ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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

Spring semester 2020

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

Solution

Issued: Saturday, 03.08.2020

Multiple Choice Questions [30 points]

There is a total of 20 correct answers over all multiple choice questions. Each correct answer gives you 1.5 point. Any missing answer gives you 0 point. **For each wrong answer,** 1.5 **point is deducted**. The minimal number of points for your answers to **all multiple choice questions** is 0.

Question 1: Nonlinear solvers

a) Which statement(s) about the convergence of Newton's method is/are correct? (Assume a good initial condition has been chosen)

- ■ For $f(x) = x^2 - 3x + 2$ the method converges quadratically.
- ☐ For $f(x) = x^{1/3}$ the method converges linearly. Update rule is $x_{i+1} = -2x_i$, solution diverges
- ☐ For $f(x) = x^2 - 4x + 4$ the method converges quadratically. Multiplicity of root is $m = 2$, converges linearly
- ■ For $f(x) = x$ the method converges in one step.

b) Which of the following statement(s) about nonlinear solvers is/are correct?

- ■ The Bisection method is guaranteed to converge.
- ☐ The order of convergence of nonlinear solvers cannot be estimated numerically. Can be estimated numerically by tracking error.
- ☐ The Bisection method converges faster the Newton's method. Bisection order 1, Newton 2
- ☐ The Secant method can only be used when the closed form of $f'(x)$ is known.

c) You goal is to find the minimum of the function $g(x, y) = x^2 + 2xy + y^3$. You choose to use the Newton-Raphson method as: $\mathbf{x_{k+1}} = \mathbf{x_{k+1}} - J^{-1}(\mathbf{x_k})F(\mathbf{x_k})$. Select the correct expressions for $F(\mathbf{x})$ and $J(\mathbf{x})$: :

- ☐ $F(\mathbf{x}) = \left(x^2 + 2xy + y^3\right)$, $J(\mathbf{x}) = \begin{pmatrix} 2x + 2y \\ 2x + 3y^2 \end{pmatrix}$
- ☐ $F(\mathbf{x}) = \begin{pmatrix} 2x + 2y \\ 2x + 3y^2 \end{pmatrix}$, $J(\mathbf{x}) = \begin{pmatrix} 2 & 6y \\ 2 & 2 \end{pmatrix}$

☐ $F(\mathbf{x}) = \begin{pmatrix} 2x + 3y^2 \\ 2x + 2y \end{pmatrix}$, $J(\mathbf{x}) = \begin{pmatrix} 2 & 6y \\ 2 & 2 \end{pmatrix}$

■ $F(\mathbf{x}) = \begin{pmatrix} 2x + 2y \\ 2x + 3y^2 \end{pmatrix}$, $J(\mathbf{x}) = \begin{pmatrix} 2 & 2 \\ 2 & 6y \end{pmatrix}$

d) The update rule for finding the root R by solving the equation $x^2 - R = 0$ using Newton's method is:

☐ $x_{i+1} = \frac{x_i}{2}$

■ $x_{i+1} = \frac{1}{2}(x_i + \frac{R}{x_i})$

☐ $x_{i+1} = \frac{1}{2}(x_i + \frac{R}{2})$

☐ $x_{i+1} = \frac{3x_i}{2}$

Question 2: Interpolation with Lagrange Polynomials and Cubic Splines

a) Which of the following statement(s) about Lagrange interpolation is/are correct?

☐ The resulting function of a Lagrange interpolation through 3 sample points of the function $f(x) = \frac{1}{3}x^3 + 5x + 2$ is an exact fit to the original function $f(x)$. (we need 4 points from the function to construct a perfect fit.)

☐ The first Lagrange basis function through the points $(x_1, y_1) = (2, 0.5)$ and $(x_2, y_2) = (3, 1)$ is $l_1(x) = \frac{x-3}{-0.5}$. ($l_1(x) = \frac{x-3}{-1}$.)

■ When fitting N noisy data points, the resulting function from the Lagrange interpolation is the same as the Least-Squares solution using a polynomial basis of degree N-1.

