Exportable technologies: MathML and SVG objects for CAA and web content

EXPORTABLE TECHNOLOGIES: MATHML AND SVG OBJECTS FOR CAA AND WEB CONTENT

Edward Ellis, Martin Greenhow and Justin Hatt

Exportable Technologies: MathML and SVG Objects for CAA and Web Content

Edward Ellis, Martin Greenhow and Justin Hatt
Department of Mathematical Sciences
Brunel University
mapgege@brunel.ac.uk
mastmmg@brunel.ac.uk
Justin.Hatt@brunel.ac.uk

Abstract

The aim of this short paper is to provide an update on our experiences with using Mathematical Mark-up Language (MathML) and Scalable Vector Graphics (SVG) within "Mathletics" – a suite of mathematics and statistics objective question styles written within Perception's QML language/Javascript. We refer here to question **style** to stress that we author according to the pedagogic and algebraic structure of a questions' content; random parameters are chosen at runtime and included within all elements of the question and feedback, including the plain text source for MathML and SVG. This results in each style having thousands, or even millions, of **realisations** seen by the users. Much of what we have developed exists in template files that contain functions called by any question style within the database; such functions are therefore independent of any particular web-based system (we user Perception), indeed, ordinary web pages. We reported on some of these functions at the last CAA Conference (Baruah, Ellis, Gill and Greenhow 2005) whilst basic concepts and terminology for MathML and SVG are introduced by Ellis (2005). It should also be noted that the user's choice of font colours & sizes, and background colour, are all incorporated within the MathML and SVG content. This means that equations and diagrams will be **accessible** to those requiring larger/differently-coloured versions of the content's default options.

This paper further exploits:

The use of tables of arbitrary length to display an algorithm presentation MathML. We here show how MathML can be generated effectively by our "display" functions and incorporated into new question types

SVG diagrams. We show examples of the use of SVG to produce dynamic diagrams and charts that accurately reflect the question's random parameters choice or statistical data. The SVG library of functions produce "objects", such as lines, text boxes, circles, etc that can be called by other functions that build up super-objects such as decision boxes, bar charts, pie charts, Venn

diagrams etc. These are then concatenated within the question, to produce, for example, a flow chart.

SVG graph plotter. Although MathML plotter applets exist, these are generally not open code and therefore cannot be tailored to meet the pedagogic needs of the question and/or feedback. We have therefore developed a graph plotter that gives full control of how any Javascript-defined function is to be plotted, including shading, labelling, highlighting of points of interest such as maxima etc. The utility of such a plotter will be demonstrated within questions.

Content MathML. The test example presented at the last CAA conference has been developed into actual questions.

Another aim of this paper is to include an introduction to our functions. We believe that this will prove useful to a wide range of disciplines that contain mathematical or graphical content. We show how such functions are exploited in an ordinary web page and speculate on the structure of a teacher/lecturer's web page containing printable versions of all our question styles (over 1000) with solutions for each student's realisation. The plan is that the teacher/lecturer will preview a question, select what he/she wants and build up a problem sheet; finally printing will produce, say, 30 realisations of the problem sheet (and matched solutions) for use in traditional teaching settings.

Example 1: The Use of Tables

Figure 1 displays parts of the feedback for a bubble sort question. Note the alert box has been triggered since the input string, although of the correct format, has incorrect length (known from the randomised length of the list of random values, between 1 and 20, given in the question). This is an extension of the checking described in the companion paper at this conference by Baruah et al (2006). The essence of the algorithm is encapsulated in the sequence of feedback tables, where cell colouring is used to show the considered pairs before and after swaps and completed cells (green). The coding for building these tables this is not long and completely general, although for more extensive data sets, the feedback can take too long to render.

