ACTIVITY ONE

BLASIUS' SOLUTION FOR LAMINAR BOUNDARY LAYER OVER A TWO-DIMENSIONAL FLAT PLATE

BOUNDARY LAYER THEORY

AS – 4004

SUBMITTED BY

ABHISHEK MUKHERJEE

2027001

8TH SEMESTER

B. TECH IN AEROSPACE ENGINEERING

FACULTY

PROF. (DR.) SUMANTA CHAUDHURI

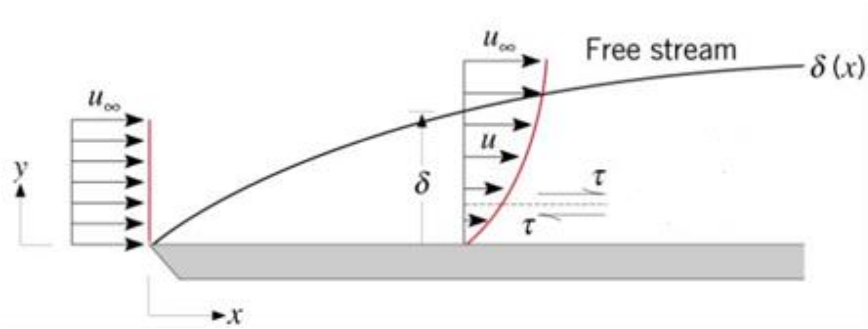
This report is submitted as a fulfilment to the curriculum requirement of the honours course 'Boundary Layer Theory' for the academic year 2023-24.

TABLE OF CONTENTS

INTRODUCTION

Heinrich Blasius' groundbreaking discovery in fluid dynamics, specifically his analysis of laminar boundary layers over flat plates, has had a significant impact on the field. His work, first published in 1908, has become a cornerstone for studying the movement of viscous fluids close to solid objects.

A boundary layer is a thin region of fluid adjacent to a solid surface where the fluid velocity changes from zero at the surface to the free stream velocity away from it. It is characterized by significant velocity gradients and shear stresses.



FLAT PLATE BOUNDARY LAYER GOVERNING EQUATIONS

The flat plate boundary layer governing equations, or simply, 'boundary layer equations' are a set of simplified Navier-Stokes equations that describe the flow of a thin layer of fluid adjacent to a solid surface. These equations are derived by accounting the following assumptions –

- Steady state
- Two-dimensional flow
- Incompressible flow
- Newtonian Working Fluid
- No Viscous Dissipation
- No Gravitational Influence

and neglecting pressure variations along the flow direction. The boundary layer equations govern the velocity and pressure distribution within the boundary layer and are derived as follows:

1. Conservation of mass

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0$$

2. Conservation of momentum
   a. $x-$ momentum

$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=v\frac{\partial^2 u}{\partial x^2}$$

   b. $y-$ momentum

$$\frac{\partial p}{\partial y}=0$$

where u, v are the velocities at x and y directions respectively, p is the pressure, ρ is the density, and υ is the kinematic viscosity.

No convenient analytical solution was available until Blasius presented his work on series solution for the same problem in 1908.

BLASIUS SOLUTION

Blasius introduced a similarity variable that combines independent variables x and y into one non-dimensional independent variable 'η' (Greek letter 'eta') to convert the given system of partial differential equations (PDEs) into a system of ordinary differential equations (ODE). This similarity parameter is defined as:

$$\eta(x, y) = \frac{y}{\delta(x)} = y\left(\frac{U_\infty}{\nu x}\right)^{\frac{1}{2}}$$

where U_∞ is the free-stream velocity along x direction and δ is the boundary layer thickness corresponding to the location in x direction.

