A Novel Algorithm for Determining Route Difficulty Rating

Erwin Bofetiado, Luke Overman, Tom Overman

November 2019

1 Introduction

The purpose of this paper is to provide further investigation and expansion of the original work of Paul Petzoldt in his guide book, *Teton Trails: A hiking guide to the Teton Range with stories, history, and personal experiences.* In 1976, Petzoldt proposed a unit of measurement useful for accurately gauging the difficulty of any one particular mountain trail, thus allowing hikers to plan effectively and make the appropriate preparations. An "energy mile", as he defined it, is the amount of energy required to traverse one mile, with an additional two energy miles for every 1,000 ft of elevation gain i.e an individual hiking one mile with the aforementioned elevation gain will have used the equivalent of three energy miles. His work led to a study conducted in 2010 by students at Western Carolina University Department of Exercise Physiology under the supervision and direction of faculty members Dr.Maridy Troy and Maurice Phipps, Ph.D. Their efforts confirmed Petzoldt energy mile theory, albeit with a value of 1.6 energy miles for every 1,000 ft of elevation. However upon further analysis we discovered that this system is likely idealized and may not necessarily translate to trails found in nature. Thus, we believe this system must incorporate the following variables; (1) variations in elevation of the projected hiking distances, (2) consideration for negative (downwards) slopes, and (3) the various terrains one might encounter e.g dirt, snow, sleet, etc.

2 Statement of Problem

Devise an algorithm that calculates the energy miles of a given route based on the elevation profile of the route. The algorithm will incorporate the use of interpolation, differentiation, and numerical integration. The system will be tested on various routes of well-established qualitative difficulty to verify accuracy.

3 Description of The Mathematics

For the old system we make our energy effort of section i of the route to be: $E_i = \Delta l_i + 10.56\Delta y_i$ where Δl_i is the overall length of the section and Δy_i is the change of elevation within the section.

We then introduce the difficulty coefficient D_i which quantifies the difficulty of traversing the section i of the route. Using this difficulty coefficient, we then adapt our energy effort equation to $\Delta E_i = D_i \Delta l_i$.

We are now tasked with finding an appropriate formulation of the difficulty coefficient as a function of the slope of the section. It is clear that the difficulty coefficient should have the following properties:

- $D_i = 1$ for $m_i = 0$
- $D_i > 1$ for $m_i > 0$
- $D_i < 1$ for $m_i < 0$

Based off of existing literature in exercise science, we can use an exponential function to model the difficulty. We introduce $D_i = e^{\alpha m_i}$, where we choose $\alpha = 4.2351$ based off of data collected from a study by Balducci et al. published in the Journal of Sports Science and Medicine on the energy costs of well-trained mountain runners on different incline grades. In this study, a chart is given comparing the energy cost of running to the percent grade of the treadmill that the runners were running on. We normalize this data so that at a zero percent grade (i.e. flat terrain) the energy cost is unity. Thus, the normalized data is readily understood as a difficulty coefficient that is a function of the inclination of the terrain. Since our proposal is to have an exponential function represent the difficulty coefficient, we are able to fit an exponential function to this normalized data with $R^2 = 0.9856$, which yields a value for α of 4.2351. It is important to note that α is independent of the route, but not necessarily independent of the physical condition of an athlete or adventurer, as the data in the study was taken from elite mountain runners in excellent physical condition. Thus, more average individuals may have higher values of α, which would, of course, yield higher difficulty coefficients.

Now in the continuum limit, that is, $\Delta l_i \to 0$, we have the differential $dE = D(l)dl = e^{\alpha p_n'(l)}dl$, where $p_n'(l)$ is the derivative of the interpolation polynomial of the elevation profile GPS data that is readily acquired with online software. Therefore, if the length of the entire route is L,

$$E = \int_0^L e^{\alpha p_n'(l)} dl.$$

For the special case of an"Out and Back" Route, the way back is exactly the opposite as the way out, (i.e. the way back has an interpolant of $-p_n'(l)$), therefore:

$$E = \int_0^{L/2} e^{\alpha p_n'(l)} dl + \int_0^{L/2} e^{-\alpha p_n'(l)} dl,$$

Thus,

$$E = 2\int_0^{L/2} cosh(\alpha p_n'(l))dl.$$

4 Description of the Algorithm

This section will describe the numerical methods used to approximately solve the problem formalized in the previous section. The two main numerical techniques required are interpolation and numerical integration.

