COMPE160 Ramishvili Amiran

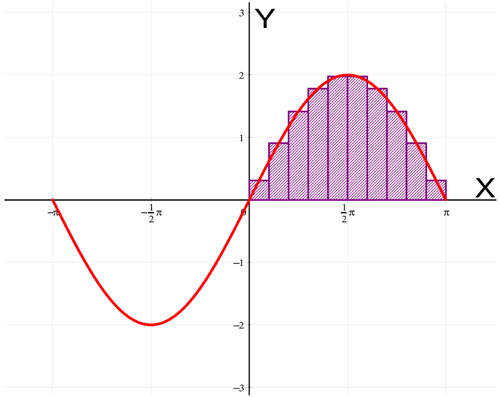
Lab 6 Part 2 - Numerical Analysis Algorithm819817616

Method:

The chosen method is Riemann sum algorithm, specifically the midpoint variant.

The algorithm provides a way to approximate the area under the curve of a function, i.e. a definite integral. This is achieved by breaking the interval down into multiple subintervals and calculating the sum of areas of rectangles whose base is the subinterval, and height is the result of the function at the midpoint of the subinterval.

So, to calculate definite integral of *f(x)* on interval a, b we break it into *N* parts *{x1, x2* … *xN}* where *x1=a & xN=b* and calculate:

***∫****abf(x)dx ~ limN*→∞∑i=1N *f*(*xi-1*/2+*xi*/2)(*xi*−*xi-1*) 

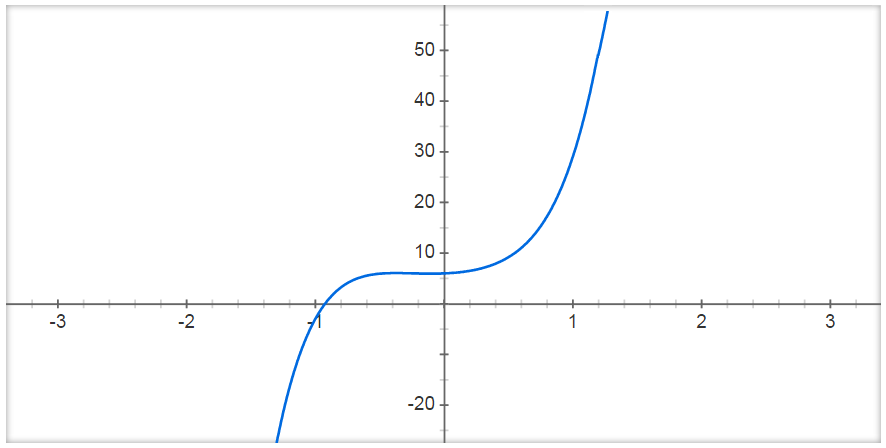
The greater the number N the better the approximation. Generally taking the limit of the sum where N→∞ yields the best result, however, as computer’s resources are limited we will set stopping criteria:

1. If maximum number of iterations (30000) is reached the program will stop and display the result.
2. If the sum converges, i.e. the (sum – previous sum) < Ɛ, where Ɛ is a small enough number the execution will stop. Ɛ=10-6

Function:

The function to be evaluated is polynomial derived from the last 6 digits of my Red ID number, which is 819817616, so:

f(x) = 8 x^5+x^4+7 x^3+6 x^2+x+6



This function has local extrema at the given points:

min{8 x^5+x^4+7 x^3+6 x^2+x+6}~~5.95301 at x~~-0.101293

max{8 x^5+x^4+7 x^3+6 x^2+x+6}~~6.06011 at x~~-0.371789

Testing:

The testing will be done for the intervals listed below, the sums will be compared to the results calculated using Wolfram Alpha website:

