MAD4401 Project #2 Spring 2020

Blaise Bowman (3603-5146)

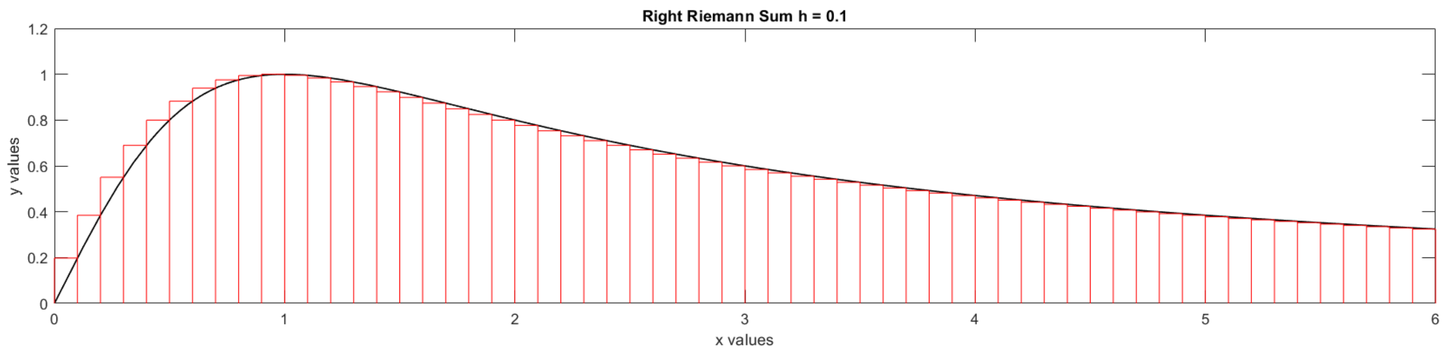
4/17/2020

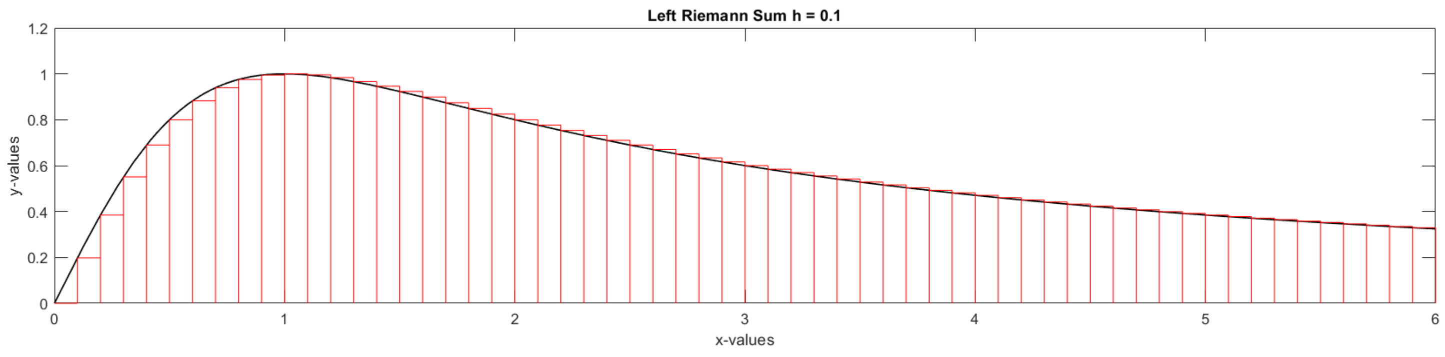
1. [Riemann Sums]

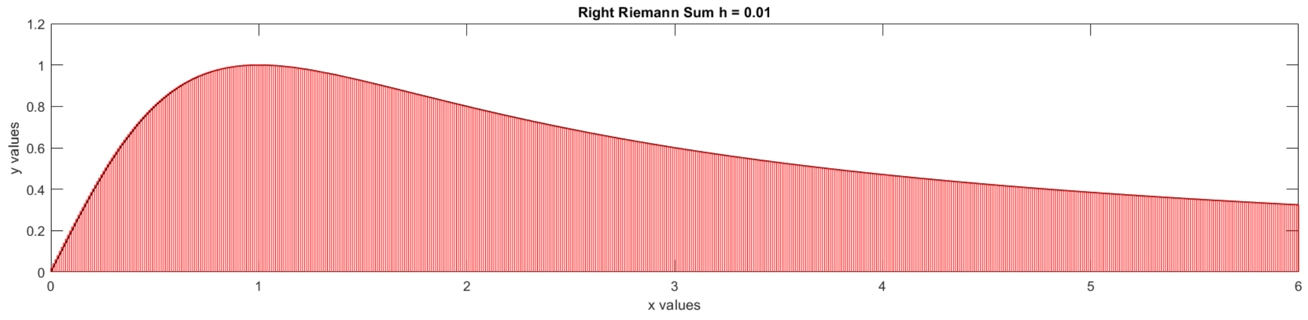
   Given the function $f(x) = \frac{2x}{1+x^2}$, we first must calculate: $\int_0^6 \frac{2x}{1+x^2} = \ln 37 =$ 3.61092. We will use this calculated value to determine how accurate the following integration techniques are.

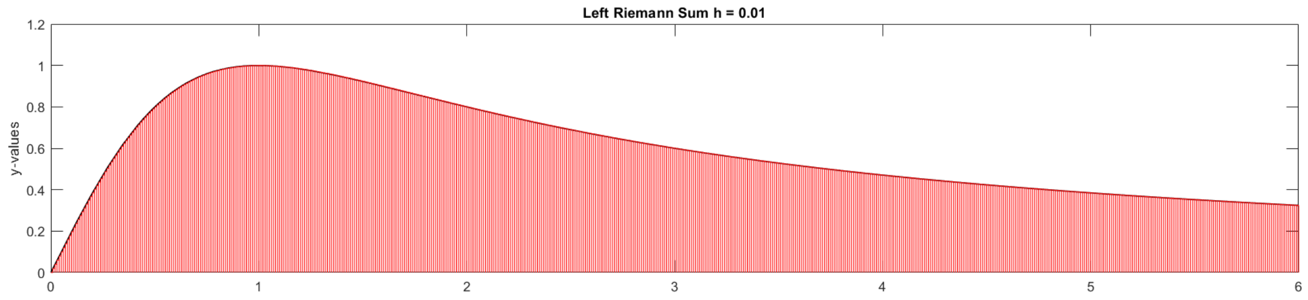
   For a Right Riemann sum, the approximated areas are: 3.62542 (for h = 0.1, 60 rectangles), 3.61252 (for h = 0.01 600 rectangles), and 3.6107555 (for h = 0.001 for 6000 rectangles).

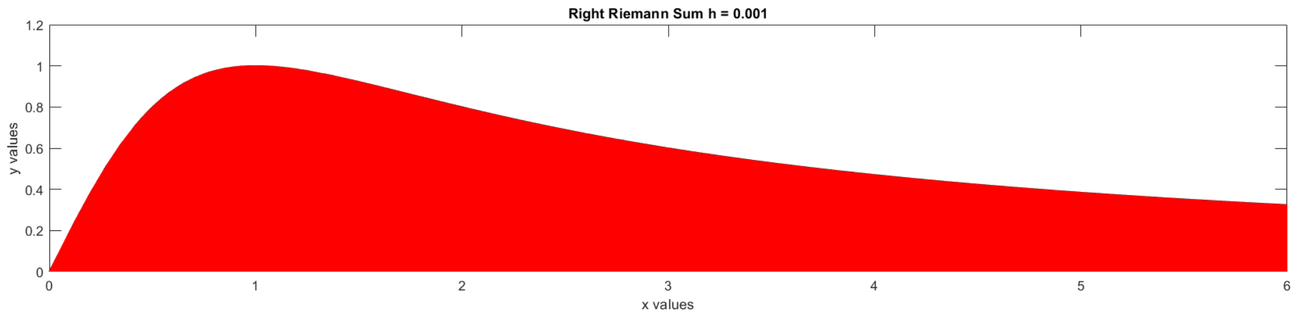
   For a Left Riemann sum, the approximated areas are: 3.59299 (for h = 0.1, 60 rectangles), 3.60927 (for h = 0.01 600 rectangles), and 3.61043 (for h = 0.001, 6000 rectangles).

