20th ECMI Modelling Week
Final Report

Kongens Lyngby, Copenhagen, Denmark
2006.08.14 — 2006.08.24

Team 5

Mathematical models for a fishing rod and the action of casting with a bait

Tefa Kaisara
Johannes Kepler University, Linz, Austria
Henry Kasumba
TU Eindhoven, The Netherlands.
Konsantin Lofink
TU Kaiserslautern,Germany.
Joanna Sylwia Pelc
TU Zielona Gora, Poland
Yayun Zhou
TU Eindhoven, The Netherlands
Ute Ziegler
Kaiserslautern, Germany
Viðar Hrafnkelsson
TU Denmark, Denmark

Bjarte Hægland
NTNU, Trondheim, Norway

Contents

List of Figures

Preface

The 20th ECMI modelling week entitled European Student Workshop on Mathematical Modelling in Industry and Commerce was held in Kongens Lyngby, Copenhagen, Denmark, August 14-24, 2006, at the Technical University of Denmark

Students from different universities were split into groups and is this the group instructed by Bjarte Hægland from the Norwegian University of Science and Technology in Trondheim. The course is valued at 5 ECST points.

Abstract

The aim is to create a mathematical models for a fishing rod and the action of casting with a bait. A static fishing rod is modeled and the model is then transformed into a system of differential equations which is then solved numerically. For the motion of the fishing rod, a dynamic equation is formed where the position of each part of the fishing rod, which is bent, is found as function of time. Then the optimal casting angle is found. Another approach in solving the problem is then proposed for future work.

1 Introduction

When fishing with a fishing rod it is sometimes desirable to be able to cast the bait long distances. This enables the fisher to cover a greater area of the water and also lets the fisher target fishing spots otherwise difficult to reach. The aim is to create mathematical models for a fishing rod and the action of casting with a bait. The model will be simplified by assuming the rod moves in a vertical plane and try to find a technique which increases the travel distance of the bait when it is released by the fisher.

Equipment: Bait casting equipment consists of four pieces: a rod, a reel, a line and a lure. The rod is made out of tubular or solid glass or sometimes bamboo or metal and is from 1.5m to 2.5m long. It is classified by action (weight) as medium, light or very light. The reel houses a spool operated with a right-hand crank, which turns the spool, winding the lane. Reel capacity for holding line varies by brand, size and price. The line is available in a variety of breaking strengths, from 200 Pa to 1000 Pa. In our case, the fishing rod is made by carbon fibre. The length of the fishing rod is 2.5m.

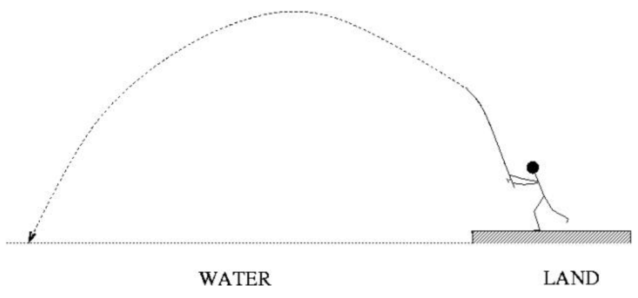

Figure 5.1: Bait casting rod.

2 Approach I

2.1 Model of static fishing rod

In order to find out the mathematical model for a fishing rod and the action of casting with a bait, the task is divided into two steps. The first step is to model a static fishing rod held by a fisherman and the second step is to model the action of casting with a bait. In the first step, the static fishing rod is considered as a elastic beam with one end fixed. The beam will bend because the effect of the weight of bait. Figure [5.2] shows the phenomenon described above.

To simplify the problem, the fishing rod is assumed weightless and the bending is only caused by the weight of bait. Then the handle is ignored and the whole fishing rod is considered as a tube with varied cross section. Figure 5.3 is the simplification of figure 5.2, from which the mathematical variables describing the physical phenomenon are defined.

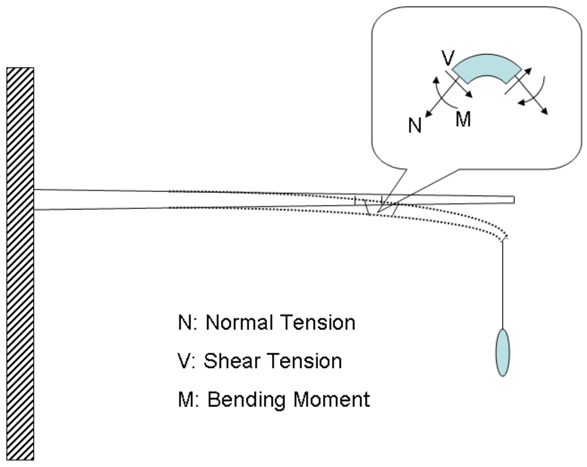

Figure 5.2: Bending of the Static Fishing Rod

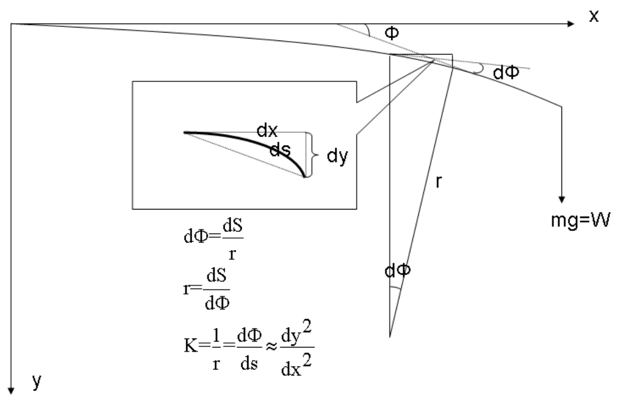

Figure 5.3: Simplification of the Bending Fishing Rod

Now focusing on a small part of the bending fishing rod and analyzing the force acting on it, figure [5.4] is aquired as shown below. According to this picture, there are two types of forces and one type

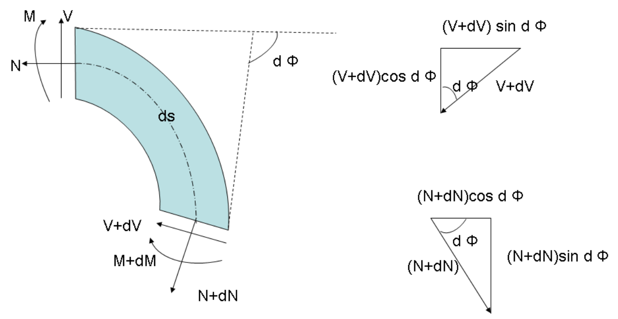

Figure 5.4: Analysis of the Fishing Rod

of moment acting on the small region. Since the fishing rod is static, all those forces and moments should be in balance. By decomposing the forces into x and y direction, the below equations are derived .

$$(N + dN)\cos d\Phi - N - (v + dv)\sin d\Phi = 0 \tag{5.1}$$

$$(V + dV)\cos d\Phi - V + (N + dN)\sin d\Phi = 0 \tag{5.2}$$

$$M + dM - M + (V + dV)ds - V + (N + dN)ds \sin d\Phi = 0 \tag{5.3}$$

Since $d\Phi \ll 1$, it is approximated that $\cos d\Phi \approx 1$ and $\sin d\Phi \approx d\Phi$, which leads to three new equations.

$$dN - (v + dv)d\Phi = 0 \tag{5.4}$$

$$dV + (N + dN)d\Phi = 0 \tag{5.5}$$

$$dM + dVs + (N + dN)dsd\Phi = 0 \tag{5.6}$$

Because $ds \to 0$ and $d\Phi, dN, dT \to 0$, by ignoring the second order term, the new equations are demonstrated below:

$$\frac{dN}{d\Phi} = V \tag{5.7}$$

$$\frac{dV}{d\Phi} = -N \tag{5.8}$$

$$dM + dVs = 0 \tag{5.9}$$

From the equation 5.7 and equation 5.8, the relationship between N and V is derived:

$$\frac{dN}{dV} = -\frac{V}{N} \Rightarrow N^2 + V^2 = C \tag{5.10}$$

In order to find out the constant C, we look at the figure 5.5:

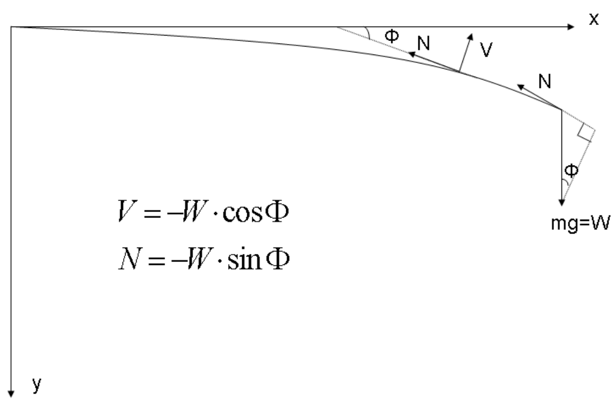

Figure 5.5: Relationship between Forces

From that figure, the relation $N^2 + V^2 = W^2$ can be seen.
Now considering the bending moment acting on the fishing rod, according to the definition of first moment of area about the z direction, the following integral is derived:

$$I_z = \int z^2 dA = I_y = \frac{1}{2}\int r^2 dA = \frac{1}{2}\int_0^{2\pi}\int_r 1^r 2r^3 dr d\theta = \frac{\pi}{4}(r_2^4 - r_1^4) \tag{5.11}$$

From figure of relationship for moment, the equation is derived:

$$d\Phi = \frac{ds}{r} = \frac{(1+\varepsilon)ds}{y+r} \tag{5.12}$$

$$\varepsilon = \frac{y}{r} = yK \tag{5.13}$$

$$\sigma = E\varepsilon \tag{5.14}$$

Where E is Young's module.
Inserting the equation [5.13] to the equation [**??**] yields:

$$\sigma = -\frac{Ey}{r} = EyK\varepsilon \tag{5.15}$$

Since $N = \int_A \sigma dA$ and $M = y \cdot N$ Now the moment is found:

$$M = \int_A y\sigma dA = \frac{E}{r}\int_A y^2 dA = \frac{E}{r}I_z = EI_z\frac{1}{r} = EI_z\frac{d\phi}{ds} \tag{5.16}$$

