Numerical Methods

**Assignment 6**

**Numerical Differentiation & Passing a function as input argument**

**Name: Hong Se Hyun**  **ID: 21700791**

**Numerical Differentiation**

**Part 1. From datasets**

**Problem**

Estimate the velocity and acceleration from datasets of position of an object.

clear all

t = 0:0.2:4;

x = [-5.87 -4.23 -2.55 -0.89 0.67 2.09 3.31 4.31 5.06 5.78 5.77 5.52 5.08 4.46 3.72 2.88 2.00 1.10 0.23 -0.59];

Create a function for numerical differentiation from a set of data.

Matrix gradient(Matrix x, Matrix y

Input: dataset (xi,yi), i=0 to m-1

Output: vector dydx at given x(i)

Also, create using 1D arrays such as

void gradient1D (double x[ ], double y[ ], double dydx[ ], int m)

|  |
| --- |
| **테이블이(가) 표시된 사진  자동 생성된 설명** |
| Figure 1.1 Table of Differential |

* With assumption of 3 or more data points,

3-Point forward difference: for the first point x[0]

2-Point central difference: for x[1]~x[m-2]

3-Point backward difference: for the last point x[m-1]

* For 2 points of data, use 2-point forward or backward

**Part 2. From a Function**

Pass a function as an input argument to another function.

* Define a function that defines the target equation.

double myFunc(const double x)

* Modify ‘gradient( )’ in Part 1 to get y data from the equation function (myFunc).

vector gradientFunc(double func(const double x), vector xin)

* Check your function with the test equation

at x=0:0.2:4

* Modify newtonraphson function to pass functions as input

double newtonRaphsonFunc(double func(const double x), double dfunc(const double x), float x0, float tol);

**What you need to submit**

Create the following functions and paste the code in the report.

These functions should be in myNM.h and myNM.cpp

|  |
| --- |
|  |
| Figure 2.1 Defining Functions in Source File .cpp |

Assignment#6 공지에 주어졌듯이, 임의의 수식 를 함수 내로 불러서 함수 활용성을 높였다.

즉, Source File에서 입력 함수를 즉각적으로 확인하고 바꿀 수 있게 코드를 작성한 것이다.

|  |
| --- |
|  |
| Figure 2.2 Using Array and arr2Mat to Defining Dataset |

마찬가지로 Assignment#6 공지에 주어졌듯이, 그동안 써 오던 Matrix구조체 뿐만 아니라 기본적인 1D array의 활용 연습을 위해 Figure 2.2와 같이 Array를 이용하여 Data를 표현하였다.

또한, 이번 과제에서는 txt2Mat함수를 사용하지 않고, 앞서 Array로 정의한 Data를 이용하기 위해 arr2Mat함수를 사용하여 배열을 행렬의 형태로 바꾸어 주었다.

1) Matrix gradient(Matrix x, Matrix y)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | |
| Figure 3.1 Main Code of Gradient | Figure 3.2 C Code of Gradient | | |
|  | |  |
| Figure 3.3 Result of Gradient(1) | | Figure 3.4 Result of Gradient(2) |

동일 개수의 Data가 Input x와 y로 주어졌는지, 그 Data가 2개 이상인지 확인하는 코드를 작성해주었다.

미분을 함에 있어서는, 첫 번째 data point에서는 3-point Forward Difference 기법을 이용하였고, 두 번째 data point에서 마지막 직전 data point까지는 2-point Central Difference 기법을 이용하였으며, 마지막 data point에서는 3-point Backward Difference 기법을 이용하여 미분을 진행해주었다.

2) void gradient1D (double x[ ], double y[ ], double dydx[ ], int m)

|  |  |  |
| --- | --- | --- |
|  |  | |
| Figure 4.1 Main Code of Gradient with Array | Figure 4.2 C Code of Gradient with Array | |
|  | |
| Figure 4.3 Result of Gradient with Array | |

알고리즘은 Figure 3.2의 Gradient와 동일하지만, Array 문법에 맞게 코드를 수정해주었다.

3) vector gradientFunc(double func(const double x), vector xin)

|  |  |  |
| --- | --- | --- |
|  |  | |
| Figure 5.1 Main Code of Gradient with Func | Figure 5.2 C Code of Gradient with Func | |
|  | |
| Figure 5.3 Result of Gradient with Func | |

4) double newtonRaphsonFunc(double func(const double x), double dfunc(const double x), float x0, float tol)

|  |  |  |
| --- | --- | --- |
|  |  | |
| Figure 6.1 Main Code of Newton-Raphson | Figure 6.2 C Code of Newton-Raphson | |
|  | |
| Figure 6.3 Result of Newton-Raphson | |

Newton-Raphson 결과 라는 함수의 좌표 값이 점점 0으로 수렴하는 것으로 보아 의 해가 0임을 확인할 수 있다.