

Creative Computing II Complex Numbers

12th January 2010

This lab sheet covers the manipulation and use of complex numbers with pen and paper and with *Octave*, and the construction of the Mandelbrot set.

1. Using pencil and paper, calculate the values of the following expressions:

(a) $(1 + 2i) + (3 - 4i)$
 $4 - 2i$

(b) $(3 + \frac{9}{4}i) - (4 - \frac{5}{8}i)$
 $-1 + \frac{23}{8}i$

(c) $2 \times (\frac{8}{3} + 2i)$
 $\frac{16}{3} + 4i$

(d) $i \times (7 - 9i)$
 $9 + 7i$

(e) $(2 + 3i) \times (1 - 4i)$
 $14 - 5i$

(f) $\frac{1}{2-4i}$
 $\frac{1}{10} + \frac{1}{5}i$

(g) $\frac{15-2i}{2-4i}$
 $\frac{19}{10} + \frac{14}{5}i$

(h) $|(3 + 4i)(5 - 12i)|$
 65

(i) $|\sqrt{(15 + 20i)}|$
 5

(j) $\arg\left(\frac{1}{2} + \frac{\sqrt{3}}{2}i\right)$
 $\frac{\pi}{3}$

(k) $\arg(\sqrt{i})$
 $\frac{\pi}{4}$

In some of these questions, you do not need to compute all intermediate results. For example, for two complex numbers z_1 and z_2 , $|z_1 z_2| = |z_1| |z_2|$, so part 1h is easily calculated as 5×13 . Similarly, $\arg z_1 z_2 = \arg z_1 + \arg z_2$, meaning that $\arg \sqrt{z} = \frac{1}{2} \arg z$.

2. For each of the expressions in part 1, check your answer using *Octave*, using the built-in support for complex numbers.

There are a few notational conversions that must be made when going between mathematical notation and Octave:

<i>Mathematics</i>	<i>Octave</i>
$\frac{9}{4}i$	<code>9i/4, (9/4)i</code>
\sqrt{z}	<code>sqrt(z)</code>
$ z $	<code>abs(z)</code>
$f \times g$	<code>f * g</code>
$(a + ib)(c + id)$	<code>(a + b*i)*(c + d*i)</code>

Otherwise, you should find that Octave and your pencil-and-paper calculations are in agreement.

3. There are many interesting applications of complex numbers; one is in the construction of *fractals*. You have already met fractal objects in CC112: the L-system objects such as the Koch curve, and the biomorph tree-like objects constructed by Richard Dawkins. This part of the lab is about another fractal object, the *Mandelbrot set*.

- (a) We will be applying a transformation repeatedly to complex numbers within the range $-2 \leq \Re z, \Im z \leq 2$. The first step is to construct a matrix representing that region of the complex plane. Construct in *Octave* a 5×5 complex matrix so that the real part of the (y, x) th entry is $3 - y$ and the imaginary part is $x - 3$. Your matrix should be printed by *Octave* like

$$\begin{array}{ccccc} -2 + 2i & -1 + 2i & 0 + 2i & 1 + 2i & 2 + 2i \\ -2 + 1i & -1 + 1i & 0 + 1i & 1 + 1i & 2 + 1i \\ -2 + 0i & -1 + 0i & 0 + 0i & 1 + 0i & 2 + 0i \\ -2 - 1i & -1 - 1i & 0 - 1i & 1 - 1i & 2 - 1i \\ -2 - 2i & -1 - 2i & 0 - 2i & 1 - 2i & 2 - 2i \end{array}$$

(hint: you may find the *Octave* function `repmat` useful in your answer – though there are many ways to achieve this)

The simplest construction of the above array is to initialize an array of the appropriate size, and then set each element to the desired value. This would look like

```
c = zeros(5,5);
for x = (1:5)
    for y = (1:5)
        c(y,x) = (x-3) + i*(3-y);
    end
end
```

This strategy is fine, but the strength of Octave is in array manipulation, and explicit loops such as in the code above are executed much slower than the equivalent array code. Using the hint in the question, it is possible to construct the array by adding together two 5×5 matrices, one for the real part and one for the imaginary part. Each of these matrices has a repeated pattern, which is why the Octave function `repmat` is appropriate:

```
c = repmat((-2:2),5,1) + repmat((2:-1:-2)',1,5)*i;
```

This solution is worth studying so that you understand how it works. It is possible to apply the same strategy using matrix multiplication rather than `repmat`: the `repmat` operation we have used is equivalent to multiplying the vector with an appropriate vector made up only of ones:

```
c = ones(5,1)*(-2:2) + (2:-1:-2)'*ones(1,5)*i;
```

It is worth studying this solution also.

- (b) The transformation we will be applying is the mapping of a point on the complex plane to itself squared, plus its original value: $z \rightarrow z^2 + c$. In order to do this, store the value of your array from part 3a in the Octave variable `c`, and use `c` to initialize the matrix `z`. Then to perform one iteration, use the Octave expression `z = z.^2 + c;`, which will elementwise square each entry in `z` and add the corresponding entry in `c`.
- (c) Repeat this mapping a few times (say four), then inspect the contents of the matrix `z`. You should find that some of the values have grown very large, while others are 0 or 1.
- (d) The *Mandelbrot set* is the set of numbers z which do not grow without bound under the iteration $z \rightarrow z^2 + c$. From your examination of the `z` matrix in part 3c, suggest numbers which are probably in the *Mandelbrot set*. (If you can, *prove* that those numbers do not diverge under the mapping).

The numbers -1 , 0 , i and $-i$ are all in the Mandelbrot set. The proof for 0 is trivial; the others follow the following repeating sequences:

- $-1 \rightarrow 0 \rightarrow -1 \rightarrow \dots$
- $i \rightarrow -1 + i \rightarrow -i \rightarrow -1 + i \rightarrow \dots$
- $-i \rightarrow -1 - i \rightarrow i \rightarrow -1 - i \rightarrow \dots$

and so none of them diverge under the mapping.

- (e) If the modulus of a complex number under the mapping exceeds 2, then the mapping will diverge. This fact enables us to visualise the results of your iteration, by treating all the numbers whose modulus exceeds 2 equally. Use the Octave expression `imagesc(min(abs(z),2))` to view

the contents of your matrix. (You may wish to read the online help for `imagesc`).

- (f) The matrix we started with is insufficiently detailed to see the structure of the Mandelbrot set. Repeat this question with a 513×513 matrix for the numbers in the range $-2 \leq \Re z, \Im z \leq 2$; you may need to do more iterations in part 3c to see the fine detail of the structure emerge.

Setting up the matrix is the same as in part 3a, except that 513-element vectors must be created rather than the 5-element vectors there. Each step is therefore $\frac{1}{128}$, so the Octave expression to construct c is

```
c = repmat((-2:1/128:2),513,1) +  
repmat((2:-1/128:-2)',1,513)*i;
```

256 iterations is enough to get a good picture of the set. You may find that it is slow to run; this is because the numbers in the matrix begin to overflow the range of representable floating point numbers. To speed it up, it is possible to replace all the elements at each iteration which will diverge with 2, to keep the numbers bounded:

```
for j = (1:256) z = z.^2 + c; z(abs(z)>2) = 2; end
```

Other resources:

- Penrose, R., *The Road to Reality*, Chapters 2-7.
- Peitgen, H.-O., Jürgens, H. and Saupe, D., *Chaos and Fractals*
- http://en.wikipedia.org/wiki/Mandelbrot_set.