University of Antwerp

Scientific Programming

Second Session
Exercise 3

Armin Halilovic - s0122210

August 22, 2016

Contents

1 Problem

We are given the function

$$f(x) = 1 + T_6(x) \quad -1 \leq x \leq 1$$

where $T_6(x)$ is the Chebyshev polynomial of degree 6. We are tasked to calculate the exact integral

$$I = \int_{-1}^{1} f(x)\,\mathrm{d}x$$

using Maple. Also, we are tasked to calculate following numerical approximations using Matlab for up to 2 significant numbers:

- The composite trapezoid rule for I
- I via a Gauss-Legendre integration rule
- Monte Carlo integration of I

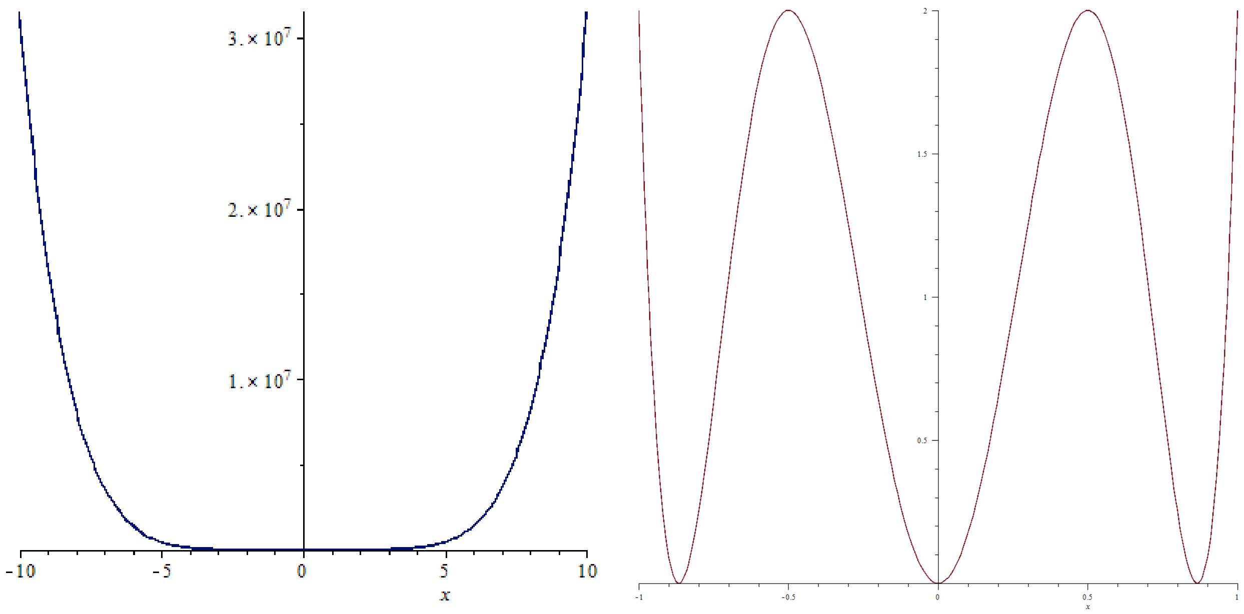

Figure 1: Two plots of the given function.

2 Using the code

All of the Maple and Matlab code can be found under maple/ and matlab/. The matlab code can aslo be found in appendix A. The file matlab/solution.m calculates all 3 given tasks and gives the results as output. An example of this output is included in appendix B.

3 Solutions

3.1 Exact calculation

Using Maple, we find the integral by

```
f := x -> 1 + ChebyshevT(6, x);
int(f(x), x = -1 .. 1);
```

and find the result to be

```
68
--
35
```

In decimal notation, this is about 1.94.

3.2 Composite trapezoid rule

We implement the following function as the composite trapeziodal rule:

$$I \approx \frac{h}{2}(f(a) + 2\sum_{i=1}^{n-1} f(x_i) + f(b))$$

To find the amount of points necessary to achieve the desired precision, we made use of the fact that the error when using this function is bounded by

$$\frac{5(b-a)^3}{12N^2} \max_{x\in[a,b]} |f''(x)|$$

If we let

$$\frac{5(b-a)^3}{12N^2} \max_{x\in[a,b]} |f''(x)| < 0.1$$

we find that we need to let n = 119 in the composite trapezoid rule to achieve the desired precision:

```
>> trapeziumN = getCompositeTrapeziumN(@f, -1, 1, 0.1)

trapeziumN =

119

>> compositeTrapeziumArea = eval(compositeTrap2(@(x) f(x), -1, 1, trapeziumN));
>> trapeziumResultAndDifference = [compositeTrapeziumArea  abs(area - compositeTrapeziumArea)]

trapeziumResultAndDifference =

       1.9446    0.0017
```

3.3 Gaussian quadrature rules

To approximate I via a Gauss-Legendre integration rule, we use the function

$$I \approx \sum_{i=0}^{n} (\frac{b-a}{2})\ w_i\ f(\frac{a+b}{2} + \frac{(b-a)x_i}{2}) \qquad x_i \in [-1, 1]$$

The x_i's are the zeroes of the Legendre polynomial of degree n+1. The weights w_i are calculated by the function

$$w_i = \frac{2}{(1 - x_i^2) * (L'_{n+1}(x_i))^2}$$

where $L_i(x)$ is the Legendre polynomial of degree i.

