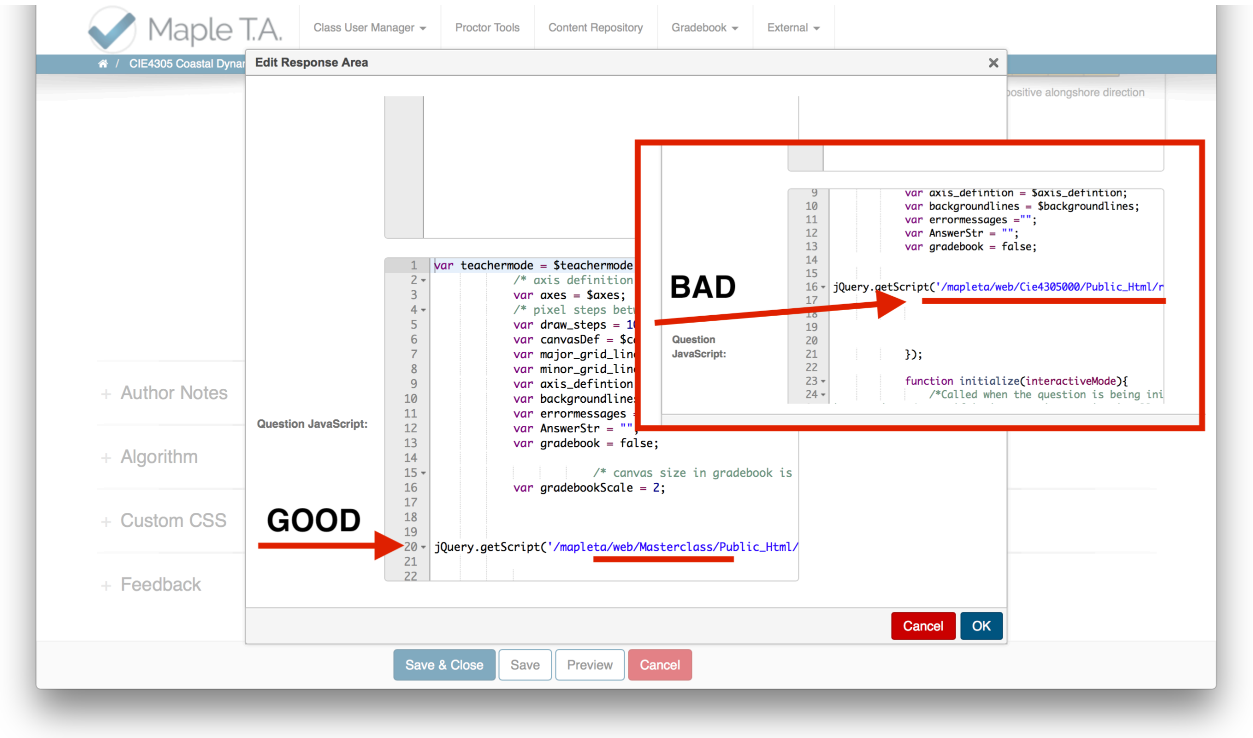
Drawing App

The drawing app is build using of paperJS library on top of HTML5 canvas API. In its current form, APP contains the following javascript files:

- run_app.js
- paper.full.min.js
- cubic_spline.js

It's advised to place these files in the Master Class. This way, librarys can be accessed from any Maple TA class on the server.
For the local development, all files can be placed in any other class. It is however advised to rename the run_app.js file for development if run_app.js is present in the Master class.

Drawing app -blank can be placed within any Maple TA question, with exception of legacy question types. To create drawing app blank from scratch, copy the HTML, CSS en Javascript to the HTML blank fields. Then check that URI is pointing to the Master class:



```
var teachermode = $teachermode
        /* axis definition
        var axes = $axes;
        /* pixel steps betw
        var draw_steps = 1(
        var canvasDef = $c(
        var major_grid_lin(
        var minor_grid_lin(
        var axis_defintion
        var backgroundline:
        var errormessages =
        var AnswerStr = ""
        var gradebook = false;

                /* canvas size in gradebook is
        var gradebookScale = 2;



jQuery.getScript('/mapleta/web/Masterclass/Public_Html/
```

```
        var axis_defintion = $axis_defintion;
        var backgroundlines = $backgroundlines;
        var errormessages ="";
        var AnswerStr = "";
        var gradebook = false;


jQuery.getScript('/mapleta/web/Cie4305000/Public_Html/r




        });

        function initialize(interactiveMode){
            /*Called when the question is being ini
```

Attention:
Cloning a Maple TA question may potentially reset the URI from Master class to the local class within the HTML blank.