The Blasius solution uses a stream function for the 2D flow. In terms of fluid velocities, we know the stream function as:

$$u = \frac{\partial \psi}{\partial y} \; and \; v = -\frac{\partial \psi}{\partial x}$$

This stream function, by definition, satisfies the continuity equation –

$$\frac{\partial}{\partial x}\frac{\partial \psi}{\partial y} - \frac{\partial}{\partial y}\frac{\partial \psi}{\partial x} = 0$$

In its scaled form, the stream function $f(\eta)$ is defined as –

$$\psi = f(\eta)\sqrt{\nu x U_\infty}$$

Here forward, $f(\eta)$ is denoted by f.

$$\therefore f = \frac{\psi}{U_\infty\sqrt{\frac{\nu x}{U_\infty}}}$$

Rewriting the terms of the x –momentum boundary layer equation in terms of f and η –

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

$$\frac{\partial u}{\partial x} = -\frac{1}{2}U_\infty \eta f'', \qquad \frac{\partial u}{\partial y} = U_\infty\sqrt{\frac{U_\infty}{\nu x}}f'', \qquad \frac{\partial^2 u}{\partial y^2} = \frac{U_\infty{}^2}{\nu x}f'''$$

Putting together all the above equations, we get –

$$\boxed{f''' + \frac{1}{2} f f'' = 0}$$

This equation is called the Blasius equation, a third-order ordinary differential equation. The boundary conditions for the Blasius equation follow from the wall no-slip condition and requirement that wall forms a streamline ($f = 0$), and the asymptotic requirement that the velocity becomes the freestream velocity outside the boundary layer:

$$f(\eta = 0) = 0, \qquad f'(\eta = 0) = 0, \qquad f'(\eta = \infty) = 1$$

To summarize –

- Non dimensional Stream Function : $f = \dfrac{\psi}{U_\infty \sqrt{\frac{\nu x}{U_\infty}}}$
- Non dimensional Velocity Profile : $f' = \dfrac{u}{U_\infty}$
- Normalized Shear Stress Function : $f'' = \dfrac{1}{U_\infty} \sqrt{\dfrac{\nu x}{U_\infty}} \dfrac{\partial u}{\partial y}$

CONVERT THE THIRD-ORDER ODE TO SYSTEM OF FIRST-ORDER ODE

Let

$$f_0 = f \qquad f_1 = f' = f_0{}' \qquad f_2 = f'' = f_1{}'$$

$$\therefore f_2' = f'''$$

From the Blasius equation,

$$f_2' = -\frac{1}{2} f_0 f_2$$

The third-order ODE is converted to a system of 3 first order coupled ODEs –

$$\boxed{\boldsymbol{f_0' = f_1, \qquad f_1' = f_2, \qquad f_2' = -\frac{1}{2} f_0 f_2}}$$

In matrix form, the system of ODEs is written as –

$$F' = \begin{bmatrix} f_0' \\ f_1' \\ f_2' \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ -\frac{1}{2} f_0 f_2 \end{bmatrix}$$

Also, the boundary conditions can be re-written as follows –

$$f_0(0) = 0, \qquad f_1(0) = 0, \qquad f_2(0) = ?$$

In the matrix form –

$$F(0) = \begin{bmatrix} 0 \\ 0 \\ ? \end{bmatrix}$$

Since $f_2(0)$ is unknown, the 'shooting method' is required to predict the value of $f_2(0)$. The algorithm of this method is as follows –

- Guess an initial value for $f_2(0)$. Here, 0.1 will be used.
- Solve the system of ODEs.
- Check if $f_1(\infty)$ asymptotically approaches 1. Set a convergence criterion to reach a closer value.
- If $f_1 > 1$, decrease the value of $f_2(0)$, else if $f_1 < 1$, increase the value of $f_2(0)$. Repeat until step 3 is satisfied. Here, the value is obtained to be 0.332.

Therefore, the updated boundary conditions are –

$$F(0) = \begin{bmatrix} 0 \\ 0 \\ 0.332 \end{bmatrix}$$

IMPLEMENTATION

The Blasius solution is implemented using a program written in Python language. To run this program, there are a few pre-requisites discussed in Appendix I.

Step #1: Import necessary python modules.

- *solve_ivp module from Scipy library.*
- *numpy package.*
- *pyplot module from Matplotlib library.*

```
from scipy.integrate import solve_ivp
import numpy as np
import matplotlib.pyplot as plt
```

Step #2: Define a system of linear ordinary differential equations.

```
def Blasius(eta, f):
    return (f[1], f[2], -0.5*f[0]*f[2])
```

Step #3: Define the system of boundary conditions.

```
f0 = [0, 0, 0.332]
```

Step #4: Solve the system of ODEs using the solve_ivp module.
solve_ivp uses Range-Kutta method to solve the given system of ODEs.

```
eta = np.linspace(0, 10, 100000)
f = solve_ivp(Blasius, [0, 10], f0, t_eval=eta)

x = f.t
f_0 = f.y[0]
f_1 = f.y[1]
f_2 = f.y[2]
```

Step #5: Extract the first instance of η at which $f'(\infty) = f_1(\infty) \sim 0.99$

```
for i in range(0, len(f_1)):
    if f_1[i] > 0.99:
        idx = i
        break

print(f"The value of η for f_1 ~ 0.99 is {round(eta[idx], 3)}")
```

Output: The value of η for f_1 ~ 0.99 is 4.912.

