This is the third and final assignment of the quarter. The assignment will cover function handles and some plotting.

**Instructions:** Attempt each problem and just do your best. Remember to use the `help` or `doc` command as much as possible. And there are definitely multiple ways to do things, though sometimes some are preferable to others.

**Submission:** Please submit three m-files (`simple_dynamics.m`, `simp.m`, `trap.m`) and one plot each (png image) for Questions 3, 4, and 5.

1. **Simple dynamics**

   Suppose we throw a ball into the air. The equations of motion for the ball are

   \[
   x(t) = v_0 \cos\left(\theta \frac{\pi}{180}\right) t,
   \]

   \[
   y(t) = h + v_0 \sin\left(\theta \frac{\pi}{180}\right) t - \frac{1}{2} gt^2,
   \]

   where \(v_0\) is the initial speed, \(\theta\) is the launch angle in degrees, \(h\) is the starting height, \(g\) is the acceleration due to gravity, and \(t\) is time.

   Our goal is to find the approximate time and \(x\)-coordinate for when the ball hits the ground; we do this by finding when \(y(t)\) becomes negative.

   To do this: Make a vector \(\mathbf{t}\) that separates the time scale 0 to 1 seconds into 1,000 evenly distributed partitions. Compute vectors \(\mathbf{x}\) and \(\mathbf{y}\) from \(\mathbf{t}\) with the following parameter values:

   \[
   v_0 = 4 \text{ m/s}, \quad g = 9.8 \text{ m/s}^2
   \]

   \[
   h = 1.5 \text{ m}, \quad \theta = 45^\circ.
   \]

   Find the first index of \(\mathbf{y}\) when the vector has negative values. Use that index and the one before it to interpolate from the values of \(\mathbf{x}\) and \(\mathbf{t}\) to give a time and \(x\)-coordinate for the moment when the ball hits the ground. Write a function `simple_dynamics` that takes as input \(v_0\), \(\theta\), \(h\) and outputs the position \(x\) and time \(t\) when the ball hits the ground.

2. **Trapezoid Rule and Simpson’s Rule**

   For this question, you will write two functions, `trap(f,a,b,n)` and `simp(f,a,b,n)`. Each function should be written in its own file, with the filename identical to the function name, i.e., `trap.m` and `simp.m`. In case you don’t remember your calculus, the Trapezoidal rule and Simpson’s rule are just different ways to approximate an integral \(\int_a^b f(x) \, dx\). The basic idea is to split up the domain from \(a\) to \(b\) into intervals and estimate the area of the function \(f\) over each interval.
The first function should compute the Trapezoidal rule:

\[ I = \frac{h}{2} \left[ f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n) \right], \quad (1) \]

\[ h = \frac{b-a}{n}, \quad x_k = a + kh, \quad (2) \]

and the second should compute Simpson’s rule (for \( n \) even):

\[ I = \frac{h}{3} \left[ f(x_0) + 4f(x_1) + 2f(x_2) + \cdots + 4f(x_{n-1}) + f(x_n) \right], \quad (3) \]

\[ h = \frac{b-a}{n}, \quad x_k = a + kh, \quad (4) \]

for function handles \( f \) and scalar inputs \( a, b, \) and \( n \).

Here are some sample function handles and ranges you can use to test your function (compare your results to what you get when you use the MATLAB built-in \texttt{integral} function):

\[
\begin{align*}
\text{f} & = @(x) x.^2; \ a = 0; \ b = 1; \\
\text{f} & = @(x) (1/sqrt(2*pi)) * exp(-x.^2/2); \ a = -10; \ b = 10;
\end{align*}
\]

Submit both your \texttt{trap.m} and \texttt{simp.m} functions to the Coursework dropbox.

3. A Nice Plot Create a function called \texttt{chaos.m}. In this script, we will be playing the “chaos game.” Let \( z_1, z_2, \) and \( z_3 \) be the vertices of an equilateral triangle. Start with a point \( z_0 \) anywhere inside the triangle. At random, pick one of the three vertices and move halfway toward it. Repeat indefinitely. If you plot all the points obtained, a very clear pattern will emerge. To do this, you can use the following skeleton code and fill in the line indicated below:

\[
\begin{align*}
\text{function } [ ] & = \text{chaos}( \text{z0, n} ) \\
& \text{figure(); } \% \text{ Create a new figure window} \\
& \text{hold on;} \\
& \text{plot(z0); } \% \text{ Plot starting point} \\
& \text{z = [0 1 .5+sqrt(3)/2*1i]; } \% \text{ Create equilateral triangle} \\
& \text{plot(z(1)); } \% \text{ Plot vertex 1} \\
& \text{plot(z(2)); } \% \text{ Plot vertex 2} \\
& \text{plot(z(3)); } \% \text{ Plot vertex 3} \\
& \text{r = randi([1 3], n);} \\
& \text{for } i=1:n \\
& \quad \text{z0 = ...; } \% \text{ [INSERT YOUR CODE IN PLACE OF THE ... ]} \\
& \quad \text{plot(z0);} \\
& \text{end} \\
& \text{hold off;} \\
& \text{end}
\end{align*}
\]
In the above code, the values $z_i$ represent complex numbers. If $z = a + ib$, then the complex number $z$ can be used to represent the point at $(a, b)$. The command `plot(z)` is equivalent to `plot(real(z), imag(z))`. Submit the resulting plot in a file called `chaos.png`.

Hint: If you wanted to move your point $z_0$, say, two-thirds of the way from point $z_0$ to point $z_1$, you could do something like:

$$z_0 = z_0 + (2/3)(z(1) - z_0);$$

4. **Assorted Plotting** Explore the `ezplot` function and its related functions like `ezsurf`, `ezplot3`, etc.

   - Plot the function $y = \sin \left( \frac{1}{x} \right)$ on the interval $[0, 1]$.
   - Plot the implicit function $y^3 \sin(x) - xy^2 \cos(y) + x^3 - 1 = 0$ on the intervals $x \in [-10, 10]$ and $y \in [-10, 10]$.
   - Plot the surface $z = \frac{xy^2}{x^2 + y^2}$ on the intervals $x \in [-10, 10]$ and $y \in [-10, 10]$.

Create one image with three subplots; since the third plot is a bit more complicated, it might be a good idea to give it more space. You can select multiple subplots at once by passing in multiple indices (e.g. `subplot(2,2,3:4)`).

5. **A Surface**

   Let $f(x, y)$ be the function

   $$f(x, y) = \exp(-x^2 - \frac{1}{2}y^2) \cos(4x) + \exp(-3((x + \frac{1}{2})^2 + \frac{1}{2}y^2))$$

   for $x \in [-3, 3]$ and $y \in [-5, 5]$.

   Use `meshgrid` and `surf` to plot

   $z(x, y) = \begin{cases} 
   f(x, y) & \text{if } -0.001 < f(x, y) < 0.001 \\
   0.001 & \text{if } f(x, y) > 0.001 \\
   -0.001 & \text{otherwise}
   \end{cases}$

What does the surface say?

Try out the following commands:

```matlab
colormap(cool)
view(35, 65)
camlight headlight
lighting phong
```