Sam Kottler

Block Project Proposal

(1) I will write a program that draws the boundary of the Mandelbrot set using the Jungreis function. This function maps the unit circle in the complex plane to the boundary of the Mandelbrot set. The function is given by

$$\Psi(z) = \sum_{n=0}^{\infty} b_n z^{-n}$$

where b_n are coefficients of the Laurent series about infinity. These can be found recursively with the following rules:

$$b_n = \begin{cases} -\frac{1}{2} & \text{for } n = 0 \\ -\frac{w_{n,n+1}}{n} & \text{otherwise} \end{cases}$$

$$w_{n,m} = \begin{cases} 0 & \text{for } n = 0 \\ a_{m-1} + w_{n-1,m} + \sum_{j=0}^{m-2} a_j w_{n-1,m-j-1} & \text{otherwise} \end{cases}$$

$$a_j = u_{0,j+1}$$

$$u_{n,k} = \begin{cases} 1 & \text{for } 2^n - 1 = k \\ \sum_{j=0}^{k} u_{n-1,j} u_{n-1,k-j} & \text{for } 2^n - 1 > k \\ 0 & \text{for } 2^{n+1} - 1 > k \\ \frac{1}{2}(u_{n+1,k} - \sum_{j=1}^{k-1} u_{n,j} u_{n,k-j}) & \text{otherwise} \end{cases}$$

There is no known closed form formula for these coefficients. Obviously I can't actually sum to infinity so I'll either hard code a maximum number of iterations or keep going until the change after each iteration is small.

(2) There are two computationally difficult parts of this program. The one that will be easier to parallelize is actually mapping points on the unit circle to the boundary of the Mandelbrot set. For this part each point is completely independent so I wont need to do any kind of synchronization. The other computationally difficult task is recursively calculating the coefficients. I think that this can be parallelized but it will be much harder. I'm still thinking about what the best way to do this will be and looking at all the parallel patterns next week will probably help. Parallelism will help make both of these faster because it will allow for multiple calculations at the same time instead of doing one at a time.

(3) My program will output a png image which will make it easy to see if the output is correct. Also, I'm creating a fractal which is fun to look at. To compare performance I will time each of the two computationally difficult sections of code. This will allow me to see how the two sections compare rather than just the whole program.