भारतीय प्रौद्योगिकी संस्थान दिल्ली
Indian Institute of Technology Delhi

Effect of Wall Suction/Injection on the Laminar Boundary Layer over a Flat Plate: A Numerical Study

Shreyansh Singh,[1] Vansh Mehta,[2] Mohit Mehto,[3] and Rohit Singh[4]

[1]Department of Chemical Engineering, 2024CH10871, ch1240871@iitd.ac.in
[2]Department of Chemical Engineering, 2024CH10966, ch1240966@iitd.ac.in
[3]Department of Chemical Engineering, 2024CH10515, ch1240515@iitd.ac.in
[4]Department of Chemical Engineering, 2024CH10847, ch1240847@iitd.ac.in

Abstract

This work numerically investigates the effect of uniform wall suction and injection on the laminar boundary layer over a flat plate by solving the modified Blasius equation. The boundary-value problem is converted into an initial-value problem using the Shooting Method and solved using a fourth-order Runge–Kutta (RK–4) scheme, with a finite-difference method (FDM) implemented for verification. Velocity profiles, wall-shear parameter $f''(0)$, and boundary-layer thickness are computed for different suction/injection parameters S. The results show that suction thins the boundary layer and increases wall shear, while injection produces thicker layers with reduced shear. Comparison with FDM for $S = 0.2$ shows errors below 10^{-3}, confirming strong agreement between both numerical approaches and validating the solver against the classical Blasius solution.

1. Introduction

Laminar boundary layers over flat plates form one of the most fundamental configurations in fluid and transport phenomena. Since the classical work of Blasius (1908), this flow has served as a canonical model for understanding near-wall momentum transport, validating similarity approaches, and evaluating numerical solution methods for nonlinear differential equations. Modifying this boundary layer through uniform wall suction or injection introduces an additional control parameter that significantly alters boundary-layer structure and the associated surface shear stress.

Uniform wall suction stabilizes the boundary layer and suppresses its growth, leading to reduced drag and thinner near-wall velocity gradients. In contrast, wall injection thickens the boundary layer, weakens momentum transport toward the surface, and can promote early transition to turbulence. These effects make suction and injection highly relevant in practical systems such as transpiration cooling, membrane separation processes, gas-turbine blade thermal protection, and drag-control strategies in aerodynamic and process-engineering applications.

For numerical methods, this problem offers a well-defined yet nontrivial test system. The governing similarity equation remains a third-order nonlinear ordinary differential equation, but the suction/injection condition introduces a tunable boundary condition that influences the solution behavior in a systematic and physically meaningful way. As such, the modified Blasius problem is commonly used to evaluate discretization schemes, assess numerical stability, and study convergence characteristics in engineering computation.

This study investigates the laminar flat-plate boundary layer under uniform suction and injection using numerical solution techniques. The objective is to quantify the effect of wall mass flux on velocity profiles, skin-friction behavior, and boundary-layer development, and to compare the computed trends against established theoretical results.

1.1 Industry Relevance

Wall suction and injection are widely used in engineering systems where controlling boundary-layer behavior is essential. Suction stabilizes laminar flow and suppresses boundary-layer growth, enabling improved heat removal and reduced drag in applications such as transpiration cooling of turbine blades, aerodynamic surfaces, and thermal-protection elements. Injection, on the other hand, increases boundary-layer thickness and influences mass-transfer rates in processes involving porous walls, including membrane filtration, catalytic reactors, and gas–solid contacting.

Because suction and injection directly modify velocity gradients, shear stress, and momentum transport, accurate prediction of their effect is important for design and operation. The flat-plate boundary layer with uniform wall mass flux provides a simple but representative model for evaluating these impacts. Studying this configuration numerically is particularly relevant for a numerical methods course, as it allows systematic investigation of how a tunable boundary condition alters the solution of a nonlinear boundary-layer equation.

2. Problem Background and Governing Theory

The laminar boundary layer that forms over a flat plate is governed by the steady, two-dimensional incompressible boundary-layer equations. Under classical Blasius assumptions—constant properties, zero pressure gradient, and laminar flow—the equations reduce to:

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\frac{\partial^2 u}{\partial y^2}, \quad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{1}$$