☐ If we gather data from a 3rd order polynomial, where noise can be present, and fit the data using Lagrange interpolation, we will always get the same interpolating function for a number of samples points $N \geq 4$. (this doesn't happen for noisy data.)

b) The figures below show three different cubic spline fits for the same data points. The only difference between them is the boundary condition, applied to both ends for every case.

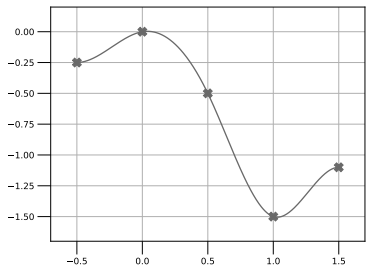
Select the correct combination that matches the letter of the picture to the boundary conditions it was generated with:

☐ (a): parabolic runout, (b): natural, (c): clamped

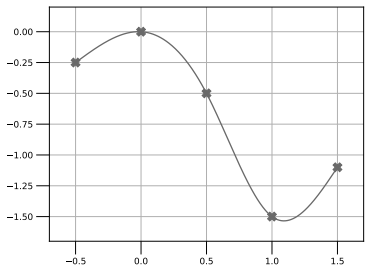
☐ (a): clamped, (b): parabolic runout, (c): natural

☐ (a): natural, (b): clamped, (c): parabolic runout

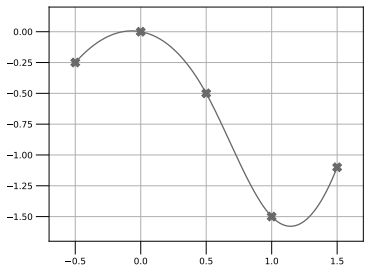
■ (a): clamped, (b): natural, (c): parabolic runout



(a)



(b)



(c)

c) Which of the following statement(s) is/are correct for Cubic Spline Interpolation through 6 points $\{(x_i, y_i)\}_{i=1,\ldots,6}$, with clamped boundary conditions at both ends?

- ■ The first derivatives of the fitted curves are continuous at the interior data points.
- ☐ The matrix system we need to solve can be reduced to a 4×4 tridiagonal system. (not for clamped ends)
- ■ If we work with equidistant segments, the coefficients A_i, B_i, C_i, D_i of the matrix system to be solved are the same for all interior points.
- ■ The second derivatives at the end points are continuous.
- ☐ The second derivatives are polynomials of order 2. (for cubic splines the 2nd derivative is always linear)
- ☐ The interpolation function will be a piecewise cubic polynomial, defined over 6 segments. (the interpolating function is defined in 5 segments)

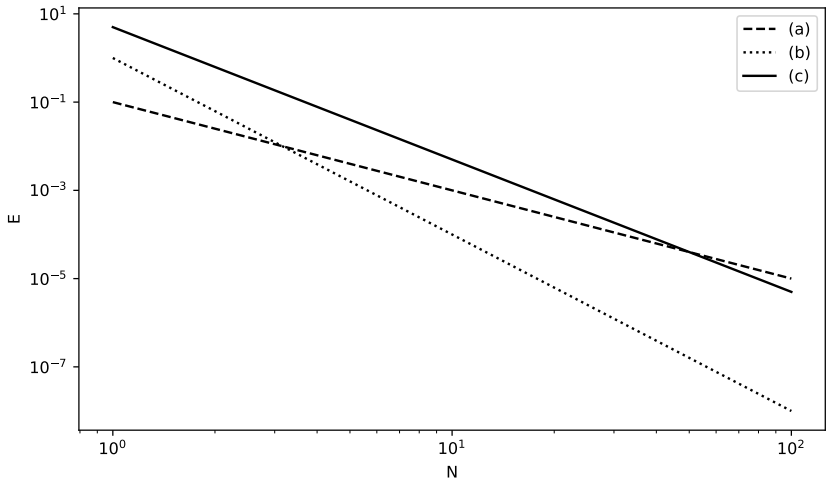
Question 3: Numerical Quadrature

a) Which of the following statement(s) about numerical quadrature is/are correct?

