APPM 4650 — HOMEWORK # 2 Solutions

1. (a) Show that $(1+x)^n = 1 + nx + o(x)$ as $x \to 0$.
   (b) Show that $x \sin\sqrt{x} = O(x^{3/2})$ as $x \to 0$.
   (c) Show that $e^{-t} = o(\frac{1}{t^2})$ as $t \to \infty$.
   (d) Show that $\int_0^\varepsilon e^{-x^2}dx = O(\varepsilon)$ as $\varepsilon \to 0$.

Soln:

(a) In this problem $f(x) = (1+x)^n - (1+nx)$ and $g(x) = x$. Then we need to look at the $\lim_{x\to 0}$ of $\frac{f(x)}{g(x)}$. If it goes to 0, we have little o.

$$\begin{aligned}\lim_{x\to 0}\frac{(1+x)^n-(1+nx)}{x} &= \lim_{x\to 0}\frac{n(1+x)^{n-1}-n}{1} \qquad \text{by L'Hopital's rule}\\ &= n-n=0\end{aligned}$$

(b) In this problem $f(x) = x\sin(\sqrt{x})$ and $g(x) = x^{3/2}$. We need to find a positive constant M such that $\left|\frac{f(x)}{g(x)}\right| \le M$.

$$\begin{aligned}\left|\frac{f(x)}{g(x)}\right| &= \left|\frac{x\sin\sqrt{x}}{x^{3/2}}\right|\\ &= \left|\frac{\sin\sqrt{x}}{x^{1/2}}\right| = \left|\frac{\sin\sqrt{x}-0}{x^{1/2}-0}\right|\\ &= \|\frac{\sin u - 0}{u-0}\| \quad \text{where} u=\sqrt{x} \text{ (Note: when } x \text{ is near 0, } u \text{ is near 0)}\\ &= \|\cos\eta\| \text{for some } \eta\in(0,u) \text{ by the Mean Value Theorem}\\ &\le 1 = M\end{aligned}$$

(c) In this problem $f(t) = e^{-t}$ and $g(x) = \frac{1}{t^2}$. Then we need to look at the $\lim_{t\to\infty}$ of $\frac{f(x)}{g(x)}$. If it goes to 0, we have little o.

$$\begin{aligned}\lim_{t\to\infty}\frac{e^{-t}}{\frac{1}{t^2}} &= \lim_{t\to\infty}\frac{t^2}{e^t} \text{ (simplification)}\\ &= \lim_{t\to\infty}\frac{2t}{e^t} \text{ by L'Hopital's rule}\\ &= \lim_{t\to\infty}\frac{2}{e^t} \text{ by L'Hopital's rule}\\ &= 0\end{aligned}$$

(d) In this problem $f(\varepsilon) = \int_0^\varepsilon e^{-x^2}dx$ and $g(\varepsilon) = \varepsilon$. We need to find a positive constant M such that $\left|\frac{f(x)}{g(x)}\right| \le M$.

$$
\begin{aligned}
\left|\frac{f(x)}{g(x)}\right| &= \left|\frac{\int_0^\varepsilon e^{-x^2}dx}{\varepsilon}\right| \\
&= \left|\frac{\int_0^\varepsilon e^{-x^2}dx - 0}{\varepsilon - 0}\right| \\
&= \|e^{-\eta^2}\| \text{ by the Mean Value Theorem } \exists \eta \in (0,\varepsilon) \text{ such that this is true} \\
&\leq 1
\end{aligned}
$$

2. Consider solving $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1+10^{-10} & 1-10^{-10} \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. The exact solution is $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and the inverse of $\mathbf{A}$ is $2\begin{bmatrix} 1-10^{10} & 10^{10} \\ 1+10^{10} & -10^{10} \end{bmatrix}$. In this problem we will investigate a perturbation in $\mathbf{b}$ of $\begin{bmatrix} \Delta b_1 \\ \Delta b_2 \end{bmatrix}$ and the numerical effects of the condition number.

   (a) Find an exact formula for the change in the solution between the exact problem and the perturbed problem Δx.

   (b) What is the condition number of $\mathbf{A}$?

   (c) Let Δb_1 and Δb_2 be of magnitude 10^{-5}. What is the relative error in the solution? How does the number of accurate digits relate to the condition number of $\mathbf{A}$ and the perturbation in the right hand side?

Soln:

   (a)

$$\Delta x = \mathbf{A}^{-1}\boldsymbol{\delta}\mathbf{b} = \begin{bmatrix} \Delta b_1 - 10^{10}(\Delta b_1 - \Delta b_2) \\ \Delta b_1 + 10^{10}(\Delta b_1 - \Delta b_2) \end{bmatrix}$$

   (b) $\kappa(\mathbf{A}) = 2 \times 10^{10}$

   (c) I let $\Delta b_1 = 10^{-5}$ and $\Delta b_2 = 2 \times 10^{-5}$ Then the solution to the peturbed problem is $\begin{bmatrix} 10^4 \\ -10^4 \end{bmatrix}$. The relative error in the solution is 10^4.

   With a condition number of $O(10^{10})$ and relative perturbation of 10^{-5} we expect the relative error to be less than or equal to $O(10^5)$ which is what we observed.

3. Let $f(x) = e^x - 1$

   (a) What is the relative condition number $\kappa(f(x))$? Are there any values of x for which this is ill-conditioned?

   (b) Consider computing $f(x)$ via the following algorithm:

```
y = math.e^x
return y -1
```

Is this algorithm stable? Justify your answer

(c) Let x have the value $9.999999995000000 \times 10^{-10}$, in which case $f(x)$ is equal to 10^{-9} up to 16 decimal places. How many correct digits does the algorithm listed above give you? Is this expected?

