PENN STATE UNIVERSITY
COLLEGE OF ENGINEERING
DEPARTMENT OF AEROSPACE ENGINEERING

AERSP 312
PROJECT 1

NUMERICAL SOLUTIONS TO THE FALKNER-SKAN EQUATIONS

Date: 18 March 2019
Section: 001
By: Brody Clark, Brett Sheeran, John Wychock, Kartik Kamdar, Joseph Cioffi, Logan Morrow, Nicholas DeCandia

PROBLEM STATEMENT

The purpose of this project was to write a computer code that evaluates the numerical solutions of the Falkner-Skan equation for different values of beta. Given, Equation 1, is as follows:

$$\frac{d^3G}{d\eta^3} + G\frac{d^2G}{d\eta^2} + \beta\left[1 - \left(\frac{dG}{d\eta}\right)^2\right] = 0, \tag{1}$$

with initial conditions:

$$G(0) = 0, \quad \frac{dG}{d\eta}(0) = 0, \quad \frac{dG}{d\eta}(\eta) \to 1, \text{ as, } \eta \to \infty$$

The x component of the velocity is given by the equation $u=u_e(df/d\eta)$
Where η (Eta) is given in Equation 2:

$$\eta = y\sqrt{\frac{U_\infty(m+1)}{2\nu L}}\left(\frac{x}{L}\right)^{(m-1)/2} \tag{2}$$

The Improved Euler method must be used in place of ode45 for solving for df/dη. Beta was evaluated at the following values: -0.19884 , -0.180, 0.000, 0.300, 1.000, and 1.33

DERIVATION

While investigating the effects of a pressure gradient on a Boundary layer, Falkner and Skan were able to find similarity solutions for the boundary layer equations when the freestream velocity was given in the form of Equation 3.

$$Ue(x) = U_{\infty}(x/L)^m \tag{3}$$

The two-dimensional laminar boundary layer equations are given as follows, where Equation 4 is referred to as the momentum equation and Equation 5 is the continuity equation.

$$u\frac{du}{dx} + v\frac{du}{dy} = -\frac{1}{\rho}\frac{dp}{dx} + \nu\frac{d^2u}{dy^2} \tag{4}$$

$$\frac{du}{dx} + \frac{dv}{dy} = 0 \tag{5}$$

Assume the relevant boundary conditions for a flat plate, i.e, y= 0 u=0: v=0; u $\rightarrow$ as y $\rightarrow \infty$. Introducing stream functions for u and v yields the following Equations, 6 & 7 respectively.

$$u = \frac{d\psi}{dy}, v = -\frac{d\psi}{dx} \tag{6}$$

$$\frac{d\psi}{dy}\frac{d^2\psi}{dxdy} - \frac{d\psi}{dx}\frac{d^2\psi}{dxdy^2} = U\frac{dU}{dx} + \nu\frac{d^3\psi}{dy^3} \tag{7}$$

Applying the similarity variable from Equation 2 and integrating Equation 7 yields the following results for stream function and similarity variable, Equations 8 & 9 respectively.

$$\psi = x^{\frac{1+m}{2}}\sqrt{\frac{2\nu a}{1+m}} * f(\eta) \tag{8}$$

$$\eta = \frac{y}{x^{\frac{1-m}{2}}}\sqrt{\frac{(1+m)a}{2\nu}} \tag{9}$$

Substituting Equations 8 & 9 into Equation 7 yields the Falkner Skan equation, Equation 1, with the indicated boundary conditions.

A function that satisfies Laplace's equation can be considered as a solution to a given potential flow problem. Consider the following streamfunction, where ψ_0 and n are constants :

$$\psi(r,\theta) = \psi_0 r^n sin(n\theta) \tag{10}$$

Computing the velocity components allows for the determination of streamline direction and determination of stagnation point locations. Differentiating Equation 10 yields the following velocity components:

$$u_r = \frac{1}{r}\frac{d\psi}{d\theta} = n\psi_0 r^{n-1} cos(n\theta) \tag{11}$$

$$u_\theta = -\frac{d\psi}{dr} = -n\psi_0 r^{n-1} sin(n\theta) \tag{12}$$

To determine body shape, start with $\psi = 0$ which yields $\theta = \frac{m\pi}{n}$, where m is an integer. When the situation is confined to the cases for m = 0,1 the wedge-shape is defined between $\theta = 0$and $\theta = \frac{\pi}{n}$. The circumferential velocity along the sides of the wedge, Equation 12, is zero. The radial velocity then depends on radial distance and is defined as such:

$$u_r(r,0) = n\psi_0 r^{n-1},\ u_r(r,\frac{\pi}{n}) = -n\psi_0 r^{n-1} \tag{13}$$

