3AN Project 1
Shooting Newton and Kantorovich with matlab

Liam Watson WTSLIA001

May 16, 2021

When modeling vortices a common Boundary Value problem that arises is as follows

$$\frac{d^2u}{dr^2} + \frac{1}{r}\frac{du}{dr} + \frac{u}{1-u^2}\left[\left(\frac{du}{dr}\right)^2 - \frac{n^2}{r^2}\right] + u(1-u^2) = 0 \tag{1}$$

$$u(0) = 0, \quad u(\infty) = 1 \tag{2}$$

Where

$n \in \mathbb{N}/\{0\}$	Is the vorticity
$u = u(r)$	Is our unknown function that is monotonically growing

In this paper we will solve the BVP (1), (2) numerically using the shooting method and the Newton-Kantorovich method with Finite differences for $n \in \{1, 2, 3\}$. The numerical solution to this BVP is sensitive to many parameters that we must choose. We will discuss the effect of our choices for parameters such as how we deal with $u(\infty)$, $\frac{u}{1-u^2}$ being undefined at $u = 1$ or $\frac{1}{r}$ for $u(0) = 0$ being undefined, the number of iterations needed, the value for n, mesh spacing, length of spacial interval, initial guess.

1 Numerical methods overview

Here we will discuss the different numerical schemes with their relative benefits and drawbacks

1.1 Shooting method

The general principle for the shooting method is to reduce our boundary value problem into an initial value problem. We do this by taking the first boundary condition as the first initial condition and some arbitrary second variable for the second initial condition, we call this value the shooting parameter. We must then find a value of the shooting parameter so that our second boundary condition is met. We can do this using the bisection or Netwton's method as we can phrase the problem as that of a root problem.

1.2 Newton-Kantorovich method

The general principle for Newton-Kantorovich is to find the system of Kantorovich equations by finding the Freche derivative. In our case this will produce a second order linear boundary value problem which we can solve using linear shooting or finite difference methods. The solution to this system is the correction to some initial guess for the problem stated in (1) and (2).

2 Derivation of needed formula

In order to begin the implementation we need to first derive the expressions needed for the two numerical methods from (1) and (2).

3 Shooting method

The differential equation in (1) is non-linear so we must use non-linear shooting to solve it.
First we must derive an initial value problem from the boundary value problem which is described bellow

$$\begin{cases} u'' = -\frac{1}{r}\frac{du}{dr} - \frac{u}{1-u^2}\left[\left(\frac{du}{dr}\right)^2 - \frac{n^2}{r^2}\right] - u(1-u^2) \\ u(0) = 0 \\ u'(0) = p \end{cases} \tag{3}$$

Where p is our shooting parameter.
We now have two choices for solving for p namely, bisection or Newton's method. We do not yet know which of these will give better results so let us implement both.
For bisection there is nothing for us do derive and the details will be covered in the implementation section.
For Newtons method we must derive a scheme to update p based on the function u and the second boundary condition.
We introduce:

$$f' = \frac{\partial}{\partial p}\left(u(p, x = \infty) - 1\right) = \frac{\partial}{\partial p}u(p, x = \infty) \tag{4}$$

$$z(p) = \frac{\partial}{\partial p}u(p, x) \tag{5}$$

Then

$$\begin{cases} z'' = \frac{\partial f}{\partial u} z + \frac{\partial f}{\partial u'} z' \\ \frac{\partial f}{\partial u} = \left(\frac{1+u^2}{(1-u^2)^2} \left[v^2 + \frac{n^2}{r^2} \right] \right) \\ \frac{\partial f}{\partial u'} = \left(\frac{-1}{r} + \frac{2uv}{1-u^2} \right) \end{cases} \tag{6}$$

We can use reduction of order to obtain the systems we must solve numerically.

$$\begin{cases} u' = v \\ v' = -\frac{1}{r}\frac{du}{dr} - \frac{u}{1-u^2}\left[\left(\frac{du}{dr}\right)^2 - \frac{n^2}{r^2} \right] - u(1-u^2) \\ z' = w \\ w = \left(\frac{1+u^2}{(1-u^2)^2} \left[v^2 + \frac{n^2}{r^2} \right] \right) z + \left(\frac{-1}{r} + \frac{2uv}{1-u^2} \right) w \end{cases} \tag{7}$$

