Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

DIRECT APPLICATION OF THE LEAST ACTION PRINCIPLE TO SOLVE (HUMAN) MOVEMENT DYNAMICS WITHOUT USING DIFFERENTIAL EQUATIONS OF MOTION

by
Amitha Kumar

This thesis explores the numerical feasibility of solving motion 2-point *boundary value problems* (BVPs) by direct numerical minimization of the action without using the *equations of motion* (EOM) as an intermediate step. The proposed *direct least action* (DLA) approach using the *downhill simplex method* (DSM) is applied to both the single and double pendulum systems as beginning test problems. The solutions so obtained are compared to numerical solutions of the corresponding Lagrange EOM solved using a first order Euler algorithm for the same mechanical problems. The output path obtained by both of the methods essentially superimpose on each other.

Future steps will be to apply DLA to more complicated multi-branched pendulum systems that are used to model the human body and to apply more efficient numerical methods than the DSM algorithm to DLA. Eventually if numerical algorithms can be developed that make the DLA approach as efficient or more efficient than using Lagrange's differential EOM to solve 2-point BVPs for multi-branched pendulum systems then such algorithms will be embedded in the software that we have developed and use in the human motion analysis and performance laboratory at NJIT to solve for human motion problems. The software employs a new method called the *Boundary Method*® a new mathematical technique developed in our laboratory. This method solves simultaneously for *both* new motions that can accomplish a given motor task *and* the net muscular joint forces required to produce those new motions.

DIRECT APPLICATION OF THE LEAST ACTION PRINCIPLE TO SOLVE (HUMAN) MOVEMENT DYNAMICS WITHOUT USING DIFFERENTIAL EQUATIONS OF MOTION

by
Amitha Kumar

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Biomedical Engineering

Department of Biomedical Engineering

January 2006

APPROVAL PAGE

DIRECT APPLICATION OF THE LEAST ACTION PRINCIPLE TO SOLVE (HUMAN) MOVEMENT DYNAMICS WITHOUT USING DIFFERENTIAL EQUATIONS OF MOTION

Amitha Kumar

Dr. H. M. Lacker, Thesis Advisor — Date
Professor of Biomedical Engineering, NJIT

Dr. Richard A. Foulds, Committee Member — Date
Associate Professor of Biomedical Engineering, NJIT

Dr. Sergei Adamovich, Committee Member — Date
Assistant Professor of Biomedical Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Amitha Kumar

Degree: Master of Science

Date: January 2006

Undergraduate and Graduate Education:

- Master of Biomedical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2006

- Bachelor of Science in Mechanical Engineering,
  MVIT, Bangalore, India, 2000

This thesis is dedicated to
my parents, Dr. Anand Kumar and Dr. Vijaya,
my husband Sachin Dev,
and
my daughter, Tanya

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. H. M. Lacker, who not only served as my research supervisor, providing valuable and countless resources, insight, and intuition, but also constantly gave me support, encouragement, and reassurance. Special thanks are given to Dr Richard A. Foulds and Dr. Sergei Adamovich for actively participating in my committee.

I would also like to thank my parents, Dr. Anand Kumar and Dr. Vijaya and my husband, Sachin Dev for providing constant support and encouragement.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

Finding the motion from a mathematical model of a mechanical system usually involves solving a system of differential equations, often referred to as equations of motion (EOM). There are a number of methods that can be used to obtain the EOM. For example, free body diagrams for the forces acting on the system can be constructed and then Newton's Second Law ($F=Ma$) can be employed to write a system of second order differential equations for each of independent vector components of the system. Another approach for a conservative system is to construct the Lagrangian function (Kinetic Energy – Potential Energy) or the Hamiltonian function (Kinetic Energy + Potential Energy) and write the EOM after taking suitable derivatives of these functions according to the differential equations of Lagrange (second order) or Hamilton (first order), respectively.

Even for a system with many independent moving parts (degrees of mechanical freedom) it is often relatively easy to write the single function that represents the total kinetic energy of the system or the total potential energy of the system as the scalar sum of the kinetic or potential energy of each of its parts. Therefore, to obtain the system Lagrangian or Hamiltonian function even for large systems is quite practical. Although in principle, it is a rather straightforward procedure to obtain the system of differential equations from the Lagrangian or Hamiltonian,as a practical matter it can be a dauntingly long and tedious process when there are many mechanical degrees of freedom in the system (Ref # 1).In this case the Lagrangian has many terms and there are a very large number of partial derivatives that need to be computed for each mechanical degree of

freedom making the procedure quite difficult and often impractical to carry out for mechanical systems with a large number of moving parts.

It can be shown that Newton's, Lagrange's, and Hamilton's EOM can be derived from a single unifying principle called the Principle of Least Action. The Action is obtained by integrating the Lagrangian function over time. For each possible motion *x(t)* of a system that starts from a given configuration $x_0 = x(0)$ and ends after a given duration T at a given target configuration $x_T = x(T)$ the Action will take on a definite value A[x(t)]. The problem of solving for the actual path *x(t)* of the mechanical system given the two boundary points $x_0 = x(0)$ and $x_T = x(T)$ is called a 2-point boundary value problem (BVP). The Principle of Least Action states that the Action for the actual physical motion will always be a critical value (almost always a minimum of the Action) of the set of all the possible paths that satisfy the 2 boundary points and any constraints that may be acting on the system during its motion. Using the Calculus of Variations to find when the Action is critical leads to a derivation of Newton's, Lagrange's and Hamilton's general differential equations. It is these equations that are then implemented to obtain the EOM for any particular conservative mechanical system.

This thesis begins to explore the computational feasibility of solving the 2-point BVP of a model physical system from the Lagrangian function *using* the Principle of Least Action *directly without* using any EOM that may be derived from it and therefore avoiding the difficulties mentioned above.

Given the EOM for a particular mechanical system it is usually much more difficult to use them computationally to solve a 2-point BVP than to use the EOM to solve an initial value problem (IVP). In an IVP the starting configuration $x_0 = x(0)$ and

the starting system velocity $v_0 = \dot{x}(0)$ are given and the EOM are often used iteratively to find the system configuration and velocity at successive time increments until a desired time T is reached. Finding the solution to a given human motor task by applying the EOM to an IVP is often not practical because it usually is not known what initial velocity to give each body segment so that the system will arrive at a desired target position from a given starting configuration. Therefore for many human motion problems the 2-point BVP is a more practical and natural formulation of the motor task to be accomplished. A new mathematical technique developed in our laboratory called the *Boundary Method* solves human motion problems as a concatenation of independent 2-point BVPs. This method also avoids the difficulty of requiring a priori knowledge of muscle force terms in the EOM. The Boundary Method solves simultaneously for *both* motions that can accomplish a given motor task *and* the net muscular joint forces required to produce those motions.

One technique called the shooting method solves the 2-point BVP by successive iteration of IVP solutions. The shooting method starts with a guess for $v_0 = \dot{x}(0)$ and the EOM are solved as an IVP up to time T. The error of this solution at time T when compared to the target configuration x_T is calculated as $e = x(T) - x_T$. This error together with similar error vectors obtained from IVP solutions from a finite and usually small number of *nearby* velocity guesses is used to generate a *linear* best guess for $v_0 = \dot{x}(0)$ that will solve the 2-point BVP as an IVP. This process continues iteratively until e achieves an acceptably small magnitude. Therefore using the EOM in the shooting method to solve a 2-point BVP usually involves solving the EOM many times and this can be computationally expensive.

The Principle of Least Action naturally poses the motion problem as a 2-point BVP and therefore for the reasons described in the paragraphs above its application to solving human motion problems is more direct than using EOM and in particular for applications that use the new *Boundary Method* developed in our laboratory. In this thesis we will begin to explore the numerical feasibility of solving motion 2-point BVPs by direct numerical minimization of the action without using the EOM as an intermediate step. In this approach we start with a guess $x_g(t),\ 0 \le t \le T$ or a finite number of guesses of the mechanical solution path. Each guess of the given 2-point BVP to be solved starts at the known starting configuration $x_0 = x(0)$ and ends at the known final target configuration $x_T = x(T)$ and is represented numerically by a vector of points $(x_o, x_1, \cdots, x_N, x_T)$ where $x_i = x_g(t_i), i = 1, \cdots, N$. For each guessed solution the Lagrangian is calculated and integrated over the given time interval T to obtain the Action value associated with that solution guess $A(x_o, x_1, \cdots, x_N, x_T)$. The action values for nearby guesses are then used to generate new solution paths with lower action values that also satisfy the boundary conditions. Continuing in this systematic iterative fashion paths are generated that converge towards a minimum or least action solution path. As shown by Lagrange (see Chapter 2) this solution is a critical point of the Action and therefore satisfies the Lagrange EOM and consequently represents a 2-point BVP solution of the mechanical system that starts at x_0 and ends after a time interval of T at the target x_T.

Many numerical techniques are available that can be used to minimize a multidimensional function $f(x_1, \cdots, x_N)$. The chosen method in this thesis the *downhill*

simplex method (DSM) (Ref # 2). Although there are many faster minimization methods than the DSM, this algorithm was chosen because it is very simple to implement numerically requiring only a minimum of code lines (less than 100 lines of C code) and also because it has proved to be a very stable minimization method for a robust set of functions in part because it does not require derivative evaluation.

In this thesis the proposed *direct least action* (DLA) approach using the DSM is applied to both the single and double pendulum systems as beginning test problems. The solutions so obtained are compared to numerical solutions of the corresponding Lagrange EOM for the same mechanical problems. The Lagrange EOM are solved using an explicit first order (Euler) algorithm.

CHAPTER 2

LAGRNAGIAN METHOD

2.1 Introduction

In this chapter we will define the Lagrangian function, the Action functional, and show how the Lagrange EOM are generated from the principle of least action (PLA). A solution that satisfies Lagrange EOM will have an action value that is extreme (minimal) relative to all other possible nearby paths that satisfy both the boundary conditions and any other constraints on the dynamic variables that are used to describe a mechanical system. More precisely, in Section 2.3 we will show that the Lagrange EOM emerge when the functional derivative of the action at the critical physical path is set to 0. In Section 2.4 the theory and equations developed in Section 2.3 are applied to determine the Lagrangian, Action and EOM for some simple examples including the single and double pendulum that will be used in Chapter 3 as beginning tests for the DLA approach proposed in this thesis.

2.2 The Principle Of Least Action (PLA)

Consider a simple particle moving in a one-dimensional space and acted upon by a single conservative force field. A concrete example would be a particle moving up or down in a single vertical line where the only force that acts upon it is gravity. Suppose also that it starts at a given position x_0 and arrives at time T at given target position x_T. Let the actual physical motion path taken by the particle that solves this 2-point BVP be $\bar{x}(t)$. For the concrete example of a particle moving vertically in a constant gravitational field $\bar{x}(t)$ will be parabolic.

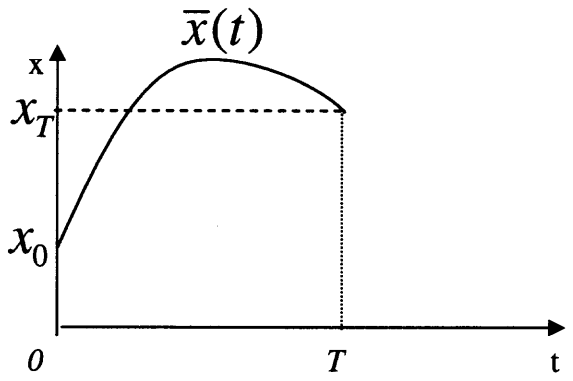


Figure 2.1 Sketch of the graph of the actual motion path $\bar{x}(t)$ followed by a particle in a conservative force field.

