GRAND CHALLENGE: MAPPING THE HUMAN GENOME

Some of the goals of the Human Genome Project are to

- identify all the genes in human DNA,
- determine the sequences of the chemical base pairs that make up human DNA,
- store this information in databases,
- improve tools for data analysis,
- transfer related technologies to the private sector, and
- address the ethical, legal, and social issues that may arise from the project.

The majority of the human genome was sequenced in the year 2000, and 90 percent of the sequence of the genome's three billion base-pairs was published February 2001. One of the surprises from the results was the smaller-than-expected number of genes in the human genome. Early estimates of the number of genes ranged from 50,000 to 100,000. The deciphering of the human genome resulted in mapping roughly 30,000 genes. Another interesting result occurred when the human genome was matched to a mouse genome. When scientists compared the human and mouse

SECTIONS

4.1 Matrix Operations and Functions
4.2 Solutions to Systems of Linear Equations

OBJECTIVES

After reading this chapter, you should be able to

- perform operations that apply to an entire matrix as a unit and
- solve simultaneous equations using MATLAB.
genomes, they discovered that more than 90 percent of the mouse genome could be lined up with a region on the human genome.

The U.S. Department of Energy sponsors a comprehensive website containing information about the Human Genome Project at http://www.ornl.gov/hgmds/

4.1 MATRIX OPERATIONS AND FUNCTIONS

Many engineering computations use a matrix as a convenient way to represent a set of data. In this chapter, we are generally concerned with matrices that have more than one row and more than one column. Recall that scalar multiplication and matrix addition and subtraction are performed element by element. Matrix multiplication is covered in this section. Matrix division is presented in the next section and is used to compute the solution to a set of simultaneous linear equations.

4.1.1 Transpose

The transpose of a matrix is a new matrix in which the rows of the original matrix are the columns of the new matrix. We use a superscript \( T \) after the name of a matrix to refer to the transpose of the matrix. For example, consider the following matrix and its transpose:

\[
\mathbf{A} = \begin{bmatrix}
2 & 5 & 1 \\
7 & 3 & 8 \\
4 & 5 & 21 \\
16 & 13 & 0 \\
\end{bmatrix}, \quad \mathbf{A}^T = \begin{bmatrix}
2 & 7 & 4 & 16 \\
5 & 3 & 5 & 13 \\
1 & 8 & 21 & 0 \\
\end{bmatrix}
\]

If we consider a couple of the elements, we see that the value in position (3,1) of \( \mathbf{A} \) has now moved to position (1,3) of \( \mathbf{A}^T \), and the value in position (4,2) of \( \mathbf{A} \) has now moved to position (2,4) of \( \mathbf{A}^T \). In general, the row and column subscripts are interchanged to form the transpose; hence, the value in position \((i,j)\) is moved to position \((j,i)\).

In MATLAB, the transpose of the matrix \( \mathbf{A} \) is denoted by \( \mathbf{A}' \). Observe that the transpose will have a different size than the original matrix if the original matrix is not a square matrix. We frequently use the transpose operation to convert a row vector to a column vector or a column vector to a row vector.

The dot product is a scalar computed from two vectors of the same size. This scalar is the sum of the products of the values in corresponding positions in the vectors, as shown in the following summation equation, which assumes that there are \( n \) elements in the vectors \( \mathbf{A} \) and \( \mathbf{B} \):

\[
\text{dot product} = \mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{n} a_i b_i
\]

In MATLAB, we can compute the dot product with the following statement:

\[
\text{dot_product} = \text{sum}(\mathbf{A} \cdot \mathbf{B});
\]

