

Mathematically Modeling  
A Distance Runner's Race

Bo Waggoner  
Duke University  
Math 160S  
April 2009

Distance running, one of the simplest acts we attempt as humans, is nevertheless incredibly complex. Considering the problem in terms of physics, the body is an amazing, intricate system and even a simple physical approximation of the energy systems of the body must be detailed. Considering a run covering a certain distance in a certain amount of time, we must take into account the effects of fatigue, the energy dynamics of the human body, and external forces, as well as the manner in which these factors interact.

In this paper, this approach – modeling a run over distance and time – will be developed with the use of numerical methods. We will consider middle distance races ranging from 1500 meters to 10000m. In these ranges, both aerobic and anaerobic metabolism play a part, and initial acceleration must be considered. We will attempt to develop a model which relates the energy production of the body to the energy spent during the course of a run, and relate this to the velocity, distance, and time (themselves related). We may then, by allowing the runner to vary power output, see if we can develop a strategy for covering a certain distance in the least amount of time possible – the fastest race.

Let us first note the relationship of distance, time, and velocity:

$$1) \quad D = \int_0^T v(t) dt \quad , \text{ with time } t \text{ varying from } 0 \text{ to } T \text{ and distance } D \text{ predetermined.}$$

One goal is therefore be to find a formula for velocity and integrate it numerically from the start of the race ( $x = 0$ ,  $t = 0$ ) to the end ( $x = D$ ). Newton's second law gives the velocity as a combination of net forces acting upon the body. Net propulsive forces are generated by the runner as she pushes off from the ground. Net resistive forces to forward motion include air resistance.

$$2) \quad \frac{dv}{dt} = F_{propulsive} - F_{resistive} \quad , \text{ with both } F \text{ being force per unit of mass}$$

We will expand both Fs in order to come up with complete formulae for each.

During running, many other forces are at work which are ignored because they do not, in the

long term, contribute to velocity. These include the force of gravity, the forces generated by the runner to resist gravity (including normal force against the ground), and the forces generated by the runner to overcome the inertia of her own limbs and/or friction from moving muscles and limbs. These forces have a net zero effect on position or forward velocity during the course of the event, although their effects may be large in the short term (i.e. over the course of one step). It is understood that some of the energy generated by the runner goes towards producing/resisting these forces, but this energy will not be considered. Instead, we will attempt to set an efficiency coefficient that will convert a certain amount of power produced by the body to the amount of power which goes toward the actual forward motion. This will allow us to obtain a net forward force.

Researchers such as Maronki have attempted to model the race process by assuming a certain constant store of initial energy, which is exhausted over the course of the race. His method, though developed in a complex manner, is too simplistic to accurately reflect the biological processes which contribute to energy production and usage, especially over longer distances during which many different energy systems are used by the body. The classic model proposed by Keller is also too simple in assuming a linear dependence of air resistance forces on velocity.

We assume that the runner is able to vary her effort from zero (standing still) to some maximum value. This effort is one input to the net force function. We also assume that the runner uses this energy as efficiently as she can; thus, more energy used will produce a higher velocity, all other things being equal. However, the relationship is far from linear, because of the way the body produces and uses energy. It is much more efficient to produce energy aerobically; however, past certain levels of effort, anaerobic energy production is necessary to increase velocity. This will be explored in the function for power; first, we consider power and efficiency together combining to form force. Propulsive force per unit mass is a function of several variables:

3) 
$$F_{propulsive} = \frac{P_{out} * t * E_{ff}}{m * d}$$
, where Eff is efficiency (the percent of power produced that goes toward forward motion), Pout is power produced, m is mass, t is time, and d is distance traveled.

We will approximate Eff by finding that value which provides the most realistic fit, as compared to actual times over various distances.

The other component of change in velocity (equation 2) consists of resistive forces. Assuming no slipping upon footstrike, the only external resisting force on a runner is air resistance. The formula for resistive forces therefore becomes the formula for drag in air per unit mass:

4) 
$$F_{resistive} = P_2(v) = -\frac{1}{2} \rho A C_d v^2 \approx -.01 v^2$$
,  $\rho$  is air density, A is area, and Cd is the drag coefficient.

This approximation is found in, for example, Arsac and Locatelli as well as Prempere et. al. and is dependent on the characteristics of the individual runner.