(d) Find a polynomial approximation of $f(x)$ that is accurate to 16 digits for $x = 9.999999995000000 \times 10^{-10}$. Hint: use Taylor series, and remember that 16 digits of accuracy is a relative error, not an absolute one.

(e) Verify that your answer from part (d) is correct.

(f) [Optional] How many digits of precision do you have if you do a simpler Taylor series?

(g) [Fact; no work required] Matlab provides `expm1` and Python provides `numpy.expm1` which are special-purpose algorithms to compute $e^x - 1$ for $x \approx 0$. You could compare your Taylor series approximation with `expm1`.

Soln:

(a) As we found in class, the condition number is defined as

$$\kappa(f(x)) = \frac{|f'(c)||x|}{|f(x)|}$$

for some c between x and the perturbed value. Since we assume the perturbation is small, we take $c = x$. Thus we get

$$\kappa(f(x)) = \frac{e^x|x|}{|e^x - 1|}.$$

The only point we have to be careful at is $x = 0$ but we can show that $\kappa(f(0)) = 1$. So the condition number is large when x is large.

(b) For $x \approx 0$, we have $e^x \approx 1$ and we are subtracting nearly equal numbers. This is an unstable algorithm.

(c) The computation gives us about 8 digits of accuracy. Below, we can see that the computation for $f(x)$ is incorrect in the 9th digit (an 8 instead of a 0). We also see that the relative error is on the order of 10^{-8} which is consistent with 8 digits of accuracy. This is expected as subtracting near equal numbers is unstable.

```
>>> import numpy as np
>>> x = 9.999999995e-10
>>> f = lambda x: np.exp(x) - 1
>>> f(x)
1.000000082740371e-09
>>> fx = 1e-9
>>> ((f(x) - fx))/fx
8.274037093680878e-08
```

(d) The Taylor remainder term is

$$R_n \leq \frac{1}{(n+1)!}|\xi|^{(n+1)} \leq \frac{1}{(n+1)!}|x|^{(n+1)}$$

for some $\xi \in (-x, x)$. Since we care about at $x = 9.999999995e - 10$, we find that we need two terms in the Taylor expansion in order to achieve our desired error.

(e,f) The Taylor polynomial $T_2(x) = x(1+\frac{x}{2})$ satisfies this. Below we see that T_1 gives around 10 digits of accuracy, but T_2 is accurate to machine precision. The value below for x is

```
>>> import numpy as np
>>> x = 9.999999995e-10
>>> T1 = lambda x: x
>>> T2 = lambda x: x*(1 + x/2)
>>> f = 1e-9
>>> (T1(x) - f)/f
>>> (T1(x) - f)/f
-5.000000746056408e-10
>>> (T2(x) - f)/f
0.0
```

4. *Practicing using your package*

(a) Create a vector $\mathbf{t}$ with entries starting at 0 incrementing by $\frac{\pi}{30}$ to π. Then create the vector $\mathbf{y} = \cos(\mathbf{t})$.
Write a code that evaluates the following sum:

$$S = \sum_{k=1}^{N} \mathbf{t}(k)\mathbf{y}(k)$$

Print the statement "the sum is: S with the numerical value of S using the `prini` package.

(b) *Wavy circles.* In one figure plot the parametric curve

$$x(\theta) = R(1 + \delta r \sin(f\theta + p))\cos(\theta)$$
$$y(\theta) = R(1 + \delta r \sin(f\theta + p))\sin(\theta)$$

for $0 \le \theta \le 2\pi$ and for $R = 1.2$, $\delta r = 0.1$, $f = 15$ and $p = 0$. Make sure to adjust the scale so that the axis have the same scale.
In a second figure use a for loop to plot 10 curves and let with $R = i$, $\delta r = 0.05$, $f = 2+i$ for the i^{th} curve. Let the value of p be a uniformly distributed random number (look up `random.uniform`) between 0 and 2.

Soln:

(a)
```
import numpy as np
import math
```

```
N = 30
t = np.linspace(0, math.pi, 30)
y = np.cos(t)


S = 0
k = 0
while k < N:
    S = S+ t[k]*y[k]
    k = k +1

print('The sum is', '%16.16e' % S)
```

(b)
```
import matplotlib.pyplot as plt
import numpy as np
import math


R = 1.2
delr = 0.1
f = 15.
p = 0.

theta =np.linspace(0, 2*math.pi, 300)

xx = R*(1+delr*np.sin(f*theta + p))*np.cos(theta)
yy = R*(1+delr*np.sin(f*theta + p))*np.sin(theta)

plt.plot(xx,yy)
plt.savefig('prob4bi.pdf')
plt.show()

input()

p = s = np.random.uniform(0,2,1)
delr = 0.05
icount = 1
R = icount
f = 2+icount
xx = R*(1+delr*np.sin(f*theta + p))*np.cos(theta)
yy = R*(1+delr*np.sin(f*theta + p))*np.sin(theta)

plt2 = plt.plot(xx,yy)
while icount <10:
    icount = icount + 1
```

```
    R = icount
    f = 2+icount
    xx = R*(1+delr*np.sin(f*theta + p))*np.cos(theta)
    yy = R*(1+delr*np.sin(f*theta + p))*np.sin(theta)
    plt2 = plt.plot(xx,yy)

plt2 = plt.savefig('prob4bii.pdf')
plt2 = plt.show()
```