Numerical Analysis 1 – Class 9

Thursday, March 17th, 2023

Subjects covered

- Linear interpolation in 1D.
- Polynomial interpolation in 1D – classical Lagrange and Barycentric.
- Chebyshev polynomals and their application to interpolation.
- Runge phenomenon.
- 1D Splines.
- Bilinear interpolation with application to images.
- Meshes and interpolation in triangles.

Reading

- "Barycentric Lagrange Interpolation", J-P. Berrut and L. N. Trefethen. (Linked on Canvas).
- "Chebyshev Expansions", chapter 3 from "*Numerical Methods for Special Functions*", Amparo Gil, Javier Segura, and Nico M. Temme. (Linked on Canvas).
- Kutz, Chapter 3

Problems

Most of the following problems require you to write a program. For each program you write, please make sure you also write a test which validates your program. Please use Canvas to upload your submissions under the "Assignments" link for this problem set.

Problem 1

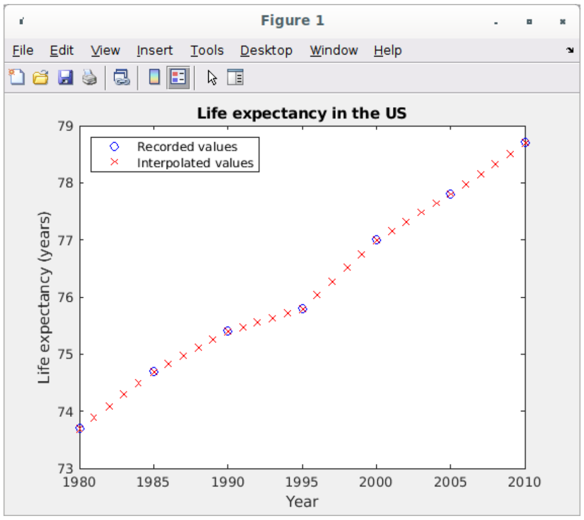
A common use of interpolation is to fill in missing parts of a dataset. Here is a simple example: Consider the life expectancy data for the USA in the period 1980 – 2010. The data is recorded every five years. Here is the data:

Year	1980	1985	1990	1995	2000	2005	2010
Life ex.	73.7	74.7	75.4	75.8	77.0	77.8	78.7

(The post-2020 life expectancy has dropped due to COVID.) The natural question is, what is the estimated life expectancy in the years between the recorded dates? For example, what is a good guess for the life expectancy in 1997? Please do this:

- Write a program which returns the life expectancy for any year between 1980 and 2020 based on the above data. Use linear interpolation to fill in the "between values". Feel free to use Matlab's linterp() function, or my implementation of linear interpolation which you can find on Canvas. Note that if you use my version you will need to implement some sort of hack to deal with the year 2010 since my program will error out if you input an X value exactly equal to the end of the domain. I suggest you just pad the above table with another set of values to avoid this problem.

- Write a program which calls into your above implementation and makes a plot of the estimated life expectancy for all years between 1980 and 2010 inclusive. My plot is shown below. Regarding testing, its enough to get a sensible plot similar to that shown below.



Problem 2

Please recall the finite-difference expressions for first and second derivative we derived in class a couple of sessions ago. They are:

- First derivative, forward difference formula: $y'_n \approx \frac{y_{n+1} - y_n}{h}$
- First derivative, centered difference formula: $y'_n \approx \frac{y_{n+1} - y_{n-1}}{2h}$
- Second derivative: three point formula: $y''_n \approx \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2}$

In class, we manipulated a Taylor series to get these expressions. A different way to derive the expressions involves starting with the Lagrange interpolation formula for $y(x)$ presented in class. Please do the following:

- Write down the two-point Lagrange interpolation formula for $y(x)$ between two points called x_0 and x_1. Note that this is linear interpolation.
- Take the derivative of the above polynomial and then use it to derive the forward difference formula for the first derivative above. Keep in mind that $h = x_1 - x_0$ is a step size.
- Now write down the three-point Lagrange interpolation formula for $y(x)$. Call your three points x_{-1}, x_0 x_1.
- Take the derivative $y(x)$ and from it derive the central difference formula.
- Finally, from the three point Lagrange interpolation formula, derive the expression for the second derivative.
- Answer this question: Based on considering the type of approximation implied by the interpolation formula, why would you expect the central difference formula to have better error