Now we must determine power output, as a function of effort and fatigue. Running, or any muscular activity, is necessarily a combination of these two factors. A fatigued runner giving maximal effort may not achieve as much propulsive force or as high a velocity as a fresh runner giving submaximal effort. Put more simply, someone who is exhausted but going all-out may not run as quickly as a runner who is not tired but giving less than full effort.

Many models, such as Keller's or Maronski's, predict a competitor running out of energy shortly before finishing and completing the race with a deceleration from that point to the finish. This is perhaps consistent with the assumption that humans have a finite supply of energy which they are physically capable of exhausting, but not with observed results – elite runners generally maintain their

maximum velocity all the way through the finish line. It is hoped that our model of propulsive force, in which a runner may be required to slow down by fatigue but will not actually “run out” of energy (energy being ATP molecules produced primarily by glucose), will provide some understanding of this phenomenon. This assumption works well for distances under 18.6 miles, for which muscle stores of glycogen are sufficient for competition (Martin and Coe, 71).

It is necessary to consider the methods whereby the body produces energy and accumulates fatigue in order to resolve this issue. Muscles use adenosine triphosphate, ATP, to do work, and this will be considered the limiting agent. ATP is obtained for use in exercise in several ways:

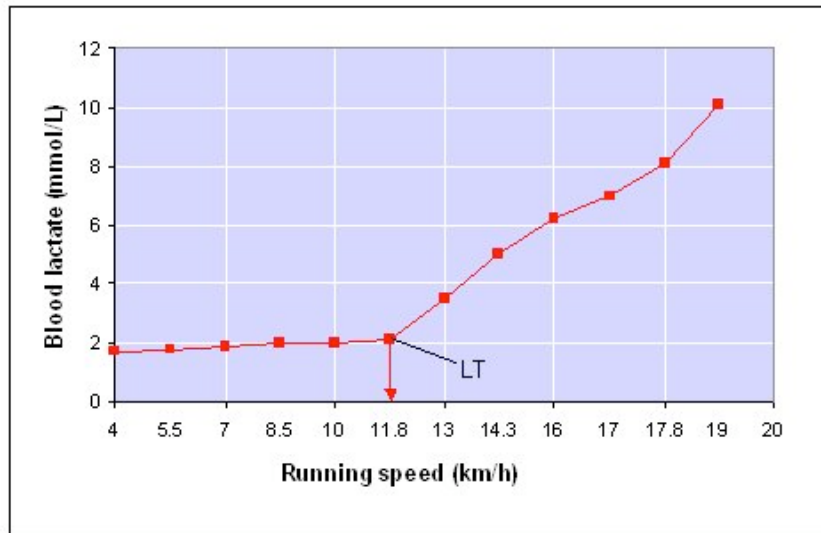
- a. A small amount of ATP is always present in the muscles, along with a supply of creatine phosphate, which is used to generate ATP. This is the primary energy source for the first 20 seconds or so of running (Martin and Coe, 61).
- b. The aerobic breakdown of fuels, including both glucose and fatty acids. This involves the use of oxygen from the bloodstream to break down glucose or fatty acid molecules into pyruvate, then utilizing mitochondria to produce ATP. One molecule of glucose, by this method, yields 36 molecules of ATP (Martin and Coe).
- c. Anaerobic breakdown of glucose. This process begins in the same manner as method b, but if there is insufficient oxygen available, the pyruvate will be broken down into lactic acid. One molecule of glucose will then yield 2 ATP (Martin and Coe).

Method c is obviously less efficient in terms of the ratio of glucose supply used to ATP produced.

Meanwhile, method c produces lactic acid as a waste product. Lactic acid raises the acidity of the bloodstream, and contributes to fatigue – in fact, blood lactate levels may be considered an indicator of fatigue.

As runners increase their effort, they eventually maximize their reliance upon aerobic

metabolism. As they begin to rely upon anaerobic metabolism, they reach a point after which lactic acid is accumulated in the bloodstream faster than it can be eliminated. This will eventually force the runner to slow down. There is in fact an exact effort level and pace for which this happens for every individual, as can be seen by examining a graph relating lactic acid levels to running pace:



<http://www.sport-fitness-advisor.com/anaerobictreshold.html>

Therefore, while running below this certain pace, fatigue accumulates due to almost exclusively aerobic metabolism. Above this pace, fatigue due to aerobic metabolism still accumulates, but now the anaerobic system is in play in addition; lactic acid causes increasingly heightened fatigue.

