Coastal Dynamics I: Maple TA drawing questions

Question setup and standard grading code

Introduction

A drawing question in Maple TA follows the following principle: the student selects points on a canvas with a set of axis and grid lines and through these points a polygon is automatically drawn. By means of a JavaScript code that retrieves information from the central Maple TA server (which can be accessed only by the Maple TA operators, i.e. Kasja de Jong) on which an application (run.app.js) is located, the information of the student's drawing is collected by the local Maple TA server, which can be accessed by the course instructor. Using a HTML algorithm, certain parameters that the run.app.js uses can be set to the desired values to design the questions. Then finally a HTML grading code is used to use the information retrieved by the JavaScript code and use this to come to a grade for the student.

Let's go through this one step at a time. In the first chapter of this document a step-by-step manual is given on how to set-up a drawing question in Maple TA on both the trial- and the exam server. In chapter two a more detailed manual is given for the different ways to implement the grading code in a standard fashion.

For more information or specific questions, you can contact Matthijs Buijs at M.Buijs@tudelft.nl.

Set-up drawing questions

First of all, it is important to go to the parent class to create a new question. For the trial exam this can be done by going to the *Maple TA: Question Bank* on the Coastal Dynamics I Brightspace page and for the exam server this can be done by going to https://mapletae.tudelft.nl:8443/mapleta/login/login.do, log in and then going to the parent class.

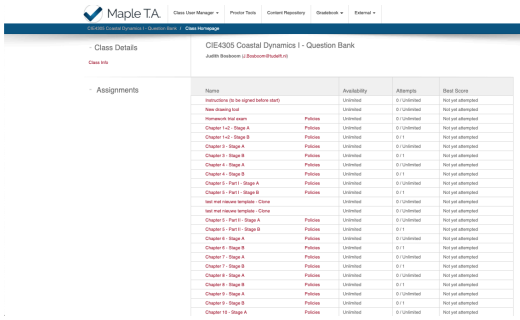


Figure 1: Screenshot trial server

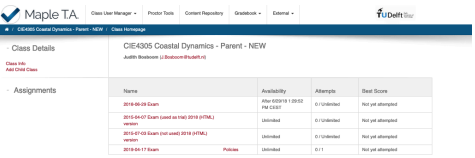


Figure 2: Screenshot exam server

The next step to take is to go to the *Content Repository,* then under *Current Class* go to *Questions*. For the trial server go to *CD1 Questions per chapter* and then to the folder in which you want to create a new question (in this case Chapter 8A or 8B).
For the exam server go to *CD1 Exams* and then to the folder in which you want to create a new question (for a new exam in the following fashion: *year, month, type of question, (specific question)*. e.g. 201904 Tidal(US)).

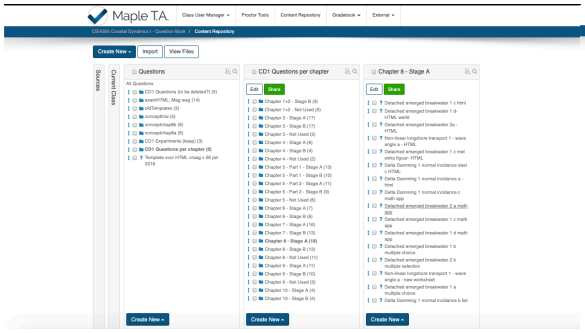


Figure 3: Screenshot trial server

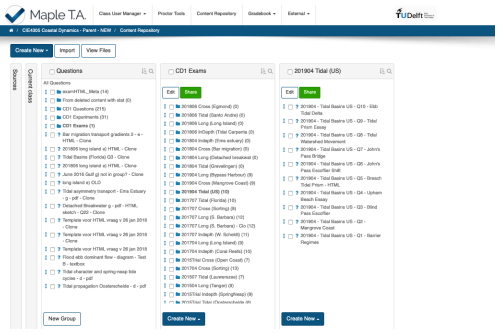
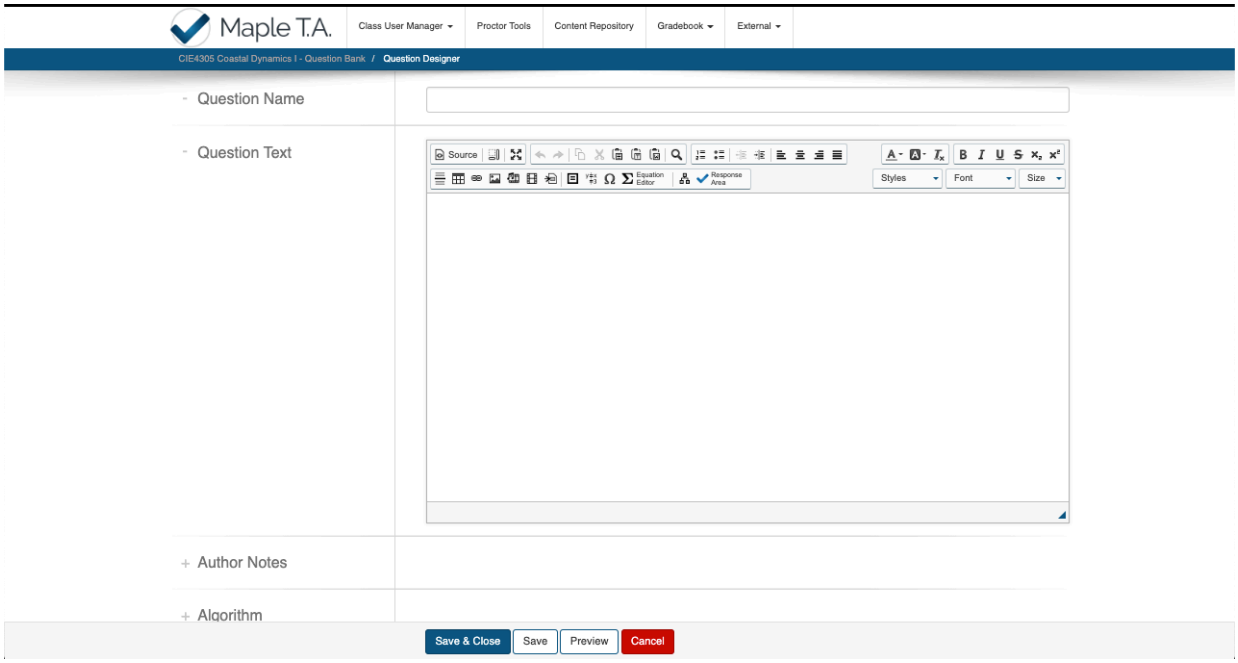


Figure 4: Screenshot exam server

From this point on most of the steps will be similar for both the trial and the exam server, so the most steps will be for the trial and the exam server.