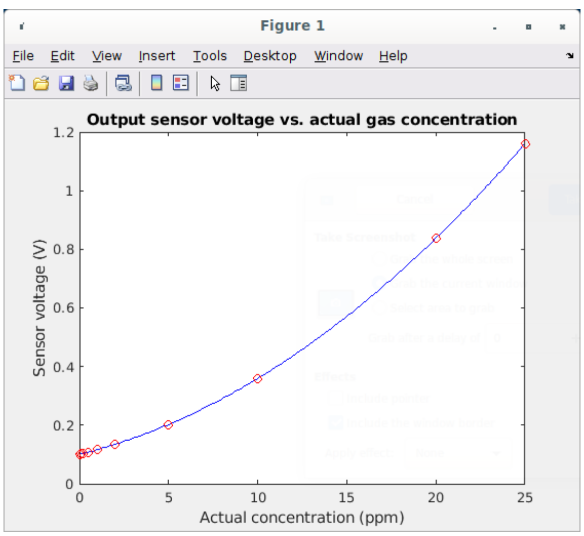
performance than the forward difference formula?

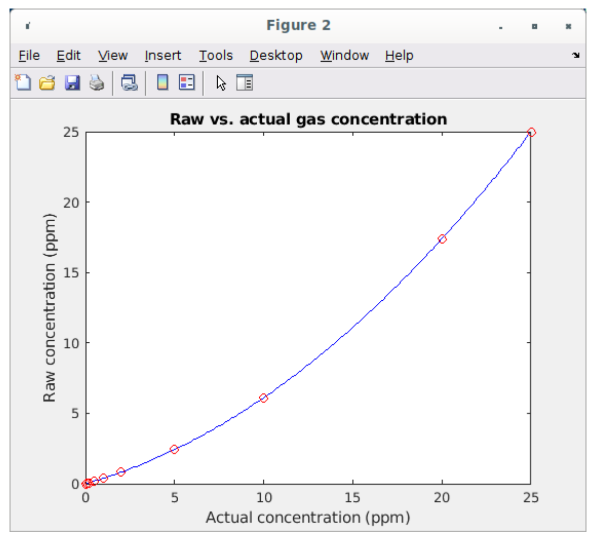
This is a paper-and-pencil exercise. Please turn in your derivations.

Problem 3

A "gas analyzer" is an instrument encountered in many different industrial applications. It is a measurement device which measures the concentration of some target gas (for example, O2) in air. A gas analyzer incorporates a sensor which outputs a voltage related to the target gas concentration passed through the instrument. The analyzer holds electronics which measures the sensor voltage and uses the measured voltage to compute the original gas's concentration for display to the user.

A typical plot of the sensor voltage vs. gas concentration concentration is shown below left. Ideally, the plot should be a straight line. However, non-idealitites in the sensor mean that the output voltage is not linear in the target gas concentration. Therefore, the plot has curvature which must be corrected. To handle this effect, an instrument is *linearized* using calibration gasses at the factory prior to shipment to the customer.