Using three streamlines of m= -1,0,1 gives the flow off of a sharp trailing edge, assuming ψ_0>0. This situation is depicted in Figure 1.

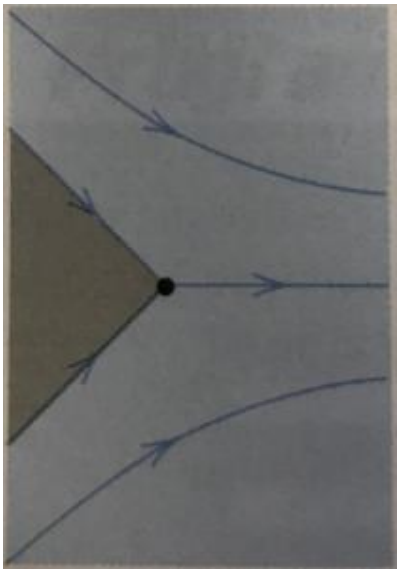

Figure 1. Trailing-edge flow

SUMMARY

The solution to the Falkner-Skan equation involves numerical methods of integration. In this instance, the Improved Euler Method was used to find values for $dF/d\eta$ as η goes from 0 to 10. A bisection function was also used to find an initial condition for $d^2F/d\eta^2$ that would allow this $dF/d\eta$ to converge to 1 at large values of η.

In Figure 2 below, the algorithm is drawn out in the form of a flowchart to demonstrate the process of finding the proper solutions. The initial conditions and step sizes are given and then the Improved Euler Method function is used to update the initial values. The largest value of $dF/d\eta$ is recorded and compared to 1. This information is then passed to the bisection function to update $d^2F/d\eta^2$ accordingly. Once this is completed, the program returns to the start of the process with new initial conditions. The process continues until $dF/d\eta(\eta_{max}) = 1$ or until the reiteration limit of 100 is reached.

This limit produced sufficient results within immeasurable differences from previous iterations. Once satisfactory results for $dF/d\eta$ were obtained, the data was written to a file and plotted.

The freestream velocity follows a power law (given in equation 3 above) with the parameter $\beta=2m/(m+1)$. The parameter β is the measure of the wedge's half angle. The wedge generates potential flow ($u_e(x)$) on its surface. (Figure 2)