- ☐ Simpson's rule is able to perfectly integrate a polynomial of order 4 or less.
- ■ The order of accuracy of a numerical quadrature scheme over an interval $[a, b]$, split into N sub-intervals $[x_i, x_{i+1}]$ of width $\Delta_i = \frac{b-a}{N}$, is one less than the order of accuracy on one of the sub-interval $[x_i, x_{i+1}]$.
- ☐ The mid point rule has a higher order of accuracy than the trapezoidal rule.
- ■ When using Simpson's rule over an interval $[a, b]$, to reduce the error on I by a factor 1000, 10 times more data points must be used.

b) The figure below shows the evolution of the error with respect to the number of sub-intervals N for three quadrature rules over an interval $[a, b]$. Select the correct combination of quadrature rule and plot label:

- ☐ (a): Trapezoidal, (b): Simpson, (c): Rectangle
- ☐ (a): Simpson, (b): Rectangle, (c): Trapezoidal
- ■ (a): Rectangle, (b): Simpson, (c): Midpoint
- ☐ (a): Midpoint , (b): Trapezoidal, (c): Rectangle



c) Which of the following statement(s) about Romberg integration $I_k^n = \frac{4^k I_{k-1}^{2n} - I_{k-1}^n}{4^k - 1}$ where I_0^n is the trapezoidal rule over n subintervals is/are correct?

- [x] I_1^n is 4th order accurate.
- [ ] If you have already computed $I^2 n_0$, you need n additional function evaluations to compute I_0^n.No, you have already done all function evaluations
- [x] Number of function evaluations for I_2^2 is twice as that of I_2^1.
- [ ] To compute I_3^1, three intervals are sufficient, that is, we only need to compute I_0^1, I_0^2, I_0^4.

d) Which of the following statement(s) about Gauss quadrature is/are true?

- [ ] The two-point Gauss quadrature is identical to trapezoidal rule.
- [ ] The derivation of the two-point Gauss quadrature requires finding 5 unknowns.
- [ ] Gauss quadrature is a method to find the optimal number of quadrature points on an interval $[a, b]$.
- [x] The two-point Gaussian quadrature recovers integration of third order polynomial exactly.

e) Which of the following statement(s) about adaptive integration is/are correct?

- [x] The error $\epsilon(h/2) \approx G(h/2) - G(h)$, derived from Richardson extrapolation, can be used as a stopping criterion for the refinement process.
- [ ] The function f we wish to integrate must have an analytical expression to estimate the local numerical error.
- [ ] The adaptive refining procedure stops automatically without an external criterion.
- [x] Refining the grid locally does not change the order of accuracy of the underlying integration scheme.

f) For Monte Carlo Integration $I = \int_A f(x)\mathrm{d}x \approx I_M = \frac{1}{M}\sum_{i=1}^{M} f(x^{(i)})$ where $x^{(i)}$ are uniformly sampled on A, which of the following statement(s) is/are true?

- [ ] Monte Carlo Integration suffers the Curse of Dimensionality and thus can only be applied for low dimensional problems.
- [x] The error is estimated by the variance of the Monte-Carlo Integral $\varepsilon_M = \sqrt{\mathrm{Var}\left[I_M\right]}$.
- [x] For Monte Carlo integration the error estimate behaves as $\varepsilon_M = \mathcal{O}(M^{-1/2})$.
- [ ] The error for Monte Carlo integration is computed as the distance between the true integral I and the Monte Carlo estimate $\varepsilon_M = |I - I_M|$.

Numerical Problems [70 points]

Question 4: Linear Least Squares [15 Points]

You have gathered the following data:

t	1	2	3
x	2	5	10

Based on you knowledge of kinematics you expect the relation between x and t to follow:

$$x(t) = \frac{1}{2}gt^2 + x_0\,. \tag{1}$$

a) Using a linear least squares fit, estimate the value of the intercept x_0 and the gravitational acceleration g.

