CS370 Winter 2025: Assignment 1

Instructors: Leili Rafiee Sevyeri (email: leili.rafiee.sevyeri@uwaterloo.ca)
Eugene Zima (email: ezima@uwaterloo.ca)

Due January 31, Friday, at 5:00pm EST

Submit all components of your solutions (written/analytical work, code/scripts, figures, plots, output, etc.) to Crowdmark in PDF form for each question (multiple pages are allowed). Note that you may resubmit as often as necessary until the due date - only the final submission will be marked.

You must *also* separately submit a single zip file containing any and all code/scripts you write to the Assignment 1 DropBox on LEARN, in runnable format (and if necessary, any data needed to reproduce your results).

You have a number of options of how to prepare your solutions. You can typeset your solutions in a word-processing application (MS Word, LaTeX, etc.), you can write on a tablet computer, or you can write on paper and take photos. **It is your responsibility to ensure that your submission is sufficiently legible. This is particularly important if you are submitting photos of handwritten pages.** TAs have the right to take marks off for illegible answers.

Note that your solutions will be judged not only for correctness but also for the quality of your presentation and explanations.

1. **(6 marks)** Floating Point Basics

   Consider a floating point system $F\{\beta, t, L, U\}$ with base parameter $\beta = 2$, mantissa t being a positive integer, and the exponent range given by $L = -20$ and $U = 20$. Under this system, a non-zero floating point number has the form

   $$x = \pm 0.d_1 d_2 \cdots d_t \times \beta^p,$$

   with $d_1 \neq 0$ and $-20 \leq p \leq 20$.

   (a) (2 marks) The distance between 17 and the smallest representable floating point number larger than 17 is 2^{-5}. What is the smallest floating point number in F larger than 70? Express your final answer in the form specified above.

   (b) (2 marks) What are the smallest and largest *positive* numbers representable in F?

   (c) (2 marks) What is the value of machine epsilon, assuming truncation is used?

2. **(6 marks)** Effects of Cancellation

   Consider the function

   $$f(x) = \frac{e^x - e^{-x}}{2x}. \tag{1}$$

   Suppose you are given that $e^{0.1} = 1.105171$ and $e^{-0.1} = 0.904837$.

   (a) Simulating the five-digit accuracy of the floating point number system $F(10, 5, -10, 10)$ with truncation, evaluate $f(x)$ at $x = 0.1$.

(b) Using Taylor expansions $e^x \approx 1 + x + 0.5x^2 + 0.166x^3 + 0.04166x^4$ and $e^{-x} \approx 1 - x + 0.5x^2 - 0.166x^3 + 0.04166x^4$, obtain a new formulation to compute (1) when x is close to 0. What is the approximate value of $f(x)$ at $x = 0.1$ using your new formula? (Again, using F(10,5,-10,10) with truncation.)

(c) Which computation, (a) or (b), is more accurate? Why?

3. **(8 marks)** Floating Point Error Analysis

Suppose x is a number that can be exactly represented in some floating point system F. Give a bound on the relative error of the floating point calculation

$$(x \otimes x) \ominus 1$$

with respect to the true value of $x^2 - 1$ in terms of machine epsilon E (assuming we can drop/ignore terms proportional to E^2 and higher powers of E, and no overflow/underflow occurs). Recall that $\ominus$, $\oplus$ and $\otimes$ indicate floating point subtraction, addition, and multiplication, respectively. You may assume the number 1 is exactly represented in F.

Repeat your analysis for the equivalent expression $(x \ominus 1) \otimes (x \oplus 1)$.

Which of the methods is better?

4. **(6 marks)** Polynomial Interpolation

Given 2 constant parameters, u and v, determine whether there exists a unique cubic polynomial, $p(x)$ such that

$$p(0) = 1,\ p'(0) = v,\ p'(1) = v + 6u,\ p''(1) = 8u.$$

If such a polynomial exists, determine the coefficients of $p(x)$ in terms of u and v.