In all, the mathematical model is derived:

$$\frac{d}{ds}(EI_z(s)\frac{d\phi}{ds}) = W \cdot \cos(\phi) \tag{5.17}$$

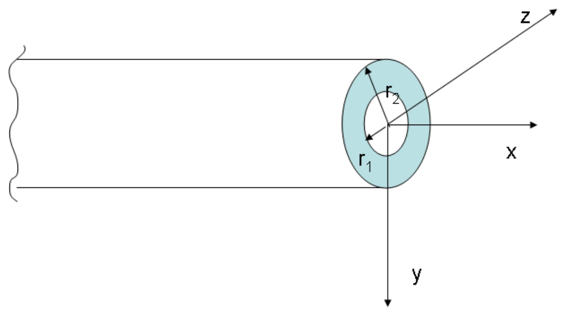

Figure 5.6: Cross Section of the Fishing Rod

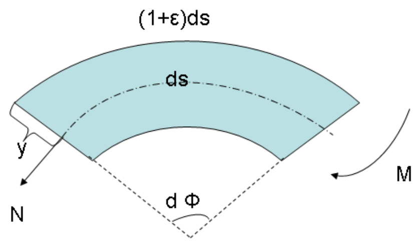

Figure 5.7: Relationship for Moment

2.2 Solution for the Model of Static Fishing Rod

In the last section, the mathematical model for the static fishing rod was derived:

$$\frac{d}{ds}(EI_z(s)\frac{d\phi}{ds}) = w \cdot \cos(\phi) \tag{5.18}$$

where

$$I_z(s) = \frac{\pi}{4}(r_2^4(s) - r_1^4(s)) \tag{5.19}$$

First the left side of equation 5.18 is expanded.

$$E\frac{d}{ds}\left[\frac{\pi}{4}(r_2^4(s) - r_1^4(s))\frac{d\phi}{ds}\right] = w \cdot \cos(\phi) \tag{5.20}$$

$$\frac{\pi E}{4}\left[r_2^4(s) - r_1^4(s)\right]\frac{d^2\phi}{ds^2} + \pi E\left[r_2^3(s)\frac{dr_2}{ds} - r_1^3(s)\frac{dr_1}{ds}\right]\frac{d\phi}{ds} = w \cdot \cos(\phi) \tag{5.21}$$

$$\frac{\pi E}{4w}\left[r_2^4(s) - r_1^4(s)\right]\frac{d^2\phi}{ds^2} + \frac{\pi E}{w}\left[r_2^3(s)\frac{dr_2}{ds} - r_1^3(s)\frac{dr_1}{ds}\right]\frac{d\phi}{ds} = \cos(\phi) \tag{5.22}$$

The the inner and outer radii for the fishing rod are approximated to be linear functions of s.

$$r_1(s) = a_1 s + b_1 \tag{5.23}$$

$$r_2(s) = a_2 s + b_2 \tag{5.24}$$

2.3 Normalization of the equation

Since equation 5.22 has many components of different units, the equation is normalized. First variable along the rod is normalized by introducing a new variable s' which is on the interval [0,1]:

$$s = R \cdot s' \tag{5.25}$$

where $R = 2.1m$ is the length of the rod. Now the following equations are derived:

$$\begin{aligned} r_2(s) &= R \cdot r_2'(s') \\ r_1(s) &= R \cdot r_1'(s') \end{aligned} \tag{5.26}$$

so

$$r_2'(s') = \frac{1}{R}(Rs') = \frac{1}{R}(a_2 Rs' + b_2) \tag{5.27}$$

and therefore the normalized radii functions are derived.

$$\begin{aligned} r_2'(s') &= \frac{r_2(s)}{R} \\ r_1'(s') &= \frac{r_1(s)}{R} \end{aligned} \tag{5.28}$$

and differentiated:

$$\frac{dr_2'}{ds'} = \frac{1}{R}\frac{d}{ds'}r_2'(Rs') = \frac{1}{R}r_2'(Rs) \cdot R \tag{5.29}$$

Therefore:

$$\begin{aligned} \frac{dr_2'}{ds'} &= \frac{dr_2}{ds} \\ \frac{dr_1'}{ds'} &= \frac{dr_1}{ds} \end{aligned} \tag{5.30}$$

This means that the derivatives are dimensionless. Now the normalized functions for the radii are set in the equation 5.22.

$$\frac{\pi E}{4w}\left[(Rr_2'(s'))^4 - (Rr_1'(s'))^4\right]\frac{1}{R^2}\frac{d^2\phi}{ds'^2} + \frac{\pi E}{w}\left[(Rr_2'(s'))^3\frac{dr_2'}{ds'} - (Rr_1'(s'))^3\frac{dr_1'}{ds'}\right]\frac{1}{R}\frac{d\phi}{ds'} = \cos(\phi) \tag{5.31}$$

$$\frac{\pi ER^2}{4w}\left[(a_2 s' + \frac{b_2}{R})^4 - (a_1 s' + \frac{b_1}{R})^4\right]\frac{d^2\phi}{ds'^2} + \frac{\pi ER^2}{w}\left[(a_2 s' + \frac{b_2}{R})^3 a_2 - (a_1 s' + \frac{b_1}{R})^3 a_1\right]\frac{d\phi}{ds'} = \cos(\phi) \tag{5.32}$$

Then the constant γ is defined:

$$\gamma = \frac{\pi ER^2}{w} \tag{5.33}$$

so the scale of γ is

$$[\gamma] = \frac{N}{m^2}\frac{m^2}{N} = 1 \tag{5.34}$$

More constants are also define:

$$\begin{aligned} \alpha_1 = a_1 \ ; \ \beta_1 = \frac{b_1}{R} \\ \alpha_1 = a_2 \ ; \ \beta_2 = \frac{b_1}{R} \end{aligned} \tag{5.35}$$

Then the equation is dimensionless:

$$\frac{\gamma}{4}\left[(\alpha_2 s' + \beta_2)^4 - (\alpha_1 s' + \beta_1)^4\right]\frac{d^2\phi}{ds'^2} + \gamma\left[(\alpha_2 s' + \beta_2)^3\alpha_2 - (\alpha_1 s' + \beta_1)^3\alpha_1\right]\frac{d\phi}{ds'} = \cos(\phi) \tag{5.36}$$

The equation is simplified by introducing function $A(s')$ and $B(s')$:

$$\begin{aligned} A(s') &:= \frac{\gamma}{4}\left[(\alpha_2 s' + \beta_2)^4 - (\alpha_1 s' + \beta_1)^4\right] \\ B(s') &:= \gamma\left[(\alpha_2 s' + \beta_2)^3\alpha_2 - (\alpha_1 s' + \beta_1)^3\alpha_1\right] \end{aligned} \tag{5.37}$$

so the equation becomes:

$$A\frac{d^2\phi}{ds'^2} + B\frac{d\phi}{ds'} = \cos(\phi) \tag{5.38}$$

2.4 Solving the equation numerically

First the equation is transformed into a system of ODE's by defining:

$$\begin{aligned} u_1 &:= \phi \\ u_2 &:= \frac{d\phi}{ds} \end{aligned} \tag{5.39}$$

and then the system becomes:

$$\begin{bmatrix} 1 & 0 \\ 0 & A \end{bmatrix}\begin{bmatrix} u_1' \\ u_2' \end{bmatrix} = \begin{bmatrix} u_2 \\ cos(u_1) - Bu_2 \end{bmatrix} \tag{5.40}$$

The paramaters for this system are γ,the α's and the β's which are all dimensionless.

Now this system of ODE's is solved in Matlab using an built-in ODE-solver and ϕ for every part of the fishing rod is acquired. Figure 2.4 shows a comparison of the measured form of the fishing rod and the one acquired by solving the system of ODE's.

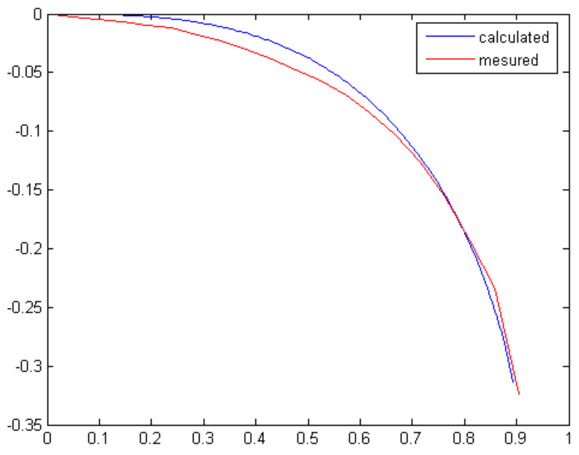

Figure 5.8: Comparision between the measured and computed bending of the rod

Since the Young's Module, E was only given to be at some interval, it had be optimized to make the calculated form of the fishing rod the most like the measured form of the fishing rod. The E derived was $E = 2.5 \cdot 10^9$ while the radii functions were set to:

$$r_1(s) = -\frac{1}{1000}s + \frac{3}{400} \tag{5.41}$$

$$r_2(s) = -\frac{1}{1000}s + \frac{19}{2000} \tag{5.42}$$

2.5 Description of Movement

In this section the movement of the fishing rod with a bait attached to it's tip will be discussed.

Just as it is impossible to pull a car with a slack rope, it is impossible to move a fly with a slack line. Thus we make sure that we start with the line straight. Every casting stroke is a smooth acceleration followed by a stop. The acceleration bends the rod and loads it like a catapult. While it is accelerating the bend increases, when it stops the rod recovers and straightens. It is the stop that transfers the stored energy in the catapult (rod) to the line and makes the cast. The line always follows the rod tip and when the rod stops the line projects in the direction that the rod tip is going in when the stop is made.