Therefore, the thickness of the laminar boundary layer (δ) is –

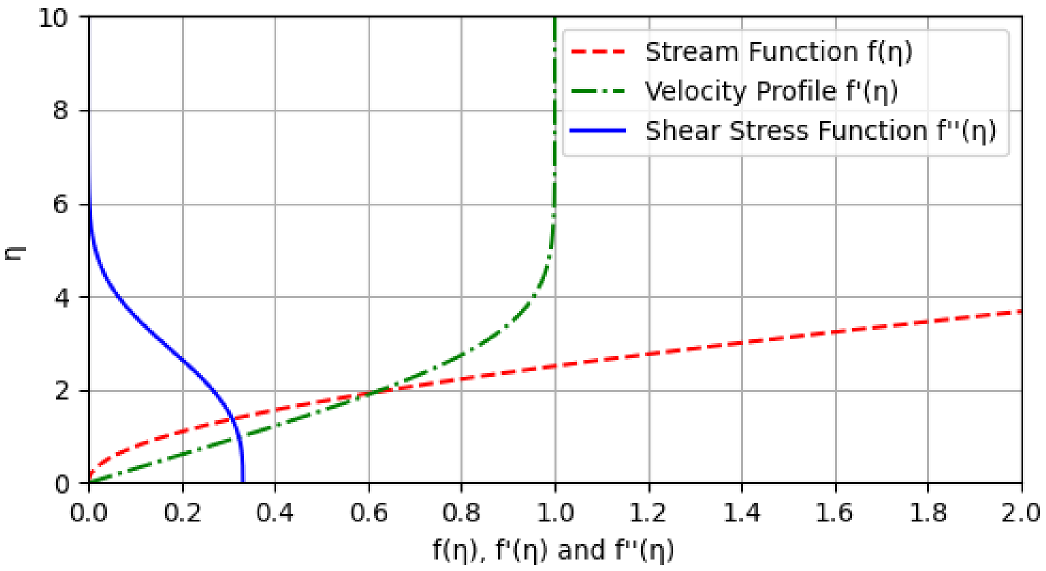
$$4.912 = \delta\left(\frac{U_\infty}{\nu x}\right)^{\frac{1}{2}}$$

$$\therefore\ \delta(x) = \frac{4.912x}{\sqrt{Re_x}}$$

Step #6: Plotting the solution.

```
plt.plot(f_0, x, label = "Stream Function f(η)")
plt.plot(f_1, x, label = "Velocity Profile f'(η)")
plt.plot(f_2, x, label = "Shear Stress Function f''(η)")

plt.legend()

plt.xlim(0, 2)
plt.xlabel("f(η), f'(η) and f''(η)")
plt.xticks(np.arange(0, 2.1, step = 0.2))

plt.ylim(0, 10)
plt.ylabel("η")

plt.grid(True)
plt.show()
```

Output:



FURTHER EXPLORING THE SOLUTION

For example, if the given flow properties are –

$$U_\infty = 100\, m/s\,, \qquad \rho = 1.225\, kg/m^3\,, \qquad \mu = 1.789 \times 10^{-5}\ Ns/m^2$$