At each time t along the path the particle has a potential energy $P(\bar{x}(t))$ and a kinetic energy $K = \frac{1}{2} m\dot{\bar{x}}^2(t)$ where m is the particle mass and $\dot{\bar{x}}(t)$ is the slope (velocity) of the path at t. The Lagrangian for the particle is defined as $L \equiv K - P$ and its value at each time t depends upon both the particles position and velocity at t,

$$\bar{L}(t) = L(\bar{x}(t), \dot{\bar{x}}(t)) = \frac{1}{2} m\dot{\bar{x}}^2(t) - P(\bar{x}(t)) \qquad (2.2.1a)$$

For a particle moving in a constant gravitational field with constant acceleration g, $P(\bar{x}(t)) = mg\bar{x}(t)$ and,

$$\bar{L}(t) = L(\bar{x}(t), \dot{\bar{x}}(t)) = \frac{1}{2} m\dot{\bar{x}}^2(t) - mg\bar{x}(t). \qquad (2.2.1b)$$

The action $\bar{A}$ for the actual physical particle path is defined as the integral or area under the $\bar{L}(t)$ curve between 0 and T,

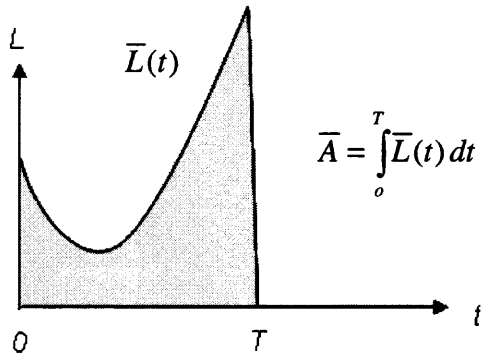


Figure 2.2 Sketch of a graph of the Lagrangian function and Action value associated with the physical particle path $\bar{x}(t)$ moving in a conservative force field.

Consider a path *x(t)* other than the actual physical particle path but which also satisfies the two boundary positions at *0* and *T*.

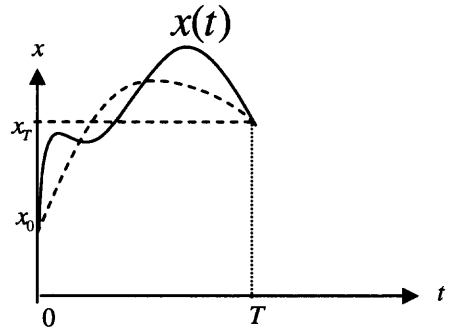


Figure 2.3 Sketch of a graph of the path *x(t)(solid curve)* and the actual particle path $\bar{x}(t)$ *(dashed curve).* Both curves satisfy the same boundary positions at *t=0* and *t=T*.

For this different (non-physical) path *x(t)* the particle would have a different a potential energy $P(x(t))$ and a kinetic energy $K = \frac{1}{2}m\dot{x}^2(t)$ than for the actual path $\bar{x}(t)$ and therefore the Lagrangian for the different path $L(t) = L(x(t), \dot{x}(t)) = \frac{1}{2}m\dot{x}^2(t) - P(x(t))$ will be different than the $\bar{L}(t)$ associated with the actual path $\bar{x}(t)$. Consequently the area under the $L(t)$ curve and the $\bar{L}(t)$ curve will differ and therefore the action values

$A[x(t)]$ and $\overline{A}[\overline{x}(t)]$ associated with the two paths will also not be the same. *The Principle of Least Action states that the action for the actual physical motion (path) will always be a critical value (almost always a minimum) of the set of all the possible nearby paths that satisfy the 2 given boundary positions and any constraints that may be acting on the dynamic variable of the mechanical system during its motion* (Ref # 3).

2.3 Derivation of Lagrange Equation of Motion (EOM) for a Single Particle from the Principle of Least Action (PLA)

Consider any smooth test function $\eta(t)$ defined on the interval $t \in [0,T]$ and taking on the value of $0 = \eta(0) = \eta(T)$ at the end points.

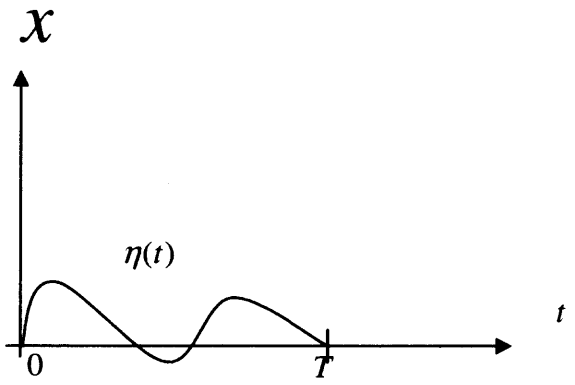


Figure 2.4 Sketch of a graph of a test function $\eta(t)$ used to obtain a path $x(t)$ near to the physical motion path $\overline{x}(t)$ such that both paths satisfy the same positions at the boundary points.

Note that if we add any such $\eta(t)$ to the actual particle path $\overline{x}(t)$ we will obtain another path $x(t) = \overline{x}(t) + \eta(t)$ that satisfies the given boundary points $x_0 = x(0)$ and $x_T = x(T)$. A new path $x(t) = \overline{x}(t) + \varepsilon\,\eta(t)$ can be made arbitrarily close to $\overline{x}(t)$ if the scaling factor ε is a number sufficiently close to 0. The action associated with $x(t)$ is given by

$$A[x(t)] = A[\bar{x}(t) + \varepsilon\eta(t)] = \int_0^T L(\bar{x}(t) + \varepsilon\eta(t), \dot{\bar{x}}(t) + \varepsilon\dot{\eta}(t))\, dt \tag{2.3c}$$

Expanding $L(\bar{x} + \varepsilon\eta, \dot{\bar{x}} + \varepsilon\dot{\eta})$ in a Taylor series about $L(\bar{x}, \dot{\bar{x}})$ gives,

$$L(\bar{x} + \varepsilon\eta, \dot{\bar{x}} + \varepsilon\dot{\eta}) = L(\bar{x}, \dot{\bar{x}}) + \frac{\partial L}{\partial x}\varepsilon\eta + \frac{\partial L}{\partial \dot{x}}\varepsilon\dot{\eta} + O(\in^e) \tag{2.3d}$$

Substitution of (2.3d) back into (2.3c) yields,

$$A[\bar{x}(t) + \varepsilon\eta(t)] = \int_0^T (L(\bar{x}, \dot{\bar{x}}) + \frac{\partial L}{\partial x}\varepsilon\eta + \frac{\partial L}{\partial \dot{x}}\varepsilon\dot{\eta} + O(\in^e))dt \tag{2.3e}$$

The change in the action between the nearby path and the actual path is

$$A[\bar{x}(t) + \varepsilon\eta(t)] - A[\bar{x}(t] = \int_0^T (\frac{\partial L}{\partial x}\varepsilon\eta + \frac{\partial L}{\partial \dot{x}}\varepsilon\dot{\eta} + O(\in^e))dt\,. \tag{2.3f}$$

We define the rate of change of the action at $\bar{x}(t)$ in the $\eta(t)$ direction by

$$\left.\frac{\delta A}{\delta \bar{x}}\right|_\eta = \lim_{\varepsilon\to 0}\frac{A[\bar{x}(t) + \varepsilon\eta(t)] - A[\bar{x}(t]}{\varepsilon} = \lim_{\varepsilon\to 0}\int_0^T (\frac{\frac{\partial L}{\partial x}\varepsilon\eta + \frac{\partial L}{\partial \dot{x}}\varepsilon\dot{\eta} + O(\in^e)}{\varepsilon})dt$$

$$= \int_0^T \left(\frac{\partial L}{\partial x}\eta + \frac{\partial L}{\partial \dot{x}}\dot{\eta}\right)dt \tag{2.3g}$$

Defining $u(t) \equiv \dfrac{\partial L}{\partial \dot{x}}$ and $dv \equiv \dot{\eta}dt$ and using integration by parts gives,

$$\int_0^T \frac{\partial L}{\partial \dot{x}}\dot{\eta}\, dt = \int_{t=0}^{t=T} u\, dv = uv\Big|_{t=0}^{t=T} - \int_{t=0}^{t=T} v\, du$$

$$= \frac{\partial L}{\partial \dot{x}}\eta\Big|_{t=0}^{t=T} - \int_{t=0}^{t=T} \eta\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right)dt \tag{2.3h}$$

Since $0 = \eta(0) = \eta(T)$ equation (2.3h) simplifies to,

$$\int_0^T \frac{\partial L}{\partial \dot{x}} \dot{\eta}\, dt = -\int_{t=0}^{t=T} \eta \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) dt \tag{2.3i}$$

Substituting equation (2.3i) into equation (2.3g) results in,

$$\left.\frac{\delta A}{\delta \bar{x}}\right|_{\eta} = \int_0^T \left(\frac{\partial L}{\partial x} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right)\right)\eta\, dt \tag{2.3j}$$

Define $w(t) \equiv \frac{\partial L}{\partial x} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right)$ then equation (2.3j) defines a (Hilbert) vector function dot product between the two functions w and η,

$$\left.\frac{\delta A}{\delta \bar{x}}\right|_{\eta} = \int_0^T \left(\frac{\partial L}{\partial x} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right)\right)\eta\, dt = w \bullet \eta \tag{2.3k}$$

According to PLA the actual physical path will the one which will be critical for A and therefore the derivative $\left.\frac{\delta A}{\delta \bar{x}}\right|_{\eta} = w \bullet \eta$ must be 0 for all possible smooth choices of η functions (that satisfy $0 = \eta(0) = \eta(T)$) and that would be used to define any nearby path to $\bar{x}(t)$ satisfying the 2 boundary point conditions. If $w \bullet \eta = 0$ for all possible choices of $\eta(t)$ then the only vector function $w(t)$ in the (Hilbert) vector space that is orthogonal to all other vector functions $\eta(t)$ in the space is the 0 function that takes on the value 0 for all $t \in [0,T]$. Therefore,

$$\frac{\partial L}{\partial x} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0 \tag{2.3l}$$

Equation (2.3l) is equivalent to the Lagrange EOM for a single particle, which is usually written as,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right)-\frac{\partial L}{\partial x}=0 \tag{2.3m}$$

A similar argument can be made to show when a system consists of many moving parts with N degrees of freedom in a conservative force field then the PLA yields the Lagrange EOM for the system.

$$\boxed{\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_i}\right)-\frac{\partial L}{\partial x_i}=0 \quad i=1,\cdots,N} \tag{2.3n}$$

In this case the motion of the system will be described by the vector $\bar{x}(t)=(\bar{x}_1(t),\cdots,\bar{x}_N(t))$ and the Lagrangian of the system $L(\bar{x},\dot{\bar{x}})=K-P$ is a scalar function of 2N-unconstrained variables.

2.4 The Lagrangian, Action and EOM for Some Simple Physical Examples

2.4.1 Single Particle Moving Vertically in a Constant Gravitational Field

As shown in Equation (2.2.1b) the Lagrangian for a free particle of mass *m* moving freely up and down in a constant gravitational field with gravitation constant *g* is given by, $L(x,\dot{x}) = K - P = \frac{1}{2}m\dot{x}^2 - mgx$, where *x(t)* denotes the height of the particle above the ground (*x= 0*). For such a system the partial and Eulerian derivatives are,

$$\frac{\partial L}{\partial x} = -mg\ , \quad \frac{\partial L}{\partial \dot{x}} = m\dot{x}\ , \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = \frac{d}{dt}(m\dot{x}) = m\ddot{x} \tag{2.4.1a}$$

Applying the results of Equation (2.4.1a) above to the Lagrange EOM for a single particle Equation (2.3m) derived in the previous section yields the familiar Newtonian equation for vertical motion in a constant gravitational field,

$$m\ddot{x} + mg = 0 \text{ or equivalently } \ddot{x} = -g. \tag{2.4.1b}$$

Integrating Equation (2.4.1b) twice gives the familiar Galilean solution for the IVP,

$$x(t) = x_0 + v_0 t - \frac{1}{2}g t^2 \tag{2.4.1c}$$

Where x_0 and v_0 are the initial height and velocity of the particle. The solution of the 2-point BVP is Equation (2.4.1c) but v_0 is now a variable to be determined from the starting height, x_0, target height, x_T and time to target, T, and is given by,

$$\text{Where } v_0 = \left(\frac{x_T - x_0}{T}\right) + \frac{1}{2}g T \tag{2.4.1d}$$

The value of the action for the physical path is given by,

$$\overline{A} = \int_o^T \overline{L}(t)\,dt = \int_o^T \left[\frac{1}{2}m(v_0 - g t)^2 - mg\left(x_0 + v_0 t - \frac{1}{2}g t^2\right)\right]dt \tag{2.4.1e}$$

2.4.2 Unconstrained Single Pendulum Moving in a Constant Vertical Gravitational Field

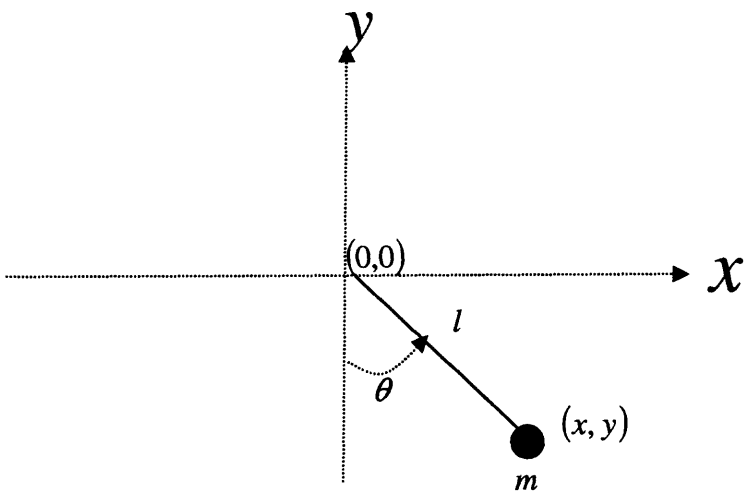


Figure 2.5 Single Pendulum Model.