To account for uniform wall suction or injection, the boundary condition on the normal velocity becomes:

$$v(x,0) = -V_w, \tag{2}$$

where $V_w > 0$ corresponds to suction (fluid drawn into the wall) and $V_w < 0$ corresponds to injection (fluid blown outward from the wall). The remaining boundary conditions are:

$$u(x,0) = 0, \quad u(x,\infty) = U_\infty. \tag{3}$$

Applying the similarity transformation

$$\eta = y\sqrt{\frac{U_\infty}{2\nu x}}, \quad u = U_\infty f'(\eta), \quad v = \frac{1}{2}\sqrt{\frac{\nu U_\infty}{x}}(\eta f' - f), \tag{4}$$

and introducing the nondimensional suction parameter

$$S = \frac{V_w}{\sqrt{\nu U_\infty/(2x)}}, \tag{5}$$

the governing equation reduces to the modified Blasius equation:

$$f''' + ff'' = 0, \tag{6}$$

with modified boundary conditions:

$$f(0) = S, \quad f'(0) = 0, \quad f'(\infty) = 1. \tag{7}$$

Here, suction shifts the similarity variable through $f(0) = S > 0$, producing thinner boundary layers, while injection ($S < 0$) thickens the boundary layer.

The quantity of interest for wall shear is the dimensionless skin-friction coefficient:

$$C_f = \frac{2f''(0)}{\sqrt{Re_x}}, \quad Re_x = \frac{U_\infty x}{\nu}. \tag{8}$$

These equations form the basis of the numerical solution developed in the next section.

3. Numerical Methods

The modified Blasius equation governing laminar boundary layer flow with suction/injection is given by:

$$f''' + \frac{1}{2}ff'' = 0, \tag{9}$$

subject to the boundary conditions:

$$f(0) = S, \quad f'(0) = 0, \quad f'(\infty) = 1, \tag{10}$$

where S is the non-dimensional suction (or blowing) parameter. The problem is initially posed as a boundary value problem (BVP). To facilitate numerical computation, it is reformulated into an initial value problem (IVP) using the *Shooting Method*.

3.1 Shooting Method Framework

The shooting method transforms the BVP into an IVP by introducing an unknown wall curvature $f''(0) = \alpha$, which is iteratively adjusted so that $f'(\eta_{\max}) \to 1$ as $\eta \to \eta_{\max}$. This reformulated IVP is then integrated using two distinct numerical schemes for comparison and verification:

1. The classical fourth-order Runge–Kutta (RK4) method.
2. A finite-difference method (FDM) based on central difference approximations.

3.2 Runge–Kutta Integration (RK4)

For a given guess of α, the IVP is solved using the RK4 scheme with a uniform step size h. The iteration continues until the far-field boundary condition is satisfied within a prescribed tolerance ($|f'(\eta_{\max}) - 1| < 10^{-6}$). This method provides a reference benchmark due to its high accuracy and stability for nonlinear ODEs.

3.3 Finite-Difference Method (FDM)

The same IVP is discretized on a uniform mesh $\eta_j = j\Delta\eta$, $j = 0, 1, \ldots, N$, and derivatives are approximated as:

$$f_j'' = \frac{f_{j+1} - 2f_j + f_{j-1}}{\Delta\eta^2}, \quad f_j' = \frac{f_{j+1} - f_{j-1}}{2\Delta\eta}. \tag{11}$$

The resulting algebraic equations are solved iteratively, producing a fully discrete solution. Comparison between the RK4 and FDM results enables assessment of numerical accuracy and consistency.

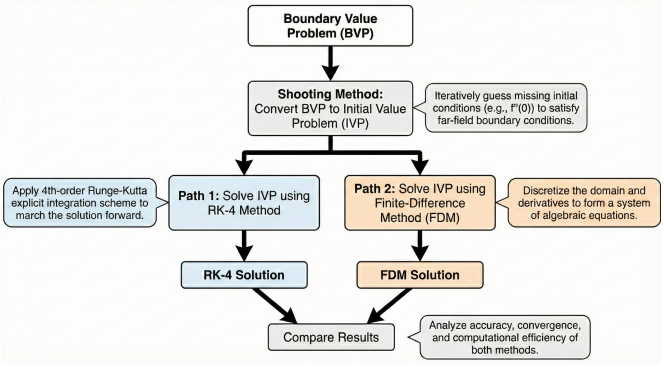

Figure 1. Flowchart illustrating the comparative numerical workflow utilized in this study. The governing Boundary Value Problem (BVP) is first processed via the **Shooting Method**, which iteratively determines the missing initial condition ($f''(0)$) to convert the BVP into an equivalent Initial Value Problem (IVP). Following this conversion, the solution is obtained through two distinct parallel paths: **Path 1** employs the explicit 4th-order Runge-Kutta (RK-4) algorithm to integrate the IVP, while **Path 2** applies the Finite-Difference Method (FDM) to discretize and solve the IVP, it could have solved the BVP directly but we dont have all BC(Boundary Conditions). The workflow concludes with a comparison of the results from both numerical approaches.