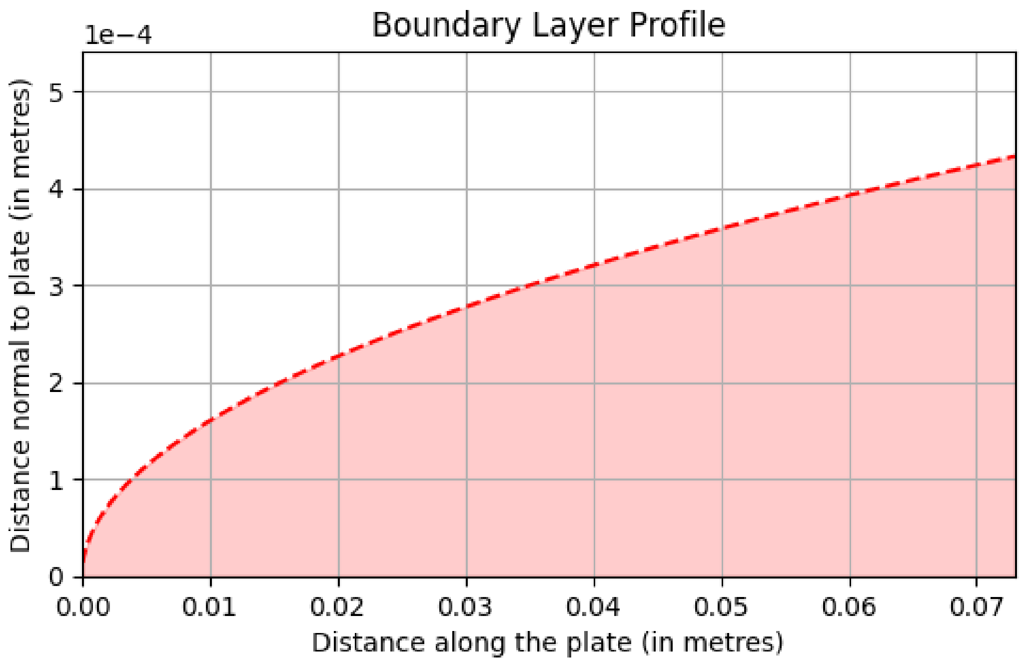
and the critical Reynold's number ($Re_{critical}$), i.e., the value Reynold's Number up to which the flow exhibits laminar behaviour, for flow over a flat plate is 5×10^5. Then

$$x_{crit} = \frac{Re_{crit} \times \mu}{\rho \times U_\infty} = \frac{5 \times 10^5 \times 1.789 \times 10^{-5}}{1.225 \times 100} = 0.073\, m$$

and

$$\delta_{max} = \frac{4.912 \times 0.073}{\sqrt{\frac{1.225 \times 100 \times 0.073}{1.789 \times 10^{-5}}}} = 4.33 \times 10^{-4}\, m$$

For this flow problem, the boundary profile will be –



REFERENCES

1. Ansys. (2020). *Flat-Plate Blasius Solution*. Ansys Innovation Space. https://courses.ansys.com/wp-content/uploads/2020/08/Lesson4-Flat-Plate-Blasius-Solution-Handout.pdf
2. M. H. T. (2019, March 14). *Blasius Solution*. YouTube. https://www.youtube.com/watch?v=0L4h-hqZY2Y
3. A. B. I. K. (2018, December 6). *[GNU OCTAVE] L4 Blasius equation and shooting method*. YouTube. https://www.youtube.com/watch?v=95ATrAZdIqA
4. M. H. T. (2019, March 14). *Blasius Derivation*. YouTube. https://www.youtube.com/watch?v=Lw6aQJGD3FU
5. Mattioli. (n.d.). Chapter 24. Blasius Solution. Principles of Fluid Dynamics. http://www.fluiddynamics.it/capitoli/blas.pdf

APPENDIX I

The full Python script is attached in Appendix I. To execute this program, the pre-requisites are –

- Python (https://www.python.org/ftp/python/3.12.3/python-3.12.3-amd64.exe)
- Any Text Editor (Visual Studio Code is preferred, but it requires a couple of steps to setup VSCode for Python. PyCharm is an alternative recommendation with no setup steps required)
- SciPy Library (run command in command prompt: pip install scipy)
- NumPy Module (run command in command prompt: pip install numpy)
- Matplotlib Library (run command in command prompt: pip install matplotlib)

```python
from scipy.integrate import solve_ivp
import numpy as np
import matplotlib.pyplot as plt
import math

def Blasius(eta, f):
    return (f[1], f[2], -0.5*f[0]*f[2])

eta = np.linspace(0, 10, 100000)
f0 = [0, 0, 0.332]

f = solve_ivp(Blasius, [0, 10], f0, t_eval=eta)

x = f.t
f_0 = f.y[0]
f_1 = f.y[1]
f_2 = f.y[2]

for i in range(0, len(f_1)):
    if f_1[i] > 0.99:
        idx = i
        break

print(f"The value of η for f_1 ~ 0.99 is {round(eta[idx], 3)}")

plt.plot(f_0, x, 'r--' , label = "Stream Function f(η)")
plt.plot(f_1, x, 'g-.', label = "Velocity Profile f'(η)")
plt.plot(f_2, x, 'b', label = "Shear Stress Function f''(η)")

plt.legend()

