Math 128a - Homework 4 - Due Sept 28 at the beginning of class

1) Section 3.3, Problem 8

Answer: We have to show that, if $V_1 = a_1$ and $V_i = V_{i-1}(1+r) + a_i$ for $i > 1$ then, letting $x = 1 + r$:

$$V_n = \sum_{i=1}^{n} a_i x^{n-i}$$

Proof by induction: When $n = 1$ this just says $V_1 = a_1$ which we are given. Assume the statement is true for $n-1$. Thus

$$\begin{aligned} V_{n-1} &= \sum_{i=1}^{n-1} a_i x^{n-1-i} \quad \text{so :} \\ V_n &= (\sum_{i=1}^{n-1} a_i x^{n-1-i})x + a_n \\ &= (\sum_{i=1}^{n-1} a_i x^{n-i}) + a_n x^{n-n} \\ &= \sum_{i=1}^{n} a_i x^{n-i} \end{aligned}$$

2) You have just had a new baby, and realize that you have to start saving a lot of money to afford to send the child to college. Planning for the worst case, you ask yourself how much you would have to save to send your child to Stanford in 18 years. Assuming 7% inflation for college tuition (a widely recommended number for concerned parents), you compute that the current \$30,000/year cost of Stanford (out-of-date cost, but good enough for this question) will rise to $\$30000 \cdot 1.07^{18} \approx \$101398$ per year, or \$405592 for four years. By eating a lot of macaroni and cheese, you figure you can save \$10000/year to invest for the next 18 years. Assume that all your investment income will be taxed at a 20% rate at the end of 18 years. What interest rate do you need to get to have saved \$405592, after taxes, after 18 years? Given that the historic rate of return of the stock market is around 10%, do you expect to have saved enough? Hint: Use the formulation from the last question. Note that you pay tax only on the interest you earn, not the \$180000 you invest (which would be double taxation). Explain how you got your answer; "I plugged it into an HP-12C financial calculator" is not good enough. We note that the solver for this problem on the HP-12C was designed by Prof. W. Kahan in our department.

Answer: This problem can be cast in terms of the previous problem letting $a_i = 10^4$. Given an interest rate r and letting $x = 1 + r$, after 18 years we will have saved

$$\begin{aligned} V_{18} &= 10^4 \sum_{i=1}^{18} x^{18-i} \\ &= 10^4(x^{17} + x^{16} + \ldots + x^2 + x + 1) \end{aligned}$$

Then we take 20% in taxes off, but we don't get taxed on the total amount we put in, so we are left with $V_{18} - (0.2)(V_{18} - 18(10^4)) = 0.8 \cdot V_{18} + 3.6 \cdot 10^4$; we want to find an r that makes this quantity at least 405592. In other words, solve the following for x and then set $r = x - 1$:

$$0.8(10^4)x^{17} + 0.8(10^4)x^{16} + \ldots + 0.8(10^4)x + ((0.8 \cdot 10^4 + 3.6 \cdot 10^4) - 405592) = 0$$

To find the roots of this polynomial I ran the following line on Matlab:

```
≫ p = [.8e4*ones(1,17), .8e4 + 3.6e4 - 405592]; roots(p)
ans =
-1.2738+ 0.2321i
-1.2738- 0.2321i
-1.1080+ 0.6653i
-1.1080- 0.6653i
-0.7989+ 1.0096i
-0.7989- 1.0096i
-0.3887+ 1.2192i
-0.3887- 1.2192i
0.0667+ 1.2665i
0.0667- 1.2665i
0.5044+ 1.1461i
0.5044- 1.1461i
0.8625+ 0.8752i
0.8625- 0.8752i
1.0852+ 0.4904i
1.0852- 0.4904i
1.1013
```

So the root that matters to us is $x = 1.1013$, or $r = 10.03\%$. If we can make more that 10.03% interest on our investments we should be able to do it, so the stock market will have to do slightly better than 10% for our child to go to Stanford. Just to double check I ran:

```
≫ V=1e4;for cnt=2:18,V=V*1.1013+1e4;end,V=V*.8 + 1.8e5*.2
V =
405546.90
```

which is (almost) what we want. Roundoff in this last computation can only affect V in its last figures, because V is computed by multiplying and adding positive numbers, so a floating point error bound will turn out to be $O(V \cdot macheps)$.

Note that all the coeffients of the polynomial we wanted to solve are positive except the constant term, which is negative, which means that the slope is positive for positive arguments, and $p(0) < 0$ so there must be a unique positive real root, which is what we want.

3) Consider the following system of two simultaneous equations in two unknowns x and y:

$$\begin{aligned} f_1(x,y) = x^2 + a \cdot y^2 - 1 &= 0 \\ f_2(x,y) = (x-1)^2 + y^2 - 1 &= 0 \end{aligned}$$

where a is a parameter.

Part 1. Solve the equations explicitly by hand. Show that for all but a finite number of (complex) values of a, there are four (possibly complex) solutions (x, y), but for this finite set of a's, there are fewer solutions. Exhibit all solutions, and the finite set of a's, explicitly. Four is the upper bound on the number of solutions given by Bezout's theorem as described in class (the product of the degrees of the polynomials f_1 and f_2, which are both 2). This example shows that the upper bound is attainable for most, but not necessarily all, coefficients.