Consider a simple pendulum freely moving in a constant vertical gravitational field consisting of a single mass point with mass m and weight mg attached to the end of a weightless rod of fixed length l pivoting about the origin (see Fig 2.5). Denote the counterclockwise angle that the pendulum makes with respect to the vertical at time t by $\theta(t)$. The coordinates of the mass point and its velocity components are,

$$x = l * \sin\theta \qquad \dot{x} = l * \dot{\theta} * \cos\theta$$

$$y = -l * \cos\theta \qquad \dot{y} = l * \dot{\theta} * \sin\theta \tag{2.4.2a}$$

The kinetic energy of the pendulum is,

$$K = \frac{1}{2}mv^2 = \frac{1}{2} * m * l^2 * \dot{\theta}, \text{ where } v^2 = \dot{x}^2 + \dot{y}^2 = l^2\dot{\theta}^2 \tag{2.4.2b}$$

The Potential Energy of the pendulum is given by, $P = mgh$ where h denotes the height of the pendulum above its vertical equilibrium position ($\theta = 0$). Since $h = l(1-\cos\theta)$,

$$P = mgl(1-\cos\theta) \tag{2.4.2c}$$

Therefore the Lagrangian of the pendulum system expressed in terms of the dynamic angle θ is given by,

$$L(\theta,\dot{\theta}) = K - P = \frac{1}{2}ml^2\dot{\theta}^2 - mgl(1-\cos\theta) \tag{2.4.2d}$$

Differentiating Equation (2.4.2d) we obtain,

$$\frac{\partial L}{\partial \dot{\theta}} = ml^2\dot{\theta}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = ml^2\ddot{\theta}$$

$$\frac{\partial L}{\partial \theta} = -mgl\sin\theta \tag{2.4.2e}$$

Substituting Equations (2.4.2e) into Lagrange's EOM (Equation 2.3m) gives,

$$ml^2\ddot{\theta} + mgl\sin\theta = 0$$
$$ml^2\ddot{\theta} = -mgl\sin\theta \tag{2.4.2f}$$

Equation (2.4.2f) is the same as would be derived from Newton's Second Law, applying the usual free body diagram approach and considering the components of the force in the tangential and radial directions. Equation (2.4.2f) is Newton's force law applied to the tangential direction since,

$$F_T = -mgl\sin\theta \tag{2.4.2f*}$$

is the component of the weight in the tangential direction, and the acceleration in that direction is $\ddot{s} = \frac{d^2 s}{dt^2} = \frac{d^2(l\theta)}{dt^2} = l * \ddot{\theta}$ where s is arc length. There are no dynamics associated with the radial direction since the mass is constrained in the radial direction to be a fixed length l from the stationary central pivot point. Motion only occurs in the tangential direction. As in the case of the free particle moving in a constant gravitational field the motion path is independent of the mass of the pendulum since m is on both sides of Equation (2.4.2f) and can be cancelled out yielding the EOM for the simple pendulum without friction.

$$\ddot{\theta} = -\frac{g}{l}\sin\theta \tag{2.4.2g}$$

Note that for small oscillations $\sin\theta \approx \theta$ and Equation (2.4.3h) formally approaches the linear equation that describes the simple harmonic oscillator (SHO) modeled by a linear spring- mass system without friction,

$$\ddot{\theta} = -\frac{g}{l}\theta \tag{2.4.2h}$$

The solution to Equation (2.4.3h) is sinusoidal,

$$\theta(t) = a\cos(\omega t - b) \tag{2.4.2i}$$

Where the oscillation frequency,

$$\omega = \sqrt{\frac{g}{l}} \tag{2.4.2j}$$

and the constants a and b are determined from the initial or boundary conditions. If $\theta(0) = \theta_0$ and $\dot{\theta}(0) = 0$ then $a = \theta_0$ and $b = 0$. For small θ, the Taylor expansion

of $\cos\theta \approx 1-\frac{\theta^2}{2}$. Thus, for small enough oscillations the pendulum Lagrangian function of Equation (2.4.2d) approaches,

$$L(\theta,\dot{\theta}) = K - P = \frac{1}{2}ml^2\dot{\theta}^2 - \frac{1}{2}mgl\theta^2 \qquad (2.4.2k)$$

Substituting $x = l\theta$ (arc length) into Equation (2.4.2k) formally yields the Lagrangian for the SHO,

$$L(x,\dot{x}) = K - P = \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2 \qquad (2.4.2l)$$

Where x is the displacement of the spring from its equilibrium position and the spring stiffness constant is $k = \frac{mg}{l}$ which is consistent with $\omega^2 = \frac{k}{m}(SHO) = \frac{g}{l}$ (Equation 2.4.2j). Since the restoring force of the spring is F=-kx , its potential energy (negative work integral) will be given by,

$$P = -\int_0^x F\, d\tilde{x} = -\int_0^x (-k\tilde{x})d\tilde{x} = \frac{1}{2}kx^2 \qquad (2.4.2m)$$

The restoring force for a pendulum in small oscillation is,

$$F = -kx = -\left(\frac{mg}{l}\right)(l\theta) = -mg\theta \cong -mgl\sin\theta = F_T$$

which approximates the tangential force component of the pendulum's weight (Equation 2.4.2f*) for small θ.

The Action A for the true pendulum motion is given by,

$$\overline{A} = \int_0^T \left[\frac{1}{2}ml^2\dot{\theta}^2(t) - mgl(1-\cos\theta(t))\right] dt \qquad (2.4.2n)$$

For a pendulum in small oscillation $\overline{A}$ can be directly calculated by integrating Equation (2.4.2k) using the SHO solution for $\theta(t)$ (Equation 2.4.2i). The result is,

$$\overline{A} = -\frac{1}{4} m\omega l^2 a^2 [\sin(2(\omega T - b) + \sin(2b)] \tag{2.4.2o}$$

In particular, for a pendulum that starts from rest at small angle θ_0 ($a = \theta_0$ and $b = 0$), a swings for half a period (from θ_0 to $-\theta_0$) then $T = \frac{\pi}{\omega}$ and,

$$\overline{A} = -\frac{1}{4} m\omega l^2 x_0{}^2 \sin(2\omega T) = 0 \tag{2.4.2p}$$

2.4.3 Unconstrained Double Pendulum Moving in a Constant Vertical Gravitational Field

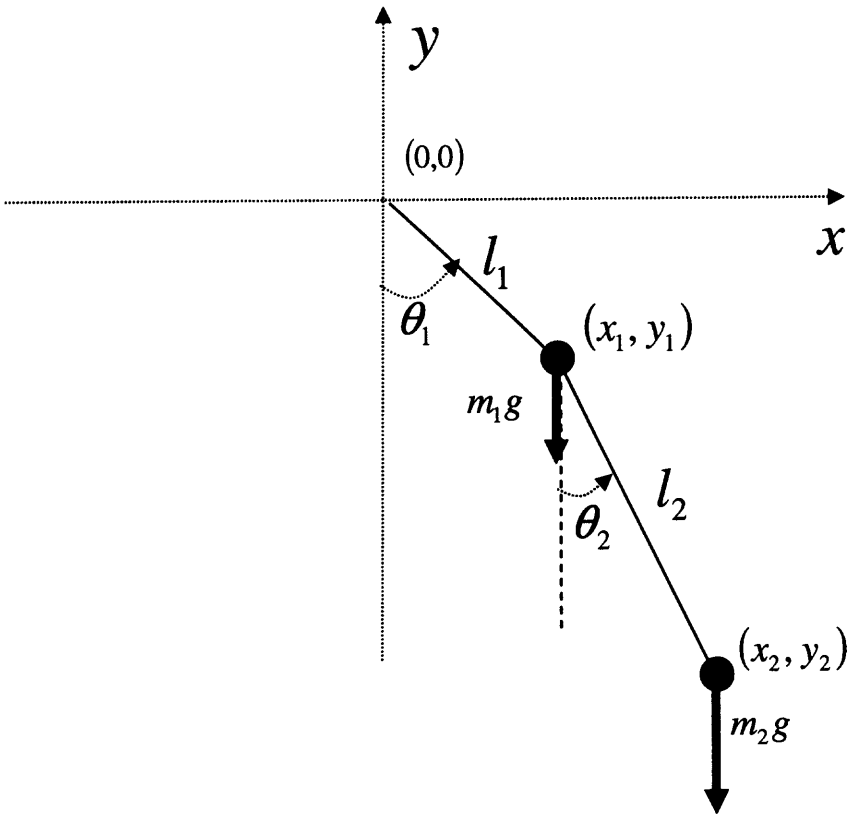


Figure 2.6 Double Pendulum Model.

Consider a double pendulum system with masses m_1 and m_2 attached to rigid rods of length l_1 and l_2 as shown in the above figure .Let the two pendulums make angle θ_1 and θ_2 with the vertical respectively (Figure 2.6). The x and y coordinates of the two point masses are given by

$$x_1 = l_1 \sin\theta_1 \qquad (2..4.3a)$$

$$y_1 = -l_1 \cos\theta_1 \qquad (2..4.3b)$$

$$x_2 = l_1 \sin\theta_1 + l_2 \sin\theta_2 \qquad (2..4.3c)$$

$$y_2 = -l_2 \sin\theta_1 - l_2 \cos\theta_2 \qquad (2..4.3d)$$

The total potential energy of a system can be calculated by adding up potential energies of both of the point masses of the system,

$$P = m_1 g h_1 + m_2 g h_2 = -(m_1 g l_1 \cos\theta_1) - m_2 g [l_1 \cos\theta_1 + l_2 \cos\theta_2]$$

$$P(\theta_1, \theta_2) = -(m_1 + m_2) g l_1 \cos\theta_1 - m_2 g l_2 \cos\theta_2 \qquad (2..4.3e)$$

The total kinetic energy of the system can be calculated by adding up kinetic energies of both point masses of the system.

$$K = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 \qquad (2.4.3f)$$

Differentiating Equations (2.4.3a-b) with respect to time and using the fact that

$$v_1^2 = v_{1x}^2 + v_{y1}^2 \text{ yields } v_1^2 = l_1^2 \dot{\theta}_1^2 \qquad (2.4.3g)$$

Similarly,

$$v_2^2 = l_1^2 \dot{\theta}_1^2 + 2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + l_2^2 \dot{\theta}_2^2 \qquad (2.4.3h)$$

Substitution into Equation (2.4.3f) yields,

$$K = \frac{1}{2} m_1 l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 \left(l_1^2 \dot{\theta}_1^2 + 2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + l_2^2 \dot{\theta}_2^2 \right) \qquad (2.4.3i)$$

The kinetic energy Equation (2.4.3i) can be written more compactly in a form similar to that of a single particle as,

$$K = \frac{1}{2} \dot{X}^T M \dot{X} \qquad (2.4.3j)$$

where,

$$M_{ij} = \begin{bmatrix} l^2{}_1 (m_1 + m_2) & l_1 l_2 m_2 \cos(\theta_1 - \theta_2) \\ l_1 l_2 m_2 \cos(\theta_1 - \theta_2) & l^2{}_2 m_2 \end{bmatrix} \qquad (2.4.3k)$$

is a generalized mass matrix and ,

$$X = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}, \quad \dot{X} = \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix} \tag{2.4.3l}$$

are respectively the generalized position and velocity vectors. Note that M is not a constant matrix but is a function of the generalized position. It is not a function of the generalized velocity. It is also symmetric and positive definite; (kinetic energy is never negative). The Action for the double pendulum system is,

$$A[X(t)] = \int_0^T L\left(X(t), \dot{X}(t)\right) dt \tag{2.4.3m}$$

where,

$$\begin{aligned} L\left(X, \dot{X}\right) &= K(X, \dot{X}) - P(X) = \frac{1}{2} \dot{X}^T M \dot{X} - P(X) \\ &= \frac{1}{2}(m_1 + m_2) l^2{}_1 \dot{\theta}_1{}^2 + \frac{1}{2} m_2 l_2{}^2 \dot{\theta}_2{}^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \\ &\quad + (m_1 + m_2) g l_1 \cos\theta_1 + m_2 g l_2 \cos\theta_2 \end{aligned} \tag{2.4.3n}$$

is the Lagrangian for the double pendulum.