4 Newton-Kantorovich

The first step for the Newton-Kantorovich is to find the Kantorovich equations by following these steps:

1. Find $F(u) = 0$
2. Find $F(u + hz)$
3. Find $\frac{\partial F}{\partial h}$
4. Find $\lim_{h\to 0} \frac{\partial F}{\partial h}$

1.

$$F(u) = \begin{cases} \frac{d^2u}{dr^2} + \frac{1}{r}\frac{du}{dr} + \frac{u}{1-u^2}\left[\left(\frac{du}{dr}\right)^2 - \frac{n^2}{r^2} \right] + u(1-u^2) = 0 \\ u(0) = 0 \\ u(\infty) = 1 \end{cases} \tag{8}$$

2.

$$F(u+hz) = \begin{cases} u'' + hz'' + \frac{1}{r}\left(u' + hz'\right) + \frac{u+zh}{1-(u+zh)^2}\left[(u'+hz')^2 - \frac{n^2}{r^2} \right] + (u+hz)(1-(u+hz)^2) = 0 \\ u(0) + hz(0) = 0 \\ u(\infty) + hz(\infty) = 1 \end{cases} \tag{9}$$

3. and 4. assuming we know some approximation u^n

$$\begin{cases} z'' + \frac{1}{r}z' + z(1-3u^2) + \frac{1}{1-u^2}\left[(u')^2 - \frac{n^2}{r^2} \right] = -(u^n)'' - \frac{1}{r}(u^n)' - \frac{u^n}{1-(u^n)^2}\left[((u^n)')^2 - \frac{n^2}{r^2} \right] - u^n(1-(u^n)^2) \\ z(0) = -u^n(0) \\ z(\infty) = u^n(\infty) + 1 \end{cases} \tag{10}$$

This system (10) is now linear in z so we can solve it using any numerical scheme for linear BVP's. Knowing that the standard form for a linear BVP is

$$u'' = p(r)u' + q(r)u + \xi(r) \tag{11}$$

Trivially we can find the corresponding p, q, ξ for (10)

$$\begin{cases} p = \frac{1}{r} - \frac{2uu'}{1-u^2} \\ q = -3u^2 + 1 + \frac{1}{1-u^2}\left[(u')^2 - \frac{n^2}{r^2} \right] \\ \xi = -u_j'' + f_j \end{cases}$$

We choose to solve for z using the finite difference method. First we must define a mesh.

$$h = \frac{1-0}{N} \qquad \text{Where we are free to choose a large N} \tag{12}$$

$$x_j = (j-1)h, \quad j = 1, 2, 3, ..., N+1 \tag{13}$$

$$u_j = u(x_j) \tag{14}$$

We use the following finite difference approximations for u' and u''

$$(u_j)' = \frac{u_{j+1} - u_{j-1}}{2h} \tag{15}$$

$$(u_j)'' = \frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} \tag{16}$$

$$\tag{17}$$

Then have our matrix-vector system defined as

$$\begin{bmatrix} -2-h^2q_2 & 1-\frac{hp_2}{2} & 0 & & ... & 0 \\ 1+\frac{hp_3}{2} & -2-h^2q_3 & 1-\frac{hp_3}{2} & 0 & ... & 0 \\ 0 & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & ... & 0 & & 1+\frac{hp_N}{2} & -2-h^2q_N \end{bmatrix} \begin{bmatrix} z_2 \\ z_3 \\ . \\ . \\ . \\ z_N \end{bmatrix} = \begin{bmatrix} h^2\xi_2 \\ h^2\xi_3 \\ . \\ . \\ . \\ h^2\xi_N - 1(1-\frac{hp_N}{2}) \end{bmatrix} \tag{18}$$

Which we can solve for z and simplify in matrix-vector notation to

$$\vec{z} = \vec{U}A^{-1} \tag{19}$$

We then wish to update u using the correction z

$$u^{n+1} = u^n + z \tag{20}$$

5 Implementation

Here we will discuss the implementation details of the two numerical methods in matlab. The code is listed bellow with in-line comments for explanation of the implementation.
If the reader would like to investigate implementation regarding testing and experimental data further see the github repository for this investigation.

