NUMERICAL SCIENTIFIC COMPUTING

MINI PROJECT

Jimmy J. Nielsen and Ondrej Franek

February 8, 2021

1 | The mini-project

This mini-project relates to the Mandelbrot set (after the mathematician Benoit Mandelbrot), which leads to compelling two-dimensional fractal patterns. A fractal contains elements of self-similarity when plotted. The Mandelbrot set is a quadratic complex mapping of the type:

$$z_{i+1} \;=\; z_i^2 + c, \quad i = 0,1,\ldots,I-1 \tag{1.1}$$

where $c \in \mathbb{C}$ is a point in the complex plane, and $z_i \in \mathbb{C}$ for $i = 0,1,\ldots,I$. This provides us with $z_0, z_1, \ldots, z_I$ where the initial condition is $z_0 = 0 + j\cdot 0$ and $z_1,\ldots,z_I$ are referred to as iteratively achieved outputs. In Equation (1.1) the iteration number is $i+1 \in \{1,2,\ldots,I\}$ and the total number of iterations is I. For each observed complex point c we then compute I iterations of Equation (1.1). For each complex point c we then determine:

$$\imath(c) \;=\; \min\mathbb{T}, \quad \mathbb{T} = \left\{ i \;\middle|\; |z_i| > T, i = 1,2,\ldots,I \right\} \cup \{I\} \tag{1.2}$$

where T is a threshold value and the initial condition is always chosen such that $|z_0| \leq T$. From Equation (1.2) we see that $1 \leq \imath(c) \leq I$. So, if $|z_{i+1}| \leq T$, $\forall i = 0,\ldots,I-1$ we obtain $\mathbb{T} = \emptyset \cup \{I\} = \{I\}$ meaning that $\imath(c) = I$. Obviously, in a computational implementation we need not compute all I iterations if an $i+1 < I$ leads to $|z_{i+1}| > T$. We then just set $\imath(c)$ to the smallest $i+1$ that leads to $|z_{i+1}| > T$. For plotting purposes we then form a simple linear mapping as:

$$\mathcal{M}(c) \;=\; \frac{\imath(c)}{I}, \quad 0 < \mathcal{M}(c) \leq 1 \tag{1.3}$$

The smaller $\mathcal{M}(c)$ value we have, the faster that specific complex point c makes $|z_{i+1}|$ in Equation (1.1) increase. In a case with the number of iterations $I = 100$ a value of $0 < \mathcal{M}(c) \lesssim 0.1$ indicates an extremely 'active' (unstable) point whereas a value of $\mathcal{M}(c) = 1$ indicates a stable point. A point c is said to belong to the Mandelbrot set if $|z_{n+1}|$ remains bounded (finite) for $n \to \infty$.

In this mini-project, the computational task is then to determine $\mathcal{M}(c)$ for a c-mesh, which we limit c-wise to $-2 \leq \Re\{c\} \leq 1$ and $-1.5 \leq \Im\{c\} \leq 1.5$. We then select a certain number of points for each of $\Re\{c\}$ and $\Im\{c\}$ as p_{re} and p_{im}, respectively. The complex plane limited by the mesh is described by a complex matrix $\mathbf{C}$ as:

$$\mathbf{C} \;=\; \begin{bmatrix} -2.0 & \cdots & 1.0 \\ \vdots & & \vdots \\ -2.0 & \cdots & 1.0 \end{bmatrix} + j\cdot \begin{bmatrix} 1.5 & \cdots & 1.5 \\ \vdots & & \vdots \\ -1.5 & \cdots & -1.5 \end{bmatrix} \;\in\; \mathbb{C}^{p_{\text{re}} \times p_{\text{im}}} \tag{1.4}$$

You should select p_{re} and p_{im} according to the computational resources you have available and the desired resolution. A likely number could be something like $p_{\text{re}} = 5000$ and $p_{\text{im}} = 5000$. We use a threshold of $T = 2$. An example is shown in Figure 1.1.