```

```
plt.xlim(0, 2)
plt.xlabel("f(η), f'(η) and f''(η)")
plt.xticks(np.arange(0, 2.1, step = 0.2))

plt.ylim(0, 10)
plt.ylabel("η")

plt.grid(True)
ratio = 9/16
plt.gca().set_aspect(abs((2-0)/(0-10))*ratio)

plt.show()

# Boundary Layer Profile

U_infty = 100
rho = 1.225
mu = 1.789e-5
nu = mu / rho

Re_crit = 5e5
x_crit = Re_crit * nu / U_infty

x = np.linspace(0, x_crit, 100000)
delta = 4.192 * x / (rho * x / nu)**(1/2)
delta_max = delta[-1]

print(f"The value of x_crit is {x_crit} metres")
print(f"The value of delta_max is {delta_max} metres")

plt.plot(x, delta, 'r--')
plt.fill_between(x, delta, 0, color = "red", alpha = 0.2)

plt.title("Boundary Layer Profile")

plt.xlim(0, x_crit)
plt.xlabel("Distance along the plate (in metres)")

plt.ylim(0, delta_max*1.25)
plt.ylabel("Distance normal to plate (in metres)")

plt.gca().set_aspect(abs((x_crit-0)/(0-delta_max*1.25))*ratio)
plt.ticklabel_format(style='sci', scilimits=(-3, 0),axis='both')
plt.grid(True)

plt.show()
```

APPENDIX II

η	f_0	f_1	f_2
0.000000	0.000000	0.000000	0.332000
0.023411	0.000091	0.007773	0.332000
0.046823	0.000364	0.015545	0.331999
0.070234	0.000819	0.023318	0.331997
0.093645	0.001456	0.031090	0.331992
0.117057	0.002275	0.038862	0.331985
0.140468	0.003275	0.046635	0.331975
0.163880	0.004458	0.054406	0.331960
0.187291	0.005823	0.062178	0.331940
0.210702	0.007369	0.069949	0.331914
0.234114	0.009098	0.077719	0.331882
0.257525	0.011008	0.085488	0.331843
0.280936	0.013101	0.093257	0.331796
0.304348	0.015375	0.101024	0.331741
0.327759	0.017831	0.108790	0.331677
0.351171	0.020469	0.116554	0.331603
0.374582	0.023288	0.124316	0.331518
0.397993	0.026289	0.132077	0.331422
0.421405	0.029471	0.139835	0.331314
0.444816	0.032834	0.147590	0.331193
0.468227	0.036380	0.155343	0.331059
0.491639	0.040106	0.163092	0.330911
0.515050	0.044014	0.170837	0.330749
0.538462	0.048103	0.178579	0.330570
0.561873	0.052374	0.186316	0.330376
0.585284	0.056826	0.194048	0.330165
0.608696	0.061459	0.201775	0.329936
0.632107	0.066272	0.209497	0.329689
0.655518	0.071267	0.217212	0.329424
0.678930	0.076442	0.224921	0.329139
0.702341	0.081798	0.232623	0.328834
0.725753	0.087334	0.240317	0.328508
0.749164	0.093051	0.248004	0.328161
0.772575	0.098947	0.255682	0.327792
0.795987	0.105023	0.263352	0.327401
0.819398	0.111279	0.271011	0.326986
0.842809	0.117714	0.278661	0.326548

