MAT 128C - 1st Order IVP Unit Testing

Douglas Sherman #: 913348406

April 21st, 2017

To test that the numerical computations are returning accurate numerical approximations, we test on a variety of known solutions. Moreover, by ensuring that the error does descrease according to the expected local truncation error by Taylor's theorem for each method, we can instill more confidence in the result. Consider the following initial value problems (IVP).

$\mathbf{y' = f(t, y(t))}$	$\mathbf{y(t_0) = y_0}$	$\mathbf{y(t)}$
$y' = 2y(t)/t$	$y(-1) = 3$	$y(t) = 3t^2$
$y' = 2t^2 y(t)$	$y(0) = 2$	$y(t) = 2e^{2t^3/3}$
$y' = ty(t)^2$	$y(-1) = 1$	$y(t) = 2/(3 - t^2)$

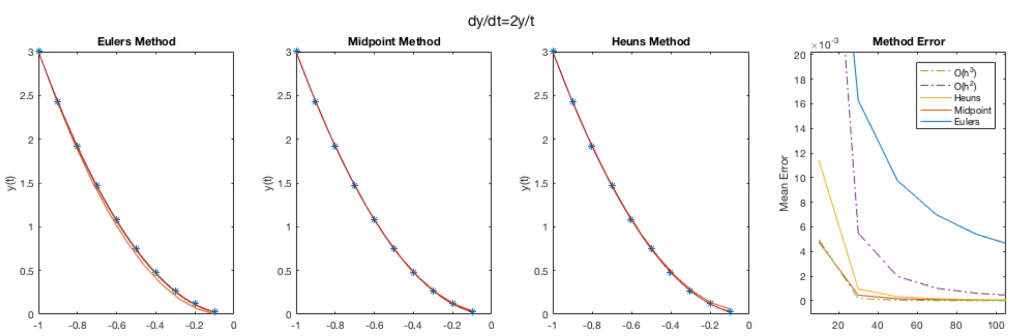
1. Selection of known Initial Value Problems using SymPy for Confirmation

Consistency of $\mathbf{Y(t)}$

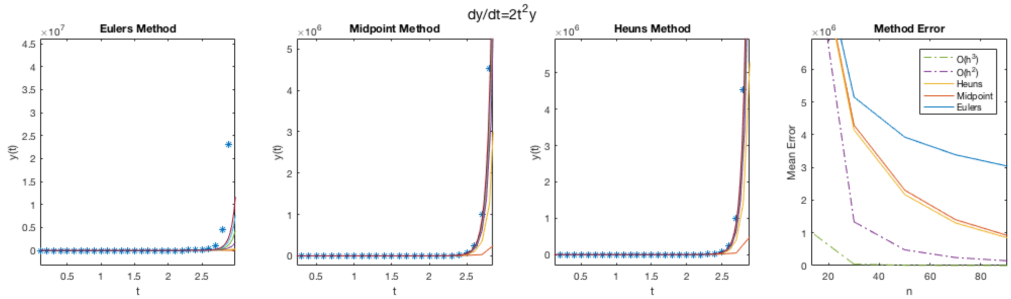
Once the known IVPs were determined we tested the average accuracy of different numerical approximation method including Euler's Method, the midpoint method, and Heun's Method. The accuracy was measured by

$$\sum_{i=1}^{n} \frac{1}{n} |y(t_k) - y_k|$$

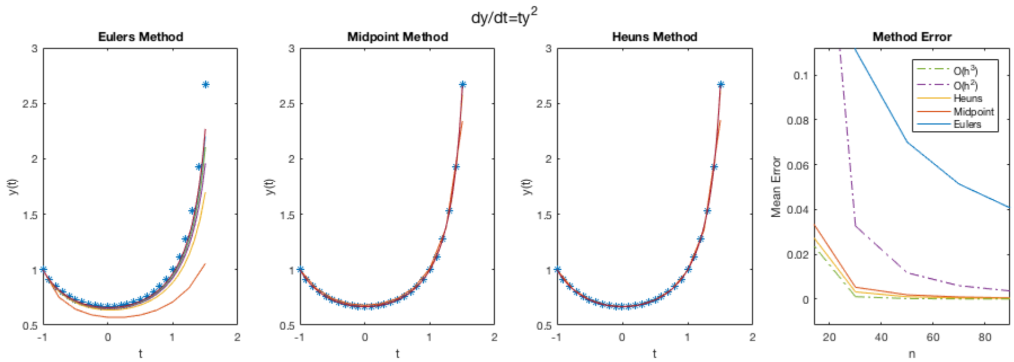
where $n = (b - a)/h$. The Following illustrates the accuracy for the various number of intervals n.