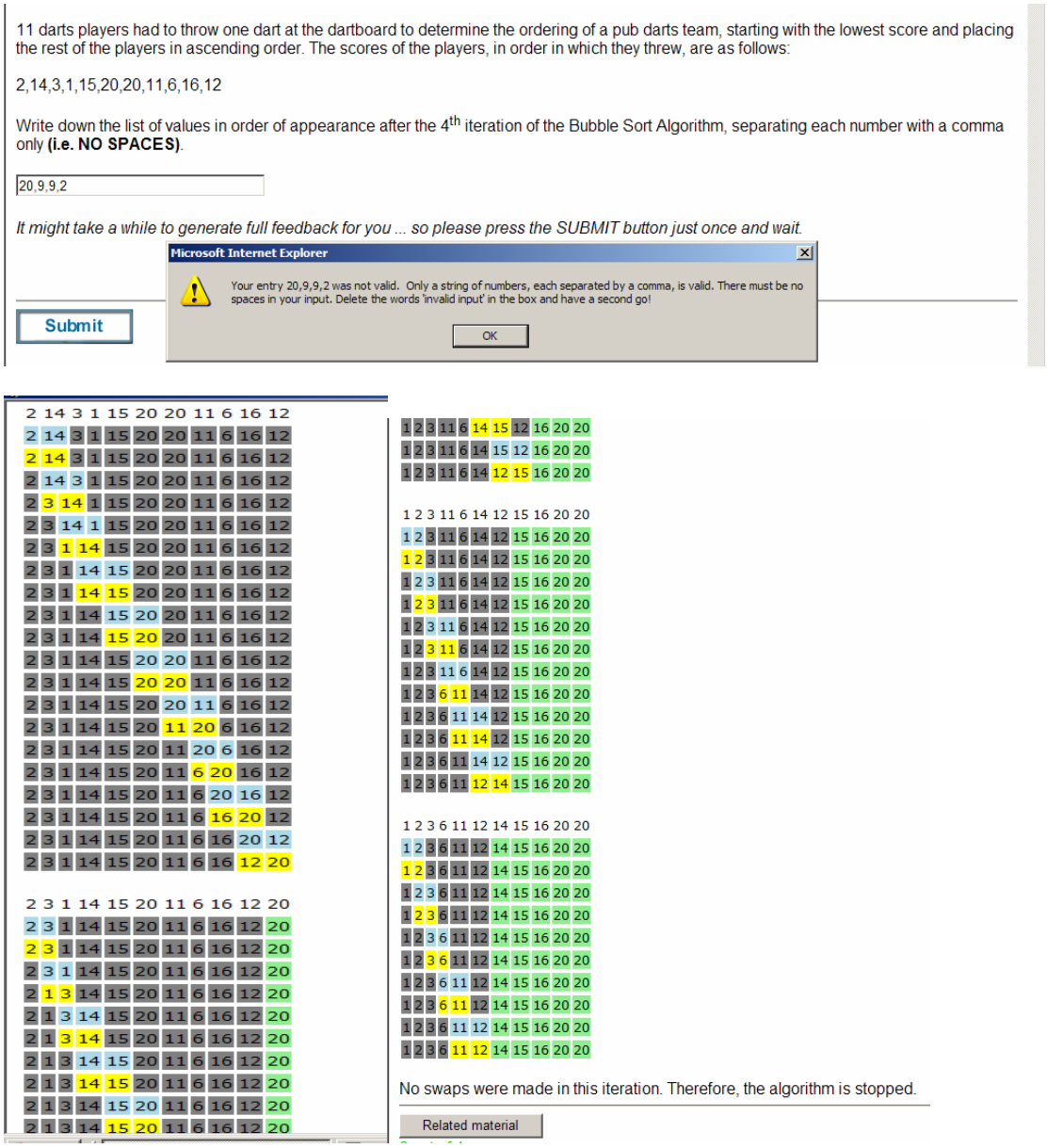

Figure 1. Checking the input validity for a string match question and parts of the feedback tables showing the bubble sort algorithm in action.

