КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Робота учбова практика

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

2016

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:

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

де $h_i = X_{i+1} - X_i$.

Тоді (враховуючи 4 Тип обмежень) отримуємо систему рівнянь:

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

Для якої можна записати матрицю

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

Систему рівнянь розв'язуємо методом лівої прогонки.

Для обрахування саме коефіцієнтів сплайну використовуються наступні формули:

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

3 Код

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

main.m

```
function [] = main(f, x, graphx)
    if ! exist('f')
        f = @(t)(sin(t^2));
    end;
    if ! exist('x')
        x = ((0 : 0.05 : 1) .^(3/5) ) * 5;
    end;
    if ! exist('graphx')
        graphx = 0 : 0.001 : 5;
    end;

    [ interpolSpline , splinef ] = CreateSpline(x, f);
    spline0 = @(t)( splinef (0, t));
    spline1 = @(t)( splinef (1, t));
    spline2 = @(t)( splinef (2, t));

    figure('units','normalized','outerposition',[0 0 1 1], 'paperorientation',
        'landscape');
    if strcmp(class(f), 'function_handle')
        plot(graphx, arrayfun(f, graphx), 'g', graphx, arrayfun( spline0 , graphx), 'r',
            x, arrayfun(f, x), 'kx');
        legend(' interpolated ␣function', ' interpolation ␣spline', 'pivot␣x', 'location',
            'southoutside');
    else
        plot(graphx, arrayfun( spline0 , graphx), 'k', x, arrayfun(f, x), 'kx');
        legend(' interpolation ␣spline', 'pivot␣x', 'location', 'southoutside');
    end;
    title ( sprintf('Maximal␣deviation:␣%e', max(abs(arrayfun(f, x) - arrayfun( spline0 ,
        x)))));
    grid minor;
    print -dpdf ./ result .pdf;
    figure('units','normalized','outerposition',[0 0 1 1], 'paperorientation',
        'landscape');
    plot(graphx, arrayfun( spline1 , graphx), 'g', graphx, arrayfun( spline2 , graphx), 'k');
    legend(' spline␣ first ␣ derivative ', ' spline␣second␣ derivative ', 'location',
        'southoutside');
    grid minor;
    print -dpdf -append ./ result .pdf;
end;
```

Функція, що здійснює побудову сплайна.

CreateSpline.m

```
function [ interpolSpline , splinef ] = CreateSpline(x, func)
    if strcmp(class(func), 'function_handle')
        y = arrayfun(func, x);
    elseif length(func) == length(x)
        y = func;
    else
        error('Unknown format of input argument func.');
    end;
    if isrow(x)
        x = x';
    end;
    if isrow(y)
        y = y';
    end;
    [matrix, splineX] = CreateMatrix(x, y);
    solution = Solve(matrix);
    interpolSpline = FormSpline(x, y, solution);
    splinef = @(derivative, t)(EvaluateSpline(x, splineX, interpolSpline , derivative ,
        t));
end;

function result = EvaluateSpline(x, splineX, interpolSpline , derivative , t)
    [row, segmentValue] = SelectRow(x, splineX, interpolSpline , t);
    Coeff = EvaluateCoeff(length(row), derivative);
    powers = segmentValue .^ (length(row) - derivative - 1 : -1 : 0);
    result = sum(row(1 : length(powers)) .* powers .* Coeff(1 : length(powers)));
end;

function Coeff = EvaluateCoeff(rowLength, derivative)
    if derivative == 0
        Coeff = ones(1, rowLength);
        return;
    end;
    Coeff = prod((ones( derivative , 1) * (rowLength - 1 : -1 : 0)) - ((0 : derivative -
        1)' * ones(1, rowLength)), 1);
end;

function [row, segmentValue] = SelectRow(x, splineX, interpolSpline , t)
    i = max([0; find((t - splineX) >= 0)]) + 1;
    row = interpolSpline (i, :);
    segmentValue = t - x(i);
end;
```

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

CreateMatrix.m

```
function [matrix, splineX] = CreateMatrix(x, y)
    n = length(x);
    segments = x(2 : end) - x(1 : end - 1);
    splineX = x(2 : end) - segments / 2;
    deltas = (y(2 : end) - y(1 : end - 1)) ./ segments(1 : end);
    matrix = [diag(segments(1 : end - 1)), zeros(n - 2, 2)] + [zeros(n - 2, 2),
        diag(segments(2 : end))] +...
     3 * [zeros(n - 2, 1), diag(segments(1: end - 1)) + diag(segments(2 : end)), zeros(n
        - 2, 1)];
    matrix = [-1, 1, zeros(1, n - 2); matrix; zeros(1, n - 2), 1, -1];
    rSide = [0; 8 * (deltas(2 : end) - deltas(1 : end - 1)); 0];
    matrix = [matrix, rSide];

    matrix(2, :) -= matrix(1, :) * matrix(2, 1);
    matrix(end - 1, :) -= matrix(end, :) * matrix(end - 1, end - 1);
end;
```