4.1 Interpolation

The elevation profile data we receive from various online tools consists of ordered pairs of the form (total distance, altitude). We have a finite number of points but our algorithm requires information at points between these given points. Therefore we need an effective interpolation technique.

The first interpolation technique we will explore is polynomial interpolation. This is simply fitting an n degree polynomial to $n+1$ data points. There are various basis functions we can use to achieve this, but we will use the Lagrange form because we can analytically find an expression of its derivative without much trouble. The polynomial is given by:

$$p_n(l) = \sum_{i=0}^{n} \ell_i(l) f(l_i)$$

The derivative of this polynomial is given by:

$$p_n'(l) = \sum_{i=0}^{n} \ell_i'(l) f(l_i)$$

where

$$\ell_i'(l) = \sum_{j=0, j\neq i}^{n} \left[\frac{1}{l_i - lj} \prod_{m=0, m\neq(i,j)}^{n} \frac{l - l_m}{l_i - l_m} \right]$$

However, since the data we will be using consists of a very large number of points, the polynomial interpolant will be of very high degree. High degree polynomials tend to be very oscillatory and thus will not accurately predict the points between the nodes very well.

The deficiencies of high degree polynomial interpolation motivates the need for a better interpolation technique. We used a natural cubic spline interpolation to correct these wild oscillations while still maintaining differentiability and the correct nature of the route. The cubic spline equation for section i is given

below. The values of z_i are found once at the beginning of the program then used when needed.

$$C_i(x) = \frac{z_{i+1}}{6h_i}(x-t_i)^3 + \frac{z_i}{6h_i}(t_{i+1}-x)^3 + (\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1})(x-t_i) + (\frac{y_i}{h_i} - \frac{h_i}{6}z_i)(t_{i+1}-x)$$

The first derivative of this interpolant can be found analytically. This expression is given below and is easy to compute.

$$C_i'(x) = \frac{z_{i+1}}{2h_i}(x-t_i)^2 - \frac{z_i}{2h_i}(t_{i+1}-x)^2 + (\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}) - (\frac{y_i}{h_i} - \frac{h_i}{6}z_i)$$

4.2 Numerical Integration

Now that a valid interpolation technique has been found, we can approximate the integral describing the difficulty rating of the route given in the previous section. We will approach this approximation using two different techniques: Composite Trapezoid Rule and Composite Simpson's Rule.

The Composite Trapezoid rule for an "out-and-back" route is given below:

$$E = 2\int_0^{L/2} \cosh(\alpha C'(l))dl \approx 2\sum_{i=0}^{n-1}(\frac{x_{i+1}-x_i}{2})(\cosh(\alpha C'(x_i))+\cosh(\alpha C'(x_{i+1})))$$

The Composite Simpson's rule for an "out-and-back" route is given below:

$$E = 2\int_0^{L/2} f(l)dl \approx 2\sum_{i=1}^{n/2}\frac{h}{3}(f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i}))$$

where $f(x) = \cosh(\alpha C'(x))$

For small spacing, h, we expect the Composite Simpson's Rule to be a more accurate approximation than the Composite Trapezoid Rule.

5 Results

We tested this algorithm first on a steep hiking route to the highest peak in Colorado. This helped reveal the best numerical methods to use for other routes. We then demonstrated the utility of the algorithm for comparing two different routes by examining two different marathon routes.

5.1 Mt. Elbert East Ridge Trail Study

The first step was to develop a proper interpolation procedure for the elevation profile data. Using polynomial interpolation with 250 points yielded a 249 degree polynomial interpolant that had massive fluctuations near the endpoints. This is a common problem with high-degree polynomials called Runge's Phenomenon. We reduced the system to a 4 degree polynomial which alleviated

Runge's phenomenon but smoothed the profile so much that it no longer had the defining characteristics of the route.