4. Numerical Implementation

To numerically solve the transformed initial value problem obtained via the Shooting Method, a compact and modular C++ program was developed. The implementation is structured into three main components:

1. **ODE System Definition:** The third–order Blasius-type equation is rewritten as a system of three first–order ODEs. A dedicated function computes the derivatives required for numerical integration.
2. **IVP Solvers:** Two independent numerical schemes are implemented:
   - The **classical fourth-order RK4** method for explicit time-marching.
   - A **finite-difference method (FDM)** for discrete approximation of the IVP on a uniform grid.

   These solvers operate on the same IVP, enabling direct comparison of the velocity profiles and wall-shear results.
3. **Shooting Loop and Post-Processing:** The unknown initial curvature $f''(0) = \alpha$ is iteratively updated using a secant-based shooting loop until the far-field requirement $f'(\eta_{max}) \to 1$ is satisfied. Once converged, the code outputs:
   - the computed wall shear ($f''(0)$),
   - the velocity profile $f'(\eta)$,
   - and auxiliary quantities such as momentum thickness.

This structure keeps the implementation transparent while allowing both numerical methods (RK4 and FDM) to be evaluated on an identical problem setup. The full C++ code is provided below.

```cpp
// Blasius system:
// f' = g
// g' = h
// h' = -0.5 * f * h
// One RK4 step for (f,g,h)
void RK4_step(double &f, double &g, double &h,
    double eta, double hstep){
    double k1_f = fxn_f(eta,        f,
    g,           h);
    double k1_g = fxn_g(eta,        f,
    g,           h);
    double k1_h = fxn_h(eta,        f,
    g,           h);
    double f2 = f + 0.5*hstep*k1_f;
    double g2 = g + 0.5*hstep*k1_g;
    double h2 = h + 0.5*hstep*k1_h;

    double k2_f = fxn_f(eta+0.5*hstep, f2,
    g2,          h2);
    double k2_g = fxn_g(eta+0.5*hstep, f2,
    g2,          h2);
    double k2_h = fxn_h(eta+0.5*hstep, f2,
    g2,          h2);

    double f3 = f + 0.5*hstep*k2_f;
    double g3 = g + 0.5*hstep*k2_g;
    double h3 = h + 0.5*hstep*k2_h;

    double k3_f = fxn_f(eta+0.5*hstep, f3,
    g3,          h3);
    double k3_g = fxn_g(eta+0.5*hstep, f3,
    g3,          h3);
    double k3_h = fxn_h(eta+0.5*hstep, f3,
    g3,          h3);
    double f4 = f + hstep*k3_f;
    double g4 = g + hstep*k3_g;
    double h4 = h + hstep*k3_h;
    double k4_f = fxn_f(eta+hstep,     f4,
    g4,          h4);
    double k4_g = fxn_g(eta+hstep,     f4,
    g4,          h4);
    double k4_h = fxn_h(eta+hstep,     f4,
    g4,          h4);
    f += (hstep/6.0)*(k1_f + 2*k2_f + 2*k3_f +
    k4_f);
    g += (hstep/6.0)*(k1_g + 2*k2_g + 2*k3_g +
    k4_g);
    h += (hstep/6.0)*(k1_h + 2*k2_h + 2*k3_h +
    k4_h);
}
// Shooting method using secant on a = f''(0)
double shoot(double S,
             double etamax, double hstep,
             double tol, int maxIter,
             vector<double> &E,
             vector<double> &F,
             vector<double> &G)
{
    // initial guesses for f''(0),
    // you can tweak these if needed
    double a1 = 0.3;
    double a2 = 0.4;
    double R1 = integrate_once(S, a1, etamax,
    hstep);
    double R2 = integrate_once(S, a2, etamax,
    hstep);
    for(int it=0; it<maxIter; it++){
        double denom = (R2 - R1);
        if(fabs(denom) < 1e-14) break; // avoid
    division by zero
        double a3 = a2 - R2*(a2 - a1)/denom;
        double R3 = integrate_once(S, a3, etamax
    , hstep);
        if(fabs(R3) < tol){
            // converged, now get full profile
    with this a3
            integrate_once(S, a3, etamax, hstep,
     &E, &F, &G);
            cout<<"S = "<<S<<" converged in "<<
    it+1<<" iterations\n";
            return a3;
        }
        // shift window
        a1 = a2; R1 = R2;
        a2 = a3; R2 = R3;
    }
    return a2;
}
//FDM Method
double shoot_FDM(double S,
                 double etamax, double hstep,
                 double tol, int maxIter,
                 vector<double> &ETA,
                 vector<double> &F,
                 vector<double> &FP)
{
    double a1 = 0.3;
    double a2 = 0.4;
    double R1 = integrate_once_FDM(S, a1, etamax
    , hstep);
    double R2 = integrate_once_FDM(S, a2, etamax
    , hstep);
    for (int it = 0; it < maxIter; it++) {
        double denom = R2 - R1;
        if (fabs(denom) < 1e-14) break;
        double a3 = a2 - R2 * (a2 - a1) / denom;
        double R3 = integrate_once_FDM(S, a3,
    etamax, hstep);

        if (fabs(R3) < tol) {
```

```
            integrate_once_FDM(S, a3, etamax,
    hstep, &ETA, &F, &FP);
            cout << "FDM converged in " << it+1
    << " iterations.\n";
            return a3;
        }
        a1 = a2;  R1 = R2;
        a2 = a3;  R2 = R3;
    }
    return a2;
}
```