Recall that \( \mathbf{A} \cdot \mathbf{B} \) contains the results of an elementwise multiplication of \( \mathbf{A} \) and \( \mathbf{B} \). When \( \mathbf{A} \) and \( \mathbf{B} \) are both row vectors or are both column vectors, \( \mathbf{A} \cdot \mathbf{B} \) is also a vector. We then sum the elements in this vector, thus yielding the dot product. The \text{dot} function may also be used to compute the dot product:

\[
\text{dot}(\mathbf{A}, \mathbf{B});
\]
EXAMPLE 4.1: To illustrate, assume that $A$ and $B$ are the following vectors:

$$A = \begin{bmatrix} 4 & -1 & 3 \end{bmatrix} \quad B = \begin{bmatrix} -2 & 5 & 2 \end{bmatrix}$$

The dot product is then

$$A \cdot B = 4 \cdot (-2) + (-1) \cdot 5 + 3 \cdot 2$$
$$= (-8) + (-5) + 6$$
$$= -7$$

You can test this result by typing:

```python
dot(A, B)
```
In this example, we cannot compute $BA$, because $B$ does not have the same number of elements in each row as $A$ has in each column.

An easy way to decide if a matrix product exists is to write the sizes of the two matrices side by side. Then, if the two inside numbers are the same, the product exists, and the size of the product is determined by the two outside numbers. To illustrate, in the previous example, the size of $A$ is $2 \times 3$, and the size of $B$ is $3 \times 3$. Therefore, if we want to compute $AB$, we write the sizes side by side:

$$2\times3, \ 3\times3 \uparrow \underrightarrow{\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad}$$

The two inner numbers are both the value 3, so $AB$ exists, and its size is determined by the two outer numbers, $2 \times 3$. If we want to compute $BA$, we again write the sizes side by side:

$$3\times3, \ 2\times3 \uparrow \underrightarrow{\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad}$$

The two inner numbers are not the same, so $BA$ does not exist. If the two inner numbers are the same, then $A$ is said to be conformable for multiplication to $B$.

In MATLAB, matrix multiplication is denoted by an asterisk. Thus, the command to perform matrix multiplication of matrices $A$ and $B$ is

$$A \ast B;$$

EXAMPLE 4.2: Generate the matrices in our previous example, and then compute the matrix product:

$$A = \begin{bmatrix} 2, 5,1; & 0,3,-1; \\ \end{bmatrix}$$

$$B = \begin{bmatrix} 1,0,2; & -1,4,-2; & 5,2,1; \\ \end{bmatrix}$$

$$C = A \ast B;$$

The results are as follows:

$$C = \begin{bmatrix} 2 & 22 & -5 \\ -8 & 10 & -7 \end{bmatrix}$$

Note that $B \ast A$ does not exist, because the number of columns of $B$ does not equal the number of rows of $A$. In other words, $B$ is not conformable for multiplication with $A$. Execute the MATLAB command $C = B \ast A$:

$$C = B*A;$$

You will get the following warning message:

??? Error using ==> *
    Inner matrix dimensions must agree.

Assume that $I$ is a square identity matrix. (Recall that an identity matrix is a matrix with ones on the main diagonal and zeros elsewhere.) If $A$ is a square matrix of the same size, then $A \ast I$ and $I \ast A$ are both equal to $A$. 
EXAMPLE 4.3: Generate a square identity matrix \( I \) of order \( 3 \times 3 \):

\[
I = \text{eye}(3);
\]

Use the square matrix \( A \) from the preceding discussion to verify that \( A \cdot I = I \cdot A \):

\[
\begin{align*}
A \cdot I &= I \cdot A \quad \text{if true, 0 if false} \\
C &= A \cdot I; \\
C &= \\
&= \begin{bmatrix}
1 & 0 & 2 \\
-1 & 4 & -2 \\
5 & 2 & 1 \\
\end{bmatrix} \\
C &= I \cdot A; \\
C &= \\
&= \begin{bmatrix}
1 & 0 & 2 \\
-1 & 4 & -2 \\
5 & 2 & 1 \\
\end{bmatrix}
\]

4.1.3 Matrix Powers
Recall that if \( A \) is a matrix, then the operation \( A^2 \) squares each element in \( A \). If we want to square the matrix—that is, to compute \( A^2 \)—we use the operation \( A^2 \). \( A^4 \) is equivalent to \( A^2 \cdot A^2 \). To perform a matrix multiplication between two matrices, the number of rows in the first matrix must be the same value as the number of columns in the second matrix. Therefore, to raise a matrix to a power, the number of rows must equal the number of columns, and thus the matrix must be a square matrix.