Now click on *Create New* at the bottom of the folder in which you want to create the question and select *Question/Text*. The new question looks as follows:



A *Question Name* is required. For the trial server, this can best be done by giving a description of the question with *HTML* at the end, e.g. *Detached Emerged Breakwater 2a – HTML*. For the exam server a more standard way is needed, here the question name is defined as such: *year, month, type of question, specific question, question number, question description.* Also, here end the question with '-HTML' at the end to make it easy to spot the drawing questions. An example is*: 201904 – Tidal Basins US – Q5 – Breach Tidal Prism – HTML.*

In *Question Text,* as one can imagine, the question is described. For the trial server, the entire question is written down and for the exam server only the question itself and not the lead up to the question is written down. After the question has been written down, click on *Response Area* (this has a blue mark in front of it). In the figure below, one can see the different options given in *Edit Response Area*.

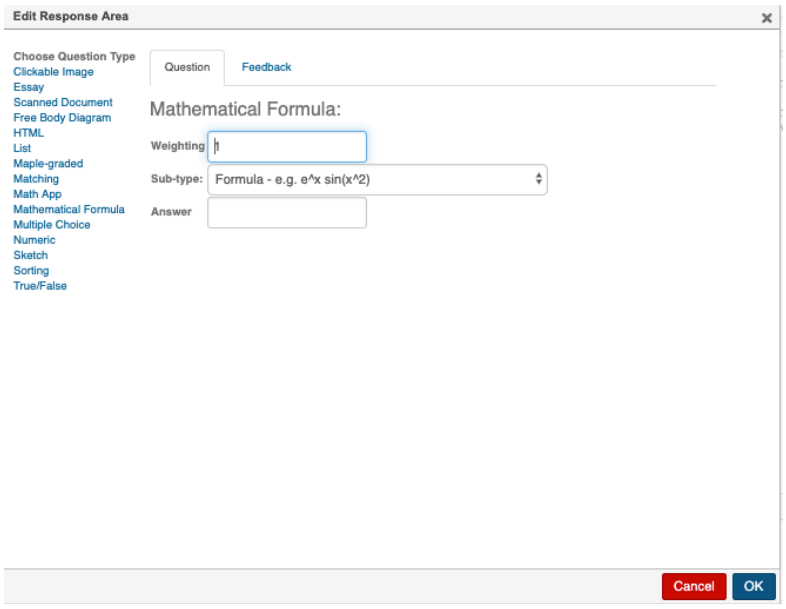


Figure 5: Edit Response Area window

For the drawing questions, select *HTML*. In the window that is now shown, one can see several things. A *Weighting, Answer, Grading Code, Question HTML, Question CSS and Question JavaScript* section exists.

- *Weighting* can be set to *1* at all times
- *Answer* can be set to *"$anspoints"* at all times
- *Grading Code* is the HTML code that evaluates the student's response and this will be discussed later in this chapter and more in depth in the Chapter on the standard grading.
- *Question HTML* gives information to the browser on how to present the drawing canvas and the standard HTML code for this can be found in Appendix A: Question HTML (identical for trial and exam server)
- *Question CSS* translates the information from the *Question HTML.* A standard code can be found in Appendix B: Question CSS (identical for trial and exam server)
- *Question JavaScript* retrieves all the information from the central Maple TA server, where the *run_app.js* is installed. Here it is important to discriminate between the trial server and the exam server. Depending on the type of drawing question, one can choose between two standard JavaScript codes. The difference between these will be described later in this chapter. In Appendix C: Question JavaScript for both the trial and the exam server the scripts can be found.

By clicking on *OK,* the question can be completed. The following screenshot shows the *Algorithm* part of the question. As mentioned, here the specific characteristics of the question are defined.