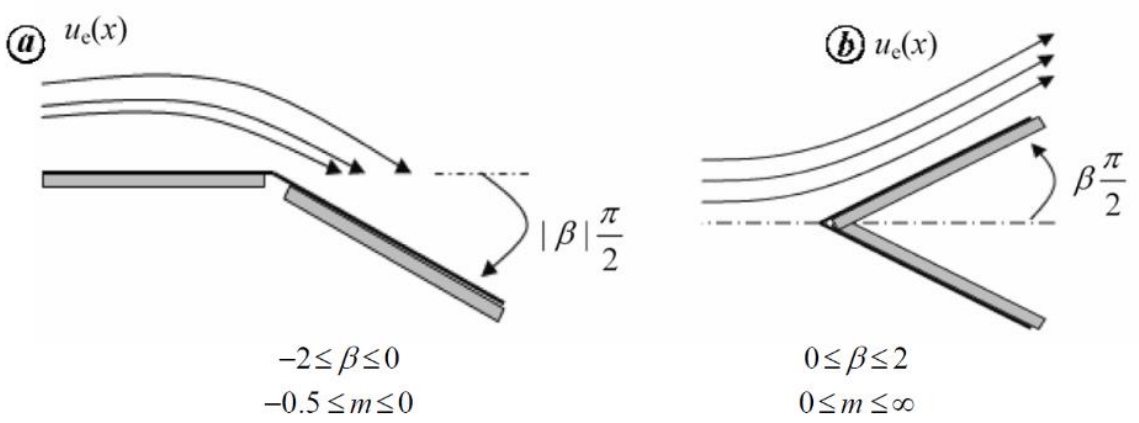

Figure 2. Potential Flows over Wedge

If values of β are less than zero (part a of Figure 2) then there is an adverse pressure gradient and if β is greater than zero (part b of Figure 1) there is a favorable pressure gradient. Both favorable and adverse pressure gradients were used to compare the potential flows over the wedge's surface. In order to determine the potential flow, first dF/dη needs to be found. It is known that the x-component of the velocity off the wedge is $u = u_e (dF/d\eta)$. So, when dF/dη is plugged into the power law equation (equation 3 from above) it yields the potential flow over the wedges surface.

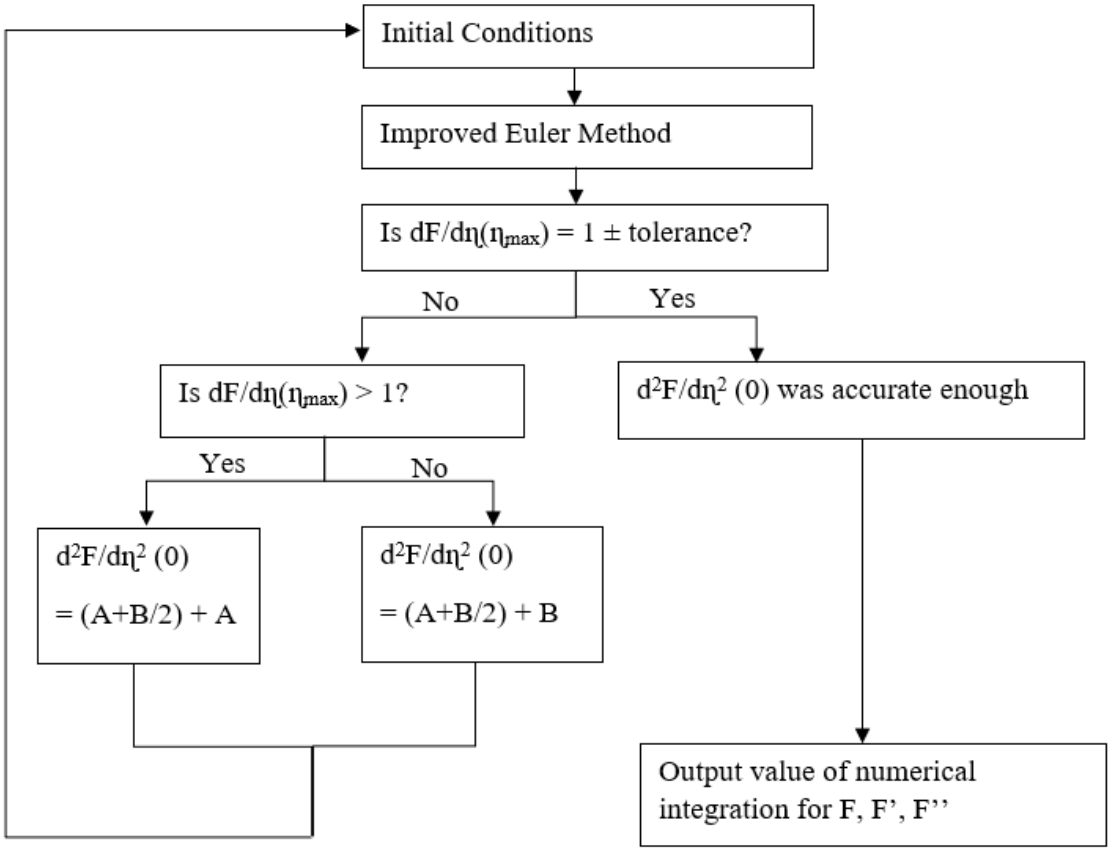

Figure 3. Algorithm flow chart

RESULTS

Using the Improved Euler Method to integrate the Falkner-Skan equation for various values of Beta produced velocity profiles with similar results. As seen in Figure 4 below, the plots of the velocity profiles overlap. The initial condition for the profile is that $dF/d\eta\ (0) = 0$ and as $\eta \rightarrow \infty$, $dF/d\eta$ approaches 1. This convergence was easy to obtain with the lower values of beta for relatively larger step sizes such as 0.3 and narrow limits for $d^2F/d\eta^2(0)$ such as A=0 and B=1. As beta grew, the requirements for convergence became more strict. At beta values of 1 and 1.33, the step size needed to be reduced to 0.1 and the upper limit of $d^2F/d\eta^2(0)$ needed to be raised to 2. This allowed the bisection function to find an initial value for $d^2F/d\eta^2$ which allowed $dF/d\eta$ to converge as η became large.