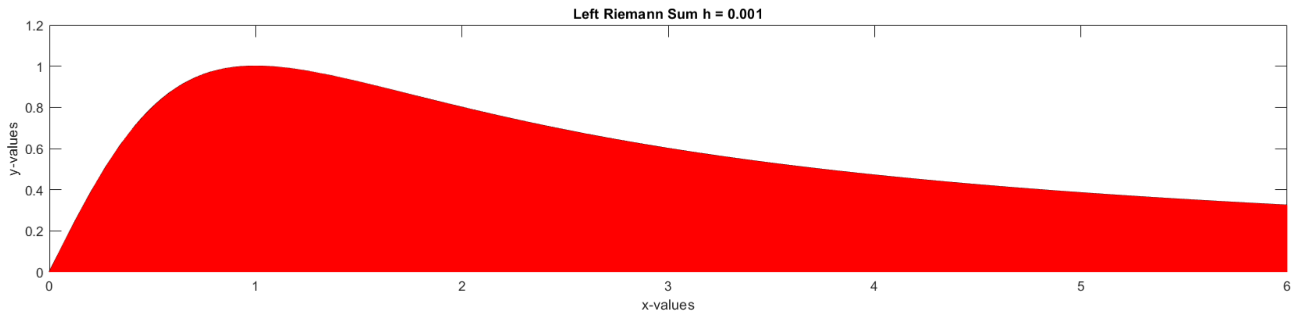










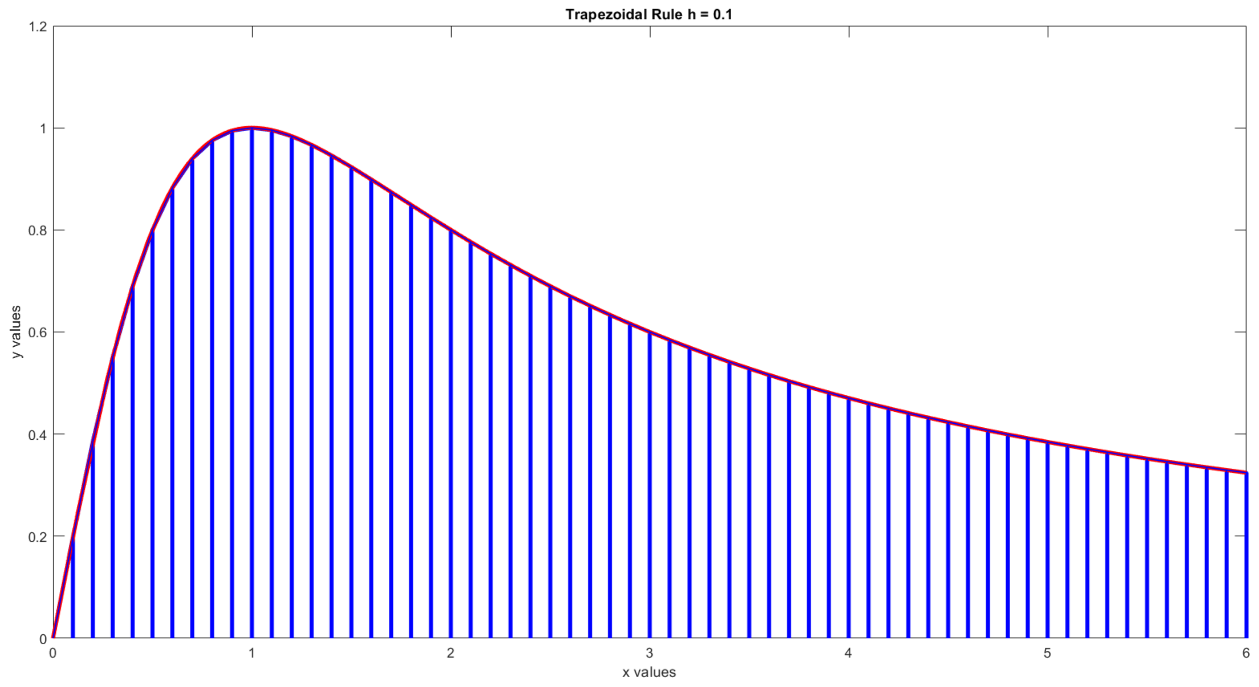


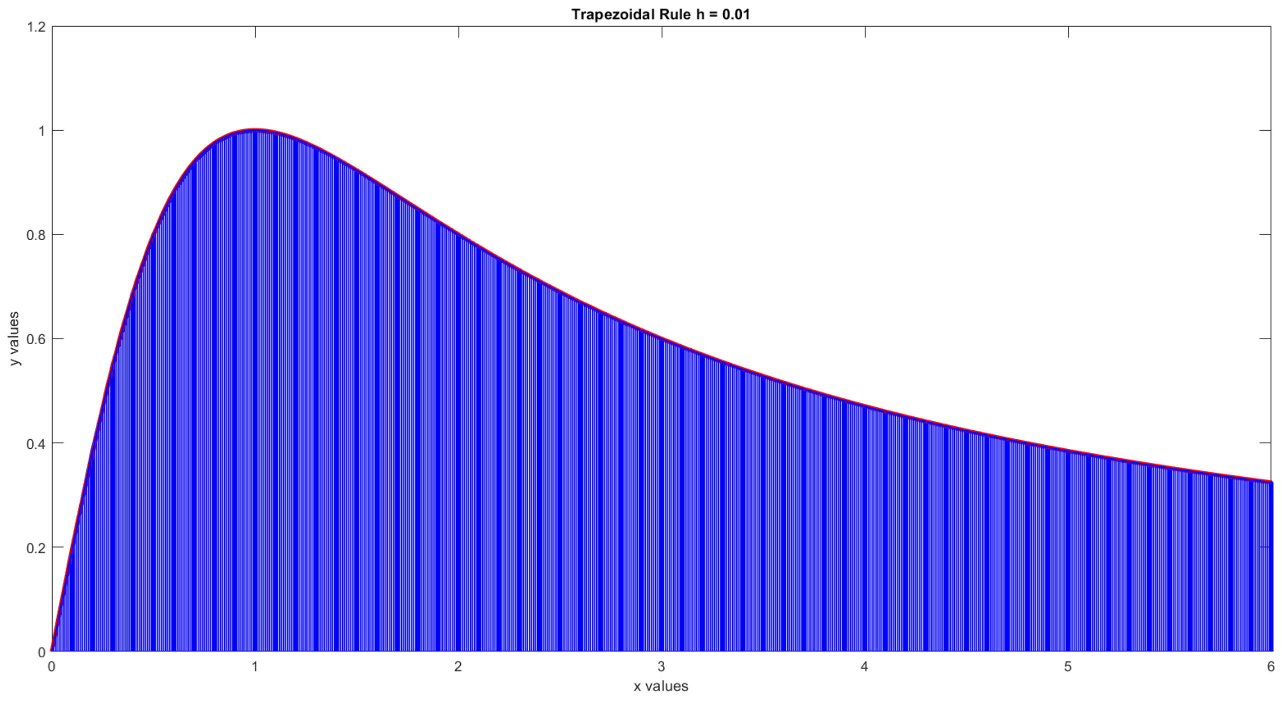
2. [Trapezoid Rule]

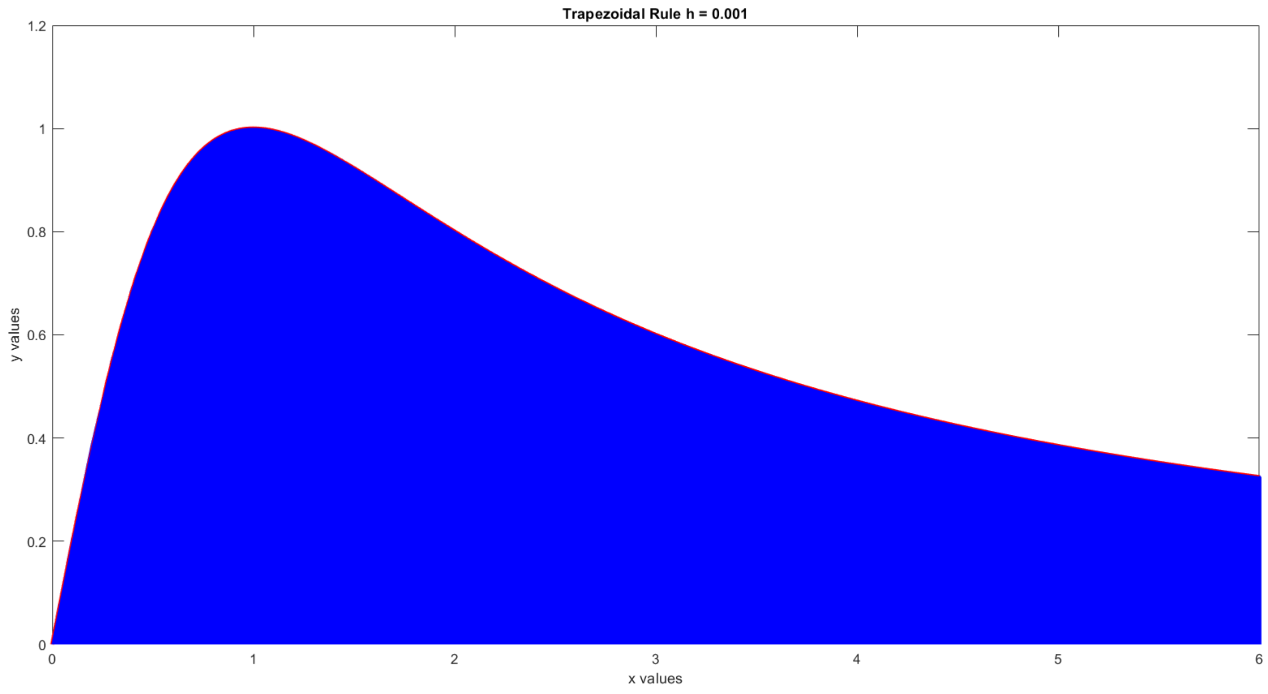
Given the function $f(x) = \frac{2x}{1+x^2}$, we first must calculate: $\int_0^6 \frac{2x}{1+x^2} = \ln 37 =$ 3.61092. We will use this calculated value to determine how accurate the following integration techniques are.

The trapezoid rule is defined as:

For the Trapezoid Rule, the approximated areas are: 3.60921 (for h = 0.1, 60 rectangles), 3.61090 (for h = 0.01 600 rectangles), and 3.61059 (for h = 0.001 for 6000 rectangles).







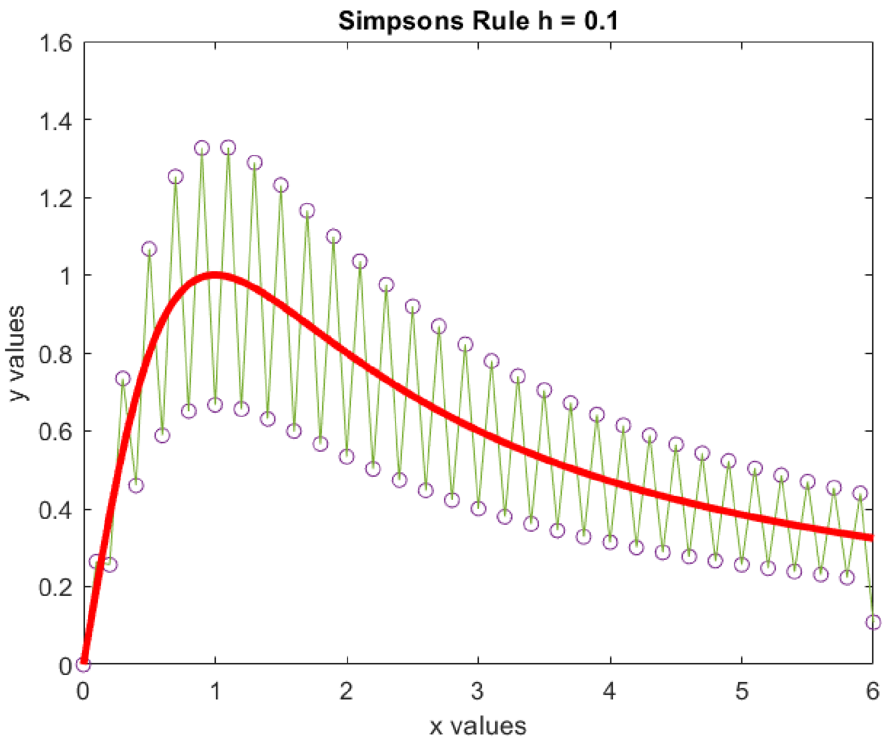
3. [Simpson’s Rule]

