This watermark does not appear in the registered version - http://www.clicktoconvert.com

Simulation and Refinement of a Modified Rowing Ergometer

MAE 491 Project

By: Drew C. Tennant (dct7@cornell.edu)
Advisor: Andy Ruina (Ruina@cornell.edu)
Dept. of Theoretical and Applied Mechanics

May 21st, 2004

This watermark does not appear in the registered version - http://www.clicktoconvert.com

II: Abstract

The Modified Ergometer Project has been ongoing since 1991. The goal is to create an ergometer that accurately simulates the feel of rowing a boat. The specific goals for this year were to create a computer simulation of the current ergometer, and to improve its working condition. Many repairs were made and the dynamics of the machine were modeled and incorporated into a MATLAB program. The velocity profile created by the simulation differed significantly from that derived from laboratory data, which suggest inaccuracies in the simulation. Comparing mine to another computer simulation of rowing showed unrealistic modeling of the rower's body motions is largely responsible for the errors. Varying simulation parameters showed that the moment of inertia of the flywheel and radius of the ergometer's wheels had significant affect on the shape of the velocity curve, while the stiffness of the position return spring, surprisingly, had almost no affect. Future workers should develop a more accurate model of the rower's body motions and try to transform the modified ergometer into a marketable product.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

III: Introduction

III.1:Background on rowing ergometers

III.1.i: Stationary Ergometers

Rowers often train on rowing machines or ergometers when access boats and water is not available. The standard rowing ergometer is produced by a company called Concept II. The major limitation of such products is that they are completely stationary. The rower must completely stop his body at each end of the stroke, quickly accelerating in the mid parts of the stroke. The rower goes through the basic rowing motions, but the velocity and acceleration of various body parts, and the forces in the handle at various points in the stroke are much different than that seen in a boat. Because of this difference, rowers who use an ergometer frequently may develop habits that generate power efficiently on an ergometer, but do these habits often do not translate to efficient rowing in a boat. The rower develops strength and fitness, but not rowing ability.

III.1.ii: Dynamic Ergometers

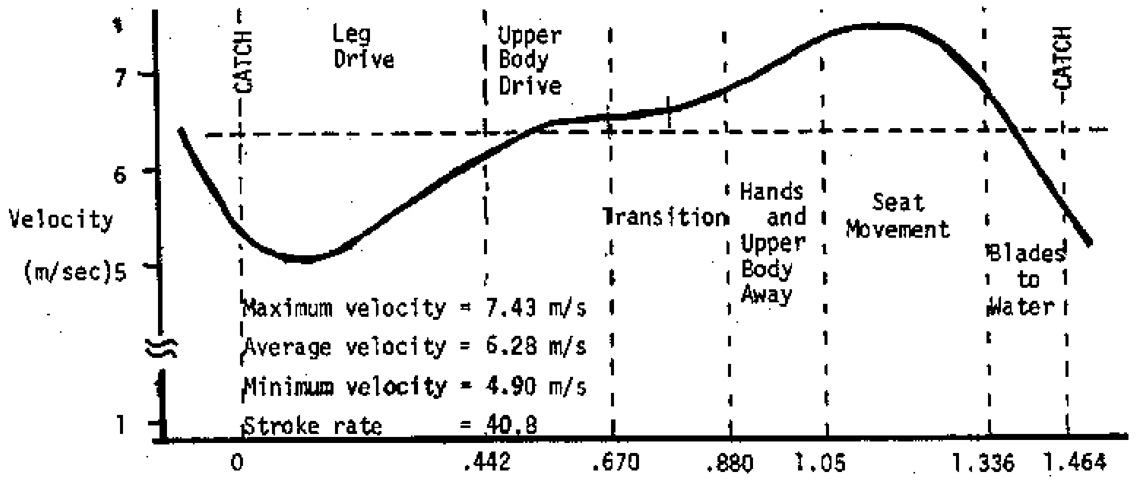
Several ergometers on the market that try to combat these problems. The two most popular dynamic ergometers are the RowPerfect by CARE and the Concept II slides. Information about these two products can be found on the web at www.Rowperfect.com and www.concept2.com, respectively. In each, the ergometer is set on wheels, freeing it to slide back and forth with a certain range. The rower and ergometer oscillate back and forth around a stationary center of mass. Because the erg is much lighter than the rower, it moves much more than the rower. The rower is able to move back and forth through the stroke in a more smooth and natural manner. The limitation of this type of ergometer

This watermark does not appear in the registered version - http://www.clicktoconvert.com

is that the center of mass of the system does not accelerate or decelerate as is seen in real rowing.

III.1.ii: Modified ergometer Project at Cornell

The modified ergometer project has been going on since 1991 with the overarching goal to build a rowing ergometer that better simulates the “feel” of rowing a boat. In this case, “feel” refers to the velocities, accelerations and forces experienced by rower in a boat. Previous researchers have focused mainly on matching velocites. Ideally the velocity profile of the ergometer would resemble that of a boat on the water, as seen below:



. Velocity of crew shell during rowing cycle.
(Ferris, 1981: 55)

The 1991-1992 researches produced a prototype that they felt was a vast improvement over any rowing ergometers currently available. The original design was accomplished by mounting a standard Concept II rowing ergometer on wheels that are driven by the pulling of the handle, so that as the user pulled the handle, the ergometer accelerated as a boat would. Data showed that the ergometer had a velocity profile closely resembling that of a boat, but the machine had many mechanical shortcomings.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

Students continued work in 1992 1993, 1999, seeking to refine the machine and make it more user friendly.

By 1999, the ergometer worked more consistently. Don Nelson, who worked on the project that year, advised that further work should be directed towards making the ergometer more reliable and usable. In 2002 two students began working with the goal of making the machine more "robust." They fixed many mechanical problems, yet in the process, they altered several aspect of the project that was key to performance.

When I received the ergometer at the beginning of the 2003-2004 academic year, it had been altered in ways that seriously affected its "feel." Data collected showed that the ergometer did not model the velocity profile of a boat as well as it had in 1999, or even as well as it did in 1992. It seems that the researchers in 2002, and possibly others, poorly understood the system with which they were working. As a result, they "improved" it in ways that actually hurt the project as a whole.

III.2: New Objectives

III.2.i: Need

Students researchers on this project would need a way to gain understanding and insight on the workings of the modified ergometer. That way they would fully understand the affect of any of their modifications. This would minimize the possibility of future students experiencing the frustration that I underwent this semester while trying to understand where previous workers had gone wrong.

III.2.ii: Problem Statement:

Previous researchers have not fully understood the ergometer (i.e. how each component affects the overall performance) and thus have altered the project in ways that they did

This watermark does not appear in the registered version - http://www.clicktoconvert.com

not fully understand. Additionally, previous researchers have left poor records, making it difficult to track the progress of the project over the years, learn from the improvements and mistake of other researchers, or continue iterating design improvements in a logical way. Students working on this project need a way that they can explore the different parameters of the machine and see the affect of various improvements without actually altering the machine itself.

III.2.iii: Solution:

Professor Ruina suggested that I create a computer simulation of the ergometer. The basic tasks of the simulation would be to and integrate the ergometer's equations of motion and produce plots of position and velocity - as well as several other related parameters – that would illustrate how the ergometer is working. Additional objectives for the simulation that it is comprehensible to future students, allows easy variation of design parameters, and accurately models the dynamics of the ergometer

This task could be accomplished through the following steps:

(1) gaining a basic understanding of the ergometer through use and inspection
(2) deriving the equations of motions for the ergometer
(3) writing a MATLAB program that integrates these equations using an ODE solver and outputs vectors of position and velocity over time

III.3: Overview of experiments

III.3.i: Set up

In order to become familiar with the project, I read the reports of previous researcher teams, as well as journals articles on the dynamics and kinematics of rowing. The

This watermark does not appear in the registered version - http://www.clicktoconvert.com

modified ergometer was then assembled and studied first hand. This led to the realization that the ergometer needed several repairs.

III.3.ii: Repairs

A new floating pulley for the ergometer and several other small components were produced. The bungees used to maintain tension on the ergometer handle were replaced with a stock chain-tension assembly from a Concept II ergometer. The bungee group connected to the drive wheels (which returns the erg to its original position after each stroke) was modified, effectively dividing the spring constant of the group by 2. Modifications to the ergometer were made continually throughout the semester in efforts to make the machine more reliable and accurate.

III.3.iii: Modeling the Dynamics:

The next step was to derive the equations of motion for the ergometer. Starting with a free-body diagram and Newton's Second Law, I eventually arrived at the set of equations found in "Methods". The equations describe the acceleration of the ergometer and the angular acceleration of the flywheel. They are dependent upon the motion of the rower at a given time.

III.3.iv: MATLAB Simulation:

These equations were incorporated into a MATLAB function that solves over a given time span, plots the results, and animates the resulting motions. I decided to use MATLAB for the computer simulation because I had the most familiarity with it and felt it would be the easiest to use. The rower's body motions were determined using a MATLAB program written by a previous Cornell student. Spring constants, erg and body dimensions, and other inputs were measured in the laboratory. My equations of motion

This watermark does not appear in the registered version - http://www.clicktoconvert.com

were also incorporated into a more sophisticated rowing simulation developed by Dave Cabrera.

III.3.v: Video Data:

Video was taken of the ergometer at several times throughout the project. By stepping through the video frame-by-frame and recording the position of the ergometer and the position of the rower's center of mass at each frame, I was able to generate position and velocity profiles for the ergometer and rower over the course of one stroke.

III.3.vi: Investigating a possible alteration:

Professor Ruina has suggested to several students that the floating pulley be replaced with an open differential to split the torque between the flywheel and the drive train. In order to investigate the effect of such a switch, I derived the equations of motion for such a system and incorporated them into the simulation as well.

III.4: Results

As a result of repairs made this year, the ergometer now rows more smoothly and reliably than it did when I found it. Data taken from video of the ergometer reveals that it models the movements of a boat better than it did a year ago, but still does not perform as well as Nelson's model did in 1999.

Output from the computer simulation was compared to the video data. This showed that the predicted motions of the ergometer match the actual motions to a certain extent, although there are still some discrepancies. Altering parameters such as spring constants and gear ratio's in the simulation has illustrated how each parameter affects overall behavior. In the end, such manipulation provided valuable insight, but did not provide a clear explanation of which parameters could be changed to optimize the

This watermark does not appear in the registered version - http://www.clicktoconvert.com

performance of the machine. Comparing simulated results to for the differential drive mechanism to those with the floating pulley showed very little difference in performance.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

IV: Methods

IV.1: Ergometer Repairs

IV.1.i: floating pulley

Misalignments in the ergometer pulled the sprocket in the floating pulley to one side, causing the chain to rub against the housing of the floating pulley and occasionally jam. The pulley was connected to the drive train by a small hook that was unable to endure heavy loads and frequently sheared. I machined a new floating pulley wide enough to accommodate the chain, and machined washers to hold the sprocket in place without interfering with the chain. I then made a direct connection between floating pulley and drive chain, eliminating the connecting hook used before.

IV.1.ii: chain tension bungees

Previously, tension in the ergometer chain was maintained by a long bungee run through a series of pulleys. Excessive friction in the pulleys caused the system to bind up, making the chain go slack. I replaced the entire group of bungees with the stock chain-tension mechanism from a concept II ergometer.

IV.1.iii position return spring

A group of bungees inside the monorail was connected directly to the ergometers drive train. These bungees acted as a position return mechanism, simulating boat drag. I attached a pulley system to the position return spring and ran the chain through this pulley system. This decreased the amount that the spring extended throughout each stroke, so that it provided a more constant force.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

IV.2 Dynamic Modeling

IV.2.i Ergometer with floating pulley

Entire System, FBD:

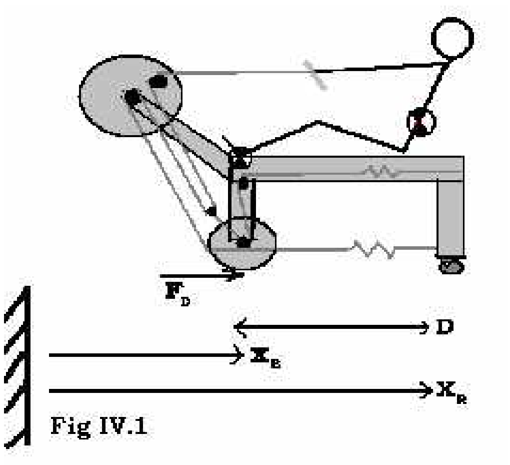


Fig IV.1

$$\sum F_X = \sum m_i a_i$$

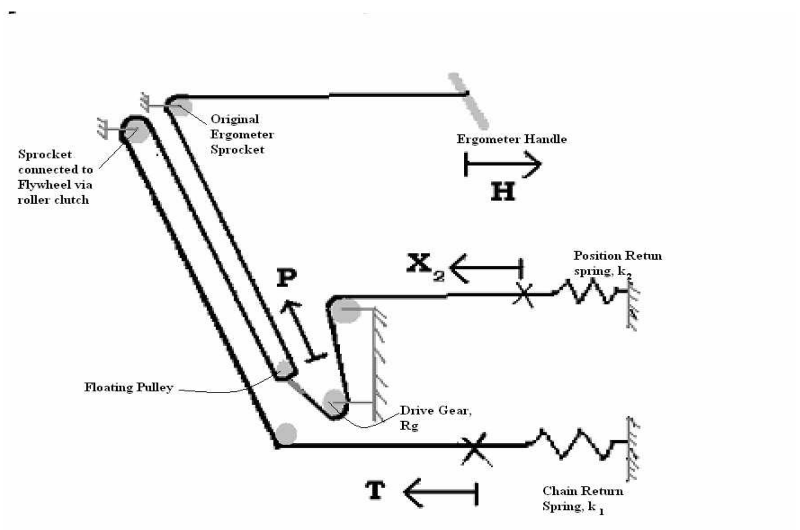
$$F_D = m_R \ddot{x}_R + m_E \ddot{x}_E$$

$$F_D = m_R(\ddot{x}_E + \ddot{D}) + m_E \ddot{x}_E$$

$$F_D = m_R \ddot{D} + (m_E + m_R)\ddot{x}_E \qquad (1)$$

The rower and erg have mass M_R and $M_{E.}$ The rowers center of mass is assumed to be concentrated midway between the butt and shoulder. D is the distance between the rowers feet (rooted to the erg) and CM. It is an independent variable, to be defined later. F_D is a dependent variable, determined by the motion of the rower's body and various components of the erg.

