"

Bryce Lindsey - MAE5943 HW7 (Hiemenz flow)

```
This document summarizes the HW7 assignment for MAE5943
(problem statement seen below)

submitted by:
    Bryce Lindsey
    bryce.lindsey@okstate.edu
```

HW7 Problem Statement :

The stagnation point flow problem towards an infinite flat plate can be represented using the ordinary differential equation (ODE) below:

$$\varphi''' + \varphi\varphi'' - (\varphi')^2 + 1 = 0$$

Boundary Conditions:

$$\varphi_{(\eta=0)} = 0$$

$$\varphi'_{(\eta=0)} = 0$$

$$\varphi'_{(\eta\to\infty)} \to 0$$

Solve the Hiemenz flow problem using the numerical methods discussed in the lecture.

- Verify the formulas for coefficients A, B, C, and D of the tridiagonal matrix.
- Investigate the effect of $\eta_{\mathbf{max}}$ and $N_{\mathbf{max}}$.
  - Assume that $N_{\mathbf{max}}$=40 and compare the solutions for $\eta_{\mathbf{max}}$=[5, 10, 20].
  - Assume $\eta_{\mathbf{max}}$=10 and compare the solution for $N_{\mathbf{max}}$=[10, 20, 40, 80].
- Discuss your observations and submit your findings in a technical report format.
- Include your code in the appendix.

Hiemenz Flow

Hiemenz flow is an approach to solve a flow approaching a body, where a stagnation point exists, by splitting the analysis into the inviscid outer region and the viscous boundary layer region.

We showed in lecture that Hiemenz flow can be represented by the following ODE:

$$\phi''' + \phi\phi'' - \phi'^2 + 1 = 0$$

In this ODE, the parameter φ' corresponds to the velocity profile at a particular location along the body undergoing the stagnation flow. φ' is a function of η, which is a non-dimensional term which can be associated with a location on the body (if desired).

Boundary Conditions:

$$\phi'(0) = 0$$

$$\phi(0) = 0$$

$$\phi'(\eta \to \infty) = 1$$

Tranformation:

with φ' = h and φ = p, and using Thomas' algorithm, this ODE can be numerically solved.

Hiemenz flow function

My Hiemenz flow function is defined in the block below

Inputs

- *ηmax* = Maximum value of η (treat as infinity boundary condition)
- *N* = number of discrete points throughout η to use
- *error* = Each iteration of "h" will compute the error between the current h and the last h. Choose "error" to be small so that h converges.

Output

- *η* = Output values of η.
- *h* = Is refined throughout the h convergence loops and is equivalent to the primary output parameter we want, φ' (which corresponds to a local velocity profile).
- *p_hchange* = Optional output plot to capture. This is a plot that shows the convergence of "h". Not necessary, but can be useful as a sanity check.

hiemenz

Computes the Hiemenz flow transformed solution

```
hiemenz(ηmax, N, error)
```

If the arguments are missing, it will use the default values.

```
hiemenz(ηmax=10, N=50, error=1e-6)
```

```
"""
Computes the Hiemenz flow transformed solution

    hiemenz(ηmax, N, error)

If the arguments are missing, it will use the default values.

    hiemenz(ηmax=10, N=50, error=1e-6)

"""
function hiemenz(ηmax=10, N=50, error=1e-6)
    plot_pallette = palette(:seaborn_rocket_gradient, 23)
```