1.  integral_(-5)^(-4) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -309481\/20 ~~ -15474
2.  integral_(-4)^(-3) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -273773\/60 ~~ -4562.9
3.  integral_(-3)^(-2) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -55003\/60 ~~ -916.72
4.  integral_(-2)^(-1) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -1711\/20 ~~ -85.550
5.  integral_(-1)^0 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 277\/60 ~~ 4.6167
6.  integral_0^1 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 707\/60 ~~ 11.783
7.  integral_1^2 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 2759\/20 ~~ 137.95
8.  integral_2^3 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 65347\/60 ~~ 1089.1
9.  integral_3^4 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 302117\/60 ~~ 5035.3
10.  integral_4^5 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 331409\/20 ~~ 16570.
11.  integral_(-0.5)^0.5 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 6.5125
12.  integral_(-1)^1 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 82\/5 ~~ 16.400
13.  integral_(-2)^2 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 344\/5 ~~ 68.800
14.  integral_(-3)^3 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 1206\/5 ~~ 241.20
15.  integral_(-4)^4 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 3568\/5 ~~ 713.60
16.  integral_(-5)^5 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 1810
17.  integral_1.5^4.2 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 8264.39
18.  integral_(-3.4)^2.1 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -1919.61
19.  integral_12.56^13.1 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 1.52721×10^6
20.  integral_22.12^33.56 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 1.75801×10^9
21.  integral_(-19.5)^(-10.2) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -7.14853×10^7
22.  integral_(-49.5)^(-39.2) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -1.47416×10^10
23.  integral_0.1^0.2 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 0.631771
24.  integral_0.01^0.02 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 0.0601643
25.  integral_0.001^0.002 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 0.00600151
26.  integral_(-4.005)^(-4.002) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -24.9661
27.  integral_(-4.00005)^(-4.00002) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -0.248591
28.  integral_20.0001^20.0003 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 5163.94
29.  integral_20^20.000005 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 129.092
30.  integral_100^100.000005 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 400535.
31.  integral_400^400.0000000005 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 40972.6
32.  integral_0^0.1 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 0.607178
33.  integral_0^0.01 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 0.060052
34.  integral_0^0.001 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 0.0060005
35.  integral_0^0.0001 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 0.000600005
36.  integral_0^0.00001 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 0.0000600001
37.  integral_0^(1.×10^-6) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 6.×10^-6
38.  integral_99^100 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 4687886554277\/60 ~~ 78131442571
39.  integral_150^151 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 37095149682707\/60 ~~ 618252494712
40.  integral_200^200.5 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 1.28886×10^12
41.  integral_1000^1000.1 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 8.00301×10^14
42.  integral_10000^10000.1 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 80003000747028692992
43.  integral_100000^100000.0001 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 8000010400680121466880
44.  integral_1^1 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 0

The following are from intervals using the function’s extrema:

1.  integral_(-0.371789)^(-0.101293) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 1.62434
2.  integral_(-0.371789)^3 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 1241.08
3.  integral_(-0.101293)^3 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 1239.45
4.  integral_(-3)^(-0.101293) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -998.255
5.  integral_(-3)^(-0.371789) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -999.879

Examples with b<a):

1.  integral_2^1 (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = -2759\/20 ~~ -137.95
2.  integral_(-49.5)^(-60) (8 x^5+x^4+7 x^3+6 x^2+x+6) dx = 4.25097×10^10

Pseudocode:

BEGIN MAIN

FOR EACH test interval

RIEMANN\_CYCLE(f, interval, epsilon, maxIteration);

PRINT N, elapsed time, interval, sum, integral, error;

END MAIN

BEGIN RIEMANN\_CYCLE(f, interval, epsilon, maxIteration)

WHILE N++ < maxIteration

prevSum = sum;

sum = RIEMANN(f, interval, N);

if(-epsilon < sum-prevSum < epsilon) break;

RETURN sum, N;

END RIEMANN\_CYCLE

BEGIN RIEMANN(f, interval, N)

delta = intervalEnd-intervalStart/N;

FOR each subInterval in interval

sum += delta \* f(midPoint);

RETURN sum;

END RIEMANN

Code:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

#define SIZE 6 //The number of the coefficients of the polynomial

#define TESTSIZE 51 //The number of the test cases

#define MAXITER 30000 //the maximum number of iterations

#define EPSILON 0.000001 //the convergence criterion

#define AUTHOR "Amiran Ramishvili" //my name :D

#define REDID 819817616 //my Red ID number

void CalcCoeffs(int n, int \*coeffs, int size); //calculates the coefficients of the polynomial

void PrintFunc(int \*coeffs, int size); //prints the polynomial

double RedIdFunc(int \*coeffs, int size, double x); //the polynomial itself

double Riemann(int \*coeffs, int size, double a, double b, int n); //calculates Reimann sum for given interval and N

double RiemannCycle(int \*coeffs, int size, double a, double b, double epsilon, int max, int \*count); //iterates Reimann function until the maximum number of iterations is reached or the sums converge

int main()

{

int coeffArr[SIZE]; //array to store coefficients

double sum;

int iterCounter;

int i;

time\_t startTime, endTime;

