Introduction to MATLAB linear algebra computations

MATRIX COMPUTATIONS IN MATLAB

1. Enter the matrix \[
\begin{pmatrix}
1 & 0 & 3 \\
2 & 1 & 1 \\
0 & -1 & 1
\end{pmatrix}
\]
into MATLAB and assign it to the variable A by typing:

\[
\text{>> } A = [1, 0, 3; 2, 1, 1; 0, -1, 1]
\]

Notice that the column entries are separated by commas, rows by semicolons.

Arrays (matrices) are the fundamental MATLAB data type. Numbers are \(1 \times 1\) arrays. The [] is the concatenation operator that joins matrices into a bigger matrix. The command above merged the \(1 \times 1\) matrices into one \(3 \times 3\) matrix.

2. Now use the MATLAB command \texttt{eye} to create a \(3 \times 3\) identity matrix and assign it to the variable I:

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

3. Use [ ] to create the augmented matrix \((A | I)\) and assign it to the variable B.

4. Find the reduced row echelon form of B using \texttt{rref}:

\[
\text{>> } R = \text{rref}(B)
\]

According to what we learned in class, what is the right side of R?

5. Using the parenthesis operator () we can access individual elements of a matrix. Try the following:

\[
\text{>> } R(1,2) \\
\text{>> } R(3,2) \\
\text{>> } R(7,1)
\]

Notice that if you don’t assign the result to a variable, MATLAB always assigns the result by default to the variable \texttt{ans}.

The () operator can also retrieve arbitrary sub-matrices defined by lists of indexes. Do this:

\[
\text{>> } R([1,3],[2,4]) \\
\text{>> } R([2,3],[1,2,3,4,5])
\]

6. We can generate lists containing arithmetic sequences more efficiently using the range operator “:”. Try this:

\[
\text{>> } 1:10
\]

Assign the right hand side of R to a new matrix C, using the range operator. Then compute the matrix product

\[
\text{>> } A*C
\]

Is this what you expected?

7. If you wish to extract a sub matrix that contains an entire row or entire column, you may use the range operator without argument:

\[
\text{>> } R(:, 4:6)
\]
8. What do you think \( R(:,:) \) is?

9. You can create an \( n \times m \) matrix filled with random numbers between 0 and 1 by using \( \text{rand} \). You can create an \( n \times m \) matrix filled with all ones using \( \text{ones} \). If you supply only one argument, you get a square matrix.

\[
\begin{align*}
&\text{>> rand(3)} \\
&\text{>> rand(2, 3)} \\
&\text{>> ones(5)} \\
&\text{>> ones(6,2)}
\end{align*}
\]
Using \( \text{rand} \) and \( \text{ones} \), create a random \( 7 \times 7 \) matrix filled with numbers between -2 and 6.

10. Assign a random \( 3 \times 3 \) matrix to \( A \), a random \( 3 \times 3 \) matrix to \( B \) and compute \( A*B \) and \( B*A \).

Run this experiment several times to convince yourself that a matrix multiplication is not commutative.

Use the up key to reuse previous keyboard input.

11. Define

\[
\begin{pmatrix}
3 & -2 & 3 \\
1 & 3 & 7 \\
5 & 10 & 1
\end{pmatrix}
\quad \text{and} \quad
\begin{pmatrix}
2 \\
1 \\
-1
\end{pmatrix}
\]

Now compute the solution of \( Ax = b \) by

\[
\text{>> inv(A) * b}
\]

(\( \text{inv(A)} \) gives the inverse of the matrix \( A \)),

MATLAB provides a neat shortcut syntax for this type of calculation. Multiplying by the inverse of a matrix is the matrix algebra version of division, so MATLAB lets you “left-divide” the vector \( b \) by \( A \) to get the same answer:

\[
\text{>> A\b}
\]

Note that this second way to solve the system is much more computationally efficient than using the inverse.

12. The matrix division operators work for undetermined systems too! Consider the following undetermined, consistent system:

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
=
\begin{pmatrix}
1 \\
1 \\
1
\end{pmatrix}
\]

You can see that \( x = 1 \) and \( y = 1 \) and \( z \) is a free variable. Solve this system with MATLAB and see what you get.

13. Let’s try to solve an overdetermined system

\[
\begin{pmatrix}
1 & 0 \\
0 & 1 \\
1 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y
\end{pmatrix}
=
\begin{pmatrix}
1 \\
1 \\
1.9
\end{pmatrix}
\]

You can see that there is no solution; nevertheless, if you attempt to solve this in MATLAB using \( \backslash \), you will get a solution. MATLAB gives you the least squares solution here; it is an optimal approximate solution that you will learn more about in section 5.3.

14. You can use \( \text{det} \) to compute the determinant of a square matrix.

15. Using the apostrophe operator you find the transpose of a matrix. '

Exercise: try to discover a \( 2 \times 3 \) matrix \( A \) so that the \( 3 \times 3 \) matrix \( A^T A \) is singular, but the \( 2 \times 2 \) matrix \( AA^T \) is nonsingular.
16. Using the pointwise (or element-by-element) power operator .^ you may raise the elements of a matrix individually to a power. Do the following and explain each operation:

\[
\begin{align*}
\text{>> A} & = \begin{bmatrix} 2, & 1; \\ 0, & 1 \end{bmatrix} \\
\text{>> A}^2 & \\
\text{>> A} .^2 & \\
\end{align*}
\]

For which type of matrix will the two operations produce the same answer?

17. Use \textit{diag} to generate a diagonal matrix with entries provided by a list of diagonal entries:

\[
\text{>> diag([1, 2, 3])}
\]

\textbf{Exercises:} do the following as efficiently as you can.
- Create a \textbf{9×9} diagonal matrix containing the numbers 10 to 19 on the diagonal in order.
- Create a \textbf{12×12} diagonal matrix with the diagonal entries 1,4,9,16,25, etc.

18. \textit{diag} can do more. Using an additional integer \( k \) as input parameter, you can tell MATLAB to shift the diagonal \( k \) up. If \( k \) is negative, the diagonal is shifted down.

\[
\text{>> diag([1,2,3,4],1)}
\]

If \( L \) is a list of numbers of size \( N \), what is the size of \( \text{diag}(L,k) \)?

19. A tridiagonal matrix is a matrix where only the elements on and directly above or below the diagonal are zero. Using what you learned, generate a \textbf{14×14} tridiagonal matrix with all 5s on, all \(-1\)s below and all \(-2\)s above the diagonal \textbf{Do this without typing out lists of elements.}

20. You can change the way MATLAB displays numbers by using the format command. Type \textit{help format} to learn more about this command. Then use it to compute the inverse of \([1, 2; -7,3]\) in terms of fractional entries rather than decimals.