0.866221	0.124328	0.286301	0.326086
0.889632	0.131120	0.293929	0.325598
0.913043	0.138091	0.301545	0.325086
0.936455	0.145241	0.309149	0.324547
0.959866	0.152568	0.316741	0.323982
0.983278	0.160073	0.324319	0.323390
1.006689	0.167755	0.331882	0.322771
1.030100	0.175614	0.339431	0.322123
1.053512	0.183649	0.346965	0.321447
1.076923	0.191860	0.354482	0.320742
1.100334	0.200247	0.361982	0.320008
1.123746	0.208810	0.369465	0.319243
1.147157	0.217547	0.376930	0.318448
1.170569	0.226459	0.384376	0.317622
1.193980	0.235544	0.391802	0.316764
1.217391	0.244802	0.399209	0.315873
1.240803	0.254232	0.406597	0.314950
1.264214	0.263834	0.413964	0.313994
1.287625	0.273606	0.421310	0.313003
1.311037	0.283549	0.428634	0.311980
1.334448	0.293662	0.435933	0.310922
1.357860	0.303945	0.443208	0.309831
1.381271	0.314398	0.450458	0.308705
1.404682	0.325019	0.457681	0.307545
1.428094	0.335809	0.464877	0.306351
1.451505	0.346766	0.472044	0.305123
1.474916	0.357891	0.479181	0.303860
1.498328	0.369182	0.486289	0.302562
1.521739	0.380639	0.493365	0.301230
1.545151	0.392261	0.500409	0.299864
1.568562	0.404048	0.507420	0.298463
1.591973	0.415999	0.514397	0.297028
1.615385	0.428113	0.521340	0.295559
1.638796	0.440390	0.528247	0.294055
1.662207	0.452829	0.535117	0.292517
1.685619	0.465428	0.541950	0.290946
1.709030	0.478188	0.548745	0.289340
1.732441	0.491107	0.555502	0.287701
1.755853	0.504185	0.562218	0.286028
1.779264	0.517420	0.568895	0.284322
1.802676	0.530812	0.575530	0.282582

1.826087	0.544359	0.582123	0.280810
1.849498	0.558062	0.588673	0.279006
1.872910	0.571918	0.595181	0.277169
1.896321	0.585927	0.601643	0.275300
1.919732	0.600088	0.608062	0.273400
1.943144	0.614400	0.614434	0.271468
1.966555	0.628862	0.620761	0.269506
1.989967	0.643472	0.627041	0.267513
2.013378	0.658230	0.633273	0.265489
2.036789	0.673134	0.639457	0.263436
2.060201	0.688184	0.645592	0.261354
2.083612	0.703377	0.651678	0.259243
2.107023	0.718713	0.657714	0.257103
2.130435	0.734191	0.663700	0.254936
2.153846	0.749809	0.669634	0.252742
2.177258	0.765567	0.675517	0.250520
2.200669	0.781462	0.681347	0.248273
2.224080	0.797493	0.687125	0.246000
2.247492	0.813660	0.692849	0.243702
2.270903	0.829960	0.698520	0.241379
2.294314	0.846392	0.704136	0.239033
2.317726	0.862956	0.709697	0.236664
2.341137	0.879649	0.715203	0.234272
2.364548	0.896470	0.720654	0.231858
2.387960	0.913417	0.726048	0.229424
2.411371	0.930489	0.731385	0.226969
2.434783	0.947685	0.736666	0.224495
2.458194	0.965002	0.741888	0.222002
2.481605	0.982440	0.747053	0.219491
2.505017	0.999997	0.752160	0.216964
2.528428	1.017670	0.757208	0.214420
2.551839	1.035459	0.762197	0.211860
2.575251	1.053361	0.767127	0.209286
2.598662	1.071378	0.771996	0.206698
2.622074	1.089509	0.776804	0.204096
2.645485	1.107752	0.781550	0.201483
2.668896	1.126105	0.786235	0.198859
2.692308	1.144567	0.790859	0.196225
2.715719	1.163137	0.795420	0.193584
2.739130	1.181812	0.799920	0.190936
2.762542	1.200593	0.804358	0.188282

