ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Models, Algorithms and Data (MAD): Introduction to Computing

Spring semester 2021

Prof. Dr. Jens Honore Walther
Dr. Georgios Arampatzis
ETH Zentrum, CLT
CH-8092 Zürich

Solution

Issued: Saturday, 03.08.2021

Multiple Choice Questions [30 Points]

In the following Multiple Choice Questions, you are asked to tick the correct answers using a 'X' in the corresponding box. In total there are **20 correct answers** in Questions 1-7. Each correctly selected answer is rewarded with 1.5 points. If a correct answer is not selected it is rewarded 0 points. **If you select a wrong answer, 1.5 points are deducted**. The minimal number of points for your answers to all multiple choice questions is 0 out of 30 points.

Question 1: Linear Least Squares

Assume that you did an experiment, and you plotted the resulting data $\{(x_i, y_i)\}_{i=1}^{N=100}$ in Figure 1.

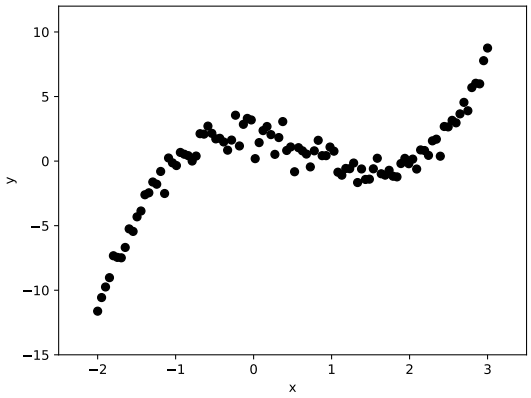


Figure 1: Data from an experiment.

a) You want to find the function $f(x)$ that best approximates the data $y_i \approx f(x_i)$ using a linear least squares fit. This requires

- ☐ selecting the measure of "best".
- ■ selecting a set of basis functions. The error is defined as "least squares", and "linear" means that we use a linear combination of basis functions. The basis functions have to be selected. Ideally this selection happens based on knowledge of an underlying physical law, however this is not necessary.

- ☐ initializing the weights.We solve the normal equation directly for the coefficient, not via an iterative method.
- ☐ knowing the physical law underlying the data. Knowing the underlying physical law is not necessary.

b) The data plotted in figure 1 suggest that we should use

- ■ a polynomial basis $\varphi_k(x) = x^k$
- ☐ an exponential basis $\varphi_0(x) = \exp(kx)$
- ☐ a trigonometric basis $\varphi_k(x) = \cos(kx)$

for $k = 0, 1, 2, 3$. The data indicates correctly that it is generated from a cubic polynomial. The other options will not be sufficient to fit this data.

c) The inversion of the matrix $A^\top A$ in the normal equation $A^\top A\boldsymbol{w} = A^\top \boldsymbol{y}$

- ☐ is impossible, because it is has a very high condition number.
- ☐ can always be done analytically.
- ■ is possible if the chosen basis functions are linearly independent. Chosing linearly independent basis function yields a symmetric, positive definite matrix $A^\top A$, that always can be inverted.

Question 2: Nonlinear systems

We want to solve a general non-linear equation $f(x^\star) = 0$ for the unknown root $x^\star$.

a) The order of convergence can be computed from iterations $x_0, \ldots, x_k$ of the algorithm by

- ☐ taking two subsequent errors $e_{k-1} = x_{k-1} - x^\star$ and $e_k = x_k - x^\star$ and computing

$$r = \frac{\log|e_k|}{\log|e_{k-1}|}.$$

- ☐ cannot be computed as it is only defined in the limit $k \to \infty$.
- ■ computing the sequence of errors $e_k = x_k - x^\star$ and estimating the order r as

$$r \approx \frac{\log\left|\frac{e_{k+2}}{e_{k+1}}\right|}{\log\left|\frac{e_{k+1}}{e_k}\right|}.$$

b) The bisection method is guaranteed to converge. This statement is

- ■ true This follows from the intermediate value theorem.
- ☐ false

Question 3: Interpolation with Lagrange Polynomials and Cubic Splines

a) The following subquestions consider the interpolation of data from a 2π periodic function, that is a linear combination of $\cos(x)$ and $\sin(x)$.

1. We interpolate the function with Cubic Splines interpolation, using 3, 4 and 7 sampling points in $[0, 2\pi]$. The resulting interpolating polynomials are shown in Figure 2. Select the correct combination of letter and number of points that were used for the interpolation.
   - ☐ (a): 4, (b): 7, (c): 3
   - ☐ (a): 3, (b): 4, (c): 7
   - ■ (a): 7, (b): 3, (c): 4 Counting the number of intersections with the function-to-be-interpolated gives the number of points that were used for the interpolation.

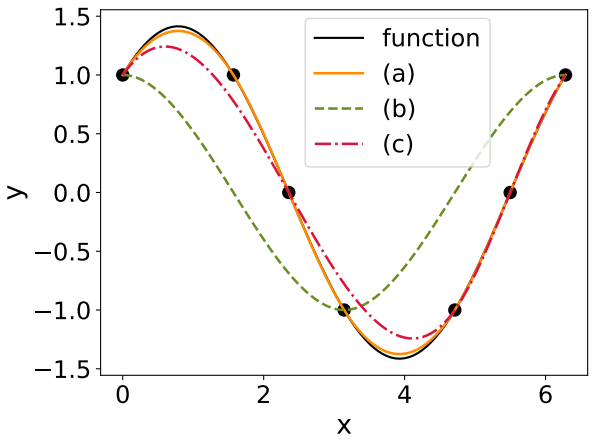


Figure 2: Interpolation of a periodic function.

