International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(REVIEW ARTICLE)

Check for updates

Numerical Solution for Plane Stagnation Point Flow

Stanley A. Omenai *

[1] *Department of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA.*

International Journal of Science and Research Archive, 2024, 12(02), 1088–1098

Publication history: Received on 15 June 2024; revised on 20 July 2024; accepted on 23 July 2024

Article DOI: https://doi.org/10.30574/ijsra.2024.12.2.1357

Abstract

The plane stagnation point flow, where a fluid stream impinges perpendicularly on a flat surface, is a classic problem in fluid dynamics with significant theoretical and practical implications. This report presents a comprehensive numerical solution to the plane stagnation point flow using the fourth order Runge-Kutta approximation. The numerical approach is developed to solve the governing Hiemenz Flow equation.

Key flow characteristics, including velocity, are analyzed, offering insights into the fluid behavior near the stagnation point.

Keywords: Similarity; Plane Stagnation; Hiemenz Flow; Runge-Kutta

1. Introduction

The study of fluid flow behavior near stagnation points is of paramount importance in both theoretical and applied fluid dynamics. A plane stagnation point flow, characterized by the deceleration of a fluid as it impinges perpendicularly on a surface, is a fundamental problem with wide-ranging applications in aerodynamics, thermal management, and various engineering processes. Understanding the flow characteristics in this region is crucial for optimizing performance and ensuring stability in systems where such flow conditions are encountered.

This study aims to develop and validate a robust numerical solution for the plane stagnation point flow using the fourth order Runge-Kutta approximation. The results obtained will not only enhance our theoretical understanding but also offer practical insights for applications in engineering design and optimization.

The following sections will detail the mathematical formulation of the problem and the numerical methods employed. Through this work, we aim to contribute to the ongoing advancement of numerical fluid dynamics and its application to stagnation point flow phenomena.

2. Problem Description

The similarity solution for Hiemenz flow can be expressed by the non-dimensional stream function $F(\eta)$ satisfying the ordinary differential equation

$$F''' + FF'' + 1 - F'2 = 0$$

subject to the boundary conditions

* Corresponding author: Stanley A. Omenai

$$F(0) = 0 \; ; \; F'(0) = 0 \; ; \; F'(\infty) = 1$$

This is known as a two-point boundary value problem since the conditions are imposed at two different locations. However, for simplicity we replace [0,∞] by a finite interval of η = 4.8

The above ODE may be re-cast into a system of 3 coupled first order equations. Thus, defining the variables Y1 = F, Y2 = F' and Y3 = F'', we obtain

$$\frac{dY_1}{d\eta} = Y_2$$

$$\frac{dY_2}{d\eta} = Y_3$$

$$\frac{dY_3}{d\eta} = {Y_2}^2 - Y_1 Y_3 - 1$$

with the boundary conditions Y1(0) = 0, Y2(0) = 0 and Y2(∞) = 1.

Since Y3(0) is unknown we can guess a value for it and integrate the system forward from η = 0 by solving the ODEs posed as initial-value problems until the condition Y2(∞) = 1 is satisfied.

We will also develop a program that iteratively guesses the correct value of Y3(0) that satisfies the condition Y2(∞) = 1 within some prescribed tolerance level

3. Methodology

This problem shall be solved numerically using the fourth order Runge-Kutta approximation. We shall carry out numerical integration of the coupled first order ODEs with prescribed initial conditions.

MATLAB codes shall be developed to solve for F and obtain results for F and its derivatives in tabulated and graphical forms. Comment statements shall be provided to show the logic of the algorithm.

MATLAB code shall also be developed that iteratively guesses the correct value of Y3(0) that satisfies the condition Y2(∞) = 1 within some prescribed tolerance level. This shall be embedded within the main program.

A discussion of the physical implications of the results and implications for the velocity components shall be provided. A summary of results: tables and plots, shall also be presented as part of the report.

4. Numerical Solution

Given the third order ordinary differential equation for Hiemenz flow

$$F''' + FF'' + 1 - F'2 = 0$$

subject to the boundary conditions

$$F(0) = 0 \; ; \; F'(0) = 0 \; ; \; F'(\infty) = 1$$

We define new variables: Y1 = F, Y2 = F' and Y3 = F'', and obtain the following set of first order coupled ODEs

$$\frac{dY_1}{d\eta} = Y_2$$

$$\frac{dY_2}{d\eta} = Y_3$$

$$\frac{dY_3}{d\eta} = {Y_2}^2 - Y_1 Y_3 - 1$$

with the boundary conditions Y1(0) = 0, Y2(0) = 0 and Y2(4.8) = 1.

