КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Лабораторна робота №4

Виконав:
студент 4-го курсу
спеціальність Математика
Шатохін Михайло

2016

1 Постановка задачі

Необхідно обчислити інтеграл $I = \int_{\alpha}^{\beta} \rho(x)U(x)\,dx$ за допомогою квадратурної формули:

$$I = \int_a^b \rho(x)U(x)\,dx \approx \sum_{k=1}^{n} A_k f(x_k)$$

В даній задачі:

$\rho(x) = 1 + x,$

$[a, b] = [0, 1],$

$n = 7,$

$[\alpha, \beta] = [7, 21],$

$U(x) = e^{\frac{-x}{21+x}}.$

2 Практична реалізація

У функції main відбувається ініціалізація параметрів задачі, виклик основної частини алгоритму та виведення результату.

main.m

```
function [] = main()
    [func, weight, interpolationInterval , approximationInterval , degree] =
        Init ('params.mat');

    [points , coefficients ] = GetInterpolationCoefficients ( interpolationInterval , degree,
        weight)

     interpolatedIntegral = Interpolate (func, weight, points , coefficients ,
         interpolationInterval , approximationInterval )
    approximatedIntegral = SimpsonsMethod(approximationInterval, @(t)(func(t) *
        weight(t)), 1e-10)
    printf(' difference is %f\n', interpolatedIntegral - approximatedIntegral)
end;

function [func, weight, interpolationInterval , approximationInterval , degree] =
    Init (fileName)
    load(fileName);
    weight = @(t)(weight(t, g));
    func = @(t)(func(t, g, k));
    degree = min(g + 6, max(d, k));
     approximationInterval = [min(k, d), max(k, d)];
end;
```

У функції GetInterpolationCoefficients відбувається пошук квадратурної формули найвищого степеня точності.

GetInterpolationCoefficients.m

```
function [points, coefficients] = GetInterpolationCoefficients(interval, degree,
    weightFunction, integrationPrecision = 1e-5)
    moments = zeros(1, 2*degree);
    for i = 0:2*degree - 1;
        moments(i + 1) = SimpsonsMethod(interval, @(t)(weightFunction(t) * (t^i)),
            integrationPrecision);
    end;

    matrix = [ones(degree, 1) * (degree:-1:1) + (0:degree-1)' * ones(1, degree),
        (1:degree)' + degree];
    matrix = moments(matrix);
    matrix(:, end) *= -1;

    ortPolynomial = matrix(:, 1:end - 1) \ matrix(:, end);
    ortPolynomial = [1, ortPolynomial'];
    points = sort(roots(ortPolynomial));

    polynomial = @(t, coef)(prod(t - coef));
    coefficients = zeros(degree, 1);
    for i = 1:degree
        denominator = polynomial(points(i), points(1:end ~= i));
        coefficients(i) = SimpsonsMethod(interval, @(t)(polynomial(t, points(1:end ~=
            i)) / denominator * weightFunction(t)), integrationPrecision);
    end;
end;
```

У функції Interpolate відбувається підрахунок інтеграла за допомогою квадратурної формули.

Interpolate.m

```
function result = Interpolate(func, weight, points, coefficients, interpolationInterval,
    approximationInterval)
    coef = [(interpolationInterval(2) - interpolationInterval(1)) \
        (approximationInterval(2) - approximationInterval(1)), approximationInterval(1)
        - interpolationInterval(1)];
    coefficients = coef(1) * coefficients ./ arrayfun(weight, points);
    points = coef(1) * points + coef(2);
    coefficients = coefficients .* arrayfun(weight, points);

    result = sum(coefficients .* arrayfun(func, points));
end;
```

Функція SimpsonsMethod здійснює обрахунок інтеграла методом Сімпсона.

SimpsonsMethod.m

```
function result = SimpsonsMethod(interval, func, precision, maxIterations = 20)
    divisionNumber = 2;
    divisionLength = (interval(2) - interval(1)) / 2;
    oddSumElements = divisionLength / 3 * (func(interval(1) + divisionLength));
    evenSumElements = 0;
    endpointsValue = divisionLength / 3 * (func(interval(1)) + func(interval(2)));
    result = endpointsValue + 4 * oddSumElements; %+ 2 * evenSumElements, but it's 0
```

```
    for i = 1:maxIterations
        approximation = result;
        evenSumElements = (evenSumElements + oddSumElements) / 2;
        endpointsValue /= 2;
        oddSumElements = divisionLength / 6 * sum(arrayfun(func, interval(1) +
            divisionLength / 2 + (0:divisionNumber - 1) * divisionLength));
        divisionLength /= 2;
        divisionNumber *= 2;
        result = endpointsValue + 4 * oddSumElements + 2 * evenSumElements;
        if abs(result - approximation) < precision
            return;
        end;
    end;
end;
```

Файл params.mat містить параметри задачі. Вони наведені у зрозумілому людині вигляді.

params.mat

```
# name: func
# type: function handle
 @<anonymous>
 @(t, g, k)(exp(-(t ^ g)/(t + k)))
# name: weight
# type: function handle
 @<anonymous>
 @(t, g)(1 + (t ^ g))
# name: interpolationInterval
# type: matrix
# rows: 1
# columns: 2
 0 1
# name: k
# type: scalar
 21
# name: g
# type: scalar
 1
# name: d
# type: scalar
 13
```

В результаті роботи програми отримуємо такий результат:

output.txt

```
>>main
points =

  -1.974775
   0.038479
   0.200605
   0.446455
   0.697703
```

```
    0.889715
    0.984829

 coefficients =

    1.8360e-06
    1.0336e-01
    2.6011e-01
    3.7770e-01
    3.9135e-01
    2.7758e-01
    8.9895e-02

  interpolatedIntegral = 91.902
 approximatedIntegral = 91.902
 difference is 0.000070
>>
```
