Numerical Analysis

Homework 4

Student: Frunze Vladislav

Problem 4.1

MATLAB program:

```
f = inline('t.^(x-1).*exp(-t)','t','x');
a = 0; b = 1e3; h_min = 1e-3;
x = [1 2 5 10];
for i = 1:4
fprintf('For x = %d,\n', x(i));

disp('a) Error of Simpson rule for a given h:');
disp(' h       Simpson'); k = 0; h = b-a;
while h >= h_min, k = k+1;
err_s = abs((Simpson(f,a,b,h,x(i))-gamma(x(i)))/gamma(x(i)));
fprintf('%8.1e %10.3e\n', h, err_s); h = h/10;
end; disp(' ');

disp('b) Error and function evaluations for given tolerance tol:');
disp(' quad     quadl');
disp(' tol  err  evals  err  evals');
for k = 1:7
tol = 10^(-k);
[Q, fcnt] = quad(f,a,b,tol,[],x(i)); err_q = abs((Q-gamma(x(i)))/...
    gamma(x(i)));
[Ql, fcntl] = quadl(f,a,b,tol,[],x(i)); err_ql = abs((Ql-gamma(x(i)))/...
    gamma(x(i)));
fprintf('%8.1e %10.3e %5d %10.3e %5d\n', tol, err_q, fcnt, err_ql, fcntl);

end; disp(' ');
end

function [I] = Midpoint(f, a, b, h, x)
t = [a+h/2:h:b-h/2]'; I = h*sum(feval(f,t,x));
end
function [I] = Trapezoid(f, a, b, h, x)
t = [a:h:b]'; I = h*(sum(feval(f,t,x))-(feval(f,a,x)+feval(f,b,x))/2);
end
function [I] = Simpson(f, a, b, h, x)
I = (2*Midpoint(f,a,b,h,x)+Trapezoid(f,a,b,h,x))/3;
end
```

Output:

```
For x = 1,
a) Error of Simpson rule for a given h:
 h       Simpson
 1.0e+03  1.657e+02
 1.0e+02  1.567e+01
 1.0e+01  7.117e-01
 1.0e+00  3.372e-04
 1.0e-01  3.471e-08
 1.0e-02  3.472e-12
 1.0e-03  3.331e-16

b) Error and function evaluations for given tolerance tol:
 quad     quadl
 tol   err   evals   err   evals
 1.0e-01  4.073e-01     29  3.238e-05    78
 1.0e-02  1.831e-03     37  8.622e-07   108
 1.0e-03  7.511e-05     41  8.622e-07   108
 1.0e-04  3.971e-05     45  8.622e-07   108
 1.0e-05  9.797e-06     57  6.664e-12   138
 1.0e-06  4.941e-07     73  6.664e-12   138
 1.0e-07  3.580e-08    105  9.579e-13   198
```

```
For x = 2,
a) Error of Simpson rule for a given h:
 h       Simpson
 1.0e+03  1.000e+00
 1.0e+02  1.000e+00
 1.0e+01  7.739e-01
 1.0e+00  9.918e-04
 1.0e-01  1.041e-07
 1.0e-02  1.042e-11
 1.0e-03  8.882e-16

b) Error and function evaluations for given tolerance tol:
 quad     quadl
 tol   err   evals   err   evals
 1.0e-01  1.000e+00     13  1.000e+00     18
 1.0e-02  1.000e+00     13  1.000e+00     18
 1.0e-03  1.000e+00     13  1.000e+00     18
 1.0e-04  1.000e+00     13  1.000e+00     18
 1.0e-05  1.000e+00     13  1.000e+00     18
 1.0e-06  1.000e+00     13  1.000e+00     18
 1.0e-07  1.000e+00     13  1.000e+00     18
```

For x = 5,

a) Error of Simpson rule for a given h:

h	Simpson
1.0e+03	1.000e+00
1.0e+02	1.000e+00
1.0e+01	2.372e-01
1.0e+00	4.299e-05
1.0e-01	5.158e-11
1.0e-02	4.441e-16
1.0e-03	1.480e-16

b) Error and function evaluations for given tolerance tol:

	quad		quadl	
tol	err	evals	err	evals
1.0e-01	1.000e+00	13	1.000e+00	18
1.0e-02	1.000e+00	13	1.000e+00	18
1.0e-03	1.000e+00	13	1.000e+00	18
1.0e-04	1.000e+00	13	1.000e+00	18
1.0e-05	1.000e+00	13	1.000e+00	18
1.0e-06	1.000e+00	13	1.000e+00	18
1.0e-07	1.000e+00	13	1.000e+00	18