So we would like to develop a model of power output which is a function of time, effort, and fatigue, in the following abstract manner ( $P_{out}$  is in Watts):

$$5) \quad P_{out} = \begin{cases} C_{eff} * P_{cp} & , 0 \leq t \leq T_{cp} \\ C_{eff} * P_{aer} & , t > T_{cp} , C_{eff} \leq C_{LT} \\ C_{LT} * P_{aer} + (C_{eff} - C_{LT}) * P_{anaer} & , t > T_{cp} , C_{eff} > C_{LT} \end{cases} * C_{fat}$$

Here,  $C_{eff}$  is a coefficient of effort varying from zero (no effort) to one (maximal effort). The  $P$ 's represent maximum power output available from the creatine-phosphate, aerobic, and anaerobic energy

systems respectively. For the first  $T_{cp}$  seconds of a run, no matter the effort, all power comes from creatine phosphate and stored ATP. After that point, when the runner uses less than CLT effort, all power comes from the aerobic energy system. If the runner uses an effort higher than CLT, the runner will use all power available from the aerobic energy system ( $CLT * P_{aer}$ ) plus a certain amount from the anaerobic system.

$C_{fat}$  is a coefficient of fatigue, which will be a function of the total accumulated fatigue. It is the form of this function, and of how fatigue will be accumulated, that we now examine.

Fatigue is one of the more intriguing concepts in muscular biology. To illustrate the variety of theories on the causes of fatigue, the major competing theories will be mentioned here (this paragraph is entirely skippable). There are many different models of fatigue, based on entirely different principles. Some attribute fatigue to physical breakdown of muscle tissue; some to loss of necessary nutrients/electrolytes such as calcium. Some believe that fatigue stems mainly from lack of oxygen to muscles; some say that it is caused by the inability to eliminate waste products such as lactic acid quickly enough (though some now believe that lactic acid, far from being harmful, is a necessary and helpful byproduct). It has been argued that fatigue is caused mainly by a decrease in functionality of neurotransmitters and the number of muscle units that can be recruited by a nerve impulse over time. Finally, a recently proposed model states that fatigue is mainly a concept of the human brain, which gathers data as to how hard and for how long the body has been working and sets limits on what it will allow the body to continue to do, in order to prevent too much actual physical fatigue (in the form of bodily harm) to occur. And of course, some argue that all of these models must be considered to some extent (Abbiss and Laursen).

This is an incredible variety, which makes it daunting to pick one model for our current task. However, it might also be noted that the model of fatigue which yields the most realistic results – for

example, one in which a runner sprints hard at the end of the race rather than running slower and slower as the race progresses – might be more plausible than a model of fatigue which does not coincide with actual results. However, it is admitted that we are unlikely to be able to obtain such results with parameters such as “cognizance of the nearness of the finish line.”

Luckily, all of these theories are designed to cover a real-world, measurable phenomenon, and researchers such as Ding et. al. have in fact studied the relation of muscle fatigue (in terms of how much work has been done) to power output, showing an approximately decreasing exponential relationship.

How then, can we model power output while taking fatigue into account? Peronnet and Thibault note, however, that capacity to do work decreases steadily over the course of the race and reflect this fact with an exponential relationship of their own. They represent power output due to aerobic energy sources over the course of a race as shown in 6a, and due to anaerobic sources as in 6b (these equations have been modified to reflect instantaneous power rather than total work):

6) a) 
$$P_{aer} = BMR + (MAP - BMR) * (1 - e^{-\frac{t}{k_1}})$$
 , with BMR being Basal Metabolic Rate, MAP being Maximal Aerobic Power, and k1 being 30 seconds.

b) 
$$P_{anaer} = A(1 - e^{-\frac{t}{k_2}})$$
 , with A being total anaerobic reserves of energy, and k2 being set at 20 seconds.

Peronnet and Thibault also make an important distinction for races lasting longer than 420 seconds, based on support from several researchers. They modify these equations further in order to take this into account.

However, we wish to allow the runner to vary his/her effort, and must assume that fatigue will be dependent upon this effort. So this particular model, which does not account for adjustment of

effort, will not be entirely useful. We will first therefore set up the manner in which we increase, or time-step, fatigue as a function of effort. Then we will attempt to incorporate fatigue into our power output formulae in an exponential manner.

To increase fatigue as a function of effort, we will use this relationship:

$$7) \quad \frac{dFat}{dt}(C_{eff}) = \begin{cases} K_{fat, aer} * (C_{eff})^2 & , C_{eff} < C_{LT} \\ K_{fat, anaer} * (C_{eff} + (C_{eff} - C_{fat})^2)^2 & , else \end{cases}$$

The Ks are constants and Ceff is the coefficient of effort, which can vary from 0 to 1. So we assume that fatigue depends quadratically upon effort, and that fatigue due to anaerobic metabolism is greater than that due to aerobic.