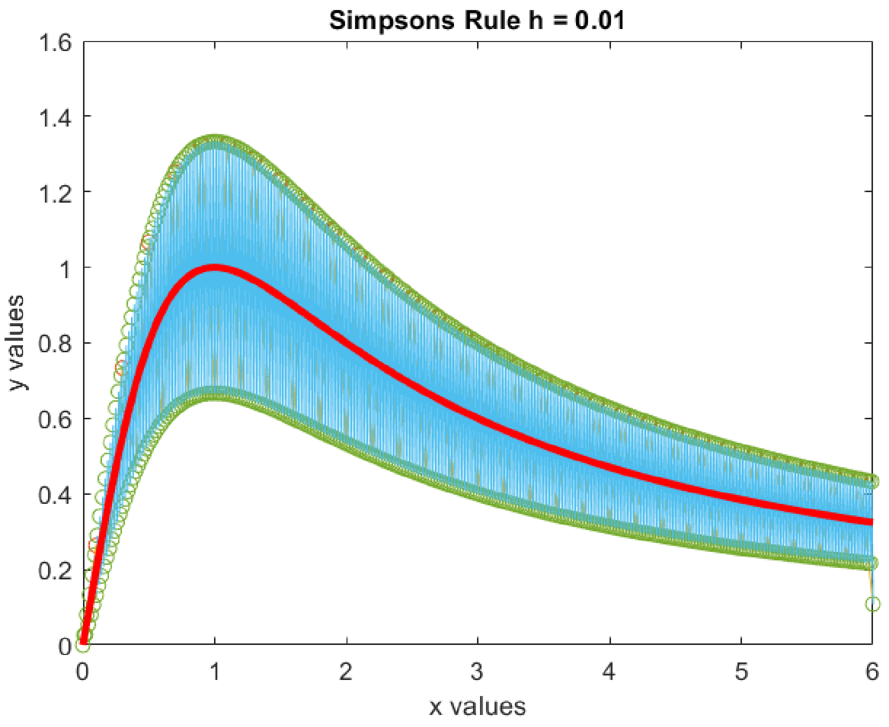
   Given the function $f(x) = \frac{2x}{1+x^2}$, we first must calculate: $\int_0^6 \frac{2x}{1+x^2} = \ln 37 =$ 3.61092.

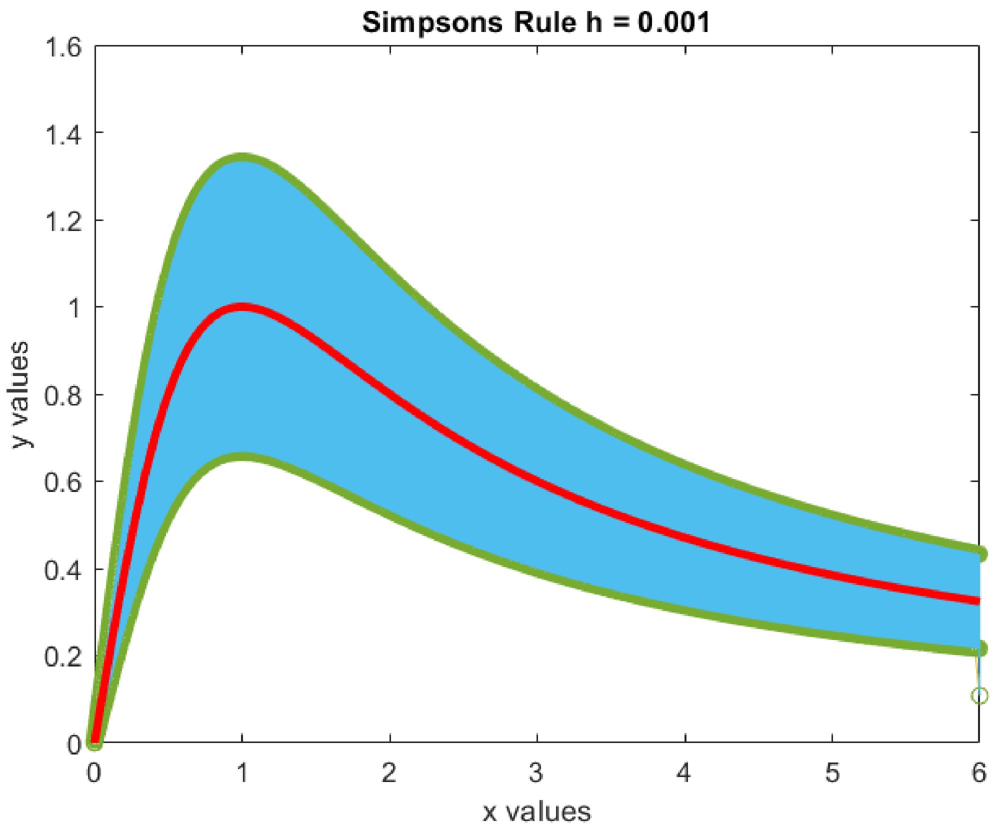
   Simpson’s Rule is a numerical integration method that utilizes a quadratic polynomial to help approximate a definite integral.

   Simpson’s Rule is defined by:

   For Simpson’s Rule, the approximated areas are: 3.610924 (for h = 0.1, 60 rectangles), 3.610917 (for h = 0.01 600 rectangles), and 3.610485 (for h = 0.001 for 6000 rectangles).


Simpsons Rule h = 0.1
y values
x values
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
1
2
3
4
5
6


Simpsons Rule h = 0.01
y values
x values
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
1
2
3
4
5
6



4 [Comparison]:
Below are the three numerical integration method's results at h = 0.1.

For the Right Riemann Sum (h = 0.1) : Area = 3.62542
Error = 3.61092 – 3.62542 = - 0.0145
For the Left Riemann Sum (h = 0.1): Area = 3.59299
Error = 3.61092 – 3.59299 = 0.01793
For the Trapezoid Rule (h = 0.1): Area = 3.60921
Error = 3.61092 – 3.60921 = 0.00171
For Simpson's Rule: (h = 0.1): Area = 3.610924
Error = 3.61092 – 3.610924 = -0.000004

With h = 0.1, the error for Simpson's Rule is -0.000004, and the next closest error is the Trapezoid Rule with an error of 0.00171. Therefore, of the three numerical integration techniques presented, Simpson's Rule is most accurate.

5[Backwards, Forwards, Symmetric]:

Given the function $f(x) = \ln(1 + x^2)$, explore the following three possible differentiation techniques on [0,6]. Use the values h = 0.1, 0.01. Remember that the method requires special exceptions on the endpoints.

Given a function $\{f(x_k)\}$, where $\{x_k\} = a + hk$, with $k = 0 \ldots N$ and $h = \frac{b-a}{N}$. The goal is to estimate $\{f'(x_k)\}$.

Backwards:

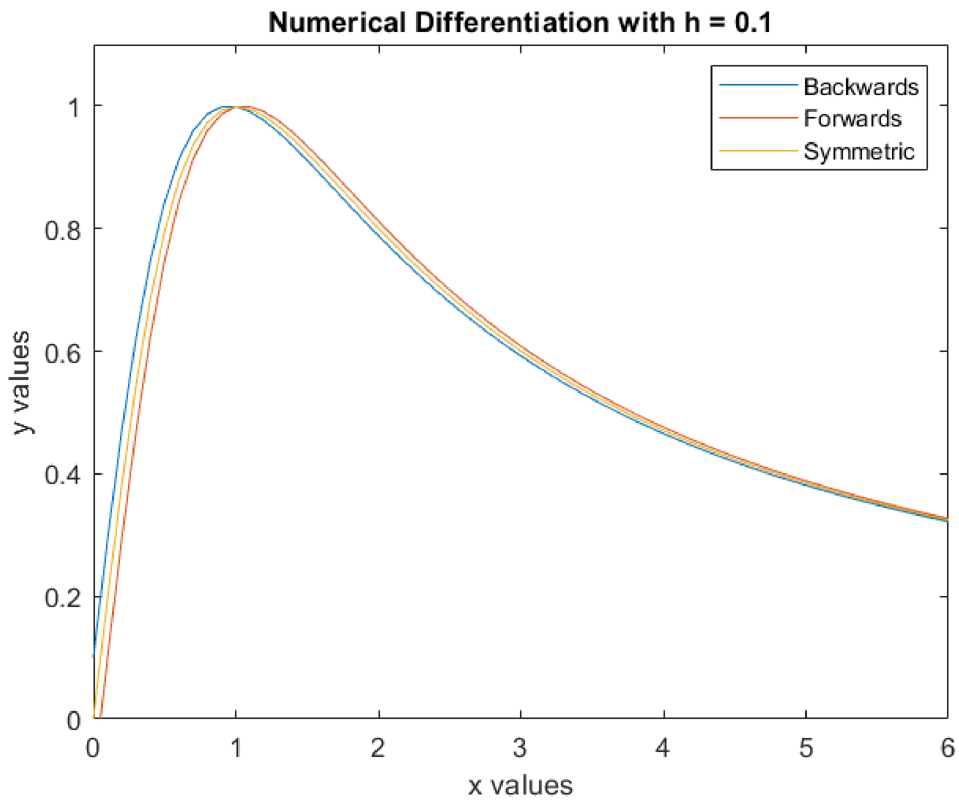
$$f(x_{k-1}) = f(x_k) - f'(x_k)h + f''(\varepsilon_x)\frac{h^2}{2}$$
$$\frac{f(x_{k-1}) - f(x_k)}{h} = f'(x_k) + f''(\varepsilon_x)\frac{h^2}{2}$$

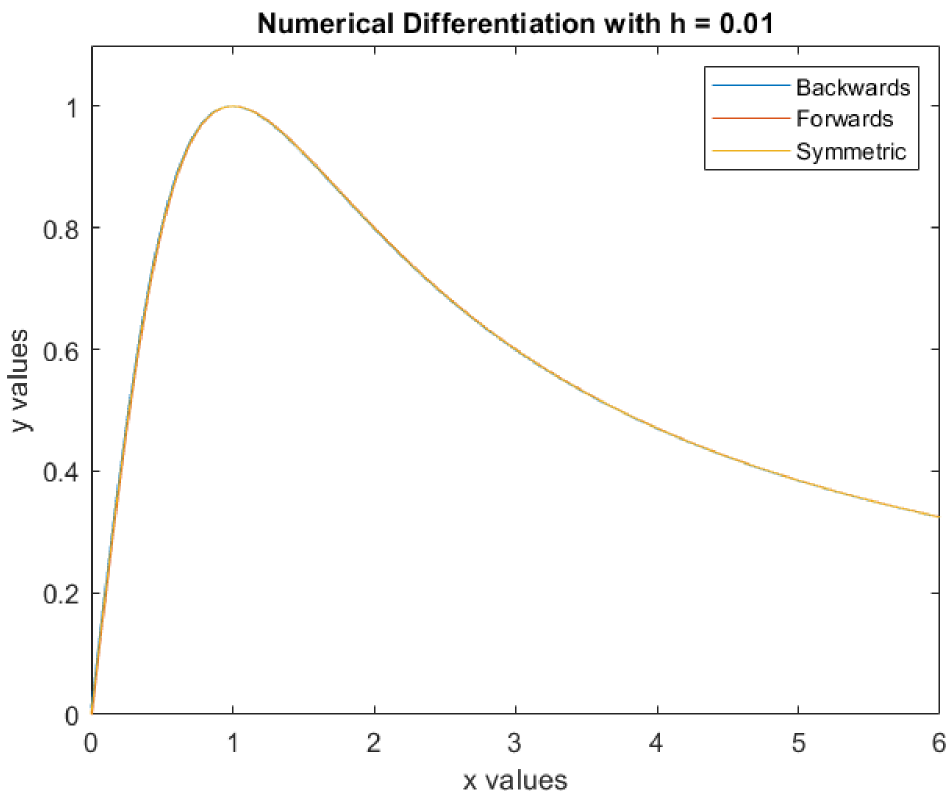
Forwards:

$$f(x_{k+1}) = f(x_k) - f'(x_k)h + f''(\varepsilon_x)\frac{h^2}{2}$$
$$\frac{f(x_{k+1}) - f(x_k)}{h} = f'(x_k) + f''(\varepsilon_x)\frac{h^2}{2}$$

