CPE 593 - Data Structures and Algorithms

Homework 2 - Numerical Integration

Due 2/18

Xinnan Liu

10385999

---

1. Q: Compute the following integration problem using three different methods. Given $f(x) = x^2$ integrated on the interval [0,1], the area is $\int_0^1 f(x)dx$ and the analytical answer is $\frac{x^3}{3} = \frac{1-0}{3} = \frac{1}{3}$ Compute the estimate of the above integral using the brute force method for n=2 and n=4 slices. Show the answer and the error.

A:

For n=2:

According to the brute force method, $\int_0^1 f(x)dx \approx \sum_{i=1}^{2} f(a + i\Delta x)\Delta x$

$\Delta x = \frac{(b-a)}{2} = \frac{1}{2}$ **, so the result is** $\frac{1}{2} * [f(\frac{1}{2}) + f(1)] = \frac{5}{8}$

The error is $\frac{5}{8} - \frac{1}{3} = \frac{7}{24} \approx 0.2917$

For n=4:

According to the brute force method, $\int_0^1 f(x)dx \approx \sum_{i=1}^{4} f(a + i\Delta x)\Delta x$

$\Delta x = \frac{(b-a)}{4} = \frac{1}{4}$ **, so the result is** $\frac{1}{4} * [f(\frac{1}{4}) + f(\frac{1}{2}) + f(\frac{3}{4}) + f(1)] = \frac{15}{32}$

The error is $\frac{15}{32} - \frac{1}{3} = \frac{13}{96} \approx 0.1354$

---

2. Q: Compute the estimate of the above integral using the trapezoidal method for n=2 and n=4 slices. Recall that the trapezoidal method is $sum_1 = \sum_{i=0}^{n} fi$, $I_1 = sum_1$* h, where $h = \frac{b-a}{n}$ Remember to compute sum_1, then sum_2 with twice as many slices as sum_1, and compare the two. Keep doubling as long as: $|I_2 - I_1| > \epsilon$

A:

For n=2:

$$sum_1 = \sum_{i=0}^{2} f(i) = \frac{f(x_0)}{2} + f(x_1) + \frac{f(x_2)}{2} = \frac{0}{2} + \frac{1}{4} + \frac{1}{2} = \frac{3}{4}$$

$$h_1 = \frac{b-a}{n} = \frac{1-0}{2} = \frac{1}{2}, I_1 = sum_1 * h_1 = \frac{3}{4} * \frac{1}{2} = \frac{3}{8}$$

For n=4:

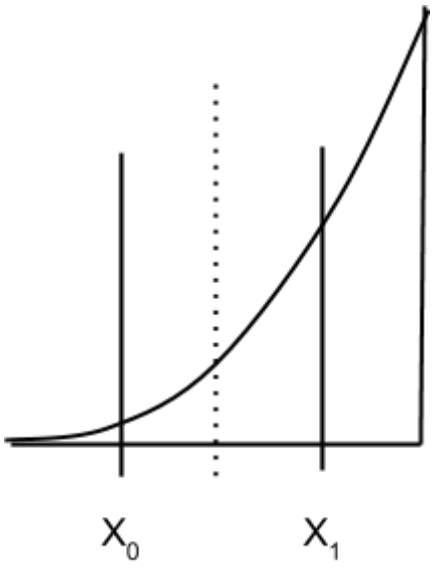
$$sum_2 = \sum_{i=0}^{4} f(i) = \frac{f(x_0)}{2} + f(x_1) + f(x_2) + f(x_3) + \frac{f(x_4)}{2} = \frac{0}{2} + \frac{1}{16} + \frac{1}{4} + \frac{9}{16} + \frac{1}{2} = \frac{11}{8}$$

$$h_2 = \frac{b-a}{n} = \frac{1-0}{4} = \frac{1}{4}, I_2 = sum_2 * h_2 = \frac{11}{8} * \frac{1}{4} = \frac{11}{32}$$

---

3. Q: Use Gaussian Quadrature (2nd order) to calculate the answer with just two points in the interval. You can look up the weights and coefficients in Wikipedia:

http://en.wikipedia.org/wiki/Gaussian_quadrature



$$x_0, x_1 = \pm \frac{b-a}{2} \sqrt{\frac{1}{3}}, w_0 = w_1 = 1$$

A:

For 2nd order Gaussian Function:

$$\int_{-1}^{1} f(x)dx \approx f(\frac{1}{\sqrt{3}}) + f(\frac{-1}{\sqrt{3}})$$

According to this function, we can denote the integration problem into below form:

***1st*, we need change the interval for the integration, let** $x = \frac{a+b}{2} + \frac{b-a}{2} t = \frac{1}{2} + \frac{1}{2} t$

***2nd*, change the original integration use the function above:**

$\int_0^1 x^2 dx = \frac{1}{2}\int_{-1}^1 (\frac{1}{2}+\frac{t}{2})^2 dt = \frac{1}{8}\int_{-1}^1 (1+t)^2 dt$

***3rd*, using the Gaussian Function,** $\int_{-1}^1 (1+t)^2 dt \approx (1+\frac{1}{\sqrt{3}})^2 + (1+\frac{-1}{\sqrt{3}})^2 = \frac{8}{3}$