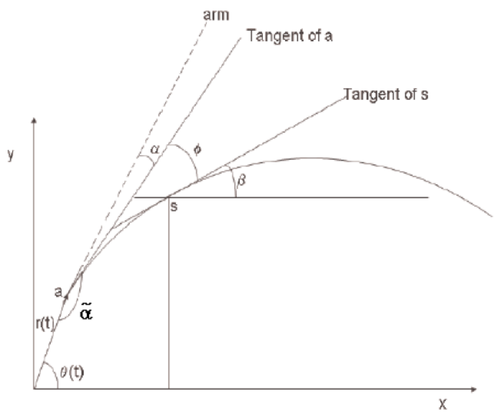

Figure 5.9: motion of the arm with the rod

The expressions for the displacement of the rod at a time t are given by the equations

$$x(s,t) = r(t)cos(\theta(t)) + \int_0^s \cos(\beta(s,t))ds, \tag{5.43a}$$

$$y(s,t) = r(t)sin(\theta(t)) + \int_0^s \sin(\beta(s,t))ds \tag{5.43b}$$

where

$$x_o(t) = r(t)cos(\theta(t)) \text{ and } y_o(t) = r(t)sin(\theta(t)).$$

$r(t)$ is the distance between the fisher's fist and the toes.
$\theta(t)$ is the angle between this line and the horizontal.
$\alpha(t)$ is the angle between and $r(t)$
Taking the second derivative of equations [5.43a] and [5.43b] with respect to t gives

$$\frac{\partial^2 x(L,t)}{\partial t^2} = f_x - \int_0^L \cos(\beta(s,t)).\left(\frac{\partial\beta(s,t)}{\partial t}\right)^2 + \sin(\beta(s,t)).\left(\frac{\partial^2\beta(s,t)}{\partial t^2}\right) ds \tag{5.44}$$

and

$$\frac{\partial^2 y(L,t)}{\partial t^2} = f_y + \int_0^L \cos(\beta(s,t)).\left(\frac{\partial^2\beta(s,t)}{\partial t^2}\right) - \sin(\beta(s,t)).\left(\frac{\partial\beta(s,t)}{\partial t}\right)^2 ds \tag{5.45}$$

where

$$f_x = \left(\frac{\partial^2 r(t)}{\partial t^2}\right)\cos(\theta(t)) - 2\left(\frac{\partial r(t)}{\partial t}\right)\sin(\theta(t)) - r(t)\cos(\theta(t))\left(\frac{\partial\theta}{\partial t}\right)^2 - r(t)\sin(\theta(t))\left(\frac{\partial^2\theta}{\partial t^2}\right)$$

and

$$f_y = \left(\frac{\partial^2 r(t)}{\partial t^2}\right)\sin(\theta(t)) + 2\left(\frac{\partial r(t)}{\partial t}\right)\cos(\theta(t)) - r(t)\sin(\theta(t))\left(\frac{\partial\theta}{\partial t}\right)^2 + r(t)\cos(\theta(t))\left(\frac{\partial^2\theta}{\partial t^2}\right)$$

The rod is assumed weightless and the only force that is acting on the bait is at the tip of the rod. This force is given by newton's second law of motion i.e

$$F = ma \tag{5.46}$$

where

$$a = \begin{pmatrix} \ddot{x}(L,t) \\ \ddot{y}(L,t) \end{pmatrix}$$

The momentum of the particle is given by

$$M(s,t) = \int_L^s F.n(\sigma)d\sigma \tag{5.47}$$

also from equation [5.16] momentum is given by

$$M(s,t) = EI_z(s)K \tag{5.48}$$

equating equations [5.47] and equation [5.48] ,we obtain

$$EI_z(s)K = \int_L^s F.n(\sigma)d\sigma \tag{5.49}$$

But from Fig [5.9], it can be seen that

$$\beta(s,t) = \theta(t) - \alpha(t) - \phi(s,t) \tag{5.50}$$

differentiating equation [5.50] with respect to s leads to

$$K = \frac{d\phi}{ds} = -\frac{d\beta}{ds} \tag{5.51}$$

subsituting the expression for K in [5.49] leads to

$$EI_z(s)(-\frac{d\beta}{ds}) = \int_L^s F.n(\sigma)d\sigma \tag{5.52}$$

the normal vector n to the rod is given by

$$n(s,t) = \begin{pmatrix} -\sin(\beta(s,t)) \\ \cos(\beta(s,t)) \end{pmatrix} \tag{5.53}$$

Therefore, substituting [5.46], and [5.53] into [5.52], we obtain

$$\begin{aligned} -EI_z(s)\frac{d\beta(s,t)}{ds} &= m.\int_L^s \left[(-\sin\beta)f_x(t) + \sin\beta\int_0^L [\sin\beta\left(\frac{\partial^2\beta}{\partial t^2}\right) + \cos\beta\left(\frac{\partial\beta}{\partial t}\right)^2]ds\right]d\sigma \\ &+ m\int_L^s \left[(\cos\beta)f_y(t) + \cos\beta\int_0^L [-\sin\beta\left(\frac{\partial\beta}{\partial t}\right)^2 + \cos\beta\left(\frac{\partial^2\beta}{\partial t^2}\right)]ds\right]d\sigma \end{aligned} \tag{5.54}$$

Therefore re-writing equation [5.54] leads to

$$\begin{aligned} -EI_z(s)\frac{d\beta(s,t)}{ds} &= m\int_s^L \left[(\sin\beta(\sigma,t))f_x(t) - \sin\beta(\sigma,t)\int_0^L [\sin\beta\left(\frac{\partial^2\beta}{\partial t^2}\right) + \cos\beta\left(\frac{\partial\beta}{\partial t}\right)^2]ds\right] d\sigma \\ &- m\int_s^L \left[(\cos\beta(\sigma,t))f_y(t) + \cos\beta(\sigma,t)\int_0^L [-\sin\beta\left(\frac{\partial\beta}{\partial t}\right)^2 + \cos\beta\left(\frac{\partial^2\beta}{\partial t^2}\right)]ds\right] d\sigma \end{aligned} \tag{5.55}$$

Equation [5.55] is our model equation for the movement of a fishing rod. This equation will be solved numerically in the next section to obtain value of β . The value of beta obtained can then be substituted back into equations [5.43a] and [5.43b] to obtain the displacement of the rod. Equations [5.43a] and [5.43b] are then differentiated to obtain the velocity of the tip of the rod. This velocity will then be regarded as the initial velocity of the bait as it leaves the tip. In the next subsection, the equations of motion of the bait as it leaves the tip of the rod is discussed.

Motion of the bait from the tip of the fishing rod

In this section,the motion of the fishing bait as it leaves the tip of the rod will be discussed. The bait's trajectory is assumed as parabolic when it leaves the tip. The bait that is projected from the tip of the fishing rod with initial velocity v_0 follows a parabolic path as shown in the figure [5.10]. The x and y coordinates of the particle at a time t are given by the equations

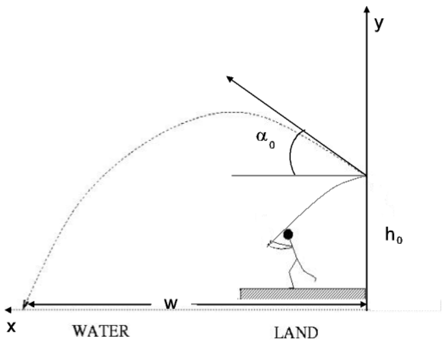

Figure 5.10: motion of a bait

$$\begin{aligned} x &= v_0 t\cos\alpha_0 && (5.56) \\ y &= h_0 + v_o t\sin\alpha_0 - \frac{1}{2}gt^2 && (5.57) \end{aligned}$$

where $v_0 = \begin{bmatrix} \dot{x}(L,t) \\ \dot{y}(L,t) \end{bmatrix}$ and α_0 is the initial angle of projection.
The time taken by the bait to land at any given point on a horizontal plane is given by

$$T = \frac{v_0\sin\alpha_0 \pm \sqrt{v_0^2\sin^2\alpha_0 + 2gh_0}}{g}. \tag{5.58}$$

The horizontal distance w the bait travels is given by

$$\begin{aligned} w &= v_0 \cos\alpha_0 T && (5.59) \\ &= v_0 \cos\alpha_0 \left(\frac{v_0 \sin\alpha_0 + \sqrt{v_0^2 \sin^2\alpha_0 + 2gh_0}}{g} \right) && (5.60) \end{aligned}$$

2.6 Numerical Analysis of the Dynamic equation for β

In this section the numerical solution of equation [5.55] is discussed. This equation is a non-linear partial integral differential equation. A closed solution to this equation is complicated, thus we seek a numerical solution to this equation.

Normalization of the model equation

Since equation [5.55] has many components of different units, the equation is normalized.
At first it is simplified to:

$$\begin{aligned} -EI_z(s)\frac{d\beta(s,t)}{ds} = m\int_s^L \Bigg[sin\beta(\sigma,t)f_x(t) - sin\beta(\sigma,t)\int_0^L \Bigg(sin\beta(s,t)\left(\frac{d^2\beta(s,t)}{dt^2}\right) + cos\beta(s,t)\left(\frac{d\beta(s,t)}{dt}\right)^2 \Bigg) \\ -m\int_s^L \Bigg[cos\beta(\sigma,t)f_x(t) + cos\beta(\sigma,t)\int_0^L \Bigg(cos\beta(s,t)\left(\frac{d^2\beta(s,t)}{dt^2}\right) - sin\beta(s,t)\left(\frac{d\beta(s,t)}{dt}\right)^2 \Bigg) \end{aligned} \quad (5.61)$$

To normalize the equation the following dimensionless variables are introduced:

$$\begin{aligned} t' &= Tt \\ s' &= Ls \\ \sigma' &= L\sigma \end{aligned} \quad (5.62)$$

Then the following equations are derived:

$$\begin{aligned} I_z'(s') &= I_m \cdot I_z(s) \\ I_m &= I_z'(0) \end{aligned} \quad (5.63)$$

and a dimensionless constant:

$$\gamma = \frac{mL^3}{I_m E T^2} \quad (5.64)$$

Now a normalized function F of β is defined:

$$F(\beta) = I_z(s)\frac{d\beta(s,t)}{ds}$$
$$+\gamma \int_s^1 \left[sin\beta(\sigma,t) f_x(t) - sin\beta(\sigma,t) \int_0^1 \left(sin\beta(s,t) \left(\frac{d^2\beta(s,t)}{dt^2}\right) + cos\beta(s,t) \left(\frac{d\beta(s,t)}{dt}\right)^2 \right) ds \right] d\sigma$$
$$-\gamma \int_s^1 \left[cos\beta(\sigma,t) f_x(t) + cos\beta(\sigma,t) \int_0^1 \left(cos\beta(s,t) \left(\frac{d^2\beta(s,t)}{dt^2}\right) - sin\beta(s,t) \left(\frac{d\beta(s,t)}{dt}\right)^2 \right) ds \right] d\sigma \quad (5.65)$$

Approximating the equation for β

To solve the equation [5.55] for β is the same as solving $F(\beta) = 0$ using equation [5.65]. That is done using the Newton's Method.