Entire System, Kinematics:


Sprocket connected to Flywheel via roller clutch
Original Ergometer Sprocket
Ergometer Handle
H
P
X_2
Position Return spring, k_2
Floating Pulley
Drive Gear, Rg
T
Chain Return Spring, k_1

$$H = T + 2P$$

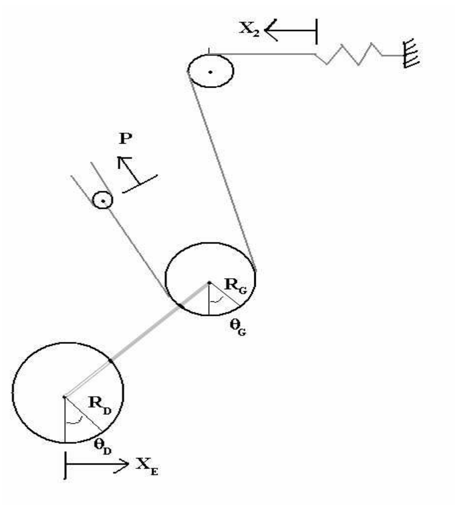
$$\dot{H} = \dot{T} + 2\dot{P}$$

$$\ddot{H} = \ddot{T} + 2\ddot{P} \qquad (2)$$

This watermark does not appear in the registered version - http://www.clicktoconvert.com

All pulleys and sprockets are assumed to be massless and frictionless, and all except the floating pulley are motionless with respect to the erg. The chain connecting the handle to the chain return spring is inextensible, which gives us the above constraint (2).

Drive Train, Kinematics:

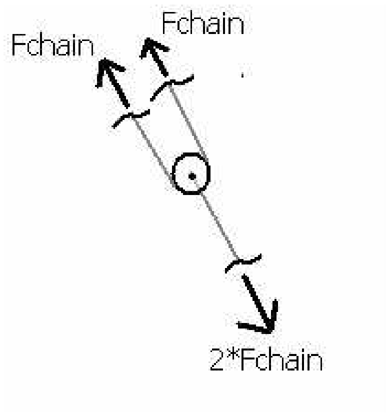


$$P = R_G\theta_G = X_2$$
$$X_E = R_D\theta_D$$
$$\theta_G = \theta_D$$
$$P = \frac{R_G}{R_D} X_E = X_2$$
$$\ddot{P} = \frac{R_G}{R_D}\ddot{X}_E \quad (3)$$

The motion of the floating pulley is directly related to the motion of the ergometer. An inextensible chain connects the floating pulley to the drive gear, which is connected to the drive wheel by a rigid shaft.

Drive Train, FBDs:

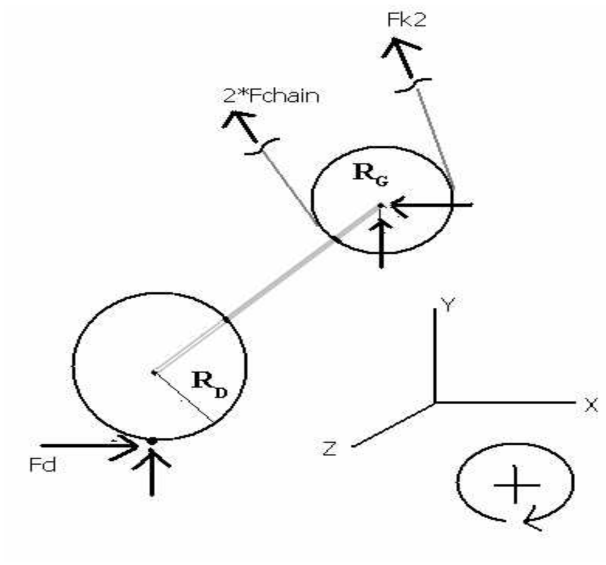
An expression for the force in the chain (F_{chain}) is found by looking at free body diagrams of the floating pulley and the drive train. The floating pulley lacks mass or rotational inertia, thus of 2*F_{chain} is delivered to the drive gear.



$$\sum F = ma = 0$$
$$\sum M = I\alpha = 0$$

This watermark does not appear in the registered version - http://www.clicktoconvert.com

The masses of the drive gear, drive wheel and connecting axel are also neglected.



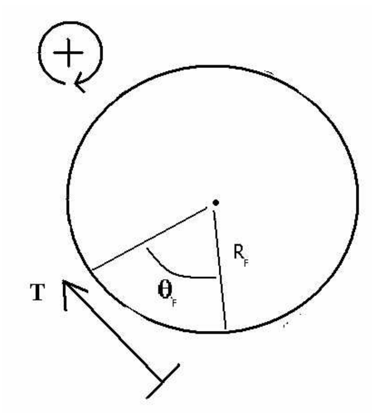
$$\sum M_{\hat{Z}} = I\alpha$$
$$2(F_{chain} - F_{k2})R_G - F_D R_D = 0$$
$$F_{chain} = \left(\frac{R_D}{2R_G}\right)F_D + \frac{1}{2}F_{k2}$$
$$F_{k2} = k_2 x_2 = k_2\left(\frac{R_G}{R_D}\right)x_E$$
$$F_{chain} = \left(\frac{R_D}{2R_G}\right)F_D + k_2\left(\frac{R_G}{2R_D}\right)x_E$$

Substituting equation (1) into this expression gives:

$$F_{chain} = \left(\frac{R_D}{2R_G}\right)\left[m_R\ddot{D} + (m_E + m_R)\ddot{x}_E\right] + \frac{1}{2}F_{k2} \quad (4)$$

Flywheel/Sprocket, Kinematics, (Case I):

There is a clutch between the chain sprocket and the flywheel. When engaged (Case I), the rotation of the (heavy) flywheel and (massless) chain sprocket are directly coupled. Therefore:



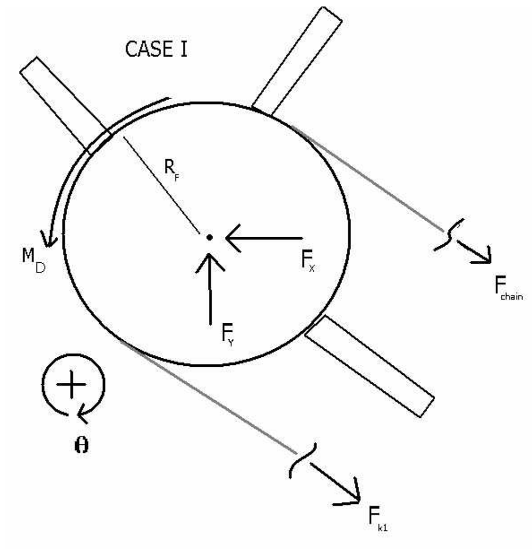
$$T = R_F\theta_F$$
$$\dot{T} = R_F\dot{\theta}_F$$
$$\ddot{T} = R_F\ddot{\theta}_F \quad (5)$$

$$\dot{\theta}_F = \frac{\dot{T}}{R_F} = \frac{\dot{H} - 2\dot{P}}{R_F} = \frac{\dot{H} - 2\left(R_G/R_D\right)\dot{X}_E}{R_F} \quad (6)$$

This watermark does not appear in the registered version - http://www.clicktoconvert.com

Flywheel, FBD, (Case I):

Moments are provided by the chain and by air drag. F_X and F_Y are reaction forces from the erg frame. F_{chain} was defined in equation (4); F_{k1} is from the chain return spring, and is assumed constant.



$$\sum M_{\hat{Z}} = I_F \alpha_F$$
$$(F_{chain} - F_{k1})R_F - M_D = I_F \ddot{\theta}_F$$
$$\underline{where\ldots} M_D = C_D \dot{\theta}^2$$

$$\ddot{\theta}_F = (F_{chain} - F_{k1})\left(\frac{R_F}{I_F}\right) - \frac{C_D \dot{\theta}^2}{I_F} \quad (7)$$

Combining (5), (6) and (7) gives the following expression for $\ddot{T}$:

$$\ddot{T} = R_F \ddot{\theta}_F = (F_{chain} - F_{k1})\left(\frac{R_F^2}{I_F}\right) - \frac{R_F C_D \dot{\theta}^2}{I_F}$$

$$\ddot{T} = (F_{chain} - F_{k1})\left(\frac{R_F^2}{I_F}\right) - \left(\frac{R_F C_D}{I_F}\right) \bullet \left(\frac{\dot{H} - 2\left(R_G \middle/ R_D\right)\dot{X}_E}{R_F}\right)^2$$

Equation (2) then becomes:

$$\ddot{H} = \left\{\left[\left(\frac{R_D}{2R_G}\right)\left[m_R \ddot{D} + (m_E + m_R)\ddot{x}_E\right] + k_2\left(\frac{R_G}{R_D}\right)x_E\right] - F_{k1}\right\}\left(\frac{R_F^2}{I_F}\right) - \ldots$$

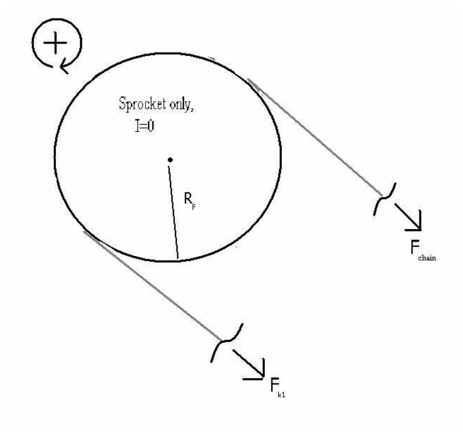
$$\ldots\left(\frac{R_F C_D}{I_F}\right) \bullet \left(\frac{\dot{H} - 2\left(R_G \middle/ R_D\right)\dot{X}_E}{R_F}\right)^2 + 2\left(\frac{R_G}{R_D}\right)\ddot{X}_E$$

This watermark does not appear in the registered version - http://www.clicktoconvert.com

This equation can be solved explicitly in terms for $\ddot{X}_E$, giving our first equation of interest. By substituting equation (4) into equation (7) we get an independent expression for $\ddot{\theta}_F$. Both are presented below, in the summary.

Sprocket, FBD, (Case II):

When the clutch between flywheel and chain sprocket is not engaged, analysis is more simple. The massless chain sprocket rotates freely of the flywheel. A moment diagram of the sprocket shows that $F_{chain} = F_{k1}$.



$$\sum M_{\hat{Z}} = I\alpha = 0$$

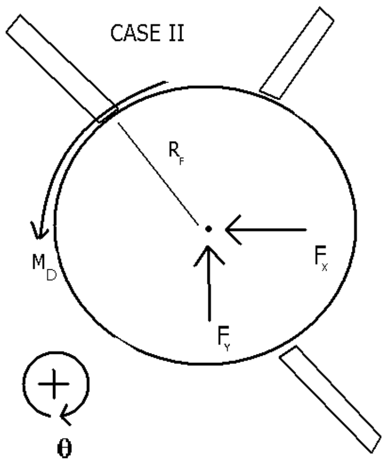
$$\longrightarrow F_{chain} = F_{k1}$$

Substituting in equation (4) yeilds:

$$F_{k1} = F_{chain} = \left(\frac{R_D}{2R_G}\right)\left[m_R\ddot{D} + (m_E + m_R)\ddot{x}_E\right] + \frac{1}{2}F_{k2}$$

$$\ddot{x}_E = \frac{\left(2F_{k1} - k_2\left(\frac{R_G}{R_D}\right)x_E\right)\frac{R_D}{R_G} - m_R\ddot{D}}{(m_E + m_R)}$$

Flywheel, FBD, (Case II):



$$\sum M_{\hat{Z}} = I_F\alpha_F$$

$$-M_D = I_F\ddot{\theta}_F$$

$$\ddot{\theta}_F = -\frac{C_D\dot{\theta}^2}{I_F}$$

This watermark does not appear in the registered version - http://www.clicktoconvert.com

Summary:

CASE I:

$$\ddot{\theta} = \left(\left(\frac{R_D}{2R_G}\right)\left[m_R\ddot{D} + (m_E + m_R)\ddot{x}_E\right] + \frac{1}{2}F_{k2} - F_{k1}\right)\left(\frac{R_F}{I_F}\right) - \left(\frac{R_F C_D}{I_F}\right) \bullet \left(\frac{\dot{H} - 2\left(R_G / R_D\right)\dot{X}_E}{R_F}\right)^2$$

$$\ddot{x}_E = \frac{\ddot{H} - \left\{\left(\left(\frac{R_D}{2R_G}\right)\left[m_R\ddot{D}\right] + \frac{1}{2}k_2\left(\frac{R_G}{R_D}\right)x_E - F_{k1}\right)\left(\frac{R_F{}^2}{I_F}\right) - \left(\frac{R_F C_D}{I_F}\right) \bullet \left(\frac{\dot{H} - 2\left(R_G / R_D\right)\dot{x}_E}{R_F}\right)^2\right\}}{2\left(\frac{R_G}{R_D}\right) + R_F\left(\frac{R_F}{I_F}\right)\left(\frac{R_D}{2R_G}\right)(m_R + m_E)}$$

CASE II:

$$\ddot{\theta}_F = -\frac{C_D\dot{\theta}^2}{I_F}$$

$$\ddot{x}_E = \frac{\left(2F_{k1} - k_2\left(\frac{R_G}{R_D}\right)x_E\right)\frac{R_D}{R_G} - m_R\ddot{D}}{(m_E + m_R)}$$

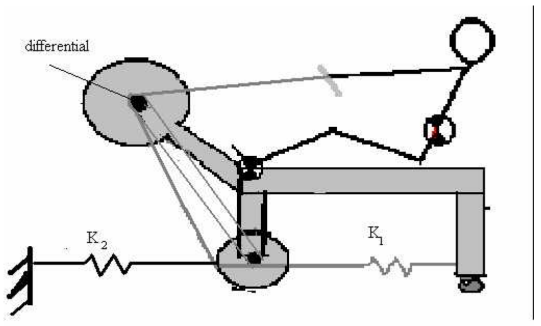
NOTE: Case I and case II are defined by the clutch between sprocket and flywheel. The clutch disengages if the tension in the chain drops to zero. The clutch re-engages when the sprocket accelerates to the speed of the flywheel, and begins to accelerate the flywheel again, creating tension in the chain.

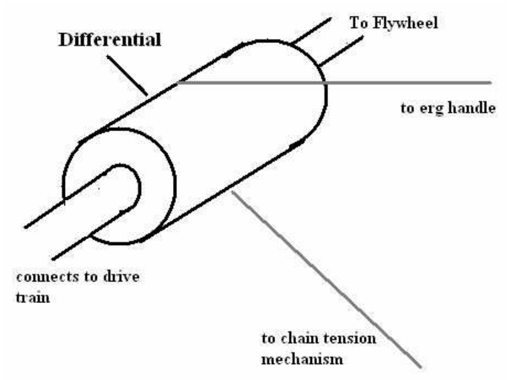
This watermark does not appear in the registered version - http://www.clicktoconvert.com

IV.2.ii Ergometer with differential:

Equations of motion were also studied for an ergometer with an open differential in the drive mechanism. For the derivation, I used an external position return spring

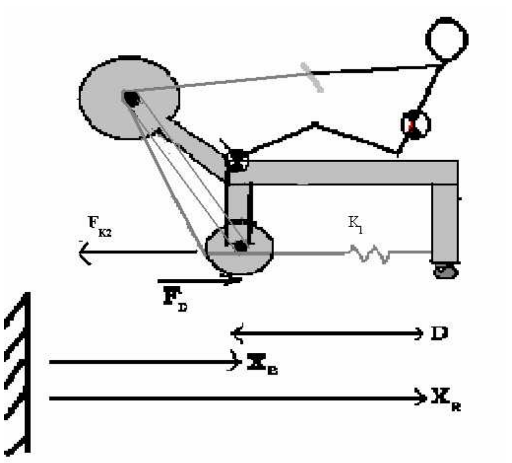
Overview:





As above there are two "cases" and they are determined in the same way. I present the derivation *only* for case I because the equations of motion for case II are nearly identical to those presented above. The derivation for case I was also largely the same, and so the following derivation is not as thoroughly presented.