Because the double pendulum is an $N=2$ degree of freedom system, the Lagrange EOM will be a system of two differential equations; one obtained by taking partial derivatives of the Lagrangian Equation (2..4.3n) with respect to θ_1 and $\dot{\theta}_1$ and the other differential equation by taking partial derivatives of the Lagrangian with respect to the second dynamic variable θ_2 and $\dot{\theta}_2$. Applying the general Lagrange EOM (2.3n) that were developed in the previous section for an N-degree of freedom system with $\theta_1 = x_1$ and the L of Equation (2.4.3n) results in,

$$(m_1 + m_2) l_1{}^2 \ddot{\theta}_1 + m_2 l_1 l_2 \cos(\theta_1 - \theta_2) \ddot{\theta}_2 + m_2 l_1 l_2 \sin(\theta_1 - \theta_2) \dot{\theta}_2{}^2 + (m_1 + m_2) g l_1 \sin\theta_1 = 0 \tag{2.4.3o}$$

Applying Equation (2.3n) with the L of Equation (2.4.3n) and with $\theta_2 = x_2$ gives,

$$m_2 l_1 l_2 \cos(\theta_1 - \theta_2)\ddot{\theta}_1 + m_2 l_2^2 \ddot{\theta}_2 - m_2 l_1 l_2 \sin(\theta_1 - \theta_2)\dot{\theta}_1^2 + m_2 g l_2 \sin\theta_2 = 0 \qquad (2.4.3p)$$

Equations (2.4.3o and 2.4.3p) can be arranged as a system in a more compact form that looks formally similar to Newton's Second Law :

$$M\ddot{X} = F \qquad (2.4.3q)$$

where M is the generalized mass matrix of Equation (2.4.3k) and where

$$F = F_g + F_s \qquad (2.4.3r)$$

The generalized force terms in (2.4.3r) are,

$$F_g(X) = -\nabla P = -\begin{bmatrix} \frac{\partial P}{\partial \theta_1} \\ \frac{\partial P}{\partial \theta_2} \end{bmatrix} = -\begin{bmatrix} (m_1 + m_2) g l_1 \sin\theta_1 \\ m_2 g l_2 \sin\theta_2 \end{bmatrix} \qquad (2.4.3s)$$

which is a generalized gravitational force vector obtained from the gradient of the potential energy and

$$F_s(X, \dot{X}) = S(X)\ \dot{X}^{(2)} = \begin{bmatrix} 0 & -l_1 l_2 m_2 \sin(\theta_1 - \theta_2) \\ l_1 l_2 m_2 \sin(\theta_1 - \theta_2) & 0 \end{bmatrix} \begin{bmatrix} (\dot{\theta}_1)^2 \\ (\dot{\theta}_2)^2 \end{bmatrix} \qquad (2.4.3t)$$

Like the generalized M matrix, the S matrix is a function of position but not of velocity and although it is not symmetric it is skew-symmetric. In fact, both matrices can be shown to be closely related to the following non-dynamic symmetric matrix,

$$\mathrm{C} = \begin{bmatrix} (m_1 + m_2) l_1^2 & m_2 l_1 l_2 \\ m_2 l_1 l_2 & m_2 l_2^2 \end{bmatrix} \qquad (2.4.3u)$$

in that $M_{ij} = C_{ij}\cos(\theta_i - \theta_j)$ and $S_{ij} = C_{ij}\sin(\theta_i - \theta_j)$ $\quad i = 1,2 \quad j = 1,2$

CHAPTER 3

TWO SOLUTION METHODS FOR MECHANICAL SYSTEMS

In this chapter we describe two approaches to numerically determine the motion of a mechanical system: (1) the Direct Least Action (DLA) approach proposed in this thesis and (2) numerical integration of the differential EOM. The DLA approach is explained in Section 3.1 and its implementation by the downhill simplex method is described in Section 3.2. In Section 3.3 Eulers method is described to solve the EOM for IVPs. In section 3.4 the solutions to the same single and double pendulum problems by the two approaches are obtained and compared. Concluding remarks are made in Section 3.5.

3.1 The Direct Least Action (DLA) Approach for Solving Mechanical 2-Point Boundary Value Dynamics Problems

In the previous chapter, it was shown that the Action integral assigns a scalar to any possible motion path that satisfies boundary or other constraints on the dynamic variables. This mathematically defines the Action as a scalar functional rather than a scalar function. More precisely, a scalar function of multiple variables $f(\vec{x}) = f(x_1, \cdots, x_N)$ acts upon a domain set that consists of vectors $\vec{x} = (x_1, \cdots, x_N)$ and assigns only one scalar value to any given member of the domain set. This is also true of a scalar functional except that the domain set is not a collection finite dimensional vectors but rather a set of functions.

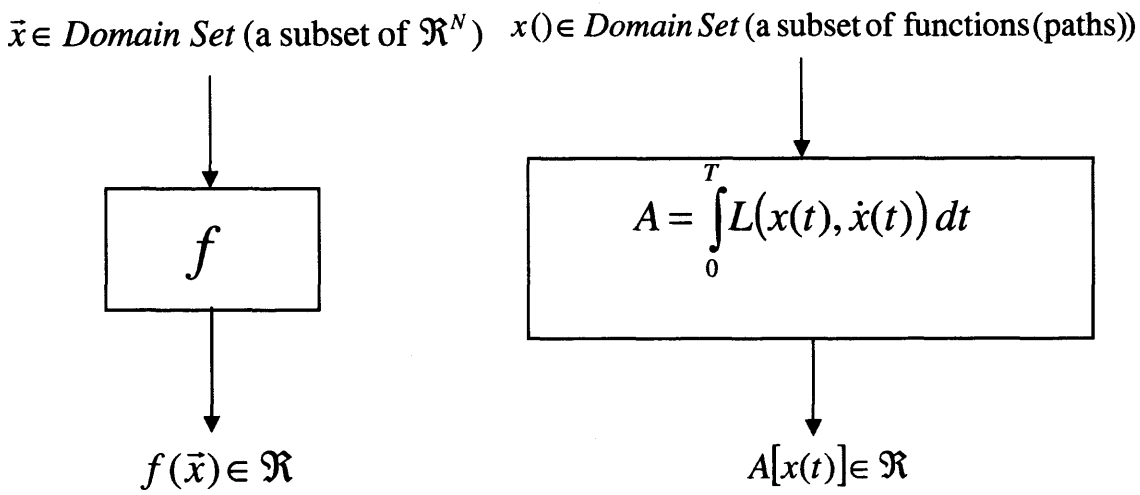


Figure 3.1 Comparison of a real valued function of multiple variables to a real valued functional. The Domain of the function is a set of vectors and the domain of a functional is a set of functions (paths x(t)).

The numerical application of the DLA principle requires that the action functional be approximated by scalar function of multiple variables. This can be accomplished by suitably approximating a function $x(t)$ by a finite dimensional vector that also satisfies the boundary and other constraints on dynamic variables of the system. This can be accomplished as follows. Let $x(t)$ represent any such function and consider the vector, $\vec{x} = (x_o, x_1, \cdots, x_N, x_T)$ whose components are defined by , $x_i = x(t_i), i = 1, \cdots, N$ so that they agree with the function at each of the times $t_i, i = 1, \cdots, N$ as well as at the boundaries $x_0 = x(0)$ and $x_T = x(T)$. For example the times of agreement can be chosen to be equally spaced at intervals of $\Delta t = \frac{T}{N}$ between the [0,T] if $t_i = i\Delta t, i = 1, \cdots, N$ .In the limit as $N \to \infty$ $\vec{x} = x(t)$ and therefore a function can be regarded as a vector in an infinite dimensional vector space and the scalar functional can formally to be considered a scalar function with an infinite number of arguments. $A(x_1, x_2, \cdots, x_\infty)$.

$$A[x(t)] = \int_0^T L(x(t), \dot{x}(t))\, dt = \lim_{N\to\infty} A(\vec{x} = (x_1, \cdots x_N)) \equiv \lim_{N\to\infty} \sum_{i=0}^{N} L(\vec{x}_i, \dot{\vec{x}}_i)\, \Delta t$$

For large enough N, $A[x(t)] \cong A(\vec{x} = (x_1, \cdots x_N))$. The vector $\dot{\vec{x}}$ can be obtained from $\vec{x}$ by numerical differentiation. For example a centered difference method can be employed, $\dot{\vec{x}}_i = \frac{x_{i+1} - x_{i-1}}{2\Delta t} \cong \dot{x}(t_i) \quad i = 1, \cdots, N$. The problem of finding an actual motion that solves the 2-point BVP for the mechanical system has been transformed into finding the local extreme of the scalar multivariable function defined on a domain that is a subset of a finite dimensional vector space that satisfies the mechanical constraints of the system. This defines the DLA approach of this thesis and any multivariable minimization algorithm may be applied to implement it. In the next section we will describe the downhill simplex method (DSM) that is used in this thesis. For the double pendulum problem in which the Lagrangian is a function of 2 dynamic variables, two finite vectors $\vec{\theta}_1$ and $\vec{\theta}_2$ are used in the Lagrangian to approximate the functions $\theta_1(t)$ and $\theta_2(t)$.

3.2 Multidimensional Minimization of a Function

Multidimensional minimization function finds the minimum of a function of more than one independent variable. There are several different methods to minimize a function "*f*" that has "N" independent variables like Downhill simplex method, Powell's method, Conjugate gradient method and Quasi–Newton method.

There are several factors that play key role in selecting the suitable minimization function for particular usage. Some methods need only evaluation of the functions to be minimized and other methods require evaluations of the derivative of that function. The amount of storage required and conciseness of the program are also important.

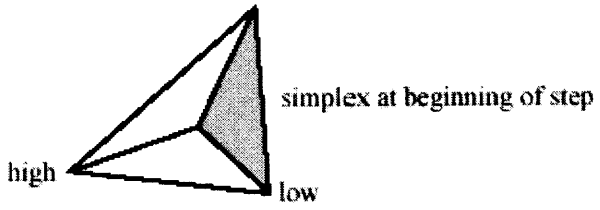
Downhill simplex method (DSM) is an N dimensional geometrical figure with N+1 points (or vertices). In two dimensions, a simplex is a triangle and in three dimensions it is a tetrahedron (Ref # 1). Nedler and Mead developed this method. The simplex method is an algorithm which makes its way downhill through an N-dimensional topography, until it encounters a (local, at least) minimum. The method is concise, completely self-contained and evaluates only the function and not derivatives. A general N-dimensional minimization program is less than 100 lines and the storage requirement is of the order N^2.