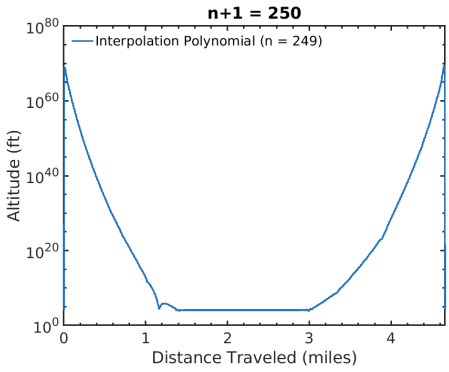

Figure 1: 249 Degree Polynomial Interpolant

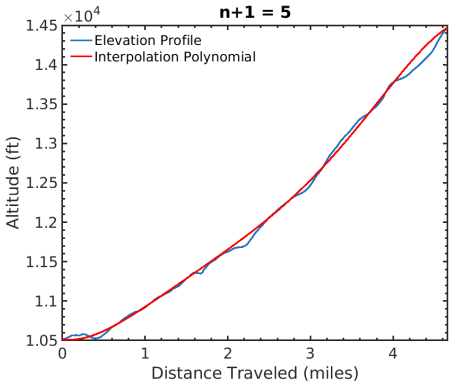

Figure 2: 4 Degree Polynomial Interpolant

Clearly, a different interpolation method had to be used. In order to maintain the complete differentiability at all points on the elevation profile while maintaining the right flavor of the route, we used cubic spline interpolation. The natural cubic spline interpolation maintained the characteristics of the route without any abnormal fluctuations. Clearly, cubic spline interpolation would be the method we would use. Finding the first derivative of the interpolant also proved to be simple.

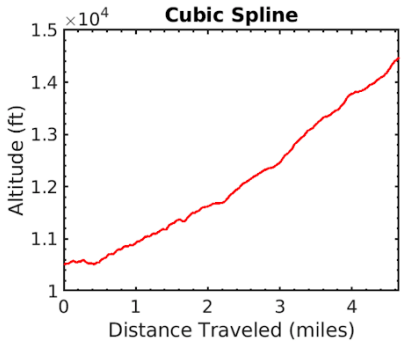

Figure 3: Natural Cubic Spline Interpolant

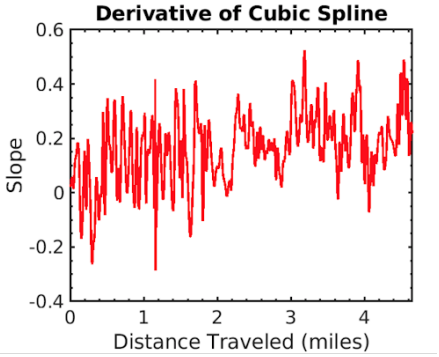

Figure 4: First Derivative of Cubic Spline Interpolant

The next step was to approximate the integral mentioned previously. We

approached this with two different techniques: Composite Trapezoid Rule and Composite Simpson's Rule. Our spacing between points was so small that we expect Composite Simpson's Rule to have less error because it is fourth order accurate. For this simulation, both methods gave similar results because of the very small step size of $h = 0.0047$.

The Composite Trapezoid Rule gave a rating of 13.4398 energy miles while the Composite Simpson's Rule gave a rating of 13.4199 energy miles. The actual length of the trail is 9.318 miles. The additional 4.1218 miles are due to the extremely steep portions of the route that our algorithm incorporates into the rating because of the difficulty of traversing these steep regions.

5.2 Berlin vs. Rim Rock Marathon Comparative Study

We compared the Berlin and Rim Rock marathons to demonstrate the comparative utility of the algorithm. The Berlin marathon is a very flat and "easy" course while the Rim Rock marathon is a very steep and difficult marathon through the Rocky Mountains of Colorado. Clearly, the Rim Rock marathon should give a much larger energy mile rating.

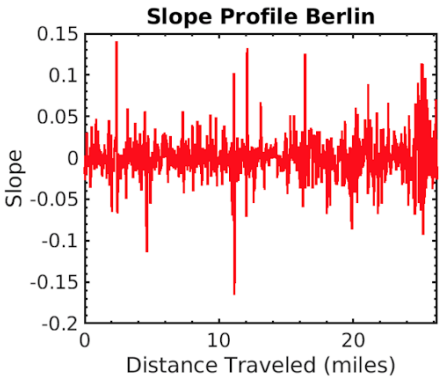

Figure 5: Slope of Berlin Marathon

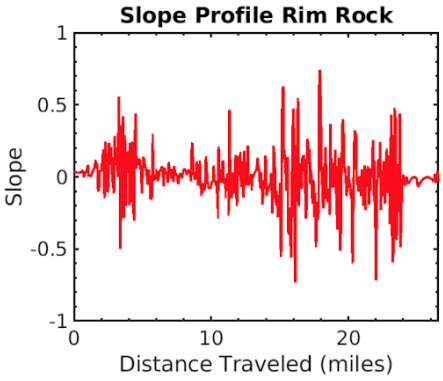

Figure 6: Slope of Rim Rock Marathon

We used cubic spline interpolation with the composite Simpson's Rule to approximate the difficult of the routes. The Berlin Marathon was found to have a difficulty rating of 26.4132 energy miles while the Rim Rock marathon had a difficulty rating of 35.4437 energy miles. Clearly, the Rim Rock marathon is the more challenging course and would be about equivalent to running 35.4437 miles on a perfectly flat course.