We can rewrite the problem in matrix form as:

$$A\mathbf{w} = \mathbf{x}\ , \quad \begin{bmatrix} 1 & t_1^2 \\ 1 & t_2^2 \\ 1 & t_3^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

(2 point)

With:

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 4 \\ 1 & 9 \end{bmatrix} \text{ and } x = \begin{bmatrix} 2 \\ 5 \\ 10 \end{bmatrix}$$

(1 point)

The goal is to solve the normal equation:

$$\left(A^T A\right)\mathbf{w} = A^T\mathbf{x}$$

(2 point)

With

$$A^T A = \begin{bmatrix} 3 & 14 \\ 14 & 98 \end{bmatrix} \text{ and } A^T\mathbf{x} = \begin{bmatrix} 17 \\ 112 \end{bmatrix}$$

(2 point)

The solution to this system gives:

$$\mathbf{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

(4 point)

Thus $x_0 = 1$ and $g = 2$. (1 point)

b) Assume that the gravitational field varies as $g(t) = \sin(w_g t)$. Keeping the same kinematic formulation with the new expression for g as in eq.1, can you estimate x_0 and w_g using linear least squares ? Justify you answer.

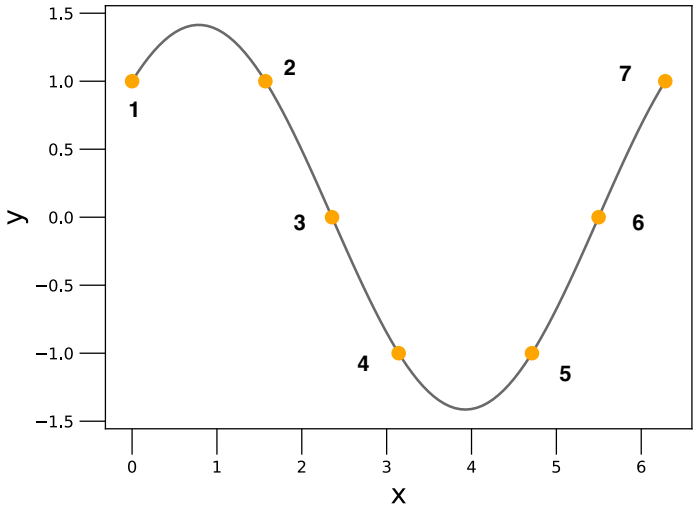
No this problem is not solvable by linear least squares as the parameters w_g doest not enter linearly in the formulation for $x(t)$.

Even with additional data it is not possible.

(3 point)

Question 5: Minimum of a graph function [25 Points]

You are given the following graph of a function $f(x)$ that is a linear combination of the functions $\cos(x)$ and $\sin(x)$.



Moreover, you are given a set of points $\{(x_i, y_i)\}_{i=1}^{7}$ with $y_i = f(x_i)$, given in the table below. Further assume that the function is 2π-periodic.

i	1	2	3	4	5	6	7
x_i	0	$\frac{\pi}{2}$	$\frac{3\pi}{4}$	π	$\frac{3\pi}{2}$	$\frac{7\pi}{4}$	2π
y_i	1	1	0	-1	-1	0	1

You want to fit the data and to find the minimum of the function $f(x)$. Two interpolation methods are available : Lagrange interpolation and Cubic Spline interpolation.

a) Using the Lagrange interpolation method and all the available data, what is the degree of the resulting polynomial? Are there any dangers in fitting this kind of data with Lagrange interpolation?

In order to fit the curve with Lagrange interpolation, we make use of all the 7 given points, therefore, we will end up with a 6-order polynomial. (2 points)

The data of the curve originates from a function without noise, therefore, we are in no danger fitting the noise, by using Lagrange interpolation. In this case, there are also no straight parts in the boundaries that we have to worry about. In this case, the more points we use, the best the interpolating polynomial will be. (2 points)

b) Using Cubic Spline interpolation, specify the boundary conditions that you would use to fit the data from the function $f(x)$. Write down the equations you would need to solve for the second derivatives on the boundaries.