Symmetric:

$$\frac{f(x_{k+1}) - f(x_{k-1})}{2h}$$
$$\frac{1}{2}\left(\frac{f(x_{k+1}) - f(x_k)}{h} + \frac{f(x_{k-1}) - f(x_k)}{h}\right) = \frac{1}{2}\left(\frac{f(x_{k+1}) - f(x_k - 1)}{h}\right) = \frac{f(x_{k+1}) - f(x_{k-1})}{2h}$$


Numerical Differentiation with h = 0.1
Backwards
Forwards
Symmetric
y values
x values
0
0.2
0.4
0.6
0.8
1
0
1
2
3
4
5
6


Numerical Differentiation with h = 0.01
Backwards
Forwards
Symmetric
y values
x values
0
0.2
0.4
0.6
0.8
1
0
1
2
3
4
5
6

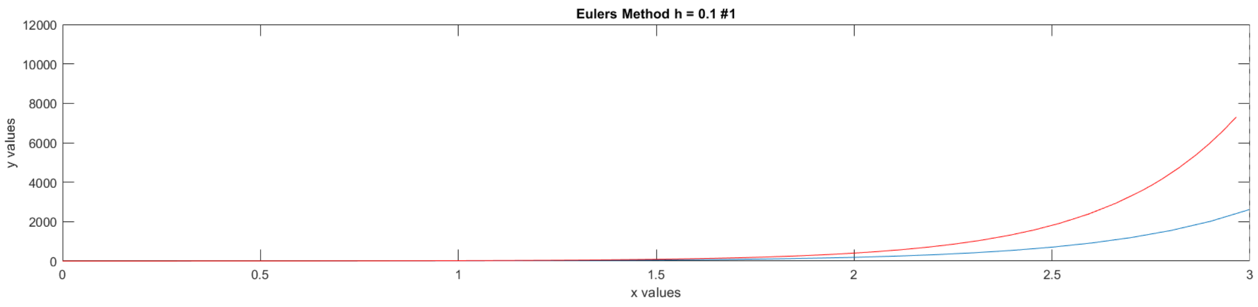
Of the three numerical differentiation techniques, prior to reaching the point x = 1, the order of highest error (in descending order) is backwards, symmetric, forwards. After passing the point, x = 1, the order reverses, and backwards has the lowest error, and forwards has the highest error.

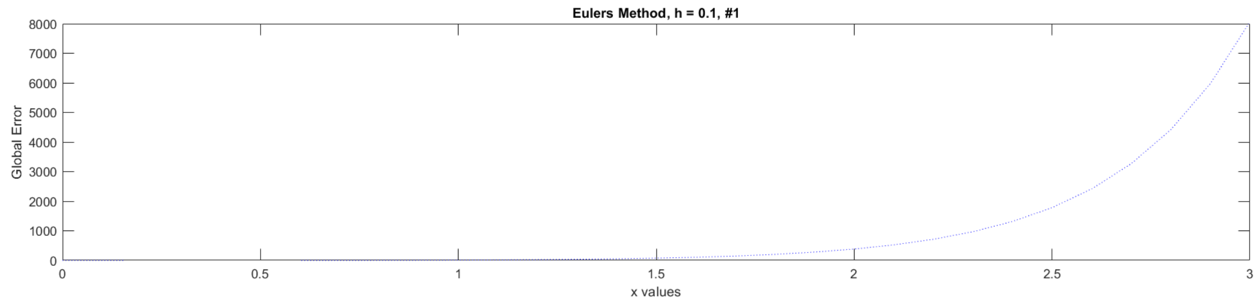
[Numerical ODE Solving Routines]

Consider the following initial value ODE problems, with h = 0.1, 0.01, 0.001. Use the maximum error as the test of accuracy.

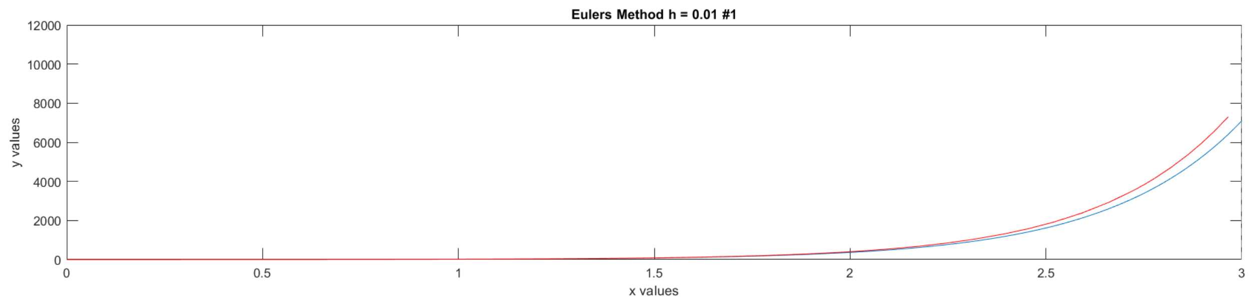
1. $y' = 3y,\ y(0) = 1$ on [0,3]
2. $y' = \frac{1}{1+x^2} - 2y^2, y(0) = 0$ with the solution $y(x) = \frac{x}{1+x^2}$ on [0,10].

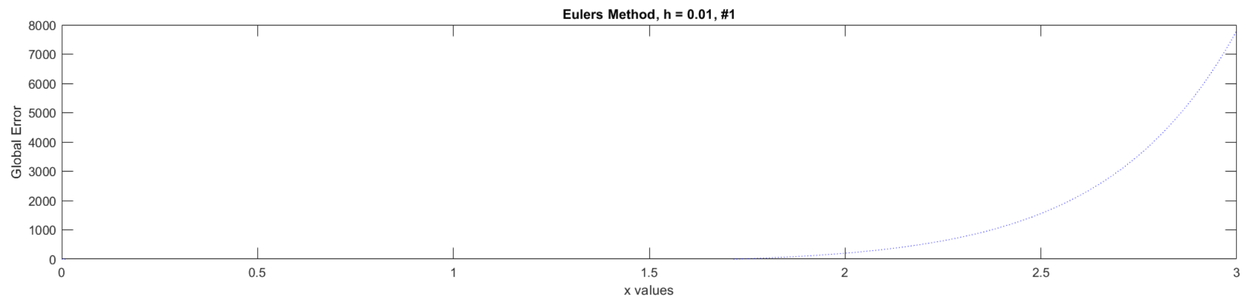
6. [Euler's Method]:



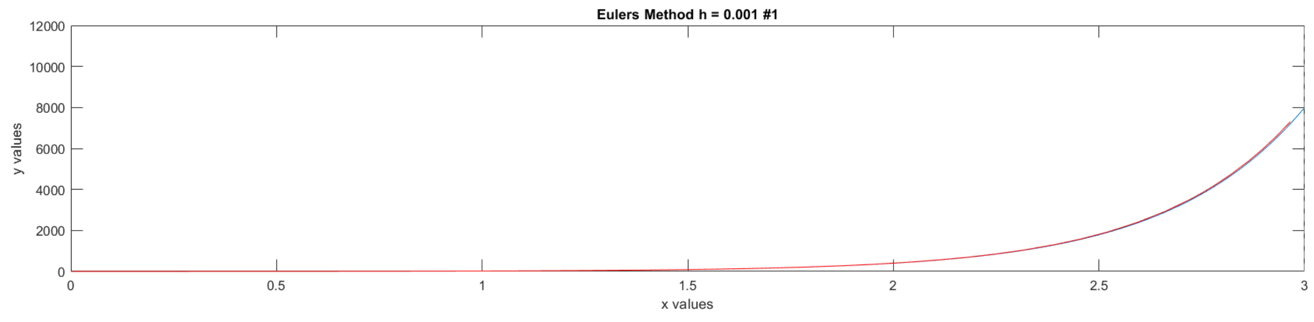


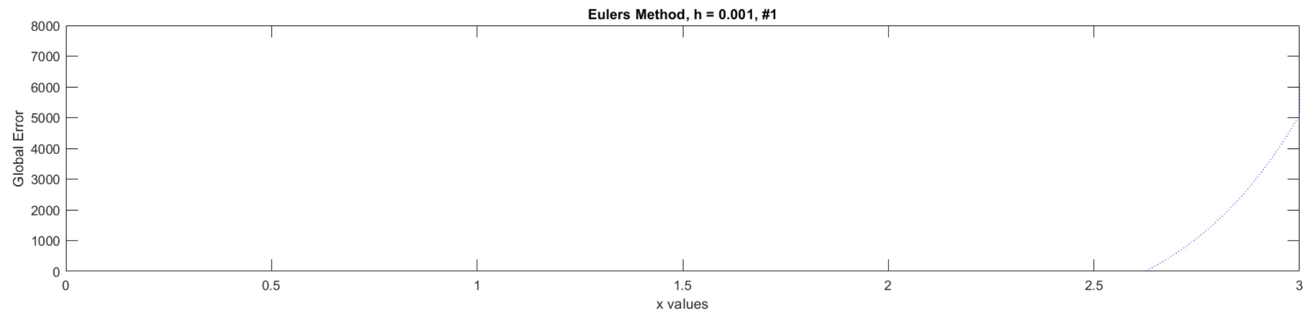
Above: graph produced through Euler's Method, to solve the ODE $y' = 3y,\ y(0) = 1$ on [0,3]. Here, h = 0.1.