4.1.4 Matrix Inverse
By definition, the inverse of a square matrix \( A \) is the matrix \( A^{-1} \) such that the matrix products \( AA^{-1} \) and \( A^{-1}A \) are both equal to the identity matrix. For example, consider the following two matrices \( A \) and \( B \):

\[
A = \begin{bmatrix}
2 & 1 \\
4 & 3 \\
\end{bmatrix} \\
B = \begin{bmatrix}
1.5 & -0.5 \\
-2 & 1 \\
\end{bmatrix}
\]

If we compute the products \( AB \) and \( BA \), we obtain the following matrices (do the matrix multiplications by hand to be sure you follow the steps):

\[
AB = \begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix} \\
BA = \begin{bmatrix}
1 & 0 \\
1 & 1 \\
\end{bmatrix}
\]

Therefore, \( A \) and \( B \) are inverses of each other, or \( A = B^{-1} \) and \( B = A^{-1} \).

Computing the inverse of a matrix is a tedious process; fortunately, MATLAB contains an \texttt{inv} function that performs the computations for us. (We do not present the steps for computing an inverse in this text. Refer to a linear algebra text if you are interested in the techniques for computing an inverse.) Thus, if we execute \texttt{inv(A)}, using the matrix \( A \) defined previously the result will be the matrix \( B \). Similarly, if we execute \texttt{inv(B)}, the result should be the matrix \( A \). Try this yourself.

There are matrices for which an inverse does not exist; these matrices are called \texttt{singular}, or \texttt{ill-conditioned matrices}. When you attempt to compute the inverse of an ill-conditioned matrix in MATLAB, an error message is printed.
**EXAMPLE 4.4:** Create the matrix \( A = [1, 2; 3, 4] \).
Raise the matrix to the second power using the following command:

\[
C = A^2;
\]

The results will be as follows:

\[
C = \begin{bmatrix}
7 & 10 \\
15 & 22
\end{bmatrix}
\]

Note that raising \( A \) to the **matrix power** of two is different from raising \( A \) to the **array power** of two:

\[
C = A.^2;
\]

Raising \( A \) to the array power of two produces the following results:

\[
C = \begin{bmatrix}
1 & 4 \\
9 & 16
\end{bmatrix}
\]

### 4.1.5 Determinants

A **determinant** is a scalar computed from the entries in a square matrix. Determinants have various applications in engineering, including computing inverses and solving systems of simultaneous equations. For a \( 2 \times 2 \) matrix \( A \), the determinant is

\[
|A| = a_{1,1}a_{2,2} - a_{1,2}a_{2,1}
\]

Therefore, the determinant of \( A \), or \( |A| \), is equal to 8 for the following matrix:

\[
A = \begin{bmatrix}
1 & 3 \\
-1 & 5
\end{bmatrix}
\]

For a \( 3 \times 3 \) matrix \( A \), the determinant is the following:

\[
|A| = a_{1,1}a_{2,2}a_{3,3} + a_{1,2}a_{2,3}a_{3,1} + a_{1,3}a_{2,1}a_{3,2} - a_{1,2}a_{2,1}a_{3,3} - a_{1,3}a_{2,3}a_{3,1} - a_{1,1}a_{2,3}a_{3,2}
\]

If

\[
A = \begin{bmatrix}
1 & 3 & 0 \\
-1 & 5 & 2 \\
1 & 2 & 1
\end{bmatrix}
\]

then \( |A| \) is equal to \( 5 + 6 + 0 - 0 - 4 - (-3) \), or 10.

