COMPSCI 4X03

Assignment 2

Mingzhe Wang

McMaster University

October 25, 2021

Problems 1

without pivoting

without pivoting

$$\left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 1 & 5 & -1 & 7 \\ 6 & 3 & 7 & 15 \end{array}\right]$$

③ = ③ − 2 × ①

② = ② − 3.3333×10^{-1} × ①

$$\left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 0 & 3.6667 & -2 & 3.6667 \\ 0 & -5 & 1 & -5 \end{array}\right]$$

③ = ③ − (−1.3636) × ②

$$\left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 0 & 3.6667 & -2 & 3.6667 \\ 0 & 0 & -1.7272 & -8.7880\times10^{-5} \end{array}\right]$$

$$x_3 = (-8.7880 \times 10^{-5}) / -1.7272 = 5.0880 \times 10^{-5}$$

$$\begin{aligned} x_2 &= (3.6667 - ((-2) \times (5.0880 \times 10^{-5}))) / 3.6667 \\ &= (3.6667 - (-1.0176 \times 10^{-4})) / 3.6667 \\ &= 3.6668 / 3.6667 \\ &= 1 \end{aligned}$$

$$\begin{aligned} x_1 &= (10 - (4 \times 1 + 3 \times (5.0880 \times 10^{-5}))) / 3 \\ &= (10 - (4 + 3 \times (5.0880 \times 10^{-5}))) / 3 \\ &= (10 - (4 + 1.5264 \times 10^{-4})) / 3 \\ &= (10 - 4.0002) / 3 \\ &= 5.9998 / 3 \\ &= 1.9999 \end{aligned}$$

with partial pivoting

pivoting

$$\left[\begin{array}{ccc|c} 3 & 4 & 3 & 10 \\ 1 & 5 & -1 & 7 \\ 6 & 3 & 7 & 15 \end{array}\right]$$

$$① \leftrightarrow ③ \quad \left[\begin{array}{ccc|c} 6 & 3 & 7 & 15 \\ 1 & 5 & -1 & 7 \\ 3 & 4 & 3 & 10 \end{array}\right]$$

$$\begin{array}{l} ③ = ③ - 0.5 \times ① \\ ② = ② - 1.6667 \times 10^{-1} \times ① \end{array} \quad \left[\begin{array}{ccc|c} 6 & 3 & 7 & 15 \\ 0 & 4.5 & -2.1667 & 4.5 \\ 0 & 2.5 & -0.5 & 2.5 \end{array}\right]$$

$$③ = ③ - 5.5556 \times 10^{-1} \times ② \quad \left[\begin{array}{ccc|c} 6 & 3 & 7 & 15 \\ 0 & 4.5 & -2.1667 & 4.5 \\ 0 & 0 & 7.0373 \times 10^{-1} & -2 \times 10^{-5} \end{array}\right]$$

$$x_3 = (-2 \times 10^{-5}) / (7.0373 \times 10^{-1}) = -2.8420 \times 10^{-5}$$

$$\begin{aligned} x_2 &= (4.5 - (-2.1667) \times (-2.8420 \times 10^{-5})) / 4.5 \\ &= (4.5 - 66.1573 \times 10^{-5})) / 4.5 \\ &= 4.4999 / 4.5 \\ &= 1 \end{aligned}$$

$$\begin{aligned} x_3 &= (15 - (3 \times 1 + 7 \times (-2.8420 \times 10^{-5}))) / 6 \\ &= (15 - (3 + 7 \times (-2.8420 \times 10^{-5}))) / 6 \\ &= (15 - (3 + (-1.9894 \times 10^{-4}))) / 6 \\ &= (15 - 2.998) / 6 \\ &= 12.0002 / 6 = 2 \end{aligned}$$

Problems 2

values of ϵ

```
sqrt_eps_machine =

     1.220703125000000e-04

ε            |x_1 - 1|    |x_2 - ε|/ε      cond(A)
1.22e-07     1.62e-03      1.33e+04      2.69e+14
1.22e-06     1.69e-05      1.39e+01      2.68e+12
1.22e-05     9.54e-07      7.81e-02      2.68e+10
1.22e-04     0.00e+00      0.00e+00      2.68e+08
```

conclusion

If $cond(A) \approx 10^k$, then about k decimal digits are lost when solving Ax = b. In this case, because matlab's default precision is 16 digits, when $cond(A) \approx 10^k$ and $\epsilon \approx 10^m$, then the relative error for $x_1 \approx 10^{k-16}$ and the relative error for $x_2 \approx 10^{k-16-m}$.

