Numerical Differentiation

Tamas Kis | tamas.a.kis@outlook.com | https://github.com/tamaskis

Contents

1 NUMERICAL DIFFERENTIATION

1.1 Domain Discretization

When analyzing a function numerically, the first thing we do is discretize the domain. Essentially, we consider a function $f(x)$ not as a continuous function, but rather as values corresponding to discrete locations, called **nodes**, in space. To discretize the domain, we first need to specify three quantities:

1. a: the left endpoint of the domain (i.e. the minimum value of x)
2. b: the right endpoint of the domain (i.e. the maximum value of x)
3. N: the number of subintervals (if we specify N subintervals, we will have $N+1$ points)

The length L of the domain is then

$$L = b - a$$

The discrete values of x (i.e. $x_1, ..., x_{N+1}$) are the nodes. Collectively, the set of nodes is referred to as the **mesh**. There are many different ways to create a mesh. In our case, we use a uniform mesh; this means that the nodes are equally spaced [1]. Thus, for a uniform mesh, the **grid spacing** is given by

$$\Delta x = \frac{L}{N}$$

The vector $\mathbf{x}$ storing the nodes can be defined as

$$\boxed{\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_N \\ x_{N+1} \end{bmatrix} = \begin{bmatrix} a \\ a+\Delta x \\ a+2\Delta x \\ \vdots \\ a+(i-1)\Delta x \\ \vdots \\ a+(N-1)\Delta x \\ a+N\Delta x \end{bmatrix}} \tag{1}$$

We denote the evaluation of $f(x)$ at a node x_i as

$$f_i = f(x_i)$$

We can then also define a vector $\mathbf{f}$ to store all the f_i's:

$$\mathbf{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_{N+1} \end{bmatrix}$$

We can consider the vector $\mathbf{f}$ and $\mathbf{x}$ as defining a discrete form of $f(x)$, or as vectors storing a data set:

$$f(x) \xrightarrow{\text{discretization}} \mathbf{f} \text{ vs. } \mathbf{x}$$

$$\mathbf{f} \text{ vs. } \mathbf{x} \quad \equiv \quad \{(x_i, f(x_i))\}_{i=1}^{N+1}$$

Thus, we can be given a data set $\mathbf{f}$ *vs.* $\mathbf{x}$*, and using the methods we introduce, we can find the derivative of this data set without having any knowledge of the underlying function* $f(x)$ *from which the data set is sampled.*

An example of the discretization of a univariate function onto a uniform 1D mesh is shown in Fig. 1 below.