5.1 Shooting method

```
hold off
clf
clc
clear

for n = [1,2,3]
    a = 0.1; % Boundary one
    b = 15; % infinity [1,2,3]
    ph = 4; %bisection upper bound
    pl = -4; %Bisection lower bound
    p = (ph + pl)/2; %bisection first mid point
    %p = 0.2; %Arbitrary initial shooting parameter used for NM
    tol = 1; %Set initial tolerance
    i = 1; %Counter for number of iterations
    u0 = [0,p,0,1]; %Set initial conditions for our system
    step_size = 0.2; %Set step size
    while i < 100 && tol > 1e-7
        %Integrate the system using runge-kutta method
        [r,u] = ode45(@(r,u) shoot(r,u,n), [a:step_size:b], u0,n);
        tol = abs(1-u(end,1)) %Display the tolerance
        tor = u(end,1); %Purely aesthetic
        %Bisection scheme for updating p.
        if(tor > 1)
            ph = p; %Move upper bound
            p = (pl + ph)/2; %Bisect
            u0 = [0,p,0,1]; %Update IC's with new p value
        end
        if(tor < 1) %Note that we exclude tor = 1 explicitly as if tor = 1 we are finished.
            pl = p; %Move lower bound
            p = (pl + ph)/2; %Bisect
            u0 = [0,p,0,1]; %Update IC's with new p value
        end
        %End of Bisection scheme

        %NM scheme. Note that we use bisection primarily for stability
        %p = p - ((u(end,1)-1)/(u(end,3))); %Update p according to NM scheme
        %u0 = [0,p,0,1]; %Update the IC's with new p value

        i = i + 1; %Update iteration counter
        figure(1);plot(r,u(:,1)); %Plot the updated solution curve
        hold off;
        pause(0.1);
    end
    fig2 = figure(2);
    plot(r,u(:,1)) %Plot updated solution curve for all n
    hold on;
    legend("n=1", "n=2", "n=3");
    title("Nonlinear shooting for a = 0.1, b = 15, tol = 1e-7");
    xlabel('r');
    ylabel('u');
    pause(1);
end

%The system of equations for u and z. Note z is only needed for NM scheme
function [du] = shoot(r,u, n)
    uu = u(1);
    v = u(2);
    z = u(3);%Extract the IC's
    w = u(4);
    %Update scheme needed for integrating u
    du(1,1) = v;
    du(2,1) = (-1/r)*v  - (uu/(1-uu^2))*(v^2-(n^2)/(r^2)) - uu*(1-uu^2);
    %Update scheme needed for integrating z
    du(3,1) = w;
    du(4,1) = z*( -((1 + uu^2)/((1-uu^2)^2))*(v^2 - (n^2)/(r^2)) - 1 + 3*uu^2) + w*(-1/r  - ...
        ((2*uu)*v)/(1-uu^2));
end
```

