第 1 回 演習課題

1026-30-8137
多田 拓生[*1]

2020 年 10 月 29 日

[*1] tada.takumi.34w@st.kyoto-u.ac.jp

電気電子計算工学及演習

1026-30-8137
多田 拓生

説明日
2020/10/8

課題 1.1

二分法およびニュートン法を用いて非線形方程式を解くプログラムをそれぞれソースコード 1、ソースコード 2 に示す。

まず二分法を用いたソースコード 1 について説明する。

bisection_method 関数は、引数として range、e、f を受け取る。これらはそれぞれ範囲、許容誤差、関数である。まず初期区間を range として与えると、bisection_method は bisection_method_inner 関数に range、e、f を渡し、さらに回数として times に 1 を、反復回数上限値として limit に 1,000,000 を与える。二分法は適切に初期区間を与えると必ず近似解が求まるため limit はオプションである。bisection_method_inner 関数は範囲を半分に区切り、解が存在すると思われる範囲を再帰的に渡して times を一つ進める。この時その範囲が許容誤差内に収まったなら、半分に区切った時の値を近似解として返す。ソースコード中に含まれる println!関数は、課題で反復回数と誤差の csv ファイルを作成するために標準出力に値を渡しているだけで求解に直接は影響しない。

次にニュートン法を用いたソースコード 2 について説明する。

まず、ニュートン法で非線形方程式を解くには関数を微分する必要がある。関数の微分には、微分係数の定義である

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

を用いて微分した関数を返す differential_f 関数を作成した。

newton_raphson_method 関数は次のようなアルゴリズムで方程式を解く。まず、関数には $f(x)$ と初期近似解を与える。すると、その関数を微分し、

$$g(x) = x - \frac{f(x)}{f'(x)} \tag{2}$$

となる $g(x)$ を計算する newton_transform 関数に $f(x)$、$f'(x)$ を渡し、また閾値と回数として 1、反復回数上限として 1,000,000 とともに newton_method 関数に渡す。

newton_method 関数では、$g(x)$ を用いて近似解の候補を求め、元の x との距離が閾値よりも小さい時、その計算した値を近似解として返す。閾値よりも大きかった場合は計算した値を再帰的に

newton_method に渡す。それを繰り返すことで非線形方程式を解く。最初に next の値をチェックしているのは、$g(x)$ の値が想定していない値になった時の処理をまとめてあるだけであり、アルゴリズムに直接は影響しない。これについては後で言及する。

ここでも後で値を plot するために println!関数を用いて標準出力に値を渡している。

ソースコード 1　bisection_method.rs

```
use std::ops::Range;
use std::rc::Rc;

fn bisection_method(
    mut range: Range<f64>,
    e: f64,
    f: Rc<dyn Fn(f64) -> f64>
)-> f64 {
    bisection_method_inner(range, e, f, 1, 1000000)
}

fn bisection_method_inner(
    mut range: Range<f64>,
    e: f64,
    f: Rc<dyn Fn(f64) -> f64>,
    times: usize,
    limit: usize,
) -> f64 {
    let x_new = (range.end + range.start) / 2.;
    if times == limit {
        return x_new;
    }
    if f(x_new) * f(range.start) >= 0. {
        range.start = x_new;
    } else {
        range.end = x_new;
    }
```

```
    println!("{}, {}", times, (x_new - 1.414213566237).abs());
    if range.end - range.start <= e {
        x_new
    } else {
        bisection_method_inner(range, e, f, times + 1, limit)
    }
}
```