The drawing app is meant to be configured from the 'Algorithm' portion of the question, and doesn't require any HTML/CSS/JS adjustments. The definition is written in JSON form and stored as a string to work with Maple TA, so pay a close attention to the quotation (") marks!

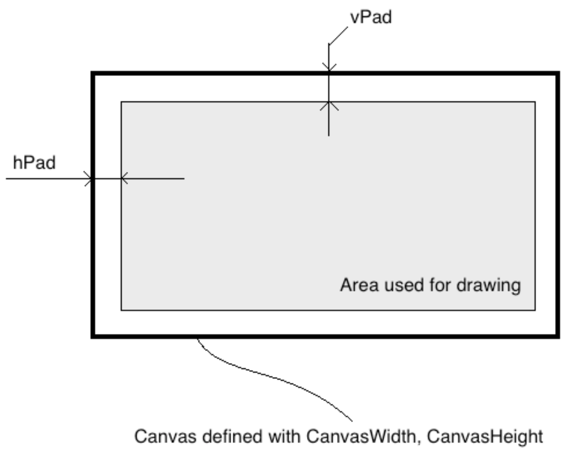
Overall, definition file is split in the following sections:

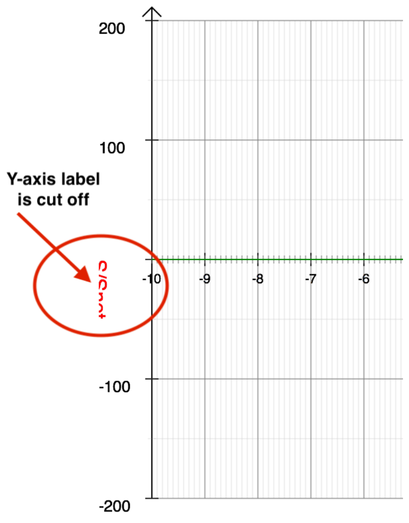
- Axes
- Canvas
- Gridlines
- Teachermode

The canvas definition is expressed redundantly in form:

```
$canvasWidth = "800px";
$canvasHeight = "400px";
$canvasDef = "{
            width: 800,
            height: 400,
            vPad: 20,
            hPad: 20
        }";
```

This is required to accommodate for unpredictable sequence of initialisation in different browsers. The width and heights of both declarations should match! The *vPad* and *hPad* parameters define additional padding on top/bottom and left/right from the edge of the app:





Please note that padding simply shrinks the drawing area, and does not expand the defined canvas parameters. (The outer dimensions don't change)

Adjusting padding may be used to add blank space next to the drawing. For example, to solve axis-labels or numbering getting cut-off.

The aspect ratio, the height / width, can be selected based on the shape of the graph in the drawing app. The actual values are expressed in browser-coordinates and have no direct relation to the axis coordinates within the drawing app. For example drawing app width of 800px can be used for any x-axis definition, whether it's -10 > x > 10 or -10,000 > x > 10,000.

The axes of the graph should be defined as:

```
$axes = "[-10,10,-200,200]";
```

As [x_min, x_max, y_min, y_max]

The teacher mode ($teachermode = "true") allows you to print additional metadata while authoring and testing the blank. This can be handy for the grading algorithm development. It should be turned off for deployment in assignments. (Code may check disable teacher mode in within assignment, but I'm not sure anymore)

Axes definition determines the axes layout. The horizontal and vertical axes have the same functionality. Regardless the axis, following parameters should be set:

```
y_axis_position: 'auto',
yLabelNumberPrecision: 0,
yLabelColor: 'black',
yLabelJustification: 'left',
yLabelPositionVertical: +10,
yLabelPositionHorizontal:  -40 ,
yLabelShowZero: false,
yAxisFlipped: false,
yAxisArrow: true,
yFontSize: 10,
yAxisName: 'S/Snet' ,
yAxisNameFontSize : 15,
yAxisNameFontColor : 'red',
yAxisNameJustification : 'left',
yAxisNameVertical: 0,
yAxisNameHorizontal: -45,
yAxisNameOrientation: 90,
```

Axis position is a switch used to force axis allocation. For example, vertical-axis can be forced to be placed on the left of the screen, so horizontal axis would run from -10 to 10, without y-axis at 0 mark. Position can be set to 'auto', 'left', 'right', 'top', 'bottom' depending direction. Normally, it should be set to '**auto**'.

