Topic of this homework: Greatest common divisors, Pell’s equation, Fibonacci sequence, continued fractions

Deliverable: Answers to problems

1 Algehraic generalization of the GCD algorithm

In this problem we are looking for integer solutions \((m, n)\) to the equation \(ma + nb = \gcd(a, b)\) given positive integers \((a, b) \in \mathbb{Z}^+\). While it is not explicitly stated in the text, this requires that either \(m\) or \(n\) be negative. Integral (whole number) polynomial relations are known as ‘Diophantine equations.’ The above equation is a linear Diophantine equation, possibly the simplest form of such relations.

1. By inspection, find at least one integer pair \((m, n)\) that satisfies \(2m + 3n = 1\).

2. Use the Euclidean GCD algorithm to solve part (1). Complete the Euclidean algorithm below for \((a, b) = (2, 3)\), simultaneously keeping track of the operations on \(a\) and \(b\), as in this example:

\[
\begin{align*}
a_1 &= 2, b_1 = 3 & a_1 &= a, b_1 &= b \\
a_2 &= 3 - 2 = 1, b_2 = 2 & a_2 &= b_1 - a_1, b_2 &= a_1 \\
a_3 &= 2 - 1 = 1, b_3 = 1 & a_3 &= b_2 - a_2, b_3 &= a_2
\end{align*}
\]

What are \((a_3, b_3)\) in terms of \((a, b)\)? Because \(\gcd(a, b) = 1\), both \(a_3\) and \(b_3\) give equations of the form \(ma + nb = \gcd(a, b)\). Find two solutions \((m, n)\) from \(a_3\) and \(b_3\).

3. Using the Euclidean algorithm, find two integer pairs \((m, n)\) that satisfy \(12m + 15n = 3\). Note: You may do this using the ‘algebraic’ method shown in part (2), or use the Matrix approach described below.

4. Does the equation \(12m + 15n = 1\) have integer solutions for \(n\) and \(m\)? Why, or why not?

Note: Matrix approach
It can be difficult to keep track of the a’s and b’s when the algorithm has many steps. Here is another way to run the Euclidean algorithm, using matrix algebra. Define

\[
\begin{bmatrix} a_0 \\ b_0 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} Q = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} S = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}
\]

where \(Q\) sets \(a_{i+1} = a_i - b_i\) and \(b_{i+1} = b_i\) assuming \(a_i > b_i\), and \(S\) is a ‘swap-matrix’ which swaps \(a_i\) and \(b_i\) if \(a_i < b_i\). Using these matrices, the algorithm is implemented by assigning

\[
\begin{bmatrix} a_{i+1} \\ b_{i+1} \end{bmatrix} = Q \begin{bmatrix} a_i \\ b_i \end{bmatrix} \text{ for } a_i > b_i, \quad \begin{bmatrix} a_{i+1} \\ b_{i+1} \end{bmatrix} = QS \begin{bmatrix} a_i \\ b_i \end{bmatrix} \text{ for } a_i < b_i.
\]

The result of this method is a cascade of \(Q\) and \(S\) matrices, which are multiplied together to generate two solutions for \((m, n)\). For part (2) above, the result is as follows:

\[
\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix}
\]

This matrix method is a more transparent approach to the operations on \((a, b)\). It can also be classified in terms of standard matrix operations, to be discuss later in the course.
2 Continued fractions

The Continued Fraction Algorithm (CFA, p. 46-48) naturally follows from the concept of the Euclidean Algorithm (EA, p. 37, 3.3) which we used to find $gcd(a, b)$ in the previous problem. The two are so similar that Gauss may have confused them (e.g., Exercise 3.4.1, p. 48). Maybe Gauss knew something we don’t? Here we explore the CFA in more detail.

In its simplest form the CFA starts with a number we denote $\alpha \in \mathbb{R}$. Let us work with an irrational number, $\pi \in \mathbb{I}$, as an example, because its CFA representation will be infinitely long. We can represent the CFA using a set of integers $n_k$, $k = 1, 2\ldots \infty$ for which

$$\alpha = n_1 + \frac{1}{n_2 + \frac{1}{n_3 + \frac{1}{n_4 + \ldots}}}$$

Like the EA, the CFA is recursive, with two steps per iteration. Let us define $\alpha_0 \equiv \alpha$. On the first iteration, the first step (i) is to find the integer part of $\alpha_0$ (for $\alpha_0 = \pi$, $n_1 = 3$). The second step (ii) is to find $\alpha_1$, defined as the reciprocal of the difference between $\alpha_0$ and $n_1$, $\alpha_1 \equiv 1/(\alpha_0 - n_1)$ (for $\alpha = \pi$, $\alpha_1 = 1/0.1416 = 7.0625\ldots$). Thus,

$$\alpha = n_1 + \frac{1}{\alpha_1} = n_1 + \frac{1}{n_2 + \frac{1}{\alpha_2} = \ldots}$$

In matlab code:

```matlab
%Note: In Matlab, vectors are indexed starting from 1, so here alpha(1) represents alpha_0 and n(2) represents n_1.
alpha0 = pi;
K=10;
n=zeros(1,K); alpha=zeros(1,K);
alpha(1)=alpha0;
for k=2:K %k=1 to K
    n(k)=round(alpha(k-1));
    %n(k)=fix(alpha(k-1));
    alpha(k)= 1/(alpha(k-1)-n(k));
    %disp([fix(k), round(n(k)), alpha(k)]); pause(1)
end
disp([n; alpha]);
%Now compare this to Matlab's rat() function
rat(alpha0,1e-20)
```

1. By hand (you may use Matlab as a calculator), find the first 3 values of $n_k$ for $\alpha = e^\pi$.

2. For part (1), what is the error (remainder) when you truncate the continued fraction after $n_1\ldots n_3$ (set $\alpha_3 = 1$)?

