Vel-Norm Problem Followup

Sidharth Baskaran

July 3, 2021

1 Simplifying expression for $w_p(\phi)$

The sine function is a phase shift of the cosine function, so

$$\sin(2k) = \sin(2\phi + \pi/2) = \cos(2\phi). \tag{1}$$

Further,

$$2\sin^2(2k) = 1 - \cos(2k). \tag{2}$$

Because $\cos(2k) = -\sin(2\phi)$ by a similar argument,

$$2\sin^2(2k) = 1 + \sin(2\phi). \tag{3}$$

Finally

$$w_p(\phi) = \underbrace{\frac{1}{\sin(2\phi+\pi/2)}}_{\cos(2\phi)}[1 \; \underbrace{-\frac{a_T r_0^2}{\mu}}_{-\frac{4}{\beta^2(3\pi+8)}} \; (3\phi+2)] + \frac{2}{\beta^2(3\pi+8)}\underbrace{(2\sin^2(\phi+\pi/4)+3)}_{\sin(2\phi)+4} \tag{4}$$

$$= \frac{1}{\cos(2\phi)}\left[1 - \frac{4}{\beta^2(3\pi+8)}(3\phi+2)\right] + \frac{2}{\beta^2(3\pi+8)}(\sin(2\phi)+4) \tag{5}$$

2 Numerical evaluation of $y(\phi_m) = 0$

To evaluate $y_p(\phi) = 0$, we first define `phi = (1:1:1000)*pi/4/1000`. Using this vector to formulate `y_p(Beta)`, where `Beta` is a variable constant, we have a vector of length 1000 representing $y_p(\phi)$. Note that

$$y_p(\phi) = \frac{\sqrt{\varphi(\phi)}}{\beta^2(3\pi+8)\sin(2\phi+pi/2)}\sqrt{\frac{\mu}{r_0}}\cot(\phi+\pi/4) \tag{6}$$

This implies that $\sin(2\phi + \pi/2) \neq 0$, and further that y_p is only defined for $\varphi(\phi) \geq 0$. Thus, the vector `y_p(Beta)` must be shortened to reflect this and exclude complex numbers. From calculations, we find that the real representation of y_p is `y_p(Beta)(1:548)` using $a_T = 0.2, r_0 = 1, \mu = 1$ with $\beta \approx 1.07$. To find the real portion of this vector, we have called a function `realBreakpoint(vector)` in Figure 1. Sorting this vector and determining the corresponding ϕ_m solves the problem, where we expect `y_minValue = 0` and `phi_m` to be the corresponding value of ϕ:

```
[y_values index_vector] = sort(y_p(Beta)(1:548));
y_minValue = y_values(1);
phi_m = index_vector(1);
```

A similar approach can be employed for a changing parameter β. Since $\delta = \frac{1}{\beta^2} \in (0.001, 1)$, $\beta = \sqrt{\frac{1}{\delta}} \in (1, \sqrt{1000})$. In Octave/MATLAB, we can implement it as

```
delta = (1:1:1000)/1000;
Beta_vec = sqrt(1./delta);
```

Mathematically, we say that

$$\vec{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} \text{ and } \vec{\delta} = \begin{bmatrix} 1/\beta_1^2 \\ \vdots \\ 1/\beta_n^2 \end{bmatrix} \tag{7}$$

where we choose $n = 1000$. Thus, by iterating through the values of `Beta_vec`, we can generate a corresponding vector `phi_m1` to represent how ϕ_m changes with respect to it:

```
phi_m1 = zeros(1,n);
w_1 = zeros(1,n);
w_min = zeros(1,n);
y_0 = zeros(1,n);

for i = 1:n
    y_real = y_p(Beta_vec(i))(1:realBreakpoint(y_p(Beta_vec(i))));
    [yvals idx] = sort(y_real);
    if (length(idx) <= 1)
        phi_m1(i) = max(phi_m1);
        w_1(i) = max(w_1);
        y_0(i) = min(y_0);
    else
        w_1(i) = w_p(Beta_vec(i))(idx(2));
        phi_m1(i) = phi(idx(2));
        y_0(i) = yvals(2);
    endif
end
```