The **Labels** are numerical values displayed next to the checkmarks on the axis. It should be noted that the number are calculated and are therefore floating numbers (0.5). Current computers are inherently bad in storing floating point values. For example in python. 1.2 - 1.0 = 0.19999..96. To solve this problem, all floating numbers should be rounded off. This is done with using **yLabelNumberPrecision** that determines number of digits after the decimal separator. When set to 0, the number will be converted to integer (e.g. 0,1,2,3). In case the axis is defined in steps of 0.5, this may lead to interesting results: 0, 1, 1 , 2 , 2 instead of 0 , 0.5 , 1 , 1.5, 2. To solve this, simply set precision to "1".

The **style**:

```
yLabelColor: 'black',
yLabelJustification: 'left',
```

Can be used to set different colours and alignment. Due to inherent uncertainty over font size and character lengths, the numbers (labels) may misalign with the actual axis checkmarks. To solve this, the labels should be moved. This is done with

```
yLabelPositionVertical: +10,
yLabelPositionHorizontal:  -40 ,
```

The vertical + 10 moves the label with 10px down, in accordance to the browser coordinate system. Horizontal - 40, will move checkmarks 40 pixels to the left. The value selection can be done in a trial-and-error manner, or with browser plugins (e.g. PageRuler in chrome) that can assist with measuring distances on the screen.

The last option for **Label** is

```
yLabelShowZero: false
```

When false, the zero (0) digit won't be shown. The axis numbering would be -5, -4, -3, -2, -1, 1 ..

yAxisFlipped: true allows user to re-define axis from right to left, or top to bottom. It doesn't require any other adjustments in the code or definition (e.g. background lines etc).

AxisName, is the text next placed next to the axis. App does not support Maple commands or Latex. Theoretically, greek characters can be used inserted with HTML codes, however your milage may vary.

Attention!
JavaScript runs on the clients (yours and students browser) and execution is highly dependant on the browser. For example, adding special characters to Axis text may work on your computer in your browser (for example Firefox), by no means will guarantee same results on someone else's (for example Safari).

Justification parameter can be used to align the text, if axis name spans over multiple lines.

NameVertical and **NameHorizontal**, allows you to place text freely anywhere within the canvas of the app. The value offset the position with respect to the axis centre (horizontally) and top (vertically). Additionally, text can be rotated with **NameOrientation**. The value is expressed in **degrees**, both positive and negative is accepted.

The axis-lines can be modified using:

```
AxisLineColor: 'black',
AxisLineThickness: 1,
AxisArrowSize: 10,
AxisArrowAngle: 135,
AxisArrowLineThickness: 1,
AxisArrowLineColor: 'black'
```

The arrow can be disabled using `xAxisArrow: false` if required.

Minor_grid_lines and **major_grid_lines** provide a construct to modify the minor (no labels) and major (numbered/labeled) lines. The position is defined on base of frequency:

```
xStep: 0.1,
yStep: 5,
```

In this case the horizontal gridlines will be drawn every 0.1 (0, 0.1, 0.2, 0.3 etc) and vertical every 5 (0,5,10,15 etc). The gridlines follow the axis coordinate system.

Attention! Attention! Attention!
App is designed to start and end the graph with a major line! APP is dumb and will start drawing regardless. For example, setting major line to xStep to 5 and defining the x-axis from -8 to 8 will give incorrect results.

