Homework 4

Due April 15, 19:00

Cristian Brinza FAF-212

Problem 4.1

a) C++

```cpp
#include <iostream>
#include <cmath> // for pow and exp
double* linspace(double a, double b, double n)
{
double step = (b - a) / (n - 1);
double* arr = new double[n];
for (int i = 0; i < n; i++)
arr[i] = a + step * i;
return arr;
}
double f(double x, double t)
{
return std::pow(t, x - 1) * std::exp(-t);
}
double gamma(double a, double b, double x, int n)
{
const double h = (b - a) / n;
double* values = linspace(a, b, n);
double approx = f(x, values[0]) + f(x, values[n - 1]);
for (int i = 1; i < n - 2; i += 2)
{
approx += 4 * f(x, values[i]) + 2 * f(x, values[i + 1]);
}
delete[] values;
return h * approx / 3;
}
int main()
{
std::cout.precision(10);
for (double i = 1; i <= 10; i += 0.5)
std::cout << gamma(0, 30, i, 1000) << " G(" << i << ")" << s
return 0;
}
```

```
0.9990000045  G(1)
0.8849163944  G(1.5)
0.9989999865  G(2)
1.328013341  G(2.5)
1.998000027  G(3)
3.320027526  G(3.5)
5.99399997  G(4)
11.62009665  G(4.5)
23.97599991  G(5)
52.29043452  G(5.5)
119.8799973  G(6)
287.5973773  G(6.5)
719.2799149  G(7)
1869.382576  G(7.5)
5034.957344  G(8)
14020.35804  G(8.5)
40279.59696  G(9)
119172.705  G(9.5)
362514.5196  G(10)
```

b) matlab

```
f.m:
function y = f(t)
 y = t .^ (6) .* exp(-t);
end
p4_1.m:
format long
disp(quad(@f, 0, 100)) % 6! G(7)
```

->

```
>> test
   7.200000000557895e+02
```

c) matlab

```
t = 1: 0.5 : 10;
for t = t
 disp("G(" + t + ") = " + gamma(t))
end
```

->

```
G(1) = 1
G(1.5) = 0.88623
G(2) = 1
G(2.5) = 1.3293
G(3) = 2
G(3.5) = 3.3234
G(4) = 6
G(4.5) = 11.6317
G(5) = 24
G(5.5) = 52.3428
G(6) = 120
G(6.5) = 287.8853
G(7) = 720
G(7.5) = 1871.2543
G(8) = 5040
G(8.5) = 14034.4073
G(9) = 40320
G(9.5) = 119292.462
G(10) = 362880
```

Problem 4.2

a) C++

```cpp
#include <iostream>
#include <cmath>
const double PI = std::acos(0) * 2;
double* linspace(double a, double b, double n)
{
double step = (b - a) / (n - 1);
double* arr = new double[n];
for (int i = 0; i < n; i++)
arr[i] = a + step * i;
return arr;
}
double func(double x)
{
return 4 / (1 + std::pow(x, 2));
}
void simpsons(double(*f)(double), double a, double b, int n)
{
int operations = 0;
const double h = (b - a) / n;
operations += 2; // Here I consider linspace as a single operation
double* values = linspace(a, b, n);
operations++;
double approx = f(values[0]) + f(values[n - 1]);
operations += 2; // f + f
for (int i = 1; i < n - 2; i += 2)
{
approx += 4 * f(values[i]) + 2 * f(values[i + 1]);
operations += 4; // 4 * f, 2 * f, 2x f(...)
}
delete[] values;
const double pi = h * approx / 3;
operations += 3;
std::cout << "Simpsons pi = " << pi << " with " << operations << " operations" <<
std::endl;
}
void trapezoidal(double(*f)(double), double a, double b, int n)
{
int operations = 0;
const double h = (b - a) / n;
operations += 2;
double* values = linspace(a, b, n);
operations++; // Here I consider linspace as a single operation
double approx = f(values[0]) + f(values[n - 1]);
operations += 2; // f + f
for (int i = 1; i < n - 1; i++)
{
approx += 2 * f(values[i]);
operations += 2; // f(...), 2 * f
}
delete[] values;
const double pi = approx * h / 2;
operations += 3;
std::cout << "Trapezoidal pi = " << pi << " with " << operations << " operations" <<
std::endl;
}
void midpoint(double(*f)(double), double a, double b, int n)
{
int operations = 0;
const double h = (b - a) / n;
operations += 3;
double* values = linspace(a, b, n);
operations++;
double approx = 0;
for (int i = 1; i < n - 1; i++)
{
approx += f(values[i]);
operations++;
}
delete[] values;
const double pi = approx * h;
operations++;
std::cout << "Midpoint pi = " << pi << " with " << operations << " operations" <<
std::endl;
}
```

```cpp
int main()
{
std::cout.precision(5);
simpsons(func, 0, 1, 1000);
trapezoidal(func, 0, 1, 1000);
midpoint(func, 0, 1, 1000);
std::cout << "PI = " << PI;
return 0;
}
```

```
Simpsons    pi = 3.1378 with 2004 operations
Trapezoidal pi = 3.1385 with 2004 operations
Midpoint    pi = 3.1355 with 1003 operations
PI = 3.1416
```

