

Лабораторна робота учбової практики

Виконав:
студент 4-го курсу
спеціальність математика
Колінько Микола Миколайович

1 Постановка задачі

Необхідно побудувати інтерполяційний сплайн $S(x, u)$ другого степеня дефекту 1, з крайовими умовами типу 4.

2 Теоретичні відомості

Для побудови системи рівнянь для коефіцієнтів використаємо другі похідні $2a_i = M_i = S''(X_i, u)$. Використовуючи рівність поділених різниць сплайну та інтерпольованої функції в точках X отримуємо обмеження на M_i :

$$\begin{aligned} h_{i-1}M_{i-1} + 3(h_{i-1} + h_i)M_i + h_iM_i &= 8u(X_{i-1}; X_i; X_{i+1})(h_{i-1} + h_i); \\ h_{i-1}M_{i-1} + 3(h_{i-1} + h_i)M_i + h_iM_i &= 8(u(X_i; X_{i+1}) - u(X_{i-1}; X_i)). \end{aligned}$$

Де $h_i = X_{i+1} - X_i$. Тоді (враховуючи 4 Тип обмежень) отримуємо систему рівнянь:

$$\begin{aligned} -M_1 + M_2 &= 0; \\ h_{i-1}M_{i-1} + 3(h_{i-1} + h_i)M_i + h_iM_i &= 8(u(X_i; X_{i+1}) - u(X_{i-1}; X_i)), i = \overline{2, n-1}; \\ M_{n-1} - M_n &= 0. \end{aligned}$$

Для якої можна записати матрицю

$$\left(\begin{array}{cccccc|c} -1 & 1 & 0 & \dots & 0 & 0 & 8(u(X_2; X_3) - u(X_1; X_2)) \\ h_1 & 3(h_1 + h_2) & h_2 & \dots & 0 & 0 & \\ \dots & \dots & \dots & \dots & \dots & \dots & \\ \dots & 0 & h_{n-2} & 3(h_{n-2} + h_{n-1}) & h_{n-1} & \dots & 8(u(X_{n-1}; X_n) - u(X_{n-2}; X_{n-1})) \\ 0 & \dots & 0 & -1 & 1 & 0 & 0 \end{array} \right)$$

Систему рівнянь розв'язуємо методом лівої прогонки.

Для обрахування саме коефіцієнтів сплайну використовуються наступні формули:

$$\begin{aligned} a_i &= \frac{M_i}{2}; \\ c_i &= u_i; \\ b_1 &= u(X_1, X_2) - \frac{1}{8}h_1(3M_1 + M_2); \\ b_i &= u(X_{i-1}; X_i) + \frac{1}{8}h_1(M_{i-1} + 3M_i), i = \overline{2, n}. \end{aligned}$$

3 Код

Функція, що здійснює перевірку правильності побудови сплайну: побудову графіків та розрахунок сіткової норми.

main.m

```
1 function [] = main(f, x, graphx)
2     if ! exist('f')
3         f = @(t)(sin(t^2));
4     end;
5     if ! exist('x')
6         x = ((0 : 0.05 : 1) .^(3/5)) * 5;
7     end;
8     if ! exist('graphx')
9         graphx = 0 : 0.001 : 5;
10    end;
11
12    [ interpolSpline , splinef ] = CreateSpline(x, f);
13    spline0 = @(t)( splinef (0, t));
14    spline1 = @(t)( splinef (1, t));
15    spline2 = @(t)( splinef (2, t));
16
17    figure('units','normalized','outerposition',[0 0 1 1], 'paperorientation',
18           'landscape');
19    if strcmp(class(f), 'function_handle')
20        plot(graphx, arrayfun(f, graphx), 'g', graphx, arrayfun(spline0, graphx), 'r',
21              x, arrayfun(f, x), 'kx');
22        legend('interpolated _function', 'interpolation _spline', 'pivot_x', 'location',
23               'southoutside');
24    else
25        plot(graphx, arrayfun(spline0, graphx), 'k', x, arrayfun(f, x), 'kx');
26        legend('interpolation _spline', 'pivot_x', 'location', 'southoutside');
27    end;
28    title ( sprintf('Maximal_deviation:_%e', max(abs(arrayfun(f, x) - arrayfun(spline0,
29                                          x)))));
30    grid minor;
31    print -dpdf ./ result .pdf;
32    figure('units','normalized','outerposition',[0 0 1 1], 'paperorientation',
33           'landscape');
34    plot(graphx, arrayfun(spline1, graphx), 'g', graphx, arrayfun(spline2, graphx), 'k');
35    legend('spline _first _derivative', 'spline _second _derivative', 'location',
36           'southoutside');
37    grid minor;
38    print -dpdf -append ./ result .pdf;
39 end;
```

Функція, що здійснює побудову сплайна.