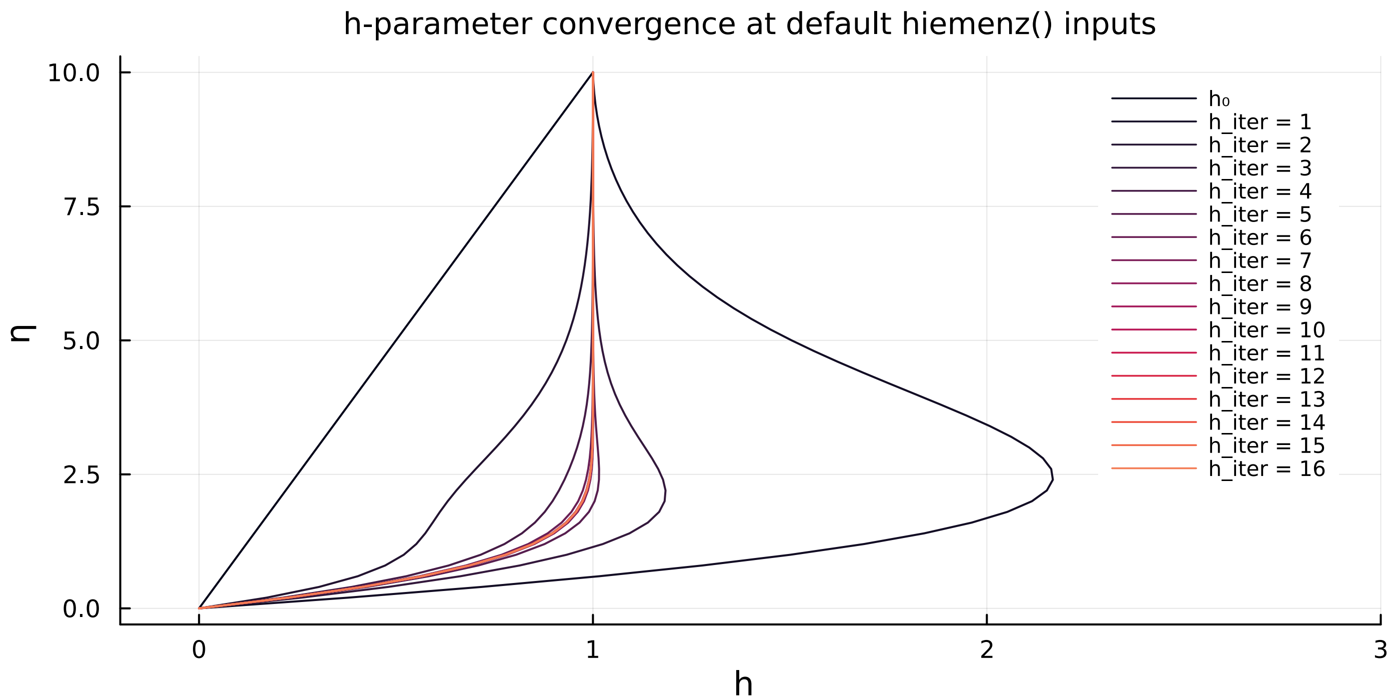
```julia
Δη = ηmax/N
Δη²= Δη^2;

# Initialization of the arrays
A = zeros(N+1)
B = zeros(N+1)
C = zeros(N+1)
D = ones(N+1)
G = zeros(N+1)
H = zeros(N+1)
P = zeros(N+1)
η = zeros(N+1)

# make η vector
η = [(i-1)*Δη for i=1:N+1];

# initial guess at h (linear)
h = Vector(LinRange(0,1,N+1))

G[1] = 1.0
H[1] = 0.0; #Assumed
P[1] = 0.0

gp=0        # G[i-1], used in loop
hp=0        # H[i-1], used in loop

p_hchange = plot(h, η,
color=plot_pallette[1],
label="h₀")

ϵ = 100*ones(length(h))     # initial error of h (high values)
hi_counter = 1
while maximum(abs.(ϵ)) > error

    for i=2:N+1
        P[i] = P[i-1] + ((h[i-1]+h[i])*Δη)/2
    end


    A = [ 1/Δη² + P[i]/(2*Δη) for i=1:N+1]
    B = [-2/Δη² - h[i]        for i=1:N+1]
    C = [ 1/Δη² - P[i]/(2*Δη) for i=1:N+1]

    for i=1:N+1
        if i == 1
            H[i] = 0
            G[i] = 0
        else
            gp = G[i-1]
```

```
                hp = H[i-1]
                G[i] = -(C[i]*gp + D[i])/(B[i] + C[i]*hp)
                H[i] = -A[i]/(B[i] + C[i]*hp)
            end
        end

        h_old = copy(h);
        for i = N:-1:2
            h[i] = G[i] + H[i]*h[i+1]
        end
        ϵ = h_old.-h

        plot!(p_hchange,h,η,
        linecolor=plot_pallette[1+hi_counter],
        label="h_iter = $hi_counter")
        hi_counter += 1

    end

    plot!(p_hchange,
    legend_foreground_color=nothing,
    legend_font=7)
    return η,h, p_hchange


end


# end of function
```

Example of "h" (ultimately, φ') convergence:

To show the convergence of "h", I'll execute my Hiemenz flow function at its default values and keep the output h-convergence plot