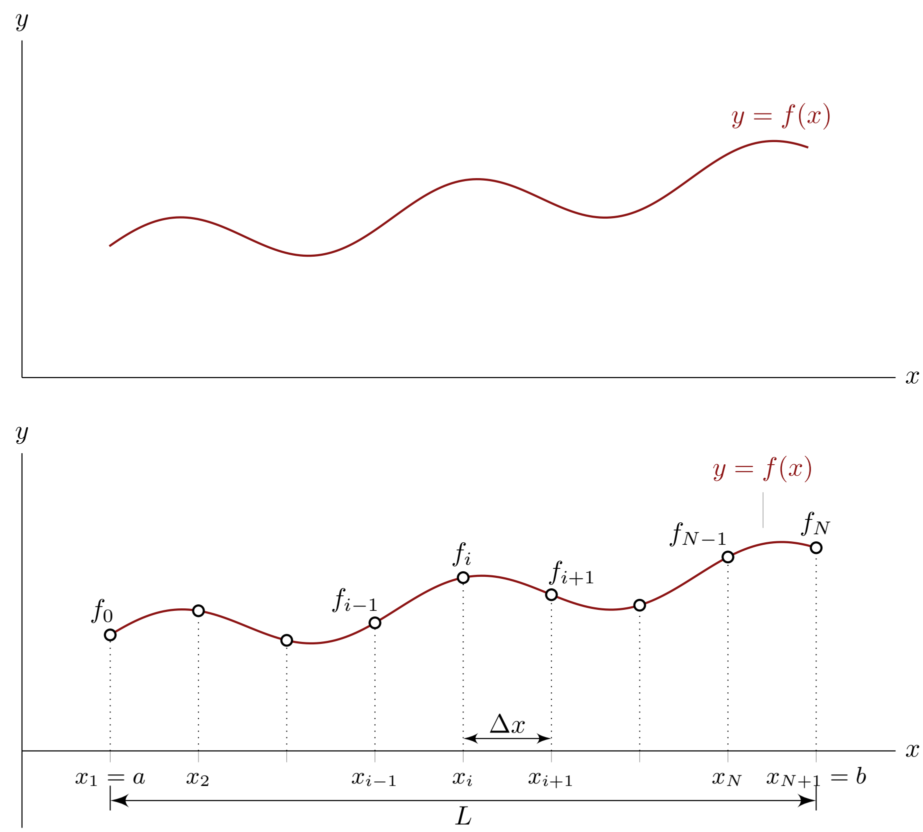

Figure 1: Domain discretization.

1.2 Finite Difference Approximations

Eq. (2) computes the **forward approximation** of the derivative [1, 2].

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_i} \approx \frac{f_{i+1} - f_i}{x_{i+1} - x_i}} \tag{2}$$

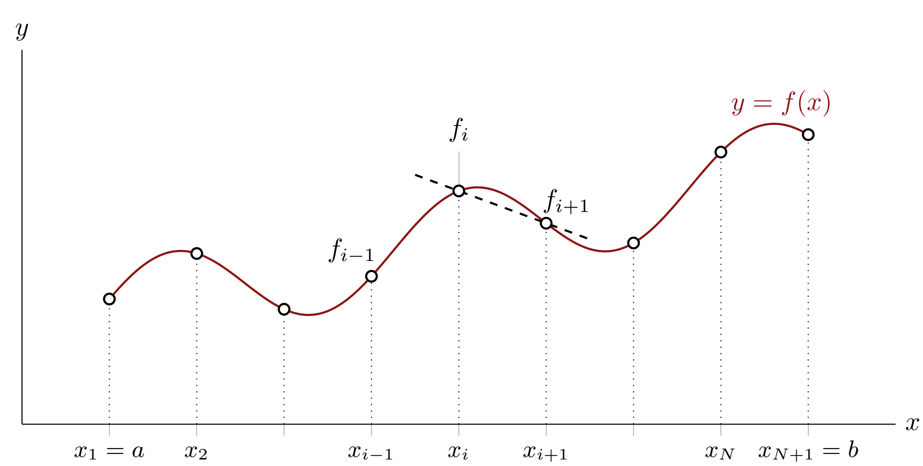

Figure 2: Forward approximation.

Eq. (3) computes the **backward approximation** of the derivative [1, 2].

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_i} \approx \frac{f_i - f_{i-1}}{x_i - x_{i-1}}} \tag{3}$$

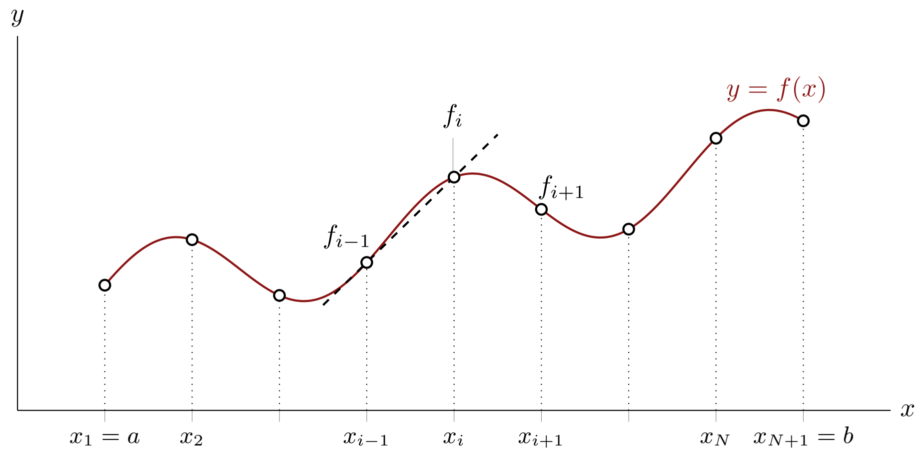

Figure 3: Backward approximation.

