Math 128a - Programming Project 1 - ANSWERS

The main program is called **ZeroCurve.m**, and it calls three other functions:

- **Dist.m** which just computes Euclidean distances between points,
- **NextZero.m** which, given a point on the curve $f(x, y) = 0$, attempts to find the next point on the curve using Newton's method, and
- **NextZeroC.m** which is a controlled version of NextZero.m, using the step-control algorithm described in the homework.

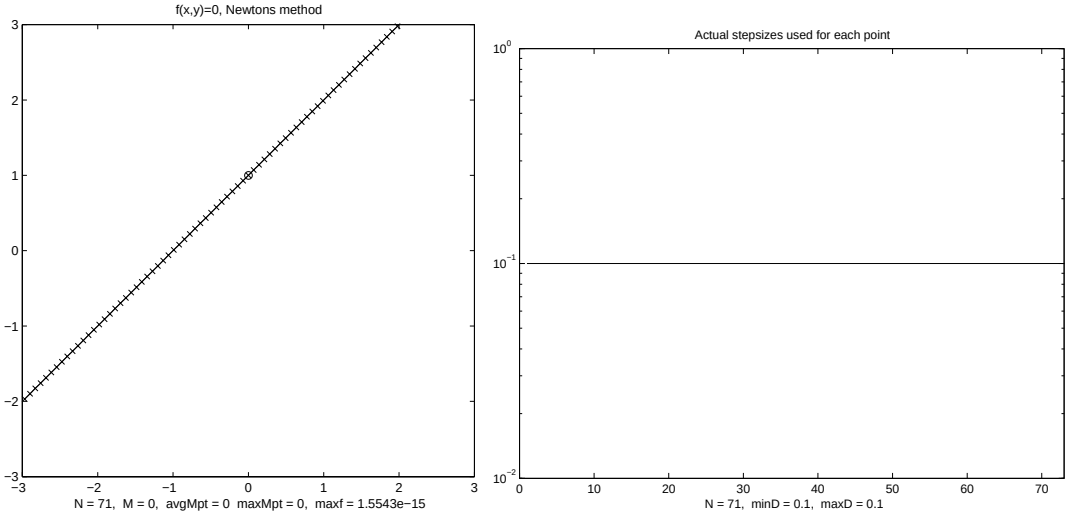
Non-obvious aspects of the algorithm:

- **NextZero.m** The idea here is to compute a vector V which is the gradient of f scaled to have length 1. Then we move a certain given distance from (x_1, y_1) in a direction perpendicular to V, and from that point we move in a direction parallel to V looking for the next zero of f. The distance is the variable *stepD*, which *can be negative*, allowing us to reverse direction in which we search for the next zero.
- **Stopping criteria in ZeroCurve.m** The trickiest stopping criterion is the closed curve criterion. To make sure that we don't stop after one iteration, we require at least 50 points to pass since the last time we were at (x_1, y_1) before stopping if we are close enough to (x_1, y_1).

Cost analysis: Much of the time is spent on checking various stopping criteria, especially distance computations. Of course, for each different function we want to plot, the complexity involved in evaluating the function and its derivatives is different. I have put in some checks for singularities (both partial derivatives 0), but if one misses the exact singularity slightly, the computer can waste a huge amount of time sitting near that singularity getting nowhere. For example, my algorithm doesn't recognize the singularity in the figure 8 but rather bounces back and forth between two points around it.