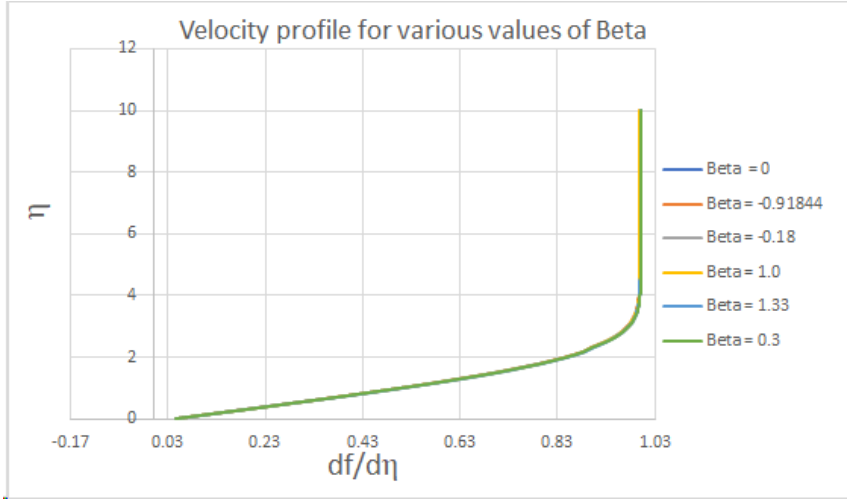

Figure 4. Comparison of velocity profiles with varying Beta values

One special case of the Falkner-Skan equation is when beta = 0. This condition represents flow over a flat plate, known as the Blasius solution. In Figure 5 below, this solution is plotted, comparing the results of the Improved Euler Method iteration for various step sizes. As the step size grew to 0.5, the solution diverged, as noted by the grey line in the plot. For smaller step sizes of 0.1 and 0.05, as denoted by the orange and blue lines, the solution converges to 1. The shape of the profile is consistent with would be expected in laminar flow past a flat plate.