For x = 10,

a) Error of Simpson rule for a given h:

h	Simpson
1.0e+03	1.000e+00
1.0e+02	1.000e+00
1.0e+01	1.145e-01
1.0e+00	5.986e-11
1.0e-01	1.604e-16
1.0e-02	1.123e-15
1.0e-03	6.416e-16

b) Error and function evaluations for given tolerance tol:

	quad		quadl	
tol	err	evals	err	evals
1.0e-01	1.000e+00	13	1.000e+00	18
1.0e-02	1.000e+00	13	1.000e+00	18
1.0e-03	1.000e+00	13	1.000e+00	18
1.0e-04	1.000e+00	13	1.000e+00	18
1.0e-05	1.000e+00	13	1.000e+00	18
1.0e-06	1.000e+00	13	1.000e+00	18
1.0e-07	1.000e+00	13	1.000e+00	18

Problem 4.2

a)

MATLAB program:

```
f = inline('4./(1+x.^2)', 'x'); a = 0; b = 1; hmin = 1e-5;
k = 0; h = b-a; I_true = pi;
disp('The approximations of pi with the rules');
disp(' h        Midpoint  Trapezoid   Simpson');
while h >= hmin, k = k+1;
fprintf('%.2e %f %f %f\n', h, Midpoint(f,a,b,h), Trapezoid(f,a,b,h),...
    Simpson(f, a, b, h));
h = h/10;
end; disp(' ');

function [I] = Midpoint(f, a, b, h)
x = [a+h/2:h:b-h/2]'; I = h*sum(feval(f,x));
end
function [I] = Trapezoid(f, a, b, h)
x = [a:h:b]'; I = h*(sum(feval(f,x))-(feval(f,a)+feval(f,b))/2);
end
function [I] = Simpson(f, a, b, h)
I = (2*Midpoint(f,a,b,h)+Trapezoid(f,a,b,h))/3;
end
```

Output:

```
The approximations of pi with the rules
 h        Midpoint  Trapezoid   Simpson
1.00e+00 3.200000 3.000000 3.133333
1.00e-01 3.142426 3.139926 3.141593
1.00e-02 3.141601 3.141576 3.141593
1.00e-03 3.141593 3.141592 3.141593
1.00e-04 3.141593 3.141593 3.141593
1.00e-05 3.141593 3.141593 3.141593
```

The order of the error of the rules are the following:

Midpoint rule: 2.

Trapezoidal rule: 2.

Simpson’s rule: 4.

Changing a little the above program, we can compare the accuracy of the rules with the following chart:

```
Errors of the rules
 h          Midpoint       Trapezoid       Simpson
1.00e+00 5.840735e-02 1.415927e-01 8.259320e-03
1.00e-01 8.333314e-04 1.666665e-03 6.200076e-10
1.00e-02 8.333333e-06 1.666667e-05 0.000000e+00
1.00e-03 8.333333e-08 1.666667e-07 0.000000e+00
1.00e-04 8.333307e-10 1.666669e-09 2.220446e-15
1.00e-05 8.333778e-12 1.666667e-11 0.000000e+00
```

Yes, indeed, for smaller values of h there is no further improvement because the computer cannot store more digits.

b)

MATLAB program:

```
f = inline('4./(1+x.^2)', 'x'); a = 0; b = 1; hmin = 1e-5;
k = 0; h = b-a; I_true = pi;
disp('b):');
disp('                    quad                        quadl');
disp(' tol        val          err     evals     val        err         evals');
for k = 1:9
tol = 10^(-k); [Q, fcnt] = quad(f,a,b,tol); err_q = abs(Q-I_true);
[Ql, fcntl] = quadl(f,a,b,tol); err_ql = abs(Ql-I_true);
fprintf('%8.1e %f %10.3e %5d %f %10.3e %5d\n', tol, quad(f,a,b,tol), ...
    err_q, fcnt, quadl(f,a,b,tol), err_ql, fcntl);
end; disp(' ');

function [I] = Midpoint(f, a, b, h)
x = [a+h/2:h:b-h/2]'; I = h*sum(feval(f,x));
end
function [I] = Trapezoid(f, a, b, h)
x = [a:h:b]'; I = h*(sum(feval(f,x))-(feval(f,a)+feval(f,b))/2);
end
function [I] = Simpson(f, a, b, h)
I = (2*Midpoint(f,a,b,h)+Trapezoid(f,a,b,h))/3;
end
```