5.2 Newton-Kantorovich method

```
hold off
clf
clc
clear

for n = [1,2,3] % Loop over all n=1,2,3 and obtain numerical solutions
    a = 0; %The left BC
    b = 4; %infity (the right BC)
    N = 50; %Define number of mesh points
    h = (b-a)/N; %Define the mesh step size according to scheme
    h2 = h*h; %Simplification of code
    r = (a:h:b)'; %Set up r mesh with step size h between a and b
    u = r/b; %Initial guess is linear, other guesses are bellow
    %u = log(r + 1)/log(b+1) -a;
    %u = -exp(-(r)) + 1 ;
    %u = 1./(1+exp(-r + b/2));
    rhs=[u(1);u(1:N-1)-2*u(2:N)+u(3:N+1) + h2.*f(r,u,h, N, n);u(N+1)-1]; %Set initial RHS based on ...
        guess
    tol = 1; %Initialize tolerance value
    count = 1; %Initialize iteration counter
    while count < 100 && tol > 1e-7
        dd = [0; -2 - h2.*q(r,u,h,N, n) ; 0];
        upper = diag([0;1 - (h*p(r,u,h,N))./2],1);
        lower = diag([1 + (h*p(r,u,h,N))./2;0],-1);
        J = diag(dd,0) + upper + lower;  %Set three diagonal
        J(1,1)=1; %Set BC 1
        J(N+1,N+1)=1; %Set BC 2
        z = -J\rhs; %Solve for correction
        u = u + z; %Update u with new correction
        rhs=[u(1);(u(1:N-1)-2*u(2:N)+u(3:N+1)) + h2.*f(r,u,h,N,n);u(N+1)-1]; %Update RHS values
        tol = norm(rhs); %Display tolerance
        fig1 = figure(1);
        plot(r,u); %Plot updated curve of u
        hold off;
        pause(0.1);
        count = count+1; %Increment iteration counter
    end
    %Plot final solution u for each n
    fig2 = figure(2);
    plot(r,u);
    hold on;
    legend("n=1", "n=2", "n=3");
    title("NK for b = 7, tol = 1e-7, logarithm guess, N = 50");
    xlabel('r');
    ylabel('u');
    pause(1);
end %End of n for loop

%This is f in y'' = f
function [fu] = f(r, u, h, N, n)
    uBefore = u(1:N-1); %This is u_{j-1}
    uAfter = u(3:N+1); %This is u_{j+1}
    u = u(2:N); %This is u_{j}
    %Evaluate and return f(2:N). Note the FD approximations of u'
    fu = (1./r(2:N)).*((uAfter - uBefore)/(2*h)) + ((u)./(1-u.^2)).*(((uAfter - uBefore)/(2*h)).^2 ...
        - n^2./(r(2:N).^2)) + u.*(1-u.^2);
end

%This is p(r) in a linear Kantorovich equations
function [p] = p(r,u,h, N)
    uBefore = u(1:N-1); %This is u_{j-1}
    uAfter = u(3:N+1); %This is u_{j+1}
    uu = u(2:N); %This is u_{j}
    r = r(2:N); %Find r excluding the end points
    %Evaluate and return p(2:N)
    p = -1*(1./r + (2*uu.*((uAfter-uBefore)/(2*h))))./(1-uu.^2));
end

%This is q(r) in the linear Kantorovich equations
function [q] =  q(r,u, h , N, n)
    r = r(2:N); %Find r excluding the end points
    uBefore = u(1:N-1);  %This is u_{j-1}
    uAfter = u(3:N+1); %This is u_{j+1}
    uu = u(2:N); %This is u_{j}
    %Evaluate and return q(2:N)
    q= -1*(-3*uu.^2 + 1 + 1./((1-uu.^2)).*(((uAfter-uBefore)/(2*h)).^2 - (n^2)./(r.^2) ));
end
```

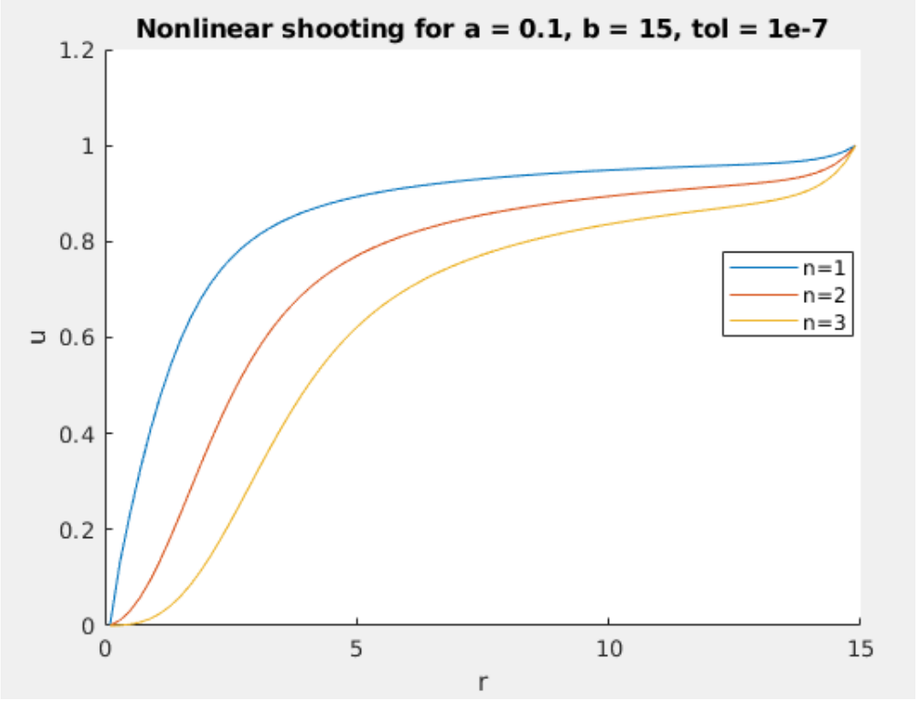
6 Results and discussion

Here we will discuss how the two methods performed and what we learned.
The testing strategy for the hyper-parameters was as follows: We must have base stable parameters that we can fix while we vary the others. Let these stable parameters be

$$
\begin{aligned}
&a = 0.1 \\
&b = 5 \\
&N = 50 \\
&stepsize = 0.2 \\
&u^0 = \frac{r}{b} \\
&p0 = 0.2 \quad \text{For Newton's method in Nonlinear shooting} \\
&ph = 4 \quad \text{For bisection in Nonlinear shooting} \\
&pl = -4 \quad \text{For bisection in Nonlinear shooting}
\end{aligned}
$$