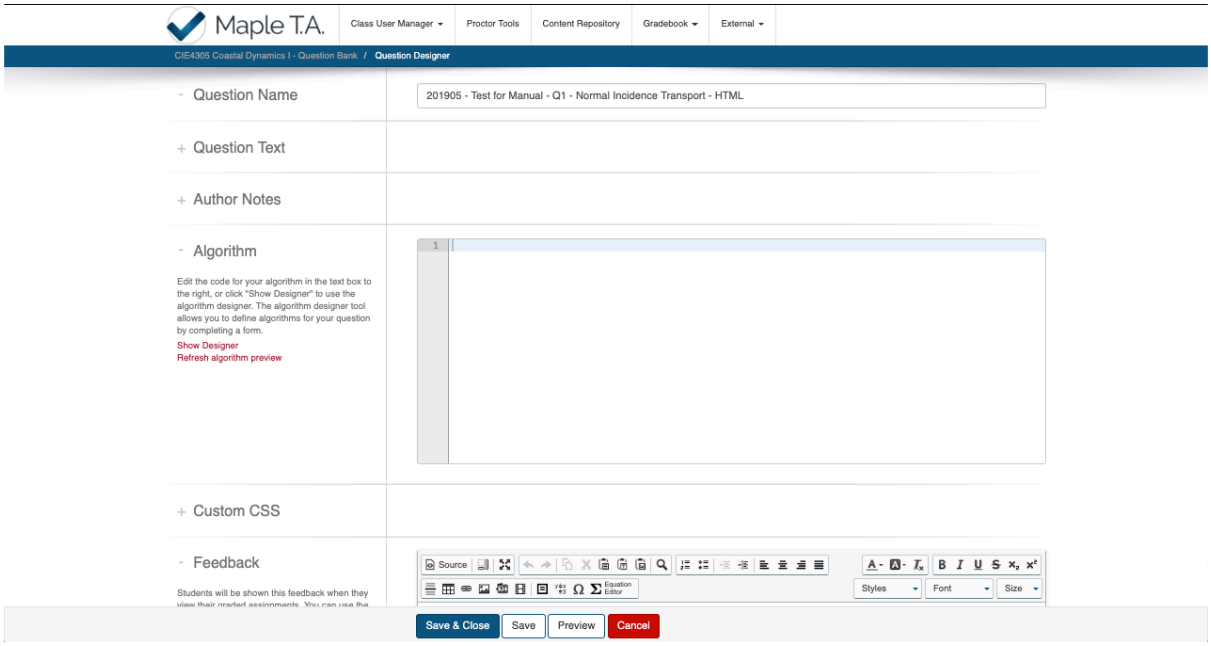


Figure 6: Algorithm of question

The algorithm consists of several parts that can be adjusted as such to the specific question. In Appendix D: Algorithm, an example of an algorithm can be found, which can be adjusted to the question that you want to design.

This algorithm consists of the definition of the correct answer ‘$answerpoints’ and the points which are checked by the grading code ‘$antw’. Also, in the algorithm the axes and such are defined, but this will speak for itself once you get started with it.

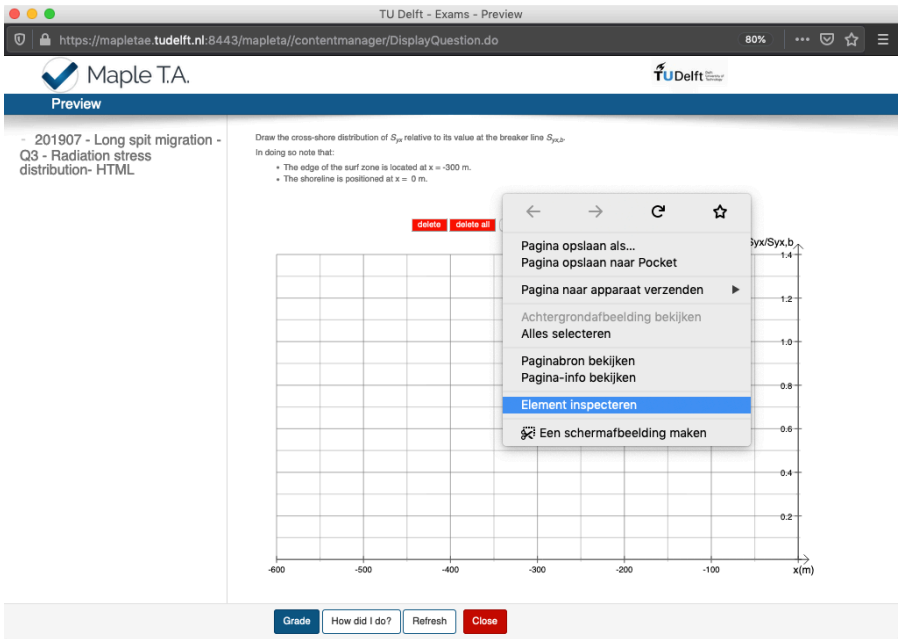
The example that is shown in Appendix D contains all the different functionalities that have been used during this course. Changing the values of the different parts stands to reason and can be quickly evaluated by clicking ‘Preview’ and seeing how it works in practice.

Standard grading code

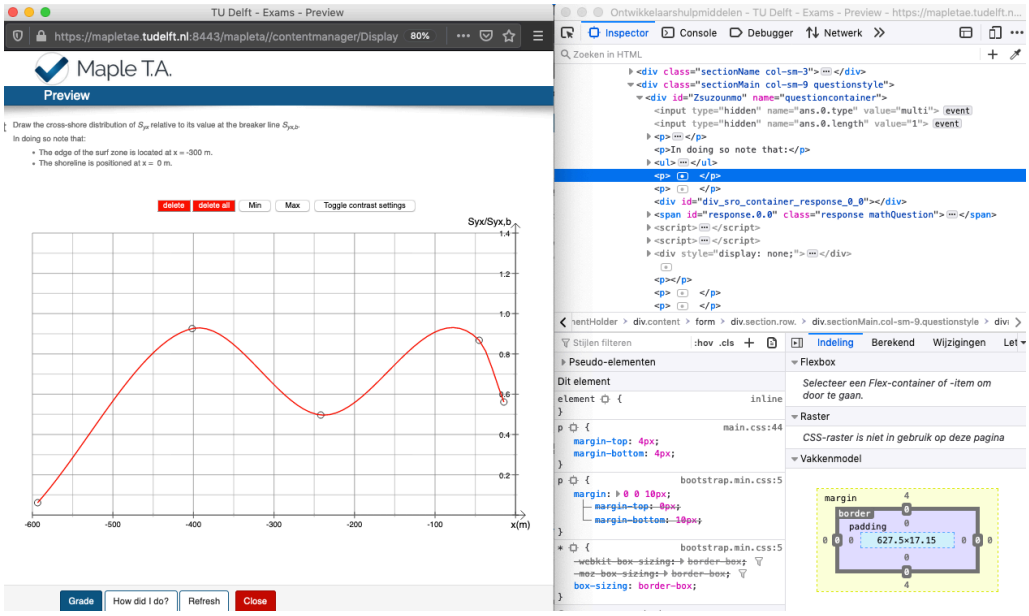
By clicking the button ‘HTML’ and going to the ‘Grading Code’, one can start to write an appropriate grading code for the designed question.

The grading code is written in html language and it can be implemented in the computer program Maple. As described in the next, the x and y values that are generated by the spline generator can be called from your web browser as such.