Output:

```
b):
                    quad                        quadl
 tol          val          err     evals     val        err          evals
 1.0e-01 3.141595  2.597e-06    13 3.141593  5.344e-08    18
 1.0e-02 3.141595  2.597e-06    13 3.141593  5.344e-08    18
 1.0e-03 3.141595  2.597e-06    13 3.141593  5.344e-08    18
 1.0e-04 3.141595  2.597e-06    13 3.141593  5.344e-08    18
 1.0e-05 3.141593  5.662e-08    17 3.141593  5.344e-08    18
 1.0e-06 3.141593  2.933e-08    21 3.141593  5.344e-08    18
 1.0e-07 3.141593  8.076e-10    33 3.141593  5.344e-08    18
 1.0e-08 3.141593  1.436e-10    61 3.141593  1.332e-14    48
 1.0e-09 3.141593  1.631e-10    77 3.141593  1.332e-14    48
```

c)

For a: Elapsed time is 0.068548 seconds.

For b: Elapsed time is 0.063331 seconds.

Thus, for part a), elapsed time is longer.

Problem 4.3

MATLAB program:

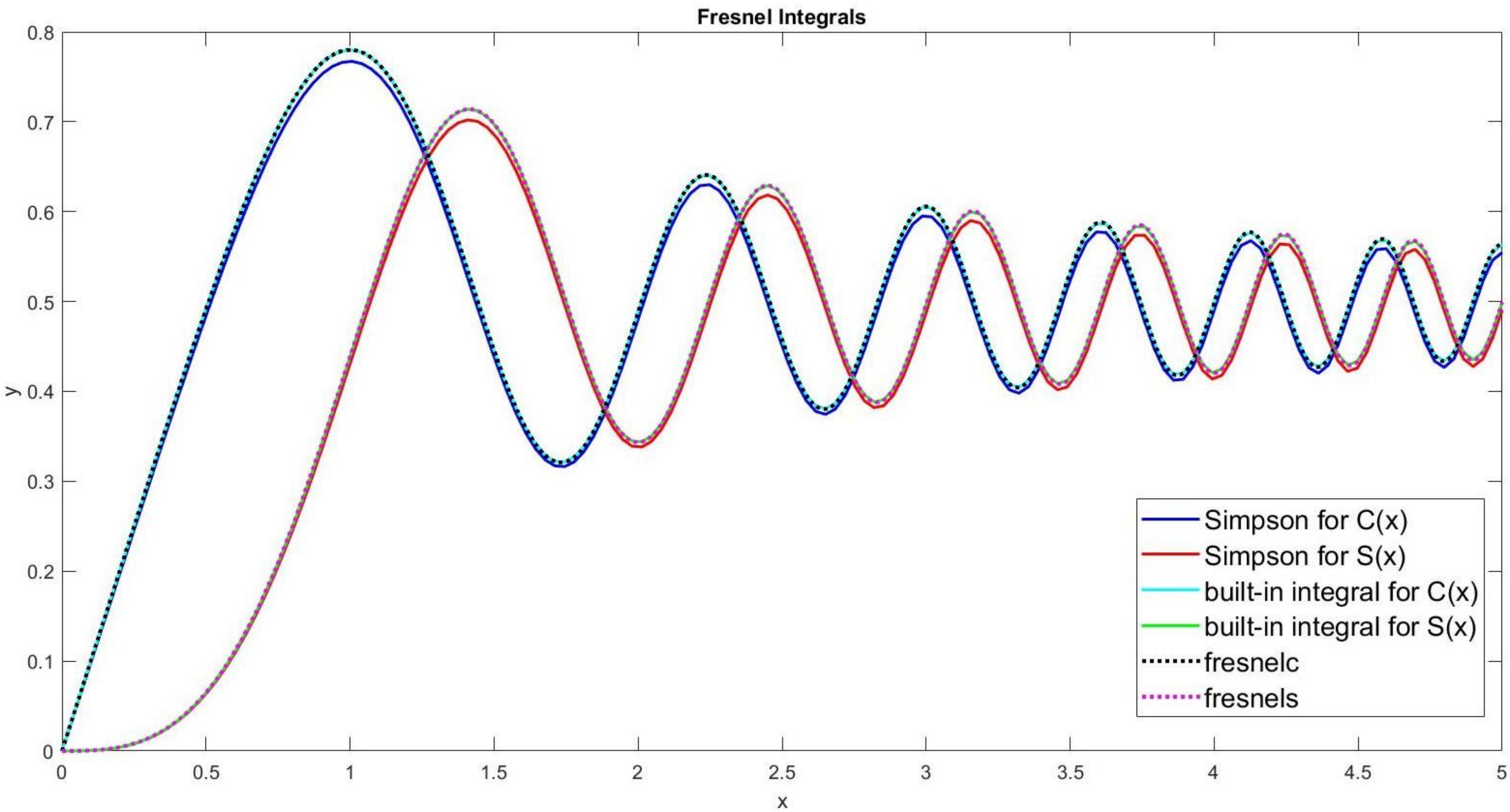
```
integrand1 = @(t) cos(pi*t.^2/2);
integrand2 = @(t) sin(pi*t.^2/2);
n = 150; x = linspace(0,5,n);
C = zeros(n,1); S = zeros(n,1);
C2 = zeros(n,1); S2 = zeros(n,1);
h = 10^(-3);
for i = 2:n
C(i) = C(i-1)+Simpson(integrand1, x(i-1), x(i), h);
S(i) = S(i-1)+Simpson(integrand2, x(i-1), x(i), h);
C2(i) = C2(i-1)+integral(integrand1, x(i-1), x(i));
S2(i) = S2(i-1)+integral(integrand2, x(i-1), x(i));

end
plot(x,C,'b-',x,S,'r-', 'linewidth', 1.5);
hold on
plot(x,C2,'c-',x,S2,'g-', 'linewidth', 1.5);
syms t;
fplot(fresnelc(t), [0, 5], 'k:', 'linewidth', 2);
hold on
fplot(fresnels(t), [0, 5], 'm:', 'linewidth', 2);
xlabel('x'); ylabel('y');

legend('Simpson for C(x)', 'Simpson for S(x)', ...
    'built-in integral for C(x)', 'built-in integral for S(x)', ...
    'fresnelc', 'fresnels');
title('Fresnel Integrals');

function [I] = Midpoint(f, a, b, h)
x = [a+h/2:h:b-h/2]'; I = h*sum(feval(f,x));
end
function [I] = Trapezoid(f, a, b, h)
x = [a:h:b]'; I = h*(sum(feval(f,x))-(feval(f,a)+feval(f,b))/2);
end
function [I] = Simpson(f, a, b, h)
I = (2*Midpoint(f,a,b,h)+Trapezoid(f,a,b,h))/3;
end
```