Entire System, FBD:



$$\sum F_X = \sum m_i a_i$$

$$F_D - F_{k2} = m_R \ddot{x}_R + m_E \ddot{x}_E$$

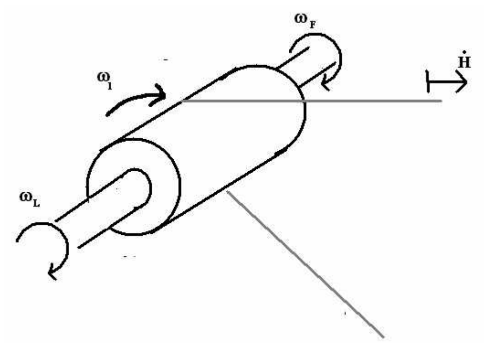
$$F_D = m_R(\ddot{x}_E + \ddot{D}) + m_E \ddot{x}_E + F_{k2}$$

$$F_D = m_R \ddot{D} + (m_E + m_R)\ddot{x}_E + k_2 x_E \text{ (1a)}$$

Differential, Kinematics:

This watermark does not appear in the registered version - http://www.clicktoconvert.com

The rotation of the differential housing is directly linked to the motion of the handle, which gives equation (2a). Using the assumption that an open differential splits the torque equally gives equation (3a). Then assuming that power is conserved, we can relate torque to angular velocity to get (4a).



$$H = \theta_1 \cdot R_1$$

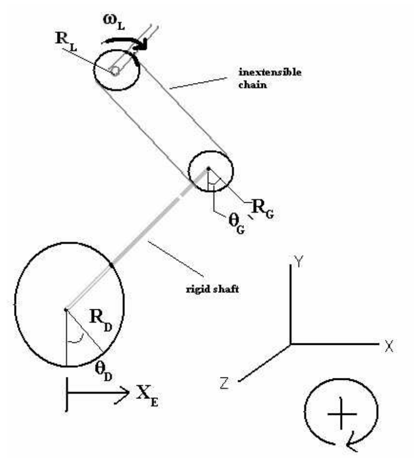
$$\dot{\theta}_1 = \frac{\dot{H}}{R_1} \quad (2a)$$

$$\frac{T_1}{2} = T_F = T_L \quad (3a)$$

Power conserved--> $T_1\dot{\theta}_1 = T_F\dot{\theta}_F + T_L\dot{\theta}_L$

$$\dot{\theta}_1 = \frac{\dot{\theta}_F}{2} + \frac{\dot{\theta}_L}{2} \quad (4a)$$

Drive Train, Kinematics:



$$\begin{aligned} X_E &= \theta_D R_D \\ \theta_D &= \theta_G \end{aligned} \quad (5a)$$

$$\begin{aligned} \theta_L R_L &= \theta_G R_G \\ \theta_G &= \frac{R_L}{R_G}\theta_L \end{aligned} \quad (6a)$$

$$X_E = \frac{R_L}{R_G}\theta_L \cdot R_D$$

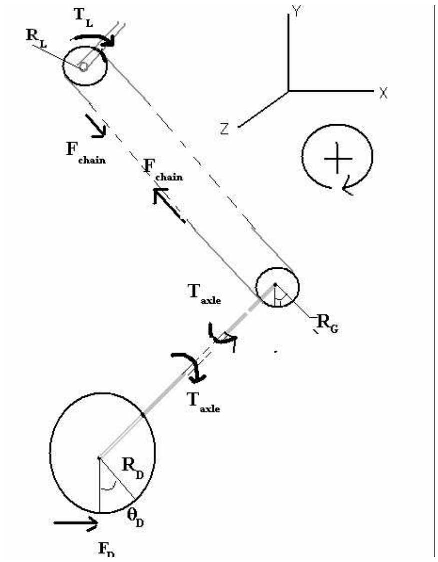
$$\dot{\theta}_L = \frac{R_G}{R_L R_D}\dot{X}_E \quad (7a)$$

This watermark does not appear in the registered version - http://www.clicktoconvert.com

Rigid coupling between the drive gear and wheels give us the first two constraints (5a). From the inextensible chain, we get the next equation (6a). Putting these two equations together and taking the derivative gives (7a).

Drive Train, Force and Moment Diagram:

Torque from the differential (T_L) produces tension in the rigid chain connected to the drive train (F_{chain}). This tension produces a torque on the axle (T_{axle}), which is transferred to the ground in the drive force (F_D). Combining these conditions gives equation 8a.

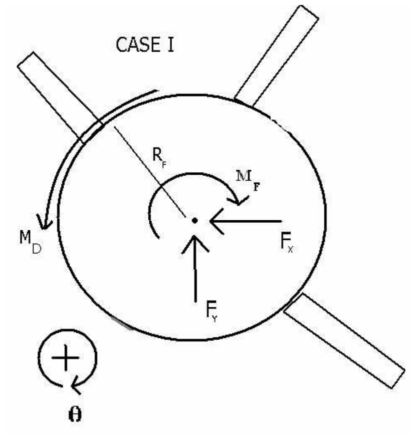


$$T_L = \frac{T_1}{2} = F_{chain} R_L$$

$$F_D R_D = T_{axle} = F_{chain} R_G$$

$$F_D = \left(\frac{T_1}{2}\right)\left(\frac{R_g}{R_D R_L}\right) \quad \text{(8a)}$$

Flywheel, FBD:



$$\sum M_{\hat{z}} = I_F \alpha_F$$

$$T_F - M_D = I_F \ddot{\theta}_F$$

$$\frac{T_1}{2} - C_D \dot{\theta}_F^{\,2} = I_F \ddot{\theta}_F$$

$$\frac{T_1}{2} = C_D \dot{\theta}_F^{\,2} + I_F \ddot{\theta}_F \quad \text{(9a)}$$

This watermark does not appear in the registered version - http://www.clicktoconvert.com

Equating expressions (1a), and (8a) gives

$$F_D = m_R\ddot{D} + (m_E + m_R)\ddot{x}_E + k_2 x_E = \left(\frac{T_1}{2}\right)\left(\frac{R_g}{R_D R_L}\right)$$

(9a) can then be substituted in to yield:

$$F_D = m_R\ddot{D} + (m_E + m_R)\ddot{x}_E + k_2 x_E = \left(C_D\dot{\theta}_F{}^2 + I_F\ddot{\theta}_F\right)\left(\frac{R_g}{R_D R_L}\right) \quad \text{(10a)}$$

rearranging and differentiating (4a) gives

$$\begin{aligned} \theta_F &= 2\theta_1 - \theta_L \\ \dot{\theta}_F &= 2\dot{\theta}_1 - \dot{\theta}_L \qquad \textbf{(11a)} \\ \ddot{\theta}_F &= 2\ddot{\theta}_1 - \ddot{\theta}_L \end{aligned}$$

We can arrive at an explicit expression for the ergometer acceleration by substituting plugging (2a) into (11a), substituting (11a) into (10a) and solving for $\ddot{x}_E$. This yields the following system:

$$\ddot{x}_E = \frac{\left[C_D\left\{2\frac{\dot{H}}{R_1} - \dot{x}_E\left(\frac{R_L}{R_G R_D}\right)\right\}^2 + I_F\left(\frac{\ddot{H}}{R_1}\right)\right]\left(\frac{R_g}{R_D R_L}\right) - m_R\ddot{D} - k_2 x_E}{(m_E + m_R) + \left(\frac{R_g}{R_D R_L}\right)^2 I_F}$$

$$\ddot{\theta}_F = 2\frac{\ddot{H}}{R_1} - \frac{R_G}{R_L R_D}\ddot{x}_E$$

This watermark does not appear in the registered version - http://www.clicktoconvert.com

IV.3: MATLAB simulation

IV.3.i: Setup

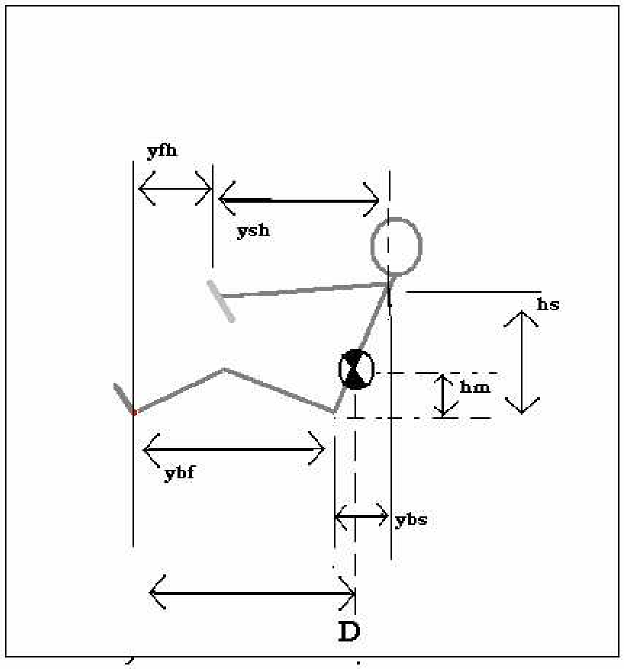
After deriving these equation, the next step was to create a series of MATLAB functions to simulate the ergometer's motions. In the equations above, $\ddot{x}_E$ is a function of x_E and $\dot{x}_E$; $\ddot{\theta}_F$ is a function of $\dot{\theta}_F$. Therefore, I could use one of MATLAB's ODE solvers to integrate the equations of motion. The state variable is defined:

$$x = \left[x_E, \dot{x}_E, \theta_F, \dot{\theta}_F\right]$$

$$\dot{x} = \left[\dot{x}_E, \ddot{x}_E, \dot{\theta}_F, \ddot{\theta}_F\right]$$

$\ddot{x}_E$ and $\ddot{\theta}_F$ also depend on motions of the rower's body segments ($\ddot{D}, \dot{H}, \ddot{H}$).

I used a separate program called joint.m to model the rower. Joint.m was written by Tim Cardahana in 1993 as part of computer simulation of boat rowing. Given the current time, period of the stroke, and body parameters, joint.m calculates the position, velocity and acceleration of a given body segment. The diagram bellows shows the relations between body segments.



D and H are then defined as follows:

$$D = y_{bf} + \left(\frac{h_m}{h_s}\right) y_{bs}$$

$$H = y_{fh} = \left(y_{bf} + y_{bs} - y_{sh}\right)$$

This watermark does not appear in the registered version - http://www.clicktoconvert.com

IV.3.ii: RunErg.m

I created a governing program called "RunErg" to set up the simulation and solve the equations of motion over a set period of time. This program defines needed parameters, initializes variables, calls the ODE solver, stores results, generates plots and animates the motion of rower and ergometer. Joint.m is called both within the ODE solver, and also in a loop that calls joint.m at each time and stores the position of each body segment as well as other dependent variables RunErg.m calculates, stores, and plots the following quantities:

-shoulder/torso length (ybs),	-armlength (ysh)
-seat length(ybf)	-position of the rower's CM relative to the erg (D),
-hand position (yfh),	-chain velocity,
-erg position (x_E),	-erg velocity ($\dot{x}_E$),
-flywheel sprocket velocity,	flywheel angular velocity($\dot{\theta}_F$),

IV.3.iii: RootRunErg.m

On seeing the results of RunErg.m, Dave Cabrera advised that I create a second program that would allow me to get a better look at the motions of ergometer over a single stroke. This led to the production of RootRunErg.m. This program solves the equations of motion for the ergometer over a period of one stroke. A "while" loop in the program causes the simulation to run over again with modified initial conditions until it finds the initial conditions that produce steady, periodic motion, which defined by a stroke for which the position and velocity of the ergometer are the same at the beginning and end of

This watermark does not appear in the registered version - http://www.clicktoconvert.com

the stroke. Finding steady state motion was then essentially a 2-D root finding problem, the function of interest being:

$$F = \begin{Bmatrix} x_{E_F} - x_{E0} \\ \dot{x}_{E_F} - \dot{x}_{E0} \end{Bmatrix} = 0$$

IV.4 Video Data Collection

I recorded a video of myself rowing on the ergometer to use for position and velocity data. I generated a position profile by stepping through the video frame by frame. Using MATLAB, I fit an 11th order curve to the data and derived a velocity curve for the data.

IV.5: Collaboration with Dave Cabrera:

I met with Dave Cabrera, a graduate student who has developed an excellent computer simulation of on-the-water rowing. He was able to incorporate the equations of motion that I derived for the modified ergometer into the framework of his simulation in a program called Rowing.m, and we were able to compare results from the two.

IV.6 Ergometer with differential

I set up Dave’s simulation to solve the equations of motion for the ergometer with differential, in order to see what affect this would have on ergometer performance.