In multidimensional minimization, the input is a starting guess, which is an N-vector of independent variables as the first point. Then DSM must be started with N+1 points, defining an initial simplex. If k_0 is the initial starting point then you can take the other N points to be

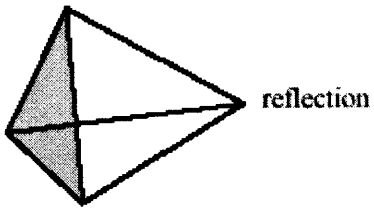
$$k_i = k_0 + \delta_i$$

Where i is N unit vectors and δ is a constant, which is a guess of the problem's characteristics length scale. The downhill simplex method takes a series of steps, moving the point of the simplex where the function is largest (highest point) through the opposite face of the simplex to a lower point. These steps are called reflections and are constructed to conserve the volume of the simplex to maintain its non-degeneracy. When it can do so, the method expands the simplex in one or another direction to take larger steps. When it reaches the valley floor the method contracts itself in the transverse direction. If there is a

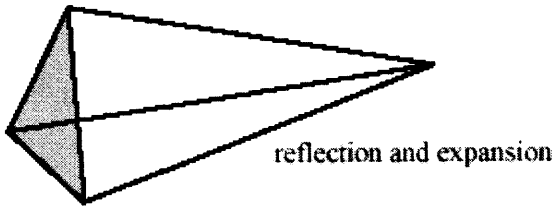
situation where the simplex is trying to pass through the eye of a needle, it contracts itself in all direction, pulling itself in around its lowest/best point .The routine name amoeba is used to describe this behavior. The basic steps are summarized in the figure below. The simplex at the beginning of a step is a tetrahedron.



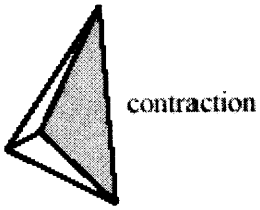
At the end the simplex can be one of following. An appropriate sequence of such step will always converge to a minimum of the function.



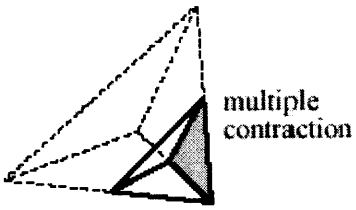
A reflection away from the high point or



A reflection and expansion away from the highest point or



A contraction along one dimension from the high point, or



A contraction along all the dimensions towards the low point.

Figure 3.2 Shows the starting and possible outcomes in the DSM.

The termination criterion for the simplex method is when the vector distance moved in a step is fractionally smaller than a some tolerance " tol ". Alternatively, the function can be terminated when the terminating step is fractionally smaller than some tolerance "ftol".

The flow chart below indicated the sequence of steps to achieve the least action path using simplex method

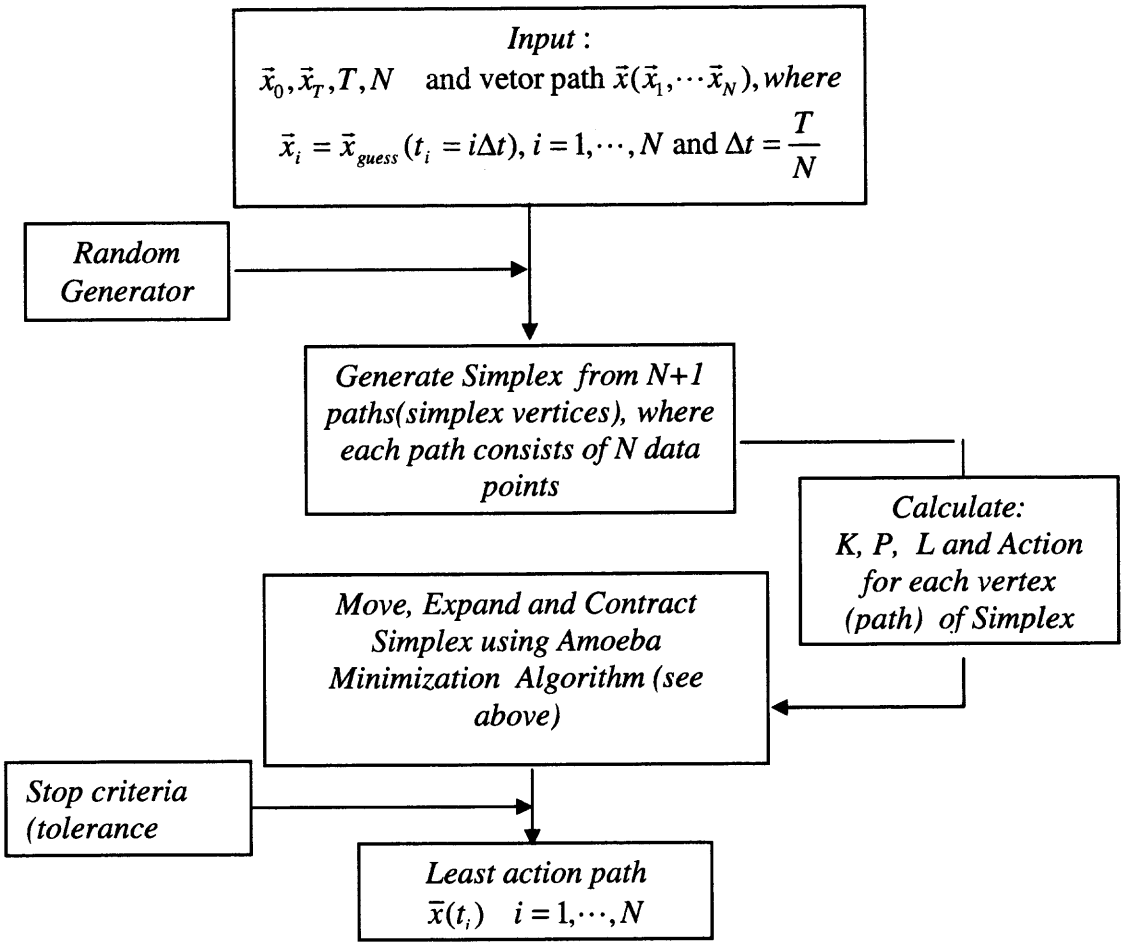


Figure 3.3 Flow chart indicating the sequence of steps to get least action path using simplex method.

3.3 Euler's Method for Solving the IVP

Euler's method is an iterative algorithm in time that approximates the pair,

$$\vec{y}(t++\Delta t)=\begin{pmatrix}\vec{x}(t+\Delta t)\\ \vec{v}(t+\Delta t)\end{pmatrix} \tag{3.3a}$$

from the similar pair at the previous time step,

$$\vec{y}(t)=\begin{pmatrix}\vec{x}(t\\ \vec{v}(t)\end{pmatrix} \tag{3.3b}$$

The EOM can often be written in matrix vector form as a generalized

Newton's 2nd Law:

$$M\ (\vec{x})\ \dot{\vec{v}}\ =\ F\ (\vec{x}\ ,\vec{v}\ ) \tag{3.3c}$$

combining this with $\dot{\vec{x}}=v$ yields the following system:

$$\begin{pmatrix}I & 0\\ 0 & M(\vec{x}(t)\end{pmatrix}\begin{pmatrix}\dot{\vec{x}}(t)\\ \dot{\vec{v}}(t)\end{pmatrix}=\begin{pmatrix}\vec{v}(t)\\ F(\vec{x}(t),\vec{v}(t))\end{pmatrix} \tag{3.3d}$$

where I is the identity matrix and M and F are the generalized mass matrix and force vectors respectively.

For example, for the single pendulum

$$I=1, M=m, \vec{x}=\theta, \vec{v}=\omega=\dot{\theta}, \text{and } F=-\frac{g}{l}\sin\theta \text{ (see Section 2.4.2)} \tag{3.3e}$$

For the double pendulum,

$$I=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$

$$M = \begin{bmatrix} l_1^{\,2}(m_1 + m_2) & l_1 l_2 m_2 \cos(\theta_1 - \theta_2) \\ l_1 l_2 m_2 \cos(\theta_1 - \theta_2) & l_2^{\,2} m_2 \end{bmatrix} \tag{3.3f}$$

$$\vec{x} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}, \vec{v} = \begin{pmatrix} \omega_1 \\ \omega_2 \end{pmatrix} = \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix} \tag{3.3g}$$

and

$$F(\vec{x}, \vec{v}) = -\begin{bmatrix} (m_1 + m_2) g l_1 \sin\theta_1 \\ m_2 g l_2 \sin\theta_2 \end{bmatrix} + \begin{bmatrix} 0 & -l_1 l_2 m_2 \sin(\theta_1 - \theta_2) \\ l_1 l_2 m_2 \sin(\theta_1 - \theta_2) & 0 \end{bmatrix} \begin{bmatrix} (\omega_1)^2 \\ (\omega_2)^2 \end{bmatrix} \tag{3.3h}$$

(see Section 2.4.3)

If $\vec{x}(t)$ and $\vec{v}(t)$ are known at any time *t*, (for the IVP they are given at *t=0* to start the iterative algorithm) then their substitution into Equation (3.3c) yields the linear algebra problem for the system,

$$A\, \dot{\vec{y}}(t) = b$$

where $A = \begin{pmatrix} I & 0 \\ 0 & M(\vec{x}(t)) \end{pmatrix}$

and

$$b = \begin{pmatrix} \vec{v}(t) \\ F(\vec{x}(t), \vec{v}(t)) \end{pmatrix}$$

are known and therefore the inverse A^{-1} can be computed at that time *t*. The inverse matrix A^{-1} can be used to solve for

$$\dot{\vec{y}}(t) = \begin{pmatrix} \dot{\vec{x}}(t) \\ \dot{\vec{v}}(t) \end{pmatrix} = A^{-1} b \tag{3.3i}$$

The first iteration of this equation in the algorithm would give, $\dot{\vec{y}}(0) = \begin{pmatrix} \dot{\vec{x}}(0) \\ \dot{\vec{v}}(0) \end{pmatrix} = A^{-1} b\big|_{t=0}$. The vector $\dot{\vec{y}}(t)$ is used to estimate the pair at the next time step from the approximation, $\vec{y}(t++\Delta t) = \begin{pmatrix} \vec{x}(t+\Delta t) \\ \vec{v}(t+\Delta t) \end{pmatrix} \cong \vec{y}(t) + \dot{\vec{y}}(t)\Delta t$, which is first order accurate in Δt. In the first iteration Equation (3.3i) would be used to obtain $\begin{pmatrix} \vec{x}(\Delta t) \\ \vec{v}(\Delta t) \end{pmatrix}$ from $\begin{pmatrix} \vec{x}_0 \\ \vec{v}_0 \end{pmatrix}$.The next iteration would yield $\begin{pmatrix} \vec{x}(2\Delta t) \\ \vec{v}(2\Delta t) \end{pmatrix}$ from $\begin{pmatrix} \vec{x}(\Delta t) \\ \vec{v}(\Delta t) \end{pmatrix}$ and so on. In this way the solution to the IVP is approximated for $\begin{pmatrix} \vec{x}(t_i = i\Delta t) \\ \vec{v}(t_i = i\Delta t) \end{pmatrix}$ $i = 0,1,2\cdots$ and converges to the exact solution of the IVP as $\Delta t \to 0$.

For the single pendulum,

$$A^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & m^{-1} \end{pmatrix}$$

For the double pendulum,

$$A^{-1} = \begin{pmatrix} I & 0 \\ 0 & M^{-1} \end{pmatrix}$$

where

$$M^{-1} = \begin{bmatrix} \dfrac{-1}{l_1^{\,2}[-(m_1+m_2)+m_2(\cos(\theta_1-\theta_2))^2} & \dfrac{-\cos(\theta_1-\theta_2)}{l_1 l_2(-m_2(\cos(\theta_1-\theta_2))^2+(m_1+m_{2)}} \\ \dfrac{\cos(\theta_1-\theta_2)}{[-l_1 l_2(m_1+m_2)+l_1 l_2(\cos(\theta_1-\theta_2))^2} & \dfrac{(m_1+m_2)}{l_2^{\,2} m_2(-m_2(\cos(\theta_1-\theta_2))^2+(m_1+m_2))} \end{bmatrix}$$

3.4 Results

3.4.1 Single Pendulum

The boundary conditions are at $t = 0, \theta_0 = \frac{\pi}{2} radians$ and at $t = T = 2.2 \sec onds, \theta_f = -\frac{\pi}{2} radians$. Time interval $\Delta t = 0.022 \sec onds$ and number of data points = 100.