Problems 3

a

GE

```
function B = GE(A)
    [n, ~] = size(A);
    L = eye(n);
    for k = 1:n - 1
        M = eye(n);
        % for j=k+1:n
        j = k + 1:n;
        % lj,k = A(j, k) ./ A(k, k);
        M(j, k) = - A(j, k) ./ A(k, k);
        M_inv = -tril(M, -1) + eye(n);
        A = M * A;
        L = L * M_inv;
    end
    U = A;
    B = tril(L, -1) + triu(U, 0);
end
```

GEPP

```
function [B, ipivot] = GEPP(A)
    [n, ~] = size(A);
    P = 1:n;
    L = eye(n);
    for k = 1:n - 1
        M = eye(n);
        % pick max row
        [~, max_ind] = max(abs(A(P(k:n),k)));
        % fake swap
        P_col = 1:n;
        P_col(k) = max_ind + k - 1;
        P_col(max_ind + k - 1) = k;
        temp = P(k);
        P(k) = P(max_ind + k - 1);
        P(max_ind + k - 1) = temp;
        % for j=k+1:n
        j = k + 1:n;
```

```
        % lj,k = A(j, k) ./ A(k, k);
        M(j, k) = - A(P(j), k) ./ A(P(k), k);
        A(P(1:n), :) = M * A(P(1:n), :);
        L = L * inv(M(:, P_col(1:n)));
    end
    L = L(P(1:n), :);
    U = A(P(1:n), :);
    B = tril(L, -1) + triu(U, 0);
    ipivot = P;
end
```

backward

```
function x = backward(B, b, ipivot)
    % To solve Ax = b, we write first P*A*x = L*U*x = L*y = P*b.
    % Solve L*y = P*b for y and then U*x = y for x
    [n, ~] = size(B);
    L = tril(B, -1) + eye(size(B));
    U = triu(B, 0);
    y = L\b(ipivot(1:n));
    x = U\y;
end
```

b

```
clear;clc;
n = 2000;
m = 5;
fprintf('        A div b          no pivioting                    pivoting
             cond(A)\n')
for i = 1:m
    A = rand(n, n);
    x = ones(n, 1);
    b = A * x;

    % matlab
    x_matlab = A \ b;

    % GE
    B = GE(A);
    x_GE = backward(B, b, 1:n);

    % GEPP
    [B, ipivot] = GEPP(A);
    x_GEPP = backward(B, b, ipivot);
```

```
    % calculate and plot
    rel_err_matlab = norm(x_matlab - x)/norm(x);
    rel_err_no_p = norm(x_GE - x)/norm(x);
    rel_err_p = norm(x_GEPP - x)/norm(x);
    condA = cond(A);
    fprintf('%d      %.2e        %.2e                    %.2e              %.2e
       \n', ...
        i, rel_err_matlab, rel_err_no_p, rel_err_p, condA);
end
```

c

	A div b	no pivioting	pivoting	cond(A)
1	1.24e-12	6.39e-09	3.28e-12	2.53e+05
2	9.51e-13	4.31e-10	7.54e-13	1.38e+05
3	3.81e-12	8.65e-09	2.13e-12	6.88e+05
4	1.07e-11	8.25e-10	6.25e-12	1.30e+06
5	1.88e-11	1.60e-10	8.57e-12	1.43e+06

fx >>

d

As the condition numbers get larger, relative errors of all three methods get larger. If the condition number $\approx 10^m$ and the relative error $\approx 10^k$, then they have a relation $k - m = 16$, where 16 is matlab's default precision digit number. In other word, If $cond(A) \approx 10^k$, then about k decimal digits are lost when solving Ax = b.