2. We now consider data with a Gaussian random error term, i.e. $y_i = f(x_i) + \varepsilon$, where $i = 1, \ldots, N$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. In Figure 3, select the correct combination of letter indicating the graph, and interpolating method with N interpolation points.
   - ☐ (a): lagrange interpolation $N = 20$, (b): cubic splines $N = 20$, (c): lagrange interpolation $N = 7$.
   - ☐ (a): cubic splines $N = 7$, (b): lagrange interpolation $N = 20$, (c): cubic splines $N = 7$.
   - ■ (a): cubic splines $N = 20$, (b): lagrange interpolation $N = 20$, (c): lagrange interpolation $N = 7$. (a) passes trough all datapoints, thus it corresponds to a case with $N = 20$. Since for (b) we can clearly observe Runges phenomenon, thus this is Lagrange interpolation, and correspondingly (a) Splines.

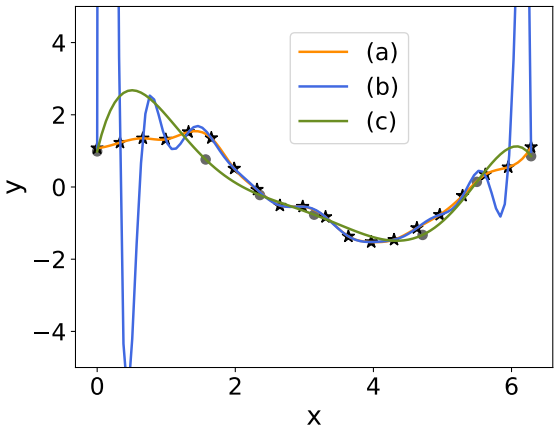


Figure 3: Interpolation of a periodic function with noisy data.

b) Which of the following statements for Lagrange and cubic spline interpolation is/are correct?

- ■ The Lagrange interpolating polynomial through 7 sample points from $f(x) = \frac{1}{2}x^4 + 2x + 3$ without addition of noise will be the original function $f(x)$.. As we have seen in the lecture, if there is no noise in the data, the higher order terms will cancel in the Lagrange interpolating polynomial.
- ☐ Lagrange interpolation is robust against outliers. (No, Lagrange interpolation will pass exactly through the outliers.)
- ☐ If I want to interpolate through the data points: $x_1 = (1, 3)$, $x_2 = (2, 5)$ and $x_3 = (3, 7)$ with Lagrange interpolation, the second Lagrange polynomial will correspond to: $l_2(x) = -(x^3 - 4x + 3)$. No, it is $l_2(x) = -(x^2 - 4x + 3)$. You could see it is not true directly, cause the proposed l_2 is of order 3 which is not possible.
- ☐ For a dataset of 10 non-overlapping datapoints, the Lagrange interpolating function $p(x)$ will have 10 roots, i.e. $p(x) = 0$. (the interpolating function is of degree 9, so 9 roots)
- ☐ For any boundary condition we can use the Thomas Algorithm (TDMA) to solve the system of the unknown second derivatives on the data nodes. No, depends on the boundary condition, periodic BC for example does not result in a tridiagonal system

Question 4: Numerical Quadrature

In the next multiple choice questions, choose the correct(s) answer(s).

a) The two-segment trapezoidal rule is able to exactly integrate at most polynomials of order:

- ■ one
- ☐ two
- ☐ three
- ☐ four

the two-segment trapezoidal rule has the same order as the one-segment trapezoidal rule. Trapezoidal rule is 2nd order accurate, which means that at most it can integrate exactly first order functions.

b) For a second order (quadratic) polynomial, the two-point Gauss quadrature rule will give the same result as the integration rule:

- ☐ Rectangle Rule
- ☐ Midpoint Rule
- ■ Simpson's Rule
- ☐ Trapezoidal Rule

The two-point Gauss quadrature is constructed to be accurate for third order polynomials. The Simpson's rule approximates the integrand using a quadratic Lagrange polynomial. Thus they will both integrate quadratic polynomials exactly.

c) Deriving a three-point Gauss quadrature requires computing _____ unknowns:

- ☐ 2
- ☐ 3
- ☐ 4
- ■ 6

For three point quadrature rule we need three point function evaluations and three weights, therefore in total 6 unknowns.

Question 5: Romberg extrapolation and adaptive integration

a) The approximation error of an integral with n intervals of width h is given by $I_0^n = I - c_1h^3 - c_2h^6 - c_3h^9 + \dots$. Using Richardson extrapolation and the integral evaluated at twice the intervals I_0^{2n} we can eliminate the $\mathcal{O}(h^3)$ terms. Select the correct expression(s) for the error of the resulting approximation I_1^n.