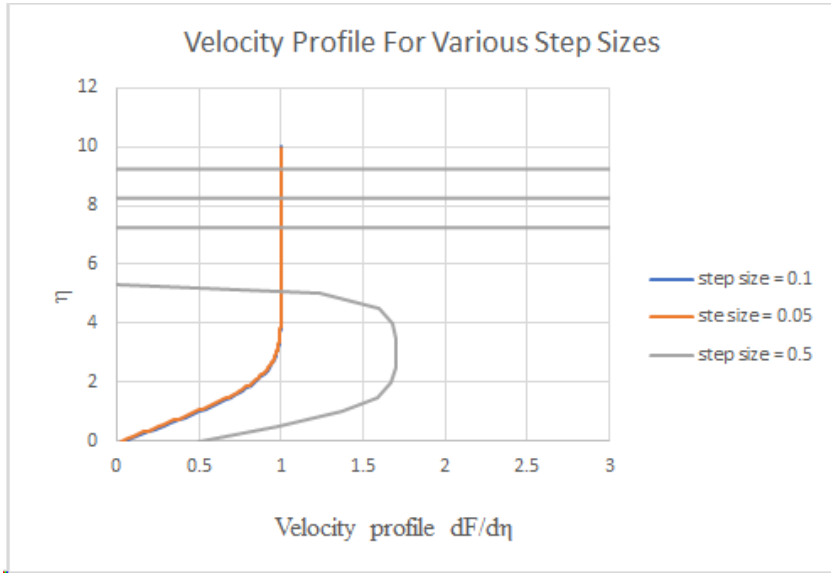

Figure 5. Comparison of step sizes on Blasius solution

The data produced by the Improved Euler Method is similar to the results given at the beginning of this project. As seen from the table in the Appendix, the final values of dF/dη for various values of Beta are all close to the value of 1. The data that is given initially showed dF/dη beginning to converge around η values of 4.8, whereas with the Improved Euler Method, it began to converge earlier. As seen in the data table, most dF/dη values begin to converge to 1 around 3.2 even with the same step size. Comparing the results of $d^2F/d\eta^2$ (0) in the table to the theoretical value of 1.0, the largest percent error is 0.1005% and the lowest is 0.008%. The average percent error was .07623% which is well within reason considering the step size used was 0.1.

The plot below represents the data obtained for the variable F(η), which is the velocity function. For all values of beta, it continues to grow with increasing η values. This is because the velocity, initially at zero, will accelerate along a favorable pressure gradient. This condition is met considering there is an angled wedge instead flat plate for beta values greater than zero which mimics the leading edge of an airfoil.

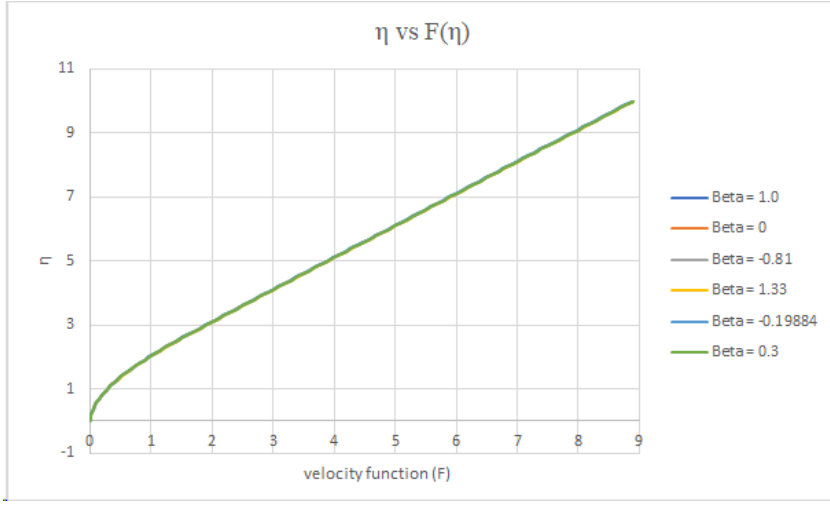

Figure 6. Plot of F(η)

DISCUSSION

The success of numerical integration methods depends on which method is used and the precision of parameters involved. Obtaining convergence for small values of beta such as 0 or -0.180 required little effort, but experimentation was required for convergence at values greater than 1. Step sizes of 1.0, 0.5, 0.1, 0.05 were all used to see the effects they would have on converge. It was noted that as the step sizes decreased, the iterations converge to one more successfully. The initial condition parameters for $d^2F/d\eta^2$ were also important for convergence. As beta grew beyond 1, the range from A to B needed to be expanded. Originally A was set to 0 and B was set to 1. considering the equation:

$$d^3G/d\eta^3 + G*d^2G/d\eta^2 + \beta[1-(dG/d\eta)^2]=0$$

when converted to state space form,

$$d^3G/d\eta^3 = - G*d^2G/d\eta^2 - \beta[1-(dG/d\eta)^2]$$

it can be seen that as beta values increase, the initial condition for $d^3G/d\eta^3$ must also increase beyond 1. This means that the upper limit of A and B needed to be increased beyond 1. The range instead became 0-2. The improved Euler Method is a useful method in numerical integration. It produces a result to a higher degree of accuracy than the Euler method as it takes the solution of the Euler method, then averages with a corrector that estimates the derivative at the end point of the step interval.