The boundary condition Y2(∞) is approximated as Y2(4.8) = 1 since it will be difficult to define infinity on a computer.

The numerical integration is done using the fourth order Runge-Kutta method with an interval size of h = 0.0005. The Runge-Kutta approximation is expressed as:

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + O(h_5)$$

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1)$$

$$k_3 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2)$$

$$k_4 = hf(x_n + h, y_n + k_3)$$

Using successive guesses, we obtain a value of Y3(0) = 1.232 that satisfies the boundary condition Y2(4.8) ≈ 1. The code is provided below.

```
clear;

clc;

% Define Parameters
a = 0; % lower limit of integration
b = 4.8; % upper limit of integration
h = 5E-4; % step size
n = (b-a)/h+1; % number of points

% Define Initial Conditions
Y1(1) = 0;
Y2(1) = 0;
Y3(1) = 1.232; % obtained by iteration

% Define the system of ODE function handle
eta(1) = 0;
f1 = @(Y2) Y2;
f2 = @(Y3) Y3;
f3 = @(Y2,Y3,Y1) Y2*Y2-Y1*Y3-1

% 4th Order Runge-Kutta Loop
for i = 1:n-1
 eta(i+1) = i*h;
 k1_1 = h*feval(f1,Y2(i));
 k2_1 = h*feval(f1,Y2(i)+k1_1/2);
 k3_1 = h*feval(f1,Y2(i)+k2_1/2);
 k4_1 = h*feval(f1,Y2(i)+k3_1);
 Y1(i+1) = Y1(i)+(1/6)*(k1_1+2*k2_1+2*k3_1+k4_1);
 k1_2 = h*feval(f2,Y3(i));
 k2_2 = h*feval(f2,Y3(i)+k1_2/2);
 k3_2 = h*feval(f2,Y3(i)+k2_2/2);
 k4_2 = h*feval(f2,Y3(i)+k3_2);
 Y2(i+1) = Y2(i)+(1/6)*(k1_2+2*k2_2+2*k3_2+k4_2);
 k1_3 = h*feval(f3,Y2(i),Y3(i),Y1(i));
 k2_3 = h*feval(f3,Y2(i)+k1_1/2,Y3(i)+k1_2/2,Y1(i)+k1_3/2);
 k3_3 = h*feval(f3,Y2(i)+k2_1/2,Y3(i)+k2_2/2,Y1(i)+k2_3/2);
 k4_3 = h*feval(f3,Y2(i)+k3_1,Y3(i)+k3_2,Y1(i)+k3_3);
 Y3(i+1) = Y3(i)+(1/6)*(k1_3+2*k2_3+2*k3_3+k4_3);
```

```
end

% Plot results
plot(eta,Y2)
title("Plot of \eta versus Y2")
grid on
xlabel('\eta')
ylabel('Y2')
```

