수치해석 HW#2

2018008559 컴퓨터소프트웨어학부 신상윤

1. Introduction

1. 목적

여러 root finding algorithm을 실행해보고 직접 해를 구하여 본다. 또한 각각의 수렴 속도도 비교해본다.

2. Process & Result

2.1 구간 [1, 10] 안에서 Bessel function의 해들

"C:\Users\kas12\OneDrive\바탕 화면\한양대\3학년 2학기\수치해석\

```
solution at [2.35 , 2.44]
by bisection method          iterate 16 times  2.40483
by Linear interpolation      iterate 3  times  2.40483
by Secant                    iterate 3  times  2.40483
by Newton-Raphson            iterate 2  times  2.40483
by Newton with bracketing    iterate 2  times  2.40483
by Muller's method           iterate 2  times  2.40483

solution at [5.5 , 5.59]
by bisection method          iterate 16 times  5.52008
by Linear interpolation      iterate 2  times  5.52008
by Secant                    iterate 2  times  5.52008
by Newton-Raphson            iterate 2  times  5.52008
by Newton with bracketing    iterate 2  times  5.52008
by Muller's method           iterate 2  times  5.52008

solution at [8.65 , 8.74]
by bisection method          iterate 16 times  8.65373
by Linear interpolation      iterate 2  times  8.65373
by Secant                    iterate 2  times  8.65373
by Newton-Raphson            iterate 2  times  8.65373
by Newton with bracketing    iterate 2  times  8.65373
by Muller's method           iterate 1  times  8.65373
```

먼저 bracketing routine을 사용하여 해가 있는 구간을 찾는다.

```
int main(){
    Vec_O_DP a(10),b(10);
    int n;
    NR::zbrak(NR::bessj0,1,10,100,a,b,n);
    for(int i=0; i<n; i++){
        cout << "solution at [" << a[i] << " , " << b[i] << "]\n";
        check_at(a[i],b[i]);
    }
```

코드를 보면 구간 [1, 10]을 100등분 하여 해들을 a와 b에 저장해준다. 결과는 위와 같이 구간 [2.35, 2.44], [5.5, 5.59], [8.65, 8.74]에 각각 한 개씩 해가 존재한다.

이후 각 구간마다 root finding algorithm을 적용하는 함수

```
void check_at(double n,double m){
    cout << "by bisection method          " << NR::rtbis(NR::bessj0,n,m,1e-6) << "\n";
    cout << "by Linear interpolation      " << NR::rtflsp(NR::bessj0,n,m,1e-6) << "\n";
    cout << "by Secant                    " << NR::rtsec(NR::bessj0,n,m,1e-6) << "\n";
    cout << "by Newton-Raphson            " << NR::rtnewt(myfunction,n,m,1e-6) << "\n";
    cout << "by Newton with bracketing " << NR::rtsafe(myfunction,n,m,1e-6) << "\n";
    cout << "by Muller's method           " << muller(NR::bessj0,n,m,1e-6) << "\n\n";
}
```

check_at을 실행시켜 정확도 10^-6 수준의 해를 구해준다.

결과적으로 Bessel function의 해는 정확도 10^-6 수준에서 2.40483, 5.52008, 8.65373이다.

2.2 Muller method

```
double muller(DP f(const DP),double p0,double p2,double TOL){
    double tmp = 999;
    double p3,a,b,c;
    double p1 = (p0+p2)/2;
    int cnt = 0;
    while(true){
        c = f(p2);
        b = ((p0-p2)*(p0-p2)*(f(p1)-f(p2))-(p1-p2)*(p1-p2)*(f(p0)-f(p2)))/((p0-p2)*(p1-p2)*(p0-p1));
        a = ((p1-p2)*(f(p0)-f(p2))-(p0-p2)*(f(p1)-f(p2)))/((p0-p2)*(p1-p2)*(p0-p1));
        int sig = b > 0 ? 1 : -1;
        p3 = p2 - 2*c/(b+sig*sqrt(b*b-4*a*c));
        if(abs(f(p3) - tmp) < TOL){
            cout << "iterate "<< cnt << "  times  ";
            break;
        }
        tmp = f(p3);
        p0 = p1;
        p1 = p2;
        p2 = p3;
        cnt++;
    }
    return p3;
}
```

직접 Muller method를 짠 코드이다.

알고리즘은 함수 위의 점 p0, p1, p2를 시작으로 세 점을 지나는 이차 함수로 근사하여 p3를 찾는 것을 반복하는 것이다.
따라서 p0, p1, p2의 값을 p1, p2, p3로 대입하는 것을 정확도 수준을 만족할 때까지 반복하면 된다. 처음 p0, p2를 해를 찾아야 하는 구간 양 끝점으로 하였고, p1을 그 중점으로 초기값을 주었다.

2.3 Discuss of the convergence speed

각 root finding 알고리즘을 비교하기 위하여 함수별로 몇 번이나 반복하였는지 출력하도록 하였는데 구간을 100개로 나누었을 때
[2.35, 2.44]에서 bisection method가 16번, false position이 3번, secant method가 3번, 나머지는 2번 번복하였다. 정확도 1e-10수준에서는 bisection method가 29번, false position이 5번, secant method가 4번, 나머지는 2번 반복하였다. 반복 횟수가 수렴 속도와 비례하므로 bisection > false position > secant method > other 순으로 수렴 속도가 느리다고 할 수 있다.

2.4 Solve nonlinear equation $\sin x = e^{-x}$ at [0, 7]

```
find a solution of sin x = e^-x for [0, 7] by Newton with bracketing
solution at [0.56 , 0.63]
by Newton with bracketing iterate 2  times  0.588533
solution at [3.08 , 3.15]
by Newton with bracketing iterate 2  times  3.09636
solution at [6.23 , 6.3]
by Newton with bracketing iterate 2  times  6.28505
```

Newton with bracketing을 이용하여 $\sin x = e^{-x}$의 해를 구해보았다. 구간 [0, 7]에서 해는 0.588533, 3.09636, 6.28505가 나왔다.

3. Reference

[1] numerical analysis 10th edition
[2] http://numerical.recipes/book/book.html