At first β is discretized both in space (s) and time (t) using finite difference. Then $B_i^k \approx \beta(s_i, t_k) =$

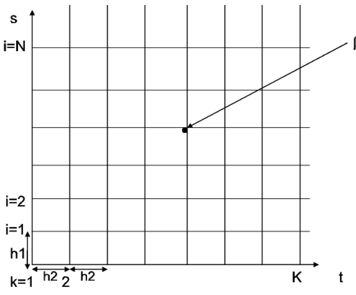

Figure 5.11: discretisation domain

$\beta(i.\Delta s, k.\Delta t)$. The boundary conditions are

$$\frac{\partial\beta(0,t)}{\partial s} = 0 \quad (5.66)$$
$$\beta(0,t) = \theta(t) - \tilde{\alpha}(t), \quad \phi(0,t) = 0 \quad (5.67)$$
$$\tilde{\alpha}(t) = \pi - \alpha(t)$$

The functions $\alpha(t) = \frac{\pi}{2} - \frac{\pi}{2}t$ and $\theta(t) = 1.73 - 0.46t$ $\theta(t)$, are estimated using measurements of the traces of the movement of the hand. From the several values obtained from the measurements, polynomial interpolation is used to estimate the functions.
The initial condition is

$$\beta(s,0) = \beta^0(s) = \theta(t) - \alpha(t) - \phi(s) \quad (5.68)$$

Therefore the unknowns are

$$\beta_i^k \quad (5.69)$$

where

$$i = 1, ..., N$$
$$k = 1, ..., M$$

So β is defined as a matrix:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^1 & \cdots & \cdots & \beta_1^M \\ \beta_2^1 & \ddots & & \beta_2^M \\ \vdots & & \ddots & \vdots \\ \beta_N^1 & \cdots & \cdots & \beta_N^M \end{bmatrix} \tag{5.70}$$

and each column is defined as a vector:

$$\boldsymbol{\beta}^k = \begin{bmatrix} \beta_0^k \\ \beta_1^k \\ \beta_2^k \\ \vdots \\ \beta_N^k \end{bmatrix} \tag{5.71}$$

To approximate the derivatives, Finite Difference for are used. For the derivatives of t for $k = 1, \ldots, M-1$ they are:

$$\begin{aligned} \left(\frac{d\beta}{dt}\right)^k &\approx \frac{\boldsymbol{\beta}^{k+1} - \boldsymbol{\beta}^{k-1}}{2\Delta t} \\ \left(\frac{d^2\beta}{dt^2}\right)^k &\approx \frac{\boldsymbol{\beta}^{k+1} - 2\boldsymbol{\beta}^k + \boldsymbol{\beta}^{k-1}}{(\Delta t)^2} \end{aligned} \tag{5.72}$$

and for $k = M$ they are:

$$\begin{aligned} \left(\frac{d\beta}{dt}\right)^k &\approx \frac{\boldsymbol{\beta}^{k} - \boldsymbol{\beta}^{k-1}}{\Delta t} \\ \left(\frac{d^2\beta}{dt^2}\right)^k &\approx \frac{\boldsymbol{\beta}^{k} - 2\boldsymbol{\beta}^{k-1} + \boldsymbol{\beta}^{k-2}}{(\Delta t)^2} \end{aligned} \tag{5.73}$$

For the derivatives of s for $i = 0 \ldots, N-1$ they are

$$\left(\frac{d\beta}{ds}\right)_i^k \approx \frac{\beta_{i+1}^k - \beta_{i-1}^k}{2\Delta s} \tag{5.74}$$

and for $i = N$ they are:

$$\left(\frac{d\beta}{ds}\right)_i^k \approx \frac{\beta_{i}^k - \beta_{i-1}^k}{\Delta s} \tag{5.75}$$

To approximate the integrals the Trapezoidal rule is used. First a vector $\boldsymbol{h}$ is defined:

$$\boldsymbol{h} = \begin{bmatrix} h(0,t) \\ h(\Delta s, t) \\ \vdots \\ h(N\Delta s, t) \end{bmatrix} \tag{5.76}$$

then the Trapezoidal rule gives:

$$\int_0^1 h(s,t)\, ds \approx \left[\frac{1}{2} 1 \ldots 1 \frac{1}{2}\right] \boldsymbol{h} \Delta s \tag{5.77}$$

For s_i to 1 the integral becomes:

$$\int_{si}^1 h(s,t)\, ds \approx \left[0 \ldots 0 \frac{1}{2} 1 \ldots 1 \frac{1}{2}\right] \boldsymbol{h} \Delta s \tag{5.78}$$

To represent equation 5.65 on a matrix form, the following matrices are now defined:

$$\boldsymbol{D} = \frac{1}{2\Delta s} \begin{pmatrix} -1 & 0 & 1 & & & \\ & -1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 0 & 1 \\ & & & & -2 & 2 \end{pmatrix} \quad \text{an } N \times (N+1) \text{ matrix,} \tag{5.79}$$

$$\boldsymbol{I_z} = \begin{pmatrix} I_z(\Delta s) & & & \\ & I_z(2\Delta s) & & \\ & & \ddots & \\ & & & I_z(N\Delta s) \end{pmatrix} \quad \text{an } N \times N \text{ matrix,} \tag{5.80}$$

$$\boldsymbol{S_1} [\ \tfrac{1}{2} \quad 1 \quad \ldots \quad 1 \quad \tfrac{1}{2}\ ] \quad \text{an } (N+1) \times 1 \text{ vector and} \tag{5.81}$$

$$\boldsymbol{S_2} \begin{pmatrix} 0 & \frac{1}{2} & 1 & \ldots & \ldots & 1 & \frac{1}{2} \\ \vdots & 0 & \frac{1}{2} & 1 & \ldots & 1 & \frac{1}{2} \\ \vdots & & & \ddots & \ddots & & \\ & & & & & \frac{1}{2} & \frac{1}{2} \\ 0 & \ldots & \ldots & & & & 0 \end{pmatrix} \quad \text{an } N \times (N+1) \text{ matrix.} \tag{5.82}$$

Now for each column of $\boldsymbol{\beta}$, $\boldsymbol{\beta}^k$ the following vectors are defined:

$$\begin{aligned} \boldsymbol{h}_1^k &:= \sin\boldsymbol{\beta}^k * \left(\frac{d^2\beta}{dt^2}\right)^k + \cos\boldsymbol{\beta}^k * \left(\frac{d\beta^k}{dt}\right)^2 \\ \boldsymbol{h}_1^k &:= \cos\boldsymbol{\beta}^k * \left(\frac{d^2\beta}{dt^2}\right)^k - \sin\boldsymbol{\beta}^k * \left(\frac{d\beta^k}{dt}\right)^2 \end{aligned} \tag{5.83}$$

So now for each column of $\boldsymbol{\beta}$ equation 5.65 is represented on a matrix form by:

$$\begin{aligned}\boldsymbol{F}(\boldsymbol{\beta}^k) = \boldsymbol{I_z} \cdot \boldsymbol{D} \cdot \boldsymbol{\beta}^k + \\ \gamma \Delta s \cdot \boldsymbol{S}_2 \left[\sin\boldsymbol{\beta}^k f_x(t_k) - \sin\boldsymbol{\beta}^k (\boldsymbol{S}_1 \cdot \boldsymbol{h}_1) \Delta s \right] \\ -\gamma \Delta s \cdot \boldsymbol{S}_2 \left[\cos\boldsymbol{\beta}^k f_x(t_k) + \cos\boldsymbol{\beta}^k (\boldsymbol{S}_1 \cdot \boldsymbol{h}_2) \Delta s \right]\end{aligned} \tag{5.84}$$

2.7 Solving the equation for β using the Newton method

To solve the equation for β, the Newton method is used. First the matrix $\boldsymbol{\beta}$ is transformed into a vector by defining:

$$\boldsymbol{X} := \begin{bmatrix} \beta_1^1 \\ \beta_2^1 \\ \vdots \\ \beta_N^1 \\ \beta_1^2 \\ \vdots \\ \beta_N^M \end{bmatrix}$$

Then for each iteration the Newton method is used:

$$\boldsymbol{X}_{n+1} = \boldsymbol{X}_n - \frac{\boldsymbol{F}(\boldsymbol{X}_n)}{\boldsymbol{J}(X_n)} \tag{5.85}$$

where an approximation of the Jacobian is as follows:

$$\boldsymbol{J}(i,j) = \frac{\boldsymbol{F}(\boldsymbol{X}_{n+1}(i)) - \boldsymbol{F}(\boldsymbol{X}^*_{n+1}(i))}{h} \tag{5.86}$$

for $i, j = 1, \dots, N \cdot M$ where:

$$\boldsymbol{X}^*(i) = \boldsymbol{X}(i) + h \tag{5.87}$$

Then the matrix for $\boldsymbol{\beta}$ using Δt and Δs as 0.1 derived:

$t:$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$s:$	$---$	$----$	$---$	$---$	$---$	$---$	$---$	$---$	$---$	$---$	$---$
0.0|	0	0.2618	0.5236	0.7854	1.0472	1.3090	1.5708	1.8326	2.0944	2.3562	2.6180
0.1|	−0.0142	0.2118	0.4759	0.7468	1.0172	1.2870	1.5562	1.8252	2.0939	2.3626	2.6311
0.2|	−0.0446	0.1559	0.4274	0.7080	0.9872	1.2649	1.5417	1.8178	2.0935	2.3689	2.6442
0.3|	−0.0939	0.0952	0.3782	0.6690	0.9571	1.2429	1.5272	1.8104	2.0930	2.3753	2.6573
0.4|	−0.1634	0.0309	0.3288	0.6301	0.9272	1.2210	1.5127	1.8031	2.0926	2.3816	2.6703
0.5|	−0.2539	−0.0351	0.2801	0.5920	0.8980	1.1997	1.4986	1.7959	2.0922	2.3878	2.6829
0.6|	−0.3705	−0.1004	0.2336	0.5558	0.8701	1.1794	1.4853	1.7891	2.0917	2.3936	2.6950
0.7|	−0.5187	−0.1612	0.1913	0.5230	0.8451	1.1610	1.4732	1.7830	2.0914	2.3989	2.7058
0.8|	−0.6979	−0.2129	0.1561	0.4957	0.8242	1.1458	1.4631	1.7779	2.0911	2.4033	2.7148
0.9|	−0.9136	−0.2493	0.1316	0.4767	0.8097	1.1353	1.4562	1.7743	2.0908	2.4064	2.7211
1.0|	−1.1705	−0.2638	0.1219	0.4692	0.8040	1.1311	1.4534	1.7729	2.0908	2.4076	2.7236

(5.88)

and figure [5.12] shows the fishing rod at different times.