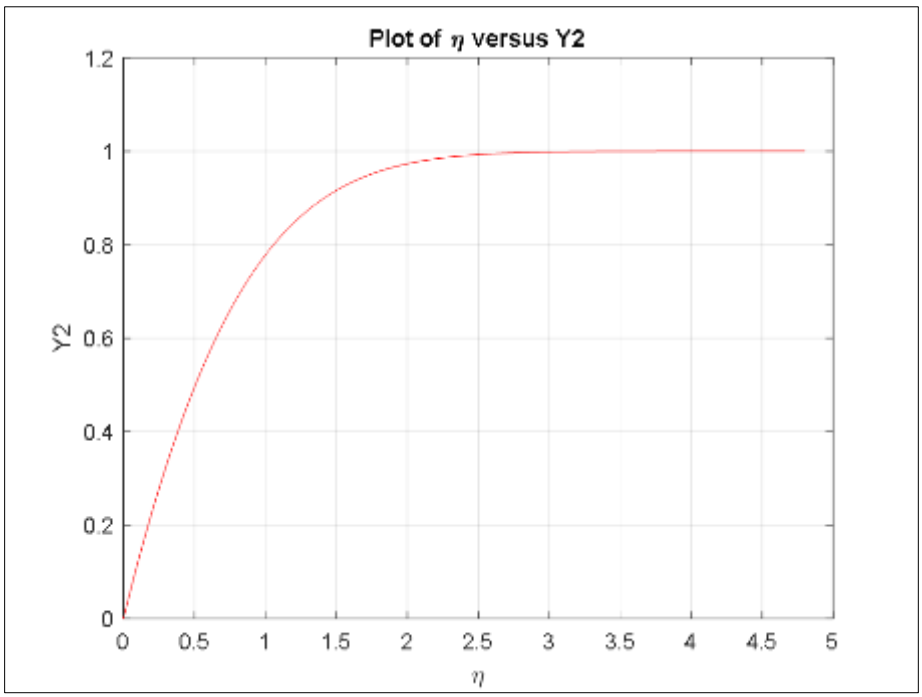

Figure 1 Plot of first derivative of non-dimensional stream function

Y2(4.8) = 1

Next, we develop MATLAB code that iteratively determines the correct value of Y3(0) that satisfies the condition Y2(4.8) = 1 within a tolerance level of 1 x 10-3.

The code proposes a new function as follows.

$$g = Y_2(\eta = 4.8)$$

The value of the function, g, depends on the initial value of Y3.

$$g = g[Y_3(0)]$$

This function approximately becomes 1 when the true value of Y3(0) is approached, thus, the following equation is satisfied.

$$g[Y_3(0)] - 1 = 0$$

The code then obtains corresponding values of g for values of Y3(0) within the interval [0:2] using steps of 0.001, which is the desired level of accuracy.

A plot of the function g[Y_3 (0)]-1 is made against Y_3 (0). The desired value of Y_3 (0) occurs where the curve crosses the x-axis. This value is Y_3 (0)=1.2320, which is the same as earlier obtained by iteration.

A breakdown of the code is provided below.

```
clear;

clc;

% Define Parameters
a = 0; % lower limit of integration
b = 4.8; % upper limit of integration
h = 5E-4; % step size
n = (b-a)/h+1; % number of points

% Define Initial Conditions
Y1(1) = 0;
Y2(1) = 0;
p = 1:1E-3:2;
r = length(p);
for j = 1:r
Y3(1) = p(j);

% Define the system of ODE function handle
eta(1) = 0;
f1 = @(Y2) Y2;
f2 = @(Y3) Y3;
f3 = @(Y2,Y3,Y1) Y2*Y2-Y1*Y3-1;

% 4th Order Runge-Kutta Loop
for i = 1:n-1
 eta(i+1) = i*h;
 k1_1 = h*feval(f1,Y2(i));
 k2_1 = h*feval(f1,Y2(i)+k1_1/2);
 k3_1 = h*feval(f1,Y2(i)+k2_1/2);
 k4_1 = h*feval(f1,Y2(i)+k3_1);
 Y1(i+1) = Y1(i)+(1/6)*(k1_1+2*k2_1+2*k3_1+k4_1);
 k1_2 = h*feval(f2,Y3(i));
 k2_2 = h*feval(f2,Y3(i)+k1_2/2);
 k3_2 = h*feval(f2,Y3(i)+k2_2/2);
 k4_2 = h*feval(f2,Y3(i)+k3_2);
 Y2(i+1) = Y2(i)+(1/6)*(k1_2+2*k2_2+2*k3_2+k4_2);
 k1_3 = h*feval(f3,Y2(i),Y3(i),Y1(i));
 k2_3 = h*feval(f3,Y2(i)+k1_1/2,Y3(i)+k1_2/2,Y1(i)+k1_3/2);
 k3_3 = h*feval(f3,Y2(i)+k2_1/2,Y3(i)+k2_2/2,Y1(i)+k2_3/2);
 k4_3 = h*feval(f3,Y2(i)+k3_1,Y3(i)+k3_2,Y1(i)+k3_3);
 Y3(i+1) = Y3(i)+(1/6)*(k1_3+2*k2_3+2*k3_3+k4_3);
end
 q(j) = Y2(end);
end
qi = 1;
Y3_0 = interp1(q,p,qi)

% Plot results
plot (p,q-1)
xlabel('Y3(\eta=0)')
ylabel('Y2(\eta=4.8) - 1')
axis([1 2 -10 50]);
grid on
grid minor
```