Figure 1: Numerically finding $\vec{\phi_m}$ over $\vec{\delta}$

In Figure 1 above, the vector `y_0` is updated with the value $y_p(\phi_{m,i})$ in each iteration for each value of β, and is expected to be 0. The vector `w_1` is updated with $w_p(\phi_{m,i})$, which is expected to be unity. As was done with `y_p(Beta)`, we have defined a vector `w_p(Beta)` to represent $w_p(\phi)$ where `Beta` is a constant that can be varied. Thus, we should expect $\vec{w}_p(\vec{\phi}_m) = \vec{1}$, $\vec{y}_p(\vec{\phi}_m) = \vec{0}$, and the existence of $\vec{\phi}_m = \begin{bmatrix} \phi_{m,1} \\ \vdots \\ \phi_{m,n} \end{bmatrix}$ at the end of the loop. The conditional checks are used to deal with numerical error which results in `y_p(Beta_vec(i))` being full of complex values. The following plot of ϕ_m vs. δ captures these results with minor numerical error.

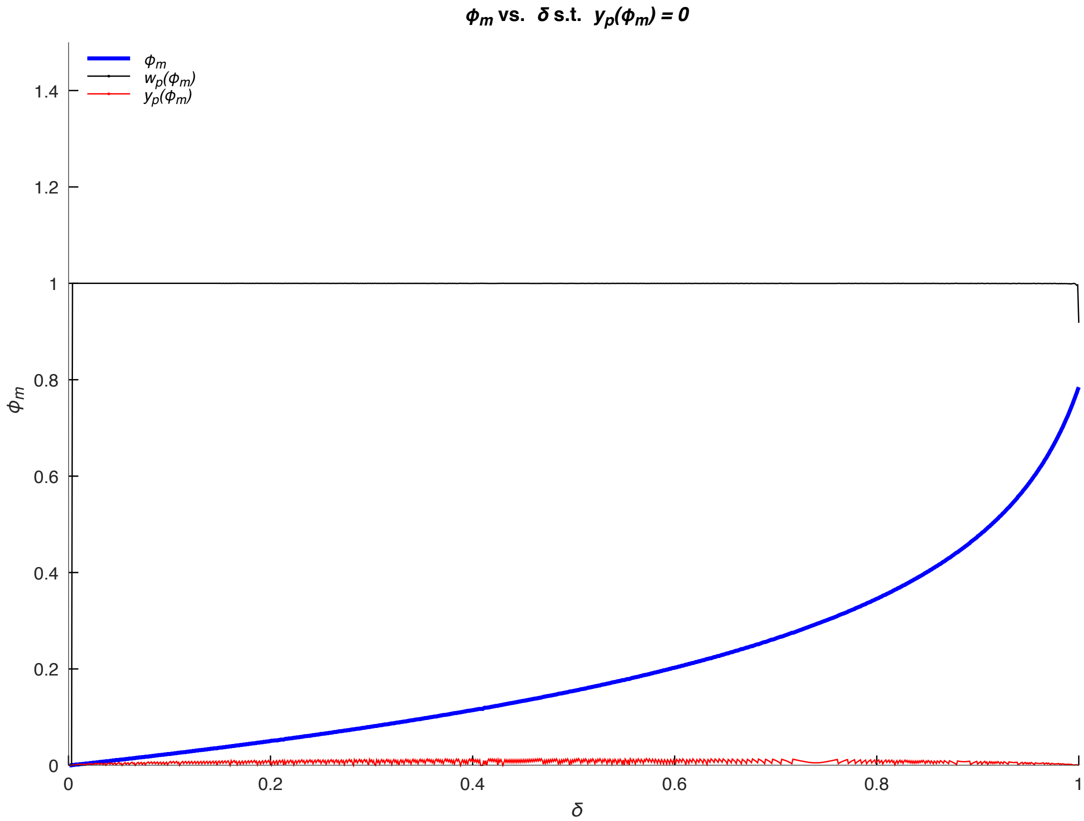

Figure 2: A plot of thes results

We see that $y_p(\phi_m) \approx 0,\ w_p(\phi_m) \approx 1\ \forall\ \phi_m$, which is consistent with the expectations.