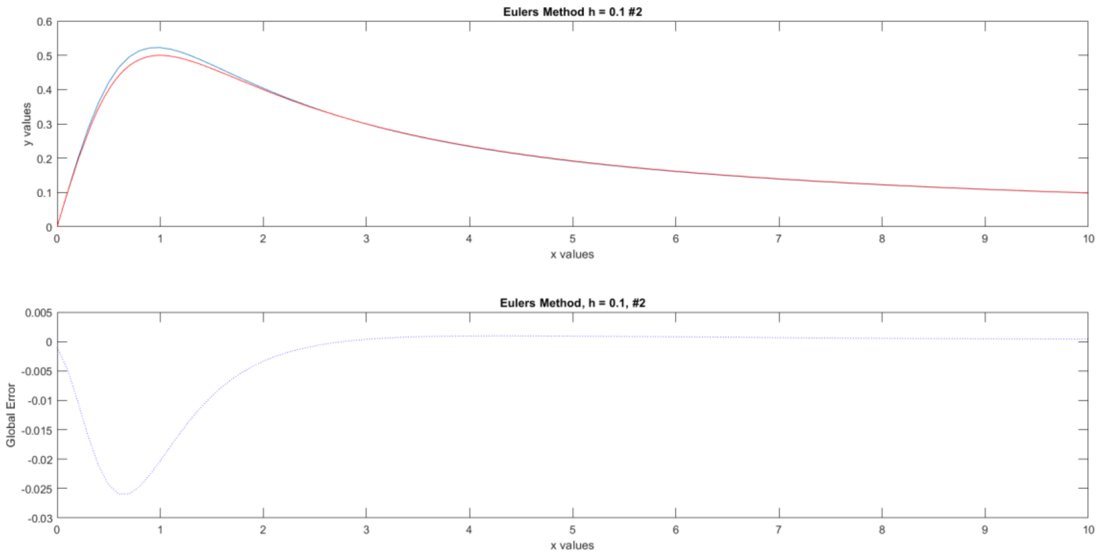
Above: graph produced through Euler's Method, to solve the ODE $y' = 3y,\ y(0) = 1$ on [0,3]. Here, h = 0.01. When compared to the previous graph (h = 0.1), the maximum global error is smaller; the maximum y-value of the lower-graph is lower than the maximum y-value of the previous graph (h = 0.1). Thus, with an increased number of steps taken, the approximation becomes more accurate.



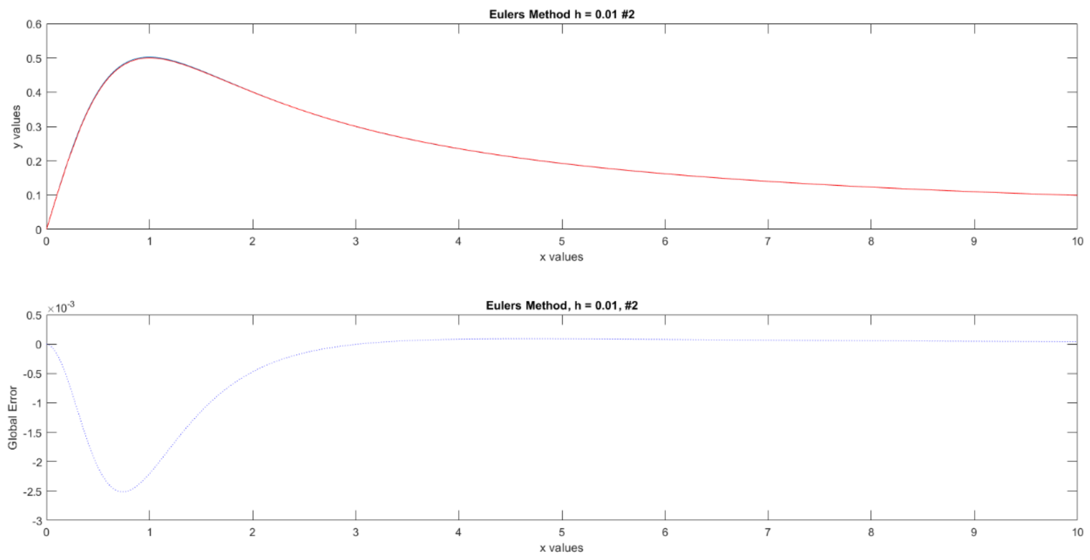


Above: graph produced through Euler's Method, to solve the ODE $y' = 3y,\ y(0) = 1$ on [0,3]. Here, h = 0.001. Similarly, the maximum Global Error when h = 0.001 is smaller than the previous cases (h = 0.1 and h = 0.001). When compared to the previous two graphs (h = 0.1 and h = 0.01), the maximum global error is smaller; the maximum y-value of the lower-graph is lower than the maximum y-value of the previous graphs (h = 0.1 and h = 0.01). Thus, while it is relatively slow compared to the ODE solving methods, it is beneficial to use a smaller h-value with Euler's method.

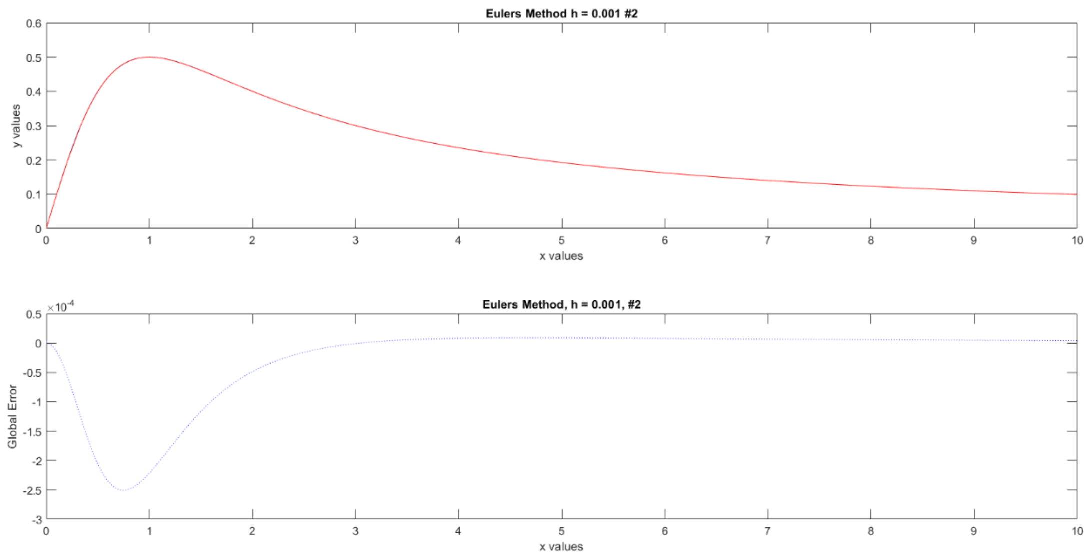
2. Given $y' = \frac{1}{1+x^2} - 2y^2, y(0) = 0$ with the solution $y(x) = \frac{x}{1+x^2}$ on [0,10].



Above: graph produced through Euler's Method, given $y' = \frac{1}{1+x^2} - 2y^2, y(0) = 0$ with the solution $y(x) = \frac{x}{1+x^2}$ on [0,10]. Here, h = 0.1. The blue line is the numerical approximation, and the red line is the value of the function at the given value of x.

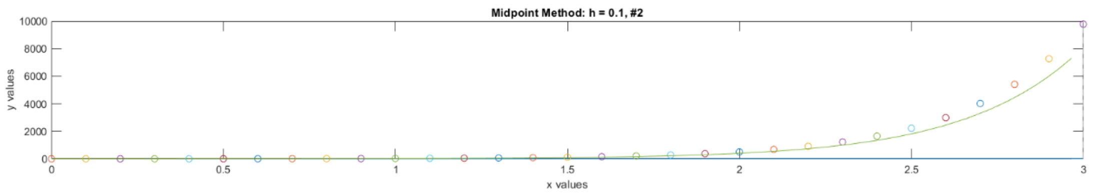


Above: graph produced through Euler's Method, given $y' = \frac{1}{1+x^2} - 2y^2, y(0) = 0$ with the solution $y(x) = \frac{x}{1+x^2}$ on [0,10]. Here, h = 0.01. When compared to the previous graph (h = 0.1), this graph shows a lower maximum error. The distance between the lines that represents the approximation and the actual value have decreased. The maximum error decreased with an increased number of steps.

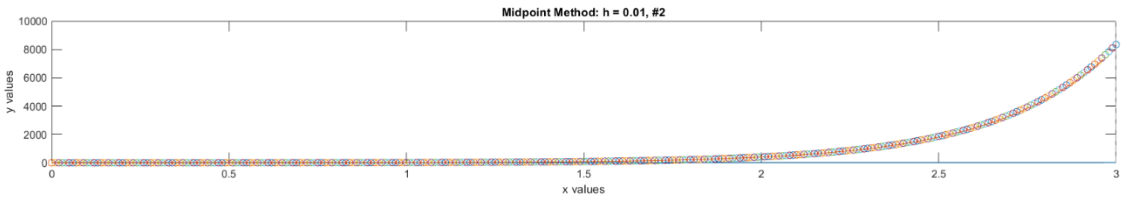


Above: graph produced through Euler's Method, given $y' = \frac{1}{1+x^2} - 2y^2, y(0) = 0$ with the solution $y(x) = \frac{x}{1+x^2}$ on [0,10]. Here, h = 0.001. When compared to the two previous graphs (h = 0.1 and h=0.01), this graph shows a lower maximum error. Of the two previous graphs (h = 0.1 and h = 0.01), this graph reflects the most accurate answer.

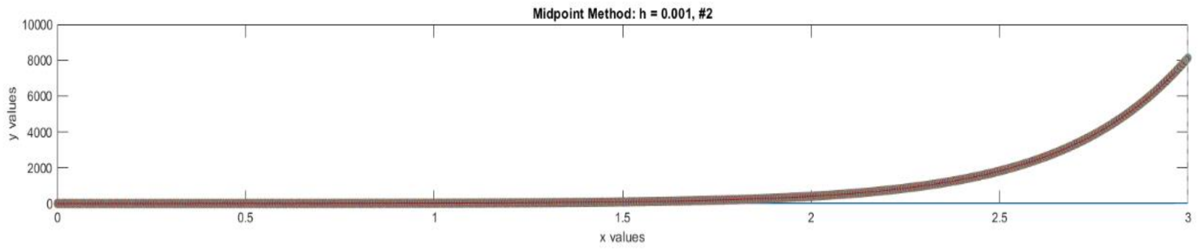
7. [Midpoint Method]:
    1. Given $y' = 3y,\ y(0) = 1$ on [0,3]



Above: The graph generated by solving the first given ODE. Here, h = 0.1, and the separated and dotted line represents the data provided by the midpoint method.