Answer: Eliminating y we get

$$(a-1)x^2 - 2ax + 1 = 0$$

Case 1: $a = 1$ Then $-2x + 1 = 0$, so $x = 1/2$, plugging back into $f_1 = 0$ we get $y^2 = 3/4$ or $y_\pm = \pm\sqrt(3)/2$. Thus we have two solutions in this case:

$$r_1 = (\frac{1}{2}, \frac{\sqrt(3)}{2}), r_2 = (\frac{1}{2}, -\frac{\sqrt(3)}{2})$$

Case 2: $a \neq 1$ Then we apply the quadratic formula:

$$x_\pm = \frac{a \pm \sqrt{a^2 - a + 1}}{a - 1}$$

Case 2a: $a^2 - a + 1 = 0$ In this case we only get one solution for x: $x = a/(a-1)$; note that $a^2 - a + 1 = 0$ means that $a - 1 = a^2$ so in fact $x = a/a^2 = a^{-1}$. Now we solve $a^2 - a + 1 = 0$ to get two solutions $a_\pm = e^{\pm i\pi/3}$ so $x_\pm = e^{\mp i\pi/3}$. Then $y_\pm = \pm\sqrt{(1-x^2)/a}$ so the solutions are:

If $a = e^{i\pi/3}$ then

$$r_1 = \left(e^{-i\pi/3}, \sqrt{\frac{1 - e^{-2i\pi/3}}{e^{i\pi/3}}}\right)$$
$$r_2 = \left(e^{-i\pi/3}, -\sqrt{\frac{1 - e^{-2i\pi/3}}{e^{i\pi/3}}}\right)$$

If $a = e^{-i\pi/3}$ then

$$r_1 = \left(e^{i\pi/3}, \sqrt{\frac{1 - e^{2i\pi/3}}{e^{-i\pi/3}}}\right)$$
$$r_2 = \left(e^{i\pi/3}, -\sqrt{\frac{1 - e^{2i\pi/3}}{e^{-i\pi/3}}}\right)$$

Case 2b: $a^2 - a + 1 \neq 0$ In this case we will get four distinct solutions, using $y =$

$\pm\sqrt{1-(x-1)^2}$ since we might have $a = 0$:

$$
\begin{aligned}
r_1 &= \left(\frac{a+\sqrt{a^2-a+1}}{a-1}, \sqrt{1-\left(\frac{a+\sqrt{a^2-a+1}}{a-1}-1\right)^2} \right) \\
r_2 &= \left(\frac{a+\sqrt{a^2-a+1}}{a-1}, -\sqrt{1-\left(\frac{a+\sqrt{a^2-a+1}}{a-1}-1\right)^2} \right) \\
r_3 &= \left(\frac{a-\sqrt{a^2-a+1}}{a-1}, \sqrt{1-\left(\frac{a-\sqrt{a^2-a+1}}{a-1}-1\right)^2} \right) \\
r_4 &= \left(\frac{a-\sqrt{a^2-a+1}}{a-1}, -\sqrt{1-\left(\frac{a-\sqrt{a^2-a+1}}{a-1}-1\right)^2} \right)
\end{aligned}
$$

Part 2. Write down two explicit formulas for Newton iteration for this system. First, write it in the form

$$
\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - J^{-1} \cdot \begin{bmatrix} x_i^2 + a \cdot y_i^2 - 1 \\ (x_i - 1)^2 + y_i^2 - 1 \end{bmatrix}
$$

where you explicitly exhibit the Jacobian J and its inverse J^{-1}. Second, evaluate this expression explicitly, i.e. multiply it out and simplify.

Answer: First we explicitly calculate J and J^{-1}

$$
\begin{aligned}
J &= 2\begin{pmatrix} x & ay \\ x-1 & y \end{pmatrix} \\
|J| &= 4y[x - ax + a] \\
J^{-1} &= \frac{2}{|J|}\begin{pmatrix} y & -ay \\ 1-x & x \end{pmatrix} \\
&= \frac{1}{2y[x-ax+a]}\begin{pmatrix} y & -ay \\ 1-x & x \end{pmatrix} \quad \text{so that :} \\
\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} &= \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \frac{1}{2y_i[x_i - ax_i + a]}\begin{pmatrix} y_i & -ay_i \\ 1-x_i & x_i \end{pmatrix}\begin{pmatrix} x_i^2 + ay_i^2 - 1 \\ (x_i-1)^2 + y_i^2 - 1 \end{pmatrix}
\end{aligned}
$$

Multiplying out and simplifying we get:

$$
\begin{aligned}
x_{i+1} &= \frac{x_i}{2} + \frac{1 - ax_i}{2(x_i - ax_i + a)} \\
y_{i+1} &= \frac{y_i}{2} + \frac{x_i^2 - x_i + 1}{2y_i(x_i - ax_i + a)}
\end{aligned}
$$

Part 3. Using these formulas, write a (very short) Matlab program to implement Newton iteration just for this example. It need only implement one step, i.e. not test test convergence, so that you can just run it "by hand" and look at the iterates to determine convergence. Thus it need only be a few lines long at most.