Eq. (4) computes the **central approximation** of the derivative. The central approximation is of higher accuracy than the forward and backward approximations [1, 2].

$$\boxed{\left.\frac{df}{dx}\right|_{x=x_i} \approx \frac{f_{i+1} - f_{i-1}}{x_{i+1} - x_{i-1}}} \tag{4}$$

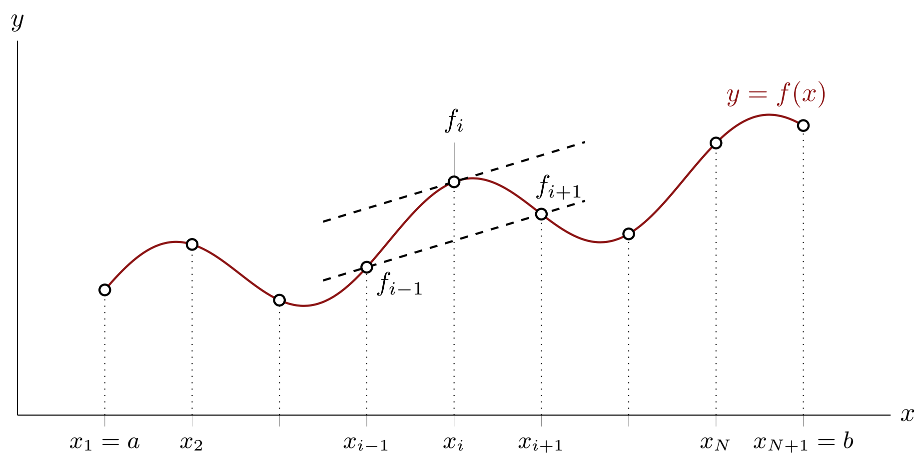

Figure 4: Central approximation.

1.3 Differentiation Over an Interval (Cumulative Differentiation)

Consider the vectors $\mathbf{f}$ and $\mathbf{x}$ storing sampled points from a function $f(x)$. We can consider these vectors as a set of points or a data set:

$$\mathbf{f} \text{ vs. } \mathbf{x} \quad \equiv \quad \{(x_i, f(x_i))\}_{i=1}^{N+1}$$

$$\therefore \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_{N+1} \end{bmatrix}, \qquad \mathbf{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_{N+1} \end{bmatrix}$$

Note that we use the notation $f_i = f(x_i)$.

Our goal is to find the derivative $f'(x)$, but without knowledge of $f(x)$, we cannot use simple algebraic differentiation rules. Instead, since we know a set of data $\mathbf{f}$ vs. $\mathbf{x}$ essentially stores sampled values of $f(x)$, we can numerically estimate the value of the derivative at all the points stored in the vector $\mathbf{x}$. The result of this numerical differentiation is a vector $\mathbf{df}$ that stores the numerical evaluation of $f'(x)$ at all the points in $\mathbf{x}$.

$$\mathbf{df} = \begin{bmatrix} df_1 \\ \vdots \\ df_{N+1} \end{bmatrix} = \begin{bmatrix} \left.\frac{df}{dx}\right|_{x=x_1} \\ \vdots \\ \left.\frac{df}{dx}\right|_{x=x_{N+1}} \end{bmatrix}$$

I refer to this numerical differentiation process as **cumulative differentiation**, analogous to cumulative integration in the case of numerical integration. The algorithm I use to perform cumulative differentiation is rather simple. At all interior nodes, I use a central approximation to approximate the derivative. At the left endpoint, I use a forward approximation, since there is no x_1 or f_1. At the right endpoint, I use a backward approximation, since there is no x_{N+2} or f_{N+2}.

Algorithm 1:
Cumulative differentiation.