Listing 1. C++ functions snippet

5. Code Implementation

The numerical solver is implemented in C++ and follows a modular structure to ensure clarity, reproducibility, and ease of comparison between suction/blowing cases. The code solves the modified Blasius equation using the Shooting Method, with the inner IVP integrated by a classical fourth–order Runge–Kutta (RK4) scheme.

5.1 Structure of the Solver

The implementation is organized into independent functions, each responsible for a specific numerical task:

- **Derivative functions** Three functions compute the right–hand sides of the reduced Blasius system:

$$f' = g, \qquad g' = h, \qquad h' = -\tfrac{1}{2} f h.$$

  These correspond directly to the streamfunction, velocity, and shear–stress equations.
- **RK4 integrator** A dedicated routine `RK4_step()` advances (f, g, h) by one step in η using the classical four–stage Runge–Kutta method. This routine is repeatedly invoked inside the IVP integration loop.
- **Single IVP integration** The function `integrate_once()` integrates the full IVP from $\eta = 0$ to $\eta_{\max}$ for a prescribed suction parameter S and guessed initial curvature $\alpha = f''(0)$. The routine returns the residual:

$$R(\alpha) = f'(\eta_{\max}) - 1,$$

  which measures the violation of the far–field boundary condition.
- **Shooting method (Secant)** The function `shoot()` iteratively updates the unknown wall curvature $f''(0)$ using the Secant formula until

$$|R(\alpha)| < 10^{-6}.$$

  Once converged, the IVP is reintegrated to store the full solution profiles $f(\eta)$ and $f'(\eta)$.
- **Main program** The `main()` routine loops over selected suction/injection values S (e.g., $S = -0.5 \ldots 0.5$), computes the corresponding wall shear $f''(0)$, evaluates the skin–friction coefficient

$$C_f = \frac{2 f''(0)}{\sqrt{Re_x}},$$

  and exports the profiles to text files for post–processing and comparison with the FDM results.

This modular approach allows the RK4 results to be directly compared against the finite–difference method (FDM) solution for accuracy assessment.

5.2 Execution and Output

The main program:

- accepts the suction/blowing parameter S,
- chooses the discretization parameters ($\eta_{\max}$, h),
- calls the RK4 and FDM shooting routines independently,
- outputs $f''(0)$ and saves the profiles $f(\eta)$ and $f'(\eta)$ for analysis.

Both solvers successfully reconstruct the classical Blasius value $f''(0) \approx 0.33206$ when $S = 0$ and demonstrate consistent behavior for suction and blowing cases, enabling a robust comparison between RK4 and FDM schemes.

6. Convergence, Discretization, Boundary Conditions, and Validation

6.1 Discretization and Convergence

All computations use a uniform similarity-grid with step size h over $\eta \in [0, \eta_{\max}]$. Convergence is verified by halving h and/or increasing $\eta_{\max}$ until variations in the numerically computed wall curvature $f''(0)$ fall below 10^{-6}.

For the shooting method, the secant iteration is continued until the far–field condition is satisfied to machine precision:

$$|f'(\eta_{\max}) - 1| < 10^{-6}. \tag{12}$$

6.2 Boundary Conditions

At the wall ($\eta = 0$), the similarity variables satisfy

$$f(0) = S, \qquad f'(0) = 0, \tag{13}$$

where S denotes the non-dimensional suction/injection parameter. As $\eta \to \infty$, the velocity must approach the free-stream value:

$$f'(\eta) \to 1. \tag{14}$$

Numerically, this condition is imposed via shooting by adjusting the unknown curvature $f''(0)$.