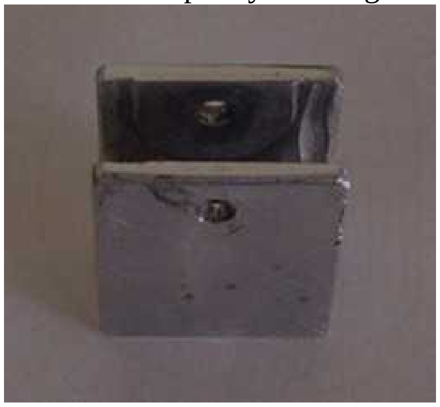
This watermark does not appear in the registered version - http://www.clicktoconvert.com

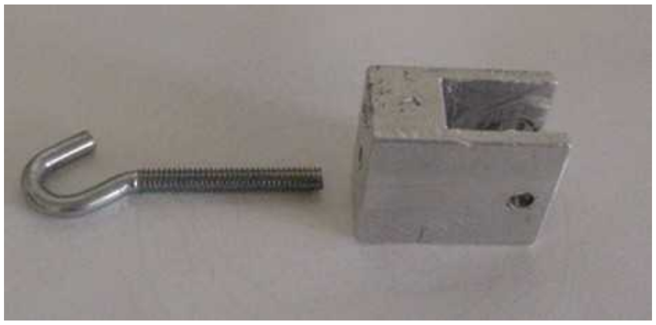
V. Results

V.1: Ergometer Repairs

V.1.i: Floating Pulley

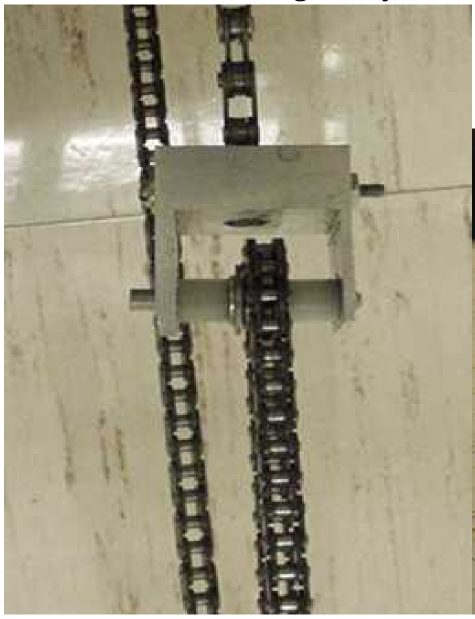
Picture 1: old pulley showing ruts from chain rub

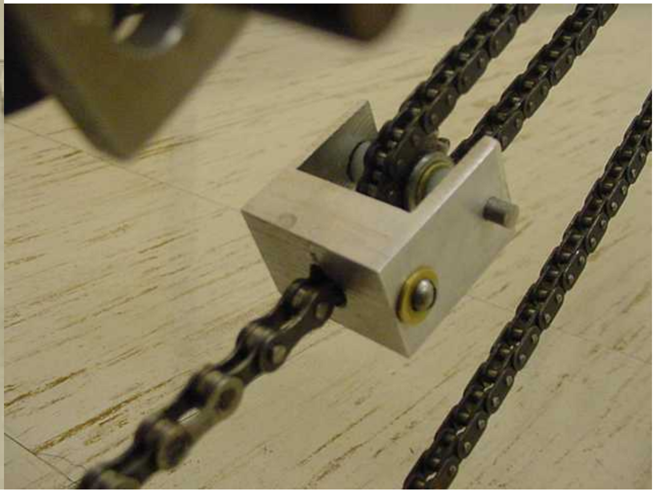




Picture 2: with the hook connector

Picture 3: New Floating Pulley

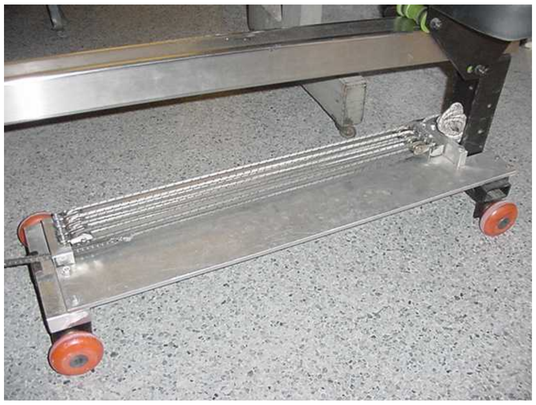




Picture 4: showing direct connection to chain

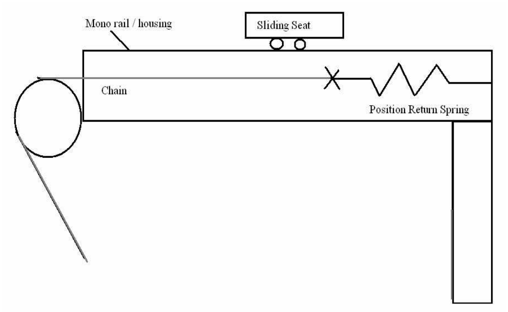
This watermark does not appear in the registered version - http://www.clicktoconvert.com

V.1.ii: Chain-tension Bungees

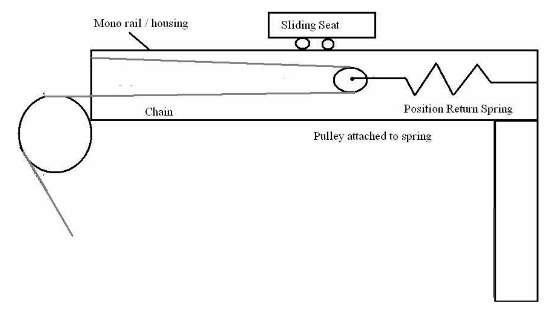

Picture 5: previous bungee assembly

Picture 6: replaced by stock CII assembley

V.1.iii: Position Return Spring



Picture 7: schematic of previous setup



Picture 8: schematic showing modification

V.2: Computer Simulation
Printouts of RunErg.m, RootRunErg.m, and subprograms joint.m, RowErg_D.m, RowErg_R.m, and Animate.m can be found in the appendix. Dave Cabrera's simulation for the ergometer can be accessed online at http://www.tam.cornell.edu/~dcabrera/drew/
Figures and plots are below.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

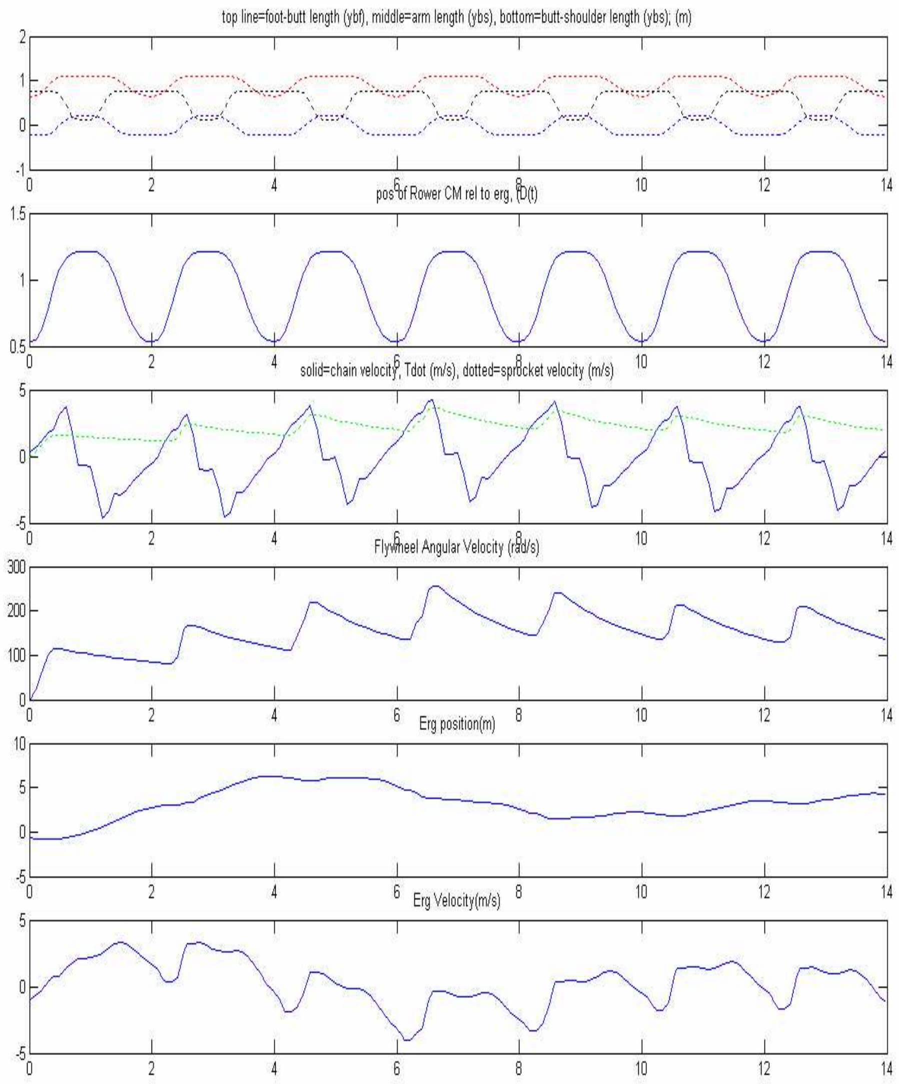


Figure 1: Produced by running RunErg.m over a period of 7 strokes (14 seconds). The top two plots track the lengths of body segments as defined by Joint.m; the next tracks the velocity of the ergometer chain - defined by both the motion of the rower and the ergometer - and the velocity of the chain sprocket - determined by multiplying the flywheel angular velocity, $\dot{\theta}_F$, by the sprocket diameter, R_F. The bottom three plots show the angular velocity of the flywheel and the position and velocity of the ergometer.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

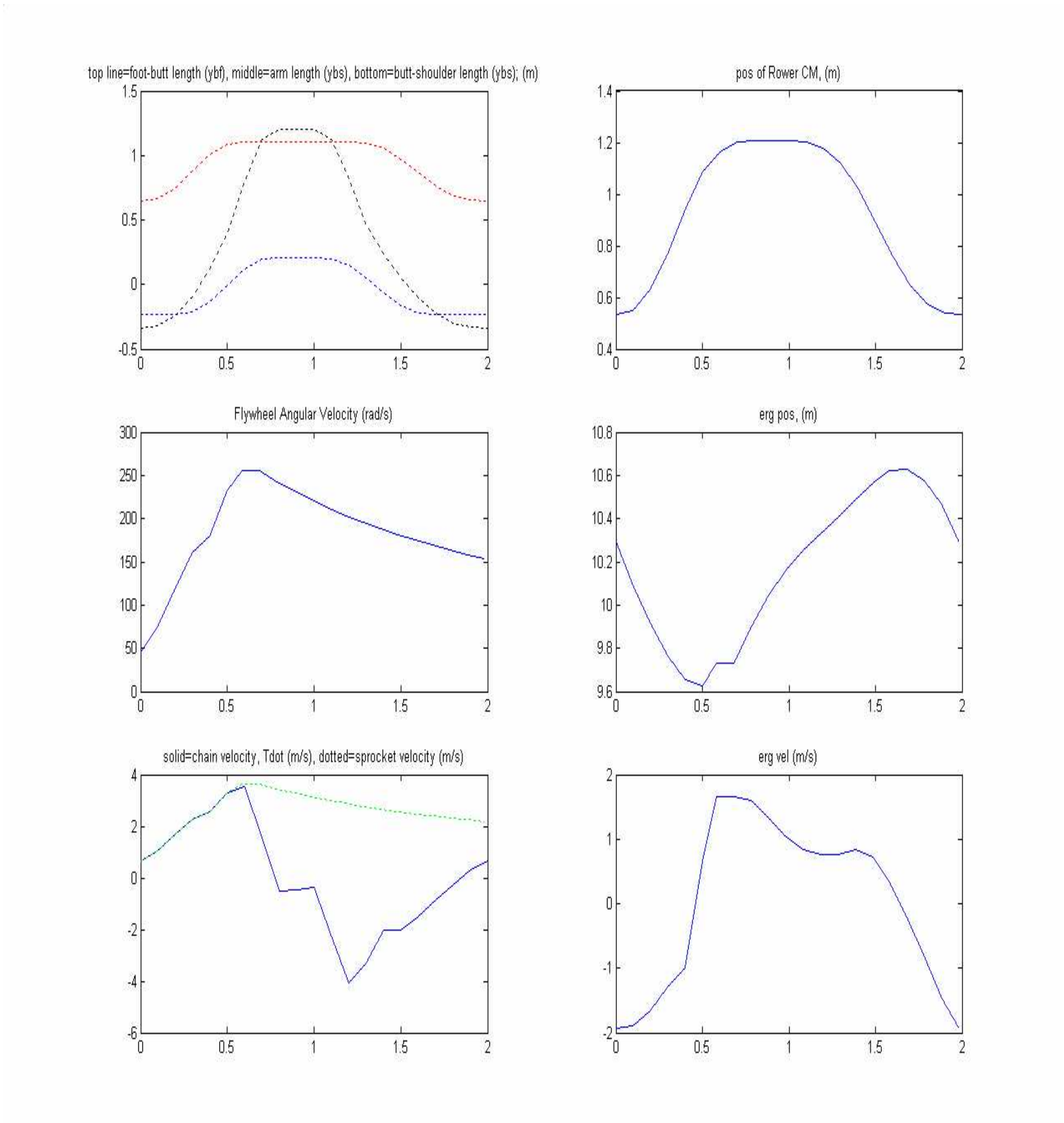


Figure 2: Behavior of the ergometer and rower over the course of one stroke after steady, periodic motion has been reached. Produced by RootRunErg.m after 5 iterations with a tolerance of 10e-4.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

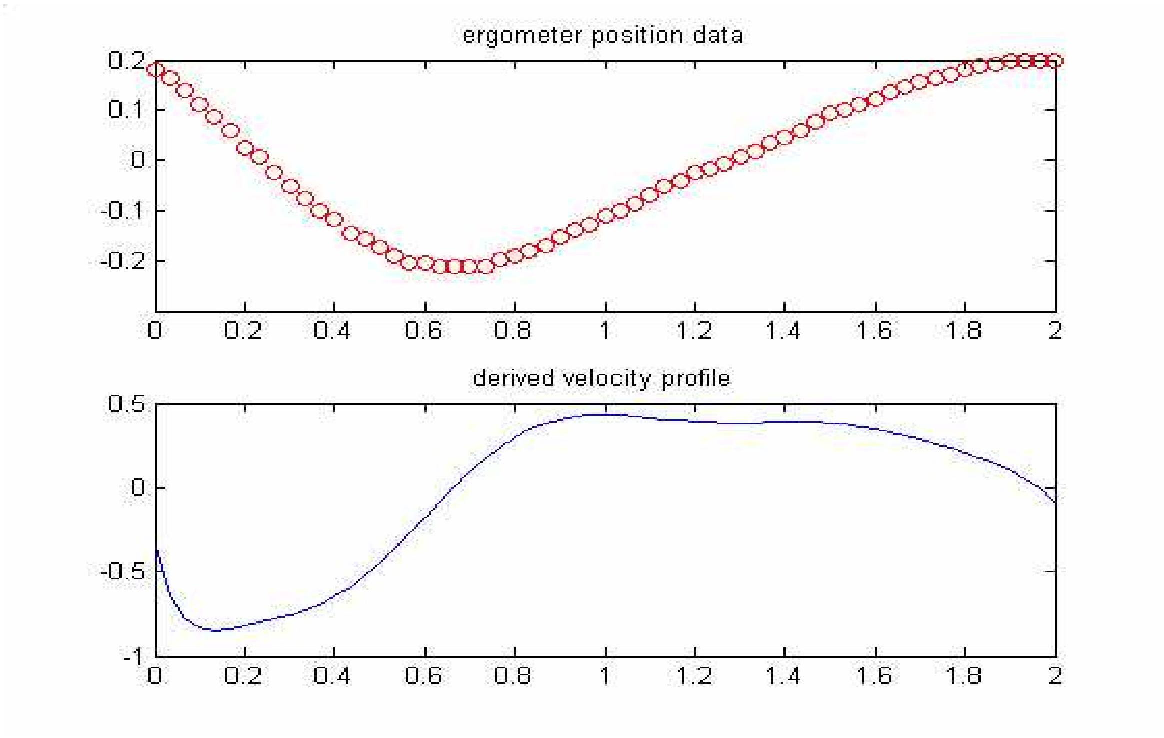


Figure 3: Position and velocity curves obtained from video data.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