As you can see, it took 16 iterations for h to converge using the default values.




```julia
begin
# call the function
    ηₒᵤₜ, ϕₒᵤₜ, hₚₗₒₜ = hiemenz();

    # plot the output hₚₗₒₜ plot object
    plot!(hₚₗₒₜ,
    xlabel="h \n",
    ylabel="\n η",
        xlim=[-.2, 3],
    title="h-parameter convergence at default hiemenz() inputs",
    titlefontsize=10,
    fontfamily="sans-serif",
    size=(700,350))
    current()
end
```

Tasks given in problem statement

Task 1: Validate A,B,C, and D Terms

HW 7: A,B,C,D validation - Bryce Lindsey

from lecture → $\left\{ \begin{array}{l} h'' + ph' - h^2 + 1 = 0 \\ p' - h = 0 \end{array} \right\}$; $\phi = p$, $\phi' = h$

applying finite difference gives
and ignoring $O(\Delta\eta^2)$ terms gives: $h_n'' = \dfrac{h_{n+1} - 2h_n + h_{n-1}}{(\Delta\eta)^2} + O(\Delta\eta^2)$

$$h' = \frac{h_{n+1} - h_{n-1}}{2\Delta\eta} + O(\Delta\eta^2)$$

plugging these into ODE gives

$$\frac{h_{n+1} - 2h_n + h_{n-1}}{(\Delta\eta)^2} + P_n \frac{h_{n+1} - h_{n-1}}{2\Delta\eta} - h_2^2 + 1 = 0$$

$$\hookrightarrow \quad h_{n+1}\left[\frac{1}{\Delta\eta^2} + \frac{P_n}{2\Delta\eta}\right] + h_n\left[\frac{(-2)}{\Delta\eta^2} - h_n\right] + h_{n-1}\left[\frac{1}{\Delta\eta^2} - \frac{P_n}{2\Delta\eta}\right] + 1 = 0$$

Final F.D. system of equations: $A_n h_{n+1} + B_n h_n + C_n h_{n-1} + D_n = 0$

Thus,

$$\underbrace{h_{n+1}\left[\frac{1}{\Delta\eta^2} + \frac{P_n}{2\Delta\eta}\right]}_{A_n} + \underbrace{h_n\left[\frac{(-2)}{\Delta\eta^2} - h_n\right]}_{B_n} + \underbrace{h_{n-1}\left[\frac{1}{\Delta\eta^2} - \frac{P_n}{2\Delta\eta}\right]}_{C_n} + \underbrace{1}_{D_n} = 0$$

$$\Rightarrow \quad \boxed{\begin{aligned} A_n &= \frac{1}{(\Delta\eta)^2} + \frac{P_n}{2\Delta\eta} \\ B_n &= \frac{-2}{(\Delta\eta)^2} - h_n \\ C_n &= \frac{1}{(\Delta\eta)^2} - \frac{P_n}{2\Delta\eta} \\ D_n &= 1 \end{aligned}}$$