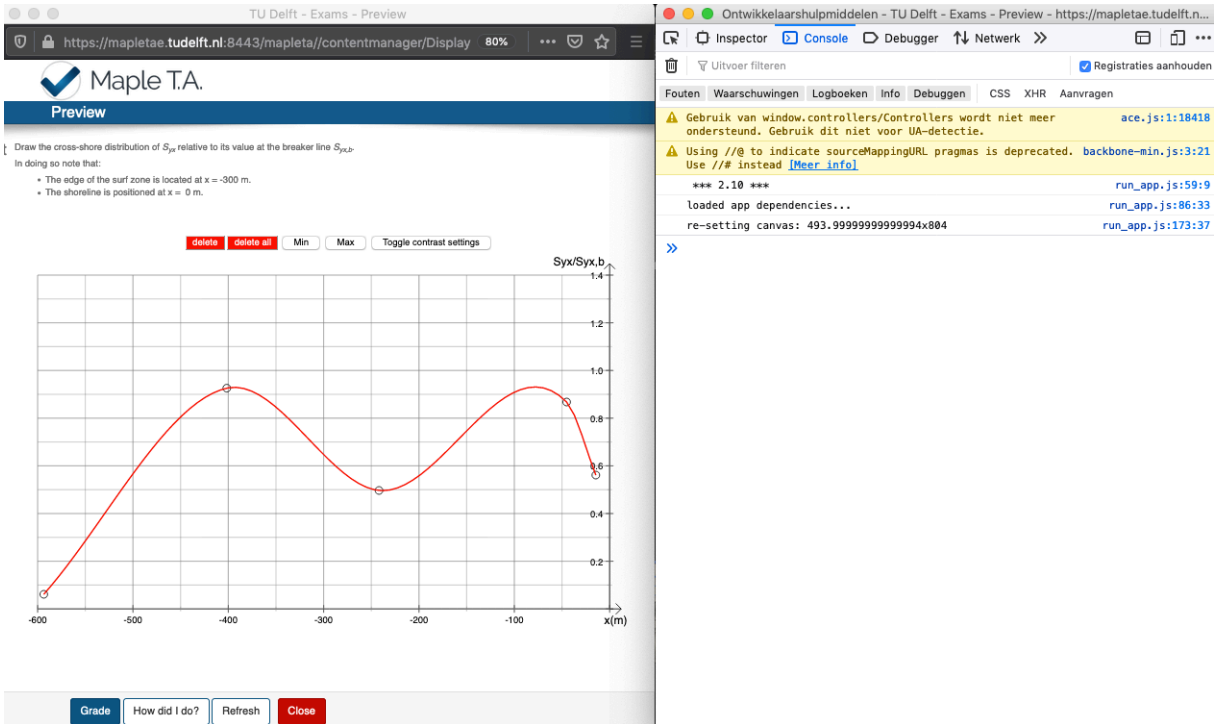
1. Right click your mouse next to the grid and select ‘Inspect Element’ (here in Dutch)



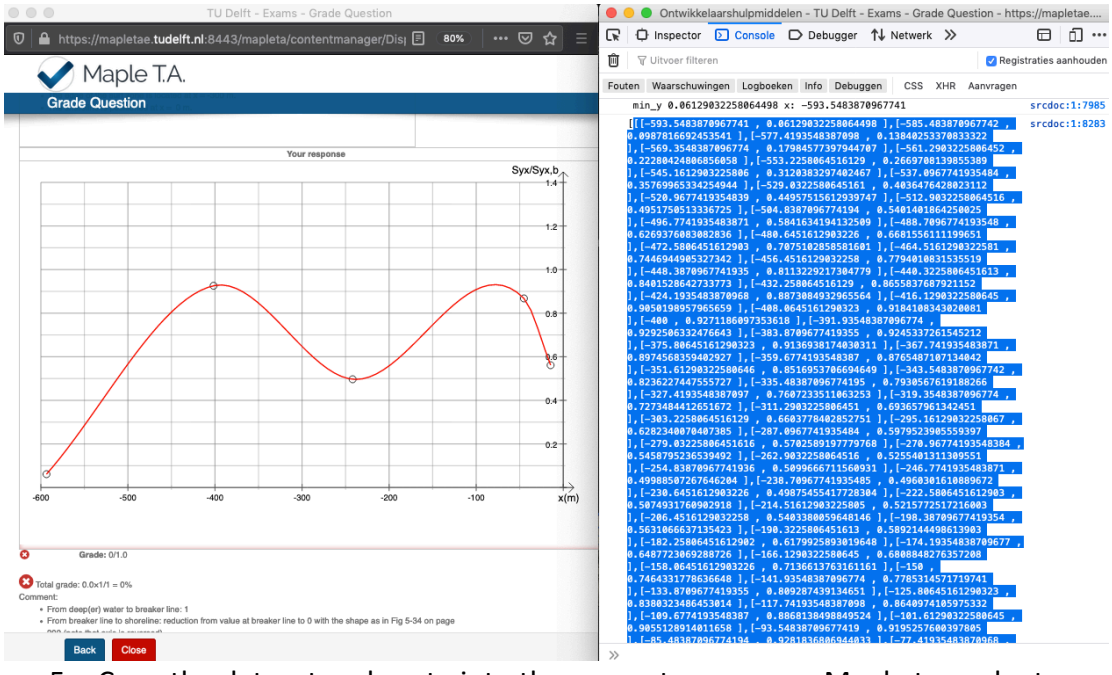
2. Select points on the grid and see a spline get drawn through them. Next to this screen one can see the ‘Inspector’.



3. Go to ‘Console’ and select ‘Keep registrations’. Now, one can select ‘Grade’ on the left screen.



4. Now one can see the responses from the script that has been run. The response is given in the as a list with the following layout: [$RESPONSE1, $RESPONSE2, $RESPONSE3, $RESPONSE4, $RESPONSE5]. Here $REPONSE is given in the following form: [[x1,y1], [x2,y2], [x3,y3], ….,[xi,yi]].