The function we are trying to fit is periodic in the boundaries $x = 0, 2\pi$. Therefore, we have to assure continuity of the first and second derivatives between the boundary points: $f'(x_1) = f'(x_7)$ and $f''(x_1) = f''(x_7)$. Also, continuity in the values of the function is required $f(x_1) = f(x_7)$. (3 points)

In this case, the equations to be solved on the boundaries are derived from the conditions:

$$f''(x_1) = f''(x_7) \Rightarrow 1 \cdot f''(x_1) - 1 \cdot f''(x_7) = 0$$

$f'(x_1) = f'(x_N) \Rightarrow$

$$-f_1''\frac{(x_2 - x_1)^2}{2\Delta_1} + 0 + \frac{y_2 - y_1}{\Delta_1} - (f_2'' - f_1'')\frac{\Delta_1}{6} = \quad (2)$$

$$0 + f_N''\frac{(x_N - x_{N-1})^2}{2\Delta_{N-1}} + \frac{y_N - y_{N-1}}{\Delta_{N-1}} - (f_N'' - f_{N-1}'')\frac{\Delta_{N-1}}{6} \quad (3)$$

$$-f_1''\frac{\Delta_1}{2} + \frac{y_2 - y_1}{\Delta_1} - (f_2'' - f_1'')\frac{\Delta_1}{6} = f_N''\frac{\Delta_{N-1}}{2} + \frac{y_N - y_{N-1}}{\Delta_{N-1}} - (f_N'' - f_{N-1}'')\frac{\Delta_{N-1}}{6} \quad (4)$$

$$-f_1''\frac{2\Delta_i}{3} - f_2''\frac{\Delta_i}{6} - f_{N-1}''\frac{\Delta_i}{6} = -\frac{y_2 - y_1}{\Delta_i} + \frac{y_N - y_{N-1}}{\Delta_i} \quad (5)$$

, using

$$f'(x) = -f_i''\frac{(x_{i+1} - x)^2}{2\Delta_i} + f_{i+1}''\frac{(x - x_i)^2}{2\Delta_i} + C_i$$

where $C_i = \frac{y_{i+1} - y_i}{\Delta_i} - (f_{i+1}'' - f_i'')\frac{\Delta_i}{6}$

So, in the end the first and last row of the matrix system to be solved are:

$$1 \cdot f''(x_1) - 1 \cdot f''(x_7) = 0$$

$$0 \cdot f_1'' + \frac{\Delta_1}{6}f_2'' + \frac{\Delta_6}{6}f_6'' + \frac{2\Delta_6}{3}f_7'' = \frac{y_2 - y_1}{\Delta_1} - \frac{y_7 - y_6}{\Delta_6}$$

(4 points)

c) Using a smart selection of data points, fit the data using Lagrange interpolation and calculate the minimum of $f(x)$.

A second order curve is already a good approximation of the local behaviour of $f(x)$ around the minimum.

From the form of the curve around the minimum, it is obvious that a 2-order polynomial would suffice to capture the behaviour of $f(x)$ around the minimum. Therefore, we select the points: $\{x_1, y_1\} = \{\frac{3\pi}{4}, 0\}$, $\{x_2, y_2\} = \{\pi, -1\}$ and $\{x_3, y_3\} = \{\frac{3\pi}{2}, -1\}$ (3 points)

We compute the basis polynomials:

$$l_1(x) = \frac{(x-\pi)(x-\frac{3\pi}{2})}{(\frac{3\pi}{4}-\pi)(\frac{3\pi}{4}-\frac{3\pi}{2})} = \frac{x^2 - \frac{5\pi}{2}x + \frac{3\pi^2}{2}}{\frac{3\pi^2}{16}} \tag{6}$$

$$l_2(x) = \frac{(x-\frac{3\pi}{4})(x-\frac{3\pi}{2})}{(\pi-\frac{3\pi}{4})(\pi-\frac{3\pi}{2})} = \frac{x^2 - \frac{9\pi}{4}x + \frac{9\pi^2}{8}}{-\frac{\pi^2}{8}} \tag{7}$$

$$l_3(x) = \frac{(x-\frac{3\pi}{4})(x-\pi)}{(\frac{3\pi}{2}-\frac{3\pi}{4})(\frac{3\pi}{2}-\pi)} = \frac{x^2 - \frac{7\pi}{4}x + \frac{3\pi^2}{4}}{\frac{3\pi^2}{8}} \tag{8}$$

(6 points)