Problems 4

$n = 5$

$h = (b-a)/n = \frac{1}{5}$

$M = \max\limits_{0 \leq t \leq 1} | e^{t} | = e$

Then

$|f(x) - p_n(x)| \leq \frac{M}{4(n+1)} h^{n+1} = \frac{e}{4 \times 6} \times (\frac{1}{5})^6 \approx 7.2488 \times 10^{-8}$

$|f(x) - p_n(x)| \leq \frac{e}{4(n+1)} \cdot (\frac{1}{n})^{n+1}$

we want $|f(x) - p_n(x)| \leq 10^{-8}$

when $n = 7$, $|f(x) - p_n(x)| \approx 1.4735 \times 10^{-8} > 10^{-8}$

when $n = 8$, $|f(x) - p_n(x)| \approx 5.6258 \times 10^{-10} < 10^{-8}$.

Therefore, I will use ~~deg~~ degree $= 8$.

Problems 5

```
clear all; close all;
n = 3;
% degree = 2;
a = 0; b = 0.3;
x = linspace(a,b,n+1);
f = @(x) sqrt(x+1);
p = polyfit(x,f(x), n);

xx = linspace(0, 0.3, 1000);
error = abs(f(xx)- polyval(p,xx));

approx1 = polyval(p,0.05)
error1 = abs(f(0.05) - approx1)
approx2 = polyval(p,0.15)
error2 = abs(f(0.15) - approx2)
```

```
h   = (b-a)/n;
M = 15/16;
ons = ones(length(xx), 1);
error_bound =  M / (4*(n+1)) * h^(n+1)
semilogy(xx, error, xx, error_bound*ons)
legend('error', 'error bound')
```

a

The approximation for $\sqrt{0.05+1}$ is $1.024692660787484e+00$. The approximation for $\sqrt{0.15+1}$ is $1.072381890220326e+00$.

b

$n = 3$

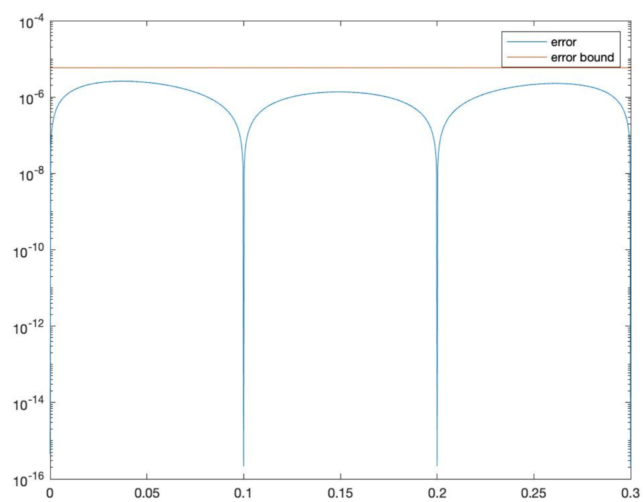
$h = (0.3-0)/3 = 0.1$

$M = \max_{0 \le t \le 0.3} |f^{(4)}(t)| = \max_{0 \le t \le 0.3} \left| -\frac{15}{16}(x+1)^{-\frac{7}{2}} \right| = \frac{15}{16}$

$\text{error\_bound} = \frac{M}{4(n+1)} h^{(n+1)} \approx 5.8594 \times 10^{-6}$

c

As we can see from the figure, the error bound is always larger than the actual error.



Problems 6

Below are four figures for $f(x) = |x|$ on $[-1, 1]$.