5. Copy the dataset and paste into the computer program Maple to evaluate you grading code.

The grading code consists of multiple sections:

- Calling the values of the drawing (1)
- Calling the global maximum and minimum values (2)
- Defining the starting grade, error and assigning the values of the spline (3)
- Calculating the error of the spline compared some defined values (4)
- Defining the error of the maximum and minimum in x and y direction (5)

```
curv:= $RESPONSE[1]: (1)
maxi := $RESPONSE[4];(2)
mini := $RESPONSE[5];
grade := 0; erro := 0; spl := CurveFitting[Spline](curv, x); (3)
for pt in $antw1 do erro := erro+abs(pt[2]-(eval(spl, x = pt[1]))) end do; (4)
errox := abs(maxi[1][2]-check2[1][1])+abs(mini[1][2]-check2[2][1]);
erroy := abs(maxi[1][1]-check2[1][2])+abs(mini[1][1]-check2[2][2]);(5)
if erro <= 400
then grade:=grade+0.33 end if; (5)
```

This is the first steps into setting up the grading code. Some more options are listed here:

1. Check the slope to be positive or negative

```
if is(eval(diff(spl, x), x = -800.0001), positive)
then grade:=grade+0.33 end if;
```

A double differential can be calculated, but if the existence of a double differential is wanted, it is advised to compare the value of the slope at two nearby x-values with each other. The second derivative is not a fault-proof system, since the spline consists of many different polynomials which are not continuous necessarily.

2. Check whether the slope is steep enough

```
if is(abs(eval(diff(spl, x), x = -800.0001)) >.25)
then grade:=grade+0.33 end if;
```

3. Check the total area under the spline

xvalues := curv[1 .. nops(curv), 1]:	*Make list of xvalues from curv(list)*
yvalues := curv[1 .. nops(curv), 2]:	*Make list of yvalues from curv(list)*
y_array := convert(yvalues, Array):	*Convert list into array*
y_array_abs := abs(y_array):	*Get absolute values from y_array*
y_array_abs_list := convert(y_array_abs, list):	*Convert the absolute values of y back to a list*
L := zip(`[]`, xvalues, y_array_abs_list):	*Combine two lists*
spl_abs := CurveFitting[Spline](L, x):	*Fit a curve through the new data set*
spl_int := abs(int(spl, x = -2.9 .. 2.9)):	*Take integral of original spline*
spl_abs_int := abs(int(spl_abs, x = -2.9 .. 2.9)):	*Take integral of the absolute spline*
ratio_int := spl_int/spl_abs_int:	*Take ratio of the two integrals*
threshold:=0.20:	*Set an adequate threshold*
if is(ratio_int<threshold) then grade := grade+1/3	*Determine if the threshold is not exceeded and the grade is deserved*

The total grading code should have the following format:

```
curv:= $RESPONSE[1]:
maxi := $RESPONSE[4];
mini := $RESPONSE[5];
grade := 0; erro := 0; spl := CurveFitting[Spline](curv, x);
for pt in $antw1 do erro := erro+abs(pt[2]-(eval(spl, x = pt[1]))) end do;
errox := abs(maxi[1][2]-check2[1][1])+abs(mini[1][2]-check2[2][1]);
erroy := abs(maxi[1][1]-check2[1][2])+abs(mini[1][1]-check2[2][2]);
if erro <= 400
then grade:=grade+0.33 end if;
grade
```

4. The presence or absence of second derivatives can be best done by comparing the derivative at one x-value with the derivative at another x-value.

```
if is(abs(eval(diff(spl, x), x = -200.0001))>abs(eval(diff(spl, x), x = -100.0001)))
then grade:=grade+0.25
```

Tips and important notes

Think about what you want the students to answer. First design the question and the according grading code as such that your ‘correct’ answer is actually graded as correct.

Next step is to play with the values and locations of your points and define some local minima and maxima. See if the drawing is still graded correctly each time when it should be. When no correct answer is obtained when it should be, be sure to change the required accuracy of the grading code. If you are able to obtain a wrong answer by trying a few times, the students will definitely be having problems.

Next try to think of possible faulty thinking by students and apply this logic to the drawings. See what grades the students will get and see if you find that the partial grading applied is fair and if not, think of a proper division of the points.

Another technique is to just randomly pick points and see if you are able to collect points. If this is the case, it is a wise decision to add some extra requirements for each of the partial grades to reduce this random effect. Be aware that adding too many extra requirements might reduce the extra benefit of partial grading.

In the Feedback section of the question it is wise to explain the way the question is graded. By giving each of the sections a percentage that is not the same each time, it is easy to see where the question made a mistake and whether or not the grading code performed well.

Lastly, let some of your peers try out your question and see if they have any strange outcomes. This should reduce any bias in the set of possible outcomes that you have come up with.

Appendix A: Question HTML

```
<div style="text-align:center">
      <input type="button" style="color: white; background-color: red;" value="delete"
id="dd" />
      <input type="button" style="color: white; background-color: red;" value="delete all"
id="ddall" />
      <input type="button" style="color: black;" value="Min" id="mm" />
      <input type="button" style="color: black;" value="Max" id="mmax" />
      <input type="button" style="color: black;" value="Toggle contrast settings" id="cont" />
      <div id="contrast" style="display: none; text-align:left">
         Minor lines: <input type="range" style="width: 200px; " id="minorR" min="0"
max="9" value=""+minor_grid_lines.lineColor*10>
         Major lines: <input type="range" style="width: 200px; " id="majorR" min="0"
max="9" value=""+major_grid_lines.lineColor*10>
      </div>
      </div>
     <canvas id="myCanvas" resize></canvas>
```

Appendix B: Question CSS

```
canvas[resize] {
   width: $canvasWidth ;
   height: $canvasHeight;
}
```

