This paper builds on the work of Nichols & Greenhow [2] and Nichols, Gill & Greenhow [3] on the development of online objective questions for mathematics formative and diagnostic testing using Question Mark Perception. Questions in linear algebra exploit the use of random parameters to spectacular effect, whereby the 49 question styles developed so far will produce of the order of $10^{25}$ question realisations seen by the students. Copying will therefore be impossible and it is hoped that, after a few attempts and serious study of the feedback offered, the students will realise that, despite the different numbers displayed, they are actually answering the same question from the algebraic viewpoint. They will, perhaps, gain a deeper understanding of concepts such as linear dependence. Evidence for this conjecture will be collected in the next academic year and reported on.

Central to this randomisation is the use of the MathML mark-up language into which random parameters can be inserted for the question’s stem, key, distracters and feedback. An added advantage is that user font preferences (type, size and colour) can be read from a previously defined cookie and incorporated in the mstyle commands enclosing the specific equation syntax. This usability feature is still under development, but it is worth remarking that much of the work we have done is independent of Perception, or indeed of any CAA package, being standard html/JavaScript. This means that these features can be incorporated into any web page, such as a page of teaching material with ever changing examples and displayed as chosen by the student, for example, in 48-point, white Times Roman font on a blue background, or whatever. Even if randomisation is not required in web-based teaching material, the usual embedded graphic approach to displaying mathematics seems very limited in comparison, perhaps not even fulfilling the “reasonable provision” for disabled/dyslexic students required by the SENDA legislation. Moreover, for web pages of any complexity, editing and organising mathematical content in MathML is likely to prove far easier than managing numerous graphics in the long run. On the downside, at present the students will need to download a (free) MathML viewer from www.dessci.com/en/support/tsn/tsn87.htm

![Fig 1 Numerical input question about notation](image)
but it is hoped that browsers will support MathML natively in the near future.

The question styles typically call several external JavaScript functions held in a brunel LinearAlg.format file that is “included” when Perception runs the question. We have so far developed 17 functions of two types; those that return the value(s) of some mathematical operation, eg. multiplication of two matrices, and those that return the MathML string needed to display the result. All these functions can be called with matrices of arbitrary size; for the display functions, the size of the matrix is used to loop round, each pass concatenating the next bit of the MathML string.

The questions

Presently we have 12 question styles on matrix notation (including writing simultaneous equations and matrix equations and vice versa), 22 on determinants (easy, harder and properties), 6 on inverses and 9 on multiplication (easy and harder). In each case, tested and assumed skills are recorded in the question and outcome metadata that is written to the answer files. The questions are also tagged according to their A level module (from the next academic year this will be FP3) and level of difficulty (easy, intermediate and hard).

The general appearance of a realisation of numerical input question about notation is shown in Fig 1.

This question illustrates how versatile the method is; for another realisation, all the elements will change of course, and so will the shape of the matrix. Incorrect input will trigger a restatement of the question with the required element highlighted, followed by an explanation of what the subscript notation $C_{6,4}$ means. A common error might be to input $C_{4,6}$; this can be detected and trigger a specific feedback message (we call this responsive numerical input).

Clearly it would be quite easy to defeat competent students by the sheer numerical difficulty of eg. multiplying two large matrices together, or finding a large determinant. From the question setter’s point of view, it is well worth setting such questions in a general way and then restricting the size at the final stage. That way, most of the coding for a general $n \times m$ matrix can be recycled for similar questions with matrices up to eg. 4x4. Another possibility is shown in Fig 2.

Find the values of $x$ and $y$ such that

$x = \begin{bmatrix} -2 & -3 & 2 & 3 \\ -4 & -1 & 7 & -1 \\ -6 & -5 & 5 & 7 \\ -5 & 3 & 2 & 7 & -4 \end{bmatrix}$

$y = \begin{bmatrix} 2 & -1 & -3 & 2 & 3 \\ 1 & 3 & -3 & -5 & 1 \\ 4 & 5 & -2 & 3 & 5 \\ 3 & 3 & -2 & -1 & 2 \end{bmatrix}$

$x = \begin{bmatrix} 33 & 33 & 31 & 21 & 69 & 13 \\ -10 & 0 & 27 & -5 & 10 \\ -31 & 7 & 0 & 9 & -81 \\ 17 & -11 & 11 & -10 & y \end{bmatrix}$

$y = \begin{bmatrix} 28 & 24 & 16 & 65 & 46 \\ 55 & 8 & 16 & 56 & 27 \\ 46 & -11 & 5 & 40 & 25 \end{bmatrix}$

Fig 2 Multiplying two large matrices

Here, virtually everything that can change does change; Fig 3 is another realisation of the same question style:

Find the values of $x$ and $y$ such that

$x = \begin{bmatrix} 2 & 4 & 6 & 5 & -6 & 2 & 6 \\ -2 & 6 & 1 & 3 & 6 & 4 \\ 1 & 4 & 6 & 3 & 6 & -6 \\ 6 & -1 & -1 & -2 & 2 & 3 \end{bmatrix}$

$y = \begin{bmatrix} 3 & -1 & -4 & 2 & 3 \\ -5 & -2 & 3 & 1 & -5 \\ -4 & 1 & 2 & -4 & 3 \\ 1 & 5 & -2 & 4 & 5 \\ 0 & 1 & 2 & -3 & 1 \\ -1 & -3 & 3 & -5 & 2 \end{bmatrix}$

$y = \begin{bmatrix} 47 & 25 & 6 & -84 & 35 & -12 \\ 15 & 45 & -64 & x & 60 & 24 \\ 12 & 51 & -23 & 51 & 17 & 90 \\ y & -33 & 40 & -10 & -32 & 27 \end{bmatrix}$

Fig 3 Another multiplication

This question is, potentially, a useful teaching tool. Students can check their methodology for doing the multiplication against a few of the results in the answer matrix before inputting the required two elements. Whether students will do this, and indeed, how students will attempt the questions in general will be interesting to observe. The capabilities of calculators (to say nothing of symbolic manipulators!) to allow students to get questions correct without any understanding may mean that tests will need to be invigilated. On the other hand, careful question design can deter calculator use, as follows:

Fig 4 Question design helps to deter calculator use

It is worth noting that randomisation can change the algebraic structure of a problem. For example, multiplying a matrix containing an $x$ by one containing
a y could result in either linear or nonlinear simultaneous equations in x and y. Any student doing these questions can be assumed to be able to handle this. However, if the above question is modified to include two ps, the determinant will be linear/quadratic in p if the ps do/do not occur in the same row or column. The could be confusing and related questions where the student is required to solve e.g., det(A) = 0 for p might need complex numbers. The important thing is to know what the question is actually testing and which skills are being assumed.

The following example in Fig 5 seeks to capitalise on the student’s engagement with the question via the feedback.

Find the determinant of:

\[
\begin{bmatrix}
-3 & 4 & 2 & -2 \\
-1 & -4 & 1 & 1 \\
3 & -4 & -1 & -1 \\
\end{bmatrix}
\times
\begin{bmatrix}
5 & -2 & 2 \\
3 & 4 & -1 \\
2 & 4 & 3 \\
\end{bmatrix}
\]

Here an incorrect answer will produce very extensive feedback that seeks to remind the student to think about the problem before rushing in. One almost hopes that the student will have rushed in and found the question hard or impossible, and hence be more receptive to the feedback example shown in Fig 6.

Note that the absolute size of the determinant can easily become large. There is, of course, nothing wrong with this, but for the input specified, care is needed when choosing the parameters’ ranges so that the student’s calculator stays in exact mode (rather than switching to scientific mode).

A general problem with matrix questions is that many steps are often needed, e.g., for finding an inverse matrix. It may be better to break the question down into smaller parts, as shown in Fig 7.
Given the matrix $A$, find the cofactor of $A_{4,1}$:

\[
\begin{bmatrix}
-8 & 8 & 9 & 8 \\
-7 & 8 & 3 & -9 \\
5 & -7 & 6 & -4 \\
-9 & -3 & -2 & 1
\end{bmatrix}
\]

**Fig 7** Question broken down into smaller parts

Alternatively one could give some or all of the cofactors etc, or to ask for a student to check Fred’s working in the following “hot line” question:

Fred is trying to find the inverse of the given matrix $A$, but he may have made a mistake.

\[
A = \begin{bmatrix} 6 & -4 & -3 \\ 4 & 1 & 3 \end{bmatrix}
\]

\[\text{det}(A) = -798\]

The cofactor matrix is given by

\[
\begin{bmatrix}
51 & 48 & 60 \\
-52 & 2 & -64
\end{bmatrix}
\]

Therefore, the adjoint matrix is given by

\[
\begin{bmatrix}
51 & -19 & -52 \\
-48 & -86 & 2 \\
-60 & -38 & -64
\end{bmatrix}
\]

Hence, we get $A^{-1} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$

**Please input the line number where the mistake is, or input 0 if there is no mistake.**

The error is in line: __________

**Fig 8** Student checks working in hot line question

Hot line questions work by randomly choosing a “scenario” to display; any one of a number of mistakes can occur in any line, with subsequent lines following on properly from the mistake, not correcting it and not making further mistakes either. The number of scenarios is limited only by the number of plausible mistakes invented by the question setter. (Incidentally, the name “Fred” has been chosen randomly from a list reflecting the sex and ethnicity balance of British 16-25 year-olds.)