Розв'язання системи лінійних рівнянь за визначеним в умові методом

Solve.m

```
function solution = Solve(matrix)
    [rows, cols] = size(matrix);
    core = matrix(:, 1 : rows);
    trig = diag(ones(1, rows)) + diag(ones(1, rows - 1), 1) + diag(ones(1, rows - 1),
        -1);
    if max(abs(core - core .* trig)) < 1e-10
        rSide = matrix(:, rows + 1 : end);
        gam = delt = zeros(rows + 1, cols - rows);
        for i = rows : -1 : 2
            gam(i, :) = -matrix(i, i - 1) ./ (matrix(i, i) + gam(i + 1, :) * matrix(i, i
                + 1));
            delt(i, :) = (rSide(i, :) - matrix(i, i + 1) .* delt(i + 1, :)) ./
                (matrix(i, i) + gam(i + 1, :) * matrix(i, i + 1));
        end;
        solution = zeros(rows, cols - rows);
        solution(1, :) = (rSide(1, :) - matrix(1, 2) * delt(2, :)) ./ (matrix(1, 1) +
            gam(2, :) * matrix(1, 2));
        for i = 2 : rows
            solution(i, :) = gam(i, :) .* solution(i - 1, :) + delt(i, :);
        end;
    else
        error('wrong matrix');
    end;
end;
```

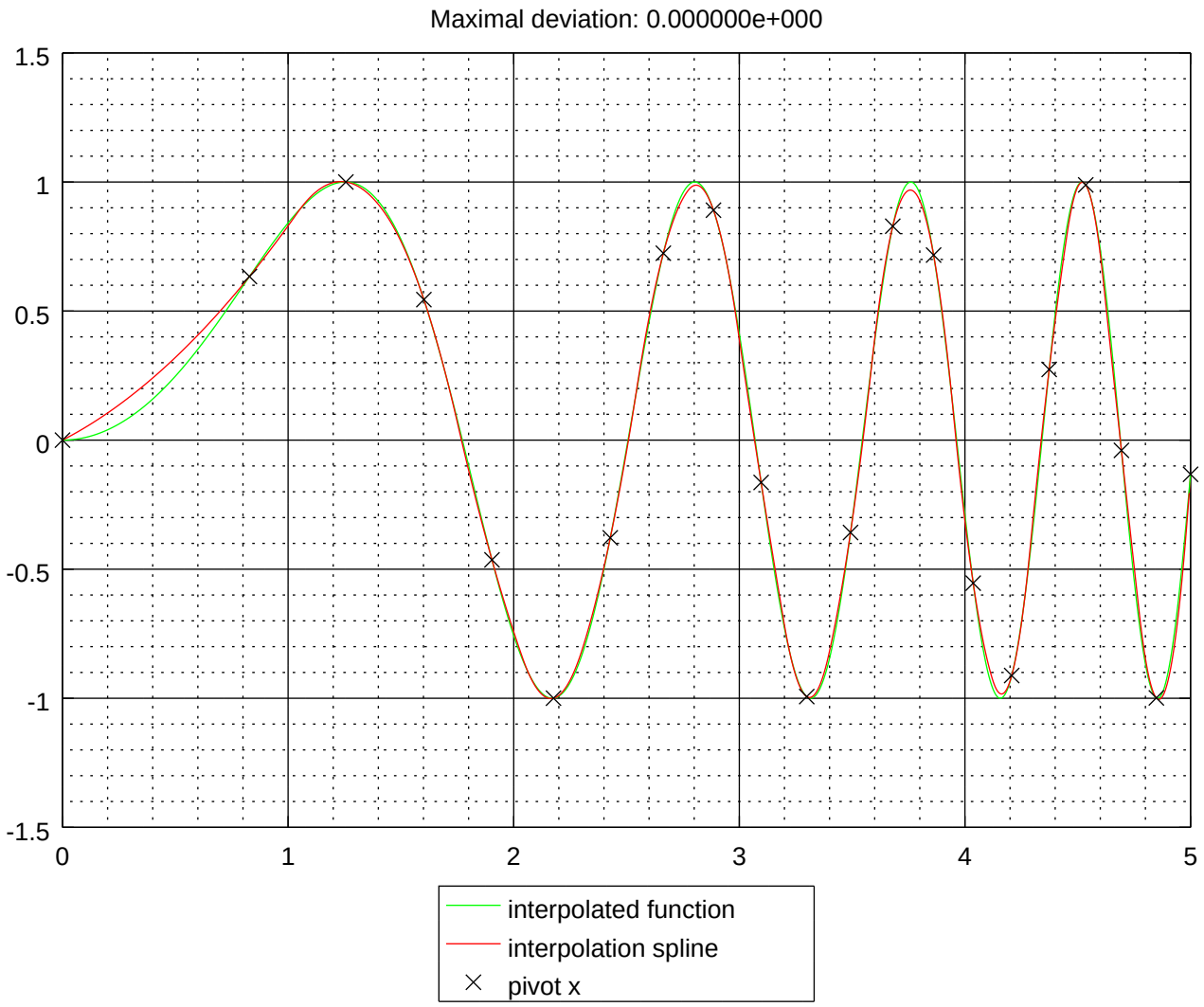
Формування коефіцієнтів сплайну.

