $x > 0$)

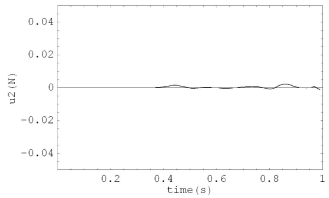


Fig5.8.8 u_y(In Case of $x > 0$)

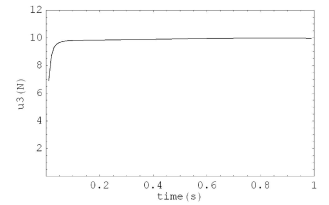


Fig5.8.9 u_z(In Case of $x > 0$)

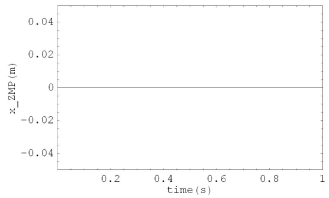


Fig5.8.10 u_{ZMP_x}(In Case of $x > 0$)

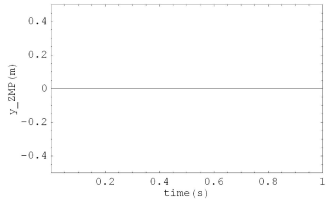


Fig5.8.11 u_{ZMP_y}(In Case of $x > 0$)

Table 5.4: Simulation Data(Cut Across x=0)

Simulation Time	1.0 s
dt	2.0 ms
Continuation Terminal Time	T=0.5
Initial Condition	$[-0.25, 0.0, 0.8, 0.5, 0.0, 0.0]^T$
Terminal Condition	$[0.25, 0.0, 0.8, 0.5, 0.0, 0.0]^T$
S_f	diag[1,1,100,0.1,0.1,100]
R	diag[1,1,1,5,5]
Q	diag[1,1,1,1,1,1]
Number of time steps of *tau* axis	50
ζ	300

5.5.2 In case of Cut Across x=0

If the mass cuts across $x = 0$, effect of the ZMP balance condition alternates from $x < 0$ to $x > 0$. u_x increases in $x < 0$, however u_x decrease immediately after the mass acrossed $x = 0$. u_{ZMP} also perturbs along with the effect of u_x.This behavior is very interesting point about legged robot. At the anterior half of the support phase, the robot needs acceleration. However at the last half of the support phase, the robot can draw on gravity.

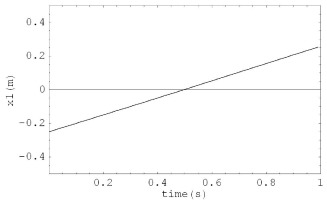


Fig5.9.1 x(Across $x = 0$)

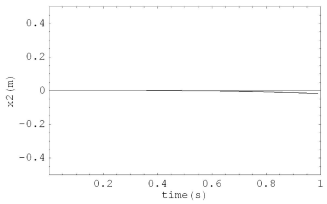


Fig5.9.2 y(Across $x = 0$)

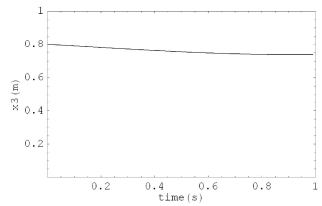


Fig5.9.3 z(Across $x = 0$)

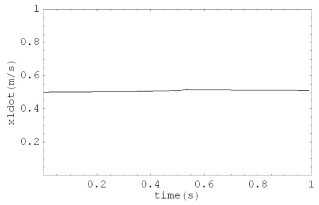


Fig5.9.4 $\dot{x}$(Across $x = 0$)

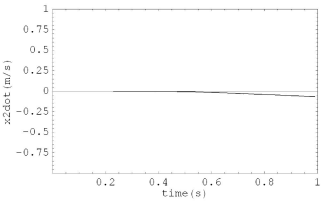


Fig5.9.5 $\dot{y}$(Across $x = 0$)

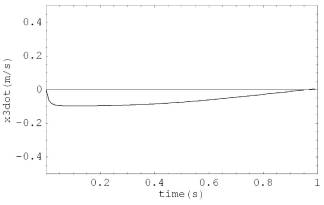


Fig5.9.6 $\dot{z}$(Across $x = 0$)

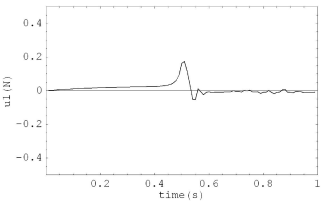


Fig5.9.7 u_x(Across $x = 0$)

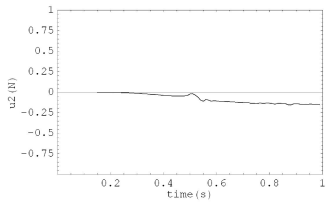


Fig5.9.8 u_y(Across $x = 0$)

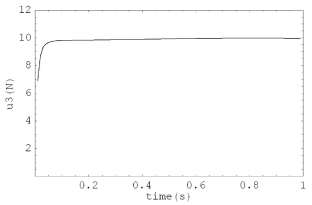


Fig5.9.9 u_z(Across $x = 0$)

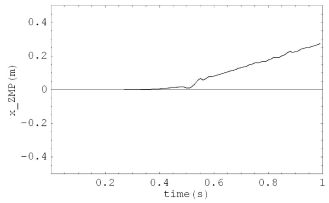


Fig5.9.10 u_{ZMP_x}(Across $x = 0$)

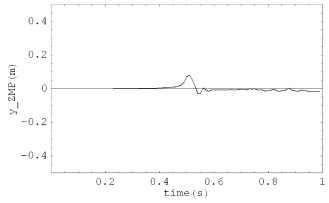


Fig5.9.11 u_{ZMP_y}(Across $x = 0$)

Table 5.5: Simulation Data(Sudden Acceleration)

Simulation Time	1.0 s
dt	2.0 ms
Continuation Terminal Time	T=0.5
Initial Condition	$[-0.25, 0.0, 0.8, 0.5, 0.0, 0.0]^T$
Terminal Condition	$[0.25, 0.0, 0.8, 0.0, 0.0, 0.0]^T$
S_f	diag[10,1,100,1200,0.1,100]
R	diag[5,1,1,5,5]
Q	diag[1,1,1,1,1,1]
Number of time steps of *tau* axis	50
ζ	300

5.5.3 Sudden Acceleration

In this case the mass is suddenly accelerated from $\dot{x} = 0.0$ to $\dot{x} = 0.5$(Fig.5.10.4,Fig.5.10.7). Accompanying the process of sudden acceleration, ZMP perturbs wildly. Fig.5.10.10 shows that ZMP_x trajectory comes up to 4m.

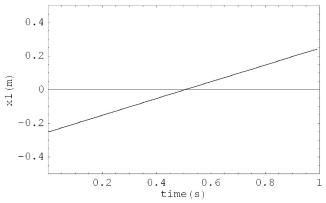


Fig5.10.1 x(Sudden Acceleration)

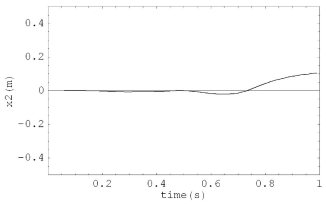


Fig5.10.2 y(Sudden Acceleration)

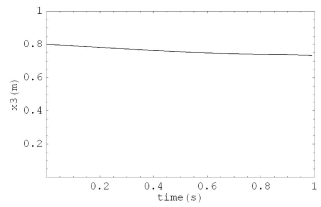


Fig5.10.3 z(Sudden Acceleration)

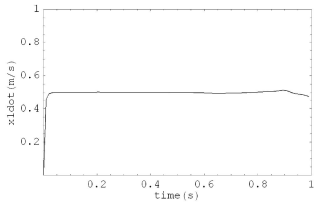


Fig5.10.4 $\dot{x}$(Sudden Acceleration)

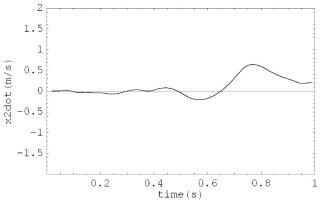


Fig5.10.5 $\dot{y}$(Sudden Acceleration)

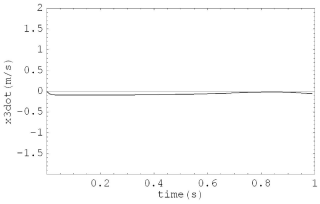


Fig5.10.6 $\dot{z}$(Sudden Acceleration)

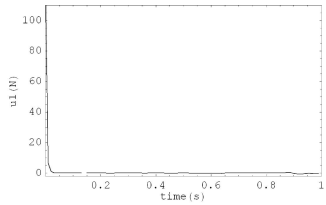


Fig5.10.7 u_x(Sudden Acceleration)

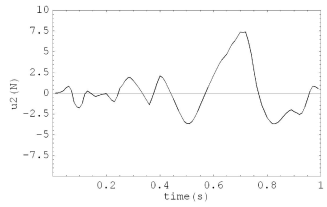


Fig5.10.8 u_y(Sudden Acceleration)

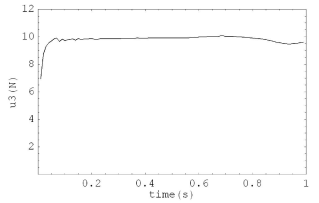


Fig5.10.9 u_z(Sudden Acceleration)

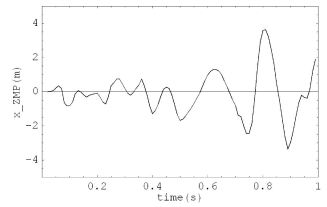


Fig5.10.10 u_{ZMP_x}(Sudden Acceleration)

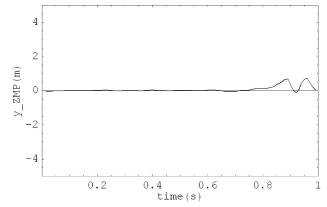


Fig5.10.11 u_{ZMP_y}(Sudden Acceleration)

Table 5.6: Simulation Data(Sudden Stop)

Simulation Time	1.0 s
dt	2.0 ms
Continuation Terminal Time	T=0.5
Initial Condition	$[-0.25, 0.0, 0.8, 0.5, 0.0, 0.0]^T$
Terminal Condition	$[-0.25, 0.0, 0.8, 0.0, 0.0, 0.0]^T$
S_f	diag[10,1,100,1200,0.1,100]
R	diag[5,1,1,5,5]
Q	diag[1,1,1,1,1,1]
Number of time steps of *tau* axis	50
ζ	300

5.5.4 Sudden Stop

In this case the point mass has the velocity 0.5m/s at first, and rapidly decelerated from $\dot{x} = 0.5$ to $\dot{x} = 0.0$. The trajectory x does not move ahead(Fig.5.11.1). u_x generates minus acceleration. Accompanying the process of sudden deceleration, ZMP perturbs wildly(Fig.5.11.10).

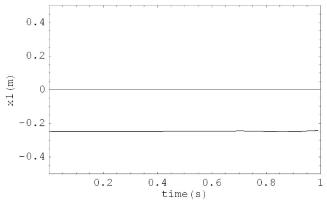


Fig5.11.1 x(Sudden Stop)

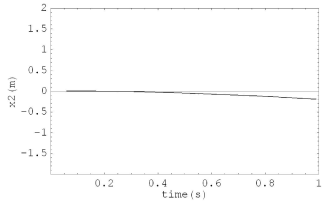


Fig5.11.2 y(Sudden Stop)

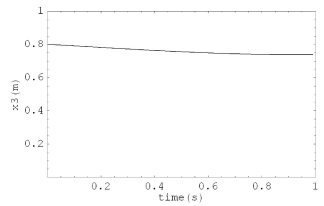


Fig5.11.3 z(Sudden Stop)

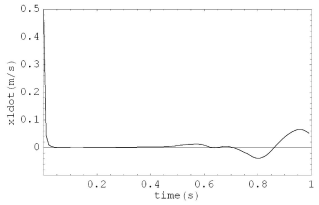


Fig5.11.4 $\dot{x}$(Sudden Stop)

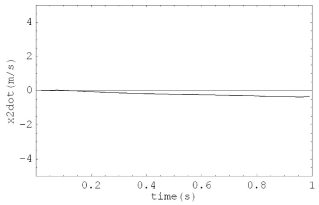


Fig5.11.5 $\dot{y}$(Sudden Stop)

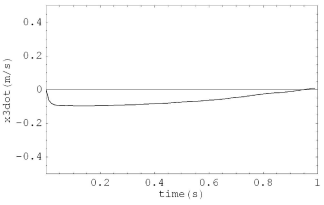


Fig5.11.6 $\dot{z}$(Sudden Stop)

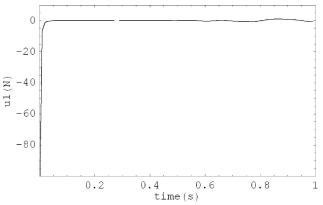


Fig5.11.7 u_x(Sudden Stop)

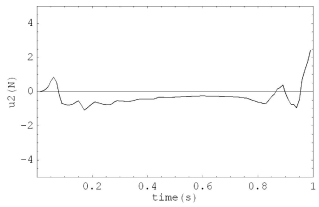


Fig5.11.8 u_y(Sudden Stop)

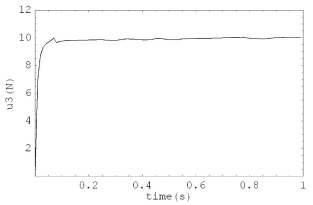


Fig5.11.9 u_z(Sudden Stop)

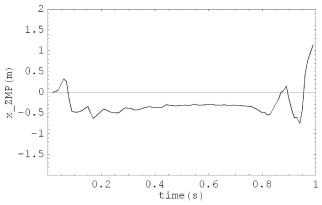


Fig5.11.10 u_{ZMP_x}(Sudden Stop)

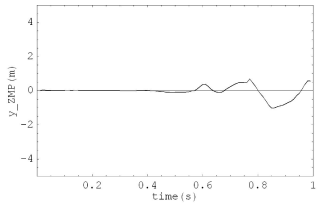


Fig5.11.11 u_{ZMP_y}(Sudden Stop)

Table 5.7: Simulation Data(Make u_{ZMP} small)

Simulation Time	1.0 s
dt	2.0 ms
Continuation Terminal Time	T=0.5
Initial Condition	$[-0.25, 0.0, 0.8, 0.5, 0.0, 0.0]^T$
Terminal Condition	$[0.25, 0.0, 0.8, 0.5, 0.0, 0.0]^T$
S_f	diag[1,1,100,0.1,0.1,100]
R	diag[1,1,1,10,10]
Q	diag[1,1,1,1,1,1]
Number of time steps of tau axis	50
ζ	300

5.6 Weight Matrix

We can make some inputs value small using weight matrix R, however this is trade-off between another variables. Everything could not be small. If weight matrix R makes ZMP input small, the trajectory of u_x fluctuates.