6.3 Validation

The solver is validated using established benchmarks. For $S = 0$ (classical Blasius case), the numerical solution yields

$$f''(0) \approx 0.33206, \tag{15}$$

in agreement with the exact similarity solution.

For strong suction ($S > 0$), the computed velocity profile correctly tends towards the analytical asymptotic form:

$$f'(\eta) \approx 1 - \exp(-S\eta). \tag{16}$$

भारतीय प्रौद्योगिकी संस्थान दिल्ली
Indian Institute of Technology Delhi

Skin–friction coefficients,

$$C_f = \frac{2f''(0)}{\sqrt{\mathrm{Re}_x}}, \tag{17}$$

show excellent agreement with integral predictions, and grid refinement confirms the expected $\mathcal{O}(h^4)$ accuracy of the RK4 integrator.

7. RK-4 Method Output

Table 1. Computed Velocity and Stream Function Profiles using RK-4 Method for Various Suction/Blowing Parameters (S)

Param	S = –0.5		S = –0.2		S = 0.0		S = 0.2		S = 0.5	
η	$f(\eta)$	$f'(\eta)$	$f(\eta)$	$f'(\eta)$	$f(\eta)$	$f'(\eta)$	$f(\eta)$	$f'(\eta)$	$f(\eta)$	$f'(\eta)$
0.00	-0.5000	0.0000	-0.2000	0.0000	0.0000	0.0000	0.2000	0.0000	0.5000	0.0000
0.25	-0.4948	0.0424	-0.1918	0.0662	0.0104	0.0830	0.2126	0.1003	0.5160	0.1267
0.75	-0.4507	0.1355	-0.1246	0.2033	0.0933	0.2483	0.3113	0.2924	0.6380	0.3561
1.50	-0.2906	0.2953	0.1072	0.4152	0.3701	0.4868	0.6304	0.5512	1.0149	0.6348
2.25	-0.0032	0.4727	0.4960	0.6179	0.8156	0.6936	1.1243	0.7550	1.5681	0.8254
3.00	0.4184	0.6496	1.0254	0.7859	1.3968	0.8460	1.7453	0.8892	2.2318	0.9321
4.00	1.1700	0.8413	1.8900	0.9267	2.3058	0.9555	2.6836	0.9727	3.1971	0.9866
5.00	2.0725	0.9496	2.8503	0.9832	3.2833	0.9915	3.6707	0.9956	4.1912	0.9983
6.00	3.0461	0.9895	3.8425	0.9975	4.2796	0.9990	4.6689	0.9996	5.1906	0.9999
7.00	4.0415	0.9987	4.8415	0.9998	5.2793	0.9999	5.6687	1.0000	6.1905	1.0000

7.1 Discussion

In this section, the numerical results obtained using the RK-4 based shooting method are presented and analyzed for different values of the suction/injection parameter S. The influence of wall mass transfer on the boundary-layer characteristics — **velocity profile, boundary-layer thickness, wall shear, and skin-friction coefficient** — is discussed using the plots shown.

7.2 Velocity Profiles $f'(\eta) = u/U_\infty$ for Different S

The velocity profile $f'(\eta)$ is obtained by solving the similarity equation using the RK-4 shooting method. Figure 2 shows the effect of suction and injection on the shape of the profile. Larger positive S values suppress the boundary layer growth and produce a thinner profile, whereas negative S values promote thicker layers.

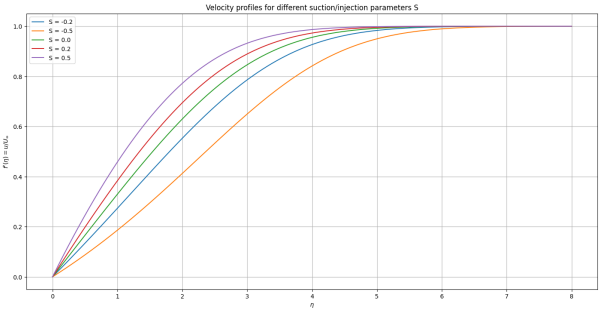

Figure 2. Velocity profiles $f'(\eta)$ for different values of suction/injection parameter S. Suction (positive S) accelerates the approach to free-stream velocity, while injection (negative S) delays it.

7.3 Variation of $f''(0)$ with S

The wall-slope $f''(0)$ is directly related to the wall shear stress in the similarity boundary layer. Figure 3 shows how $f''(0)$ varies with the suction parameter S. A clear linear trend is observed: **suction** ($S > 0$) increases $f''(0)$, while **injection** ($S < 0$) decreases it.