APPENDIX

η	Beta = -0.1988	Beta = -0.18	Beta=0	Beta = 0.3	Beta=1	Beta=1.33
0	0.046795	0.0468961	0.046973	0.0470352	0.047246	0.0475492
0.1	0.0935846	0.0937868	0.09394	0.0940642	0.094483	0.0950866
0.2	0.140336	0.140639	0.140868	0.141053	0.141673	0.142572
0.3	0.186995	0.187399	0.187703	0.187947	0.188758	0.189945
0.4	0.233486	0.233989	0.234368	0.234668	0.235657	0.23712
0.5	0.279712	0.280314	0.280765	0.281118	0.282269	0.283993
0.6	0.325556	0.326255	0.326777	0.327178	0.328471	0.33044
0.7	0.370885	0.371678	0.372269	0.372713	0.374124	0.376317
0.8	0.415548	0.416433	0.417088	0.417571	0.419075	0.421468
0.9	0.459381	0.460355	0.461072	0.461587	0.463155	0.465723
1	0.502213	0.503271	0.504045	0.504586	0.506188	0.508902
1.1	0.543865	0.545002	0.545829	0.546388	0.547994	0.550824
1.2	0.584158	0.585369	0.586243	0.586813	0.588392	0.591307
1.3	0.622917	0.624195	0.625111	0.625685	0.627205	0.630173
1.4	0.659974	0.661313	0.662266	0.662835	0.664267	0.667258
1.5	0.695178	0.696571	0.697554	0.698113	0.699429	0.702412
1.6	0.728394	0.729835	0.730842	0.731384	0.732559	0.735505
1.7	0.759513	0.760993	0.762019	0.762538	0.763551	0.766435
1.8	0.788451	0.789963	0.791001	0.791493	0.792325	0.795124
1.9	0.815152	0.81669	0.817735	0.818194	0.818813	0.821527
2	0.839593	0.84115	0.842197	0.84262	0.843057	0.84563
2.1	0.86178	0.86335	0.864395	0.86478	0.86501	0.86745
2.2	0.881751	0.883328	0.884367	0.884713	0.884735	0.887036
2.3	0.899572	0.901151	0.902181	0.902487	0.902305	0.904462
2.4	0.915333	0.91691	0.917929	0.918195	0.917816	0.919828
2.5	0.929148	0.930719	0.931725	0.931953	0.931386	0.933257
2.6	0.941145	0.942708	0.9437	0.943891	0.94315	0.944884
2.7	0.951468	0.953022	0.953998	0.954155	0.953252	0.954858
2.8	0.960268	0.96181	0.962772	0.962897	0.961848	0.963334
2.9	0.967698	0.969229	0.970175	0.970272	0.969092	0.970469
3	0.973913	0.975432	0.976364	0.976435	0.97514	0.976418
3.1	0.979061	0.980568	0.981488	0.981535	0.98014	0.981332
3.2	0.983286	0.984782	0.985689	0.985717	0.984235	0.985351
3.3	0.986719	0.988205	0.989101	0.989112	0.987557	0.988607
3.4	0.989482	0.990959	0.991846	0.991842	0.990225	0.99122
3.5	0.991685	0.993153	0.994032	0.994016	0.992349	0.993296
3.6	0.993424	0.994885	0.995757	0.995731	0.994022	0.994931
3.7	0.994785	0.996239	0.997105	0.997071	0.995328	0.996205
3.8	0.995838	0.997287	0.998148	0.998107	0.996337	0.997189
3.9	0.996646	0.998091	0.998948	0.998902	0.99711	0.997942
4	0.997261	0.998702	0.999555	0.999505	0.997696	0.998512
4.1	0.997723	0.999161	1.00001	0.999958	0.998137	0.99894
4.2	0.998068	0.999503	1.00035	1.0003	0.998464	0.999258
4.3	0.998322	0.999756	1.0006	1.00054	0.998706	0.999492
4.4	0.998509	0.999941	1.00079	1.00073	0.998882	0.999663
4.5	0.998644	1.00007	1.00092	1.00086	0.99901	0.999786
4.6	0.998741	1.00017	1.00101	1.00095	0.999101	0.999875
4.7	0.99881	1.00024	1.00108	1.00102	0.999166	0.999938
4.8	0.998858	1.00029	1.00113	1.00107	0.999212	0.999982
4.9	0.998892	1.00032	1.00116	1.0011	0.999243	1.00001
5	0.998916	1.00034	1.00119	1.00112	0.999265	1.00003