To achieve a precision of 2 significant numbers, we need n to equal 3. This means we have only 4 data points, as the data points are the zeroes of the Legendre polynomial with degree 4.

```
>> gaussLegendreResult = [gaussLegendre(@(x) f(x), -1, 1, 3)]

gaussLegendreResult =

 1.942857142857143
```

3.4 Monte Carlo integration

To approximate I via a Monte Carlo integration, we used the general formula

$$I \approx (measure\ of\ A) * (average\ of\ f\ over\ n\ random\ points\ in\ A)$$

In our case, the measure of A is the length of the interval we are integrating over: $(1 - -1) = 2$. Thus, we take n random points x_i in the interval $[-1, 1]$, and then approximate I by $2 * \frac{1}{n} \sum_{i=1}^{n} f(x_i)$.

Because of the randomness, it is hard to know how many points are necessary to get a precision of 2 significant numbers. We see this in the following example, where n = 200, 500, 1000, 2000, 3500, and 5000.

```
monteCarloResults = [];

for n = [200 500 1000 2000 3500 5000]
    monteCarloArea = monteCarloIntegration(@(x) f(x), -1, 1, n);
    monteCarloResults = [monteCarloResults
                         n monteCarloArea abs(area - monteCarloArea)];
end
monteCarloResults
```

```
monteCarloResults =

         200        1.8318       0.11108
         500        1.8744       0.06848
        1000        1.9475     0.0046128
        2000        1.9068      0.036096
        3500        1.9722      0.029297
        5000        1.9223      0.020535
```

In this case, we see that 500 points was enough to get a precision of 2 significant numbers, and that the result was best when n = 1000. We notice that a higher n does not necessarily cause a higher precision.

Appendices

A Code

A.1 f.m

```
function [ y ] = f(x)
    y = 1 + chebyshevT(6, x);
end
```

A.2 compositeTrap2.m

```
function [ y ] = compositeTrap2(fun, a, b, n)
    h = (b - a) / n;

    temp = 0;
    for i = 1:n-1
        temp = temp + fun(a + i*h);
    end

    y = (h / 2) * (fun(a) + 2 * temp + fun(b));
end
```

A.3 getCompositeTrapeziumN.m

```
% calculate the amount of points that are necessary in the composite trapezoid rule to achieve the desired precision
% calculate the amount by using the error bound for the composite trapezoid rule
function [ y ] = getCompositeTrapeziumN(fun, a, b, precision)
    syms x N
    secondDerivative(x) = diff(fun(x), 2);

    maxXSecondDerivative = fminbnd(matlabFunction(-1 * secondDerivative), -1, 1);
    maxSecondDerivative = max([secondDerivative(maxXSecondDerivative) secondDerivative(-1) secondDerivative(1)]);

    error = ((5 * (b - a)^3) / (12 * N^2)) * maxSecondDerivative;

    y = max(floor(solve(error < precision, N, 'Real', true)));
end
```

A.4 gaussLegendre.m

```
% degree n -> n + 1 nodes and n+1 weights
function [ y ] = gaussLegendre(fun, a, b, n)
    syms z
    %zeroes = eval(roots(coeffs(int(diff(legendreP(n+1, z))), 'All')));
    zeroes = vpasolve(legendreP(n+1, z) == 0);

    syms y
    differentiatedLegendre(y) = diff(legendreP(n+1, y));

    y = 0;
    for i = 1:length(zeroes)
        w_i = 2 / ( (1-zeroes(i)^2) * (differentiatedLegendre(zeroes(i))^2) );
        f_i = fun( ((a+b)/2) + ((b-a)*zeroes(i)/2) );

        y = y + ((b-a)/2 * w_i) * f_i;
```

```
    end
end
```

A.5 monteCarloIntegration.m

```
function [ y ] = monteCarloIntegration(fun, x_1, x_2, n)
    % rescale points using formula (oldValue - OldMin) * NewRange / OldRange + NewMin
    x = rand(n, 1) * (x_2 - x_1) + x_1;

    sum = 0;
    for i = 1:n
        sum = sum + fun(x(i));
    end

    y = (x_2 - x_1) * (sum / n);
end
```

A.6 solution.m

```
format shortG
area = 68/35

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

trapeziumN = getCompositeTrapeziumN(@f, -1, 1, 0.1)
compositeTrapeziumArea = eval(compositeTrap2(@(x) f(x), -1, 1, trapeziumN));
trapeziumResultAndDifference = [compositeTrapeziumArea  abs(area - compositeTrapeziumArea)]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

gaussLegendreResult = [gaussLegendre(@(x) f(x), -1, 1, 3)]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

monteCarloResults = [];

for n = [200 500 1000 2000 3500]
    monteCarloArea = monteCarloIntegration(@(x) f(x), -1, 1, n);
    monteCarloResults = [monteCarloResults
                         n monteCarloArea abs(area - monteCarloArea)]
end
```

B Output

B.1 solutionExampleOutput.txt

```
>> solution

area =

      1.9429


trapeziumN =

119
```

trapeziumResultAndDifference =

1.9446	0.0016942

gaussLegendreResult =

1.942857142857143

monteCarloResults =

200	1.8318	0.11108
500	1.8744	0.06848
1000	1.9475	0.0046128
2000	1.9068	0.036096
3500	1.9722	0.029297
5000	1.9223	0.020535