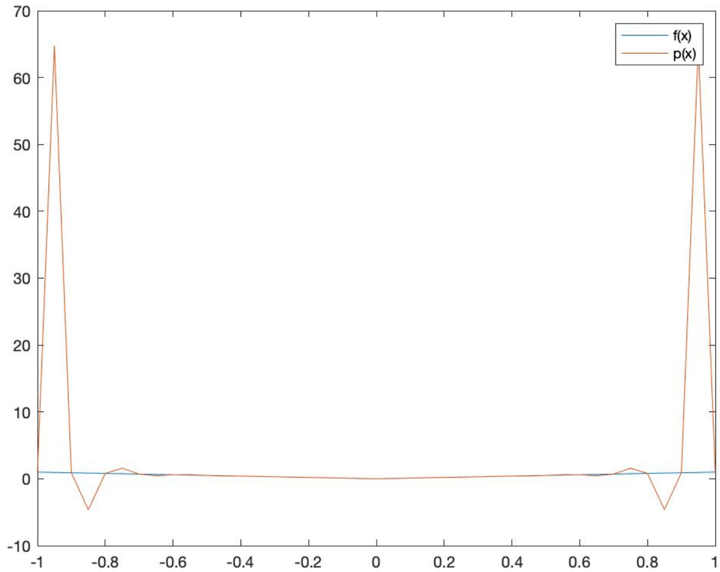


Figure 1: $f(x)$ and $p(x)$ for equally spaced points

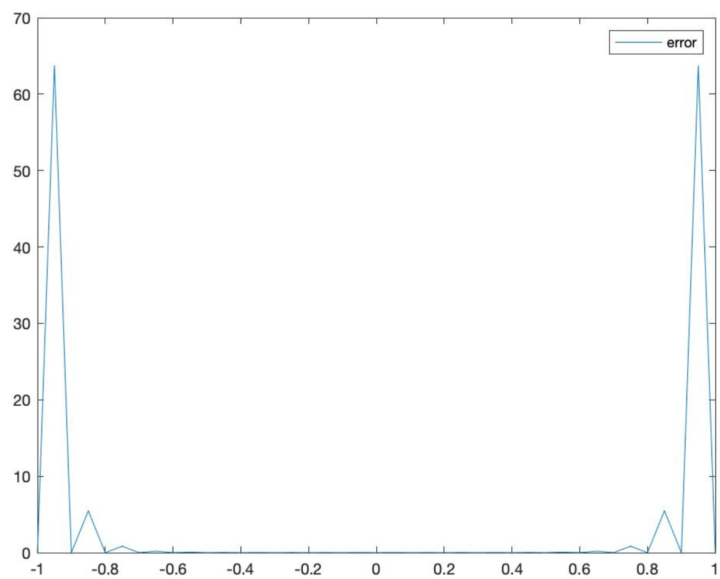


Figure 2: error $|f(x) - p(x)|$ for equally spaced points

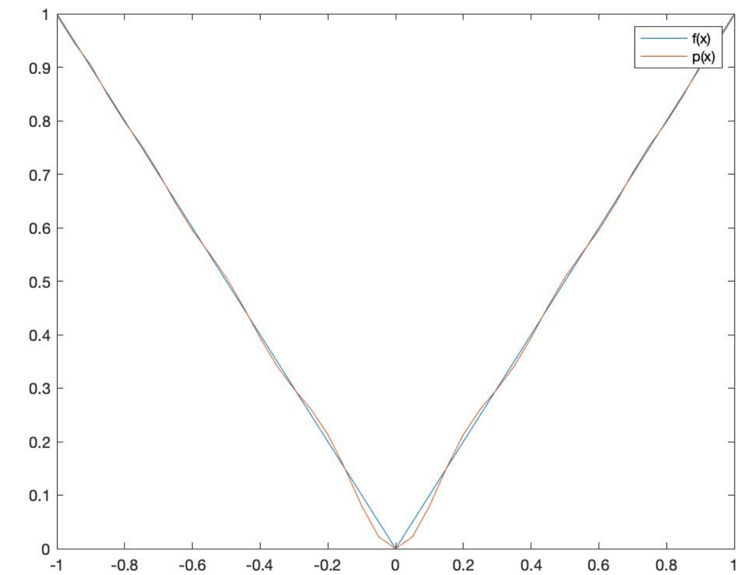


Figure 3: $f(x)$ and $p(x)$ for Chebyshev points

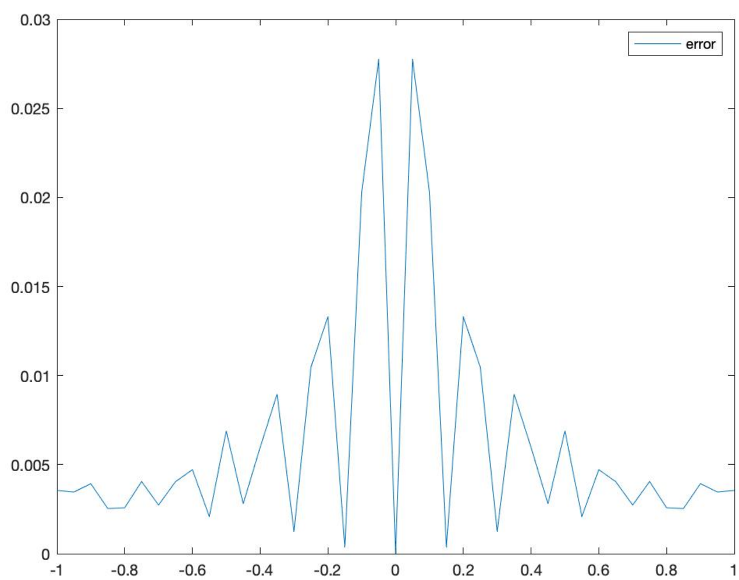


Figure 4: error $|f(x) - p(x)|$ for Chebyshev points

Problems 7

Below are four figures for $f(x) = sin(x)$ on $[-\pi, \pi]$.