If an athlete wanted to qualify for a prestigious race they would want to achieve their fastest time. Using this system, they could quantitatively determine the easiest course that would give them the best chance at qualifying.

6 Conclusions

The novel technique for quantitatively determining route difficulty that was developed in this report is an improvement on the traditional method described in the introduction in that it can be applied to any route given GPS data of the route and that it accounts for intermediate points along the route in between the endpoints of the hike. However, the algorithm still preserves the "energy mile" developed by Petzoldt that many hikers are familiar with. Thus, difficulties in units of energy miles can be assigned to any route for backpackers to accurately assess their energy needs, for runners to gauge the difficulty of a run and train accordingly, or for casual hikers to have some idea of what they are to embark on on their next day hike.

However, there are some limitations in using this algorithm to quantify the difficulty of a route. In determining the constant in the exponent of the difficulty coefficient, α, a study was used that tested elite mountain runners and determined their energy costs depending on the incline of the treadmill they ran on. It is obvious that elite mountain runners have far-above-average physical conditioning, thus the values obtained in this report may not be completely representative for less-fit individuals. Additionally, the study only provided three points to fit the exponential curve to, which decreases confidence in the value for α that was obtained. Also, the study did not include any information on negative slopes, so the claim that $D < 1$ for $m < 0$ was made purely off of the personal experience of the authors, who deem it reasonable for the moderately negative slopes encountered in most routes. Finally, the algorithm was not tested for slopes outside of the interval [-1,1] where the exponential difficulty coefficient may not accurately reflect the true difficulty. Without data that investigates the energy cost at such extreme inclines and declines in the exercise science literature, we cannot conclude this algorithm's validity outside of this interval of slopes. However, most hikes, and certainly almost all runs, do not encounter such extreme incline grades, so this is not such a pressing issue.

There is still ample room to further refine this algorithm we developed. As aforementioned, the value for α obtained in this report may not be representative for more average-conditioned individuals. If a new study were to emerge in exercise science literature that used average-conditioned subjects, a new value for α could be found to further increase the validity of this tool for all people. However, even as is, this algorithm still can serve as a powerful comparative tool for more average-conditioned users to compare the difficulty of a route to the difficulty of one they have already completed. Additionally, it is reasonable to retain from Petzoldt's original theorem that when $E < 5$, the route is easy, when $5 \leq E < 10$, the route is intermediate, and when $E \geq 10$, the route is strenuous for most people. Furthermore, there are several other physical parameters such as altitude, climate, and ground conditions (snow, dirt, etc.) that could be implemented with the support of studies in exercise science to more accurately rate the difficulty of a course. Of these physical parameters, altitude is certainly the most influential on route difficulty, as the concentration of oxygen in the air decreases significantly at higher altitudes. Of course, acclimatization to

higher altitudes would minimize this effect on the route difficulty. Finally, with the hopes of increasing the accessibility of this tool, we plan on making the codebase open-source on Github to help others more accurately plan their next outdoor adventure.

With this novel technique for rating route difficulties, we hope to improve the experience of any athlete, adventurer, or casual hiker on their next endeavor by providing a reliable estimation of what they should expect on their route.

7 Program Listing

```
%Evaluates the cubic spline interpolant at given point
function val =  eval_spline(n,t,y,z,x)
    for i=n-1:-1:1
         if x-t(i)>=0
             break;
         end
    end
    h=t(i+1) -t(i);
    tmp = (z(i)/2) + (x-t(i))*(z(i+1)-z(i))/(6*h);
    tmp = -(h/6)*(z(i+1)+2*z(i))+(y(i+1)-y(i))/h + (x-t(i)
        )*tmp;
    val = y(i) + (x-t(i))*tmp;
end
```

```
%Evaluates the derivative of cubic spline interpolant at
    given point
function val =  eval_spline_deriv(n,t,y,z,x)
    for i=n-1:-1:1
         if x-t(i)>=0
             break;
         end
    end
    h=t(i+1) -t(i);
    tmp = (z(i+1)/(2*h)) * (x-t(i))^2 - (z(i)/(2*h)) * (t(
        i+1)-x)^2 + (y(i+1)/h - h*z(i+1)/6) - (y(i)/h-h*z(i
        )/6);
    val = tmp;
end
```

```
close all; clear all;
%Read in data values
thing = readmatrix("mt_elbert_east_ridge.csv");
t = thing(:,2);
y = thing(:,1);