- ☐ $I_1^n = I + \frac{1}{4}c_2h^6 + \frac{9}{16}c_3h^9 \dots$
- ☐ $I_1^n = I + \frac{1}{4}c_2h^6 + \frac{9}{32}c_3h^9 \dots$
- ■ $I_1^n = I + \frac{1}{8}c_2h^6 + \frac{9}{64}c_3h^9 \dots$ Solution of the calculation $I_1^n = \frac{8I_0^{2n} - I_0^n}{7}$.
- ☐ $I_1^n = I + \frac{1}{8}c_2h^6 + \frac{9}{128}c_3h^9 \dots$

b) Mark the correct statement(s) regarding adaptive integration:

- ■ Adaptive integration can give an equispaced (uniform) subdivision of the whole interval. E.g. when integrating a constant function.
- ■ Adaptive integration can give a non-uniform subdivision of the whole interval. When integrating a function with changing curvature.
- ☐ In areas of large function values we expect a more fine grained subdivision of the integration interval than in areas of small function values. Subdivision depends on curvature.
- ☐ Using adaptive integration, the desired accuracy is always smaller than the integration error. No, the stopping criterion is based on an 'error estimate' and hence it is not a strict bound.

Question 6: Probability Theory and Monte Carlo Integration

In the next multiple choice questions, choose the correct answer(s).

a) Mark the correct statement(s) regarding probability density functions.

- ■ The probability density function is always non-negative. Definition of pdf.
- ☐ The standard deviation of a random variable is always smaller than the expectation value.
- ■ Continuous probability density functions integrate up to one. Definition of pdf.
- ☐ The cumulative distribution function F is strictly monotonically increasing (i.e. $F(a) > F(b)$ if $a > b$). Wrong because of 'strictly'.

b) Assume a uniform distribution $\mathcal{U}([0,1])$ which is defined in an interval $[0,1]$. The variance of a random variable sampled from this probability distribution is

- ☐ 1
- ☐ $\frac{1}{3}$
- ☐ $\frac{1}{6}$
- ■ $\frac{1}{12}$ The solution when calculating the variance for uniform distribution on (0,1).

c) The Monte Carlo method for integration is more efficient than the Simpson's rule for any dimension higher than

- ☐ 1
- ☐ 2
- ☐ 4
- ■ 8The order of Simpson's rule in d dimension is $4/d$, where for Monte Carlo integration the order is $1/2$.

d) In order to reduce the error ϵ_M of the Monte Carlo integral estimator by a factor of 10, how many more samples are required?

- ☐ $\sqrt{10}$
- ☐ 10
- ■ 100 Error behaves as $M^{-1/2}$.
- ☐ 1000

e) Using the inverse transform sampling method, we wish to generate a random sample X sampled from the Bernoulli distribution with $P(X = 1) = \frac{2}{3}$ and $P(X = 0) = \frac{1}{3}$. First, we generated a sample from the uniform distribution $u = 0.6$. The corresponding sample X evaluates as

- ☐ 0
- ☐ 0.6
- ☐ 2/3
- ■ 1 $P(X = 1)$ for $u > 1/3$.

Numerical Problems [70 Points]

Question 7: Linear Least Squares [10 Points]

Assume you are given the following data with time t and distance d for a car passing by:

$t[s]$	1	2	3	4	5
$d[m]$	4.5	9	10.5	16	17

The distance for a car with velocity v starting from an initial position d_0 takes the form

$$d(t; d_0, v) = d_0 + vt\,. \tag{1}$$

Perform a linear-least squares fit to determine the unknown coefficients $d_0^\star, v^\star$.

The normal equation reads

$$A^\top w = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{pmatrix} \begin{pmatrix} d_0 \\ v \end{pmatrix} = \begin{pmatrix} 4.5 \\ 9 \\ 10.5 \\ 16 \\ 17 \end{pmatrix} = y \qquad (1 \text{ pt})$$

In order to invert it we have to multiply it from the left side with $A^\top$ and compute the resulting products

$$A^\top A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{pmatrix} = \begin{pmatrix} 5 & 15 \\ 15 & 55 \end{pmatrix} \qquad (2 \text{ pt})$$

and

$$A^\top y = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} \begin{pmatrix} 4.5 \\ 9 \\ 10.5 \\ 16 \\ 17 \end{pmatrix} = \begin{pmatrix} 57 \\ 203 \end{pmatrix} \qquad (2 \text{ pt})$$

The resulting 2x2 system of equations reads

$$\begin{aligned} 5d_0 + 15v &= 57 \\ 15d_0 + 55v &= 203 \end{aligned} \qquad (1 \text{ pt})$$

If we subtract the 3 times the first equation from the second equation we obtain the value for v

$$10v = 32 \quad \Rightarrow \quad v = \frac{32}{10} \qquad (2 \text{ pt})$$

We can now plug this into the first or second equation we find

$$5d_0 + 15\frac{32}{10} = 57 \text{ or } 15d_0 + 55\frac{32}{10} = 203 \quad \Rightarrow \quad d_0 = \frac{9}{5} \qquad (2 \text{ pt})$$

Question 8: Newtons Method [10 Points]

You are given the function

$$f(x) = 3x^2 - 2\sin(x)\,, \tag{2}$$

for which you want to find the root $x^\star$, i.e.

$$f(x^\star) = 0\,. \tag{3}$$

a) Write down the Newton iteration and compute the derivative $f'(x)$ of the function $f(x)$.