ソースコード 2 newton_raphson_method.rs

```
use std::rc::Rc;
use std::result::Result;

fn newton_raphson_method(
    f: Rc<dyn Fn(f64) -> f64>,
    init: f64
) -> Result<f64, String> {
    let threshold = 0.1e-10;
    let f_dir = differential_f(f.clone());
    newton_method(newton_transform(f, f_dir),
                  init,
                  threshold,
                  1,
                  1000000)
}

fn differential_f(f: Rc<dyn Fn(f64) -> f64>)
    -> Rc<dyn Fn(f64) -> f64> {
    let dx = 0.1e-10;
    let f_dir = move |x: f64|
                    -> f64 { (f(x + dx) - f(x)) / dx };
    Rc::new(f_dir)
}

fn newton_transform(
    f: Rc<dyn Fn(f64) -> f64>,
```

```
    f_dir: Rc<dyn Fn(f64) -> f64>,
) -> Rc<dyn Fn(f64) -> f64> {
    Rc::new(move |x: f64| -> f64 { x - f(x) / f_dir(x) })
}

fn newton_method(
    f: Rc<dyn Fn(f64) -> f64>,
    guess: f64,
    threshold: f64,
    times: usize,
    limit: usize,
) -> Result<f64, String> {
    let next = f(guess);
    if next == f64::NEG_INFINITY
    || next == f64::INFINITY
    || next.is_nan() {
        return Err(
            format!(
                "x^(k+1) is not a number: last value is {}.",
                guess
        ));
    }
    if limit == times + 1 {
        return Err(format!(
            "solution doesn't converge: last value is {}.",
            next
        ));
    }
    if (next - guess).abs() <= threshold {
        Ok(next)
    } else {
        println!("{}, {},", times, (next - 1.414213566237).abs());
        newton_method(f, next, threshold, times + 1, limit)
    }
```

}

1　課題 1.1.1

$$f(x) = x^5 - 3x^4 + x^3 + 5x^2 - 6x + 2 \tag{3}$$

とする。5 次方程式 $f(x) = 0$ の解を最初に説明した二分法およびニュートン法を用いたプログラムを実行して解く。

二分法の初期期間を [-2, 0] とし、ニュートン法の初期近似解を-1 とする。そして反復回数を横軸に、それぞれの手法で得られた近似解と真値 $(\sqrt{2})$ との誤差の絶対値を縦軸にとった片対数グラフをそれぞれ図 1、図 2 に作成し、示す。

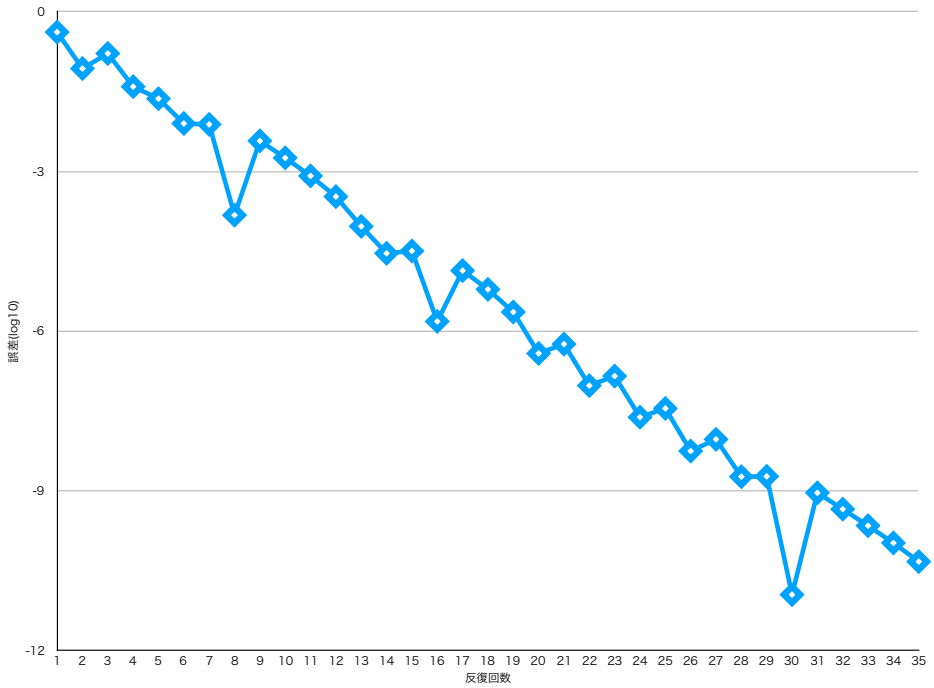

図 1　二分法の収束の速さ

2　課題 1.1.2

電気電子計算工学及演習

1026-30-8137
多田 拓生

説明日
2020/10/*

課題 1.2

3　**課題** 1.2.1

4　**課題** 1.2.2

5　**課題** 1.2.3

電気電子計算工学及演習

1026-30-8137
多田 拓生

説明日
2019/*/*

課題 1.3

6　課題 1.3.1

7　課題 1.3.2

参考文献

[1] 森正武. 『数値解析 (第 2 版)』. 共立出版, 2018.
[2] 藤野和建伊理正夫. 『数値計算の常識』. 共立出版, 2011.