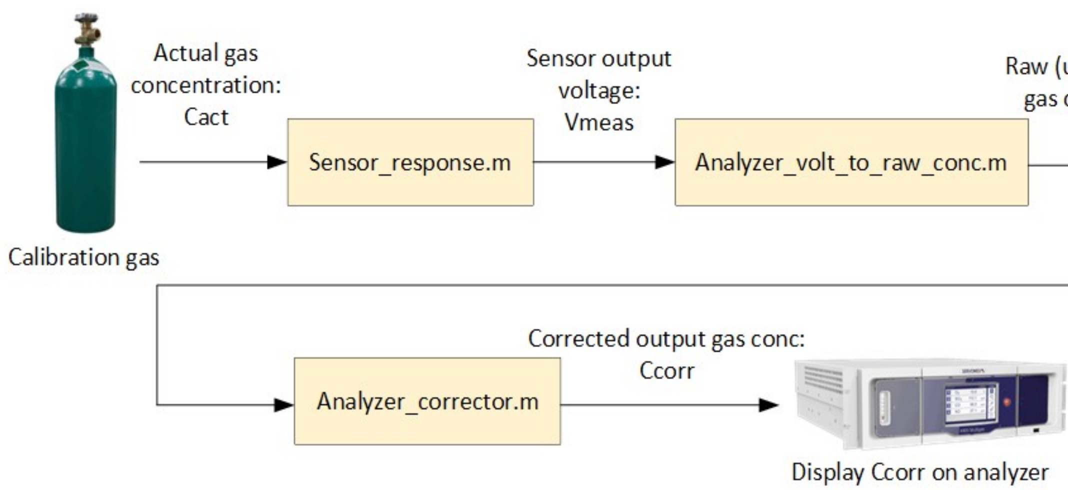


Linearization involves passing an exact concentration of the target gas through the instrument and recording the measured voltage as well as the raw concentration, which is like a first guess of the actual input concentration. A plot of raw concentration vs. actual input concentration is shown above right. Linearization is performed for a small number of points, finding the error between the raw and the actual gas concentration at each point, and adding the difference back to the raw concentration. Since the error is sampled at a small number of points, interpolation is required to "fill in" the gaps between the linearization points. The linearization points are shown as red circles in the above plot. The result is a linearization table like the one shown below.

Actual target concentration Cact (ppm)	**Sensor voltage Vmeas (V)**	**Raw reported concentration Craw (ppm)**
0.0	0.100000	0.000000

0.1	0.101501	0.035401
0.2	0.103024	0.071321
0.5	0.107725	0.182193
1.0	0.116000	0.377358
2.0	0.134200	0.806604
5.0	0.202000	2.405660
10.0	0.359000	6.108491
20.0	0.838000	17.405660
25.0	1.160000	25.000000

Correcting the non-linear response of the instrument is frequently done by simple linear interpolation. Your job is to correct the instrument response. A diagram showing the computational flow is shown below. Please do this:



- At each calibration point in the table, compute the error between the actual (target) and the reported concentration. Call the error δ_i where i denotes the calibration point.
- I have placed a zip file on Blackboard containing the following functions:
    - globals.m – holds global variables needed by all functions. Source this in your top-level test program.
    - sensor_response.m – accepts an actual gas concentration Cact input to a sensor, and outputs the voltage produced by the sensor, Vmeas.
    - analyzer_volt_to_raw_conc.m – This simulates the computation performed by an uncorrected analyzer which takes the voltage to an uncorrected concentration, Craw.
    - analyzer_corrector.m – The starting point for your work. It takes the sensed voltage Vmeas and the uncorrected concentration Craw, corrects it by applying a correction

factor determined by linear interpolation on Craw, and emits the corrected gas concentration, Ccorr, for display to the user. It also makes plots of corrected vs. actual gas concentration. The interpolation step is left out of this program for you to fill in.

- Modify the program analyzer_corrector.m to take Vmeas and Craw and compute the concentration error δ_i at the particular Vmeas using linear interpolation. Then linearize (correct) the raw instrument response by subtracting the error from the inferred concentration. The place to insert your code is in analyzer_corrector.m.   Feel free to use the linear interpolation demo program I placed on Canvas as a component of your work, or use Matlab's linterp() if preferred.
- Make a plot of your corrected instrument response by plotting the reported vs. actual concentration before and after linearization. For reference, my plot is shown below.
- Imagine that the instrument's published specifications say that the displayed concentration (Ccorr) must remain within 5% of the actual input gas concentration over the entire input range of the instrument. Your test program should do this:
    - Make a plot of Craw and Ccorr vs. Cact similar to that shown below.
    - Verify that Ccorr remains within 5% of Cact over the entire input range of the instrument (0 – 25ppm).