Each time step, we will increase the amount of Fatigue, Fat, using this formula. Incorporating our value for fatigue into our formula for the coefficient of fatigue in Equation 5 will then be relatively easy:

$$8) \quad C_{fat} = K_{fat, mult} * e^{\frac{-Fat}{K_{fat}}} , \text{ the Ks constants.}$$

We now have all of the pieces in place to describe the change in velocity over time. Here is the complete set of functions required:

$$\frac{dv}{dt} = F_{propulsive} - F_{resistive}$$

$$F_{propulsive} = \frac{P_{out} * E_{ff}}{m * v} \quad F_{resistive} = -.01 v^2$$

$$9) \quad P_{out} = \left\{ \begin{array}{ll} C_{eff} * P_{cp} & , 0 \leq t \leq T_{cp} \\ C_{eff} * P_{aer} & , t > T_{cp} , C_{eff} \leq C_{LT} \\ C_{LT} * P_{aer} + (C_{eff} - C_{LT}) * P_{anaer} & , t > T_{cp} , C_{eff} > C_{LT} \end{array} \right\} * C_{fat}$$

$$C_{fat} = K_{fat,mult} * e^{\frac{-Fat}{K_{fat}}}$$

For which given constants are: Eff (efficiency); m (mass); Pcp, Paer, Panaer (max power available from each energy system); T<sub>cp</sub> (time at which energy source switches from creatine phosphate); CLT (effort level at which anaerobic metabolism becomes a factor and lactic acid accumulates); K<sub>fat,mult</sub>, K<sub>fat</sub> (coefficients for fatigue calculation).

We will use this formula to numerically integrate the following initial value problem:

$$10) \quad \begin{array}{l} D = \int_0^T v dt \\ v(0) = 0 \\ x(0) = 0 \\ Fat(0) = 0 \\ C_{eff} = ? \end{array} \quad \text{where all variables are expressed functions of time } t.$$

The coefficient of effort (Ceff) is up to the runner – how much effort to give at a certain time. We may for now set it to a constant throughout, and will later explore ways in which to find the optimal Ceff.

Each iteration, we will time step by adding a given amount to time t. We will determine the next values of our variables by the following calculations:

$$11) \quad \begin{array}{l} \frac{dv}{dt} = \text{as described in equations 9} \\ \frac{dx}{dt} = v \\ \frac{d Fat}{dt} = \text{as described in equation 7} \end{array}$$

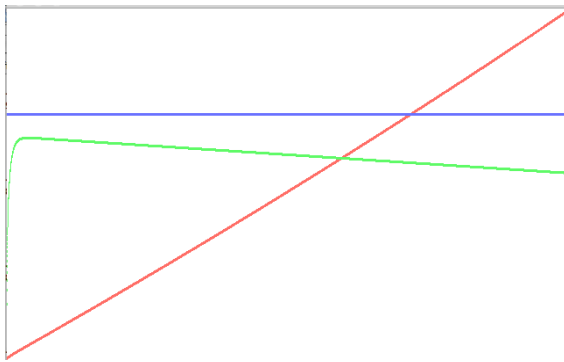
Integration continues until x reaches the desired distance D.

Any desired numerical integration method may be used. It was found that Forward Euler, with a time step of .001 seconds, gave sufficient accuracy (dividing the time step by 10 changed the final time by .0006% in the mile, .0014% in the 5000m, and .00013% in the 10000m).

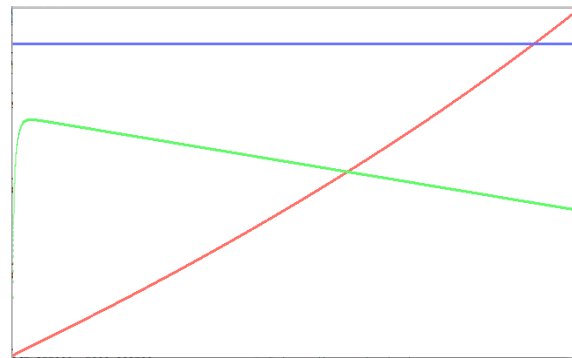
Here are some example graphs generated. The x-axis is distance, the red line is t(time), the blue line is Ceff (effort), and the green line is v (velocity).

