MAT 305: Review #7

April 16, 2014

Directions: The usual counsels apply.

Part I
The Gaussian integers

Gaussian integers have the form $a+bi$, where a, b are integers. For example, 7, $2+3i$ and $-3+i$ are Gaussian integers. Mathematicians write $\mathbb{Z}[i]$ as a shortcut for "the Gaussian integers".

Let a and b be the last two digits of your student ID number.

1. Choose several *positive* integers c. For each one, compute $c(a+bi)$. Combine into one the plots of each result as an arrow, where $\alpha+\beta i$ is the arrow connecting the origin to (α, β). Use a different color for each plot. Describe how multiplying $a+bi$ by a *positive* integer affects the resulting arrow.
2. Repeat the previous exercise, only choose several *negative* integers c.
3. Choose several *positive* integers d. For each one, compute $di(a+bi)$. Combine into one the plots of each result as an arrow, using a different color for each plot. Describe how multiplying $a+bi$ by di, for a *positive* integer d, affects the resulting arrow.
4. Repeat the previous exercise, only choose several *negative* integers d.
5. If a, b, c, and d are all integers, what would be the result of multiplying $(c+di)(a+bi)$? Verify your answer by plotting $a+bi$, $c(a+bi)$, $di(a+bi)$, and $(c+di)(a+bi)$ as four arrows, each a different color.

When you are done with this section, you should have a *very good idea* of how multiplication of Gaussian integers works. If you don't, talk to me, either in class or in my office.

Part II
Division of Gaussian integers

6. Given the way multiplication of Gaussian integers works, how would you propose someone *divide* two Gaussian integers? Your explanation should be highly geometrical, based on the insights you gained above. If it helps, first think about how division of integers (such as 15 by 4) works on the number line; then try to do something similar with Gaussian integers.

The following pseudocode will divide two Gaussian integers (not very efficiently, though):

```
algorithm Gaussian division

  inputs
    $n, d \in \mathbb{Z}[i]$
  outputs
    $q, r \in \mathbb{Z}[i]$ such that $n = qd + r$, and $|r| < |d|$
    (Here, $|a + bi| = \sqrt{a^2 + b^2}$.)
  do
    — Stage 1
    if $|n - d| < |n + d|$ then
      $z = 1$
    else
      $z = -1$
    $q = 0$
    while $|n - qd| > |n - (q + z)d|$ do
      Add $z$ to $q$
    — Stage 2
    if $|n - (q + i)| > |n - (q - i)d|$ then
      $z = i$
    else
      $z = -i$
    while $|n - qd| > |n - (q + z)d|$ do
      Add $z$ to $q$
    return $q$, $n - qd$
```

7. Explain how the pseudocode agrees or disagrees with your answer to #6. In particular, what are the geometric meanings of qd, $|n - qd|$, and $|n - (q + z)d|$?

8. Implement the pseudocode as Python code. (Note: you can use Sage's *norm* function to find $|w|$ for any $w \in \mathbb{Z}[i]$. Or you can write your own function to do it.)

Part III
The Euclidean algorithm

The *Euclidean algorithm* is a very efficient technique to compute the greatest common divisor of two integers. It also applies to any set with a "legitimate" division operation; I omit the details, but we call such sets *Euclidean domains*. Since you don't know at the outset how many loops it will take to compute the gcd, you have to use a **while** loop.

```
algorithm Euclidean algorithm
  inputs
    $a, b \in S$, where $S$ is a Euclidean Domain
    a division operation on $S$
  outputs
    a greatest common divisor of $a$ and $b$
  do
    Let $s = \min(a, b)$ and $t = \max(a, b)$
    while $s \neq 0$ do
      Let $r$ be the remainder of dividing $t$ by $s$
      Let $t = s$
      Let $s = r$
    return $s$
```

1. Write Python code to implement the Euclidean algorithm.

2. Test your code over the integers. For a division operation, use the following Python code:

   ```
   def int_div(a,b):  return a.quo_rem(b)
   ```

   The result will be a tuple with quotient and remainder; you only want the remainder.

3. Test your code with single-variable polynomials. For a division operation, use the following Python code:

   ```
   def poly_div(a,b):
     a = a.polynomial(QQ)
     b = b.polynomial(QQ)
     return a.quo_rem(b)
   ```

   As above, the result will be a tuple with quotient and remainder; you only want the remainder.

4. Test your code over the Gaussian integers, using the Python code you wrote above for the division operation.