Example 2: Javascript and Presentation MathML

By considering a question in linear algebra (LU factorisation) we demonstrate the utility of function to perform calculations and present the results in MathML. The question type is interesting since the required element positions (and question wording) change with each realisation – we call this *positional numerical input* (PNI). Although quite extensive coding is required, the initial set up that guarantees integer values for the answers is quite terse:

```
LT = getrandomtriangularmatrix( random, random, -5, 5, 0,0,1,0 );
//creates the lower triangular matrix.
UT = getrandomtriangularmatrix( random, random, -5, 5, 0,1,0,0 );
//creates the upper triangular matrix.
Bigmatrix = multimatrix(LT,UT);
//multiplies LT and UT together.
```

Here we have essentially started with the answer matrices LT and UT, calling the getrandomtriangularmatrix function:

```
// Function getrandomtriangularmatrix(Nrow,Ncolumn,min,max,allowzero,LU,diagonalones)
will create a matrix of size Nrow x Ncolumn
// elements from min to max and if allowzero !=0 then zero is allowed and if LU=0 then a
Lower matrix is created and
// if LU=1 then an Upper matrix is created
function getrandomtriangularmatrix(Nrow,Ncolumn,min,max,allowzero,LU,diagonalones){
if (LU != 0 && LU != 1){alert("getrandomtriangularmatrix called illegally with LU = "+LU+". This
should be either 0 for a lower triangular matrix, or 1 for an upper triangular matrix")};
var Randomatrix = new Array
Randomatrix[0] = new Array;
for (k = 1 ; k <= Nrow ; k++) {Randomatrix[k] = new Array;}
for (var i = 1; i <= Nrow; i++)
        {for (var j = 1; j <= Ncolumn; j++){
        if (LU == 1){if(i <= j){number = displayarray(1,min,max,allowzero)}else{number = 0};}
          else{
          if (i >= j){number = displayarray(1,min,max,allowzero)}else{number = 0};}
                Randomatrix[i][j] = number;}
        }
if(diagonalones == 1){for(k = 1 ; k <= Nrow ; k++){Randomatrix[k][k] = 1;}}
return (Randomatrix);
}
```

The matrix on the right-hand side (Bigmatrix) is generated by the multimatrix function, i.e. it is correct matrix arithmetic according to this "**reverse engineering**" approach, typical of these questions where one needs to keep control of the complexity of the arithmetic. Certain elements are then overwritten as e.g. $U_{1,3}$ etc before the display matrices are processed by a displaymatrix function that returns the presentation MathML required for rendering. This function (too long to present here) loops round column and

rows to concatenate a returned MathML string that is rendered by the WebEQ viewer applet. Relevant parts of the (shortened) code are:

```
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] + "</mi></mtd>";}
                else {rowelements[k] += "<mtd><mi>" + Rmtrix[k][i] + "</mi></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;
```

We see here the highlighting capabilities of MathML (although figure 2 uses a slightly different technique).