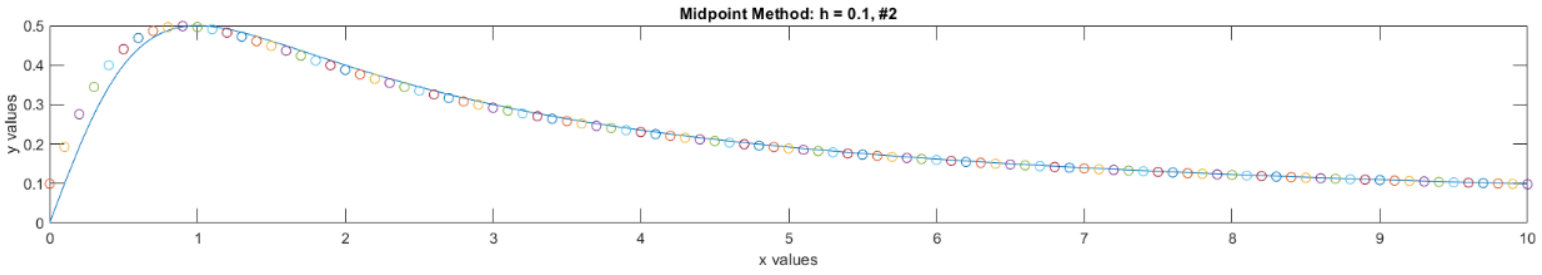


Above: The graph generated by solving the first given ODE. Here, h = 0.01. Compared to the previous graph (h = 0.1), the results generated are more accurate. The distance between the line that represents the numerical solution and the exact result of solving the ODE has decreased, which indicates a lower error.

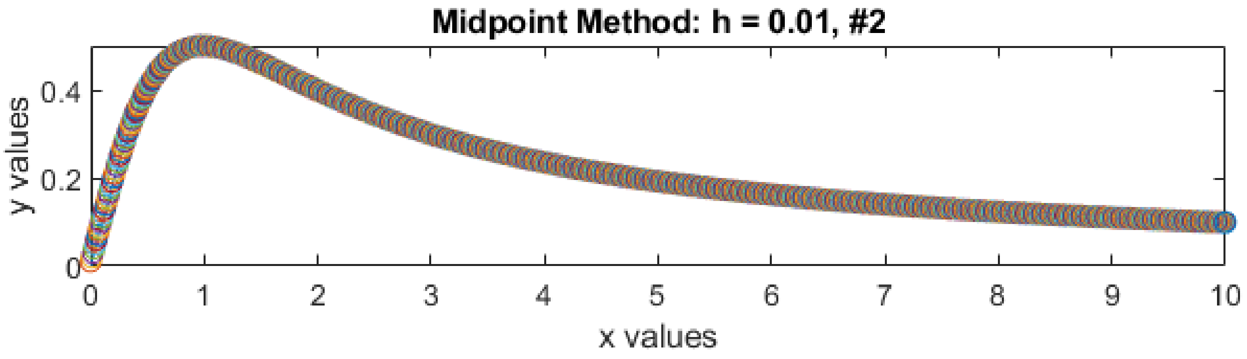


Above: The graph generated by solving the first given ODE. Here, h = 0.001. Compared to the previous graphs (h = 0.1 and h = 0.01), the results generated are even more accurate. The distance between the line that represents the numerical solution and the exact result of solving the ODE has decreased again, which indicates a lower error. Thus, h = 0.001 (6000 steps) is more accurate than h = 0.01 (600 steps) and h = 0.1 (60 steps).

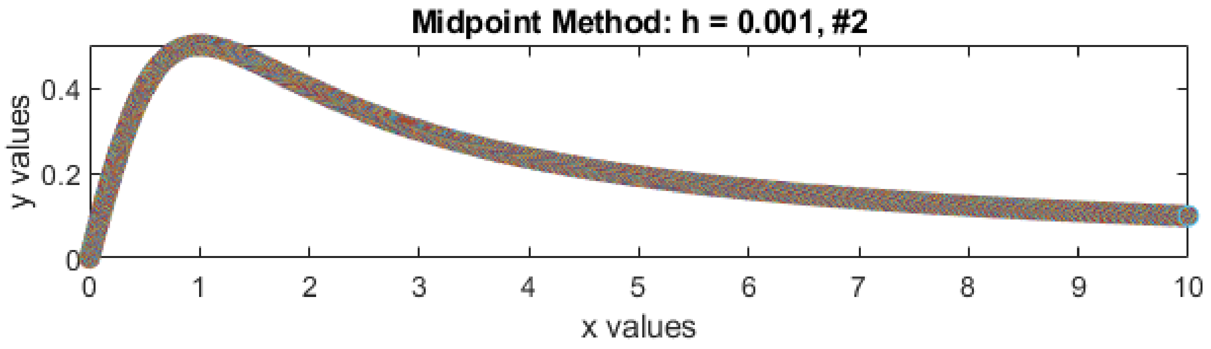
2. Given $y' = \frac{1}{1+x^2} - 2y^2, y(0) = 0$ with the solution $y(x) = \frac{x}{1+x^2}$ on [0,10]



Above: The graph generated by the midpoint method with h = 0.1. In this graph, the solid blue line represents the exact result (or value) at a given point. The line of circled dots represents a data point that corresponds to a step. Here, we see that the greatest error (or difference) between the exact result and the numerical solution is greatest before x = 1, and it gradually decreases as x approaches 10.



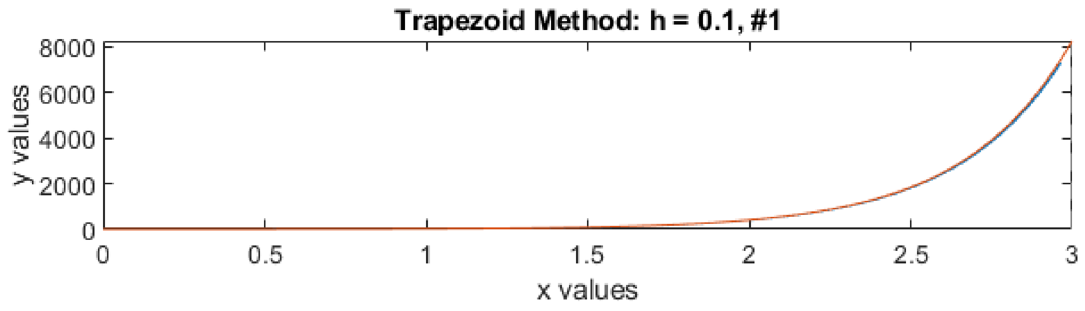
Above: The graph generated by the midpoint method with h = 0.01. In this graph, we notice the blue line, which represents the exact result, is barely visible. The distance between the exact solution's line and the numerical solutions line has decreased. Thus, with the increased number of steps, the accuracy of the solution increases.



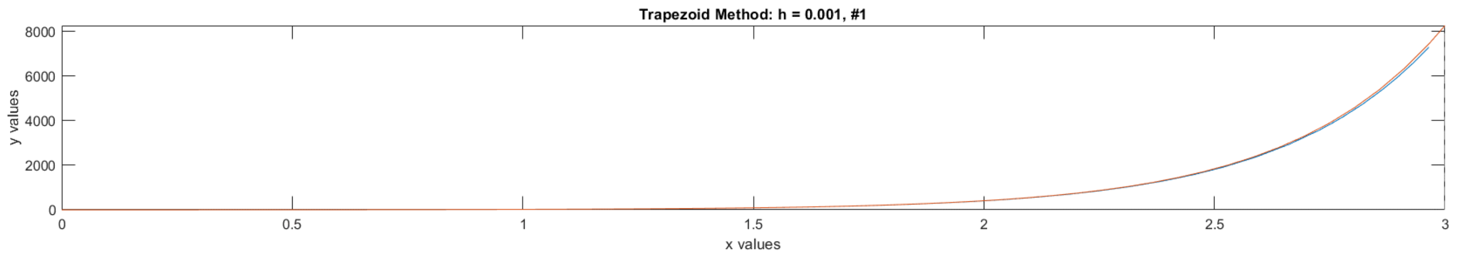
Above: The graph generated by the midpoint method with h = 0.001. In this graph, we notice the blue line, which represents the exact result, becomes even harder to distinguish from the numerical solution line. As expected, with an increased number of steps, the error improved, and the accuracy of the numerical solution increased.

8. [Trapezoid Method]:

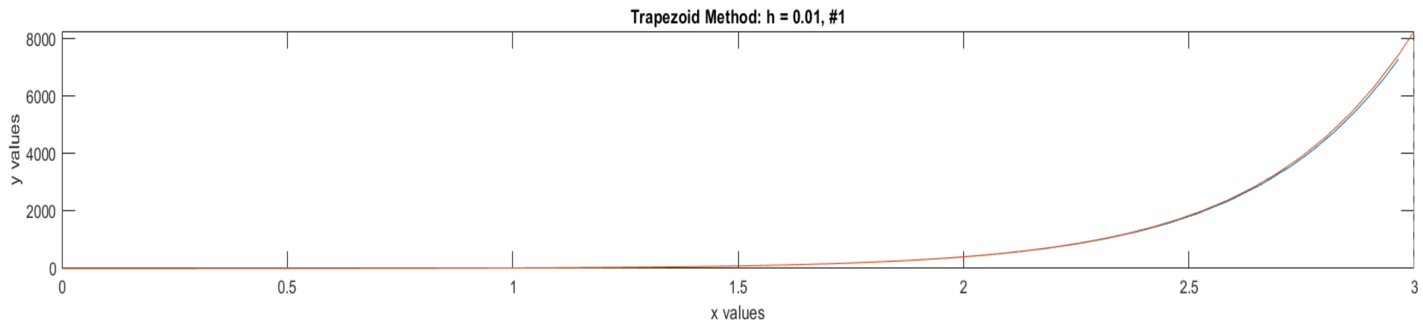
   1. Given $y' = 3y,\ y(0) = 1$ on [0,3]



Above: The Trapezoid method for the first ODE, with h = 0.1., to solve the ODE $y' = 3y,\ y(0) = 1$ on [0,3]. Here, h = 0.01. When compared to the previous graph (h = 0.1), the maximum global error is smaller; the maximum y-value of the lower-graph is lower than the maximum y-value of the previous graph (h = 0.1). Thus, with an increased number of steps taken, the approximation becomes more accurate.

