AME 60614: Numerical Methods
Fall 2021

Problem Set 0

Your Name Here!

1 Code Documentation

1. Download the document `sample_ps0.tex` and `sample_figure.eps` from the course Sakai page. Rename the file `lastname_firstname_ps0.tex`, and edit the report to add your name. Compile it using `pdflatex` or the LaTeX editor of your choice.
2. Document your response to Problem 2 in your report. Use appropriately highlighted code listings for codes requested in Problem 2. Print a copy and bring it to class on the due date.

Here is an example of how to insert a figure with `includegraphics{}`. In general, figures should be at the top of a page. However, in this case, the title is at the top, so we specify explicitly to put the figure here using "[h]". You can refer to the figure by its label: Figure 1.

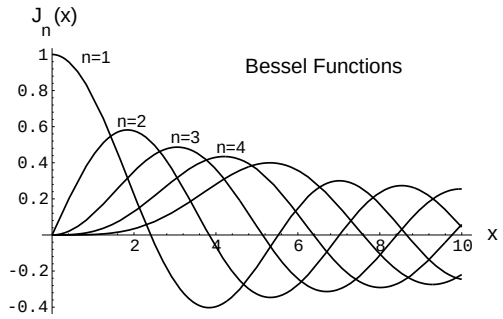

Figure 1: Sample figure showing Bessel functions.

2 Computing in Parallel

The Mandelbrot set is defined as the collection of points $c = (x, iy)$ on $\Omega = [-2, 0.5] \times [-2i, 2i] \subset \mathbb{C}$ for which the iterative sequence

$$z_{n+1}(c) = z_n^2(c) + c$$

remains bounded, where $z_n(c) \in \Omega$ is the value of the sequence at a point c for iteration n, $n \in [0, \infty)$, and $z_0 = 0$.

It may be shown that the sequence becomes unbounded if

$$|z_n(c)| > 2$$

for any n. To determine if a point is a member of the Mandelbrot set, one must check each point in the domain under the infinite sequence and exclude those points for which $|z_n(c)| > 2$. In practice, this involves creating a grid of finite resolution and iterating on each point a maximum of `MAX_ITER` times.

The following is an example code listing. Modify it to include your source code for solving the Mandelbrot set. Do not forget the other items requested in the problem set document.

```c
#include <stdio.h>
#include <stdlib.h>

int main() {

  // Define an integer counter
  int counter = 0;
  counter = 0;

  // Allocate double-precision data on the heap
  //   Note: *data denotes a pointer to memory
  int N = 10;
  double *data = malloc(N*sizeof(double));

  // Loop to perform operations
  for (int i=0; i<N; i++) {
    data[i] = 2.5*i;
    printf("Hello world %d %4.3f\n", counter, data[i]);
    counter++;
  }

  // Read the saved data in reverse
  for (int i=N; i>0; i--) {
    printf("%d %4.3f\n", i, data[i-1]);
  }

  // Clean up
  free(data);
  data = NULL;

  return 0;
}
```