1. Comparison of Multiple Methods for $y' = 2y/t$



2. Comparison of Multiple Methods for $y' = 2t^2y$



3. Comparison of Multiple Methods for $y' = ty^2$

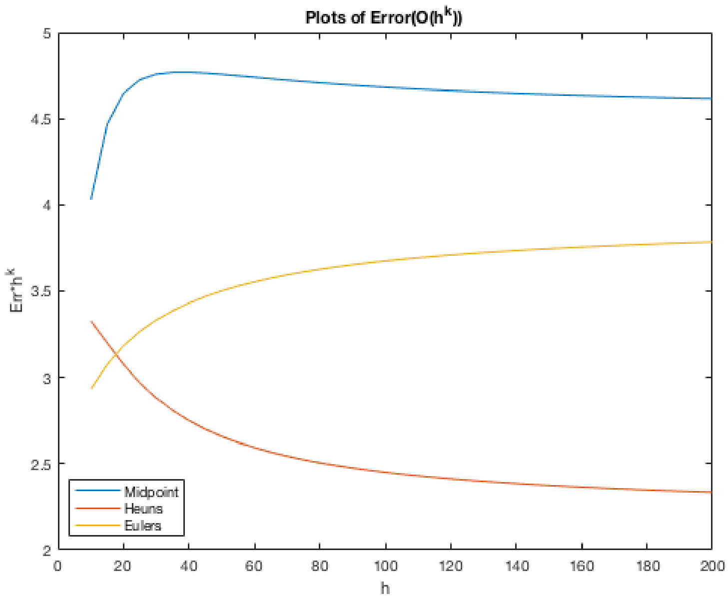
Here we see that the known solution to each IVP shows good agreement with the approximated solution for different values of h. Moreover, we see the smaller the h the better the solution, which supports that the numerical solutions are behaving appropriately.

Local Truncation Error

Another method we can use to verify our methodology is correct is to compare the local truncation error. For example, in Euler's method we obtain,

$$\left| \frac{y(t_{k+1}) - y(t_k)}{h} - f(t_k, y(t_k)) \right| = |hy''(\xi)| = \mathcal{O}(h)$$

So the error of Euler's method is $\mathcal{O}(h)$, and similarly both Midpoint and Heun's method are $\mathcal{O}(h^2)$. Trivially, we see that the error in Heun's and the Midpoint method converge to 0 much quicker than Euler's, but we can analyze the error in detail to support the correctness of our algorithms. Below displays Error$\times h^k$ where $k = 1$ for Euler's and 2 otherwise.



4. Error of Each Method Times its Truncation Error Order

Here we see that the error multiplied by its truncation error order converges to a constant value confirming that each algorithm does indeed follow the model's expected truncation error. Thus, between plots above illustrating convergence to $y(t)$ and the error plot illustrating that this convergence happens at the expected rate, we have significant confidence in each model's correctness.

Appendix

The following are the algorithms implimented for Euler's, Midpoint, and Heun's method.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Impliments Eulers Method   %
% for approximating an IVP   %
% y' = f(t,y(t)), y(a)=y0    %
% across n intervals for     %
% t in [a,b]                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [y,t] = Eulers(f,y0,a,b,n)
    h = (b-a)/n;
    y = ones(1,n+1).*y0;
    t = a+(0:n)*h;
    for k = 1:n
        y(k+1) = y(k) + h*f(t(k),y(k));
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Impliments Midpoint Method %
% for approximating an IVP   %
% y' = f(t,y(t)), y(a)=y0    %
% across n intervals for     %
% t in [a,b]                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [y,t] = Midpoint(f,y0,a,b,n)
    h = (b-a)/n;
    y = ones(1,2*n+1).*y0;
    t = a+(0:2*n)*h/2;
    for k = 2:2:2*n
        y(k)   = y(k-1) + h/2*f(t(k-1),y(k-1));
        y(k+1) = y(k-1) + h*f(t(k),y(k));
    end
    y = y(1:2:length(y));
    t = t(1:2:length(t));
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Impliments Heun's Method   %
% for approximating an IVP   %
% y' = f(t,y(t)), y(a)=y0    %
% across n intervals for     %
% t in [a,b]                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [y,t] = Heuns(f,y0,a,b,n)
    h = (b-a)/n;
    y = ones(1,n+1).*y0;
    t = a+(0:n)*h;
    for k = 1:n
        y(k+1) = y(k) + h*f(t(k),y(k));
        y(k+1) = y(k) + h/2*(f(t(k),y(k))+f(t(k+1),y(k+1)));
    end
end
```