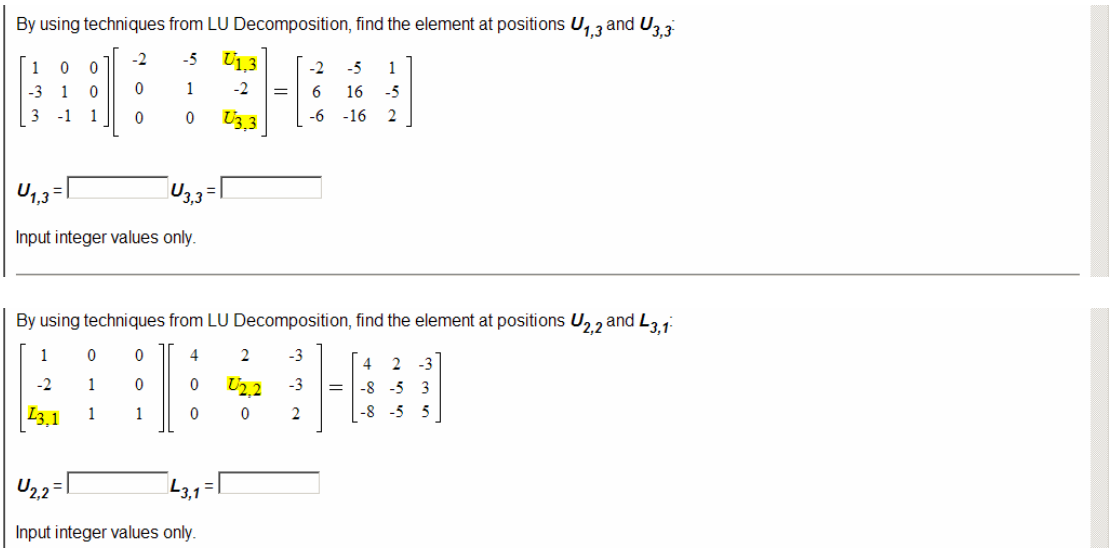

Figure 2. Two realisations of a positional numerical input question.

Examples Using Scalable Vector Graphics

This section looks at the potential of scalable vector graphics (SVG) to enhance the question design or feedback utility. The use of both geometrically accurate diagrams and schematics for mechanics questions has been reported by Gill and Greenhow (2006). Here we look at possibilities in other areas. Figure 3 shows an obvious application, namely geometry.

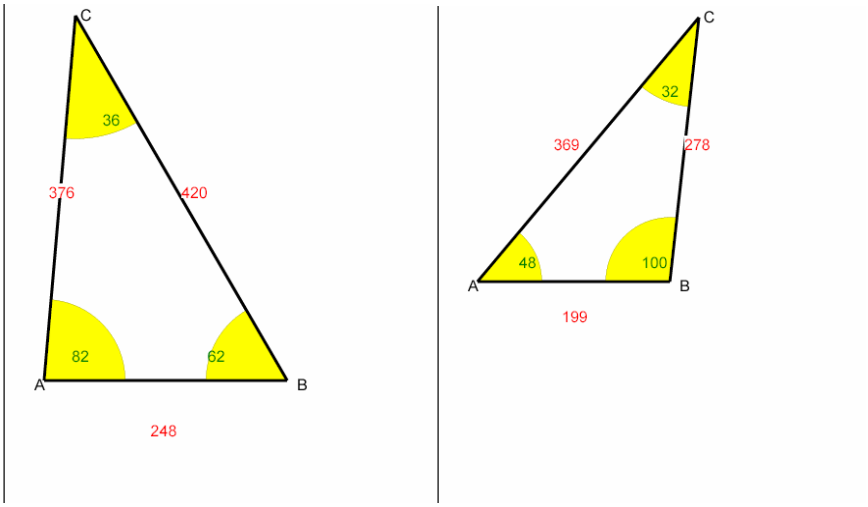

Figure 3. Two realisations of a triangle display.

The coding behind the called function, SVG_triangle, returns the SVG plain text code for rendering by the SVG viewer web page plug-in is quite instructive, but too long to present here. However it is worth noting that the arguments for all coordinates, lengths of sides, angles, labels are all listed, but could be empty strings. This avoids writing many similar functions to handle display where different input data is given. Figure 3 show all arguments, whereas a real question would, for example, omit one of the sides. The SVG_triangle calls functions returning "*atomic components*", such as lines, text boxes, sectors (shown with a yellow background in figure 3) which handle the accessibility features, such as colours and font sizes. Geometric objects such as lines are rescaled according to the font size (both length and line thickness) and use the user's choice of font colour by default. A helper function angle_from_xy(x,y) is also called. It returns the polar angle of point (x,y), needed since Javascripts' arctan function returns the principal value. Finally note that the order of concatenation of the SVG string can be important, see Ellis (2005); in figure 3, the required string order is yellow sectors, then angles, then lines of triangle, then lengths of sides (with opaque background boxes reading the background colour of the page).

Another example of the efficacy of SVG is given in figure 4. The student is asked to apply the first-fit algorithm to the data (the table length, names and weights are randomised). The algorithm produces a shown matrix, but it would be quite natural in class to draw this as a diagram. Dropping the random weights and names into the string-generation loop allows this to happen, producing an accurate and meaningful diagram in the feedback.