//this array stores the test cases

double testArr[TESTSIZE][3] = {{-5, -4, -15474},{-4, -3, -4562.9},{-3, -2, -916.72},{-2, -1, -85.550},{-1, 0, 4.6167},{0, 1, 11.783},{1, 2, 137.95},{2, 3, 1089.1},{3, 4, 5035.3},{4, 5, 16570},{-0.5, 0.5, 6.5125},{-1, 1, 16.4},{-2, 2, 68.8},{-3, 3, 241.2},{-4, 4, 713.6},{-5, 5, 1810},{1.5, 4.2, 8264.39},{-3.4, 2.1, -1919.61},{12.56, 13.1, 1527210},{22.12, 33.56, 1758010000},{-19.5, -10.2, -71485300},{-49.5, -39.2, -14741600000},{0.1, 0.2, 0.631771},{0.01, 0.02, 0.0601643},{0.001, 0.002, 0.00600151},{-4.005, -4.002, -24.9661},{-4.00005, -4.00002, -248591},{20.0003, 20.0001, 5163.94},{20, 20.000005, 129.092},{100, 100.000005, 400535},{400, 400.0000000005, 40972.6},{0, 0.1, 0.0607178},{0, 0.01, 0.060052},{0, 0.001, 0.006005},{0, 0.0001, 0.000600005},{0, 0.00001, 0.0000600001},{0, 0.000001, 0.000006},{99, 100, 78131442571.0},{150, 151, 618252494712.0},{200, 200.5, 1288860000000.0},{1000, 1000.1, 800301000000000.0},{10000, 10000.1, 80003000747028692992.0},{100000, 100000.0001, 8000010400680121466880.0},{1, 1, 0.0},{-0.371789, -0.1012932, 1.62434},{-0.371789, 3, 1241.08},{-0.1012932, 3, 1239.45},{-3, -0.1012932, -998.255},{-3, -0.371789, -999.879},{2, 1, -137.95},{-49.5, -60, 42509700000.0}};

printf("%s - Red ID: %d\n\n", AUTHOR, REDID);

CalcCoeffs(REDID, coeffArr, SIZE);

PrintFunc(coeffArr, SIZE);

printf("The maximum number of iterations is:\t%d\n\n", MAXITER);

printf("The convergence criterion epsilon is:\t%.1e\n\n", EPSILON);

printf("%-5s%-7s%-6s%-19s%-19s%-35s%-35s%-22s\n", "No.", "Iter.", "time", "a", "b", "ReimannSum", "WolframRes", "Error");

printf("%-5s%-7s%-6s%-19s%-19s%-35s%-35s%-22s\n", "---", "-----", "----", "-----------------", "-----------------", "---------------------------------", "---------------------------------", "----------------------");

//this loop iterates over the test cases, calculates and prints the Reimann sum for each one

for(i=0; i<TESTSIZE; i++){

time(&startTime);

sum = RiemannCycle(coeffArr, SIZE, testArr[i][0], testArr[i][1], EPSILON, MAXITER, &iterCounter);

time(&endTime);

printf("%-5d%-7d%-6.0lf%17.10lf%2s%17.10lf%2s%33.10lf%2s%33.10lf%2s%22.10lf\n",

i+1,

iterCounter,

difftime(endTime,startTime), //seconds

testArr[i][0], //start of the interval

" ",

testArr[i][1], //end of the interval

" ",

sum, //the Reimann sum

" ",

testArr[i][2], //the value calculated beforehand

" ",

sum-testArr[i][2]);

}

return 0;

}

void CalcCoeffs(int n, int \*coeffs, int size){

int i, j;

//the loop singles out the last 6 digits of the red ID

for(i=0, j=(int)(pow(10,6)); i<size; i++, j/=10)

coeffs[i] = (n%j)/(j/10);

}

void PrintFunc(int \*coeffs, int size){

int i, j;

printf("f(x) = ");

for(i=0, j=size-1; i<size; i++, j--)

if(j==0) printf("%d\n", coeffs[i]);

else if(j==1) printf("%dx + ", coeffs[i], j);

else printf("%dx^%d + ", coeffs[i], j);

printf("\n");

}

double RedIdFunc(int \*coeffs, int size, double x){

double res = 0;

int i, j;

//the loop calculates the result of the polynomial with given coefficients

for(i=0, j=size-1; i<size; i++, j--)

res += coeffs[i] \* pow(x, j);

return res;

}

double Riemann(int \*coeffs, int size, double a, double b, int n){

double delta, sum = 0, i;

delta = (b-a)/n; //the size of the subinterval

//iterate over the subintervals and calculate the sum of the areas of rectangles using midpoints

for(i=a; i<b; i+=delta){

sum += delta \* RedIdFunc(coeffs, SIZE, (i + delta/2));

}

return sum;

}

double RiemannCycle(int \*coeffs, int size, double a, double b, double epsilon, int max, int \*count){

double sum = 0, prevSum = 0;

int i = 1;

//increment N until the maximum number of repetitions is reached or the sum converges

while(i < max){

prevSum = sum;

sum = Riemann(coeffs, size, a, b, i);

if((prevSum-sum) < epsilon && (prevSum-sum)>-epsilon) break; //if the sum converges break