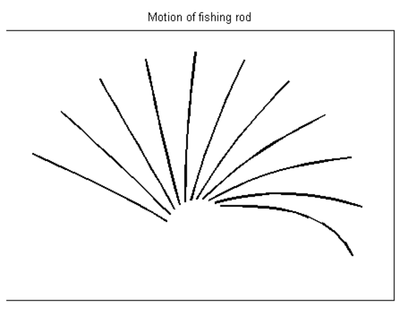

Figure 5.12: motion of fishing rod at different times

2.8 Discussions and Comments

An experimental method was used for this project. There are a lot of measurements and generally it is impossible to make an exact measurement, therefore there are uncertainties in measured quantities due to systematic errors, e.g instrumental errors: venire caliper for measuring the outer and inner radii of the fishing rod and the meter ruler for measuring the lengths, and random errors, e.g rounding off the lengths as well as parallax. The data is also used to obtain functions for calculating unknown quantities,therefore there is a possibility of error propagation.

For simplification purposes a number of assumptions are made. The assumptions are that: the fishing rod is weightless, the difference between the inner and the outer radii is the same at all points,the fishing rod is made out of only graphite fibres range in order to use its young's modulus range (220-1000 GPa). The reel is also assumed to be friction-less.

One person does the casting and the following assumptions are made: his standing point is constant through out the cast,time for casting is estimated to be 1 seconds since it is difficult to measure the exact time, the velocity of the motion is assumed to be constant through out, the motion is sketched on a grid paper in order to measure quantities. Air resistance is neglected.

2.9 Optimal casting angle

By assuming that no forces act on the fishing line when the bait is released, the trajectory from any casting angle can be found by using equation [5.56]

$$x = v_0 t \cos \alpha_0 \tag{5.89}$$

$$y = h_0 + v_o t \sin \alpha_0 - \frac{1}{2} g t^2 \tag{5.90}$$

also the horizontal distance from each cast can be found by using equation 5.59

$$w = v_0 \cos \alpha_0 T \tag{5.91}$$

$$= v_0 \cos \alpha_0 \left(\frac{v_0 \sin \alpha_0 + \sqrt{v_0^2 \sin^2 \alpha_0 + 2gh_0}}{g} \right) \tag{5.92}$$

Where $\alpha_0(t) = \pi/2\beta(L,t)$ is the initial angle of projection.

Since $\alpha_0(t)$ and $av_0(t)$ are only found as a discreet function, they are both interpolated using Δt and the optimal initial angle within error bounds of Δt is found by using Matlab.

For $\Delta = 0.001$ the optimal angle is found to be is $\alpha_0 = 0.9974$ Rad and the longest vertical distance is therefore $w = 6.1129$. Figure 5.13 shows trajectories of different casts with the one with the longest vertical distance highlighted.

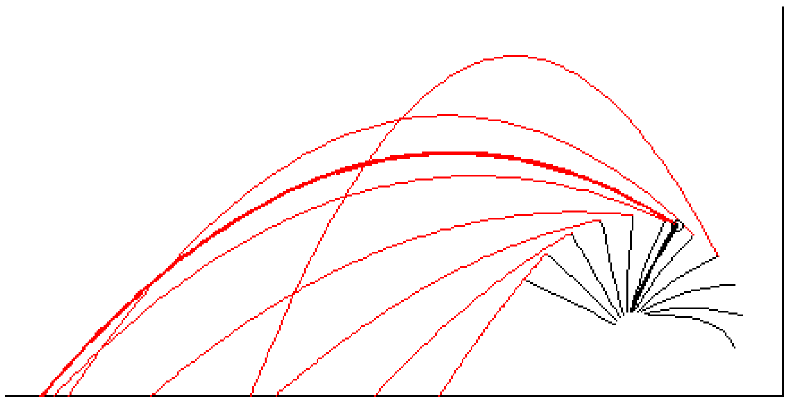

Figure 5.13: Trajectories of different casts

3 Approach II

A flexible beam undergoing large overall motions is typically formulated relative to coordinate system that follows the rigid body motion of the beam. The introduction of this floating frame is motivated by the assumption of infinitesimal stains. With the assumption of small strains, the use of floating strain allows a simple expression for the total potential energy of the beam. The kinetic energy of the system is reduced to a quadratic uncoupled form simply by referring the motion of the system to the inertial frame. This result is a drastic simplification of the inertia operator, which now becomes linear and uncoupled, while the stiffness operator emanating from the potential energy functional becomes nonlinear. As a basis for our discussion, a specific problem is chosen to introduce our formulation: the model problem consists of a flexible beam with one end at the origin of the inertial frame.

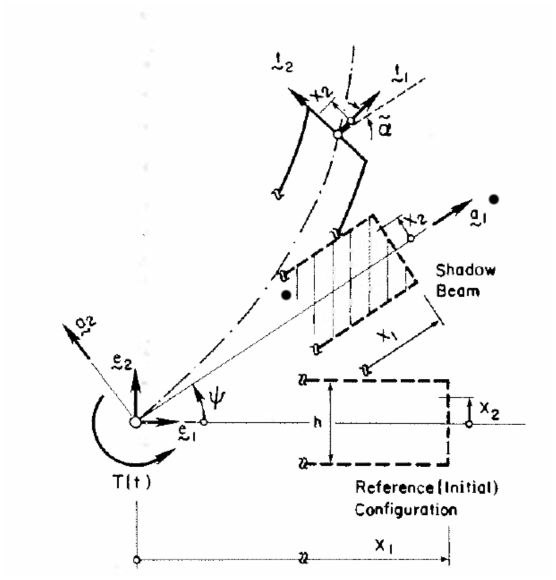

Figure 5.14: Basic Kinematics: Floating and Inertial Frame

Inertial frame: The basic kinematic assumption is the plane sections normal to the axis of the beam in the undeformed configuration remain plane after deformation:

$$\Phi(X_1, X_2, t) := \Phi_0(X_1, t) + X_2 t_2(X_1, t) \tag{5.93}$$

where Φ is the position vector of a material particle initially located at $x = x_1 * e_1 + x_2 * e_2$ in the undeformed configuration.

Kinetic energy: It is possible to show that the kinetic energy of the system relative to the inertial basis reduces to the standard quadratic uncoupled form. Then the following expressions for the kinetic energy of the system are derived:

$$K = \frac{1}{2}\int_{[0,L]} [A_\rho(\dot{u}_1^2 + \dot{u}_2^2) + I_\rho \dot{\theta}^2] dX_1 \tag{5.94}$$

Here the inertia coefficients A_ρ and I_ρ are given by equation

$$A_\rho := \int_{[-\frac{h}{2},\frac{h}{2}]} \rho(X_1, X_2) dX_2 I_\rho := \int_{[-\frac{h}{2},\frac{h}{2}]} \rho(X_1, X_2) X_2^2 dX_2 \tag{5.95}$$

Potential energy: The potential energy is expressed as

$$\Pi := \frac{1}{2}\int_{[0,L]} [EA\Gamma_1^2 + GA_s\Gamma_2^2 + WI(\theta^{'})^2] dS + \Pi_{ext} - T(t)\theta(0,t) \tag{5.96}$$

where Π_{ext} is the potential energy of the external loading acting on the beam.

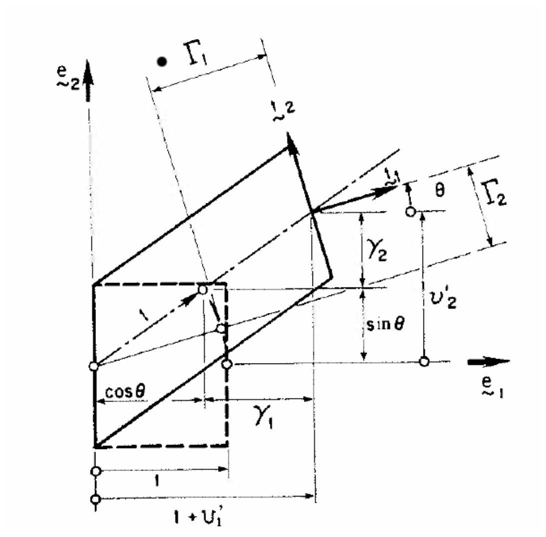

Figure 5.15: Physical Interpretation of the Strain Components of a Beam in the Finite Strain Case

Equations of motion: The equations of motion governing the evolution of the system may be systematically obtained from Hamilton's principle.
Excursion: Hamilton's principle. The Lagrangian L is defined as

$$L = T - V \tag{5.97}$$

where T is the kinetic energy and V is the potential energy of the system in question. Generally speaking, the potential energy of a system depends on the coordinates of all its particles; this may be written as

$$V = V(x_1, y_1, z_1, x_2, y_2, z_2, ...) \tag{5.98}$$

The kinetic energy generally depends on the velocities, which uses the notation $v_x = \frac{dx}{dt} = x$, may be written

$$T = T(x_1, y_1, z_1, x_2, y_2, z_2, ...) \tag{5.99}$$

Thus, a dynamic problem has six dynamic variables for each particle- they are x_1, y_1, z_1 and x_2, y_2, z_2 -and the Lagrangian depends on all 6N variables if there are N particles.