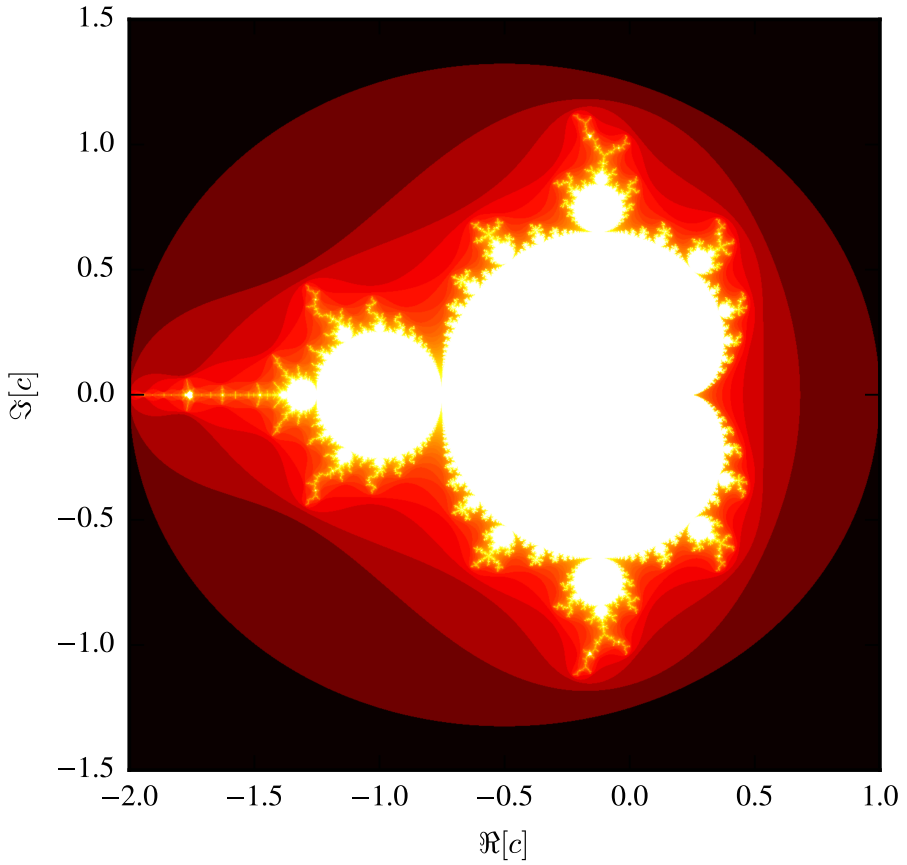

Figure 1.1:: Mandelbrot set plotting $\mathcal{M}(c)$ with $p_{\text{re}} = 5000$ and $p_{\text{im}} = 5000$ using the `matplotlib.pyplot.cm.hot` colour mapping in Pythons Matplotlib.

1.1 Tasks

This section aims to clarify some of the expectations we have for the mini-project and the documentation you deliver.
Output:

1. **Implementation:**
   Implement the following different versions of the computational code:

   (a) naïve version (write code using Python's standard library with focus on readability and ability to validate the implementation, i.e. no Numpy functionality);

   (b) a vectorised (NumPy) version;

(c) a cython or f2py version (for M.Sc. students);

(d) a multi-core parallel processing version;

(e) a multiprocessing (cluster) version;

(f) a GPU-processing version.

These implementations must be validated for correctness. The code must include docstrings and explanatory comments where necessary. The code must conform to PEP-8 (`https://www.python.org/dev/peps/pep-0008/`) – you can use the `pep8` command-line tool to check this.

2. **Output:**
   You must deliver plots for the Mandelbrot sets – you can use a colormap such as `matplotlib.pyplot.cm.hot`. Explore `matplotlib.pyplot` for relevant plotting functionality. Also, show the execution time for the different implementations – and for the parallel version also show execution time and speed-up versus number of computational units. Save all the figures in PDF format. Also save the relevant simulation data including the output **z**. Use the `h5py` package to store data to HDF files.

3. **Software design:**
   Explain briefly about your considerations for the overall design of the modules and functions you wish to use for the mini-project. You should provide a name for all functionalities, input, output and core task. In particular your considerations in relation to optimisation should be included. We also expect that you deliver an algorithm for the computational part of the mini-project. Considerations for data types, parameter passing between functions (avoiding global variables) etc. should be included.

4. **Test plan:**
   Briefly explain about the tests you intend to use to validate your implementation. You do not need to make a huge test scenario but include something to demonstrate that you have tried to implement testing. The modules `doctest` or `unittest` can be helpful for defining tests.

5. **Profiling and benchmarking:**
   Show the key execution time/profiling for the implementations. Benchmarking (using e.g. `time.time`, IPython's `%timeit` command, or the `line_profiler` package) must be provided such that we for example see speed-up versus number of computational cores. Show what you find relevant to demonstrate the quality of your optimisation.

You must deliver a zip-file including the code and documentation. The code must be accompanied by a short PDF document describing the above-mentioned aspects of the mini-project. Include a `README` file to explain what files you have included and how we should use the code.