In Dropmore County First School's annual tug-of-war competition, 25 contestants have to be placed into groups by the order of their introduction by Reverend J. P. Smythe-Jones-Hamilton, the Vicar. The order of introduction of the contestants, along with their respective weights, are listed in the following chart:

Nadia	Claudia	Julie	Fatima	Lottie	Dennis	Kay	Sandeep	
99	94	115	67	73	71	64	72	
Elizabeth	Alan	Sarah	Asfia	Alan	Clare	Wendy	Stephie	Ingrid
101	88	65	98	71	91	108	84	68
George	Sophie	Lisa	Ajay	Alexandria	Paul	Mary	Daniel	
67	95	74	81	65	86	68	75	

Each contestant's weight is measured in lbs.

The Vicar wants to allocate each group so that its total weight is, at most, 300 lbs (regardless of how many contestants each group has). Calculate the spare capacity of group 6.

[ ] lbs.

Group 1	99	94	67	
Group 2	115	73	71	
Group 3	64	72	101	
Group 4	88	65	98	
Group 5	71	91	108	
Group 6	84	68	67	74
Group 7	95	81	65	
Group 8	86	68	75	

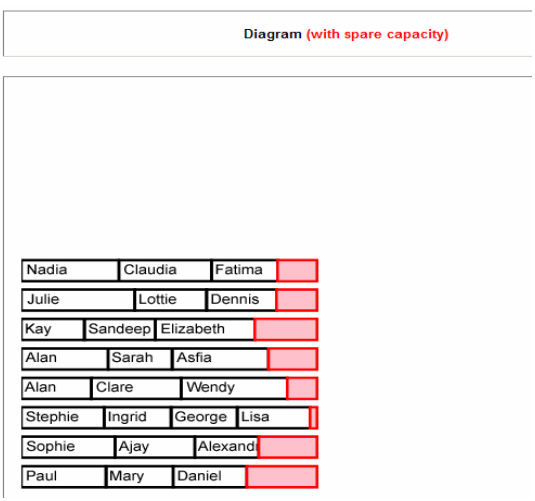

Figure 4. Question stem and components of the feedback for a first-fit question. The SVG diagram accurately displays the data in an effective way.

SVG Graph Plotter

Other developments include an SVG graph plotter. Plotting graphs in SVG has a number of advantages over using either images, or Java Applets. Advantages compared to images have already been covered. The advantages compared to Java Applets is that one can literally draw over the top of the graph. This can prove invaluable in some case. For example, highlighting the roots, turning points, or other significant features of functions.

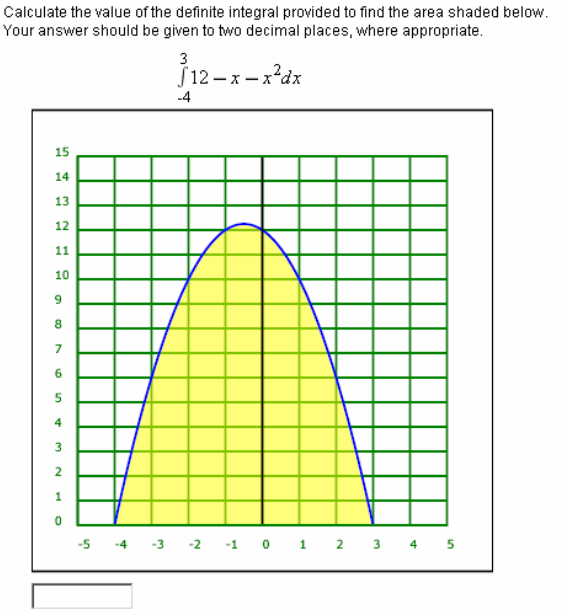

Figure 5. Question stem of an integration question. The SVG graph plots the function to be integrated, according to the random parameters in the integrand and integration limits.