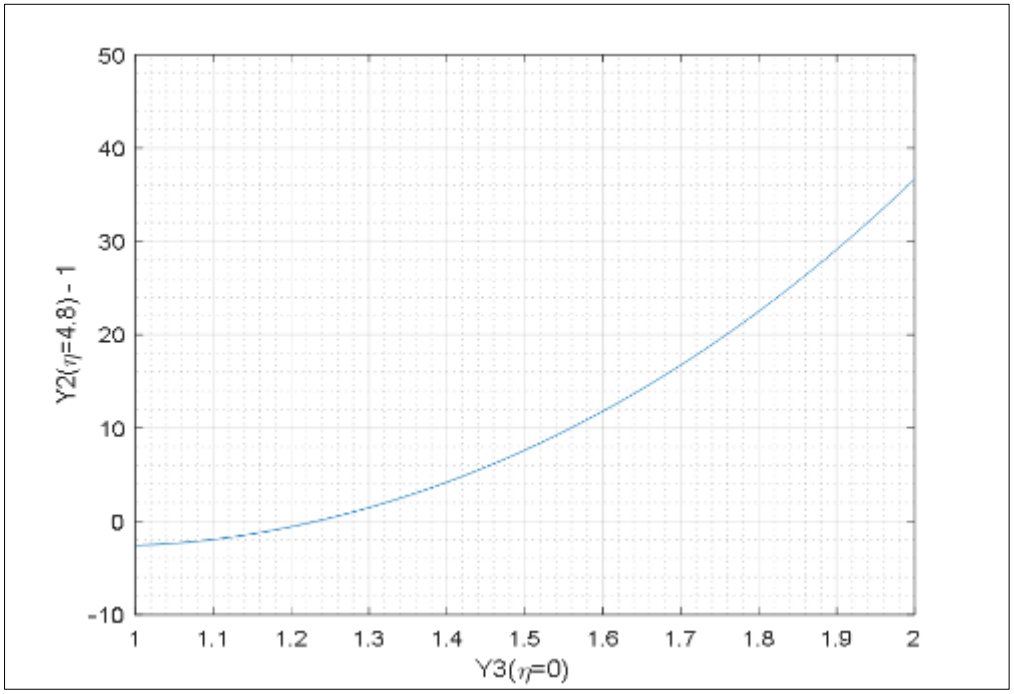

Figure 2 Plot of first derivative vs second derivative of stream function

Y3(0) = 1.2320

Finally, the subroutine that produces Y1, Y2 and Y3 is embedded within the iterative loop that solves for the correct value of Y3(0) that satisfies the boundary condition of Y2(4.8) = 1. The final code is shown below.

```
clear;

clc;

% Define Parameters
a = 0; % lower limit of integration
b = 4.8; % upper limit of integration
h = 5E-4; % step size
n = (b-a)/h+1; % number of points

% Define Initial Conditions
Y1(1) = 0;
Y2(1) = 0;

% Loop for boundary condition Y3(0)
p = 1:1E-3:2;
r = length(p);
for j = 1:r
Y(1) = p(j);
eta(1) = 0;
f1 = @(Y2) Y2;
f2 = @(Y) Y;
f3 = @(Y2,Y,Y1) Y2*Y2-Y1*Y-1;
for i = 1:n-1
 eta(i+1) = i*h;
 k1_1 = h*feval(f1,Y2(i));
 k2_1 = h*feval(f1,Y2(i)+k1_1/2);
 k3_1 = h*feval(f1,Y2(i)+k2_1/2);
 k4_1 = h*feval(f1,Y2(i)+k3_1);
 Y1(i+1) = Y1(i)+(1/6)*(k1_1+2*k2_1+2*k3_1+k4_1);
 k1_2 = h*feval(f2,Y(i));
 k2_2 = h*feval(f2,Y(i)+k1_2/2);
 k3_2 = h*feval(f2,Y(i)+k2_2/2);
 k4_2 = h*feval(f2,Y(i)+k3_2);
```

```
 Y2(i+1) = Y2(i)+(1/6)*(k1_2+2*k2_2+2*k3_2+k4_2);
 k1_3 = h*feval(f3,Y2(i),Y(i),Y1(i));
 k2_3 = h*feval(f3,Y2(i)+k1_1/2,Y(i)+k1_2/2,Y1(i)+k1_3/2);
 k3_3 = h*feval(f3,Y2(i)+k2_1/2,Y(i)+k2_2/2,Y1(i)+k2_3/2);
 k4_3 = h*feval(f3,Y2(i)+k3_1,Y(i)+k3_2,Y1(i)+k3_3);
 Y(i+1) = Y(i)+(1/6)*(k1_3+2*k2_3+2*k3_3+k4_3);
end
 q(j) = Y2(end);
end
qi = 1;