5000m with constant effort  $C_{eff} = .7$ :

5000m with constant effort  $C_{eff} = .9$ :



Time: 14:25.86



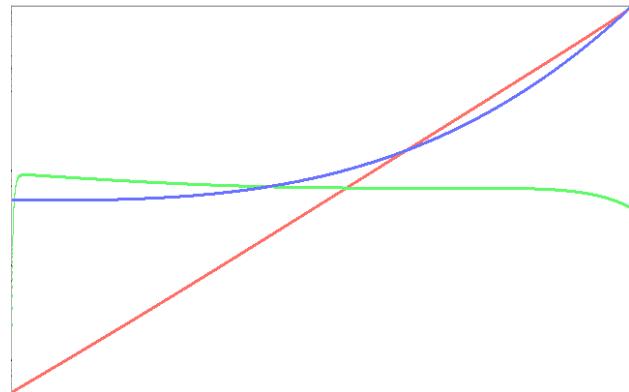
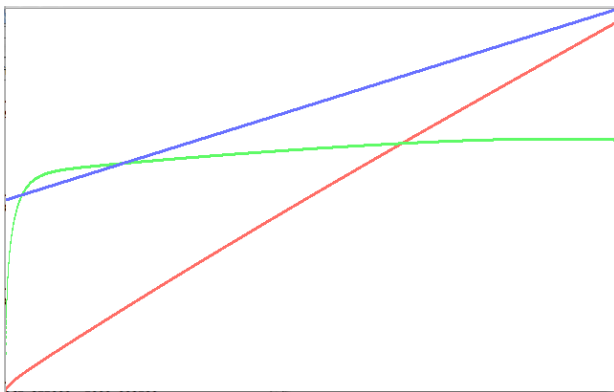
Time: 15:23.22

Here, CLT has been set at .75. So we see that, in the second example, giving too much effort caused a buildup in fatigue, leading to a worse time. (The velocity decreases much more over the course of the race although it begins higher.)

Two more examples are now given. One is from a mile race, run with effort linearly increasing from .5 to 1. The other is from a 10k in which effort increases cubically from .5 to 1.

Mile: 4:22.43

10k: 31:17.30



In the left graph, velocity (green) increases throughout; in the right, it falls although effort increases.

Now we have the tools with which to address the initial question: How should a runner run in order to minimize the time of the race? We can find out by allowing the runner to vary his or her effort throughout the race and attempt to discover the best possible strategy.

In order to do so, we will divide the race into  $N$  components by distance. For each of the components  $R_1 \dots R_N$  we allow the value for  $C_{eff}$  to vary between 0 and 1 – from no effort to maximum effort.

To determine the best effort during each component we will use Newton's method for systems. We will assume that there is a single “best” set  $\{C_{eff-1} \dots C_{eff-N}\}$  which yields the fastest time. To apply Newton's method for systems, our victory condition, our control function  $F$ , will be the partial derivative of final time with respect to each coefficient  $C_{eff-k}$ . When a small change in none of the values can improve the final time, we will have found the best values.

The results will be reported in the final draft.

# Bibliography

- Abbiss, Chris R., and Paul B. Laursen. "Models to Explain Fatigue During Prolonged Endurance Cycling." *Sports Med* 35 (2005).
- Arsac, Laurent M., and Elio Locatelli. "Modeling the energetics of 100-m running using speed curves of world champions." *J Appl Physiol* 92 (2001): 1781-788.
- "Drag Coefficient." *Engineering ToolBox*. 03 Apr. 2009 <[http://www.engineeringtoolbox.com/drag-coefficient-d\\_627.html](http://www.engineeringtoolbox.com/drag-coefficient-d_627.html)>.
- Gibson, Alan St Clair, Michael I. Lambert, and Timothy D. Noakes. "Neural Control of Force Output During Maximal and Submaximal Exercise." *Sports Med* 31 (2001).
- "How to Determine Anaerobic Threshold." *Sports Fitness Advisor -- Sports Training Tips for Athletic Peak Performance*. 03 Apr. 2009 <<http://www.sport-fitness-advisor.com/anaerobicthreshold.html>>.
- Maronski, Ryszard. "Minimum-Time Running and Swimming: An Optimal Control Approach." *J. Biomechanics* 29 (1995): 245-49.
- Martin, David E., and Peter Coe. *Better training for distance runners*. 2nd ed. Champaign, IL: Human Kinetics, 1997.
- Peronnet, Francois, and Guy Thibault. "Mathematical analysis of running performance and world running records." *Modeling methodology forum*.
- Prampero, P.E. Di, C. Capelli, P. Pagliaro, G. Antonutto, M. Girardis, P. Zamparo, and R. G. Soule. "Energetics of best performances in middle distance running."
- Ward-Smith, A. J. "Aerobic and anaerobic energy conversion during high-intensity exercise." *Medicine and Science in Sports and Exercise* 31 (1999): 1855-860.

