＝＝＝
6 月 26 日（金）第 7 回数値解析Ⅰ 提出課題 19TM054 浅野 駿介
提出日：2020/06/26
＝＝＝
＜作成プログラム＞

(前進差分法)

```
#define _USE_MATH_DEFINES
#include<math.h>
#include<stdio.h>

double func(double x) {

	return(x * x -54);
}

double dfunc2(double x) {//刻み幅 dh を定義し x を与えられた場合の解を返す
	double dh = 0.001;
	return((func(x + dh) - func(x)) / dh);//前進差分法
}

double newton(double x, double e) {

	double x_next = x;
	double x_old;

	do {
		x_old = x_next;
		x_next = x_old - func(x_old) / dfunc2(x_old);
		printf("old_x=%lf-->new_x=%lf¥n", x_old, x_next);
	} while (fabs(x_next - x_old) > e);

	return(x_next);
}
void main() {
	double e = 0.01, x = 4.0, result = 0.0;
```

```
	result = newton(x, e);

	printf("e=%lf, result=%lf¥n", e, result);
}
```

(中央差分法)

```
#define _USE_MATH_DEFINES
#include<math.h>
#include<stdio.h>

double func(double x) {
	double d = 0.5;
	return(x*x-54);
}

double dfunc2(double x) {//刻み幅 dh を定義し x を与えられた場合の解を返す
	double dh = 0.001;
	return((func(x + dh) - func(x-dh)) / (2*dh));//前進差分法
}

double newton(double x, double e) {

	double x_next = x;
	double x_old;

	do {
		x_old = x_next;
		x_next = x_old - func(x_old) / dfunc2(x_old);
		printf("old_x=%lf-->new_x=%lf¥n", x_old, x_next);
	} while (fabs(x_next - x_old) > e);

	return(x_next);
}
void main() {
	double e = 0.01, x = 4, result = 0.0;
```

```
        result = newton(x, e);

        printf("e=%lf, result=%lf¥n", e, result);
}
```

＜出力結果＞

前進差分法を用いた場合

第五回の関数 f(x)=d-sinx の場合

old_x=4.000000-->new_x=2.076122
old_x=2.076122-->new_x=2.850103
old_x=2.850103-->new_x=2.628152
old_x=2.628152-->new_x=2.618026
old_x=2.618026-->new_x=2.617994
e=0.010000, result=2.617994

第六回の関数 f(x)=x^2-d の場合

old_x=4.000000-->new_x=8.749406
old_x=8.749406-->new_x=7.460700
old_x=7.460700-->new_x=7.349321
old_x=7.349321-->new_x=7.348469
e=0.010000, result=7.348469

中央差分法を用いた場合

第六回の関数 f(x)=x^2-d の場合

old_x=54.000000-->new_x=27.500000
old_x=27.500000-->new_x=14.731818
old_x=14.731818-->new_x=9.198677
old_x=9.198677-->new_x=7.534543
old_x=7.534543-->new_x=7.350767
old_x=7.350767-->new_x=7.348470
old_x=7.348470-->new_x=7.348469
e=0.001000, result=7.348469

第六回のサンプルプログラムを用いた場合

第六回の関数 f(x)=x^2-d の場合

old_x=54.000000-->new_x=27.500000

old_x=27.500000-->new_x=14.731818
old_x=14.731818-->new_x=9.198677
old_x=9.198677-->new_x=7.534543
old_x=7.534543-->new_x=7.350767
old_x=7.350767-->new_x=7.348470
old_x=7.348470-->new_x=7.348469
e=0.001000, result=7.348469

＜理解した内容、感想、注意点など＞
・数値積分を用いた場合導関数を求めようとしなくてよいので, 微分するのが面倒な関数に関しては前回のプログラムよりも簡単だと感じた
・中央差分法を用いて計算を行った場合と前回のプログラムを用いて変え計算を行った場合を比較すると計算過程に差は生じなかった.