2.785953	1.219476	0.808735	0.185624
2.809365	1.238460	0.813049	0.182963
2.832776	1.257545	0.817301	0.180300
2.856187	1.276728	0.821492	0.177636
2.879599	1.296008	0.825620	0.174972
2.903010	1.315384	0.829686	0.172310
2.926421	1.334853	0.833691	0.169650
2.949833	1.354416	0.837634	0.166993
2.973244	1.374069	0.841515	0.164342
2.996656	1.393813	0.845334	0.161695
3.020067	1.413645	0.849093	0.159056
3.043478	1.433563	0.852789	0.156424
3.066890	1.453568	0.856425	0.153800
3.090301	1.473656	0.859999	0.151186
3.113712	1.493827	0.863513	0.148582
3.137124	1.514080	0.866966	0.145989
3.160535	1.534412	0.870359	0.143408
3.183946	1.554824	0.873691	0.140841
3.207358	1.575312	0.876964	0.138287
3.230769	1.595876	0.880177	0.135748
3.254181	1.616515	0.883331	0.133224
3.277592	1.637227	0.886426	0.130716
3.301003	1.658011	0.889462	0.128225
3.324415	1.678866	0.892439	0.125751
3.347826	1.699789	0.895359	0.123296
3.371237	1.720781	0.898222	0.120860
3.394649	1.741839	0.901027	0.118444
3.418060	1.762963	0.903775	0.116047
3.441472	1.784151	0.906467	0.113672
3.464883	1.805401	0.909103	0.111318
3.488294	1.826713	0.911684	0.108986
3.511706	1.848085	0.914210	0.106676
3.535117	1.869517	0.916681	0.104390
3.558528	1.891006	0.919099	0.102128
3.581940	1.912551	0.921463	0.099889
3.605351	1.934152	0.923775	0.097675
3.628763	1.955806	0.926034	0.095486
3.652174	1.977514	0.928242	0.093323
3.675585	1.999273	0.930399	0.091185
3.698997	2.021083	0.932505	0.089073
3.722408	2.042942	0.934562	0.086989

3.745819	2.064849	0.936569	0.084931
3.769231	2.086802	0.938528	0.082900
3.792642	2.108802	0.940440	0.080897
3.816054	2.130846	0.942305	0.078921
3.839465	2.152933	0.944123	0.076974
3.862876	2.175063	0.945896	0.075055
3.886288	2.197233	0.947624	0.073165
3.909699	2.219444	0.949308	0.071303
3.933110	2.241693	0.950949	0.069470
3.956522	2.263980	0.952547	0.067666
3.979933	2.286304	0.954105	0.065892
4.003344	2.308663	0.955621	0.064146
4.026756	2.331057	0.957098	0.062430
4.050167	2.353484	0.958536	0.060743
4.073579	2.375943	0.959936	0.059086
4.096990	2.398434	0.961299	0.057458
4.120401	2.420955	0.962625	0.055860
4.143813	2.443506	0.963915	0.054292
4.167224	2.466087	0.965169	0.052753
4.190635	2.488697	0.966387	0.051244
4.214047	2.511334	0.967570	0.049764
4.237458	2.533999	0.968719	0.048315
4.260870	2.556691	0.969835	0.046895
4.284281	2.579408	0.970917	0.045505
4.307692	2.602151	0.971967	0.044144
4.331104	2.624917	0.972986	0.042813
4.354515	2.647707	0.973974	0.041511
4.377926	2.670520	0.974931	0.040238
4.401338	2.693356	0.975859	0.038994
4.424749	2.716213	0.976758	0.037779
4.448161	2.739090	0.977628	0.036592
4.471572	2.761988	0.978471	0.035433
4.494983	2.784905	0.979286	0.034302
4.518395	2.807842	0.980075	0.033199
4.541806	2.830796	0.980839	0.032123
4.565217	2.853768	0.981578	0.031074
4.588629	2.876757	0.982292	0.030051
4.612040	2.899763	0.982982	0.029054
4.635452	2.922785	0.983650	0.028083
4.658863	2.945821	0.984295	0.027137
4.682274	2.968873	0.984919	0.026215