A simulation case which makes weight of ZMP inputs 10 shows Fig.5.12. The trajectory of ZMP input comes to be $0.04 \geq \|u_{ZMP_x}\|$. However, the trajectory of u_x is transformed.

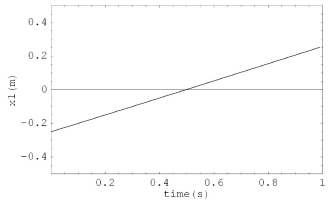


Fig5.12.1 x(Make u_{ZMP} small)

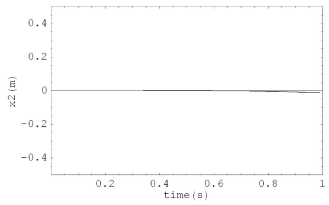


Fig5.12.2 y(Make u_{ZMP} small)

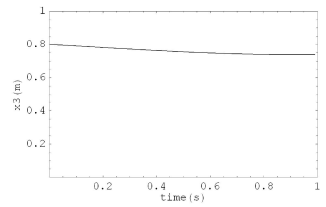


Fig5.12.3 z(Make u_{ZMP} small)

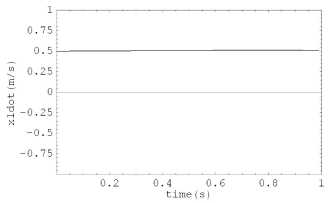


Fig5.12.4 $\dot{x}$(Make u_{ZMP} small)

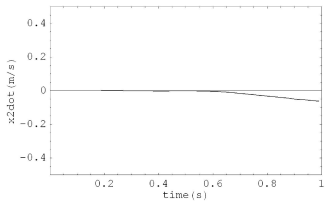


Fig5.12.5 $\dot{y}$(Make u_{ZMP} small)

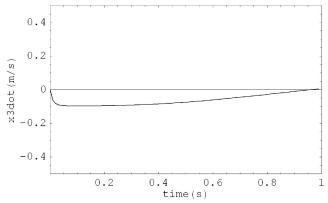


Fig5.12.6 $\dot{z}$(Make u_{ZMP} small)

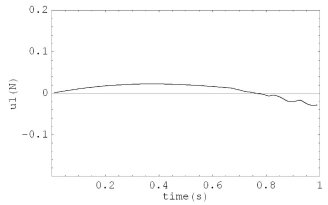


Fig5.12.7 u_x(Make u_{ZMP} small)

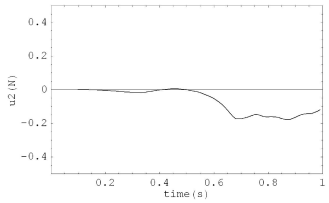


Fig5.12.8 u_y(Make u_{ZMP} small)

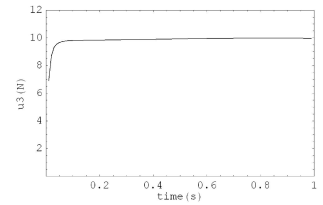


Fig5.12.9 u_z(Make u_{ZMP} small)

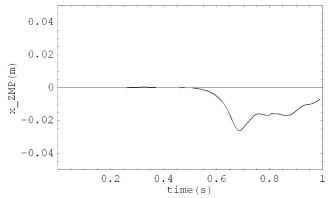


Fig5.12.10 u_{ZMP_x}(Make u_{ZMP} small)

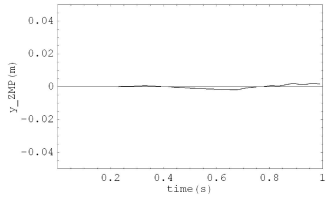


Fig5.12.11 u_{ZMP_y}(Make u_{ZMP} small)

5.7 Application

A legged robot is generally a nonlinear mechanical multi-link system. In order to compensate such nonlinearity, using this technique together with the conventional nonlinear control technique is also worth consideration. The real-time optimum solution using Receding Horizon Control can be used as reference trajectories. On the other hand, nonlinear control in modern control theory performs control of the whole multi-link system of the robot to converge to the reference trajectories. When we have a powerful CPU for real-time control, it is possible that it formulates everything in a nonlinear model using Receding Horizon Control alone. Using the conventional control method for the legged robot, the optimum reference trajectory have to be prepared before the real-time control. If the robot encounters an unexpected situation, the robot cannot cope with it. In regards to this point, the technique proposed is practical.
The optimum ZMP input, which is obtained using Receding Horizon Control, can generate the reference trajectory. Then a control system block diagram could be constructed so that actual ZMP may follow this reference trajectory. Since u_x, u_y, and u_z are obtained for optimum inputs, they are newly regarded as reference trajectories. Thus, this simple method proposed is applicable also to a biped robot, a quadruped robot, and other multiped robots.

5.7.1 In the Case of a Biped

In the case of the biped, the phase of leg behavior could be divided into a support phase and a swing phase. The ZMP must be respectively settled in the sole planes. The ZMP must transition in these planes, if the initial point and the terminal point are set as in Fig.5.13. Control of the nonlinear force is performed in the nonlinear control part as is shown in the Fig.5.14 control block diagram. Collecting the calculation results in which each parameter is changed makes it possible to predict the range of ZMP sway in a sole. This is a very interesting point. In design of a biped robot, an index can be obtained which has an influence on the design dimensions of the sole plane.

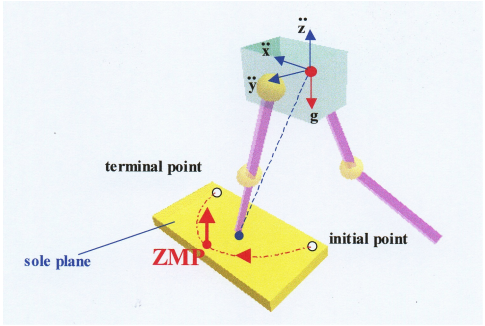


Figure 5.13: Case of Biped

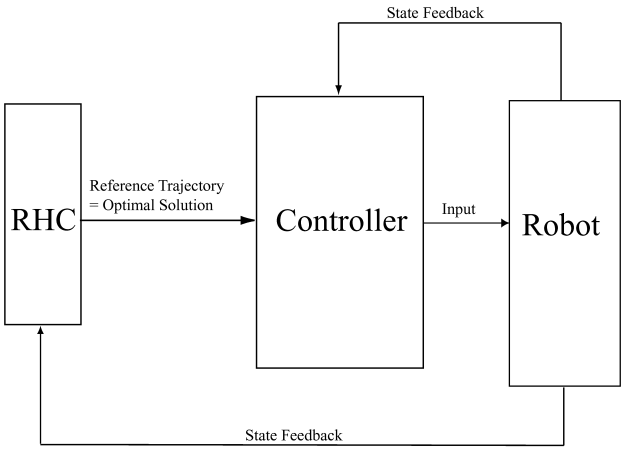


Figure 5.14: Control Block Diagram

5.7.2 In the Case of a Quadruped

The legs can be treated as the support legs pair and the swing legs pair by turns in ideal gaits. The polygon which is constituted by some sets of legs can secure a larger plane compared with a biped robot's case. For example, when a quadruped robot performs a "trot gait", it is necessary to repeat a support phase - swing phase for the diagonal leg entirely by turns. In Fig.5.15 , step planes cross in the trot gait. If the starting point of ZMP is set at the tip of this plane, it can run to the tip of the following plane. Then a stable locomotion pattern will be achieved.

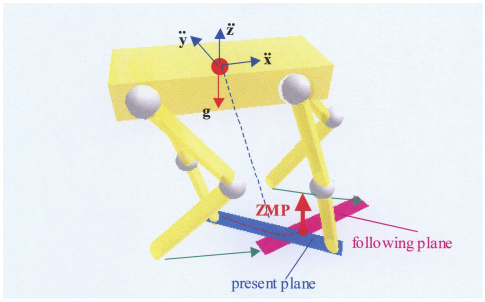


Figure 5.15: Case of Quadruped

5.8 Conclusion

It is proposed that Receding Horizon Control is used adding an equality constraint and then the formulation is performed. We discussed that the ZMP condition is used in the formulation using an equality constraint. Generally, it is difficult to take a ZMP variable as a state variable in a state equation. To ensure an optimal result, careful consideration of the ZMP conditions is required for the formulation. Then, this paper

shows that the ZMP is defined as one of the input variables. Since ZMP input is obtained as an optimal solution, the technique proposed can generate a true ZMP trajectory on real time. Simulation results are performed by both the models in a two dimensional plane and the three dimensional space.

This simple modeling enables an easy application to biped, quadruped and the other multiple robots. This simple modeling forms a basis for robot control.

Chapter 6

State Variable Inequality Constraint of Swing Leg

6.1 Introduction

The point mass modeling for formulation of Receding Horizon Control is introduced in the preceding chapter. That idea is to look at the robot model in perspective. We are in the next stage how the swing legs should be treated. The legged robot motion can be categorized into two phases support leg phase and swing leg phase depending on whether or not they are in contact with the floor while the legged robot is moving. The support legs support the robot against gravity while the swing legs are swung forward and then prepared for the subsequent support phase. Since a real robot's legs behave nonlinearly, nonlinear forces interfere with the motion of the swing legs when they move. It is not practical to include all nonlinearities of the robot in a state equation. Therefore, this chapter proposes a simple formulation that reflects the nonlinearity of the motion of swing legs as far as possible. The root joint of the swing leg is connected to the center of gravity of the robot in sagittal plane and the acceleration is transferred to the root joint while the center of gravity satisfies the conditions of the ZMP constraint. This formulation can be easily applied to multi-legged robots such as biped, quadruped, and hexapod by simple addition of the equations of motion of the swing legs. It is necessary to apply the constraint on the swing legs such that their position is always above floor level. The absence of this constraint could create a situation in which the legs would be positioned below the floor. Constraints such as these can be described as inequality state variable constraints. This chapter describes a method for solving the problem by means of the slack variable method.

6.2 Modeling of Swing Leg

Generally, a mathematical model of a legged robot increases in complexity if we try to describe the nonlinearity of its movement precisely, with the result that the equations of motion become too complex for the robot control to apply. A mathematical model that requires less calculation is needed to shorten the control interval and allow real-time control of the swing legs. [28] describes the ZMP condition on the sagittal plane Fig.5.4

using equal constraint.

$$x_{ZMP}(t)(u_z(t)-g) = x(t)(u_z(t)-g) - z(t)u_x(t) \tag{6.1}$$

This indicates that the moment, which consists of x_{ZMP} and the reaction force of the floor, balances with the gross moment around the origin of the coordinate system. Kinetic balance is maintained as long as x_{ZMP} is in the area of the sole (biped), or in the area of the polygon described by the points where the toes of the grounding legs touch the floor (quadruped). The state equation of the center of gravity of a robot in sagittal plane is described as follows.

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ z(t) \\ \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ u_x(t) \\ u_z(t)-g \end{bmatrix} \tag{6.2}$$

At practical control of legged robot, the control of swing legs must also be taken into account. We have therefore described the swing legs as a nonlinear two-link coordinate system and appended it to the state equation of the center of gravity of the robot. This approach is based on the assumption that the root joint of the swing leg will receive the acceleration of the center of gravity of the whole robot.

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ z(t) \\ \theta_1(t) \\ \theta_2(t) \\ \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta_1}(t) \\ \dot{\theta_2}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta_1}(t) \\ \dot{\theta_2}(t) \\ u_x(t) \\ u_z(t)-g \\ M^{-1}(\Theta)(u(t)-V(\Theta,\dot{\Theta})-G(\Theta)) \end{bmatrix} \tag{6.3}$$

$\Theta = [\theta_1, \theta_2]^T$

Where M is the inertia matrix, V is the coriolis term, and G is the gravity term. The acceleration of inertia originating in the acceleration of the center of gravity of the whole robot is input to the acceleration of the root joint of the swing leg at the first step of the Newton-Euler method. We set it at $[u_x, u_z - g]$ in the Newton-Euler method for swing leg. Equation(6.3) can be easily expanded to biped, quadruped, and another type of legged robot by appending nonlinear equations of motion of swing legs to the state equation of the center of gravity of the robot. In this method,

1. The nonlinearity of swing legs can be taken into account.
2. The state equation is simple and practical.
3. The equations for motion of swing legs are easily appended to state equations. Therefore, this formulation can be extended to multi-legged robots.

These enable us to make feasible formation.

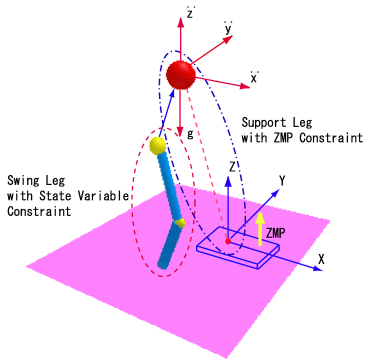


Figure 6.1: Image of formulization

6.3 Performance Index

Equation(6.4) is used as an evaluation function.

$$J = \phi[x(t+T)] + \frac{1}{2}\int_t^{t+T} (x(\tau)-x_f)^T \cdot Q \cdot (x(\tau)-x_f) + (u(\tau)-u_f)^T \cdot R \cdot (u(\tau)-u_f)d\tau \tag{6.4}$$

$$\phi[x(t+T)] = (x(t+T)-x_f)^T \cdot S_f \cdot (x(t+T)-x_f)$$

$$(u(\tau)-u_f)^T \cdot R \cdot (u(\tau)-u_f) = r_1 u_x^2(\tau) + r_2(u_z(\tau)-g)^2 + r_3 u_{ZMP}^2(\tau) + r_4 u_{\theta 1}^2(\tau) + r_5 u_{\theta 2}^2$$

The first term is a terminal constraint. Including u_{zmp} in the performance index is a novel approach which allows us to obtain a solution that minimizes input of acceleration in each axis direction, input of each joint torque of swing leg, and norm of ZMP. It reduces sway of ZMP in the area of the robot sole (biped) or in the area of the polygon whose vertexes are described by the toes of the grounding legs (quadruped).