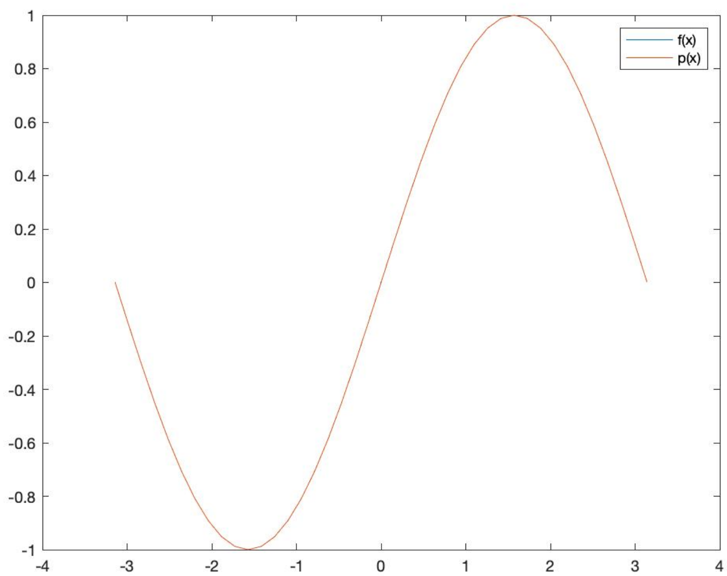


Figure 5: $f(x)$ and $p(x)$ for equally spaced points

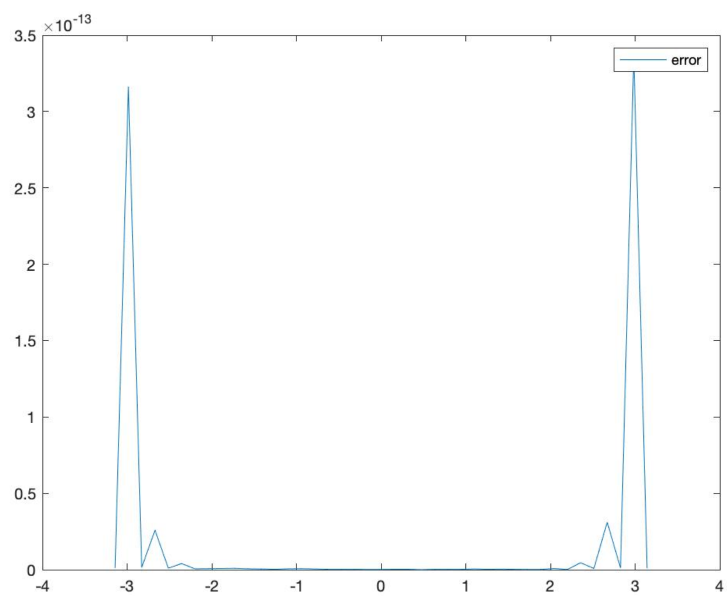


Figure 6: error $|f(x) - p(x)|$ for equally spaced points

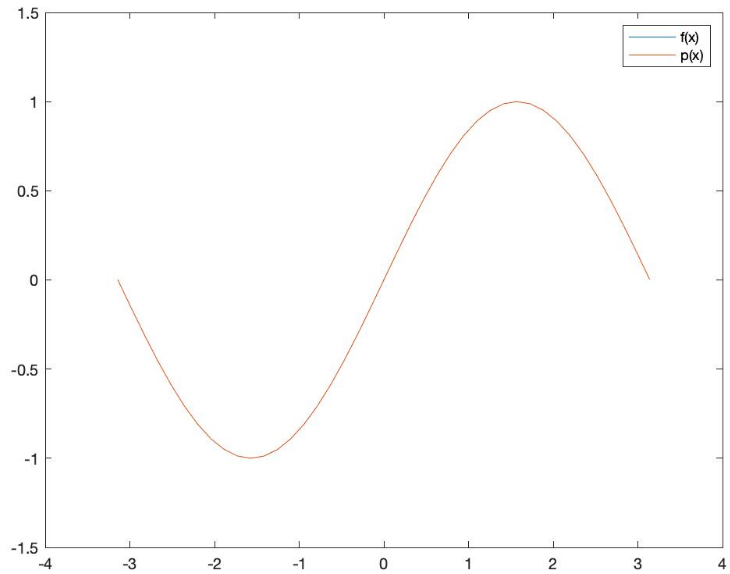


Figure 7: $f(x)$ and $p(x)$ for Chebyshev points

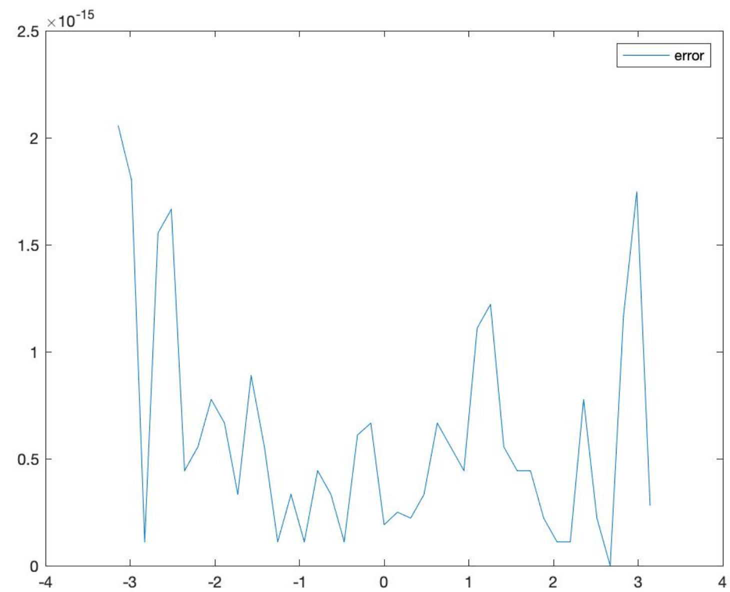


Figure 8: error $|f(x) - p(x)|$ for Chebyshev points

explanation

Compared between the equally spaced points and Chebyshev points, either for $f(x) = sin(x)$ or $f(x) = |x|$, interpolation on Chebyshev points has a lower error bound, which is because of Chebyshev points's Min-max property.
Compared between $f(x) = sin(x)$ and $f(x) = |x|$, either for the equally spaced points or Chebyshev points, $f(x) = sin(x)$ can be easily interpolated, because of its cyclicity.

Problems 8

newton

```
function cs = newton(xs, ys)
    % assume xs and ys are column vectors with the same size.
    % return a column vector that contains all coefficients. e.g. c_0 =
        cs(1)
    [num_of_points, ~] = size(xs);