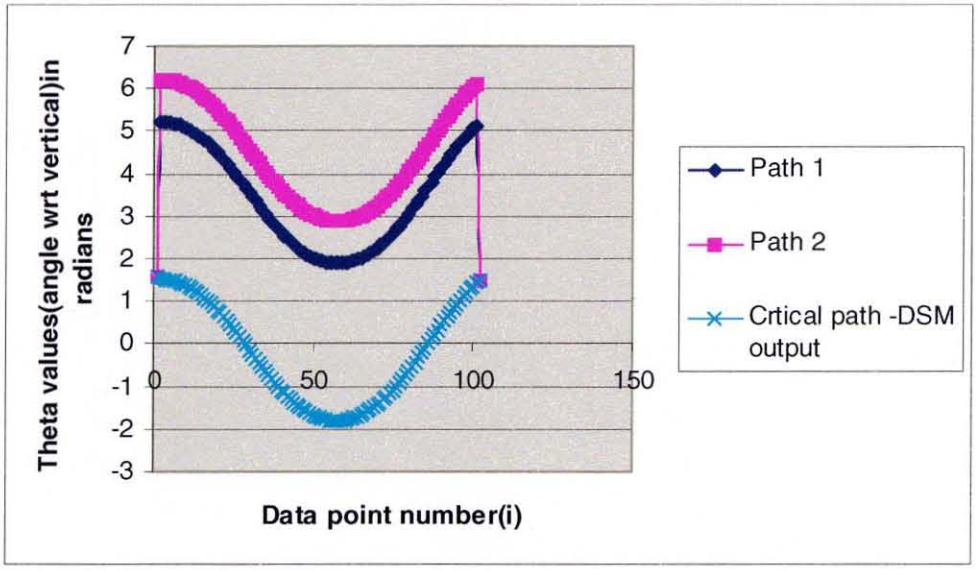


Figure 3.4 The graph shows the theta values (angle with respect to vertical) of the input paths and DSM output for the single pendulum model.

The above graph shows some of the input path that was generated by adding a random value to the guess path and the final output path given by simplex. All the paths have same initial and final position.

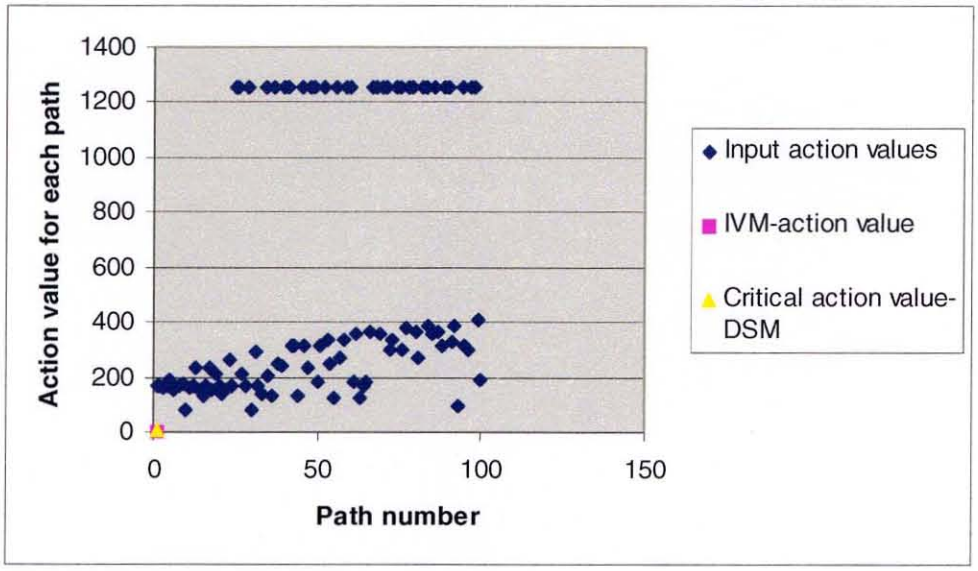


Figure 3.5 The graph shows the action values for the initial input paths (simplex vertices) and output paths generated by the DSM.

The above graph shows the action values for all input paths and the final action value generated by simplex. The final output path generated by simplex has the least action value.

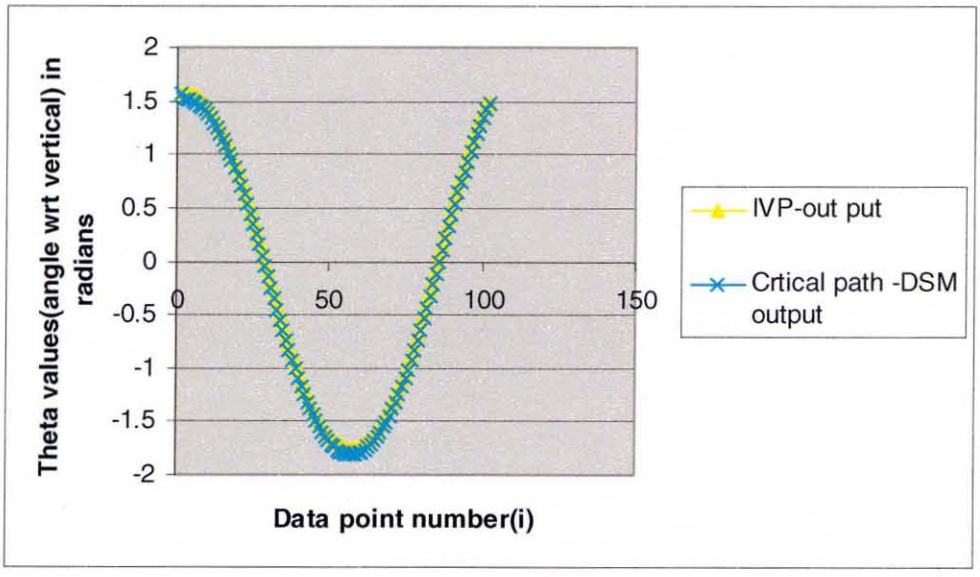


Figure 3.6 The graph shows the theta values (angle with respect to vertical) of the IVP numerical solution and DLA numerical solution using DSM.

The graph (Figure 3.6) shows the least action path generated by the DSM and numerical solutions of the corresponding Lagrange EOM solved using a first order Euler algorithm. Both solutions essentially superimpose on each other.

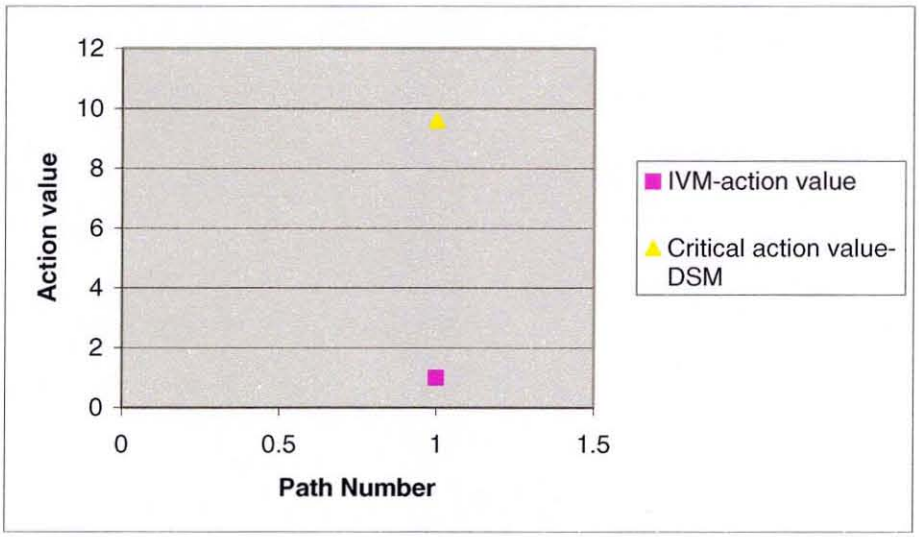


Figure 3.7 The graph shows the action values of the IVP numerical solution and DLA numerical solution using DSM.

The above graph shows the action value generated by simplex and numerical solution of the corresponding Lagrange EOM solved using first order Eulers algorithm.

3.4.2 Double Pendulum

Boundary conditions are $t = 0, \theta_i(0) = \frac{\pi}{2} radians, \theta_2(0) = \frac{\pi}{2} radians$ and at $t = T, \theta_i(0) = -0.6251 radians, \theta_2(0) = -1.149 radians$.

Time interval Δt=0.022 sec$onds$ and number of data points =100.

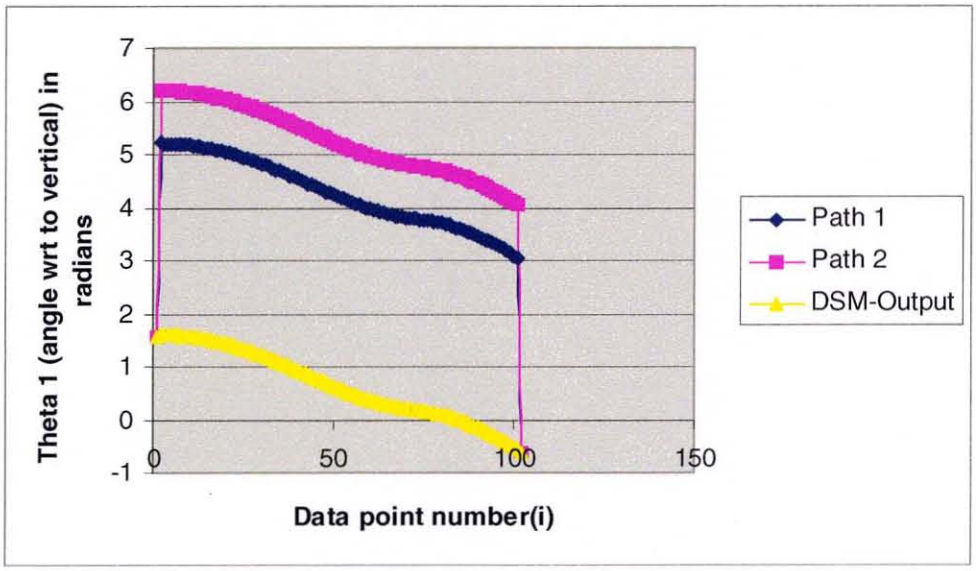


Figure 3.8 The graph shows the theta 1 values (angle with respect to vertical) of the input paths and DSM output for a double pendulum model.

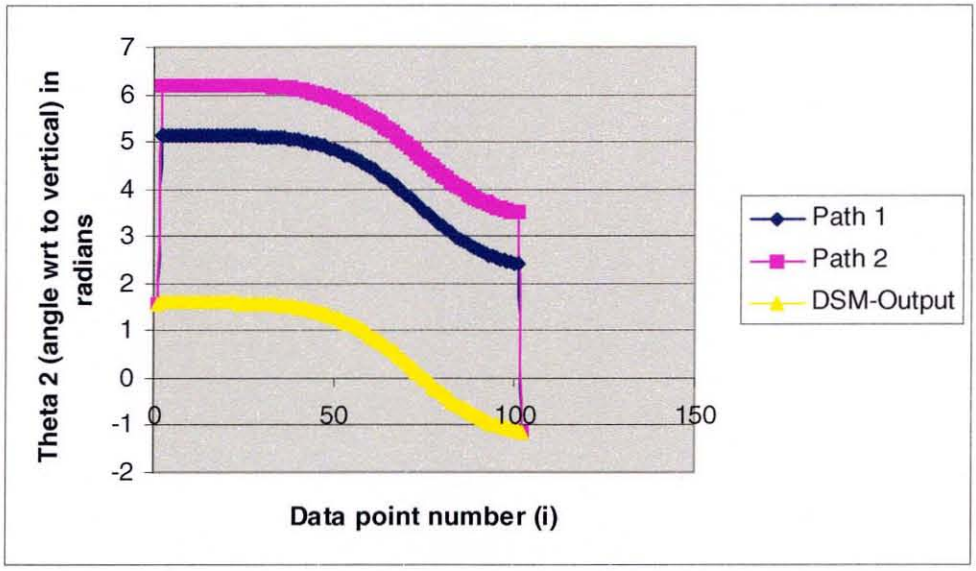


Figure 3.9 The graph shows the theta 2 values (angle with respect to vertical) of the input paths and DSM output.

The above graphs (Figure 3.8 and 3.9) show θ_1 *and* θ_2 values for some of the input paths generated by adding a random value to the guess path and the final output path given by simplex. All the input path and the final output path have same initial and final position.