η	Beta = -0.1988	Beta = -0.18	Beta=0	Beta = 0.3	Beta=1	Beta=1.33
5.1	0.998932	1.00036	1.0012	1.00114	0.99928	1.00005
5.2	0.998943	1.00037	1.00121	1.00115	0.99929	1.00006
5.3	0.99895	1.00038	1.00122	1.00116	0.999297	1.00006
5.4	0.998955	1.00038	1.00122	1.00116	0.999301	1.00007
5.5	0.998959	1.00039	1.00123	1.00117	0.999304	1.00007
5.6	0.998961	1.00039	1.00123	1.00117	0.999306	1.00007
5.7	0.998962	1.00039	1.00123	1.00117	0.999307	1.00008
5.8	0.998963	1.00039	1.00123	1.00117	0.999307	1.00008
5.9	0.998964	1.00039	1.00123	1.00117	0.999308	1.00008
6	0.998964	1.00039	1.00123	1.00117	0.999308	1.00008
6.1	0.998964	1.00039	1.00123	1.00117	0.999308	1.00008
6.2	0.998964	1.00039	1.00123	1.00117	0.999308	1.00008
6.3	0.998965	1.00039	1.00123	1.00117	0.999308	1.00008
6.4	0.998965	1.00039	1.00123	1.00117	0.999307	1.00008
6.5	0.998965	1.00039	1.00123	1.00117	0.999307	1.00008
6.6	0.998965	1.00039	1.00123	1.00117	0.999307	1.00008
6.7	0.998965	1.00039	1.00123	1.00117	0.999307	1.00008
6.8	0.998965	1.00039	1.00123	1.00117	0.999307	1.00008
6.9	0.998965	1.00039	1.00123	1.00117	0.999306	1.00008
7	0.998965	1.00039	1.00123	1.00117	0.999306	1.00008
7.1	0.998965	1.00039	1.00123	1.00117	0.999306	1.00008
7.2	0.998965	1.00039	1.00123	1.00117	0.999306	1.00008
7.3	0.998965	1.00039	1.00123	1.00117	0.999306	1.00008
7.4	0.998965	1.00039	1.00123	1.00117	0.999305	1.00008
7.5	0.998965	1.00039	1.00123	1.00117	0.999305	1.00008
7.6	0.998965	1.00039	1.00123	1.00117	0.999305	1.00008
7.7	0.998965	1.00039	1.00123	1.00117	0.999305	1.00008
7.8	0.998965	1.00039	1.00123	1.00117	0.999305	1.00008
7.9	0.998965	1.00039	1.00123	1.00117	0.999304	1.00008
8	0.998965	1.00039	1.00123	1.00117	0.999304	1.00008
8.1	0.998965	1.00039	1.00123	1.00117	0.999304	1.00008
8.2	0.998965	1.00039	1.00123	1.00117	0.999304	1.00008
8.3	0.998965	1.00039	1.00123	1.00117	0.999304	1.00008
8.4	0.998965	1.00039	1.00123	1.00117	0.999304	1.00008
8.5	0.998965	1.00039	1.00123	1.00117	0.999303	1.00008
8.6	0.998965	1.00039	1.00123	1.00117	0.999303	1.00008
8.7	0.998965	1.00039	1.00123	1.00117	0.999303	1.00008
8.8	0.998965	1.00039	1.00123	1.00117	0.999303	1.00008
8.9	0.998965	1.00039	1.00123	1.00117	0.999303	1.00008
9	0.998965	1.00039	1.00123	1.00117	0.999302	1.00008
9.1	0.998965	1.00039	1.00123	1.00117	0.999302	1.00008
9.2	0.998965	1.00039	1.00123	1.00117	0.999302	1.00008
9.3	0.998965	1.00039	1.00123	1.00117	0.999302	1.00008
9.4	0.998965	1.00039	1.00123	1.00117	0.999302	1.00008
9.5	0.998965	1.00039	1.00123	1.00117	0.999301	1.00008
9.6	0.998965	1.00039	1.00123	1.00117	0.999301	1.00008
9.7	0.998965	1.00039	1.00123	1.00117	0.999301	1.00008
9.8	0.998965	1.00039	1.00123	1.00117	0.999301	1.00008
9.9	0.998965	1.00039	1.00123	1.00117	0.999301	1.00008
10	0.998965	1.00039	1.00123	1.00117	0.9993	1.00008

```
/*      AERSP 312 Project 1
        March 14, 2019
```

```
	Submitted by: Brody Clark, Brett Sheeran, Kartik Kamdar, Logan Morrow, Joey
Cioffi, Nick DeCandia, John Wychock
*/

#include <iostream>
#include <iomanip>
#include <cmath>
#include <fstream>
#include <string>
using namespace std;

void IEM(double h, double &y1, double &y2, double &y3, double beta);  //prototype for
Improved Euler Method function

void bisection(double y2max, double& A, double& B); //function for narrowing in on the
proper value of F''(0)

int main()
{
	char ans = 'Y';		//character for while loop continuation
	double y1, y2, y3, A, B;	//variable for storing initial conditions. These will
be changed during iterations