A more involved process is necessary for computing determinants of matrices with more than three rows and columns. We do not include a discussion of the process for computing a general determinant here, because **MATLAB** will automatically compute a determinant using the `det` function, with a square matrix as its argument, as in `det(A)`. 


PRACTICE!

Use MATLAB to define the following matrices:

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
\]

Compute the matrix specified in each problem, if the matrix exists:

a. \( \text{det}(B) \)

\begin{align*}
\text{b. } & AC^T \\
\text{c. } & (CB)D^T \\
\text{d. } & B - A \\
\text{e. } & AC^T \\
\text{f. } & (AC^T)^T \\
\text{g. } & \text{det}(B) \\
\text{h. } & \text{det}(A^*C^T)
\end{align*}

4.2 SOLUTIONS TO SYSTEMS OF LINEAR EQUATIONS

Consider the following system of three equations with three unknowns:

\[
\begin{align*}
3x + 2y - z &= 10 \\
-x + 3y + 2z &= 5 \\
x - y - z &= -1
\end{align*}
\]

We can rewrite this system of equations using the following matrices:

\[
\begin{bmatrix}
3 & 2 & -1 \\
-1 & 3 & 2 \\
1 & -1 & -1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix} =
\begin{bmatrix}
10 \\
5 \\
-1
\end{bmatrix}
\]

Using matrix multiplication, the system of equations can then be written as \( AX = B \).

Go through the multiplication to convince yourself that this matrix equation yields the original set of equations.

To simplify the notation, we designate the variables as \( x_1, x_2, x_3 \), and so on. Rewriting the initial set of equations using this notation, we have

\[
\begin{align*}
3x_1 + 2x_2 - x_3 &= 10 \\
-x_1 + 3x_2 + 2x_3 &= 5 \\
x_1 - x_2 - x_3 &= -1
\end{align*}
\]
This set of equations is then represented by the matrix equation \( AX = B \), where \( X \) is the column vector \([x_1, x_2, x_3]^T\). We now present two methods for solving a system of \( N \) equations with \( N \) unknowns.

### 4.2.1 Solution Using the Matrix Inverse

One way to solve a system of equations is by using the matrix inverse. For example, assume that \( A, X, \) and \( B \) are the matrices defined earlier in this section:

\[
A = \begin{bmatrix} 3 & 2 & -1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad B = \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}
\]

Then \( AX = B \). If we premultiply both sides of this matrix equation by \( A^{-1} \), we have \( A^{-1}AX = A^{-1}B \). However, because \( A^{-1}A \) is equal to the identity matrix \( I \), we have \( IX = A^{-1}B \), or \( X = A^{-1}B \). In MATLAB, we can compute this solution using the following command:

\[
X = \text{inv}(A)^*B;
\]

---

**EXAMPLE 4.5:** As an example, we will solve the following system of equations:

\[
3x_1 + 5x_2 = -7 \\
2x_1 - 4x_2 = 10
\]

Type the following MATLAB commands to define \( A \) and \( B \):

\[
A = \begin{bmatrix} 3 & 5 \\ 2 & -4 \end{bmatrix}; \quad B = \begin{bmatrix} -7 \\ 10 \end{bmatrix};
\]

Now solve for \( X \) using the inverse of \( A \):

\[
X = \text{inv}(A)^*B;
\]

MATLAB finds the following solution:

\[
X = \\
\quad \begin{bmatrix} 1.0000 \\ -2.0000 \end{bmatrix}
\]

---

### 4.2.2 Solution Using Matrix Left Division

A better way to solve a system of linear equations is to use the matrix division operator:

\[
X = A\backslash B;
\]

This method produces the solution using Gaussian elimination, without forming the inverse. Using the matrix division operator is more efficient than using the matrix inverse and produces a greater numerical accuracy.
EXAMPLE 4.6: As an example, we will solve the same system of equations used in the previous example:

\[
\begin{align*}
3x_1 + 5x_2 &= -7 \\
2x_1 - 4x_2 &= 10
\end{align*}
\]

However, now solve for \(X\) by using matrix left division:

\[
X = A \backslash B;
\]

Again, MATLAB finds the following solution:

\[
X = \begin{bmatrix} 1 \\ -2 \end{bmatrix}
\]

To confirm that the values of \(X\) do indeed solve each equation, we can multiply \(A\) by \(X\) using the expression \(A \times X\). The result is the column vector \([-7, 10]^T\).