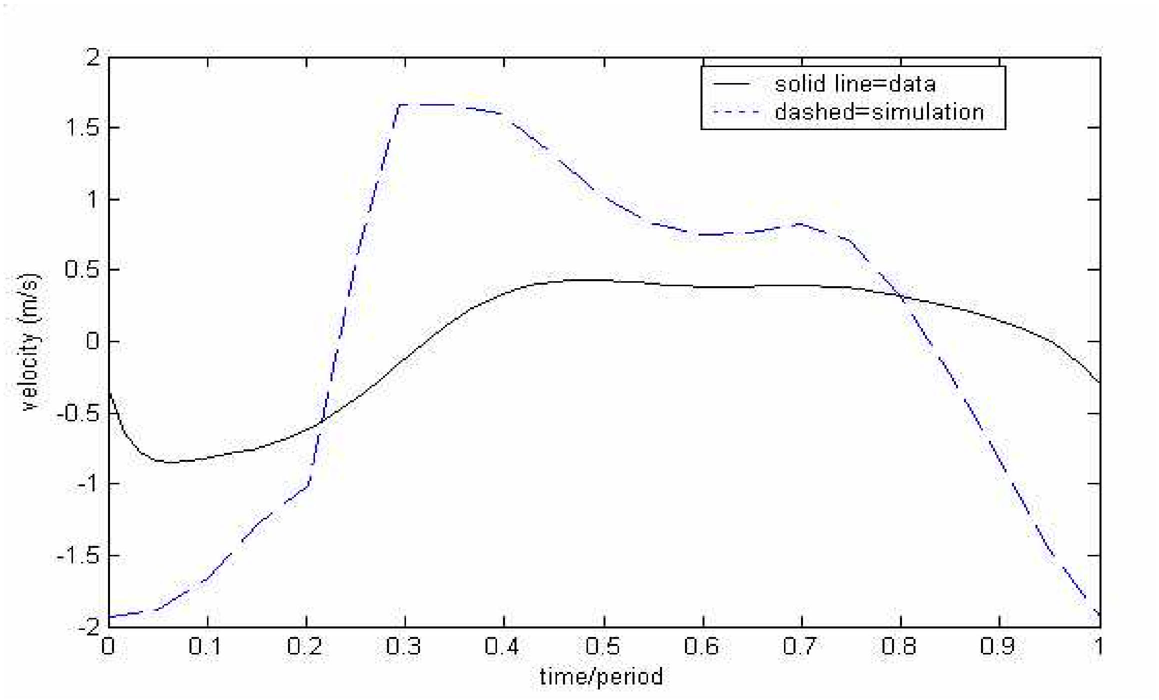


Figure 4: Velocity profiles from RootRunErg.m and video data

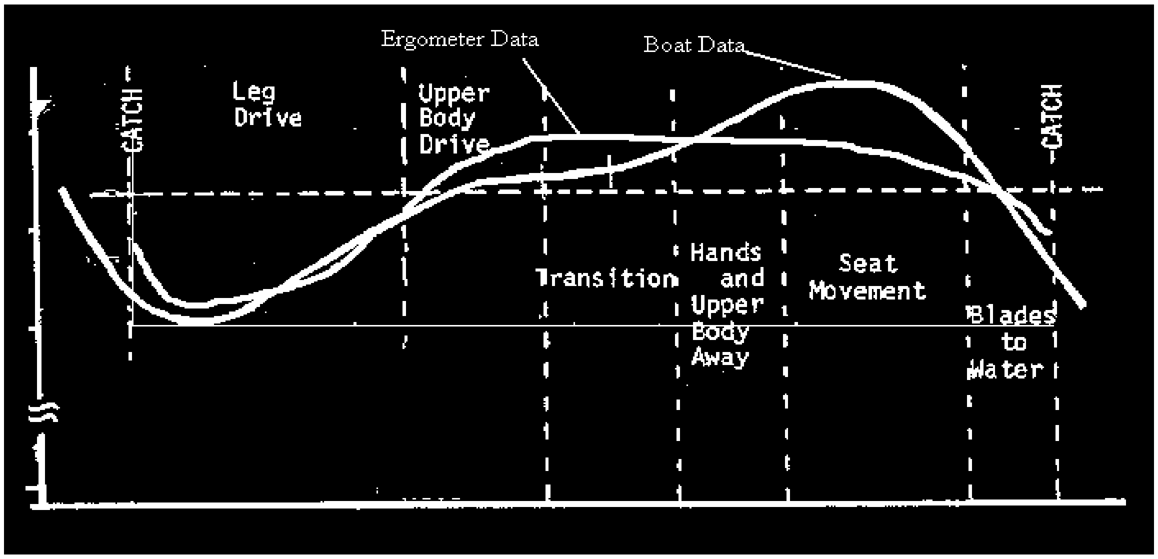


Figure 5: Overlay of the velocity profile from laboratory data onto the velocity curve of a boat recorded by John Ferris.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

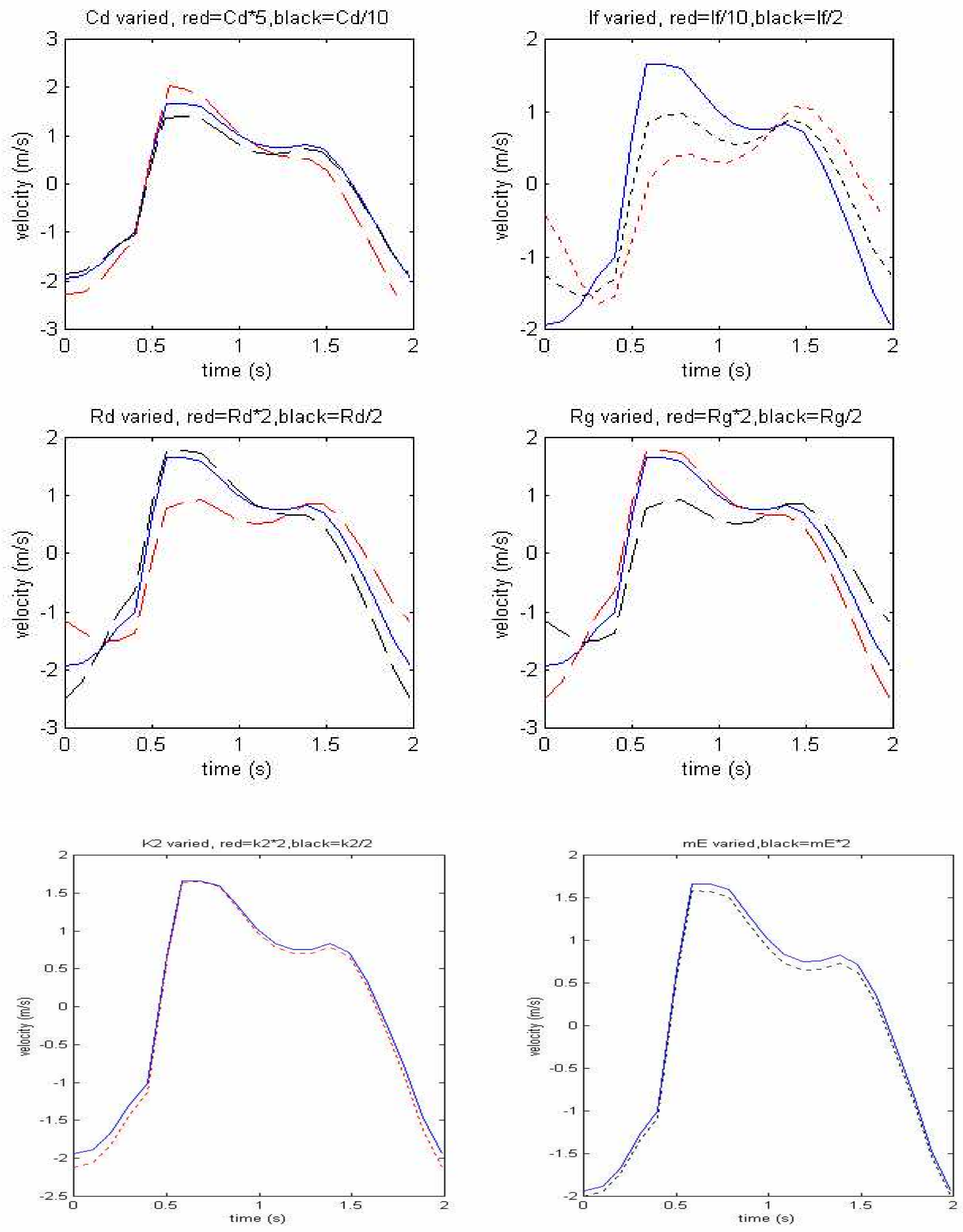


Figure 6: Affect of varying parameters on the steady state velocity profile. Produced by RootrunErg.m

This watermark does not appear in the registered version - http://www.clicktoconvert.com

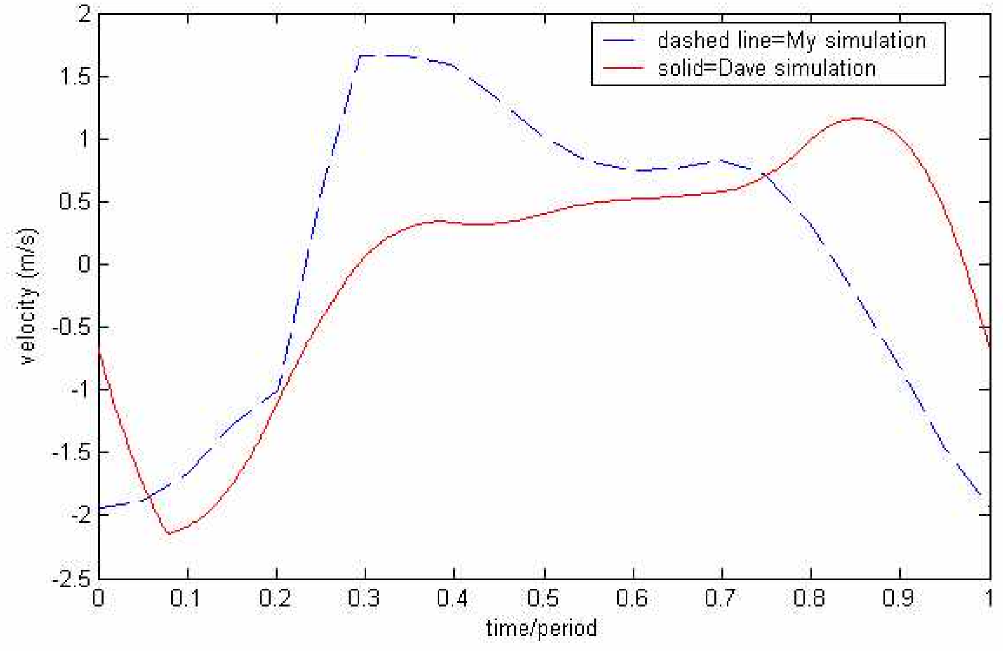


Figure 7 overlays the velocity curves generated by RootRunErg.m and Rowing.m.

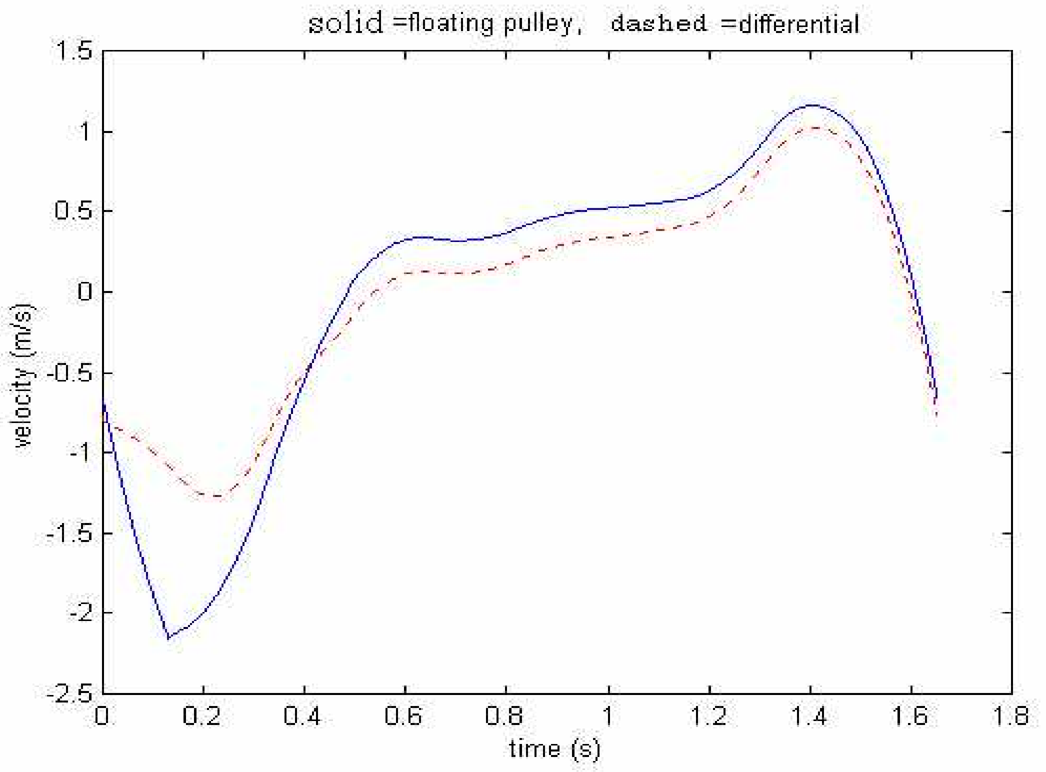


Figure 8: Results of Dave's Simulation (Rowing.m) using Eq's for floating pulley and differential.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

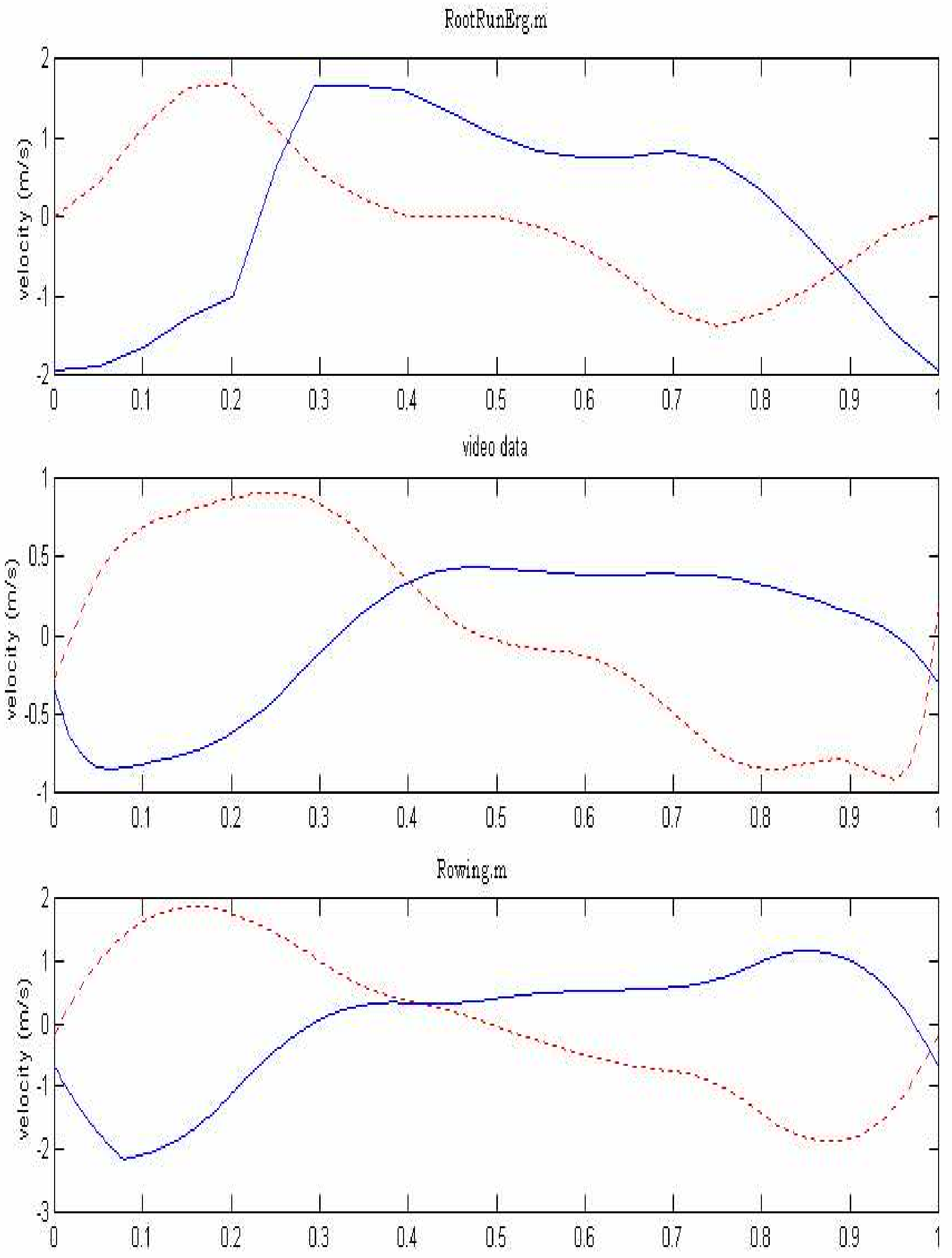


Figure 9: velocity profiles of the ergometer generated by the two simulations and that derived from laboratory data. The solid blue line in each plot is the ergometer velocity, the dashed, red line is the velocity of the rower's center of mass, relative to the ergometer.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

VI.Discussion

VI.1 Ergometer repairs

VI.1.i: Floating Pulley

As expected, the new floating pulley is a vast improvement. The increased width prevents the housing from interfering with the chain and the direct connection to the chain is more durable and easier to align. Since redesigning the floating pulley, I have not had problems with the chain catching, or the attachment breaking, as frequently occurred before.