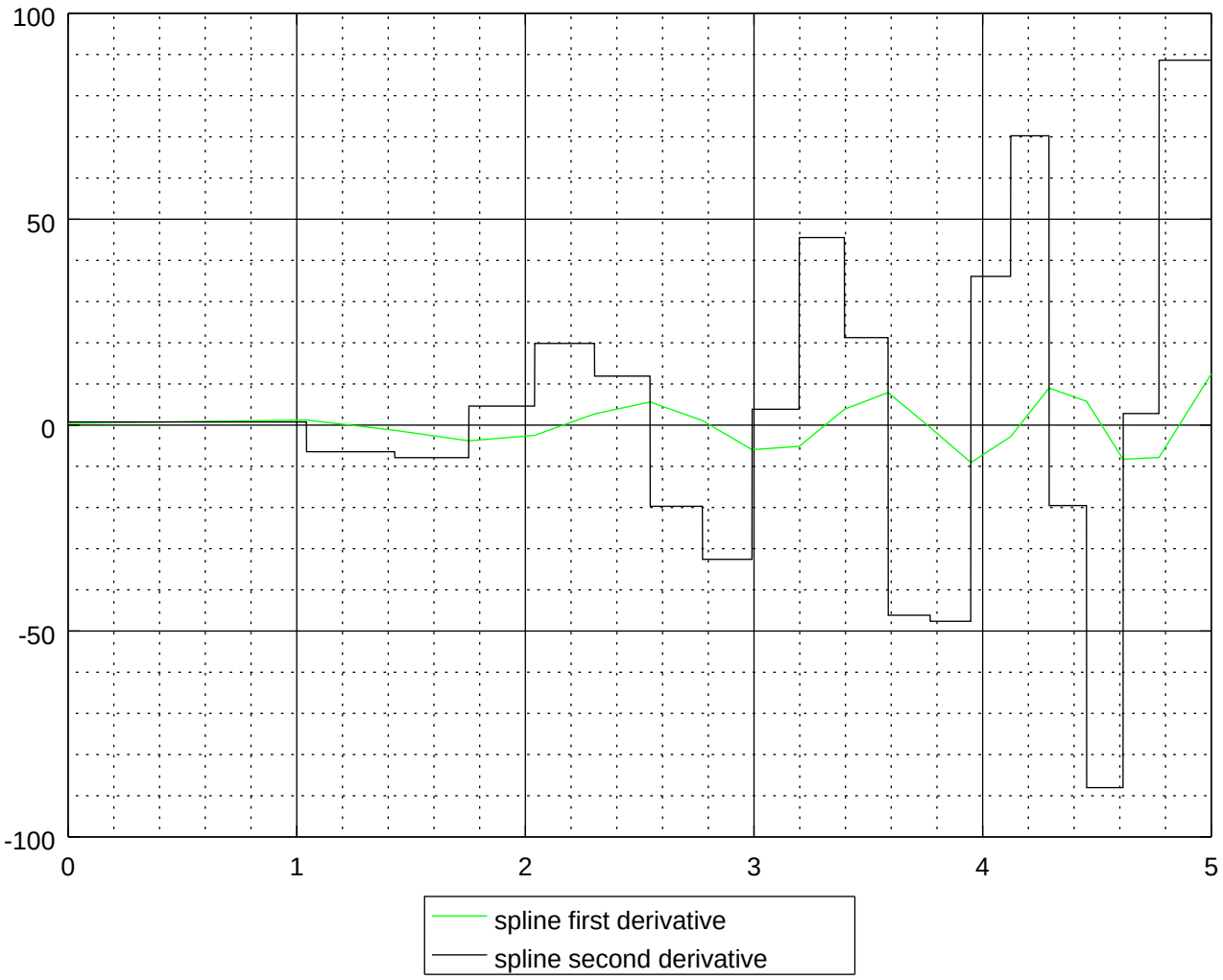
FormSpline.m

```
function interpolSpline = FormSpline(x, y, solution)
    n = length(x);
    segments = x(2 : end) - x(1 : end - 1);
    deltas = (y(2 : end) - y(1 : end - 1)) ./ segments(1 : end);

    interpolSpline = [solution / 2, zeros(n, 1), y];
    interpolSpline(:, 2) = [deltas(1) - 0.125 * segments(1) * (3 * solution(1) + solution(2)) ;...
     deltas(1 : end) + 0.125 * segments(1 : end) .* (solution(1 : end - 1) + 3 * solution(2 : end))];
end;
```

Нижче наведений результат роботи програми:


Maximal deviation: 0.000000e+000
1.5
1
0.5
0
-0.5
-1
-1.5
0
1
2
3
4
5
interpolated function
interpolation spline
pivot x


100
50
0
-50
-100
0
1
2
3
4
5
spline first derivative
spline second derivative