Then, the interpolating function is written as:

$$f(x) = 0 \cdot l_1(x) + (-1) \cdot l_2(x) + (-1) \cdot l_3(x) = \frac{x^2 - \frac{9\pi}{4}x + \frac{9\pi^2}{8}}{\frac{\pi^2}{8}} - \frac{x^2 - \frac{7\pi}{4}x + \frac{3\pi^2}{4}}{\frac{3\pi^2}{8}} \tag{9}$$

(2 points)

Note that, since $y_1 = 0$, we could also omit the calculation of $l_1(x)$.
To find the minimum of the function, we need to set its first derivative equal to zero:

$$f'(x) = -\frac{2x - \frac{7\pi}{4}}{\frac{3\pi^2}{8}} + \frac{2x - \frac{9\pi}{4}}{\frac{\pi^2}{8}} = 0 \Longrightarrow x = \frac{5\pi}{4} \tag{10}$$

(3 points)

Question 6: Inverse Transform Sampling [15 points]

You are given the probability density function for the Cauchy distribution

$$p(x) = \frac{1}{\pi}\frac{1}{(1+x^2)}, \quad \forall x \in \mathbb{R}. \tag{11}$$

In class you learned that you can obtain samples from a distribution by using the inverse transform method.

a) Compute the cumulative distribution function $F(x)$ for the Cauchy-distribution

$$F(x) = \int_{-\infty}^{x} p(x')\mathrm{d}x' \tag{12}$$

You might want to use $x = \tan(\theta)$ as substitution.
We substitute $x = \tan(\theta)$ and thus find

$$\frac{\mathrm{d}x}{\mathrm{d}\theta} = \frac{1}{\cos^2(\theta)} \quad \Rightarrow \quad dx = \frac{\mathrm{d}\theta}{\cos^2(\theta)} \tag{13}$$

(2 points)

Furthermore observing that $1 + \tan^2(\theta) = 1/\cos^2(\theta)$ we realize that the integral is trivial
(2 points)

$$F(x) = \frac{1}{\pi}\int_{-\infty}^{x} \frac{1}{1+\tan(\theta)^2}\mathrm{d}x' = \frac{1}{\pi}\int_{\tan^{-1}(-\infty)}^{\tan^{-1}(x)} \mathrm{d}\theta = \frac{1}{2} + \frac{1}{\pi}\arctan(x) \tag{14}$$

(6 points)

Where we used that $\tanh^{-1}(-\infty) = -\pi/2$ (2 points)

b) Compute the inverse $F^{-1}(u)$ of the cumulative distribution function $F(x)$ found in the previous subquestion.

We start with the cumulative distribution function found in the previous subquestion

$$u = \frac{1}{2} + \frac{1}{\pi}\arctan(x) \tag{15}$$

(1 point)

Inverting this gives

$$x = \tan\left(\pi\left(u - \frac{1}{2}\right)\right) \tag{16}$$

(2 points)

Question 7: Richardson Extrapolation [15 points]

Finite differences are used to approximate derivatives in computer simulations. The central difference approximation of $f'(x)$ is expressed as:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

a) Using the Taylor expansion $f(x+h) = f(x) + hf'(x) + h^2\frac{f''(x)}{2} + h^3\frac{f'''(x)}{6} + \mathcal{O}(h^4)$, find the order of accuracy of the central difference approximation.

Start by combining

$$f(x+h) = f(x) + hf'(x) + h^2\frac{f''(x)}{2} + h^3\frac{f'''(x)}{6} + \mathcal{O}(h^4)$$

(1 point)

and

$$f(x-h) = f(x) - hf'(x) + h^2\frac{f''(x)}{2} - h^3\frac{f'''(x)}{6} + \mathcal{O}(h^4)$$

(1 point)

Into

$$f(x+h) - f(x-h) = 2hf'(x) + 2h^3\frac{f'''(x)}{6}$$

(1 point)

Dividing by h^2 gives:

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + 2h^2\frac{f'''(x)}{6}$$

(1 point)

Thus

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2)$$

(2 points)

b) Use Richardson extrapolation to derive a new method that is fourth order accurate. *You can express the new approximation as a function of* $D(h)$.