Newtons method is given by the iteration

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad (1 \text{ pt})$$

The gradient of the function from equation 2 is

$$f'(x) = 6x - 2\cos(x) \qquad (1 \text{ pt})$$

b) Given the initial guess $x_0 = \pi/2$, perform an iteration of Newtons method.

Using the initial guess $x_0 = \pi/2$ the function from equation 2 can be evaluated as

$$f(\pi/2) = \frac{3\pi^2}{4} - 2\sin(\pi/2) = \frac{3\pi^2}{4} - 2\,. \qquad (1 \text{ pt})$$

The gradient of the function in equation 2 at the initial guess is

$$f'(\pi/2) = 3\pi - 2\cos(\pi/2) = 3\pi \qquad (1 \text{ pt})$$

Thus we can compute the first Newton step as

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = \frac{\pi}{4} + \frac{2}{3\pi} \qquad (1 \text{ pt})$$

c) Assume you are given the following iterations of a non-linear solver:

x_2	x_3	x_4	$\cdots$	$x^\star$
0.7312	0.6386	0.6245	$\cdots$	0.6242

Provide an crude estimate of the order of convergence based on these iterations and make an educated guess what method produced this sequence.

Hint: Take into account the order of magnitude, i.e. $0.03 \approx 10^{-2}$.

We can estimate the order of the method using

$$r \approx \frac{\log \left| \frac{e_{k+2}}{e_{k+1}} \right|}{\log \left| \frac{e_{k+1}}{e_k} \right|}, \qquad (1 \text{ pt})$$

where $\epsilon_k = x_k - x^\star$ (1pt). From the given data we can compute

$$\epsilon_2 = 0.107\,, \quad \epsilon_3 = 0.0144\,, \qquad \epsilon_4 = 0.0003 \qquad (1 \text{ pt})$$

As we have no calculator at hand we approximate

$$\log \left| \frac{e_4}{e_3} \right| \approx 2\,, \quad \log \left| \frac{e_3}{e_2} \right| \approx 1\,, \qquad (1 \text{ pt})$$

from which we conclude that the order is approximately 2 and thus the sequence most probably steams from Newtons method (1pt if correct)

Question 9: Cubic Splines [25 Points]

We are trying to fit the simple cubic function $f(x) = x^3$ with cubic spline interpolation, using three datapoints sampled from the function: (x_i, y_i), with $i = 0, 1, 2$: $[(0,0), (1,1), (2,8)]$.

a) We first try to interpolate through the three datapoints with cubic splines with natural boundary conditions. Write down the matrix system you need to solve for the second derivatives on the data nodes for natural boundary conditions. Derive the coefficients of the matrix system and solve the system. Finally, write down the piecewise cubic spline for the intervals denoted by the three points. (10 points))

Note: You can use the formula: $(a-b)^3 = a^3 - 3a^2b + 3ab^2 - b^3$.

Here, we are given the points: $(x_0, y_0) = (0,0)$, $(x_1, y_1) = (1,1)$, $(x_2, y_2) = (2,8)$. For any generic boundary conditions we must solve the following system:

$$\begin{bmatrix} B_0 & C_0 & 0 \\ A_1 & B_1 & C_1 \\ 0 & A_2 & B_2 \end{bmatrix} \cdot \begin{bmatrix} f_0'' \\ f_1'' \\ f_2'' \end{bmatrix} = \begin{bmatrix} D_0 \\ D_1 \\ D_2 \end{bmatrix}$$

For natural BCs, we already know that $f_0'' = 0$ and $f_2'' = 0$. (2 points)
The values for the remaining coefficients are:

$$A_1 = \Delta_1 = 1\ ,\ B_1 = 2(\Delta_1 + \Delta_2) = 4\ ,\ C_1 = \Delta_2 = 1 \text{ (1 points)}$$

$$D_1 = 6\left(\frac{y_2 - y_1}{\Delta_2} - \frac{y_1 - y_0}{\Delta_1}\right) = 6\left(\frac{8-1}{1} - \frac{1-0}{1}\right) = 36 \text{ (1 points)}$$

Solving the equation $A_1 f_0'' + B_1 f_1'' + C_1 f_2'' = D_1$ for the above derived coefficients and for $f_0'' = f_2'' = 0$, we obtain: $f_1'' = 9$. (1 point)
The piecewise cubic spline is then computed from the relation for every segment i:

$$f_i(x) = f_{i-1}'' \frac{(x_i - x)^3}{6\Delta_i} + f_i'' \frac{(x - x_{i-1})^3}{6\Delta_i} + \left(\frac{y_i - y_{i-1}}{\Delta_i} - (f_i'' - f_{i-1}'')\frac{\Delta_i}{6}\right)(x - x_{i-1}) + \left(y_{i-1} - f_{i-1}''\frac{\Delta_i^2}{6}\right) \qquad (4)$$