**The functions**

We here show some of the shorter functions used in the above questions and elsewhere (a complete set is available from martin.greenhow@brunel.ac.uk, see also [1]). A fairly standard function shows the syntax involved in multiplying matrices in JavaScript:

```javascript
function multimatrix(Mtrix1, Mtrix2) {
    sizeA = MatrixSize(Mtrix1);
    NrowA = sizeA[0];
    NcolumnA = sizeA[1];
    sizeB = MatrixSize(Mtrix2);
    NrowB = sizeB[0];
    NcolumnB = sizeB[1];
    if (NcolumnA == NrowB) {
        D = getRandomMatrix( NrowA, NcolumnB, 0, 1, 0);
        for (row = 1; row <= NrowA; row++) {
            for (column = 1; column <= NcolumnB; column++) {
                result = 0;
                for (i = 1; i <= NrowB; i++) {
                    result += Mtrix1[row][i] * Mtrix2[i][column];
                }
                D[row][column] = result;
            }
        }
        return D;
    } else {
        alert(“The 2 matrices are not conformable! (“+NrowA+"x"+NcolumnA+" and "+NrowB+"x"+NcolumnB+”)");
    }
    return D;
}
```

Students should probably never see the “alert” message but it is helpful to include this when debugging questions where matrix sizes are calculated or randomised.

The following example shows how functions can call others, including themselves (recursively):

```javascript
function determinant(Rmtrix) {
    var number = 0;
    var i;
    var Nrow;
    var size = new Array();
    size = MatrixSize(Rmtrix);
    Nrow = size[0];
    var number;
    if (Nrow == 1) {
        number = Rmtrix[1][1];
    } else {
        for (i = 1; i <= Nrow; i++) {
            number += Rmtrix[1][i] * Math.pow(-1, (i + 1)) * determinant(dominor(Rmtrix, 1, i));
        }
    }
    return number;
}
```

Finally, the following shows how to generate the MathML string needed to display a matrix of any size, possibly with a highlighted element, row or column. This removes much of the burden from a question setter and, given its rather complex structure, can be used as a black box. However, readers should note the important (and transferable) idea of concatenating the MathML string to be returned, and an example of this is shown in bolded font.
function displayMatrix_coloured(Rmtrix,a,b) {
    inside = "";
    inside = displayinside(Rmtrix,a,b);
    start = "<mtr><mtd"></mtd>";
    end = "</mtr>
    return start + inside + end;
}

function displayinside(Rmtrix,a,b) {
    inside = "";
    number = MatrixSize(Rmtrix);
    rowNumber = number[0];
    columnNumber = number[1];
    rowelements = new Array();
    therow = new Array();
    matrixrows = new Array();
    if (a > 0 && b > 0) {
        for (k=1 ; k<=rowNumber ; k++) {
            for (i=1; i<=columnNumber; i++) {
                if (k==a && i==b) {
                    rowelements[k] += `<mtd><mi color=RED
background=yellow>" + Rmtrix[k][i] + 
"</mtd>`;
                } else {
                    rowelements[k] += 
`<mtd><mi>" + Rmtrix[k][i] + 
"</mtd>`;
                }
            }
        }
    } else if (a < 0 && b < 0) {
        for (k=1 ; k<=rowNumber ; k++) {
            for (i=1; i<=columnNumber; i++) {
                if (k==(-a) || i==(-b)) {
                    rowelements[k] += `<mtd><mi
background=yellow>" + Rmtrix[k][i] + 
"</mtd>`;
                } else {
                    rowelements[k] += 
`<mtd><mi>" + Rmtrix[k][i] + 
"</mtd>`;
                }
            }
        }
    } else {
        for (k=1 ; k<=rowNumber ; k++) {
            for (i=1; i<=columnNumber; i++) {
                if (k==a || i==b) {
                    rowelements[k] += `<mtd><mi
background=yellow>" + Rmtrix[k][i] + 
"</mtd>`;
                } else {
                    rowelements[k] += 
`<mtd><mi>" + Rmtrix[k][i] + 
"</mtd>`;
                }
            }
        }
    }
    for (p=1 ; p<=rowNumber ; p++) {
        therow[p] = `<mtr>` + rowelements[p] + 
"</mtr>
    }
    for (f=1 ; f<=rowNumber ; f++) {
        matrixrows[f] = therow[f];
    }
    for (j=1; j<=rowNumber; j++) {
        inside += matrixrows[j];
    }
    return inside;
}

Future work

Whilst these questions (and some on vectors not shown here) will be useful in a wide range of courses involving basic linear algebra, there are areas (such as eigenproblems) that are typically included in level 1 linear algebra modules and will therefore need to be covered. Future questions in these areas will often need to be “reverse engineered”, for example by having the code calculate and then present a matrix with a numerically simple set of eigenvalues or (normalised) eigenvectors. Another area that looks feasible is to ask for the next step in a common procedure such as Gaussian elimination and back substitution or the factorisation of a matrix into lower/upper triangular matrices. Collaboration on such tasks would be welcomed.

References