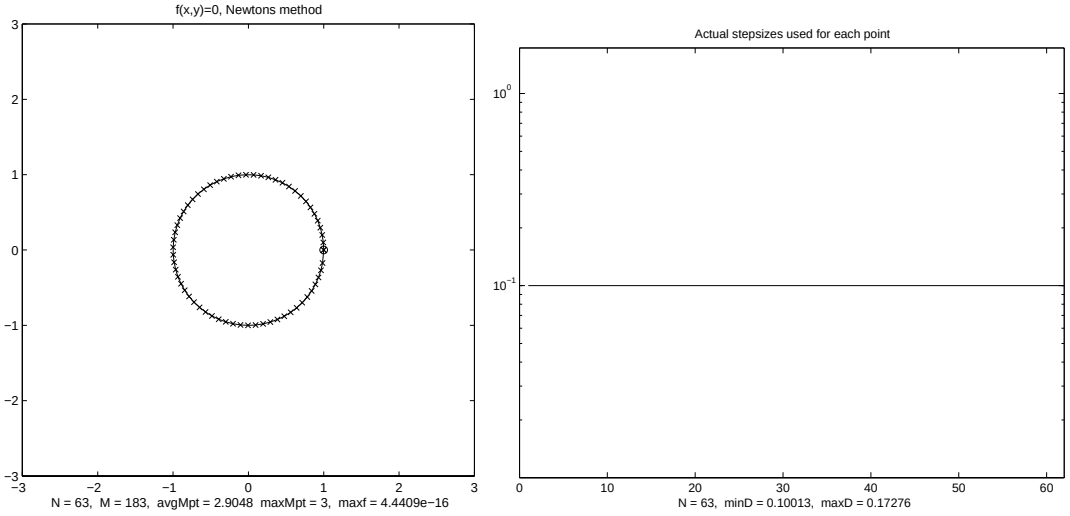
Test cases: The test cases in the assignments are written as functions in the followings files: lin.m (straight lin), circle.m (circle), cubic.m (cubic curve), rotsin.m (rotated sine curve), compsin.m (compressed sine curve), point.m (single point), fig8.m (figure 8), and cusp.m (cubic cusp). Note that the compressed sine curve and the figure 8 *should* cause trouble. The defining function for the compressed sine curve is not even defined when $x = 0$, so we run into troubles there. The figure 8 has a singularity where it crosses itself; my program missed the singularity and ended up producing points that bounced back and forth on either side of the singularity. The cusp could conceivably cause trouble if you landed right at the cusp, but I was always lucky and missed it and the iterations "made it around the corner".

1. Straight Line. The correct curve is a straight with slope 1 and passing through (0,1). The same correct picture is gotten with stepcontrol = 0 or stepcontrol = 1. In both cases stepD remained equal to .1, and no Newton steps were needed, since a straightline approximation to the curve yielded no error.



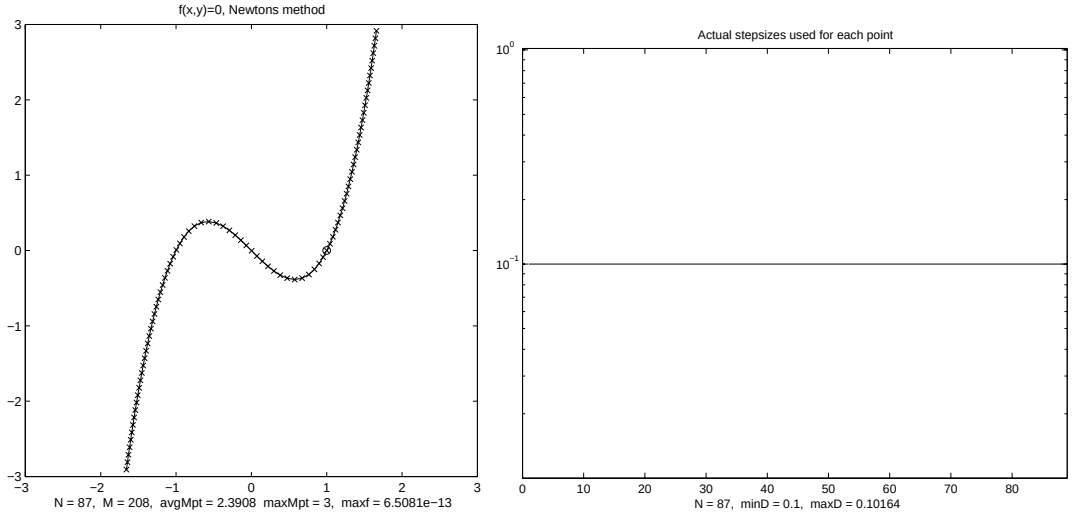
Curve	stepcontrol	N	M	avgMpt	maxMpt	maxf	maxD	minD
line	0	71	0	0	0	1.5e-15	.1	.1
line	1	71	0	0	0	1.5e-15	.1	.1

2. Circle. The correct curve is a circle with radius 1 and center at the origin. The same correct picture is gotten with stepcontrol = 0 or stepcontrol = 1. In both cases stepD remained equal to .1. Nearly every point required 3 Newton steps.