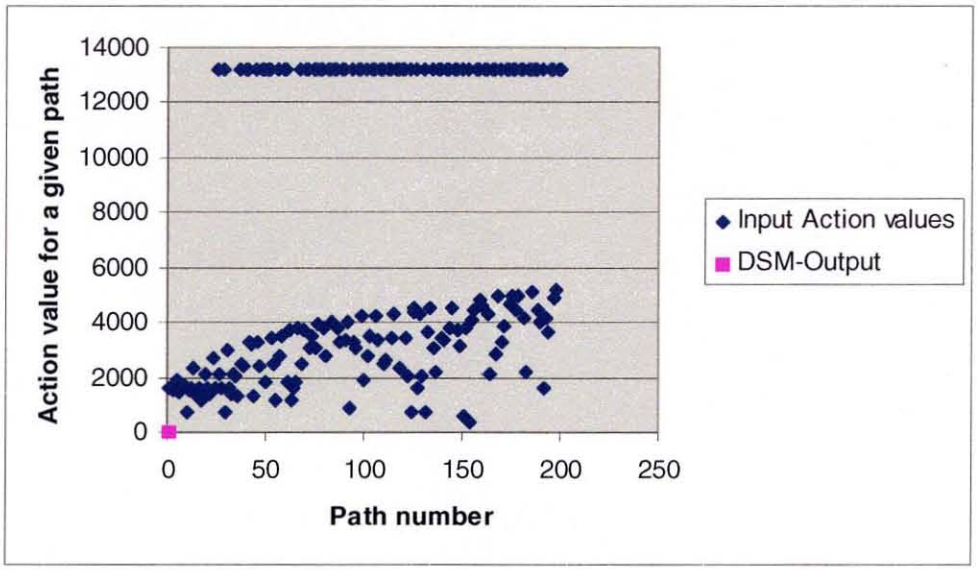


Figure 3.10 The graph shows the action values for input paths and DSM out put for a double pendulum model.

The above graph shows the action values for all input paths and the final action value generated by simplex. The out path generated by simplex has the least action value.

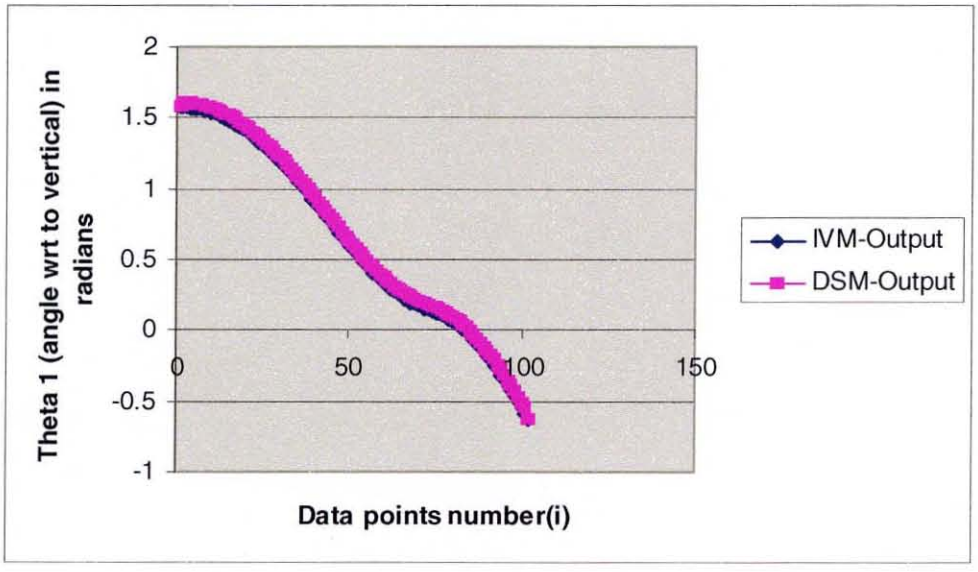


Figure 3.11 The graph shows the theta1 values of the IVP numerical solution and DLA numerical solution using DSM.

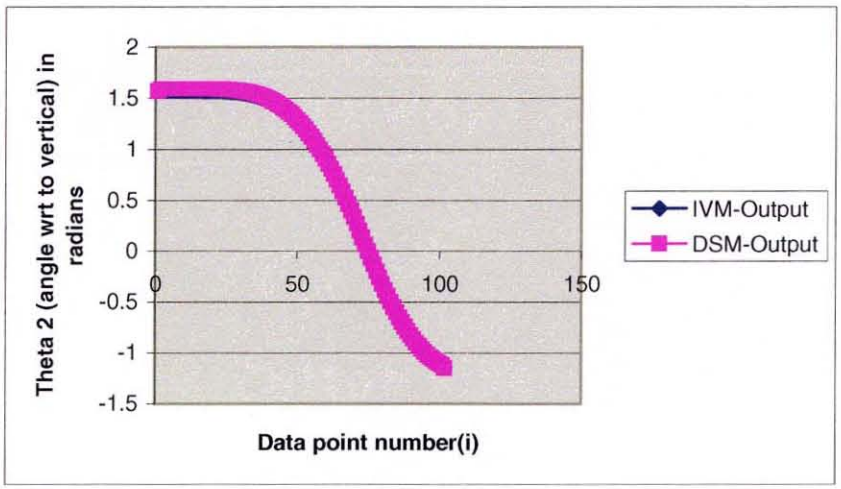


Figure 3.12 The graph shows the theta 2 values (angle with respect to vertical) of the IVP numerical solution and DLA numerical solution using DSM.

The above graphs (Figure 3.11 and 3.12) show the least action path generated by simplex and by solving EOM by Eulers algorithm using initial value method. Both the path superimposes.

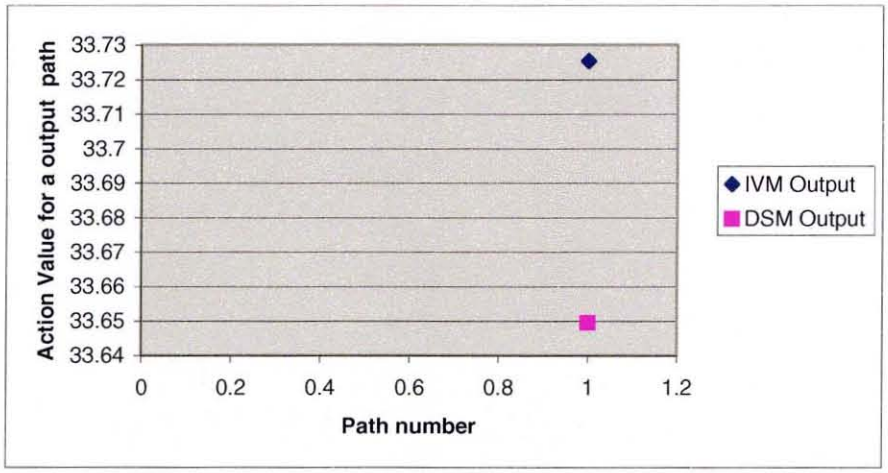


Figure 3.13 The graph shows the action values of the IVP numerical solution and DLA numerical solution using DSM.

The graph (Figure 3.13) shows the action value generated by simplex and numerical solution of the corresponding Lagrange EOM solved using first order Eulers algorithm.

3.5 Conclusion

The numerical feasibility of solving motion 2-point BVPs by direct numerical minimization of the action without using the EOM as an intermediate step has been tested. The results obtained from the proposed DLA approach using the DSM for both the single and double pendulum systems are compared to numerical solutions of the corresponding Lagrange EOM for the same mechanical problems and the results are shown in section 3.4.

The two solutions are relatively close but may not be exact. The numerical accuracy of the results in both methods of solution can be improved by decreasing the size of the numerical time increment Δt (increasing the number of points N used to approximate a path in the DLA). Higher order algorithms like Runga-Kutta (fourth order) to solve EOM can be used to yield more accurate results. Also more accurate and more efficient numerical multivariable minimization algorithms methods than DSM can be used to implement the DLA approach.

Future steps will be to apply DLA to more complicated multi branched pendulum systems that are used to model the human body. Eventually if numerical algorithms can be developed that make the DLA approach as efficient or more efficient than using Lagrange's differential EOM to solve 2-point BVPs for multi branched pendulum systems then such algorithms will be embedded in the software that has been developed in the human motion analysis and performance laboratory at NJIT to solve human motion problems by the boundary method.

APPENDIX A

C PROGRAM TOGENERATE LEAST ACTION PATH USING DOWNHILL SIMPLEX METHOD FOR DOUBLE PENDULUM MODEL

```
#include <time.h>
#include <fstream.h>
#include <string.h>
#include "nr.h"
#include "nrutil.h"
int main(void)
{
        //Initializing values
        int i,nfunc,j,no_of_points, MP, NP, count,count1 = 0;
        double *x,*y,**p,*input_points,*input_points_p2;

        // No of data point excluding initial & final points
        printf ("\nEnter the number of data points: ");
         scanf ("%d", &no_of_points);

        // Print  output
        FILE *output;
        output = fopen("output.txt","w");
        fprintf(output,"No of Points");
        fprintf(output,"\tNo of Eval");
        fprintf(output,"\t\tAction Value\n");

        //Initialize variables
        t=0;initial_time=0; NP = no_of_points; MP = no_of_points + 1;
        input_points=vector(1,no_of_points);
        input_points_p2 = vector(1,no_of_points)
        char word[100];
        char *myword = "void";
        char *pl;
        FILE *fptr; /* declare a FILE pointer  */

        /* open a text file for reading theta values*/
        fptr = fopen("inputvalues.txt", "r");
        if(fptr==NULL) {
        printf("Error: can't open file.\n");
        /* fclose(file); DON'T PASS A NULL POINTER TO fclose !! */
        return 1;
        }
        else {
        printf("File opened successfully. Contents:\n\n");
```

```
count1=1; count=0;
while(fscanf(fptr, "%s", word) > 0){
        input_points[count1] = strtod(word,&pl);
        count1++;
}
printf("\n\nNow closing file...\n");
fclose(fptr);
}

//initializing the space
x=vector(1,NP);
y=vector(1,MP);
p=matrix(1,MP,1,NP);

//Generating N+1 paths consisting of N data points
for (i=1;i<=MP;i++) {
srand((i*time(NULL))%i);
x_increment = rand();
x_increment = log(x_increment);

printf("random number is %2.8f\n", x_increment);
        for (j=1;j<=NP;j++)
        {
                if(i==j){
                        x[j] = p[i][j] = input_points[j]+x_increment;
                }
                else{
                        x[j] = p[i][j] = input_points[j]+x_increment;
                }
                if(i==MP){
                        x[j] = p[i][j] = input_points[j];
                }
        }
y[i]=func(x, NP);
}

//Printing initial values
FILE *abc;
abc = fopen("results.txt","w");
fprintf(abc,"Vertices of final 3-d simplex and\n");
fprintf(abc,"function values at the vertices:\n\n");
//printf("%3s %10s %12s %12s %14s\n\n","i","x[i]","y[i]","z[i]","function");

for (i=1;i<=MP;i++) {
        fprintf(abc,"%9d ",i);
        for (j=1;j<=NP;j++)
```

```
                        fprintf(abc,"%12.9f ",p[i][j]);
                        fprintf(abc,"%12.9f\n",y[i]);
        }
        fprintf(abc,"\n\n\n\n\n\n\n");

        //Calling the minimization function ameoba
        amoeba(p,y,NP,FTOL,func,&nfunc);

        // printing the final  values
        fprintf(abc,"\nNumber of function evaluations: %3d\n",nfunc);
        fprintf(abc,"Vertices of final 3-d simplex and\n");
        fprintf(abc,"function values at the vertices:\n\n");

        for (i=1;i<=MP;i++) {
                fprintf(abc,"%6d ",i);
                for (j=1;j<=NP;j++)
                fprintf(abc,"%12.6f ",p[i][j]);
                fprintf(abc,"%12.9f\n",y[i]);
    }

        fprintf(output,"%d",no_of_points);
        fprintf(output,"\t%3d",nfunc);
        fprintf(output,"\t\t%12.9f\n",y[i-1]);
        free_matrix(p,1,MP,1,NP);
        free_vector(y,1,MP);
        free_vector(x,1,NP);
        return 0;
}


// Sub Function "Action Calculator"

#include "nr.h"
#include "nrutil.h"

double func(double theta[], int no_of_pts)
{

        double pi_radians,theta_final,theta_initial,delta_time,Action_Sum,theta_finalp2;
        double *theta_dot,*theta_dot_p2,*KE,*PE,*L,*A;
        int NP, MP;

        NP = no_of_pts;
        MP = no_of_pts + 1;
        Action_Sum = 0.0;
        delta_time=0.001;
```

```
        //Allocating space
        theta_dot = vector(1,NP/2);
        theta_dot_p2 = vector(1,NP/2);
        KE = vector(1,NP/2);
        PE = vector(1,NP/2);
        L = vector(1,NP/2);
        A = vector(1,NP/2);

        //Calculating Velocity Component
        for(int incr=1; incr<=NP/2;incr++){

                if(incr==1){
                        theta_initial= 1.570796327;
                        theta_dot[incr] = (theta[incr+1] - theta_initial) / (2 * delta_time);
                        theta_dot_p2[incr] = (theta[NP/2+incr+1] - theta_initial) / (2 *
delta_time);
                }
                else if(incr==NP/2){
                        theta_final = 1.5212518;
                        theta_finalp2=1.570812;
                        theta_dot[incr] = (theta_final - theta[incr-1]) / (2 * delta_time);
                        theta_dot_p2[incr] = (theta_finalp2 - theta[(NP/2 + incr)-1]) / (2 *
delta_time);

                }else{
                        theta_dot[incr] = (theta[incr+1] - theta[incr-1])/( 2 * delta_time);
                        theta_dot_p2[incr] = (theta[NP/2+incr+1] - theta[NP/2+incr-
1])/(2*delta_time);
                }
      PE[incr] = -(2*9.81*1*cos(theta[incr])) - (1*9.81*1*cos(theta[NP/2+incr]));

      KE[incr] = (0.5*mass*length*length*theta_dot[incr]*theta_dot[incr]) +  (0.5*mass
* ( (length*length*theta_dot[incr]*theta_dot[incr]) +
(length*length*theta_dot_p2[incr]*theta_dot_p2[incr]) +
(2*length*length*theta_dot[incr]*theta_dot_p2[incr]*cos(theta[incr]-theta[NP/2+incr])) )
);

                L[incr] = KE[incr] - PE[incr];
                A[incr] = L[incr] * delta_time;
                Action_Sum = Action_Sum + A[incr];
        }

        return Action_Sum;
}
```