6.4 Numerical Calculation

Fig.6.2 shows one of results of simulation of this formulation. The link parameters used in the simulation are given in Table6.1 and explanations of the link parameters are given in Fig.6.3. The parameters used in optimization are shown in Table6.2. The stick diagram Fig.6.2.14 shows correct movement of the swing leg occurring by chance in this case, in spite of no floor level constraint being imposed. Slight sway of the ZMP is appeared because the time step on the τ axis is rough(Fig.6.2.13). The simulation time was set at 1.0s and its calculation took 0.9s. The control interval was set at 2 ms. A personal

Table 6.1: Two Links Mechanical Parameters

l_1	0.5m
l_2	0.5m
l_{1c}	0.25m
l_{2c}	0.25m
m_1	0.5kg
m_2	0.5kg

Table 6.2: Parameters for Simulation(without constraint)

S_f	diag[50.0,50.0,15.0,15.0,100.0,10.0,1.0e^{-4},1.0e^{-4}]
Q	diag[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
R	diag[1.0,2.0,1.0,1.0,6.0]
x_0	$[0.0, 1.0, -1.7, -0.6, 0.2, 0.0, 0.0, 0.0]^T$
x_f	$[0.2, 1.0, -1.2, -1.6, 0.2, 0.0, 0.0, 0.0]^T$
u_f	$[0.0, 9.8, 0.0, 0.0, 0.0]^T$
T_f	0.1s
Δt	2ms
Number of divide	5
ζ	450

computer with a 333MHz Celeron processor and running a Linux OS was employed. The result shows that this type of computer enables real-time calculation and that there is ample time for real-time calculation of the equation of motion for a nonlinear 2-link system case.

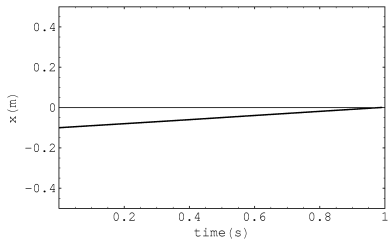


Fig6.2.1 x(No Constraint Case)

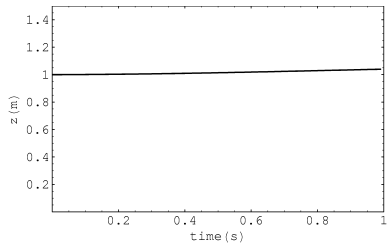


Fig6.2.2 z(No Constraint Case)

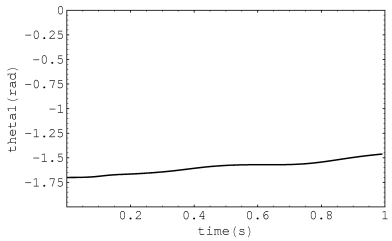


Fig6.2.3 θ_1(No Constraint Case)

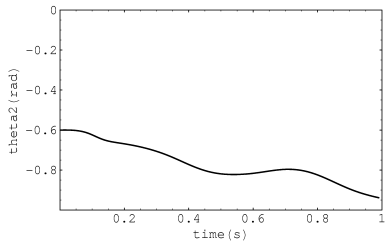


Fig6.2.4 θ_2(No Constraint Case)

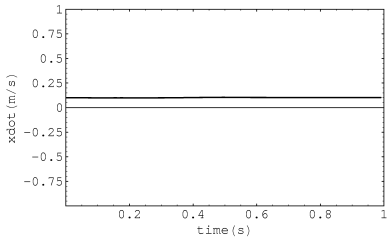


Fig6.2.5 $\dot{x}$(No Constraint Case)

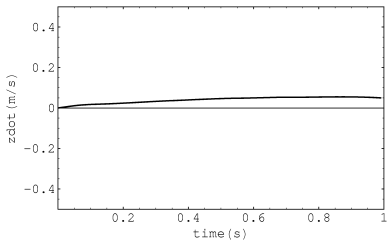


Fig6.2.6 $\dot{z}$(No Constraint Case)

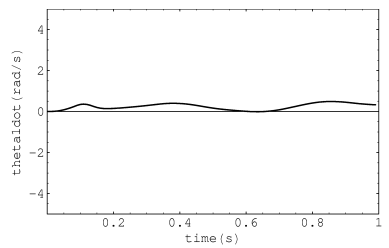


Fig6.2.7 $\dot{\theta}_1$(No Constraint Case)

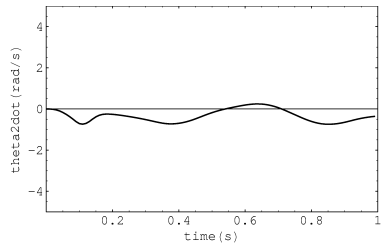


Fig6.2.8 $\dot{\theta}_2$(No Constraint Case)

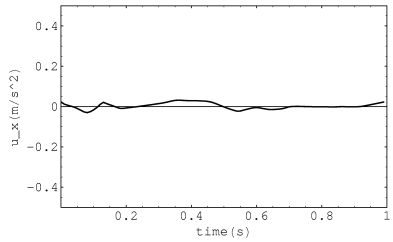


Fig6.2.9 u_x(No Constraint Case)

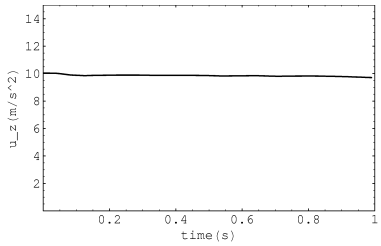


Fig6.2.10 u_z(No Constraint Case)

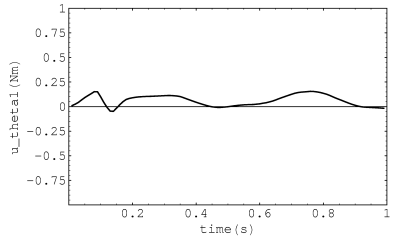


Fig6.2.11 u_{θ_1}(No Constraint Case)

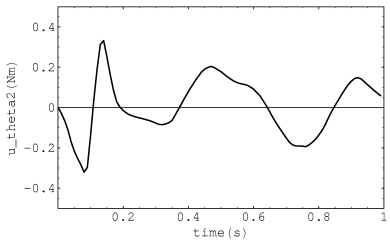


Fig6.2.12 u_{θ_2}(No Constraint Case)

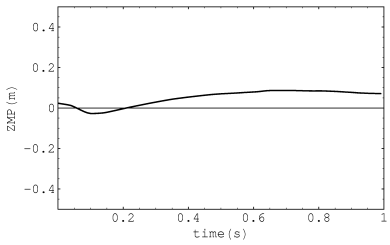


Fig6.2.13 u_{ZMP}(No Constraint Case)

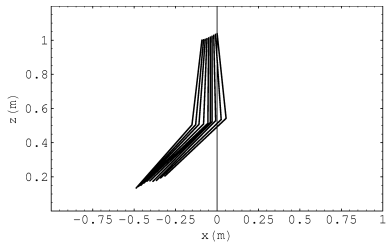


Fig6.2.14 Stick Figure(No Constraint Case)

6.5 Constraint for Swing Legs

6.5.1 State Variable Constraint

The constraint that the position of the swing leg must be above floor level is required if the motion of swing legs is applied to the formulation. This section examines how this condition is applied to the formulation. According to Fig.6.3, the condition where the position of the tip of the swing leg is above the floor is described as an inequality constraint.

$$height + l_1 sin\theta_1(t) + l_2 sin(\theta_1 + \theta_2) \geq 0 \tag{6.5}$$

This is the state variable inequality constraint. This problem is difficult to handle since the optimal path must enter tangentially onto a constrained arc. The control variable u is not explicit in this inequality equation. Although the trajectory of state variable is constrained, we have no control input to solve the equations. Dreyfus and Speyer gave gradient method forsolving optimization problem with state variable constraint[25][26].

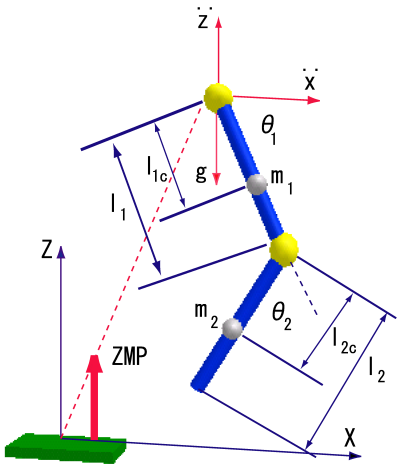


Figure 6.3: Parameters

6.5.2 Slack Variable Method

The slack variable d is introduced to make the control input explicit.

$$height + l_1 sin\theta_1(t) + l_2 sin(\theta_1 + \theta_2) - d^2(t) = 0 \tag{6.6}$$

First, the inequality constraint is converted to a equality constraint equation. We described this as S(x,t) = 0, and differentiate it with respect to time until the control input appears in the equation.

$$\frac{dS}{dt} = \frac{\partial S}{\partial t} + \frac{\partial S}{\partial x}\dot{x} = \frac{\partial S}{\partial t} + \frac{\partial S}{\partial x} f(x, u, t) \tag{6.7}$$

In this case, the control input will appear by differentiating the equation twice.

$$l_1 cos(\theta_1(t))\dot{\theta}_1(t) + l_2 cos(\theta_1(t) + \theta_2(t))(\dot{\theta}_1(t) + \dot{\theta}_2(t)) - 2d(t)\dot{d}(t) = 0 \tag{6.8}$$

$$\begin{aligned} &l_1 cos(\theta_1(t))u_{\theta_1}(t) + l_2 cos(\theta_1(t) + \theta_2(t))(u_{\theta_1(t)} + u_{\theta_2}(t)) - \\ &l_1 sin(\theta_1(t))\theta_1^2(t) - l_2 sin(\theta_1(t) + \theta_2(t))(\dot{\theta}_1(t) + \\ &\dot{\theta}_2(t))^2 - 2u_{slack}(t)d(t) - 2d^2(t) = 0 \end{aligned} \tag{6.9}$$

By introducing new state variables to the state equation,

$$\frac{d}{dt}d(t) = \dot{d}(t) \tag{6.10}$$

$$\frac{d}{dt}(\dot{d}(t)) = u_{slack}(t) \tag{6.11}$$

The equation can be written with the new input variable u_{slack}. The initial values d and $\dot{d}$ which satisfy equations (16) and (18) need to be determined. In the case of Table 3, d = 0.36, and $\dot{d} = 0.0$.

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ z(t) \\ \theta_1(t) \\ \theta_2(t) \\ \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ d(t) \\ \dot{d}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ u_x(t) \\ u_z(t) - g \\ u_{\theta 1}(t) \\ u_{\theta 2}(t) \\ \dot{d}(t) \\ u_{slack}(t) \end{bmatrix} \tag{6.12}$$

The number of state variables becomes 10 on addition of 2 new variables, and the input variables become 6 by adding one. According to Reference [3], the ZMP condition can be described as follows.

$$u_{ZMP}(t)(u_z(t) - g) = x(t)(u_z(t) - g) - z(t)u_x(t) \tag{6.13}$$

Equation(6.13) and Equation(6.9) derived by the slack variable method are 2 constraints on this condition.

6.5.3 Performance Index

Equation(6.14) is used as an performance index.

$$\begin{aligned} J &= \phi[x(t+T)] + \frac{1}{2}\int_t^{t+T} (x(\tau)-x_f)^T \cdot Q \cdot (x(\tau)-x_f) \\ &\quad +(u(\tau)-u_f)^T \cdot R \cdot (u(\tau)-u_f) d\tau \end{aligned} \tag{6.14}$$

$$\phi[x(t+T)] = (x(t+T)-x_f)^T \cdot S_f \cdot (x(t+T)-x_f)$$

$$\begin{aligned} (u(\tau)-u_f)^T \cdot R \cdot (u(\tau)-u_f) &= r_1 u_x^2(\tau) + r_2(u_z(\tau)-g)^2 + \\ & \quad r_3 u_{ZMP}^2(\tau) + r_4 u_{\theta 1}^2(\tau) + r_5 u_{\theta 2}^2(\tau) \end{aligned}$$

The first term is a terminal constraint. After the state variables are converged to the reference states, the inputs will be 0. In solution by Receding Horizon Control, the input of the Z-axis component will start to be generated again when the altitude of the center of gravity begins to drop. Major transition in the altitude causes major transition in its ZMP variable. To decrease these, we devised a method in which the terms related to u_z in the function of performance index are replaced by $u_z - g$ and a constant value is previously allocated to u_z. This method is effective for problems including the gravity term.

6.5.4 Numerical Calculation(Linear Case)

To verify the effectiveness of the slack variable method, we performed a simulation with the state equation(6.12) that is simple and contains no nonlinear terms. The link parameters used in this calculation are listed in Table 1 and the parameters used in the optimization are shown in Table6.3. The height of the center of gravity was set at 1.0 m. The simulation time was set at 1.0 s and its calculation took 0.3s. The control interval was set at 2.0 ms. A personal computer with a 333 MHz Celeron processor and running a Linux OS was employed. The result shows that it is adequate for real-time simulation. Fig.6.4.15 shows the optimum input of ZMP variable. By setting the ZMP variable at $u_{ZMP} - 0.1$, Abs(ZMP) can be set under 0.1m. This method is useful for the design of legged robots. Using this method, a designer can set the ZMP variable within a range in advance. 0.1m also means a practical value for the sole area of a biped robot or polygon described by the grounding legs of a quadruped robot. Fig.6.4.17 and Fig.6.4.18 are, respectively, the constraint equation of the ZMP condition and an undetermined multiplier ρ of the inequality state variable constraint. The first constraint is the ZMP constraint and the second is the slack variable condition. ρ_2 increases with decreased altitude of the tip of the swing leg.