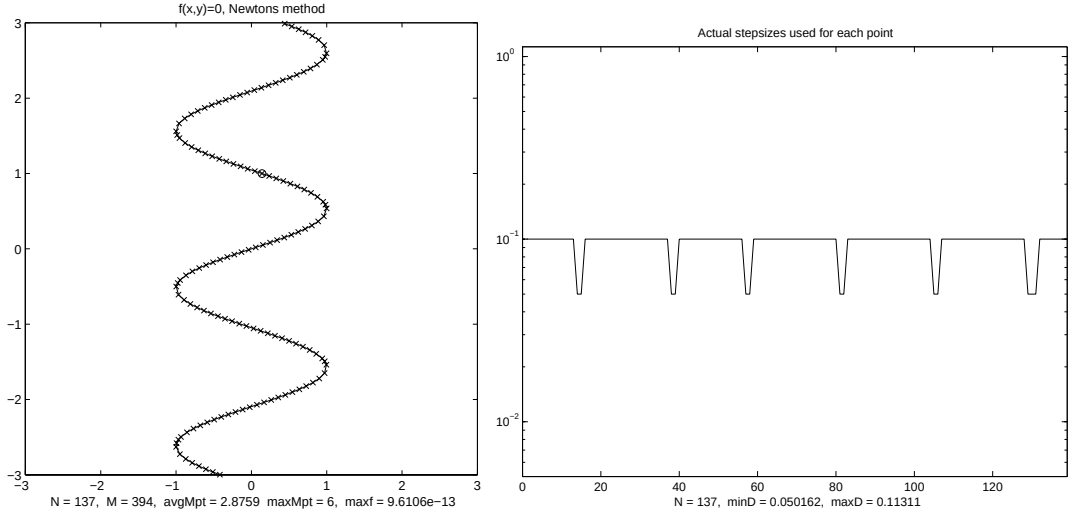
Curve	stepcontrol	N	M	avgMpt	maxMpt	maxf	maxD	minD
circle	0	63	183	2.90	3	4.4e-16	.173	.100
circle	1	63	183	2.90	3	4.4e-16	.173	.100

3. Cubic. The correct curve is a cubic and was plotted correctly with with stepcontrol = 0 or stepcontrol = 1. In both cases stepD remained equal to .1. Most points required 3 Newton steps.



Curve	stepcontrol	N	M	avgMpt	maxMpt	maxf	maxD	minD
cubic	0	87	208	2.39	3	6.5e-13	.102	.100
cubic	1	87	208	2.39	3	6.5e-13	.102	.100

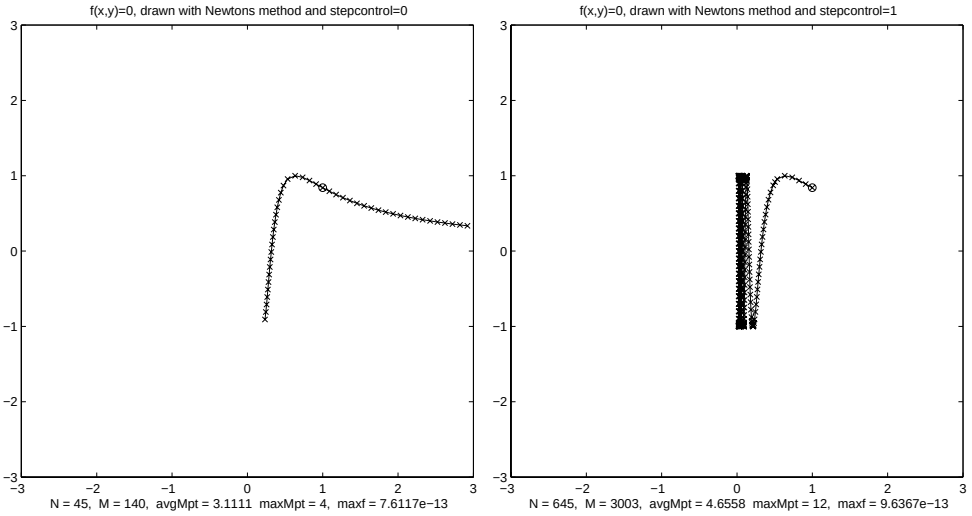
4. Rotated Sine Curve. The correct curve is a sine curve rotated 90 degrees. The same qualitative picture was computed with stepcontrol = 0 or 1, but with stepcontrol=1, the step size dropped by a factor of 2 near where the curve was vertical and made its "sharpest turns". The program did more work with stepcontrol=1 in order to make sure it took small steps where the curve turned.



