1 Maple

Maple is great for doing simple matrix operations. It even has a built-in pivot command. This is how it works.

Consider the following matrix, from section notes 2:

\[
\begin{bmatrix}
1 & 2 & 3 \\
2 & 3 & 4
\end{bmatrix}
\]

To declare the matrix in Maple, we first use the linalg package:

```maple
with(linalg);
```

and then declare a matrix \( A \) as follows:

```maple
A := <<1,2>|<2,3>|<3,4>>;
```

Notice that in the above notation Maple is taking in columns, such that \(<1,2>\) is the column vector in the first column of \( A \), and so on.

Okay. Let’s say we want to isolate the 1 in row 1, column 1. We can use the pivot command, which takes a matrix, a row and a column:

```maple
pivot(A, 1, 1);
```

results in the following output from Maple:

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

Notice that the element in row 1, column 1 is now “isolated”.

Here we have just called the pivot command, but did not save the output of the command into a variable. If I check the value of the matrix \( A \) (by typing matrix(A) and pressing enter), I will see that its unchanged. So let’s just recall the command again, this time storing the resulting matrix in a matrix \( B \):

```maple
B := pivot(A, 1, 1);
```

In general, Maple does not normalize the isolated value to 1. To fix this, you can normalize using the mulrow command, which takes a matrix, a row, and a normalizing constant to multiply by. For example:

```maple
B := mulrow(B, 1, 1/B[1,1]);
```

would multiply the first row of \( B \) by \( 1/B[1,1] \), which has the effect of normalizing the first row such that \( A[1,1] \) is going to be 1. The resulting matrix is stored in \( B \) (now \( B \) is the normalized matrix) in this example, but you could store it in another matrix (say \( C \)). From here, you can call pivot again on any rows or columns you desire to isolate.

2 Matlab

Unlike Maple, Matlab does not have a built-in pivot command, but it is easy to write a function that will perform the operation:\footnote{Adapted from http://classes.apl.washington.edu/Math407Summer2005/pivot.xml}

```matlab
function B = pivot(A, r, c)
    B = A;
    for i = 1:r
        B(i,:) = B(i,:) / B(i,c);
    end
    for j = 1:size(A,2)
        B(r,j) = B(r,j) - B(r,c) * B(c,j);
    end
end
```

This function takes a matrix \( A \), a row \( r \), and a column \( c \) as inputs, and returns a new matrix \( B \) with the \( r \)-th row isolated from \( A \).
function R = pivot(M, r, c)
    [d, w] = size(M); % Get matrix dimensions
    R = zeros(d, w); % Initialize to appropriate size
    R(r,:) = M(r,:) / M(r,c); % Copy row r, normalizing M(r,c) to 1
    for k = 1:d % For all matrix rows
        if (k ~= r) % Other then r
            R(k,:) = M(k,:) - M(k,c) * R(r,:); % Minus a multiple of normalized row r, making R(k,c)=0
        end
    end
end

With this function defined, we can perform the identical operation as above:
A = [1 2 3; 2 3 4]; B = pivot(A, 1, 1);
which we set B equal to:

<table>
<thead>
<tr>
<th>1  2  3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  -1  -2</td>
</tr>
</tbody>
</table>

3 Mathematica

This is a Mathematica translation of the code above

Pivot[M_,r_,c_] := (sizem = Dimensions[M];
    R = M;
    R[[r]] = M[[r]]/M[[r, c]]; (* normalize row *)
    For[k = 1, k <= sizem[[1]], k++, (* init, check, incr. *)
        R[[k]] = If[k != r, (* condition *)
            M[[k]] - M[[k, c]]*R[[r]], (* if true do this *)
            R[[k]]] (* if false do this *)
    ];
    Return[R] (* return R as fcn out *)
)

With this function defined, we can perform the identical operation as above:
A = {{1, 2, 3}, {2, 3, 4}}; B = Pivot[A, 1, 1];