The action, denoted by S, is the time integral of the Lagrangian:

$$S = \int Ldt \tag{5.100}$$

Let q_0 and q_1 be the coordinates at respective initial and final times t_0 and t_1. Using the calculus of variations, it can be shown the Lagrange's equations are equivalent to Hamilton's principle: The system undergoes the trajectory between t_0 and t_1 whose action has a stationary value.
By stationary, the action does not vary to first-order for infinitesimal deformations of the trajectory, with the end-points (q_0, t_0) and (q_1, t_1) fixed. Hamilton's principle can be written as:

$$\int L \triangle S = 0 \tag{5.101}$$

Thus, instead of thinking about particles accelerating in response to applied forces, one might think of them picking out the path with a stationary action.

In terms of the Lagrangian, the classical equations of motion are given by Euler-Lagrange equation:

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} = 0 \tag{5.102}$$

where q_i is the general coordinates of the system. An important property of the Lagrangian formulation is that it can be used to obtain the equations of motion of a system in any set of coordinates, not just the standard Cartesian coordinates.

This approach is concerned on the work of J.C.Simo(stanford University) and L.Vu-Quoc (University of California). We don't go to detail this approach due to the time constraint. Another reason is also the missing accouterment as goniometry or instruments for measurement of the rod. But this Approach is also a possibility to solve our problem.

4 Conclusion

In this project, the static mathematical model of fishing rod was derived by analyzing the forces acting on the rod, the dynamic model was derived based on the static model and some assumption of movements. The parameters used in the models are derived by measuring method. In order to simplify the model, some factors which may have slight influence on the distance of bait casting are excluded. Finally, a system of ODE's for static model and a system of PDE's for dynamic model are derived.

For the static case, the system of ODE's derived for ϕ was solved and the young modulus was fitted to $E = 2.5 \cdot 10^9$ and the radii of the fishing to:

$$r_1(s) = -\frac{1}{1000}s + \frac{3}{400} \tag{5.103}$$

$$r_2(s) = -\frac{1}{1000}s + \frac{19}{2000} \tag{5.104}$$

and then is used in the dynamic case.

For the dynamic case the differential equation for the angle $\beta(s,t)$ was solved by using the Newton method with an approximation for the Jacobian. Then the initial casting angle $\alpha_0(t) = \pi/2 + \beta(s,t)$ was used to find the trajectory of the bait and optimized to give the furthest vertical distance.

Bibliography

[1] F. P. Beer, and E. R. Johnston, *Vector Mechanics for Engineers*, vol. 1, Mc Graw-Hill, New York, 1990.

[2] J. C. Merian, and L. G. Kraige, *Engineering Mechanics*, vol. 1, Statics., John Wiley & Sons, 1987.

[3] Spolek, G.A., 1986,"The mechanics of flycasting": The flyline, *American Journal of Physics*, Vol. 54, No. 9, pp. 832-835.

[4] Shann Paul Jones, http://www.staff.uaf.edu/fnspj/Bulletins/B4%20AKAHPERD%20'05.pdf#search =%22The%20mechanics%20of%20flycasting%3A%20The%20flyline%2C%E2%80%9D%22

[5] J.C. Simo and L. Vu-Quoc,"On the Dynamics of Flexible beams Under Large Overall Motions-The plane case: part I" *Journal of Applied Mechanics*, Vol. 53/854, December 1986

[6] J.C. Simo and L. Vu-Quoc"On the Dynamics of Flexible beams Under Large Overall Motions-The plane case: part II" *Journal of Applied Mechanics*, Vol. 53/855, December 1986

1 Appendix

1.1 Program Used in Static model

- A.m
  matlab script :

```
function res =  A(s)

global epsilon alpha1 alpha2 beta1 beta2

res = (epsilon/4)*((alpha2*s+beta2)^4-(alpha1*s+beta1)^4);
```

- B.m
  matlab script :

```
function res =  B(s)

global epsilon alpha1 alpha2 beta1 beta2

res = epsilon*((alpha2*s+beta2)^3*alpha2-(alpha1*s+beta1)^3*alpha1);
```

- rod.m
  matlab script :

```
clear all
close all
clc
global gamma alpha1 alpha2 beta1 beta2

x = 0:10:190;
y = [0 0.5 1 1.5 2 2.5 3.8 4.8 6.3 8 10 12 14.5 17.8 21.5 26.6 32.5 40.5 49 68];

xx = 0:1:190;
yy = spline(x,-y,xx);
figure
plot(x,-y,'o',xx,yy)
axis equal

for i = 2:190


 talfa = (yy(i)-yy(i-1))/1;
    fi2(i) = atand(talfa);
end;

for i = 2:190
    talfa = (yy(i)-yy(i-1))/1;
    fiRAD(i)  = atan(talfa);
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
options=odeset('RelTol',1e-2,'AbsTol',1e-2,'Mass',@mass);

%Constants
E = 2.5*10^9;
R = 2.10; % length of rod measured in meters
L = 1;
m = 0.3;  % weight in kg
g = 9.8;  % [m/s^2]
w = m*g;  % Newton

%radii of the rod
% a1 = -1/300;
% a2 = -4/4000;
% b1 = 3/400;
% b2 = 19/2000;
```

```
a1 = -1/1000;
a2 = -1/1000;
b1 = 3/400;
b2 = 19/2000;


%Normalization of the equation
gamma = pi*E*R^2/w;
alpha1 = a1;
alpha2 = a2;
beta1 = b1/R;
beta2 = b2/R;

%Initial values
u0 =[0;0];

[s,fi]=ode15s(@f,[0:1/189:L],u0,options);

%Derivation of coordinates
xc(1) = 0;
yc(1) = 0;
for i = 2:size(s)
    xc(i) = xc(i-1) + (s(i)-s(i-1))*cos(fi(i,1));
    yc(i) = yc(i-1) + (s(i)-s(i-1))*sin(fi(i,1));
end
figure
plot(xc,-yc)
hold on
xm = linspace(0,190,20)/(100*R);
ym = [0 0.5 1 1.5 2 2.5 3.8 4.8 6.3 8 10 12 14.5 17.8 21.5 26.6 32.5 40.5 49 68]/(100*R);

plot(xm,-ym,'r')
legend('calculated','mesured')
```

- mass.m
  matlab script :

```
function res =  mass(s,u)

global gamma alpha1 alpha2 beta1 beta2

res = [1 0;0 A(s)];
```

- f.m
  matlab script :

```
function du =  f(s,u)

global epsilon alpha1 alpha2 beta1 beta2
```

```
du = zeros(2,1);
du(1) = u(2);
du(2) = cos(u(1))-B(s)*u(2);
```

1.2 Program Used in Dynamic model

- A.m
  matlab script :

```
function res =  A(s)

global epsilon alpha1 alpha2 beta1 beta2

res = (epsilon/4)*((alpha2*s+beta2)^4-(alpha1*s+beta1)^4);
```

- alpha.m
  matlab script :

```
function a = alpha(t)

alpha0 = theta0(0);
eta    = pi / 6;
alpha1 = -(pi - theta0(1) - eta);

a = (alpha1-alpha0)*(t-1) + alpha1;
```

- B.m
  matlab script :

```
function res =  B(s)

global epsilon alpha1 alpha2 beta1 beta2

res = epsilon*((alpha2*s+beta2)^3*alpha2-(alpha1*s+beta1)^3*alpha1);
```

- d2r0dt2.m
  matlab script :

```
function d2rdt2 = d2r0dt2(t)
 d2rdt2 = -0.1318 * 3 * 2 * t - 0.3948 * 2;
 d2rdt2 = d2rdt2 / 2.1;
```

- d2theta0dt2.m
  matlab script :

```
function d2adt2 = d2theta0dt2(t)
d2adt2 = 0.0;
```

- dr0dt.m
  matlab script :

```
function drdt = dr0dt(t)
 drdt = -0.1318 * 3 * t.^2 - 0.3948 * 2 * t + 0.3687;
 drdt = drdt / 2.1;
```

- dtheta0dt.m
  matlab script :

```
function dadt = dtheta0dt(t)
dadt = 0.46;
```

- f.m
  matlab script :

```
function du =  f(s,u)

global epsilon alpha1 alpha2 beta1 beta2

du = zeros(2,1);
du(1) = u(2);
du(2) = cos(u(1))-B(s)*u(2);
```

- fBox.m
  matlab script :

```
function res = fBox(x)

global n m

BB   = reshape(x,n,m);
Fmtx = FF(BB);                 % matrix F(beta)
res  = reshape(Fmtx,n*m,1);    % vector col
```