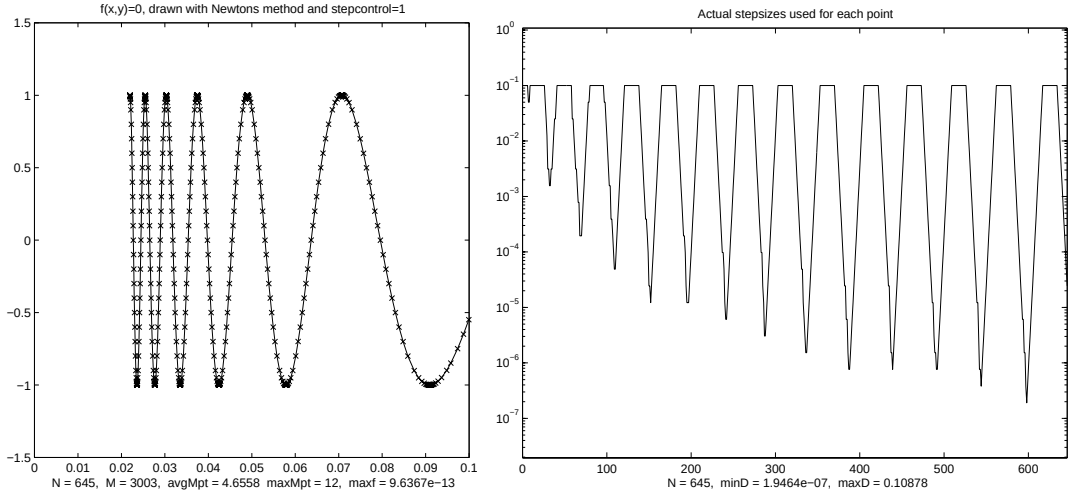
Curve	stepcontrol	N	M	avgMpt	maxMpt	maxf	maxD	minD
rotsin	0	130	340	2.62	4	9.1e-13	.113	.100
rotsin	1	137	394	2.87	6	9.6e-13	.113	.050

5. Compressed Sine Curve. The correct curve is a "sine" curve oscillating infinitely often between ± 1 as $x \to 0$, so the curve turns very sharply for small x. Now stepcontrol = 0 and 1 differ greatly. With stepcontrol=0, the figure stops near the first sharp turn, but with stepcontrol=1, the step size is made smaller and smaller near "turning points". The program stopped with stepcontrol=1 because the maximum number of Newton iterations was exceeded (so it did not go back to plot the curve going in the other direction from the starting point). The program stopped with stepcontrol=0 because the next Newton step would have taken it outside the plotting window (the large step size gave Newton a bad starting point).

   The left figure below is with stepcontrol=0, and the right is with stepcontrol=1.



   Both figures below are with stepcontrol=1. The left figure shows the curve blown up near the origin, to see how many sharp turns it took. The right figure shows that the step size was very small near where the curve turned, and got large again when it was straighter. This can be seen on the left figure by noting how close together the x's are.

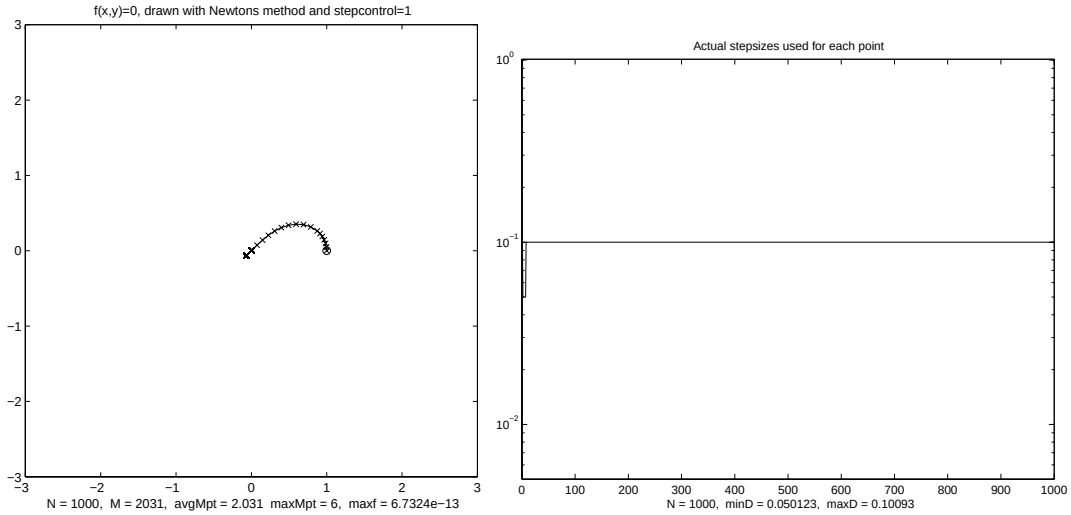


Curve	stepcontrol	N	M	avgMpt	maxMpt	maxf	maxD	minD
compsin	0	45	140	3.11	4	7.6e-13	.109	.100
compsin	1	645	3003	2.87	6	9.6e-13	.113	.050

6. Single Point. The correct "curve" is the point (1,1). Whether stepcontrol=0 or 1, the algorithms stops right away signaling a singularity, taking no steps. The plots are simply a single point (1,1).