3. Use the program to find the first 10 values of $n_k$ for $\alpha = e^\pi$, and verify your result using the Matlab command `rat()`.

4. Discuss the similarities and differences between the EA and CFA.
3 The Fibonacci sequence

The Fibonacci sequence is famous in mathematics. Its not entirely clear why this is, but it has been observed to play a role in the mathematics of genetics. The underlying equation (Section 10.6, p. 192) is called a generating function. This may be analyzed as a ‘discrete time difference equation’ (a recurrence relation), where the current time sample is equal to the sum of the previous two samples. If \( f_n \) is the Fibonacci sequence,

\[
f_n = f_{n-1} + f_{n-2}.
\]

To start the computation for \( f_2 \) we need to know \( f_1 \) and \( f_0 \). Typically we take them to be \( \{1,0\} \), which leads to the Fibonacci sequence \( \{0, 1, 1, 2, 3, 5, 8, 13, \ldots \} \)

1. Write a matlab program to compute \( f_n \), and turn in your code (you can write it down). Calculate \( f_{40} \).

2. What happens when you calculate \( f_{100} \)? *Hint: how does this relate to the Matlab command `intmax`?*

3. The general solution to this equation is given on page 194 (Section 10.6) as

\[
f_n = \frac{1}{\sqrt{5}} \left[ \left( \frac{1 + \sqrt{5}}{2} \right)^n - \left( \frac{1 - \sqrt{5}}{2} \right)^n \right]
\]

(a) Check that your computer program and the above formula agree.

(b) Explain where the two numbers \( (1 \pm \sqrt{5})/2 \) come from, and show how they are determined. *Hint: How do they relate to the generating function \( f(x) \), as discussed on p. 193?*

(c) Give the numeric values of these two numbers.

(d) What happens to each of the two terms \( [(1 \pm \sqrt{5})/2]^n \) in the solution as the power \( n \to \infty \)?

(e) What happens to \( f_{n+1}/f_n \) as \( n \to \infty \)

4. Replace the Fibonacci sequence with

\[
f_n = \frac{f_{n-1} + f_{n-2}}{2},
\]

such that the value \( f_n \) is the average of the previous two values in the sequence.

(a) Modify your computer program to calculate this recurrence relation. Explain what happens as \( n \to \infty \)?

(b) Can you find the general solution for \( f_n \)?

(c) Now consider

\[
f_n = \frac{f_{n-1} + 1.01f_{n-2}}{2}.
\]

Now what happens as \( n \to \infty \)? Why?
4 Pell’s equation

Pell’s equation is one of the most important equations of Greek number theory. Like the Fibonacci equation, it has a recursive solution. Here we seek integer solutions \((x, y)\) to

\[ x^2 - Ny^2 = 1 \]

As shown in the lecture notes the solution for the case of \(N = 2\) is given by the recursion

\[
\begin{bmatrix}
  x_{n+1} \\
  y_{n+1}
\end{bmatrix} = i \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix}
  x_n \\
  y_n
\end{bmatrix}
\]

with \([x_0, y_0]^T = [1, 0]^T\) and \(i = \sqrt{-1}\). It follows that the general solution to the above equation is

\[
\begin{bmatrix}
  x_n \\
  y_n
\end{bmatrix} = i^n \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}^n \begin{bmatrix}
  x_0 \\
  y_0
\end{bmatrix}
\]

1. Work this out by hand to find the first 3 values of \([x_n, y_n]^T\) and show that they satisfy Pell’s equation for \(N=2\).

2. By hand, find the *eigenvalues* \(\lambda_{\pm}\) of the matrix

\[
A = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}
\]

*Hint: The eigenvalues are found from \(|A - \lambda I| = 0\). You can check your work using the Matlab command \([E, \text{Lambda}] = \text{eig}(A)\).*

3. By hand, show that the matrix of *eigenvectors* \(E\) is

\[
E = \frac{1}{\sqrt{3}} \begin{bmatrix} -\sqrt{2} & \sqrt{2} \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.8165 & -0.8165 \\ 0.5774 & 0.5774 \end{bmatrix}
\]

*Note: The eigenvectors may also be found using the Matlab command \([E, \text{Lambda}] = \text{eig}(A)\).*

4. Show that

\[
E^{-1}AE = \Lambda \equiv \begin{bmatrix} \lambda_+ & 0 \\ 0 & \lambda_- \end{bmatrix}
\]

where \(\lambda_{\pm}\) are the eigenvalues.

5. Inverting the expression in part (4), we find the following formula for \(A^n\)

\[
A^n = (E\Lambda E^{-1})^n = E \begin{bmatrix} \lambda_+^n & 0 \\ 0 & \lambda_-^n \end{bmatrix} E^{-1}
\]

We say that \(A\) has been *diagonalized*. Diagonalization allows us to compute \(A^n\) without \(n\) matrix multiplications (e.g. \(A^3 = (E\Lambda E^{-1})^3 = E\Lambda E^{-1}E\Lambda E^{-1}E\Lambda E^{-1} = E\Lambda^3 E^{-1}\)). Use this diagonalization to solve for the \(n = 10\) solution \([x_{10}, y_{10}]^T\) to Pell’s equation with \(N = 2\).