    n = num_of_points - 1;
    % initialize
    dp = zeros(n+1, n+2);
    for i = 1:n+1
        dp(i, 1) = xs(i);
        dp(i, 2) = ys(i);
    end
    % calculate (1,2), (2,2), (3,2) ... (n+1, 2), (1,3), (2,3)...
    for j = 3:n+2
        for i = j - 1:n+1
            dp(i, j) = (dp(i, j-1) - dp(i-1, j-1)) / (dp(i, 1) - dp(i -
                j + 2, 1));
        end
    end
    res = dp(:, 2:n+2);
    % c_0 = cs(1), c_1 = cs(2), ...
    cs = diag(res);
end
```

hornerN

```
function yx = hornerN(xx, xs, cs)
    % assume input are col vectors
    % xx - the points for interpoation
    % xs - the actual points already have
    % cs - the coefficient sequences return by newton.m
    % yx - return the interpolation vals (col vectors corresponding to
       xx)
    [m, ~] = size(xx);
    [num_of_points, ~] = size(xs);
```

```
    n = num_of_points - 1;
    res = zeros(m, n+1);
    res(:, n+1) = cs(n+1);
    for k = n:-1:1
        res(:,k) = cs(k)*ones(m,1) + (xx - xs(k)*ones(m,1)).*res(:,k+1)
            ;
    end
    yx = res(:,1);
end
```