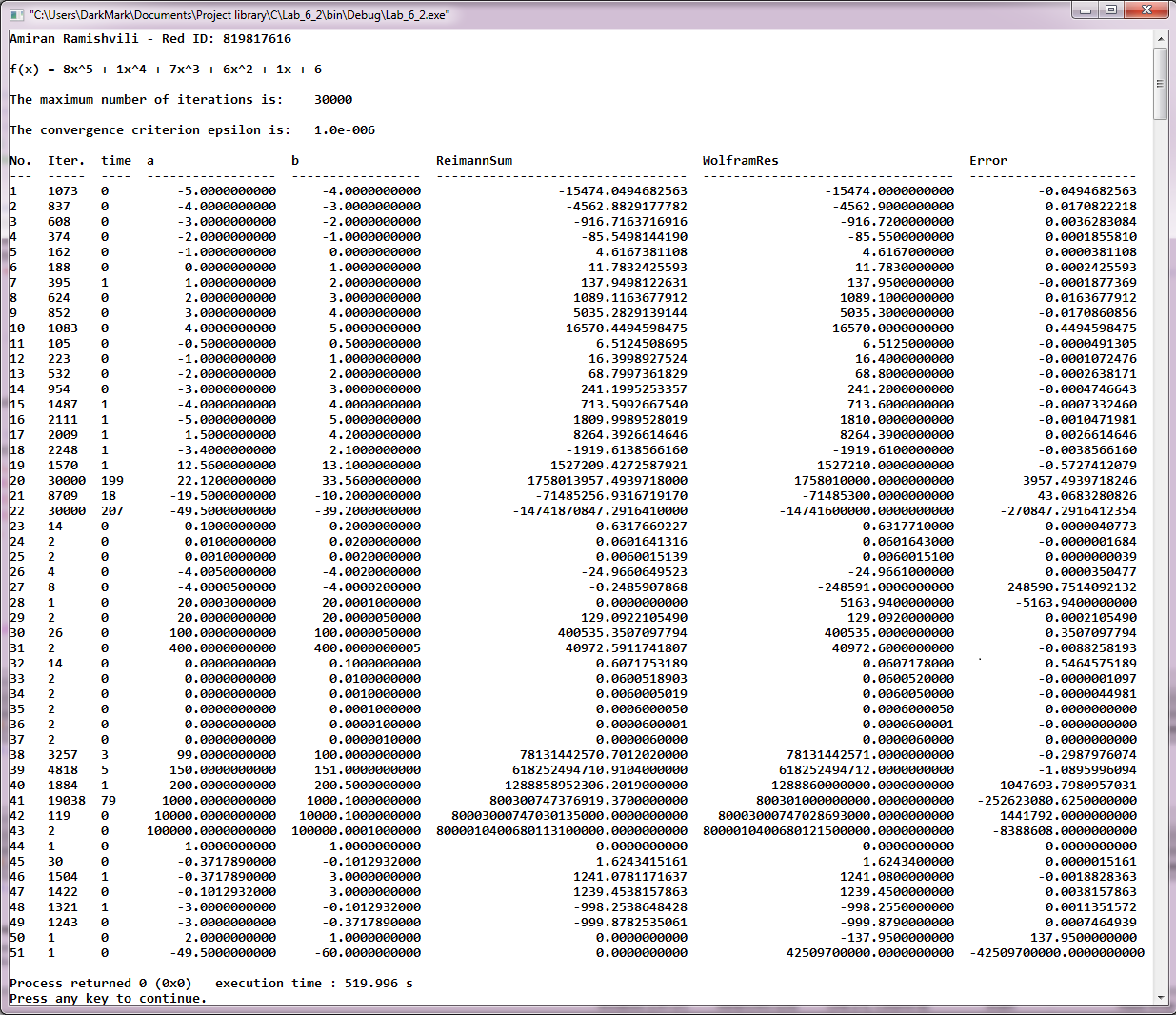
i++;

}

\*count = i;

return sum;

}

Output:

\* positive error means that Riemann sum was greater than the precalculated result, negative – it was smaller.

\* the time is shown in seconds.

Analysis:

Overall, the program performed quite well.

The iterations maxed out only twice, and the error in these cases was large, but compared to the result of the calculations not that large.

In several other cases the error was also large, however again, not so large compared to the size of the sums.

The intervals that were large, or, were located in the segments where the function sloped considerably (from the graph we can see that it slopes more the further we go from x=0 in both directions) proved to be the most error-prone.

Cases #50 and #51 were not calculated correctly, because the input was invalid, in both cases b<a and ReimannSum function relies on a being less than b.