Backgroundlines allow user to draw additional curves in the background of the image. There's no limit on the number of lines, but it will likely affect the performance. Lines are defined as:

```
$backgroundlines = "{
        lijn1: {
           x: [-10,-7,-6.5, -6,   0, 6, 6.5, 7, 10],
           y: [  0, 0,   0,  0, 1.5, 0,   0, 0,  0],
           lineColor: 'grey',
           lineColorGreyShade: -1,
           lineThickness: 1
        },
        lijn2: {
           bgcoord: [[-10, 0],[-7, 0],[-6.5, 0], [-6, 0], [0, 1.5], [6, 0], [6.5, 0],[7, 0],[10, 0]],
           lineColor: 'grey',
           lineColorGreyShade: -1,
           lineThickness: 1
        }
    }";
```

More lines can be added by copying and renaming the block (e.g. lijn1, lijn2 etc).

Background line is always drawn as a spline. By default, as a cubic **monotonic**. It can be set to 'normal' cubic spline in the code of the app by changing *MonotonicCubicSpline* to *CubicSpline* in *draw_axis()* function. It will affect all background lines.

The coordinates of background lines are defined either as two arrays, or as a single double array. The format is [[x1,y1] , [x2,y2]] or x: [x1, x2] and y: [y1, y2]. The colour of background line can be set as text (e.g. 'grey', 'red', 'green' etc) or as a scale of black (*lineColorGreyShade*). In latter, a float between 0 and 1 should be provided.

By default, the background line is interpolated and drawn within the domain of provided points. If curve is interpolated incorrectly, more points can be added in the begin or end to force line to a certain shape. To prevent them from being shown *x_limit_max*, and *x_limit_min* can be added:

```
        x_limit_max: 4,
```

By doing so, the curve will be interpolated till the last point in the x-array, but drawn till the point x = 4.

Attention! Attention! Attention!
App is not smart, and expects the x-values to be sorted from small to large, from negative to positive, even when X-axis is flipped (negative defined to the left).

Below, is a typical algorithm section for a drawing app:

```
$teachermode = "false";

$antw = "[[-10,2],[-6,2],[-3,6],[0,2],[3,6],[6,2],[10,2]]"; # is $antw1 in mathapp
$antw1=maple("$antw");
$answerpoints =maple("[[-8,2],[-6.1,2],[-6,2],[-3,6],[-2.9,9],[0,2],[2.9,3 ],[3,0],[6,2],[6.1,2],[8,2]]");
$answerplot = plotmaple("p1 := plot(CurveFitting[Spline]($answerpoints,x), x =-10 .. 10,
thickness=2,color=blue):p2 := plot($answerpoints, style = point, symbol = solidcircle, symbolsize =
20,color=brown):plots[display]({p1,p2},view=[-10..10,-4..4],labels=[``,``]),axis = [gridlines = [majorlines = 1]]");

$axes = "[-10,10,-4,4]"; # is $listranges maple("[-10..10,-2..2]");
$canvasWidth = "800px";
$canvasHeight = "400px";
$canvasDef = "{
            width: 800,
            height: 400,
            vPad: 20,
            hPad: 20
        }";
#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: {
                x: [-10,-7,-6.5, -6,   0, 6, 6.5, 7, 10],
                y: [  0, 0,   0,  0, 1.5, 0,   0, 0,  0],
                lineColor: 'grey',
                lineColorGreyShade: -1,
                lineThickness: 1
            }
        }";
$axis_defintion = "{
            y_axis_position: 'auto',
            yLabelNumberPrecision: 0,
            yLabelColor: 'black',
            yLabelJustification: 'center',
            yLabelPositionVertical: +5,
            yLabelPositionHorizontal:  15,
            yLabelShowZero: false,
            yAxisFlipped: false,
            yAxisArrow: true,
            yFontSize: 10,
            yAxisName: 'tu' ,
            yAxisNameFontSize : 15,
            yAxisNameFontColor : 'black',
            yAxisNameJustification : 'center',
            yAxisNameVertical: -180,
            yAxisNameHorizontal: -25,
            yAxisNameOrientation: 0,
            x_axis_position: 'auto',
            xLabelColor: 'black',
            xLabelJustification: 'center',
            xLabelPositionVertical: +18,
            xLabelPositionHorizontal: 0,
            xLabelShowZero: false ,
            xLabelFontSize : 10,
            xAxisFlipped: false,
            xAxisArrow: true,
            xAxisName: 'km' ,
            xAxisNameFontSize : 15,
            xAxisNameFontColor : 'black',
            xAxisNameJustification : 'center',
            xAxisNameVertical: 30,
            xAxisNameHorizontal: 380,
            AxisLineColor: 'black',
            AxisLineThickness: 1,
            AxisArrowSize: 10,
            AxisArrowAngle: 135,
            AxisArrowLineThickness: 1,
            AxisArrowLineColor: 'black'
        }";
$minor_grid_lines = "{
            xStep: 0.1,
            yStep: 0.1,
            lineWidth: 0.2,
            lineColor: 0.5,
            checkmark_offset: 3,
            checkmark_color: 'grey',
            checkmark_width: 0.5
        }";
$major_grid_lines = "{
            xStep: 1,
            yStep: 1,
            lineWidth: 0.5,
            lineColor: 0.5,
            checkmark_offset: 5,
            checkmark_color: 'black',
            checkmark_width: 0.8
        }";
```