Table 6.3: Parameters for Simulation(Linear Case)

S_f	diag[10.0,10.0,100.0,100.0,1.0e^{-4},1.0e^{-4},1.0e^{-4},1.0e^{-4},0.0,0.0]
Q	diag[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,0.0,0.0]
R	diag[1.0,1.0,1.0,1.0,1.0,3.5]
x_0	$[0.0, 1.0, -1.7, -0.6, 0.2, 0.0, 0.0, 0.0, 0.36, 0.0]^T$
x_f	$[0.2, 1.0, -1.2, -0.6, 0.2, 0.0, 0.0, 0.0]^T$
u_f	$[0.0, 9.8, 0.0, 0.0, 0.1, 0.0]^T$
T_f	0.2s
Δt	2ms
Number of divide	5
ζ	450
height	1.0m

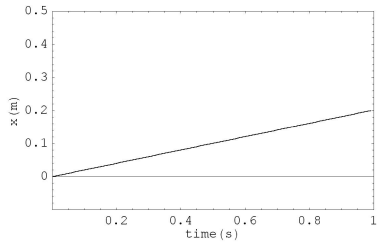


Fig6.4.1 x(Linear Case)

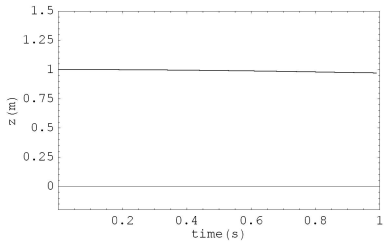


Fig6.4.2 z(Linear Case)

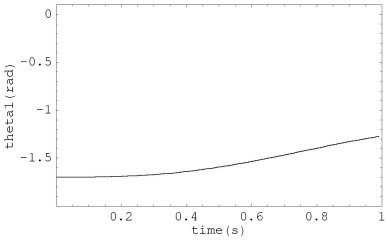


Fig6.4.3 θ_1(Linear Case)

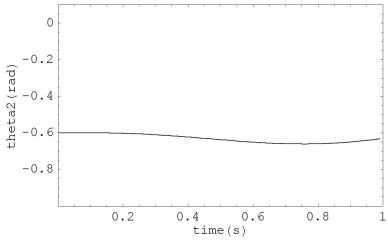


Fig6.4.4 θ_2(Linear Case)

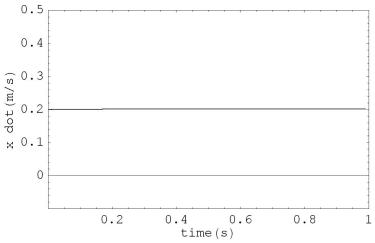


Fig6.4.5 $\dot{x}$(Linear Case)

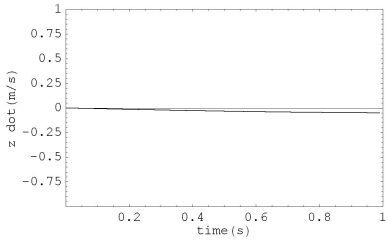


Fig6.4.6 $\dot{z}$(Linear Case)

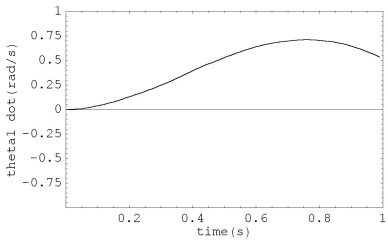


Fig6.4.7 $\dot{\theta}_1$(Linear Case)

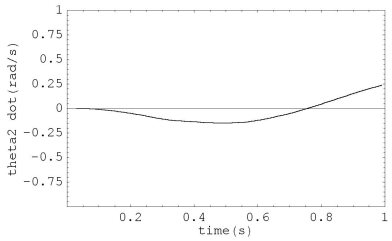


Fig6.4.8 $\dot{\theta}_2$(Linear Case)

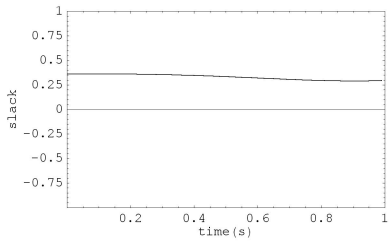


Fig6.4.9 slack d(Linear Case)

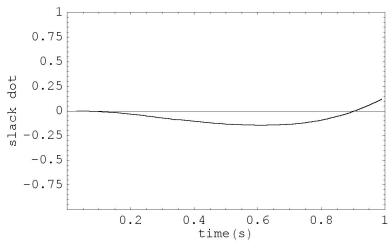


Fig6.4.10 slack $\dot{d}$(Linear Case)

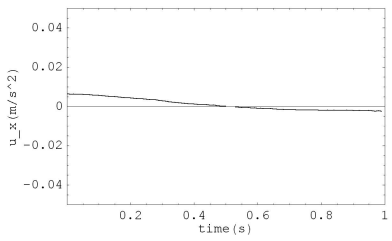


Fig6.4.11 u_x(Linear Case)

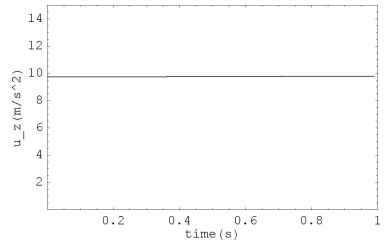


Fig6.4.12 u_z(Linear Case)

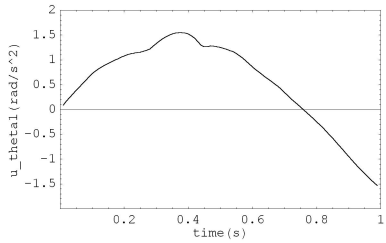


Fig6.4.13 u_{θ_1}(Linear Case)

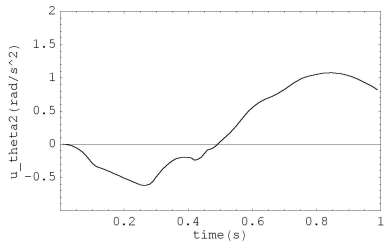


Fig6.4.14 u_{θ_2}(Linear Case)

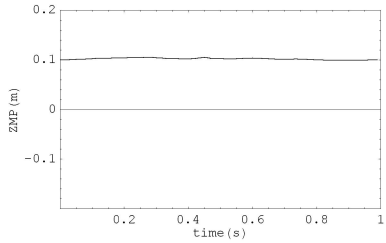


Fig6.4.15 u_{ZMP}(Linear Case)

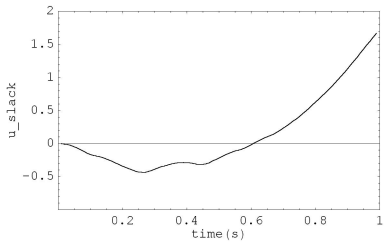


Fig6.4.16 u_{slack}(Linear Case)

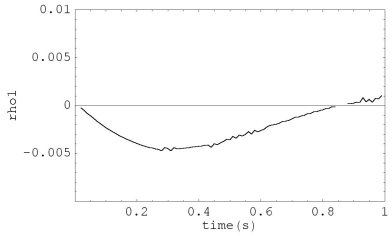


Fig6.4.17 ρ_1(Linear Case)

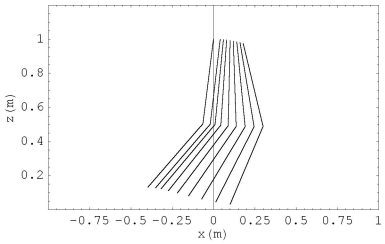


Fig6.4.19 Stick Figure(Linear Case)

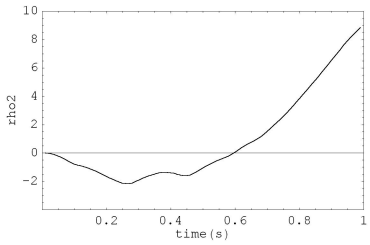


Fig6.4.18 ρ_2(Linear Case)

6.5.5 Numerical Calculation (Nonlinear Case)

We then formulated the problem, including the nonlinear term of the swing legs.

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ z(t) \\ \theta_1(t) \\ \theta_2(t) \\ \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ d(t) \\ \dot{d}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \\ \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ u_x(t) \\ u_z(t) - g \\ M^{-1}(\Theta)(u(t) - V(\Theta, \dot{\Theta}) - G(\Theta)) \\ \dot{d}(t) \\ u_{slack}(t) \end{bmatrix} \tag{6.15}$$

In this case, Equation(6.16) is substituted into u_1 and u_2 in Equation(6.9).

$$\begin{bmatrix} u_{\theta_1}(t) \\ u_{\theta_2}(t) \end{bmatrix} = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) \tag{6.16}$$

The link parameters used in the calculation are listed in Table.6.1 and the parameters used in the optimization are shown in Table.6.4. The simulation time was set at 1.0s and its calculation took 0.9s. The simulation solution converged more rapidly than the case of linear. The control interval is set at 2.0ms. How much nonlinearity can be taken into account in the formulation depends on the capacity of the computer used. More nonlinearity can be included in the formulation if a higher performance computer is used.

Table 6.4: Parameters for Simulation(Nonlinear Case)

S_f	diag[10.0,200.0,20.0,10.0,1.0e^{-4},1.0e^{-4},1.0e^{-4},1.0e^{-4},1.0,1.0]
Q	diag[1.0,4.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0]
R	diag[1.0,1.0,1.0,1.0,10.0,1.0]
x_0	$[0.0, 1.0, -1.7, -0.6, 0.27, 0.0, 0.36, 0.0]^T$
x_f	$[0.2, 1.0, -1.2, -1.0, 0.27, 0.0, 0.0, 0.0]^T$
u_f	$[0.0, 9.8, 0.0, 0.0, 0.1, 0.0]^T$
T_f	0.1s
Δt	2ms
Number of divide	5
ζ	450
height	1.0m

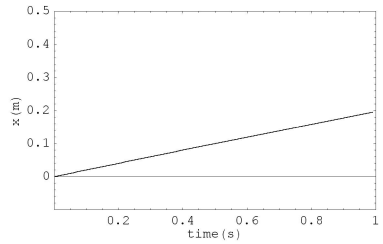


Fig6.5.1 x(Nonlinear Case)

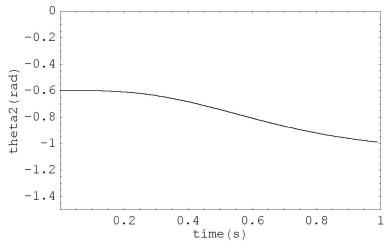


Fig6.5.4 θ_2(Nonlinear Case)

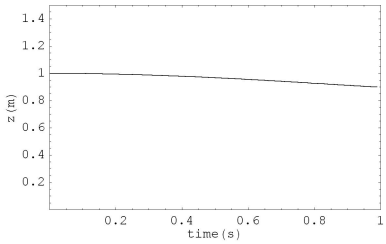


Fig6.5.2 z(Nonlinear Case)

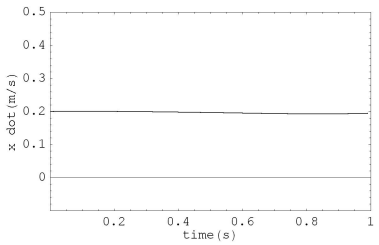


Fig6.5.5 $\dot{x}$(Nonlinear Case)

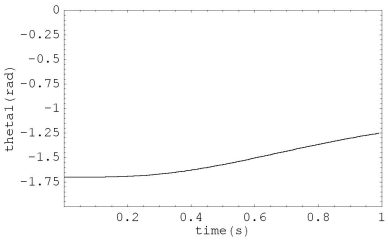


Fig6.5.3 θ_1(Nonlinear Case)

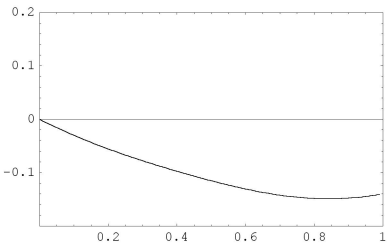


Fig6.5.6 $\dot{z}$(Nonlinear Case)

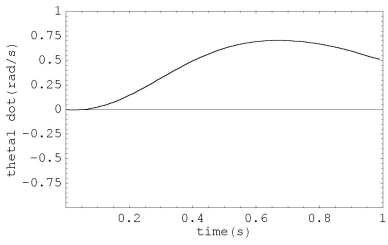


Fig6.5.7 $\dot{\theta}_1$((Nonlinear Case)

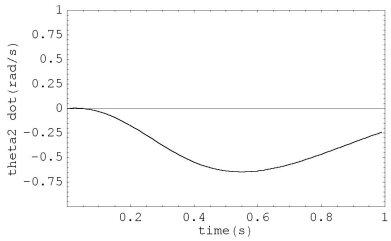


Fig6.5.8 $\dot{\theta}_2$(Nonlinear Case)

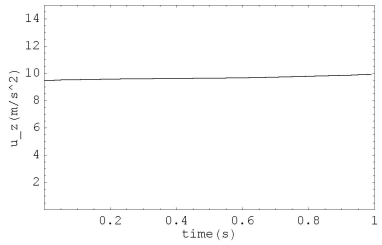


Fig6.5.12 u_z(Nonlinear Case)

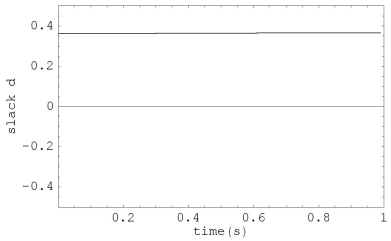


Fig6.5.9 slack d(Nonlinear Case)

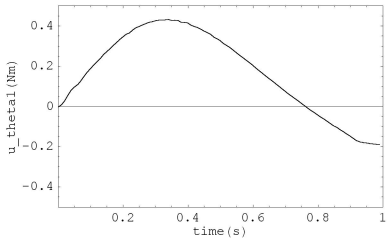


Fig6.5.13 u_{θ_1}(Nonlinear Case)

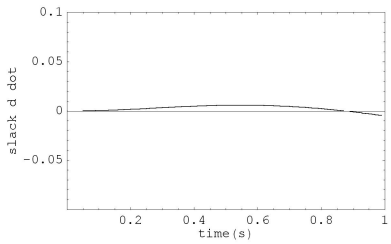


Fig6.5.10 slack $\dot{d}$(Nonlinear Case)

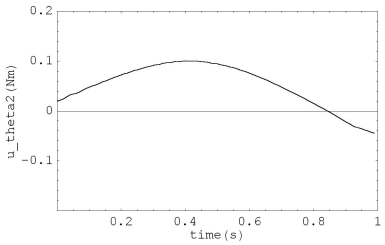


Fig6.5.14 u_{θ_2}(Nonlinear Case)

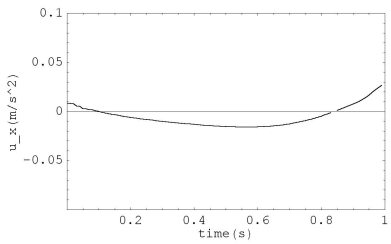


Fig6.5.11 u_x(Nonlinear Case)

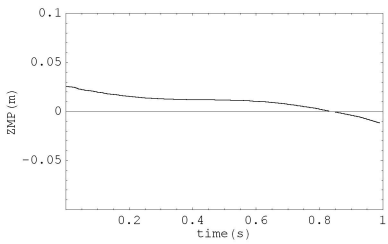


Fig6.5.15 u_{ZMP}(Nonlinear Case)

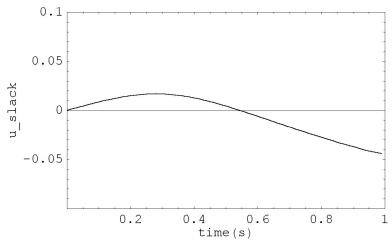


Fig6.5.16 u_{slack}(Nonlinear Case)

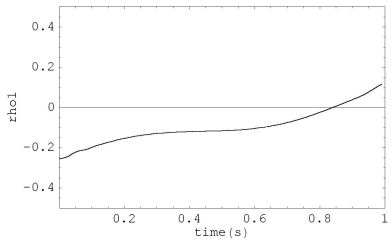


Fig6.5.17 ρ_1(Nonlinear Case)

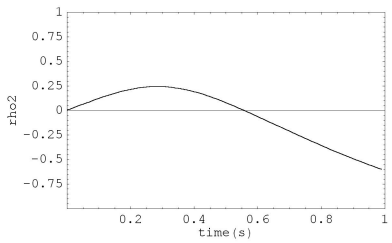


Fig6.5.18 ρ_2(Nonlinear Case)

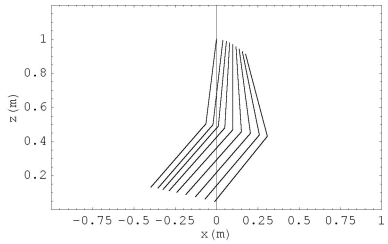


Fig6.5.19 Stick Figure(Nonlinear Case)

6.6 Conclusion

We have discussed the formulation with consideration of swing legs of a legged robot. The inequality state variable constraint (the position of a swing leg must be above the floor) must be included. In general, this inequality constraint is difficult to solve, so the slack variable method was added to the Receding Horizon Control. The method we proposed has the following characteristics:

★ Nonlinearity of swing legs can be taken into account.