Y3(1) = interp1(q,p,qi);

% Define the system of ODE function handle
eta(1) = 0;
f1 = @(Y2) Y2;
f2 = @(Y3) Y3;
f3 = @(Y2,Y3,Y1) Y2*Y2-Y1*Y3-1;

% 4th Order Runge-Kutta Loop
for i = 1:n-1
 eta(i+1) = i*h;
 k1_1 = h*feval(f1,Y2(i));
 k2_1 = h*feval(f1,Y2(i)+k1_1/2);
 k3_1 = h*feval(f1,Y2(i)+k2_1/2);
 k4_1 = h*feval(f1,Y2(i)+k3_1);
 Y1(i+1) = Y1(i)+(1/6)*(k1_1+2*k2_1+2*k3_1+k4_1);
 k1_2 = h*feval(f2,Y3(i));
 k2_2 = h*feval(f2,Y3(i)+k1_2/2);
 k3_2 = h*feval(f2,Y3(i)+k2_2/2);
 k4_2 = h*feval(f2,Y3(i)+k3_2);
 Y2(i+1) = Y2(i)+(1/6)*(k1_2+2*k2_2+2*k3_2+k4_2);
 k1_3 = h*feval(f3,Y2(i),Y3(i),Y1(i));
 k2_3 = h*feval(f3,Y2(i)+k1_1/2,Y3(i)+k1_2/2,Y1(i)+k1_3/2);
 k3_3 = h*feval(f3,Y2(i)+k2_1/2,Y3(i)+k2_2/2,Y1(i)+k2_3/2);
 k4_3 = h*feval(f3,Y2(i)+k3_1,Y3(i)+k3_2,Y1(i)+k3_3);
 Y3(i+1) = Y3(i)+(1/6)*(k1_3+2*k2_3+2*k3_3+k4_3);
end

% Plot results
plot(eta,Y1,eta,Y2,eta,Y3)
title("Plot of \eta versus Y1, Y2 and Y3")
grid on
xlabel('\eta')
ylabel('Y1,Y2,Y3')
legend('Y1','Y2','Y3')
plot(eta,Y1,'color', 'b')
title("Plot of \eta versus Y1")
grid on
xlabel('\eta')
ylabel('Y1')
plot(eta,Y2,'color', 'r')
title("Plot of \eta versus Y2")
grid on
xlabel('\eta')
ylabel('Y2')
plot(eta,Y3,'color', 'y')
title("Plot of \eta versus Y3")
grid on
xlabel('\eta')
ylabel('Y3')
```

5. Results and Discussions

Using the numerical solution developed, we present a tabulation of 100 values of the results for F and its derivatives as a function of η.