VI.1.ii: Chain tension bungees

The new chain tension mechanism is also am improvement. The mechanism works consistently and provides an appropriate amount of tension on the handle. Because the mechanism is fully enclosed, there is not a concern about the dirt accumulating in the components. The long, metal monorail does not fit very well where it is currently placed. Modifications could be made so that it fits snugly in place, if this could be done without exposing or altering the mechanism. Although the metal bar may look precariously placed, it actually stays in place better than the previous bungee assembly. In all, it is much more “robust” than the former system. As an added benefit, it makes use of standard Concept II parts instead of requiring custom machining.

VI.1.iii: Position return spring:

The addition of the pulley to the bungees had the effect of making them more compliant. As a result the ergometer displaces farther from its initial resting position in order to reach equilibrium. However, the change in bungee length within one stroke is only half of what it was before. The bungees then provide a smaller range of forces, more closely

This watermark does not appear in the registered version - http://www.clicktoconvert.com

approximating a constant force. Yet, rowing on the ergometer I was unable to feel a significant difference in the ergometers behavior after this modification.

VI.2: Equations of motion

The equations of motion I derived appear to be dynamically correct and accurate for the system. I have revised them many times, and all the main equations have been checked over by both Professor Ruina and Dave Cabrera. One limitation on the model is that it does not include friction on the ergometer wheels. Initially, I was confident that friction could be neglected with little or no consequence on the final result. The lack of friction in the simulation may cause it to predict a larger magnitude for the ergometer's oscillations than actually occurs, but it should not affect the nature of oscillations.

VI.3 Computer Simulation

IV.3.i: Utility and general performance of programs

RunErg.m is useful for tracking the transient behavior of the erg. It allows the user to observe overall behavior and shows the interactions of various parameters. RootRunErg.m allows the user to get a much better look at the ergometer's specific behavior. The user can easily observe the affect of individual parameters on the steady state behavior. Thus it could potentially be very useful to future researchers. Rowing.m could be of similar utility to future researchers, although it makes use of more subprograms and is generally more complex, which could make it daunting to beginners.

VI.3.ii: Accuracy

VI.3.ii.a: Transient Behavior

The plots of erg position and velocity generated by RunErg.m show a region of transient, decaying oscillations. After about 5 strokes (10 seconds) both plots show, steady,

This watermark does not appear in the registered version - http://www.clicktoconvert.com

repeating oscillations. At this point there is still a small-amplitude, large-period oscillation present in the ergometer's motions, which I expect will gradually die out. The ergometer chain, sprocket, and flywheel behave as expected. The three parameters show coupled motion during the drive portion of the stroke, and during the recovery, the speed of the sprocket and flywheel should slowly decay.

VI.3.iii.b: Steady State Behavior

The primary method of judging the accuracy of the simulation is by comparing the steady state velocity profiles. The curve generated by RootRunErg.m should match the curve derived from the video data and, ideally, both curves would the match that of a boat on the water.

In actuality, the shape of the steady state velocity curve generated by RootRunErg.m deviates significantly from the data. There is a difference in both the magnitude and the shape of the velocity curves. The data shows a small, negative velocity at the catch, which decreases through the early part of the stroke, then grows to a small positive value where it stays relatively constant until late in the stroke when it drops again. The simulated velocity starts at a larger, more-negative value, and increases sharply early in the stroke. There is no initial dip as seen in the data. It then rises to a significantly higher positive value and falls through the later half of the stroke.

The fact that the curves do not match is a cause for concern. Lack of friction in the model may account for the larger magnitude of oscillations. The *shape* of the velocity curve is a function of two things: (1)the ergometer dimensions, i.e. mass, gear ratios and spring constants and (2) the motion of the simulated rower. Flaws or inaccuracies in either one of these elements could be the cause of the errant shape of the velocity curve.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

To add to this concern, the velocity curve of the ergometer does not match that of a boat. The ergometer's velocity rises too quickly early in the stroke and then plateaus, where as a boat’s velocity increases slowly, peaking out late in the stroke. Also, the magnitude of the ergometer’s oscillations are too small. This implies that some aspects of the ergometer need to be modified to produce better performance.

In order to examine the affect of each ergometer dimension on the overall performance, I altered many of the parameters in the simulation. Initially, I felt that varying the stiffness of the position return spring (K_2) would be the most significant parameter since several previous students have focused on refining this part of the ergometer. Yet, changing this parameter in the simulation produced almost no difference in the steady state velocity. The most significant parameters seemed to be the flywheel moment of inertia (I_F) and the radius of the erg wheels (R_D). Reducing I_F by a factor of 10 produced a velocity curve that looked much more like that of a boat. Increasing R_D by a factor of 2 had similar consequences.

I would like to conclude that erroneous values for these parameters are causing problems with the simulation, but I am quite confident in the accuracy of these two parameters. I obtained a value for IF from a published paper by Charles Atkinsopht, and R_D was simply measured in lab. Thus I feel that shape of the velocity curve was most negatively impacted by the way that I modeled the motion of the rower in my simulation. This is mentioned further in the next section.

VI.3.iii: Comparison of Simulations

The two simulated velocity curves have similar magnitudes. Both predict a minimum velocity of about -1.5m/s and a maximum of about 1m/s, while the actual velocity curve

This watermark does not appear in the registered version - http://www.clicktoconvert.com

has much smaller magnitude. This supports my thought that lack of friction in the equations of motion causes a difference in magnitude.

The velocity curves produced by the two simulations have greatly different shapes, and both simulated curves deviate significantly from the actual ergometer motions. RootRunErg.m predicts too rapid a rise in velocity at the beginning of the stroke. Rowing.m predicts a pronounced drop in velocity immediately after the catch, followed by a slow increase, then a final jump late in the stroke. Unfortunately, the ergometer's actual behavior lies somewhere in between.

I was surprised to see such a significant difference between the velocity curves produced by the two simulations, since both programs use the same equations of motion. The only major difference in the simulations lies in the way each one defines the rower's body motions. Figure 5 shows that there is noticeable variation in the motions of the rower's CM between the two simulations, and the video data also shows that the actual movement lies somewhere in between. In light of this, it seems that the small differences in the body motions between simulations and data have a great impact on the resulting velocity profiles.

VI.3.iv: Results for ergometer with Differential

Putting the equations of motion for the ergometer with differential into Rowing.m produced a slightly smoother velocity profile. The velocity dips slightly less at the beginning of the strokes and increases slightly less near the end of the stroke, but the motion is largely the same. Initially I would have expected a more significant difference, since several previous students have suggested that the floating pulley be replaced with a differential.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

VI.4.iv: Suggestions for Future Work

- Further refinements of the rowers body motions are needed, since the way these motions are defined seems to have a great impact on the motion of the ergometer. In analyzing my simulation, my biggest regret has been that I settled on Joint.m to model the rower motions. I would suggest that anyone who wants to use this program first create a program to defines the motion of the rower based on data of themselves rowing on the erg.
- Including friction in the simulation of the erg may help to correct the amplitude of the velocity curve generated by the simulation. Conversely, finding ways to eliminate friction from the ergometer may be more beneficial, since, the ergometer currently oscillates less than a it should to accurately simulate a boat.
- The ergometer works consistently, despite its shortcomings. The next big step for this project is to make the ergometer into a practical product. The concept has been refined and reworked enough. It needs to be put to use.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

VII: Conclusion

Three major parts of the ergometer were repaired: floating pulley, the chain tension bungee system and the position return spring. Only minor changes were made to each one, but overall, the ergometer now works much more smoothly and reliably. The equations of motion were derived for the ergometer. By analyzing the dynamics and kinematics of the ergometer, I was able to derive a system of equations that could be entered into a MATLAB ODE solver. These equations are an adequate model, but neglect friction.

Two programs, Runerg.m and RootRunErg.m were developed to simulate the motions of the ergometer. RunErg shows the transient behavior, providing an more general view, while RooRunErg provides a closer look at the steady state motion of the ergometer. By changing the values of several parameters in the simulations, I was able to gain a much more thorough understanding of the ergometer and how each part affects the whole.

The accuracy of my simulation was assessed by comparing the steady state velocity curve generated by RootRunErg.m to a curve derived from video data of the ergometer. This revealed flaws in the simulation. Both the shape and magnitude of the predicted velocity curve were off. I attribute this to the lack of friction in the simulation and unrealistic modeling of the rower’s body motions for the simulation.

Dave Cabrera integrated my equations of motion into Rowing.m, a program he has developed to simulate on the water rowing. The results of his simulation varied from the data and from the results of my simulation. I believe this is largely due to difference

This watermark does not appear in the registered version - http://www.clicktoconvert.com

in the modeling of the rower's body motions. Incorporating the equations of motion for the ergometer with an open differential produced largely the same results.

Overall, the simulations are useful and usable. Flaws still exist, but they do not completely tarnish the results. Future students should benefit from using the simulations. Similarly, the ergometer itself is flawed but functional. Because of the repairs I made this year, the ergometer works much better than it did a year ago. However, the velocity profile of the ergometer does not match that of a boat as well as it did in 1999 or even 1992. This means that the ergometer was actually a better simulator of rowing 12 years ago.

Anyone who takes up this project in the future should study the reports left by previous researchers, ask Professor Ruina a lot of questions, and try to really get a sense of what has already been done on this project, what has not, and what they can feasibly do. Try to contact people who have worked on the project in the past and ask them for help, ask Andy more questions, and work on the project with another person. Most importantly, future students should make sure that they understand the workings of the ergometer very well before they modify the system.

Future work should be directed towards making this project a product, something that can be produced and sold to rowers and coaches who could make use of it. Redesigning the ergometer so that it is more like a *modification* to a standard Concept II ergometer, instead of an entirely new product, would be a huge step.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

VIII: Acknowledgments

This project was funded by the LIFE office at Cornell University (life_eng@cornell.edu). My advisor was Andy Ruina (Ruina@cornell.edu). Most of my research was conducted at Cornell's Human Power, Biomechanics, and Robotics Lab, (306 Kimball Hall, Cornell, 607-255-7108, http://www.tam.cornell.edu/~ruina/hplab/)

I received much guidance and support from Dave Cabrera (dsc16@cornell.edu), who also works in the Human Power Lab. Aside from helping me fix problems in my simulation, he incorporated my work into his simulation of on the water rowing (Rowing.m), which turned out to be very helpful.

Thanks also to Sarah Greer, ME, class of 2005, who let me borrow her MATLAB book and often let me use her computer when mine crashed.

IX: References

Atkinson, William. "Modeling the Fixed-footboard Fanwheel Ergometer." http://www.atkinsopht.com/row/erg/ergabstr.htm, 30 October, 2003.

Cardanha, Timothy. "Development of an Algorithm Simulating Rowing." Cornell University Human Power Lab, 20 May 1993.

Cardanha, Tim, and Naik, Manish. "Improved Ergometer Project." Master of Engineering Project. Cornell University Human Power Lab, 18 Dec 1992.

Concept 2 website. Online. Internet. 2002. Available http://www.concept2.com

Estupifian, Jaime. "Improvements on a Dynamically-Correct Rowing Ergometer." Cornell University Human Power Lab, May 1992.

Kriedler, Scott and Wei, Hong-Fan. "Robust Rowing Machine." Master of Engineering Project. Cornell University Human Power Lab, July 2002.

Nelson, Jacob. "Improved Ergometer Project." Master of Engineering Project. Cornell University Human Power Lab, 14 Dec 1999.

Patrap, Rudra. "Getting Started with MATLAB 5: A Quick Introduction for Scientists and Engineers" New York: Oxford University Press, 1999.

Rowperfect website. "The CARE ROWPERFECT rowing simulator." Online. Internet. 2004. Available http://www.rowperfect.com

Santos, Michel and Mukherjee, Sudip. "An Algomrithm for Rowing Simulation." Cornell University Human Power Lab, 19 May 1996.