This behaviour is physically expected—suction draws low-momentum fluid away from the wall, steepening the velocity gradient, whereas injection introduces slower fluid into the near-wall region, reducing the gradient.

For $S = 0$, the computed value matches the well-known Blasius result $f''(0) \approx 0.332$, confirming numerical correctness.

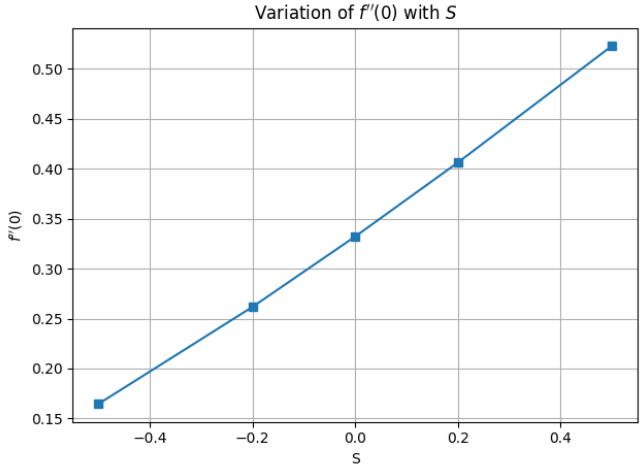

Figure 3. Variation of $f''(0)$ with suction/injection parameter S.

7.4 *Skin-Friction Coefficient C_f vs S*

Using:

$$C_f = \frac{2f''(0)}{\sqrt{Re_x}} \tag{18}$$

The C_f vs S plot (Figure 4) also shows a **monotonically increasing trend**.

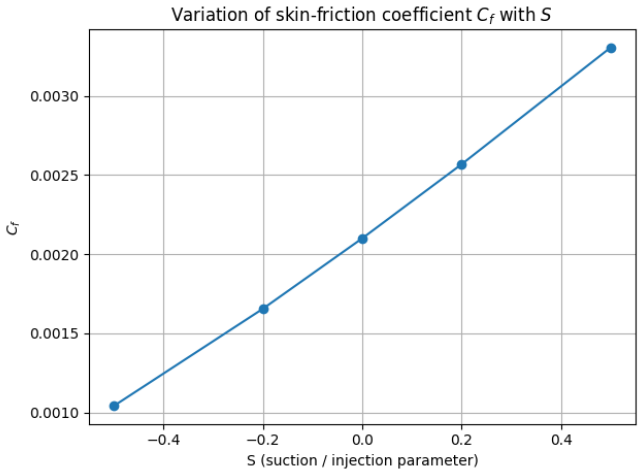

Figure 4. Plot for _C_f vs S

- **Injection** ($S < 0$)**:** Lowest skin friction → boundary layer relaxed.
- **Suction** ($S > 0$)**:** Highest skin friction → boundary layer tightened.
- C_f varies almost linearly with S, similar to $f''(0)$.

This confirms that **wall mass transfer strongly affects drag**, which is a key engineering insight.

7.5 *Variation of Boundary-Layer Thickness η_δ With S*

The boundary-layer thickness in similarity space is defined using the condition:

$$f'(\eta_\delta) = 0.99, \tag{19}$$

which corresponds to the point where the streamwise velocity reaches 99% of the free-stream value.

Figure 5 shows how η_δ varies with the suction/injection parameter S. A clear decreasing trend is observed: **suction** ($S > 0$) leads to a **thinner boundary layer**, while **injection** ($S < 0$) results in a **thicker boundary layer**.

This behavior is physically consistent, since suction removes low-momentum fluid near the wall, compressing the boundary layer, whereas injection introduces additional mass and causes the layer to grow. The extracted values plotted in Figure 5 clearly show:

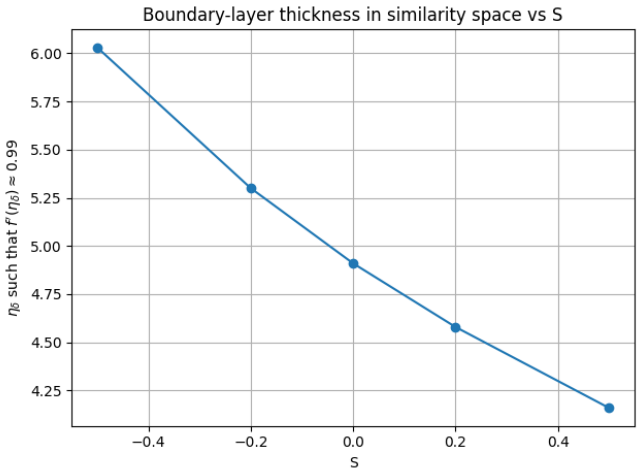

Figure 5. Variation of boundary-layer thickness η_δ (defined by $f'(\eta_\delta) = 0.99$) with suction/injection parameter S.