5. **(10 marks)** Spline Construction

(a) (4 marks) Write down the linear system of five equations needed to determine the derivative values s_1, s_2, s_3, s_4, s_5 for the cubic spline $S(x)$ going through the points $(-2, 2), (0, -1), (1, -1), (3, 3), (6, 2)$ and having clamped boundary conditions at both ends with $S'(x) = 1$. The system should have the form $T\vec{s} = \vec{r}$ where T is a matrix, $\vec{s}$ is the vector of unknowns comprised of the s_i's, and $\vec{r}$ is the right-hand-side vector, as described in the course notes, Section 2.5. Be sure to show how you arrived at your solution.

(b) (3 marks) Next, use Python (or the mathematical tool of your choice) to solve the linear system from part (a), and then (manually) use the results to determine the cubic spline $S(x)$. Specifically, determine the coefficients of $S_1(x), S_2(x), S_3(x), S_4(x)$ where

$$\begin{aligned}
S_1(x) &= a_1 + b_1(x+2) + c_1(x+2)^2 + d_1(x+2)^3 && \text{for } x \in [-2, 0] \\
S_2(x) &= a_2 + b_2(x-0) + c_2(x-0)^2 + d_2(x-0)^3 && \text{for } x \in [0, 1] \\
S_3(x) &= a_3 + b_3(x-1) + c_3(x-1)^2 + d_3(x-1)^3 && \text{for } x \in [1, 3] \\
S_4(x) &= a_4 + b_4(x-3) + c_4(x-3)^2 + d_4(x-3)^3 && \text{for } x \in [3, 6]
\end{aligned}$$

represent the piecewise polynomials of the cubic spline. (You can give the coefficients to 4 significant digits. You can use a calculator or Matlab to perform the individual calculations needed to find the coefficients; just show your work. There is no need to write or include code for this part.)

(c) (3 marks) Write a short Jupyter notebook that produces a single plot of the resulting smooth curve by evaluating the polynomials you found over their respective domains. Submit your code and plots.

6. **(10 marks)** Using Splines for Parametric Curves

Create a parametric curve representation of your nickname written in your handwriting. This representation should be based on parametric spline interpolation. Your nickname must have at least two curved segments. Figure 1 below shows an example using 3 coarsly-sampled segments.

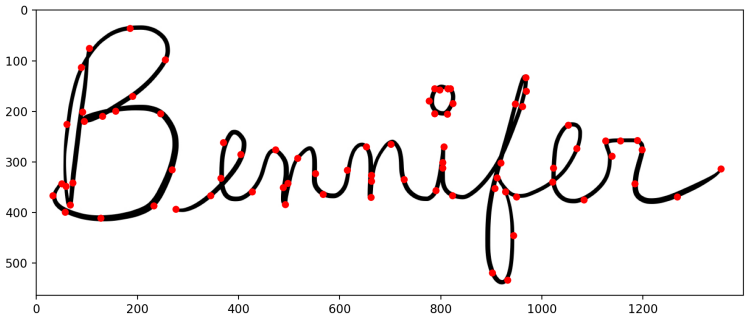

Figure 1: The nickname "Bennifer" written in 3 segments

Steps:

(a) (2 marks) Draw your nickname and select interpolation points. You can do this on graph paper by hand, or you can use `ginput`. Make sure the interpolation points are not too close together. Your curved segments should be sampled sparingly. Figure 1 shows an example.

(b) (2 marks) Add code to the notebook (`A1Q6.ipynb`) to initialize and store the interpolation points for each segment. For example, you could store the coordinates of the interpolation points for the first segment in the arrays `x1` and `y1`, and you would have one `x-y` pair of arrays for each segment. Your data arrays should be hardcoded into your script.

(c) (3 marks) Complete the function `ParametricSpline`; it takes the interpolation points of one segment as input, and outputs a pair of cubic spline functions (one for `x(t)` and one for `y(t)`, along with an array, `t`, that holds the parameter value at the interpolation points. See the function's documentation for more details. The parameter `t` must reflect the pseudo-arclength of the curve. You can use `scipy.make_interp_spline` (and return the spline functions that it creates).

(d) (1 mark) Add lines to the notebook to call `ParametricSpline` for each segment in your nickname.

(e) (2 marks) Add more lines to the notebook to plot your nickname. The plot should graph the parametric spline segments using a refined selection of parameter values (eg. 1000 points per segment). The plot should also display the original interpolation points, and the `x` and `y` axes should be rendered using the same scale ("`plt.axis('equal')`"). Figure 1 is an example of what your output should look like.