**Crowdmark Due Date:** Monday 4 February, 2019 at 11:55

- Please follow the *Guidelines for Computing Assignments* found on the Canvas FAQ.
- Submit a one-page report via Crowdmark, grading scheme is also found on the FAQ.
- Acknowledge any and all collaborations and assistance from classmates/TAs/instructor.

---

# Root Finding 2D Contour

One dimensional root finding has many applications, and a common situation is called *continuation*, where a sequence of (related) roots are to be solved for. An elementary example of a continuation is the numerical construction of a contour (or level curve) of a 2D function. A contour plot is a 2D representation of a function of two variables, $f(x, y)$ that plots all curves of the form $f(x, y) = c$, for various constants $c$. Figure 1 shows a colour plot of the $HI(x, y)$ function, including the zero contour ($HI(x, y) = 0$ is the white dashed curve) . While it may seem that finding solutions to $HI(x, y) = 0$ is a 2D problem, there is a way to apply the 1D methods presented in class.

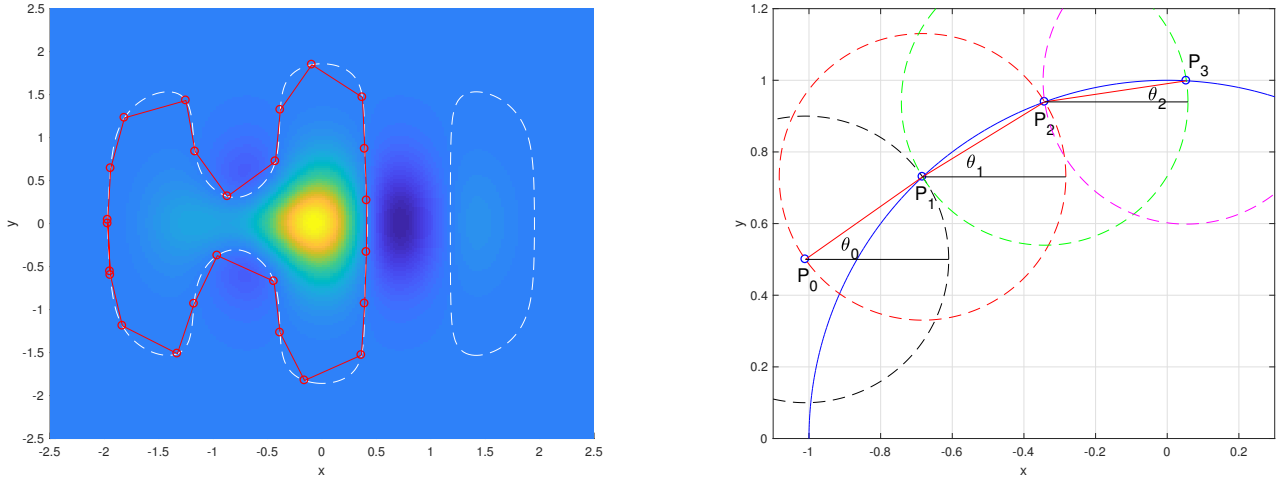


Figure 1: and Figure 2

Suppose we have found the points $\{(x_1, y_1),\ \ldots\ (x_n, y_n)\}$ on our zero contour. To find our next point, $(x_{n+1}, y_{n+1})$, we will root search only on a circle, centred at $P_n = (x_n, y_n)$, with a radius $\Delta s$. The points on this circle are a function of angle $\theta_n$, and can write $P_{n+1} = (x_{n+1}, y_{n+1})$ as

$$x_{n+1}(\theta_n) = x_n + \Delta s\, \cos(\theta_n) \quad ; \quad y_{n+1}(\theta_n) = y_n + \Delta s\, \sin(\theta_n)\ . \tag{1}$$

We can find $\theta_n$ by solving the equation $HI(x_{n+1}(\theta_n), y_{n+1}(\theta_n)) = 0$, which is now a 1D root-finding problem in $\theta_n$. The radius $\Delta s$ is a numerical parameter for this procedure, and as you will discover, controls the graphical quality of the contour produced. Figure 1 shows the zero contour of $HI(x, y)$ computed by this method (in red). The computation uses only $\Delta s = 0.6$ and finds 24 points on the contour, so the computed contour does not appear smooth.

A cartoon for our procedure is Figure 2 where the point $P_3$ on the contour (thick blue curve) has just been located at the angle $\theta_2$. Note that there is actually a second point on the (green) circle around $P_2$, but it is the previous point $P_1$! Hence, special care needs to be taken that our root-finding algorithm doesn't

reverse itself. We can now repeat the procedure on the (magenta) circle around $P_3$ to iterate the search procedure to find $P_4$.

But how can we find the first point? We can use the same angle solve procedure provided that we begin from a point $P_0$ that only needs to be within a distance $\Delta s$ from our contour. Note in the cartoon, $P_0$ is not on the contour, but the general procedure still works. For this assignment, you can manually set $P_0$ as an 'eyeballed' point that sits near your contour.

**The goal of this assignment is to compare the bisection and secant methods, in terms of efficiency and robustness, when computing the zero contour of 2D functions**. Begin by downloading *CA3_demo.m*, a code that does the root-finding using Matlab's *fzero* command — you will need to replace *fzero* calls inside the contour finding loop. Make sure you download the most recent versions of *BiSsqrt.m* and *SMsqrt.m* from Canvas to help you modify *CA3_demo.m*. We will benchmark our code on the $HI(x, y)$ function.

Both the bisection and secant methods need an initial interval — for this you will need to think about the geometry. It turns out that the "right" interval for finding the angle $\theta_n$ is $[\theta_{n-1} - \delta, \theta_{n-1} + \delta]$, where $\delta$ gives $HI(x(\theta_n), y(\theta_n))$ opposite signs at the endpoints. You should start by setting $\delta = \pi/2$ for both methods, but you will be required to experiment with this parameter. Consider all of the following when completing your report:

- Start by reproducing Figure 1 using your bisection code, with $\Delta s = 0.6$ and $\delta = 3\pi/4$.
- If you try to redo the above with secant method, the method will fail for the same choice of $\Delta s$ (try it and see). Instead, set $\Delta s = 0.1$ and $\delta = \pi/2$ to trace the contour using the secant method.
- Test both root finders for a variety of $\Delta s$ values smaller than $0.1$. What value of $\Delta s$ is needed for a visually smooth curve, say for a plot as printed on a full letter-size page?
- Using half the $\Delta s$ value you found above, experiment next by changing $\delta$. Your discussion should report on any $\delta$ values that produce convergence failures, as well as how computational cost of tracing the contour changes with $\delta$. Cost should be considered as the average number of function evaluations required per point traced on the contour.
- What have you learned about the trade-offs between using bisection or the secant method for tracing contours?

Your report should include a plot of the $HI(x, y)$ function, with your computed zero contour traced in red. Make sure you include which root finding method you used to create this plot, as well as the values of $\Delta s$ and $\delta$.

**Optional:** If you wish, instead of tracing contours of the $HI$ function, you may instead trace contours of the cellular function:

$$f(x, y) = \sin(\pi x)\sin(\pi y) + a_1 \cos\left(\frac{\pi(\pi x - y)}{2}\right) + a_2 \cos\left(\frac{\pi(x - \pi y)}{2}\right)$$

where you should choose your own personal $a_1, a_2$ coefficients. The file *CA3_newfunc.m* produces contour plots of these functions, with the zero contours plotted as white dashes. You may use a function of this form to complete the $\delta$ testing of each method; if you do this, then the plot you include with your report should be of $f(x, y)$ instead of $HI(x, y)$.