★ The state equation is simple and practical.

★ The equations of motion of swing legs are easily appended to state equations. Therefore, this formulation can be extended to include multi-legged robots.

We performed a simulation of the equations including a ZMP constraint and the inequality state variable constraint in which the legs are always above the floor, and examined the results of the calculation. The fact that the calculation time was shorter than the simulation time indicates that this method can be used in real time.

Chapter 7

Nonlinear Receding Horizon Gradient Method

7.1 Introduction

Generally, Gradient Method is most popular method in optimization technique. However, its long calculation time has made hurdle to use at real-time optimization. The procedure which arbitrary trajectory for whole time converges into optimal solution using gradient makes a defect. This procedure also may cause the trajectory sink to local minimum. Feasible real-time execution of optimization requires the least formula manipulation, then it leads to feasibility that can treat with a large structure model. However past algorithms of Receding Horizon Control involves some particular formula manipulation. It makes the whole procedure taking time. The algorithm of [7][8] realized real-time optimization using backward sweep method. The procedure of the backward sweep requires additional matrix manipulation. It takes long time if the numerical model is large scaled and complicated. To reduce this problem, a new algorithm of Receding Horizon Control is proposed in this paper. This technique is assumed increasingly significant as a tool to generate robot motion real-timely. It amounts to nothing and no-intelligent controller if trajectory is not generated real-timely.

7.2 Continuation Method and Gradient

7.2.1 Continuation Method

Gradient Method has conventionally eliminated a problem that the initial condition of state equation and terminal condition of co-state equation are known although terminal condition of co-state equation and initial condition of co-state equation is unknown in TPBVP. Gradient Method is the method that the initial trajectory which is assigned as whole time($t = 0 \rightarrow t = t_f$) on ral-time axis converges to optimal solution along its gradient. Defects of this method are:

1. It could converge to minimum solution
2. It takes a good amount of time to converge.

3. This initial trajectory must be considered for immediate convergence each time.

Then a Continuation Method eliminated these problems in place of Gradient Method [7] [8]. If the interval time in Euler-Lagrange equations set 0, the solution comes to be trivial. The optimal solution can be chased to extend the time little by little based on this trivial solution. Generally, Predictor-Corrector Method is used for pursuit in Continuation Method. However it is too slow to execute on real-time control sequence.

Then a method which transverse condition is applied to balance to 0 eliminated this problem [7] [8] [28]. The process to handle matrix operation needs longer calculation time if the numerical model is larger scale. In this research, a proposed method eliminates formula operation without Euler-Lagrange equations as much as possible. Such attempt may be also said that it returns to its basic focus on.

7.2.2 Gradient

When the terminal time T on performance index interval perturbs T+dT, the trajectory also perturbs. The extended performance index is described as:

$$J^* = \phi[x^*(t,\tau)] + \int_t^{t+T} L + \lambda^{*T}(t,\tau) \cdot (f[x(t,\tau), u(t,\tau)] - \dot{x}(t,\tau))d\tau \tag{7.1}$$

The first order variation is described as:

$$\begin{aligned} \delta J^* &= [\frac{\partial \phi}{\partial \tau}]_{\tau=T} dT + [\phi_x(t,\tau) - \lambda^*(t,\tau)]_{\tau=T} dx(t,\tau) \\ &\quad + \int_t^{t+T} [(H_x) \cdot \delta x + H_u \delta u + \delta\lambda^{*T}(f - \dot{x})]d\tau \end{aligned} \tag{7.2}$$

The refreshed TPBVP is set newly. Then it satisfies transversality condition, state equations, and co-state equations. The terminal time T is fixed on the refreshed TPBVP. Then, the variation is described as:

$$\delta J^* = \int_t^{t+T} H_u \delta u d\tau \tag{7.3}$$

We can define H_u as the gradient here. If the trajectory satisfies Euler-Lagrange equations, the gradient must be 0. If the trajectory perturb from the optimal solution, it cause the generation of gradient. In Continuation Method, the gradient of the initial trajectory equals 0, then it perturbs T+dT, causes a bit gradient, and it is recovered.

7.2.3 Sampling Interval

The movement of performance index interval along real-time is executed each sampling interval. Although the time length of the performance index interval is extended, it is little and changes smoothly. This enables the time length re-scaled along the extension. In [7] [8], the number of the sampling interval arrays is fixed. In this case the sampling interval is growing longer along real-time, and it must be considered how to get re-scaled

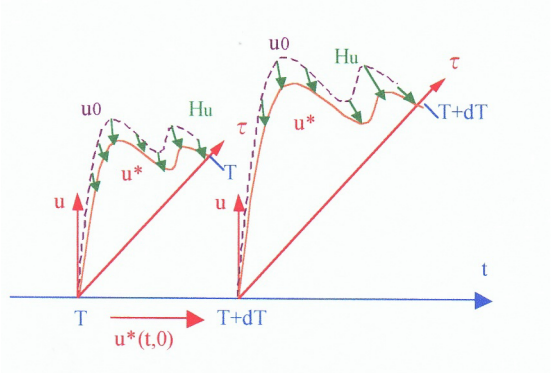


Figure 7.1: Differential changes in the terminal time

initial state value, co-state value, and input value. In this thesis, the time length of the each array is fixed. The valid number of these arrays is growing along real-time on the performance index interval. This aims for practical use. It can replace the initial array of x, λ, u on performance index interval as the next step array of x, λ, u on real-time directly. The results of "7.3.Simulation" confirm that this is feasible. Therefore the algorithm proposed here is:

1. t=0, T=0 , trivial solution
2. Increase the number of the interval arrays like as $T_{i+1} = T_i + d\tau = T_i + dT$
3. Rescale trajectory array of input variable on τ axis
4. Integrate the state equation along τ axis
5. Integrate the inhour co-state equation along τ axis
6. Calculate the gradient H_u
7. If the gradient sufficiently closed to 0, then go to (10)
8. Update the trajectory of input variable on axis
   $u_{new} = u + \alpha * H_u$, then 1-Dimension search
9. Go to (4)
10. The initial array of input variable on τ axis replaces the next step array of input variable on t axis.
11. Integrate the state equation and co-state equation using the input on (10)
12. Go to (2)

At (3), various ways of rescaling could be considered. One of those is to use the input variable trajectory one step ago and rescale it like as Fig.7.2. Anyway somewhat gradient is invoked by rescaled input variable trajectory, however, such gradient could be expected small quantity from the viewpoint of Continuation Method.

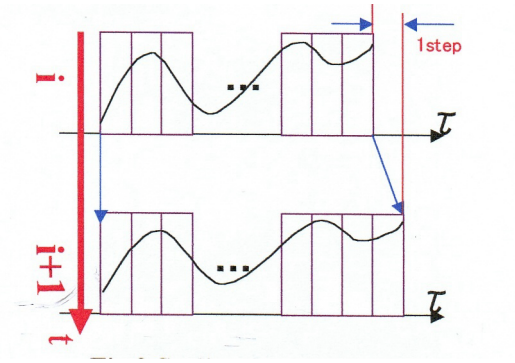


Figure 7.2: scaling for input variable

7.3 Simulation

7.3.1 Example

A simple example is arranged here[11]. The state equation is described as:

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} (1 - x_1^2(t) - x_2^2(t)) \cdot x_1(t) - x_2(t) + u(t) \\ x_1(t) \end{bmatrix} \tag{7.4}$$

The state variables are x1,x2, and input variable is u. Performance index is described as:

$$J = 2 \cdot (x_1^2(t_f) + x_2^2(t_f)) + \int_{t_0}^{t_f} (x_1^2(t) + x_2^2(t) + u^2(t) +) dt \tag{7.5}$$

This example is same example as [4]. The evaluated interval is moving and extended along real-time in Receding Horizon Control procedure, and then it is described as:

$$J = 2 \cdot (x_1^2(t,T) + x_2^2(t,T)) + \int_{t}^{t+T} (x_1^2(t,\tau) + x_2^2(t,\tau) + u^2(t,\tau)) dt \tag{7.6}$$

Fig.7.3 red dashed line shows the result by past Gradient Method (Steepest Descent Method). Blue solid line shows the result of proposed method. The trajectories of state variables and input variables are almost matched each other. Fig.7.3.6,Fig.7.3.7 error values transition described as Equation(7.7) endorse this fact. The each error is closed to 0 sufficiently.

Table 7.1: Simulation Data

-	RHGM	Gradient Method
Simulation Time	5.0 s	5.0 s
dt	10.0 ms	50.0ms
Continuation Terminal Time	$T_{i+1} = T_i + 10.0ms$	-
Maximum of Terminal Time	0.5s	-
Maximum number of iteration in a step	40	30
Initial Condition x	[0.0, 2.0]	[0.0, 2.0]
Reference xf	[0.0, 0.0]	[0.0, 0.0]

$$error(t) = \lambda^*(t,T) - \phi_x[x(t,T)] \tag{7.7}$$

Mathematica3.0 on Windows OS executes this calculation.

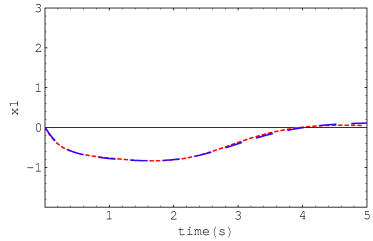


Fig7.3.1 x_1(dashed line: gradient method, solid line: RHGM)

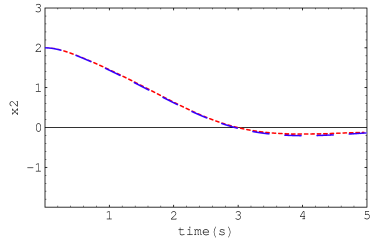


Fig7.3.2 x_2(dashed line: gradient method, solid line: RHGM)

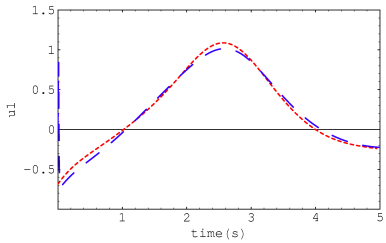


Fig7.3.3 u_1(dashed line: gradient method, solid line: RHGM)

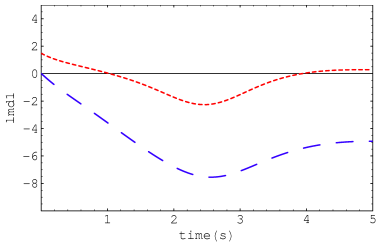


Fig7.3.4 λ_1(dashed line: gradient method, solid line: RHGM)

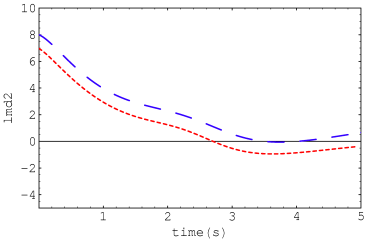


Fig7.3.5 λ_2(dashed line: gradient method, solid line: RHGM)

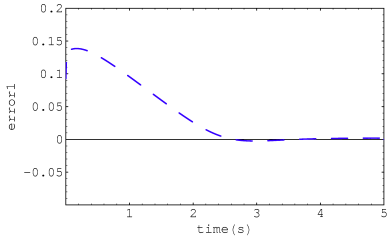


Fig7.3.6 $Error_1$(RHGM)

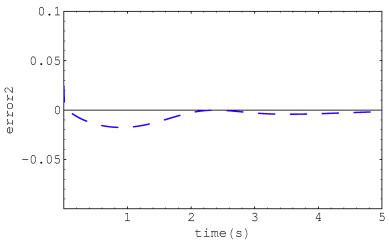


Fig7.3.7 $Error_2$(RHGM)

7.3.2 Nonlinear Two Link System

Nonlinear 2-link system can also be solved by proposed method. The state equation is described as:

$$\frac{d}{dt}\begin{bmatrix}\theta_1(t)\\ \theta_2(t)\\ \dot{\theta}_1(t)\\ \dot{\theta}_2(t)\end{bmatrix} = \begin{bmatrix}\dot{\theta}_1(t)\\ \dot{\theta}_2(t)\\ M^{-1}(\Theta)\cdot(u(t)-(\Theta,\dot{\Theta})-G(\Theta))\end{bmatrix} \Theta = \begin{bmatrix}\theta_1(t)\\ \theta_2(t)\end{bmatrix} \tag{7.8}$$

The equation involves nonlinearity of vertical two link system. M denotes inertial matrix, V denotes coriolis term, G denotes gravity term. Performance index is defined as:

$$\begin{aligned} J &= (x_f - x(t,T))^T \cdot S_f \cdot (x_f - x(t,T)) + \\ &\int_{t+T}^{t} (u^T(t,\tau)\cdot R\cdot u(t,\tau) + x^T(t,\tau)\cdot Q\cdot x(t,\tau))d\tau \end{aligned} \tag{7.9}$$

Fig.7.6 shows the result of this simulation. Table7.2 describes calculation parameters and physical parameters of the links. Fig.7.2.11-14 shows the error of the transversality condition (Equation(7.7)). These errors are under 0.02, then it provides the result that the trajectory have been able to chase the optimal solution.