Each test will be run until either 100 iterations is reached or the tolerance less than 10^{-7}. If the tolerance is less than 10^{-7} we consider the solution stable otherwise unstable.
The parameters that we will analyse for stability are a, b, N, step size, u^0 (the initial guess for finite differences) and bisection vs Newton's method for shooting parameter update schemes.
A note on interpreting results: For brevity convergence to a stable solution has been represented by a 1 and divergence as defined above is represented by a 0.

6.1 Shooting method



We can see a clear pattern forming as n grows. If we continue to solve for $n = 4, 5, 6$ we see a similar trend as u goes to 1 increasingly slowly. Another interesting insight is that the curves sharply rize to 1 as they near the b value. This occurs because of the numerical schemes themselves forcing the boundary condition in finite r, if we were to let r go to infinity we would not see this behavior.

6.1.1 Varying a

a	n=1	n=2	n=3
0	0	0	0
0.001	1	1	0
0.002	1	1	0
0.003	1	1	0
0.004	1	1	0
0.005	1	1	0
0.006	1	1	0
0.007	1	1	0
0.008	1	1	0
0.009	1	1	1
0.01	1	1	1

Table 1: This table shows the convergence of the nonlinear shooting method with varying a for $n = 1, 2, 3$

We can see that the shooting method is unstable at $a = 0$ as this implies $r = 0$ which creates undefined results with the $\frac{1}{r}$ terms (7), producing invalid results. This is why in future testing $a = 0.1$ is used to ensure stability here. Future investigation could look into using a power series to capture the behavior for small or zero r.
Additionally we can see that for non-zero a solutions, $n = 1$ and $n = 2$ are stable, however, $n = 3$ begins converging at larger a, namely $a = 0.009$. This is likely due to the solutions slow growth to 1 causing poor stability as we force the solution towards 1 which can be seen in figure 1 as mentioned before.

6.1.2 Varying b

b	n=1	n=2	n=3
4	1	1	1
6	1	1	1
8	1	1	1
10	1	1	1
12	1	1	1
14	1	1	1
16	0	1	1
18	0	0	0
20	0	0	0
22	0	0	0
24	0	0	0

Table 2: This table shows the convergence of the nonlinear shooting method with varying b for $n = 1, 2, 3$

While varying b (which is a finite representation for infinity) we find that stability is lost for a choice of $b > 14$ in the case of $n = 1$ and $b > 16$ for $n = 2, 3$.
The instability of smaller n in this case can be attributed to the slower growth of the solution as we choose larger n. Looking at system (7) there are multiple terms containing $\frac{1}{1-u^2}$ which causes division by a small number the closer u is to 1. Solutions for larger n exhibit better stability properties as they have less values that are close to one than solutions with small n.

6.1.3 Varying step size

Testing the step size with the chosen fixed values does not yield any interesting results as solutions are stable for all rational step sizes greater than zero.
However the step size does play a role in stability as a small step size can cause many more divisions by small r if we choose $a = 0$ or some other very small a. The a that we fix here is sufficiently large to negate this impact for all values of n.