Output:



Fresnel Integrals
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
y
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
x
Simpson for C(x)
Simpson for S(x)
built-in integral for C(x)
built-in integral for S(x)
fresnelc
fresnels

Problem 4.4

MATLAB program for Simpson's rule:

```
format compact
integrand = @(x) 1/(x-1).^(5/2);
n = [4, 8, 16, 32, 64];
h = 10^(-2);
disp('For Simpson rule:');
for i = 1:5
    x = linspace(1, 4, n(i));
    disp(' ')
    fprintf('For n = %d\n', n(i));
    integral = 0;
    for j = 2:n(i)
        integral = abs(Simpson(integrand, x(j-1), x(j), h)) + integral;
    end
    fprintf('Integral value = %f', integral);
end

function [I] = Midpoint(f, a, b, h)
x = [a+h/2:h:b-h/2]'; I = h*sum(feval(f,x));
end
function [I] = Trapezoid(f, a, b, h)
x = [a:h:b]'; I = h*(sum(feval(f,x))-(feval(f,a)+feval(f,b))/2);
end

function [I] = Simpson(f, a, b, h)
I = (2*Midpoint(f,a,b,h)+Trapezoid(f,a,b,h))/3;
end
```

Output:

```
For n = 4
Integral value = Inf
For n = 8
Integral value = Inf
For n = 16
Integral value = Inf
For n = 32
Integral value = Inf
For n = 64
Integral value = Inf>>
```

The output suggests that the integral is divergent on the given interval.

MATLAB program for Gaussian quadrature:

```
format compact
integrand = @(x) 1/(x-1).^(5/2);
n = [4, 8, 16, 32, 64];
disp('For Gaussian quadrature:');
for i = 1:5
    disp(' ')
    fprintf('For n = %d\n', n(i));
    [nodes, weights] = gaussquadrule(n(i), 'hermite');
    syms x
    value = int(1/(x-1).^(5/2),[1,4]);
    fprintf('Integral value = %f', vpa(value));
end
```

The output is going to give the same values, therefore, we conclude that the integral on the given interval is divergent, that is, infinite.