If there is not a unique solution to a system of equations, an error message is displayed. The solution vector may contain values of NaN, \(\infty\), or \(-\infty\), depending on the values of the matrices \(A\) and \(B\).

PRACTICE!

Solve the given system of equations using MATLAB left division and inverse matrices. Check with MATLAB whether each solution solves the system of equations.

SUMMARY

In this chapter, we defined the transpose, the inverse, and the determinant of a matrix. We also defined the computation of a dot product (between two vectors) and a matrix product (between two matrices). Two methods for solving a system of \(N\) equations with \(N\) unknowns using matrix operations were presented. One method used matrix left division, and the other used the inverse of a matrix.

MATLAB SUMMARY

This MATLAB summary lists and briefly describes all of the special characters, commands, and functions that were defined in this chapter.
Special Characters

' indicates a matrix transpose

* indicates matrix multiplication

\ indicates matrix left division

Commands and Functions

det computes the determinant of a matrix

inv computes the inverse of a matrix

KEY TERMS
determinant

dot product

identity matrix

inverse

matrix multiplication

transpose

system of equations

Problems

1. Compute the dot product of the following pairs of vectors by hand, showing your work, and then give two different MATLAB commands to compute the dot product of each pair:
   a. \( \mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} -3 & -2 & 4 \end{bmatrix} \)
   b. \( \mathbf{A} = \begin{bmatrix} 0 & -1 & -4 & -8 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 4 & -2 & -3 & 24 \end{bmatrix} \)

2. Compute the matrix product \( \mathbf{A}^T \mathbf{B} \) of the following pairs of matrices, showing your work:
   a. \( \mathbf{A} = \begin{bmatrix} 12 & 4 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 2 & 12 \end{bmatrix} \)
   b. \( \mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} -2 & 4 \end{bmatrix} \)

3. Given the array \( \mathbf{A} = \begin{bmatrix} -1 & 3 & 4 & 2 \end{bmatrix} \), raise \( \mathbf{A} \) to the second power by array exponentiation. Raise \( \mathbf{A} \) to the second power by matrix exponentiation. Compute by hand and show your work. Give the MATLAB commands for matrix and array exponentiation of \( \mathbf{A} \).

4. Given the array \( \mathbf{A} = \begin{bmatrix} -1 & 3 & 4 \end{bmatrix} \), compute the determinant of \( \mathbf{A} \) by hand. Show your work. Give the MATLAB command for computing the determinant of \( \mathbf{A} \).

5. If \( \mathbf{A} \) is conformable to \( \mathbf{B} \) for addition, then a theorem states that \( (\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T \). Use MATLAB to test this theorem on the following matrices:
   \[
   \begin{bmatrix}
   2 & 12 & -5 \\
   -3 & 0 & -2 \\
   4 & 2 & -1
   \end{bmatrix}
   \]
   \[
   \begin{bmatrix}
   4 & 0 & 12 \\
   2 & 2 & 0 \\
   -6 & 3 & 0
   \end{bmatrix}
   \]