3 Numerical minimization of $w_p(\phi)$

We use a similar approach as before to minimize $w_p(\phi)$ numerically. In Figure 1, `w_1(i)` was updated to reflect the value of $w_p(\phi_{m,i})$ where $\phi = \phi_{m,i}$ minimized $y_p(\phi)$ for corresponding values of β_i, δ_i from Eq. 7. The following code defines a vector `phi_m2` of length $n = 1000$ and populates it with the value of $\phi_{m,i}$ that minimizes $w_p(\phi)$ for β_i. The minimum value is stored in another vector `w_min`.

```
phi_m2 = zeros(1,n);
w_min = zeros(1,n);
for i = 1:n
    [w_vals idx] = sort(w_p(Beta_vec(i)));
    phi_m2(i) = phi(idx(1));
    w_min(i) = w_vals(1);
end
```

Figure 3: Iteratively finding $\vec{\phi}_m$ for $\min(w_p(\phi))$ for various β

Figure 4 below, showing ϕ_m (representing `phi_m2`) vs. δ, summarizes these results.

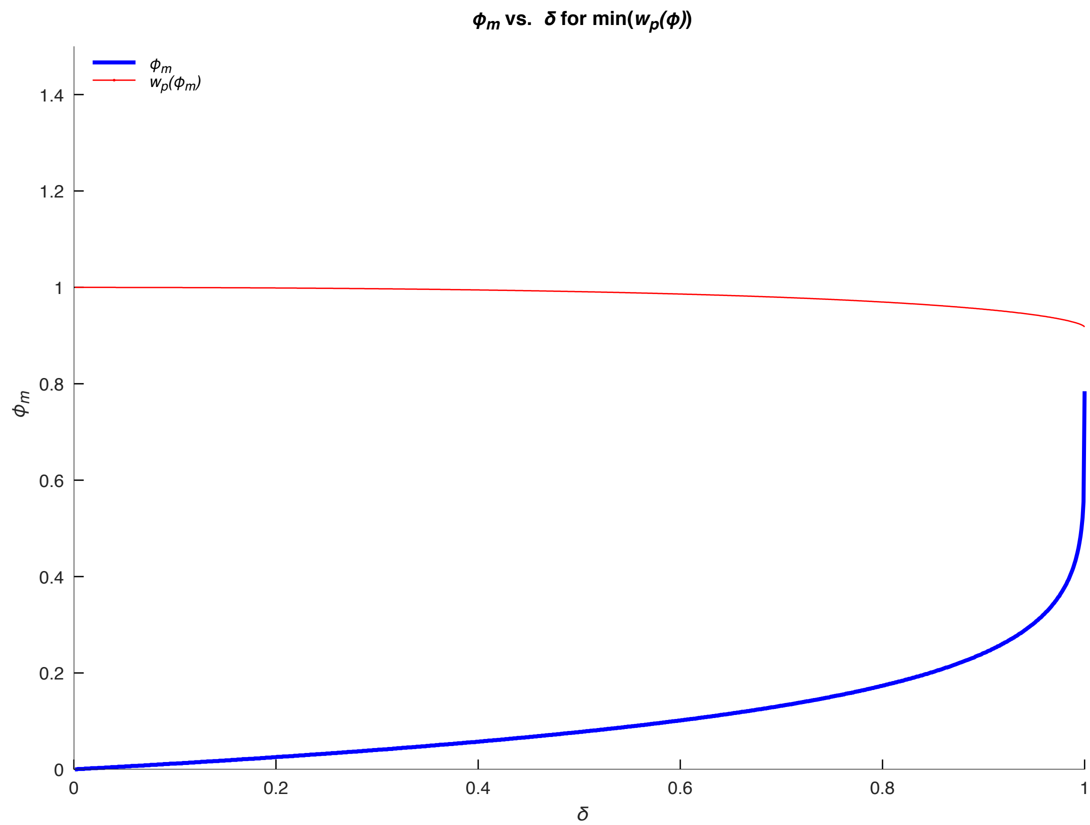

Figure 4: Values of ϕ_m that minimize $w_p(\phi)$ for various β