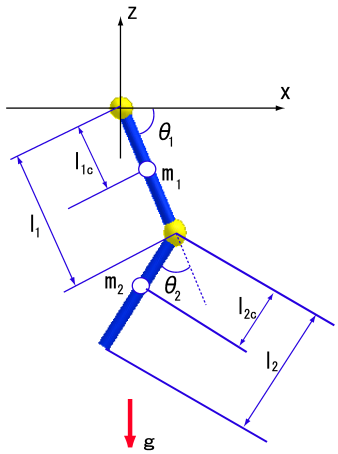


Figure 7.4: Nonlinear Two link system

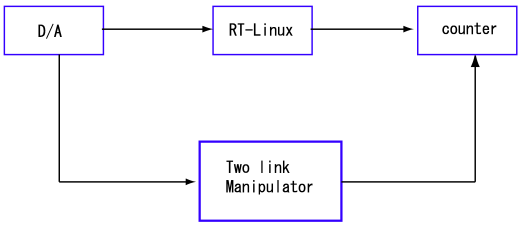


Figure 7.5: Experiment System

Table 7.2: Simulation Data

Simulation Time	4.0 s
dt	5.0 ms
Continuation Terminal Time	$T_{i+1} = T_i + dt$
Maximum of Terminal Time	0.5 s
MMaximum number of iteration in a step	40
Initial Condition	$[-1.57, 0.0, 0.0, 0.0]^T$
Reference Condition	$[-1.57, 1.57, 0.0, 0.0]^T$
S_f	diag[1.0,1.0,0.1,0.1]
R	diag[1.0,1.0]
Q	diag[1.0,1.0,1.0,1.0]
m_1	0.5kg
m_2	0.5kg
l_1	0.3m
l_{1c}	0.2m
l_2	0.3m
l_{2c}	0.2m

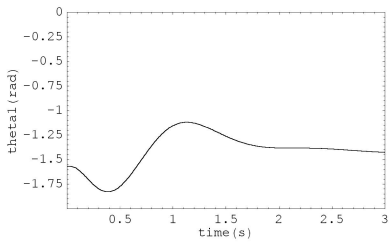


Fig7.6.1 θ_1(Nonlinear Two link System (vertical))

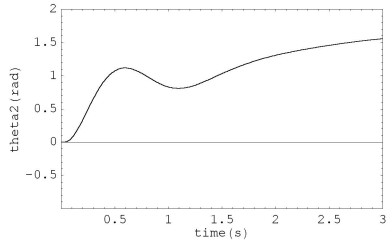


Fig7.6.2 θ_2(Nonlinear Two link System (vertical))

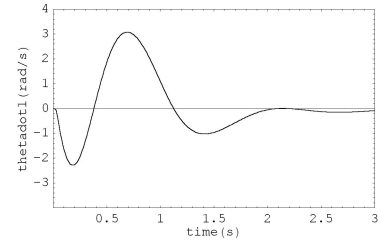


Fig7.6.3 $\dot{\theta}_1$(Nonlinear Two link System (vertical))

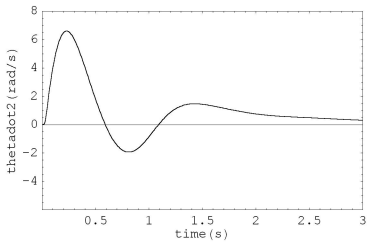


Fig7.6.4 $\dot{\theta}_2$(Nonlinear Two link System (vertical))

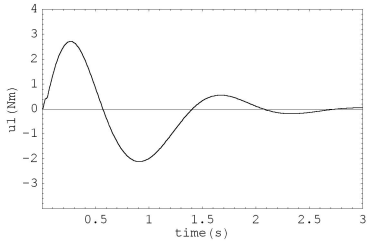


Fig7.6.5 u_{θ_1}(Nonlinear Two link System (vertical))

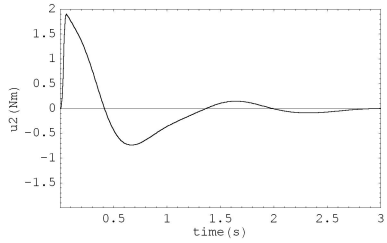


Fig7.6.6 u_{θ_1}(Nonlinear Two link System (vertical))

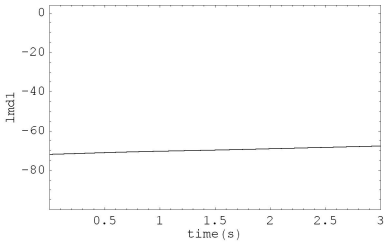


Fig7.6.7 λ_1(Nonlinear Two link System (vertical))

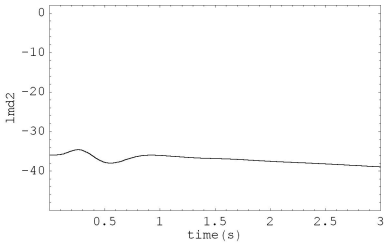


Fig7.6.8 λ_2(Nonlinear Two link System (vertical))

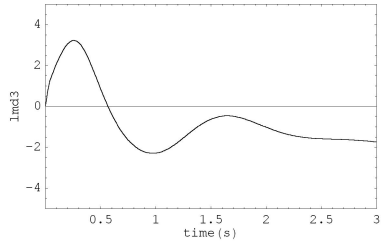


Fig7.6.9 λ_3(Nonlinear Two link System (vertical))

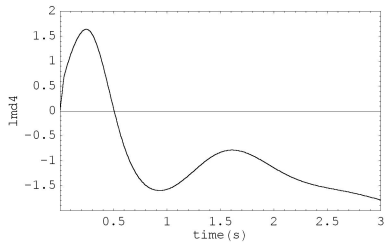


Fig7.6.10 λ_4(Nonlinear Two link System (vertical))

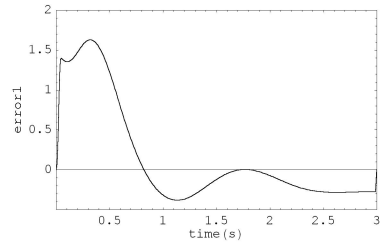


Fig7.6.11 $Error_1$(Nonlinear Two link System (vertical))

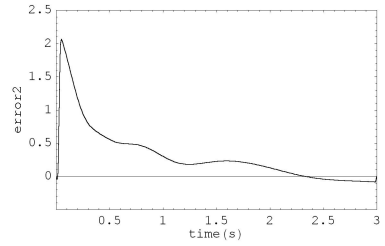


Fig7.6.12 $Error_2$(Nonlinear Two link System (vertical))

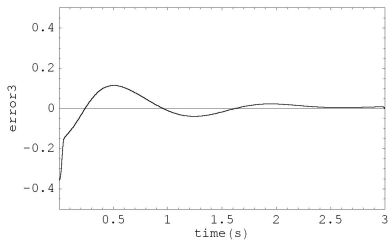


Fig7.6.13 $Error_3$(Nonlinear Two link System (vertical))

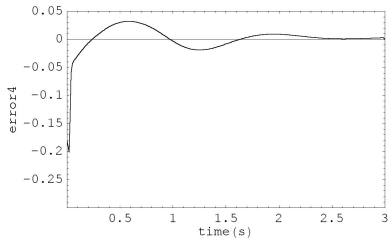


Fig7.6.14 $Error_4$(Nonlinear Two link System (vertical))

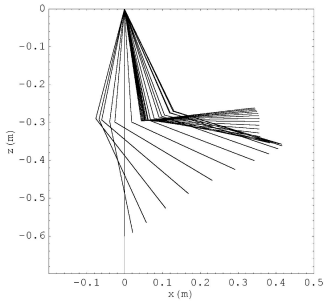


Fig7.6.15 Stick Figure(Nonlinear Two link System (vertical))

7.4 Practical Installation into Real-Time System

One of the advantages in this method is successive gradient that is slight transition. Because a real-time system has a restricted time scheduling and capacity, its search algorithm is preferably simple and less numerous iterations.

A reduced search algorithm is involved in a RT-Linux 1.1 system. Version1.1 is pretty old version. Because of considering load to real-time OS, simple searching described below is adopted.

1. Calculate gradient H_u
2. $u_{new} = u_{old} + const \cdot H_u$
3. Integrate state equation using u_{new}
4. Transversality condition
5. Integrate inhour co-state equation
6. If the index cost $<$ const, then Exit, Else if Go to (1)

The algorithm written by GNU C is also installed on RT-Linux ver.1.1. 50ms sampling interval execution has already been confirmed. The maximum limit of the iteration number of search is 10 in this case. The terminal time T is extended from 0.0s to 1.0s by each 50ms step. CPU: Celeron333MHz, Memory 32MB. The real-time task involves D/A conversion and pulse count from encoders. More performance is expected if we use cutting edge devices near future.

Table 7.3: Experiment Spec and Data

Device	Spec
Servo Motor	Maxon RE-35(90W) + Planetary Gear 260:1
Servo Amp	Titech Robot Driver
Encoder	Hewlett Packerd HED-5500 $512pulse \times 4$
Real-time OS	RT-Linux ver.1.1
Real-time interval	50ms
Initial Condition	$[-1.57, 0.0, 0.0, 0.0]^T$
Reference Condition	$[0.0, 0.0, 0.0, 0.0]^T$
S_f	diag[1.0,1.0,0.1,0.1]
R	diag[1.0,1.0]
Q	diag[1.0,1.0,1.0,1.0]
m_1	0.5kg
m_2	0.5kg
l_1	0.3m
l_{1c}	0.2m
l_2	0.3m
l_{2c}	0.2m

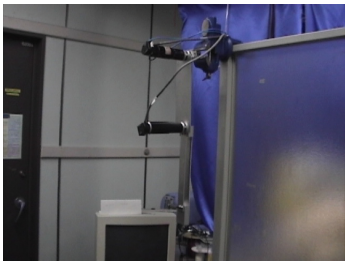


Figure 7.7: Experiment Device)

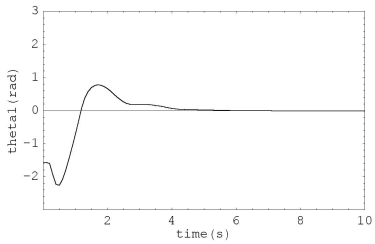


Fig7.8.1 θ_1(from counter)(Experiment)

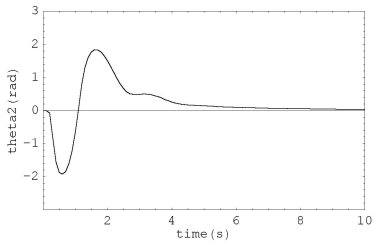


Fig7.8.2 θ_2(from counter)(Experiment)

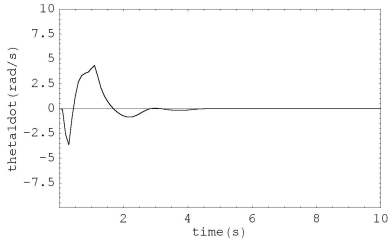


Fig7.8.3 $\dot{\theta}_1$(from counter)(Experiment)

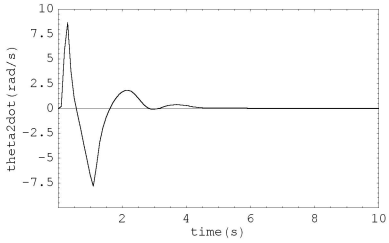


Fig7.8.4 $\dot{\theta}_2$(from counter)(Experiment)

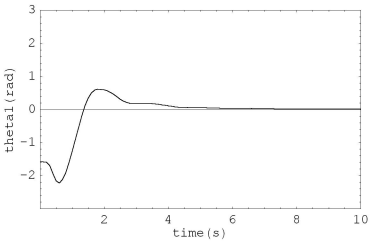


Fig7.8.5 $\theta_{1reference}$(Experiment)

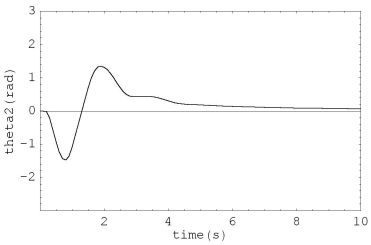


Fig7.8.6 $\theta_{2reference}$(Experiment)

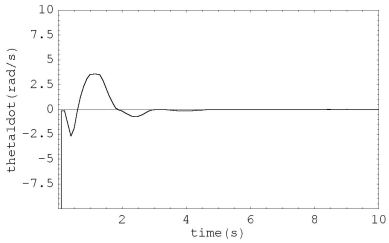


Fig7.8.7 $\dot{\theta}_{1reference}$(Experiment)

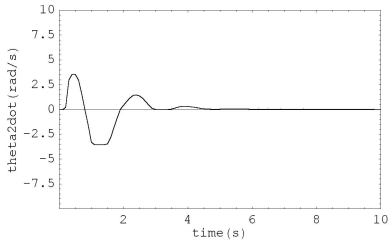


Fig7.8.8 $\dot{\theta}_{2reference}$(Experiment)

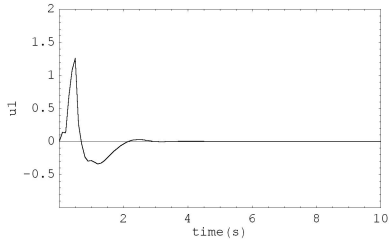


Fig7.8.9 u_1(Experiment)

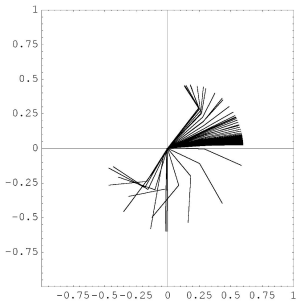


Fig7.8.11 Stick Figure(Experiment)

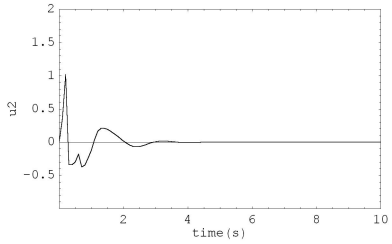


Fig7.8.10 u_2(Experiment)

7.5 Singular Point

As mentioned in 4.4, singular point problem is one of the important problems in robotics. Kinematic problem is described in 4.4, however, dynamical problem of singular point is more difficult.

$$\dot{x} = J \cdot \dot{\Theta} \tag{7.10}$$

$$\ddot{x} = \dot{J} \cdot \dot{\Theta} + J \cdot \ddot{\Theta} \tag{7.11}$$

$$\ddot{\Theta} = J^{+} \cdot (\ddot{x} - \dot{J} \cdot \dot{\Theta}) + (I - J^{+}J) \cdot k \tag{7.12}$$

$J^{+} \in R^{n \times m}$ denotes pseudo inverse of Jacobian. "I" denotes identity matrix. k denotes arbitrary constant vector. Nonlinear dynamics is described as:

$$\ddot{\Theta}(t) = M^{-1}(\Theta(t)) \cdot (u(t) - V(\Theta(t), \dot{\Theta(t)}) - G(\Theta(t))) \tag{7.13}$$

$\ddot{\Theta}$ can be substituted for Equation(7.12). Although we can treat Jacobian in nonlinear dynamics above, the measure goes to be impractical generally. Receding Horizon Control can treat this problem without Jacobian problem. The target position in Casrtesian coordination is translated to joint angle coordination at terminal condition.

$$x_f = \gamma(\Theta) \tag{7.14}$$

$$J = (x - x_f)^T \cdot S_f \cdot (x - x_f) + \int_t^{t+T} L[x^*(t,\tau), u^*((t,\tau))]d\tau \tag{7.15}$$

Simulation result of kinematic three links manipulator is shown in Fig.7.9. The terminal position at the tip of the manipulator is $[0.9, 0.0]^T$. This is one of the singular points. Simulation result of nonlinear dynamics two links manipulator is shown in Fig.7.6. One of advantages in Receding Horizon Control is that it can treat singular point problem with ease.