1. Try it for each of the finite values of a that only have two solutions, and starting guesses equal to the true solutions times (1+rand(1)/100), i.e. change the true solution by about 1%. How many steps does it take to converge to machine precision? Do the number of correct digits roughly double at each step?
2. Try it for 3 random complex choices of a (=randn(1)+sqrt(-1)*randn(1)), and again take the true solutions, perturb them randomly by 1%, and see if Newton converges quadratically.

Answer: The following short program works (using 2-by-1 column vectors):

```
function [u1] = f1f2Newton(u0,a)

    x0 = u0(1); y0 = u0(2);
    x1 = x0./2 + (1 - a.*x0)./(2.*(x0 - a.*x0 + a));
    y1 = y0./2 + (x0^2-x0+1)/(2.*y0.*(x0 - a.*x0 + a));
    u1 = [x1; y1];
```

1. When we only have two solutions, two cases:

   Case $a = 1$ The true solutions are

$$r_1 = (5.000000000000000e-01, 8.660254037844386e-01)$$
$$r_2 = (5.000000000000000e-01, -8.660254037844386e-01)$$

   My random perturbations were:

$$\tilde{r_1} = (5.022823383258417e-01, 8.661856500355939e-01)$$
$$\tilde{r_2} = (5.041070358214763e-01, -8.698766478912214e-01)$$

   The following is a list of the relative errors in the successive Newton iterations:

r1		
iteration	x error	y error
0	4.564676651683364e-03	1.850364324822745e-04
1	0	3.489185770072777e-06
2	0	6.087330680015810e-12
3	0	0

r2		
iteration	x error	y error
0	8.214071642952625e-03	4.447033643532061e-03
1	0	2.103965223123728e-05
2	0	2.213289028823506e-10
3	0	0

   This shows that it took 3 steps in both cases to converge and that the number of digits did roughly double at each step. Although there seem to be multiple roots so that we might expect convergence not to be quadratic, in fact, when $a = 1$, the system of equations simplifies to the following system:

$$\begin{aligned} 2x - 1 &= 0 \\ x^2 + y^2 - 1 &= 0 \end{aligned}$$

so that we really have a degree 1 polynomial and a degree 2 polynomial and we should only expect 2 solutions, so in fact these two roots are all the roots we should expect. Another way to say this is that the Jacobian is not actually singular (determinant 0) at these roots.

Case $a^2 - a + 1 = 0$ I will just show the output from one of these cases as they are all roughly the same. Take $a = e^{-i\pi/3}$, so:

$$\begin{aligned} r_2 = (\quad & 5.000000000000001e - 01 + 8.660254037844386e - 01i, \\ & -1.271229878418706e + 00 - 3.406250193166064e - 01i) \end{aligned}$$

and the perturbation is:

$$\begin{aligned} \tilde{r_2} = (\quad & 5.030771617405005e - 01 + 8.713552042620925e - 01i, \\ & -1.281297218656743e + 00 - 3.433225550033177e - 01i) \end{aligned}$$

The results were as follows. Note that we do not get quadratic convergence; instead the error approximately halves at every step. This is exactly the behavior we expect from Newton near a double root, as described in the homework.

r2	
x error	y error
6.154323481000892e-03	7.919370374270396e-03
3.077161740507443e-03	1.789652392442283e-03
1.538580870271995e-03	8.883170149424706e-04
7.692904351798341e-04	4.441500477180157e-04
⋮	⋮
(about 50 iterations later)	
1.415373477949795e-08	7.667170197736771e-09
1.784999861185183e-08	1.304599068391936e-08
1.616403172349245e-08	7.548663659410948e-09
1.343578819234603e-08	1.146385187630224e-08

2. In the general case here is one of the choices I made:

$$a = (1.253323064748307e - 01 + 2.876764203585488e - 01i)$$

and one root is (I won't include them all):

$$\begin{aligned} r_4 = (\quad & 9.438893783273251e - 01 - 1.543912042730986e - 01i, \\ & -1.010327552332886e + 00 + 8.574433541432508e - 03i) \end{aligned}$$

Perturbed root is:

$$\tilde{r_4} = (\quad 9.525902730462289e - 01 - 1.558144024197772e - 01i, \\ -1.017785863530629e + 00 + 8.637730631122583e - 03i)$$

The following are the relative errors for each successive iteration:

iteration	x error	y error
0	9.218129707448048e-03	7.382072458106622e-03
1	4.058560323436797e-05	6.779658244202396e-05
2	7.932815817136543e-10	2.977020893438061e-09
3	1.196520692413957e-16	3.948937458207650e-17
4	2.901988940035250e-17	6.867717318621998e-18
5	1.297808907974276e-16	2.208974879556959e-16

Newton does appear to have converged quadratically, i.e. number of correct digits doubled with each step, and it took about 3 iterations to converge.