Table 1 Table of Results for non-dimensional Stream Function

η	F	F'	F''		η	F	F'	F''		η	F	F'	F''
0	0	0	1.232		1.63	1.01	0.937	0.139		3.26	2.617	0.999	0.002
0.05	0.001	0.058	1.184		1.68	1.055	0.943	0.126		3.31	2.665	0.999	0.002
0.1	0.006	0.114	1.136		1.73	1.1	0.949	0.115		3.36	2.712	1	0.002
0.14	0.012	0.167	1.089		1.78	1.146	0.954	0.105		3.41	2.76	1	0.002
0.19	0.021	0.218	1.042		1.82	1.192	0.959	0.095		3.46	2.808	1	0.001
0.24	0.033	0.267	0.995		1.87	1.238	0.963	0.086		3.5	2.856	1	0.001
0.29	0.047	0.314	0.949		1.92	1.284	0.967	0.078		3.55	2.904	1	0.001
0.34	0.063	0.358	0.904		1.97	1.331	0.971	0.07		3.6	2.953	1	0.001
0.38	0.081	0.401	0.86		2.02	1.378	0.974	0.064		3.65	3.001	1	0.001
0.43	0.102	0.441	0.817		2.06	1.424	0.977	0.057		3.7	3.049	1	0.001
0.48	0.124	0.479	0.775		2.11	1.471	0.98	0.051		3.74	3.097	1	0.001
0.53	0.148	0.516	0.734		2.16	1.518	0.982	0.046		3.79	3.145	1	0.001
0.58	0.173	0.55	0.694		2.21	1.566	0.984	0.041		3.84	3.193	1	0.001
0.62	0.2	0.582	0.656		2.26	1.613	0.986	0.037		3.89	3.241	1	0
0.67	0.229	0.613	0.618		2.3	1.66	0.988	0.033		3.94	3.289	1	0
0.72	0.259	0.642	0.582		2.35	1.708	0.989	0.029		3.98	3.337	1	0
0.77	0.291	0.669	0.547		2.4	1.755	0.99	0.026		4.03	3.385	1	0
0.82	0.323	0.694	0.514		2.45	1.803	0.992	0.023		4.08	3.433	1	0
0.86	0.357	0.718	0.482		2.5	1.851	0.993	0.021		4.13	3.481	1	0
0.91	0.392	0.74	0.451		2.54	1.898	0.994	0.018		4.18	3.529	1	0
0.96	0.428	0.761	0.421		2.59	1.946	0.994	0.016		4.22	3.577	1	0
1.01	0.465	0.781	0.393		2.64	1.994	0.995	0.014		4.27	3.625	1	0
1.06	0.503	0.799	0.366		2.69	2.041	0.996	0.012		4.32	3.673	1	0
1.1	0.542	0.816	0.341		2.74	2.089	0.996	0.011		4.37	3.721	1	0
1.15	0.582	0.832	0.316		2.78	2.137	0.997	0.01		4.42	3.769	1	0
1.2	0.622	0.847	0.294		2.83	2.185	0.997	0.008		4.46	3.817	1	0
1.25	0.663	0.86	0.272		2.88	2.233	0.998	0.007		4.51	3.865	1	0
1.3	0.705	0.873	0.251		2.93	2.281	0.998	0.006		4.56	3.913	1	0
1.34	0.747	0.884	0.232		2.98	2.329	0.998	0.006		4.61	3.961	1	0
1.39	0.789	0.895	0.214		3.02	2.377	0.999	0.005		4.66	4.009	1	0
1.44	0.833	0.905	0.197		3.07	2.425	0.999	0.004		4.7	4.057	1	0
1.49	0.876	0.914	0.181		3.12	2.473	0.999	0.004		4.75	4.105	1	0
1.54	0.92	0.922	0.166		3.17	2.521	0.999	0.003		4.8	4.153	1	0
1.58	0.965	0.93	0.152		3.22	2.569	0.999	0.003					

Plots of the results for F and its derivatives as functions of η are also provided below.