Given:

- $\mathbf{f} \in \mathbb{R}^{N+1}$ - vector storing evaluations of $f(x)$ at every point in $\mathbf{x}$
- $\mathbf{x} \in \mathbb{R}^{N+1}$ - vector of x values

Procedure:

1. Determine the number of subintervals, N, given that $\mathbf{x}, \mathbf{f} \in \mathbb{R}^{N+1}$.
2. Preallocate $\mathbf{df} \in \mathbb{R}^{N+1}$ to store the cumulative derivative.
3. Calculate derivative at left endpoint using forward difference approximation.

$$df_1 = \frac{f_2 - f_1}{x_2 - x_1}$$

4. Calculate derivative at right endpoint using backward difference approximation.

$$df_{N+1} = \frac{f_{N+1} - f_N}{x_{N+1} - x_N}$$

5. Calculate derivatives at all other points using central difference approximation.

for $i = 2$ **to** N

$$df_i = \frac{f_{i+1} - f_{i-1}}{x_{i+1} - x_{i-1}}$$

end

Return:

- $\mathbf{df} \in \mathbb{R}^{N+1}$ - vector storing the evaluation of $f'(x)$ at every point in $\mathbf{x}$

1.4 Differentiation at a Point (Point Differentiation)

Previously, we introduced an algorithm (Algorithm 1) for calculating the derivative ${f_i}'$ at every node x_i. In this section, we only want to find f' at a specific point (or at a specific set of points). Note that these points do *not* have to be the nodes we used to discretize $f(x)$. We refer to this as **point differentiation**. To perform point differentiation, we first find the derivative at every node using cumulative differentiation (i.e. Algorithm 1). Then, we use linear interpolation to linearly interpolate a value for f_j' at every x_j^*.

Consider the case where there are p points x_j^* (where $j = 1, ..., p$) at which we wish to evaluate the derivative of $f(x)$. The vector $\mathbf{x}^*$ is then

$$\mathbf{x}^* = \begin{bmatrix} x_1^* \\ \vdots \\ x_p^* \end{bmatrix}$$

Let $\mathbf{df}^*$ be the vector in which we store the evaluations of $f'(x_j^*)$. Then

$$\mathbf{df}^* = \begin{bmatrix} df_1 \\ \vdots \\ df_p \end{bmatrix} = \begin{bmatrix} \left.\dfrac{df}{dx}\right|_{x=x_1^*} \\ \vdots \\ \left.\dfrac{df}{dx}\right|_{x=x_p^*} \end{bmatrix}$$

Algorithm 2:
Point differentiation.

Given:

- $\mathbf{f} \in \mathbb{R}^{N+1}$ - vector storing evaluations of $f(x)$ at every point in $\mathbf{x}$
- $\mathbf{x} \in \mathbb{R}^{N+1}$ - vector of x values
- $\mathbf{x}^* \in \mathbb{R}^p$ - point(s) at which to differentiate

Procedure:

1. Find $\mathbf{df}$ (i.e. the cumulative derivative of $\mathbf{f}$ vs. $\mathbf{x}$) using Algorithm 1.
2. Find $\mathbf{df}^*$ (i.e. the point derivatives at $\mathbf{x}^*$) by linearly interpolating/extrapolating $\mathbf{df}$ at every point in $\mathbf{x}^*$ (can be done using MATLAB's `interp1` function with the `` `linear` `` and `` `extrap` `` options specified).

Return:

- $\mathbf{df}^* \in \mathbb{R}^p$ - vector storing the evaluation of $f'(x)$ at every point in $\mathbf{x}^*$

REFERENCES

[1] *Finite difference method*. Wikipedia. https://en.wikipedia.org/wiki/Finite_difference_method (accessed: November 24, 2019).

[2] Todd Young and Martin J. Mohlenkamp. *Lecture 27: Numerical Differentiation*. Introduction to Numerical Methods and Matlab Programming for Engineers. http://www.ohiouniversityfaculty.com/youngt/IntNumMeth/lecture27.pdf (accessed: June 10, 2020).