CreateSpline.m

```
1 function [ interpolSpline , splinef ] = CreateSpline(x, func)
2     if strcmp(class(func), 'function_handle')
3         y = arrayfun(func, x);
4     elseif length(func) == length(x)
5         y = func;
6     else
7         error('Unknown format of input argument func. ');
8     end;
9     if isrow(x)
10         x = x';
11     end;
12     if isrow(y)
13         y = y';
14     end;
15     [matrix, splineX] = CreateMatrix(x, y);
16     solution = Solve(matrix);
17     interpolSpline = FormSpline(x, y, solution);
18     splinef = @(derivative, t)( EvaluateSpline(x, splineX, interpolSpline, derivative,
19         t));
20 end;
21 function result = EvaluateSpline(x, splineX, interpolSpline, derivative, t)
22     [row, segmentValue] = SelectRow(x, splineX, interpolSpline, t);
23     Coeff = EvaluateCoeff(length(row), derivative);
24     powers = segmentValue.^ (length(row) - derivative - 1 : -1 : 0);
25     result = sum(row(1 : length(powers)) .* powers .* Coeff(1 : length(powers)));
26 end;
27 function Coeff = EvaluateCoeff(rowLength, derivative)
28     if derivative == 0
29         Coeff = ones(1, rowLength);
30         return;
31     end;
32     Coeff = prod((ones(derivative, 1) * (rowLength - 1 : -1 : 0)) - ((0 : derivative -
33         1)' * ones(1, rowLength)), 1);
34 end;
35
36 function [row, segmentValue] = SelectRow(x, splineX, interpolSpline, t)
37     i = max([0; find((t - splineX) >= 0)]) + 1;
38     row = interpolSpline(i, :);
39     segmentValue = t - x(i);
40 end;
```

Побудова матриці відповідно до методу побудови системи лінійних рівнянь, що зазначений в умові.

CreateMatrix.m

```

1 function [matrix, splineX] = CreateMatrix(x, y)
2     n = length(x);
3     segments = x(2 : end) - x(1 : end - 1);
4     splineX = x(2 : end) - segments / 2;
5     deltas = (y(2 : end) - y(1 : end - 1)) ./ segments(1 : end);
6     matrix = [diag(segments(1 : end - 1)), zeros(n - 2, 2)] + [zeros(n - 2, 2),
7         diag(segments(2 : end))] +...
8     3 * [zeros(n - 2, 1), diag(segments(1 : end - 1)) + diag(segments(2 : end)), zeros(n
9         - 2, 1)];
10    matrix = [-1, 1, zeros(1, n - 2); matrix; zeros(1, n - 2), 1, -1];
11    rSide = [0; 8 * (deltas(2 : end) - deltas(1 : end - 1)); 0];
12    matrix = [matrix, rSide];
13
14    matrix(2, :) -= matrix(1, :) * matrix(2, 1);
15    matrix(end - 1, :) -= matrix(end, :) * matrix(end - 1, end - 1);
16 end;

```

Розв'язання системи лінійних рівнянь за визначеним в умові методом

Solve.m

```

1 function solution = Solve(matrix)
2     [rows, cols] = size(matrix);
3     core = matrix(:, 1 : rows);
4     trig = diag(ones(1, rows)) + diag(ones(1, rows - 1), 1) + diag(ones(1, rows - 1),
5         -1);
6     if max(abs(core - core .* trig)) < 1e-10
7         rSide = matrix(:, rows + 1 : end);
8         gam = delt = zeros(rows + 1, cols - rows);
9         for i = rows : -1 : 2
10             gam(i, :) = -matrix(i, i - 1) ./ (matrix(i, i) + gam(i + 1, :) * matrix(i, i
11                 + 1));
12             delt(i, :) = (rSide(i, :) - matrix(i, i + 1) .* delt(i + 1, :)) ./
13                 (matrix(i, i) + gam(i + 1, :) * matrix(i, i + 1));
14         end;
15         solution = zeros(rows, cols - rows);
16         solution(1, :) = (rSide(1, :) - matrix(1, 2) * delt(2, :)) ./ (matrix(1, 1) +
17             gam(2, :) * matrix(1, 2));
18         for i = 2 : rows
19             solution(i, :) = gam(i, :) .* solution(i - 1, :) + delt(i, :);
20         end;
21     else
22         error('wrong matrix');
23     end;
24 end;

```

Формування коефіцієнтів сплайну.

FormSpline.m

```
1 function interpolSpline = FormSpline(x, y, solution)
2     n = length(x);
3     segments = x(2 : end) - x(1 : end - 1);
4     deltas = (y(2 : end) - y(1 : end - 1)) ./ segments(1 : end);
5
6     interpolSpline = [ solution / 2, zeros(n, 1), y];
7     interpolSpline (:, 2) = [ deltas (1) - 0.125 * segments(1) * (3 * solution (1) +
8         solution (2)) ;...
9         deltas (1 : end) + 0.125 * segments(1 : end) .* ( solution (1 : end - 1) + 3 *
            solution (2 : end))];
10 end;
```

Нижче наведений результат роботи програми:

Maximal deviation: 0.000000e+000