The approximation $f'(x) \approx D(h) = \frac{f(x+h)-f(x-h)}{2h}$ is second order accurate. We can write the approximation as :

$$G_1(h) = D(h) = \frac{f(x+h) - f(x-h)}{2h}$$

(2 points)

Using the formula:

$$G_n(h) = \frac{1}{2^n - 1}\left(2^n\, G_{n-1}(h/2) - G_{n-1}(h)\right)$$

(2 points)

We can get the expression for $G_2(h)$ as:

$$G_2(h) = \frac{4G_1(h/2) - G_1(h)}{3} = G + G_n(h)$$

(2 points)

The general expression of $G_n(h) = G + G_n(h^{n+1})$ does not hold in this case as all odd error terms are cancelled out in the central difference formulation.

$G_2(h)$ is thus 4^{th} order accurate.

(3 points)

Pseudocode [20 Points]

Question 8: Artificial Neural Network Training Loop [20 Points]

You want to predict the new daily cases of COVID-19 using a neural network. Assume your colleagues have already implemented a functioning artificial neural network (ANN) model with the following functions:

- `forward_pass(x)`: Given an input `x`, returns the output `y` through the neural network.
- `compute_loss(x,y)`: Computes and returns the loss (L2 norm in this case) between 2 entries `x` and `y`.
- `compute_gradients(x,y)`: Computes the gradient of `x` with respect to `y`, i.e. $\frac{\partial x}{\partial y}$.
- `update_weights(x)`: Updates the weights of the model by `x`, i.e. $\mathbf{w}^{k+1} = \mathbf{w}^k + \mathbf{x}$.

Additionally, the following helper functions have been implemented:

- `split_dataset(X,r)`: Given a dataset `X` of shape `[N,n_dim]`, where `N` is the number of samples, and `n_dim` the dimension of a sample, splits the dataset into 2 subsets of relative size r and $1-r$.
- `shuffle(X,Y)`: Shuffles the content of `X` and `Y`.
- `get_batch(X,i,n_batch)`: Returns the i-th batch of size `n_batch` from the dataset `X`.

You are also given the input dataset $X = [x_1, x_2, ..., x_N]$ where $x_i \in \mathbb{R}^d$ and the target dataset $Y = [y_1, y_2, ..., y_N]$ where $y_i \in \mathbb{R}$. The goal of the ANN is to learn the mapping $y = f_{ANN}(x)$ between the input and the target.

a) Using the functions defined above and the given datasets, write the full training loop of the ANN as a pseudocode. Make sure you that:
   - The number of training epochs, the batch size and the learning rate are defined and used in your pseudo-code.
   - The network is trained using batch SGD.
   - You have a mechanism to prevent overfitting.

Algorithm 1 ANN Training Loop

Input:

X, {Input dataset} (1 point)
Y, {Target dataset} (1 point)
n_batch, { batch size} (1 point)
n_epochs, {number of training epochs} (1 point)
η, {learning rate} (1 point)

Output:

W, {weights} or $y = f_{ann}(x)$, {the mapping} (1 point)

Steps:

X_train, X_test = split_dataset(X) Split input (0.5 point)
Y_train, Y_test = split_dataset(Y) Split output (0.5 point)
testing_loss = [ ]
for i in range(n_epochs) **do** Epoch loop (1 points)
    X_train,Y_train = shuffle(X_train,Y_train) Shuffle dataset (1 point)
    for j in range(N // n_batch) **do** Batch loop (1 point)
        X_batch,Y_batch = get_batch(X_batch,Y_batch,j, n_batch) Get batch of data(1 point)
        out = forward_pass(X_batch) (1 point)
        L = compute_loss(out,Y_batch) (1 point)
        grad = compute_gradient(L,W) (1 point)
        update_weigths(η*grad) update weight + learning rate (2 points)
    end for
    out_test = forward_pass(X_test) Test output (1 point)
    L_test = compute_loss(out_test,Y_test) Test loss (1 point)
    testing_loss.append(L_test)
    if testing_loss[i] > testing_loss[i-1] **then** Check the loss on the test set for overfitting. (2 points)
        Stop training
    end if
end for

Good luck!