Task 2: Investigate effects of η_{max} changes

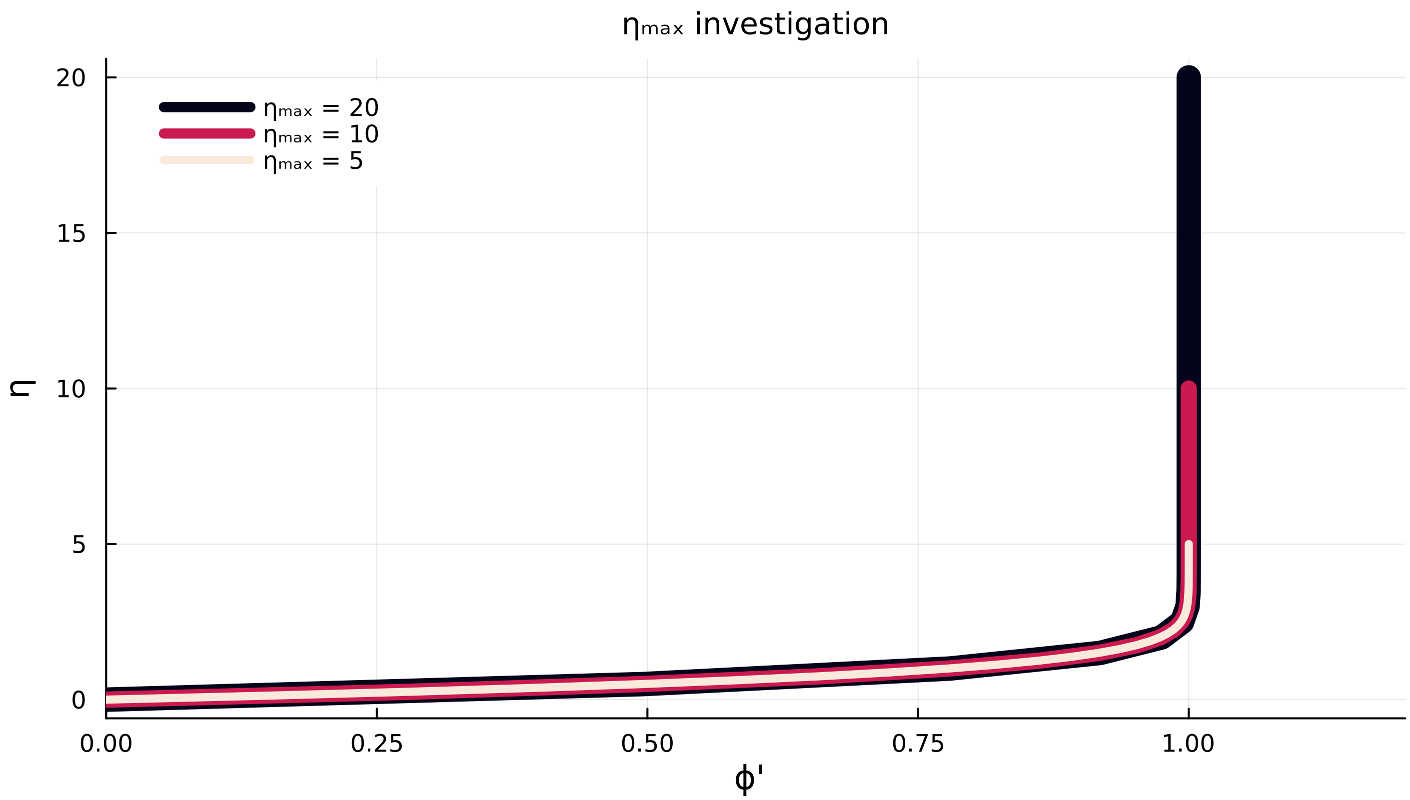
According to the problem statement, the following inputs will be used

N = 40

ηmax = 5 , 10, 20

error = 1e-6 (my default)

The results of η vs φ' will be plotted for each ηmax



```julia
let
    p_ηmax_comp = plot()
    my_ηmax = [5, 10, 20]
    plot_pallette = palette(:seaborn_rocket_gradient, length(my_ηmax))
    lwscale = 4
    my_lws = range(lwscale*length(my_ηmax), lwscale*1, length(my_ηmax))
    ic=1
    for ηmax_i = reverse(my_ηmax)
        # the h convergence plot is not needed
        cur_η, cur_h, p = hiemenz(ηmax_i, 40, 1e-6)
        plot!(p_ηmax_comp, cur_h, cur_η;
            linecolor=plot_pallette[ic],
            linewidth=my_lws[ic],
            label="ηₘₐₓ = $ηmax_i")
        ic+=1
    end
    plot!(p_ηmax_comp,
        xlabel="ϕ' \n",
        ylabel="η",
        xlim=[0, 1.2],
        legend_foreground_color=nothing,
        legend_font=8,
        title="ηₘₐₓ investigation",
        titlefontsize=10,
        fontfamily="sans-serif",
        size=(700,400))
    current()

end
```

Task 2 Discusion

As ηmax increases, regions beyond the boundary layer are better captured. However, at the lowest ηmax value of 5, the entire viscous region is captured. Just how high to set ηmax depends on how much of the inviscous region one would like to capture, but a ηmax value of 20 does a very good job of capturing all of the velocity profile out to the freestream and beyond.