## Methods

The program for this investigation was written in C. Its source code is included after this page, in its entirety. The source code should compile and run without problems (though it will not produce the graphs without the allegro library); however, no guarantees – run at your own risk. The main risk of an infinite loop comes from setting the effort too high and having the runner run out of energy, collapsing from fatigue – velocity goes to zero, and the “race” is never completed.

```

/*
 * runner.c
 *
 * Created by Bo Waggoner on 4/1/09.
 *
 * A program to model the course of a distance race by numeric integration.
 *
 * Main loop calls run_prog(), which loops calling for_euler(), then calls plot().
 * Alternatively, main loop just calls for_euler() and plot().
 *
 */

//-----
// Options! Please read.

// include allegro game programming library? (used to make graphs)
// comment this out in order to simply output text / results
//#define USE_ALLEGRO

// print out splits during integration? If so, prints out a split for every
SPLIT_FREQ number of recordings of data (see MAX_ELEMENTS below)
// comment this line out to avoid printing splits while integrating
#define PRINT_SPLITS
#define SPLIT_FREQ 50

// print out all recorded steps after drawing plot? If so, prints out one recorded
value for every RECORD_FREQ (see MAX_ELEMENTS below) ... it takes a while for 10000
printfs
// will only print out splits for the last for_euler() run
// comment out this line to avoid printing splits after drawing plot
//#define PRINT_RECORDS
#define RECORD_FREQ 10

//-----
// Definitions etc.

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>

// include the allegro header file
#ifdef USE_ALLEGRO
#include <Allegro/allegro.h>
#endif

#define MILE 1609.344 // number of meters in a mile

```

```

// record this many different steps of the integration, equally spaced (for a
10000m race recording 10000 splits, record one per meter)
const int MAX_ELEMENTS = 10000;

const double DISTANCE = 10000;    // how many meters to run
const double mass = 55;           // mass of runner
const double STEP_SIZE = .001;    // in seconds, how far to time-step

const double Eff = .2;            // efficiency (how much power produced by the body
is actually transfered to forward motion)
const double CLT = .75;          // percent of effort at which the runner hits
lactate threshold
const double Tcp = 20;           // time in seconds during which creatine phosphate
system is primarily utilized
const double MAX_POW = 1000;     // maximum power output of the person, in watts
const double K_FAT = 780;       // constant related to how quickly we fatigue
(exponential part)
const double SPEED_MULT = 1;     // scale the power output by this factor

#define Pcp MAX_POW              // maximum power output from each energy system
#define Paer (MAX_POW*CLT)
#define Panaer (MAX_POW*(1-CLT))

#define NUM_E_VALS 1             // number of different stages to break
the race into
#define E_STEP_SIZE (DISTANCE/NUM_E_VALS); // size of each stage, in meters

// the low and high effort values -- bounding the optimal value (hopefully), at
each stage
const double E_VALS[2][NUM_E_VALS] =
{
    {.20},

    {.95}
};

//-----
// Global variables and function prototypes

double *Vel;    // arrays of steps recorded during integration for velocity,
position, time, and effort
double *Pos;
double *Time;
double *Effort;

inline double effort(double x, double t, double *e_vals);    //