For the first interval the expression reads:

$$f_1(x) = f_0'' \frac{(x_1 - x)^3}{6\Delta_1} + f_1'' \frac{(x - x_0)^3}{6\Delta_1} + \left(\frac{y_1 - y_0}{\Delta_1} - (f_1'' - f_0'') \frac{\Delta_1}{6} \right) (x - x_0) + \left(y_0 - f_0'' \frac{\Delta_1^2}{6} \right) \quad \text{(2.5 points)} \tag{5}$$

$$f_1(x) = \frac{3}{2}x^3 - \frac{1}{2}x \tag{6}$$

Following the same procedure for the second interval the following expression is derived:

$$f_2(x) = f_1'' \frac{(x_2 - x)^3}{6\Delta_2} + f_2'' \frac{(x - x_2)^3}{6\Delta_2} + \left(\frac{y_2 - y_1}{\Delta_2} - (f_2'' - f_1'') \frac{\Delta_2}{6} \right) (x - x_1) + \left(y_1 - f_1'' \frac{\Delta_2^2}{6} \right) \quad \text{(2.5 points)} \tag{7}$$

$$f_2(x) = 3 - \frac{19}{2}x + 9x^2 - \frac{3}{2}x^3. \tag{8}$$

b) The resulting piecewise cubic polynomials are depicted in Figure 4 below, along with the actual function curve. We notice that the interpolating cubic spline does not fit the function very well, although we are trying to fit a cubic function with a cubic piecewise function! Why do you think this discrepancy between the two curves occurs? Can you propose a solution to this problem? (5 points)

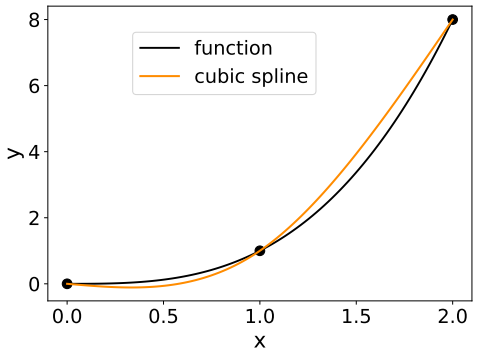


Figure 4:

The cubic spline with natural boundary conditions does not fit the actual cubic polynomial function, as for natural BCs, we force the second derivative of the function to be zero at both free ends. (2 points)
This is true for $x = 0$, as the 2nd derivative of x^3 at this point is actually 0, whereas it is not a good choice for $x = 2$, as for this point the 2nd derivative $f''(x) = 6x$ of the function is 12. (1 point)
We can solve this issue, if we modify the boundary condition at $x = 2$, so that $f_2'' = 12$. With this modification, the fit will be perfect. (2 points)

c) You then consider applying the parabolic runout boundary conditions for splines to your problem. Remember that this means that $f_0'' = f_1''$ and $f_{N-1}'' = f_N''$ for a system of N equations. Rewrite and solve the matrix system for the second derivatives at the data nodes for the parabolic runout boundary condition. What can you say regarding the resulting splines, without deriving the final formulas? How do you think that this boundary condition is going to affect the interpolating spline? (10 points)

For the given points: $(x_0, y_0) = (0, 0)$, $(x_1, y_1) = (1, 1)$, $(x_2, y_2) = (2, 8)$, the matrix system to solve for is:

$$\begin{bmatrix} B_0 & C_0 & 0 \\ A_1 & B_1 & C_1 \\ 0 & A_2 & B_2 \end{bmatrix} \cdot \begin{bmatrix} f_0'' \\ f_1'' \\ f_2'' \end{bmatrix} = \begin{bmatrix} D_0 \\ D_1 \\ D_2 \end{bmatrix}$$

The second row $A_1 f_0'' + B_1 f_1'' + C_1 f_2'' = D_1$ remains the same as before (interior node). (1 point)
We have to derive the formulas $B_0 f_0'' + C_0 f_1'' = D_0$ and $A_2 f_1'' + B_2 f_2'' = D_2$. Parabolic runout BC assumes that $f_0'' = f_1''$, and $f_1'' = f_2''$. We can easily rewrite the equations in the tridiagonal matrix system format as:

$$f_0'' - f_1'' = 0 \tag{9}$$
$$f_1'' - f_2'' = 0 \text{ (2 points)} \tag{10}$$

by simply setting $\tilde{B}_0 = \tilde{A}_2 = 1$, $\tilde{C}_0 = \tilde{B}_2 = -1$ and $\tilde{D}_0 = \tilde{D}_2 = 0$. (1 point)
The resulting matrix system is:

$$\begin{bmatrix} 1 & -1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} f_0'' \\ f_1'' \\ f_2'' \end{bmatrix} = \begin{bmatrix} 0 \\ 36 \\ 0 \end{bmatrix} \text{ (2 points)}$$

The system is easy to solve, since $f_0'' = f_1'' = f_2''$. Solving the equation system, we obtain from row 1:

$$f_0'' + 4f_1'' + f_2'' = 36 \tag{11}$$
$$6f_1'' = 36 \text{ (2 points)} \tag{12}$$

Therefore the solution is: $f_0'' = f_1'' = f_2'' = 6$ (1 point)
The solution of the matrix system results in a piecewise spline which has the same second derivative across the segments. This can only mean that the second derivative is constant, and therefore, the resulting spline is of 2nd order. The parabolic runout BC in this case results in parabolic piecewise polynomials and it does not improve the fit to the actual function $f(x) = x^3$. (2 points)

Question 10: Numerical Quadrature [10 points]

You are asked to estimate the water flow rate in a pipe of radius $R = 2$ m. The flow rate Q is defined as the integral of the velocity along the pipe circumference over the radial direction, where $r = 0$ is the center of the pipe and $r = R = 2$ is the end point, i.e. the radius of the pipe.

$$Q = \int_0^2 2\pi r \nu \mathrm{d}r \tag{13}$$

You do not know how the velocity ν varies along the radial direction and due to technical reasons you are asked to use exactly two measurement points of the velocity along the radial direction of the pipe.

a) Which method of numerical integration do you choose in order to most accurately predict the water flow rate, given that you only are allowed to use two probing points?

