## MATH 420 Homework #3 SOLUTIONS

1. How would you computationally find the maximum of $f$ over $\mathbb{R}^2$ where

$$f(x,y) = \frac{(\sin(10x+5)\,\cos(10y-6)+2)}{\sqrt{x^2+y^2+1}}?$$

Use what you know, and perhaps look over other optimization options at
http://cran.r-project.org/web/views/Optimization.html
to **(a)** describe how you plan to tackle this problem. Next, **(b)** implement it in R for the objective function below (or something equivalent), and finally **(c)** provide your answer and some additional information or arguments that characterizes how much confidence you have that you found the maximum. Recall the 5-step method and related discussions from the start of the semester.

```
obj = function(z) {
  x=z[1]
  y=z[2]
  return(-(sin(10*x+5)*cos(10*y-6)+2)/sqrt(x^2+y^2+1))
}
```

**Answer:** **(a)** Approach #1: Plotting the objective surface, the global maximum occurs on the top of one of three peaks. My plan was to chose $xy$ values near those peaks, and using those as initial conditions for three runs of `optim()` with the (default) optimization method Nelder-Mead. The largest of the three values would determine the maximum.

Approach #2: Recognizing this is a global optimization problem with multiple local maxima, a little reading suggests that using global optimization routines like the simulated annealing algorithm implemented in `GenSA` might be easier.

**(b) Approach #1:**

```
> x0=c(-.15,.1,-.15)
> y0=c(-.25, 0, .3)
>
> fit1=optim(c(x0[1],y0[1]),obj)
> fit2=optim(c(x0[2],y0[2]),obj)
> fit3=optim(c(x0[3],y0[3]),obj)
> fit1$value
[1] -2.984763
> fit2$value
[1] -2.99909
> fit3$value
[1] -2.984763
> # The Maximum
> c(fit2$par[1],fit2$par[2],fit2$value)
[1]  0.1568632086 -0.0000294735 -2.9990898398
```

Thus, the maximum occurs at approximately $(x,y) = (0.15686, 2.9\text{x}10^{-5})$.

**Approach #2** gives almost exactly the same answer (slightly better!) with less work.

```
> library(GenSA)
> fit=GenSA(par=c(2,2),fn=obj,lower=c(-3,-3),upper=c(3,3))
> c(fit$par[1],fit$par[2],fit$value)
[1]  1.568483e-01  1.710852e-12 -2.999090e+00
```

Thus, the maximum occurs at approximately $(x, y) = (0.15685, 1.7\text{x}10^{-12})$.

**(c)** We can be very confident in the maximum found via approach #1 because the function values approach zero away from the region around the origin, and because the three peaks found by optim in the code above are clearly the highest three peaks. We can be a little more confident in our results without plotting the point on the surface (although it never hurts to check!) because GenSA is a global optimization algorithm. To check this, additional initial conditions were checked, and all lead to the same answer.

2. If we let $N(0) = N_0 > 0$, the Ordinary Differential Equation (ODE)

$$\frac{dN(t)}{dt} = \lambda N(t)$$

has the solution

$$N(t) = N_0 \exp(\lambda t).$$

This implies

$$\log(N(t)) = \log(N_0) + \lambda\, t$$

Estimate parameters $\lambda$ and $N_0$ from the following data in two ways: using `lm()` and by writing your own objective function to minimize the sum of squared error. Since `t()` is the transpose function in R, we will avoid confusion by using $x$ in place of $t$:

```
x=1:10
N=c(1.21, 1.45, 1.83,  1.68, 2.71,  3.98,  2.71,  5.83, 5.84, 17.4)
```

**Hint:** Your objective function should take guesses at the two unknown parameter values in the form of a single vector, then calculate sum of squared differences between the given $N$ values, and those of the line equation.

**Answer:** From the R session below, we can see that both approaches yield estimates of $N_0 \approx 0.79$ and $\lambda \approx 0.25$.

```
> SSE = function(z) {   #return sum of (logN.line - logN.obs)^2
+     LN0=z[1];  m=z[2];
+     return(sum((LN0+m*x - log(N))^2))
+ }
>
> # Best fit parameters, two ways
> fitlm = lm(log(N)~x) # exactly the same as logN=log(N); fit(logN~x)
> fitSSE=optim(c(0,1),SSE)
>
> # In class on Wednesday, the line above had 'obj' instead of 'SSE',
> # which was the error that lead to inconsistent estimates.
>
> ## Comparison
> rbind(lm=c(N0=exp(as.numeric(fitlm$coeff[1])), lambda=as.numeric(
   fitlm$coeff[2])),
+       SSE=c(N0=exp(fitSSE$par[1]), lambda=fitSSE$par[2])  )
           N0    lambda
lm  0.7912814 0.2506492
SSE 0.7917954 0.2505654
```