6.1.4 Varying bisection vs Newton

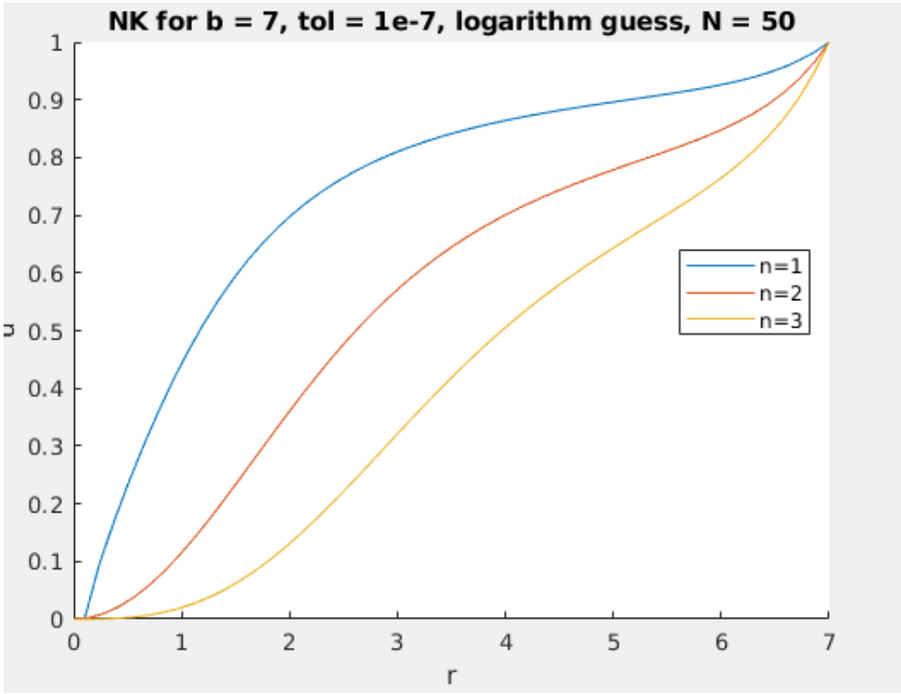
Comparing the stability of these two schemes for updating the shooting parameter reveals that Newton's method performs poorly and only suitably updates the shooting parameter under very specific conditions namely, large a, small b and a very good initial guess for p. This behavior is due to two main issues with the method:

1. Solving the system
   System (7) contains many "dangerous" as $u \to 1$ or $r \to 0$ which ties the updating of the shooting parameter to the numerical solution. If we choose a poor initial guess for the shooting parameter the produced update will be poor since the solution is unstable.

2. Division by a large number
   Following from the previous point we know that when the system is unstable it produces large values for z which causes the updates to the shooting parameter to be very small as we divide by z in the Newton's method formula.

Due to these issues bisection is clearly the choice for stability, however, Newton's method may converge to a correct shooting value in less time.

6.2 Newton-Kantorovich method



6.2.1 Varying a

a	n=1	n=2	n=3
0.00	0	1	1
0.01	0	1	1
0.02	0	1	1
0.03	0	1	1
0.04	0	1	1
0.05	0	1	1
0.06	1	1	1
0.07	0	1	1
0.08	0	0	0
0.09	0	0	0
0.1	0	0	0

Table 3: This table shows the convergence of the NK method with varying a for $n = 1, 2, 3$

These results have some interesting characteristics that are a large contrast to those of the shooting method. We see that for $n = 2, 3$ that small and zero values for a produce stable solutions and large values for a, namely $a > 0.07$ induces instability. This is likely due to the nature of the method relying less heavily on terms containing $\frac{1}{r}$ as well as not evaluating the dangerous functions at the boundaries as those values should be fixed.
The case for $n = 1$ is strange as we only see a stable solution produced for $a = 0.06$, it is not clear why this is however, it is likely due to some complex interactions between a variety of different elements. More analysis is needed here to find the exact reason, it may be an error in the implementation of the method.

6.2.2 Varying b

b	n=1	n=2	n=3
4	1	1	1
6	0	1	1
8	0	0	1
10	0	0	1
12	0	0	0
14	0	0	0
16	0	0	0
18	0	0	0
20	0	0	0
22	0	0	0
24	0	0	0

Table 4: This table shows the convergence of the NK method with varying b for $n = 1, 2, 3$

As with the shooting method we see that larger values of n have a larger range of stability in b. This is likely due once again the the slower growth to 1. There is little more insight to be gained here, however, one should note that the stability of this method for the same fixed parameters is worse than that of the nonlinear shooting method as b grows.