- FF.m
  matlab script :

```
function x = FF(BB)  % BB --> matrix
                     % BB(i,k) = Beta(s(i),t(k))
                     % ds step for s
                     % st dtep for t


global E Gama ds dt t n m  L
```

```
Bnew = zeros(n+1,m+1);
Bnew(2:n+1,2:m+1) = BB;

Bnew(:,1) = theta0(0) - alpha(0) - fi(n+1,0);    % initial condition
Bnew(1,:) = theta0(t) - alpha(t);                % boundry condition

x  = zeros(n,m);

D = zeros(n,n+1);
for j = 1:n
    D(j,j) = -1;
end
for j =1:n-1
    D(j,j+2) = 1;
end
D(n,n)   = -2;
D(n,n+1) =  2;
D = 1/(2*ds)*D;

Iz = diag(Iz_nondim(ds:ds:1, L));

s1      = ones(n+1,1);
s1(1)   = 1/2;
s1(n+1) = 1/2;

s2 = zeros(n,n+1);
for j =1:n
    s2(j,j+1) = 1/2;
    for i = j+1:n
        s2(j,i+1)=1;
    end
    s2(j,n+1) = 1/2;
end

for k = 2:m+1
      b  = Bnew(:,k);
      bt = Bnew(:,k-1);

      if k == m+1
          b2 = Bnew(:,k-2);
          h1 = 1/(dt)^2*(sin(b).*(b-2*bt+b2) + cos(b).*((b-bt).^2));
          h2 = 1/(dt)^2*(cos(b).*(b-2*bt+b2) - sin(b).*((b-bt).^2));
      else
          bp = Bnew(:,k+1);
          h1 = 1/(dt)^2*(sin(b).*(bp-2*b+bt) + 1/4*cos(b).*((bp-bt).^2));
          h2 = 1/(dt)^2*(cos(b).*(bp-2*b+bt) - 1/4*sin(b).*((bp-bt).^2));
      end
    x(:,k-1) = FunF(b,k,h1,h2,s1,s2,Iz,D) ;
end
```

- fi.m
  matlab script :

```
function result = fi(mm,stAngle);

global epsilon alpha1 alpha2 beta1 beta2

options=odeset('RelTol',1e-2,'AbsTol',1e-2,'Mass',@mass);

% Constants
E = 18.0*10^8;  % Youngs module
R = 2.10;       % length of rod measured in meters
m = 0.3;        % weight in kg
g = 9.81;       % [m/s^2] acceleration of gravity
w = m*g;        % Newton

% radii of the rod
a1 = -1/300;
b1 = 3/400;

a2 = -1/1000;
b2 = 19/2000;

% Normalization of the equation
epsilon  = pi*E*R^2/w;
alpha1   = a1;
alpha2   = a2;
beta1    = b1/R;
beta2    = b2/R;

% Initial values
u0 =[0;stAngle];

[s,phi] =  ode15s(@f,[0:1/(mm-1):1],u0,options);
result  =  phi(:,1);
```

- FishingRod.m
  matlab script :

```
function [beta,BetaOfTips,xLastLine,yLastLine,dt,ds,m] = FishingRod;

global E ds dt t n m mw L T

%Constants
E = 18.0*10^8;  % Youngs module
mw  = 0.3;      % weight of the bait in kg
L  = 2.10;      % length the fishing rod in m
T  = 1;         % time the motion of fishing rod in sec

ds = 0.1;
dt = 0.1;
s  = 0:ds:1;
t  = 0:dt:1;    % time :)
n  = size(s,2)-1;
m  = size(t,2)-1;
```

```
vecB = theta0(t) - alpha(t);
vecB = vecB(2:size(vecB,2));
w = zeros(n,m);

% starting guess - Beta0

% for i=1:m                         % starting guess from the static
%    w(i,:) = fi(n,vecB(i))';   % fishing rod
% end

w = [];                             % starting guess from the straight
for i=1:n                           % fishing rod
   w = [w;vecB];
end

beta0 = w;

nGuess = 5;

disp('starting newton')
beta = NEWTON(beta0,nGuess);
disp('finished newton')
Bnew = zeros(n+1,m+1);
Bnew(2:n+1,2:m+1) = beta;

Bnew(:,1) = theta0(0) - alpha(0) - fi(n+1,0);    % initial condition
Bnew(1,:) = theta0(t) - alpha(t);                % boundry condition

beta = Bnew;

BetaOfTips = beta(end,:);

% plot of the fishing rod

fill([-6.5 1.5 1.5 -6.5],[0 0 4 4],'w')
x = zeros(n+1,1);
y = zeros(n+1,1);
for k=1:m+1
    vec = beta(:,k);
    x(1:end,k) = r0((k-1)*dt)*cos(theta0((k-1)*dt));
    y(1:end,k) = r0((k-1)*dt)*sin(theta0((k-1)*dt));
    for i = 2:n+1
        s = [0:i-1]'*ds;
        cp = cos(vec(1:i));
        sp = sin(vec(1:i));
        x(i,k) = x(1,k) + trapz(s,cp);
        y(i,k) = y(1,k) + trapz(s,sp);
    end

    hold on
    plot(x(:,k),y(:,k),'k','LineWidth',2)
    axis([-6.5 1.5 0 4])
    axis equal
    axis([-6.5 1.5 0 4])

    P(k) = getframe;
end;
% movie(P)

xLastLine = x(end,:);
yLastLine = y(end,:);
```

- flight.m
  matlab script :

```
clear all
close all
clc

[beta,BetaOfTips,xLastLine,yLastLine,dt,ds,m] = FishingRod;

g  = 9.81;    %  acceleration of gravity
mw = 0.3;     %  weight of the bait

% plot of the flight

for k =2 :m+1
    hx = xLastLine(k-1);
    hy = yLastLine(k-1);
    alpha0 = BetaOfTips(k-1)+pi/2;

    dBetadt = (BetaOfTips(k)-BetaOfTips(k-1))/dt;
    vx = dr0dt((k-1)*dt)*cos(theta0((k-1)*dt))-r0((k-1)*dt)*dtheta0dt((k-1)*dt)*sin(theta0((k-1)*dt))+.
         trapz(0:ds:1,-sin(beta(:,k-1))*dBetadt);

    vy = dr0dt((k-1)*dt)*sin(theta0((k-1)*dt))+r0((k-1)*dt)*dtheta0dt((k-1)*dt)*cos(theta0((k-1)*dt))+.
         trapz(0:ds:1,cos(beta(:,k-1))*dBetadt);

    v0 = sqrt(vx^2+vy^2)*2

    T    = (v0*sin(alpha0)+sqrt(v0^2*(sin(alpha0))^2+2*g*hy))/g;
    w    = v0*cos(alpha0)*T;
    maxh = hy+v0^2*(sin(alpha0))^2/(2*g);
    t    = linspace(0,T);
    x    = hx + v0*t*cos(alpha0);
    y    = hy + v0*t*sin(alpha0)-g*t.^2/2;
    if k == 7
        plot(x,y,'r','LineWidth',2)
    else
        plot(x,y,'r--')
    end
end
title('Flight of the bait')
axis off
```

- FunF.m
  matlab script :

```
function xVec = FunF(b,k,h1,h2,s1,s2,I,D)

global E  ds dt t L mw T
gamma = mw * L^3 / (E * T^2 * Iz_dimensional(0));

xVec = I*D*b + gamma*ds*(s2*sin(b))* (fx((k-1)*dt) - (s1'*h1)*ds);
xVec = xVec  - gamma*ds*(s2*cos(b))* (fy((k-1)*dt) + (s1'*h2)*ds);
```

- fx.m
  matlab script :

```
function res = fx(t)

%drdt: dr/dt
%d2rdt2: d^2r/dt^2 and so on

%the input are vectors of the values of the functions of r and teta and
%their first an second derivatives

res = d2r0dt2(t)*cos(theta0(t))-2*dr0dt(t)*sin(theta0(t))*dtheta0dt(t)-r0(t)*cos(theta0(t))*dtheta0dt(t)
      r0(t)*sin(theta0(t))*d2theta0dt2(t);
```

- fy.m
  matlab script :

```
function res = fy(t)

%drdt: dr/dt
%d2rdt2: d^2r/dt^2 and so on

%the input are vectors of the values of the functions of r and teta and
%their first an second derivatives

res = d2r0dt2(t)*sin(theta0(t))+2*dr0dt(t)*cos(theta0(t))*dtheta0dt(t)-r0(t)*sin(theta0(t))*dtheta0dt(t)
      r0(t)*cos(theta0(t))*d2theta0dt2(t);
```

- Iz_dimensional.m
  matlab script :

```
function I = Iz_dimensional(s)
I = pi * (r2N(s).^4 - r1N(s).^4)/4;
```

- Iz_nondim.m
  matlab script :

```
function I = Iz_nondim(s, L)
Im = Iz_dimensional(0);
I = Iz_dimensional(L * s) / Im;
```

- Jac.m
  matlab script :

```
function res = Jac(x)

global n m

N    = size(x,1);     % x vec col
fvec = fBox(x);
ep   = sqrt(eps);

for j=1:N
  tmp  = x(j);
  hh   = ep*abs(tmp);

  if (hh < 1.0e-13)
     hh = ep;
  end

  x(j) = x(j)+hh;
  hh   = x(j) - tmp;
  fnew = fBox(x);
  x(j) = tmp;

  for i=1:N
      res(i,j)=(fnew(i)-fvec(i))/hh;
  end
end
```

- mainNewNorm.m
  matlab script :

```
clear all
close all
clc

global E ds dt t n m mw L T

%Constants
E = 18.0*10^8;  % Youngs module
mw  = 0.3;      % weight of the bait in kg
L  = 2.10;      % length the fishing rod in m
T  = 1;         % time the motion of fishing rod in sec

ds = 0.1;
dt = 0.1;
s  = 0:ds:1;
t  = 0:dt:1;    % time :)
n  = size(s,2)-1;
m  = size(t,2)-1;

% starting guess - Beta0

vecB = theta0(t) - alpha(t);    % starting guess from the static
vecB = vecB(2:size(vecB,2));    % fishing rod
w = zeros(n,m);
```

```
% for i=1:m                     % starting guess from the straight
%    w(i,:) = fi(n,vecB(i))';   % fishing rod
% end

w = [];
for i=1:n
   w = [w;vecB];
end

beta0 = w;

% x = FF(beta0,n,m);
nGuess = 5;

disp('starting newton')
beta = NEWTON(beta0,nGuess);
disp('finished newton')
Bnew = zeros(n+1,m+1);
Bnew(2:n+1,2:m+1) = beta;

Bnew(:,1) = theta0(0) - alpha(0) - fi(n+1,0);    % initial condition
Bnew(1,:) = theta0(t) - alpha(t);                % boundry condition

beta = Bnew;

% plot & movie

fill([-1.3 1.3 1.3 -1.3],[0.2 0.2 2 2],'w')

x = zeros(n+1,1);
y = zeros(n+1,1);
for k=1:m+1
    vec = beta(:,k);
    x(1:end) = r0((k-1)*dt)*cos(theta0((k-1)*dt));
    y(1:end) = r0((k-1)*dt)*sin(theta0((k-1)*dt));
    for i = 2:n+1
        s = [0:i-1]'*ds;
        cp = cos(vec(1:i));
        sp = sin(vec(1:i));
        x(i) = x(1) + trapz(s,cp);
        y(i) = y(1) + trapz(s,sp);
    end
    hold on
    plot(x,y,'k','LineWidth',2)
    axis([-1.3 1.3 0.2 2])
    axis equal
    axis([-1.3 1.3 0.2 2])

    P(k) = getframe;
end;
movie(P)
title('Motion of fishing rod')
axis off
```