Gauss quadrature, and more specifically, the two-point gauss quadrature is able to approximate the flow rate with up to third order accuracy, therefore higher accuracy than the known trapezoidal and midpoint rules, which also make use of two probing points. Therefore, 2-point gauss quadrature is most appropriate for this task, as it will yield the highest accuracy for the estimation of the integral than any other rule. (2 points)

b) Compute the two probing points you need to design your measuring system, i.e. r_1, r_2 with $r_1 < r_2$.

Note: you may use the fact that $\frac{1}{\sqrt{3}} = 0.58$

According to the lecture notes, the two-point gauss quadrature takes place with the function evaluations taken at:

$$r_1 = \frac{b-a}{2}\left(-\frac{1}{\sqrt{3}}\right) + \frac{b+a}{2} \tag{14}$$

$$r_2 = \frac{b-a}{2}\left(\frac{1}{\sqrt{3}}\right) + \frac{b+a}{2}. \quad \text{(1 point)} \tag{15}$$

Given that $[a, b] = [0, 2]$, the two points are:

$$r_1 = 1 - \frac{1}{\sqrt{3}} \qquad = 0.42 \text{ m} \tag{16}$$

$$r_2 = 1 - \frac{1}{\sqrt{3}} \qquad = 1.58 \text{ m. (2 points)} \tag{17}$$

c) You end up manufacturing your measuring system and you probe the velocity at the aforecalculated locations $\nu_1 = \nu(x_1) = 1$ m/s and $\nu_2 = \nu(x_2) = 2$ m/s. Compute the water flowrate for your measurement conditions.

Since the flowrate is given as the integral: $Q = \int_0^2 2\pi r \nu \mathrm{d}r$, the function we are integrating is: $f(r) = 2\pi r \nu$.

Fron the lecture notes, we know that an integral approximated with the two-point gauss rule is approximated as:

$$\int_a^b f(r)\mathrm{d}r \approx c_1 f(r_1) + c_2 f(r_2)\ , \ \text{(1 point)} \tag{18}$$

where

$$r_1 = \frac{b-a}{2}\left(-\frac{1}{\sqrt{3}}\right) + \frac{b+a}{2} \tag{19}$$

$$r_2 = \frac{b-a}{2}\left(\frac{1}{\sqrt{3}}\right) + \frac{b+a}{2} \tag{20}$$

the abscissas calculated before, i.e. $r_1 = 1 - \frac{1}{\sqrt{3}}$ m and $r_2 = 1 + \frac{1}{\sqrt{3}}$ m and

$$c_1 = \frac{b-a}{2} \qquad = 1 \tag{21}$$

$$c_2 = \frac{b-a}{2} \qquad = 1. \text{ (1 point)} \tag{22}$$

Inserting these, along with the probed velocities in the function formula:

$$f(r_1) = 2\pi r_1 \nu_1 \qquad = 2\pi \left(1 - \frac{1}{\sqrt{3}}\right) 1 \tag{23}$$

$$f(r_2) = 2\pi r_2 \nu_2 \qquad = 2\pi \left(1 + \frac{1}{\sqrt{3}}\right) 2. \text{ (2 points)} \tag{24}$$

Therefore, the integral, i.e. the flowrate is estimated as:

$$Q = c_1 f(r_1) + c_2 f(r_2) \tag{25}$$

$$= 1 \cdot 2\pi \left(1 - \frac{1}{\sqrt{3}}\right) \cdot 1 + 1 \cdot 2\pi \left(1 + \frac{1}{\sqrt{3}}\right) 2 \tag{26}$$

$$= 6\pi + \frac{2\pi}{\sqrt{3}} \tag{27}$$

$$= 22.5 \text{ m}^3/\text{s (1 point)} \tag{28}$$

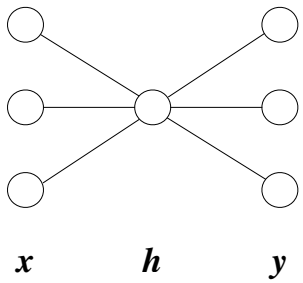
Question 11: Backpropagation and neural networks [15 points]

A neural network architecture is assembled using a 3-neuron input layer, a single-neuron hidden layer, and a 3-neuron output layer. It employs identity and ReLU activation functions on the hidden and output layers (respectively), but does not make use of any biases.
The ReLU function is defined as:

$$\phi(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

a) Draw the network, and label the input layer $\boldsymbol{x}$, hidden layer $\boldsymbol{h}$, and output layer $\boldsymbol{y}$.