Table 7.4: Simulation Data for Kinematic Three Links

Simulation Time	2.5s
dt	5.0ms
Continuation Terminal Time	$T_{i+1} = T_i + dt$
Maximum of Terminal Time	0.5 s
MMaximum number of iteration in a step	40
Initial Condition	$[-1.57, 0.0, 0.0, 0.0, 0.0, 0.0]^T$
Reference Condition	$[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]^T$
S_f	diag[200,100]
R	diag[1.0,1.0,1.0]
Q	diag[1.0, 1.0, 1.0, 0.1, 0.1, 0.1]
l_1	0.3m
l_2	0.3m
l_3	0.3m
terminal condition ϕ	$[x - x_{ref}, z - z_{ref}]^T$

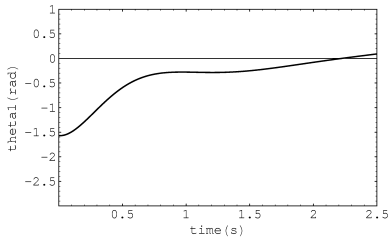


Fig7.9.1 θ_1(Kinematic Three Links)

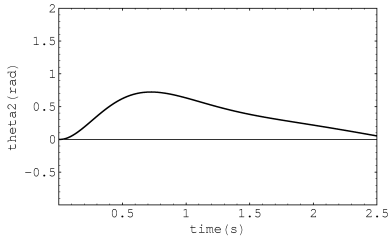


Fig7.9.2 θ_2(Kinematic Three Links)

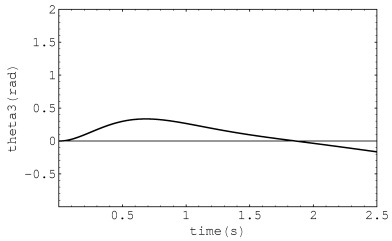


Fig7.9.3 θ_3(Kinematic Three Links)

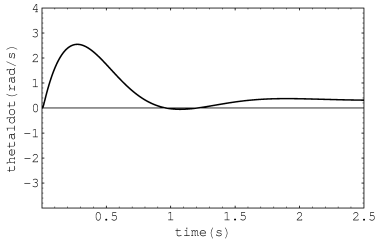


Fig7.9.4 $\dot{\theta}_1$(Kinematic Three Links)

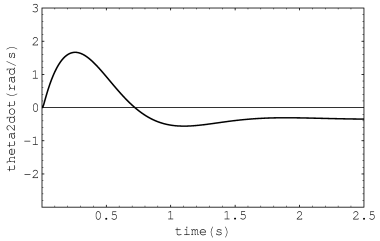


Fig7.9.5 $\dot{\theta}_2$(Kinematic Three Links)

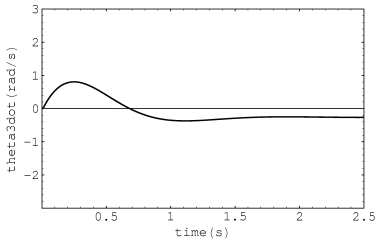


Fig7.9.6 $\dot{\theta}_3$(Kinematic Three Links)

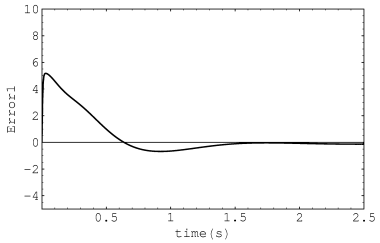


Fig7.9.7 $Error_1$(Kinematic Three Links)

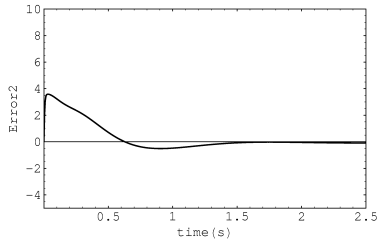


Fig7.9.8 $Error_2$(Kinematic Three Links)

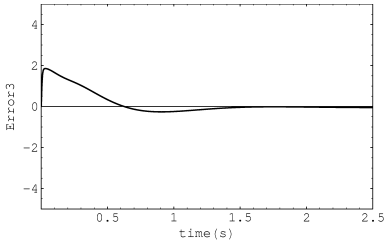


Fig7.9.9 $Error_3$(Kinematic Three Links)

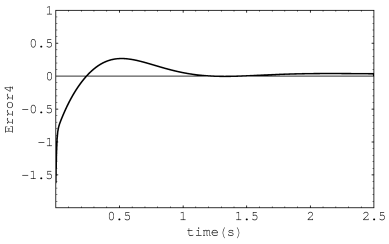


Fig7.9.10 $Error_4$(Kinematic Three Links)

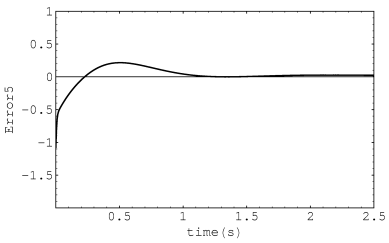


Fig7.9.11 $Error_5$(Kinematic Three Links)

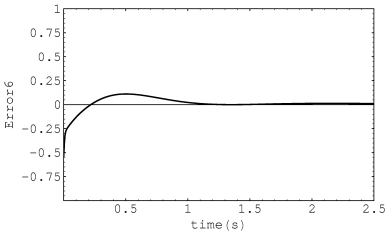


Fig7.9.12 $Error_6$(Kinematic Three Links)

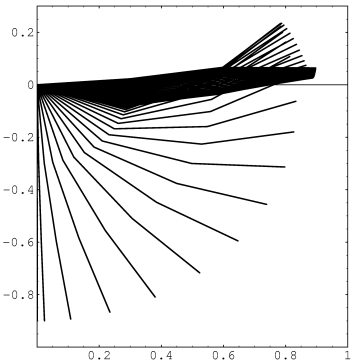


Fig7.9.13 Stick Figure(Kinematic Three Links)

7.6 Conclusion

The method proposed in this thesis consists of a minimum configuration of Euler-Lagrange equation, making the algorithm comparatively simple. The advantage of this simplicity is apparent when the method is applied to cases marked by complicated state equations or large-scaled models. A disadvantage is that the gradient method requires long calculation periods-a problem that can be eliminated using the continuation method. If the gradient method is installed at the moving horizon interval, the algorithm load can be reduced. In such cases, the algorithm is only required to eliminate a slight gradient invoked by a perturbation from the optimal trajectory in the preceding step. It is important to note that this is only one of many Receding Horizon Control algorithms, therefore its use should be considered situation-dependent.

Chapter 8

Conclusion and Recommendation

8.1 Overall Perspective

The thesis discusses application of Receding Horizon Control on real-time system and proposes an algorithm of Receding Horizon Control.

This study settled on the subject matter of legged robot as an application. Any type of legged robot has instability associated with Zero Moment Point balance. This kind of control has been technical difficulty for formulation. Formulation as an optimization problem with equal constraint and Receding Horizon Control make this problem enable its formulation. The variable of Zero Moment Point is described as an input variable, and then it can be obtained as one of the optimized solutions. We can confirm the effectiveness of Receding Horizon Control through this formulation.

Constraint has an important role in formulation of Receding Horizon Control, especially at real-time control scheme for mechanical object. Conditions like as Zero Moment Point could be included in formulation. However inequality constraint is used to describe a condition such as lower bound or upper bound requires somewhat ingenuity to involve in formulation. We can look at the Receding Horizon Control formulation with swing leg constraint through the viewpoint. This problem also involves state variable constraint without input variable. Then slack variable method is also introduced to solve the problem.

These two applications use a Receding Horizon Control algorithm that Ohtsuka developed. Through reconstructing this algorithm many times, some demands were emerged. Such demands made Receding Horizon Gradient Method. Essence of homotopy theory and continuation method is well utilized in Receding Horizon Gradient Method. The proposed method involves mere components where are in Euler-Lagrange equation, then the algorithm goes to be quite simple. This brings out advantage when it is applied into a case which state equation is complicate or large scaled model. However, this algorithm is mere a one of Receding Horizon Control algorithms, its use may depend on the situation. The best algorithm is described in the bible of "rule of nature" and quest for better algorithm should be continued.

For the reason above mentioned, real-time optimal robot motion is benefit of Receding Horizon Control. Real-time optimization technique discussed here can be applied to various industrial problem such as aerospace, railway, automobile, and so on.

In this thesis, I will use the example of legged robots to discuss the application of

Receding Horizon Control on real-time systems and to propose a Receding Horizon Control algorithm. All types of legged robots have instability issues associated with Zero Moment Point balance that present significant technical difficulties for formulation. Here I view formulation as an optimization problem with equal constraint, which allows for the use of Receding Horizon Control to address the formulation problem. The Zero Moment Point variable, described as an input variable, can be obtained as one optimized solution, and the effectiveness of Receding Horizon Control can be confirmed using this formulation.

Constraint plays an important role in Receding Horizon Control formulation, especially in real-time control schemes for mechanical objects. While Zero Moment Point and similar conditions can be included in such formulations, an inequality constraint is used to describe conditions (e.g., lower or upper bounds) that require a considerable amount of ingenuity to make such a combination successful. This is one way of viewing an Receding Horizon Control formulation with a swing leg constraint. This problem also involves a state variable constraint without an input variable, which can be solved by introducing a slack variable method.

These two applications use the Receding Horizon Control algorithm developed by Ohtsuka. Repeated algorithm reconstruction resulted in a number of emerging demands. Such demands required the use of a Receding Horizon Gradient method that entailed features of homotopy theory and continuation methodology. The method proposed in this thesis consists of a minimum configuration of Euler-Lagrange equation, making the algorithm comparatively simple. This is advantageous when applied to complicated state equations or large-scaled models. However, the algorithm is only one of many Receding Horizon Control algorithms, meaning that its use is situation-dependent. The search for a better algorithm should be continued.

8.2 Recommendations for Future Research

Industrial applications of Receding Horizon Control have not yet attracted a great deal of research interest. However, various industrial applications should be tested in the same manner that this thesis looked at legged robots as control objects. Several different types of Receding Horizon Control algorithms have recently been proposed, each with its own specific advantages and disadvantages. Individual uses in individual cases need to be considered in order to determine optimal algorithms for specific applications.

The new method proposed in Chapter7 can be applied into large scaled numerical model like as LSS(Large Space Structure) or hyper-redundant manipulator.

Receding Horizon Control provides benefit to nonholonomic dynamics and constraint. The optimal solution can be obtained from nonholonomic dynamics which ordinal optimal regulator could not control.

Conceivable extension of nonlinear optimal feedback control is differential game. It is well known that robust control problem can be formulated as differential game.

The best algorithm is described in the bible of "rule of nature" and quest for better algorithm should be continued.

Appendix A

Transition Matrix

Linear differential equation:

$$\dot{x}(t) = A(t) \cdot x(t) \tag{A.1}$$

A(t) denotes $n \times n$ square matrix. We can make a equation below using the A(t).

$$\dot{X}(t) = A(t) \cdot X(t) \tag{A.2}$$

The solution which $X(t_0)$ is regular is called as fundamental matrix. The transition matrix of Equation(A.1) is described as:

$$\Phi(t, \tau) = X(t) \cdot X^{-1}(\tau) \tag{A.3}$$

$X(t), X(\tau)$ are fundamental matrixes which equal to a regular matrix X_0 at $t = t_0$. Transition matrix $\Phi(t, \tau)$ equals to the solution of Equation(A.2) at $t = \tau$ with $X(\tau) = I$.
We can get solutionn of a differential equation below using transition matrix.

$$\dot{x}(t) = A(t) \cdot x(t) + f(t) \tag{A.4}$$

$$x(t) = \Phi(t, t_0) \cdot x(t_0) + \int_{t_0}^{t} \Phi(t, \tau) \cdot f(\tau) d\tau \tag{A.5}$$

If matrix A(t) is constant, then t_0 can be take as 0, and we have

$$\Phi(t, t_0) \triangleq \Phi(t, 0) \triangleq \Phi(t) \triangleq e^{At} \triangleq \sum_{k=0}^{\infty} \frac{A^k}{k!} \cdot t^k \tag{A.6}$$

$$x(t) = e^{A(t-t_0)} \cdot x(t_0) \tag{A.7}$$

Bibliography

[1] L.S. Pontryagin and V.G.Boltyanskii,The Mathematical Theory of Optimal Processes,1961

[2] A.K.Wu and A.Miele Sequential Conjugate Gradient Restoration Algorithm for Optimal Control Problems with Nondifferential Constraints and General Boundary Conditions, Part1, Optimal Control Applications and Methods, vol.1-1,1980

[3] S.Gonzalez and S.Rodriguez, Modified Quasi-liniearization Algorithm for Optimal Control Problems with Nondifferential Constraints and General Boundary Conditions, Journal off Optimization Theory and Applications, vol.50-1,1986

[4] Carl.B.Boyer,The History of the Calculus and its Conceptual Development,Dover Publications Inc.,1949

[5] W.M.Priestley,Calculus:An Histrical ApproachSpringer-Verlag,1979

[6] Robert H. Goddard,A Method of Reaching Extreme Altitudes,Smithsonian Miscellaneous Publication No. 2540,1920

[7] Ohtsuka,T.,Fujii,H., Stabilized Continuation Method for solving Optimal Control Problems, Journal of Guidance, Control, and Dynamics. 17-5, p.p.950-957,1994

[8] Ohtsuka,T.,Fujii,H., Real-Time Optimization Algorithm for Nonlinear Receding Horizon Control, Automatica, 33-6,p.p.1147-1154, 1997