Task 3: Investigate effects of N changes

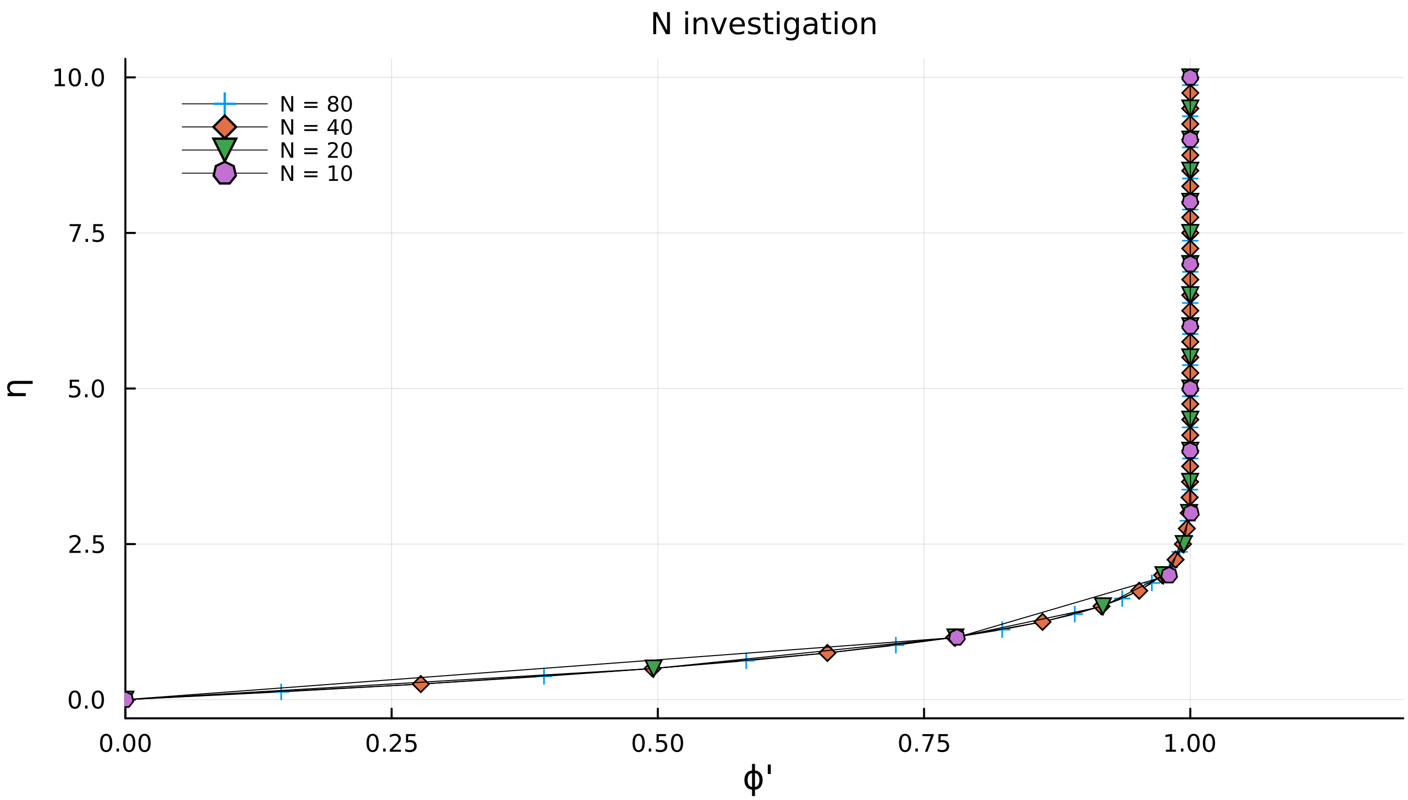
According to the problem statement, the following inputs will be used

N = 10, 20, 40, 80

ηmax = 10

error = 1e-6 (my default)

The results of η vs φ' will be plotted for each ηmax



```
let
    p_N_comp = plot()
    N_changes = [10, 20, 40, 80]
    my_markers = Plots.supported_markers();
    ic=1
    for N_i = reverse(N_changes)
        cur_η, cur_h, p = hiemenz(10, N_i, 1e-6)
        plot!(p_N_comp, cur_h, cur_η;
            marker=my_markers[ic+3],
            linewidth=.5,
            linecolor=:black,
            label="N = $N_i")
        ic+=1
    end
    plot!(p_N_comp,
        xlabel="ϕ' \n",
        ylabel="η",
        xlim=[0, 1.2],
        legend_foreground_color=nothing,
        legend_font=7,
        title="N investigation",
        titlefontsize=10,
        fontfamily="sans-serif",
        size=(700,400))
    current()

end
```

Task 3 Discussion

The full "shape" of the transformed velocity profile, φ', is captured in each instance of different values of "N". However, as the value of N increases, the resolution of the solution increases. The line that describes φ' becomes more refined and confidence in answers rise.

END OF ASSIGNMENT :)