MathML Input (Content MathML)

Content MathML is exploited using a MathML Input question type previously described (Baruah, Ellis, Gill and Greenhow 2005). Entry of free-form Mathematical expressions allows question authors to move away from Multiple Choice questions styles. A great deal of useful information can be obtained from students in this fashion. For example, a question on partial fractions is able to determine the number of fractions the student entered, and the contents of each numerator and/or denominator. Such information can be used to provide targeted feedback.

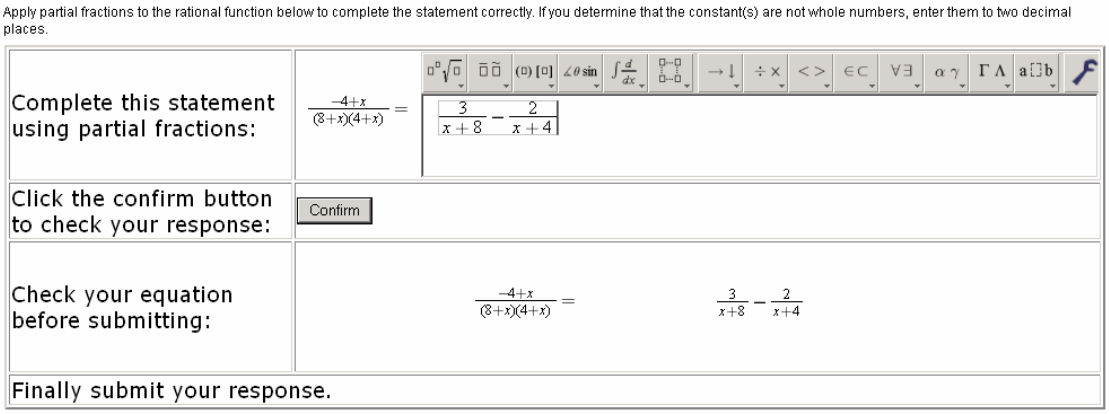

Figure 6. Question stem of a partial fractions question. Use of Content MathML allows detailed analysis of a student's response, without the disadvantages of multiple choice questions types.

A Short List of Available Functions

All of the JavaScript functions can be placed within one of four classes.

1) Generate internal representations of mathematical entities.
2) Manipulate existing internal representations.
3) Convert internal representations into useful alternative representations.
4) Support functions, known as glue.

Examples:

All random generators are in class (1). Examples of these include:

a) rndGraphPoly(degree). This function returns an array representing a polynomial. The polynomial has the property that all turning points exist in the square where x exists [-1,1] and y exists [-1,1]. It is often used in collaboration with the SVG graph plotter.

b) displayarray(num_elements,min,max,allowzero) returns a JavaScript array. That array holds ‘num_elements’ numbers, each in the range [min,max], with the option of excluding zero from that range.

Class (2) is mainly occupied by functions that perform calculations. Examples include:

a) addpolynomial(coeffs1,n,coeffs2,m). This function takes two arrays representing polynomials as arguments. It then returns a new array that represents the sum of the first two arrays.
b) custRound(x,places) rounds the number ‘x’ to the number of decimal places given by places.

Every MathML and SVG generating functions fit in class (3). Example are:

a) displaymatrix(Rmtrix) which returns the presentation MathML representation of the two-dimensional array provided as an argument.

b) SVG_triangle() which takes many arguments. It generates an SVG representation of a triangle, details of which are specified by its arguments. Figure 3 was created using this function.

Accessibility functions and other functions fit in class (4). For Example:

a) femalename(i) returns a female first name from an ethnically balance set of names.

b) getFgColor() retrieves the current foreground setting, stored in the cookie.

A more extensive list will be made available via the MSOR Centre website by the summer of 2006.