Appendix C: Question JavaScript

Trial server

Trial: Short version

```
var teachermode = $teachermode;
	/* axis definition. [ x_min, x_max , y_min, y_max] style. */
	var axes = $axes;
	/* pixel steps between lines for spline drawing, 10 is sufficient */
	var draw_steps = 10;
	var canvasDef = $canvasDef;
	var major_grid_lines = $major_grid_lines;
	var minor_grid_lines = $minor_grid_lines;
	var axis_defintion = $axis_defintion;
	var backgroundlines = $backgroundlines;
	var errormessages ="";
	var AnswerStr = "";
	var gradebook = false;

		/* canvas size in gradebook is 1/2 of the normal "student view" canvas */
	var gradebookScale = 2;



jQuery.getScript('/mapleta/web/Masterclass/Public_Html/run_app.js', function(){




	});

	function initialize(interactiveMode){
	   /*Called when the question is being initialized.
interactiveMode:	if it is true, interaction is allowed on question.
*/
	   /*Your code starts from here:*/
	   gradebook = !interactiveMode;
	   if (gradebook){
	      \$( "#dd" ).remove();
	      \$( "#ddall" ).remove();
	      \$( "#mm" ).remove();
	      \$( "#mmax" ).remove();
	      \$( "#cont" ).remove();
	      \$( "#minorR" ).remove();
	      \$( "#majorR" ).remove();
	      \$( "#contrast" ).remove();
	      \$( "#xmin" ).remove();
	      \$( "#xmax" ).remove();
	      \$( "#ymin" ).remove();
```

```
            \$( "#ymax" ).remove();
            \$( "#kk" ).remove();
            \$( "#ymax" ).remove();
            }
        };

        function setFeedback(response, answer){
            /*called when response or answer is going to be rendered.
response:       student's response of question.
answer:         correct answer of question.*/

            if (response == "No answer" && answer == null) {
                /* not yet attempted, not in the gradebook */
                var temp = [];
                run(temp, 1);

            } else if (answer == null) {
                run(response, 2);


            } else if (answer != null) {
                /* attempted, in gradebook */
                run(answer, 3);


            }

        };

        function getResponse(){
            /*called when grade button is clicked,
to retrieve back student's response.*/
            /*Your code starts from here:*/
            console.log(AnswerStr);
            return AnswerStr;
        };
```

Trial: Long version

```
var teachermode = $teachermode;
        jQuery.getScript('/mapleta/web/Masterclass/Public_Html/run_app.js', function(){




        });
        /* axis definition. [ x_min, x_max , y_min, y_max] style. */
        var axes = $axes;
        /* pixel steps between lines for spline drawing, 10 is sufficient */
        var draw_steps = 10;
        var canvasDef = $canvasDef;
        var major_grid_lines = $major_grid_lines;
        var minor_grid_lines = $minor_grid_lines;
        var axis_defintion = $axis_defintion;
        var backgroundlines = $backgroundlines;
        var errormessages ="";
        var AnswerStr = "";
        var gradebook = false;

                /* canvas size in gradebook is 1/2 of the normal "student view" canvas */
        var gradebookScale = 2;
        var deltax = "10";
        var deltay = "1";
        var precision = 0.05;


                        jQuery.getScript('/mapleta/web/Masterclass/Public_Html/run_app.js',
function(){});



        function initialize(interactiveMode){
           /*Called when the question is being initialized.
interactiveMode: if it is true, interaction is allowed on question.
*/
           /*Your code starts from here:*/
           gradebook = !interactiveMode;
           if (gradebook){
              \$( "#dd" ).remove();
              \$( "#ddall" ).remove();
              \$( "#mm" ).remove();
              \$( "#mmax" ).remove();
              \$( "#cont" ).remove();
              \$( "#minorR" ).remove();
              \$( "#majorR" ).remove();
```

```
            \$( "#contrast" ).remove();
            \$( "#xmin" ).remove();
            \$( "#xmax" ).remove();
            \$( "#ymin" ).remove();
            \$( "#ymax" ).remove();
            \$( "#kk" ).remove();
            \$( "#ymax" ).remove();
            }
        };

        function setFeedback(response, answer){
            /*called when response or answer is going to be rendered.
response: student's response of question.
answer: correct answer of question.*/

            if (response == "No answer" && answer == null) {
                /* not yet attempted, not in the gradebook */
                var temp = [];
                run(temp, 1);

            } else if (answer == null) {
                run(response, 2);


            } else if (answer != null) {
                /* attempted, in gradebook */
                run(answer, 3);


            }

        };

        function prevent_ofb_left(value){
            if (value == 0) {
                return 0;
            }  else {
                return value - 1;
            }
        };

        function prevent_ofb_right(value, length){
            if (value < length) {
                return value + 1;
            }  else {
                return value ;
            }
```

```
        };

        function getResponse(){
            /*called when grade button is clicked,
to retrieve back student's response.*/
            /*Your code starts from here:*/

            var res =  JSON.parse(AnswerStr);
            var response_edited = res[0];
            var pathsPointsfitsX = [];
            var pathsPointsfitsY = [];
            for (i = 0 ; i < response_edited.length ; i++) {
               pathsPointsfitsX.push(response_edited[i][0]);
               pathsPointsfitsY.push(response_edited[i][1]);
            }
            var mySpline = new MonotonicCubicSpline(pathsPointsfitsX, pathsPointsfitsY);

            var max_y = Math.max.apply(null, pathsPointsfitsY);
            var min_y = Math.min.apply(null, pathsPointsfitsY);
            length = pathsPointsfitsX.length;

            val_pos_max = pathsPointsfitsY.indexOf(max_y);
            val_pos_min = pathsPointsfitsY.indexOf(min_y);

            console.log(val_pos_max);
            console.log(val_pos_min);

            var max_x = pathsPointsfitsX[val_pos_max];
            var min_x = pathsPointsfitsX[val_pos_min];

            for (x_t = prevent_ofb_left(pathsPointsfitsX[val_pos_min]);x_t <=
prevent_ofb_right(pathsPointsfitsX[val_pos_min], length); x_t = x_t+precision  ){
               if ( mySpline.interpolate(x_t) < min_y ) {
                  min_y = mySpline.interpolate(x_t);
                  min_x = x_t;
               }
            }

            for (x_t = prevent_ofb_left(pathsPointsfitsX[val_pos_max]);x_t <=
prevent_ofb_right(pathsPointsfitsX[val_pos_max], length); x_t = x_t+precision  ){
               if ( mySpline.interpolate(x_t) > max_y ) {
                  max_y = mySpline.interpolate(x_t);
                  max_x = x_t;
               }
            }

            console.log(" max_y "+ String(max_y) + " x: " + String(max_x));
```

```
            console.log(" min_y "+ String(min_y) + " x: " + String(min_x));

            AnswerStr = AnswerStr.slice(0, -1);
            AnswerStr = AnswerStr + ", [[" + String(max_y) + " , " + String(max_x) + " ]] , [[" +
String(min_y) + " , " + String(min_x) + "]]]";
            console.log(AnswerStr);
            return AnswerStr;

        };
```