8. Comparision of RK-4 Method with FDM for S=0.2

To assess the accuracy and reliability of the numerical procedures used in solving the Blasius equation with suction/injection, a detailed comparison between the RK-4 shooting method and the Finite-Difference Method (FDM) is performed for a representative suction/injection parameter S = 0.2. The comparison is carried out for identical values of the similarity variable η, ensuring that both methods are evaluated under the same physical and numerical conditions.

Table below show the computed values of $f(\eta)$ and $f'(\eta)$ from FDM Method for **S=0.2**

Table 2. Computed Velocity and Stream Function Profiles using Finite Difference Method (FDM) for $S = 0.2$

η	$f(\eta)$	$f'(\eta)$
0	0.2	0
0.25	0.212	0.0998
0.75	0.309	0.291
1.5	0.626	0.549
2.25	1.117	0.753
3	1.736	0.888
4	2.674	0.972
5	3.661	0.995
6	4.659	0.999
7	5.659	0.999973

8.1 *Comparison of Computed Profiles*

For identical locations of the similarity variable η, the profiles $f(\eta)$ and $f'(\eta)$ obtained from both methods show very close agreement. Across the entire boundary–layer region, the deviation between the two methods remains of the order of 10^{-3} or smaller, indicating that both discretization strategies capture the Blasius-type behaviour reliably. The agreement remains

excellent both near the wall (where the curvature is highest) and in the far field (where $f'(\eta) \to 1$).

8.2 Behaviour of the Pointwise Error Curve

The pointwise error in the velocity profile, defined as

$$\varepsilon(\eta) = f'_{\mathrm{FDM}}(\eta) - f'_{\mathrm{RK4}}(\eta),$$

is shown in the comparison plot in Figure 6:

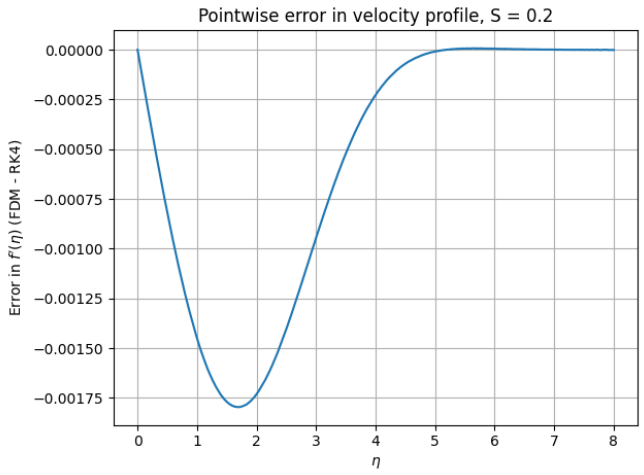

Figure 6. Comparative error analysis between RK-4 and FDM

The error curve exhibits a smooth, monotonic structure: the maximum deviation occurs around $\eta \approx 1.8$, corresponding to the region of largest solution curvature. This behaviour is consistent with classical numerical analysis, where second–order schemes such as central FDM accumulate their peak truncation error in regions with strong gradients. Beyond $\eta > 4$, the error rapidly decays to zero, demonstrating that both methods capture the asymptotic approach to the free stream identically.

8.3 Conclusions from the Comparative Study

The comparative results confirm the following:

- Both RK-4 and FDM generate nearly indistinguishable similarity profiles for $S = 0.2$, validating the correctness and robustness of the solvers.
- The RK-4 method exhibits marginally better accuracy in the region of maximum curvature, while the FDM solution converges exactly in the far field, as seen in Figure 6.
- The magnitude and smoothness of the error curve indicate numerical consistency between the two approaches, and no instability or systematic bias is observed.

Overall, the excellent agreement between RK-4 and FDM for $S = 0.2$ provides a strong cross-validation of both numerical frameworks and reinforces the reliability of the computed boundary–layer characteristics.

Conclusion

In this work, the modified Blasius equation governing laminar boundary-layer flow with wall suction and injection was solved numerically using the Shooting Method. The boundary-value problem was transformed into an initial-value problem and integrated using two independent numerical schemes: a fourth-order Runge–Kutta (RK-4) method and a central finite-difference method (FDM). This dual-approach framework directly addresses the core computational challenge of accurately resolving nonlinear boundary-layer equations under mass-transfer effects.

