# Extra Credit Programming Projects
# MAT 300/500 – Spring 2025

For each part, the student should implement all of the required details. If there are suggestions for changes to the described interfaces, please bring those up to the TA and instructor for consideration.

Grading for these projects will be out of 10 points. This grade will be converted to allow a maximum 2% flat increase to course grade for each project completed.

Final due date for any and all extra credit projects is: Friday, April 11. Absolutely no exceptions or extensions will be made to this date.

## EC I: Hermite Interpolation (Osculation)

In this part you will extend the polynomial interpolation to include derivatives. This is called osculation, or Hermite interpolation. The base case is still polynomial interplation. Additionally, the user will need to be able to enter derivative information. For the first derivative, you should have an option to do this graphically in the following way: The user should be able to right-click on one of the input points and have a drop down menu pop up. In this menu one of the items should be to show tangents. If this is selected, a small tangent line segment will appear with default slope zero at the point. The slope of this line should then be adjustable with mouse control (either by simply moving the mouse or the wheel, etc.) A further click should select the current slope and assign it to that point.

Higher derivatives will have to be entered through an input box. This input should also be an option for the first derivative, or slope, as well. Once a collection of points has been assigned derivative values the curve should be redrawn, solving the given osculation problem with the appropriate degree polynomial.

## EC II: Best Fit Line and Parabola

In this part you will compute a best fit line and parabola to the input points. There should be an option for line or parabola, which defaults to line. When the user has clicked on two input points, the line through the two points should be graphed. When the user clicks on three or more points, an approximate line is graphed according to the following procedure:

First, the mean (or average) point $M = (x_M, y_M)$ is calculated by averaging all of the $x$ coordinates to get $x_M$ and averaging all of the $y$ coordinates to get $y_M$. The desired line will pass through $M$. A precomputed circle of points is now used around $M$ to form a new collection of lines through $M$ and one of the points on the circle. (Alternatively, a box can be used. In either case, this is simply to give a discrete set of points which can be used together with $M$ to determine a discrete set of lines which pass through $M$ and are evenly distributed like the spokes of a wheel.) Each line $L$ in this set is tested by computing the sum $S(L)$ of the perpendicular distances between the line and each input point. We want to minimize this sum. Hence, we choose the line with the smallest value of $S(L)$. To increase speed and efficiency, a small number of points $k$ on the circle (or box) can be used initially, say $k = 100$. Then the three lines with smallest value for $S(L)$ are identified. Call these lines $L_1$, $L_2$, and $L_3$, and call the points that they pass through on the circle (or box) $P_1$, $P_2$

and $P_3$. Now identify a new set of $k$ points on the circle (or box) in between these three points, by evenly spacing them along the arc that contains $P_1$, $P_2$, and $P_3$. This process can be interated. A tradeoff bewteen the number of iterations and the value of $k$ can be tested for speed and efficiency. Call the best fit line $L_{bf}$.

Once the best fit line is computed, the best fit parabola is based on this line. First find the perpendicular bisector $B$ which passes through $M$ and is perpendicular to $L_{bf}$. Now group the input points into two groups which are on either side of the line $B$. (Leave out points on $B$.) Call these sets of points $S_1$ and $S_2$. Compute the mean points for these two sets, say $M_1$ and $M_2$. Next form a family of parabolas by choosing control points $Q_0 = M_1$ and $Q_2 = M_2$ where $Q_1$ will be any point on the line $B$. Choose a discrete set of such points $Q_1$ which are distributed evenly on the segment which extends a maximum distance of $4c$ where $c$ is the maximum of the two distances $d_1$ from $M_1$ to $M$ and $d_2$ from $M_2$ to $M$. For each such $Q_1$ there is a quadratic Bezier curve defined by the control points $Q_0$, $Q_1$, and $Q_2$. Call this curve $C_t$. For each $C_t$ compute the sum $S(C_t)$ of the perpendicular distances between $C_t$ and $P_i$ for each of the input points $P_i$. Choose the $C_t$ with the smallest value of $S(C_t)$. This is the first candidate for the best fit parabola $C_{bf}$. A similar approach to enhance speed and efficiency that was used to find $L_{bf}$ can be used here by working with subintervals of the line segment to refine the search for the smallest value of $S(C_t)$.

Finally, for reasons of symmetry, we repeat all of the above paragraph with the role of $B$ and $L_{bf}$ swapped. This produces another candidate for $C_{bf}$, so we can choose the one with smaller value of $S(C_t)$, and we call it the best fit parabola.

## EC III: Audio Signals with Bernstein Polynomials

This project is an audio enhancement to the first project. The interface to the first project stays the same, but now you would compute an audio buffer of data in the background which simulates one cycle of a polynomial spline curve. The first half-cycle of the curve, say $f(t)$, is exactly the one being displayed in the window on the interval $[0, 1]$, and the second half-cycle, say $g(t)$, is $f(t)$ inverted and graphed over the interval $[1, 2]$. By inverted we mean that the curve is evaluated as

$$g(t) = -f(2 - t).$$

This gives one complete cycle of the piecewise function over the interval $[0, 2]$. Note: the default case, with all control coefficients equal to 1, gives one cycle of a square wave.

This function is then scaled to fit into one cycle for an audio buffer, which should have the default frequency of 441 Hz (cycles per second) with discrete sampling rate of 44100 Hz. This means that one cycle fits into $44100/441 = 100$ samples along the $t$ axis for audio output. The cycle should also be normalized to have maximum value set to 0.85. Such cycles should be continuously fed to the audio output buffer so that these are played back through the speakers. Then, as the user changes the shape of the curve through the interface by moving control points and changing the degree, the sound quality, or "timbre", should change in real-time. In order to maintain consistent output, the maximum value of the function should be normalized to 0.85, even as the user is changing the shape of the curve. The user should also be able to change the frequency with a slider.

As a starting point one can use the JUCE tutorial called SineSynthTutorial.