Web Page Implementation of the Functions

It is important to stress that all of the above can be implemented in any web based system or indeed, ordinary web pages, such as that shown in figure 6. We believe that extracting the questions' contents to such web pages will be useful for teachers/lecturers who are not able or do not want to use a full CAA system. Whilst marking functionality and answer file writing (and hence analysis) is lost, there are practical advantages to paper-based objective exercise sheets, not least that students can show their workings in the blank spaces to the right of the questions and hand them in.

The anatomy of the web page is quite straightforward: functions are included within script tags in the head, whilst the button, accessibility and credits at the top right of the screen and content is included with a series of question functions in the body (no processing function is needed). Thus a teacher can alter font sizes and colours before printing and randomisation features in question content, including MathML and SVG, is retained.

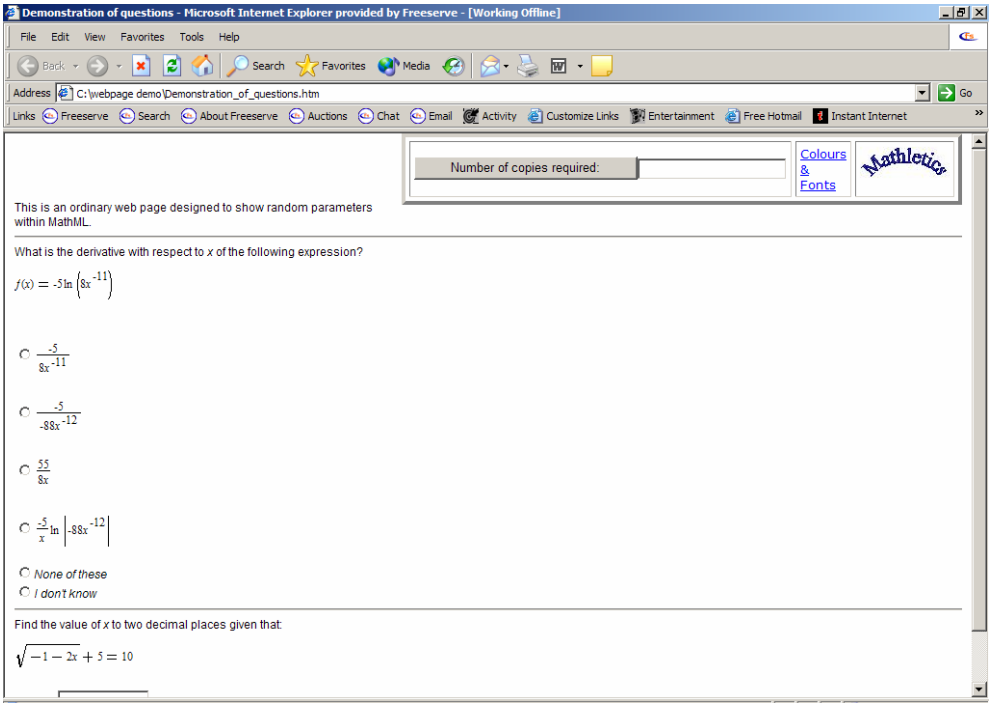

Figure 7. Implementation of functions and MathML in an ordinary web page. The "Number of copies required:" button prints out this problem sheet, reloads it thereby giving questions with new random parameters, prints again etc giving the required number of copies and, separately, numbered answer sheets (planned development).

References

Baruah, N. Gill, M and Greenhow, M 2006 Issues with setting online objective mathematics questions and testing their efficacy *Proc 10th CAA Conf, Loughborough, July*. http://www.caaconference.com/

Ellis, E. 2005 An Introduction to MathML, SVG and JavaScript. MSOR CAA Series. http://mathstore.ac.uk/articles/maths-caa-series/dec2005/

Ellis, E., Baruah, N., Gill, M., Greenhow, M. 2005 Recent developments in setting objective tests in mathematics using QM Perception *Proc 9th CAA Conference, Loughborough, July* http://www.caaconference.com

Gill, M. & Greenhow, M. 2006, Computer-Aided Assessment in Mechanics: what can we do; what can we learn; how far can we go? *Proc IMA Conf Mathematical Education of Engineers, Loughborough, April.*