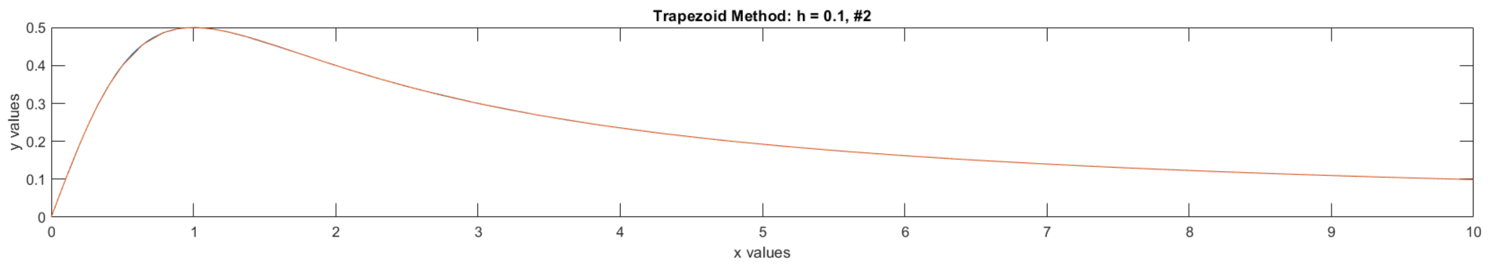


Above: The Trapezoid method for the first ODE, with h = 0.01., to solve the ODE $y' = 3y,\ y(0) = 1$ on [0,3]. When compared to the previous graph (h = 0.1) the maximum global error is smaller; the maximum y-value of the lower-graph is lower than the maximum y-value of the previous graph (h = 0.1). Thus, with an increased number of steps taken, the approximation becomes more accurate.

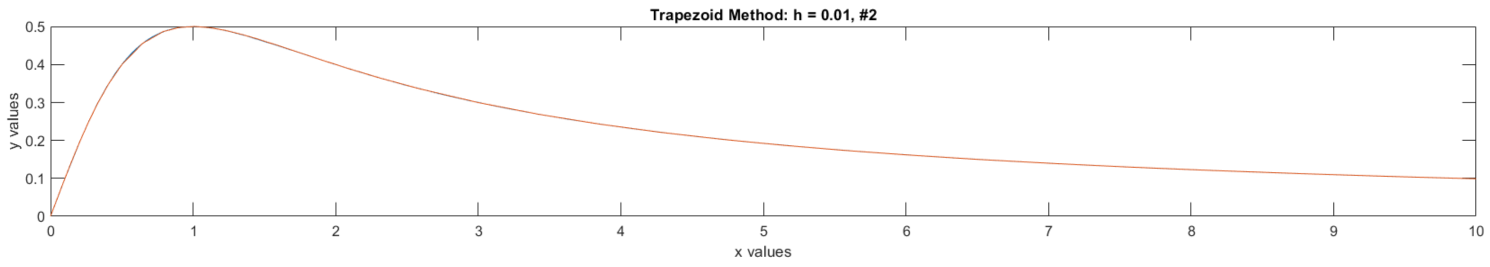


Above: The Trapezoid method for the first ODE, with h = 0.001., to solve the ODE $y' = 3y,\ y(0) = 1$ on [0,3]. When compared to the previous graph (h = 0.1 and h = 0.01), the maximum global error is smaller; the maximum y-value of the lower-graph is lower than the maximum y-value of the previous graphs (h = 0.1 and h = 0.01). Thus, with an increased number of steps taken, the approximation becomes more accurate.

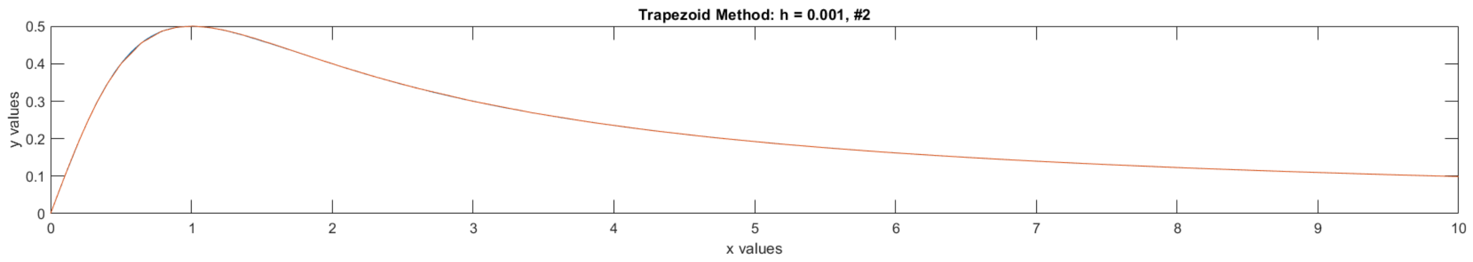
2. Given $y' = \frac{1}{1+x^2} - 2y^2, y(0) = 0$ with the solution $y(x) = \frac{x}{1+x^2}$ on [0,10].



Above, the Trapezoid method for the second ODE, with h = 0.1.



Above: The Trapezoid method for the second ODE, with h = 0.01. In comparison to the previous graph (h = 0.1), the error decreased, and thus the accuracy of the solution increased.

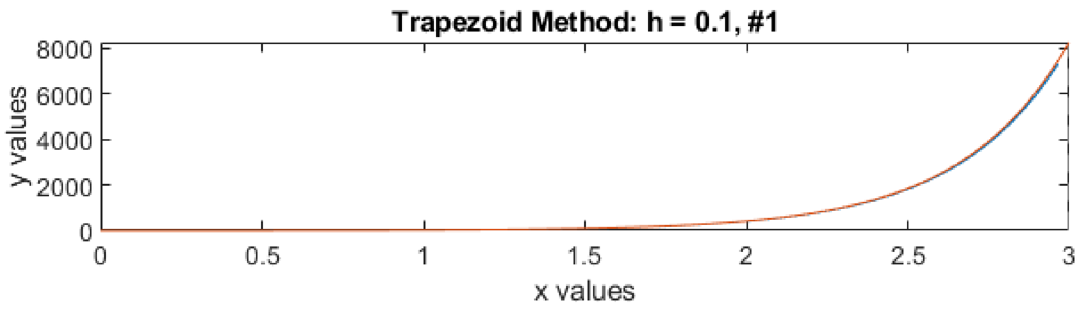


Above: The Trapezoid method for the second ODE, with h = 0.001. In comparison to the previous graph (h = 0.1 and h = 0.01), the error decreased, and thus the accuracy of the solution increased once again.

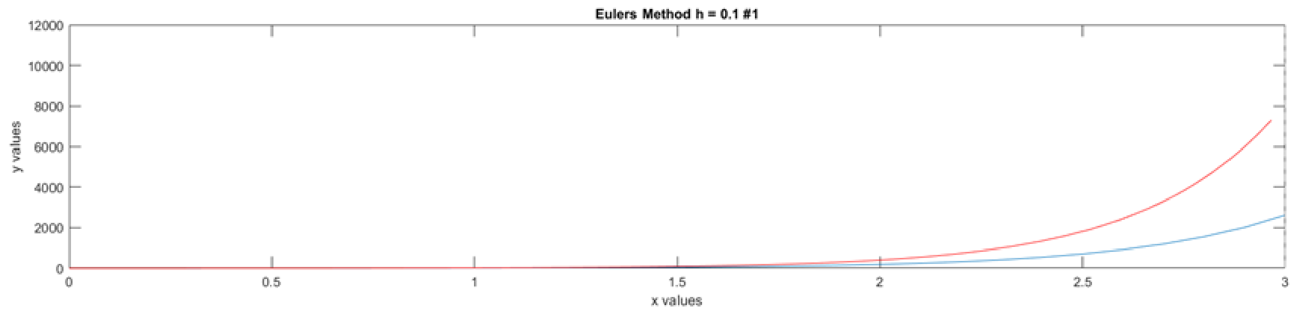
[Comparison]:

In terms of speed, the trapezoid method appears to be the quickest solution for reducing error. However, when comparing maximum potential error, the trapezoid method has a higher potential for a greater error value. Additionally we use $y = e^{3x}$ to find the exact solution.

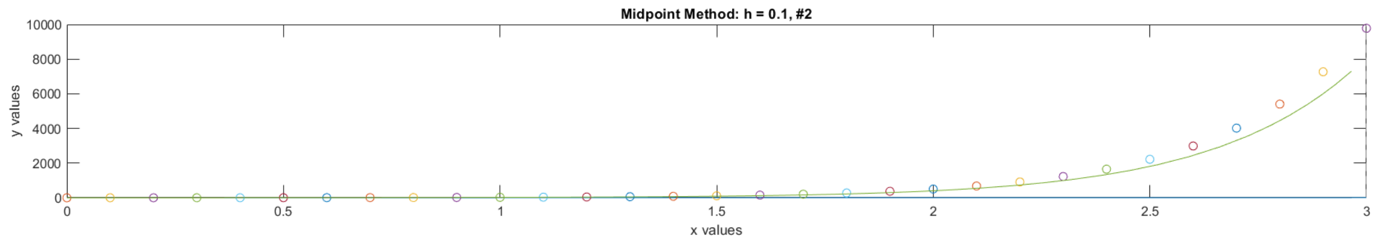
For example, compare the following graphs:



At x = 3, the value of the trapezoid method is: approximately 8000. Error = 8103.8 - ~8000 ~ 100.



At x =3, Euler’s method’s y-value is 2620. Error = 8103.8 – 2620 – 5483.8



At x = 3, the Midpoint method’s y-value is 9789. Error = 9789 – 8103.8 = 1685.2. However, the rate at which the Trapezoid’s method error is increasing is higher.