This watermark does not appear in the registered version - http://www.clicktoconvert.com

X: Appendices

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%RunErg.m
%performs simulation, using event detection to switch between two ode
solvers
%RowErg_D solve eqs of motion on the drive, RowErg_R, for the recovery

%each integration period is stopped by 'event detection' (ED), when

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
clear
clf
clc



global T timestep simlength
global pbf1 pbf2 pbf3 pbf4 phasebf ybf1 ybf2
global pbs1 pbs2 pbs3 pbs4 phasebs ybs1 ybs2
global psh1 psh2 psh3 psh4 phasesh ysh1 ysh2
global g mR mE hm hs rF rG rD iF Fk1 k2 Cd
global butt shoulder armlength hand Fc Fd

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%defining the motion of the rower
%taken from rowboat.m by Tim Cardanha
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%physical constants
mR=80; mE=15;
hm=0.279; hs=0.558;

%erg dimensions
rF=0.014175;            %radius of sprocket driving the flywheel (1 in)
rG=0.0254;              %radius of drive gear(1 in)
rD=0.1524;              %radius of drive wheel (6 in)
iF=0.1001;              %moment of inertia of flywheel (kg*m^2)
Cd=0.000199;               %coefficient of drag on the flywheel
Fk1=24;                 %force in bottom spring, assumed constant (not
actually constant)
k2=2276;                %spring constant of spring attached to drive
wheel (from 2002 report)
mu=0.6;


%leg timing points
pbf1=0; pbf2=0.3;                                %drive times
pbf3=0.6;pbf4=0.99;                              %recovery times
phasebf=[pbf1 pbf2 pbf3 pbf4];
ybf1=0.649; ybf2=1.105;                                %start and stop
distances

%back timing points
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
pbs1=0.1; pbs2=0.4;                              %drive times
pbs3=0.5;pbs4=0.85;                              %recovery times
phasebs=[pbs1 pbs2 pbs3 pbs4];
ybs1=-0.23; ybs2=0.209;                    %start and stop distances

%arm timing points
psh1=0.2; psh2=0.4;                                    %drive times
psh3=0.5; psh4=0.7;                                  %recovery times
phasesh=[psh1 psh2 psh3 psh4];
ysh1=0.754;ysh2=0.108;                                 %start and stop
distances

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%

%use ODE solver to get erg pos and vel

xE0=-0.7;xEdot0=-1;thetaFly0=0;omegaFly0=0;
z0D= [xE0, xEdot0, thetaFly0, omegaFly0];

%time constants
T=2;                    %period of stroke in seconds
timestep =0.1;
tstartD= 0; tfinal = 6;

tspan=[tstartD:timestep:tfinal];
options = odeset('events','on');

time=[];erg=[];vel=[];thetaF=[];omegaF=[];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%


while tstartD < tfinal

      %call first ODE solver, goes untill end of "drive"
   [t, z]=ode23('RowErg_D', tspan, z0D, options);
   %accumulate output
   time = [time; t];
   erg = [erg; z(:,1)];
   vel = [vel; z(:,2)];
   thetaF = [thetaF; z(:,3)];
      omegaF = [omegaF; z(:,4)];

   %set IC's for recovery
    tstartR=t(end)+timestep ; tspan=[tstartR:timestep:tfinal];
      xE0R = z(end,1); xEdot0R = z(end,2);
      thetaFly0R = z(end,3);omegaFly0R = z(end,4);
      z0R = [xE0R xEdot0R thetaFly0R omegaFly0R ];

   if tstartR<tfinal
      %call second ODE solver,goes untill end of "recovery",
      [t, z]=ode23('RowErg_R', tspan, z0R, options);
   %acculmulate output
   time = [time; t];
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
   erg = [erg; z(:,1)];
   vel = [vel; z(:,2)];
   thetaF = [thetaF; z(:,3)];
      omegaF = [omegaF; z(:,4)];

   %set IC's for next drive
    tstartD=t(end)+timestep  ;tspan=[tstartD:timestep:tfinal];
   xE0D = z(end,1); xEdot0D = z(end,2);
      thetaFly0D =z(end,3);omegaFly0D = z(end,4);
      z0D = [ xE0D xEdot0D thetaFly0D omegaFly0D ];
      else tstartD=tstartR
      end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%loop to generate matrix of body lengths, handle pos.
%some taken from Rowboat.m, some written by me

%initialize vectors
butt=[];
shoulder=[];
hand=[];
armlength=[];
rower=[];

i=1;     %counter
for i=1:length(time)
%for t=0:timestep:simlength                        previous loop counter
   %generate vector of position, vel, and accel at current time,
   %One vector for each joint length
     [ybf,ybfdot,ybfddot]=joint(time(i,1),T,phasebf,ybf1,ybf2);
     [ybs,ybsdot,ybsddot]=joint(time(i,1),T,phasebs,ybs1,ybs2);
     [ysh,yshdot,yshddot]=joint(time(i,1),T,phasesh,ysh1,ysh2);

  %the position of the rower's body segments are defined by joint.m,
  %independent of drive or recovery
  %the location of CM and handle are similarly defined

      yfh    = ybf+ybs-ysh;                           %position of
handle relative to foot/erg
         yfhdot = ybfdot+ybsdot-yshdot;              %vel of handle rel
to foot/erg
      yfhddot= ybfddot+ybsddot-yshddot;       %acceleration of handle:
H-double-dot in notes

      xR    = ybf+(hm/hs)*ybs;               %posotion of rower CM,
relative to erg
      xRdot = ybfdot+(hm/hs)*ybsdot;            %velocity of rower CM,
relative to erg
      xRddot= ybfddot+(hm/hs)*ybsddot;        %accel of rower CM, rel.
to erg, D_double_dot in notes

    %before ending for loop, store all desired quantites in vectors,
update counter variable
    %store body positions for only three strokes
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
        butt(i)=ybf;
        shoulder(i)=ybs;
        hand(i)=yfh;
        handVel(i)=yfhdot;
        armlength(i)=ysh;
        rower(i)=xR;
        CM_vel(i)=xRdot;

        i=i+1;

end
%%%%%%%%%%%%%%%%%%%%%%%%5
%shortime = (time(8*29:10*29));
 %   shortvel = (vel(8*29:10*29));
  %  shortpos = (erg(8*29:10*29));



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%plot results
%body positions
figure(1)
    subplot(5,1,1)
        plot(time',butt)
        title('seat length, ybf')

    subplot(5,1,2)
        plot(time',shoulder)
        title('shoulder length, ybs')

    subplot(5,1,3)
        plot(time',armlength)
        title('arm length, ysh')

    subplot(5,1,4)
        plot(time',rower)
        title('pos of Rower CM, xR')

    subplot(5,1,5)
        plot(time',hand)
        title('hand position, yfh')

%figure(1)
 %   subplot(3,1,1)
  %      plot(time',rower)
   %     title('pos of Rower CM rel to erg, (D(t)')

    %subplot(3,1,2)
%    plot(time',erg')
 %   title('erg pos')

  %  subplot(3,1,3)
   % plot (time',vel')
    %title('Erg vel vs Time')

    figure(2)
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
    title('results of MATLAB simulation, plotted vs time (s)')
    subplot(5,1,1)
    plot(time',rower)
        title('pos of Rower CM rel to erg, (D(t)')

    %plot(time',hand)
    %title('pos of Rower hands rel to erg (m) vs. time (s)')

 subplot(5,1,4)
      plot (time,erg)
    title('Erg position(m)')

    subplot(5,1,5)

    plot(time,vel)
    title('Erg Velocity(m/s)')

    subplot(5,1,2)

     plot(time',(handVel-2*(rG/rD)*vel'), 'b-', time,(omegaF*rF),'g:')
      title('solid=chain velocity, Tdot (m/s), dotted=sprocket velocity
(m/s)')

   subplot(5,1,3)

     plot(time,omegaF,time,zeros(length(omegaF),1),'k-')
    title('Flywheel Angular Velocity (rad/s)')

 %figure(3)
 %subplot(2,1,1)
    %  plot(shortime,shortvel,'b',shortime,zeros(length(shortime),1),'k-
')
  %  title('Erg Velocity')
   % xlabel('time (s)')
    %ylabel('velocity (m/s)')

    %subplot(2,1,2)
     %plot(shortime,shortpos,'b',shortime,zeros(length(shortime),1),'k-
')
 %   title('Erg position')
  %  xlabel('time (s)')
   % ylabel('Position (m/s)')




 %plot(time',omegaF'/100,'m',
time',hand,'k',time',rower,'r',time',(handVel-2*(rG/rD)*vel'), 'b-',
time,(omegaF*rF),'g:');
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
%RootRunErg.m
%This program sets up and runs the ergometer simulation,
%a "root find" method is used to integrate for the IC's that give
stable,
%periodic motion.
%event detection is used to switch between two ode solvers:
%RowErg_D solve eqs of motion on the drive, RowErg_R, for the recovery
%each integration period is stopped by 'event detection' (ED),
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%5
clear
clc

global T timestep simlength
global pbf1 pbf2 pbf3 pbf4 phasebf ybf1 ybf2
global pbs1 pbs2 pbs3 pbs4 phasebs ybs1 ybs2
global psh1 psh2 psh3 psh4 phasesh ysh1 ysh2
global g mR mE hm hs rF rG rD iF Fk1 k2 Cd
global butt shoulder armlength hand Fc Fd


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%physical constants
mR=80; mE=15;
hm=0.279; hs=0.558;




%erg dimensions
rF=0.014175;             %radius of sprocket driving the flywheel (1 in)
rG=0.0254;               %radius of drive gear(1 in)
rD=0.1524;               %radius of drive wheel (6 in)
iF=0.1001;               %moment of inertia of flywheel (kg*m^2)
Cd=0.000199;                %coefficient of drag on the flywheel
Fk1=24;                  %force in chain return spring, assumed constant
(not actually constant)
k2=1138;             %spring constant of position return spring
                     %attached to drive wheel (% was 2276, halved on may
10th);


%stroke time constants
T=2;                     %period of stroke in seconds
timestep =0.1;           %time between iterations
simlength = 20;          %total simulation length

%constants used to define rower body motion in joint.m
%leg timing points
pbf1=0; pbf2=0.3;                                %drive times
pbf3=0.6;pbf4=0.99;                              %recovery times
phasebf=[pbf1 pbf2 pbf3 pbf4];
ybf1=0.649; ybf2=1.105;                                %start and stop
distances

%back timing points
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
pbs1=0.1; pbs2=0.4;                              %drive times
pbs3=0.5;pbs4=0.85;                              %recovery times
phasebs=[pbs1 pbs2 pbs3 pbs4];
ybs1=-0.23; ybs2=0.209;                    %start and stop distances

%arm timing points
psh1=0.2; psh2=0.4;                                    %drive times
psh3=0.5; psh4=0.7;                                  %recovery times
phasesh=[psh1 psh2 psh3 psh4];
ysh1=0.754;ysh2=0.108;                                 %start and stop
distances

%constants for ODE solver and root find
xE0=0;xEdot0=0;thetaFly0=0;omegaFly0=0;                  %IC's
z0D= [xE0, xEdot0, thetaFly0, omegaFly0];                %IC's
tspan=[0:timestep:T];                                    %timespan for
one stroke
epsx = 1;epsv=1; tol = 10^(-4); total = 20; k = 0;      %constants for
root finding method
options = odeset('events','on');                         %tell the ODE
solvers to look for events
time=[];erg=[];vel=[];thetaF=[];omegaF=[];               %initialize
storage vectors

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%loop to generate matrix of body lengths, handle pos.

butt=[]; shoulder=[]; hand=[]; armlength=[]; rower=[];   %storage
vectors
i=1;     %counter

for i=1:length(tspan)
   %generate vector of body part position, vel, and accel at current
time,
   %One vector for each joint length
     [ybf,ybfdot,ybfddot]=joint(tspan(i),T,phasebf,ybf1,ybf2);
     [ybs,ybsdot,ybsddot]=joint(tspan(i),T,phasebs,ybs1,ybs2);
   [ysh,yshdot,yshddot]=joint(tspan(i),T,phasesh,ysh1,ysh2);

 %the position of the rower's body segments are defined by joint.m,
 %independent of drive or recovery
 %the location of CM and handle are similarly defined
      yfh    = ybf+ybs-ysh;                            %position of
handle relative to foot/erg
        yfhdot = ybfdot+ybsdot-yshdot;               %vel of handle rel
to foot/erg
      yfhddot= ybfddot+ybsddot-yshddot;       %acceleration of handle:
H-double-dot in notes

      xR    = ybf+(hm/hs)*ybs;               %posotion of rower CM,
relative to erg
      xRdot = ybfdot+(hm/hs)*ybsdot;             %velocity of rower CM,
relative to erg
      xRddot= ybfddot+(hm/hs)*ybsddot;        %accel of rower CM, rel.
to erg, D_double_dot in notes
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
     %before ending for loop, store all desired quantites in vectors,
update counter variable
    %store body positions for only three strokes
        butt(i)=ybf;
        shoulder(i)=ybs;
         hand(i)=yfh; handVel(i)=yfhdot; handAccel(i)=yfhddot;
        armlength(i)=ysh;
         rower(i)=xR; CM_vel(i)=xRdot; CM_accel(i)=xRddot;
        i=i+1;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%root finding method iterates to find IC's that produce periodic
oscillations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%theory/system/method of guessing IC's
%f=[fx;fv]; x=[x0,v0];
%f1 = fx = xf-x0
%f2 = fv = vf-v0

%taylor approx to find f=0;
%f =[fx]   =  [0]   =   J *  [xx0-x0]
%   [fv]      [0]             [vv0-v0]

%J = [f1x f1v]
%    [f2x f2v]

%[xx0] = [x0]   - J^(-1) * [fx]
%[vv0]   [v0]              [fv]

%f1x = dfx/dx0 = fx(x0+delta,v0) - fx(x0,v0) / delta
%f1v=  dfx/dv0 = fx(x0,v0+delta) - fx(x0,v0) / delta
%f2x=  dfv/dx0 = fv(x0+delta,v0) - fv(x0,v0) / delta
%f2v = dfv/dv0 = fv(x0,v0+delta) - fv(x0,v0) / delta

%fx(x0+delta,v0) = xf(x0+delta,v0) - x0; run ODE solver with
x0=x0+delta
%fv(x0+delta,v0) = vf(x0+delta,v0) - v0;
%fx(x0,v0+delta) = xf(x0,v0+delta) - x0; run ODE solver with
v0=v0+delta
%fv(x0,v0+delta) = vf(x0,v0+delta) - v0;
%fx(x0,v0) = xf(x0,v0) - x0;             run ODE solver with x0=x0;
%fv(x0,v0) = vf(x0,v0) - v0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%end theory, start
code%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

while ((epsx > tol) & (epsv > tol)& (k < total)) % three termination
criteria (third needed?)