[9] Ohtsuka,T.,Control of Distributed Parameter Systems and Nonlinear Systems in AeroSpace Engineering, Doctor thesis, Tokyo Metropolitan Institute of Technology, Department of AeroSpace Engineering,1994

[10] John J.Craig, Introduction to Robotics -Mechanics and Control-, Addison-Wesley,1986

[11] H.Kano, Theory and Optimization of System , corona, p.217, 1987, (Japanese Language)

[12] Kabamba,P.T.,Longman,R.W., and Jian-Guo,.,A Homotopy Approach to the Feedback Stabilization of Linear Systems, Journal of Guidance, Control, and Dynamics, vol.10, No.5, p.p.422-432,1987

[13] Abraham Silbershatz, Peter B.Galvin, GregGagne, Operating System Concepts,John Wiley & Sons Inc.,2003

[14] Y.A.Thomas,Linear Quadratic Optimal Estimation and Control with Receding Horizon,Electric Letters,11-1, pp.19-21,1975

[15] C.C.Chen,L.Shaw, On Receding Horizon Feedback Control, Automatica, 18-3,pp.349-352,1982

[16] Vukobratovic,M.,Juricic,D.,Contributions to the synthesis of biped gait. IEEE Trans on Biomedical Engineering,BME-16:1-6,1969

[17] Vukobratovic,M.,Frank,M.A.A.,Juricic,D.,On the Stability of Biped Locomotion,IEEE Trans on Biomedical Engineering,BME-17,No.1,pp.25-36,1970

[18] Hirai,K.,Hirose,M.,Haikawa,Y.,Takenaka,T.,The Development of Honda humanoid Robot,Proc.IEEE Int.Conf.Robotics and Automation,pp.1321-1326,1998

[19] D.Q.Mayne,H.Michalska, Receding Horizon Control of Nonlinear Systems,IEEE Trans, AC-35-7, pp.814-824, 1990

[20] Mayne,D.Q.,Michalska,H.,Robust Receding Horizon Control of Constrained Nonlinear Systems,IEEE Trans. on AC,Vol.38, No.11, pp.1623-1633, 1993

[21] Eaton,J.W.and Rawlings,J.B.,Feedback Control of Chemical Processes Using On-Line Optimization Techniques, Computers and Chemical Engineering,Vol.14, No.4-5,pp.469-479,1990

[22] Kazuhisa Mitobe・Atsushi Masuyama・Tatsuo Shibata・Mitsuhiro Yamano・Yasuo Nasu,A ZMP Manipulation Method for Walking Robots and its Application to Angular Momentum Control,Journal of Robot Society of Japan Vol.20, No.5, pp.53-58,2002

[23] Bryson Jr,A.E., Ho, Y.C., Applied Optimal Control, Hemisphere, 1975

[24] Bryson Jr,A.E.,Ho, Dynamic Optimization, Addison-Wesley, 1999

[25] S.E.Dreyfus,Dynamic Programming and the Calculus of Variations, Academic Press,1966

[26] J.L.Speyer andA.E.Bryson Optimal Programming Problems with a Bounded State Space,AIAA journal, vol.6, p.p.1488-1492,1968

[27] J.Prims, V.Nevisic, J.Doyle, On Receding Horizon Extensions and Control Lyapunov Functions, ACC Procedings 1998

[28] Takeuchi ,H., Real Time Optimization for Robot Control using Receding Horizon Control with Equal Constraint, Journal of Robotic Systems, Vol. 20, No. 1, 2003

Biographical Information

Publications

Papers published in Journals

1. H.Takeuchi and T.Ohtsuka, Receding Horizon Control Applied to Optimization for Mechanical Link Systems -Analysis using the Continuation Method-, Journal of Robotics Society of Japan, Vol.17, No.3, pp.92-97,1999
(竹内、大塚、Receding Horizon Control を用いた機械的リンク系の最適化計算　ー連続変形法を用いた数値解に関する検討ー、日本ロボット学会論文、vol.17, No.3, p.p.402-407,1999)

2. H.Takeuchi, Numerical analysis for attitudes of multi-legged robot - concept of leg functions distribution -, Japan Society of Computational Engineering and Science, transaction of JSCES, vol.2 ,p.p.1-6, 2000
(竹内、多脚式ロボットの脚姿勢の定量的解析 - 脚機能分担化の概念-、日本計算工学会論文、vol.2、論文番号 20000005, p.p.47-52, 2000)

3. H.Takeuchi, Real Time Optimal Control for Legged Robot - Autonomous Generation of ZMP reference using Receding Horizon Control with Equal Constraint - , Japan Society of Computational Engineering and Science, transaction of JSCES, vol.3 ,p.p.1-6, 2001
(竹内、脚式ロボットのリアルタイム最適制御 ー 等式拘束条件付 Receding Horizon Control による目標 ZMP 軌道の生成、日本計算工学会論文、vol.3、論文番号 20010001 , p.p.1-6, 2001)

4. H.Takeuchi, Real Time Optimization and Control of Legged Robot - Formulization with Inequality State Constraint for Swing Leg - , Japan Society of Computational Engineering and Science, transaction of JSCES, vol.4 ,p.p.1-6, 2002
(竹内、脚式ロボットのリアルタイム最適化と制御 -不等式状態量拘束条件を伴う遊脚の定式化　-、日本計算工学会論文、vol.4、論文番号 20020003, p.p.131-137, 2002)

5. H.Takeuchi, Nonlinear Receding Horizon Gradient Method, Japan Society of Computational Engineering and Science, transaction of JSCES, vol.5, p.p.181-187, 2004

(竹内、非線形 Receding Horizon 勾配法、日本計算工学会論文、vol.5、論文番号 20020003, 2004)

6. Hiroki Takeuchi, Real Time Optimization for Robot Control using Receding Horizon Control with Equal Constraint, Journal of Robotic Systems, Vol. 20, No. 1, p.p.3-13, 2003

7. Hiroki Takeuchi, Real Time Optimization and Control of Legged Robot - Formulization with Inequality State Constraint for - , Journal of Robotic Systems, Vol. 21, No. 4, p.p.153-166, 2004

International Conference Proceedings

1. Hiroki Takeuchi, Development of MEL HORSE, 3rd France-Japan Congress and 1st Europe-Asia Congress on Mechatronics Proceedings, p.p.348-353, 1996

2. Hiroki Takeuchi, Development of MEL HORSE, 2nd ECPD International Conference on Advanced Robotics, Intelligent Automation and Active Systems Proceedings, p.p.289-293, 1996

3. Hiroki TAKEUCHI, Development of MEL HORSE, 3rdEuropean Conference Peace and Development Conference on Advanced Robotics, Intelligent Automation and Active Systems Proceedings,1997

4. Hiroki Takeuchi, Development of Leg-Functions coordinated Robot "MEL HORSE", IEEE International Conference on Advanced Robotics, Proceedings, p.p.59-64, 1997

5. Hiroki Takeuchi, Development of MEL HORSE, 1st International Symposium on Mobile, Climbing and Walking Robots, p.p.21-26, 1998

6. Hiroki Takeuchi, Development of MEL HORSE, IEEE International Conference on Robotics and Automation, p.p.1057-1062, 1999

7. Hiroki Takeuchi, Development of MEL HORSE, IEEE International Conference on Intelligent Robots and Systems 2000 Proceedings, p.p.2018-2024,2000

8. Hiroki Takeuchi, Development of MEL HORSE, IEEE International Conference on Robotics and Automation 2001 Proceedings, p.p.3165-3171, 2001

9. Hiroki Takeuchi, Slack Variable Method for State Variable Constraint, IEEE International Conference on Robotics and Automation 2003 Proceedings, p.p.2350-2355, 2003

10. Hiroki Takeuchi, Nonlinear Receding Horizon Gradient Method, IEEE CCA2004 Proceedings, p.p.1615-1620, 2004

Domestic Conference Proceedings

1. 竹内，歩行ロボットの最適歩行問題，日本機械学会ロボティクス・メカトロニクス講演会'98，1998/06/01
(H.Takeuchi,Optimal Walk Problem for Legged Robot,Japan Society of Mechanical Engineers Robotics and Mechatronics Conference, 1998)

2. 竹内，MEL HORSE の開発，機械技術研究所研究発表会，1998/06/01
(H.Takeuchi, Development of MEL HORSE, Mechanical Engineering Laboratory meeting for announcing the results,1998)

3. 竹内，MELHORSE の開発，第 16 回日本ロボット学会学術講演会，1998/09/01
(H.Takeuchi,Development of MEL HORSE, Robotics Society of Japan Annual Conference,1998)

4. 竹内，2 足歩行ロボットの最適歩行問題，竹内裕喜，第 16 回日本ロボット学会学術講演会，1998/09/01
(H.Takeuchi,Optimal Walk Problem for Biped Robot, Robotics Society of Japan Annual Conference,1998)

5. 竹内，磯部，上田，FEM によるマニピュレータの並列制御，第 4 回計算工学会講演会，1999/05/01
(D.Isobe,and T.Ueda,and H.Takeuchi, Parallel Control Method for Manipulators by Using FEM, Japan Society of Computational Engineering and Science Annual Conference,1999)

6. 竹内，歩行ロボットの最適歩行問題，日本機械学会ロボティクスメカトロニクス講演会，1999/06/01
(H.Takeuchi,Optimal Walk Problem for Legged Robot,Japan Society of Mechanical Engineers Robotics and Mechatronics Conference, 1999)

7. 竹内，2 足歩行ロボットの最適歩行問題，第 17 回日本ロボット学会学術講演会，1999/09/01
(H.Takeuchi,Optimal Walk Problem for Biped Robot, Robotics Society of Japan Annual Conference,1999)

8. 竹内，MEL HORSE の開発，第 17 回日本ロボット学会学術講演会,1999/09/01
(H.Takeuchi,Development of MEL HORSE, Robotics Society of Japan Annual Conference,1999)

9. 竹内，脚式ロボットの最適運動問題，日本機械学会ロボティクスメカトロニクス講演会，2000/05/01
(H.Takeuchi,Optimal Motion Problem of Legged Robot,Japan Society of Mechanical Engineers Robotics and Mechatronics Conference, 2000)

10. 磯部、上田，竹内, 超冗長マニピュレータの関節トルク算出法，日本計算工学会講演会，2000/05/01
(D.Isobe,and T.Ueda,and H.Takeuchi, A Numerical Scheme for Calculating Joint Torque of Hyper-Redundant Manipulators, Japan Society of Computational Engineering and Science Annual Conference,2000)

11. 竹内，多脚式ロボットの脚姿勢の定量的解析 -脚機能分担化の概念-，竹内裕喜，日本計算工学会講演会，2000/05/01
(H.Takeuchi, Numerical analysis for attitudes of multi-legged robot - concept of leg functions distribution -, Japan Society of Computational Engineering and Science Annual Conference,2000)

12. 竹内，MEL HORSE の開発，第 18 回日本ロボット学会学術講演会,2000/09/01
(H.Takeuchi,Development of MEL HORSE, Robotics Society of Japan Annual Conference,2000)

13. 竹内，2 足歩行ロボットの最適歩行問題，第 18 回日本ロボット学会学術講演会，2000/09/01
(H.Takeuchi,Optimal Walk Problem for Biped Robot, Robotics Society of Japan Annual Conference,2000)

14. 竹内，脚式ロボットのリアルタイム最適制御　，日本計算工学会第 6 回講演会，東京　法政大学,2001/06/01
(H.Takeuchi,Real-Time Optimization for Legged Robot,Japan Society for Computational Engineering and Science Annual Conference,2001)

15. 竹内，脚式ロボットの最適化問題，日本機械学会ロボティクスメカトロニクス講演会，高松市,2001/06/10

(H.Takeuchi,Optimization for Legged Robot,Japan Society of Mechanical Engineers Robotics and Mechatronics Conference, 2001)

16. 竹内，MEL HORSE の開発，日本ロボット学会学術講演会，東京、2001/09/15
(H.Takeuchi,Development of MEL Horse, Robotics Society of Japan Annual Conference,2001)

17. 2 足歩行ロボットの最適歩行問題，竹内 裕喜，日本ロボット学会学術講演会，東京、2001/09/15
(H.Takeuchi,Optimal Walk Problem for Biped Robot, Robotics Society of Japan Annual Conference,2001)

18. 竹内，脚式ロボットの最適化問題，計測自動制御学会，東京工業大学長津田キャンパス、2002/03/28
(H.Takeuchi,Optimization for Legged Robot,Society of Instrument and Control Engineers Committee Conference(at Tokyo Institute of Technology),2002)

19. 竹内，脚式ロボットのリアルタイム最適化と制御 -不等式状態量拘束条件を伴う遊脚の定式化-，日本計算工学会，東京都千代田区、2002/05/20
(H.Takeuchi,Optimization and Control for Legged Robot -Formulation for Swing Leg Condition with Inequality State Constraint-,Japan Society for Computational Engineering and Science Annual Conference,2002)

20. ，竹内，脚式ロボットの最適化と制御, 日本機械学会ロボティクスメカトロ二クス講演会，島根県松江市,2002/06/09
(H.Takeuchi,Optimization and Control for Legged Robot,Japan Society of Mechanical Engineers Robotics and Mechatronics Conference, 2002)

21. 竹内，4 脚式ロボットの最適化と制御，日本ロボット学会学術講演会，大阪大学、2002/10/14
(H.Takeuchi,Optimization and Control for Quadraped Robot, Robotics Society of Japan Annual Conference,2002)

22. 竹内，2 足歩行ロボットのリアルタイム最適化と制御，日本ロボット学会学術講演会，大阪大学、2002/10/14
(H.Takeuchi,Real-Time Optimization and Control for Biped Robot, Robotics Society of Japan Annual Conference,2002)

23. 竹内，脚式ロボットの運動最適化問題，日本計算工学会講演会，tokyo、2003/05/21
(H.Takeuchi,Optimization for Legged Robot,Japan Society for Computational Engineering and Science Annual Conference,2003)

24. 竹内，脚式ロボットの最適化問題，日本機械学会ロボティクスメカトロニクス講演会，北海道　函館市,2003/05/25
(H.Takeuchi,Optimization for Legged Robot,Japan Society of Mechanical Engineers Robotics and Mechatronics Conference, 2003)

25. 竹内, 脚式ロボットの運動最適化問題, 計測自動制御学会部門大会, 神戸市,2003/05/28
(H.Takeuchi,Optimization for Legged Robot,Society of Instrument and Control Engineers Conference in Kobe,2003)

26. 竹内，非線形 Receding Horizon 勾配法, 日本計算工学会講演会，tokyo,2004/05
(H.Takeuchi,Nonlinear Receding Horizon Gradient Method, Japan Society for Computational Engineering and Science Annual Conference, 2004)