Exam server

Exam: Short version

```
var teachermode = $teachermode;
        /* axis definition. [ x_min, x_max , y_min, y_max] style. */
        var axes = $axes;
        /* pixel steps between lines for spline drawing, 10 is sufficient */
        var draw_steps = 10;
        var canvasDef = $canvasDef;
        var major_grid_lines = $major_grid_lines;
        var minor_grid_lines = $minor_grid_lines;
        var axis_defintion = $axis_defintion;
        var backgroundlines = $backgroundlines;
        var errormessages ="";
        var AnswerStr = "";
        var gradebook = false;


jQuery.getScript('/mapleta/web/Cie4305005/Public_Html/run_app.js', function(){




        });

        function initialize(interactiveMode){
           /*Called when the question is being initialized.
interactiveMode:       if it is true, interaction is allowed on question.
*/
           /*Your code starts from here:*/
           gradebook = !interactiveMode;
           if (gradebook){
              \$( "#dd" ).remove();
              \$( "#ddall" ).remove();
              \$( "#mm" ).remove();
              \$( "#mmax" ).remove();
              \$( "#cont" ).remove();
              \$( "#minorR" ).remove();
              \$( "#majorR" ).remove();
              \$( "#contrast" ).remove();
              \$( "#xmin" ).remove();
              \$( "#xmax" ).remove();
              \$( "#ymin" ).remove();
              \$( "#ymax" ).remove();
              \$( "#kk" ).remove();
              \$( "#ymax" ).remove();
              }
```

```
        };

        function setFeedback(response, answer){
            /*called when response or answer is going to be rendered.
response:       student's response of question.
answer:         correct answer of question.*/

            if (response == "No answer" && answer == null) {
               /* not yet attempted, not in the gradebook */
               var temp = [];
               run(temp, 1);

            } else if (answer == null) {
               run(response, 2);


            } else if (answer != null) {
               /* attempted, in gradebook */
               run(answer, 3);


            }

        };

        function getResponse(){
            /*called when grade button is clicked,
to retrieve back student's response.*/
            /*Your code starts from here:*/
            console.log(AnswerStr);
            return AnswerStr;
        };
```

Exam: Long version

```
var teachermode = $teachermode;
        jQuery.getScript('/mapleta/web/Cie4305005/Public_Html/run_app.js', function(){




        });
        /* axis definition. [ x_min, x_max , y_min, y_max] style. */
        var axes = $axes;
        /* pixel steps between lines for spline drawing, 10 is sufficient */
        var draw_steps = 10;
        var canvasDef = $canvasDef;
        var major_grid_lines = $major_grid_lines;
        var minor_grid_lines = $minor_grid_lines;
        var axis_defintion = $axis_defintion;
        var backgroundlines = $backgroundlines;
        var errormessages ="";
        var AnswerStr = "";
        var gradebook = false;

                /* canvas size in gradebook is 1/2 of the normal "student view" canvas */
        var gradebookScale = 2;
        var deltax = "10";
        var deltay = "1";
        var precision = 0.05;


                        jQuery.getScript('/mapleta/web/Cie4305005/Public_Html/run_app.js',
function(){});



        function initialize(interactiveMode){
           /*Called when the question is being initialized.
interactiveMode: if it is true, interaction is allowed on question.
*/
           /*Your code starts from here:*/
           gradebook = !interactiveMode;
           if (gradebook){
              \$( "#dd" ).remove();
              \$( "#ddall" ).remove();
              \$( "#mm" ).remove();
              \$( "#mmax" ).remove();
              \$( "#cont" ).remove();
              \$( "#minorR" ).remove();
              \$( "#majorR" ).remove();
```

```
            \$( "#contrast" ).remove();
            \$( "#xmin" ).remove();
            \$( "#xmax" ).remove();
            \$( "#ymin" ).remove();
            \$( "#ymax" ).remove();
            \$( "#kk" ).remove();
            \$( "#ymax" ).remove();
            }
        };

        function setFeedback(response, answer){
            /*called when response or answer is going to be rendered.
response: student's response of question.
answer: correct answer of question.*/

            if (response == "No answer" && answer == null) {
                /* not yet attempted, not in the gradebook */
                var temp = [];
                run(temp, 1);

            } else if (answer == null) {
                run(response, 2);


            } else if (answer != null) {
                /* attempted, in gradebook */
                run(answer, 3);


            }

        };

        function prevent_ofb_left(value){
            if (value == 0) {
                return 0;
            }  else {
                return value - 1;
            }
        };

        function prevent_ofb_right(value, length){
            if (value < length) {
                return value + 1;
            }  else {
                return value ;
            }
```

```
        };

        function getResponse(){
            /*called when grade button is clicked,
to retrieve back student's response.*/
            /*Your code starts from here:*/

            var res =  JSON.parse(AnswerStr);
            var response_edited = res[0];
            var pathsPointsfitsX = [];
            var pathsPointsfitsY = [];
            for (i = 0 ; i < response_edited.length ; i++) {
               pathsPointsfitsX.push(response_edited[i][0]);
               pathsPointsfitsY.push(response_edited[i][1]);
            }
            var mySpline = new MonotonicCubicSpline(pathsPointsfitsX, pathsPointsfitsY);

            var max_y = Math.max.apply(null, pathsPointsfitsY);
            var min_y = Math.min.apply(null, pathsPointsfitsY);
            length = pathsPointsfitsX.length;

            val_pos_max = pathsPointsfitsY.indexOf(max_y);
            val_pos_min = pathsPointsfitsY.indexOf(min_y);

            console.log(val_pos_max);
            console.log(val_pos_min);

            var max_x = pathsPointsfitsX[val_pos_max];
            var min_x = pathsPointsfitsX[val_pos_min];

            for (x_t = prevent_ofb_left(pathsPointsfitsX[val_pos_min]);x_t <=
prevent_ofb_right(pathsPointsfitsX[val_pos_min], length); x_t = x_t+precision  ){
               if ( mySpline.interpolate(x_t) < min_y ) {
                  min_y = mySpline.interpolate(x_t);
                  min_x = x_t;
               }
            }

            for (x_t = prevent_ofb_left(pathsPointsfitsX[val_pos_max]);x_t <=
prevent_ofb_right(pathsPointsfitsX[val_pos_max], length); x_t = x_t+precision  ){
               if ( mySpline.interpolate(x_t) > max_y ) {
                  max_y = mySpline.interpolate(x_t);
                  max_x = x_t;
               }
            }

            console.log(" max_y "+ String(max_y) + " x: " + String(max_x));
```

```
            console.log(" min_y "+ String(min_y) + " x: " + String(min_x));

            AnswerStr = AnswerStr.slice(0, -1);
            AnswerStr = AnswerStr + ", [[" + String(max_y) + " , " + String(max_x) + " ]] , [[" +
String(min_y) + " , " + String(min_x) + "]]]";
            console.log(AnswerStr);
            return AnswerStr;

        };
```