APPENDIX B

C PROGRAM TO SOLVE EOM USING INITIAL VALUE METHOD

```
#include <stdio.h>
#include <math.h>

// Main Program
int main(void)
{
        double theta1_initial, theta2_initial, velocity1_initial, velocity2_initial,
velocity_initial;
        double delta_t, gravity, length, length1, length2,m1,m2, pi_radians;
        double theta[102], velocity[102], velocity_dot[102], velocity_dot1[102],
velocity_dot2[102], theta1[102], theta2[102], velocity1[102], velocity2[102];
        double numerator11, numerator12, numerator22, numerator21, denominator11,
denominator21;

        delta_t = 0.001; velocity_initial = 0.0; velocity1_initial = 0;
        velocity2_initial = 0;
        pi_radians = 4 * atanf(1); theta1_initial=pi_radians/2; theta2_initial=pi_radians/2;
        gravity = 9.81; length1 = 1; length2 = 1; m1=m2=1;

        for (int count=0;count<=101;count++) {
                if (count==0){
                        theta1[count]=theta1_initial;
                        velocity1[count]=velocity1_initial;
                        theta2[count]=theta2_initial;
                        velocity2[count]=velocity2_initial;
                        velocity_dot1[count]=0;
                        velocity_dot2[count]=0;
                }
                else{
                        theta1[count]=theta1[count-1] + (velocity1[count-1] * delta_t);
                        numerator11 = (-(m1 + m2)*gravity*sin(theta1[count-1])) +
(length2*m2*sin(theta1[count-1]-theta2[count-1])*powf(velocity2[count-1],2));
                        denominator11 = length1 * ((m1+m2) - (m2 *
powf(cos(theta1[count-1]-theta2[count-1]),2)));
                        numerator12 = cos(theta1[count-1]-theta2[count-
1])*((m2*gravity*sin(theta2[count-1])) + (length1*m2*sin(theta1[count-1]-theta2[count-
1])*powf(velocity1[count-1],2)));

        velocity_dot1[count]=(numerator11+numerator12)/denominator11;
        velocity1[count] = velocity1[count-1] + velocity_dot1[count] * delta_t;
        theta2[count]=theta2[count-1] + (velocity2[count-1] * delta_t);
```

```
numerator21 = - cos(theta1[count-1]-theta2[count-1]) * (-
(m1+m2)*gravity*sin(theta1[count-1])+(length2*m2*sin(theta1[count-1]-theta2[count-
1])*powf(velocity2[count-1],2)));
numerator22 = (m1+m2)*(-gravity*sin(theta2[count-1])-(length1*sin(theta1[count-1]-
denominator21 = length2 * ((m1+m2) - (m2 * powf(cos(theta1[count-1]-theta2[count-
1]),2)));

theta2[count-1])*powf(velocity1[count-1],2)));
velocity_dot2[count]=(numerator21+numerator22)/denominator11;
        velocity2[count] = velocity2[count-1] + velocity_dot2[count] * delta_t;
        }
    }

        FILE *output;
        output = fopen("output.txt","w");
        for (count=0;count<=101;count++){
                fprintf(output,"%3.7f\t",theta1[count]);
                fprintf(output,"%3.7f\n",theta2[count]);
                //fprintf(output,"%3.7f\n",velocity[count]);
        }
        printf("Generated values. Please enter a character to exit");
        getchar();

        return 0;
}
```

APPENDIX C

C PROGRMA TO GENERATE LEAST ACTION PATH USING DSM FOR A SINGLE PENDULUM MODEL

```
#include <time.h>
#include "nr.h"
#include "nrutil.h"

int main(void)
{
        int i,nfunc,j,no_of_points = 0, MP,NP,loop,increment, count,count1;
        double *x,*y,**p,*input_points,x_increment=0,t;
        printf ("\nEnter the number of data points: ");
        scanf ("%d", &no_of_points);

        printf ("\nEnter the number of cycles: " );
        scanf ("%d", &loop);
        printf ("\nEnter the increment for the cycle: " );
        scanf ("%d", &increment);

        //Initialize variables
        t=0;initial_time=0; NP = no_of_points; MP = no_of_points + 1;
        input_points=vector(1,no_of_points);

        char word[100]; char *pl;
        char *myword = "void";
        FILE *fptr; /* declare a FILE pointer  */

        /* open a text file for reading input theta values*/
        fptr = fopen("input.txt", "r");
        if (fptr==NULL) {
        printf("Error: can't open file.\n");
        /* fclose(file); DON'T PASS A NULL POINTER TO fclose !! */
        return 1;
        }
        else {
        printf("File opened successfully. Contents:\n\n");
        count1=1; count=0;
        while(fscanf(fptr, "%s", word) > 0) {
                input_points[count1] = strtod(word,&pl);
                count1++;
        }
        printf("\n\nNow closing file...\n");
        fclose(fptr);
```

```
}
//initializing the space
x=vector(1,NP);
y=vector(1,MP);
p=matrix(1,MP,1,NP);

//Inputing the different coordinates for starting simplex
for (i=1;i<=MP;i++) {
srand((i*time(NULL))%i);
x_increment = rand(); x_increment = log(x_increment);
printf("random number is %2.8f\n", x_increment);
        for (j=1;j<=NP;j++) {
                if(i==j){
                        x[j] = p[i][j] = input_points[j]+x_increment;
                }
                else{
                        x[j] = p[i][j] = input_points[j]+x_increment;
                }
                if(i==MP) {
                        x[j] = p[i][j] = input_points[j];
                }
        }
y[i]=func(x, no_of_points);
}
//printing initial values
FILE *abc;
abc = fopen("results.txt","w");
fprintf(abc,"Vertices of final 3-d simplex and\n");
fprintf(abc,"function values at the vertices:\n\n");

for (i=1;i<=MP;i++) {
        fprintf(abc,"%6d ",i);
        for (j=1;j<=NP;j++)
        fprintf(abc,"%12.6f ",p[i][j]);
        fprintf(abc,"%12.9f\n",y[i]);
}
fprintf(abc,"\n\n\n\n\n\n");

//calling the minimization function ameoba
amoeba(p,y,NP,FTOL,func,&nfunc);

// Printing the final values
fprintf(abc,"\nNumber of function evaluations: %3d\n",nfunc);
fprintf(abc,"Vertices of final 3-d simplex and\n");
fprintf(abc,"function values at the vertices:\n\n");
```

```
for (i=1;i<=MP;i++) {
                fprintf(abc,"%6d ",i);
                for (j=1;j<=NP;j++)
                fprintf(abc,"%12.6f ",p[i][j]);
                fprintf(abc,"%12.9f\n",y[i]);    }
        no_of_points = no_of_points + increment;
        free_matrix(p,1,MP,1,NP);               free_vector(y,1,MP);
        free_vector(x,1,NP);
        return 0;
}

// Action calculator
#include "nr.h"
#include "nrutil.h"
#define MIDDLE 2
double func(double theta[], int no_of_pts)
{
        Doubletime_final,pi_radians,time_initial,theta_final,theta_initial,delta_time,
        Action_Sum, *theta_dot,*KE,*PE,*L,*A,theta_max;
        int NP, MP;  NP = no_of_pts; MP = no_of_pts + 1;  delta_time =0.022;
        //Allocating space
        theta_dot = vector(1,NP);       KE = vector(1,NP);
        PE = vector(1,NP);              L = vector(1,NP);
        A = vector(1,NP);
        for(int incr=1; incr<=NP;incr++){
                if(method == MIDDLE){
                        if(incr==1){
                                theta_initial= 1.570796327;
                        theta_dot[incr] = (theta[incr+1] -1.570796327)/ (2 * delta_time);
                        }
                        else if(incr==NP){
                        theta_dot[incr] = (1.4799464- theta[incr-1]) / (2 * delta_time);
                        }else{
                        theta_dot[incr] = (theta[incr+1]-theta[incr-1])/( 2 * delta_time);
                        }
                }
                else{ }
        PE[incr] =mass*gravity*length*(1- theta[incr]);
        KE[incr] = 0.5* mass*theta_dot[incr]*theta_dot[incr];
        L[incr] = KE[incr] - PE[incr];
        A[incr] = L[incr] * delta_time;
        Action_Sum = Action_Sum + A[incr];
        }
        return Action_Sum;
}
```

APPENDIX D

C PROGRAM TO SOLVE EOM USING INITIAL VALUE METHOD FOR A SINGLE PENDULUM SYSTEM

```
#include <stdio.h>
#include <math.h>

// Main Program

int main(void)
{

        double delta_t, velocity_initial, theta_initial, gravity, length, pi_radians;
        double theta[102], velocity[102];
        delta_t = 0.022;
        velocity_initial = 0.0;
        pi_radians = 4 * atanf(1);
        theta_initial = pi_radians/2;
        gravity = 9.81;
        length = 1;

        for (int count=0;count<=101;count++) {
                if (count==0){
        theta[count]=theta_initial;
        velocity[count]=velocity_initial;
                }
                else{
        theta[count]=theta[count-1] + (velocity[count-1] * delta_t);
velocity[count] = velocity[count-1] - (gravity * sin(theta[count-1]) * delta_t /length) ;

                }
                                                        }
                FILE *output;
        output = fopen("output.txt","w");
        for (count=0;count<100;count++){
                fprintf(output,"%3.7f\n",theta[count]);
                //fprintf(output,"%3.7f\n",velocity[count]);
                                                }
        printf("Generated values. Please enter a character to exit");
        getchar();

        return 0;
}
```

REFERENCES

1. Kane, T.R., Levinson, D.A. and David, A. Formulation of Equation of Motion for a Complex Spacecraft: J of Guidance and Control. Article No.80-4014 pp.103-105, 1980.

2. Teukolsky, W. H., Vetterling, S.A. and Flannery, B. P. Numerical Recipes in C: The Art of Scientific Computing: Cambridge University Press: Ch. 10, pp. 394-397, 1993.

3. Feynman,R., Sands. and Leighton. The Feynman Lectures on Physics: Addison Wesley Publications Company, NY. Ch. 19, pp. 1-14, 1964.