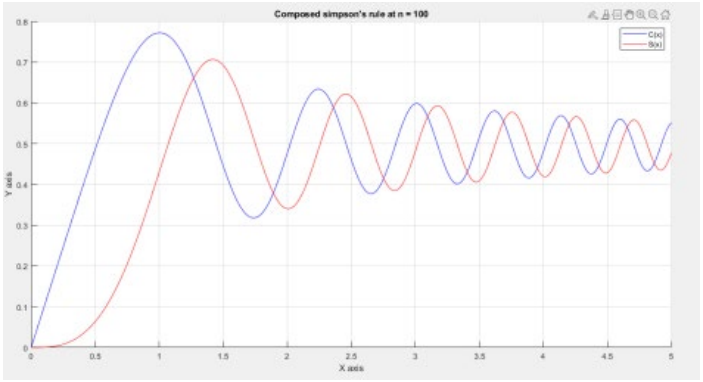
b) matlab

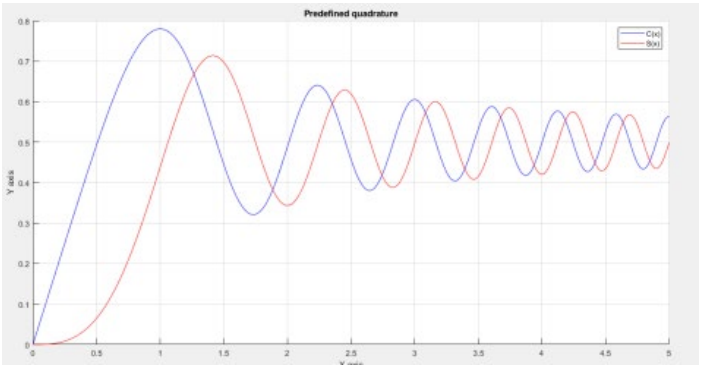
```
f.m
function y = f(x)
 y = 4 ./ (1 + x .^ 2);
end
p4_2.m
quad(@f, 0, 1)
```

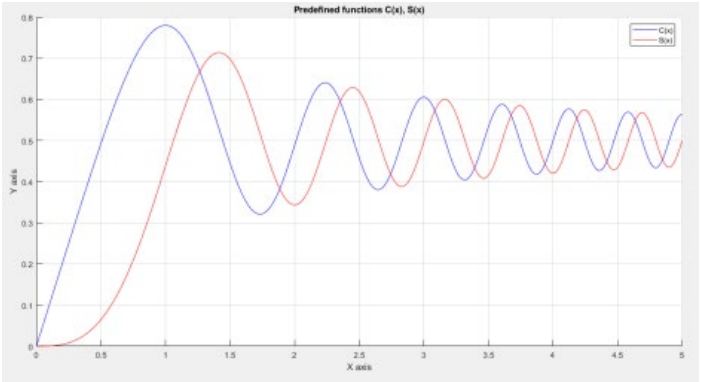
```
ans =

    3.1416
```

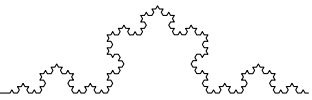
Problem 4.3

```
fresnel_c.m
function y = fresnel_c(t)
 y = cos(pi.*t.^2 / 2);
end

frensel_s.m
function y = fresnel_s(t)
y = sin(pi * t .^ 2 / 2);
end

p4_3.m
n = 100;
x = 0 : 0.01 : 5;
c = [];
s = [];
for v = x
c(end + 1) = simpsons_c(v, n);
s(end + 1) = simpsons_s(v, n);
end
figure(1)
hold on
grid on
plot(x, c, "b")
plot(x, s, "r")
title("Composed simpson's rule at n = " + n)
legend("C(x)", "S(x)")
xlabel("X axis")
ylabel("Y axis")
figure(2)
hold on
grid on
c = [];
s = [];
for v = x
c(end + 1) = quad(@fresnel_c, 0, v);
s(end + 1) = quad(@fresnel_s, 0, v);
end
plot(x, c, "b")
plot(x, s, "r")
title("Predefined quadrature")
legend("C(x)", "S(x)")
xlabel("X axis")
ylabel("Y axis")
figure(3)
hold on
grid on
plot(x, fresnelc(x), "b")
plot(x, fresnels(x), "r")
title("Predefined functions C(x), S(x)")
legend("C(x)", "S(x)")
xlabel("X axis")
ylabel("Y axis")
```









Problem 4.4

matlab - Simpsons's rule

```
function y = func_4_4(x)
    y = 1 / ((x - 1) ^ (5 / 2));
   end
   simpsons4_4.m
   function y = simpsons_4_4(n)
    a = 1;
    b = 4;
    h = (b - a) / n;
    values = linspace(a, b, n);
    approx = fresnel_c(a) + fresnel_c(b);
    for i = 2 : 2 : n - 2
    approx = approx + 4 * func_4_4(values(i)) + 2 * func_4_4(values(i + 1));
    end
    y = h * approx / 3;
   end
   p4_4.m
   clear
   clc
   n = [4 8 16 32 64 128 2^8 2^9 2^10 2^20 2^21];
   for i = n
    disp(i + ": " + simpsons_4_4(i))
   end
```

```
4: 1.3384
8: 5.0805
16: 17.0246
32: 52.3724
64: 154.3504
128: 445.4871
256: 1272.6861
512: 3617.5988
1024: 10257.3934
1048576: 336939640.597
2097152: 953010355.1826
```

python - Gaussian quadratue

```
import numpy.polynomial.legendre as leg
from numpy import linspace
def f(x: float) -> float:
 return 1 / ((x - 1) ** 2.5)
def f_(t: float) -> float:
 return f((5 + t * 3) / 2)
def gauss(n: int) -> float:
 approx = 0
 [points, weights] = leg.leggauss(n)
 for i in range(0, n):
 approx += f_(points[i]) * weights[i]
 return approx
for i in [2, 4, 8, 16, 32, 6]:
    print(f"{i}: {gauss(i)}")
```

```
2: 3.24091846802418
4: 18.37628600685488
8: 122.33443258290843
16: 891.2095136484222
32: 6802.5261373610965
64: 53157.312940088326
```