Appendix D: Example Algorithm

```
$teachermode = "false";
$antw = "[[-1200, 400], [-1000, 400], [-500, 0], [500, 0], [900, 400], [1200, 400]]"; # is $antw1
in mathapp
$antw1=maple("$antw");
$anspoints = "[[-1200, 400],[-1000,400],[-590,950],[-600, 950], [-500, 0],[-490,
0],[490,0], [500, 0],[600,-550],[900,400],[1200,400]]";
$answerplot=plotmaple("p1 := plot(CurveFitting[Spline]($anspoints,x), x = 0 .. 6,
thickness=2,color=blue):p2 := plot($answerpoints, style = point, symbol = solidcircle,
symbolsize = 20,color=brown):plots[display]({p1,p2},view=[0..6,-
3..3],labels=[``,`I__l/S__0`]),axis = [gridlines = [majorlines = 1]]");

$axes = "[-1200,1200,-1000,1000]"; # is $listranges maple("[-6..0,-3..3]");
$cavansWidth = "800px";
$canvasHeight = "500px";
$canvasDef = "{
            width: 800,
            height: 500,
            vPad: 30,
            hPad: 30
        }";
#is $BgPoints maple("[[-10, 0],[-7, 0],[-6.5, 0], [-6, 0], [0, 1.5], [6, 0], [6.5, 0],[7, 0],[10, 0]]");
$backgroundlines = "{
            lijn1: {
               bgcoord: [[-500, 0],[-350,300],[-100,600],[100,600],[350,300],[500,0]],
               x: [-1200, -600 ,0 , 600, 1200],
               y: [ 0, 0 ,  0 , 0,  0],
               lineColor: 'grey',
               lineColorGreyShade: -1,
               lineThickness: 1,
               x_limit_min:-1200,
               x_limit_max:1200
            }
        }";
$axis_defintion = "{
            y_axis_position: 'auto',
            yLabelNumberPrecision: 0,
            yLabelColor: 'black',
            yLabelJustification: 'center',
            yLabelPositionVertical: +5,
            yLabelPositionHorizontal:  -17,
            yLabelShowZero: false,
            yLabelFontSize : 12,
            yAxisFlipped: false,
            yAxisArrow: true,
            yFontSize: 10,
            yAxisName: 'S [*10^3 m3/year]' ,
```

```
            yAxisNameFontSize : 15,
            yAxisNameFontColor : 'black',
            yAxisNameJustification : 'center',
            yAxisNameVertical: -230,
            yAxisNameHorizontal: -70,
            yAxisNameOrientation: 0,
            x_axis_position: 'auto',
            xLabelColor: 'black',
            xLabelJustification: 'center',
            xLabelPositionVertical: +18,
            xLabelPositionHorizontal: 0,
            xLabelShowZero: false ,
            xLabelFontSize : 12,
            xAxisFlipped: false,
            xAxisArrow: true,
            xAxisName: 'x(m)' ,
            xAxisNameFontSize : 15,
            xAxisNameFontColor : 'black',
            xAxisNameJustification : 'center',
            xFontSize: 10,
            xAxisNameVertical: 40,
            xAxisNameHorizontal: 380,
            AxisLineColor: 'black',
            AxisLineThickness: 1,
            AxisArrowSize: 10,
            AxisArrowAngle: 135,
            AxisArrowLineThickness: 1,
            AxisArrowLineColor: 'black'
        }";
$minor_grid_lines = "{
            xStep: 100,
            yStep: 50,
            lineWidth: 0.5,
            lineColor: 0.5,
            checkmark_offset: 3,
            checkmark_color: 'grey',
            checkmark_width: 0.5
        }";
$major_grid_lines = "{
            xStep: 200,
            yStep: 200,
            lineWidth: 0.5,
            lineColor: 0.5,
            checkmark_offset: 5,
            checkmark_color: 'black',
            checkmark_width: 0.8
        }";
```