```

```
%find cubic spline z values
[n,r]=size(t)
for i=1:n-1
    h(i) = t(i+1)-t(i);
    b(i) = (y(i+1)-y(i))/h(i);
end
u(1) = 2*(h(1)+h(2));
v(1) = 6*(b(2)-b(1));
for i=2:n-2
    u(i) = 2*(h(i+1) + h(i)) - h(i)^2/u(i-1);
    v(i) = 6*(b(i+1)-b(i)) - h(i)*v(i-1)/u(i-1);
end
z(n)=0;
for i=n-1:-1:2
    z(i) = (v(i-1)-h(i)*z(i+1))/u(i-1);
end
z(1)=0

%find and plot elevation profile interpolant
dist=linspace(0,4.659,1000);
for x=1:1000
    sol(x) = eval_spline(n,t,y,z,dist(x));
end
figure('Position',[10,10,900,700])
hold on
box on
set(gca,'Linewidth',2.0,'FontSize',20)
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
set(gca,'XLim',[0,max(t)])
title("Cubic Spline")
xlabel('Distance Travelled (mi)')
ylabel('Elevation (ft above sea level)')
plot(dist, sol, 'r','linewidth',2.5,'markersize',8);
ylabel("Altitude (ft)");
xlabel("Distance Traveled (miles)")

%find and plot derivative of elevation profile
    interpolant
dist=linspace(0,4.659,1000);
for x=1:1000
    sol(x) = eval_spline_deriv(n,t,y,z,dist(x));
end
sol = sol./5280
figure('Position',[10,10,900,700])
hold on
```

```
box on
set(gca,'Linewidth',2.0,'FontSize',20)
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
set(gca,'XLim',[0,max(t)])
title("Derivative of Cubic Spline")
xlabel('Distance Travelled (mi)')
ylabel('Elevation (ft above sea level)')
plot(dist, sol, 'r','linewidth',2.5,'markersize',8);
ylabel("Slope");
xlabel("Distance Traveled (miles)")

%composite trapezoid rule
x=linspace(0,t(250),1000);
app = 0;
a = 4.2351;
for i=1:999
    app = app + ((x(i+1) - x(i))/2) *(cosh(a*
        eval_spline_deriv(n,t,y,z,x(i+1))/5280) + cosh(a*
        eval_spline_deriv(n,t,y,z,x(i))/5280));
end
app=app*2

%composite simpson's rule
x=linspace(0,t(250),1000);
app = 0;
a = 4.2351;
h=(x(1000)-x(1))/1000;
for i=2:500
    app = app + (h/3)*(cosh(a*eval_spline_deriv(n,t,y,z,x
        (2*i-2))/5280) + 4*cosh(a*eval_spline_deriv(n,t,y,
        z,x(2*i-1))/5280) + cosh(a*eval_spline_deriv(n,t,y
        ,z,x(2*i))/5280));
end
app=app*2
```

```
close all; clear all;
thing = readmatrix("berlin_marathon.csv");
t = thing(:,2);
y = thing(:,1);
[n,r]=size(t)
for i=1:n-1
    h(i) = t(i+1)-t(i);
    b(i) = (y(i+1)-y(i))/h(i);
end
u(1) = 2*(h(1)+h(2));
```

```
v(1) = 6*(b(2)-b(1));
for i=2:n-2
    u(i) = 2*(h(i+1) + h(i)) - h(i)^2/u(i-1);
    v(i) = 6*(b(i+1)-b(i)) - h(i)*v(i-1)/u(i-1);
end
z(n)=0;
for i=n-1:-1:2
    z(i) = (v(i-1)-h(i)*z(i+1))/u(i-1);
end
z(1)=0

dist=linspace(0,max(t),1000);
for x=1:1000
    sol(x) = eval_spline_deriv(n,t,y,z,dist(x));
end
sol = sol./5280
figure('Position',[10,10,900,700])
hold on
box on
set(gca,'Linewidth',2.0,'FontSize',20)
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
set(gca,'XLim',[0,max(t)])
title("Slope Profile Berlin")
xlabel('Distance Travelled (mi)')
ylabel('Elevation (ft above sea level)')
plot(dist, sol, 'r','linewidth',2.5,'markersize',8);