4.705686	2.991939	0.985521	0.025317
4.729097	3.015018	0.986103	0.024443
4.752508	3.038111	0.986666	0.023592
4.775920	3.061216	0.987209	0.022763
4.799331	3.084333	0.987734	0.021957
4.822742	3.107463	0.988240	0.021173
4.846154	3.130604	0.988728	0.020412
4.869565	3.153756	0.989198	0.019672
4.892977	3.176919	0.989652	0.018954
4.911388	**3.200093**	**0.990089**	**0.018257**
4.939799	3.223276	0.990510	0.017581
4.963211	3.246470	0.990914	0.016926
4.986622	3.269673	0.991304	0.016291
5.010033	3.292885	0.991679	0.015677
5.033445	3.316106	0.992039	0.015082
5.056856	3.339335	0.992385	0.014506
5.080268	3.362572	0.992718	0.013949
5.103679	3.385817	0.993037	0.013410
5.127090	3.409070	0.993344	0.012889
5.150502	3.432329	0.993638	0.012386
5.173913	3.455596	0.993921	0.011899
5.197324	3.478869	0.994193	0.011429
5.220736	3.502148	0.994453	0.010975
5.244147	3.525434	0.994703	0.010536
5.267559	3.548725	0.994943	0.010112
5.290970	3.572021	0.995174	0.009702
5.314381	3.595322	0.995395	0.009306
5.337793	3.618629	0.995608	0.008922
5.361204	3.641940	0.995812	0.008551
5.384615	3.665255	0.996008	0.008193
5.408027	3.688575	0.996196	0.007847
5.431438	3.711900	0.996377	0.007514
5.454849	3.735228	0.996549	0.007192
5.478261	3.758560	0.996714	0.006883
5.501672	3.781896	0.996872	0.006585
5.525084	3.805236	0.997024	0.006298
5.548495	3.828579	0.997168	0.006022
5.571906	3.851926	0.997306	0.005757
5.595318	3.875276	0.997438	0.005502
5.618729	3.898629	0.997564	0.005257
5.642140	3.921985	0.997684	0.005022

5.665552	3.945343	0.997799	0.004797
5.688963	3.968704	0.997908	0.004581
5.712375	3.992068	0.998012	0.004373
5.735786	4.015435	0.998112	0.004174
5.759197	4.038803	0.998207	0.003983
5.782609	4.062174	0.998297	0.003800
5.806020	4.085547	0.998384	0.003624
5.829431	4.108921	0.998466	0.003455
5.852843	4.132298	0.998545	0.003293
5.876254	4.155676	0.998620	0.003137
5.899666	4.179056	0.998692	0.002987
5.923077	4.202437	0.998761	0.002844
5.946488	4.225820	0.998826	0.002706
5.969900	4.249205	0.998888	0.002574
5.993311	4.272591	0.998947	0.002448
6.016722	4.295978	0.999003	0.002328
6.040134	4.319366	0.999057	0.002213
6.063545	4.342756	0.999107	0.002103
6.086957	4.366147	0.999156	0.001998
6.110368	4.389539	0.999201	0.001898
6.133779	4.412932	0.999245	0.001803
6.157191	4.436327	0.999286	0.001712
6.180602	4.459722	0.999324	0.001626
6.204013	4.483118	0.999361	0.001544
6.227425	4.506515	0.999396	0.001465
6.250836	4.529913	0.999429	0.001391
6.274247	4.553311	0.999460	0.001320
6.297659	4.576711	0.999490	0.001252
6.321070	4.600110	0.999518	0.001187
6.344482	4.623511	0.999545	0.001126
6.367893	4.646912	0.999571	0.001067
6.391304	4.670314	0.999595	0.001010
6.414716	4.693716	0.999618	0.000956
6.438127	4.717118	0.999640	0.000904
6.461538	4.740522	0.999661	0.000855
6.484950	4.763925	0.999680	0.000809
6.508361	4.787329	0.999699	0.000765
6.531773	4.810734	0.999716	0.000722
6.555184	4.834139	0.999733	0.000683
6.578595	4.857544	0.999748	0.000645
6.602007	4.880950	0.999763	0.000609

6.625418	4.904356	0.999777	0.000575
6.648829	4.927762	0.999790	0.000543
6.672241	4.951168	0.999802	0.000513
6.695652	4.974575	0.999814	0.000484
6.719064	4.997983	0.999825	0.000457
6.742475	5.021390	0.999835	0.000431
6.765886	5.044798	0.999845	0.000407
6.789298	5.068206	0.999854	0.000384
6.812709	5.091614	0.999862	0.000362
6.836120	5.115022	0.999871	0.000341
6.859532	5.138430	0.999878	0.000322
6.882943	5.161839	0.999886	0.000303
6.906355	5.185248	0.999893	0.000285
6.929766	5.208657	0.999899	0.000268
6.953177	5.232066	0.999905	0.000252
6.976589	5.255475	0.999911	0.000237
7.000000	5.278884	0.999916	0.000222