***Hence*, the answer is** $\frac{1}{8} * \frac{8}{3} = \frac{1}{3}$

---

4. `Q`: Write the bisection algorithm. After testing your code, search for the root of $f(x) = x^2 - 1.501x - 0.5005$. **This factors into** $f(x) = (x-0.5)(x-1.001)$ **so you should know exactly what the roots are.Start with initial values of a=0.7 and b = 1.2, with epsilon (the error tolerance) of 0.001. Try a higher epsilon as well.The original problem was** $f(x) = x^3 - 0.502x^2 - 1.001x + 1.503$ **That equation factors into** $(x+1.5)(x-1)(x-1.001)$. **If you wish for extra credit you can figure out which of those roots you could find using the bisection algorithm.**

`A`:

The original algorithm:

```
public class bisection_original {

    public static double f(double x) {
        return Math.pow(x, 3) - 0.502 * Math.pow(x, 2) - 2.001 * x + 1.503;
    }

    public static double bisection(double a, double b, double eps) {
        double mid = 0;//find the middle value
        while (b - a > eps) {//compare the error
            mid = (a + b) / 2;
            if (f(mid) * f(b) < 0) {//bisection begin
                a = mid;
            } else {
                b = mid;
            }
        }
        return mid;
    }

    public static void main(String arg[]) {
        double a = -2;
        double b = 0;
        double eps = 0.000001;
        double res = bisection(a, b, eps);
        System.out.print(res);
    }
}
```

The modified algorithm:

```
import java.util.*;
public class Bisection {
    public static double f(double x) {
        return Math.pow(x, 3) - 0.502 * Math.pow(x, 2) - 2.001 * x + 1.503;
    }

    public static void bisection(double a, double b, double eps, ArrayList<Double>
res) {
        if (f(a) * f(b) < 0) {
            if(b-a > eps) {
                bisection(a,(a+b)/2,eps,res);
                bisection((a+b)/2,b,eps,res);
            }
            else{
                res.add(a);
            }
        }
        else if(f(a)*f(b) > 0) {
            if(b-a > eps){
                bisection(a,(a+b)/2,eps,res);
                bisection((a+b)/2,b,eps,res);
            }
        }
    }

    public static void main(String arg[]) {
        double a =0.7;
        double b =1.2;
        double eps = 0.000001;
        ArrayList<Double> res=new ArrayList<Double>();
        bisection(a, b, eps,res);
        System.out.print(res);
    }
}
```

---

5. `Q`: Write the following algorithms:

1. T1rapezoidal Integration

2. Gaussian Quadrature (2nd or 3rd order, your choice)

3. Romberg. **Recall this is Trapezoidal Integration where you compute I_1 and I_2, and then cancel the leading term of the error with: $\frac{(4I_2 - I_1)}{3}$**

`A`:

1. Trapezoidal Integration:

```
public class Integration {

    public static double f(double x) {
        return x * x;
    }

    public static double trapezoidal(double a, double b, int n) {
        double h1 = (b - a) / n;// height
        double sum = 0.5 * (f(a) + f(b));
        double I1 = sum;
        for (int i = 1; i < n; i++) {
            I1 = I1 + f(a + i * h1);
        }
        I1 = I1 * h1;// area of first trapezoidal
        return I1;
    }

    public static double trapezoidal(double a, double b, int n, double eps) {
        double I1 = trapezoidal(a, b, n);// compute 2 area of trapezoidal
        double I2 = trapezoidal(a, b, n * 2);
        int h = n * 2;
        while ((Math.abs(I2 - I1) > eps)) {
            I1 = I2;
            I2 = trapezoidal(a, b, h * 2);// call the recursion when error is
//not accurate
            h = h * 2;
        }
        return I1;
    }

    public static void main(String arg[]) {
        double ans = 0;
        int n = 2;
        for (double eps = 0.1; eps > 0.000001; eps /= 10) {
            ans = trapezoidal(0.0, 1.0, n, eps);
            System.out.println("The slice is: " + n);
            System.out.print("The estimate result is: ");
            System.out.println(ans);
            n = n * 2;
        }
    }
}
```

2. Gaussian Quadrature (2nd or 3rd order, your choice):

```
/********      For 2nd order function     ********/
public class GaussianQuadratue {
    public static double f(double x){
        return x*x;//function
    }

    public static double GaussianQuadrature(double a, double b){
        double ans=0;//result
        double t1=1/Math.sqrt(3);//point1
        double t2=-1/Math.sqrt(3);//point2
        double x1=(a+b)/2+t1*(b-a)/2;//transfer the interval
        double x2=(a+b)/2+t2*(b-a)/2;
        ans=(f(x1)+f(x2))*(b-a)/2;//compute the integration
        return ans;
    }

    public static void main(String arg[]){
        double ans=0;
        ans=GaussianQuadrature(0,1);
        System.out.print(ans);
    }
}
```

3. Romberg:

Recall this is Trapezoidal Integration where you compute I_1 and I_2, and then cancel the leading term of the error with: $\frac{(4I_2-I_1)}{3}$