	double h;  //variable for step size
	double beta;
	string name = "output at beta = ";
	string base = ".txt"
	double y2max; //this will be set to the value of F' at eta max, and will be
compared to 1.

	ofstream output2;	//creates files for outputting estimated F' values (y2)




//outer do/while-loop is for continuously entering beta values and writing data
	do {


		cout << "Enter the value for beta: " << endl;
		cin >> beta;  //sets new value of beta
		A = 0; //A and B contribute to the initial condition for y3
		B = 2;

		for (int j = 0; j <= 10; j++)	//loops through the Improved euler method
& y3 update 20 times
		{
```

```
                    //opens files for outputting F' values
                    output2.open(name + to_string(beta) + base);


      //initializes Y'' based on A and B, y1=0, y2=0 and makes step size .1

                y3 = (A + B) / 2;
                y1 = 0;
                y2 = 0;
                h = .1;


                for (double eta = 0; eta <= 10; eta += h)    //loop for reiterating IEM
function to produce new values each step. 100 total iterations
                    {

                  IEM(h, y1, y2, y3, beta);           //Improved Euler Method function
call, updates y1, y2, and y3 based on step size h and Beta


                     output2 << y1 << endl; //outputs estimated answers for Y1(F) to a
file


                      y2max = y2;   //sets y2max value based on final F' at max eta
value. In this case, the max eta value is 10
                    }

                    bisection(y2max, A, B);//bisection method function call, updates A
and B

                          //closes file
                    output2.close();



             }
             cout << "do you wish to perform another integration with a different Beta
value? Y/N" << endl;        //allows user to enter different beta values without
restarting entire code
             cin >> ans;
      } while (ans == 'Y' || ans == 'y');

      return 0;
}


void bisection(double check, double& A, double& B)  //bisection method for narrowing in
on proper y3 value, changes values of A and B based on y2 value (check)
{
```

```
	if (check > 1)	//compares F' at eta max to 1. if greater, lowers value of B
	{
		B = (A + B) / 2.0;
	}
	else if (check < 1)  //if greater, increases A
	{
		A = (A + B) / 2.0;
	}

}

void IEM(double h, double &y1, double &y2, double &y3, double beta)		//improved Euler method function. accepts step size and three reference variables as parameters.
{

	double y11, y22, y33;			//new values of Y1 and Y2 and Y3
	double dy = y2;				// equation for y1'
	double dy2 = y3;				//y2'
	double dy3 = -y1 * y3 - beta * (1 - pow(y2, 2));	//equation for F'''

	double k1[3] = { dy, dy2 , dy3 }; //K1 array, has element K11 and K12 and K13

	double y1star = y1 + h * k1[0];	//increased value of y1 based on step size
	double y2star = y2 + h * k1[1];	//increased value of  Y2 based on step size
	double y3star = y3 + h * k1[2];	//increased value of Y3 based on step size
	double k2[3] = { y2star, y3star, -y1star * y3star + beta * ( 1-pow(y2star,2)) };
//this becomes the increased values of of the evaluation point based on step size


	y11 = y1 + (h / 2)*(k1[0] + k2[0]);		//solves for new values of Y1 Y2 and Y3
	y22 = y2 + (h / 2)*(k1[1] + k2[1]);
	y33 = y3 + (h / 2)*(k1[2] + k2[2]);

	y1 = y11;	//changes value of y1, y2, and y3 for next iteration to use.
	y2 = y22;
	y3 = y33;


}
```