The last part of the app is the grading code. The grading is done on server-side, meaning that the grading code is not accessible for the students. The technicalities are explained more detailed further the document, but it is important to realise that grading is done in a Maple-graded-like environment using Maple.

App returns the students answer in form of array with 3 elements:

- Spline
- Student points in browser coordinates
- Student points is graph coordinates

The first element is array containing points of the spline student drown on the screen by the student. Spline is expressed as [[x1,y1] , [x2,y2] ... [x_N,y_N]] in the graph coordinates and can be used for grading. All x-values are ordered ascending (from -10 to +10 for example, regardless the axis orientation).

The second element is the array of student drawn points in browser coordinates. The third element is an array of student points in graph coordinates. Both arrays are expressed as [[x1,y1] , [x2,y2] ... [x_N,y_N]] due to useful for Maple Grading.

In the grading code, the spline is typically reconstructed from set of points of the first array. It's **IMPORTANT** to realise the following:

- Maple selects the best fit for given set of points automatically
- Maple does not have algorithm for monotonic cubic spline

To solve these problems and to ensure that the browser-drawn cubic spline will have the minimal error to Maple drawn spline regardless the shape, a lot of points will be exported. The number of points will depend on the length of the drawn spline, but can be changed by modifying the draw_steps = 10 variable in the JS portion of the blank. Changing this value will increase the error between student drawn and displayed spline and maple drawn spline used for the grading.

While creating grading code please realise that:

- Student drawn spline is only defined in the domain of drawn points.
- Maple spline spans from negative infinity to positive infinity

But most importantly, **Maple spline is defined only between the spline points**! Consider the following hypothetical set of points:

[[0,0], [5,5] ,[10,10]]

Calculated spline will be **UNDEFINED** in points [0,0] , [5,5] etc. While calculating value of a spline in a given point, be aware that the result may be undefined. A hack to resolve this problem, is to look at location very close to the required point, for example [0.000000001,0].

Development

This section address the inner working of the app, explaining some of decisions w.r.t. Maple TA and Mobius.

All MapleTA/Mobius HTML5 blanks are executed within an individual iframe. HTML blanks are not designed to communicate directly with each other. In the grade book, the correct and student's response blanks are drawn separately.

All blanks are initialised with the following functions:

```
function initialize(interactiveMode)
function setFeedback(response, answer)
function getResponse()
```

HTML blank has the following states:

Assignment: not answered, first attempt
Assignment: not answered, not the first attempt (student browsed to a different question and came back)
Assignment: answered, not the first attempt (student browsed to a different question and came back)

Gradebook: students answer
Gradebook: correct answer

Preview: not answered, first attempt
Preview: answered, gradebook-like overview

App can determine the state using the following variables:

interactiveMode
response
answer

interactiveMode is *true* when question is either opened in the assignment or in the preview.

response is either "*No answer*" (and translations), *null* or *whatever* your app returns with *getResponse*(). *response* is set to *null* when app is opened in the grade book to show the correct answer. In all other cases it's either "*No response*" (or translation) or *whatever* was previously returned with *getResponse*().
In the assignment, response is initialized with "*No response*" both on the first attempt as on every revisit of the question as long student didn't interact with the app causing *getResponse*() to be called.
answer is either *null* or whatever is set in the Correct-field of the blank.