```

```

return current effort
inline double airRes(double v); //
return air resistance
inline double propForce(double x,double v,double t,double Ceff,double Fat); //
return propulsive force per unit mass
inline double manPower(double t,double x,double v,double Ceff,double Fat); //
return total power generated by runner
inline double delta_fatigue(double h, double Ceff); //
return how much to change fatigue
double for_euler(double h, double *e_vals); //
forward euler method, passing e_vals to effort() in order to determining Ceff
void plot(void); // plot
Vel, Time, and Pos vs. effort. If not using allegro, does nothing
int main(int argc, const char *argv[]); // main
method
void run_prog(void); //
currently, uses bisection to determine optimal constant value for effort

//-----
// Forward Euler and helper functions

// effort
// given position x, time t, and array of e_vals, output appropriate e_val (effort
// value) ... or make up a function for effort
inline double effort(double x, double t, double *e_vals)
{
    /*double prop = x/DISTANCE;
    int step = prop / E_STEP_SIZE;
    if(x/DISTANCE < .95) return e_vals[step];
    return 1;*/
    double temp = x/DISTANCE;
    return temp * temp * temp / 2 + .5; // vary effort based on percent
finished with race
}

// force per unit mass due to air resistance, given velocity v
// return force per mass in m /s^2
inline double airRes(double v)
{
    return .01 * v * v;
}

// propulsive force
// given distance x, velocity v, time t, effort Ceff, and fatigue Fat
// return the propulsive force per unit mass (in m / s^2)
inline double propForce(double x,double v,double t,double Ceff,double Fat)
{
    return (manPower(t,x,v,Ceff,Fat) * Eff / (mass * v));
}

```

```

}

// power generated by the body
// given time t, distance x, velocity v, effort Ceff, and fatigue Fat
// return power in Watts
inline double manPower(double t, double x, double v, double Ceff, double Fat)
{
    // at less than Tcp, all energy comes from creatine phosphate system and stored
    ATP
    double answer;
    if(t < Tcp) {
        answer = Ceff * Pcp;
    }

    else {

        // at less than CLT effort, all energy comes from aerobic metabolism
        double c_diff = Ceff - CLT;
        if(c_diff < 0) answer = (Ceff * Paer) / CLT;

        // at greater than that effort, energy comes from both aerobic and
        anaerobic metabolism
        else {
            answer = Paer + (c_diff * Panaer)/(1 - CLT);
        }
    }

    // change answer to deal with fatigue
    answer *= SPEED_MULT * exp(-Fat/K_FAT);
    return answer;
}

// update fatigue
// given step size h and current effort Ceff
// return the change in Fat, fatigue level
inline double delta_fatigue(double h, double Ceff)
{
    // if only using aerobic power
    double c_diff = Ceff - CLT;
    if(c_diff < 0) return Ceff * Ceff * h;

    // if using both aerobic and anaerobic power
    else {
        double ouch = Ceff + 10*c_diff*c_diff;
        return ouch * ouch * h;
    }
}

// forward euler
// h is the step size, e_vals[] is a list of effort variables

```

```

// print splits (if PRINT_SPLITS is uncommented) and final result
// return time in seconds
double for_euler(double h, double *e_vals)
{
    double t = 0;
    double x = 0;
    double v = .1;

    double old_x = 0;

    double Fat = 0;    // fatigue

    // record MAX_ELEMENTS steps: each time our x-value passes next_record,
    // we record the current values and move next_record forward by record_step
    double record_step = DISTANCE / MAX_ELEMENTS;
    double next_record = record_step;
    int count = 1;
    Vel[0] = 0;
    Pos[0] = 0;
    Time[0] = 0;

    // main loop of numeric integration
    while(x < DISTANCE)
    {
        // our dv/dx = f(v,x,t,Ceff,Faer,Fanaer) is the change in velocity, v is
what we ae integrating
        double Ceff = effort(x,t,e_vals);
        // calculate current propulsive and resistive net forces, use it to get
change in velocity
        double pout = propForce(x, v, t, Ceff, Fat);
        double resist = airRes(v);
        double dVdt = pout - resist;

        // advance the variables
        old_x = x;
        v += dVdt * h;
        x += v*h;
        t += h;
        Fat += delta_fatigue(h,Ceff);

        // record data
        if(x > next_record && count < MAX_ELEMENTS) {
            next_record += record_step;
// if we want to print out splits along the way
#ifdef PRINT_SPLITS
                if(count % SPLIT_FREQ == 0) printf("v,t,x,eff:  %f, %f, %f,  %f\n",
v,t,x,Ceff);
#endif
                Vel[count] = v;
                Pos[count] = x;