- mass.m
  matlab script :

```
function res =  mass(s,u)

global gamma alpha1 alpha2 beta1 beta2

res = [1 0;0 A(s)];
```

- NEWTON.m
  matlab script :

```
function res = NEWTON(beta,nGuess)

global m n

x0 = reshape(beta,n*m,1);

xOLD = x0;
for k = 1 :nGuess
    k
    A = fBox(xOLD);
    J = Jac(xOLD);
    D  = -J\A;
    xNEW = xOLD + D;
    xOLD = xNEW;
end

res = reshape(xNEW,n,m);
```

- ro.m
  matlab script :

```
function r = r0(t)
 r = -0.1318 * t.^3 - 0.3948 * t.^2 + 0.3687 * t + 1.7654;
 r = r / 2.1;
```

- r1N.m
  matlab script :

```
function res =  r1N(s)


res = -s/300+3/400;

% inner radius of rod measured in m.
% s  measured in m
% r1 measured in m
```

- r2N.m
  matlab script :

```
function res =  r2N(s)


res = -s/1000+19/2000;

% outer radius of rod measured in m.
% s  measured in m
% r2 measured in m
```

- theta0.m
  matlab script :

```
function theta = theta0(t)
theta = pi - 1.73 + 0.46 * t;
```

- BestAlfa.m
  matlab script :

```
clear all
clc

[beta,BetaOfTips,xLastLine,yLastLine,dt,ds,m] = FishingRod;


alfa = BetaOfTips
kx  = 0:(size(alfa,2)-1)
% plot(kx,alfa,'o')

%% SPLINE %%
figure                          %% continuous version of alfa points

xx = 0:.25:10;
yy = spline(kx,alfa,xx);        %% xx corerespond with k  (steps)
plot(kx,alfa,'mo',xx,yy)        %% yy corerespond with alfa points (angles of relasing the bait)
```

- BestFlight.m
  matlab script :

```
function BestFlight(xS,yS,alfaS,v0)

g  = 9.81;    %  acceleration of gravity
mw = 0.3;     %  weight of the bait

% plot of the best flight


hx = xS;
hy = yS;
alpha0 = alfaS + pi/2;

T    = (v0*sin(alpha0)+sqrt(v0^2*(sin(alpha0))^2+2*g*hy))/g;
w    = v0*cos(alpha0)*T;
```

```
maxh = hy+v0^2*(sin(alpha0))^2/(2*g);
t    = linspace(0,T);
x    = hx + v0*t*cos(alpha0);
y    = hy + v0*t*sin(alpha0)-g*t.^2/2;

plot(x,y,'r','LineWidth',2)
```

- FishingRod2.m
  matlab script :

```
function [beta,BetaOfTips,x,y,xLastLine,yLastLine,dt,ds,m] = FishingRod2;

global E ds dt t n m mw L T

%Constants
E = 18.0*10^8;  % Youngs module
mw  = 0.3;      % weight of the bait in kg
L  = 2.10;      % length the fishing rod in m
T  = 1;         % time the motion of fishing rod in sec

ds = 0.1;
dt = 0.1;
s  = 0:ds:1;
t  = 0:dt:1;    % time :)
n  = size(s,2)-1;
m  = size(t,2)-1;


vecB = theta0(t) - alpha(t);
vecB = vecB(2:size(vecB,2));
w = zeros(n,m);

% starting guess - Beta0

% for i=1:m                        % starting guess from the static
%    w(i,:) = fi(n,vecB(i))';      % fishing rod
% end

w = [];                            % starting guess from the straight
for i=1:n                          % fishing rod
   w = [w;vecB];
end

beta0 = w;

nGuess = 5;

disp('starting newton')
beta = NEWTON(beta0,nGuess);
disp('finished newton')
Bnew = zeros(n+1,m+1);
Bnew(2:n+1,2:m+1) = beta;

Bnew(:,1) = theta0(0) - alpha(0) - fi(n+1,0);    % initial condition
Bnew(1,:) = theta0(t) - alpha(t);                % boundry condition

beta = Bnew;

BetaOfTips = beta(end,:);

% plot of the fishing rod

fill([-6.5 1.5 1.5 -6.5],[0 0 4 4],'w')
x = zeros(n+1,1);
y = zeros(n+1,1);

for k=1:m+1
    vec = beta(:,k);
    x(1:end,k) = r0((k-1)*dt)*cos(theta0((k-1)*dt));
    y(1:end,k) = r0((k-1)*dt)*sin(theta0((k-1)*dt));
    for i = 2:n+1
        s = [0:i-1]'*ds;
        cp = cos(vec(1:i));
        sp = sin(vec(1:i));
        x(i,k) = x(1,k) + trapz(s,cp);
        y(i,k) = y(1,k) + trapz(s,sp);
    end

    hold on
    plot(x(:,k),y(:,k),'k','LineWidth',1)
    axis([-6.5 1.5 0 4])
    axis equal
    axis([-6.5 1.5 0 4])

    P(k) = getframe;
end;
% movie(P)

xLastLine = x(end,:);
yLastLine = y(end,:);
```

- OptymalSol.m
  matlab script :

```
function [DistMax,alfaMax,HandAlfaMax,xLastMax,yLastMax,xHandMax,yHandMax]=OptymalSol

[r,alfa,HandAlfa,X,Y,xL,yL,V0] = ReachedPoints;  %% reached points

h = 0.25;   %% spline step

%% reached distances depends on step
rx  = 0:(size(r,1)-1);
ry  = -r;

%% SPLINE RP %%
%% continuous version of reached points

xx = 0:h:10;                  %% xx correspond with k  (steps)
```

```
yy = spline(rx,ry,xx);         %% yy correspond with RP (reached points)

%% THE FURTHEST FLIGHT
[yMax,kMax] = max(yy);         %% yMax is the maximum distance
xMax        = xx(kMax);        %% xMax correspond with k  (steps)
DistMax     = yMax;

%% THE BEST ALFA
kx  = 0:(size(alfa,2)-1);

%% SPLINE ALFA %%
%% continuous version of alfa points

xx = 0:h:10;                   %% xx correspond with k  (steps)
yy = spline(kx,alfa,xx);       %% yy correspond with alfa points (angles of relasing the bait)
alfaMax  = yy(kMax);

yy = spline(kx,HandAlfa,xx);       %% yy correspond with alfa points (angles of relasing the bait)
HandAlfaMax  = yy(kMax);

for i=1:size(X(:,end),1)
    yy = spline(kx,X(i,:),xx);       %% yy correspond with X points
    X_Max(i)  = yy(kMax);

    yy = spline(kx,Y(i,:),xx);       %% yy correspond with Y points
    Y_Max(i)  = yy(kMax);

%     hold on
%     plot(X_Max(i),Y_Max(i),'ko');
end

xLastMax = X_Max(end);
yLastMax = Y_Max(end);

xHandMax = X_Max(1);
yHandMax = Y_Max(1);

% figure
V0 = [V0,V0(end)];             %% fake velocity point
xx = 0:h:10;                   %% xx correspond with k  (steps)
yy = spline(kx,V0,xx);         %% yy correspond with start velocity
%  plot(kx,V0,'go',xx,yy)
v0 = yy(kMax);

hold on
BestFlight(xLastMax,yLastMax,alfaMax,v0)

hold on
plot(X_Max,Y_Max,'k','LineWidth',2)
hold on
plot(xLastMax,yLastMax,'ko')
```

- ReachedPoints.m
  matlab script :

```
function [res,alfa,HandAlfa,xx,yy,xL,yL,V] = ReachedPoints      % Reached Points / Places

[beta,BetaOfTips,X,Y,xLastLine,yLastLine,dt,ds,m] = FishingRod2;

alfa     = BetaOfTips;
HandAlfa = beta(1,:);
xL   = xLastLine;
yL   = yLastLine;
xx   = X;
yy   = Y;

g  = 9.81;    %  acceleration of gravity
mw = 0.3;     %  weight of the bait

% plot of the flight
r  = [];
vv = [];
for k =2 :m+1
    hx = xLastLine(k-1);
    hy = yLastLine(k-1);
    alpha0 = BetaOfTips(k-1)+pi/2;

    dBetadt = (BetaOfTips(k)-BetaOfTips(k-1))/dt;
    vx = dr0dt((k-1)*dt)*cos(theta0((k-1)*dt))-r0((k-1)*dt)*dtheta0dt((k-1)*dt)*sin(theta0((k-1)*dt))+..
         trapz(0:ds:1,-sin(beta(:,k-1))*dBetadt);

    vy = dr0dt((k-1)*dt)*sin(theta0((k-1)*dt))+r0((k-1)*dt)*dtheta0dt((k-1)*dt)*cos(theta0((k-1)*dt))+..
         trapz(0:ds:1,cos(beta(:,k-1))*dBetadt);

    v0 = sqrt(vx^2+vy^2)*2;
    vv = [vv,v0];

    T    = (v0*sin(alpha0)+sqrt(v0^2*(sin(alpha0))^2+2*g*hy))/g;
    w    = v0*cos(alpha0)*T;
    maxh = hy+v0^2*(sin(alpha0))^2/(2*g);
    t    = linspace(0,T);
    x    = hx + v0*t*cos(alpha0);
    y    = hy + v0*t*sin(alpha0)-g*t.^2/2;
    if k > 4
        plot(x,y,'r')
    end
    r    = [r ; x(end)];
end
title('Flight of the bait')
axis off

res = r;
V   = vv;
```