6.2.3 Varying N

N	n=1	n=2	n=3
10	0	1	1
110	0	1	1
210	0	1	1
310	0	1	1
410	0	1	1
510	0	1	1
610	0	1	1
710	0	1	1
810	0	1	1
910	1	1	1
1010	0	1	1

Table 5: This table shows the convergence of the NK method with varying N for $n = 1, 2, 3$

The results here are quite surprising, with solutions for $n = 1$ only being stable for $N = 910$ while solutions of larger n are stable for all N in the testing. This is likely due to poor choice of testing parameters for $n = 1$ as they inherently cause instability, see 6.2.3 for similar results.
Collectively we can infer that N has a similar effect to that of the step size in the shooting method as a low number of mesh points can give rise to stability by ignoring problematic areas while a large number of mesh points can also give rise to stability where critical detail is needed to form a stable solution. This means that choosing a good value for N is problem dependent. For our vortex solutions we know that when using the Newton-Kantorovich method, there are no points that cause instability hence a low N is not required, we can see in the varying of N that when the choice of other parameters is poor, a good N value can still produce stable solutions.

6.2.4 Varying initial guess

These results are valuable as they give us insight on the effect of our initial guess on the stability of the numerical solutions for u.
Each table shows the effect of different functions chosen for each a value of n. The functions chosen are:

$$\text{linear: } u = \frac{r}{b}$$
$$\text{sigmoid: } u = \frac{1}{1 + e^{-r+b/2}}$$
$$\text{exponential: } u = -e^{-r} + 1$$
$$\text{logarithmic: } u = \frac{log(r+1)}{log(b+1)}$$

Note that all the functions satisfy the boundary conditions except the sigmoid which does not.

b	linear	sigmoid	exponential	logarithmic
4	1	1	1	1
6	0	0	0	0
8	0	0	0	0
10	0	0	0	0
12	0	0	0	0
14	0	0	0	0
16	0	0	0	0

Table 6: This table shows the convergence of the NK method with varying N and initial guess u^0 for $n = 1$

b	linear	sigmoid	exponential	logarithmic
4	1	1	1	1
6	1	1	1	0
8	0	0	1	0
10	0	0	0	0
12	0	0	0	0
14	0	0	0	0
16	0	0	0	0

Table 7: This table shows the convergence of the NK method with varying N and initial guess u^0 for $n = 2$

b	linear	sigmoid	exponential	logarithmic
4	1	1	1	1
6	1	1	1	0
8	1	1	1	0
10	1	0	1	0
12	0	0	0	0
14	0	0	0	0
16	0	0	0	0

Table 8: This table shows the convergence of the NK method with varying N and initial guess u^0 for $n = 3$

We can see from tables 6,7 and 8 that for $n = 2, 3$ the exponential function exhibits the best stability properties as b varies. Once again $n = 1$ shows poor stability since the other parameters are poorly chosen to satisfy its stability needs. The exponential guess likely performs the best as it matches the solution most closely which in turn requires less correction from the numerical method, giving rise to better stability properties.
Interestingly the linear function performs well for all n and is likely a good start for any numerical method where the solution curve is not already known.
The sigmiod performs well despite not satisfying the boundary conditions, this is due to corrections being applied at the boundaries. This result is good as it indicates that when choosing an initial guess we are free to choose any function that best matches our goal, no restriction for boundary conditions.
The logarithmic function performs the worst of all the guesses which is likely due to its shape. But it is not clear why this would be an issue as the curve matches the solution curve well, further investigation here would likely yield interesting results.

7 Conclusion and final remarks

The two numerical methods exhibit some shared stability properties but with some differences. Nonlinear shooting struggles when there are undefined values at the curve boundaries where as Newton-Kantorovich with Finite differences does not.
Both methods exhibit similar behaviour from step size and mesh density, both of which often require tuning for the specific problem.
Both methods exhibit instability as b grows however, the nonlinear shooting method has far better stability properties in this regard for the conditions that were tested.
The initial guess for Newton-Kantorovich with finite differences had a large impact on the stability of solutions. However in most cases a simple linear guess will be sufficient and can be tuned later on.
Nonlinear shooting with the Newton's method update scheme exhibited poor stability due to issues arising from division of small and large numbers, however, this issue will not be the case for all problems. The bisection method is a good starting point for its simplicity and lack of reliance on the stability of the solution to update.
Both numerical methods provide unique advantages and disadvantages with the shooting method being easy to implement but providing poor stability at the boundaries and Newton-Kantorovich with finite differences exhibiting poorer stability over all but solving this issue. Additionally Newton-Kantorovich gives more flexibility as we can use any linear numerical method to solve for the update parameter. When solving a Boundary Value Problem one should consider these parameters and make a choice based on them.