```

```

        Time[count] = t;
        Effort[count] = Ceff;
        count++;
    }
}

printf("Final:    v,t,x: %f, %f, %f\n",v,t,x);
int min = t/60;
int sec = t - (60*min);
double hund = t - 60*min - sec;
printf("Average velocity: %f\n",x/t);
printf("Time: %d:%d:%.0f\n\n",min,sec,hund*100);
return t;
}

//-----
//-----
// main function

int main(int argc, const char *argv[])
{
// install/initialize allegro
#ifdef USE_ALLEGRO
    if(allegro_init() != 0) return -1;
    if(install_keyboard() != 0) return -2;
    if(set_gfx_mode(GFX_AUTODETECT, 640, 400, 0, 0)) return -3;
#endif

    Vel = malloc(sizeof(double) * MAX_ELEMENTS);
    Pos = malloc(sizeof(double) * MAX_ELEMENTS);
    Time = malloc(sizeof(double) * MAX_ELEMENTS);
    Effort = malloc(sizeof(double) * MAX_ELEMENTS);

    int i;
    for(i=0; i<MAX_ELEMENTS; i++) {
        Vel[i] = 0;
        Pos[i] = 0;
        Time[i] = 0;
        Effort[i] = 0;
    }

    // rather than running the bisection method run_prog, just do forward euler for
one effort value
// right now, will not matter what ev is! change effort() function instead
double *ev = malloc(sizeof(double));
ev[0] = .7;
for_euler(STEP_SIZE, ev);
plot();
free(ev);

```

```

//run_prog();

free(Vel);
free(Pos);
free(Time);
free(Effort);

return 0;
}
#ifdef USE_ALLEGRO
END_OF_MAIN(); // macro required by allegro
#endif

//-----
// Other functions

// currently uses bisection method to find the best Ceff that is constant, that
yields the fastest time
void run_prog(void)
{

// our current experimental values will be stored here
double *e_vals = malloc(sizeof(double) * NUM_E_VALS);

// find a (low bound)
memcpy(e_vals,E_VALS[0],sizeof(double) * NUM_E_VALS);
double a = e_vals[0];
double fa = for_euler(STEP_SIZE,e_vals);

// find b (upper bound)
memcpy(e_vals,E_VALS[1],sizeof(double) * NUM_E_VALS);
double b = e_vals[0];
double fb = for_euler(STEP_SIZE,e_vals);

// use bisection to find smallest time
while(abs(100*a - 100*b) != 0) {
printf("a,b: %f, %f\n",a,b);
double c1 = (a + b)/2;
double c2 = c1 + .001; // try two slightly different initial
conditions to determine which way the slope goes
e_vals[0] = c1;
double fc1 = for_euler(STEP_SIZE,e_vals);
e_vals[0] = c2;
double fc2 = for_euler(STEP_SIZE,e_vals);
if(fc2 > fc1) {
b = c2;
fb = fc2;
}
}
}

```

```

        else {
            a = c1;
            fa = fc1;
        }

// if using allegro, allow user to break while loop
#ifdef USE_ALLEGRO
    if(key[KEY_ESC]) break;
#endif

    }

    free(e_vals);

    plot();
}

// plot the results and wait for a keypress
void plot(void)
{
#ifdef USE_ALLEGRO
    int tCol = makecol(240,80,30);
    int vCol = makecol(90,240,140);
    int eCol = makecol(60,100,240);
    double TMAX = Time[MAX_ELEMENTS-1] + 10;
    double VMAX = 10;

    clear_to_color(screen, makecol(255, 255, 255));

    int i;
    // plot each element in its own color
    for(i=0; i<MAX_ELEMENTS; i++) {
        double x = Pos[i] * (double)SCREEN_W / (double)DISTANCE;
        double y = SCREEN_H - (Time[i] * ((double)SCREEN_H) / ((double)TMAX)) - 1;
        int yt;
        for(yt = y-1; yt<=y+1; yt++) {
            if(yt >= 0 && yt < SCREEN_H) _putpixel(screen,x,yt,tCol);
        }
        y = SCREEN_H - Vel[i] * (double)SCREEN_H / (double)VMAX - 1;
        for(yt = y-1; yt<=y+1; yt++) {
            if(yt >= 0 && yt < SCREEN_H) _putpixel(screen,x,yt,vCol);
        }
        y = SCREEN_H - (Effort[i] * (double)SCREEN_H);
        for(yt = y-1; yt<=y+1; yt++) {
            if(yt >= 0 && yt < SCREEN_H) _putpixel(screen,x,yt,eCol);
        }
    }

    // wait for a keypress
    readkey();

```

```
#endif

#ifdef PRINT_RECORDS
    printf("Recorded data:\nnum    time, distance, velocity, effort\n");
    for(i=0; i<MAX_ELEMENTS; i++) {
        if(i%RECORD_FREQ == 0) printf("%d:    %f, %f, %f,
%f\n", i, Time[i], Pos[i], Vel[i], Effort[i]);
    }
#endif
}
```