(1.0 Point)



b) Perform a forward pass using a sample $\boldsymbol{x}^{(1)} = [4, 2, 1]$, and compute the reconstruction loss $E^{(1)} = \|\boldsymbol{x}^{(1)} - \boldsymbol{y}^{(1)}\|_2^2$. The current weights of the hidden and output layers are given as follows:

$$\mathbf{W}^h = \begin{bmatrix} 3 & -2 & -4 \end{bmatrix}, \mathbf{W}^y = \begin{bmatrix} 2 \\ 1 \\ -3 \end{bmatrix}$$

The forward propagation expressions are given by

$$z^h = \phi_1(a^h) \quad \text{with} \quad a^h = \sum_{j=1}^{3} W_j^h x_j \quad \text{and} \quad \phi_1(x) = x \tag{29}$$

$$z_i^y = \phi_2(a_i^y) \quad \text{with} \quad a_i^y = W_i^h z^h \quad \text{for } i = 1, 2, 3 \quad \text{and} \quad \phi_2(x) = \text{ReLU}(x) \tag{30}$$

(2.0 Points)

hence the hidden node value is computed as

$$h = z^h = \phi_1(a^h) = \begin{bmatrix} 3 & -2 & -4 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = 4 \tag{31}$$

(1.0 Point)

and the output becomes

$$\boldsymbol{y} = \boldsymbol{z}^y = \phi_2(\boldsymbol{a}^y) = \phi_2(\begin{bmatrix} 2 \\ 1 \\ -3 \end{bmatrix} \cdot 4) = \begin{bmatrix} 8 \\ 4 \\ 0 \end{bmatrix} \tag{32}$$

(1.0 Point) resulting in an error of

$$E^{(1)} = (4-8)^2 + (2-4)^2 + (1-0)^2 = 21 \tag{33}$$

(0.5 Point)

c) Perform the weight update rule for $W_2^{y,new}$ given the current weight using a gradient descent step with $\eta = 0.10$.

The weight update rule is given by

$$W_2^{y,new} = W_2^y - \eta \frac{\partial E^{(1)}}{\partial W_2^y} \tag{34}$$

(1.0 Point) which entails the computation of the error gradient. Using the chain rule, we have the expression

$$\frac{\partial E}{\partial W_2^y} = \frac{\partial E}{\partial y_2} \frac{\partial y_2}{\partial a_2^y} \frac{\partial a_2^y}{\partial W_2^y} \tag{35}$$

(1.0 Point)

The first term is given by

$$\frac{\partial E}{\partial y_2} = -2 \cdot (x_2 - y_2) \tag{36}$$

(0.5 Point) The next one involves the variation of the activated node value versus the deactivated one i.e. we make use of the derivative of the ReLU activation function

$$\frac{\partial y_2}{\partial a_2^y} = \begin{cases} 1, & \text{if } a_2^y \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{37}$$

(0.5 Point) and the remaining term is

$$\frac{\partial a_2^y}{\partial W_2^y} = z^h \tag{38}$$

(0.5 Point)

Hence we obtain

$$\frac{\partial E}{\partial W_2^y} = (2 \cdot (2-4)) \cdot (1) \cdot (4) = +16 \tag{39}$$

(0.5 Point)

Finally the weight update rule results in:

$$\begin{aligned} W_2^{y,new} &= W_2^y - \eta \frac{\partial E^{(1)}}{\partial W_2^y} \\ &= 1 - 0.10 \cdot 16 \\ &= -0.6 \end{aligned} \tag{40}$$

(1.0 Point)

d) A new fully-connected linear layer is added right after the input layer (with the same number of nodes as the input). Comment on whether we should expect enhanced expressiveness of the network.

In this configuration, we are stacking two linear layers next to each other. In practice, a single linear layer is enough to approximate the same class of functions. As a result, the network doesn't gain anything as far as learning representations. Rather, it results in increased training runtime and a waste of computational resources. (1.5 Point)

e) We take the network from (d) and decide to add biases in every layer, count the total number of parameters of the resulting network.

For feed-forward neural networks, we can count the number of parameters as follows:

$$\begin{aligned} n_{params} = n_{connections} + n_{biases} &= (i \cdot h_1 + h_1 \cdot h_2 + h_2 \cdot y) + (h_1 + h_2 + y) \\ &= (3 \cdot 3 + 3 \cdot 1 + 1 \cdot 3) + (3 + 1 + 3) \\ &= 22 \end{aligned} \tag{41}$$

(1.5 Point)

f) Taking the original network from (a), do you think the architecture is well-suited for an autoencoder?

This is an odd configuration where the network uses linear layers to encode information to the latent space and non-linear ones to decode it. Its applicability is limited.

(1.5 Point)

Pseudocode [20 Points]

Question 12: Pseudocode for Romberg integration [5 Points]

Below in Algorithm 1 you find a pseudocode for the Romberg integration. Each of the three main blocks contain one error. Identify the three errors and correct them.