For example by:

```
function setFeedback(response, answer){
        if (response == "No answer" && answer == null) {
            /* not yet attempted  */
            run([], 1);
        } else if (answer == null) {
            /* previously attempted */
            run(response, 2);
        } else if (answer != null) {
            /* show correct answer in the gradebook */
```

```
                run(answer, 3);
            }
        };
```

The translations are handled within the app itself.

When writing code directly in Javascript portion of the HTML blank (and thus not loading it externally like done in the app), developer should be aware of the following **big no-no's:**

ISSUE 1: Realise that js code is wrapped to a single line. Add line comments "//" comment all of your code and all of hidden Maple TA functions. Questions breaks with

TypeError: e.setFeedback is not a function. (In 'e.setFeedback(t=="undefined"||t==null?null:t,n=="undefined"||n==null?null:n)', 'e.setFeedback' is undefined)

SOLUTION: Comment using /* my comment that won't break the code */

ISSUE 2: Maple TA and Mobius, uses "$" for algorithm variables. For example $canvasWidth. To use jquery that's available within TA, escape the character: **\$** or **jQuery**

ISSUE 3: Writing code directly in JS field of the HTML5-blank is a bad idea. If code breaks, but works when loaded outside TA, try adding "**;**" at the end of function declarations. Adding or removing arbitrary whitespaces may or may not work. If it doesn't work, try minifying the code. When you realise it still doesn't work:
SOLUTION: use: jQuery.getScript('/mapleta/.../my_app.js', function(){}, put your code in a separate .js file and relax.

ISSUE 4: Realise that JS runs within its container: iframe. iFrame is loaded when assignment is loaded, what happens next will depend on the browser. It may resize container and then pass control to the app. OR it may pass control to the app and then resize it. In the app, all depends on the algorithm-coded canvas dimensions, to prevent inconsistencies. App tries to resize the container twice.

ISSUE 5: Maple TA and Mobius use requireJS to manage internal library dependencies. If your library uses requireJS, before coding, validate that it can be loaded. If it doesn't load, please contact Maplesoft support.

THE WORKINGS OF THE APP.

The drawing app is based on paperJS library and under the hood draws SVG objects using HTML5 canvas. In practice, since browser HTML5 implementation is used directly, the drawing may look different.

SVG objects can be layered and grouped. In the app, three groups are defined and layered in the following order:

- PermanentElements
- curveGroup
- listendGroup

The permanent elements are drawn once, and contain axis, grids and background lines. Curve group contains the student's drawn spline. The listendGroup contains students drawn points.

The listener is not attached to the listendGroup directly. This is a consequence of a bug in the current version of paperJS. Instead, a separate object, called Tool is created and attached to the canvas itself.

The grading arrays, as explained in the previous section, need to be defined as strings. Passing javascript objects to Maple will likely fail. In theory, array.toString method can be overloaded, instead, I opted to create string objects on the fly. Why?

1) There's no debugger
2) String is not modified, everything else will.
3) String can be dumped in the console and pasted directly in Maple program for analysis.

It should be noted, that theoretically, it is possible to use Maple code to work with javascript objects. However, in practice, string objects turned out to be much easier alternative.

Known problems:

Major: Grid lines
The HTM5 canvas is weird. In the app, the line width is often set to less than one, at least defined as < 1. This is handled by the library, however, width = 1 should result in 1px wide line. But it's not the case, width = 1, results in much much thicker curves.
This is a consequence of the following problem. Let's say we draw a 1px line at coordinate (10,10,0_. This will fill pixel (10,10). However, if line thickness is 0.5, coordinate 10, will be filled with "half of the colour". If we try to fill coordinate (9.5, 10, 0) with 0.5px wide object, pixel (9,10) will be filled with the full colour. Also filling coordinate (9.5, 10, 0) with 1px wide object, will fill pixels (9,10) and (10,10) with half of the original colour.
Most importantly, this also depends on the browser. Chrome doesn't like it and create interference patterns of the grid lines. Issue with Chrome can be solved by adding +0.5px to each line, however it visible shifts the gridlines.
Somehow this also applies to colour. The uneven colours (grey scale) are shown differently in chrome.

Minor:

- Setting different horizontal and vertical padding (vPad != hPad) breaks the graph (probably swapped them somewhere)
- Dragging speed is limited
- X-points on the spline can be dragged close together e.g. (4.5, -10) and (4.8,50). This combination may break the spline