At x = 3, the Midpoint’s Method’s y-value is the highest of the three methods. However, the rate at which the Trapezoid’s method error is increasing is higher. Therefore, the Midpoint Method is a good compromise, as it is faster than Euler’s method and more accurate than both Euler’s method and the Trapezoid Method. Overall, the midpoint method seems to be the most practical method

Source Code:

1 [Riemann Sums]:

```
function [xeq,yeq,i,value,x,y] = RiemannSums(h)
x = zeros((6/h),1);
y = zeros((6/h),1);
f = @(x) 2.*x/(1+x.^2);
i = 0;
xeq = 1;
yeq = 1;
value = 0;
while (i <= 6)
    value = value + h*f(i);
    x(xeq) = i;
    y(yeq) = value;
    i = i + h;
    xeq = xeq + 1;
    yeq = yeq + 1;
end
x = linspace(h,6,60);
tiledlayout(2,1);
ax1 = nexttile;
fplot(ax1,f,[0,6],'k','LineWidth', 1);
ylim([0 1.2])
title(ax1, 'Right Riemann Sum h = 0.01');
xlabel(ax1, 'x values');
ylabel(ax1, 'y values');
hold on;
i = 0;
while i<6-h
     xz = [i,i,i+h,i+h];
     yz = [0,f(i+h),f(i+h),0];
     plot(xz,yz,'-r','LineWidth', 0.2);
     i = i+h;
end

i = h;
xeq = 1;
yeq = 1;
value = 0;
while (i <= 6)
       value = value + h*f(i-h);
       x(xeq) = i;
```

```
        y(yeq) = value;
        i = i + h;
        xeq = xeq + 1;
        yeq = yeq + 1;
end
ax2 = nexttile;
fplot(ax2,f, [0,6], 'k','LineWidth', 1);
ylim([0 1.2])
title(ax2, 'Left Riemann Sum h = 0.01');
xlabel(ax2, 'x-values');
ylabel(ax2, 'y-values');
hold on;
i = 0;
while i<6-h
     xz = [i,i,i+h,i+h];
     yz = [0,f(i),f(i),0];
     plot(xz,yz,'-r','LineWidth', 0.2);
     i = i+h;
end
```

2 [Trapezoid Rule]:

```
function [xeq,yeq,i,value,x,y] = TrapezoidRule(h)
x = zeros((6/h),1);
y = zeros((6/h),1);
f = @(x) 2.*x/(1+x.^2);
i = 0;
xeq = 1;
yeq = 1;
value = 0;
sum = 0;
while (i <= 6)
    if i == 0
        value = ((f(i))./2);
         x(xeq) = i;
    y(yeq) = value;
    %sum = sum + y(yeq)./2;
    elseif (i <= 6-h)
        value = value + (2*f(i));
          x(xeq) = i;
            y(yeq) = y(yeq) + value;
            sum = sum + y(yeq);
         value = value - (2*f(i));

    else
        value = (value + (f(i)));
         x(xeq) = i;
         y(yeq) = value ;
         sum = sum + value;
         value = value - (f(i) * h);
    end
    %disp(sum);
    i = i + h;
    xeq = xeq + 1;
    yeq = yeq + 1;

end
%disp(value);
sum = sum * (h/2);
disp(sum);
%disp(y);
% p = linspace(h,6,6000);
% q = log(1+p.^2);
% plot(x,y,p,q);
```

```
fplot(f,[0,6],'-r','LineWidth', 3);
ylim([0 1.2])
title('Trapezoidal Rule h = 0.001');
xlabel('x values');
ylabel('y values');
hold on;
i = 0;
while i<6-h
     xz = [i,i,i+h,i+h];
     az = [i,i,i+h,i+h];
     yz = [0,f(i),f(i+h),0];
     bz = [0,f(i),f(i+h),0];
     xaz = (xz + az)./2;
     ybz = (yz + bz)./2;
     plot(xaz,ybz,'b','LineWidth', 1);
     i = i+h;
end
```

3 [Simpson’s Rule]:

```
function [xeq,yeq,i,value,x,y] = SimpsonsRule(h)
x = zeros((6/h),1);
y = zeros((6/h),1);
f = @(x) 2.*x/(1+x.^2);
i = 0;
xeq = 1;
yeq = 1;
value = 0;
sum = 0;
count = 0;
cc = 1;
c = cell(1,6000);
while (i <=6)
    if i == 0
        value = value + f(i);
        y(yeq) = f(i) / 3;
        sum = sum + value;
    elseif i >= 6 -h
        value = value + f(i);
        sum = sum + value;
        y(yeq) = f(i) / 3;
    elseif count == 1
        value = value + 4*f(i);
        y(yeq) = 4*f(i) /3;
        sum = sum + value;
    elseif count == 2
        count = 0;
        value = value + 2*f(i);
        sum = sum + value;
        y(yeq) = 2*f(i) /3 ;
    end
    x(xeq) = i;
    c{cc} = table(x,y);
    i = i + h;
    xeq = xeq + 1;
    yeq = yeq + 1;
    count = count + 1;
    %disp(sum);
    disp(value)
```

```
end
t = vertcat(c{:});
disp(t);
T = table(x,y);
plot(t.x,t.y,'o')
[p,~,mu] = polyfit(t.x,t.y,10);
fs = polyval(p,y,[],mu);
%hold on;
plot (x,y);
%hold off;
disp(value);
disp(value * ((2*h)/6));
disp(yeq);
fplot(f,[0,6],'-r','LineWidth', 3);
ylim([0 1.6])
title('Simpsons Rule h = 0.001');
xlabel('x values');
ylabel('y values');
hold on;
i = 0;
while i<6-h
     xz = [i,i,i+h,i+h];
     az = [i,i,i+h,i+h];
     yz = [0,f(i),f(i+h),0];
     bz = [0,f(i),f(i+h),0];
     xaz = (xz + az)./2;
     ybz = (yz + bz)./2;
     %plot(xaz,ybz,'r','LineWidth', 1);
     i = i+h;
end
```

5 [Backwards, Forwards, Symmetric]:

```
function BFS(h)
x = zeros((6/h),1);
y = zeros((6/h),1);
f = @(x) log(1+x.^2);
i = 0;
xeq = 1;
yeq = 1;
%backwards difference formula
while (i <= 6)
    ii = i + h;
    x(xeq) = i;
    y(yeq) = (f(ii)-f(i))/h;
    i = i + h;
    xeq = xeq + 1;
    yeq = yeq + 1;
    disp(y);
end
disp('split here');
%forwards difference formula
xf = zeros((6/h),1);
yf = zeros((6/h),1);
xeq = 1;
yeq = 1;
i = 0;
while (i <= 6)
    ii = i - h;
    xf(xeq) = i;
    yf(yeq) = (f(i)-f(ii))/h;
    i = i + h;
    xeq = xeq + 1;
    yeq = yeq + 1;
    disp(y);
end

%symmetric difference formula
xs = xf;
ys = (yf + y)/2;
plot(x,y,xf,yf,xs,ys);
ylim([0 1.1]);
title('Numerical Differentiation with h = 0.1');
xlabel('x values');
ylabel('y values');
```

```
legend('Backwards','Forwards','Symmetric');
```

6 [Euler’s Method]:

```
function Euler(ti,tf,h,y0)
%F = (@(x,y) (1./(1+x.^2)-2*y.^2));
F = @(x,y)(3*y);
x = 0:h:tf;
y = zeros(size(x));
y(1)  = y0;
n = numel(y);
disp(n);
for i=1:n
    %f = ((1./(1+x(i).^2))-2*y(i).^2);
    %disp(f);
    %disp(y);
    y(i+1) = y(i) + h*F(x(i),y(i));
    x(i+1) = x(i)+ h;
end
disp(y(31));
tiledlayout(2,1);
ax1 = nexttile;
plot(ax1,x,y);
xlim([0 10]);
xlabel(ax1,'x values');
ylabel(ax1,'y values');
title(ax1,'Eulers Method h = 0.001 #2');
hold on;
fplot(ax1,@(x)(x/(1+x^2)),[0,10],'r');

F2 = @(x) (x/(1+x.^2));
for i=1:n
    y(i) = F2(x(i+1)) - y(i+1);
    y(i+1) = y(i) + y(i)* h;
   %i = i + 1;
end
ax2 = nexttile;

plot(ax2,x,y,':b');

title(ax2, 'Eulers Method, h = 0.001, #2');
xlabel(ax2,'x values');
ylabel(ax2, 'Global Error');
xlim([0 10]);
```

7 [Midpoint Method]:

```
function yout =  midpoint(F,t0,h,tfinal,y0)
 %F = @(x,y)(1./(1+x.^2))-2*y.^2;
 %F = @(t,y)(3*y*t);

  tiledlayout(3,1);
 ax1 = nexttile;
 fplot(ax1,@(y)(3*y),[0,3])
 %fplot(ax1,@(x)(x/(1+x^2)),[0,10]);
y = y0;
yout = y;
for x = t0 : h : tfinal
    s1 = F(x,y);
    ymid = y + h*s1/2;
    s2 = F(x+h/2, ymid);
    y = y + h*s2;
    yout = [yout; y];
    hold on;
    plot(ax1,x,y,'o');

%     disp(y);
%     plot(t,y);
end
hold off;
disp(y);
disp(x);



  xlim([0,10]);
  %ylim([0,1.2]);
  xlabel(ax1,'x values');
 ylabel(ax1,'y values');
 title(ax1,'Midpoint Method: h = 0.001, #2');
 hold on;
 %fplot(ax1,@(x)(exp(x*3)),[0,3]);
 %fplot(ax1,@(x)(x/(1+x^.2)),[0,10]);
 hold off;
%  x = 0;
%  h = h * 0.1; % 0.01
%  x = a:h/2:b;
%  y = 0;
%  y = zeros(size(x));
%  y(1)=0;
%   for n=1:length(x)-1
%      %y(n) = y(n-1) + h*F(x(n-1) + h/2, y(n-1) +
(h/2)*F(x(n-1),y(n-1)));
%      y(n+1) = y(n) + h*F(x(n) + h/2, y(n) +
(h/2)*F(x(n),y(n)));
%   end
%  ax2 = nexttile;
%  plot(ax2,x,y,':ro','LineWidth', 0.5);
%  xlabel(ax2,'x values');
%  ylabel(ax2,'y values');
%  title(ax2,'Midpoint Method: h = 0.01, #2');
%
%  x = 0;
%  h = h * 0.1; % 0.01
%  x = a:h/2:b;
%  y = 0;
%  y = zeros(size(x));
%  y(1)=0;
%   for n=1:length(x)-1
%      %y(n) = y(n-1) + h*F(x(n-1) + h/2, y(n-1) +
(h/2)*F(x(n-1),y(n-1)));
%      y(n+1) = y(n) + h*F(x(n) + h/2, y(n) +
(h/2)*F(x(n),y(n)));
%   end
%  ax3 = nexttile;
%  plot(ax3,x,y,':ro','LineWidth', 0.5);
%  xlabel(ax3,'x values');
%  ylabel(ax3,'y values');
%  title(ax3,'Midpoint Method: h = 0.001, #2');
%
```

8 [Trapezoid Method]:

```
function TrapezoidMethod(F,t0,h,tf,y0)
 tiledlayout(3,1);
 ax1 = nexttile;
 fplot(ax1,@(y)(exp(3*y)),[0,3]);
 %fplot(ax1,@(x)(x/(1+x.^2)), [0,10]);
hold on;
tspan = [t0 tf];
[t,y] = ode23t(F,tspan,y0);
disp(y(15));
plot(ax1,t,y);
xlabel(ax1,'x values');
 ylabel(ax1,'y values');
 title(ax1,'Trapezoid Method: h = 0.1, #1');
 hold off;
%midpoint(@(x,y)(1./(1+x.^2))-2*y.^2,0)
%fplot(ax1,@(y,x)exp(3*x),[0,3], 0.1, 3, 1);
%Trapezoid(@(y)(3*y),0,0.1,3,1);
TrapezoidMethod(@(y)(3*y),0,0.1,3,1);
 %TrapezoidMethod(@(x,y)(3*y),0,0.1,3,1);
```