6. Given that matrices \( \mathbf{A}, \mathbf{B}, \) and \( \mathbf{C} \) are conformable for multiplication, then the associative property holds—i.e., \( \mathbf{A} (\mathbf{B} \mathbf{C}) = (\mathbf{A} \mathbf{B}) \mathbf{C} \). Test the associative property using matrices \( \mathbf{A} \) and \( \mathbf{B} \) from Problem 1, along with matrix \( \mathbf{C} \), which is as follows:
   \[
   \begin{bmatrix}
   4 \\
   -3 \\
   0
   \end{bmatrix}
   \]
7. Recall that not all matrices have an inverse. A matrix is singular (i.e., it doesn't have an inverse) iff $|A| = 0$. Test the following matrices using the determinant function to see if each has an inverse:

$$A = \begin{bmatrix} 2 & -1 \\ 4 & 5 \end{bmatrix} \hspace{1cm} B = \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix} \hspace{1cm} C = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 2 & 2 \\ 5 & -4 & 0 \end{bmatrix}$$

If an inverse exists, compute the inverse.

8. Solve the following system of equations using both the matrix left division and the inverse matrix methods:

$$x_1 - x_2 - x_3 - x_4 = 5$$
$$x_1 + 2x_2 + 3x_3 + x_4 = -2$$
$$2x_1 + 2x_3 + 3x_4 = 3$$
$$3x_1 + x_2 + 2x_4 = 1$$

9. Time each method that you used in Problem 4 by using the `clock` function and the `etime` function, which measures elapsed time. Which method is faster?

```matlab
t0 = clock;
...
% code to be timed
...
etime(clock, t0)
```

10. **Single-Voltage-Source Electrical Circuit.** This problem relates to a system of equations generated by the electrical circuit shown in Figure 4.1, which contains a single voltage source and five resistors. The following set of equations defines the currents in this circuit:

$$-V_1 + R_2(i_1 - i_2) + R_4(i_1 - i_3) = 0$$
$$R_2i_2 + R_3(i_2 - i_3) + R_5(i_2 - i_1) = 0$$
$$R_3(i_3 - i_2) + R_3i_2 + R_4(i_3 - i_1) = 0$$

Compute the currents using resistor values ($R_1, R_2, R_3, R_4$) and a voltage value ($V_1$) entered from the keyboard.

![Figure 4.1. Circuit with one voltage source.](image)
11. **Amino Acids.** The amino acids in proteins contain molecules of oxygen (O), carbon (C), nitrogen (N), sulfur (S), and hydrogen (H), as shown in Table 4.1. The molecular weights for oxygen, carbon, nitrogen, sulfur, and hydrogen are as follows:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Oxygen</td>
<td>15.9994</td>
</tr>
<tr>
<td>Carbon</td>
<td>12.011</td>
</tr>
<tr>
<td>Nitrogen</td>
<td>14.00674</td>
</tr>
<tr>
<td>Sulfur</td>
<td>32.066</td>
</tr>
<tr>
<td>Hydrogen</td>
<td>1.00794</td>
</tr>
</tbody>
</table>

a. Write a program in which the user enters the number of oxygen atoms, carbon atoms, nitrogen atoms, sulfur atoms, and hydrogen atoms in an amino acid. Compute and print the corresponding molecular weight. Use a dot product to compute the molecular weight.

b. Write a program that computes the molecular weight of each amino acid in Table 4.1, assuming that the numeric information in this table is contained in a data file named `elements.dat`. Generate a new data file named `weights.dat` that contains the molecular weights of the amino acids. Use matrix multiplication to compute the molecular weights.

### Table 4.1 Amino Acid Molecules

<table>
<thead>
<tr>
<th>AMINO ACID</th>
<th>O</th>
<th>C</th>
<th>N</th>
<th>S</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alanine</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Arginine</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>Asparagine</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Aspartic</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Cysteine</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Glutamic</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Glutamine</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Glycine</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Histidine</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Isoleucine</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>Leucine</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>Lysine</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>Methionine</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Phenylalanine</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>Proline</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Serine</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Threonine</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Tryptophan</td>
<td>2</td>
<td>11</td>
<td>2</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>Valine</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>11</td>
</tr>
</tbody>
</table>
c. Modify the program developed in part (b) so that it also computes and prints the average amino acid molecular weight.

d. Modify the program developed in part (b) so that it also computes and prints the minimum and maximum molecular weights.