Pivoting in Maple/Matlab/Mathematica

AM121/ES121

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:

$$\begin{vmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{vmatrix}$$

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

```
with(linalg);
```

and then declare a matrix A as follows:

```
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:

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

results in the following output from Maple:

$$\begin{vmatrix} 1 & 2 & 3 \\ 0 & -1 & -2 \end{vmatrix}$$

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:

```
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:

```
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:[1]

[1] Adapted from http://classes.apl.washington.edu/Math407Summer2005/pivot.xml

```
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,:) ...       % Set them equal to the original matrix
             - 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:`

$$\begin{vmatrix} 1 & 2 & 3 \\ 0 & -1 & -2 \end{vmatrix}$$

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];`