     %call first ODE solver, goes untill end of "drive"
      [tD, zD]=ode23('RowErg_D', tspan, z0D, options);
     %set IC's for recovery
      tspan=[(tD(end)+timestep):timestep:T]; z0R = [zD(end,1) zD(end,2)
zD(end,3) zD(end,4)];
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
      %call second ODE solver,goes untill end of "recovery",
       [tR, zR]=ode23('RowErg_R', tspan, z0R, options);
      %accululmulate output
      time = [tD; tR];
      erg = [zD(:,1); zR(:,1)];    vel = [zD(:,2); zR(:,2)];
      thetaF = [zD(:,3); zR(:,3)]; omegaF = [zD(:,4); zR(:,4)];

x0=erg(1);xf=erg(end);
v0=vel(1);vf=vel(end);
fx=xf-x0; fv=vf-v0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%find final position and velocity if IC's are perturbed by small
amount,
%move x0 and v0 by "delta" run ODE solvers for each case
%store the partial derivatives in J, the jacobian matrix
%J = makeJ(x0,v0,delta)   (Should this be a seperate function?)
delta=10^-3;
J=[0 0;0 0];
   %perturb x0 and call ode solver over one stroke
   tspan=[0:timestep:T];
    z0D=[x0+delta v0 0 (handVel(1)-2*(rG/rD)*v0)/rF];
     [tD, zD]=ode23('RowErg_D', tspan, z0D, options);

     tspan=[(tD(end)+timestep):timestep:T]; z0R = [zD(end,1) zD(end,2)
zD(end,3) zD(end,4)];
    [tR, zR]=ode23('RowErg_R', tspan, z0R, options);

    ergx0 = [zD(:,1); zR(:,1)];    velx0 = [zD(:,2); zR(:,2)];
    thetaFx0 = [zD(:,3); zR(:,3)]; omegaFx0 = [zD(:,4); zR(:,4)];
x0x0=ergx0(1);xfx0=ergx0(end);
v0x0=velx0(1);vfx0=velx0(end);

%perturb v0
%need to reset initial conditions for new run
    tspan=[0:timestep:T];
    z0D=[x0 v0+delta 0 (handVel(1)-2*(rG/rD)*(v0+delta))/rF];
     [tD, zD]=ode23('RowErg_D', tspan, z0D, options);

     tspan=[(tD(end)+timestep):timestep:T]; z0R = [zD(end,1) zD(end,2)
zD(end,3) zD(end,4)];

     [tR, zR]=ode23('RowErg_R', tspan, z0R, options);
    ergv0 = [zD(:,1); zR(:,1)];    velv0 = [zD(:,2); zR(:,2)];
    thetaFv0 = [zD(:,3); zR(:,3)]; omegaFv0 = [zD(:,4); zR(:,4)];

x0v0=ergv0(1);xfv0=ergv0(end);
v0v0=velv0(1);vfv0=velv0(end);

 f1x= ( (xfx0-x0x0) - (xf-x0) )/ delta;
 f1v= ( (xfv0-x0v0) - (xf-x0) )/ delta;
 f2x= ( (vfx0-v0x0) - (vf-v0) ) / delta;
 f2v =( (vfv0-v0v0) - (vf-v0) ) / delta;

 J=[f1x f1v;f2x f2v];
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
 JJ=J^-1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%finish it up, iterate for next x0, v0

xx0 = x0 - ( JJ(1,1)*fx+JJ(1,2)*fv );
vv0 = v0 - ( JJ(2,1)*fx+JJ(2,2)*fv );

epsx=abs(xx0-xf);
espv=abs(vv0-vf);

z0D=[xx0 vv0 0 (handVel(1)-2*(rG/rD)*vv0)/rF];
tspan=[0:timestep: T];

k=k+1;

end
%figure(1)

%subplot(3,2,1)
%plot(tspan,butt,'r:',tspan,shoulder,'g:',tspan,hand,'c:')
%title('top line=seat length (ybf), middle=shoulder length (ybs),
top=arm length (ysh); (m)');

%subplot(3,2,2)
%plot(tspan,rower)
%title('pos of Rower CM, (m)');

%subplot(3,2,4)
%plot(time,erg)
%title('erg pos, (m)');

%subplot(3,2,6)
%plot(time,vel);
%title('erg vel (m/s)');


%subplot(3,2,3)
%plot(time,omegaF,time,zeros(length(omegaF),1),'k-')
%title('Flywheel Angular Velocity (rad/s)')

%subplot(3,2,5)
%plot(tspan,(handVel-2*(rG/rD)*vel'), 'b-', time,(omegaF*rF),'g:')
%title('solid=chain velocity, Tdot (m/s), dotted=sprocket velocity
(m/s)')



%figure(5)
%plot(tspan,CM_vel,'r:',time,vel
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
function [y, ydot, yddot] = joint(t,T,phases,y1,y2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% this is the file 'joint.m', called by RunErg.m
% inputs: current time, period, length and phase of joint movement
% outputs: length, vel, and accel of joint
% Aside from this comment, this program is copied exactly from Tim
Cardanha 8/7/93
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%t                       present time
%T                       Period of one stroke
%phases            the four start and stop times
%y1,y2             the extrema of motion

%first get t inside one peiod by subtracting out T
%find present phase, p =t/T
t = rem(t,T); p =t/T;

% first check the still joint cases
if (p<=phases(1))
   y=y1; ydot=0; yddot=0; return;end;
if ( (p>=phases(2))& (p<=phases(3)))
   y=y2; ydot=0; yddot=0; return;end;
if (p>=phases(4))
   y=y1; ydot=0; yddot=0; return;end;

%define variables for stroke motion
if ((p>=phases(1))&(p<=phases(2)))
   p1=phases(1); p2=phases(2);end;

%make recovery motion the same as stroke motion
if ((p>=phases(3))&(p<=phases(4)))
   p1=phases(3);p2=phases(4);
   temp=[y1,y2]; y1=temp(2);y2=temp(1);         %switch y1 and y2
end;

%calculate the scale factors
halfdiff=(y2-y1)/2; yave=(y1+y2)/2;
halfdeltat=T*(p2-p1)/2; pave=(p1+p2)/2; halfdeltap=(p2-p1)/2;

%calculate the normalized position, vel, and acc
s=(p-pave)/halfdeltap;
z      = 15*s/8 - 5*s^3/4 + 3*s^5/8;
zprime = 15/8   -15*s^2/4 + 15*s^4/8;
zpprime= -15*s/2 + 15*s^3/2;

%calculate y and its derivatives
y = yave + z*halfdiff;
ydot = zprime*halfdiff/halfdeltat;
yddot = zpprime*halfdiff/halfdeltat^2;

return;
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
function [value,isterminal,direction] = RowErg_D(t,z,flag)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%this function integrated the equation of motion of the erg during
%the drive, (when the clutch is engaged).
%this is defined by the event chainVel >= sprocket vel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%bring in variable defined in RunErg.m
global T timestep simlength
global pbf1 pbf2 pbf3 pbf4 phasebf ybf1 ybf2
global pbs1 pbs2 pbs3 pbs4 phasebs ybs1 ybs2
global psh1 psh2 psh3 psh4 phasesh ysh1 ysh2
global g mR mE hm hs rF rG rD iF Fk1 k2 Cd Fc Fd
%pull values from input variable z
xE=z(1); xEdot=z(2); thetaFly=z(3); omegaFly=z(4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
%find body segment lengths at this time by calling joint.m
%needed variables defined globaly in RunErg.m
    [ybf,ybfdot,ybfddot]=joint(t,T,phasebf,ybf1,ybf2);
    [ybs,ybsdot,ybsddot]=joint(t,T,phasebs,ybs1,ybs2);
    [ysh,yshdot,yshddot]=joint(t,T,phasesh,ysh1,ysh2);

    %calculate vel accel of handle: H-double-dot in notes
    yfhdot = ybfdot +ybsdot -yshdot;
    yfhddot= ybfddot+ybsddot-yshddot;
    %acceleration of rowers COM rel to erg, D-double-dot in derivation
     xRddot=ybfddot+(hm/hs)*ybsddot;    %if using two seperate mass
points xRddot=0.25*mR*ybsddot+mR*ybfddot

    %equation of motion
      xEddot = ( yfhddot-( (
(rD/(2*rG))*(mR*xRddot)+0.5*(k2*(rG/rD)*xE) -Fk1)*(rF/iF) -
(Cd/iF)*((yfhdot-2*(rG/rD)*xEdot)/rF)^2 )*rF )...
      / ( 2*(rG/rD) + rF*(rF/iF)*(rD/(2*rG))*(mR+mE) );

      alphaFly = (yfhddot-2*(rG/rD)*xEddot)/rF;

if nargin < 3 | isempty(flag),
   value=[xEdot, xEddot, omegaFly, alphaFly]';
else
   switch flag
   case 'events'
        value=
      ((rD/(2*rG))*(mR*xRddot+(mR+mE)*xEddot)+0.5*(k2*(rG/rD)*xE)-Fk1);
      %omegaFly*rF-(yfhdot-2*(rG/rD)*xEdot); %  yfhddot;    %yfhdot-
z(4)*rF;
        isterminal = 1;
        direction = -1;
   otherwise
        error(['Unknown flag ''' flag '''.']);
   end
end
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
function  [value,isterminal,direction]  =  RowErg_R(t,z,flag)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%this function integrated the equation of motion of the erg during
%the drive, (when the clutch is engaged).
%this is defined by the event chainVel >= sprocket vel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%bring in variable defined in RunErg.m
global T timestep simlength
global pbf1 pbf2 pbf3 pbf4 phasebf ybf1 ybf2
global pbs1 pbs2 pbs3 pbs4 phasebs ybs1 ybs2
global psh1 psh2 psh3 psh4 phasesh ysh1 ysh2
global g mR mE hm hs rF rG rD iF Fk1 k2 Cd Fc Fd
%pull values from input variable z
xE=z(1); xEdot=z(2); thetaFly=z(3); omegaFly=z(4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%find body segment lengths at this time by calling joint.m
%needed variables defined globaly in RunErg.m
    [ybf,ybfdot,ybfddot]=joint(t,T,phasebf,ybf1,ybf2);
    [ybs,ybsdot,ybsddot]=joint(t,T,phasebs,ybs1,ybs2);
    [ysh,yshdot,yshddot]=joint(t,T,phasesh,ysh1,ysh2);
    %calculate vel accel of handle: H-double-dot in notes
    yfhdot = ybfdot +ybsdot -yshdot;
    yfhddot= ybfddot+ybsddot-yshddot;
     %acceleration of rowers COM rel to erg, D-double-dot in derivation
     xRddot=ybfddot+(hm/hs)*ybsddot;    %if using two seperate mass
points  xRddot=0.25*mR*ybsddot+mR*ybfddot

   %Equations of motion
   xEddot = ( 2*Fk1-(k2*(rG/rD)*xE)*(rG/rD) - (mR*xRddot) )/(mR+mE);
   alphaFly = -1*(Cd/iF)*omegaFly^2;


if nargin < 3 | isempty(flag),
      value=[xEdot, xEddot, omegaFly, alphaFly]';
else
   switch flag
   case 'events'
      value=      yfhdot-2*(rG/rD)*xEdot-omegaFly*rF;
%((rD/(2*rG))*(0.25*mR*ybsddot+mR*ybfddot+(mR+mE)*xEddot)+0.5*(k2*(rG/r
D)*xE)-Fk1);
      isterminal = 1;
      direction = 1;
   otherwise
      error(['Unknown flag ''' flag '''.']);
   end
end
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
function animate(Xerg)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%



global T timestep simlength
global pbf1 pbf2 pbf3 pbf4 phasebf ybf1 ybf2
global pbs1 pbs2 pbs3 pbs4 phasebs ybs1 ybs2
global psh1 psh2 psh3 psh4 phasesh ysh1 ysh2
global g mR mE hm hs rF rG rD iF Fk1 k2 Cd
global butt shoulder armlength hand

n=simlength/timestep+1;                 %counter, used later

%lenght of body segments
Ll=ybf2*0.5;         %lower leg
Lu=ybf2*0.5;         %upper leg(thigh)
Al=ysh1*0.5;         %forearm
Au=ysh1*0.5;         %upper arm


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
% Define the positions of various body parts, with respect to the foot
% footboard of the erg is the origin (0,0)
% x is horizontal, y is verticle
% B ->butt, S->Shoulder, H->hand, CM->center of mass, K->Knee, E->elbow
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
    xB=butt;
    yB=zeros(1,n);

    xS=butt+shoulder;
    yS=hs*cos(asin(shoulder/hs));

    xH=butt+shoulder-armlength;
    yH=yS;

    xCM=butt+(hm/hs)*shoulder;
    yCM=(hm/hs)*yS;

    xK = Ll*((Ll^2-Lu^2+xB.^2)./(2*Ll.*xB));
    yK = Ll*sin(real(acos((Ll^2-Lu^2+xB.^2)./(2*Ll.*xB))));

    xhe = Al*((Al^2-Au^2+armlength.^2)./(2*Al.*armlength));   %dist
from hand to elbow
    xE=xH+xhe;
    yE = yS-Al*sin(real(acos((Al^2-
Au^2+armlength.^2)./(2*Al.*armlength))));



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
clf
axis([-2 2 -1 1])
axis equal
xlabel('x-position (m)')
ylabel('y-position (m)')
zlabel('z-position (m)')
title('Side view of the motion of the rower and erg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%draw line for body segments in intial position
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lowerleg = line( [0     xK(1)], [0     yK(1)],
'linewidth',3,'erase','xor');
upperleg = line( [xK(1) xB(1)], [yK(1) yB(1)],
'linewidth',3,'erase','xor');
back     = line( [xB(1) xS(1)], [0     yS(1)],
'linewidth',3,'erase','xor');
upperarm = line( [xS(1) xE(1)], [yS(1) yE(1)],
'linewidth',3,'erase','xor');
lowerarm = line( [xE(1) hand(1)],[yE(1)
yS(1)],'linewidth',3,'erase','xor');

frame1   = line( [0     0      ],[0     -0.125
],'linewidth',1,'erase','xor');
frame2   = line( [1     1      ],[0     -0.125
],'linewidth',1,'erase','xor');
frame3   = line( [-1     1      ],[0     0
],'linewidth',1,'erase','xor');
frame4   = line( [-1     1      ],[-0.125     -
0.125],'linewidth',1,'erase','xor');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
%animate rower body w/ respect to the erg
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%-k=2;
%while k<n
  %set(lowerleg,'xdata',[0      xK(k)],'ydata',[0      yK(k)]);
  %set(upperleg,'xdata',[xK(k) xB(k)],'ydata',[yK(k) yB(k)]);
  %set(back,'xdata',     [xB(k) xS(k)],'ydata',[yB(k) yS(k)]);
   %set(upperarm,'xdata',[xS(k) xE(k)],'ydata',[yS(k) yE(k)]);
   %set(lowerarm,'xdata',[xE(k) xH(k)],'ydata',[yE(k) yH(k)]);

    %drawnow
    %k=k+1;
    %pause(0.25)
    %end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%animate the erg and rower
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k=2;
while k<n

  %rower's body
  set(lowerleg,'xdata',[0+Xerg(k)      xK(k)+Xerg(k)],'ydata',[0
yK(k)]);
```

This watermark does not appear in the registered version - http://www.clicktoconvert.com

```
    set(upperleg,'xdata',[xK(k)+Xerg(k)  xB(k)+Xerg(k)],'ydata',[yK(k)
yB(k)]);
   set(back,'xdata',     [xB(k)+Xerg(k)  xS(k)+Xerg(k)],'ydata',[yB(k)
yS(k)]);
    set(upperarm,'xdata',[xS(k)+Xerg(k)  xE(k)+Xerg(k)],'ydata',[yS(k)
yE(k)]);
   set(lowerarm,'xdata',[xE(k)+Xerg(k)  xH(k)+Xerg(k)],'ydata',[yE(k)
yH(k)]);

  %erg frame
  set(frame1,'xdata',  [0+Xerg(k)      0+Xerg(k)    ],'ydata',[0     -
0.125]);
  set(frame2,'xdata',  [1+Xerg(k)      1+Xerg(k)    ],'ydata',[0     -
0.125]);
  set(frame3,'xdata',  [-1+Xerg(k)     1+Xerg(k)    ],'ydata',[0      0
]);
  set(frame4,'xdata',  [-1+Xerg(k)     1+Xerg(k)    ],'ydata',[-0.125 -
0.125]);
  drawnow
    k=k+1;
    pause(0.25)
end
```