ylabel("Slope");
xlabel("Distance Traveled (miles)")


x=linspace(0,t(2922),10000);
app = 0;
m=(y(2922)-y(1))/(5280*(t(2922)-t(1)));
a = 4.2351;
for i=1:9999
    app = app + ((x(i+1) - x(i))/2) *(exp(a*
        eval_spline_deriv(n,t,y,z,x(i+1))/5280) + exp(a*
        eval_spline_deriv(n,t,y,z,x(i))/5280));
end
app

x=linspace(0,t(2922),10000);
app = 0;
m=(y(2922)-y(1))/(5280*(t(2922)-t(1)));
a = 4.2351;
```

```
h=(x(10000)-x(1))/10000;
for i=2:5000
    app = app + (h/3)*(exp(a*eval_spline_deriv(n,t,y,z,x
        (2*i-2))/5280) + 4*exp(a*eval_spline_deriv(n,t,y,z
        ,x(2*i-1))/5280) + exp(a*eval_spline_deriv(n,t,y,z
        ,x(2*i))/5280));
end
app
```

```
close all; clear all;
thing = readmatrix("rim_rock.csv");
t = thing(:,1);
y = thing(:,2);
[n,r]=size(t)
for i=1:n-1
    h(i) = t(i+1)-t(i);
    b(i) = (y(i+1)-y(i))/h(i);
end
u(1) = 2*(h(1)+h(2));
v(1) = 6*(b(2)-b(1));
for i=2:n-2
    u(i) = 2*(h(i+1) + h(i)) - h(i)^2/u(i-1);
    v(i) = 6*(b(i+1)-b(i)) - h(i)*v(i-1)/u(i-1);
end
z(n)=0;
for i=n-1:-1:2
    z(i) = (v(i-1)-h(i)*z(i+1))/u(i-1);
end
z(1)=0

dist=linspace(0,max(t),1000);
for x=1:1000
    sol(x) = eval_spline_deriv(n,t,y,z,dist(x));
end
sol = sol./5280
figure('Position',[10,10,900,700])
hold on
box on
set(gca,'Linewidth',2.0,'FontSize',20)
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
set(gca,'XLim',[0,max(t)])
title("Slope Profile Rim Rock")
xlabel('Distance Travelled (mi)')
ylabel('Elevation (ft above sea level)')
plot(dist, sol, 'r','linewidth',2.5,'markersize',8);
```

```
ylabel("Slope");
xlabel("Distance Traveled (miles)")


x=linspace(0,t(648),2000);
app = 0;
a = 4.2351;
for i=1:1999
    app = app + ((x(i+1) - x(i))/2) *(exp(a*
        eval_spline_deriv(n,t,y,z,x(i+1))/5280) + exp(a*
        eval_spline_deriv(n,t,y,z,x(i))/5280));
end
app

x=linspace(0,t(648),2000);
app = 0;
a = 4.2351;
h=(x(2000)-x(1))/2000;
for i=2:1000
    app = app + (h/3)*(exp(a*eval_spline_deriv(n,t,y,z,x
        (2*i-2))/5280) + 4*exp(a*eval_spline_deriv(n,t,y,z
        ,x(2*i-1))/5280) + exp(a*eval_spline_deriv(n,t,y,z
        ,x(2*i))/5280));
end
app
```

8 References

Balducci, P., Clémençon, M., Morel, B., Quiniou, G., Saboul, D., & Hautier, C. A. (2016, May 23). Comparison of Level and Graded Treadmill Tests to Evaluate Endurance Mountain Runners. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4879436/.

Mount Elbert East Ridge Trail. (n.d.). Retrieved from https://www.alltrails.com/explore/trail/us/colorado/mount-elbert-east-ridge-trail?ref=sidebar-view-full-map.

Rocky Mountain National Park hiking trails. (n.d.). Retrieved from http://www.rockymountainhikingtrails.com/.

The Home of Colorado's Highest Peaks. (n.d.). Retrieved from https://www.14ers.com/.

Troy, M. M. N., & Phipps, M. L. (2010). The Validity of Petzoldt's Energy Mile Theory. Retrieved from https://digitalcommons.wku.edu/jorel/vol2/iss3/4/.