The numerical results clearly reveal how wall suction and injection modify the structure of the boundary layer. Suction ($S > 0$) stabilizes the flow by thinning the boundary layer, steepening the velocity gradient at the wall, and increasing the wall-shear parameter $f''(0)$. Injection ($S < 0$) produces the opposite effect, thickening the boundary layer and reducing shear. These trends align precisely with classical boundary-layer theory, validating the physical consistency of the computed solutions.

The comparison between RK-4 and FDM for the representative case $S = 0.2$ demonstrates strong numerical agreement. The pointwise error in $f'(\eta)$ remains below 10^{-3}, with the largest deviation occurring in regions of high curvature where discretization sensitivity is expected. The rapid decay of error in the far-field confirms that both solvers accurately capture the asymptotic free-stream behaviour. This cross-verification ensures robustness of the implemented methods and reliability of the numerical predictions.

Overall, the study successfully solves the modified Blasius problem under suction and injection, quantifies key boundary-layer characteristics, and validates the numerical scheme through an independent FDM comparison. The results provide a coherent and accurate interpretation of mass-transfer effects on laminar boundary layers and offer a dependable computational framework for further analysis of surface-transport and flow-control applications.

9. Appendix

9.1 Appendix A: Source Code Repository

All source codes used for the numerical simulations, including:

- C++ implementation of the Blasius equation using the Shooting + RK4 method,
- Python scripts for post–processing and plotting,
- FDM implementation for verification,
- All output `.txt` data files for different values of S,

are available at the following link:

Github Repository:
https://github.com/Shreyansh-Singh-code/CLL113termpaper

9.2 Appendix B:Derivation

The momentum thickness θ is defined as the distance by which the boundary should be displaced to compensate for the reduction in momentum of the flowing fluid on account of boundary layer formation.

$$\theta = \int_0^\infty \frac{u}{U_\infty}\left(1 - \frac{u}{U_\infty}\right) dy \tag{20}$$

Using the similarity transformation $\eta = y\sqrt{\frac{U_\infty}{2\nu x}}$, we have $dy = \sqrt{\frac{2\nu x}{U_\infty}}d\eta$. Substituting $u/U_\infty = f'(\eta)$:

$$\theta = \sqrt{\frac{2\nu x}{U_\infty}} \int_0^\infty f'(\eta)(1 - f'(\eta))d\eta \tag{21}$$

In dimensionless form, the integral $I = \int_0^\infty f'(1 - f')d\eta$ is computed numerically within the C++ code using the Trapezoidal rule.

9.3 Appendix C: Nomenclature

The following symbols and notation are used throughout this study.

Symbol	Description	SI Unit
x, y	Cartesian coordinates (along and normal to the plate)	m
u, v	Velocity components in x and y directions	m/s
U_∞	Free-stream velocity	m/s
V_w	Normal velocity at the wall (suction/injection)	m/s
ν	Kinematic viscosity	m^2/s
ρ	Fluid density	kg/m^3
η	Dimensionless similarity variable	–
$f(\eta)$	Dimensionless stream function	–
$f'(\eta)$	Dimensionless velocity profile (u/U_∞)	–
$f''(0)$	Dimensionless wall shear parameter	–
S	Dimensionless suction/injection parameter	–
Re_x	Local Reynolds number ($U_\infty x/\nu$)	–
C_f	Skin-friction coefficient	–
τ_w	Wall shear stress	N/m^2
θ	Momentum thickness	m
η_δ	Boundary layer thickness in similarity space	–

Self Assessment

In the beginning, we did not know much about boundary-layer equations or how suction and injection change the flow. Even the numerical methods like the Shooting Method, RK-4, and the Finite Difference Method (FDM) felt confusing at first. However, while coding, plotting, and verifying the results, the concepts gradually became clearer. We made many mistakes during the process, especially in coding and selecting appropriate parameters, but correcting them helped us understand the underlying ideas more deeply.

We believe that the report explains the problem in a clear and simple manner, and our results are consistent with the expected theoretical behaviour, which gives us confidence in the correctness of our calculations. At the same time, we recognise that there is always room for improvement — such as writing cleaner code, arranging the figures more neatly, and explaining certain steps with greater clarity.

Overall, we feel satisfied with our work because we genuinely learned a great deal while completing this project, both in terms of numerical methods and in understanding how to write a proper technical report.