Curve	stepcontrol	N	M	avgMpt	maxMpt	maxf	maxD	minD
point	0	1	0	0	0	0	-	-
point	0	1	0	0	0	0	-	-

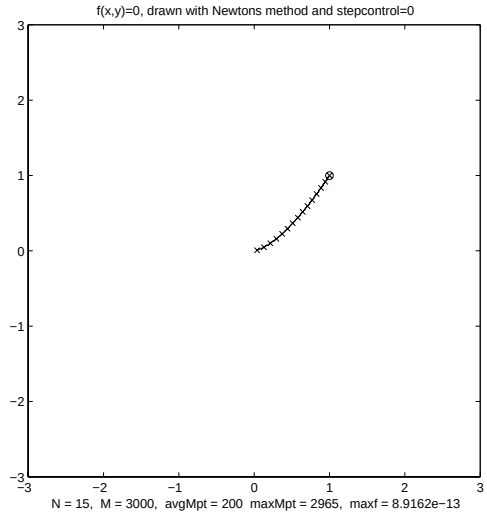
7. Figure 8. The correct curve is a figure 8, lying on its side between $x = -1$ and $x = +1$, centered at the origin. A problem is expected near (0,0) because the curve intersects itself, and indeed the gradient is zero there. The same qualitative picture was computed with stepcontrol = 0 or 1, both of them missing the singularity near (0,0) (because it did not land exactly at (0,0)) and so ended up bounding back and forth between two points straddling (0,0) until the maximum number of points (1000) was exceeded. So only one quarter of the figure 8 was plotted (the part with $x \geq 0$ and $y \geq 0$).



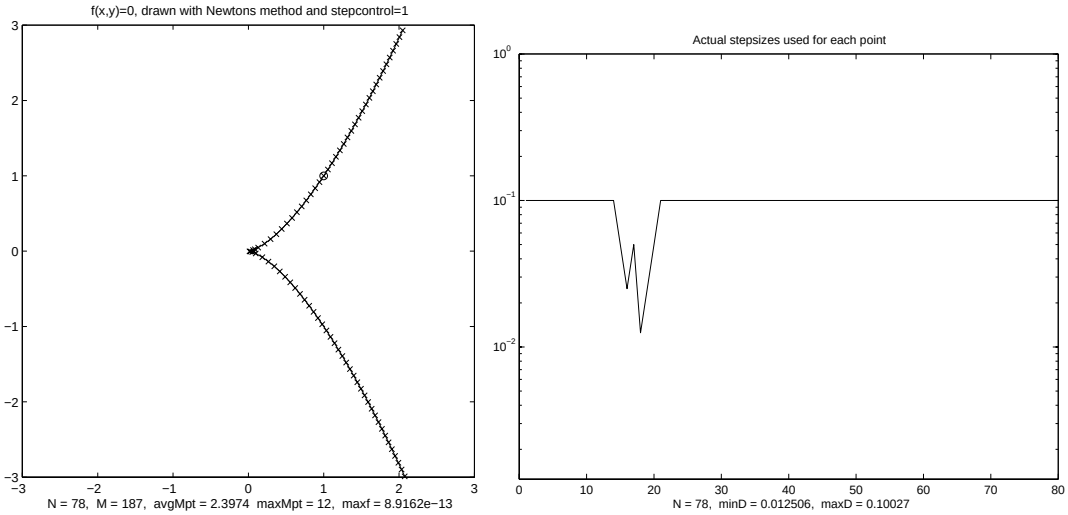
Curve	stepcontrol	N	M	avgMpt	maxMpt	maxf	maxD	minD
fig8	0	1000	2182	2.18	4	1.0e-12	.101	.100
fig8	1	1000	2031	2.03	6	6.7e-13	.101	.050

8. Cusp. The correct curve is a cusp with its sharp point pointing left at the origin. When stepcontrol=0, we start at (1,1) and follow the curve to the origin, but get stuck there; it is a singularity (the gradient is zero) and we exceed the maximum number of Newton steps. But with stepcontrol=1, the step size dropped by a factor of 8 near the origin, and the correct cusp was traced.

   The following is the curve with stepcontrol = 0.



   The following are the curve and stepsizes with stepcontrol = 1.



Curve	stepcontrol	N	M	avgMpt	maxMpt	maxf	maxD	minD
cusp	0	15	3000	200	2965	8.9e-13	.100	.100
cusp	1	78	187	2.40	12	8.9e-13	.100	.013