Algorithm 1 Romberg integration

Input:

function $f(x)$
interval boundaries a, b
number of iterations K

Output:

$I_K^1 = \texttt{integral}[K, 0]$ approximation to the integral $\int_a^b f(x)\ \mathrm{d}x$

Steps:

$\texttt{maxNumIntervals} \leftarrow 2^K$

// Precompute and store function evaluations (Block 1)
$\texttt{hmin} \leftarrow (b-a)/\texttt{maxNumIntervals}$ (0.5+0.5 points)
for $i \leftarrow 0, \ldots, \texttt{maxNumIntervals}$ **do**
  $\texttt{fvalues}[i] \leftarrow f(a + i * \texttt{hmin})$
end for

// Compute level 0 integrals (Block 2)
for $r \leftarrow 0, \ldots, K$ **do** // refinement
  $\texttt{numIntervals} \leftarrow 2^r$
  $\texttt{step} \leftarrow 2^{K-r}$ // step between two function evaluations for this refinement
  $\texttt{result} \leftarrow 0$
  for $i \leftarrow \texttt{step}, 2 * \texttt{step}, 3 * \texttt{step}, \ldots, \texttt{maxNumIntervals} - \texttt{step}$ **do** (1+1 points)
    $\texttt{result} \leftarrow \texttt{result} + \texttt{fvalues}[i]$
  end for
  // composite trapezoidal rule:
  $\texttt{integral}[0, r] \leftarrow 0.5 \frac{b-a}{\texttt{numIntervals}} (\texttt{fvalues}[0] + \texttt{fvalues}[\texttt{maxNumIntervals}] + 2 * \texttt{result})$
end for

//Advance to higher precision according to Romberg (Block 3)
for $l \leftarrow 1, \ldots, K$ **do** //level
  for $r \leftarrow 0, \ldots, K - l$ **do** // refinement
    $\texttt{integral}[l, r] \leftarrow \frac{4^l * \texttt{integral}[l-1, r+1] - \texttt{integral}[l-1, r]}{4^l - 1}$ (1+1 points)
  end for
end for

Question 13: PCA for Dimensionality Reduction [15 Points]

Given a sample dataset $X = [x_1, x_2, ..., x_N]$ with $x_i \in \mathbb{R}^D$, write a pseudocode showing all the steps needed to perform a dimensionality reduction of the dataset using PCA. Your goal is to keep at least 90% of the original variance. Make sure you specify the dimensions of each matrix or vector in your pseudocode, and clearly indicate the input, output and steps of the algorithm. To assist you in your work, you are provided with the following helper functions:

- `computeMean(X, shape, dim)`: Computes the mean of matrix `X` of specified `shape` over the dimensions `dim`. Specify the last two arguments as a tuple e.g. `shape`=(5, 4, 2). `dim`=2.
- `transpose(X)`: Returns the transpose of an input matrix `X`.
- `L, V = eigenDec(X)`: Computes the eigendecomposition of an input matrix `X`, returning a vector of eigenvalues `L`, as well as a matrix `V` containing the associated eigenvectors (in no predefined order).
- `newVec, shiftVec = sortVec(X, ord)`: Sorts a vector in ascending or descending order. Specify `ord=asc` or `ord=desc` if you wish to sort it in ascending or descending order (respectively). An indexation vector of the shifting (`shiftVec`) is also given. For example, after sorting the initial vector `[5, 6, 8, 7]` in descending order, the vector `newVec` become `[8, 7, 6, 5]`, and the corresponding the shifting indexation vector is `[2, 3, 1, 0]` - index 2 is placed first because it contains the largest value in the input vector.
- `vec_sum(a)`: Returns the sum of the elements of a vector a. If a scalar is used as input, the output will be that same scalar.

Algorithm 2 PCA for dimensionality reduction

Input:

X, {Dataset, $\in \mathbb{R}^{N \times D}$} (0.5 point)

Output:

r, {Number of PCA modes to keep} (0.5 point)
V_r, {Matrix containing the r first PCA components, $\in \mathbb{R}^{D \times r}$} (0.5 point)
Y_r, {Compressed representation of the data, $\in \mathbb{R}^{r \times N}$} (0.5 point)

Steps:

$\bar{x} =$ `computeMean(X, (N, D), 0)`, $\bar{x} \in \mathbb{R}^D$ (0.5 + 0.5 point)
for i in range(N) **do**
    $\hat{X}[i,:] = X[i,:] - \bar{x}$, $\hat{X} \in \mathbb{R}^{N \times D}$(0.5+0.5 point)
end for
$C = \frac{1}{N-1}$`transpose(`$\hat{X}$`)`$\hat{X}, C \in \mathbb{R}^{D \times D}$ (0.5+0.5+0.5 point)
L, V = `eigenDec`(C), $L \in \mathbb{R}^D$, $V \in \mathbb{R}^{D \times D}$ (0.5 + 0.5 + 0.5 point)
L_sorted, shiftVec = `sortVec`(L, ord=desc) (1 point)
for i in range(D) **do**
    V_sorted[:, i]=V[:, shiftVec[i]] (1.5 point)
end for
tot_variance= `vec_sum`(L_sorted) (1 point)
for i in range(D) **do**
    partial_sum= `sum`(L_sorted[:i]) (1 point)
    if partial_sum≥0.9*tot_variance **then**
        r = i (1 point)
        Break (0.5 point)
    end if
end for
V_r=V_sorted[:, :r] (1 point)
Y_r = `transpose`(V_r)`transpose`(X) (1 point)

Good luck!