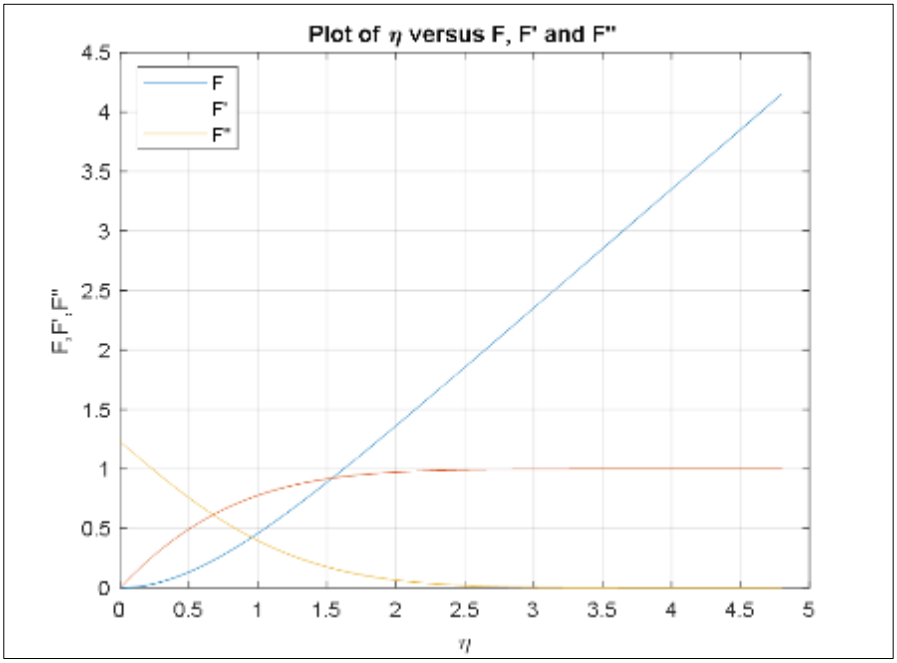

Figure 3 Plot of Stream Function together with its First and Second Derivatives

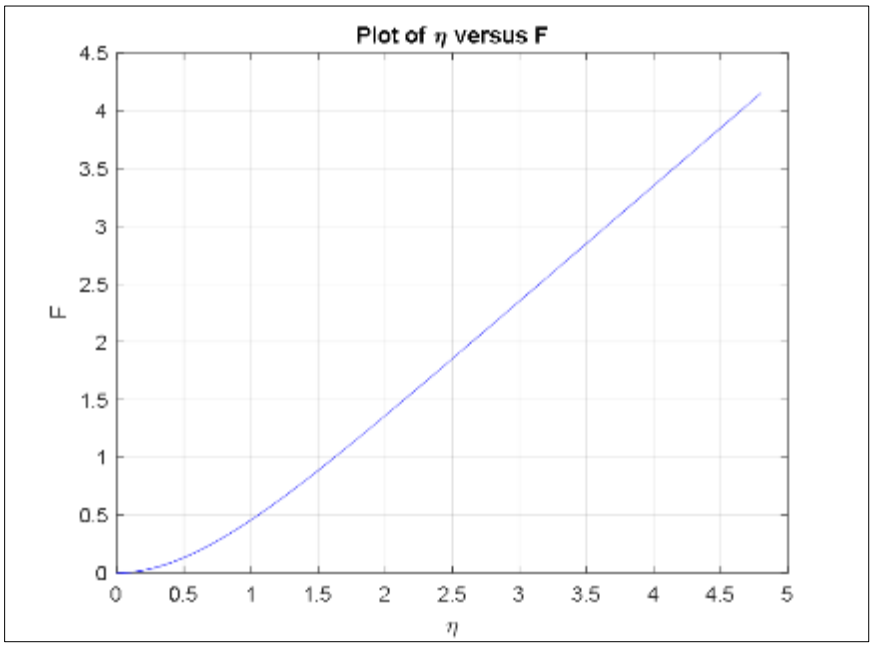

Figure 4 Plot of Stream Function only

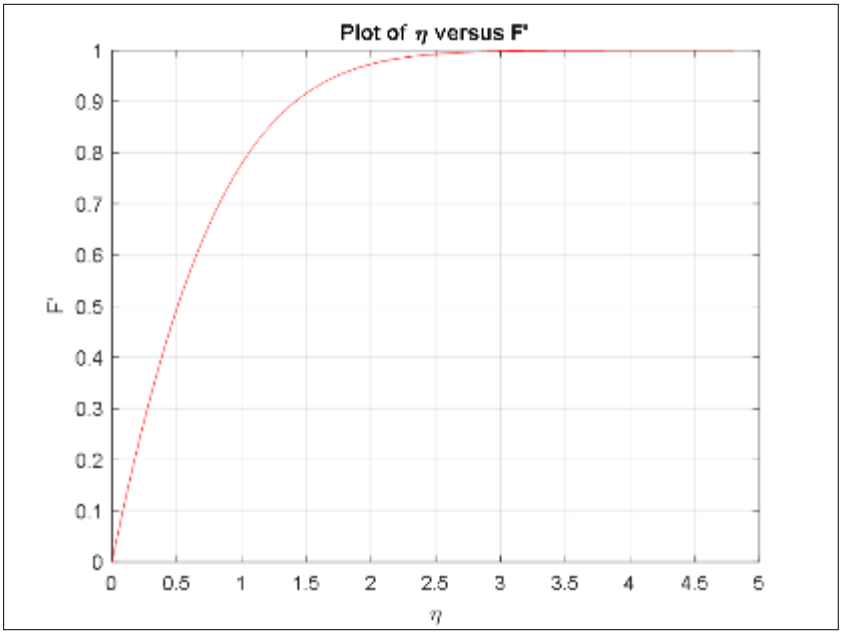

Figure 5 Plot of first derivative of Stream Function only

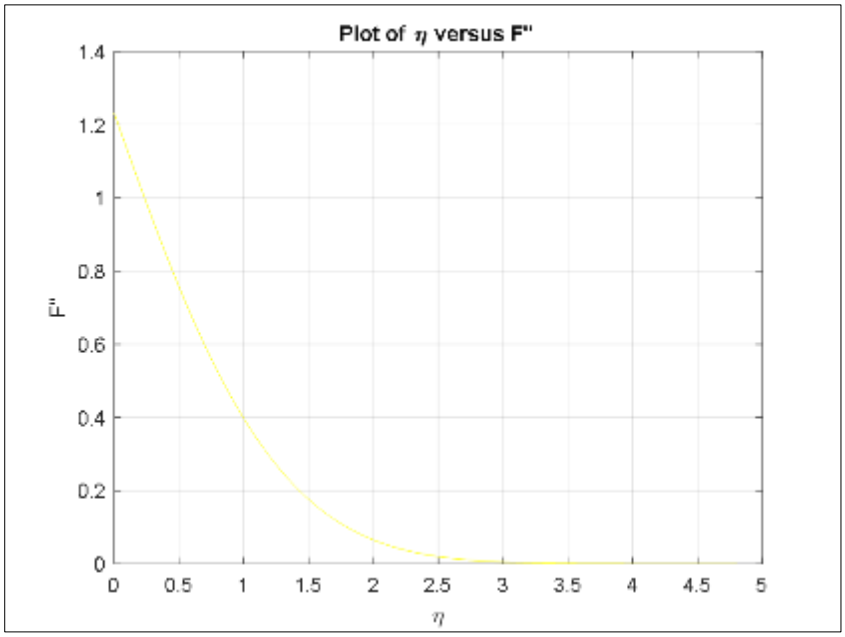

Figure 6 Plot of second derivative of Stream Function only

From the foregoing analysis, we can make the following inferences

- Near the wall, F increases at a very slow rate with η. This rate increases parabolically until it becomes linear at larger values of η.
- The first derivative of F initially increases linearly with η at lower values of η. The rate then decreases parabolically until F' eventually approaches a constant value of 1 as η increases.
- The second derivative of F becomes zero for large values of η.
- For plane stagnation flow, the similarity variable, η is defined as

$$\eta = y\sqrt{\frac{k}{\nu}}$$

- This shows that the similarity variable does not depend on the streamwise direction
- Also, the velocity components are defined as

$$u = kxF'$$

$$v = -\sqrt{\nu k}F$$

- Where ν is the kinematic viscosity and k is an arbitrary constant representing strain rate
- It is obvious from the above definition that the v-component of velocity depends only on the function F, and thus, increases with η at a slower (parabolic) rate for small values of η i.e., near the wall. This rate increases slightly and becomes linear for larger values of η.
- The u-component of velocity depends on x and F'. This means that for large values of η, u will depend only on the streamwise direction, x since F' approaches unity.

6. Conclusion

In this study, we have successfully developed a numerical solution for the plane stagnation point flow using the fourth order Runge-Kutta approximation. The solution required numerical integration of the coupled first order ODEs with prescribed initial conditions. Our findings demonstrate that the proposed numerical approach effectively captures the key characteristics of the stagnation point flow, including the velocity distributions.

Overall, the numerical solution presented in this study offers a valuable tool for understanding and predicting the complex behavior of plane stagnation point flows. Future work can extend this approach to more complex geometry and boundary conditions, further enhancing its applicability in engineering and scientific research.

References

[1] Abramowitz, M., Stegun, I., Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. Applied Mathematics Series [June 1964], United States Department of Commerce, National Bureau of Standards, Dover Publications.

[2] Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, Brian P. Numerical Recipes: The Art of Scientific Computing Third Edition (2007), Cambridge University Press

[3] Frank M. White. Viscous Fluid Flow, 3rd Edition. McGraw Hill Company.

[4] Hermann Schlichting, Klaus Gersten. Boundary Layer Theory, 9th Edition. Springer.